Intel® Technology Journal

MARCH 2009

Advances in Embedded Systems Technology

INTEL® TECHNOLOGY JOURNAL ADVANCES IN EMBEDDED SYSTEM TECHNOLOGY

Articles

Intel Technology Journal

Publisher
Richard Bowles

Managing Editor
David King

Content Architect
Todd Knibbe

Program Manager
Marleen Lundy

Technical Editor
David Clark

Technical Illustrators
Richard Eberly
Margaret Anderson

Content Design
Peter Barry
Marcie M Ford
Todd Knibbe
Atul Kwatra

Technical and Strategic Reviewers

Steven Adams	Dennis B Fallis	Michael G Millsap
Peter Barry	Al Fazio	Udayan Mukherjee
Mark Brown	Rajesh Gadiyar	Staci Palmer
Tom Brown	Javier Galindo	Michael A Rothman
Jason M Burris	Gunnar Gaubatz	Lindsey A Sech
Lynn A Comp	Byron R Gillespie	Shrikant M Shah
John Cormican	Marc A Goldschmidt	Brian J Skerry
Pete Dice	Knut S Grimsrud	Durgesh Srivastava
Richard Dunphy	Chris D Lucero	Edwin Verplanke
Jerome W Esteban	Lori M Matassa	Chad V Walker

Intel Technology Journal

Copyright © 2009 Intel Corporation. All rights reserved.
ISSN: 1535-864X
ISBN 978-1-934053-21-8
Intel Technology Journal
Volume 13, Issue 1

This book is printed on acid-free paper. ♾

Publisher: Richard Bowles
Managing Editor: David King

Library of Congress Cataloging in Publication Data:

Printed in

10 9 8 7 6 5 4 3 2 1

First printing March 2009

INTEL® TECHNOLOGY JOURNAL ADVANCES IN EMBEDDED SYSTEM TECHNOLOGY

Articles

FOREWORD

Pranav Mehta
Sr. Principal Engineer & CTO
Embedded & Communications Group
Intel Corporation

"An embedded device is a differentiated compute platform that is either invisible, being part of a larger infrastructure, or predetermined to expose limited capabilities in deference to a dominant usage."

"So, what do you mean by an embedded device?" is a question I get asked frequently. Many in academia and industry alike have offered and debated versions of its definition. While many achieve varying degrees of admirable brevity, insight, and accuracy, often these definitions leave an impression that embedded devices are somewhat less capable, outdated technology. Having been associated with Intel's embedded products group for almost two decades, I find such characterizations lacking. Yes, embedded devices have their special requirements, from technical as well as business perspectives, but less capable technology is certainly not one of them. At the risk of inflaming the definition debate, here is my version as an Intel technologist: An embedded device is a differentiated compute platform that is either invisible, being part of a larger infrastructure, or predetermined to expose limited capabilities in deference to a dominant usage. Implicit in this definition are the notion of an embedded device having its unique requirements, its inconspicuous pervasiveness throughout infrastructures supporting modern lifestyle, as well as an allusion to the underlying platform capable of much more than what is exposed in service of a primary set of use functions.

"The Intel multi-core processor architecture and related technologies ensure continuation of the performance treadmill famously articulated by Moore's Law, which is critical for the majority of embedded platforms that constitute the Internet infrastructure as new usage models involving video, voice, and data create an insatiable demand for network throughput and cost efficiency."

This edition of Intel Technology Journal marks the intersection of several major trends and events in the embedded world. As eloquently articulated in the ITU paper Internet of Things, embedded devices appear poised to lead the next wave of evolution of the Internet as they add Internet connectivity as a key platform attribute. Against this backdrop, two groundbreaking technology innovations from Intel—Power efficient Intel® Core™ microarchitecture with an increasing number of cores and the introduction of the Intel® Atom™ processor, both benefitting immensely from the breakthrough High-K/Metal gate process technology—create a unique opportunity to accelerate this embedded transformation with Intel® architecture. The Intel multi-core processor architecture and related technologies ensure continuation of the performance treadmill famously articulated by Moore's Law, which is critical for the majority of embedded platforms that constitute the Internet infrastructure as new usage models involving video, voice, and data create an insatiable demand for network throughput and cost efficiency. On the other hand, the Intel Atom processor opens up possibilities for a completely new class of ultra low power and highly integrated System-on-a-Chip (SoC) devices with Intel architecture performance that were unimaginable before.

Over the last several years, Intel's Embedded and Communications Group has introduced several products that achieve the best-in-class "power efficient performance" and push the boundaries of integration for SoC devices. We have done that while preserving the fundamental premise of Intel architecture—software scalability. Now, equipped with these new technologies and product capabilities, we are delighted to have the opportunity to accelerate the phenomenon of the embedded Internet.

While I am proud to offer technical articles from members of Intel ECG's technical team, I am equally proud to offer articles from developers who have embraced our embedded systems platforms and put them to use. Finally, I look forward to revisiting embedded systems technology in a few years' time. I believe that we will witness enormous progress over the years to come.

"Finally, I look forward to revisiting embedded systems technology in a few years' time. I believe that we will witness enormous progress over the years to come."

PERFORMANCE ANALYSIS OF THE INTEL® SYSTEM CONTROLLER HUB (INTEL® SCH) US15W

Contributor

Scott Foley
Intel Corporation

Index Words

Intel® System Controller Hub US15W
Intel® Atom™ processor
I/O performance analysis
optimization

"The article provides an overview of the different use cases and their effects on I/O performance, including memory bandwidth utilization from video decoding applications."

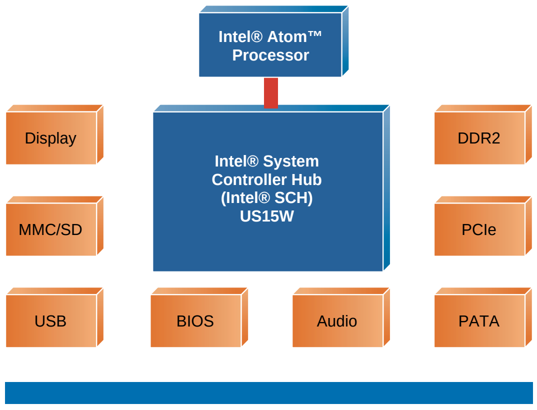


Figure 1: Intel® Atom™ processor-based platform.

Abstract

The platform comprising the Intel® Atom™ processor and Intel® System Controller Hub (Intel® SCH) US15W has recently been introduced into the embedded systems marketplace, for a wide range of uses. And with this pairing, Intel now has an incredibly low power (<5W maximum thermal design power [TDP]) platform built on Intel® architecture. This compares favorably to other platforms, such as ones based on Intel® Core™2 Solo processors, which have processor TDP's alone of 5.5W. Given the wide range of applications this new platform encompasses though, some design tradeoffs are inevitable. Performance is a notable tradeoff, and the performance of the Intel SCH is a good example to look at.

Instead of looking into processing performance, this article takes a systematic look at the architecture of the Intel SCH US15W. It investigates the performance of the Intel SCH backbone and the effects that that backbone has on PCI Express* (PCIe*) bandwidth in particular. This includes not just empirical data, but also theoretical analysis of the Intel SCH internals. Furthermore, the article provides an overview of the different use cases and their effects on I/O performance, including memory bandwidth utilization from video decoding applications. With this knowledge in hand, platform, hardware, and software engineers can better design systems that work efficiently with the Intel Atom processor's low power offerings.

Intel® System Controller Hub (Intel® SCH) US15W Architecture

The Intel® System Controller Hub (Intel® SCH) US15W is the chipset used in small, low power Intel® Atom™ processor-based platform designs. It connects the Intel Atom processor and various I/O devices to system memory. At the heart of the Intel SCH and any platform is the memory controller. All of the various peripherals are vying for its attention, as can be seen in Figure 1. The processor's Front Side Bus (FSB) is a constant consumer of memory bandwidth, as is the integrated graphics controller. These receive priority in the Intel SCH. Then comes everything else: USB, PATA, SDIO, PCIe, Audio, Legacy, and so on. Most of these devices have sporadic memory bandwidth needs. The possible exception would be PCI Express* (PCIe*), which we will discuss later. Before we understand how everything is connected though, we need to further understand the design goals of the Intel SCH.

Tradeoffs

Power, performance, size, features—these are just some of the things designers have to balance when designing a chipset like the Intel SCH US15W. Each of these metrics interoperates with the others, and must be carefully considered. With the Intel SCH, the balance has shifted significantly towards power optimization, with power to performance efficiency being the utmost importance.

"With the Intel SCH, the balance has shifted significantly towards power optimization, with power to performance efficiency being the utmost importance."

Media applications, in particular, are given more of the power budget in this platform. Thus, graphics are given more bandwidth and higher priority to access memory. The processor, via the FSB, is also given more of the memory controller's time for running applications and/or performing software-based DSP functions and the like. The processor is the general purpose controller, and is expected to do most of the heavy lifting. Finally, the I/O devices were considered to typically require low to medium amounts of bandwidth and, as we will see, that is what they get.

Another requirement was to fit a large number of I/O types (features) in and still keep power low. To make a balance between features and power, tradeoffs must be made compared to more traditional Intel® architecture designs. The I/O devices' connection to the memory controller reflects as much. As this article will show, all the I/O devices on the Intel SCH run on two equally shared unidirectional busses called the backbone, which has less access to memory bandwidth than either the integrated graphics controller or the FSB. How much bandwidth? Each bus of the backbone has a little more than the original PCI bus could handle. But of course PCIe allows for higher bandwidths than that, right? To know for sure, let us take a look at how PCI Express* works.

"To make a balance between features and power, tradeoffs must be made compared to more traditional Intel® architecture designs."

PCI Express* (PCIe*) Details

PCI Express, or PCIe, is a serialized/packetized implementation of the Peripheral Component Interconnect (PCI). While they share the same name and software concepts, the implementation of the two couldn't be more different. PCI is parallel while PCIe is serial; PCI has a shared bus while PCIe employs a switched point-to-point topology.

At a high level, PCIe looks a lot like a typical network stack. It has many of the same or similar layers as found in the OSI model for networked communications. Lower layers consist of physical transmission—the electrical/signaling aspects—and eventually progresses all the way up to transaction-based packets.

"PCIe looks a lot like a typical network stack. It has many of the same or similar layers."

The transaction layer is responsible for all of the read/write transactions, or packets, implicit in data communication. And, among many other things, it also contains concepts for tracking buffer space. Technically this is done at a lower layer via Flow Control (FC) credits, but the concepts are very similar. These credits allow end points to know when to transmit data, and how much data they can safely transmit. They do this by updating each other on how much buffer space is available to the receiver. This eliminates wasted time and bandwidth by sending packets that cannot be accepted, and allows end points to speak to each other with a high degree of confidence.

"Lanes are bidirectional serial interconnects (physically, two differential pairs) of a specific bandwidth."

As we move further down the PCIe protocol stack, we see how the end points are actually connected. PCIe uses the concepts of links and lanes.

Lanes are bidirectional serial interconnects (physically, two differential pairs) of a specific bandwidth. For the first generation of PCIe, as in the Intel SCH US15W, this bandwidth is 2.5 Gbps in each direction of the lane thanks to a 2.5-GHz signaling rate. Since lanes use 8/10-bit encoding, a single 8-bit byte requires 10 encoded bits of transmission bandwidth. This makes our effective bandwidth calculation easy; dividing by ten we end up with 250 MBps of bandwidth in each direction, for any given lane.

"Data is striped among all the possible lanes, much like with RAID-0 disk arrays or memory accesses to DIMMs with many DRAM devices."

Links are built of one or more lanes. A link may have four lanes, eight lanes, or even sixteen. Data is striped among all the possible lanes, much like with RAID-0 disk arrays or memory accesses to DIMMs with many DRAM devices. This way, hardware PCIe devices can scale their bandwidth accordingly. Links on the Intel SCH US15W only consist of one lane, and there are two links. Therefore we say both links are x1 (by-one).

While each PCIe lane is capable of 250-MBps data rates in each direction, some of the data transmitted consists of control and other useful information. Like the transaction layer's overhead, we can't expect even theoretical applications to hit 250-MBps data throughputs on a link. This additional overhead can be taken into account of course. Because the maximum payload size on the Intel SCH US15W is 128 bytes and the header and other packet/framing overhead typically consumes 20 "bytes" (STP=1, sequence=2, header=12, LCRC=4, END=1) on a x1 link, our maximum posted transaction bandwidth would actually be closer to 216 MBps in each direction in this absolutely best-case scenario.

PCI Express* (PCIe*) Performance Measurements

Performance data collection and analysis on the Intel SCH US15W was performed using a combination of custom PCIe traffic generators and off-the-shelf PCIe logic analyzers. The traffic generators are highly configurable DMA engines, with a collection of custom software to create various traffic patterns and execute them. All tools are running on Linux. While these tools are capable of some data collection, data was validated using the logic analyzer. The test setup is shown in Figure 2.

As we can see, Figure 2 shows two traffic generators each connected to the Intel SCH via a x1 PCIe link; between are logic analyzers. In actual testing, only one analyzer was used. Initial test runs on both ports showed only one analyzer was needed for the tests presented in this article.

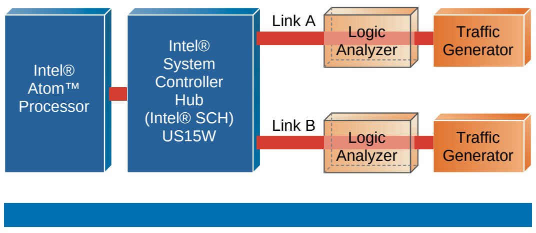


Figure 2: Test setup.

This article will call transactions moving data from a traffic generator to the Intel SCH ***writes*** to memory. They are also called ***upstream*** transactions. ***Reads*** from memory are initiated by the traffic generator and are completed by moving data from the Intel SCH's memory to the traffic generator. These completions are also called ***downstream*** transactions.

Test patterns were generated using random physical memory addresses above a certain limit. That limit is also passed to the kernel at boot time to prevent conflicts. This eliminates the operating system overhead having much of an affect on our measurements. Four different test patterns were used for these tests: two write patterns and two read patterns. One write pattern generates 64-byte writes, while the other generates 128-byte writes. Similarly, one of the read patterns generates 64-byte read requests, while the other generates 128-byte requests.

These test patterns are then used in various combinations to generate a test series. A test series consists of patterns from the same packet size, but enumerates all possible states for each port. Therre are three states: reading (RD), writing (WR), and sitting idle (N/A). After enumerating all possible state combinations, duplicates and the double idle test case are ignored. This leaves five tests per series.

"A test series consists of patterns from the same packet size, but enumerates all possible states for each port."

Lastly, two different systems were tested, each with production stepping engineering samples. One system with a Z530 processor running at 1.6 GHz, and the other with a Z510 processor running at 1.1GHz. Both use the Intel SCH US15W as the chipset, but with appropriate memory speeds for their respective processor.

PCI Express* (PCIe*) Performance of 1.1-GHz Intel® Atom™ Processor-based Platform

The first test series collected on a 1.1GHz Intel Atom processor and the Intel SCH US15W for 128-byte packets is shown in Figure 3.

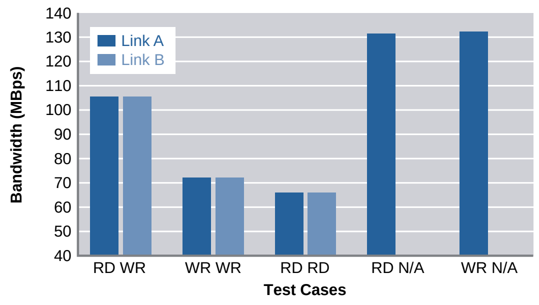


Figure 3: Basic read and write results with a 128-byte payload.

Looking at this series running on the Intel SCH US15W, we see some interesting traits straight away. First of all, as expected, the PCIe link doesn't reach the full 216-MBps PCIe line rates we calculated earlier. This system reaches no more than about 133 MBps for reads and writes individually on a single link. When utilizing both links for reads we see a combined throughput of about 133 MBps as well, also evenly split. Yet with both links performing simultaneous writes we see 145 MBps, but again, evenly split. Finally, when performing reads on one link and writes on the other, the combined bandwidth is 212 MBps split evenly between the two links. These consistently even splits are clear evidence of round robin arbitration. Next let us look at the 64-byte packet series, shown in Figure 4.

This series does not just look about the same. For all practical purposes, it is the same. This is good, as it reinforces our assumptions thus far. Let us review the second platform's results before digging in any deeper.

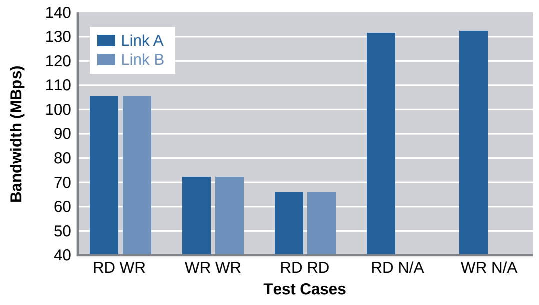


Figure 4: Basic read and write results with a 64-byte payload.

"On the 1.6-Ghz platform we see noticeable bandwidth increases over the 1.1-Ghz platform."

PCI Express* (PCIe*) Performance of 1.6-GHz Intel® Atom™ Processor-based Platform

Now let us try running the two test series on a 1.6-GHz Intel Atom processor with the same Intel SCH US15W. Figure 5 shows the results for a 64-byte test series and Figure 6 shows the results for a 128-byte test series.

On the 1.6-GHz platform we see noticeable bandwidth increases over the 1.1-GHz platform. In fact, it increases proportional to the FSB speed. On the 1.1-GHz platform, the FSB was 400 MT/s, and on the 1.6-Ghz platform it is 533 MT/s. For throughputs, the two-link read test case (RD RD) measures in at 177 MBps combined bandwidth. On the 400 MT/s FSB we saw 133 MBps for the same test case. Setting up the equality between FSB ratios and PCIe throughput ratios matches up closely. We'll see why this is later.

There's one other major observation. There's an obvious delta between the single link read and single link write cases. With the 400 MT/s FSB part we saw equivalent bandwidths for the equivalent cases, but with the higher speed FSB we see that neither case matches. In fact they practically alternate between the two packet sizes. Read throughputs at 64 bytes equal those of 128-byte write case and vice versa. We'll look into why that may be later.

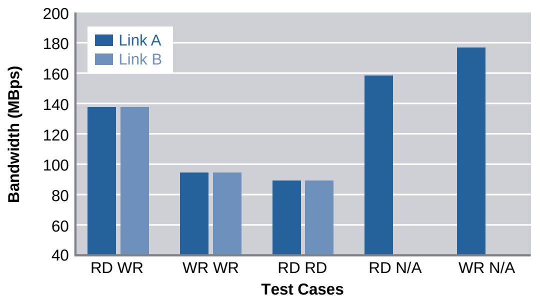


Figure 5: 64-byte test series on 1.6-GHz platform.

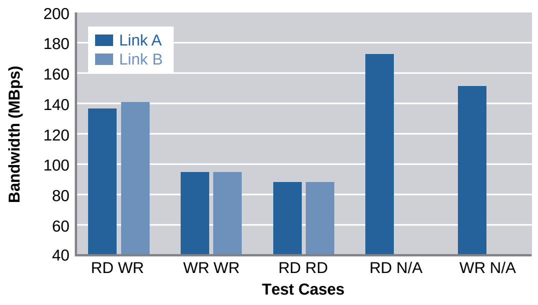


Figure 6: 128-byte test series on 1.6-GHz platform.

Intel® System Controller Hub (Intel® SCH) US15W Backbone

We now have enough data to start to make detailed inferences as to the structure of the Intel SCH US15W's backbone, shown in Figure 7, by analyzing the 1.1-GHz platform. We'll start with the combined bandwidth of each test case for a series as shown in Figure 8. This data, the higher bandwidth Read/Write (RD WR) case in particular, is a good indicator that the internal bus consists of two independent buses, one for upstream and another for downstream.

Now, ignoring the RD WR test case, each direction (upstream and downstream) appears to have equal bandwidth: 133 MBps. If this is the case, that each bus has equal bandwidth, why is the read/write (RD WR) case not double the other cases? That is because the dual independent bus backbone comprises two simple parallel buses, each conceptually similar to PCI. Each of those busses share address and data on the same wires (so it is at least a 32-bit bus). Therefore, read requests take up additional bandwidth on this bus, limiting the write requests. Since we would expect 133 MBps for each stream and we only see about 106 MBps, that additional bandwidth for the requests works out to roughly 27 MBps of useable bandwidth. We'll touch on this later.

Another oddity when looking at the combined bandwidth chart is the slightly higher bandwidth of the dual-link writes (WR WR) case: 145 MBps versus 133 MBps for all others. This could be an artifact of the arbitration scheme used, or, possibly, a flow control buffer effect. Assuming each link has buffers that get filled quickly and only empties when a full transaction has completed internally over the backbone, ping ponging between the two (doubling the buffers) should allow each end point to feed data to the Intel SCH US15W at the highest rate the backbone supports: 145 MBps.

Backbone Clock Frequency

Analyzing the bus further, we should be able to determine the clock rate of the backbone. This may provide us with clues to the width and efficiency of the backbone, in addition to understanding its clock rate.

First of all, the clock rate is 25 MHz for the 1.1-GHz platform. While this article will not go into the full details, one way to determine this frequency is by adding delays via the traffic generator and sweeping across a number of different delays. By analyzing the resulting traffic captures, one can see the latencies, on average, lining up into specific quanta or averages. Measuring the delta in delays between the different quantum levels provides a good guess as to the period of the internal clock—in the case of the platform with the 1.1-GHz Intel Atom processor, approximately 40 nanoseconds, corresponding to the 25-MHz clock. This analysis is not always feasible of course, but it is one possible approach. Repetitive patterns in the trace captures also reveal this periodicity or multiples of it.

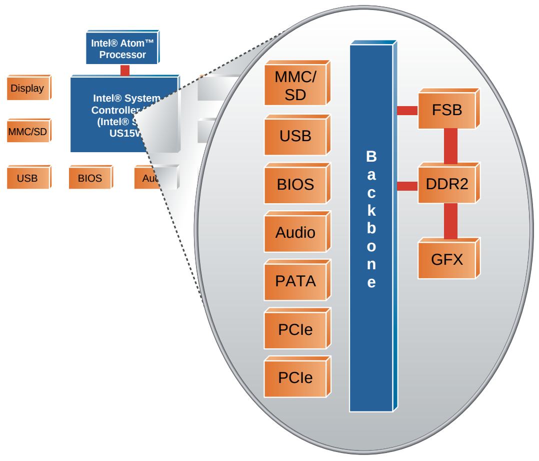


Figure 7: Intel® System Controller Hub (Intel® SCH) US15W backbone architecture

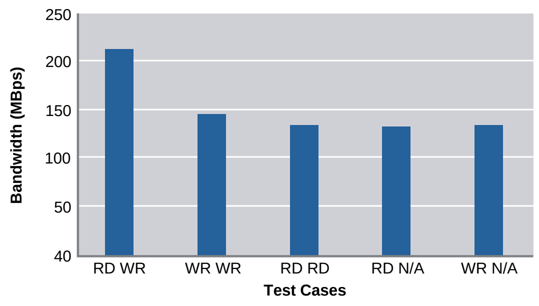


Figure 8: 64-byte transactions on 1.1-GHz platform.

"A slower clock is good for reducing design and validation time among other things."

This begs the question—why the relatively slow 25-MHz clock? The answer is that a slower signaling rate reduces power significantly (P~kfV^2 after all). This is one of those tradeoffs we mentioned earlier. It also minimizes high frequency noise and other negative signal integrity effects. Furthermore, a slower clock is good for reducing design and validation time among other things. But power appears to be the main reason.

Backbone Bus Width and Burst Length

Now that we know the clock frequency we can predict the backbone's data bus width. This one was determined by a process of elimination and a bit of common sense. At 64 bits, the theoretical maximum bandwidth is 200 MBps for a bus running at 25 MHz. Higher or lower than that, but by powers of two, and you either get bandwidths that are too low or too high. At 128 bits, for example, the theoretical bandwidth doubles to 400 MBps. That would mean the efficiency of the backbone is approximately 36 percent given the 145 MBps maximum we were able to achieve empirically. This efficiency is simply far too low to be practical. And at 32 bits, we are left with a maximum bandwidth lower than we observed. Thus, 64 bits it is.

Next, we can guess a transaction's burst size. Since we know the backbone is a parallel bus, similar to PCI, transactions can vary in size. For each address phase, we want to determine the number of data phases. Assuming a 64-byte transaction takes as long to setup as a 128-byte transaction (address/control/turnaround phases), it seems obvious from the data collected that 64-byte bursts are the only logical option. If they were larger than that, we should have seen an increase in bandwidth at the 128-byte write packet sizes where efficiency increases. Note that this means running smaller than 64-byte packet sizes should show lower bandwidth.

Further evidence of 64-byte bursts are also seen when analyzing read transactions. When making 128-byte or larger requests, multiple 64-byte completions are always returned. The hardware simply completes the transactions on the PCIe side as soon as the internal 64-byte transaction (a full cache line) is completed.

Backbone Model

Knowing the backbone burst transaction size can help us understand efficiency, but we already know that. What it really helps us do is map out the backbone. We can create a chart of all the possible bandwidths that are feasible when PCIe is bursting. As a confirmation, this data should match up with data we have collected. With this affirmation, the additional insight gained will allow us to make better predictions of bandwidths at other transaction sizes, or when other devices are accessing the backbone. Figure 9 shows what a model of this looks like visually.

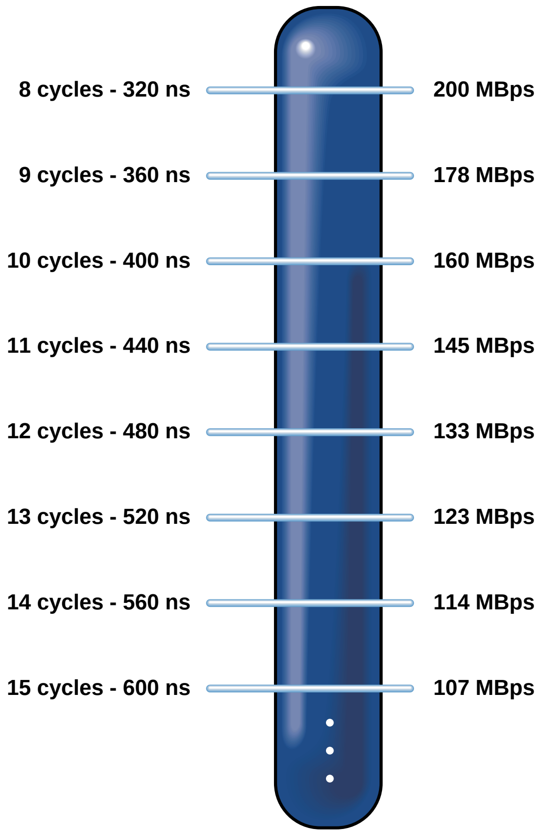


Figure 9: Possible backbone bandwidths based on 64-byte transfers over a 64-bit bus clocked at 25 MHz (1.1-GHz platform).

In Figure 9 we see a list of times measured in clock cycles and equivalent time in nanoseconds (assuming a 40-ns period). For each time we list the instantaneous bandwidth if transmitting 64 bytes given by the equation: 64 bytes/time.

Many of the bandwidths we have measured on the 1.1-GHz Intel Atom processor with the 400-MT/s FSB part show up in this graph. This graph also tells us additional information about what is happening in the Intel SCH. If we look back at the combined bandwidth analysis we did for Figure 8, we saw read requests requiring overhead of 27 MBps for the (RD WR) test. What we were really measuring was the number of clock cycles taken by a read request. In Figure 9, 106 MBps is about 15 cycles and 133 MBps is about 12 cycles. Thus we see a read request, on average, takes three backbone clock cycles to transmit. With this kind of knowledge we can start breaking down each transaction type to make better predictions of more complex test cases.

"With this kind of knowledge we can start breaking down each transaction type to make better predictions of more complex test cases."

So what about the 1.6-GHz platform with the 533-MT/s FSB? We noticed earlier that the FSB frequency changes created equally proportional changes in PCIe throughputs. Basically the backbone operates at one quarter the FSB BCLK rate, which is itself one quarter the FSB data transfer rate. As such, similar analysis and charts can be created using a 30-ns period of the 33.3-MHz clock in this platform instead of the 40-ns period of the 25-MHz clock in the 1.1-GHz platform. Digging deeper into test results for the 1.6-GHz platform will show more instability compared to the predicted model. It is believed this is due to clock boundary crossing of the 30-ns backbone with the 4-ns based transfer rate of PCIe, but no additional effort was taken to verify this.

Further, this model does not predict the change in bandwidth between the RD N/A and WR N/A cases of the 1.6-GHz platform. Typically we would expect throughputs to increase or remain the same as transaction sizes increase. With the writes, we see the opposite behavior. This might be best explained as an effect caused by full flow control buffers. Once these buffers fill up, transactions are allowed only when enough buffer space for a new transaction is free. The buffers should be sized to a multiple of the cache line size (64 bytes), with at least enough buffer to handle the maximum payload size (128 bytes). When write transactions are performed though, it is likely that the transaction size determines when buffers are free. That is, if 128-byte PCIe transactions have been posted to the buffers, the backbone performs two 64-byte internal transactions before buffer space is freed, the flow control update is sent, and another PCIe transaction is permitted. In the 64-byte case, a form of pipelining can take place. Here, when one internal transaction has completed, an update can be sent and another packet generated with a relatively short latency. At 33.3 MHz, the backbone can complete 64-byte transactions about as fast as the PCIe link can generate them (not so for the 25-MHz backbone). With 128-byte transactions though, the PCIe device must wait. This time spent waiting on the two backbone transactions to complete offsets the gains brought by the improved efficiency on the PCIe link.

"Basically the backbone operates at one quarter the FSB BCLK rate, which is itself one quarter the FSB data transfer rate."

Intel® System Controller Hub (Intel® SCH) US15W Under Pressure

Because the backbone is a shared bus, multiple simultaneous accesses have significant impacts to instantaneous bandwidth (also known as latency). This means that for high bandwidth applications, any other traffic on the bus will diminish the bandwidth temporarily, but by a significant amount. Disk accesses, for example, may not change the overall bandwidth that much on average, but it will halve the available bandwidth for short durations/bursts.

"The buffers should be sized to a multiple of the cache line size (64 bytes), with at least enough buffer to handle the maximum payload size (128 bytes)."

"Most transactions by I/O devices occur in bursts."

Why? Most transactions by I/O devices occur in bursts. As such, even "low" bandwidth applications will request the full bandwidth of the bus for short times. For these short periods, "low" and "high" bandwidth requirements are indeterminable. This means that average bandwidth does not necessarily guarantee a consistent bandwidth. If three devices all require the full bandwidth of the bus at the same time, each is only going to get one third of it for that time given round-robin arbitration. But this is not the only way the bandwidth may be impacted.

Even when the backbone is completely free for a given I/O device, bandwidth can be robbed at the memory controller level, as, for example, with video decoding, as shown in Figure 10.

Here we see our PCIe traffic generator writing data to memory. When decoding high definition video using the Intel SCH US15W's hardware decoder, we see regular dips in PCIe traffic (highlighted in Figure 10). These blips occur at regular intervals that align to each frame in the video being decoded (approximately 41 ms for 24P video). After eliminating all possible I/O impacts, it was confirmed that memory bandwidth was the culprit.

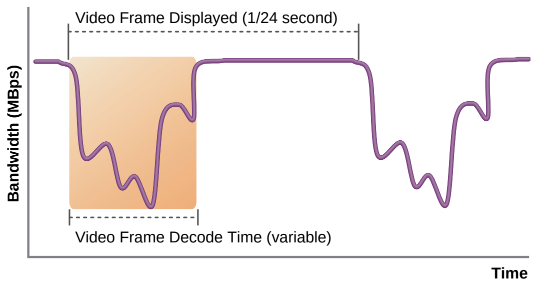


Figure 10: 720P video decode impacts PCIe* traffic—each frame decoded corresponds to one of the dips as highlighted.

Decoding high definition video is fairly memory intensive it turns out. Each uncompressed frame of video is bounced to and from memory several times. Adding additional CPU accesses for memory on top of this will only exacerbate things.

Applying Architecture to Optimization

Putting together all of the architecture and performance information we have uncovered so far, we can now see what optimization is possible, starting with a recap of the architecture.

The Intel SCH US15W chipset employs a shared backbone architecture, not unlike PCI. Rather than a single shared bus as in PCI though, the Intel SCH US15W's backbone consists of two independent 64-bit buses, each running at 1/16 the system's front side bus transfer rate (25 MT/s or 33.3 MT/s). One bus is used for writes and read requests to memory, while the other is dedicated to read completions.

Transactions on the backbone's bus take, at best, three backbone clock cycles for the address, request, and arbitration phases, and allow for a maximum payload size of 64 bytes. Given the 64-bit width of the bus, 64 bytes of data adds an additional 8 clock cycles to the transaction. Thus, 64-byte data transactions take at least 11 clock cycles to complete on the backbone for a maximum efficiency that is almost 73 percent.

"Thus, 64-byte data transactions take at least 11 clock cycles to complete on the backbone for a maximum efficiency that is almost 73 percent."

How can this architectural understanding be used for optimizations? Because of the shared parallel bus nature of the backbone, PCIe hardware running on this platform performs better when conforming to certain parallel bus (think PCI) concepts. For example, PCIe devices on the Intel SCH cannot expect constant bandwidth at all times since they must share it equally with other peripherals. As such, PCIe devices that can vary their requirements (throttle) to maintain an average throughput will work very well. Devices that do not throttle and instead enforce overly rigid bandwidth requirements with inadequate buffers will suffer. Devices must also be able to operate under the maximum throughputs uncovered earlier.

"PCIe devices that can vary their requirements (throttle) to maintain an average throughput will work very well."

Beyond hardware, drivers may be optimized much like they would be on a shared PCI bus. Driver tweaks can further improve the ability of the hardware to adapt to differing bandwidth conditions. Application software also plays a role here because applications can be created to minimize concurrent activities on the bus.

PCIe device performance can also be optimized with packet size changes and memory use. Read requests should be made 128 bytes or larger for maximum throughput. Writes should be 64 bytes for optimal performance, even though 128 bytes would historically be better. Because of odd differences like this, hardware flexibility is ideal to maintain high performance on a wide range of platforms.

Last, but not least, even memory organization can affect performance. Though not discussed in this article, aligning transactions to cache line boundaries is very important, and this applies to all Intel architecture platforms.

"Read requests should be made 128 bytes or larger for maximum throughput. Writes should be 64 bytes for optimal performance, even though 128 bytes would historically be better."

Conclusion

All of the discussed topics—architecture, performance, optimization—are vital aspects to consider when developing platforms with the Intel Atom processor and the Intel SCH US15W. As we work to develop high performance systems with small power budgets, closer attention needs to be paid to a full system understanding. All aspects of a platform impact each other, and that becomes very clear when we push opposing design goals to their limits with new technologies. This analysis and uncovering of the backbone architecture hopefully imparts a better understanding of the platform such that software and drivers can be written efficiently, hardware can be designed efficiently, and the platform as a whole can be used as efficiently and effectively as possible.

"As we work to develop high performance systems with small power budgets, closer attention needs to be paid to a full system understanding."

Author Biography

Scott Foley: Scott Foley is a technical marketing engineer (TME) in Intel's Low power Embedded Division (LEPD). He has been working on platforms based on the Intel® Atom™ processor for a little over a year, and has been at Intel for over three years. He has lectured inside and outside of Intel about various topics, including most recently a lecture at the Univeristy of Colorado at Boulder on the microarchitecture of the Intel® Atom™ processor. Before being a TME he spent time working in an embedded performance architecture group and helped in the development of tools used for this article. He can be reached via email at scott.n.foley at intel.com. Spreadsheet versions of the data presented may be requested at this address.

Copyright

CONFIGURING AND TUNING FOR PERFORMANCE ON INTEL® 5100 MEMORY CONTROLLER HUB CHIPSET BASED PLATFORMS

Contributor

Perry Taylor
Intel Corporation

Index Words

Intel® 5100 Memory Controller Hub chipset
Intel® 5100 MCH
performance
tuning
memory bandwidth
single channel
dual channel
dual-rank
single-rank
single socket
dual socket
quad-core
dual-core
DDR2-533
DDR2-667
I/O device placement
platform tuning
thermal throttle
electrical throttle
global throttle
processor prefetching
hardware prefetch
second sector prefetch
adjacent sector prefetch
BIOS settings

Abstract

A system architect must make platform configuration choices based on multiple architecture tradeoffs between cost, power, and performance. Understanding the end impact on performance that these configuration decisions have is critical in designing competitive solutions around the Intel® 5100 Memory Controller Hub chipset (Intel® 5100 MCH chipset).

This article presents performance-related architecture topics for the Intel 5100 MCH chipset to assist system architects in designing a high performance solution. It will help engineers and architects make decisions with an awareness of performance implications, such as CPU population, memory configuration, and I/O device placement. This article also addresses performance tuning options for the Intel 5100 MCH chipset that can help increase performance for specific usage models.

Introduction

Intel® 5100 Memory Controller Hub chipset (Intel® 5100 MCH chipset) solutions provide board designers with a "blank slate," allowing flexible design and layout of memory, front side bus, generation 1 PCI Express* capability and legacy I/O. While this design flexibility is desirable, it also allows for less than optimal performance configurations. Likewise, even with a board laid out for optimal performance, designers or end users can cripple system performance by populating the platform with lower performing processors, low throughput memory configurations, and poorly placed I/O endpoints. Not only the hardware selection but also hardware placement is important to system performance. Memory placement and I/O placement can have significant impact on performance as we will later demonstrate. Once all hardware choices have been made, some additional performance may be possible with platform tuning for specific usage models.

This article attempts to help with these three areas related to performance:

- Intel® 5100 MCH configuration
- Hardware configuration
- Platform tuning

The information provided is intended to help designers and end users make performance aware decisions in regards to these three areas, allowing them to balance cost, power and thermals with performance needs on the Intel 5100 MCH chipset.

"The information provided is intended to help designers and end users make performance aware decisions in regards to these three areas, allowing them to balance cost, power and thermals with performance needs on the Intel 5100 MCH chipset."

Performance Test Configuration

Performance data presented in this article is collected on the following system configuration:

- Williamsport Customer Reference Board (revision B)
- Dual Socket Intel® Xeon® processor E5410 2.33GHz (Quad-Core)
- Dual Socket Intel® Xeon® processor E5220 2.33GHz (Dual-Core)
- 2 memory channels, 2 DIMMs per channel, 4GB of system memory
- 4x1GB DDR2-667, dual rank, CL5
- BIOS version WSPTG015
- 32 Bit Linux* (Cent OS 4.4)

Intel® 5100 Memory Controller Hub Chipset Feature and Technology Overview

The Intel 5100 Memory Controller Hub (Intel 5100 MCH) chipset) is a low power memory controller hub designed specifically for embedded applications. The Intel 5100 MCH chipset is derived from the Intel® 5000P Memory Controller Hub chipset (Intel® 5000P MCH chipset), a high performance server class chipset and as such, the Intel 5100 MCH chipset has many of the same features, technologies, and performance capabilities as the Intel 5000 Memory Controller Hub chipset (Intel® 5000 MCH chipset). The primary difference with the Intel 5100 MCH chipset as compared to the Intel® 5000 MCH chipset is the removal of the fully buffered DIMMs (FBDs) that were replaced with a native DDR2 controller. This architecture change reduces the total cost of platform ownership by reducing the overall platform power consumption while still delivering high performance.

Figure 1 details the platform block diagram.

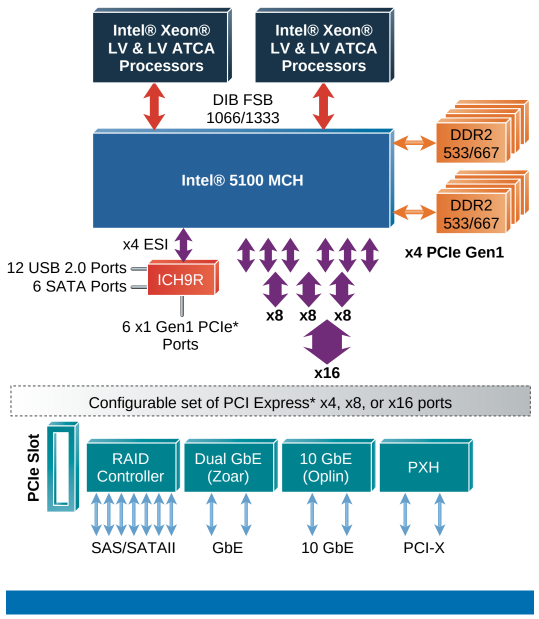


Figure 1: Intel® 5100 Memory Controller Hub chipset. Source: Intel Corporation, 2008

Like the Intel 5000 MCH chipset, the Intel 5100 MCH chipset is designed with a dual independent front side bus (FSB) for improved bandwidth and efficiency over previous generations, supporting 667, 1066, and 1333 MT/s. There are six x4 (pronounced "by four") Generation 1 PCI Express GB links available for direct connect I/O and a x4 direct media interface (DMI) link available to interface with an I/O Controller Hub (ICH). The six x4 PCI Express links can be combined to form various combinations of x8 links and/or x16.

The Intel 5100 MCH chipset is designed with a native DDR2 memory controller supporting registered ECC DDR2533 and DDR2-667. Dual independent memory channels provide improved bandwidth and efficiency supporting up to 3 DIMM modules per channel. Intel 5100 MCH chipset supports singe rank, dual rank, and quad rank DIMMs up to a maximum of 6 ranks per channel and a total capacity of 48 GB.

For more information on the Intel 5100 MCH chipset features, see: *http://www.intel.com/Assets/PDF/datasheet/318378.pdf* [2]

"Intel 5100 MCH chipset supports singe rank, dual rank, and quad rank DIMMs up to a maximum of 6 ranks per channel and a total capacity of 48 GB."

"Platform hardware and software tuning can help extract the last bit of performance out of a system, but the hardware configuration ultimately determines the performance potential (or pitfalls) of a platform."

Performance-Related Architecture Considerations

The most important and often overlooked factor attributing to system performance is the hardware configuration. Platform hardware and software tuning can help extract the last bit of performance out of a system, but the hardware configuration ultimately determines the performance potential (or pitfalls) of a platform.

System architects may unintentionally end up designing a lower performing solution by choosing a less than desirable configuration in an attempt to save cost, power, thermals or design time. These sections address architecture considerations to help system architects and end users understand the performance impact of various tradeoffs when selecting CPU, memory, and I/O configurations.

CPU

The best processor configuration for computation performance will be the CPU with the maximum possible CPU frequency, cores/threads, sockets, and FSB speed. However this configuration does not always fit into the end budget, thermal, and power constraints. In this section we will explore performance impacts of choosing FSB frequency, dual-core versus quad-core; single socket versus dual socket, and finally a single socket quad-core compared to a dual socket dual-core on the Intel 5100 MCH chipset architecture.

FSB Frequency

As explained earlier, the Intel 5100 MCH chipset supports FSB of 667, 1067, and 1333 MT/s. For the best performance, processors supporting FSB of 1333 MT/s should be used. Note that higher FSB frequency not only results in higher effective FSB bandwidth and lower latency to memory but also makes the Intel 5100 MCH chipset core frequency proportionally faster. We will review FSB and memory frequency performance data later in the article.

Quad-Core versus Dual-Core

Here we illustrate the performance benefit of the quad-core architecture versus dual-core with the benchmark SPEC CPU2006. SPEC CPU2006 is a suite of tests many of which are CPU compute–bound while some are CPU to memory bandwidth–bound. For complete information on SPEC CPU2006 refer to: *http://www.spec.org*. [3]

The performance gain on quad-core will vary per sub-test and depends on the limiting factors mentioned above. Sub-tests limited by CPU-memory path bandwidth see little to no benefit from quad-core while those limited by CPU computation scale almost perfectly. The end score is an averaging of these sub-tests.

Lab testing shows that the quad-core platform configuration improves the SPECfp_rate_base2006(est.) score by an average of 40 percent and the SPECint_rate_base2006(est.) score by an average of 57 percent over the dual-core platform configuration. As expected, some computation-heavy sub-tests show nearly perfect scaling from dual-core to quad-core while memory intensive sub-tests show no scaling from dual-core to quad-core.

Single Socket versus Dual Socket

Now we show the performance benefit of a single socket architecture with a quad core CPU versus dual socket architecture with a quad-core CPU again using the benchmark SPEC CPU2006. This configuration compares four cores on one socket against eight cores across two sockets. Again, the scaling results come down to CPU computation power and memory bandwidth. Our computation power is doubling so we can expect about the same scaling here as with dual-core versus quad-core, but we also expect higher bandwidth with the additional FSB.

Tests show a 48 percent benefit on SPECfp_rate_base2006(est.) and 59 percent for SPECint_rate_base2006(est.) for this configuration.

Dual-Core and Dual Socket versus Quad-Core and Single Socket

Here we examine the performance difference between platforms configured with two dual-core processors versus a single socket populated with a quad-core. This is an interesting comparison since we are comparing an equal number of processor cores, but varying the number of physical processors and FSBs used.

Generally, the dual-core dual socket configuration yields higher performance but also increases system cost and power consumption. At first glance, the performance seems very similar with a 6 percent increase for SPECfp_rate_base2006(est.) and SPECint_rate_base2006(est.) is within test noise of 3 percent.

To better understand this we need to look at the sub-tests of SPEC CPU2006 Floating Point. Many of the tests perform the same on the two configurations. These tests are CPU-bound and performance is dependent on the processor core count and frequency. Some of the tests perform significantly better on the dual socket configuration. These tests are not CPU-bound, but rather memory-bound and dual socket configuration gives higher CPU-memory throughput. Figure 2 shows the tests and percentage of increase for those tests sensitive to CPU-memory throughput such as: 410.bwaves, 433.milc, 437.leslie3d, 450.soplex, 459.GemsFDTD, 470.lbm, 481.wlf, and 482.sphinx3. The remaining floating point tests are those that are computation-bound.

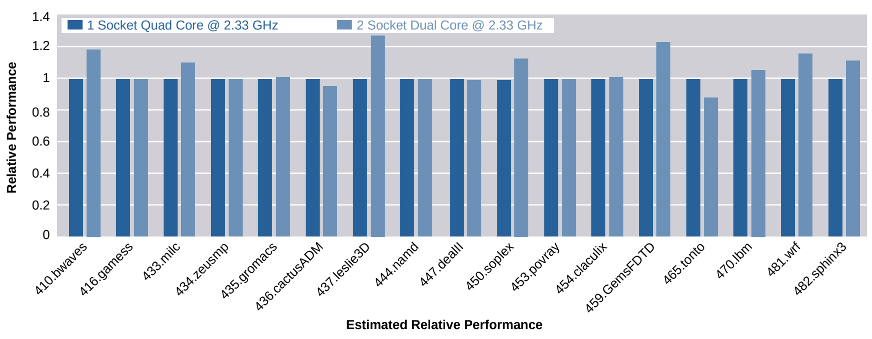


Figure 2: CPU2006fp performance, single socket quad-core versus dual socket dual-core. Source: Intel Corporation, 2009

"The memory subsystem is a vital component to platform performance and often becomes the limiting factor of benchmark throughput."

Based on SPEC CPU2006, we see that the computation performance of dual core with dual socket design when compared to a quad core single socket design is similar, but memory throughput improves on the dual socket design compared to single socket, showing a 27 percent improvement using 2 sockets on 437.leslie3d(est.).

Memory

The memory subsystem is a vital component to platform performance and often becomes the limiting factor of benchmark throughput. Therefore it is critical to populate memory with end performance in mind. This section explores the performance impact of populating one versus two channels, one DIMM per channel, two DIMMs per channel, three DIMMs per channel, dual rank DIMMs, single rank DIMMs, and memory frequency.

Channel Population

The Intel 5100 MCH chipset features two independent DDR2 channels with each channel having its own independent memory controller. In order to get the most benefit out of the memory system, it is vital to populate both channels. Memory configurations with two or more DIMMs should divide DIMMs equally between the channels.

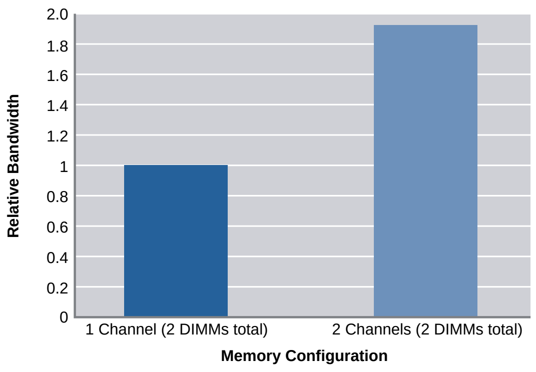


Figure 3: One channel versus two channel memory performance with 66% read 33% write
Source: Intel Corporation, 2009

Figure 3 illustrates the performance delta when two DIMMs are placed in one channel versus divided between channels. Memory bandwidth tests results are from an Intel internal benchmark that behaves much like Stream Benchmark, with higher memory efficiency. A 92 percent increase is observed from one channel to two channels with 1-GB, dual rank, DDR2-667 modules when CPU is issuing 66 percent read 33 percent write requests. From this data it is clear that utilizing both memory channels of the MCH is vital for memory performance. Based on the data in Figure 3, populating both memory channels is highly recommended.

DIMMs per Channel

Each memory channel on the Intel 5100 MCH chipset supports up to three DDR2 DIMMs. It is important to understand the performance impact of using one, two, or three DIMMs per channel to design a cost-effective product with high performance.

The estimated performance gains from one to two to three DIMM configurations with 1-GB, dual rank, 667 DDR2 modules is a 4.5 percent improvement from one DIMM to two DIMMs. Adding a third DIMM per channel does not typically increase bandwidth potential unless a benchmark is memory capacity limited. Populating three DIMMs per channel may potentially yield higher application/benchmark performance for capacity limited usage, but actually memory bandwidth will not increase. These data points indicate that utilizing both memory channels is much more important than populating multiple DIMMs on just one channel. Usage models requiring high memory throughput should populate two DIMMs per channel to gain the additional bandwidth while models with strict power and cost limits may consider using only one DIMM per channel. One must also consider the target software applications that will be executing on the platform. If maximum memory capacity is required then all three DIMMs per channel should be populated to achieve 4848 GB of total system memory.

"These data points indicate that utilizing both memory channels is much more important than populating multiple DIMMs on just one channel."

Dual Rank versus Single Rank

Dual rank DIMMs are recommended over single rank for performance and to enable full rank interleaving of 4:1. Figure 4 shows the measured benefit of dual rank with 66 percent read 33 percent write traffic on various memory configurations. For the maximum configuration the dual rank memory provides an additional 6.5 percent throughput. Note: Three DIMMs per channel provides additional capacity, but not bandwidth (not shown in Figure 4).

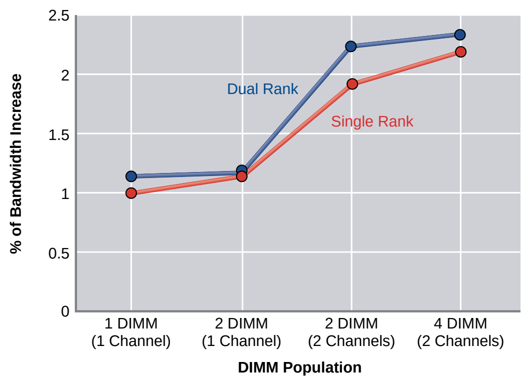


Figure 4: Single rank versus dual rank performance. Source: Intel Corporation, 2009

DDR2-533 versus DDR2-667

When selecting memory frequency, the FSB frequency must also be considered due to the impact of memory gearing. Memory gearing refers to the frequency ratio between the front side bus and memory interface. When the ratio is not 1:1, additional memory latency occurs. Table 1 shows possible frequency ratios related to memory gearing.

FSB Frequency	DDR2 Frequency	Memory Gearingvia I/O Controller Hub (ICH)
1067 MT/s	533 MHz	1:1 Ratio
1067 MT/s	667 MHz	Not a 1:1 Ratio
1333 MT/s	533 MHz	Not a 1:1 Ratio
1333 MT/s	667 MHz	1:1 Ratio

Table 1: Memory gearing table

Testing shows that the reduced memory latency of 667 MHz over 533 MHz makes up for the negative impact of memory gearing. So a simple rule of thumb is that higher memory frequency provides higher performance. Figure 5 reports the relative CPU-memory bandwidth recorded with each combination in dual socket configuration. Note that there is no gain from a 1067/533 configuration to 1067/667, but we show an estimated 22 percent bandwidth improvement with the 1333/667 configuration.

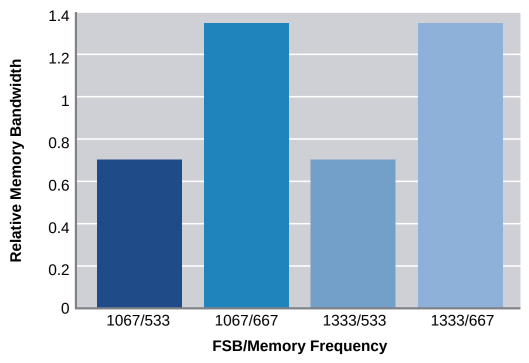


Figure 5: FSB and memory frequency bandwidth with 66% read 33% write.
Source: Intel Corporation, 2009

PCI Express*

Recall from the block diagram in Figure 1 that the PCI Express (PCIe) ports of the Intel 5100 MCH chipset can be configured in many ways. There are six x4 PCIe lanes available directly from the MCH. These can be configured as: a single x16 link with a 1x8 link, three x8 links, or six x4 links. There are also six x1 links available from the I/O controller hub (ICH), which can be combined for form one x4 and two x1 links. This section addresses performance related to these choices.

Lane Width

PCI Express lane width should be chosen based on required bandwidth of the I/O device. Peak PCI Express bandwidth efficiency is about 81 percent for reads and 86 percent for writes on the Intel 5100 MCH chipset.

Another important concept with link width is transaction latency. Transmit time of PCI Express packets increases as the link width decreases. For this reason, performance can benefit from link width even when higher bandwidth is not demanded.

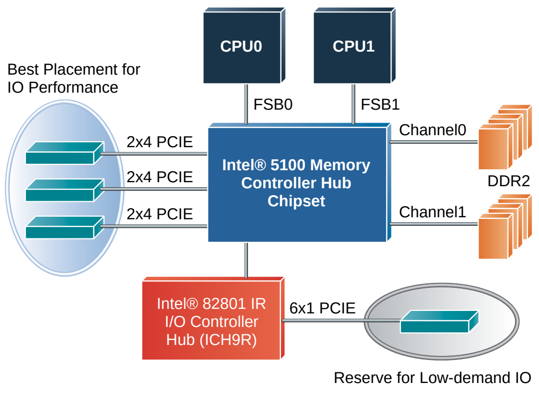

Best Placement for IO Performance
CPU0
CPU1
FSB0
FSB1
2x4 PCIE
2x4 PCIE
2x4 PCIE
Intel® 5100 Memory Controller Hub Chipset
Channel0
DDR2
Channel1
Intel® 82801 IR I/O Controller Hub (ICH9R)
6x1 PCIE
Reserve for Low-demand IO

Figure 6: I/O device placement recommendation.
Source: Intel Corporation, 2009

I/O Device Placement

For best performance, place I/O devices as close to the Intel 5100 MCH chipset as possible. Hence the use of direct MCH attached PCIe interfaces is recommended for performance sensitive IO devices. On the other hand, having relatively higher latency to memory, the PCIe slots on the I/O Controller Hub are recommended for less performance-sensitive applications. Note that increased latency also impacts throughput; how much will depend on how many outstanding transactions are pending. Tables 2 and 3 show the relative latency and bandwidth measured on a x4 PCI Express link via Intel 5100 MCH chipset and I/O Controller Hub. Note: The bandwidth measurement are carried out with up to 32 outstanding requests, hiding much of the latency impact on bandwidth while latency measurements are carried using one outstanding transaction at a time. Figure 6 illustrates recommended IO device placement for best performance.

X4 PCI Express* (PCIe*) 64-B Memory Read	Intel® 5100 Memory Controller Hub Chipset PCIe Slot	via I/O Controller Hub PCIe Slot
Relative Latency	1	2.07

Table 2: x4 PCI Express* (PCIe*) latency (lower is better)

X4 PCI Express* (PCIe*) 2-KB Requests to Memory	via Intel® 5100 Memory Controller Hub Chipset PCIe Slot	via I/O Controller Hub PCIe Slot
Relative Read Bandwidth	1	.90
Relative Write Bandwidth	1	.99
Relative Read/Write Bandwidth	1	.866

Table 3: x4 PCI Express* (PCIe*) bandwidth (higher is better)

"Populate PCI Express slots with the intent of balance loading traffic between the IOUs."

Another important concept in I/O device placement is I/O Unit (IOU) balance within the MCH. As shown in Figure 7, the Intel 5100 MCH chipset's available PCI Express slots are divided between IOU0 and IOU1. IOU0 contains the x4 DMI link and 2x4 or 1x8 PCI Express links. IOU1 contains 4x4 or 2x8 PCI Express links. Populate PCI Express slots with the intent of balance loading traffic between the IOUs. Figure 7 also illustrates an example ordering preference to ensure IOU balance.

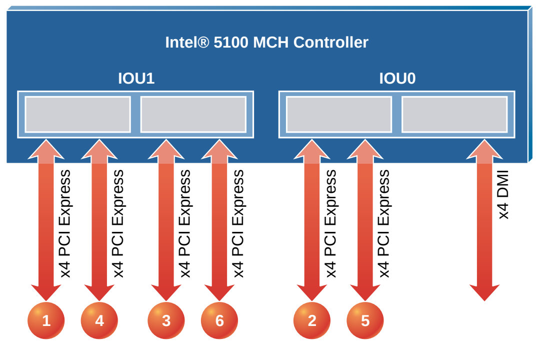


Figure 7: I/O unit layout.
Source: Intel Corporation, 2009

Out of the Box Performance

Prior to product release, Intel conducts performance testing and analysis to determine ideal default chipset and processor settings. This work results in the best "out of the box" performance for general usage models and in most cases additional tuning is not necessary.

Additional performance might be achieved depending on the CPU and memory configuration used, or the specific usage model of interest. Some of these additional tuning options are presented in the following section.

Accessing the Intel® 5100 Memory Controller Hub Chipset Control Registers

The following sections about hardware tuning require an understanding of chipset control registers. Inside the Intel 5100 MCH chipset there are registers related to status, capabilities and control. These registers are defined in the datasheet [2]. Changing values within these registers can control specific behaviors of the Intel 5100 MCH chipset. Registers are mapped to the PCI configuration space and they can be accessed with the correct physical address. The addressing nomenclature of the PCI configuration space is: bus, device, function, offset, and bits.

"Inside the Intel 5100 MCH chipset there are registers related to status, capabilities and control."

Linux provides the commands "lspci" and "setpci" for listing and changing the PCI configuration space. For example, let us say we want to disable FSB1. According to Table 4 we need to set bit 30 for bus 0, device 16, function 0 at offset 78h.

Register	Field	Bus	Device	Function	Offset	Bit	Value	Result
FSBC[1]	FSB1_dis	0	16	0	78h	30	1	FSB1 Disabled

Table 4: Disable FSB 1 example. Source: Intel Corporation, 2009

lspci –s 0:10.0 –xxx dumps config space for bus 0, device 10h, function 0. From this output we can locate the register at offset 78h (highlighted below in bold text).

```
[root]# lspci -s 0:10.0 -xxx

00:10.0 Host bridge: Intel Corporation: Unknown device 65f0 (rev 90)
00: 86 80 f0 65 00 00 00 00 90 00 00 06 00 00 80 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 86 80 86 80
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40: 00 00 ff 07 00 08 00 00 00 00 00 fe 00 00 00 00
50: 00 00 02 00 00 00 04 04 00 10 11 01 00 00 31 33
60: 00 12 08 01 00 e0 00 00 ff ff ff ff 00 00 00 00
70: 8c c0 e2 0f 00 00 00 00 8c c0 e2 0f 00 00 00 00
80: 01 00 80 00 04 02 80 00 00 00 00 00 00 00 00 00
90: 02 01 80 00 08 03 80 00 00 00 00 00 00 00 00 00
a0: 00 00 00 00 00 00 00 00 10 04 80 00 40 06 80 00
b0: 00 00 00 00 00 00 00 00 20 05 80 00 80 07 80 00
c0: 00 00 00 00 5a 5a 5a 5a 5a 5a 5a 5a 00 00 00 00
d0: 00 00 00 00 00 00 00 00 0c 2c 00 00 03 01 00 00
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
f0: dc 7f 40 00 06 81 58 81 f4 08 d8 03 80 00 00 00
```

"Remember that this is little endian, requiring us to flip it backwards per byte."

Remember that this is little endian, requiring us to flip it backwards per byte, resulting in: 0F E2 C0 8C. For completeness, this is broken down to bit level with bit 30 highlighted:

Default Setting:	0**0**00	1111	1110	0010	1100	0000	1000	1100
New Setting:	0**1**00	1111	1110	0010	1100	0000	1000	1100

Working backwards, the result is 4F E2 C0 8C, showing that we need to set offset 7Bh = 4Fh as follows:

```
setpci –s 0:10.0 7b=4f
```

Hardware Tuning Recommendations

There are several hardware level settings that can be changed through BIOS menus or chipset control registers that can help improve performance. Here we discuss a few of these options for the Intel 5100 Memory Controller Hub.

"Intel® Core™ microarchitecture has two hardware level prefetch mechanisms to help reduce CPU memory read latency."

Intel® Processor Prefetching

Intel® Core™ microarchitecture has two hardware level prefetch mechanisms to help reduce CPU memory read latency. These prefetchers bring data into processor L2 cache before the processor requires it in an attempt to produce a cache hit rather than a miss, resulting in increased performance.

There are two prefetchers available within the CPU architecture and can be set within BIOS, the hardware prefetch, also known as Data Prefetch Logic or DPL and the L2 Streaming Prefetch (L2S), also known as Adjacent Sector Prefetch.

Tuning processor prefetching is a topic within itself and is beyond the scope of this article. Please refer to the following paper for further information on processor prefetching:
http://software.intel.com/en-us/articles/optimizing-application-performance-on-intel-coret-microarchitecture-using-hardware-implemented-prefetchers [4]

Hardware Prefetch

The default recommendation is to enable hardware prefetch. However, depending on individual usage models, hardware prefetch may also fetch more cache lines than are needed by an application. This can result in increased memory bus utilization and may affect performance under usage models that require heavy memory bandwidth. It is left to the user to choose hardware prefetch settings that best suit the application under consideration. It is beneficial to test with HW Prefetching ON and also OFF in order to determine the optimal performance for the specific usage model.

"It is beneficial to test with HW Prefetching ON and also OFF in order to determine the optimal performance for the specific usage model."

L2 Streaming Prefetch

L2S improves performance under some usage models with sequential memory addressing and/or spatial locality. L2S may be enabled/disabled via BIOS setting. Again, the effect of L2S on the performance is application-specific.

FSB Tuning for Single Socket Configurations

In single socket configurations, the second, unused socket should not be populated with a processor and BIOS should disable the unused FSB. The motivation is to improve performance by allowing the bus to switch to in-order mode rather than deferred mode; this reduces transaction latency and protocol overhead.

Table 4 above shows the bit in the FSBC[1] register that can be checked to verify that BIOS is disabling the second FSB in single socket configurations. For complete register definition, please see the Intel 5100 MCH Chipset Datasheet [2].

"Memory timing settings are set to optimum values by default, providing the best performance possible within specification."

Memory Tuning

Memory timing settings are set to optimum values by default, providing the best performance possible within specification. DRAM timing registers should not be modified by end users except through preset BIOS settings.

For recommended memory settings, the following parameters should be selected in BIOS:

- MCH Channel Mode: Channel Interleave
- Channel Dependent Sparing: Disabled
- Channel Specific Sparing: Disabled
- Rank Interleave = 4:1
- Channel 0: Enabled
- Channel 1: Enabled
- DIMM Calibration Reuse: Enabled
- Read Completion Coalesce: Auto

PCI Express* Tuning with Maximum Payload Size

The Intel 5100 MCH chipset supports a PCI Express Maximum Payload Size (MPS) of 128 bytes and 256 bytes. The default and recommended setting is 128. A 128 byte payload size allows opportunistic split completion combining (coalescing) for read requests, a feature not supported with a 256-byte MPS.

Under specific I/O usage models that perform high percentages of inbound writes with large payloads and few inbound reads, it will benefit performance to disable coalescing and increase MPS to 256 bytes. This allows I/O devices to send up to 256 bytes of data per write packet, improving write throughput but limits the maximum read completion size to 64 bytes, reducing read throughput potential.

"Under specific I/O usage models that perform high percentages of inbound writes with large payloads and few inbound reads, it will benefit performance to disable coalescing and increase MPS to 256 bytes."

Table 5 defines the register changes required in the Intel 5100 MCH chipset to implement a 256-byte MPS tweak. Note that end devices must also be changed to 256-byte MPS.

Register	Field	Bus	Device	Function	Offset	Bit(s)	Value
PEXCTRL[7:2,0]	COALESCE_EN	0	7-2,0	0	48h	10	0
PEXDEVCTRL[7:2,0]	MPS	0	7-2,0	0	74h	5-7	001b

Table 5: Enable 256-byte maximum payload size. Source: Intel Corporation, 2009

Throttling Mechanisms

During performance benchmarking it is sometimes useful to disable the throttling technologies of the Intel 5100 MCH chipset to verify that performance limits are not throttle related.

Thermal Throttle

Thermal throttle allows for dynamic frequency scaling based on defined thermal thresholds, but the Intel 5100 MCH chipset does not implement thermal throttling.

Other platform components may support thermal throttle, such as the CPU. Please refer to component-specific documentation on how to disable thermal throttle.

Global Throttle

Global throttle allows for software controlled throttling on memory activations for a long time window, as shown in Table 6. If the number of activations to a memory rank exceeds the specified limit, then further requests are blocked for the remainder of the activation window. Setting the following bit values to zero will disable global throttle features.

Register	Field	Bus	Device	Function	Offset	Bit(s)	Value
THRTHIGH	THRTHIGHLM	0	16	1	65h	7-0	0
THRTLOW	THRTLOWLM	0	16	1	64h	7-0	0
GBLACT	GBLACTLM	0	16	1	60h	7-0	0

Table 6: Disable global throttle. Source: Intel Corporation, 2009

Electrical Throttle

Electrical throttling is a mechanism that limits the number of activations within a short time interval that would otherwise cause silent data corruption on the DIMMs. Disable electrical throttle with the configuration settings listed in Table 7.

Register	Field	Bus	Device	Function	Offset	Bit(s)	Value
MTR[1:0][3:0]	ETHROTTLE0	0	22	0	15Ah	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	22	0	158h	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	22	0	156h	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	22	0	154h	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	21	0	15Ah	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	21	0	158h	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	21	0	156h	9	0
MTR[1:0][3:0]	ETHROTTLE0	0	21	0	154h	9	0

Table 7: Disable electrical throttle. Source: Intel Corporation, 2009

Conclusion

When designing a solution with the Intel 5100 MCH chipset the designer should make architecture decisions based on CPU population, FSB frequency, memory population, memory frequency, I/O device selection, and placement with an understanding of how it will impact end performance. All areas of architecture should be carefully thought out with the end solution in mind to balance power, performance, cost and thermals.

"All areas of architecture should be carefully thought out with the end solution in mind to balance power, performance, cost and thermals."

Disclaimers

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations at: *http://www.intel.com/performance/resources/limits.htm*

Intel, and the Intel logo, Intel Core, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

Intel processor model numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families.

SPECint*_rate_base2006 and SPECfp*_rate_base2006 are capacity-based metrics used to measure throughput of a computer that is performing a number of tasks. This is achieved by running multiple copies of each benchmark simultaneously with the number of copies set to set to the number of logical hardware cores seen by the operating system. SPEC* CPU2006 provides a comparative measure of compute intensive performance across the widest practical range of hardware. The product consists of source code benchmarks that are developed from real user applications. These benchmarks depend on the processor, memory and compiler on the tested system. SPEC, SPECint, SPECfp, SPECrate are trademarks of the Standard Performance Evaluation Corporation. For more information go to: www.spec.org/spec/trademarks.html. [1]

All SPEC CPU2006 data in this document is estimated based on measurements of Intel internal reference platforms; the data is being provided as described in the CPU2006 Run Rules 4.5 Research and Academic usage of CPU2006.

References

[1] Use of SPEC Trademarks and Service Marks, URL: *http://www.spec.org/spec/trademarks.html*

[2] Intel® 5100 Memory Controller Hub Chipset Datasheet, July 2008, Revision 003US

[3] Standard Performance Evaluation Corporation (SPEC), URL: *www.spec.org*

[4] Hedge, Ravi. 2008. "Optimizing Application Performance on Intel® Core™ Microarchitecture Using Hardware-Implemented Prefetchters." Intel® Software Network, URL: *http://software.intel.com/en-us/articles/optimizing-application-performance-on-intel-coret-microarchitecture-using-hardware-implemented-prefetchers/*

Author Biography

Perry Taylor: Perry Taylor is a senior performance engineer in the Embedded and Communications Group at Intel Corporation. Perry has been at Intel for 8 years and in that time has focused on performance analysis and tuning of Intel® architecture and has received multiple divisional awards for his contributions.

Copyright

SOLID STATE DRIVE APPLICATIONS IN STORAGE AND EMBEDDED SYSTEMS

Contributors

Sam Siewert, PhD
Atrato Incorporated

Dane Nelson
Intel Corporation

Index Words

Intel® X25-E SATA Solid-State Drive
Intel® X25-M SATA Solid-State Drive
SSDs
RAID

Abstract

Intel® X25-E and X25-M SATA Solid-State Drives have been designed to provide high performance and capacity density for use in applications which were limited by traditional hard disk drives (HDD), input/output (I/O) performance bottlenecks, or performance density (as defined by bandwidth and I/Os/sec per gigabyte, per Rack Unit (RU), per Watt required to power, and per thermal unit waste heat. Solid State Drives (SSDs) have also found a place to assist in capacity density, which is the total gigabytes/terabytes per RU, per Watt, and per thermal unit waste heat. Enterprise, Web 2.0, and digital media system designers are looking at SSDs to lower power requirements and increase performance and capacity density. First as a replacement for high end SAS or Fiber Channel drives, but longer term for hybrid SSD + Hard Disk Drive (HDD) designs that are extremely low power, high performance density, and are highly reliable. This article provides an overview of the fundamentals of Intel's Single Level Cell (SLC) and Multi Level Cell (MLC) NAND flash Solid State Drive technology and how it can be applied as a component for system designs for optimal scaling and service provision in emergent Web 2.0, digital media, high performance computing and embedded markets. A case study is provided that examines the application of SSDs in Atrato Inc.'s high performance storage arrays.

Introduction

Flash memory, especially NAND flash memory, has been steadily encroaching into new markets as the density has increased and the cost per gigabyte (GB) has decreased. First it was digital cameras, then cell phones, portable music players, removable digital storage, and now we are seeing the emergence of NAND based solid state drives (SSDs) in the consumer PC market. Some industry analysts predict that SSDs could be the single largest NAND market segment (in billions of GB shipped) by 2010.

The combined performance, reliability, and power of SSDs compared to traditional hard disk drives (HDD), explains the attraction of SSDs in the consumer marketplace.

Intel Corporation has launched a line of high performance NAND based solid state drives; the Intel® X25-M and X18-M Mainstream SATA Solid-State Drives utilizing MLC NAND, and Intel® X25-E SATA Solid-State Drive utilizing SLC NAND. The higher performance density and lower price point make them a practical choice for the storage and embedded markets.

"Some industry analysts predict that SSDs could be the single largest NAND market segment (in billions of GB shipped) by 2010."

"The rotating media and servo-actuated read/write heads used to access HDD data are subject to mechanical failure and introduce seek and rotate latency."

The purpose of this article is to examine the unique benefits of Intel® Solid State Drive (Intel® SSD) over traditional HDDs and competing SSDs, and to explore the benefits one could realize in using these new high performance SSDs in storage and embedded applications.

Solid State Drives versus Hard Disk Drives

Solid State Drives have no moving parts, unlike HDDs. The rotating media and servo-actuated read/write heads used to access HDD data are subject to mechanical failure and introduce seek and rotate latency. Capacity growth due to areal density advancement and low cost per gigabyte stored have been the main advantages of HDDs, but fast random access has always been a significant limitation.

Physical Differences

The main difference between an HDD and SSD is the physical media in which the data is stored. A HDD has platters that encode digital data with magnetically charged media. These magnetic platters spin at a high rate of speed (5,400, 7,200, or 15,000 revolutions per minute, or RPM) so that a servo-controlled read/write head can be positioned over the cylinders/tracks of sector data for data access. In an SSD the digital data is stored directly in silicon NAND flash memory devices. An SSD has no mechanical moving parts, which improves the durability in resisting physical shock or mechanical failure and increases performance density. An SSD then uses a controller to emulate a mechanical hard disk drive, making it a direct replacement for mechanical hard disk drives but with much faster data access due to the lack of the servo positioning latency in HDDs.

"An SSD then uses a controller to emulate a mechanical hard disk drive, making it a direct replacement for mechanical hard disk drives but with much faster data access."

In addition to the memory, a solid state drive contains; an interface connector and controller, memory subsystem and controller, and a circuit board where all the electronics are housed.

Performance Differences

The main performance difference between HDDs and SSDs has to do with the limitation of the HDDs caused by the spinning mechanical platters. Two performance metrics that improve greatly are the random reads and writes per second, and the time it takes to enter and resume from a low power state.

Random read and write performance is measured in inputs/outputs per second, or IOPs. This is simply the number of reads or writes that can be completed in one second. A typical high performance 15-K RPM SAS hard drive can usually complete about 300 IOPs of random 4-kilobyte (KB) data. By comparison, the Intel X25-E SATA Solid-State Drive is able to process over 35,000 random 4-KB read IOPs, a difference of 117 times. The reason for this is that logical data locations on an HDD are directly mapped to ordered physical locations on the spinning physical disks. To access (read or write) that data, the disk must spin around to the correct location and the read/write head must move to the correct radius to access the data. Therefore, random data accesses require multiple ordered physical movements incurring significant mechanical access latency and significantly limiting performance.

"The main performance difference between HDDs and SSDs has to do with the limitation of the HDDs caused by the spinning mechanical platters."

In a NAND SSD the data is stored in a virtual memory map on the NAND flash. Accessing any part of that is as simple as changing the address and executing the next read or write. Since most workloads are random in nature, especially as industries move toward multi-core compute engines, multithreaded operating systems, and virtual machines, random disk performance will only increase in importance.

The other main performance difference is the ability to resume operation from a low power state quickly. The lower power state for a HDD is accomplished by parking the read/write head off to the side and stopping the spinning platter. When the next read or write is requested the platter needs to be spun back up and the head has to be moved back in place, which can take on the order of seconds. In an SSD however, when it is not processing read or write requests, it can put itself into a low power state (through Device Initiated Power Management, or DIPM) and recover within a few milliseconds to service the next request. Hard drives take closer to seconds to do this, so they do not take full advantage of DIPM.

"In an SSD however, when it is not processing read or write requests, it can put itself into a low power state (through Device Initiated Power Management, or DIPM) and recover within a few milliseconds to service the next request."

Barriers to Adoption

With SSDs' higher performance, added reliability, and lower power, one may ask why they have not completely displaced HDDs. The main reason is cost, because SSDs cost many times more per gigabyte than mechanical hard drives today. There has been early adoption in markets that absolutely must have the performance, reliability, lower power, or resilience to shock and vibration, but mass consumer adoption will only happen when the cost of a comparably sized device approaches that of a Small Form Factor (SFF) hard disk drive.

"With SSDs' higher performance, added reliability, and lower power, one may ask why they have not completely displaced HDDs."

The second barrier is capacity, because SSDs typically have much smaller capacity than mechanical hard drives. As NAND flash densities increase, however, the capacity of SSDs will be large enough for most consumer, enterprise, and embedded marketplace needs.

Third, the limited number of write cycles per storage cell is a barrier to applications that mostly ingest data (50-percent writes, 50-percent reads) for later access. Flash memory, as it is erased and rewritten, will lose the capability to hold a charge after many program/erase cycles. This makes flash memory a consumable resource. However, with increased density, along with more comprehensive write wear-leveling algorithms, the longevity of solid state drives has improved.

Ideal Use Cases

Just because SSDs have an inherent advantage over HDDs in random read/write performance and in resumption from low power states doesn't mean that SSDs are better than HDDs in every case. Hard disk drives are excellent for storing massive amounts of data such as movies, music, and large amounts of digital content in general, due to their very low cost per gigabyte and continued improvements in areal density (gigabits/square-inch). Areal density improvements in HDD media technology have in fact followed or exceeded Moore's Law (capacity density doubling every 18 months or less), but access to that data has not improved at the same pace. The servo and rotational latency for access to data in HDDs has in fact been nearly the same for decades if you look at year-over-year improvements.

"Just because SSDs have an inherent advantage over HDDs in random read/write performance and in resumption from low power states doesn't mean that SSDs are better than HDDs in every case."

"Solid state drives, on the other hand, excel when the system requirements are more skewed toward performance, reliability, or power."

Solid state drives, on the other hand, excel when the system requirements are more skewed toward performance, reliability, or power. The performance and power metrics were described above, and the reliability of NAND storage in SSDs has been shown to be many times more reliable than mechanical hard disk drives especially in harsh environments.

Intel® Solid State Drive (Intel® SSD) Architecture and Design Considerations

The following sections describe the physical design and performance of Intel Solid State Drives.

Physical Design

Most solid state drives have a similar architecture; they all have a circuit board, interface controller, memory subsystem, and a bank of NAND flash memory. Intel SSD is no exception; it has 10 channels of NAND flash attached to the controller and it has a complex flash controller with advanced firmware, which allows it to achieve high random read write performance while at the same time managing the physical NAND to achieve the longest possible use of the drive.

Measuring SSD Performance

Traditional hard disk drive performance criteria apply directly to SSDs. The most common performance testing metrics are random and sequential sustained read/write bandwidth, random and sequential read/write IOPs, and power consumed in both active and idle states.

Sequential sustained read/write rates are mainly a reflection of the amount of parallel NAND channels that can be activated at once. Intel's 10 NAND channels allow for a very fast sequential throughput to the raw NAND, as is seen in the graph in Figure 1 showing sustained throughput versus data transfer size.

Random sustained read/write rates are mainly due to how well the controller and firmware can handle multiple outstanding requests. Newer SATA system architectures incorporate Native Command Queuing (NCQ), which allows multiple outstanding disk requests to be queued up at the same time. In random performance the Intel X25-M and X18-M Mainstream SATA Solid-State Drives, and Intel X25-E SATA Solid-State Drives provide read performance that is four times that of a typical 2.5" SFF HDD, twice that of a 3.5" enterprise HDD and for random IOPs they provide improvement by several orders of magnitude.

Sequential and random IOPs in SSDs are affected by the number of channels accessing the NAND memory, as well as the architecture of the controller and data management firmware running on that controller. In Figure 1 and Figure 2 you can see Intel's performance across various workload sizes.

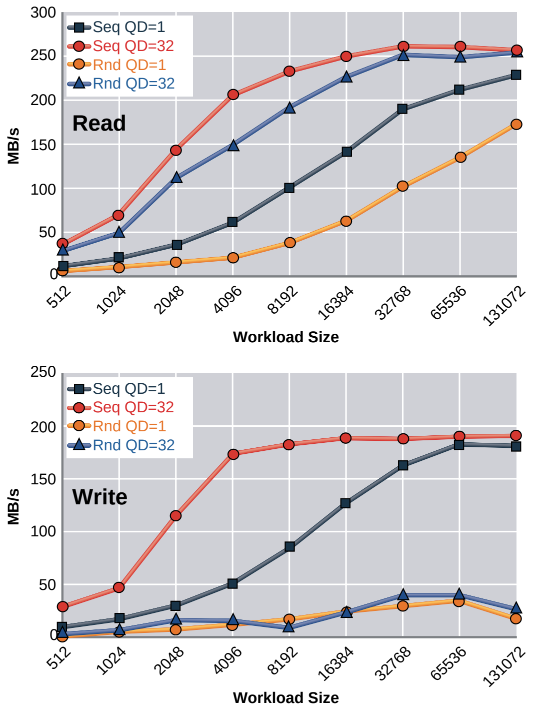


Figure 1: Performance for random/sequential read/write.Source: Intel Corporation, 2009

Wear Leveling and Write Amplification

Since NAND flash wears out after a certain number of program and erase cycles, the challenge is to extract maximum use from all of the NAND cells. The SSD's controller firmware must make sure that the various program and erase cycles that come to the SSD from the host system are evenly distributed over all sectors of NAND memory providing even wear over the entire drive. If not designed correctly, a log file or page table can wear out one section of the NAND drive too quickly. Figure 3 shows how Intel handles these small writes and spreads the wear over the whole drive, which is shown by charting the program/erase cycle count of each NAND cell within the drive. As one can see, Intel's controller wears evenly across every cell in the drive by distributing the writes evenly.

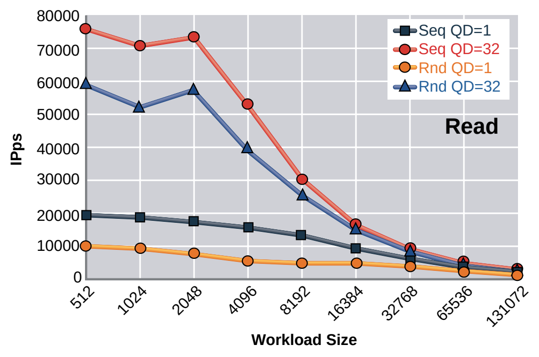
The second main attribute that contributes to wear on the drive is called Write Amplification (WA), which is basically the amount of data written to the raw NAND divided by the amount of data written to the SSD by the host. This is an issue because NAND cells are only changeable in erase block sizes of at least 128 KB, so if you want to change 1 byte of data in the SSD you have to first erase the block that byte resides in and then update the entire block with that 1 byte modified. The problem arises that more program/erase cycles are being used up than the actual amount of data sent to the drive by the host. Without careful NAND data management, WA levels can range from 20—40x. This means more erases (20–40x) of the NAND are being done then required based on new data sent to the SSD. The ideal case would be a WA of 1.0, which means that exactly the same amount of data would be written to the NAND as would be written to the SSD by the host.

Intel has taken a very close look at how to overcome this significant problem and has designed their controller accordingly. Intel's proprietary algorithms bring the WA of most compute applications very close to the ideal, and as one can see in the graph in Figure 4 for Microsoft Windows XP running MobileMark 2007 we measure a WA of less than 1.1.

Combining optimizations in both wear leveling and WA result in large increases to Intel SSD product longevity.

New Tier of Caching/Storage Subsystem

So far we have looked at a direct comparison between SSDs and HDDs without much examination of their application. There is the obvious direct-replacement market where HDDs are not meeting either the performance or reliability or power requirements of today's compute platforms. With high performance density SSDs the product designer has new options when designing embedded and scalable storage systems. The following sections examine how SSDs fit in today's storage and embedded products as well as how they could possibly be used in new ways to define tiered storage that enables new levels of access performance combined with scalability to many petabytes of capacity using both HDDs and SSDs.



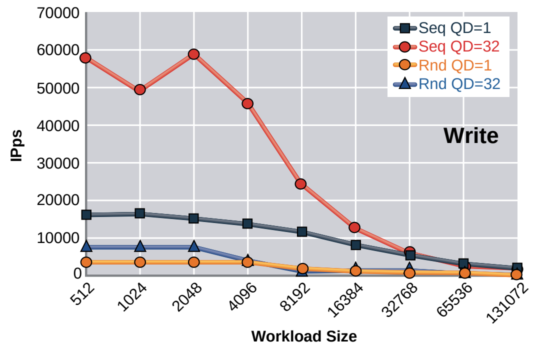


Figure 2: IOPs performance.
Source: Intel Corporation, 2009

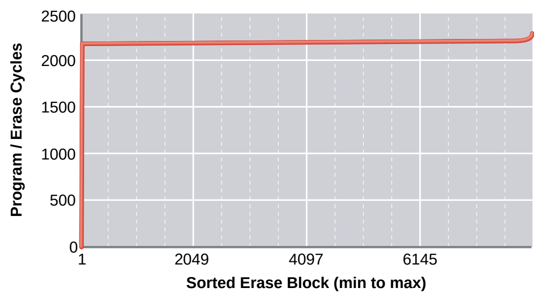


Figure 3: Erase cycle count showing Wear Leveling while running MobileMark 2007.
Source: Intel Corporation, 2009

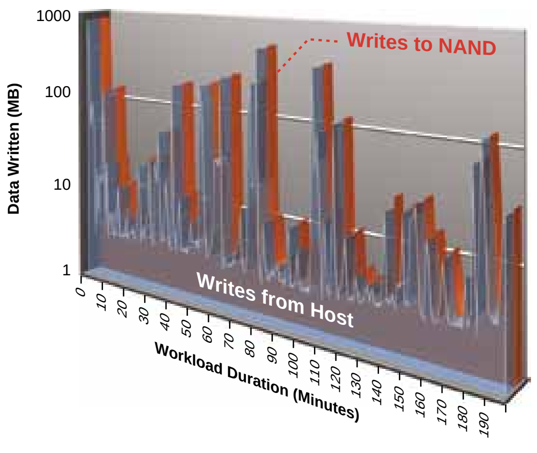


Figure 4: Microsoft* Windows XP Mobile workload writes. Source: Intel Corporation, 2009

"Storage in embedded mobile devices such as High Defintion (HD) digital cameras (1920 x 1080, 2048 x 1080, Digital Cinema 4096 x 2160, Red Scarlet* 6000 x 4000 high resolution frame formats) and consumer devices are pushing embedded storage requirements to terabyte levels."*

"The emergence of affordable SSD for laptops over the next few years will help accelerate the demand for more on-demand high definition content."

SSD in Embedded Storage Applications

Emerging markets in Internet Protocol Television (IPTV), Video on Demand (VoD), the Digital Cinema Initiative (DCI), and Web 2.0 are bringing more on-demand high definition content to a broader base of users. This means that increased capacity and performance density is required from embedded devices as well as head-end, content distribution, and edge service systems. Storage in embedded mobile devices such as High Defintion (HD) digital cameras (1920 x 1080, 2048 x 1080, Digital Cinema 4096 x 2160, Red* Scarlet* 6000 x 4000 high resolution frame formats) and consumer devices are pushing embedded storage requirements to terabyte levels. Likewise, capacity requirements for head-end digital media services are reaching petabyte levels. For example, one hour of HD content un-encoded raw format for 1080p at 24 fps would require 489 gigabytes. A library of that content with 10,000 hours would require around 5 petabytes of formatted capacity. Most often video is encoded for delivery to consumer devices, with compression ratios that are 30 to 1 or more. Even with encoding, a library of 100,000 hours (similar to total content at Netflix*) encoded in typical high definition distribution/transport format requires 2 to 4 gigabytes per encoded hour on average, so at least 200,000 gigabytes or 200 terabytes total storage. Because of the multiplicity of transport encodings, content is stored in many formats, so capacity requirements are increased again. In this next section, we'll analyze how combinations of SSD and high capacity and performance density hard disk drives (HDDs) in tiered storage can help eliminate storage and I/O bottlenecks from both embedded and server systems and make the all-digital-content revolution a reality.

Embedded Storage Growth

The increased capability of embedded storage and transport I/O in consumer devices has enabled the consumption of content at much higher bit rates. Progress in this embedded system segment has created demand for more high definition content from deeper content libraries. The emergence of affordable SSD for laptops over the next few years will help accelerate the demand for more on-demand high definition content. This means that the sources of content, starting with cameras, post-production, distribution, and finally delivery to consumers must all likewise upgrade to keep up.

General Architecture of Storage Applications

Today, most storage applications utilize HDDs, sometimes with redundant arrays of inexpensive disks (RAIDs) to scale capacity and performance. Embedded storage applications often make use of flash devices to store digital media, and small form factor HDDs to store larger content libraries. Content is often distributed on IEEE 1394 (such as FireWire*), USB 2.0 (Universal Serial Bus), or eSATA (external Serial Advanced Technology Attachment) external HDD when capacity is an issue, but this is less portable and often a significant I/O bottleneck. Media flash devices provide great I/O performance, but with very limited capacity (64 gigabytes is a typical high end device). For portable or semi-portable capacity and performance density, SSDs and SSD arrays will help change the landscape for portable storage architectures scaling to terabytes of capacity. As SSD cost continues down, the convenience, performance density, power, and durability of SSDs will likely drive mobile content storage completely to SSD. For system level content management with petabyte scale requirements, it is unlikely that SSD will replace HDDs for a very long time.

Today, most tiered storage moves content between flash media or SSD tiers and HDD tiers at a file level, with users actively managing how content is allocated between HDD and SSD tiers.

"Today, most tiered storage moves content between flash media or SSD tiers and HDD tiers at a file level, with users actively managing how content is allocated between HDD and SSD tiers."

System Issues Today Using HDD

If we look at a 2K/4K format digital video camera typically used in cinema today, these cameras can produce 250 Megabits per second (Mb/sec) in JPEG 2000 (Joint Photographic Expert Group) format, which is about 25 MB/sec or 90 GB/hour. Today's 2.5" SFF mobile class HDDs can keep up with this data rate and have capacities up to 500 gigabytes, which provides reasonable capture support for a single camera. The drawbacks though are that one HDD can not support multiple cameras, they have lower MTBF (Mean Time Between Failure) when used in harsh environments (often the case in filming), and they are slower to download from the HDD to a backup RAID for post production. Some cameras support raw 2K/4K video capture, which is 53-MB per frame and at 30 frames/sec, 1.5-GB/sec data capture per stream. These types of emergent capture rates will require solid-state storage solutions.

"Emergent capture rates will require solid-state storage solutions."

How SSDs Overcome These Issues

SSDs offer high-end digital 2K/4K/6K cameras the same advantages that smaller flash media provide consumers, but at capacities (160GB for Intel® X25-M SATA Solid-State Drive) that now make this a competitive option to HDD capture. This capacity offers approximately 2 hours of filming time and a capacity density that is competitive with SFF HDDs. The SSDs in this case would replace camera HDDs and offer lower power operation, equating to longer battery life, durability for filming in harsh environments, and high speed downloads to post-production RAID systems. The read rate of an Intel X25-E or X25-M SATA Solid-State Drive in sequential mode is at least four times that of typical SFF HDDs, so the download time will be far less. Even at raw 2K/4K rates of 1.5-GB/sec for uncompressed video ingest, it only requires 8 X25 SSDs to achieve full performance, however, at today's capacities (160 GB/SSD), the duration of ingest would only be 14 minutes (1.28 terabytes total SSD capacity for RAID0 mapping). One hundred percent ingest, rather than more typical 50 percent/50 percent write/read workloads is also a challenge for today's SSDs. Hybrid solutions with HDD backing SSD where SLC SSD is used as an ingest FIFO are perhaps a better approach and discussed in more detail in upcoming sections of this article.

"The read rate of an Intel X25-E or X25-M SATA Solid-State Drive in sequential mode is at least four times that of typical SFF HDDs, so the download time will be far less."

Future Design Possibilities Exploiting SSD Advantages

The packaging of flash media into 2.5" and 1.8" SFF SAS/SATA (Serial Attached SCSI/Serial Advanced Technology Attachment) drives that are interchangeable with current SFF HDDs will help SSD adoption in the embedded segment of the digital media ecosystem. The SCSI (Small Computer System Interface) command set or ATA (Advanced Technology Attachment) command sets can both be transported to HDDs or SSDs over SAS with SATA tunneling protocols. This provides a high degree of interoperability with both embedded applications and larger scale RAID storage systems. As SSD cost per gigabyte is driven down and durability and maximum capacity per drive driven up by adoption of SSDs on the consumer side, the attractiveness of SSD replacement of HDDs for cameras will increase. Building hybrid arrays of SSD and HDD even for mobile field arrays provides a much better adoption path where cost/benefit tradeoffs can be made and systems right-sized. A

"As SSD cost per gigabyte is driven down and durability and maximum capacity per drive driven up by adoption of SSDs on the consumer side, the attractiveness of SSD replacement of HDDs for cameras will increase."

"RAID storage system developers like Atrato Inc. have adopted SFF HDDs to increase performance density of HDD arrays."

key factor to success however is the development of software that can manage tiered SSD/HDD storage arrays for smaller mobile systems. This is even more important for post production, content delivery services, and the head-end side of the digital media ecosystem and will be covered in more detail in the following sections of this article.

Storage Challenges

Since magnetic media storage density has kept pace with Moore's Law, both storage consumers and the storage industry have focused on cost per gigabyte and capacity density as the key metric. However, access to that stored data in general has not kept pace. Most often access performance is scaled through RAID systems that stripe data and protect it with mirroring or parity so that more HDD actuators can be used in parallel to speed up access. The upper bound for HDD random data access is in milliseconds, which has meant that the only way to scale access to storage is to scale the number of spindles data is striped over and to pack more spindles into less physical space. RAID storage system developers like Atrato Inc. have adopted SFF HDDs to increase performance density of HDD arrays. The Atrato V1000 SAID (Self-Maintaining Array of Identical Disks) has 160 SFF HDDs (spindles) packed into a 3RU (rack unit) array. This is presently the highest performance density of any HDD RAID solution available. At the same time, the emergence of SSDs in capacities that approach HDD (today on can get a 160-GB Intel X25-M SATA Solid-State Drive compared to 500-GB 2.5" SATA HDD) and cost per gigabyte that is only ten times that of HDD, has made tiered hybrid storage solutions for terabyte and petabyte scale storage very attractive. Rather than direct HDD replacement, tiered storage solutions add SSDs to enhance HDD access performance. The key is a hybrid design with RAID storage that is well matched to SSD tier-0 storage used to accelerate data access to larger HDD-backed multi-terabyte or petabyte stores. The fully virtualized RAID10 random access, no cache performance of the Atrato V1000 array is up to 2-GB/sec at large block sizes with IOPs up to 17K at small block size (measured with an HP DL580 G5 controller, where the rate limiting factor is the PCI Express* generation 1 and memory controller).

"Rather than direct HDD replacement, tiered storage solutions add SSDs to enhance HDD access performance."

General Architecture of Storage Applications

Today most storage includes RAM-based I/O cache to accelerate writes on data ingest and to provide egress acceleration of reads through I/O cache read-ahead and hits to frequently accessed data. However, read cache often does little good for workloads that are more random and because the RAM cache sizes (even at 256 to 512 GB) are a very small fraction of capacity compared to petabyte back-end RAID storage (far less than one percent). Likewise, the cache miss penalty for missing RAM and going to an HDD backend is on the order of a 1000 to 1 or more (microsecond RAM cache access compared to millisecond HDD access). So, misses in RAM cache are likely and the penalty is huge, making RAM cache a wasted expenditure.

"As SSD cost per gigabyte is driven down and durability and maximum capacity per drive driven up by adoption of SSDs on the consumer side, the attractiveness of SSD replacement of HDDs for cameras will increase."

Figure 5 shows access patterns to storage that range from fully predictable/sequential to full random unpredictable access. Both SSDs and the high spindle density solutions perform well for random access. The SSDs provide this with the best overall performance and capacity density compared even to the high density HDD arrays like the SAID if cost per gigabyte is not an issue. The most interesting aspect of both of these emergent storage technologies is that they provide a performance matched tier-0 and tier-1 for highly scalable storage. In summary, the SSDs are about ten times the cost per gigabyte, but ten times the capacity/performance density of the SAID and the SAID is ten times the capacity/performance density of traditional enterprise storage. This can further be combined with a 3.5" SATA lowest cost per gigabyte capacity tier-2 (archive) when very low cost infrequently accessed storage is needed.

In the following sections, we'll examine how to tier arrays with an SSD tier-0.

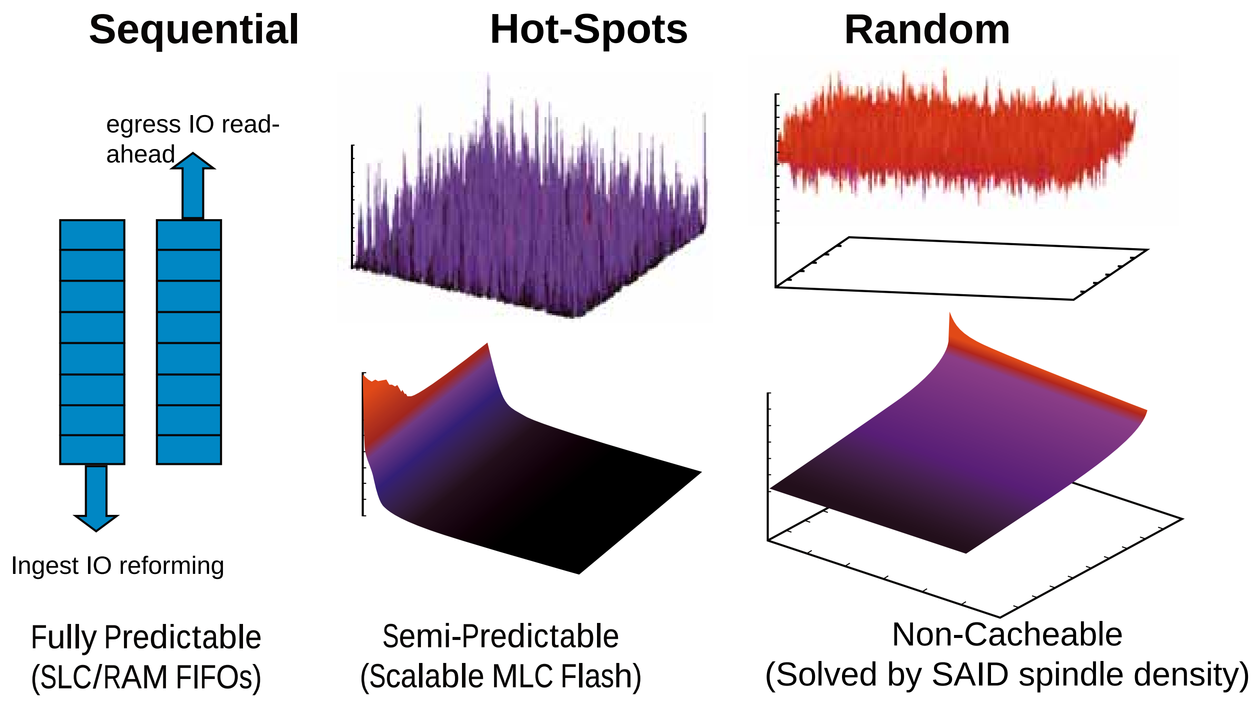

Sequential
Hot-Spots
Random
egress IO read-ahead
Ingest IO reforming
Fully Predictable (SLC/RAM FIFOs)
Semi-Predictable (Scalable MLC Flash)
Non-Cacheable (Solved by SAID spindle density)

Figure 5: Performance range of access patterns observed by ApplicationSmart* Profiler. Source: Atrato, Inc., 2009

In Figure 5, totally random workloads are best served by storage devices with high degrees of concurrent access, which includes both SSD flash and devices like the Atrato SAID with a large number of concurrent HDD actuators. The biggest challenge arises for workloads that are totally random and access hundreds of terabytes to petabytes of storage. For this case, the SAID is the most cost-effective solution. For much smaller stores with totally random access (such as hundreds of gigabytes to terabytes), SSD provides the best solution. It is not possible to effectively cache data in a tier-0 for totally random workloads, so workloads like this simply require mapping data to an appropriate all SSD or highly concurrent HDD

"For totally predictable sequential workloads, FIFOs (First-In-First-Out queues) can be employed, with SLC SSDs used for an ingest FIFO and a RAM FIFOs used for block read-ahead."

array like the SAID based on capacity needed. The most common case however is in the middle, where data access is semi-predictable, and where SSD and HDD arrays like the SAID can be coordinated with intelligent block management so that access hot spots (LBA storage regions much more frequently accessed compared to others) can be migrated from the HDD tier-1 up to the SSD tier-0. Finally, for totally predictable sequential workloads, FIFOs (First-In-First-Out queues) can be employed, with SLC SSDs used for an ingest FIFO and a RAM FIFOs used for block read-ahead. The ingest FIFO allows applications to complete a single I/O in microseconds and RAID virtualization software is used to reform and complete I/O to an HDD tier-1 with threaded asynchronous I/O, keeping up with the low latency of SSD by employing parallel access to a large number of HDDs. The exact mechanisms Atrato has designed to provide optimal handling of this range of potential workloads is provided in more detail in upcoming sections after a quick review of how RAID partially addresses the HDD I/O bottleneck, so we can later examine how to combine SSDs with HDD RAID for an optimal hybrid solution.

Performance Bottlenecks that Exist Today

"In general an HDD has a mean time between failure (MTBF) somewhere between 500,000 and 1 million hours."

The most significant performance bottleneck in today's storage is the HDD itself, limited by seek actuation and rotational latency for any given access, which is worst case when accesses are random distributed small I/Os. Most disk drives can only deliver a few hundred random IOPs and at most around 100 MB/sec for sequential large block access. Aggregating a larger number of drives into a RAID helps so that all actuators can be concurrently delivering I/O or portions of larger block I/O. In general an HDD has a mean time between failure (MTBF) somewhere between 500,000 and 1 million hours, so in large populations (hundreds to thousands of drives) failures will occur on a monthly basis (two or more drives per hundred annually). Furthermore, environmental effects like overheating can accelerate failure rates and failure distributions are not uniform. So, RAID-0 has been enhanced to either stripe and mirror (RAID-10), mirror stripes (RAID-0+1), or add parity blocks every nth drive so data striped on one drive can be recovered from remaining data and parity blocks (RAID-50). Advanced double fault protection error correction code (ECC) schemes like RAID-6 can likewise be striped (RAID-60). So RAID provides some scaling and removes some of the single direct-attached drive bottleneck, but often requires users to buy more capacity than they need just to get better access performance, data loss protection, and reliability. For example, one may have 10 terabytes of data and need gigabyte bandwidth from it with small request sizes (32 K), which requires 32,768 IOPs to achieve 1 GB/sec. If each of the drives in the RAID array can deliver 100 IOPs, I need at least 320 drives! At 500 GB of capacity per drive that is 160 terabytes and I only need 10 terabytes. One common trick to help when more performance is needed from the same capacity is to "short-stroke" drives whereby only the outer diameter of each drive is used which often provides a 25-percent acceleration based on the areal density of the media.

"Virtualization of a collection of drives also requires RAID mapping and presentation of a virtual logical unit number (LUN) or logical disk to an operating system."

Virtualization of a collection of drives also requires RAID mapping and presentation of a virtual logical unit number (LUN) or logical disk to an operating system. This means that all I/O requested from the RAID controller must be re-formed in a RAM buffer and re-initiated to the disk array for the original request. The virtualization makes RAID simple to use and also can handle much of the error recovery protocol (ERP) required for reliable/resilient RAID, but comes at the cost of additional processing, store-and-forward buffering, and I/O channels between the RAID controller, the ultimate user of the RAID system (initiator), and the back-end array of drives. Applications not written with RAID in mind that either do not or cannot initiate multiple asynchronous I/Os often will not get full advantage of the concurrent disk operation offered by large scale RAID. Even with striping, if an application issues one I/O and awaits completion response before issuing the next, full RAID performance will not be realized. As shown in Figure 6, even if each I/O is large enough to stripe all the drives in a RAID set (unlikely for hundreds of drives in large scale RAID), the latency between I/O requests and lack of a queue (backlog) of multiple requests outstanding on the RAID controller will reduce performance.

"A much more ideal system would combine the capacity and performance scaling of RAID along with the performance density scaling of SSD in a hybrid array."

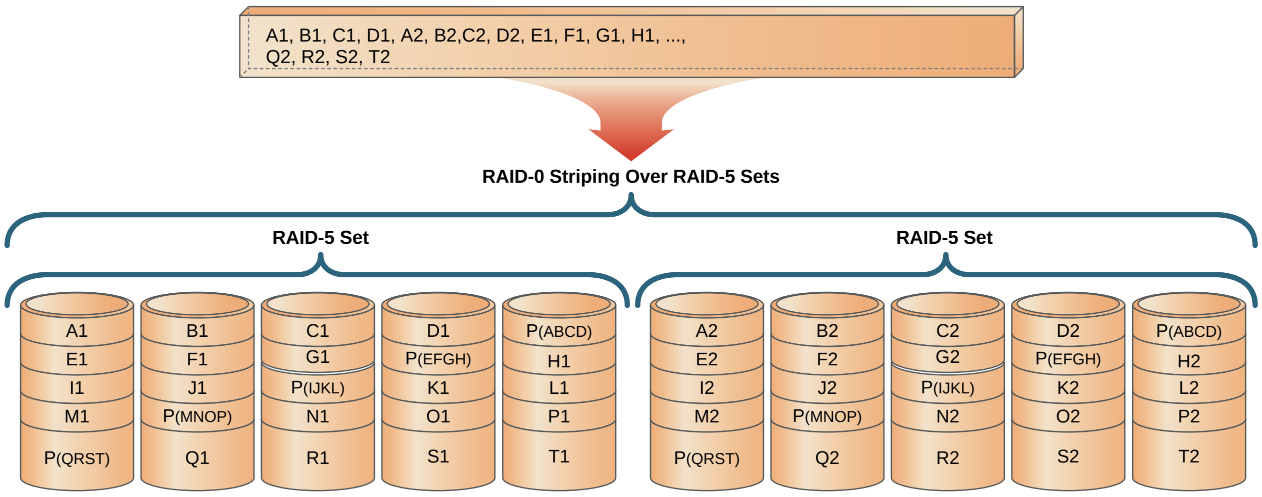


Figure 6: RAID set striping and striding example. Source: Atrato, Inc., 2009

A much more ideal system would combine the capacity and performance scaling of RAID along with the performance density scaling of SSD in a hybrid array so that users could configure a mixture of HDDs and SSDs in one virtualized storage pool. In order to speed up access with 10 terabytes of SSDs, one would have to combine 64 SSD drives into a virtualized array and stripe them with RAID-0. If they wanted data protection with RAID-10 it would increase the number of SSDs to 128. Even with lowering costs, this would be an expensive system compared to an HDD array or hybrid HDD+SSD array.

How SSDs Avoid These Bottlenecks

The bottleneck in embedded systems can be avoided by simply replacing today's HDDs with SSDs. The superior random read (and to a less extent write) provides a tenfold performance increase in general, albeit at ten times the cost per gigabyte. For small scale storage (gigabytes up to a few terabytes) this makes sense since one only pays for the performance increase needed and with no excess capacity. So, for embedded systems, the solution is simple drive replacement, but for larger capacity systems this does not make economic sense. What SSDs bring to larger scale systems is a tier that can be scaled to terabytes so that it can provide a 1-percent to 10-percent cache for 10 to 100 terabytes per RAID expansion unit (or SAID in the case of the Atrato Inc. system). Furthermore, the Intel X25-E and X25-M SATA

"The superior random read provides a tenfold performance increase in general, albeit at ten times the cost per gigabyte."

"An intelligent block-level managed solid-state tier-0 with HDD tier-1 can then accelerate ingest of data to a RAID back-end store, sequential read-out of data from the back-end store, and can serve as a viable cache for the back-end HDD store that is much lower cost than RAM cache."

Solid-State Drive SFF design allows them to be scaled along with the HDD arrays using common SFF drives and protocols. An intelligent block-level managed solid-state tier-0 with HDD tier-1 can then accelerate ingest of data to a RAID back-end store, sequential read-out of data from the back-end store, and can serve as a viable cache for the back-end HDD store that is much lower cost than RAM cache. In the following sections we will look at how SSDs are uniquely positioned to speed up HDD back-end stores geometrically with the addition of intelligent block management and an SSD tier-0.

Tiered Storage Using SSD and High Density HDD Arrays

The tiered approach described in the previous section can be managed at a file level or a block level. At the file level, intelligent users must partition databases and file systems and move data at the file container level based on access patterns for files to realize the speed-up made possible by tiers. Automated block level management using intelligent access pattern analysis software provides an increased level of precision in managing the allocation of data to the SSD tier0 and allows for SSD to be used as an access accelerator rather than a primary store. This overcomes the downside of the cost per gigabyte of SSDs for primary storage and makes optimal use of the performance density and low latency that SSDs have to offer.

Figures 7 through 9 show the potential for a coordinated SSD tier-0 with HDD tier-1 that is managed and virtualized by the Atrato Inc. virtualization engine. Figure 7 shows ingest acceleration through an SLC FIFO. Figure 8 shows sequential read-ahead acceleration through a RAM FIFO that can be combined with an MLC SSD semi-random read cache. The semi-random access SSD read cache has

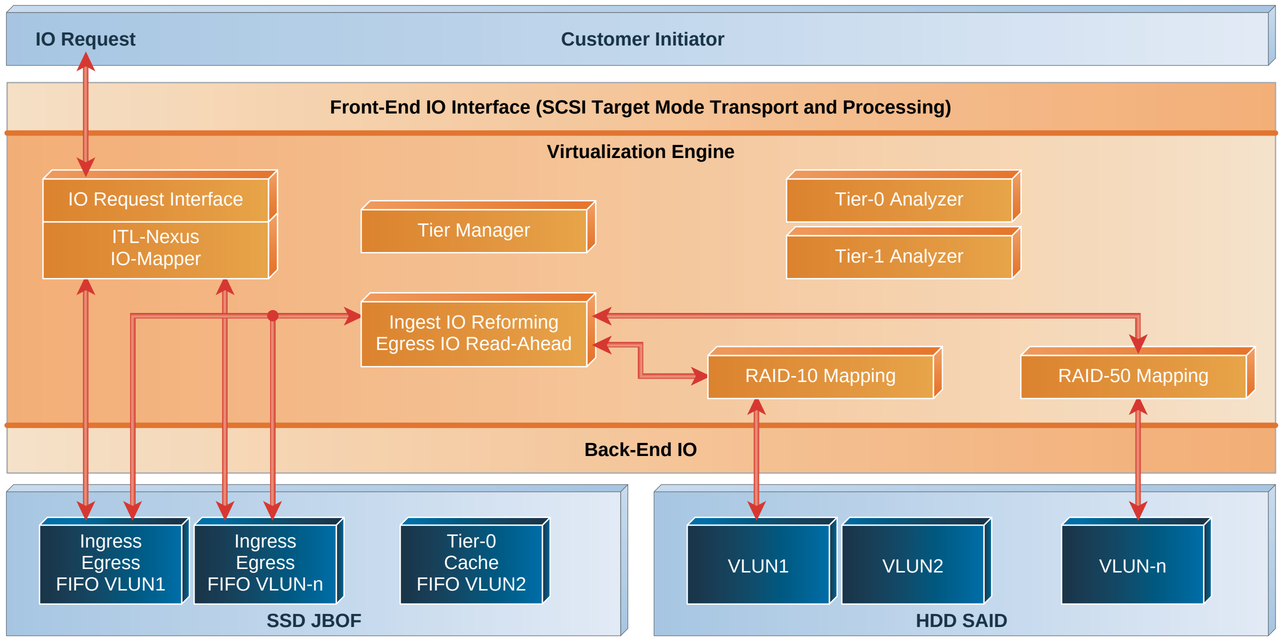


Figure 7: Ingest I/O reforming using SLC SSD and Egress read-ahead RAM cache. Source: Atrato, Inc., 2009

read hit/miss, write-through, and write-back-to-SSD operations. It can also be pre-charged with known high access content during content ingest. Any high access content not pre-charged will be loaded into SSD as this is determined by a TAM (Tier Access Monitor) composed of a Tier block manager and tier-0 and tier-1 access profile analyzers.

Ingest I/O acceleration provides a synergistic use of SSD performance density and low latency so that odd size single I/Os as shown in Figure 8 can be ingested quickly and then more optimally reformed into multiple I/Os for a RAID back-end HDD storage array.

Likewise, for semi-random access to large data stores, SSD provides a tier-0 block cache that is managed by the TAM profile analyzer and intelligent block manager so that the most frequently accessed LBA ranges (hot spots) are always replicated in the SSD tier-0. Figure 9 shows one of the many modes of the intelligent block manager where it replicates a frequently accessed block to the SSD tier-0 on a read I/O—the profile analyzer runs in the I/O path and constantly tracks the most often accessed blocks up to a scale that matches the size of the tier-0.

Overall, Figure 9 shows one mode of the intelligent block manager for write-back-to-SSD on a cache miss and HDD back-end read. The intelligent block manager also includes modes for write-through (during content ingest), read hits, and read misses. These tiered-storage and cache features along with access profiling have been combined into a software package by Atrato Inc. called ApplicationSmart* and overall forms a hybrid HDD and SSD storage operating environment.

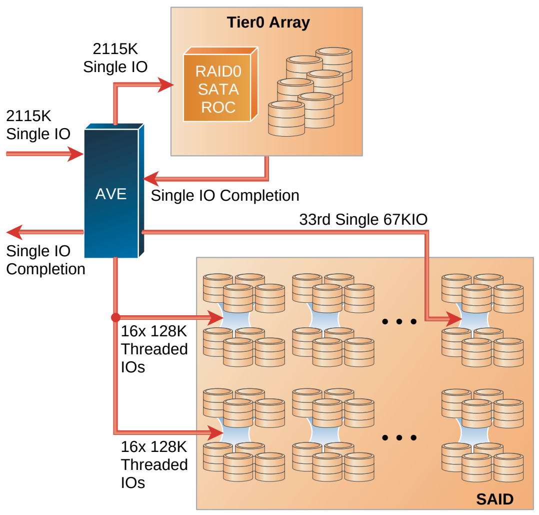


Figure 8: Ingest I/O reforming using SLC SSD and Egress read-ahead RAM cache. Source: Atrato, Inc., 2009

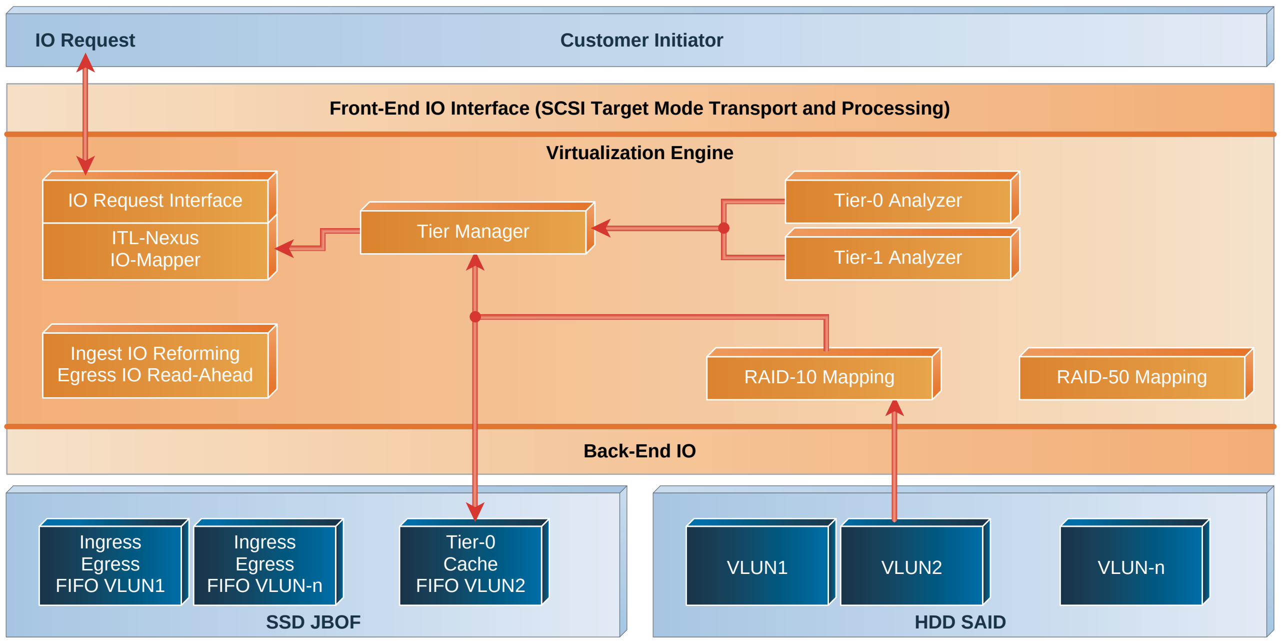


Figure 9: MLC SSD Tier-0 read cache opportunistic load of high access blocks on a read request. Source: Atrato, Inc., 2009

"The ability to scale to petabytes and maintain performance density comparable to SSD alone is the ultimate goal for digital media head-ends, content delivery systems, and edge servers."

This design for hybrid tiered storage with automatic block-level management of the SSD tier-0 ensures that users get maximum value out of the very high performance density SSDs and maximum application acceleration while at the same time being able to scale up to many petabytes of total content. Compared to file-level tiered storage with an SSD tier-0, the block-level tier management is a more optimal and precise use of the higher cost, but higher performance density SSDs.

SSD in Atrato Storage Application

For larger scale systems (tens to hundreds of terabytes up to many petabytes), SSDs are a great option for HDD access acceleration compared to RAM I/O cache due to scalability, persistence features, and cost per gigabyte compared to RAM. The ability to scale to petabytes and maintain performance density comparable to SSD alone is the ultimate goal for digital media head-ends, content delivery systems, and edge servers. As discussed previously, a tiered storage approach is much more efficient than simply adding additional HDDs in large arrays where more performance is needed even though the capacity is not.

Employing MLC Intel X25-M SATA Solid-State Drives as a read cache intelligently managed by the Atrato Inc. ApplicationSmart software and SLC Intel X25-E SATA Solid-State Drives for an ingest FIFO along with a RAM-based egress read-ahead FIFO, Atrato has shown the ability to double, triple, and quadruple performance from an existing V1000 RAID system without adding wasted capacity. Figure 10 shows a range of configurations for the Atrato V1000 with capacity ranging from 80 to 320 terabytes total capacity with SSD tier-0 1RU expansion units for access acceleration. This example was composed assuming the use of an Intel® Microarchitecture, codenamed Nehalem, the dual Intel® X58 Express chipset with off-the-shelf controller, which has at least 64 lanes of gen2 PCI-Express* and 8 total PCI-Express slots, 4 of which can be used for back-end SAID/SSD I/O and 4 of which can be used for front-end SAN or VOD transport I/O.

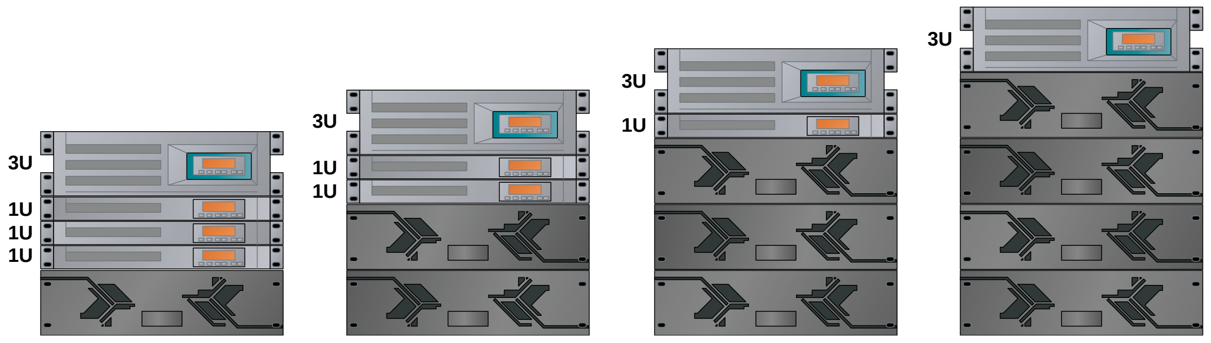


Figure 10: Scaling of SAIDs and SSD expansion units for access acceleration. Source: Atrato, Inc., 2009

There are 12 potential configurations that will allow customers to "dial in" the capacity and performance needed. Table 1 summarizes the configurations and the speed-up provided by SSD tier expansion units.

#SAID, #SSD Units	SSD Read Cache (TBs)	SSD ingest, egress (TBs)	BW (GBps)	IOPs	Capacity (TBs)	Cost, Capacity, Performance Normalized Score
4, 0	0	0	5.6	64000	320	2.4
3, 1	1.6	0.896	5.7	102000	240	2.4
2, 2	3.2	0.896	5.8	140000	160	2.3
3, 0	0	0	4.2	48000	240	2.2
2, 1	1.6	0.896	4.3	86000	160	2.1
1, 3	4.8	0.896	5.9	178000	80	2.1
2, 0	0	0	2.8	32000	160	2.0
1, 2	3.2	0.896	4.4	124000	80	1.9
1, 1	1.6	0.896	2.9	70000	80	1.8
1, 0	0	0	1.4	16000	80	1.7
0, 4	6.4	0.896	6	216000	6.4	1.2
0, 3	4.8	0.896	4.5	162000	4.8	1.0
0, 2	3.2	0.896	3	108000	3.2	0.7
0, 1	1.6	0.896	1.5	54000	1.6	0.2

Table 1: Cost, capacity, performance tradeoffs for SSD and HDD expansion units. Source: Atrato, Inc., 2009

Looking at a chart of the cost-capacity-performance (CCP) scores and total capacity, this would allow a customer to choose a hybrid configuration that has the best value and does not force them to purchase more storage capacity than they need (nor the power and space to host it). The CCP scores are composed of average cost per gigabyte, capacity density, and equally valued IOPs and bandwidth in performance, with equal weight given to each category so that a maximum possible score was 3.0. As can be seen in Figure 10 and in Table 1, if one needs between 100 and 200 terabytes total capacity, a 2 SAID + 2 SSD Expansion Unit configuration would be optimal. Furthermore, this would deliver performance that would exceed 4 SAIDs assuming that the access pattern is one that can cache 3.2 terabytes of the most frequently accessed blocks out of 160 terabytes (2-percent cache capacity).

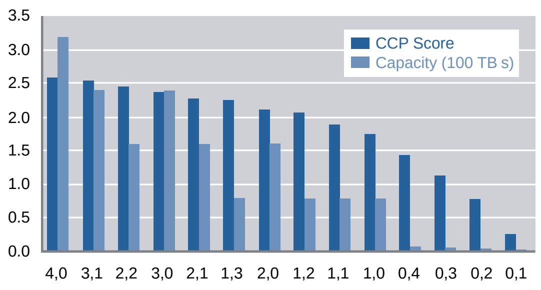


Figure 11: Cost, capacity, and performance score tradeoff. Source: Atrato, Inc., 2009

"Looking at a chart of the cost-capacity-performance (CCP) scores and total capacity, this would allow a customer to choose a hybrid configuration that has the best value."

"In general, storage I/O is such that I/Os can complete out of order and there are rarely data dependencies like there might be in a CPU cache."

Computing the value of a read cache is tricky and requires a good estimation of the hit/miss ratio and the miss penalty. In general, storage I/O is such that I/Os can complete out of order and there are rarely data dependencies like there might be in a CPU cache. This means the penalty is fairly simple and not amplified as it might be when CPU cache causes a CPU pipeline to stall. A miss most often simply means an extra SAID back-end I/O and one less tier-0 I/O. The Atrato ApplicationSmart algorithm is capable of quickly characterizing access patterns, detecting when they change, and recognizing patterns seen in the past. The ApplicationSmart Tier-Analyzer simply monitors, analyzes, and provides a list of blocks to be promoted (most frequently accessed) from the back-end store and provides a list of blocks to be evicted from the tier-0 (least frequently accessed in cache). This allows the intelligent block manager to migrate blocks between tiers as they are accessed through the Atrato virtualization engine in the I/O path.

Figure 12 shows a test access pattern and Figure 13 shows the sorted test access pattern. As long as the most frequently accessed blocks fit into the tier-0, speed-up can be computed based on total percentage access to SSD and total percentage access to the back-end HDD storage. The equations for speed-up from SSD tier-0 replication of frequently accessed blocks are summarized here:

$$tier0\_LBA\_size = \left(\sum_{i=0}^{(sizeof\_sorted\_access\_counts-1)} evaluate\,(sorted\_access\_counts[i] > 0) \right) \times LBA\_set\_size$$

$$tier0\_hosted\_IOs = \sum_{i=0}^{(tier\,0\_LBA\_sets-1)} sorted\_access\_counts[i]$$

$$total\_sorted\_IOs = \sum_{i=0}^{(sizeof\_sorted\_access\_counts-1)} evaluate\,(sorted\_access\_counts[i]$$

$$tier0\_access\_fit = \frac{tier0\_hosted\_IOs}{total\_sorted\_IOs}$$

$$hit\_rate = tier0\_access\_fit \times tier0\_efficiency = 1.0 \times 0.6$$

$$speed\_up = \frac{T_{HDD\_only}}{T_{SSD\_hit} + T_{HDD\_miss}}$$

$$speed\_up = \frac{ave\_HDD\_latency}{(hit\_rate \times ave\_SSD\_latency) + ((1 - hit\_rate) \times ave\_HDD\_latency)}$$

$$speed\_up = \frac{1000\,\mu\,\text{sec}}{(0.6 \times 800\,\mu\,\text{sec}) + (0.4 \times 10000\,\mu\,\text{sec})}$$

In the last equation, if we assume average HDD latency is 10 milliseconds (10,000 microseconds) and SSD latency for a typical I/O (32 K) is 800 micro-seconds, then with a 60-percent hit rate in tier-0 and 40-percent access rate on misses to the HDD storage, the speed-up is 2.1 times. As seen in Figure 12, we can organize the semi-random access pattern using ApplicationSmart so that 4000 of the most frequently accessed regions out of 120,000 total (3.2 terabytes of SSD and 100 terabytes of HDD back-end storage) can be placed in the tier-0 for a speed-up of 3.8 with an 80-percent hit rate in tier-0.

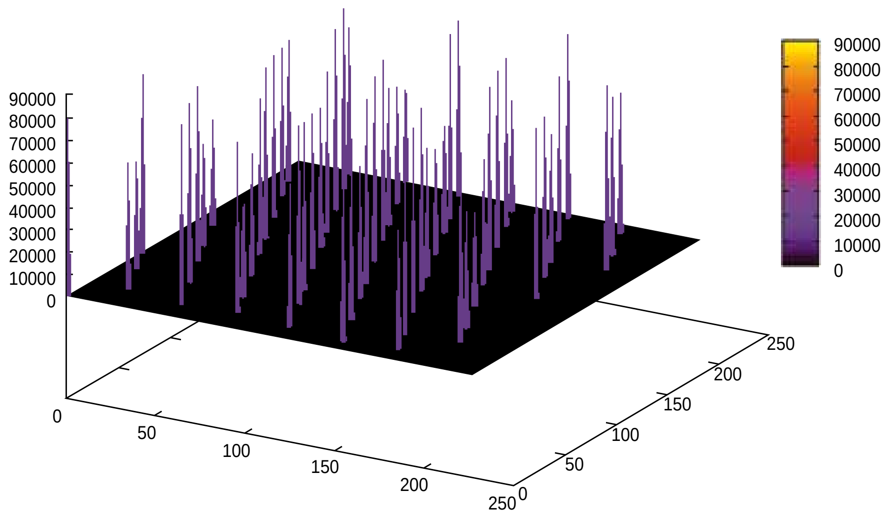


Figure 12: Predictable I/O access pattern seen by ApplicationSmart Profiler. Source: Atrato, Inc., 2009

Figure 13 shows the organized (sorted) LBA regions that would be replicated in tier-0 by the intelligent block manager. The graph on the left shows all nonzero I/O access regions (18 x 16 = 288 regions). The graph on the right shows those 288 re-gions sorted by access frequency. Simple inspection of these graphs shows us that if we replicated the 288 most frequently accessed regions, we could satisfy all I/O requests from the faster tier-0. Of course the pattern will not be exact over time and will require some dynamic recovery, so with a changing access pattern, even with active intelligent block management we might have an 80-percent hit rate. The intelligent block manager will evict the least accessed regions from the tier-0 and replace them with the new most frequently accessed regions over time. So the algorithm is adaptive and resilient to changing access patterns.

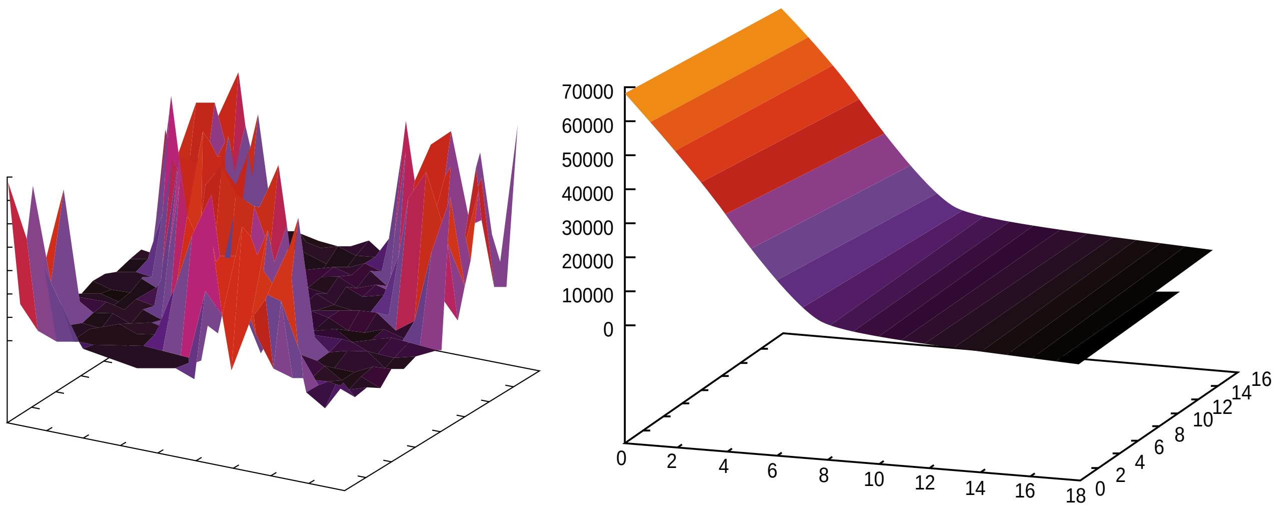


Figure 13: Sorted I/O access pattern to be replicated in SSD Tier-0. Source: Atrato, Inc. 2009

In general, the speed-up can be summarized as shown in Figure 14, where in the best case the speed-up is the relative performance advantage of SSD compared to HDD, and otherwise scaled by the hit/miss ratio in tier-0 based on how well the intelligent block manager can keep the most frequently accessed blocks in tier-0 over time and based on the tier-0 size.

It can clearly be seen that the payoff for intelligent block management is nonlinear and while a 60-percent hit rate results in a double speed-up, a more accurate 80-percent provides triple speed-up.

The ingest acceleration is much simpler in that it requires only an SLC SSD FIFO where I/Os can be ingested and reformed into more optimal well-striped RAID I/Os on the back-end. As described earlier, this simply allows applications that are not written to take full advantage of RAID concurrent I/Os to enjoy speed-up through the SLC FIFO and I/O reforming. The egress acceleration is an enhancement to the read cache that provides a RAM-based FIFO for read-ahead LBAs that can be burst into buffers when a block is accessed where follow-up sequential access in that same region is likely. These features bundled together as ApplicationSmart along with SSD hardware are used to accelerate access performance to the existing V1000 without adding more spindles.

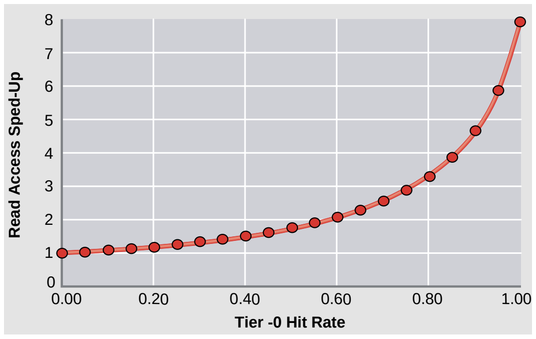


Figure 14: I/O access speed-up with hit rate for tier-0. Source: Atrato, Inc., 2009

Overview of Atrato Solution

The Atrato solution is overall an autonomic application-aware architecture that provides self-healing disk drive automation [9] and self-optimizing performance with ApplicationSmart profiling and intelligent block management between the solid-state and SAID-based storage tiers as described here and in an Atrato Inc. patent [1].

Related Research and Storage System Designs

The concept of application aware storage has existed for some time [2] and in fact several products have been built around these principles (Bycast StorageGRID, IBM Tivoli Storage Manager, Pillar Axiom). The ApplicationSmart profiler, Intelligent Block Manager and Ingest/Egress Accelerator features described in this article provide a self-optimizing block-level solution that recognizes how applications access information and determines where to best store and retrieve that data based on those observed access patterns. One of the most significant differences between the Atrato solution and others is the design of the ApplicationSmart algorithm for scaling to terabytes of tier-0 (solid-state storage) and petabytes of tier-1 (HDD storage) with only megabytes of required RAM meta-data to do so. Much of the application-aware research and system designs have been focused on distributed hierarchies [4] and information hierarchy models with user hint interfaces to gauge file-level relevance. Information lifecycle management (ILM) is closely related to application-aware storage and normally focuses on file-level access, age, and relevance [7] as does hierarchical storage management (HSM), which uses similar techniques, but with the goal to move files to tertiary storage (archive) [5][9][10]. In general, block-level management is more precise than file-level, although the block-level ApplicationSmart features can be combined with file-level HSM or ILM since it is focused on replicating highly accessed, highly relevant data to solid-state storage for lower latency (faster) more predictable access. Ingest RAM-based cache for block level read-ahead is used in most operating systems as well as block-storage devices. Ingest write buffering is employed in individual disk drives as well as virtualized storage controllers (with NVRAM or battery-backed RAM). Often these RAM I/O buffers will also provide block-level cache and employ LRU (Least Recently Used) and LFU (Least Frequently Used) algorithms. However, for a 35-TB formatted LUN, this would require 256 GB of RAM to track LRU or LFU for LBA cache sets of 1024 LBAs each or an approximation of LRU/LFU–these traditional algorithms simply do not scale well. Furthermore, as noted in [9] the traditional cache algorithms are not precise or adaptive in addition to requiring huge amounts of RAM for the LRU/LFU meta-data compared to ApplicationSmart.

"One of the most significant differences between the Atrato solution and others is the design of the ApplicationSmart algorithm for scaling to terabytes of tier-0 (solid-state storage) and petabytes of tier-1 (HDD storage) with only megabytes of required RAM meta-data to do so."

Architecture

The Atrato solution for incorporating SSD into high capacity, high performance density solutions that can scale to petabytes includes five major features:

- Ability to profile I/O access patterns to petabytes of storage using megabytes of RAM with a multi-resolution feature-vector-analysis algorithm to detect pattern changes and recognize patterns seen in the past.
- Ability to create an SSD VLUN along with traditional HDD VLUNs with the same RAID features so that file-level tiers can be managed by applications.
- Ability to create hybrid VLUNs that are composed of HDD capacity and SSD cache with intelligent block management to move most frequently accessed blocks between the tiers.
- Ability to create hybrid VLUNs that are composed of HDD capacity and are allocated SLC SSD ingest FIFO capacity to accelerate writes that are not well-formed and/or are not asynchronously and concurrently initiated.
- Ability to create hybrid VLUNs that are composed of HDD capacity and allocated RAM egress FIFO capacity so that the back-end can burst sequential data for lower latency sequential read-out.

"Often these RAM I/O buffers will also provide block-level cache and employ LRU (Least Recently Used) and LFU (Least Frequently Used) algorithms. These traditional algorithms simply do not scale well."

"In general, this algorithm easily profiles down to a single VoD 512-K block size using one millionth the RAM capacity for the HDD capacity it profiles."

With this architecture, the access pattern profiler feature allows users to determine how random their access is and how much an SSD tier along with RAM egress cache will accelerate access using the speed-up equations presented in the previous section. It does this by simply sorting access counts by region and by LBA cache-sets in a multi-level profiler in the I/O path. The I/O path analysis uses an LBA-address histogram with 64-bit counters to track number of I/O accesses in LBA address regions. The address regions are divided into coarse LBA bins (of tunable size) that divide total useable capacity into 256-MB regions (as an example). If, for example, the SSD capacity is 3 percent of the total capacity (for instance, 1 terabyte (TB) of SSD and 35 TB of HDD), then the SSDs would provide a cache that replicates 3 percent of the total LBAs contained in the HDD array. As enumerated below, this would require 34 MB of RAM-based 64-bit counters (in addition to the 2.24 MB course 256-MB region counters) to track access patterns for a useable capacity of 35 TB. In general, this algorithm easily profiles down to a single VoD 512-K block size using one millionth the RAM capacity for the HDD capacity it profiles. The hot spots within the highly accessed 256-MB regions become candidates for content replication in the faster access SSDs backed by the original copies on HDDs. This can be done with a fine-binned resolution of 1024 LBAs per SSD cache set (512 K) as shown in this example calculation of the space required for a detailed two-level profile.

- Useable capacity for a RAID-10 mapping with 12.5 percent spare regions
  - Example: (80 TB – 12.5 percent)/2 = 35 TB, 143360 256-MB regions, 512-K LBAs per region
- Total capacity required for histogram
  - 64-bit counter per region
  - Array of structures with {Counter, DetailPtr}
  - 2.24 MB for total capacity level 1 histogram
- Detail level 2 histogram capacity required
  - Top X%, Where X = (SSD_Capacity/Useable_Capacity) x 2 have detail pointers with 2x over-profiling
  - Example: 3 percent, 4300 detail regions, 8600 to 2x oversample
  - 1024 LBAs per cache set, or 512 K
  - Region_size/LBA_set_size = 256 MB/512 K = 512 64-bit detail counters per region
  - 4 K per detail histogram x 8600 = 34.4 MB

"Feature vector analysis mathematics is employed to determine when access patterns have changed significantly."

With the two-level (coarse region level and fine-binned) histogram, feature vector analysis mathematics is employed to determine when access patterns have changed significantly. This computation is done so that the SSD block cache is not re-loaded too frequently (cache thrashing). The proprietary mathematics for the ApplicationSmart feature-vector analysis is not presented here, but one should understand how access patterns change the computations and indicators.

When the coarse region level histogram changes (checked on a tunable periodic basis) as determined by ApplicationSmart ΔShape, a parameter that indicates the significance of access pattern change, then the fine-binned detail regions may be either re-mapped (to a new LBA address range) when there are significant changes in the coarse region level histogram to update detailed mapping, or when change is less significant this will simply trigger a shape change check on already existing detailed fine-binned histograms. The shape change computation reduces the frequency and amount of computation required to maintain access hot-spot mapping significantly. Only when access patterns change distribution and do so for sustained periods of time will re-computation of detailed mapping occur. The trigger for remapping is tunable through the ΔShape parameters along with thresholds for control of CPU use, to best fit the mapping to access pattern rates of change, and to minimize cache thrashing where blocks replicated to the SSD. The algorithm in ApplicationSmart is much more efficient and scalable than simply keeping 64-bit counters per LBA and allows it to scale to many petabytes of HDD primary storage and terabytes of tier-0 SSD storage in a hybrid system with modest RAM requirements.

"Only when access patterns change distribution and do so for sustained periods of time will re-computation of detailed mapping occur."

Performance

Performance speed-up using ApplicationSmart is estimated by profiling an access pattern and then determining how stable access patterns perform without addition of SSDs to the Atrato V1000. Addition of SLC for write ingest acceleration is always expected to speed-up writes to the maximum theoretical capability of the V1000 since it allows all writes to be as perfectly re-formed as possible with minimal response latency from the SLC ingest SSDs. Read acceleration is ideally expected to be equal to that of a SAID with each 10 SSD expansion unit added as long as sufficient cache-ability exists in the I/O access patterns. This can be measured and speed-up with SSD content replication cache computed (as shown earlier) while customers run real workloads. The ability to double performance using 8 SSDs and one SAID was shown compared to one SAID alone during early testing at Atrato Inc. Speed-ups that double, triple, and quadruple access performance are expected.

"Atrato Inc. has been working with Intel X25-M and Intel® X25-E Solid-State Drives since June of 2008."

SSD Testing at Atrato

Atrato Inc. has been working with Intel X25-M and Intel® X25-E Solid-State Drives since June of 2008 and has tested hybrid RAID sets, drive replacement in the SAID array, and finally decided upon a hybrid tiered storage design using application awareness with the first alpha version demonstrated in October 2008, a beta test program in progress this March, and release planned for the second quarter of 2009.

SSDs Make a Difference

Atrato Inc. has tested SSDs in numerous ways including hybrid RAID sets where an SSD is used as the parity drive in RAID-4, simple SSD VLUNs with user allocation of file system metadata to SSD and file system data to HDD in addition to the five features described in the previous sections. Experimentation showed that the most powerful uses of hybrid SSD and HDD are for ingest/egress FIFOs, read cache based on access profiles, and simple user specification of SSD VLUNs. The Atrato design for ApplicationSmart uses SSDs such that access performance improvement is considerable for ingest, for semi-random read access, and for sequential large

"Experimentation showed that the most powerful uses of hybrid SSD and HDD are for ingest/egress FIFOs, read cache based on access profiles, and simple user specification of SSD VLUNs."

"Atrato Inc. has found the Intel X25-E and Intel X25-M SATA Solid-State Drive integrate well with HDD arrays given the SATA interface."

block predictable access. In the case of totally random small transaction I/O that is not cache-able at all, the Atrato design recognizes this with the access profiler and offers users the option to create an SSD VLUN or simply add more SAIDs that provide random access scaling with parallel HDD actuators. Overall, SSDs are used where they make the most difference and users are able to understand exactly the value the SSDs provide in hybrid configurations (access speed-up).

Conclusions Made about Intel SSDs

Atrato Inc. has found the Intel X25-E and Intel X25-M SATA Solid-State Drive integrate well with HDD arrays given the SATA interface, which has scalability through SAS/SATA controllers and JBOF* (Just a Bunch of Flash*). The Intel SSDs offer additional advantages to Atrato including SMART data for durability and life expectancy monitoring, write ingest protection, and ability to add SSDs as an enhancing feature to the V1000 rather than just as a drive replacement option.

Atrato Inc. plans to offer ApplicationSmart with Intel X25-E and X25-M SATA Solid-State Drives as an upgrade to the V1000 that can be configured by customers according to optimal use of the SSD tier.

"The Intel X25-E SATA Solid-State Drives provide ingest acceleration at lower cost and with greater safety than RAM ingest FIFOs."

Future Atrato Solution Using SSDs

The combination of well managed hybrid SSD+HDD is synergistic and unlocks the extreme IOPs capability of SSD along with the performance and capacity density of the SAID enabled by intelligent block management.

Issues Overcome by Using SSDs

Slow write performance to the Atrato V1000 has been a major issue for applications not well-adapted to RAID and could be solved with a RAM ingest FIFO. However this presents the problem of lost data should a power failure occur before all pending writes can be committed to the backing-store prior to shutdown. The Intel X25-E SATA Solid-State Drives provide ingest acceleration at lower cost and with greater safety than RAM ingest FIFOs. Atrato needed a cost-effective cache solution for the V1000 that could scale to many terabytes and SSDs provide this option whereas RAM does not.

"For customers that need for example 80 terabytes total capacity, the savings with SSD is significant."

Performance Gained by Using Intel SSD

The performance density gains will vary by customer and their total capacity requirements. For customers that need for example 80 terabytes total capacity, the savings with SSD is significant since this means that 3 1RU expansion units can be purchased instead of 3 more 3RU SAIDs and another 240 terabytes of capacity that aren't really needed just to scale performance. This is the best solution for applications that have cache-able workloads, which can be verified with the Atrato ApplicationSmart access profiler.

Future Possibilities Opened Due to Intel SSDs

Future architectures for ApplicationSmart include scaling of SSD JBOFs with SAN attachment using Infiniband or 10G iSCSI such that the location of tier-0 storage and SAID storage can be distributed and scaled on a network in a general fashion giving customers even greater flexibility. The potential for direct integration of SSDs into SAIDs in units of 8 at a time or in a built-in expansion drawer is also being investigated. ApplicationSmart 1.0 is in beta testing now with a planned release for May 2009.

"The potential for direct integration of SSDs into SAIDs in units of 8 at a time or in a built-in expansion drawer is also being investigated."

Conclusion

Using Intel® Solid State Drive (Intel® SSD) for Hybrid Arrays

The Intel X25-E SATA Solid-State Drive provides a cost effective option for hybrid arrays with an SSD-based tier-0. As an example, Atrato has been able to integrate the Intel X25-E SATA Solid-State Drives in the V1000 tier-0 and with the overall virtualization software for the SAID so that performance can be doubled or even quadrupled.

A New Storage and Caching Subsystem

The use of RAM cache for storage I/O is hugely expensive and very difficult to scale given the cost as well as the complexity of scalable memory controllers like FB-DIMM or R-DIMM beyond terabyte scale. Solid state drives are a better match for HDDs, while being an order of magnitude faster for random IOPs and providing the right amount of additional performance for the additional cost, providing for easily justifiable expense to obtain comparable application speed-up.

SSDs for Multiple Embedded Storage Needs

The use of SSDs as drive replacements in embedded applications is inevitable and simple. On the small scale of embedded digital cameras and similar mobile storage devices, SSDs will meet a growing need for high performance, durable, low power direct-attach storage. For larger scale RAID systems, SSDs in hybrid configurations meet ingest, egress, and access cache needs far better than RAM and at much lower cost. Until SSD cost per gigabyte reaches better parity with HDD, which may never happen, hybrid HDD+SSD is here to stay, and many RAID vendors will adopt tiered SSD solutions given the cost/benefit advantage.

"Until SSD cost per gigabyte reaches better parity with HDD, which may never happen, hybrid HDD+SSD is here to stay, and many RAID vendors will adopt tiered SSD solutions given the cost/benefit advantage."

Acknowledgements

Nick Nielsen (Senior Software Engineer), Phillip Clark (Senior Software Engineer), Lars Boehenke (Software Engineer), Louis Morrison (Senior Electrical Design Engineer), and the entire Atrato, Inc. team who have all contributed to the ApplicationSmart software and integration of solid state disks with the V1000 intelligent RAID system.

References

[1] "Systems and Methods for Block-Level Management of Tiered Storage," US Patent Application # 12/364,271, February, 2009.

[2] "Application Awareness Makes Storage More Useful," Neal Leavitt, IEEE Computer Society, July 2008.

[3] "Flash memories: Successes and challenges," S.K. Lai, IBM Journal of Research and Development, Vol. 52, No. 4/5, July/September, 2008.

[4] "Galapagos: Model driven discovery of end-to-end application-storage relationships in distributed systems," K. Magoutis, M. Devarakonda, N. Joukov, N.G. Vogl, IBM Journal of Research and Development, Vol. 52, No. 4/5, July/September, 2008.

[5] "Hierarchical Storage Management in a Distributed VOD System," David W. Brubeck, Lawrence A. Rowe, IEEE MultiMedia, 1996.

[6] "Storage-class memory: The next storage system technology," R.F. Freitas, W.W. Wilcke, IBM Journal of Research and Development, Vol. 52, No. 4/5, July/September, 2008.

[7] "Information valuation for Information Lifecycle Management," Ying Chen, Proceedings of the Second International Conference on Autonomic Computing, September, 2005.

[8] "File classification in self-* storage systems," M. Mesnier, E. Thereska, G.R. Ganger, D. Ellard, Margo Seltzer, Proceedings of the First International Conference on Autonomic Computing, May, 2004.

[9] "Atrato Design for Three Year Zero Maintenance," Sam Siewert, Atrato Inc. White Paper, March 2008.

Author Biographies

Dr. Sam Siewert: Dr. Sam Siewert is the chief technology officer (CTO) of Atrato, Inc. and has worked as a systems and software architect in the aerospace, telecommunications, digital cable, and storage industries. He also teaches as an Adjunct Professor at the University of Colorado at Boulder in the Embedded Systems Certification Program, which he co-founded in 2000. His research interests include high-performance computing and storage systems, digital media, and embedded real-time systems.

Dane Nelson: Dane Nelson is a field applications engineer at Intel Corporation. He has worked in multiple field sales and support roles at Intel for over 9 years and is currently a key embedded products field technical support person for Intel's line of solid state drives.

Copyright

FANLESS DESIGN FOR EMBEDDED APPLICATIONS

Contributors

Chun Howe Sim
Intel Corporation

Jit Seng Loh
Intel Corporation

Index Words

Fanless
computational fluid dynamics
Grashof number
Rayleigh number
natural convection JEDEC 51-2 standard
Point Of Sale
Optimal plate fin spacing

Abstract

Embedded systems opportunities for Intel® architecture components exist in point-of-sale, digital signage, and digital security surveillance, to name a few. When selecting Intel architecture, several key metrics are performance/watt, thermal design power (TDP), and fanless thermal solutions. The objective of this article is to provide readers with key reference fanless system design considerations to utilize in embedded applications. This article emphasizes analytical hand calculation for first-order approximations and provides computational fluid dynamics (CFD) simulation techniques to determine Intel architecture feasibility in fanless systems. Examples depicted illustrate fanless cooling design considerations for a point-of-sale system.

Introduction

In markets for embedded systems, customers usually are looking for small form factors, low cost, high reliability and low power. The Embedded and Communications Group (ECG) within Intel Corporation addresses these specific needs for different embedded market segments, offering a wide range of products from performance to ultra low power to system on chip (SOC) solutions. Ultra low power solutions are often considered by many customers in fanless applications: examples include, point-of-sale terminals, digital signage, in-vehicle infotainment, and digital security surveillance.

For obvious reasons, fanless applications are getting more and more attention; simply adopting the currently available heatsink is no longer feasible. A clear understanding of natural convection heat transfer and how this theory can be applied to component level and system level thermal solution design is crucial. This article provides a reference for designing a fanless heatsink solution for a low voltage Intel® Architecture Processor.

"A clear understanding of natural convection heat transfer and how this theory can be applied to component level and system level thermal solution design is crucial."

This article is divided into three main sections, starting with an analytical hand calculation to approximate an optimum fin spacing of a heatsink for a natural convection heat transfer, and then using industry standards in component level numerical simulation, applied design on experiment (DOE) to determine natural convection heatsink with optimal plate fin spacing. The final section is a system level computational fluid dynamics (CFD) analysis where a printed circuit board (PCB) form factor, component placement, and chassis vent holes are highlighted in the design considerations.

Thermal Solution Design (Analytic)

Hand calculation uses fluid dynamics, heat, and mass transfer fundamental theories to derive thermal solution design equations.

Natural Convection Theory

Natural convection, also known as free convection and a more commonly marketing term fanless, is a sub-classification of convection heat transfer. Unlike forced convection which is caused by external means (fans, pumps, or atmospheric winds), natural convection airflow is induced by buoyancy forces: a result of density differences caused by temperature variation in the fluid. In the semiconductor industry, most of the time air is the "fluid" unless otherwise specified. For additional information on natural convection theory, please refer to references [1] and [2].

"Natural convection airflow is induced by buoyancy forces: a result of density differences caused by temperature variation in the fluid."

Apart from natural convection, another major heat dissipating mode in natural convection is radiation heat transfer. Analytical hand calculation on heatsink radiation is comprehensive and complex. This article does not derive equations of radiation where details of emissivity and absorptivity between components, wavelength, components geometry, and transmissivity angle are required in the study; rather we utilize computational fluid dynamics (CFD) to calculate components heat radiation; inputs of components geometry, material properties and surface finishing are needed for the computation. For further reading on radiation (analytic) please refer to Chapter 12 in reference [1].

In natural convection, where the velocity of moving air is unknown, no single velocity is analogous to the free stream velocity that can be used to characterize the flow. The Reynolds number (Re) is usually used when performing dimensional analysis of fluid dynamics problems. For example it is used to characterize different flow regimes such as laminar or turbulent flow in a circular pipe. Thus, one cannot use the Reynolds number in the computation. Instead, the use of the Grashof number (Gr) and Rayleigh number (Ra) to correlate natural convection flows and heat transfer is recommended. Grashof number is defined as follows:

"The use of the Grashof number (Gr) and Rayleigh number (Ra) to correlate natural convection flows and heat transfer is recommended."

$$Gr = \frac{g \beta \rho^2 (T_S - T_f) L^3}{\mu^2} = \frac{Ra}{\mathrm{Pr}} \qquad (1)$$

where:

g = acceleration of gravity (m/s^2)	L = characteristic length (m)
β = volume expansivity (1/K)	μ = viscosity of fluid (Ns/m^2)
ρ = density of fluid (kg/m^3)	Ra = Rayleigh number
T_S = surface temperature (K)	Pr = Prandtl number
T_f = fluid temperature (K)	

The Grashof number is a dimensionless number that approximates the ratio of the buoyancy to viscous force acting on a fluid.

The Rayleigh number for a fluid is a dimensionless number that is associated with the ratio of buoyancy driven flow and thermal momentum diffusivities. The Rayleigh number is the product of the Grashof number and the Prandtl number, which will be used in the later section to determine optimal plate fin spacing.

In summary, for natural convection airflow characterization use the Grashof number and for force convection airflow characterization use the Reynolds number. For natural convection airflow characterization with heat transfer use the Rayleigh number.

Volumetric Expansivity

Volumetric expansivity of a fluid provides a measure of the amount the density changes in response to a change in temperature at constant pressure. In most cases, a specific desirable volumetric expansivity of an application requires lab testing. This article does not emphasize the lab testing but rather uses the Ideal Gas Law to compute β for air.

$$\beta = \frac{1}{T_f} \qquad (2)$$

"The optimal plate fin spacing is needed to allow airflow between fins to stream as freely as possible from the heatsink base to the outer edge of heatsink fins."

where T_f in the Ideal Gas Law must be expressed on an absolute scale (Kelvin or Rankine). For more information please see reference [2].

Substituting Equation 1 into Equation 2 becomes

$$Gr = \frac{g\rho^2 (T_S - T_f) L^3}{T_f \mu^2} \qquad (3)$$

Converting the Grashof number to the Rayleigh number, Equation 3 becomes

$$Ra = Gr\,Pr = \frac{g\rho^2 (T_S - T_f) L^3 Pr}{T_f \mu^2} \qquad (4)$$

Optimized Plate Fins Spacing

Determining optimal plate fin spacing of a natural convection small-form-factor heatsink is an effective way to improve heatsink performance. Recapping the natural convection theory section, natural convection occurs mainly due to buoyancy forces; the optimal plate fin spacing is needed to allow airflow between fins to stream as freely as possible from the heatsink base to the outer edge of heatsink fins. Convection heat transfer is optimized when the velocity boundary layer developed by buoyancy force and optimal plate fin spacing equals the thermal boundary layer developed by the plate fins. In steady state condition, where heatsink temperature reaches equilibrium, for the scope of this article, design engineers could assume that each plate fin is close to isothermal (ΔT across fin surface equals to zero). Then a first-order approximation of optimal plate fin spacing can be defined with a known heatsink volume (W x D x H) as

$$S = 2.714 \frac{L}{(Ra)^{1/4}} \tag{5}$$

where:

S = optimum fin spacing

L = fin length parallel to airflow direction

Ra = Rayleigh number

Knowing Ra from Equation 4, we can now substitute it into Equation 5.

$$S = 2.714 \frac{L(T_f \mu^2)^{1/4}}{\{g\rho^2(T_s - T_f)L^3 \,\mathrm{Pr}\}^{1/4}} \tag{6}$$

For more information on optimum fin spacing please see reference [7].

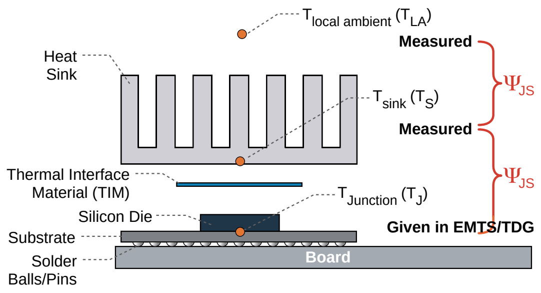


Figure 1: Component level bare die package thermal solution stackup.
Source: Intel Corporation, 2009

Bare Die Type Package Natural Conveciton Thermal Solution Stackup

An Intel central processing unit (CPU) mainly consists of a bare die type package and integrated heat spreader (IHS) type package. In this section, we focus on component level bare die type package and its natural convection thermal solution stackup shown in Figure 1. A natural convection thermal solution consists of a heatsink, thermal interface material (TIM), and fastening mechanism.

This is a typical 2D reference picture showing the three main temperature measurement points. These temperature measurement points are used to compute thermal performance of a heatsink. They are as follows:

T_J is the junction temperature; it is the temperature of the hottest spot at silicon die level in a package.

T_S is the heatsink temperature; it is the temperature of the center-bottom surface of the thermal solution base. One has to machine the thermal solution base per Intel specification for zero degree thermocouple attachment method and measure TS. For more information please see reference [10].

T_{LA} is the local ambient temperature measurement within the system boundary. For natural convection T_{LA} point is located at the side of the thermal solution, approximately 0.5”–1.0” away. It is recommended to use average T_{LA} from a few T_{LA} measurement points. For more details on exact measurement and location point please see references [4] and [5].

Thermal Performance Characterization for Bare Die

Thermal performance and thermal impedance are often confused and loosely used in the industry. Thermal performance (ψ) is an industrial standard to characterize heatsink cooling performance. This is a basic thermal engineering parameter that is used to evaluate and compare different heatsinks. Thermal performance from junc-

“Thermal performance (ψ) is an industrial standard to characterize heatsink cooling performance.”

"The PDF is Intel's power density factor and is available upon request through a field application engineer (FAE)."

tion to ambient is a sum of thermal impedance of silicon die, TIM, and heatsink as shown in Equation 7.

$$\psi_{JA} = \psi_{JC} + \psi_{CS} + \psi_{SA} \tag{7}$$

The ψ_{JC} value can be obtained from the chipset/processor manufacturer datasheet. The ψ_{SA} value is available through the heatsink vendors' support collateral. For custom designs, CFD and/or lab experimentation determine ψ_{SA} value. For more information on ψ_{JA} and ψ_{SA} calculations please see reference [4].

$$\psi_{CS} = R_{TIM}\,(PDF) \tag{8}$$

The PDF is Intel's power density factor and is available upon request through a field application engineer (FAE). Most of the TIM manufacturers will also provide engineers with the thermal resistance value R_{TIM}, which is used to compute ψ_{CS}. (Refer to Equation 8.)

Thermal resistance (θ) on the other hand is the characterization of a package's temperature rise per watt. This value dictates what heatsink to use or design. To calculate thermal impedance θ_{JA} of the CPU, first define the local ambient temperature, then obtain the maximum junction temperature and TDP from the Intel Thermal Design Guide (TDG). (See Equation 9.)

$$\theta_{JA} = \frac{T_J - T_A}{TDP} \tag{9}$$

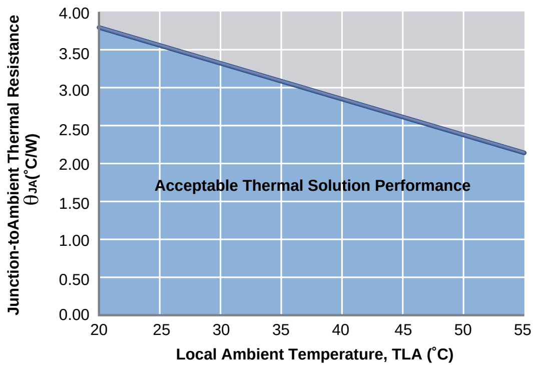


Figure 2: Thermal impedance of a CPU with respect to a range of local ambient temperature.

Figure 2 is an example of a range of local ambient temperatures versus thermal impedance plot. As shown in the graph, the area below the blue line highlights an acceptable heatsink performance for cooling. A heatsink of performance ψ_{JA} or better must be used must for effective cooling.

In summary, the thermal performance ψ_{JA} value of a heatsink must be equal or lower than thermal impedance θ_{JA} value at specified local ambient temperature range.

Example of an Optimized Plate Fin Extruded Thermal Solution Spacing Calculation

The following example illustrates the use of to determine an optimal plate fin heatsink. First, several engineering parameters must be defined; the heatsink material is a solid extruded aluminum grade Al6063. Next, the heatsink base thickness is fixed at 2 mm, an optimal thickness for small form factor heatsink of a solid aluminum grade Al6063. The details of heat constriction are beyond the scope of this article and are therefore not discussed here. Next, obtain the Prandtl number using air property at atmospheric pressure with a surrounding temperature of 300°K. From reference [1] Appendix A, Table A-4 we know Pr = 0.707. Finally the temperature difference ($T_S – T_F$) is set to 50°C. This is the temperature difference between the fin walls and the air envelope around the fins. With the above engineering parameters the optimal fin spacing is calculated as shown in Table 1.

Fin Length, L (mm)	Optimal fin spacing, S (mm)
35.0	4.48
37.5	4.56
40.0	4.63
42.5	4.70
45.0	4.77
47.5	4.84
50.0	4.90

Note: Make sure to use the correct measurement units as specified in above sections.

Table 1: Optimum plate fin spacing for natural convection heat transfer

A heatsink with characteristic length of 50 mm shown in Table 1 requires an optimal fin spacing of 4.90 mm. Using optimal fin spacing (S) and Equation 10, heatsink fin count and fin thickness is determined. Manufacturing process technology and capabilities will influence heatsink fin height and fin thickness design. It is the design engineer's responsibility to understand and take this into design consideration. For more information on the manufacturing process please refer to reference [8].

"Manufacturing process technology and capabilities will influence heatsink fin height and fin thickness design. It is the design engineer's responsibility to understand and take this into design consideration."

$$t = \frac{L - S(n-1)}{n} \tag{10}$$

where:

t = fin thickness (mm)

L = thermal solution length/size (mm)

S = optimum fin spacing (mm)

n = number of fins

Fin Length, L (mm)	Optimum fin spacing, S (mm)	No. of fins, n	Fin thickness, t (mm)
35.0	4.48	7	1.16
37.5	4.56	8	0.70
40.0	4.63	8	0.95
42.5	4.70	8	1.20
45.0	4.77	9	0.76
47.5	4.84	9	0.97
50.0	4.90	10	0.59

Table 2: Calculated number of fins and fin thickness per optimum fin spacing

As shown in Table 2, an optimized natural convection heatsink with characteristic length of 50 mm and an optimal fin spacing of 4.90 mm requires 10 fins with 0.59 mm thick for each fin.

"CFD uses numerical methods and algorithms to solve and analyze problems that involve fluid flows."

In summary, this section is a step by step analytical hand calculation. First, determine key parameters for designing a heatsink. Then determine the working/boundary conditions. Next, determine the optimal parallel plate fin spacing. Finally, calculate heatsink fin thickness and the number of fins.

Thermal Solution Design (Numerical)

CFD uses numerical methods and algorithms to solve and analyze problems that involve fluid flows; software like Flotherm*, Icepak* Cfdesign* are industrial accepted CFD software packages that are capable of solving fluid flow and heat transfer. In this document, all CFD and results reported are based on Flotherm v7.1.

Component Level CFD

CFD simulation often started off with a component level simulation follow by system level. The advantages of a component level simulation over system level are that there are fewer components in a simulation model, microscopic detail level of analysis is feasible, and faster simulation in convergence and often errors (if any) are easily traceable.

The example shown here uses a predefined boundary condition (JEDEC 51-2 standard) to characterize and compare thermal performance of the three heatsinks: one optimized and the other two non-optimized natural convection heatsinks. The internal volume is modeled with a dimension of 304.8 x 304.8 x 304.8 mm; the enclosure material is polycarbonate and the thickness is 6 mm. A wall is used to position a thermal test vehicle (TTV) at the center of the enclosure. All simulation components are attached with radiation attributes. The radiation exchange factor will be calculated automatically by the software. Figure 3 shows the location of the local ambient temperature (T_{LA}) measurement point with respect to the model setup. For more information on JEDEC51-2 setup and material used, see reference [9].

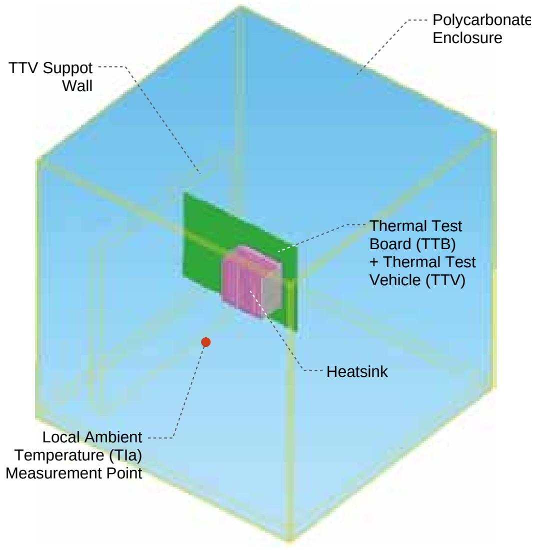

Polycarbonate Enclosure
TTV Suppot Wall
Thermal Test Board (TTB) + Thermal Test Vehicle (TTV)
Heatsink
Local Ambient Temperature (Tla) Measurement Point

Figure 3: Natural convection CFD simulation based on JEDEC51-2

The TTV model used in simulation is an Intel® Pentium® M processor on 90 nm process. The Flotherm model is available upon request through your Intel field application engineer (FAE). From the Intel Pentium M processor on 90 nm process Thermal Design Guide, refer to reference [4], Tj maximum = 100°C, and TDP is 10 W. Table 3 shows the CPU thermal impedance requirement based on a range of local ambient temperature. The thermal test board (TTB) is modeled as a generic block with conductivity of 10 W/mK.

T_{LA} (°C)	30	35	40	45	50	55	60
θ_{JA} (°C/W)	7.0	6.5	6.0	5.5	5.0	4.5	4.0

Table 3: Intel® Pentium® M Processor on 90 nm process thermal impedance θ_{JA} for range of T_{LA}

The three heatsink dimensions, geometries, and thermal performances are shown in Table 4.

	Optimized HS	Non-optimized HS1	Non-optimized HS2
HS Base (mm)	50 x 50 x 2		
Fin Height	28 mm		
Fin Thickness	0.59 mm	1.00 mm	1.00 mm
Fin Count	10	6	14
T_J (°C)	77.55	79.87	82.85
T_{LA} (°C)	33.04		
Ψ_{CS} (°C/W)	0.17		
Ψ_{JA} (°C/W)	4.62	4.85	5.11

Note: Referring to Shin-Etsu TIM datasheet, X23-7783D contact resistance is 7.3 mm^{2}K/W.

Table 4: Plate fin heatsink dimension and thermal performance

"This component level CFD example utilizes design on experiment (DOE) to determine the accuracy of the first-order approximation hand calculation."

In summary, this component level CFD example utilizes design on experiment (DOE) to determine the accuracy of the first-order approximation hand calculation in the earlier section. From the CFD model setup, grease type thermal interface material (TIM) was not factored in (the physical nature of grease TIM exhibits undeterminable measurement capability analysis (MCA) and modeling in CFD will cause inaccurate end results). Grease TIM performance depends on bond line thickness (BLT), contact pressure, surface roughness, and heat cycle. The detail discussion of grease TIM is beyond the scope of this article. Use Equation 8 to calculate TIM performance as is in this article. Table 4 shows the corrected thermal performance Ψ_{JA}. Take note that the final thermal performance Ψ_{JA} could differ pending on what TIM is used; the higher performance TIM used the lower final thermal performance.

System Level CFD

This section depicts a specific system level simulation using point of sale as an example. The goal is to enable the system designer to understand how CFD predicts an optimized natural convection heatsink performance under point-of-sale system boundary conditions. The CFD example illustrated here is a 12.1" touchscreen LCD vertical standing POS system; refer to Figure 4. The enclosure is an aluminum box chassis with external dimensions of 300 x 250 x 65 mm. The enclosure is simulated with top and bottom vent openings; total free area ratio (FAR) is set to 20 percent for both top and bottom vents. The hole pattern for the vents are 5-mm hexagons uniformly distributed across the entire top and bottom surface. Vent holes governed by FAR are important as it determines whether the system/platform will experience heat soak. When heat soak occurs within a system, temperature rise (local ambient temperature minus room temperature) will increase.

There is a polyimide insulating film separating the LCD from a single board computer (SBC) and other peripherals. Above the SBC is a DC to DC power PCB, 2.5" HDD and a CD-ROM drive modeled at the side (shown in Figure 4 as silver color blocks). A 12.1" LCD is located right behind the insulating film. The SBC orientation as shown in Figure 4 is to accommodate side-accessible I/O ports and position the processor at the bottom, closest to the vents. The processor is placed at the lowest

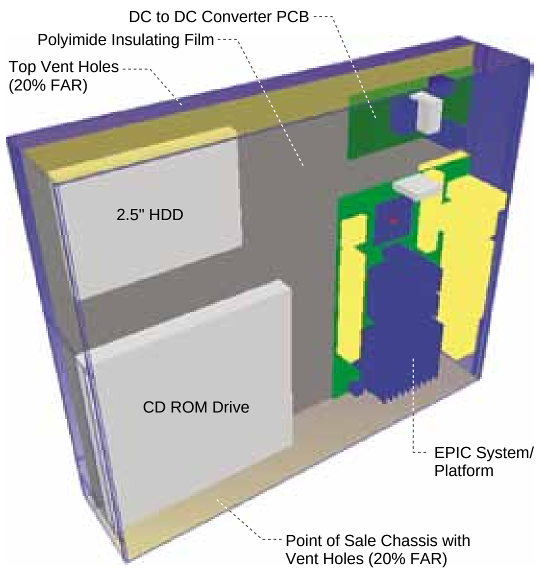


Figure 4: System level CFD - 12.1" POS (vertical)

region of the enclosure to deliver fresh cooler air from the bottom vent openings. The SBC used is an Embedded Platform for Industrial Computing (EPIC) small form factor board with the Intel Pentium M processor built on 90-nm process paired with the Intel® 855GME Graphics Memory Controller Hub (GMCH) and Intel® 82801DB I/O Controller Hub 4 (ICH4). The thermal solution is a 50 x 50 x 30 mm heatsink mentioned in the previous section. The orientation of the heatsink is aligned such that its plate fins are parallel to the direction of gravity. All simulation components are attached with radiation attributes. The radiation exchange factor will be calculated automatically by the software. Table 5 is a list of components used in the CFD simulation; most of the materials are found in the Flotherm built-in material library. The right column is each component's TDP and is used for power budgeting.

Component	Material	Power (W)
12.1" LCD	Alumina	–
Insulating Film	Polyimide	–
Enclosure	Al 6063	–
Power Board	FR4	6 (assume)
Capacitors	Ethylene Glycol	–
Connectors	Polycarbonate	–
2.5" HDD	Alumina	0.6
CD ROM	Alumina	–
I/O ports	Polycarbonate	–
EPIC SFF	FR4	4 (assume)
SODIMM	Heat Block	3.6
CPU Heatsink	Al 6063	
MCH Heatsink	Al 6063	
CPU	Complex model	10
MCH	Complex model	4.3
ICH	Complex model	2.5

Table 5: List of components material and power used in the CFD simulation

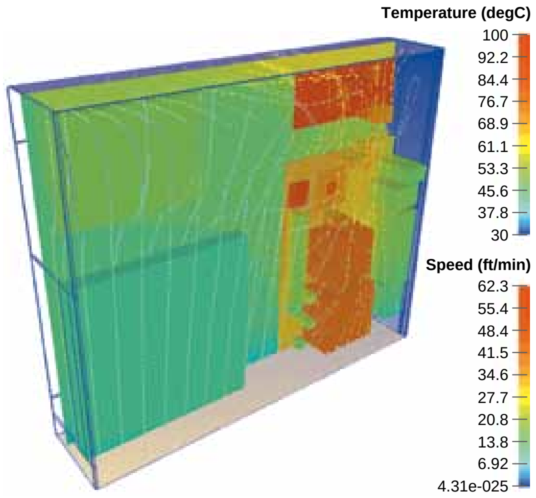


Figure 5: System level CFD – temperature and velocity plot.

A single scenario example is used to illustrate system level CFD. Some components are modeled as simple resistance blocks and in real application it may dissipate power. It is up to the user to specify these values based on their power budget estimate. The focus is on CPU, MCH, and ICH; detail modeling with finer meshing is put within this area in the simulation to improve the accuracy of the results. The total system power dissipated is assumed to be approximately 30 W; this is by adding all the TDP values shown in Table 5. TDP summation in simulation is not a real world application and the example here is to simulate a worst case scenario only.

Figure 5 shows the components' temperature plot and particle plot. DC to DC converter PCB temperature shown is hottest mainly due to preheat from system/platform below and heat soak. It is important not to place heat sensitive or low operating temperature components directly on top of a system/platform. When one looks more closely at the particle plot representing airflow within the enclosure, one can see that the top vent holes with 20 percent FAR are insufficient to remove the hot air generated. Top vent air is not exhausting linearly and air swirling is about to developed on the top right corner of the enclosure.

"With the proper concept and design process, a fanless thermal solution is feasible on Intel architecture."

Local ambient temperature shown in Table 6 is an average temperature surrounding the CPU heatsink. Unlike JEDEC 51-2 standard, there is no designated T_{LA} measurement point so it is the system engineer's responsibility to make sure several measurement points are used to represent the actual local ambient temperature within the system boundary.

T_{LA} (°C)	T_S (°C)	T_J (°C)	Ψ_{TIM} (°C/W)	*Ψ_{JA} (°C/W)
31.0	72.98	76.58	0.17	4.72

Table 6: Thermal performance of the CPU in system level CFD

Component level simulation shows that the same heatsink used in system level simulation has a slightly better thermal performance ψ_{JA}, 4.62°C/W versus 4.72°C/W. The primary reason is due to the fact that component level CFD has a single heat source and ample air volume for natural convection. In system level CFD, components are closely packed and experience mutual heating. The other reason is CFD meshing; component level has the advantage of modeling simplicity hence an optimal mesh ratio is easily achievable. In system level CFD engineers are often testing to balance between modeling accuracy and overall solving duration/convergence.

"In system level CFD engineers are often testing to balance between modeling accuracy and overall solving duration/convergence."

Conclusion

The Intel Embedded and Communications Group is now very much focused on low power and its efficiency. With the proper concept and design process, a fanless thermal solution is feasible on Intel architecture. This article serves as a reference solution to fanless cooling design for embedded applications.

Table of Acronyms and Symbols

CFD	Computational Fluid Dynamic
CPU	Central Processing Unit
DOE	Design on Experiment
ECG	Embedded and Communications Group
EPIC	Embedded Platform for Industrial Computing
FAR	Free Area Ratio
FCBGA	Flip Chip Ball Grid Array
Gr	Grashof number; Ratio of buoyancy forces to viscous forces
ICH	I/O Controller Hub
HIS	Integrated Heat Spreader
MCH	Memory Controller Hub
PCB	Printed Circuit Board
PDF	Power Density Factor
Pr	Prandtl number; Ratio of the momentum and thermal diffusivities
Ra	Rayleigh number; Ratio of the buoyancy force and momentum – thermal diffusivities
Re	Reynolds number; Ratio of the inertia and viscous forces
SBC	Single Board Computing
TDP	Thermal Design Power
TIM	Thermal Interface Material
TTV	Thermal Test Vehicle
ULV	Ultra Low Voltage
T_J	Junction Temperature
T_C	Case Temperature
T_S	Heatsink Temperature
T_{LA}	Local Ambient Temperature
Θ	Theta is used to characterize thermal impedance of a package
Ψ	Psi is used to characterize thermal performance of a heatsink
U	U is the standard unit of measure for designating the vertical usable space or height of racks and cabinets; 1U = 44.45 mm

References

[1] Fundamentals of Heat and Mass Transfer, 6th Edition, F. P. Incropera, D.P. Dewitt, T. L. Bergman, Lavine, A.S., John Wiley & Sons, Inc.

[2] Introduction to Thermal & Fluid Engineering, D. A. Kaminski, M. K. Jensen, John Wiley & Sons, Inc.

[3] ULV Intel® Celeron® M Processor @ 600MHz for fanless set top box application, 18741

[4] Intel® Pentium® M Processor on 90nm process for embedded application TDG, 302231

[5] Intel® Celeron® M Processor ULV 373, Intel® 852GM Graphics Memory Controller Hub (GMCH) & Intel® 82801DB I/O Controller Hub (ICH4) TDG for EST, 313426

[6] Thermal Modeling of Isothermal Cuboids & Rectangular Heat Sink Cooled by Natural Convection, J. R. Culham, M. M. Yovanovich, Seri Lee, IEEE transactions on components, packaging and manufacturing technology part A, Vol. 18, No. 3 September 1995

[7] Frigus Primore, A volumetric Approach to Natural Convection

[8] Design for manufacturability of forced convection air cooled fully ducted heat sinks, Electronics Cooling, Volume 13, No. 3, August 2007

[9] EIA/JEDEC51-2 Standard – Integrated Circuits Thermal Test Method Environment Conditions – Natural Convection (Still Air)

[10] TC attachment power point foils (internal)

Author Biographies

Chun Howe Sim: Chun Howe Sim (chun.howe.sim at intel.com) is a senior thermal mechanical application engineer in the Embedded and Communications Group at Intel Corporation. CH graduated from Oklahoma State University with a bachelor of science degree in mechanical engineering. CH joined Intel in 2005 as a thermal mechanical application engineer and has presented various embedded thermal design tracks at Taiwan IDF, the India embedded solution seminar, and the PRC annual ICA. As a thermal mechanical engineer, CH supports Low Power Intel® Architecture (LPIA) products, Digital Security Surveillance (DSS), and works on natural convection thermal solutions for embedded applications. Prior to joining Intel, CH worked for American Power Conversion (APC) as a DC Network solution system design engineer who supported Cisco,* Lucent,* AT&T,* and HuaWei.* CH was part of the mechanical design engineer team developing the highly scalable APC* InfraStruXure* architecture for the DC network.

Loh Jit Seng: Loh Jit Seng works as a thermal/mechanical application engineer in the Embedded and Communications Group (ECG) at Intel Corporation, supporting internal and external development of scalable and low-power embedded Intel architecture devices. His interests include fanless and liquid cooling technologies. Prior to joining Intel, he worked with iDEN Advanced Mechanics Group of Motorola,* working on structural and impact simulation of mobile phones. He received his bachelor's degree in engineering and is currently pursuing his master of science degree from the University Science Malaysia. His e-mail is jit.seng.loh at intel.com.

Copyright

SECURITY ACCELERATION, DRIVER ARCHITECTURE AND PERFORMANCE MEASUREMENTS FOR INTEL® EP80579 INTEGRATED PROCESSOR WITH INTEL® QUICKASSIST TECHNOLOGY

Contributors

Sundaram Ramakesavan
Intel Corporation

Sunish Parikh
Intel Corporation

Brian A. Keating
Intel Corporation

Index Words

Security
Acceleration
Cryptography
Intel® QuickAssist Technology
Intel® EP80579 Integrated Processor

Abstract

This article describes how the Intel® QuickAssist Technology components in the Intel® EP80579 Integrated Processor offload the compute-intensive cryptographic operations from the Intel® architecture core to a low-power cryptographic accelerator, thus making the processor ideal for security appliances requiring high throughput cryptography, high value-add applications, and a low power profile. This article also describes the Intel QuickAssist Technology Cryptographic API, which is developed jointly with Intel's partners in the Intel QuickAssist Technology community to allow application scalability across multiple hardware and software vendors. The article concludes with performance data as measured at the API level and at the level of a typical IPsec VPN application.

Introduction

The Intel® EP80579 Integrated Processor is a single chip that integrates in one die an Intel® Pentium® M processor, integrated memory controller hub (IMCH), integrated I/O controller hub (IICH) with two SATA and two USB 2.0 controllers, a PCI Express* (PCIe*) module, an I/O complex with three Gigabit Ethernet MACs (GbE), two Controller Area Network (CAN) interfaces, a IEEE 1588 timing module for both the GbE and CAN interfaces, a high precision watchdog timer (WDT), and a local expansion bus (LEB) interface. The integration of numerous functions usually available in discrete chips results in a cost-effective platform with significant footprint savings and time-to-market advantages. The high level of integration in a single die also means that less power is required to drive signals between different components and allows for the consolidation of clock and power delivery infrastructure, both of which result in a reduced power profile for the processor and platform. The Intel EP80579 Integrated Processor product line is available in 2 versions, the "embedded" version described above and also an "accelerated" version that includes Intel® QuickAssist Technology.

"The integration of numerous functions usually available in discrete chips results in a cost-effective platform with significant footprint savings and time-to-market advantages."

The Intel EP80579 Integrated Processor with Intel QuickAssist Technology is a pin-compatible version of the same processor family that comes with additional integrated components, including an integrated cryptographic accelerator that supports both symmetric and asymmetric cryptographic operations and throughput that is best-in-class compared to other leading external accelerators. The cryptographic accelerator allows the Intel EP80579 Integrated Processor to use a low power Intel® architecture core and still achieve impressive cryptographic performance. Furthermore, the integrated cryptographic accelerator requires less power to drive signals between the accelerator, processor core, and DRAM.

Note that the version of the chip that includes Intel QuickAssist Technology has other integrated components that facilitate both legacy and IP Telephony applications, but this article will focus on security applications.

The following sections describe how the Intel EP80579 Integrated Processor with Intel QuickAssist Technology can be used for developing security applications, hardware components of Intel QuickAssist Technology that are relevant to security applications, and the Intel QuickAssist Technology Cryptographic API, which is part of the enabling software and provides a software interface to accelerate the cryptographic operations.

"We describe how the Intel EP80579 Integrated Processor with Intel QuickAssist Technology can be used for the development of security applications, including IPsec VPNs, SSL VPNs, and SSL Gateways."

Security Applications

In this section, we describe how the Intel EP80579 Integrated Processor with Intel QuickAssist Technology can be used for the development of security applications, including IPsec VPNs, SSL VPNs, and SSL Gateways.

Security applications need to perform many cryptographic operations. For example, VPNs provide secure communications by providing confidentiality, integrity, and authentication using encryption and cryptographic hash functions. Cryptographic algorithms are, by design, computationally expensive. By offloading these operations from the Intel architecture core onto the integrated cryptographic accelerator, valuable CPU cycles are preserved, which can be used instead to add differentiating features and capabilities to the application. The Intel EP80579 Integrated Processor with Intel QuickAssist Technology supports offloading and acceleration of the cipher and hash algorithms required by the popular IPsec and SSL protocols.

"VPNs provide secure communications by providing confidentiality, integrity, and authentication using encryption and cryptographic hash functions."

IPSec and SSL protocols employ supporting protocols such as IKE and SSL handshakes that use public key cryptography to securely exchange keys for their secure communications channels. Many of these public key algorithms rely on large random numbers and prime numbers, and modular arithmetic and exponentiation operations involving these large numbers. The Intel EP80579 Integrated Processor with Intel QuickAssist Technology has accelerators that can perform modular exponentiation and inversion for large numbers up to 4096 bits long. Random number generation requires a greater degree of unpredictability than is generated by traditional pseudo-random number generators (PRNGs) found on many security coprocessors today. The Intel EP80579 Integrated Processor with Intel QuickAssist Technology strengthens the public key cryptography by offering a True Random Number Generation (TRNG) by including a Non-Deterministic Random Bit generator that periodically seeds a pseudo-random number generator.

"The Intel EP80579 Integrated Processor with Intel QuickAssist Technology strengthens the public key cryptography by offering a True Random Number Generation (TRNG) by its inclusion of a Non-Deterministic Random Bit generator that periodically seeds a pseudo-random number generator."

Security applications can use the cryptographic accelerators by directly calling the Intel QuickAssist Technology Cryptographic API. Alternatively, existing applications that use the open-source OpenBSD Cryptographic Framework (OCF) API can use the supplied OCF shim, which provides an implementation of the OCF API, to offload and accelerate their applications without any modifications to their code.

"Each Intel EP80579 Integrated Processor comes with redistributable and royalty-free enabling software that includes an implementation of the Intel QuickAssist Technology-based Cryptographic API."

Thus, there are simple mechanisms to offload and accelerate compute intensive operations of security applications and free up Intel architecture cycles for new and existing security applications.

Since the Intel EP80579 Integrated Processor is built on the Intel architecture, existing x86 software will run with minimal, if any, modification. In addition, for customers porting from non-Intel architecture platforms, it opens opportunities to reuse numerous existing applications and utilities to add capabilities to the product, choose from a variety of development tools, and make use of a highly optimizing Intel® compiler.

Hardware Components

Figure 1 is a block diagram of the Intel EP80579 Integrated Processor with Intel QuickAssist Technology showing the various integrated components. The key components relevant to security applications are the Intel® Architecture Processor where the security application is run, the Acceleration and I/O Complex (AIOC), and PCIe, which can be used to attach external NICs or other cards. Within the AIOC, the three Gigabit Ethernet (GbE) MACs are provided, while the combination of the Acceleration Services Unit (ASU) and Security Services Unit (SSU) act as the cryptographic accelerator. The ASU acts as a micro-sequencer for the SSU, invoking DMA between DRAM and the SSU's own internal memory and providing the Intel architecture core with an asynchronous request/response interface to the cryptographic acceleration where the requests and responses are sent via "rings" or circular buffers. Software running on the Intel architecture sends cryptographic requests by writing to these request rings and receives responses by reading from the response rings. The Response rings can also be configured to generate interrupts.

Intel® QuickAssist Technology-based Cryptographic API

Each Intel EP80579 Integrated Processor comes with redistributable and royalty-free enabling software that includes an implementation of the Intel QuickAssist Technology-based Cryptographic API. This API is grouped into various functional categories. One category supports session-based invocation of symmetric cryptography (cipher and authentication algorithms) that allows cryptographic parameters such as cipher, mode, or keys to be specified once, rather than on every invocation of the operation. Other categories include asymmetric public key algorithms like RSA, DSA, and Diffie-Hellman. Another category of APIs accelerates modular exponentiation and inversion of large numbers. Further API categories generate true random numbers, test the primality of numbers, and generate keys for SSL/TLS. Some miscellaneous API functions provide maintenance and statistics collection.

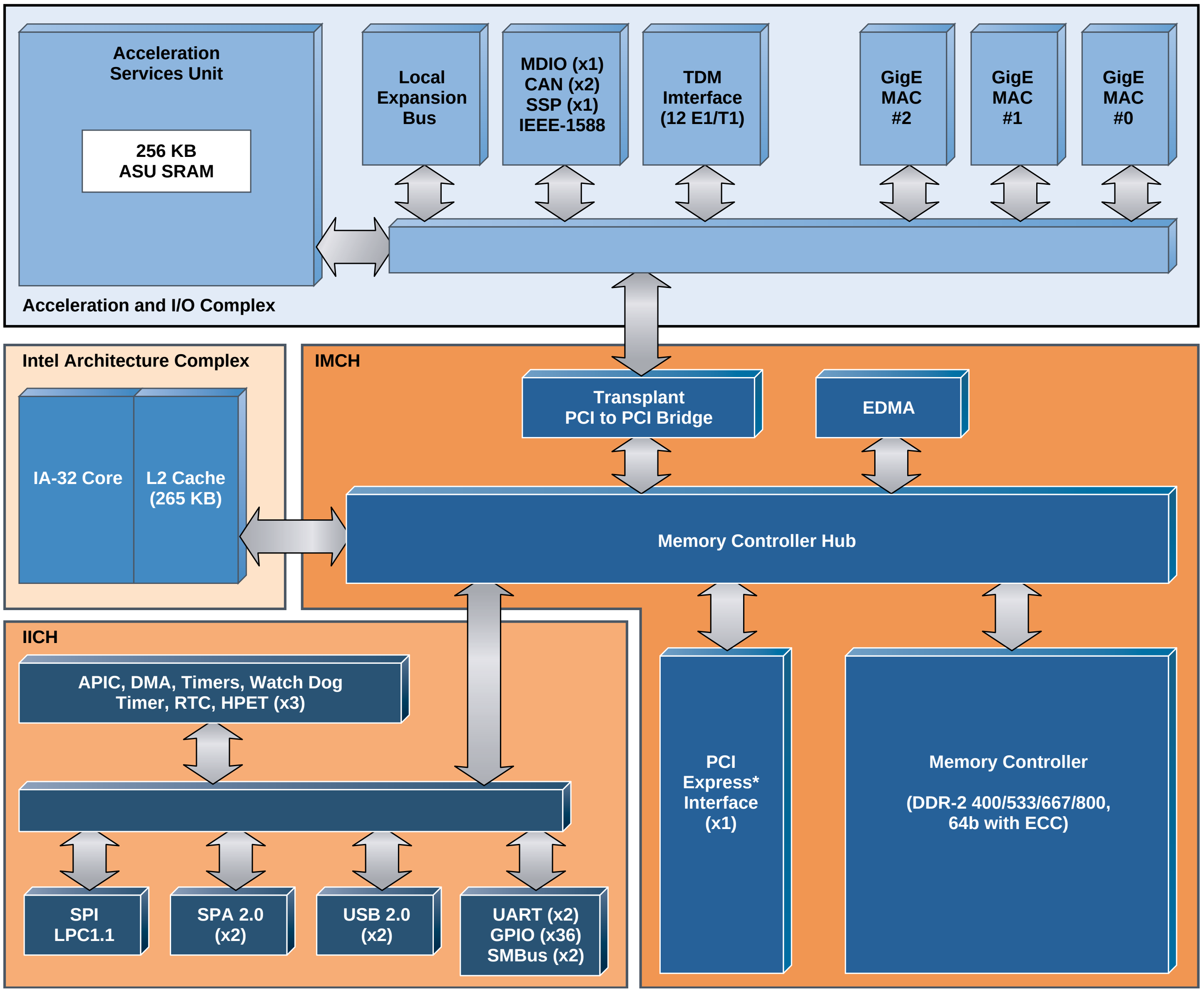

Acceleration Services Unit
256 KB ASU SRAM
Local Expansion Bus
MDIO (x1) CAN (x2) SSP (x1) IEEE-1588
TDM Imterface (12 E1/T1)
GigE MAC #2
GigE MAC #1
GigE MAC #0
Acceleration and I/O Complex
Intel Architecture Complex
IA-32 Core
L2 Cache (265 KB)
IMCH
Transplant PCI to PCI Bridge
EDMA
Memory Controller Hub
IICH
APIC, DMA, Timers, Watch Dog Timer, RTC, HPET (x3)
PCI Express* Interface (x1)
Memory Controller
(DDR-2 400/533/667/800, 64b with ECC)
SPI LPC1.1
SPA 2.0 (x2)
USB 2.0 (x2)
UART (x2) GPIO (x36) SMBus (x2)

Figure 1: Intel® EP80579 Integrated Processor product line with Intel® QuickAssist Technology. Source: Intel Corporation, 2009

The API supports various modes of operation that result in overall performance enhancement and allow for design flexibility. For example, the APIs used for symmetric and asymmetric cryptographic acceleration support both asynchronous and synchronous modes for receiving the cryptographic results, and in-place and out-of-place copying of cryptographic results for design flexibility. The symmetric cryptography API also supports algorithm chaining (to allow a cipher and hash to be performed, in either order, in a single request to the hardware), algorithm nesting, and partial-packet processing that improve the overall performance of security applications.

"Intel QuickAssist Technology-based Cryptographic APIs have been developed in collaboration with Intel's partners in the Intel QuickAssist Technology community to ensure its suitability and scalability."

The implementation of the API generates the request message and sends it to the ASU via a ring. The request message contains all the information required by the hardware accelerator, including the cryptographic parameters, pointers to the data buffers on which to operate, and so on. If synchronous operation was requested, the API implementation now blocks pending the response arriving. Otherwise, control is returned immediately to the caller.

Once the operation is complete, a response message containing information such as the encrypted result, computed key, and status are sent back to the Intel architecture core via another ring configured to generate an interrupt. Using information in the response, the API implementation either unblocks the caller or invokes the requested callback function in a bottom-half context. See Figure 2 for a stack of various software components.

Intel QuickAssist Technology-based Cryptographic APIs have been developed in collaboration with Intel's partners in the Intel QuickAssist Technology community to ensure its suitability and scalability. Partners with cryptographic accelerators are developing implementations of the API for their own accelerators. This allows application developers to easily port their security applications developed for Intel EP80579 Integrated processor to other Intel architecture platforms using cryptographic accelerators from other vendors if a different performance/power/price ratio is required.

Intel QuickAssist Technology-based Cryptographic APIs are also extensible. Future revisions of Intel QuickAssist Technology could offer higher performance, include additional cryptographic algorithms in support of wireless 3GPP standards, as well as non-cryptographic acceleration such as compression and regular expression pattern matching.

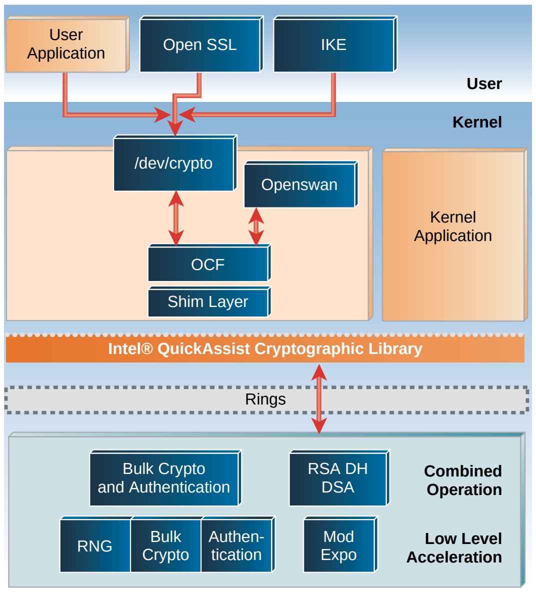


Figure 2: Intel® QuickAssist Technology-based Cryptographic Library, user application, middleware, accelerator stack.
Source: Intel Corporation, 2009

Performance

Security performance on the Intel EP80579 Integrated Processor with Intel QuickAssist Technology can be measured at the API level as well as at the level of a full application like IPSec. The API level measurements gives the potential best case performance, while the application level measurement shows how a typical non-optimized open-source application can benefit by using the transparent acceleration offered by the OCF software shim and the underlying cryptographic accelerator. The application-level performance measurement was made using the popular open-source IPSec application Openswan.*

API-Level Performance

Here we compare the execution of the encryption algorithm 3DES-CBC, which is known to be computationally expensive at the API-level, against the OCF's default software implementation in the cryptosoft module. The performance test involves generating random plaintext messages of varying size in memory and requesting encrypt operations to be performed. The returned cipher text is not verified in the performance measuring test; however, prior to taking measurements the tests are executed and the result sampled and verified. System performance is a measurement of the total time taken from the submission of the first operation to the return of the final callback when the last operation has completed.

API-Level Performance Test Setup

The API-level performance test setup is detailed in Tables 1 and 2, for hardware and software respectively.

Platform	Intel® EP80579 Integrated Processor Customer Reference Board
Processor	Intel EP80579 Integrated Processor with Intel® QuickAssist Technology
Core Frequency	1.2 GHz
L2 Cache	256 KB
Front Side Bus	133 MHz (Quad pumped)
PCIx/PCI Express* (PCIe*)	PCIe x4
Memory	1 GB DDR2 Registered 800 MHz DIMM, Single Rank
Ethernet	3 on-board NICs

Table 1: API-level performance test hardware setup

Operation System	Redhat* Enterprise Linux* 5 Client; kernel version 2.6.18
Enabling Software for Security Applications on Intel® QuickAssist Technology	Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Version L.1.0.104
BIOS	Intel® EP80579 Integrated Processor CRB BIOS version 057

Table 2: API-level performance test software setup

Figure 3 shows the raw throughput performance when directly accessing the Intel QuickAssist Technology Cryptographic APIs that Intel provides. As evident from the chart, at large packet sizes, using the hardware acceleration engines in the Intel EP80579 Integrated Processors for cryptographic operations gives about a 43x performance boost over doing cryptographic operations in software.

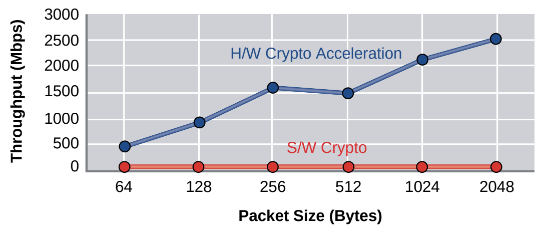


Figure 3: Look-aside Cryptographic API result. Source: Intel Corporation, 2009

Application Performance Using Openswan*

In this case, we measure the combined performance of the 3DES-CBC encryption and decryption with chained HMAC-SHA1 authentication for the open-source IPSec VPN application. Openswan natively uses Linux* Kernel Crypto API, but was patched to work with the OCF framework using a patch available from the ocf-linux open source project. In order to benchmark the Intel EP80579 Integrated Processor, measurements were taken first by configuring the OCF to use Intel QuickAssist Technology-based Cryptographic API via the OCF Shim, and then measured again by configuring the OCF to use software cryptographic module cryptosoft. The measurements were made on a 1.2-GHz Intel EP80579 Integrated Processor Customer Reference Board (CRB) using Spirent* SmartFlow* to generate and terminate traffic.

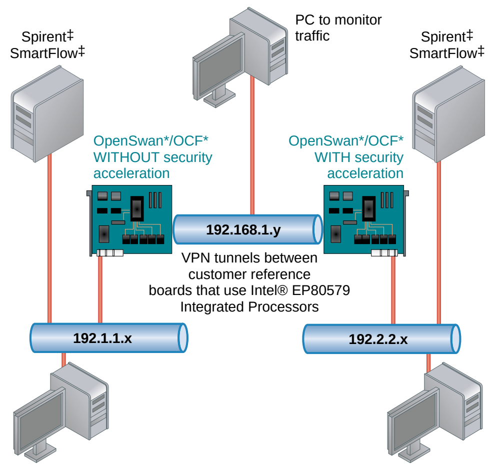

Spirent‡ SmartFlow‡
PC to monitor traffic
Spirent‡ SmartFlow‡
OpenSwan*/OCF* WITHOUT security acceleration
OpenSwan*/OCF* WITH security acceleration
192.168.1.y
VPN tunnels between customer reference boards that use Intel® EP80579 Integrated Processors
192.1.1.x
192.2.2.x

Figure 4: IPSec configuration used in performance measurement.

Application-Level Performance Test Setup

The test setup for Openswan, shown in Figure 4, consisted of two Intel EP80579 Integrated Processor CRBs connected via Openswan VPN tunnels. Spirent Smart-Flow application software was used to generate and terminate traffic across configured tunnels. The monitoring PC running Wireshark* (formerly called Ethereal*) was used to verify the tunnels established prior to taking test measurements and was disconnected for the actual performance measurements. The application-level performance test setup is detailed in Tables 3 and 4, for hardware and software respectively.

Platform	Intel® EP80579 Integrated Processor Customer Reference Board
Processor	Intel EP80579 Integrated Processor with Intel® QuickAssist Technology
Core Frequency	1.2 GHz
L2 Cache	256 KB
Front Side Bus	133 MHz (Quad Pumped)
PCIx/PCI Express* (PCIe*)	PCIe x4
Memory	1 GB DDR2 Registered 800 MHz DIMM, Single Rank
Ethernet	3 on-board NICs

Table 3: Application-level performance test hardware setup

Operation System	Redhat* Enterprise Linux* 5 Client; kernel version 2.6.18
Openswan*	Openswan VPN software version 2.4.9
Framework	Open Cryptographic Framework Patch 20070727
Enabling Software for Security Applications on Intel QuickAssist Technology	Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Release 1.0.1_RC2
BIOS	Intel® EP80579 Integrated Processor CRB BIOS version 061

Table 4: Application-level performance test software setup

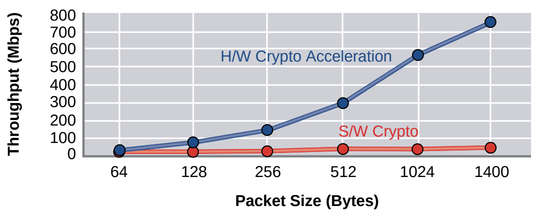


Figure 5: Application level performance results using Openswan*

"The Intel EP80579 Integrated Processor product line integrates together an Intel architecture core, chipset, and cryptographic accelerator into a single System on a Chip."

Using the cryptographic accelerator in the Intel EP80579 Integrated Processor increases the throughput of an IPSec VPN application by almost 17x (Figure 5) as compared to using software only to secure the traffic. The results above were obtained using a single tunnel between two Intel EP80579 Integrated Processors with Intel QuickAssist Technology–based gateway systems.

Conclusion

The Intel EP80579 Integrated Processor product line integrates together an Intel architecture core, chipset, and cryptographic accelerator into a single System on a Chip. With its high level of integration and embedded lifecycle support, it provides a combination of performance, footprint savings, cost-effectiveness, time to market advantages, and low power profile that compare very favorably to discrete, multi-chip solutions. When using the Intel EP80579 Integrated Processor with Intel QuickAssist Technology, the security processing can be offloaded onto the integrated accelerators, freeing up CPU cycles that can then be used to increase throughput, or to add differentiating features and capabilities to the Intel architecture application, or both. The Intel QuickAssist Technology-based Cryptographic API was defined in collaboration with Intel's partners. Applications developed against this API can be run on other Intel architecture platforms using cryptographic accelerators from Intel's partners to achieve different performance/power/price ratios.

References

See *www.intel.com/go/soc* for all hardware documentation, software documentation, application notes, and white papers.

Author Biographies

Sundaram Ramakesavan: Sundaram Ramakesavan has a master of science degree in computer science from Queen's University Canada. He has worked in various telecommunication, data communication, security, user interface, and localization projects at Nortel Networks and Intel Corporation. He is currently a technical marketing engineer specializing in cryptography and security applications. His e-mail is ramu.ramakesavan at intel.com

Sunish Parikh: Sunish Parikh has been at Intel Corporation since August 2000. He has a master of science degree in computer engineering from Florida Atlantic University. Sunish has previously worked in the area of software performance optimizations in the enterprise applications market. He is currently working on performance of software and hardware products in the Embedded and Communications Group at Intel Corporation. His email is sunish.u.parikh at intel.com

Brian A. Keating: Brian A. Keating is a software architect with the Performance Processor Division at Intel Corporation's Embedded and Communications Group. Brian has been with Intel for seven years, during which time he has worked in software development on a number of Intel's network processors, communications processors, and related products. Brian is currently the lead software architect for the Intel® EP80579 Integrated Processor product family, with a focus on security applications. Previously, Brian has architected and developed software for media gateways with a leading telecommunications and networking vendor, and developed security software with a leading computer vendor.

His email is brian.a.keating at intel.com

Copyright

METHODS AND APPLICATIONS OF SYSTEM VIRTUALIZATION USING INTEL® VIRTUALIZATION TECHNOLOGY (INTEL® VT)

Contributor

David Kleidermacher, CTO
Green Hills Software, Inc.

Index Words

Intel® vPro™ technology
Intel® Virtualization Technology
Intel® VT
security
virtualization
hypervisor
Common Criteria
Intel® Atom™ processor
microkernel
mobile Internet devices
real-time

Abstract

The motivations for system virtualization technology in the data center are well known, including resource optimization and improved service availability. But virtualization technology has broader applications throughout the enterprise and in the home, including security-enabled mobile devices, virtual appliances, secure servers, personal/corporate shared use laptops, trusted web-based transactions, and more. This vision is made possible due to Intel® Virtualization Technology (Intel® VT), which is hardware virtualization technology that scales from embedded and mobile devices up to server-class computing. This article provides an overview of the evolution of hypervisor architectures, including both software and hardware trends, and how they affect the security and practicality of system virtualization. We shall also discuss a range of compelling applications for secure virtualization across a variety of communities of interest.

Introduction

Computer system virtualization was first introduced in mainframes during the 1960s and 1970s. Although virtualization remained a largely untapped facility during the 1980s and 1990s, computer scientists have long understood many of the applications of virtualization, including the ability to run distinct and legacy operating systems on a single hardware platform.

At the start of the millennium, VMware proved the practicality of full system virtualization, hosting unmodified, general purpose, "guest" operating systems such as Windows on common Intel® architecture-based hardware platforms.

In 2005, Intel launched Intel® Virtualization Technology (Intel® VT), which both simplified and accelerated virtualization. Consequently, a number of virtualization software products have emerged, alternatively called virtual machine monitors or hypervisors, with varying characteristics and goals.

"A number of virtualization software products have emerged, alternatively called virtual machine monitors or hypervisors, with varying characteristics and goals."

While Intel VT may be best known for its application in data center server consolidation and provisioning, Intel VT has proliferated across desktop- and laptop-class chipsets, and has most recently found its way into Intel® Atom™ processors, built for low power and designed for embedded and mobile applications.

The availability of Intel VT across such a wide range of computing platforms provides developers and technologists with the ultimate open platform: the ability to run any flavor of operating system in any combination, creating an unprecedented flexibility for deployment and usage. This article introduces some of these emerging uses, with an emphasis on the latest platforms enabled with Intel VT: embedded and mobile. Because embedded and mobile platforms often have resource and security constraints that differ drastically from enterprise computing platforms, this article also focuses on the impact of hypervisor architecture upon these constraints.

"Initially, virtualization was used for time sharing, similar to the improved hardware utilization driving modern data center server consolidation."

Applications of System Virtualization

Mainframe virtualization was driven by some of the same applications found in today's enterprise systems. Initially, virtualization was used for time sharing, similar to the improved hardware utilization driving modern data center server consolidation. Another important usage involved testing and exploring new operating system architectures. Virtualization was also used to maintain backward compatibility of legacy versions of operating systems.

Environment Sandboxing

Implicit in the concept of consolidation is the premise that independent virtual machines are kept securely separated from each other. The ability to guarantee separation is highly dependent upon the robustness of the underlying hypervisor software. As we'll soon discuss, researchers have found flaws in commercial hypervisors that violate this separation assumption. Nevertheless, an important theoretical application of virtual machine compartmentalization is to "sandbox" software that is not trusted. For example, a web browser connected to the Internet can be sandboxed in a virtual machine so that Internet-borne malware or browser vulnerabilities are unable to infiltrate or otherwise adversely impact the user's primary operating system environment.

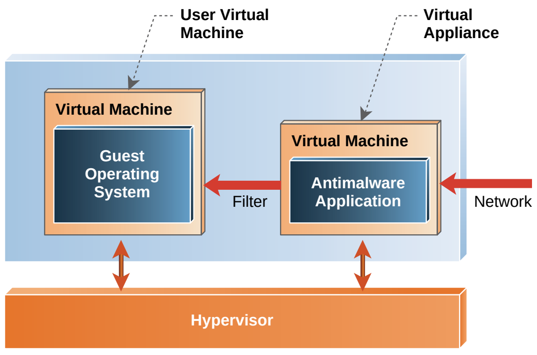


Figure 1: Virtual security appliance.
Source: Green Hills Software, 2008

Virtual Security Appliances

Another example, the virtual security appliance, does the opposite: sandbox trusted software away from the user's operating system environment. Consider anti-virus software that runs on a Mobile Internet Device (MID). A few years ago, the "Metal Gear" Symbian Trojan was able to propagate itself by disabling the mobile device's anti-malware software. [1] Virtualization can solve this problem by placing the anti-malware software into a separate virtual machine, as shown in Figure 1.

The virtual appliance can analyze data going into and out of the user's environment or hook into the user's operating system for demand-driven processing.

"Virtualization can solve this problem by placing the anti-malware software into a separate virtual machine."

Hypervisor Architectures

Hypervisor architectures vary along several dimensions. Some are open source, others are proprietary. Some comprise thin hypervisors augmented with specialized guest operating systems. Others employ a monolithic hypervisor that is fully self-contained. In this section, we shall compare and contrast currently available architectures.

Monolithic Hypervisor

Hypervisor architectures seen in commercial applications most often employ a monolithic architecture, as shown in Figure 2. Similar to monolithic operating systems, the monolithic hypervisor requires a large body of operating software,

"Hypervisor architectures vary along several dimensions."

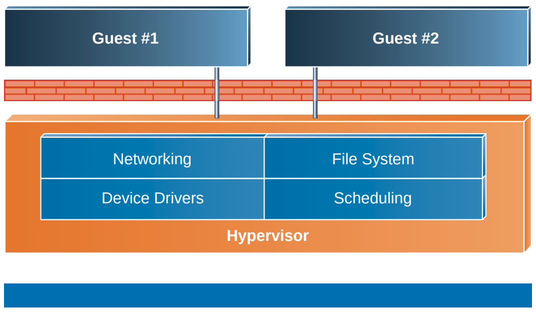


Figure 2: Monolithic hypervisor architecture. Source: Green Hills Software, 2008

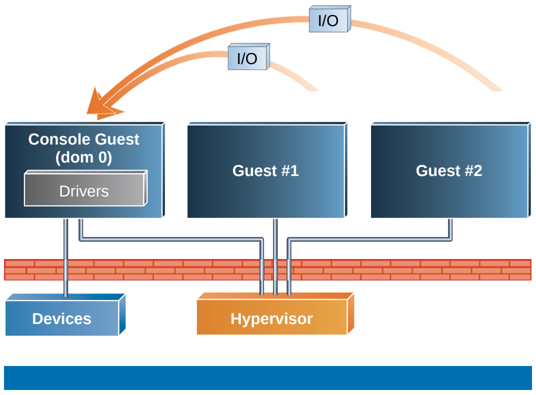


Figure 3: Console guest hypervisor architecture. Source: Green Hills Software, 2008

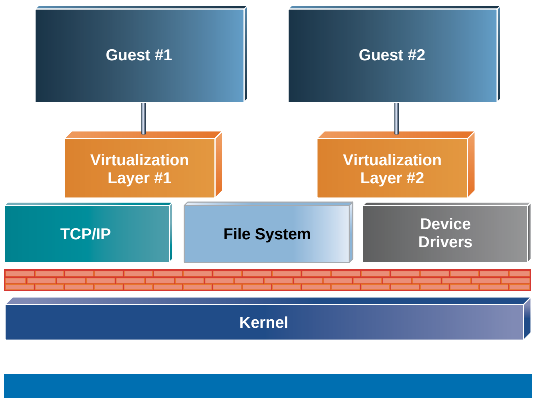


Figure 4: Microkernel-based hypervisor architecture. Source: Green Hills Software, 2008

including device drivers and middleware, to support the execution of one or more guest environments. In addition, the monolithic architecture often uses a single instance of the virtualization component to support multiple guest environments. Thus, a single flaw in the hypervisor may result in a compromise of the fundamental guest environment separation intended by virtualization in the first place.

Console Guest Hypervisor

An alternative approach uses a trimmed down hypervisor that runs in the microprocessor's privileged mode but employs a special guest operating system partition to handle the I/O control and services for the other guest operating systems Thus, a complex body of software must still be relied upon for system security. As shown in Figure 3, a typical console guest, such as Linux operating system, may add far more code to the virtualization layer than found in a monolithic hypervisor.

Microkernel-based Hypervisor

The newest hypervisor architecture was designed specifically to provide robust separation between guest environments. Figure 4 shows the microkernel-based hypervisor architecture.

This architecture places the computer virtualization complexity into user-mode processes outside the trusted operating system microkernel, as, for example, in Green Hills Software's Integrity. A separate instance of the virtualization layer is used for each guest environment. Thus, the virtualization layer need only meet the equivalent (and, typically, relatively low) robustness level of the guest itself.

Paravirtualization

System virtualization can be implemented with full virtualization or paravirtualization, a term first coined in the 2001 Denali project. [2] With full virtualization, unmodified guest operating systems are supported. With paravirtualization, the guest operating system is modified in order to improve the ability of the underlying hypervisor to achieve its intended function.

Paravirtualization is often able to provide improved performance and lower power consumption. For example, device drivers in the guest operating system can be modified to make direct use of the I/O hardware instead of requiring I/O accesses to be trapped and emulated by the hypervisor.

Contrary to enterprise computing requirements, most of the virtualization deployed within low power embedded systems have used paravirtualization. This trend is likely to change, however, due to the inclusion of Intel VT in low power chipsets. The advantage to full virtualization is the ability to use unmodified versions of operating systems that have a proven fielded pedigree and do not require the maintenance associated with custom modifications. This maintenance savings is especially important in embedded devices where I/O peripherals tend to vary dramatically across designs.

Leveraging Intel® VT

Intel VT has been a key factor in the growing adoption of full virtualization throughout the enterprise computing world. Intel VT for IA-32, Intel® 64 and Intel® Architecture (Intel VT-x) provides a number of hypervisor assistance capabilities. For example, true hardware hypervisor mode enables unmodified ring-0 guest operating systems to execute with reduced privilege. Intel VT-x will also prevent a guest operating system from referencing physical memory beyond what has been allocated to the guest's virtual machine. In addition, Intel VT-x enables selective exception injection, so that hypervisor-defined classes of exceptions can be handled directly by the guest operating system without incurring the overhead of hypervisor software interposing.

"True hardware hypervisor mode enables unmodified ring-0 guest operating systems to execute with reduced privilege."

Early Results with Intel® VT

In 2006, Green Hills Software demonstrated virtualization using Intel VT-x. Prior to this, in 2005, Green Hills demonstrated a full virtualization solution on platforms without Intel VT capabilities. We did so by using selective dynamic translation techniques conceptually similar to that employed by original versions of VMware.

Green Hills Software's previous desktop solution was able to support no more than two simultaneous full-motion audio/video clips (each in a separate virtual machine) without dropping frames. With Intel VT-x on similar class desktops, the number of simultaneous clips was limited only by the total RAM available to host multiple virtual machines. General PC benchmarks showed an approximate factor of two performance improvement for Intel VT-x over earlier platforms. In addition, the Green Hills virtualization layer was radically simplified due to the Intel VT-x capabilities.

Recent Improvements

In 2008, Green Hills Software demonstrated its virtualization technology enabled by Intel VT-x on Intel Atom processors, thereby taking advantage of the scalability of Intel VT-x across low power embedded systems, laptops and desktops, and server-class systems.

"General PC benchmarks showed an approximate factor of two performance improvement for Intel VT-x over earlier platforms."

In 2007, Green Hills demonstrated the use of Intel VT for Directed I/O (Intel VT-d) in its desktop-based offerings. In 2008, Green Hills demonstrated the use of Intel VT-d in Intel® Centrino® 2 processor technology-based laptops. Intel VT-d's DMA remapping capability further enhances virtualization performance and reduces hypervisor software complexity by enabling select I/O peripherals to be controlled directly by the guest operating system, with little or no intervention from virtualization software.

Intel VT has enabled Green Hills Software and other technology suppliers to leverage the power of full system virtualization across a wide range of hardware platforms, vertical industries, and emerging usage scenarios (some of which we shall discuss in the section "Emerging Applications for Virtualization").

"Intel VT has enabled Green Hills Software and other technology suppliers to leverage the power of full system virtualization across a wide range of hardware platforms, vertical industries, and emerging usage scenarios."

Hypervisor Security

Some tout virtualization as a technique in a "layered defense" for system security. The theory postulates that since only the guest operating system is exposed to external threats, an attacker who penetrates the guest will be unable to subvert the rest of the system. In essence, the virtualization software is providing an isolation function similar to the process model provided by most modern operating systems.

"A number of studies of virtualization security and successful subversions of hypervisors have been published."

Published Hypervisor Subversions

However, common enterprise virtualization products have not met security requirements for high robustness and were never designed or intended to meet these levels. Thus, it should come as no surprise that the theory of security via virtualization has no existence proof. Rather, a number of studies of virtualization security and successful subversions of hypervisors have been published.

In 2006, the SubVirt project demonstrated hypervisor rootkits that subverted both VMware and VirtualPC. [3]

The BluePill project took hypervisor rootkits a step further by demonstrating a malware payload that was itself a hypervisor that could be installed on-the-fly, beneath a natively running Windows operating system. [4]

Tavis Ormandy performed an empirical study of hypervisor vulnerabilities. The researchers generated random I/O activity into the hypervisor, attempting to trigger crashes or other anomalous behavior. The project discovered vulnerabilities in QEMU, VMware* Workstation and Server, Bochs, and a pair of unnamed proprietary hypervisor products. [5]

Clearly, the risk of an "escape" from the virtual machine layer, exposing all guests, is very real. This is particularly true of hypervisors characterized by monolithic code bases. As one analyst has said, "Virtualization is essentially a new operating system …, and it enables an intimate interaction between underlying hardware and the environment. The potential for messing things up is significant." [6]

"Clearly, the risk of an 'escape' from the virtual machine layer, exposing all guests, is very real."

At the 2008 Black Hat conference, security researcher Joanna Rutkowska and her team presented their findings of a brief research project to locate vulnerabilities in Xen. [7] One hypothesis was that Xen would be less likely to have serious vulnerabilities, as compared to VMware and Microsoft* Hyper-V, due to the fact that Xen is an open source technology and therefore benefits from the "many-eyes" exposure of the code base.

Rutkowka's team discovered three different and fully exploitable vulnerabilities that the researchers used to commandeer the computer by way of the hypervisor. Ironically, one of these attacks took advantage of a buffer overflow defect in Xen's Flask layer. Flask is a security framework that is the same one used in SELinux. It was added to Xen to improve security.

Rutkowka's results further underscore an important principle: software that has not been designed for and evaluated to high levels of assurance must be assumed to be subvertible by determined and well-resourced entities.

"Rutkowka's results further underscore an important principle: software that has not been designed for and evaluated to high levels of assurance must be assumed to be subvertible by determined and well-resourced entities."

High Assurance Approach

However, the hypervisor need not hamper security efforts. For example, Integrity is an operating system that has achieved a high assurance Common Criteria security certification. [8] Designed for EAL 7, the highest security level, Integrity meets what the National Security Agency deems is required for "high robustness:" protection of high value resources against highly determined and sophisticated attackers.

Our operating system is being used in NSA-approved cryptographic communications devices, avionics systems that control passenger and military jets, life-critical medical systems, secure financial transaction systems, and a wide variety of other safety and security-critical systems.

We have found that a security kernel can provide domain separation with virtualization duties relegated to user-mode applications. This approach achieves a high level of assurance against hypervisor escapes.

"We have found that a security kernel can provide domain separation with virtualization duties relegated to user-mode applications."

Integrity provides a full-featured applications programming interface (API) and software development kit (SDK), enabling the creation and deployment of secure applications that cannot be trusted to run on a guest. Thus, critical security applications and data such as firewalls, databases, and cryptographic subsystems can be deployed both alongside and securely separated from general purpose operating environments such as Windows or Linux.

The combination of virtualized and native applications results in a powerful hybrid operating environment, as shown in Figure 5, for the deployment of highly secure yet richly functional systems. In the following section, we shall discuss how this hybrid architecture is especially critical for the flexibility required in embedded systems.

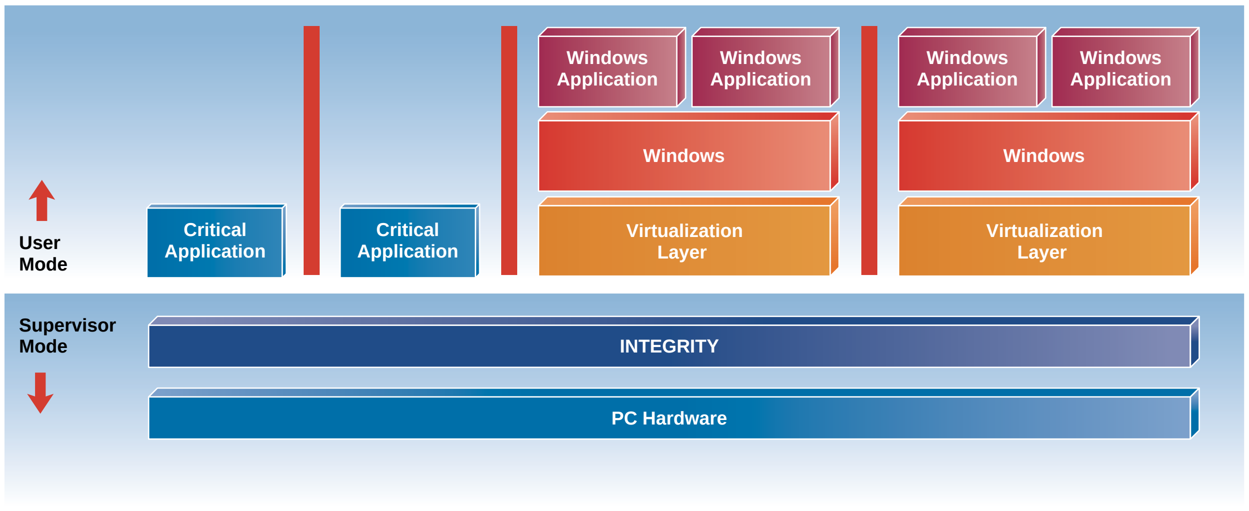


Figure 5: Virtualized environments alongside native applications. Source: Green Hills Software, 2008

Emerging Applications for Virtualization

The use of virtualization outside of traditional enterprise PC and server markets is nascent, and yet presents a significant opportunity. In this section, we shall discuss a sample of emerging applications with significant promise.

"The use of virtualization outside of traditional enterprise PC and server markets is nascent."

"Electronic Flight Bag (EFB) is a general purpose computing platform that flight crews use to perform flight management tasks."

Telecom Blade Consolidation

Virtualization enables multiple embedded operating systems, such as Linux and VxWorks, to execute on a single telecom computer, such as an AdvancedTCA blade server based on Intel® Architecture Processors. In addition, the microkernel-based virtualization architecture enables real-time applications to execute natively. Thus, control plane and data plane applications, typically requiring multiple blades, can be consolidated. Telecom consolidation provides the same sorts of size, weight, power, and cost efficiencies that enterprise servers have enjoyed with VMware.

Electronic Flight Bag

Electronic Flight Bag (EFB) is a general purpose computing platform that flight crews use to perform flight management tasks, including calculating take-off parameters and viewing navigational charts more easily and efficiently. EFBs replace the stereotypical paper-based flight bags carried by pilots. There are three classes of EFBs, with class three being a device that interacts with the onboard avionics and requires airworthiness certification.

Using the hybrid virtualization architecture, a class three EFB can provide a Windows environment (including common applications such as Microsoft Excel) for pilots while hosting safety-critical applications that validate parameters before they are input into the avionics system. Virtualization enables class three EFBs to be deployed in the portable form factor that is critical for a cramped cockpit.

"Virtualization enables class three EFBs to be deployed in the portable form factor that is critical for a cramped cockpit."

Intelligent Munitions System

Intelligent Munitions System (IMS) is a next-generation U.S. military net-centric weapons system. One component of IMS includes the ability to dynamically alter the state of munitions (such as mines) to meet the requirements of an evolving battlescape. Using the hybrid virtualization architecture, the safety-critical function of programming the munitions and providing a trusted display of weapons state for the soldier is handled by secure applications running on the safety-certified microkernel. A standard Linux or Windows graphical interface is enabled with virtualization.

In-Vehicle Infotainment

Demand for more advanced infotainment systems is growing rapidly. In addition to theater-quality audio and video and GPS navigation, wireless networking and other office technologies are making their way into the car. Despite this increasing complexity, passenger expectations for "instant on" and high availability remain. At the same time, automobile systems designers must always struggle to keep cost, weight, power, and component size to a minimum.

"The currently deployed solution, found on select high-end automobiles, is to divide the infotainment system onto two independent hardware platforms."

Although we expect desktop operating systems to crash occasionally, automobile passengers expect the radio and other traditional "head-unit" components never to fail. In fact, a failure in one of these components is liable to cause an expensive (for the automobile manufacturer) visit to the repair shop. Even worse, a severe design flaw in one of these systems may result in a recall that wipes out the profit on an entire model year of cars. Exacerbating the reliability problem is a new generation of security threats: bringing the Internet into the car exposes it to all the viruses and worms that target networked Windows-based computers.

The currently deployed solution, found on select high-end automobiles, is to divide the infotainment system onto two independent hardware platforms, placing the high-reliability, real-time components onto a computer running a real-time operating system, and the Windows component on a separate PC. This solution is highly undesirable, however, because of the need to tightly constrain component cost, size, power, and weight within the automobile.

The hybrid virtualization architecture provides an ideal solution. Head unit applications running under control of the real-time kernel are guaranteed to perform flawlessly. Because the real-time kernel is optimized for the extremely fast boot times required by automotive systems, instant-on requirements are met.

Multiple instances of Windows, powered by multiple instances of the virtual machine, can run simultaneously on the same computer. In the back seat, each passenger has a private video monitor. One passenger could even reboot Windows without affecting the second passenger's email session.

"Multiple instances of Windows, powered by multiple instances of the virtual machine, can run simultaneously on the same computer."

Next Generation Mobile Internet Devices

Using the hybrid virtualization architecture, mobile device manufacturers and service providers can leverage traditional operating systems and software, such as the Linux-based Moblin platform [9], while guaranteeing the integrity, availability, and confidentiality of critical applications and information (Figure 6).

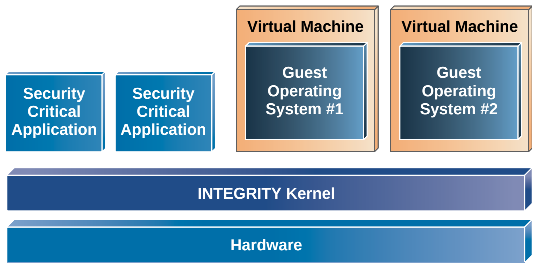

Virtual Machine
Guest Operating System #1
Virtual Machine
Guest Operating System #2
Security Critical Application
Security Critical Application
INTEGRITY Kernel
Hardware

Figure 6: Virtualization environment for Mobile Internet Devices (MID). Source: Green Hills Software, 2008

We bring our mobile devices wherever we go. Ultimately, consumers would like to use mobile devices as the key to the automobile, a smart card for safe Internet banking, a virtual credit card for retail payments, a ticket for public transportation, and a driver's license and/or passport. There is a compelling world of personal digital convenience just over the horizon.

The lack of a high security operating environment, however, precludes these applications from reaching the level of trust that consumer's demand. High assurance secure platform technology, taking maximum advantage of Intel silicon features such as Intel VT, enables this level of trust. Furthermore, security applications can be incorporated alongside the familiar mobile multimedia operating system on one chip (SoC), saving precious power and production cost.

"There is a compelling world of personal digital convenience just over the horizon."

Reducing Mobile Device Certification Cost

A certified high assurance operating system can dramatically reduce the cost and certification time of mobile devices, for two main reasons. First, because it is already certified to protect the most sensitive information exposed to sophisticated attackers, the operating system can be used to manage the security-critical subsystems. The certified operating system comes with all of its design and testing artifacts available to the certification authority, thus precluding the cost and time of certifying an operating system.

Second, the operating system and virtualization software take advantage of Intel VT and the Intel architecture Memory Management Unit (MMU) to partition security-critical components from the user's multimedia environment. For example, a bank may require certification of the cryptographic subsystems used to authenticate and encrypt banking transaction messages, but the bank will not care about certifying the system's multimedia functions.

"A certified high assurance operating system can dramatically reduce the cost and certification time of mobile devices."

"No matter how badly the Internet instance is compromised with viruses and Trojans, the malware cannot affect the user's critical instance."

Split Mobile Personalities

With secure virtualization technology, the mobile device can host multiple instances of mobile operating systems. For example, the device can incorporate one instance of Linux that the consumer uses for the phone function, e-mail, and other "critical" applications. A second instance of Linux can be used specifically for browsing the Internet. No matter how badly the Internet instance is compromised with viruses and Trojans, the malware cannot affect the user's critical instance. The only way for files to be moved from the Internet domain to the critical user domain is by using a secure cut and paste mechanism that requires human user interaction and cannot be spoofed or commandeered. A simple key sequence or icon is used to switch between the two Linux interfaces.

Secure virtualization can also be used to provide an MID with multiple operating system personalities, enabling service providers, phone manufacturers, and consumers to provide and enjoy a choice of environments on a single device. Furthermore, by virtualizing the user environment, personas (personal data, settings, and so on) can be easily migrated across devices, in much the same way that virtual machines are migrated for service provisioning in the data center.

In a recent article discussing the growth of mobile devices in corporate environments, *USA Today* stated that "mobile devices represent the most porous piece of the IT infrastructure." [10] The same problems that plague desktops and servers are afflicting mobile devices. Secure operating systems and virtualization technology provide a solution to the demand for enhanced security in the resource-constrained environment of portable consumer devices.

"Formerly isolated assets are being connected to networks at risk of cyber attack."

Gaming Systems

Gaming systems manufacturers are promoting the use of open network connectivity in next-generation gaming systems and properties. This vision provides for some exciting possibilities, yet the security challenges that arise in this architecture are not unlike other network-centric initiatives, such as the military's Global Information Grid (GIG): in both cases, formerly isolated assets are being connected to networks at risk of cyber attack. Clearly, gaming systems are an attractive target for well-resourced hostile entities.

The same hybrid virtualization architecture previously discussed can enhance user-to-game and game-to-server interactions. Secure communications components, including network security protocols and key management, can be securely partitioned away from the gaming multimedia environment (such as Linux, for example) which is hosted in a virtual machine using Intel VT. This is done in both the game console clients as well as in the servers, providing secure end-to-end encryption, authentication, and transaction verification.

"Secure communications components, including network security protocols and key management, can be securely partitioned away from the gaming multimedia environment."

Conclusion

In the past decade, virtualization has reemerged as a disruptive technology in the enterprise. However, due to resource constraints and different usage scenarios, virtualization has seen slower adoption in other areas of the computing world, in particular mobile and embedded systems. This is likely to change, due to two significant recent innovations. First, low power, Intel Atom processors now incorporate the same kind of hypervisor hardware acceleration enjoyed by desktop and server processors. Second, the advent of a powerful hybrid architecture incorporating certified high robustness security kernels, augmented with secure virtualization using Intel VT, represents a better fit for resource-constrained systems that often have rigorous safety, security, reliability, real-time, memory-efficiency, and/or power-efficiency requirements. The future for Intel VT-enabled applications is indeed bright.

References

[1] Larry Garfield. "'Metal Gear' Symbian OS Trojan disables anti-virus software." *http://www.infosyncworld.com/*, 2004.

[2] Whitaker, et al. "Denali: Lightweight Virtual Machines for Distributed and Networked Applications." USENIX Annual Technical Conference. 2002.

[3] Samuel King, et al. "SubVirt: Implementing malware with virtual machines." IEEE Symposium on Security and Privacy. 2006.

[4] Joanna Rutkowska. "Subverting Vista Kernel for Fun and Profit." Black Hat USA. 2006.

[5] Tavis Ormandy. "An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments." *http://taviso.decsystem.org/virtsec.pdf*, 2006.

[6] Denise Dubie. "Security concerns cloud virtualization deployments." *http://www.techworld.com/*, 2007.

[7] Joanna Rutkowska, Alexander Tereshkin, and Rafal Wojtczuk. "Detecting and Preventing the Xen Hypervisor Subversions;" "Bluepilling the Xen Hypervisor;" "Subverting the Xen Hypervisor." Black Hat USA. 2008.

[8] Common Criteria Validated Products List. *http://www.niap-ccevs.org/*, 2008.

[9] Moblin.org. *http://moblin.org*

[10] Byron Acohido, "Cellphone security seen as profit frontier." *http://www.usatoday.com/*, 2008.

Author Biography

David Kleidermacher: David Kleidermacher is chief technology officer at Green Hills Software where he has been responsible for operating system and virtualization technology over the past decade and has managed the team responsible for implementing Intel®-based solutions, including operating systems, hypervisors, and compilers. David helped launch the new Intel® vPro™ technology with Intel at Intel Developer Forum (IDF) in 2007, demonstrating the use of Intel® Virtualization Technology (Intel® VT) and Intel® Trusted Execution Technology (Intel® TXT). David can be contacted at davek at ghs.com.

Copyright

BUILDING AND DEPLOYING BETTER EMBEDDED SYSTEMS WITH INTEL® ACTIVE MANAGEMENT TECHNOLOGY (INTEL® AMT)

Contributor

Jose Izaguirre
Intel Corporation

Index Words

Intel® Active Management Technology
Intel® AMT
remote management
embedded
Point-of-Sale
POS
manageability
out-of-band
Serial-over-LAN
IDE-Redirection

"Automated teller machines, point-of-sale workstations, vending kiosks, self-checkout systems, slot machines, and airline check-in terminals are all examples of customer facing embedded systems used in retail establishments."

Abstract

Intel® Active Management Technology (Intel® AMT) is a technology intended to provide enhanced remote management of computing devices, primarily notebook and desktop PCs. But the benefits of Intel AMT extend far beyond the PC and are equally important to practically any industry that depends on customer facing computing equipment to run their day-to-day operations. Industries such as banking, retail, entertainment, and travel, for example, all rely on embedded computing equipment to run their businesses. For these industries and many others, mission critical equipment such as automated teller machines, point-of-sale (POS) workstations, slot machines, and airline check-in terminals, respectively, downtime means lost revenue. It is therefore paramount for the embedded computing equipment to be reliable, secure, highly available, and manageable. These are all fundamental attributes of Intel AMT and therefore make Intel AMT extremely valuable to a significant number of embedded applications.

This article provides a high level description of Intel Active Management Technology, explains some key benefits of the technology and presents a case study of how Intel AMT can be successfully applied to point-of-sale workstations to offer enhanced energy efficiency and advanced remote management capabilities to retail IT enterprises.

Introduction

Corporations have always looked to reduce costs and improve operational efficiency by employing technology to automate as many business processes as possible. The automation occurs at all levels of the enterprise but of particular importance, especially for retail businesses, is the automation that is customer facing, or in other words, the equipment with which the end customer interacts. This equipment is often a function-specific device also commonly referred to as an embedded system. Automated teller machines, point-of-sale workstations, vending kiosks, self-checkout systems, slot machines, and airline check-in terminals are all examples of customer facing embedded systems used in retail establishments. Of course, if the equipment is not operational it often means lost revenue for the business. Perhaps more importantly, however, it is the negative customer experience that is the most damaging. In retail, it is all about customer service, and a bad customer experience can impact customer loyalty and damage the corporate brand. Therefore, the retail IT enterprise must balance the deployment of customer-facing embedded systems that are cost effective yet very reliable and highly available. Intel® Active Management Technology (Intel® AMT) enhances embedded system performance on all of these metrics.

Overview of Intel® Active Management Technology (Intel® AMT)

Intel Active Management Technology is a hardware-based solution that uses out-of-band communication for management access to client systems. Intel AMT is one of the key technology ingredients of Intel® vPro™ technology, a platform brand of business optimized computers. In situations where a remote client system is inaccessible, such as when the machine is turned off, the hard disk drive has crashed or the operating system is hung, Intel AMT provides the mechanism by which a server running remote management software would be able to still access the client system and perform basic management tasks. Intel AMT is dedicated hardware contained within certain Intel® mobile and desktop chipsets, such as the Mobile Intel® GM45 Express chipset or the Intel® Q35/Q45 Express chipsets and which are also used in many embedded devices. Figure 1 describes the client side Intel AMT hardware components.

"Intel AMT is one of the key technology ingredients of Intel® vPro™ technology, a platform brand of business optimized computers."

At a high level, client-side Intel AMT is made up of the following components:

- Intel® Manageability Engine (Intel® ME) – a microcontroller-based subsystem that provides an out-of-band (OOB) management communication channel, maintains a TCP/IP stack and runs the Intel ME firmware. The Intel ME is the heart of Intel AMT and resides in the chipset's Memory Control Hub (MCH).
- Nonvolatile memory – persistent memory used to store the compressed Intel ME firmware as well as hardware and software information for IT staff to access using the OOB channel. This includes approximately 192 KB of third party data storage space (3PDS) for general purpose use by OEM platform software or third party software applications. The 3PDS space could optionally be use for encryption of sensitive data or secure keys. This nonvolatile memory resides in flash memory and is often combined onto a single SPI flash device along with the system's BIOS, Video BIOS, LAN ROM, and so on.
- System Memory – a portion of the system's main DRAM (channel 0) is used to run the decompressed Intel ME Firmware similar to what happens with system's BIOS. Intel ME requires DRAM channel 0 in order for the Intel ME to run and be initialized properly. If no memory is populated in channel 0 then Intel ME will be disabled.
- Intel AMT–capable networking hardware – specific Intel wired or wireless networking silicon with necessary hooks to support Intel AMT. These hooks implement filters that interact with inbound and outbound TCP/IP networking traffic.

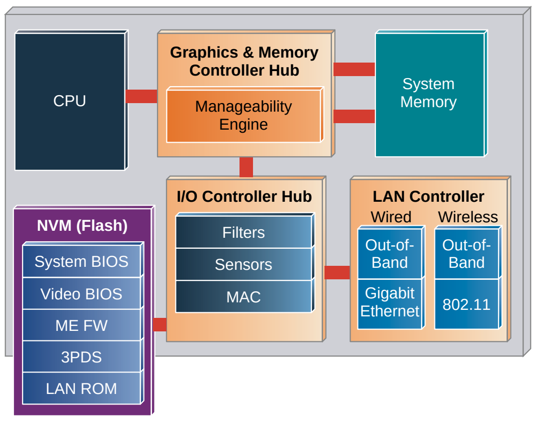


Figure 1: Intel® Active Management Technology (Intel® AMT) hardware architecture.
Source: Intel Corporation, 2008

Key Features for Embedded Applications

Intel AMT is designed with a complete set of management functions to meet the deployment needs of IT administrators. Let us take a closer look at just four key enabling features of Intel AMT of particular importance to Point-of-Sale as well as to other mission-critical embedded applications.

Out of Band Management

Prior to Intel AMT, remote management depended on the operating system as well

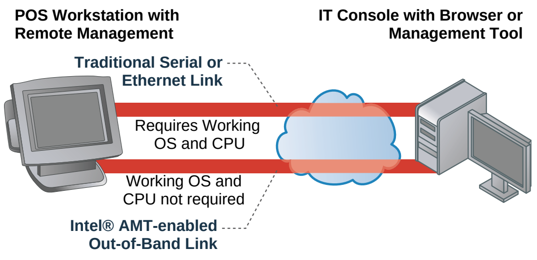

POS Workstation with Remote Management
IT Console with Browser or Management Tool
Traditional Serial or Ethernet Link
Requires Working OS and CPU
Working OS and CPU not required
Intel® AMT-enabled Out-of-Band Link

Figure 2: Out-of-band remote management
Source: Intel Corporation, 2008

as having a remote management software agent up and running on the client. If the operating system (OS) was locked up, then the software agent was prevented from working and the remote management capability was lost. Intel AMT provides a completely separate hardware subsystem that runs a dedicated TCP/IP stack and thus creates an "out-of-band" management communication channel. This capability makes it possible to inspect inbound/outbound packets before the OS has visibility to them. Effectively what you end up with is two logical network connections (one in-band, one out-of-band) using one physical RJ45 networking connector. This allows Intel AMT to offer a substantial number of management tasks that can significantly improve uptime and reduce maintenance costs. As illustrated in Figure 2, having a completely independent communication channel also allows for remote management functions to take place effectively 100 percent of the time and without regard to the state of the OS, such that blue screens and even powered down systems are still accessible by the help desk or IT personnel. Maintaining connectivity enables support personnel to more rapidly and accurately diagnose the failure condition, which in turn reduces the number of physical support visits.

"Serial-over-LAN (SOL) is a mechanism that allows the input and output of the serial port of the client system to be redirected using Internet Protocol (IP) to other computers on the network, in this case, the remote management server(s)."

Serial-over-LAN Redirection Capability

One of the key features of Intel AMT is its support for Serial-over-LAN redirection. Serial-over-LAN (SOL) is a mechanism that allows the input and output of the serial port of the client system to be redirected using Internet Protocol (IP) to other computers on the network, in this case, the remote management server(s). With Serial-over-LAN, the POS client's text-based display output could be redirected to the remote management console. This allows the help desk see the remote client's Power On Self Test (POST) sequence or navigate and control the client's BIOS settings.

IDE Redirection Capability

IDE Redirection (IDER) allows an administrator to redirect the client's IDE interface to boot from an image, floppy, or CD device located in or accessible by the remote management server. Once an IDER session is established, the managed client can use the server device as if it were directly attached to one of its own IDE channels. Intel AMT registers the remote device as a virtual IDE device on the client. This can be useful for remotely booting an otherwise unresponsive computer. A failing client, for example, could be forced to boot from a diagnostic image anywhere on the network. The administrator could then take action and perform any operation, ranging from a basic boot sector repair to a complete reformatting of the client disk thereby restoring the client back to a working state.

"Once an IDER session is established, the managed client can use the server device as if it were directly attached to one of its own IDE channels."

Both SOL and IDER may be used together.

Security

Is Intel AMT secure? This is an important question that is often asked in the early stages of Intel AMT evaluation, especially for organizations handling personal information or financial transactions. This is the case with many embedded systems such as ATMs and point-of-sale workstations. Intel AMT integrates comprehensive security measures to provide end-to-end data integrity, both within the client as well as between the client and the remote management server(s). IT administrators can optionally encrypt all traffic between the management console and the Intel AMT clients. This encryption is based on standard Secure Socket Layer (SSL)/ Transport Layer Security (TLS) encryption protocols that are the same technologies used today on secure Web transactions. Each major component of the Intel AMT framework is protected.

Intel® Manageability Engine Firmware Image Security

Only firmware images approved by Intel can run on the Intel AMT subsystem hardware. The signing method for the flash code is based on public/private key cryptography. The Intel AMT firmware images are encrypted using a firmware signing key (FWSK) pair. When the system powers up, a secure boot sequence is accomplished by means of the Intel ME boot ROM verifying that the public FWSK on flash is valid, based on the hash value in ROM. If successful, the system continues to boot from flash code.

Network Traffic Security

Network security is provided by the industry standard SOAP/HTTPS protocol, which is the same communication security employed by leading e-commerce and financial institutions. They cannot be changed.

Network Access Security

Intel AMT supports 802.1x network access security. This allows Intel AMT to function in network environments requiring this higher level of access protection. This capability exists on both the Intel AMT-capable wired and wireless LAN interfaces.

"Encryption is based on standard Secure Socket Layer (SSL)/ Transport Layer Security (TLS) encryption protocols that are the same technologies used today on secure Web transactions."

Available authentication methods include:

- Transport Layer Security (TLS)
- Tunneled Transport Layer Security (TTLS)
- Microsoft Challenge Handshake Authentication Protocol version 2 (MS-CHAP v2)
- Protected Extensible Authentication Protocol (PEAP)
- Extensible Authentication Protocol (EAP)
- Generic Token Card (GTC)
- Flexible Authentication via Secure Tunneling (FAST)

Intel AMT also supports combination of authentication methods such as EAPFAST TLS, PEAP MS-CHAP v2, EAPFAST MS-CHAP v2, EAP GTC, and EAPFAST GTC.

These key attributes of Intel AMT can be utilized and designed into embedded platforms to enhance the product's reliability, manageability, and serviceability.

"NCR had existing remote management solutions but looked forward to enhancing their offerings by leveraging the OOB capabilities."

NCR* Case Study

NCR Corporation is a global technology company and leader in automated teller machines, as well as self- and assisted-service solutions including point of sale. Early in the development of Intel AMT, NCR recognized the potential this technology had for its customer base so the company began to explore how it could incorporate the technology and apply it to its hardware and software products. NCR had existing remote management solutions but looked forward to enhancing their offerings by leveraging the OOB capabilities in Intel AMT to increase the number of issues that could be fixed remotely thus decreasing the number of expensive field visits. NCR thoroughly reviewed the Intel AMT feature set and decided to take a phased approach to enable its own remote management solution, called NCR* Retail Systems Manager, to support Intel AMT. The objective was to start in the first release with a subset of the overall Intel AMT features most easily implemented by their end customers then build from there and add additional Intel AMT capabilities over time.

Why Intel® Active Management Technology (Intel® AMT) for Point-of-Sale Workstations?

NCR saw several benefits in Intel AMT that would allow the organization to make huge strides in operational efficiency by a) reducing "truck rolls" b) increasing accuracy of problem resolution and c) improving help desk productivity. NCR was initially attracted to the power control capabilities of Intel AMT for remote control of unattended remote POS terminals as well as for the opportunity for power savings during off hours. NCR's service organization also reviewed its service call records and realized that Intel AMT could potentially make a significant impact on servicing POS terminal hard disk drive failures. The failure analysis reports revealed that hard disk drives were one of the top failing hardware components besides fans and certain peripherals attached to the POS like receipt printers and scanners; however, a significant percentage of returned hard disk drives were later found to be in perfect working order. While the problem appeared as a disk failure, in most cases the root cause was a corrupted file or other software problem and not a hardware problem at all. Immediately NCR realized the hard disk drive "false" failures could easily be reduced by employing out-of-band management and running remote disk diagnostics via IDE redirection thus verifying if the drive was indeed bad prior to sending out a field engineer. The total cost of ownership (TCO) value derived from Intel AMT is compelling. A recent study by Global Retail Insights finds the cost savings from advanced manageability (improvements in service calls, power-off automation, and asset deployment/tracking) to be approximately USD 205 per POS terminal per year.[1] Over a typical 7 year asset life, the advanced manageability benefit amounts to nearly 60 percent of the hardware acquisition cost.

"While the problem appeared as a disk failure, in most cases the root cause was a corrupted file or other software problem and not a hardware problem at all."

Point of Sale Clients

The Intel AMT enabled clients in this case are point–of-sale workstations as well as self service kiosks supporting a mix of Intel AMT v2.2 on Intel® Q965 Express chipset platforms as well as Intel AMT v4.0 on Mobile Intel GM45 Express chipset platforms, both chipsets are part of Intel's embedded long-life roadmap. NCR's

POS and kiosk products are manufactured in Asia through a contract manufacturer who pre-configures the systems' flash image according to NCR specifications. Enterprise mode was chosen as the default configuration due to the fact that most NCR customers for this line of POS and kiosk product are large retailers with centralized IT organizations.

Retail Enterprise

The retail IT enterprise system architecture and infrastructure varies depending on the size of the retailer and the number of POS workstations. A small neighborhood convenience store may only have 1 POS while a large department store chain may have thousands of stores each with 30 or more POS terminals. Figure 3 represents a typical retail IT infrastructure architecture. Many large retail IT enterprises are centralized and maintain their own IT help desk. Remote management services, leveraging Intel AMT, could be provided by either the retailer's IT organization or outsourced to a third party or even a mixture of both. Intel AMT requires certain ports to be "open" in order to allow management traffic to go through them. The Intel AMT ports are 16992 (non-TLS), 16993 (TLS), 16994 (non-TLS redirection), 16995 (TLS redirection) and 9971. Port 9971 is the default provisioning port used to listen for "hello" packets from Intel AMT clients. These ports have been assigned to Intel by the Internet Assigned Numbers Authority (IANA) but can be used by the customer's IT organization, third party remote management service providers, or equipment manufacturers. In NCR's case, the ability to enhance their remote management solutions with Intel AMT allows the company to offer a more competitive and profitable solution, which therefore allows NCR to grow their services business. NCR estimates the addressable services market for the industries they serve to grow to USD 8.2 billion by 2011.[5]

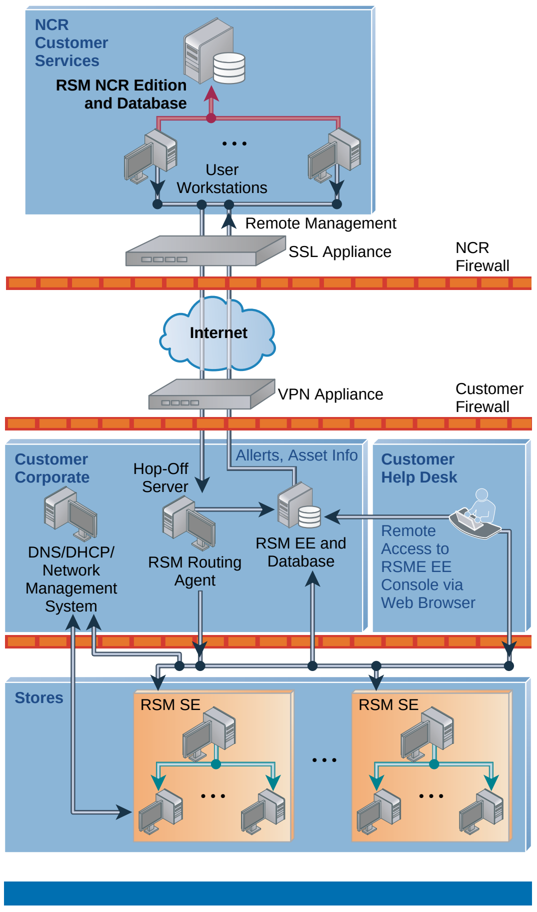


Figure 3: Typical retail IT enterprise system architecture. Source: NCR Corporation, 2009

NCR* Retail Systems Manager

The NCR Retail Systems Manager (RSM) is a software package for monitoring retail POS workstations, peripherals and applications. RSM operates independently from the POS application and provides remote access, 24/7 remote monitoring and alerting, remote diagnostics, and remote resolution through a user friendly Web-based interface.

There are three versions of RSM: Local, Site, and Enterprise Editions. RSM Local Edition (RSM LE) resides on the POS workstations themselves and provides local diagnostics capability; RSM Site Edition (RSM SE) serves as the in-store monitoring point; while RSM Enterprise Edition (RSM EE) provides same functionality as Site Edition but adds centralized management as well as third party management capability. All three versions have been modified to support Intel AMT.

RSM LE

RSM LE runs locally on the terminal and is targeted for standalone, non-networked clients or for attended operations at the client. It provides the ability to configure the POS workstation and its peripherals and to run basic diagnostics. RSM LE can be used by the customer to configure and diagnose problems on an individual client or POS workstation.

Once a valid RSM license file is detected, RSM LE assumes two additional functions. The first is to be an agent that feeds information upward in the RSM architecture and allows control of the client via RSM. The second is to awaken a state processing engine that manages the terminal and peripherals through states that are predefined for customers.

"Remote management can be performed from a server within the store or from a remote location such as the retailer's helpdesk, store, or headquarters."

RSM SE

RSM SE runs on a store server and provides the important role of traffic routing and store-level management. It provides the ability to manage groups of terminals or individual terminals within the store. RSM SE is accessible via a web browser both in the store and from RSM Enterprise Edition. The web browser can be running locally on the RSM SE server or remotely from any other server or workstation within the network. Therefore, remote management can be performed from a server within the store or from a remote location such as the retailer's helpdesk, store, or headquarters.

For those environments that do not have a store server, RSM LE and RSM SE have been certified to run in a workstation-server configuration on the same workstation.

RSM EE

"RSM EE provides an estate wide view of the terminal and peripheral assets in terms of asset information and state-of-health."

RSM EE runs on an enterprise server in conjunction with a Microsoft SQL Server database. RSM EE provides an estate wide view of the terminal and peripheral assets in terms of asset information and state-of-health. RSM EE also provides a graphical user interface for navigation in the retailer's estate of stores and terminals.

Intel® Active Management Technology (Intel® AMT) Enabling and Provisioning

NCR's RSM product was an existing member of the company's remote management solution and preceded Intel AMT, so in order for RSM to become capable of implementing Intel AMT, it was necessary for NCR to make modifications to RSM and develop an Intel AMT plug-in for their existing remote management software. NCR accomplished this by making use of the AMT Software Development Kit (SDK)[2]. This SDK contains a Network Interface Guide, which includes all of the necessary APIs for RSM to be able to communicate with and send specific commands to the Intel Manageability Engine on the POS workstations. NCR software engineers added support for the Intel AMT APIs into the RSM product. This required minor architectural changes to RSM based on the fact it now had to perform certain tasks within the context of Intel AMT[6]. These tasks, for example, included the "zero touch" remote configuration functionality, where the server can provision the Intel AMT–enabled client without the need to physically touch the client in the process. Remote configuration can therefore be performed on "bare-bones" systems, before the OS and/or software management agents are installed. Remote configuration allows the retailer to purchase and install the equipment and then set up and configure the Intel AMT capability at a later date without incurring the higher costs of physically touching every machine already deployed.

"Remote configuration allows the retailer to purchase and install the equipment and then set up and configure the Intel AMT capability at a later date without incurring the higher costs of physically touching every machine already deployed."

Once both the client hardware and remote management console software are ready for Intel AMT and the customer has deployed the necessary equipment, the next phase is provisioning the equipment in the IT enterprise. Provisioning refers to the process by which an Intel AMT client is configured with the attributes necessary for the client to become manageable within a specific IT environment. There are two modes of Intel AMT provisioning: Small Business Mode (less complex and suitable for small volume deployments) and Enterprise Mode (more complex and suitable for large volume deployments). A typical large centralized retailer employing Intel AMT Enterprise Mode would provision for Intel AMT as follows:

- Pre-shared secrets are generated and associated to the provisioning server.
- The pre-shared secrets are distributed to the Intel Management Engine (Intel ME).
- With the Intel Management Engine in setup mode, an IP address and associated DHCP options are obtained.
- The Intel Management Engine requests resolution of "ProvisionServer" based on the specified DNS domain.
- POS Intel AMT enabled client sends "hello" packet to ProvisionServer. Domain_Name.com upon connecting to network
- Provisioning requests are received by provisioning server (handled by either RSM EE or RSM SE depending on customer configuration).
- The POS Intel AMT client and provisioning server exchange keys, establish trust, and securely transfer configuration data to the Intel AMT client

For more detailed descriptions, please refer to the *Intel® Active Management Technology (Intel® AMT) Setup and Configuration Service Installation and User Manual.*[3]

"Provisioning refers to the process by which an Intel AMT client is configured with the attributes necessary for the client to become manageable within a specific IT environment."

Target Usage Models

There are three basic usage models in which Intel AMT plays a central role: remote power control, remote repair, and remote asset and software management. All three models have direct cost-saving advantage for both the equipment manufacturer as well as the IT enterprise.

Remote Power On/Off Power Savings

Many retailers today leave their machines up and running during store off-hours for a number of reasons, such as the potential for deployment of software patches, the inconvenience of having people manually turn the machines off or the time required for the machines to fully become operational when business resumes the next day. Also, while some companies enable sleep states while the machine is idle, the reality is that most POS in the field today remain fully powered even when the system is not in use. Intel AMT may be utilized to automatically and remotely power down the POS clients during store off-hours and then remotely power them back up before store employees arrive the next business day to reopen the store. The study by Global Retail Insights mentioned earlier finds that a retailer with 200 stores, 10 POS workstations per store and operating 14 hours per day, 360 days per year could save approximately USD 162,000 annually simply by implementing power-off automation.[1] Also, if you consider that hardware using Intel AMT is inherently more energy efficient due to the newer technology microprocessors and chipsets, and that it takes approximately 3 watts of power to cool the store for every 1 watt of power

"Intel AMT may be utilized to automatically and remotely power down the POS clients during store off-hours and then remotely power them back up before store employees arrive the next business day to reopen the store."

"Retailers could realize an additional 70-percent reduction in terminal cooling costs. This equates to an additional USD 120,000 per year according to Global Retail Insights."

that is placed into the store, retailers could realize an additional 70-percent reduction in terminal cooling costs. This equates to an additional USD 120,000 per year according to Global Retail Insights. Thus, implementing remote power down during off hours could potentially save USD 282,000 per year (USD 162,000000 + 120,000). Over an asset life of 7 years, the savings adds up to nearly USD 2,000,000.

Remote power on/off automation can be implemented using Intel AMT by simply sending the encrypted power on/off command from the IT management console at predetermined times that can be programmed into the console. Intel AMT supports power on, power off, and power cycle (power off then back on) commands. IT personnel may also remotely manage the clients when in sleep modes S3 (suspend to RAM) or S4 (suspend to disk) as long as the clients remain plugged into AC power and connected using wired networking (wireless power policies place greater priority on battery life and therefore shut down the Intel ME). This allows for even further reductions in energy consumption since in most retail environments there is a considerable amount of time when the machine is idle and not in use.

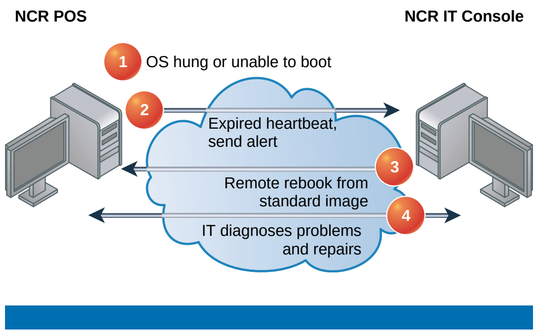

NCR POS
NCR IT Console
1 OS hung or unable to boot
2 Expired heartbeat, send alert
3 Remote rebook from standard image
4 IT diagnoses problems and repairs

Figure 4: Remote diagnostics and repair sequence. Source: NCR Corporation, 2008

Remote Diagnostics and Repair

Another important use case for the retail IT enterprise is the ability to perform remote diagnostics and repair. As stated earlier, if the machines are down, the company is most likely not making money. In many cases a machine may be unable to boot the operating system due to a number of reasons such as missing or corrupt OS files, drivers, or registry entries. NCR RSM can leverage the power control capability in Intel AMT to power cycle the machine, employ IDER to boot from a remote image containing a small OS such as DOS, and then run diagnostic software to pinpoint the problem. In the same fashion, IT personnel can push updated drivers at runtime and patch them into the main OS. Figure 4 illustrates the sequence.

Preventive maintenance is another area where Intel AMT adds significant value, particularly for mission critical equipment. The ability to predict when a component might fail and take action prior to it failing is a tremendous benefit. The 3PDS area of an Intel AMT–enabled POS workstation, for example, can be used to store information about field replaceable system components. Peripheral component information such as manufacturer, model numbers, and serial numbers, as well as information like the number of hours the power supply is on, the number of lines a receipt printer has printed, the number of card swipes a magnetic stripe reader has read, or the number of times the solenoid of a cash drawer has fired could all be tracked. Thresholds can be set according to historical reliability data so that alerts can go back to the Intel AMT–enabled remote console and allow the service personnel to take action before the component actually fails and the service can be performed at a convenient time for the customer. Global Retail Insights reports that a conservative 15-percent reduction in physical service calls can save approximately USD 108,000 per year.

"The number of lines a receipt printer has printed, the number of card swipes a magnetic stripe reader has read, or the number of times the solenoid of a cash drawer has fired could all be tracked."

Remote Asset and Software Management

Tracking important system information such as equipment location, serial numbers, asset numbers, and installed software are extremely important to an IT organization. Having this information readily available allows the enterprise to better control their hardware and software inventory as well as manage software patches and licensing.

Intel AMT allows IT administrators to reduce the support costs and keep their systems operating at peak performance by ensuring their clients have the latest software updates. With Intel AMT, software patches and updates can be scheduled during times that minimize impact to the business such as store off hours or off-peak times. The remote console could also be designed to support mass deployment of software update distribution following a one-to-many model (from one management console to many remote clients simultaneously). This is a key benefit for a retail enterprise because it allows for software image uniformity required to deliver consistent device behavior and customer service. A one-to-many deployment model allows IT administrators to create groups or collections of Intel AMT enabled clients and then distribute BIOS or software updates with a single command to all clients within the group, thereby significantly reducing the time and cost it takes to make BIOS changes over a wide range of terminals.

"With Intel AMT, software patches and updates can be scheduled during times that minimize impact to the business such as store off hours or off-peak times."

Challenges in Activating Intel® Active Management Technology (Intel® AMT)

While there are substantial benefits to be gained from Intel AMT, there are also a number of challenges to deal with. The good news is that these challenges can certainly be overcome with some up front planning and infrastructure preparation. Once an IT enterprise gains a basic understanding of the technology and its potential benefits and decides to move forward with Intel AMT activation, the following are a few things for the organization to consider:

"The good news is that these challenges can certainly be overcome with some up front planning and infrastructure preparation."

Establish goals and objectives – the organization should outline what it wants to accomplish and set appropriate objectives to meet both short term and long term goals. Define which Intel AMT features will be implemented and in what timeframe. Start small then build from there.

Measure benefits – the organization should determine the key metrics to measure before and after Intel AMT activation; for example, percentage energy savings or percentage reduction in physical support visits or percentage reduction in total support costs, so that benefits can be quantified and then determine if a positive ROI exists.

Define enterprise infrastructure impact – Implementing Intel AMT often means doing things a little differently. The organization should ask: Is the necessary infrastructure in place? (DNS/DHCP servers, provisioning server, keys/certificates, remote management console that supports desired implementation features). What internal processes need to change to support this technology?

"Implementing Intel AMT often means doing things a little differently."

Define the appropriate security level for the customer environment – Insufficient security allows for potential attacks or may expose sensitive financial or personal consumer data. However, too much security is more complex to implement and may require additional expertise.

Allocate appropriate resources – there is certainly a learning curve required to successfully implement Intel AMT and OEMs as well as retail IT must allow for adequate time and resources. There is an extensive number of tools, utilities, software, and documentation available to assist with the learning curve.

Conclusion

Intel AMT is a powerful technology with broad and direct applicability to customer-facing, mission-critical embedded equipment. Intel AMT can save power, reduce service calls, improve uptime, and reduce overall product maintenance and support costs. Intel AMT can deliver compelling total cost of ownership savings of approximately USD 200 per machine per year and lifecycle benefit equivalent to nearly 60 percent of the original purchase price. For mission critical embedded applications, Intel AMT in most cases delivers a positive return on investment and therefore becomes a key differentiator for the OEM. While implementing the technology is not a trivial task, with appropriate planning and preparation, it can be successfully integrated into embedded, mission-critical devices and deployed into the corresponding IT environment. Intel AMT serves as an enabler for companies like NCR to build better products and deliver proactive service intelligence ultimately leading to improvements in operational efficiency, profitability, and significant increases in customer service.

Acknowledgements

I would like to acknowledge and thank the technical reviewers of this article: Jerome Esteban, Dennis Fallis, and Mike Millsap for their valuable input. Special thanks also go to Alan Hartman and Roger Farmer of NCR Corporation for their support of this article as well as their many contributions to the successful development and deployment of Intel AMT technology in NCR products.

References

[1] S. Langdoc, Global Retail Insights, an IDC Company. "Advanced CPUs: The Impact on TCO Evaluations of Retail Store IT Investments." September 2008

[2] Intel® Active Management Technology (Intel® AMT) Software Development Kit, the reference for Intel AMT developers. *http://www.intel.com/software/amt-sdk*

[3] Intel® vPro™ Expert Center. *http://www.intel.com/go/vproexpert*

[4] Manageability Developer Tool Kit (DTK), a complete set of freely available Intel AMT tools and source code. *http://www.intel.com/software/amt-dtk*

[5] NCR* Analyst Day presentation, December 2008 *http://www.ncr.com*

[6] NCR correspondence

Author's Biography

Jose Izaguirre: Jose Izaguirre is part of Intel Corporation's Sales and Marketing Group and has held the role of field applications engineer for the past 8 years. In this position he is responsible for driving strategic embedded systems customer engagements, participating in pre-sales technical activities, and providing post-sales customer technical support. Jose joined Intel following more than 10 years at NCR Corporation where he held a number of engineering roles that included POS and kiosk system architecture as well as motherboard architecture, design, and development. Jose received a bachelor's degree in electrical engineering from Vanderbilt University and also holds a master's of business administration degree from Georgia State University.

Copyright

IMPLEMENTING FIRMWARE FOR EMBEDDED INTEL® ARCHITECTURE SYSTEMS: OS-DIRECTED POWER MANAGEMENT (OSPM) THROUGH THE ADVANCED CONFIGURATION AND POWER INTERFACE (ACPI)

Contributors

John MacInnis
Intel Corporation

Abstract

A firmware component is essential for embedded systems using Intel® architecture designs.

Embedded Intel® Architecture designs must include a firmware stack that initializes the platform hardware and provides support for the operating system (OS). Power management is a design priority for both server equipment and battery-operated devices. The firmware layer plays a critical role in embedded system power management. OS-directed power management using the Advanced Configuration and Power Interface (ACPI) methodology is a solution that allows for cost-effective firmware development and quick time to market. Pushing state machine management and decision policies to the OS and driver layer allows post-production flexibility for tuning power management. This article explores how ACPI has provided efficiencies over APM and BIOS-directed power management and provides a condensed overview of APCI technology.

Index Words

Intel® Architecture
Firmware
Advanced Configuration and Power Interface (ACPI)
OS-Directed Power Management
Embedded

Introduction

Embedded systems using the Intel® architecture must include a firmware stack that initializes CPU cores, memory, I/O, peripherals, graphics, and provides runtime support for operating systems. While Intel architecture-based PC designs typically use a full BIOS solution as a firmware stack, many embedded systems are designed with a more optimized firmware layer known as a boot loader. The job of the boot loader is to quickly initialize platform hardware and boot the system to an embedded real-time operating system (RTOS) or OS. Until recently, many embedded operating systems were designed to boot the device and enable all the drivers and networking on the board with no power management per se.

"Many embedded systems are designed with a more optimized firmware layer known as a boot loader."

As Intel architecture expands into more differentiated types of embedded systems, power management becomes increasingly important both for saving electricity costs as well as maximizing battery life in mobile systems.

"For system developers, an ACPI design can help yield full PM control with quick time to market and cost savings."

OS-directed Power Management (OSPM) using ACPI methodology provides an efficient power management option. For system developers, an ACPI design can help yield full PM control with quick time to market and cost savings. It offers flexibility by pushing state machine management and policy decisions to the OS and driver layer. The OS creates policy decisions based on system use, applications, and user preferences. From a maintenance and support perspective, patches, updates and bug fixes are better managed at the OS and driver layer than in the firmware.

A Note About Firmware Terminology

Since the first IBM* clones in the early 1980s, the PC BIOS has been the predominant firmware layer in most of Intel architecture system designs commonly referred to as x86. It has been observed that many Embedded Intel® Architecture product designers have unique requirements not always completely satisfied by the standard PC BIOS. This paper uses the terms firmware and boot loader to denote the distinct differences between a PC BIOS and the hybrid firmware required for many of today's embedded systems.

"Many Embedded Intel® Architecture product designers have unique requirements not always completely satisfied by the standard PC BIOS."

Dynamic System Power Management

Many types of embedded systems built on Intel architecture are necessarily becoming more power-savvy. Implementing power management involves complex state machines that encompass every power domain in the system. Power domains can be thought of globally as the entire system, individual chips, or devices that can be controlled to minimize power use, as illustrated in the diagram in Figure 1.

Power and Thermal Management States

G0, G1, G2, and G3 signify global system states physically identifiable by the user

G3 – Mechanical Off

G2 – Soft Off

G1 – Sleeping

G0 - Working

S0, S1, S2, S3, S4 signify different degrees of system sleep states invoked during G1.

D0, D1,…, Dn signify device sleep states. ACPI tables include device-specific methods to power down peripherals, while preserving Gx and Sx system states; for example, powering down a hard disk, dimming a display or powering down peripheral buses when they are not being used.

C0, C1, C2, C3, and C4 signify different levels of CPU sleep states. The presumption is that deeper sleep states save more power at the tradeoff cost of longer latency to return to full on.

P0, P1, P2,…, Pn signify CPU performance states while the system is on and the CPU is executing commands or in the C0 state.

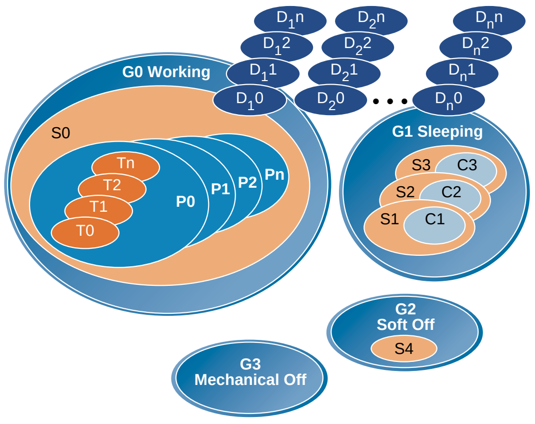


Figure 1: System power state diagram.
Source: Intel Corporation, 2009

"The presumption is that deeper sleep states save more power at the tradeoff cost of longer latency to return to full on."

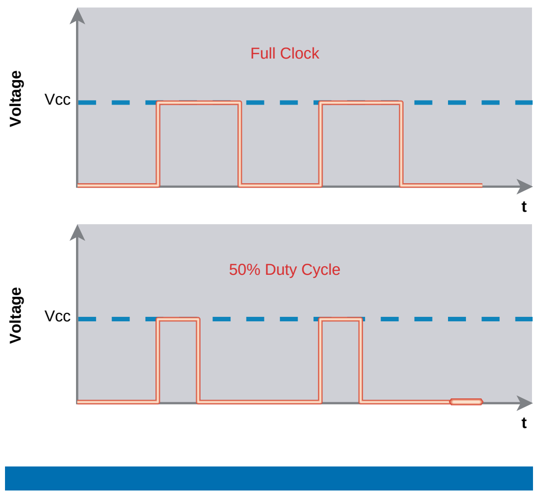


Figure 2: Clock throttling.
Source: Intel Corporation, 2009

T0, T1, T2,…, Tn signify CPU-throttled states while the CPU is in the P0 operational mode. Clock throttling is a technique used to reduce a clock duty cycle, which effectively reduces the active frequency of the CPU. The throttling technique is mostly used for thermal control. Throttling can also be used for things such as controlling fan speed. Figure 2 shows a basic conceptual diagram of a clock throttled to 50 percent duty cycle.

Power Consumption and Battery Life

Power consumption is inversely related to performance, which is why a handheld media player can play 40 hours of music but only 8 hours of video. Playing video requires more devices to be powered on as well as computational CPU power. Since battery life is inversely proportional to system power draw, reducing power draw by 50 percent doubles the remaining battery life, as shown in Equation1.

$$\text{Remaining Battery Life (h)} = \frac{\text{Remaining Capacity (Wh)}}{\text{System Power Draw (W)}} \quad (1)$$

System PM Design Firmware – OS Cooperative Model

In Intel architecture systems, the firmware has unique knowledge of the platform power capabilities and control mechanisms. From development cost and maintenance perspectives, it is desirable to maintain the state machine complexity and decision policies at the OS layer. The best approach for embedded systems using Intel architecture is for the firmware to support the embedded OS by passing up control information unique to the platform while maintaining the state machine and decision policies at the OS and driver layer. This design approach is known as OS-directed power management or OSPM.

Under OSPM, the OS directs all system and device power state transitions. Employing user preferences and knowledge of how devices are being used by applications, the OS puts devices in and out of low-power states. The OS uses platform information from the firmware to control power state transition in hardware. APCI methodology serves a key role in both standardizing the firmware to OS interface and optimizing power management and thermal control at the OS layer.

"BIOS-based power management engines are costly to implement and offer little in the way of flexibility in the field or at the OS layer."

Advantages of ACPI over Previous Techniques

Before ACPI technology was adopted, Intel architecture systems first relied on BIOS-based power management schemes and then later designs based on Advanced Power Management (APM).

BIOS-based Power Management

BIOS-based power management engines are costly to implement and offer little in the way of flexibility in the field or at the OS layer. In BIOS-based power management, a PM state machine was designed and managed inside the BIOS firmware and then ported for each specific platform. The BIOS relied on system usage indicators such as CPU cache lines, system timers, and hardware switches to determine PM state switching triggers. In this scheme the validation and testing phase was fairly complex. Updating firmware in the field is a nontrivial task and riskier than installing OS patches or updating drivers. Once the BIOS-driven power management engine shipped in a product it was difficult to modify, optimize or fix compatibility bugs. In the field systems could and sometimes did unexpectedly hang due to insufficient system monitoring or incompatibility with OS and runtime applications.

Advanced Power Management (APM)

In the 1990s APM brought a significant improvement by adding a rich API layer used for a more cooperative model between the OS and the BIOS. Using APM the OS was required to call the BIOS on a predetermined frequency in order to reset counters thereby indicating system use. APM also employed APIs to allow the OS to make policy decisions and make calls into the BIOS to initiate power management controls.

"APM expanded power management choices to scalable levels of sleep states to balance power savings and wake latency."

APM expanded power management choices to scalable levels of sleep states to balance power savings and wake latency. It also allowed for power managing devices independently. For example the OS could elect to put the hard drive in sleep mode while keeping the rest of the system awake.

APM was an improvement in overall system PM capability and allowed for better management from the OS but it had the negative effect of requiring the BIOS to maintain a more complex state machine than before. This involved increased development and testing/validation costs. When quality and unexpected errors occurred they were difficult and costly to fix downstream from the factory where the BIOS is typically finalized. The APM scheme was responsible for many infamous "blue screens," which were challenging to work around in the field and sometimes required BIOS field upgrades, a costly and somewhat risky operation.

Advanced Configuration and Power Interface (ACPI)

APCI solved many problems by creating a scheme where the BIOS or embedded boot loader firmware is only responsible for passing its knowledge of the hardware control mechanisms and methods to the OS while pushing state machine management and PM policy decisions to the OS and driver layer. APCI methodology can simplify the BIOS or firmware implementation and testing cycle. Instead of testing a complex state machine, firmware validation can cycle through forced system states exercising all ACPI control methods and verify correct operation and desired results from the hardware, as illustrated in Figure 4. This can save significant time and cost at the BIOS and firmware design center and can help achieve greater quality objectives at the firmware layer which in turn eliminate costly and risky BIOS or firmware upgrades in the field.

ACPI Overview

First published in 1999, the Advanced Configuration and Power Interface (ACPI) specification is an open industry specification co-developed by Hewlett-Packard,* Intel, Microsoft,* Phoenix,* and Toshiba.* [1]

The ACPI specification was developed to establish industry common interfaces enabling robust OS-directed motherboard device configuration and power management of both devices and entire systems. In compliant systems, ACPI is the key element in operating system–directed configuration and Power Management (OSPM). ACPI evolves a preexisting collection of power management BIOS code. Advanced Power Management (APM) application programming interfaces (APIs), PNPBIOS APIs, multiprocessor specification (MPS) tables and so on into a well-defined power management and configuration interface specification. ACPI remains a key component of later Universal Extensible Firmware Interface (UEFI) specifications.

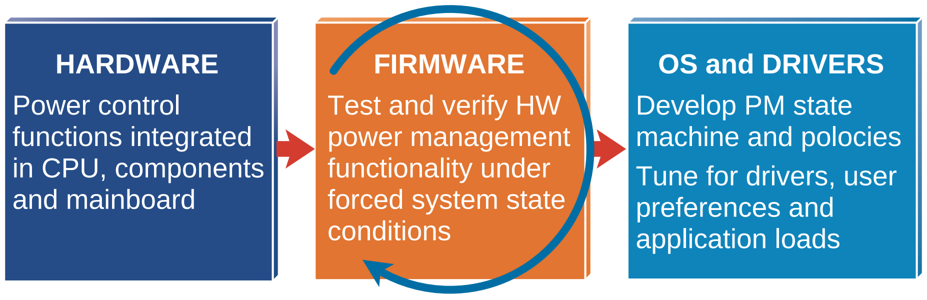

HARDWARE
Power control functions integrated in CPU, components and mainboard
FIRMWARE
Test and verify HW power management functionality under forced system state conditions
OS and DRIVERS
Develop PM state machine and polocies
Tune for drivers, user preferences and application loads

Figure 3: System power management development and operational phases
Source: Intel Corporation, 2009

ACPI specifications define ACPI hardware interfaces, ACPI software interfaces, and ACPI data structures. The specifications also define the semantics of these interfaces. ACPI is not a software specification; it is not a hardware specification, although it addresses both software and hardware and how they must behave. ACPI is, instead, an interface specification comprised of both software and hardware elements, as shown in Figure 4.

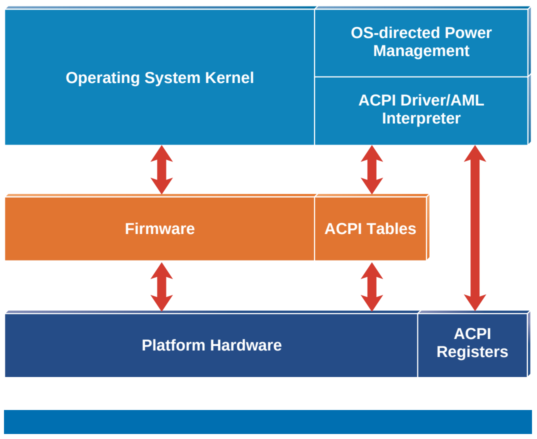


Figure 4: ACPI system architecture.
Source: Intel Corporation, 2009

Firmware creators write definition blocks using the ACPI control method source language (ASL) and operating systems use an ACPI control method language (AML) interpreter to produce byte stream encoding.

ACPI Operation Overview

The way ACPI works is that the firmware creates a hierarchical set of objects that define system capabilities and methods to control system hardware. The APCI objects are then passed to the operating system in a well defined handshake during OS boot. The OS loads the ACPI objects into its kernel and then uses the information along with OS level system drivers to define and execute dynamic hardware configuration, thermal management control policies, and power management control policies.

"Firmware creates a hierarchical set of objects that define system capabilities and methods to control system hardware."

Before OS boot, the firmware places a series of pointers and table arrays in memory for the OS to locate. During boot the OS searches for a signature indicating presence of a root system description pointer (RSDP). The pointer is found either by scanning predefined memory space for the signature, "RSD PTR" or through and Extensible Firmware Interface (EFI) protocol. In the case of an EFI-compliant system, the RSDP is detected through the presence of a unique GUID in the EFI System Table, which specifies the location of the RSDP.

Root System Description Pointer (RSDP)

The RSDP contains a 32-bit pointer to the Root System Description Table (RSDT) and/or a 64-bit pointer to the Extended System Description Table (XSDT). The RSDT and the XSDT hold equivalent data, one for 32-bit systems and the other for 64-bit systems respectively. In this way a single firmware image can support both 32- and 64-bit operating systems.

"A single firmware image can support both 32- and 64-bit operating systems."

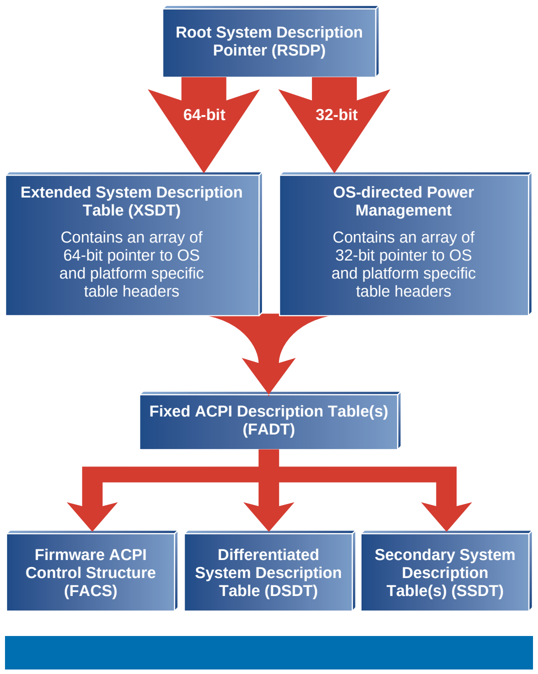

Root System Description Pointer (RSDP)
64-bit
32-bit
Extended System Description Table (XSDT)
Contains an array of 64-bit pointer to OS and platform specific table headers
OS-directed Power Management
Contains an array of 32-bit pointer to OS and platform specific table headers
Fixed ACPI Description Table(s) (FADT)
Firmware ACPI Control Structure (FACS)
Differentiated System Description Table (DSDT)
Secondary System Description Table(s) (SSDT)

Figure 5: ACPI firmware table structure.
Source: Intel Corporation, 2009

Root System Description Table (RSDT)

The RSDT/XSDT tables point to platform-specific table headers, which in turn contain platform-specific ACPI objects. The ACPI firmware table structure is illustrated in Figure 5. One such table is the Fixed ACPI Description Table (FADT) which contains the Firmware ACPI Control Structure (FACS) and pointer to the Differentiated System Description Table (DSDT).

Differentiated System Description Table (DSDT)

The DSDT contains information for base support of the platform including objects, data, and methods that define the platform hardware and how to work with it including power state transitions. The DSDT is unique and always loaded in the OS kernel and once loaded cannot be unloaded during the runtime cycle of the system. Secondary System Description Tables (SSDTs) can be included to augment the DSDT or differentiate between platform SKUs. The SSDTs cannot replace the DSDT or override its functionality.

The ACPI Name Space

Using the table data, the OS creates what is known as the ACPI namespace, which becomes part of the runtime kernel. The ACPI namespace is a hierarchical tree structure of named objects and data used to manage dynamic hardware configuration and to create and execute power and thermal management policies.

The information in the ACPI namespace comes from the DSDT, which contains the Differentiated Definition Block, and one or more other definition blocks. A definition block contains information about the hardware in the form of data and control methods encoded in ACPI Machine Language (AML). A control method is a definition of how the OS can perform hardware related tasks. The firmware author writes control methods using ACPI Source Language (ASL) which is then compiled to AML using an Intel® ASL compiler.

ASL Programming Language

"ACPI Source Language (ASL) is a language for defining ACPI objects especially for writing ACPI control methods."

ACPI Source Language (ASL) is a language for defining ACPI objects especially for writing ACPI control methods. Firmware developers define objects and write control methods in ASL and then compile them into ACPI Machine Language (AML) using a translator tool commonly known as a compiler. For a complete description of ASL, refer to Chapter 17 of the ACPI Specification revision 3.0b. The following code provides an example of basic ASL code used to define and APCI definition block and some basic control methods.

```
// ACPI Control Method Source Language (ASL) Example
DefinitionBlock (
        "forbook.aml",          // Output Filename
        "DSDT",                         // Signature
        0x02,                           // DSDT Compliance Revision
        "OEM",                          // OEMID
        "forbook",                      // TABLE ID
        0x1000                          // OEM Revision
)
{       // start of definition block
        OperationRegion(\GIO, SystemIO, 0x125, 0x1)
        Field(\GIO, ByteAcc, NoLock, Preserve)     {
                CT01,   1,
        }
        Scope(\_SB)     {       // start of scope
                Device(PCI0) {          // start of device
                     PowerResource(FET0, 0, 0) {          // start of pwr
                                Method (_ON)    {
                                     Store (Ones, CT01)     // assert power
                                        Sleep (30)             // wait 30m
                                }
                                Method (_OFF) {
                                     Store (Zero, CT01)     // assert reset#
                                }
                                Method (_STA) {
                                        Return (CT01)
                                }
                        } // end of power
                } // end of device
        } // end of scope
} // end of definition block
```

CPU Power and Thermal Management

APCI is used to help implement CPU controls for both thermal and power management. Clock throttling for example is commonly used for passive thermal control, meaning without turning on fans. The CPU dissipates less heat when it is actively throttled. Switching CPU power states, known as Cx states, is commonly used to save power when the full performance capabilities of the CPU are not required.

"Clock throttling for example is commonly used for passive thermal control, meaning without turning on fans."

Processor Control Block

CPU control can be done through ACPI standard hardware using the processor control block registers named P_CNT, P_LVL2 and P_LVL3.

Processor Control (P_CNT) - The CLK_VAL field is where the duty setting of the throttling hardware is programmed as described by the DUTY_WIDTH and DUTY_OFFSET values in the FADT. Table 1 lists the processor control register bits.

Bit	Name	Description
0–3	CLK_VAL	Possible locations for the clock throttling value.
4	THT_EN	This bit enables clock throttling of the clock as set in the CLK_VAL field. The THT_EN bit must be reset to LOW when changing the CLK_VAL field (changing the duty setting).
5–31	CLK_VAL	Possible locations for the clock throttling value.

Table 1: Processor control register bits. Source: Advanced Configuration and Power Interface Specification; Revision 3.0b (2006)

Table 2 shows CPU clock throttling information. Writes to the control registers allow for programming the clock throttling duty cycle.

Field	Byte Length	Byte Offset	Description
DUTY_OFFSET	1	104	The zero-based index of where the processor's duty cycle setting is within the processor's P_CNT register.
DUTY_WIDTH	1	105	The bit width of the processor's duty cycle setting value in the P_CNT register. Each processor's duty cycle setting allows the software to select a nominal processor frequency below its absolute frequency as defined by: THTL_EN = 1 BF * DC/(2DUTY_WIDTH) Where: BF–Base frequency DC–Duty cycle setting When THTL_EN is 0, the processor runs at its absolute BF. A DUTY_WIDTH value of 0 indicates that processor duty cycle is not supported and the processor continuously runs at its base frequency.

Table 2: FADT processor throttle control information. Source: Advanced Configuration and Power Interface Specification; Revision 3.0b (2006)

Processor LVL2 Register (P_LVL2) - The P_PVL2 register is used to transition the CPU into the C2 low power state. Similarly the P_PVL3 register is used to transition the CPU to the C3 low power state and so on. In general, a higher number means more power savings at the expense of conversely longer wake time latency. Table 3 describes the P_LVL2 control for invoking the C2 state.

Bit	Name	Description
0-7	P_LVL2	Reads to this register return all zeros; writes to this register have no effect. Reads to this register also generate an "enter a C2 power state" to the clock control logic.

Table 3: Processor LVL2 register bits. Source: Advanced Configuration and Power Interface Specification; Revision 3.0b (2006)

CPU Throttling Control through Software

CPU throttling control can be directed through software using CPU control methods written in ASL and passed to the operating system through the DSDT or SSDT. Primary control methods include the following.

_PTC (Processor Throttling Control) - Defines throttling control and status registers. This is an example usage of the _PTC object in a Processor object list:

```
  Processor (
          \_SB.CPU0,               // Processor Name
          3,                               // ACPI Processor number
          0x120,                   // PBlk system IO address
          6 )                              // PBlkLen
   { //Object List

          Name(_PTC, Package ()             // Processor Throttling Control object
          {
                  ResourceTemplate(){Register(FFixedHW, 0, 0, 0)},
// Throttling_CTRL
                  ResourceTemplate(){Register(FFixedHW, 0, 0, 0)}
// Trottling_STATUS
          }) // End of _PTC object
   } // End of Object List
```

_TSS (Throttling Supported States) – Defines a table of throttling states and control/status values. This is an example usage of the _TSS object in a Processor object list:

```
Name (_TSS, Package()
{                 // Field Name                               Field
Type
          Package ()
// Throttle State 0 Definition – T0
          {
                  FreqPercentageOfMaximum,            // %CPU core
freq in T0 state
                  Power,
// Max Power dissipation in mW for T0
                  TransitionLatency,                          // Worst
case transition latency Tx->T0
                  Control,
// Value to be written to CPU Ctrl register
                  Status
// Status register value after transition
          },
          .
          .
          .
                    Package ()
          // Throttle State n Definition – Tn
          {
                    FreqPercentageOfMaximum,                    // %CPU core
freq in Tn state
                    Power,
// Max Power dissipation in mW for Tn
                    TransitionLatency,                                // Worst
case transition latency Tx->Tn
                    Control,
// Value to be written to CPU Ctrl register
                    Status
// Status register value after transition
          }
}) // End of _TSS object
```

"ACPI includes well defined limits of firmware functionality that help yield high quality firmware while keeping production costs downs and time to market fast."

_TPC (Throttling Present Capabilities) - Specifies the number of currently available throttling states. Platform notification signals re-evaluation. This is an example usage of the _TPC object in a Processor object list:

```
   Method (_TPC, 0)                    // Throttling Present Capabilities method
                    {
                              If (\_SB.AC)
                              {
                                        Return(0)                  // All Throttle
States are available for use.
                              }
                              Else
                              {
                                        Return(2)                  // Throttle States 0
and 1 won't be used.
                              }
                    } // End of _TPC method
```

Conclusion

Today's embedded systems built on Intel architecture have distinctly different requirements from the standard PC BIOS. For embedded systems requiring power management, an ACPI based model is recommended. ACPI includes well defined limits of firmware functionality that help yield high quality firmware while keeping production costs downs and time to market fast. At the same time ACPI can enable very flexible and efficient power management. It is encouraged that embedded firm-ware or boot loader developers work closely with embedded OS/RTOS designers to understand and build fully optimized boot loader to OS protocols and interfaces.

References

[1] Advanced Configuration and Power Interface Speciation; Revision 3.0b October 10, 2006; Hewlett-Packard,* Intel, Microsoft,* Phoenix,* Toshiba*

Author Biography

John C. MacInnis: John C. MacInnis, Embedded and Communications Group, Intel Corporation, Technical Marketing, has held engineering, management, product marketing and technical marketing positions managing Intel® architecture firmware and BIOS for over 15 years. He holds an MBA from the University of Phoenix and a BSECE from the University of Michigan.

Copyright

A REAL-TIME HPC APPROACH FOR OPTIMIZING INTEL MULTI-CORE ARCHITECTURES

Contributors

Dr. Aljosa Vrancic
National Instruments

Jeff Meisel
National Instruments

Index Words

Multi-Core
Symmetric Multiprocessing (SMP)
High-Performance Computing (HPC)
Nehalem
Cache Optimization
Real-Time Operating System (RTOS)
LabVIEW

Abstract

Complex math is at the heart of many of the biggest technical challenges facing today's engineers. With embedded multi-core processors, the type of calculations that would have traditionally required a supercomputer can now be performed at lower power in a real-time, embedded environment. This article presents findings that demonstrate how a novel approach with Intel hardware and software technology is allowing for real-time high-performance computing (HPC) in order to solve engineering problems with multi-core processors that were not possible only five years ago. First, we will review real-time concepts that are important for understanding this domain of engineering problems. Then, we will compare the traditional HPC approach with the real-time HPC approach outlined in this article. Next, we will outline software architecture approaches for utilizing multi-core processors along with cache optimizations. Finally, industry examples will be considered that employ this methodology.

Introduction to Real-Time Concepts

Because tasks that require acceleration are so computationally intensive, your typical HPC problem could not traditionally be solved with a normal desktop computer, let alone an embedded system. However, disruptive technologies such as multi-core processors enable more and more HPC applications to now be solved with off-the-shelf hardware.

Where the concept of real-time HPC comes into the picture is with regard to number crunching in a deterministic, low-latency environment. Many HPC applications perform offline simulations thousands and thousands of times and then report the results. This is not a real-time operation because there is no timing constraint specifying how quickly the results must be returned. The results just need to be calculated as fast as possible.

Previously, these applications have been developed using a message passing protocol (such as MPI or MPICH) to divide tasks across the different nodes in the system. A typical distributed computer scenario looks like the one shown in Figure 1, with one head node that acts as a master and distributes processing to the slave nodes in the system.

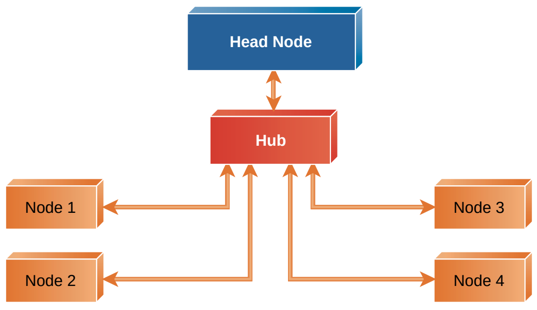


Figure 1: Example configuration in a traditional HPC system .

By default, it is not real-time friendly because of latencies associated with networking technologies (like Ethernet). In addition, the synchronization implied by the message passing protocol is not necessarily predictable with granular timing in the millisecond ranges. Note that such a configuration could potentially be made real-time by replacing the communication layer with a real-time hardware and software layer (such as reflective memory), and by adding manual synchronization to prioritize and ensure completion of tasks in a bounded timeframe. Generally speaking though, the standard HPC approach was not designed for real-time systems and presents serious challenges when real-time control is needed.

"Generally speaking though, the standard HPC approach was not designed for real-time systems and presents serious challenges when real-time control is needed."

An Embedded, Real-Time HPC Approach with Multi-Core Processors

The approach outlined in this article is based on a real-time software stack, as described in Table 1, and off-the-shelf multi-core processors.

Real-Time Software Stack	Description
Development Tool or Programming Language	The development tool or programming language must provide support to target the RTOS of choice, and allow for threading correctness and optimization. Debugging and tracing capabilities are provided to analyze real-time multi-core systems.
Libraries	Libraries are thread-safe, and by making them reentrant, may be executed in parallel. Algorithms will not induce jitter and avoid dynamic memory allocation or account for it in some way.
Device Drivers	Drivers are designed for optimal multithreaded performance.
RTOS	The RTOS supports multithreading and multitasking, and it can load-balance tasks on multi-core processors with SMP.

Table 1: Real-Time Software Stack.

Real-time applications have algorithms that need to be accelerated but often involve the control of real-world physical systems—so the traditional HPC approach is not applicable. In a real-time scenario, the result of an operation must be returned in a predictable amount of time. The challenge is that until recently, it has been very hard to solve an HPC problem while at the same time closing a loop under 1 millisecond.

Furthermore, a more embedded approach may need to be implemented, where physical size and power constraints place limitations on the design of the system.

Now consider a multi-core architecture, where today you can find up to 16 processing cores.

From a latency perspective, instead of communicating over Ethernet, with a multi-core architecture that can be found in off-the-hardware there is inter-core communication that is determined by system bus speeds. So return-trip times are much more bounded. Consider a simplified diagram of a quad-core system shown in Figure 2.

In addition, multi-core processors can utilize symmetric multiprocessing (SMP) operating systems—a technology found in general purpose operating systems like Microsoft* Windows,* Linux, and Apple Mac OS* for years to automatically load-balance tasks across available CPU resources. Now real-time operating systems are offering SMP support. This means that a developer can specify timing and prioritize

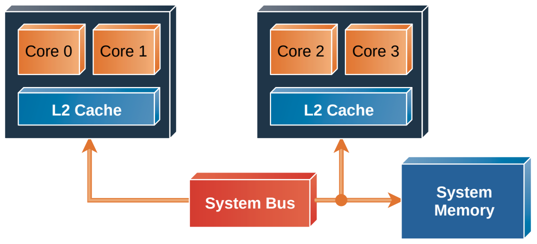


Figure 2: Example configuration in a multicore system. Source: Adapted from Tian and Shih, "Software Techniques for Shared-Cache Multi-Core Systems," Intel Software Network.

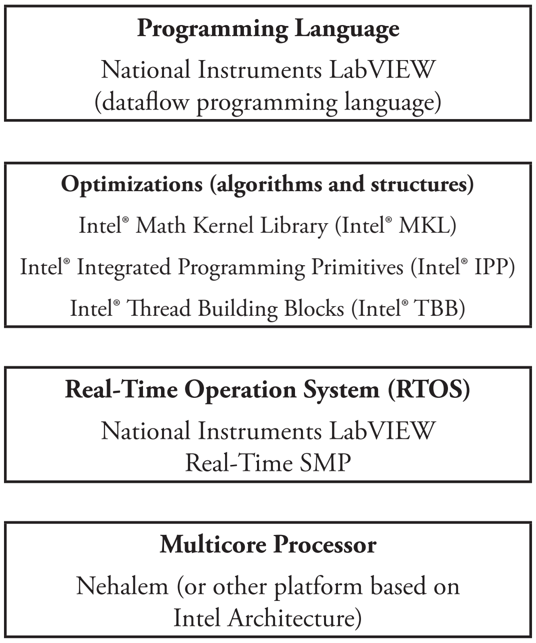

Programming Language
National Instruments LabVIEW
(dataflow programming language)
Optimizations (algorithms and structures)
Intel® Math Kernel Library (Intel® MKL)
Intel® Integrated Programming Primitives (Intel® IPP)
Intel® Thread Building Blocks (Intel® TBB)
Real-Time Operation System (RTOS)
National Instruments LabVIEW
Real-Time SMP
Multicore Processor
Nehalem (or other platform based on
Intel Architecture)

Figure 3: Example Software and Hardware Components in Real-Time HPC System

thread interactions. This is a tremendous simplification compared with message-passing and manual synchronization, and it can all be done in real-time.

Real-Time HPC System Description

For the approaches outlined in this article, Figure 3 represents the general software and hardware approach that has been applied.

Note: The optimizations layer is included as part of the LabVIEW language; however, it deserves mentioning as a separate component.

Multi-core Programming Patterns for Real-Time Math

From a software architecture perspective, the developer should look to incorporate a parallel pattern that best suites the real-time HPC problem. Before choosing the appropriate pattern, both application characteristics and hardware architecture should be considered. For example, is the application computation or I/O bound? How will cache structure, overall memory hierarchy, and system bus speeds affect the ability to meet real-time requirements? Because of the wide range of scenarios that are dependent on the specific application, the LabVIEW language includes hardware-specific optimizations, timing, and querying capabilities to help the developer utilize the multi-core architecture in the most efficient manner possible. (From a historical perspective, LabVIEW originated as a programming tool for test and measurement applications, and therefore it was very important to include timing and synchronization capabilities in the form of native constructs in the language.)

"For many control engineers, 1 millisecond (ms) is viewed as the longest acceptable "round trip time", so any software component that induces > 1 ms of jitter would make the system unfeasible."

As we will observe, these patterns map well to real-world applications that contain characteristics that are common for real-time HPC applications: (a) the patterns execute code in a loop that may be continuous, and (b) the patterns are communicating with I/O. By I/O, in this sense, we are talking about analog to digital conversion or digital to analog conversion that would be used to control real-world phenomena or control system. (For many control engineers, 1 millisecond (ms) is viewed as the longest acceptable "round trip time", so any software component that induces > 1 ms of jitter would make the system unfeasible.)

Entire books are dedicated to programming patterns, and for completeness we will at a high-level consider three such patterns that can be applied to a real-time HPC application (without delving into the intricacies):

- Pipelining
- Data parallelism
- N-dimensional grid

Pipelining

Pipelining is known as the "assembly line" technique, as shown in Figure 4. This approach should be considered in streaming applications and anytime a CPU-intensive algorithm must be modified in sequence, where each step takes considerable time.

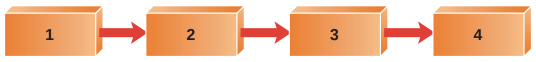


Figure 4: Sequential stages of an algorithm

Like an assembly line, each stage focuses on one unit of work, with the result passed to the next stage until the end of the line is reached.

By applying a pipelining strategy to an application that will be executed on a multi-core CPU, you can break the algorithm into steps that have roughly the same unit of work and run each step on a separate core. The algorithm may be repeated on multiple sets of data or in data that is streaming continuously, as shown in Figure 5.

The key here is to break your algorithm up into steps that take equal time as each iteration is gated by the longest individual step in the overall process. Caveats to this technique arise when data falls out of cache, or when the penalty for inter-core communication exceeds the gain in performance.

A code example in LabVIEW is demonstrated in Figure 6. A loop structure is denoted by a black border with stages S1, S2, S3, and S4 representing the functions in the algorithm that must execute in sequence. Since LabVIEW is a structure dataflow language, the output of each function passes along the wire to the input of the next. A special feedback node, which appears as an arrow with a small dot underneath, is used to denote a separation of the functions into separate pipeline stages. A non-pipelined version of the same code would look very similar, without the feedback nodes. Real-time HPC examples that commonly employ this technique include streaming applications where fast Fourier transforms (FFTs) require manipulation one step at a time.

Data Parallelism

Data Parallelism is a technique that can be applied to large datasets by splitting up a large array or matrix into subsets, performing the operation, and then combining the result.

First consider the sequential implementation, whereby a single CPU would attempt to crunch the entire dataset by itself, as illustrated in Figure 7.

Instead, consider the example of the same dataset in Figure 8, but now split into four parts. This can be spread across the available cores to achieve a significant speed-up.

Now let's examine how this technique can be applied, practically speaking. In real-time HPC, a very common, efficient, and successful strategy in applications such as control systems is the parallel execution of matrix-vector multiplications of considerable size. The matrix is typically fixed, and it can be decomposed in advance. The vector is provided on a per-loop basis as the result of measurements gathered by sensors. The result of the matrix-vector could be used to control actuators, for example.

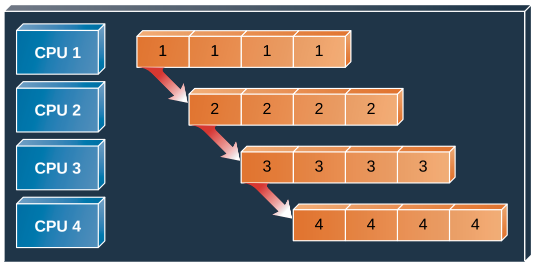


Figure 5: Pipelined approach.

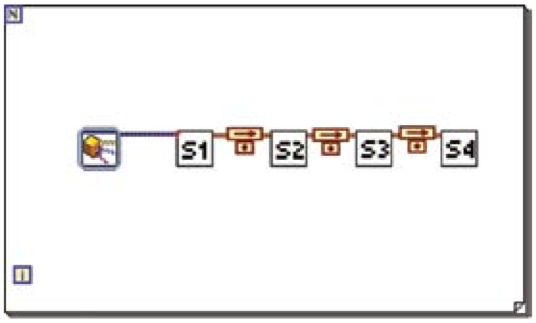


Figure 6: Pipelined approach in LabVIEW.

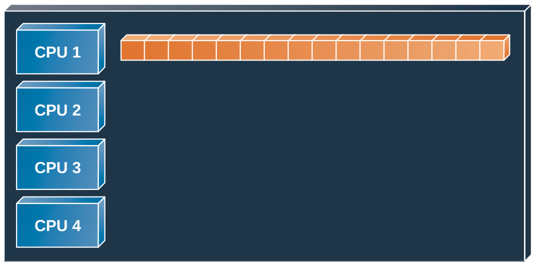


Figure 7: Operation over a large dataset utilizing one CPU.

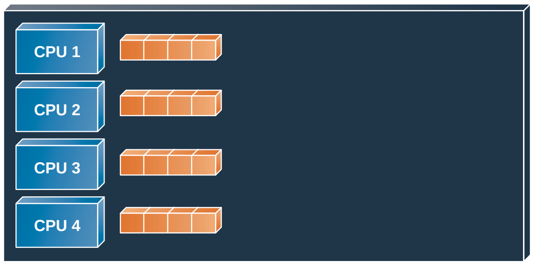


Figure 8: Data Parallelism technique.

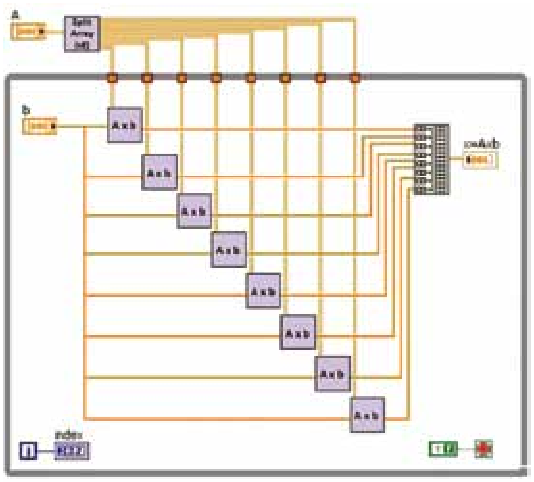


Figure 9: Matrix-vector multiplication using data parallelism technique.

A matrix-vector multiplication distributed to 8 cores is shown in Figure 9 (execution is performed from left to right). The matrix is split before it enters the while-loop. Each block is multiplied by the vector and the resulting vectors are combined to form the final result.

Structured Grid

The structured grid pattern is at the heart of many computations involving physical models, as illustrated in Figure 10. A 2D (or *N*D) grid is calculated every iteration and each updated grid value is a function of its neighbors. The parallel version would split the grid into sub-grids where each sub-grid is computed independently. Communication between workers is only the width of the neighborhood. Parallel efficiency is a function of the area to perimeter ratio.

For example, in the LabVIEW diagram in Figure 11, one can solve the heat equation, where the boundary conditions are constantly changing. The 16 visible icons represent tasks that can solve the Laplace equation of a certain grid size; these 16 tasks map onto 16 cores (the Laplace equation is a way to solve the heat equation). Once per iteration of the loop, boundary conditions are exchanged between those cores and a global solution is built up. The arrows represent data exchange between elements. Such a diagram can also be mapped onto a 1-, 2-, 4-, or 8-core computer. A very similar strategy could also be used for machines with greater numbers of cores as they become available.

A key element to any design pattern is how to map to the underlying memory hierarchy. The next section reviews important cache considerations that should be followed for optimal CPU performance in real-time HPC applications.

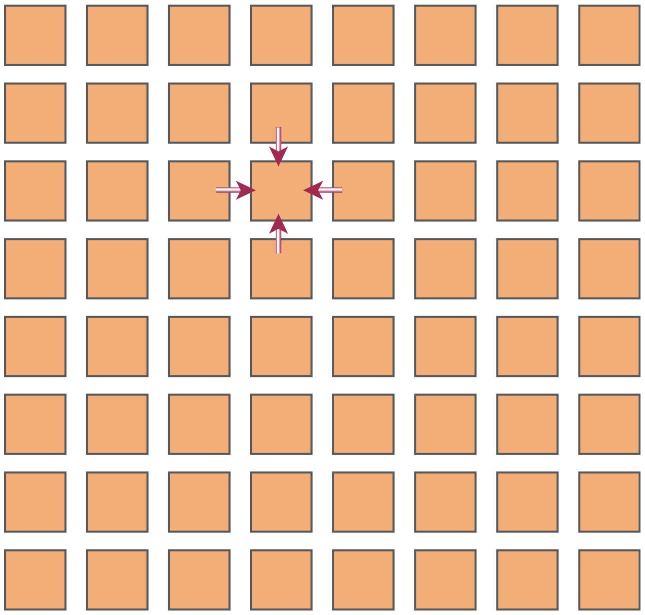


Figure 10: Structured grid approach.

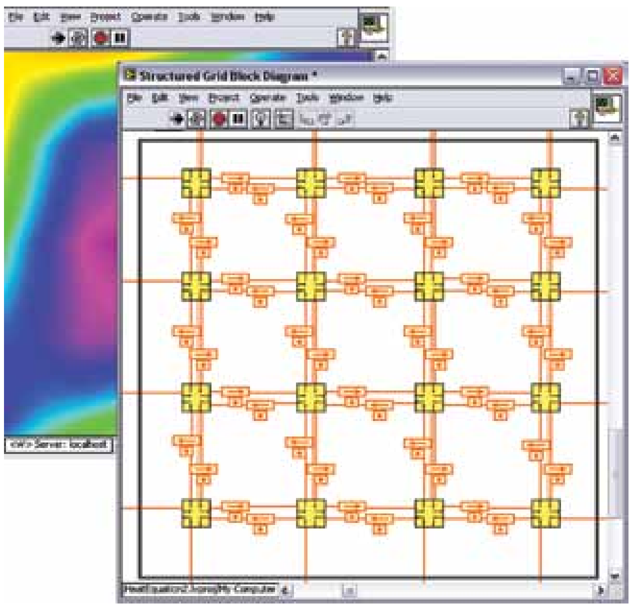


Figure 11: Laplace Equation implemented with Structured grid approach.

Cache Considerations

In traditional embedded systems, CPU caches are viewed as a necessary evil. The evil side shows up as a nondeterministic execution time inversely proportional to the amount of code and/or data of a time-critical task located inside the cache when the task execution has been triggered. For demonstration purposes, we will profile cache performance to better understand some important characteristics. The technique applied is using a structure within LabVIEW called a *timed loop*, shown in Figure 12.

The timed loop acts as a regular while loop, but with some special characteristics that lend themselves to profiling hardware. For example, the structure will execute any code within the loop in a single thread. The timed loop can be configured with microsecond granularity, and it can be assigned a relative priority that will be handled by the RTOS. In addition, it can set processor affinity, and it can also react to hardware interrupts. Although the programming patterns shown in the previous section do not utilize the timed loop, it is also quite useful for dealing with real-time HPC applications, and parallelism is harvested through the use of multiple structures and queue structures to pass data between the structures. The following describes benchmarks that were performed to understand cache performance.

An execution time of a single timed loop iteration as a function of the amount of cached code/data is shown in Figure 13. The loop runs every 10 milliseconds, and we use an indirect way to cause the loop's code/data to be flushed from the cache; a lower priority task that runs after each iteration of the loop adds 1 to each element of an increasingly larger array of doubles flushing more and more of time critical task's data from the CPU cache. In addition to longer runtime, in the worst-case scenario the time goes from 4 to 30 microseconds for an increase by a factor of 7.5. Figure 13 also shows that decaching also increases jitter. The same graph can be also used to demonstrate the "necessary" part of the picture. Even though some embedded CPUs will go as far as completely eliminating cache to increase determinism, it is obvious that such measures will also significantly reduce performance. Besides, few people are willing to go back one or two CPU generations in performance especially as the amounts of L1/L2/L3 cache are continuously increasing providing enough room for most applications to run while incurring only minor cache penalties.

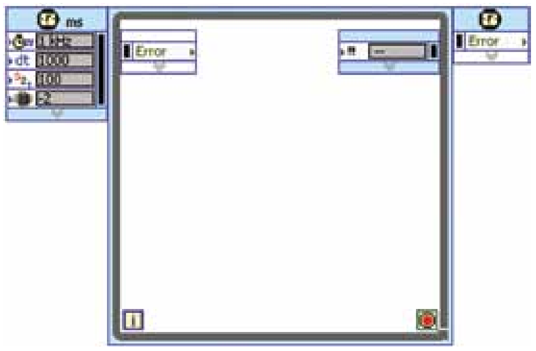


Figure 12: Timed loop structure (used for benchmark use-cases).

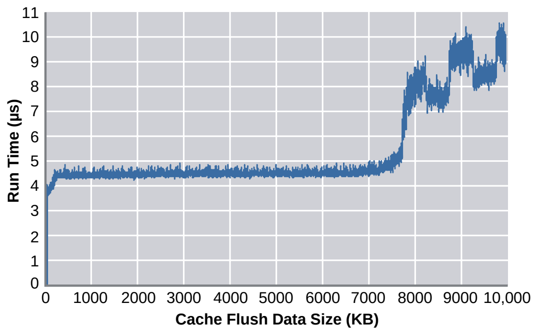


Figure 13: Execution time of a simple time-critical task as a function of amount of cached code/data on 3.2 GHz/8-MB L3 cache i7 Intel CPU using LabVIEW Real Time. Initial ramp-up due to 256K L2 cache.

"In real-time HPC, the cache is extremely important because of its ability to keep the CPU's computational resources busy."

In real-time HPC, the cache is extremely important because of its ability to keep the CPU's computational resources busy. For example, let's assume that we want to add 1 to all elements in a single-precision array on a 3-GHz processor that can execute one single-precision floating point operation every clock cycle—a task of only 3 GFLOPs. The memory bandwidth required to keep the FPU busy would have to be at least 24 GB/s (12 GB/s in each direction), bandwidth above the latest generation of processors with built in memory controllers. The three-channel i7 CPU tops out at 18 GB/s. However, the CPUs can perform more than one FLOP per cycle because they contain multiple FPUs. Using SSE instructions, one can add four single precision floating point numbers every cycle so our array example would require at least 96 GB/s memory bandwidth to prevent stalls. The red curve in Figure 14 contains benchmark results for the described application. Figure 14 shows GFLOPs as a function of array size on 3.2 GHz i7 (3.2 GHz, 8 MB L3) for x[i] = x[i] + 1 (red curve) and x[i] = A*x[i]2 + B*x[i] + c (black curve) operation on each element of a single-precision floating point array using SSE instructions. The second graph zooms into first 512 KB region. Three steps are clearly visible: L1 (32K), L2 (256K) and L3 (8M). Benchmark executed on LabVIEW Real-Time platform thus minimum amount of jitter. When all data fits in L1, one CPU can achieve approximately 8.5 GFLOPs requiring 72 GB/s memory bandwidth. When running out of L2, CPU delivers 4.75 GFLOPs requiring 38 GB/s. Once data does not fit into CPU caches any more, the performance drops to 0.6 GFLOPs completely bounded by 4.8 GB/s memory bandwidth.

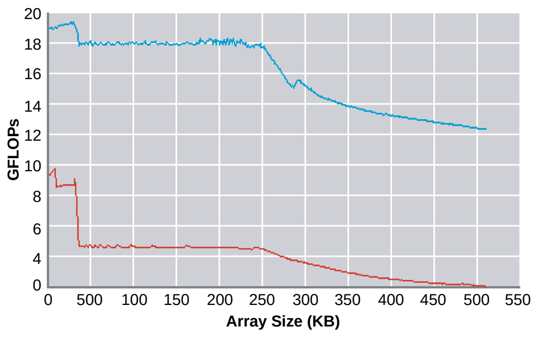


Figure 14: GFLOPs as a function of array size on 3.2 GHz i7.

Zooming into the plot further also shows additional step at the beginning for the red curve, which may point to another level of cache 8K.

The ratio between maximum and minimum performance is a whopping 14x. The situation gets worse on a quad core CPU since the application can easily be parallelized. In the best case, the four CPUs can deliver 36 GFLOPs since the caches are independent and in the worst case the performance stays at 0.6 GFLOPs since the memory bandwidth is shared among the processors. The resulting maximum/minimum performance ratio jumps to 56x. As a verification, we run another test for which more FLOPs are performed for each array element brought into the FPU. Instead of only adding one to the element, we calculate a second order polynomial, which requires four FLOPs compared to one originally. Results are shown in Figure 14 (black curve). AS expected, the maximum performance goes up to 15 GFLOPs since the memory is being accessed less. For the same reason, the performance difference between data completely located in L1 and L2 caches, respectively, drops. As main memory latency and bandwidth becomes a gating factor, we again see large drop-off in the GFLOPs performance, though to a lesser value of "only" 8x.

"Multiple cores with their fast caches can deliver better-than-linear performance increase if the problem that did not originally fit into a single-CPU cache can fit into multiple CPU caches after it has been parallelized."

The above simple example demonstrates that multiple cores with their fast caches can deliver better-than-linear performance increase if the problem that did not originally fit into a single-CPU cache can fit into multiple CPU caches after it has been parallelized. However, Figure 14 also implies that any unnecessary data transfer between the CPUs can seriously degrade performance especially if data has to be moved to/from main memory. Causing cache misses while parallelizing the application can not only eat up all performance improvements resulting from an increased number of CPUs, but it can also cause an application to run tens of times slower than on single CPU.

So, what can one do if real-time HPC application data does not fit into cache? The answer depends on the amounts of data used for the time-critical versus non-time-critical tasks. For example, the data used in a control algorithm must be readily available at all times and should be kept in cache at all cost. The data that is used for user interface display or calculation of real-time parameters can be flushed.

If the data used by the time-critical code fits into the cache, the application must overcome cache strategy that can be simply described as "use it or lose it," by, well, using the data. In other words, accessing data even when it is not required will keep it in the cache. In some cases, the CPU may offer explicit instructions for locking down parts of the cache but that is more exception than a rule. For example, if there is control data that may be used as the plant approaches some critical point, accessing data in each control step is a must. An argument that executing control code each iteration that may otherwise be required to run only once every 1000 iterations is overkill, since even in the worst case the out-of-cache execution may be only 20x slower, is correct, at least when viewed form an HPC point of view. Following the described approach would yield 50x worst CPU utilization and proportionally slower execution. Unfortunately, in the real-time HPC this line of argument is false because a 20x slower execution of control algorithm can result in serious damage—the real-time requirement states that every control action must be taken before a given deadline, which may be much shorter that the worst-case out-of-cache 20x longer execution time.

"If the data used by the time-critical code fits into the cache, the application must overcome cache strategy that can be simply described as 'use it or lose it'."

Another way to keep data in the CPU cache is to prevent any other thread from execution on the given processor. This is where ability of an RTOS to reserve certain CPUs for execution of a single task becomes extremely powerful. One does have to keep in mind that certain caches may be shared between multiple CPUs (for example, L3 cache in Intel's i7 architecture is shared between up to 8 CPUs) residing on the same physical chip so reserving a core on the processor that churns a lot of data on its other cores will be ineffective.

"This is where ability of an RTOS to reserve certain CPUs for execution of a single task becomes extremely powerful."

Finally, what can one do if the real-time data does not fit in the cache? Short of redesigning the underlying algorithm to use less data, or further prioritizing importance of different real-time tasks and devising a scheduling scheme that will keep the most important data in cache, there is not much that can be done. The penalty can be reduced if one can design an algorithm that can access data in two directions. If the data is always accessed in the same order, once it does not fit into the cache any more, each single element access will result in the cache miss. On the other hand, if the algorithm alternates between accessing data from first-to-last and last-to-first element, cache misses will be limited only to the amount of data actually not fitting into the cache: the data accessed last in the previous step is now accessed first and is thus already located in the cache. While this approach will always reduce algorithm execution time in the absolute terms, the relative performance benefit will decrease as more and more data does not fit into the cache.

"While this approach will always reduce algorithm execution time in the absolute terms, the relative performance benefit will decrease as more and more data does not fit into the cache."

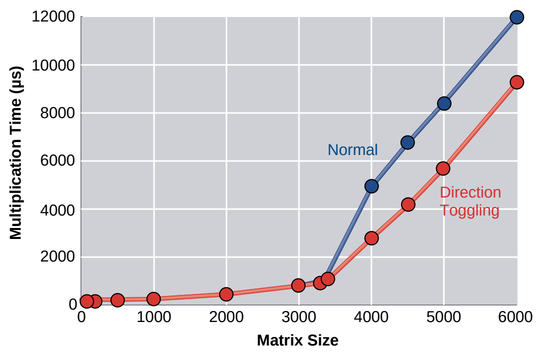


Figure 15: Matrix vector multiplication time (worst case) as a function of matrix size.

Figure 15 and 16 show benchmarks from a real-world application to which we applied all cache management methods described above. The figures show time required to multiply a symmetric matrix with a vector. The multiplication is part of a control algorithm that has to calculate 3000 actuator control values based on 6000 sensor input values in less than 1 ms. (This industry example is from the European Southern Observatory and is mentioned in the final section of the paper).

Initially, we tried to use standard libraries for matrix vector multiplication but we could not achieve desired performance. The algorithms were top notch but they were not optimized for real-time HPC. So, we developed a new vector-matrix multiplication algorithm that took advantage of the following:

- In the control applications, the interaction matrix whose inverse is used for calculation of actuator values does not change often. Consequently, expensive offline preprocessing and repackaging of the matrix into a form that takes advantage of L1/L2 structure of the CPU as well as exercises SSE vector units to their fullest when performing online real-time calculation is possible.
- By dedicating CPUs to control a task only, the control matrix stays in the cache and offers the highest level of performance.
- Splitting vector matrix multiplication into parallel tasks increases the amount of cache available for the problem.
- The new algorithm can access matrix data from both ends.

Figure 15 shows a Matrix vector multiplication time (worst case) as a function of matrix size. Platform: Dual Quad-core 2.6-GHz Intel® Xeon processors, with a 12-MB cache each. The results were achieved by using only 4 cores, 2 on each processor. Utilizing all 8 cores resulted in further multiplication time for 3k x 3k matrix of 500 us.

Figure 16 depicts the Nehalem (8M L3) – cache boundary approximately 1900. The curve is similar to that of curve for the Intel® Xeon® processor. The difference is due to direction toggling smaller because of a much larger increase in memory bandwidth compared to the increase in computation power. Excellent memory performance is visible for the 6K x 6K case: for 20% CPU clock increase, there is a 160% increase in performance (210% for non-toggling approach).

The results show that we are able to achieve 0.7 ms multiplication time for the required 3k x 3k matrix. The 1-millisecond limit is reached for the matrix size of about 3.3k x 3.3k, which is also close to the total L2 cache size (2 processors x 12 MB = 24 MB L2). Increasing the matrix size 4 times (6k x 6k) speeds execution time 17 times, implying a 4x degradation in GFLOPs. Using direction toggling approach results in up to 50 percent relative performance improvements for data sizes slightly larger than the available L2. The speed-up reduces in relative terms as the matrix is getting larger.

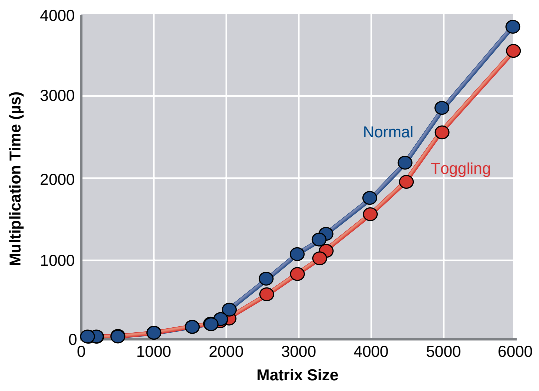


Figure 16: Nehalem (8M L3) – cache boundary approximately 1900.

Industry Examples

The following industry examples show how real-time HPC is being applied today, in many cases where only five years ago the computational results would not be achievable. All of these examples were developed with the LabVIEW programming language to take advantage of multi-core technology.

Structural Health Monitoring on China's Donghai Bridge

The Donghai Bridge, shown in Figure 17, is China's first sea-crossing bridge, stretching across the East China Sea and connecting Shanghai to Yangshan Island. The bridge has a full length of 32.50 km, a 25.32-km portion of which is above water.

Figure 17: Donghai Bridge.
Source: Wikipedia Commons

Obviously, the monitoring system for Donghai Bridge is of a large scale with a variety of quantities to be monitored and transmitted.

Modal analysis methods can be used to reflect the dynamic properties of the bridge. In fact, modal analysis is a standard engineering practice in today's structural health monitoring (SHM).

To cope with the modal analysis on large structures like bridges, however, a relatively new type of modal analysis method has been developed, which works with the data gathered at the same time the structure being analyzed is working. This is operational modal analysis. In this method, no explicit stimulus signal is applied to the structure; rather, the natural forces from the environment and the work load applied to the structure serve as the stimuli, which are random and unknown. Only the signals measured by the sensors put on the structure can be obtained and used, which serve as the response signals.

"No explicit stimulus signal is applied to the structure; rather, the natural forces from the environment and the work load applied to the structure serve as the stimuli, which are random and unknown."

Within the operational modal analysis domain, there is a type of method that employs output-only system identification (or in other terms, time series analysis) techniques, namely, stochastic subspace identification (SSI).

In order to monitor a bridge's health status better, some informative quantities are needed to be tracked in real-time. In particular, it is highly desirable that the resonance frequencies are monitored in real-time. The challenge now is to do resonance frequency calculation online, which is a topic of current research for a wide range of applications.

To enable SSI methods to be working online, SSI needs to be reformulated to some sort of recursive fashion so as to reach the necessary computational efficiency. This is recursive stochastic subspace identification (RSSI). With RSSI, the multichannel sampled data are read and possibly decimated. The decimated data then are fed to the RSSI algorithm. Each time a new decimated data sample is fed in, a new set of resonance frequencies of the system under investigation are produced. That is, the resonance frequencies are updated as the data acquisition process goes on. If the RSSI algorithm is fast enough, this updating procedure is running in real-time.

"If the RSSI algorithm is fast enough, this updating procedure is running in real-time."

Although further experiments need to be performed to validate the RSSI method, the results so far have shown feasibility and effectiveness of this method under the real-time requirement. With this method, the important resonance frequencies of the bridge can be tracked in real-time, which is necessary for better bridge health monitoring solutions.

"TORC Technologies and Virginia Tech used LabVIEW to implement parallel processing while developing vision intelligence in its autonomous vehicle for the 2007 DARPA Urban Challenge."

Vision Perception for Autonomous Vehicles

In an autonomous vehicle application, TORC Technologies and Virginia Tech used LabVIEW to implement parallel processing while developing vision intelligence in its autonomous vehicle for the 2007 DARPA Urban Challenge. LabVIEW runs on two quad-core servers and performs the primary perception in the vehicle.

This type of application is a clear example of where high-computation must be obtained in an embedded form factor, in order not only to meet the demands of the application but also to fit within low power constraints.

Nuclear Fusion Research

At the Max Planck Institute for Plasma Physics in Garching, Germany, researchers implemented a tokamak control system to more effectively confine plasma.

For the primary processing, they developed a LabVIEW application that split up matrix multiplication operations using a data parallelism technique on an octal-core system.

"Dr. Louis Giannone, the lead researcher on the project, was able to speed up the matrix multiplication operations by a factor of five while meeting the 1-millesecond real-time control loop rate."

Dr. Louis Giannone, the lead researcher on the project, was able to speed up the matrix multiplication operations by a factor of five while meeting the 1-millesecond real-time control loop rate.

Real-Time Control of the World's Largest Telescope

The European Southern Observatory (ESO) is an astronomical research organization supported by 13 European countries, and has expertise developing and deploying some of the world's most advanced telescopes. The organization is currently working on a USD 1 billion 66-antenna submillimeter telescope scheduled for completion at the Llano de Chajnantor in 2012.

One current project on their design board is the Extremely Large Telescope. The design for this 42 m primary mirror diameter telescope is in phase B and received USD 100 million in funding for preliminary design and prototyping. After phase B, construction is expected to start in late 2010.

The system, controlled by LabVIEW software, must read the sensors to determine the mirror segment locations and, if the segments move, use the actuators to realign them. LabVIEW computes a 3,000 by 6,000 matrix by 6,000 vector product and must complete this computation 500 to 1,000 times per second to produce effective mirror adjustments.

"One current project on their design board is the Extremely Large Telescope. The design for this 42 m primary mirror diameter telescope is in phase B."

Sensors and actuators also control the M4 adaptive mirror. However, M4 is a thin deformable mirror—2.5 m in diameter and spread over 8,000 actuators. This problem is similar to the M1 active control, but instead of retaining the shape, we must adapt the shape based on measured wave front image data. The wave front data maps to a 14,000 value vector, and we must update the 8,000 actuators every few milliseconds, creating a matrix-vector multiply of an 8 by 14 k control matrix by a 14 k vector. Rounding up the computational challenge to 9 by 15 k, this requires about 15 times the large segmented M1 control computation.

Jason Spyromillo from the European Southern Observatory, describe the challenge as follows: "Our approach is to simulate the layout and design the control matrix and control loop. At the heart of all these operations is a very large LabVIEW matrix-vector function that executes the bulk of the computation. M1 and M4 control requires enormous computational ability, which we approached with multiple multi-core systems. Because M4 control represents 15 3 by 3 k submatrix problems, we require 15 machines that must contain as many cores as possible. Therefore, the control system must command multi-core processing."

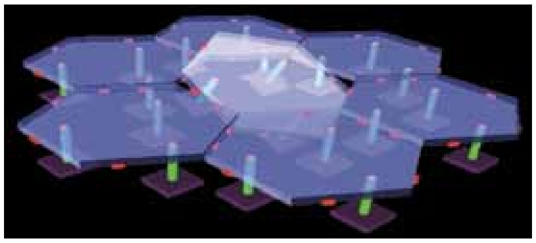


Figure 18: Example Section of M1 Mirror, simulated in LabVIEW.

"Smart Car" Simulation for Adaptive Cruise Control and Lane Departure Systems

Over the last 15 years, passive safety technologies such as ABS, electronic stability control, and front/side airbags have become ubiquitous features on a wide range of passenger vehicles and trucks.

The adoption of these technologies has greatly accelerated the use of simulation software into vehicle engineering. Using a combination of CarSim (Mechanical Simulation's internationally validated, high-fidelity simulation software) and Lab-VIEW, engineers routinely design, test, optimize, and verify new controller features months before a physical vehicle is available for the test track.

Now that vehicles are monitoring their environment with several vision and radar sensors and actually communicating with other cars on the road, it is essential that every vehicle in the test plan has a highly accurate performance model because each car and truck will be automatically controlled near physical limitations.

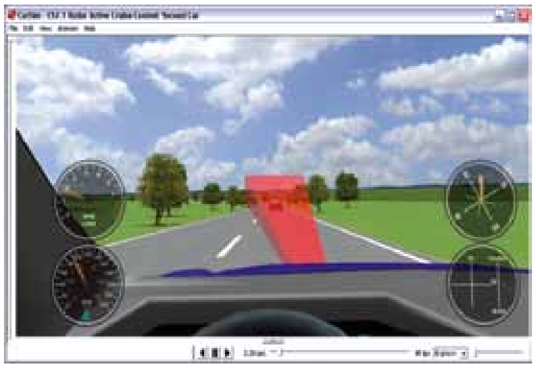


Figure 19: Simulation of Adaptive Cruise Control using CarSim.*

To address these requirements, CarSim has been integrated with National Instruments multi-core real-time processors and LabVIEW RT to allow vehicle designers to run as many as sixteen high fidelity vehicles on the same multi-core platform. This extraordinary power allows an engineer to design a complex, coordinated traffic scenario involving over a dozen cars with confidence that each vehicle in the test will behave accurately. This type of a test would be impossible at a proving ground.

Advanced Cancer Research Using Next Generation Medical Imaging Techniques

Optical coherence tomography (OCT) is a noninvasive imaging technique that provides subsurface, cross-sectional images of translucent or opaque materials. OCT images enable us to visualize tissues or other objects with resolution similar to that of some microscopes. There has been an increasing interest in OCT because it provides much greater resolution than other imaging techniques such as magnetic resonance imaging (MRI) or positron emission tomography (PET). Additionally, the method is extremely safe for the patients.

"There has been an increasing interest in OCT because it provides much greater resolution than other imaging techniques."

To address this challenge, Dr. Kohji Ohbayashi from Kitasato University led a team of researchers to design a systembased on LabVIEW and multi-core technology. The hardware design utilized a patented light-source technology along with a high-speed

"The end goal of this research is to help detect cancer sooner in patients and increase their quality of life."

(60 MS/s) data acquisition system with 32 NI PXI-5105 digitizers to provide 256 simultaneously sampled channels.

The team at Kitasato University was able to create the fastest OCT system in the world, achieving a 60 MHz axial scan rate.

From a pure number crunching perspective, 700,000 FFTs were calculated per second. The end goal of this research is to help detect cancer sooner in patients and increase their quality of life.

Conclusion

This article presented findings that demonstrate how a novel approach with Intel hardware and software technology is allowing for real-time HPC in order to solve engineering problems with multi-core processing that were not possible only five years ago. This approach is being deployed in widely varying applications, including the following: structural health-monitoring, vehicle perception for autonomous vehicles, tokamak control, "smart car" simulations, control and simulation for the world's large telescope, and advanced cancer research through optical coherence tomography (OCT).

Acknowledgements

The authors would like to acknowledge the following for their contributions to this article: Rachel Garcia Granillo, Dr. Jin Hu, Bryan Marker, Rob Dye, Dr. Lothar Wenzel, Mike Cerna, Jason Spyromilio, Dr. Ohbayashi, Dr. Giannone, and Michael Fleming.

References

Akhter and Roberts. Multi-Core Programming. 2006 Intel Press.

Bridge Health Monitoring System. Shanghai Just One Technology. *http://zone.ni.com/devzone/cda/tut/p/id/6624*

Cleary and Hobbs, California Institute of Technology. "A Comparison of LAM-MPI and MPICH Messaging Calls with Cluster Computing." *http://www.jyi.org/research/re.php?id=752*

Domeika, Max. Software Development for Embedded Multi-core Systems: A Practical Guide Using Embedded Intel® Architecture. Newnes 2008.

Eadline, Douglas. "Polls, Trends, and the Multi-core Effect." September 18th, 2007 *http://www.linux-mag.com/id/4127*

Giannone, Dr. Louis. Real-Time Plasma Diagnostics. *ftp://ftp.ni.com/pub/branches/italy/rnd_table_physics/rnd_table_physics08_max_plank.pdf*

Meisel and Weltzin. "Programming Strategies for Multicore Processing: Pipelining." *www.techonline.com/electronics_directory/techpaper/207600982¨*

Ohbayashi, Dr. Kohji. Advanced Cancer Research Using Next Generation Medical Imaging. *http://sine.ni.com/cs/app/doc/p/id/cs-11321*

Spyromilio, Jason. Developing Real-Time Control for the World's Largest Telescope. *http://sine.ni.com/cs/app/doc/p/id/cs-11465*

Tian and Shih. "Software Techniques for Shared-Cache Multi-Core Systems." Intel Software Network. July 9th, 2007. *http://softwarecommunity.intel.com/articles/eng/2760.htm*

Author Biographies

Dr. Aljosa Vrancic: Dr. Aljosa Vrancic is a principal engineer at National Instruments. He holds a B.S. in electrical engineering from the University of Zagreb, and an M.S. degree and PhD in Physics from Louisiana State University. He is a leading technical authority in the areas of deterministic protocols, real-time SMP operating systems, and software optimization for large scale computation.

Jeff Meisel: Jeff Meisel is the LabVIEW product manager at National Instruments and holds a B.S. in computer engineering from Kansas State University. He represents National Instruments as a member of the Multi-core Association and has published over 20 articles on the topic of multi-core programming techniques. He has presented at industry conferences such as Embedded World Germany, Embedded Systems Conference, and the Real-Time Computing Conference.

Copyright

DIGITAL SIGNAL PROCESSING ON INTEL® ARCHITECTURE

Contributors

David Martinez-Nieto
Intel Corporation

Martin McDonnell
Intel Corporation

Peter Carlston
Intel Corporation

Ken Reynolds
Intel Corporation

Vasco Santos
Intel Corporation

Index Words

Embedded DSP
Intel® architecture
Vector Processing
Parallel Processing
Optimization

Abstract

The suitability of Intel® multi-core processors for embedded digital signal processing (DSP) applications is now being reevaluated. Major advances in power-efficient transistor technology, optimized multi-core processor microarchitectures and the evolution of Intel® Streaming SIMD Extensions (Intel® SSE) for vector processing have combined to produce favorable GFLOPS/watt and GFLOPS/size ratios. In addition, other factors such as code portability across the entire range of Intel® processors and a large set of Intel and third-party software development tools and performance libraries often mean that software development and support costs can be substantially reduced.

This article explores the main differences between traditional digital signal processors and modern Intel general purpose processor architectures and gives some orientation on how DSP engineers can most effectively take advantage of the resources available in Intel processors. We then show how these techniques were used to implement and benchmark performance of medical ultrasound, wireless infrastructure, and advanced radar processing algorithms on a variety of current Intel processors.

Introduction

Digital signal and image processing (DSP) is ubiquitous: From digital cameras to cell phones, HDTV to DVDs, satellite radio to medical imaging. The modern world is increasingly dependent on DSP algorithms. Although, traditionally, special-purpose silicon devices such as digital signal processors, ASICs, or FPGAs are used for data manipulation, general purpose processors (GPPs) can now also be used for DSP workloads. Code is generally easier and more cost-effective to develop and support on GPPs than on large DSPs or FPGAs. GPPs are also able to combine general purpose processing with digital signal processing in the same chip, a major advantage for many complex algorithms. Intel's processor microarchitecture, instruction set, and performance libraries have features and benefits that can be exploited to deliver the performance and capability required by DSP applications.

"The modern world is increasingly dependent on DSP algorithms."

We first consider the main techniques that should be considered when programming DSP algorithms on Intel® architecture, and then illustrate their use in medical ultrasound, wireless infrastructure, and advanced radar post-processing algorithms.

Clock Speed and Cache Size

DSP performance on a GPP is closely related to the clock speed of the processor, and, depending on the workload, the size of its on-chip memory caches. NA Software Ltd* (NASL) recently compared the performance of their VSIPL* library functions for Intel® Architecture Processors with their VSIPL library for PowerPC* architecture processor.[1] (VSIPL is an industry standard, hardware-agnostic API for DSP and vector math.) Table 1 shows the effect of processor frequency and cache size on the time it takes to complete a complex vector multiply operation with vectors of various lengths (N) on a single core of three processors.

"DSP performance on a GPP is closely related to the clock speed of the processor, and, depending on the workload, the size of its on-chip memory caches."

Normalized for clock speed, all processors exhibit roughly the same performance. But the data clearly shows that the speed of the processor is the predominant determinant of performance: the 1.0-GHz Freescale* processor takes longer to complete the complex vector multiply than the 1.88- and 2.18-GHz Intel® processors; the 2.18-GHz processor is always faster than the 1.88-GHz processor, except when N = 128. The clue to this apparent anomaly is L2 cache sizes. The complex vector multiply calculation repeatedly works on the same area of memory—for N = 128, 3 MB of memory are required (128K x sizeof(complex) x 3 vectors). So the N = 128 K calculation requires three-fourths of the Intel® Core™2 Duo processor T7400's cache, resulting in a higher percentage of cache misses: its N = 128 times are 4x its N = 64 K times. The data only requires half of the Intel® Core™2 Duo processor SL9400's 6-MB cache: the N = 128 times are almost precisely 2x its N = 64 K times. With only 1 MB of L2 cache a Freescale MPC 8641D core is at a disadvantage with all N values from 32 K upwards.

Processor Name	**Clock Speed & L2 Cache Size**	**Value of N**						
		256	**1 K**	**4 K**	**16 K**	**32 K**	**64 K**	**128 K**
Freescale* MPC 8641D	1.0 GHz; 1 MB per core	0.78	2.5	18.7	74	*145*	*3,391*	*9,384*
Intel® Core™2 Duo Processor T7400	2.18 GHz; 4 MB shared	0.42	1.8	8.3	33	66	131	*527*
Intel® Core™2 Duo Processor SL9400	1.88 GHz; 6 MB shared	0.44	2.0	8.8	35	75	151	*300*

Table 1: Complex vector multiply v1(n):= v2(n)*v3(n); times in microseconds. Single core. Times in italic indicate that the data requires a significant portion or is too large to fit into the processor's L2 cache. Source: NA Software Ltd

But what about performance per watt? The MPC 8641D has published thermals of around 25 W, the Intel Core 2 Duo processor T7400 around 45 W (including chipset) and the Intel Core 2 Duo processor SL9400 (also including chipset) around 28 W. So the Intel Core 2 Duo processor SL9400 has the highest performance/watt ratio of the three processors when doing these types of calculations.

"Filters and Fast Fourier Transforms (FFTs), for example, can be implemented using simple multiply and accumulate instructions."

Vectorization

Although DSP algorithms tend to be mathematically intensive, they are often fairly simple in concept. Filters and Fast Fourier Transforms (FFTs), for example, can be implemented using simple multiply and accumulate instructions. Modern GPPs use Single Instruction Multiple Data (SIMD) techniques to increase their performance on these types of low-level DSP functions. Current Intel® Core™ processor family and Intel® Xeon® processor have 16 128-bit vector registers that can be configured as groups of 16, 8, 4 or 2 samples depending on the data format and precision required. For single-precision (32-bit) floating point SIMD processing, for example, four floating point (FP) numbers which need to be multiplied by a second value are loaded into vector register 1 with the multiplicand(s) in register 2. Then the multiply operation is executed on all four numbers in a single processor clock cycle. Current Intel® Core™2 processor family and Intel Xeon processor have a 4-wide instruction pipeline with two FP Arithmetic Logical Units, so potentially 8 single-precision FP operations can be done per clock cycle per core. This number will increase to 16 operations per clock when the Intel® Advanced Vector Extensions (Intel® AVX) Instruction Set Architecture debuts in 2010 "Sandy Bridge" generation processors since AVX SIMD registers will be 256 bits wide.[2]

FIR filters are used in a large percentage of DSP algorithms. They can be easily vectorized since there is no dependency between the calculation of the current frame and the output of the previous frame. This makes them perform very well on SIMD processors.

On the other hand, when contiguous input/output interdependence exists (as in recursive filter implementations), efficient vectorization is not always possible. In some cases, however, a careful analysis of the algorithm may still reveal opportunities for vectorized processing as presented in the LTE Turbo Encoder case study.

Parallelization

Intel architecture, as a multi-core architecture, is suited for executing multiple threads in parallel. In terms of DSP programming, there are several approaches for achieving parallelism:

- Pipelined execution: The algorithm is divided in stages and each of these stages is assigned to a different thread.
- Concurrent execution: The input data is partitioned, and each thread processes its own portion of the input data through the whole algorithm. This is only possible if the functionality of the algorithm is not compromised.

"In terms of DSP programming, there are several approaches for achieving parallelism."

Both approaches can also be combined in order to maximize performance and efficient resource utilization.

When evaluating parallelism, the programmer should also consider cache hierarchy. For maximum throughput, each thread should ideally have its input/output data fit within local caches (L1, L2), minimizing cache trashing due to inter-core coherency overheads. On every stage of the algorithm, threads should guarantee that their output data is contiguously stored in blocks with size that is a multiple of the internal cache line width. Inter-thread data dependencies should be minimized and pipelined to reduce algorithm lock-ups. Thread affinity [3] should also be defined carefully.

"For maximum throughput, each thread should ideally have its input/output data fit within local caches (L1, L2), minimizing cache trashing due to inter-core coherency overheads."

Memory Organization

Memory organization on a DSP differs from that found on Intel architecture. On traditional DSP architectures, the developer manually partitions the memory space in order to reduce the number of accesses to external memories. Program, data, temporary buffers and look-up tables all need to be allocated carefully, as accessing the external memory is costly in terms of latencies introduced.

By comparison, Intel architecture is populated with large amounts of cache while DSPs traditionally include dedicated internal memory. On one hand, this overcomes the strict separation of fast/slow memory devices, enabling more "transparent" memory management strategies. On the other hand, all data, look-up tables and program are originally located in "far" memory. Applications may need to warm the cache with the appropriate data at start-up. To maximize platform performance, it is also important to understand the differences between local and shared caches, as well as cache coherency, especially in the context of multi-threaded applications that span multiple processor cores.

"Applications may need to warm the cache with the appropriate data at start-up."

To further reduce the latency penalties due to cache misses, Intel architecture includes an automatic hardware pre-fetcher, details on which can be found in [4]. Output data should ideally be generated sequentially, or at least in a way in which concurrent threads output do not generate cache line invalidations, that is, threads working on the same cache line should be executed in the same core. Accessing memory in a scattered pattern across multiple pages should be avoided whenever possible.

Fixed and Floating Point Performance

Fixed-point implementations have been traditionally used as a result of the lack of availability or lower performance typically associated with floating-point operations. Fixed-point operations though, usually require additional range-checking computation for overflow and saturation which increases the complexity of the implementation, consequently penalizing performance. On Intel architecture, SIMD floating-point code is almost on par as fixed-point (performance-wise, for the same data width), and may even be faster depending on the implementation overheads associated with the latter, so in a number of cases the above tradeoffs are no longer necessary.

"Fixed-point implementations have been traditionally used as a result of the lack of availability or lower performance typically associated with floating-point operations."

Intel® Performance Libraries

The Intel® Performance Libraries provide optimized foundation-level building block functions for accelerating the development of high performance applications. This set of libraries consists of the Intel® Integrated Performance Primitives (Intel® IPP), the Intel® Math Kernel Library (Intel® MKL), and the Intel® Threading Building Blocks (Intel® TBB). These performance libraries and tools are optimized across the full range of Intel processors. Upon their initialization, they automatically detect the type of processor on which they are running and use the sub-function optimizations for that specific processor family. All Intel IPP and Intel MKL are thread-safe, and those functions that benefit from multi-threading are already threaded. More detailed information about these libraries and the type of efficiency, portability, and performance scalability they provide can be found at the Intel® Software Network Web site [9].

"All Intel IPP and Intel MKL are thread-safe, and those functions that benefit from multi-threading are already threaded."

Performance Measurement and Tuning

Quickly identifying and eliminating performance bottlenecks in complex DSP software often requires the aid of specialized tools. Intel® VTune™ Performance Analyzer is an example of a tool than can greatly facilitate tuning a DSP application for maximum performance. Among other advantages, Intel VTune Performance Analyzer provides low-overhead profiling and system wide analysis (OS, drivers, third party libraries). The tool provides both command line and graphical interfaces. Profiling using such tools as Intel VTune Performance Analyzer provides helpful hints in addressing the types of parallelization issues mentioned in Section 0. For example, an increase in L2, L3 cache line invalidations may indicate a loss of efficiency due to the way memory is being addressed.

"Profiling using such tools as Intel VTune Performance Analyzer provides helpful hints in addressing the types of parallelization issues."

Coding for Automatic Vectorization

Efficiently taking advantage of the vector processing units on modern CPUs can be accomplished by assembly-level programming. The instruction set reference and optimization manuals [4] detail the necessary low-level functionality description and performance provided by the underlying processing units. Although low-level programming potentiates a higher performance level, effective code portability, maintainability and development efficiency can only be attained by using higher-level languages.

Automatic vectorization is available on mainstream compilers such as GCC and Intel® C++ Compiler, and consists of a series of methods that identify and implement vectorizable loops according to the version of the SIMD instruction set specified. Although it works transparently to the programmer, increasing the percentage of code amenable to vectorization requires developers to be aware of issues related to, for example, data dependence and memory alignment. [5][6][7][8]

"To work around data dependence, both vectorized and nonvectorized versions of the loop are implemented, and the selection of which version to run is based on the test results."

In cases where it is not possible to resolve data dependence or memory alignment, compilers may automatically add test code constructs prior to the loop. To work around data dependence, both vectorized and nonvectorized versions of the loop are implemented, and the selection of which version to run is based on the test results.

To circumvent memory misalignment issues, the compiler may "peel" a number of iterations off the loop so that at least part of it runs vectorized [6]. Besides obvious increases in program size, this overhead also affects overall loop performance. The use of special *#pragma* directives and other keywords can guide the compiler through its code generation process, avoiding this overhead.

"The use of #pragma directives gives hints to the compiler regarding vectorization."

Code listing 1, Code listing 2, Code listing 3, Code listing 4, and Code listing 5 show different versions of a simple vector multiply-accumulate function, where the use of *#pragma* directives gives hints to the compiler regarding vectorization.

```
void vecmac( float* x, float* a, float* y, int len )
{
  /* The loop below is already vectorizable as-is. */
  int i;
  for( i = 0; i < len; i++ )
    y[i] += x[i] * a[i];
}
```

Code listing 1: Vector multiply-accumulate function.

```
void vecmac_nv( float* x, float* a, float* y, int len )
{
  int i;
  /* Do not vectorize loop */
  #pragma novector
  for( i = 0; i < len; i++ )
    y[i] += x[i] * a[i];
}
```

Code listing 2: Vector multiply-accumulate function, hinting for non-vectorization.

```
void vecmac_al( float* x, float* a, float* y, int len )
{
  int i;
  /* Assume data is aligned in memory. An exception is
               caused if this assumption is not valid. */
  #pragma vector aligned
  for( i = 0; i < len; i++ )
    y[i] += x[i] * a[i];
}
```

Code listing 3: Vector multiply-accumulate function, asserting memory alignment property.

```
void vecmac_iv( float* x, float* a, float* y, int len )
{
  int i;
  /* Discard data dependence assumptions. Results may
     differ if arrays do overlap in memory. */
  #pragma ivdep
  for( i = 0; i < len; i++ )
    y[i] += x[i] * a[i];
}
```

Code listing 4: Vector multiply-accumulate function, discarding assumed data dependences.

```
void vecmac_al_iv( float* x, float* a, float* y, int len )
{
  int i;
  #pragma vector aligned
  #pragma ivdep
  for( i = 0; i < len; i++ )
    y[i] += x[i] * a[i];
}
```

Code listing 5: Vector multiply-accumulate function, asserting memory alignment property and discarding assumed data dependences.

Comparison on both generated assembly (ASM) code size and performance was carried out for the different versions of the *vecmac* function, on an Intel® Core™2 Duo processor platform (2.533 GHz, 6 MB L2 cache) running Linux* 2.6.18 and Intel C++ Compiler 11.0. Table 2 summarizes the results obtained for random input vectors with *len* = 1000. The impact of memory alignment is also included in the performance numbers, which are normalized to the *vecmac_nv* version having nonaligned input data.

Intel® Core™2 Duo Processor (2.533 GHz, 6 MB L2 cache) Linux* 2.6.18, Intel® C++ Compiler 11.0			
Version	Data is aligned in memory?	ASM code size (number of instructions)	Performance ratio (higher is better)
vecmac_nv	No	68	1x (reference)
Vecmac	No	118	2.31x
vecmac_iv	No	84	2.32x
vecmac_nv	Yes	68	1.002x
Vecmac	Yes	118	2.88x
vecmac_iv	Yes	84	2.9x
vecmac_al	Yes	89	3.71x
vecmac_al_iv	Yes	47	3.75x

Table 2: Assembly code size and performance comparison for the various versions of the vector multiply-accumulate function.

The insignificant performance impact in using the *#pragma ivdep* directive is due to the fact that, in this case, there is no overlap (aliasing) between the memory regions occupied by *x[ ]*, *a[ ]* and *y[ ]*. A vectorized version of the loop is always run, even when this hint is not given to the compiler. The only difference is the initial overlap tests performed to these arrays, hence the differences in resulting assembly code size.

The effect of having the arrays aligned in memory is visible in the performance values for the *vecmac* and *vecmac_iv* implementations. Although the loop is still vectorized in both versions, nonaligned memory access introduces performance penalties.

Finally, it is seen that fully vectorized versions of the same loop outperform the nonvectorized code by a factor close to 4x, as initially anticipated for 32-bit floating point processing.

"Fully vectorized versions of the same loop outperform the nonvectorized code by a factor close to 4x."

Programming with Intel® Streaming SIMD Extensions (Intel® SSE) Intrinsics

In most cases, using performance libraries as building blocks and coding for efficient vectorization, together with carefully-designed multi-threaded software architectures, will provide high performance levels. However, in cases where performance libraries cannot be used, or when tuning a specific portion of an algorithm can provide significant performance improvements, lower-level programming can be used. Intrinsic functions provide an intermediate abstraction level between assembly code and higher-level C code. The abstraction level at which the programmer works is low, allowing vector operations, but some details like register allocation are hidden from the developer. Also, the compiler can still perform optimizations over the code that uses intrinsics (in contrast with inline ASM).

"The abstraction level at which the programmer works is low, allowing vector operations, but some details like register allocation are hidden from the developer."

Code listing 6 presents an example of Intel SSE intrinsic programming that calculates the complex reciprocal (conjugate of the number divided by the squared modulo) of a series of 32-bit floating-point input samples. Four input samples are processed at the same time in order to take best advantage of the SIMD arithmetic units.

```
/* load data  */
sseX[0] = _mm_load_ps(&x[i+0]);
sseX[1] = _mm_load_ps(&x[i+2]);
/* Negate Imaginary part */
sseX[0] = _mm_xor_ps(sseX[0], NegFx);
sseX[1] = _mm_xor_ps(sseX[1], NegFx);
/* multiply to calculate real and imaginary squares */
sseM[0] = _mm_mul_ps(sseX[0], sseX[0]);
sseM[1] = _mm_mul_ps(sseX[1], sseX[1]);
/* real and imaginary parts are now squared and placed
 * horizontally. Add them in that direction/
sseI = _mm_hadd_ps(sseM[0], sseM[1]);
/* calculate the four reciprocals */
sseI = _mm_rcp_ps(sseI);
/* reorder to multiply both real and imag on samples */
sseT[0] = _mm_shuffle_ps(sseI,sseI,0x50); // 01 01 00 00
sseT[1] = _mm_shuffle_ps(sseI,sseI,0xFA); // 11 11 10 10
/* multiply by conjugate */
sseY[0] = _mm_mul_ps(sseT[0],sseX[0]);
sseY[1] = _mm_mul_ps(sseT[1],sseX[1]);
/* store */
_mm_store_ps(&y[i+0], sseY[0]);
_mm_store_ps(&y[i+2], sseY[1]);
```

Code listing 6: Complex reciprocal implementation using Intel® Streaming SIMD Extensions (Intel® SSE) intrinsics

```
complex float *x, *y; // pointers to input and output data
float Ix, Qx; // real and imaginary parts
float Rcp; // reciprocal
/* load */
Ix = *(float *) &x[i];
Qx = *(1+(float *) &x[i]);
/* calculate inverse of power */
Rcp = 1.0f/(Ix*Ix +Qx*Qx);
/* assign */
y[i] = Rcp*(Ix – I*Qx);
```

Code listing 7: Complex reciprocal implementation using standard C

The versions presented above where tested [19] over 1000 samples, the Intel SSE intrinsics version showed a performance advantage of 31.4 percent against the standard C implementation.

Case Study: Medical Ultrasound Imaging

Medical ultrasound imaging is a field that demands a significant amount of embedded computational performance, even on lower-end portable devices. Even though the physical configurations, parameters, and functions provided vary widely across the available device ranges, basic functions such as B-mode imaging share the same basic algorithmic pattern: beamforming, envelope extraction, and polar-to-Cartesian coordinate translation.

"Basic functions such as B-mode imaging share the same basic algorithmic pattern: beamforming, envelope extraction, and polar-to-Cartesian coordinate translation."

Figure 1 shows the block diagram of a typical, basic ultrasound imaging implementation. The transducer array comprises a number of ultrasound emitters/receivers that connect to an analog frontend (AFE) which are responsible for conditioning the ultrasound signals. These signals are converted to/from a digital representation by means of a series of ADCs/DACs. The transmit and receive beamformer components delay and weight each of the transducer elements during transmission and reception, dynamically focusing the transducer array in a sequence of directions during each image frame, without the need for mechanical moving parts or complex analog circuitry (at the cost of a significant increase in digital computational requirements). An envelope detector extracts the information carried by the ultrasound signals, which is then stored and prepared for display. Common systems also have the ability to detect and measure the velocity of blood flow, usually carried out by a Doppler processing algorithm. Image compression and storage for post-analysis is also a common feature.

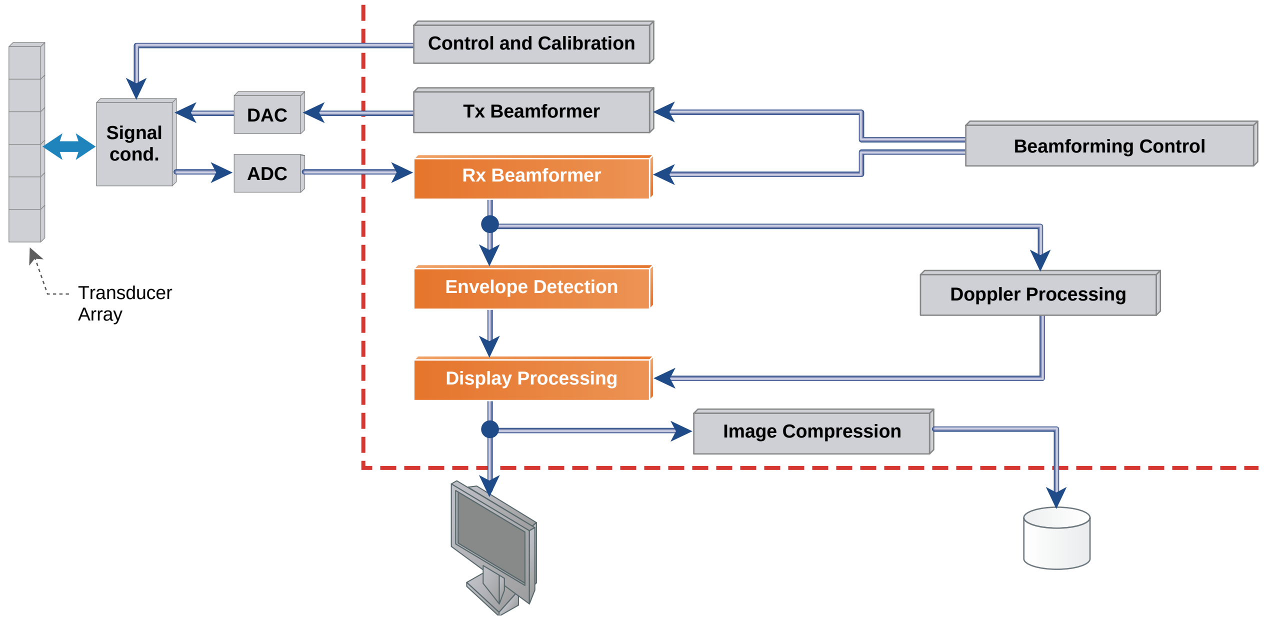


Figure 1: Block diagram of a typical ultrasound imaging application. The dashed line separates the hardware and software components
Source: Intel, 2009

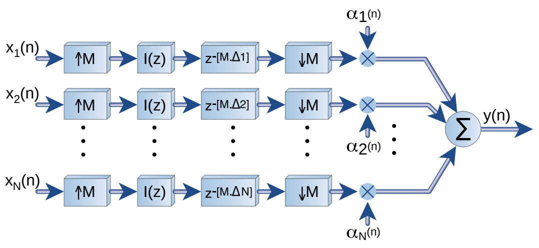


Figure 2: Block diagram of the receive beamformer. Source: Intel, 2009

The highlighted blocks in Figure 1 have been prototyped and measured for performance on the Intel Core 2 Duo and Intel® Atom™ processors, looking at a B-mode imaging application. The Intel IPP was used thoroughly in this prototype. A brief discussion on the architecture, parameters and corresponding estimated performance requirements for each of these blocks follows next. Table 3 lists some of the overall parameter values for this prototype system.

Parameter	Value
Number of transducers	128
Number of scanned lines per frame (steering directions)	128
Angle aperture	90 degrees
Number of samples acquired, per transducer per line	3000
Output image dimensions	640x480 (pixels)
Image resolution	8-bit grayscale
Target number of frames per second	30
Input signal resolution	12-bit fixed point
Output signal resolution	8-bit fixed point
Computational precision (all stages)	32-bit floating point

Table 3: Overall parameters for the ultrasound prototype.

Receive Beamformer

This block implements delay-and-sum synthetic receive focusing with linear interpolation and dynamic apodization. Figure 2 shows the DSP block diagram for this module.

For each scan line that is acquired, each signal stream $x_k(n)$ coming from the transducer elements passes through an upsampler, an interpolation filter *I(z)*, a delay element, a downsampler, and is multiplied by a dynamically varying apodization coefficient. The resulting signals are then accumulated and a single stream *y(n)* is sent to the next processing stage. The delay values are pre-computed [10], multiplied by *M* and rounded to the nearest integer value (operator *[ ]*) and stored in a look-up table (LUT), and are recomputed each time a new line starts to be acquired. The apodization function [11] updates itself for each sampling period of the input streams. All its coefficients are also pre-computed are stored in a LUT.

The interpolation filter is a first-order linear interpolating filter.

If this filter is decomposed into its *M* polyphase components [12], only *N/M* of its taps need to be computed (*N* being the total number of taps). An interpolation/decimation factor of 4 was chosen for this prototype, which means that the filter has a 7-tap, linear-phase FIR configuration.

In terms of number of floating-point DSP operations per second, and assuming that each of the filter instances processes 2 taps per input sample, the structure of Figure 2 would require more than 7.3 GFLOPs for real-time, 30 fps B-mode imaging, a performance level that is difficult to achieve using typical DSP or GPP architectures. Figure 3 shows a rearrangement of the same block diagram, where the 128 parallel filters are transformed into a single filter having the same impulse response. This block diagram is equivalent to the previous one, apart from a loss of accuracy in the delays applied to the apodization coefficients.

Although the channel streams are accumulated at the higher sampling rate, at most the same number of additions is performed since, for each *M* samples of the upsampled signals, only one is not equal to zero. The interpolating filter is now a decimating filter, and an efficient polyphase implementation is also possible.

Assuming the worst-case scenario in which all delay values are the same, the number of operations for the beamforming algorithm is now 3.1 GFLOPs. Still being a high performance target, this represents a reduction of more than 57 percent in computational complexity when compared to the algorithm of Figure 2.

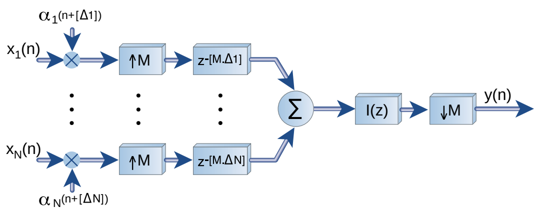


Figure 3: Simplified block diagram of the receive beamformer. Source: Intel, 2009

Envelope Detector

The envelope detector algorithm uses a Hilbert transformer as its central building block. The incoming signals are seen as being modulated in amplitude, where the ultrasound pulses carry (modulate) the information to be displayed. Figure 4 shows its block diagram. The order *L* of the Hilbert transformer is 30.

Assuming that the logarithm consumes 10 DSP operations per sample, the computational requirements for this block would be 437.8 MFLOPs.

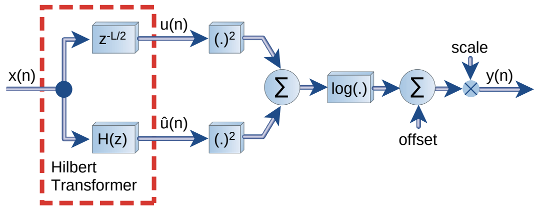


Figure 4: Block diagram of the envelope detector. Source: Intel, 2009

Display Processing

The main responsibility of this block is to convert the array containing all the information to be displayed from polar to Cartesian coordinates. Figure 5 illustrates the transformation performed in this module, in which a hypothetical scanned object (a rectangle) is "de-warped" for proper visual representation.

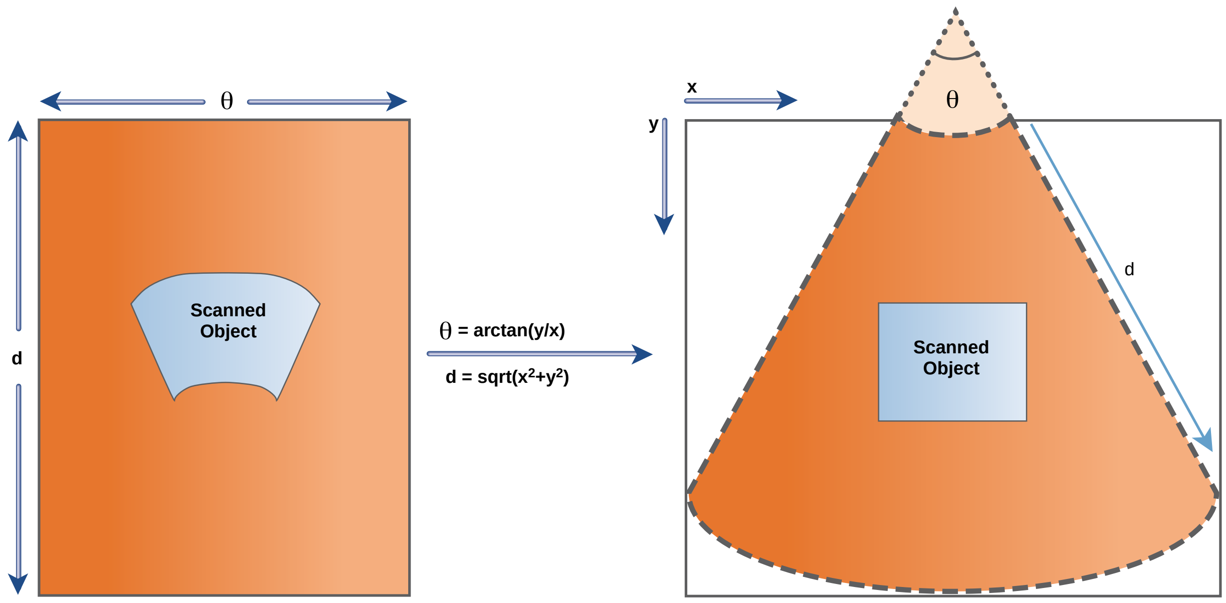


Figure 5: Polar-to-Cartesian conversion of a hypothetically-scanned rectangular object. Source: Intel, 2009

"For a 640x480 pixel image and for a 90 degree angle aperture, the number of active pixels is about 150,000."

In Figure 5, θ represents the steering angle, d is the penetration depth, and x and y are the pixel coordinates in the output image.

During initialization, the application takes the physical parameters of the system and determines the active pixels in the output target frame. Using the conversion formulas in the figure, a LUT is built that stores all the information required for mapping between coordinate spaces. Bilinear interpolation is performed in the (d, θ) space for an increased quality of the output images.

For a 640 x 480 pixel image and for a 90 degree angle aperture, the number of active pixels is about 150,000. To obtain the output pixel amplitude values, the 4 nearest values are read from the polar-coordinate space, and bilinear interpolation is performed using the mapping information computed upon initialization. Figure 6 illustrates this process. For each pixel, 13 DSP operations in total are performed. For a 30 fps system, 58.5 MFLOPs are required.

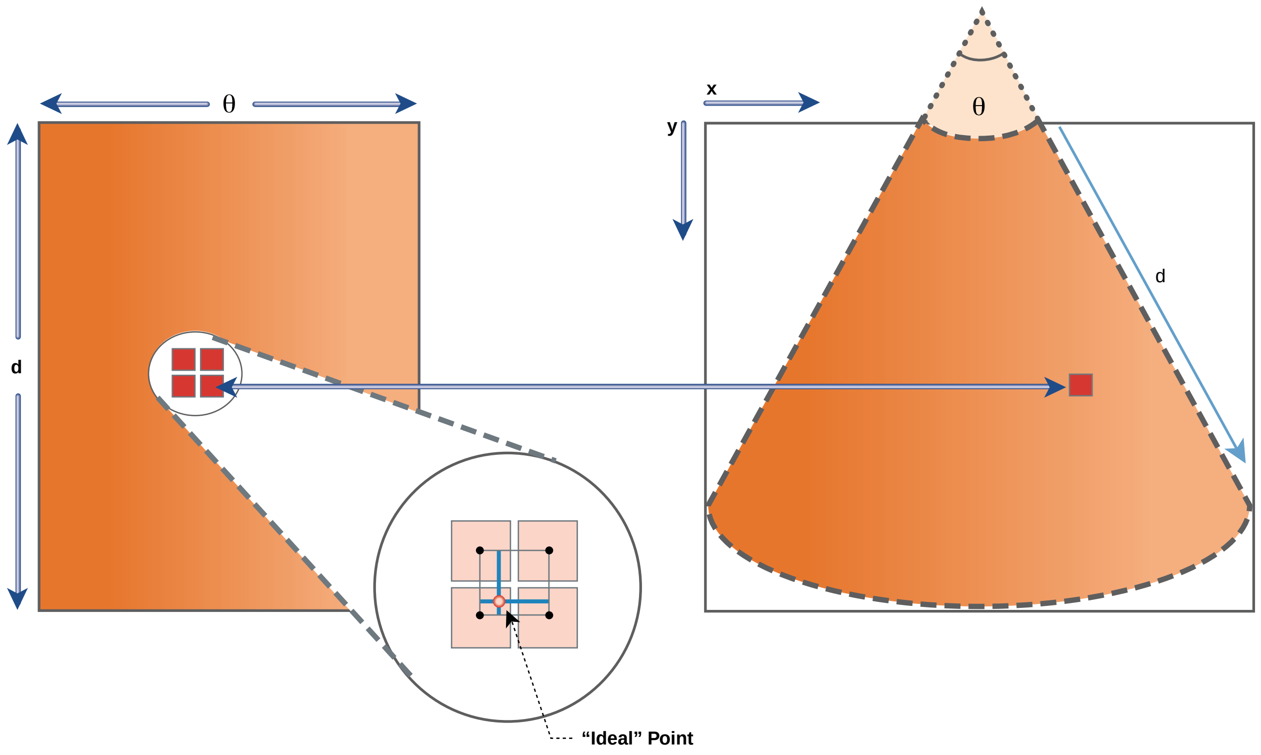


Figure 6: Illustration of the process for obtaining the output pixels values. Source: Intel, 2009

Performance Results

Table 4 and Table 5 show the performance results for each of the 3 DSP modules described above, running on Intel Core 2 Duo and Intel Atom processors. The benchmark was run on a single-thread, single–core configuration; that is, no high-grain parallelism was taken into consideration. Linux 2.6.18 was running on the Intel Core 2 Duo processor system (GCC 4.1.1 was used for compiling the application), and Linux 2.6.24 on the Intel Atom processor platform (GCC 4.2.3). Intel IPP version 6.0 was installed on both platforms.

Intel® Core™2 Duo Processor (2.533 GHz, 6 MB L2 cache)			
Algorithm	Processing requirements (MFLOPs)	Time to process one frame (ms)	Equivalent processing throughput (MFLOPs)
RX beamforming	3098.9	227.8	453.45
Envelope detection	437.8	4.55	3207. 3
Display processing	58.5	3.39	575.22
Total	3595.2	235.74	508.36
Total (excluding beamformer)	496.3	7.94	2083.5

Table 4: Performance results for the Intel® Core™2 Duo processor.

Intel® Atom™ Processor N270 (1.6 GHz, 512 KB L2 cache)			
Algorithm	Processing requirements (MFLOPs)	Time to process one frame (ms)	Equivalent processing throughput (MFLOPs)
RX beamforming	3098.9	1177.3	87.74
Envelope detection	437.8	22.78	640.62
Display processing	58.5	24.73	78.85
Total	3595.2	1224.8	97.85
Total (excluding beamformer)	496.3	47.51	348.21

Table 5: Performance results for the Intel® Atom™ processor.

Besides the lower clock frequency, other factors influence the lower performance values obtained with the Intel Atom processor: total available cache size and number of instructions retired per clock cycle. Being optimized for low-power applications, the instruction pipeline on Intel Atom processor is not able to retire as many instructions per clock cycle (in average) as the Intel Core 2 Duo processor.

Due to its more straightforward nature in terms of memory accessing, the envelope detector is the most efficient part of the processing chain.

The low performance values for the display processing algorithm are heavily due to the nonsequential memory access patterns. Besides generating many cache line and page misses, this also makes the algorithm unsuitable for vectorization, although it could still operate in a parallel fashion on a multi-core, multi-threaded platform.

One of the largest performance bottlenecks of the beamforming algorithm is caused by the varying delay values applied to the signals causing many nonaligned memory access patterns. Usually, and because of its high performance requirements, this part of the algorithm is offloaded to external, dedicated circuitry, mostly based on FP-GAs. Table 6 shows the benchmark results in terms of number of frames per second attainable for each of the platforms tested, excluding the beamformer algorithm.

"One of the largest performance bottlenecks of the beamforming algorithm is caused by the varying delay values applied to the signals causing many nonaligned memory access patterns."

Target frames/second	Benchmark results Intel® Core™2 Duo Processor	Intel® Atom™ Processor N270
30	125.94	21.05

Table 6: Benchmark results in frames/second excluding beamformer.

"This section, however, will show that two of the most demanding next-generation baseband processing algorithms can be effectively implemented on modern Intel processors."

While the Intel Atom processor seems not to be able to reach the initial 30 fps target, the Intel Core 2 Duo processor clearly does it and provides headroom to accommodate other runtime control and processing tasks needed in a fully functional ultrasound imaging application. It is also worth noting that opportunities for parallel processing exist in several parts of the algorithm, though they were not taken into consideration throughout this study.

Case Study: Wireless Baseband Signal Processing

In wireless communication systems, the physical layer (PHY) (baseband signal processing) is usually implemented in dedicated hardware (ASICs), or in a combination of DSPs and FPGAs, because of its extremely high computational load. GPPs (such as Intel architecture) have traditionally been reserved for higher, less demanding layers of the associated protocols.

This section, however, will show that two of the most demanding next-generation baseband processing algorithms can be effectively implemented on modern Intel processors—LTE turbo encoder [18] and channel estimation.

The following discussion assumes Intel architecture as the target platform for implementation, and the parameters shown in Table 7.

LTE Bandwith	20 MHz
FFT length	2048
Spatial Antenna MIMO	4x4, 1 sector
OFDM symbols per slot	7
Slots per frame	20
Frame duration	10 ms
Encoding	1/3 Parallel Concatenated Convolutional Turbo-Encoder
Raw Bit-rate	172 Mbit/s
Bit-rate at TE input	57 Mbit/s

Table 7: Parameters for LTE algorithm discussion.

LTE Turbo Encoder

The Turbo encoder is an algorithm that operates intensively at bit level. This is one of the reasons why it is usually offloaded to dedicated circuitry. As will be shown further, there are software architecture alternatives that can lead to an efficient realization on an Intel architecture platform.

"There are software architecture alternatives that can lead to an efficient realization on an Intel architecture platform."

The LTE standard [18] specifies the Turbo encoding scheme as depicted in Figure 7.

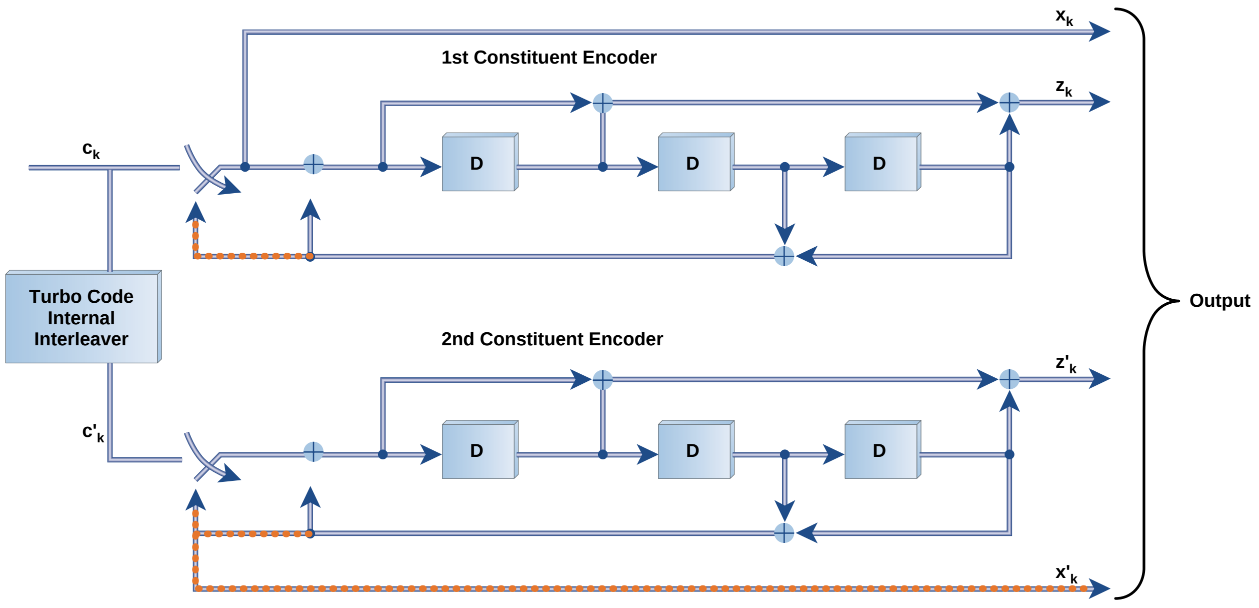


Figure 7: Block diagram of the LTE Turbo encoder. Source: 3GPP

The scheme implements a Parallel Concatenated Convolutional Code (PCCC) using two 8-state constituent encoders in parallel, and comprises an internal interleaver. For each input bit, 3 bits are generated at the output.

Internal Interleaver

The relationship between the input *i* and output *π(i)* bit indexes (positions in the stream) is defined by the following expression:

$$\pi(i) = (f_1 \cdot i^2 + f_2 \cdot i)_{mod\,(K)}$$

K is the input block size in number of bits (188 possible values ranging from 40 to 6144). The constants *f1* and *f2* are predetermined by the standard, and depend solely on *K*.
At a cost of a slightly larger memory footprint (710 kilobytes), it is possible to pre-generate the *π(i)* LUTs for each allowed value of *K*. For processing a single data frame, only the portion of the table referring to the current *K* value will be used (maximum 12 KB).

Computing the permutation indexes at runtime would require 4 multiplications, 1 division and 1 addition, giving a total of 6 integer operations per bit.

Convolutional Encoders

Each convolutional encoder implements a finite state machine (FSM) that cannot be completely vectorized or parallelized due to its recursive nature.

In terms of complexity, the implementation of this state machine requires 4 XOR operations per bit per encoder. If all the possible state transitions are expanded and stored in a LUT, the number of operations is 8 per byte per encoder.

"Each convolutional encoder implements a finite state machine (FSM) that cannot be completely vectorized or parallelized due to its recursive nature."

"For parallel implementation, each thread receives a portion of the input data stream within the 6144-bit maximum range."

Total Computational Requirements

For an input rate of 57 Mbit/s, the Turbo encoder requires 57.34 MOP (Million Integer Operations) for processing a single 10-ms frame.

Internal Interleaver Implementation

In order to allow parallelization and vectorization, the algorithm was changed by replacing the mod operation with a comparison test and a subtraction. Also, the inter-sample dependence was reassessed for allowing 8-way vectorized implementation as follows:

$$\rho_L(n, i) = \pi(i - L) + (f_1 \cdot L + f_2 \cdot L \cdot (2 \cdot n + L))_{mod\,(K)}$$

$$\tau_L(i) = \rho_L(and(i, L - 1), i)$$

$$\pi(i) = \tau_L(i) - K \cdot \tau_L(i) < K ?)$$

For parallel implementation, each thread receives a portion of the input data stream within the 6144-bit maximum range. The results (in CPU cycles) per input byte are given in Table 8 for the reference system described below. These results are included in the overall Turbo encoder performance measurements presented ahead. As can be seen from the results in Table 8, performance scales in an almost linear manner with the number of threads.

Threads	Cycles per byte
1	19.76
2	9.745
4	4.99

Table 8: CPU cycle counts per byte on multithreaded implementation of internal interleaver.

Convolutional Encoder Implementation

The implementation for this block comprises two steps:

- Generate all possible FSM state transitions and output values for each input value, regardless of the current state of the FSM. Generation of the output values is done in parallel.
- From the results generated in the previous step, select the one that corresponds to the actual state. All other generated results are discarded. The two convolutional encoders operate in parallel during this step.

A LUT is used that stores the pre-computed transition matrix (for each possible value of the input and FSM state). The size of this LUT depends on the number of bits sent to the encoders in each iteration.

"A LUT is used that stores the pre-computed transition matrix (for each possible value of the input and FSM state)."

Table 9 shows the number of cycles it takes to encode the input stream, as well as the memory footprint, per iteration. It can be seen that on Intel architecture, the large cache size allows a more flexible tradeoff between performance and memory usage. In this case, a 128-KB table is used.

input (bits)	LUT Size (bytes)	Clks	Clks/Byte
1	32	6.41	51.28
4	256	6.54	13.08
8	4096	6.57	6.57
12	131072	6.55	4.37
16	2097152	9.51	4.76

Table 9: CPU cycle counts for the LUT-based encoder.

Overall Performance Results for the Complete Turbo Encoder

From Tables 8 and 9, the internal interleaver takes 4.99 cycles per byte, using 4 independent threads for an input block size of 6144 bits, while the encoder, which uses 2 threads, takes 4.76 cycles per byte. As there is no inter-block dependence it is possible to run two encoders in parallel on the reference platform [19].

As a result, a 10-ms frame (57 Mbps) is encoded in 159.1 microseconds corresponding to a total CPU usage of 1.59 percent.

Channel Estimation

On the next generation of mobile wireless standards the estimation of the channel characteristics is necessary to provide high data throughputs. LTE includes a number of reference signals in its data frame that are used to compute the estimation, as illustrated in Figure 8.

These reference signals are sent every 6 subcarriers with alternate frequency offsets. They are sent on the first and fourth OFDM symbols of each slot, so two channel estimations are computed per slot.

The estimation consists of a time average of the current reference frame and the 5 previous ones, in order to minimize noise distortion.

Figure 9 represents the high-level view of the channel estimator, comprising a complex reciprocal operation (rcp(z)), a complex multiplication per each set of reference values, an averaging operator (Σ in Figure 9) and a polyphase interpolator (H(z)).

In terms of computational complexity per sample:

- Reciprocal calculation: 6 multiplications, 1 division and 1 addition.
- Complex multiplication: 4 multiplications and 2 additions.
- Averaging operation: 6 additions and 1 multiplication.
- Polyphase interpolator: 6 multiplications and 3 additions.
- Total number of operations: 30.

For a 10-ms full 4x4 MIMO, 20-MHz frame, the algorithm computes 120 channel estimations, where only 340 samples per frame are used. Multiplying this by the total number of operations per sample we get a total of 1.224 MFLOP per frame.

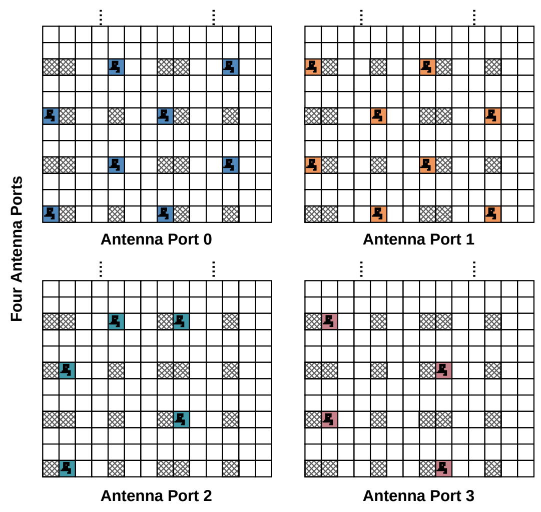


Figure 8: Spacing of Reference signals on each antenna. Source: 3GPP LTE Standard

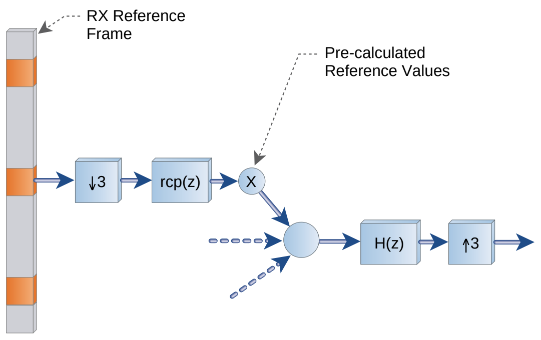


Figure 9: High-level view of the channel estimator. Source: Intel Corporation, 2009

Implementation

The input data parameters are assumed as described in Table 10.

Input Type	Fixed point 16-bit IQ pairs
ADC	12 bits
Frame size	2048 complex samples
Reference points in frame (per antenna)	340 complex samples

Table 10: Input data format for channel estimation.

Only the complex multiplications and reciprocals are computed in floating point. Reciprocals in particular are implemented with SSE intrinsics for a higher throughput. The performance results in CPU cycles per reference input sample are presented in Table 11.

	Reciprocal Calculation	Complex multiplication	Interpolation	Averaging	Total
Cycles/sample	7.53	4.51	7.68	0.29	20.01

Table 11: CPU cycles per complex input sample for each stage of the channel estimation algorithm.

For a 10-ms frame, and assigning two cores per MIMO channel on our system [19], each thread computes a total of 20 estimations per frame, resulting in 47.2 microseconds processing time per frame, and a total CPU usage of 0.48 percent.

Overall Turbo Encoder and Channel Estimation Performance

Table 12 summarizes the performance results of the Intel architecture implementation for both algorithms. The first column states the computational complexity of the algorithm in terms of millions of (floating-point) operations per frame. The second shows the actual time taken by our reference system to process the data (using the 8 cores available). The final column is the total CPU usage for processing the 57 Mbps data stream.

Dual Intel® Core™ i7 Processor 2120 MHZ (4 cores/CPU, 1.5 GB DDR3/CPU, 8 MB cache/CPU)			
Algorithm	10-ms Frame Processing requirements (MOP/MFLOP)	Time to process a 10-ms frame (microsecs)	CPU usage
Turbo-Encoder	57.34	159	1.59%
Channel Estimation	1.224	47.3.	0.48%

Table 12: Summary of performance results for selected baseband proccessing.

While the actual partitioning of the system will depend on the amount of baseband processing offloaded or/and throughput required, the results show that it is possible to move several portions of the baseband processing into an Intel arechitecture-based platform.

“Sophisticated military radar post-processing is certainly not the first embedded DSP application that comes to mind.”

Case Study: SARMTI—An Advanced Radar Post-Processing Algorithm

Sophisticated military radar post-processing is certainly not the first embedded DSP application that comes to mind. Yet processing efficiency is always a concern: projects are always seeking the highest performance possible within a fixed thermal and volume footprint.[13]

A major US defense contractor asked Intel and N.A. Software Ltd[14] (NASL) to determine how much the performance of a highly innovative radar algorithm "SARMTI" could be increased by multi-threading it and running it across multiple Intel architecture cores. The results could then be used to estimate the minimum size, weight, and power systems of various capabilities would require.

SARMTI was developed by Dr. Chris Oliver, CBE, of InfoSAR[15]. It combines the best features of the two predominant radar types in use today: Moving Target Indication (MTI) and Synthetic Aperture Radar (SAR). The computational loads to perform the SARMTI processing in needed time frames is an N^X computational problem. But by focusing on the underlying physics, Dr. Oliver has discovered a way to transform the problem into a much more manageable "N*x" problem.

*"The computational loads to perform the SARMTI processing in needed time frames is an NX computational problem. But by focusing on the underlying physics, Dr. Oliver has discovered a way to transform the problem into a much more manageable 'N*x' problem."*

The basic difficulty with currently deployed airborne systems is that they must often consist of both MTI and SAR radar systems, with their separate waveforms, processing, and display modules. This is because MTI systems are very good at tracking fast-moving ground and airborne objects, but slow-moving or stationary objects degrade the image. Imaging radar systems, such as SAR, are capable of resolving stationary ground objects and features to less than 1 meter, but any movement (of the airplane or objects on the ground) shifts and blurs the image, so positions of moving objects cannot be accurately determined. Therefore current systems must rely on highly trained radar operators to register the moving target data collected/processed during one time period using MTI waveforms with the images of the ground collected using SAR waveforms during a different time period. Once registered, analysis and correlation of the disparate images is also often performed manually.

NASL began the SARMTI multi-threading project by using the GNU profiler gprof to determine where to focus their work. It showed that the serial algorithm was spending 64 percent of its time compressing complex data and about 30 percent of its time detecting targets. So NASL threaded those areas, resulting in an overall algorithm structure diagrammed in Figure 10.

Since SARMTI is a post-processing algorithm, it begins after a raw SAR image (>14 MB) is loaded into memory. Some serial (non-threaded) processing is done at the beginning, then again during a small synchronization process in the middle and then at the end to display the image. But during the data compression and target detection phases, data tiles are processed independently on each core (represented by the TH[read] boxes.) NASL did not use core or processor affinity to assign specific threads to specific cores or processors; they let Linux dynamically place each process on the core with the least load. Analysis showed that core utilization was in fact quite balanced.

NASL next turned their attention to optimizing FFT and vector math operations, since SARMTI contains many billions of FFT and math operations. The original SARMTI code used FFT performance libraries from FFTW,* so the Intel Math Kernel Library (Intel MKL) "FFTW Wrappers" were substituted. In addition, the original C versions of complex vector add, conjugate, and multiply operations

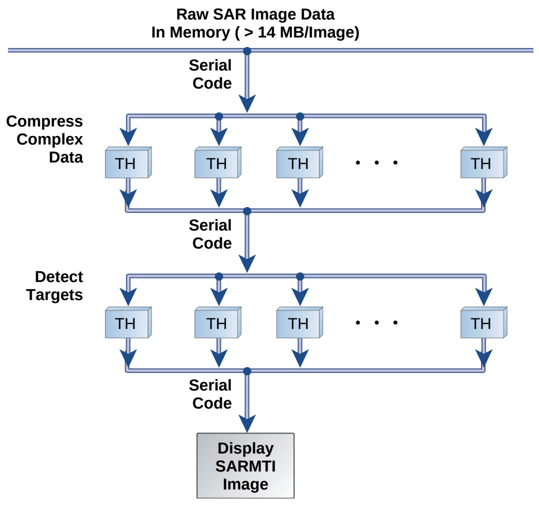


Figure 10: Conceptual Structure of SARMTI
Source: NA Software Ltd., 2009

were replaced with the corresponding functions in the Intel MKL. Total Intel MKL speed-up by itself ranged from 14.7 to 18.4 percent.

Table 13 summarizes the overall results of these efforts when the multi-threaded algorithm was run on a four-socket Intel rack mount server.[17]

Test Scenario	Original Non-Threaded Time	Hardware Threads (Cores)						Speed Up (1T→24T)	Total Speed Up (0→24T)
		1T	2T	4T	8T	16 T	24 T		
test1	85.4	36.2	18.4	9.5	5.5	4.1	2.9	12.6X	29X
test2	120.2	44.8	23	12.2	7.1	5.1	3.7	12X	32X
test3	104	35.3	17.9	9.3	5.4	4	2.8	12.4X	17X
test4	166.2	59.5	30.9	16.5	9.5	6.6	4.9	12X	33X

Table 13: Total SARMTI performance increase: 0 to 24-cores and threads (in seconds) (4x Intel® Xeon® Processors X7460)
Source: NA Software Ltd, 2009

"The large performance gains from the original, totally serial code, to the multi-threaded version (1T) were realized by optimizing the algorithm during the multi-threading process."

The overall speed up from the original serial code to the multi-threaded code running with 24 threads across 24 cores ranged between 17 and 33 times. The large performance gains from the original, totally serial code, to the multi-threaded version (1T) were realized by optimizing the algorithm during the multi-threading process (in addition to the previously mentioned gains from Intel MKL). The speed-up from multi-threaded code running on 1 core to 24 threads running on 24 cores was about 12X for all test scenarios. Figure 11 shows how performance scaled per core.

The slope of the curves shows that SARMTI scales quite well from one to eight cores. The rate of increase slows after eight threads/cores, but performance did continue to increase.

NASL investigated a number of areas to see if scaling per core could be increased. Neither front side bus nor memory bandwidth turned out to be issues. Cache thrashing was also not a problem since NASL had been careful to use localized memory for each thread. Early portions of the data compression stage are the only place where threads do process data from the same area of memory since they are all starting with the same input image. But changing the algorithm to make N copies of the image and then processing that unique memory block on each thread introduced overhead that actually increased execution times.

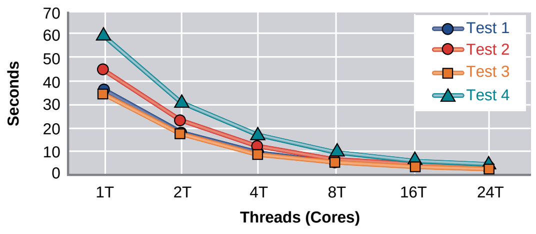


Figure 11: SARMTI scalability graphed per number of cores. Source: NA Software, Ltd.

It turned out that some parts of the algorithm simply threaded more efficiently than others. Different portions of the algorithm use differently sized data sets, whose sizes change dynamically as the geometry changes. Some of the data sets simply do not thread efficiently across 24 cores.

The next phase of the project will be to determine the performance increases (and hence potential reduction in system size, weight and power) that tightly coupling FPGAs to Intel Xeon processors will bring.[17]

Conclusions

Modern Intel general purpose processors incorporate a number of features of real value to DSP algorithm developers, including high clock speeds, large on-chip memory caches and multi-issue SIMD vector processing units. Their multiple cores are often an advantage in highly parallelizable DSP workloads, and software engineers can write applications at whatever level of abstraction makes sense: they can use higher-level languages and take advantage of compiler's automatic vectorization features. They can further optimize performance by linking in Intel IPP and MKL functions. In addition, if certain areas require it, SSE intrinsics are available, or the rich and growing set of specialized SSE and other assembly instructions can be used directly.

The ultrasound and LTE studies we have summarized indicate that current Intel Architecture Processors may now be suitable for a surprising amount of intensive DSP work, while the SARMTI performance optimization study demonstrates the kind of impressive performance increases multi-threading can unlock.

References

[1] The Freescale* MPC 8641D processor and Intel® Core™2 Duo processor T7400 were measured as installed in GE Fanuc* DSP230 and VR11 embedded boards. The VXWorks* 6.6 version of NASL's VSIPL* library was used. The Intel® Core™2 Duo processor SL9400 was measured in an HP* 2530P laptop and is included to provide performance figures for a later, lower-power version of the Intel® Core™2 Duo processor architecture. NA Software* has both Linux* and VxWorks 6.6 versions of their VSIPL libraries for Intel® architecture, and used the Linux versions with the Intel® processors. There is no significant performance difference between the VXWorks and Linux versions in these applications. They chose to use the Linux version for these tests because Linux was easier to install on the HP laptop. All timings are with warm caches.

[2] Intel® Advanced Vector Extensions (Intel® AVX) information is available at *http://software.intel.com*

[3] Thread affinity is the ability to assign a thread to a single processor core.

[4] Intel Corporation. "Intel® 64 and IA-32 Architectures Software Developer's Manuals." URL: *http://www.intel.com/products/processor/manuals/*

[5] Aart J. C. Bik. "The Software Vectorization Handbook". Intel Press. May, 2004.

[6] Aart Bik, et al. "Programming Guidelines for Vectorizing C/C++ Compilers." Dr. Dobb's Newsletter. February, 2003. URL: *http://www.ddj.com/cpp/184401611*

[7] Aart J. C. Bik, et al. "Automatic Intra-Register Vectorization for the Intel® Architecture." International Journal of Parallel Programming, Vol. 30, No. 2, April 2002.

[8] Aart Bik, et al. "Efficient Exploitation of Parallelism on Intel® Pentium® III and Intel® Pentium® 4 Processors-Based Systems." Intel Technology Journal. February, 2001.

[9] Information on Intel® software products can be found at *http://software.intel.com*

[10] H. T. Feldkamper, et al. "Low Power Delay Calculation for Digital Beamforming in Handheld Ultrasound Systems." IEEE Ultrasonics Symposium, pp. 1763-1766, 2000

[11] Jacob Kortbek, Svetoslav Nikolov, Jørgen Arendt Jensen. "Effective and versatile software beamformation toolbox." Medical Imaging 2007: Ultrasonic Imaging and Signal Processing. Proceedings of the SPIE, Volume 6513

[12] P. P. Vaidyanathan. "Multirate Systems and Filter Banks." Prentice Hall, 1993.

[13] See, for example proceedings of the High Performance Embedded Computing Workshop at *http://www.ll.mit.edu/HPEC*

[14] N.A. Software Ltd. information is at *http://www.nasoftware.co.uk/*

[15] More information on SARMTI can be found at *http://www.infosar.co.uk*

[16] NA Software Ltd. measured the performance of SARMTI on the Intel® SFC4UR system with four Intel® Xeon® Processors X7460, each with 6 cores running at 2.66 GHz, and 16 MB of shared L3 cache. Sixteen MB 667-MHz FBDIMMs, Fedora* release 8 (Werewolf*) for x86_64 architecture (Linux* 2.6.23 kernel), GCC 4.1.2, flags: -O3 –Xt –ip –fno_alias –fargument-noalias, Intel® C++ Compiler 10.0; Compile flags: icc –O3 –Xt –ip –fno_alias –fargument-noalias;* Intel® Math Kernel Library (Intel® MKL) version 10.0

[17] See the Intel® QuickAssist Technology references to Xilinx* and Altera* FPGA in-socket accelerator modules available from XtremeData* and Nallatech* at *http://www.intel.com/technology/platforms/quickassist/*

[18] The LTE specification is available at *www.3gpp.org*

[19] Reference system: Dual Intel® Core™ i7 Processor 2112 MHz (8 MB Cache/CPU, 1.5 GB DDR3 800MHz/CPU. 64-bit CentOS 5.0, Intel® C++ Compiler 10.0.0.64, 80 GB HD Samsung* 5400 rpm)

Author Biographies

David Martinez: David Martinez joined the DSP on IA team in February 2008. Since then, he has been working on implementing Wireless Baseband algorithms on Intel® architecture. Previously, David had been working for three years on codec and signal processing optimization for mobile devices, mainly in H264, MPEG-4 decoding and DVB-H demodulation.

He received his ME on Telecommunications and Electrical Engineering at the Polytechnic University of Madrid (UPM) and the Ecole Superieure d'Electricite of Paris in 2005.

Vasco Santos is a DSP software engineer working in the DSP on IA team based in Shannon, Ireland, where he has been developing work on medical ultrasound imaging and wireless baseband signal processing.

Prior to joining Intel in May 2008, Vasco was a senior digital design engineer at Chipidea Microelectronica, S.A., Portugal, where he spent 4 years developing efficient ASIC DSP architectures for delta-sigma audio data converters and wireless baseband frontends. Vasco also has 2 years of research and development experience in semantic characterization of audio signals. He received his B.S. and M.S. degrees in electrical and computer engineering from the Faculty of Engineering of the University of Porto, Portugal, in 2002 and 2005, respectively.

Martin Mc Donnell: Martin Mc Donnell is a system architect working in the ADS team based in Shannon, Ireland, where he is active in the fields of voice and wireless processing. He has over 20 years experience in the embedded communications arena. Prior to joining Intel's Accelerated DSP Software team, Martin worked in a number of companies (including Digital Equipment Corp., Tellabs, Avocent Corp.) specializing in the field of data and multimedia communications, producing products in the IP networking, telephony, multimedia over IP, and ultra low latency video codec technology areas.

Ken Reynolds: Ken Reynolds is engineering manager for the ADS (Accelerated DSP Software) team based in Shannon, Ireland.

Ken has nearly 18 years experience in the industry, initially in the area of high speed digital design (encompassing embedded, ASIC, FPGA, RF, and DSP) and more recently in leading research and development teams, which, in his previous two companies (Azea Networks and Alcatel), focused on signal conditioning and error correction for high bit rate optical communication systems. He has worked mostly in the telecommunications and defense industries in the UK and USA.

Ken joined Intel in January, 2008.

Peter Carlston: Peter Carlston is a platform architect with Intel's Embedded Computing Division. He has held a wide variety of software and systems engineering positions at Unisys and Intel.

Copyright

IA-32 FEATURES AND FLEXIBILITY FOR NEXT-GENERATION INDUSTRIAL CONTROL

Contributors

Ian Gilvarry
Intel Corporation

Index Words

Intel® Atom™ processor
programmable logic controllers
fieldbus
real-time Ethernet
software-plc

Abstract

Industrial control systems are rapidly evolving towards standardized, general-purpose platforms that incorporate concepts traditionally associated with the domain of Information Technology (IT). The push of IT into the industrial sector is occurring both at the field level, where sensors and actuators are more and more intelligent, and at the control level, to replace the dedicated hardware approach found in previously designed applications. New programmable logic controllers (PLCs) are being designed using commercial off the shelf (COTS) hardware based on embedded PCs. Key to the design are benefits associated with PC software architectures where designers have many choices to incorporate the reliability, determinism, and control functions that are required. This makes the PC software extremely flexible and well suited for complex applications. These new types of industrial controllers are in effect open control platforms that bring into scope the advantages inherent in the PC industry including open programming, connectivity, and greater flexibility.

This article describes a suggested design approach for an open control platform using the Intel® Atom™ processor. It illustrates how these new processors provide the benefits of IA-32 open architectures while at the same time meeting the power and cost envelope associated with designs at the control level in industrial factory automation.

Traditional Industrial Automation Control

Traditional industrial automation control has been implemented using the programmable logic controller (PLC), a programmable microprocessor-based device used to control assembly lines and machinery on the shop floor as well as many other types of mechanical, electrical, and electronic equipment in a plant. Typically programmed in an IEC 61131 programming language, a PLC was designed for real-time use in rugged, industrial environments. Connected to sensors and actuators, PLCs were categorized by the number and type of I/O ports they provided and by their I/O scan rate.

For over two decades PLCs were engineered using proprietary architectures. These PLCs were based on dedicated hardware platforms, with real-time operating systems (RTOS), and functions strictly limited to the actions to be performed. With such PLCs, when you selected a particular vendor and PLC family you were locked into the corresponding boards and functions that were available to that particular line. While this approach offers easy-to-integrate hardware, high quality components, and knowledgeable support, it also is closed to unusual implementations or deviations from standard configurations.

"With such PLCs, when you selected a particular vendor and PLC family you were locked into the corresponding boards and functions that were available to that particular line."

PLCs have served well as individual islands of manufacturing control. However, digital factory automation has evolved into complex, interconnected manufacturing cells. Process control data flows upwards from the cell into the MRP system, as dynamically reconfigurable process steps flow downwards. "Just in Time" (JIT) product distribution and an increasing number of offered products drive companies towards reconfigurable manufacturing. Connecting the work cells into the plant's MRP system requires new communication interfaces and also creates a demand for statistical information and additional data acquisition at the work cell. Work cells are also increasing in sensor count and complexity. Often these newer sensors are difficult to interface with traditional PLC hardware. The communications interface, the statistical functions, the data acquisition functions, and the new sensors are often difficult to add to the traditional PLC.

"Connecting the work cells into the plant's MRP system requires new communication interfaces and also creates a demand for statistical information and additional data acquisition at the work cell."

Towards Embedded Processors for PC-Based Industrial Automation

In recent years industrial control systems have been transitioning more towards standardized, general-purpose platforms based on the adoption of PC technology. One of the fundamental drivers for this has been the desire by end users to merge their information technology and automation engineering systems into one complete end-to-end platform. The push of information technology into automation is happening at the field Level where sensors and actuators are more and more intelligent, and also at the control level, where embedded PC technology is being used to replace the traditional PLC.

As well as convergence of information systems and automation engineering, the deployment of web-service based architectures and the proliferation of industrial Ethernet are additional factors that are influencing end users and original equipment manufacturers (OEMs) to migrate industrial control systems to PC-based architectures.

Special software packages for embedded PC platforms implement the functions that traditionally were implemented in separated dedicated hardware. Advantages here are many including:

- No need of dedicated hardware
- Integration of different functions in a single machine (HMI and PLC run in a single embedded PC)
- Ease of interfacing basic control functions with high level functions
- Native remote communication using Ethernet or the Internet

Today a digital factory system includes a network of intelligent field devices and one or more dedicated devices for running the control tasks that are called controllers. Additional devices may be used for human machine interface (HMI), remote communication, data storage, advanced control, and other tasks.

"One of the fundamental drivers for this has been the desire by end users to merge their information technology and automation engineering systems into one complete end-to-end platform."

"Traditionally the terms 'low power' and 'ultra low power' when used in relation to Intel® processor platforms have been at odds with the definitions used in embedded designs."

Traditionally the terms "low power" and "ultra low power" when used in relation to Intel® processor platforms have been at odds with the definitions used in embedded designs. Typically there was an order of magnitude difference between the two, with Intel's lowest power platform, of the order of 10 W, compared to a typical 1-W envelope for a low power embedded platform. This challenge was a barrier for the adoption of Intel® architecture into the fanless, completely sealed designs commonly required in the typical harsh working environment of industrial control. Designers were faced with the dilemma of designing expensive thermal solutions to be able to adopt the benefits of PC architectures into the industrial control arena.

Realizing the Threshold for Fanless Industrial Control Designs

The Intel® Atom™ processors are the first of a new generation of processors over the coming years from Intel that will focus on addressing demand for performance in the tight constraints and harsh operating environments typically associated with industrial automation. Designs will benefit from being designed with the open architectures associated with PC technology but at the same time meet the demands of miniaturization associated with small form factors platforms, and cost-effectively meet the demand for more distributed intelligence in the factory.

The Intel® Atom™ processor Z5xx series brings the Intel® architecture to small form factor, thermally constrained, and fanless embedded applications. Implemented in 45 nm technology, these power-optimized processors provide robust performance-per-watt in an ultra-small 13x14 mm package.

These processors are validated with the Intel® System Controller Hub US15W (Intel® SCH US15W), which integrates a graphics memory controller hub and an I/O controller hub into one small 22x22 mm package. This low-power platform has a combined thermal design power under 5 W, and average power consumption typically less than 2 W.

"Designs will benefit from being designed with the open architectures associated with PC technology but at the same time meet the demands of miniaturization associated with small form factors platforms, and cost-effectively meet the demand for more distributed intelligence in the factory."

Intel® Atom™ Processor Features

- Intel's 45 nm technology, based on a Hafnium, high-K metal gate formula, is designed to reduce power consumption, increase switching speed, and significantly increase transistor density over previous 65 nm technology.
- Multiple micro-ops per instruction are combined into a single micro-op and executed in a single cycle, resulting in improved performance and power savings.
- In-order execution core consumes less power than out-of-order execution.
- Intel® Hyper-Threading Technology (Intel® HT Technology; 1.6-GHz version only) provides high performance-per-watt efficiency in an in-order pipeline. Intel HT Technology provides increased system responsiveness in multitasking environments. One execution core is seen as two logical processors, and parallel threads are executed on a single core with shared resources.

The evolution of low power Intel architecture was realized through a number of technological advances and some common sense power budget analysis. The power considerations of typical embedded platforms break down into two key areas, heat dissipated and average power consumed.

Using the Intel® Pentium® M processor, the analysis focused on identifying the main power consumers within the instruction pipeline. This is a 14-stage, 3-way superscalar pipeline whose instruction execution engine is based on an out-of-order execution scheduler. This analysis highlighted not only the large amount of power required for the execution scheduler logic but also significant power consumed by the ancillary logic, which optimizes instruction flow to the scheduler.

The pipeline stages were deconstructed and rebuilt as a 2-way superscalar, in-order pipeline, allowing many of the power-hungry stages to be removed or reduced, leading to a power savings of over 60 percent compared to the Intel Pentium M processor, as Figure 1 illustrates. Following this, the next stage was to examine the delivery of instructions and data to the pipeline. This highlighted two major elements, the caches and front side bus (FSB).

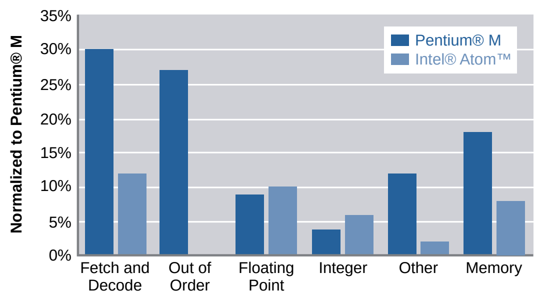


Figure 1: Pipeline power savings.

The L2 cache was designed as an 8-way associative 512-KB unit, with the capability of reducing the number of ways to zero, through save of dynamic cache sizing to use power. L2 pre-fetchers are implemented to maintain an optimal placement of data and instructions for the processor core.

"A CMOS FSB implementation was found to be more suited to low power applications, consuming less than 40 percent of an AGTL interface."

The FSB interface connects the processor and system controller hub (SCH). The FSB was originally designed to support multiprocessor systems, where the bus could extend to 250 mm and up to four loads: this is reflected in the choice of logic technology used in the I/O buffers. AGTL+ logic, while providing excellent signal integrity, consumes a relatively large amount of power. A CMOS FSB implementation was found to be more suited to low power applications, consuming less than 40 percent of an AGTL interface.

One of the key enabling technologies for low power Intel architecture was the transition in manufacturing process to 45-nm High-K metal gate type transistors. As semiconductor process technology gets ever smaller, the materials used in the manufacture of transistors has come under scrutiny, particularly the gate oxide leakage of SiO2. To implement 45-nm transistors effectively, a material with a high dielectric constant was required (High-K). One such material is Hafnium (Hf) and provides excellent transistor characteristics, when coupled with a metal gate.

"The Intel Atom processor can use Intel HT Technology on its two execution pipelines to increase performance by up to 30 percent on applications that can make use of the multi-threaded environment."

In embedded systems in-order pipelines can suffer from the problem of stalls due to memory access latency issues. The resolution of this problem came from an unusual source. Intel HT Technology enables the creation of logical processors, within a single physical core, capable of executing instructions independent of each other. As a result of sharing physical resources, Intel HT Technology relies on the processor stall time on individual execution pipelines to allow the logical processors to remain active for a much longer period of time. The Intel Atom processor can use Intel HT Technology on its two execution pipelines to increase performance by up to 30 percent on applications that can make use of the multi-threaded environment.

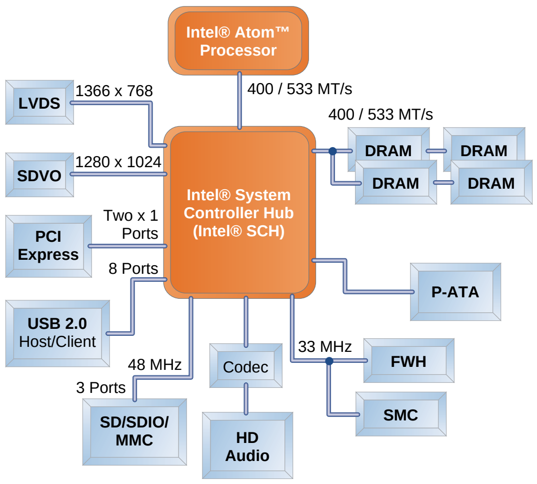


Figure 2: Typical Intel® Atom™ processor platform.

In order to maximize the performance of the pipeline, the Intel® compiler has added "in-order" extensions, which allow up to 25-percent performance improvement compared with code compiled using standard flags.

As has been long included in the standard IA-32 instruction set, the Intel Atom processor supports the SIMD extensions up to Intel® Streaming SIMD Extensions 3.1 (Intel® SSE3.1). These instructions can be used to implement many media and data processing algorithms. Traditionally considered the domain of the DSP, the SSE instructions are executed in dedicated logic within the execution pipeline.

Delivering a low power processor on its own does not necessarily meet the needs of an embedded low power market, where low power, small platform footprint and low chip count tend to be the key cornerstones of a typical design.

To address this, the Intel Atom processor platform is paired with an Intel® System Controller Hub (Intel® SCH), which takes the traditional components of memory controller, graphics and I/O complex, integrated into a single chip, attached to the Intel Atom processor platform over a 400-MHz/533-MHz FSB. Figure 2 shows a typical Intel Atom processor platform.

To meet the need for a small footprint, the processor and chipset are offered in ultra-small footprint packages, with a size of 13 mm x 14 mm and 22 mm and 22 mm respectively. This footprint enables complete platforms to be developed with an area of less than 6000 mm.^{2}

"To meet the need for a small footprint, the processor and chipset are offered in ultra-small footprint packages, with a size of 13 mm x 14 mm and 22 mm and 22 mm respectively."

The Intel System Controller Hub continues the delivery of features and attributes suitable for the low power embedded market. The main features of the Intel System Controller Hub are described below:

- The memory interface is a single channel 32-bit DDR-2 memory, capable of implementing un-terminated memory-down solutions of up to 2 GB locked to the FSB speed.
- Closely coupled to the memory controller is the 3D graphics subsystem, sharing system memory in a Unified Memory Architecture (UMA) configuration.
- The graphics controller offers respectable 3D performance and also has the ability in hardware to completely decode a range of video streams (MPEG 2 and 4, H.264 WMV9/VC1, and others), removing this task from the main processor core.
- The graphics controller can output two simultaneous independent streams using an LVDS and sDVO interface, these display interfaces may be configured using the embedded graphics driver configuration tool.

Embedded applications are usually defined by their I/O requirements. The Intel SCH provides the designer a range of interfaces, from USB ports, which may operate in Host or Client mode, SDIO/MMC controllers supporting a wide range of card types and an eIDE P-ATA controller, which enables the use of the latest solid state drives (SSDs) and provides the designer with a storage interface that can easily be switched in and out of low power states (SATA interfaces require more link management and cannot easily be turned on and off when not in use). In addition to the integrated features, the Intel SCH offers two PCI Express* x1 ports for further expansion.

"The Intel SCH provides the designer a range of interfaces."

Systems Architecture Considerations for Embedded PC-based Platforms versus Dedicated Hardware

In contrast to traditional PLCs, the next-generation industrial controllers based on embedded PC processors, which are sometimes also referred to as programmable automation controllers, handle multiple domains: not only logic, but motion, drives, and process control on a single platform. This brings key advantages including the ability to use a single development environment and also enables the use of open architectures for programming languages, and network interfaces.

"The next-generation industrial controllers based on embedded PC processors handle multiple domains: not only logic, but motion, drives, and process control on a single platform."

Characteristics of industrial controllers with embedded PC processors:

- Tight integration of controller hardware and software
- Programmable to enable design control programs to support a process that "flows" across several machines or units
- Operation on open, modular architectures that mirror industry applications, from machine layouts in factories to unit operation in process plants
- Employment of de facto standards for network interfaces, languages, and protocols
- Provision of efficient processing and I/O scanning

The key to realizing industrial control designs is to incorporate support for the many bus networks that exist in the factory environment. This takes into account the situation today in which machine builders, OEMs, systems integrators, and users have a plethora of fieldbus solutions they have to consider for use on their automation projects. These fieldbus solutions allow the common support of field measurement, control, status, and diagnostic information. For motion control and real-time tasks this information needs to be exchanged in a deterministic manner between field devices and automation controllers.

"The key to realizing industrial control designs is to incorporate support for the many bus networks that exist in the factory environment."

Commonly accepted fieldbus protocols for industrial automation applications are summarized in Table 1.

The industrial automation community often refers to "real-time" when discussing capabilities of industrial automation systems. But what are the requirements for industrial real-time? It needs to be put into context as different applications have different real-time needs. The most stringent requirements for motion control involve cycle times of around 50 microseconds and permissive jitter (deviation from the desired cycle time) of around 10 microseconds. Special applications with requirements tighter than this must be handled with application-specific hardware; normal industrial fieldbus–based systems cannot handle those applications. Typical cycle times for position control lie in the 1 to 4 milliseconds range, but have very short jitter times, usually less than 20 microseconds. Pure PLC sequential logic usually does not require less than 10 milliseconds cycle times and jitter can be in milliseconds range. Communication with higher level computers will be in the seconds range.

"But what are the requirements for industrial real-time? It needs to be put into context as different applications have different real-time needs."

Field bus technology	Standards	General information
Foundation Fieldbus (FF)	IEC/EN 61784-1 CPF 1, IEC61158 Type 1	Process bus, up to 32 devices, speed 31,25 kbit/s, 2.5 Mbit/s or 10 Mbit/s, up to 1900 range at lowest speed
ControlNet	IEC/EN 61784-1 CPF 2, IEC61158 Type 2	Universal Ethernet/IP bus, up to 99 nodes, 5Mb/s, 1000/3000 meters
Profibus	IEC/EN 61784-1 CPF 3, IEC61158 Type 3	Universal bus, up to 32 nodes per segment and up to 125 nodes in network, electrically RS-485, speeds from 9.6 kbit/s to 12 Mbit/s, up to 1200 meters at low speeds
P-Net	IEC/EN 61784-1 CPF 4, IEC61158 Type 4	Two wire circular network, up to 32 hosts / 125 devices, electrically RS-485, sped 78.6 kbit/s
FP High Speed Ethernet (HSE)	IEC/EN 61158 Type 5	Adaptation of Foundation Fieldbus to Ethernet, uses 100 Mbit/s Ethernet media
WorldFIP	IEC/EN 61784-1 CPF 5, IEC61158 Type 7	Universal bus, up to 256 nodes per bus, speeds 31.25 kbit/s, 1 Mbit/s and 2.5 Mbit/s, up to 2000 meters
Interbus-S	IEC/EN 61784-1 CPF 6, IEC61158 Type 8	Sensor bus, master-slave data transfer and common frame protocol, supports up to 4096 I/O points, speed 500 kbit/s, up to 400 meters
Fieldbus Messaging Specification (FMS)	IEC/EN 61158 Type 9	This is OSI layer 7 command set (Fieldbus Messaging Specification), does not specify any physical bus
Profinet	IEC/EN 61158 Type 10	Ethernet based Profibus protocol
Acutuator Sensor Interface (ASI)	IEC 62026-2:2000, EN 50295:1999	Binary sensor bus, up to 31 slaves, up to 124 binary operations, 5 ms, 100 meters
DeviceNet	ISO 11898, IEC 62026-3:2000, EN 50325-2:2000	Sensor bus, transport layer is based on CAN technology, 125-500 kbit/s, 500-100 meters
SDS	ISO 11898, IEC 62026-5:2000, EN 50325-3:2001	Sensor bus, transport layer is based on CAN technology, 125 kbit/s - 1 Mbit/s
CANopen	ISO 11898, EN 50325-4:2002	Up to 2032 objects, 125 kbit/s - 1 Mbit/s, up to 40 meters at full speed
LON-Works	Manufacturer specific system	Used mostly in building automation, 255 segments, 127 nodes per segment, maximum 32385 nodes in system
Modbus		MODBUS Protocol is a messaging structure that is widely used to establish master-slave communication between intelligent devices. The MODBUS protocol comes in 2 flavors: ASCII transmission mode and RTU transmission mode. MODBUS is traditionally implemented using RS232, RS422, or RS485 over a variety of media (fiber, radio, cellular, etc.).
Modbus TCP/IP		MODBUS Protocol is a messaging structure that is widely used to establish master-slave communication between intelligent devices. MODBUS TCP/IP uses TCP/IP and Ethernet to carry the MODBUS messaging structure.
Modbus RTPS	IEC PAS 62030:2004	On-going MODBUS standardizing work

Table 1: Fieldbus protocols. Source: Intel Corporation, 2009

Architecturally embedded PC industrial control systems can be split into the following subsystems:

- physical I/O modules
- fieldbus network
- interface card
- OPC client/server for connecting the interface card and the soft PLC
- the soft PLC package
- OPC client/server between the SoftPLC and the HMI
- the HMI

The key to unlocking the power of these new industrial controllers is the software. Software must provide the stability and reliability of the real-time OS to handle I/O and system timing, execution priorities, and to enable multi-loop execution. The software must also offer a breadth of control and analysis functions. This should include typical control functions such as digital logic and PID, and less common algorithms such as fuzzy logic and the capability to run model-based control. The software must also provide the analysis algorithms for machine vision and motion control, the capability to log data, and the network communications support to connect into back-end IT office systems, and to other systems on the factory floor.

"The software must also provide the analysis algorithms for machine vision and motion control, the capability to log data, and the network communications support to connect into back-end IT office systems, and to other systems on the factory floor."

In an embedded PC-based industrial control solution there are several software components interacting for determining the final behavior.

The Soft PLC

One of the core software components for the new class of controllers based on embedded PC technology is a soft PLC. A soft PLC is a runtime environment used for simulation of a PLC in an embedded PC. Using the soft PLC, part of the CPU is reserved for simulation of the PLC system for controlling a machine and the other part is designated to the operating system. The soft PLC operation is identical to normal PLC operation: it implements the control logic with the standard IEC 61131-3 programming syntax. It receives data from field devices, processes them through the logic implemented with an IEC 61131-3 compliant language, and at the end of the cycle it sends the outputs to the field devices and to the HMI.

"A soft PLC is a runtime environment used for simulation of a PLC in an embedded PC."

Key to accepting the design concept for PC-based industrial controller is verification of the real-time performance of the soft PLC application. The sometimes random behavior of PCs cannot be accepted for applications of industrial control. The most important feature of a controller is not only to perform the task in a certain time slot, but also the ability to perform the cyclic tasks always with the same time.

"The most important feature of a controller is not only to perform the task in a certain time slot, but also the ability to perform the cyclic tasks always with the same time."

OLE for Process Control

A key part of a PC-based system is the interface between the field devices and the soft PLC. Logically, the interface is between data transmitted by a fieldbus and a software tool running in the PC. This connection is obtained by means of an I/O interface that communicates with the fieldbus devices and transfers data to the PC by means of a software interface. One such interface is defined by the OPC Foundation: OLE for Process Control (OPC). This defines a set of standard inter-faces based upon Microsoft OLE/COM technology. The application of the OPC standard interface makes possible interoperability between automation/control applications, field systems/devices and business/office applications, typically an OLE for Process Control (OPC) server. The OPC server guarantees a standard data format that can be accessed and used by every OPC client, like the soft PLC itself. The soft PLC acts as an OPC client for reading and writing the data received from the field through the interface card that integrates an OPC server.

"The OPC server guarantees a standard data format that can be accessed and used by every OPC client, like the soft PLC itself."

The OPC client/server architecture is used not only for the interface between the field and the control layers, but also for the interface between the soft PLC and the HMI.

Considering the above described SW architecture, The data exchange between separate software packages plays a fundamental role in the PC-based solutions and cannot be neglected. Data conversion may become a task with longer time requirements that the control functions. A control system in PC environment is made of a set of cyclic processes, mainly the soft PLC and the OPC client/server cycles.

When analyzing the time behavior of a PC-based solution, it is mandatory to measure the regularity of each cycle time under different system conditions. In the overall system, we also have other processes that consume time, such as the fieldbus communication, the interface card conversion time, the I/O module response time.

"The OS guarantees the multitasking by setting the time scheduling of the running tasks. There are three different types of scheduling algorithms: timesharing, multi-programming, and real-time."

Operating System Considerations

Different scenarios are possible for these types of systems based on the choice of operating system (OS).

The embedded PC could run a general purpose multitasking OS where several applications run in time sharing with the soft PLC sharing the same computational resources (CPU and memory). The OS guarantees the multitasking by setting the time scheduling of the running tasks. There are three different types of scheduling algorithms: timesharing, multi-programming, and real-time.

In a timesharing scheduler, a precise time slot is assigned to each running task. The task must abandon the CPU before the assigned time expiration either voluntarily (the operation has finished) or by the action of the OS (hardware interrupt). The time-sharing scheduler is designed to execute several processes simultaneously, or better in rapidly successive time slots. The CPU communicates with all the peripherals of the embedded PC via one or more internal buses. Several processes manage these buses and must be scheduled by the OS together with the soft PLC and the other applications. The time assignment to each process depends on its priority that can be only partially defined by the user. For this reason, it is not easy to determine which processes are served by the OS in a given time slot. In the default conditions all the processes have the same priority level. This means they have the same CPU time at their disposal. Therefore, a general purpose, multitasking OS is intrinsically not deterministic in running concurrent applications. The running time of a control application (like a soft PLC) cannot be guaranteed with these operating systems. This is a theoretical limit that cannot be overcome unless an RTOS is used.

"The CPU communicates with all the peripherals of the embedded PC via one or more internal buses. Several processes manage these buses and must be scheduled by the OS together with the soft PLC and the other applications."

The real-time operating system performances can be divided into two different categories according to the effects on the system of the missing of a deadline: *hard real-time* and *soft real-time*.

In a *hard real-time* behavior, a specific action must be performed at a given time that cannot be missed unless losing the performance. A RTOS for hard real-time applications operates at low level, with a close interaction with the hardware platform.

These RTOSs are normally based on a priority driven preemptive scheduler that allocate a fixed bandwidth of the processor capacity to the real-time processes or threads.

For less critical applications (*soft real-time*) it is possible to use conventional PCs running a real-time extension of a general purpose multitasking OS. The real-time applications are scheduled by the real-time extension that guarantees an almost deterministic behavior. In such application, all the wanted real-time applications must run the real-time environment.

"For less critical applications (soft real-time) it is possible to use conventional PCs running a real-time extension of a general purpose multitasking OS."

A further possibility is simply running the real-time applications in a non-RTOS verifying that the system performances are adequate for reaching the desired results. In other words, we can run the soft PLC in a normal Windows* or Linux* PC, accepting that the PC response is driven by a nondeterministic operating system, provided that the overall performances are anyway sufficient for ensuring the control functions effectiveness. Such an approach means that the PC environment is performing so well that the random variations of its throughput remains well within the acceptable limits for a given control application. This process is in progress for the soft PLCs that will run more and more in conventional PCs. For this reason, it is mandatory to define a benchmark for evaluating the performances of such PC-based systems. The benchmark should include:

- The definition of the PC environment where the control applications run
- The tools for measuring the time behavior of the system in terms of response time to events for interrupt based functions, and jitter for cyclic functions

Referencing requirements and concepts presented in this section, the hardware and software requirements are summarized in Figure 3.

"Such an approach means that the PC environment is performing so well that the random variations of its throughput remains well within the acceptable limits for a given control application."

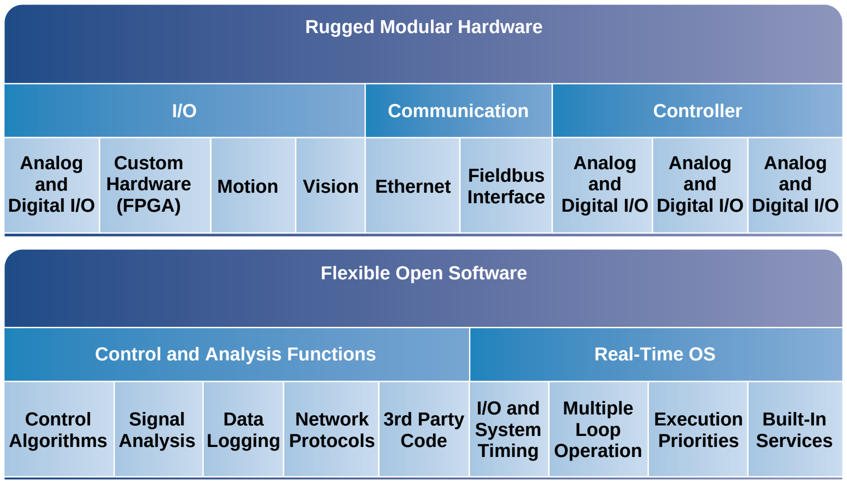


Figure 3: Summary of rugged modular hardware.

"By incorporating one of the industrial temperatures versions available in the Intel Atom processor family, which has the thermal footprint that enables fanless systems to be developed, the design is well-suited to the harsh environment found in many industrial settings."

"The Intel System Controller Hub measures 22 mm x 22 mm and provides integrated graphics, a digital-audio interface, a main-memory interface, and numerous peripheral I/O interfaces."

A Design Approach Based on the Intel® Atom™ Processor

At the hardware level a high level block diagram for a modular PLC based on the Intel Atom processor is shown in Figure 4. By incorporating one of the industrial temperatures versions available in the Intel Atom processor family, which has the thermal footprint that enables fanless systems to be developed, the design is well-suited to the harsh environment found in many industrial settings.

The combination of the Intel Atom processor paired with the Intel System Controller Hub provides the majority of the interfacing required for the industrial control application. The Intel System Controller Hub measures 22 mm x 22 mm and provides integrated graphics, a digital-audio interface, a main-memory interface, and numerous peripheral I/O interfaces. It supports single-channel DDR2 memory. Front side bus (FSB) speed is 400 MHz or 533 MHz. Maximum memory is 2 GB. There are eight USB 2.0 host ports and one USB 2.0 client port. The parallel ATA interface supports two disk drives. System designers will add DRAM and physical-layer (PHY) chips for the features they wish to support. Additionally there are three fabrics: one for memory traffic, a second for I/O traffic, and a third message-based network that handles almost everything else. To manage these fabrics, the north bridge integrates an 8051 eight-bit microcontroller. The integrated 2D/3D graphics engine can drive a 1,366-pixel x 768-pixel display in 24-bit color. The integrated video engine can decode 1080p HDTV streams at 30 frames per second, using only 150 mW in the process. It supports MPEG1, MPEG2, MPEG4, and H.264 video and is compatible with Microsoft* DirectX* 9 and DirectX 10 graphics.

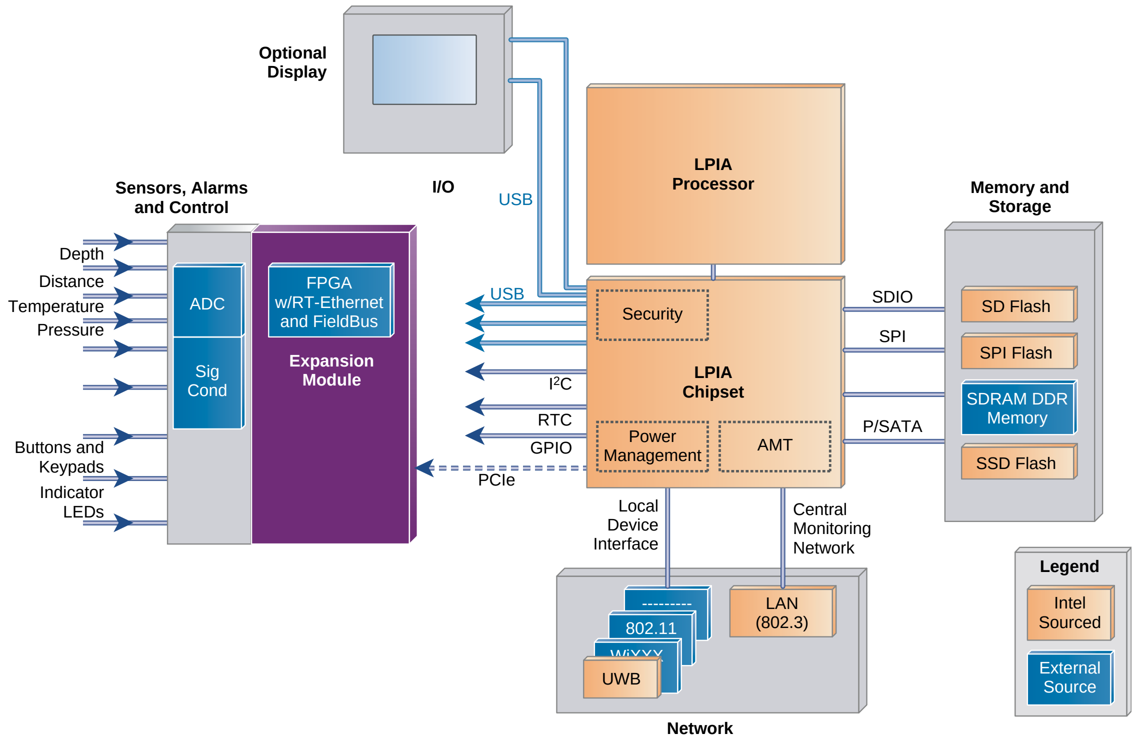


Figure 4: Typical Intel® Atom™ processor platform.

To enable all the fieldbus and real-time Ethernet protocols support is typically realized either with the OEM's own ASIC, or by using an FPGA. The FPGA in this case acts as an intelligent peripheral extender to this platform to add the industrial I/O not contained in the base Intel System Controller Hub.

Functionally the FPGA will interface to the Intel System Controller Hub through a single lane PCI Express interface. The FPGA is usually architected in such a way that it can be easily extended (or modified) for additional peripherals.

The interface allows exchanging the data between CPU and FPGA with very short latency times, typically in the range of microseconds.

Software running on Intel Atom processors will implement the protocol processing for fieldbus and real-time Ethernet. A number of independent software vendors deliver software stacks all delivering support for the IA-32 instruction set.

This design approach maximizes how open modular systems can be built for industrial automation. By incorporating the CPU and chipset on to a module and the industrial fieldbus and real-time Ethernet I/O on an FPGA, this solution easily scales to incorporate new CPU modules and to incorporate variances and new standards associated with industrial I/O.

"This design approach maximizes how open modular systems can be built for industrial automation."

The advantages with this design include:

- Maximum flexibility to incorporate support for industrial I/O.
- Low power that enables high performance fanless designs.
- PCI Express for high performance I/O.
- Extreme low power that enables rugged solutions for harsh environments.
- Integrated graphics for embedded HMI.
- Intel Hyper-Threading Technology (Intel HT Technology) that enhances real-time performance.
- Very slim housing and possible small form factor.
- Power over Ethernet (including TFT-Display).
- Miscellaneous functions integrated into one peripheral FPGA (LPC, FWH-I/F, keyboard touch controller, bus-interface like Ethernet, CAN, and so on).
- Easy adoption of various industrial buses I/F with standard interconnect modules.

Conclusion

Today traditional industrial control using proprietary architectures has been superseded by new PC-based control systems that are generically referred to as *open control*. Open control gives the engineer the freedom of choice of the hardware platform, the operating system, and the software architecture. Instead of fitting an application into a predefined architecture, the designer has the choice of hardware and software components, to exactly meet the requirements of the design while drastically reducing costs and time to market. Open control provides standardization. Open control systems can be programmed using any of the IEC 61131 standard languages. Commonly available processors such as the Intel Architecture family can be used. Commonly available solutions can be provided by manufacturers or a customer specific design can be created by selecting the appropriate component.

The Intel Atom processor is designed in the Intel architecture tradition of providing general purpose computing platforms. The power of the customer application is unlocked by the versatility and power of the software applications that may be designed on the platform. The Intel Atom processor is fully compliant with the IA-32 architecture, enabling designers to use the vast software ecosystem to ensure fast time to market for new designs.

The advantage to end users includes the ability to leverage a common well known architecture from high end industrial PCs right down to low level intelligent field devices and PLCs. Developing on a common platform architecture also simplifies the convergence challenge between corporate IT and automation systems. The ability to develop real-time systems using open standards on scaleable platforms can bring significant benefits to developers in terms of engineering reuse as well as bringing products to market quickly and efficiently.

In conclusion, designing with the Intel Atom processor, brings all of the benefits traditionally associated with Intel architecture designs to the low power or "real" embedded market. If you interested in learning more about Intel's embedded product family, please check out ***http://rethink.intel.com***.

Author Biography

Ian Gilvarry: Ian Gilvarry is currently the worldwide industrial automation marketing manager within the Intel Embedded and Communications Group. He leads the market development activities to define and position Intel platforms strategically to drive for new use cases and applications in the industrial segment. He has been with Intel since 2000. Prior to assuming his current responsibilities in 2007, he previously held roles in product marketing and business development within Intel's Network Processor Divison. His e-mail address is ian.gilvarry at intel.com.

Copyright

LOW POWER INTEL® ARCHITECTURE PLATFORM FOR IN-VEHICLE INFOTAINMENT

Contributors

Suresh Marisetty
Intel Corporation

Durgesh Srivastava
Intel Corporation

Joel Hoffmann
Intel Corporation

Brad Starks
Intel Corporation

Index Words

Automotive
Infotainment
IVI
Intel® Atom™ Processor
Moblin
Head Unit
SoC
Embedded

Abstract

Automotive manufacturers today face a tremendous challenge in trying to bridge the historically long development cycles of a vehicle to the ever-changing I/O and multimedia demands of the consumer. The main function of the car's entertainment system or the head unit is enabling a variety of functions like navigation, radio, DVD players, climate control, Bluetooth*, and so on. Further, with the promise of the connected car becoming a reality enabled through broad deployment of multimedia-capable mobile wireless technologies, the automotive industry sees an opportunity to deliver new value-added services to the consumer. However, with today's proprietary head-unit solutions they have limited ability to offer such services. A cost-effective solution to address this need is to use standards-based platform technologies that can take advantage of the huge ecosystem built around PC standards and consumer-oriented applications and services. The platforms based on Intel® architecture have been evolving in tandem with various I/O and multimedia technologies and have been adopting these technologies in a seamless way. An in vehicle infotainment (IVI) platform is an architecture based on these building blocks, but with optimizations for the automotive environment.

This article presents the architecture of this platform for the IVI market segment powered by the Intel® Atom™ processor family of low power embedded processors and standards-based platform hardware and software ecosystem. An overview of the key technology blocks that make up the Intel-based IVI platform is presented, followed by a brief description of the challenges faced in optimization and incorporating these into the Intel-based IVI platform. In addition, the opportunities presented by the Intel-based IVI platform for future usage models are also highlighted. The challenges and opportunities are presented both from a hardware and software perspective to meet the power, performance, size, differentiation, and other needs of the automotive environment and usage models.

Introduction

We will start with the architecture of an IVI platform with a brief introduction to the platform stack and delve into each of the stack components, both from an hardware and software perspective; we will also examine their interdependencies. The theme of discussion for each of these technology areas is as follows:

- Overview with usage models
- Bullet Body 10/12. Praesent feugiat.
  - By car OEM and end customers
- Challenges that:
  - Were overcome in optimizing and enabling various technology blocks for an Intel-based IVI platform
  - Remain to be addressed now and in the future by Intel Corporation, the car OEM, IHV/ISV/OSV, and academia for various usage models
- Opportunities that present themselves to:
  - Car OEM for product differentiation
  - Third party software and hardware vendors to enable new markets—ecosystem enabling
  - Academia for identifying areas of advanced research and technology development

An in-depth discussion follows covering the following technology building blocks for blocks for an Intel-based IVI platform:

- Intel-based IVI platform overview
- Usage models and software environments
- System on a Chip (SoC) Architectures for an Intel-based IVI platform
- Platform boot solution and latencies
- Multimedia (graphics/video/display/audio)
- Generic and automotive-specific I/O fabric
- Intel technologies with focus on Intel® Virtualization Technology (Intel® VT)
- Manageability and security
- Seamless connectivity
- Power management

Intel-Based IVI Platform Overview

The framework or stack for an Intel-based IVI platform consists consists of software and hardware components with well defined interfaces between them to boot an operating system (OS) supporting the key application functionality of an automotive head-unit, as shown in Figure 1.

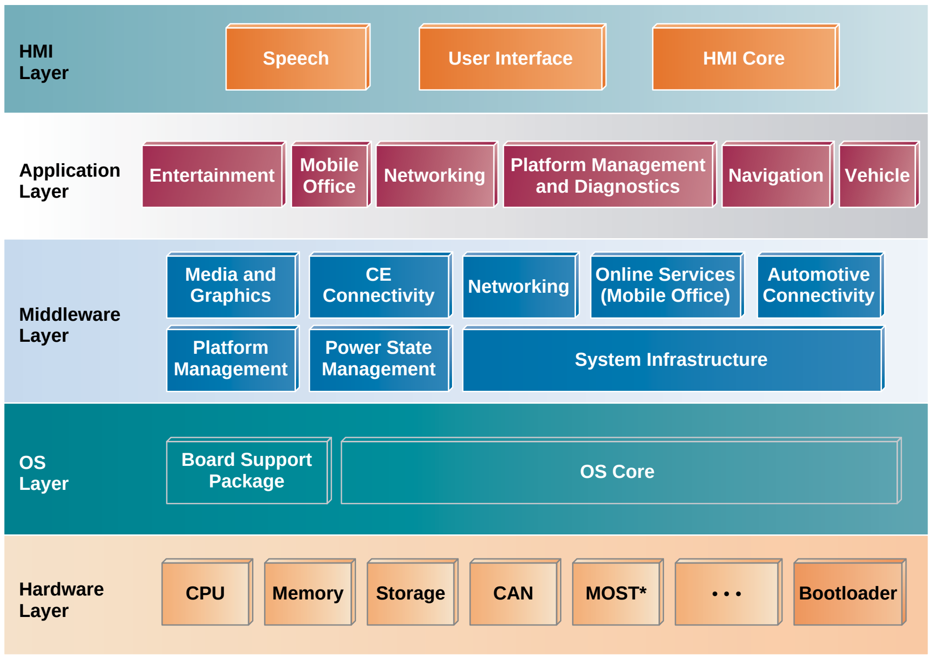


Figure 1: Stack components for an Intel-based IVI platform

The following is the brief description of each of the components of the stack:

- Hardware Layer: The core part of the hardware layer is comprised of Intel® Atom™ processor with all the necessary hardware and firmware to boot any off-the-shelf or embedded OS. This layer is further complemented with the inclusion of a set of automotive OEM-specific I/O devices, such as MOST*/CAN buses, connected through an industry standard I/O fabric, such as PCI Express*. The use of the Intel Atom processor–based SoC solution facilitates the inclusion of many other extended inputs/outputs available for the Intel® architecture platform, without affecting the core platform functions. This allows the car manufacturers to be able to provide end solutions with many options with little to no additional cost or software development effort, facilitating product differentiation. A typical Intel-based IVI platform configuration built around the Intel Atom processor is as shown in the Table 1.

Hardware Function	Description
Intel® Atom™ Processor	CPU supporting frequencies for sufficient integer and floating point performance Supports Inte!® Hyper-Threading Technology (Intel® HT Technology), Intel® Virtualization Technology (Intel® VT)
Memory Controller	Support low cost 1-2 MB DIMM/UDIMM like DDR2-533 and DDR2-667
Video Decoder	Full hardware decode pipeline for MPEG2, MPEG4, VC1, WMV9, H.264 (main and high profile level 4.1), DivX*
Graphics Engine	Performance: Fill rate of at least 400 megapixels/sec and 3DMark*05 score of 120
HD Audio	High definition audio based on the Intel® High Definition Audio (Intel® HD Audio) specification or its equivalent (http://www.Intel.com/standards/hdaudio/)
Display	Dual simultaneous display hardware support like LVDS/DVI/dRGB/TV Out WXGA 1280x800 18 bpp; XGA 1024x768 24 bpp
I/O Fabric	Gen1 PCI Express* x1 Expansion slots and USB 2.0
Compatibility I/O Block	PC compatibility core system block components like PIC, RTC, Timer, GPIO, Power Management, Firmware Hub Interface, and LPC, to allow shrink-wrap OS boot
Car OEM Automotive Specific I/O	MOST*, CAN, SPI, Bluetooth*, UART, SDIO, Ethernet, Radio Tuner, Video Capture, GPS, GRYO, etc.

Table 1: Typical Intel-based IVI platform configuration

- OS Layer: Given the platform's Intel architecture compatibility lineage, a range of operating systems are enabled, including embedded real-time OS (RTOS) and commercial off-the-shelf operating systems that run on a standard PC platform. This layer also includes drivers that are specific to automotive I/O.
- Middleware Layer: The Intel-based IVI platform middleware can include a rich set of components and interfaces to realize all functional areas of the application layer, such as Bluetooth* with support for various profiles and CAN/MOST protocol stacks.
- Application Layer: The applications include the ones designed into many mobile Internet devices (MIDs) or handheld devices like Web browsers, calendar, Bluetooth phone, vehicle management functionalities, multimedia entertainment system, and so on. This layer can provide a rich set of applications and many customization options that conform to Intel architecture binary format.
- HMI Layer: The Human Machine Interface (HMI) is the central interface to the user of the IVI system. The HMI has control of the display of the HMI Head Unit and has the responsibility to process and react to all user inputs coming into the system, such as speech recognition and touch screen input.

In regards to the overall Intel-based IVI platform stack itself, the key challenges are the integration or seamless porting of various applications and middleware to the automotive-specific user interface standards. The ecosystem of this software includes independent software, OS vendors (ISVs/OSVs) or the Linux* Open Source community.

"The key challenges are the integration or seamless porting of various applications and middleware to the automotive-specific user interface standards."

The automotive environment requires hardware components that are highly reliable. Intel is now offering the Intel Atom processors with industrial temperature options (minus 40° to 85° C). For further platform differentiation beyond the solution from Intel, the car OEM may be limited to picking third-party vendor hardware IP blocks that meet the reliability requirements.

ISVs and OSVs can provide powerful user interface (HMI) tools or development kits, to enable easy OEM HMI customization across their product line. Third-party hardware vendors can provide various automotive-specific I/O solutions to allow easy car OEM product differentiation. In addition, it is a new opportunity for application developers to port Intel architecture applications to the Intel-based IVI platform ecosystem and maximize reuse and applicability of their software across a number of Intel architecture platforms.

Usage Model

In-vehicle infotainment platforms that are well connected, blending embedded and vehicle-independent services and content with bidirectional communication capabilities to the outside world, do not exist today. While a range of nomadic device services and proprietary embedded vehicle systems can be found in some segments, these discrete services are not operating within a comprehensive OEM defined environment. Figure 2 outlines some of the challenges.

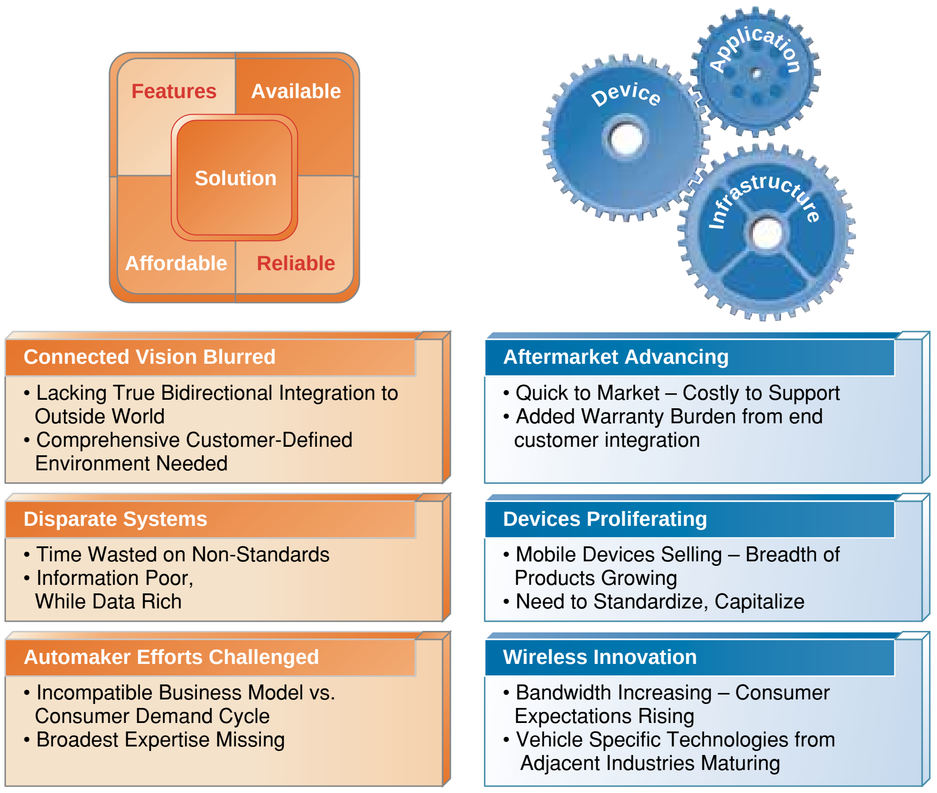

Features
Available
Solution
Affordable
Reliable
Device
Application
Infrastructure
Connected Vision Blurred
• Lacking True Bidirectional Integration to Outside World
• Comprehensive Customer-Defined Environment Needed
Disparate Systems
• Time Wasted on Non-Standards
• Information Poor, While Data Rich
Automaker Efforts Challenged
• Incompatible Business Model vs. Consumer Demand Cycle
• Broadest Expertise Missing
Aftermarket Advancing
• Quick to Market – Costly to Support
• Added Warranty Burden from end customer integration
Devices Proliferating
• Mobile Devices Selling – Breadth of Products Growing
• Need to Standardize, Capitalize
Wireless Innovation
• Bandwidth Increasing – Consumer Expectations Rising
• Vehicle Specific Technologies from Adjacent Industries Maturing

Figure 2: In-Vehicle Infotainment (IVI) platform use case challenges.

Global automakers have come to realize that customers desire connectivity to content and services that are not possible to achieve with existing business models and currently used embedded systems. In addition, automakers could leverage the expertise, knowledge, or business structure from other embedded platforms to provide the hardware, applications, data or communications conduits to support the breadth of needs.

"Global automakers have come to realize that customers desire connectivity to content and services that are not possible to achieve with existing business models and currently used embedded systems."

There is significant momentum within the industry and major automakers are exploring ways to deliver content and services desired by customers to the vehicle. The exploration is primarily driven by the advancements and maturity of the communication technologies and protocols like cellular, satellite, Wi-Fi*/WiMAX*, and DSRC.

Although every automaker would like to provide content and services, they incur huge risks being first to market if other automakers do not participate. This creates the dichotomy of balancing confidential efforts with the need for industry-wide, cross-vehicle, and cross-brand solutions to capture the interest of large service providers.

"Although every automaker would like to provide content and services, they incur huge risks being first to market if other automakers do not participate."

Since automakers historically have engaged tier-1 suppliers to develop, integrate, and deliver components, the value chain was straightforward and limited in scope. With the need to provide a means for customers to communicate externally to the vehicle for information and entertainment systems, automakers now must become directly familiar with all of the stakeholder domains that impact this larger ecosystem.

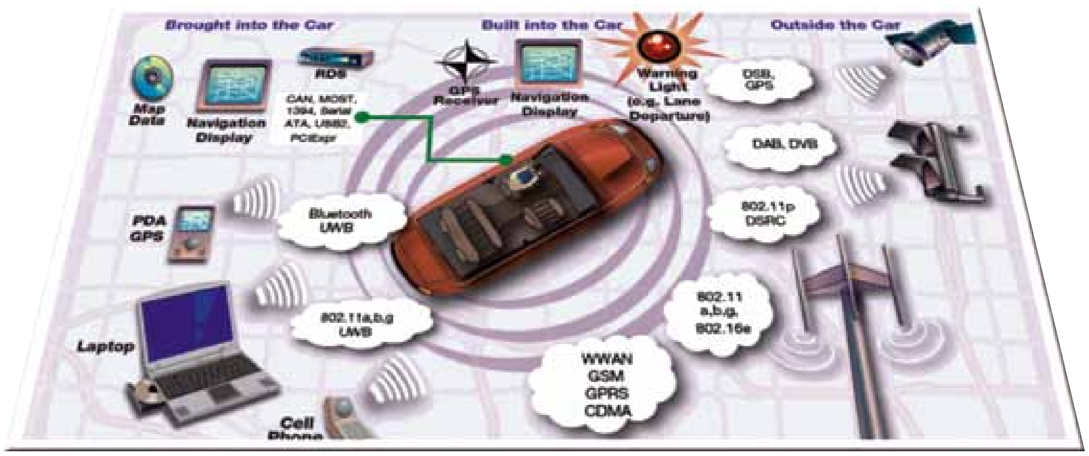


Figure 3: In-Vehicle Infotainment platform business challenges and opportunities.

"The industry segment alignment needs to be developed and implemented by the key providers to distribute the cost of developing and marketing innovative connected services."

There are a significant number of commodity software and hardware components that can be leveraged, leaving the OEM to focus on adding value. In order to capitalize on this potential, a strong partnership between key providers of devices, infrastructure, and applications will be essential to the acceptance of infotainment services on a broad scale. Meanwhile the solution needs to support the traditional requirements for automakers: availability, reliability, affordability, and desirable features for consumers. Therefore, the industry segment alignment needs to be developed and implemented by the key providers to distribute the cost of developing and marketing innovative connected services. As consumer and business awareness grows, more services can be offered at prices acceptable to the market.

In-Vehicle Infotainment Operating Systems

An Intel-based IVI platform can run many of the commercial generic operating systems like Linux, Microsoft* Windows* XP Embedded, and real-time operating systems like QNX*, Windows CE, VxWorks*, and Embedded Linux. Most of the operating systems that run on an Intel architecture platform will run unchanged, offering a wide choice to the car OEM.

"The key challenges that the car OEM and the automotive suppliers face is the choice of the OS and the ecosystem built around each."

Some embedded and real-time operating systems are optimized for the automotive environment with attributes of smaller OS footprints, sub-second boot times, and optimization for power and performance. Key examples of such operating systems are QNX Neutrino*, Wind River* Linux Platform for Infotainment, Moblin* IVI (Moblin.org), Microsoft Auto, or variants of Linux from various ISVs and tier-1 customers.

OS vendors are faced with new challenges of porting the generic OS to automotive user interfaces like touch screen and voice commands to assure safer driving experiences. In addition, traditional shrink-wrap operating systems require a PC-like BIOS with high boot latencies and make them not very desirable. The key challenges that the car OEM and the automotive suppliers face is the choice of the OS and the ecosystem built around each. Too much choice is a good thing, but at the same time it is hard to settle on one over the other, as each choice has its own compelling advantages. Due to the flexibility of IVI platform, a customer may demand for an OS/application suite other than what the car OEM wants to bundle, leaving the OEM in a dilemma.

"Making shrink-wrap operating systems to boot with IVI latencies is a challenging area and requires some innovation both by the BIOS vendors and OS vendors."

The OS vendors can help develop seamless plug-in interfaces to enable their own or third-party user interfaces, while leveraging their underlying core OS features. Making shrink-wrap operating systems to boot with IVI latencies is a challenging area and requires some innovation both by the BIOS vendors and OS vendors. The variety of operating system choices is opening up new opportunities. One such opportunity to meet the customer demands is the use of Intel Virtualization Technology offered by the Atom processor, allowing not only the car OEM but also the end customer to simultaneously run multiple operating systems and benefit from the ecosystem built around each of the Intel architecture platform operating systems. We cover more on this in the subsequent section on Intel Virtualization Technology.

System on a Chip Architecture

The first generation Intel-based IVI platform is based on the Intel Atom processor with extended functions. The processor and its companion I/O chip were repackaged to meet the extended temperature and low defects per million (DPM) requirements for automotive and embedded customers.

The Intel-based IVI platform addresses the challenges of getting to market quickly and easily differentiating car OEM products by allowing reuse from the ecosystem.

- Includes all the legacy support that is required to run an OS like: connectivity to flash for boot firmware, the 8259 interrupt controller, I/O APIC, SMBus, GPIO, power management, real-time clock, and timers.
- Includes all Intel-proprietary hardware blocks like: graphics/video, Intel® High Definition Audio module, and so on.
- Includes a scalable industry-standard PCI Express (PCIe) interconnect
- Allows third-party vendors to focus on building many different flavors of I/O hubs using standard "jelly-beans" (the standard input/output functions) from external companies.
- The Intel-based IVI platform is architected in such a way that the functionality in the SoC partition will be a common denominator across all OEMs, including the associated platform firmware, also known as the BIOS or boot loader.
- All automotive-specific I/O functionality follows the PCIe add-on card model without requiring any platform changes but a set of device drivers for the target OS. Alternatively, the same software transparency can be achieved through the USB and SDIO interfaces plug-in device model as well.

We see more opportunities than challenges in this context. The SoC being designed for the Intel-based IVI platform is flexible enough that a car OEM can enable multiple variations of products to cater to different end-user needs and cost structures and not require major reworking of software.

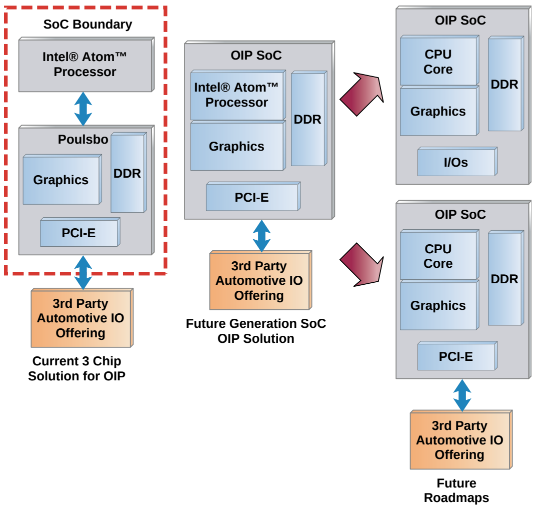


Figure 4: Intel-based IVI platform hardware architecture and directions.

Platform Boot Solution

Users expect an instant power-on experience, similar to that of most consumer appliances like TV. To meet the same expectation, one of the key requirements of the Intel-based IVI platform is sub-second cold boot times to help facilitate this user experience when the ignition is turned on. The typical boot latencies are as illustrated in Figure 5.

For an Intel-based IVI platform, multiple types of OS boot loaders shall be supported for various operating systems as follows:

- ACPI-compliant UEFI BIOS with an EFI OS boot loader (such as eLilo). This is typically used with after-market products that may run embedded versions of a shrink-wrap OS such as Standard Embedded Linux or Windows XPe that requires PC compatibility and is readily available from the by BIOS vendors or original device manufacturers (ODMs). This solution provides the most flexibility for seamless addition of I/O, but at the expense of higher boot latencies. Many of the initialization sequences in the boot path are optimized to reduce the latencies significantly in the order of 5-10 seconds.

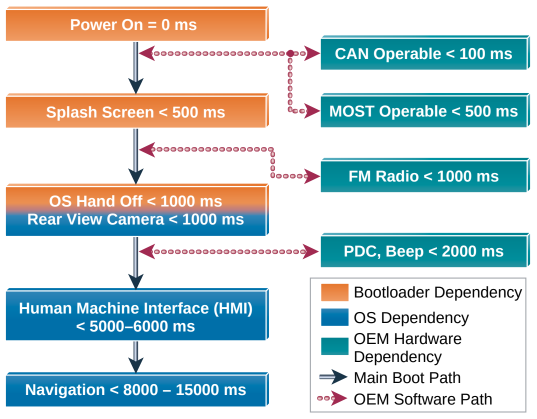


Figure 5: Intel-based IVI platform boot latencies.

"Getting this HMI active latency down to 5–6 seconds with an active splash screen in <500 ms was a big challenge."

- Embedded OS boot loader. This is the highly optimized solution for an Intel-based IVI platform for low boot the functions on the SoC. This solution is meant to work with an OS that does not rely on the PC BIOS compatibility, such as an embedded OS and some variants of Linux. The boot latencies are achieved at the expense of giving up flexibility of seamless support for new I/O hardware outside the fixed function SoC, during the OS pre-launch environment. However, any such I/O will still be enabled in the OS through appropriate device driver software.

Traditional platforms typically have boot latencies of about 10–40 seconds before the HMI is activated. Getting this HMI active latency down to 5–6 seconds with an active splash screen in <500 ms was a big challenge. To reduce time to market and product development cost, it was highly desired to make the same boot firmware and OS solution scale across different car OEM platforms with varying topology, but based on the same SoC core. Many optimizations were done, to both the BIOS and boot loader solutions from the norm, to fit into an Intel-based IVI platform, the key being the reordering and early initialization of user-visible I/O like display activation, initial program load (IPL) boot menus, enabling processor cache usage at boot as high speed RAM, and so on.

"There are opportunities for the OS vendors to come up with innovative optimizations within the OS boot flows, such as replacing graphics hardware initialization in firm-ware with early OS initialization for the same."

The need for a boot solution that is low cost, has a smaller footprint, offers low boot latencies, and is platform-agnostic is an exciting opportunity for ISVs and OSVs. This also creates opportunities for car OEMs to provide creative solutions with their own IP, to make their products competitive and unique. In addition, there is an opportunity for the device vendors to provide hardware IP that are self-initializing, thereby relieving the boot software from doing the same and giving back some time to improve latencies, such as initializing and activating the CAN interface. The challenge that remains to be addressed is a single solution that can boot both shrink-wrap OS requiring PC compatibility and embedded OS, but with the flexibility of allowing platform differentiation and low boot latencies. There are opportunities for the OS vendors to come up with innovative optimizations within the OS boot flows, such as replacing graphics hardware initialization in firmware with early OS initialization for the same.

Graphics

The most compelling advantage to using an Intel architecture platform as the basis for IVI is the ability to repurpose familiar desktop/mobile applications without substantial development. An Intel-based IVI platform includes an Intel® Graphics Media Accelerator 500 series graphics controller with supported drivers for Linux, Windows XP and Windows CE operating systems, with frequencies of 200 – 400 MHz, fill rates of 400-800 Mpixels/sec and 3DMark'05 of 130 and up. High-level industry standard APIs such as OpenGL*, OpenGL ES, and D3D are available to support a wide range of application development opportunities. By using a graphics device with full support for desktop operating systems and industry standard APIs, an Intel-based IVI platform makes it easy to develop custom human machine interfaces and to integrate applications familiar to the Intel® architecture system user.

Two significant challenges are faced when using shrink-wrap operating systems on an Intel-based IVI platform. First, the user expects to be presented with an appealing graphics display as early as possible after turning the key and second, the user expects the familiar applications to perform with reasonable speed while the hardware is optimized for low power consumption.

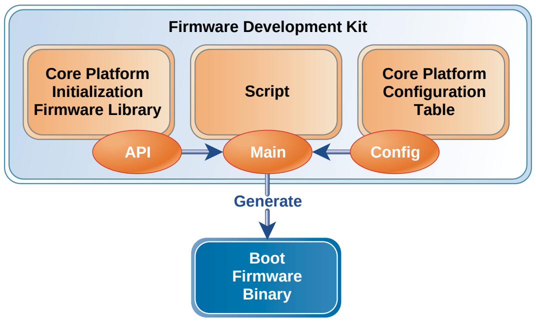

Firmware Development Kit
Core Platform Initialization Firmware Library
Script
Core Platform Configuration Table
API
Main
Config
Generate
Boot Firmware Binary

Figure 6: Firmware architecture model.

Boot time display has been addressed on the Intel-based IVI platform using three technologies to ensure an immediate and stable graphics display.

- Splash Screen: The platform boot firmware is enabled with a tuned graphics driver with the ability to present a static splash screen or video from a backup camera within 400 ms from power on.
- Seamless Display: A technology whereby the splash screen image and display mode are retained by the graphics driver during initialization.
- Operating System Suppression: Shrink-wrap operating systems such as Linux should not be allowed to display during the boot process. Any intermediate mode changes or partially rendered graphics are avoided. The Intel-based IVI platform OS drivers instead maintain the splash screen until the full human machine interface is rendered and immediately does a flip from the splash screen to the final intended interface.

"The user expects to be presented with an appealing graphics display as early as possible after turning the key."

These three technologies combined provide an immediate and appealing graphics display that remains viewable without glitches from mode setting until the full HMI is available. Never is the user presented with a partial or unstable display as is commonplace in desktop or mobile boot processes.

"These three technologies combined provide an immediate and appealing graphics display that remains viewable without glitches from mode setting until the full HMI is available."

The second challenge is that the Intel-based IVI platform hardware is optimized for low power consumption. The graphics capabilities must be chosen such that the performance level is an appropriate match for the applications used. This problem is exacerbated when using desktop applications that may be poorly tuned for low power systems. This challenge is overcome by taking special care to choose applications that are well suited for the hardware platform. Combining applications from software stacks such as Moblin, Windows CE, or Neutrino will help to minimize the investment cost for the OEM, as those applications have been optimized for low power platforms by design.

The human machine interface options that are made available by using a common set of graphics APIs is greatly increased thereby giving each OEM significant room for differentiation. The application stack can be developed and debugged in a full desktop environment with full access to familiar debuggers and tool chains and then needs only to be tuned for the Intel-based IVI platform. This development methodology reduces time to market for the OEM. Additionally, by using industry standard APIs, the application software can be reused and extended to future platforms with minimal rework.

Display

The Intel-based IVI platform hardware comes equipped with a flexible display controller capable of supporting a wide range of display devices. The display controller supports two fully independent display pipelines. One display pipeline connects to LVDS displays ranging from VGA-sized 640x480 to full high-definition video at 1080p. The second display pipeline uses the sDVO protocol to connect to an independent display encoder; sDVO display encoders are available supporting LVDS, VGA, DVI, HDMI, or analog TV. The capabilities of the Intel-based IVI platform hardware platform are a good match for the in-car use models using both front- and rear-seat entertainment.

"The application stack can be developed and debugged in a full desktop environment with full access to familiar debuggers and tool chains."

"This feature will allow the platform to decode high quality video content such as Blu-Ray for rear-seat entertainment without exhausting the CPU resources on the platform."

Video

Intel-based IVI platform graphics drivers will have support for hardware acceleration of video decode for the following standards: MPEG2, MPEG4, H.264, VC1, through a set of OS-specific APIs. This feature will allow the platform to decode high quality video content such as Blu-Ray for rear-seat entertainment without exhausting the CPU resources on the platform. In this manner the platform will enable rear-seat entertainment without sacrificing the usability of the in-dash HMI or navigation applications.

Audio

The Intel High Definition Audio Specification describes a new audio hardware architecture that has been developed as the successor to the Intel AC '97 codec and controller specification. The audio system will also support a distributed digital audio infrastructure such as Media Oriented Systems Transport (MOST).

The Intel-based IVI platform HD Audio architecture provides a uniform programming interface for digital audio controllers, enabled through a standard-based register set that is uniform across all implementations. The HD audio is one of the technologies that is a simple adoption from other Intel architecture-based platforms and will take advantage of the huge ecosystem built around it, such as software and codecs.

The HD audio supports up to eight simultaneous audio channels with each channel in turn supporting a variety of audio codecs, such as Dolby* Digital 5.1. This architecture fits very naturally because of the automotive usage model of having multiple audio sources and synchronizes well with various combinations of audio mixing and separation options such as streaming audio over to the front and rear passenger with one supporting a hands-free phone/headset, while the rear entertainment system is playing back a Blu-Ray audio track coupled with the dual independent displays.

Having the capability to support 8 audio channels on the Intel-based IVI platform creates an opportunity for the OEM to harness the richness of this environment and innovate with new products. We can only imagine the possibilities of different usage models with multiple audio stream sources, mixing, and destinations throughout the automobile.

Generic I/O

Generic I/O typically refers to the I/O fabric and devices that are required for running a general purpose OS. This includes a set of devices and I/O fabric falling into the following categories: memory controller, video decoder, graphics engine, HD audio bus interface; display; I/O fabric, and the compatibility I/O block as shown in Table 1. In the Intel-based IVI platform architecture, these are integrated as part of the SoC silicon. Various implementations of the Intel-based IVI platform may move some of this functionality in and out of the SoC for ease of design and to allow more flexibility. However, the software model will be agnostic to this physical partitioning and is the key benefit of the Intel-based IVI platform architecture.

"The HD audio is one of the technologies that is a simple adoption from the Intel architecture-based platforms and will take advantage of the huge ecosystem built around it, such as software and codecs."

Automotive I/O

This is the key variable component of the Intel-based IVI platform architecture, which is specific to each car OEM. The Intel-based IVI platform allows this extended functionality through an add-on PCIe device model, which may be attached to a daughter card type module. The same OEM may have different versions of this add-on module, for their low-end to value-line products. As outlined in this article, enabling of any of the I/O functionality does not require SoC hardware or firmware changes, but an incremental inclusion of device drivers for the extended PCIe device functions on the target OS. This hardware and software model lends itself to a highly scalable platform. The current generation Intel-based IVI platform includes a standard set of core automotive I/O like MOST, CAN, SPI, Bluetooth, SDIO, Ethernet, radio tuner, video capture for cameras, GPS/GRYO, digital TV, and the Apple* iPod interface.

"Enabling of any of the I/O functionality does not require SoC hardware or firmware changes, but an incremental inclusion of device drivers for the extended PCIe device functions on the target OS."

One of the key technology areas that is evolving is that of the connected vehicle with many compelling applications through WiFi, Wi-Max/3G/4G connectivity. Some of the interesting applications that might be developed and enabled are: social networking, services like local search (POI), real-time traffic, imagery collections, Web search, widgets, and so on.

Intel® Technologies

Intel includes various technologies in its products, but the relevant ones for the Intel-based IVI platform are covered here from the IVI usage perspective. Each of the Intel® platform solutions has varying levels of technology support, due to independencies on various platform hardware component features. While we cover these technologies and their applicability to the Intel-based IVI platform, one must reference each of the product SKU specifications for the available support.

Virtualization

Virtualization creates a level of abstraction between physical hardware and the operating system. The abstraction layer, referred to as the hypervisor, executes the OS as a guest within a virtual machine (VM) environment and virtualizes or emulates the platform resources (CPU, memory, and I/O) to the guest OS. Multiple guest operating systems can be supported by the hypervisor, each encapsulated within its own VM, executing unmodified software stacks with user applications (fully virtualized) or modified to run in conjunction with the hypervisor (para-virtualized).

Intel Virtualization Technology (Intel VT) applicable to the Intel-based IVI platform is based on two different components, namely Intel® Virtualization Technology (Intel® VT) for IA-32, Intel® 64 and Intel® Architecture (Intel® VT-x) support on the processor and Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d) support in the controller hub. Intel VT-x constitutes a set of virtual-machine extensions (VMXs) that support virtualization of the processor hardware. Intel VT-d provides IO device assignment to the VM with hardware-assisted DMA and interrupt remapping from the I/O devices. For complete details of Intel VT, visit ***http://developer.intel.com.***

Each of the Intel platform solutions has a varying set of hardware capabilities for virtualization. The key virtualization usage models for the Intel-based IVI platform that a car OEM can use with the appropriate built-in Intel VT hardware are described in the following paragraphs.

Consolidation: This usage model is the concept of combining multiple applications, each of them executing on a separate hardware platform, onto a single hardware platform without modification of the application or the OS. Executing on a virtualized platform, each application executes within its own OS environment as a guest within a separate VM. These embedded applications are typically characterized as running under a real-time OS (RTOS) with one or more dedicated I/O devices. The driving function behind consolidation is the cost reduction associated with fewer platforms and lower maintenance costs, power consumption, heat dissipation and cooling, and weight, while increasing platform reliability due to fewer components, as illustrated in Figure 7.

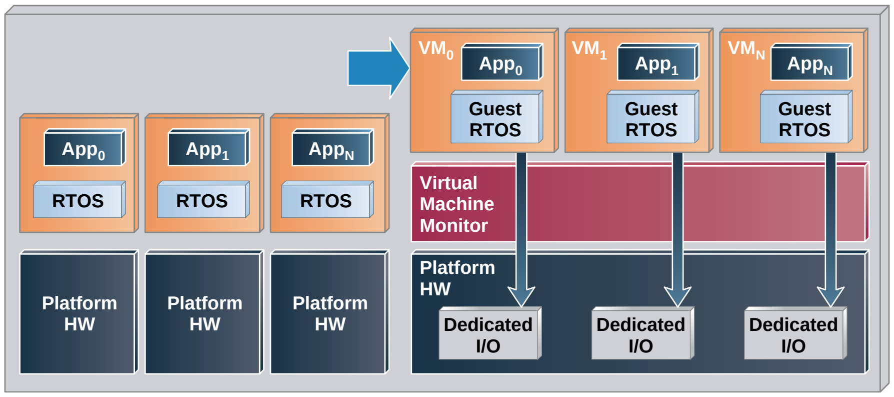


Figure 7: Consolidation usage model.

Examples of IVI and vehicle applications that could be consolidated are listed below.

- Engine information: alerts, warnings, and diagnostics
- Auto control, information: wipers, lights, turn signal, tire pressure
- Driver assist: lane departure warning, blind spot detection, front/rear proximity, external temperature, and directional information
- Fuel economy: average and instantaneous MPG, optimum speed, distance remaining to refuel
- Environmental controls: interior lighting, temperature regulation, mirror and seat positioning
- Electronic dashboard

Hybrid: The hybrid usage model diverges from the typical embedded hypervisor as a thin virtualization layer by integrating a RTOS or kernel into the hypervisor or partially-virtualizes an RTOS for closer coupling with the platform I/O, as shown in Figure 8. This usage model has particular value to the IVI market segment where the existing RTOS along with the IVI applications executing on it can be either integrated or partially-virtualized with the hypervisor, while new applications offered by a general purpose OS (GPOS) can be quickly brought to the IVI platform by executing it in a separate VM. Another consideration for the RTOS and GPOS partitioning is the boot time of the OS and availability requirements of the application. Applications or devices that require immediate availability are allocated to the RTOS partition, while those applications which are tolerant of a few seconds of delay in availability can execute in the GPOS partition.

"Applications or devices that require immediate availability are allocated to the RTOS partition, while those applications which are tolerant of a few seconds of delay in availability can execute in the GPOS partition."

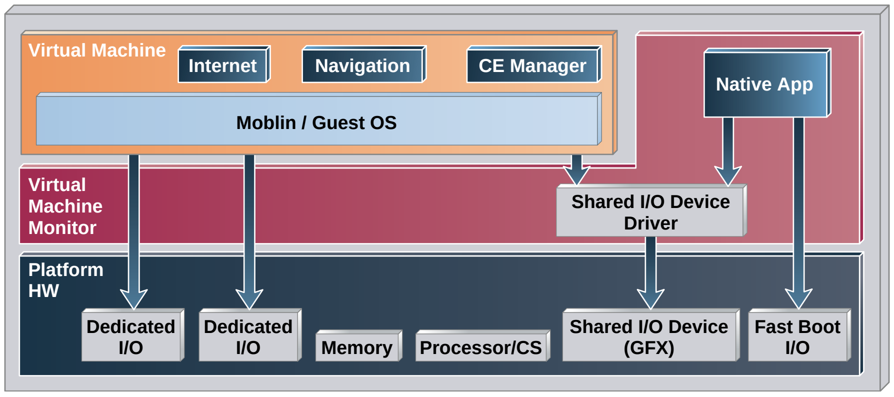


Figure 8: Hybrid usage model.

Examples of applications that could execute in a hybrid usage model and their applicable partition are listed below.

- RTOS partition
  Rearview camera
  Audio, video playback
  Digital radio
  Cell phone hands free
- GPOS partition
  Games
  Navigation
  Electronic owner's manual

One of the challenges is to provide I/O access to VMs through an efficient and secure implementation. Three such implementations are device emulation or partial virtualization or hardware assisted virtualization. Device emulation implements the physical device driver in the hypervisor, which emulates existing interfaces and incurs a latency penalty as each I/O must traverse the emulation driver. Partial virtualization allows a VM to directly access an I/O device to eliminate the latency penalty through a set of VMM specific interfaces, which require changes in the guest OS. The hardware-assisted I/O virtualization requires support in the platform chipset hardware.

"Optimizing the hypervisor to reduce virtualization overhead and achieve near real-time latency is an opportunity for continuous performance improvements."

The virtualization hypervisor induces overhead on the IVI platform performance, whether it is through the virtualization of the processor or emulation of the IO. Optimizing the hypervisor to reduce virtualization overhead and achieve near real-time latency is an opportunity for continuous performance improvements. The consolidation and hybrid usage models presented previously are just two examples of how virtualization can be implemented in an Intel-based IVI platform. Developing other models and application partitioning provides numerous opportunities for product differentiation and value-add to the end customer. Depending on the VMM model chosen, appropriate hardware may need to be designed into the platform upfront or select the appropriate Intel architecture-based processor and chipset solution for the Intel-based IVI platform.

Intel® Hyper Threading Technology (Intel® HT Technology)

This is one of the Intel technologies that is enabled by default on the Intel-based IVI platform in hardware. The key dependency to leverage from this is the support in the OS for symmetric multiprocessing (SMP). Some applications have been benchmarked and are known to show an improvement in performance by 30 percent with Intel® Hyper Threading Technology (Intel® HT Technology). The enabling of Intel HT Technology is transparent to the application, in the sense that the same applications running on a uni-processor machine can run in a seamless way. Future Intel-based IVI platform processors may support multiple cores and the same SMP software would run unchanged.

Maximum benefit of Intel HT Technology to the end-customer depends on the collaborative effort by the IBV, OSV, and ISV. Supporting SMP by default in the OS is an opportunity for the OS vendors to help facilitate execution of heavier workloads more efficiently. The application vendors in turn can develop high performance applications through development of multi-threaded applications that can execute in parallel. The benefits can be further extended by the ISVs making their middleware and device driver software MP-safe such as with reentrancy and non-blocking APIs.

"The enabling of Intel HT Technology is transparent to the application, in the sense that the same applications running on a uni-processor machine can run in a seamless way."

Security

There are two key aspects of security that an Intel-based IVI platform is targeted to support for "open and closed device" usage models and they are (1) to enable a tamper-resistant software environment to protect against malicious attacks and (2) to offer ability to playback DRM-protected content like Blu-Ray for rear-seat entertainment. The usage model shown in Table 2 for the Intel-based IVI platform exposes it to various types of threats and therefore presents the need to protect against them.

Usage Model	Threats
Internet Connectivity	Malware attack, DoS Attacks, Packet Replay/Reuse, etc.
Secure Internet Transaction	Steal Privacy sensitive data
DRM Content Usage	Steal DRM protected content
Browser Usage	Malware attack, Phishing
Software Downloads/Updates	Change OS/Software Stack
Device Management	DoS attack, Illegal device connections
ID Management	Dictionary Attacks, Stolen Privacy Data
One Time Provisioning	Steal OEM data, Unauthorized Activation
Full Featured OS	All of the above
Biometrics (Finger print sensor)	Steal user data, authentication credentials

Table 2: Usage model and security threats.

Based on the usage model described in Table 2, the assets on the platform that need to be protected that a hacker could attempt to compromise are as follows:

- Platform resources such as CPU, memory, network (3G, WiMax, WiFi).
- Privacy-sensitive data such as personal identification, address book, location, e-mail messages, DRM-protected copyrighted content like music and video.
- Trusted services such as financial, device management and provisioning, trusted kernel components

The mitigation against the security threats shall require the Intel-based IVI platform security architecture to use a combination of hardware and software security ingredients like:

- Trusted boot, secure storage and key management with Trusted Computing Group's Trusted Platform Module (TPM), coupled with appropriate hardware based root of trust such as Intel® Trusted Execution Technology (Intel® TXT). The Intel TXT feature is available only on certain Intel® Architecture Processors. In its absence, an appropriate alternative mechanism may be supported. These security features are becoming pervasive on most mobile platforms and would be very applicable to the Intel-based IVI platform as well.
- DRM content protection based on commercial media players executing on Intel architecture.
- Application isolation through OS-based mechanisms.
- Trusted domains and domain isolation through virtualization.
- Anti-virus through third-party software libraries and application design.

"Security is a strong requirement, considering the new threats the automobile is exposed to due to connectivity to the Internet and rich features."

Security is a strong requirement, considering the new threats the automobile is exposed to due to connectivity to the Internet and rich features as shown in Table 2. Providing a secure car in an open usage model has become a challenge. Not all of the above features are present in the current generation of the Intel-based IVI platforms. The expectation is that over time, many of these will be enabled in a phased manner.

The Intel-based IVI platform with an Intel Atom processor and an industrial grade TPM device will allow a car OEM to deliver differentiating security. This is also a great opportunity for third-party vendors to provide various platform security ingredients as outlined in the Table 3.

Threats	Security Ingredient
Malware attack	Application isolation, domain isolation, tamper-resistant software (TRS), anti-virus protection
Steal privacy sensitive data	Trusted boot, secure storage
Steal DRM protected content	Application isolation, domain isolation, tamper-resistant software
Change OS/software stack	Tamper-resistant software, trusted OS, secure boot
DoS attack	Tamper-resistant software, trusted OS, Anti-virus protection, trusted boot
Steal OEM data	Tamper-resistant software
Phishing	Anti-virus
Steal user data	Secure key management, secure storage, tamper-resistant software

Table 3: Security mitigation strategies.

The support for each of the security ingredients translates into the following opportunities:

- Silicon vendors or IHVs: Hardware for trusted boot and DRM, such as TPM
- OS and software vendors: Develop and deliver hardware and OS-assisted trusted boot, domain isolation, application isolation, anti-virus and DRM-enabled media players
- Academia: More research into robust crypto algorithms, audio/video encoding/decoding standards and a balanced hardware/software solution that would make efficient use of the CPU.

Manageability

The manageability or device management (DM) framework provides services on the client platform, for use by IT personnel remotely. These services facilitate the key device management functions like provisioning, platform configuration changes, system logs, event management, software inventory, and software/firmware updates. The actual services enabled on a particular platform are choice for the car OEM.

"These services facilitate the key device management functions like provisioning, platform configuration changes, system logs, event management, software inventory, and software/firmware updates."

Open Mobile Alliance - Device Management (OMA-DM) is one of the popular protocols that would allow manufacturers to cleanly build DM applications that fit well into the IVI usage model. Many of the standard operating systems support OMA-DM or a variation of it with enhanced security. The data transport for OMA-DM for the Intel-based IVI platform is typically over wireless connectivity like WiMax or 3G/4G. This protocol can run well on top of the transport layers like HTTPS, OBEX, and WAP-WSP. The Intel-based IVI platform would be able to support this, as long as the OEM supports the connectivity and the client services. One of the limitations of OMA-DM is that the Intel-based IVI platform is fully powered with the DM client services activated on top of a fully functional OS.

The other possible framework for manageability is Intel® Active Management Technology (Intel® AMT). Intel AMT provides full featured manageability that can discover failures, proactively alert, remotely heal, recover, and protect. Intel AMT Out of Band (OOB) device management allows remote management regardless of device power or OS state. Remote troubleshooting and recovery could significantly reduce dealer service calls. Proactive alerting decreases downtime and minimizes time to repair.

"Remote troubleshooting and recovery could significantly reduce dealer service calls. Proactive alerting decreases downtime and minimizes time to repair."

In the manageability space, making Intel AMT available on the Intel-based IVI platform is an opportunity that allows car OEM differentiation and provides a much richer manageability features. This is also an opportunity for IHVs/ISVs to pursue, while research into retrofitting OMA-DM with the benefits of the Intel AMT features on the part of academia would be welcomed to enable Intel AMT on the current generation the Intel-based IVI platform with limited to no hardware changes.

In-Car Connectivity

The rising popularity of passenger entertainment systems and cameras in and on the car places an increasing burden on the car's network infrastructure to transport large amounts of high-bandwidth, low latency video and audio data between built-in devices such as the head unit, DVD players and changers, cameras, rear-seat entertainment system, amplifiers, speakers, and remote displays. Legacy in-car connection technologies such as CAN, LIN, and Flexray are cost-effective and well-suited for messaging between engine control units (ECUs) in the car, but these technologies lack the necessary bandwidth to distribute video content. Blu-Ray video, for example, requires up to 54 megabits per second of bandwidth, well above the capability of legacy car networks. Technologies that have been explored for interconnectivity of in-car multimedia devices include MOST, Ethernet, and IDB-1394. MOST and Ethernet will both be enabled by the Intel-based IVI platform.

"Technologies that have been explored for interconnectivity of in-car multimedia devices include MOST, Ethernet, and IDB-1394. MOST and Ethernet will both be enabled by the Intel-based IVI platform."

MOST*

Currently, the leading in-car multimedia transport technology is Media Oriented Systems Transport (MOST). MOST requires each device in the network to be equipped with a MOST Network Interface Controller (NIC). MOST is a circuit-switched, time-domain multiplexed network technology that enables the transfer of digital audio, video, data, and control information between multiple networked devices. MOST scales in bandwidth from 25 Mbps (MOST25) up to 150 Mbps (MOST150). MOST150 is capable of transferring content at 150 Mbps over a polymer optical fiber (POF) wire harness. In addition to higher bandwidth, MOST150 features new isochronous transport mechanisms to support extensive video applications, as well as an Ethernet channel for efficient transport of IP-based packet data. MOST has a strong presence in automotive and many multimedia devices currently support MOST, so its popularity in high-end cars is not expected to diminish in the foreseeable future. The Intel-based IVI platform with the Intel Atom processor will feature a glueless, high-bandwidth dedicated interface to a MOST Intelligent Network Interface Controller (INIC). Drivers and a MOST protocol stack will also be available to enable MOST integration.

"The Intel-based IVI platform with the Intel Atom processor will feature a glueless, high-bandwidth dedicated interface to a MOST Intelligent Network Interface Controller (INIC)."

Ethernet

While the synchronous nature of MOST makes it ideal for transporting high-bandwidth, low-latency multimedia traffic between endpoints in a network, nothing surpasses Ethernet networking in terms of its raw bandwidth, cost, extensive eco-system, hardware availability, and software support. Historically, Ethernet has been used in the car strictly for diagnostic services and software updates. Ethernet cost-effectively scales up to 1 Gigabit per second, which allows large amounts of data exchange between the service infrastructure and the car. However, Ethernet quality of service (QoS) mechanisms have been inadequate for distributing low-latency, jitter-free synchronized audio and video traffic throughout the car. For example, if the car's head unit is distributing multiple channels of audio to network attached speakers and synchronized video to a network-attached display, the speakers and the display must share a common clock in order to prevent speaker channels from drifting apart and to maintain lip synchronization between the audio and video. Some new techniques being proposed by the recently-formed 802.1 Audio/Video Bridging (AVB) Task Group promise to provide time-synchronized low latency streaming services over Ethernet could address this shortcoming. Some car OEMs are also performing trials to explore data transport of IP over Ethernet with QoS mechanisms implemented in the IP layer. Still, AVB is likely to become the most cost-effective and efficient solution for audio and video streaming over Ethernet. The Intel-based IVI platform with the Intel Atom processor will include an Ethernet controller that includes support for AVB with a clock output pin to drive an external audio clock.

"Ethernet cost-effectively scales up to 1 Gigabit per second, which allows large amounts of data exchange between the service infrastructure and the car."

"The Intel-based IVI platform with the Intel Atom processor will include an Ethernet controller that includes support for AVB with a clock output pin to drive an external audio clock."

In-Cabin Connectivity (Car-to-Portable Device)

The demand for on-the-go access to audio/video content and information has exploded over the past few years. Consumers are increasingly consuming content on the go through iPods and portable multimedia players. Having a multimedia-capable platform in the car offers an opportunity to bring portable content in the car and use the high-quality displays, sound systems, and controls of the car to render that content in a more enjoyable way. Historically, portable devices have been connected to the car primarily through a USB port or a proprietary wired analog interface. An increasing number of devices now include wireless interfaces that enable devices to stream content to the car with more quality and less effort by the consumer. Bluetooth also enables hands-free cell phone operation. The Intel-based IVI platform includes a high-speed synchronous serial port to allow streaming of low-latency audio and voice from a Bluetooth chipset. The platform also offers a complete Bluetooth software stack and Bluetooth profiles for audio streaming and hands-free calling. WiFi controllers can be attached to the platform through PCI Express, USB, or SDIO.

Power Management

The Intel-based IVI platform has its own local power management architecture to manage the SoC and the automotive I/O fabric, with the associated policies. In addition, an automotive power state manager (PSM) can exist as a central entity to keep track of the overall power states in an automobile and respond to events like Ignition On/Off, Drive-Idle, Under/Over Voltage Events, Manageability Service calls, and so on. It is beyond the scope of this article to describe the PSM.

"An automotive power state manager (PSM) can exist as a central entity to keep track of the overall power states in an automobile and respond to events like Ignition On/Off, Drive-Idle, Under/Over Voltage Events, Manageability Service calls, and so on."

Traditional Intel architecture platforms support various power management capabilities, to conserve power and this applies to both battery and AC powered platforms. The Intel-based IVI platform system does support the power management states as shown in Table 4.

Description	System	CPU	Device	Usage
Fully On	S0	C0-C6	D0	Ignition On
Low On/User Idle	S1		D1/D3	Thermal Management
Standby/System Idle	S2		D2/D3	Thermal Management
Sleep/Suspend to RAM	S3	Power Off	D3	Not Used
Hibernate/Suspend to NVM or Soft Off	S4/S5	Power Off	D4	Not Used

Table 4: Intel-based IVI platform power state usage.

As it was highlighted earlier, one of the key design goals of the Intel-based IVI platform is a fast boot in the order of seconds. Typically, any resumption from Suspend/Hibernate back to active state involves restoring the previous state. In an automobile environment with multiple users of the same vehicle, it is challenging to resume from suspend, as the users might change across these transitions and one users context could get inadvertently restored for another user. This makes the fast boot with a completely fresh state on every power on a key requirement for the platform, so that every user always starts with a new context. In addition, long duration S3 states may drain battery power if not handled properly.

The built-in clock throttling feature of the Intel-based IVI platform can be used for thermal management, in addition to power managing the idle I/O devices. However, the actual usage model is left to the OEM and can be used in conjunction with the climate control system of the automobile.

Conclusions

Enabling the standards-based the Intel-based IVI platform on Intel architecture is opening up many opportunities both from a technical innovation and business opportunity standpoint. The scope of these opportunities will further expand with the wide availability of WiMax/4G infrastructure for the connected car usage model.

In consideration of the combined technical and market requirements in automotive infotainment, leading automobile manufacturers and suppliers announced in March 2009 the formation of the GENIVI Alliance, a nonprofit organization committed to driving the development and broad adoption of an open source In-Vehicle Infotainment (IVI) reference platform.

By removing the costly duplication in specifications and lower level software functions, opportunities for new services desired by automotive customers can be developed instead, resulting in new sources of revenue. Based primarily on key technical working groups, the outcome of vetted specifications and reference implementations will be available to all members for commercial development along with many elements planned for release into open source.

"This makes the fast boot with a completely fresh state on every power on a key requirement for the platform, so that every user always starts with a new context."

The new Alliance (expected to reach up to 150 companies by 2010) will unite industry leading automotive, consumer electronics, communications, application development and entertainment companies investing in the IVI market to align requirements, deliver reference implementations, offer certification programs and foster a vibrant IVI community with the purpose of removing waste from the development cycle.

References:

1. Microsoft Corporation, Microsoft* Auto Platform Overview *http://download.microsoft.com/download/6/5/0/6505FA0E-1F39-4A34-BDC9-A655A5D3D2DB/MicrosoftAutoPlatformOverview3%201.pdf*
2. Moblin.org, Moblin* for IVI – Software Architecture Overview, *http://moblin.org/pdfs/moblin_ivi_sao_v0.8.pdf,* May 2008.
3. Intel Corporation, Intel® Atom™ Processor Z5xx Series and Intel® System Controller Hub US15W Development Kit, *http://www.intel.com/design/intarch/devkits/index.htm?iid=embnav1+devkit*
4. Intel Corporation, Intel® Active Management Technology (Intel® AMT), *http://www.intel.com/technology/platform-technology/intel-amt/*
5. Media Oriented System Transport Technology, *http://www.mostcooperation.com/home/index.html*

Author Biographies

Suresh Marisetty: Suresh Marisetty is a software and systems architect at Intel Corporation and has been involved with enabling of various end-to-end Intel® architecture server, desktop, mobile, and embedded platform technology ingredients across the industry, in close collaboration with OEM customers, OS vendors, and standards bodies. He has been with Intel for about 20 years and currently has 17 patents and over half a dozen internal and external papers on various topics. His current focus areas include security, manageability, low latency boot solutions and end-to-end IVI platform software architecture.

Durgesh Srivastava: Durgesh Srivastava is silicon and systems architect for Intel Corporation's Low Power Embedded Products Division (LEPD). He has been involved in debug hooks architecture, power reduction, platform power management, CPU uncore/MCH, memory subsystem architecture, and silicon debug. He is currently driving 32-nm System-on-a-Chip solutions for the next generation embedded products.

Joel Andrew Hoffmann: Joel Andrew Hoffmann is the Strategic Market Development Manager for Intel's In-Vehicle Infotainment Group. In this capacity, he leads automotive business developments in wireless networking and enterprise IT solutions. He is also responsible for managing Intel's Infotainment vision, which includes spearheading Intel's efforts with GENIVI, a cross-industry consortium of companies with the goal to accelerate adoption of entertainment, navigation and communication services in automobiles. Within GENIVI, Hoffmann promotes community support for translating open source innovation into commercial solutions adopted by automakers, suppliers and a growing infotainment ecosystem.

Since joining Intel Corporation in 2000, Hoffmann has been instrumental in establishing an early incubation of connected vehicles for DaimlerChrysler, now Chrysler, by promoting standards-based software and hardware alignment and helped institute a high performance computing cluster which reduced the auto company's costs substantially. With a background of sales and technical experience in automotive, telecommunications and consumer electronics, Hoffman has previously created business plans for the Michigan Connected Vehicle Proving Center, developed marketing strategies for Skyway Systems, now Inilex, and held a board seat in the Connected Vehicle Trade Association on Intel's behalf.

Previously, as Global Automotive Industry Manager, Hoffmann developed Intel's first automotive vertical strategy focused on business transformation, new market growth and engagement with auto industry companies such as BMW, Toyota, Ford and DaimlerChrysler.

Acknowledgements

The authors would like to acknowledge the following for their feedback and contributions to the paper: Staci Palmer, Peter Barry, Lomesh Agarwal, Matt Sottek, Lindsey Sech, Steve Adams, Scott Hoke, Lynn Comp, Sam Lamagna, and Mark Brown.

Copyright

Intel Technology Journal

Peer Reviewers

Lomesh Agrawal	Dr. Giannone	Chris Oliver
Stephen Ahern	John Griffin	Chun Keang Ooi
Lars Boehenke	Alan Hartman	Staci Palmer
John Browne	Joel A. Hoffman	Frank Poole
Mike Cerna	Scott Hoke	Gina M Rophael
Robert Chavez	Dr. Jin Hu	Greg Schultz
Phillip Clark	Robert M Kowalczyk	Li Shang
James A Coleman	Sam Lamagna	Matt Sottek
Mike Delves	Chris D Lucero	Jason Spyromilio
Scott M Doyle	Bryan Marker	Mark R. Swanson
Richard Dunphy	Louis Morrison	Blake A Thompson
Rob Dye	Dave Murray	Tom Tucker
Roger Farmer	James E Myers	Hwan Ming Wang
Michael Flemming	Nick Nielsen	Dr. Luthar Wenzel
Rachel Garcia Granillo	Dr. Ohbayashi	

For further information on embedded systems technology, please visit the Intel® Embedded Design Center at: http://intel.com/embedded/edc

For further information about the Intel Technology Journal, please visit http://intel.com/technology/itj

ISBN 978-1-934053-21-8

9 781934 053218

7 35858 20967 0

$49.95 US