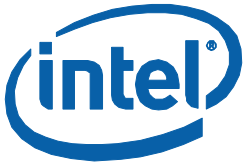


White Paper

Sathish Venkataramani

Technical Marketing Engineer

Intel Corporation

Embedded Intel® Architecture Board Bring Up Procedure

August 2009

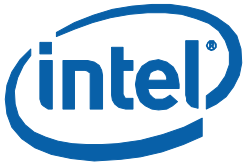
322504

Executive Summary

Embedded Intel® architecture has grown complex over the generations with the additions of newer architectural features and high speed devices. Once the board design has been completed using the guidelines provided the engineer will need to do a board bring up. This paper will explain the tests that can be done to make sure the board is stable, and also give debug methods to solve issues occurring during board bring up.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. www.intel.com/embedded/edc. §

Contents



Background

All embedded Intel® architecture products vary in their uses, but they have one thing in common: the complexity of their design. As an engineer at Intel or an external customer you need to understand the steps involved in bringing up a board after the design stage has been completed and the boards are built.

This white paper discusses some of the major factors that need to be considered in bringing up and debugging an embedded Intel® architecture-based board and the various factors that come into play when the board is switched on for the first time. Reading this paper will help you to identify issues as and when they occur and fix them to get the best performance from your hardware.

Groundwork

In the world of high speed digital design and Intel® architecture, once the schematics and layout are checked and finalized, the board is built.

Board bring up of an early sample is one of the most important steps for a design team. The early sample must pass through a varied set of tests to see that the hardware is rock solid. To have the system boot, the board first needs to be checked before turning on the power to the system. The following spot checks can be used to lessen the problems that happen during the first boot.

Stop! Look and Proceed

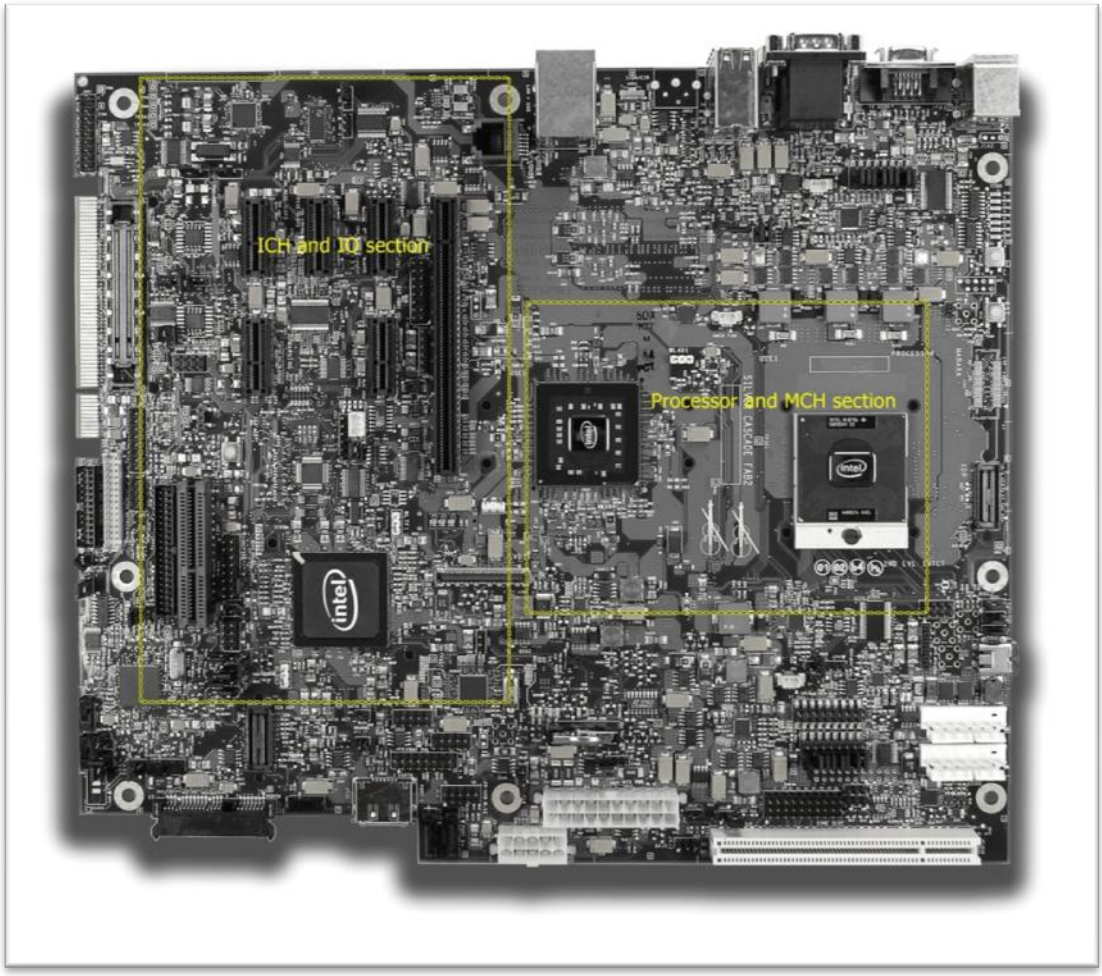
When the board is built most designers, in their eagerness to get the project rolling, will try to power it on immediately. But before that there are a few spot checks that can quickly identify problems that can damage the hardware when powering it on. The first check is to look over the board and check for any parts that are loose or partly soldered. This does not mean looking at every inch of the board, but spotting the main components (processor, memory, chipset, etc.) and the areas around them. Figure 1 shows a sample Intel® board with the main sections marked.

Also, now would be good time to look more closely at the processor socket of the board to see if there are any bent pins. This is by far the most common cause for errors, including no fetches, during boot up. Once you are sure the socket is not damaged and the processor seats correctly you can move your attention to the second most important component: the memory. Inspect the

DIMM socket and check for any small metal pieces or foreign elements inside the socket. This occurs especially if the board is new and went through a packaging environment. These pieces have a high chance of causing a short between two DIMM signals which might either result in data errors or trigger a short circuit fault.

Once these spot checks are complete and the checks revealed no problems you can proceed to the next step.

Figure 1. Sample Customer Board Showing the Different Components



Debug

Note: Debug or Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware thus making it behave as expected. (*Source*: Wikipedia)

Note: The term "bug" in the meaning of technical error dates back to at least 1878 and Thomas Edison and "debugging" seems to have been used as a term in aeronautics before entering the world of computers. (*Source*: Wikipedia)

Debugging is an art which is generally painful to learn the first time as it is a time-consuming process, but the success of a good debugger lies in the fact that he has learned from his previous debug attempts. Just like a software debugger has the ability to set break points to start, stop and monitor the execution of a program, the hardware engineer can follow the basic methods outlined below to make the process of debugging easier.

Everything Needs Power

The board has many components doing multiple tasks at any given moment. The one common need of all the devices is power, which makes this the most important thing that a design engineer needs to pay attention to. The power supply provides the major (12V, 5V etc) to the system. Once the power gets into the board the various voltage regulators step the voltage down to levels required by each component (3.3V, 1.8V, 1.05V etc).

When the board is switched on, you can use a multi-meter to test the output voltage pins on all the regulators to make sure they are operating at the rated voltage value. This test makes it easy to debug power failures on board. If for some reason a certain voltage rail is not being driven to the specified level, then you can concentrate on that particular circuit, checking for any shorts, enable faults (when the enable line to a voltage regulator does not trigger) or for opens and solder joint issues on the particular circuit.

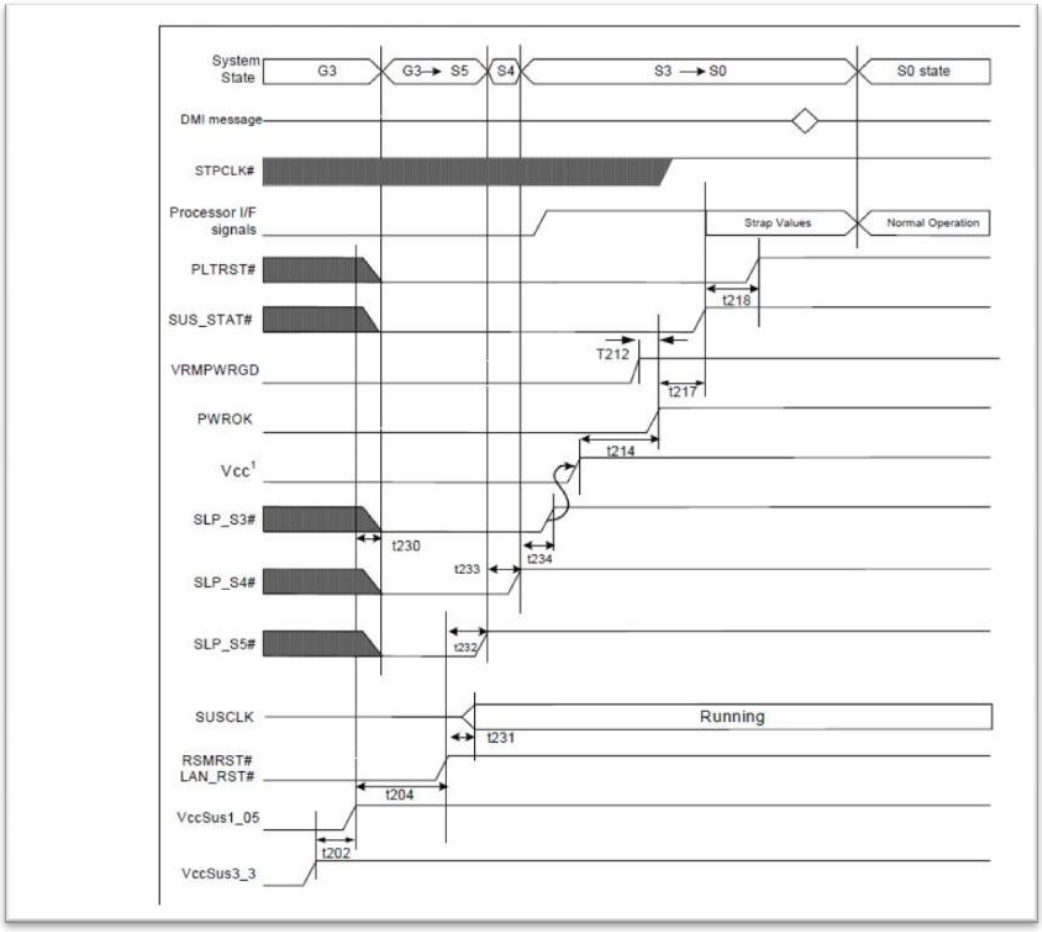
In Order, Please!

What happens in the initial few seconds before the processor starts fetching the code? In this part of the boot sequence the system power good is generated based on all the power rails coming up to the normal levels and also coming up in the right order.

Normally, the order in which the rails need to come up are specified in the Platform Design Guide (PDG) and also on the component External Design Spec (EDS). Figure 2 shows a sample start up power sequence from Mechanical Off to a fully active S0 state (On state) given in the Intel® I/O Controller Hub 9 spec. This does not provide all the signals but gives an

overview of the main signals that need to be active before a power good (PWRGD) signal is asserted.

Figure 2. Example of Power Sequencing from the Intel® I/O Controller Hub 9 Specification



The timings for the signals are provided in Figure 3:

Figure 3. Example of Power Sequencing Timings From the Intel® I/O Controller Hub 9 Specification

Sym	Parameter	Min	Max	Units	Notes	Fig
t209	V5REF active to Vcc3_3 active	0	—	ms	1	8-14
t211	Vcc1_5 active to V_CPU_IO active	—	—	-	4	8-14
t212	VRMPWRGD active to PWROK active	3	—	ms		8-15 8-17 8-18
t213	VccSus supplies active to Vcc supplies active	0	—	ms	3	8-14
t214	Vcc supplies active to PWROK Note: PWROK assertion indicates that PCICLK has been stable for at least 1 ms.	99	—	ms		8-14 8-15 8-17 8-18
t217	PWROK and VRMPWRGD active to SUS_STAT# inactive and Processor interface signals latched to strap value	33	71	RTCCLK	5, 10, 12	8-15 8-17 8-18 8-20
t218	SUS_STAT# inactive to PLTRST# inactive	2	3	RTCCLK	10	8-15 8-17 8-18 8-20
t219	PLTRST# assertion to VccGLANPLL inactive for platforms using ICH9 integrated GbE LAN	200	—	µs	11	
t228	HDA_RST# active low pulse width	1	—	µs		
t229	HDA_RST# inactive to HDA_BIT_CLK startup delay	162.8	—	ns		

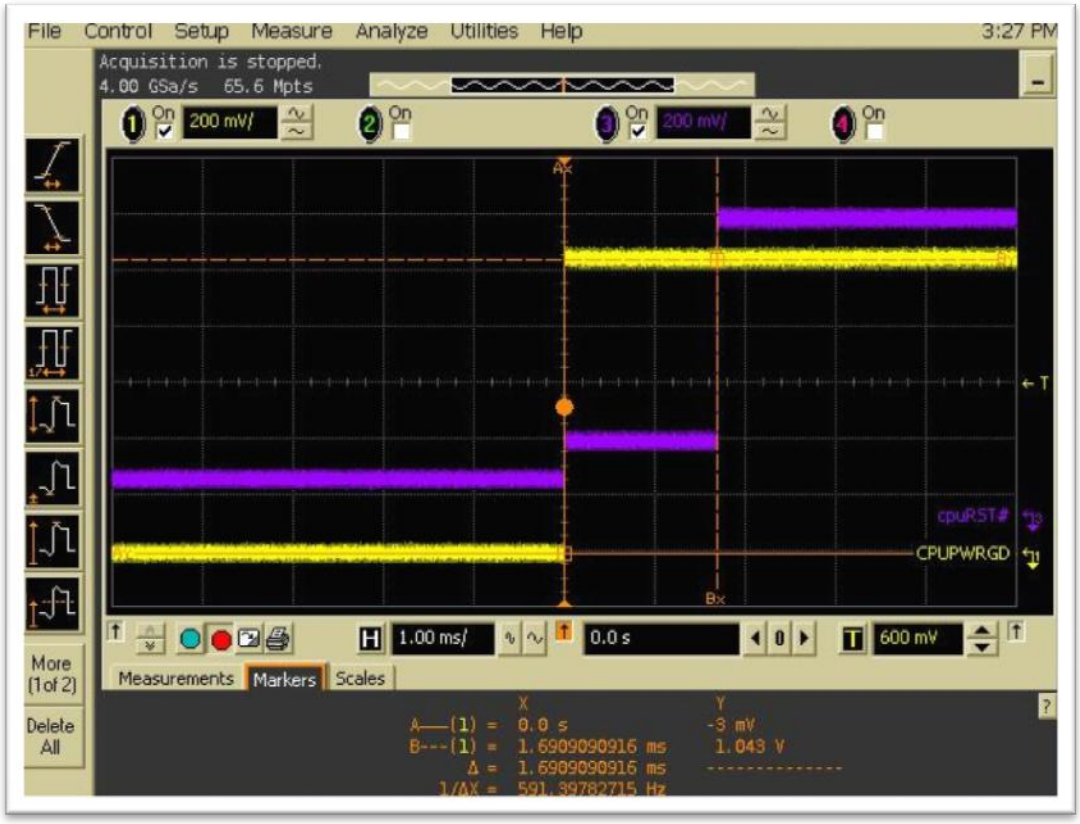
One of the methods used to debug the sequencing is to use an oscilloscope to view the main power signals. Use the platform design guide (PDG) and the power sequence information normally included in the schematic to check for the timings and the order of the signals. The power up of the rails is normally handled by an embedded controller or a CPLD based on the design. If the measurement reveals wrong sequencing, the code or the firmware for the controller needs to be debugged and the order checked.

Note: It is good practice to include the power sequence diagram as a part of the schematics.

Once the system power good is asserted, this power good is also most likely routed through logic to provide the CPU power good (CPU_PWRGD) signal. The next signal to assert would be the CPU reset (CPU_RESET#) signal. The oscilloscope shot taken below shows a sample of the CPU reset after the CPU power good asserts. This is point of handoff of the system to the CPU and the fetching of the code starts. This scope shot shows the importance of timing as

given in the tables above. They are to be followed as specified to get the board booting quickly without any errors.

Figure 4. Example Snapshot of CPU Reset After the Power Good Assertion



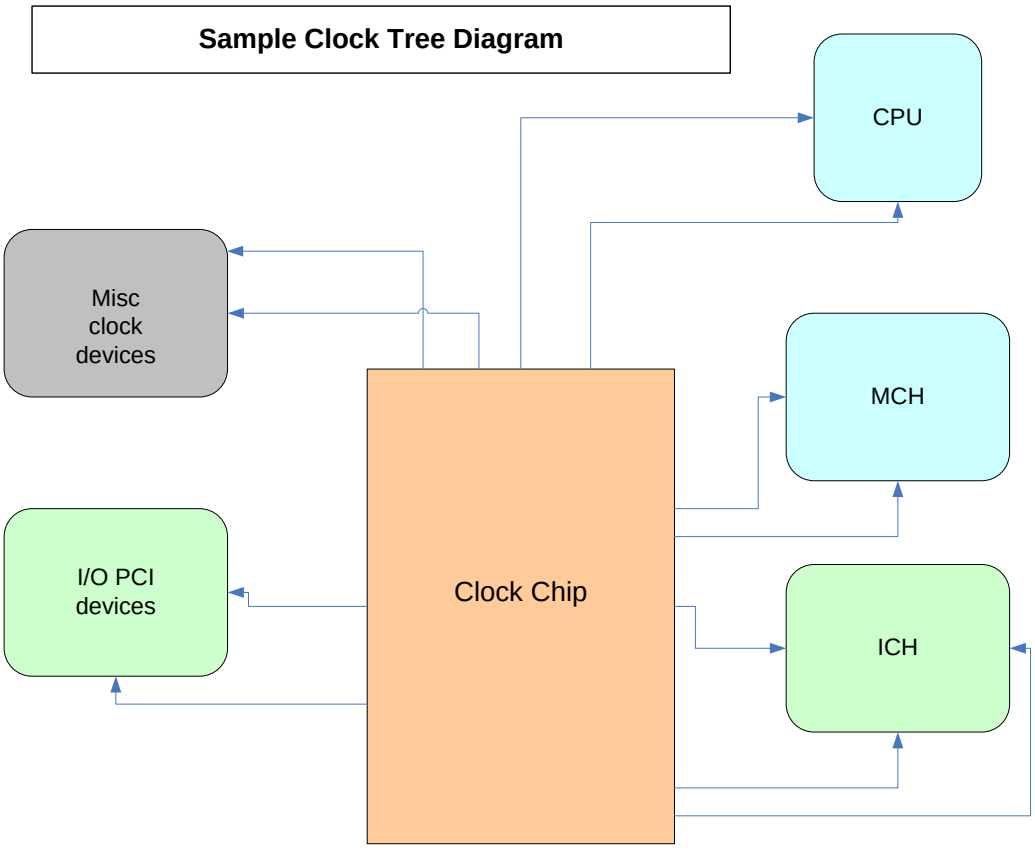
A Clock on Time Saves Nine!

Even if the power rails all work in accordance with the spec and the sequencing is perfect, there is still a chance the board will not come out of reset to start fetching code out of the bios. In that case the most likely place to check will be the clocks of the system.

Clocks play a critical role in the boot up of the system. In addition to power rails, CPUs, chipsets and other components need reference clocks in order to start up. On the system board there normally is a clock generator chip. A quick check using an oscilloscope will let you know if all the clocks are up and running. This check should be a part of any board bring up effort. The following figure shows a sample clock tree which is often included in the schematics.

Note: It is often good practice to include the clock tree as a part of the schematics.

Figure 5. Example Clock Tree Diagram



Code of Conduct

One other key factor during board boot up is the BIOS. This is the most critical piece of the code the machine will run every time it boots. The problem with legacy BIOS is that it has been written and re written so many times that it is like a black box. If you were to remove a module for a legacy device that is no longer used, there is an inherent risk that it will affect some other device. The BIOS sets up the entire system, runs the post process which initializes the interrupt controllers, the DMA controllers, and checks the PCI bus finding devices and adding them to the device table.

The initial drivers which are used by the system to access the keyboard, mouse, and video are all a part of the bios. The BIOS engineer needs to pay attention to the functionality of the system and make sure the code is written correctly according to the spec of the devices he is dealing with on the board. The post process can also be useful as a debug tool as it can be programmed to provide beep codes depending on the error encountered. These beep codes

vary based on the vendor. A sample list of the beep codes is provided in Table 1.

Table 1. Post Beep Codes

American Megatrends Beep Codes		
Beeps	Error Message	Meaning
1	DRAM Refresh Failure	Motherboard refresh circuits Faulty
2	Parity Error	Parity Error in first 64k of RAM
3	Base 64KB or CMOS failure	First 64k or CMOS memory faulty
4	System Timer Failure	Motherboard timer not working
5	Processor Error	The CPU reported an internal error
6	8042 error or A20 gate failure	Keyboard controller bad, or bios cannot switch to protected mode
7	CPU exception	Internal CPU, or interrupt error
8	Display memory R/W error	Video card is not responding or RAM faulty
9	ROM BIOS checksum error	ROM checksum not same as value in BIOS
10	CMOS shutdown R/W error	CMOS shutdown register faulty
11	External Cache bad	External cache not operational

For a detailed debug of the post process, it is useful for the post to output the error code to the I/O ports. The normally used I/O port for errors is port 80. The port is connected to an LED display which provides codes which point to specific error messages. Many modern motherboards incorporate this feature during the testing phase of the boards. Other instruments that are useful in debugging the post process are logic analyzers and In-Circuit Emulators (ICE). Logic analyzers capture signals on the bus in real time and allow play-back for a detailed study. In-circuit emulators connect to the connectors of system components and capture data that exits and enters the component, emulate dataflow on components such as CPU, chipset etc.

The Intel BIOS Writers Guide provides detailed information about writing the bios for a particular platform and processor. More information can be found on the developer website at http://developer.intel.com and the embedded design center http://edc.intel.com.

We Debugged, We Fixed, and It Works!

This paper is a sample of lot of techniques board design engineers use to debug their designs. For more information on Design Debug principles please check various publications by Mindshare*, Be the Signal*, or Signal Consulting*, Inc.

For more information on BIOS please check various publications by Techstream*, Inc.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://intel.com/embedded/edc.

References

1. "High Speed Digital Design", Two Days of Black Magic with Dr. Howard Johnson, presentation manual, HSDD 2007 V .13

2. High Speed Digital Design: A Handbook of Black Magic, Dr. Howard Johnson, Prentice Hall publications

3. High-Speed Digital System Design: A Handbook of Interconnect Theory and Design Practices

4. "BIOS ESSENTIALS", Techstream Inc, presentation manual, www.techstream.com

Authors

Sathish Venkataramani is a Technical Marketing Engineer with the Embedded Computing Group (ECG) at Intel Corporation.

Acronyms

DIMM	Memory
EDS	External Design Specification
Ghz	Gigahertz
PCI	Peripheral Component Interconnect
PDG	Platform Design Guide
PWRGD	Power Good
Scope	Oscilloscope