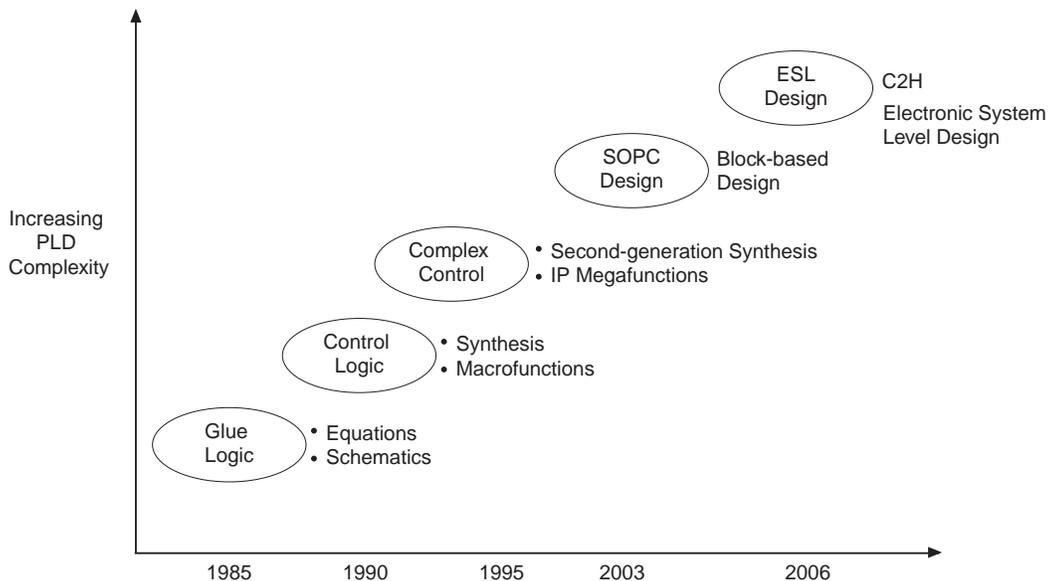


Introduction

The cost of designing traditional standard cell ASICs is increasing every year. In addition to non-recurring engineering (NRE) and mask costs, development costs are increasing due to design complexity. Issues such as power, signal integrity, clock tree synthesis, and manufacturing defects can add significant risk and time-to-market delays. FPGAs offer a viable and competitive option to traditional standard cell ASIC development by reducing the risk of re-spins, high NRE costs, and time-to-market delays.

Programmable logic has progressed from being used as glue logic to today's FPGAs, where complete system designs can be implemented on a single device. The number of gates and features has increased dramatically to compete with capabilities that have traditionally only been offered through ASIC devices. Figure 1 illustrates the evolution of FPGA applications that have led to higher density devices, intellectual property (IP) integration, and high-speed I/O interconnects technology. All of these elements have allowed FPGAs to play a central role in digital systems implementations. With the availability of multimillion-gate FPGA architectures, and support for various third-party EDA tools, you can use a design flow similar to that of traditional standard cell ASIC devices to create system-on-a-programmable-chip (SOPC) designs in FPGAs. In addition, with the Altera® Stratix® series of FPGAs, you have a seamless path to HardCopy® ASICs.

Figure 1. Application of FPGA Devices from 1985 to the Present



With the device sizes and architectures that are available today, FPGAs can effectively implement systems that were once possible only in traditional standard cell ASICs. Because of their programmable capability, FPGAs reduce the time to bring up a system as well as minimize the financial risk involved with new designs. Some of the newer FPGA devices have resources such as on-chip transceivers for different physical layer (PHY) protocols, providing the capability to interface with external memories and implement large blocks of internal memory. All of these aspects help to reduce device count on a board, in turn bringing down the cost associated with the product.

Cyclone® series FPGAs provide a low-cost alternative for applications that currently use low-to-moderate-density traditional standard cell ASICs. A rich feature set makes Cyclone series FPGAs suitable for a broad range of applications including displays, wireless communication, video and image processing, automotive, and military at a cost per device that is comparable to traditional standard cell ASICs.

Stratix series FPGAs provide a good alternative to more complex traditional standard cell ASICs. These are high-performance, high-end FPGAs that have resources such as large internal memory, fast external memory interfaces, on-chip transceivers, and a large number of internal clock networks. Stratix GX series devices and Arria™ GX devices have on-chip transceivers that support a number of industry-standard serial interfaces in addition to other logic resources. Stratix series FPGA devices, which includes Stratix, Stratix II, Stratix III and Stratix IV, also provides a cost effective migration option for going to volume production with HardCopy ASIC offerings from Altera. By choosing to go to HardCopy ASICs, you can utilize the programmable feature during the design, verification, and prototyping stages, reduce the time to market, and get the cost, power, and performance benefits in production. You can also reduce the typical problems associated with traditional ASICs, such as low yield and issues with testability and reliability.



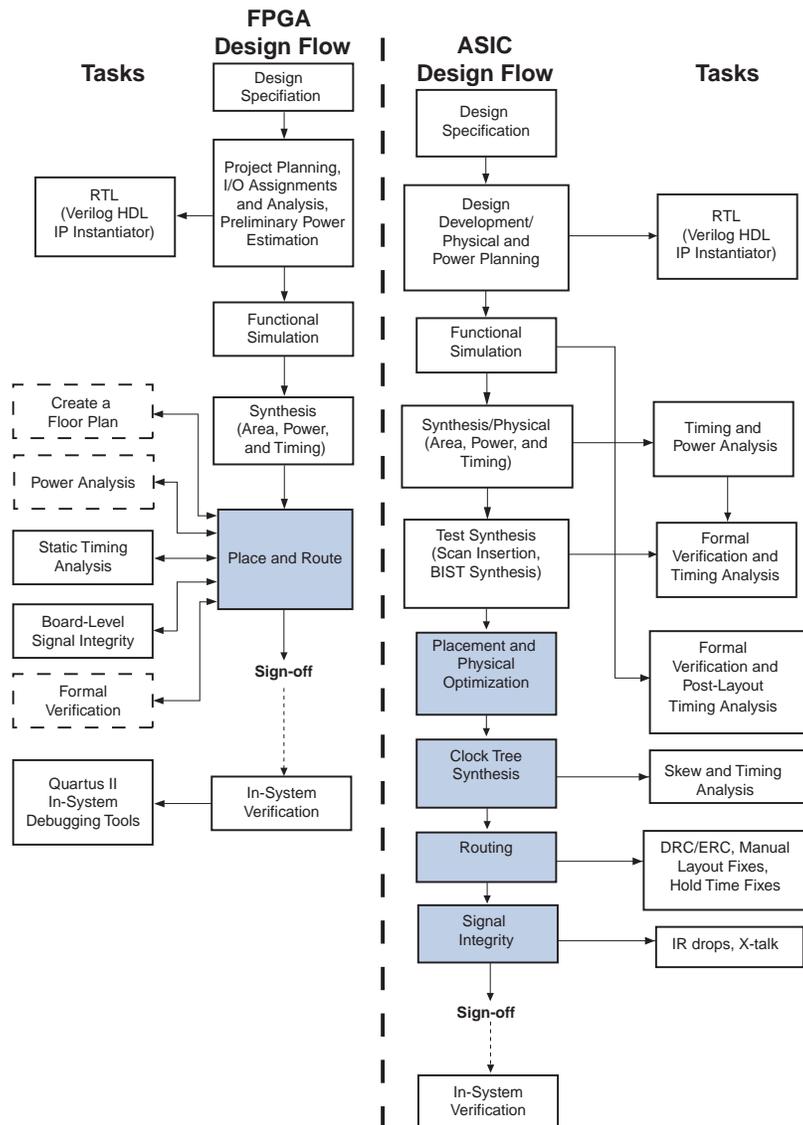
For more details about HardCopy ASICs, and for design guidelines for targeting your designs to HardCopy series devices, refer to the HardCopy device handbooks at www.altera.com.

This document is intended for traditional standard cell ASIC designers considering FPGA or HardCopy ASIC implementation of their designs, either for prototyping or for production. This application note gives some guidelines for efficient FPGA implementation of their designs. It also discusses the salient features of Altera's Quartus® II software that make it easy for building true system-on-a-programmable-chip (SOPC) solutions. Comparisons between typical FPGA and traditional standard cell ASIC design flow are also made whenever applicable.

Traditional Standard Cell ASIC and FPGA Design Flows

Typical traditional standard cell ASIC and FPGA design flows are shown in [Figure 2](#). The back-end design of a traditional standard cell ASIC device involves a wide variety of complex tasks, including placement and physical optimization, clock tree synthesis, signal integrity analysis, and routing using different EDA software tools. When compared to the design of traditional standard cell ASIC devices, the physical, or back-end design of Altera FPGAs is very simple and is accomplished with a single the Quartus II software. The Quartus II software is a fully integrated, architecture-independent package offering a full spectrum of logic design capabilities for designing with Altera FPGAs.

By using Altera FPGAs in place of traditional standard cell ASICs, you can potentially reduce the complexity of the design process as well as significantly reduce cost. This document discusses and compares each of the tasks involved in the design flows for both FPGA and traditional standard cell ASIC devices.

Figure 2. A Comparison of Traditional Standard Cell ASIC and FPGA Design Flows

Design Specification

The design specification stage includes the following activities, which will impact the choice of FPGA device:

- I/O specification
- System timing requirements and timing constraints
- Memory requirements
- Verification methodology
- Selection of the FPGA family and device, including the speed grade

I/O Specification

Altera devices support a wide variety of I/O standards. I/O resources vary depending on the device and family. When you design a traditional standard cell ASIC, you can instantiate I/O pads for a design by specifying the technology I/O buffers in a Verilog HDL or VHDL file to perform simulation and synthesis. At the foundry, the I/Os specified in the RTL are replaced with the technology I/O pads.

In an Altera FPGA design flow, you choose the type, location, and I/O settings for all the pins in your design with the Pin Planner, which is part of the Quartus II software.

With the Pin Planner, you can validate your I/O assignments by performing legality checks on your design's I/O pins and surrounding logic. These checks include proper reference-voltage pin usage, valid pin location assignments, and acceptable mixed I/O standards.

As part of I/O planning, especially with high-speed designs, you should take board-level signal integrity and timing into account. When you have an FPGA device with high-speed interfaces on a board, the quality of the signal at the far end of the board route, as well as the propagation delay in getting there, are vital for proper system operation.

The Quartus II software provides features to take these factors into consideration, making the software "board-aware." The Quartus II software can take into account board routing and external devices to generate advanced timing reports and board simulation modeling files.

The Quartus II software provides the following methods of signal integrity analysis:

- I/O timing using a default or user-specified capacitive load with no signal integrity analysis (default)
- The **Enable Advanced I/O Timing** option utilizing a user-defined board trace model to produce enhanced timing reports from accurate, "board-aware" simulation models
- Full board routing simulation in third-party tools using IBIS or HSPICE I/O models generated by the Quartus II software



For additional information about using the Pin Planner, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Handbook*.

Some Altera device families have dedicated circuitry associated with I/O cells to support multiple channels of serial transceivers. The on-chip transceiver circuitry provides physical coding sublayer (PCS) and physical media attachment (PMA) implementation for PHY protocols, such as PCI Express (PCIe), Gigabit Ethernet, XAUI, SPI, and SONET. By using an FPGA that has integrated transceivers you may be able to reduce the overall device count on your boards and reduce cost.

Number of I/O Pins

Determine the exact number of I/O pins used in your design. With this information, you can select a specific device and package.

Location of I/O Pins

Carefully analyze the impact of the I/O pin locations on the board layout to minimize potential problems. On an FPGA, you must know the I/O standards that are supported by a device so that modules in the design that require a certain I/O standard can be physically placed near the corresponding I/O bank. Some Altera IP cores (such as PCIe) specify pinout constraints automatically that may be needed to meet any special requirement. Use the Pin Planner to specify the locations for all other I/Os. If you anticipate a change in the target device at a later time in the design cycle, the Pin Planner also helps you plan for device migration.

-  To learn more about the location of I/O banks and supported standards within an I/O bank for a specific device, see the corresponding Altera device handbook.

I/O Timing

I/O timing is affected by the I/O standard and the drive strength and pin loading. In addition, to meet I/O requirements, Altera FPGAs have I/O pins with fast, dedicated registers. The device uses these registers, depending on the clock setup time (t_{SU}) and clock-to-output delay (t_{CO}) requirements. You can enable the use of fast I/O registers by setting the Fast Input Register and Fast Output register options using the Assignment Editor in the Quartus II software.

-  Redefine I/O specifications for every new design because different FPGA families may support different I/O standards. Even within a device family, different devices have different numbers of I/O pins.

Starting with the Quartus II software version 7.0, you can use the Advanced I/O Timing feature to specify the board parameters for each I/O individually so that the timing analysis is very accurate.

-  For more details on this feature, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Memory Requirements

You must identify embedded RAM capacity and speed requirements early in the design specification process. The size of embedded memory you can implement in an FPGA depends on the selected device. Stratix series devices support between 4 MBits and 27 MBits of embedded memory. Cyclone series devices support between 288 KBits to 3.8 MBits of embedded RAMs.

-  Refer to the appropriate device handbook for details about available embedded RAM.

Memory capacity in traditional standard cell ASIC technology is higher than in FPGA technology, but integrating and testing memory in the standard cell ASIC device requires significantly more work, because it generally involves instantiating a special module in the register transfer level (RTL) design, as well as creating custom circuitry around the RAM blocks for testing. For Altera FPGAs, the Quartus II software allows you to arrange memory blocks to meet the system requirement automatically without having to create any special blocks for testing. The In-System Memory Content Editor in the Quartus II software helps verify the contents of the memories in the FPGA when you are in the test phase.

For more information about how to address your memory requirements in Altera FPGAs, refer to [“Specification of External and Internal Memory”](#) on page 15.



For more details about the memory organization in a specific device, refer to the appropriate device handbook.

Verification Methodology

Most of the traditional standard cell ASIC projects start with an exhaustive verification plan and design specifications that are completely frozen. A similar strategy of starting a verification plan early in the design cycle is desirable for today's complex FPGA designs. Decisions such as whether to use formal verification methods should be made early. The Altera design flow supports industry standard simulation and formal verification tools to aid you with verification.

A key difference in verification methodology between standard cell ASIC-based systems and FPGA/HardCopy ASIC-based systems is the ability to utilize in-system, at-speed verification of your hardware and software, enabling true software and hardware co-design and co-verification. This change in the verification methodology of your chip and system designs is a key contributor to saving millions of dollars and months of time in getting your system to market.

FPGA Device Sizing

Estimate the size of your design so that you can arrive at the power and heat dissipation requirements. After the size of the logic is estimated and operating frequencies are specified, the approximate size and speed grade of the device can be selected.

Altera FPGAs also support vertical migration within the same package. In this context, you can migrate between devices whose dedicated pins, configuration pins, and power pins are the same for a given package across device densities. For example, you can migrate a design from an EP1S10B672C6 device to an EP1S20B672C6 device in the 672-pin package.

Number of Low-Skew Signals

Identify the number of clocks required in your design to choose the FPGA for your implementation. There is usually an upper limit to the number of different low-skew signals available in a device. Clock tree insertion is a manual process in the traditional standard cell ASIC flow, whereas in an FPGA flow it is an automatic process. In most cases, the Quartus II software can automatically choose which low-skew resources to assign to which signals. In addition, it can also determine which signals should be routed on low-skew resources. However, you can make these assignments manually.

Altera FPGAs provide a variety of clocking resources in each device. Clocks that are to be used in all regions of the chip can be routed over global clock nets, which are guaranteed to be low skew. The global clock networks span the entire general purpose logic arrays, feeding all architectural structures. Internal logic, PLL outputs, or device inputs might drive these low-skew global clock resources. You can also use these low-skew global resources for other device-wide signals with large fan-outs, such as asynchronous resets or clock enables. The number of device wide low-skew resources

available is device dependent. Stratix devices provide fast regional clocks in addition to the global low-skew resources. Similarly, Stratix II, Stratix III, and Stratix IV devices provide a hierarchical clocking structure consisting of global clocks (GCLKs), regional clocks (RCLKs), and peripheral clocks (PCLKs). You can use a combination of these low-skew signals to route clocks and other high fan-out signals in your design.

In addition to the clock networks described previously, Stratix GX series devices feature separate clock distribution resources that connect directly to the clocking resources of the device logic array. This architecture makes it flexible for reference clock generation, clock domain translation, and support of multi-channel functionality.

 The Quartus II software turns off unused clock networks to reduce power dissipation in the device. It can also disable unused portions of clock networks for Stratix II, Stratix III, and Stratix IV devices, and can optimize the placement to minimize the portion of the clock network used to further reduce power.

 For more information about regional clocks and fast regional clocks, refer to the [Area and Timing Optimization](#) chapter in volume 2 of the *Quartus II Handbook*.

 For more information about the clocking resources such as GCLK, PCLK, and RCLK within a specific device, refer to the appropriate device handbook.

For more information about clock tree synthesis and clock requirements, refer to [“Clock Tree Synthesis” on page 23](#).

Phase-Locked Loop (PLL) Requirements

Inserting PLL circuitry into a traditional standard cell ASIC device is typically a manual process in which you instantiate special PLL blocks in the design. In the Altera FPGA flow, you can create configurable PLLs with the MegaWizard® Plug-In Manager, available in the Quartus II software. You can control PLL parameters such as phase shift, clock switchover, and PLL bandwidth using the MegaWizard Plug-In Manager. In the PLLs available in Altera devices, each output of the PLL can be programmed independently, creating customizable clock frequencies that are independent of other input or output clocks. Inherent jitter filtration and fine-grained control of the configurable range help you generate the high-performance precision clocks required in your system. The number of PLLs available in FPGA technology is usually limited, whereas PLL quantities are virtually unlimited in traditional standard cell ASIC technology.

 For more information about the number and type of PLLs supported in a specific device, refer to the appropriate chapter on *Clock Networks and PLLs* in the Altera device handbooks.

Clock Frequencies

The frequency at which your design is expected to operate is an important factor in choosing the appropriate Altera FPGA for your design. The maximum frequency of operation is affected by several factors, such as logic utilization, routing congestion, and the speed grade of the chosen device.

- Refer to the appropriate Altera device handbook to determine the maximum clock frequency for a given device and speed grade.

Number of Simultaneously Switching Outputs (SSOs)

The number and the placement of SSOs has a direct impact on the number of power and ground pins required for a traditional standard cell ASIC. This is usually not an issue with an FPGA, because the placement and number of power and ground pins is pre-determined. The Quartus II software helps you with placement of user pins. You can check the pin placement based on the I/O rules with I/O assignment analysis.

Power Requirements

The Quartus II software supports two power analysis methods: the PowerPlay Early Power Estimator (EPE), which is a spreadsheet-based tool, and the PowerPlay Power Analyzer, which is a part of the Quartus II software. Performing preliminary power analysis, based on the estimated logic size and speed, helps you determine the device power requirements. This usually leads to defining the device's cooling and package requirements.

When you design a traditional standard cell ASIC, you might have to use stand-alone power analysis tools to estimate the power dissipation in your design, and a good estimate might not be available until late in the design cycle. When you target your design to an Altera FPGA, you can use the PowerPlay EPE to get an early estimate of power dissipation in your design. You can run the more elaborate PowerPlay power analysis later in the design cycle. You can also optimize synthesis and the Fitter to focus on reducing power consumption by turning on the **Extra Effort PowerPlay** option in power driven compilation.

- The Quartus II PowerPlay Power Analyzer helps you get fast and accurate power consumption in your design. For more details about using this tool, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.
- For more details about power driven synthesis and other power saving optimization techniques, refer to the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Design Development

Design development includes the following steps for both FPGAs and traditional standard cell ASICs:

- Top-down or bottom-up methodology selection
- RTL coding
- Specification of the external and internal memory
- Synthesis

Methodology Selection

Altera's FPGA design methodology supports both top-down and bottom-up design methodology. FPGA design flows support modular design approaches for bottom-up methodology and hierarchical design partitioning for top-down design methodology, similar to the process used for traditional standard cell ASIC devices. Altera's design software also supports newer standards such as SystemVerilog that are becoming a part of traditional standard cell ASIC design methodology. For team-based designs that are typical of ASIC-size designs, the Quartus II software supports incremental compilation methodology. For designs using HardCopy ASICs to achieve higher performance than the companion FPGA, it is recommended that you use the HardCopy first flow.

Incremental Compilation

In an incremental compilation flow, you can split a large design into smaller partitions. Team members can work on partitions independently, which can simplify the design process and reduce compilation time. The Quartus II incremental compilation feature preserves the results and performance for unchanged logic in your design as you make changes elsewhere, allowing you to perform more design iterations per day and achieve timing closure more efficiently.

If you want to take advantage of the compilation time savings and performance preservation of Quartus II incremental compilation, plan for an incremental compilation flow from the beginning of your design cycle. Good partition and floorplan design helps lower-level design blocks meet top-level design requirements, reducing the time spent integrating and verifying the timing of the top-level design.

 For more information about design planning and different design approaches, refer to the *Design Planning with the Quartus II Software* and the *Quartus II Incremental Compilation for Hierarchical and Team Based Design* chapters in volume 1 of the *Quartus II Handbook*.

 For further information about using different approaches in design methodology when using third-party synthesis tools, refer to the corresponding chapter in the *Synthesis* section of the *Quartus II Handbook*, volume 1.

RTL Coding

RTL coding style becomes important when you target your design to an FPGA because the resources in a device are finite. This application note provides some coding guidelines to improve design performance by taking advantage of the FPGA architecture.

This section provides the following guidelines for the RTL coding of your design when you target an Altera FPGA:

- Synchronous design practices versus asynchronous designs
- Synchronous versus asynchronous resets
- Gated clocks versus clock enables
- Divided clocks
- Using data pipelining

- Using encoding schemes
- Using look-ahead techniques
- Using logic duplication
- Using internal buses

 Only some of the RTL coding guidelines are discussed here. For more details about coding guidelines, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter and the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Synchronous Design Practices Versus Asynchronous Designs

Use synchronous design practices to avoid glitches and race conditions inherent in asynchronous design. Conforming to a synchronous design style also makes it easier to perform timing analysis and achieve timing closure.

 For a detailed discussion about synchronous design practices, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in the *Synchronous FPGA Design Practices* section in volume 1 of the *Quartus II Handbook*.

Another problem caused by asynchronous signals is that the sampling clock at the receiving end might be too slow compared to the initiating clock. The slow clock at the destination might cause a transition on a signal to be completely missed. Implement a good handshake protocol between asynchronous blocks to ensure proper operation.

The following typical design styles contribute to asynchronous designs:

- Gated clocks
- Latch inferences
- Multiple clocks
- Derived clocks

Synchronous Versus Asynchronous Resets

An asynchronous reset is defined as a way of clearing the contents of a register, independent of the associated clock. Standard cell ASIC libraries consist of registers with and without a built-in reset/clear pin. A register with a built-in reset/clear pin is generally bigger than a register without one.

Sometimes, traditional standard cell ASIC designers use registers without asynchronous reset pins to obtain extra speed and reduce area in a design by using an external gate on the data path of the register for a reset. When the reset is routed through the data pin, the clock must be running when the reset is asserted. Additionally, a synchronous reset signal is treated as any other data signal, so no extra care is required during the routing and timing optimization phase.

In traditional standard cell ASIC designs, internally generated asynchronous resets from a state machine can cause problems in scan testing. A typical problem results from the shifting of test vectors through the flipflops of the state machine, which triggers unintended resets. This is not an issue for FPGAs, which do not have to be scan tested.

You can also control the asynchronous reset from an input pin on the device. Such a reset signal should be buffered like a clock tree to reach all of the asynchronous reset pins of all the device registers. In traditional standard cell ASICs, a reset tree is generated just like a clock tree. You must use static timing analysis or use an independent start signal to prevent different state machines from becoming out-of-sync immediately after the reset release when you are designing a standard cell ASIC.

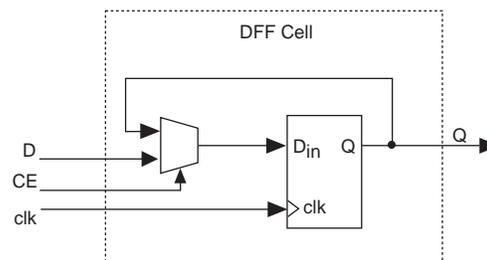
In FPGAs, a reset tree is already in place. All registers have a built-in asynchronous reset, so no area savings is attained by not utilizing the available reset capability.

Gated Clocks Versus Clock Enables

Clock gating is used in traditional standard cell ASIC designs for power optimization. However, gated clocks can introduce glitches and register incorrect data due to the delay on the combinational path.

Registers in FPGAs have a clock enable pin. You can avoid clock gating by using the clock enable pin. Using the clock enable pin to disable the clocking of the flipflop does not mean that the flipflop is not clocked. It means that the current state of the flipflop is clocked continuously. Using this clock enable does not reduce the power consumption of your design. An example of this implementation of the flipflop is shown in [Figure 3](#).

Figure 3. A Clock Enable Pin Disabling Clocking of a Flipflop



The following RTL code segment shows how to infer a flipflop with the clock enable shown in [Figure 3 on page 12](#):

```
module clock_en(in,out,clk,data_en);
input in,clk,data_en;
output out;
reg out;
always @(posedge clk) begin
if (data_en)
out <= in;
else
out <= out;
end
endmodule
```

 Refer to the appropriate device datasheet for device specific information about clocking structures.

If reducing the power is the primary reason for using clock gating in your design, you can use dedicated resources available in Altera devices to perform clock gating. Altera FPGAs from the Cyclone III, Cyclone II, Stratix IV, Stratix III, and Stratix II families have dedicated clock control blocks to perform clock gating. You can use these blocks to shut down selected clock networks. Altera recommends that you use the dedicated circuitry when available rather than using a multiplexing logic structure.



Stratix IV and Stratix III devices use Altera's innovative Programmable Power Technology, which lets you select higher performance or lower power on a logic cell basis and reduce the overall power.



For more details about power management with Stratix III devices, refer to the *Stratix III Power Management Design Guide*. For more details about power management for Stratix IV devices, refer to *AN 514: Power Optimization in Stratix IV FPGAs*.



If you must use clock gating, follow the recommended clock gating method described in the "Gated Clocks" section in the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in volume 1 of the *Quartus II Handbook*.

Divided Clocks

Many designs require clocks generated by division of a master clock. If you have to create a divided clock, Altera recommends that you use the dedicated PLL circuitry for clock division. If you use clock division logic, make sure you use synchronous counters or state machines to perform clock division. Using ripple clock division can make timing analysis difficult.



For more details on internally generated clocks, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in volume 1 of the *Quartus II Handbook*.

Using Data Pipelining

Because FPGAs have many registers, you can use the registers for a pipelined architecture to improve the device performance without incurring any area penalty.



When you pipeline your design, make sure you equalize data and control path latency, and modify the testbenches to capture the outputs at the right time.

Using Encoding Schemes

Almost all of today's designs contain a number of state machines. While designing traditional standard cell ASICs it is common to use some sort of binary encoded state machine to reduce area. On the other hand, FPGAs are register rich, and using a one-hot encoded state machine to reduce the combinational logic required between different states. Because of this, one-hot encoding generally produces better performance results for your state machines in your designs.

Using Look-Ahead Techniques

Look-ahead techniques force a portion of the large combinational logic function into the previous clock cycle, and force the remaining logic function to be performed in the next clock cycle. This technique, also known as register balancing, balances the levels of logic between registers, resulting in much faster execution. Thus, by splitting the combinational logic over two clock cycles, your logic can run faster without additional latency.

- The Quartus II software can perform register retiming without any changes in the RTL if the corresponding physical synthesis option is set. For more details about setting this option, refer to the “Gate Level Register Retiming” section in the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Using Logic Duplication

You can improve FPGA performance by minimizing the routing delays. High fan-out is one of the sources of routing delays. You can reduce the number of high fan-out registers by logic duplication. The Quartus II software supports logic duplication as part of the netlist optimization. You can choose to do logic duplication in your RTL as well. Some synthesis tools might see the logic replication in the RTL code as redundant, and might optimize the replicated logic. Synthesis attributes might be required to keep the intended replicated logic.

- For further information about the available Physical Synthesis Optimization options, refer to the *Netlist Optimization and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*.
- For more details about register duplication and register preservation attributes for synthesis tools, Refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

Using Internal Buses

Internal buses in traditional standard cell ASIC devices allow various internal modules and external devices to communicate. It is not good design practice to have internal tristate buses. Implement tristate functions only at the I/O level.

If your code contains internal tristate inference, the Quartus II software implements it using multiplexers. However, if your design contains multiple partitions, and you use incremental compilation, the Quartus II software might not be able to infer the correct multiplexing logic due to the lack of visibility. Therefore, it is a good practice not to use internal tristates. Altera FPGAs support tristate buses through the I/O interface to communicate with various on-board devices.

- For more details about coding guidelines, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* and the *Recommended HDL Coding Styles* chapters of the *Quartus II Handbook*, volume 1.

Specification of External and Internal Memory

Memory block sizes and configurations are major considerations for SOPC designs. Performance requirements determine whether to have device memory on- or off-chip. The versatile high-memory bandwidth internal memory can implement a variety of memory functions such as RAM, CAM, FIFO, true dual-port memory, synchronous memory, and asynchronous memory. The amount of internal memory that can be implemented in an Altera FPGA is device dependent.

-  To learn more about the available memory configurations and features available in a specific device, refer to the corresponding device handbook.
-  For more details on using the internal memory as FIFOs, refer to the *Single- and Dual-Clock FIFO Megafunction User Guide*.

Memory Implementation—Flexibility and Efficiency

If a memory configuration does not fit in a single memory block, the Quartus II software implements the required configuration by combining two or more memory blocks. For example, if the design requires a 512×18 -bit memory block, but the only blocks available are 512×9 -bit, the Quartus II software can combine two 512×9 -bit blocks.

-  For more information about memory configurations and the Quartus II software settings, see the Altera website and *AN 207: TriMatrix Memory Selection Using the Quartus II Software*.

Instantiating Altera RAM in Place of a Standard Cell ASIC RAM

In traditional standard cell ASIC design flow, memory is instantiated using models provided by the foundry.

The memory model provided by the foundry contains timing information for synthesis and a behavioral model for simulation. If the design has to be ported to an Altera FPGA, you must configure the memory using the MegaWizard Plug-In Manager. The wizard generates all the files required for synthesis and simulation of the memory blocks.

Timing information for the memory models, also known as clear box models, can be generated by the Quartus II software. Third-party synthesis tools can perform an accurate area and timing estimate with the clear box models generated by the Quartus II software.

-  For a more detailed discussion of RTL coding for efficient mapping of memory in Altera devices, refer to the *Design Planning with the Quartus II Software* chapter in volume 1 of the *Quartus II Handbook*.

-  When designing with Altera FPGAs, you must select the device during the design planning process, because the available memory resource varies according to the family and device.

The Stratix series of devices support only synchronous memory interfaces. However, it is possible to operate these synchronous memories in a pseudo-asynchronous mode if the design already contains asynchronous references to memory blocks.

 For more information about synchronous memories, refer to [AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs](#).

External Memory Interfaces

For many of today's high-memory bandwidth applications, the memory must reside outside the standard cell ASIC or FPGA to provide the required buffer size. The Altera IP MegaStore has pre-verified memory controllers that make it easy for you to design a system with external memory. Altera's complete memory interface design solutions address today's high-speed memory interface challenges such as memory controller, I/O design, and board level signal integrity issues. Altera's solutions include advanced FPGA architectures, customizable MegaCore[®] functions, Quartus II design software, reference designs, demonstration boards, and simulation models. Altera supports interfaces to SDR SDRAM, DDR SDRAM, DDR2 SDRAM, DDR3 SDRAM and reduced latency DRAM (RLDRAM) with clock frequencies of up to 400 MHz, achieving a throughput up to 1600 Mbps.

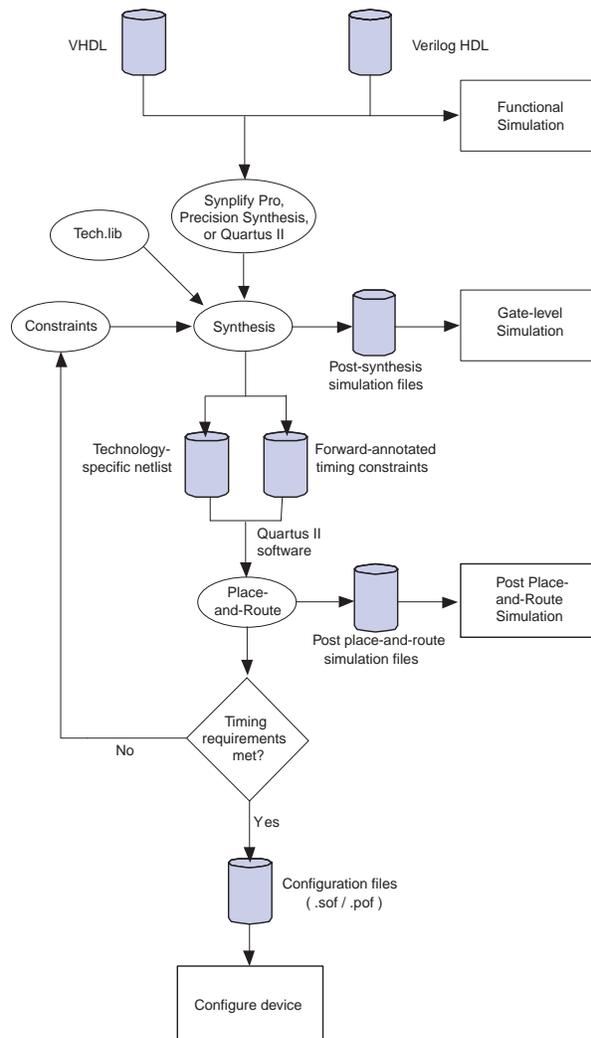
 For more details about different external memory interfaces available with different Altera FPGA families, refer to the documentation and reference designs at [Altera's External Memory Solutions Center](#).

Synthesis

Synthesis is the process of converting a design representation from RTL code to a gate-level netlist. The Quartus II software has an integrated synthesis engine. Altera also supports the use of third-party synthesis tools such as Synplify Pro and Precision RTL from Mentor Graphics. [Figure 4](#) shows a typical synthesis flow. Compared to traditional standard cell ASIC tools, FPGA synthesis tools are much easier to use in terms of complexity and scripting. These tools support all of the popular standard cell ASIC synthesis techniques, including the following techniques:

- Top-down or bottom-up approach
- Modular design flow
- Scripting

 For more information about design methodologies with different synthesis tools supported in the Altera design flow, refer to the [Synthesis](#) section in volume 1 of the *Quartus II Handbook*.

Figure 4. Typical Synthesis Design Flow

Third-Party EDA Tool Support for Synthesis

In addition to the Quartus II integrated synthesis engine, you can use the following third-party tools to perform synthesis:

- Synplify Pro and Synplify from Synopsys' Synplicity Business Group
- LeonardoSpectrum and Precision Synthesis from Mentor Graphics
- Design Compiler from Synopsys

 For more details about using third party synthesis tools, refer to the corresponding chapter in the *Synthesis* section in volume 1 of the *Quartus II Handbook*.

Design Development Tools

Altera provides you with the following development tools, all of which are available within the Quartus II software:

- SOPC Builder
- DSP Builder
- MegaWizard Plug-In Manager
- C2H Compiler

You can use these tools to ease SOPC design development and potentially reduce time-to-market. The following sections present an overview of these tools, and their advantages.

SOPC Builder

SOPC Builder provides a standardized, graphical environment for creating SOPC designs composed of components including CPUs, memory interfaces, standard peripherals, and user-defined peripherals.

SOPC Builder enables the combination of components such as embedded processors, standard peripherals, IP cores, on-chip memory, interfaces to off-chip memory, and user-defined logic into a custom system module. This tool generates a single system module that instantiates these components, and automatically generates the necessary bus logic to connect them. SOPC Builder also generates RTL design files, which you can use for the functional simulation of your system.

Figure 5 shows an example of a typical set of system components with which SOPC Builder can be used to generate a system-level module in minutes. SOPC Builder also generates the test bench.

Figure 5. Elements of a Typical SOPC Design

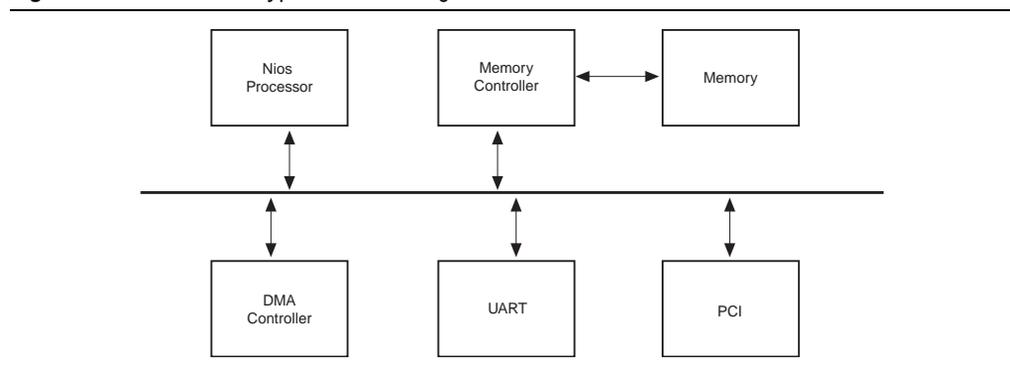


Figure 5 shows the SOPC Builder user interface after a set of peripherals are connected to a Nios[®] processor.

 For more details about the SOPC Builder, refer to *SOPC Builder* in volume 4 of the *Quartus II Handbook*.

 You can get more details about various **IP Megafunctions** available on the Altera.

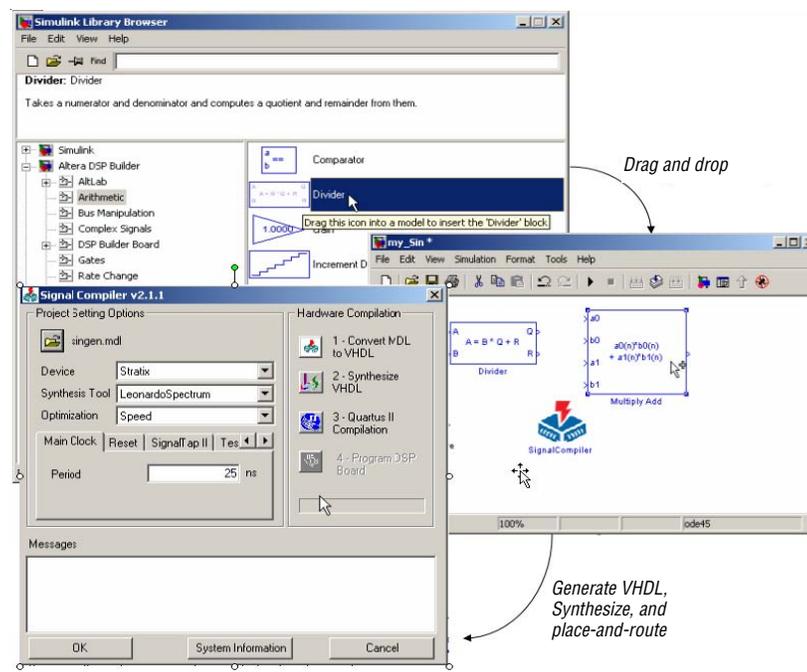
DSP Builder

Altera FPGAs have digital signal processing (DSP) blocks, such as dedicated multiplier blocks, and support embedded processors. These features make Altera FPGAs very suitable for use in such applications. In addition, the Altera design flow provides tools that make designing for DSP much easier.

The Altera DSP portfolio consists of proven, high-performance, standard algorithms and functions created to meet today's rapidly evolving technologies. Every function in Altera's MegaCore function library has been rigorously tested and meets the exacting requirements of various industry standards. Altera provides an extensive portfolio of drop-in DSP functions. The DSP portfolio includes everything you require to build SOPC solutions. You can choose blocks of IP from a comprehensive range of standard DSP functions to create a complete SOPC solution. You can instantiate each function multiple times in different designs. The IP library includes functions such as filters, transforms, encoders, error detection and correction circuits, and video and image processors.

Altera's DSP Builder integrates the Quartus II software with high-level algorithmic development tools such as MATLAB and Simulink software. The DSP Builder software helps you create the hardware representation of a DSP design in an algorithm-friendly development environment. Figure 6 shows a simple design, built from concept-to-implementation, using the DSP Builder. Several ready-to-use mathematical functions are available within both the MATLAB and Simulink tools, along with simulation models, which can be used to create a schematic. DSP Builder can generate a RTL description of the design, along with the test bench.

Figure 6. DSP Builder Design Flow



The MegaWizard Plug-In Manager

The MegaWizard Plug-In Manager helps you create or modify design files that contain custom megafunction variations, which can then be instantiated in a design file.

C2H Compiler

With the evolution of HDLs, digital systems are being described at increasingly higher levels of abstraction. Newer standards such as SystemVerilog and SystemC are offshoots of this development. Many traditional standard cell ASIC designers now model their designs in a high-level language such as C for verification of the architectural design and model implementation. Several design tools are now available to help you to take your concept from a high-level language to hardware implementation, thereby reducing the design cycle.

To support the high-level design flow, Altera provides the C2H Compiler as part of the Nios II Embedded Design Suite. This tool creates custom hardware for functionality that would otherwise require processor usage. Creating a dedicated logic in hardware can improve the execution performance. You can decide on the blocks required for the acceleration requirement you may decide to generate. The C2H Compiler takes code written in ANSI-style C and maps it into resources in an Altera FPGA.



For more details on using the Nios II C2H Compiler, refer to the [Nios II C2H Compiler User Guide](#).

IP Availability and Flow

IP blocks that are pre-verified reduce the design time, solve many time-to-market issues, and simplify verification.

Altera has an extensive offering of specialty IP cores which you can incorporate directly into your design. These include embedded processors, specialty communications interfaces, and memory controllers. These pre-verified cores help you reduce the time to market. These IP cores are designed to take advantage of Altera's device architecture, thus ensuring optimal fit results. You can also find many reference designs using these IP cores on Altera's website.



For a list of IP cores available for Altera devices, go to www.altera.com/literature/lit-ip.jsp.

Altera delivers IP cores as a single MegaCore® IP library package that is built into the Quartus II software download file and installation process. The MegaWizard Plug-In Manager allows you to quickly and easily view documentation, specify parameters, set up third-party tools, and generate the files necessary for integrating the parameterized Altera IP core into a design.



For more information about the IP design flow, refer to the [Designing With Altera Intellectual Property](#) page on the Altera website.

Functional Simulation

Functional simulation verifies the functionality of the RTL design. The following third-party EDA tools are supported:

- NC-Sim from Cadence
- VCS and VCS-MX from Synopsys
- ModelSim[®] from Mentor Graphics[®]
- Active-HDL and Riviera-PRO from Aldec

 When designing ASICs, you may run gate-level simulation and/or static timing analysis on a synthesized netlist even before you perform place and route. However, for Altera FPGA designs, run static timing analysis with the TimeQuest Timing Analysis only after place and route. Refer to “[Gate-Level Simulation](#)” on page 22 and “[Static Timing Analysis](#)”. You can generate a functional simulation netlist that does not contain any timing information if you want to run a functional simulation of the design after place and route to use with your RTL simulation test bench.

 For additional and detailed information about performing functional simulation with different industry standard simulators, refer to the [Simulation](#) section in volume 3 of the *Quartus II Handbook*.

Test Synthesis

The manufacturer tests its FPGA devices for manufacturing defects, eliminating the memory BIST, SCAN insertion, or other tests requirements typically used to detect manufacturing faults in a traditional standard cell ASIC. This completely eliminates the complex task of test synthesis.

Place-and-Route

Use the Quartus II Fitter to place-and-route your design targeted to an Altera FPGA. The Quartus II software reads standard EDIF, VHDL, and Verilog HDL netlist files, and generates VHDL and Verilog HDL netlist files for a convenient interface to other industry-standard EDA tools.

The Quartus II software can perform various netlist optimizations during the Fitter stage to meet performance and area requirements.

 For more details about optimization techniques available in Quartus II for the place-and-route stage of design compilation, refer to the [Area and Timing Optimization](#) chapter in volume 2 of the *Quartus II Handbook*.

The Chip Planner in the Quartus II software allows you to assign logic to a specific location or range of locations within a device or logic cell (including embedded and I/O cells). The Chip Planner also lets you view logic placement made by the Fitter and/or by user assignments, make and view LogicLock region assignments, and view critical path information, physical timing estimates, and routing congestion.

Post Place-and-Route Verification

Typical design verification tasks after place-and-route include static timing analysis, logic equivalence checking, and power estimation. The Quartus II software supports various third-party EDA tools to perform simulation and formal verification tasks.

Gate-Level Simulation

The Quartus II software can generate a netlist to run simulation with industry standard simulators and has the option of generating either of the following netlists:

- Verilog HDL or VHDL netlist to perform functional simulation—You can generate a functional simulation netlist that does not contain any timing information, and use this netlist to simulate your design on any of the supported industry standard simulators.
- Verilog HDL or VHDL netlist with timing information to perform timing simulation—You can generate a netlist with all the timing information included in the Standard Delay File (.sdf). With this netlist, you can run a gate-level timing simulation of your design using any of the supported industry standard simulators.

 For additional and detailed information about performing gate-level simulation with different industry standard simulators, refer to the *Simulation* section in volume 3 of the *Quartus II Handbook*.

Static Timing Analysis

Altera provides the TimeQuest Timing Analyzer, an ASIC strength SDC-based timing analysis tool that is part of the Quartus II software. In addition to the TimeQuest Timing Analyzer, you can also use PrimeTime from Synopsys, which is the industry-standard STA sign-off tool used for standard cell ASICs. If you select the PrimeTime EDA tool for timing analysis, the Quartus II software writes the Verilog HDL or VHDL gate-level netlist (.sdf) and the Tcl script to run static timing analysis in PrimeTime. In addition, PrimeTime libraries are shipped with the Quartus II software.

 For more details about using Synopsys PrimeTime for your FPGA designs, refer to the *Synopsys PrimeTime Support* chapter in volume 3 of the *Quartus II Handbook*.

 For additional information about static timing analysis with TimeQuest, refer to the *Quartus II TimeQuest Timing Analyzer* and *Switching to the Quartus II TimeQuest Timing Analyzer* chapters in volume 3 of the *Quartus II Handbook*.

Formal Verification

Formal verification is widely used to verify traditional standard cell ASIC designs. By running formal verification in conjunction with static timing analysis, you can confirm that the post-route netlist is the same as the RTL design in functionality. This reduces the need to run resource intensive and time-consuming gate-level simulations.

Altera design flow supports industry-standard Cadence Conformal LEC for formal verification. If you use Quartus II integrated synthesis to synthesize your design, you can verify the equivalence between the RTL and post-fit design using Conformal LEC. If you use Synplify Pro to synthesize your design, you can verify the equivalence between the `.vqm` netlist and the Quartus II post-fit netlist using Conformal LEC.

 For additional information about formal verification support, see the *Formal Verification* section in volume 3 of the *Quartus II Handbook*.

Power Estimation

As design and device sizes grow, power becomes a very important consideration in system design. The Altera FPGA design flow supports several methods to estimate power at different stages of the design. These estimations help you determine the power consumption accurately and develop the right power budget for your system. Use the PowerPlay EPE to calculate the estimated power consumption based on typical conditions.

 For more information about the PowerPlay Early Power Estimator, go to www.altera.com/support/devices/estimator/pow-powerplay.jsp.

The PowerPlay Power Analyzer, which is typically run in a later stage in the design cycle, uses simulation data to accurately predict the power dissipation. By choosing simulation data that is a true representation of the design's operation, you can predict the power consumption fairly accurately.

The Stratix III device family of devices provides you with an architecturally advanced, high-performance solution together with low power consumption. In these devices, you can choose to increase performance in regions where you require it, and decrease power consumption everywhere else. With architecturally advanced features and Programmable Power Technology, Stratix III devices are a good alternative to traditional standard cell ASICs.

 For more details about running the PowerPlay Early Power Estimator and PowerPlay Power Analysis, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*. Additional information on power estimation is also available in the Quartus II software Help.

Clock Tree Synthesis

Clock tree synthesis (CTS) is an important step in the traditional standard cell ASIC design methodology and is performed after placement. CTS builds a clock network to reduce the clock skew between registers in the design. CTS synthesis is performed either by third-party EDA tools that have the capacity to perform physical synthesis, or by the foundry tools. CTS can take a great deal of time, and also might require several iterations before the required clock skew is achieved. Because Altera FPGAs include pre-built, low-skew networks, you do not have to perform CTS.

Test Methodology

Testing and fault coverage is an important aspect of the traditional standard cell ASIC development process. In an ASIC design flow, all the following must be considered and analyzed: scan insertion, built-in self-test (BIST), signature analysis, IDDQ, and automatic test pattern generation (ATPG). ASIC testing typically involves test vector generation, using ATPG tools to test the device for manufacturing defects under the “single stuck-at” model. ATPG is usually performed after completion of scan insertion. Scan insertion is performed to improve the obtained fault coverage by reducing the sequential testing problem to a combinational testing problem. Using FPGA technology, you do not have to worry about device testing, because FPGA devices are pretested at the factory. Therefore, compared to standard cell ASIC devices, FPGA devices are much more likely to be free of manufacturing defects.

System designers use boundary scan testing to ensure pin connectivity and functionality on a board. In traditional standard cell ASIC design flow, you have to spend considerable manual effort to insert and simulate boundary scan logic. Boundary scan insertion normally requires the use of third-party EDA tools. However, boundary scan logic (JTAG tap controller) is built into all Altera FPGAs. You do not have to use any third-party tools or do additional work to access the boundary scan logic available on the device.

Scripting

Traditionally, many standard cell ASIC designers use scripts to run their compilation, simulation, synthesis, and verification. Altera’s design software supports the use of scripts. The TimeQuest Timing Analyzer uses scripts written in Tcl and allows you to specify timing constraints in SDC format. FPGA synthesis tools such as Synplify and Synplify Pro, and Precision Synthesis support scripts written in Tcl and SDC format. You can generate a Tcl Script File (.tcl) from an existing project created with the Quartus II software, or you can use Quartus II Tcl templates to create Tcl scripts. You can run Tcl scripts or individual Tcl commands from within the Quartus II software and other EDA software tools.



For more details about using Tcl Scripting with Altera’s design flow and Quartus II software, refer to the [Command-Line Scripting](#) and [Tcl Scripting](#) chapters in volume 2 of the *Quartus II Handbook*.

Quartus II Software On-Chip Debugging Features

When you design a standard cell ASIC, you might have limited options for on-chip debugging. The number of test nodes on a standard cell ASIC depends on the spare I/O on the chip. Also, once the chip is fabricated, you have no way to bring out any more test points. However, when you design with an Altera FPGA, you have a number of on-chip debugging options for system-level debugging. Based on your requirements, you can use one of the following tools:

- SignalTap[®] II Embedded Logic Analyzer
- SignalProbe[™] Incremental Routing
- Logic Analyzer Interface
- In-System Memory Content Editor

- In-System Sources and Probes Editor

SignalTap II Embedded Logic Analyzer

In traditional standard cell ASIC devices, test points and pins are used to probe various nodes to help identify the source of a problem. With these test points/pins and a logic analyzer, you can effectively determine the source of most problems. However, standard cell ASICs are less flexible to troubleshoot because once a chip is fabricated, you cannot bring out additional internal signals to the pin level. Any logic for test modes, and assignment of specific signals to the I/Os, must be designed beforehand.

The SignalTap II Embedded Logic Analyzer captures the data of internal nodes and transfers the data in real time, via a download cable, to the Quartus II software. This data transfer takes place because the SignalTap II Embedded Logic Analyzer supports trigger positions and trigger events, including the device's power-up sequence, in the same way that a standard external logic analyzer does.

Because the SignalTap II Embedded Logic Analyzer uses the programming interface to communicate with the computer, you do not require extra test pins. Acquired data is saved to the device's internal RAM and then streamed off-device via the JTAG communication port. This is advantageous when BGA packages are used and access to pins is difficult, or impossible.

SignalProbe Incremental Routing

The SignalProbe feature gives you quick access to internal design signals for system-level debugging. You can use a logic analyzer to probe these internal signals.

The SignalProbe feature supports incremental routing, and allows you to route signals to I/O pins for efficient signal verification without affecting the rest of the design. When device memory is limited and there is no access to a JTAG communication port, you can perform signal debugging and hardware verification using the SignalProbe feature.

The Quartus II software lets you select the nodes to be routed to pre-specified SignalProbe pins. You can then use a logic analyzer to analyze your selected internal nodes.

Logic Analyzer Interface

Using this interface, you can connect and transmit FPGA internal signals to an external logic analyzer. You can use this feature to connect a large number of internal signals to a small number of output pins for debugging.

In-System Memory Content Editor

This feature provides read and write access to FPGA memories through the JTAG interface. This makes it easy to test changes to memory contents when the device is functioning in a system.

In-System Sources and Probes Editor

You can use the In-System Sources and Probes editor to set up register chains to drive or sample instrumented nodes in your design. You can input virtual stimuli and capture the value of instrumented nodes. Thus, you can create simple test vectors to exercise your design without external test equipment.

- For more information about the system-level debug options for Altera FPGAs, refer to the *In-System Design Debugging* section in volume 3 of the *Quartus II Handbook*.

Device Programming

After a standard cell ASIC returns from the foundry, you cannot change the functionality of the device. However, FPGA devices can be reconfigured many times, providing more flexibility. The ability to change the functionality of a device and add enhancements and fixes is a very useful feature during the prototyping stages.

Altera FPGAs can be configured using a download cable or by using a configuration device. A download cable is used more frequently during prototyping stages. Using a download cable makes design changes very simple and fast, but the board must be reconfigured using the Quartus II software every time the power is applied to the board. There are various download cables you can use to connect to the serial, USB, or parallel port of your computer, and each download cable supports different configuration methods.

In systems where it is not practical to configure a device using a download cable, Altera FPGAs can be configured using flash-based or EEPROM-based configuration devices that store the configuration data and configure the FPGA. When the board is powered on, the configuration device sends the configuration data from memory to the FPGA device.

- For more information about configuration using the download cable, refer to volume 1 of the *Configuration Handbook* and the appropriate device handbook.
- For a comprehensive list of configuration options available for different device families, refer to the *Configuration Center*.

HardCopy ASIC Devices

Altera's low power, cost-effective HardCopy ASICs are supported by Stratix series seamless prototyping—delivering the benefits of FPGAs in design and the benefits of ASICs in production. Stratix series FPGAs enable software and hardware co-design, which cuts 9 to 12 months from your design cycle. The low NRE, coupled with the fast and predictable turnaround time for HardCopy ASICs, delivers your products to market faster than any other methodology.

Altera's Quartus II software allows you to create one design, with one RTL and one suite of IP, while delivering two implementations: the FPGA prototype and the HardCopy ASIC. This dramatically reduces total cost and risk in your ASIC projects.

Using HardCopy ASICs, Altera's SOPC solutions can be leveraged from prototype to production, while reducing cost and speeding time-to-market. Unlike traditional standard cell ASIC device development, the HardCopy ASIC design does not require the generation of testbenches, test vectors, or timing and functional simulation. The HardCopy conversion process requires only the Quartus II software-generated output files. Altera HardCopy Design Center performs the conversion and delivers functional prototypes in eight weeks.

- For more information about HardCopy ASICs, and for details about migrating Altera FPGA designs to HardCopy ASICs, refer to the HardCopy device handbooks at www.altera.com.

Conclusion

There are several advantages to using Altera FPGAs or HardCopy ASICs in place of traditional standard cell ASICs in your systems. You can reduce the time spent on prototyping and have more time for volume production. You can start the firmware development early in the design cycle because you can program the FPGA even while your design is not yet complete and bug free. Modifications, to either the firmware or the design, can be accomplished with a short turnaround time. Because devices can be reconfigured, there is no NRE cost associated with design iterations, and design modifications are easy to accommodate—taking minutes as compared to months.

The Quartus II software performs place-and-route and generates all the files necessary to build the FPGA. The Quartus II software provides an easy interface with industry-standard simulation and timing verification EDA tools. Altera also offers the migration from FPGAs to cost effective HardCopy ASIC for high volume applications.

- For more information about Altera products, refer to the [Literature and Technical Documentation](#) page on the Altera website.

Document Revision History

Table 1 shows the revision history for this application note.

Table 1. Document Revision History

Date and Version	Changes Made	Change Summary
April 2009 v3.1	<ul style="list-style-type: none"> Updated “IP Availability and Flow” on page 20 Updated “Functional Simulation” on page 21 Updated “Gate-Level Simulation” on page 22 Updated “Static Timing Analysis” on page 22 Updated “Post Place-and-Route Verification” on page 22 	Restructured sections for better clarity.
June 2008 v3.0	<ul style="list-style-type: none"> Updated to follow the new naming convention for HardCopy ASICs Removed FPGA Financial Benefit section 	—

Table 1. Document Revision History

Date and Version	Changes Made	Change Summary
October 2007 v2.0	Updated sections: <ul style="list-style-type: none"> ■ FPGA Financial Benefit ■ Design Specifications ■ Design Development ■ Design Development Tools ■ Gate-Level Simulation and Timing Analysis ■ Post Place-and-Route Verification ■ Quartus II Software On-Chip Debug Features ■ Device Programming (Prototype) ■ HardCopy Devices ■ Added Referenced Documents 	Updated for the Quartus II software version 7.2.
July 2003 v1.0	Initial release.	



101 Innovation Drive
 San Jose, CA 95134
www.altera.com
 Technical Support
www.altera.com/support

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001