HardCopy II Device Handbook, Volume 2

101 Innovation Drive
San Jose, CA 95134
www.altera.com

H5V2-4.5

Contents

Section II. HardCopy Design Center Migration Process

Chapter 3. Back-End Design Flow for HardCopy Series Devices

Chapter 4. Back-End Timing Closure for HardCopy Series Devices

Chapter Revision Dates

The chapters in this book, *HardCopy Series Handbook, Volume 1*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Design Guidelines for HardCopy Series Devices
Revised: *September 2008*
Part number: *H51011-3.4*

Chapter 2. Power-Up Modes and Configuration Emulation in HardCopy Series Devices
Revised: *September 2008*
Part number: *H51012-2.5*

Chapter 3. Back-End Design Flow for HardCopy Series Devices
Revised: *September 2008*
Part number: *H51019-1.4*

Chapter 4. Back-End Timing Closure for HardCopy Series Devices
Revised: *September 2008*
Part number: *H51013-2.4*

About this Handbook

This handbook provides comprehensive information about the Altera® HardCopy® devices.

How to Contact Altera

For the most up-to-date information about Altera products, refer to the following table.

Contact	Contact Method	Address
Technical support	Website	www.altera.com/support/
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Altera literature services	Email	literature@altera.com
Non-technical (General)	Email	nacomp@altera.com
(SoftwareLicensing)	Email	authorization@altera.com

Note to table:
(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*

Visual Cue	Meaning
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>*.**pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading" Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Section I. General HardCopy Series Design Considerations

This section provides information about hardware design considerations for HardCopy® II devices.

This section contains the following:

- Chapter 1, Design Guidelines for HardCopy Series Devices
- Chapter 2, Power-Up Modes and Configuration Emulation in HardCopy Series Devices

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the complete handbook.

1. Design Guidelines for HardCopy Series Devices

H51011-3.4

Introduction

HardCopy® series devices provide dramatic cost savings, performance improvement, and reduced power consumption over their programmable counterparts. In order to ensure the smoothest possible transfer from the FPGA device to the equivalent HardCopy series device, you must meet certain design rules while the FPGA implementation is still in progress. A design that meets standard, accepted coding styles for FPGAs, adheres easier to recommended guidelines. This chapter describes some common situations that you should avoid. It also provides alternatives on how to design in these situations.

Design Assistant Tool

The Design Assistant tool in the Quartus® II software allows you to check for any potential design problems early in the design process. The Design Assistant is a design-rule checking tool that checks the compiled design for adherence to Altera® recommended design guidelines. It provides a summary of the violated rules that exist in a design together with explicit details of each violation instance. You can customize the set of rules that the tool checks to allow some rule violations in your design. This is useful if it is known that the design violates a particular rule that is not critical. However, for HardCopy design, you must enable all of the Design Assistant rules. All Design Assistant rules are enabled and run by default in the Quartus II software when using the HardCopy Timing Optimization Wizard in the **HardCopy Utilities** (Project menu). The HardCopy Advisor in the Quartus II software also checks to see if the Design Assistant is enabled.

The Design Assistant classifies messages using the four severity levels described in Table 1–1.

Table 1–1. Design Assistant Message Severity Levels (Part 1 of 2)

Severity Level	Description
Critical	The rule violation described in the message critically affects the reliability of the design. Altera cannot migrate the design successfully to a HardCopy device without closely reviewing these violations.
High	The rule violation described in the message affects the reliability of the design. Altera must review the violation before the design is migrated to a HardCopy device.

Table 1–1. Design Assistant Message Severity Levels (Part 2 of 2)

Severity Level	Description
Medium	The rule violation described in the message may result in implementation complexity. The violation may impact the schedule or effort required to migrate the design to a HardCopy series device.
Information only	The message contains information regarding a design rule.

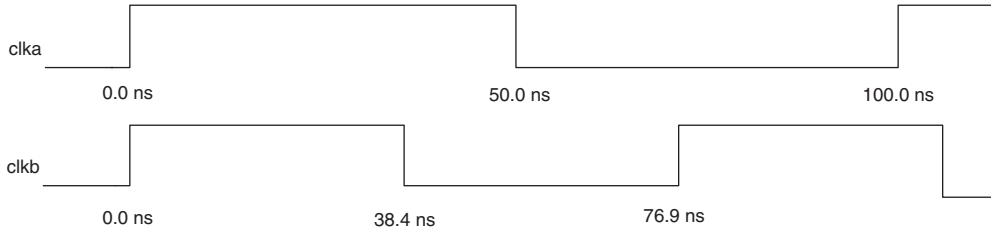
A design that adheres to Altera recommended design guidelines does not produce any critical, high, or medium level Design Assistant messages. If the Design Assistant generates these kinds of messages, Altera's HardCopy Design Center (which performs the migration) carefully reviews each message before considering implementing the FPGA design into a HardCopy design. After reviewing these messages with your design team, Altera may be able to implement the design in a HardCopy device. Informational messages are primarily for the benefit of the Altera HardCopy Design Center and are used to gather information about your design for the migration process from FPGA prototype to HardCopy production device.

Asynchronous Clock Domains

A design contains several clock sources, each driving a subsection of the design. A design subsection, driven by a single clock source is called a clock domain. The frequency and phase of each clock source can be different from the rest.

The timing diagram in Figure 1–1 shows two free-running clocks used to describe the nature of asynchronous clock domains. If the two clock signals do not have a synchronous, or fixed, relationship, they are asynchronous to each other. An example of asynchronous signals are two clock signals running at frequencies that have no obvious harmonic relationship.

Figure 1–1. Two Asynchronous Clock Signals *Notes (1), (2)*



Notes to Figure 1–1:
(1) `clka` = 10 MHz; `clkb` =13 MHz.
(2) Both clocks have 50% duty cycles.

In Figure 1–1, the `clka` signal is defined with a rising edge at 0.0 ns, a falling edge at 50 ns, and the next rising edge at 100 ns (1/10 MHz = 100 ns). Subsequent rising edges of `clka` are at 200 ns, 300 ns, 400 ns, and so on.

The `clkb` signal is defined with a rising edge at 0.0 ns, a falling edge at 38.45 ns, and the next rising edge at 76.9 ns. The subsequent rising edges of `clkb` are at 153.8 ns, 230.7 ns, 307.6 ns, 384.5 ns, and so on.

Not until the thousandth clock edge of `clkb` (1000 × 76.9 = 76,900 ns) or the 7,690th clock edge of `clka` (7,690 × 100 = 769,000 ns), does `clka` and `clkb` have coincident edges. It is very unlikely that these two clocks are intended to synchronize with each other every 76,900 ns, so these two clock domains are considered asynchronous to each other.

A more subtle case of asynchronous clock domains occurs when two clock domains have a very obvious frequency and phase relationship, especially when one is a multiple of the other. Consider a system with clocks running at 100 MHz and 50 MHz. The edges of one of these clocks are always a fixed distance away, in time, from the edges of the other clock. In this case, the clock domains may or may not be asynchronous, depending on what your original intention was regarding the interactions of these two clock domains.

Similarly, two clocks running at the same nominal frequency may be asynchronous to each other if there is no synchronization mechanism between them. For example, two crystal oscillators, each running at 100 MHz on a PC board, have some frequency variations due to temperature fluctuations, and this may be different for each oscillator. This results in the two independent clock signals drifting in and out of phase with each other.

Transferring Data between Two Asynchronous Clock Domains

If two asynchronous clock domains need to communicate with each other, you need to consider how to reliably perform this operation. The following three examples shows how to transfer data between two asynchronous clock domains.:

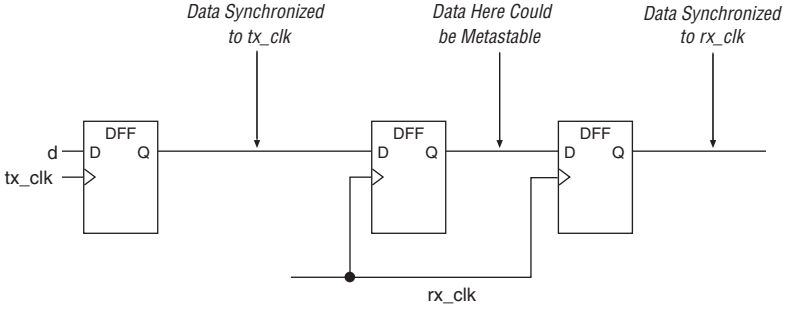
- Using a double synchronizer
- Using a first-in first-out (FIFO) buffer
- Using a handshake protocol

The choice of which to use depends on the particular application, the number of asynchronous signals crossing clock boundaries, and the resources available to perform the cross-domain transfers.

Using a Double Synchronizer for Single-Bit Data Transfer

Figure 1–2 shows a double synchronizer for single-bit data transfer consisting of a 2-bit shift register structure clocked by the receiving clock. The second stage of the shift register reduces the probability of metastability (unknown state) on the data output from the first register propagating through to the output of the second register. The data from the transmitting clock domain should come directly from a register. This technique is recommended only if single-data signals (for example, non-data buses) need to be transferred across clock domains. This is because it is possible that some bits of a data bus are captured in one clock cycle while other bits get captured in the next. More than two stages of the synchronizer circuit can be used at the expense of increased latency. The benefit of more stages is that the mean time between failures (MTBF) is increased with each additional stage.

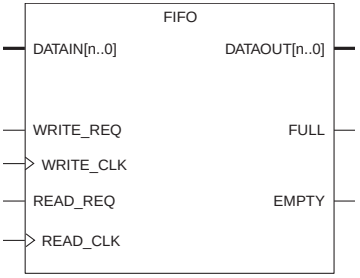
Figure 1–2. A Double Synchronizer Circuit



Using a FIFO Buffer

The advantage of using a FIFO buffer, shown in Figure 1–3, is that Altera's MegaWizard® Plug-In Manager makes it very easy to design a FIFO buffer. A FIFO buffer is useful when you need to transfer a data bus signal across an asynchronous clock domain, and it is beneficial to temporary storage of this data. A FIFO buffer circuit should not generate any Design Assistant warnings unless an asynchronous clear is used in the circuit. An asynchronous clear in the FIFO buffer circuit results in a warning stating that a reset signal generated in one clock domain is not being synchronized before being used in another clock domain. This occurs because a dual-clock FIFO megafunction only has one `aclr` pin to reset the entire FIFO buffer circuit. You cannot remove this warning in the case of a dual-clock FIFO buffer circuit. As a safeguard, Altera recommends using a reset signal that is synchronous to the clock domain of the write side of the FIFO buffer circuit.

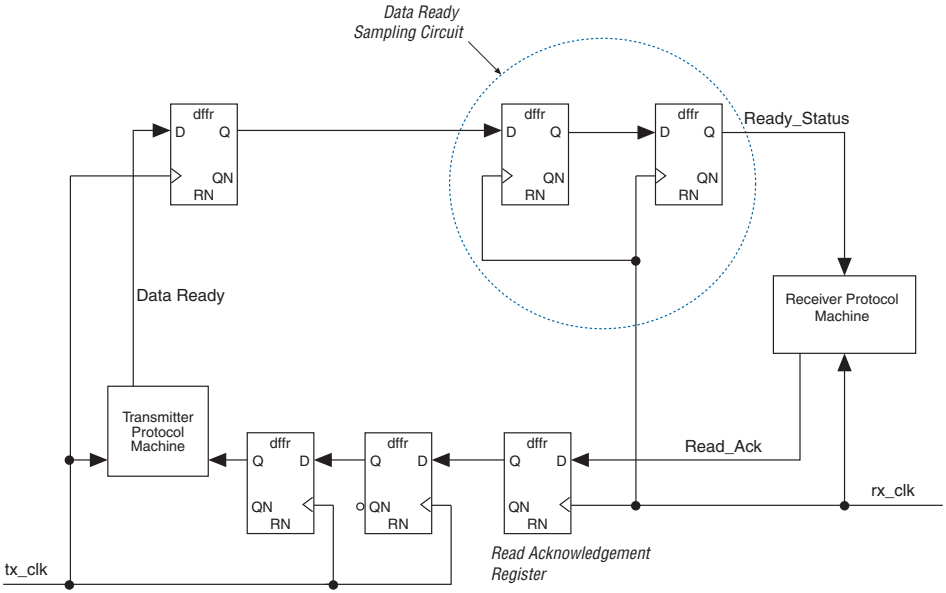
Figure 1–3. A FIFO Buffer



Using a Handshake Protocol

A handshake protocol circuit uses a small quantity of logic cells to implement and guarantee that all bits of a data bus crossing asynchronous clock domains are registered by the same clock edge in the receiving clock domain. This circuit, shown in Figure 1–4, is best used in cases where there is no memory available to be used as FIFO buffers, and the design has many data buses to transfer between clock domains.

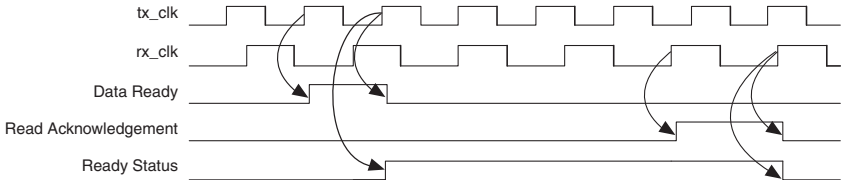
Figure 1–4. A Handshake Protocol Circuit



This circuit is initiated by a data ready signal going high in the transmitting clock domain `tx_clk`. This is clocked into the data ready sampling registers and causes the `Ready_Status` signal to go high. The `Data Ready` signal must be long enough in duration so that it is successfully sampled in the receiver domain. This is important if the `rx_clk` signal is slower than `tx_clk`.

At this point, the receiving clock domain `rx_clk` can read the data from the transmitting clock domain `tx_clk`. After this read operation has finished, the receiving clock domain (`rx_clk`) generates a synchronous `Read_Ack` signal, which gets registered by the read acknowledge register. This registered signal is sampled by the `Read_Ack` sampling circuit in the transmitter domain. The `Read_Ack` signal must be long enough in duration so that it is successfully sampled in the transmitter domain. This is important if the transmitter clock is slower than the receiver clock. After this event, the data transfer between the two asynchronous domains is complete, as shown by the timing diagram in Figure 1–5.

Figure 1–5. Data Transfer Between Two Asynchronous Clock Domains



Gated Clocks

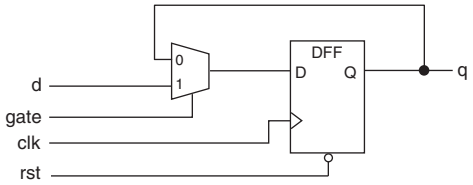
Clock gating is sometimes used to "turn off" parts of a circuit to reduce the total power consumption of a device. The gated clock signal prevents any of the logic driven by it from switching so the logic does not consume any power. This works best if the gating is done at the root of the clock tree. If the clock is gated at the leaf-cell level (for example, immediately before the input to the register), the device does not save much power because the whole clock network still toggles. The disadvantage in using this type of circuit is that it can lead to unexpected glitches on the resultant gated clock signal if certain rules are not adhered to. Rules are provided in the following subsections:

- Preferred Clock Gating Circuit
- Alternative Clock Gating Circuits
- Inverted Clocks
- Clocks Driving Non-Clock Pins
- Clock Signals Should Use Dedicated Clock Resources
- Mixing Clock Edges

Preferred Clock Gating Circuit

The preferred way to gate a clock signal is to use a purely synchronous circuit, as shown in Figure 1–6. In this implementation, the clock is not gated at all. Rather, the data signal into a register is gated. This circuit is sometimes represented as a register with a clock enable (CE) pin. This circuit is not sensitive to any glitches on the gate signal, so it gets generated directly from a register or any complex combinational function. The constraints on the gate or clock enable signal are exactly the same as those on the 'd' input of the gating multiplexer. Both of these signals must meet the setup and hold times of the register that they feed into.

Figure 1–6. Preferred Clock-Gating Circuit



This circuit only takes a few lines of VHDL or Verilog hardware description language (HDL) to describe.

The following is a VHDL code fragment for a synchronous clock gating circuit.

```
architecture rtl of vhdl_enable is
begin
    process (rst, clk)
    begin
        if (rst = '0') then
            q <= '0';
        elsif clk'event and clk = '1' then
            if (gate = '1') then
                q <= d;
            end if;
        end if;
    end process;
end rtl;
```

The following is a Verilog HDL code fragment for a synchronous clock gating circuit.

```
always @ (posedge clk or negedge rst)
    begin
        if (!rst)
            q <= 1'b0;
        else if (gate)
            q <= d;
        else
            q <= q;
    end
```

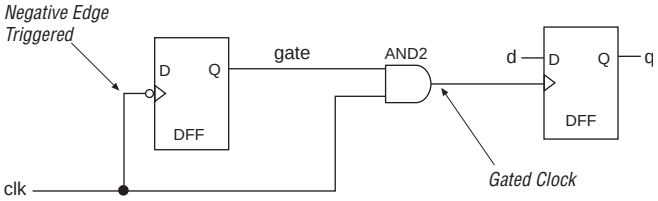
Alternative Clock Gating Circuits

If a clock gating circuit is absolutely necessary in the design, one of the following two circuits may also be used. The Design Assistant does not flag a violation for these circuits.

Clock Gating Circuit Using an AND Gate

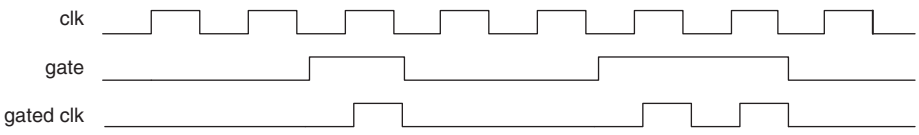
Designs can use a two-input AND gate for a gated clock signal that feeds into positive-edge-triggered registers. One input to the AND gate is the original clock signal. The other input to the AND gate is the gating signal, which should be driven directly from a register clocked by the negative edge of the same original clock signal. Figure 1–7 shows this type of circuit.

Figure 1–7. Clock Gating Circuit Using an AND Gate



Because the register that generates the gate signal is triggered off of the negative edge of the same clock, the effect of using both edges of the same clock in the design should be considered. The timing diagram in Figure 1–8 shows the operation of this circuit. The `gate` signal occurs after the negative edge of the clock and comes directly from a register. The logical AND of this `gate` signal, with the original un-inverted clock, generates a clean clock signal.

Figure 1–8. Timing Diagram for Clock Gating Circuit Using an AND Gate

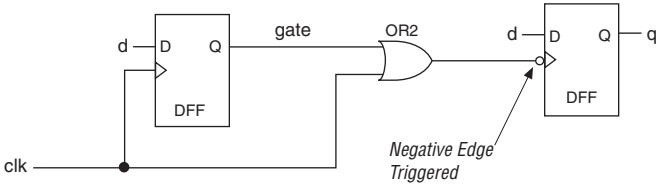


If the delay between the register that generates the `gate` signal and the `gate` input to the AND gate is greater than the low period of the clock, (one half of the clock period for a 50% duty cycle clock), the clock pulse width is narrowed.

Clock Gating Circuit Using an OR Gate

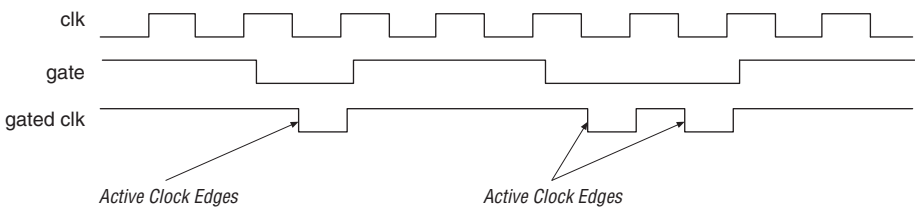
Use a two-input OR gate for a gated clock signal that feeds into a negative-edge-triggered register. One input to the OR gate is the original clock signal. The other input to the OR gate is the gating signal, which should be driven directly from a register clocked by the positive edge of the same original clock signal. Figure 1–9 shows this circuit.

Figure 1–9. Clock Gating Circuit Using an OR Gate



Because the register that generates the gate signal is triggered off the positive edge of the same clock, you need to consider the effect of using both edges of the same clock in your design. The timing diagram in Figure 1–10 shows the operation of this circuit. The `gate` signal occurs after the positive edge of the clock, and comes directly from a register. The logical OR of this `gate` signal with the original, un-inverted clock generates a clean clock signal. This clean, gated clock signal should only feed registers that use the negative edge of the same clock.

Figure 1–10. Timing Diagram for Clock Gating Circuit Using an OR Gate



If the delay between the register that generates the `gate` signal and the `gate` input to the AND gate is greater than the low period of the clock, (one half of the clock period for a 50% duty cycle clock), the clock pulse width is narrowed.

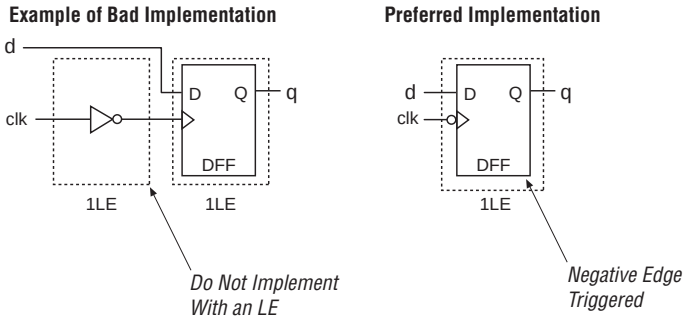
Altera recommends using a synchronous clock gating circuit because it is the only way to guarantee the duty cycle of the clock and to align the clock to the data.

Inverted Clocks

A design may require both the positive edge and negative edge of a clock, as shown in Figure 1–11. In Altera FPGAs, each logic element (LE) has a programmable clock inversion feature. Use this feature to generate an inverted clock.

Do not instantiate a LE look-up-table (LUT) configured as an inverter to generate the inverted clock signal.

Figure 1–11. An LE LUT Configured as an Inverter



Using a LUT to perform the clock inversion may lead to a clock insertion delay and skew, which poses a significant challenge to timing closure of the design. It also consumes more device resources than are necessary. Refer to "Mixing Clock Edges" on page 1–14 for more information on this topic.

Do not generate schematics or register transfer level (RTL) code that instantiates LEs used to invert clocks. Instead, let the synthesis tool decide on the implementation of inverted clocks.

Clocks Driving Non-Clock Pins

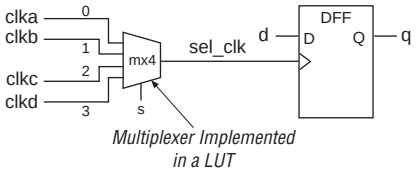
As a general guideline, clock sources should only be used to drive the register clock pins. There are exceptions to this rule, but every effort should be taken to minimize these exceptions or remove them altogether.

One category of exception is for various gated clocks, which are described in "Preferred Clock Gating Circuit" on page 1–7.

You should avoid another exception, when possible, in which you use a clock multiplexer circuit to select one clock from a number of different clock sources, to drive non-clock pins. This type of circuit introduces

complexity into the static timing analysis of HardCopy and FPGA implementations. For example, as shown in Figure 1–12, in order to investigate the timing of the `sel_clk` clock signal, it is necessary to make a clock assignment on the multiplexer output pin, which has a specific name. This name may change during the course of the design unless you preserve the node name in the Quartus II software settings. Refer to the Quartus II Help for more information on preserving node names.

Figure 1–12. A Circuit Showing a Multiplexer Implemented in a LUT



In the FPGA, a clock multiplexing circuit is built out of one or more LUTs, and the resulting multiplexer output clock may possibly no longer use one of the dedicated clock resources. Consequently, the skew and insertion delay of this multiplexed clock is potentially large, adversely impacting performance. The Quartus II Design Assistant traces clocks to their destination and, if it encounters a combinational gate, it issues a gated clock warning.

If the design requires this type of functionality, ensure that the multiplexer output drives one of the global routing resources in the FPGA. For example, this output should drive a fast line in an APEX™ 20KE device, or a global or regional clock in a Stratix® or Stratix II device.

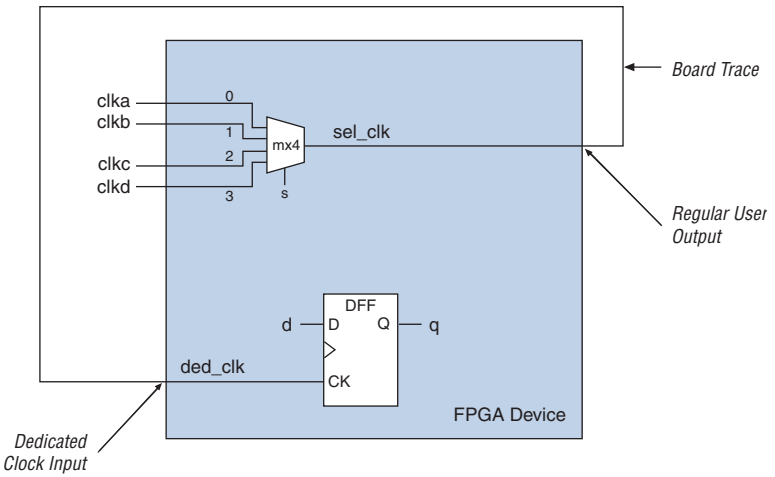
Enhanced PLL Clock Switchover

Clock source multiplexing can be done using the enhanced PLL clock switchover feature in Stratix and Stratix II FPGAs, and in HardCopy Stratix and HardCopy II structured ASICs. The clock switchover feature allows multiple clock sources to be used as the reference clock of the enhanced PLL. The clock source switchover can be controlled by an input pin or internal logic. This generally eliminates the need for routing a multiplexed clock signal out to a board trace and bringing it back into the device, as shown in Figure 1–13.

Routing a multiplexed clock signal, as shown in Figure 1–13, is only intended for APEX 20K FPGA and HardCopy APEX devices. This alternative to a clock multiplexing circuit ensures that a global clock resource is used to distribute the clock signal over the entire device by

routing the multiplexed clock signal to a primary output pin. Outside of the device, this output pin then drives one of the dedicated clock inputs of the same device, possibly through a phase-locked loop (PLL) to reduce the clock insertion delay. Although there is a large delay through the multiplexing circuit and external board trace, the resulting clock skew is very small because the design uses the dedicated clock resource for the selected clock signal. The advantage that this circuit has over the other implementations is that the timing analysis becomes very simple, with only a single-clock domain to analyze, whose source is a primary input pin to the APEX 20K FPGA or HardCopy APEX device.

Figure 1–13. Routing a Multiplexed Clock Signal to a Primary Output Pin



Clock Signals Should Use Dedicated Clock Resources

All clock signals in a design should be assigned to the global clock networks that exist in the target FPGA. Clock signals that are mapped to use non-dedicated clock networks can negatively affect the performance of the design. This is because the clock must be distributed using regular FPGA routing resources, which can be slower and have a larger skew than the dedicated clock networks. If your design has more clocks than are available in the target FPGA, you should consider reducing the number of clocks, so that only dedicated clock resources are used in the FPGA for clock distribution. If you need to exceed the number of dedicated clock resources, implement the clock with the lowest fan-out with regular (non-clock network) routing resources. Give priority to the fastest clock signals when deciding how to allocate dedicated clock resources.

In the Quartus II software, you can use the **Global Signal Logic** option to specify that a clock signal is a global signal. You can also use the auto **Global Clock Logic** option to allow the Fitter to automatically choose clock signals as global signals.

Altera recommends using the FPGA's built-in clock networks because they are pre-routed for low skew and for short insertion delay.

Mixing Clock Edges

You can use both edges of a single clock in a design. An example where both edges of a clock must be used in order to get the desired functionality is with a double data rate (DDR) memory interface. In Stratix II, Stratix, HardCopy II, and HardCopy Stratix devices, this interface logic is built into the I/O cell of the device, and rigorous simulation and characterization is performed on this interface to ensure its robustness. Consequently, this circuitry is an exception to the rule of using both edges of a clock. However, for general data transfers using generic logic resources, the design should only use a single edge of the clock. A circuit needs to use both edges of a single clock, then the duty cycle of the clock has to be accurately described to the Static Timing Analysis tool, otherwise inaccurate timing analysis could result. Figure 1–14 shows two clock waveforms. One has a 50% duty-cycle, the other has a 10% duty cycle.

Figure 1–14. Clock Waveforms with 50% and 10% Duty Cycles

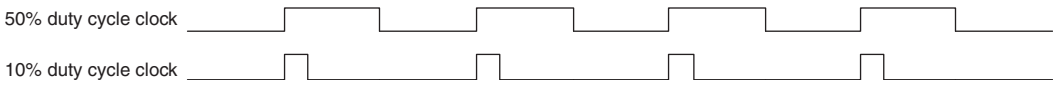


Figure 1–15 shows a circuit that uses only the positive edge of the clock. The distance between successive positive clock edges is always the same; for example, the clock period. For this circuit, the duty cycle of the clock has no effect on the performance of the circuit.

Figure 1–15. Circuit Using the Positive Edge of a Clock

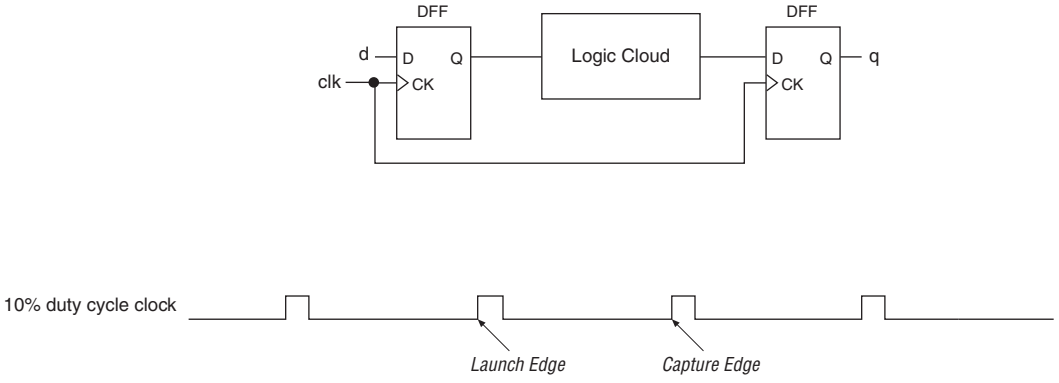
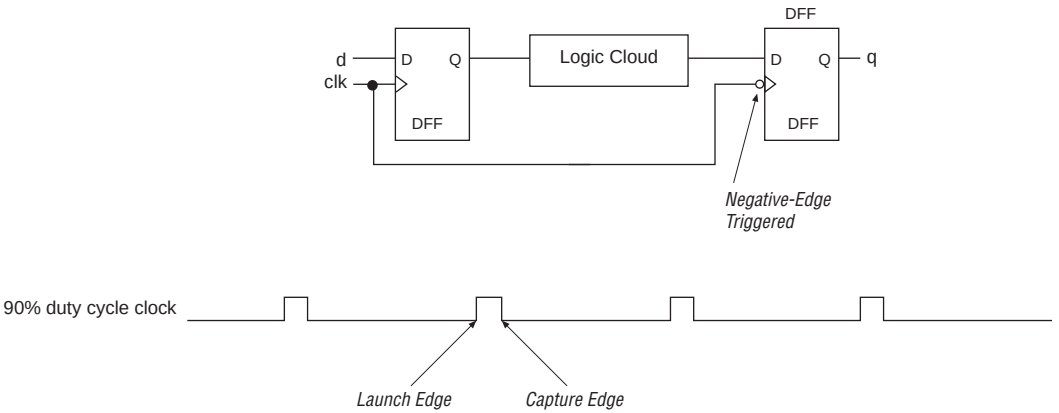


Figure 1–16 shows a circuit that used the positive clock edge to launch data and the negative clock edge to capture this data. Since this particular clock has a 10% duty cycle, the amount of time between the launch edge and capture edge is small. This small gap makes it difficult for the synthesis tool to optimize the cloud of logic so that no setup-time violations occur at the capture register.

Figure 1–16. Circuit Using the Positive and Negative Edges of a Clock



If you design a circuit that uses both clock edges, you could get the Design Assistant warning "Registers are Triggered by Different Edges of Same Clock." You do not get this warning under the following conditions:

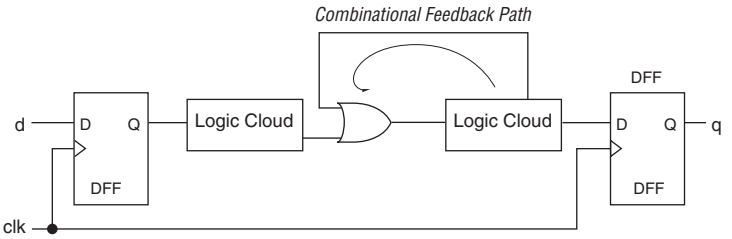
- If the opposite clock edge is used in a clock gating circuit
- A double data rate memory interface circuit is used

Try to only use a single edge of a clock in a design.

Combinational Loops

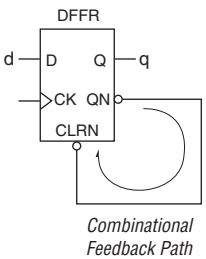
A combinational loop exists (Figure 1–17) if the output of a logic gate (or gates) feeds back to the input of the same gate without first encountering a register. A design should not contain any combinational loops.

Figure 1–17. A Circuit Using a Combinational Loop



It is also possible to generate a combinational loop using a register (Figure 1–18) if the register output pin drives the reset pin of the same register.

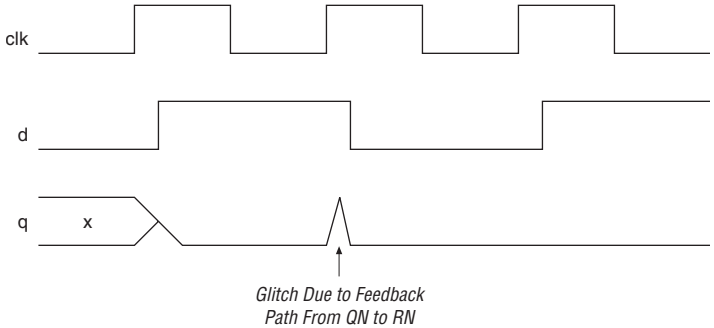
Figure 1–18. Generation of a Combinational Loop Using a Register



The timing diagram for this circuit is shown in Figure 1–19. When a logic 1 value on the register D input is clocked in, the logic 1 value appears on the Q output pin after the rising clock edge. The same clock event causes the QN output pin to go low, which in turn, causes the

register to be reset through RN. The Q register output consequently goes low. This circuit may not operate if there isn't sufficient delay in the QN-to-RN path, and is not recommended.

Figure 1–19. Timing Diagram for the Circuit Shown in Figure 1–18



Combinational feedback loops are either intentionally or unintentionally introduced into a design. Intentional feedback loops are typically introduced in the form of instantiated latches. An instantiated latch is an example of a combinational feedback loop in Altera FPGAs because its function has to be built out of a LUT, and there are no latch primitives in the FPGA logic fabric. Unintentional combinational feedback loops usually exist due to partially specified IF-THEN or CASE constructs in the register transfer level (RTL). The Design Assistant checks your design for these circuit structures. If any are discovered, you should investigate and implement a fix to your RTL to remove unintended latches, or re-design the circuit so that no latch instantiation is required. In Altera FPGAs, many registers are available, so there should never be any need to use a latch.

Combinational loops can cause significant stability and reliability problems in a design because the behavior of a combinational loop often depends on the relative propagation delays of the loop's logic. This combinational loop circuit structure behaves differently under different operation conditions. A combinational loop is asynchronous in nature, and EDA tools operate best with synchronous circuits.

A storage element such as a level-sensitive latch or an edge-triggered register has particular timing checks associated with it. For example, there is a setup-and-hold requirement for the data input of an edge-triggered register. Similarly, there is also a setup-and-hold timing requirement for the data to be stable in a transparent latch when the gate signal turns the latch from transparent to opaque. When latches are built

out of combinational gates, these timing checks do not exist, so the static timing analysis tool is not able to perform the necessary checks on these latch circuits.

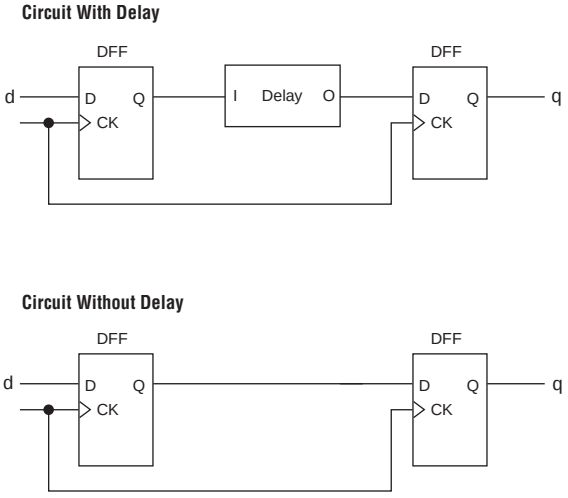
Check your design for intentional and unintentional combinational loops, and remove them.

Intentional Delays

Altera does not recommend instantiating a cell that does not benefit a design. This type of cell only delays the signal. For a synchronous circuit that uses a dedicated clock in the FPGA (Figure 1–20), this delay cell is not needed. In an ASIC, a delay cell is used to fix hold-time violations that occur due to the clock skew between two registers, being larger than the data path delay between those same two registers. The FPGA is designed with the clock skew and the clock-to-Q time of the FPGA registers in mind, to ensure that there is no need for a delay cell.

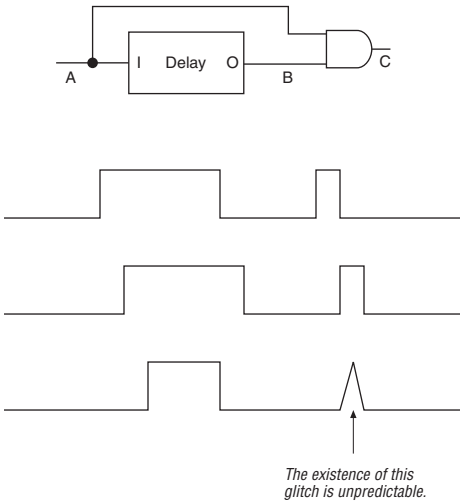
Figure 1–20 shows two versions of the same shift registers. Both circuits operate identically. The first version has a delay cell, possibly implemented using a LUT, in the data path from the Q output of the first register to the D input of the second register. The function of the delay cell is a non-inverting buffer. The second version of this circuit also shows a shift register function, but there is no delay cell in the data path. Both circuits operate identically.

Figure 1–20. Shift Register With and Without an Intentional Delay



If delay chains exist in a design, they are possibly symptomatic of an asynchronous circuit. One such case is shown in the circuit in Figure 1–21. This circuit relies on the delay between two inputs of an AND gate to generate a pulse on the AND gate output. The pulse may or may not be generated, depending on the shape of the waveform on the `A` input pin.

Figure 1–21. A Circuit and Corresponding Timing Diagram Showing a Delay Chain



Using delay chains can cause various design problems, including an increase in a design's sensitivity to operating conditions and a decrease in design reliability.

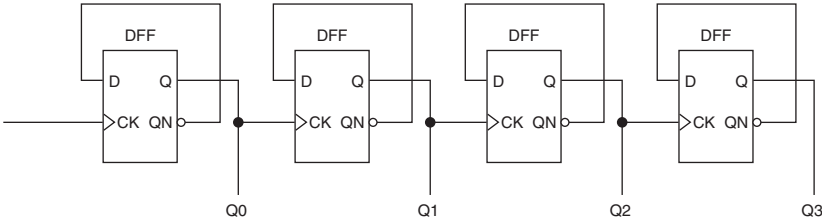
Be aware that not all cases of delay chains in a design are due to asynchronous circuitry. If the Design Assistant report states that you have delay chains that you are unaware of (or are not expecting), the delay chains may be a result of using pre-built intellectual property (IP) functions. Pre-built IP functions may contain delay chains which the Design Assistant reports. These functions are usually parameterizable, and have thousands of different combinations of parameter settings. The synthesis tool may not remove all unused LEs from these functions when particular parameter settings are used, but the resulting circuit is still synchronous. Check all Design Assistant delay chain warnings carefully.

Avoid designing circuits that rely on the use of delay chains, and always carefully check any Design Assistant delay chain warnings.

Ripple Counters

Designs should not contain ripple counters. A ripple counter, shown in Figure 1–22, is a circuit structure where the Q output of the first counter stage drives into the clock input of the following counter stage. Each counter stage consists of a register with the inverted QN output pin feeding back into the D input of the same register.

Figure 1–22. A Typical Ripple Counter



This type of structure is used to make a counter out of the smallest amount of logic possible. However, the LE structure in Altera FPGA devices allows you to construct a counter using one LE per counter-bit, so there is no logic savings in using the ripple counter structure. Each stage of the counter in a ripple counter contributes some phase delay, which is cumulative in successive stages of the counter. Figure 1–23 shows the phase delay of the circuit in Figure 1–22.

Figure 1–23. Timing Diagram Showing Phase Delay of Circuit Shown in Figure 1–22

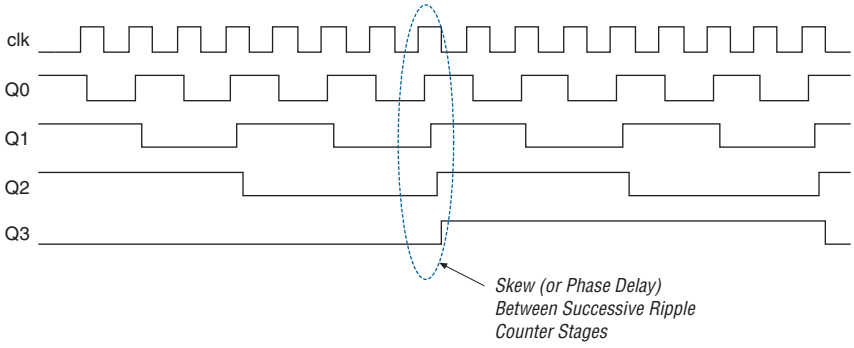
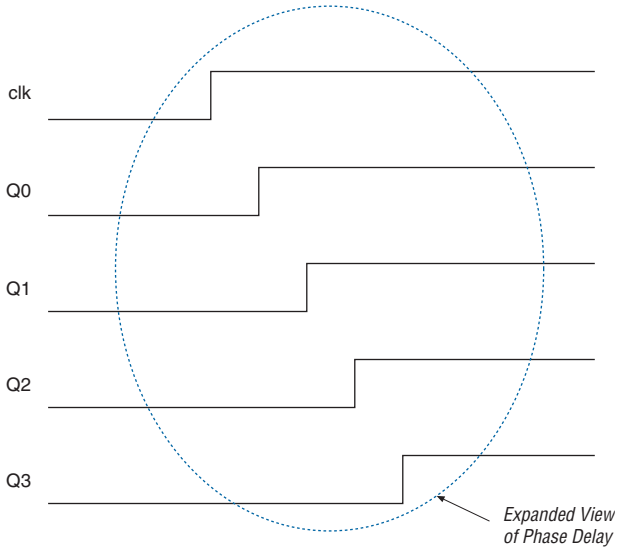


Figure 1–24 shows detailed view of the phase delay shown in Figure 1–23.

Figure 1–24. Detailed View of the Phase Delay Shown in Figure 1–23



This phase delay is problematic if the ripple counter outputs are used as clock signals for other circuits. Those other circuits are clocked by signals that have large skews.

Ripple counters are particularly challenging for static timing analysis tools to analyze as each stage in the ripple counter causes a new clock domain to be defined. The more clock domains that the static timing analysis tool has to deal with, the more complex and time-consuming the process becomes.

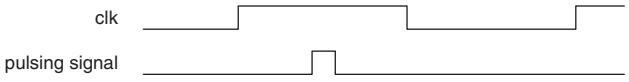
Altera recommends that you avoid using ripple counters under any circumstances.

Pulse Generators

A pulse generator is a circuit that generates a signal that has two or more transitions within a single clock period. Figure 1–25 shows an example of a pulse generator waveform.

For more information on pulse generators, refer to "Intentional Delays" on page 1–18.

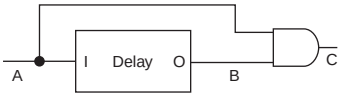
Figure 1–25. Example of a Pulse Generator Waveform



Creating Pulse Generators

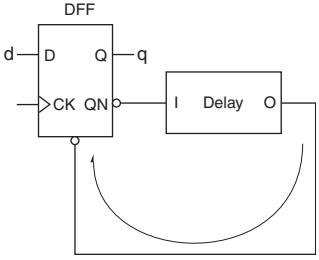
Pulse generators can be created in two ways. The first way to create a pulse generator is to increase the width of a glitch using a 2-input AND, NAND, OR, or NOR gate, where the source for the two gate inputs are the same, but the design delays the source for one of the gate inputs, as shown in Figure 1–26.

Figure 1–26. A Pulse Generator Circuit Using a 2-Input AND



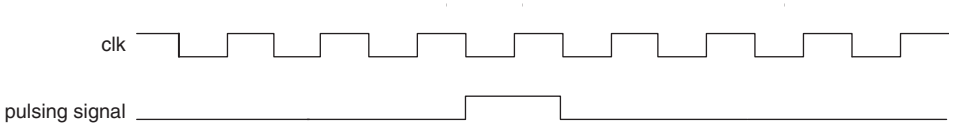
The second way to create a pulse generator is by using a register where the register output drives its own asynchronous reset signal through a delay chain, as shown in Figure 1–27.

Figure 1–27. Pulse Generator Circuit Using a Register Output to Drive a Reset Signal Through a Delay Chain



These pulse generators are asynchronous in nature and are detected by the Design Assistant as unacceptable circuit structures. If you need to generate a pulsed signal, you should do it in a purely synchronous manner. That is, where the duration of the pulse is equal to one or more clock periods, as shown in Figure 1–28.

Figure 1–28. An Example of a Synchronous Pulse Generator



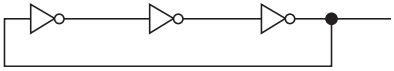
A synchronous pulse generator can be created with a simple section of Verilog HDL or VHDL code. The following is a Verilog HDL code fragment for a synchronous pulse generator circuit.

```
reg [2:0] count;
reg pulse;
always @ (posedge clk or negedge rst)
begin
    if (!rst)
        begin
            count[2:0] <= 3'b000;
            pulse <= 1'b0;
        end
    else
        begin
            count[2:0] <= count[2:0] + 1'b1;
            if (count == 3'b000)
                begin
                    pulse <= 1'b1;
                end
            else
                begin
                    pulse <= 1'b0;
                end
            end
        end
    end
end
```

Combinational Oscillator Circuits

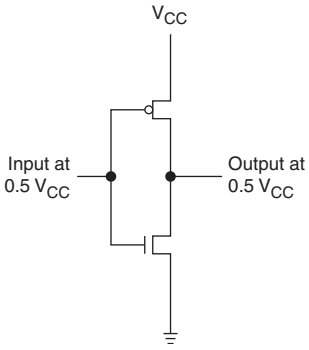
The circuit shown in Figure 1–29 on page 1–24 consists of a combinational logic gate whose inverted output feeds back to one of the inputs of the same gate. This feedback path causes the output to change state and; therefore, oscillate.

Figure 1–29. A Combinational Ring Oscillator Circuit



This circuit is sometimes built out of a series of cascaded inverters in a structure known as a ring oscillator. The frequency at which this circuit oscillates depends on the temperature, voltage, and process operating conditions of the device, and is completely asynchronous to any of the other clock domains in the device. Worse, the circuit may fail to oscillate at all, and the output of the inverter goes to a stable voltage at half of the supply voltage, as shown in Figure 1–30. This causes both the PMOS and NMOS transistors in the inverter chain to be switched on concurrently with a path from V_{CC} to GND, with no inverter function and consuming static current.

Figure 1–30. An Inverter Biased at 0.5 V_{CC}



Avoid implementing any kind of combinational feedback oscillator circuit.

Reset Circuitry

Reset signals are control signals that synchronously or asynchronously affect the state of registers in a design. The special consideration given to clock signals also needs to be given to reset signals. Only the term "reset" is used in this document, but the information described here also applies to "set," "preset," and "clear" signals. Reset signals should only be used to put a circuit into a known initial condition. Also, both the `set` and `reset` pins of the same register should never be used together. If the signals driving them are both activated at the same time, the logic state of the register may be indeterminate.

Gated Reset

A gated reset is generated when combinational logic feeds into the asynchronous reset pin of a register. The gated reset signal may have glitches on it, causing unintentional resetting of the destination register. Figure 1–31 shows a gated reset circuit where the signal driving into the register reset pin has glitches on it causing unintentional resetting.

Figure 1–31. A Gated Reset Circuit and its Associated Timing Diagram

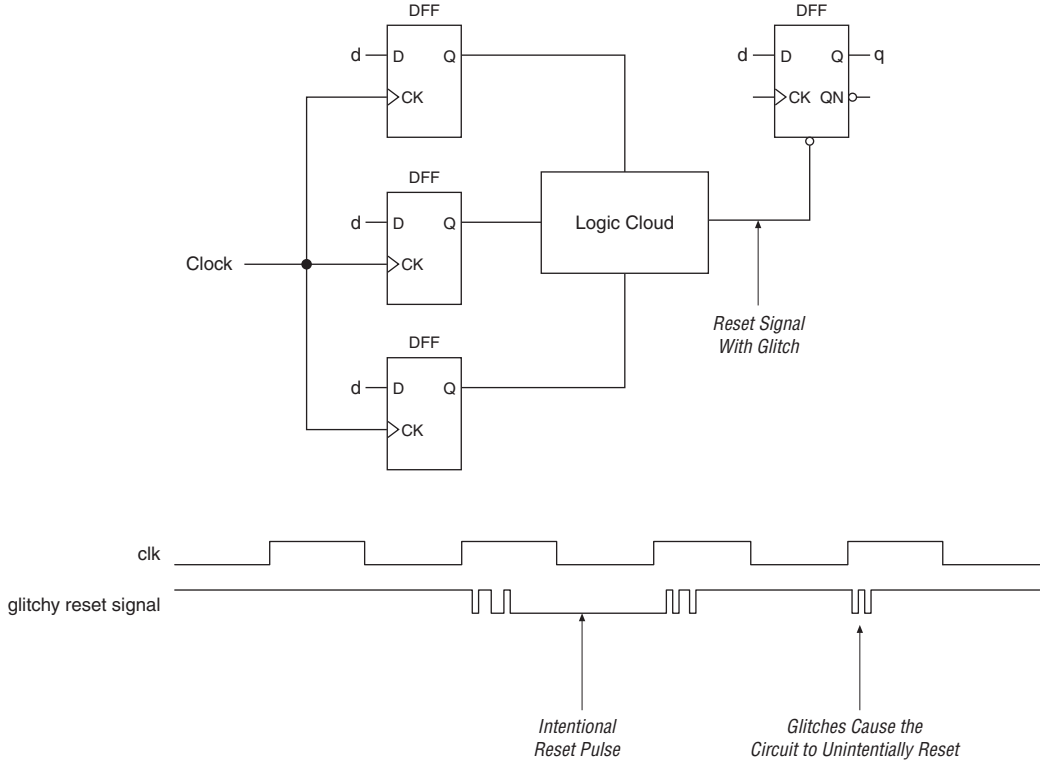
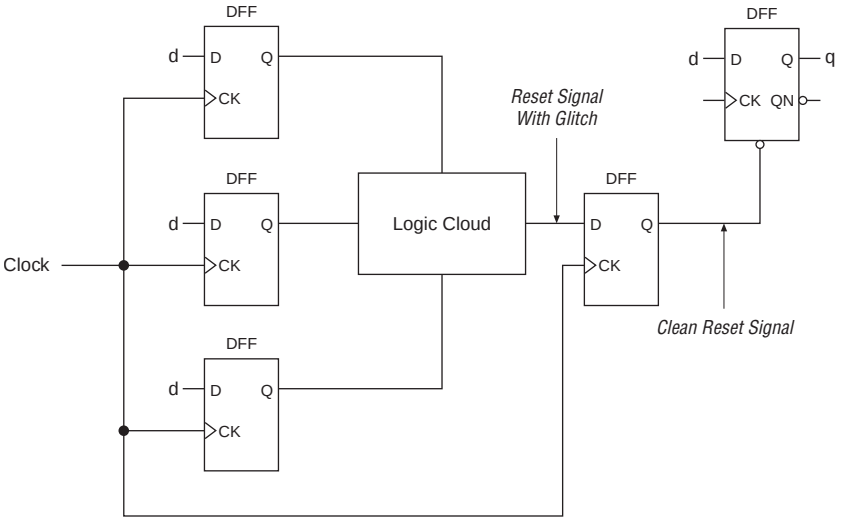


Figure 1–32 shows a better approach to implement a gated reset circuit, by placing a register on the output of the reset-gating logic, thereby synchronizing it to a clock. The register output then becomes a glitch-free reset signal that drives the rest of the design. However, the resulting reset signal is delayed by an extra clock cycle.

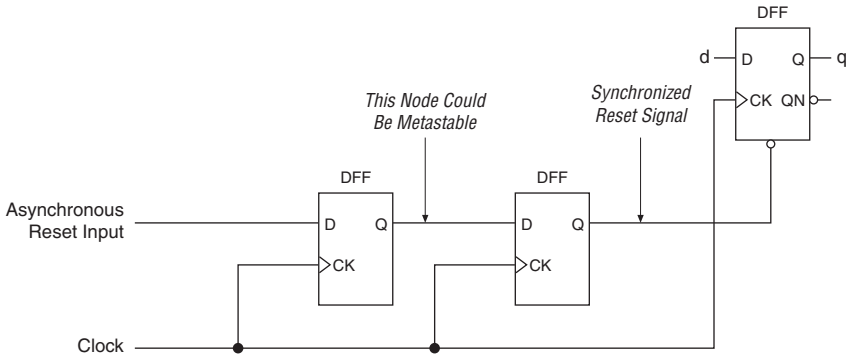
Figure 1–32. A Better Approach to the Gated Reset Circuit in Figure 1–31



Asynchronous Reset Synchronization

If the design needs to be put into a reset state in the absence of a clock signal, the only way to achieve this is through the use of an asynchronous reset. However, it is possible to generate a synchronous reset signal from an asynchronous one by using a double-buffer circuit, as shown in Figure 1–33.

Figure 1–33. A Double-Buffer Circuit



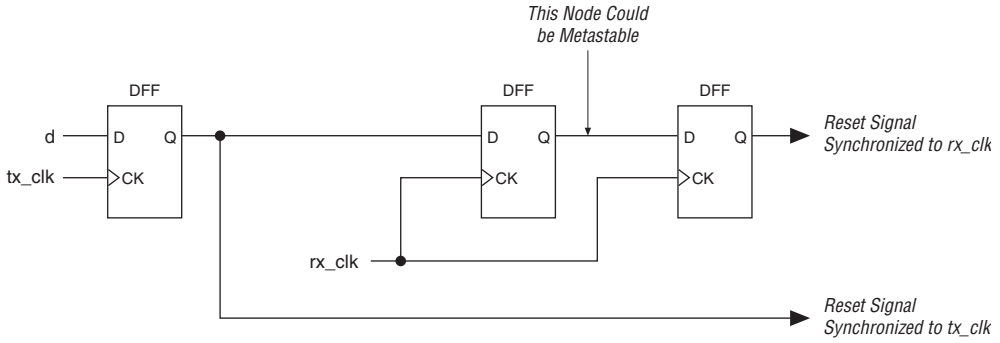
Synchronizing Reset Signals Across Clock Domains

In a design, an internally generated reset signal that is generated in one clock domain, and used in one or more other asynchronous clock domains, should be synchronized. A reset signal that is not synchronized can cause metastability problems.

The synchronization of the gated reset should follow these guidelines, as shown in Figure 1–34.

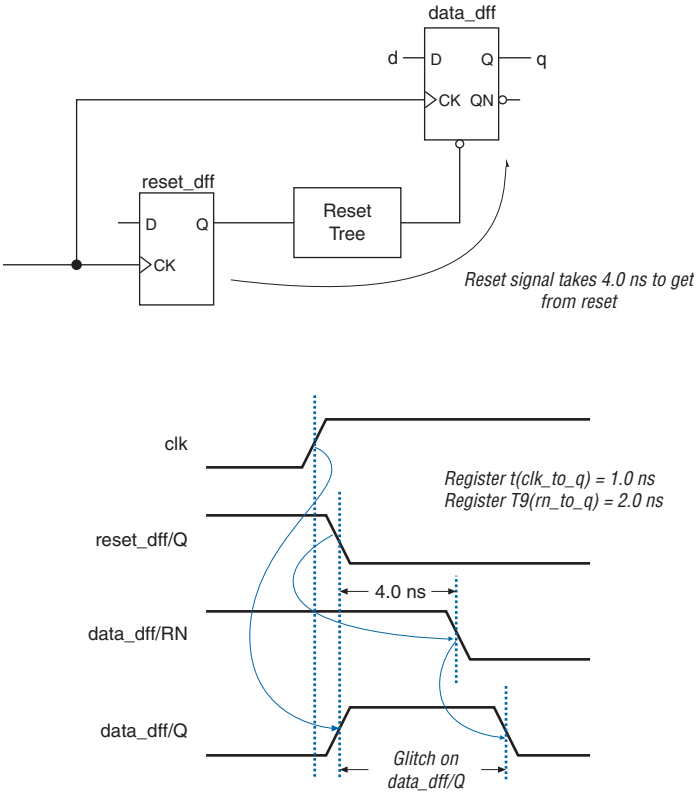
- The reset signal should be synchronized with two or more cascading registers in the receiving asynchronous clock domain.
- The cascading registers should be triggered on the same clock edge.
- There should be no logic between the output of the transmitting clock domain and the cascaded registers in the receiving asynchronous clock domain.

Figure 1–34. Circuit for a Synchronized Reset Signal Across Two Clock Domains



With either of the reset synchronization circuits described in Figures 1–33 and 1–34, when the reset is applied, the Q output of the registers in the design may send a wrong signal, momentarily causing some primary output pins to also send wrong signals. The circuit and its associated timing diagram, shown in Figure 1–35, demonstrate this phenomenon.

Figure 1–35. Common Problem with Reset Synchronization Circuits



A purely synchronous reset circuit does not exhibit this behavior. The following Verilog HDL RTL code shows how to do this.

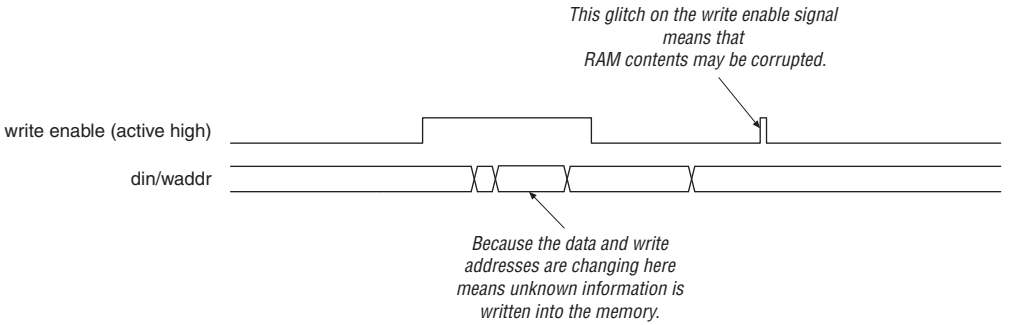
```
always @ (posedge clk)
       begin
       if (!rst)
          q <= 1'b0;
       else
          q <= d; end
```

Avoid using reset signals for anything other than circuit initialization, and be aware of the reset signal timing if reset-synchronizing circuitry is used.

Asynchronous RAM

Altera FPGA devices contain flexible embedded memory structures that can be configured into many different modes. One possible mode is asynchronous RAM. The definition of an asynchronous RAM circuit is one where the write-enable signal driving into the RAM causes data to be written into it, without a clock being required, as shown in Figure 1–36. This means that the RAM is sensitive to corruption if any glitches exist on the write-enable signal. Also, the data and write address ports of the RAM should be stable before the write pulse is asserted, and must remain stable until the write pulse is de-asserted. These limitations in using memory structures in this asynchronous mode imply that synchronous memories are always preferred. Synchronous memories also provide higher design performance.

Figure 1–36. Potential Problems of Using Asynchronous RAM Structures



Stratix, Stratix II, HardCopy Stratix, and HardCopy II device architectures do not support asynchronous RAM behavior. These devices always use synchronous RAM input registers. Altera recommends using RAM output registering; this is optional, however, not using output registering degrades performance.

APEX 20K FPGA and HardCopy APEX support both synchronous and asynchronous RAM using the embedded system block (ESB). Altera recommends using synchronous RAM structures. Immediately registering both input and output RAM interfaces improves performance and timing closure.

Conclusion

Most issues described in this document can be easily avoided while a design is still in its early stages. These issues not only apply to HardCopy devices, but to any digital logic integrated circuit design, whether it is a standard cell ASIC, gate array, or FPGA.

Sometimes, violating one or more of the above guidelines is unavoidable, but understanding the implications of doing so is very important. One must be prepared to justify to Altera the need to break those rules in this case, and to support it with as much documentation as possible.

Following the guidelines outlined in this document can ultimately lead to the design being more robust, quicker to implement, easier to debug, and fitted more easily into the target architecture, increasing the likelihood of success.

Document Revision History

Table 1–2 shows the revision history for this chapter.

Table 1–2. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
September 2008, v3.4	Updated chapter number and metadata.	—
June 2007, v3.3	Minor text edits.	—
December 2006 v3.2	• Added revision history.	Added revision history.
March 2006	Formerly chapter 14; no content change.	—
October 2005, v3.1	• Graphic updates • Minor edits	—
May 2005, v3.0	Updated the Using a FIFO Buffer section.	—
January 2005, v2.0	• Chapter title changed to *Design Guidelines for HardCopy Series Devices*. • Updated *Quartus® II Software Supported Versions* • Updated *HardCopy® Design Center Support* • Updated heading U*sing a Double Synchronizer for Single-Bit Data Transfer* • Added *Stratix® II support for a global or regional clock* • Added *Support for Stratix II and HardCopy II to Mixing Clock Edges*	—

Table 1–2. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
August 2003, v1.1	Edited hierarchy of section headings.	
May 2003, v1.0	Initial release	

2. Power-Up Modes and Configuration Emulation in HardCopy Series Devices

H51012-2.5

Introduction

Configuring an FPGA is the process of loading the design data into the device. Altera's SRAM-based Stratix® II, Stratix, APEX™ 20KC, and APEX 20KE FPGAs require configuration each time the device is powered up. After the device is powered down, the configuration data within the Stratix II, Stratix, or APEX device is lost and must be loaded again on power up.

There are several ways to configure these FPGAs. The details on the various configuration schemes available for these FPGAs are explained in the *Configuration Handbook*.

HardCopy® series devices are mask-programmed and cannot be configured. However, in addition to the capability of being instantly on upon power up (like a traditional ASIC device), these devices can mimic the behavior of the FPGA during the configuration process if necessary.

This chapter addresses various power-up options for HardCopy series devices. This chapter also discusses how configuration is emulated in HardCopy series devices while retaining the benefits of seamless migration and provides examples of how to replace the FPGAs in the system with HardCopy series devices.

HardCopy Power-Up Options

HardCopy series devices feature three variations of instant on power-up modes and a configuration emulation power-up mode. They are as follows:

- Instant on
- Instant on after 50 ms
- Configuration emulation of an FPGA configuration sequence

You must choose the power-up option when submitting the design database to Altera for migrating to a HardCopy series device. Once the HardCopy series devices are manufactured, the power-up option cannot be changed.

HardCopy II and some HardCopy Stratix devices do not support configuration emulation. Refer to "Configuration Emulation of FPGA Configuration Sequence" on page 2–9 for more information.

HardCopy II and HardCopy Stratix devices retain the functionality of VCCSEL and PORSEL pins from the prototyping Stratix and Stratix II FPGAs. The signals can affect the HardCopy series power-up behavior using any power up option. Refer to the *Stratix Device Handbook* or the *Stratix II Device Handbook* for proper use of these additional signals.

Instant On Options

Instant on is the traditional power-up scheme of most ASIC and non-volatile devices. The instant on mode is the fastest power-up option of a HardCopy series device and is used when the HardCopy series device powers up independently while other components on the board still require initialization and configuration. Therefore, you must verify all signals that propagate to and from the HardCopy series device (for example, reference clocks and other input pins) are stable or do not affect the HardCopy series device operation.

There are two variations of instant on power-up modes available on all HardCopy devices.

- Instant on (no added delay)
- Instant on after 50 ms (additional delay)

Instant On (No Added Delay)

In the instant on power-up mode, once the power supplies ramp up above the HardCopy series device's power-on reset (POR) trip point, the device initiates an internal POR sequence. When this sequence is complete, the HardCopy series device transitions to an initialization phase, which releases the CONF_DONE signal to be pulled high. Pulling the CONF_DONE signal high indicates that the HardCopy series device is ready for normal operation. Figures 2–1 to 2–3 show the instant on timing waveform relationships of the configuration signals, $V_{CC,}$ and user I/O pins with respect to the HardCopy series device's normal operation mode.

During the power-up sequence, internal weak pull-up resistors can pull the user I/O pins high. Once POR and the initialization phase is complete, the I/O pins are released. Similar to the FPGA, if the nIO_pullup pin transitions high, the weak pull-up resistors are disabled. Refer to the table that provides recommended operating conditions in the handbook for the specific device.

The value of the internal weak pull-up resistors on the I/O pins is in the Operating Conditions table of the specific FPGA's device handbook.

Instant On After 50-ms Delay

The instant on after 50-ms delay power-up mode is similar to the instant on power-up mode. However, in this case, the device waits an additional 50 ms following the end of the internal POR sequence before releasing the CONF_DONE pin. This option is useful if other devices on the board (such as a microprocessor) must be initialized prior to the normal operation of the HardCopy series device.

An on-chip oscillator generates the 50-ms delay after the power-up sequence. During the POR sequence and delay period, all user I/O pins can be driven high by internal, weak pull-up resistors. Just like the instant on mode, these pull-up resistors are affected by the nIO_pullup pin.

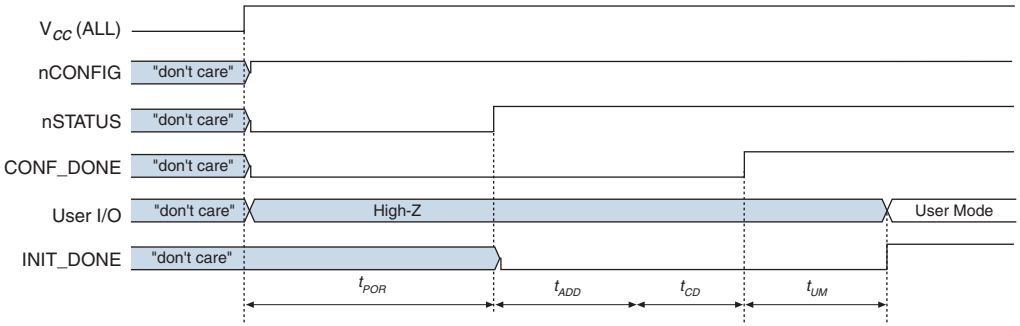
Similar to APEX 20K FPGAs, HardCopy APEX devices do not have an nIO_pullup function. Their internal, weak pull-up resistors are enabled during the power-up and initialization phase.

On the FPGA, an initialization phase occurs immediately after configuration where registers are reset, any PLLs used are initialized, and any I/O pins used are enabled as the device transitions into user mode. When the HardCopy series device uses instant on and instant on after 50-ms modes, a configuration sequence is not necessary, so the HardCopy series device transitions into the initialization phase after a power-up sequence immediately or after a 50-ms delay.

Figures 2–1 to 2–3 show instant on timing waveform relationships of the configuration signals, $V_{CC,}$ and user I/O pins with respect to the HardCopy series device's normal operation mode. Tables 2–1 to 2–3 define the timing parameters for each of the HardCopy series device waveforms, and also show the effect of the PORSEL pin on power up. The nCE pin must be driven low externally for these waveforms to apply.

Figure 2–1 shows an instant on power-up waveform, where the HardCopy device is powered up, and the nCONFIG, nSTATUS, and CONF_DONE are not driven low externally.

Figure 2–1. Timing Waveform for Instant On Option *Notes (1), (2), (3), (4), (5)*



Notes to Figure 2–1:

(1) V_{CC} (ALL) represents either all of the power pins or the last power pin powered up to specified operating conditions. All HardCopy power pins must be powered within specifications as described under *Hot Socketing* sections.

(2) nCONFIG, nSTATUS, and CONF_DONE must not be driven low externally for this waveform to apply.

(3) User I/O pins may be tri-stated or driven before and during power up. See the *Hot Socketing* sections for more details. The nIO_pullup pin can affect the state of the user I/O pins during the initialization phase.

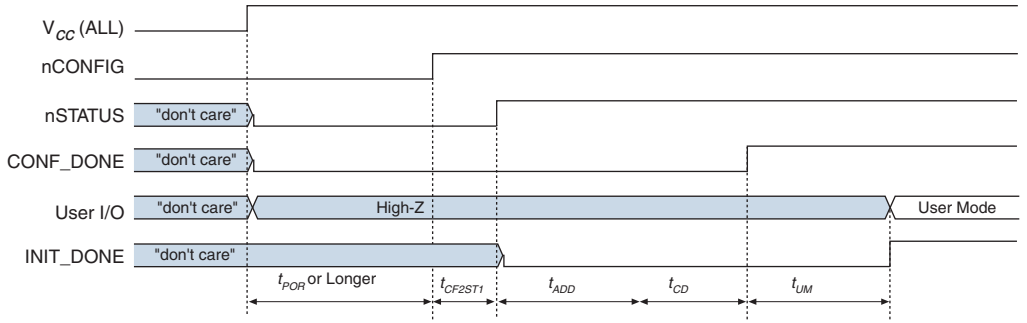
(4) INIT_DONE is an optional pin that can be enabled on the FPGA using the Quartus II software. HardCopy series devices carry over the INIT_DONE functionality from the prototyped FPGA design.

(5) The nCEO pin is asserted about the same time the CONF_DONE pin is released. However, the nCE pin must be driven low externally for this waveform to apply.

An alternative to the power-up waveform in Figure 2–1 is if the nCONFIG pin is externally held low longer than the PORSEL delay. This delays the initialization sequence by a small amount as indicated in Figure 2–2.

In addition, Figure 2–2 is an instant on power-up waveform where nCONFIG is momentarily held low and nSTATUS and CONF_DONE are not driven low externally.

Figure 2–2. Timing Waveform for Instant On Option Where nCONFIG is Held Low After Power Up *Notes (1), (2), (3), (4), (5), (6)*



Notes to Figure 2–2:

(1) This waveform applies if nCONFIG is held low longer than t_{POR} delay.

(2) V_{CC} (ALL) represents either all of the power pins or the last power pin powered up to specified operating conditions. All HardCopy power pins must be powered within specifications as described under *Hot Socketing* sections.

(3) nCONFIG, nSTATUS, and CONF_DONE must not be driven low externally for this waveform to apply.

(4) User I/O pins may be tri-stated or driven before and during power up. See the *Hot Socketing* sections for more details. The nIO_pullup pin can affect the state of the user I/O pins during the initialization phase.

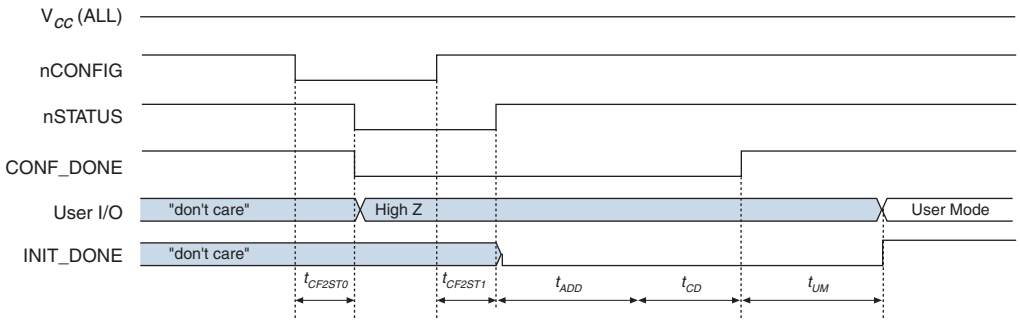
(5) INIT_DONE is an optional pin that can be enabled on the FPGA using the Quartus II software. HardCopy devices carry over the INIT_DONE functionality from the prototyped FPGA design.

(6) The nCEO pin is also asserted about the same time the CONF_DONE pin is released. However, the nCE pin must be driven low externally for this waveform to apply.

Pulsing the nCONFIG signal on an FPGA re-initializes the configuration sequence. The nCONFIG signal on a HardCopy series device also restarts the initialization sequence.

Figure 2–3 shows the instant on behavior of the configuration signals and user I/O pins if the nCONFIG pin is pulsed while the V_{CC} supplies are already powered up and stable.

Figure 2–3. Timing Waveform for Instant On Option When Pulsing NConfig Notes (1), (2), (3), (4), (5)



Notes to Figure 2–3:

(1) V_{CC} (ALL) represents either all of the power pins or the last power pin powered up to specified operating conditions. All HardCopy power pins must be powered within specifications as described under *Hot Socketing* sections.

(2) nSTATUS and CONF_DONE must not be driven low externally for this waveform to apply.

(3) The nIO_pullup pin can affect the state of the user I/O pins during the initialization phase.

(4) INIT_DONE is an optional pin that can be enabled on the FPGA using the Quartus II software. HardCopy devices carry over the INIT_DONE functionality from the prototyped FPGA design.

(5) The nCEO pin is also asserted about the same time the CONF_DONE pin is released. However, the nCE pin must be driven low externally for this waveform to apply.

In the FPGA, the INIT_DONE signal remains high for several clock cycles after the nCONFIG signal is asserted, after which time INIT_DONE goes low. In the HardCopy series device, the INIT_DONE signal starts low, as shown in Figure 2–3, regardless of the logic state of the nCONFIG signal. The INIT_DONE signal transitions high only after the CONF_DONE signal transitions high.

Tables 2–1 through 2–3 show the timing parameters for the instant on mode. These tables also show the time taken for completing the instant on power-up sequence in Figure 2–1 on page 2–4 for HardCopy series devices. This option is typical of an ASIC's functionality.

Table 2–1. Timing Parameters for Instant On Mode in HardCopy II Devices

Parameter	Description	Condition	Min	Typical	Max	Units
t_{POR}	PORSEL delay *(1)*	12		12		ms
		100		100		ms
t_{CF2ST0}	nCONFIG low to nSTATUS low *(1)*				800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high *(1)*				100	µs
t_{ADD}	Additional delay	Instant on	33		60	µs
		After 50 ms added delay	50		90	ms
t_{CD}	CONF_DONE delay		600		1100	ns
t_{UM}	User mode delay		25		55	µs

Note to Table 2–1:
(1) This parameter is similar to the Stratix II FPGA specifications. Refer to the *Configuration Handbook* for more information.

Table 2–2. Timing Parameters for Instant On Mode in HardCopy Stratix Devices

Parameter	Description	Condition	Min	Typical	Max	Units
t_{POR}	PORSEL delay	2	1	2		ms
		100	70	100		ms
t_{CF2ST0}	nCONFIG low to nSTATUS low *(1)*				800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high *(1)*				40	µs
t_{ADD}	Additional delay	Instant on	4		8	ms
		After 50 ms added delay	25	50	75	ms
t_{CD}	CONF_DONE delay		0.5		3	µs
t_{UM}	User mode delay		6.0		28	µs

Note to Table 2–2
(1) This parameter is similar to the Stratix FPGA specifications. Refer to the *Configuration Handbook* for more information.

Table 2–3. Timing Parameters for Instant On Mode in HardCopy APEX Devices

Parameter	Description	Condition	Min	Typical	Max	Units
t_{POR}	POR delay			5		µs
t_{CF2ST0}	nCONFIG low to nSTATUS low *(1)*				200	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high *(1)*				1	µs
t_{ADD}	Additional delay	Instant on		0		µs
		After 50 ms added delay		50		ms
t_{CD}	CONF_DONE delay		0.5		3	µs
t_{UM}	User mode delay		2.5		8	µs

Note to Table 2–3:

(1) This parameter is similar to the APEX FPGA specifications. Refer to the *Configuration Handbook* for more information.

For correct operation of a HardCopy series device using the instant on option, pull the nSTATUS, nCONFIG, and CONF_DONE pins to V_{CC}. In the HardCopy series devices, these pins are designed with weak internal resistors pulled up to V_{CC}. Many FPGA configuration schemes require pull-up resistors on these I/O pins, so they may already be present on the board. In some HardCopy series device applications, you can remove these external pull-up resistors.

Altera recommends leaving external pull-up resistors on the board if one of the following conditions exists.

For more information, refer to the *Designing with 1.5-V Devices* chapter in the *Stratix Device Handbook*.

- There is more than one HardCopy series and/or FPGA on the board
- The HardCopy design uses configuration emulation
- The design uses MultiVolt I/O configurations

In the FPGA, you can enable the INIT_DONE pin in the Quartus II software. If you used the INIT_DONE pin on the FPGA prototype, the HardCopy series device retains its function.

- In HardCopy series devices, the INIT_DONE settings option is masked-programmed into the device. You must submit these settings to Altera with the final design prior to migrating to a HardCopy series device. The use of the INIT_DONE option and other option pins (for example, DEV_CLRn and DEV_OE) are available in the Fitter Device Options sections of the Quartus II report file.
- For HardCopy II and HardCopy Stratix devices, the PORSEL pin setting delays the POR sequence similar to the prototyping FPGA. For more information on PORSEL settings for the FPGA, refer to the *Configuration Handbook*.

In some FPGA configuration schemes, inputs DCLK and DATA[7..0] float if the configuration device is removed from the board. In the HardCopy series devices, these I/O pins are designed with weak, internal pull-up resistors, so the pins can be left unconnected on the board.

Configuration Emulation of FPGA Configuration Sequence

In configuration emulation mode, the HardCopy series device emulates the behavior of an APEX or Stratix FPGA during its configuration phase. When this mode is used, the HardCopy device uses a configuration emulation circuit to receive configuration bit streams. When all the configuration data is received, the HardCopy series device transitions into an initialization phase and releases the CONF_DONE pin to be pulled high. Pulling the CONF_DONE pin high signals that the HardCopy series device is ready for normal operation. If the optional open-drain INIT_DONE output is used, the normal operation is delayed until this signal is released by the HardCopy series device.

HardCopy II and some HardCopy Stratix devices do not support configuration emulation mode.

During the emulation sequence, the user I/O pins can be pulled high by internal, weak pull-up resistors. Once the configuration emulation and initialization phase is completed, the I/O pins are released. Similar to the FPGA, if the nIO_pullup pin is driven high, the weak pull-up resistors are disabled. The value of the internal weak pull-up resistors on the I/O pins can be found in the Operating Conditions table of the specific FPGA's device handbook.

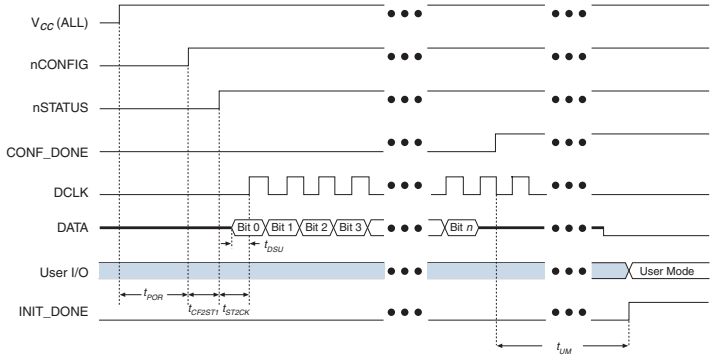
Similar to APEX 20K FPGAs, HardCopy APEX devices do not have an nIO_pullup function. Their internal weak pull-up resistors are enabled during the power up and initialization phase.

Similar to Stratix or APEX FPGAs, HardCopy Stratix or HardCopy APEX devices enter initialization phase immediately after a successful configuration sequence. At this time, registers are reset, any PLLs used are initialized, and any I/O pins used are enabled as the device transitions into user mode.

One application of the configuration emulation mode occurs when multiple programmable devices are cascaded in a configuration chain and only one device is replaced with a HardCopy series device. In this case, programming control signals and clock signals used to program the FPGA must also be used for the HardCopy series device. If this is not done, the HardCopy series device remains in the configuration emulation phase, the emulation sequence never ends, and the HardCopy CONF_DONE pin remains de-asserted. The proper configuration data stream and data clock is necessary so the HardCopy series device has the accurate emulation behavior.

Figure 2–4 shows a waveform of the configuration signals and the user I/O signals using configuration emulation mode.

Figure 2–4. Timing Waveform for Configuration Emulation Mode *Notes (1), (2), (3), (4), (5)*



Notes to Figures 2–4:

(1) V_{CC} (ALL) represents either all of the power pins or the last power pin powered up to specified operating conditions. All HardCopy power pins must be powered within specifications as described under *Hot Socketing* sections.

(2) `nCONFIG`, `nSTATUS`, and `CONF_DONE` must not be driven low externally for this waveform to apply.

(3) User I/O pins may be tri-stated or driven before and during power up. See the *Hot Socketing* sections for more details. The `nIO_pullup` pin can affect the state of the user I/O pins during the initialization phase.

(4) `INIT_DONE` is an optional pin that can be enabled on the FPGA using the Quartus II software. HardCopy devices will carry over the `INIT_DONE` functionality from the prototyped FPGA design.

(5) The `nCEO` pin is also asserted about the same time the `CONF_DONE` pin is released. However, the `nCE` pin must be driven low externally for this waveform to apply.

Configuration Emulation Timing Parameters

Tables 2–4 and 2–5 provide the timing parameters for the configuration emulation mode.

Table 2–4. Timing Parameters for Configuration Emulation Mode in HardCopy Stratix Devices *Note (1)*

Parameter	Description *(2)*	Condition	Min	Typ	Max	Units
t_{POR}	PORSEL delay	2	1	2		ms
		100	70	100		ms
t_{DSU}	Data setup time		7			ns
t_{CF2ST1}	nCONFIG high to nSTATUS				40	µs
t_{ST2CK}	nSTATUS to DCLK		1			µs
t_{UM}	User mode delay		6.0		28	µs

Notes to Table 2–4:
(1) HC1S80, HC1S60, and HC1S25 devices do not support emulation mode.
(2) These parameters are similar to the Stratix FPGA specifications. Refer to the *Configuration Handbook* for more information.

Table 2–5. Timing Parameters for Configuration Emulation Mode in HardCopy APEX Devices

Parameter	Description *(1)*	Min	Typical	Max	Units
t_{POR}	POR delay		5		µs
t_{DSU}	Data setup time	10			ns
t_{CF2ST1}	nCONFIG high to nSTATUS			1	µs
t_{ST2CK}	nSTATUS to DCLK	1		3	µs
t_{UM}	User mode delay	2		8	µs

Notes to Table 2–5:
(1) These parameters are similar to the APEX FPGA specifications. Refer to the *Configuration Handbook* for more information.

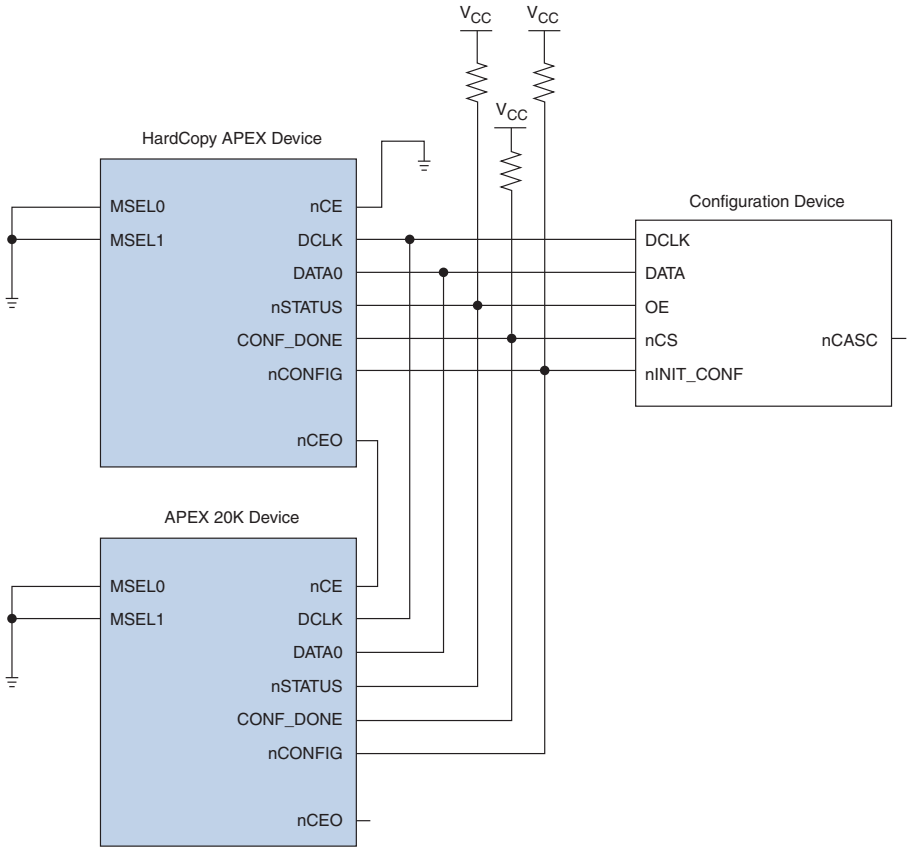
Benefits of Configuration Emulation

Configuration emulation in HardCopy series devices provides several advantages, including the following:

- Removes any necessity for changes to software, especially if the FPGA is configured using a microprocessor. Not having to change the software benefits the designer because microprocessor software changes demand significant system verification and qualification efforts, which also impact development time.
- Allows HardCopy series devices to co-exist with other FPGAs in a cascaded chain. None of the components need to be modified or added, and no design changes to the board are required. Additionally, no configuration software changes need to be made.
- Supports all configuration options available for the FPGA.

In this example, a single configuration device originally configured two APEX FPGAs. In Figure 2–5, a HardCopy APEX device replaces an APEX FPGA.

Figure 2–5. Emulation of Configuration Sequence



A HardCopy series device in configuration emulation mode requires the same configuration control signals as the FPGA that was replaced. In configuration emulation mode, the HardCopy series device responds in exactly the same way as the FPGA. The CONF_DONE signal of the HardCopy series device is asserted at exactly the same time as the FPGA.

Power-Up Options Summary When Designing With HardCopy Series Devices

When designing a board for the prototyping FPGA with the intent of eventually replacing it with a HardCopy device, there are three power-up options that you should consider.

- Instant on
- Instant on after 50 ms
- Configuration emulation of an FPGA configuration sequence

You must choose the power-up option when submitting the design database to Altera for migrating to a HardCopy series device. Once the HardCopy series devices are manufactured, the power-up option cannot be changed.

HardCopy II and some HardCopy Stratix devices do not support configuration emulation mode.

HardCopy II and HardCopy Stratix devices retain the functionality of the VCCSEL and PORSEL pins from the prototyping Stratix II or Stratix FPGAs. For HardCopy II and HardCopy Stratix devices, the PORSEL pin setting delays the POR sequence similar to the prototyping FPGA.

For more information on PORSEL settings for the FPGA, refer to the *Configuration Handbook*.

The nCE and nCEO pins are functional in HardCopy series devices. The nCE pin must be held low for proper operation of the nCEO pin. If the nCE pin is driven low, the nCEO pin will be asserted after the initialization is completed and the CONF_DONE pin is released.

On the HardCopy II device, the nCE pin delays the initialization if it is not driven low. Like in the Stratix II device, nCEO and TDO of the HardCopy II device are powered by V_{CCIO}.

If you used the INIT_DONE pin on the FPGA prototype, the HardCopy series device retains its function. In HardCopy series devices, the INIT_DONE settings option is masked-programmed into the device. These settings must be submitted to Altera with the final design prior to migrating to a HardCopy series device. The use of the INIT_DONE option and other option pins (for example, DEV_CLRn and DEV_OE) are available in the Fitter Device Options sections of the Quartus II report file.

HardCopy II devices do not support the user-supplied start-up clock option available for Stratix II devices. The HardCopy II device uses its own internal clock for power-up circuitry. The startup clock selection is an option for configuring the FPGA, which you can set in the Quartus II software under Device and Pin Options.

HardCopy devices support device-wide reset (DEV_CLRn) and device-wide output enable (DEV_OE). The HardCopy settings follow the prototyping FPGA setting, which you set in the Quartus II software under Device and Pin Options.

For correct operation of a HardCopy series device using the instant on option, pull the nSTATUS, nCONFIG, and CONF_DONE pins to V_{CC}. In the HardCopy series devices, these pins are designed with weak, internal resistors pulled up to V_{CC}. Many FPGA configuration schemes require pull-up resistors on these I/O pins, so they may already be present on the board. In some HardCopy series device applications, you can remove these external pullup resistors.

Altera recommends leaving external pull-up resistors on the board if one of the following conditions exists:

- There is more than one HardCopy series and/or FPGA on the board
- The HardCopy design uses configuration emulation
- The design uses MultiVolt™ I/O configurations

For more information, refer to the *Designing with 1.5-V Devices* chapter in the *Stratix Device Handbook*.

In some FPGA configuration schemes, inputs DCLK and DATA[7..0] float if the configuration device is removed from the board. In the HardCopy series devices, these I/O pins are designed with weak internal pull-up resistors, so the pins can be left unconnected on the board.

When designing a board with a Stratix II prototype device and its companion HardCopy II device, most configuration pins required by the Stratix II device are not required by the HardCopy II device. To maximize I/O pin counts with HardCopy II device utilization, Altera recommends minimizing power-up and configuration pins that do not carry over from a Stratix II device into a HardCopy II device. More information can be found on the *Migrating Stratix II Device Resources to HardCopy II Devices* chapter.

HardCopy devices support the MSEL settings used on the FPGA. You are not required to change these settings on the board when replacing the prototyping FPGA with the HardCopy series device.

HardCopy II devices do not use MSEL pins and these pin locations are not connected in the package. It is acceptable to drive these pins to V_{CC} or GND as required by the prototyping Stratix II device.

Pulsing the nCONFIG signal on an FPGA re-initializes the configuration sequence. The nCONFIG signal on a HardCopy series device also restarts the initialization sequence.

The HardCopy device JTAG pin locations match their corresponding FPGA prototypes. Like the FPGAs, the JTAG pins have internal weak pull ups or pull downs on the four input pins TMS, TCK, TDI, and TRST. There is no requirement to change the JTAG connections on the board when replacing the prototyping FPGA with the HardCopy series device. More information on JTAG pins is the corresponding *Boundary-Scan Support* chapter for each device.

Power-Up Option Selection and Examples

The HardCopy series device power-up option is mask-programmed. Therefore, it is important that the board design is verified to ensure that the HardCopy series device power-up option chosen will work properly. This section provides recommendations on selecting a power-up option and provides some examples.

Table 2–6 shows a comparison of applicable FPGA and HardCopy power up options.

Table 2–6. FPGA Configuration Modes and HardCopy Series Power-Up Schemes (Part 1 of 2)

Power Up Scheme	Device Family					
	Stratix II	Stratix	APEX 20K APEX 20KE APEX 20KC	HardCopy II *(1)*	HardCopy Stratix *(2)*	HardCopy APEX
Instant on				✓	✓	✓
Instant on after 50 ms				✓	✓	✓
Passive serial (PS)	✓	✓	✓		✓	✓
Active serial (AS)	✓					
Fast passive parallel (FPP)	✓	✓			✓	
Passive parallel synchronous (PPS)			✓			✓
Passive parallel asynchronous (PPA)	✓	✓	✓		✓	✓
Joint Test Action Group (JTAG)	✓	✓	✓		✓	✓
Remote local update FPP *(3)*	✓	✓				

Table 2–6. FPGA Configuration Modes and HardCopy Series Power-Up Schemes (Part 2 of 2)

Power Up Scheme	Device Family					
	Stratix II	Stratix	APEX 20K APEX 20KE APEX 20KC	HardCopy II *(1)*	HardCopy Stratix *(2)*	HardCopy APEX
Remote local update PPA *(3)*	✓	✓				
Remote local update PS *(3)*		✓				

Notes to Table 2–6:
(1) HardCopy II devices do not support emulation mode.
(2) HC1S80, HC1S60, and HC1S25 devices do not support emulation mode.
(3) The remote/local update feature of Stratix devices is not supported in HardCopy Stratix devices.

Power-up option recommendations depend on the following board configurations:

- Single HardCopy series device replacing a single FPGA on the board
- One or more HardCopy series devices replacing one or more FPGA of a multiple-device configuration chain
- All HardCopy series devices replacing all FPGAs of a multiple-device configuration chain

In a multiple-device configuration chain, more than one FPGA on a board obtains configuration data from the same source.

Replacing One FPGA With One HardCopy Series Device

Altera recommends using the instant on or instant on after 50 ms mode when replacing an FPGA with a HardCopy series device regardless of the board configuration scheme. Table 2–7 gives a summary of HardCopy series device power-up options when a single HardCopy series device replaces a single FPGA on the board.

Table 2–7 does not include HardCopy II options because HardCopy II devices only support instant on and instant on after 50 ms modes.

Table 2–7. Summary of Power-Up Options for One HardCopy Series Device Replacing One FPGA

Configuration Scheme	HardCopy APEX Options	HardCopy Stratix Options	Comments
PS with configuration device(s) or download cable *(1)*	• Instant on • Instant on after 50 ms	• Instant on • Instant on after 50 ms	The configuration device(s) must be removed from the board.
FPP with enhanced configuration devices	• Not available	• Instant on • Instant on after 50 ms	The configuration device(s) must be removed from the board.
PS, PPA, PPS, FPP, with a microprocessor *(2)*	• Emulation	• Emulation *(3)*	If the microprocessor code can be changed, the design should use the instant on or instant on after 50 ms mode. However, the microprocessor still needs to drive a logic '1' value on the HardCopy `nCONFIG` pin
JTAG configuration	• Instant on after 50 ms • Emulation	• Instant on after 50 ms • Emulation *(3)*	Configuration emulation mode can be used but delays the initialization of the board or device.

Notes to Table 2–7:

(1) Download cable used may be either MasterBlaster™, USB Blaster, ByteBlaster™ II, or ByteBlasterMV™ hardware.

(2) For parallel programming modes, `DATA[7..1]` pins have weak pull up resistors on the HardCopy series device, which can be optionally enabled or disabled through metallization. `DCLK` and `DATA[0]` pins have internal weak pull-up resistors.

(3) HC1S80, HC1S60, and HC1S25 devices do not support emulation mode.

Replacing One or More FPGAs With One or More HardCopy Series Devices in a Multiple-Device Configuration Chain

Altera recommends using the instant on or instant on after 50 ms mode when replacing an FPGA with a HardCopy series device, regardless of configuration scheme. Table 2–8 gives a summary of HardCopy series device power-up options when a single HardCopy series device replaces a single FPGA of a multiple-device configuration chain.

When using the instant on or instant on after 50 ms mode, the HardCopy series device could be in user-mode and ready before other configured devices on the board. It is important to verify that any signals that communicate to and from the HardCopy series device are stable or will not affect the HardCopy series device or other device operation while the devices are still in the power up or configuration stage. For example, if the HardCopy series design used a PLL reference clock that is not available until after other devices are fully powered up, the HardCopy series device PLL will not operate properly unless the PLLs are reset.

Table 2–8 does not include HardCopy II options because HardCopy II devices only support instant on and instant on after 50 ms modes.

Table 2–8. Power-Up Options for One or More HardCopy Series Devices Replacing FPGAs in a Multiple-Device Configuration Chain (Part 1 of 2)

Configuration Scheme	HardCopy APEX Options	HardCopy Stratix Options	Comments
PS with configuration device(s) or download cable *(1)* FPP with enhanced configuration device *(4)*	• Emulation • Instant on *(3)* • Instant on after 50 ms *(3)*	• Emulation *(2)* • Instant on *(3)* • Instant on after 50 ms *(3)*	Instant on or instant on after 50 ms modes can be used if the nCE pin of the following APEX or Stratix device can be tied to logic 0 on the board and the configuration data is modified to remove the HardCopy series device configuration data. The configuration sequence then skips the HardCopy series device.
PS, PPA, PPS, FPP, with a microprocessor *(4)*	• Emulation	• Emulation *(2)*	If the microprocessor code can be changed, the design should use the instant on or instant on after 50 ms mode. However, the microprocessor still needs to drive a logic '1' value on the HardCopy series device nCONFIG pin.

Table 2–8. Power-Up Options for One or More HardCopy Series Devices Replacing FPGAs in a Multiple-Device Configuration Chain (Part 2 of 2)

Configuration Scheme	HardCopy APEX Options	HardCopy Stratix Options	Comments
JTAG configuration	• Emulation	• Emulation *(2)*	If the HardCopy series device is put in BYPASS mode and the JTAG programming data is modified to remove the HardCopy configuration information, instant on or instant on after 50 ms modes can be used.

Notes to Table 2–8:
(1) Download cable used may be either MasterBlaster, USB Blaster, ByteBlaster II, or ByteBlasterMV hardware.
(2) HC1S80, HC1S60, and HC1S25 devices do not support emulation mode.
(3) If the HardCopy series device is the last device in the configuration chain, Altera recommends using instant on modes.
(4) For parallel programming modes, DATA[7..1] pins have weak pull up resistors on the HardCopy series device, which can be optionally enabled or disabled through metallization. DCLK and DATA[0] pins also have weak pull-up resistors.

Replacing all FPGAs with HardCopy Series Devices in a Multiple-Device Configuration Chain

When all Stratix II, Stratix, and APEX FPGAs are replaced by HardCopy II, HardCopy Stratix, and HardCopy APEX devices, respectively, Altera recommends using the instant on or instant on after 50 ms mode, regardless of configuration scheme.

Once the HardCopy series devices replace the FPGAs, any configuration devices used to configure the FPGAs should be removed from the board. Microprocessor code, if applicable, should be changed to account for the HardCopy series device power-up scheme. You can use the JTAG chain to perform other JTAG operations except configuration.

FPGA to HardCopy Configuration Migration Examples

The following are examples of how HardCopy series devices replace FPGAs that use different FPGA configuration schemes.

HardCopy Series Device Replacing a Stand-Alone FPGA

In this example, the single HardCopy series device uses the instant on power-up option, as shown in Figure 2–7. The configuration device, now redundant, is removed, and no further board changes are necessary. The pull-up resistors on the nCONFIG, nSTATUS, and CONF_DONE pins can be removed, but should be left on the board if configuration emulation or multiple-voltage I/O standards are used. You could also use the instant on after 50 ms power-up mode in this example.

Figures 2–6 and 2–7 show how a HardCopy series device replaces an FPGA previously configured with an Altera configuration device.

Figure 2–6. Configuration of a Stand-Alone FPGA *Note (1)*

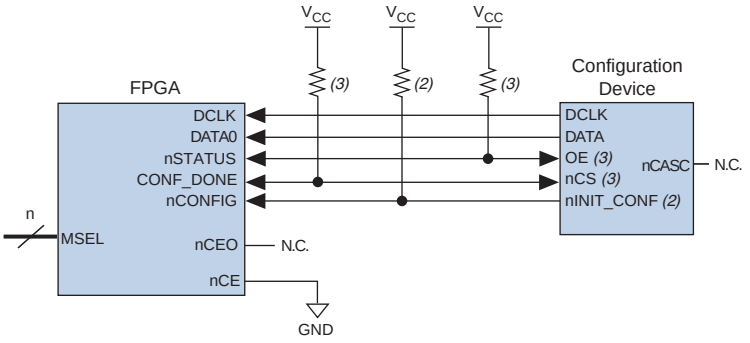
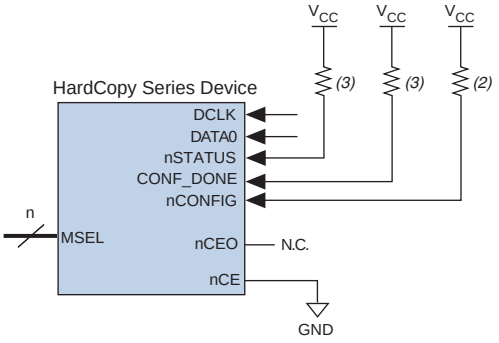


Figure 2–7. HardCopy Series Device Replacing Stand-Alone FPGA *Note (1)*



Notes to Figures 2–6 and 2–7:

(1) For details on configuration interface connections, refer to the *Configuration Handbook*. The handbook includes information on `MSEL` pins set to PS mode.

(2) The `nINIT_CONF` pin (available on enhanced configuration and EPC2 devices) has an internal pull-up resistor that is always active. Therefore, the `nINIT_CONF/nCONFIG` line does not require an external pull-up resistor. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used or not available, use a resistor to pull the `nCONFIG` pin to V_{CC}.

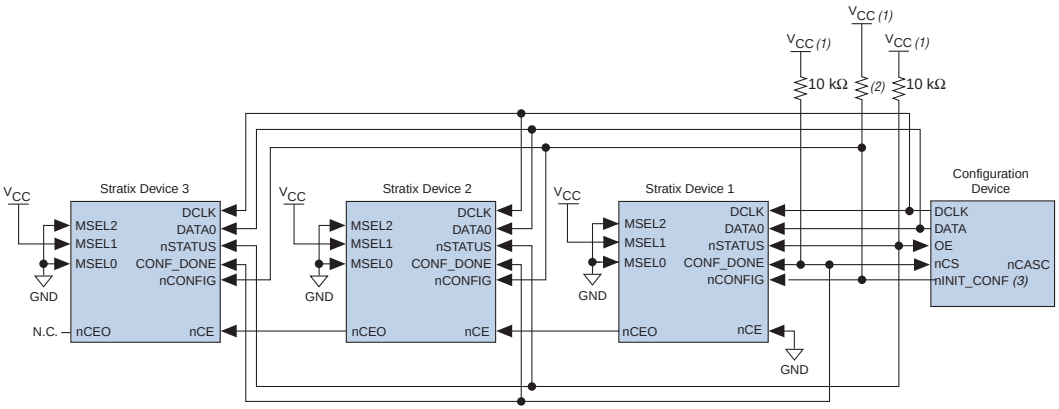
(3) Enhanced configuration and EPC2 devices have internal programmable pull-up resistors on `OE` and `nCS` pins. Refer to the *Configuration Handbook* for more details of this application in FPGAs. HardCopy series devices have internal weak pull-up resistors on `nSTATUS`, `nCONFIG`, and `CONF_DONE` pins.

HardCopy Series Device Replacing an FPGA in a Cascaded Configuration Chain

Figure 2–8 shows a design where the configuration data for the Stratix devices is stored in a single configuration device, and the FPGAs are connected in a multiple-device configuration chain. The second device in the chain is replaced with a HardCopy Stratix device, as shown in Figure 2–9.

For more information on Stratix FPGA configuration schemes, refer to the *Configuration Handbook*.

Figure 2–8. Configuration of Multiple FPGAs in a Cascade Chain



Notes to Figure 2–8:

(1) The pull-up resistors are connected to the same supply voltage as the configuration device.

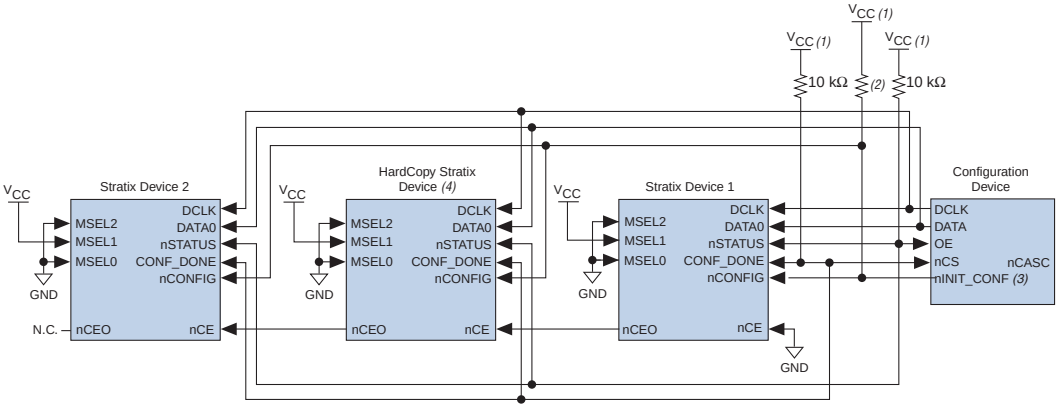
(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-up resistors on the `OE` and `nCS` pins. Refer to the *Configuration Handbook* for more details.

(3) The `nINIT_CONF` pin is available on EPC16, EPC8, EPC4, and EPC2 devices. Refer to the *Configuration Handbook* for more details.

Configuration with the HardCopy Series Device in the Cascade Chain

Figure 2–9 shows the same cascade chain as Figure 2–8, but the second FPGA in the chain has been replaced with a HardCopy Stratix device.

Figure 2–9. Replacing an FPGA with a HardCopy Equivalent in the Cascade Chain



Notes to Figure 2–9:

(1) The pull-up resistors are connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-up resistors on the OE and nCS pins. Refer to the *Configuration Handbook* for more details.

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. Refer to the *Configuration Handbook* for more information.

(4) HC1S80, HC1S60, and HC1S25 devices do not support emulation mode and cannot be used in this method.

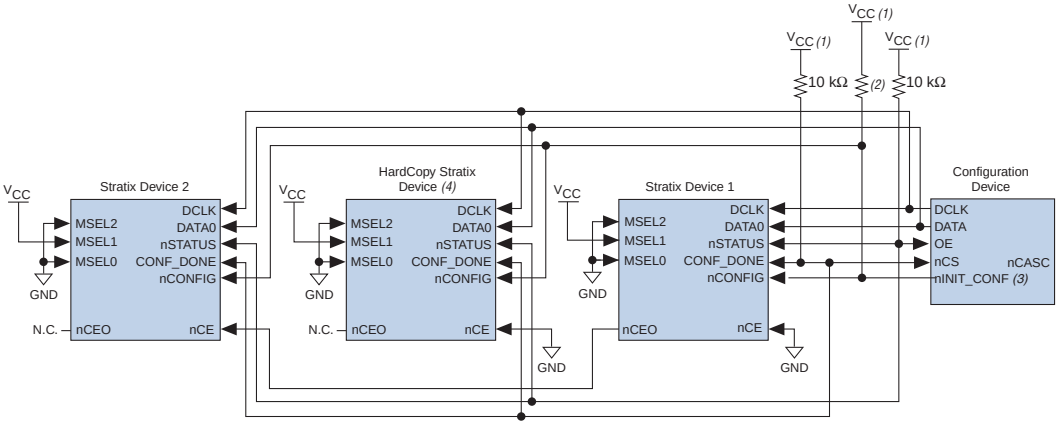
In this example, the HardCopy Stratix device can only be configured using the configuration emulation mode. The configuration device cannot be removed, as it is still required by other Stratix devices in the chain. While the HardCopy Stratix device does not need the data stored in the configuration device, the data in the configuration device is not modified to reflect this. The emulation mode ensures that the HardCopy series device nCEO pin is asserted correctly after the emulation of the configuration sequence. The nCEO pin enables the next device in the chain to receive the correct configuration data from the configuration device. Additionally, with the configuration emulation mode, you do not need to make any changes to the board.

Configuration With the HardCopy Series Device Removed From the Cascade Chain

An alternative method to configure FPGAs on a board with both HardCopy series devices and FPGAs is to remove the HardCopy series device from the cascade chain. Figure 2–10 shows how the devices are connected with the HardCopy series device removed from the chain.

The data in the configuration device should be modified to exclude the HardCopy series device configuration data. The HardCopy series device can use any of the three power-up options.

Figure 2–10. Configuration With the HardCopy Series Device Removed From the Cascade Chain



Notes to Figure 2–10:

(1) The pull-up resistors are connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-up resistors on the `OE` and `nCS` pins. Refer to the *Configuration Handbook* for more details.

(3) The `nINIT_CONF` pin is available on EPC16, EPC8, EPC4, and EPC2 devices. Refer to the *Configuration Handbook* for more information.

(4) HC1S80, HC1S60, and HC1S25 devices do not support emulation mode and cannot be used in this method.

Eliminating the HardCopy series device from the configuration chain requires the following changes on the board:

- The `nCE` pin of the HardCopy series device must be tied to `GND`.
- The `nCE` pin of the FPGA that was driven by the HardCopy series `nCEO` pin must now be driven by the `nCEO` pin of the FPGA that precedes the HardCopy series device in the chain.

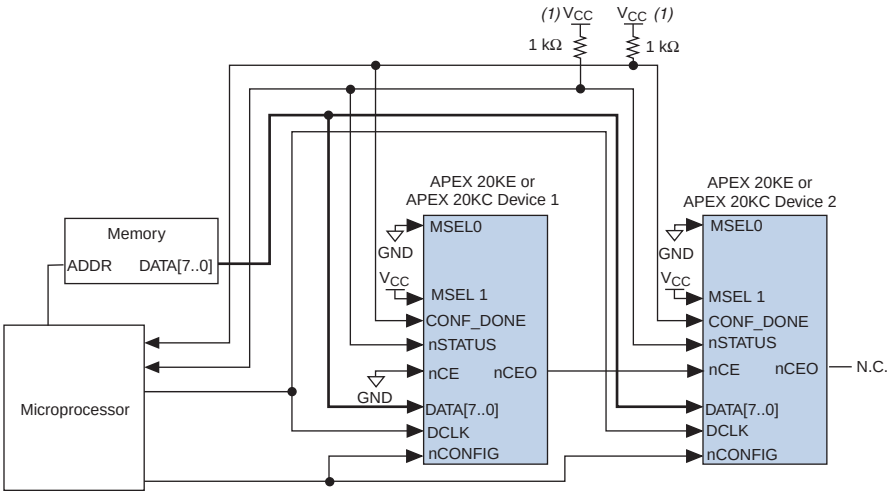
HardCopy Series Device Replacing an FPGA Configured Using a Microprocessor

The HardCopy series device can replace FPGAs that are configured using a microprocessor, as shown in Figures 2–12 and 2–13. While the instant on mode is the most efficient, designers can also use the instant on after 50 ms and configuration emulation mode.

Figure 2–11 shows an application where APEX FPGAs are configured using a microprocessor in the PPS configuration scheme.

For more information on the PPS configuration scheme, refer to the *Configuration Handbook*.

Figure 2–11. Configuring FPGAs Using a Microprocessor



Note to Figure 2–11:
(1) Connect the pull-up resistors to a supply that provides an acceptable input signal for all devices in the chain.

When the HardCopy series device replaces the last FPGA of the configuration sequence (as shown in Figure 2–12), use the instant on or instant on after 50 ms mode. However, you must modify the microprocessor code to eliminate the configuration data for the last FPGA of the configuration chain.

Figures 2–12 and 2–13 show the HardCopy APEX device replacing APEX FPGAs either first or last in the configuration chain.

Figure 2–12. Replacement of Last FPGA in the Chain With a HardCopy Series Device

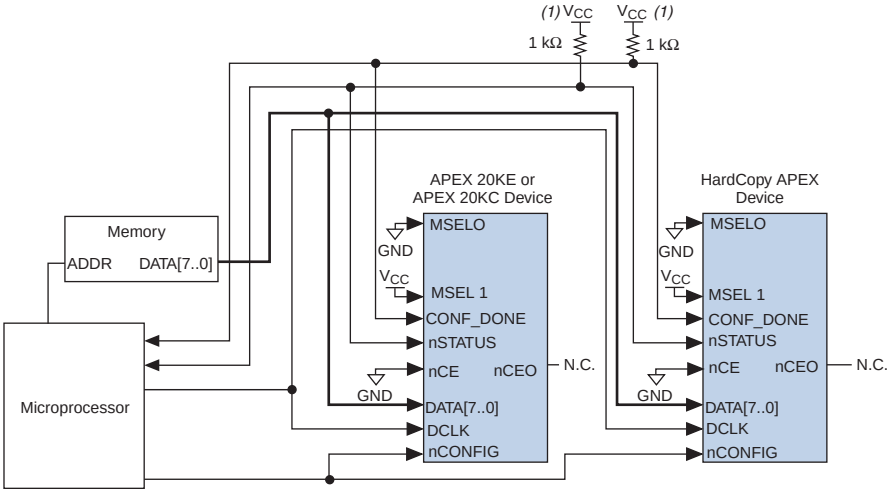
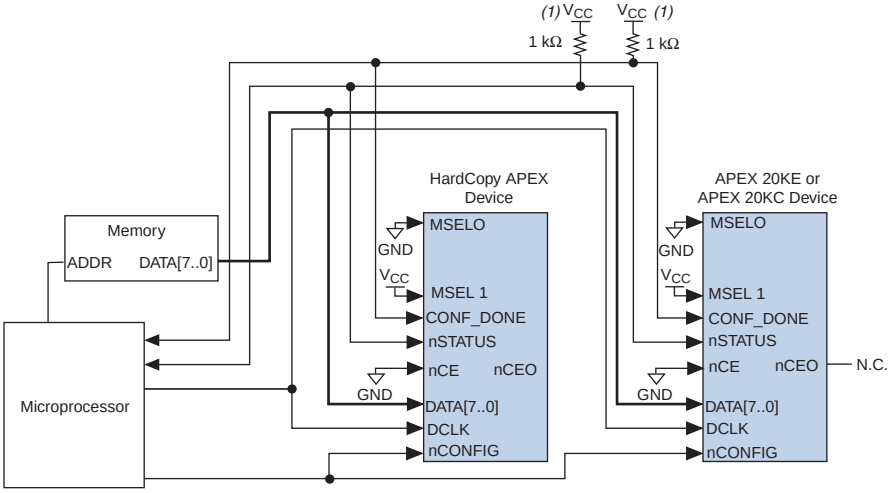


Figure 2–13. Replacement of First FPGA in the Chain With a HardCopy Series Device



Note to Figures 2–12 and 2–13:

(1) Connect the pull-up resistors to a supply that provides an acceptable input signal for all devices in the chain.

If the HardCopy series device is the first device in the chain as opposed to the second (as shown in Figure 2–13), you must take the following into consideration, depending on the HardCopy power-up option used.

- Instant on mode—The microprocessor program code must be modified to remove the configuration code relevant to the HardCopy series device. The microprocessor must delay sending the first configuration data word to the FPGA until the nCEO pin on the HardCopy series device is asserted. The microprocessor then loads the first configuration data word into the FPGA.
- Instant on after 50 ms mode—The boot-up time of the microprocessor must be greater than 50 ms. The HardCopy series device asserts the nCEO pin after the 50-ms delay which, in turn, enables the following FPGA. The microprocessor can send the first configuration data word to the FPGA after the FPGA is enabled.
- Emulation mode—This option should be used if the microprocessor code pertaining to the configuration of the above devices cannot be modified.

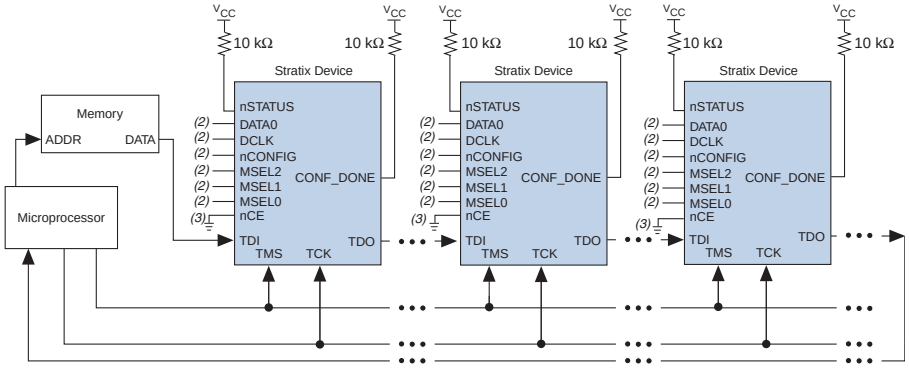
HardCopy Stratix Device Replacing FPGA Configured in a JTAG Chain

In this example, the circuit connectivity is maintained and there are no changes made to the board. The HardCopy series device can use either of the following power-up options when applicable.

- Instant on mode—Use the instant on power up mode if the microprocessor code can be modified so that it treats the HardCopy series device as a non-configurable device. The microprocessor can achieve this by issuing a BYPASS instruction to the HardCopy series device. With the HardCopy series device in BYPASS mode, the configuration data passes through it to the downstream FPGAs.
- Configuration emulation mode—Use the configuration emulation power up mode if the microprocessor code pertaining to the configuration of the above devices cannot be modified. HC1S80, HC1S60, and HC1S25 devices do not support this mode.

Figure 2–14 shows an example where there are multiple Stratix FPGAs. These devices are connected using the JTAG I/O pins for each device, and programmed using the JTAG port. An on-board microprocessor generates the configuration data.

Figure 2–14. Configuring FPGAs in a JTAG Chain Using a Microprocessor *Note (1)*



Notes to Figure 2–14:

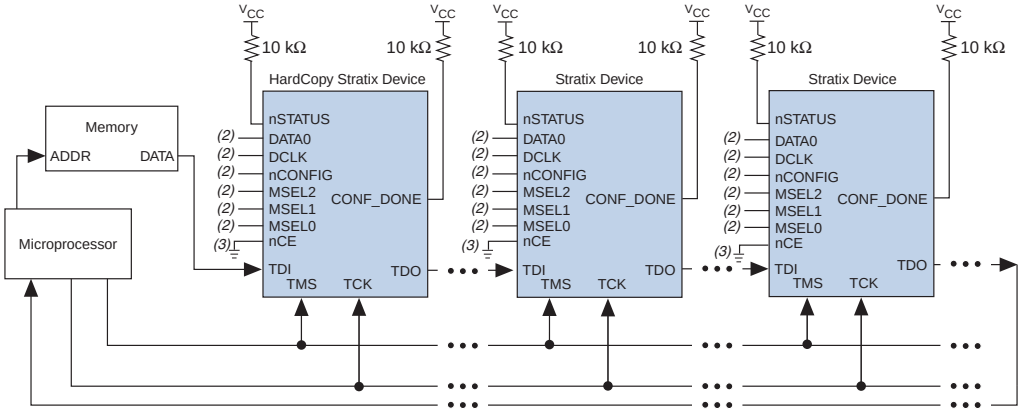
(1) Stratix II, Stratix, and APEX 20K devices can be placed within the same JTAG chain for device programming and configuration.

(2) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to either high or low.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

Figure 2–15 shows an example where the first Stratix device in the JTAG chain is replaced by a HardCopy Stratix device.

Figure 2–15. Replacement of the First FPGA in the JTAG Chain With a HardCopy Series Device *Note (1)*



Notes to Figure 2–15:

(1) Stratix II, Stratix, and APEX 20K devices can be placed within the same JTAG chain for device programming and configuration.

(2) Connect the nCONFIG, MSEL0, MSEL1, and MSEL2 pins to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to V_{CC}, and MSEL0, MSEL1, and MSEL2 to ground. Pull DATA0 and DCLK to either high or low.

(3) nCE must be connected to GND or driven low for successful JTAG configuration.

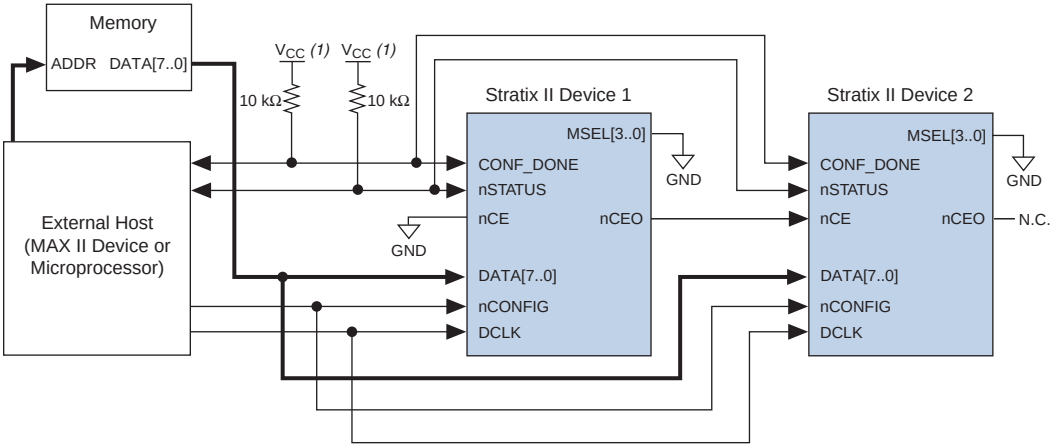
HardCopy II Device Replacing Stratix II Device Configured With a Microprocessor

When replacing a Stratix II FPGA with a HardCopy II device, the HardCopy II device can only use the instant on and instant on after 50 ms modes. This example does not require any changes to the board. However, the microprocessor code must be modified to treat the HardCopy II device as a non-configurable device.

Figure 2–16 shows an example with two Stratix II devices configured using a microprocessor or MAX® II device and the FPP configuration scheme.

For more information on Stratix II configuration, refer to the *Configuration Handbook*.

Figure 2–16. Multiple-Device FPP Configuration Using a Microprocessor or MAX II Device

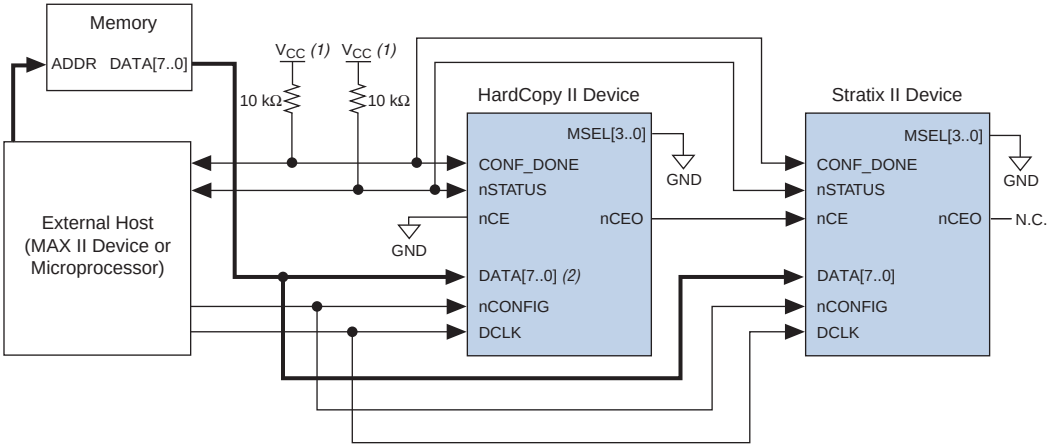


Note to Figure 2–16:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. The V_{CC} voltage meets the I/O standard's V_{IH} specification on the device and the external host.

Figure 2–17 shows how the first Stratix II device is replaced by a HardCopy II device. In this case, the microprocessor code must be modified to send configuration data only to the second device (the Stratix II device) of the configuration chain. The microprocessor can only send this data after its `nCE` pin is asserted by the first device (the HardCopy II device).

Figure 2–17. Replacement of the First FPGA in the FPP Configuration Chain With a HardCopy Series Device



Notes to Figure 2–17:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all devices in the chain. The V_{CC} voltage meets the I/O standard's V_{IH} specification on the device and the external host.

(2) The `DATA[7..0]` pins are not used on the HardCopy II device, but they preserve the pin assignment and direction from the Stratix II device, allowing drop-in replacement.

Conclusion

HardCopy series devices can emulate a configuration sequence while maintaining the seamless migration benefits of the HardCopy methodology. Instant on mode, which is the simplest of the available options, provides ASIC-like operation at power on. This mode can be used in most cases without regard to the original FPGA configuration mode and without any hardware and/or software changes.

In some cases, however, a software revision and/or a board re-design may be necessary to guarantee that correct configuration data is sent to the remaining programmable devices. Such modifications are easily made in the early stages of the board design process if it is determined that one or more of the FPGAs will be replaced with an equivalent HardCopy series device. Board-design techniques like jumper connectors and 0-Ω resistors enable such modifications without the necessity to re-design the board.

The instant on after 50 ms mode is suitable in cases where a delay is necessary to accommodate the configuration device to become operational, or to allow one or more pre-determined events to be completed before the HardCopy series device asserts its `CONF_DONE` pin.

Finally, the emulation mode is the option to choose if software or hardware modifications are not possible. In such cases, the HardCopy series device co-exists with other FPGAs.

Document Revision History

Table 2–9 shows the revision history for this chapter.

Table 2–9. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
September 2008, v2.5	Updated chapter number and metadata.	—
June 2007, v2.4	Minor text edits.	—
December 2006 v2.3	Added revision history.	—
May 2006, v2.2	• Updated Tables 20-1, 20-3, and 2-5.	—
March 2006, v2.1	• Formerly chapter 16. • Re-organized *HardCopy Power-Up Options* section to eliminate redundancy. • Updated Figures 20-1, 20-2, and 20-3. • UpdatedTables 20-1 to 20-5, and Table 20-7. • Added *Power Up Options Summary When Designing With HardCopy Series Devices* section.	—
October 2005, v2.0	Moved from Chapter 15 to Chapter 16 in Hardcopy Series Device Handbook 3.2	—

Table 2–9. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
January 2005, v2.0	• Chapter title changed to *Power-Up Modes and Configuration Emulation in HardCopy Series Devices*. • Added HardCopy II device information. • Updated external resistor requirements depending on chip configuration. • Added reference to some control and option pins that carry over functions from the FPGA design and affect the HardCopy power up. • Updated information on which HardCopy devices do not support emulation mode. • Added Table 15–9 which lists what power up options are supported by FPGAs and their HardCopy counterpart. • Added "Replacing One FPGA With One HardCopy Series Device", "Replacing One or More FPGAs With One or More HardCopy Series Devices in a Multiple-Device Configuration Chain", and "Replacing all FPGAs with HardCopy Series Devices in a Multiple-Device Configuration Chain" sections, including Tables 15-10 and 15-11, highlighting power up recommendations for each HardCopy series family.	—
June 2003, v1.0	Initial release of Chapter 15, Power-Up Modes and Configuration Emulation in HardCopy Series Devices.	—

Section II. HardCopy Design Center Migration Process

This section provides information about software support for HardCopy® Stratix® devices.

This section contains the following:

- Chapter 3, Back-End Design Flow for HardCopy Series Devices
- Chapter 4, Back-End Timing Closure for HardCopy Series Devices

Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the complete handbook.

3. Back-End Design Flow for HardCopy Series Devices

H51019-1.4

Introduction

This chapter discusses the back-end design flow executed by the HardCopy® Design Center when developing your HardCopy series device. The chapter is divided into two sections:

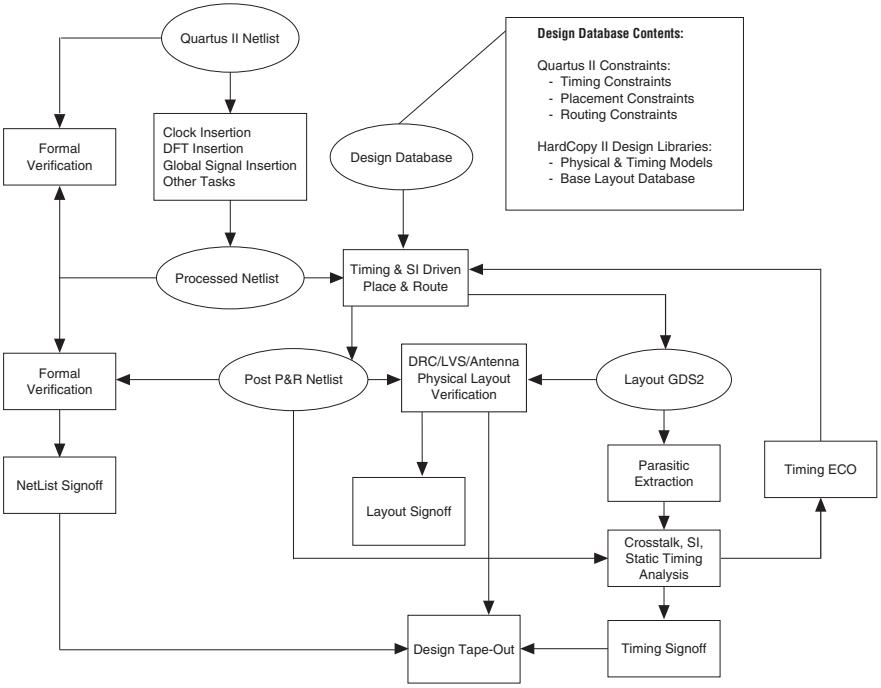
- HardCopy II Back-End Design Flow
- HardCopy Stratix® and HardCopy APEX™ Back-End Design Flow

For more information on the HardCopy II, HardCopy Stratix, and HardCopy APEX families, refer to the respective sections for these families in the *HardCopy Series Handbook*.

HardCopy II Back-End Design Flow

This section outlines the back-end design process for HardCopy II devices, which occurs in several steps. Figure 3–1 illustrates these steps. The design process uses both proprietary and third-party EDA tools. The HardCopy II device design flow is different from that of previous HardCopy families (HardCopy Stratix and HardCopy APEX devices). The following sections outline these differences.

Figure 3–1. HardCopy II Back-End Design Flow



Device Netlist Generation

For HardCopy II designs, the Quartus® II software generates a complete Verilog gate-level netlist of your design. The HardCopy Design Center uses the netlist to start the migration process. HardCopy Stratix and HardCopy APEX designs use the SRAM Object file (**.sof**) to program the FPGA, as the primary starting point for generating the HardCopy device netlist.

HardCopy Stratix and HardCopy APEX designs use the **.sof** file to program the FPGA, as the primary starting point for generating the HardCopy device netlist. In addition to the Verilog gate level netlist and the **.sof** file, the Quartus II software generates additional information as part of the design database submitted to the HardCopy Design Center. This information includes timing constraints, placement constraints, global routing information, and much more. Generation of this database provides the HardCopy Design Center with the necessary information to complete the design of your HardCopy II device.

Design for Testability Insertion

The HardCopy Design Center inserts the necessary test structures into the HardCopy II Verilog netlist. These test structures include full-scan capable registers and scan chains, JTAG, and memory testing. After adding the test structures, the modified netlist is verified using third-party EDA formal verification software against the original Verilog netlist to ensure that the test structures have not broken your netlist functionality. The "Formal Verification of the Processed Netlist"section explains the formal verification process.

Clock Tree and Global Signal Insertion

Along with adding testability, the HardCopy Design Center adds an additional local layer of clock tree buffering to connect the global clock resources to the locally placed registers in the design. Global signals with high fan-out may also use dedicated Global Clock Resources built into the base layers of all HardCopy II devices. The HardCopy Design Center does local buffering.

Formal Verification of the Processed Netlist

After all design-for-testability logic, clock tree buffering, and global signal buffering are added to the processed netlist, the HardCopy Design Center uses third-party EDA formal verification software to compare the processed netlist with your submitted Verilog netlist generated by the Quartus II software. Added test structures are constrained to bypass mode during formal verification to verify that your design's intended functionality was not broken.

Timing and Signal Integrity Driven Place and Route

Placement and global signal routing is principally done in the Quartus II software before submitting the HardCopy II design to the HardCopy Design Center. Using the Quartus II software, you control the placement and timing driven placement optimization of your design. The Quartus II software also does global routing of your signal nets, and passes this information in the design database to the HardCopy Design Center to do the final routing. After submitting the design to the HardCopy Design Center, Altera® engineers use the placement and global routing information provided in the design database to do final routing and timing closure and to perform signal integrity and crosstalk analysis. This may require buffer and delay cell insertion in the design through an engineering change order (ECO). The resulting post-place and route netlist is verified again with the source netlist and the processed netlist to guarantee that functionality was not altered in the process.

Parasitic Extraction and Timing Analysis

After doing placement and routing on the design by the HardCopy Design Center, it generates the `gds2` design file and extracts the parasitic resistance and capacitance values for timing analysis. Parasitic extraction uses the physical layout of the design stored in a **.gds2** file to extract these resistance and capacitance values for all signal nets in the design. The HardCopy Design Center uses these parasitic values to calculate the path delays through the design for static timing analysis and crosstalk analysis.

Layout Verification

When the Timing Analysis reports that all timing requirements are met, the design layout goes into the final stage of verification for manufacturability. The HardCopy Design Center performs physical Design Rule Checking (DRC), antenna checking of long traces of signals in the layout, and a comparison of layout to the design netlist, commonly referred to as Layout Versus Schematic (LVS). These tasks guarantee that the layout contains the exact logic represented in the place-and-route netlist, and the physical layout conforms to 90-nm manufacturing rules.

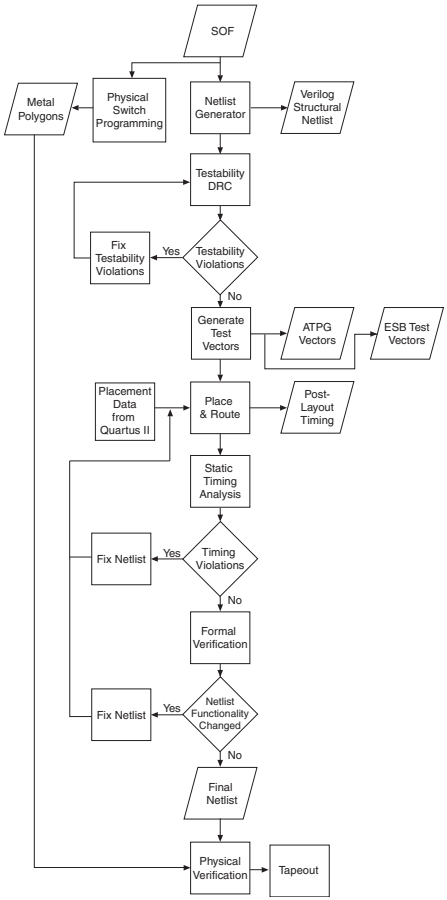
Design Signoff

The Altera HardCopy II back-end design methodology has a thorough verification and signoff process, guaranteeing your design's functionality. Signoff occurs after confirming the final place-and-route netlist functional verification, confirming layout verification for manufacturability, and the timing analysis reports meeting all requirements. After achieving all three signoff points, Altera begins the manufacturing of the HardCopy II devices.

HardCopy Stratix and HardCopy APEX Migration Flow

Design migration for HardCopy Stratix and HardCopy APEX devices occurs in several steps, outlined in this section and shown in Figure 3–2. The migration process uses both proprietary and third-party EDA tools.

Figure 3–2. HardCopy Stratix and HardCopy APEX Migration Flow Diagram



Netlist Generation

For HardCopy Stratix and HardCopy APEX designs, Altera migrates the Quartus II software-generated **.sof** file to a Verilog HDL structural netlist that describes how the following structural elements are configured in the design and how each structural element is connected to other structural elements:

- Logic element (LE)
- Phase-locked loop (PLL)
- Digital signal processing (DSP) block
- Memory block
- Input/output element (IOE)

The information that describes the structural element configuration is converted into a physical coordinate format so that metal elements can be implemented on top of the pre-defined HardCopy series device-base array. Using the **.sof** file for netlist extraction helps ensure that the HardCopy series device contains the same functional implementation that was used in the FPGA version of the design.

Testability Audit

The Design Center performs an audit for testability violations when the Verilog HDL netlist is available. This audit ensures that all built-in scan chain structures will work reliably while testing the HardCopy series devices. Certain circuit structures, such as gated clocks, gated resets, oscillators, pulse generators, or other types of asynchronous circuit structures makes the performance of scan chain structures unreliable. During the testability audit, all such circuit structures are detected and disabled when the device is put into test mode.

Placement

Beginning with version 4.2, the Quartus II software supports all HardCopy series devices. The HardCopy Timing Optimization Wizard in the Quartus II software is used for HardCopy Stratix devices and generates placement information of the design when it is mapped to the HardCopy Stratix base array. This placement information is read in and directly used by the place-and-route tool during migration to the equivalent HardCopy Stratix device.

For more information on how to use the HardCopy Timing Optimization Wizard, refer to the *Quartus II Support for HardCopy Stratix Devices* chapter. For more information on Quartus II features for HardCopy II devices, refer to the *Quartus II Support for HardCopy II Devices* chapter.

To generate placement data, the Quartus II software uses the **.sof** file to generate the netlist, as described in "Netlist Generation" on page 3–6. The netlist is then read into a place-and-route tool. The placement optimization is based on the netlist connectivity and the design's timing constraints. The placement of all IOEs is fixed. After placement is complete, the Quartus II software generates the scan chain ordering information so the scan paths can be connected.

Test Vector Generation

Memory test vectors and memory built-in self-test (BIST) circuitry ensure that all memory bits function correctly. Automatic test pattern generation (ATPG) vectors test all LE, DSP, and IOE logic. These vectors ensure that a high stuck-at-fault coverage is achieved. The target fault coverage for all HardCopy Stratix devices is near 100%.

When the testability audit is successfully completed and the scan chains have been re-ordered, the Design Center can generate memory and ATPG test vectors. When test vector generations are complete, they are simulated to verify their correctness.

Routing

Routing involves generating the physical interconnect between every element in the design. At this stage, physical design rule violations are fixed. For example, nodes with large fan-outs need to be buffered. Otherwise, these signal transition times are too slow, and the device's power consumption increases. All other types of physical design rule violations are also fixed during this stage, such as antenna violations, crosstalk violations, and metal spacing violations.

Extracted Delay Calculation

Interconnect parasitic capacitance and resistance information is generated after the routing is complete. This information is then converted into a Standard Delay File (**.sdf)** with a delay calculation tool, and timing is generated for minimum and maximum delays.

Static Timing Analysis and Timing Closure

The design timing is checked and corrected after place and route using the post-layout generated **.sdf** file. Setup time violations are corrected in two ways. First, extra buffers can be inserted to speed up slow signals. Second, if buffer insertion does not completely fix the setup violation, the placement can be re-optimized.

Setup time violations are rare in HardCopy II and HardCopy Stratix devices because the die sizes are considerably smaller than the equivalent Stratix II and Stratix devices. Statistically, the interconnect loading and distance is much smaller in HardCopy Stratix devices, so the device operates at a higher clock frequency. Hold-time violations are fixed by inserting delay elements into fast data paths.

As part of the timing analysis process, crosstalk analysis is also performed to remove any crosstalk effects that could be encountered in the device after it has been manufactured. This ensures signal integrity in the device resulting in proper functionality and satisfactory performance.

After implementing all timing violation corrections in the netlist, the place and route is updated to reflect the changes. This process is repeated until all timing violations are removed. Typically, only a single iteration is required after the initial place and route. Finally, static functional verification is tested after this stage to double-check the netlist integrity.

Formal Verification

After any change to the netlist, you must verify its integrity through static functional verification (or formal verification) techniques. These techniques show whether two versions of a design are functionally identical when certain constraints are applied. For example, after test fixes, the netlist must be logically equivalent to the netlist state before test fixes, when the test mode is disabled. This technique does not rely on any customer-supplied functional simulation vectors. Altera uses third-party formal verification software to confirm that the back-end implementation matches the netlist generated from the FPGA's **.sof** programming file.

Physical Verification

Before manufacturing the metal customization layers, the physical programming information must be verified. This stage involves cross-checking for physical design rule violations in the layout database, and also checking that the circuit was physically implemented correctly. These processes are commonly known as running design rule check and layout-versus-schematic verification.

Manufacturing

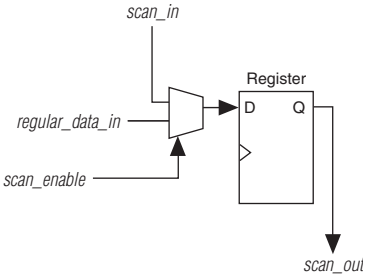
Metallization masks are created to manufacture HardCopy series devices. After manufacturing, the parts are tested using the test vectors that were developed as part of the implementation process.

Testing

HardCopy series devices are fully tested as part of the manufacturing process. Testing does not require user-specific simulation vectors, because every HardCopy series device utilizes full scan path technology. This means that every node inside the device is both controllable and observable through one or more of the package pins of the device. The scan paths, or "scan chains," are exercised through ATPG. This ensures a high-confidence level in the detection of all manufacturing defects.

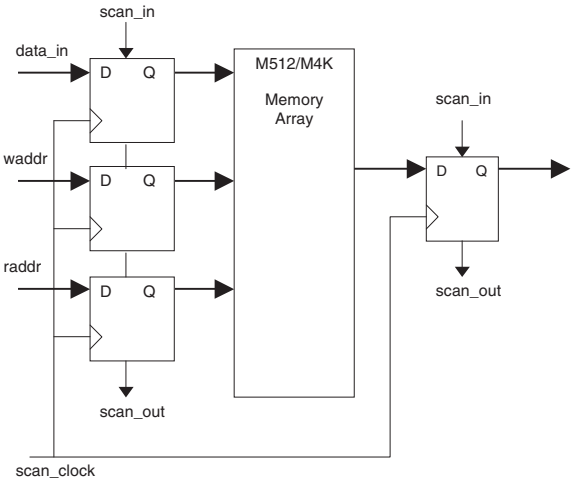
Every register in the HardCopy series device belongs to a scan chain. Scan chains are test features that exist in ASICs to ensure that there is access to all internal nodes of a design. With scan chains, defective parts can be screened out during the manufacturing process. Scan chain registers are constructed by combining the original FPGA register with a 2-to-1 multiplexer. In normal user mode, the multiplexer is transparent to the user. In scan mode, the registers in the device are connected into a long-shift register so that automatic test pattern generation vectors can be scanned into and out of the device. Several independent scan chains exist in the HardCopy series device to keep scan chain lengths short, and are run in parallel to keep tester time per device short. Figure 3–3 shows a diagram of a scan register.

Figure 3–3. HardCopy Stratix Scan Chain Circuitry



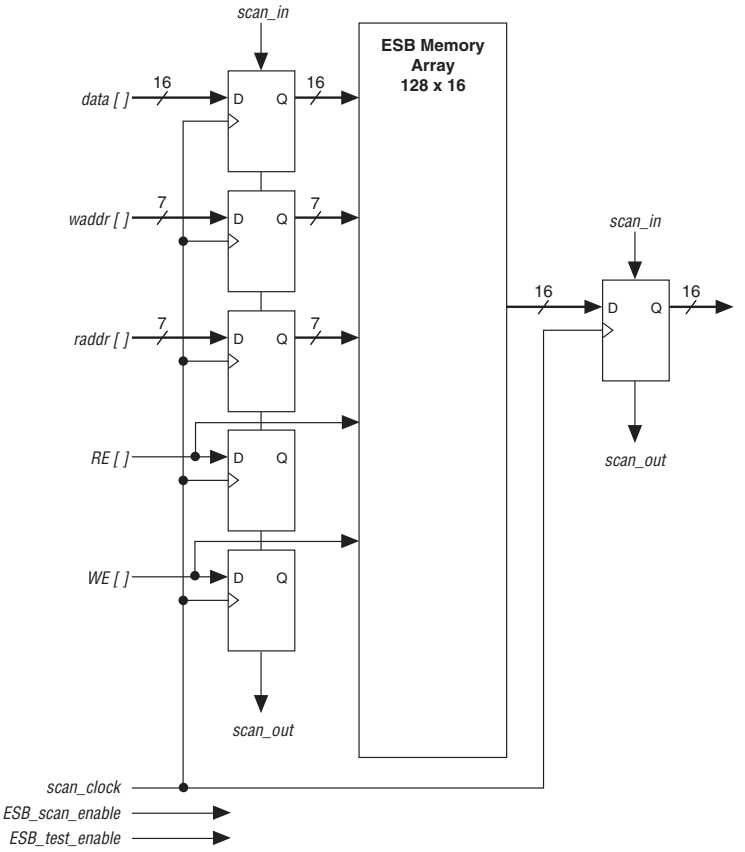
In addition to the scan circuitry (Figure 3–3), which is designed to test all LEs and IOEs, both M512 and M4K blocks (Figure 3–4) have the same scan chain structure so that all bits inside the memory array are tested for correct operation. The M512 and M4K RAM bits are tested by scanning data into the M512 and M4K blocks' `data_in`, write address (`waddr`), and read address (`raddr`) registers. After each vector has been scanned into the HardCopy Stratix device, a write enable (`WE`) pulse is generated to write the data into the M512 and M4K blocks. A read enable (`RE`) pulse is also generated to read data out of the M512 and M4K blocks. The data read back from the M512 and M4K blocks are scanned out of the device via the `data_out` registers. Figure 3–4 shows the M512 and M4K blocks' scan chain connectivity.

Figure 3–4. HardCopy Stratix M512 and M4K Block Scan Chain Connectivity



For HardCopy APEX devices, every embedded system block (ESB) contains dedicated test circuitry so that all bits inside the memory array are tested for correct operation. Access to the ESB memory is also facilitated through scan chains. The ESB also offers an ESB test mode in which the ESB is reconfigured into a 128 × 16 RAM block. In this mode, data is scanned into the ESB I/O registers and written into the ESB memory. For ESBs configured as product-term logic or ROM, the write-enable signal has no effect on the ESB memory array data. When the test mode is disabled (the default), the ESB reverts to the desired user functionality. Figure 3–5 shows the ESB test mode configuration.

Figure 3–5. HardCopy APEX ESB Test Mode Configuration



PLLs and M-RAM blocks are tested with BIST circuitry and test point additions. All test circuitry is disabled once the device is installed into the end user system so that the device then behaves in the expected normal functional mode.

Unused Resources

Unused resources in a customer design still exist in the HardCopy base. However, these resources are configured into a "parked" state. This is a state where all input pins of an unused resource are tied off to V_{CC} or GND so that the resource is in a low-power state. This is achieved using the same metal layers that are used to configure and connect all resources used in the design.

Conclusion

The HardCopy series back-end design methodology ensures that your design seamlessly migrates from your prototype FPGA to a HardCopy device. This methodology, matched with Altera's unique FPGA prototyping and migration process, provides an excellent way for you to develop your design for production.

For more information about how to start building your HardCopy series design, contact your Altera Field Applications Engineer.

For more information on HardCopy products and solutions, refer to the *HardCopy Series Handbook*.

Document Revision History

Table 3–1 shows the revision history for this chapter.

Table 3–1. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
September 2008, v1.4	Update chapter number and metadata.	—
June 2007, v1.3	Minor text edits.	—
December 2006 v1.2	Added revision history.	—
March 2006	Formerly chapter 13; no content change.	—
October 2005 v1.1	• Graphic updates • Minor edits	—
January 2005 v1.0	Initial release of Chapter 13, Back-End Design Flow for HardCopy Series Devices.	—

4. Back-End Timing Closure for HardCopy Series Devices

H51013-2.4

Introduction

Back-end implementation of HardCopy® series devices meet design requirements through a timing closure process similar to the methodology used for today's standard cell ASICs.

The Quartus® II software provides a pre-layout estimation of your HardCopy design performance and then the Altera® HardCopy Design Center uses industry leading EDA software to complete the back-end layout and extract the final timing results prior to tape-out.

For more information on the HardCopy back-end design flow, refer to the *HardCopy Series Back-End Design Flow* chapter in the *HardCopy Series Device Handbook*.

This chapter describes how Altera ensures that HardCopy series devices meet their required timing performance.

Timing Analysis of HardCopy Prototype Device

You should perform timing analysis on the FPGA prototype implementation of the design before migrating to HardCopy. For HardCopy II designs, timing analysis should also be performed after successfully fitting the design in a HardCopy II device with Quartus II software. Timing analysis determines whether the design's performance meets the required timing goals.

The timing analysis must be done for both setup and hold time checks on all design paths, including internal paths and input and output paths. Measuring these parameters against performance goals ensures that the FPGA design functions as planned in the end-target system.

For more information on timing analysis of Altera devices, refer to the *Timing Analysis* section in volume 3 of the *Quartus II Handbook*.

After the FPGA design is stabilized, fully tested in-system and satisfies the HardCopy series design rules, the design can be migrated to a HardCopy series device. Altera performs rigorous timing analysis on the HardCopy series device during its implementation, ensuring that it meets the required timing goals. Because the critical timing paths of the HardCopy version of a design may be different from the corresponding paths in the FPGA version, meeting the required timing goals constrained in the Quartus II software is particularly important. Additional

performance gains are design dependent, and the percentage of performance improvement can be different for each clock domain of your design.

Timing differences between the FPGA design and the equivalent HardCopy series device can exist for several reasons. While maintaining the same set of features as the corresponding FPGA, HardCopy series devices have a highly optimized die size to make them as small as possible. Because of the customized interconnect structure that makes this optimization possible, the delay through each signal path is different from the original FPGA design.

Cell Structure

Meeting system timing goals in an ASIC design can be very challenging and can easily consume many months of engineering effort. The slower development process exists because, in today's silicon technology (0.18 µm, 0.13 µm, and 90 nm), the delay associated with interconnect dominates the delay associated with the transistors used to make the logic gates. Consequently, ASIC performance is sensitive to the physical placement and routing of the logic blocks that make up the design.

HardCopy II

HardCopy II devices use timing constraints to drive placement and routing of logic into the fabric of HCells. Each Stratix II Adaptive Look-up Table (ALUT) is implemented in HCell Macros in the HardCopy II device. HCell Macros are pre-defined and characterized libraries built out of HCells. The Quartus II software performs the placement and global routing of all HCell Macros and this information is forward-annotated to the HardCopy Design Center for final back-end implementation and timing closure.

HardCopy Stratix, HardCopy APEX

HardCopy Stratix® and HardCopy APEX™ are structurally identical to their respective FPGA counterparts. There is no re-synthesis or library re-mapping required. Since the interconnect lengths are much smaller in the HardCopy series device than they are in the FPGA, the place-and-route engine compiling the HardCopy series design has a considerably less difficult task than it does in an equivalent ASIC development. Coupled with detailed timing constraints, the place-and-route is timing driven.

Clock Tree Structure

The following section describes the clock tree structure for the HardCopy device family.

HardCopy II

HardCopy II devices offer a fine-grained architecture of HCells which are used to build HCell Macros for standard logic functions. The pre-built metal layers of HardCopy II devices contain the same global clock tree resources as those available in Stratix II devices, though they are smaller in HardCopy II devices because of the difference in die size. The top levels of the dedicated global clock networks in HardCopy II are pre-routed in the non-custom metal layers. The lowest level of clock tree buffering and routing is done using custom metal routing. Local buffering can be done using HCell Macros to fix any clock skew issues. HCell Macros are used to create registers, and local custom routing is needed to connect the clock networks to these HCell Macro registers. These tasks are performed as part of the HardCopy Design Center process.

HardCopy Stratix

HardCopy Stratix devices have the same global clock tree resources as Stratix FPGA devices. The construction of non-customizable layers of silicon minimizes global clock tree skew. HardCopy Stratix devices with clock trees using global clock resources have smaller clock insertion delay than Stratix FPGA devices because the HardCopy Stratix devices have a smaller die area. The use of clock tree synthesis to build small localized clock trees using the existing buffer resources in HardCopy Stratix devices automatically implements clock trees using fast regional clock resources in Stratix FPGA devices.

HardCopy APEX

The HardCopy APEX device architecture is based on the APEX 20KE and APEX 20KC devices. The same dedicated clock trees (`CLK[3..0]`) that exist in APEX 20KE and APEX 20KC devices also exist in the corresponding HardCopy APEX device. These clock trees are carefully designed and optimized to minimize the clock skew over the entire device. The clock tree is balanced by maintaining the same loading at the end of each point of the clock tree, regardless of what resources (logic elements [LEs], embedded system blocks [ESBs], and input/output elements [IOEs]) are used in any design. The insertion delay of the HardCopy APEX dedicated clock trees is marginally faster than in the corresponding APEX 20KE or APEX 20KC FPGA device because of the smaller footprint of the HardCopy device silicon. This difference is less than 1 ns.

Because there is a large area overhead for the global signals that may not be used on every design, the FAST bidirectional pins (FAST[3..0]) do not have dedicated pre-built clock or buffer trees in HardCopy APEX devices. If any of the FAST signals are used as clocks, the place-and-route tool synthesizes a clock tree after the placement of the design has occurred. The skew and insertion delay of these synthesized clock trees is carefully controlled, ensuring that the timing requirements of the design are met. You can also use the FAST signals as high fan-out reset or enable signals. For these cases, skew is usually less important than insertion delay. To reiterate, a buffer tree is synthesized after the design placement.

The clock or buffer trees that are synthesized for the FAST pins are built out of special cells in the HardCopy APEX base design. These cells do not exist in the FPGA, and they are used in the HardCopy APEX design exclusively to meet timing and testing goals. They are not available to make any logical changes to the design as implemented in the FPGA. These resources are called the strip of auxiliary gates (SOAG). There is one strip per MegaLAB™ structure in HardCopy APEX devices. Each SOAG consists of a number of primitive cells, and there are approximately 10 SOAG primitive cells per logic array block (LAB). Several SOAG primitives can be combined to form more complex logic, but the majority of SOAG resources are used for buffer tree, clock tree, and delay cell generation.

For detailed information on the HardCopy APEX series device architecture, including SOAG resources, refer to the *HardCopy APEX Device Family Data Sheet* section in volume 1 of the *HardCopy Series Handbook*.

Importance of Timing Constraints

After capturing the information, Altera directly checks all timing of the HardCopy series device before tape-out occurs. It is important to constrain the FPGA and HardCopy devices for the exact timing requirements that need to be achieved. Timing violations seen in the Quartus II project or in the HardCopy Design Center migration process must be fixed or waived prior to the design being manufactured.

Correcting Timing Violations

After generating the customized metal interconnect for the HardCopy series device, Altera checks the design timing with a static timing analysis tool. The static timing analysis tool reports timing violations and then the HardCopy Design Center corrects the violations.

Hold-Time Violations

Because the interconnect in a HardCopy series device is customized for a particular application, it is possible that hold-time (tH) violations exist in the HardCopy series device after place-and-route occurs. A hold violation exists if the sum of the delay in the clock path between two registers plus the micro hold time of the destination register is greater than the delay of the data path from the source register to the destination register. The following equation describes this relationship.

tH slack = data delay - clock delay - µtH

If a negative slack value exists, a hold-time violation exists. Any hold-time violation present in the HardCopy series design database after the interconnect data is generated is removed by inserting the appropriate delay in the data path. The inserted delay is large enough to guarantee no hold violation under fast, low-temperature, high-voltage conditions.

An Example HardCopy APEX Hold-Time Violation Fix

Table 4–1 shows an example report of a Synopsys PrimeTime static timing analysis of a HardCopy APEX design. The first report shows that the circuit has a hold-time violation and a negative slack value. The second result shows the timing report for the same path after fixing the hold violation. Part of the HardCopy implementation process is to generate the instance and cell names shown in these reports. The physical location of those elements in the device determines the generation of the names.

Table 4–1. HardCopy APEX Static Timing Analysis Before Hold-Time Violation Fix

```
Startpoint: GR23_GC0_L19_LE1/um6
(falling edge-triggered flip-flop clocked by CLK0')
Endpoint: GR23_GC0_L20_LE8/um6
(falling edge-triggered flip-flop clocked by CLK0')
Path Group: CLK0
Path Type: min
```

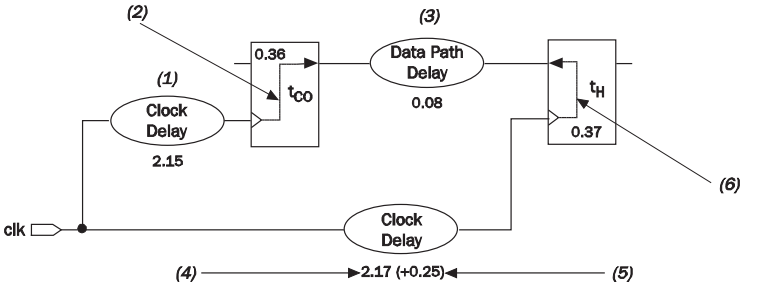
Point	Incr	Path	Reference Point *(1)*
clock CLK0' (fall edge)	0.00	0.00	
clock network delay (propagated)	2.15	2.15	(1)
GR23_GC0_L19_LE1/um6/clk (c1110)	0.00	2.15 f	(2)
GR23_GC0_L19_LE1/um6/regout (c1110)	0.36 *	2.52 r	(2)
GR23_GC0_L19_LE1/REGOUT (c1000_2d7a8)	0.00	2.52 r	(2)
GR23_GC0_L20_LE8/LUTD (c1000_56502)	0.00	2.52 r	(3)
GR23_GC0_L20_LE8/um1/datad (indsim)	0.01 *	2.52 r	(3)
GR23_GC0_L20_LE8/um1/ndsim (indsim)	0.01 *	2.53 f	(3)
GR23_GC0_L20_LE8/um5/ndsim (mxcascout)	0.00 *	2.53 f	(3)
GR23_GC0_L20_LE8/um5/cascout	0.06 *	2.59 f	(3)
GR23_GC0_L20_LE8/um6/dcout (c1110)	0.00 *	2.59 f	(3)
data arrival time		2.59	
clock CLK0' (fall edge)	0.00	0.00	
clock network delay (propagated)	2.17	2.17	(4)
clock uncertainty	0.25	2.42	(5)
GR23_GC0_L20_LE8/um6/clk (c1110)		2.42 f	(6)
library hold time	0.37 *	2.79	
data required time		2.79	
data arrival time		2.59	
data required time		-2.79	
slack (VIOLATED)		-0.20	

Note to Table 4–1:

(1) This column does not exist in the actual report. It is included in this document to provide corresponding reference points to Figure 4–1.

Figure 4–1 shows the circuit described by the Table 4–1 static timing analysis report.

Figure 4–1. Circuit With a Hold-Time Violation



Placing the values from the static timing analysis report into the hold-time slack equation results in the following:

$$t_H \text{ slack} = \text{data delay} - \text{clock delay} - \mu t_H$$

$$t_H \text{ slack} = (2.15 + 0.36 + 0.08) - (2.17 + 0.25) - 0.37$$

$$t_H \text{ slack} = -0.20 \text{ ns}$$

This result shows that there is negative slack in this path, meaning that there is a hold-time violation of 0.20 ns.

After fixing the hold violation, the timing report for the same path is re-generated (Table 4–2). The netlist changes are in ***bold italic*** type.

Table 4–2. HardCopy APEX Static Timing Analysis After Hold-Time Violation Fix

```
Startpoint: GR23_GC0_L19_LE1/um6
(falling edge-triggered flip-flop clocked by CLK0')
Endpoint: GR23_GC0_L20_LE8/um6
(falling edge-triggered flip-flop clocked by CLK0')
Path Group: CLK0
Path Type: min
Static Timing Analysis After Hold-Time Violation Fix
```

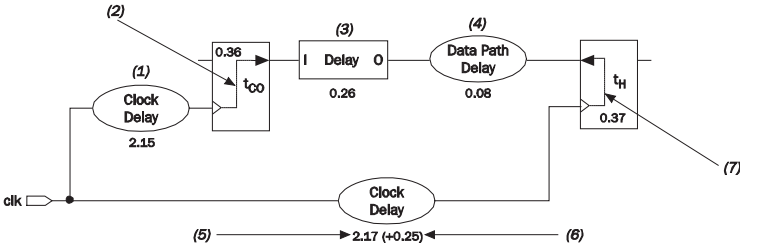
Point	Incr	Path	Reference Point *(1)*
clock CLK0' (fall edge)	0.00	0.00	(1)
clock network delay (propagated)	2.15	2.15	(1)
GR23_GC0_L19_LE1/um6/clk (c1110)	0.00	2.15 f	(2)
GR23_GC0_L19_LE1/um6/regout (c1110)	0.36 *	2.52 r	(2)
GR23_GC0_L19_LE1/REGOUT (c1000_2d7a8)	0.00	2.52 r	(2)
thc_916/A (de105)	***0.01 ****	***2.52 r***	(3)
thc_916/Z (de105)	***0.25 ****	***2.78 r***	(3)
GR23_GC0_L20_LE8/LUTD (c1000_56502)	0.00	2.78 r	(3)
GR23_GC0_L20_LE8/um1/datad (indsim)	0.01 *	2.78 r	(3)
GR23_GC0_L20_LE8/um1/ndsim (indsim)	0.01 *	2.79 f	(3)
GR23_GC0_L20_LE8/um5/ndsim (mxcascout)	0.00 *	2.79 f	(3)
GR23_GC0_L20_LE8/um5/cascout (mxcascout)	0.06 *	2.85 f	(3)
GR23_GC0_L20_LE8/um6/dcout (c1110)	0.00 *	2.85 f	(3)
data arrival time		2.85	
clock CLK0' (fall edge)	0.00	0.00	
clock network delay (propagated)	2.17	2.17	(4)
clock uncertainty	0.25	2.42	(5)
GR23_GC0_L20_LE8/um6/clk (c1110)		2.42 f	(6)
library hold time	0.37 *	2.79	
data required time		2.79	
data arrival time		2.85	
data required time		-2.79	
slack (MET)		0.06	

Note to Table 4–2:

(1) This column does not exist in the actual report. It is included in this document to provide corresponding reference points to Figure 4–2.

Figure 4–2 shows the circuit described by the Table 4–2 static timing analysis report.

Figure 4–2. Circuit Including a Fixed Hold-Time Violation



Placing the values from the static timing analysis report into the hold-time slack equation, results in the following.

$$t_H \text{ slack} = \text{data delay} - \text{clock delay} - \mu t_H$$

$$t_H \text{ slack} = (2.15 + 0.36 + 0.26 + 0.08) - (2.17 + 0.25) - 0.37$$

$$t_H \text{ slack} = + \ 0.06 \text{ ns}$$

In this timing report, the slack of this path is reported as 0.06 ns. Therefore, this path does not have a hold-time violation. This path was fixed by the insertion of a delay cell (`del05`) into the data path, which starts at the `REGOUT` pin of cell `GR23_GC0_L19_LE1` and finishes at the `LUTD` input of cell `GR23_GC0_L20_LE8`. The instance name of the delay cell in this case is `thc_916`.

This timing report specifies a clock uncertainty of 0.25 ns, and adds extra margin during the hold-time calculation, making the design more robust. This feature is a part of the static timing analysis tool, not of the HardCopy series design.

The SOAG resources that exist in the HardCopy APEX base design create the delay cell. The HardCopy Stratix base design contains auxiliary buffer cells of varying drive strength used to fix setup and hold time violations.

Setup-Time Violations

A setup violation exists if the sum of the delay in the data path between two registers plus the micro setup time (t_{SU}) of the destination register is greater than the sum of the clock period and the clock delay at the destination register. The following equation describes this relationship:

$$t_{SU} \text{ slack} = \text{clock period} + \text{clock delay} - (\text{data delay} + \mu t_{SU})$$

If there is a negative slack value, a setup-time violation exists. Several potential mechanisms can cause a setup-time violation. The first is when the synthesis tool is unable to meet the required timing goals. However, a HardCopy series design does not rely on any re-synthesis to a new cell library; synthesis results are generated as part of the original FPGA design, meaning that the HardCopy implementation of a design uses exactly the same structural netlist as its FPGA counterpart. For example, if you used a particular synthesis option to ensure that a particular path only contain a certain number of logic levels, the HardCopy series design contains exactly the same number of logic levels for that path. Consequently, if the FPGA was free of setup-time violations, no setup-time violations will occur in the HardCopy series device due to the netlist structure.

The second mechanism that can cause setup-time violations is differing placement of the resources in the netlist for the HardCopy series device compared to the original FPGA. This scenario is extremely unlikely as the place-and-route tool used during the HardCopy implementation performs timing-driven placement. In extreme cases, some manual placement modifications are necessary. The placement is performed at the LAB and ESB level, meaning that the placement of logic cells inside each LAB is fixed, and is identical to the placement of the FPGA. IOEs have fixed placement to maintain the pin and package compatibility of the original FPGA.

The third, and most likely, mechanism for setup-time violations occurring in the HardCopy series device is a signal with a high fan-out. In the FPGA, high fan-out signals are buffered by large drivers that are integral parts of the programmable interconnect structure. Consequently, a signal that was fast in the FPGA can be initially slower in the HardCopy version. The place-and-route tool detects these signals and automatically creates buffer trees using SOAG resources, ensuring that the heavily loaded, high fan-out signal is fast enough to meet performance requirements.

An Example HardCopy APEX Setup-Time Violation Fix

Table 4–3 shows the timing report for a path in a HardCopy APEX design that contains a high fan-out signal before the place-and-route process. Table 4–4 shows the timing report for a path that contains a high fan-out signal after the place-and-route process. Before the place-and-route process, there is a large delay on the high fan-out net driven by the pin `GR12_GC0_L2_LE4/REGOUT`. This delay is due to the large capacitive load that the pin has to drive. Figure 4–3 shows the timing report information.

Table 4–3. HardCopy APEX Timing Report Before Place-and-Route Process

```
Startpoint: GR12_GC0_L2_LE4/um6
(falling edge-triggered flip-flop clocked by clkx')
Endpoint: GR4_GC0_L5_LE2/um6
(falling edge-triggered flip-flop clocked by clkx')
Path Group: clkx
Path Type: max
```

Point	Incr	Path	Reference Point *(1)*
clock clkx' (fall edge)	0.00	0.00	(1)
clock network delay (propagated)	2.18	2.18	(1)
GR12_GC0_L2_LE4/um6/clk (c1110)	0.00	2.18 f	(2)
GR12_GC0_L2_LE4/um6/regout (c1110)			(2)
GR12_GC0_L2_LE4/REGOUT (c1000_7f802) <-			(2)
GR4_GC0_L5_LE0/LUTC (c1000_0029a)			(3)
GR4_GC0_L5_LE0/um4/ltb (lt53b)	2.36	9.18 f	(3)
GR4_GC0_L5_LE0/um5/cascout (mxcascout)	0.07	9.24 f	(3)
GR4_GC0_L5_LE0/um2/COMBOUT (icombout)	0.09	9.34 r	(3)
GR4_GC0_L5_LE0/COMBOUT (c1000_0029a)	0.00	9.34 r	(3)
GR4_GC0_L5_LE2/LUTC (c1000_0381a)	0.00	9.34 r	(3)
GR4_GC0_L5_LE2/um4/ltb (lt03b)	0.40	9.73 r	(3)
GR4_GC0_L5_LE2/um5/cascout (mxcascout)	0.05	9.78 r	(3)
GR4_GC0_L5_LE2/um6/dcout (c1110)	0.00	9.78 r	(3)
data arrival time		9.79	(3)
clock clkx' (fall edge)	7.41	7.41	
clock network delay (propagated)	2.18	9.59	(4)
clock uncertainty	-0.25	9.34	(5)
GR4_GC0_L5_LE2/um6/clk (c1110)		9.34 f	

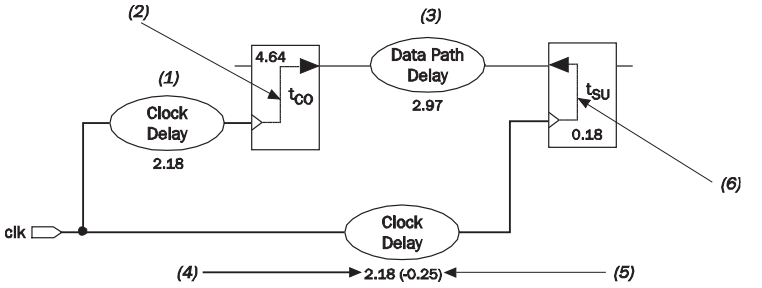
Point	Incr	Path	Reference Point *(1)*
library setup time	-0.18	9.16	(6)
data required time		9.16	
data required time		9.16	
data arrival time		-9.79	
slack (VIOLATED)		-0.63	

Note to Table 4–3:

(1) This column does not exist in the actual report. It is included in this document to provide corresponding reference points to Figure 4–3.

Figure 4–3 shows the circuit that Table 4–3 static timing analysis report describes.

Figure 4–3. Circuit That Has a Setup-Time Violation



The timing numbers in this report are based on pre-layout estimated delays.

Placing the values from the static timing analysis report into the set-up time slack equation, results in the following.

$$t_{SU} \text{ slack} = \text{clock period} + \text{clock delay} - (\text{data delay} + \mu t_{SU})$$

$$t_{SU} \text{ slack} = 7.41 + (2.18 - 0.25) - (2.18 + 4.64 + 2.97 + 0.18)$$

$$t_{SU} \text{ slack} = -0.63 \text{ ns}$$

This result shows that there is negative slack for this path, meaning that there is a setup-time violation of 0.63 ns.

After place-and-route, a buffer tree is constructed on the high fan-out net and the setup-time violation is fixed. Table 4–4 shows the timing report for the same path. The changes to the netlist are in ***bold italic*** type. Figure 4–4 shows more information on this timing report.

Table 4–4. HardCopy APEX Timing Report After the Place-and-Route Process

```
Startpoint: GR12_GC0_L2_LE4/um6
(falling edge-triggered flip-flop clocked by clkx')
Endpoint: GR4_GC0_L5_LE2/um6
(falling edge-triggered flip-flop clocked by clkx')
Path Group: clkx
Path Type: max
```

Point	Incr	Path	Reference Point *(1)*
clock clkx' (fall edge)	0.00	0.00	
clock network delay (propagated)	2.73	2.73	(1)
GR12_GC0_L2_LE4/um6/clk (c1110)	0.00	2.73 f	(2)
GR12_GC0_L2_LE4/um6/regout (c1110)	0.69 *	3.42 r	(2)
GR12_GC0_L2_LE4/REGOUT (c1000_7f802) <-	0.00	3.42 r	(2)
N1188_iv06_1_0/Z (iv06)	***0.06 ****	***3.49 f***	(3)
N1188_iv06_2_0/Z (iv06)	***0.19 ****	***3.68 r***	(3)
N1188_iv06_3_0/Z (iv06)	***0.12 ****	***3.80 f***	(3)
N1188_iv06_4_0/Z (iv06)	***0.10 ****	***3.90 r***	(3)
N1188_iv06_5_0/Z (iv06)	***0.08 ****	***3.97 f***	(3)
N1188_iv06_6_2/Z (iv06)	***1.16 ****	***5.13 r***	(3)
GR4_GC0_L5_LE0/LUTC (c1000_0029a)	0.00	5.13 r	(4)
GR4_GC0_L5_LE0/um4/ltb (lt53b)	1.55 *	6.68 f	(4)
GR4_GC0_L5_LE0/um5/cascout (mxcascout)	0.06 *	6.74 f	(4)
GR4_GC0_L5_LE0/um2/COMBOUT (icombout)	0.09 *	6.84 r	(4)
GR4_GC0_L5_LE0/COMBOUT (c1000_0029a)	0.00	6.84 r	(4)
GR4_GC0_L5_LE2/LUTC (c1000_0381a)	0.00	6.84 r	(4)
GR4_GC0_L5_LE2/um4/ltb (lt03b)	0.40 *	7.24 r	(4)
GR4_GC0_L5_LE2/um5/cascout (mxcascout)	0.05 *	7.28 r	(4)
GR4_GC0_L5_LE2/um6/dcout (c1110)	0.00 *	7.28 r	(4)
data arrival time		7.28	(4)

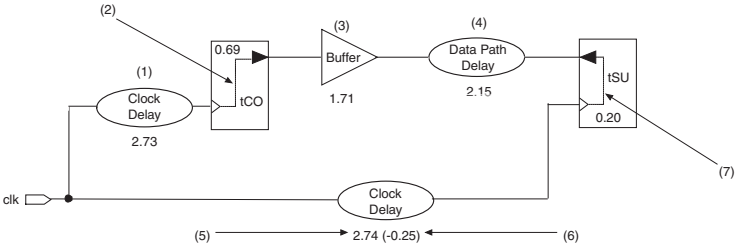
Point	Incr	Path	Reference Point *(1)*
clock clkx' (fall edge)	7.41	7.41	
clock network delay (propagated)	2.74	10.15	(5)
clock uncertainty	-0.25	9.90	(6)
GR4_GC0_L5_LE2/um6/clk (c1110)		9.90 f	
library setup time	-0.20 *	9.70	(7)
data required time		9.70	
data required time		9.70	
data arrival time		-7.28	
slack (MET)		2.42	

Note to Table 4–4:

(1) This column does not exist in the actual report. It is included in this document to provide corresponding reference points to Figure 4–4.

The GR12_GC0_L2_LE4/REGOUT pin now has the loading on it reduced by the introduction of several levels of buffering (in this case, six levels of inverters). The inverters have instance names similar to N1188_iv06_1_0, and are of type iv06, as shown in the static timing analysis report. As a result, the original setup-time violation of –0.63 ns turned into a slack of +2.42 ns, meaning the setup-time violation is fixed. Figure 4–4 illustrates the circuit that the static timing analysis report shows. The buffer tree (buffer) is shown as a single cell.

Figure 4–4. Circuit Post Place-and-Route



Placing the values from the static timing analysis report into the setup-time slack equation, results in the following:

$$t_{SU} \text{ slack} = \text{clock period} + \text{clock delay} - (\text{data delay} + \mu t_{SU})$$

$$t_{SU} \text{ slack} = 7.41 + (2.74 - 0.25) - (2.73 + 0.69 + 1.71 + 2.15 + 0.20)$$

$$t_{SU} \text{ Slack} = +2.42 \text{ ns}$$

This result shows that there is positive slack for this path, meaning that there is now no setup-time violation.

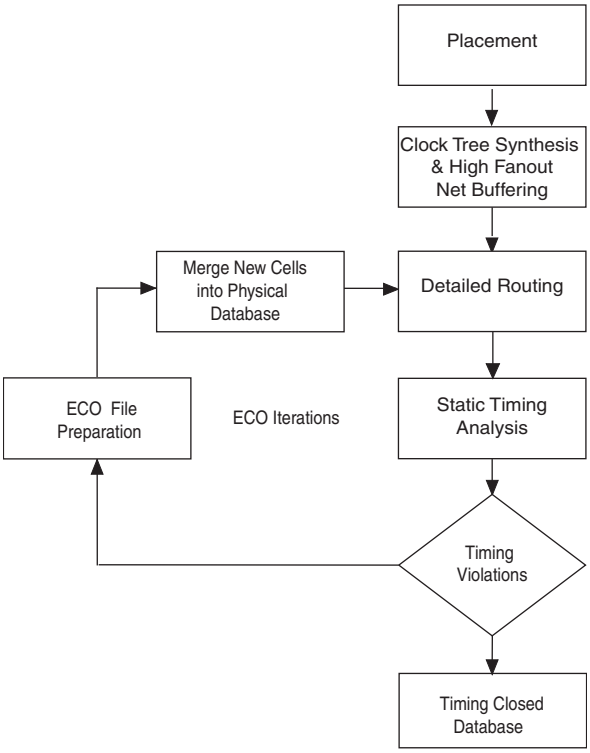
Timing ECOs

In an ASIC, small incremental changes to a design database are termed engineering change orders (ECOs). In the HardCopy series design flow, ECOs are performed after the initial post-layout timing data is available.

You run static timing analysis on the design, which generates a list of paths with timing violations. An automatically updated netlist reflects changes that correct these timing violations (for example, the addition of delay cells to fix hold-time violations). After the netlist update, the updated place-and-route database reflects the netlist changes. The impact to this database is made minimal by maintaining all of the pre-existing placement and routing, and only changing the routing of newly inserted cells.

The parasitic (undesirable, but unavoidable) resistances and capacitances of the customized interconnect are extracted, and are used in conjunction with the static timing analysis tool to re-check the timing of the design. Detected crosstalk violations on signals are fixed by adding additional buffering to increase the setup or hold margin on victim signals. In-line buffering and small buffer tree insertion is done for signals with high fan-out, high transition times, or high capacitive loading. Figure 4–5 shows this flow in more detail.

Figure 4–5. ECO Flow Diagram



The back-end flow in HardCopy produces the final sign-off timing for your HardCopy device. The Quartus II software produces the timing report for HardCopy based on a global route and does not factor in exact physical parasitics of the routed nets, nor does it factor in the crosstalk effect that neighboring nets can have on interconnect capacitance.

Conclusion

It is critical that you fully constrain your HardCopy series design for timing. Although HardCopy series devices are functionally equivalent to their FPGA prototype companion, they have inevitable timing differences. Fully constrained timing paths are a cornerstone of designing for HardCopy series devices.

Consult with Altera if you have questions on what areas to concentrate your efforts in to achieve timing closure within the Quartus fitter for HardCopy design submission.

Document Revision History

Table 4–5 shows the revision history for this chapter.

Table 4–5. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
September 2008, v2.4	Updated chapter number and metadata.	—
June 2007, v2.3	Minor text edits.	—
December 2006 v2.2	• Minor updates for the Quartus II software version 6.1.0 • Moved *Checking the HardCopy Series Device Timing* section to Chapter 7	A minor update to the chapter, due to changes in the Quartus II software version 6.1 release; also, *Checking the HardCopy Series Device Timing* section moved to Chapter 7.
March 2006	Formerly chapter 17; no content change.	—
October 2005 v2.1	• Moved *Chapter 16 Back-End Timing Closure for Hardcopy Series Devices* to Chapter 17 in *HardCopy Series Device Handbook* release 3.2 • Updated graphics • Minor edits	—

Table 4–5. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
January 2005 v2.0	• Chapter title changed to *Back-End Timing Closure for HardCopy Series Devices*. • Sizes of silicon technology updated in "Timing Closure" on page 17–2. • HardCopy® Stratix® and HardCopy APEX™ equivalence to their respective FPGA is updated on page 17–2. • Stratix II migration added. • Updated Table 17–2 on page 17–12. • Updated last paragraph in "Timing ECOs" on page 17–18.	—
June 2003 v1.0	Initial release of Chapter 17, Back-End Timing Closure for HardCopy Series Devices.	—