Stratix GX Device Handbook, Volume 3

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

SGX5V3-1.2

Contents

Section II. Design Guidelines

Chapter 3. Transitioning APEX Designs to Stratix & Stratix GX Devices

Chapter 4. Stratix GX Board Design Guidelines

Chapter Revision Dates

The chapters in this book, *Stratix GX Device Handbook, Volume 3*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Configuring Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52013-3.2*

Chapter 2. Remote System Configuration with Stratix & Stratix GX Devices
Revised: *September 2004*
Part number: *S52015-3.1*

Chapter 3. Transitioning APEX Designs to Stratix & Stratix GX Devices
Revised: *July 2005*
Part number: *S52012-3.0*

Chapter 4. Stratix GX Board Design Guidelines
Revised: *February 2005*
Part number: *SGX53001-1.0*

Chapter 5. Quartus II Software
Fitter Warnings
Revised: *March 2005*
Part number: *SGX53002-1.0*

About This Handbook

This handbook provides comprehensive information about the Altera® Stratix® GX family of devices.

How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	USA & Canada	All Other Locations
Technical support	www.altera.com/mysupport/	www.altera.com/mysupport/
	(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)	+1 408-544-8767 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
Product literature	www.altera.com	www.altera.com
Altera literature services	literature@altera.com	literature@altera.com
Non-technical customer service	(800) 767-3753	+ 1 408-544-7000 7:00 a.m. to 5:00 p.m. (GMT -8:00) Pacific Time
FTP site	ftp.altera.com	ftp.altera.com

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	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
	The warning indicates information that should be read prior to starting or continuing the procedure or processes
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Section I. Configuration & Remote System Upgrades

This section provides information on Stratix® and Stratix GX device configuration and remote system upgrades. It also provides configuration information for the supported configuration schemes in Stratix and Stratix GX devices.

This section includes the following chapters:

- Chapter 1, Configuring Stratix & Stratix GX Devices
- Chapter 2, Remote System Configuration with Stratix & Stratix GX Devices

Revision History

The table below shows the revision history for Chapters 1 and 2.

Chapter(s)	Date / Version	Changes Made
1	July 2005, v3.2	Updated as part of the *Stratix Device Handbook* update.
	September 2004 v3.1	Added chapter to *Stratix GX Device Handbook*.
2	September 2004 v3.1	Added chapter to *Stratix GX Device Handbook*.

1. Configuring Stratix & Stratix GX Devices

S52013-3.2

Introduction

You can configure Stratix® and Stratix GX devices using one of several configuration schemes. All configuration schemes use either a microprocessor, configuration device, or a download cable. See Table 1–1.

Table 1–1. Stratix & Stratix GX Device Configuration Schemes

Configuration Scheme	Typical Use
Fast passive parallel (FPP)	Configuration with a parallel synchronous configuration device or microprocessor interface where eight bits of configuration data are loaded on every clock cycle.
Passive serial (PS)	Configuration with a serial synchronous microprocessor interface or the MasterBlaster™ communications cable, USB Blaster, ByteBlaster™ II, or ByteBlasterMV parallel port download cable.
Passive parallel asynchronous (PPA)	Configuration with a parallel asynchronous microprocessor interface. In this scheme, the microprocessor treats the target device as memory.
Remote/local update FPP	Configuration using a Nios™ (16-bit ISA) and Nios® II (32-bit ISA) or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the FPP scheme to load data.
Remote/local update PS	Passive serial synchronous configuration using a Nios or other embedded processor. Allows you to update the Stratix or Stratix GX device configuration remotely using the PS scheme to load data.
Remote/local update PPA	Passive parallel asynchronous configuration using a Nios or other embedded processor. In this scheme, the Nios microprocessor treats the target device as memory. Allows you to update the Stratix or Stratix GX device configuration remotely using the PPA scheme to load data.
Joint Test Action Group (JTAG)	Configuration through the IEEE Std. 1149.1 JTAG pins. You can perform JTAG configuration with either a download cable or an embedded device. Ability to use SignalTap® II Embedded Logic Analyzer.

This chapter discusses how to configure one or more Stratix or Stratix GX devices. It should be used together with the following documents:

- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *USB Blaster USB Port Download Cable Development Tools Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheets*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet*
- *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet*

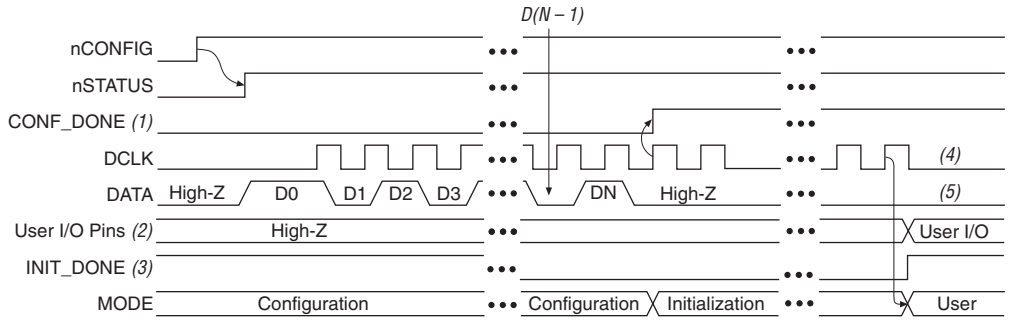
- The *Remote System Configuration with Stratix & Stratix GX Devices* chapter

For more information on setting device configuration options or generating configuration files, see the *Software Setting* chapter in *Volume 2* of the *Configuration Handbook*.

Device Configuration Overview

During device operation, the FPGA stores configuration data in SRAM cells. Because SRAM memory is volatile, you must load the SRAM cells with the configuration data each time the device powers up. After configuration, the device must initialize its registers and I/O pins. After initialization, the device enters user mode. Figure 1–1 shows the state of the device during the configuration, initialization, and user mode.

Figure 1–1. Stratix & Stratix GX Configuration Cycle



Notes to Figure 1–1:

(1) During initial power up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.

(2) User I/O pins are tri-stated during configuration. Stratix and Stratix GX devices also have a weak pull-up resistor on I/O pins during configuration that are enabled by nIO_PULLUP. After initialization, the user I/O pins perform the function assigned in the user's design.

(3) If the INIT_DONE pin is used, it will be high because of an external 10 kΩ resistor pull-up when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low.

(4) DCLK should not be left floating. It should be driven high or low.

(5) DATA0 should not be left floating. It should be driven high or low.

You can load the configuration data for the Stratix or Stratix GX device using a passive configuration scheme. When using any passive configuration scheme, the Stratix or Stratix GX device is incorporated into a system with an intelligent host, such as a microprocessor, that controls the configuration process. The host supplies configuration data from a storage device (e.g., a hard disk, RAM, or other system memory). When using passive configuration, you can change the target device's

functionality while the system is in operation by reconfiguring the device. You can also perform in-field upgrades by distributing a new programming file to system users.

The following sections describe the MSEL[2..0], VCCSEL, PORSEL, and nIO_PULLUP pins used in Stratix and Stratix GX device configuration.

MSEL[2..0] Pins

You can select a Stratix or Stratix GX device configuration scheme by driving its MSEL2, MSEL1, and MSEL0 pins either high or low, as shown in Table 1–2.

Table 1–2. Stratix & Stratix GX Device Configuration Schemes

Description	MSEL2	MSEL1	MSEL0
FPP configuration	0	0	0
PPA configuration	0	0	1
PS configuration	0	1	0
Remote/local update FPP *(1)*	1	0	0
Remote/local update PPA *(1)*	1	0	1
Remote/local update PS *(1)*	1	1	0
JTAG-based configuration *(3)*	*(2)*	*(2)*	*(2)*

Notes to Table 1–2:

(1) These schemes require that you drive a secondary pin RUnLU to specify whether to perform a remote update or local update.

(2) Do not leave MSEL pins floating. Connect them to V_{CCIO} or GND. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used you should connect the MSEL pins to ground.

(3) JTAG-based configuration takes precedence over other configuration schemes, which means the MSEL pins are ignored.

The MSEL[] pins can be tied to V_{CCIO} of the I/O bank they reside in or ground.

V_{CCSEL} Pins

You can configure Stratix and Stratix GX devices using the 3.3-, 2.5-, 1.8-, or 1.5-V LVTTL I/O standard on configuration and JTAG input pins. VCCSEL is a dedicated input on Stratix and Stratix GX devices that selects between 3.3-V/2.5-V input buffers and 1.8-V/1.5-V input buffers for dedicated configuration input pins. A logic low supports 3.3-V/2.5-V signaling, and a logic high supports 1.8-V/1.5-V signaling. A logic high can also support 3.3-V/2.5-V signaling. VCCSEL affects the configuration

related I/O banks (3, 4, 7, and 8) where the following pins reside: TDI, TMS, TCK, TRST, MSEL0, MSEL1, MSEL2, nCONFIG, nCE, DCLK, PLL_ENA, CONF_DONE, nSTATUS. The VCCSEL pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic high level. There is an internal 2.5-kΩ pull-down resistor on VCCSEL. Therefore, if you are using a pull-up resister to pull up this signal, you need to use a 1-kΩ resistor.

VCCSEL also sets the power-on-reset (POR) trip point for all the configuration related I/O banks (3, 4, 7, and 8), ensuring that these I/O banks have powered up to the appropriate voltage levels before configuration begins. Upon power-up, the FPGA does not release nSTATUS until V_{CCINT} and all of the V_{CCIO}s of the configuration I/O banks are above their POR trip points. If you set VCCSEL to ground (logic low), this sets the POR trip point for all configuration I/O banks to a voltage consistent with 3.3-V/2.5-V signaling. When VCCSEL = 0, the POR trip point for these I/O banks may be as high as 1.8 V. If V_{CCIO} of any of the configuration banks is set to 1.8 or 1.5 V, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which will not allow the FPGA to begin configuration.

If the V_{CCIO} of I/O banks 3, 4, 7, or 8 is set to 1.5 or 1.8 V and the configuration signals used require 3.3-V or 2.5-V signaling you should set VCCSEL to V_{CC} (logic high) in order to lower the POR trip point to enable successful configuration.

Table 1–3 shows how you should set the VCCSEL depending on the V_{CCIO} setting of the configuration I/O banks and your configuration input signaling voltages.

Table 1–3. VCCSEL Setting

V_{CCIO} (banks 3,4,7,8)	Configuration Input Signaling Voltage	V_{CCSEL}
3.3-V/2.5-V	3.3-V/2.5-V	GND
1.8-V/1.5-V	3.3-V/2.5-V/1.8-V/1.5-V	VCC
3.3-V/2.5-V	1.8-V/1.5-V	Not Supported

The VCCSEL signal does not control any of the dual-purpose pins, including the dual-purpose configuration pins, such as the DATA[7..0] and PPA pins (nWS, nRS, CS, nCS, and RDYnBSY). During configuration, these dual-purpose pins drive out voltage levels corresponding to the V_{CCIO} supply voltage that powers the I/O bank containing the pin. After configuration, the dual-purpose pins inherit the I/O standards specified in the design.

PORSEL Pins

PORSEL is a dedicated input pin used to select POR delay times of 2 ms or 100 ms during power-up. When the PORSEL pin is connected to ground, the POR time is 100 ms; when the PORSEL pin is connected to VCC, the POR time is 2 ms. There is an internal 2.5-kΩ pull-down resistor on PORSEL. Therefore if you are using a pull-up resistor to pull up this signal, you need to use a 1-kΩ resistor.

When using enhanced configuration devices to configure Stratix devices, make sure that the PORSEL setting of the Stratix device is the same or faster than the PORSEL setting of the enhanced configuration device. If the FPGA is not powered up after the enhanced configuration device exits POR, the CONF_DONE signal will be high since the pull-up resistor is pulling this signal high. When the enhanced configuration device exits POR, OE of the enhanced configuration device is released and pulled high by a pull-up resistor. Since the enhanced configuration device sees its nCS/CONF_DONE signal also high, it enters a test mode. Therefore, you must ensure the FPGA powers up before the enhanced configuration device exits POR.

For more margin, the 100-ms setting can be selected when using an enhanced configuration device to allow the Stratix FPGA to power-up before configuration is attempted (see Table 1–4).

Table 1–4. PORSEL Settings

PORSEL Settings	POR Time (ms)
GND	100
V_{CC}	2

nIO_PULLUP Pins

The nIO_PULLUP pin enables a built-in weak pull-up resistor to pull all user I/O pins to VCCIO before and during device configuration. If nIO_PULLUP is connected to V_{CC} during configuration, the weak pull-ups on all user I/O pins and all dual-purpose pins are disabled. If connected to ground, the pull-ups are enabled during configuration. The nIO_PULLUP pin can be pulled to 1.5, 1.8, 2.5, or 3.3-V for a logic level high. There is an internal 2.5-kΩ pull-down resistor on nIO_PULLUP. Therefore, if you are using a pull-up resistor to pull up this signal, you need to use a 1-kΩ resistor.

TDO & nCEO Pins

TDO and nCEO pins drive out the same voltage levels as the V_{CCIO} that powers the I/O bank where the pin resides. You must select the V_{CCIO} supply for the bank containing TDO accordingly. For example, when using the ByteBlasterMV cable, the V_{CCIO} for the bank containing TDO must be powered up at 3.3-V. The current strength for TDO is 12 mA.

Configuration File Size

Tables 1–5 and 1–6 summarize the approximate configuration file size required for each Stratix and Stratix GX device. To calculate the amount of storage space required for multi-device configurations, add the file size of each device together.

Table 1–5. Stratix Configuration File Sizes

Device	Raw Binary File (.rbf) Size (Bits)
EP1S10	3,534,640
EP1S20	5,904,832
EP1S25	7,894,144
EP1S30	10,379,368
EP1S40	12,389,632
EP1S60	17,543,968
EP1S80	23,834,032

Table 1–6. Stratix GX Configuration File Sizes

Device	Raw Binary File Size (Bits)
EP1SGX10C	3,579,928
EP1SGX10D	3,579,928
EP1SGX25C	7,951,248
EP1SGX25D	7,951,248
EP1SGX25F	7,951,248
EP1SGX40D	12,531,440
EP1SGX40G	12,531,440

You should only use the numbers in Tables 1–5 and 1–6 to estimate the file size before design compilation. The exact file size may vary because different Altera® Quartus® II software versions may add a slightly

different number of padding bits during programming. However, for any specific version of the Quartus II software, any design targeted for the same device has the same configuration file size.

Altera Configuration Devices

The Altera enhanced configuration devices (EPC16, EPC8, and EPC4 devices) support a single-device configuration solution for high-density FPGAs and can be used in the FPP and PS configuration schemes. They are ISP-capable through its JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.

For information on enhanced configuration devices, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Using Altera Enhanced Configuration Devices* chapter in the *Configuration Handbook*.

The EPC2 and EPC1 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 and EPC1 can be cascaded to hold large configuration files.

For more information on EPC2, EPC1, and EPC1441 configuration devices, see the *Configuration Devices for SRAM-Based LUT Devices Data Sheet*.

Configuration Schemes

This section describes how to configure Stratix and Stratix GX devices with the following configuration schemes:

- PS Configuration with Configuration Devices
- PS Configuration with a Download Cable
- PS Configuration with a Microprocessor
- FPP Configuration
- PPA Configuration
- JTAG Programming & Configuration
- JTAG Programming & Configuration of Multiple Devices

PS Configuration

PS configuration of Stratix and Stratix GX devices can be performed using an intelligent host, such as a MAX® device, microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Stratix devices via the `DATA0` pin at each rising edge of `DCLK`.

PS Configuration with Configuration Devices

The configuration device scheme uses an Altera configuration device to supply data to the Stratix or Stratix GX device in a serial bitstream (see Figure 1–3).

In the configuration device scheme, nCONFIG is usually tied to V_{CC} (when using EPC16, EPC8, EPC4, or EPC2 devices, nCONFIG may be connected to nINIT_CONF). Upon device power-up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target device then drives the open-drain CONF_DONE pin low, which in-turn drives the configuration device's nCS pin low. When exiting power-on reset (POR), both the target and configuration device release the open-drain nSTATUS pin.

Before configuration begins, the configuration device goes through a POR delay of up to 200 ms to allow the power supply to stabilize (power the Stratix or Stratix GX device before or during the POR time of the configuration device). This POR delay has a maximum of 200 ms for EPC2 devices. For enhanced configuration devices, you can select between 2 ms and 100 ms by connecting PORSEL pin to VCC or GND, accordingly. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin. When the target and configuration devices complete POR, they release nSTATUS, which is then pulled high by a pull-up resistor.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. When all devices are ready, the configuration device clocks data out serially to the target devices using an internal oscillator.

After successful configuration, the Stratix FPGA starts initialization using the 10-MHz internal oscillator as the reference clock. After initialization, this internal oscillator is turned off. The CONF_DONE pin is released by the target device and then pulled high by a pull-up resistor. When initialization is complete, the FPGA enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

If an error occurs during configuration, the target device drives its nSTATUS pin low, resetting itself internally and resetting the configuration device. If the **Auto-Restart Configuration on Frame Error** option—available in the Quartus II **Global Device Options** dialog box (Assign menu)—is turned on, the device reconfigures automatically if an error occurs. To find this option, choose **Compiler Settings** (Processing menu), then click on the **Chips & Devices** tab.

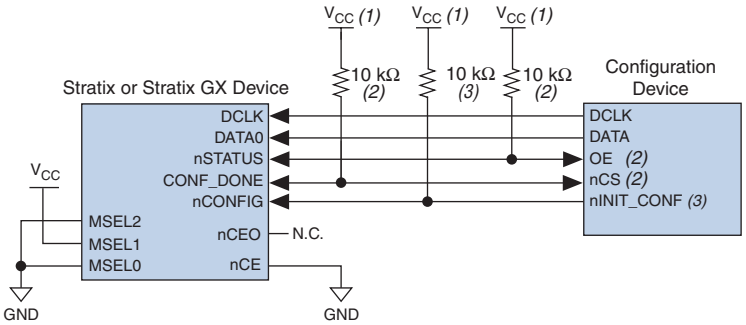
If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low to restart configuration. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. When configuration is complete, the target device releases CONF_DONE, which disables the configuration device by driving nCS high. The configuration device drives DCLK low before and after configuration.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the target device has not configured successfully. In this case, the configuration device pulses its OE pin low for a few microseconds, driving the target device's nSTATUS pin low. If the **Auto-Restart Configuration on Frame Error** option is set in the software, the target device resets and then pulses its nSTATUS pin low. When nSTATUS returns high, the configuration device reconfigures the target device. When configuration is complete, the configuration device drives DCLK low.

Do not pull CONF_DONE low to delay initialization. Instead, use the Quartus II software's **Enable User-Supplied Start-Up Clock (CLKUSR)** option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together. When CONF_DONE is driven low after device configuration, the configuration device recognizes that the target device has not configured successfully.

Figure 1–2 shows how to configure one Stratix or Stratix GX device with one configuration device.

Figure 1–2. Single Device Configuration Circuit



Notes to Figure 1–2:

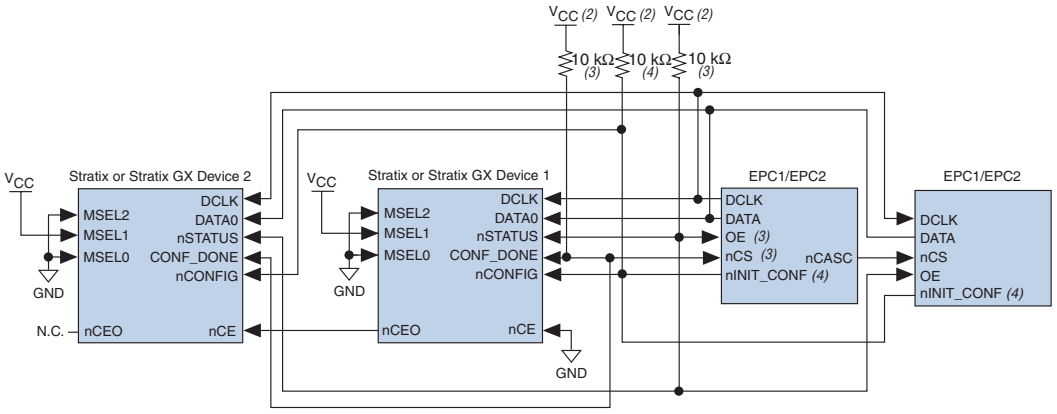
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. he nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

Figure 1–3 shows how to configure multiple Stratix and Stratix GX devices with multiple EPC2 or EPC1 configuration devices.

Figure 1–3. Multi-Device Configuration Circuit Note (1)



Notes to Figure 1–3:

(1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof**) from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

(2) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(4) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second device's nCE pin, prompting the second device to begin configuration. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

In addition, all nSTATUS pins are tied together; thus, if any device (including the configuration devices) detects an error, configuration stops for the entire chain. Also, if the first configuration device does not detect CONF_DONE going high at the end of configuration, it resets the chain by pulsing its OE pin low for a few microseconds. This low pulse drives the OE pin low on the second configuration device and drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state.

If the **Auto-Restart Configuration on Frame Error** option is turned on in the software, the Stratix or Stratix GX device releases its nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration devices reconfigure the chain. If the **Auto-**

Restart Configuration on Frame Error option is not turned on, the Stratix or Stratix GX devices drive nSTATUS low until they are reset with a low pulse on nCONFIG.

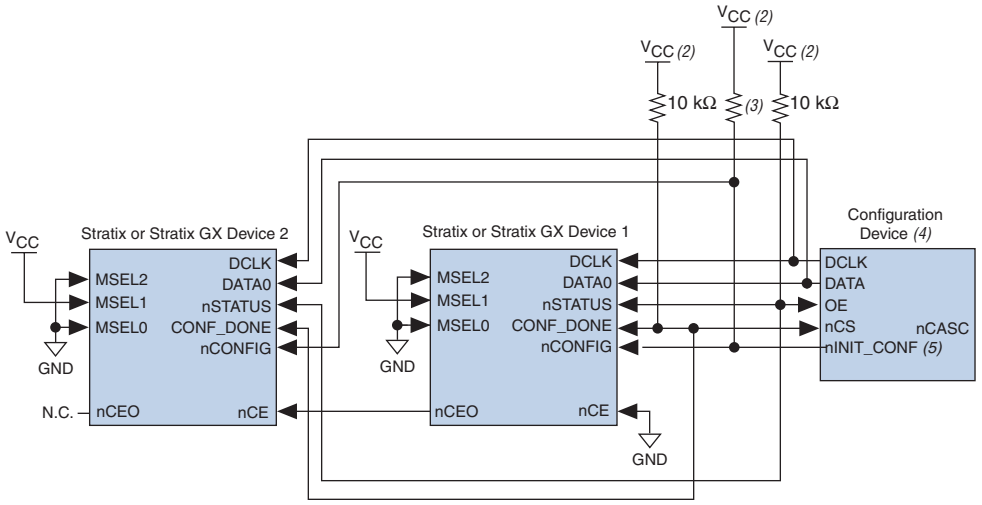
You can also cascade several EPC2/EPC1 configuration devices to configure multiple Stratix and Stratix GX devices. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the subsequent configuration device. Because a configuration device requires less than one clock cycle to activate a subsequent configuration device, the data stream is uninterrupted.

You cannot cascade enhanced (EPC16, EPC8, and EPC4) configuration devices.

You can use a single configuration chain to configure multiple Stratix and Stratix GX devices. In this scheme, the nCEO pin of the first device is connected to the nCE pin of the second device in the chain. If there are additional devices, connect the nCE pin of the next device to the nCEO pin of the previous device. To configure properly, all of the device CONF_DONE and nSTATUS pins must be tied together.

Figure 1–4 shows an example of configuring multiple Stratix and Stratix GX devices using a configuration device.

Figure 1–4. Configuring Multiple Stratix & Stratix GX Devices with A Single Configuration Device Note (1)



Notes to Figure 1–4:

(1) When performing multi-device active serial configuration, you must generate the configuration device programmer object file (**.pof)** from each project's SOF. You can combine multiple SOFs using the Quartus II software through the **Device & Pin Option** dialog box. For more information on how to create configuration and programming files, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

(2) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(3) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on `OE` and `nCS`. You should only use the internal pull-ups of the configuration device if the `nSTATUS` and `CONF_DONE` signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(4) EPC16, EPC8, and EPC4 configuration devices cannot be cascaded.

(5) The `nINIT_CONF` pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to V_{CC} through a resistor. The `nINIT_CONF` pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the `nINIT_CONF` pin.

Table 1–7 shows the status of the device DATA pins during and after configuration.

Table 1–7. DATA Pin Status Before & After Configuration

Pins	Stratix or Stratix GX Device	
	During	After
DATA0 *(1)*	Used for configuration	User defined
DATA[7..1] *(2)*	Used in some configuration modes	User defined
I/O Pins	Tri-state	User defined

Notes to Table 1–7:

(1) The status shown is for configuration with a configuration device.

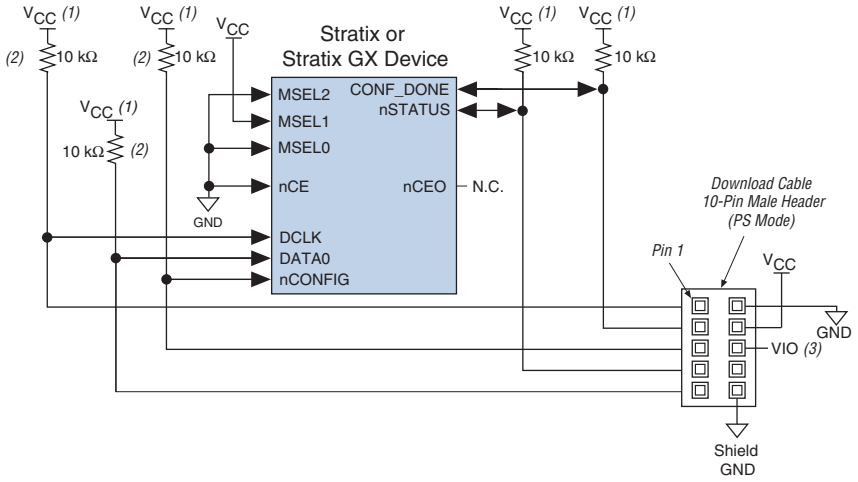
(2) The function of these pins depends upon the settings specified in the Quartus II software using the **Device & Pin Option** dialog box (see the *Software Settings* section in the *Configuration Handbook, Volume 2*, and the Quartus II Help software for more information).

PS Configuration with a Download Cable

In PS configuration with a download cable, an intelligent host transfers data from a storage device to the Stratix or Stratix GX device through the MasterBlaster, USB-Blaster, ByteBlaster II or ByteBlasterMV cable. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin. The programming hardware then places the configuration data one bit at a time on the device's DATA0 pin. The data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using programming hardware for the Stratix or Stratix GX device, turning on the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle because the Quartus II software must restart configuration when an error occurs. Additionally, the **Enable User-Supplied Start-Up Clock (CLKUSR)** option has no affect on the device initialization since this option is disabled in the SOF when programming the FPGA using the Quartus II software programmer and a download cable. Therefore, if you turn on the **CLKUSR** option, you do not need to provide a clock on CLKUSR when you are configuring the FPGA with the Quartus II programmer and a download cable. Figure 1–5 shows PS configuration for the Stratix or Stratix GX device using a MasterBlaster, USB-Blaster, ByteBLaster II or ByteBlasterMV cable.

Figure 1–5. PS Configuration Circuit with a Download Cable



Notes to Figure 1–5:

(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

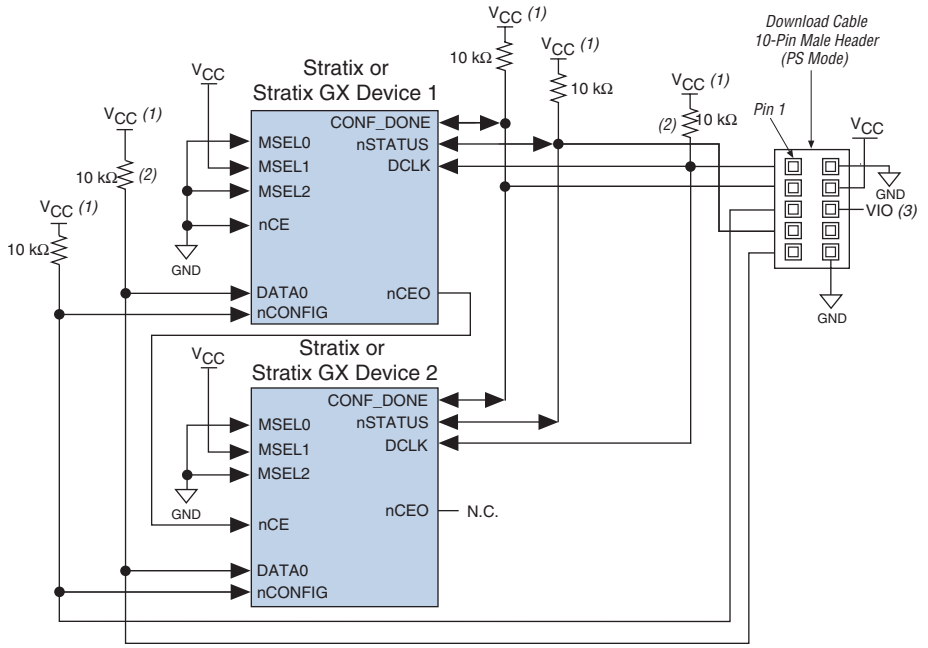
(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. This pin is a no-connect pin for the ByteBlasterMV header.

You can use programming hardware to configure multiple Stratix and Stratix GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. All other configuration pins are connected to each device in the chain.

Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time. In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. In this situation, the Quartus II software must restart configuration; the **Auto-Restart Configuration on Frame Error** option does not affect the configuration cycle.

Figure 1–6 shows how to configure multiple Stratix and Stratix GX devices with a MasterBlaster or ByteBlasterMV cable.

Figure 1–6. Multi-Device PS Configuration with a Download Cable



Notes to Figure 1–6:

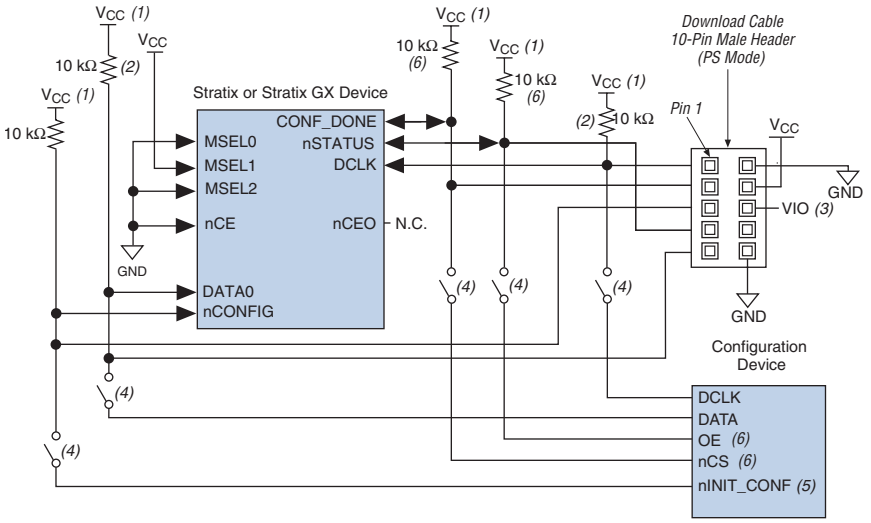
(1) You should connect the pull-up resistor to the same supply voltage as the MasterBlaster (VIO pin) or ByteBlasterMV cable.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

If you are using a download cable to configure device(s) on a board that also has configuration devices, you should electrically isolate the configuration devices from the target device(s) and cable. One way to isolate the configuration devices is to add logic, such as a multiplexer, that can select between the configuration devices and the cable. The multiplexer device should allow bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (CONF_DONE, nSTATUS, DCLK, nCONFIG, and DATA0) between the cable and the configuration devices. The last option is to remove the configuration devices from the board when configuring with the cable. Figure 1–7 shows a combination of a configuration device and a download cable to configure a Stratix or Stratix GX device.

Figure 1–7. Configuring with a Combined PS & Configuration Device Scheme



Notes to Figure 1–7:

(1) You should connect the pull-up resistor to the same supply voltage as the configuration device.

(2) The pull-up resistors on the DATA0 and DCLK pins are only needed if the download cable is the only configuration scheme used on the board. This is to ensure that the DATA0 and DCLK pins are not left floating after configuration. For example, if the design also uses a configuration device, the pull-up resistors on the DATA0 and DCLK pins are not necessary.

(3) Pin 6 of the header is a V_{IO} reference voltage for the MasterBlaster output driver. V_{IO} should match the target device's V_{CCIO}. This is a no-connect pin for the ByteBlasterMV header.

(4) You should not attempt configuration with a download cable while a configuration device is connected to a Stratix or Stratix GX device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device. Remove the download cable when configuring with a configuration device.

(5) If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} either directly or through a resistor.

(6) If external pull-ups are used on CONF_DONE and nSTATUS pins, they should always be 10 kΩ resistors. You can use the internal pull-ups of the configuration device only if the CONF_DONE and nSTATUS signals are pulled-up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V).

For more information on how to use the MasterBlaster or ByteBlasterMV cables, see the following documents:

- *USB-Blaster USB Port Download Cable Data Sheet*
- *MasterBlaster Serial/USB Communications Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*
- *ByteBlaster II Parallel Port Download Cable Data Sheet*

PS Configuration with a Microprocessor

In PS configuration with a microprocessor, a microprocessor transfers data from a storage device to the target Stratix or Stratix GX device. To initiate configuration in this scheme, the microprocessor must generate a low-to-high transition on the nCONFIG pin and the target device must release nSTATUS. The microprocessor or programming hardware then places the configuration data one bit at a time on the DATA0 pin of the Stratix or Stratix GX device. The least significant bit (LSB) of each data byte must be presented first. Data is clocked continuously into the target device until CONF_DONE goes high.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the **clkusr** option, after all data is transferred clkusr must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation.

Handshaking signals are not used in PS configuration modes. Therefore, the configuration clock speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

If the target device detects an error during configuration, it drives its nSTATUS pin low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on in the Quartus II software, the target device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the target device without needing to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. If the microprocessor sends all data and the initialization clock starts but CONF_DONE and INIT_DONE have not gone high, it must reconfigure the target device. By default the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on **Enable INIT_DONE output** option in the Quartus II software.

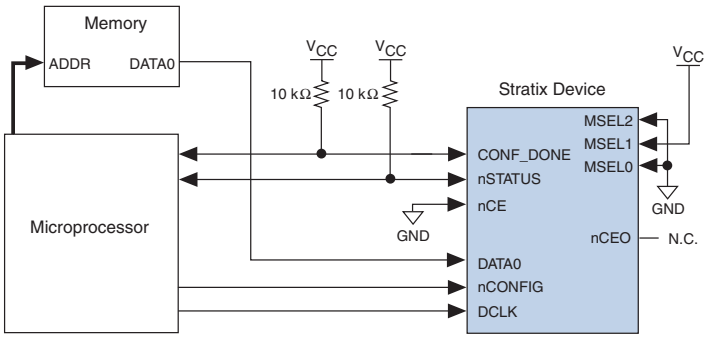
If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see Table 1–8) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.

If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

Figure 1–8 shows the circuit for PS configuration with a microprocessor.

Figure 1–8. PS Configuration Circuit with Microprocessor



PS Configuration Timing

Figure 1–9 shows the PS configuration timing waveform for Stratix and Stratix GX devices. Table 1–8 shows the PS timing parameters for Stratix and Stratix GX devices.

Table 1–8. PS Timing Parameters for Stratix & Stratix GX Devices

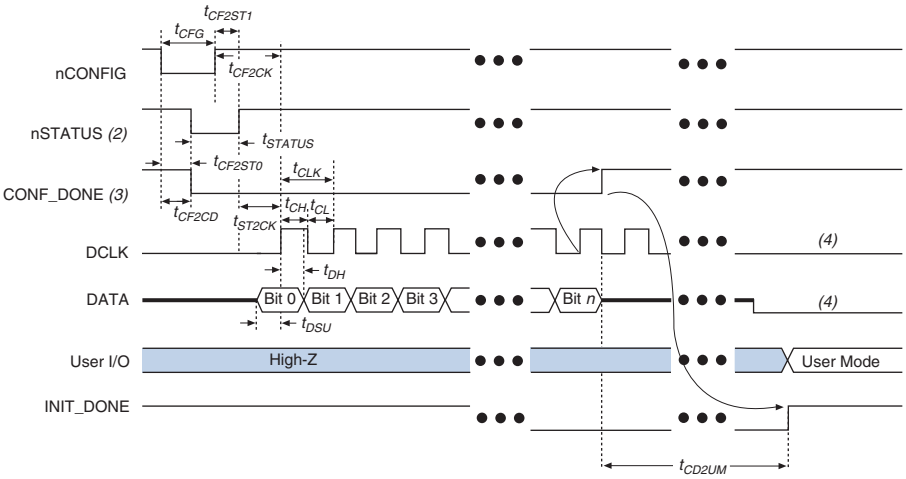
Symbol	Parameter	Min	Max	Units
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{CFG}	nCONFIG low pulse width	40		µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{ST2CK}	nSTATUS high to first rising edge on DCLK	1		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns
f_{MAX}	DCLK maximum frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs

Notes to Table 1–8:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

Figure 1–9. PS Timing Waveform for Stratix & Stratix GX Devices Note (1)



Notes to Figure 1–9:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA[] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

FPP Configuration

Parallel configuration of Stratix and Stratix GX devices meets the continuously increasing demand for faster configuration times. Stratix and Stratix GX devices can receive byte-wide configuration data per clock cycle, and guarantee a configuration time of less than 100 ms with a 100-MHz configuration clock. Stratix and Stratix GX devices support programming data bandwidth up to 800 megabits per second (Mbps) in this mode. You can use parallel configuration with an EPC16, EPC8, or EPC4 device, or a microprocessor.

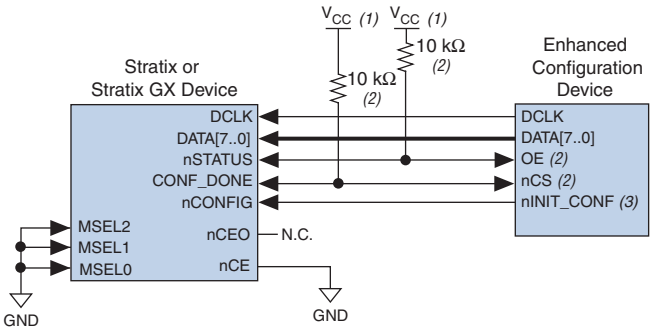
This section discusses the following schemes for FPP configuration in Stratix and Stratix GX devices:

- FPP Configuration Using an Enhanced Configuration Device
- FPP Configuration Using a Microprocessor

FPP Configuration Using an Enhanced Configuration Device

When using FPP with an enhanced configuration device, it supplies data in a byte-wide fashion to the Stratix or Stratix GX device every DCLK cycle. See Figure 1–10.

Figure 1–10. FPP Configuration Using Enhanced Configuration Devices



Notes to Figure 1–10:

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

(2) The enhanced configuration devices and EPC2 devices have internal programmable pull-ups on OE and nCS. You should only use the internal pull-ups of the configuration device if the nSTATUS and CONF_DONE signals are pulled up to 3.3 V or 2.5 V (not 1.8 V or 1.5 V). If external pull-ups are used, they should be 10 kΩ

(3) The nINIT_CONF pin is available on EPC16, EPC8, EPC4, and EPC2 devices. If nINIT_CONF is not used, nCONFIG must be pulled to V_{CC} through a resistor. The nINIT_CONF pin has an internal pull-up resistor that is always active in EPC16, EPC8, EPC4, and EPC2 devices. These devices do not need an external pull-up resistor on the nINIT_CONF pin.

In the enhanced configuration device scheme, nCONFIG is tied to nINIT_CONF. On power up, the target Stratix or Stratix GX device senses the low-to-high transition on nCONFIG and initiates configuration. The target Stratix or Stratix GX device then drives the open-drain CONF_DONE pin low, which in-turn drives the enhanced configuration device's nCS pin low.

Before configuration starts, there is a 2-ms POR delay if the PORSEL pin is connected to V_{CC} in the enhanced configuration device. If the PORSEL pin is connected to ground, the POR delay is 100 ms. When each device determines that its power is stable, it releases its nSTATUS or OE pin. Because the enhanced configuration device's OE pin is connected to the target Stratix or Stratix GX device's nSTATUS pin, configuration is delayed until both the nSTATUS and OE pins are released by each device. The nSTATUS and OE pins are pulled up by a resistor on their respective

devices once they are released. When configuring multiple devices, connect the nSTATUS pins together to ensure configuration only happens when all devices release their OE or nSTATUS pins. The enhanced configuration device then clocks data out in parallel to the Stratix or Stratix GX device using a 66-MHz internal oscillator, or drives it to the Stratix or Stratix GX device through the EXTCLK pin.

If there is an error during configuration, the Stratix or Stratix GX device drives the nSTATUS pin low, resetting itself internally and resetting the enhanced configuration device. The Quartus II software provides an **Auto-restart configuration after error** option that automatically initiates the reconfiguration whenever an error occurs. See the *Software Settings* chapter in Volume 2 of the *Configuration Handbook* for information on how to turn this option on or off.

If this option is turned off, you must set monitor nSTATUS to check for errors. To initiate reconfiguration, pulse nCONFIG low. The external system can pulse nCONFIG if it is under system control rather than tied to V_{CC}. Therefore, nCONFIG must be connected to nINIT_CONF if you want to reprogram the Stratix or Stratix GX device on the fly.

When configuration is complete, the Stratix or Stratix GX device releases the CONF_DONE pin, which is then pulled up by a resistor. This action disables the EPC16, EPC8, or EPC4 enhanced configuration device as nCS is driven high. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode. The enhanced configuration device drives DCLK low before and after configuration.

CONF_DONE goes high one byte early in parallel synchronous (FPP) and asynchronous (PPA) modes using a microprocessor with **.rbf**, **.hex**, and **.ttf** file formats. This does not apply to FPP mode for enhanced configuration devices using **.pof** file format. This also does not apply to serial modes.

If, after sending out all of its data, the enhanced configuration device does not detect CONF_DONE going high, it recognizes that the Stratix or Stratix GX device has not configured successfully. The enhanced configuration device pulses its OE pin low for a few microseconds, driving the nSTATUS pin on the Stratix or Stratix GX device low. If the **Auto-restart configuration after error** option is on, the Stratix or Stratix GX device resets and then pulses its nSTATUS low. When nSTATUS returns high, reconfiguration is restarted (see Figure 1–11 on page 1–25).

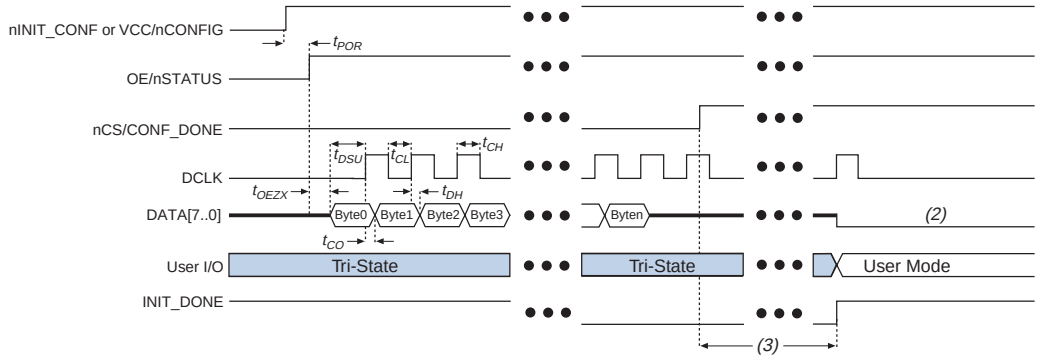
Do not drive CONF_DONE low after device configuration to delay initialization. Instead, use the **Enable User-Supplied Start-Up Clock (CLKUSR)** option in the **Device & Pin Options** dialog box. You can use this option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain initialize together.

After the first Stratix or Stratix GX device completes configuration during multi-device configuration, its nCEO pin activates the second Stratix or Stratix GX device's nCE pin, prompting the second device to begin configuration. Because CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time. Because nSTATUS pins are tied together, configuration stops for the whole chain if any device (including enhanced configuration devices) detects an error. Also, if the enhanced configuration device does not detect a high on CONF_DONE at the end of configuration, it pulses its OE low for a few microseconds to reset the chain. The low OE pulse drives nSTATUS low on all Stratix and Stratix GX devices, causing them to enter an error state. This state is similar to a Stratix or Stratix GX device detecting an error.

If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release their nSTATUS pins after a reset time-out period. When the nSTATUS pins are released and pulled high, the configuration device reconfigures the chain. If the **Auto-restart configuration after error** option is off, nSTATUS stays low until the Stratix and Stratix GX devices are reset with a low pulse on nCONFIG.

Figure 1–11 shows the FPP configuration with a configuration device timing waveform for Stratix and Stratix GX devices.

Figure 1–11. FPP Configuration with a Configuration Device Timing Waveform Note (1)



Notes to Figure 1–11:

(1) For timing information, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet*.
(2) The configuration device drives DATA high after configuration.
(3) Stratix and Stratix GX devices enter user mode 136 clock cycles after CONF_DONE goes high.

FPP Configuration Using a Microprocessor

When using a microprocessor for parallel configuration, the microprocessor transfers data from a storage device to the Stratix or Stratix GX device through configuration hardware. To initiate configuration, the microprocessor needs to generate a low-to-high transition on the nCONFIG pin and the Stratix or Stratix GX device must release nSTATUS. The microprocessor then places the configuration data to the DATA[7..0] pins of the Stratix or Stratix GX device. Data is clocked continuously into the Stratix or Stratix GX device until CONF_DONE goes high.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists. You can pause configuration by halting DCLK for an indefinite amount of time.

After all configuration data is sent to the Stratix or Stratix GX device, the CONF_DONE pin goes high to show successful configuration and the start of initialization. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. If you are using the **clkusr** option, after all data is transferred clkusr must be clocked an additional 136 times for the Stratix or Stratix GX device to initialize properly. Driving DCLK to the device after configuration is complete does not affect device operation. By

default, the INIT_DONE output is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software.

If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for maximum value of t_{CD2UM} (see Table 1–9) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.

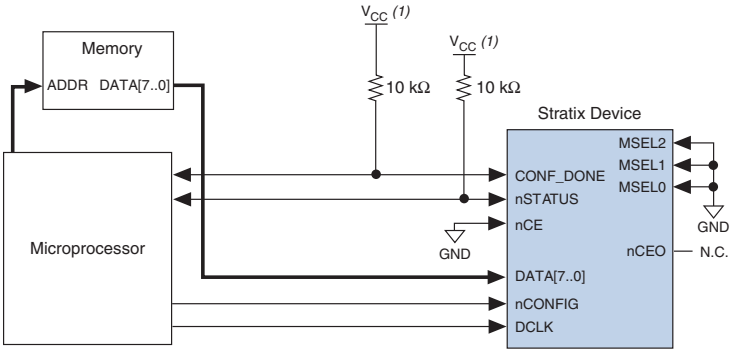
If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

If the Stratix or Stratix GX device detects an error during configuration, it drives nSTATUS low to alert the microprocessor. The pin on the microprocessor connected to nSTATUS must be an input. The microprocessor can then pulse nCONFIG low to restart the configuration error. With the **Auto-restart configuration after error** option on, the Stratix or Stratix GX device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the Stratix or Stratix GX device without pulsing nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. If the microprocessor sends all the data and the initialization clock starts but CONF_DONE and INIT_DONE have not gone high, it must reconfigure the Stratix or Stratix GX device. After waiting the specified 136 DCLK cycles, the microprocessor should restart configuration by pulsing nCONFIG low.

Figure 1–12 shows the circuit for Stratix and Stratix GX parallel configuration using a microprocessor.

Figure 1–12. Parallel Configuration Using a Microprocessor

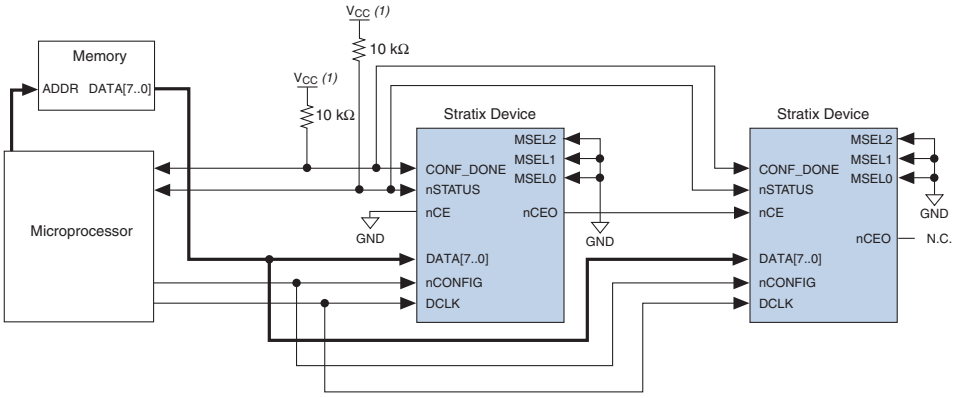


Note to Figure 1–12:
(1) The pull-up resistors should be connected to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

For multi-device parallel configuration with a microprocessor, the nCEO pin of the first Stratix or Stratix GX device is cascaded to the second device's nCE pin. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the microprocessor. Because the CONF_DONE pins of the devices are connected together, all devices initialize and enter user mode at the same time.

Because the nSTATUS pins are also tied together, if any of the devices detects an error, the entire chain halts configuration and drives nSTATUS low. The microprocessor can then pulse nCONFIG low to restart configuration. If the **Auto-restart configuration after error** option is on, the Stratix and Stratix GX devices release nSTATUS after a reset time-out period. The microprocessor can then reconfigure the devices once nSTATUS is released. Figure 1–13 shows multi-device configuration using a microprocessor. Figure 1–14 shows multi-device configuration when both Stratix and Stratix GX devices are receiving the same data. In this case, the microprocessor sends the data to both devices simultaneously, and the devices configure simultaneously.

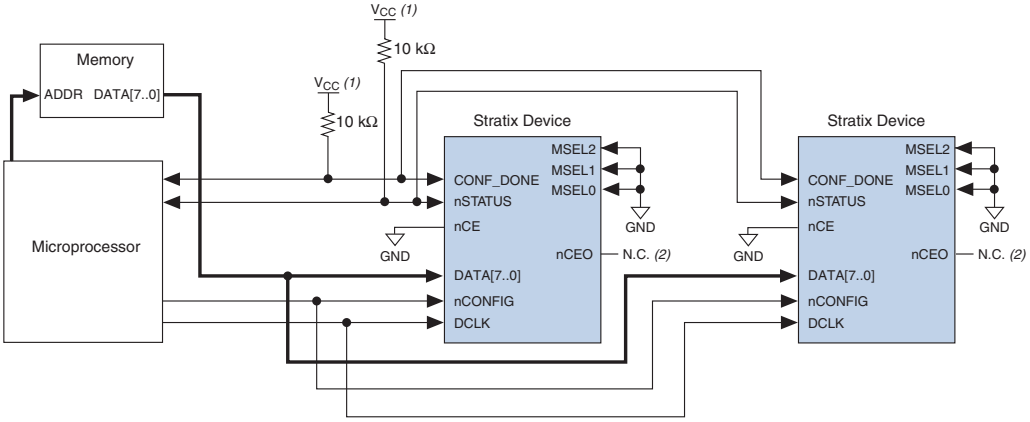
Figure 1–13. Parallel Data Transfer in Serial Configuration with a Microprocessor



Note to Figure 1–13:

(1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

Figure 1–14. Multiple Device Parallel Configuration with the Same Data Using a Microprocessor



Notes to Figure 1–14:

(1) You should connect the pull-up resistors to any V_{CC} that meets the Stratix high-level input voltage (V_{IH}) specification.

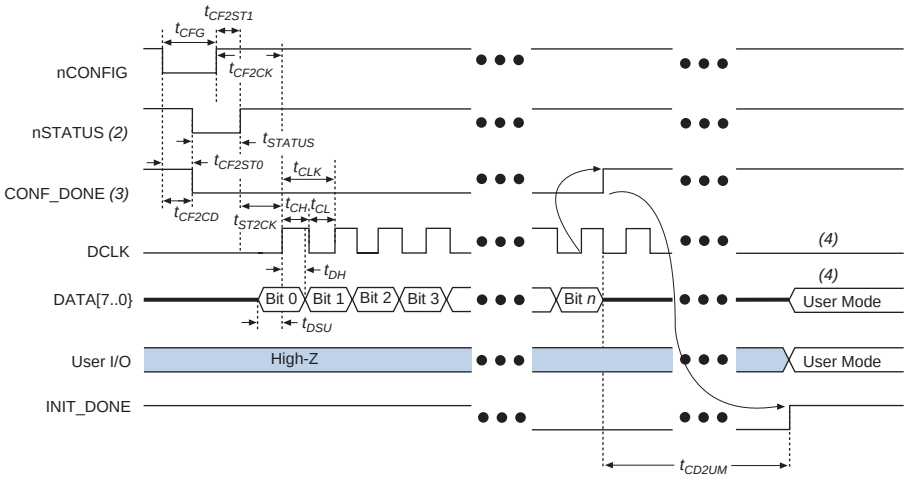
(2) The nCEO pins are left unconnected when configuring the same data into multiple Stratix or Stratix GX devices.

For more information on configuring multiple Altera devices in the same configuration chain, see the *Configuring Mixed Altera FPGA Chains* chapter in the *Configuration Handbook, Volume 2*.

FPP Configuration Timing

Figure 1–15 shows FPP timing waveforms for configuring a Stratix or Stratix GX device in FPP mode. Table 1–9 shows the FPP timing parameters for Stratix or Stratix GX devices.

Figure 1–15. Timing Waveform for Configuring Devices in FPP Mode *Note (1)*



Notes to Figure 1–15:

(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Stratix II device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is convenient. DATA[] is available as user I/Os after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 1–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 1 of 2)

Symbol	Parameter	Min	Max	Units
t_{CF2CK}	nCONFIG high to first rising edge on DCLK	40		µs
t_{DSU}	Data setup time before rising edge on DCLK	7		ns
t_{DH}	Data hold time after rising edge on DCLK	0		ns
t_{CFG}	nCONFIG low pulse width	40		µs
t_{CH}	DCLK high time	4		ns
t_{CL}	DCLK low time	4		ns
t_{CLK}	DCLK period	10		ns

Table 1–9. FPP Timing Parameters for Stratix & Stratix GX Devices (Part 2 of 2)

Symbol	Parameter	Min	Max	Units
f_{MAX}	DCLK frequency		100	MHz
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{ST2CK}	nSTATUS high to firstrising edge of DCLK	1		µs

Notes to Table 1–9:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtainable if users do not delay configuration by extending the nSTATUS low pulse width.

PPA Configuration

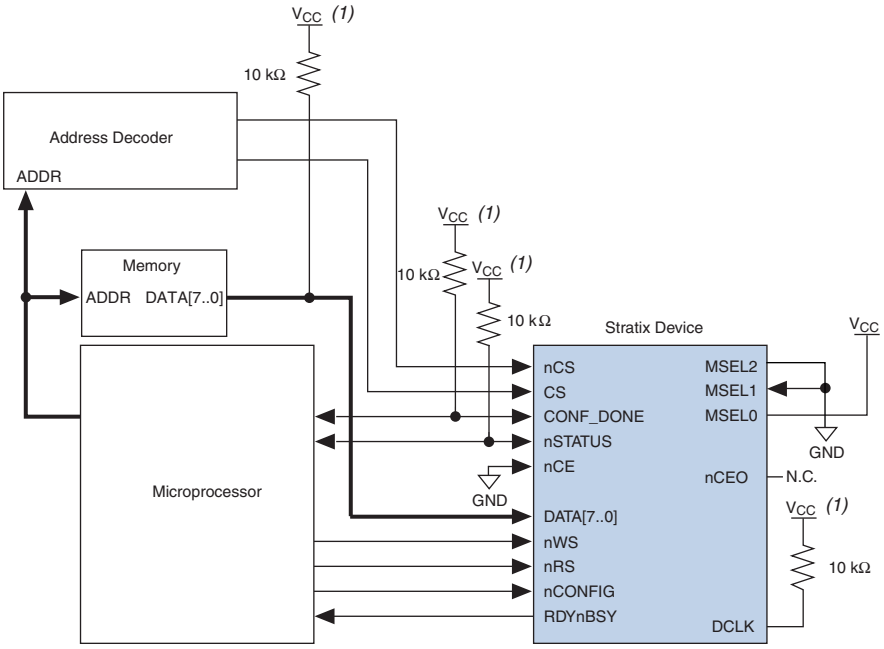
In PPA schemes, a microprocessor drives data to the Stratix or Stratix GX device through a download cable. When using a PPA scheme, use a 1-kΩ pull-up resistor to pull the DCLK pin high to prevent unused configuration pins from floating.

To begin configuration, the microprocessor drives nCONFIG high and then asserts the target device's nCS pin low and CS pin high. Next, the microprocessor places an 8-bit configuration word on the target device's data inputs and pulses nWS low. On the rising edge of nWS, the target device latches a byte of configuration data and then drives its RDYnBSY signal low, indicating that it is processing the byte of configuration data. The microprocessor then performs other system functions while the Stratix or Stratix GX device is processing the byte of configuration data.

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is high and CONF_DONE is low, the microprocessor sends the next data byte. If nSTATUS is low, the device is signaling an error and the microprocessor should restart configuration. However, if nSTATUS is high and all the configuration data is received, the device is ready for initialization. At the beginning of initialization, CONF_DONE goes high to indicate that configuration is complete. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Initialization, by default, uses an internal oscillator, which runs at 10 MHz. After initialization, this internal oscillator is turned off. When initialization is complete, the Stratix or Stratix GX device enters user mode.

Figure 1–16 shows the PPA configuration circuit. An optional address decoder controls the device's nCS and CS pins. This decoder allows the microprocessor to select the Stratix or Stratix GX device by accessing a particular address, simplifying the configuration process.

Figure 1–16. PPA Configuration Circuit



Note to Figure 1–16:
(1) The pull-up resistor should be connected to the same supply voltage as the Stratix or Stratix GX device.

The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications for t_{CSSU}, t_{WSP}, and t_{CSH} given in Table 1–10 on page 1–36. The microprocessor can also directly control the nCS and CS signals. You can tie one of the nCS or CS signals to its active state (i.e., nCS may be tied low) and toggle the other signal to control configuration.

Stratix and Stratix GX devices can serialize data internally without the microprocessor. When the Stratix or Stratix GX device is ready for the next byte of configuration data, it drives RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor strobes the next byte of configuration data into the device. Alternatively, the nRS signal can be strobed, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to

be monitored, reading the state of the configuration data by strobing nRS low saves a system I/O port. Do not drive data onto the data bus while nRS is low because it causes contention on DATA7. If the nRS pin is not used to monitor configuration, you should tie it high. To simplify configuration, the microprocessor can wait for the total time of t_{BUSY} (max) + t_{RDY2WS} + t_{W2SB} before sending the next data bit.

After configuration, the nCS, CS, nRS, nWS, and RDYnBSY pins act as user I/O pins. However, if the PPA scheme is chosen in the Quartus II software, these I/O pins are tri-stated by default in user mode and should be driven by the microprocessor. To change the default settings in the Quartus II software, select **Device & Pin Option** (Compiler Setting menu).

If the Stratix or Stratix GX device detects an error during configuration, it drives nSTATUS low to alert the microprocessor. The microprocessor can then pulse nCONFIG low to restart the configuration process. Alternatively, if the **Auto-Restart Configuration on Frame Error** option is turned on, the Stratix or Stratix GX device releases nSTATUS after a reset time-out period. After nSTATUS is released, the microprocessor can reconfigure the Stratix or Stratix GX device. At this point, the microprocessor does not need to pulse nCONFIG low.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The microprocessor must monitor the nSTATUS pin to detect errors and the CONF_DONE pin to determine when programming completes (CONF_DONE goes high one byte early in parallel mode). If the microprocessor sends all configuration data and starts initialization but CONF_DONE is not asserted, the microprocessor must reconfigure the Stratix or Stratix GX device.

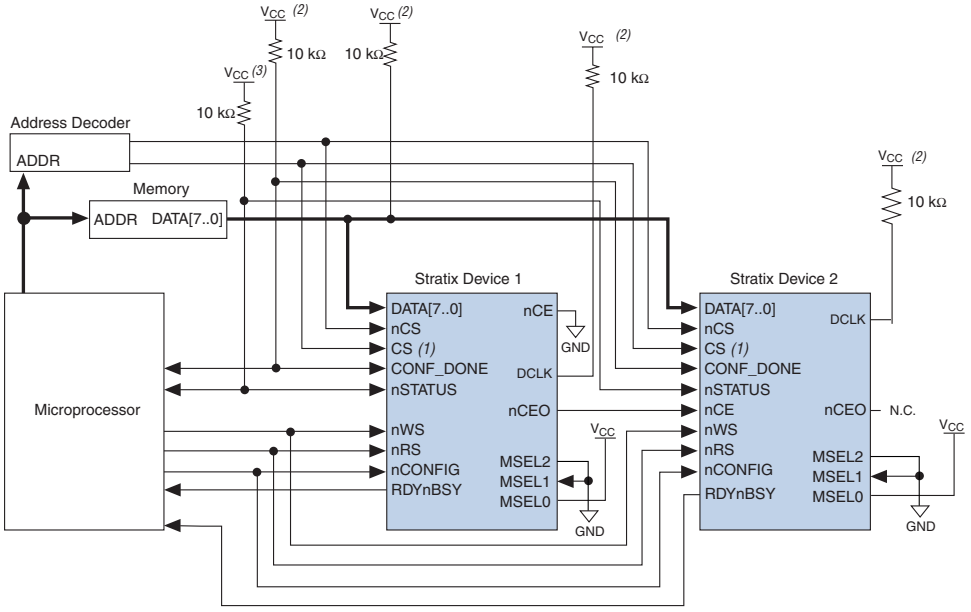
By default, the INIT_DONE is disabled. You can enable the INIT_DONE output by turning on the **Enable INIT_DONE output** option in the Quartus II software. If you do not turn on the **Enable INIT_DONE output** option in the Quartus II software, you are advised to wait for the maximum value of t_{CD2UM} (see Table 1–10) after the CONF_DONE signal goes high to ensure the device has been initialized properly and that it has entered user mode.

During configuration and initialization, and before the device enters user mode, the microprocessor must not drive the CONF_DONE signal low.

If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 40 µs).

You can also use PPA mode to configure multiple Stratix and Stratix GX devices. Multi-device PPA configuration is similar to single-device PPA configuration, except that the Stratix and Stratix GX devices are cascaded. After you configure the first Stratix or Stratix GX device, nCEO is asserted, which asserts the nCE pin on the second device, initiating configuration. Because the second Stratix or Stratix GX device begins configuration within one write cycle of the first device, the transfer of data destinations is transparent to the microprocessor. All Stratix and Stratix GX device CONF_DONE pins are tied together; therefore, all devices initialize and enter user mode at the same time. See Figure 1–17.

Figure 1–17. PPA Multi-Device Configuration Circuit



Notes to Figure 1–17:

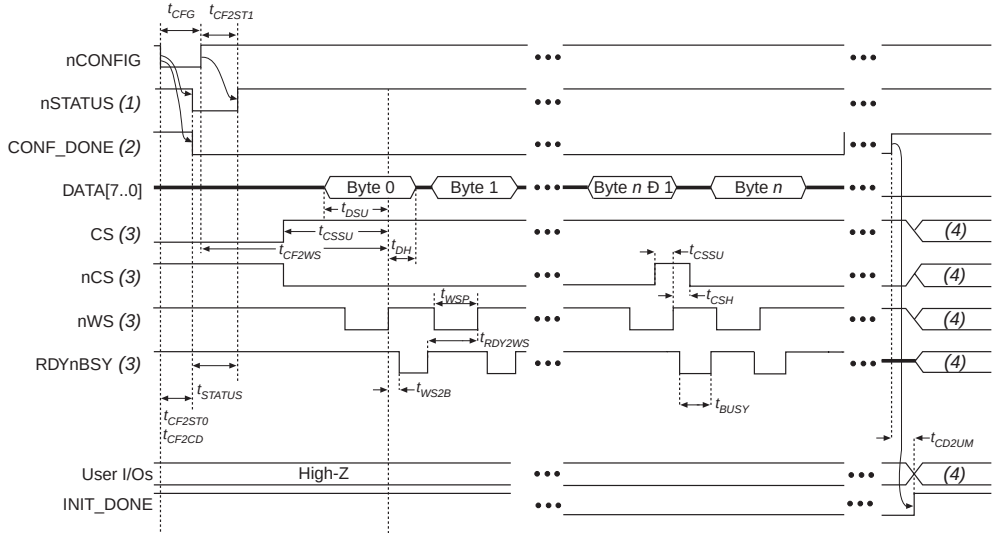
(1) If not used, you can connect the CS pin to V_{CC} directly. If not used, the nCS pin can be connected to GND directly.

(2) Connect the pull-up resistor to the same supply voltage as the Stratix or Stratix GX device.

PPA Configuration Timing

Figure 1–18 shows the Stratix and Stratix GX device timing waveforms for PPA configuration.

Figure 1–18. PPA Timing Waveforms for Stratix & Stratix GX Devices

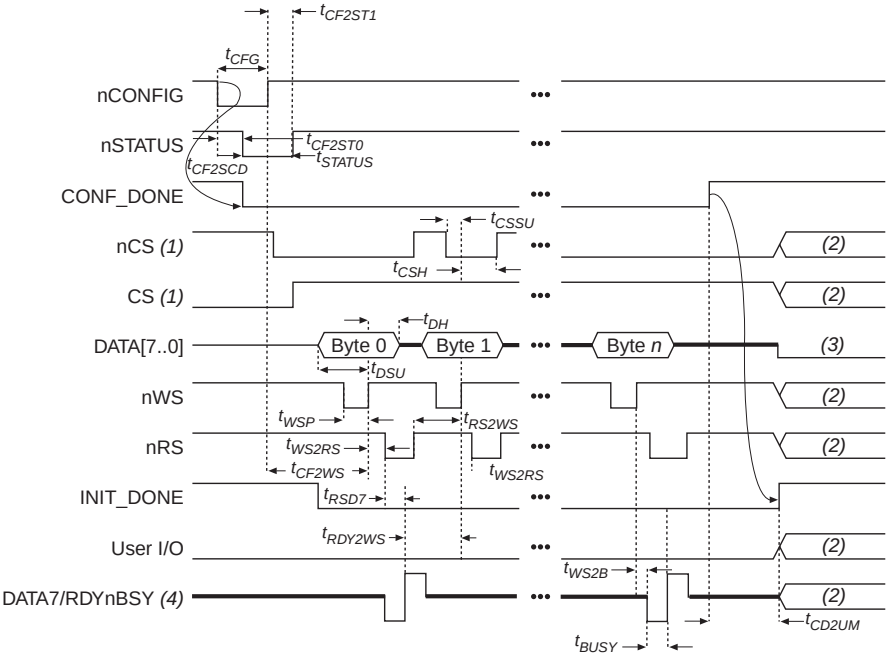


Notes to Figure 1–18:

(1) Upon power-up, nSTATUS is held low for the time of the POR delay.
(2) Upon power-up, before and during configuration, CONF_DONE is low.
(3) After configuration, the state of CS, nCS, nWS, and RDYnBSY depends on the design programmed into the Stratix or Stratix GX device.
(4) Device I/O pins are in user mode.

Figure 1–19 shows the Stratix and Stratix GX timing waveforms when using strobed nRS and nWS signals.

Figure 1–19. PPA Timing Waveforms Using Strobed nRS & nWS Signals



Notes to Figure 1–19:

(1) The user can toggle nCS or CS during configuration if the design meets the specification for t_{CSSU}, t_{WSP}, and t_{CSH}.

(2) Device I/O pins are in user mode.

(3) The DATA[7..0] pins are available as user I/Os after configuration and the state of theses pins depends on the dual-purpose pin settings. Do not leave DATA[7..0] floating. If these pins are not used in user-mode, you should drive them high or low, whichever is more convenient.

(4) DATA7 is a bidirectional pin. It represents an input for data input, but represents an output to show the status of RDYnBSY.

Table 1–10 defines the Stratix and Stratix GX timing parameters for PPA configuration

Table 1–10. PPA Timing Parameters for Stratix & Stratix GX Devices

Symbol	Parameter	Min	Max	Units
t_{CF2WS}	nCONFIG high to first rising edge on nWS	40		µs
t_{DSU}	Data setup time before rising edge on nWS	10		ns
t_{DH}	Data hold time after rising edge on nWS	0		ns
t_{CSSU}	Chip select setup time before rising edge on nWS	10		ns
t_{CSH}	Chip select hold time after rising edge on nWS	0		ns
t_{WSP}	nWS low pulse width	15		ns
t_{CFG}	nCONFIG low pulse width	40		µs
t_{WS2B}	nWS rising edge to RDYnBSY low		20	ns
t_{BUSY}	RDYnBSY low pulse width	7	45	ns
t_{RDY2WS}	RDYnBSY rising edge to nWS rising edge	15		ns
t_{WS2RS}	nWS rising edge to nRS falling edge	15		ns
t_{RS2WS}	nRS rising edge to nWS rising edge	15		ns
t_{RSD7}	nRS falling edge to DATA7 valid with RDYnBSY signal		20	ns
t_{CD2UM}	CONF_DONE high to user mode *(1)*	6	20	µs
t_{STATUS}	nSTATUS low pulse width	10	40 *(2)*	µs
t_{CF2CD}	nCONFIG low to CONF_DONE low		800	ns
t_{CF2ST0}	nCONFIG low to nSTATUS low		800	ns
t_{CF2ST1}	nCONFIG high to nSTATUS high		40 *(2)*	µs

Notes to Table 1–10:

(1) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device. If the clock source is CLKUSR, multiply the clock period by 136 to obtain this value.

(2) This value is obtained if you do not delay configuration by extending the nstatus to low pulse width.

For information on how to create configuration and programming files for this configuration scheme, see the *Software Settings* section in the *Configuration Handbook, Volume 2*.

JTAG Programming & Configuration

The JTAG has developed a specification for boundary-scan testing. This boundary-scan test (BST) architecture offers the capability to efficiently test components on printed circuit boards (PCBs) with tight lead spacing. The BST architecture can test pin connections without using physical test

probes and capture functional data while a device is operating normally. You can also use the JTAG circuitry to shift configuration data into the device.

For more information on JTAG boundary-scan testing, see *AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*.

To use the SignalTap® II embedded logic analyzer, you need to connect the JTAG pins of your Stratix device to a download cable header on your PCB.

For more information on SignalTap II, see the *Design Debugging Using SignalTap II Embedded Logic Analyzer* chapter in the *Quartus II Handbook, Volume 2*.

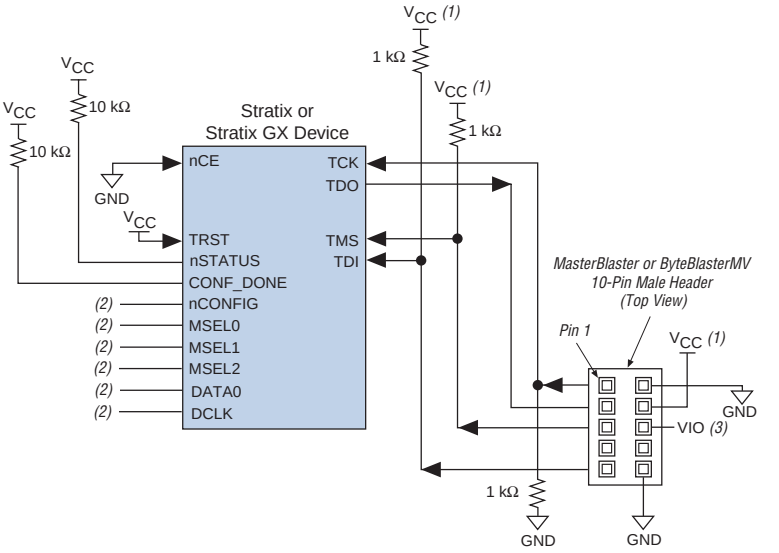
A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The four JTAG input pins (TDI, TMS, TCK and TRST) have weak, internal pull-up resistors, whose values range from 20 to 40 kΩ All other pins are tri-stated during JTAG configuration. Do not begin JTAG configuration until all other configuration is complete. Table 1–11 shows each JTAG pin's function.

Table 1–11. JTAG Pin Descriptions

Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The VCCSEL pin controls the input buffer selection.
TDO	Test data output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The high level output voltage is determined by VCCIO.
TMS	Test mode select	Input pin that provides the control signal to determine the transitions of the Test Access Port (TAP) controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The VCCSEL pin controls the input buffer selection.
TCK	Test clock input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The VCCSEL pin controls the input buffer selection.
TRST	Test reset input (optional)	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The VCCSEL pin controls the input buffer selection.

During JTAG configuration, data is downloaded to the device on the PCB through the MasterBlaster or ByteBlasterMV header. Configuring devices through a cable is similar to programming devices in-system. One difference is to connect the TRST pin to V_{CC} to ensure that the TAP controller is not reset. See Figure 1–20.

Figure 1–20. JTAG Configuration of a Single Device



Notes to Figure 1–20:

(1) You should connect the pull-up resistor to the same supply voltage as the download cable.

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

(3) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

To configure a single device in a JTAG chain, the programming software places all other devices in BYPASS mode. In BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

Stratix and Stratix GX devices have dedicated JTAG pins. You can perform JTAG testing on Stratix and Stratix GX devices before and after, but not during configuration. The chip-wide reset and output enable pins on Stratix and Stratix GX devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Stratix and Stratix GX devices, you should consider the regular configuration pins. Table 1–12 shows how you should connect these pins during JTAG configuration.

Table 1–12. Dedicated Configuration Pin Connections During JTAG Configuration

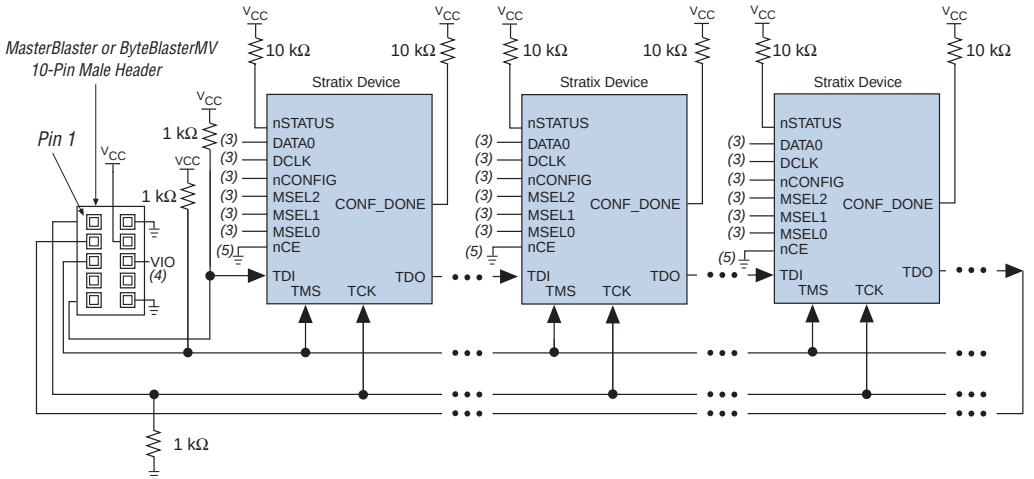
Signal	Description
nCE	On all Stratix and Stratix GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device PS, FPP or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain.
nCEO	On all Stratix and Stratix GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device. See nCE pin description above.
MSEL	These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, you should tie both pins to ground.
nCONFIG	nCONFIG must be driven high through the JTAG programming process. Driven high by connecting to V_{CC}, pulling high via a resistor, or driven by some control circuitry.
nSTATUS	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to V_{CC} individually. nSTATUS pulling low in the middle of JTAG configuration indicates that an error has occurred.
CONF_DONE	Pull to V_{CC} via a 10-kΩ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to V_{CC} individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration.
DCLK	Should not be left floating. Drive low or high, whichever is more convenient on your board.
DATA0	Should not be left floating. Drive low or high, whichever is more convenient on your board.

JTAG Programming & Configuration of Multiple Devices

When programming a JTAG device chain, one JTAG-compatible header, such as the ByteBlasterMV header, is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capacity of the download cable. However, when more than five devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

JTAG-chain device programming is ideal when the PCB contains multiple devices, or when testing the PCB using JTAG BST circuitry. Figure 1–21 shows multi-device JTAG configuration.

Figure 1–21. Multi-Device JTAG Configuration *Notes (1), (2)*



Notes to Figure 1–21:

(1) Stratix, Stratix GX, APEX™ II, APEX 20K, Mercury™, ACEX® 1K, and FLEX® 10K devices can be placed within the same JTAG chain for device programming and configuration.

(2) For more information on all configuration pins connected in this mode, see Table 1–11 on page 1–37.

(3) 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.

(4) V_{IO} is a reference voltage for the MasterBlaster output driver. V_{IO} should match the device's V_{CCIO}. See the *MasterBlaster Serial/USB Communications Cable Data Sheet* for this value.

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

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device PS, FPP and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, you should make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device drives nCE of the next device low when it has successfully been JTAG configured.

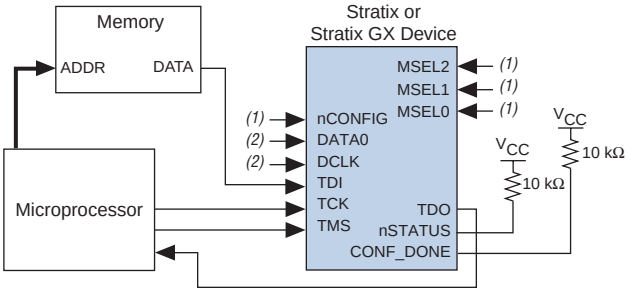
The Quartus II software verifies successful JTAG configuration upon completion. The software checks the state of CONF_DONE through the JTAG port. If CONF_DONE is not in the correct state, the Quartus II software indicates that configuration has failed. If CONF_DONE is in the correct state, the software indicates that configuration was successful.

If VCCIO is tied to 3.3 V, both the I/O pins and JTAG TDO port drive at 3.3-V levels.

Do not attempt JTAG and non-JTAG configuration simultaneously. When configuring through JTAG, allow any non-JTAG configuration to complete first.

Figure 1–22 shows the JTAG configuration of a Stratix or Stratix GX device with a microprocessor.

Figure 1–22. JTAG Configuration of Stratix & Stratix GX Devices with a Microprocessor



Notes to Figure 1–22:

(1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.

(2) Pull DATA0 and DCLK to either high or low.

Configuration with JRunner Software Driver

JRunner is a software driver that allows you to configure Altera FPGAs through the ByteBlasterMV download cable in JTAG mode. The programming input file supported is in Raw Binary File (**.rbf**) format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code has been developed for the Windows NT operating system. You can customize the code to make it run on other platforms.

For more information on the JRunner software driver, see the *JRunner Software Driver: An Embedded Solution to the JTAG Configuration White Paper* and zip file.

Jam STAPL Programming & Test Language

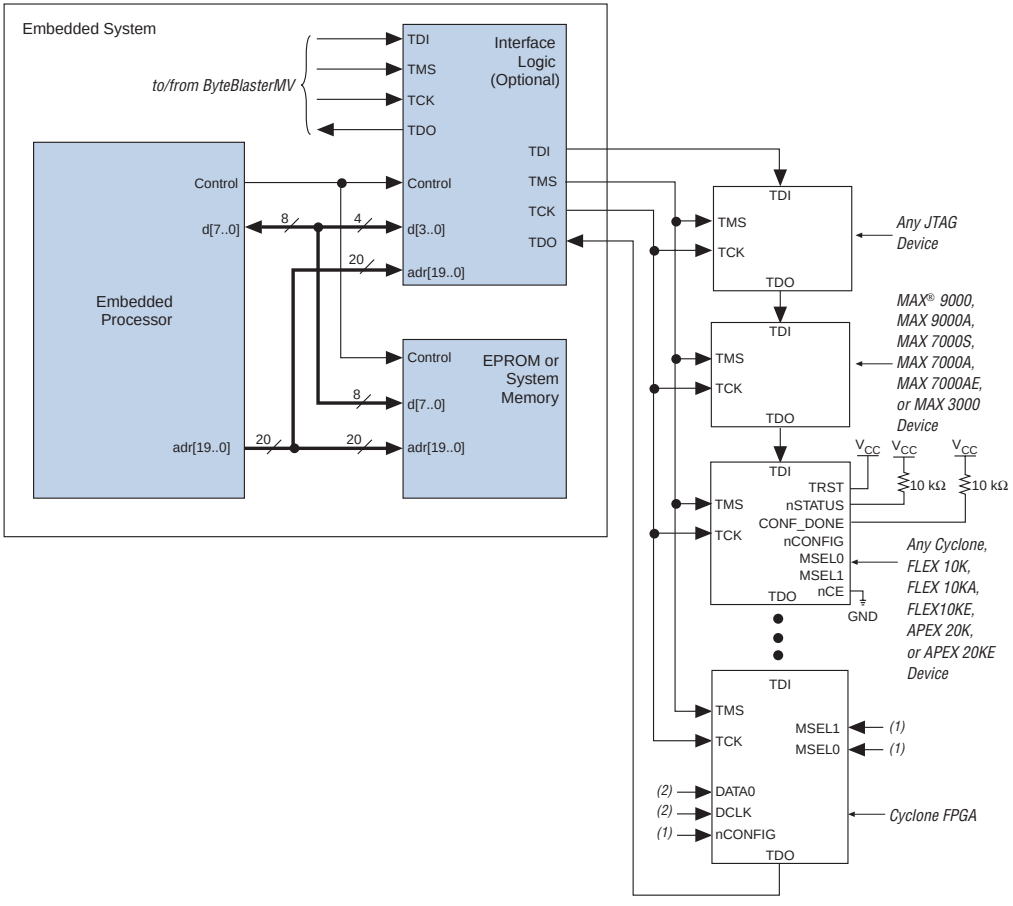
The Jam™ Standard Test and Programming Language (STAPL), JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

Connecting the JTAG Chain to the Embedded Processor

There are two ways to connect the JTAG chain to the embedded processor. The most straightforward method is to connect the embedded processor directly to the JTAG chain. In this method, four of the processor pins are dedicated to the JTAG interface, saving board space but reducing the number of available embedded processor pins.

Figure 1–23 illustrates the second method, which is to connect the JTAG chain to an existing bus through an interface PLD. In this method, the JTAG chain becomes an address on the existing bus. The processor then reads from or writes to the address representing the JTAG chain.

Figure 1–23. Embedded System Block Diagram



Notes to Figure 1–23:

(1) Connect the nCONFIG, MSEL2, MSEL1, and MSEL0 pins to support a non-JTAG configuration scheme. If your design only uses JTAG configuration, connect the nCONFIG pin to V_{CC} and the MSEL2, MSEL1, and MSEL0 pins to ground.

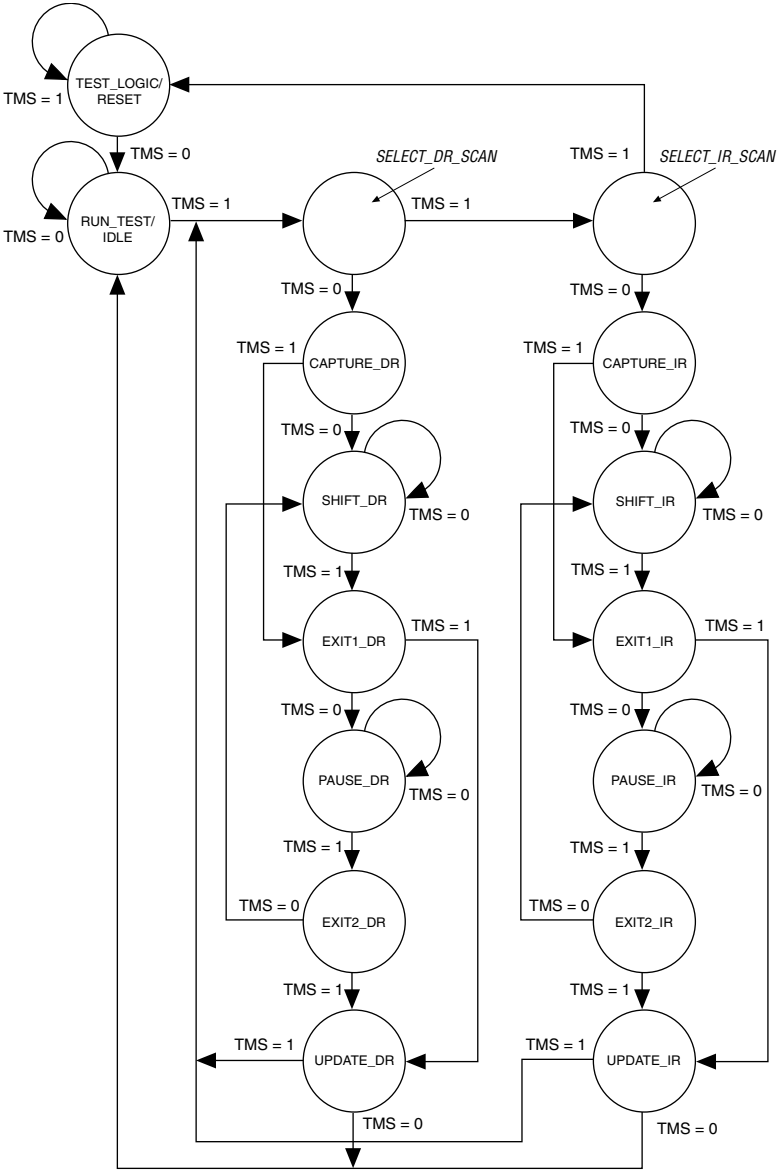
(2) Pull DATA0 and DCLK to either high or low.

Both JTAG connection methods should include space for the MasterBlaster or ByteBlasterMV header connection. The header is useful during prototyping because it allows you to verify or modify the Stratix or Stratix GX device's contents. During production, you can remove the header to save cost.

Program Flow

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine. The TAP controller is a 16-state state machine that is clocked on the rising edge of TCK, and uses the TMS pin to control JTAG operation in a device. Figure 1–24 shows the flow of an IEEE Std. 1149.1 TAP controller state machine.

Figure 1–24. JTAG TAP Controller State Machine



While the Jam Player provides a driver that manipulates the TAP controller, the Jam Byte-Code File (**.jbc**) provides the high-level intelligence needed to program a given device. All Jam instructions that

send JTAG data to the device involve moving the TAP controller through either the data register leg or the instruction register leg of the state machine. For example, loading a JTAG instruction involves moving the TAP controller to the SHIFT_IR state and shifting the instruction into the instruction register through the TDI pin. Next, the TAP controller is moved to the RUN_TEST/IDLE state where a delay is implemented to allow the instruction time to be latched. This process is identical for data register scans, except that the data register leg of the state machine is traversed.

The high-level Jam instructions are the DRSCAN instruction for scanning the JTAG data register, the IRSCAN instruction for scanning the instruction register, and the WAIT command that causes the state machine to sit idle for a specified period of time. Each leg of the TAP controller is scanned repeatedly, according to instructions in the JBC file, until all of the target devices are programmed.

Figure 1–25 illustrates the functional behavior of the Jam Player when it parses the JBC file. When the Jam Player encounters a DRSCAN, IRSCAN, or WAIT instruction, it generates the proper data on TCK, TMS, and TDI to complete the instruction. The flow diagram shows branches for the DRSCAN, IRSCAN, and WAIT instructions. Although the Jam Player supports other instructions, they are omitted from the flow diagram for simplicity.

Figure 1–25. Jam Player Flow Diagram (Part 1 of 2)

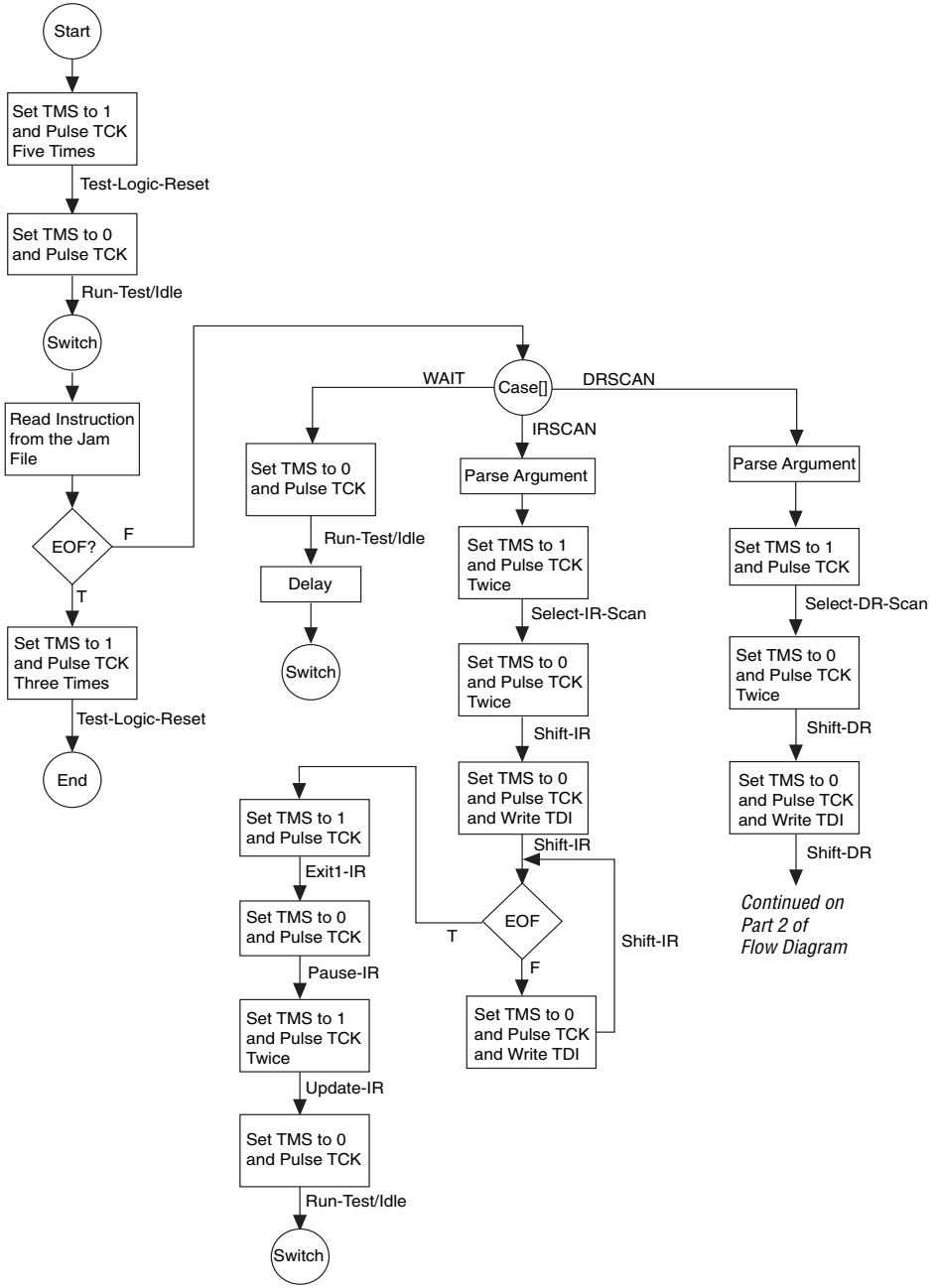
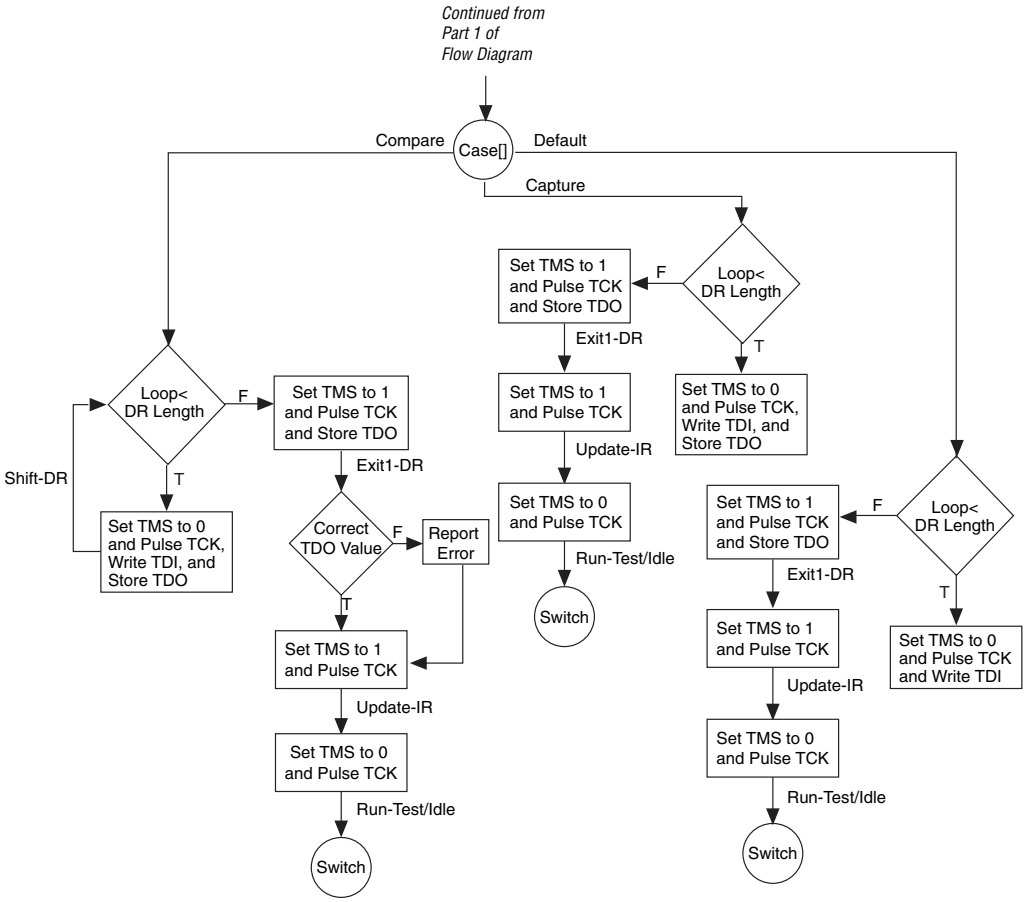


Figure 1–26. Jam Player Flow Diagram (Part 2 of 2)



Execution of a Jam program starts at the beginning of the program. The program flow is controlled using GOTO, CALL/RETURN, and FOR/NEXT structures. The GOTO and CALL statements see labels that are symbolic names for program statements located elsewhere in the Jam program. The language itself enforces almost no constraints on the organizational structure or control flow of a program.

The Jam language does not support linking multiple Jam programs together or including the contents of another file into a Jam program.

Jam Instructions

Each Jam statement begins with one of the instruction names listed in Table 1–13. The instruction names, including the names of the optional instructions, are reserved keywords that you cannot use as variable or label identifiers in a Jam program.

Table 1–13. Instruction Names

BOOLEAN	INTEGER	PREIR
CALL	IRSCAN	PRINT
CRC	IRSTOP	PUSH
DRSCAN	LET	RETURN
DRSTOP	NEXT	STATE
EXIT	NOTE	WAIT
EXPORT	POP	VECTOR *(1)*
FOR	POSTDR	VMAP *(1)*
GOTO	POSTIR	–
IF	PREDR	–

Note to Table 1–13:
(1) This instruction name is an optional language extension.

Table 1–14 shows the state names that are reserved keywords in the Jam language. These keywords correspond to the state names specified in the IEEE Std. 1149.1 JTAG specification.

Table 1–14. Reserved Keywords (Part 1 of 2)

IEEE Std. 1149.1 JTAG State Names	Jam Reserved State Names
Test-Logic-Reset	RESET
Run-Test-Idle	IDLE
Select-DR-Scan	DRSELECT
Capture-DR	DRCAPTURE
Shift-DR	DRSHIFT
Exit1-DR	DREXIT1
Pause-DR	DRPAUSE
Exit2-DR	DREXIT2
Update-DR	DRUPDATE
Select-IR-Scan	IRSELECT

Table 1–14. Reserved Keywords (Part 2 of 2)	
IEEE Std. 1149.1 JTAG State Names	**Jam Reserved State Names**
Capture-IR	IRCAPTURE
Shift-IR	IRSHIFT
Exit1-IR	IREXIT1
Pause-IR	IRPAUSE
Exit2-IR	IREXIT2
Update-IR	IRUPDATE

Example Jam File that Reads the IDCODE

Figure 1–27 illustrates the flexibility and utility of the Jam STAPL. The example reads the IDCODE out of a single device in a JTAG chain.

The array variable, I_IDCODE, is initialized with the IDCODE instruction bits ordered the LSB first (on the left) to most significant bit (MSB) (on the right). This order is important because the array field in the IRSCAN instruction is always interpreted, and sent, MSB to LSB.

Figure 1–27. Example Jam File Reading IDCODE

```
BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000; 'assumed
BOOLEAN ONES_DATA[32] = HEX FFFFFFFF;
INTEGER i;
'Set up stop state for IRSCAN
IRSTOP IRPAUSE;
'Initialize device
STATE RESET;
IRSCAN 10, I_IDCODE[0..9]; 'LOAD IDCODE INSTRUCTION
STATE IDLE;
WAIT 5 USEC, 3 CYCLES;
DRSCAN 32, ONES_DATA[0..31], CAPTURE
read_data[0..31];
'CAPTURE IDCODE
PRINT "IDCODE:";
FOR i=0 to 31;
PRINT read_data[i];
NEXT i;
EXIT 0;
```

Configuring Using the MicroBlaster Driver

The MicroBlaster™ software driver allows you to configure Altera devices in an embedded environment using PS or FPP mode. The MicroBlaster software driver supports a Raw Binary File (**.rbf**) programming input file. The source code is developed for the Windows NT operating system, although you can customize it to run on other operating systems. For more information on the MicroBlaster software driver, go to the Altera web site (**www.altera.com**).

Device Configuration Pins

The following tables describe the connections and functionality of all the configuration related pins on the Stratix or Stratix GX device. Table 1–15 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 1 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
VCCSEL	N/A	All	Input	Dedicated input that selects which input buffer is used on the configuration input pins; nCONFIG, DCLK, RUnLU, nCE, nWS, nRS, CS, nCS and CLKUSR. The VCCSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. See the "V_{CCSEL} Pins" section for more details.
PORSEL	N/A	All	Input	Dedicated input which selects between a POR time of 2 ms or 100 ms. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects a POR time of about 2 ms and a logic low selects POR time of about 100 ms. The PORSEL input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 2 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nIO_PULLUP	N/A	All	Input	Dedicated input that chooses whether the internal pull-ups on the user I/Os and dual-purpose I/Os (DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, PGM[], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) turns off the weak internal pull-ups, while a logic low turns them on. The nIO_PULLUP input buffer is powered by V_{CCINT} and has an internal 2.5 kΩ pull-down resistor that is always active.
MSEL[2..0]	N/A	All	Input	3-bit configuration input that sets the Stratix or Stratix GX device configuration scheme. See Table 1–2 for the appropriate connections. These pins can be connected to V_{CCIO} of the I/O bank they reside in or ground. This pin uses Schmitt trigger input buffers.
nCONFIG	N/A	All	Input	Configuration control input. Pulling this pin low during user-mode causes the FPGA to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level initiates a reconfiguration. If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to V_{CC} or to the configuration device's nINIT_CONF pin. This pin uses Schmitt trigger input buffers.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 3 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nSTATUS	N/A	All	Bidirectional open-drain	The device drives nSTATUS low immediately after power-up and releases it after the POR time. Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device. Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state. Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low causes the configuration device to attempt to configure the FPGA, but since the FPGA ignores transitions on nSTATUS in user-mode, the FPGA does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used. This pin uses Schmitt trigger input buffers.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 4 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
CONF_DONE	N/A	All	Bidirectional open-drain	Status output. The target FPGA drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE. Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize. Driving CONF_DONE low after configuration and initialization does not affect the configured device. The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. When using EPC2 devices, only external 10-kΩ pull-up resistors should be used. This pin uses Schmitt trigger input buffers.
nCE	N/A	All	Input	Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain. The nCE pin must also be held low for successful JTAG programming of the FPGA. This pin uses Schmitt trigger input buffers.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 5 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCEO	N/A	All Multi-Device Schemes	Output	Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The voltage levels driven out by this pin are dependent on the V_{CCIO} of the I/O bank it resides in.
DCLK	N/A	Synchronous configuration schemes (PS, FPP)	Input (PS, FPP)	In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the FPGA on the rising edge of DCLK. In PPA mode, DCLK should be tied high to V_{CC} to prevent this pin from floating. After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK is driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device. This pin uses Schmitt trigger input buffers.
DATA0	I/O	PS, FPP, PPA	Input	Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin. The V_{IH} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in. After configuration, DATA0 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings. After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 6 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
DATA[7..1]	I/O	Parallel configuration schemes (FPP and PPA)	Inputs	Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0]. The V_{IH} and V_{IL} levels for these pins are dependent on the V_{CCIO} of the I/O banks that they reside in. In serial configuration schemes, they function as user I/Os during configuration, which means they are tri-stated. After PPA or FPP configuration, DATA[7..1] are available as a user I/Os and the state of these pin depends on the **Dual-Purpose Pin** settings.
DATA7	I/O	PPA	Bidirectional	In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low. The V_{IL} and V_{IL} levels for this pin are dependent on the V_{CCIO} of the I/O bank that it resides in. In serial configuration schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
nWS	I/O	PPA	Input	Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0] pins. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nWS is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 7 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nRS	I/O	PPA	Input	Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin. If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nRS is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.
RDYnBSY	I/O	PPA	Output	Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte. In PPA configuration schemes, this pin drives out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, RDYnBSY is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings.

Table 1–15. Dedicated Configuration Pins on the Stratix or Stratix GX Device (Part 8 of 8)

Pin Name	User Mode	Configuration Scheme	Pin Type	Description
nCS/CS	I/O	PPA	Input	Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization. During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to GND while CS is toggled to control configuration.In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated. After PPA configuration, nCS and CS are available as a user I/Os and the state of these pins depends on the **Dual-Purpose Pin** settings.
RUnLU	N/A if using Remote Configuration; I/O if not	Remote Configuration in FPP, PS or PPA	Input	Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update. When not using remote update or local update configuration modes, this pins is available as general-purpose user I/O pin.
PGM[2..0]	N/A if using Remote Configuration; I/O if not using	Remote Configuration in FPP, PS or PPA	Input	These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote configuration mode. When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins.

Table 1–16 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore during configuration, these pins function as user I/O pins and are tri-stated with weak pull-ups.

Table 1–16. Optional Configuration Pins

Pin Name	User Mode	Pin Type	Description
CLKUSR	N/A if option is on. I/O if option is off.	Input	Optional user-supplied clock input. Synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock (CLKUSR)** option in the Quartus II software.
INIT_DONE	N/A if option is on. I/O if option is off.	Output open-drain	Status pin. Can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high and the FPGA enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software.
DEV_OE	N/A if option is on. I/O if option is off.	Input	Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/Os are tri-stated. When this pin is driven high, all I/Os behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable (DEV_OE)** option in the Quartus II software.
DEV_CLRn	N/A if option is on. I/O if option is off.	Input	Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset (DEV_CLRn)** option in the Quartus II software.

Table 1–17 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. If you plan to use the SignalTap II Embedded Logic Analyzer, you will need to connect the JTAG pins of your device to a JTAG header on your board.

Table 1–17. Dedicated JTAG pins

Pin Name	User Mode	Pin Type	Description
TDI	N/A	Input	Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers.
TDO	N/A	Output	Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected.
TMS	N/A	Input	Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to V_{CC}. This pin uses Schmitt trigger input buffers.
TCK	N/A	Input	The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.
TRST	N/A	Input	Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. This pin uses Schmitt trigger input buffers.

2. Remote System Configuration with Stratix & Stratix GX Devices

S52015-3.1

Introduction

Altera® Stratix® and Stratix GX devices are the first programmable logic devices (PLDs) featuring dedicated support for remote system configuration. Using remote system configuration, a Stratix or Stratix GX device can receive new configuration data from a remote source, update the flash memory content (through enhanced configuration devices or any other storage device), and then reconfigure itself with the new data.

Like all Altera SRAM-based devices, Stratix and Stratix GX devices support standard configuration modes such as passive serial (PS), fast passive parallel (FPP), and passive parallel asynchronous (PPA). You can use the standard configuration modes with remote system configuration.

This chapter discusses remote system configuration of Stratix and Stratix GX devices, and how to interface them with enhanced configuration devices to enable this capability. This document also explains some related remote system configuration topics, such as the watchdog timer, remote system configuration registers, and factory or application configurations files. The Quartus® II software (version 2.1 and later) supports remote system configuration.

Remote Configuration Operation

Remote system configuration has three major parts:

- The Stratix or Stratix GX device receives updated or new data from a remote source over a network (or through any other source that can transfer data). You can implement a Nios™ (16-bit ISA) or Nios® II (32-bit ISA) embedded processor within either a Stratix or Stratix GX device or an external processor to control the read and write functions of configuration files from the remote source to the memory device.
- The new or updated information is stored into the memory device, which can be an enhanced configuration device, industry-standard flash memory device, or any other storage device (see Figure 2–2).
- The Stratix or Stratix GX device updates itself with the new data from the memory.

Figure 2–1 shows the concept of remote system configuration in Stratix and Stratix GX devices.

Figure 2–1. Remote System Configuration with Stratix & Stratix GX Devices

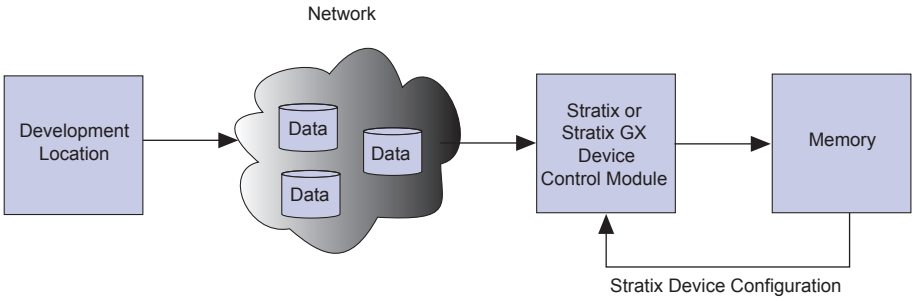
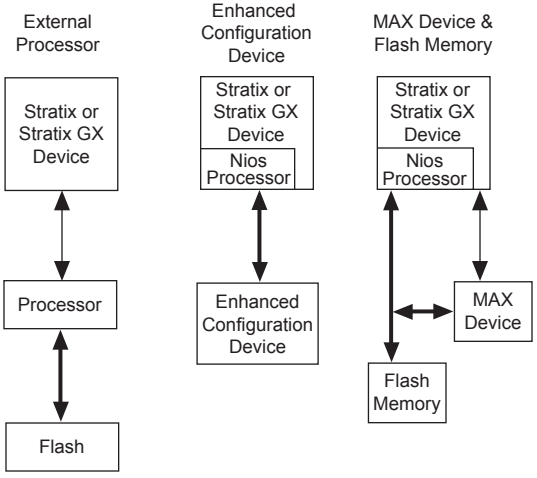


Figure 2–2. Different Options for Remote System Configuration



Remote System Configuration Modes

Stratix and Stratix GX device remote system configuration has two modes: remote configuration mode and local configuration mode. Table 2–1 shows the pin selection settings for each configuration mode.

Table 2–1. Standard, Remote & Local Configuration Options Note (1)

RUnLU (2)	MSEL[2] (3)	MSEL[1..0]	System Configuration Mode	Configuration Mode
–	0	00	Standard	FPP
–	0	01	Standard	PPA
–	0	10	Standard	PS
1	1	00	Remote	FPP
1	1	01	Remote	PPA
1	1	10	Remote	PS
0	1	00	Local	FPP
0	1	01	Local	PPA
0	1	10	Local	PS

Notes to Table 2–1:

(1) For detailed information on standard PS, FPP, and PPA models, see the *Configuring Stratix & Stratix GX Devices* chapter of the *Stratix Device Handbook, Volume 2*.

(2) In Stratix and Stratix GX devices, the RUnLU (remote update/local update) pin, selects between local or remote configuration mode.

(3) The MSEL[2] select mode selects between standard or remote system configuration mode.

Remote Configuration Mode

Using remote configuration mode, you can manage up to seven different application configurations for Stratix and Stratix GX devices. The seven-configuration-file limit is due to the number of pages that the PGM[] pins in the Stratix or Stratix GX device and enhanced configuration devices can select.

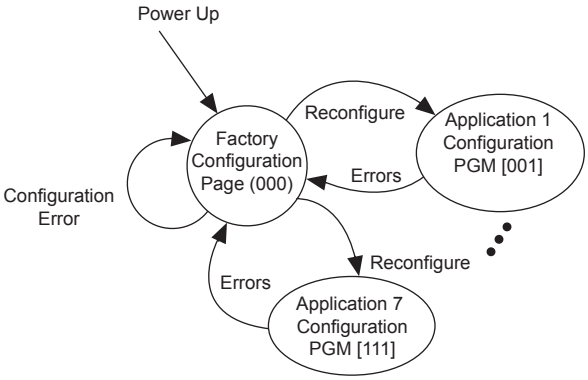
If more than seven files are sent to a system using remote configuration mode, previous files are overwritten.

Stratix and Stratix GX devices support remote configuration mode for PS, FPP, and PPA modes. Specify remote configuration mode by setting the MSEL2 and RUnLU pins to high. (See Table 2–1).

On power-up in remote configuration mode, the Stratix or Stratix GX device loads the user-specified factory configuration file, located in the default page address 000 in the enhanced configuration device. After the device configures, the remote configuration control register points to the

page address of the application configuration that should be loaded into the Stratix or Stratix GX device. If an error occurs during user mode of an application configuration, the device reloads the default factory configuration page. Figure 2–3 shows a diagram of remote configuration mode.

Figure 2–3. Remote Configuration Mode



Local Configuration Mode

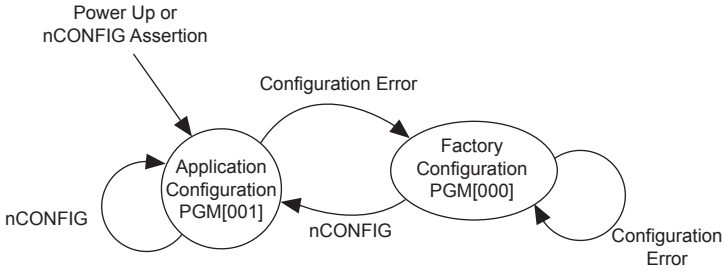
Local configuration mode—a simplified version of remote configuration mode—is suitable for systems that load an application immediately upon power-up. In this mode you can only use one application configuration, which you can update either remotely or locally.

In local configuration mode, upon power-up, or when `nCONFIG` is asserted, the Stratix or Stratix GX device loads the application configuration immediately. Factory configuration loads only if an error occurs during the application configuration's user mode. If you use an enhanced configuration device, page address `001` is the location for the application configuration data, and page address `000` is the location for the factory configuration data.

If the configuration data at page address `001` does not load correctly due to cyclic redundancy code (CRC) failure, or it times-out of the enhanced configuration device, or the external processor times-out, then the factory configuration located at the default page (page address `000`) loads into the Stratix or Stratix GX device.

In local configuration mode (shown in Figure 2–4), the user watchdog timer is disabled. For more information on the watchdog timer, see "Watchdog Timer" on page 2–7.

Figure 2–4. Local Configuration Mode



In local configuration mode, one application configuration is available to the device. For remote or local configuration mode selection, see Table 2–1.

Remote System Configuration Components

The following components are used in Stratix and Stratix GX devices to support remote and local configuration modes:

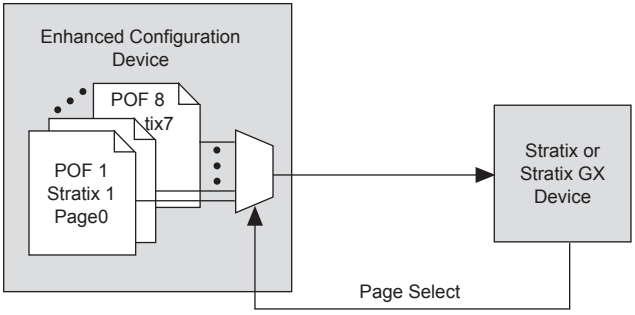
- Page mode feature
- Factory configuration
- Application configuration
- Watchdog timer
- Remote update sub-block
- Remote configuration registers

A description of each component follows.

Page Mode Feature

The page mode feature enables Stratix and Stratix GX devices to select a location to read back data for configuration. The enhanced configuration device can receive and store up to eight different configuration files (one factory and seven application files). Selection of pages to read from is performed through the `PGM[2..0]` pins on the Stratix or Stratix GX device and enhanced configuration devices. These pins in the Stratix or Stratix GX device can be designated user I/O pins during standard configuration mode, but in remote system configuration mode, they are dedicated output pins. Figure 2–5 shows the page mode feature in Stratix or Stratix GX devices and enhanced configuration devices.

Figure 2–5. Page Mode Feature in Stratix or Stratix GX Devices & Enhanced Configuration Devices



Upon power-up in remote configuration mode, the factory configuration (see description below) selects the user-specified page address through the Stratix or Stratix GX PGM[2..0] output pins. These pins drive the PGM[2..0] input pins of the enhanced configuration device and select the requested page in the memory.

If an intelligent host is used instead of an enhanced configuration device, you should create logic in the intelligent host to support page mode settings similar to that in enhanced configuration devices.

Factory Configuration

Factory configuration is the default configuration data setup. In enhanced configuration devices, this default page address is 000. Factory configuration data is written into the memory device only once by the system manufacturer and should not be remotely updated or altered. In remote configuration mode, the factory configuration loads into the Stratix or Stratix GX device upon power-up.

The factory configuration specifications are as follows:

- Receives new configuration data and writes it to the enhanced configuration or other memory devices
- Determines the page address for the next application configuration that should be loaded to the Stratix or Stratix GX device
- Upon an error in the application configuration, the system reverts to the factory configuration
- Determines the reason for any application configuration error
- Determines whether to enable or disable the user watchdog timer for application configurations

- Determines the user watchdog timer's settings if the timer is enabled (remote configuration mode)
- If the user watchdog timer is not reset after a predetermined amount of time, it times-out and the system loads the factory configuration data back to the Stratix or Stratix GX device

If a system encounters an error while loading application configuration data, or if the device re-configures due to `nCONFIG` assertion, the Stratix or Stratix GX device loads the factory configuration. The remote system configuration register determines the reason for factory re-configuration. Based on this information, the factory configuration determines which application configuration needs to be loaded.

Application Configuration

The application configuration is the configuration data received from the remote source and updated into different locations or pages of the memory storage device (excluding the factory default page).

Watchdog Timer

A watchdog timer is a circuit that determines whether another mechanism functions properly. The watchdog timer functions like a time-delay relay that remains in the reset state while an application runs properly. This action periodically sends a reset command from the working application to the watchdog timer. Stratix and Stratix GX devices are equipped with a built-in watchdog timer for remote system configuration.

A user watchdog timer prevents a faulty application configuration from indefinitely stalling the Stratix or Stratix GX device. The timer functions as a counter that counts down from an initial value, which is loaded into the device from the factory configuration. This is a 29-bit counter, but you use only the upper 12 bits to set the value for the watchdog timer. You specify the counter value according to your design needs.

The timer begins counting once the Stratix or Stratix GX device goes into user mode. If the application configuration does not reset the user watchdog timer after the specified time, the timer times-out. At this point, the Stratix or Stratix GX device is re-configured by loading the factory configuration and resetting the user watchdog timer.

The watchdog timer is disabled in local configuration mode.

Remote Update Sub-Block

The remote update sub-block is responsible for administrating the remote configuration feature. This sub-block, which is controlled by a remote configuration state machine, generates the control signals required to control different remote configuration registers.

Remote Configuration Registers

Remote configuration registers are a series of registers required to keep track of page addresses and the cause of configuration errors. Table 2–2 gives descriptions of the registers' functions. You can control both the update and shift registers; the status and control registers are controlled by internal logic, but can be read via the shift register.

Table 2–2. Remote Configuration Registers

Register	Description
Control register	This register contains the current page address, the watchdog timer setting, and one bit specifying if the current configuration is a factory or application configuration. During a capture in an application configuration, this register is read into the shift register.
Update register	This register contains the same data as the control register, except that it is updated by the factory configuration. The factory configuration updates the register with the values to be used in the control register on the next re-configuration. During capture in a factory configuration, this register is read into the shift register.
Shift register	This register is accessible by the core logic and allows the update, status, and control registers to be written and sampled by the user logic. The update register can only be updated by the factory configuration in remote configuration mode.
Status register	This register is written into by the remote configuration block on every re-configuration to record the cause of the re-configuration. This information is used by factory configuration to determine the appropriate action following a re-configuration.

Figure 2–6 shows the control, update, shift, and status registers and the data path used to control remote system configuration.

Figure 2–6. Remote Configuration Registers & Related Data Path

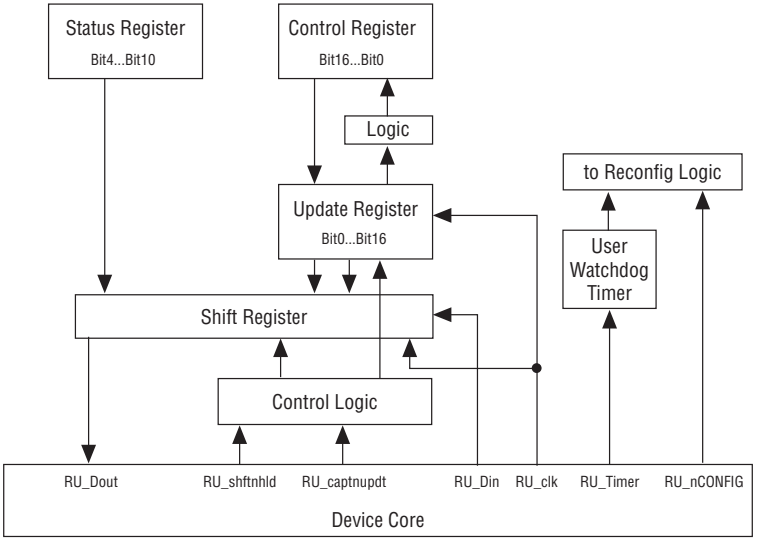


Table 2–3 describes the user configuration signals that are driven to/from the device logic array. The remote configuration logic has one input signal to the device logic array and six output signals from the device logic array.

Table 2–3. User Configuration Signals To/From Device Core (Part 1 of 2)

Signal Name	To/From Device Core	Description
RU_Timer	Output from the core to the remote update block	Request from the application to reset the user watchdog timer with its initial count. A falling edge of this signal triggers a reset of the user watchdog timer.
RU_nCONFIG	Output from the core to the remote update block	When driven low, this signal triggers the device to reconfigure. If requested by the factory configuration, the application configuration specified in the remote update control register is loaded. If requested by the application configuration, the factory configuration is loaded.
RU_Clk	Output from the core to the remote update block	Clocks the remote configuration shift register so that the contents of the status and control registers can be read out, and the contents of update register can be loaded. The shift register latches data on the rising edge of the RU_Clk.

Table 2–3. User Configuration Signals To/From Device Core (Part 2 of 2)

Signal Name	To/From Device Core	Description
RU_shftnhld	Output from the core to the remote update block	If its value is "1", the remote configuration shift register shifts data on the rising edge of RU_Clk. It its value is "0" and RU_captnupdt is "0", the shift register updates the update register. If its value is "0", and RU_captnupdt is "1", the shift register captures the status register and either the control or update register (depending on whether the configuration is factory or application).
RU_captnupdt	Output from the core to the remote update block	When RU_captnupdt is at value "1" and RU_shftnhld is at value"0", the system specifies that the remote configuration shift register should be written with the content of the status register and either the update register (in a factory configuration) or the control register (in an application configuration). This shift register is loaded on the rising edge of RU_Clk. When RU_captnupdt is at value "0" and RU_shftnhld is at value"0", the system specifies that the remote configuration update register should be written with the content of the shift register in a factory configuration. The update register is loaded on the rising edge of RU_Clk. This pin is enabled only for factory configuration in remote configuration mode (it is disabled for the application configuration in remote configuration or for local configuration modes). If RU_shftnhld is at value "1", RU_captnupdt has no function.
RU_Din	Output from the core to the remote update block	Data to be written into the remote configuration shift register on the rising edge of RU_Clk. To load into the shift register, RU_shftnhld must be asserted.
RU_Dout	Input to the core from the remote update block	Output of the remote configuration shift register to be read by core logic. New data arrives on each rising edge of RU_Clk.

All of the seven device core signals (see Figure 2–6), are enabled for both remote and local configuration for both factory and application configuration, except RU_Timer and RU_captnupdt. Figure 2–7 and Table 2–4 specify the content of control register upon power-on reset (POR).

The difference between local configuration and remote configuration is how the control register is updated during a re-configuration and which core signals are enabled.

Figure 2–7. Remote System Configuration Control Register

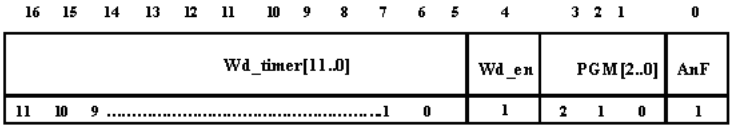


Table 2–4 shows the content of the control register upon POR.

Table 2–4. Control Register Contents

Parameter	Definition	POR Reset Value	Comment
AnF	Current configuration is factory or applications	1 bit '1'	Applications
		1 bit '0'	Factory
PGM[2..0]	Page mode selection	3 bits '001'	Local configuration
		3 bits '000'	Remote configuration
Wd_en	User watchdog timer enable	1 bit '0'	–
Wd_timer [11..0]	User watchdog timer time-out value	12 bits '0'	High order bits of 29 bit counter

The status register specifies the reason why re-configuration has occurred and determines if the re-configuration was due to a CRC error, nSTATUS pulled low due to an error, the device core caused an error, nCONFIG was reset, or the watchdog timer timed-out. Figure 2–8 and Table 2–5 specify the content of the status register.

Figure 2–8. Remote System Configuration Status Register

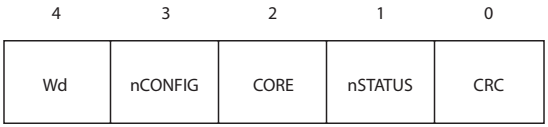


Table 2–5 shows the content of the status register upon POR.

Table 2–5. Status Register Contents

Parameter	Definition	POR Reset Value
CRC (from configuration)	CRC caused re-configuration	1 bit '0'
nSTATUS	nSTATUS caused re-configuration	1 bit '0'
CORE *(1)*	Device core caused re-configuration	1 bit '0'
nCONFIG	NCONFIG caused re-configuration	1 bit '0'
Wd	Watchdog Timer caused re-configuration	1 bit '0'

Note to Table 2–5:

(1) Core re-configuration enforces the system to load the application configuration data into the Stratix or Stratix GX device. This occurs after factory configuration specifies the appropriate application configuration data.

Quartus II Software Support

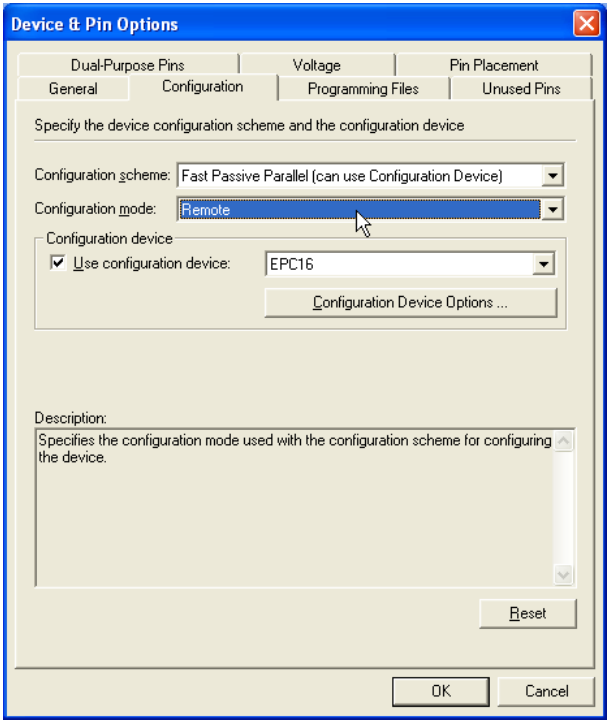
The Quartus II software supports implementation of both remote and local configuration modes in your Stratix or Stratix II device. To include the remote or local configuration feature to your design, select remote or local as the configuration mode under the **Device & Pin Options** compiler settings (prior to compilation). This selection reserves the dual-purpose RUnLU and PGM[2:0] pins for use as dedicated inputs in remote/local configuration modes.

To set the configuration mode as remote or local, follow these steps (See Figure 2–9):

1. Open the **Device & Pin Options** settings window under the **Assignments** menu.
2. Select **Device & Pin Options** dialog box. The **Device & Pin Options** dialog box is displayed.
3. Click the **Configuration** tab.
4. In the **Configuration mode** list, select **Remote** or **Local**.

The Standard mode selection disables the remote system configuration feature. In addition to the mode selection, you can specify the configuration scheme and configuration device (if any) used by your setup.

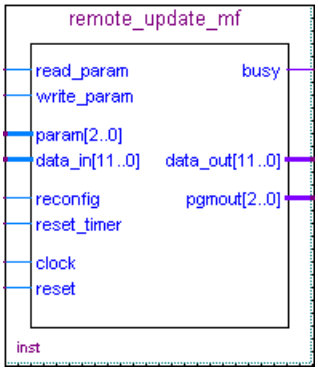
Figure 2–9. Device & Pin Options Dialog Box



Additionally, the remote configuration mode requires you to either instantiate the `altremote_update` megafunction or the WYSIWYG (what-you-see-is-what-you-get) atom into your design. Without this atom or megafunction, you are not be able to access the dedicated remote configuration circuitry or registers within the Stratix or Stratix GX device. See Figure 2–10 for a symbol of the `altremote_update` megafunction.

The local configuration mode, however, can be enabled with only the device **Configuration Options** compiler setting.

Figure 2–10. altremote_update Megafunction Symbol



altremote_update Megafunction

A remote update megafunction, `altremote_update`, is provided in the Quartus II software to provide a memory-like interface to allow for easy control of the remote update parameters. Tables 2–6 and 2–7 describe the input and output ports available on the `altremote_update` megafunction. Table 2–8 shows the `param[2..0]` bit settings.

Table 2–6. Input Ports of the altremote_update Megafunction (Part 1 of 2)

Port Name	Required	Source	Description
clock	Y	Logic Array	Clock input to the `altremote_update` block. All operations are performed with respects to the rising edge of this clock.
reset	Y	Logic Array	Asynchronous reset, which is used to initialize the remote update block. To ensure proper operation, the remote update block must be reset before first accessing the remote update block. This signal is not affected by the busy signal and will reset the remote update block even if busy is logic high. This means that if the reset signal is driven logic high during writing of a parameter, the parameter will not be properly written to the remote update block.
reconfig	Y	Logic Array	When driven logic high, reconfiguration of the device is initiated using the current parameter settings in the remote update block. If busy is asserted, this signal is ignored. This is to ensure all parameters are completely written before reconfiguration begins.
reset_timer	N	Logic Array	This signal is required if you are using the watchdog timer feature. A logic high resets the internal watchdog timer. This signal is not affected by the busy signal and can reset the timer even when the remote update block is busy. If this port is left connected, the default value is 0.

Table 2–6. Input Ports of the altremote_update Megafunction (Part 2 of 2)

Port Name	Required	Source	Description
read_param	N	Logic Array	Once `read_param` is sampled as a logic high, the busy signal is asserted. While the parameter is being read, the busy signal remains asserted, and inputs on `param[]` are ignored. Once the busy signal is deactivated, the next parameter can be read. If this port is left unconnected, the default value is 0.
write_param	N	Logic Array	This signal is required if you intend on writing parameters to the remote update block. When driven logic high, the parameter specified on the `param[]` port should be written to the remote update block with the value on `data_in[]`. The number of valid bits on `data_in[]` is dependent on the parameter type. This signal is sampled on the rising edge of clock and should only be asserted for one clock cycle to prevent the parameter from being re-read on subsequent clock cycles. Once `write_param` is sampled as a logic high, the busy signal is asserted. While the parameter is being written, the busy signal remains asserted, and inputs on `param[]` and `data_in[]` are ignored. Once the busy signal is deactivated, the next parameter can be written. This signal is only valid when the `Current_Configuration` parameter is factory since parameters cannot be written in application configurations. If this port is left unconnected, the default value is 0.
param[2..0]	N	Logic Array	3-bit bus that selects which parameter should be read or written. If this port is left unconnected, the default value is 0.
data_in[11..0]	N	Logic Array	This signal is required if you intend on writing parameters to the remote update block 12-bit bus used when writing parameters, which specifies the parameter value. The parameter value is requested using the `param[]` input and by driving the `write_param` signal logic high, at which point the busy signal goes logic high and the value of the parameter is captured from this bus. For some parameters, not all 12-bits will be used in which case only the least significant bits will be used. This port is ignored if the `Current_Configuration` parameter is set to an application configuration since writing of parameters is only allowed in the factory configuration. If this port is left unconnected, the default values is 0.

Note to Table 2–6:

(1) Logic array source means that you can drive the port from internal logic or any general-purpose I/O pin.

Table 2–7. Output Ports of the altremote_update Megafunction

Port Name	Required	Destination	Description
busy	Y	Logic Array	When this signal is a logic high, the remote update block is busy either reading or writing a parameter. When the remote update block is busy, it ignores its `data_in[]`, `param[]`, and `reconfig` inputs. This signal will go high when `read_param` or `write_param` is asserted and will remain asserted until the operation is complete.
pgm_out[2..0]	Y	PGM[2..0] pins	3-bit bus that specifies the page pointer of the configuration data to be loaded when the device is reconfigured. This port must be connected to the `PGM[]` output pins, which should be connected to the external configuration device
data_out[11..0]	N	Logic Array	12-bit bus used when reading parameters, which reads out the parameter value. The parameter value is requested using the `param[]` input and by driving the `read_param` signal logic high, at which point the busy signal will go logic high. When the busy signal goes low, the value of the parameter will be driven out on this bus. The `data_out[]` port is only valid after a read_param has been issued and once the busy signal is de-asserted. At any other time, its output values are invalid. For example, even though the `data_out[]` port may toggle during a writing of a parameter, these values are not a valid representation of what was actually written to the remote update block. For some parameters, not all 12-bits will be used in which case only the least significant bits will be used.

Note to Table 2–7:
(1) Logic array destination means that you can drive the port to internal logic or any general-purpose I/O pin.

Table 2–8. Parameter Settings for the altremote_update Megafunction (Part 1 of 2)

Selected Parameter	param[2..0] bit setting	width of parameter value	POR Reset Value	Description
Status Register Contents	000	5	5 bit '0	Specifies the reason for re-configuration, which could be caused by a CRC error during configuration, `nSTATUS` being pulled low due to an error, the device core caused an error, `nCONFIG` pulled low, or the watchdog timer timed-out. This parameter can only be read.
Watchdog Timeout Value	010	12	12 bits '0	User watchdog timer time-out value. Writing of this parameter is only allowed when in the factory configuration.
Watchdog Enable	011	1	1 bit '0	User watchdog timer enable. Writing of this parameter is only allowed when in the factory configuration

Table 2–8. Parameter Settings for the altremote_update Megafunction (Part 2 of 2)

Selected Parameter	param[2..0] bit setting	width of parameter value	POR Reset Value	Description
Page select	100	3	3 bit '001' - Local configuration 3 bit '000' - Remote configuration	Page mode selection. Writing of this parameter is only allowed when in the factory configuration.
Current configuration (AnF)	101	1	1 bit '0' - Factory 1 bit '1' - Application	Specifies whether the current configuration is factory or and application configuration. This parameter can only be read.
Illegal values	001			
	110			
	111			

Remote Update WYSIWYG ATOM

An alternative to using the `altremote_update` megafunction is to directly instantiate the remote update WYSIWYG atom. This atom should be included in the factory configuration and any application configuration image to access the remote configuration shift registers.

When implementing the atom, you should consider following:

1. Only one atom can be used in the circuit; more than one gives a no-fit.
2. All signals for the cell must be connected. The clock port (`CLK`) must be connected to a live cell. The others can be constant V_{CC} or GND.
3. The `pgmout` port must be connected and must feed `PGM[2.0]` output pins (it cannot be connected to anything else but output pins).
4. The Quartus II software reserves `RUnLU` as an input pin, and you must connect it to V_{CC}.

The Stratix and Stratix GX remote update atom ports are:

Stratix_rublock <*rublock_name*>
(
.clk(<*clock source*>**),**
.shiftnld(<*shiftnld source*>**),**
.captnupdt(<*shiftnld source*>**),**
.regin(<*regin input source from the core*>**),**
.rsttimer(<*input signal to reset the watchdog timer*>**),**
.config(<*input signal to initiate configuration*>**),**
.regout(<*data output destination to core*>**),**
.pgmout(<*program output destinations to pins*>**)**

Table 2–9 shows the remote update block input and output port names and descriptions.

Table 2–9. Remote Update Block Input & Output Ports

Ports	Definition
<*rublock_name*>	The unique identifier for the instance. This identifier name can be anything as long as it is legal for the given description language (i.e., Verilog, VHDL, AHDL, etc.). This field is required.
.clk(<*clock source*>)	Designates the clock input of this cell. All operation is with respect to the rising edge of this clock. This field is required.
.shiftnld(<*shiftnld source*>)	An input into the remote configuration block. When **.shiftnld** = 1, the data shifts from the internal shift registers to the `regout` port at each rising edge of `clk`, and the data also shifts into the internal shift registers from `regin` port. This field is required.
.captnupdt(<*shiftnld source*>)	An input into the remote configuration block. This controls the protocol of when to read the configuration mode or when to write into the registers that control the configuration. This field is required.
.regin(<*regin input source from the core*>)	An input into the configuration block for all data loading into the core. The data shifts into the internal registers at the rising edge of `clk`. This field is required.
.rsttimer(<*input signal to reset the watchdog timer*>)	An input into the watchdog timer of the remote update block. When this is high, it resets the watchdog timer. This field is required.
.config(<*input signal to initiate configuration*>)	An input into the configuration section of the remote update block. When this signal goes high, the part initiates a re-configuration. This field is required.
.regout(<*data output destination to core*>)	A 1-bit output, which is the output of the internal shift register, and updated every rising edge of `clk`. The data coming out depends on the control signals. This field is required.
.pgmout(<*program output destinations to pins*>)	A 3-bit bus. It should always be connected only to output pins (not `bidir` pins). This bus gives the page address (`000` to `111`) of the configuration data to be loaded when the device is getting configured. This field is required.

For more information on the control signals for the remote block, see Table 2–3 on page 2–9.

Using Enhanced Configuration Devices

This section describes remote system configuration of Stratix and Stratix GX devices with the Nios embedded processor using enhanced configuration devices. Enhanced configuration devices are composed of a standard flash memory and a controller. The flash memory stores configuration data, and the controller reads and writes to the flash memory.

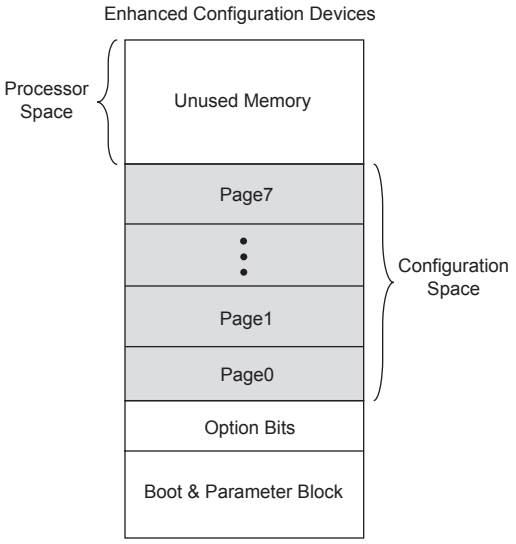
In remote system configuration, only PS and FPP modes are supported using an enhanced configuration device. A Stratix or Stratix GX device running a Nios embedded processor can receive data from a remote source through a network or any other appropriate media. A specific page of the enhanced configuration device stores the received data.

This scheme uses the page mode option in Stratix and Strati GX devices. Up to eight pages can be stored in each enhanced configuration device, each of which can store a configuration file.

In enhanced configuration devices, a page is a section of the flash memory space. Its boundary is determined by the Quartus II software (the page size is programmable). In the software, you can specify which configuration file should be stored in which page within the flash memory. To access the configuration file on each page, set the three input pins (`PGM[2..0]`), which provide access to all eight pages. Because the `PGM[2..0]` pins of an enhanced configuration device connect to the same pins of the Stratix or Stratix GX device, the Stratix or Stratix GX device selects one of the eight memory pages as a target location to read from. Figure 2–11 shows the allocation of different pages in the enhanced configuration device.

For more information on enhanced configuration devices, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Altera Enhanced Configuration Devices* chapter.

Figure 2–11. Memory Map in Enhanced Configuration Device

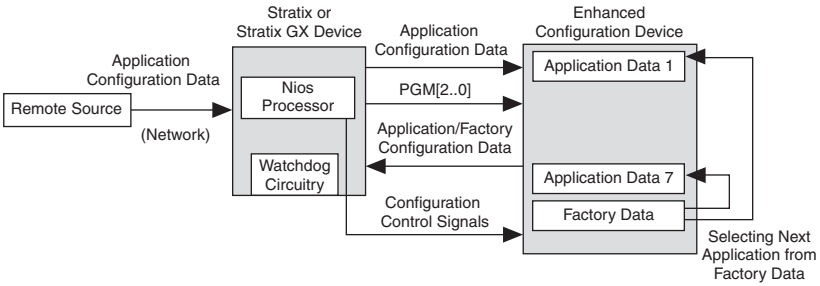


When the Stratix or Stratix GX device powers-up in remote configuration mode, the devices loads configuration data located at page address 000. You should always load the factory default configuration data at this location and make sure this information is not altered.

The factory configuration contains information to determine the next application configuration to load into the Stratix or Stratix GX device. When the Stratix or Stratix GX device successfully loads the application configuration from the page selected by the PGM[2..0] pins, it enters user mode.

In user mode, the Nios embedded processor (or any other logic) assists the Stratix or Stratix GX device in detecting remote system configuration information. In remote system configuration, the Nios embedded processor receives the incoming data from the remote source via the network, writes it to the ECP16 enhanced configuration device, and then initiates loading of the factory configuration into the Stratix or Stratix GX device. Factory configuration reads the remote configuration status register and determines the appropriate application configuration to load into the Stratix or Stratix GX device. Figure 2–12 shows the remote system configuration.

Figure 2–12. Remote System Configuration Using Enhanced Configuration Devices



The user watchdog timer in Stratix and Stratix GX devices ensures that an application configuration has loaded successfully and checks if the application configuration is operating correctly in user mode. The watchdog timer must be continually reset by the user logic. If an error occurs while the application configuration loads, or if the watchdog timer times-out during user mode, the factory configuration is reloaded to prevent the system from halting in an erroneous state. Figure 2–3 on page 2–4 illustrates the remote configuration mode.

Upon power-up in local configuration scheme, the application configuration at page 001 (`PGM[001]` of the enhanced configuration device) loads into the Stratix or Stratix GX device. This application can be remotely or locally updated. If an error occurs during loading of the configuration data, the factory configuration loads automatically (see Figure 2–4 on page 2–5). The rest is identical to remote configuration mode.

Local Update Programming File Generation

This section describes the programming file generation process for performing remote system upgrades. The Quartus II convert programming files (CPF) utility generates the initial and partial programming files for configuration memory within the enhanced configuration devices.

The two pages that local configuration mode uses are a factory configuration stored at page 000, and an application configuration stored at page 001. The factory configuration cannot be updated after initial production programming. However, the application configuration can be erased and reprogrammed after initial system deployment.

In local update mode, you would first create the initial programming file with the factory configuration image and a version of the application configuration. Subsequently, you can generate partial programming files to update the application configuration (stored in page 001). Quartus II CPF can create partial programming files in .**hex** (Hexadecimal file), JAM, **.jbc** (JAM Byte-Code File), and POF formats.

In addition to the two configuration pages, user data or processor code can also be pre-programmed in the bottom boot and main data areas of the enhanced configuration device memory. The CPF utility accepts a HEX input file for the bottom and main data areas, and includes this data in the POF output file. However, this is only supported for initial programming file generation. Partial programming file generation for updating user HEX data is not supported, but can be performed using the enhanced configuration device external flash interface.

Initial Programming File Generation

The initial programming file includes configuration data for both factory and application configuration pages. The enhanced configuration device option's bits are always located between byte addresses `0x00010000` and `0x0001003F`. Also, page 0 always starts at `0x00010040` while its end address is dependent on the size of the factory configuration data.

The two memory allocation options that exist for the application configuration are auto addressing and block addressing. In auto addressing mode, Quartus II automatically allocates memory for the application configuration. All the configuration memory sectors that are not used by the page 0 factory configuration are allocated for page 1. The memory allocated is maximized to allow future versions of the application configuration to grow and have bigger configuration files (when the compression feature is enabled). Processor or user data storage (HEX input file) is only supported by the bottom boot area in auto addressing mode.

The following steps and screen shot (see Figure 2–13) describe initial programming file generation with auto addressing mode.

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File (*.pof)** from the drop-down list titled **Programming File Type**.

3. Select the enhanced configuration device used (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration Device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.

4. In the **Input files to convert** box, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.

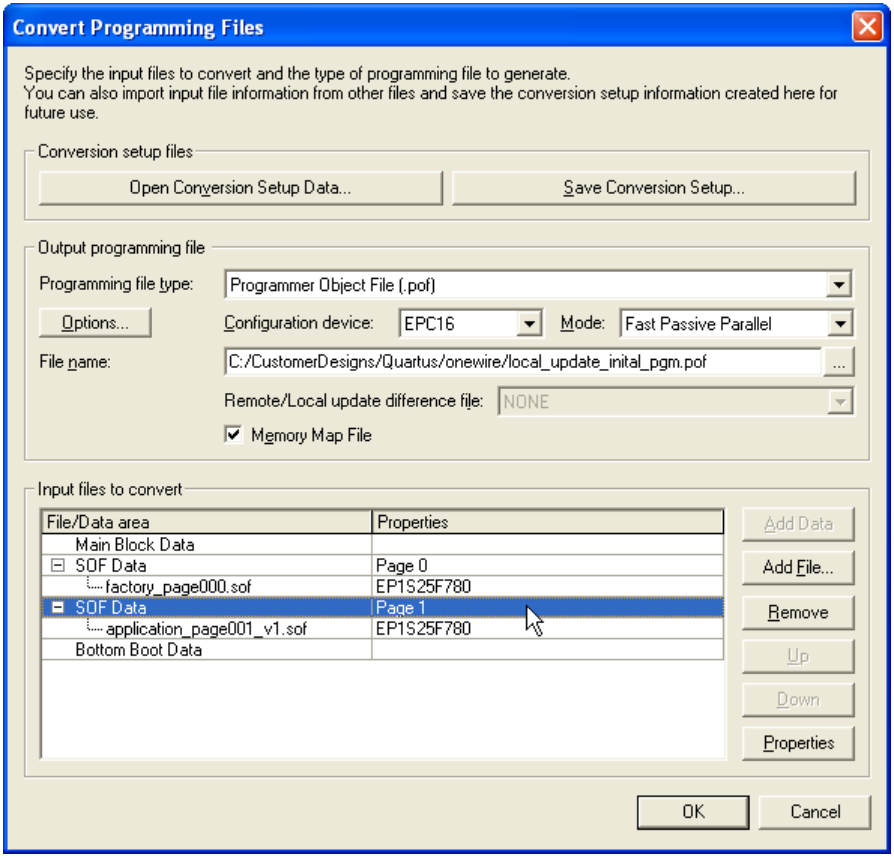
5. Repeat Step 4 for the **Page 1** application configuration page.

6. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

7. Save the CPF setup (optionally), by selecting **Save Conversion Setup**… and specifying a name for the **.cof** output file.

8. Click **OK** to generate initial programming and memory map files.

Figure 2–13. CPF Setup for Initial Programming File (Auto Addressing)



A sample memory map output file for the preceding setup is shown below. Configuration option bits and page 0 data occupy main flash sectors 0 through 4. See the *Sharp LHF16J06 Flash memory used in EPC16 devices* Data Sheet at **www.altera.com** to correlate memory addresses to the EPC16 flash sectors. In auto addressing mode, page 1 allocates all unused flash sectors. For this example, this unused area includes main sectors 5 through 30, and all of the bottom boot sectors. While this large portion of memory is allocated for page 1, the real application configuration data is top justified within this region with filler 1'b1 bits in lower memory addresses. Notice that the page 1 configuration data

wraps around the top of the memory and fills up the bottom boot area. The wrap around does not occur if the bottom boot area is used for processor/user HEX data file storage.

Block	Start Address	End Address
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054CC8
PAGE 1	0x001CB372	0x0000FFFD wrapped around

The block addressing mode allows better control of flash memory allocation. You can allocate a specific flash memory region for each application configuration page. This allocation is done by specifying a block starting and block ending address. While selecting the size of the region, you should account for growth in compressed configuration bitstream sizes due to design changes and additions. In local update mode, all configuration data is top justified within this allotted memory. In other words, the last byte of configuration data is stored such that it coincides with the highest byte address location within the allotted space. Lower unused memory address locations within the allotted region are filled with 1's. These filler bits are transmitted during a configuration cycle using page 1, but are ignored by the Stratix device. The memory map output file provides the exact byte address where real configuration data for page 1 begins. Note that any partial update of page 1 should erase all allotted flash sectors before storing new configuration data.

In the block addressing mode, HEX input files can be optionally added to the bottom boot and main flash data areas (one HEX file per area is allowed). The HEX file can be stored with relative addressing or absolute addressing. For more information on relative and absolute addressing, see the *Using Altera Enhanced Configuration Devices* chapter of the *Configuration Handbook*.

Figures 2–14 and 2–15, and the following steps illustrate generating an initial programming file with block addressing for local update mode. This example also illustrates preloading user HEX data into bottom boot and main flash sectors.

1. Open the **Convert Programming Files** window from the **File** menu.

2. Select **Programmer Object File (.pof)** from the drop-down list titled **Programming file type**.

3. Select the enhanced configuration device (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.

4. In the **Input files to convert** box, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.

5. Repeat Step 4 for the Page 1 application configuration page.

6. For enabling block addressing, select the **SOF Data** entry for **Page 1**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–15).

7. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. Note that for partial programming support, the block start and end addresses should be aligned to a flash sector boundary. This prevents two configuration pages from overlapping within the same flash boundary. See the flash memory datasheet for data sector boundary information. Click **OK** to save SOF data properties.

8. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

9. Save the CPF setup (optionally), by selecting **Save Conversion Setup**… and specifying a name for the COF output file.

10. Click **OK** to generate initial programming and memory map files.

Figure 2–14. CPF Setup for Initial Programming File Generation (Block Addressing)

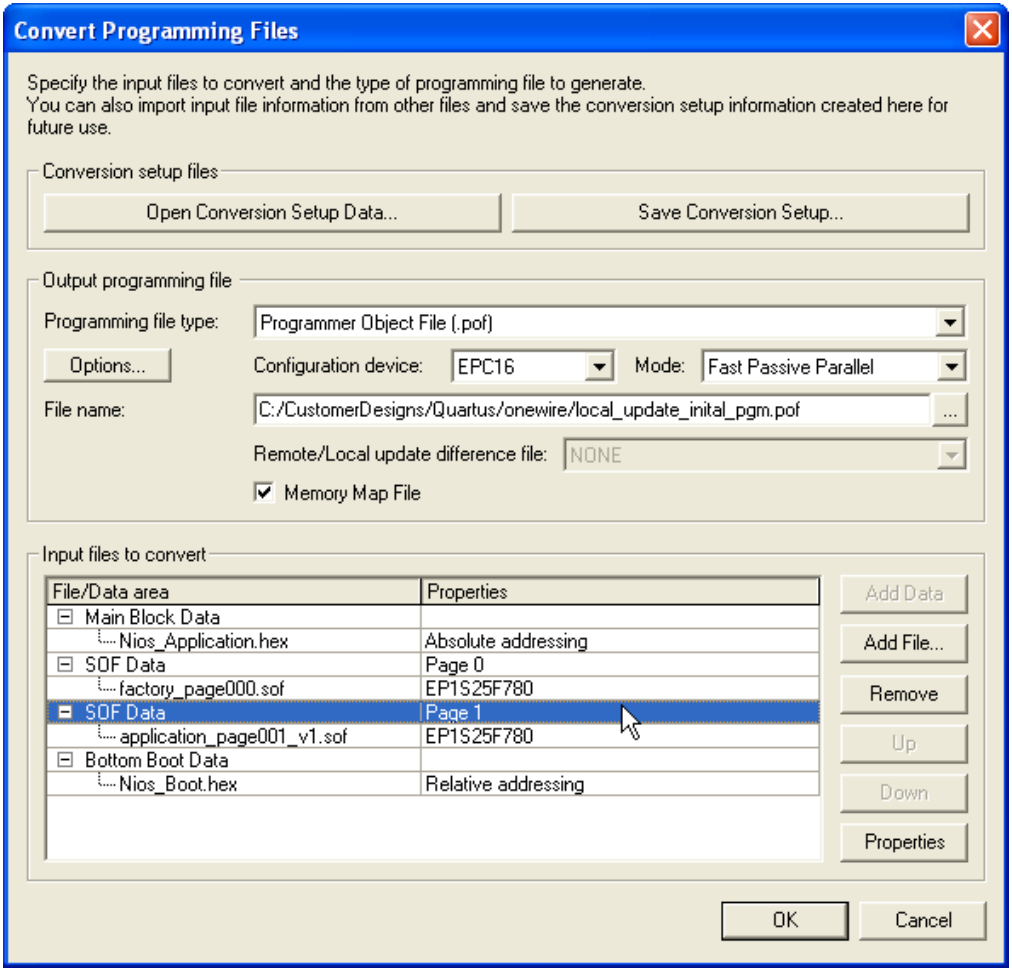
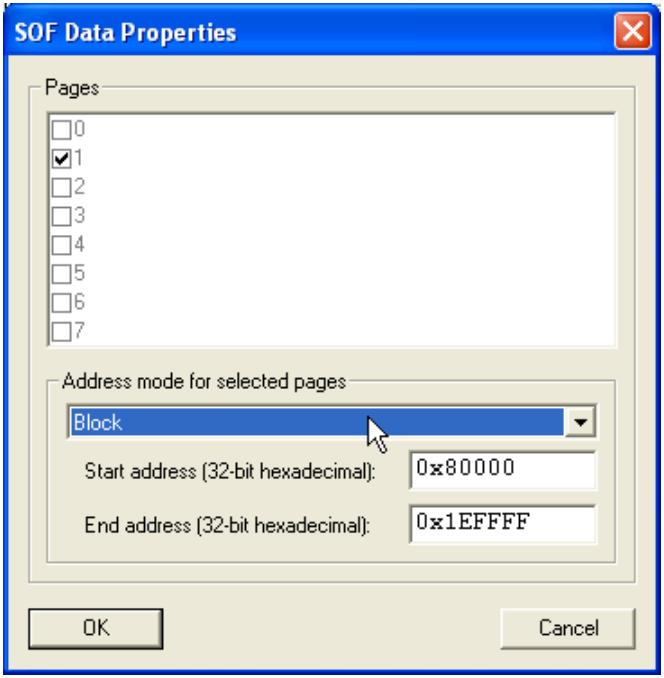


Figure 2–15. Specifying Block Addresses for Application Configuration



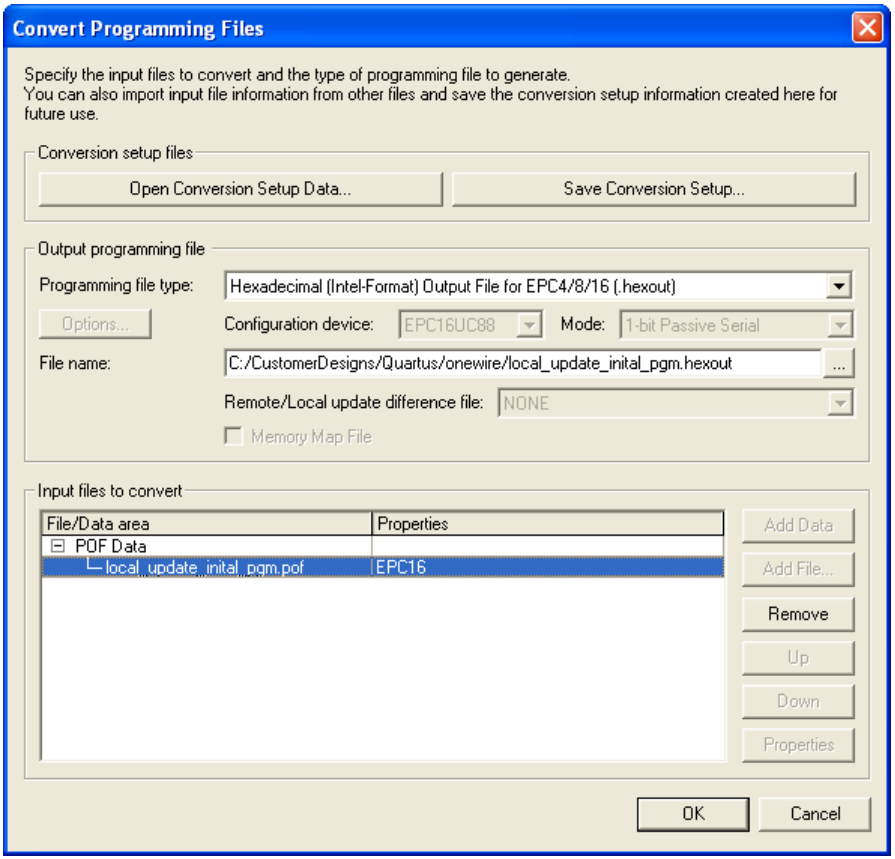
A sample memory map output file for the preceding example is shown below. Note that the allocated memory for page 1 is between 0x00080000 and 0x001EFFFF, while the actual region used by the current application configuration bitstream is between 0x001AB36C and 0x001EFFF7. The configuration data is top justified within the allocated SOF data region.

Block	Start Address	End Address
BOTTOM BOOT	0x00000000	0x000001FF
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054CC8
PAGE 1	0x001AB36C	0x001EFFF7
TOP BOOT/MAIN	0x001F0000	0x001F01FF

Also note that the HEX data stored in the main data area uses absolute addressing. If relative addressing were to be used, the main data contents would be justified with the top (higher address locations) of the memory.

The initial programming file (POF) can be converted to an Intel Hexadecimal format file (*.HEXOUT) using the Quartus II CPF utility. See Figure 2–16.

Figure 2–16. Converting POF Programming File to Intel HEX Format



Partial Programming File Generation

The enhanced Quartus II CPF utility allows an existing application configuration page to be replaced with new data. Partial programming files are generated to perform such configuration data updates.

In order to generate a partial programming file, you have to input the initial programming file (POF) and new configuration data (SOF) to the Quartus II CPF utility. In addition, you have to specify the addressing mode (auto or manual) that was used during initial POF creation. And if

block addressing was used, you should specify the block start and end addresses. With this information, Quartus II ensures that the partial programming file only updates the flash region containing the application configuration. The factory configuration (page 0) and configuration option bits are left unaltered during this process.

Figure 2–17 and the following steps illustrate generation of a partial programming file:

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File for Local Update (.pof)** from the drop-down list titled **Programming file type**, and specify an output **File name**.
3. In the **Input files to convert box**, highlight **POF Data** and click **Add File**. Select the initial programming POF file for this design and insert it.
4. In the **Input files to convert box**, highlight **SOF Data** and click **Add File.** Select the new application configuration bitstream (SOF) and insert it.
5. When using block addressing, select the **SOF Data** entry for **Page 1**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–18).
6. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. These addresses should be identical to those used to generate the initial programming file. Click **OK** to save SOF data properties.
7. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of the new application configuration data in page 1.
8. Pick a local update difference file from the **Remote/Local Update Difference File** drop-down menu. You can select between an Intel HEX, JAM, JBC, and POF output file types. The output file name is the same as the POF output file name with a _dif suffix.
9. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.
10. Click **OK** to generate initial programming and memory map files.

Figure 2–17. Local Update Partial Programming File Generation

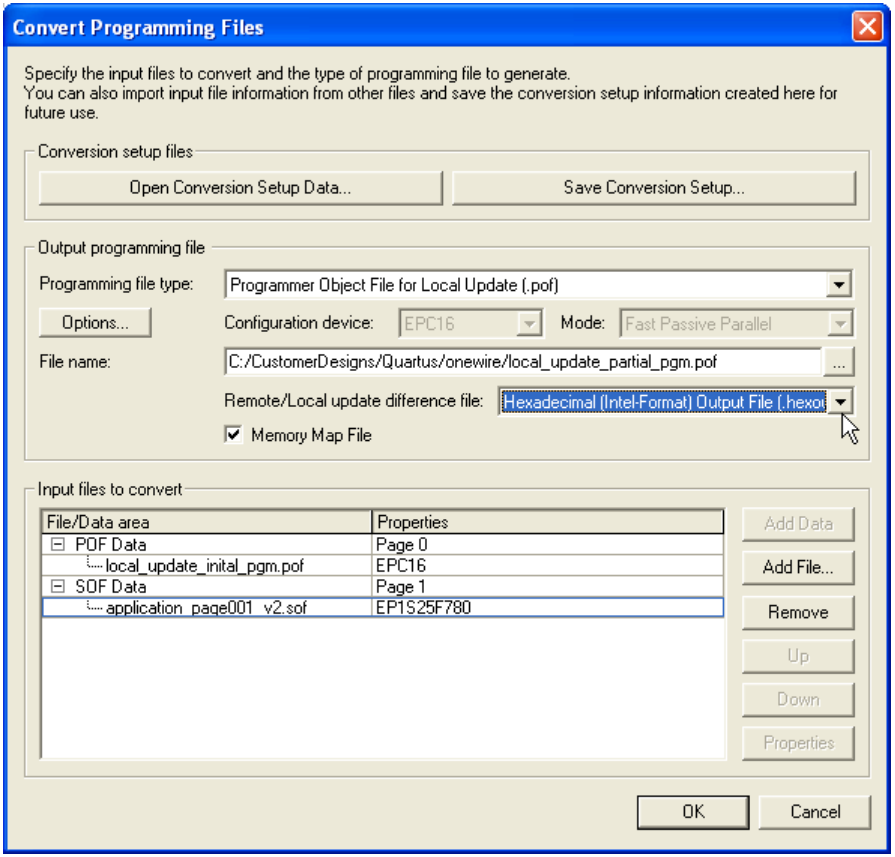
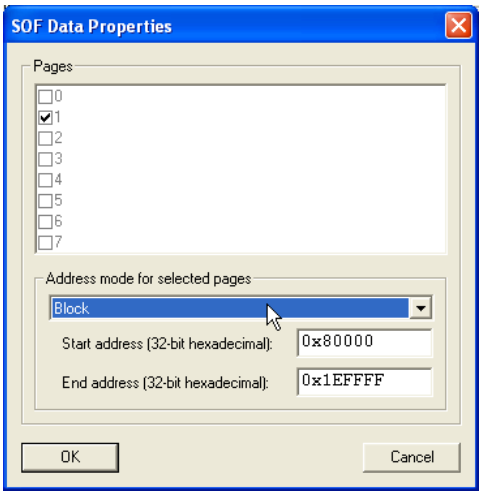


Figure 2–18. Specifying Block Addresses for Application Configuration



Remote Update Programming File Generation

This section describes the programming file generation process for performing remote system upgrades. The Quartus II CPF utility generates the initial and partial programming files for configuration memory within the enhanced configuration devices.

Remote configuration mode uses a factory configuration stored at page 0, and up to seven application configurations stored at pages 1 through 7. The factory configuration cannot be updated after initial production programming. However, the most recent application configuration can be erased and reprogrammed after initial system deployment. Alternatively, a new application configuration can be added provided adequate configuration memory availability.

In remote update mode, you would first create the initial programming file with the factory configuration image and the application configuration(s). Subsequently, you can generate partial programming files to update the most recent application configuration or add a new application configuration. Quartus II CPF can create partial programming files in HEX, JAM, JBC, and POF formats.

In addition to the configuration pages, user data or processor code can also be pre-programmed in the bottom boot and main data areas of the enhanced configuration device memory. The CPF utility accepts a HEX input file for the bottom and main data areas, and includes this data in the

POF output file. However, this is only supported for initial programming file generation. Partial programming file generation for updating user HEX data is not supported, but can be performed using the enhanced configuration device external flash interface.

Initial Programming File Generation

The initial programming file includes configuration data for both factory and application configuration pages. The enhanced configuration device option's bits are always located between byte addresses `0x00010000` and `0x0001003F`. Also, page 0 always starts at `0x00010040` while its end address is dependent on the size of the factory configuration data.

Two memory allocation options exist for application configurations: auto addressing and block addressing. In auto addressing mode, Quartus II packs all application configurations as close together as possible. This maximizes the number of application configurations that can be stored in memory. However, when auto addressing is used you cannot update existing application configurations. Only new application configurations can be added to the memory.

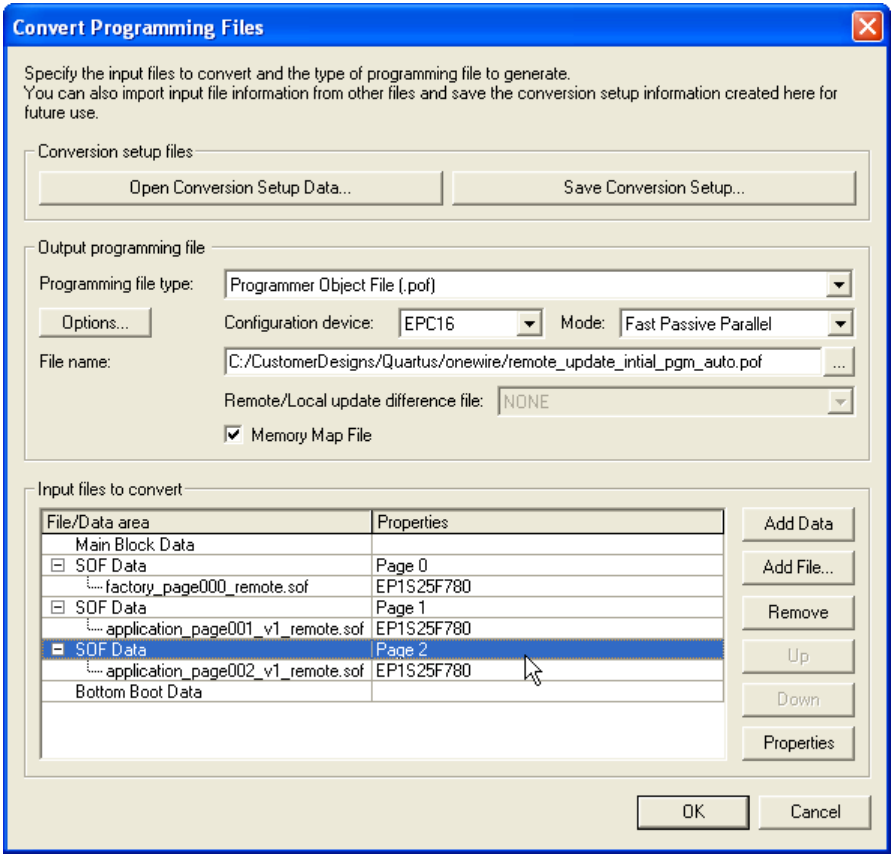
The following steps and screen shot (see Figure 2–19) describe initial programming file generation with auto addressing mode.

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File (*.pof)** from the drop-down list titled **Programming file type**.
3. Select the enhanced configuration device used (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.
4. In the **Input files to convert box**, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.
5. Repeat Step 4 for all application configurations (up to 7 maximum).
6. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

7. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.

8. Click **OK** to generate initial programming and memory map files.

Figure 2–19. CPF Setup for Initial Programming File Generation (Auto Addressing)



A sample memory map output file for the preceding setup is shown below. Notice all configuration pages are packed such that two pages can share a flash data sector. This disallows partial programming of application configurations in auto addressing mode.

Block	Start Address	End Address
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054EFA
PAGE 1	0x00054EFC	0x00099DB6
PAGE 2	0x00099DB8	0x000DEC72

See the *Sharp LHF16J06 Data Sheet Flash memory used in EPC16 devices* at **www.altera.com** to correlate memory addresses to the EPC16 flash sectors.

The block addressing mode allows better control of flash memory allocation. You can allocate a specific flash memory region for each application configuration page. This allocation is done by specifying a block starting and block ending address. While selecting the size of the region, you should account for growth in compressed configuration bitstream sizes due to design changes and additions. In remote update mode, all configuration data is top justified within this allotted memory. In other words, the last byte of configuration data is stored such that it coincides with the highest byte address location within the allotted space. Lower unused memory address locations within the allotted region are filled with 1's. These filler bits are transmitted during the application configuration cycle, but are ignored by the Stratix device. The memory map output file provides the exact byte address where real application configuration data for each page begins. Note that any partial update of the most recent application configuration should erase all allotted flash sectors for that page before storing new configuration data.

In the block addressing mode, HEX input files can be optionally added to the bottom boot and main flash data areas (one HEX file per area is allowed). The HEX file can be stored with relative addressing or absolute addressing. For more information on relative and absolute addressing, see the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter of the *Configuration Handbook, Volume 2*.

Figures 2–20 and 2–21, and the following steps illustrate generating an initial programming file with block addressing for remote update mode. This example also illustrates preloading user HEX data into bottom boot and main flash sectors.

1. Open the **Convert Programming Files** window from the **File** menu.

2. Select **Programmer Object File (*.pof)** from the drop-down list titled **Programming file type**.

3. Select the enhanced configuration device used (EPC4, EPC8, EPC16), and the mode used (1-bit Passive Serial or Fast Passive Parallel). Only during the initial programming file generation can you specify the **Options**, **Configuration device**, or **Mode** settings. While generating the partial programming file, all of these settings are grayed out and inaccessible.

4. In the **Input** files to convert box, highlight **SOF Data** at **Page 0** and click **Add File**. Select input SOF file(s) for this configuration page and insert them.

5. Repeat Step 4 for all the application configuration pages (pages 1 and 2 in this example).

6. For enabling block addressing, select the **SOF Data** entry for **Page 1**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–21).

7. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. Note that for partial programming support, the block start and end addresses should be aligned to a flash sector boundary. This prevents two configuration pages from overlapping within the same flash boundary. See the flash memory datasheet for data sector boundary information. Click **OK** to save SOF data properties.

8. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of each configuration page and user data blocks.

9. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.

10. Click **OK** to generate initial programming and memory map files.

Figure 2–20. CPF Setup for Initial Programming File Generation (Block Addressing)

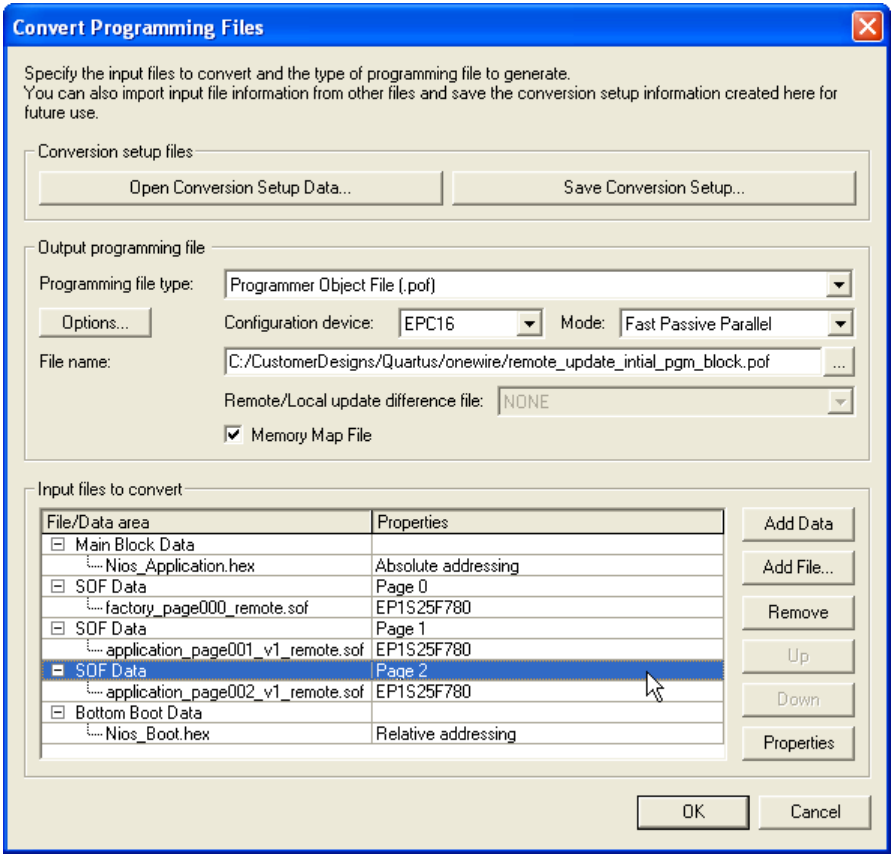
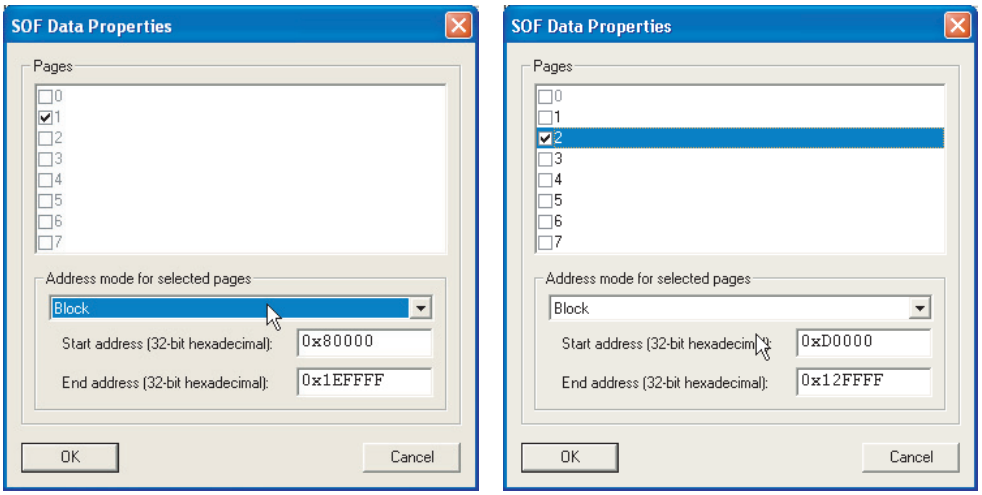


Figure 2–21. Specifying Block Addresses for an Application Configuration



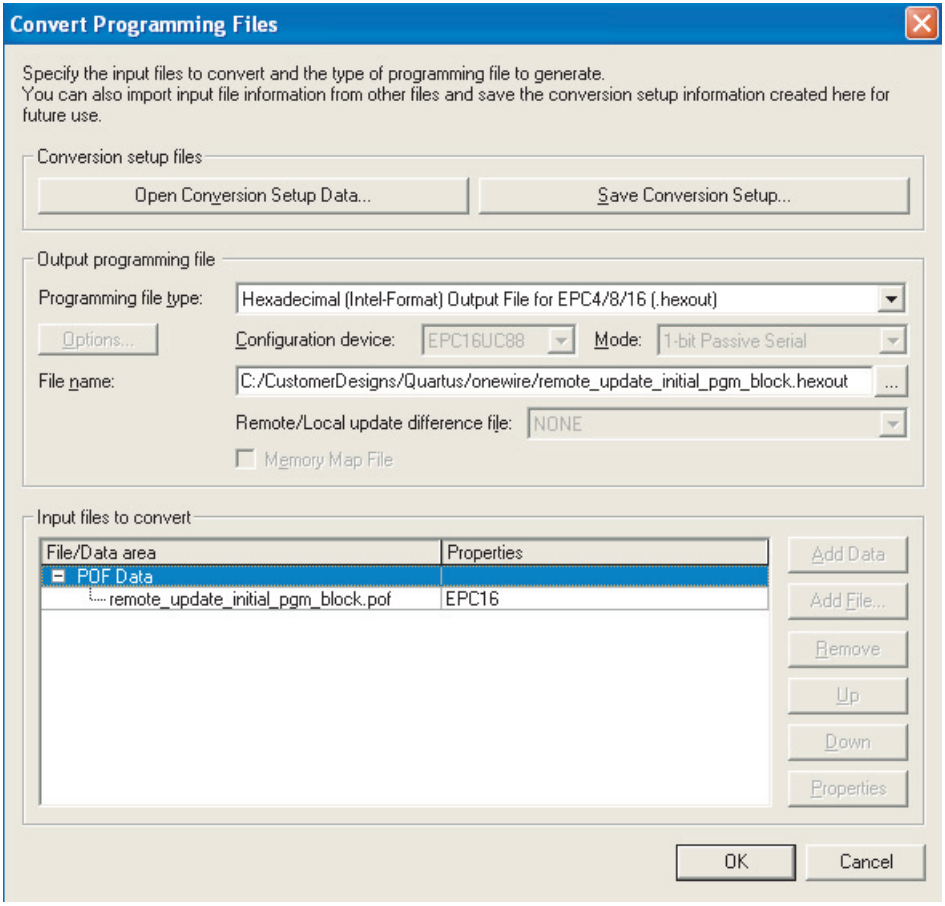
A sample memory map output file for the preceding example is shown below. Note that the allocated memory for page 1 is between `0x00070000` and `0x000BFFFF`, while the actual region used by the current application configuration bitstream is between `0x0007B144` and `0x000BFFFF`. The configuration data is top justified within the allocated SOF data region. Similarly, the allocated memory for page 2 is between `0x000D0000` and `0x0012FFFF`, while the actual region used by the application configuration is between `0x000EB13E` and `0x0012FFF9`.

Block	Start Address	End Address
BOTTOM BOOT	0x00000000	0x000001FF
OPTION BITS	0x00010000	0x0001003F
PAGE 0	0x00010040	0x00054EFA
PAGE 1	0x0007B144	0x000BFFFF
PAGE 2	0x000EB13E	0x0012FFF9
TOP BOOT/MAIN	0x001F0000	0x001F01FF

Also note that the HEX data stored in the main data area uses absolute addressing. If relative addressing were to be used, the main data contents would be justified with the top (higher address locations) of the memory.

The initial POF can be converted to an Intel Hexadecimal format file (*.HEXOUT) using the Quartus II CPF utility. See Figure 2–22.

Figure 2–22. Converting POF Programming File to Intel HEX Format



Partial Programming File Generation

In remote update mode, the Quartus II CPF utility allows an existing application configuration page to be replaced with new data, or a new application configuration to be added. Partial programming files are generated to perform such configuration data updates.

In order to generate a partial programming file, you have to input the initial POF and new configuration data (SOF) to the Quartus II CPF utility. In addition, you have to specify the addressing mode (auto or manual) that was used during initial POF creation. And if block addressing was used, you should specify the block start and end

addresses. With this information, Quartus II ensures that the partial POF only updates the flash region containing the application configuration. The factory configuration (page 0) and configuration option bits are left unaltered during this process. The only exception is when a new application configuration is added, the configuration options bits are updated to include start/end addresses for the new page. All existing page addresses and other configuration options bits remain unchanged.

Figure 2–23 and the following steps illustrate generation of a partial programming file to replace the most recent application configuration. In this example, the initial programming file contained one factory and two application configurations. Hence, the page 2 application configuration is being updated with new data.

1. Open the **Convert Programming Files** window from the **File** menu.
2. Select **Programmer Object File for Remote Update (*.pof)** from the drop-down list titled **Programming file type**, and specify an output file name.
3. In the **Input files to convert** box, highlight **POF Data** and click **Add File**. Select the initial programming POF file for this design and insert it.
4. In the **Input files to convert box**, highlight **SOF Data** and click **Add File**. Select the new application configuration bitstream (SOF) and insert it.
5. When using block addressing, select the **SOF Data** entry for **Page 2**, and click **Properties**. This opens the **SOF Data Properties** dialog box (see Figure 2–24 on page 2–42).
6. Pick **Block** from the **Address Mode** drop down selection, and enter 32-bit Hexadecimal byte address for block **Starting Address** and **Ending Address**. These addresses should be identical to those used to generate the initial programming file. Click **OK** to save SOF data properties.
7. Check the **Memory Map File** box to generate a memory map output file that specifies the start/end addresses of the new application configuration data in page 1.
8. Pick a remote update difference file from the **Remote/Local Update Difference File** drop-down menu. You can select between an Intel HEX, JAM, JBC, and POF output file types. The output file name is the same as the POF output file name with a _dif suffix.

9. Save the CPF setup (optionally), by selecting **Save Conversion Setup…** and specifying a name for the COF output file.

10. Click **OK** to generate initial programming and memory map files.

Figure 2–23. Remote Update Partial Programming File Generation

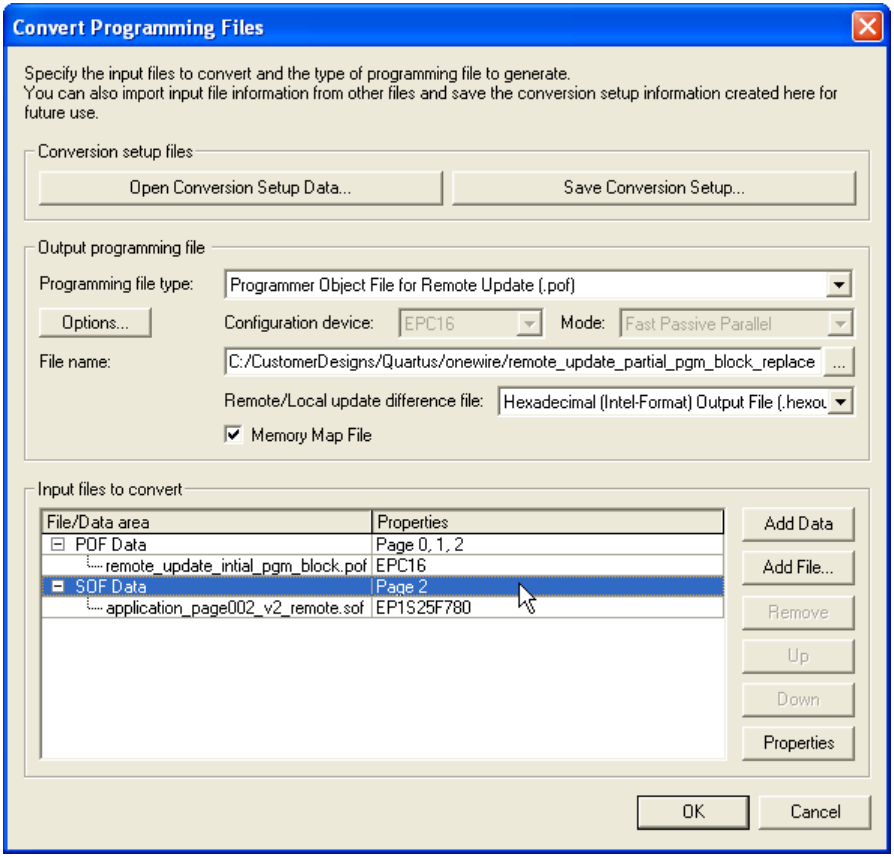
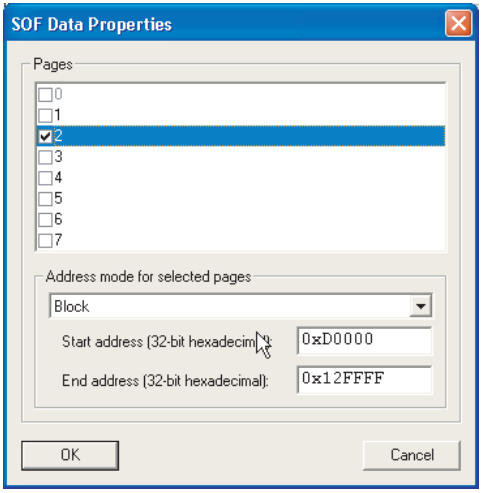


Figure 2–24. Specifying Block Addresses for Application Configuration



For adding a new application configuration, follow the steps listed above with one modification. In Step 5, select **SOF Data** and click on **Properties**. In the **SOF Data Properties** dialog box, select a new page (for example, page 3) and specify the addressing mode information. Continue with steps 7 through 10. When a new page is added, the memory map output file lists the start/end addresses for this page. A sample is shown below:

Block	Start Address	End Address
OPTION BITS	0x00010000	0x0001003F
PAGE 3	0x0012FFFA	0x00174EB4

Combining MAX Devices & Flash Memory

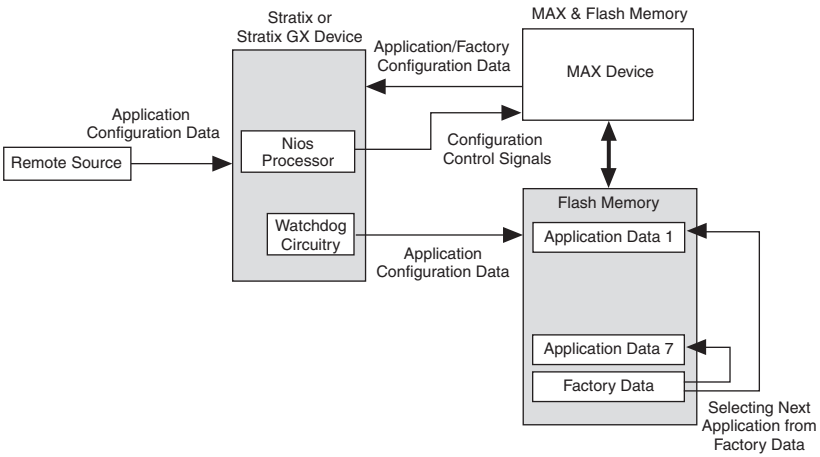
This section describes remote system configuration with the Stratix or Stratix GX device and the Nios embedded processor, using a combination of MAX® devices and flash memory.

You can use MAX 3000 or MAX 7000 devices and an industry-standard flash memory device instead of enhanced configuration devices. In this scheme, flash memory stores configuration data, and the MAX device controls reading and writing to the flash memory, keeping track of address locations.

The MAX device determines which address location and at what length to store configuration data in flash memory. The Nios embedded processor, running in the Stratix or Stratix GX device, receives the

incoming data from the remote source and writes it to the address location in flash memory. The Nios embedded processor initiates loading of factory configuration into the Stratix or Stratix GX device. Figure 2–25 shows remote system configuration using a MAX device and flash memory combination.

Figure 2–25. Remote System Configuration Using a MAX Device & Flash Memory



You can use both remote and local configuration modes in this scheme. You should specify a default page for factory configuration and make sure it is not altered or removed at any time. In remote system configuration mode, PS, FPP, and PPA modes are supported when configuring with MAX and flash devices.

Using an External Processor

This section describes remote system configuration with Stratix or Stratix GX devices and the Nios embedded processor, using an external processor and flash memory devices.

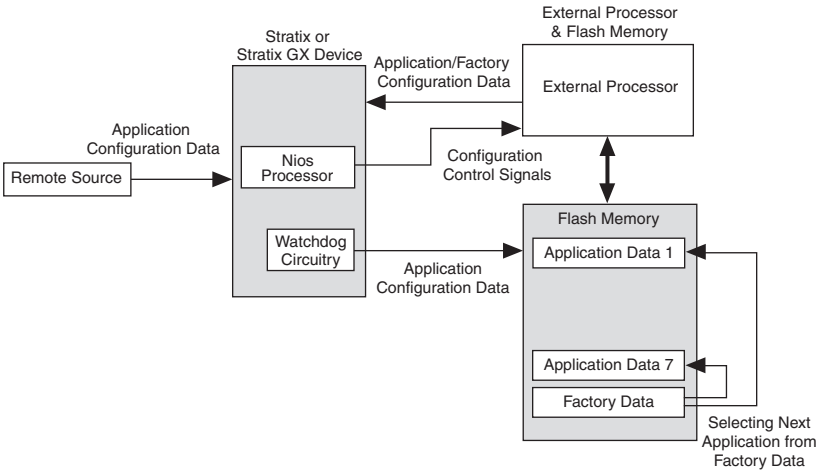
In this scheme, the external processor and flash memory device replace the enhanced configuration device. Flash memory stores configuration data, and the processor controls reading and writing to the flash memory and also keeps track of the address location. This type of remote system configuration supports PS, FPP, and PPA modes.

The processor determines at which address which length to store the configuration data in flash memory. The Nios embedded processor receives the incoming data from a remote source and writes it to the address location in the flash memory, and then initiates loading of factory

configuration data into the Stratix or Stratix GX device. Figure 2–26 shows the remote system configuration using a Nios embedded processor and flash memory.

You can use both remote and local configuration modes in this scheme. You should specify a default page for factory configuration and make sure it is not altered or removed at any time.

Figure 2–26. Remote System Configuration Using External Processor & Flash Memory



Conclusion

Stratix and Stratix GX devices are the first PLDs with dedicated support for remote system configuration. By allowing real-time system upgrades from a remote source, you can use Stratix and Stratix GX devices in a variety of applications that require automatic configuration updates. With the built-in watchdog timer circuitry, Stratix and Stratix GX devices avoid incorrect or erroneous states. Using Stratix and Stratix GX devices with remote system configuration enhances design flexibility and reduces time to market.

Section II. Design Guidelines

This section provides information on design transition, board design guidelines, and Stratix GX device path delay issues.

This section includes the following chapter:

- Chapter 3, Transitioning APEX Designs to Stratix & Stratix GX Devices
- Chapter 4, Stratix GX Board Design Guidelines
- Chapter 5, Quartus II Software Fitter Warnings

Revision History

The table below shows the revision history for Chapters 3 through 5.

Chapter(s)	Date / Version	Changes Made
3	February 2005 v3.0	Added chapter to *Stratix GX Device Handbook*.
4	February 2005 v1.0	Added chapter to *Stratix GX Device Handbook*.
5	March 2005 v1.0	Added chapter to *Stratix GX Device Handbook*.

3. Transitioning APEX Designs to Stratix & Stratix GX Devices

S52012-3.0

Introduction

Stratix® and Stratix GX devices are Altera's next-generation, system-on-a-programmable-chip (SOPC) solution. Stratix and Stratix GX devices simplify the block-based design methodology and bridge the gap between system bandwidth requirements and programmable logic performance.

This chapter highlights the new features in the Stratix and Stratix GX devices and provides assistance when transitioning designs from APEX™ II or APEX 20K devices to the Stratix or Stratix GX architecture. You should be familiar with the APEX II or APEX 20K architecture and available device features before using this chapter. Use this chapter in conjunction with the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

General Architecture

Stratix and Stratix GX devices offer many new features and architectural enhancements. Enhanced logic elements (LEs) and the MultiTrack™ interconnect structure offer reduced resource utilization and considerable design performance improvement. The MultiTrack interconnect uses DirectDrive™ technology to ensure the availability of deterministic routing resources for any design block, regardless of its placement within the device.

All architectural changes between Stratix and Stratix GX and APEX II or APEX 20K devices described in this section do not require any design changes. However, you must resynthesize your design and recompile in the Quartus® II software to target Stratix and Stratix GX devices.

Logic Elements

Stratix and Stratix GX device LEs include several new, advanced features that improve design performance and reduce logic resource consumption (see Table 3–1). The Quartus II software automatically uses these new LE features to improve device utilization.

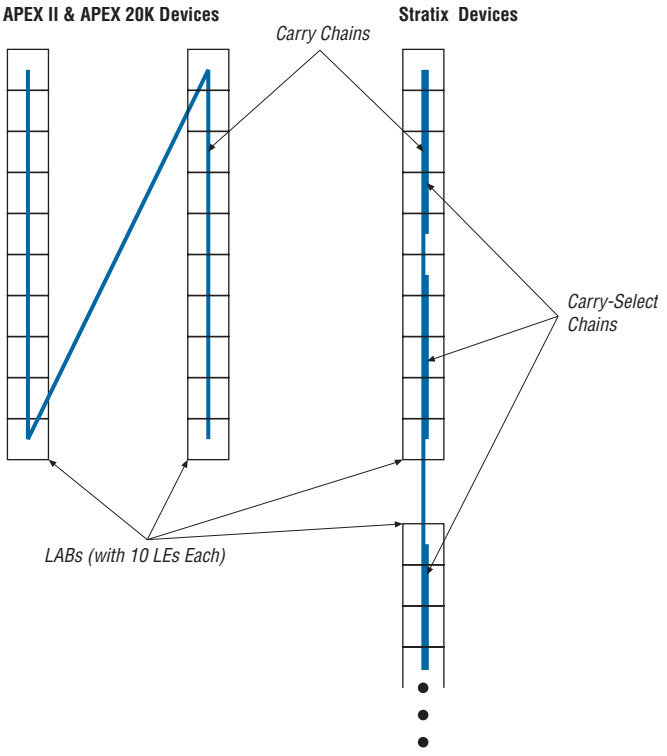
Table 3–1. Stratix & Stratix GX LE Features

Feature	Function	Benefit
Register chain interconnects	Direct path between the register output of an LE and the register input of an adjacent LE within the same logic array block (LAB)	■ Conserves LE resources ■ Provides fast shift register implementation ■ Saves local interconnect routing resources within an LAB
Look-up table (LUT) chain interconnects	Direct path between the combinatorial output of an LE and the fast LUT input of an adjacent LE within the same LAB	■ Allows LUTs within the same LAB to cascade together for high-speed wide fan-in functions, such as wide XOR operations ■ Bypasses local interconnect for faster performance
Register-to-LUT feedback path	Allows the register output to feed back into the LUT of the same LE, such that the register is packed with its own fan-out LUT	■ Enhanced register packing mode ■ Uses resources more efficiently
Dynamic arithmetic mode	Uses one set of LEs for implementing both an adder and subtractor	■ Improves performance for functions that switch between addition and subtraction frequently, such as correlators
Carry-select chain	Calculates outputs for a possible carry-in of 1 or 0 in parallel	■ Gives immediate access to result for both a carry-in of 1 or 0 ■ Increases speed of carry functions for high-speed operations, such as counters, adders, and comparators
Asynchronous clear and asynchronous preset function	Supports direct asynchronous clear and preset functions	■ Conserves LE resources ■ Does not require additional logic resources to implement NOT-gate push-back

In addition to the new LE features described in Table 3–1, there are enhancements to the chains that connect LEs together. Carry chains are implemented vertically in Stratix and Stratix GX devices, instead of horizontally as in APEX II and APEX 20K devices, and continue across rows, instead of across columns, as shown in Figure 3–1. Also note that the Stratix and Stratix GX architectures do not support the cascade primitive. Therefore, the Quartus II Compiler automatically converts

cascade primitives in APEX II and APEX 20K designs to a wire primitive when compiled for Stratix and Stratix GX devices. These architectural changes are transparent to the user and do not require design changes.

Figure 3–1. Carry Chain Implementation in APEX II & APEX 20K Devices vs. Stratix & Stratix GX Devices



MultiTrack Interconnect

Stratix and Stratix GX devices use the MultiTrack interconnect structure to provide a high-speed connection between logic resources using performance-optimized routing channels of different lengths. This feature maximizes overall design performance by placing critical paths on routing lines with greater speed, resulting in minimal propagation delay.

Stratix and Stratix GX device MultiTrack interconnect resources are described in Table 3–2.

Table 3–2. Stratix & Stratix GX Device MultiTrack Interconnect Resources

Routing Type	Interconnect	Span
Row	Direct link	Adjacent LABs and/or blocks
Row	R4	Four LAB units horizontally
Row	R8	Eight LAB units horizontally
Row	R24	Horizontal routing across the width of the device
Column	C4	Four LAB units vertically
Column	C8	Eight LAB units vertically
Column	C16	Vertical routing across the length of the device

Direct link routing saves row routing resources while providing fast communication paths between resource blocks. Direct link interconnects allow an LAB, digital signal processing (DSP) block, or TriMatrix™ memory block to drive data into the local interconnect of its left and right neighbors. LABs, DSP blocks, and TriMatrix memory blocks can also use direct link interconnects to drive data back into themselves for feedback.

The Quartus II software automatically uses these routing resources to enhance design performance.

For more information about LE architecture and the MultiTrack interconnect structure in Stratix and Stratix GX devices, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

DirectDrive Technology

When using APEX II or APE 20K devices, you must place critical paths in the same MegaLAB™ column to improve performance. Additionally, you should place critical paths in the same MegaLAB structure for optimal performance. However, this restriction does not exist in Stratix and Stratix GX devices because they do not contain MegaLAB structures. With the new DirectDrive™ technology in Stratix and Stratix GX devices, the actual distance between the source and destination of a path is the most important criteria for meeting timing performance. DirectDrive technology ensures that the same routing resources are available to each design block, regardless of its location in the device.

Architectural Element Names

The architectural element naming system within Stratix and Stratix GX devices differs from the row-column coordinate system (for example, LC1_A2, LAB_B1) used in previous Altera device families. Stratix and Stratix GX devices uses a new naming system based on the X-Y coordinate system, (X, Y). A number (N) designates the location within the block where the logic resides, such as LEs within an LAB. Because the Stratix and Stratix GX architectures are column-based, this naming simplifies location assignments. Stratix and Stratix GX architectural blocks include:

- LAB: logic array block
- DSP: digital signal processing block
- DSPOUT: adder/subtractor/accumulator or summation block of the DSP block
- M512: 512-bit memory block
- M4K: 4-Kbit memory block
- M-RAM: 512-Kbit memory block

Elements within architectural blocks include:

- LE: logic element
- IOC: I/O element
- PLL: phase-locked loop
- DSPMULT: DSP block multiplier
- SERDESTX: transmitter serializer/deserializer
- SERDESRX: receiver serializer/deserializer

Table 3–3 highlights the new location syntax used for Stratix and Stratix GX devices.

Table 3–3. Stratix & Stratix GX Location Assignment Syntax

Architectural Elements	Element Name	Location Syntax	Example of Location Syntax	
			Location	Description
Blocks	LAB, DSP, DSPOUT, M512, M4K, M-RAM	<*element_name*>_X<*number*>_Y<*number*>	LAB_X1_Y1	Designates the LAB in row 1, column 1
Logic	LE, IOC, PLL, DSPMULT, SERDESTX, SERDESRX	<*element_name*>_X<*number*>_Y<*number*>_N<*number*>	LC_X1_Y1_N0	Designates the first LE, N0, in the LAB located in row 1, column 1
Pins *(1)*	I/O pins	pin_<*pin_label*>	pin_5	Pin 5

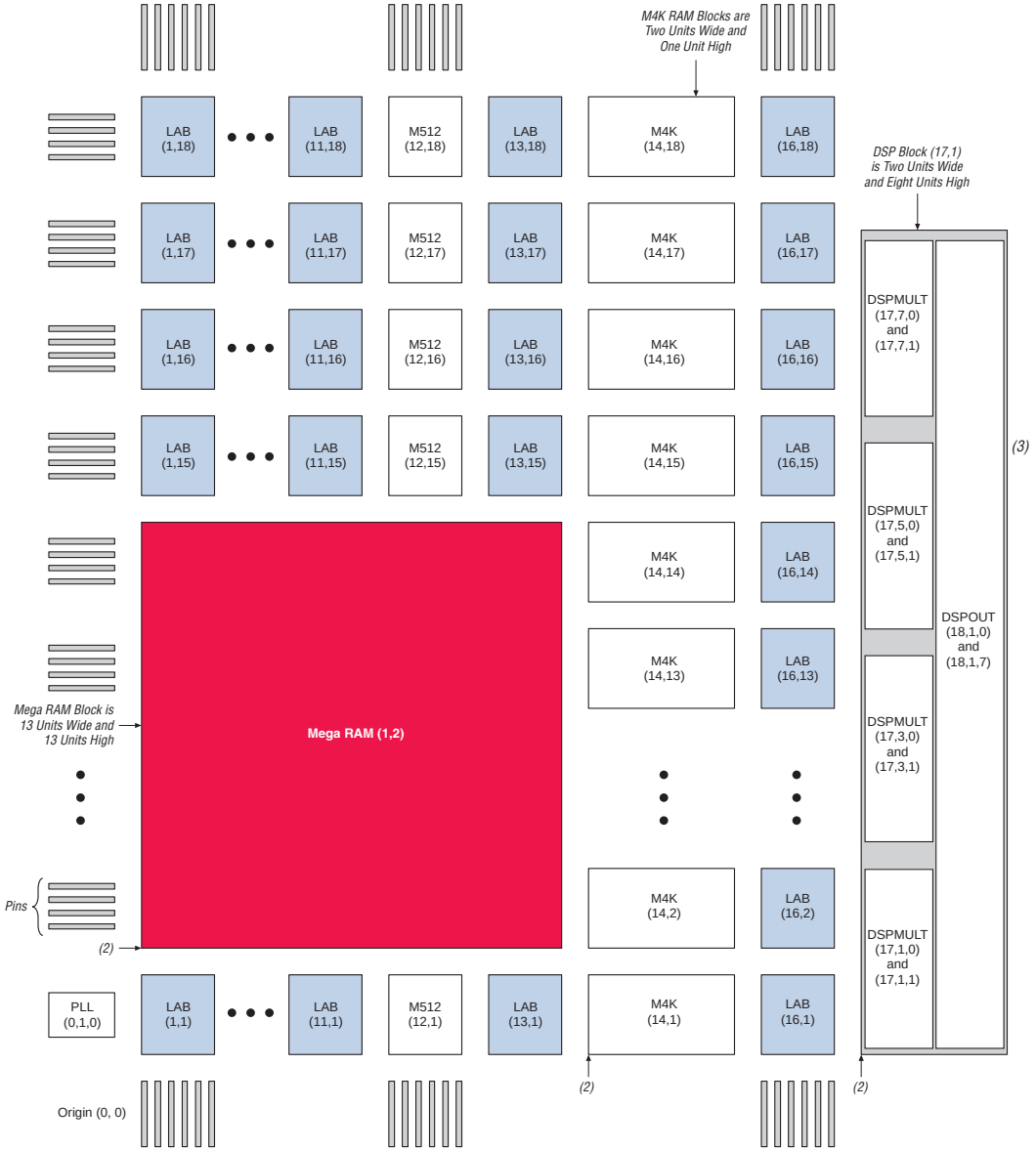
Note to Table 3–3:
(1) You can make assignments to I/O pads using IOC_X<*number*>_Y<*number*>_N<*number*>.

Use the following guidelines with the new naming system:

- The anchor point, or origin, in Stratix and Stratix GX devices is in the bottom-left corner, instead of the top-left corner as in APEX II and APEX 20K devices.
- The anchor point, or origin, of a large block element (e.g., a M-RAM or DSP block) is also the bottom-left corner.
- All numbers are zero-based, meaning the origin at the bottom-left of the device is X0, Y0.
- The I/O pins constitute the first and last rows and columns in the X-Y coordinates. Therefore, the bottom row of pins resides in X<*number*>, Y0, and the first left column of pins resides in X0, Y<*number*>.
- The sub-location of elements, N, numbering begins at the top. Therefore, the LEs in an LAB are still numbered from top to bottom, but start at zero.

Figure 3–2 show the Stratix and Stratix GX architectural element numbering convention. Figure 3–3 displays the floorplan view in the Quartus II software.

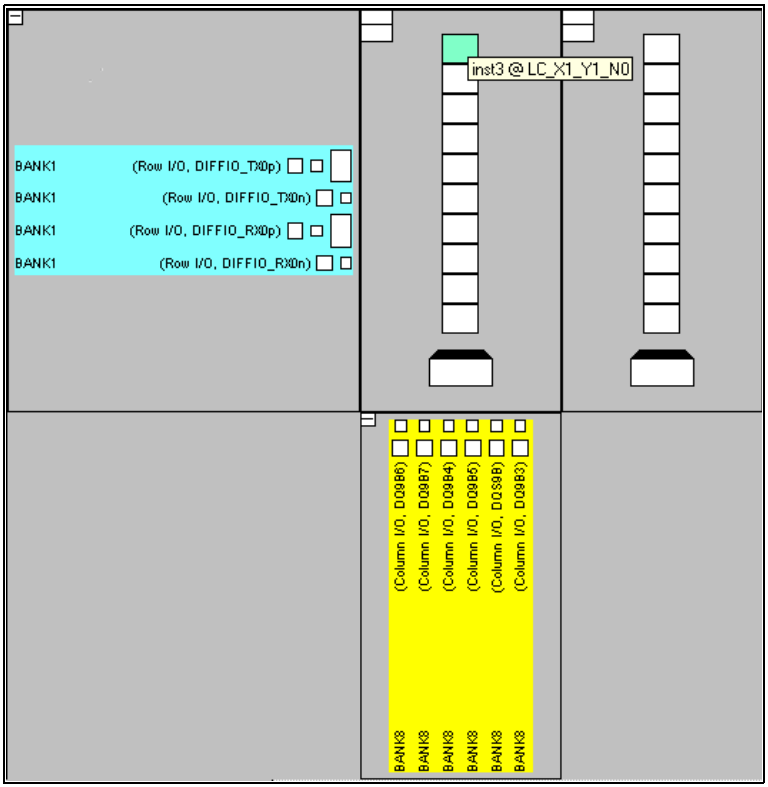
Figure 3–2. Stratix & Stratix GX Architectural Elements Note (1)



Notes to Figure 3–2:

(1) Figure 3–2 shows part of a Stratix and Stratix GX device.
(2) Large block elements use their lower-left corner for the coordinate location.
(3) The Stratix GX architectural elements include transceiver blocks on the right side of the device.

Figure 3–3. LE Numbering as Shown in the Quartus II Software



TriMatrix Memory

TriMatrix memory has three different sizes of memory blocks, each optimized for a different purpose or application. M512 blocks consist of 512 bits plus parity (576 bits), M4K blocks consist of 4K bits plus parity (4,608 bits), and M-RAM blocks consist of 512K bits plus parity (589,824 bits). This new structure differs from APEX II and APEX 20K devices, which feature uniformly sized embedded system blocks (ESBs) either 4 Kbits (APEX II devices) or 2 Kbits (APEX 20K devices) large. Stratix and Stratix GX TriMatrix memory blocks give you advanced control of each memory block, with features such as byte enables, parity bit storage, and shift-register mode, as well as mixed-port width support and true dual-port mode operation.

Table 3–4 compares TriMatrix memory with ESBs.

Table 3–4. Stratix & Stratix GX TriMatrix Memory Blocks vs. APEX II & APEX 20K ESBs

Features	Stratix & Stratix GX			APEX II ESB	APEX 20K ESB
	M512 RAM	M4K RAM	M-RAM		
Size (bits)	576	4,608	589,824	4,096	2,048
Parity bits	Yes	Yes	Yes	No	No
Byte enable	No	Yes	Yes	No	No
True dual-port mode	No	Yes Includes support for mixed width	Yes Includes support for mixed width	Yes Includes support for mixed width	No
Embedded shift register	Yes	Yes	No	No	No
Dedicated content-addressable memory (CAM) support	No	No	No	Yes	Yes
Pre-loadable initialization with a **.mif** *(1)*	Yes	Yes	No	Yes	Yes
Packed mode *(2)*	No	Yes	No	Yes	Yes
Feed-through behavior	Rising edge	Rising edge	Rising edge	Falling edge	Falling edge
Output power-up condition	Powers up cleared even if using a **.mif** *(1)*	Powers up cleared even if using a **.mif** *(1)*	Powers up with unknown state	Powers up cleared or to initialized value, if using a **.mif** *(1)*	Powers up cleared or to initialized value, if using a **.mif** *(1)*

Notes to Table 3–4:
(1) **.mif**: Memory Initialization File.
(2) Packed mode refers to combining two single-port RAM blocks into a single RAM block that is placed into true dual-port mode.

Stratix and Stratix GX TriMatrix memory blocks only support pipelined mode, while APEX II and APEX 20K ESBs support both pipelined and flow-through modes. Since all TriMatrix memory blocks can be pipelined, all input data and address lines are registered, while outputs can be either registered or combinatorial. You can use Stratix and Stratix GX memory block registers to implement input and output registers without utilizing additional resources. You can compile designs containing pipelined memory blocks (inputs registered) for Stratix and Stratix GX devices without any modifications. However, if an APEX II or

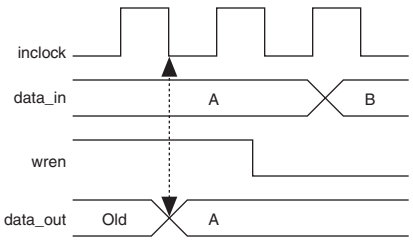
APEX 20K design contains flow-through memory, you must modify the memory modules to target the Stratix and Stratix GX architectures (see "Memory Megafunctions" on page 3–12 for more information).

For more information about TriMatrix memory and converting flow-through memory modules to pipelined, see the *TriMatrix Embedded Memory Blocks in Stratix & Stratix GX Devices* chapter in the *Stratix GX Device Handbook* and *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs*.

Same-Port Read-During-Write Mode

In same-port read-during-write mode, the RAM block can be in single-port, simple dual-port, or true dual-port mode. One port from the RAM block both reads and writes to the same address location using the same clock. When APEX II or APEX 20K devices perform a same-port read-during-write operation, the new data is available on the falling edge of the clock cycle on which it was written, as shown in Figure 3–4. When Stratix and Stratix GX devices perform a same-port read-during-write operation, the new data is available on the rising edge of the same clock cycle on which it was written, as shown in Figure 3–5. This holds true for all TriMatrix memory blocks.

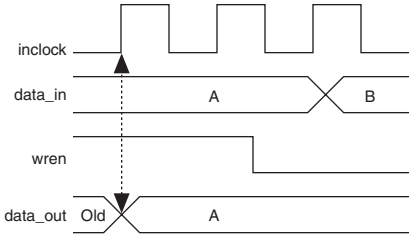
Figure 3–4. Falling Edge Feed-Through Behavior (APEX II & APEX 20K Devices) *Note (1)*



Note to Figure 3–4:

(1) Figures 3–4 and 3–5 assume that the address stays constant throughout and that the outputs are not registered.

Figure 3–5. Rising Edge Feed-Through Behavior (Stratix & Stratix GX Devices) Note (1)

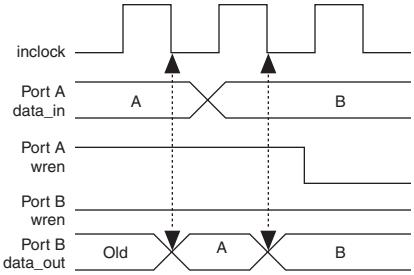


Note to Figure 3–5:
(1) Figures 3–4 and 3–5 assume that the address stays constant throughout and that the outputs are not registered.

Mixed-Port Read-During-Write Mode

Mixed-port read-during-write mode occurs when a RAM block in simple or true dual-port mode has one port reading and the other port writing to the same address location using the same clock. In APEX II and APEX 20K designs, the ESB outputs the old data in the first half of the clock cycle and the new data in the second half of the clock cycle, as indicated by Figure 3–6.

Figure 3–6. Mixed-Port Feed-Through Behavior (APEX II & APEX 20K Devices) Note (1)

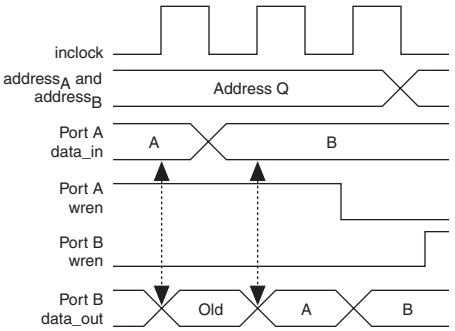


Note to Figure 3–6:
(1) Figure 3–6 assumes that outputs are not registered.

Stratix and Stratix GX device RAM outputs the new data on the rising edge of the clock cycle immediately after the data was written. When you use Stratix and Stratix GX M512 and M4K blocks, you can choose whether to output the old data at the targeted address or output a don't care value during the clock cycle when the new data is written. M-RAM blocks

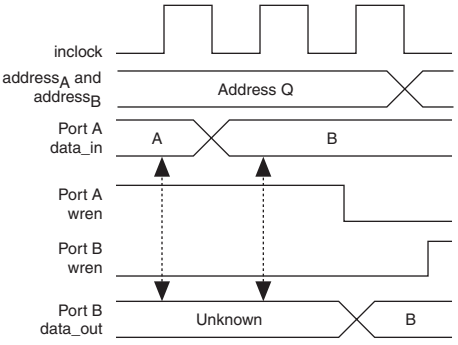
always output a don't care value. Figures 3–7 and 3–8 show the feed-through behavior of the mixed-port mode. You can use the `altsyncram` megafunction to set the output behavior during mixed-port read-during-write mode.

Figure 3–7. Mixed-Port Feed-Through Behavior (OLD_DATA) *Note (1)*



Note to Figure 3–7:
(1) Figures 3–7 and 3–8 assume that the address stays constant throughout and that the outputs are not registered.

Figure 3–8. Mixed-Port Feed-Through Behavior (DONT_CARE) *Note (1)*



Note to Figure 3–8:
(1) Figures 3–7 and 3–8 assume that the address stays constant throughout and that the outputs are not registered.

Memory Megafunctions

To convert RAM and ROM originally targeting the APEX II or APEX 20K architecture to Stratix or Stratix GX memory, specify Stratix or Stratix GX as the target family in the MegaWizard Plug-In Manager. The software

updates the memory module for the Stratix or Stratix GX architecture and instantiates the new synchronous memory megafunction, altsyncram, which supports both RAM and ROM blocks in the Stratix and Stratix GX architectures.

FIFO Conditions

First-in first-out (FIFO) functionality is slightly different in Stratix and Stratix GX devices compared to APEX II and APEX 20K devices. Stratix and Stratix GX devices do not support simultaneous reads and writes from an empty FIFO buffer. Also, Stratix and Stratix GX devices do not support the lpm_showahead parameter when targeting a FIFO buffer because the TriMatrix memory blocks are synchronous. The lpm_showahead parameter for APEX II and APEX 20K devices puts the FIFO buffer in "read-acknowledge" mode so the first data written into the FIFO buffer immediately flows through to the output. Other than these two differences, all APEX II and APEX 20K FIFO functions are fully compatible with the Stratix and Stratix GX architectures.

Design Migration Mode in Quartus II Software

The Quartus II software features a migration mode for simplifying the process of converting APEX II and APEX 20K memory functions to the Stratix or Stratix GX architecture. If the design can use the Stratix or Stratix GX altsyncram megafunction as a replacement for a previous APEX II or APEX 20K memory function while maintaining functionally similar behavior, the Quartus II software automatically converts the memory. The software produces a warning message during compilation reminding you to verify that the design migrated correctly.

For memory blocks with all inputs registered, the existing megafunction is converted to the new altsyncram megafunction. The software generates a warning when the altsyncram megafunction is incompatible. For example, a RAM block with all inputs registered except the read enable compiles with a warning message indicating that the read-enable port is registered.

You can suppress warning messages for the entire project or for individual memory blocks by setting the SUPPRESS_MEMORY_CONVERSION_WARNINGS parameter to "on" as a global parameter by selecting **Assignment Organizer** (Tools menu). In the **Assignment Organizer** window, click **Parameters** in the **Assignment Categories** box. Type SUPPRESS_MEMORY_CONVERSION_WARNINGS in the **Assignment Name** box and type ON in the **Assignment Setting** box. To suppress these warning messages on a per-memory-instance basis, set the SUPPRESS_MEMORY_CONVERSION_WARNINGS parameter in the Assignment Organizer to "on" for the memory instance.

If the functionality of the APEX II or APEX 20K memory megafunction differs from the `altsyncram` functionality and at least one clock feeds the memory megafunction, the Quartus II software converts the APEX II or APEX 20K memory megafunction to the Stratix or Stratix GX `altsyncram` megafunction. This conversion is useful for an initial evaluation of how a design might perform in Stratix or Stratix GX devices and should only be used for evaluation purposes. During this process, the Quartus II software generates a warning that the conversion may be functionally incorrect and timing results may not be accurate. Since the functionality may be incorrect and the compilation is only intended for comparative purposes, the Quartus II software does not generate a programming file. A functionally correct conversion requires manually instantiating the `altsyncram` megafunction and may require additional design changes.

If the previous memory function does not have a clock (fully asynchronous), the fitting-evaluation conversion results in an error message during compilation and does not successfully convert the design.

See *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs* for more information.

Table 3–5 summarizes the possible scenarios when using design migration mode and the resulting behavior of the Quartus II software.

The most common cases where design-migration mode may have difficulty converting the existing design are when:

- A port is reading from an address that is being written to by another port (mixed-port read-during-write mode). If both ports are using the same clock, the read port in Stratix and Stratix GX devices do not see the new data until the next clock cycle, after the new data was written.

- There are differences in power-up behavior between APEX II, APEX 20K, and Stratix and Stratix GX devices. You should manually account for these differences to maintain desired operation of the system.

Table 3–5. Migration Mode Summary

Memory Configuration	Conditions	Possible Instantiated Megafunctions	Quartus II Warning Message(s)	Programming File Generated

Single-port	All inputs are registered.	`altram`
		`altrom`
		`lpm_ram_dq`
		`lpm_ram_io`
		`lpm_rom`
			Power-up differences. *(1)*	Yes

Multi-port (two-, three-, or four-port functions)	All inputs are registered.	`altdpram`
		`lpm_ram_dp`
		`altqpram`
		`alt3pram`
			Power-up differences.
			Mixed-port read- during-write. *(1)*
				Yes

Dual-port	Read-enable ports are unregistered.
	Other inputs registered.
		`altdpram`
		`lpm_ram_dp`
		`altqpram`
		`alt3pram`
			Power-up differences.
			Mixed-port read- during-write.
			Read enable will be registered. *(1)*
				Yes

Dual-port	Any other unregistered port except read-enable ports.
	Clock available.
		`altdpram`
		`lpm_ram_dp`
		`altqpram`
		`alt3pram`
			Compile for fitting- evaluation purposes.	No

Single-port	At least one registered input.
	Clock available.
		`altram`
		`lpm_ram_dq`
		`lpm_ram_io`
			Compile for fitting- evaluation purposes.	No

No clock	No clock.	`altram`
		`altrom`
		`altdpram`
		`altqpram`
		`alt3pram`
		`altdpram`
		`lpm_ram_dq`
		`lpm_ram_io`
		`lpm_rom`
		`lpm_ram_dp`
		`lpm_ram_dp`
			Error – no conversion possible.	No

Note to Table 3–5:

(1) If the `SUPPRESS_MEMORY_COUNVERSION_WARNINGS` parameter is turned on, the Quartus II software does not issue these warnings.

DSP Block

Stratix and Stratix GX device DSP blocks outperform LE-based implementations for common DSP functions. Each DSP block contains several multipliers that can be configured for widths of 9, 18, or 36 bits. Depending on the mode of operation, these multipliers can optionally feed an adder/subtractor/accumulator or summation unit.

You can configure the DSP block's input registers to efficiently implement shift registers for serial input sharing, eliminating the need for external shift registers in LEs. You can add pipeline registers to the DSP block for accelerated operation. Registers are available at the input and output of the multiplier, and at the output of the adder/subtractor/accumulator or summation block.

DSP blocks have four modes of operation:

- Simple multiplier mode
- Multiply-accumulator mode
- Two-multipliers adder mode
- Four-multipliers adder mode

Associated megafunctions are available in the Quartus II software to implement each mode of the DSP block.

DSP Block Megafunctions

You can use the `lpm_mult` megafunction to configure the DSP block for simple multiplier mode. You can set the `lpm_mult` **Multiplier Implementation** option in the MegaWizard Plug-In Manager to either use the default implementation, ESBs, or the DSP blocks. If you select the **Use Default** option, the compiler first attempts to place the multiplier in the DSP blocks. However, under certain conditions, the compiler may implement the multiplier in LEs. The placement depends on factors such as DSP block resource consumption, the width of the multiplier, whether an operand is a constant, and other options chosen for the megafunction.

Stratix and Stratix GX devices do not support the **Use ESBs** option. If you select this option, the Quartus II software tries to place the multiplier in unused DSP blocks.

You can recompile APEX II or APEX 20K designs using the `lpm_mult` megafunction for Stratix and Stratix GX devices in the Quartus II software without changing the megafunction. This makes converting `lpm_mult` megafunction designs to Stratix or Stratix GX devices straightforward.

APEX II and APEX 20K designs use pipeline stages to improve registered performance of LE-based multipliers at the expense of latency. However, you may not need to use pipeline stages when targeting Stratix and Stratix GX high-speed DSP blocks. The DSP blocks offer three sets of dedicated pipeline registers. Therefore, Altera recommends that you reduce the number of pipeline stages to three or fewer and implement them in the DSP blocks. Additional pipeline stages are implemented in LEs, which add latency without providing any performance benefit.

For example, you can configure a DSP block for 36 × 36-bit multiplication using the `lpm_mult` megafunction. If you specify two pipeline stages, the software uses the DSP block input and pipeline registers. If you specify three pipeline stages, the software places the third pipeline stage in the DSP block output registers. This design yields the same performance with three pipeline stages because the critical path for a 36 × 36-bit operation is within the multiplier. With four or more pipeline stages, the device inefficiently uses LE resources for the additional pipeline stages. Therefore, if multiplier modules in APEX II or APEX 20K designs are converted to Stratix or Stratix GX designs and do not require the same number of pipeline stages, the surrounding circuitry must be modified to preserve the original functionality of the design.

A design with multipliers feeding an accumulator can use the `altmult_accum (MAC)` megafunction to set the DSP block in multiply-accumulator mode. If the APEX II or APEX 20K design already uses LE-based multipliers feeding an accumulator, the Quartus II software does not automatically instantiate the new `altmult_accum (MAC)` megafunction. Therefore, you should use the MegaWizard Plug-In Manager to instantiate the `altmult_accum (MAC)` megafunction. You can also use LeonardoSpectrum™ or Synplify synthesis tools, which have DSP block inference support, to instantiate the megafunction.

Designs that use multipliers feeding into adders can instantiate the new `altmult_add` megafunction to configure the DSP blocks for two-multipliers adder or four-multipliers adder mode. You can also use the `altmult_add` megafunction for stand-alone multipliers to take advantage of the DSP blocks features such as dynamic sign control of the inputs and the input shift register connections. These features are not accessible through the `lpm_mult` megafunction. If your APEX II or APEX 20K designs already use multipliers feeding an adder/subtractor, the Quartus II software does not automatically infer the new `altmult_add` megafunction. Therefore, you should step through the MegaWizard Plug-In Manager to instantiate the new `altmult_add` megafunction or use LeonardoSpectrum or Synplify synthesis tools, which have DSP block inference support.

Furthermore, the altmult_add and altmult_accum (MAC) megafunctions are only available for Stratix and Stratix GX devices because these megafunctions target Stratix and Stratix GX DSP blocks, which are not available in other device families. If you attempt to use these megafunctions in designs that target other Altera device families, the Quartus II software reports an error message. Use lpm_mult and an lpm_add_sub or an altaccumulate megafunction for similar functionality in other device families.

If you use a third-party synthesis tool, you may be able to avoid the megafunction conversion process. LeonardoSpectrum and Synplify provide inference support for lpm_mult, altmult_add, and altmult_accum (MAC) to use the DSP blocks.

If your design does not require you to implement all the multipliers in DSP blocks, you must manually set a global parameter or a parameter for each instance to force the tool to implement the lpm_mult megafunction in LEs. Depending on the synthesis tools, inference of DSP blocks is handled differently.

For more information about using DSP blocks in Stratix and Stratix GX devices, see the *DSP Blocks in Stratix & Stratix GX Devices* chapter of the *Stratix Device Handbook*.

PLLs & Clock Networks

Stratix and Stratix GX devices provide exceptional clock management with a hierarchical clock network and up to four enhanced phase-locked loops (PLLs) and eight fast PLLs versus the four general-purpose PLLs and four True-LVDS™ PLLs in APEX II devices. By providing superior clock interfacing, numerous advanced clocking features, and significant enhancements over APEX II and APEX 20K PLLs, the Stratix and Stratix GX device PLLs increase system performance and bandwidth.

Clock Networks

There are 16 global clock networks available throughout each Stratix or Stratix GX device as well as two fast regional and four regional clock networks per device quadrant, resulting in up to 40 unique clock networks per device. The increased number of dedicated clock resources available in Stratix and Stratix GX devices eliminate the need to use general-purpose I/O pins as clock inputs.

Stratix EP1S25 and smaller devices have 16 dedicated clock pins and EP1S30 and larger devices have four additional clock pins to feed various clocking networks. In comparison, APEX II devices have eight dedicated clock pins and APEX 20KE and APEX 20KC devices have four dedicated clock pins.

The dedicated clock pins in Stratix and Stratix GX devices can feed the PLL clock inputs, the global clock networks, and the regional clock networks. PLL outputs and internally-generated signals can also drive the global clock network. These global clocks are available throughout the entire device to clock all device resources.

Stratix and Stratix GX devices are divided into four quadrants, each equipped with four regional clock networks. The regional clock network can be fed by either the dedicated clock pins or the PLL outputs within its device quadrant. The regional clock network can only feed device resources within its particular device quadrant.

Each Stratix and Stratix GX device provides eight dedicated fast clock I/O pins `FCLK[7..0]` versus four dedicated fast I/O pins in APEX II and APEX 20K devices. The fast regional clock network can be fed by these dedicated `FCLK[7..0]` pins or by the I/O interconnect. The I/O interconnect allows internal logic or any I/O pin to drive the fast regional clock network. The fast regional clock network is available for general-purpose clocking as well as high fan-out control signals such as clear, preset, enable, `TRDY` and `IRDY` for PCI applications, or bidirectional or output pins.

EP1S25 and smaller devices have eight fast regional clock networks, two per device quadrant. The quadrants in EP1S30 and larger devices are divided in half, and each half-quadrant can be clocked by one of the eight fast regional networks. Additionally, each fast regional clock network can drive its neighboring half-quadrant (within the same device quadrant).

PLLs

Table 3–6 highlights Stratix and Stratix GX PLL enhancements to existing APEX II, APEX 20KE and APEX 20KC PLL features.

Table 3–6. Stratix & Stratix GX PLL vs. APEX II, APEX 20KE & APEX 20KC PLL Features (Part 1 of 2)

Feature	Stratix & Stratix GX		APEX II PLLs	APEX 20KE & APEX 20KC PLLs
	Enhanced PLLs	Fast PLLs		
Number of PLLs	Two (EP1S30 and smaller devices); four (EP1S40 and larger devices) *(9)*	Four (EP1S25 and smaller devices); eight (EP1S30 and larger devices) *(10)*	Four general-purpose PLLs and four LVDS PLLs	Up to four general-purpose PLLs. Up to two LVDS PLLs. *(1)*
Minimum input frequency	3 MHz	15 MHz	1.5 MHz	1.5 MHz
Maximum input frequency	250 to 582 MHz *(2)*	644.5 MHz *(11)*	420 MHz	420 MHz

Table 3–6. Stratix & Stratix GX PLL vs. APEX II, APEX 20KE & APEX 20KC PLL Features (Part 2 of 2)

Feature	Stratix & Stratix GX		APEX II PLLs	APEX 20KE & APEX 20KC PLLs
	Enhanced PLLs	Fast PLLs		
Internal clock outputs per PLL	6	3 *(3)*	2	2
External clock outputs per PLL	Four differential/eight singled-ended or one single-ended *(4)*	Yes *(5)*	1	1
Phase Shift	Down to 160-ps increments *(6)*	Down to 125-ps increments *(6)*	500-ps to 1-ns resolution	0.4- to 1-ns resolution
Time shift	250-ps increments for ± 3 ns *(7)*	No	No	No
M counter values	1 to 512	1 to 32	1 to 160	2 to 160
N counter values	1 to 512	N/A	1 to 16	1 to 16
PLL clock input sharing	No	Yes	Yes	Yes
T1/E1 rate conversion *(8)*	No	No	Yes	Yes

Notes to Table 3–6:

(1) EP20K200E and smaller devices only have two general-purpose PLLs. EP20K400E and larger devices have two LVDS PLLs and four general-purpose PLLs. For more information, see *AN 115: Using the ClockLock & ClockBoost PLL Features in APEX Devices*.

(2) The maximum input frequency for Stratix and Stratix GX enhanced PLLs depends on the I/O standard used with that input clock pin. For more information, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1*.

(3) Fast PLLs 1, 2, 3, and 4 have three internal clock output ports per PLL. Fast PLLs 7, 8, 9, and 10 have two internal clock output ports per PLL.

(4) Every Stratix device has two enhanced PLLs with eight single-ended or four differential outputs each. Two additional enhanced PLLs in EP1S80, EP1S60, and EP1S40 devices each have one single-ended output.

(5) Any I/O pin can be driven by the fast PLL global or regional outputs as an external clock output pin.

(6) The smallest phase shift unit is determined by the voltage-controlled oscillator (VCO) period divided by 8.

(7) There is a maximum of 3 ns between any two PLL clock outputs.

(8) The T1 clock frequency is 1.544 MHz and the E1 clock frequency is 2.048 MHz, which violates the minimum clock input frequency requirement of the Stratix PLL.

(9) Stratix GX EP1SGX10 and EP1SGX25 contain two. EP1SGX40 contains four.

(10) Stratix GX EP1SGX10 and EP1SGX25 contain two. EP1SGX40 contains four.

(11) Stratix GX supports clock rates of 1 Gbps using DPA.

Enhanced PLLs

Stratix and Stratix GX devices provide up to four enhanced PLLs with advanced PLL features. In addition to the feature changes mentioned in Table 3–6, Stratix and Stratix GX device PLLs include many new,

advanced features to improve system timing management and performance. Table 3–7 shows some of the new features available in Stratix and Stratix GX enhanced PLLs.

Table 3–7. Stratix & Stratix GX Enhanced PLL Features

Feature	Description
Programmable duty cycle *(1)*	Allows variable duty cycle for each PLL clock output.
PLL clock outputs can feed logic array *(1)*	Allows the PLL clock outputs to feed data ports of registers or combinatorial logic.
PLL locked output can feed the logic array *(1)*	Allows the PLL locked port to feed data ports of registers or combinatorial logic.
Multiplication allowed in zero-delay buffer mode or external feedback mode	The PLL clock outputs can be a multiplied or divided down ratio of the PLL input clock.
Programmable phase shift allowed in zero-delay buffer mode or external feedback mode *(2)*	The PLL clock outputs can be phase shifted. The phase shift is relative to the PLL clock output.
Phase frequency detector (PFD) disable	Allows the VCO to operate at its last set control voltage and frequency with some long term drift.
Clock output disable *(3)*	PLL maintains lock with output clocks disabled. *(4)*
Programmable lock detect & gated lock	Holds the lock signal low for a programmable number of input clock cycles.
Dynamic clock switchover	Enables the PLL to switch between two reference input clocks, either for clock redundancy or dual-clock domain applications.
PLL reconfiguration	Allows the counters and delay elements within the PLL to be reconfigured in real-time without reloading a programmer object file (**.pof**).
Programmable bandwidth	Provides advanced control of the PLL bandwidth by using the programmable control of the PLL loop characteristics.
Spread spectrum	Modulates the target frequency over a frequency range to reduce electromagnetic interference (EMI) emissions.

Notes to Table 3–7:

(1) These features are also available in fast PLLs.
(2) In addition to the delay chains at each counter, you can specify the programmable phase shift for each PLL output at fine and coarse levels.
(3) Each PLL clock output has an associated clock enable signal.
(4) If the PLL is used in external feedback mode, the PLL will need to relock.

Fast PLLs

Stratix and Stratix GX fast PLLs are similar to the APEX II True-LVDS PLLs in that the *W* setting, which governs the relationship between the clock input and the data rate, and the *J* setting, which controls the width

of the high-speed differential I/O data bus, do not have to be equal. Additionally, Stratix and Stratix GX fast PLLs offer up to three clock outputs, two multiplied high-speed PLL clocks to drive the serializer/deserializer (SERDES) block and/or an external pin, and a low-speed clock to drive the logic array. You can use fast PLLs for both high-speed interfacing and for general-purpose PLL applications.

Table 3–8 shows the differences between Stratix and Stratix GX fast PLLs and APEX II and APEX 20K True-LVDS PLLs.

Table 3–8. Stratix & Stratix GX Fast PLL vs. APEX II & APEX 20K True-LVDS PLL

Feature	Stratix & Stratix GX	APEX II	APEX 20KE APEX 20KC
Number of fast PLLs or True-LVDS PLLs *(1)*	Four (EP1S25 and smaller devices) fast PLLs Eight (EP1S30 and larger devices) fast PLLs *(4)*	Four True-LVDS PLLs	Two True-LVDS PLLs *(2)*
Number of channels per transmitter/receiver block	20	18	18
VCO frequency	300 to 840 MHz *(5)*	200 MHz to 1GHz	200 to 840 MHz
Minimum input frequency *M* = 4, 5, 6	300 – *M* MHz	50 MHz	50 MHz *M* = 4 *(3)*
Minimum input frequency *M* = 7, 8, 9, 10	300 – *M* MHz	30 MHz	30 MHz *M* = 7, 8 *(3)*

Notes to Table 3–8:
(1) You can also use Stratix and Stratix GX device fast PLLs for general-purpose PLL applications.
(2) EP20K400E and larger devices have two True-LVDS PLLs.
(3) In APEX 20KE and APEX 20KC devices, *M* = 4, 7, or 8.
(4) Stratix GX EP1SGX10 and EP1SGX25 contain two. EP1SGX10 contains four.
(5) Stratix GX supports a frequency range of 300–1000 MHz (using DPA).

The Stratix and Stratix GX fast PLL VCO frequency range is 300 to 840 MHz, and the APEX II True-LVDS PLL VCO frequency range is 200 MHz to 1 GHz. Therefore, you must update designs that use a data rate of less than 300 megabits per second (Mbps) to use the enhanced PLLs and M512 RAM blocks in SERDES bypass mode. Additionally, you must update designs that use a data rate faster than 840 Mbps.

altpll Megafunction

Altera recommends that you replace instances of the `altclklock` megafunction with the `altpll` megafunction to take advantage of new Stratix and Stratix GX PLL features. Although in most cases you can retarget your APEX II or APEX 20K design to a Stratix or Stratix GX

device with the altclklock megafunction, there are specific cases where you must use the altpll megafunction, as explained in this section.

In the MegaWizard Plug-In Manager, select the altpll megafunction in the I/O directory from the **Available Megafunctions** box (see Figure 3–9). The altclklock megafunction is also available from the Quartus II software for backward compatibility, but instantiates the new altpll megafunction when targeting Stratix or Stratix GX devices. The Quartus II Compiler automatically selects whether the altpll module uses either an enhanced PLL or a fast PLL based on the design's PLL needs and the feature requirements of each PLL.

Figure 3–9. altpll Megafunction Selection in the MegaWizard Plug-In Manager

You can compile APEX II, APEX 20KE, and APEX 20KC designs using the altclklock megafunction in normal mode for Stratix and Stratix GX devices without updating the megafunction. However, you should replace the altclklock megafunction with the altpll megafunction. If the Quartus II software cannot implement the requested clock multiplication and division of the PLL, the compiler reports an error message with the appropriate reason stated.

APEX II, APEX 20KE, and APEX 20KC devices have only one external clock output available per PLL. Therefore, when retargeting an APEX II, APEX 20KE, or APEX 20KC design that uses PLLs in zero delay buffer mode or external feedback mode to a Stratix or Stratix GX device, you should replace instances of the `altclklock` megafunction. If an APEX II, APEX 20KE, or APEX 20KC `altclklock` module only uses one PLL clock output (internal or external) and is compiled to target a Stratix or Stratix GX device, the design compiles successfully with a warning that the design uses the Stratix or Stratix GX PLL external clock output, `extclk0`. However, if the APEX II, APEX 20KE, or APEX 20KC PLL has more than one PLL clock output, you must replace instances of the `altclklock` megafunction with the `altpll` megafunction because the Quartus II Compiler does not know which PLL clock output is fed to an external output pin or fed back to the Stratix or Stratix GX device `fbin` pin. For example, if an APEX II, APEX 20KE, or APEX 20KC design with an `altclklock` megafunction uses the `clock0` output port to feed the external clock output pin and the `clock1` output port to feed the internal logic array, the Quartus II software generates an error during compilation and you must use the MegaWizard Plug-In Manager to instantiate the `altpll` megafunction. By using the `altpll` megafunction, you can choose which of the four external clock outputs to use and take advantage of the new Stratix and Stratix GX PLL features now available in the zero delay buffer mode or external feedback mode.

Timing Analysis

When the Quartus II software performs a timing analysis for APEX II, APEX 20KE, or APEX 20KC designs, PLL clock settings override the project clock settings. However, during timing analysis for Stratix and Stratix GX designs using PLLs, the project clock settings override the PLL input clock frequency and duty cycle settings. The MegaWizard Plug-In Manager does not use the project clock settings to determine the `altpll` parameters. This saves time with designs that use features such as clock switchover or PLL reconfiguration because the Quartus II software can perform a timing analysis without recompiling the design. It is important to note the following:

- A warning during compilation reports that the project clock settings overrides the PLL clock settings.
- The project clock setting overrides the PLL clock settings for timing-driven compilation.
- The compiler will check the lock frequency range of the PLL. If the frequency specified in the project clock settings is outside the lock frequency range, the PLL clock settings will not be overridden.
- Performing a timing analysis without recompiling your design does not change the programming files. You must recompile your design to update the programming files.

- A **Default Required f_{MAX}** setting does not override the PLL clock settings. Only individual clock settings override the PLL clock settings.

Therefore, you can enter different project clock settings corresponding to new PLL settings and accelerate timing analysis by eliminating a full compilation cycle.

For more information about using Stratix and Stratix GX PLLs, see the *General-Purpose PLLs in Stratix & Stratix GX Devices* chapter.

I/O Structure

The Stratix and Stratix GX I/O element (IOE) architecture is similar to the APEX II architecture, with a total of six registers and a latch in each IOE. The registers are organized in three sets: two output registers to drive a single or double-data rate (DDR) output path, two input registers and a latch to support a single or DDR input path, and two output enable registers to enhance clock-to-output enable timing or for DDR SDRAM interfacing. A new synchronous reset signal is available to each of the three sets of registers for preset or clear, or neither. In addition to the advanced IOE architecture, the Stratix and Stratix GX IOE features dedicated circuitry for external RAM interfacing, new I/O standards, differential on-chip termination, and high-speed differential I/O standard support.

External RAM Interfacing

The advanced Stratix and Stratix GX IOE architecture includes dedicated circuitry to interface with external RAM. This circuitry provides enhanced support for external high-speed memory devices such as DDR SDRAM and FCRAM. The DDR SDRAM interface uses a bidirectional signal, DQS, to clock data, DQ, at both the transmitting and receiving device. Stratix and Stratix GX devices transmit the DQS signal with the DQ data signals to minimize clock to data skew.

Stratix and Stratix GX devices include groups of programmable DQS and DQ pins, in the top and bottom I/O banks of the device. Each group consists of a DQS pin that supports a fixed number of DQ pins. The number of DQ pins depends on the DQ bus mode. When using the external RAM interfacing circuitry, the DQS pin drives a dedicated clock network that feeds the DQ pins residing in that bank. The Stratix and Stratix GX IOE has programmable delay chains that can phase shift the DQS signal by 90° or 72° to ensure data is sampled at the appropriate point in time. Therefore, the Stratix and Stratix GX devices make full use of the IOEs, and remove the need to build the input data path in the logic array. You can make these I/O assignments in the Quartus II Assignment Organizer.

For more information on external RAM interfacing, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* or the *Stratix GX Device Family Data Sheet* in the *Stratix GX Device Family Handbook, Volume 1*.

I/O Standard Support

The Stratix and Stratix GX devices support all of the I/O standards that APEX II and APEX 20K devices support, including high-speed differential I/O standards such as LVDS, LVPECL, PCML, and HyperTransport™ technology, differential HSTL on input and output clocks, and differential SSTL on output clocks. Stratix and Stratix GX devices also introduce support for SSTL-18 Class I & II. Similar to APEX II devices, Stratix and Stratix GX devices only support certain I/O standards in designated I/O banks. In addition, `vref` pins are dedicated pins in Stratix and Stratix GX devices and now support up to 40 input pins.

For more information about I/O standard support in Stratix and Stratix GX devices, see the *Selectable I/O Standards in Stratix & Stratix GX Devices* chapter.

High-Speed Differential I/O Standards

Stratix and Stratix GX devices support high-speed differential interfaces at speeds up to 840 Mbps using high-speed PLLs that drive a dedicated clock network to the SERDES. Each fast PLL can drive up to 20 high-speed channels. Stratix and Stratix GX devices use enhanced PLLs and M512 RAM blocks to provide up to 420 Mbps performance for SERDES bypass clock interfacing. There is no restriction on the number of channels that can be clocked using this scenario.

Stratix and Stratix GX devices have a different number of differential channels than APEX II devices. Tables 3–9 and 3–10 highlight the number of differential channels supported in Stratix and Stratix GX devices.

Table 3–9. Number of Dedicated DIfferential Channels in Stratix Devices (Part 1 of 2) *Note (1)*

Device	Pin Count	Number of Receiver Channels	Number of Transmitter Channels
EP1S10	672	36	36
	780	44	44
EP1S20	672	50	48
	780	66	66

Table 3–9. Number of Dedicated DIfferential Channels in Stratix Devices (Part 2 of 2) Note (1)

Device	Pin Count	Number of Receiver Channels	Number of Transmitter Channels
EP1S25	672	58	56
	780	66	70
	1,020	78	78
EP1S30	780	66	70
	956	80	80
	1,020	80	80
		2	2
EP1S40	956	80	80
	1,020	80	80
		10	10
	1,508	80	80
		10	10
EP1S60	956	80	80
	1,020	80	80
		10	12
	1,508	80	80
		36	36
EP1S80	956	80	80
		0	40
	1,508	80	80
		56	72

Note to Table 3–9:

(1) For information on channel speeds, see the *Stratix Device Family Data Sheet* section of the *Stratix Device Handbook, Volume 1* and the *High-Speed Differential I/O Interfaces* chapter in the *Stratix Device Handbook, Volume 2*.

Table 3–10. Number of Dedicated DIfferential Channels in Stratix GX Devices (Part 1 of 2) Note (1)

Device	Pin Count	Number of Transceivers	Number of Source-Synchronous Channels
EP1SGX10 C	672	4	22
EP1SGX10 D	672	8	22

Table 3–10. Number of Dedicated DIfferential Channels in Stratix GX Devices (Part 2 of 2) Note (1)

Device	Pin Count	Number of Transceivers	Number of Source-Synchronous Channels
EP1SGX25 C	672	4	39
EP1SGX25 D	672/1,020	8	39
EP1SGX25 F	1,020	16	39
EP1SGX40 D	1,020	8	45
EP1SGX40 G	1,020	20	45

Note to Table 3–10:

(1) For information on channel speeds, see the *Stratix GX Device Family Data Sheet* section of the *Stratix GX Device Handbook, Volume 1* and the *High-Speed Source-Synchronous Differential I/O Interfaces in Stratix GX Devices* chapter of the *Stratix GX Device Handbook, Volume 2*.

The differential I/O within Stratix GX also provides dynamic phase alignment (DPA). DPA enables the differential I/O to operate up to 1 Gbps per channel. DPA automatically and continuously tracks fluctuations caused by system variations and self-adjusts to eliminate the phase skew between the multiplied clock and the serial data. The block contains a dynamic phase selector for phase detection and selection, a SERDES, a synchronizer, and a data realigner circuit. You can bypass the dynamic phase aligner without affecting the basic source-synchronous operation of the channel by using a separate deserializer.

If you compile an APEX II LVDS design that uses clock-data synchronization (CDS) for a Stratix or Stratix GX device, the Quartus II software issues a warning during compilation that Stratix and Stratix GX devices do not support CDS.

Stratix and Stratix GX devices offer a flexible solution using new byte realignment circuitry to correct for byte misalignment by shifting, or slipping, data bits. Stratix and Stratix GX devices activate the byte realignment circuitry when an external pin (`rx_data_align`) or an internal custom-made state machine asserts the `SYNC` node high.

APEX II, APEX 20KE, and APEX 20KCdevices have a dedicated transmitter clock output pin (`LVDSTXOUTCLK`). In Stratix and Stratix GX devices, a transmitter `dataout` channel with an LVDS clock (fast clock) generates the transmitter clock output. Therefore, you can drive any channel as an output clock to an I/O pin, not just dedicated clock output pins. This solution offers better versatility to address various applications that require more complex clocking schemes.

For more information on differential I/O support, data realignment, and the transmitter clock output in Stratix and Stratix GX devices, see the *High-Speed Differential I/O Interfaces in Stratix Devices* chapter.

altlvds Megafunction

To take full advantage of the high-speed differential I/O standards available in Stratix and Stratix GX devices, you should update each instance of the `altlvds` megafunction in APEX II, APEX 20KE, and APEX 20KC designs. In the MegaWizard Plug-In Manager, choose the `altlvds` megafunction, select Stratix or Stratix GX as the target device family, update the megafunction, and recompile your design.

The `altlvds` megafunction supports new Stratix and Stratix GX parameters that are not available for APEX II, APEX 20KE, and APEX 20KC devices. Tables 3–11 and 3–12 describe the new parameters for the LVDS receiver and LVDS transmitter, respectively.

Table 3–11. New altlvds Parameters for Stratix LVDS Receiver Note (1)

Parameter	Function
`input_data_rate` *(2)*	Specifies the data rate in Mbps. This parameter replaces the multiplication factor *W*.
`inclock_data_alignment`	Indicates the alignment of `rx_inclk` and `rx_in` data.
`rx_data_align`	Drives the data alignment port of the fast PLL and enables byte realignment circuitry.
`registered_data_align_input`	Registers the `rx_data_align` input port to be clocked by `rx_outclock`.
`common_rx_tx_pll` *(3)*	Indicates the fast PLL can be shared between receiver and transmitter applications.

Table 3–12. New altlvds Parameters for Stratix LVDS Transmitter (Part 1 of 2) Note (1)

Parameter	Function
`output_data_rate` *(2)*	Specifies the data rate in Mbps. This parameter replaces the multiplication factor *W*.
`inclock_data_alignment`	Indicates the alignment of `tx_inclk` and `tx_in` data.
`outclock_alignment`	Specifies the alignment of `tx_outclock` and `tx_out` data.
`registered_input`	Specifies the clock source for the input synchronization registers, which can be either `tx_inclock` or `tx_coreclock`. Used only when the **Registered Inputs** option is selected.

Table 3–12. New altlvds Parameters for Stratix LVDS Transmitter (Part 2 of 2) Note (1)	
Parameter	**Function**
common_rx_tx_pll (3)	Indicates the fast PLL can be shared between receiver and transmitter applications.

Notes to Tables 3–11 and 3–12:
(1) You can specify these parameters in the MegaWizard Plug-In Manager.
(2) You must specify a data rate in the MegaWizard Plug-In Manager instead of a *W* factor.
(3) The same fast PLL can be used to clock both the receiver and transmitter only if both are running at the same frequency.

Above the standard I/O offered by APEX II, APEX 20K, and Stratix devices, Stratix GX devices provide up to 20 3.175 Gbps transceivers. The transceivers provide high-speed serial links for chip-to-chip, backplane, and line-side connectivity and support a number of the emerging high-speed protocols. You can find more information in the *Stratix GX Family Data Sheet* in the *Stratix GX Family Handbook, Volume 1*.

Configuration

The Stratix and Stratix GX devices supports all current configuration schemes, including the use of enhanced configuration devices, passive serial (PS), passive parallel asynchronous (PPA), fast passive parallel (FPP), and JTAG. Stratix and Stratix GX devices also provide a number of new configuration enhancements that you can take advantage of when migrating APEX II and APEX 20K designs to Stratix and Stratix GX devices.

Configuration Speed & Schemes

You can configure Stratix and Stratix GX devices at a maximum clock speed of 100 MHz, which is faster than the 66-MHz and 33-MHz maximum configuration speeds for APEX II and APEX 20K devices, respectively. Similar to APEX II devices, you can use 8-bit parallel data to configure Stratix and Stratix GX devices (the target device can receive byte-wide configuration data on each clock cycle) significantly speeding up configuration times.

You can select a configuration scheme based on how the MSEL pins are driven. Stratix and Stratix GX devices have three MSEL pins (APEX II and APEX 20K devices have two MSEL pins) for determining the configuration scheme.

For more information about Stratix and Stratix GX configuration schemes, see the *Configuring Stratix & Stratix GX Devices* chapter.

Remote Update Configuration

The APEX 20K device family introduced the concept of remote update configuration, where you could send the APEX 20K device new configuration files from a remote source and the device would store the files in flash memory and reconfigure itself with the new configuration data. The Stratix and Stratix GX devices enhance support for remote update configuration with new, dedicated circuitry to handle and recover from errors. If an error occurs either during device configuration or in user mode, this new circuitry reconfigures the Stratix or Stratix GX device to a known state. Additionally, the Stratix and Stratix GX devices have a user watchdog timer to ensure the application configuration data executes successfully during user mode. User logic must continually reset this watchdog timer in order to validate that the application configuration data is functioning properly.

For more information about how to use the remote and local update modes, see the *Remote System Configuration with Stratix & Stratix GX Devices* chapter.

JTAG Instruction Support

Stratix and Stratix GX devices support two new JTAG instructions, `PULSE_NCONFIG` and `CONFIG_IO`. The `PULSE_NCONFIG` instruction emulates pulsing the `nCONFIG` signal low to trigger reconfiguration, while the actual `nCONFIG` pin on the device is unaffected. The `CONFIG_IO` instruction allows you to use the JTAG chain to configure I/O standards for all pins. Because this instruction interrupts device configuration, you should reconfigure the Stratix or Stratix GX device after you finish JTAG testing to ensure proper device operation. Table 3–13 compares JTAG instruction support in Stratix and Stratix GX devices versus APEX II and APEX 20K devices. For further information about the supported JTAG instructions, see the appropriate device family data sheet.

Table 3–13. JTAG Instruction Support (Part 1 of 2)

JTAG Instruction	Stratix	APEX II	APEX 20K
SAMPLE/PRELOAD	✓	✓	✓
EXTEST	✓	✓	✓
BYPASS	✓	✓	✓
USERCODE	✓	✓	✓
IDCODE	✓	✓	✓
ICR Instructions	✓	✓	✓

Table 3–13. JTAG Instruction Support (Part 2 of 2)

JTAG Instruction	Stratix	APEX II	APEX 20K
SignalTap™ II Instructions	✓	✓	✓
HIGHZ	✓	✓	
CLAMP	✓	✓	
PULSE_NCONFIG	✓		
CONFIG_IO	✓		

Conclusion

The Stratix and Stratix GX devices extend the advanced features available in the APEX II and APEX 20K device families to deliver a complete system-on-a-programmable-chip (SOPC) solution. By following these guidelines, you can easily transition current APEX II and APEX 20K designs to take advantage of the new features available in Stratix and Stratix GX devices.

4. Stratix GX Board Design Guidelines

SGX53001-1.0

Introduction

Digital board design has become more complicated over the years. Devices have 0.13-µm gate lengths and are powered by 1.5-V power supplies. While the trend toward miniaturization continues, device speeds are increasing. FPGAs now have embedded transceivers that can support serial data transmission at 3.125 Gigabits per second (Gbps). The low-voltage supply results in decreased noise margin; therefore, small voltage noise on the board can cause a bit to flip. Similarly, voltage noise directly increases the clock jitter, which reduces the timing margin. If the jitter is large enough, the bit transition boundaries can become fuzzy, and the current data bit could be confused for the past or future bit. To reduce voltage and timing noise and ensure error-free data transmission, good design techniques are essential.

The following things are critical when designing a successful high-speed digital board:

- Clean power supply design (see "Power Circuitry" on page 4–6)
- Decoupling capacitor selection (see "Decoupling Circuitry Design" on page 4–13)
- Analog ground and power isolation (see "Ground Plane & Island Design" on page 4–30)
- Transmission line termination (see "Transmission Line Termination" on page 4–52)
- Crosstalk minimization (see "Crosstalk" on page 4–72)
- Impedance discontinuity minimization (see "Transmission Line Routing" on page 4–58)
- Proper differential signal routing (see "Transmission Line Routing" on page 4–58)

This document explains high-speed board design concepts and techniques as applied to Stratix® GX devices and explains the major aspects of successful high-speed board design. The Stratix GX device and the Stratix GX development board are used as the platforms for the tests and measurements described in this document. Topics covered include clock circuitry design, power circuitry design, power supply decoupling, transmission line topologies, length matching, AC versus DC coupling, termination techniques, time domain reflectometer (TDR) usage, and S-parameters.

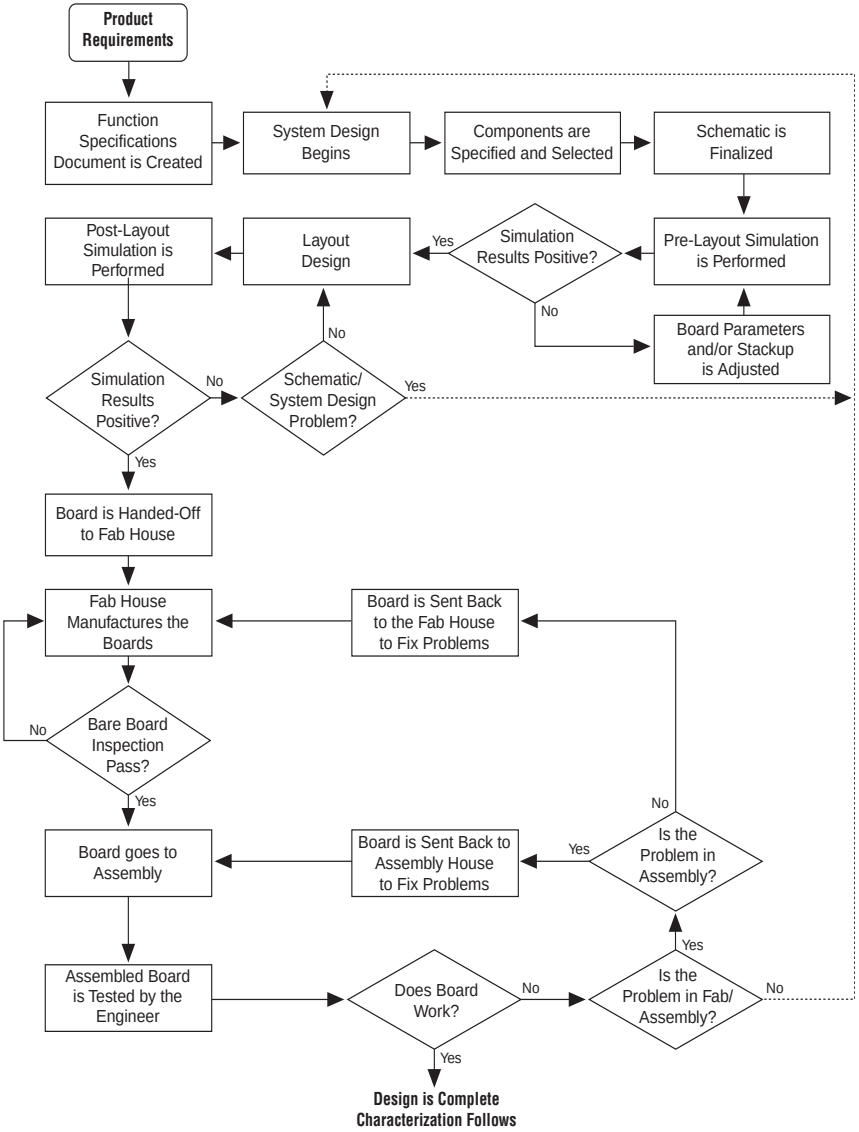
Board Design Overview

A typical high-speed board design begins with a product requirements document, a concept review, and a functional specification. After the specification is finalized, actual system design begins with the selection of key components. At this point, timing analysis for all the interfaces and power analysis—to determine the power requirements—is performed. Based on the power analysis, power supply modules and regulators are chosen, and a decoupling scheme is specified. Based on the timing analysis, length-matching criteria for the buses is established. Upon completion of these tasks, as well as completion of the system design and selection of remaining components, the schematics are created. These are typically reviewed among engineers who make any necessary changes, after which they are given to the layout designer. A layout guidelines document, which specifies how to place components on the board and how to route the traces, is created and given to the layout designer as well.

A pre-layout simulation is performed to determine the ideal stackup, trace widths, spacing, and other routing requirements. Any changes to the schematic, based on the simulation results, are incorporated in the schematic and provided to the layout designer. When the layout is complete, a post-layout simulation is performed on the critical sections of the board to ensure that there are no major signal integrity problems. Based on the results of the post-layout simulation, any changes required are incorporated into the layout, and finally the layout is released to the fabrication house for board manufacturing.

Figure 4–1 shows a typical board design process.

Figure 4–1. Simplified View of a Typical High-Speed Board Design Process



Support Circuitry Design

Support circuitry includes clocks, power, and decoupling. This section contains detailed guidelines for designing these parts of a high-speed board.

Clock Circuitry

Clocks serve as references for digital signals and must be designed very carefully. Any noise on the clock signal reduces the timing margin for digital signals. Typically, on-board clocks are derived from crystal-based oscillators. For the Stratix GX development board, Altera use the standard 3.3-V SM7745DV series CMOS crystals from Pletronics for high-frequency applications. This crystal has ± 50 ppm stability and less than 1 ps RMS jitter from 12 kHz to 20 MHz from the carrier. For low-frequency applications Altera uses the standard 3.3 V SM7745HV series CMOS crystals from Pletronics. The stability of this crystal is ± 50 ppm, but the jitter is not specified. Altera recommends these or equivalent crystals.

For maximum noise immunity, the crystal oscillator's CMOS output typically needs to be converted to a differential standard like LVDS before it is fed to the Stratix GX device. An ICS8545 CMOS to LVDS buffer from Integrated Circuit Systems, Inc. achieves this conversion. A side benefit of using a clock buffer like this one is that it provides multiple outputs. This feature is very helpful because clock lines need to have one source and one destination for best signal integrity.

Other things to remember for optimal clock performance are:

- Place the crystal oscillator close to the driver circuitry for best jitter performance.
- Route the differential clock lines tightly coupled to each other.
- The oscillators and the drivers need clean power supplies.
- Place series termination at the clock output if the trace is too long.
- Place parallel differential termination close to the receiver pins.

Whether a particular trace is too long depends on the edge rate of the clock and on the speed at which the signal propagates on the board. If the electrical length of the trace is small (1/10 or 10%)[1] compared to the rise time, then no termination is needed. Otherwise, use series termination. The propagation speed for microstrip and stripline, two of the commonly used transmission line topologies, is shown in the following equations[2]:

$$Speed(Microstrip) = \frac{1}{85\sqrt{0.475\varepsilon_r + 0.67}} inches / ps$$

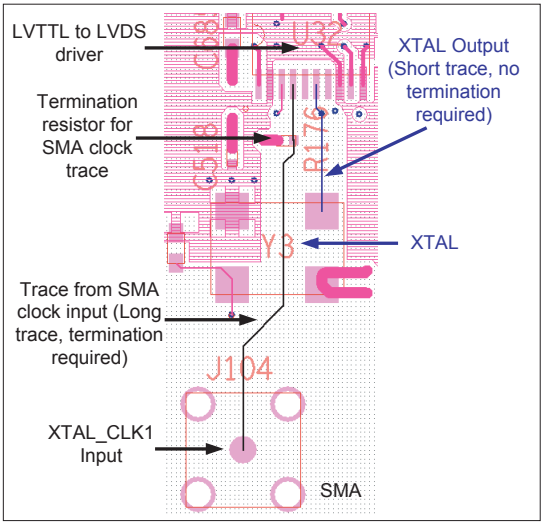
[1] Some designers use 1/6 or 16% as the cut-off point. The important thing to remember is that the length needs to be small.
[2] Refer to *High-Speed Digital Design: A Handbook of Black Magic* by Johnson and Graham, pp186-188.

$$Speed(Stripline) = \frac{1}{85\sqrt{\varepsilon_r}} inches / ps$$

Using ε_r of 4.5 for typical FR-4 material, the propagation speed for microstrip and stripline traces works out to be 0.007021 inches/ps and 0.005546 inches/ps, respectively. Expressed another way, the signals travel 142 ps/inch in microstrip traces and 180 ps/inch in stripline traces.

Case Study: As a case study, the microstrip trace length from the output of the 156.25 MHz crystal to the driver (see Figure 4–2) is about 0.28 in on the Stratix GX development board. The signal takes 142 * 0.28 = 39.76 ps to traverse this distance. The rise time of the crystal output is 1.0 ns (see the *Pletronics SM7745HV* data sheet). The signal transit time is less than 4% of the signal rise time, so no termination is needed. When the clock is being brought to the board through an SMA connector (J104), the trace length from the SMA to the clock driver is about four times the crystal transit time calculated earlier, in addition to the SMA cable length. In this case, a termination is definitely needed. Use R176 for termination, as shown in Figure 4–2.

Figure 4–2. Layout Example for Clock Distribution



The specifications for some I/O standards always require termination regardless of the trace length. These include PCML, LVDS, LVPECL, SSTL-II, and HSTL-II.

Isolated Power & Ground Plane Design

Isolated power and ground planes are created by using ferrite beads. A ferrite bead acts as a DC short but presents an impedance to the passage of high-frequency noise. It can be beneficial in systems that have lots of switching noise. Sources of switching noise include switching power supplies and simultaneous switching outputs. For Stratix GX devices, Altera recommends isolating the high-speed ground (GND_GXB) from the digital ground (GND), so that the noise from the digital ground does not reach the high-speed circuitry. In the Stratix GX development board, there are three ground planes: GND, GND_PLL, and GND_GXB, for characterization purposes. For most high-density designs, Altera recommends only two ground planes: GND and GND_GXB. If the board does not have lots of switching circuitry and is otherwise cleanly designed, one ground plane is acceptable.

Currently, the benefits of split power planes for the transceiver quads are under evaluation. Preliminary tests suggest that for most cases the benefits of isolating power for the quads are minimal. See "Power Plane & Island Design" and "Ground Plane & Island Design" for more information.

Power Circuitry

As gate lengths shrink, the power supply voltages of transistors also decrease. As a result, the noise margin also decreases. The reduced noise margin makes the transistors even more sensitive to noise on the power supply. The features of the board must be understood before the power supply can be designed. Some key characteristics of a high-density board with one or more FPGAs include:

- Hundreds of I/O pins with varying voltage standards switching at several frequencies.
- The core LE use is highly unpredictable and depends on the actual design loaded. The user could load a design that uses close to 100% of the LEs, embedded memory, and hard logic (like transceivers), and run everything at full speed.
- Analog and digital signals on the board. For example, TTL switching. is digital while Stratix GX transceiver outputs are analog signals.
- The potential for noise to couple between different power and ground planes.

Regulator Selection

Altera recommends selecting linear regulators as much as possible, because they are easy to design and their performance is less critical to layout. You must sometimes select switching regulators, because linear

regulators cannot supply the desired amount of power efficiently. In those cases, you must design and layout the switching regulators very carefully. All major switching components and traces must be closely contained on the same layer.

The Stratix GX development board uses Fairchild FAN5066 switching regulators (U36 and U37) for 3.3-V and 1.5-V digital supplies. Refer to "Switching Regulator Layout Example" on page 4–8 for details on the layout of these regulators on the Stratix GX development board.

To support a board with the characteristics listed earlier, the power circuitry must be able to do the following:

- Supply large amounts of current to the core and I/O voltages.
- Supply clean power and ground to sensitive analog components such as PLLs and transceivers.
- Maintain good efficiency to prevent excess heat dissipation on the regulators.

Voltage regulators come in two forms: linear and switching. *Linear* regulators are easy to design and provide very clean output voltage, but often they cannot provide large amounts of current without getting very hot. *Switching* regulators can provide large amounts of currents (over 10 A) with good efficiency, so there is no need for a heat sink. Switching regulators require very good design and layout practices to achieve good noise performance.

Use a switching regulator for the core supply, because current draw on the core can be high, and linear regulators cannot supply the necessary current without requiring a large heat sink. For example, if the core current needs to be 5 A, and the input and output voltages of the linear regulator are 3.3/1.5 volts, the heat dissipation would be approximately 5 * (3.3 – 1.5) = 9 Watts. This much heat requires a large heat sink on the linear regulator, which may not be possible to use on the board because of form factors, costs, or aesthetics.

For sensitive circuitry such as phase-locked loops (PLLs), transceivers, and double-data rate (DDR) memory, linear regulators are preferred. Choose the linear regulator based on the current requirements. Sometimes a small heat sink is required, depending on the current being drawn from the regulator and the voltage drop from the input to the output. Refer to *AN 185: Thermal Management Using Heat Sinks* for more details on regulator and heat sink selection.

If there are hundreds of LVTTL signals on the I/O, all of which can potentially switch at the same time, a switching regulator may be necessary, because the current requirement can be several Amperes.

Switching Regulator Layout Example

Linear regulators are relatively easy to design and lay out. Switching regulators, on the other hand, require a great deal of care. Typically schematics for both the linear and switching regulators can be taken directly from the vendor data sheet or the application note. Some of the linear regulators Altera recommends include Micrel MIC29502BU, National LMS1585 and National LP2995M. Similarly, Altera has used Fairchild FAN5066 for switching regulators with good results.

With switching regulators, component layout is extremely important. To explain this point further, consider an actual example. Figure 4–3 shows a capture plot of the FAN5066 circuit from the Stratix GX development board.

This switching regulator runs from a 5.0-V supply and can generate an output voltage that equals the voltage reference. Set the reference voltage on the board by applying the required voltage on the `CNTRL` pin (pin 16). The 12.0-V supply is used only for biasing and draws very little current. The overall circuitry consists of the FAN5066 switching controller, external FETs (Q3 and Q4) to boost the current output capability, and the associated resistors, diodes, inductors, and capacitors as recommended by the manufacturer.

The switching regulator is essentially a feedback loop consisting of a pulse width modulator (PWM). The feedback loop compares the actual output voltage to the desired voltage (reference voltage). If the actual output is less than desired, then the high drive (HIDRV) output is asserted, which turns the upper field effect transistor (FET) on and charges the output capacitor (C73). The end result is that the output voltage climbs. If the actual output is more than desired, the low drive (LODRV) output is asserted, which turns the lower FET on and discharges the output capacitor (C73). The end result is that the output voltage drops. In steady state the output voltage equals the input reference voltage. Figure 4–4 shows the schematic of the switching regulator used in the SGX development board.

Figure 4–3. Switching Regulator Schematic

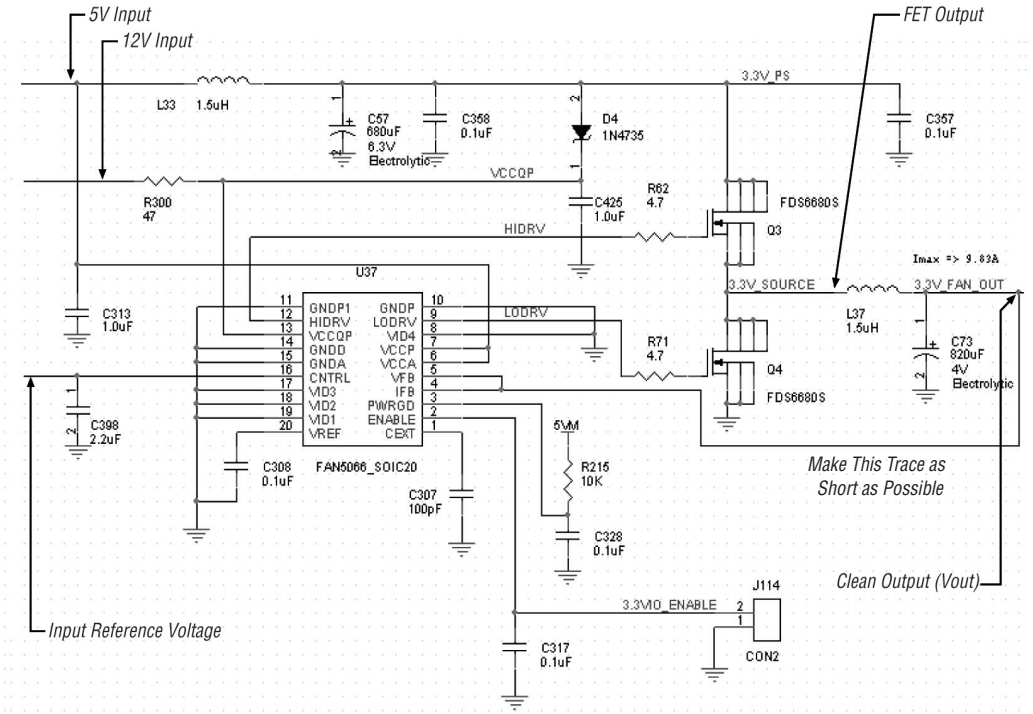
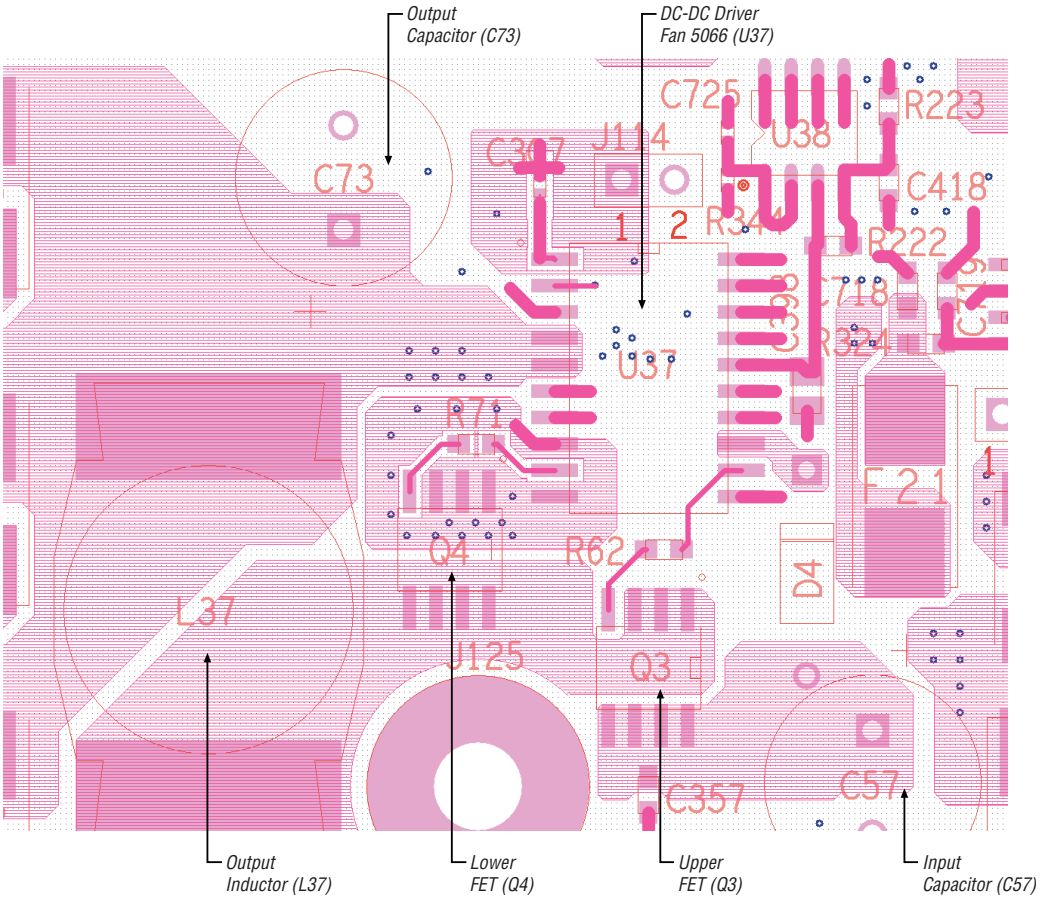


Figure 4–4. Switching Regulator Layout



Figures 4–5 through 4–7 show the waveforms at the gates of the upper and the lower FET. While the lower FET switches from about 0 V to 5 V, the upper FET switches from approximately 0 V to 10 V. These are large magnitudes on a board with 3.3-, 2.5-, or 1.5-volt logic signals. Routing fast switching signals close to sensitive analog and digital signals and across multiple layers can cause the switching signals to radiate energy to other critical traces. Consequently, it is very important to contain the large switching signals compactly on the same layer.

Figure 4–5. Switching Waveforms at the Output of the FET

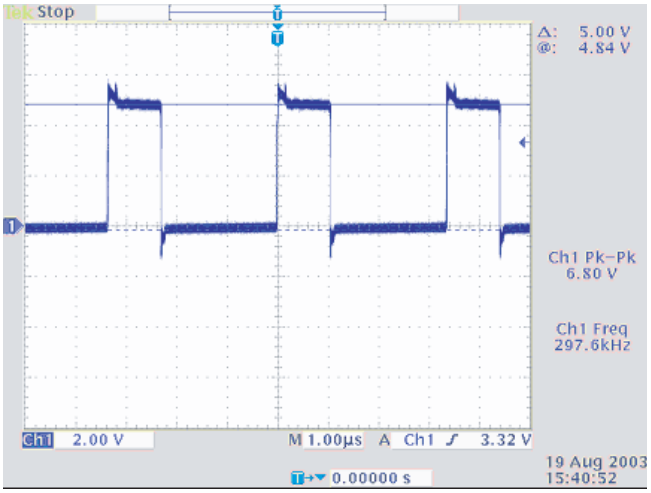


Figure 4–6. Switching Waveforms at the Gate of the Upper FET

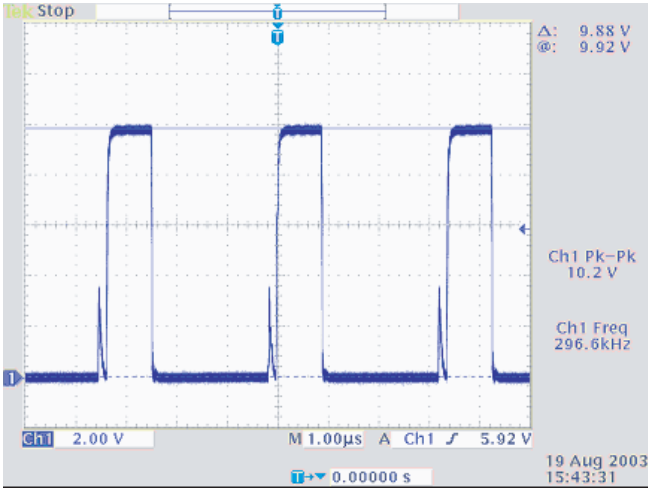
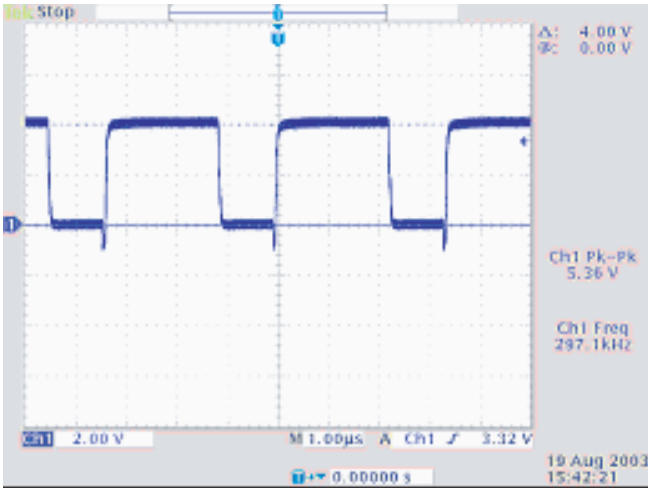
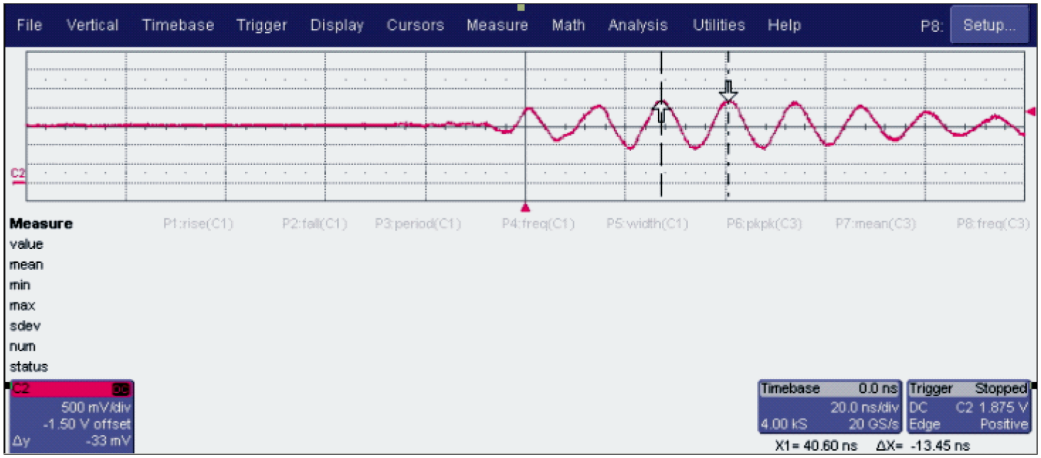


Figure 4–7. Switching Waveforms at the Gate of the Lower FET



Altera has compared the performance improvement of the output ripple when using a good layout versus a poor layout. The results of excessive ripple at the output of a switching regulator because of poor board layout, are shown in Figure 4–8. The output ripple is almost 1.5 V peak to peak. This example may be an extreme case, but the board in this example was designed with an intentionally poor layout. The layout was later improved by reducing the length of the HIDRV and LODRV signals and minimizing via count to determine how much improvement would result. With the improved layout, the noise decreased to about 71 mV peak to peak.

Figure 4–8. Excessive Ripple at the Output of a Switching Regulator



Altera recommends that you follow the switching regulator layout of the Stratix GX development board and review the Fairchild Semiconductor FAN5066 data sheet in detail. The guidelines are summarized here:

- Place all components on the same layer, if possible.
- Minimize the trace lengths for the gate drives (HIDRV and LODRV).
- Place all the 0.1 µF decoupling capacitors close to the pins.
- Place the HIDRV, LODRV, VCCQP and the FET output traces far away from the analog sections of the chip, including VCNTRL (pin 16), VFB, and IFB (feedback pins 4 and 5), and C_{EXT} (pin 1). Do all routing for the switching signals (VCCQP, HIDRV, LODRV, and FET output) on the component layer to avoid radiation to multiple layers through the vias.
- Surround the capacitor connected to pin 1 (C_{EXT}) with a ground guard trace and place a ground plane beneath it.

Decoupling Circuitry Design

You can divide decoupling capacitor circuitry into two categories: bulk decoupling and local decoupling. Bulk decoupling uses large decoupling capacitors (often tens to hundreds of microfarads), usually placed close to the regulators and designed to work for low frequencies. Local decoupling involves smaller capacitors (typically 2.2 µF, 0.1 µF, or 0.01 µF) that are placed close to the chips and are designed to decouple high frequency noise from the power supply.

Determining the number of decoupling capacitors depends on the expected current draw from the power supply, the rate of current change, and the desired impedance between the power plane and the ground plane at the frequency of interest. For switching (or sometimes even linear) regulators, the manufacturers specify a certain minimum bulk decoupling at the output of the regulator for stability and for ensuring a clean output.

Bulk Decoupling

The amount of bulk decoupling needed depends on the number of outputs switching, the load capacitance, the slew rate, and the inductance of the planes and routing traces and vias.

Estimate the amount of bulk decoupling needed as follows:

1. Determine the worst case current draw on the power net. This draw depends on how many gates can switch at the same time and how much load they need to drive.

   For example, assume there are 100 3.3-V LVTTL I/O pins switching, each with 11.5 pF loads, and the switch is completed in 1 ns. (These values are fairly typical for Stratix and Stratix GX devices.) The total worst case current required is:

$$\Delta I = NC\frac{\delta V}{\delta t} = 100 \times 11.5\,pF\,\frac{3.3V}{1ns} = 3.8\,Amps$$

2. Determine the maximum noise margin degradation allowed in the logic circuit. This value depends on the standard being used. For LVTTL, the threshold is shown in Table 4–1.

   The noise margin between V_{OH} and V_{IH} is 2.4 – 1.7 = 0.7 V. The margin between V_{OL} and V_{IL} is 0.7 – 0.45 = 0.25 V. Therefore, the worst case noise margin is 0.25 V, the absolute maximum value of noise allowed on the power supply while still guaranteeing successful data transfer. A good design allows for safety margin, so this example specifies 0.1 V as the maximum value of noise permitted. Call this parameter ΔV.

Table 4–1. LVTTL Parameters (Part 1 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{CCIO}	Output supply voltage		3.0	3.6	V
V_{IH}	High-level input voltage		1.7	4.1	V

Table 4–1. LVTTL Parameters (Part 2 of 2)

Symbol	Parameter	Conditions	Minimum	Maximum	Units
V_{IL}	Low-level input voltage		–0.5	0.7	V
V_{OH}	High-level output voltage	I_{OH} = –4 to –24 mA	2.4		V
V_{OL}	Low-level output voltage	I_{OH} = 4 to 24 mA		0.45	V

3. Using ΔV from step 2 and ΔI from step 1, determine the maximum impedance allowed on the power supply plane. In the example, it equals Z_{max} = ΔV/ΔI = 0.1/3.8 = 26.3 mW.

4. Next, determine the power supply wiring inductance. This inductance depends on the actual routing path of the power supply to the plane and the device pin. This path includes the vias, inductance of the balls to the actual connection on the die, inductance of the regulator output pin, and inductance of the actual routing from the regulator to the via. Approximate the inductance of the via with the following formula:

$$L_{via} = 5.08h[\ln(\frac{4h}{d}) + 1]$$

Assume that the height (h) of the via is 63 mils and that the diameter is 10 mils. Using these values in the above formula, the via inductance is 1.4 nH. These values are typical for vias on the Stratix GX development board for a signal traversing about 2/3 of the board thickness. Divide the via inductance by the number of vias that are in parallel. For 10 vias, the total via inductance is 0.14 nH. Other factors that add to this result are the ball-to-die connection and the inductance of the traces, which are harder to estimate and depend on the exact layout. Inductance of the plane is very small (less than 1 nH) if the physical dimensions are at least 1" by 1." You can lump all these inductances into a single approximation of 5 nH (L_{total} = 5 nH), which represents the worst case inductance on a typical system, assuming wide traces for power distribution.

Given the value for the power supply inductance, you can now estimate the frequency above which decoupling is required. Use this equation:

$$F_{critical} = \frac{Z_{\max}}{2\pi L_{tot}} = \frac{26.3\times10^{-3}}{2\pi 5\times10^{-9}} = 838kHz$$

5. Finally, use this equation to calculate the value of the decoupling capacitor:

$$C_{bypass} = \frac{1}{2\pi F_{critical} Z_{\max}} = \frac{1}{2.\pi.838\times10^{3}\times26.3\times10^{-3}} = 7.25\mu F$$

Consider the example on the Stratix GX development board. There are about 350 LVTTL signals on the Stratix device (on banks 1, 2, 3, 4, 7, and 8). The calculation above was for 100 LVTTL signals. For 350 signals, the required value of bulk decoupling increases by a factor of $(3.5)^2$, or 12.25. Therefore, bulk decoupling equals 7.25 * 12.25 = 89 µF. The development board uses a 100 µF capacitor on the 3.3-V power supply net (3.3V_S_IO) for bulk decoupling for the Stratix device. This is in addition to the bulk decoupling required by the switching regulators and the associated filters at the output. With this level of decoupling, there is good performance.

Local Decoupling

Local decoupling is designed to work at high frequencies. Unfortunately, at high frequencies, discrete capacitors are rarely effective by themselves. There must be a combination of discrete capacitors and plane capacitance. The analysis is quite involved and its accuracy is not easy to verify. So, experimental data is more helpful for creating design guidelines. Altera has conducted several experiments on this topic. Based on these experiments, Altera recommends about one decoupling capacitor per power pin. The results of simulations using Agilent design system (ADS) software determined the values of capacitors required.

The amount of local decoupling is more difficult to determine, because high frequency effects are more pronounced for local decoupling. Altera recommends using one capacitor per power pin. More detailed guidelines for specific situations will be available in the near future. You should not mix values of capacitors, because doing so can create resonant peaks of impedances. In most cases it is best to choose the highest value of the capacitance available in the package required by the designer.

Simulating Decoupling Circuits

The efficacy of a decoupling scheme is indicated by the impedance between the power plane and the ground plane across the frequency of interest. The goal of a power supply decoupling scheme is to achieve the lowest possible impedance across the frequency of interest. If the impedance is low, then the noise on the power rail is shunted to the ground, whereas if the impedance is high, the noise stays on the power rail and can cause the circuitry to malfunction.

In this simulation, Altera placed 25 decoupling capacitors of different values in parallel and measured the impedance from 1 MHz to 10 GHz. The frequency of interest can be different for each application, so focus on the frequency band of interest for your application. Using 25 decoupling capacitors gives a realistic number of capacitors used in a typical decoupling scheme. The behavior of vias, pads, and traces were captured using the multilayer transmission line models of ADS, which considers the dielectric material and its dissipative losses.

Eight different cases, summarized in Table 4–2, were simulated to get a wide range of capacitor ratios.

Table 4–2. Simulation Cases

Case	Number of Capacitors					Waveform Name
	470 pF	1000 pF	0.01 µF	0.1 µF	2.2µF	
1	10	8	4	2	1	Z_double_decade
2	5	5	5	5	5	Z_equal_ratio
3	1	2	4	8	10	Z_reverse_decade
4	25	0	0	0	0	Z_all_470pf_caps
5	0	25	0	0	0	Z_all_1000pf_caps
6	0	0	25	0	0	Z_all_point_01_caps
7	0	0	0	25	0	Z_all_point_1_caps
8	0	0	0	0	25	Z_all_2_point_2_caps

Figure 4–9 shows the simulation setup with only one capacitor (only one capacitor is shown for clarity), including the parasitics, pads, routing trace to the vias, and the vias themselves. For the cases shown in Table 4–2, the model of the capacitor shown in Figure 4–9 was applied repeatedly in parallel.

Each capacitor is modeled as a series RLC circuit. The parasitic resistance (R) and the parasitic inductance (L) were obtained from the data sheet published by the manufacturer of the capacitor. The capacitor pads on the board were modeled as 42-mil by 32-mil square pads. The vias were modeled as circular vias with pad diameters of 20 mils and hole diameters of 10 mils. The routing distance from the capacitor pad to the via was modeled as a trace, 50 mils long and 25 mils wide, equal to the total distance for routing on both sides of the capacitor. These are typical values and were extracted from the Stratix GX development board layout design.

Figure 4–9. Simulation Setup for One Capacitor

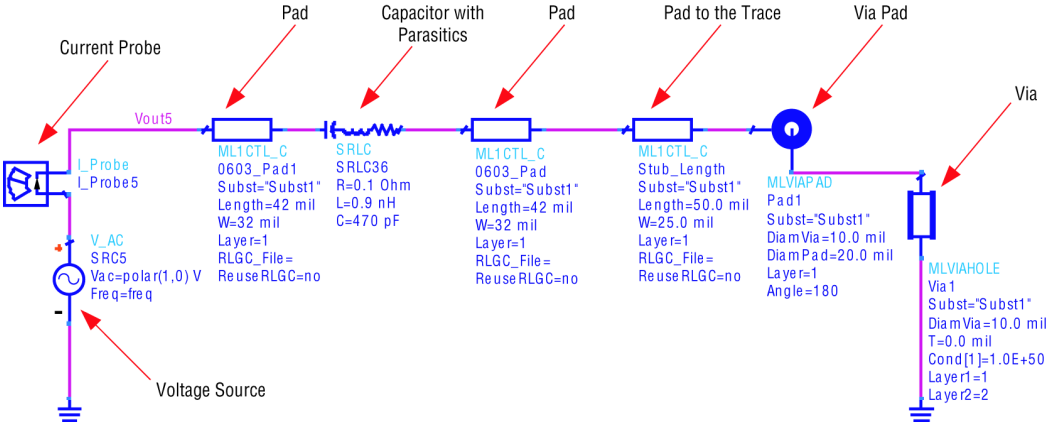


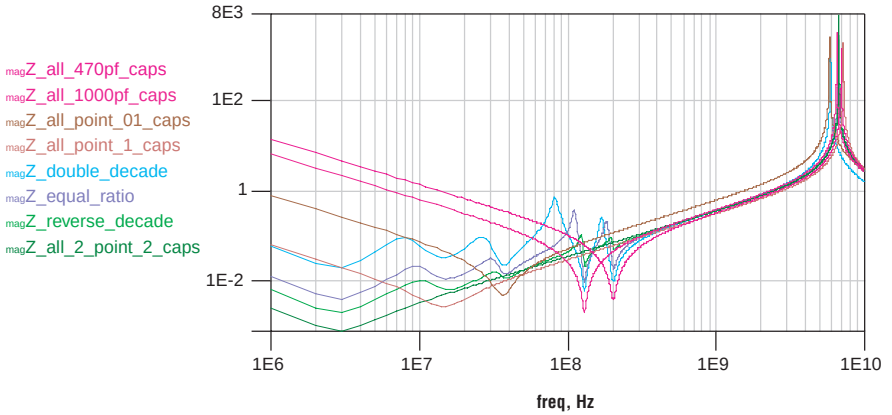
Table 4–3 shows the extracted models of 470 pF, 1000 pF, 0.01 µF, 0.1 µF, and 2.2 µF capacitors from the vendor datasheets (Panasonic and AVX).

Table 4–3. Extracted Models of Capacitor Parasitics

Capacitor Value	Part Number	Vendor	Parasitic L (nH)	Parasitic R (Ω)	SRF (MHz)
0.1 µF	ECJ1VB1C104K	Panasonic	1.50	0.07	13
0.01 µF	ECJ1VB1C103K	Panasonic	0.83	0.10	55
2.2 µF	08056D225KAT2A	AVX/Kyocera	1.00	0.02	3.4
470 pF	ECJ-0EB1E471K	Panasonic	0.86	0.10	250
0.001 µF	ECJ-1VB1H102K	Panasonic	1.13	0.05	150

Figure 4–10 shows the simulated impedance profile for the eight cases listed in Table 4–2.

Figure 4–10. Impedance Simulation Results From Table 4–2



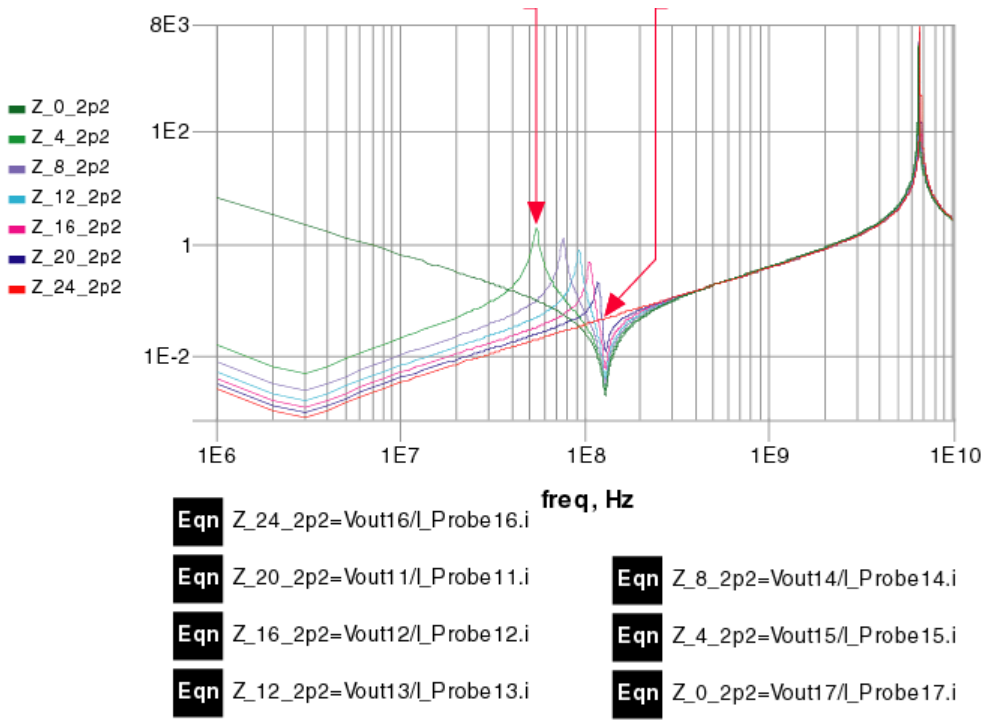
The following observations were made:

- Decoupling at very high frequencies cannot be achieved by discrete capacitors. It must be done with a sandwiched power and ground plane.
- Traditionally, the number of capacitors is doubled for every lower decade in value. For example, beginning with one 0.1 µF capacitor, you would use two 0.01 µF capacitors. Case 1 in Table 4–2 on page 4–17 approximates this behavior. However, this approach often performs worse than when using equal numbers of capacitors of equal values, or even reversing the ratio (number of capacitors halved for each decade lower in value).
- Using a very large number of values can cause resonance spikes at multiple frequencies, so use as few values as possible to contain the magnitude of the resonance spike.

- Using all 2.2 µF capacitors gives the lowest impedance profile without any resonance spikes. Although 1000 pF capacitors (when used in conjunction with 2.2 µF caps) do give the low impedance dip at higher frequencies, they also result in the corresponding resonance spike.

To study how the ratio of 2.2 µF versus 1000 pF capacitors affects the impedance profile, Altera performed further simulations. Figure 4–11 shows the impedance profiles when using all 24 2.2-µF caps versus using a mix of 2.2 µF and 1000 pF. Only 24 capacitors were used in this simulation as opposed to 25, allowing for easy ratios. This difference is not significant. The lower red graph in Figure 4–11 shows the case when all 24 capacitors are 2.2 µF. It has the smoothest impedance profile and is lower in impedance than other cases, except for a dip at 129 MHz. If a particular board has significant noise at the frequency of the dip (approximately 129 MHz), a 1000 pF capacitor is helpful. However, you must ensure that there is negligible noise at the resonant peaks. The resonant peaks occur at 54 MHz, 75 MHz, 92 MHz, 106 MHz, and 118 MHz, depending on the ratio of 1000 pF caps to 2.2 µF caps.

Figure 4–11. Impedance Profile With a Combination of 2.2 µF & 0.001 µF Capacitors



Based on the simulation results, it is important to select the highest value possible in a particular package, because these capacitors have the lowest impedance for the largest band of frequency. Examples include 2.2 µF, 1.0 µF, and 0.1 µF. Do not go lower than that unless there is a compelling reason to do so, such as the presence of a strong noise spike at the exact frequency where the lower value capacitor resonates. Also, do not mix capacitor values (for example using 10 of 0.1 µF and 10 of 0.001 µF). This mixing can create undesired resonances.

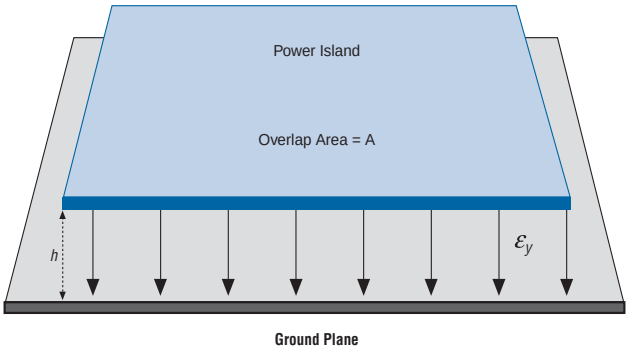
The Stratix GX development board uses many mixed capacitor values for decoupling, not because it is the best approach but because it allows the flexibility to test several approaches and values.

Plane Capacitance

For very high frequencies (above 300 MHz), decoupling using discrete capacitors is less effective. Use power plane capacitance for decoupling noise at these frequencies.

You can understand the concept of plane capacitance by studying the classic parallel plate capacitor, shown in Figure 4–12.

Figure 4–12. Parallel Plane Capacitance



An electric field is created when there is a power plane next to a ground plane. The upper area in Figure 4–12 shows the power island or plane, the lower area shows the ground plane, and the arrows represent the electric field lines. This electric field gives rise to a capacitance, the magnitude of which is shown by:

$$C = \frac{\varepsilon_0 \varepsilon_r A}{h}$$

where

ε_0 = permittivity of free space

ε_r = relative permittivity of the dielectric used

A = area of overlap

h = separation of the planes.

If there are ground planes on both sides of the power island, then the capacitance needs to be calculated for each side and added to determine the total capacitance.

Plane capacitance is the primary way of decoupling at high frequencies, so it must be an integral part of any high speed design. At high frequencies (above 300 MHz), the discrete capacitors are not very effective.

As an example, consider the following.

Example: Determine the parallel plate capacitance for 1 square inch of area overlap in an FR-4 dielectric (ε_r = 4.5) and a separation of 4 mils.

Solution:

h = 4 mils = $1.016 * 10^{-4}$ m

ε_0 = permittivity of free space = $8.85 * 10^{-12}$ F/m

A = 1 sq inch = $6.4516 * 10^{-4}$ m^2

ε_r = 4.5

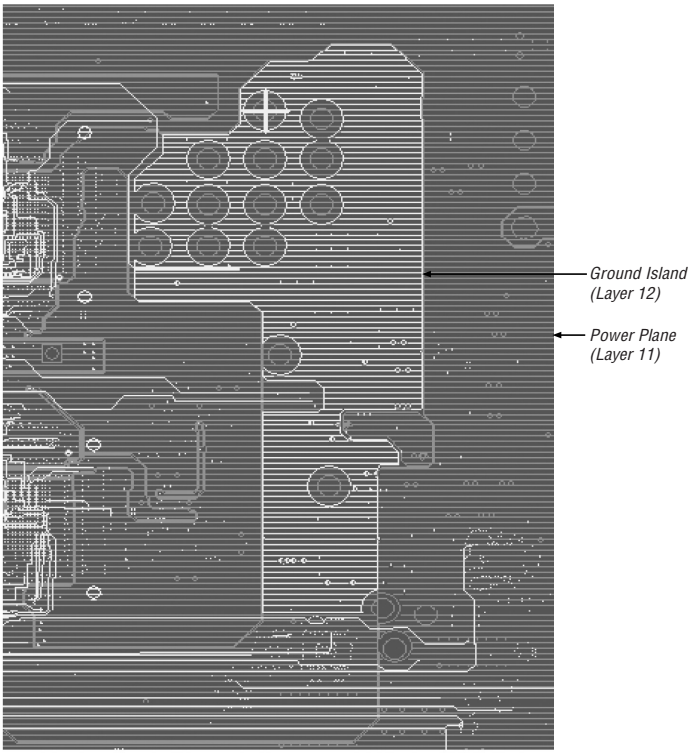
Applying these numbers to the equation on page 4–21 yields C = 253 pF. Therefore, there is about 253 pF per square inch of area overlap on a typical FR-4 board with 4 mils of separation. The value scales inverse linearly with separation and linearly with area.

Altera has successfully used plane capacitance in several of its boards.

Case Study: Altera designed two boards with the Stratix GX test chip. On the first board, the separation between the ground and the 1.5-V transceiver power plane was about 16 mils. On the second board, ground

islands were added (with approximate areas of 0.13 square inches) next to the 1.5 V transceiver plane. The islands were on the next layer and the separation was 4 mils. Figure 4–13 shows the islands as they appear on the board layout.

Figure 4–13. Ground Island Used With the 1.5V_XCVR Plane to Increase Decoupling at High Frequencies



Figures 4–14 and 4–15 show the reduction in noise after adding the islands. The addition of the islands reduced the noise from 70 mV p-p to less than 50 mV p-p under the same experimental conditions.

Figure 4–14. Noise on the 1.5V_XCVR Plane With the Ground Plane 16 mils Away

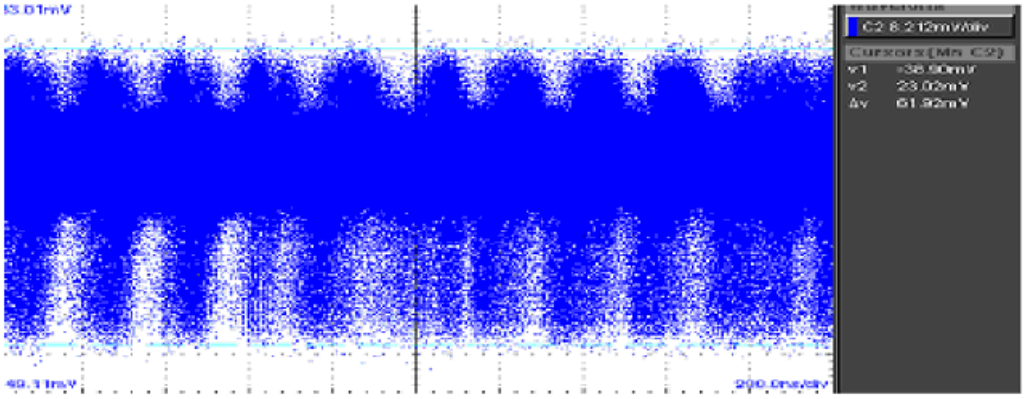
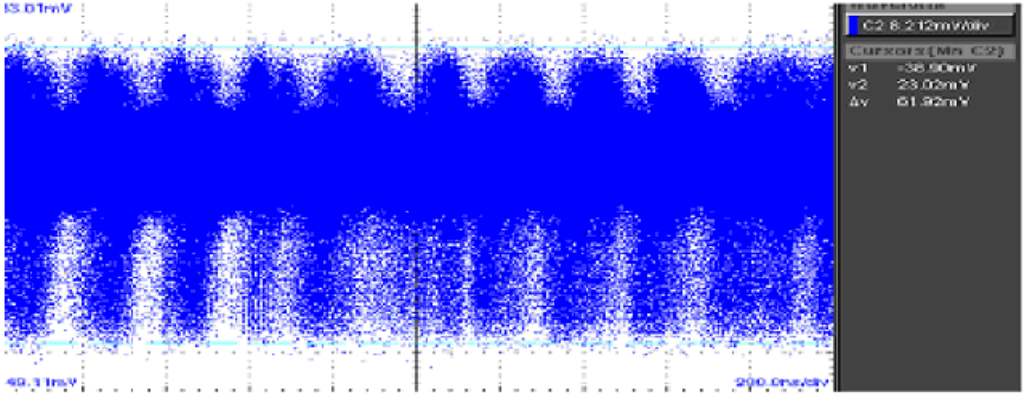


Figure 4–15. Noise on the 1.5V_XCVR Plane With the 0.13 Square Inch Island 4 mils Away



When using split ground planes, it is best to pair the correct ground plane with the correct power plane. For example, 1.5V_XCVR and GND_GXB planes and islands should be used rather than 1.5V_XCVR and GND.

Plane & Island Design

The following sections discuss various issues related to the design of power planes, ground planes, and power and ground islands.

Power Plane & Island Design

Stratix GX devices have many power and ground pins, and it is important to know how to connect them on the board. Table 4–4 shows the list of power and ground pins and how they connect on the Stratix GX development board. For the actual pin numbers, refer to the pin tables at **www.altera.com**.

Table 4–4. Stratix GX Power & Ground Pins (Part 1 of 2)

Pin Name	Description	Voltage
VCCP_B[17..13]	Digital power for Stratix GX transceiver quads (banks) 13 through 17	1.5 V (linear)
VCCT_B[17..13]	Transmitter power for Stratix GX transceiver quads (banks) 13 through 17	1.5 V (linear)
VCCR_B[17..13]	Receiver power for Stratix GX transceiver quads (banks) 13 through 17	1.5 V (linear)
VCCG_B[17..13]	Power for Stratix GX transceiver quads (banks) 13 through 17 guard rings.	1.5 V (linear)
VCCA_B[17..13]	Analog power for Stratix GX transceiver quads (banks) 13 through 17.	3.3 V (linear)
VCCG_PLL[12,11,8,6,5,2,1]	Power supply for the guard ring of PLLs	1.5 V (linear)
VCCA_PLL[12,11,8,6,5,2,1]	Analog power supply for PLLs	1.5 V (linear)
VCC_PLL5_OUTA	PLL output buffer supply for PLL5 outputs [1..0]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCC_PLL5_OUTB	PLL output buffer supply for PLL5 outputs [3..2]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCC_PLL6_OUTA	PLL output buffer supply for PLL6 outputs [1..0]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCC_PLL6_OUTB	PLL output buffer supply for PLL6 outputs [3..2]	3.3 V, 2.5 V or 1.5 V *(1)* (linear)
VCCINT	Digital power supply for device internal power	1.5 V (linear or switching)
VCCIO[8..7, 4..1]	Power supply for I/Os in the banks 8, 7, 4..1	Variable *(2)* (linear or switching)

Table 4–4. Stratix GX Power & Ground Pins (Part 2 of 2)		
Pin Name	**Description**	**Voltage**
GNDG_PLL[12,11,8,5,2,1]	Ground for the guard rings of PLLs 1, 2, 5, 6, 8, 11, and 12	0 V
GNDA_PLL[12,11,8,6,5,2,1]	Analog ground for PLLs 1, 2, 5, 6, 8, 11, and 12	0 V
GND_GXB	Ground for the high-speed circuitry in the 3.125 Gbps transceivers	0 V
GND	General-purpose ground	0 V

Notes for Table 4–4:

(1) Connect this pin to the desired voltage depending on the I/O standard for the PLL outputs chosen. For example, in the Stratix GX development board, VCC_PLL6_OUTA and VCC_PLL6_OUTB connect to 2.5 V to allow an SSTL-II clock (2.5V I/O) on the PLL6 output for the double data rate (DDR) memory application.

(2) Connect this pin to the I/O standard of the signals on the bank. This value is 3.3 V for LVTTL, 2.5 V for SSTL-II, and so on. On the Stratix GX development board, VCCIO7 connects to 2.5 V because the board uses the SSTL_II I/O standard on that bank. On other banks, it connects to 3.3 V.

Table 4–4 also shows the recommended regulator (linear or switching). The circuitry pertaining to PLLs and transceivers should be connected to voltages generated from linear power supplies. General purpose I/O and core supplies can be switching or linear. It is always better to use linear if possible, but for some I/Os and cores, the current draw might be too large.

The transceivers have many types of power supply pins. Altera recommends that you isolate the receive (VCCR_B[17..13]) and transmit (VCCT_B[17..13]) power supplies of each quad with a ferrite bead. The reason behind the isolation is to prevent noise from one quad leaking to the other quads. The digital power supply (VCCP_B[17..13]) can be shared among all the quads because it is less sensitive to noise. The guard power supply VCCG_B[17..13] can also be shared among quads. Use the islands shown in Table 4–5:

Table 4–5. Islands Used for Stratix GX Transceiver Quads (Part 1 of 2)	
Islands	**Quads**
Island 1	Use for quad 1 VCCT and VCCR pins (VCCT_B13 and VCCR_B13)
Island 2	Use for quad 2 VCCT and VCCR pins (VCCT_B14 and VCCR_B14)
Island 3	Use for quad 3 VCCT and VCCR pins (VCCT_B15 and VCCR_B15)

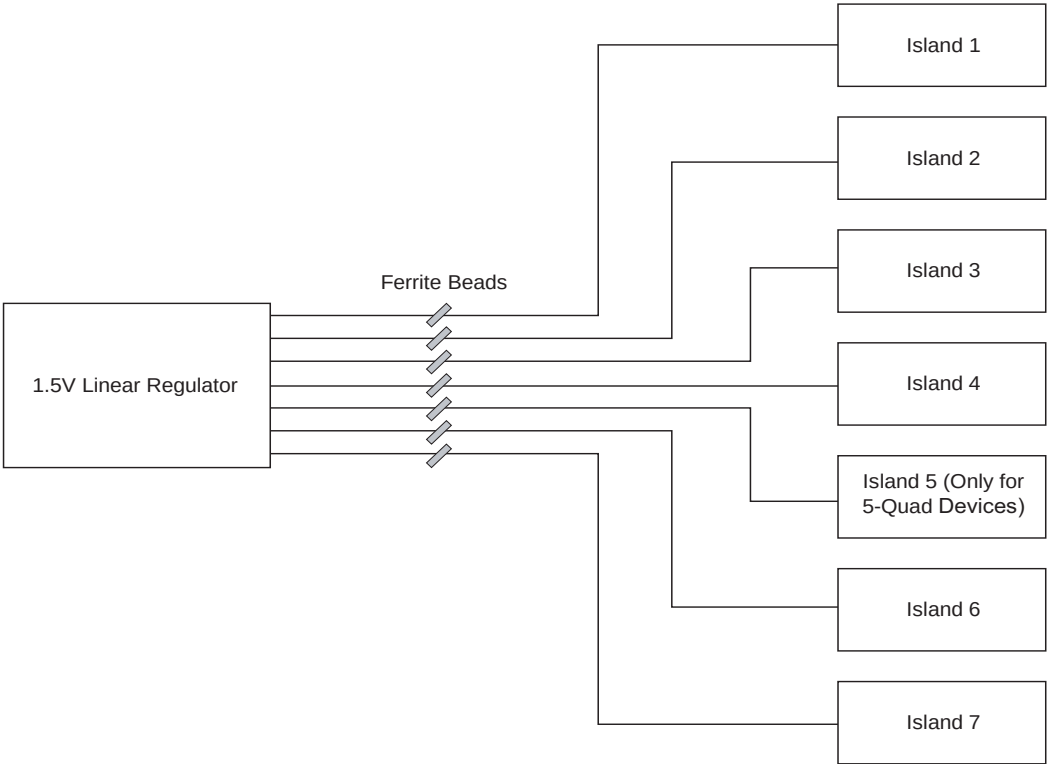
Table 4–5. Islands Used for Stratix GX Transceiver Quads (Part 2 of 2)

Islands	Quads
Island 4	Use for quad 4 VCCT and VCCR pins (VCCT_B16 and VCCR_B16)
Island 5	Use for quad 5 VCCT and VCCR pins (VCCT_B17 and VCCR_B17)
Island 6	Use for `VCCP_B[17..13]`, for example, digital supply for all quads
Island 7	Use for `VCCG_B[17..13]`, for example, guard supply for all quads

Note to Table 4–5:
(1) Island 5 is required only for the 5-quad devices.

Figure 4–16 shows a block diagram of the islands.

Figure 4–16. Block Diagram Representation for Quad Isolation



Note to Figure 4–16:
(1) Island 5 is required only for the 5-quad devices.

Altera has successfully used the island approach to powering the Stratix GX transceivers on the Stratix GX development board. Figure 4–17 shows the section of the schematic that shows the islands. The net named 1.5V_XCVR is the output of a linear regulator that is split into seven islands, namely: 1.5V_XCVR1, 1.5V_XCVR2, 1.5V_XCVR3, 1.5V_XCVR4, 1.5V_XCVR5, 1.5V_VCCP, and 1.5V_VCCG.

Altera is currently evaluating the degree to which the islands help in reducing jitter.

Figure 4–17. Section of Schematic Showing the Islands for the Transceiver Quads

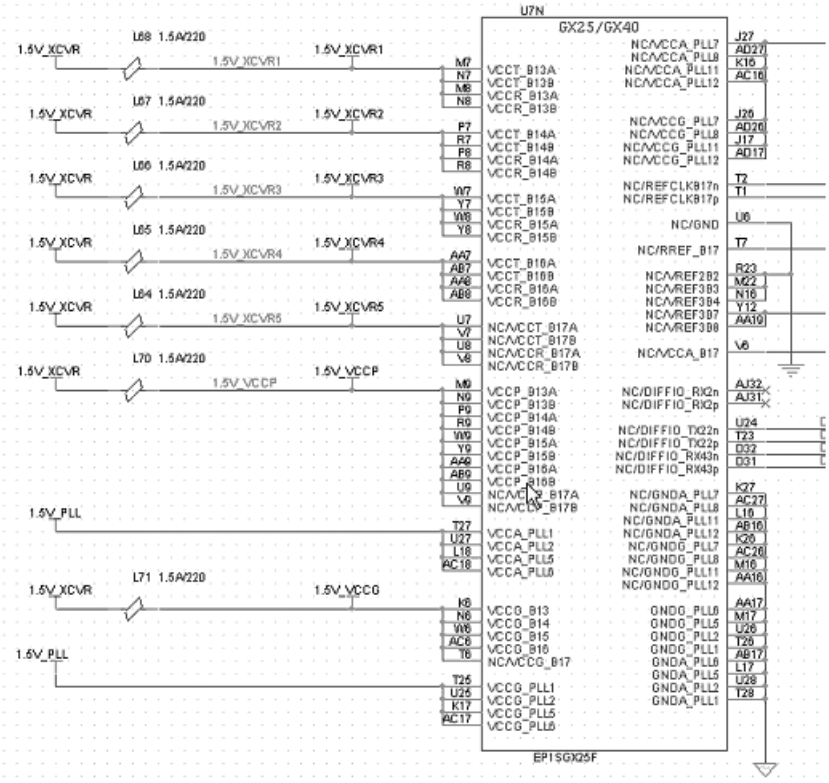
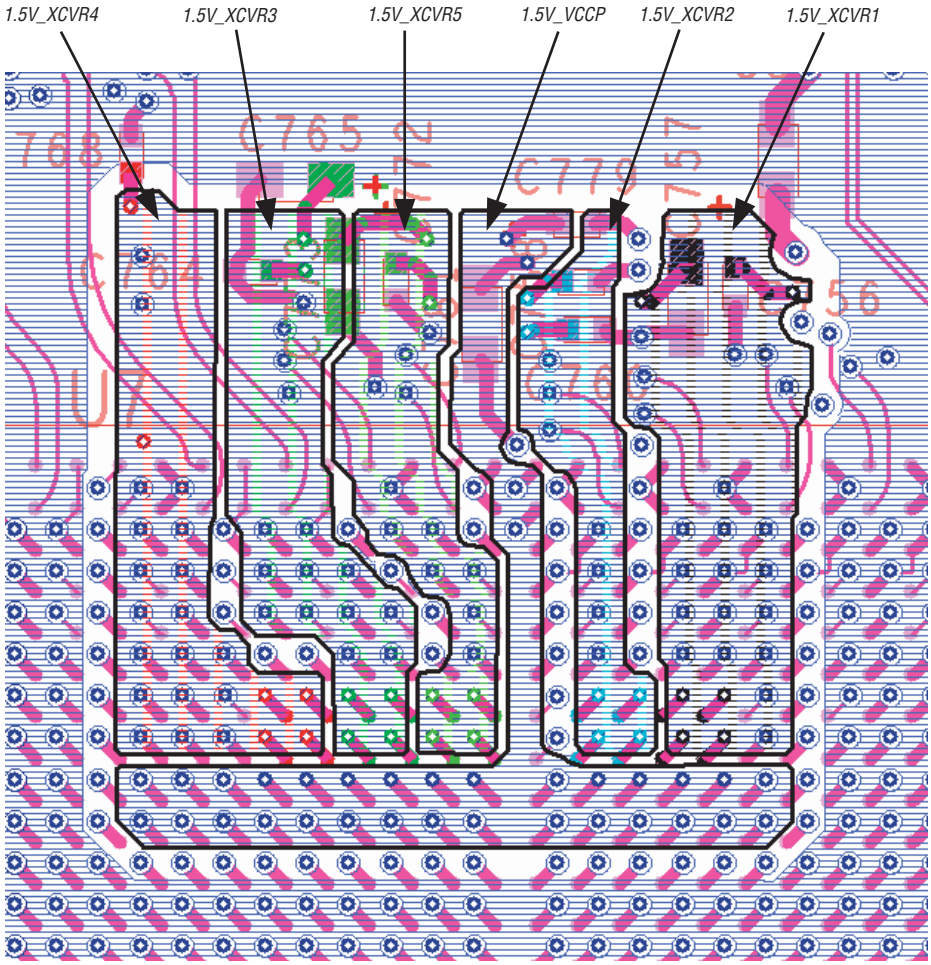


Figure 4–18 shows the layout of six of the islands. The 1.5V_VCCG (guard) island is on a different layer because of layout constraints.

Figure 4–18. Layout of the Islands for the Transceiver Quads



Ground Plane & Island Design

The Stratix GX development board has three types of ground pins: GND, GND_GXB and GND_PLL. GND_GXB is used for 3.125 Gbps transceiver ground, GND_PLL is used for PLL ground, and GND is used for general purpose ground on the board. In the die, GND_GXB and GND are separate planes, but they are connected by a trace at the package level. GND_PLL does not have a separate plane and is connected to the GND plane at the package level.

When designing your board you must decide whether to connect all the ground pins to a single ground plane on the board or create separate ground planes for GND_GXB, GND, and GND_PLL and connect them by using ferrite beads. This is a difficult trade-off, and there are two schools of thought in the industry regarding splitting ground planes.

Proponents of isolating ground planes say that it is good for systems that can generate a lot of noise because of switching I/Os or power supplies. The idea is that the ferrite beads attenuate the noise leaking from the system ground (which can be noisy) to the clean ground. Noise-sensitive analog subsystems like the transceivers require clean power and ground, and the isolation provides better performance. The ferrite beads must be carefully selected to allow for plenty of current handling, low DC voltage loss, and high AC impedance. See "Resistors, Capacitors, Inductors & Ferrite Beads" on page 4–77 for more details.

Supporters of the solid ground plane approach claim that there is no benefit to isolating ground planes, because noise cannot simply disappear from the ground plane and reaches the analog circuitry by some path or other. The layout is also complicated by having isolated ground planes because more layers might be needed. If the same plane is split into two, instead of adding a new layer, the layout designer must ensure that no critical signals cross over the split, because the split causes impedance discontinuity.

There is no one answer on this topic that applies to all systems. If the system is relatively simple without too many fast switching I/Os, and if there are no switching regulators, a solid ground plane is unlikely to cause problems. However, if the system has many fast, high-amplitude digital signals switching and noisy external components such as switching regulators, Altera recommends isolating ground planes.

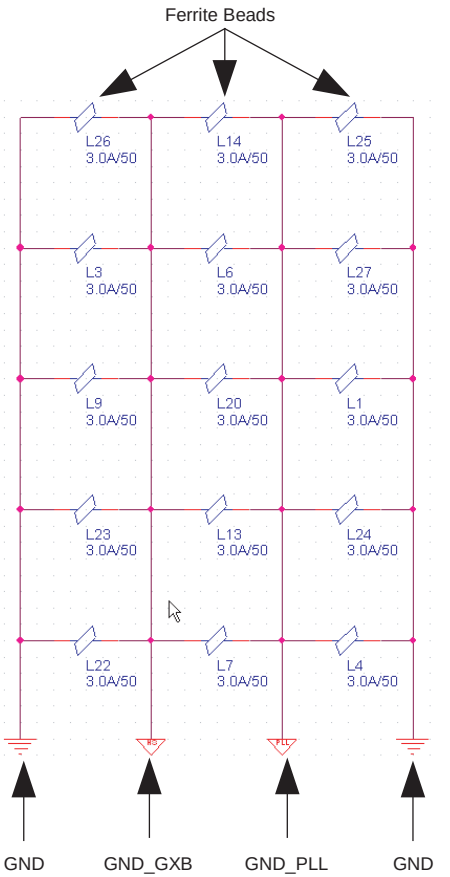
The Stratix GX development board uses an isolated ground approach. There are three distinct ground nets: GND, GND_PLL, and GND_GXB, separated by several ferrite beads. Figure 4–19 shows a section of the Stratix GX development board schematic highlighting the isolation of the ground planes. Many ferrite beads were used in parallel to facilitate testing with several possible combinations.

Altera tested the performance of this system with the ferrite beads intact, as shown in Figure 4–19, and with them shorted to a wire. The Stratix GX EP1SGX40 device had about 72% of its LEs used with a design that had Gray code counters. The Stratix GX device had a DDR interface running at 200 MHz and 17 channels of SPI-4.2 data running at 1 Gbps per channel. The LE usage was about 46%. The Stratix GX device also had 19 channels of transceivers running at PRBS 2^5 - 1 data at 3.125 Gbps in an external loopback. The transmitter output of the 20th channel was sent to

an oscilloscope to observe the jitter and the eye diagram. This setup should result in a fairly heavy load for the devices and should generate a good deal of noise on the power planes. The test setup was identical for both tests, except for the shorted ferrite beads.

The tests showed a jitter improvement of approximately 10% when isolating the ground planes. If the devices are loaded even more fully, the improvement is even greater. If the devices are lightly loaded, the improvement is less.

Figure 4–19. Isolation of GND, GND_GXB & GND_PLL



Based on the test results and the benefit of isolating ground planes for noisy boards, Altera® recommends isolating the GND_GXB and GND planes with ferrite beads. To determine how many ferrite beads to use,

consider the following. The power consumption per quad is about 450 mW, so the current draw is about 450 mW/1.5 = 300 mA. Therefore, the total expected current draw is about 1.5 Amps. Each ferrite bead has 25 mW of DC resistance. So, if there is one ferrite bead, the voltage drop is 37.5 mV, which is 2.5% of the 1.5-V supply. To reduce this drop, use multiple ferrite beads in parallel. On the Stratix GX development board, there are five beads in parallel, so that the voltage drop is reduced to 0.5% of the supply voltage.

For the GND_PLL plane, it is not crucial to use a separate ground plane. At the device package level, this plane shares the ground plane with the general-purpose logic ground. There is a separate plane on the development board for GND_PLL for testing flexibility.

Altera recommends two ground planes, GND and GND_GXB, isolated by ferrite beads. These planes are especially important for complex boards with many noisy signals and switching regulators. If the board does not have switching regulators or switching signals, and it is otherwise very cleanly designed, one ground plane is acceptable. Altera recommends connecting the PLL ground pins directly to the system GND.

Transmission Lines

This section discusses transmission line designs for high-speed digital boards.

Transmission Line Topologies

This section provides a brief overview of the main transmission line topologies commonly used in high-speed digital boards. Refer to any transmission line design book, such as Brian Wadell's *Transmission Line Design Handbook*, for impedance equations and design techniques.

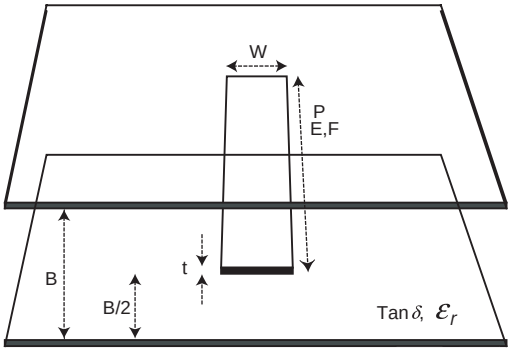
Striplines

The following sections detail the variations on stripline design topologies.

Simple Stripline

Simple stripline is a planar type transmission line well suited for multilayer PCB design. Figure 4–20 shows the basic structure of this transmission line. A thin, centered conductive strip with width W placed between two conductive planes separated by a dielectric with thickness B is the most basic stripline structure.

Figure 4–20. Simple Stripline



Figures 4–21 and 4–22 show the frequency domain simulation setup and results for a 50 Ω simple stripline, respectively.

Figure 4–21. Simulation Setup for a Simple Stripline

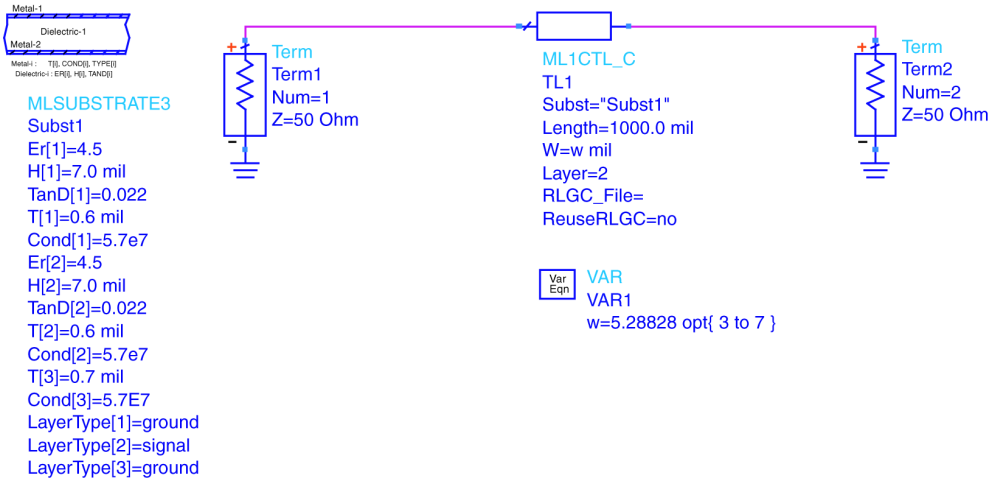
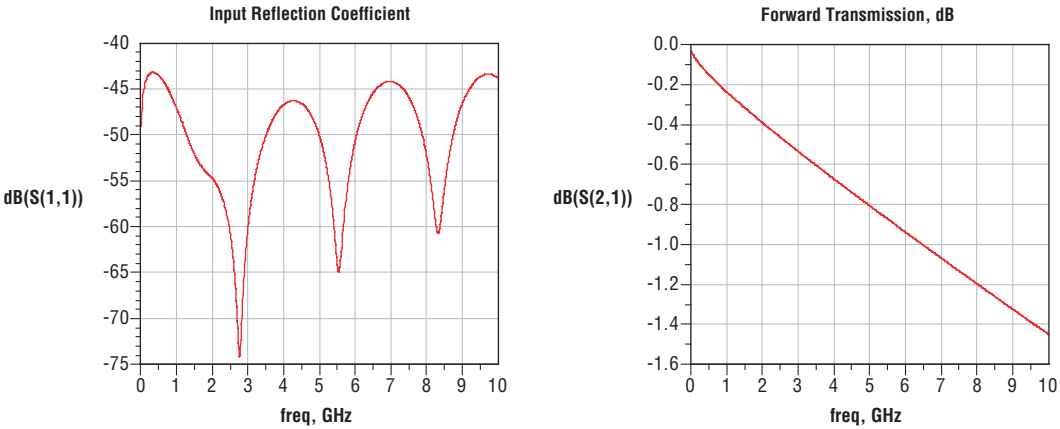


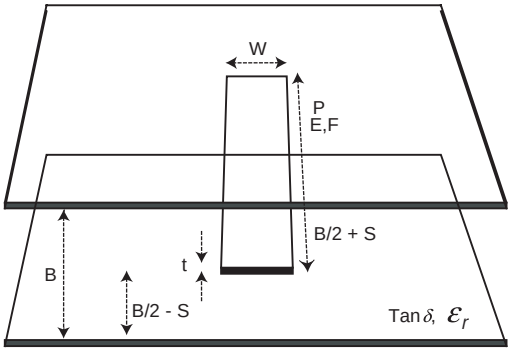
Figure 4–22. Simulation Results for a Simple Stripline



Offset Stripline

The offset stripline configuration is shown in Figure 4–23. The only difference between the structure of the offset stripline and the normal stripline is that the conductor strip is not placed at the center (B/2 distance from any of the planes). The relative position of the conductor strip is S from B/2. This transmission line is useful for single-ended signals routed between a ground plane and a power plane. The strip is placed closer to the ground plane, which is less noisy than the power plane. In this way there is less noise power coupled to the signal and an improved S/N ratio without the need for an extra ground plane. You must use this structure when the dielectric between the planes is constructed of three layers of dielectrics (two cores, one prepeg, or vice versa). In the Stratix GX development board there are two cores and one prepeg. This arrangement of dielectrics produces better results because the cores have lower permittivity and thickness tolerances than prepegs.

Figure 4–23. Offset Stripline



Figures 4–24 and 4–25 show the frequency domain simulation setup and results for a 50 Ω offset stripline, respectively.

Figure 4–24. Simulation Setup for an Offset Stripline

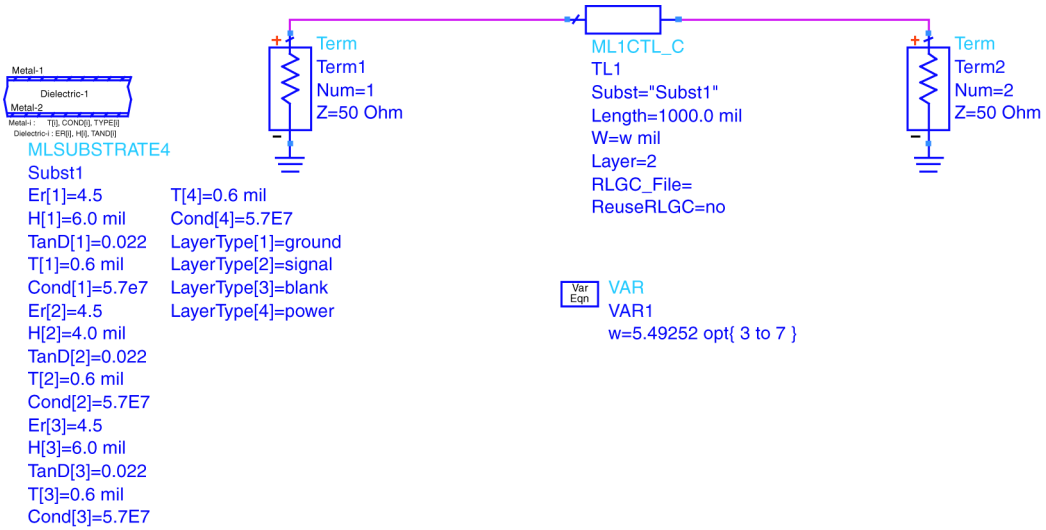
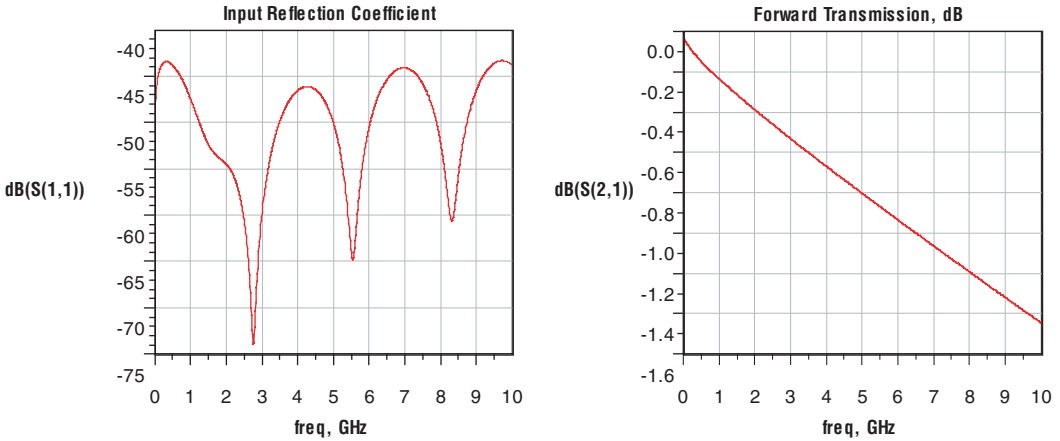


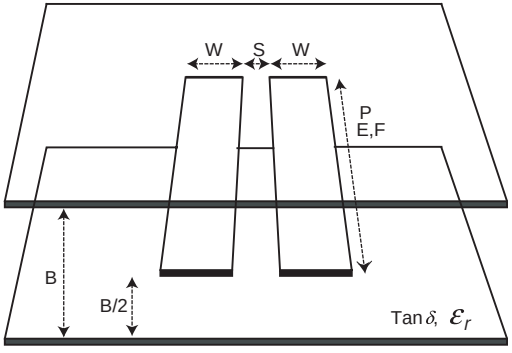
Figure 4–25. Simulation Results for an Offset Stripline



Edge-Coupled Differential Stripline

The edge-coupled differential stripline is shown in Figure 4–26. Altera recommends using differential transmission lines in noisy environments where the common mode noise is suppressed in the receiver. This type of transmission line requires some degree of coupling between the two conductive strips. The coupling is directly proportional to the distance S. The differential transmission lines have two operation modes. Even mode, in which both traces are excited with equal amplitude and phase, and odd mode, in which the traces are excited with equal amplitude but one is 180° out of phase. This structure must be analyzed as one unit and with the proper excitation. For differential signaling, the odd mode is the desired operation mode for canceling common mode noise.

Figure 4–26. Edge-Coupled Differential Stripline



Figures 4–27 and 4–28 show the frequency domain simulation setup and results for a 100-Ω edge-coupled differential stripline, respectively.

The differential impedance is:

$$Z_{DIFF} = 2\ (Zoo)$$

If you excite one port and terminate all others (three) with its characteristic impedance *Zo* (50 Ω) the input impedance is:

$$Zo = \sqrt{Zoe * Zoo}$$

where:

Z_{oe} is the even mode characteristic impedance

Z_{oo} is the odd mode characteristic impedance

Z_{DIFF} is the differential characteristic impedance

Z_o is the characteristic impedance

Figure 4–27. Simulation Setup for an Edge-Coupled Differential Stripline

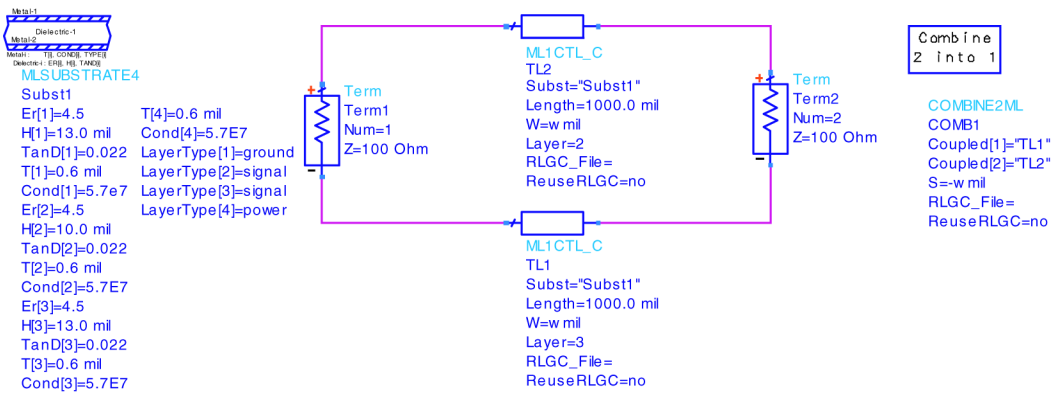
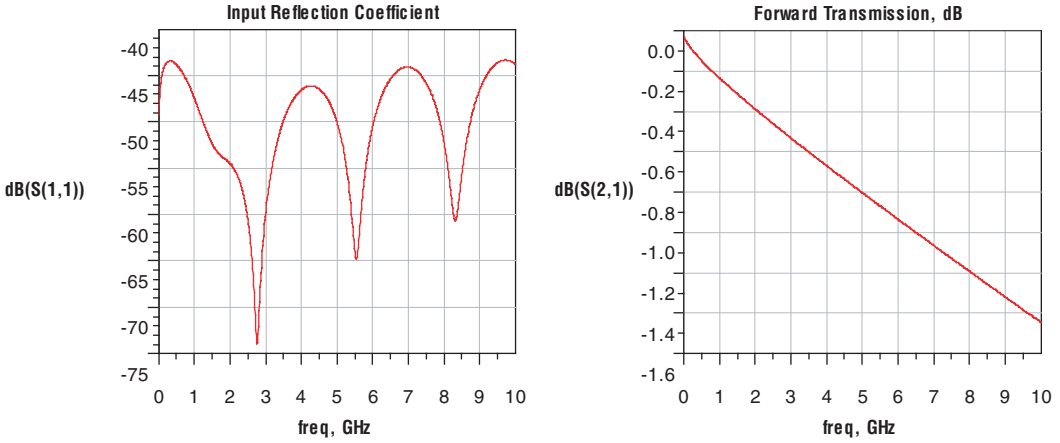


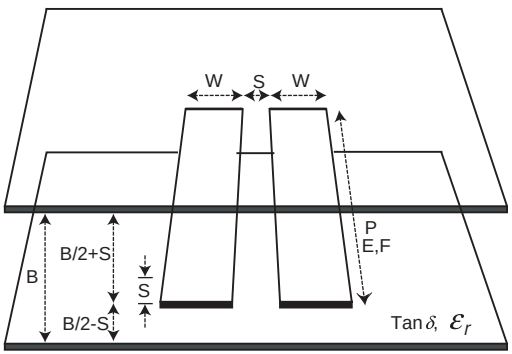
Figure 4–28. Simulation Results for an Edge-Coupled Differential Stripline



Edge-Coupled Differential Offset Stripline

The edge-coupled differential offset stripline topology, shown in Figure 4–29, is used when an odd number of dielectric layers exists between the planes. The signals in this dielectric arrangement are routed closer to one of the planes. This transmission line offers all the advantages of the differential transmission lines, plus the flexibility to use wider traces without the penalty on the total board thickness.

Figure 4–29. Edge-Coupled Differential Offset Stripline



Figures 4–30 and 4–31 show the frequency domain simulation setup and results for a 100-Ω edge-coupled differential offset stripline, respectively. The Stratix GX development board contains many edge-coupled

differential striplines. Examples from the board are discussed in "Crosstalk" on page 4–72. This topology is also known as *dual stripline*, because there are two signal layers between the two power planes.

Figure 4–30. Simulation Setup for an Edge-Coupled Differential Offset Stripline

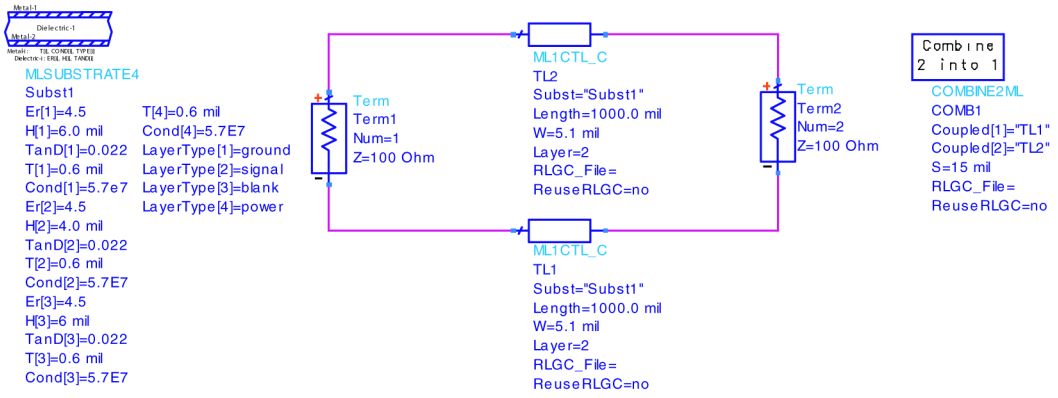
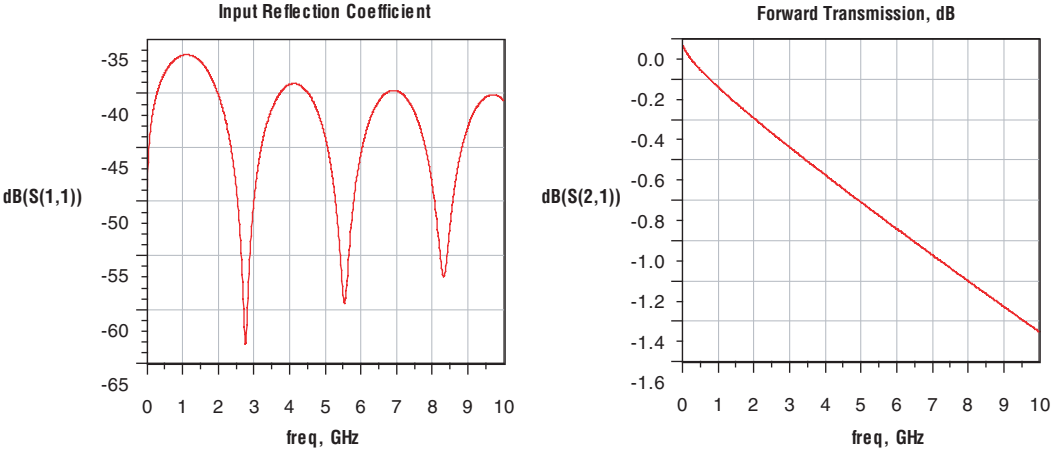


Figure 4–31. Simulation Results for an Edge-Coupled Differential Offset Stripline



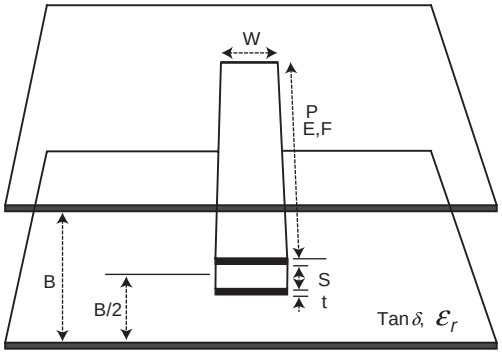
Broadside-Coupled Differential Stripline

Broadside-coupled differential stripline, shown in Figure 4–32, has some advantages over the edge-coupled differential stripline, including higher coupling and easier routing. The disadvantages are the possibility of the

traces becoming too narrow for the manufacturability of the board and the board becoming too thick to compensate for the trace-width thickness ratio.

This configuration is helpful when differential high-speed lines reside in the inner rows of the BGA, and it is impossible to route two lines (one differential pair) between vias or pads.

Figure 4–32. Broadside-Coupled Differential Stripline



Figures 4–33 and 4–34 show the frequency domain simulation setup and results for a 100-Ω broadside-coupled differential stripline, respectively.

Figure 4–33. Simulation Setup for a Broadside-Coupled Differential Stripline

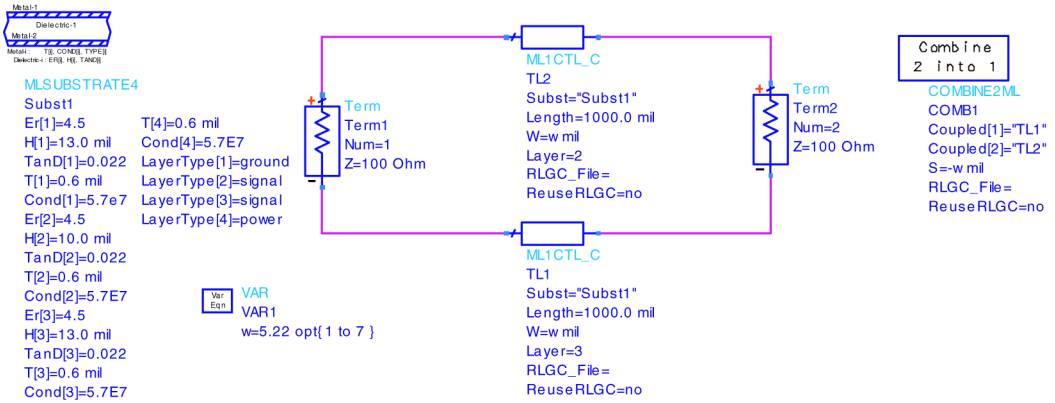
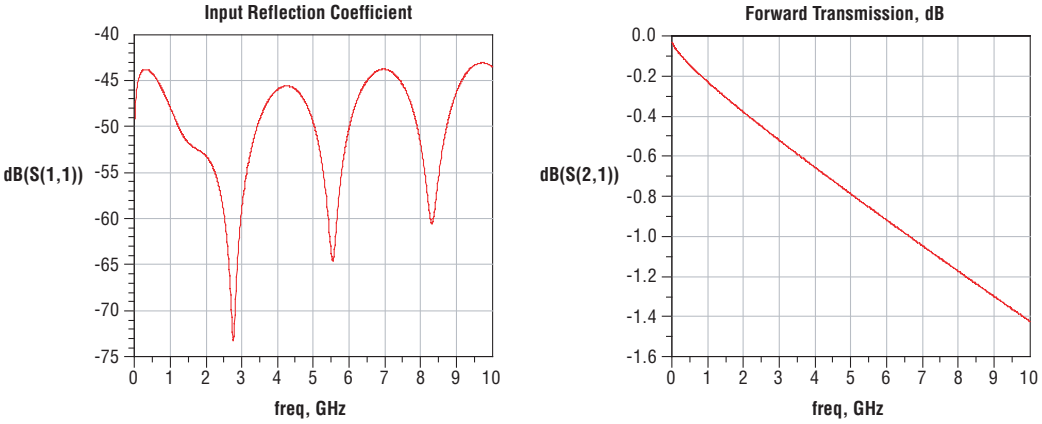


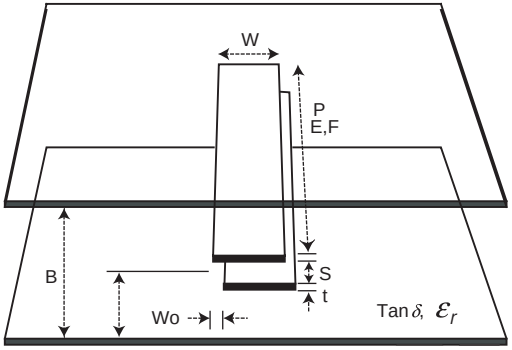
Figure 4–34. Simulation Results for a Broadside-Coupled Differential Stripline



Broadside-Coupled Differential Offset Stripline

The broadside-coupled differential offset stripline structure, shown in Figure 4–35, has the advantages of the broadside-coupled differential stripline with the added flexibility that allows you to adjust the differential impedance by offsetting the two conductor strips. The differential impedance increases as the offset dimension *Wo* increases.

Figure 4–35. Broadside-Coupled Differential Offset Stripline



Figures 4–36 and 4–37 show the frequency domain simulation setup and results for a 100-Ω broadside-coupled differential offset stripline, respectively.

The Stratix GX development board uses edge coupling instead of broadside coupling, with loose coupling between the positive and negative traces. Although tight coupling provides greater noise immunity, it is necessary to loosen the coupling because of the mechanical placement requirements of SMA connectors. With loose coupling, if the routing is broadside coupled, the thickness of the board increases. With loose coupling, edge-coupled routing is preferred unless there is the flexibility to increase board thickness.

Figure 4–36. Simulation Setup for a Broadside-Coupled Differential Offset Stripline

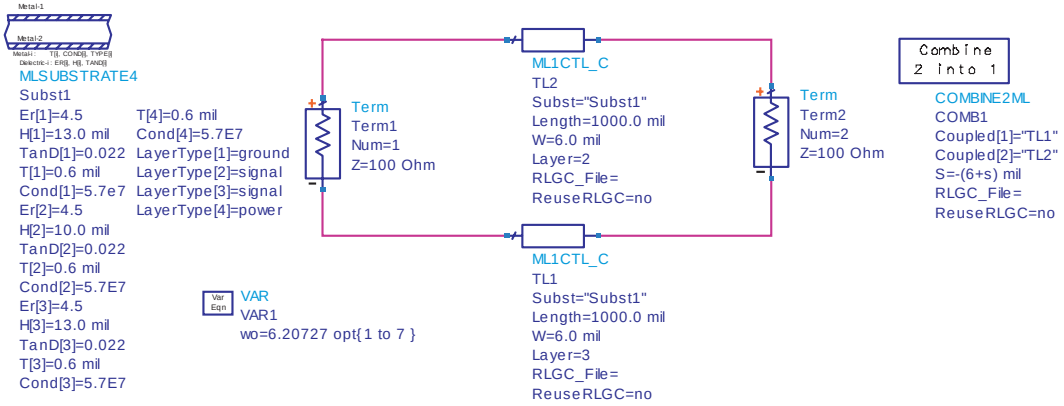
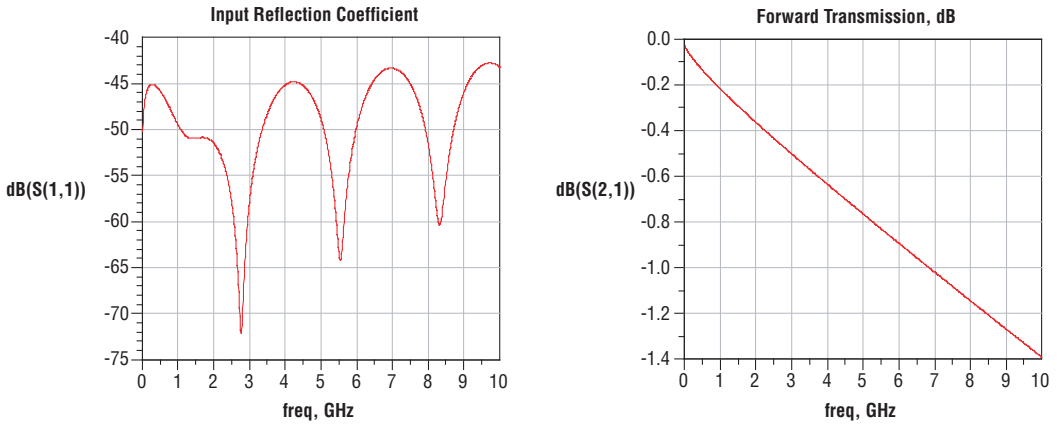


Figure 4–37. Simulation Results for a Broadside-Coupled Differential Offset Stripline



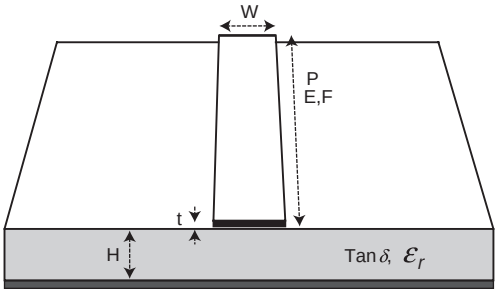
Microstrips

Microstrip topologies can be either simple or differential. Microstrips are the most popular planar transmission lines because of their low cost and easy integration with other passive and active components on PCBs.

Simple Microstrip

Figure 4–38 shows the simple microstrip topology. A microstrip is a conductive strip of width W and thickness t, placed on top of a dielectric material backed with a ground plane.

Figure 4–38. Simple Microstrip



Figures 4–39 and 4–40 show the frequency domain simulation setup and results for a 50-W simple microstrip, respectively.

Figure 4–39. Simulation Setup for a Simple Microstrip

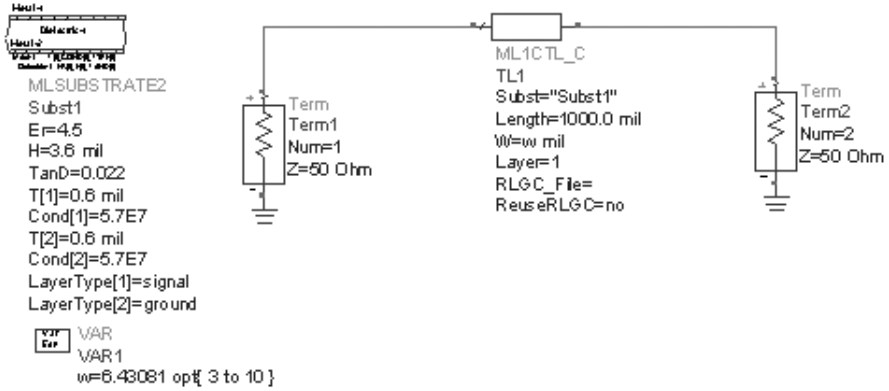
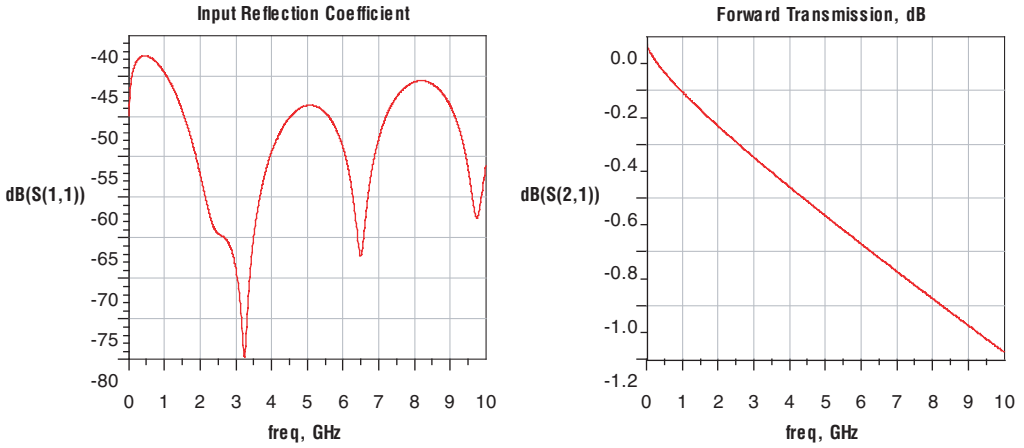


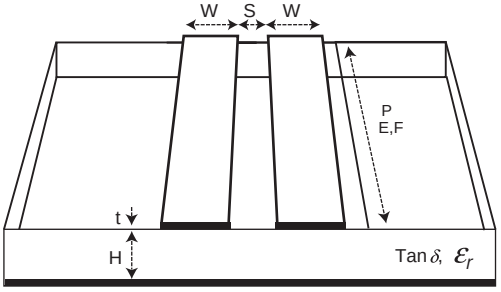
Figure 4–40. Simulation Results for a Simple Microstrip



Differential Microstrip

The differential microstrip topology, shown in Figure 4–41, has the advantages of the simple microstrip, but also provides common mode noise cancellation. The main problem with differential microstrips is that the coupling between the positive and negative traces is low.

Figure 4–41. Differential Microstrip



Figures 4–42 and 4–43 show two variants of the differential microstrip extracted from the Stratix GX development board.

Figure 4–42. Loosely Coupled Differential Microstrip

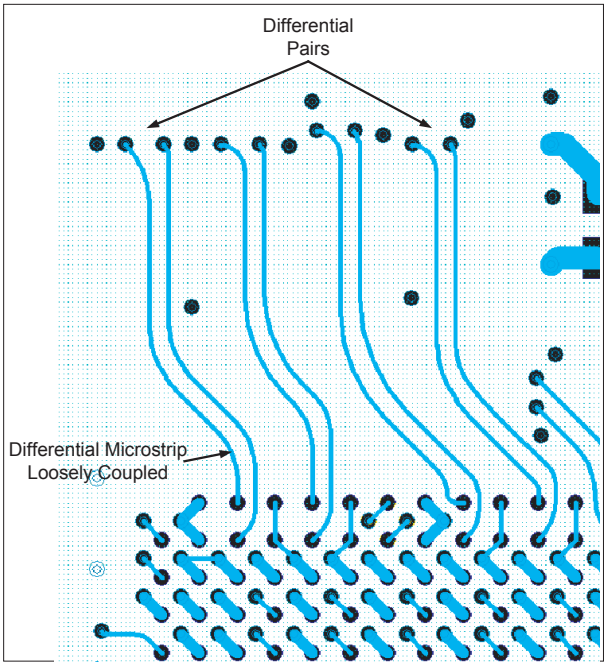
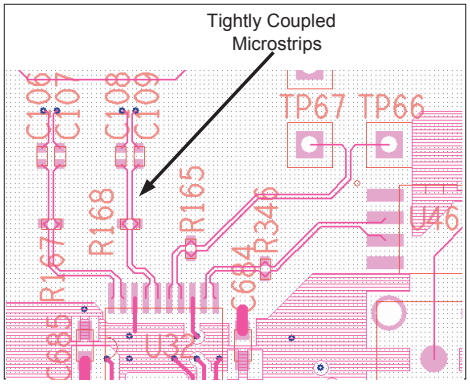


Figure 4–43. Tightly Coupled Differential Microstrip



Figures 4–44 and 4–45 show the frequency domain simulation setup and results for a 100-Ω tightly coupled differential microstrip, respectively.

Figure 4–44. Simulation Setup for a Tightly Coupled Differential Microstrip

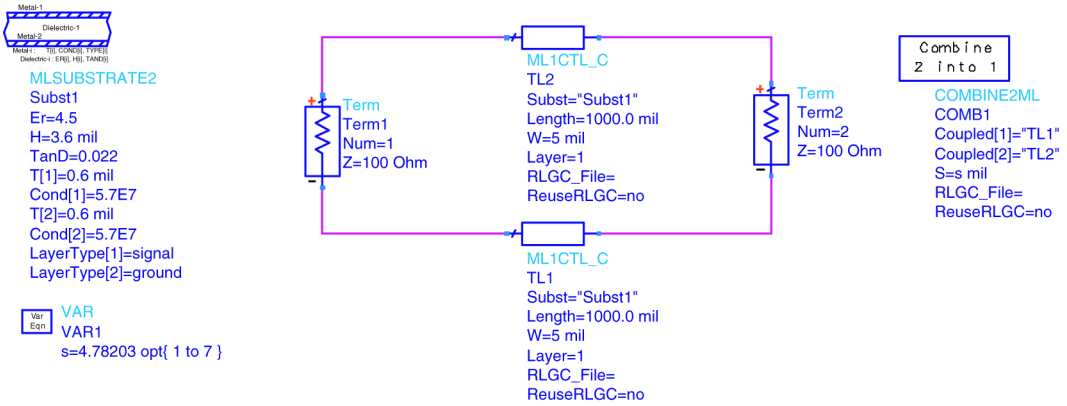
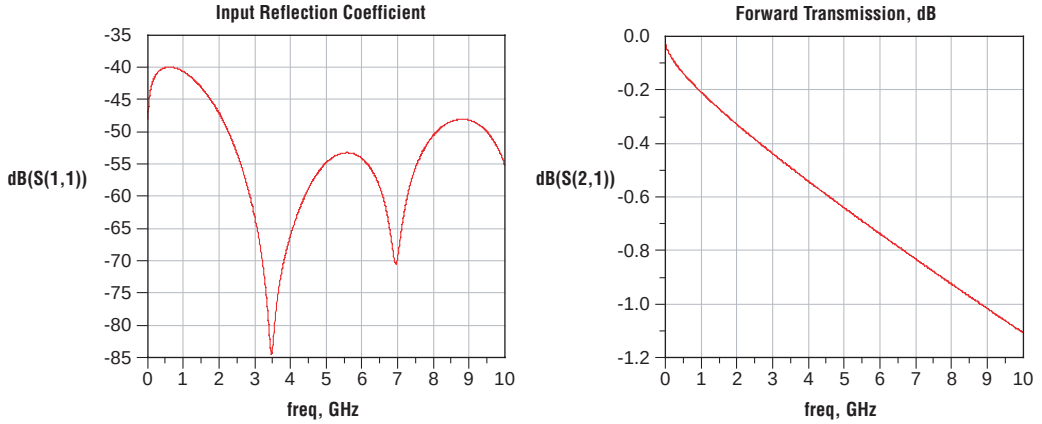


Figure 4–45. Simulation Results for a Tightly Coupled Differential Microstrip



Coplanar Wave Guides

There are three types of coplanar wave guides: simple, grounded, and grounded differential.

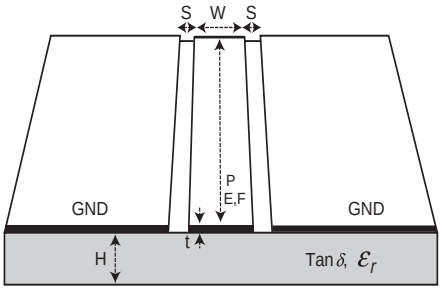
Simple Coplanar Wave Guide

The simple coplanar wave guide, shown in Figure 4–46, is used primarily in microwave systems. This structure does not require vias, and you can easily mount passive or active devices in the signal path, resulting in a

low-loss, high-speed transmission line. The simple coplanar wave guide requires that the substrate thickness (H) be "infinite," so that the fields remain outside the dielectric.

You cannot use this structure on multilayer boards because it cannot have a second layer.

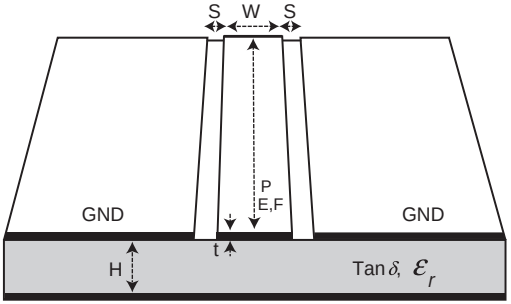
Figure 4–46. Simple Coplanar Wave Guide



Grounded Coplanar Wave Guide

The grounded coplanar wave guide is used extensively in communications systems that integrate different technologies such as surface mount technology, multichip modules, and multilayer boards. The dielectric thickness (H) must be at least five times the spacing (S) to be considered a grounded coplanar wave guide; otherwise, it is considered a microstrip with a ground shield. Figure 4–47 shows the grounded coplanar wave guide topology.

Figure 4–47. Grounded Coplanar Wave Guide



Figures 4–48 and 4–49 show the frequency domain simulation setup and results for a 50-Ω grounded coplanar wave guide, respectively.

Figure 4–48. Simulation Setup for a Grounded Coplanar Wave Guide

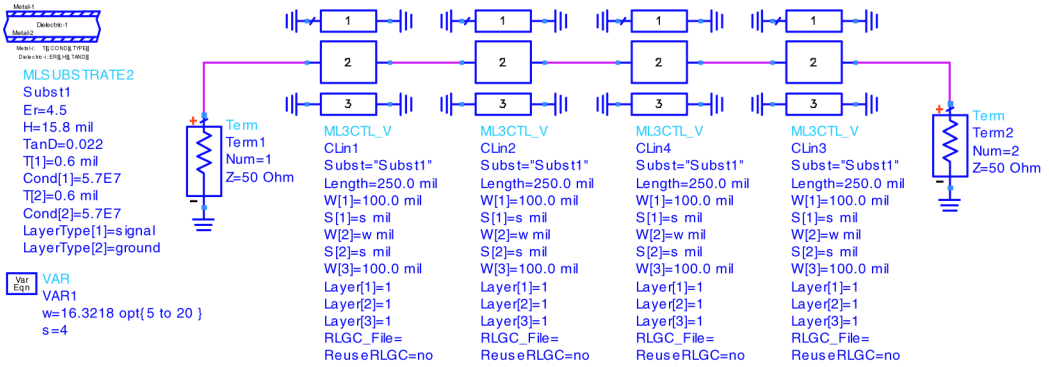


Figure 4–49. Simulation Results for a Grounded Coplanar Wave Guide

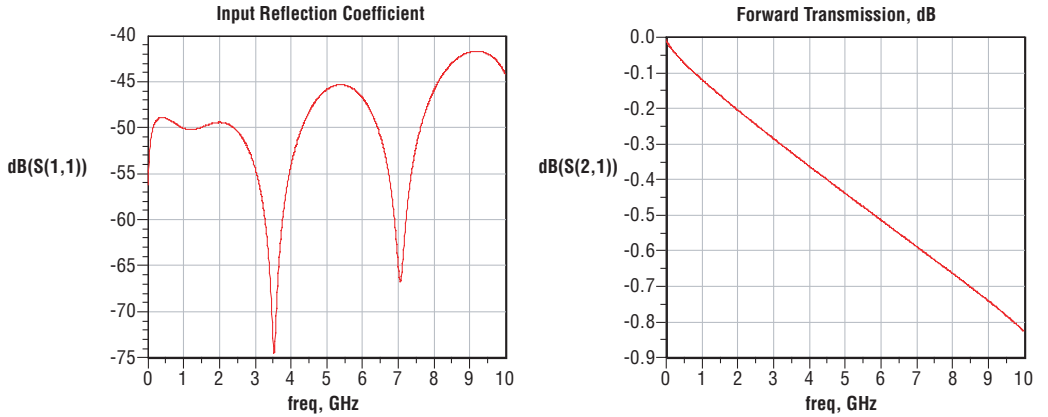
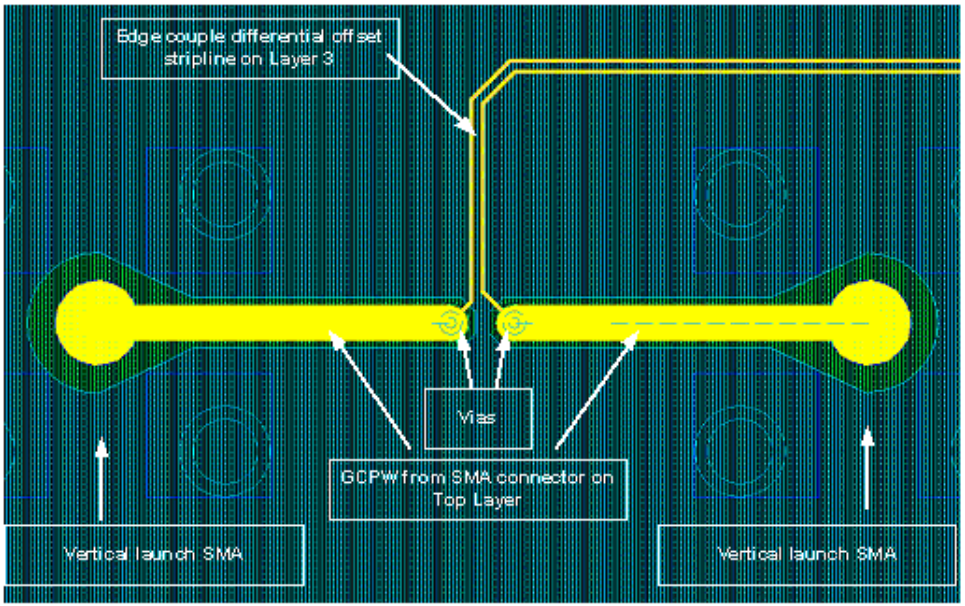


Figure 4–50 shows the Stratix GX development board's grounded coplanar wave guide and its edge-coupled differential offset stripline layout capture plot. This particular differential pair was used for the 1.0-Gbps source synchronous lines.

Figure 4–50. Layout Capture Plot for the Grounded Coplanar Wave Guide & Edge-Coupled Differential Offset Stripline

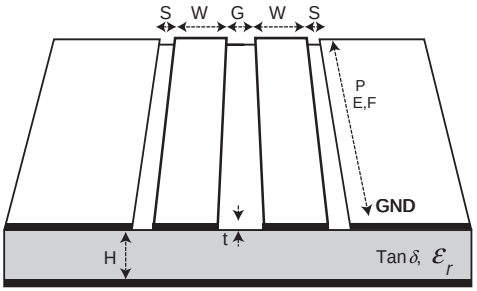


Grounded Differential Coplanar Wave Guide

The grounded differential coplanar wave guide is the differential version of the grounded coplanar wave guide and is used in high-speed digital systems that require maximum noise immunity. Figure 4–51 shows the topology. The same limitations for S and G apply in this topology as for the grounded coplanar wave guide.

This topology does not appear on the Stratix GX development board because of the loose coupling requirements. See "Broadside-Coupled Differential Offset Stripline" on page 4–42 for more information.

Figure 4–51. Grounded Differential Coplanar Wave Guide



Figures 4–52 and 4–53 show the frequency domain simulation setup and results for a 100-Ω grounded differential coplanar wave guide, respectively.

Figure 4–52. Simulation Setup for a Grounded Differential Coplanar Wave Guide

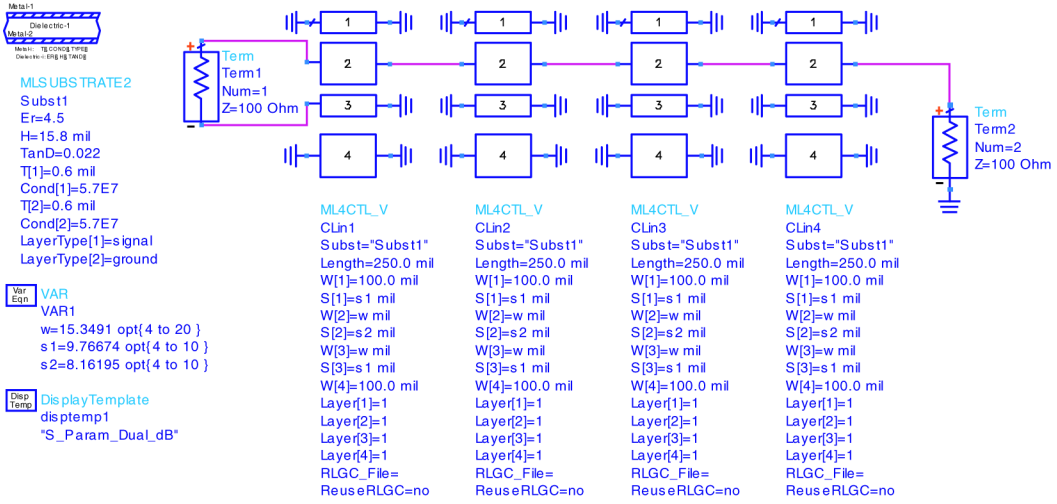
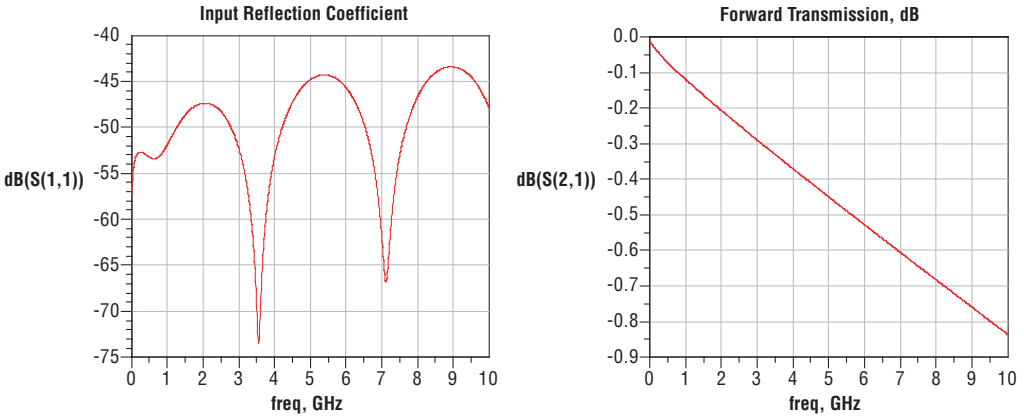


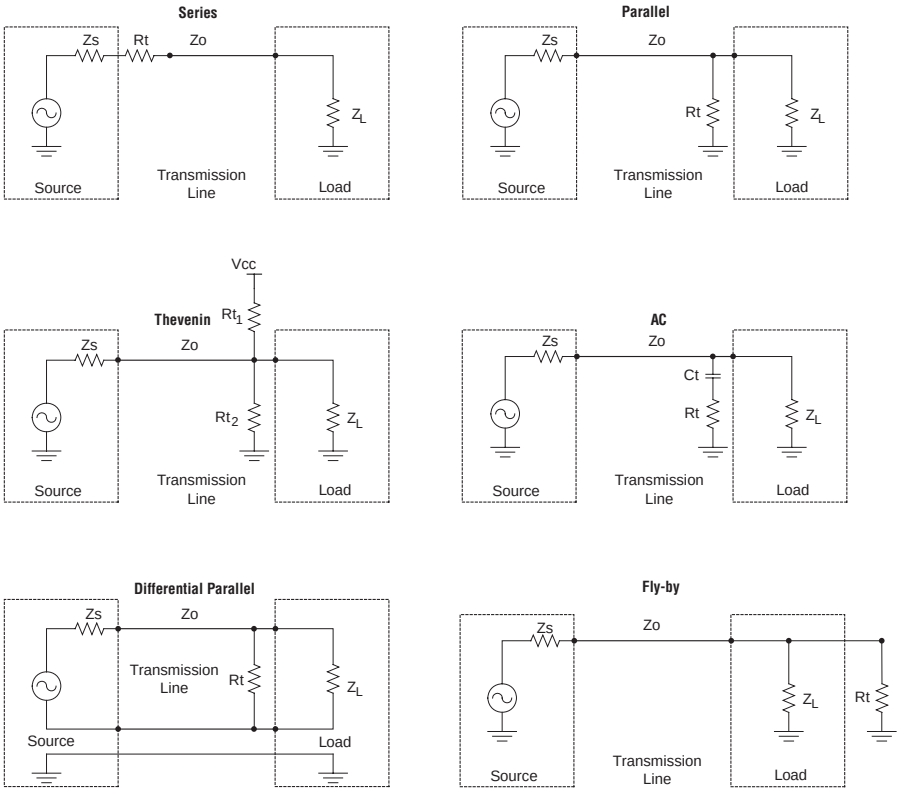
Figure 4–53. Simulation Results for a Grounded Differential Coplanar Wave Guide



Transmission Line Termination

High-speed signals require termination at the beginning, end, or both sides of the transmission line to prevent the reflected signal from the ends from corrupting the original signal. This section briefly describes several termination techniques. For more details about each technique, refer to *AN315: Guidelines for Designing High-Speed FPGA PCBs*. Figure 4–54 shows the different termination techniques.

Figure 4–54. Termination Techniques



Series Termination

With series termination, you place the termination resistor in series with the transmission line at the source end of the transmission line. The value of the resistor (R_s) and the value of the output impedance of the buffer must add up to 50 Ω for optimal performance. This is not always possible in cases when the buffer output impedance depends on the current setting or the voltage swing. In most cases, using a 22-Ω resistor in series works well for typical CMOS buffers such as those in Stratix GX devices. Series termination minimizes the DC power dissipation, but perfect impedance matching is not always possible, because the buffer output impedance varies.

Parallel Termination

With parallel termination, you place a resistor whose value is equal to the impedance of the trace at the end of the line on the receiver side. Ideally, the resistor should be directly on the pin. Parallel termination is better than series termination when the output impedance of the buffer is unknown. With parallel termination, you get an almost perfect impedance match between the transmission line and the termination resistor. The disadvantage with parallel termination is that it increases the DC power dissipation because of the DC path to ground.

Thévenin Termination

Thévenin termination is identical to parallel termination, except that you use two resistors to achieve the impedance match. The two resistors can also provide a DC level. Two criteria determine the value of the resistors. First, the two resistors in parallel must equal the line impedance. Second, the resistors must form a voltage divider so that the restored DC voltage equals the desired DC level. With parallel termination, you can achieve an almost perfect impedance match between the transmission line and the termination resistors. You can also restore DC voltage to AC-coupled signals. The main disadvantages to thévenin termination are the DC power dissipation and the added elements.

AC Termination

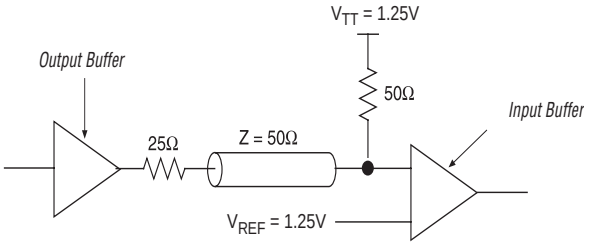
AC termination is identical to parallel termination except that the termination impedance consists of a resistor and a capacitor. The capacitor acts as a DC block, reducing static power dissipation. The capacitor must be chosen so that the RC time constant is about one rise time of the signal. Perform simulations to choose the best value of the capacitor. AC termination provides the advantages of parallel termination without the DC power dissipation. The main disadvantages are the rise and fall time degradation and the added elements.

Differential Parallel Termination

With differential parallel termination, you place the termination resistor between the positive and negative signals, close to the receiver pins. This type of termination is a subset of parallel termination.

Specific I/O standards, for example, SSTL-II for the DDR interface, sometimes require both series and parallel termination. Figure 4–55 shows the termination shown in this particular standard.

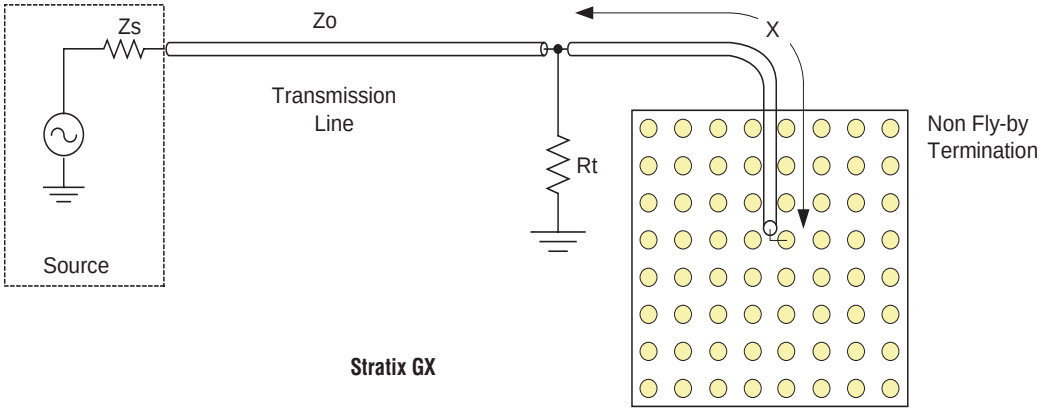
Figure 4–55. Class I Termination for DDR Memory Interfaces



Fly-By Termination

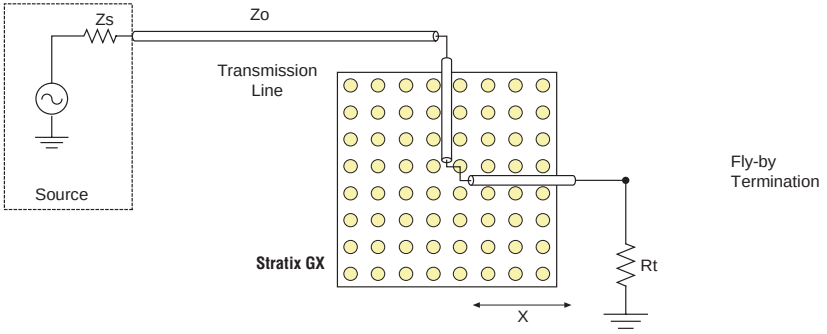
Fly-by termination is not precisely a termination technique, but a placement technique for termination resistors. To understand fly-by, consider the parallel (non fly-by) termination shown in Figure 4–56. Layout constraints might prevent the termination resistor, R_t, from being placed close to the receiver pin (the load). If the pin is in the middle of the device and there is only an inch of trace length between the termination point and the receiver pin, the setup degrades the signal quality, because the extra stub length acts as a capacitive load.

Figure 4–56. Normal Termination Resistor Placement



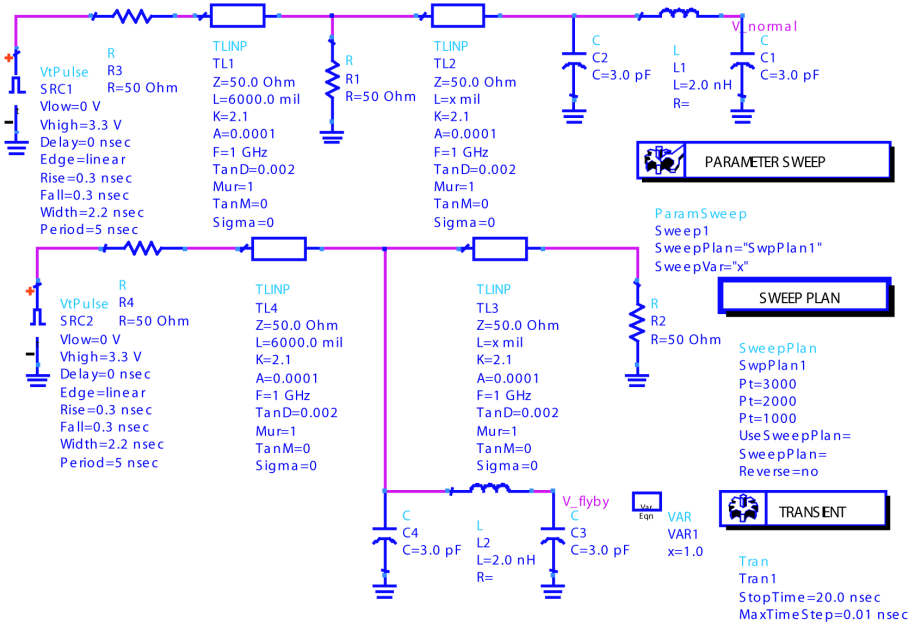
To prevent this degradation, place the termination resistor after the load, as shown in Figure 4–57. Even if the resistor is a couple of inches away, the termination occurs at the end of the transmission line, and there are no significant reflections. The trace from the transmission line to the receiver pin can be kept small.

Figure 4–57. Fly-By Termination Resistor Placement Technique



The simulation setup for fly-by termination is shown in Figure 4–58. The upper circuit in this diagram is for fly-by termination resistor placement, and the lower is for normal placement. Figure 4–59 shows some simulation results that show how the fly-by termination technique helps improve signal integrity.

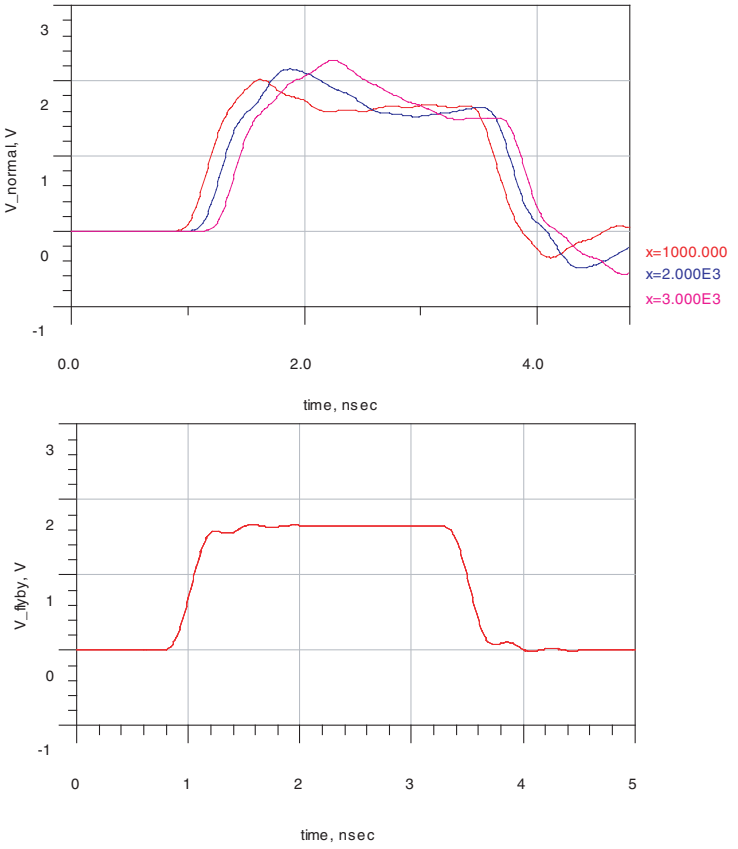
Figure 4–58. ADS Simulation Setup for Normal Versus Fly-By Termination



The simulation results are shown in Figure 4–59. The signal quality remains the same regardless of how far away the resistor is, provided that it is placed in fly-by mode.

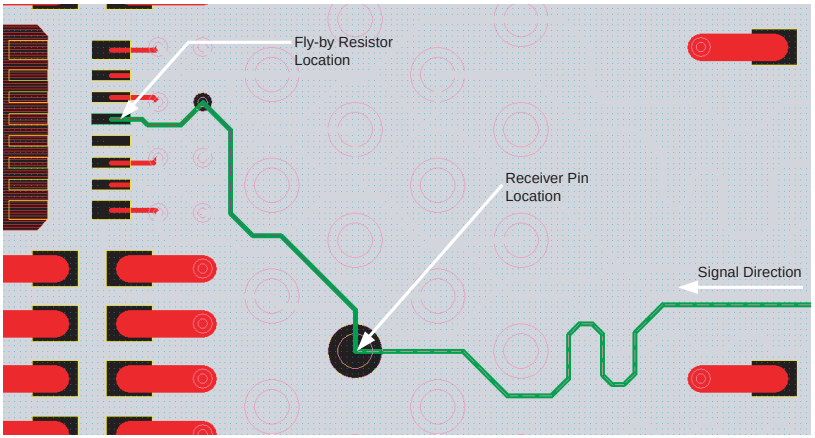
For the non fly-by case, the signal has an overshoot, which worsens when you place the resistor farther away.

Figure 4–59. Signal Quality at the Receiver Device Pins



Altera uses the fly-by resistor placement technique extensively on the Stratix GX development board. Figure 4–60 shows an example capture plot.

Figure 4–60. Fly-By Termination Example



Transmission Line Routing

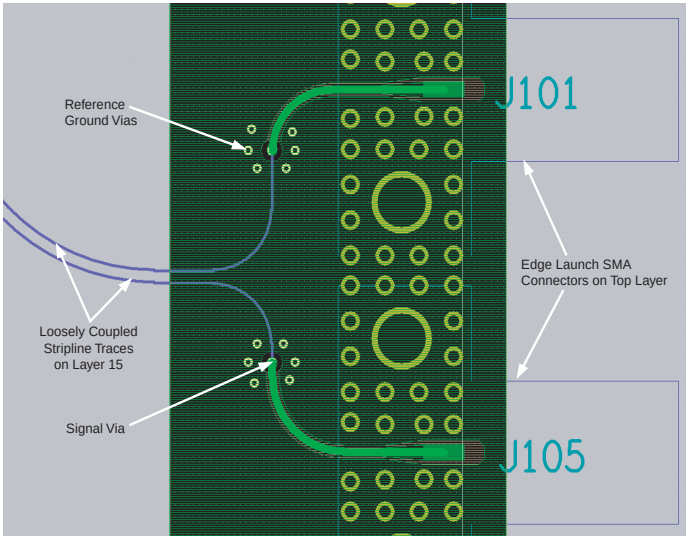
This section describes the things to keep in mind when designing transmission line routing for your high-speed board.

General Guidelines

This section provides a summary of routing guidelines.

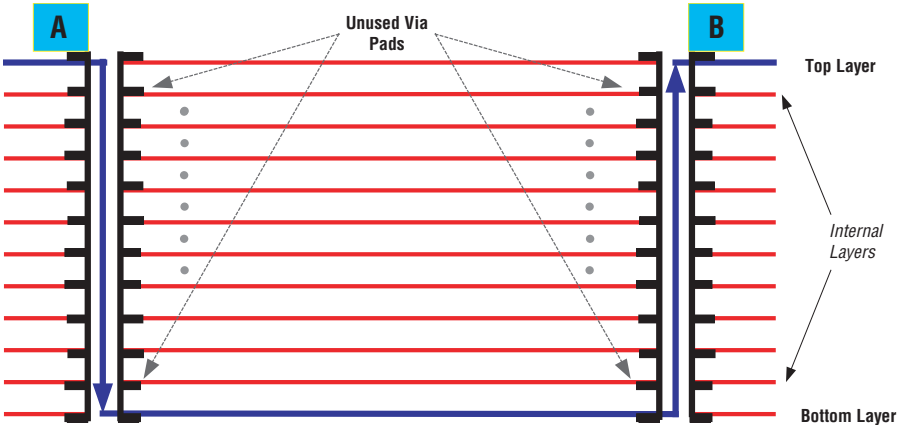
- Use tight coupling for differential traces as much as possible. If it is not possible to maintain tight coupling throughout the trace length, then you must use loose coupling for the entire trace. Figure 4–61 shows 3.125-Gbps differential transceiver traces, which are loosely coupled on layer 15 and the top layer of the Stratix GX development board. Tight coupling is not possible on the top layer because of the minimum separation required (because of the mechanical constraints of the SMA connectors). If you used tight coupling on layer 15, the signal would need to transition from tight to loose coupling, which would introduce impedance discontinuities.

Figure 4–61. Loose Coupling & Ground Reference Example



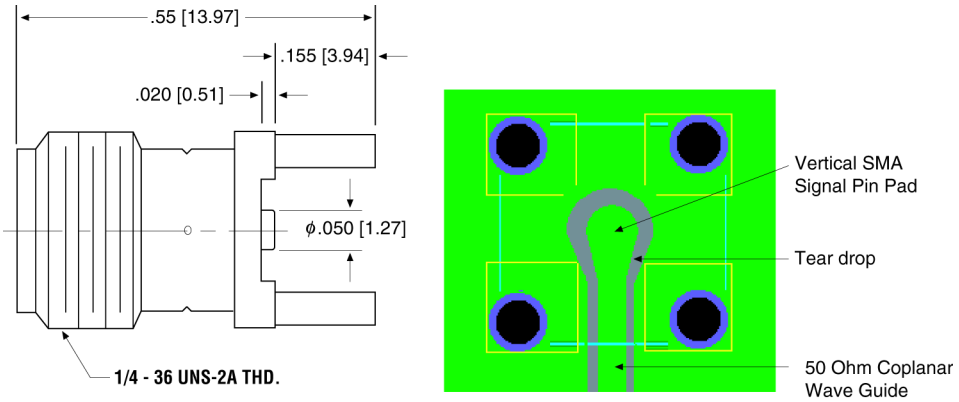
- Microstrip and stripline transmission lines are both fine for routing. Stripline tends to have more loss than microstrip, but it also shields the signals from possible noise from other sources. The constraints during layout design often dictate on which layer to do the routing.
- Use rounded corners while routing. Do not use 90° bends, which introduce impedance discontinuities. A 45° bend provides a compromise, but rounded corners offer the best performance.
- Both broadside and edge coupling are fine. Generally, broadside coupling makes it easy to route out of the BGA, but requires more layers. Edge coupling makes it harder to route out of the BGA, but requires fewer layers. Both provide acceptable signal integrity performance.
- Do not allow high-speed signals to cross over plane splits. A crossover causes signals to see a longer return path, which increases trace inductance. The increased inductance changes the impedance of the line, causing signal integrity problems.
- Remove all unused via pads during the manufacturing process. Some fabrication houses refer to unused via pads as "nonfunctional via pads." Unused via pads add additional capacitance on the signal path. Figure 4–62 shows examples of unused via pads.

Figure 4–62. Unused Via Pads



- If the signal must be routed through a via, route it so that the via stub length is minimized. The option shown in Figure 4–64 minimizes the via stub length and is preferred over the option shown in Figure 4–65. Altera used this technique in the Stratix GX development board. The high-speed traces were taken to layer 16 from the top layer through the vias to use as much of the via length as possible and to minimize the stubs.
- For dual stripline traces (Power 1 – Signal 1 – Signal 2 - Power 2 configuration), ensure that the signals on the Signal 1 layer are orthogonal to the signals on the Signal 2 layer if they cross each other. If they do not cross each other, ensure that they are at least 4W away, where W is the width of the traces.
- If a connector has a section that does not have controlled impedance, try to minimize the length of the connector. For example, for SMA connectors, reduce the center conductor length as much as you can (less than 20 mils if possible). This way the parasitic inductance is reduced.
- Use "teardropping" to reduce impedance discontinuity when going from a wide pin and trace to a narrow pin or trace. For example, when you interface an SMA connector to a trace, use teardropping. Figure 4–63 shows the dimensions of an SMA connector and a screen capture of how Altera used teardropping on the Stratix GX development board. The SMA connector has a 50-mil diameter center pin, but the board traces might be 5-mils wide. Teardropping helps minimize the impedance discontinuity.

Figure 4–63. SMA Teardropping



- Avoid routing a high-speed signal over many layers, because this type of routing introduces via discontinuities.
- Use reference ground vias wherever signal vias are necessary. The reference ground vias should be placed close to the signal vias. Figure 4–61 on page 4–59 shows an example of using ground reference vias near the signal vias where a high-speed trace changes layers. This example is from the edge-launch SMA connectors of the 3.125-Gbps transceivers on the Stratix GX development board.
- Avoid having long stubs on the via, using most of the via length for routing. For example, if you have a signal that originates on the top layer and you have to use an internal layer for routing, use the one closest to the bottom layer. This approach avoids excess via length. The option shown in Figure 4–64 on page 4–63 is preferred over the option shown in Figure 4–65 on page 4–63.
- Avoid reference plane changes for high-speed signals. Reference plane changes occur when you reference part of the trace to a ground plane and the other part to another ground plane or a power plane.
- Avoid using power planes for references.

Length Matching

Source-synchronous interfaces require length matching, except when using dynamic phase alignment. You must ensure that this requirement is clearly stated in the layout guidelines. How closely the lengths must be matched depends on the data rate and the available timing margin. For example, consider a source synchronous link at 840 Mbps where the bit unit interval equals 1.19 ns. In this example, assume that you allocate a length mismatch of 2% for the maximum timing margin loss. At 1.19 ns

unit intervals, this value equals approximately 24 ps. Using the microstrip equation on page 4–4, for a microstrip line with FR-4 dielectric, the signal travels 1 inch in 142 ps. In 24 ps the signal can travel 24/142 = 0.17 inches, or 170 mils. Therefore, you must match the lengths of all the signals (including data, clocks, addresses, and any controls signals) to within ± 85 mils.

Examples of source-synchronous interfaces include the 840-Mbps interface available on Stratix and Stratix GX devices, the SSTL interface for DDR memory, and the HSTL interface for Quad Data Rate (QDR) interface.

A notable exception to the length matching requirement is the source-synchronous interface with DPA available in Stratix GX devices, which allows 1-Gbps data transfer without requiring length matching. The data and clock lines can literally be inches longer or shorter than each other and the interface still operates robustly.

You must account for the length of every via a signal traverses through and include the accumulated value in the length report (a file generated by the layout tool). Do not include the entire length of the via, just the portion the signal traverses (ignore the stub length). For example, Figures 4–64 and 4–65 show two possibilities for signal routing on a board with 12 signal layers. For both options, the signal originates at point A and ends at point B on the top layer. In Figure 4–65, the signal is routed on layer 3; in Figure 4–64, the signal is routed on the bottom layer. The total length the signal travels, for the respective options, is:

$$L_{routing,option1} = (z' + y + z') = (2z' + y)$$

$$L_{routing,option2} = (z + y + z) = (2z + y)$$

Figure 4–64. Optimal Routing Path With Through-Hole Vias

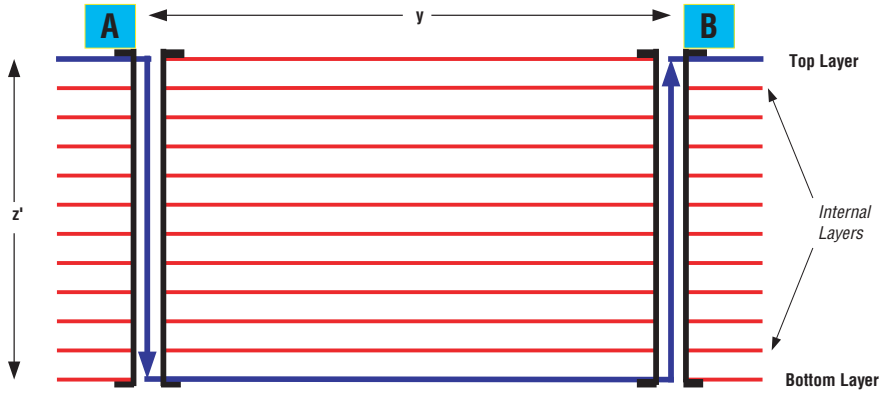
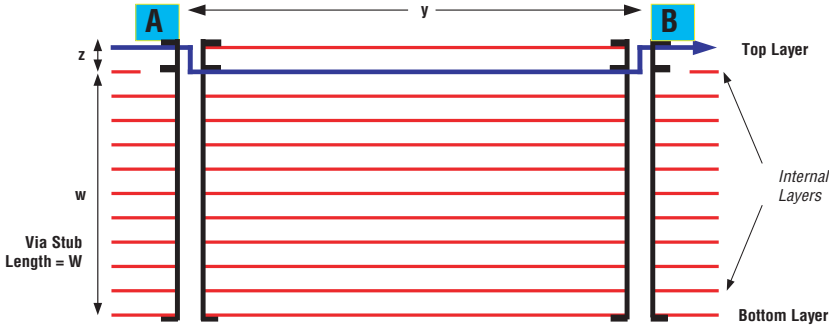


Figure 4–65. Sub-Optimal Routing Path With Through-Hole Vias



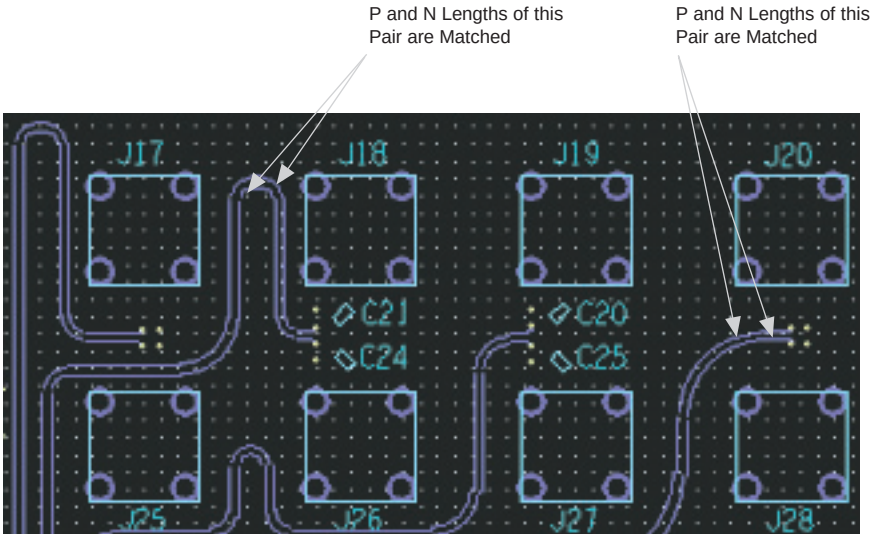
In Figure 4–65, W represents the extra stub that you need to consider in length calculations only if you use layer 16 for routing.

In the equations on page 4–62, the total difference in via lengths between these two routing options is 2z'-2z = 2w. This distance can be over 100 mils for thick boards.

Because this kind of calculation can quickly become intractable, it is best to use as few vias as possible on the board and to tighten the length-matching specification. For instance, in the example above, you can tighten the specification from ± 85 mils to ± 50 mils.

For differential traces, it is important to match the length of the positive and negative signals. A mismatch in lengths causes duty cycle distortion, which can decrease the timing margin. Altera recommends that the lengths be matched to ± 50 mils. If the signal goes through a turn and the outer trace is longer than the inner trace, you can use a second turn in the opposite direction as an offset. Figure 4–66 shows an example of high-speed transceiver routing to achieve length matching on the Stratix GX development board.

Figure 4–66. Transceiver Signal Routing for Length Matching



Other Transmission Line Issues

This section describes transmission line issues other than routing.

Stackup

The stackup design is an iterative process in which the relative permittivity (ε_r), trace width, layer count, and transmission line structures are set by engineers to achieve the best system performance at the minimum cost. These parameters are selected based on the

engineering team's knowledge of the system, the fabrication house, and the simulations. Table 4–6 shows the stackup for the Stratix GX development board.

Table 4–6. Stratix GX Stack-Up

Layer Number	Type	Cu Weight (oz)	Foil Thickness (in.)	Dielectric Thickness (in.)	Construction	ε_r
1	Signal	0.5	0.0007	0.0036	1 × 1080	3.76
2	Plane	1.0	0.0014	0.0040	1080 + 2113	4.00
3	Signal	0.5	0.0007	0.0061	2116	3.86
4	Signal	0.5	0.0007	0.0040	1080 + 2113	4.00
5	Plane	1.0	0.0014	0.0045	1080 + 2113	3.86
6	Signal	0.5	0.0007	0.005	2116	3.63
7	Signal	0.5	0.0007	0.0045	1080 + 2113	3.86
8	Plane	1.0	0.0014	0.0040	1080 + 2113	4.00
9	Signal	0.5	0.0007	0.0061	2116	3.86
10	Signal	0.5	0.0007	0.0045	1080 + 2113	4.00
11	Plane	1.0	0.0014	0.005	1080 + 2113	4.00
12	Signal	0.5	0.0007	0.0045	2116	3.86
13	Signal	0.5	0.0007	0.005	1080 + 2113	4.00
14	Plane	1.0	0.0014	0.0045	1080 + 2113	4.00
15	Signal	0.5	0.0007	0.0061	2116	3.86
16	Signal	0.5	0.0007	0.0040	1080 + 2113	4.00
17	Plane	1.0	0.0014	0.0036	1 × 1080	3.76
18	Signal	0.5	0.0007			
Total Thickness Cu to Cu (in.)		0.093 ± 0.006				

Because of the large number of power and ground planes needed on the Stratix GX development board, Altera uses a symmetrical stackup that is optimized for striplines. The dielectric thickness, permittivity, and trace width were chosen so that the total board thickness was approximately 0.093 in. A minimum trace width of 0.004 in keeps costs low and reduces the effects of fabrication deviations. The layer count is set based on the density of the board.

The development board material is NELCO FR-4 4000-6, which is an enhanced multifunctional, high temperature resin system that shows a relative permittivity of ~3.86. and tanδ of 0.022. The stackup consists of 6 solid planes and 12 signal layers arranged symmetrically from the center.

For transmission line topologies, Altera uses differential microstrip, grounded coplanar wave guide, and differential dual stripline (offset edge-coupled differential stripline). The impedance is 50 Ω for single-ended traces and 100 Ω (differential) for differential traces.

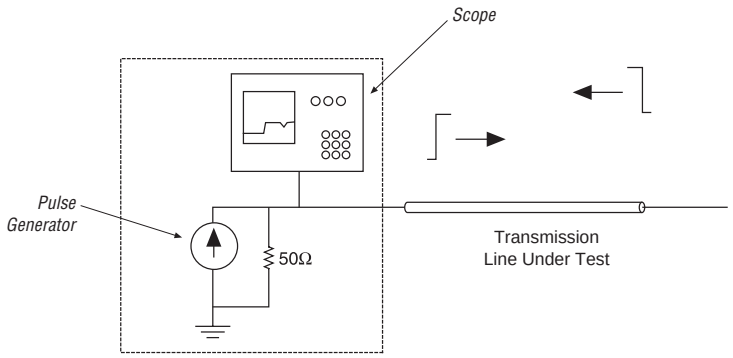
In the stackup design process, the fabrication house has considerable influence during the fine-tuning stage. Typically, the engineer provides a preliminary stackup, and the fabrication house responds with a modified stackup based on their process tolerances. They typically guarantee the impedance required on the traces. Typical tolerances are ± 10%, although ± 5% is fairly common for high-speed boards. Altera uses ± 5% for better system performance.

Discontinuities

You can characterize discontinuities in the transmission path using either a frequency domain tool or a time domain tool. In the frequency domain, you can use a network analyzer for characterization. A commonly used time domain tool is the time domain reflectometer (TDR). For digital systems that are commonly described in time domain, the TDR is often easier and more intuitive to use. This section summarizes the principles of a TDR and how to interpret the results.

A TDR is an oscilloscope with the capability of sending a very fast signal pulse. The pulse travels down the transmission line and is reflected wherever it sees a discontinuity. Figure 4–67 shows a conceptual diagram of a TDR. The reflected wave takes $2T$ time units to be displayed in the scope, because it takes T time units for the pulse to reach the discontinuity and another T time units for it to return to the scope.

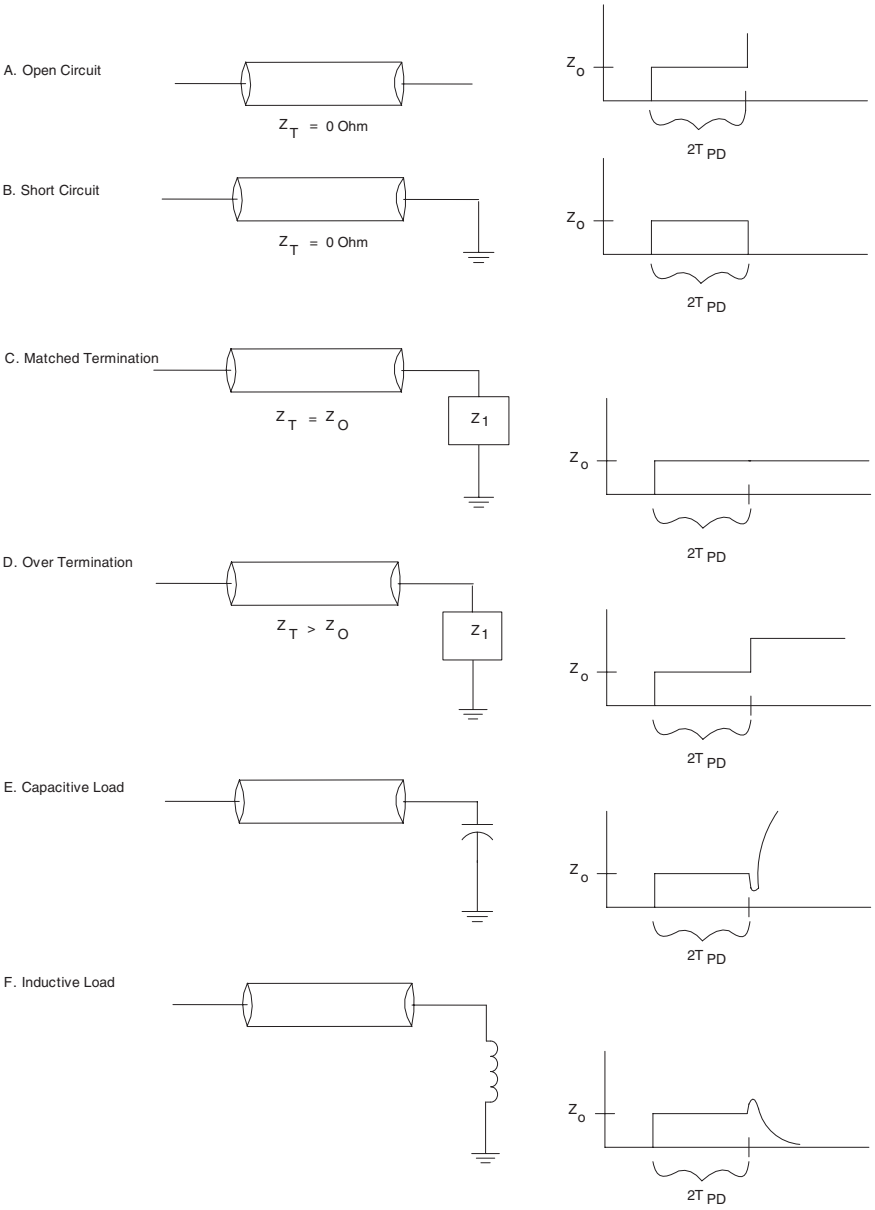
Figure 4–67. Conceptual Diagram of a TDR System



Using a TDR, resistive discontinuities show a staircase profile, capacitive discontinuities cause a dip, and inductive discontinuities cause a bump. The reason for this is that initially the capacitor acts as a short-circuit for a high-frequency pulse, and the inductor acts like an open circuit. Based on the location of the impedance discontinuity on a TDR plot, you can trace it to the exact board location.

Figure 4–68 shows the different types of terminations and the associated TDR responses.

Figure 4–68. TDR Responses to Different Terminations & Discontinuities



To determine the value of the resistive discontinuity, look at the size of the step in the staircase waveform. To determine the value of the capacitive or inductive discontinuity, look at the rise time of the associated response. Then use the following formulas:

Resistive Termination

$$R = Z_0 \frac{1+\Gamma}{1-\Gamma}$$

where Γ is the reflection coefficient. For a 1-V initial signal, Γ equals the step of the staircase. For a different initial step, you need to scale Γ linearly.

Capacitive Termination

$$tau = 2.2t_r = RC$$

for first order R-C circuits.

Inductive Termination

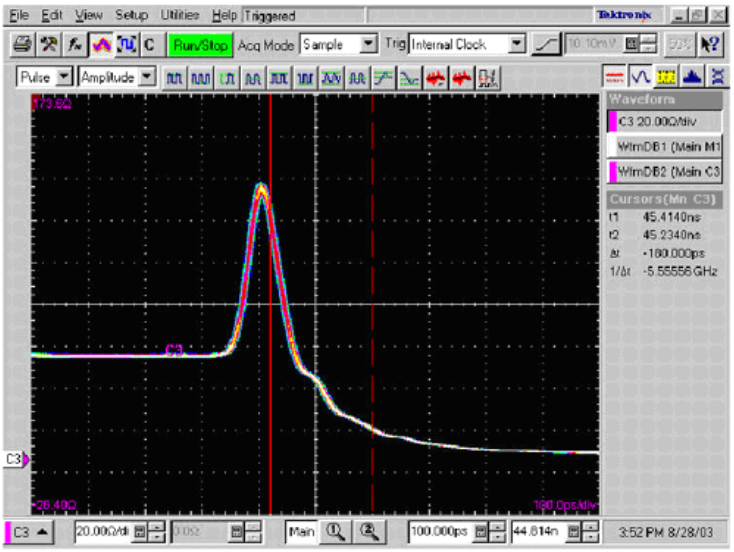
$$tau = 2.2t_r = L / R$$

for first order L-R circuits.

In the previous equations, R is the effective resistance and impedance in the circuit. The rise time (t_r) is determined from the TDR response. The only variables remaining are L and C, either of which can be computed.

Example: Find the value of the inductance on the 0.1-in pitch header pin connector from the TDR response shown in Figure 4–69.

Figure 4–69. TDR Plot of a Header Connected to a 50-Ω Cable



Solution: From the plot of the header discontinuity, the signal fall time is about 180 ps, and the cable's impedance is 50 Ω Therefore,
$L = 2.2 \times t_r \times R = 2.2 \times 180 \times 50 = 19{,}800$ pH = 19.8 nH.

Real-world TDR responses are often complicated and hard to interpret because of multiple reflections and a combination of discontinuities (capacitive, inductive, and resistive) that could be present on the board. One approach is to isolate each part and measure the TDR by soldering the part at the end of an SMA cable so that each part can be characterized separately.

As a real-world example of a TDR response, consider the following measurements, which were taken to characterize SMA connectors being evaluated during the design of the Stratix GX development board. The SMA connectors are shown in Figure 4–70, and the results are shown in Figure 4–71.

The first measurement Figure 4–71 shows the TDR response of the SMA cable used. The second shows the response of the SMA cable with the connector connected at the end. You can see quite a bit of ringing, signifying the presence of parasitics. The third shows the response of the SMA cable and connector assembly, with the connector's center conductor stripped down to about 10 mils in length. This response is almost identical to the response of the SMA cable only, except for the

presence of a fixed delay. Comparing these plots shows that the center conductor of the SMA connector adds quite a few unwanted parasitics, but stripping down the conductor greatly reduces the parasitics.

Figure 4–70. Top & Side Views of Original & Modified SMA

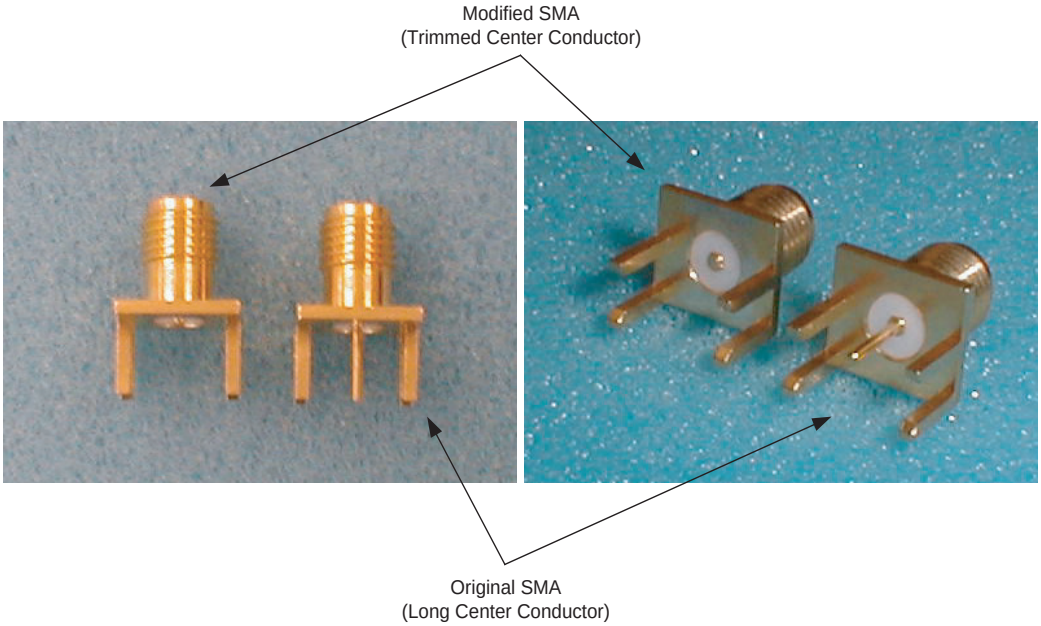
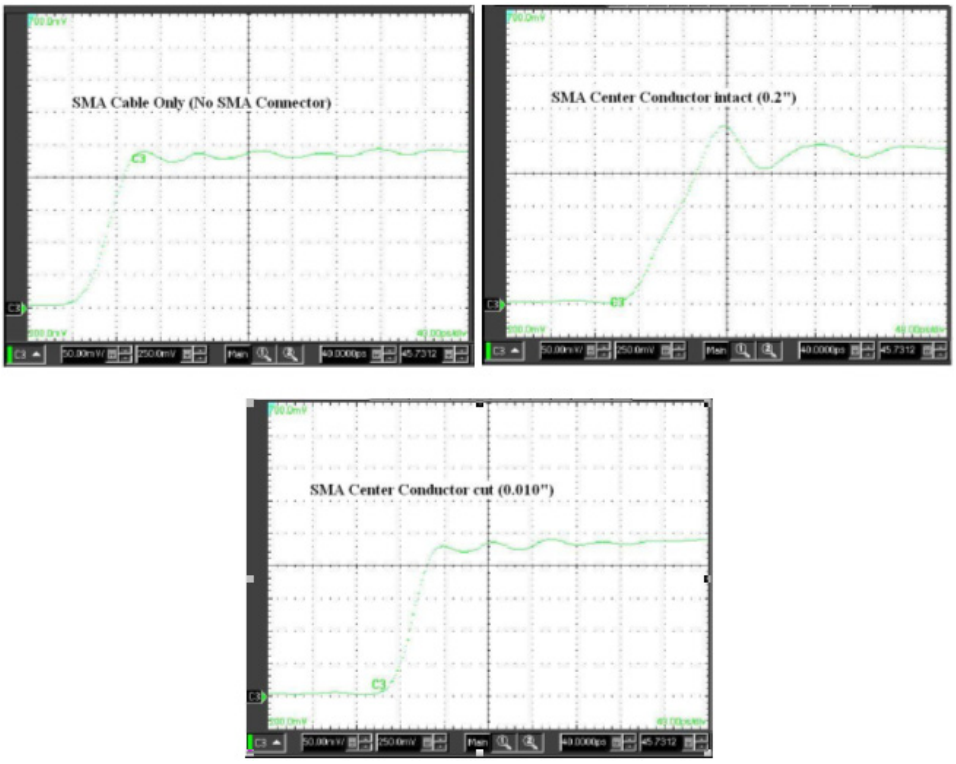


Figure 4–71. Impedance Profiles of SMA Connectors, Measured by TDR



Crosstalk

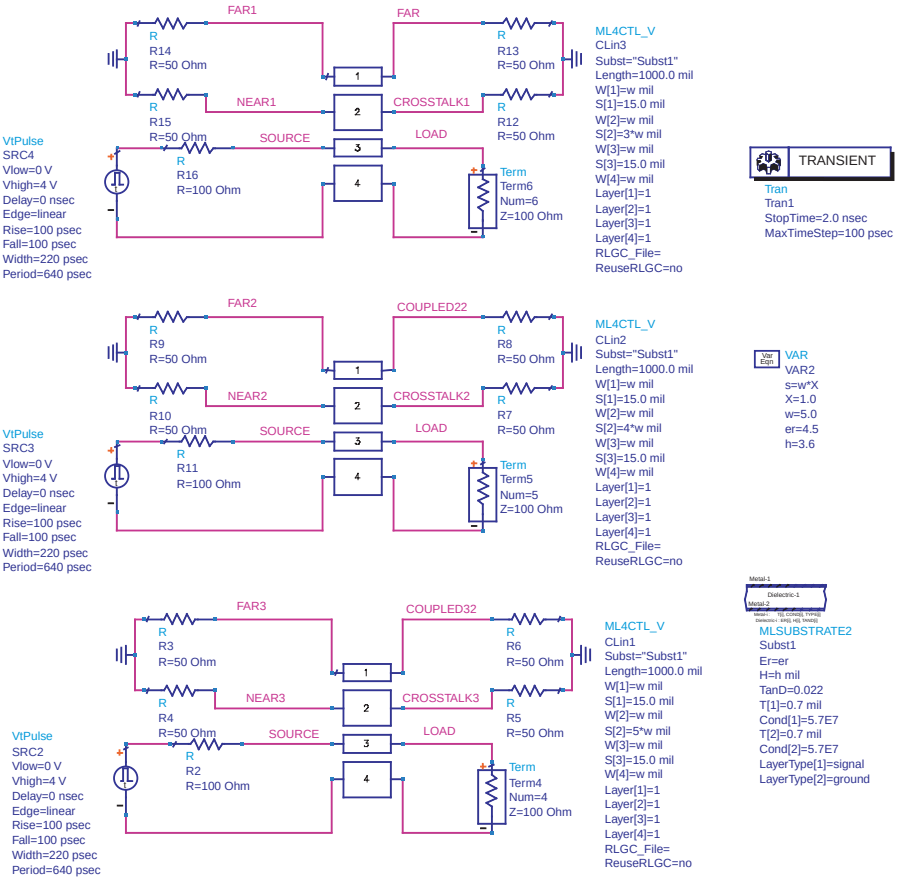
If two traces are close to each other, the signal switching on one trace can excite a voltage on the other, resulting in crosstalk. Crosstalk control must be a key objective of any board design. With high-speed designs it is especially important because of the reduced noise margins.

The following crosstalk case studies are simulations that vary the spacing between the aggressor net and the victim net. Both simulations used differential traces.

Crosstalk Case Study 1 – Loosely Coupled Microstrip Traces
This case study used two pairs of 1-inch long microstrip differential traces. The width (W) of all the traces is 5 mils. The spacing between the positive and negative loosely coupled traces is 15 mils. The aggressor trace had the same configuration, but was placed at varying distances to

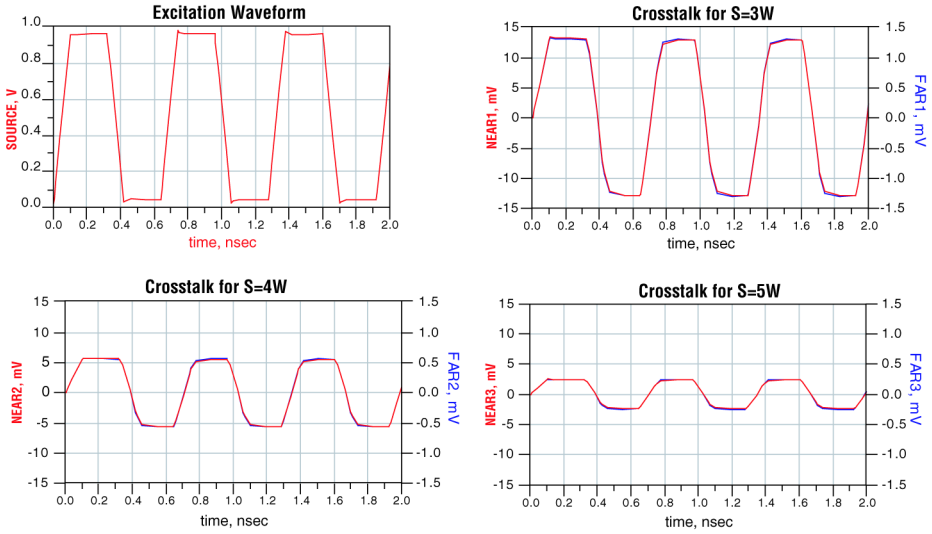
see the effect of spacing on crosstalk. This test setup is based on the microstrip lines routed on the top layer of the Stratix GX development board. The simulation setup is shown in Figure 4–72.

Figure 4–72. ADS Test Setup for Loosely Coupled Microstrip Crosstalk Simulation



The simulation results are shown in Figure 4–73. The input waveform is at the top left. Crosstalk on the near and far victim lines for S = 3W appears in the top right. The bottom left shows crosstalk for S = 4W, and the bottom right shows crosstalk for S = 5W.

Figure 4–73. Crosstalk Results for Loosely Coupled Microstrip



When the aggressor trace is S = 3W away, the crosstalk on the nearest victim line is approximately 13.5 mV, and the crosstalk on the farthest victim line is approximately 0.9 mV. When the spacing is increased to S = 4W, the crosstalk on the nearest victim line decreases to 2.5 mV, and the crosstalk on the farthest victim line decreases to 0.6 mV. The decreases on the farthest line are not significant because the lines are loosely coupled: increasing the spacing by 1W does not significantly increase the overall distance to the farthest line. When the spacing is increased to S = 5W, the crosstalk on the nearest victim line decreases to 1.5 mV and the crosstalk on the farthest victim line decreases to 0.6 mV. These three situations allow you to determine what level of noise you can tolerate on your victim lines. Altera recommends maintaining a minimum S = 4W, which is the guideline followed for the Stratix GX development board.

Crosstalk Case Study 2 – Tightly Coupled Stripline Traces

This case study used two pairs of 1-inch-long tightly coupled stripline traces. This setup replicates the tightly coupled stripline traces of the Stratix GX development board. The width of all traces is 3.7 mils, and the spacing between positive and negative traces is 5.3 mils. The aggressor trace used the same configuration but was placed at varying distances to see the effect of spacing on crosstalk. Figure 4–74 shows the simulation setup, and Figure 4–75 shows the results. In Figure 4–75, the input waveform is on the top left. Crosstalk on the near and far victim lines for S = 3W is shown in the top right. The bottom left shows crosstalk for S = 4W, and the bottom right shows crosstalk for S = 5W.

Figure 4–74. ADS Test Setup for Tightly Coupled Stripline Crosstalk Simulation

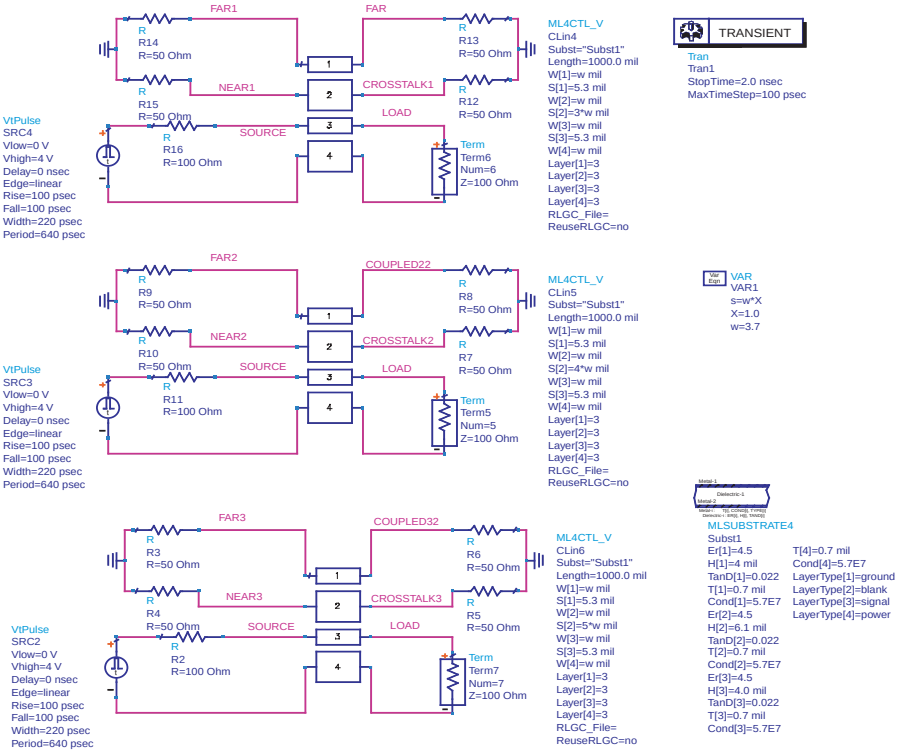
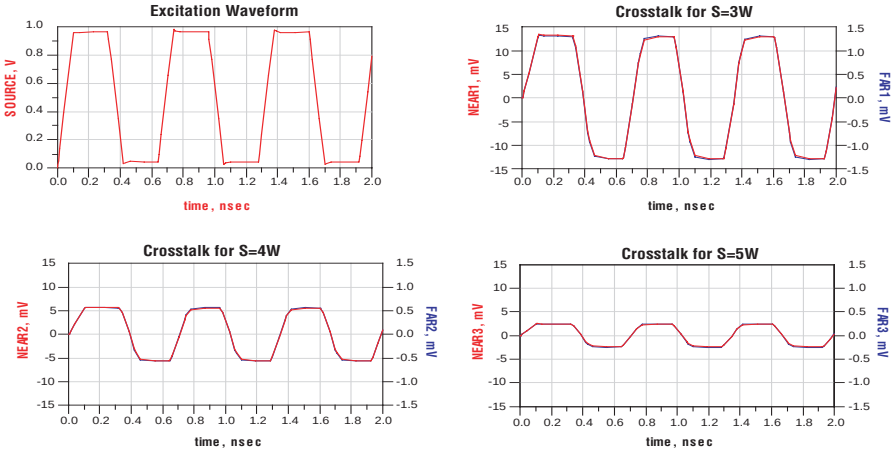


Figure 4–75. Crosstalk Results for Tightly Coupled Stripline



When the aggressor trace is S = 3W away, the crosstalk on the nearest victim line is approximately 13.5 mV, and the crosstalk on the farthest victim line is approximately 1.35 mV. When the spacing is increased to S = 4W, the crosstalk on the nearest victim line decreases to 5.5 mV, and the crosstalk on the farthest victim line decreases to 0.55 mV. The decreases on the farthest line are not significant because the lines are tightly coupled: increasing the spacing by 1W does not significantly increase the overall distance to the farthest line. When the spacing is increased to S = 5W, the crosstalk on the nearest victim line decreases to 2.5 mV, and the crosstalk on the farthest victim line decreases to 0.25 mV. These three situations allow you to determine what level of noise you can tolerate on your victim lines. Altera recommends maintaining a minimum S = 4W, which is the guideline it follows for the Stratix GX development board.

Miscellaneous

This section describes miscellaneous topics, including component selection, S-parameters, the Smith Chart, AC and DC coupling, unused pin connections, and power trace thickness.

Component Selection for High-Speed Design

Deciding which components to use is an important part of board design. This section provides information and guidelines for selecting discrete components for the PCB.

Resistors, Capacitors, Inductors & Ferrite Beads

You should choose the smallest footprint available when selecting discrete components such as resistors and capacitors. The small footprint means that the pad on the board can be small and that the parasitic capacitance and inductance will also be small. Altera typically uses 40 mil × 20 mil (0402) package components for the high-speed signals.

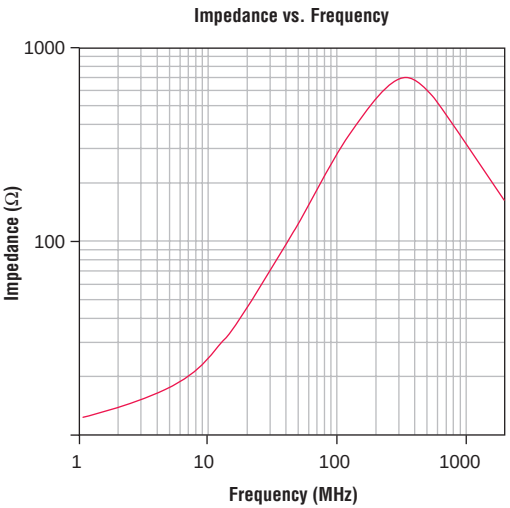
Inductors typically require bigger footprints because they are often used for power supply filtering and must be bigger to support high currents without saturating.

The three parameters of interest when choosing ferrite beads are:

- DC resistance
- AC impedance
- Current handling capability

A good ferrite bead has low DC resistance, high AC impedance, and high current handling capability. However, as the current handling capability increases, the AC impedance tends to drop; consequently there is a trade-off involved. The impedance versus frequency plot of a typical ferrite bead is shown in Figure 4–76.

Figure 4–76. Typical Impedance Profile of a Ferrite Bead



In Figure 4–76, the impedance at 2 GHz is more than 100 Ω The ratio between that impedance and the power supply impedance, which is often lower than 1 Ω, is more than 100. As a result, most of the noise is blocked by the ferrite bead and is shunted to ground instead.

Altera recommends Steward MI0805M221R-00 ferrite beads for transceiver power and ground planes. The DC resistance for this part is lower than 50 mΩ, and it can handle 2.5 A of current. The impedance is over 200 Ω at 1 GHz. Altera also recommends the Murata BLM31PG500SN1 ferrite bead. It has 25 mΩ of DC resistance, 3 A of current, and has 75 Ω of AC impedance at 1 GHz. Two Steward ferrite beads connected in parallel can provide 5 A of current capability with 25 mΩ of DC resistance and over 100 Ω of AC impedance. This performance level is adequate for most applications.

SMA Connectors

SMA connectors are typically used for high-speed signals because of their controlled impedance, mechanical robustness, and good signal integrity. These connectors come in the form of edge launch or vertical launch. For edge launch, the connector is connected to the board edge, the central conductor stays flush with the board, and the board is sandwiched between the ground conductors.

The vertical launch type is through-hole or surface mount. For the through-hole type, all the legs and the center conductor go through the board. For a surface mount type, the center conductor barely touches the top layer of the board.

The type of launch technique does not affect the signal quality very much. However, a long center conductor adds inductance to the transmission path and degrades the signal.

With vertical launch surface mount, you can often strip the center conductor down to below 20 mils. This stripping might be slightly harder to achieve in edge launch configuration, but the choice is dependent on the capability of the SMA manufacturers. Altera has stripped the center conductor down to less than 20 mils short in its designs. Altera recommends using Lighthorse Technologies and Northrop Grumman for SMA connector requirements.

SMA Cables

SMA cables have 18-GHz bandwidth or more, which is sufficient for 3.125-Gbps applications. However, be careful that the cable is crimped securely at both ends to avoid unpredictable behavior.

Probes

Use a good differential probe with short tips for differential signals. Differential probes make the measurement immune to common mode noise and pickup on the probes.

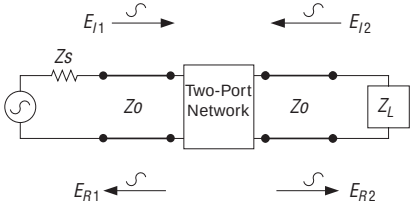
S-Parameters

It is difficult to define voltages or currents in transmission lines and to measure them at microwave frequencies, because direct measurements usually involve the magnitude (inferred from power) and phase of a wave traveling in a given direction, or the standing wave. Thus, equivalent voltages and currents, and the related impedance and admittance matrices, become somewhat of an abstraction when dealing with high-frequency networks.

A representation more consistent with direct measurements, and with the ideas of incident, reflected, and transmitted waves, is provided by the scattering matrix (S-parameters). S-parameters are often used to quantify the impedance, total losses, input return loss, insertion loss, and isolation and crosstalk for the different transmission lines structures. These concepts are most useful at those frequencies where you must consider distributed rather than lumped parameters.

To understand S-parameters, consider the two-port network shown in Figure 4–77. The network can contain a transmission line or any linear time invariant component. The incident voltages at port 1 and 2 are E_{I1} and E_{I2}; and the reflected voltages are E_{R1} and E_{R2}. Z_o is the characteristic impedance of the transmission line. Z_S and Z_L are the source and the load impedances.

Figure 4–77. S-Parameters



Taking the incident and reflected voltage waves on each side of the two-port network and dividing them by the square root of the characteristic impedance, Z_o, results in new variables (a_1, a_2, b_1 and b_2), which are the normalized voltage wave amplitudes at each port.

The square of the magnitude of a_1 ($|a_1|^2$) represents the incident power in port 1, and ($|b_1|^2$) represents the reflected power from this port. The same relations apply to port 2.

$$a_1 = \frac{E_{I1}}{\sqrt{Zo}} \quad a2 = \frac{E_{I2}}{\sqrt{Zo}}$$

$$b_1 = \frac{E_{R1}}{\sqrt{Zo}} \quad b_2 = \frac{E_{I2}}{\sqrt{Zo}}$$

The resulting parameters relate the scattered wave (reflected wave) from the network to the incidents waves. These parameters are called S-parameters and are defined by:

$$b_1 = S_{11}a_1 + S_{12}a_2$$

$$b_2 = S_{21}a_1 + S_{22}a_2$$

where S_{11} is the input reflection coefficient, S_{21} is the forward transmission through the network, S_{12} is the reverse transmission through the network, and S_{22} is the output reflection coefficient.

The measurement setup for S-parameters is shown in Figure 4–78 and Figure 4–79. Apply the following conditions for these measurements:

$$S_{11} = \left.\frac{a_1}{b_1}\right|_{a_2=0}$$

$$S_{21} = \left.\frac{b_2}{a_1}\right|_{a_2=0}$$

$$S_{12} = \left.\frac{b_1}{a_2}\right|_{a_1=0}$$

$$S_{22} = \left.\frac{b_2}{a_2}\right|_{a_1=0}$$

Figure 4–78. Measurement Setup for S_{11} & S_{21}

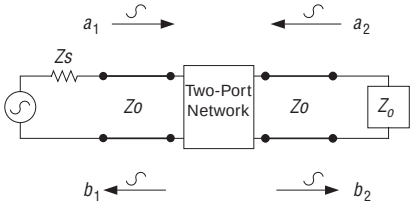
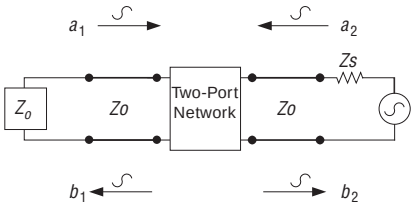


Figure 4–79. Measurement Setup for S_{22} & S_{12}



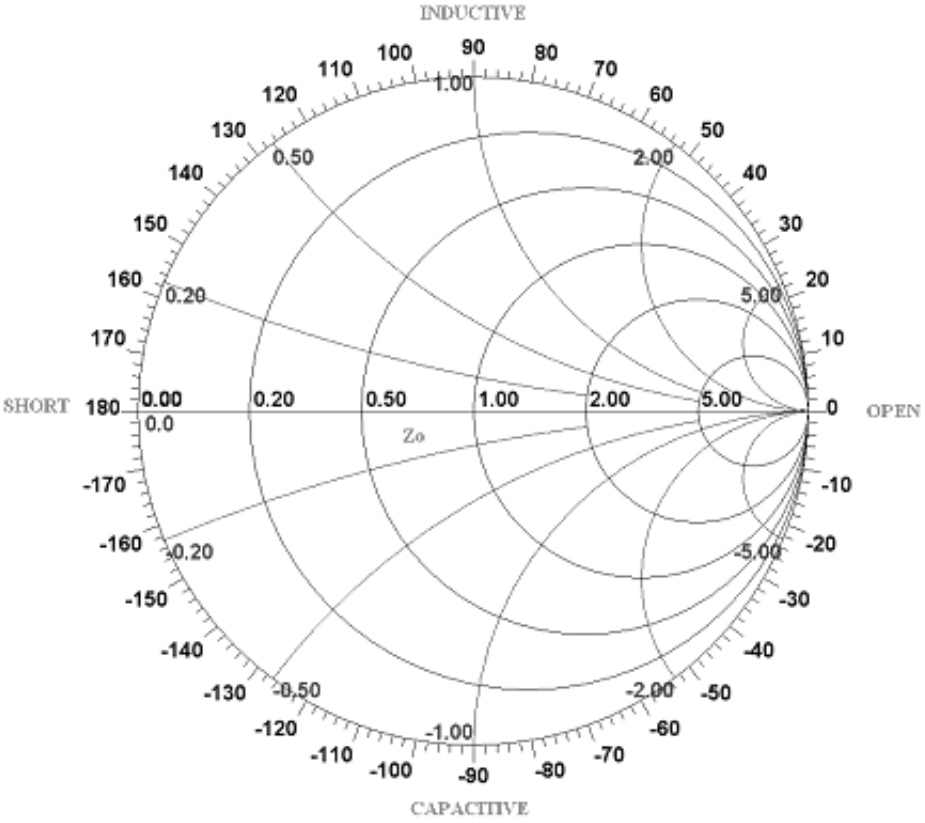
For example, to measure S_{11}, you have to ensure that $a_2 = 0$. The way to do so is by making sure that there is no reflection from the load (in other words, set $Z_L = Z_o$).

S-parameters are typically measured using network analyzers.

Smith Chart

The Smith Chart is a graphical representation of the impedances in a complex plane. You can use this tool to analyze transmission line impedance matching networks and capacitive or inductive behavior of loads. A typical Smith Chart is shown in Figure 4–80. If the impedance is capacitive, it shows up in the top half of the circle; if it is inductive, it shows up in the bottom half. The far-right point in the middle represents an open circuit; the far left represents a short circuit. The middle point (label 1.0) represents a perfect load match to the characteristic impedance of the line.

Figure 4–80. Smith Chart



AC Versus DC Coupling

AC coupling refers to the use of a series capacitor on a signal to block the DC signals from going through. DC coupling refers to the case where this capacitor is not present and the signal passes through without any interruption. In AC coupling, a DC restore circuit is generally required after the capacitor to ensure that the common mode voltage requirements of the receiver are met. Sometimes, as in the Stratix GX transceiver inputs, the DC restore circuitry is built into the device. In that case, external DC restore circuitry is not necessary. DC coupling works only in cases where the output common mode voltage of the transmitter is in the required range of the input common mode voltage of the receiver.

The advantage of AC coupling is that it allows chips with different common mode voltages to interface with each other. The disadvantage is that it requires an extra capacitor, which can add some jitter or other degradation if not properly selected.

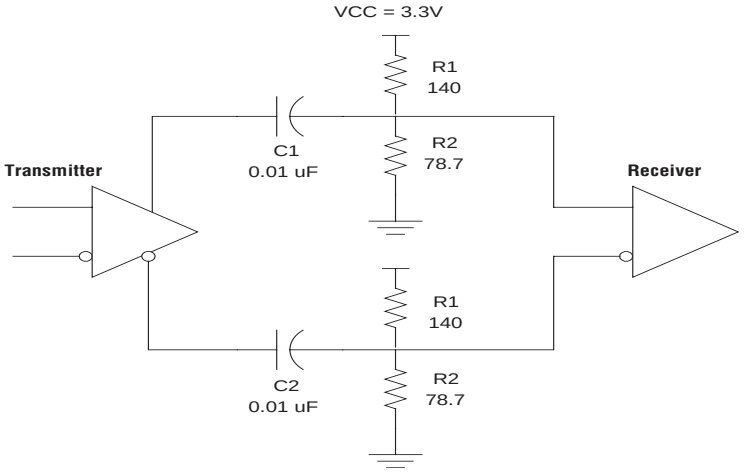
If you are certain that the common mode voltage requirements of the receiver are a subset of the common mode voltage output of the transmitter, use DC coupling. If you are in a borderline case, or if the requirements are not satisfied, then use AC coupling.

In choosing the value of the coupling capacitor, consider what happens if the capacitor is too big or too small. If the capacitor is too big, it can significantly slow the signal down and also responds poorly to fast changing input signals due to the long charge and discharge times. If the capacitor is too small, it presents a fair deal of impedance and can increase attenuation and change the characteristic impedance of the path. A good balance between these two conflicting requirements is a 0.01 µF capacitor, which Altera uses for its 3.125 Gbps transceiver designs.

When selecting components, use the smallest size possible, because smaller size components have smaller size pads, which reduce the discontinuity. Altera has used 0402 components (40 mils by 20 mils) in its designs.

You can design the DC restore circuitry in a variety of ways. Altera typically uses a simple resistive voltage divider (see Figure 4–81). Be sure to use precision resistors (0.1% or 1%) for differential signals, so that the restored DC levels on the positive and negative signals are very closely matched. In Figure 4–81, the DC restore circuitry restores the DC level to 3.3 * 78.7/(140 + 78.7) = 1.1875 Volts.

Figure 4–81. AC Coupling



Stratix GX devices have DC biasing on the high-speed transceivers inputs and reference clock inputs (REFCLKB[17..13]n and REFCLKB[17..13]p) designed for the 1.5-V PCML standard, so AC coupling is not required. This saves components and board space. If you are using other I/O standards such as LVPECL or LVDS, then you need to AC-couple them, because their common mode voltage is different from the 1.5-V PCML common mode voltage. External biasing networks are not needed, because the common mode is generated internally in the device. Altera used external biasing network on the Stratix GX development board for evaluation purpose only.

Unused Pin Connections

The unused I/O pins should be driven to ground. The unused clock inputs should be grounded as well, but they can be left floating. Similarly, the VCC/GND pins for unused PLLs should be tied to their respective supplies and grounds. Refer to the pin tables of the particular devices for more details.

Power Trace Thickness

Power traces must be carefully specified to ensure that they can deliver the required currents to the destination with minimal voltage drop and minimal temperature rise. This requires the traces to have a certain thickness. Calculators are available to calculate the required trace width to support a certain amount of current at a specified temperature rise. You can find one such calculator at

http://www.geocities.com/CapeCanaveral/Lab/9643/TraceWidth.htm. A good layout designer should also have charts and calculators for this purpose.

5. Quartus II Software Fitter Warnings

SGX53002-1.0

Introduction

The Stratix® GX device does not have delay information for the following paths:

- Transmitter PLL input clock to `coreclk_out`
- Reference clock pin (`inclk`) to the transmitter PLL
- Reference clock pin (`rx_cruclk`) to `rx_clkout` (recovered clock)
- Reference clock pin (`rx_cruclk`) to the receiver PLL
- Interquad (IQ) lines (in a multi-transceiver block application, if the clocking scheme utilizes interquad lines for clock distribution or routing, IQ lines are not modeled)

The Fitter in the Quartus® II software generates warning messages for paths that involve these clocks or IQ lines. The warning messages state clearly that the connections are not recommended because accurate delay information is not available.

There are three starting points for the checks:

- Transmitter PLL `coreclk_out`
- Receiver PLL `rx_clkout`
- IQ lines used for clocking the reference clock of transceivers (`refclkb` pins)

The warnings are specific to any path with one of the three starting points. The warnings appear in the Fitter's processing log and are associated with unreliable timing paths that contain the following:

- Incorrect clock to output time (t_{CO})
- Incorrect propagation delay (t_{PD})
- Incorrect register-to-register timing

The warnings generated by the Fitter fall into four categories and are associated with the following:

- Signals that originate from the GXB connected to an IO pin or a signal that originates from the GXB that feeds through a register to an IO pin
- Data that originates from a pin that is clocked by a clock that originates from the GXB

- A clock from one of the starting points feeds a register that is involved in a register-to-register transfer and the destination register is being clocked by a clock related to the starting point
- IQ delay line warnings because the design uses REFCLKB pins for reference clocks when using multiple transceivers in the device

Altera® strongly recommends that you review and resolve the warnings generated by the Fitter in the Quartus II software.

If the starting point is the transmitter PLL, the Fitter in the Quartus II software generates the warning messages for the following configurations:

- t_{PD} warning messages
  The starting point (GXB Transmitter PLL coreclk) feeds an I/O pin directly. This configuration causes the Fitter to incorrectly report t_{PD}. Figure 5–1 shows the t_{PD} (propagation delay) from inclk to the transmitter PLL through coreclk_out to the IO pin. Some example warnings are provided below.

  Warning: Clock connectivity to I/O pins from a GXB transmitter PLL coreclk is not recommended

  Warning: I/O pin coreclk_out is fed by GXB transmitter PLL<*design-specific path*>

- t_{CO} warning messages
  The starting point feeds an I/O register directly. This configuration causes the Fitter to incorrectly report t_{CO}. Clock connectivity to I/O pins through registers originating from a GXB transmitter PLL coreclk is not recommended (see Figure 5–1). In this case, coreclk_out from the transmitter PLL clocks a register (inst3) and the output of that register is connected directly to an output pin. Because coreclk_out is related to inclk and there is no delay information from inclk through the transmitter PLL to coreclk_out, the Fitter reports incorrect t_{CO}.

  The Fitter flags a warning if any similar type of connectivity occurs. An example warning is shown below.

  Warning: Register inst3 clocked by GXB transmitter PLL<*design-specific path*>

- t_{SU} and t_H messages:
  The starting point feeds a register that is involved in a register-to-register transfer from a register that is clocked by a clock related to the starting point. In this configuration, the Fitter performs

incorrect register-to-register analysis. This could result in the Fitter providing incorrect t_{SU} (setup time) and t_H (hold time). Register-to-register transfer between different clock domains is not recommended if one of the clocks is from either the GXB transmitter PLL (`coreclk`) or from the receiver PLL (`rx_clkout`). In the example warning shown below, there is a clock domain crossing between `inclk` (the starting point) and `coreclk_out` (the clock related to the starting point)

Warning: Data is transferred from source register inst4 (clocked by clock `inclk`) to destination register inst3 (clocked by the GXB transmitter PLL<*design-specific path*>.

Figure 5–1 highlights the paths that trigger warning messages from the Fitter. The Fitter reports an incorrect path delay for these paths.

t_{CO} delays are also incorrect for these paths (any register that is being clocked by `coreclk_out`).

The Fitter generates register-to-register warning messages because of the clock domain crossing from `inclk` (input clock to the transmitter PLL) to `coreclk_out` (output of the transmitter PLL).

Figure 5–1. Transmit PLL coreclk_out Showing t_{PD} & t_{CO} Registers

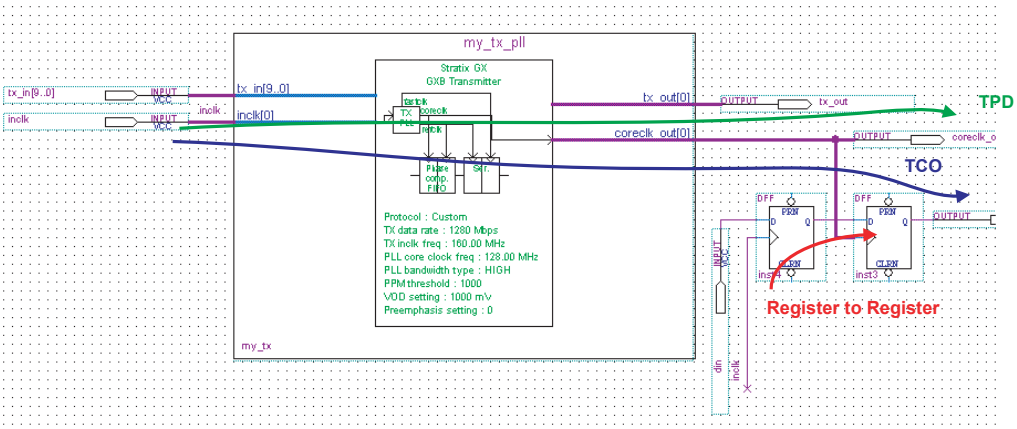


Figure 5–2 shows the t_{PD} (propagation delay) from `rx_cruclk` through the receiver PLL, through `rx_clkout`, and to the I/O pin. The Fitter reports an incorrect path delay for this path.

t_{CO} delays are also incorrect for this path (any register that is clocked by `rx_clkout` (inst3 in Figure 5–2)).

The `rx_clkout` is related to the `rx_cruclk` (receive input reference clock).

The Fitter generates register-to-register warning messages because of the clock domain crossing from `rx_cruclk` (input clock to the receiver PLL) to `rx_clkout` (output of the receiver PLL). This configuration is shown in Figure 5–2, with data transfer from register 1 (inst4) to register 2 (inst3).

Figure 5–2. Receiver PLL rx_clkout Showing t_{PD} & t_{CO} Registers

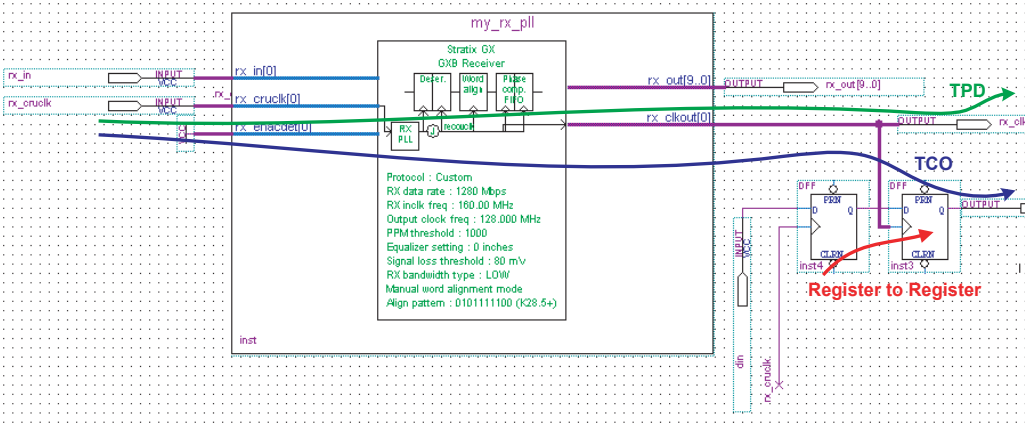
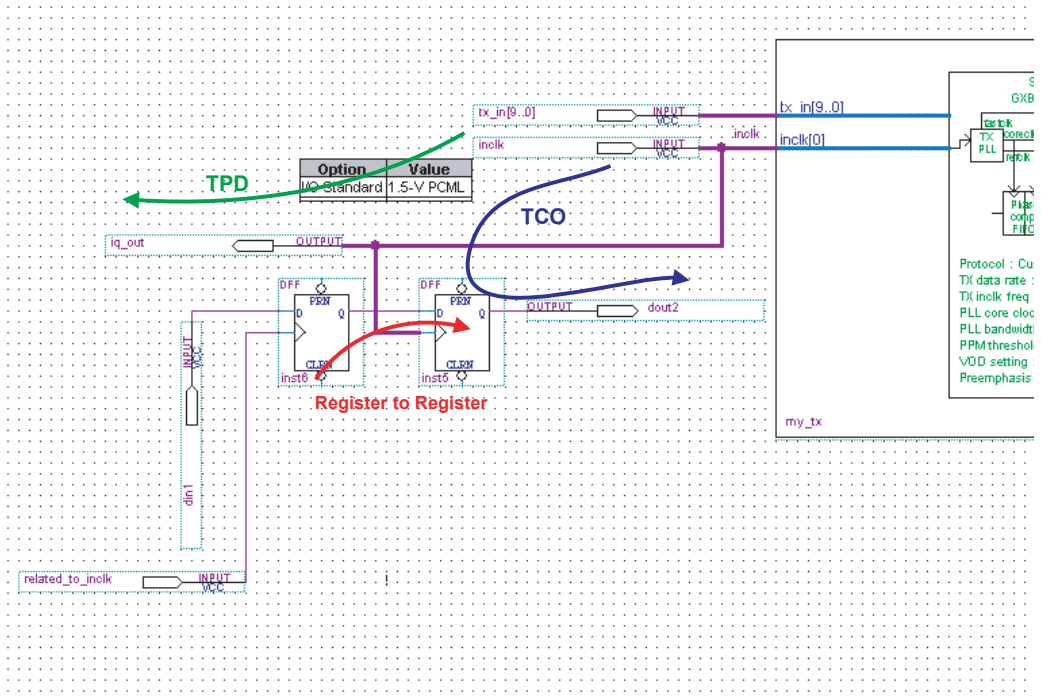


Figure 5–3 shows the usage of IQ lines and multi-transceiver blocks. It shows the t_{PD} (propagation delay) from `inclk` to the `iq_out` output pin. The Fitter reports an incorrect path delay for this path.

The Fitter also reports incorrect t_{CO} (for the path using the same `inclk` as the clock that drives an output pin).

The Fitter also issues a register-register warning because of the clock domain crossing from the clock related to `inclk` to `inclk`.

Figure 5–3. Multi-Transceiver Block Clocking Using IQ Lines Showing t_{PD} & t_{CO} Registers



Suppressing Fitter Warnings

Altera strongly recommends that you review the paths that generate the warning messages. If you decide on this connectivity for design purposes after evaluating the warnings, you can suppress the Fitter warnings.

Use the **Cut** command (in the Assignment Editor for the specific path) to suppress these warnings. The command suppresses the messages in the Fitter and instructs the Timing Analyzer to not perform timing analysis on these paths.

Design Suggestions

This section offers some possible design solutions to help you avoid the paths that cause errors and generate warnings.

Register-to-Register Warnings

If the warning messages generated by the Fitter are of type register-to-register, you should revisit the clock scheme and data path to effectively decouple different clock domains. You do this by clock-decoupling logic and FIFOs as required.

Registers Originating From GXB Connected to IO Pins

Avoid this connectivity as much as possible. If the clocking scheme cannot be changed because of design requirements, then you must resolve this issue at the pin level of the receive device. The receiver on the other end must include some type of dynamic phase adjustment of the clock with respect to the data bus.

In some cases, you might monitor certain GXB signals for test and debug purposes. If you feed these signals to an I/O pin for monitoring, you can suppress the warnings by using the **Cut** command.

Using REFCLKB Pins for Input Reference Clock

If you assign a REFCLKB pin as the input reference clock, the Fitter automatically routes the input using IQ lines. The *Stratix GX Analog Description* chapter of the *Stratix GX Device Handbook, Volume 2* provides information on how IQ lines help in clocking multiple transceivers with one REFCLKB pin and its connectivity across multiple quads. Because of the lack of accurate delay information on IQ lines (used to route the clocks to different transceivers in the device), you should use global clocks for the input reference clock to the transceivers as much as possible (try to avoid using REFCLKB pins). Applications that are sensitive to jitter typically use IQ lines because they are exclusive routing resources for transceivers.

It is not necessary to avoid using REFCLKB pins entirely, but be aware of the issues of using IQ lines for routing input clocks.

Refer to the *REFCKLB Pin Constraints* chapter of the *Stratix GX Device Handbook, Volume 2* before using REFCLKB pins for system clocking.

Refer to the *Stratix GX Transceiver User Guide* section of the *Stratix GX Handbook, Volume 2* for details on the various clocks and functional modes of operation of the Stratix GX device.

You can use decoupling FIFOs to address the lack of delay information. The transmit clock can be an external clock (tx_coreclk) from the PLD or an internal clock (output from the transmitter PLL). On the receive side, it can be either rx_coreclk fed from the PLD or various internal

clocks clocking out the data. In a synchronous system, the decoupling FIFOs must be able to handle phase differences between the clock domains of GXB output clocks (`rx_clkout`) and the PLD system clock. In asynchronous systems, the decoupling FIFOs must also compensate for frequency variations between the recovered clock (`rx_clkout`) and the PLD system clock. See the *Stratix GX Transceiver User Guide* section of the *Stratix GX Device Handbook, Volume 2* for information on possible clocking configurations.

PLD-to-Transceiver Clocking

Figure 5–4 shows a system-level clocking scheme you can use to work around the issues described in this chapter. This clocking scheme may not be the most optimal for designs that have tight jitter requirements because of cascading PLLs.

Figure 5–4. PLD & Transceiver Clocking Scheme