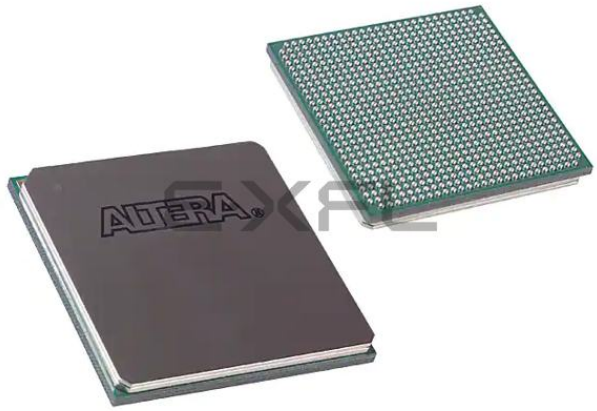



**Welcome to E-XFL.COM**

## Understanding Embedded - FPGAs (Field Programmable Gate Array)

Embedded - FPGAs, or Field Programmable Gate Arrays, are advanced integrated circuits that offer unparalleled flexibility and performance for digital systems. Unlike traditional fixed-function logic devices, FPGAs can be programmed and reprogrammed to execute a wide array of logical operations, enabling customized functionality tailored to specific applications. This reprogrammability allows developers to iterate designs quickly and implement complex functions without the need for custom hardware.

## Applications of Embedded - FPGAs

The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications.

| Details | |
|---|---|
| Product Status | Obsolete |
| Number of LABs/CLBs | 2700 |
| Number of Logic Elements/Cells | 67500 |
| Total RAM Bits | 2699264 |
| Number of I/O | 488 |
| Number of Gates | - |
| Voltage - Supply | 0.86V ~ 1.15V |
| Mounting Type | Surface Mount |
| Operating Temperature | 0°C ~ 85°C (TJ) |
| Package / Case | 780-BBGA, FCBGA |
| Supplier Device Package | 780-FBGA (29x29) |
| Purchase URL | https://www.e-xfl.com/product-detail/intel/ep3sl70f780c3n |

# 1. Stratix III Device Family Overview

SIII51001-1.8

The Stratix® III family provides one of the most architecturally advanced, high-performance, low-power FPGAs in the marketplace.

Stratix III FPGAs lower power consumption through Altera's innovative Programmable Power Technology, which provides the ability to turn on the performance where needed and turn down the power consumption for blocks not in use. Selectable Core Voltage and the latest in silicon process optimizations are also employed to deliver the industry's lowest power, high-performance FPGAs.

Specifically designed for ease of use and rapid system integration, the Stratix III FPGA family offers two variants optimized to meet different application needs:

- The Stratix III *L* family provides balanced logic, memory, and multiplier ratios for mainstream applications.
- The Stratix III *E* family is memory- and multiplier-rich for data-centric applications.

Modular I/O banks with a common bank structure for vertical migration lend efficiency and flexibility to the high-speed I/O. Package and die enhancements with dynamic on-chip termination, output delay, and current strength control provide best-in-class signal integrity.

Based on a 1.1-V, 65-nm all-layer copper SRAM process, the Stratix III family is a programmable alternative to custom ASICs and programmable processors for high-performance logic, digital signal processing (DSP), and embedded designs.

Stratix III devices include optional configuration bit stream security through volatile or non-volatile 256-bit Advanced Encryption Standard (AES) encryption. Where ultra-high reliability is required, Stratix III devices include automatic error detection circuitry to detect data corruption by soft errors in the configuration random-access memory (CRAM) and user memory cells.

## Features Summary

Stratix III devices offer the following features:

- 48,000 to 338,000 equivalent logic elements (LEs) (refer to Table 1–1)
- 2,430 to 20,497 Kbits of enhanced TriMatrix memory consisting of three RAM block sizes to implement true dual-port memory and FIFO buffers
- High-speed DSP blocks provide dedicated implementation of 9×9, 12×12, 18×18, and 36×36 multipliers (at up to 550 MHz), multiply-accumulate functions, and finite impulse response (FIR) filters
- I/O:GND:PWR ratio of 8:1:1 along with on-die and on-package decoupling for robust signal integrity
- Programmable Power Technology, which minimizes power while maximizing device performance

Figure 4–5 shows the address clock enable waveform during the write cycle for M9K and M144K.

**Figure 4–5.** Stratix III Address Clock Enable during Write Cycle Waveform for M9K and M144K

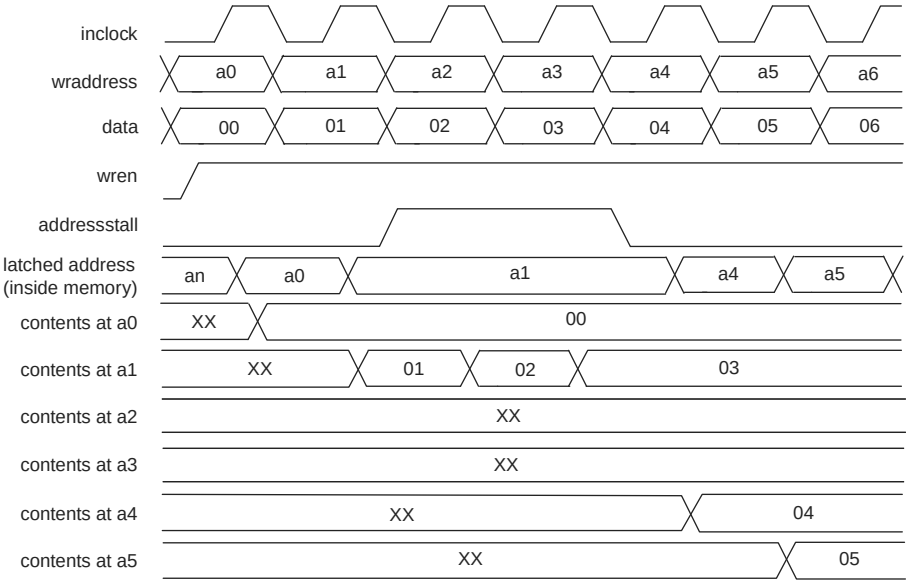
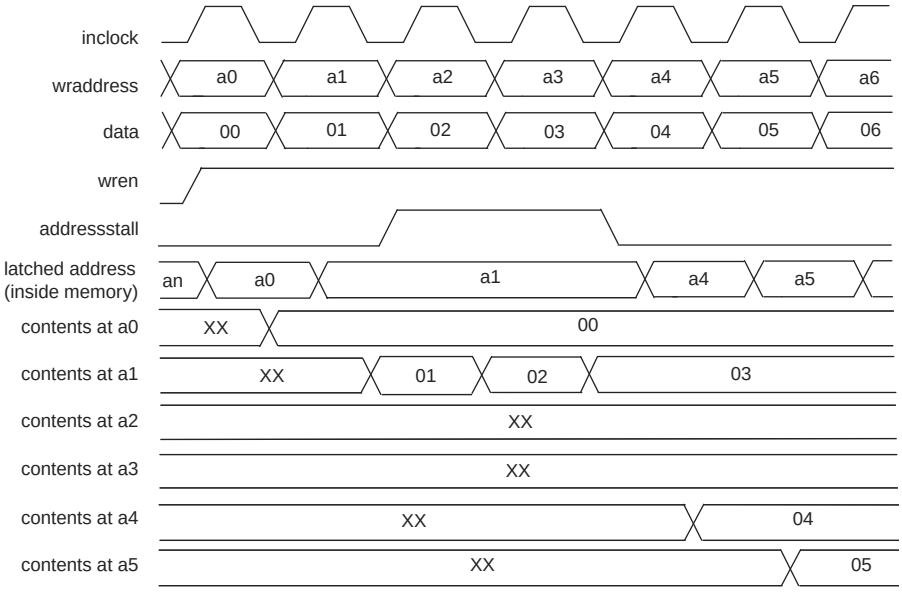


Figure 4–6 shows the address clock enable waveform during the write cycle for MLABs.

**Figure 4–6.** Stratix III Address Clock Enable during Write Cycle Waveform for MLABs




## Mixed Width Support

M9K and M144K memory blocks inherently support mixed data widths. MLABs can support mixed data widths through emulation via the Quartus II software. When using simple dual-port or true dual-port mixed width support allows you to read and write different data widths to a memory block. Refer to "Memory Modes" on page 4–10 for details on the different widths supported per memory mode.

☞ You cannot use the ECC on M144 memory blocks when using the mixed width support.

☞ MLABs do not support mixed-width FIFO mode.

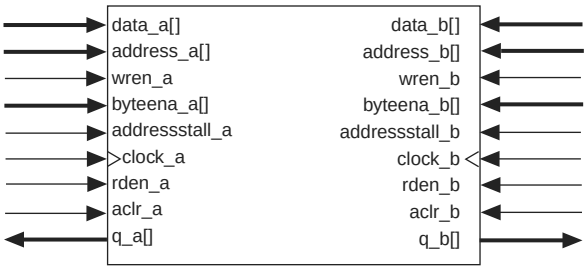
## Asynchronous Clear

Stratix III M9K and M144K memory blocks support asynchronous clears on the output latches and output registers. MLABs supports asynchronous clear on the output registers only as the output is not latched. Therefore, if your M9K and M144K are not using the output registers, you can still clear the RAM outputs via the output latch asynchronous clear. The functional waveform in Figure 4–7 shows this functionality.

## True Dual-Port Mode

Stratix III M9K and M144K blocks support true dual-port mode. Sometimes called bi-directional dual-port, this mode allows you to perform any combination of two port operations: two reads, two writes, or one read and one write at two different clock frequencies. Figure 4–15 shows the true dual-port RAM configuration.

**Figure 4–15.** Stratix III True Dual-Port Memory *(Note 1)*




**Note to Figure 4–15:**

(1) True dual-port memory supports input/output clock mode in addition to the independent clock mode shown.

The widest bit configuration of the M9K and M144K blocks in true dual-port mode is as follows:

- 512 × 16-bit (×18-bit with parity) (M9K)
- 4K × 32-bit (×36-bit with parity) (M144K)

Wider configurations are unavailable because the number of output drivers is equivalent to the maximum bit width of the respective memory block. Because true dual-port RAM has outputs on two ports, its maximum width equals half of the total number of output drivers. Table 4–7 lists the possible M9K block mixed-port width configurations in true dual-port mode.

**Table 4–7.** Stratix III M9K Block Mixed-Width Configuration (True Dual-Port Mode)

| Read Port | Write Port | | | | | | |
|---|---|---|---|---|---|---|---|
| | 8K×1 | 4K×2 | 2K×4 | 1K×8 | 512×16 | 1K×9 | 512×18 |
| 8K×1 | ✓ | ✓ | ✓ | ✓ | ✓ | — | — |
| 4K×2 | ✓ | ✓ | ✓ | ✓ | ✓ | — | — |
| 2K×4 | ✓ | ✓ | ✓ | ✓ | ✓ | — | — |
| 1K×8 | ✓ | ✓ | ✓ | ✓ | ✓ | — | — |
| 512×16 | ✓ | ✓ | ✓ | ✓ | ✓ | — | — |
| 1K×9 | — | — | — | — | — | ✓ | ✓ |
| 512×18 | — | — | — | — | — | ✓ | ✓ |

A single DSP block can implement up to two independent 44-bit accumulators.

The dynamic `accum_sload` control signal is used to clear the accumulation. A `logic 1` value on the `accum_sload` signal synchronously loads the accumulator with the multiplier result only, while a `logic 0` enables accumulation by adding or subtracting the output of the DSP block (accumulator feedback) to the output of the multiplier and first-stage adder.

☞ The control signal for the accumulator and subtractor is static and therefore has to be configured at compile time.

This mode supports the round and saturation logic unit as it is configured as an 18-bit multiplier accumulator. You can use the pipeline registers and output registers within the DSP block to increase the performance of the DSP block.

## Shift Modes

Stratix III devices support the following shift modes for 32-bit input only:

- Arithmetic shift left, `ASL[N]`
- Arithmetic shift right, `ASR[32-N]`
- Logical shift left, `LSL[N]`
- Logical shift right, `LSR[32-N]`
- 32-bit rotator or Barrel shifter, `ROT[N]`

☞ You can switch the shift mode between these modes using the dynamic rotate and shift control signals.

The shift mode in a Stratix III device can be easily used by the soft embedded processor such as Nios® II to perform the dynamic shift and rotate operation. Figure 5–20 shows the shift mode configuration.

The shift mode makes use of the available multipliers to logically or arithmetically shift left, right, or rotate the desired 32-bit data. The DSP block is configured like the independent 36-bit multiplier mode to perform the shift mode operations.

The arithmetic shift right requires signed input vector. During arithmetic shift right, the sign is extended to fill the MSB of the 32-bit vector. The logical shift right uses unsigned input vector. During logical shift right, zeros are padded in the most significant bits shifting the 32-bit vector to the right. The barrel shifter uses unsigned input vector and implements a rotation function on a 32-bit word length.

Two control signals `rotate` and `shift_right` together with the `signa` and `signb` signals, determining the shifting operation. Examples of shift operations are listed in Table 5–5 on page 5–31.

**Table 5–10.** DSP Block Dynamic Signals (Part 2 of 2)

| Signal Name | Function | Count |
|---|---|---|
| ena0<br>ena1<br>ena2<br>ena3 | Input and Pipeline Register enable signals | 4 |
| aclr0<br>aclr1<br>aclr2<br>aclr3 | DSP block-wide asynchronous clear signals (active low). | 4 |
| — | Total Count per Full Block | 34 |

# Application Examples

## FIR Example

A finite impulse response filter is a common function used in many systems to perform spectral manipulations. The basic form is shown in Equation 5–6.

**Equation 5–6.** Finite Impulse Response Filter Equation

$$y(n) = \sum_{k=0}^{N-1} x(n-k) \times c(k)$$

In this equation, x(n) is the input samples to the filter, c(k) are the filter coefficients, and y(n) are the filtered output samples. Typically, the coefficients do not change in time in most applications such as Digital Down Converters (DDC). FIR filters can be implemented in many forms, the most simple being the tap-delay line approach.

Stratix III DSP block can implement various types of FIR filters very efficiently. To form the tap-delay line, the input register stage of the DSP block has the ability to cascade the input in a chained fashion in 18-bit wide format. Unlike the Stratix II DSP block, which has two built-in parallel input register scan paths, Stratix III supports only one built-in 18-bit parallel input register scan path for 288 data input.

For a pair of 18-bit input buses, the A input for the first 18-bit bus is fed back to be registered again at the input of the second (lower) pair of inputs. Refer to Figure 5–22 for details.

The B input of the multiplier feeds from the general routing. You can scan in the data in 18-bit parallel form and multiply it by the 18-bit input bus from general routing in each cycle.

When you use **both** the input cascade and chainout features, the DSP block uses an 18-bit delay register in the boundary of each half-DSP block or from block-to-block to synchronize the input scan chain data with the chainout data. The top half computes the sum of product and chains the output to the next block after the output register. The output register uses the delay register to delay the cascade input by one clock cycle to compensate the latency for the bottom half.

For applications in which the system clock is slower than the speed of the DSP block, the multipliers can be time-multiplexed to improve efficiency. This makes multi-channel and semi-parallel FIR structures possible. The structure to achieve this is similar to Figure 5–22 and Figure 5–23. The main difference is that the input cascade chain is no longer used and each half-DSP block is used in Four-Multiplier Mode with independent inputs. Figure 5–24 shows an example for chained cascaded summation.

In most cases, only the final stage FIR tap with the rounding and saturation unit is deployed.

### Bypassing PLL

Bypassing a PLL counter results in a multiply (m counter) or a divide (n and C0 to C9 counters) factor of one.

Table 6–20 lists the settings for bypassing the counters in Stratix III PLLs.

**Table 6–20.** PLL Counter Settings

| PLL Scan Chain Bits [0..10] Settings | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **LSB** *(2)* | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | **[7]** | **[8]** | **[9]** | **MSB** *(1)* | **Description** |
| 0 | X | X | X | X | X | X | X | X | X | 1 *(3)* | PLL counter bypassed |
| X | X | X | X | X | X | X | X | X | X | 0 *(3)* | PLL counter not bypassed because bit 10 (MSB) is set to 0 |

**Notes to Table 6–20:**

(1) Most significant bit (MSB).
(2) Least significant bit (LSB).
(3) Counter-bypass bit.

To bypass any of the PLL counters, set the bypass bit to 1. The values on the other bits are ignored. To bypass the VCO post-scale counter (K), set the corresponding bit to 1.

### Dynamic Phase-Shifting

The dynamic phase-shifting feature allows the output phases of individual PLL outputs to be dynamically adjusted relative to each other and to the reference clock without the need to send serial data through the scan chain of the corresponding PLL. This feature simplifies the interface and allows you to quickly adjust clock-to-out ($t_{CO}$) delays by changing the output clock phase-shift in real time. This adjustment is achieved by incrementing or decrementing the VCO phase-tap selection to a given C counter or to the M counter. The phase is shifted by 1/8 of the VCO frequency at a time. The output clocks are active during this phase-reconfiguration process.

Table 6–21 lists the control signals that are used for dynamic phase-shifting.

**Table 6–21.** Dynamic Phase-Shifting Control Signals (Part 1 of 2)

| Signal Name | Description | Source | Destination |
|---|---|---|---|
| PHASECOUNTERSELECT [3:0] | Counter select. Four bits decoded to select either the M or one of the C counters for phase adjustment. One address maps to select all C counters. This signal is registered in the PLL on the rising edge of SCANCLK. | Logic array or I/O pins | PLL reconfiguration circuit |
| PHASEUPDOWN | Selects dynamic phase shift direction; 1= UP; 0= DOWN. Signal is registered in the PLL on the rising edge of SCANCLK. | Logic array or I/O pin | PLL reconfiguration circuit |
| PHASESTEP | Logic high enables dynamic phase shifting. | Logic array or I/O pin | PLL reconfiguration circuit |

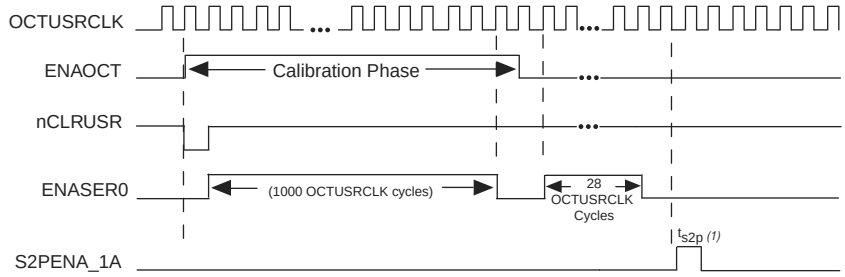
## OCT Calibration

Figure 7–18 shows the user-mode signal-timing waveforms. To calibrate OCT block[N] (where N is a calibration block number), you must assert ENAOCT one cycle before asserting ENASER[N]. Also, nCLRUSR must be set to low for one OCTUSRCLK cycle before ENASER[N] signal is asserted. An asserted ENASER[N] signals for 1000 OCTUSRCLK cycles to perform $OCTR_S$ and $OCTR_T$ calibration. ENAOCT can be deasserted one clock cycle after the last ENASER is deasserted.

## Serial Data Transfer

When calibration is complete, you must serially shift out the 28-bit OCT calibration code (14-bit OCT RS code and 14-bit OCT RT) from each OCT calibration block to the corresponding I/O buffers. Only one OCT calibration block can send out the codes at any given time by asserting only one ENASER[N] signal at a time. After ENAOCT is deasserted, you must wait at least 1 OCTUSRCLK cycle to enable any ENASER[N] signal to begin serial transfer. To shift 28-bit code from OCT calibration block[N], ENASER[N] must be asserted for exactly 28 OCTUSRCLK cycles. There must be at least one OCTUSRCLK cycle gap between two consecutive asserted ENASER signals. For these requirements, refer to Figure 7–18.

**Figure 7–18.** OCT User-Mode Signal Timing Waveform for One OCT Block




**Note to Figure 7–18:**

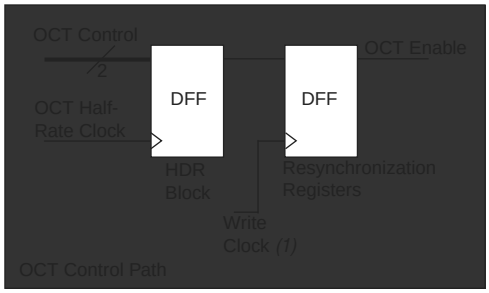
(1) ts2p ≥ 25 ns

After calibrated codes are shifted serially to the corresponding I/O buffers, they must be converted from serial format to parallel format before being used in the I/O buffers. Figure 7–18 shows S2PENA signals that can be asserted at any time to update the calibration codes in each I/O bank. All I/O banks that received the codes from the same OCT calibration block can have S2PENA asserted at the same time, or at a different time, even while another OCT calibration block is calibrating and serially shifting codes. The S2PENA signal is asserted one OCTUSRCLK cycle after ENASER is deasserted for at least 25 ns. You cannot use I/Os for transmitting or receiving data when their S2PENA is asserted for parallel codes transfer.

## Example of Using Multiple OCT Calibration Blocks

Figure 7–19 shows a signal timing waveform for two OCT calibration blocks doing $R_S$ and $R_T$ calibration. Calibration blocks can start calibrating at different times by asserting ENASER signals at different times. ENAOCT must stay asserted while any calibration is ongoing. nCLRUSR must be set to low for one OCTUSRCLK cycle before each ENASER[N] signal is asserted. In Figure 7–19, when nCLRUSR is set to 0 for the second time to initialize OCT calibration block 0, this does not affect OCT calibration block 1, whose calibration is already in progress.

**Figure 8–19.** Stratix III Dynamic OCT Control Block




**Note to Figure 8–19:**

(1) The write clock comes from either the PLL or the write leveling delay chain.

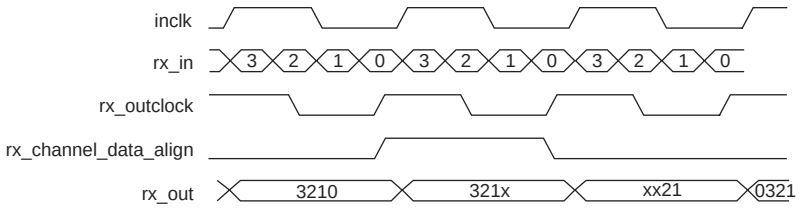
## IOE Registers

The IOE registers have been expanded to allow source-synchronous systems to have faster register-to-register transfers and resynchronization. Both top/bottom and left/right IOEs have the same capability with left/right IOEs having extra features to support LVDS data transfer.

Figure 8–20 shows the registers available in the Stratix III input path. The input path consists of the DDR input registers, resynchronization registers, and HDR block. You can bypass each block of the input path.

Figure 9–7 shows receiver output (RX_OUT) after one bit slip pulse with the deserialization factor set to **4**.

**Figure 9–7.** Data Realignment Timing



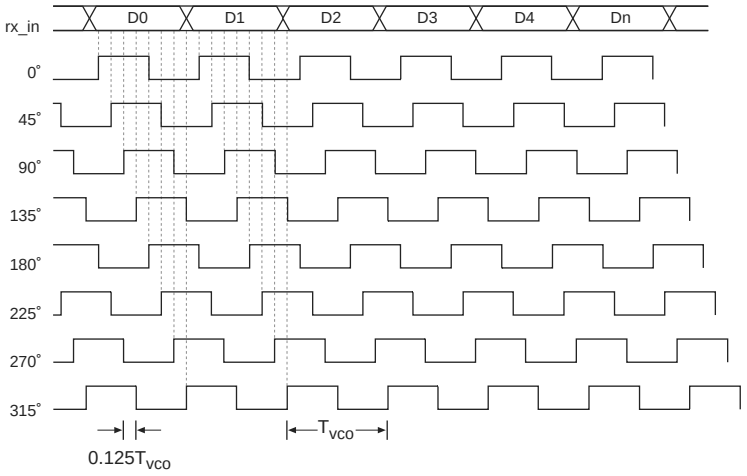

The data realignment circuit can have up to 11 bit-times of insertion before a rollover occurs. The programmable bit rollover point can be from 1 to 11 bit-times, independent of the deserialization factor. An optional status port, RX_CDA_MAX, is available to the FPGA from each channel to indicate when the preset rollover point is reached.

## Dynamic Phase Aligner (DPA)

The DPA block takes in high-speed serial data from the differential input buffer and selects one of the eight phase clocks from the left/right PLL to sample the data. The DPA chooses the phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is 1/8 UI, which is the maximum quantization error of the DPA. The eight phases of the clock are equally divided, giving a 45° resolution.

Figure 9–8 shows the possible phase relationships between the DPA clocks and the incoming serial data.

**Figure 9–8.** DPA Clock Phase-to-Serial Data Timing Relationship




The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if required. You can prevent the DPA from selecting a new clock phase by asserting the optional RX_DPLL_HOLD port, which is available for each channel.

**Figure 9–22.** Invalid Placement of DPA-Disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center Left/Right PLLs

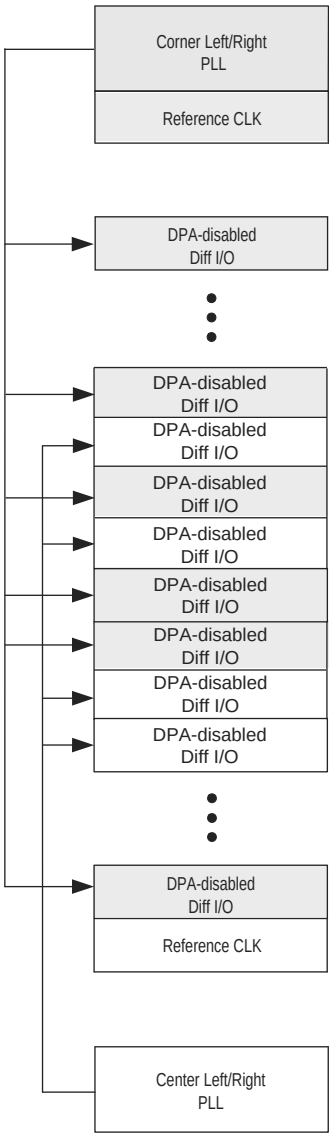
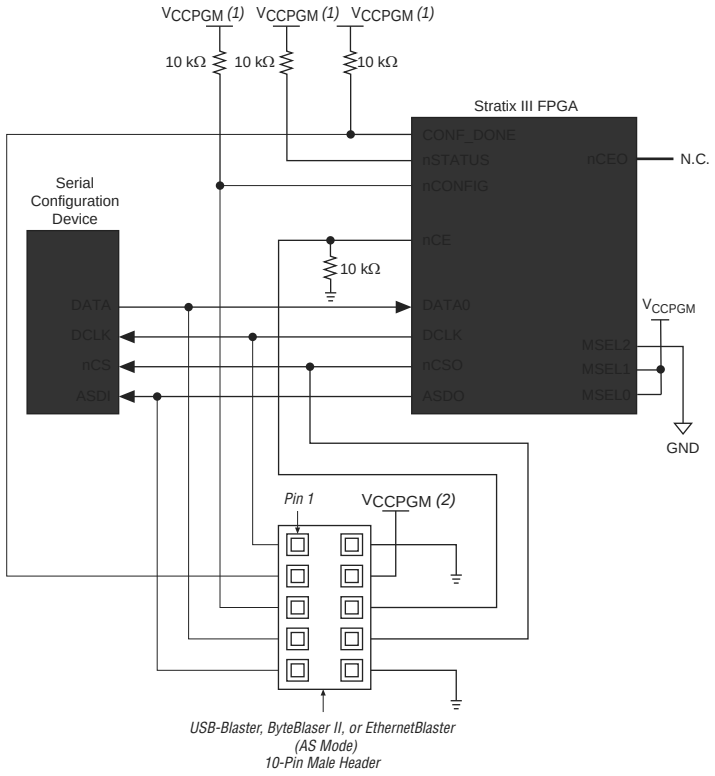


**Figure 11–12.** In-System Programming of Serial Configuration Devices




**Notes to Figure 11–12:**

(1) Connect the pull-up resistors to $V_{CCPGM}$ at 3.3-V supply.

(2) Power up the USB-Blaster, ByteBlaster II, or EthernetBlaster cable's $V_{CC(TRGT)}$ with $V_{CCPGM}$.

You can program serial configuration devices with the Quartus II software using the Altera programming hardware and the appropriate configuration device programming adapter.

In production environments, you can program serial configuration devices using multiple methods. You can use Altera programming hardware or other third-party programming hardware to program blank serial configuration devices before they are mounted onto PCBs. Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

You can program a serial configuration device in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming that can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data (.**rpd**) file and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for EPCS Programming* and the source code on the Altera website at www.altera.com.

**Table 11–10.** PS Timing Parameters for Stratix III Devices (Part 2 of 2)

| Symbol | Parameter | Minimum | Maximum | Units |
|---|---|---|---|---|
| t | Input fall time | — | 40 | ns |
| $t_{CD2UM}$ | CONF_DONE high to user mode *(2)* | 20 | 100 | µs |
| $t_{CD2CU}$ | CONF_DONE high to CLKUSR enabled | 4 × maximum DCLK period | — | — |
| $t_{CD2UMC}$ | CONF_DONE high to user mode with CLKUSR option on | tCD2CU + (4,436 × CLKUSR period) | — | — |

**Notes to Table 11–10:**

(1) This value is applicable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(2) The minimum and maximum numbers apply only if you choose the internal oscillator as the clock source for starting the device.

Device configuration options and how to create configuration files are discussed further in the *Device Configuration Options and Configuration File Formats* chapters in volume 2 of the *Configuration Handbook*.

## PS Configuration Using a Microprocessor

In this PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Stratix III device.

You can do a PS configuration using MicroBlaster™ Passive Serial Software Driver. For more information, refer to *AN423: Configuring the MicroBlaster Passive Serial Software Driver*.

For all configuration and timing information, refer to "PS Configuration Using a MAX II Device as an External Host" on page 11–27. This section is also applicable when using a microprocessor as an external host.

## PS Configuration Using a Download Cable

In this section, the generic term *download cable* includes the Altera USB-Blaster USB port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster II parallel port download cable, ByteBlasterMV™ parallel port download cable, and the EthernetBlaster download cable.

In PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device by using the USB-Blaster, MasterBlaster, ByteBlaster II, EthernetBlaster, or ByteBlasterMV cable.

**Table 11–14.** Dedicated Configuration Pins on the Stratix III Device (Part 5 of 5)

| Pin Name | User Mode | Configuration Scheme | Pin Type | Description |
|---|---|---|---|---|
| DCLK *(1)* | N/A | Synchronous configuration schemes (PS, FPP, AS) | Input (PS, FPP)<br>Output (AS) | DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active.<br>In AS mode, DCLK is an output from the Stratix III device that provides timing for the configuration interface. After AS configuration, this pin is driven to an inactive state. In schemes that use a configuration device, DCLK will be 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. |
| DATA0 *(1)* | N/A in AS mode. I/O in PS or FPP mode | PS, FPP, AS | Input | Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin.<br>In AS mode, DATA0 has an internal pull-up resistor that is always active.<br>After PS or FPP configuration, DATA0 is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. |
| DATA[7..1] | I/O | Parallel configuration schemes (FPP) | Inputs | Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0].<br>In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated.<br>After configuration, DATA[7..1] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings. |

**Note to Table 11–14:**

(1) To tri-state AS configuration pins in AS configuration scheme, turn on **Enable input tri-state on active configuration pins in user mode** option from the **Device and Pin Options** dialog box. This tri-states DCLK, nCSO, Data0, and ASDO pins. Dual-purpose Pins Setting for Data0 is ignored. To set Data0 to a different setting, for example to use Data0 pin as a regular I/O in user mode, turn off **Enable input tri-state on active configuration pins in user mode** option and set your desired setting from the Dual-purpose Pins Setting menu.

# 14. Design Security in Stratix III Devices

SIII51014-1.5

## Introduction

This chapter provides an overview of the design security feature and its implementation on Stratix® III devices using advanced encryption standard (AES) as well as security modes available in Stratix III devices.

As Stratix III devices start to play a role in larger and more critical designs in competitive commercial and military environments, it is increasingly important to protect the designs from copying, reverse engineering, and tampering. Stratix III devices address these concerns and are the industry's only high-density, high-performance devices with both volatile and non-volatile security feature support. Stratix III devices have the ability to decrypt configuration bitstreams using the AES algorithm, an industry standard encryption algorithm that is FIPS-197 certified. They also have a design security feature that utilizes a 256-bit security key.

Altera® Stratix III devices store configuration data in static random access memory (SRAM) configuration cells during device operation. Because SRAM memory is volatile, SRAM cells must be loaded with configuration data each time the device powers-up. It is possible to intercept configuration data when it is being transmitted from the memory source (flash memory or a configuration device) to the device. The intercepted configuration data could then be used to configure another device.

When using the Stratix III design security feature, the security key is stored in the Stratix III device. Depending on the security mode, you can configure the Stratix III device using a configuration file that is encrypted with the same key, or for board testing, configured with a normal configuration file.

The design security feature is available when configuring Stratix III devices using the fast passive parallel (FPP) configuration mode with an external host (such as a MAX® II device or microprocessor), or when using fast active serial (AS) or passive serial (PS) configuration schemes. However, the design security feature is also available in remote update with fast AS configuration mode. The design security feature is not available when you are configuring your Stratix III device using Joint Test Action Group (JTAG)-based configuration. For more information, refer to "Supported Configuration Schemes" on page 14–5.

## Stratix III Security Protection

Stratix III device designs are protected from copying, reverse engineering, and tampering using configuration bitstream encryption.

### Security Against Copying

The security key is securely stored in the Stratix III device and cannot be read out through any interfaces. In addition, as configuration file read-back is not supported in Stratix III devices, the design information cannot be copied.

**Figure 15–1.** Error Detection Block Diagram

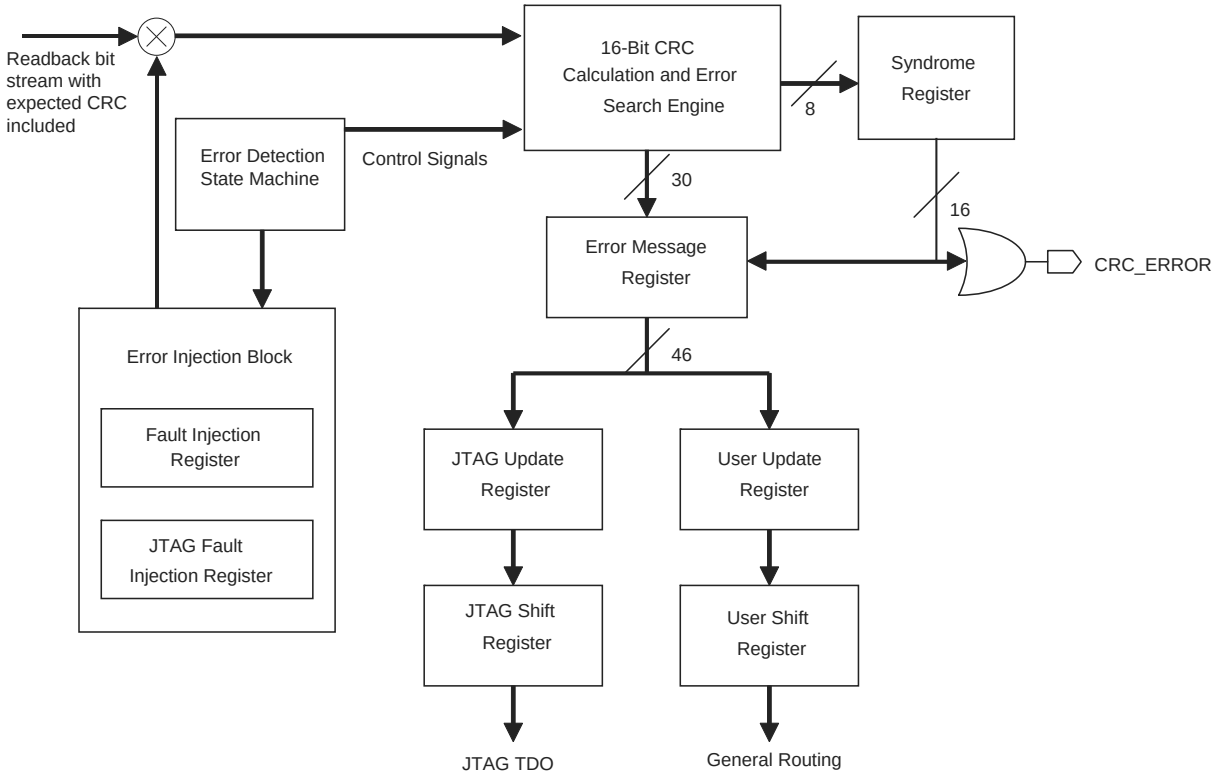



Table 15–5 lists the registers shown in Figure 15–1.

**Table 15–4.** Error Detection Registers (Part 1 of 2)

| Register | Description |
|---|---|
| Syndrome Register | This register contains the CRC signature of the current frame through the error detection verification cycle. The CRC_ERROR signal is derived from the contents of this register. |
| Error Message Register | This 46-bit register contains information about the error type, location of the error, and the actual syndrome. The types of errors and location reported are single and double adjacent bit errors. The location bits for other types of errors are not identified by the Error Message Register. You can shift out the content of the register through the JTAG SHIFT_EDERROR_REG instruction or to the core through the core interface. |
| JTAG Update Register | This register is automatically updated with the contents of the Error Message Register one cycle after the 46-bit register content is validated. It includes a clock enable which needs to be asserted prior to being sampled into the JTAG Shift Register. This requirement ensures that the JTAG Update Register is not being written into by the contents of the Error Message Register at exactly the same time that the JTAG Shift Register is reading its contents. |
| User Update Register | This register is automatically updated with the contents of the Error Message Register, one cycle after the 46-bit register content is validated. It includes a clock enable which needs to be asserted prior to being sampled into the User Shift Register. This requirement ensures that the User Update Register is not being written into by the contents of the Error Message Register at exactly the same time that the User Shift Register is reading its contents. |

**Table 15–4.** Error Detection Registers (Part 2 of 2)

| Register | Description |
|---|---|
| JTAG Shift Register | This register is accessible by the JTAG interface and allows the contents of the JTAG Update Register to be sampled and read out by the JTAG instruction SHIFT_EDERROR_REG. |
| User Shift Register | This register is accessible by the core logic and allows the contents of the User Update Register to be sampled and read by the user logic. |
| JTAG Fault Injection Register | This 21-bit register is fully controlled by the JTAG instruction EDERROR_INJECT. This register holds the information of the error injection that you want in the bitstream. |
| Fault Injection Register | The content of the JTAG Fault Injection Register is loaded in this 21-bit register when it is being updated. |

# Error Detection Timing

When the CRC feature is enabled through the Quartus II software, the device automatically activates the CRC process upon entering user mode, after configuration, and after initialization is complete.

If an error is detected within a frame, CRC_ERROR is driven high at the end of the error location search, and after the Error Message Register gets updated. At the end of this cycle, the CRC_ERROR pin is pulled low for a minimum 32 clock cycles. If the next frame also contains an error, the CRC_ERROR is driven high again after the Error Message Register gets overwritten by the new value. You can start to unload the error message on each rising edge of CRC_ERROR pin. The error detection runs until the device is reset.

Error detection circuitry runs off an internal configuration oscillator with a divisor that sets the maximum frequency. Table 15–5 lists the minimum and maximum error detection frequencies.

**Table 15–5.** Minimum and Maximum Error Detection Frequencies

| Device Type | Error Detection Frequency | Maximum Error Detection Frequency | Minimum Error Detection Frequency | Valid Exponents (*n*) |
|---|---|---|---|---|
| Stratix III | 100 MHz / $2^n$ | 50 MHz | 390 kHz | 1, 2, 3, 4, 5, 6, 7, 8 |

You can set a lower clock frequency by specifying a division factor in the Quartus II software (refer to "Software Support" on page 15–11). The divisor is a power of two (2), where *n* is between 1 and 8. The divisor ranges from 2 through 256. Refer to Equation 15–1.

**Equation 15–1.**

$$\text{Error detection frequency} = \frac{100\,MHz}{2^n}$$

The error detection frequency reflects the frequency of the error detection process for a frame because the CRC calculation in Stratix III devices is done on a per-frame basis.

# Recovering From CRC Errors

The system that contains the Stratix III device must control the device reconfiguration. After detecting an error on the CRC_ERROR pin, strobing the nCONFIG signal low directs the system to perform the reconfiguration at a time when it is safe for the system to reconfigure the device.

When the data bit is rewritten with the correct value by reconfiguring the device, the device functions correctly.

While soft errors are uncommon in Altera devices, certain high-reliability applications may require a design to account for these errors.

# Chapter Revision History

Table 15–8 lists the revision history for this chapter.

**Table 15–8.** Chapter Revision History

| Date | Version | Changes Made |
|---|---|---|
| March 2010 | 1.7 | Updated for the Quartus II software version 9.1 SP2 release:<br>■ Updated Table 15–6.<br>■ Minor text edits. |
| May 2009 | 1.6 | Updated "User Mode Error Detection" and "CRC_ERROR Pin" sections. |
| February 2009 | 1.5 | ■ Updated "Error Detection Timing" section.<br>■ Removed "Referenced Documents", Critical Error Detection", and "CRITICAL ERROR Pin" sections. |
| October 2008 | 1.4 | ■ Updated "Introduction" and "Referenced Documents" sections.<br>■ Updated New Document Format. |
| May 2008 | 1.3 | ■ Updated "Configuration Error Detection", "User Mode Error Detection", and "Error Detection Timing" sections.<br>■ Updated Table 15–3, Table 15–6, and Table 15–7.<br>■ Updated Figure 15–2 and Figure 15–3. |
| October 2007 | 1.2 | ■ Minor edits to Table 15–3.<br>■ Added new section "Referenced Documents".<br>■ Added live links for references. |
| May 2007 | 1.1 | ■ Minor edits to page 2, 3, 4, and 14.<br>■ Updated Table 15–5. |
| November 2006 | 1.0 | Initial Release. |

# Section V. Power and Thermal Management

This section provides information on Power and Thermal Management for the Stratix® III devices.

- Chapter 16, Programmable Power and Temperature-Sensing Diodes in Stratix III Devices

## Revision History

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