The MAX® 10 devices consist of the following:

- Logic array blocks (LABs)
- Analog-to-digital converter (ADC)
- User flash memory (UFM)
- Embedded multiplier blocks
- Embedded memory blocks (M9K)
- Clocks and phase-locked loops (PLL)
- General purpose I/O
- High-speed LVDS I/O
- External memory interfaces
- Configuration flash memory (CFM)

now part of Intel

Figure 1: Typical Device Floorplan for MAX 10 Devices

- The amount and location of each block varies in each MAX 10 device.
- Certain MAX 10 devices may not contain a specific block.

Related Information

- **MAX 10 Device Datasheet**
  Provides more information about specification and performance for MAX 10 devices.
- **MAX 10 FPGA Device Overview**
  Provides more information about maximum resources in MAX 10 devices
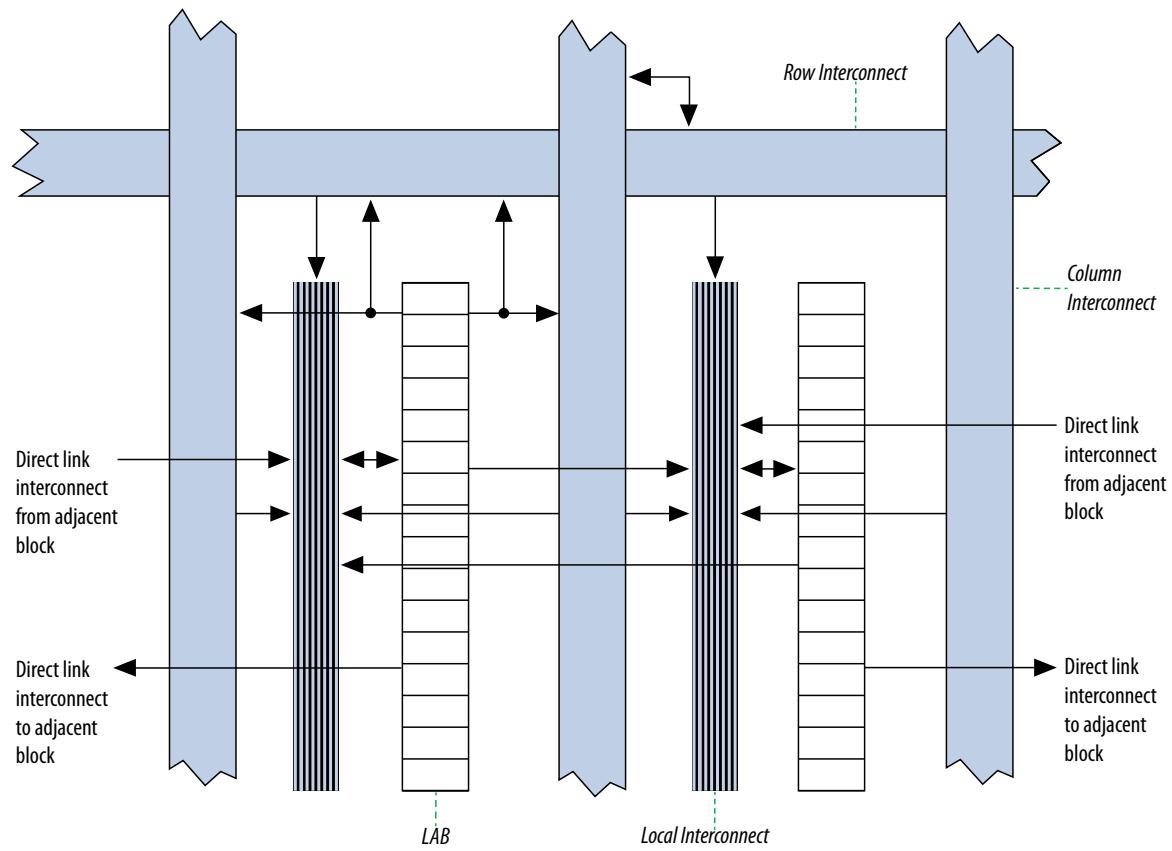
Logic Array Block

The LABs are configurable logic blocks that consist of a group of logic resources.

Each LAB consists of the following:

- 16 logic elements (LEs)—smallest logic unit in MAX 10 devices
- LE carry chains—carry chains propagated serially through each LE within a LAB
- LAB control signals—dedicated logic for driving control signals to LEs within a LAB
- Local interconnect—transfers signals between LEs in the same LAB
- Register chains—transfers the output of one LE register to the adjacent LE register in a LAB

Figure 2: LAB Structure of MAX 10 Devices

The Quartus® Prime Compiler places associated logic in a LAB or adjacent LABs, allowing the use of local and register chain connections for performance and area efficiency.
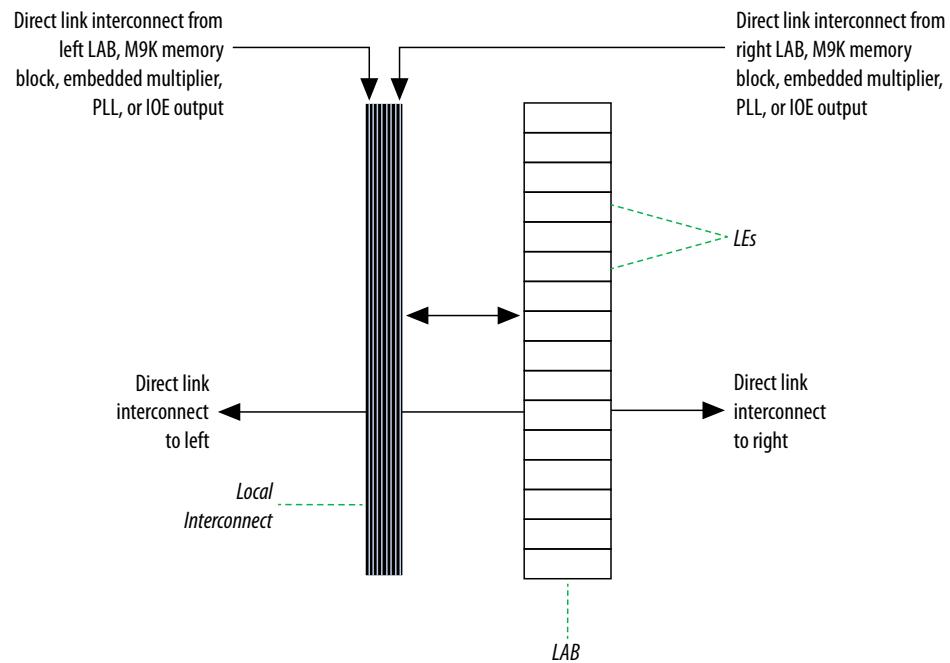
LAB Interconnects

The LAB local interconnect is driven by column and row interconnects and LE outputs in the same LAB.

The direct link connection minimizes the use of row and column interconnects to provide higher performance and flexibility. The direct link connection enables the neighboring elements from left and right to drive the local interconnect of a LAB. The elements are:

- LABs
- PLLs
- M9K embedded memory blocks
- embedded multipliers

Each LE can drive up to 48 LEs through local and direct link interconnects.

Figure 3: LAB Local and Direct Link Interconnects for MAX 10 Devices

LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its LEs.

The control signals include:

- Two clock signals
- Two clock enable signals
- Two asynchronous clear signals
- One synchronous clear signal
- One synchronous load signal

Figure 4: LAB-Wide Control Signals for MAX 10 Devices
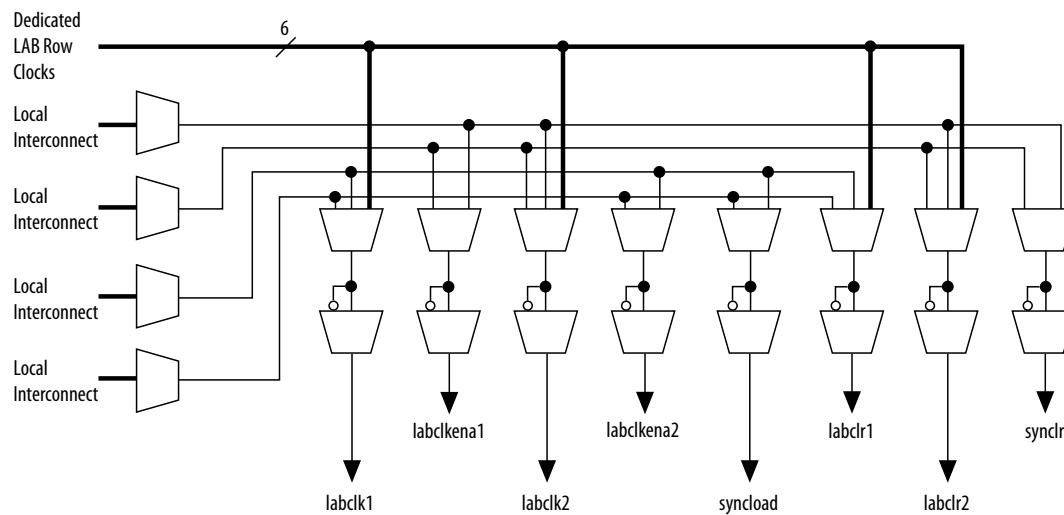
Table 1: Control Signal Descriptions for MAX 10 Devices

| Control Signal	Description
labclk1 labclk2	• Each LAB can use two clocks signals. The clock and clock enable signals of each LAB are linked. For example, any LE in a particular LAB using the labclk1 signal also uses the labclkena1 signal. • If the LAB uses both the rising and falling edges of a clock, it also uses both LAB-wide clock signals. • The LAB row clocks [5..0] and LAB local interconnect generate the LAB-wide clock signals. The MultiTrack interconnect inherent low skew allows clock and control signal distribution in addition to data distribution.
labclkena1 labclkena2	• Each LAB can use two clock enable signals. The clock and clock enable signals of each LAB are linked. For example, any LE in a particular LAB using the labclk1 signal also uses the labclkena1 signal. • Deasserting the clock enable signal turns off the LAB-wide clock signal.
labclr1 labclr2	Asynchronous clear signals: • LAB-wide control signals that control the logic for the clear signal of the register. • The LE directly supports an asynchronous clear function.
syncload synclr	Synchronous load and synchronous clear signals: • Can be used for implementing counters and other functions • LAB-wide control signals that affect all registers in the LAB

You can use up to eight control signals at a time. Register packing and synchronous load cannot be used simultaneously.

Each LAB can have up to four non-global control signals. You can use additional LAB control signals as long as they are global signals.

A LAB-wide asynchronous load signal to control the logic for the preset signal of the register is not available. The register preset is achieved with a NOT gate push-back technique. MAX 10 devices only support either a preset or asynchronous clear signal.

In addition to the clear port, MAX 10 devices provide a chip-wide reset pin (DEV_CLRn) to reset all registers in the device. An option set before compilation in the Quartus Prime software controls this pin. This chip-wide reset overrides all other control signals.

Logic Elements

LE is the smallest unit of logic in the MAX 10 device family architecture. LEs are compact and provide advanced features with efficient logic usage.
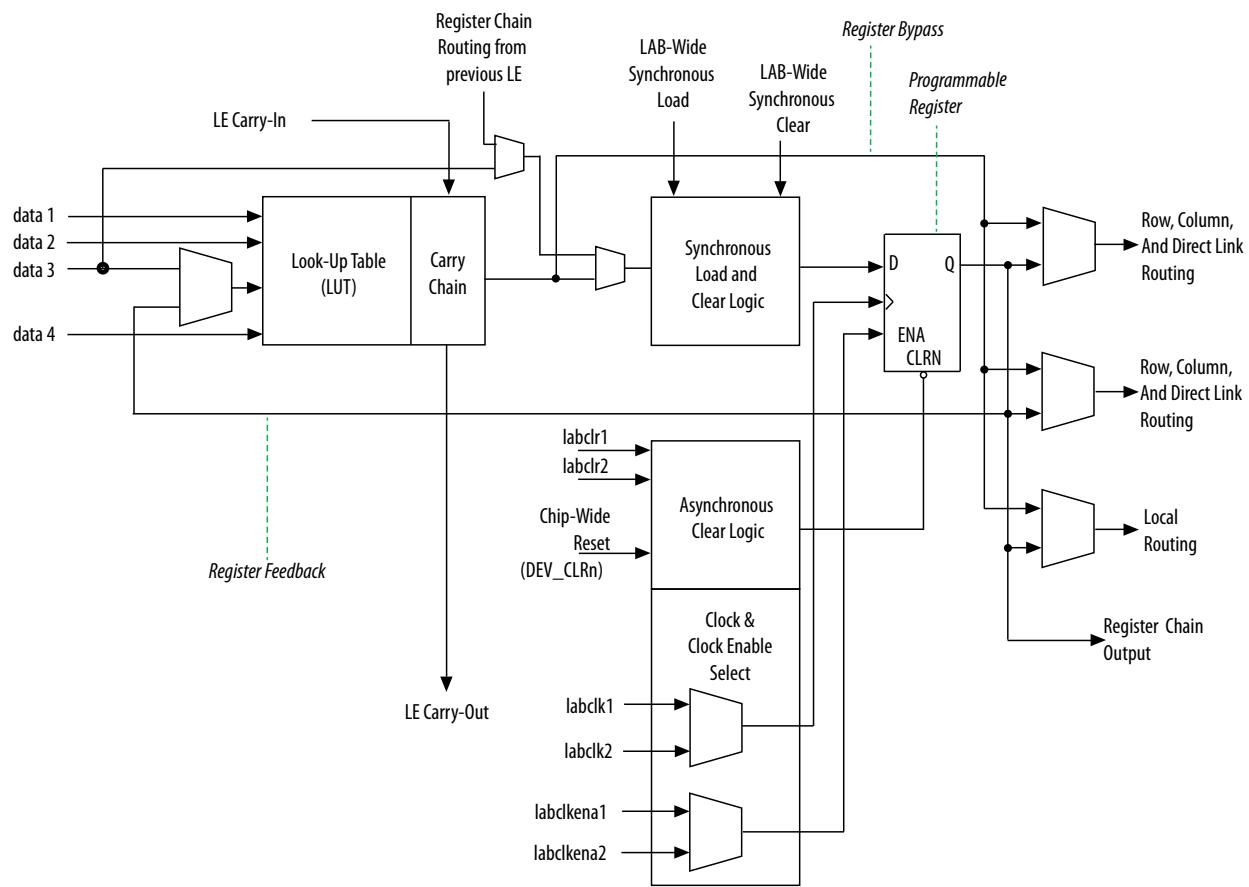
Each LE has the following features:

- A four-input look-up table (LUT), which can implement any function of four variables
- A programmable register
- A carry chain connection
- A register chain connection
- The ability to drive the following interconnects:

  - local
  - row
  - column
  - register chain
  - direct link
- Register packing support
- Register feedback support

LE Features

LEs contain inputs, outputs and registers to enable several features.

Figure 5: LE High-Level Block Diagram for MAX 10 Devices.

LE Inputs

There are six available inputs to the LE in both mode LE operating modes, Normal Mode and Arithmetic Mode. Each input is directed to different destinations to implement the desired logic function. The LE inputs are:

- four data inputs from the LAB local interconnect
- an LE carry-in from the previous LE carry-chain
- a register chain connection

LE Outputs

Each LE has three outputs which are:

- Two LE outputs drive the column or row and direct link routing connections
- One LE output drives the local interconnect resources

The register packing feature is supported in MAX 10 devices. With register packing, the LUT or register output drives the three outputs independently. This feature improves device utilization by using the register and the LUT for unrelated functions.

The LAB-wide synchronous load control signal is not available when using register packing.

Register Chain Output

Each LE has a register chain output which allows registers in the same LAB to cascade together. This feature speed up connections between LABs and optimize local interconnect resources by allowing the following:

- LUTs to be used for combinational functions
- registers to be used for an unrelated shift register implementation

Programmable Register

You can configure the programmable register of each LE for D, T, JK, or SR flipflop operation. Each register has the following inputs:

- clock—can be driven by signals that use the global clock network, general-purpose I/O pins or the internal logic
- clear—can be driven by signals that use the global clock network, general-purpose I/O pins or the internal logic
- clock enable—can be driven by general-purpose I/O pins or the internal logic

For combinational functions, the LUT output bypasses the register and drives directly to the LE outputs.

Register Feedback

The register feedback mode allows the register output to feed back into the LUT of the same LE. This is to ensure that the register is packed with its own fan-out LUT which provides another mechanism for improved fitting. The LE can also drive out registered and unregistered versions of the LUT output.

LE Operating Modes

The LEs in MAX 10 devices operate in two modes.

- Normal mode
- Arithmetic mode

These operating modes use LE resources differently. Both LE modes have six available inputs and LAB-wide signals.

The Quartus Prime software automatically chooses the appropriate mode for common functions, such as counters, adders, subtractors, and arithmetic functions, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions.
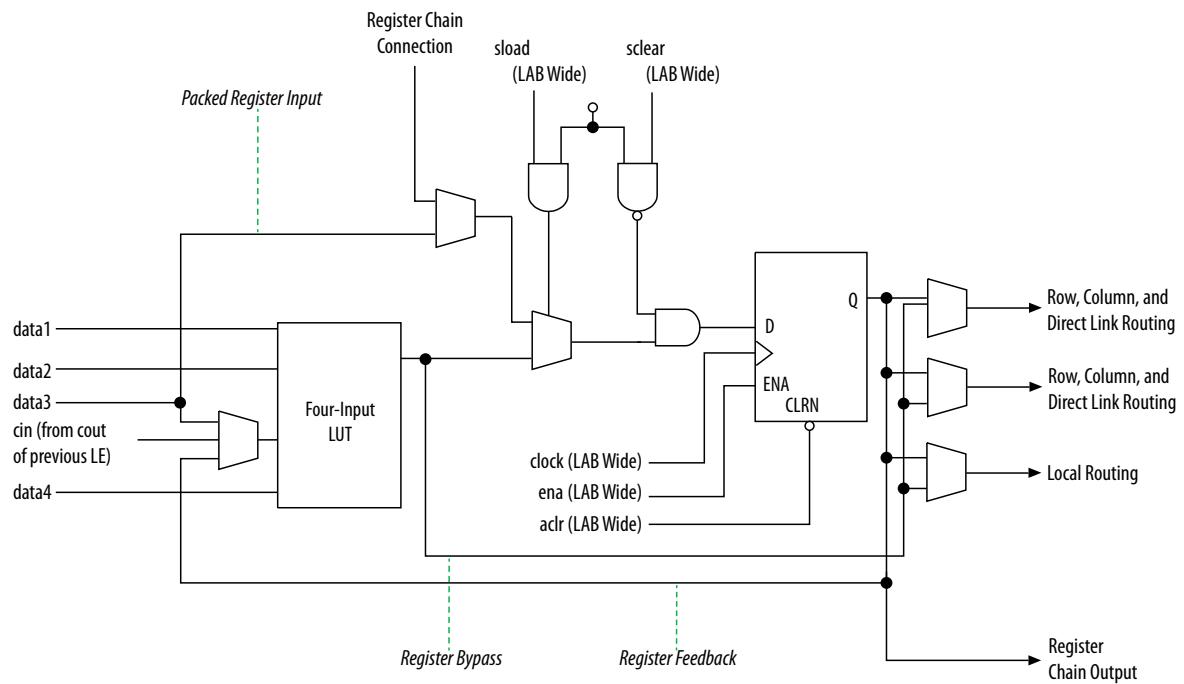
You can also create special-purpose functions that specify which LE operating mode to use for optimal performance.

Normal Mode

Normal mode is suitable for general logic applications and combinational functions.

In normal mode, four data inputs from the LAB local interconnect are inputs to a four-input LUT. The Quartus Prime Compiler automatically selects the carry-in (cin) or the data3 signal as one of the inputs to the LUT. LEs in normal mode support packed registers and register feedback.

Figure 6: LE Operating in Normal Mode for MAX 10 devices
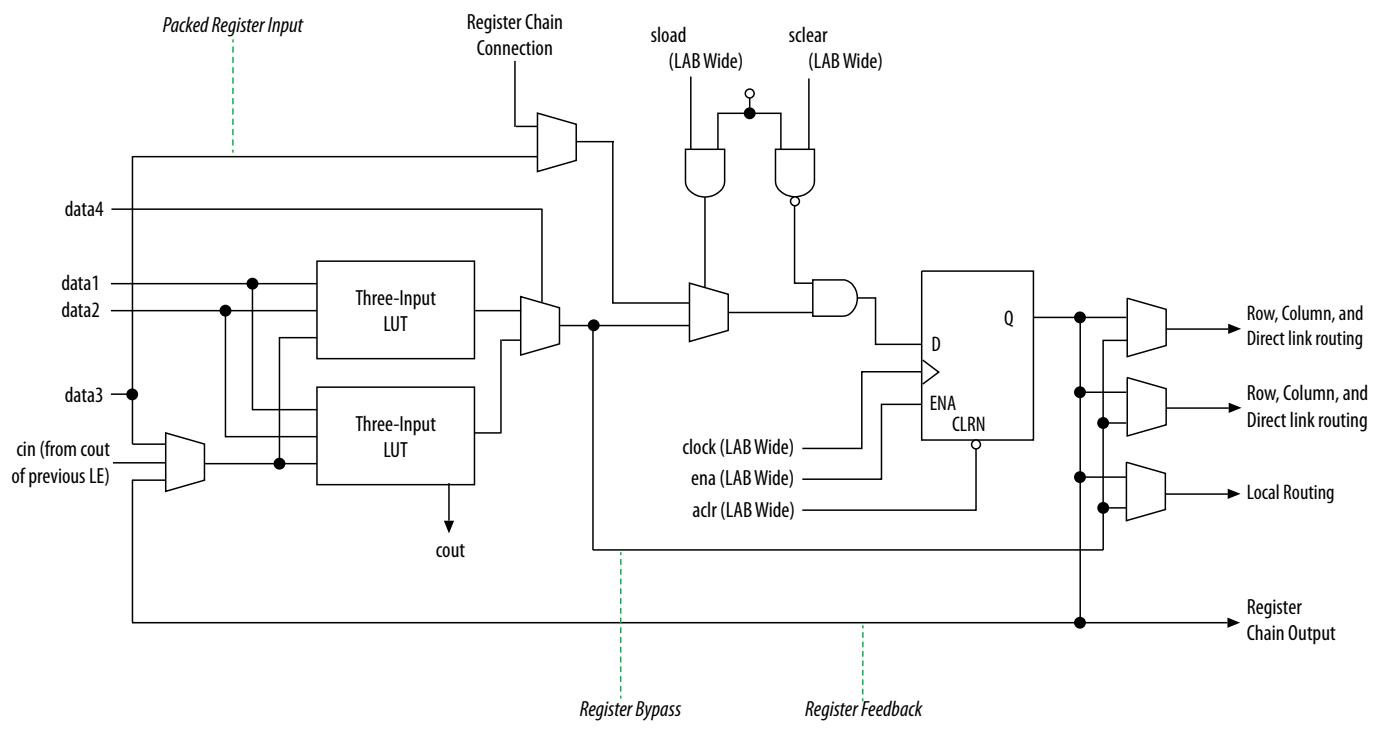
Arithmetic Mode

Arithmetic mode is ideal for implementing adders, counters, accumulators, and comparators.

The LE in arithmetic mode implements a 2-bit full adder and basic carry chain. LEs in arithmetic mode can drive out registered and unregistered versions of the LUT output. Register feedback and register packing are supported when LEs are used in arithmetic mode.

Figure 7: LE Operating in Arithmetic Mode for MAX 10 devices

Carry Chain

The Quartus Prime Compiler automatically creates carry chain logic during design processing. You can also manually create the carry chain logic during design entry. Parameterized functions, such as LPM functions, automatically take advantage of carry chains for the appropriate functions. The Quartus Prime Compiler creates carry chains longer than 16 LEs by automatically linking LABs in the same column.

To enhanced fitting, a long carry chain runs vertically, which allows fast horizontal connections to M9K memory blocks or embedded multipliers through direct link interconnects. For example, if a design has a long carry chain in a LAB column next to a column of M9K memory blocks, any LE output can feed an adjacent M9K memory block through the direct link interconnect.

If the carry chains run horizontally, any LAB which is not next to the column of M9K memory blocks uses other row or column interconnects to drive a M9K memory block.

A carry chain continues as far as a full column.

Embedded Memory

The MAX 10 embedded memory block is optimized for applications such as high throughput packet processing, embedded processor program, and embedded data storage.

The MAX 10 embedded memory structure consists of 9,216-bit (including parity bits) blocks. You can use each M9K block in different widths and configuration to provide various memory functions such as RAM, ROM, shift registers, and FIFO.

MAX 10 embedded memory supports the following general features:

- 8,192 memory bits per block (9,216 bits per block including parity).
- Independent read-enable (`rden`) and write-enable (`wren`) signals for each port.
- Packed mode in which the M9K memory block is split into two 4.5 K single-port RAMs.
- Variable port configurations.
- Single-port and simple dual-port modes support for all port widths.
- True dual-port (one read and one write, two reads, or two writes) operation.
- Byte enables for data input masking during writes.
- Two clock-enable control signals for each port (port A and port B).
- Initialization file to preload memory content in RAM and ROM modes.

Related Information
MAX 10 Embedded Memory User Guide

Embedded Multiplier

MAX 10 embedded memory supports the following general features:

- 8,192 memory bits per block (9,216 bits per block including parity).
- Independent read-enable (`rden`) and write-enable (`wren`) signals for each port.
- Packed mode in which the M9K memory block is split into two 4.5 K single-port RAMs.
- Variable port configurations.
- Single-port and simple dual-port modes support for all port widths.
- True dual-port (one read and one write, two reads, or two writes) operation.
- Byte enables for data input masking during writes.
- Two clock-enable control signals for each port (port A and port B).
- Initialization file to preload memory content in RAM and ROM modes.

You can use an embedded multiplier block in one of two operational modes, depending on the application needs:

- One 18-bit x 18-bit multiplier
- Up to two 9-bit x 9-bit independent multipliers

You can also use embedded multipliers of the MAX 10 devices to implement multiplier adder and multiplier accumulator functions. The multiplier portion of the function is implemented using embedded multipliers. The adder or accumulator function is implemented in logic elements (LEs).
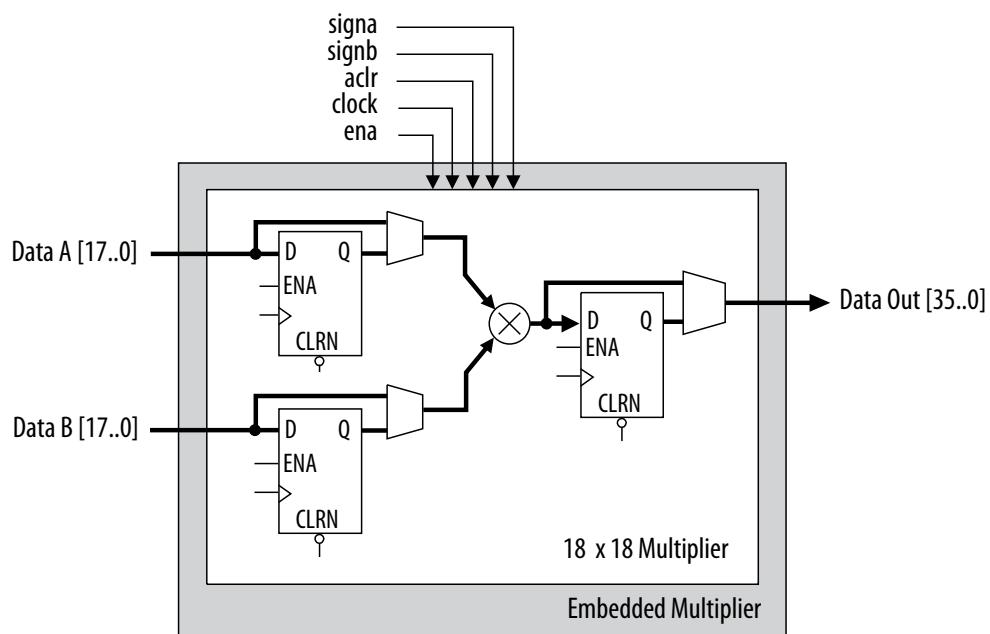
Related Information
MAX 10 Embedded Multiplier User Guide

18-Bit Multipliers

You can configure each embedded multiplier to support a single 18 x 18 multiplier for input widths of 10 to 18 bits.

The following figure shows the embedded multiplier configured to support an 18-bit multiplier.

Figure 8: 18-Bit Multiplier Mode
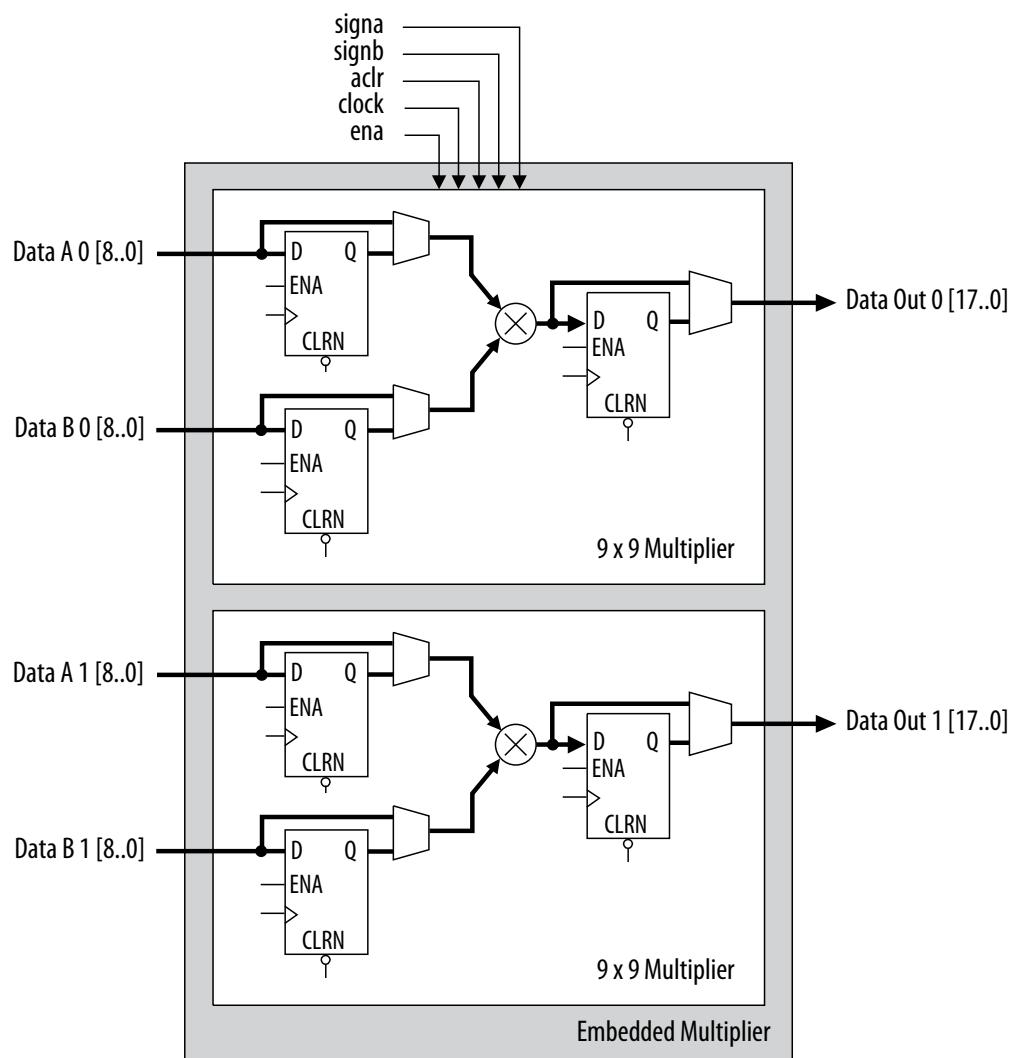
All 18-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both. Also, you can dynamically change the `signa` and `signb` signals and send these signals through dedicated input registers.

9-Bit Multipliers

You can configure each embedded multiplier to support two 9 × 9 independent multipliers for input widths of up to 9 bits.

The following figure shows the embedded multiplier configured to support two 9-bit multipliers.

Figure 9: 9-Bit Multiplier Mode


All 9-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both.

Each embedded multiplier block has only one `signa` and one `signb` signal to control the sign representation of the input data to the block. If the embedded multiplier block has two 9 × 9 multipliers the following applies:

- The `Data A` input of both multipliers share the same `signa` signal
- The `Data B` input of both multipliers share the same `signb` signal

Clocking and PLL

MAX 10 devices support global clock network (GCLK) and phase-locked loop (PLL).

Clock networks provide clock sources for the core. You can use clock networks in high fan out global signal network such as reset and clear.

PLLs provide robust clock management and synthesis for device clock management, external system clock management, and I/O interface clocking.

Related Information
MAX 10 Clock Networks and PLLs User Guide

Global Clock Networks

GCLKs drive throughout the entire device, feeding all device quadrants. All resources in the device, such as the I/O elements, logic array blocks (LABs), dedicated multiplier blocks, and M9K memory blocks can use GCLKs as clock sources. Use these clock network resources for control signals, such as clock enables and clears fed by an external pin. Internal logic can also drive GCLKs for internally-generated GCLKs and asynchronous clears, clock enables, or other control signals with high fan-out.

**Figure 10: GCLK Network Sources for 10M02, 10M04, and 10M08 Devices**
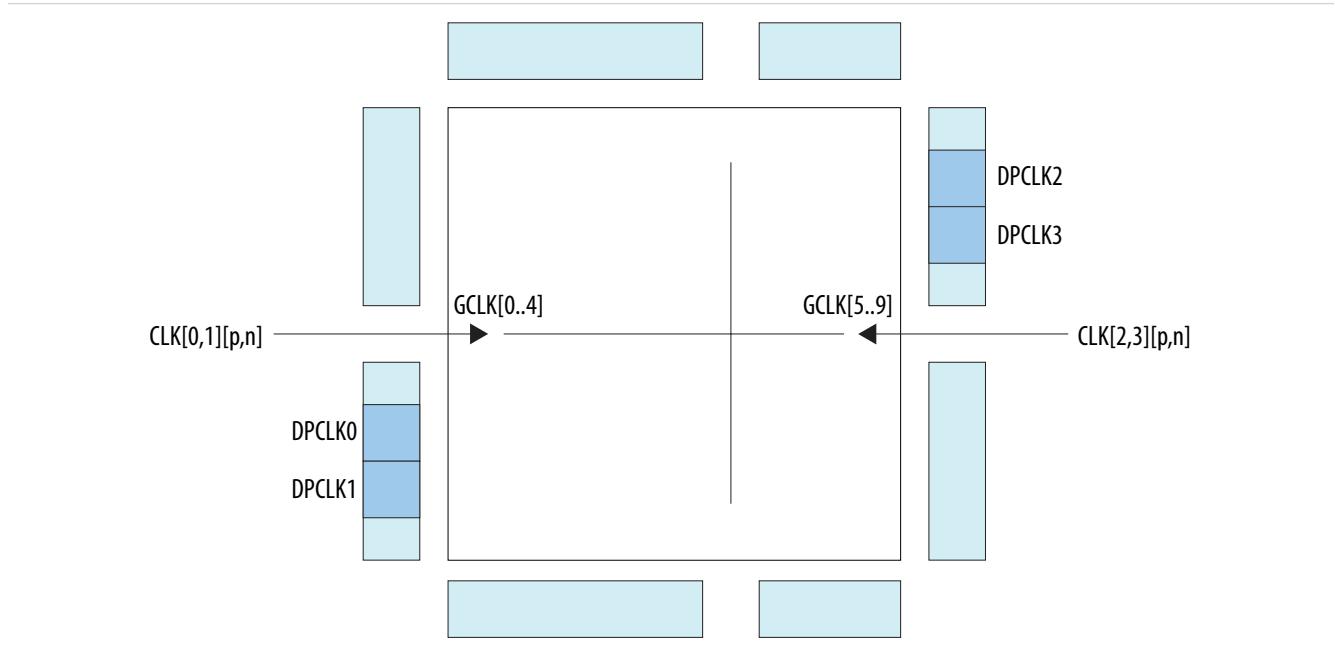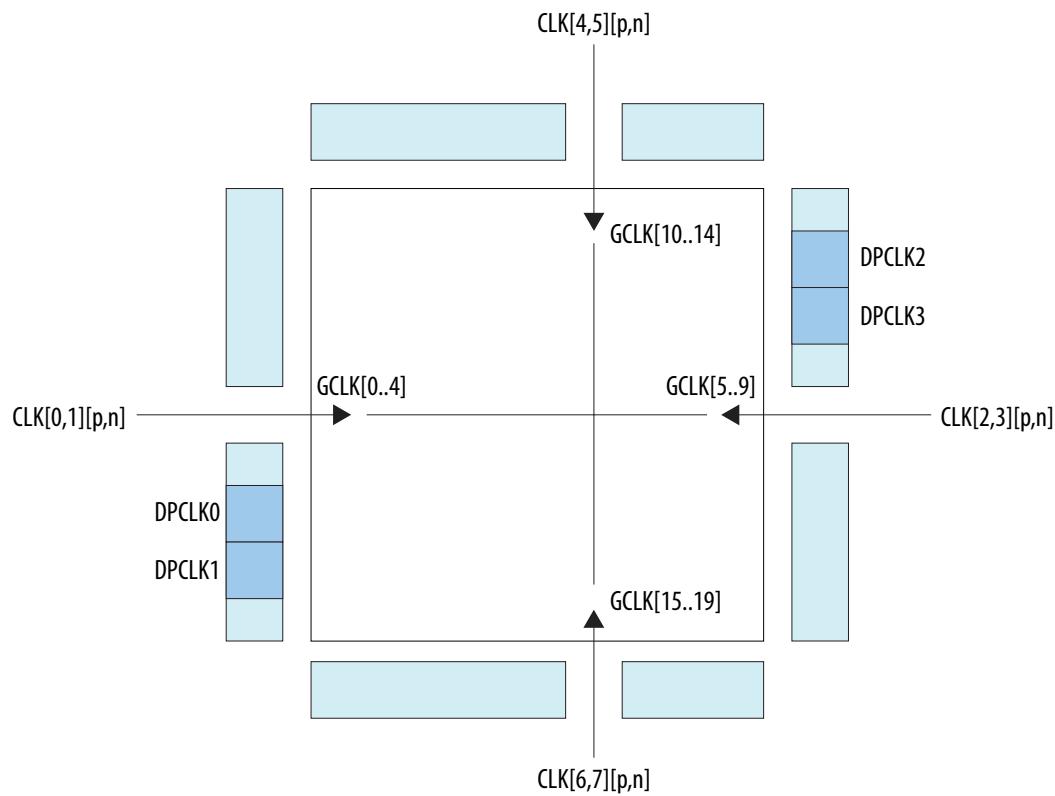
Figure 11: GCLK Network Sources for 10M16, 10M25, 10M40, and 10M50 Devices

Internal Oscillator

MAX 10 devices have built-in internal ring oscillator with clock multiplexers and dividers. The internal ring oscillator operates up to 232 MHz which is not accessible. This operating frequency further divides down to slower frequencies.
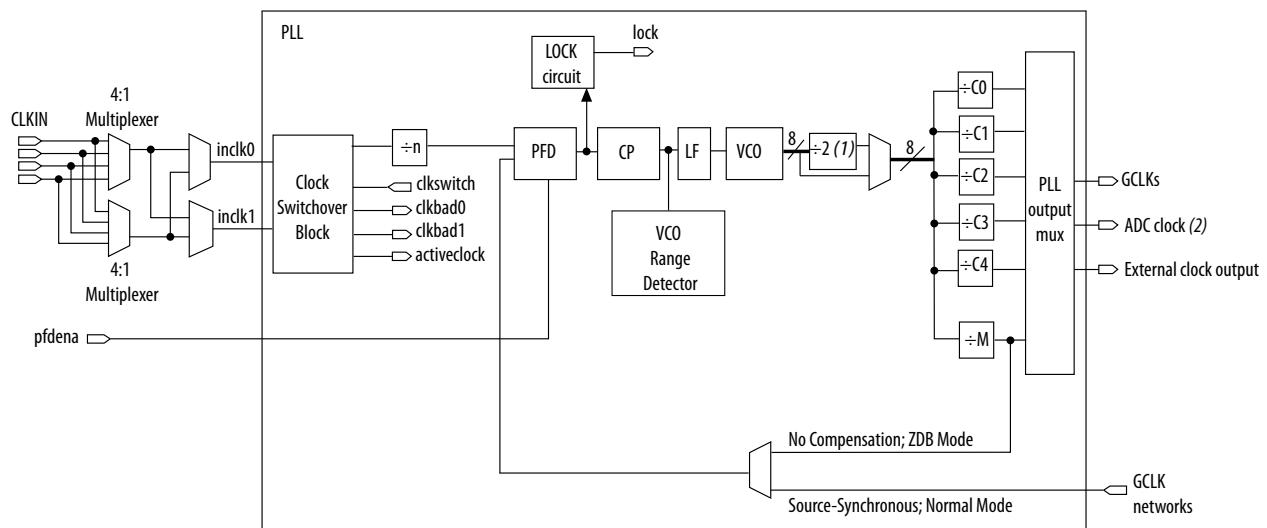
When the `oscena` input signal is asserted, the oscillator is enabled and the output can be routed to the logic array through the `clkout` output signal. When the `oscena` signal is set low, the `clkout` signal is constant high. You can analyze this delay using the TimeQuest timing analyzer.

PLL Block and Locations

The main purpose of a PLL is to synchronize the phase and frequency of the voltage-controlled oscillator (VCO) to an input reference clock.

Figure 12: MAX 10 PLL High-Level Block Diagram
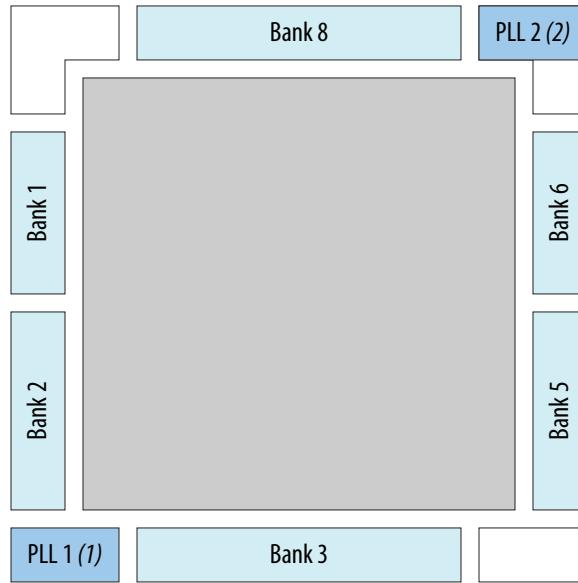
Each clock source can come from any of the two or four clock pins located on the same side of the device as the PLL.

Notes:
(1) This is the VCO post-scale counter K.
(2) Only counter C0 of PLL1 and PLL3 can drive the ADC clock.

The following figures show the physical locations of the PLLs. Every index represents one PLL in the device. The physical locations of the PLLs correspond to the coordinates in the Quartus Prime Chip Planner.
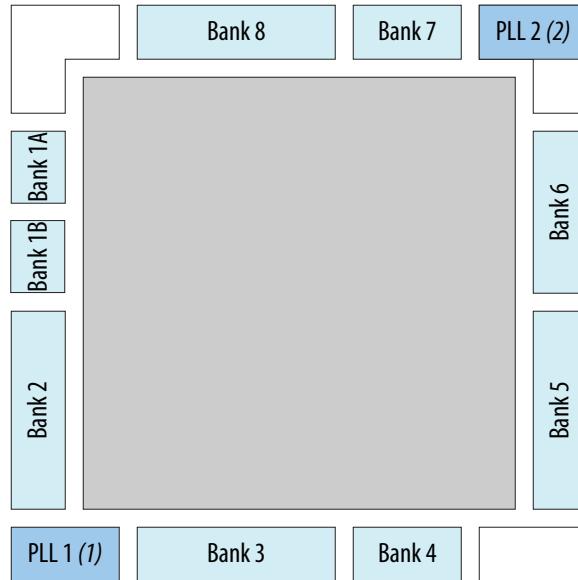
Figure 13: PLL Locations for 10M02 Device

Notes:
(1) Available on all packages except V36 package.
(2) Available on U324 and V36packages only.
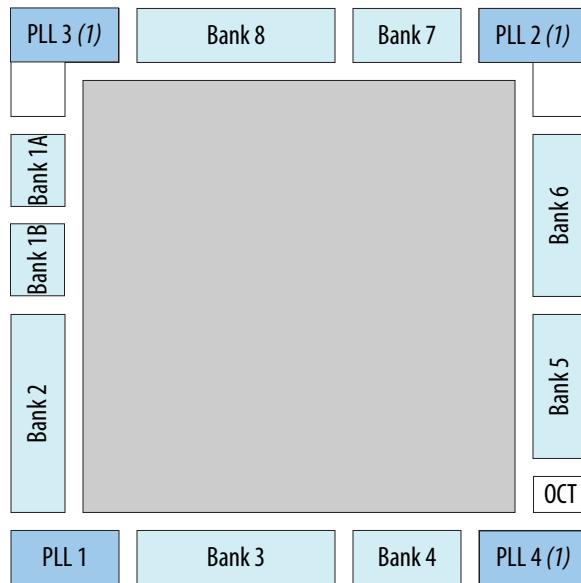
Figure 14: PLL Locations for 10M04 and 10M08 Devices

Notes:
(1) Available on all packages except V81 package.
(2) Available on F256, F484, U324, and V81 packages only.

Figure 15: PLL Locations for 10M16, 10M25, 10M40 and 10M50 Devices

Note:
(1) Available on all packages except E144 and U169 packages.

General Purpose I/O

The I/O system of MAX 10 devices support various I/O standards. In the MAX 10 devices, the I/O pins are located in I/O banks at the periphery of the devices. The I/O pins and I/O buffers have several programmable features.

Related Information
MAX 10 General Purpose I/O User Guide

MAX 10 I/O Banks Architecture

The I/O elements are located in a group of four modules per I/O bank:

- High speed DDR3 I/O banks—supports various I/O standards and protocols including DDR3. These I/O banks are available only on the right side of the device.
- High speed I/O banks—supports various I/O standards and protocols except DDR3. These I/O banks are available on the top, left, and bottom sides of the device.
- Low speed I/O banks—lower speeds I/O banks that are located at the top left side of the device.

For more information about I/O pins support, refer to the pinout files for your device.

MAX 10 I/O Banks Locations

The I/O banks are located at the periphery of the device.

For more details about the modular I/O banks available in each device package, refer to the relevant device pin-out file.

Figure 16: I/O Banks for MAX 10 02 Devices—Preliminary
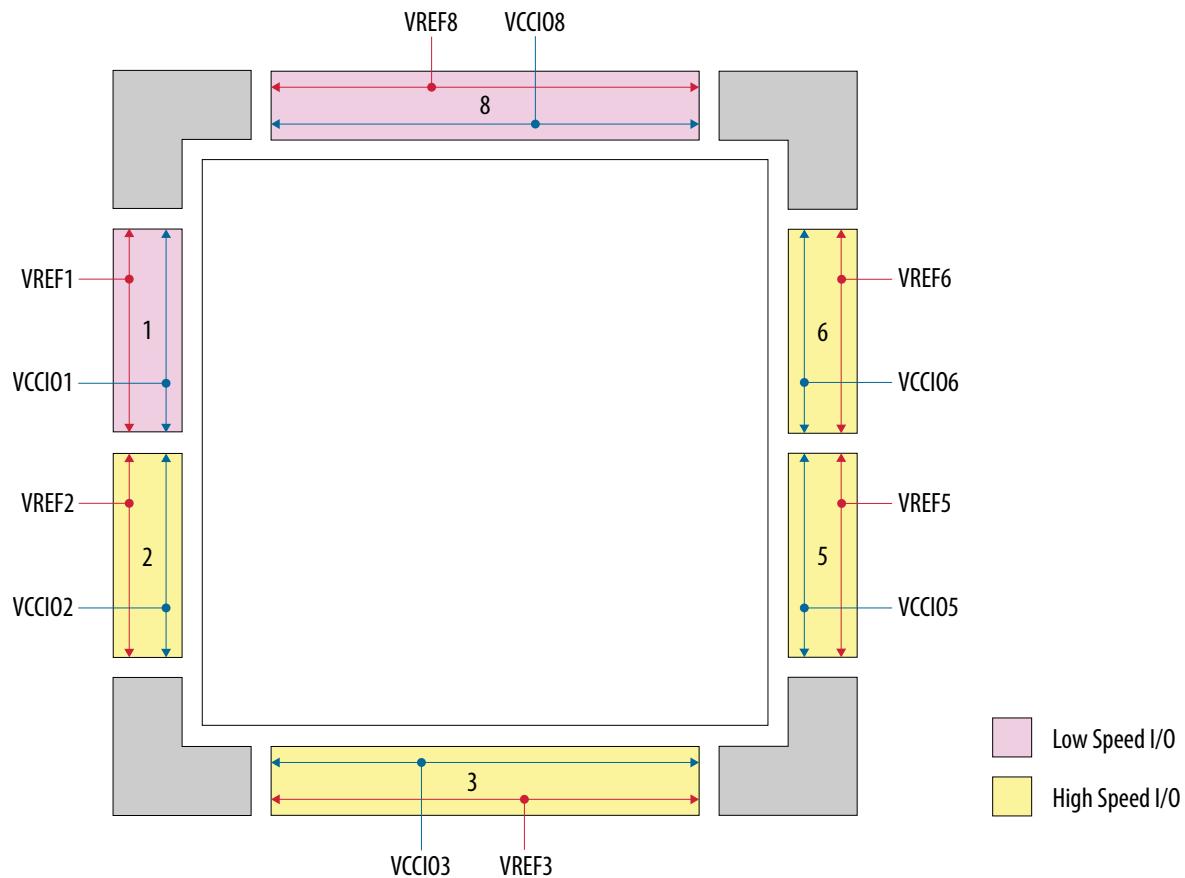
**Figure 17: I/O Banks for MAX 10 04 and 08 Devices—Preliminary**
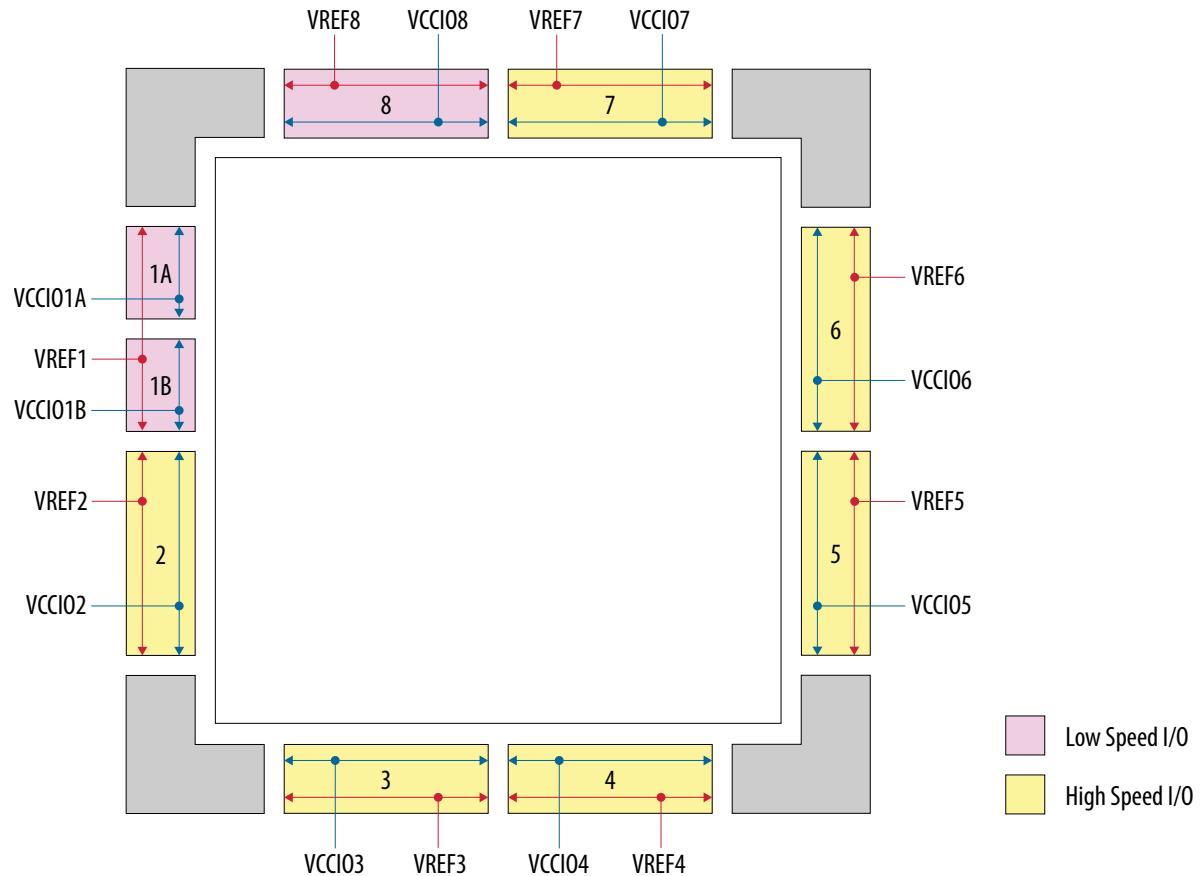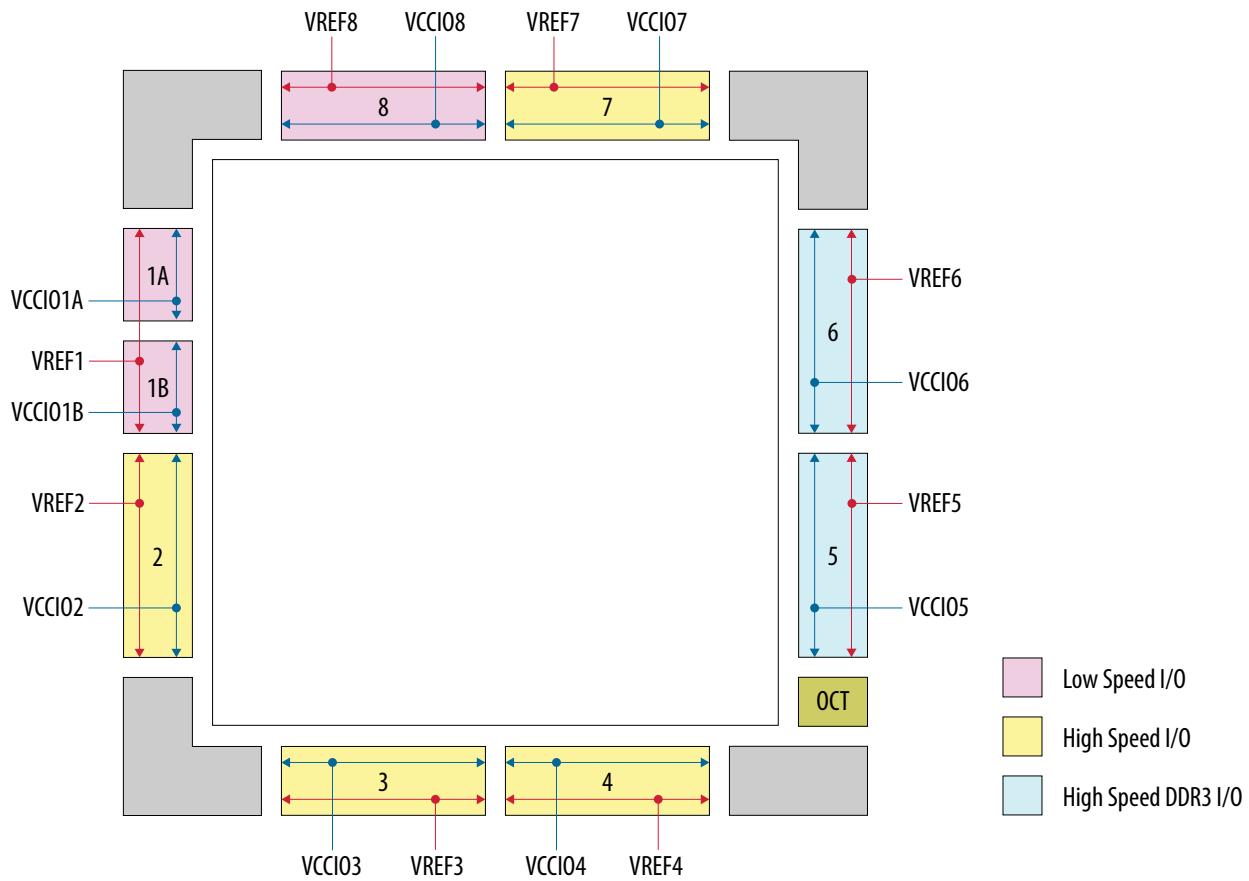
Figure 18: I/O Banks for MAX 10 16, 25, 40, and 50 Devices—Preliminary

High-Speed LVDS I/O

The MAX 10 device family supports high-speed LVDS protocols through the LVDS I/O banks and the Altera Soft LVDS IP core.

The MAX 10 devices use registers and logic in the core fabric to implement LVDS input and output interfaces.

- For LVDS transmitters and receivers, MAX 10 devices use the the double data rate I/O (DDIO) registers that reside in the I/O elements (IOE). This architecture improves performance with regards to the receiver input skew margin (RSKM) or transmitter channel-to-channel skew (TCCS).
- For the LVDS serializer/deserializer (SERDES), MAX 10 devices use logic elements (LE) registers.

Related Information
MAX 10 High-Speed LVDS I/O User Guide

MAX 10 High-Speed LVDS Circuitry

The LVDS solution uses the I/O elements and registers in the MAX 10 devices. The Altera Soft LVDS IP core implements the serializer and deserializer as soft SERDES blocks in the core logic.

The MAX 10 devices do not contain dedicated serialization or deserialization circuitry:

- You can use I/O pins and core fabric to implement a high-speed differential interface in the device.
- The MAX 10 solution uses shift registers, internal PLLs, and I/O elements to perform the serial-to-parallel and parallel-to-serial conversions of incoming and outgoing data.
- The Quartus Prime software uses the parameter settings of the Altera Soft LVDS IP core to automatically construct the differential SERDES in the core fabric.

Figure 19: Soft LVDS SERDES

This figure shows a transmitter and receiver block diagram for the soft LVDS SERDES circuitry with the interface signals of the transmitter and receiver data paths.

MAX 10 High-Speed LVDS I/O Location

The I/O banks in MAX 10 devices support true LVDS input and emulated LVDS output on all I/O banks. Only the bottom I/O banks support true LVDS output.

Figure 20: LVDS Support in I/O Banks of 10M02 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2 and 6.

	TX		RX	
LVDS				
Emulated LVDS				
RSDS				
Emulated RSDS				
Mini-LVDS				
Emulated Mini-LVDS				
PPDS				
Emulated PPDS				
BLVDS				
LVPECL				
TMDS				
Sub-LVDS				
SLVS				
HiSpi				
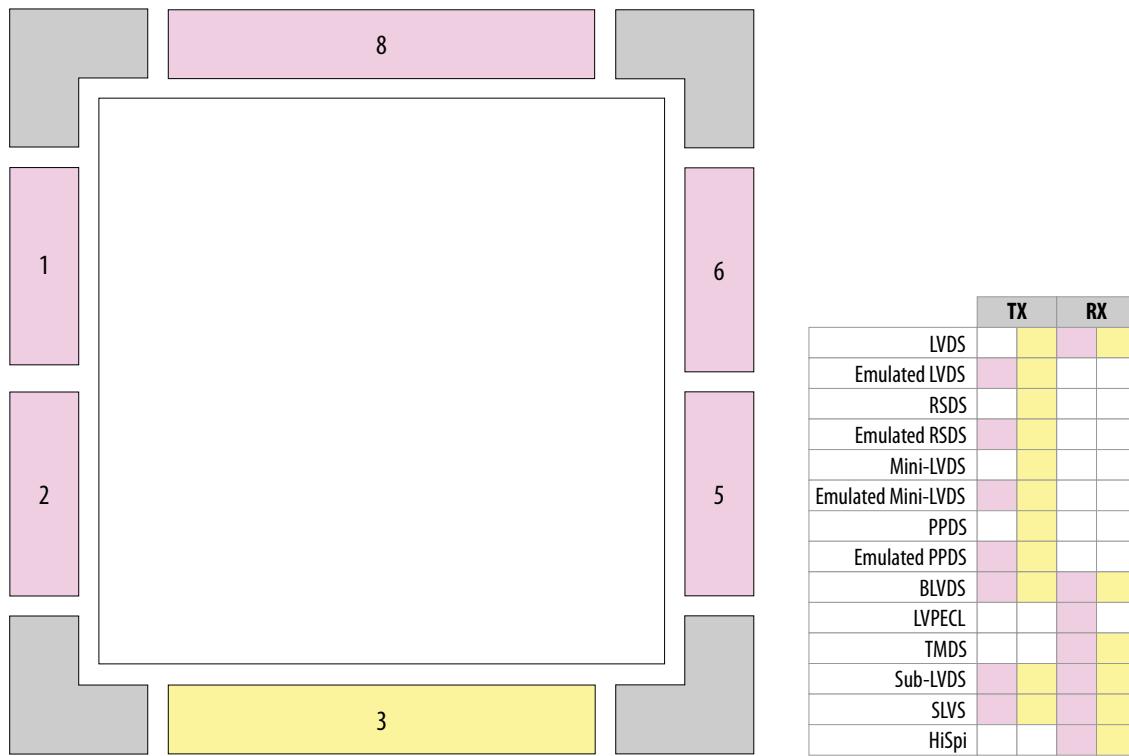
Figure 21: LVDS Support in I/O Banks of 10M04 and 10M08 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2 and 6.

	TX	RX
LVDS		
Emulated LVDS		
RSDS		
Emulated RSDS		
Mini-LVDS		
Emulated Mini-LVDS		
PPDS		
Emulated PPDS		
BLVDS		
LVPECL		
TMDS		
Sub-LVDS		
SLVS		
HiSpi		

Figure 22: LVDS Support in I/O Banks of 10M16, 10M25, 10M40, and 10M50 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2, 3, 6, and 8.

	TX		RX	
LVDS		■	■	■
Emulated LVDS	■	■		
RSDS		■		
Emulated RSDS	■	■		
Mini-LVDS		■		
Emulated Mini-LVDS	■	■		
PPDS		■		
Emulated PPDS	■	■		
BLVDS	■	■	■	■
LVPECL			■	■
TMDS			■	■
Sub-LVDS	■	■	■	■
SLVS	■	■	■	■
HiSpi			■	■
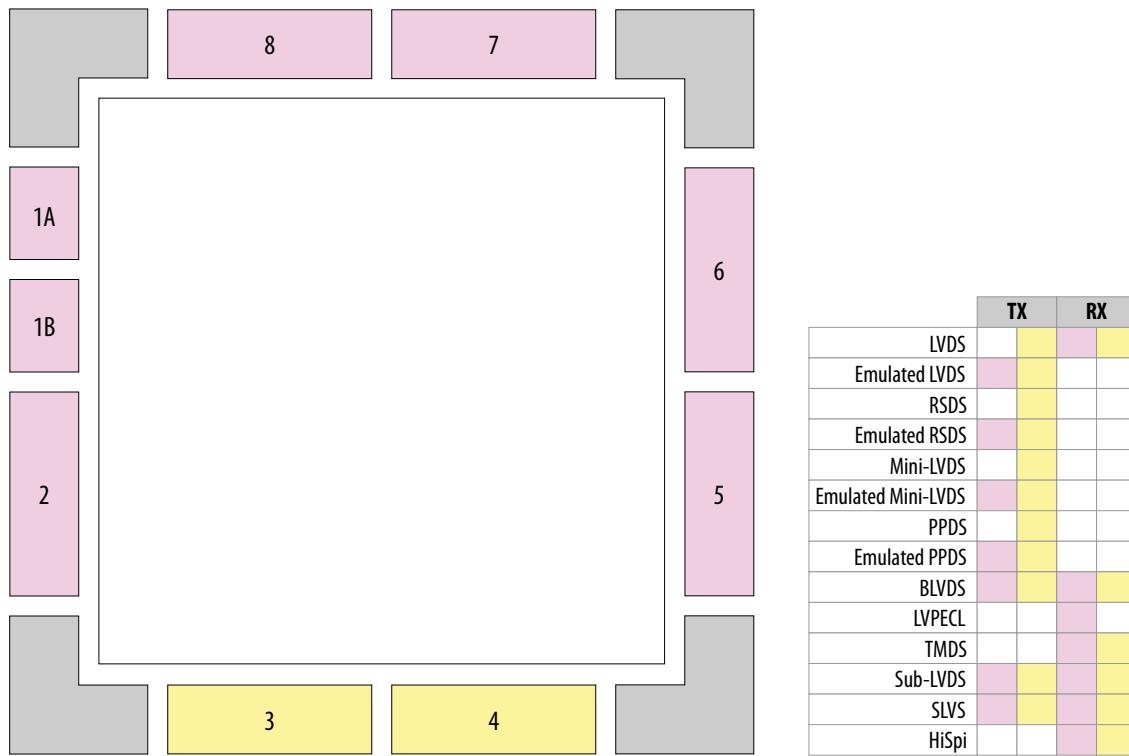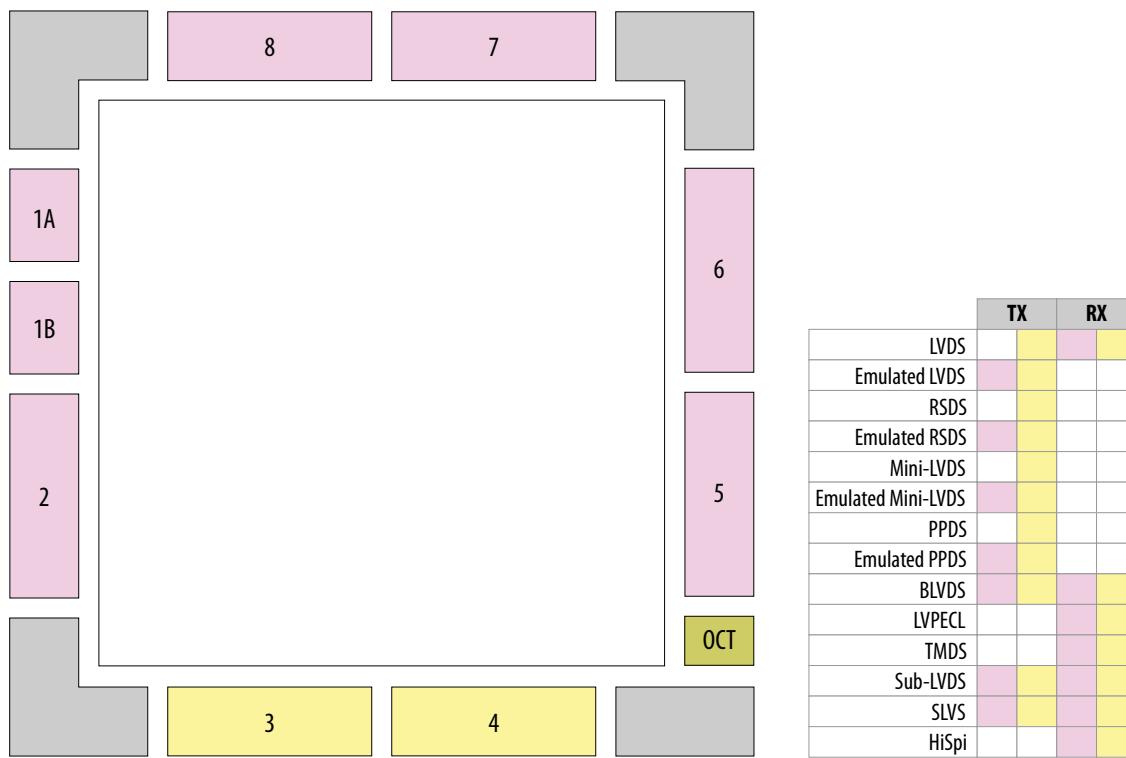
External Memory Interface

The MAX 10 devices are capable of interfacing with a broad range of external memory standards.

This capability allows you to use the MAX 10 devices in a wide range of applications such as image processing, storage, communications, and general embedded systems.

The external memory interface solution in MAX 10 devices consist of:

- The I/O elements that support external memory interfaces.
- The UniPHY IP core that allows you to configure the memory interfaces to support different external memory interface standards.
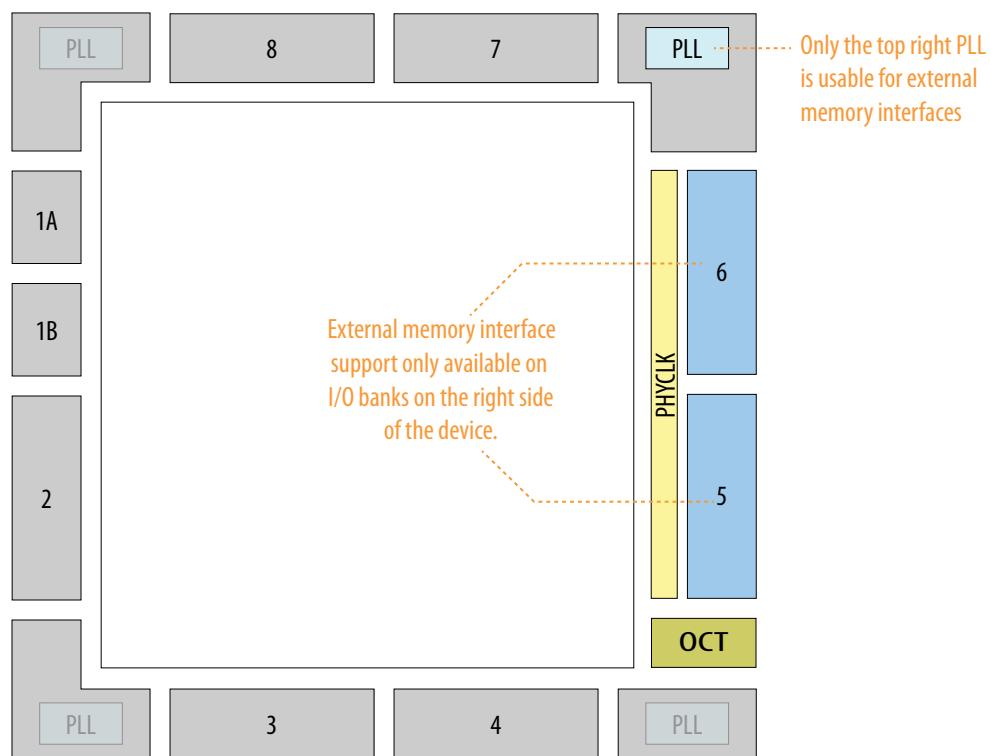
Related Information
[MAX 10 External Memory Interface User Guide](#)

MAX 10 I/O Banks for External Memory Interface

In MAX 10 devices, external memory interfaces are supported only on the I/O banks on the right side of the device. You must place all external memory I/O pins on the I/O banks on the right side of the device.

Figure 23: I/O Banks for External Memory Interfaces

This figure represents the top view of the silicon die that corresponds to a reverse view of the device package.

External memory interfaces support is available only for 10M16, 10M25, 10M40, and 10M50 devices.

Analog-to-Digital Converter

MAX 10 devices feature up to two analog-to-digital converters (ADC). The ADCs provide the MAX 10 devices with built-in capability for on-die temperature monitoring and external analog signal conversion.

The ADC solution consists of hard IP blocks in the MAX 10 device periphery and soft logic through the Altera Modular ADC IP core.

The ADC solution provides you with built-in capability to translate analog quantities to digital data for information processing, computing, data transmission, and control systems. The basic function is to provide a 12 bit digital representation of the analog signal being observed.

The ADC solution works in two modes:

- Normal mode—monitors up to 18 single-ended external inputs with a cumulative sampling rate of one megasymbols per second (Msps).
- Temperature sensing mode—monitors internal temperature data input with a sampling rate of up to 50 kilosymbols per second (ksps).

Figure 24: ADC Hard IP Block in MAX 10 Devices

ADC Block Locations

The ADC blocks are located at the top left corner of the MAX 10 device periphery.

**Figure 25: ADC Block Location in MAX 10 04 and 08 Devices**

Send Feedback
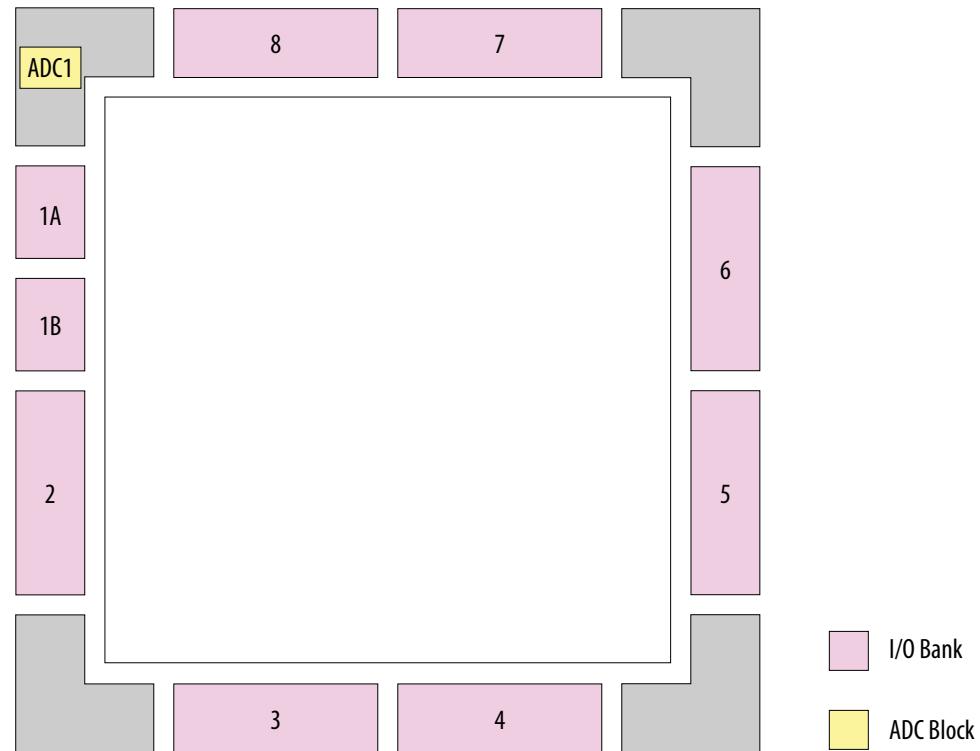
**Figure 26: ADC Block Location in MAX 10 16 Devices**
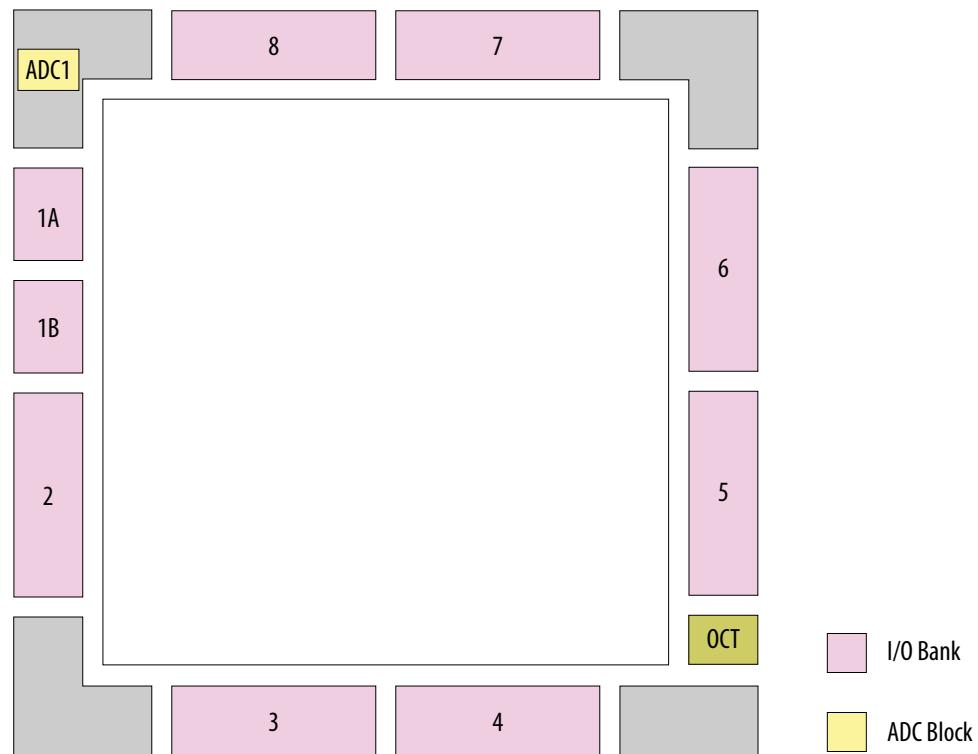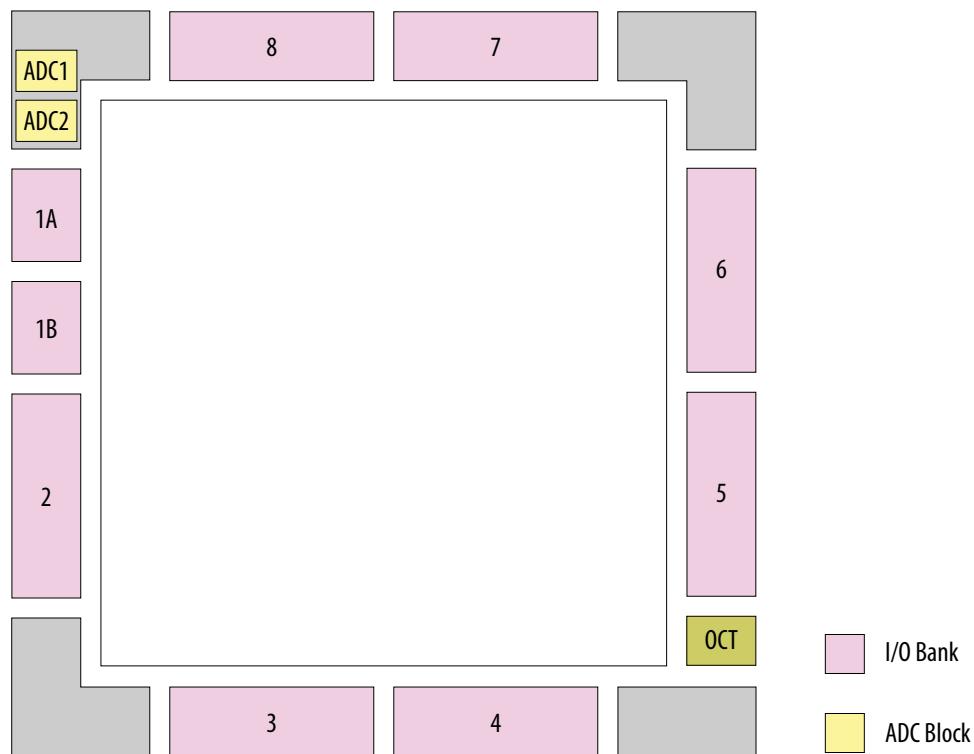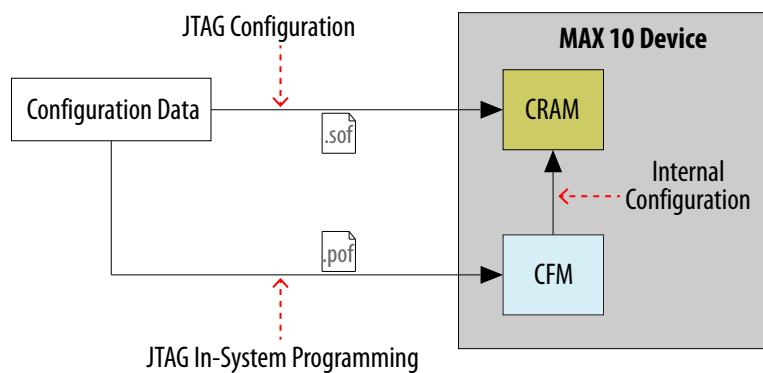
Figure 27: ADC Block Location in MAX 10 25, 40, and 50 Devices

Package E144 of these devices have only one ADC block.

Configuration Schemes

Figure 28: High-Level Overview of JTAG Configuration and Internal Configuration for MAX 10 Devices

JTAG Configuration

In MAX 10 devices, JTAG instructions take precedence over the internal configuration scheme.

Using the JTAG configuration scheme, you can directly configure the device CRAM through the JTAG interface—TDI, TDO, TMS, and TCK pins. The Quartus Prime software automatically generates an SRAM

Object File (**.sof**). You can program the **.sof** using a download cable with the Quartus Prime software programmer.

Internal Configuration

You need to program the configuration data into the configuration flash memory (CFM) before internal configuration can take place. The configuration data to be written to CFM will be part of the programmer object file (**.pof**). Using JTAG In-System Programming (ISP), you can program the **.pof** into the internal flash.

During internal configuration, MAX 10 devices load the CRAM with configuration data from the CFM.
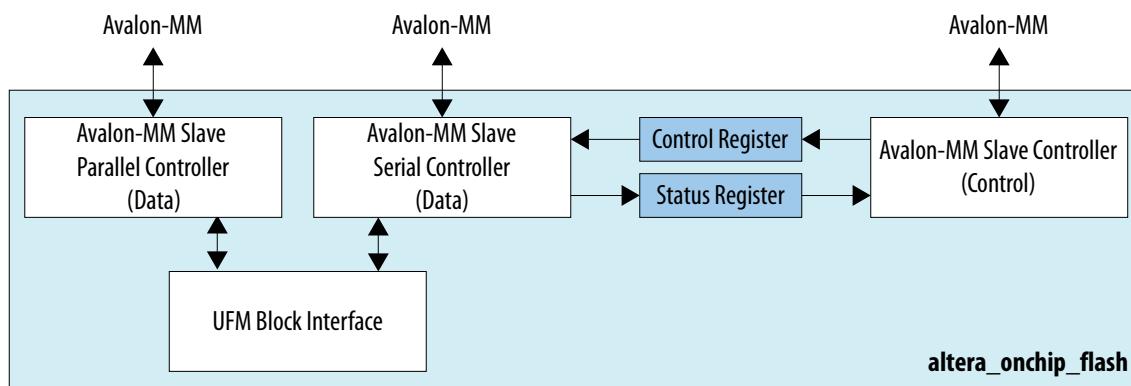
User Flash Memory

Altera MAX 10 devices feature a user flash memory (UFM) block that stores non-volatile information.

The UFM is part of the internal flash available in MAX 10 devices.

The UFM architecture of MAX 10 devices is a combination of soft and hard IPs. You can only access the UFM using the Altera On-Chip Flash IP core in the Quartus II software.

Figure 29: Altera On-Chip Flash IP Block Diagram

This IP block has two Avalon-MM slave controllers:

- Data—a wrapper of the UFM block that provides read and write accesses to the flash.
- Control—the CSR and status register for the flash, that is required only for write operations.

Related Information
MAX 10 User Flash Memory (UFM) User Guide

Power Management

MAX 10 power optimization features are as follows:

- Single-supply or dual-supply device options
- Power-on reset (POR) circuitry
- Power management controller scheme
- Hot socketing
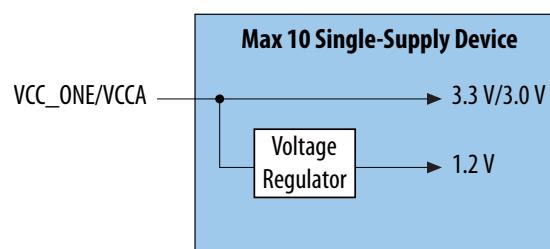
Related Information
Power Management User Guide

Single-Supply Device

MAX 10 single-supply devices only need either a 3.0- or 3.3-V external power supply. The external power supply serves as an input to the MAX 10 device VCC_ONE and VCCA power pins. This external power supply is then regulated by an internal voltage regulator in the MAX 10 single-supply device to 1.2 V. The 1.2-V voltage level is required by core logic operation.
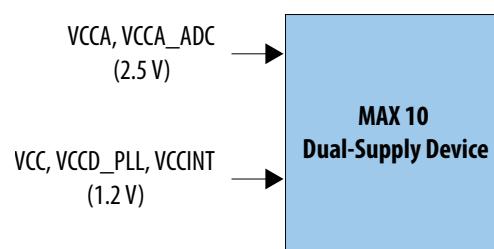
Figure 30: MAX 10 Single-Supply Device

Dual-Supply Device

MAX 10 dual-supply devices require 1.2 V and 2.5 V for the device core logics and periphery operations.

Figure 31: MAX 10 Dual-Supply Device

Power Management Controller Scheme

The power management controller scheme allows you to allocate some applications in sleep mode during runtime. This enables you to to turn off portions of the design, thus reducing dynamic power consumption. You can re-enable your application with a fast wake-up time of less than 1 ms.

Hot Socketing

The MAX 10 device offers hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove the MAX 10 device on a board in a system during system operation. This does not affect the running system bus or the board that is inserted into the system.

The hot-socketing feature removes some encountered difficulties when using the MAX 10 device on a PCB that contains a mixture of devices with different voltage levels.

With the MAX 10 device hot-socketing feature, you no longer need to ensure a proper power-up sequence for each device on the board. MAX 10 device hot-socketing feature provides:

- Board or device insertion and removal without external components or board manipulation
- Support for any power-up sequence
- Non-intrusive I/O buffers to system buses during hot insertion

Document Revision History for MAX 10 FPGA Device Architecture

Date	Version	Changes
May 2016	2016.05.13	• Added internal oscillator architectural information. • Updated section name from *Clock Networks and PLL* to *Clocking and PLL*. • Added high-speed LVDS circuity information. • Added power management controller scheme and hot socketing information.
May 2015	2015.05.04	• Removed 'Internal Configuration' figure. • Added 'Overview of of JTAG Configuration and Internal Configuration for MAX 10 Devices' figure in 'Configuration'.
December 2014	2014.12.15	• Updated Altera On Chip Flash IP core block diagram for user flash memory. • Updated links.
September 2014	2014.09.22	Initial release.

MAX 10 Embedded Memory User Guide

Contents

2015.11.02

UG-M10MEMORY ✉ **Subscribe** 💬 **Send Feedback**

MAX® 10 embedded memory block is optimized for applications such as high throughput packet processing, embedded processor program, and embedded data storage.

ISO 9001:2008 Registered

✉ **Subscribe**   💬 **Send Feedback**

The MAX 10 embedded memory structure consists of 9,216-bit (including parity bits) blocks. You can use each M9K block in different widths and configuration to provide various memory functions such as RAM, ROM, shift registers, and FIFO.

The following list summarizes the MAX 10 embedded memory features:

- Embedded memory general features
- Embedded memory operation modes
- Embedded memory clock modes

Related Information
MAX 10 Device Overview
For information about MAX 10 devices embedded memory capacity and distribution

MAX 10 Embedded Memory General Features

MAX 10 embedded memory supports the following general features:

- 8,192 memory bits per block (9,216 bits per block including parity).
- Independent read-enable (`rden`) and write-enable (`wren`) signals for each port.
- Packed mode in which the M9K memory block is split into two 4.5 K single-port RAMs.
- Variable port configurations.
- Single-port and simple dual-port modes support for all port widths.
- True dual-port (one read and one write, two reads, or two writes) operation.
- Byte enables for data input masking during writes.
- Two clock-enable control signals for each port (port A and port B).
- Initialization file to preload memory content in RAM and ROM modes.

Control Signals
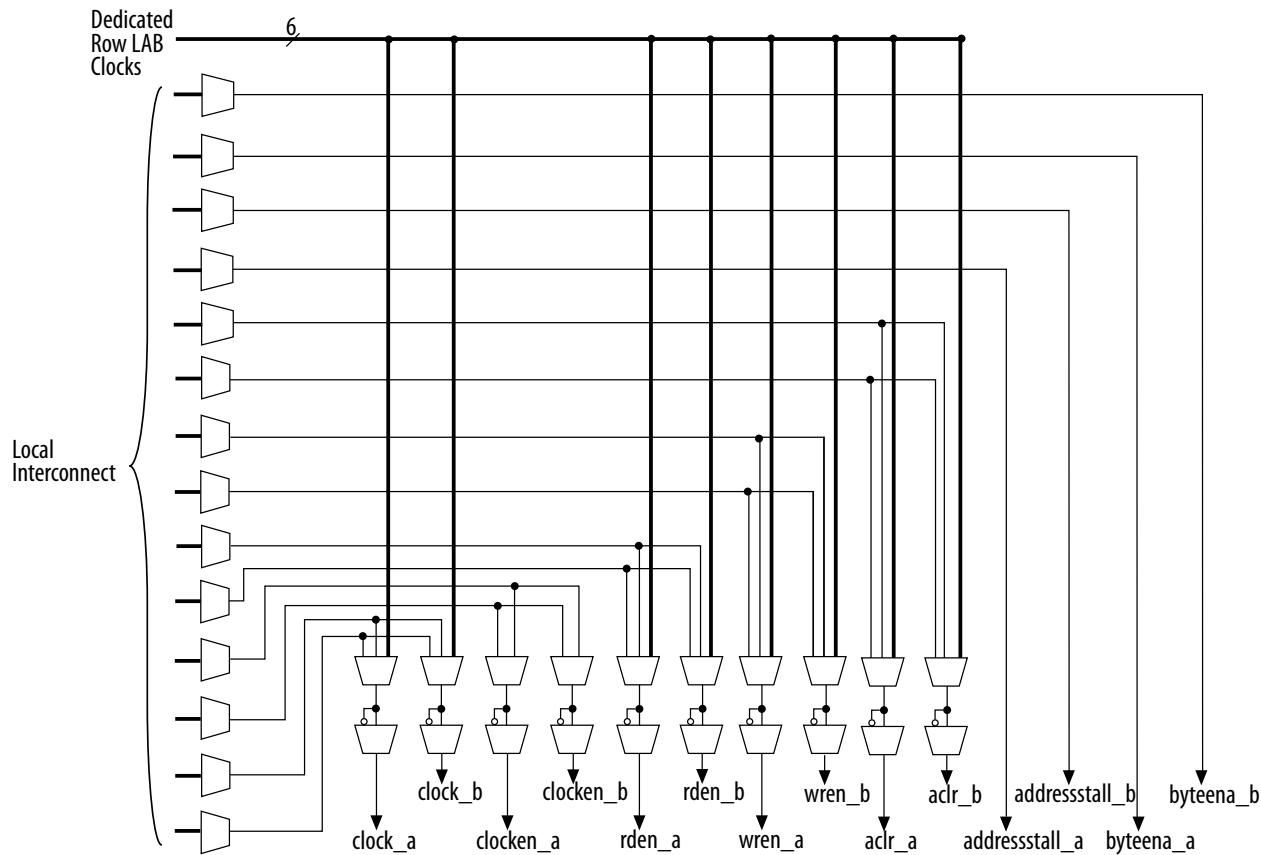
The clock-enable control signal controls the clock entering the input and output registers and the entire M9K memory block. This signal disables the clock so that the M9K memory block does not see any clock edges and does not perform any operations.

ISO
9001:2008
Registered

The `rden` and `wren` control signals control the read and write operations for each port of the M9K memory blocks. You can disable the `rden` or `wren` signals independently to save power whenever the operation is not required.

Figure 2-1: Register Clock, Clear, and Control Signals Implementation in M9K Embedded Memory Block

Parity Bit

You can perform parity checking for error detection with the parity bit along with internal logic resources. The M9K memory blocks support a parity bit for each storage byte. You can use this bit as either a parity bit or as an additional data bit. No parity function is actually performed on this bit. If error detection is not desired, you can use the parity bit as an additional data bit.

Read Enable

M9K memory blocks support the read enable feature for all memory modes.

If you...	...Then
Create the read-enable port and perform a write operation with the read enable port deasserted	The data output port retains the previous values that are held during the most recent active read enable.

If you...	...Then
- Activate the read enable during a write operation, or - Do not create a read-enable signal	The output port shows: - the new data being written, - the old data at that address, or - a "Don't Care" value when read-during-write occurs at the same address location.

Read-During-Write

The read-during-write operation occurs when a read operation and a write operation target the same memory location at the same time.

The read-during-write operation operates in the following ways:

- Same-port
- Mixed-port

Related Information

Customize Read-During-Write Behavior on page 3-1

Byte Enable

- Memory block that are implemented as RAMs support byte enables.
- The byte enable controls mask the input data, so that only specific bytes of data are written. The unwritten bytes retain the values written previously.
- The write enable (`wren`) signal, together with the byte enable (`byteena`) signal, control the write operations on the RAM blocks. By default, the `byteena` signal is high (enabled) and only the `wren` signal controls the writing.
- The byte enable registers do not have a `clear` port.
- M9K blocks support byte enables when the write port has a data width of ×16, ×18, ×32, or ×36 bits.
- Byte enables operate in a one-hot fashion. The LSB of the `byteena` signal corresponds to the LSB of the data bus. For example, if `byteena = 01` and you are using a RAM block in ×18 mode, `data[8:0]` is enabled and `data[17:9]` is disabled. Similarly, if `byteena = 11`, both `data[8:0]` and `data[17:9]` are enabled.
- Byte enables are active high.

Byte Enable Controls

Table 2-1: M9K Blocks Byte Enable Selections

byteena[3:0]	Affected Bytes. Any Combination of Byte Enables is Possible.			
	datain x 16	datain x 18	datain x 32	datain x 36
[0] = 1	[7:0]	[8:0]	[7:0]	[8:0]
[1] = 1	[15:8]	[17:9]	[15:8]	[17:9]
[2] = 1	—	—	[23:16]	[26:18]
[3] = 1	—	—	[31:24]	[35:27]
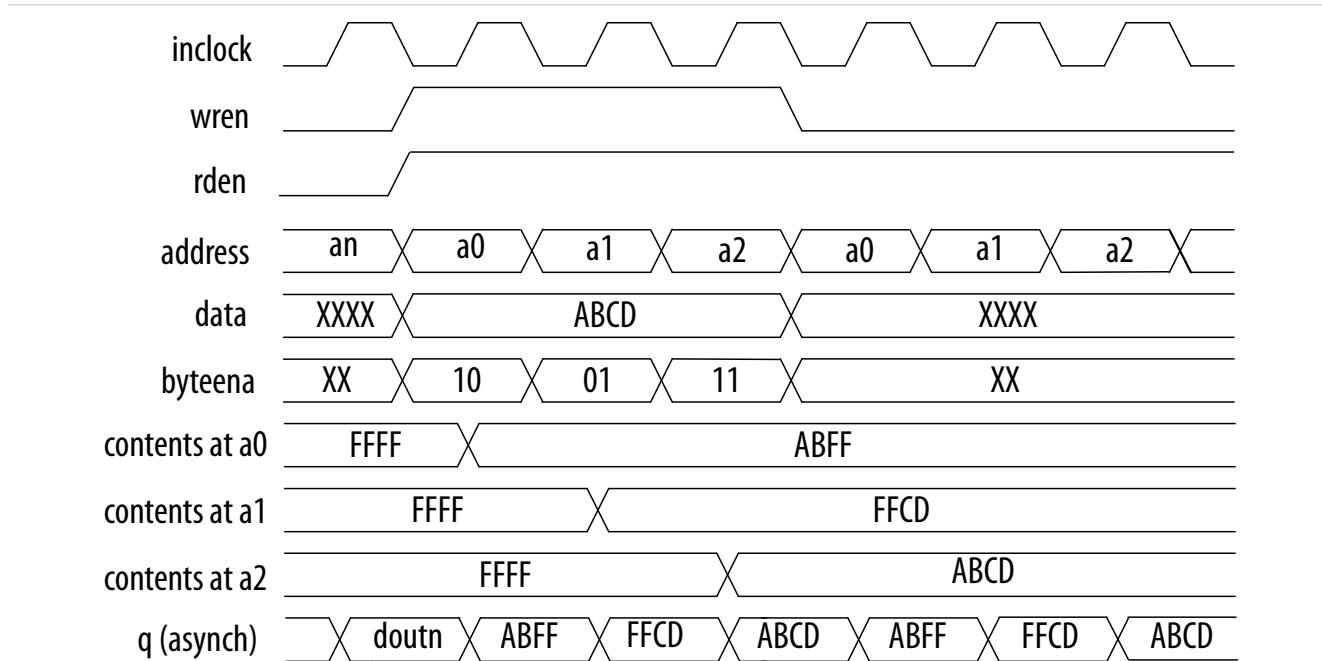
Data Byte Output

If you...	...Then
Deassert a byte-enable bit during a write cycle	The old data in the memory appears in the corresponding data-byte output.
Assert a byte-enable bit during a write cycle	The corresponding data-byte output depends on the Quartus® Prime software setting. The setting can be either the newly written data or the old data at that location.

RAM Blocks Operations

This figure shows how the `wren` and `byteena` signals control the RAM operations.

Figure 2-2: Byte Enable Functional Waveform

For this functional waveform, New Data Mode is selected.

Packed Mode Support

You can implement two single-port memory blocks in a single block under the following conditions:

- Each of the two independent block sizes is less than or equal to half of the M9K block size. The maximum data width for each independent block is 18 bits wide.
- Each of the single-port memory blocks is configured in single-clock mode.
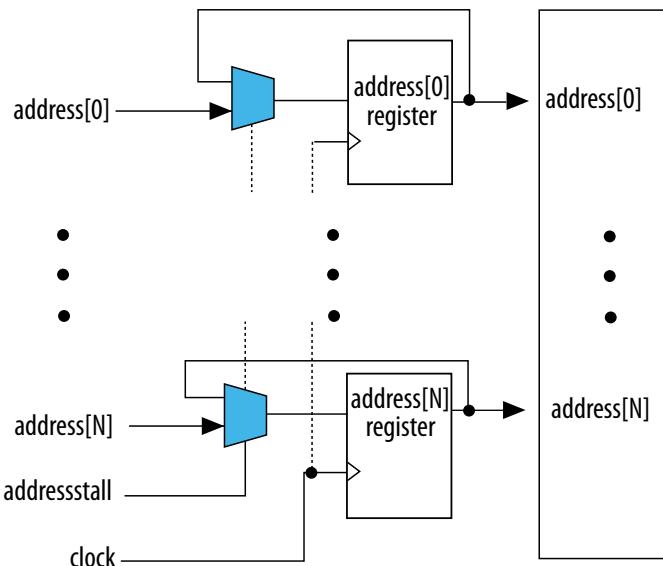
Related Information

Address Clock Enable Support
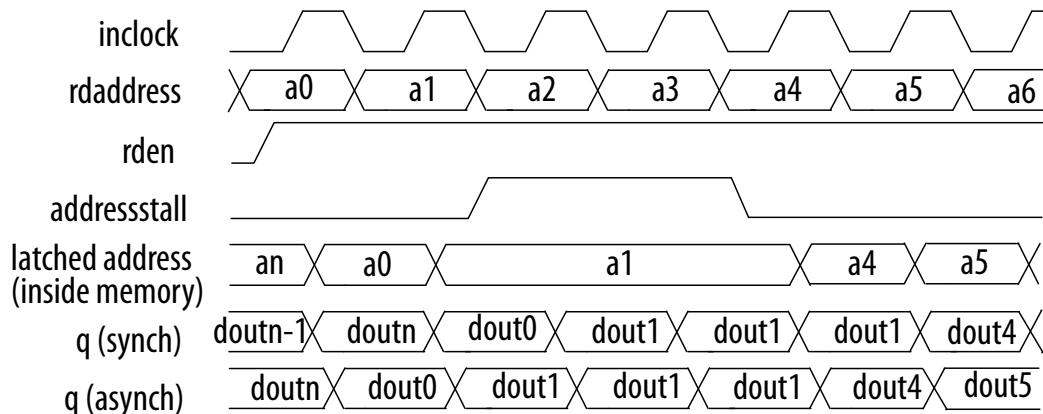
- The address clock enable feature holds the previous address value for as long as the address clock enable signal (`addressstall`) is enabled (`addressstall = 1`).
- When you configure M9K memory blocks in dual-port mode, each port has its own independent address clock enable.
- Use the address clock enable feature to improve the effectiveness of cache memory applications during a cache-miss.
- The default value for the `addressstall` signal is low.
- The address register output feeds back to its input using a multiplexer. The `addressstall` signal selects the multiplexer output.

Figure 2-3: Address Clock Enable Block Diagram
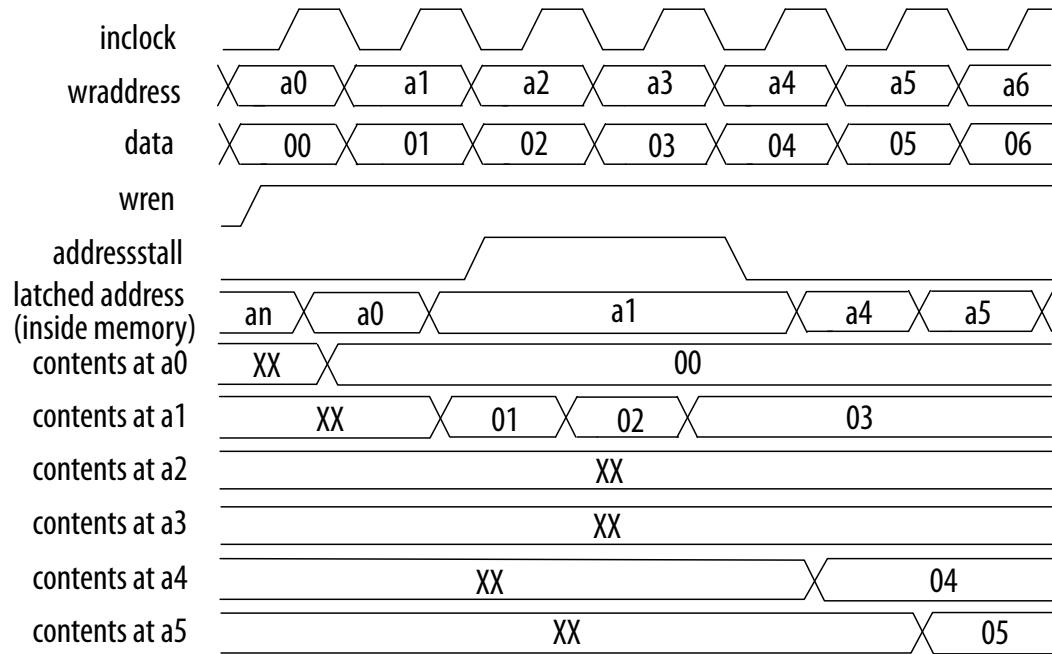
Address Clock Enable During Read Cycle Waveform

Figure 2-4: Address Clock Enable Waveform During Read Cycle

Address Clock Enable During Write Cycle Waveform

Figure 2-5: Address Clock Enable Waveform During Write Cycle

Asynchronous Clear
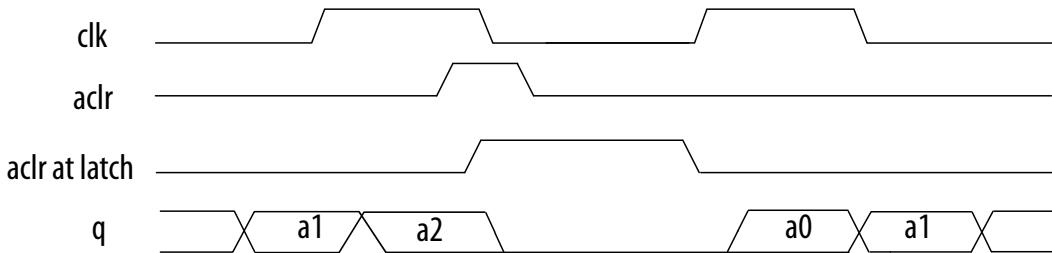
You can selectively enable asynchronous clear per logical memory using the RAM: 1-PORT and RAM: 2-PORT IP cores.

The M9k block supports asynchronous clear for:

- Read address registers: Asserting asynchronous clear to the read address register during a read operation might corrupt the memory content.
- Output registers: When applied to output registers, the asynchronous clear signal clears the output registers and the effects are immediately seen. If your RAM does not use output registers, you can still clear the RAM outputs using the output latch asynchronous clear feature.
- Output latches

Note: Input registers other than read address registers are not supported.

Figure 2-6: Output Latch Asynchronous Clear Waveform

Related Information

- **Internal Memory (RAM and ROM) User Guide.**

Resetting Registers in M9K Blocks

There are three ways to reset registers in the M9K blocks:

- Power up the device
- Use the `aclr` signal for output register only
- Assert the device-wide reset signal using the **DEV_CLRn** option

MAX 10 Embedded Memory Operation Modes

The M9K memory blocks allow you to implement fully-synchronous SRAM memory in multiple operation modes. The M9K memory blocks do not support asynchronous (unregistered) memory inputs.

Note: Violating the setup or hold time on the M9K memory block input registers may corrupt memory contents. This applies to both read and write operations.

Supported Memory Operation Modes

Table 2-2: Supported Memory Operation Modes in the M9K Embedded Memory Blocks

Memory Operation Mode	Related IP Core	Description
Single-port RAM	RAM: 1-PORT IP Core	Single-port mode supports non-simultaneous read and write operations from a single address. Use the read enable port to control the RAM output ports behavior during a write operation: • To show either the new data being written or the old data at that address, activate the read enable during a write operation. • To retain the previous values that are held during the most recent active read enable, perform the write operation with the read enable port deasserted.
Simple dual-port RAM	RAM: 2-PORT IP Core	You can simultaneously perform one read and one write operations to different locations where the write operation happens on port A and the read operation happens on port B.
True dual-port RAM	RAM: 2-PORT IP Core	You can perform any combination of two port operations: • two reads, two writes, or, • one read and one write at two different clock frequencies.

Memory Operation Mode	Related IP Core	Description
Single-port ROM	ROM: 1-PORT IP Core	Only one address port is available for read operation. You can use the memory blocks as a ROM. • Initialize the ROM contents of the memory blocks using a **.mif** or **.hex** file. • The address lines of the ROM are registered. • The outputs can be registered or unregistered. • The ROM read operation is identical to the read operation in the single-port RAM configuration.
Dual-port ROM	ROM: 2-PORT IP Core	The dual-port ROM has almost similar functional ports as single-port ROM. The difference is dual-port ROM has an additional address port for read operation. You can use the memory blocks as a ROM. • Initialize the ROM contents of the memory blocks using a **.mif** or **.hex** file. • The address lines of the ROM are registered. • The outputs can be registered or unregistered. • The ROM read operation is identical to the read operation in the single-port RAM configuration.
Shift-register	Shift Register (RAM-based) IP Core	You can use the memory blocks as a shift-register block to save logic cells and routing resources. The input data width (w), the length of the taps (m), and the number of taps (n) determine the size of a shift register (w × m × n). You can cascade memory blocks to implement larger shift registers.
FIFO	FIFO IP Core	You can use the memory blocks as FIFO buffers. • Use the FIFO IP core in single clock FIFO (SCFIFO) mode and dual clock FIFO (DCFIFO) mode to implement single- and dual-clock FIFO buffers in your design. • Use dual clock FIFO buffers when transferring data from one clock domain to another clock domain. • The M9K memory blocks do not support simultaneous read and write from an empty FIFO buffer.
Memory-based multiplier	ALTMEMMULT IP Core	You can use the memory blocks as a memory-based multiplier.

Related Information
MAX 10 Embedded Memory Related IPs

MAX 10 Embedded Memory Clock Modes

Clock Mode	Description	Modes				
		True Dual-Port	Simple Dual-Port	Single-Port	ROM	FIFO
Independent Clock Mode	A separate clock is available for the following ports: • Port A—Clock A controls all registers on the port A side. • Port B—Clock B controls all registers on the port B side.	Yes	—	—	Yes	—
Input/Output Clock Mode	• M9K memory blocks can implement input or output clock mode for single-port, true dual-port, and simple dual-port memory modes. • An input clock controls all input registers to the memory block, including `data`, `address`, `byteena`, `wren`, and `rden` registers. • An output clock controls the data-output registers.	Yes	Yes	Yes	Yes	—
Read or Write Clock Mode	• M9K memory blocks support independent clock enables for both the read and write clocks. • A read clock controls the data outputs, read address, and read enable registers. • A write clock controls the data inputs, write address, and write enable registers.	—	Yes	—	—	Yes
Single-Clock Mode	A single clock, together with a clock enable, controls all registers of the memory block.	Yes	Yes	Yes	Yes	Yes

Related Information

• Packed Mode Support on page 2-4
• Control Clocking to Reduce Power Consumption on page 3-5
• Output Read Data in Simultaneous Read and Write on page 2-10

Asynchronous Clear in Clock Modes

In all clock modes, asynchronous clear is available only for output latches and output registers. For independent clock mode, this is applicable on port A and port B.

Output Read Data in Simultaneous Read and Write

If you perform a simultaneous read/write to the same address location using the read or write clock mode, the output read data is unknown. If you want the output read data to be a known value, use single-clock or input/output clock mode and then select the appropriate read-during-write behavior in the RAM: 1-PORT and RAM: 2-PORT IP cores.

Related Information

Independent Clock Enables in Clock Modes

Table 2-3: Supported Clock Modes for Independent Clock Enables

Clock Mode	Description
Read/write	Supported for both the read and write clocks.
Independent	Supported for the registers of both ports.

MAX 10 Embedded Memory Configurations

Table 2-4: Single-port Memory Configurations for M9K Blocks

Configuration	M9K Block
Depth × width	8192×1
	4096×2
	2048×4
	1024×8
	1024×9
	512×16
	512×18
	256×32
	256×36

Port Width Configurations

The following equation defines the port width configuration: Memory depth (number of words) × Width of the data input bus.

- If your port width configuration (either the depth or the width) is more than the amount an internal memory block can support, additional memory blocks (of the same type) are used. For example, if you configure your M9K as 512 × 36, which exceeds the supported port width of 512 × 18, two M9Ks are used to implement your RAM.
- In addition to the supported configuration provided, you can set the memory depth to a non-power of two, but the actual memory depth allocated can vary. The variation depends on the type of resource implemented.
- If the memory is implemented in dedicated memory blocks, setting a non-power of two for the memory depth reflects the actual memory depth.
- When you implement your memory using dedicated memory blocks, refer to the Fitter report to check the actual memory depth.

Memory Configurations for Dual-Port Modes

Table 2-5: Simple Dual-port Memory Configurations

Read Port	Write Port								
	8192 × 1	4096 × 2	2048 × 4	1024 × 8	512 × 16	256 × 32	1024 × 9	512 × 18	256 × 36
8192 × 1	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
4096 × 2	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
2048 × 4	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
1024 × 8	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
512 × 16	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
256 × 32	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
1024 × 9	—	—	—	—	—	—	Yes	Yes	Yes
512 × 18	—	—	—	—	—	—	Yes	Yes	Yes
256 × 36	—	—	—	—	—	—	Yes	Yes	Yes

Table 2-6: True Dual-port Memory Configurations

Read Port	Write Port						
	8192 × 1	4096 × 2	2048 × 4	1024 × 8	512 × 16	1024 × 9	512 × 18
8192 × 1	Yes	Yes	Yes	Yes	Yes	—	—

Read Port	Write Port						
	8192 × 1	4096 × 2	2048 × 4	1024 × 8	512 × 16	1024 × 9	512 × 18
4096 × 2	Yes	Yes	Yes	Yes	Yes	—	—
2048 × 4	Yes	Yes	Yes	Yes	Yes	—	—
1024 × 8	Yes	Yes	Yes	Yes	Yes	—	—
512 × 16	Yes	Yes	Yes	Yes	Yes	—	—
1024 × 9	—	—	—	—	—	Yes	Yes
512 × 18	—	—	—	—	—	Yes	Yes

Maximum Block Depth Configuration

The **Set the maximum block depth** parameter allows you to set the maximum block depth of the dedicated memory block you use. You can slice the memory block to your desired maximum block depth. For example, the capacity of an M9K block is 9,216 bits, and the default memory depth is 8K, in which each address is capable of storing 1 bit (8K × 1). If you set the maximum block depth to 512, the M9K block is sliced to a depth of 512 and each address is capable of storing up to 18 bits (512 × 18).

Use this parameter to save power usage in your devices and to reduce the total number of memory blocks used. However, this parameter might increase the number of LEs and affects the design performance.

When the RAM is sliced shallower, the dynamic power usage decreases. However, for a RAM block with a depth of 256, the power used by the extra LEs starts to outweigh the power gain achieved by shallower slices.

The maximum block depth must be in a power of two, and the valid values vary among different dedicated memory blocks.

This table lists the valid range of maximum block depth for M9K memory blocks.

Table 2-7: Valid Range of Maximum Block Depth for M9K Memory Blocks

Memory Block	Valid Range
M9K	256 - 8K. The maximum block depth must be in a power of two.

The IP parameter editor prompts an error message if you enter an invalid value for the maximum block depth. Altera recommends that you set the value of the **Set the maximum block depth** parameter to **Auto** if you are unsure of the appropriate maximum block depth to set or the setting is not important for your design. The **Auto** setting enables the Compiler to select the maximum block depth with the appropriate port width configuration for the type of internal memory block of your memory.

There are several considerations that require your attention to ensure the success of your designs.

Implement External Conflict Resolution
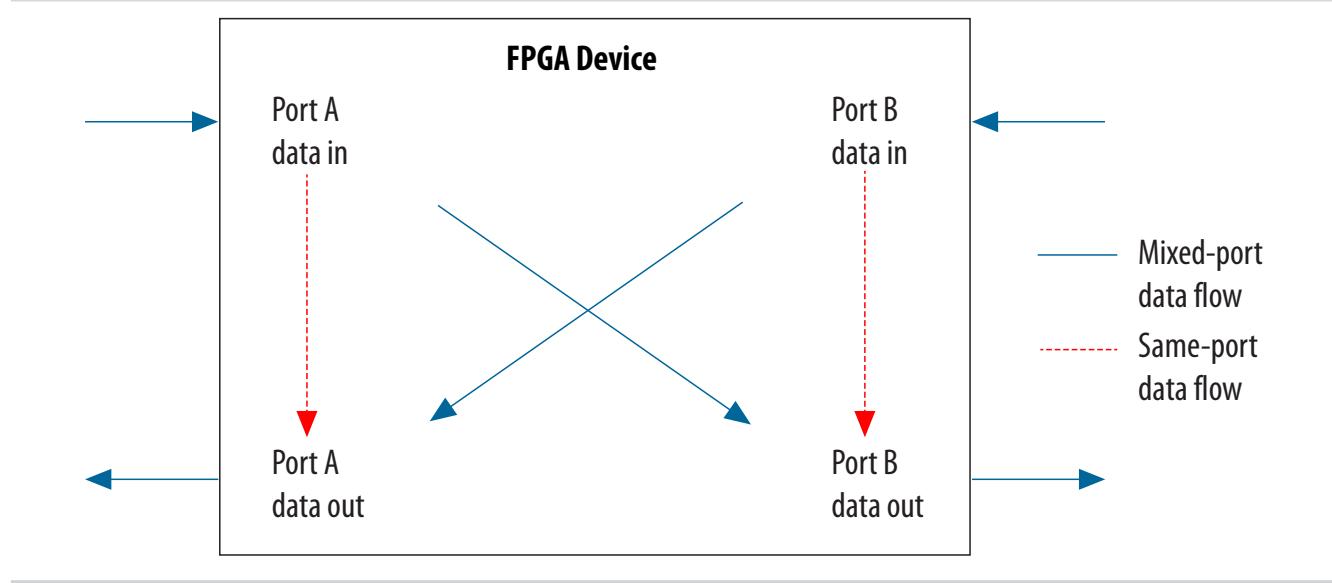
In the true dual-port RAM mode, you can perform two write operations to the same memory location. However, the memory blocks do not have internal conflict resolution circuitry.

To avoid unknown data being written to the address, implement external conflict resolution logic to the memory block.

Customize Read-During-Write Behavior

Customize the read-during-write behavior of the memory blocks to suit your design requirements.

Figure 3-1: Difference Between the Two Types of Read-during-Write Operations —Same Port and Mixed Port.

Related Information

[Read-During-Write](#) on page 2-3

Same-Port Read-During-Write Mode

The same-port read-during-write mode applies to a single-port RAM or the same port of a true dual-port RAM.

Table 3-1: Output Modes for Embedded Memory Blocks in Same-Port Read-During-Write Mode

This table lists the available output modes if you select the embedded memory blocks in the same-port read-during-write mode.

Output Mode	Description
"new data" (flow-through)	The new data is available on the rising edge of the same clock cycle on which the new data is written.
	When using New Data mode together with byte enable, you can control the output of the RAM.
	When byte enable is high, the data written into the memory passes to the output (flow-through).
	When byte enable is low, the masked-off data is not written into the memory and the old data in the memory appears on the outputs. Therefore, the output can be a combination of new and old data determined by byteena.
"don't care"	The RAM outputs reflect the old data at that address before the write operation proceeds.

Figure 3-2: Same-Port Read-During-Write: New Data Mode
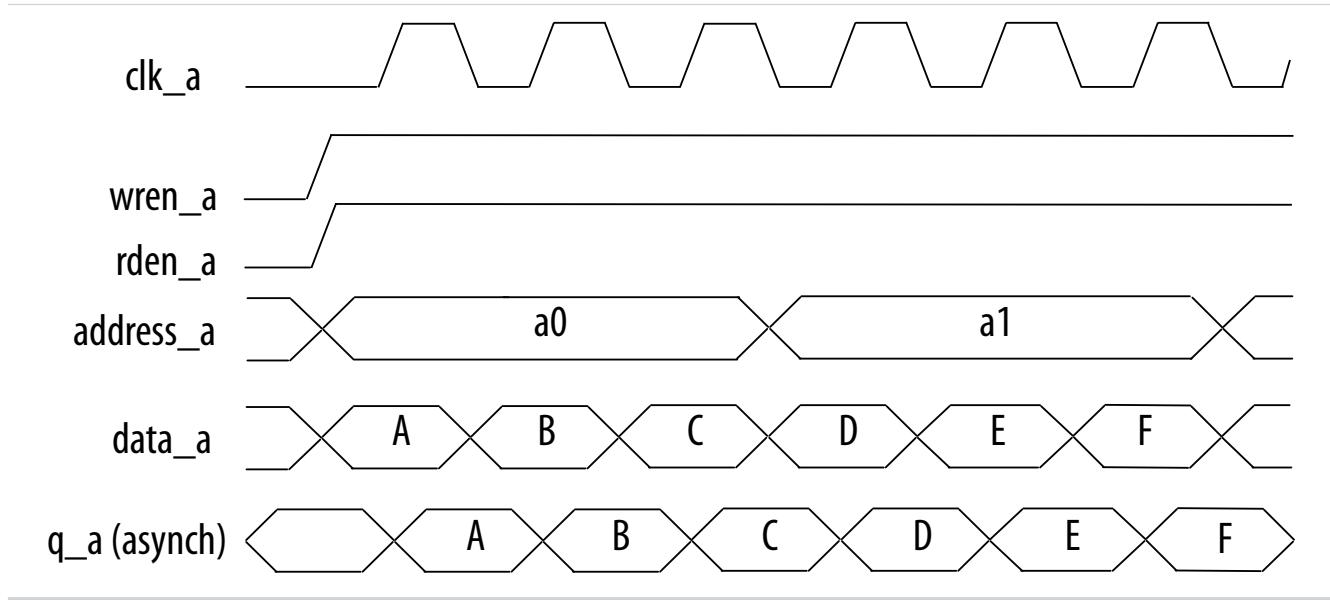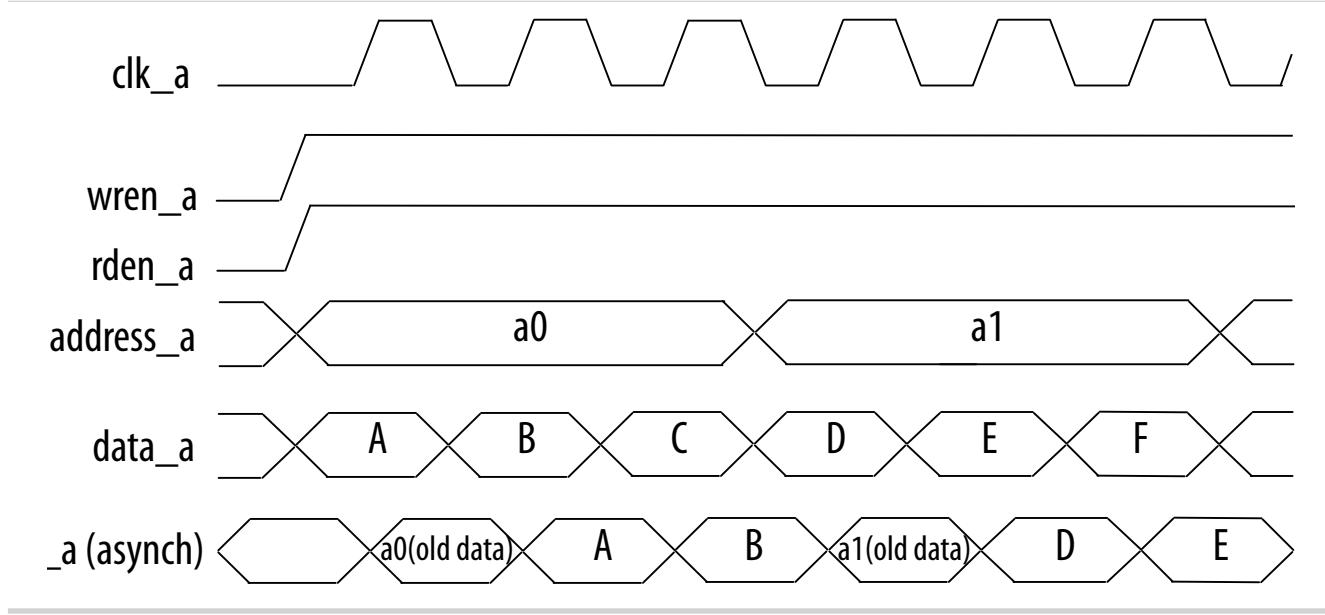
Figure 3-3: Same Port Read-During-Write: Old Data Mode
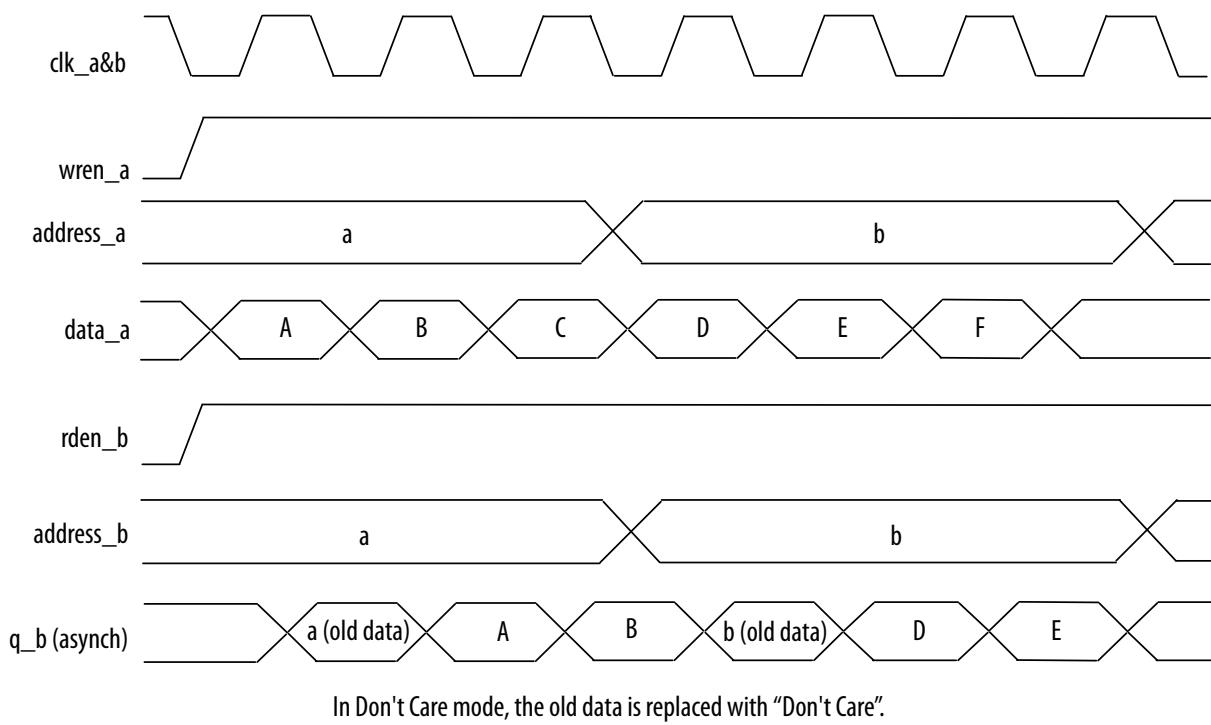
Mixed-Port Read-During-Write Mode

The mixed-port read-during-write mode applies to simple and true dual-port RAM modes where two ports perform read and write operations on the same memory address using the same clock—one port reading from the address, and the other port writing to it.

Table 3-2: Output Modes for RAM in Mixed-Port Read-During-Write Mode

Output Mode	Description
"old data"	A read-during-write operation to different ports causes the RAM output to reflect the "old data" value at the particular address.
"don't care"	The RAM outputs "don't care" or "unknown" value.

Figure 3-4: Mixed-Port Read-During-Write: Old Data Mode

In Don't Care mode, the old data is replaced with "Don't Care".

Mixed-Port Read-During-Write Operation with Dual Clocks

For mixed-port read-during-write operation with dual clocks, the relationship between the clocks determines the output behavior of the memory.

If You...	...Then
Use the same clock for the two clocks	The output is the old data from the address location.
Use different clocks	The output is unknown during the mixed-port read-during-write operation. This unknown value may be the old or new data at the address location, depending on whether the read happens before or after the write.

Consider Power-Up State and Memory Initialization

Consider the power-up state of the different types of memory blocks if you are designing logic that evaluates the initial power-up values, as listed in the following table:

Table 3-3: Initial Power-Up Values of Embedded Memory Blocks

Memory Type	Output Registers	Power Up Value
M9K	Used	Zero (cleared)
	Bypassed	Zero (cleared)

By default, the Quartus Prime software initializes the RAM cells to zero unless you specify a **.mif**.

All memory blocks support initialization with a **.mif**. You can create **.mif** files in the Quartus Prime software and specify their use with the RAM IP when you instantiate a memory in your design. Even if a memory is preinitialized (for example, using a **.mif**), it still powers up with its output cleared. Only the subsequent read after power up outputs the preinitialized values.

Only the following MAX 10 configuration modes support memory initialization:

- Single Compressed Image with Memory Initialization
- Single Uncompressed Image with Memory Initialization

Related Information

- **Selecting Internal Configuration modes.**
  Provides more information about selecting MAX 10 internal configuration modes.
- **MAX 10 Device Feature Options**
  Provides information on devices that support memory initialization.

Control Clocking to Reduce Power Consumption

Reduce AC power consumption in your design by controlling the clocking of each memory block:

- Use the read-enable signal to ensure that read operations occur only when necessary. If your design does not require read-during-write, you can reduce your power consumption by deasserting the read-enable signal during write operations, or during the period when no memory operations occur.
- Use the Quartus Prime software to automatically place any unused memory blocks in low-power mode to reduce static power.
- Create independent clock enable for different input and output registers to control the shut down of a particular register for power saving purposes. From the parameter editor, click **More Options** (beside the clock enable option) to set the available independent clock enable that you prefer.

Related Information
MAX 10 Embedded Memory Clock Modes on page 2-9

Send Feedback

Selecting Read-During-Write Output Choices

- Single-port RAM only supports same-port read-during-write, and the clock mode must be either single clock mode, or input/output clock mode.
- Simple dual-port RAM only supports mixed-port read-during-write, and the clock mode must be either single clock mode, or input/output clock mode.
- True dual-port RAM supports same port read-during-write and mixed-port read-during-write.
  - For same port read-during-write, the clock mode must be either single clock mode, input/output clock mode, or independent clock mode.
  - For mixed port read-during-write, the clock mode must be either single clock mode, or input/output clock mode.

Note: If you are not concerned about the output when read-during-write occurs and would like to improve performance, select **Don't Care**. Selecting **Don't Care** increases the flexibility in the type of memory block being used, provided you do not assign block type when you instantiate the memory block.

Table 3-4: Output Choices for the Same-Port and Mixed-Port Read-During-Write

Memory Block	Single-Port RAM	Simple Dual-Port RAM	True Dual-Port RAM	
	Same-Port Read-During-Write	Mixed-Port Read-During-Write	Same-Port Read-During-Write	Mixed-Port Read-During-Write
M9K	Don't Care New Data Old Data	Old Data Don't Care	New Data Old Data	Old Data Don't Care

2015.11.02

✉ **Subscribe**   💬 **Send Feedback**

The RAM: 1-Port IP core implements the single-port RAM memory mode.

Figure 4-1: RAM: 1-Port IP Core Signals with the Single Clock Option Enabled
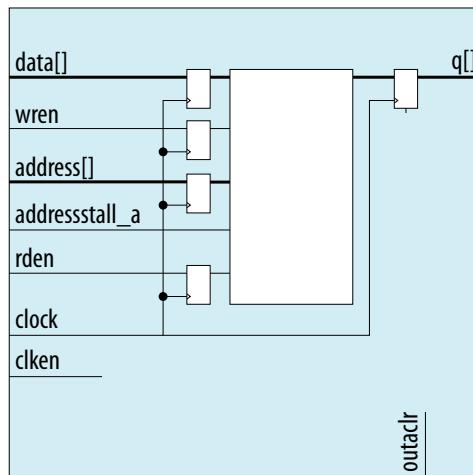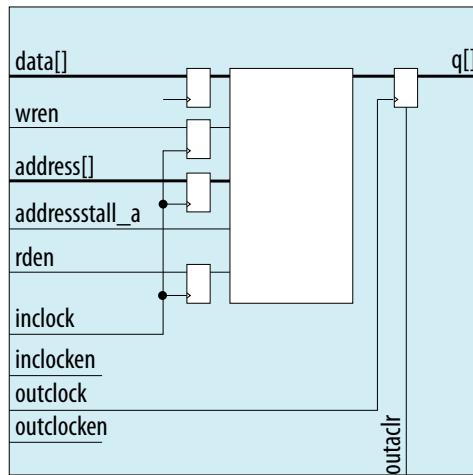
Figure 4-2: RAM: 1-Port IP Core Signals with the Dual Clock Option Enabled

**ISO
9001:2008
Registered**

ALTERA®

RAM: 1-Port IP Core Signals For MAX 10 Devices

Table 4-1: RAM:1-Port IP Core Input Signals

Signal	Required	Description
data	Yes	Data input to the memory. The `data` port is required and the width must be equal to the width of the `q` port.
address	Yes	Address input to the memory.
wren	Yes	Write enable input for the `wraddress` port.
addressstall_a	Optional	Address clock enable input to hold the previous address of `address_a` port for as long as the `addressstall_a` port is high.
clock	Yes	The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clocking modes: • Single clock—Connect your single source clock to `clock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `clock` port. All registered ports related to write operation, such as `data_a` port, `address_a` port, `wren_a` port, and `byteena_a` port are synchronized by the write clock. • Input/Output—Connect your input clock to `clock` port. All registered input ports are synchronized by the input clock. • Independent clock—Connect your port A clock to `clock` port. All registered input and output ports of port A are synchronized by the port A clock.
clkena	Optional	Clock enable input for `clock` port.
rden	Optional	Read enable input for `rdaddress` port.
aclr	Optional	Asynchronously clear the registered input and output ports. The asynchronous clear effect on the registered ports can be controlled through their corresponding asynchronous clear parameter, such as indata_aclr, wraddress_aclr, and so on.

Signal	Required	Description
`inclock`	Optional	The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `inclock` port. All registered ports related to write operation, such as `data` port, `wraddress` port, `wren` port, and `byteena` port are synchronized by the write clock. • Input/Output—Connect your input clock to `inclock` port. All registered input ports are synchronized by the input clock.
`inclocken`	Optional	Clock enable input for `inclock` port.
`outclock`	Optional	The following list describes which of your memory clock must be connected to the outclock port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your read clock to `outclock` port. All registered ports related to read operation, such as `rdaddress` port, `rdren` port, and `q` port are synchronized by the read clock. • Input/Output—Connect your output clock to `outclock` port. The registered `q` port is synchronized by the output clock.
`outclocken`	Optional	Clock enable input for `outclock` port.

Table 4-2: RAM:1-Port IP Core Output Ports

Signal	Required	Description
`q`	Yes	Data output from the memory. The `q` port must be equal to the width data port.

RAM: 1-Port IP Core Parameters For MAX 10 Devices

Table 4-3: RAM: 1-Port IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Parameter	Values	Description
Parameter Settings: Widths/Blk Type/Clks		

Parameter		Values	Description
How wide should the 'q' output bus be?		1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 36, 40, 48, 64, 72, 108, 128, 144, and 256.	Specifies the width of the 'q' output bus in bits.
How many <X>-bit words of memory?		32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, and 65536.	Specifies the number of <X>-bit words.
What should the memory block type be?			
Auto		On/Off	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
M9K		On/Off	
LC		On/Off	
Options	Use default logic cell style	On/Off	Specifies the logic cell implementation options. This option is enabled only when you choose LCs memory type.
	Use Stratix M512 emulation logic cell style	On/Off	
Set the maximum block depth to		Auto, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192	Specifies the maximum block depth in words. This option is disabled only when you choose LCs memory type.
What clocking method would you like to use?			
Single clock		On/Off	A single clock and a clock enable controls all registers of the memory block.
Dual clock: use separate 'input' and 'output' clocks		On/Off	An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables.
Parameter Settings: Regs/Clkens/Byte Enable/Aclrs			
Which ports should be registered?			
'data' and 'wren' input ports		—	This option is automatically enabled. Specifies whether to register the read or write input and output ports.
'address' input port		—	This option is automatically enabled. Specifies whether to register the read or write input and output ports.
'q' output port		On/Off	Specifies whether to register the read or write input and output ports.

Parameter		Values	Description
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.
More Options	Use clock enable for port A input registers	On/Off	Specify whether to use clock enable for port A input and output registers.
	Use clock enable for port A output registers	On/Off	Specify whether to use clock enable for port A input and output registers.
	Create an 'addressstall_a' input port	On/Off	Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.
Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.
More Options	'q' port	On/Off	Specifies whether the q port is cleared by the aclr port.
Create a 'rden' read enable signal		On/Off	Specifies whether to create a rden read enable signal.
Parameter Settings: Read During Write Option			
Single Port Read During Write Option			
What should the q output be when reading from a memory location being written to?		• Don't Care • New Data • Old Data	Specifies the output behavior when read-during-write occurs. • Don't Care—The RAM outputs "don't care" or "unknown" values for read-during-write operation. • New Data—New data is available on the rising edge of the same clock cycle on which it was written. • Old Data— The RAM outputs reflect the old data at that address before the write operation proceeds.
Get x's for write masked bytes instead of old data when byte enable is used		On/Off	Turn on this option to obtain 'X' on the masked byte.
Parameter Settings: Mem Init			
Do you want to specify the initial content of the memory?			

Parameter	Values	Description
No, leave it blank	On/Off	Specifies the initial content of the memory. Initialize the memory to zero.
Initialize memory content data to XX..X on power-up in simulation	On/Off	
Yes, use this file for the memory content data	On/Off	Allows you to specify a memory initialization file (**.mif**) or a hexadecimal (Intel-format) file (**.hex**). **Note:** The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the **Device and Pin Options** dialog box.
Allow In-System Memory Content Editor to capture and update content independently of the system clock	On/Off	Specifies whether to allow In-System Memory Content Editor to capture and update content independently of the system clock.
The 'Instance ID' of this RAM is	—	Specifies the RAM ID.

2015.11.02

The RAM: 2-PORT IP core implements the simple dual-port RAM and true dual-port RAM memory modes.

Figure 5-1: RAM: 2-Port IP Core Signals With the One Read Port and One Write Port, and Single Clock Options Enabled

Figure 5-2: RAM: 2-Port IP Core Signals with the One Read Port and One Write Port, and Dual Clock: Use Separate 'Read' and 'Write' Clocks Options Enabled
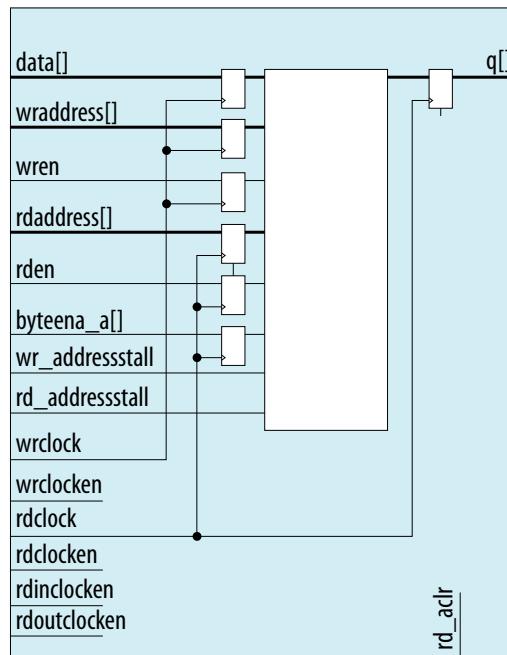
Figure 5-3: RAM: 2-Port IP Core Signals with the One Read Port and One Write Port, and Dual Clock: Use Separate 'Input' and 'Output' Clocks Options Enabled
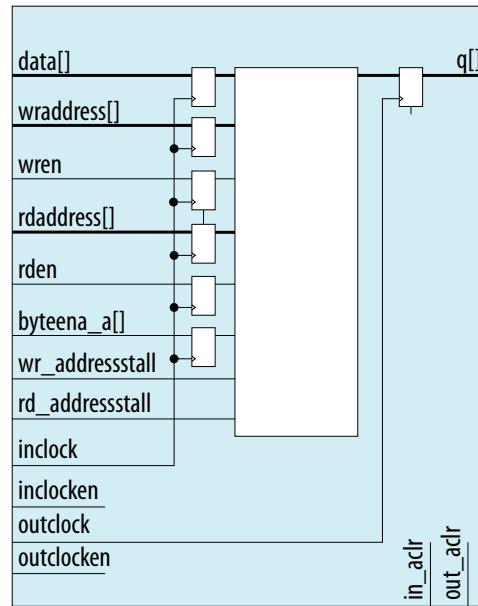
Figure 5-4: RAM: 2-Port IP Core Signals with the Two Read/Write Ports and Single Clock Options Enabled

Send Feedback
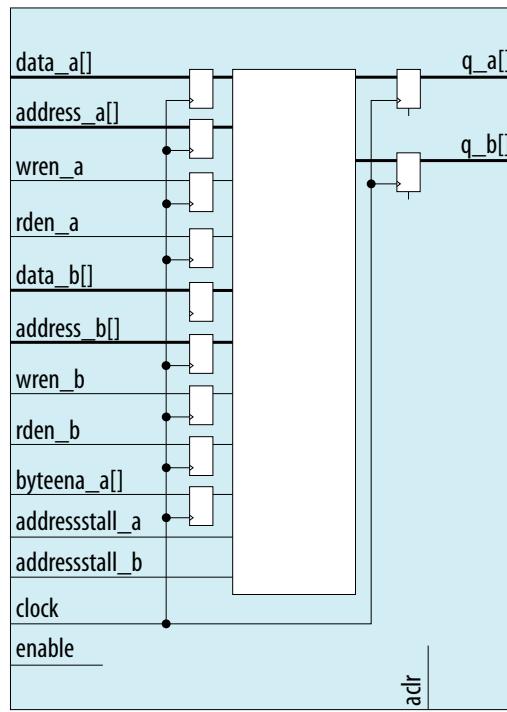
Figure 5-5: RAM: 2-Port IP Core Signals with the Two Read/Write Ports and Dual Clock: Use Separate 'Input' and 'Output' Clocks Options Enabled
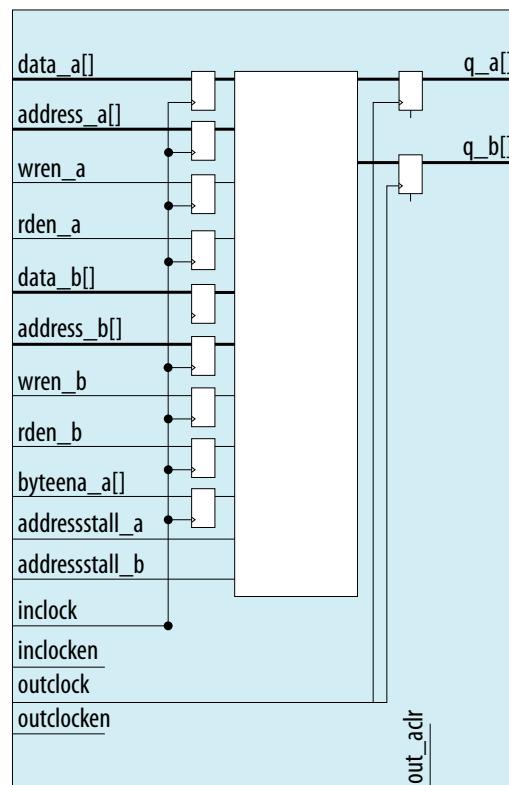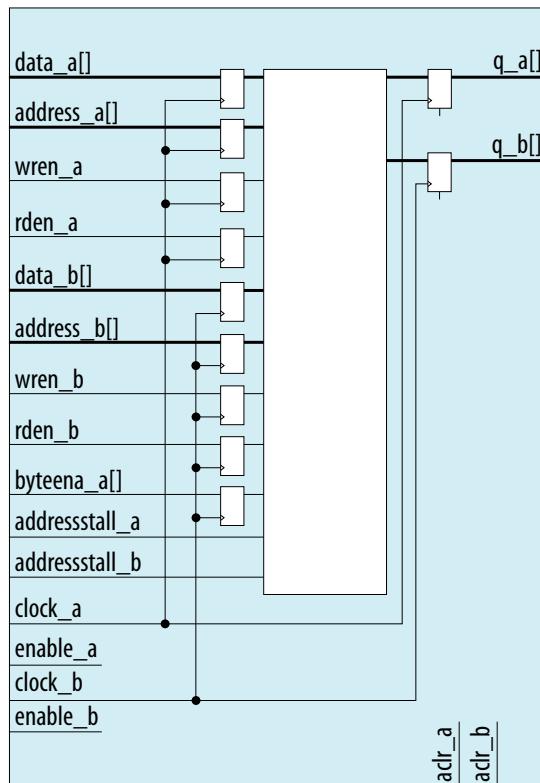
Figure 5-6: RAM: 2-Port IP Core Signals with the Two Read/Write Ports and Dual Clock: Use Separate for A and B Ports Options Enabled

RAM: 2-Ports IP Core Signals (Simple Dual-Port RAM) For MAX 10 Devices

Table 5-1: RAM: 2-Ports IP Core Input Signals (Simple Dual-Port RAM)

Signal	Required	Description
data	Yes	Data input to the memory. The data port is required and the width must be equal to the width of the q port.
wraddress	Yes	Write address input to the memory. The wraddress port is required and must be equal to the width of the raddress port.
wren	Yes	Write enable input for wraddress port. The wren port is required.
rdaddress	Yes	Read address input to the memory. The rdaddress port is required and must be equal to the width of wraddress port.

Signal	Required	Description
clock	Yes	The following list describes which of your memory clock must be connected to the `clock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `clock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `clock` port. All registered ports related to write operation, such as `data_a` port, `address_a` port, `wren_a` port, and `byteena_a` port are synchronized by the write clock. • Input/Output—Connect your input clock to `clock` port. All registered input ports are synchronized by the input clock. • Independent clock—Connect your port A clock to `clock` port. All registered input and output ports of port A are synchronized by the port A clock.
inclock	Yes	The following list describes which of your memory clock must be connected to the `inclock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `inclock` port. All registered ports related to write operation, such as `data` port, `wraddress` port, `wren` port, and `byteena` port are synchronized by the write clock. • Input/Output—Connect your input clock to `inclock` port. All registered input ports are synchronized by the input clock.
outclock	Yes	The following list describes which of your memory clock must be connected to the `outclock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your read clock to `outclock` port. All registered ports related to read operation, such as `rdaddress` port, `rdren` port, and `q` port are synchronized by the read clock. • Input/Output—Connect your output clock to `outclock` port. The registered `q` port is synchronized by the output clock.
rden	Optional	Read enable input for `rdaddress` port. The `rden` port is supported when the `use_eab` parameter is set to `OFF`. Instantiate the IP core if you want to use read enable feature with other memory blocks.

Signal	Required	Description
byteena_a	Optional	Byte enable input to mask the data_a port so that only specific bytes, nibbles, or bits of the data are written. The **byteena_a** port is not supported in the following conditions: • If the implement_in_les parameter is set to ON. • If the operation_mode parameter is set to ROM.
outclocken	Optional	Clock enable input for outclock port.
inclocken	Optional	Clock enable input for inclock port.

Table 5-2: RAM: 2-Ports IP Core Output Signals (Simple Dual-Port RAM)

Signal	Required	Description
q	Yes	Data output from the memory. The q port is required, and must be equal to the width data port.

RAM: 2-Port IP Core Signals (True Dual-Port RAM) for MAX 10 Devices

Table 5-3: RAM: 2-Port IP Core Input Signals (True Dual-Port RAM)

Signal	Required	Description
data_a	Optional	Data input to port A of the memory. The data_a port is required if the operation_mode parameter is set to any of the following values: • SINGLE_PORT • DUAL_PORT • BIDIR_DUAL_PORT
address_a	Yes	Address input to port A of the memory. The address_a port is required for all operation modes.
wren_a	Optional	Write enable input for address_a port. The wren_a port is required if you set the operation_mode parameter to any of the following values: • SINGLE_PORT • DUAL_PORT • BIDIR_DUAL_PORT
data_b	Optional	Data input to port B of the memory. The data_b port is required if the operation_mode parameter is set to BIDIR_DUAL_PORT.
address_b	Optional	Address input to port B of the memory. The address_b port is required if the operation_mode parameter is set to the following values: • DUAL_PORT • BIDIR_DUAL_PORT

Signal	Required	Description
wren_b	Yes	Write enable input for `address_b` port. The `wren_b` port is required if you set the `operation_mode` parameter to `BIDIR_DUAL_PORT`.
clock	Yes	The following list describes which of your memory clock must be connected to the `clock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `clock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `clock` port. All registered ports related to write operation, such as `data_a` port, `address_a` port, `wren_a` port, and `byteena_a` port are synchronized by the write clock. • Input/Output—Connect your input clock to `clock` port. All registered input ports are synchronized by the input clock. • Independent clock—Connect your port A clock to `clock` port. All registered input and output ports of port A are synchronized by the port A clock.
inclock	Yes	The following list describes which of your memory clock must be connected to the `inclock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `inclock` port. All registered ports related to write operation, such as `data` port, `wraddress` port, `wren` port, and `byteena` port are synchronized by the write clock. • Input/Output—Connect your input clock to `inclock` port. All registered input ports are synchronized by the input clock.
outclock	Yes	The following list describes which of your memory clock must be connected to the outclock port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your read clock to `outclock` port. All registered ports related to read operation, such as `rdaddress` port, `rdren` port, and `q` port are synchronized by the read clock. • Input/Output—Connect your output clock to `outclock` port. The registered `q` port is synchronized by the output clock.

Signal	Required	Description
rden_a	Optional	Read enable input for `address_a` port. The `rden_a` port is supported depending on your selected memory mode and memory block.
rden_b	Optional	Read enable input for `address_b` port. The `rden_b` port is supported depending on your selected memory mode and memory block.
byteena_a		Byte enable input to mask the `data_a` port so that only specific bytes, nibbles, or bits of the data are written. The `byteena_a` port is not supported in the following conditions: • If the `implement_in_les` parameter is set to `ON`. • If the `operation_mode` parameter is set to `ROM`.
addressstall_a	Optional	Address clock enable input to hold the previous address of `address_a` port for as long as the `addressstall_a` port is high.
addressstall_b	Optional	Address clock enable input to hold the previous address of `address_b` port for as long as the `addressstall_b` port is high.

Table 5-4: RAM:2-Port IP Core Output Signals (True Dual-Port RAM)

Signal	Required	Description
q_a	Yes	Data output from Port A of the memory. The `q_a` port is required if the `operation_mode` parameter is set to any of the following values: • `SINGLE_PORT` • `BIDIR_DUAL_PORT` • `ROM` The width of `q_a` port must be equal to the width of `data_a` port.
q_b	Yes	Data output from Port B of the memory. The `q_b` port is required if you set the `operation_mode` to the following values: • `DUAL_PORT` • `BIDIR_DUAL_PORT` The width of `q_b` port must be equal to the width of `data_b` port.

RAM: 2-Port IP Core Parameters for MAX 10 Devices

Table 5-5: RAM: 2-Port IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Option	Legal Values	Description
Parameter Settings: General		

Option	Legal Values	Description
How will you be using the dual port RAM?	• With one read port and one write port • With two read/write ports	Specifies how you use the dual port RAM.
How do you want to specify the memory size?	• As a number of words • As a number of bits	Determines whether to specify the memory size in words or bits.
Parameter Settings: Widths/ Blk Type		
How many <X>-bit words of memory?	—	Specifies the number of <X>-bit words.
Use different data widths on different ports	On/Off	Specifies whether to use different data widths on different ports.
Read/Write Ports	When you select **With one read port and one write port**, the following options are available: • How wide should the 'data_a' input bus be? • How wide should the 'q' output bus be? When you select **With two read/ write ports**, the following options are available: • How wide should the 'q_a' output bus be? • How wide should the 'q_b' output bus be?	1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 18, 32, 36, 64, 72, 108, 128, 144, 256, and 288	Specifies the width of the input and output ports. The **How wide should the 'q' output bus be?** and the **How wide should the 'q_b' output bus be?** options are only available when you turn on the **Use different data widths on different ports** parameter.

Option		Legal Values	Description
What should the memory block type be?		• **Auto** • **M9K** • **LCs**	Specifies the memory block type. The types of memory block that are available for selection depends on your target device. The **LCs** value is only available under the following conditions: • Turn on the **With one read port and one write port** option • Turn off **Use different data widths on different ports** option.
Option	How should the memory be implemented?	• Use default logic cell style • Use Stratix M512 emulation logic cell style	Specifies the logic cell implementation options. This option is enabled only when you choose LCs memory type.
Set the maximum block depth to		• Auto • 128 • 256 • 512 • 1024 • 2048 • 4096 • 8192	Specifies the maximum block depth in words.

Parameter Settings: Clks/Rd, Byte En

Option	Legal Values	Description
What clocking method would you like to use?	When you select **With one read port and one write port**, the following values are available: • Single clock • Dual clock: use separate 'input' and 'output' clocks • Dual clock: use separate 'read' and 'write' clocks When you select **With two read/write ports**, the following options are available: • Single clock • Dual clock: use separate 'input' and 'output' clocks • Dual clock: use separate clocks for A and B ports	Specifies the clocking method to use. • Single clock—A single clock and a clock enable controls all registers of the memory block. • Dual Clock: use separate 'input' and 'output' clocks—An input and an output clock controls all registers related to the data input and output to/ from the memory block including data, address, byte enables, read enables, and write enables. • Dual clock: use separate 'read' and 'write' clocks— A write clock controls the data-input, write-address, and write-enable registers while the read clock controls the data-output, read-address, and read-enable registers. • Dual clock: use separate clocks for A and B ports —Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively.
Create a 'rden' read enable signal	On/Off	Available when you select **With one read port and one write port** option.
Create a 'rden_a' and 'rden_b' read enable signal	On/Off	• Available when you select **With two read/write ports** option. • Specifies whether to create a read enable signal for Port A and B.

Send Feedback

Option		Legal Values	Description
Byte Enable Ports	Create byte enable for port A	On/Off	Specifies whether to create a byte enable for Port A and B. Turn on these options if you want to mask the input data so that only specific bytes, nibbles, or bits of data are written.
Parameter Settings: Regs/Clkens/Aclrs			
Which ports should be registered?	When you select **With one read port and one write port**, the following options are available: • Write input ports 'data_a', 'wraddress_a', and 'wren_a' • Read input ports 'rdaddress' and 'rden' • Read output port(s) 'q_a' and 'q_b' When you select **With two read/write ports**, the following options are available: • Write input ports 'data_a', 'wraddress_a', and 'wren_a' write input ports • Read output port(s) 'q'_a and 'q_b'	On/Off	Specifies whether to register the read or write input and output ports.
More Option	When you select **With one read port and one write port**, the following options are available: • 'q_b' port When you select **With two read/write ports**, the following options are available: • 'q_a' port • 'q_b' port	On/Off	The read and write input ports are turned on by default. You only need to specify whether to register the Q output ports.

Option		Legal Values	Description
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.

More Option	When you select **With one read port and one write port**, the following option is available:
	
	• Clock enable options
	
	    • Clock enable options: Use clock enable for write input registers
	
	• Address options
	
	    • Create an 'wr_addressstall' input port.
	    • Create an 'rd_addressstall' input port.
	
	When you select **With two read /write ports**, the following options are available:
	
	• Clock enable options
	
	    • Use clock enable for port A input registers
	    • Use clock enable for port A output registers
	
	• Address options
	
	    • Create an 'addressstall_a' input port.
	    • Create an 'addressstall_b' input port.
		On/Off	• Clock enable options—Clock enable for port B input and output registers are turned on by default. You only need to specify whether to use clock enable for port A input and output registers.
			• Address options—Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.

Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.

Option		Legal Values	Description
More Option	When you select **With one read port and one write port**, the following options are available: • 'rdaddress' port • 'q_b' port When you select **With two read /write ports**, the following options are available: • 'q_a' port • 'q_b' port	On/Off	Specifies whether the `raddress`, `q_a`, and `q_b` ports are cleared by the `aclr` port.
Parameter Settings: Output 1			
Mixed Port Read-During-Write for Single Input Clock RAM	When you select **With one read port and one write port**, the following option is available: • How should the q output behave when reading a memory location that is being written from the other port? When you select **With two read /write ports**, the following option is available: • How should the q_a and q_b outputs behave when reading a memory location that is being written from the other port?	• Old memory contents appear • I do not care (the outputs will be undefined)	Specifies the output behavior when read-during-write occurs. • **Old memory contents appear**— The RAM outputs reflect the old data at that address before the write operation proceeds. • **I do not care**—This option functions differently when you turn it on depending on the following memory block type you select: • When you set the memory block type to **Auto**, or **M9K**, the RAM outputs 'don't care' or "unknown" values for read-during-write operation without analyzing the timing path.

Option		Legal Values	Description
Do not analyze the timing between write and read operation. Metastability issues are prevented by never writing and reading at the same address at the same time.		On/Off	This option is automatically turned on when you turn on the **I do not care (The outputs will be undefined)** option. This option enables the RAM to output 'don't care' or 'unknown' values for read-during-write operation without analyzing the timing path.
Parameter Settings: Output 2 (This tab is only available when you select two read/write ports)			
Port A Read-During-Write Option	What should the 'q_a' output be when reading from a memory location being written to?	• New data • Old Data	Specifies the output behavior when read-during-write occurs. • **New Data**—New data is available on the rising edge of the same clock cycle on which it was written. • **Old Data**—The RAM outputs reflect the old data at that address before the write operation proceeds.
Port B Read-During-Write Option	What should the 'q_b' output be when reading from a memory location being written to?		
Get x's for write masked bytes instead of old data when byte enable is used		On/Off	This option is automatically turned on when you select the New Data value. This option obtains 'X' on the masked byte.
Parameter Settings: Mem Init			

Option	Legal Values	Description
Do you want to specify the initial content of the memory?	• No, leave it blank • Yes, use this file for the memory content data	Specifies the initial content of the memory. • To initialize the memory to zero, select **No, leave it blank**. • To use a Memory Initialization File (**.mif**) or a Hexadecimal (Intel-format) File (**.hex**), select **Yes, use this file for the memory content data**. **Note:** The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the **Device and Pin Options** dialog box.
The initial content file should conform to which port's dimension?	• PORT_A • PORT_B	Specifies which port's dimension that the initial content file should conform to.

✉ **Subscribe**   💬 **Send Feedback**

The ROM: 1-PORT IP core implements the single-port ROM memory mode.
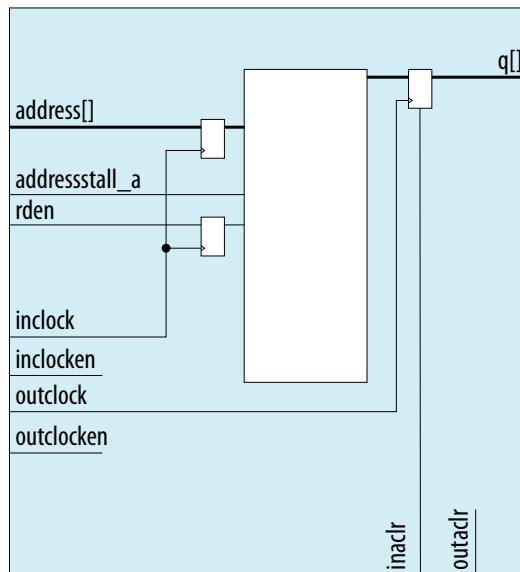
Figure 6-1: ROM: 1-PORT IP Core Signals with the Single Clock Option Enabled

Figure 6-2: ROM: 1-PORT IP Core Signals with the Dual Clock: Use Separate 'Input' and 'Output' Clocks Option Enabled

ROM: 1-PORT IP Core Signals For MAX 10 Devices

Table 6-1: ROM: 1-PORT IP Core Input Signals

Signal	Required	Description
address	Yes	Address input to the memory.
addressstall_a	Optional	Address clock enable input to hold the previous address of `address_a` port for as long as the `addressstall_a` port is high.
rden	Optional	Read enable input for `rdaddress` port. The `rden` port is supported when the `use_eab` parameter is set to `OFF`. Instantiate the IP if you want to use read enable feature with other memory blocks.

Signal	Required	Description
clock	Yes	The following list describes which of your memory clock must be connected to the `clock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `clock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `clock` port. All registered ports related to write operation, such as `data_a` port, `address_a` port, `wren_a` port, and `byteena_a` port are synchronized by the write clock. • Input/Output—Connect your input clock to `clock` port. All registered input ports are synchronized by the input clock. • Independent clock—Connect your port A clock to `clock` port. All registered input and output ports of port A are synchronized by the port A clock.
clken	Optional	Clock enable input for `clock` port.
inclock	Yes	The following list describes which of your memory clock must be connected to the `inclock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to `inclock` port. All registered ports related to write operation, such as `data` port, `wraddress` port, `wren` port, and `byteena` port are synchronized by the write clock. • Input/Output—Connect your input clock to `inclock` port. All registered input ports are synchronized by the input clock.
inclocken	Optional	Clock enable input for `inclock` port.
outclock	Yes	The following list describes which of your memory clock must be connected to the `outclock` port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to `inclock` port and `outclock` port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your read clock to `outclock` port. All registered ports related to read operation, such as `rdaddress` port, `rdren` port, and `q` port are synchronized by the read clock. • Input/Output—Connect your output clock to `outclock` port. The registered `q` port is synchronized by the output clock.

Signal	Required	Description
`outclocken`	Optional	Clock enable input for `outclock` port.

Table 6-2: ROM: 1-PORT IP Core Output Signals

Signal	Required	Description
`q`	Yes	Data output from the memory. The `q` port is required, and must be equal to the width data port.

ROM: 1-PORT IP Core Parameters for MAX 10 Devices

Table 6-3: ROM: 1-Port IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Option	Legal Values	Description
Parameter Settings: General		
How wide should the 'q' output bus be?	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 36, 40, 48, 64, 72, 108, 128, 144, and 256.	Specifies the width of the 'q' output bus in bits.
How many <X>-bit words of memory?	32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, and 65536.	Specifies the number of <X>-bit words.
What should the memory block type be?	• Auto • M9K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to	• Auto • 32 • 64 • 128 • 256 • 512 • 1024 • 2048 • 4096 • 8192	Specifies the maximum block depth in words.

Option	Legal Values	Description
What clocking method would you like to use?	• Single clock • Dual clock: use separate 'input' and 'output' clocks	Specifies the clocking method to use. • Single clock—A single clock and a clock enable controls all registers of the memory block. • Dual clock: use separate 'input' and 'output' clocks—An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables.

Parameter Settings: Regs/Clkens/Aclrs

Option		Legal Values	Description
Which ports should be registered?	• 'address' input port • 'q' output port	On/Off	Specifies whether to register the read or write input and output ports.
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.

More Options	• Clock enable options
	  • Use clock enable for port A input registers
	  • Use clock enable for port A output registers
	• Address options
	  • Create an 'address-stall_a' input port
		On/Off	• Clock enable options—Clock enable for port B input and output registers are turned on by default. You only need to specify whether to use clock enable for port A input and output registers.
			• Address options—Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.

Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.

Option		Legal Values	Description
More Options	• 'address' port • 'q' port	On/Off	Specifies whether the `address` and `q` ports are cleared by the `aclr` port.
Create a 'rden' read enable signal		On/Off	Specifies whether to create a `rden` read enable signal.
Parameter Settings: Mem Init			
Do you want to specify the initial content of the memory?		Yes, use this file for the memory content data.	Specifies the initial content of the memory. In ROM mode you must specify a Memory Initialization File (**.mif**) or a Hexadecimal (Intel-format) File (**.hex**). The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the **Device and Pin Options** dialog box.
Allow In-System Memory Content Editor to capture and update content independently of the system clock		On/Off	Specifies whether to allow In-System Memory Content Editor to capture and update content independently of the system clock.
The 'Instance ID' of this RAM is		—	Specifies the RAM ID.

This IP core implements the dual-port ROM memory mode. The dual-port ROM has almost similar functional ports as single-port ROM. The difference is dual-port ROM has an additional address port for read operation.

Figure 7-1: ROM: 2-PORT IP Core Signals with the Single Clock Option Enabled

Figure 7-2: ROM: 2-PORT IP Core Signals with the Dual Clock: Use Separate 'Input' and 'Output' Clocks Option Enabled
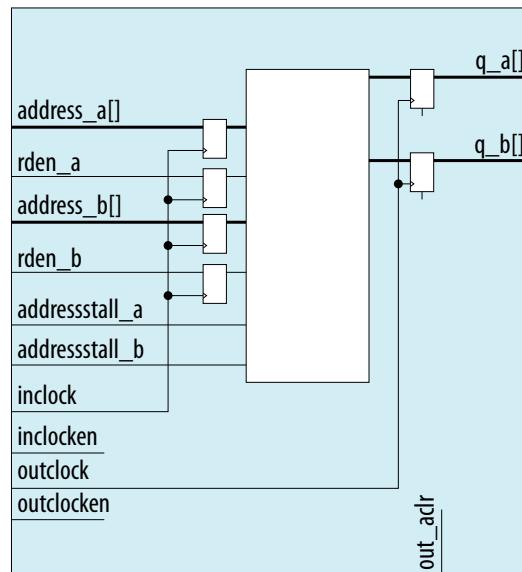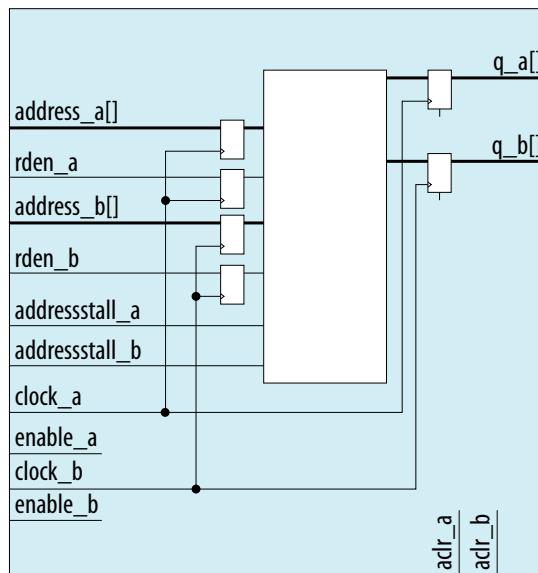
Figure 7-3: ROM: 2-PORT IP Core Signals with the Dual Clock: Use Separate Clocks for A and B Ports Option Enabled

ROM: 2-PORT IP Core Signals for MAX 10 Devices

Table 7-1: ROM: 2-PORT IP Core Input Signals

Signal	Required	Description
address_a	Yes	Address input to port A of the memory. The address_a port is required for all operation modes.
rden_a	Optional	Read enable input for address_a port. The rden_a port is supported depending on your selected memory mode and memory block.
address_b	Optional	Address input to port B of the memory. The address_b port is required if the operation_mode parameter is set to the following values: • DUAL_PORT • BIDIR_DUAL_PORT
rden_b	Optional	Read enable input for address_b port. The rden_b port is supported depending on your selected memory mode and memory block.
clock	Yes	The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to clock port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to clock port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock. • Input/Output—Connect your input clock to clock port. All registered input ports are synchronized by the input clock. • Independent clock—Connect your port A clock to clock port. All registered input and output ports of port A are synchronized by the port A clock.
addressstall_a	Optional	Address clock enable input to hold the previous address of address_a port for as long as the addressstall_a port is high.
addressstall_b	Optional	Address clock enable input to hold the previous address of address_b port for as long as the addressstall_b port is high.

Signal	Required	Description
inclock	Yes	The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your write clock to inclock port. The write clock synchronizes all registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port. • Input/Output—Connect your input clock to inclock port. The input clock synchronizes all registered input ports.
outclock	Yes	The following list describes which of your memory clock must be connected to the outclock port, and port synchronization in different clock modes: • Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. • Read/Write—Connect your read clock to outclock port. The read clock synchronizes all registered ports related to read operation, such as rdaddress port, rdren port, and q port. • Input/Output—Connect your output clock to outclock port. The output clock synchronizes the registered q port.
inclocken	Optional	Clock enable input for inclock port.
outclocken	Optional	Clock enable input for outclock port.
aclr	Optional	Asynchronously clear the registered input and output ports. The asynchronous clear effect on the registered ports can be controlled through their corresponding asynchronous clear parameter, such as indata_aclr and wraddress_aclr.

Table 7-2: ROM: 2-PORT IP Core Output Signals

Signal	Required	Description
q_a	Yes	Data output from port A of the memory. The q_a port is required if you set the operation_mode parameter to any of the following values: • SINGLE_PORT • BIDIR_DUAL_PORT • ROM The width of the q_a port must be equal to the width of the data_a port.

Signal	Required	Description
`q_b`	Yes	Data output from port B of the memory. The `q_b` port is required if you set the `operation_mode` parameter to the following values: `DUAL_PORT` `BIDIR_DUAL_PORT` The width of `q_b` port must be equal to the width of `data_b` port.

ROM:2-Port IP Core Parameters For MAX 10 Devices

Table 7-3: ROM:2-Port IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Option		Legal Values	Description
Parameter Settings: Widths/Blk Type			
How do you want to specify the memory size?		- As a number of words - As a number of bits	Determines whether to specify the memory size in words or bits.
How many \<X\>-bit words of memory?		—	Specifies the number of \<X\>-bit words.
Use different data widths on different ports		On/Off	Specifies whether to use different data widths on different ports.
Read Ports	How wide should the 'q_a' output bus be?	1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 18, 32, 36, 64, 72, 108, 128, 144, 256, and 288	Specifies the width of the input and output ports. **The How wide should the 'q_b' output bus be?** option is only available when you turn on the **Use different data widths on different ports** parameter.
	How wide should the 'q_b' output bus be?		
What should the memory block type be?		Auto, M9K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to		Auto, 128, 256, 512, 1024, 2048, 4096, 8192	Specifies the maximum block depth in words.
Parameter Settings: Clks/Rd, Byte En			

Option		Legal Values	Description
What clocking method would you like to use?		- Single clock - Dual clock: use separate 'input' and 'output' clocks - Dual clock: use separate clocks for A and B ports	Specifies the clocking method to use. **Single clock**—A single clock and a clock enable controls all registers of the memory block. **Dual Clock: use separate 'input' and 'output' clocks**—An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables. **Dual clock: use separate clocks for A and B ports**—Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively.
Create a 'rden_a' and 'rden_b' read enable signal		On/Off	Specifies whether to create read enable signals.
Parameter Settings: Regs/Clkens/Aclrs			
Which ports should be registered?	- Write input ports - Read output port(s)	On/Off	Specifies whether to register the read or write input and output ports.
More Options	- Input ports 'address_a' port 'address_b' port - Q output ports 'q_a' port 'q_b' port	On/Off	The read and write input ports are turned on by default. You only need to specify whether to register the Q output ports.
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.

Option		Legal Values	Description

More Options	• Clock enable options
	    • Use clock enable for port A input registers
	    • Use clock enable for port A output registers
	• Address options
	    • Create an 'addressstall_a' input port.
	    • Create an 'addressstall_b' input port.
	•
		On/Off	• Clock enable options—Clock enable for port B input and output registers are turned on by default. You only need to specify whether to use clock enable for port A input and output registers.
			• Address options—Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.

Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.
More Options	• 'q_a' port • 'q_b' port	On/Off	Specifies whether the 'q_a', and 'q_b' ports are cleared by the aclr port.

Parameter Settings: Mem Init

Option	Legal Values	Description
Do you want to specify the initial content of the memory?	Yes, use this file for the memory content data	Specifies the initial content of the memory. • To initialize the memory to zero, select **No, leave it blank**. • To use a Memory Initialization File (**.mif**) or a Hexadecimal (Intel-format) File (**.hex**), select **Yes, use this file for the memory content data**. **Note:** The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the **Device and Pin Options** dialog box.
The initial content file should conform to which port's dimension?	• PORT_A • PORT_B	Specifies which port's dimension that the initial content file should conform to.

The Shift Register (RAM-based) IP core contains additional features not found in a conventional shift register. You can use the memory blocks as a shift-register block to save logic cells and routing resources. You can cascade memory blocks to implement larger shift registers.

Figure 8-1: Shift Register (RAM-based) IP Core Signals

Shift Register (RAM-based) IP Core Signals for MAX 10 Devices

Table 8-1: Shift Register (RAM-based) IP Core Input Signals

Signal	Required	Description
shiftin[]	Yes	Data input to the shifter. Input port WIDTH bits wide.
clock	Yes	Positive-edge triggered clock.
clken	No	Clock enable for the clock port. clken defaults to VCC.
aclr	No	Asynchronously clears the contents of the shift register chain. The shiftout outputs are cleared immediately upon the assertion of the aclr signal.

Table 8-2: Shift Register (RAM-based) IP Core Output Signals

Signal	Required	Description
shiftout[]	Yes	Output from the end of the shift register. Output port WIDTH bits wide.

ALTERA®

Signal	Required	Description
taps[]	Yes	Output from the regularly spaced taps along the shift register. Output port WIDTH * NUMBER_OF_TAPS wide. This port is an aggregate of all the regularly spaced taps (each WIDTH bits) along the shift register.

Shift Register (RAM-based) IP Core Parameters for MAX 10 Devices

Table 8-3: Shift Register (RAM-based) IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Option	Values	Description
How wide should the "shiftin" input and the "shiftout" output buses be?	1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 24, 32, 48, 64, 96, 128, 192, and 256.	Specifies the width of the input pattern.
How many taps would you like?	1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 24, 32, 48, 64, 96, and 128.	Specifies the number of regularly spaced taps along the shift register.
Create groups for each tap output	On/Off	Creates groups for each tap output.
How wide should the distance between taps be?	3, 4, 5, 6, 7, 8, 16, 32, 64, and 128	Specifies the distance between the regularly spaced taps in clock cycles. This number translates to the number of RAM words that will be used. The value must be at least **3**.
Create a clock enable port	On/Off	Creates the clken port
Create an asynchronous clear port	On/Off	Creates the aclr port.
What should the RAM block type be?	Auto, M9K	Specifies the RAM block type.
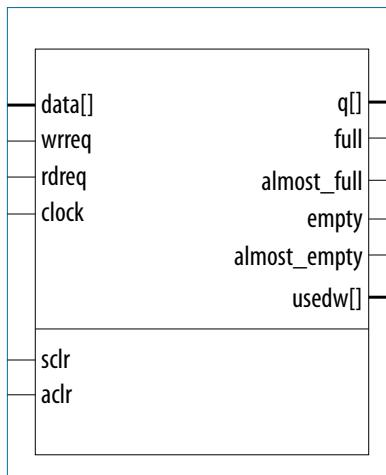
The FIFO IP core implements the FIFO mode, enabling you to use the memory blocks as FIFO buffers.

- Use the FIFO IP core in single clock FIFO (SCFIFO) and dual clock FIFO (DCFIFO) modes to implement single- and dual-clock FIFO buffers in your design.
- Dual clock FIFO buffers are useful when transferring data from one clock domain to another clock domain.
- The M9K memory blocks do not support simultaneous read and write from an empty FIFO buffer.
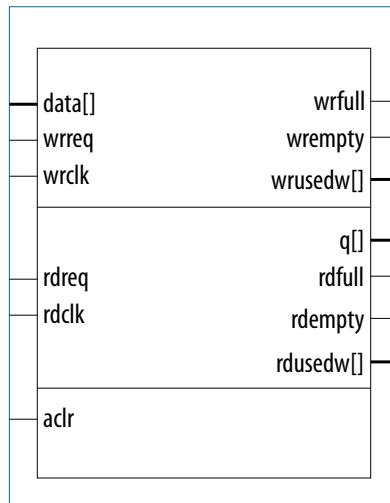
Figure 9-1: FIFO IP Core: SCFIFO Mode Signals

Figure 9-2: FIFO IP Core: DCFIFO Mode Signals

FIFO IP Core Signals for MAX 10 Devices

Table 9-1: FIFO IP Core Input Signals

Signal	Required	Description
clock	Yes	Positive-edge-triggered clock.
wrclk	Yes	Positive-edge-triggered clock. Synchronizes the following ports: • data • wrreq • wrfull • wrempty • wrusedw
rdclk	Yes	Positive-edge-triggered clock. Synchronizes the following ports: • q • rdreq • rdfull • rdempty • rdusedw
data	Yes	Holds the data to be written in the FIFO IP core when the wrreq signal is asserted. If you manually instantiate the FIFO IP core, ensure that the port width is equal to the **How wide should the FIFO be?** parameter.

Signal	Required	Description
wrreq	Yes	Assert this signal to request for a write operation. Ensure that the following conditions are met: • Do not assert the wrreq signal when the full (for the FIFO IP core in SCFIFO mode) or wrfull (for the FIFO IP core in DCFIFO mode) port is high. Enable the overflow protection circuitry or turn on the **Disable overflow checking. Writing to a full FIFO will corrupt contents** parameter so that the FIFO IP core can automatically disable the wrreq signal when it is full. • The wrreq signal must meet the functional timing requirement based on the full or wrfull signal. • Do not assert the wrreq signal during the deassertion of the aclr signal. Violating this requirement creates a race condition between the falling edge of the aclr signal and the rising edge of the write clock if the wrreq port is set to high.
rdreq	Yes	Assert this signal to request for a read operation. The rdreq signal acts differently in normal synchronous FIFO mode and show-ahead mode synchronous FIFO modes. Ensure that the following conditions are met: • Do not assert the rdreq signal when the empty (for the FIFO IP core in SCFIFO mode) or rdempty (for the FIFO IP core in DCFIFO mode) port is high. Enable the underflow protection circuitry or turn on the **Disable underflow checking. Reading from an empty FIFO will corrupt contents** parameter so that the FIFO IP core can automatically disable the rdreq signal when it is empty. The rdreq signal must meet the functional timing requirement based on the empty or rdempty signal.
sclr	No	Assert this signal to clear all the output status ports, but the effect on the q output may vary for different FIFO configurations. There are no minimum number of clock cycles for aclr signals that must remain active.
aclr	No	

Table 9-2: FIFO IP Core Output Signals

Signal	Required	Description
q		Shows the data read from the read request operation. In SCFIFO and DCFIFO modes, the width of the q port must be equal to the width of the data port. If you manually instantiate the IPs, ensure that the port width is equal to the **How wide should the FIFO be?** parameter. In DCFIFO_MIXED_WIDTHS mode, the width of the q port can be different from the width of the data port. If you manually instantiate the IP, ensure that the width of the q port is equal to the **Use a different output width** parameter. The IP supports a wide write port with a narrow read port, and vice versa. However, the width ratio is restricted by the type of RAM block, and in general, are in the power of 2.

Signal	Required	Description

`full`
`wrfull`
`rdfull`
	No	When asserted, the FIFO IP core is considered full. Do not perform write request operation when the FIFO IP core is full. In general, the `rdfull` signal is a delayed version of the `wrfull` signal. However, the `rdfull` signal functions as a combinational output instead of a derived version of the `wrfull` signal. Therefore, you must always refer to the `wrfull` port to ensure whether or not a valid write request operation can be performed, regardless of the target device.

`empty`
`wrempty`
`rdempty`
	No	When asserted, the FIFO IP core is considered empty. Do not perform read request operation when the FIFO IP core is empty. In general, the `wrempty` signal is a delayed version of the `rdempty` signal. However, the `wrempty` signal functions as a combinational output instead of a derived version of the `rdempty` signal. Therefore, you must always refer to the `rdempty` port to ensure whether or not a valid read request operation can be performed, regardless of the target device.

`almost_full`	No	Asserted when the `usedw` signal is greater than or equal to the **Almost full** parameter. It is used as an early indication of the `full` signal.
`almost_empty`	No	Asserted when the `usedw` signal is less than the **Almost empty** parameter. It is used as an early indication of the `empty` signal.

`usedw`
`wrusedw`
`rdusedw`
	No	Show the number of words stored in the FIFO. Ensure that the port width is equal to the **usedw**[] parameter if you manually instantiate the FIFO IP core in SCFIFO or DCFIFO modes. In DCFIFO_MIXED_WIDTH mode, the width of the `wrusedw` and `rdusedw` ports must be equal to the **usedw**[] and **Use a different output width** parameters respectively.

FIFO IP Core Parameters for MAX 10 Devices

Table 9-3: FIFO IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Parameter	HDL Parameter	Description
How wide should the FIFO be?	`lpm_width`	Specifies the width of the `data` and `q` ports for the FIFO IP core in SCFIFO mode and DCFIFO mode. For the FIFO IP core in DCFIFO_MIXED_WIDTHS mode, this parameter specifies only the width of the `data` port.
Use a different output width[1]	`lpm_width_r`	Specifies the width of the `q` port for the FIFO IP core in DCFIFO_MIXED_WIDTHS mode.

---

[1] Applicable in DCFIFO_MIXED_WIDTHS mode only.

Parameter	HDL Parameter	Description
Usedw[]	`lpm_widthu`	Specifies the width of the `usedw` port for the FIFO IP core in SCFIFO mode, or the width of the `rdusedw` and `wrusedw` ports for the FIFO IP core in DCFIFO mode. For the FIFO IP core in DCFIFO_MIXED_WIDTHS mode, it only represents the width of the `wrusedw` port.
How deep should the FIFO be?	`lpm_numwords`	Specifies the depths of the FIFO you require. The value must be at least **4**. The value assigned must comply with the $2^{\text{LPM\_WIDTHU}}$ equation.
Which kind of read access do you want with the rdreq signal?	`lpm_showahead`	Specifies whether the FIFO is in normal synchronous FIFO mode or show-ahead mode synchronous FIFO mode. Fornormal synchronous FIFO mode, the FIFO IP core treats the `rdreq` port as a normal read request that only performs read operation when the port is asserted. For show-ahead mode synchronous FIFO mode, the FIFO IP core treats the `rdreq` port as a read-acknowledge that automatically outputs the first word of valid data in the FIFO IP core (when the `empty` or `rdempty` port is low) without asserting the `rdreq` signal. Asserting the `rdreq` signal causes the FIFO IP core to output the next data word, if available. If you turn on this parameter, you may reduce performance.
Do you want a common clock for reading and writing the FIFO?	`lpm_type`	Identifies the library of parameterized modules (LPM) entity name. The values are **SCFIFO** and **DCFIFO.**
Disable overflow checking. Writing to a full FIFO will corrupt contents	`overflow_checking`	Specifies whether or not to enable the protection circuitry for overflow checking that disables the `wrreq` port when the FIFO IP core is full. This parameter is enabled by default.
Disable underflow checking. Reading from an empty FIFO will corrupt contents.	`underflow_checking`	Specifies whether or not to enable the protection circuitry for underflow checking that disables the `rdreq` port when the FIFO IP core is empty. This parameter is enabled by default. Note that reading from an empty SCFIFO mode gives unpredictable results.
Add an extra MSB to usedw [2]	`add_usedw_msb_bit`	Increases the width of the `rdusedw` and `wrusedw` ports by one bit. By increasing the width, it prevents the FIFO IP core from rolling over to zero when it is full. This parameter is disabled by default.

[2] Applicable in DCFIFO mode only.

Parameter	HDL Parameter	Description
How many sync stages?[2]	`rdsync_ delaypipe`	Specifies the number of synchronization stages in the cross clock domain. The value of the rdsync_delaypipe parameter relates the synchronization stages from the write control logic to the read control logic; the wrsync_delaypipe parameter relates the synchronization stages from the read control logic to the write control logic. Use these parameters to set the number of synchronization stages if the clocks are not synchronized, and set the clocks_are_synchronized parameter to FALSE. The actual synchronization stage implemented relates variously to the parameter value assigned, depends on the target device.
How many sync stages?[2]	`wrsync_ delaypipe`	Specifies the number of synchronization stages in the cross clock domain. The value of the rdsync_delaypipe parameter relates the synchronization stages from the write control logic to the read control logic; the wrsync_delaypipe parameter relates the synchronization stages from the read control logic to the write control logic. Use these parameters to set the number of synchronization stages if the clocks are not synchronized, and set the clocks_are_synchronized parameter to FALSE. The actual synchronization stage implemented relates variously to the parameter value assigned, depends on the target device.
Implement FIFO storage with logic cells only, even if the device contains memory blocks.	`use_eab`	Specifies whether or not the FIFO IP core is constructed using RAM blocks. This parameter is disabled by default. If you turn off this parameter, the FIFO IP core is implemented in logic elements, regardless of the memory block type assigned to the **What should the memory block type be** parameter.
Add circuit to synchronize 'aclr' input with 'wrclk'[2]	`write_aclr_ synch`	Specifies whether or not to add a circuit that causes the `aclr` port to be internally synchronized by the `wrclk` clock. Adding the circuit prevents the race condition between the `wrreq` and `aclr` ports that could corrupt the FIFO IP core. This parameter is disabled by default.
Add circuit to synchronize 'aclr' input with 'rdclk'	`read_aclr_ synch`	Specifies whether or not to add a circuit that causes the `aclr` port to be internally synchronized by the `rdclk` clock. Adding the circuit prevents the race condition between the `rdreq` and `aclr` ports that could corrupt the FIFO IP core. This parameter is disabled by default.

Parameter	HDL Parameter	Description
Which type of optimization do you want?[2]	`clocks_are_synchronized`	Specifies whether or not the write and read clocks are synchronized, which in turn determines the number of internal synchronization stages added for stable operation of the FIFO. The values are TRUE and FALSE. If omitted, the default value is FALSE. You must only set the parameter to TRUE if the write clock and the read clock are always synchronized and they are multiples of each other. Otherwise, set this to FALSE to avoid metastability problems. If the clocks are not synchronized, set the parameter to FALSE, and use the rdsync_delaypipe and wrsync_delaypipe parameters to determine the number of synchronization stages required.
What should the memory block type be	`ram_block_type`	Specifies the target device's memory block to be used. To get the proper implementation based on the RAM configuration that you set, allow the Quartus Prime software to automatically choose the memory type by ignoring this parameter and turn on the **Implement FIFO storage with logic cells only, even if the device contains memory blocks.** parameter. This gives the Compiler the flexibility to place the memory function in any available memory resource based on the FIFO depth required.
Would you like to register the output to maximize the performance but use more area?[3]	`add_ram_output_register`	Specifies whether to register the q output. The values are **Yes (best speed)** and **No (smallest area)**. The default value is **No (smallest area)**.
Becomes true when usedw[] is greater than or equal to:[3]	`almost_full_value`	Sets the threshold value for the almost_full port. When the number of words stored in the FIFO IP core is greater than or equal to this value, the almost_full port is asserted.
Almost full[3]		
Almost empty[3]	`almost_empty_value`	Sets the threshold value for the almost_empty port. When the number of words stored in the FIFO IP core is less than this value, the almost_empty port is asserted.
Becomes true when usedw[] is less than:[3]		
Currently selected device family	`intended_device_family`	Specifies the intended device that matches the device set in your Quartus Prime project. Use this parameter only for functional simulation.

---

[3] Applicable in SCFIFO mode only.
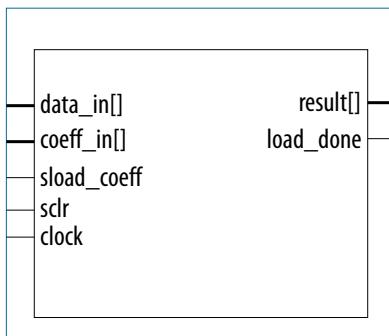
UG-M10MEMORY ✉ **Subscribe** 💬 **Send Feedback**

The ALTMEMMULT IP core creates only memory-based multipliers using on-chip memory blocks found in M9K memory blocks.

Figure 10-1: ALTMEMMULT IP Core Signals

ALTMEMMULT IP Core Signals for MAX 10 Devices

Table 10-1: ALTMEMMULT IP Core Input Signals

Signal	Required	Description
clock	Yes	Clock input to the multiplier.
coeff_in[]	No	Coefficient input port for the multiplier. The size of the input port depends on the WIDTH_C parameter value.
data_in[]	Yes	Data input port to the multiplier. The size of the input port depends on the WIDTH_D parameter value.
sclr	No	Synchronous clear input. If unused, the default value is active high.
sel[]	No	Fixed coefficient selection. The size of the input port depends on the WIDTH_S parameter value.
sload_coeff	No	Synchronous load coefficient input port. Replaces the current selected coefficient value with the value specified in the coeff_in input.

Signal	Required	Description
sload_data	No	Synchronous load data input port. Signal that specifies new multiplication operation and cancels any existing multiplication operation. If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the sload_data input port is ignored.

Table 10-2: ALTMEMMULT IP Core Output Signals

Signal	Required	Description
result[]	Yes	Multiplier output port. The size of the input port depends on the WIDTH_R parameter value.
result_valid	Yes	Indicates when the output is the valid result of a complete multiplication. If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the result_valid output port is not used.
load_done	No	Indicates when the new coefficient has finished loading. The load_done signal asserts when a new coefficient has finished loading. Unless the load_done signal is high, no other coefficient value can be loaded into the memory.

ALTMEMMULT IP Core Parameters for MAX 10 Devices

Table 10-3: ALTMEMMULT IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Option	Values	Description
How wide should the 'data_in' input bus be?	2, 3, 4, 5, 6, 7, 8, 16, 24, and 32	Specifies the width of the data_in port.
What is the representation of 'data_in'?	SIGNED, UNSIGNED	Specifies whether the data_in input port is signed or unsigned.
How wide should the coefficient be?	2, 3, 4, 5, 6, 7, 8, 16, 24	Specifies the width of the coeff_in port.
What is the representation of the coefficient?	SIGNED, UNSIGNED	Specifies whether the coeff_in input port and the pre-loaded coefficients are signed or unsigned.
What is the value of the initial coefficient?	0, 1, 2, 3, and 4	Specifies value of the first fixed coefficient.
Create ports to allow loading coefficients	On/Off	Creates the coeff_in and sload_coeff port.
Create a synchronous clear input	On/Off	Creates the sclr port.
What should the RAM block type be?	Auto, M9K	Specifies the RAM block type.

Document Revision History for MAX 10 Embedded Memory User Guide

Date	Version	Changes
November 2015	2015.11.02	• Revised the title for the tables in the Embedded Memory Configuration topic. • Added a link to the MAX 10 FPGA Device Overview in the Consider Power-Up State and Memory Initialization topic. • Changes instances of Quartus II to Quartus Prime.
May 2015	2015.05.04	• Updated 'Yes, use this file for the memory content data' parameter note for RAM:1-Port, RAM:2-Port, ROM:1-Port, and ROM:2-Port. • Added information about the internal configuration mode that supports memory initialization in 'Consider Power-Up State and Memory Initialization'
September 2014	2014.09.22	Initial release.

MAX 10 Embedded Multipliers User Guide

ALTERA

now part of Intel

Contents

MAX 10 Embedded Multiplier Block Overview

1

The embedded multiplier is configured as either one 18 x 18 multiplier or two 9 x 9 multipliers. For multiplications greater than 18 x 18, the Quartus® Prime software cascades multiple embedded multiplier blocks together. There are no restrictions on the data width of the multiplier but the greater the data width, the slower the multiplication process.

Figure 1-1: Embedded Multipliers Arranged in Columns with Adjacent LABS
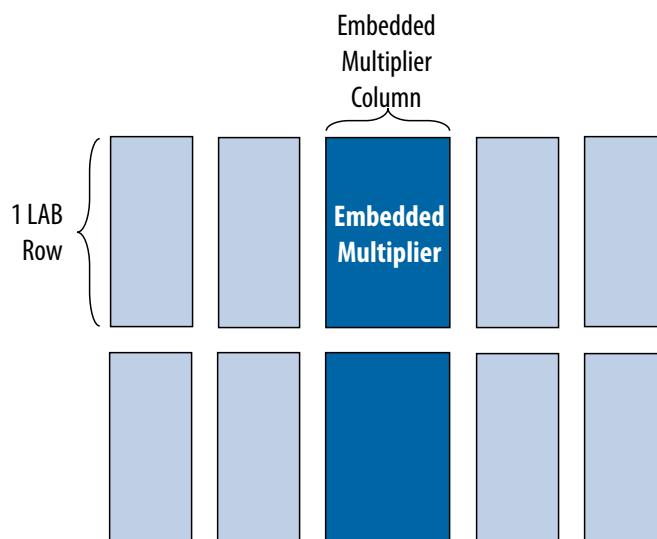
Table 1-1: Number of Embedded Multipliers in the MAX 10 Devices

Device	Embedded Multipliers	9 x 9 Multipliers [1]	18 x 18 Multipliers [1]
10M02	16	32	16
10M04	20	40	20
10M08	24	48	24
10M16	45	90	45

[1] These columns show the number of 9 x 9 or 18 x 18 multipliers for each device. The total number of multipliers for each device is not the sum of all the multipliers.

ALTERA
now part of Intel

Device	Embedded Multipliers	9 x 9 Multipliers [1]	18 x 18 Multipliers [1]
10M25	55	110	55
10M40	125	250	125
10M50	144	288	144

You can implement soft multipliers by using the M9K memory blocks as look-up tables (LUTs). The LUTs contain partial results from multiplying input data with coefficients implementing variable depth and width high-performance soft multipliers for low-cost, high-volume DSP applications. The availability of soft multipliers increases the number of available multipliers in the device.

Table 1-2: Number of Multipliers in the MAX 10 Devices

Device	Embedded Multipliers	Soft Multipliers (16 x 16)[2]	Total Multipliers [3]
10M02	16	12	28
10M04	20	21	41
10M08	24	42	66
10M16	45	61	106
10M25	55	75	130
10M40	125	140	265
10M50	144	182	326

Related Information

MAX 10 Embedded Multipliers User Guide Archives on page 8-1
Provides a list of user guides for previous versions of the LPM_MULT, ALTMULT_ACCUM, ALTMULT_ADD, and ALTMULT_COMPLEX IP cores.

---

[1] These columns show the number of 9 x 9 or 18 x 18 multipliers for each device. The total number of multipliers for each device is not the sum of all the multipliers.

[2] Soft multipliers are implemented in sum of multiplication mode. M9K memory blocks are configured with 18-bit data widths to support 16-bit coefficients. The sum of the coefficients requires 18-bits of resolution to account for overflow.

[3] The total number of multipliers may vary, depending on the multiplier mode you use.
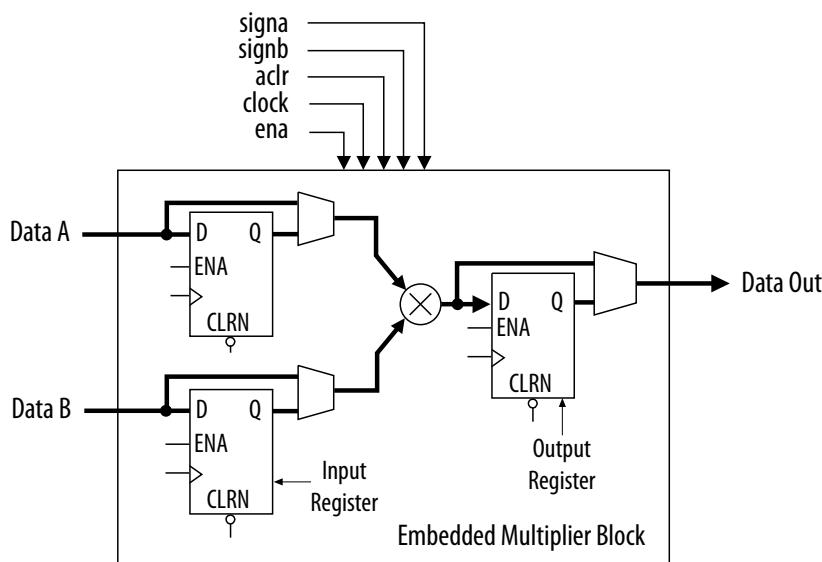
Each embedded multiplier consists of three elements. Depending on the application needs, you can use an embedded multiplier block in one of two operational modes.

Embedded Multipliers Architecture

Each embedded multiplier consists of the following elements:

- Multiplier stage
- Input and output registers
- Input and output interfaces

Figure 2-1: Multiplier Block Architecture

Input Register

Depending on the operational mode of the multiplier, you can send each multiplier input signal into either one of the following:

- An input register
- The multiplier in 9- or 18-bit sections

Each multiplier input signal can be sent through a register independently of other input signals. For example, you can send the multiplier `Data A` signal through a register and send the `Data B` signal directly to the multiplier.

The following control signals are available to each input register in the embedded multiplier:

- Clock
- Clock enable
- Asynchronous clear

All input and output registers in a single embedded multiplier are fed by the same clock, clock enable, and asynchronous clear signals.

Multiplier Stage

The multiplier stage of an embedded multiplier block supports 9 × 9 or 18 × 18 multipliers and other multipliers in between these configurations. Depending on the data width or operational mode of the multiplier, a single embedded multiplier can perform one or two multiplications in parallel.

Each multiplier operand is a unique signed or unsigned number. Two signals, `signa` and `signb`, control an input of a multiplier and determine if the value is signed or unsigned. If the `signa` signal is high, the `Data A` operand is a signed number. If the `signa` signal is low, the `Data A` operand is an unsigned number.

The following table lists the sign of the multiplication results for the various operand sign representations. The results of the multiplication are signed if any one of the operands is a signed value.

Data A		Data B		Result
`signa` Value	Logic Level	`signb` Value	Logic Level	
Unsigned	Low	Unsigned	Low	Unsigned
Unsigned	Low	Signed	High	Signed
Signed	High	Unsigned	Low	Signed
Signed	High	Signed	High	Signed

You can dynamically change the `signa` and `signb` signals to modify the sign representation of the input operands at run time. You can send the `signa` and `signb` signals through a dedicated input register. The multiplier offers full precision, regardless of the sign representation.

When the `signa` and `signb` signals are unused, the Quartus Prime software sets the multiplier to perform unsigned multiplication by default.

Output Register

You can register the embedded multiplier output using output registers in either 18- or 36-bit sections. This depends on the operational mode of the multiplier. The following control signals are available for each output register in the embedded multiplier:

- Clock
- Clock enable
- Asynchronous clear

All input and output registers in a single embedded multiplier are fed by the same clock, clock enable, and asynchronous clear signals.

Embedded Multipliers Operational Modes

You can use an embedded multiplier block in one of two operational modes, depending on the application needs:

- One 18-bit x 18-bit multiplier
- Up to two 9-bit x 9-bit independent multipliers
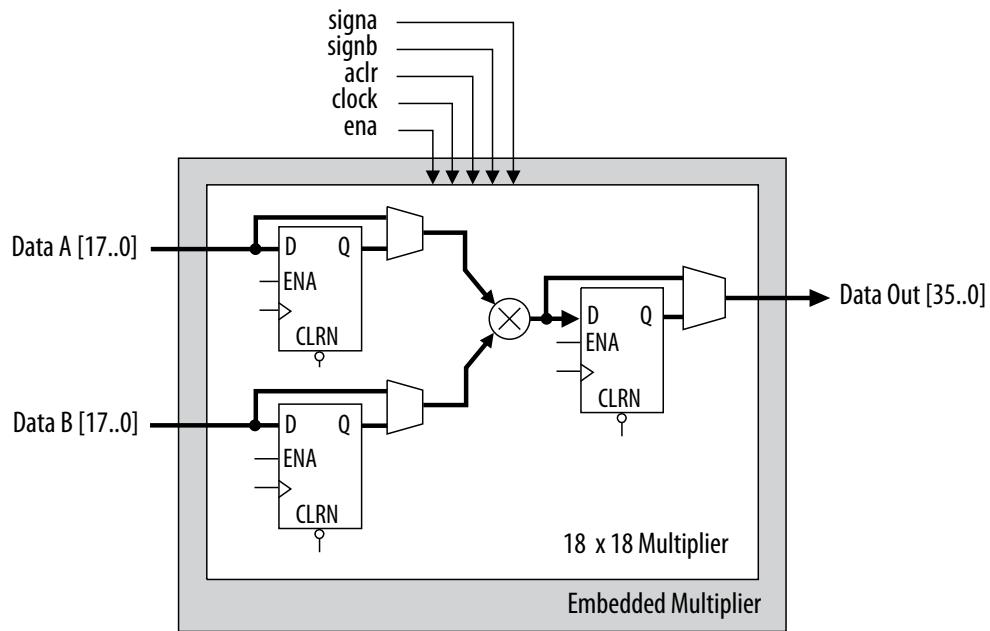
You can also use embedded multipliers of the MAX® 10 devices to implement multiplier adder and multiplier accumulator functions. The multiplier portion of the function is implemented using embedded multipliers. The adder or accumulator function is implemented in logic elements (LEs).

18-Bit Multipliers

You can configure each embedded multiplier to support a single 18 x 18 multiplier for input widths of 10 to 18 bits.

The following figure shows the embedded multiplier configured to support an 18-bit multiplier.

Figure 2-2: 18-Bit Multiplier Mode
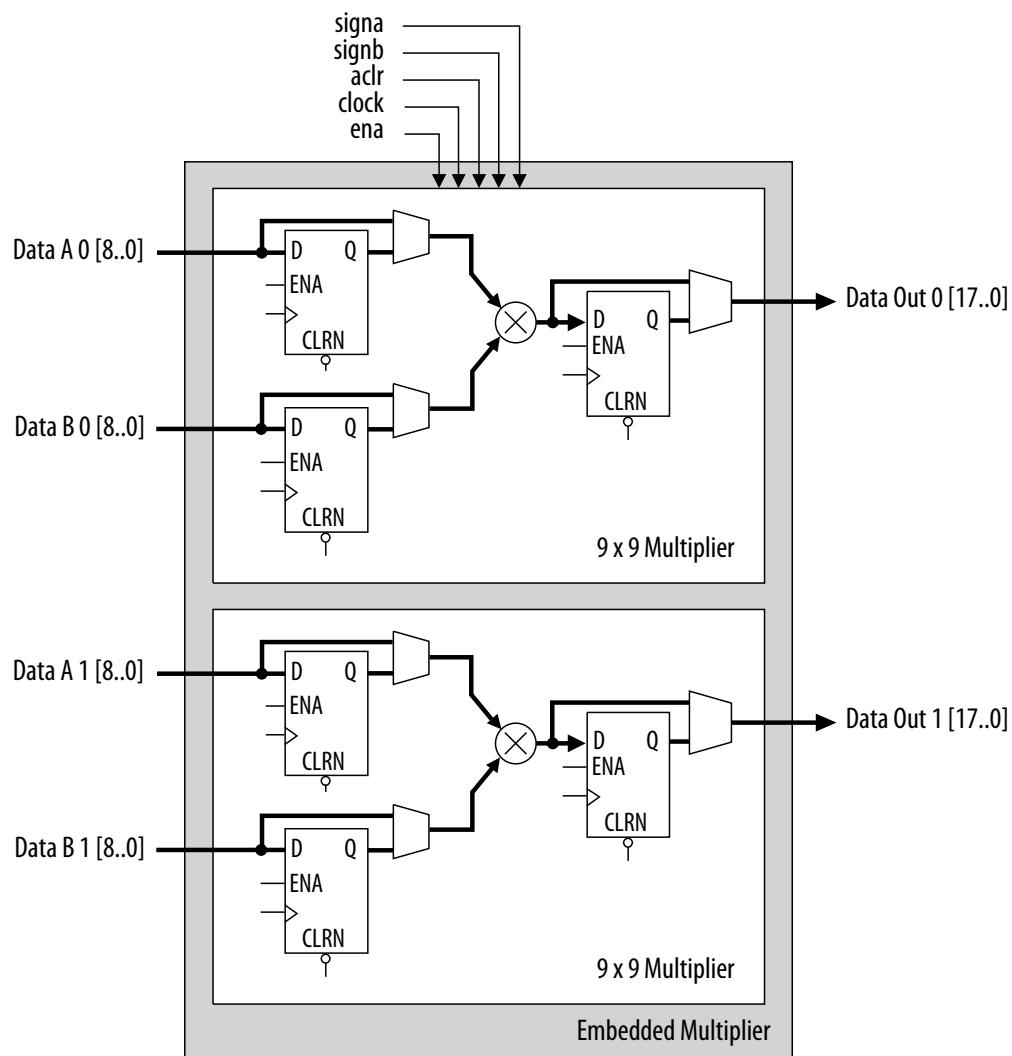
All 18-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both. Also, you can dynamically change the `signa` and `signb` signals and send these signals through dedicated input registers.

9-Bit Multipliers

You can configure each embedded multiplier to support two 9 × 9 independent multipliers for input widths of up to 9 bits.

The following figure shows the embedded multiplier configured to support two 9-bit multipliers.

Figure 2-3: 9-Bit Multiplier Mode

All 9-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both.

Each embedded multiplier block has only one `signa` and one `signb` signal to control the sign representation of the input data to the block. If the embedded multiplier block has two 9 × 9 multipliers the following applies:

- The `Data A` input of both multipliers share the same `signa` signal
- The `Data B` input of both multipliers share the same `signb` signal

The Quartus Prime software contains tools for you to create and compile your design, and configure your device.

You can prepare for device migration, set pin assignments, define placement restrictions, setup timing constraints, and customize IP cores using the Quartus Prime software.

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Files Generated by IP Cores

The following integer arithmetic IP cores use the MAX 10 device embedded multipliers block:

- LPM_MULT
- ALTMULT_ACCUM (MAC)
- ALTMULT_ADD
- ALTMULT_COMPLEX

Verilog HDL Prototype Location

You can view the Verilog HDL prototype for the IP cores in the following Verilog Design Files (**.v**):

Table 3-1: Verilog HDL Prototype Location

Integer Arithmetic Megafunctions	Directory	Verilog Design File (.v)
LPM_MULT	**<Quartus Prime installation directory>\eda\synthesis**	**lpm.v**

ISO 9001:2008 Registered

now part of Intel

Integer Arithmetic Megafunctions	Directory	Verilog Design File (.v)
• ALTMULT_ACCUM • ALTMULT_ADD • ALTMULT_COMPLEX	**\<Quartus Prime installation directory>\eda\synthesis**	**altera_mf.v**

VHDL Component Declaration Location

You can view the VHDL component declaration for the IP cores in the following VHDL Design Files (**.vhd**):

Integer Arithmetic Megafunctions	Directory	VHDL Design File (.vhd)
LPM_MULT	**\<Quartus Prime installation directory>\libraries\vhdl\lpm**	**LPM_PACK.vhd**
• ALTMULT_ACCUM • ALTMULT_ADD • ALTMULT_COMPLEX	**\<Quartus Prime installation directory>\libraries\vhdl\altera_mf**	**altera_mf_components.vhd**

LPM_MULT Parameter Settings

There are three groups of options: **General**, **General2**, and **Pipeling**.

Table 4-1: LPM_MULT Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Multiplier configuration	—	—	• Multiply 'dataa' input by 'datab' input • Multiply 'dataa' input by itself (squaring operation)	Specifies the multiplier configuration.
How wide should the 'dataa' input be?	LPM_WIDTHA	—	1–256	Specifies the width of the dataa[] port.
How wide should the 'datab' input be?	LPM_WIDTHB	—	1–256	Specifies the width of the datab[] port.
How should the width of the 'result' output be determined?	LPM_WIDTHP	—	• Automatically calculate the width • Restrict the width to [] bits	Specifies how the result width is determined.
How should the width of the 'result' output be determined? > Restrict the width to [] bits	LPM_WIDTHP	How should the width of the 'result' output be determined? > Restrict the width to [] bits = On	1–256	You can set the result width.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Table 4-2: LPM_MULT Parameters - General2

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Does the 'datab' input bus have a constant value?	—	—	• No • Yes, the value is []	You can specify the constant value of the 'datab' input bus, if any.
Which type of multiplication do you want?	LPM_REPRESENTATION	—	• Unsigned • Signed	Specifies the type of multiplication performed.
Which multiplier implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	• Use default implementation • Use the dedicated multiplier circuitry (Not available for all families) • Use logic elements	Specifies the multiplier implementation.

Table 4-3: LPM_MULT Parameters - Pipeling

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Do you want to pipeline the function?	LPM_PIPELINE	—	• No • Yes, I want output latency of [] clock cycles	You can add extra latency to the outputs, if any.
Create an 'aclr' asynchronous clear port	—	Do you want to pipeline the function? = Yes, I want output latency of [] clock cycles	On or off	Specifies asynchronous clear for the complex multiplier. Clears the function asynchronously when aclr port is asserted high.
Create a 'clken' clock enable clock	—	Do you want to pipeline the function? = Yes, I want output latency of [] clock cycles	On or off	Specifies active high clock enable for the clock port of the complex multiplier
What type of optimization do you want?	MAXIMIZE_SPEED	—	• Default • Speed • Area	You can specify if the type of optimization is determined by Quartus Prime, speed, or area.

Ports

Table 4-4: LPM_MULT Input Ports

Port Name	Required	Description
dataa[]	Yes	Data input. The size of the input port depends on the LPM_WIDTHA parameter value.
datab[]	Yes	Data input. The size of the input port depends on the LPM_WIDTHB parameter value.
clock	No	Clock input for pipelined usage. For LPM_PIPELINE values other than 0 (default), the clock port must be enabled.
clken	No	Clock enable for pipelined usage. When the clken port is asserted high, the adder/subtractor operation takes place. When the signal is low, no operation occurs. If omitted, the default value is 1.
aclr	No	Asynchronous clear port used at any time to reset the pipeline to all 0s, asynchronously to the clock signal. The pipeline initializes to an undefined (X) logic level. The outputs are a consistent, but non-zero value.

Table 4-5: LPM_MULT Output Ports

Port Name	Required	Description
result[]	Yes	Data output. The size of the output port depends on the LPM_WIDTHP parameter value. If LPM_WIDTHP < max (LPM_WIDTHA + LPM_WIDTHB, LPM_WIDTHS) or (LPM_WIDTHA + LPM_WIDTHS), only the LPM_WIDTHP MSBs are present.

ALTMULT_ACCUM Parameter Settings

There are four groups of options: **General**, **Extra Modes**, **Multipliers**, and **Accumulator**.

Table 5-1: ALTMULT_ACCUM Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
What is the number of multipliers?	NUMBER_OF_MULTIPLIERS	—	1	By default, only 1 multiplier is supported.
All multipliers have similar configurations	—	—	On	By default all multipliers have similar configurations
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
Create a 4th asynchronous clear input option	—	—	On or Off	Turn on this option if you want to create a 4th asynchronous clear input option.
Create an associated clock enable for each clock	—	—	On or Off	Turn on this option if you want to create an associated clock enable for each clock.
What is the representation format for A inputs?	REPRESENTATION_A	—	• Signed • Unsigned • Variable	Specifies the representation format for A inputs.

ISO 9001:2008 Registered

ALTERA
now part of Intel

GUI Parameter	Parameter	Condition	Value	Description
'signa' input controls the sign (1 signed/0 unsigned)	PORT_SIGNA	Input Representation > What is the representation format for A inputs? = Variable	More Options	High 'signa' input indicates signed and low 'signa' input indicates unsigned.
Register 'signa' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signa' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGN_REG_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGN_ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGN_ PIPELINE_ REG_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGN_ PIPELINE_ ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	• Signed • Unsigned • Variable	Specifies the representation format for B inputs.
'signb' input controls the sign (1 signed/0 unsigned)	PORT_SIGNB	Input Representation > What is the representation format for B inputs? = Variable	More Options	High 'signb' input indicates signed and low 'signb' input indicates unsigned.
Register 'signb' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signb' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register

GUI Parameter	Parameter	Condition	Value	Description
Input Register > What is the source for clock input?	SIGN_REG_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGN_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGN_PIPELINE_REG_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGN_PIPELINE_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

Table 5-2: ALTMULT_ACCUM Parameters - Extra Modes

GUI Parameter	Parameter	Condition	Value	Description
Create a shiftout output from A input of the last multiplier	—	—	On or Off	Turn on this option to create a shiftout output from A input of the last multiplier.
Create a shiftout output from B input of the last multiplier	—	—	On or Off	Turn on this option to create a shiftout output from B input of the last multiplier.
Add extra register(s) at the output	—	—	On	By default, output register must be enabled for accumulator.
What is the source for clock input?	OUTPUT_REG	Outputs Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the registers on the outputs.
What is the source for asynchronous clear input?	OUTPUT_ACLR	Outputs Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear signal for the registers on the outputs.
Add [] extra latency to the output	—	Outputs Configuration > More Options	0, 1, 2, 3, 4, 5, 6, 7, 8, or 12	Specifies the extra latency to add to the output.

GUI Parameter	Parameter	Condition	Value	Description
Which multiplier-adder implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	• Use the default implementation • Use dedicated multiplier circuitry (Not available for all families) • Use logic elements	Specifies the multiplier-adder implementation.

Table 5-3: ALTMULT_ACCUM Parameters - Multipliers

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Register input A of the multiplier	—	—	On or Off	Turn on to enable register input A of the multiplier.
What is the source for clock input?	INPUT_REG_A	• Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options	Clock0–Clock3	Specifies the clock port for the `dataa[]` port.
What is the source for asynchronous clear input?	INPUT_ACLR_A	• Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear port for the `dataa[]` port.
Register input B of the multiplier	—	—	On or Off	Turn on to enable register input B of the multiplier.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for clock input?	INPUT_REG_B	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	Clock0–Clock3	Specifies the clock port for the `datab[]` port.
What is the source for asynchronous clear input?	INPUT_ACLR_B	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear port for the `datab[]` port.
What is the input A of the multiplier connected to?	—	—	Multiplier input	By default, input A of the multiplier is always connected to the multiplier's input.
What is the input B of the multiplier connected to?	—	—	Multiplier input	By default, input B of the multiplier is always connected to the multiplier's input.
Register output of the multiplier	—	—	On or Off	Turn on to enable register output of the multiplier.
What is the source for clock input?	MULTIPLIER_REG	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the register that immediately follows the multiplier.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for asynchronous clear input?	MULTIPLIER_ ACLR	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear signal of the register that follows the corresponding multiplier.

Table 5-4: ALTMULT_ACCUM Parameters - Accumulator

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Create an 'accum_ sload' input port	—	—	On or off	Dynamically specifies whether the accumulator value is constant. If the accum_ sload port is high, then the multiplier output is loaded into the accumulator.
Register 'accum_sload' input	—	• Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	On or off	Turn on to enable register 'accum_sload' input.
Add an extra pipeline register	—	• Accumulator > Create an 'accum_ sload' input port = On • Accumulator > More Options	On or off	Turn on this option if you want to enable the extra pipeline register

GUI Parameter	Parameter	Condition	Value	Description
Input Register > What is the source for clock input?	ACCUM_ SLOAD_REG	• Accumulator > Create an 'accum_sload' input port = On • Accumulator > More Options	Clock0–Clock3	Specifies the clock signal for the `accum_ sload` port.
Input Register > What is the source for asynchronous clear input?	ACCUM_ SLOAD_ACLR	• Accumulator > Create an '`accum_ sload`' input port = On • Accumulator > More Options	• Aclr0–Aclr2 • None	Specifies the asynchronous clear source for the first register on the `accum_sload` input.
Pipeline Register > What is the source for clock input?	ACCUM_ SLOAD_ PIPELINE_REG	• • Accumulator > Create an '`accum_ sload`' input port = On • Accumulator > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ACCUM_ SLOAD_ PIPELINE_ ACLR	• Accumulator > Create an '`accum_soad`' input port = On • Accumulator > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Create an '`overflow`' output port	—	—	On or Off	`Overflow` port for the accumulator
Add [] extra latency to the multiplier output	EXTRA_ MULTIPLIER_ LATENCY	—	0, 1, 2, 3, 4, 5, 6, 7, 8, or 12	Specifies the number of clock cycles of latency for the multiplier portion of the DSP block. If the MULTIPLIER_REG parameter is specified, then the specified clock port is used to add the latency.

ALTMULT_ACCUM Ports

Table 5-5: ALTMULT_ACCUM IP Core Input Ports

Port Name	Required	Description
accum_sload	No	Causes the value on the accumulator feedback path to go to zero (0) or to `accum_sload_upper_data` when concatenated with 0. If the accumulator is adding and the `accum_sload` port is high, then the multiplier output is loaded into the accumulator. If the accumulator is subtracting, then the opposite (negative value) of the multiplier output is loaded into the accumulator.
aclr0	No	The first asynchronous clear input. The `aclr0` port is active high.
aclr1	No	The second asynchronous clear input. The `aclr1` port is active high.
aclr2	No	The third asynchronous clear input. The `aclr2` port is active high.
aclr3	No	The fourth asynchronous clear input. The `aclr3` port is active high.
addnsub	No	Controls the functionality of the adder. If the `addnsub` port is high, the adder performs an add function; if the `addnsub` port is low, the adder performs a subtract function.
clock0	No	Specifies the first clock input, usable by any register in the IP core.
clock1	No	Specifies the second clock input, usable by any register in the IP core.
clock2	No	Specifies the third clock input, usable by any register in the IP core.
clock3	No	Specifies the fourth clock input, usable by any register in the IP core.
dataa[]	Yes	Data input to the multiplier. The size of the input port depends on the `WIDTH_A` parameter value.
datab[]	Yes	Data input to the multiplier. The size of the input port depends on the `WIDTH_B` parameter value.
ena0	No	Clock enable for the `clock0` port.
ena1	No	Clock enable for the `clock1` port.
ena2	No	Clock enable for the `clock2` port.
ena3	No	Clock enable for the `clock3` port.

Port Name	Required	Description
signa	No	Specifies the numerical representation of the dataa[] port. If the signa port is high, the multiplier treats the dataa[] port as signed two's complement. If the signa port is low, the multiplier treats the dataa[] port as an unsigned number.
signb	No	Specifies the numerical representation of the datab[] port. If the signb port is high, the multiplier treats the datab[] port as signed two's complement. If the signb port is low, the multiplier treats the datab[]port as an unsigned number.

Table 5-6: ALTMULT_ACCUM IP Core Output Ports

Port Name	Required	Description
overflow	No	Overflow port for the accumulator.
result[]	Yes	Accumulator output port. The size of the output port depends on the WIDTH_RESULT parameter value.
scanouta[]	No	Output of the first shift register. The size of the output port depends on the WIDTH_A parameter value. When instantiating the ALTMULT_ACCUM IP core with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanouta[] port to shiftouta port.
scanoutb[]	No	Output of the second shift register. The size of the input port depends on the WIDTH_B parameter value. When instantiating the ALTMULT_ACCUM IP core with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanoutb[] port to shiftoutb port.

2016.05.02

✉ Subscribe   💬 Send Feedback

ALTMULT_ADD Parameter Settings

There are three groups of options: **General**, **Extra Modes**, and **Multipliers**.

Table 6-1: ALTMULT_ADD Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
What is the number of multipliers?	NUMBER_OF_MULTIPLIERS	—	1, 2, 3, or 4	Specifies the number of multipliers. You can specify up to four multipliers.
All multipliers have similar configurations	—	—	On or Off	Turn on this option if you want all multipliers to have similar configurations.
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
Create a 4th asynchronous clear input option	—	—	On or Off	Turn on this option if you want to create a 4th asynchronous clear input option.
Create an associated clock enable for each clock	—	—	On or Off	Turn on this option if you want to create an associated clock enable for each clock.

**ISO
9001:2008
Registered**

now part of Intel

GUI Parameter	Parameter	Condition	Value	Description
What is the representation format for A inputs?	REPRESENTATION_A	—	• Signed • Unsigned • Variable	Specifies the representation format for A inputs.
'signa' input controls the sign (1 signed/0 unsigned)	PORT_SIGNA	Input Representation > What is the representation format for A inputs? = Variable	More Options	High 'signa' input indicates signed and low 'signa' input indicates unsigned.
Register 'signa' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signa' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGNED_REGISTER_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGNED_ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGNED_PIPELINE_REGISTER_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGNED_PIPELINE_ACLR_A	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	• Signed • Unsigned • Variable	Specifies the representation format for B inputs.
'signb' input controls the sign (1 signed/0 unsigned)	PORT_SIGNB	Input Representation > What is the representation format for B inputs? = Variable	More Options	High 'signb' input indicates signed and low 'signb' input indicates unsigned.

GUI Parameter	Parameter	Condition	Value	Description
Register 'signb' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signb' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGNED_REGISTER_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGNED_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGNED_PIPELINE_REGISTER_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGNED_PIPELINE_ACLR_B	Input Representation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

Table 6-2: ALTMULT_ADD Parameters - Extra Modes

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Create a shiftout output from A input of the last multiplier	—	—	On or Off	Turn on to create a signal from A input.
Create a shiftout output from B input of the last multiplier	—	—	On or Off	Turn on to create a signal from B input.
Register output of the adder unit	—	—	On or Off	Turn on to create a register output of the adder unit.
What is the source for clock input?	OUTPUT_REGISTER	• Outputs Configuration > Register output of the adder unit = On • Outputs Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the output register.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for asynchronous clear input?	OUTPUT_ACLR	• Outputs Configuration > Register output of the adder unit = On • Outputs Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What operation should be performed on outputs of the first pair of multipliers?	MUTIPLIER1_DIRECTION	General > What is the number of multipliers? = 2, 3, or 4	• Add • Subtract • Variable	Specifies whether the second multiplier adds or subtracts its value from the sum. Values are add and subtract. If Variable is selected the `addnsub1` port is used.
'addnsub1' input controls the operation (1 add/0 sub)	—	Adder Operation > What operation should be performed on outputs of the first pair of multipliers? = Variable	More Options	High 'addnsub1' input indicates add and low 'addnsub1' input indicates subtract.
Register 'addnsub1' input	—	—	On or Off	Turn on this option if you want to enable the register of 'addnsub1' input
Add an extra pipeline register	—	—	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_REGISTER[1]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	ADDSUB_MULTIPLIER_ACLR[1]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[1]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.

GUI Parameter	Parameter	Condition	Value	Description
Pipeline Register > What is the source for asynchronous clear input?	ADDNSUB_ MULTIPLIER_ PIPELINE_ ACLR[1]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What operation should be performed on outputs of the second pair of multipliers?	MUTIPLIER3_ DIRECTION	General > What is the number of multipliers? = 4	—	Specifies whether the fourth and all subsequent odd-numbered multipliers add or subtract their results from the total. Values are add and subtract. If Variable is selected, the `addnsub3` port is used.
'addnsub3' input controls the sign (1 add/0 sub) - More Options	—	—	—	High 'addnsub3' input indicates add and low 'addnsub3' input indicates subtract.
Register 'addnsub3' input	—	—	On or Off	Turn on this option if you want to enable the register of 'addnsub3' input.
Add an extra pipeline register	—	—	On or Off	Turn on this option if you want to enable the extra pipeline register.
Input Register > What is the source for clock input?	ADDNSUB_ MULTIPLIER_ REGISTER[3]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	ADDSUB_ MULTIPLIER_ ACLR[3]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	ADDNSUB_ MULTIPLIER_ PIPELINE_ REGISTER[3]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ADDNSUB_ MULTIPLIER_ PIPELINE_ ACLR[3]	Adder Operation > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

GUI Parameter	Parameter	Condition	Value	Description
Which multiplier-adder implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	• Use the default implementation • Use dedicated multiplier circuitry (Not available for all families) • Use logic elements	Specifies the multiplier-adder implementation.

Table 6-3: ALTMULT_ADD Parameters - Multipliers

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Register input A of the multiplier	—	—	On or Off	Turn on to enable register input A of the multiplier.
What is the source for clock input?	INPUT_REGISTER_A[0..3]	• Input Configuration > Register input A of the multiplier = On • • Input Configuration > More Options	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	INPUT_ACLR_A[0..3]	• Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
Register input B of the multiplier	—	—	On or Off	Turn on to enable register input B of the multiplier.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for clock input?	INPUT_ REGISTER_ B[0..3]	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	INPUT_ACLR_ B[0..3]	• Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.
What is the input A of the multiplier connected to?	INPUT_ SOURCE_ A[0..3]	—	• Multiplier input • Shiftin input	Specifies the input A of the multiplier is connected to either multiplier input or shiftin input.
What is the input B of the multiplier connected to?	INPUT_ SOURCE_B[0..3]	—	• Multiplier input • Shiftin input	Specifies the input B of the multiplier is connected to either multiplier input or shiftin input.
Register output of the multiplier	—	—	On or Off	Turn on to enable the register for output of the multiplier.
What is the source for clock input?	MULTIPLIER_ REGISTER[]	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	Clock0–Clock3	Specifies the source for clock input.

GUI Parameter	Parameter	Condition	Value	Description
What is the source for asynchronous clear input?	MULTIPLIER_ACLR[]	• Output Configuration > Register output of the multiplier = On • Output Configuration > More Options	• Aclr0–Aclr2 • None	Specifies the source for asynchronous clear input.

ALTMULT_ADD Ports

Table 6-4: ALTMULT_ADD IP Core Input Ports

Port Name	Required	Description
dataa[]	Yes	Data input to the multiplier. Input port `[NUMBER_OF_MULTIPLIERS * WIDTH_A - 1..0]` wide.
datab[]	Yes	Data input to the multiplier. Input port `[NUMBER_OF_MULTIPLIERS * WIDTH_B - 1..0]` wide.
clock[]	No	Clock input port `[0..3]` to the corresponding register. This port can be used by any register in the IP core.
aclr[]	No	Input port `[0..3]`. Asynchronous clear input to the corresponding register.
ena[]	No	Input port `[0..3]`. Clock enable for the corresponding clock[] port.
signa	No	Specifies the numerical representation of the `dataa[]` port. If the `signa` port is high, the multiplier treats the `dataa[]` port as a signed two's complement number. If the `signa` port is low, the multiplier treats the `dataa[]` port as an unsigned number.
signb	No	Specifies the numerical representation of the `datab[]` port. If the `signb` port is high, the multiplier treats the `datab[]` port as a signed two's complement number. If the `signb` port is low, the multiplier treats the `datab[]` port as an unsigned number.

Table 6-5: ALTMULT_ADD IP Core Output Ports

Port Name	Required	Description
result[]	Yes	Multiplier output port. Output port `[WIDTH_RESULT - 1..0]` wide.
overflow	No	Overflow flag. If `output_saturation` is enabled, overflow flag is set.
scanouta[]	No	Output of scan chain A. Output port `[WIDTH_A - 1..0]` wide.
scanoutb[]	No	Output of scan chain B. Output port `[WIDTH_B - 1..0]` wide.

ALTMULT_COMPLEX Parameter Settings

There are two groups of options: **General** and **Implementation Style/Pipelining**.

Table 7-1: ALTMULT_COMPLEX Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
What is the representation format for A inputs?	REPRESENTATION_A	—	• Signed • Unsigned	Specifies the representation format for A inputs.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	• Signed • Unsigned	Specifies the representation format for B inputs.

ISO 9001:2008 Registered

now part of Intel

Table 7-2: ALTMULT_COMPLEX Parameters - Implementation Style/Pipelining

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Which implementation style should be used?	IMPLEMENTATION_STYLE	—	Automatically select a style for best trade-off for the current settings	By default automatic selection for MAX 10 device is selected. Quartus Prime software will determine the best implementation based on the selected device family and input width.
Output latency [] clock cycles	PIPELINE	—	0–14	Specifies the number of clock cycles for output latency.
Create an asynchronous Clear input	—	—	On or off	Specifies synchronous clear for the complex multiplier. Clears the function asynchronously when the `aclr` port is asserted high.
Create clock enable input	—	—	On or off	Specifies active high clock enable for the clock port of the complex multiplier.

Ports

Table 7-3: ALTMULT_COMPLEX Input Ports

Port Name	Required	Description
`aclr`	No	Asynchronous clear for the complex multiplier. When the `aclr` port is asserted high, the function is asynchronously cleared.
`clock`	Yes	Clock input to the ALTMULT_COMPLEX function.
`dataa_imag[]`	Yes	Imaginary input value for the data `A` port of the complex multiplier. The size of the input port depends on the `WIDTH_A` parameter value.
`dataa_real[]`	Yes	Real input value for the data `A` port of the complex multiplier. The size of the input port depends on the `WIDTH_A` parameter value.
`datab_imag[]`	Yes	Imaginary input value for the data `B` port of the complex multiplier. The size of the input port depends on the `WIDTH_B` parameter value.

Port Name	Required	Description
datab_real[]	Yes	Real input value for the data B port of the complex multiplier. The size of the input port depends on the WIDTH_B parameter value.
ena	No	Active high clock enable for the clock port of the complex multiplier.

Table 7-4: ALTMULT_COMPLEX Output Ports

Port Name	Required	Description
result_imag	Yes	Imaginary output value of the multiplier. The size of the output port depends on the WIDTH_RESULT parameter value.
result_real	Yes	Real output value of the multiplier. The size of the output port depends on the WIDTH_RESULT parameter value.

MAX 10 Embedded Multipliers User Guide Archives

A

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 Embedded Multipliers User Guide**
14.1	**MAX 10 Embedded Multipliers User Guide**

ALTERA
now part of Intel

Document Revision History for MAX 10 Embedded Multipliers User Guide

Date	Version	Changes
May 2016	2016.05.02	• Updated MAX 10 to each chapter in the user guide. • Added MAX 10 Embedded Multipliers User Guide Archives chapter.
November 2015	2015.11.02	• Changed instances of *Quartus II* to *Quartus Prime*. • Removed topics on generating IP cores and added links to Introduction to Altera IP Cores, Creating Version-Independent IP and Qsys Simulation Scripts, and Project Management Best Practices.
September 2014	2014.09.22	Initial release.

**ISO
9001:2008
Registered**

now part of Intel

MAX 10 Clocking and PLL User Guide

Contents

Clock Networks Overview

MAX® 10 devices support global clock (GCLK) networks.

Clock networks provide clock sources for the core. You can use clock networks in high fan-out global signal network such as reset and clear.

Internal Oscillator Overview

MAX 10 devices offer built-in internal oscillator up to 116 MHz.

You can enable or disable the internal oscillator.

PLLs Overview

Phase-locked loops (PLLs) provide robust clock management and synthesis for device clock management, external system clock management, and I/O interface clocking.

You can use the PLLs as follows:

- Zero-delay buffer
- Jitter attenuator
- Low-skew fan-out buffer
- Frequency synthesizer
- Reduce the number of oscillators required on the board
- Reduce the clock pins used in the device by synthesizing multiple clock frequencies from a single reference clock source
- On-chip clock de-skew
- Dynamic phase shift
- Counters reconfiguration
- Bandwidth reconfiguration
- Programmable output duty cycle

ISO 9001:2008 Registered

ALTERA®

- PLL cascading
- Reference clock switchover
- Drive the analog-to-digital converter (ADC) clock

Clock Networks Architecture and Features

Global Clock Networks

GCLKs drive throughout the entire device, feeding all device quadrants. All resources in the device, such as the I/O elements, logic array blocks (LABs), dedicated multiplier blocks, and M9K memory blocks can use GCLKs as clock sources. Use these clock network resources for control signals, such as clock enables and clears fed by an external pin. Internal logic can also drive GCLKs for internally-generated GCLKs and asynchronous clears, clock enables, or other control signals with high fan-out.

Clock Pins Introduction

There are two types of external clock pins that can drive the GCLK networks.

Dedicated Clock Input Pins

You can use the dedicated clock input pins (`CLK<#>[p,n]`) to drive clock and global signals, such as asynchronous clears, presets, and clock enables for GCLK networks.

If you do not use the dedicated clock input pins for clock input, you can also use them as general-purpose input or output pins.

The `CLK` pins can be single-ended or differential inputs. When you use the `CLK` pins as single-ended clock inputs, both the `CLK<#>p` and `CLK<#>n` pins have dedicated connection to the GCLK networks. When you use the `CLK` pins as differential inputs, pair two clock pins of the same number to receive differential signaling.

Dual-Purpose Clock Pins

You can use the dual-purpose clock (`DPCLK`) pins for high fan-out control signals, such as protocol signals, `TRDY` and `IRDY` signals for PCI via GCLK networks.

The `DPCLK` pins are only available on the left and right of the I/O banks.

Clock Resources

Table 2-1: MAX 10 Clock Resources

Clock Resource	Device	Number of Resources Available	Source of Clock Resource
Dedicated clock input pins	• 10M02 • 10M04 • 10M08	8 single-ended or 4 differential	`CLK[3..0][p,n]` pins on the left and right of the I/O banks
	• 10M16 • 10M25 • 10M40 • 10M50	16 single-ended or 8 differential	`CLK[7..0][p,n]` pins on the top, left, bottom, and right of the I/O banks
`DPCLK` pins	All	4	`DPCLK[3..0]` pins on the left and right of the I/O banks

For more information about the clock input pins connections, refer to the pin connection guidelines.

Related Information

MAX 10 FPGA Device Family Pin Connection Guidelines

Global Clock Network Sources

Table 2-2: MAX 10 Clock Pins Connectivity to the GCLK Networks

`CLK` Pin	GCLK
CLK0p	`GCLK[0,2,4]`
CLK0n	`GCLK[1,2]`
CLK1p	`GCLK[1,3,4]`
CLK1n	`GCLK[0,3]`
CLK2p	`GCLK[5,7,9]`
CLK2n	`GCLK[6,7]`
CLK3p	`GCLK[6,8,9]`
CLK3n	`GCLK[5,8]`
CLK4p[1]	`GCLK[10,12,14]`
CLK4n[1]	`GCLK[11,12]`
CLK5p[1]	`GCLK[11,13,14]`
CLK5n[1]	`GCLK[10,13]`
CLK6p[1]	`GCLK[15,17,19]`

---

[1] This only applies to 10M16, 10M25, 10M40, and 10M50 devices.

CLK Pin	GCLK
CLK6n[1]	GCLK[16,17]
CLK7p[1]	GCLK[16,18,19]
CLK7n[1]	GCLK[15,18]
DPCLK0	GCLK[0,2]
DPCLK1	GCLK[1,3,4]
DPCLK2	GCLK[5,7]
DPCLK3	GCLK[6,8,9]

Figure 2-1: GCLK Network Sources for 10M02, 10M04, and 10M08 Devices
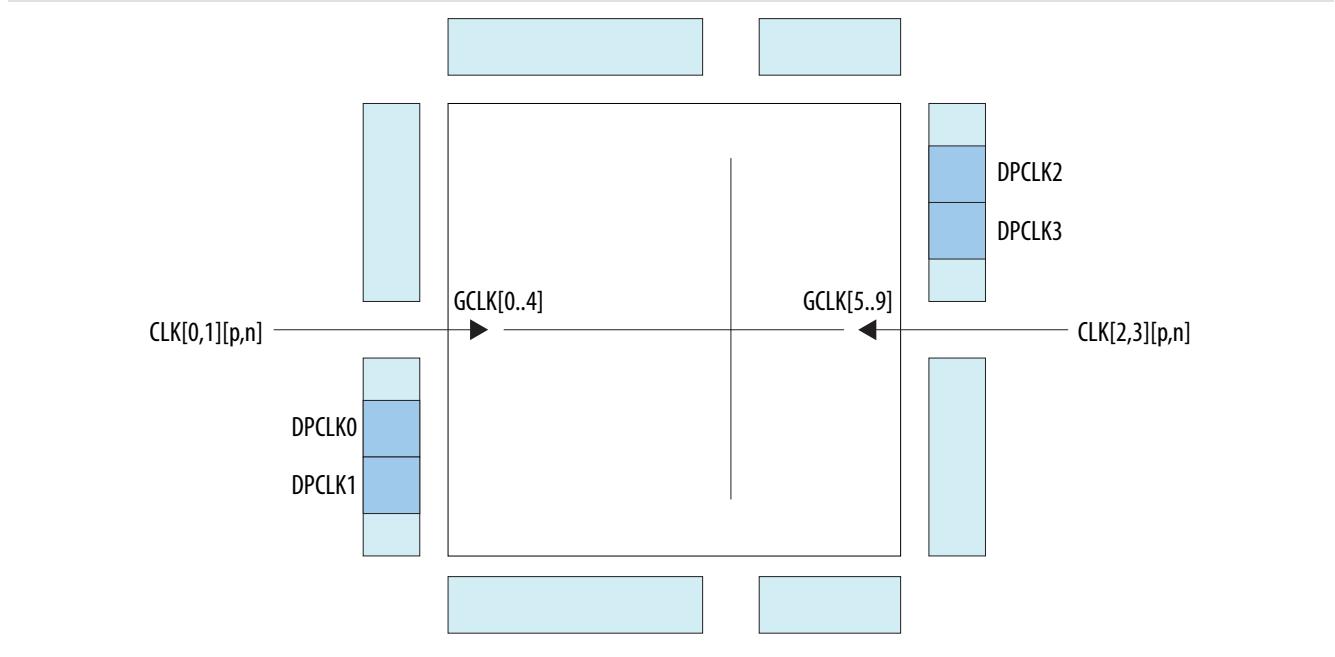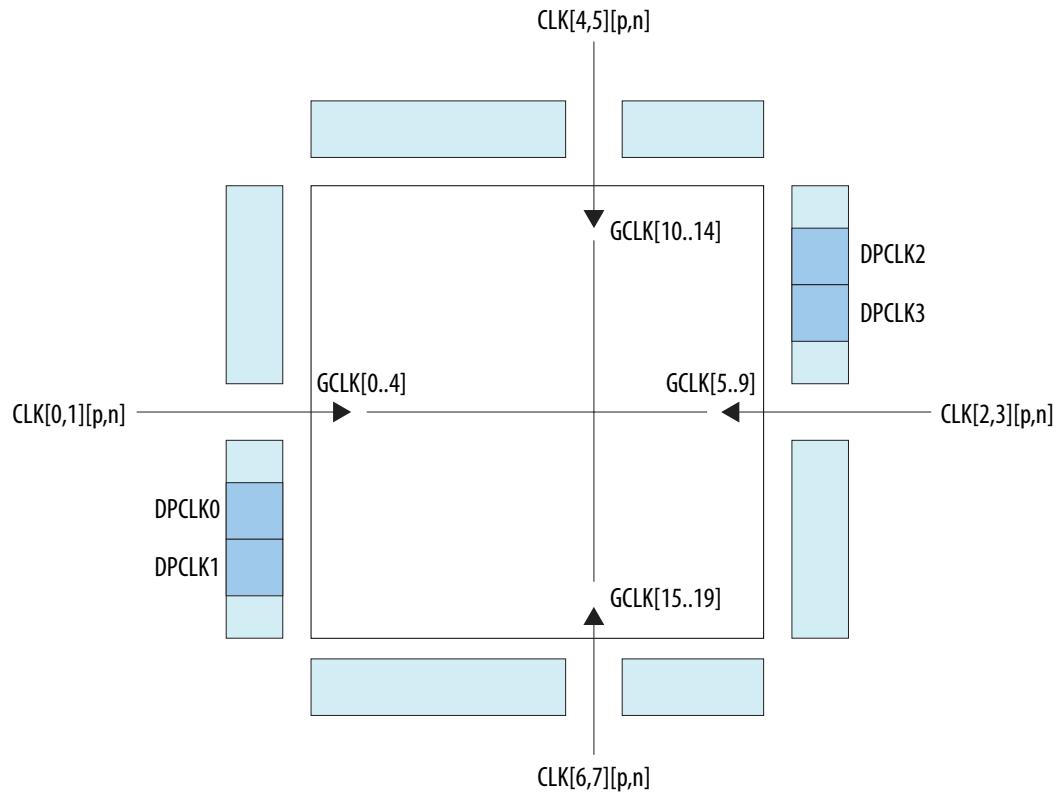
Figure 2-2: GCLK Network Sources for 10M16, 10M25, 10M40, and 10M50 Devices

Global Clock Control Block

The clock control block drives GCLKs. The clock control blocks are located on each side of the device, close to the dedicated clock input pins. GCLKs are optimized for minimum clock skew and delay.
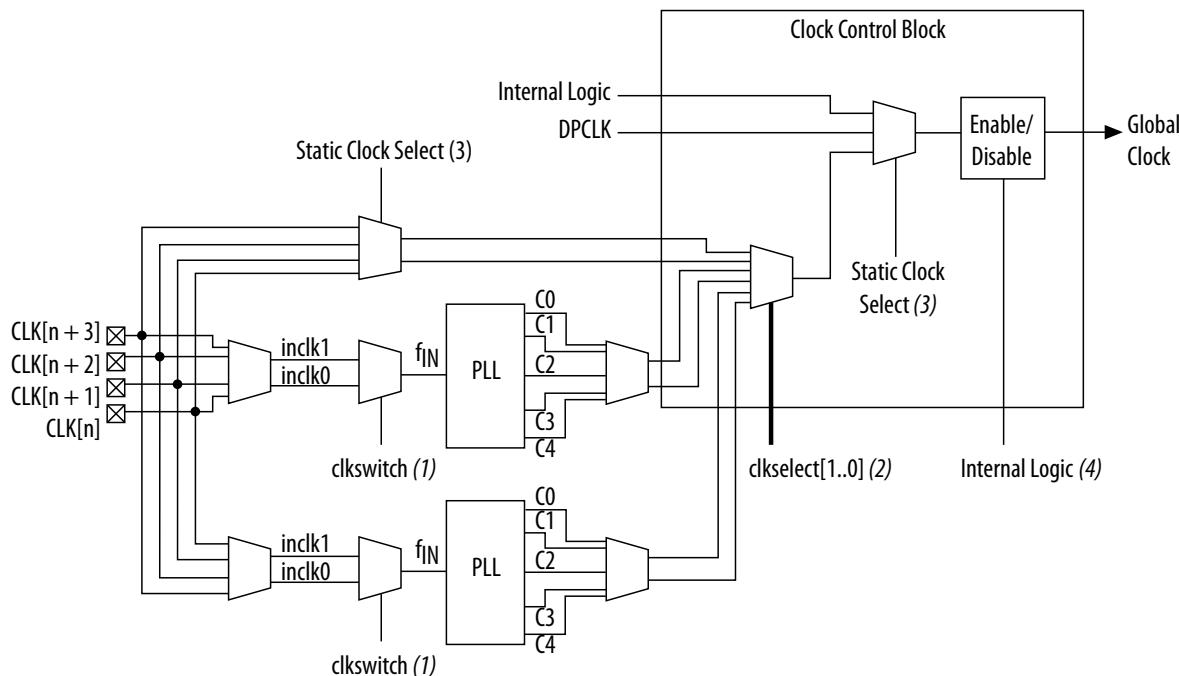
The clock control block has the following functions:

- Dynamic GCLK clock source selection (not applicable for DPCLK pins and internal logic input)
- GCLK multiplexing
- GCLK network power down (dynamic enable and disable)

Table 2-3: Clock Control Block Inputs

Input	Description
Dedicated clock input pins	Dedicated clock input pins can drive clocks or global signals, such as synchronous and asynchronous clears, presets, or clock enables onto given GCLKs.

Input	Description
DPCLK pins	DPCLK pins are bidirectional dual function pins that are used for high fan-out control signals, such as protocol signals, TRDY and IRDY signals for PCI via the GCLK. Clock control blocks that have inputs driven by DPCLK pins cannot drive PLL inputs.
PLL counter outputs	PLL counter outputs can drive the GCLK.
Internal logic	You can drive the GCLK through logic array routing to enable the internal logic elements (LEs) to drive a high fan-out, low-skew signal path. Clock control blocks that have inputs driven by internal logic cannot drive PLL inputs.

Figure 2-3: Clock Control Block

Notes:

(1) The clkswitch signal can either be set through the configuration file or dynamically set when using the manual PLL switchover feature. The output of the multiplexer is the input clock (fIN) for the PLL.

(2) The clkselect[1..0] signals are fed by internal logic. You can use the clkselect[1..0] signals to dynamically select the clock source for the GCLK when the device is in user mode. Only one PLL (applicable to PLLs on the same side) can be selected as the clock source to the GCLK.

(3) The static clock select signals are set in the configuration file. Therefore, dynamic control when the device is in user mode is not feasible.

(4) You can use internal logic to enable or disable the GCLK in user mode.

Each MAX 10 device has a maximum of 20 clock control blocks. There are five clock control blocks on each side of the device.
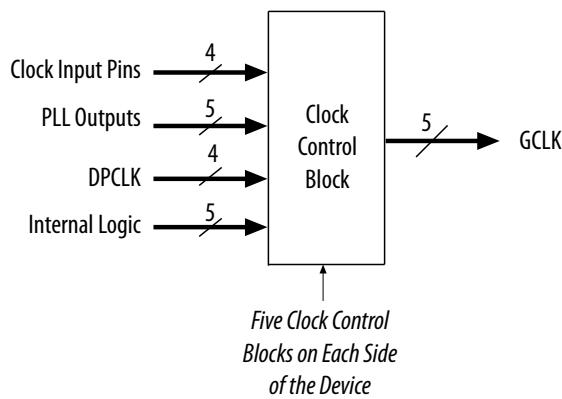
Each PLL generates five clock outputs through the `c[4..0]` counters. Two of these clocks can drive the GCLK through a clock control block.

From the Clock Control Block Inputs table, only the following inputs can drive into any given clock control block:

- Two dedicated clock input pins
- Two PLL counter outputs
- One `DPCLK` pin
- One source from internal logic

The output from the clock control block in turn feeds the corresponding GCLK. The GCLK can drive the PLL input if the clock control block inputs are outputs of another PLL or dedicated clock input pins. Normal I/O pins cannot drive the PLL input clock port.

Figure 2-4: Clock Control Block on Each Side of the Device

Out of these five inputs to any clock control block, the two clock input pins and two PLL outputs are dynamically selected to feed a GCLK. The clock control block supports static selection of the signal from internal logic.

Related Information

Global Clock Network Power Down

You can disable the MAX 10 GCLK (power down) by using both static and dynamic approaches. In the static approach, configuration bits are set in the configuration file generated by the Quartus® Prime software, which automatically disables unused GCLKs. The dynamic clock enable or disable feature allows internal logic to control clock enable or disable of the GCLKs.

When a clock network is disabled, all the logic fed by the clock network is in an off-state, reducing the overall power consumption of the device. This function is independent of the PLL and is applied directly on the clock network.

You can set the input clock sources and the `clkena` signals for the GCLK multiplexers through the ALTCLKCTRL IP core parameter editor in the Quartus Prime software.
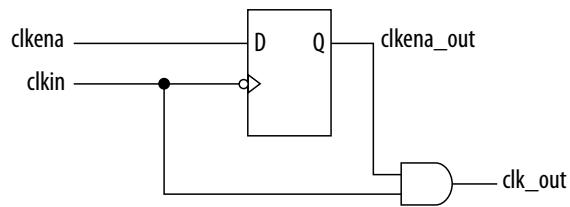
Related Information

- **ALTCLKCTRL Parameters** on page 5-1
- **ALTCLKCTRL Ports and Signals** on page 5-2

Clock Enable Signals

The MAX 10 devices support `clkena` signals at the GCLK network level. This allows you to gate off the clock even when a PLL is used. After reenabling the output clock, the PLL does not need a resynchroniza-tion or relock period because the circuit gates off the clock at the clock network level. In addition, the PLL can remain locked independent of the `clkena` signals because the loop-related counters are not affected.
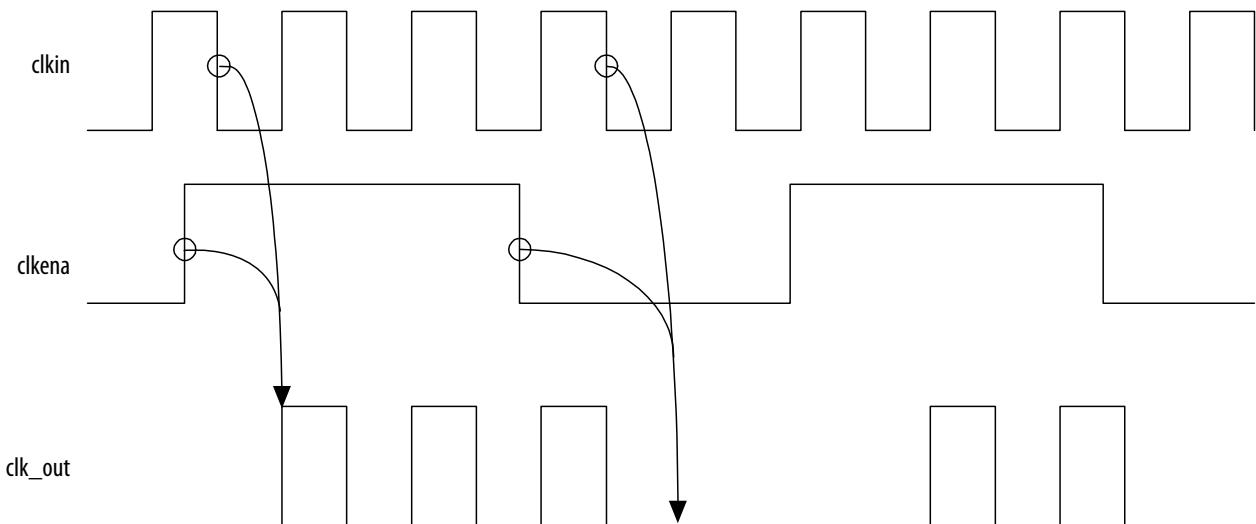
Figure 2-5: clkena Implementation

Note: The `clkena` circuitry controlling the `C0` output of the PLL to an output pin is implemented with two registers instead of a single register.

Figure 2-6: Example Waveform of clkena Implementation with Output Enable

The `clkena` signal is sampled on the falling edge of the clock (`clkin`). This feature is useful for applications that require low power or sleep mode.

The `clkena` signal can also disable clock outputs if the system is not tolerant to frequency overshoot during PLL resynchronization.

Related Information

Internal Oscillator Architecture and Features

MAX 10 devices have built-in internal ring oscillator with clock multiplexers and dividers. The internal ring oscillator operates up to 232 MHz which is not accessible. This operating frequency further divides down to slower frequencies.

By default internal oscillator is turned off in user mode. You can turn on the oscillator by asserting the `oscena` signal in the Internal Oscillator IP core.

When the `oscena` input signal is asserted, the oscillator is enabled and the output can be routed to the logic array through the `clkout` output signal. When the `oscena` signal is set low, the `clkout` signal is constant high. You can analyze this delay using the TimeQuest timing analyzer.
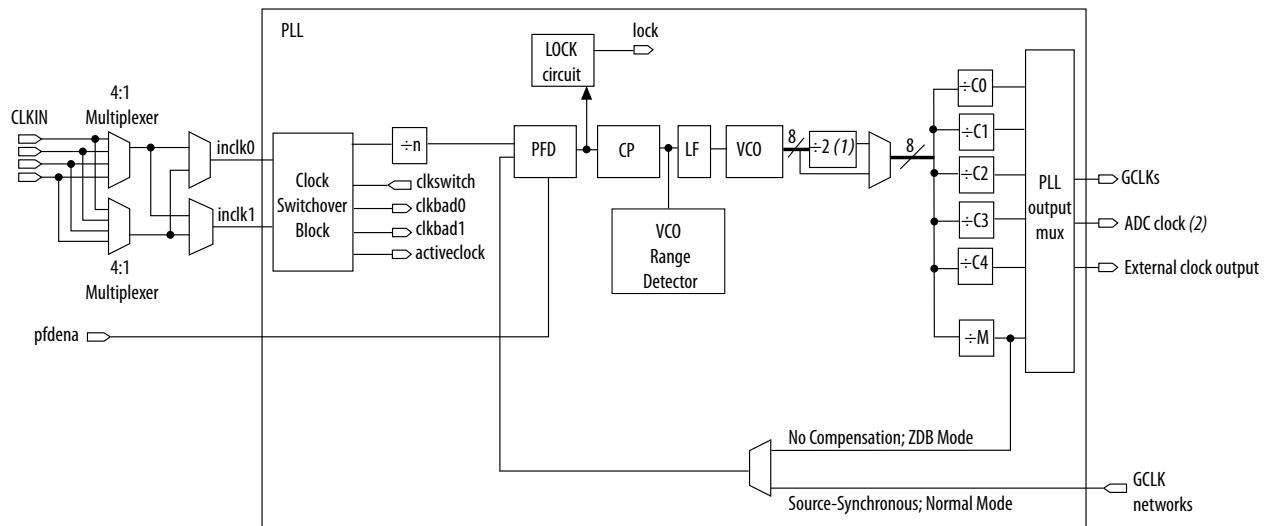
PLLs Architecture and Features

PLL Architecture

The main purpose of a PLL is to synchronize the phase and frequency of the voltage-controlled oscillator (VCO) to an input reference clock.

Figure 2-7: MAX 10 PLL High-Level Block Diagram

Each clock source can come from any of the two or four clock pins located on the same side of the device as the PLL.

Notes:
(1) This is the VCO post-scale counter K.
(2) Only counter C0 of PLL1 and PLL3 can drive the ADC clock.

Phase-Frequency Detector (PFD)

The PFD has inputs from the feedback clock, f_{FB}, and the input reference clock, f_{REF}. The PLL compares the rising edge of the input reference clock to a feedback clock using a PFD. The PFD produces an up or down signal that determines whether the VCO needs to operate at a higher or lower frequency.

Charge Pump (CP)

If the charge pump receives a logic high on the up signal, current is driven into the loop filter. If the charge pump receives a logic high on the down signal, current is drawn from the loop filter.

Loop Filter (LF)

The loop filter converts the up and down signals from the PFD to a voltage that is used to bias the VCO. The loop filter filters out glitches from the charge pump and prevents voltage overshoot, which minimizes jitter on the VCO.

Voltage-Controlled Oscillator (VCO)

The voltage from the charge pump determines how fast the VCO operates. The VCO is implemented as a four-stage differential ring oscillator. A divide counter, M, is inserted in the feedback loop to increase the VCO frequency, f_{VCO}, above the input reference frequency, f_{REF}.

The VCO frequency is determined using the following equation:

$$f_{VCO} = f_{REF} \times M = f_{IN} \times M/N \,,$$

where f_{IN} is the input clock frequency to the PLL and N is the pre-scale counter.

The VCO frequency is a critical parameter that must be between 600 and 1,300 MHz to ensure proper operation of the PLL. The Quartus Prime software automatically sets the VCO frequency within the recommended range based on the clock output and phase shift requirements in your design.

Post-Scale Counters (C)

The VCO output can feed up to five post-scale counters (C0, C1, C2, C3, and C4). These post-scale counters allow the PLL to produce a number of harmonically-related frequencies.

Internal Delay Elements

The MAX 10 PLLs have internal delay elements to compensate for routing on the GCLK networks and I/O buffers. These internal delays are fixed.

PLL Outputs

The MAX 10 PLL supports up to 5 GCLK outputs and 1 dedicated external clock output. The output frequency, f_{OUT}, to the GCLK network or dedicated external clock output is determined using the following equation:

$$f_{REF} = f_{IN}/N \text{ and}$$

$$f_{OUT} = f_{VCO}/C = (f_{REF} \times M)/C = (f_{IN} \times M)/(N \times C),$$

where C is the setting on the C0, C1, C2, C3, or C4 counter.

PLL Features

Table 2-4: MAX 10 PLL Features

Feature	Support
`C` output counters	5
`M`, `N`, `C` counter sizes	1 to 512 [2]
Dedicated clock outputs	1 single-ended or 1 differential
Dedicated clock input pins	4 single-ended or 2 differential
Spread-spectrum input clock tracking	Yes [3]
PLL cascading	Through GCLK
Source synchronous compensation	Yes
No compensation mode	Yes
Normal compensation	Yes
Zero-delay buffer compensation	Yes
Phase shift resolution	Down to 96 ps increments [4]
Programmable duty cycle	Yes
Output counter cascading	Yes
Input clock switchover	Yes
User mode reconfiguration	Yes
Loss of lock detection	Yes
4:1 multiplexer `CLK` input selection	Yes

PLL Locations
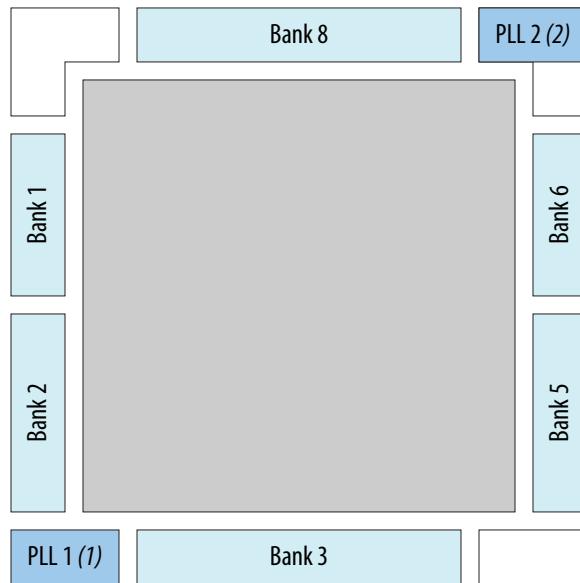
The following figures show the physical locations of the PLLs. Every index represents one PLL in the device. The physical locations of the PLLs correspond to the coordinates in the Quartus Prime Chip Planner.

---

[2] `C` counters range from 1 through 512 if the output clock uses a 50% duty cycle. For any output clocks using a non-50% duty cycle, the post-scale counters range from 1 through 256.

[3] Only applicable if the input clock jitter is in the input jitter tolerance specifications.

[4] The smallest phase shift is determined by the VCO period divided by eight. For degree increments, the MAX 10 device family can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.
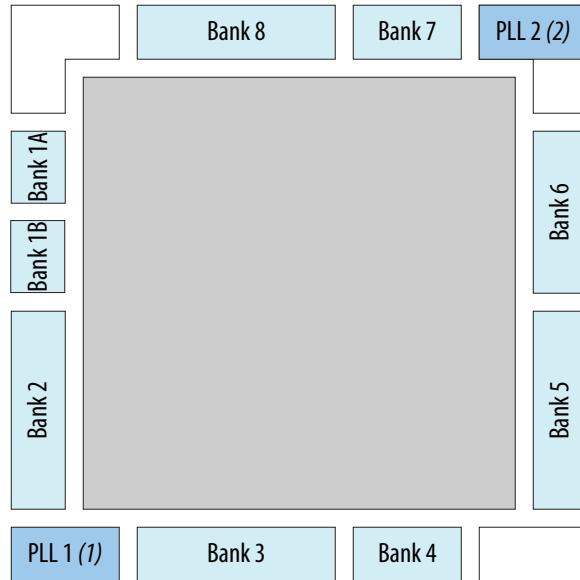
Figure 2-8: PLL Locations for 10M02 Device
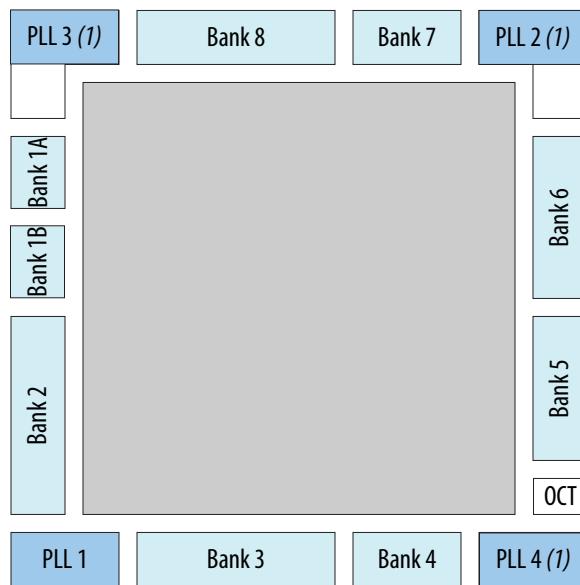
Notes:
(1) Available on all packages except V36 package.
(2) Available on U324 and V36packages only.

Figure 2-9: PLL Locations for 10M04 and 10M08 Devices

Notes:
(1) Available on all packages except V81 package.
(2) Available on F256, F484, U324, and V81 packages only.

Figure 2-10: PLL Locations for 10M16, 10M25, 10M40 and 10M50 Devices

Note:
(1) Available on all packages except E144 and U169 packages.

Clock Pin to PLL Connections

Table 2-5: MAX 10 Dedicated Clock Input Pin Connectivity to PLL

Dedicated Clock Pin	PLL
CLK[0,1][p,n]	PLL1, PLL3
CLK[2,3][p,n]	PLL2, PLL4
CLK[4,5][p,n]	PLL2, PLL3
CLK[6,7][p,n]	PLL1, PLL4

PLL Counter to GCLK Connections

Table 2-6: MAX 10 PLL Counter Connectivity to the GCLK Networks

PLL Counter Output	GCLK
PLL1_C0	GCLK[0,3,15,18]
PLL1_C1	GCLK[1,4,16,19]
PLL1_C2	GCLK[0,2,15,17]
PLL1_C3	GCLK[1,3,16,18]
PLL1_C4	GCLK[2,4,17,19]
PLL2_C0	GCLK[5,8,10,13]

PLL Counter Output	GCLK
PLL2_C1	GCLK[6,9,11,14]
PLL2_C2	GCLK[5,7,10,12]
PLL2_C3	GCLK[6,8,11,13]
PLL2_C4	GCLK[7,9,12,14]
PLL3_C0[5]	GCLK[0,3,10,13]
PLL3_C1[5]	GCLK[1,4,11,14]
PLL3_C2[5]	GCLK[0,2,10,12]
PLL3_C3[5]	GCLK[1,3,11,13]
PLL3_C4[5]	GCLK[2,4,12,14]
PLL4_C0[5]	GCLK[5,8,15,18]
PLL4_C1[5]	GCLK[6,9,16,19]
PLL4_C2[5]	GCLK[5,7,15,17]
PLL4_C3[5]	GCLK[6,8,16,18]
PLL4_C4[5]	GCLK[7,9,17,19]

PLL Control Signals

You can use the following three signals to observe and control the PLL operation and resynchronization.

pfdena

Use the pfdena signal to maintain the last locked frequency so that your system has time to store its current settings before shutting down.

The pfdena signal controls the PFD output with a programmable gate. The PFD circuit is enabled by default. When the PFD circuit is disabled, the PLL output does not depend on the input clock, and tends to drift outside of the lock window.

areset

The areset signal is the reset or resynchronization input for each PLL. The device input pins or internal logic can drive these input signals.

When you assert the areset signal, the PLL counters reset, clearing the PLL output and placing the PLL out of lock. The VCO is then set back to its nominal setting. When the areset signal is deasserted, the PLL resynchronizes to its input as it relocks.

The assertion of the areset signal does not disable the VCO, but instead resets the VCO to its nominal value. The only time that the VCO is completely disabled is when you do not have a PLL instantiated in your design.

---
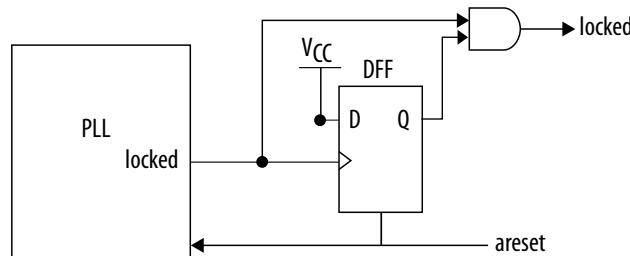
[5] This only applies to 10M16, 10M25, 10M40, and 10M50 devices.

locked

The `locked` output indicates that the PLL has locked onto the reference clock and the PLL clock outputs are operating at the desired phase and frequency set in the ALTPLL IP core parameter editor.

Altera recommends using the `areset` and `locked` signals in your designs to control and observe the status of your PLL. This implementation is illustrated in the following figure.

Figure 2-11: locked Signal Implementation

Note: If you use the SignalTap® II tool to probe the `locked` signal before the D flip-flop, the `locked` signal goes low only when `areset` is deasserted. If the `areset` signal is not enabled, the extra logic is not implemented in the ALTPLL IP core.

Related Information

- **Guideline: PLL Control Signals** on page 3-2
- **PLL Control Signals Parameter Settings** on page 6-2
- **ALTPLL Ports and Signals** on page 6-6

Clock Feedback Modes

The MAX 10 PLLs support up to four different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle.

The PLL fully compensates input and output delays only when you use the dedicated clock input pins associated with a given PLL as the clock sources.

For example, when using `PLL1` in normal mode, the clock delays from one of the following clock input pins to the PLL and the PLL clock output-to-destination register are fully compensated:

- `CLK0`
- `CLK1`
- `CLK2`
- `CLK3`

When driving the PLL using the GCLK network, the input and output delays might not be fully compensated in the Quartus Prime software.
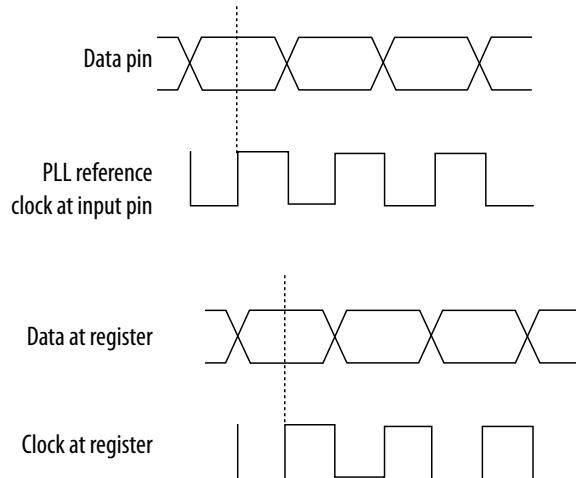
Related Information
Operation Modes Parameter Settings on page 6-1

Source Synchronous Mode

If the data and clock arrive at the same time at the input pins, the phase relationship between the data and clock remains the same at the data and clock ports of any I/O element input register.

You can use this mode for source synchronous data transfers. Data and clock signals at the I/O element experience similar buffer delays as long as both signals use the same I/O standard.

Figure 2-12: Example of Phase Relationship Between Clock and Data in Source Synchronous Mode
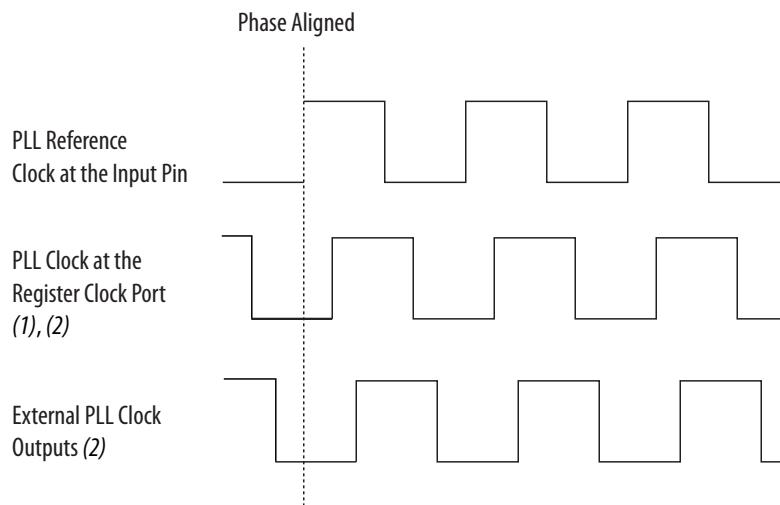
Source synchronous mode compensates for clock network delay, including any difference in delay between the following two paths:

- Data pin to I/O element register input
- Clock input pin to the PLL PFD input

For all data pins clocked by a source synchronous mode PLL, set the input pin to the register delay chain in the I/O element to zero in the Quartus Prime software. All data pins must use the **PLL COMPENSATED logic** option in the Quartus Prime software.

No Compensation Mode
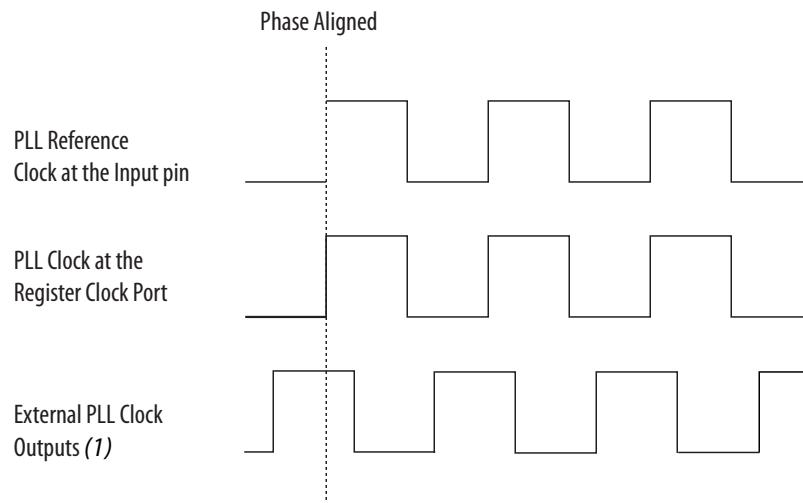
In no compensation mode, the PLL does not compensate for any clock networks. This mode provides better jitter performance because clock feedback into the PFD does not pass through as much circuitry. Both the PLL internal and external clock outputs are phase-shifted with respect to the PLL clock input.

Figure 2-13: Example of Phase Relationship Between the PLL Clocks in No Compensation Mode

Phase Aligned

PLL Reference
Clock at the Input Pin

PLL Clock at the
Register Clock Port
(1), *(2)*

External PLL Clock
Outputs *(2)*

Notes:
(1) Internal clocks fed by the PLL are phase-aligned to each other.
(2) The PLL clock outputs can lead or lag the PLL input clocks. The PLL clock outputs lag the
     PLL input clocks depending on the routine delays.

Normal Mode

In normal mode, the PLL fully compensates the delay introduced by the GCLK network. An internal clock in normal mode is phase-aligned to the input clock pin. In this mode, the external clock output pin has a phase delay relative to the input clock pin. The Quartus Prime software timing analyzer reports any phase difference between the two.
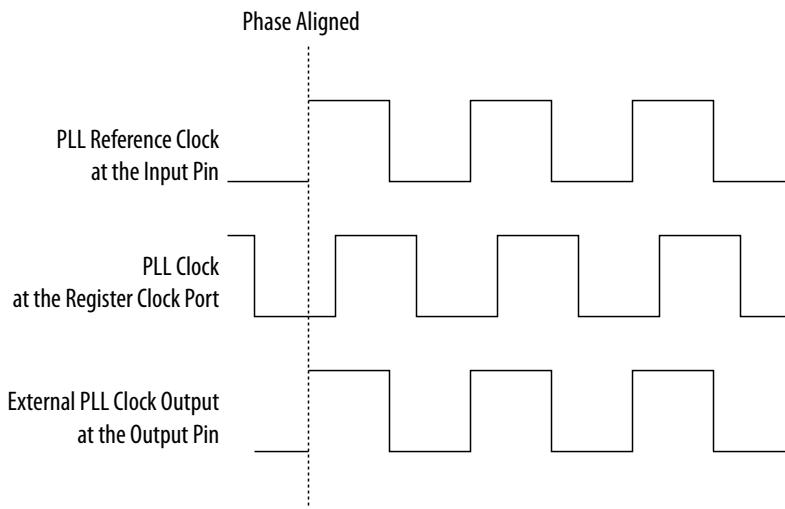
Figure 2-14: Example of Phase Relationship Between the PLL Clocks in Normal Compensation Mode

Phase Aligned

PLL Reference
Clock at the Input pin

PLL Clock at the
Register Clock Port

External PLL Clock
Outputs *(1)*

Note:
(1) The external clock output can lead or lag the PLL internal clock signals.

Zero-Delay Buffer Mode

In zero-delay buffer (ZDB) mode, the external clock output pin is phase-aligned with the clock input pin for zero delay through the device. When using this mode, use the same I/O standard for the input clock and output clocks to ensure clock alignment at the input and output pins.
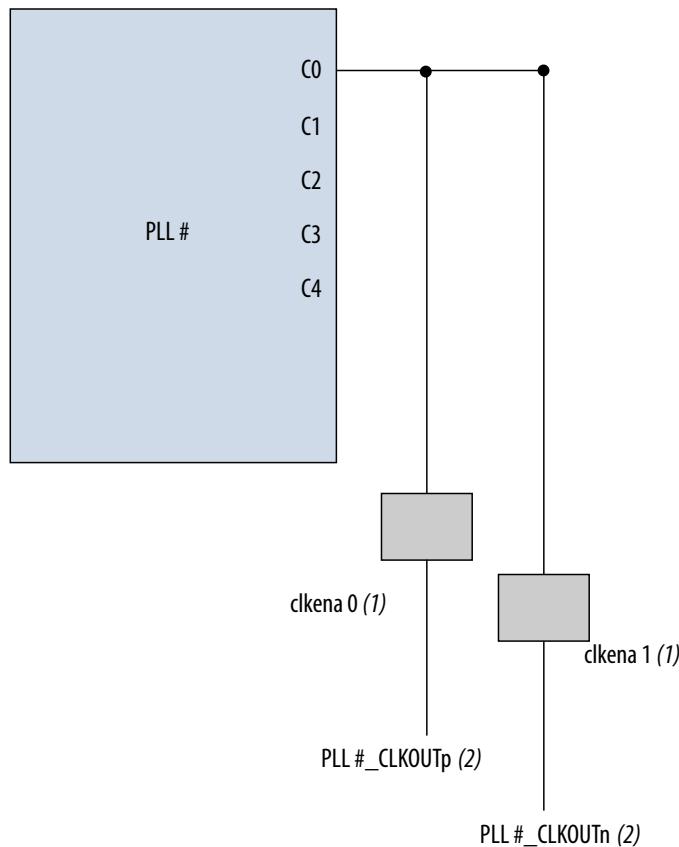
Figure 2-15: Example of Phase Relationship Between the PLL Clocks in ZDB Mode

PLL External Clock Output

Each PLL in the MAX 10 devices supports one single-ended clock output or one differential clock output. Only the C0 output counter can feed the dedicated external clock outputs without going through the GCLK. Other output counters can feed other I/O pins through the GCLK.

Figure 2-16: PLL External Clock Output

Notes:
(1) These external clock enable signals are available only when using the ALTCLKCTRL IP core.
(2) PLL#_CLKOUTp and PLL#_CLKOUTn pins are dual-purpose I/O pins that you can use as one single-ended
     or one differential clock output.

Each pin of a differential output pair is 180° out of phase. To implement the 180° out-of-phase pin in a pin pair, the Quartus Prime software places a NOT gate in the design into the I/O element.

The clock output pin pairs support the following I/O standards:

- Same I/O standard as the standard output pins (in the top and bottom banks)
- LVDS
- LVPECL
- Differential high-speed transceiver logic (HSTL)
- Differential SSTL

The MAX 10 PLLs can drive out to any regular I/O pin through the GCLK. You can also use the external clock output pins as general-purpose I/O pins if you do not require any external PLL clocking.

Related Information
MAX 10 General Purpose I/O User Guide
Provides more information about the I/O standards supported by the PLL clock output pins.

Send Feedback

ADC Clock Input from PLL

Only the `C0` output counter from `PLL1` and `PLL3` can drive the ADC clock.

Counter `C0` has dedicated path to the ADC clock input.

Spread-Spectrum Clocking

The MAX 10 devices allow a spread-spectrum input with typical modulation frequencies. However, the device cannot automatically detect that the input is a spread-spectrum signal. Instead, the input signal looks like deterministic jitter at the input of the PLL.

The MAX 10 PLLs can track a spread-spectrum input clock if the input signal meets the following conditions:

- The input signal is within the input jitter tolerance specifications.
- The modulation frequency of the input clock is below the PLL bandwidth as specified in the Fitter report.

MAX 10 devices cannot generate spread-spectrum signals internally.

PLL Programmable Parameters

Programmable Duty Cycle

The programmable duty cycle allows PLLs to generate clock outputs with a variable duty cycle. This feature is supported on the PLL post-scale counters.

The duty cycle setting is achieved by a low and high time-count setting for the post-scale counters. To determine the duty cycle choices, the Quartus Prime software uses the frequency input and the required multiply or divide rate.

The post-scale counter value determines the precision of the duty cycle. The precision is defined as 50% divided by the post-scale counter value. For example, if the `C0` counter is 10, steps of 5% are possible for duty cycle choices between 5 to 90%.

Combining the programmable duty cycle with programmable phase shift allows the generation of precise nonoverlapping clocks.

Related Information
Post-Scale Counters (C0 to C4) on page 4-4
Provides more information about configuring the duty cycle of the post-scale counters in real time.

Programmable Bandwidth

The PLL bandwidth is the measure of the PLL's ability to track the input clock and its associated jitter. The MAX 10 PLLs provide advanced control of the PLL bandwidth using the programmable characteristics of the PLL loop, including loop filter and charge pump. The 3-dB frequency of the closed-loop gain in the PLL determines the PLL bandwidth. The bandwidth is approximately the unity gain point for open loop PLL response.

Related Information

- **Programmable Bandwidth with Advanced Parameters** on page 4-3

- **Charge Pump and Loop Filter** on page 4-6
  Provides more information about the PLL components to update PLL bandwidth in real time.
- **Programmable Bandwidth Parameter Settings** on page 6-2

Programmable Phase Shift

The MAX 10 devices use phase shift to implement clock delays. You can phase shift the output clocks from the MAX 10 PLLs using one of the following methods:

- Fine resolution using VCO phase taps
- Coarse resolution using counter starting time

The VCO phase output and counter starting time are the most accurate methods of inserting delays. These methods are purely based on counter settings, which are independent of process, voltage, and temperature.

The MAX 10 devices support dynamic phase shifting of VCO phase taps only. The phase shift is configurable for any number of times. Each phase shift takes about one `scanclk` cycle, allowing you to implement large phase shifts quickly.

Fine Resolution Phase Shift

Fine resolution phase shifts are implemented by allowing any of the output counters (`C[4..0]`) or the `M` counter to use any of the eight phases of the VCO as the reference clock. This allows you to adjust the delay time with a fine resolution. The following equation shows the minimum delay time that you can insert using this method.

Figure 2-17: Fine Resolution Phase Shift Equation

f_{REF} in this equation is the input reference clock frequency

$$\Phi_{fine} = \frac{T_{VCO}}{8} = \frac{1}{8f_{VCO}} = \frac{1}{8} \times \frac{N}{M \times f_{REF}}$$
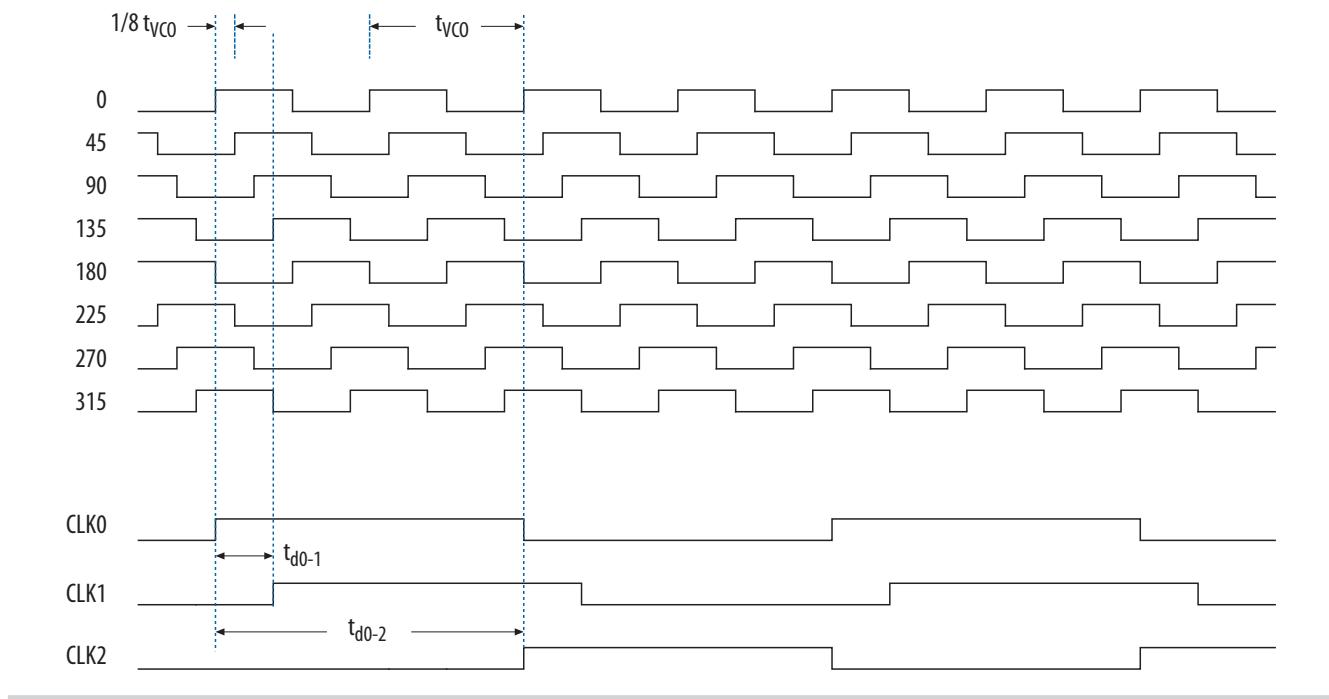
For example, if f_{REF} is 100 MHz, `N` = 1, and `M` = 8, then f_{VCO} = 800 MHz, and Φ_{fine} = 156.25 ps. The PLL operating frequency defines this phase shift, a value that depends on the reference clock frequency and counter settings.

The following figure shows an example of phase shift insertion using the fine resolution through VCO phase taps method. The eight phases from the VCO are shown and labeled for reference.

Figure 2-18: Example of Delay Insertion Using VCO Phase Output and Counter Delay Time

The observations in this example are as follows:

- CLK0 is based on 0° phase from the VCO and has the *C* value for the counter set to one.
- CLK1 signal is divided by four, two VCO clocks for high time and two VCO clocks for low time. CLK1 is based on the 135° phase tap from the VCO and has the *C* value for the counter set to one.
- CLK2 signal is also divided by four. In this case, the two clocks are offset by 3 Φ_{fine}. CLK2 is based on the 0° phase from the VCO but has the *C* value for the counter set to three. This creates a delay of two Φ_{coarse} (two complete VCO periods).

Coarse Resolution Phase Shift

Coarse resolution phase shifts are implemented by delaying the start of the counters for a predetermined number of counter clocks.

Figure 2-19: Coarse Resolution Phase Shift Equation

C in this equation is the count value set for the counter delay time—the initial setting in the PLL usage section of the compilation report in the Quartus Prime software. If the initial value is 1, C − 1 = 0° phase shift.

$$\Phi_{coarse} = \frac{C-1}{f_{VCO}} = \frac{(C-1)N}{Mf_{REF}}$$

Related Information

- **Dynamic Phase Configuration Implementation** on page 4-8
- **Dynamic Phase Configuration Counter Selection** on page 4-9
- **Dynamic Phase Configuration with Advanced Parameters** on page 4-9

- **Dynamic Phase Configuration Parameter Settings** on page 6-4

  Provides more information about the ALTPLL IP core parameter settings in the Quartus Prime software.
- **ALTPLL_RECONFIG Parameters** on page 7-1

  Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Quartus Prime software.

Clock Switchover

The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application where a system turns on the redundant clock if the previous clock stops running. The design can perform clock switchover automatically when the clock is no longer toggling or based on a user-controlled signal, `clkswitch`.

The following clock switchover modes are supported in MAX 10 PLLs:

- Automatic switchover—The clock sense circuit monitors the current reference clock. If the current reference clock stops toggling, the reference clock automatically switches to `inclk0` or `inclk1` clock.
- Manual clock switchover—The `clkswitch` signal controls the clock switchover. When the `clkswitch` signal goes from logic low to high, and stays high for at least three clock cycles, the reference clock to the PLL switches from `inclk0` to `inclk1`, or vice-versa.
- Automatic switchover with manual override—This mode combines automatic switchover and manual clock switchover. When the `clkswitch` signal goes high, it overrides the automatic clock switchover function. As long as the `clkswitch` signal is high, any further switchover action is blocked.

Related Information
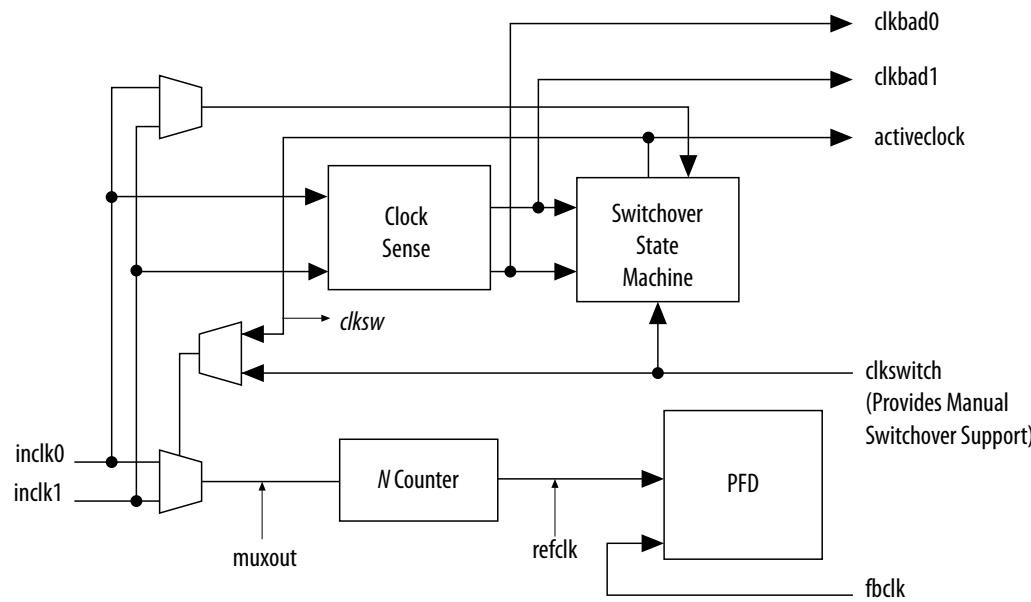
- **Guideline: Clock Switchover** on page 3-4
- **Clock Switchover Parameter Settings** on page 6-3

Automatic Clock Switchover

The MAX 10 PLLs support a fully configurable clock switchover capability.

Figure 2-20: Automatic Clock Switchover Circuit Block Diagram

This figure shows a block diagram of the automatic switchover circuit built into the PLL.

When the current reference clock is not present, the clock sense block automatically switches to the backup clock for PLL reference. You can select a clock source at the backup clock by connecting it to the `inclk1` port of the PLL in your design.

The clock switchover circuit also sends out three status signals—`clkbad[0]`, `clkbad[1]`, and `activeclock`—from the PLL to implement a custom switchover circuit in the logic array.

In automatic switchover mode, the `clkbad[0]` and `clkbad[1]` signals indicate the status of the two clock inputs. When the `clkbad[0]` and `clkbad[1]` signals are asserted, the clock sense block detects that the corresponding clock input has stopped toggling. These two signals are not valid if the frequency difference between `inclk0` and `inclk1` is greater than 20%.

The `activeclock` signal indicates which of the two clock inputs (`inclk0` or `inclk1`) is selected as the reference clock to the PLL. When the frequency difference between the two clock inputs is more than 20%, the `activeclock` signal is the only valid status signal.

Note: Glitches in the input clock may cause the frequency difference between the input clocks to be more than 20%.

When the current reference clock to the PLL stops toggling, use the switchover circuitry to automatically switch from `inclk0` to `inclk1` that runs at the same frequency. This automatic switchover can switch back and forth between the `inclk0` and `inclk1` clocks any number of times when one of the two clocks fails and the other clock is available.

For example, in applications that require a redundant clock with the same frequency as the reference clock, the switchover state machine generates a signal (`clksw`) that controls the multiplexer select input. In this case, `inclk1` becomes the reference clock for the PLL.

When using automatic clock switchover mode, the following requirements must be satisfied:
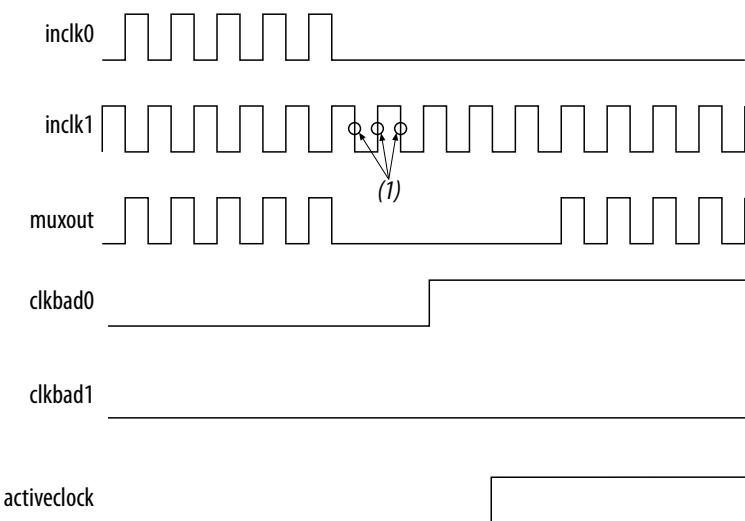
- Both clock inputs must be running when the FPGA is configured.
- The period of the two clock inputs differ by no more than 20%.

If the current clock input stops toggling while the other clock is also not toggling, switchover is not initiated and the clkbad[0..1] signals are not valid. If both clock inputs do not have the same frequency, but their period difference is within 20%, the clock sense block detects when a clock stops toggling. However, the PLL might lose lock after the switchover completes and needs time to relock.

Note: Altera recommends resetting the PLL using the areset signal to maintain the phase relationships between the PLL input and output clocks when using clock switchover.

Figure 2-21: Example of Automatic Switchover After Loss of Clock Detection

This figure shows an example waveform of the switchover feature in automatic switchover mode. In this example, the inclk0 signal remains low. After the inclk0 signal remains low for approximately two clock cycles, the clock sense circuitry drives the clkbad[0] signal high. Since the reference clock signal is not toggling, the switchover state machine controls the multiplexer through the clksw signal to switch to the backup clock, inclk1.

Note:
(1) Switchover is enabled on the falling edge of inclk0 or inclk1, depending on which clock is available. In this figure, switchover is enabled on the falling edge of inclk1.

Automatic Switchover with Manual Override

In automatic switchover with manual override mode, you can use the clkswitch signal for user- or system-controlled switch conditions. You can use this mode for same-frequency switchover, or to switch between inputs of different frequencies.

For example, if inclk0 is 66 MHz and inclk1 is 200 MHz, you must control the switchover using the clkswitch signal. The automatic clock sense circuitry cannot monitor clock input (inclk0 and inclk1) frequencies with a frequency difference of more than 20%.
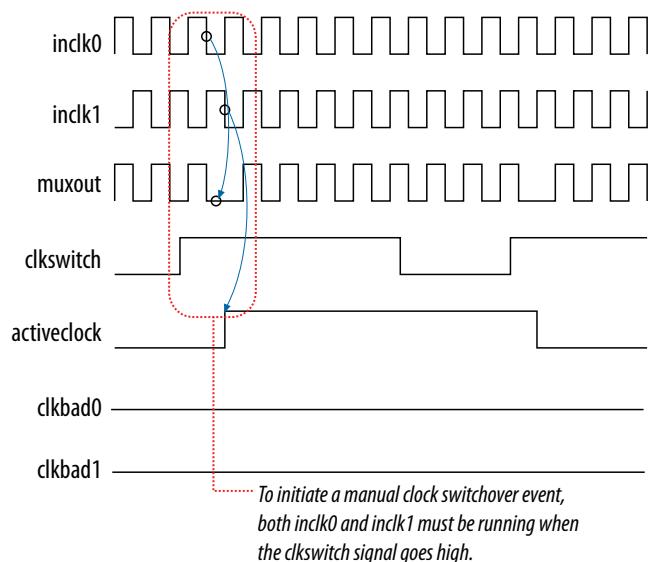
This feature is useful when clock sources originate from multiple cards on the backplane, requiring a system-controlled switchover between frequencies of operation.

You must choose the backup clock frequency and set the M, N, and C counters so that the VCO operates within the recommended frequency range.

The following figure shows a clock switchover waveform controlled by the clkswitch signal. In this case, both clock sources are functional and inclk0 is selected as the reference clock. The clkswitch signal goes high, which starts the switchover sequence. On the falling edge of inclk0, the counter's reference clock, muxout, is gated off to prevent clock glitching. On the falling edge of inclk1, the reference clock multiplexer switches from inclk0 to inclk1 as the PLL reference. The activeclock signal is asserted to indicate the clock that is currently feeding the PLL, which is inclk1.

In automatic override with manual switchover mode, the activeclock signal mirrors the clkswitch signal. Since both clocks are still functional during the manual switch, neither clkbad signal goes high. Because the switchover circuit is positive-edge sensitive, the falling edge of the clkswitch signal does not cause the circuit to switch back from inclk1 to inclk0. When the clkswitch signal goes high again, the process repeats.

Figure 2-22: Example of Clock Switchover Using the clkswitch (Manual) Control

To initiate a manual clock switchover event, both inclk0 and inclk1 must be running when the clkswitch signal goes high.

The clkswitch signal and automatic switch work only if the clock being switched to is available. If the clock is not available, the state machine waits until the clock is available.

Manual Clock Switchover

In manual clock switchover mode, the clkswitch signal controls whether inclk0 or inclk1 is selected as the input clock to the PLL. By default, inclk0 is selected.

A clock switchover event is initiated when the clkswitch signal transitions from logic low to logic high, and is being held high for at least three inclk cycles. You must bring the clkswitch signal back to low again to perform another switchover event. If you do not require another switchover event, you can leave the clkswitch signal in a logic high state after the initial switch. Pulsing the clkswitch signal high for at least three inclk cycles performs another switchover event.

If `inclk0` and `inclk1` have different frequencies and are always running, the minimum amount of time for which `clkswitch` signal is high must be greater than or equal to three of the slower-frequency `inclk0` and `inclk1` cycles.

PLL Cascading

Related Information

Guideline: PLL Cascading on page 3-3
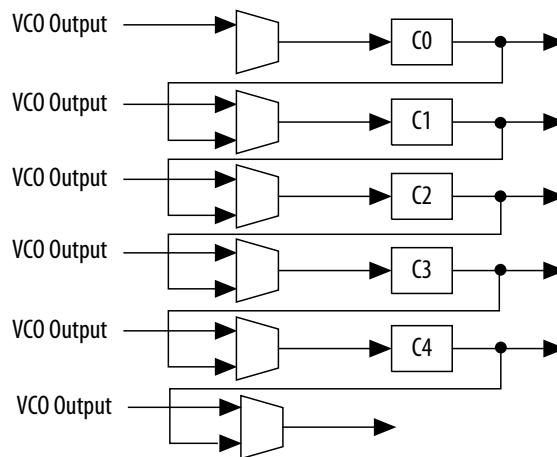
PLL-to-PLL Cascading

Two PLLs are cascaded to each other through the clock network. If your design cascades PLLs, the source (upstream) PLL must have a low-bandwidth setting and the destination (downstream) PLL must have a high-bandwidth setting.

Counter-to-Counter Cascading

The MAX 10 PLLs support post-scale counter cascading to create counters larger than 512. This is implemented by feeding the output of one `C` counter into the input of the next `C` counter.

Figure 2-23: Counter-to-Counter Cascading

When cascading counters to implement a larger division of the high-frequency VCO clock, the cascaded counters behave as one counter with the product of the individual counter settings.

For example, if `C0` = 4 and `C1` = 2, the cascaded value is `C0` x `C1` = 8.

The Quartus Prime software automatically sets all the post-scale counter values for cascading in the configuration file. Post-scale counter cascading cannot be performed using PLL reconfiguration.

PLL Reconfiguration

The PLLs use several divide counters and different VCO phase taps to perform frequency synthesis and phase shifts. In MAX 10 PLLs, you can reconfigure both counter settings and phase shift the PLL output clock in real time. You can also change the charge pump and loop filter components, which dynamically affects the PLL bandwidth.

The following PLL components are configurable in real time:

- Pre-scale counter (N)
- Feedback counter (M)
- Post-scale output counters (C0-C4)
- Charge pump current (I_{CP})
- Loop filter components (R, C)

You can use these PLL components to update the following settings in real time without reconfiguring the entire FPGA:

- Output clock frequency
- PLL bandwidth
- Phase shift
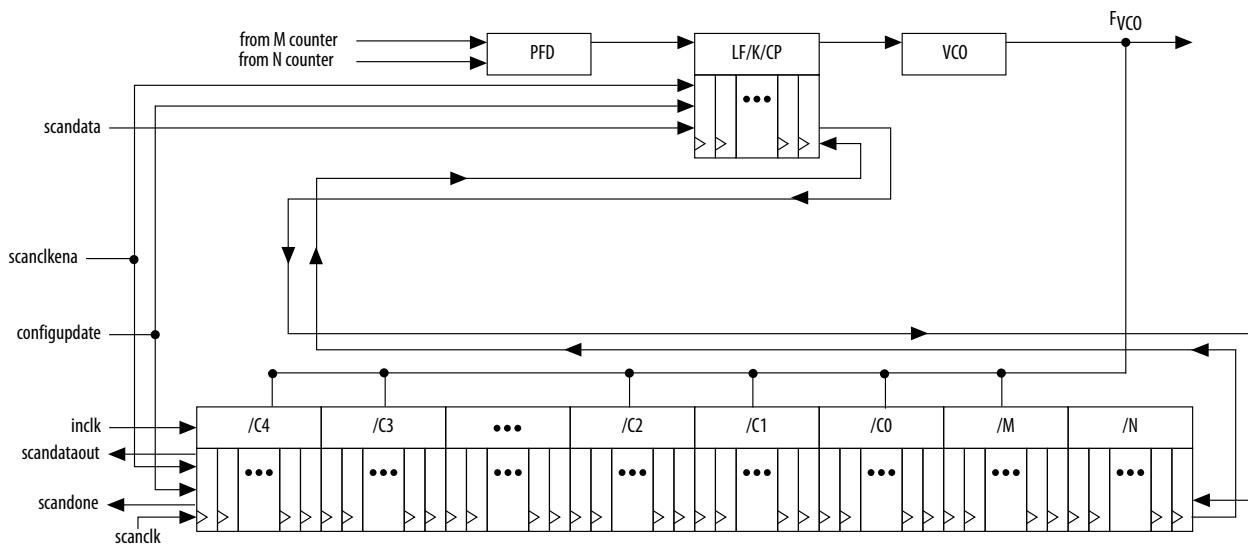
The ability to reconfigure the PLL in real time is useful in applications that may operate in multiple frequencies. It is also useful in prototyping environments, allowing you to sweep PLL output frequencies and dynamically adjust the output clock phase.

For instance, a system generating test patterns is required to generate and send patterns at 75 or 150 MHz, depending on the requirements of the device under test. Reconfiguring the PLL components in real time allows you to switch between two such output frequencies in a few microseconds.

You can also use this feature to adjust clock-to-out (t_{CO}) delays in real time by changing the PLL output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

Figure 2-24: PLL Reconfiguration Scan Chain

This figure shows the dynamic adjustment of the PLL counter settings by shifting their new settings into a serial shift register chain or scan chain. Serial data shifts to the scan chain via the scandata port, and shift registers are clocked by scanclk. The maximum scanclk frequency is 100 MHz. After shifting the last bit of data, asserting the configupdate signal for at least one scanclk clock cycle synchronously updates the PLL configuration bits with the data in the scan registers.

The counter settings are updated synchronously to the clock frequency of the individual counters. Therefore, not all counters update simultaneously.

The dynamic reconfiguration scheme uses configuration files, such as the Hexadecimal-format file (**.hex**) or the Memory Initialization file (**.mif**). These files are used together with the ALTPLL_RECONFIG IP core to perform the dynamic reconfiguration.

Related Information

- **Guideline: .mif Streaming in PLL Reconfiguration** on page 3-5
- **PLL Dynamic Reconfiguration Implementation** on page 4-4
- **PLL Dynamic Reconfiguration Parameter Settings** on page 6-4
  Provides more information about the ALTPLL IP core parameter settings in the Quartus Prime software.
- **ALTPLL_RECONFIG Parameters** on page 7-1
  Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Quartus Prime software.

Clock Networks Design Considerations

Guideline: Clock Enable Signals

Altera recommends using the `clkena` signals when switching the clock source to the PLLs or GCLK. The recommended sequence is as follows:

1. Disable the primary output clock by deasserting the `clkena` signal.
2. Switch to the secondary clock using the dynamic select signals of the clock control block.
3. Allow some clock cycles of the secondary clock to pass before reasserting the `clkena` signal. The exact number of clock cycles to wait before enabling the secondary clock depends on your design. You can build a custom logic to ensure a glitch-free transition when switching between different clock sources.

Related Information

- **Clock Enable Signals** on page 2-7
- **ALTCLKCTRL Parameters** on page 5-1
- **ALTCLKCTRL Ports and Signals** on page 5-2

Guideline: Connectivity Restrictions

The following guidelines describe the restrictions associated with the signal sources that can drive the `inclk` input:

- You must use the `inclk` ports that are consistent with the `clkselect` ports.
- When you are using multiple input sources, the `inclk` ports can only be driven by the dedicated clock input pins and the PLL clock outputs.
- If the clock control block feeds any `inclk` port of another clock control block, both clock control blocks must be able to be reduced to a single clock control block of equivalent functionality.
- When you are using the glitch-free switchover feature, the clock you are switching from must be active. If the clock is not active, the switchover circuit cannot transition from the clock you originally selected.

ALTERA®

Internal Oscillator Design Considerations

Guideline: Connectivity Restrictions

You cannot drive the PLLs with internal oscillator.

PLLs Design Considerations

Guideline: PLL Control Signals

You must include the `areset` signal in your designs if one of the following conditions is true:

- PLL reconfiguration or clock switchover is enabled in your design.
- Phase relationships between the PLL input clock and output clocks must be maintained after a loss-of-lock condition.
- The input clock to the PLL is toggling or unstable at power-up.
- The `areset` signal is asserted after the input clock is stable and within specifications.

Related Information
PLL Control Signals on page 2-13

Guideline: Connectivity Restrictions

To comply with simultaneous switching noise (SSN) design guideline, Altera recommends that you do not use unterminated I/O in the same bank as the input clock signal to the PLL.

Related Information
Guidelines: Clock and Asynchronous Control Input Signal
Provides more information about using I/O connectivity restrictions.

Guideline: Self-Reset

The lock time of a PLL is the amount of time required by the PLL to attain the target frequency and phase relationship after device power-up, after a change in the PLL output frequency, or after resetting the PLL.

A PLL might lose lock for a number of reasons, such as the following causes:

- Excessive jitter on the input clock.
- Excessive switching noise on the clock inputs of the PLL.
- Excessive noise from the power supply, causing high output jitter and possible loss of lock.
- A glitch or stopping of the input clock to the PLL.
- Resetting the PLL by asserting the `areset` port of the PLL.
- An attempt to reconfigure the PLL might cause the M counter, N counter, or phase shift to change, causing the PLL to lose lock. However, changes to the post-scale counters do not affect the PLL `locked` signal.
- PLL input clock frequency drifts outside the lock range specification.
- The PFD is disabled using the `pfdena` port. When this happens, the PLL output phase and frequency tend to drift outside of the lock window.

The ALTPLL IP core allows you to monitor the PLL locking process using a lock signal named `locked` and also allows you to set the PLL to self-reset on loss of lock.

Guideline: Output Clocks

Each MAX 10 PLL supports up to five output clocks. You can use the output clock port as a core output clock or an external output clock port. The core output clock feeds the FPGA core and the external output clock feeds the dedicated pins on the FPGA.

The ALTPLL IP core does not have a dedicated output enable port. You can disable the PLL output using the `areset` signal to disable the PLL output counters.

Guideline: PLL Cascading

Consider the following guidelines when cascading PLLs:

- Set the primary PLL to low bandwidth to help filter jitter. Set the secondary PLL to high bandwidth to track the jitter from the primary PLL. You can view the Quartus Prime software compilation report file to ensure the PLL bandwidth ranges do not overlap. If the bandwidth ranges overlap, jitter peaking can occur in the cascaded PLL scheme.

  Note: You can get an estimate of the PLL deterministic jitter and static phase error (SPE) by using the TimeQuest Timing Analyzer in the Quartus Prime software. Use the SDC command `derive_clock_uncertainty` to generate a report titled **PLLJ_PLLSPE_INFO.txt** in your project directory. Then, use `set_clock_uncertainty` command to add jitter and SPE values to your clock constraints.

- Keep the secondary PLL in a reset state until the primary PLL has locked to ensure the phase settings are correct on the secondary PLL.

- You cannot connect any of the `inclk` ports of any PLLs in a cascaded scheme to the clock outputs from PLLs in the cascaded scheme.

Related Information
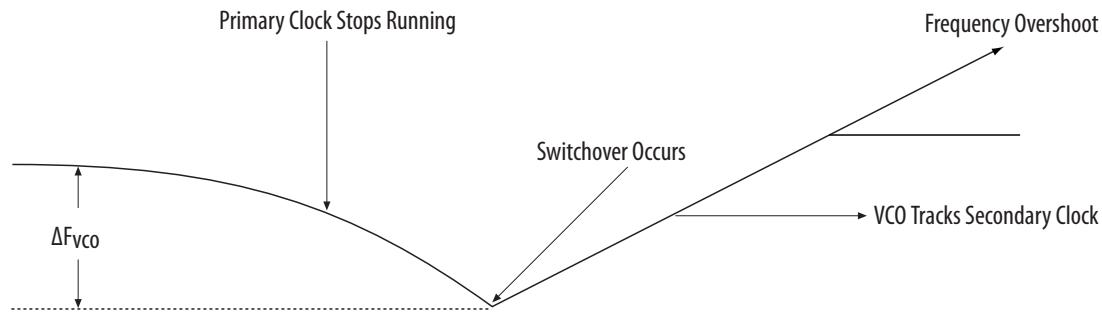
Guideline: Clock Switchover

Use the following guidelines to design with clock switchover in PLLs:

- Clock loss detection and automatic clock switchover requires that the frequency difference between `inclk0` and `inclk1` is within 20% range. Failing to meet this requirement causes the `clkbad[0]` and `clkbad[1]` signals to function improperly.
- When using manual clock switchover, the frequency difference between `inclk0` and `inclk1` can be more than 20%. However, differences between the two clock sources (frequency, phase, or both) can cause the PLL to lose lock. Resetting the PLL ensures that the correct phase relationships are maintained between the input and output clocks.
- Both `inclk0` and `inclk1` must be running when the `clkswitch` signal goes high to start the manual clock switchover event. Failing to meet this requirement causes the clock switchover to malfunction.
- Applications that require a clock switchover feature and a small frequency drift must use a low-bandwidth PLL. When referencing input clock changes, the low-bandwidth PLL reacts slower than a high-bandwidth PLL. When the switchover happens, the low-bandwidth PLL propagates the stoppage of the clock to the output at a slower speed than the high-bandwidth PLL. The low-bandwidth PLL filters out jitter on the reference clock. However, be aware that the low-bandwidth PLL also increases lock time.
- After a switchover occurs, there might be a finite resynchronization period for the PLL to lock onto a new clock. The exact amount of time it takes for the PLL to relock depends on the PLL configuration.
- The phase relationship between the input clock to the PLL and output clock from the PLL is important in your design. Assert `areset` for 10 ns after performing a clock switchover. Wait for the locked signal (or gated lock) to go high before reenabling the output clocks from the PLL.
- Disable the system during switchover if the system is not tolerant of frequency variations during the PLL resynchronization period. You can use the `clkbad[0]` and `clkbad[1]` status signals to turn off the PFD (`pfdena` = **0**) so that the VCO maintains its last frequency. You can also use the switchover state machine to switch over to the secondary clock. After enabling the PFD, the output clock enable signals (`clkena`) can disable clock outputs during the switchover and resynchronization period. After the lock indication is stable, the system can reenable the output clock or clocks.
- The VCO frequency gradually decreases when the primary clock is lost and then increases as the VCO locks onto the secondary clock, as shown in the following figure. After the VCO locks onto the secondary clock, some overshoot can occur (an over-frequency condition) in the VCO frequency.

Figure 3-1: VCO Switchover Operating Frequency

Related Information

- **Clock Switchover** on page 2-22

- **Clock Switchover Parameter Settings** on page 6-3

Guideline: .mif Streaming in PLL Reconfiguration

Consider the following guidelines when using **.mif** streaming in PLL reconfiguration:

- 10M02 devices do not support **.mif** streaming in PLL reconfiguration due to flash size limitation. Altera recommends using an external flash.
- 10M04, 10M08, 10M16, 10M25, 10M40, and 10M50 devices only support **.mif** streaming in single image mode. Altera recommends using an external flash for dual image mode. The MAX 10 devices do not support using both dual image mode and PLL reconfiguration with **.mif** simultaneously.

Related Information
PLL Reconfiguration on page 2-26

Guideline: scandone Signal for PLL Reconfiguration

`scandone` signal must be low before the second PLL reconfiguration. For `scandone` signal to go low, PLL `areset` signal must be asserted.

ALTCLKCTRL IP Core

The clock control block (ALTCLKCTRL) IP core is a clock control function for configuring the clock control block.

The common applications of the ALTCLKCTRL IP core are as follows:

- Dynamic clock source selection—When using the clock control block, you can select the dynamic clock source that drives the global clock network.
- Dynamic power-down of a clock network—The dynamic clock enable or disable feature allows internal logic to power down the clock network. When a clock network is powered down, all the logic fed by that clock network is not toggling, thus reducing the overall power consumption of the device.

The ALTCLKCTRL IP core provides the following features:

- Supports clock control block operation mode specifications
- Supports specification of the number of input clock sources
- Provides an active high clock enable control input

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

ALTPLL IP Core

The ALTPLL IP core specifies the PLL circuitry. You can use this IP core to configure the PLL types, operation modes, and advanced features of the PLL.

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Expanding the PLL Lock Range

The PLL lock range is between the minimum (Freq min lock parameter) and maximum (Freq min lock parameter) input frequency values for which the PLL can achieve lock. Changing the input frequency might cause the PLL to lose lock, but if the input clock remains within the minimum and maximum frequency specifications, the PLL is able to achieve lock. The Quartus Prime software shows these input frequency values in the PLL Summary report located under the Resource Section of the Fitter folder in the Compilation Report.

The Quartus Prime software does not necessarily pick values for the PLL parameters to maximize the lock range. For example, when you specify a 75 MHz input clock in the ALTPLL parameter editor, the actual PLL lock range might be between 70 MHz to 90 MHz. If your application requires a lock range of 50 MHz to 100 MHz, the default lock range of this PLL is insufficient.

For devices that support clock switchover in PLLs, you can use the ALTPLL IP core parameter editor to maximize the lock range.

To extract valid parameter values to maximize your PLL lock range, perform the following steps:

1. In the schematic editor, double-click the ALTPLL instance in your design to open the ALTPLL parameter editor.
2. On the **General/Modes** page, for **What is the frequency of the inclk0 input?**, type the value of the low end of your desired PLL lock range.

   For example, if your application requires a lock range of 50 MHz to 100 MHz, type `50` MHz.
3. On the **Inputs/Lock** page, turn on **Create output file(s) using the 'Advanced' PLL parameters**.
4. On the **Clock switchover** page, turn on **Create an 'inclk1' input for a second input clock** and enter the high end of your lock range as the frequency for `inclk1`.

   For example, if your application requires a lock range of 50 MHz to 100 MHz, type `100` MHz.
5. Set the rest of the parameters in the remaining pages of the ALTPLL IP core parameter editor.
6. Compile your project and note the lock range shown in the PLL Summary report. If it is satisfactory, note all of the values for the PLL from this report, such as the `M` value, `N` value, charge pump current, loop filter resistance, and loop filter capacitance.
7. In the schematic editor, double-click the ALTPLL instance in your design to open the ALTPLL parameter editor.
8. On the **Clock switchover** page, turn off **Create an 'inclk1' input for a second input clock**.
9. Click **Finish** to update the PLL wrapper file.
10. In a text editor, open the PLL wrapper file. Modify all of the values for the parameters listed in step 6. Save the changes.

- If the wrapper file is in Verilog format, go to the **defparam** section.
- If the wrapper file is in VHDL HDL, go to the **Generic Map** section.

11. Compile your project.

12. Check the PLL Summary report to confirm that the PLL lock range meets your requirements. The modified PLL should have the desired lock range.

If your input clock frequency is too close to the end of the desired PLL lock range—for example the low end of the desired lock range is 50 MHz and the input clock frequency is 50 MHz, the PLL might not maintain lock when the input clock has jitter or the frequency drifts below 50 MHz. You may choose to expand your PLL lock range to ensure your expected input clock frequency is further from the end of the range. For this example, you can enter 45 MHz and 105 MHz to ensure that your target lock range of 50 MHz to 100 MHz is within the PLL lock range.

The Quartus Prime software prompts an error message if it is unable to implement your preferred lock range using this procedure. Therefore, you have to look into other options, such as PLL reconfiguration to support your input frequency range.

Programmable Bandwidth with Advanced Parameters

An advanced level of control is also possible for precise control of the PLL loop filter characteristics. This level allows you to explicitly select the following advanced parameters:

- Charge pump current (`charge_pump_current`)
- Loop filter resistance (`loop_filter_r`)
- Loop filter capacitance(`loop_filter_c`)

This option is intended for advanced users who know the exact details of their PLL configuration. You can use this option if you understand the parameters well enough to set them optimally. The files generated are not intended to be reused by the ALTPLL IP core parameter editor. After the ALTPLL IP core output files are specified using the advanced parameters, the Quartus Prime compiler cannot change them. For example, the compiler cannot perform optimization. Thus, your design cannot benefit from improved algorithms of the compiler. The Quartus Prime compiler cannot select better settings or change some settings that the ALTPLL IP core parameter editor finds to be incompatible with your design.

The parameter settings to generate output files using advanced PLL parameters are located on the **Inputs/Lock** page of the ALTPLL IP core parameter editor.

Turn on **Create output file(s) using the 'Advanced' PLL parameters** to enable the feature.

When you turn on this option, the generated output files contain all of the initial counter values used in the PLL. You can use these values for functional simulation in a third-party simulator.

These parameter settings create no additional top-level ports.

Related Information
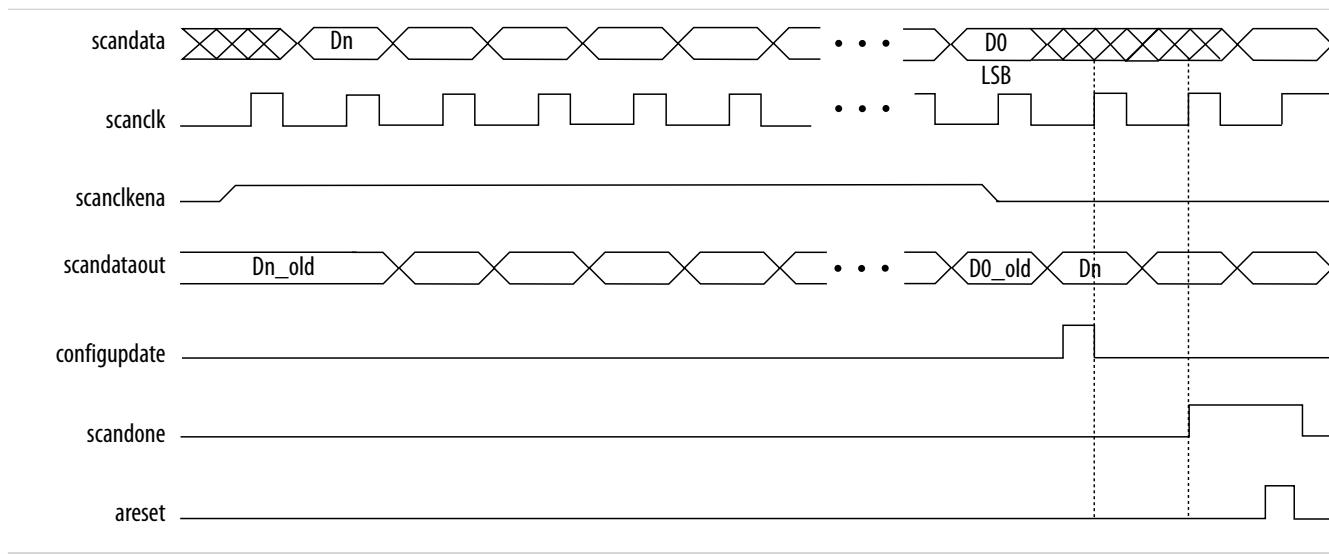
- **Programmable Bandwidth** on page 2-19
- **Charge Pump and Loop Filter** on page 4-6
  Provides more information about the PLL components to update PLL bandwidth in real time.
- **Programmable Bandwidth Parameter Settings** on page 6-2

PLL Dynamic Reconfiguration Implementation

To reconfigure the PLL counters, perform the following steps:

1. Assert the `scanclkena` signal at least one `scanclk` cycle prior to shifting in the first bit of `scandata` (`Dn`).
2. Shift the serial data (`scandata`) into the scan chain on the second rising edge of `scanclk`.
3. After all 144 bits have been scanned into the scan chain, deassert the `scanclkena` signal to prevent inadvertent shifting of bits in the scan chain.
4. Assert the `configupdate` signal for one `scanclk` cycle to update the PLL counters with the contents of the scan chain.
   The `scandone` signal goes high indicating that the PLL is being reconfigured. A falling edge indicates that the PLL counters have been updated with new settings.
5. Reset the PLL using the `areset` signal if you make any changes to the M, N, post-scale output C counters, or the I_{CP}, R, and C settings.
6. You can repeat steps 1 through 5 to reconfigure the PLL any number of times.

Figure 4-1: PLL Reconfiguration Scan Chain Functional Simulation

When reconfiguring the counter clock frequency, you cannot reconfigure the corresponding counter phase shift settings using the same interface. You can reconfigure phase shifts in real time using the dynamic phase shift reconfiguration interface. If you wish to keep the same nonzero phase shift setting (for example, 90°) on the clock output, you must reconfigure the phase shift after reconfiguring the counter clock frequency.

Related Information

PLL Reconfiguration on page 2-26

Post-Scale Counters (C0 to C4)

You can configure the multiply or divide values and duty cycle of the post-scale counters in real time. Each counter has an 8-bit high time setting and an 8-bit low time setting. The duty cycle is the ratio of output high or low time to the total cycle time, which is the sum of the two.

The post-scale counters have two control bits:

- `rbypass`—For bypassing the counter
- `rselodd`—For selecting the output clock duty cycle

When the `rbypass` bit is set to 1, it bypasses the counter, resulting in a division by one. When this bit is set to 0, the PLL computes the effective division of the VCO output frequency based on the high and low time counters. The PLL implements this duty cycle by transitioning the output clock from high-to-low on the rising edge of the VCO output clock.

For example, if the post-scale divide factor is 10, the high and low count values are set to 5 and 5 respectively, to achieve a 50–50% duty cycle. However, a 4 and 6 setting for the high and low count values, respectively, would produce an output clock with 40–60% duty cycle.

The `rselodd` bit indicates an odd divide factor for the VCO output frequency with a 50% duty cycle. The PLL implements this duty cycle by transitioning the output clock from high-to-low on a falling edge of the VCO output clock.

For example, if the post-scale divide factor is 3, the high and low time count values are 2 and 1 respectively, to achieve this division. This implies a 67%–33% duty cycle. If you need a 50%–50% duty cycle, you must set the `rselodd` control bit to 1 to achieve this duty cycle despite an odd division factor. When you set `rselodd` = 1, subtract 0.5 cycles from the high time and add 0.5 cycles to the low time.

The calculation for the example is shown as follows:

- High time count = 2 cycles
- Low time count = 1 cycle
- `rselodd` = 1 effectively equals:
  - High time count = 1.5 cycles
  - Low time count = 1.5 cycles
  - Duty cycle = (1.5/3)% high time count and (1.5/3)% low time count

Related Information
Programmable Duty Cycle on page 2-19

Scan Chain

The MAX 10 PLLs have a 144-bit scan chain.

Table 4-1: PLL Component Reprogramming Bits

Block Name	Number of Bits		
	Counter	Control Bit	Total
C4[6]	16	2[7]	18
C3	16	2[7]	18
C2	16	2[7]	18
C1	16	2[7]	18
C0	16	2[7]	18

---

[6] LSB bit for C4 low-count value is the first bit shifted into the scan chain.
[7] These two control bits include `rbypass`, for bypassing the counter, and `rselodd`, for selecting the output clock duty cycle.

Block Name	Number of Bits		
	Counter	Control Bit	Total
M	16	2[7]	18
N	16	2[7]	18
Charge Pump	9	0	9
Loop Filter[8]	9	0	9
Total number of bits			144

Figure 4-2: PLL Component Scan Chain Order
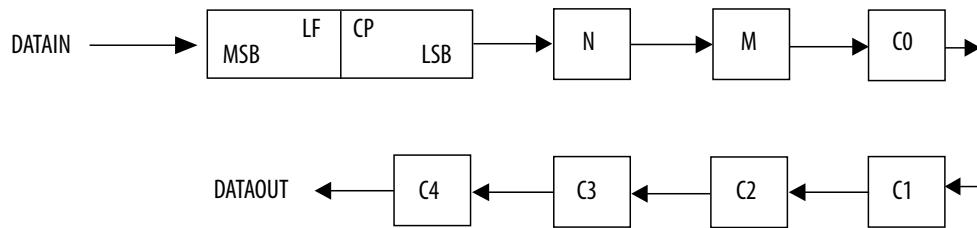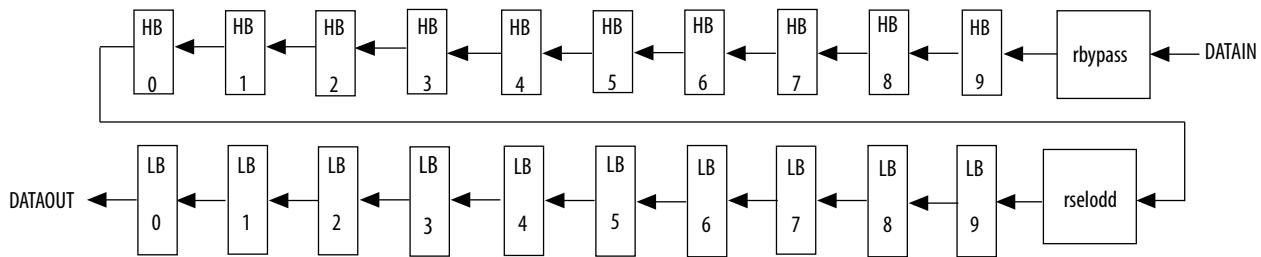
Figure 4-3: PLL Post-Scale Counter Scan Chain Bit Order

Charge Pump and Loop Filter

You can reconfigure the following settings to update the PLL bandwidth in real time:

- Charge pump (I_{CP})
- Loop filter resistor (R)
- Loop filter capacitor (C)

Table 4-2: Charge Pump Bit Control

CP[2]	CP[1]	CP[0]	Setting (Decimal)
0	0	0	0
0	0	1	1
0	1	1	3

---

[8] MSB bit for loop filter is the last bit shifted into the scan chain.

CP[2]	CP[1]	CP[0]	Setting (Decimal)
1	1	1	7

Table 4-3: Loop Filter Resistor Value Control

LFR[4]	LFR[3]	LFR[2]	LFR[1]	LFR[0]	Setting (Decimal)
0	0	0	0	0	0
0	0	0	1	1	3
0	0	1	0	0	4
0	1	0	0	0	8
1	0	0	0	0	16
1	0	0	1	1	19
1	0	1	0	0	20
1	1	0	0	0	24
1	1	0	1	1	27
1	1	1	0	0	28
1	1	1	1	0	30

Table 4-4: Loop Filter High Frequency Capacitor Control

LFC[1]	LFC[0]	Setting (Decimal)
0	0	0
0	1	1
1	1	3

Related Information

- **Programmable Bandwidth** on page 2-19
- **Programmable Bandwidth with Advanced Parameters** on page 4-3
- **Programmable Bandwidth Parameter Settings** on page 6-2

Bypassing PLL Counter

Bypassing a PLL counter results in a multiplification (M counter) or a division (N, C0 to C4 counters) factor of one.

Table 4-5: PLL Counter Settings

Description	PLL Scan Chain Bits [0..8] Settings								
	LSB								MSB
PLL counter bypassed	X	X	X	X	X	X	X	X	1[9]
PLL counter not bypassed	X	X	X	X	X	X	X	X	0[9]

To bypass any of the PLL counters, set the bypass bit to 1. The values on the other bits are ignored.
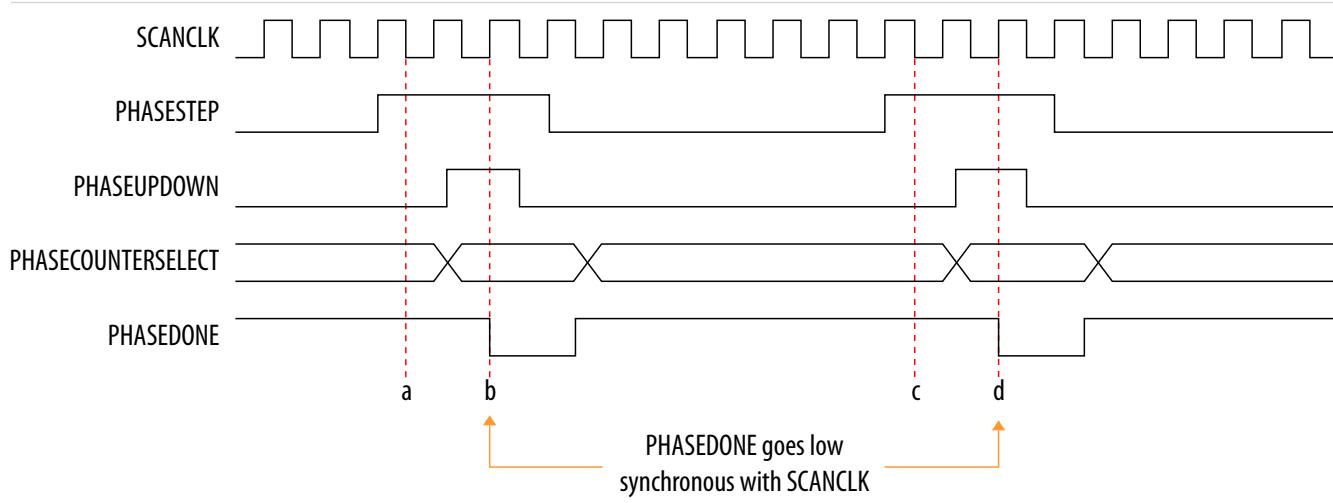
Dynamic Phase Configuration Implementation

To perform one dynamic phase shift step, perform the following steps:

1. Set PHASEUPDOWN and PHASECOUNTERSELECT as required.
2. Assert PHASESTEP for at least two SCANCLK cycles. Each PHASESTEP pulse allows one phase shift.
3. Deassert PHASESTEP after PHASEDONE goes low.
4. Wait for PHASEDONE to go high.
5. Repeat steps 1 through 4 as many times as required to perform multiple phase shifts.

PHASEUPDOWN and PHASECOUNTERSELECT signals are synchronous to SCANCLK and must meet the t_{su} and t_h requirements with respect to the SCANCLK edges.

You can repeat dynamic phase-shifting indefinitely. For example, in a design where the VCO frequency is set to 1,000 MHz and the output clock frequency is set to 100 MHz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180º, which is a phase shift of 5 ns.

Figure 4-4: Dynamic Phase Shift Timing Diagram

PHASEDONE goes low
synchronous with SCANCLK

The PHASESTEP signal is latched on the negative edge of SCANCLK (a,c) and must remain asserted for at least two SCANCLK cycles. Deassert PHASESTEP after PHASEDONE goes low.

---

[9] Bypass bit

On the second SCANCLK rising edge (b,d) after PHASESTEP is latched, the values of PHASEUPDOWN and PHASECOUNTERSELECT are latched. The PLL starts dynamic phase-shifting for the specified counters and in the indicated direction.

The PHASEDONE signal is deasserted synchronous to SCANCLK at the second rising edge (b,d) and remains low until the PLL finishes dynamic phase-shifting. Depending on the VCO and SCANCLK frequencies, PHASEDONE low time may be greater than or less than one SCANCLK cycle.

You can perform another dynamic phase-shift after the PHASEDONE signal goes from low to high. Each PHASESTEP pulse enables one phase shift. The PHASESTEP pulses must be at least one SCANCLK cycle apart.

Related Information

- **Programmable Phase Shift** on page 2-20
- **Dynamic Phase Configuration Parameter Settings** on page 6-4
  Provides more information about the ALTPLL IP core parameter settings in the Quartus Prime software.
- **ALTPLL_RECONFIG Parameters** on page 7-1
  Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Quartus Prime software.
- **PLL Dynamic Reconfiguration Parameter Settings** on page 6-4
  Provides more information about the ALTPLL IP core parameter settings in the Quartus Prime software.
- **ALTPLL_RECONFIG Parameters** on page 7-1
  Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Quartus Prime software.

Dynamic Phase Configuration Counter Selection

Table 4-6: Phase Counter Select Mapping

PLL Counter Selection	PHASECOUNTERSELECT [2]	[1]	[0]
All output counters	0	0	0
M counter	0	0	1
C0 counter	0	1	0
C1 counter	0	1	1
C2 counter	1	0	0
C3 counter	1	0	1
C4 counter	1	1	0

Related Information
Programmable Phase Shift on page 2-20

Dynamic Phase Configuration with Advanced Parameters

The finest phase shift step resolution you can get in the ALTPLL IP core is 1/8 of the VCO period. If the VCO frequency is at the lower end of the supported VCO range, the phase shift step resolution might be larger than preferred for your design.

You can modify your phase shift resolution using the dynamic phase reconfiguration feature of the PLL. If you want to modify the phase shift resolution without the dynamic phase reconfiguration feature enabled, perform the following steps:

1. Create an ALTPLL instance. Make sure you specify the speed grade of your target device and the PLL type.
2. On the **PLL Reconfiguration** page, turn on **Create optional inputs for dynamic phase reconfiguration** and **Enable phase shift step resolution**.
3. On the **Output Clocks** page, set your desired phase shift for each required output clock. Note all the internal PLL settings shown.
4. On the **Bandwidth/SS** page, click **More Details** to see the internal PLL settings. Note all of the settings shown.
5. On the **Inputs/Lock** page, turn on **Create output file(s) using the 'Advanced' PLL Parameters**.
6. Return to the **PLL Reconfiguration** page and turn off **Create Optional Inputs for Dynamic Phase Reconfiguration**.
7. Click **Finish** to generate the PLL instantiation file(s).

   When using Advanced Parameters, the PLL wrapper file (*<ALTPLL_instantiation_name>*.**v** or *<ALTPLL_instantiation_name>*.**vhd**) is written in a format that allows you to identify the PLL parameters. The parameters are listed in the **Generic Map** section of the VHDL file, or in the `defparam` section of the Verilog file.
8. Open your PLL instantiation wrapper file and locate either the **Generic Map** or the `defparam` section.
9. Modify the settings to match the settings that you noted in steps 3 and 4.
10. Save the PLL instantiation wrapper file and compile your design.
11. Verify that the output clock frequencies and phases are correct in the PLL Usage report located under the Resource section of the Fitter folder in the Compilation Report.

By using this technique, you can apply valid PLL parameters as provided by the ALTPLL IP core parameter editor to optimize the settings for your design.

Alternatively, you can leave the dynamic phase reconfiguration option enabled and tie the relevant input ports—`phasecounterselect[3..0]`, `phaseupdown`, `phasestep`, and `scanclk`—to constants, if you prefer not to manually edit the PLL wrapper file using the Advanced PLL Parameters option.

Related Information
Programmable Phase Shift on page 2-20

ALTPLL_RECONFIG IP Core

The ALTPLL_RECONFIG IP core implements reconfiguration logic to facilitate dynamic real-time reconfiguration of PLLs. You can use the IP core to update the output clock frequency, PLL bandwidth, and phase shifts in real time, without reconfiguring the entire FPGA.

Use the ALTPLL_RECONFIG IP core in designs that must support dynamic changes in the frequency and phase shift of clocks and other frequency signals. The IP core is also useful in prototyping environments because it allows you to sweep PLL output frequencies and dynamically adjust the output clock phase. You can also adjust the clock-to-output (t_{CO}) delays in real-time by changing the output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings. This operation requires dynamic phase-shifting.

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating,
  upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Obtaining the Resource Utilization Report

For details about the resource usage and performance of the ALTPLL_RECONFIG IP core, refer to the
compilation reports in the Quartus Prime software.

To view the compilation reports for the ALTPLL_RECONFIG IP core in the Quartus Prime software,
follow these steps:

1. On the Processing menu, click **Start Compilation** to run a full compilation.
2. After compiling the design, on the Processing menu, click **Compilation Report**.
3. In the Table of Contents browser, expand the Fitter folder by clicking the "+" icon.
4. Under **Fitter**, expand **Resource section**, and select **Resource Usage Summary** to view the resource
   usage information.
5. Under **Fitter**, expand **Resource section**, and select **Resource Utilization by Entity** to view the
   resource utilization information.

Internal Oscillator IP Core

The Internal Oscillator IP core specifies the internal oscillator frequencies for the devices.

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating,
  upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

✉ **Subscribe**   💬 **Send Feedback**

ALTCLKCTRL Parameters

Table 5-1: ALTCLKCTRL IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Parameter	Value	Decription
How do you want to use the ALTCLKCTRL	**For global clock**, or **For external path**	Specify the ALTCLKCTRL buffering mode. You can select from the following modes: • **For global clock**—Allows a clock signal to reach all parts of the chip with the same amount of skew; you can select input port `clkselect` to switch between the four clock inputs. • **For external path**—Represents the clock path from the outputs of the PLL to the dedicated clock output pins; only one clock output is accepted.
How many clock inputs would you like?	**1**, **2**, **3**, or **4**	Specify the number of input clock sources for the clock control block. You can specify up to four clock inputs. You can change the number of clock inputs only if you choose **For global clock** option.
Create 'ena' port to enable or disable the clock network driven by this buffer	On or Off	Turn on this option if you want to create an active high clock enable signal to enable or disable the clock network.

ISO 9001:2008 Registered

ALTERA®

Parameter	Value	Decription
Ensure glitch-free switchover implementation	On or Off	Turn on this option to implement a glitch-free switchover when you use multiple clock inputs.

You must ensure the currently selected clock is running before switching to another source. If the selected clock is not running, the glitch-free switchover implementation will not be able to switch to the new clock source.

By default, the `clkselect` port is set to `00`. A clock must be applied to `inclk0x` for the values on the `clkselect` ports to be read. |

Related Information

- **Global Clock Control Block** on page 2-4
- **Global Clock Network Power Down** on page 2-6
- **Clock Enable Signals** on page 2-7
- **Guideline: Clock Enable Signals** on page 3-1

ALTCLKCTRL Ports and Signals

Table 5-2: ALTCLKCTRL Input Ports for MAX 10 Devices

Port Name	Condition	Description
`clkselect[]`	Optional	Input that dynamically selects the clock source to drive the clock network that is driven by the clock buffer.

Input port `[1 DOWNTO 0]` wide.

If omitted, the default is GND.

If this signal is connected, only the global clock network can be driven by this clock control block.

The following list shows the signal selection for the binary value:

- `00`—`inclk[0]`
- `01`—`inclk[1]` |
| `ena` | Optional | Clock enable of the clock buffer.

If omitted, the default value is V_{CC}. |

Port Name	Condition	Description
inclk[]	Required	Clock input of the clock buffer. Input port [1 DOWNTO 0] wide. You can specify up to two clock inputs, inclk[1..0]. Clock pins, clock outputs from the PLL, and core signals can drive the inclk[] port. Multiple clock inputs are only supported for the global clock networks.

Table 5-3: ALTCLKCTRL Output Ports for MAX 10 Devices

Port Name	Condition	Description
outclk	Required	Output of the clock buffer.

Related Information

ALTPLL Parameters

The following tables list the IP core parameters applicable to MAX 10 devices.

Operation Modes Parameter Settings

You can set the operation mode for PLL in the **General/Modes** page of the ALTPLL IP core parameter editor.

Table 6-1: Operation Mode Parameter Editor Settings

Parameter	Value	Description
Which device speed grade will you be using?	**Any**, **7**, or **8**	Specify the speed grade if you are not already using a device with the fastest speed. The lower the number, the faster the speed grade.
What is the frequency of the inclock0 input?	—	Specify the frequency of the input clock signal.
Use the feedback path inside the PLL	**In normal mode**, **In source-synchronous compensation mode**, **In zero-delay buffer mode**, or **With no compensation**	Specify which operation mode to use. For source-synchronous mode and zero-delay buffer mode, you must make PLL Compensation assignments using the Assignment Editor in addition to setting the appropriate mode in the IP core. The assignment allows you to specify an output pin as a compensation target for a PLL in zero-delay buffer mode, or to specify an input pin or group of input pins as compensation targets for a PLL in source-synchronous mode.
Which output clock will be compensated for?	C0, C1, C2, C3, or C4	Specify which PLL output port to compensate. The drop down list contains all output clock ports for the selected device. The correct output clock selection depends on the operation mode that you select. For example, for normal mode, select the core output clock. For zero-delay buffer mode, select the external output clock.

Related Information
Clock Feedback Modes on page 2-14

PLL Control Signals Parameter Settings

The parameter settings for the control signals are located on the **Inputs/Lock** page of the ALTPLL IP core parameter editor.

Turn on the control signal you want to create from the options available.

Related Information
PLL Control Signals on page 2-13

Programmable Bandwidth Parameter Settings

You can configure the bandwidth of the ALTPLL IP core on the **Bandwidth/SS** page of the ALTPLL IP core parameter editor.

Table 6-2: Bandwidth Configuration Parameter Editor Settings

Parameter	Value	Description
Auto	—	The ALTPLL parameter editor chooses the best possible bandwidth values to achieve the desired PLL settings. In some cases, you can get a bandwidth value outside the **Low** and **High** preset range. You can use the programmable bandwidth feature with the clock switchover feature to get the PLL output settings that you desire. You must set the bandwidth to **Auto** if you want to enable the spread-spectrum feature.
Preset	**Low**	PLL with a low bandwidth has better jitter rejection but a slower lock time.
	Medium	PLL with a medium bandwidth has a balance between lock time and jitter rejection.
	High	PLL with a high bandwidth has a faster lock time but tracks more jitter.

The table on the right in the **Bandwidth/SS** page shows the values of the following components:

- Charge pump current
- Loop filter resistance
- Loop filter capacitance
- M counter

These parameter settings create no additional top-level ports.

Related Information

- Programmable Bandwidth on page 2-19
- Programmable Bandwidth with Advanced Parameters on page 4-3
- Charge Pump and Loop Filter on page 4-6

Clock Switchover Parameter Settings

The parameter settings for clock switchover feature are located on the **Clock switchover** page of the ALTPLL IP core parameter editor.

Table 6-3: Clock Switchover Parameter Editor Settings

Parameter	Value	Description
Create an 'inclk1' input for a second input clock	On or Off	Turn on this option to enable the switchover feature. The `inclk0` signal is by default the primary input clock signal of the ALTPLL IP core.
Create a 'clkswitch' input to manually select between the input clocks	—	Select this option for manual clock switchover mode.
Allow PLL to automatically control the switching between input clocks	—	Select this option for automatic clock switchover mode. The automatic switchover is initiated during loss of lock or when the `inclk0` signal stops toggling.
Create a 'clkswitch' input to dynamically control the switching between input clocks	On or Off	Turn on this option for automatic clock switchover with manual override mode. The automatic switchover is initiated during loss of lock or when the `clkswitch` signal is asserted.
Perform the input clock switchover after (number) input clock cycles	On or Off	Turn on this option to specify the number of clock cycles to wait before the PLL performs the clock switchover. The allowed number of clock cycles to wait is device-dependent.
Create an 'activeclock' output to indicate the input clock being used	On or Off	Turn on this option to monitor which input clock signal is driving the PLL. When the current clock signal is `inclk0`, the `activeclock` signal is low. When the current clock signal is `inclk1`, the `activeclock` signal is high.
Create a 'clkbad' output for each input clock	On or Off	Turn on this option to monitor when the input clock signal has stopped toggling. The `clkbad0` signal monitors the `inclk0` signal. The `clkbad1` signal monitors the `inclk1` signal. The `clkbad0` signal goes high when the `inclk0` signal stops toggling. The `clkbad1` signal goes high when the `inclk1` signal stops toggling. The `clkbad` signals remain low when the input clock signals are toggling.

Related Information

- **Clock Switchover** on page 2-22

- **Guideline: Clock Switchover** on page 3-4

PLL Dynamic Reconfiguration Parameter Settings

The parameter settings for the normal dynamic reconfiguration scheme are located on the **PLL Reconfiguration** page of the ALTPLL IP core parameter editor.

Table 6-4: PLL Dynamic Reconfiguration Parameter Editor Settings

Parameter	Value	Description
Create optional inputs for dynamic reconfiguration	On or Off	Turn on this option to enable all the PLL reconfiguration ports for this instantiation—scanclk, scanclkena, scandata, scandone, scandataout, and configupdate.
Initial Configuration File	—	Specify the location of the configuration file that is used to initialize the ALTPLL_RECONFIG IP core.
Additional Configuration File(s)	—	Specify additional configuration file. This file might contain additional settings for the PLL, or might be used to initialize the ALTPLL_RECONFIG IP core.

Related Information

- **PLL Reconfiguration** on page 2-26
- **Dynamic Phase Configuration Implementation** on page 4-8

Dynamic Phase Configuration Parameter Settings

The parameter settings to enable the dynamic phase configuration feature are located on the **PLL Reconfiguration** page of the ALTPLL IP core parameter editor.

Table 6-5: Dynamic Phase Configuration Parameter Editor Settings

Parameter	Value	Description
Create optional inputs for dynamic phase reconfiguration	On or Off	Turn on this option to enable the dynamic phase configuration feature. The following ports are created: - phasecounterselect[2..0] - phaseupdown - phasestep - scanclk - phasedone
Enable phase shift step resolution edit	On or Off	Turn on this option to modify the value for **Phase shift step resolution(ps)** for each individual PLL output clock on the **Output Clocks** page. By default, the finest phase shift resolution value is 1/8 of the VCO period. If the VCO frequency is at the lower end of the supported VCO range, the phase shift resolution might be larger than preferred for your design. Use this option to fine tune the phase shift step resolution.

Related Information

- **Programmable Phase Shift** on page 2-20
- **Dynamic Phase Configuration Implementation** on page 4-8

Output Clocks Parameter Settings

The **Output Clocks** page of the ALTPLL parameter editor contains the parameter settings of the clock output signals. You can configure the c0, c1, c2, c3, and c4 clock output signals of the ALTPLL IP core.

Each option has the following two columns:

- Requested settings—The settings that you want to implement.
- Actual settings—The settings closest values that can be implemented in the PLL circuit to best approximate the requested settings.

Use the values in the actual settings column as a guide to adjust the requested settings. If the requested settings for one of the output clocks cannot be approximated, the ALTPLL IP core parameter editor produces a warning message at the top of every page.

Table 6-6: Output Clocks Parameter Editor Settings

Parameter	Value	Description
Use this clock	On or Off	Turn on this option to generate an output clock port in your ALTPLL instance.
		The output clock port that is to be compensated for is enabled by default. It cannot be disabled, unless you select a different output clock port to be compensated for.
Enter output clock frequency	—	Specify the frequency of the output clock signal.
Enter output clock parameters	—	Specify the the output clock parameters instead of the frequency.
Clock multiplication factor	—	Specify the clock multiplication factor of the signal.
Clock division factor	—	Specify the clock division factor of the signal.
Clock phase shift	—	Set the programmable phase shift for an output clock signals.
		The smallest phase shift is 1/8 of VCO period. For degree increments, the maximum step size is 45 degrees. You can set smaller steps using the **Clock multiplication factor** and **Clock division factor** options.
		For example, if the post-scale counter is 32, the smallest phase shift step is 0.1°. The up and down buttons let you cycle through phase shift values. Alternatively, you can enter a number in the phase shift field manually instead of using the buttons.

Parameter	Value	Description
Clock duty cycle (%)	—	Set the duty cycle of the output clock signal.
Per Clock Feasibility Indicators	—	Indicate output clocks that contain unachievable settings. The output clock name in red is the name of the clock with unachievable settings. The clock listed in green has no settings issues, and the grayed-out names are the unselected output clocks. You must adjust the requested settings for the affected output clocks to resolve the warning messages.

The ALTPLL IP core parameter editor calculates the simplest fraction, and displays it in the actual settings column. You can use the copy button to copy values from the actual settings to the requested settings.

Figure 6-1: PLL Output Clock Frequency

$$Output\ clock\ frequency = Input\ clock\ frequency \times \frac{Multiplication\ factor}{Division\ factor}$$

For example, if the input clock frequency is 100 MHz, and the requested multiplication and division factors are 205 and 1025 respectively, the output clock frequency is calculated as $100 \times 205/1025 = 20$ MHz. The actual settings reflect the simplest fraction—the actual multiplication factor is 1, and the actual division factor is 5.

ALTPLL Ports and Signals

Table 6-7: ALTPLL Input Ports for MAX 10 Devices

Port Name[10]	Condition	Description
areset	Optional	Resets all counters to initial values, including the GATE_LOCK_COUNTER parameter.
clkswitch	Optional	The control input port to dynamically toggle between clock input ports (inclk0 and inclk1 ports), or to manually override the automatic clock switchover. You should create the clkswitch port if only the inclk1 port is created.
configupdate	Optional	Dynamic full PLL reconfiguration.

---

[10] Replace brackets, [], in the port name with integer to get the exact name. For example, inclk0 and inclk1.

Port Name[10]	Condition	Description
inclk[]	Required	The clock inputs that drive the clock network.
		If more than one `inclk[]` port is created, you must use the `clkselect` port to specify which clock is used. The `inclk0` port must always be connected; connect other clock inputs if switching is necessary.
		A dedicated clock pin or PLL output clock can drive this port.
pfdena	Optional	Enables the phase frequency detector (PFD).
		When the PFD is disabled, the PLL continues to operate regardless of the input clock. Because the PLL output clock frequency does not change for some time, you can use the `pfdena` port as a shutdown or cleanup function when a reliable input clock is no longer available.
phasecounterselect[]	Optional	Specifies counter select. You can use the `phasecounterselect[2..0]` bits to select either the `M` or one of the `C` counters for phase adjustment. One address map to select all `C` counters. This signal is registered in the PLL on the rising edge of `SCANCLK`.
phasestep	Optional	Specifies dynamic phase shifting. Logic high enables dynamic phase shifting.
phaseupdown	Optional	Specifies dynamic phase shift direction. `1`= UP, `0` = DOWN. Signal is registered in the PLL on the rising edge of `SCANCLK`.
scanclk	Optional	Input clock port for the serial scan chain.
		Free-running clock from core used in combination with `PHASESTEP` to enable or disable dynamic phase shifting. Shared with `SCANCLK` for dynamic reconfiguration.
scanclkena	Optional	Clock enable port for the serial scan chain.
scandata	Optional	Contains the data for the serial scan chain.

[10] Replace brackets, [], in the port name with integer to get the exact name. For example, `inclk0` and `inclk1`.

Table 6-8: ALTPLL Output Ports for MAX 10 Devices

Port Name[11]	Condition	Description
activeclock	Optional	Specifies which clock is the primary reference clock when the clock switchover circuit initiates. If the inclk0 is in use, the activeclock port goes low. If the inclk1 is in use, the activeclock port goes high. You can set the PLL to automatically initiate the clock switchover when the primary reference clock is not toggling correctly, or you can manually initiate the clock switchover using the clkswitch input port.
c[]	Required	The clock output of the PLL.
clkbad[]	Optional	clkbad1 and clkbad0 ports check for input clock toggling. If the inclk0 port stops toggling, the clkbad0 port goes high. If the inclk1 port stops toggling, the clkbad1 port goes high.

---

[11] Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0 and c1).

Port Name[11]	Condition	Description
locked	Optional	This output port acts as an indicator when the PLL has reached phase-locked. The locked port stays high as long as the PLL is locked, and stays low when the PLL is out-of-lock. The number of cycles needed to gate the locked signal is based on the PLL input clock. The gated-lock circuitry is clocked by the PLL input clock. The maximum lock time for the PLL is provided in the *MAX 10 Device Datasheet*. Take the maximum lock time of the PLL and divide it by the period of the PLL input clock. The result is the number of clock cycles needed to gate the locked signal. The lock signal is an asynchronous output of the PLL. The PLL lock signal is derived from the reference clock and feedback clock feeding the phase frequency detector (PFD) as follows: • Reference clock = Input Clock/N • Feedback clock = VCO/M The PLL asserts the locked port when the phases and frequencies of the reference clock and feedback clock are the same or within the lock circuit tolerance. When the difference between the two clock signals goes beyond the lock circuit tolerance, the PLL loses lock.
phasedone	Optional	This output port indicates that dynamic phase reconfiguration is completed. When phasedone signal is asserted, it indicates to core logic that the phase adjustment is complete and PLL is ready to act on a possible second adjustment pulse. This signal asserts based on internal PLL timing and deasserts on rising edge of SCANCLK.
scandataout	Optional	The data output for the serial scan chain. You can use the scandataout port to determine when PLL reconfiguration completes. The last output is cleared when reconfiguration completes.
scandone	Optional	This output port indicates that the scan chain write operation is initiated. The scandone port goes high when the scan chain write operation initiates, and goes low when the scan chain write operation completes.

---

[11] Replace the brackets, [ ], in the port name with an integer to get the exact name (for example, c0 and c1).

Related Information

ALTPLL_RECONFIG Parameters

Table 7-1: ALTPLL_RECONFIG IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Page	Parameter	Value	Description
Parameter Settings	**Currently Selected Device Family**	—	Specifies the chosen device family.
	Which scan chain type will you be using?	—	The scan chain is a serial shift register chain that is used to store settings. It acts like a cache. When you assert the `reconfig` signal, the PLL is reconfigured with the values in the cache. The type of scan chain must follow the type of PLL to be reconfigured. The scan chain type has a default value of **Top/ Bottom**.
	Do you want to specify the initial value of the scan chain?	**No, leave it blank,** **Yes, use this file for the content data**	Specifies the initial value of the scan chain. Select **No, leave it blank** to not specify a file or select **Yes, use this file for the content data** to browse for a **.hex** or **.mif** file. The option to initialize from a ROM is not available. However, you can choose to add ports to write to the scan chain from an external ROM during runtime by turning on **Add ports to write to the scan chain from external ROM during run time**.
	Add ports to write to the scan chain from external ROM during run time	On, Off	Turn on this option to take advantage of cycling multiple configuration files, which are stored in external ROMs during user mode.

ISO 9001:2008 Registered

Page	Parameter	Value	Description
EDA	**Simulation Libraries**	—	Specifies the libraries for functional simulation.
	Generate netlist	On, Off	Turn on this option to generate synthesis area and timing estimation netlist.
Summary	—	—	Specifies the types of files to be generated. A gray checkmark indicates a file that is automatically generated; an unchecked check box indicates an optional file. Choose from the following types of files: • AHDL Include file (*<function name>*.**inc**) • VHDL component declaration file (*<function name>*.**cmp**) • Quartus Prime symbol file (*<function name>*.**bsf**) • Instantiation template file (*<function name>*_**inst.v** or *<function name>*_**inst.vhd**) • Verilog HDL black box file (*<function name>*_**bb.v**) If the **Generate netlist** option is turned on, the file for that netlist is also available (*<function name>*_**syn.v**).

Related Information

ALTPLL_RECONFIG Ports and Signals

Table 7-2: ALTPLL_RECONFIG Input Ports for MAX 10 Devices

Port Name	Condition	Description
clock	Required	Clock input for loading individual parameters. This signal also clocks the PLL during reconfiguration. The clock input port must be connected to a valid clock. Refer to the *MAX 10 Device Datasheet* for the clock f_{MAX}.

Port Name	Condition	Description
reset	Required	Asynchronous reset input to the IP core. Altera recommends that you reset this IP core before first use to guarantee that it is in a valid state. However, it does power up in the reset state. This port must be connected.
data_in[]	Optional	Data input that provides parameter value when writing parameters. This 9-bit input port provides the data to be written to the scan cache during a write operation. The bit width of the counter parameter to be written determines the number of bits of data_in[] that are read into the cache. For example, the low bit count of the C0 counter is 8-bit wide, so data_in[7..0] is read to the correct cache location. The bypass mode for the C0 counter is 1-bit wide, so data_in[0] is read for the value of this parameter. If omitted, the default value is 0.
counter_type[]	Optional	Specifies the counter type. An input port in the form of a 4-bit bus that selects which counter type should be selected for the corresponding operation (read, write, or reconfig). Refer to the counter_type[3..0] settings table for the mapping between the counter_type value and the physical counter to be set.
counter_param[]	Optional	Specifies the parameter for the value specified in the counter_type port. An input port in the form of a 3-bit bus that selects which parameter for the given counter type should be updated. The mapping to each parameter type and the corresponding parameter bit-width are defined in the counter_param[3..0] settings table.

Port Name	Condition	Description
read_param	Optional	Reads the parameter specified with the `counter_type` and `counter_param` ports from cache and fed to the `data_out[]` port. When asserted, the `read_param` signal indicates that the scan cache should be read and fed to `data_out[]`. The bit location of the scan cache and the number of bits read and sent to `data_out[]` depend on the `counter_type` and `counter_param` values. The `read_param` signal is sampled at the rising clock edge. If the `read_param` signal is asserted, the parameter value is read from the cache. Assert the `read_param` signal for 1 clock cycle only to prevent the parameter from being read twice. The `busy` signal is asserted on the rising clock edge following the assertion of the `read_param` signal. While the parameter is read, the `busy` signal remains asserted. After the `busy` signal is deasserted, the value on `data_out[]` is valid and the next parameter can be loaded. While the `busy` signal is asserted, the value on `data_out[]` is not valid. When the `read_param` signal is asserted, the `busy` signal is only asserted on the following rising edge of the clock and not on the same clock cycle as the `read_param` signal.
write_param	Optional	Writes the parameter specified with the `counter_type` and `counter_param` ports to the cache with the value specified on the `data_in[]` port. When asserted, the `write_param` signal indicates that the value on `data_in[]` should be written to the parameter specified by `counter_type[]` and `counter_param[]`. The number of bits read from the `data_in[]` port depends on the parameter. The `write_param` signal is sampled at the rising clock edge. If the `write_param` signal is asserted, the parameter value is written to the cache. Assert the `write_param` signal for 1 clock cycle only to prevent the parameter from being written twice. The `busy` signal is asserted on the rising clock edge following the assertion of the `write_param` signal. While the parameter is being written, the `busy` signal remains asserted and input to `data_in[]` is ignored. After the `busy` signal is deasserted, the next parameter can be written. When the `write_param` signal is asserted, the `busy` signal is only asserted on the following rising edge of the clock. The `busy` signal is not asserted on the same clock cycle as the `write_param` signal.

Port Name	Condition	Description
reconfig	Required	Specifies that the PLL should be reconfigured with the PLL settings specified in the current cache.
		When asserted, the reconfig signal indicates that the PLL should be reconfigured with the values in the cache. The reconfig signal is sampled at the rising clock edge. If the reconfig signal is asserted, the cached settings are loaded in the PLL. Assert the reconfig signal for 1 clock cycle only to prevent reloading the PLL configuration. The busy signal is asserted on the rising clock edge following the assertion of the reconfig signal. While the PLL is being loaded, the busy signal remains asserted. After the busy signal is deasserted, the parameter values can be modified again.
		During and after reconfiguration, the scan chain data cache remains unchanged. This allows you to easily create a new set of reconfiguration settings using only one parameter.
		If write_param has not been asserted since the previous assertion of reconfig, the entire scan chain is shifted in to the PLL again.
		When the reconfig signal is asserted, the busy signal is only asserted on the following rising edge of the clock. The busy signal is not asserted on the same clock cycle as the reconfig signal.
pll_areset_in	Optional	Input signal indicating that the PLL should be reset.
		When asserted, the pll_areset_in signal indicates the PLL IP core should be reset. This port defaults to 0 if left unconnected. When using the ALTPLL_RECONFIG IP core in a design, you cannot reset the PLL in any other way. You must use this IP core port to manually reset the PLL.
pll_scandone	Optional	Input port for the ALTPLL_RECONFIG IP core. This port is driven by the PLL's scandone output signal and determines when the PLL is reconfigured.
pll_scandataout	Required	Input port driven by the scandataout signal from the ALTPLL IP core. Use this port to read the current configuration of the ALTPLL IP core. This input port holds the ALTPLL scan data output from the dynamically reconfigurable bits. The pll_scandataout port must be connected to the scandataout port of the PLL. The activity on this port can only be observed when the reconfig signal is asserted.

Table 7-3: ALTPLL_RECONFIG Output Ports for MAX 10 Devices

Port Name	Condition	Description
data_out[]	Optional	Data read from the cache when read_param is asserted. This 9-bit output bus provides the parameter data to the user. When the read_param signal is asserted, the values on counter_type[] and counter_param[] determine the parameter value that is loaded from cache and driven on the data_out[] bus. When the IP core deasserts the busy signal, the appropriate bits of the bus (for example,[0] or [3..0]) hold a valid value.
busy	Optional	Indicates that the PLL is reading or writing a parameter to the cache, or is configuring the PLL. While the busy signal is asserted, no parameters can be read or written, and no reconfiguration can be initiated. Changes to the IP core can be made only when the busy signal is not asserted. The signal goes high when the read_param, write_param, or reconfig input port is asserted, and remains high until the specified operation is complete. In the case of a reconfiguration operation, the busy signal remains high until the pll_areset signal is asserted and then deasserted.
pll_areset	Required	Drives the areset port on the PLL to be reconfigured. The pll_areset port must be connected to the areset port of the ALTPLL IP core for the reconfiguration to function correctly. This signal is active high. The pll_areset is asserted when pll_areset_in is asserted, or, after reconfiguration, at the next rising clock edge after the scandone signal goes high. If you use the ALTPLL_RECONFIG IP core, use the pll_areset output port to drive the PLL areset port.
pll_configupdate	Optional	Drives the configupdate port on the PLL to be reconfigured. When asserted, the pll_configupdate port loads selected data to PLL configuration latches. The signal is asserted after the final data bit is sent out.
pll_scanclk	Required	Drives the scanclk port on the PLL to be reconfigured. For information about the maximum scanclk frequency for the various devices, refer to the respective device handbook.

Port Name	Condition	Description
pll_scanclkena	Optional	This port acts as a clock enable for the scanclk port on the PLL to be reconfigured. Reconfiguration begins on the first rising edge of pll_scanclk after pll_scanclkena assertion. On the first falling edge of pll_scanclk, after the deassertion of the pll_scanclkena signal, the IP core stops scanning data to the PLL.
pll_scandata	Required	Drives the scandata port on the PLL to be reconfigured. This output port from the IP core holds the scan data input to the PLL for the dynamically reconfigurable bits. The pll_scandata port sends scandata to the PLL. Any activity on this port can only be observed when the reconfig signal is asserted.

ALTPLL_RECONFIG Counter Settings

Table 7-4: counter_type[3..0] Settings for MAX 10 Devices

Counter Selection	Binary	Decimal
N	0000	0
M	0001	1
CP/LF	0010	2
VCO	0011	3
C0	0100	4
C1	0101	5
C2	0110	6
C3	0111	7
C4	1000	8
Illegal value	1001	9
Illegal value	1010	10
Illegal value	1011	11
Illegal value	1100	12
Illegal value	1101	13
Illegal value	1110	14
Illegal value	1111	15

Table 7-5: `counter_param[2..0]` Settings for MAX 10 Devices

Counter Type	Counter Param	Binary	Decimal	Width (bits)
Regular counters (`C0 - C4`)	High count	000	0	8
	Low count	001	1	8
	Bypass	100	4	1
	Mode (odd/even division)	101	5	1
CP/LF	Charge pump unused	101	5	5
	Charge pump current	000	0	3
	Loop filter unused	100	4	1
	Loop filter resistor	001	1	5
	Loop filter capacitance	010	2	2
VCO	VCO post scale	000	0	1
`M/N` counters	High count	000	0	8
	Low count	001	1	8
	Bypass	100	4	1
	Mode (odd/even division)	101	5	1
	Nominal count	111	7	9

For even nominal count, the counter bits are automatically set as follows:

- `high_count` = Nominalcount$/2$
- `low_count` = Nominalcount$/2$

For odd nominal count, the counter bits are automatically set as follows:

- `high_count` = (Nominalcount + 1)$/2$
- `low_count` = Nominalcount - `high_count`
- odd/even division bit = 1

For nominal count = 1, bypass bit = 1.

Internal Oscillator Parameters

Table 8-1: Internal Oscillator IP Core Parameters for MAX 10 Devices

This table lists the IP core parameters applicable to MAX 10 devices.

Parameter	Value	Decription
Clock Frequency	55, 116	Specify the clock frequency for simulation. If not specified, the default value is 55 MHz.

Internal Oscillator Ports and Signals

Table 8-2: Internal Oscillator Input Port for MAX 10 Devices

Port Name	Condition	Description
`oscena`	Required	Input control signal to turn on or turn off the internal oscillator.

Table 8-3: Internal Oscillator Output Port for MAX 10 Devices

Port Name	Condition	Description
`clkout`	Optional	Output clock from the internal oscillator.

**ISO
9001:2008
Registered**

ALTERA ®

Document Revision History for MAX 10 Clocking and PLL User Guide

Date	Version	Changes
November 2015	2015.11.02	• Removed the topics about the IP catalog and parameter editor, generating IP cores, and the files generated by the IP core, and added a link to *Introduction to Altera IP Cores*. • Changed instances of *Quartus II* to *Quartus Prime*.
June 2015	2015.06.12	Added connectivity restriction guideline to the PLL design considerations.
May 2015	2015.05.04	Rearranged the fine resolution phase shift equation.
December 2014	2014.12.15	• Corrected the statement that if you do not use the dedicated clock input pins for clock input, you can also use them as general-purpose input or output pins. • Added description in Internal Oscillator Architecture and Features to state that the internal ring oscillator operates up to 232 MHz and this frequency is not accessible. • Added connectivity restrictions guideline for internal oscillator. • Added Internal Oscillator IP Core parameter: Clock Frequency. • Moved Internal Oscillator Frequencies table from Internal Oscillator Architecture and Features chapter to MAX 10 FPGA Device Datasheet.
September 2014	2014.09.22	Initial release.

ISO 9001:2008 Registered

ALTERA®

MAX 10 General Purpose I/O User Guide

Contents

The MAX® 10 general purpose I/O (GPIO) system consists of the I/O elements (IOE) and the Altera GPIO Lite IP core.

- The IOEs contain bidirectional I/O buffers and I/O registers located in I/O banks around the periphery of the device.
- The Altera GPIO Lite IP core supports the GPIO components and features, including double data rate I/O (DDIO), delay chains, I/O buffers, control signals, and clocking.

Related Information

- **MAX 10 I/O Architecture and Features** on page 2-1
  Provides information about the architecture and features of the I/Os in MAX 10 devices.
- **MAX 10 I/O Design Considerations** on page 3-1
  Provides I/O design guidelines for MAX 10 Devices.
- **MAX 10 I/O Implementation Guides** on page 4-1
  Provides guides to implement I/Os in MAX 10 Devices.
- **Altera GPIO Lite IP Core References** on page 5-1
  Lists the parameters and signals of Altera GPIO Lite IP core for MAX 10 Devices.
- **MAX 10 General Purpose I/O User Guide Archives** on page 6-1
  Provides a list of user guides for previous versions of the Altera GPIO Lite IP core.

MAX 10 Devices I/O Resources Per Package

Table 1-1: Package Plan for MAX 10 Single Power Supply Devices

Device	Type	Package		
		M153 153-pin MBGA	U169 169-pin UBGA	E144 144-pin EQFP
	Size	8 mm × 8 mm	11 mm × 11 mm	22 mm × 22 mm
	Ball Pitch	0.5 mm	0.8 mm	0.5 mm
10M02		112	130	101
10M04		112	130	101

now part of Intel

Device	Package			
	Type	M153 153-pin MBGA	U169 169-pin UBGA	E144 144-pin EQFP
	Size	8 mm × 8 mm	11 mm × 11 mm	22 mm × 22 mm
	Ball Pitch	0.5 mm	0.8 mm	0.5 mm
10M08		112	130	101
10M16		—	130	101
10M25		—	—	101
10M40		—	—	101
10M50		—	—	101
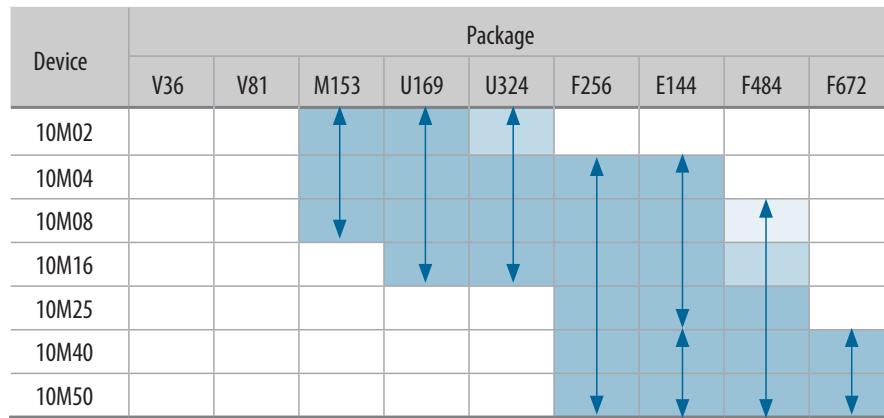
Table 1-2: Package Plan for MAX 10 Dual Power Supply Devices—Preliminary

Device	Package					
	Type	V36 36-pin WLCSP	V81 81-pin WLCSP	U324 324-pin UBGA	F256 256-pin FBGA	F484 484-pin FBGA	F672 672-pin FBGA
	Size	3 mm × 3 mm	4 mm × 4 mm	15 mm × 15 mm	17 mm × 17 mm	23 mm × 23 mm	27 mm × 27 mm
	Ball Pitch	0.4 mm	0.4 mm	0.8 mm	1.0 mm	1.0 mm	1.0 mm
10M02		27	—	160	—	—	—
10M04		—	—	246	178	—	—
10M08		—	56	246	178	250	—
10M16		—	—	246	178	320	—
10M25		—	—	—	178	360	—
10M40		—	—	—	178	360	500
10M50		—	—	—	178	360	500

MAX 10 I/O Vertical Migration Support

Figure 1-1: Migration Capability Across MAX 10 Devices

- The arrows indicate the migration paths. The devices included in each vertical migration path are shaded. Some packages have several migration paths. Devices with lesser I/O resources in the same path have lighter shades.
- To achieve the full I/O migration across product lines in the same migration path, restrict I/Os usage to match the product line with the lowest I/O count.

Device	Package								
	V36	V81	M153	U169	U324	F256	E144	F484	F672
10M02									
10M04									
10M08									
10M16									
10M25									
10M40									
10M50									

Note: To verify the pin migration compatibility, use the Pin Migration View window in the Quartus® Prime software Pin Planner.

Related Information

Verifying Pin Migration Compatibility on page 4-4

UG-M10GPIO ✉ **Subscribe** 💬 **Send Feedback**

The I/O system of MAX 10 devices support various I/O standards. In the MAX 10 devices, the I/O pins are located in I/O banks at the periphery of the devices. The I/O pins and I/O buffers have several programmable features.

Related Information

MAX 10 I/O Standards Support

MAX 10 devices support a wide range of I/O standards, including single-ended, voltage-referenced single-ended, and differential I/O standards.

Table 2-1: Supported I/O Standards in MAX 10 Devices

The voltage-referenced I/O standards are not supported in the following I/O banks of these device packages:

- All I/O banks of V36 package of 10M02.
- All I/O banks of V81 package of 10M08.
- Banks 1A and 1B of E144 package of 10M50.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
3.3 V LVTTL/3.3 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-B
3.0 V LVTTL/3.0 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-B
2.5 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-5
1.8 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-7
1.5 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-11

ISO 9001:2008 Registered

ALTERA
now part of Intel

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
1.2 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-12
3.0 V PCI	Single-ended	All	Yes	Yes	General purpose	PCI Rev. 2.2
3.3 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
2.5 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
1.8 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
1.5 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
SSTL-2 Class I	Voltage-referenced	All	Yes	Yes	DDR1	JESD8-9B
SSTL-2 Class II	Voltage-referenced	All	Yes	Yes	DDR1	JESD8-9B
SSTL-18 Class I	Voltage-referenced	All	Yes	Yes	DDR2	JESD8-15
SSTL-18 Class II	Voltage-referenced	All	Yes	Yes	DDR2	JESD8-15
SSTL-15 Class I	Voltage-referenced	All	Yes	Yes	DDR3	—
SSTL-15 Class II	Voltage-referenced	All	Yes	Yes	DDR3	—
SSTL-15[1]	Voltage-referenced	All	Yes	Yes	DDR3	JESD79-3D
SSTL-135[1]	Voltage-referenced	All	Yes	Yes	DDR3L	—
1.8 V HSTL Class I	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, and RLDRAM 2	JESD8-6
1.8 V HSTL Class II	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, and RLDRAM 2	JESD8-6
1.5 V HSTL Class I	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, QDR II, and RLDRAM 2	JESD8-6

---

[1] Available in MAX 10 16, 25, 40, and 50 devices only.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
1.5 V HSTL Class II	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, QDR II, and RLDRAM 2	JESD8-6
1.2 V HSTL Class I	Voltage-referenced	All	Yes	Yes	General purpose	JESD8-16A
1.2 V HSTL Class II	Voltage-referenced	All	Yes	Yes	General purpose	JESD8-16A
HSUL-12[1]	Voltage-referenced	All	Yes	Yes	LPDDR2	—
Differential SSTL-2 Class I and II	Differential	All	Yes[2]	Yes[3]	DDR1	JESD8-9B
Differential SSTL-18 Class I and Class II	Differential	All	Yes[2]	Yes[3]	DDR2	JESD8-15
Differential SSTL-15 Class I and Class II	Differential	All	Yes[2]	Yes[3]	DDR3	—
Differential SSTL-15	Differential	All	Yes[2]	Yes[3]	DDR3	JESD79-3D
Differential SSTL-135	Differential	All	Yes[2]	Yes[3]	DDR3L	—
Differential 1.8 V HSTL Class I and Class II	Differential	All	Yes[2]	Yes[3]	DDR II+, QDR II+, and RLDRAM 2	JESD8-6
Differential 1.5 V HSTL Class I and Class II	Differential	All	Yes[2]	Yes[3]	DDR II+, QDR II+, QDR II, and RLDRAM 2	JESD8-6
Differential 1.2 V HSTL Class I and Class II	Differential	All	Yes[2]	Yes[3]	General purpose	JESD8-16A
Differential HSUL-12	Differential	All	Yes[2]	Yes[3]	LPDDR2	—
LVDS (dedicated)[4]	Differential	All	Yes	Yes	—	ANSI/TIA/EIA-644

[2] The inputs treat differential inputs as two single-ended inputs and decode only one of them.
[3] The outputs use two single-ended output buffers with the second output buffer programmed as inverted.
[4] You can use dedicated LVDS transmitters only on the bottom I/O banks. You can use LVDS receivers on all I/O banks.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
LVDS (emulated, external resistors)	Differential	All	—	Yes	—	ANSI/TIA/EIA-644
Mini-LVDS (dedicated)[4]	Differential	All	—	Yes	—	—
Mini-LVDS (emulated, external resistor)	Differential	Dual supply devices	—	Yes	—	—
RSDS (dedicated)[4]	Differential	All	—	Yes	—	—
RSDS (emulated, external resistor, 1R)	Differential	Dual supply devices	—	Yes	—	—
RSDS (emulated, external resistors, 3R)	Differential	All	—	Yes	—	—
PPDS (dedicated)[4]	Differential	Dual supply devices	—	Yes	—	—
PPDS (emulated, external resistor)	Differential	Dual supply devices	—	Yes	—	—
LVPECL	Differential	All	Yes	—	—	—
Bus LVDS	Differential	All	Yes	Yes[5]	—	—
TMDS	Differential	Dual supply devices	Yes	—	—	—
Sub-LVDS	Differential	Dual supply devices	Yes	Yes[6]	—	—
SLVS	Differential	Dual supply devices	Yes	Yes[7]	—	—

[5] The outputs use two single-ended output buffers with the second output buffer programmed as inverted. A single series resistor is required.

[6] Requires external termination resistors.

[7] The outputs uses two single-ended output buffers as emulated differential outputs. Requires external termination resistors.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
HiSpi	Differential	Dual supply devices	Yes	—	—	—

Related Information

- **MAX 10 I/O Buffers** on page 2-13
  Provides more information about available I/O buffer types and supported I/O standards.
- **LVDS Transmitter I/O Termination Schemes, MAX 10 High-Speed LVDS I/O User Guide**

MAX 10 I/O Standards Voltage and Pin Support

Table 2-2: MAX 10 I/O Standards Voltage Levels and Pin Support

Note: The I/O standards that each pin type supports depends on the I/O standards that the pin's I/O bank supports. For example, only the bottom I/O banks support the LVDS (dedicated) I/O standard. You can use the LVDS (dedicated) I/O standard for the `PLL_CLKOUT` pin only if the pin is available in your device's bottom I/O banks. To determine the pin's I/O bank locations for your device, check your device's pin out file.

I/O Standard	V_{CCIO} (V)		V_{REF} (V)	Pin Type Support				
	Input	Output		PLL_ CLKOUT	MEM_CLK	CLK	DQS	User I/O
3.3 V LVTTL/ 3.3 V LVCMOS	3.3/3.0/ 2.5	3.3	—	Yes	Yes	Yes	Yes	Yes
3.0 V LVTTL/ 3.0 V LVCMOS	3.0/2.5	3.0	—	Yes	Yes	Yes	Yes	Yes
2.5 V LVCMOS	3.0/2.5	2.5	—	Yes	Yes	Yes	Yes	Yes
1.8 V LVCMOS	1.8/1.5	1.8	—	Yes	Yes	Yes	Yes	Yes
1.5 V LVCMOS	1.8/1.5	1.5	—	Yes	Yes	Yes	Yes	Yes
1.2 V LVCMOS	1.2	1.2	—	Yes	Yes	Yes	Yes	Yes
3.0 V PCI	3.0	3.0	—	Yes	Yes	Yes	Yes	Yes
3.3 V Schmitt Trigger	3.3	—	—	—	—	Yes	Yes[8]	Yes
2.5 V Schmitt Trigger	2.5	—	—	—	—	Yes	Yes[8]	Yes
1.8 V Schmitt Trigger	1.8	—	—	—	—	Yes	Yes[8]	Yes

---

[8] Bidirectional— use Schmitt Trigger input with LVTTL output.

I/O Standard	V_{CCIO} (V)		V_{REF} (V)	Pin Type Support				
	Input	Output		PLL_CLKOUT	MEM_CLK	CLK	DQS	User I/O
1.5 V Schmitt Trigger	1.5	—	—	—	—	Yes	Yes[8]	Yes
SSTL-2 Class I	2.5	2.5	1.25	Yes	Yes	Yes	Yes	Yes
SSTL-2 Class II	2.5	2.5	1.25	Yes	Yes	Yes	Yes	Yes
SSTL-18 Class I	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
SSTL-18 Class II	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
SSTL-15 Class I	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
SSTL-15 Class II	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
SSTL-15	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
SSTL-135	1.35	1.35	0.675	Yes	Yes	Yes	Yes	Yes
1.8 V HSTL Class I	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
1.8 V HSTL Class II	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
1.5 V HSTL Class I	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
1.5 V HSTL Class II	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
1.2 V HSTL Class I	1.2	1.2	0.6	Yes	Yes	Yes	Yes	Yes
1.2 V HSTL Class II	1.2	1.2	0.6	Yes	Yes	Yes	Yes	Yes
HSUL-12	1.2	1.2	0.6	Yes	Yes	Yes	Yes	Yes
Differential SSTL-2 Class I and II	—	2.5	—	Yes	Yes	—	Yes	—
	2.5	—	1.25	—	—	Yes	Yes	—
Differential SSTL-18 Class I and Class II	—	1.8	—	Yes	Yes	—	Yes	—
	1.8	—	0.9	—	—	Yes	Yes	—
Differential SSTL-15 Class I and Class II	—	1.5	—	Yes	Yes	—	Yes	—
	1.5	—	0.75	—	—	Yes	Yes	—
Differential SSTL-15	—	1.5	—	Yes	Yes	—	Yes	—
	1.5	—	0.75	—	—	Yes	Yes	—

I/O Standard	V_CCIO (V)		V_REF (V)	Pin Type Support				
	Input	Output		PLL_ CLKOUT	MEM_CLK	CLK	DQS	User I/O
Differential SSTL-135	—	1.35	—	Yes	Yes	—	Yes	—
	1.35	—	0.675	—	—	Yes	Yes	—
Differential 1.8 V HSTL Class I and Class II	—	1.8	—	Yes	Yes	—	Yes	—
	1.8	—	0.9	—	—	Yes	Yes	—
Differential 1.5 V HSTL Class I and Class II	—	1.5	—	Yes	Yes	—	Yes	—
	1.5	—	0.75	—	—	Yes	Yes	—
Differential 1.2 V HSTL Class I and Class II	—	1.2	—	Yes	Yes	—	Yes	—
	1.2	—	0.6	—	—	Yes	Yes	—
Differential HSUL-12	—	1.2	—	Yes	Yes	—	Yes	—
	1.2	—	0.6	—	—	Yes	Yes	—
LVDS (dedicated)	2.5	2.5	—	Yes	Yes	Yes	—	Yes
LVDS (emulated, external resistors)	—	2.5	—	Yes	Yes	—	—	Yes
Mini-LVDS (dedicated)	—	2.5	—	Yes	Yes	—	—	Yes
Mini-LVDS (emulated, external resistor)	—	2.5	—	Yes	Yes	—	—	Yes
RSDS (dedicated)	—	2.5	—	Yes	Yes	—	—	Yes
RSDS (emulated, external resistor, 1R)	—	2.5	—	Yes	Yes	—	—	Yes
RSDS (emulated, external resistors, 3R)	—	2.5	—	Yes	Yes	—	—	Yes
PPDS (dedicated)	—	2.5	—	Yes	Yes	—	—	Yes
PPDS (emulated, external resistor)	—	2.5	—	Yes	Yes	—	—	Yes
LVPECL	2.5	—	—	—	—	Yes	—	—
Bus LVDS	2.5	2.5	—	—	—	—	—	Yes
TMDS	2.5	—	—	—	—	Yes	—	Yes
Sub-LVDS	2.5	1.8	—	Yes	Yes	Yes	—	Yes

I/O Standard	V$_{CCIO}$ (V)		V$_{REF}$ (V)	Pin Type Support				
	Input	Output		PLL_CLKOUT	MEM_CLK	CLK	DQS	User I/O
SLVS	2.5	2.5	—	Yes	Yes	Yes	—	Yes
HiSpi	2.5	—	—	—	—	Yes	—	Yes

Related Information

- **MAX 10 Device Pin-Out Files**
- **MAX 10 I/O Standards Support** on page 2-1
- **MAX 10 I/O Banks Locations** on page 2-10
- **MAX 10 LVDS SERDES I/O Standards Support**
- **MAX 10 High-Speed LVDS I/O Location**

MAX 10 I/O Elements

The MAX 10 I/O elements (IOEs) contain a bidirectional I/O buffer and five registers for registering input, output, output-enable signals, and complete embedded bidirectional single data rate (SDR) and double data rate (DDR) transfer.

The I/O buffers are grouped into groups of four I/O modules per I/O bank:
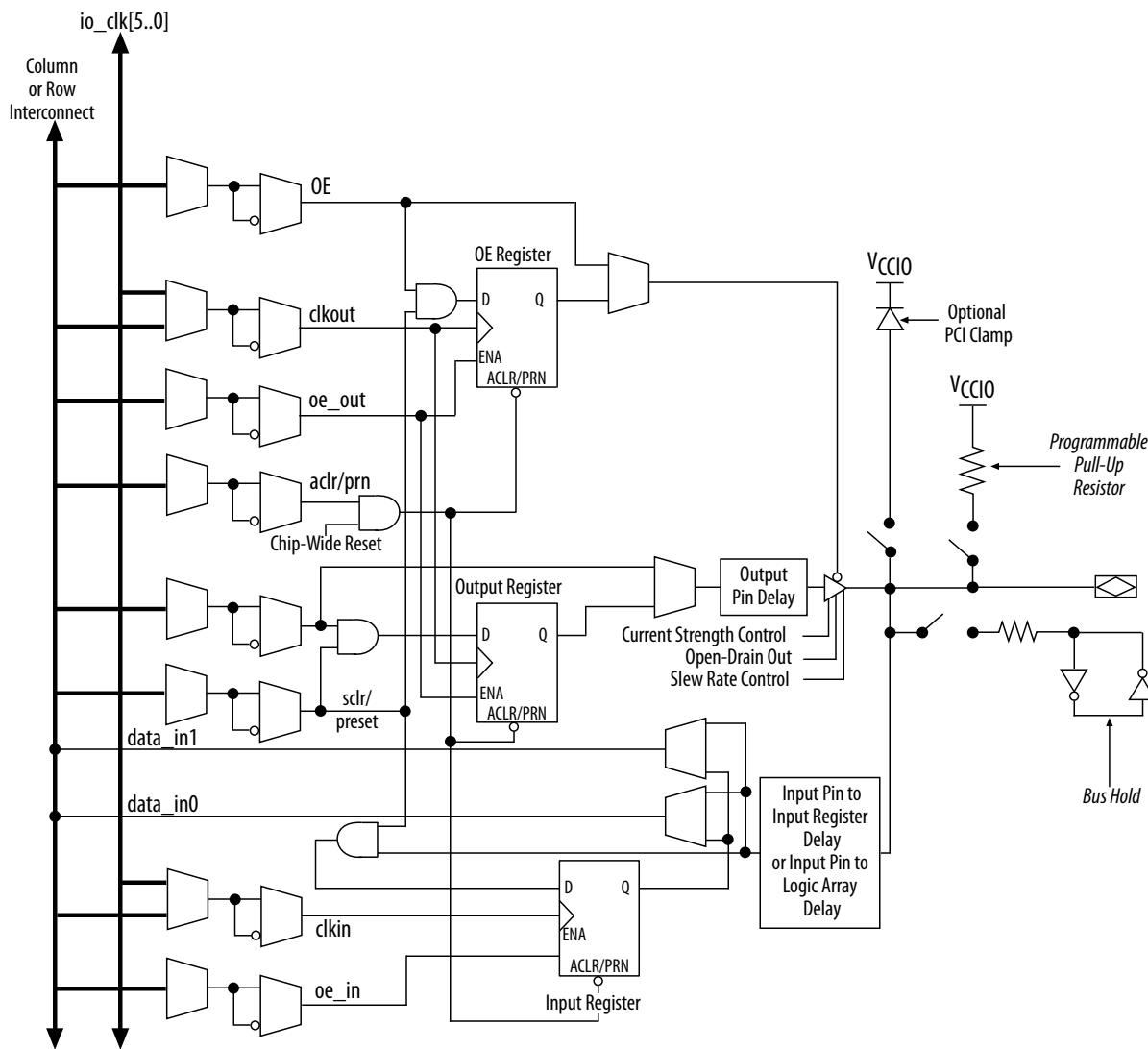
- The MAX 10 devices share the user I/O pins with the VREF, RUP, RDN, CLKPIN, PLLCLKOUT, configuration, and test pins.
- Schmitt Trigger input buffer is available in all I/O buffers.

Each IOE contains one input register, two output registers, and two output-enable (OE) registers:

- The two output registers and two OE registers are used for DDR applications.
- You can use the input registers for fast setup times and output registers for fast clock-to-output times.
- You can use the OE registers for fast clock-to-output enable times.

You can use the IOEs for input, output, or bidirectional data paths. The I/O pins support various single-ended and differential I/O standards.

Figure 2-1: IOE Structure in Bidirectional Configuration

Related Information

MAX 10 Power Management User Guide

Provides more information about the I/O buffers in different power cycles and hot socketing.

MAX 10 I/O Banks Architecture

The I/O elements are located in a group of four modules per I/O bank:

- High speed DDR3 I/O banks—supports various I/O standards and protocols including DDR3. These I/O banks are available only on the right side of the device.
- High speed I/O banks—supports various I/O standards and protocols except DDR3. These I/O banks are available on the top, left, and bottom sides of the device.
- Low speed I/O banks—lower speeds I/O banks that are located at the top left side of the device.

For more information about I/O pins support, refer to the pinout files for your device.

Related Information

MAX 10 Device Pin-Out Files

MAX 10 I/O Banks Locations

The I/O banks are located at the periphery of the device.

For more details about the modular I/O banks available in each device package, refer to the relevant device pin-out file.

Figure 2-2: I/O Banks for MAX 10 02 Devices—Preliminary
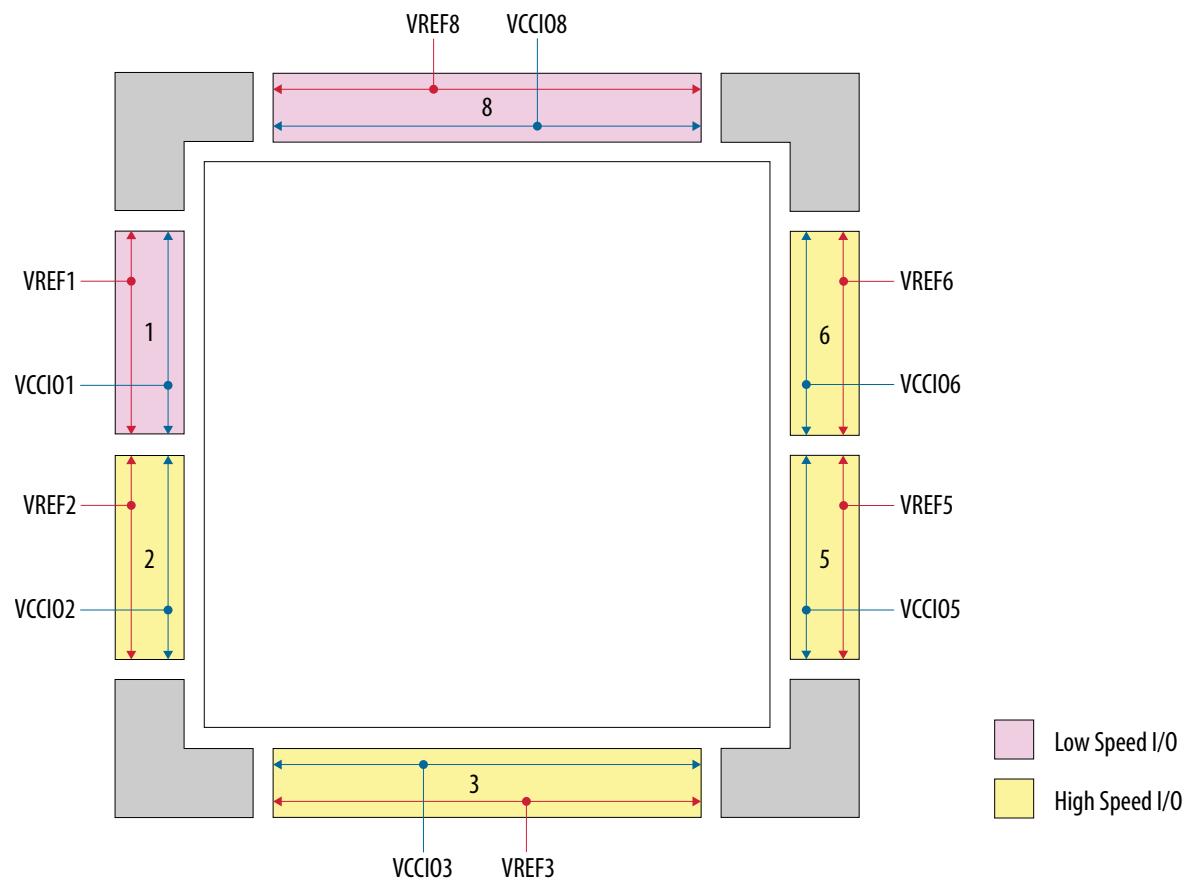
Figure 2-3: I/O Banks for MAX 10 04 and 08 Devices—Preliminary
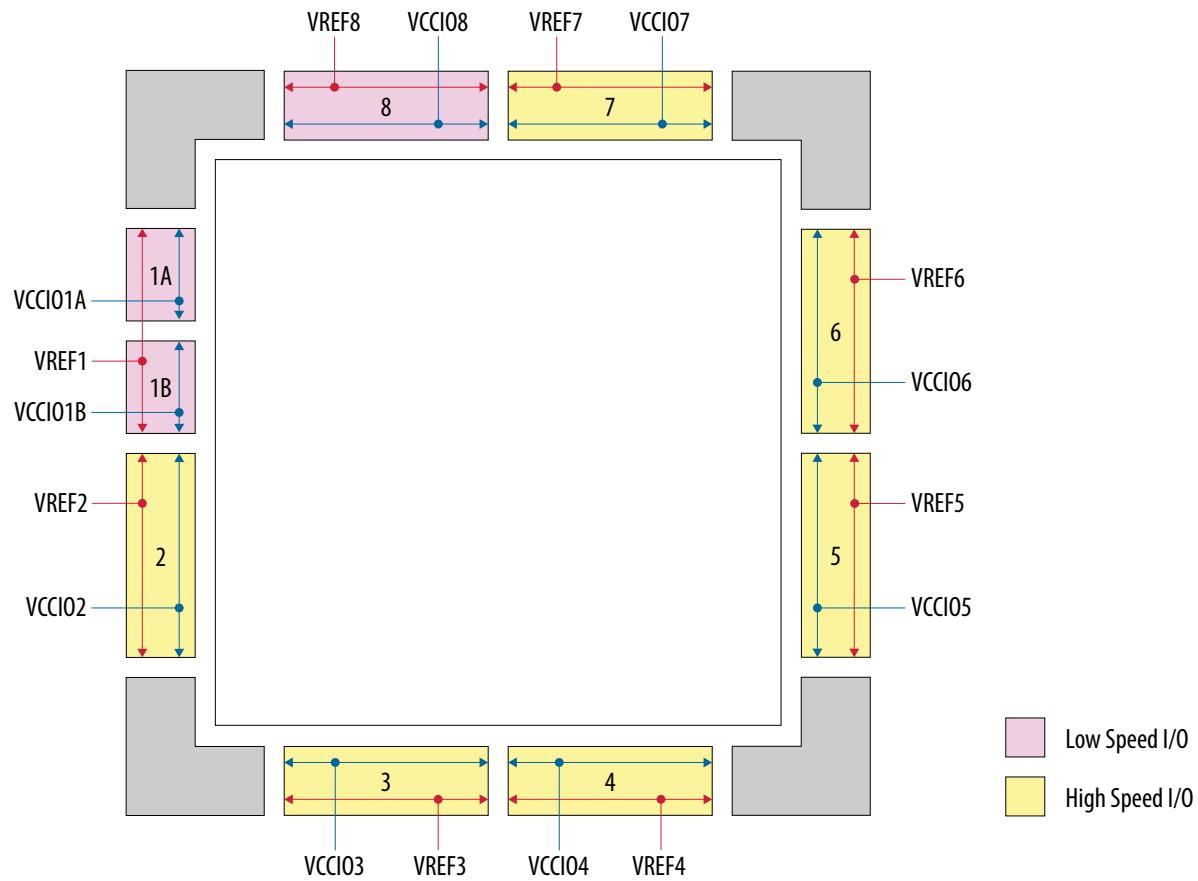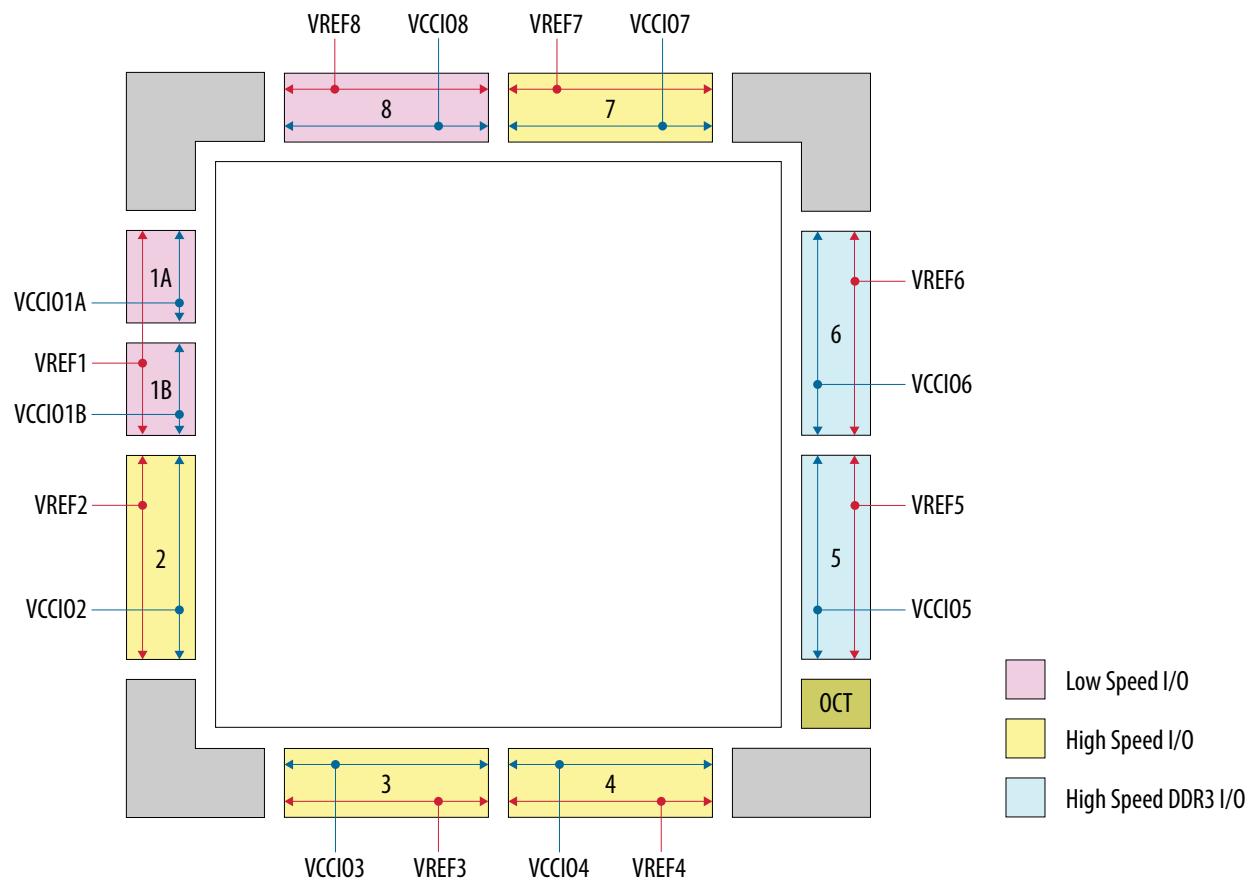
Figure 2-4: I/O Banks for MAX 10 16, 25, 40, and 50 Devices—Preliminary

Related Information

- **MAX 10 Device Pin-Out Files**
- **High-Speed I/O Specifications**
  Provides the performance information for different I/O standards in the low-speed and high-speed I/O banks.

MAX 10 I/O Buffers

The general purpose I/Os (GPIOs) in MAX 10 devices consist of LVDS I/O and DDR I/O buffers.

Table 2-3: Types of GPIO Buffers in MAX 10 Devices

LVDS I/O Buffers	DDR I/O Buffers
• Support differential and single-ended I/O standards. • Available only on I/O banks at the bottom side of the device. • For LVDS, the bottom I/O banks support LVDS transmitter, emulated LVDS transmitter, and LVDS receiver buffers.	• Support differential and single-ended I/O standards. • Available on I/O banks at the left, right, and top sides of the device. • For LVDS, the DDR I/O buffers support only LVDS receiver and emulated LVDS transmitter buffers. • For DDR, only the DDR I/O buffers on the right side of the device supports DDR3 external memory interfaces. DDR3 support is only available for MAX 10 16, 25, 40, and 50 devices.

Related Information

- **MAX 10 I/O Standards Support** on page 2-1
- **LVDS Transmitter I/O Termination Schemes, MAX 10 High-Speed LVDS I/O User Guide**

Schmitt-Trigger Input Buffer

The MAX 10 devices feature selectable Schmitt trigger input buffer on all I/O banks.

The Schmitt trigger input buffer has similar V_{IL} and V_{IH} as the LVTTL I/O standard but with better noise immunity. The Schmitt trigger input buffers are used as default input buffers during configuration mode.

Related Information
MAX 10 Device Datasheet

Programmable I/O Buffer Features

The MAX 10 I/O buffers support a range of programmable features. These features increase the flexibility of I/O utilization and provide an alternative to reduce the usage of external discrete components such as a pull-up resistor and a diode.

Table 2-4: Summary of Supported MAX 10 Programmable I/O Buffer Features and Settings

Feature	Setting	Condition	Assignment Name	Supported I/O Standards
Open Drain	On, Off (default)	To enable this feature, use the `OPNDRN` primitive.	—	• 3.0 V and 3.3 V LVTTL • 1.2 V, 1.5 V, 1.8 V, 2.5 V, 3.0 V, and 3.3 V LVCMOS • SSTL-2, SSTL-18, SSTL-15, and SSTL-135 • 1.2 V, 1.5 V, and 1.8 V HSTL • HSUL-12 • 3.0 V PCI
Bus-Hold	On, Off (default)	Disabled if you use the weak pull-up resistor feature.	Enable Bus-Hold Circuitry	
Pull-up Resistor	On, Off (default)	Disabled if you use the bus-hold feature.	Weak Pull-Up Resistor	
Slew Rate Control	0 (Slow), 1 (Medium), 2 (Fast). Default is 2.	Disabled if you use OCT.	Slew Rate	• 3.0 V LVTTL • 1.2 V, 1.5 V, 1.8 V, 2.5 V, and 3.0 V LVCMOS • SSTL-2, SSTL-18, and SSTL-15 • 1.2 V, 1.5 V, and 1.8 V HSTL • Differential SSTL-2, Differential SSTL-18, and Differential SSTL-15 • Differential 1.2 V, 1.5 V, and 1.8 V HSTL
PCI Clamp Diode	On (default for input pins), Off (default for output pins, except 3.0 V PCI)	—	PCI I/O	• 3.0 V and 3.3 V LVTTL • 2.5 V, 3.0 V, and 3.3 V LVCMOS • 3.0 V PCI • 2.5 V, 3.0 V, and 3.3 V Schmitt Trigger
Pre-Emphasis	0 (disabled), 1 (enabled). Default is 1.	—	Programmable Pre-emphasis	• LVDS • RSDS • PPDS • Mini-LVDS
Differential Output Voltage	0 (low), 1 (medium), 2 (high). Default is 2.	—	Programmable Differential Output Voltage (V_{OD})	

Programmable Open Drain

The optional open-drain output for each I/O pin is equivalent to an open collector output. If it is configured as an open drain, the logic value of the output is either high-Z or logic low.

Use an external resistor to pull the signal to a logic high.

Programmable Bus Hold

Each I/O pin provides an optional bus-hold feature that is active only after configuration. When the device enters user mode, the bus-hold circuit captures the value that is present on the pin by the end of the configuration.

The bus-hold circuitry holds this pin state until the next input signal is present. Because of this, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated.

For each I/O pin, you can individually specify that the bus-hold circuitry pulls non-driven pins away from the input threshold voltage—where noise can cause unintended high-frequency switching. To prevent over-driving signals, the bus-hold circuitry drives the voltage level of the I/O pin lower than the V_{CCIO} level.

If you enable the bus-hold feature, you cannot use the programmable pull-up option. To configure the I/O pin for differential signals, disable the bus-hold feature.

Programmable Pull-Up Resistor

Each I/O pin provides an optional programmable pull-up resistor during user mode. The pull-up resistor weakly holds the I/O to the V_{CCIO} level.

If you enable the weak pull-up resistor, you cannot use the bus-hold feature.

Programmable Current Strength

You can use the programmable current strength to mitigate the effects of high signal attenuation that is caused by a long transmission line or a legacy backplane.

Table 2-5: Programmable Current Strength Settings for MAX 10 Devices

The output buffer for each MAX 10 device I/O pin has a programmable current strength control for the I/O standards listed in this table.

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA) (Default setting in bold)
3.3 V LVCMOS	**2**
3.3 V LVTTL	**8**, 4
3.0 V LVTTL/3.0 V LVCMOS	16, **12**, 8, 4
2.5 V LVTTL/2.5 V LVCMOS	16, **12**, 8, 4
1.8 V LVTTL/1.8 V LVCMOS	16, **12**, 10, 8, 6, 4, 2
1.5 V LVCMOS	16, **12**, 10, 8, 6, 4, 2
1.2 V LVCMOS	12, 10, **8**, 6, 4, 2
SSTL-2 Class I	12, **8**
SSTL-2 Class II	**16**

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA) (Default setting in bold)
SSTL-18 Class I	12, 10, **8**
SSTL-18 Class II	**16**, 12
SSTL-15 Class I	12, 10, **8**
SSTL-15 Class II	**16**
1.8 V HSTL Class I	12, 10, **8**
1.8 V HSTL Class II	**16**
1.5 V HSTL Class I	12, 10, **8**
1.5 V HSTL Class II	**16**
1.2 V HSTL Class I	12, 10, **8**
1.2 V HSTL Class II	**14**
BLVDS	**16**, 12, 8
SLVS	**16**, 12, 8
Sub-LVDS	**12**, 8, 4

Note: Altera recommends that you perform IBIS or SPICE simulations to determine the best current strength setting for your specific application.

Programmable Output Slew Rate Control

You have the option of three settings for programmable slew rate control—0, 1, and 2 with 2 as the default setting. Setting 0 is the slow slew rate and 2 is the fast slew rate.

- Fast slew rate—provides high-speed transitions for high-performance systems.
- Slow slew rate—reduces system noise and crosstalk but adds a nominal delay to the rising and falling edges.

Table 2-6: Programmable Output Slew Rate Control for MAX 10 Devices

This table lists the single-ended I/O standards and current strength settings that support programmable output slew rate control. For I/O standards and current strength settings that do not support programmable slew rate control, the default slew rate setting is 2 (fast slew rate).

I/O Standard	I_{OH} / I_{OL} Current Strength Supporting Slew Rate Control
3.0 V LVTTL/3.0 V LVCMOS	16, 12, 8
2.5 V LVTTL/2.5 V LVCMOS	16, 12, 8
1.8 V LVTTL/1.8 V LVCMOS	16, 12, 8
1.5 V LVCMOS	16, 12, 10, 8
1.2 V LVCMOS	12, 10, 8
SSTL-2 Class I	12, 8

I/O Standard	I$_{OH}$ / I$_{OL}$ Current Strength Supporting Slew Rate Control
SSTL-2 Class II	16
SSTL-18 Class I	12, 10, 8
SSTL-18 Class II	16, 12
SSTL-15 Class I	12, 10, 8
SSTL-15 Class II	16
1.8 V HSTL Class I	12, 10, 8
1.8 V HSTL Class II	16
1.5 V HSTL Class I	12, 10, 8
1.5 V HSTL Class II	16
1.2 V HSTL Class I	12, 10, 8
1.2 V HSTL Class II	14

You can specify the slew rate on a pin-by-pin basis because each I/O pin contains a slew rate control. The slew rate control affects both the rising and falling edges.

Note: Altera recommends that you perform IBIS or SPICE simulations to determine the best slew rate setting for your specific application.

Programmable IOE Delay

You can activate the programmable IOE delays to ensure zero hold times, minimize setup times, increase clock-to-output times, or delay the clock input signal. This feature helps read and write timing margins because it minimizes the uncertainties between signals in the bus.

Each pin can have a different input delay from pin-to-input register or a delay from output register-to-output pin values to ensure that the signals within a bus have the same delay going into or out of the device.

Table 2-7: Programmable Delay Chain

Programmable Delays	Quartus Prime Logic Option
Input pin-to-logic array delay	Input delay from pin to internal cells
Input pin-to-input register delay	Input delay from pin to input register
Output pin delay	Delay from output register to output pin
Dual-purpose clock input pin delay	Input delay from dual-purpose clock pin to fan-out destinations

There are two paths in the IOE for an input to reach the logic array. Each of the two paths can have a different delay. This allows you to adjust delays from the pin to the internal logic element (LE) registers that reside in two different areas of the device. You must set the two combinational input delays with the input delay from pin to internal cells logic option in the Quartus Prime software for each path. If the pin

uses the input register, one of the delays is disregarded and the delay is set with the input delay from pin to input register logic option in the Quartus Prime software.

The IOE registers in each I/O block share the same source for the preset or clear features. You can program preset or clear for each individual IOE, but you cannot use both features simultaneously. You can also program the registers to power-up high or low after configuration is complete. If programmed to power-up low, an asynchronous clear can control the registers. If programmed to power-up high, an asynchronous preset can control the registers. This feature prevents the inadvertent activation of the active-low input of another device upon power up. If one register in an IOE uses a preset or clear signal, all registers in the IOE must use that same signal if they require preset or clear. Additionally, a synchronous reset signal is available for the IOE registers.

Related Information

- **MAX 10 Device Datasheet**
- **Timing Closure and Optimization chapter, Volume 2: Design Implementation and Optimization, Quartus Prime Handbook**
  Provides more information about the input and output pin delay settings.

PCI Clamp Diode

The MAX 10 devices are equipped with optional PCI clamp diode that you can enable for the input and output of each I/O pin.

The PCI clamp diode is available in the Quartus Prime software for the following I/O standards:
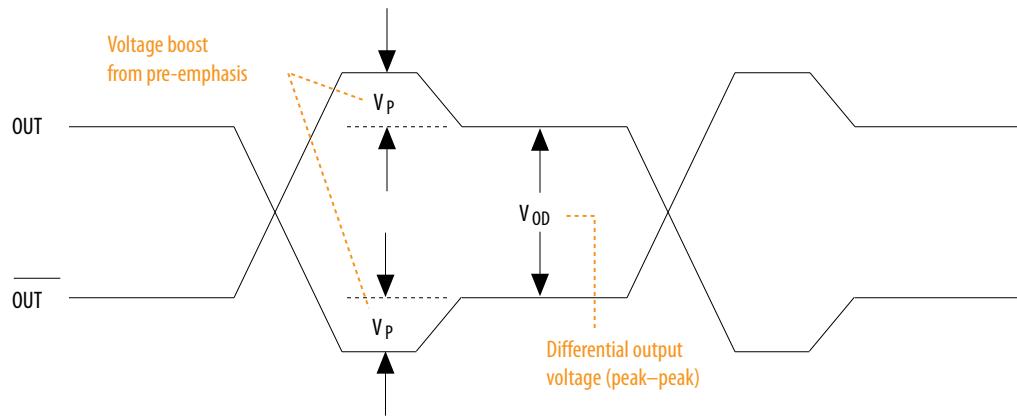
- 3.3 V LVTTL/3.3 V LVCMOS
- 3.0 V LVTTL/3.0 V LVCMOS
- 2.5 V LVTTL/2.5 V LVCMOS
- 3.0 V PCI
- 3.3 V Schmitt Trigger
- 2.5 V Schmitt Trigger

Programmable Pre-Emphasis

The differential output voltage (V_{OD}) setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full V_{OD} level before the next edge, producing pattern-dependent jitter. Pre-emphasis momentarily boosts the output current during switching to increase the output slew rate.

Pre-emphasis increases the amplitude of the high-frequency component of the output signal. This increase compensates for the frequency-dependent attenuation along the transmission line.

The overshoot introduced by the extra current occurs only during change of state switching. This overshoot increases the output slew rate but does not ring, unlike the overshoot caused by signal reflection. The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

Figure 2-5: LVDS Output with Programmable Pre-Emphasis

Table 2-8: Quartus Prime Software Assignment for Programmable Pre-Emphasis
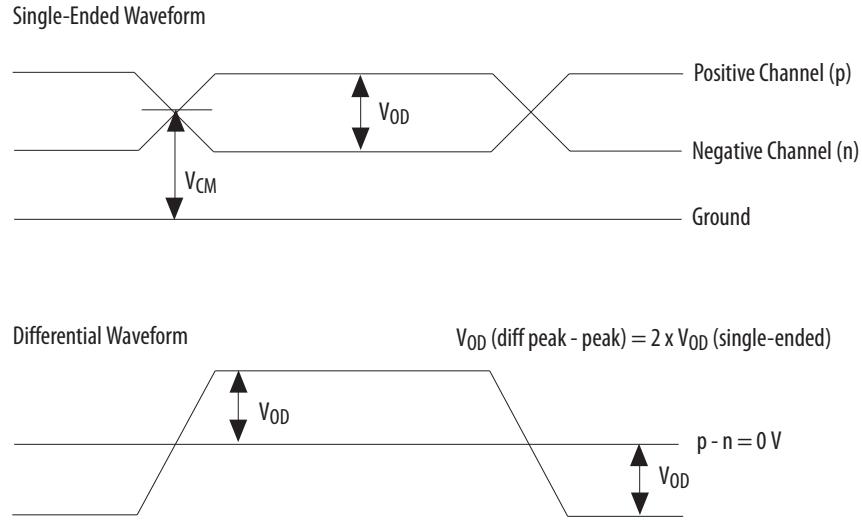
Field	Assignment
To	`tx_out`
Assignment name	Programmable Pre-emphasis
Allowed values	0 (disabled), 1 (enabled). Default is 1.

Programmable Differential Output Voltage

The programmable V_{OD} settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, and a smaller V_{OD} swing reduces power consumption.

Figure 2-6: Differential V_{OD}

This figure shows the V_{OD} of the differential LVDS output.

You can statically adjust the V_{OD} of the differential signal by changing the V_{OD} settings in the Quartus Prime software Assignment Editor.

Table 2-9: Quartus Prime Software Assignment Editor—Programmable V$_{OD}$

Field	Assignment
To	`tx_out`
Assignment name	Programmable Differential Output Voltage (V_{OD})
Allowed values	0 (low), 1 (medium), 2 (high). Default is 2.

Programmable Emulated Differential Output

The MAX 10 devices support emulated differential output where a pair of IOEs drives bidirectional I/O pins.

The emulated differential output feature is supported for the following I/O standards:

- Differential SSTL-2 Class I and II
- Differential SSTL-18 Class I and II
- Differential SSTL-15 Class I and II
- Differential SSTL-15
- Differential SSTL-135
- Differential 1.8 V HSTL Class I and II
- Differential 1.5 V HSTL Class I and II
- Differential 1.2 V HSTL Class I and II
- Differential HSUL-12
- LVDS 3R
- Mini-LVDS 3R
- PPDS 3R
- RSDS 1R and 3R
- BLVDS
- SLVS
- Sub-LVDS

Programmable Dynamic Power Down

The MAX 10 16, 25, 40, and 50 devices feature programmable dynamic power down for several I/O standards to reduce the static power consumption.

In these devices, you can apply the programmable dynamic power down feature to the I/O buffers for the following I/O standards:

- Input buffer—SSTL, HSTL, HSUL, LVDS
- Output buffer—LVDS

Related Information

MAX 10 Power Management User Guide

Provides more information about using the programmable dynamic power down feature.

I/O Standards Termination

Voltage-referenced and differential I/O standards requires different termination schemes.

The 3.3-V LVTTL, 3.0-V LVTTL and LVCMOS, 2.5-V LVTTL and LVCMOS, 1.8-V LVTTL and LVCMOS, 1.5-V LVCMOS, 1.2-V LVCMOS, and 3.0-V PCI I/O standards do not specify a recommended termination scheme per the JEDEC standard.

Voltage-Referenced I/O Standards Termination

Voltage-referenced I/O standards require an input reference voltage (V_{REF}) and a termination voltage (V_{TT}). The reference voltage of the receiving device tracks the termination voltage of the transmitting device.

Figure 2-7: HSTL I/O Standard Termination
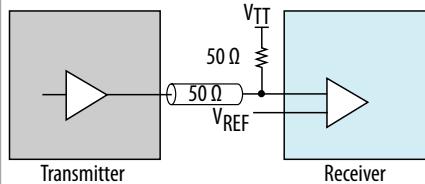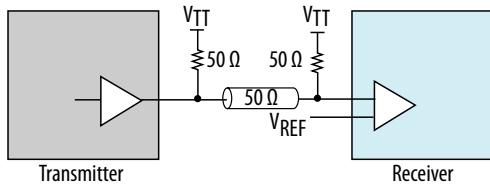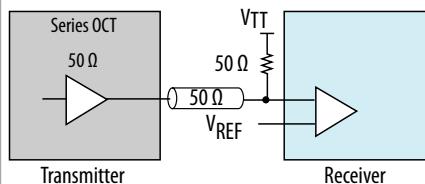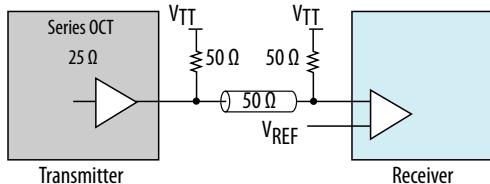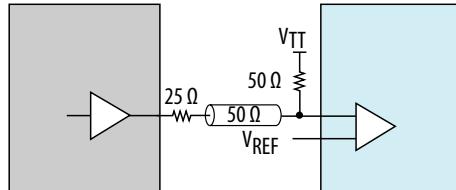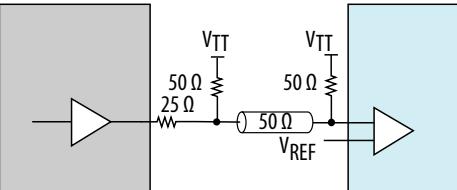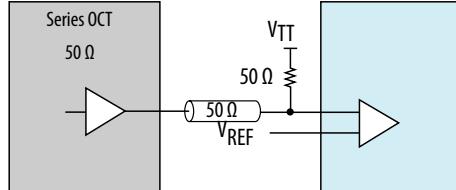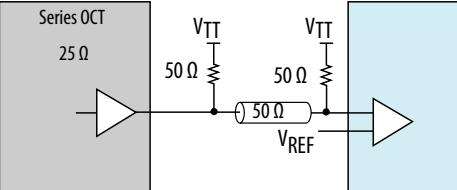
Figure 2-8: SSTL I/O Standard Termination

Differential I/O Standards Termination

Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the bus.

Figure 2-9: Differential HSTL I/O Standard Termination
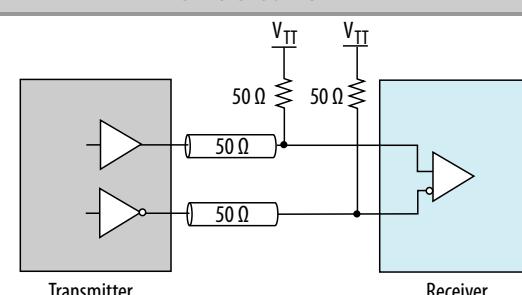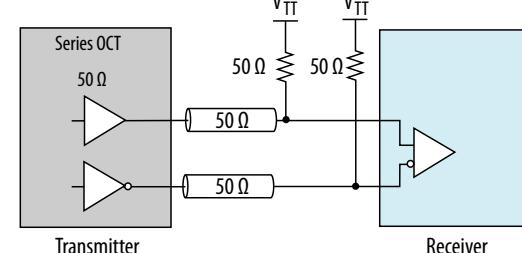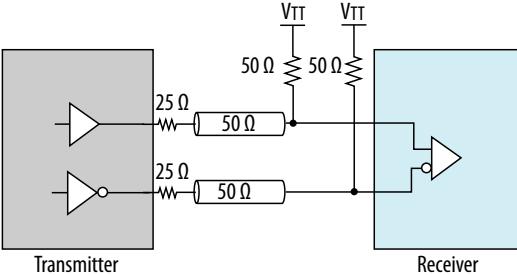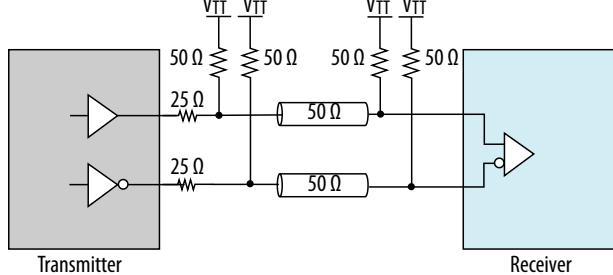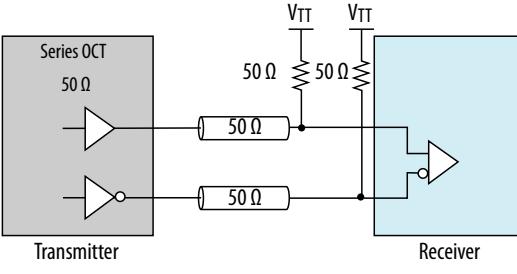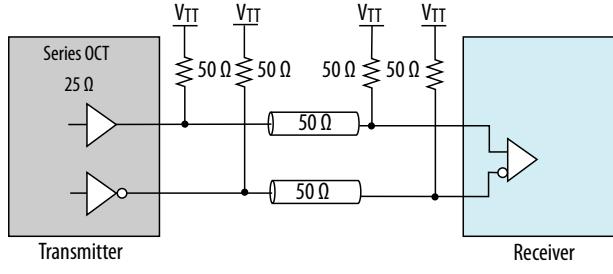
Figure 2-10: Differential SSTL I/O Standard Termination

Related Information

MAX 10 High-Speed LVDS I/O User Guide

Provides more information about differential I/O external termination.
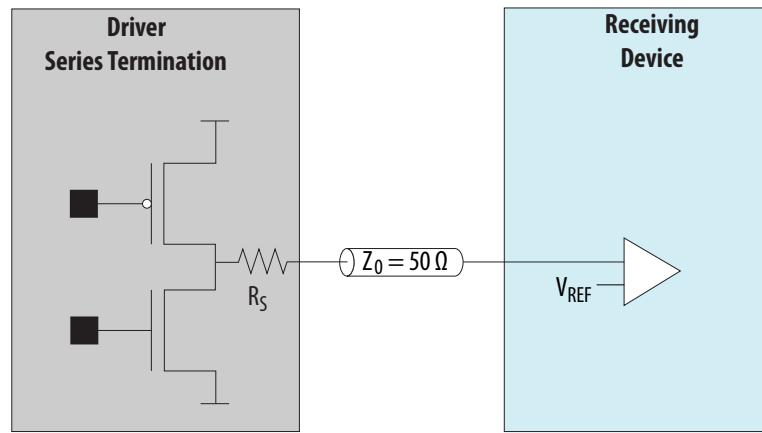
MAX 10 On-Chip I/O Termination

The on-chip termination (OCT) block in MAX 10 devices provides I/O impedance matching and termination capabilities. OCT maintains signal quality, saves board space, and reduces external component costs.

The MAX 10 devices support serial (R_S) OCT for single-ended output pins and bidirectional pins. For bidirectional pins, OCT is active for output only.

Figure 2-11: Single-ended I/O Termination (R$_S$)

This figure shows the single-ended termination scheme supported in MAX 10 device.

Table 2-10: OCT Schemes Supported in MAX 10 Devices

Direction	OCT Schemes	Device Support	I/O Bank Support
Output	R$_S$ OCT with calibration	MAX 10 16, 25, 40, and 50 devices	Right bank only
	R$_S$ OCT without calibration	All MAX 10 devices	All I/O banks

OCT Calibration

The OCT calibration circuit compares the total impedance of the output buffer to the external resistors connected to the RUP and RDN pins. The circuit dynamically adjusts the output buffer impedance until it matches the external resisters.

Each calibration block comes with a pair of RUP and RDN pins.

During calibration, the RUP and RDN pins are each connected through an external 25 Ω, 34 Ω, 40 Ω, 48 Ω, or 50 Ω resistor for respective on-chip series termination value of 25 Ω, 34 Ω, 40 Ω, 48 Ω, and 50 Ω:

- RUP—connected to VCCIO.
- RDN—connected to GND.

The OCT calibration circuit compares the external resistors to the internal resistance using comparators. The OCT calibration block uses the comparators' output to dynamically adjust buffer impedance.

During calibration, the resistance of the RUP and RDN pins varies. To estimate of the maximum possible current through the external calibration resistors, assume a minimum resistance of 0 Ω on the RUP and RDN pins.

R$_S$ OCT in MAX 10 Devices

Table 2-11: Selectable I/O Standards for R$_S$ OCT

This table lists the output termination settings for R$_S$ OCT with and without calibration on different I/O standards.

- R$_S$ OCT with calibration—supported only on the right side I/O banks of the MAX 10 16, 25, 40, and 50 devices.
- R$_S$ OCT without calibration—supported on all I/O banks of all MAX 10 devices.

I/O Standard	Calibrated OCT (Output) R$_S$ (Ω)	Uncalibrated OCT (Output) R$_S$ (Ω)
3.0 V LVTTL/3.0V LVCMOS	25, 50	25, 50
2.5 V LVTTL/2.5 V LVCMOS	25, 50	25, 50
1.8 V LVTTL/1.8 V LVCMOS	25, 50	25, 50
1.5 V LVCMOS	25, 50	25, 50
1.2 V LVCMOS	25, 50	25, 50
SSTL-2 Class I	50	50
SSTL-2 Class II	25	25
SSTL-18 Class I	50	50
SSTL-18 Class II	25	25
SSTL-15 Class I	50	50
SSTL-15 Class II	25	25
SSTL-15	34, 40	34, 40
SSTL-135	34, 40	34, 40
1.8 V HSTL Class I	50	50
1.8 V HSTL Class II	25	25
1.5 V HSTL Class I	50	50
1.5 V HSTL Class II	25	25
1.2 V HSTL Class I	50	50
1.2 V HSTL Class II	25	25
HSUL-12	34, 40, 48	34, 40, 48
Differential SSTL-2 Class I	50	50
Differential SSTL-2 Class I	25	25
Differential SSTL-18 Class I	50	50
Differential SSTL-18 Class II	25	25
Differential SSTL-15 Class I	50	50

I/O Standard	Calibrated OCT (Output)	Uncalibrated OCT (Output)
	R_S (Ω)	R_S (Ω)
Differential SSTL-15 Class II	25	25
Differential SSTL-15	34, 40	34, 40
Differential SSTL-135	34, 40	34, 40
Differential 1.8 V HSTL Class I	50	50
Differential 1.8 V HSTL Class II	25	25
Differential 1.5 V HSTL Class I	50	50
Differential 1.5 V HSTL Class II	25	25
Differential 1.2 V HSTL Class I	50	50
Differential 1.2 V HSTL Class II	25	25
Differential HSUL-12	34, 40, 48	34, 40, 48

UG-M10GPIO   ✉ **Subscribe**   💬 **Send Feedback**

There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Related Information
[MAX 10 I/O Overview](#) on page 1-1

Guidelines: V_{CCIO} Range Considerations

There are several V_{CCIO} range considerations because of I/O pin configuration function and I/O bank location.

- The shared I/O pins can only support a V_{CCIO} range of 1.5 V to 3.3 V when you access the configuration function in user mode. The configuration function of the I/O pins can only support 1.5 V to 3.3 V. If you need to access, for example, JTAG pins during user mode, the bank where the pin resides will be constrained by this V_{CCIO} range. If you want to use I/O standards within the 1.2 V to 1.35 V range, you must not use the configuration function of any of the I/O pins during user mode. This only affects bank 1 and bank 8 because only these banks have I/O pins with configuration function.
- For devices with banks 1A and 1B:

  - If you use the VREF pin or the ADC, you must supply a common V_{CCIO} voltage to banks 1A and 1B.
  - If you do not use the VREF pin or the ADC, you can supply separate V_{CCIO} voltages to banks 1A and 1B.

- If you plan to migrate from devices that has banks 1A and 1B to devices that has only bank 1, ensure that the V_{CCIO} of bank 1A and 1B are the same.
- For the V36 package of the 10M02 device, the V_{CCIO} of these groups of I/O banks must be the same:

  - Group 1—banks 1, 2 and 8
  - Group 2—banks 3, 5, and 6

- For the V81 package of the 10M08 device, the V_{CCIO} of these groups of I/O banks must be the same:

  - Group 1—banks 1A, 1B, and 2
  - Group 2—banks 5 and 6

Guidelines: Voltage-Referenced I/O Standards Restriction

These restrictions apply if you use the V_{REF} pin.

ISO 9001:2008 Registered

now part of Intel

- If you use a shared VREF pin as an I/O, all voltage-reference input buffers (SSTL, HSTL, and HSUL) are disabled.
- If you use a shared VREF pin as a voltage reference, you must enable the input buffer of specific I/O pin to use the voltage-reference I/O standards.
- The voltage-referenced I/O standards are not supported in the following I/O banks of these device packages:

  - All I/O banks of V36 package of 10M02.
  - All I/O banks of V81 package of 10M08.
  - Banks 1A and 1B of E144 package of 10M50.

- For devices with banks 1A and 1B, if you use the VREF pin, you must supply a common V_{CCIO} to banks 1A and 1B.
- Maximum number of voltage-referenced inputs for each VREF pin is 75% of total number of I/O pads. The Quartus Prime software will provide a warning if you exceed the maximum number.
- Except for I/O pins that you used for static signals, all non-voltage-referenced output must be placed two pads away from a VREF pin. The Quartus Prime software will output an error message if this rule is violated.

Related Information
MAX 10 I/O Standards Support on page 2-1

Guidelines: Enable Clamp Diode for LVTTL/LVCMOS Input Buffers

If the V_{CCIO} of the I/O bank is lower than the voltage of the LVTTL/LVCMOS input buffers, Altera recommends that you enable the clamp diode.

- 3.3 V LVCMOS/LVTTL input buffers—enable clamp diode if V_{CCIO} of the I/O bank is 3.0 V.
- 3.3 V or 3.0 V LVCMOS/LVTTL input buffers—enable clamp diode if V_{CCIO} of the I/O bank is 2.5 V.

By enabling the clamp diode under these conditions, you will be able to limit overshoot or undershoot. However, this does not comply with hot socket current specification.

If you do not enable the clamp diode under these conditions, the signal integrity for the I/O pin will be impacted and there will be overshoot or undershoot problem. In this situation, you must ensure that your board design conforms to the overshoot/undershoot specifications.

Table 3-1: Voltage Tolerance Maximum Ratings for 3.3 V or 3.0 V

This table lists the voltage tolerance specifications. Ensure that your board design conforms to these specifications if you do not want to follow the clamp diode recommendation.

Voltage	Minimum (V)	Maximum (V)
V_{CCIO} = 3.3 V	3.135	3.45
V_{CCIO} = 3.0 V	2.85	3.15
V_{IH} (AC)	—	4.1
V_{IH} (DC)	—	3.6
V_{IL} (DC)	−0.3	0.8

Guidelines: Adhere to the LVDS I/O Restrictions Rules

For LVDS applications, adhere to the I/O restriction pin connection guidelines to avoid excessive jitter on the LVDS transmitter output pins. The Quartus Prime software generates a critical warning if these rules are violated.

Related Information
MAX 10 FPGA Device Family Pin Connection Guidelines

Guidelines: I/O Restriction Rules

For different I/O standards and conditions, you must limit the number of I/O pins. This I/O restriction rule is applicable if you use LVDS transmitters or receivers.

Table 3-2: Maximum Percentage of I/O Pins Allowed for Specific I/O Standards in an I/O Bank

This table lists the maximum number of general purpose output pins recommended in a bank in terms of percentage to the total number of I/O pins available in an I/O bank if you use these combinations of I/O standards and conditions.

I/O Standard	Condition	Max Pins Per Bank (%)
2.5 V LVTTL/ LVCMOS	16 mA current strength and 25 Ω OCT (fast and slow slew rate)	25
	12 mA current strength (fast and slow slew rate)	30
	8 mA current strength (fast and slow slew rate) and 50 Ω OCT (fast slew rate)	45
	4 mA current strength (fast and slow slew rate)	65
2.5 V SSTL	—	100

Guidelines: Analog-to-Digital Converter I/O Restriction

These restrictions are applicable if you use the analog-to-digital converter (ADC) block.

The Quartus Prime software uses physics-based rules to define the number of I/Os allowed in a particular bank based on the I/O's drive strength. These rules are based on noise calculation to analyze accurately the impact of I/O placement on the ADC performance.

The physics-based rules are available for the following devices starting from these Quartus Prime software versions:

- From Quartus Prime version 14.1—MAX 10 10M04, 10M08, 10M40, and 10M50 devices.
- From Quartus Prime version 15.0.1—MAX 10 10M02, 10M16, and 10M25 devices.

Altera highly recommends that you adhere to these guidelines to ensure ADC performance. Furthermore, following these guidelines prevents additional critical warning from future versions of the Quartus Prime software when the physics-based rules are implemented.

Send Feedback

Table 3-3: I/O Restrictions Related to ADC Usage—Preliminary

This table lists the I/O restrictions by MAX 10 device package if you use the dedicated analog input (ANAIN1 or ANAIN2) or any dual function ADC I/O pins as ADC channel inputs.

Package	Restriction/Guideline
All	Disable all JTAG operation during ADC sampling. The ADC signal-to-noise and distortion ratio (SINAD) is not guaranteed during JTAG operation.

M153 U169 U324 F256 F484 F672	• Banks 1A and 1B—you cannot use GPIO pins in these banks.
	• Banks 2, 3, 4, 5, 6, and 7—you can use GPIO pins located in these banks.
	• Bank 8—you can use a percentage of the GPIO pins in this bank based on drive strength:
	  • For an example listing the percentage of GPIO pins allowed in bank 8 for the F484 package, refer to **Table 3-4**[9].
	  • Use low drive strength (8 mA and below) and differential I/O standards.
	  • Do not place transmitter pins in this bank. Use banks 2, 3, 4, 5, 6, or 7 instead.
	  • You can use static pins such as RESET or CONTROL.
	  • GPIO pins in this bank are governed by physics-based rules. The Quartus Prime software will issue a critical warning I/O settings violates any of the I/O physic-based rule.

E144	• Bank 1A, 1B, 2, and 8—you cannot use GPIO pins in these banks.
	• Banks 4 and 6—you can use GPIO pins located in these banks.
	• Banks 3, 5, and 7—you can use a percentage of the GPIO pins in this bank based on drive strength:
	  • For the percentage of GPIO pins allowed, refer to **Table 3-5**.
	  • Use low drive strength (8 mA and below) and differential I/O standards.
	  • GPIO pins in these banks are governed by physics-based rules. The Quartus Prime software will issue a critical warning I/O settings violates any of the I/O physic-based rule.

Table 3-4: I/O Usage Restriction for Bank 8 in MAX 10 F484 Package

This table lists the percentage of I/O pins available in I/O bank 8 if you use the dedicated analog input (ANAIN1 or ANAIN2) or any dual function ADC I/O pins as ADC channel. Refer to **Table 3-6** for the list of I/O standards in each group.

I/O Standards	TX	RX	Total	Availability (%)
Group 1	18	18	36	100
Group 2	16	16	32	89
Group 3	7	11	18	50
Group 4	5	7	12	33

[9] For all device packages, the software displays a warning message if the number of GPIO pins in bank 8 is more than the allowed percentage.

I/O Standards	TX	RX	Total	Availability (%)
Group 5	4	6	10	28
Group 6	4	4	8	22
Group 7	0	8	8	22

Table 3-5: I/O Usage Restriction for Banks 3, 5, and 7 in MAX 10 E144 Package

This table lists the percentage of I/O pins available in banks 3, 5, and 7 if you use the dedicated analog input (ANAIN1 or ANAIN2) or any dual function ADC I/O pins as ADC channel inputs. Refer to **Table 3-6** for the list of I/O standards in each group.

I/O Standards	Bank 3			Bank 5			Bank 7			Device I/O Availability (%)
	TX	RX	Availability (%)	TX	RX	Availability (%)	TX	RX	Availability (%)	
Group 1	7	8	88	6	6	100	4	3	100	54
Group 2	7	8	88	6	6	100	4	3	100	54
Group 3	4	5	50	6	6	100	2	0	29	45
Group 4	3	4	39	5	5	83	0	0	0	39
Group 5	2	3	28	5	5	83	0	0	0	37
Group 6	1	2	17	5	5	83	0	0	0	35
Group 7	0	0	0	5	5	83	0	0	0	32

Table 3-6: I/O Standards Groups Categorized According to Drive Strengths

I/O Standard Group	I/O Standards Name and Drive Strength
Group 1	2.5 V LVDS 2.5 V RSDS - BLVDS at 4 mA - SLVS at 4 mA

I/O Standard Group	I/O Standards Name and Drive Strength
Group 2	• BLVDS at 8 mA • SLVS at 8 mA • Sub-LVDS at 8 mA • 1.8 V, 1.5 V, and 1.2 V HSTL Class I at 8 mA • SSTL-15 at 34 Ω or 40 Ω • SSTL-135 at 34 Ω or 40 Ω • HSUL-12 at 34 Ω or 40 Ω • SSTL-2 Class I at 8 mA • SSTL-18 Class I at 8 mA • SSTL-15 Class I at 8 mA • 2.5 V and 1.8 V LVTTL at 4 mA • 2.5 V, 1.8 V, 1.5 V, and 1.2 V LVCMOS at 4 mA • 1.8 V LVTTL at 2 mA • 1.8 V, 1.5 V, and 1.2 V LVCMOS at 2 mA
Group 3	• BLVDS at 12 mA • SLVS at 12 mA • Sub-LVDS at 12 mA • SSTL-2 Class I at 10 mA or 12 mA • SSTL-18 Class I at 10 mA or 12 mA • SSTL-15 Class I at 10 mA or 12 mA • 1.8 V, 1.5 V, and 1.2 V HSTL Class I at 10 mA or 12 mA • SSTL-2 at 50 Ω • SSTL-18 at 50 Ω • SSTL-15 at 50 Ω • 1.8 V, 1.5 V and 1.2 V HSTL at 50 Ω • HSUL-12 at 48 Ω • 2.5 V and 1.8 V LVTTL at 50 Ω • 2.5 V, 1.8 V, 1.5 V, and 1.2 V LVCMOS at 50 Ω • 1.8 V LVTTL at 6 mA or 8 mA • 1.8 V, 1.5 V, and 1.2 V LVCMOS at 6 mA or 8 mA • 3.0 V LVTTL at 4 mA • 3.0 V LVCMOS at 4 mA
Group 4	• SSTL-18 Class II at 12 mA • 3.0 V LVTTL at 50 Ω • 3.0 V LVCMOS at 50 Ω • 2.5 V LVTTL at 8 mA • 2.5 V LVCMOS at 8 mA • 1.8 V LVTTL at 10 mA or 12 mA • 1.8 V, 1.5 V, and 1.2 V LVCMOS at 10 mA or 12 mA • 3.3 V LVCMOS at 2 mA

I/O Standard Group	I/O Standards Name and Drive Strength
Group 5	• SSTL-2 Class II at 16 mA • SSTL-18 Class II at 16 mA • SSTL-15 Class II at 16 mA • 1.8 V and 1.5 V HSTL Class II at 16 mA • 1.2 V HSTL Class II at 14 mA • SSTL-18 at 25 Ω • SSTL-15 at 25 Ω • SSTL-2 at 25 Ω • 1.8 V, 1.5 V, and 1.2 V HSTL at 25 Ω • 2.5 V and 1.8 V LVTTL at 25 Ω • 2.5 V, 1.8 V, 1.5 V, and 1.2 LVCMOS at 25 Ω • 1.8 V LVTTL at 16 mA • 1.8 V and 1.5 V LVCMOS at 16 mA • 2.5 V LVCMOS at 12 mA • 2.5 V LVTTL at 12 mA • 3.0 V LVCMOS at 8 mA • 3.0 V LVTTL at 8 mA • 3.3 V LVTTL at 4 mA or 8 mA
Group 6	• 2.5 V LVTTL at 16 mA • 2.5 V LVCMOS at 16 mA • 3.0 V LVTTL at 12 mA • 3.0 V LVCMOS at 12 mA • 3.0 V LVTTL at 25 Ω • 3.0 V LVCMOS at 25 Ω
Group 7	• 3.0 V LVTTL at 16 mA • 3.0 V LVCMOS at 16 mA

Guidelines: External Memory Interface I/O Restrictions

These I/O rules are applicable if you use external memory interfaces in your design.

Two GPIOs Adjacent to DQ Pin Is Disabled

This limitation is applicable to MAX 10 10M16, 10M25, 10M40, and 10M50 devices, and only if you use DDR3 and LPDDR2 SDRAM memory standards.

Table 3-7: DDR3 and LPDDR2 Memory Interface Widths and Device Packages Where Two GPIOs Adjacent to DQ Pins Are Disabled

This table lists the combination of MAX 10 10M16, 10M25, 10M40, and 10M50 device packages, and DDR3 and LPDDR2 memory interface widths where you cannot use two GPIO pins that are adjacent to the DQ pins.

Device Package	Memory Interface Width (DDR3 and LPPDR2 only)
U324	x8
F484	x8, x16, x24
F672	x8, x16, x24

Total I/O Utilization in Bank Must Be 75 Percent or Less in Some Devices

If you use DDR3 or LPDDR2 SDRAM memory interface standards, you can generally use a maximum of 75 percent of the total number of I/O pins available in a bank. This restriction differs from device to device. In some devices packages you can use all 100 percent of the I/Os. The Quartus Prime software will output an error message if the I/O usage per bank of that device is affected by this rule.

If you use DDR2 memory interface standards, you can assign 25 percent of the I/O pins as input pins only.

Guidelines: Dual-Purpose Configuration Pin

To use configuration pins as user I/O pins in user mode, you have to adhere to the following guidelines.

Table 3-8: Dual-Purpose Configuration Pin Guidelines for MAX 10 Devices

Guidelines	Pins
Configuration pins during initialization: • Tri-state the external I/O driver and drive an external pull-up resistor[10] or • Use the external I/O driver to drive the pins to the state same as the external weak pull-up resistor	• nCONFIG • nSTATUS • CONF_DONE

---

[10] If you intend to remove the external weak pull-up resistor, Altera recommends that you remove it after the device enters user mode.

Guidelines	Pins
JTAG pins: • If you intend to switch back and forth between user I/O pins and JTAG pin functions using the `JTAGEN` pin, all JTAG pins must be assigned as single-ended I/O pins or voltage-referenced I/O pins. Schmitt trigger input is the recommended input buffer. • JTAG pins cannot perform as JTAG pins in user mode if you assign any of the JTAG pin as a differential I/O pin. • You must use the JTAG pins as dedicated pins and not as user I/O pins during JTAG programming. • Do not toggle JTAG pin during the initialization stage. • Put the test access port (TAP) controller in reset state by driving the `TDI` and `TMS` pins high and `TCK` pin low for at least 5 clock cycles before the initialization.	• `TDO` • `TMS` • `TCK` • `TDI`

> **Attention:** Assign all JTAG pins as single-ended I/O pins or voltage-referenced I/O pins if you enable JTAG pin sharing feature.

Related Information

MAX 10 FPGA Configuration User Guide

Provides more information about the dual-purpose I/O pins in configuration and user modes.

Guidelines: Clock and Data Input Signal for MAX 10 E144 Package

There is strong inductive coupling on the MAX 10 E144 lead frame package. Glitch may occur on an input pin when an aggressor pin with strong drive strength toggles directly adjacent to it.

PLL Clock Input Pins

The PLL clock input pins are sensitive to SSN jitter. To avoid the PLL from losing lock, do not use the output pins directly on the left and right of the PLL clock input pins.

Data Input Pins

Potential glitch on the data input pin, leading to input read signal failure, can occur in the following conditions:

• The output pin directly adjacent to the data input pin is assigned an unterminated I/O standard, such as LVTTL and LVCMOS, with drive strength of 8 mA or higher.
• The output pin directly adjacent to the data input pin is assigned a terminated I/O standard, such as SSTL, with drive strength of 8 mA or higher.

To reduce jitter on data input pin, Altera recommends the following guidelines:

- Reduce the drive strength of the directly adjacent output pin for the different unterminated I/O standards as follows:

  - 4 mA or below—2.5 V, 3.0 V, and 3.3 V unterminated I/O standards
  - 6 mA or below—1.2 V, 1.5 V, and 1.8 V unterminated I/O standards

- For unterminated I/O standard, assign the pins directly on the left and right of the data input pin to a non-toggling signal.
- For terminated I/O standard, you can use only one pin directly on the left or right of the data input pin as toggling signal, provided that you set the slew rate setting of this pin to "0" (slow slew rate). Otherwise, assign the pins directly on the left and right of the data input pin to a non-toggling signal.
- Change the unterminated I/O standard data input pin to a Schmitt Trigger input buffer for better noise immunity. If you are using Schmitt Trigger input buffer on the data input pin, you can use the directly adjacent output pin with unterminated I/O standard at a maximum drive strength of 8 mA.

You can implement your I/O design in the Quartus Prime software. The software contains tools for you to create and compile your design, and configure your device.

The Quartus Prime software allows you to prepare for device migration, set pin assignments, define placement restrictions, setup timing constraints, and customize IP cores. For more information about using the Quartus Prime software, refer to the related information.

Related Information

MAX 10 I/O Overview on page 1-1
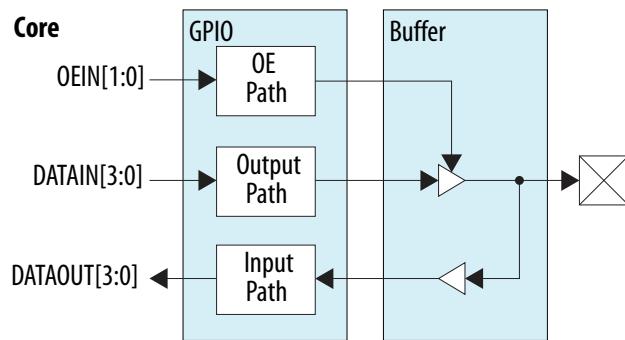
Altera GPIO Lite IP Core

The Altera GPIO Lite IP core supports the MAX 10 GPIO components. To implement the GPIOs in your design, you can customize the Altera GPIO Lite IP core to suit your requirements and instantiate it in your design.

GPIOs are I/Os used in general applications not specific to transceivers, memory-like interfaces or LVDS. The Altera GPIO Lite IP core features the following components:

- Double data rate input/output (DDIO)—A digital component that doubles the data-rate of a communication channel.
- I/O buffers—connect the pads to the FPGA.

Figure 4-1: High Level View of Single-Ended GPIO

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Altera GPIO Lite IP Core Data Paths

Table 4-1: Altera GPIO Lite Data Path Modes

Data Path	Mode		
	Bypass	**Single Register**	**DDR**
Input	Data goes from the delay element to the core, bypassing all double data rate I/Os (DDIOs).	The full-rate DDIO operates as a single register.	The full-rate DDIO operates as a regular DDIO.
Output	Data goes from the core straight to the delay element, bypassing all DDIOs.	The full-rate DDIO operates as a single register.	The full-rate DDIO operates as a regular DDIO.
Bidirectional	The output buffer drives both an output pin and an input buffer.	The full-rate DDIO operates as a single register. The output buffer drives both an output pin and an input buffer.	The full-rate DDIO operates as a regular DDIO. The output buffer drives both an output pin and an input buffer. The input buffer drives a set of three flip-flops.

If you use asynchronous clear and preset signals, all DDIOs share these same signals.

DDR Input Path

The pad sends data to the input buffer and the input buffer feeds the delay element. From the delay element, the data is fed to the DDIO stage, which consists of three registers:

- `RegAi` samples the data from `pad_in` at the positive clock edge.
- `RegBi` samples the data from `pad_in` at the negative clock edge.
- `RegCi` samples the data from `RegAi` at the negative clock edge.

**Figure 4-2: Simplified View of Altera GPIO Lite DDR Input Path**
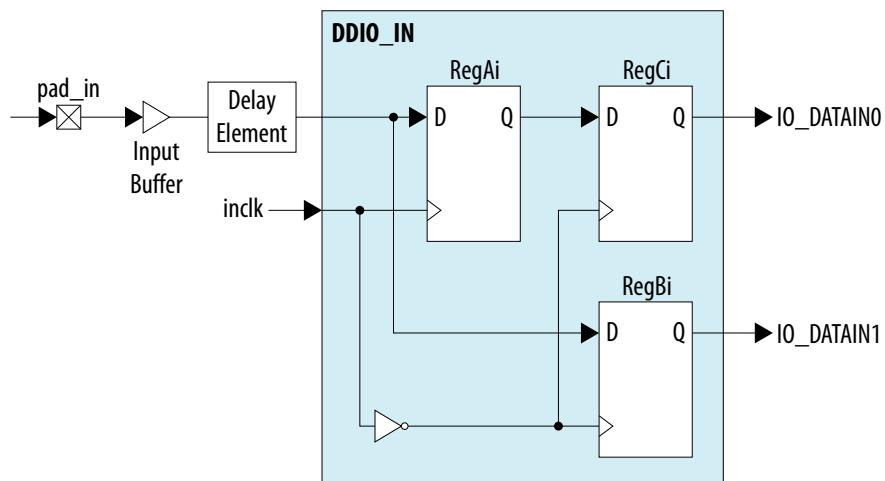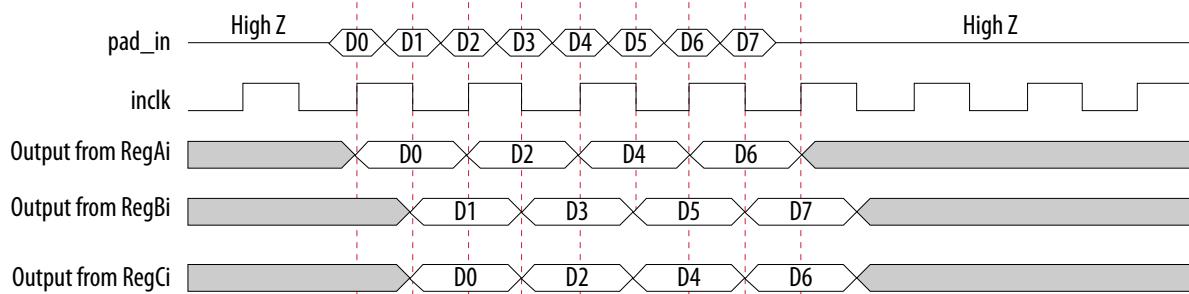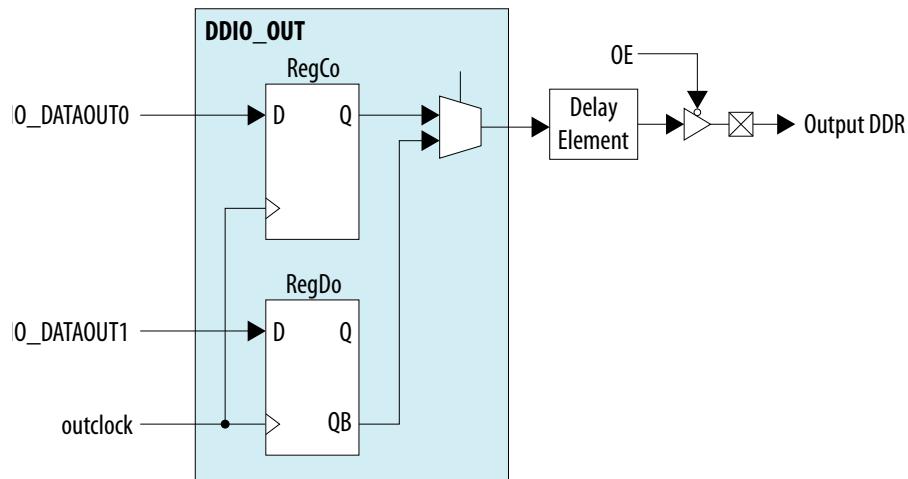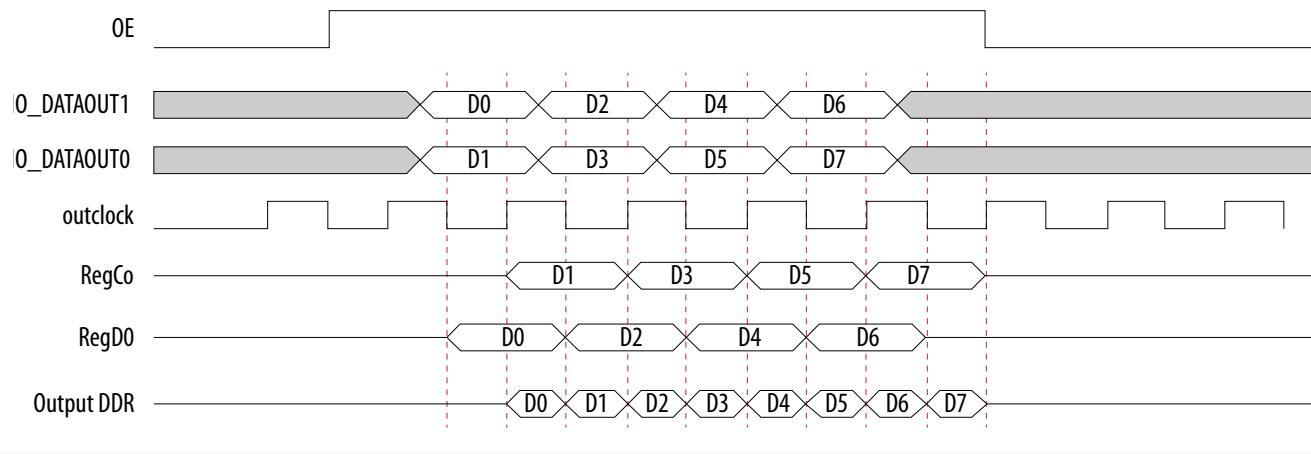
Figure 4-3: Altera GPIO Lite Input Path Timing Diagram

DDR Output Path with Output Enable

- `RegCo` samples the data from `IO_DATAOUT0` at the positive clock edge.
- `RegDo` samples the data from `IO_DATAOUT1` when `outclock` value is 0.
- `Output DDR` samples the data from `RegCo` at the positive clock edge, and from `RegDo` at the negative clock edge.

Send Feedback

Figure 4-4: Simplified View of Altera GPIO Lite DDR Output Path with Output Enable


Figure 4-5: Altera GPIO Lite Output Path Timing Diagram

Verifying Pin Migration Compatibility

You can use the **Pin Migration View** window in the Quartus Prime software Pin Planner to assist you in verifying whether your pin assignments migrate to a different device successfully.

You can vertically migrate to a device with a different density while using the same device package, or migrate between packages with different densities and ball counts.

1. Open **Assignments** > **Pin Planner** and create pin assignments.
2. If necessary, perform one of the following options to populate the Pin Planner with the node names in the design:

- Analysis & Elaboration
- Analysis & Synthesis
- Fully compile the design

3. Then, on the menu, click **View** > **Pin Migration View**.

4. To select or change migration devices:

   a. Click **Device** to open the **Device** dialog box.

   b. Under **Migration compatibility** click **Migration Devices**.

5. To show more information about the pins:

   a. Right-click anywhere in the **Pin Migration View** window and select **Show Columns**.

   b. Then, click the pin feature you want to display.

6. If you want to view only the pins, in at least one migration device, that have a different feature than the corresponding pin in the migration result, turn on **Show migration differences**.

7. Click **Pin Finder** to open the **Pin Finder** dialog box and find and highlight pins with specific functionality.

   If you want to view only the pins found and highlighted by the most recent query in the **Pin Finder** dialog box, turn on **Show only highlighted pins**.

8. To export the pin migration information to a Comma-Separated Value File (**.csv**), click **Export**.

Related Information

MAX 10 I/O Vertical Migration Support on page 1-3

You can set various parameter settings for the Altera GPIO Lite IP core to customize its behaviors, ports, and signals.

The Quartus Prime software generates your customized Altera GPIO Lite IP core according to the parameter options that you set in the parameter editor.

Related Information

MAX 10 I/O Overview on page 1-1

Altera GPIO Lite Parameter Settings

You can set the parameter settings for the Altera GPIO Lite IP core in the Quartus Prime software. There are three groups of options: **General**, **Buffer**, and **Registers**.

Table 5-1: Altera GPIO Lite Parameters - General

Parameter	Condition	Allowed Values	Description
Data direction	—	• input • output • bidir	Specifies the data direction for the GPIO.
Data width	—	1 to 128	Specifies the data width.

Table 5-2: Altera GPIO Lite Parameters - Buffer

Parameter	Condition	Allowed Values	Description
Use true differential buffer	Data direction = input or output	• On • Off	If turned on, enables true differential I/O buffers and disables pseudo differential I/O buffers.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Parameter	Condition	Allowed Values	Description
Use pseudo differential buffer	Data direction = output or bidir	• On • Off	• If turned on in output mode—enables pseudo differential output buffers and disables true differential I/O buffers. • If turned on in bidir mode—enables true differential input buffer and pseudo differential output buffer.
Use bus-hold circuitry	Data direction = input or output	• On • Off	If turned on, the bus hold circuitry can weakly hold the signal on an I/O pin at its last-driven state where the output buffer state will be 1 or 0 but not high-impedance.
Use open drain output	Data direction = output or bidir	• On • Off	If turned on, the open drain output enables the device to provide system-level control signals such as interrupt and write enable signals that can be asserted by multiple devices in your system.
Enable oe port	Data direction = output	• On • Off	If turned on, enables user input to the OE port. This option is automatically turned on for bidirectional mode.
Enable nsleep port (only available in selected devices)	Data direction = input or bidir	• On • Off	If turned on, enables the nsleep port. This option is available for the 10M16, 10M25, 10M40, and 10M50 devices.

Table 5-3: Altera GPIO Lite Parameters - Registers

Parameter	Condition	Allowed Values	Description
Register mode	—	• bypass • single-register • ddr	Specifies the register mode for the Altera GPIO Lite IP core: • **bypass**—specifies a simple wire connection from/to the buffer. • **single-register**—specifies that the DDIO is used as a simple register in single data-rate mode (SDR). The Fitter may pack this register in the I/O. • **ddr**— specifies that the IP core uses the DDIO.
Enable aclr port	• Register mode = ddr	• On • Off	If turned on, enables the ACLR port for asynchronous clears.
Enable aset port	• Data direction = output or bidir • Register mode = ddr • Set registers to power up high (when aclr and aset ports are not used) = off	• On • Off	If turned on, enables the ASET port for asynchronous preset.
Set registers to power up high (when aclr and aset ports are not used)	• Register mode = ddr • Enable aclr port = off • Enable aset port = off • Enable sclr port = off	• On • Off	If you are not using the ACLR and ASET ports: • **On**—specifies that registers power up HIGH. • **Off**—specifies that registers power up LOW.

Parameter	Condition	Allowed Values	Description
Enable inclocken/outclocken ports	Register mode = ddr	• On • Off	• **On**—exposes the clock enable port to allow you to control when data is clocked in or out. This signal prevents data from being passed through without your control. • **Off**—clock enable port is not exposed and data always pass through the register automatically.
Invert din	• Data direction = output • Register mode = ddr	• On • Off	If turned on, inverts the data out output port.
Invert DDIO inclock	• Data direction = input or bidir • Register mode = ddr	• On • Off	• **On**—captures the first data bit on the falling edge of the input clock. • **Off**—captures the first data bit on the rising edge of the input clock.
Use a single register to drive the output enable (oe) signal at the I/O buffer	• Data direction = output or bidir • Register mode = single-register or ddr • Use DDIO registers to drive the output enable (oe) signal at the I/O buffer = off	• On • Off	If turned on, specifies that a single register drives the OE signal at the output buffer.
Use DDIO registers to drive the output enable (oe) signal at the I/O buffer	• Data direction = output or bidir • Register mode = ddr • Use a single register to drive the output enable (oe) signal at the I/O buffer = off	• On • Off	If turned on, specifies that the DDR I/O registers drive the OE signal at the output buffer. The output pin is held at high impedance for an extra half clock cycle after the OE port goes high.

Parameter	Condition	Allowed Values	Description
Implement DDIO input registers in hard implementation (Only available in certain devices)	• Data direction = input or bidir • Register mode = ddr	• On • Off	• **On**—implements the DDIO input registers using hard block at the I/O edge. • **Off**—implements the DDIO input registers as soft implementation using registers in the FPGA core fabric. This option is applicable only for MAX 10 16, 25, 40, and 50 devices because the DDIO input registers hard block is available only in these devices. To avoid Fitter error, turn this option off for other MAX 10 devices.

Altera GPIO Lite Interface Signals

Depending on parameter settings you specify, different interface signals are available for the Altera GPIO Lite IP core.

Table 5-4: Pad Interface Signals

The pad interface connects the Altera GPIO Lite IP core to the pads.

Signal Name	Direction	Description
pad_in	Input	Input pad port if you use the input path.
pad_in_b	Input	Input negative pad port if you use the input path and enable the true or pseudo differential buffers.
pad_out	Output	Output pad port if you use the output path.
pad_out_b	Output	Output negative pad port if you use the output path and enable the true of pseudo differential buffers.
pad_io	Bidirectional	Bidirectional pad port if you use bidirectional paths.
pad_io_b	Bidirectional	Bidirectional negative pad port if you use bidirectional paths and enable true or pseudo differential buffers.

Table 5-5: Data Interface Signals

The data interface is an input or output interface from the Altera GPIO Lite IP core to the FPGA core.

Signal Name	Direction	Description
din	Input	Data received from the input pin. Signal width for each input pin: • DDR mode—2 • Other modes—1
dout	Output	Data to send out through the output pin. Signal width for each output pin: • DDR mode—2 • Other modes—1
oe	Input	Control signal that enables the output buffer. This signal is active HIGH.
nsleep	Input	Control signal that enables the input buffer. This signal is active LOW. This signal is available for the 10M16, 10M25, 10M40, and 10M50 devices.

Table 5-6: Clock Interface Signals

The clock interface is an input clock interface. It consists of different signals, depending on the configuration. The Altera GPIO Lite IP core can have zero, one, two, or four clock inputs. Clock ports appear differently in different configurations to reflect the actual function performed by the clock signal.

Signal Name	Direction	Description
inclock	Input	Input clock that clocks the registers in the input path.
inclocken	Input	Control signal that controls when data is clocked in. This signal is active HIGH.
outclock	Input	Input clock that clocks the registers in the output path.
ouctlocken	Input	Control signal that controls when data is clocked out. This signal is active HIGH.

Table 5-7: Reset Interface Signals

The reset interface connects the Altera GPIO Lite IP core to the DDIOs.

Signal Name	Direction	Description
aclr	Input	Control signal for asynchronous clear that sets the register output state to 0. This signal is active HIGH.
aset	Input	Control signal for asynchronous preset that sets the register output state to 1. This signal is active HIGH.
sclr	Input	Control signal for synchronous clear that sets the register output to 0. This signal is active HIGH.

2016.05.02

UG-M10GPIO   ✉ **Subscribe**   💬 **Send Feedback**

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 General Purpose I/O User Guide**
15.0	**MAX 10 General Purpose I/O User Guide**
14.1	**MAX 10 General Purpose I/O User Guide**

**ISO
9001:2008
Registered**

ALTERA
now part of Intel

Document Revision History for MAX 10 General Purpose I/O User Guide

B

2016.05.02

✉ **Subscribe**   💬 **Send Feedback**

Date	Version	Changes
May 2016	2016.05.02	• Updated the list of supported I/O standards to specify I/O standards that are supported only in dual power supply MAX 10 devices. • Updated the names of emulated differential I/O standards to improve clarity. • Updated the topic about the I/O standards voltage and pin support to clarify that the I/O standards that a pin type supports depends on pin's I/O bank. • Updated the setting information for PCI clamp diode: • On by default for input pins for all supported I/O standards • Off by default for output pins for all supported I/O standards, except 3.0 V PCI • Updated the topic about the ADC I/O restriction: • Added the list of devices with physics-based rules support from Quartus Prime version 15.0.1. • Clarified that the table listing the percentage of GPIOs allowed in bank 8 is an example for the F484 package. For all packages, the Quartus Prime software displays a warning message if you exceed the allowed GPIO percentage.

ALTERA
now part of Intel

Date	Version	Changes
November 2015	2015.11.02	• Added PCI clamp diode support for the 3.3 V and 2.5 V Schmitt Trigger I/O standards. • Added a table that summarizes the programmable I/O buffer features and settings. • Updated the topics about V_{CCIO} range consideration and VREF I/O standards restriction with guidelines for using different V_{CCIO} supplies in bank 1A and bank 1B. • Added guidelines topic about using the clock and input pins in the E144 package. • Added the **Enable nsleep port** parameter option. • Removed the topics about the IP catalog and parameter editor, generating IP cores, and the files generated by the IP core, and added a link to *Introduction to Altera IP Cores*. • Changed instances of *Quartus II* to *Quartus Prime*.
June 2015	2015.06.10	• Added related link to the MAX 10 device pin-outs in topic about I/O banks locations. The device pin-out files provide more information about available I/O pins in each I/O bank. • Updated the ADC I/O restriction guidelines topic.
May 2015	2015.05.04	• Removed the F672 package of the MAX 10 10M25 device. • Updated footnote for LVDS (dedicated) in the table listing the supported I/O standards to clarify that you can use LVDS receivers on all I/O banks. • Added missing footnote number for the DQS column of the 3.3 V Schmitt Trigger row in the table that lists the I/O standards voltage levels and pin support. • Added a table listing the I/O standards and current strength settings that support programmable output slew rate control. • Updated the topic about external memory interface I/O restrictions to add x24 memory interface width to the F484 package. • Added topic about the programmable differential output voltage. • Updated the guidelines for voltage-referenced I/O standards to add a list of device packages that do not support voltage-referenced I/O standards. • Updated the topic about the I/O restriction rules to remove statements about the differential pad placement rules. • Renamed the `input_ena` signal name to `nsleep` and updated the relevant description. • Updated the description for the **Invert DDIO inclock** parameter of the Altera GPIO Lite IP core.

Date	Version	Changes
December 2014	2014.12.15	Updated the topic about the ADC I/O restriction: • Added information about implementation of physics-based rules in the Quartus Prime software. • Updated the list of I/O standards groups for the ADC I/O restriction.
September 2014	2014.09.22	Initial release.

MAX 10 High-Speed LVDS I/O User Guide

Subscribe

Send Feedback

Contents

MAX 10 High-Speed LVDS I/O Overview

UG-M10LVDS   ✉ Subscribe   💬 Send Feedback

The MAX® 10 device family supports high-speed LVDS protocols through the LVDS I/O banks and the Altera Soft LVDS IP core.

Table 1-1: Summary of LVDS I/O Buffers Support in MAX 10 I/O Banks

I/O Buffer Type	I/O Bank Support
True LVDS input buffer	All I/O banks
True LVDS output buffer	Only bottom I/O banks
Emulated LVDS output buffer	All I/O banks

The LVDS I/O standards support differs between MAX 10 D and S variants. Refer to related information for more details.

Related Information

- **MAX 10 High-Speed LVDS Architecture and Features** on page 2-1
  Provides information about the high-speed LVDS architecture and the features supported by the device.
- **MAX 10 LVDS Transmitter Design** on page 3-1
  Provides information and guidelines for implementing LVDS transmitter in MAX 10 devices using the Altera Soft LVDS IP core.
- **MAX 10 LVDS Receiver Design** on page 4-1
  Provides information and guidelines for implementing LVDS receiver in MAX 10 devices using the Altera Soft LVDS IP core.
- **MAX 10 LVDS Transmitter and Receiver Design** on page 5-1
  Provides design guidelines for implementing both LVDS transmitters and receivers in the same MAX 10 device.
- **Altera Soft LVDS IP Core References** on page 7-1
  Lists the parameters and signals of Altera Soft LVDS IP core for MAX 10 devices.
- **MAX 10 LVDS SERDES I/O Standards Support** on page 2-7
  Lists the supported LVDS I/O standards and the support in different MAX 10 device variants.
- **MAX 10 High-Speed LVDS I/O User Guide Archives** on page 8-1
  Provides a list of user guides for previous versions of the Altera Soft LVDS IP core.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Altera Soft LVDS Implementation Overview

You can implement LVDS applications in MAX 10 devices as transmitter-only, receiver-only, or a combination of transmitters and receivers.

Figure 1-1: MAX 10 LVDS Implementation Overview

UG-M10LVDS ✉ **Subscribe** 💬 **Send Feedback**

The MAX 10 devices use registers and logic in the core fabric to implement LVDS input and output interfaces.

- For LVDS transmitters and receivers, MAX 10 devices use the the double data rate I/O (DDIO) registers that reside in the I/O elements (IOE). This architecture improves performance with regards to the receiver input skew margin (RSKM) or transmitter channel-to-channel skew (TCCS).
- For the LVDS serializer/deserializer (SERDES), MAX 10 devices use logic elements (LE) registers.

Related Information

- **MAX 10 High-Speed LVDS I/O Overview** on page 1-1
- **MAX 10 LVDS SERDES I/O Standards Support** on page 2-7
  Lists the supported LVDS I/O standards and the support in different MAX 10 device variants.

MAX 10 LVDS Channels Support

The LVDS channels available vary among MAX 10 devices. All I/O banks in MAX 10 devices support true LVDS input buffers and emulated LVDS output buffers. However, only the bottom I/O banks support true LVDS output buffers.

ISO 9001:2008 Registered

ALTERA

now part of Intel

2-2

Table 2-1: LVDS Buffers in MAX 10 Devices

This table lists the LVDS buffer support for I/O banks on each side of the devices.

Product Line	Package	Side	True LVDS Pairs		Emulated LVDS Pairs
			TX	RX	
10M02	V36	Top	0	1	1
		Right	0	3	3
		Left	0	3	3
		Bottom	3	3	3
	M153	Top	0	12	12
		Right	0	12	12
		Left	0	12	12
		Bottom	9	13	13
	U169	Top	0	12	12
		Right	0	17	17
		Left	0	15	15
		Bottom	9	14	14
	U324	Top	0	13	13
		Right	0	24	24
		Left	0	20	20
		Bottom	9	16	16
	E144	Top	0	10	10
		Right	0	12	12
		Left	0	11	11
		Bottom	7	12	12

Send Feedback

Product Line	Package	Side	True LVDS Pairs		Emulated LVDS Pairs
			TX	RX	
10M04	M153	Top	0	12	12
		Right	0	12	12
		Left	0	12	12
		Bottom	9	13	13
	U169	Top	0	12	12
		Right	0	17	17
		Left	0	15	15
		Bottom	9	14	14
	U324	Top	0	27	27
		Right	0	31	31
		Left	0	28	28
		Bottom	15	28	28
	F256	Top	0	19	19
		Right	0	22	22
		Left	0	19	19
		Bottom	13	20	20
	E144	Top	0	8	8
		Right	0	12	12
		Left	0	11	11
		Bottom	10	10	10

Product Line	Package	Side	True LVDS Pairs		Emulated LVDS Pairs
			TX	RX	
10M08	V81	Top	0	5	5
		Right	0	7	7
		Left	0	6	6
		Bottom	7	7	7
	M153	Top	0	12	12
		Right	0	12	12
		Left	0	12	12
		Bottom	9	13	13
	U169	Top	0	12	12
		Right	0	17	17
		Left	0	15	15
		Bottom	9	14	14
	U324	Top	0	27	27
		Right	0	31	31
		Left	0	28	28
		Bottom	15	28	28
	F256	Top	0	19	19
		Right	0	22	22
		Left	0	19	19
		Bottom	13	20	20
	E144	Top	0	8	8
		Right	0	12	12
		Left	0	11	11
		Bottom	10	10	10
	F484	Top	0	27	27
		Right	0	33	33
		Left	0	28	28
		Bottom	15	28	28

Product Line	Package	Side	True LVDS Pairs		Emulated LVDS Pairs
			TX	RX	
10M16	U169	Top	0	12	12
		Right	0	17	17
		Left	0	15	15
		Bottom	9	14	14
	U324	Top	0	27	27
		Right	0	31	31
		Left	0	28	28
		Bottom	15	28	28
	F256	Top	0	19	19
		Right	0	22	22
		Left	0	19	19
		Bottom	13	20	20
	E144	Top	0	8	8
		Right	0	12	12
		Left	0	11	11
		Bottom	10	10	10
	F484	Top	0	39	39
		Right	0	38	38
		Left	0	32	32
		Bottom	22	42	42

Product Line	Package	Side	True LVDS Pairs		Emulated LVDS Pairs
			TX	RX	
10M25	F256	Top	0	19	19
		Right	0	22	22
		Left	0	19	19
		Bottom	13	20	20
	E144	Top	0	8	8
		Right	0	12	12
		Left	0	11	11
		Bottom	10	10	10
	F484	Top	0	41	41
		Right	0	48	48
		Left	0	36	36
		Bottom	24	46	46
10M40	F256	Top	0	19	19
		Right	0	22	22
		Left	0	19	19
		Bottom	13	20	20
	E144	Top	0	9	9
		Right	0	12	12
		Left	0	11	11
		Bottom	10	10	10
	F484	Top	0	41	41
		Right	0	48	48
		Left	0	36	36
		Bottom	24	46	46
	F672	Top	0	53	53
		Right	0	70	70
		Left	0	60	60
		Bottom	30	58	58

Product Line	Package	Side	True LVDS Pairs		Emulated LVDS Pairs
			TX	RX	
10M50	F256	Top	0	19	19
		Right	0	22	22
		Left	0	19	19
		Bottom	13	20	20
	E144	Top	0	9	9
		Right	0	12	12
		Left	0	11	11
		Bottom	10	10	10
	F484	Top	0	41	41
		Right	0	48	48
		Left	0	36	36
		Bottom	24	46	46
	F672	Top	0	53	53
		Right	0	70	70
		Left	0	60	60
		Bottom	30	58	58

MAX 10 LVDS SERDES I/O Standards Support

The MAX 10 D and S device variants support different LVDS I/O standards. All I/O banks in MAX 10 devices support true LVDS input buffers and emulated LVDS output buffers. However, only the bottom I/O banks support true LVDS output buffers.

Table 2-2: MAX 10 LVDS I/O Standards Support

Single and dual supply MAX 10 devices support different I/O standards. For more information about single and dual supply devices, refer to the device overview.

I/O Standard	I/O Bank	TX	RX	MAX 10 Device Support		Notes
				Dual Supply Device	Single Supply Device	
True LVDS	All	Bottom banks only	Yes	Yes	Yes	• All I/O banks support true LVDS input buffers. • Only the bottom I/O banks support true LVDS output buffers.
Emulated LVDS (three resistors)	All	Yes	—	Yes	Yes	All I/O banks support emulated LVDS output buffers.
True RSDS	Bottom	Yes	—	Yes	Yes	—
Emulated RSDS (single resistor)	All	Yes	—	Yes	—	All I/O banks support emulated RSDS output buffers.
Emulated RSDS (three resistors)	All	Yes	—	Yes	Yes	All I/O banks support emulated RSDS output buffers.
True Mini-LVDS	Bottom	Yes	—	Yes	—	—
Emulated Mini-LVDS (three resistors)	All	Yes	—	Yes	—	All I/O banks support emulated Mini-LVDS output buffers.
PPDS	Bottom	Yes	—	Yes	—	—
Emulated PPDS (three resistors)	All	Yes	—	Yes	—	—

I/O Standard	I/O Bank	TX	RX	MAX 10 Device Support		Notes
				Dual Supply Device	Single Supply Device	
Bus LVDS	All	Yes	Yes	Yes	Yes	• Bus LVDS (BLVDS) output uses two single-ended outputs with the second output programmed as inverted. • BLVDS input uses LVDS input buffer. • You can tristate BLVDS output.
LVPECL	All	—	Yes	Yes	Yes	Supported only on dual function clock input pins.
TMDS	All	—	Yes	Yes	—	• Requires external termination but does not require V_{REF}. • Requires external level shifter to support 3.3 V TMDS input. This level shifter must convert the TMDS signal from AC-coupled to DC-coupled before you connect it to the MAX 10 input buffer. • TMDS receiver support uses dedicated 2.5 V LVDS input buffer.
Sub-LVDS	All	Yes	Yes	Yes	—	• Transmitter supports only emulated Sub-LVDS using emulated 1.8 V differential signal as output. • Requires external output termination. • Does not require V_{REF}. • Sub-LVDS receiver support uses dedicated 2.5 V LVDS input buffer.

I/O Standard	I/O Bank	TX	RX	MAX 10 Device Support		Notes
				Dual Supply Device	Single Supply Device	
SLVS	All	Yes	Yes	Yes	—	• SLVS transmitter support uses emulated LVDS output. • Requires external termination but does not require V_{REF}. • SLVS receiver support uses dedicated 2.5 V LVDS input buffer.
HiSpi	All	—	Yes	Yes	—	• Only input is supported because HiSpi is a unidirectional I/O standard. • Requires external termination but does not require V_{REF}. • HiSpi receiver support uses dedicated 2.5 V LVDS input buffer.

Related Information

MAX 10 High-Speed LVDS Circuitry

The LVDS solution uses the I/O elements and registers in the MAX 10 devices. The Altera Soft LVDS IP core implements the serializer and deserializer as soft SERDES blocks in the core logic.

The MAX 10 devices do not contain dedicated serialization or deserialization circuitry:

- You can use I/O pins and core fabric to implement a high-speed differential interface in the device.
- The MAX 10 solution uses shift registers, internal PLLs, and I/O elements to perform the serial-to-parallel and parallel-to-serial conversions of incoming and outgoing data.
- The Quartus® Prime software uses the parameter settings of the Altera Soft LVDS IP core to automatically construct the differential SERDES in the core fabric.

Figure 2-1: Soft LVDS SERDES

This figure shows a transmitter and receiver block diagram for the soft LVDS SERDES circuitry with the interface signals of the transmitter and receiver data paths.

Related Information

MAX 10 Clocking and PLL User Guide
Provides more information about the PLL and the PLL output counters.

MAX 10 High-Speed LVDS I/O Location

The I/O banks in MAX 10 devices support true LVDS input and emulated LVDS output on all I/O banks. Only the bottom I/O banks support true LVDS output.

Figure 2-2: LVDS Support in I/O Banks of 10M02 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2 and 6.

	TX	RX
LVDS		
Emulated LVDS		
RSDS		
Emulated RSDS		
Mini-LVDS		
Emulated Mini-LVDS		
PPDS		
Emulated PPDS		
BLVDS		
LVPECL		
TMDS		
Sub-LVDS		
SLVS		
HiSpi		
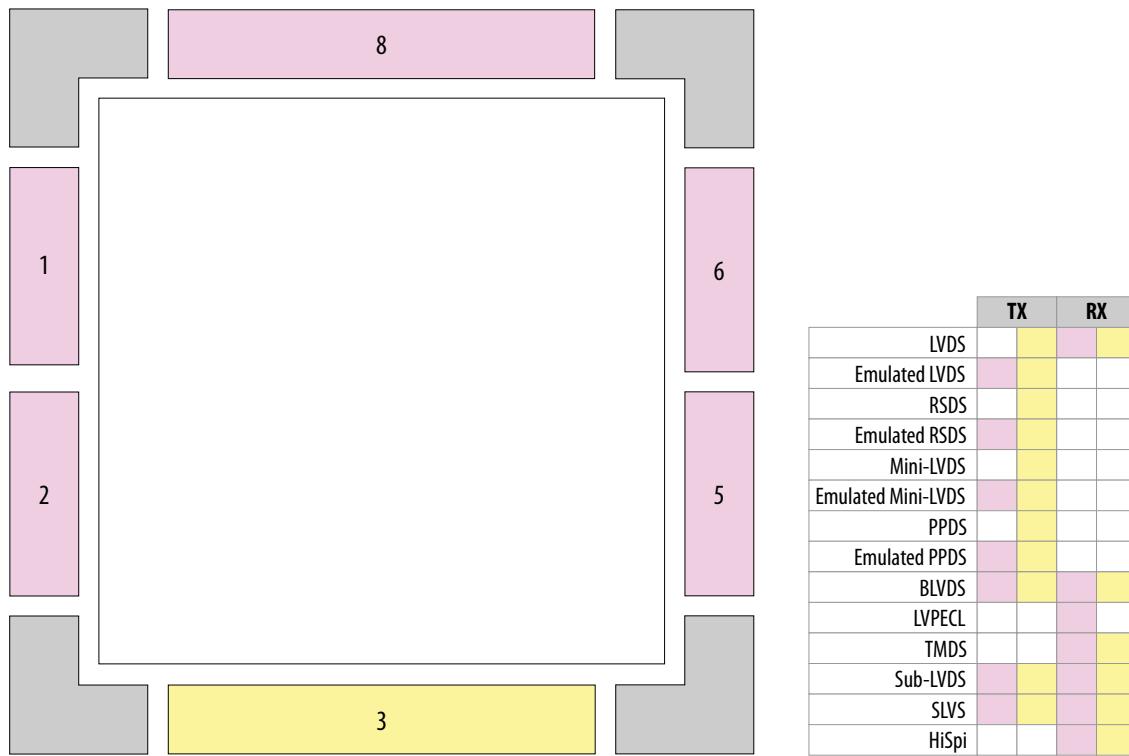
Figure 2-3: LVDS Support in I/O Banks of 10M04 and 10M08 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2 and 6.

	TX	RX
LVDS		
Emulated LVDS		
RSDS		
Emulated RSDS		
Mini-LVDS		
Emulated Mini-LVDS		
PPDS		
Emulated PPDS		
BLVDS		
LVPECL		
TMDS		
Sub-LVDS		
SLVS		
HiSpi		

Send Feedback
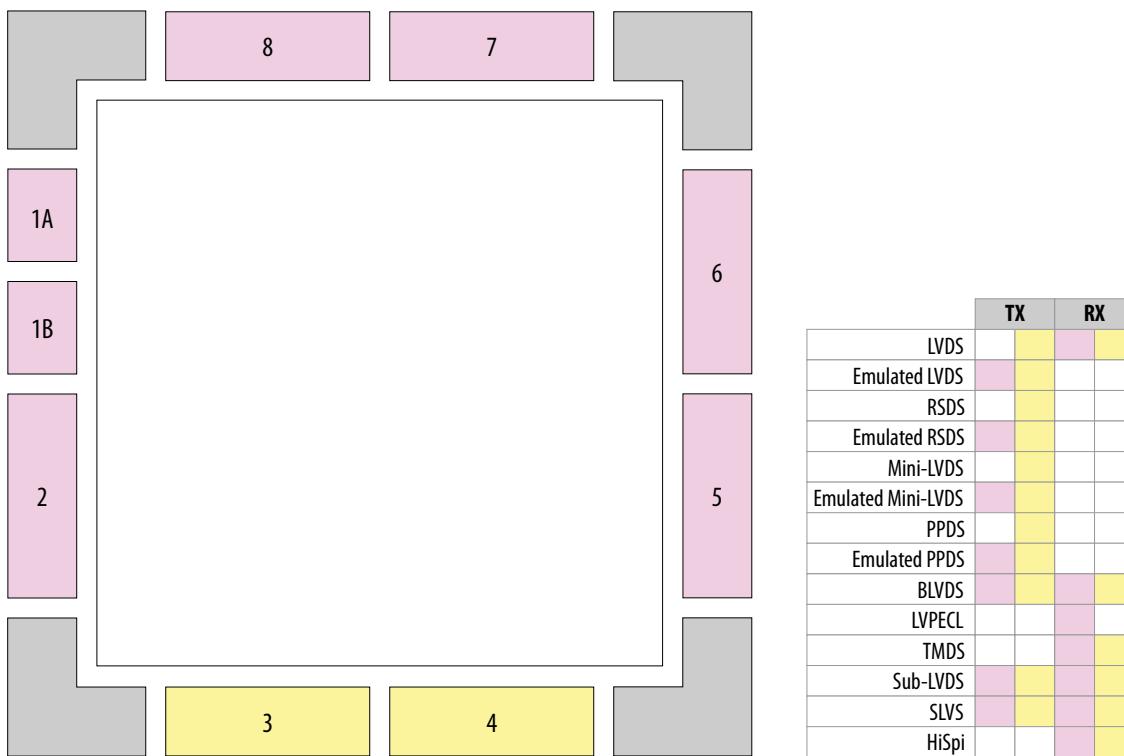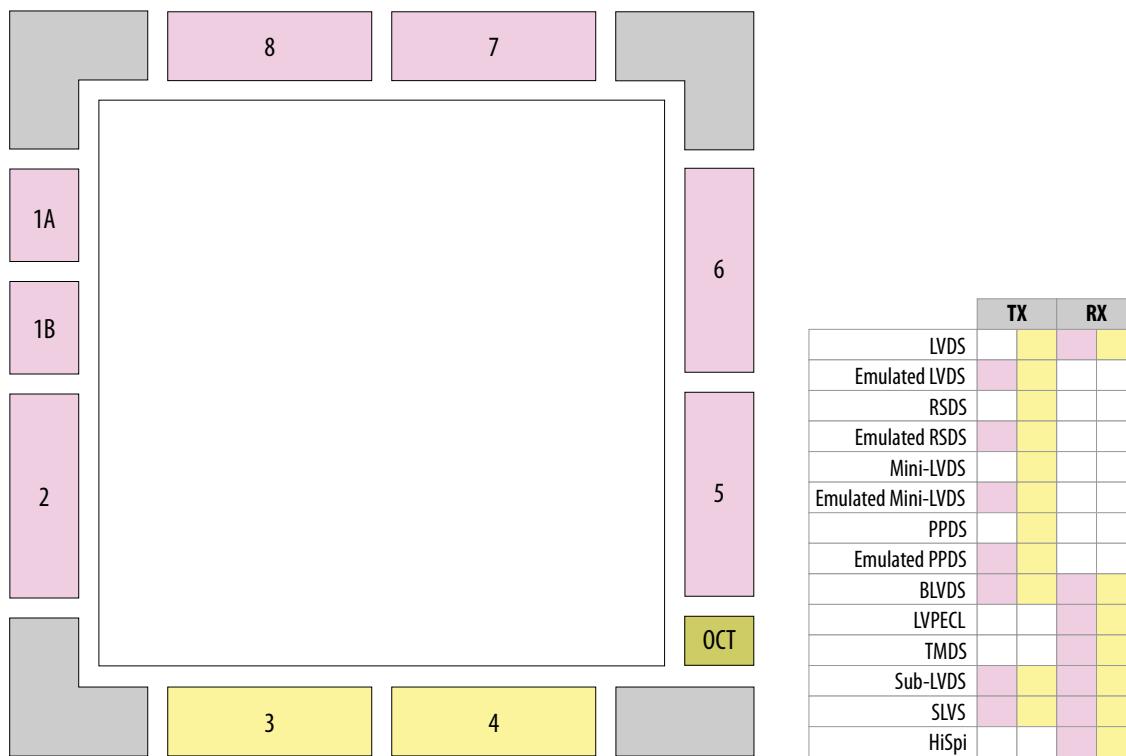
Figure 2-4: LVDS Support in I/O Banks of 10M16, 10M25, 10M40, and 10M50 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2, 3, 6, and 8.

	TX		RX	
LVDS				
Emulated LVDS				
RSDS				
Emulated RSDS				
Mini-LVDS				
Emulated Mini-LVDS				
PPDS				
Emulated PPDS				
BLVDS				
LVPECL				
TMDS				
Sub-LVDS				
SLVS				
HiSpi				

Related Information

- **PLL Specifications**
  Provides PLL performance information for MAX 10 devices.
- **High-Speed I/O Specifications**
  Provides minimum and maximum data rates for different data widths in MAX 10 devices.

Differential I/O Pins in Low Speed Region

Some of the differential I/O pins are located in the low speed region of the MAX 10 device.

- For each user I/O pin (excluding configuration pin) that you place in the low speed region, the Quartus Prime software generates an informational warning message.
- Refer to the device pinout to identify the low speed I/O pins.
- Refer to the device datasheet for the performance information of these I/O pins.

Related Information

- **MAX 10 Device Pin-Out Files**
  Provides pin-out files for each MAX 10 device.
- **MAX 10 Device Datasheet**

- **MAX 10 I/O Banks Locations, MAX 10 General Purpose I/O User Guide**
  Shows the locations of the high speed and low speed I/O banks.

You can implement transmitter-only applications using the MAX 10 LVDS solution. You can use the Altera Soft LVDS IP core to instantiate soft SERDES circuitry. The soft SERDES circuitry works with the clocks and differential I/O pins to create a high-speed differential transmitter circuit.

Related Information

- **MAX 10 High-Speed LVDS I/O Overview** on page 1-1
- **MAX 10 LVDS SERDES I/O Standards Support** on page 2-7
  Lists the supported LVDS I/O standards and the support in different MAX 10 device variants.

High-Speed I/O Transmitter Circuitry

The LVDS transmitter circuitry uses the I/O elements and registers in the MAX 10 devices. The Altera Soft LVDS IP core implements the serializer as a soft SERDES block in the core logic.

Related Information

MAX 10 High-Speed LVDS Circuitry on page 2-10

LVDS Transmitter Programmable I/O Features

You can program some features of the I/O buffers and pins in MAX 10 devices according to your design requirements. For high-speed LVDS transmitter applications, you can program the pre-emphasis setting.

Programmable Pre-Emphasis

The differential output voltage (V_{OD}) setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full V_{OD} level before the next edge, producing pattern-dependent jitter. Pre-emphasis momentarily boosts the output current during switching to increase the output slew rate.

Pre-emphasis increases the amplitude of the high-frequency component of the output signal. This increase compensates for the frequency-dependent attenuation along the transmission line.

The overshoot introduced by the extra current occurs only during change of state switching. This overshoot increases the output slew rate but does not ring, unlike the overshoot caused by signal reflection. The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

ALTERA
now part of Intel

Figure 3-1: LVDS Output with Programmable Pre-Emphasis

Table 3-1: Quartus Prime Software Assignment for Programmable Pre-Emphasis
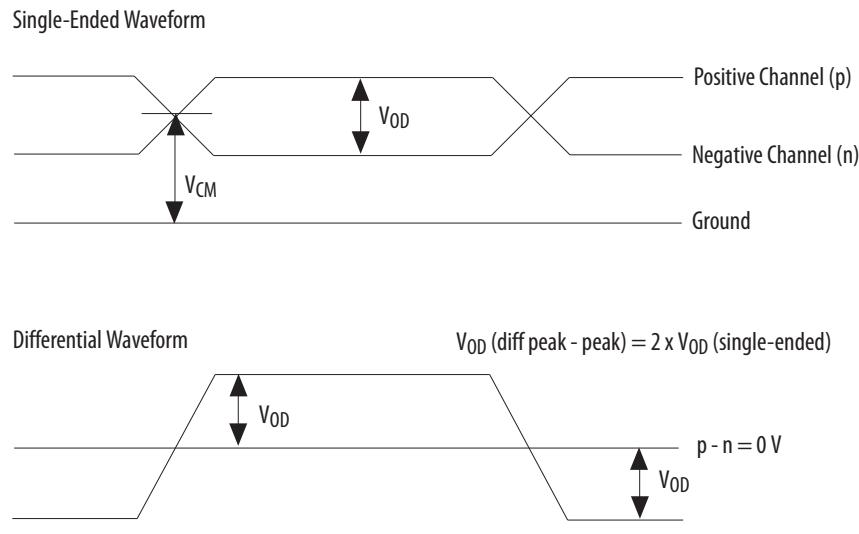
Field	Assignment
To	`tx_out`
Assignment name	Programmable Pre-emphasis
Allowed values	0 (disabled), 1 (enabled). Default is 1.

Programmable Differential Output Voltage

The programmable V_{OD} settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, and a smaller V_{OD} swing reduces power consumption.

Figure 3-2: Differential V_{OD}

This figure shows the V_{OD} of the differential LVDS output.

You can statically adjust the V_{OD} of the differential signal by changing the V_{OD} settings in the Quartus Prime software Assignment Editor.

Table 3-2: Quartus Prime Software Assignment Editor—Programmable V_{OD}

Field	Assignment
To	`tx_out`
Assignment name	Programmable Differential Output Voltage (V_{OD})
Allowed values	0 (low), 1 (medium), 2 (high). Default is 2.

LVDS Transmitter I/O Termination Schemes
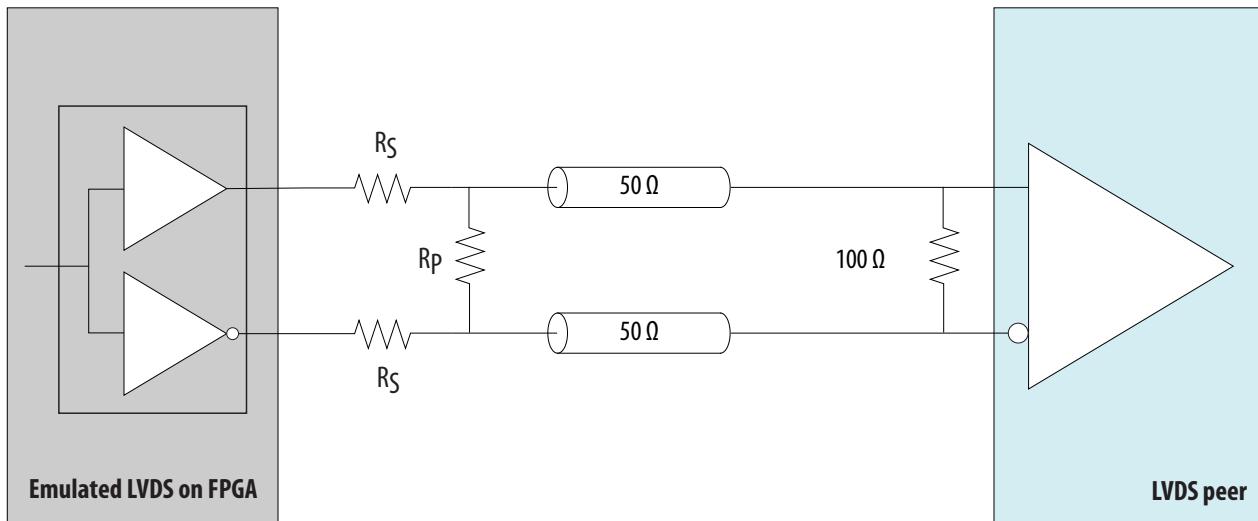
For transmitter applications in MAX 10 devices, you must implement external termination for some I/O standards.

Emulated LVDS External Termination

The emulated LVDS transmitter requires a three-resistor external termination scheme.
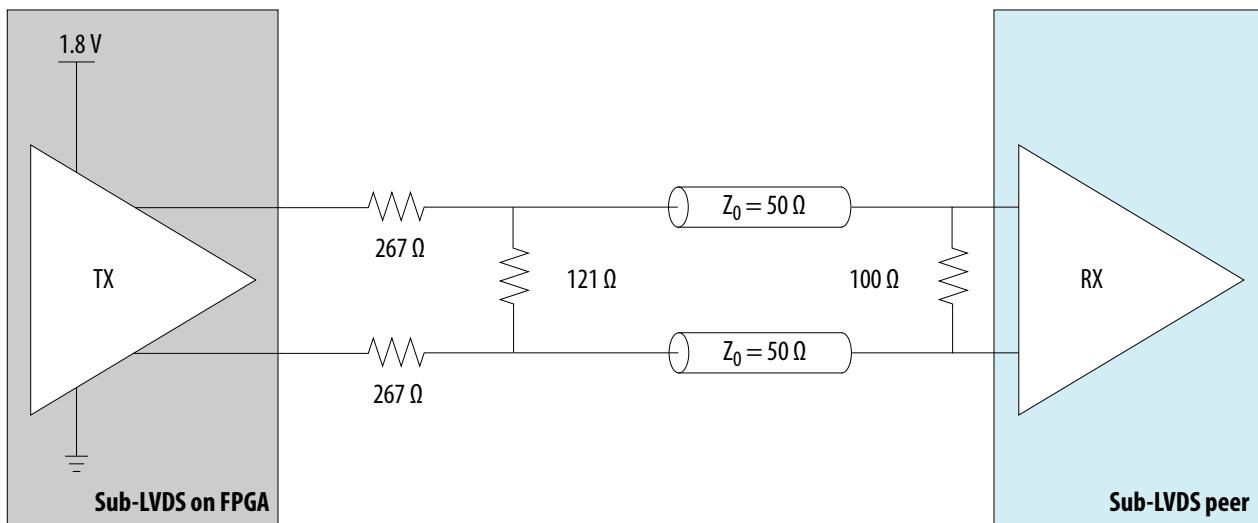
Figure 3-3: External Termination for Emulated LVDS Transmitter

In this figure, $R_S = 120 \ \Omega$ and $R_P = 170 \ \Omega$.
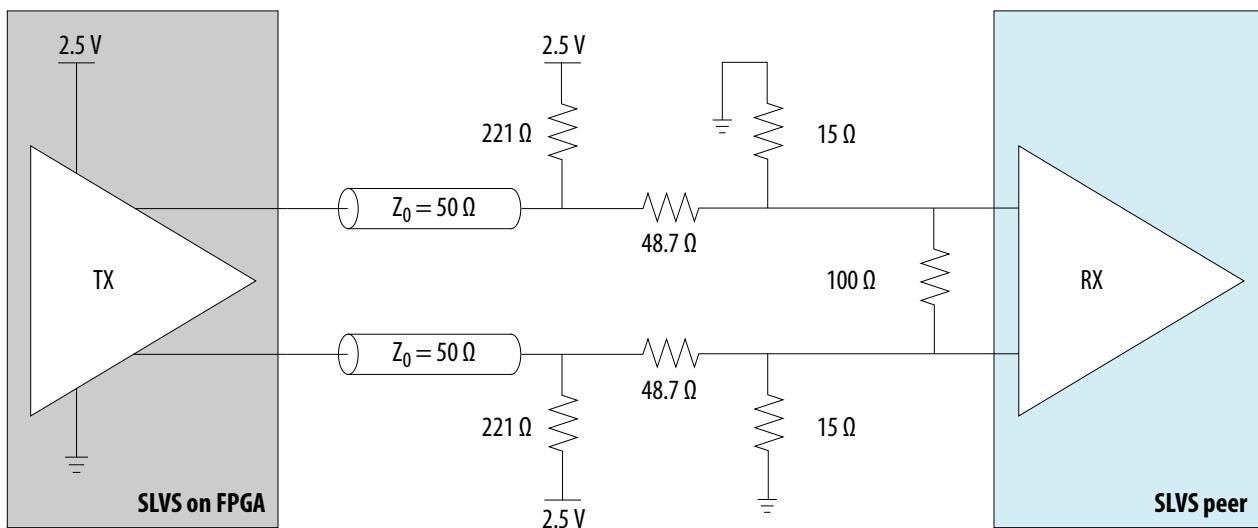
Sub-LVDS Transmitter External Termination

The Sub-LVDS transmitter requires a three-resistor external termination scheme.

Figure 3-4: External Termination for Sub-LVDS Transmitter

SLVS Transmitter External Termination

The SLVS transmitter requires a three-resistor external termination scheme.
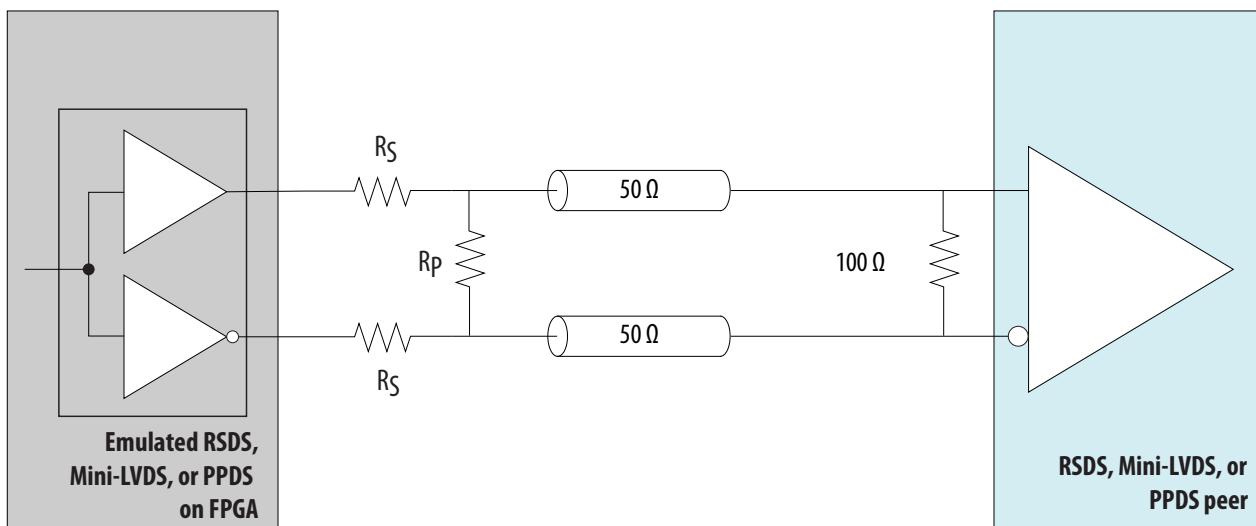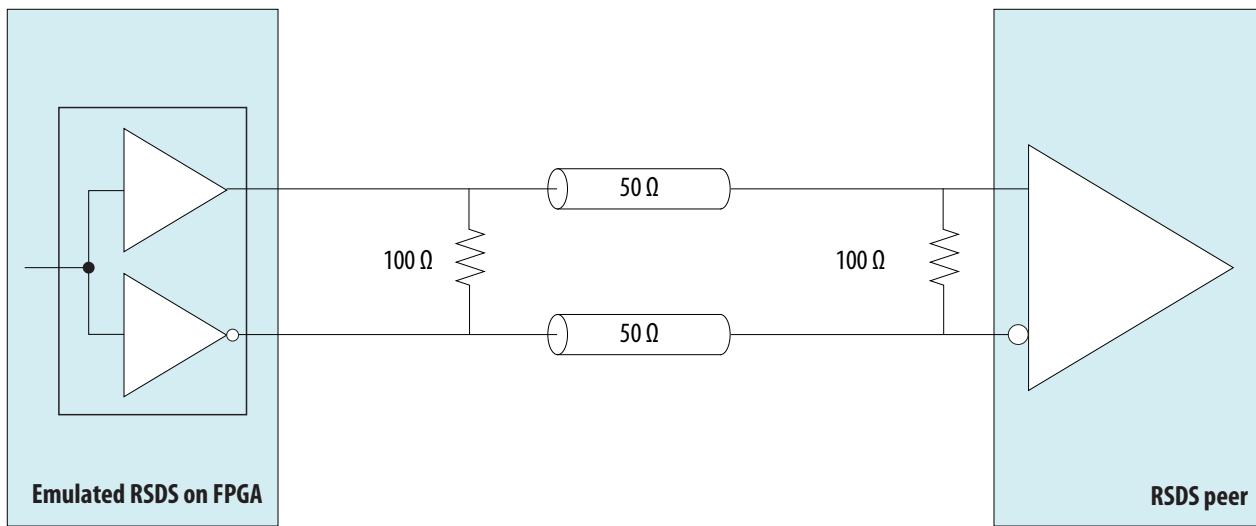
Figure 3-5: External Termination for SLVS Transmitter

Emulated RSDS, Emulated Mini-LVDS, and Emulated PPDS Transmitter External Termination

The emulated RSDS, emulated mini-LVDS, or emulated PPDS transmitter requires a three-resistor external termination scheme. You can also use a single-resistor external termination for the emulated RSDS transmitter.

Figure 3-6: External Termination for Emulated RSDS, Mini-LVDS, or PPDS Transmitter

In this figure, R_S is 120 Ω and R_P is 170 Ω.

Figure 3-7: Single-Resistor External Termination for Emulated RSDS Transmitter

LVDS Transmitter FPGA Design Implementation

MAX 10 devices use a soft SERDES architecture to support high-speed I/O interfaces. The Quartus Prime software creates the SERDES circuits in the core fabric by using the Altera Soft LVDS IP core. To improve the timing performance and support the SERDES, MAX 10 devices use the I/O registers and LE registers in the core fabric.

Altera Soft LVDS IP Core in Transmitter Mode

In the Quartus Prime software, you can design your high-speed transmitter interfaces using the Altera Soft LVDS IP core. This IP core uses the resources optimally in the MAX 10 devices to create the high-speed I/O interfaces.

- You can use the Altera Soft LVDS parameter editor to customize your serializer based on your design requirements.
- The high-speed I/O interface created using the Altera Soft LVDS IP core always sends the most significant bit (MSB) of your parallel data first.

Related Information

- **Altera Soft LVDS Parameter Settings** on page 7-1
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

PLL Source Selection for Altera Soft LVDS IP Core

You can create the LVDS interface components by instantiating the Altera Soft LVDS IP core with an internal or external PLL.

Instantiate Altera Soft LVDS IP Core with Internal PLL

You can set the Altera Soft LVDS IP core to build the SERDES components and instantiate the PLL internally.

- To use this method, turn off the **Use external PLL** option in the **PLL Settings** tab and set the necessary settings in the **PLL Settings** and **Transmitter Settings** tab.
- The Altera Soft LVDS IP core integrates the PLL into the LVDS block.
- The drawback of this method is that you can use the PLL only for the particular LVDS instance.

Instantiate Altera Soft LVDS IP Core with External PLL

You can set the Altera Soft LVDS IP core to build only the SERDES components but use an external PLL source.

- To use this method, turn on the **Use external PLL** option in the **PLL Settings** tab.
- Follow the required clock setting to the input ports as listed in the notification panel.
- You can create your own clocking source using the ALTPLL IP core.
- Use this method to optimize PLL usage with other functions in the core.

Related Information

- **MAX 10 Clocking and PLL User Guide**
  Provides more information about the PLL and the PLL output counters.
- **MAX 10 Clocking and PLL User Guide**
  Provides more information about the PLL and the PLL output counters.

Guidelines: LVDS TX Interface Using External PLL

You can instantiate the Altera Soft LVDS IP core with the **Use External PLL** option. Using external PLL, you can control the PLL settings. For example, you can dynamically reconfigure the PLL to support different data rates and dynamic phase shifts. To use this option, you must instantiate the ALTPLL IP core to generate the various clock signals.

If you turn on the **Use External PLL** option for the Altera Soft LVDS transmitter, you require the following signals from the ALTPLL IP core:

- Serial clock input to the `tx_inclock` port of the Altera Soft LVDS transmitter.
- Parallel clock used to clock the transmitter FPGA fabric logic and connected to the `tx_syncclock` port.

Related Information

MAX 10 Clocking and PLL User Guide

Provides more information about the PLL and the PLL output counters.

ALTPLL Signal Interface with Altera Soft LVDS Transmitter

You can choose any of the PLL output clock ports to generate the LVDS interface clocks.

If you use the ALTPLL IP core as the external PLL source of the Altera Soft LVDS transmitter, use the source-synchronous compensation mode.

Table 3-3: Example: Signal Interface between ALTPLL and Altera Soft LVDS Transmitter

From the ALTPLL IP Core	To the Altera Soft LVDS Transmitter
Fast clock output (`c0`) The fast clock output (`c0`) can only drive `tx_inclock` on the Altera Soft LVDS transmitter.	`tx_inclock`
Slow clock output (`c1`)	`tx_syncclock`

Determining External PLL Clock Parameters for Altera Soft LVDS Transmitter

To determine the ALTPLL IP core clock parameter for the Altera Soft LVDS IP core transmitter, follow these steps in your design:

1. Instantiate the Altera Soft LVDS IP core transmitter using internal PLL.
2. Compile the design up to TimeQuest timing analysis.
3. In the **Table of Contents** section of the **Compilation Report** window, navigate to **TimeQuest Timing Analyzer** > **Clocks**.
4. Note the clock parameters used by the internal PLL for the Altera Soft LVDS IP core transmitter.

   In the list of clocks, `clk0` is the fast clock.

Figure 3-8: Clock Parameters Example for Altera Soft LVDS Transmitter

	Clock Name	Type	Period	Frequency	Duty Cycle	Phase
1	inst|lvdstx_inst|lvds_tx_pll|clk[0]	Generated	2.500	400.0 MHz	50.00	-90.0
2	inst|lvdstx_inst|lvds_tx_pll|clk[1]	Generated	12.500	80.0 MHz	50.00	-18.0
3	tx_inclock	Base	10.000	100.0 MHz		

Configure the ALTPLL output clocks with the parameters you noted in this procedure and connect the clock outputs to the correct Altera Soft LVDS clock input ports.

Initializing the Altera Soft LVDS IP Core

The PLL locks to the reference clock before the Altera Soft LVDS IP core implements the SERDES blocks for data transfer.

During device initialization the PLL starts to lock to the reference clock and becomes operational when it achieves lock during user mode. If the clock reference is not stable, it corrupts the phase shifts of the PLL output clock. This phase shifts corruption can cause failure and corrupt data transfer between the high-speed LVDS domain and the low-speed parallel domain.

To avoid data corruption, follow these steps when initializing the Altera Soft LVDS IP core:

1. Assert the `pll_areset` signal for at least 10 ns.
2. After at least 10 ns, deassert the `pll_areset` signal.
3. Wait until the PLL lock becomes stable.

   After the PLL lock port asserts and is stable, the SERDES blocks are ready for operation.

High-Speed I/O Timing Budget

The LVDS I/O standard enables high-speed transmission of data that results in better overall system performance. To take advantage of fast system performance, you must analyze the timing for high-speed signals. Timing analysis for the differential block is different than traditional synchronous timing analysis techniques.

The basis of the source synchronous timing analysis is the skew between the data and the clock signals instead of the clock-to-output setup times. High-speed differential data transmission requires the use of timing parameters provided by IC vendors and is strongly influenced by board skew, cable skew, and clock jitter.

Transmitter Channel-to-Channel Skew

The receiver input skew margin (RSKM) calculation uses the transmitter channel-to-channel skew (TCCS)—an important parameter based on the MAX 10 transmitter in a source-synchronous differential interface. You can get the TCCS value from the device datasheet.

Related Information
MAX 10 Device Datasheet

Guidelines: LVDS Transmitter Channels Placement

To maintain an acceptable noise level on the V_{CCIO} supply, observe the placement restrictions for single-ended I/O pins in relation to differential pads.

Altera recommends that you create a Quartus Prime design, specify your device I/O assignments, and compile your design to validate your pin placement. The Quartus Prime software verifies your pin connections against the I/O assignment and placement rules to ensure that the device will operate properly.

You can use the Quartus Prime Pin Planner Package view to ease differential I/O assignment planning:

- On the **View** menu, click **Show Differential Pin Pair Connections** to highlight the differential pin pairing. The differential pin pairs are connected with red lines.
- For differential pins, you only need to assign the signal to a positive pin. The Quartus Prime software automatically assigns the negative pin if the positive pin is assigned with a differential I/O standard.

In MAX 10 devices, the routing of each differential pin pair is matched. Consequently, the skew between the positive and the negative pins is minimal. The internal routes of both pins in a differential pair are matched even if the pins are non-adjacent.

The Altera Soft LVDS IP core for MAX 10 devices supports a maximum of 18 channels per IP instantiation. Each channel can support deserialization factor (parallel data width) from one to ten bits. When you are grouping channels for an application, you must consider the channel to channel skew during Fitter placement. To minimize skew, place all LVDS channels in the group side by side. For your PCB design, Altera recommends that you perform package skew compensation to minimize skew and maximize performance.

Note: For MAX 10 devices, the Quartus Prime software does not provide a package skew compensation report.

Guidelines: LVDS Channels PLL Placement

Each PLL in the MAX 10 device can drive only the LVDS channels in I/O banks on the same edge as the PLL.

Table 3-4: Examples of Usable PLL to Drive I/O Banks in MAX 10 Devices

I/O Bank Edge	Input `refclk`	`GCLK mux`	Usable PLL
Left	Left	Left	Top left or bottom left
Bottom	Bottom	Bottom	Bottom left or bottom right
Right	Right	Right	Top right or bottom right
Top	Top	Top	Top left or top right

Guidelines: LVDS Transmitter Logic Placement

The Quartus Prime software automatically optimizes the SERDES logic placement to meet the timing requirements. Therefore, you do not have to perform placement constraints on the Altera Soft LVDS IP core logic.

To improve the performance of the Quartus Prime Fitter, you can create LogicLock™ regions in the device floorplan to confine the transmitter SERDES logic placement.

- The TCCS parameter is guaranteed per datasheet specification to the entire bank of differential I/Os that are located in the same side. This guarantee applies if the transmitter SERDES logic is placed within the LAB adjacent to the output pins.
- Constrain the transmitter SERDES logic to the LAB adjacent to the data output pins and clock output pins to improve the TCCS performance.

Related Information

Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design chapter, Volume 1: Design and Synthesis, Quartus Prime Handbook

Provides step by step instructions about creating a design floorplan with LogicLock location assignments.

Guidelines: Enable LVDS Pre-Emphasis for E144 Package

For MAX 10 devices in the E144 package, Altera recommends that you enable LVDS pre-emphasis to achieve optimum signal integrity (SI) performance. If you do not enable pre-emphasis, undesirable SI condition may be induced in the device resulting in LVDS eye height sensitivity.

LVDS Transmitter Debug and Troubleshooting

You can obtain useful information about the LVDS interface performance with board-level verification using the FPGA prototype.

Although the focus of the board-level verifcation is to verify the FPGA functionality in your end system, you can take additional steps to examine the margins. Using oscilloscopes, you can examine the margins to verify the predicted size of the data-valid window, and the setup and hold margins at the I/O interface.

You can also use the Altera SignalTap® II Logic Analyzer to perform system level verification to correlate the system against your design targets.

Related Information

In-System Debugging Using External Logic Analyzers chapter, Volume 3: Verification, Quartus Prime Handbook

Perform RTL Simulation Before Hardware Debug

Before you debug on hardware, Altera recommends that you perform an RTL simulation. Using the RTL simulation, you can check the code functionality before testing in real hardware.

For example, you can use the RTL simulation to verify that when you send a training pattern from a remote transmitter, the bitslipping mechanism in your LVDS receiver works.

Geometry-Based and Physics-Based I/O Rules

You need to consider the I/O placement rules related to LVDS. The Quartus Prime software generates critical warning or error messages if the I/O placements rules are violated.

For more information, refer to the related information.

Related Information
MAX 10 General Purpose I/O User Guide

You can implement receiver-only applications using the MAX 10 LVDS solution. You can use the Altera Soft LVDS IP core to instantiate soft SERDES circuitry. The soft SERDES circuitry works with the clocks and differential I/O pins to create a high-speed differential receiver circuit.

Related Information

- **MAX 10 High-Speed LVDS I/O Overview** on page 1-1
- **MAX 10 LVDS SERDES I/O Standards Support** on page 2-7
  Lists the supported LVDS I/O standards and the support in different MAX 10 device variants.

High-Speed I/O Receiver Circuitry

The LVDS receiver circuitry uses the I/O elements and registers in the MAX 10 devices. The deserializer is implemented in the core logic as a soft SERDES blocks.

In the receiver mode, the following blocks are available in the differential receiver datapath:

- Deserializer
- Data realignment block (bit slip)
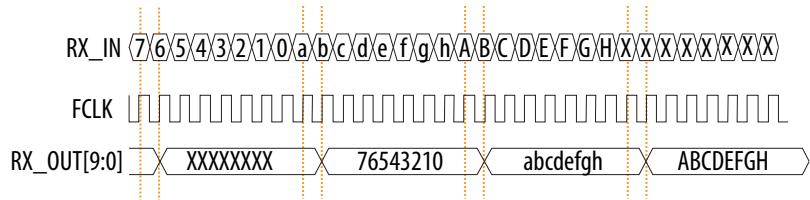
Related Information

MAX 10 High-Speed LVDS Circuitry on page 2-10

Soft Deserializer

The soft deserializer converts a 1-bit serial data stream into a parallel data stream based on the deserialization factor.

Figure 4-1: LVDS x8 Deserializer Waveform

now part of Intel

Signal	Description
rx_in	LVDS data stream, input to the Altera Soft LVDS channel.
fclk	Clock used for receiver.
loaden	Enable signal for deserialization generated by the Altera Soft LVDS IP core.
rx_out[9:0]	Deserialized data.

Data Realignment Block (Bit Slip)

Skew in the transmitted data and skew added by the transmision link cause channel-to-channel skew on the received serial data streams. To compensate for channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel contains a data realignment circuit. The data realignment circuit realigns the data by inserting bit latencies into the serial stream.

To align the data manually, use the data realignment circuit to insert a latency of one RxFCLK cycle . The data realignment circuit slips the data one bit for every RX_DATA_ALIGN pulse. You must wait at least two core clock cycles before checking to see if the data is aligned. This wait is necessary because it takes at least two core clock cycles to purge the corrupted data.
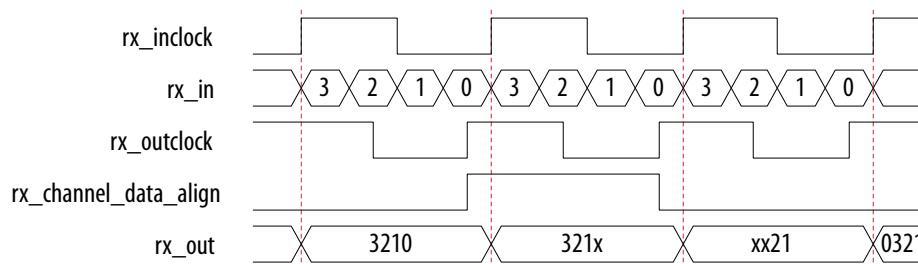
An optional RX_CHANNEL_DATA_ALIGN port controls the bit insertion of each receiver independently of the internal logic. The data slips one bit on the rising edge of RX_CHANNEL_DATA_ALIGN.

The RX_CHANNEL_DATA_ALIGN signal has these requirements:

- The minimum pulse width is one period of the parallel clock in the logic array.
- The minimum low time between pulses is one period of the parallel clock.
- The signal is edge-triggered.
- The valid data is available two parallel clock cycles after the rising edge of RX_CHANNEL_DATA_ALIGN.

Figure 4-2: Data Realignment Timing

This figure shows receiver output (RX_OUT) after one bit slip pulse with the deserialization factor set to **4**.
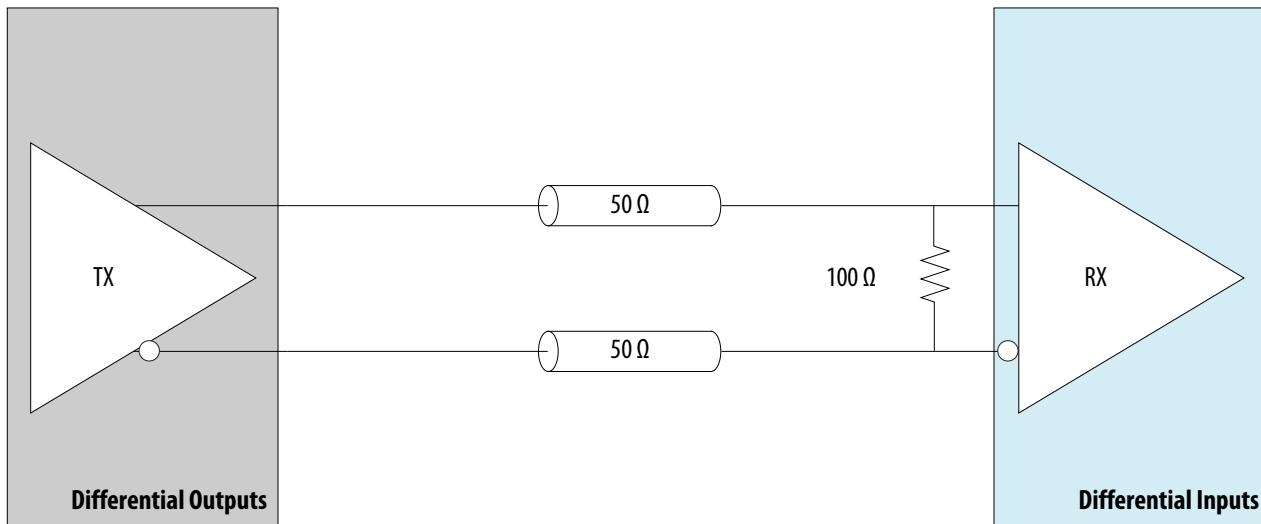
LVDS Receiver I/O Termination Schemes

All LVDS receiver channels require termination to achieve better signal quality and ensure impedance matching with the transmission line and driver.

LVDS, Mini-LVDS, and RSDS Receiver External Termination

The LVDS, mini-LVDS, or RSDS receiver requires a single resistor external termination scheme.
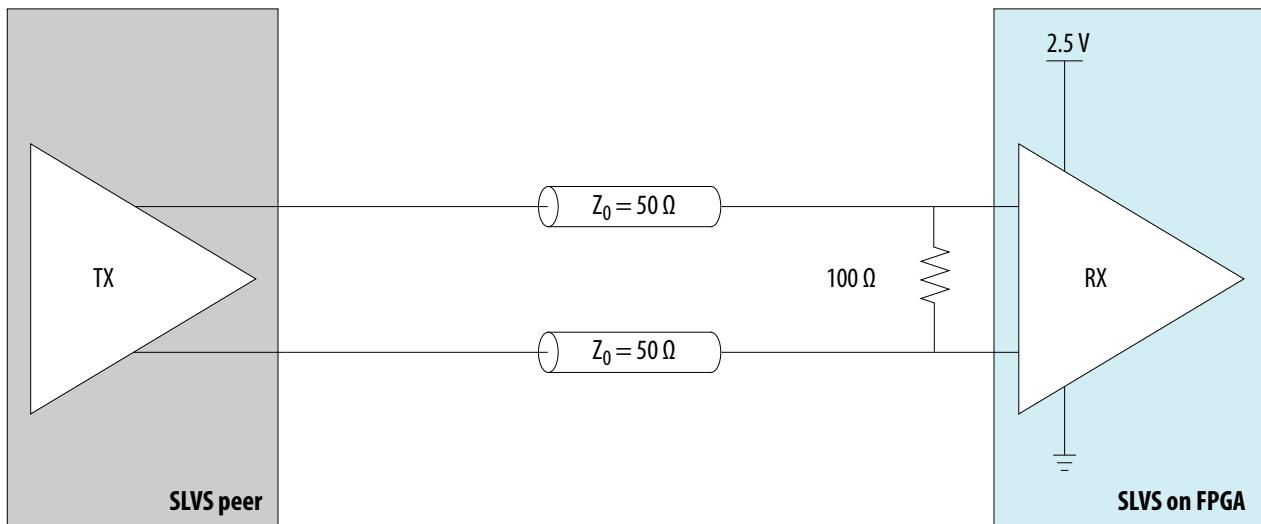
Figure 4-3: External Termination for LVDS I/O Standard

SLVS Receiver External Termination
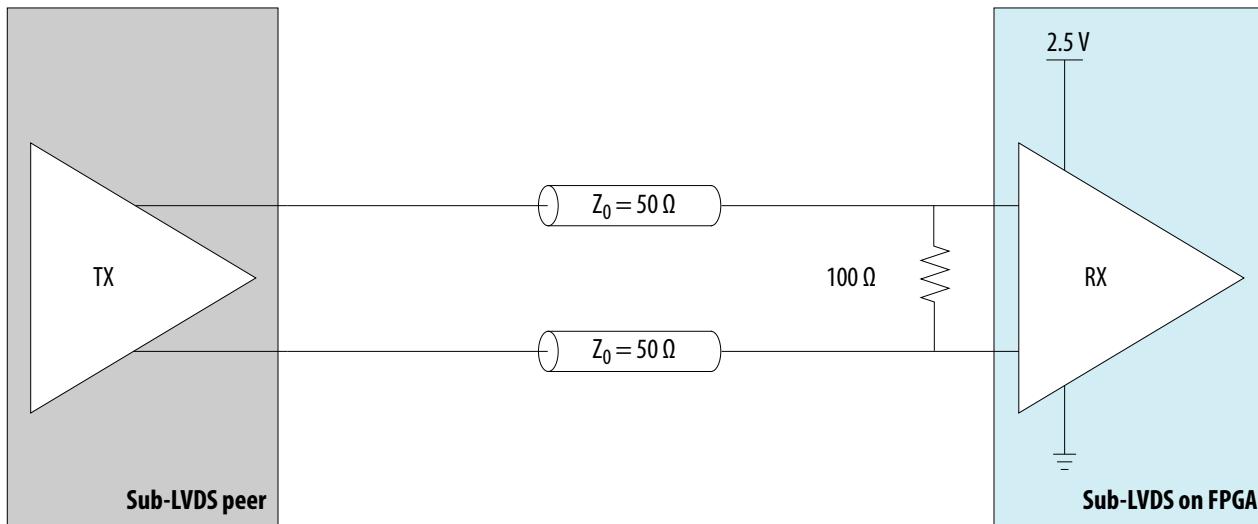
The SLVS receiver requires a single-resistor external termination scheme.

Figure 4-4: External Termination for SLVS Receiver

Sub-LVDS Receiver External Termination
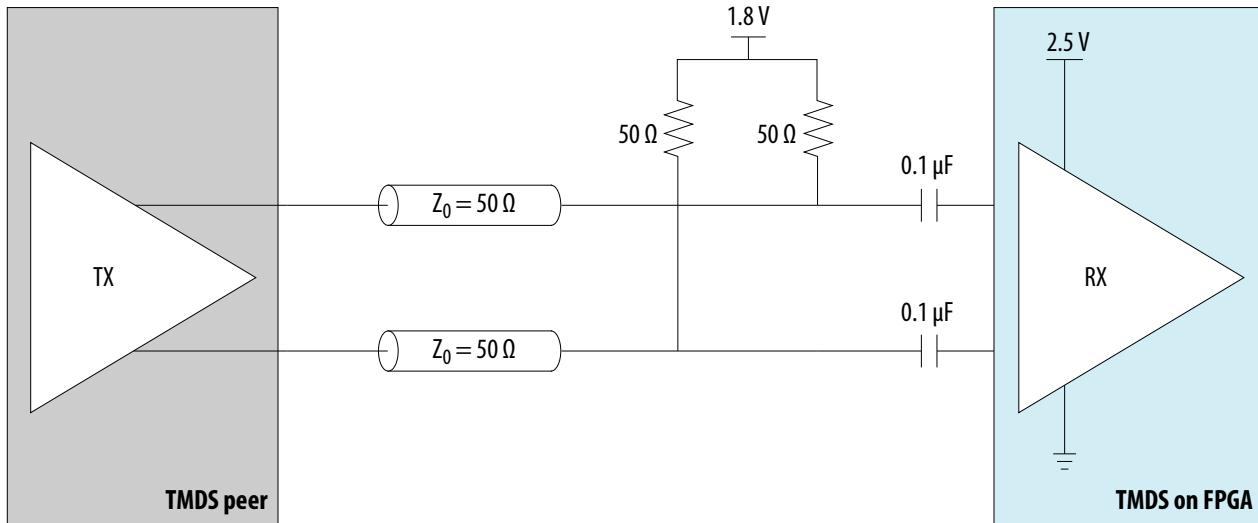
The Sub-LVDS receiver requires a single-resistor external termination scheme.

Send Feedback

Figure 4-5: External Termination for Sub-LVDS Receiver

TMDS Receiver External Termination
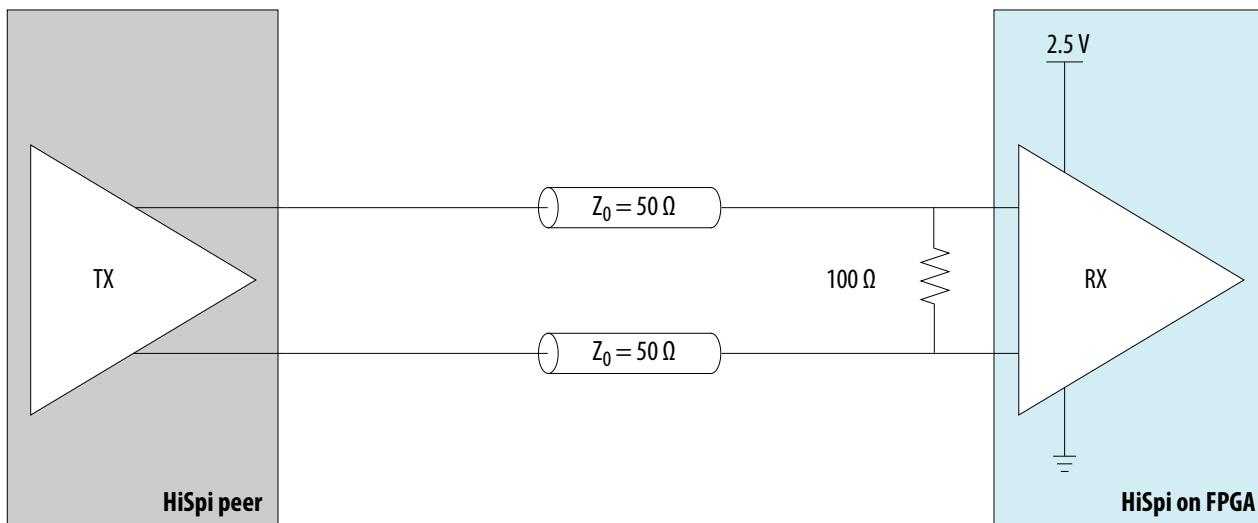
Figure 4-6: External Termination for TMDS Receiver

This diagram shows the external level shifter that is required for the TMDS input standards support in MAX 10 devices.

HiSpi Receiver External Termination

The HiSpi receiver requires a single-resistor external termination scheme.

Figure 4-7: External Termination for HiSpi Receiver
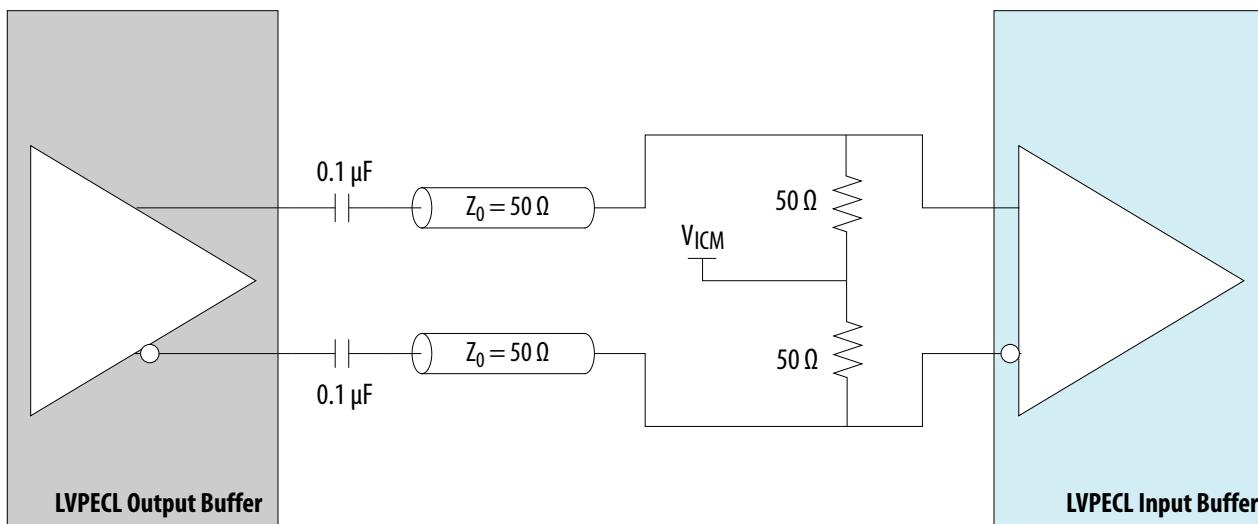
LVPECL External Termination

The MAX 10 devices support the LVPECL I/O standard on input clock pins only.
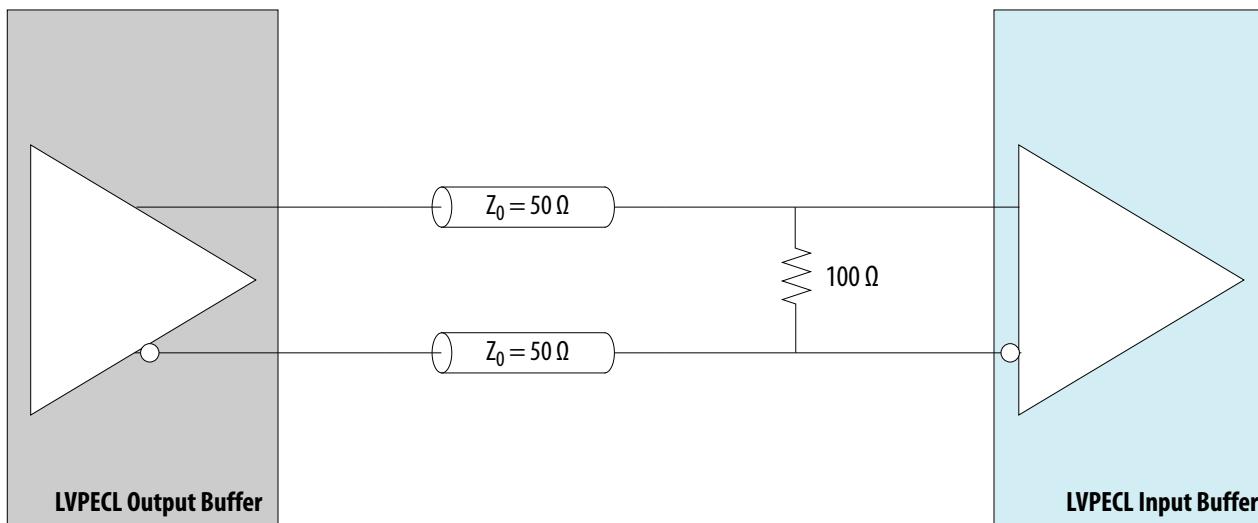
- LVDS input buffers support LVPECL input operation.
- LVPECL output operation is not supported.

Use AC coupling if the LVPECL common-mode voltage of the output buffer does not match the LVPECL input common-mode voltage.

Note: Altera recommends that you use IBIS models to verify your LVPECL AC/DC-coupled termination.

Figure 4-8: LVPECL AC-Coupled Termination

Support for DC-coupled LVPECL is available if the LVPECL output common mode voltage is within the MAX 10 LVPECL input buffer specification.

Send Feedback

Figure 4-9: LVPECL DC-Coupled Termination

For information about the V_{ICM} specification, refer to the device datasheet.

Related Information
MAX 10 Device Datasheet

LVDS Receiver FPGA Design Implementation

MAX 10 devices use a soft SERDES architecture to support high-speed I/O interfaces. The Quartus Prime software creates the SERDES circuits in the core fabric by using the Altera Soft LVDS IP core. To improve the timing performance and support the SERDES, MAX 10 devices use the I/O registers and LE registers in the core fabric.

Altera Soft LVDS IP Core in Receiver Mode

In the Quartus Prime software, you can design your high-speed receiver interfaces using the Altera Soft LVDS IP core. This IP core uses the resources in the MAX 10 devices optimally to create the high-speed I/O interfaces.

- You can use the Altera Soft LVDS parameter editor to customize your deserializer based on your design requirements.
- The Altera Soft LVDS IP core implements the high-speed deserializer in the core fabric.

Related Information

- **Altera Soft LVDS Parameter Settings** on page 7-1
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.

- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

PLL Source Selection for Altera Soft LVDS IP Core

You can create the LVDS interface components by instantiating the Altera Soft LVDS IP core with an internal or external PLL.

Instantiate Altera Soft LVDS IP Core with Internal PLL

You can set the Altera Soft LVDS IP core to build the SERDES components and instantiate the PLL internally.

- To use this method, turn off the **Use external PLL** option in the **PLL Settings** tab.
- The Altera Soft LVDS IP core integrates the PLL into the LVDS block.
- The drawback of this method is that you can use the PLL only for the particular LVDS instance.

Instantiate Altera Soft LVDS IP Core with External PLL

You can set the Altera Soft LVDS IP core to build only the SERDES components but use an external PLL source.

- To use this method, turn on the **Use external PLL** option in the **PLL Settings** tab.
- Follow the required clock setting to the input ports as listed in the notification panel.
- You can create your own clocking source using the ALTPLL IP core.
- Use this method to optimize PLL usage with other functions in the core.

Related Information

- **MAX 10 Clocking and PLL User Guide**
  Provides more information about the PLL and the PLL output counters.
- **MAX 10 Clocking and PLL User Guide**
  Provides more information about the PLL and the PLL output counters.

Guidelines: LVDS RX Interface Using External PLL

You can instantiate the Altera Soft LVDS IP core with the **Use External PLL** option. Using external PLL, you can control the PLL settings. For example, you can dynamically reconfigure the PLL to support different data rates and dynamic phase shifts. To use this option, you must instantiate the ALTPLL IP core to generate the various clock signals.

If you turn on the **Use External PLL** option for the Altera Soft LVDS receiver, you require the following signals from the ALTPLL IP core:

- Serial clock input to the `rx_inclock` port of the Altera Soft LVDS receiver.
- Parallel clock used to clock the receiver FPGA fabric logic.
- The `locked` signal for Altera Soft LVDS PLL reset port.

Related Information
MAX 10 Clocking and PLL User Guide
Provides more information about the PLL and the PLL output counters.

ALTPLL Signal Interface with Altera Soft LVDS Receiver

You can choose any of the PLL output clock ports to generate the LVDS interface clocks.

Send Feedback

If you use the ALTPLL IP core as the external PLL source of the Altera Soft LVDS receiver, use the source-synchronous compensation mode.

Table 4-1: Example: Signal Interface Between ALTPLL and Altera Soft LVDS Receiver with Even Deserialization Factor

From the ALTPLL IP Core	To the Altera Soft LVDS Receiver
Fast clock output (`c0`) The serial clock output (`c0`) can only drive `rx_inclock` on the Altera Soft LVDS receiver.	`rx_inclock`
Slow clock output (`c1`)	`rx_syncclock`

Table 4-2: Example: Signal Interface Between ALTPLL and Altera Soft LVDS Receiver with Odd Deserialization Factor

From the ALTPLL IP Core	To the Altera Soft LVDS Receiver
Fast clock output (`c0`) The serial clock output (`c0`) can only drive `rx_inclock` on the Altera Soft LVDS receiver.	`rx_inclock`
Slow clock output (`c1`)	`rx_syncclock`
Read clock (`c2`) output from the PLL	`rx_readclock` (clock input port for reading operation from RAM buffer and read counter)

Determining External PLL Clock Parameters for Altera Soft LVDS Receiver

To determine the ALTPLL IP core clock parameter for the Altera Soft LVDS IP core receiver, follow these steps in your design:

1. Instantiate the Altera Soft LVDS IP core receiver using internal PLL.
2. Compile the design up to TimeQuest timing analysis.
3. In the **Table of Contents** section of the **Compilation Report** window, navigate to **TimeQuest Timing Analyzer** > **Clocks**.
4. Note the clock parameters used by the internal PLL for the Altera Soft LVDS IP core receiver.

   In the list of clocks, `clk[0]` is the fast clock, `clk[1]` is the slow clock, and `clk[2]` is the read clock.

Figure 4-10: Clock Parameters Example for Altera Soft LVDS Receiver

	Clock Name	Type	Period	Frequency	Duty Cycle	Phase
1	inst|lvdsrx_inst|lvds_rx_pll|clk[0]	Generated	2.500	400.0 MHz	50.00	-90.0
2	inst|lvdsrx_inst|lvds_rx_pll|clk[1]	Generated	12.500	80.0 MHz	50.00	-18.0
3	inst|lvdsrx_inst|lvds_rx_pll|clk[2]	Generated	6.250	160.0 MHz	50.00	-36.0
4	rx_inclock	Base	10.000	100.0 MHz		

Configure the ALTPLL output clocks with the parameters you noted in this procedure and connect the clock outputs to the correct Altera Soft LVDS clock input ports.

Initializing the Altera Soft LVDS IP Core

The PLL locks to the reference clock before the Altera Soft LVDS IP core implements the SERDES blocks for data transfer.

During device initialization the PLL starts to lock to the reference clock and becomes operational when it achieves lock during user mode. If the clock reference is not stable, it corrupts the phase shifts of the PLL output clock. This phase shifts corruption can cause failure and corrupt data transfer between the high-speed LVDS domain and the low-speed parallel domain.

To avoid data corruption, follow these steps when initializing the Altera Soft LVDS IP core:

1. Assert the `pll_areset` signal for at least 10 ns.
2. After at least 10 ns, deassert the `pll_areset` signal.
3. Wait until the PLL lock becomes stable.

   After the PLL lock port asserts and is stable, the SERDES blocks are ready for operation.

High-Speed I/O Timing Budget

The LVDS I/O standard enables high-speed transmission of data that results in better overall system performance. To take advantage of fast system performance, you must analyze the timing for high-speed signals. Timing analysis for the differential block is different than traditional synchronous timing analysis techniques.

The basis of the source synchronous timing analysis is the skew between the data and the clock signals instead of the clock-to-output setup times. High-speed differential data transmission requires the use of timing parameters provided by IC vendors and is strongly influenced by board skew, cable skew, and clock jitter.

Receiver Input Skew Margin

Use RSKM, TCCS, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver data path.

Figure 4-11: RSKM Equation

$$RSKM = \frac{TUI - SW - TCCS}{2}$$

Conventions used for the equation:

- RSKM—the timing margin between the clock input of the receiver and the data input sampling window, and the jitter induced from core noise and I/O switching noise.
- Time unit interval (TUI)—time period of the serial data.
- SW—the period of time that the input data must be stable to ensure that the LVDS receiver samples the data successfully. The SW is a device property and varies according to device speed grade.
- TCCS—the timing difference between the fastest and the slowest output edges across channels driven by the same PLL. This measurement includes the t_{CO} variation, clock, and clock skew.

You must calculate the RSKM value, based on the data rate and device, to determine if the LVDS receiver can sample the data:

- A positive RSKM value, after deducting transmitter jitter, indicates that the LVDS receiver can sample the data properly.
- A negative RSKM, after deducting transmitter jitter, indicates that it cannot sample the data properly.

Figure 4-12: Differential High-Speed Timing Diagram and Timing Budget

RSKM Report for LVDS Receiver

For LVDS receivers, the Quartus Prime software provides an RSKM report showing the SW, TUI, and RSKM values.

- To generate the RSKM report, run the `report_RSKM` command in the TimeQuest Timing Analyzer. The RSKM report is available in the TimeQuest Timing Analyzer section of the Quartus Prime compilation report.
- To obtain a more realistic RSKM value, assign the input delay to the LVDS receiver through the constraints menu of the TimeQuest Timing Analyzer. The input delay is determined according to the data arrival time at the LVDS receiver port, with respect to the reference clock.
- If you set the input delay in the settings parameters for the **Set Input Delay** option, set the clock name to the clock that references the source-synchronous clock that feeds the LVDS receiver.
- If you do not set any input delay in the TimeQuest Timing Analyzer, the receiver channel-to-channel skew defaults to zero.
- You can also directly set the input delay in a Synopsys Design Constraint file (**.sdc**) by using the `set_input_delay` command.

Example: RSKM Calculation

This example shows the RSKM calculation for FPGA devices at 1 Gbps data rate with a 200 ps board channel-to-channel skew.

- TCCS = 100 ps
- SW = 300 ps
- TUI = 1000 ps
- Total RCCS = TCCS + Board channel-to-channel skew = 100 ps + 200 ps = 300 ps
- RSKM = (TUI – SW – RCCS) / 2 = (1000 ps – 300 ps – 300 ps) / 2 = 200 ps

If the RSKM is greater than 0 ps after deducting transmitter jitter, the receiver will work correctly.

Guidelines: Floating LVDS Input Pins

You can implement floating LVDS input pins in MAX 10 devices.

For floating LVDS input pins, apply a 100 Ω differential resistance across the P and N legs of the LVDS receiver. You can use external termination.

If you use floating LVDS input pins, Altera recommends that you use external biasing schemes to reduce noise injection and current consumption.

Guidelines: LVDS Receiver Channels Placement

To maintain an acceptable noise level on the V_{CCIO} supply, observe the placement restrictions for single-ended I/O pins in relation to differential pads.

Altera recommends that you create a Quartus Prime design, specify your device I/O assignments, and compile your design to validate your pin placement. The Quartus Prime software verifies your pin connections against the I/O assignment and placement rules to ensure that the device will operate properly.

You can use the Quartus Prime Pin Planner Package view to ease differential I/O assignment planning:

- On the **View** menu, click **Show Differential Pin Pair Connections** to highlight the differential pin pairing. The differential pin pairs are connected with red lines.
- For differential pins, you only need to assign the signal to a positive pin. The Quartus Prime software automatically assigns the negative pin if the positive pin is assigned with a differential I/O standard.

In MAX 10 devices, the routing of each differential pin pair is matched. Consequently, the skew between the positive and the negative pins is minimal. The internal routes of both pins in a differential pair are matched even if the pins are non-adjacent.

The Altera Soft LVDS IP core for MAX 10 devices supports a maximum of 18 channels per IP instantiation. Each channel can support deserialization factor (parallel data width) from one to ten bits. When you are grouping channels for an application, you must consider the channel to channel skew during Fitter placement. To minimize skew, place all LVDS channels in the group side by side. For your PCB design, Altera recommends that you perform package skew compensation to minimize skew and maximize performance.

Note: For MAX 10 devices, the Quartus Prime software does not provide a package skew compensation report.

Guidelines: LVDS Channels PLL Placement

Each PLL in the MAX 10 device can drive only the LVDS channels in I/O banks on the same edge as the PLL.

Table 4-3: Examples of Usable PLL to Drive I/O Banks in MAX 10 Devices

I/O Bank Edge	Input `refclk`	`GCLK mux`	Usable PLL
Left	Left	Left	Top left or bottom left
Bottom	Bottom	Bottom	Bottom left or bottom right
Right	Right	Right	Top right or bottom right
Top	Top	Top	Top left or top right

Guidelines: LVDS Receiver Logic Placement

The Quartus Prime software automatically optimizes the SERDES logic placement to meet the timing requirements. Therefore, you do not have to perform placement constraints on the Altera Soft LVDS IP core logic.

To improve the performance of the Quartus Prime Fitter, you can create LogicLock regions in the device floorplan to confine the transmitter SERDES logic placement.

- The TCCS parameter is guaranteed per datasheet specification to the entire bank of differential I/Os that are located in the same side. This guarantee applies if the transmitter SERDES logic is placed within the LAB adjacent to the output pins.
- Constrain the transmitter SERDES logic to the LAB adjacent to the data output pins and clock output pins to improve the TCCS performance.

Guidelines: LVDS Receiver Timing Constraints

For receiver designs that uses the core logic to implement the SERDES circuits, you must set proper timing constraints.

For LVDS receiver data paths where the PLL operation is in source-synchronous compensation mode, the Quartus Prime compiler automatically ensures that the associated delay chain settings are set correctly.

However, if the input clock and data at the receiver are not edge- or center-aligned, it may be necessary for you to set the timing constraints in the Quartus Prime TimeQuest Timing Analyzer. The timing constraints specify the timing requirements necessary to ensure reliable data capture.

LVDS Receiver Debug and Troubleshooting

You can obtain useful information about the LVDS interface performance with board-level verification using the FPGA prototype.

Although the focus of the board-level verifcation is to verify the FPGA functionality in your end system, you can take additional steps to examine the margins. Using oscilloscopes, you can examine the margins to verify the predicted size of the data-valid window, and the setup and hold margins at the I/O interface.

You can also use the Altera SignalTap II Logic Analyzer to perform system level verification to correlate the system against your design targets.

Perform RTL Simulation Before Hardware Debug

Before you debug on hardware, Altera recommends that you perform an RTL simulation. Using the RTL simulation, you can check the code functionality before testing in real hardware.

For example, you can use the RTL simulation to verify that when you send a training pattern from a remote transmitter, the bitslipping mechanism in your LVDS receiver works.

Geometry-Based and Physics-Based I/O Rules

You need to consider the I/O placement rules related to LVDS. The Quartus Prime software generates critical warning or error messages if the I/O placements rules are violated.

For more information, refer to the related information.

Related Information
MAX 10 General Purpose I/O User Guide

You can implement mixed transmitter and receiver applications using the MAX 10 LVDS solution. You can use the Altera Soft LVDS IP core to instantiate soft SERDES circuitry. The soft SERDES circuitry works with the clocks and differential I/O pins to create high-speed differential transmitter and receiver circuits.

In a mixed transmitter and receiver implementation, the transmitter and receiver can share some FPGA resources.
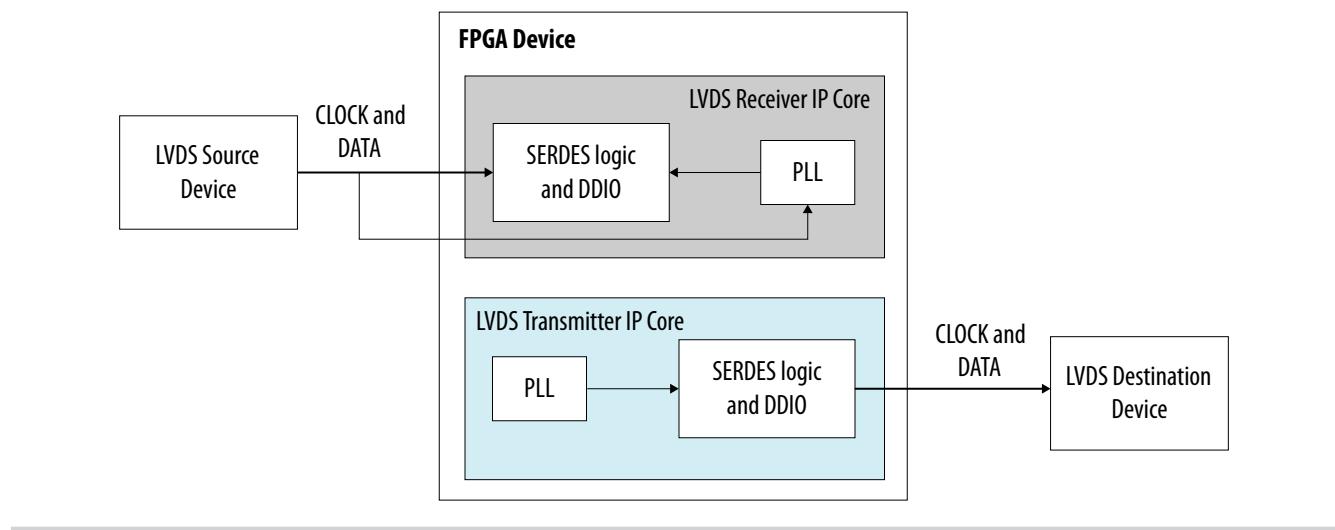
Related Information

- **MAX 10 High-Speed LVDS I/O Overview** on page 1-1
- **MAX 10 LVDS SERDES I/O Standards Support** on page 2-7
  Lists the supported LVDS I/O standards and the support in different MAX 10 device variants.

Transmitter–Receiver Interfacing

You can instantiate the components for the Altera Soft LVDS interfaces by using internal or external PLLs.
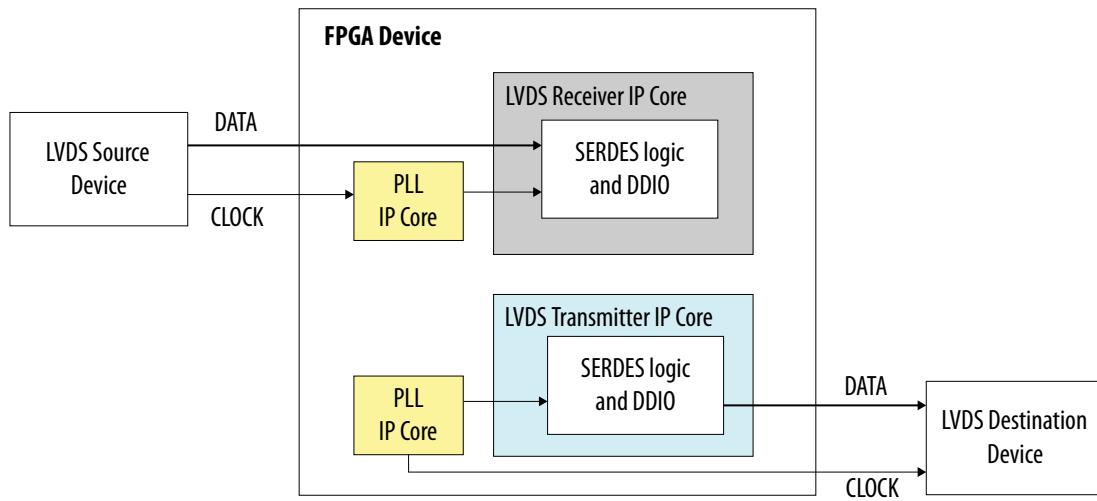
**Figure 5-1: Typical Altera Soft LVDS Interfaces with Internal PLL**

Figure 5-2: Typical Altera Soft LVDS Interfaces with External PLL

Related Information

- **MAX 10 LVDS Transmitter Design** on page 3-1
  Provides more information about specific features and support of the LVDS transmitters.
- **MAX 10 LVDS Receiver Design** on page 4-1
  Provides more information about specific features and support of the LVDS receivers.

LVDS Transmitter and Receiver FPGA Design Implementation

MAX 10 devices use a soft SERDES architecture to support high-speed I/O interfaces. The Quartus Prime software creates the SERDES circuits in the core fabric by using the Altera Soft LVDS IP core. To improve the timing performance and support the SERDES, MAX 10 devices use the I/O registers and LE registers in the core fabric.

LVDS Transmitter and Receiver PLL Sharing Implementation

In applications where an LVDS transmitter and receiver are required, you typically need two PLLs—one for each interface. Using the Altera Soft LVDS IP core, you can reduce PLL usage by sharing one PLL between the transmitter and receiver.

- Turn on the **Use common PLL(s) for receivers and transmitters** option to allow the Quartus Prime compiler to share the same PLL.
- To share a PLL, several PLLs must have the same PLL settings, such as PLL feedback mode, clock frequency, and phase settings. The LVDS transmitters and receivers must use the same input clock frequency and reset input.
- If you are sharing a PLL, you can use more counters to enable different deserialization factor and data rates for the transmitters and receivers. However, because you are using more PLL counters, the PLL input clock frequency and the PLL counter resolution cause limitations in clocking the transmitters and receivers.

Note: The number of PLLs available differs among MAX 10 packages. Altera recommends that you select a MAX 10 device package that provides sufficient number of PLL clockouts for your design.

Initializing the Altera Soft LVDS IP Core

The PLL locks to the reference clock before the Altera Soft LVDS IP core implements the SERDES blocks for data transfer.

During device initialization the PLL starts to lock to the reference clock and becomes operational when it achieves lock during user mode. If the clock reference is not stable, it corrupts the phase shifts of the PLL output clock. This phase shifts corruption can cause failure and corrupt data transfer between the high-speed LVDS domain and the low-speed parallel domain.

To avoid data corruption, follow these steps when initializing the Altera Soft LVDS IP core:

1. Assert the `pll_areset` signal for at least 10 ns.
2. After at least 10 ns, deassert the `pll_areset` signal.
3. Wait until the PLL lock becomes stable.

   After the PLL lock port asserts and is stable, the SERDES blocks are ready for operation.

LVDS Transmitter and Receiver Debug and Troubleshooting

You can obtain useful information about the LVDS interface performance with board-level verification using the FPGA prototype.

Although the focus of the board-level verifcation is to verify the FPGA functionality in your end system, you can take additional steps to examine the margins. Using oscilloscopes, you can examine the margins to verify the predicted size of the data-valid window, and the setup and hold margins at the I/O interface.

You can also use the Altera SignalTap II Logic Analyzer to perform system level verification to correlate the system against your design targets.

Perform RTL Simulation Before Hardware Debug

Before you debug on hardware, Altera recommends that you perform an RTL simulation. Using the RTL simulation, you can check the code functionality before testing in real hardware.

For example, you can use the RTL simulation to verify that when you send a training pattern from a remote transmitter, the bitslipping mechanism in your LVDS receiver works.

Geometry-Based and Physics-Based I/O Rules

You need to consider the I/O placement rules related to LVDS. The Quartus Prime software generates critical warning or error messages if the I/O placements rules are violated.

For more information, refer to the related information.

Related Information
MAX 10 General Purpose I/O User Guide

UG-M10LVDS   ✉ **Subscribe**   💬 **Send Feedback**

To achieve optimal performance from the MAX 10 device, you must consider critical issues such as impedance of traces and connectors, differential routing, and termination techniques.

Guidelines: Improve Signal Quality

To improve signal quality, follow these board design guidelines:

- Base your board designs on controlled differential impedance. Calculate and compare all parameters such as trace width, trace thickness, and the distance between two differential traces.
- Maintain equal distance between traces in differential I/O standard pairs as much as possible. Routing the pair of traces close to each other maximizes the common-mode rejection ratio (CMRR).
- Keep the traces as short as possible to limit signal integrity issues. Longer traces have more inductance and capacitance.
- Place termination resistors as close to receiver input pins as possible.
- Use surface mount components.
- Avoid 90° corners on board traces.
- Use high-performance connectors.
- Design backplane and card traces so that trace impedance matches the impedance of the connector and termination.
- Keep an equal number of vias for both signal traces.
- Create equal trace lengths to avoid skew between signals. Unequal trace lengths result in misplaced crossing points and decrease system margins as the transmitter-channel-to-channel skew (TCCS) value increases.
- Limit vias because they cause discontinuities.
- Keep toggling single-ended I/O signals away from differential signals to avoid possible noise coupling.
- Do not route single-ended I/O clock signals to layers adjacent to differential signals.
- Analyze system-level signals.

Guidelines: Control Channel-to-Channel Skew

For the MAX 10 devices, perform PCB trace compensation to adjust the trace length of each LVDS channel. Adjusting the trace length improves the channel-to-channel skew when interfacing with receivers.

ISO 9001:2008 Registered

ALTERA

now part of Intel

At the package level, you must control the LVDS I/O skew for each I/O bank and each side of the device. If you plan to vertically migrate from one device to another using the same board design, you must control the package migration skew for each migratable LVDS I/O pin.

Receiver Input Skew Margin

Use RSKM, TCCS, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver data path.

Figure 6-1: RSKM Equation

$$RSKM = \frac{TUI - SW - TCCS}{2}$$

Conventions used for the equation:

- RSKM—the timing margin between the clock input of the receiver and the data input sampling window, and the jitter induced from core noise and I/O switching noise.
- Time unit interval (TUI)—time period of the serial data.
- SW—the period of time that the input data must be stable to ensure that the LVDS receiver samples the data successfully. The SW is a device property and varies according to device speed grade.
- TCCS—the timing difference between the fastest and the slowest output edges across channels driven by the same PLL. This measurement includes the t_{CO} variation, clock, and clock skew.

You must calculate the RSKM value, based on the data rate and device, to determine if the LVDS receiver can sample the data:

- A positive RSKM value, after deducting transmitter jitter, indicates that the LVDS receiver can sample the data properly.
- A negative RSKM, after deducting transmitter jitter, indicates that it cannot sample the data properly.

Figure 6-2: Differential High-Speed Timing Diagram and Timing Budget

RSKM Report for LVDS Receiver

For LVDS receivers, the Quartus Prime software provides an RSKM report showing the SW, TUI, and RSKM values.

- To generate the RSKM report, run the `report_RSKM` command in the TimeQuest Timing Analyzer. The RSKM report is available in the TimeQuest Timing Analyzer section of the Quartus Prime compilation report.
- To obtain a more realistic RSKM value, assign the input delay to the LVDS receiver through the constraints menu of the TimeQuest Timing Analyzer. The input delay is determined according to the data arrival time at the LVDS receiver port, with respect to the reference clock.
- If you set the input delay in the settings parameters for the **Set Input Delay** option, set the clock name to the clock that references the source-synchronous clock that feeds the LVDS receiver.
- If you do not set any input delay in the TimeQuest Timing Analyzer, the receiver channel-to-channel skew defaults to zero.
- You can also directly set the input delay in a Synopsys Design Constraint file (**.sdc**) by using the `set_input_delay` command.

Example: RSKM Calculation

This example shows the RSKM calculation for FPGA devices at 1 Gbps data rate with a 200 ps board channel-to-channel skew.

- TCCS = 100 ps
- SW = 300 ps
- TUI = 1000 ps
- Total RCCS = TCCS + Board channel-to-channel skew = 100 ps + 200 ps = 300 ps
- RSKM = (TUI – SW – RCCS) / 2 = (1000 ps – 300 ps – 300 ps) / 2 = 200 ps

If the RSKM is greater than 0 ps after deducting transmitter jitter, the receiver will work correctly.

Guidelines: Determine Board Design Constraints

After you have closed timing for your FPGA design, examine your board design to determine the different factors that can impact signal integrity. These factors affect overall timing at the receiving device in the LVDS interface.

The time margin for the LVDS receiver (indicated by the RSKM value) is the timing budget allocation for board level effects such as:

- Skew—these factors cause board-level skew:
  - Board trace lengths
  - Connectors usage
  - Parasitic circuits variations
- Jitter—jitter effects are derived from factors such as crosstalk.
- Noise—on board resources with imperfect power supplies and reference planes may also cause noise.

To ensure successful operation of the Altera Soft LVDS IP core receiver, do not exceed the timing budget.

Related Information

Board Design Guidelines Solution Center
Provides resources related to board design for Altera devices.

Guidelines: Perform Board Level Simulations

After you determined the system requirements and finalized the board design constraints, use an electronic design automation (EDA) simulation tool to perform board-level simulations. Use the IBIS or HSPICE models of the FPGA and the target LVDS device for the simulation.

The board-level simulation ensures optimum board setup where you can determine if the data window conforms to the input specification (electrical and timing) of the LVDS receiver.

You can use the programmable pre-emphasis feature on the true LVDS output buffers, for example, to compensate for the frequency-dependent attenuation of the transmission line. With this feature, you can maximize the data eye opening at the far end receiver especially on long transmission lines.

Related Information

- **Altera IBIS Models**
  Provides IBIS models of Altera devices for download.
- **Altera HSPICE Models**
  Provides SPICE models of Altera devices for download.
- **IBIS Model Generation**
  Provides video that demonstrates how to generate IBIS file using the Quartus Prime software.

You can set various parameter settings for the Altera Soft LVDS IP core to customize its behaviors, ports, and signals.

The Quartus Prime software generates your customized Altera Soft LVDS IP core according to the parameter options that you set in the parameter editor.

Related Information

- **MAX 10 High-Speed LVDS I/O Overview** on page 1-1
- **MAX 10 LVDS SERDES I/O Standards Support** on page 2-7
  Lists the supported LVDS I/O standards and the support in different MAX 10 device variants.

Altera Soft LVDS Parameter Settings

There are four groups of options: **General** , **PLL Settings** , **Receiver Settings** , and **Transmitter Settings**

Table 7-1: Altera Soft LVDS Parameters - General

Parameter	Condition	Allowed Values	Description
Power Supply Mode	—	• Dual Supply • Single Supply	Specifies whether the target device is a single or dual supply device.
Functional mode	—	• RX • TX	Specifies the functional mode for the Altera Soft LVDS IP core: • **RX**—specifies the IP is an LVDS receiver. • **TX**—specifies the IP is an LVDS transmitter.
Number of channels	—	1–18	Specifies the number of LVDS channels.

Parameter	Condition	Allowed Values	Description
SERDES factor	—	1, 2, 4, 5, 6, 7, 8, 9, 10	Specifies the number of bits per channel.

Table 7-2: Altera Soft LVDS Parameters - PLL Settings

Parameter	Condition	Allowed Values	Description
Use external PLL	Not applicable for x1 and x2 modes.	• On • Off	Specifies whether the Altera Soft LVDS IP core generates a PLL or connects to a user-specified PLL.
Data rate	—	Refer to the device datasheet.	Specifies the data rate going out of the PLL. The multiplication value for the PLL is OUTPUT_DATA_RATE divided by INCLOCK_ PERIOD.
Inclock frequency	—	Depends on **Data rate**.	Specifies the input clock frequency to the PLL in MHz.
Enable rx_locked port	• General, Functional mode = RX • Use external PLL = Off	• On • Off	If turned on, enables the `rx_locked` port.
Enable tx_locked port	• General, Functional mode = TX • Use external PLL = Off	• On • Off	If turned on, enables the `tx_locked` port.
Enable pll_areset port	Always on	• On • Off	If turned on, enables the `pll_areset` port in internal PLL mode. In external PLL mode, the `pll_areset` port is always on.
Enable tx_data_ reset port	• General, Functional mode = TX • Use external PLL = On	• On • Off	If turned on, enables the `tx_data_reset` port.

Parameter	Condition	Allowed Values	Description
Enable rx_data_ reset port	• General, Functional mode = RX • Use external PLL = On	• On • Off	If turned on, enables the rx_data_reset port.
Use common PLL(s) for receivers and transmitters	Use external PLL = Off	• On • Off	• **On**—specifies that the compiler uses the same PLL for the LVDS receiver and transmitter. • **Off**—specifies that the compiler uses different PLLs for LVDS receivers and transmitters. You can use common PLLs if you use the same input clock source, deserialization factor, pll_areset source, and data rates.
Enable self-reset on loss lock in PLL	Use external PLL = Off	• On • Off	If turned on, the PLL is reset when it loses lock.
Desired transmitter inclock phase shift	• General, Functional mode = TX • Use external PLL = Off	Depends on **Data rate**.	Specifies the phase shift parameter used by the PLL for the transmitter.
Desired receiver inclock phase shift	• General, Functional mode = RX • Use external PLL = Off	Depends on **Data rate**.	Specifies the phase shift parameter used by the PLL for the receiver.

Table 7-3: Altera Soft LVDS Parameters - Receiver Settings

Parameter	Condition	Allowed Values	Description
Enable bitslip mode	General, Functional mode = RX	• On • Off	If turned on, enables the rx_data_align port.

Parameter	Condition	Allowed Values	Description
Enable independent bitslip controls for each channel	General, Functional mode = RX	• On • Off	If turned on, enables the `rx_channel_data_align` port. The `rx_channel_data_align` is an edge-sensitive bit slip control signal: • Each rising edge on this signal causes the data realignment circuitry to shift the word boundary by one bit. • The minimum pulse width requirement is one parallel clock cycle.
Enable rx_data_align_reset port	• General, Functional mode = RX • Enable bitslip mode = On • Enable independent bitslip controls for each channel = Off	• On • Off	If turned on, enables the `rx_data_align_reset` port.
Add extra register for rx_data_align port	• General, Functional mode = RX • Enable bitslip mode = On	• On • Off	If turned on, registers the `rx_data_align` port. If you turn this option off, you must pre-register the `rx_data_align[]` port in the logic that feeds the receiver.
Bitslip rollover value	• General, Functional mode = RX • Enable bitslip mode = On	1–11	Specifies the number of pulses before the circuitry restores the serial data latency to 0.
Use RAM buffer	—	• On • Off	If turned on, the Altera Soft LVDS IP core implements the output synchronization buffer in the embedded memory blocks. This implementation option uses more logic than **Use a multiplexer and synchronization register** option but results in the correct word alignment.
Use a multiplexer and synchronization register	—	• On • Off	If turned on, the Altera Soft LVDS IP core implements a multiplexer instead of a buffer for output synchronization.

Parameter	Condition	Allowed Values	Description
Use logic element based RAM	—	• On • Off	If turned on, the Altera Soft LVDS IP core implements the output synchronization buffer in the logic elements. This implementation option uses more logic than **Use a multiplexer and synchronization register** option but results in the correct word alignment.
Register outputs	General, Functional mode = RX	• On • Off	If turned on, registers the `rx_out[]` port. If you turn this option off, you must pre-register the `rx_out[]` port in the logic that feeds the receiver.

Table 7-4: Altera Soft LVDS Parameters - Transmitter Settings

Parameter	Condition	Allowed Values	Description
Enable 'tx_outclock' output port	• General, Functional mode = TX • PLL Settings, Use external PLL = Off	• On • Off	If turned on, enables the `tx_outclock` port. Every `tx_outclock` signal goes through the shift register logic, except in the following parameter configurations: • When the `outclock_divide_by` signal = 1 • When the `outclock_divide_by` signal is equal to the `deserialization_factor` signal (for odd factors only), and the `outclock_duty_cycle` signal is 50
Tx_outclock division factor	• General, Functional mode = TX • PLL Settings, Use external PLL = Off • Enable 'tx_outclock' output port = On	Depends on **SERDES factor**.	Specifies that the frequency of the `tx_outclock` signal is equal to the the transmitter output data rate divided by the selected division factor.

Parameter	Condition	Allowed Values	Description
Outclock duty cycle	• General, Functional mode = TX • PLL Settings, Use external PLL = Off • Enable 'tx_ outclock' output port = On	Depends on **SERDES factor** and **Tx_outclock division factor**.	Specifies the external clock timing constraints.
Desired transmitter outclock phase shift	• General, Functional mode = TX • PLL Settings, Use external PLL = Off • Enable 'tx_ outclock' output port = On	Depends on **Data rate**.	Specifies the phase shift of the output clock relative to the input clock.
Register 'tx_in' input port	General, Functional mode = TX	• On • Off	If turned on, registers the `tx_in[]` port. If you turn this option off, you must pre-register the `tx_in[]` port in the logic that feeds the transmitter.
Clock resource	• General, Functional mode = TX • Register 'tx_in' input port = On	• tx_inclock • tx_ coreclock	Specifies which clock resource registers the `tx_in` input port.
Enable 'tx_ coreclock' output port	General, Functional mode = TX	• On • Off	If turned on, enables the `tx_coreclock` output port.
Clock source for 'tx_coreclock'	• General, Functional mode = TX • Enable 'tx_ coreclock' output port = On	• Auto selection • Global clock • Regional clock • Dual-Regional clock	Specifies which clock resource drives the `tx_coreclock` output port.

Related Information

- **Altera Soft LVDS IP Core in Transmitter Mode** on page 3-6
- **Altera Soft LVDS IP Core in Receiver Mode** on page 4-6

Altera Soft LVDS Interface Signals

Depending on parameter settings you specify, different signals are available for the Altera Soft LVDS IP core.

Table 7-5: Transmitter Interface Signals

Signal Name	Direction	Width (Bit)	Description
pll_areset	Input	1	Asynchronously resets all counters to the initial values.
tx_data_reset	Input	<n>	Asynchronous reset for the shift registers, capture registers, and synchronization registers for all channels. • This signal is used if **Use external PLL** parameter setting is turned on. • This signal does not affect the data realignment block or the PLL.
tx_in[]	Input	<m>	This signal is parallel data that Altera Soft LVDS IP core transmits serially. Input data is synchronous to the tx_coreclock signal. The data bus width per channel is the same as the serialization factor (SF).
tx_inclock	Input	1	Reference clock input for the transmitter PLL. The parameter editor automatically selects the appropriate PLL multiplication factor based on the data and reference clock frequency.
tx_coreclock	Output	1	Output clock that feeds non-peripheral logic. FPGA fabric–transmitter interface clock— the parallel transmitter data generated in the FPGA fabric is clocked with this clock.

Signal Name	Direction	Width (Bit)	Description
tx_locked	Output	1	Provides the LVDS PLL status: • Remains high when the PLL is locked to the input reference clock. • Remains low when the PLL fails to lock.
tx_out[]	Output	*<n>*	Serialized LVDS data output signal of *<n>* channels. tx_out[(*<n>*-1)..0] drives parallel data from tx_in[(*<J>* × *<n>*)-1 ..0] where *<J>* is the serialization factor and *<n>* is the number of channels. tx_out[0] drives data from tx_in[(*<J>*-1)..0]. tx_out[1] drives data from the next *<J>* number of bits on tx_in.
tx_outclock	Output	1	External reference clock. The frequency of this clock is programmable to be the same as the data rate.

Table 7-6: Receiver Interface Signals

signal Name	Direction	Width (Bit)	Description
rx_data_reset	Input	*<n>*	Asynchronous reset for all channels, excluding the PLL. • This signal is available if **Use external PLL** parameter setting is turned on. • You must externally synchronize this signal with the fast clock.
rx_in[]	Input	*<n>*	LVDS serial data input signal of *<n>* channels. rx_in[(*<n>*-1)..0] is deserialized and driven on rx_out[(*<J>* × *<n>*)-1 ..0] where *<J>* is the deserialization factor and *<n>* is the number of channels. rx_in[0] drives data to rx_out[(*<J>*-1)..0]. rx_in[1] drives data to the next *<J>* number of bits on rx_out.

signal Name	Direction	Width (Bit)	Description
rx_inclock	Input	1	LVDS reference input clock. The parameter editor automatically selects the appropriate PLL multiplication factor based on the data rate and reference clock frequency selection.
rx_coreclk	Input	<n>	LVDS reference input clock. • Replaces the non-peripheral clock from the PLL. • One clock for each channel.
rx_locked	Output	1	Provides the LVDS PLL status: • Stays high when the PLL is locked to the input reference clock. • Stays low when the PLL fails to lock.
rx_out	Output	<m>	Receiver parallel data output. The data bus width per channel is the same as the deserialization factor (DF).
rx_outclock	Output	1	Parallel output clock from the receiver PLL. • This signal is not available if you turn on the **Use external PLL** parameter setting. • The FPGA fabric–receiver interface clock must be driven by the PLL instantiated through the ALTPLL parameter editor.
rx_data_align	Input	1	Controls the byte alignment circuitry. You can register this signal using the rx_outclock signal.
rx_data_align_reset	Input	1	Resets the byte alignment circuitry. Use the rx_data_align_reset input signal if: • You need to reset the PLL during device operation. • You need to re-establish the word alignment.
rx_channel_data_align	Input	<n>	Controls byte alignment circuitry.

signal Name	Direction	Width (Bit)	Description
rx_cda_reset	Input	*<n>*	Asynchronous reset to the data realignment circuitry. This signal resets the data realignment block. The minimum pulse width requirement for this reset is one parallel clock cycle.

2016.05.02

UG-M10LVDS ✉ **Subscribe** 💬 **Send Feedback**

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 High-Speed LVDS I/O User Guide**
15.0	**MAX 10 High-Speed LVDS I/O User Guide**
14.1	**MAX 10 High-Speed LVDS I/O User Guide**

ISO 9001:2008 Registered

ALTERA

now part of Intel

Date	Version	Changes
May 2016	2016.05.02	• Added true RSDS and emulated RSDS (three resistors) transmitter support for single supply MAX 10 devices. • Updated the transmitter and receiver channels placement topics to describe about minimizing skew when you group LVDS channels for an application. • Updated the description of the `rx_data_reset` interface signal to specify that you must externally synchronize it with the fast clock. • Updated the **General** tab of the Altera Soft LVDS parameter settings: • Added the **Power Supply Mode** option. • Updated the allowed values of the SERDES factor parameter.
November 2015	2015.11.02	• Updated the high-speed LVDS circuitry figure to correct the flow from `C1` in ALTPLL to `inclock` of ALTERA_SOFT_LVDS. Previously, the figure shows a bidirectional flow. • Updated the steps for determining the external PLL clock parameters for the receiver to clarify the clock names listed by the Quartus Prime compilation report. • Updated the topic about Altera Soft LVDS parameter settings: • Added links to topics about PLL and high-speed I/O performance in the device datasheet. • Corrected the conditions required to use the **Enable tx_data_reset port** and **Enable rx_data_reset port** parameters. You must turn on the **Use external PLL** option first. • Updated the allowed values for the **Tx_outclock division factor** and **Outclock duty cycle** parameters. • Updated the condition for the **Desired transmitter outclock phase shift** parameter. • Removed the topics about generating IP cores and the files generated by the IP core, and added a link to *Introduction to Altera IP Cores*.

**ISO
9001:2008
Registered**

ALTERA
now part of Intel

Date	Version	Changes
		• Removed the statement about getting TCCS value from the Quartus Prime compilation report. You can get TCCS value from the device datasheet. • Added guidelines topic about enabling LVDS pre-emphasis for MAX 10 devices in the E144 package. • Updated the guidelines to control channel-to-channel skew to remove statements about getting the trace delay amount from the Fitter Report panel. • Added link to video that demonstrates how to generate IBIS file using the Quartus Prime software. • Changed instances of *Quartus II* to *Quartus Prime*.

May 2015	2015.05.04	• Removed the F672 package from the 10M25 device.
		• Updated the number of bottom true receiver channels for package M153 of the 10M02 device from 49 to 13.
		• Added BLVDS output support in single-supply MAX 10 devices. Previously, BLVDS support for single-supply devices was input only.
		• Updated the RSKM definition in the topic about receiver input skew margin to include jitter induced from core noise and I/O switching noise.
		• Updated topics related to using the Altera Soft LVDS IP core (transmitter or receiver) in external PLL mode:
		  • Added `rx_readclock`, `rx_syncclock`, and `tx_synclock` ports.
		  • Removed `pll_areset` port.
		  • Added examples for odd and even serialization factors.
		  • Added procedures to obtain the external PLL clock parameters.
		  • Removed similar guidelines in the chapter for the transmitter and receiver design. The updated guidelines for the receiver only and transmitter only designs can apply for designs that use both transmitters and receivers.
		• Updated parameter settings of the Altera Soft LVDS IP core:
		  • Removed allowed values "6" and "9" from the **SERDES factor** parameter.
		  • Added allowed value "Off" to the **Enable pll_areset port** parameter.
		  • Updated the parameter label **Register_rx_bitslip_ctrl port** to **Add extra register for rx_data_align port** and specified that you must pre-register the port if you turn it on.

Date	Version	Changes
December 2014	2014.12.15	- Updated table listing LVDS channels to include LVDS channel counts for each device package. - Added information in the topics about channels placement that MAX 10 devices support x18 bundling mode. - Updated the examples in topics about channels PLL placement to provide more details. - Added link to the **MAX 10 Clocking and PLL User Guide** that provides more information about the PLL and the PLL output counters used to clock the soft SERDES.
September 2014	2014.09.22	Initial release.

MAX 10 External Memory Interface User Guide

Subscribe

Send Feedback

Contents

UG-M10EMI   ✉ **Subscribe**   💬 **Send Feedback**

The MAX® 10 devices are capable of interfacing with a broad range of external memory standards. With this capability, you can utilize MAX 10 devices in a wide range of applications such as image processing, storage, communications, and general embedded systems.

The external memory interface solution in MAX 10 devices consist of:

- The I/O elements that support external memory interfaces.
- The UniPHY IP core that allows you to configure the memory interfaces to support different external memory interface standards.

Note: Altera recommends that you construct all DDR2, DDR3, and LPDDR2 SDRAM external memory interfaces using the UniPHY IP core.[1]

Related Information

- **MAX 10 External Memory Interface Architecture and Features** on page 2-1
- **MAX 10 External Memory Interface Design Considerations** on page 3-1
- **MAX 10 External Memory Interface Implementation Guides** on page 4-1
- **UniPHY IP Core References for MAX 10** on page 5-1
- **Documentation: External Memory Interfaces**
  Provides more information about external memory system performance specification, board design guidelines, timing analysis, simulation, and debugging.
- **External Memory Interface Handbook Volume 1: Altera Memory Solution Overview and Design Flow**
  Provides more information about using Altera devices for external memory interfaces including Altera memory solution and design flow.
- **External Memory Interface Handbook Volume 2: Design Guidelines**
  Provides more information about using Altera devices for external memory interfaces including memory selection, board design, implementing memory IP cores, timing, optimization, and debugging.
- **Functional Description—MAX 10 EMIF**
  Provides more information about implementing memory IP cores for MAX 10 devices.
- **MAX 10 DDR3 Reference Design**
  Provides DDR3 UniPHY IP core reference design for MAX 10 devices.

---

[1] Licensing terms and costs for UniPHY IP core apply.

ISO 9001:2008 Registered

ALTERA
now part of Intel

- **MAX 10 External Memory Interface User Guide Archives** on page 6-1
  Provides a list of user guides for previous versions of the UniPHY IP core.

MAX 10 External Memory Interface Support and Performance

The MAX 10 devices contain circuitry that supports several external memory interface standards.

Table 1-1: Memory Standards Supported by MAX 10 Devices

External Memory Interface Standard	Voltage (V)	Max Frequency (MHz)
DDR3	1.5	303
DDR3L	1.35	303
DDR2	1.8	200
LPDDR2[2]	1.2	200[3]

Related Information

- **External Memory Interface Spec Estimator**
  Provides a parametric tool that allows you to find and compare the performance of the supported external memory interfaces in Altera devices.
- **Planning Pin and FPGA Resources chapter, External Memory Interface Handbook**
  Provides the maximum number of interfaces supported by MAX 10 devices for each memory standards, pin counts for various external memory interface implementation examples, and information about the clock, address/command, data, data strobe, DM, and optional ECC signals.
- **MAX 10 Device Datasheet**

---

[2] MAX 10 devices support only single-die LPDDR2.
[3] To achieve the specified performance, constrain the memory device I/O and core power supply variation to within ±3%. By default, the frequency is 167 MHz.
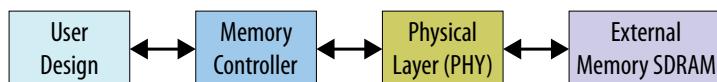
The external memory interface architecture of MAX 10 devices is a combination of soft and hard IPs.

Figure 2-1: High Level Overview of MAX 10 External Memory Interface System

This figure shows a high level overview of the main building blocks of the external memory interface system in MAX 10 devices.

| User Design | ⟷ | Memory Controller | ⟷ | Physical Layer (PHY) | ⟷ | External Memory SDRAM |

- The full rate data capture and write registers use the DDIO registers inside the I/O elements.
- PHY logic is implemented as soft logic in the core fabric.
- The memory controller is the intermediary between the user logic and the rest of the external memory interface system. The Altera® memory controller IP is a soft memory controller that operates at half rate. You can also use your own soft memory controller or a soft memory controller IP from Altera's third-party partners.
- The physical layer (PHY) serves as the bridge between the memory controller and the external memory DRAM device.

Related Information

- **MAX 10 External Memory Interface Overview** on page 1-1
- **Documentation: External Memory Interfaces**
  Provides more information about external memory system performance specification, board design guidelines, timing analysis, simulation, and debugging.
- **Intellectual Properties: Memories & Controllers**
  Provides a list of memory controller IP solutions from Altera and partners.

MAX 10 I/O Banks for External Memory Interface

In MAX 10 devices, external memory interfaces are supported only on the I/O banks on the right side of the device. You must place all external memory I/O pins on the I/O banks on the right side of the device.

ISO 9001:2008 Registered

ALTERA

now part of Intel

Figure 2-2: I/O Banks for External Memory Interfaces

This figure represents the top view of the silicon die that corresponds to a reverse view of the device package.

External memory interfaces support is available only for 10M16, 10M25, 10M40, and 10M50 devices.

MAX 10 DQ/DQS Groups

Different MAX 10 devices and packages support different numbers of DQ/DQS groups for external memory interfaces.

Table 2-1: Supported DQ/DQS Group Sizes in MAX 10 Devices and Packages

This table lists the number of DQ/DQS groups supported on different MAX 10 devices and packages. Only the I/O banks on the right side of the devices support external memory interfaces.

Device	Package	I/O Bank (Right Side)	Number of DQ Groups x8
10M16	F256, U324, and F484	B5	1
		B6	1

Device	Package	I/O Bank (Right Side)	Number of DQ Groups x8
10M25	F256	B5	1
		B6	1
	F484	B5	1
		B6	2
10M40	F256	B5	1
		B6	1
	F484	B5	1
		B6	2
	F672	B5	2
		B6	2
10M50	F256	B5	1
		B6	1
	F484	B5	1
		B6	2
	F672	B5	2
		B6	2

Related Information

Planning Pin and FPGA Resources chapter, External Memory Interface Handbook
Provides the maximum number of interfaces supported by MAX 10 devices for each memory standards, pin counts for various external memory interface implementation examples, and information about the clock, address/command, data, data strobe, DM, and optional ECC signals.

MAX 10 External Memory Interfaces Maximum Width

Different MAX 10 device packages support different maximum width for external memory interfaces.

Table 2-2: Supported Maximum External Memory Interface Width in MAX 10 Device Packages

Product Line	Package			
	F256	**U324**	**F484**	**F672**
10M16	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 LPDDR2 without ECC	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x16 LPDDR2 without ECC	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x16 LPDDR2 without ECC	—
10M25	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 LPDDR2 without ECC	—	• x8 DDR2, DDR3/3L and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x24 DDR2, and DDR3/3L with ECC • x16 LPDDR2 without ECC	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x24 DDR2, and DDR3/3L with ECC • x16 LPDDR2 without ECC
10M40	• x8 DDR2, DDR3/3L and LPDDR2 without ECC • x16 LPDDR2 without ECC	—	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x24 DDR2, and DDR3/3L with ECC • x16 LPDDR2 without ECC	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x24 DDR2, and DDR3/3L with ECC • x16 LPDDR2 without ECC

Product Line	Package			
	F256	**U324**	**F484**	**F672**
10M50	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 LPDDR2 without ECC	—	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x24 DDR2, and DDR3/3L with ECC • x16 LPDDR2 without ECC	• x8 DDR2, DDR3/3L, and LPDDR2 without ECC • x16 DDR2, and DDR3/3L with or without ECC • x24 DDR2, and DDR3/3L with ECC • x16 LPDDR2 without ECC

MAX 10 Memory Controller

MAX 10 devices use the HPC II external memory controller.

Table 2-3: Features of the MAX 10 Memory Controller

Feature	Description
Half-Rate Operation	The controller and user logic can run at half the memory clock rate.
Controller Latency	The controller has a low best-case time between a read request or a write request on the local interface, and the memory command being sent to the AFI interface.
Data Reordering	The memory controller will reorder read and write requests as necessary to achieve the most efficient throughput of data.
Starvation Control	The controller implements a starvation counter to limit the length of time that a command can go unserved. This counter ensures that lower-priority requests are not overlooked indefinitely due to data reordering. You can set a starvation limit, to ensure that a waiting command is served immediately, when the starvation counter reaches the specified limit.
Priority Bypass	The memory controller accepts user requests to bypass the priority established by data reordering. When the controller detects a high-priority request, it allows that request to bypass the current queue. The high-priority request is then processed immediate, reducing latency.

Feature	Description
Standard Interface	The memory controller uses Avalon-ST as its native interface, allowing the flexibility to extend to Avalon-MM, AXI, or a proprietary protocol with an adapter.
Avalon-MM Data Slave Local Interface	The controller supports the Altera Avalon memory-mapped protocol.
Bank Management	The memory controller will intelligently keep a page open based on incoming traffic, improving efficiency, especially for random traffic.
Streaming Reads and Writes	The memory controller has the ability to issue reads or writes continuously to sequential addresses each clock cycle, if the bank is open. This feature allows for the passage of large amounts of data, with high efficiency.
Bank Interleaving	The memory controller has the ability to issue reads or writes continuously to random addresses. The bank addresses must be correctly cycled by user logic.
Predictive Bank Management	The memory controller has the ability to issue bank management commands early, so that the correct row is already open when a read or write request occurs. This feature allows for increased efficiency.
Quasi-1T Address/Command Half-Rate	One controller clock cycle equals two memory clock cycles in a half-rate interface. To maximize command bandwidth, the memory controller provides the option to allow two memory commands on every controller clock cycle. The controller is constrained to issue a row command on the first clock phase and a column command on the second clock phase, or vice versa. Row commands include activate and precharge commands; column commands include read and write commands.
Built-In Burst Adaptor	The memory controller has the ability to accept bursts of arbitrary size on the local interface, and map these to efficient memory commands.
Self-Refresh Controls and User Auto-Refresh Controls	The memory controller has the ability to issue self-refresh commands and allow user auto-refresh through a sideband interface.
Enable Auto Power-Down	The memory controller has the ability to power-down if no commands are received.

MAX 10 External Memory Read Datapath

In MAX 10 devices, instead of using DQS strobes, the memory interface solution uses internal read capture clock to capture data directly in the double data rate I/O (DDIO) registers in the I/O elements.

- The PLL supplies memory clock to the DRAM device and generates read capture clock that is frequency-locked to the incoming data stream. The read capture clock and the incoming read data stream have an arbitrary phase relationship.
- For maximum timing margin, calibration sequence is used to position the read capture clock within the optimum sampling position in the read data eye.
- Data is captured directly in the DDIO registers implemented in the I/O periphery.

DDR Input Registers

The DDR input capture registers in MAX 10 devices are implemented in the I/O periphery.

Figure 2-3: External Memory Interface Read Datapath

Send Feedback
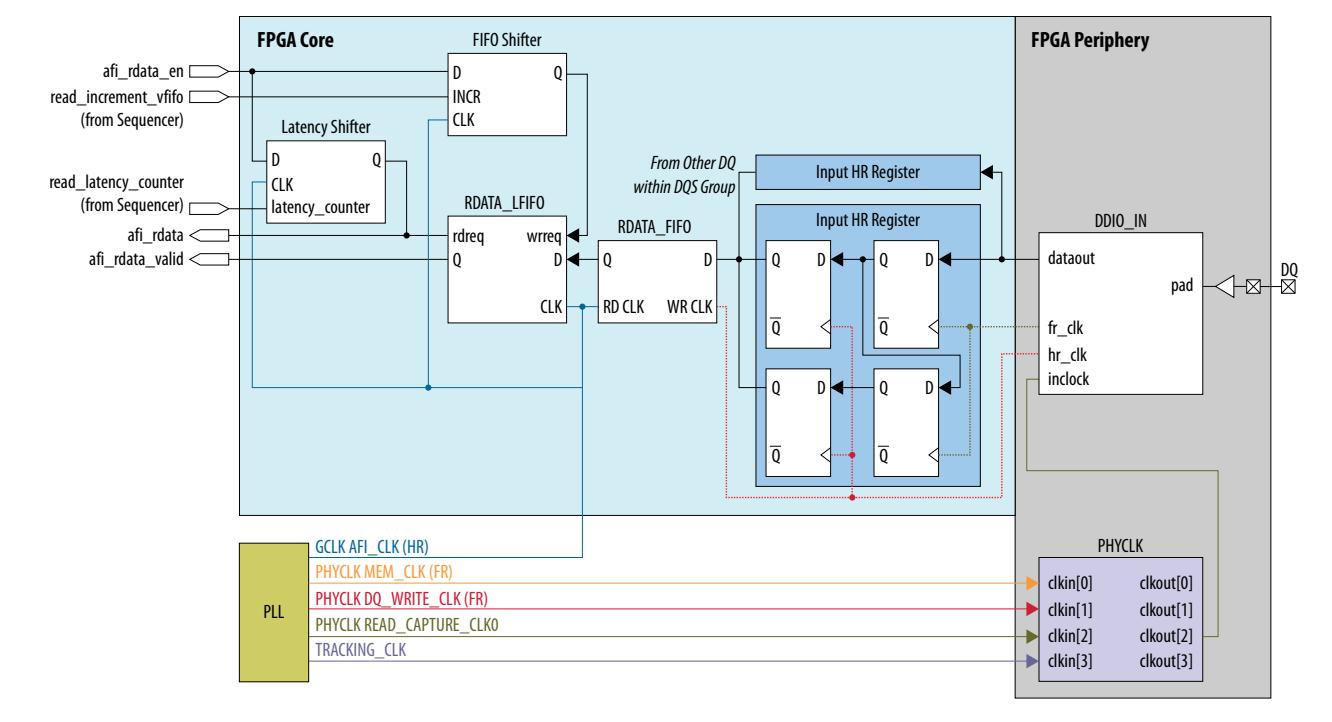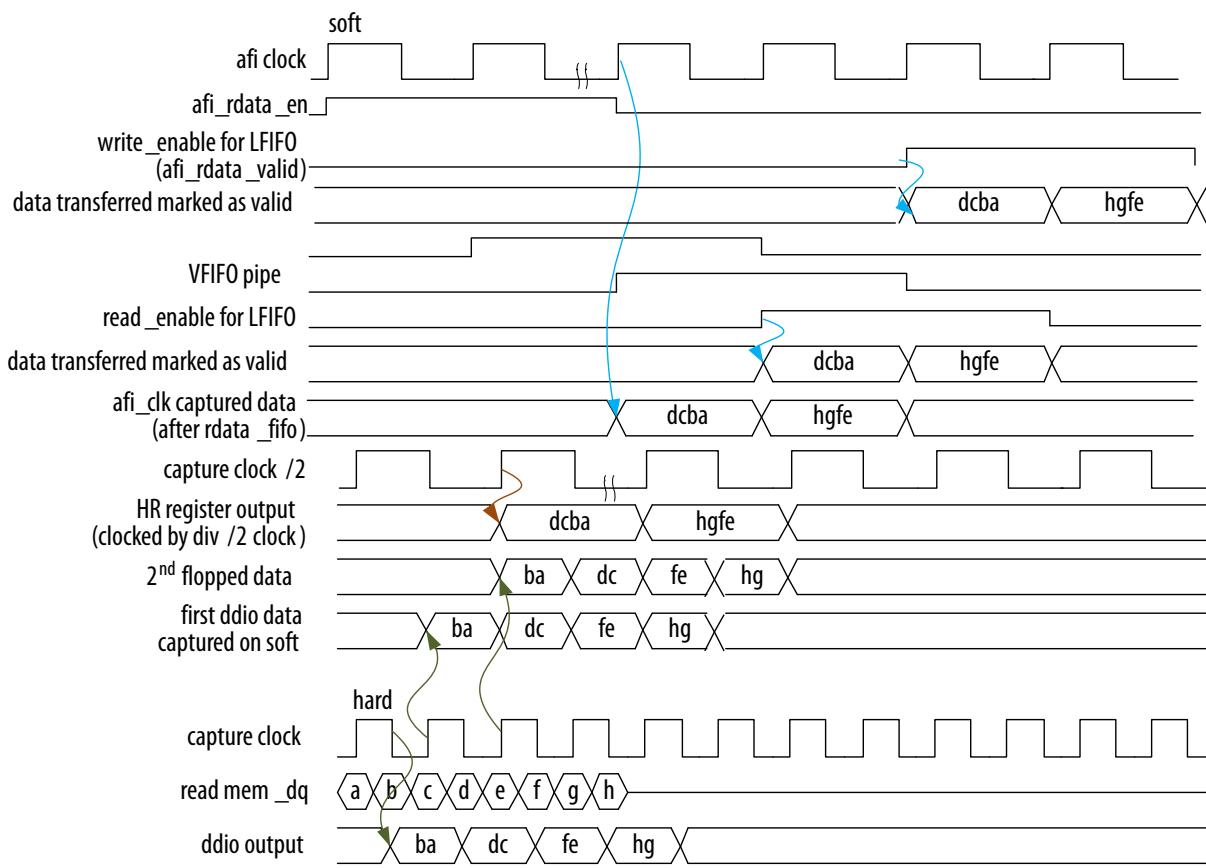
Figure 2-4: External Memory Interface Read Datapath Timing

In MAX 10 external memory interfaces, post-amble is not a concern because the read data strobe signal, DQS, is not used during read operation.

MAX 10 External Memory Write Datapath

For all DDR applications supported by MAX 10 devices, the DQS strobe is sent to the external DRAM as center-aligned to the write DQ data.

The clock that clocks DDIO registers of the DQ output is phase-shifted $-90°$ from the clock that drives the DDIO registers of the DQS strobe. This create a DQS strobe that is center-aligned to the DQ data.

The external memory write datapath is not calibrated.

DDR Output Registers

A dedicated DDIO write block is implemented in the DDR output and output enable paths.

Figure 2-5: External Memory Interface Write Datapath
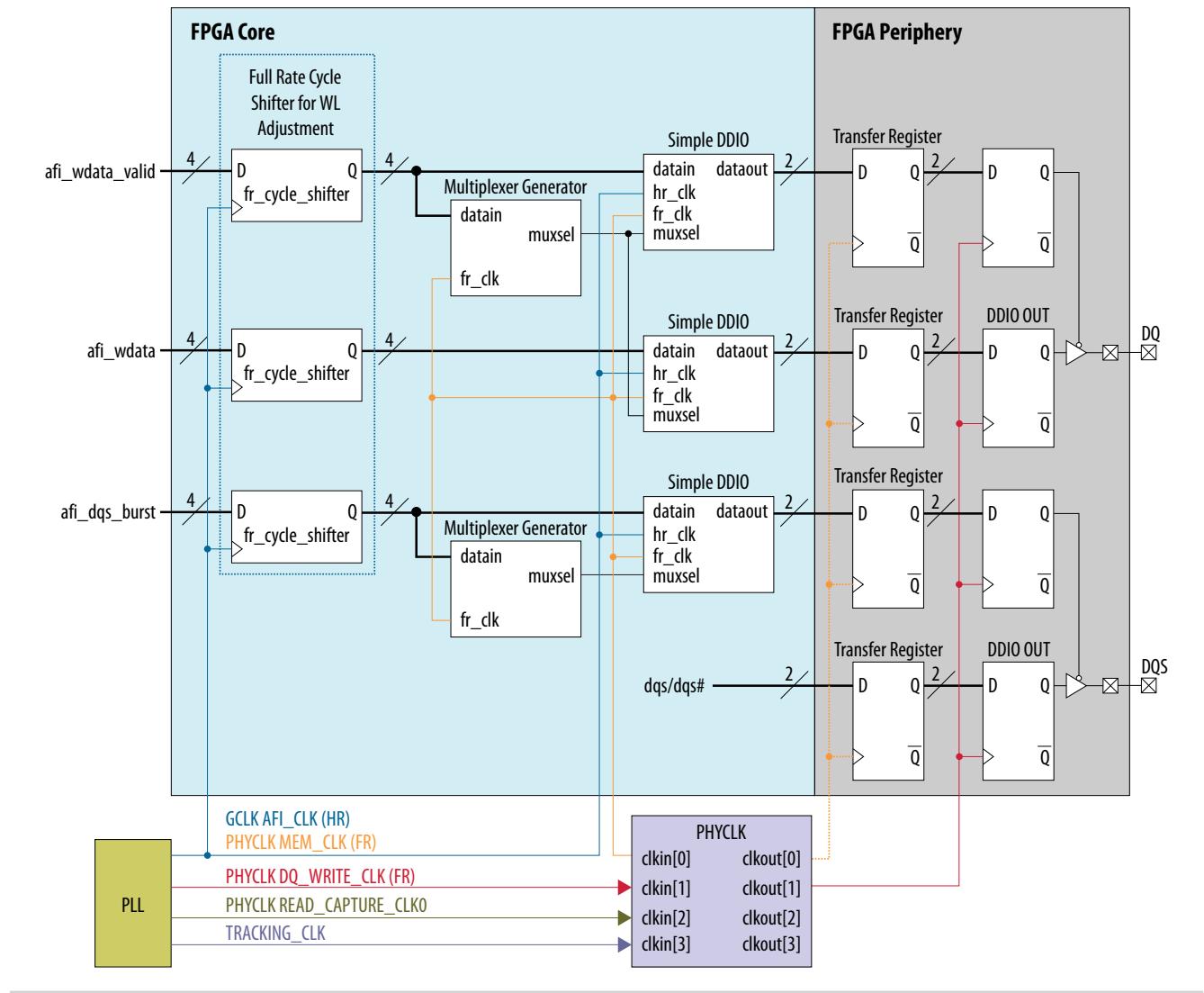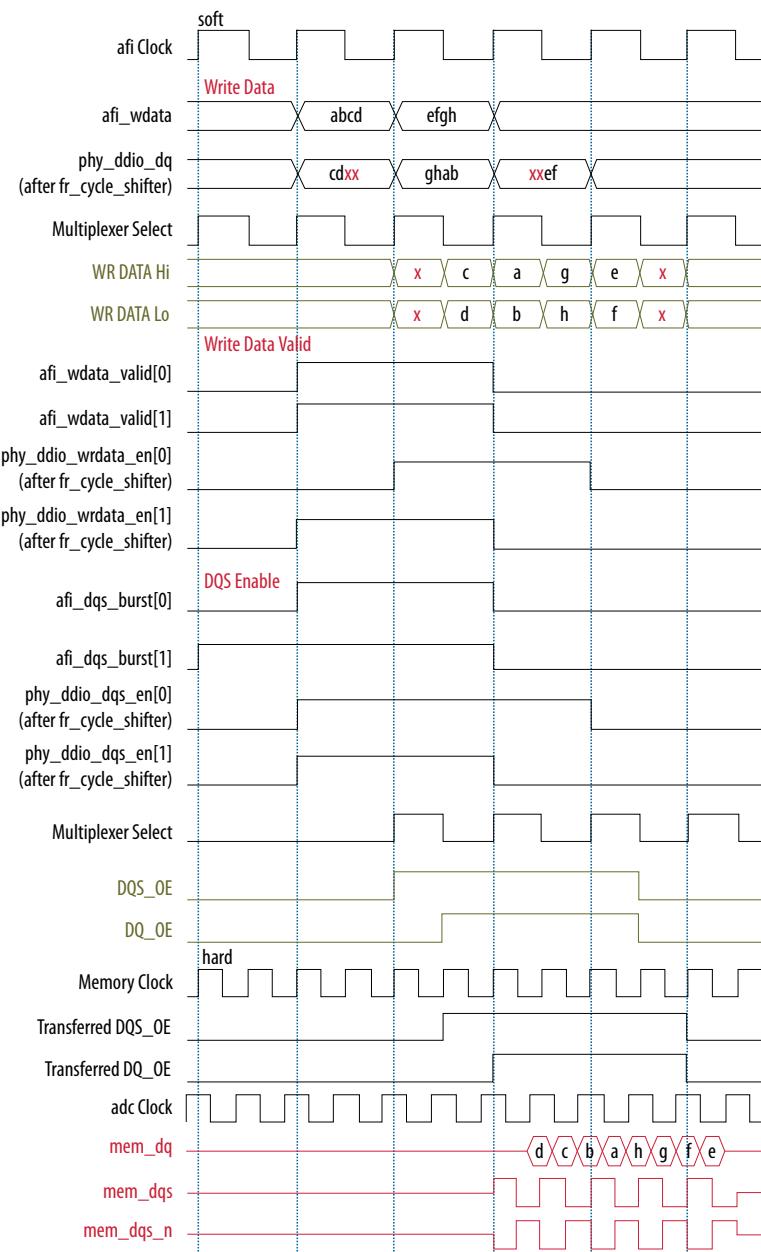
Figure 2-6: External Memory Interface Write Datapath Timing

MAX 10 Address/Command Path

Altera's soft memory controller IP and PHY IP operate at half rate and issue address/command instructions at half-rate.

- You must send the address/command instructions to the external DRAM center-aligned with respect to the external memory clock (CK/CK#).
- For LPDDR2 applications, the address/command path is double data rate (DDR). Dedicated DDIO output registers in the I/O periphery clocks out the address/command instructions to the external DRAM.
- For DDR2/3 applications, the address/command path is single data rate (SDR). Instead of dedicated DDIO output registers, simple output I/O registers in the I/O periphery clocks out the address/command instructions to the external DRAM device.
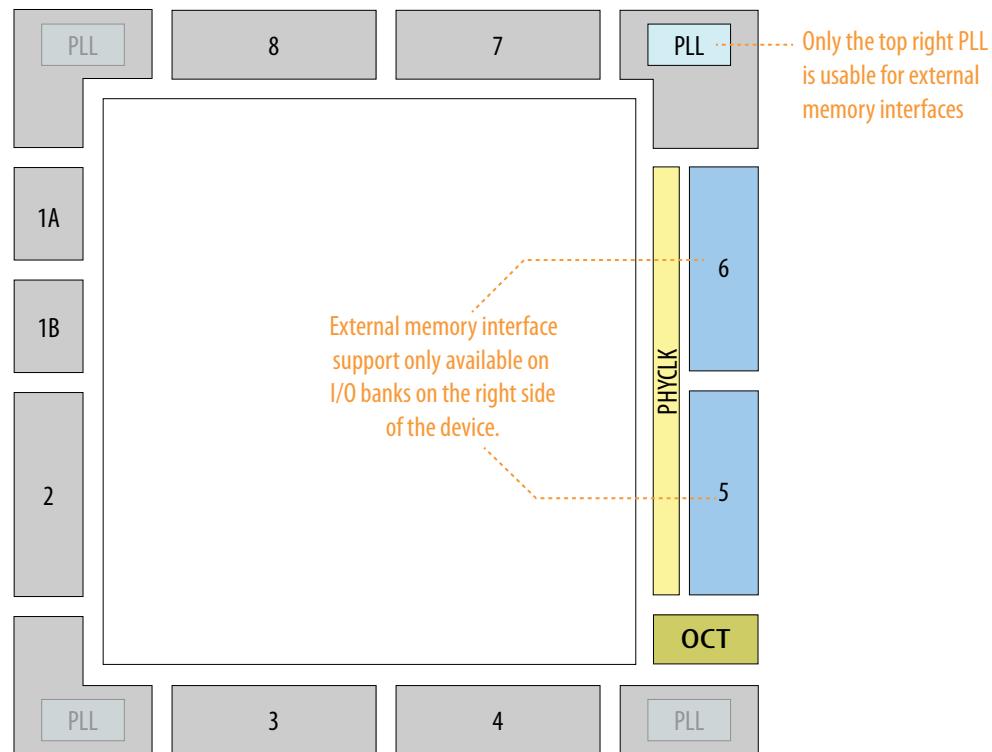
MAX 10 PHY Clock (PHYCLK) Network

The PHYCLK network is a dedicated high-speed and low skew balanced clock tree that provides better clock skew for external memory interface applications.

In MAX 10 devices, only the top right PLL is routed to the PHYCLK tree. Therefore, the PHYCLK tree is available only for the I/O banks on the right side of the MAX 10 10M16, 10M25, 10M40, and 10M50 devices.

Figure 2-7: I/O Banks for External Memory Interfaces
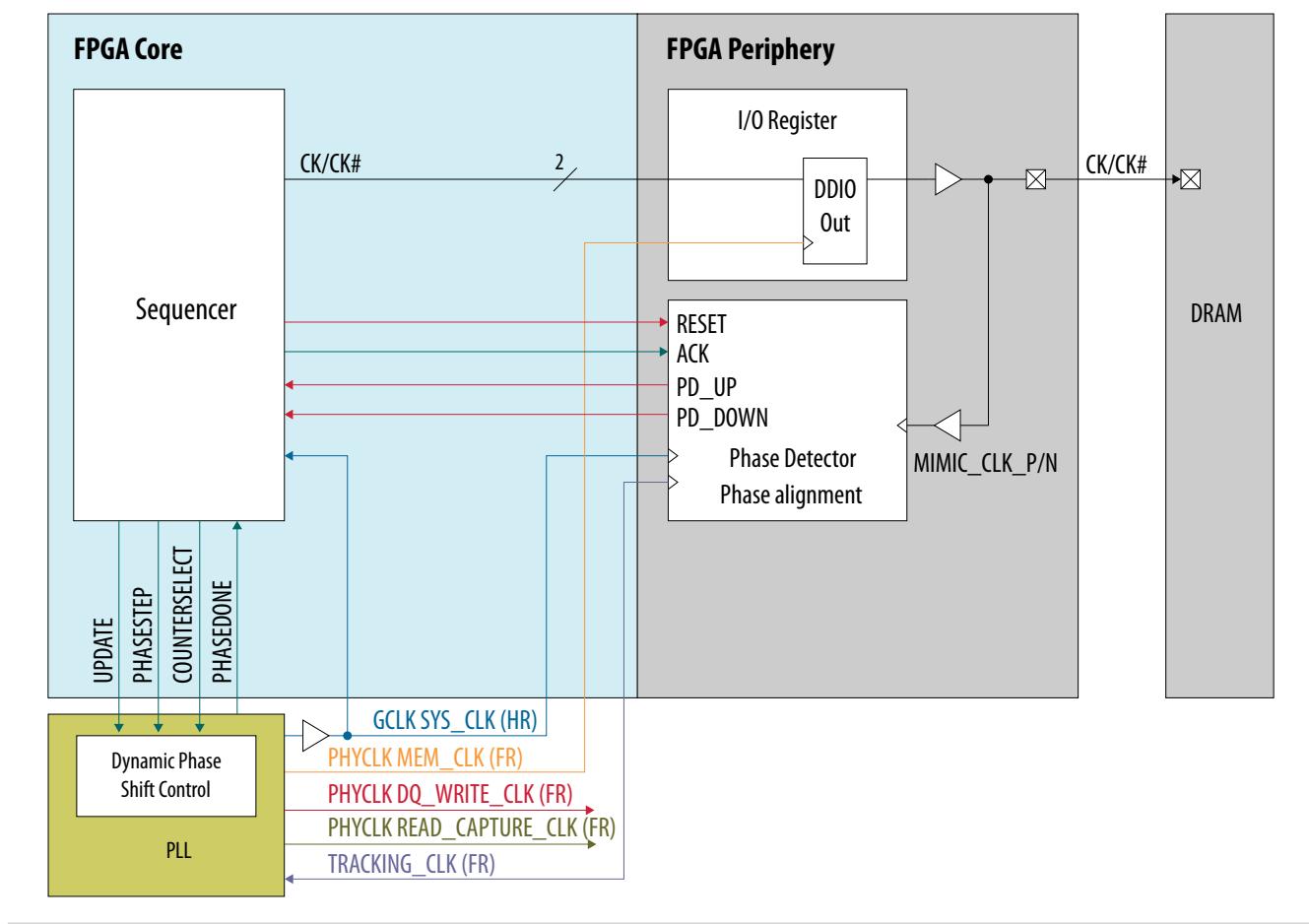
This figure represents the top view of the silicon die that corresponds to a reverse view of the device package.

Send Feedback

Phase Detector for VT Tracking

There may be variations in the read and write paths caused by voltage and temperature changes. The phase detector keeps track of the variation of the mimic clock to optimize the system timing.

Figure 2-8: VT Tracking System Overview

In the MAX 10 external memory interface solution, the memory clocks are used to mimic the read and write paths. The memory clock pins loop back to the phase detector as a mimic clock. The phase detector provides any variation of the mimic clock to the sequencer. The sequencer adjusts the read capture clock to match the clock phase change.

On-Chip Termination

The MAX 10 devices support calibrated on-chip series termination (R_S OCT) on the right side I/O banks.

- To use the calibrated OCT, use the RUP and RDN pins for each R_S OCT control block.
- You can use each OCT calibration block to calibrate one type of termination with the same V_{CCIO}.

You must set the RUP and RDN resistor values according to the R_S OCT value. For example, if the R_S OCT value is 34 Ω, then the set both RUP and RDN value to 34 Ω.

Phase-Locked Loop

For the external memory interface, the PLL generates the memory clock, write clock, capture clock, and the logic–core clock.

- The memory clock provides clock for DQS write strobe, and address and command signals.
- The write clock that is shifted –90° from the memory clock provides clock for DQ signals during memory writes.

You can use the PLL reconfiguration feature to calibrate the read–capture phase shift to balance the setup and hold margins. At startup, the sequencer calibrates the capture clock.

For external memory interfaces in MAX 10 devices, you must use the top right PLL (PLL 2).

MAX 10 Low Power Feature

The MAX 10 low power feature is automatically activated when the self refresh or low power down modes are activated. The low power feature sends the `afi_mem_clk_disable` signal to stop the clock used by the controller.

To conserve power, the MAX 10 UniPHY IP core performs the following functions:

- Tri-states the address and command signals except `CKE` and `RESET_N` signals
- Disables the input buffer of DDR input

Note: The MAX 10 low power feature is available from version 15.0 of the Quartus® Prime software. To enable this feature, regenerate your MAX 10 UniPHY IP core using the Quartus Prime software version 15.0 or later.

There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Related Information

- **MAX 10 External Memory Interface Overview** on page 1-1
- **Planning Pin and FPGA Resources chapter, External Memory Interface Handbook**
  Provides pin planning guidelines for implementing external memory interfaces with Altera devices.

MAX 10 DDR2 and DDR3 Design Considerations

DDR2/DDR3 External Memory Interface Pins

In DDR2/DDR3 interfaces, the MAX 10 devices use data (DQ), data strobe (DQS), clock, address, and command pins to interface with external memory devices. The devices also use the data mask (DM) pins to enable data masking.

Related Information

DDR2/DDR3 Recommended Termination Schemes for MAX 10 Devices on page 3-3

MAX 10 Data and Data Clock (Data Strobe) Pins

For the MAX 10 external memory interfaces, the DQ pins are the data pins for bidirectional read and write, and the DQS pins are the data strobe pins used only during write operations.

The MAX 10 devices support bidirectional data strobes. Connect the bidirectional DQ data signals to the same MAX 10 device DQ pins. The DQS pin is used only during write mode. In read mode, the MAX 10 PHY generates the read capture clock internally and ignores the DQS signal. However, you must still connect DQS signal to the MAX 10 DQS pin.

Related Information

Guidelines: Reading the MAX 10 Pin-Out Files on page 3-8

MAX 10 I/O Bank DQ/DQS Support for DDR2/DDR3

For DDR2/DDR3 SDRAM, I/O banks 5 and 6 in MAX 10 devices can support DQ and DQS signals with DQ-bus widths of 8, 16 and 24 bits.

ISO 9001:2008 Registered

101 Innovation Drive, San Jose, CA 95134

- For DDR2 and DDR3 SDRAM interfaces, the devices use ×8 mode DQS group regardless of the interface width.
- If you need to support wider interfaces, use multiple ×8 DQ groups.
- You can use any unused DQ pins as regular user I/O pins if they are not used as memory interface signals.
- The x24 interface is implemented through x16 + ECC.

Related Information

MAX 10 DQ/DQS Groups on page 2-2
Provides the supported DQ/DQS groups for each device.

Data Mask Pins

In MAX 10 devices, the data mask (DM) pins are pre-assigned in the device pinouts. Although the Quartus Prime Fitter treats the DQ and DM pins in a DQS group equally for placement purposes, the pre-assigned DQ and DM pins are the preferred pins.

Each group of DQS and DQ signals has one DM pin:

- You require data mask (DM) pins only while writing to the external memory devices.
- A low signal on the DM pin indicates that the write is valid.
- Driving the DM pin high causes the memory to mask the DQ signals.
- Similar to the DQ output signals, the DM signals are clocked by the –90º shifted clock.

DDR2/DDR3 Error Correction Coding Pins

Some DDR2 and DDR3 SDRAM devices support error correction coding (ECC). ECC is a method of detecting and automatically correcting errors in data transmission.

- In 24-bit DDR2 or DDR3 SDRAM, there are eight ECC data pins and 16 data pins.
- Connect the DDR2 and DDR3 SDRAM ECC pins to a separate DQS or DQ group in the MAX 10 device.
- The memory controller needs additional logic to encode and decode the ECC data.

Related Information

ALTECC (Error Correction Code: Encoder/Decoder) chapter, Integer Arithmetic Megafunctions User Guide
Provides more information about ALTECC_ENCODER and ALTECC_DECODER IP cores that implement ECC functionality.

DDR2/DDR3 Address and Control/Command Pins

For DDR2/DDR3 interfaces, the address signals and the control or command signals are sent at a single data rate.

You can use any of the user I/O pins on banks 5 & 6 of MAX 10 devices to generate the address and control or command signals to the external memory device.

Memory Clock Pins

At the external memory device, the memory clock signals (CK and CK#) are used to capture the address signals, and the control or command signals.

In MAX 10 devices, the double data rate I/O (DDIO) registers are used to generate the CK/CK# signals.

The memory clock pins are predefined and are listed in the device pinout files. Refer to the the relevant device pinout files to determine the locations of the memory clock pins.

Related Information

- **Pin Connection Guidelines Tables, Planning Pin and FPGA Resources chapter, External Memory Interface Handbook**
  Provides more information about CK/CK# pins placement.
- **MAX 10 Device Pin-Out Files**

DDR2/DDR3 Recommended Termination Schemes for MAX 10 Devices

If you are creating interfaces with multiple DDR2 or DDR3 components where the address, command, and memory clock pins are connected to more than one load, follow these steps:

1. Simulate the system to get the new slew-rate for the DQ/DQS, DM, address and command, and clock signals.
2. Use the derated t_{IS} and t_{IH} specifications from the DDR2 or DDR3 datasheet based on the simulation results.
3. If timing deration causes your interface to fail timing requirements, consider duplication of these signals to lower their loading, and hence improve timing.

Note: Class I and Class II termination schemes in the following tables refer to drive strength and not physical termination.

Table 3-1: Termination Recommendations for MAX 10 DDR2 Component

Signal Type	SSTL 18 I/O Standard	FPGA–End Discrete Termination	Memory–End Termination 1	Memory I/O Standard
DQ/DQS	Class I 12 mA	50 Ω parallel to V_{TT} discrete	ODT75[4]	HALF[5]
DM	Class I 12 mA	—	56 Ω parallel to V_{TT} discrete	—
Address and command	Class I with maximum drive strength	—		—
Clock	Class I 12 mA	—	• x1 = 100 Ω differential[6] • x2 = 200 Ω differential.[7]	—

---

[4] ODT75 vs. ODT50 on the memory has the effect of opening the eye more, with a limited increase in overshoot/undershoot.

[5] HALF is reduced drive strength.

[6] x1 is a single-device load.

[7] x2 is a two-device load.

Table 3-2: On Board Termination Recommendation for MAX 10 DDR3 Component

For MAX 10 devices, on board termination is required for DDR3 component.

I/O Standard	R_S OCT	On Board Termination	
		FPGA–End	Memory-End
SSTL 15 Class 1	50 Ω without calibration	80 Ω resistor	40 Ω resistor

Table 3-3: Supported External Memory Interface Termination Scheme for DDR3 and DDR2

Memory Interface Standard	I/O Standard	R_S OCT	R_{UP}, R_{DN} (Ω)
DDR3	SSTL-15	25	25
		34	34
		40	40
		50	50
DDR3L	SSTL-135	34	34
		40	40
DDR2	SSTL-18	25	25
		50	50

Related Information

Planning Pin and FPGA Resources
Provides more information about termination and signal duplication.

LPDDR2 Design Considerations

Note: MAX 10 devices support single-die LPDDR2 only.

LPDDR2 External Memory Interface Pins

In LPDDR2 interfaces, the MAX 10 devices use data (DQ), data strobe (DQS), clock, command, and address pins to interface with external memory devices. The devices also use the data mask (DM) pins to enable data masking.

MAX 10 Data and Data Clock (Data Strobe) Pins

For the MAX 10 external memory interfaces, the DQ pins are the data pins for bidirectional read and write, and the DQS pins are the data strobe pins used only during write operations.

The MAX 10 devices support bidirectional data strobes. Connect the bidirectional DQ data signals to the same MAX 10 device DQ pins. The DQS pin is used only during write mode. In read mode, the MAX 10 PHY generates the read capture clock internally and ignores the DQS signal. However, you must still connect DQS signal to the MAX 10 DQS pin.

Related Information

MAX 10 I/O Bank DQ/DQS Support for LPDDR2

For LPDDR2 SDRAM, I/O banks 5 and 6 in MAX 10 devices can support DQ and DQS signals with DQ-bus widths of 8 and 16 bits.

- For LPDDR2 SDRAM interfaces, the devices use ×8 mode DQS group regardless of the interface width.
- If you need to support wider interfaces, use multiple ×8 DQ groups.
- You can use any unused DQ pins as regular user I/O pins if they are not used as memory interface signal.

Related Information

MAX 10 DQ/DQS Groups on page 2-2

Provides the supported DQ/DQS groups for each device.

Data Mask Pins

In MAX 10 devices, the data mask (DM) pins are pre-assigned in the device pinouts. Although the Quartus Prime Fitter treats the DQ and DM pins in a DQS group equally for placement purposes, the pre-assigned DQ and DM pins are the preferred pins.

Each group of DQS and DQ signals has one DM pin:

- You require data mask (DM) pins only while writing to the external memory devices.
- A low signal on the DM pin indicates that the write is valid.
- Driving the DM pin high causes the memory to mask the DQ signals.
- Similar to the DQ output signals, the DM signals are clocked by the –90º shifted clock.

LPDDR2 Address and Control/Command Pins

For LPDDR2 interfaces, the address signals and the control or command signals are sent at double data rate.

You can use any of the user I/O pins on banks 5 & 6 of MAX 10 devices to generate the address and control or command signals to the external memory device.

Memory Clock Pins

At the external memory device, the memory clock signals (CK and CK#) are used to capture the address signals, and the control or command signals.

In MAX 10 devices, the double data rate I/O (DDIO) registers are used to generate the CK/CK# signals.

The memory clock pins are predefined and are listed in the device pinout files. Refer to the the relevant device pinout files to determine the locations of the memory clock pins.

Related Information

- **Pin Connection Guidelines Tables, Planning Pin and FPGA Resources chapter, External Memory Interface Handbook**
  Provides more information about CK/CK# pins placement.
- **MAX 10 Device Pin-Out Files**

LPPDDR2 Power Supply Variation Constraint

For an LPDDR2 interface that targets 200 MHz, constrain the memory device I/O and core power supply variation to within ±3%.

- Memory I/O power supply pin is VDDQ
- Memory core power supply pin is VDD

Related Information
MAX 10 Power Management User Guide

LPDDR2 Recommended Termination Schemes for MAX 10 Devices

Table 3-4: Supported External Memory Interface Termination Scheme for LPDDR2

Memory Interface Standard	I/O Standard	R_S OCT	R_{UP}, R_{DN} (Ω)
LPDDR2	HSUL-12	34, 40, 48	34, 40, 48

Guidelines: MAX 10 DDR3, DDR2, and LPDDR2 External Memory Interface I/O Limitation

While implementing certain external memory interface standards, the number of I/O pins available is limited.

- While implementing DDR2—for 25 percent of the remaining I/O pins available in I/O banks 5 and 6, you can assign them only as input pins.
- While implementing DDR3 or LPDDR2—the I/O pins listed in the following table are not available for use. Of the remaining I/O pins, you can assign only 75 percent of the available I/O pins in I/O banks 5 and 6 for normal I/O operation.

Table 3-5: Unavailable I/O Pins While Implementing DDR3 or LPDDR2 External Memory Interfaces in Certain Device Packages—Preliminary

Device	Package			
	F256	U324	F484	F672
10M16	N16	R15	U21	—
	P16	P15	U22	
		R18	M21	
		P18	L22	
		E16	F21	
		D16	F20	
			E19	
			F18	

Device	Package			
	F256	**U324**	**F484**	**F672**
10M25	N16 P16	—	U21 U22 M21 L22 F21 F20 E19 F18 F17 E17	—
10M40 10M50	N16 P16	—	U21 U22 M21 L22 F21 F20 E19 F18 F17 E17	W23 W24 U25 U24 T24 R25 R24 P25 K23 K24 J23 H23 G23 F23 G21 G22

Guidelines: MAX 10 Board Design Requirement for DDR2, DDR3, and LPDDR2

- For DDR2, DDR3, and LPDDR2 interfaces, the maximum board skew between pins must be lower than 40 ps. This guideline applies to all pins (address, command, clock, and data).
- To minimize unwanted inductance from the board via, Altera recommends that you keep the PCB via depth for V_{CCIO} banks below 49.5 mil.
- For devices with DDR3 interface implementation, onboard termination is required for the DQ, DQS, and address signals. Altera recommends that you use termination resistor value of 80 Ω to V_{TT}.
- For the DQ, address, and command pins, keep the PCB trace routing length less than six inches for DDR3, or less than three inches for LPDDR2.

Related Information

- **External Memory Interface Handbook Volume 1: Altera Memory Solution Overview and Design Flow**
  Provides more information about using Altera devices for external memory interfaces including Altera memory solution and design flow.
- **External Memory Interface Handbook Volume 2: Design Guidelines**
  Provides more information about using Altera devices for external memory interfaces including memory selection, board design, implementing memory IP cores, timing, optimization, and debugging.
- **Functional Description—MAX 10 EMIF**
  Provides more information about implementing memory IP cores for MAX 10 devices.
- **MAX 10 FPGA Signal Integrity Design Guidelines**
  Provides design guidelines related to signal integrity for MAX 10 devices.

Guidelines: Reading the MAX 10 Pin-Out Files

For the maximum number of DQ pins and the exact number per group for a particular MAX 10 device, refer to the relevant device pin-out files.

In the pin-out files, the `DQS` and `DQSn` pins denote the differential data strobe/clock pin pairs. The `DQS` and `DQSn` pins are listed in the MAX 10 pin-out files as `DQSXR` and `DQSnXR`:

- *X* indicates the DQ/DQS grouping number.
- `R` indicates the location of the group which is always on the right side of the device.

You can implement your external memory interface design in the Quartus Prime software. The software contains tools for you to create and compile you design, and configure your device.

In the Quartus Prime software, you can instantiate and configure the UniPHY IP core to suit your memory interface requirement.

Related Information

- **MAX 10 External Memory Interface Overview** on page 1-1
- **External Memory Interface Handbook Volume 1: Altera Memory Solution Overview and Design Flow**
  Provides more information about using Altera devices for external memory interfaces including Altera memory solution and design flow.
- **External Memory Interface Handbook Volume 2: Design Guidelines**
  Provides more information about using Altera devices for external memory interfaces including memory selection, board design, implementing memory IP cores, timing, optimization, and debugging.
- **Functional Description—MAX 10 EMIF**
  Provides more information about implementing memory IP cores for MAX 10 devices.
- **MAX 10 DDR3 Reference Design**
  Provides DDR3 UniPHY IP core reference design for MAX 10 devices.

UniPHY IP Core

The UniPHY IP core allows you to control the soft IP of the MAX 10 external memory interface solution.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Figure 4-1: MAX 10 UniPHY IP Core Block Diagram

Related Information
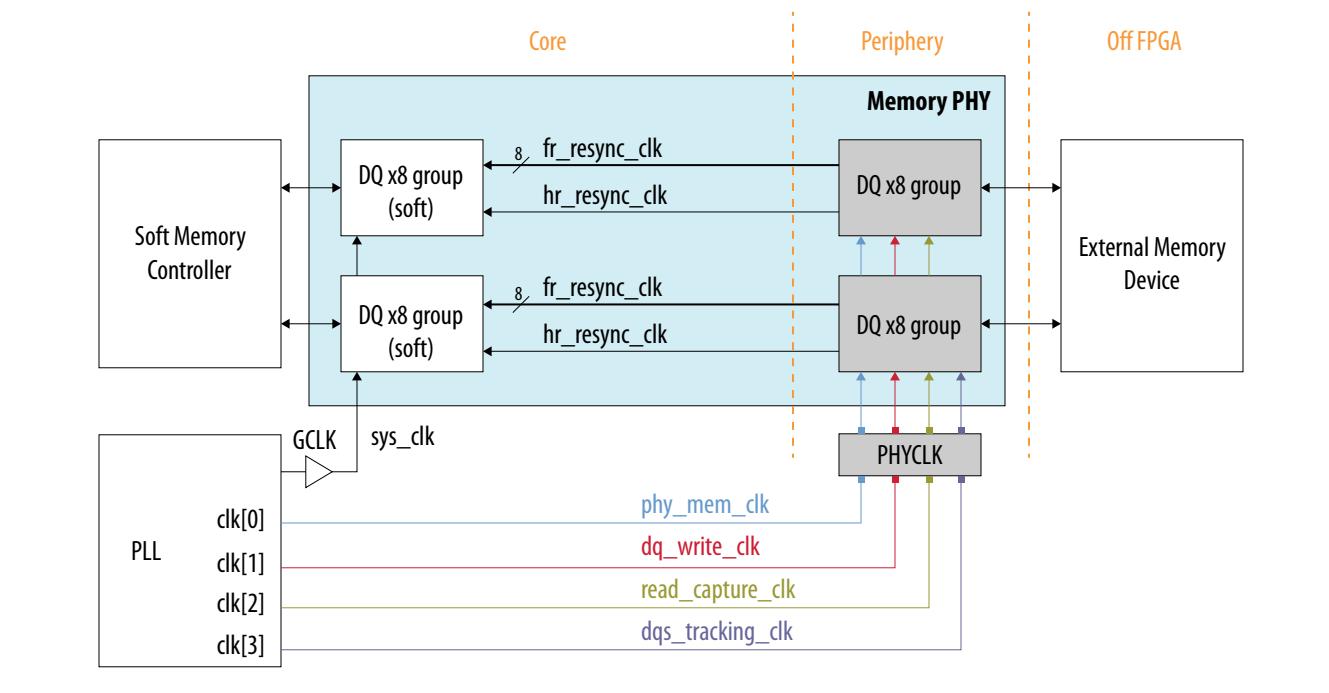
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

LPDDR2 External Memory Interface Implementation

Figure 4-2: Top Level View of LPDDR2 Architecture in MAX 10 Devices

Related Information
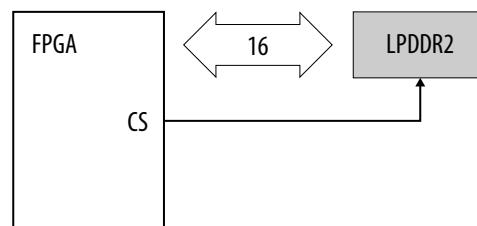
Planning Pin and FPGA Resources chapter, External Memory Interface Handbook
Provides the maximum number of interfaces supported by MAX 10 devices for each memory standards, pin counts for various external memory interface implementation examples, and information about the clock, address/command, data, data strobe, DM, and optional ECC signals.

Supported LPDDR2 Topology

For LPDDR2, the external memory interface IP for MAX 10 devices uses one capture clock and one tracking clock with one discrete device.
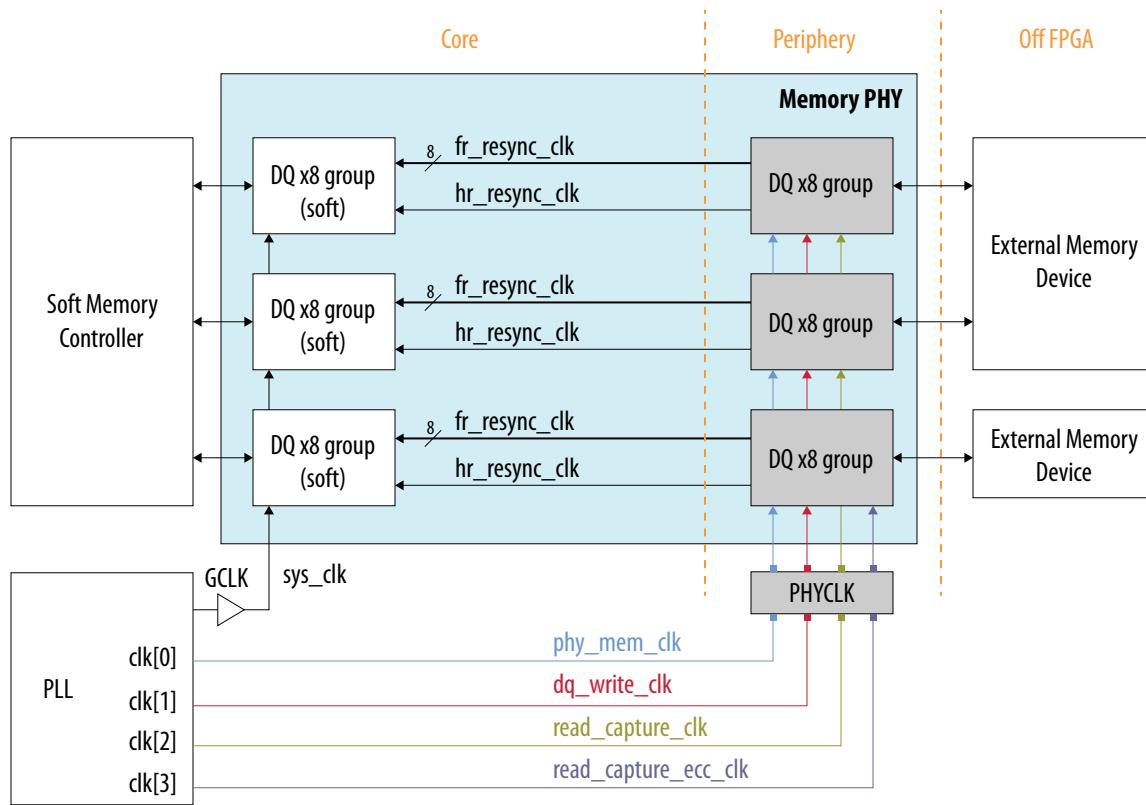
Figure 4-3: Supported Topology for LPDDR2 Memory Interfaces

This figure shows the supported LPDDR2 topology. Only one discrete LPDDR2 device is supported with a 16 bit maximum interface width. The memory interface IP in MAX 10 devices generates LPDDR2 IPs targeted for this configuration only.

DDR2 and DDR3 External Memory Interface Implementation

Figure 4-4: Top Level View of DDR2, DDR3, or DDR3L Architecture in MAX 10 Devices

Related Information
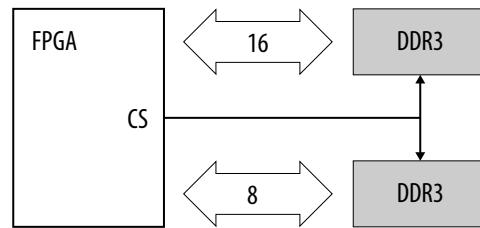
Planning Pin and FPGA Resources chapter, External Memory Interface Handbook
Provides the maximum number of interfaces supported by MAX 10 devices for each memory standards, pin counts for various external memory interface implementation examples, and information about the clock, address/command, data, data strobe, DM, and optional ECC signals.

MAX 10 Supported DDR2 or DDR3 Topology

For DDR2 or DDR3/DDR3L, the external memory interface IP for MAX 10 devices uses two capture clocks with two discrete devices.

Figure 4-5: Supported Topology for DDR2 or DDR3 Memory Interfaces

This figure shows the supported DDR2/DDR3 topology. One clock captures the lower 16 bit of data and the other clock captures the top 8 bit of data. The memory interface IP in MAX 10 devices generates DDR2 or DDR3/DDR3L IPs targeted for this configuration only.

For MAX 10 devices, there are three variations of the UniPHY IP core:

- DDR2 SDRAM Controller
- DDR3 SDRAM Controller
- LPDDR2 SDRAM Controller

Related Information

- **MAX 10 External Memory Interface Overview** on page 1-1
- **External Memory Interface Handbook Volume 1: Altera Memory Solution Overview and Design Flow**
  Provides more information about using Altera devices for external memory interfaces including Altera memory solution and design flow.
- **External Memory Interface Handbook Volume 2: Design Guidelines**
  Provides more information about using Altera devices for external memory interfaces including memory selection, board design, implementing memory IP cores, timing, optimization, and debugging.
- **Functional Description—MAX 10 EMIF**
  Provides more information about implementing memory IP cores for MAX 10 devices.

UniPHY Parameter Settings for MAX 10

You can set the parameter settings for the UniPHY IP core in the Quartus Prime software. There are six groups of options: **PHY Settings**, **Memory Parameters**, **Memory Timing**, **Board Settings**, **Controller Settings**, and **Diagnostics**.

Note: MAX 10 devices are not supported in the EMIF Debug Toolkit.

UniPHY Parameters—PHY Settings

There are three groups of options: **General Settings**, **Clocks**, and **Advanced PHY Settings**.

Table 5-1: PHY Settings - General Settings

Parameter	Description
Speed Grade	Specifies the speed grade of the targeted FPGA device that affects the generated timing constraints and timing reporting. **Note:** For MAX 10 devices, DDR3 and LPDDR2 is supported only for speed grade –6, and DDR2 for speed grades –6 and –7.
Generate PHY only	Turn on this option to generate the UniPHY IP core without a memory controller. When you turn on this option, the AFI interface is exported so that you can easily connect your own memory controller.

Table 5-2: PHY Settings - Clocks

Parameter	Description
Memory clock frequency	The frequency of the clock that drives the memory device. Use up to 4 decimal places of precision. To obtain the maximum supported frequency for your target memory configuration, refer to the External Memory Spec Estimator page on the Altera website.
Achieved memory clock frequency	The actual frequency the PLL generates to drive the external memory interface (memory clock).
PLL reference clock frequency	The frequency of the input clock that feeds the PLL. Use up to 4 decimal places of precision.
Rate on Avalon-MM interface	The width of data bus on the Avalon-MM interface. The MAX 10 supports only **Half** rate, which results in a width of 4× the memory data width.
Achieved local clock frequency	The actual frequency the PLL generates to drive the local interface for the memory controller (AFI clock).

Table 5-3: DDR3 SDRAM PHY Settings - Advanced PHY Settings

Parameter	Description
Supply voltage	The supply voltage and sub-family type of memory. This option is available for DDR3 SDRAM only.
I/O standard	The I/O standard voltage. Set the I/O standard according to your design's memory standard.

Parameter	Description
Reconfigurable PLL location	If you set the PLL used in the UniPHY IP core memory interface to be reconfigurable at run time, you must specify the location of the PLL. This assignment generates a PLL that can only be placed in the given sides.

Related Information

External Memory Interface Spec Estimator

Provides a parametric tool that allows you to find and compare the performance of the supported external memory interfaces in Altera devices.

UniPHY Parameters—Memory Parameters

There are three groups of options: **Memory Parameters**, **Memory Topology**, and **Memory Initialization Options**.

Table 5-4: Memory Parameters

Use the **Memory Parameters** options group to apply the memory parameters from your memory manufacturer's data sheet.

Parameter	Description
Memory vendor	The vendor of the memory device. Select the memory vendor according to the memory vendor you use. For memory vendors that are not listed in the setting, select JEDEC with the nearest memory parameters and edit the parameter values according to the values of the memory vendor that you use. However, if you select a configuration from the list of memory presets, the default memory vendor for that preset setting is automatically selected.
Memory format	The format of the memory device. This parameter is automatically set to **Discrete Device**.
Memory device speed grade	The maximum frequency at which the memory device can run.
Total interface width	The total number of DQ pins of the memory device. Limited to 8 to 24 bits.
DQ/DQS group size	The number of DQ bits per DQS group.
Number of DQS groups	The number of DQS groups is calculated automatically from the **Total interface width** and the **DQ/DQS group size** parameters.
Number of chip selects	The number of chip-selects the IP core uses for the current device configuration. Specify the total number of chip-selects according to the number of memory device.
Number of clocks	The width of the clock bus on the memory interface.

Parameter	Description
Row address width	The width of the row address on the memory interface.
Column address width	The width of the column address on the memory interface.
Bank-address width	The width of the bank address bus on the memory interface.
Enable DM pins	Specifies whether the DM pins of the memory device are driven by the FPGA. You can turn off this option to avoid overusing FPGA device pins when using x4 mode memory devices. When you are using x4 mode memory devices, turn off this option for DDR3 SDRAM. You must turn on this option if you are using Avalon byte enable.
DQS# Enable	Turn on differential DQS signaling to improve signal integrity and system performance. This option is available for DDR2 SDRAM only.

Table 5-5: Memory Parameters - Memory Initialization Options (DDR3 SDRAM)

This table lists the memory initialization options for DDR3 SDRAM.

Parameter		Description
Mode Register 0	Read burst type	Specifies accesses within a given burst in sequential or interleaved order. Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the **Generate PHY only** parameter is enabled on the PHY Settings tab.
	DLL precharge power down	Specifies whether the DLL in the memory device is off or on during precharge power-down.
	Memory CAS latency setting	The number of clock cycles between the read command and the availability of the first bit of output data at the memory device and also interface frequency. Refer to memory vendor data sheet speed bin table. Set this parameter according to the target memory speed grade and memory clock frequency.

Parameter		Description
Mode Register 1	**Output drive strength setting**	The output driver impedance setting at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
	Memory additive CAS latency setting	The posted CAS additive latency of the memory device. Enable this feature to improve command and bus efficiency, and increase system bandwidth. For more information about optimizing the memory controller, refer to related information.
	ODT Rtt nominal value	The on-die termination resistance at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
Mode Register 2	**Auto selfrefresh method**	Disable or enable auto selfrefresh.
	Selfrefresh temperature	Specifies the selfrefresh temperature as **Normal** or **Extended**.
	Memory write CAS latency setting	The number of clock cycles from the releasing of the internal write to the latching of the first data in, at the memory device and also interface frequency. Refer to memory vendor data sheet speed bin table and set according to the target memory speed grade and memory clock frequency.
	Dynamic ODT (Rtt_WR) value	The mode of the dynamic ODT feature of the memory device. This is used for multi-rank configurations. For more guidelines about DDR2 and DDR3 SDRAM board layout, refer to the related information. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.

Table 5-6: Memory Parameters - Memory Initialization Options (DDR2 SDRAM)

This table lists the memory initialization options for DDR2 SDRAM.

Parameter		Description
Mode Register 0	**Burst length**	Specifies the burst length.
	Read burst type	Specifies accesses within a given burst in sequential or interleaved order.
		Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the **Generate PHY only** parameter is enabled on the PHY Settings tab.
	DLL precharge power down	Determines whether the DLL in the memory device is in slow exit mode or in fast exit mode during precharge power down. For more information, refer to memory vendor data sheet.
	Memory CAS latency setting	Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device. For more information, refer to memory vendor data sheet speed bin table.
		Set this parameter according to the target memory speed grade and memory clock frequency.
Mode Register 1	**Output drive strength setting**	Determines the output driver impedance setting at the memory device.
		To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
	Memory additive CAS latency setting	Determines the posted CAS additive latency of the memory device.
		Enable this feature to improve command and bus efficiency, and increase system bandwidth.
	Memory on-die termination (ODT) setting	Determines the on-die termination resistance at the memory device.
		To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
Mode Register 2	**SRT Enable**	Determines the selfrefresh temperature (SRT). Select **1x refresh rate** for normal temperature (0-85C)or select **2x refresh rate** for high temperature (>85C).

Table 5-7: Memory Parameters - Memory Initialization Options (LPDDR2 SDRAM)

This table lists the memory initialization options for LPDDR2 SDRAM.

Parameter		Description
Mode Register 1	Burst Length	Specifies the burst length.
	Read Burst Type	Specifies accesses within a given burst in sequential or interleaved order. Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the **Generate PHY only** parameter is enabled on the PHY Settings tab.
Mode Register 2	Read latency setting	Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device. Set this parameter according to the target memory interface frequency. Refer to memory data sheet and also target memory speed grade.
Mode Register 3	Output drive strength settings	Determines the output driver impedance setting at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.

Related Information

- **External Memory Interface Handbook Volume 1: Altera Memory Solution Overview and Design Flow**
  Provides more information about using Altera devices for external memory interfaces including Altera memory solution and design flow.
- **External Memory Interface Handbook Volume 2: Design Guidelines**
  Provides more information about using Altera devices for external memory interfaces including memory selection, board design, implementing memory IP cores, timing, optimization, and debugging.
- **Functional Description—MAX 10 EMIF**
  Provides more information about implementing memory IP cores for MAX 10 devices.

UniPHY Parameters—Memory Timing

Use the **Memory Timing** options to apply the memory timings from your memory manufacturer's data sheet.

Table 5-8: Memory Timing

For each parameter, refer to the memory vendor data sheet.

Parameter	Applies To	Description	Set According To
tIS (base)	DDR2, DDR3, LPDDR2	Address and control setup to CK clock rise.	Memory speed grade
tIH (base)	DDR2, DDR3, LPDDR2	Address and control hold after CK clock rise.	Memory speed grade
tDS (base)	DDR2, DDR3, LPDDR2	Data setup to clock (DQS) rise.	Memory speed grade
tDH (base)	DDR2, DDR3, LPDDR2	Data hold after clock (DQS) rise.	Memory speed grade
tDQSQ	DDR2, DDR3, LPDDR2	DQS, DQS# to DQ skew, per access.	Memory speed grade
tQHS	DDR2, LPDDR2	DQ output hold time from DQS, DQS# (absolute time value).	Memory speed grade
tQH	DDR3	DQ output hold time from DQS, DQS# (percentage of tCK).	Memory speed grade
tDQSCK	DDR2, DDR3	DQS output access time from CK/CK#.	Memory speed grade
tDQSCK (max)	LPDDR2		
tDQSCK Delta Short	LPDDR2	Absolute value of the difference between any two tDQSCK measurements (within a byte lane) within a contiguous sequence of bursts within a 160 ns rolling window.	Memory speed grade
tDQSCK Delta Medium	LPDDR2	Absolute value of the difference between any two tDQSCK measurements (within a byte lane) within a contiguous sequence of bursts within a 1.6 μs rolling window.	Memory speed grade
tDQSCK Delta Long	LPDDR2	Absolute value of the difference between any two tDQSCK measurements (within a byte lane) within a contiguous sequence of bursts within a 32ms rolling window.	Memory speed grade
tDQSS	DDR2, DDR3, LPDDR2	First latching edge of DQS to associated clock edge (percentage of tCK).	Memory speed grade
tDQSH	DDR2, LPDDR2	DQS Differential High Pulse Width (percentage of tCK). Specifies the minimum high time of the DQS signal received by the memory.	Memory speed grade
tQSH	DDR3		
tDSH	DDR2, DDR3, LPDDR2	DQS falling edge hold time from CK (percentage of tCK).	Memory speed grade

Send Feedback

Parameter	Applies To	Description	Set According To
tDSS	DDR2, DDR3, LPDDR2	DQS falling edge to CK setup time (percentage of tCK).	Memory speed grade
tINIT	DDR2, DDR3, LPDDR2	Memory initialization time at power-up.	Memory speed grade
tMRD	DDR2, DDR3	Load mode register command period.	Memory speed grade
tMRW	LPDDR2		
tRAS	DDR2, DDR3, LPDDR2	Active to precharge time.	Memory speed grade
tRCD	DDR2, DDR3, LPDDR2	Active to read or write time.	Memory speed grade
tRP	DDR2, DDR3, LPDDR2	Precharge command period.	Memory speed grade
tREFI	DDR2, DDR3	Refresh command interval.	Memory speed grade and temperature range
tREFIab	LPDDR2	Refresh command interval (all banks).	Memory speed grade
tRFC	DDR2, DDR3	Auto-refresh command interval.	Memory device capacity
tRFCab	LPDDR2	Auto-refresh command interval (all banks).	Memory device capacity
tWR	DDR2, DDR3, LPDDR2	Write recovery time.	Memory speed grade
tWTR	DDR2, DDR3, LPDDR2	Write to read period. Calculate the value based on the memory clock frequency.	Memory speed grade and memory clock frequency
tFAW	DDR2, DDR3, LPDDR2	Four active window time.	Memory speed grade and page size
tRRD	DDR2, DDR3, LPDDR2	RAS to RAS delay time. Calculate the value based on the memory clock frequency.	Memory speed grade, page size and memory clock frequency
tRTP	DDR2, DDR3, LPDDR2	Read to precharge time. Calculate the value based on the memory clock frequency.	Memory speed grade and memory clock frequency

UniPHY Parameters—Board Settings

There are three groups of options: **Setup and Hold Derating**, **Channel Signal Integrity**, and **Board Skews**.

Send Feedback

Table 5-9: Board Settings - Setup and Hold Derating

The slew rate of the output signals affects the setup and hold times of the memory device, and thus the write margin. You can specify the slew rate of the output signals to see their effect on the setup and hold times of both the address and command signals and the DQ signals, or alternatively, you may want to specify the setup and hold times directly. You should enter information derived during your PCB development process of prelayout (line) and postlayout (board) simulation.

Parameter	Description
Derating method	Derating method. The default settings are based on Altera internal board simulation data. To obtain accurate timing analysis according to the condition of your board, Altera recommends that you perform board simulation and enter the slew rate in the Quartus Prime software to calculate the derated setup and hold time automatically or enter the derated setup and hold time directly.
CK/CK# slew rate (differential)	CK/CK# slew rate (differential).
Address/Command slew rate	Address and command slew rate.
DQS/DQS# slew rate (Differential)	DQS and DQS# slew rate (differential).
DQ slew rate	DQ slew rate.
tIS	Address/command setup time to CK.
tIH	Address/command hold time from CK.
tDS	Data setup time to DQS.
tDH	Data hold time from DQS.

Table 5-10: Board Settings - Channel Signal Integrity

Channel signal integrity is a measure of the distortion of the eye due to intersymbol interference, crosstalk, or other effects. Typically, when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss, because there are multiple stubs causing reflections. Although the Quartus Prime timing models include some channel uncertainty, you must perform your own channel signal integrity simulations and enter the additional channel uncertainty, relative to the reference eye, into the parameter editor.

Parameter	Description
Derating method	Choose between default Altera settings (with specific Altera boards) or manually enter board simulation numbers obtained for your specific board.
Address and command eye reduction (setup)	The reduction in the eye diagram on the setup side (or left side of the eye) due to ISI on the address and command signals compared to a case when there is no ISI. (For single rank designs, ISI can be zero; in multirank designs, ISI is necessary for accurate timing analysis.)

Parameter	Description
Address and command eye reduction (hold)	The reduction in the eye diagram on the hold side (or right side of the eye) due to ISI on the address and command signals compared to a case when there is no ISI.
Write DQ eye reduction	The total reduction in the eye diagram due to ISI on DQ signals compared to a case when there is no ISI. Altera assumes that the ISI reduces the eye width symmetrically on the left and right side of the eye.
Read DQ eye reduction	
Write Delta DQS arrival time	The increase in variation on the range of arrival times of DQS compared to a case when there is no ISI. Altera assumes that the ISI causes DQS to further vary symmetrically to the left and to the right.
Read Delta DQS arrival time	

Table 5-11: Board Settings - Board Skews

PCB traces can have skews between them that can reduce timing margins. Furthermore, skews between different chip selects can further reduce the timing margin in multiple chip-select topologies. This section allows you to enter parameters to compensate for these variations.

Note: Altera recommends that you use the Board Skew Parameter Tool to help you calculate the board skews. For more information, refer to the related information section.

Parameter	Description
Maximum CK delay to DIMM/device	The delay of the longest CK trace from the FPGA to the memory device is expressed by the following equation: $$max_r\left[max_n\left(CK_{n\_r}PathDelay\right)\right]$$ Where n is the number of memory clock and r is number rank of device.
Maximum DQS delay to DIMM/ device	The delay of the longest DQS trace from the FPGA to the memory device, whether on a DIMM or the same PCB as the FPGA is expressed by the following equation: $$max_r\left[max_n\left(DQS_{n\_r}PathDelay\right)\right]$$ Where n is the number of DQS and r is number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 DQS in each rank DIMM, the maximum DQS delay is expressed by the following equation: $max(DQS_1PathDelay\ rank\ 1, DQS_2PathDelay\ rank\ 1, DQS_1PathDelay\ rank\ 2, DQS_2PathDelay\ rank\ 2)$

Parameter	Description
Minimum delay difference between CK and DQS	The minimum skew or smallest positive skew (or largest negative skew) between the CK signal and any DQS signal when arriving at the same DIMM/device over all DIMMs/devices is expressed by the following equation: $$min_r \begin{bmatrix} min \\ n, m \end{bmatrix} \{(CK_{n\_r}Delay - DQS_{m\_r}Delay)\}$$ Where n is the number of memory clock, m is the number of DQS, and r is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clock and 4 DQS signals (two for each clock) for each rank DIMM, the minimum delay difference between CK and DQS is expressed by the following equation: $$min \begin{Bmatrix} (CK_{1\_1}Delay - DQS_{1\_1}Delay), (CK_{1\_1}Delay - DQS_{2\_1}Delay), (CK_{2\_1}Delay - DQS_{3\_1}Delay), (CK_{2\_1}Delay - DQS_{4\_1}Delay) \\ (CK_{1\_2}Delay - DQS_{1\_2}Delay), (CK_{1\_2}Delay - DQS_{2\_2}Delay), (CK_{2\_2}Delay - DQS_{3\_2}Delay), (CK_{2\_1}Delay - DQS_{4\_2}Delay) \end{Bmatrix}$$ This parameter value affects the write leveling margin for DDR3 interfaces with leveling in multi-rank configurations. This parameter value also applies to non-leveling configurations of any number of ranks with the requirement that DQS must have positive margins in Timequest Report DDR. For multiple boards, the minimum skew between the CK signal and any DQS signal when arriving at the same DIMM over all DIMMs is expressed by the following equation, if you want to use the same design for several different boards: $$\overset{boards}{\underset{b}{Min}} \left[\overset{groups}{\underset{g}{Min}} \left[CK_{g\_b} - DQS_{g\_b} \right] \right]$$

Parameter	Description
Maximum delay difference between CK and DQS	The maximum skew or smallest negative skew (or largest positive skew) between the CK signal and any DQS signal when arriving at the same DIMM/device over all DIMMs/devices is expressed by the following equation: $$max_r\left[\begin{matrix}max\\n,m\end{matrix}\{(CK_{n\_r}Delay - DQS_{m\_r}Delay)\}\right]$$ Where n is the number of memory clock, m is the number of DQS, and r is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clock and 4 DQS signals (two for each clock) for each rank DIMM, the maximum delay difference between CK and DQS is expressed by the following equation: $$max\left\{\begin{matrix}\left((CK_{1\_1}Delay - DQS_{1\_1}Delay),(CK_{1\_1}Delay - DQS_{2\_1}Delay),(CK_{2\_1}Delay - DQS_{3\_1}Delay),(CK_{2\_1}Delay - DQS_{4\_1}Delay)\right)\\\left((CK_{1\_2}Delay - DQS_{1\_2}Delay),(CK_{1\_2}Delay - DQS_{2\_2}Delay),(CK_{2\_2}Delay - DQS_{3\_2}Delay),(CK_{2\_1}Delay - DQS_{4\_2}Delay)\right)\end{matrix}\right\}$$ This value affects the write Leveling margin for DDR3 interfaces with leveling in multi-rank configurations. This parameter value also applies to non-leveling configurations of any number of ranks with the requirement that DQS must have positive margins in Timequest Report DDR. For multiple boards, the maximum skew (or largest positive skew) between the CK signal and any DQS signal when arriving at the same DIMM over all DIMMs is expressed by the following equation, if you want to use the same design for several different boards: $$\underset{b}{\overset{boards}{Max}}\left[\underset{g}{\overset{groups}{Max}}\left[CK_{g\_b} - DQS_{g\_b}\right]\right]$$
Maximum skew within DQS group	The largest skew among DQ and DM signals in a DQS group. This value affects the read capture and write margins for DDR2 and DDR3 SDRAM interfaces in all configurations (single or multiple chip-select, DIMM or component). For multiple boards, the largest skew between DQ and DM signals in a DQS group is expressed by the following equation: $$\underset{b}{\overset{boards}{Max}}\left[\underset{g}{\overset{groups}{Max}}\left[maxDQ_{g\_b} - minDQ_{g\_b}\right]\right]$$

Parameter	Description
Maximum skew between DQS groups	The largest skew between DQS signals in different DQS groups. This value affects the resynchronization margin in memory interfaces without leveling such as DDR2 SDRAM and discrete-device DDR3 SDRAM in both single- or multiple chip-select configurations. For multiple boards, the largest skew between DQS signals in different DQS groups is expressed by the following equation, if you want to use the same design for several different boards: $$\underset{b}{\overset{boards}{Max}}\left[\underset{g}{\overset{groups}{Max}}\left[DQS_{g\_b}\right]-\underset{b}{\overset{boards}{Min}}\left[\underset{g}{\overset{boards}{Min}}\left[DQS_{g\_b}\right]\right]\right]$$
Average delay difference between DQ and DQS	The average delay difference between each DQ signal and the DQS signal, calculated by averaging the longest and smallest DQ signal delay values minus the delay of DQS. The average delay difference between DQ and DQS is expressed by the following equation: $$\frac{\sum{\binom{n=n}{n=1}}\left[\left(\dfrac{\begin{matrix}Longest\ DQ\ PathDelay\\in\ DQS_n\ group\end{matrix}+\begin{matrix}Shortest\ DQ\ PathDelay\\in\ DQS_n\ group\end{matrix}}{2}\right)-DQS_n PathDelay\right]}{n}$$ where n is the number of DQS groups. For multi-rank or multiple CS configuration, the equation is: $$\frac{\sum{\binom{r=r}{r=1}}\left[\begin{matrix}Average\ delay\ difference\\between\ DQ\ and\ DQS\ in\ rank\ r\end{matrix}\right]}{r}$$
Maximum skew within address and command bus	The largest skew between the address and command signals for a single board is expressed by the following equation: $$\frac{(MaxACdelay-MinCKdelay)-(MinACdelay-MaxCKdelay)}{2}$$ For multiple boards, the largest skew between the address and command signals is expressed by the following equation, if you want to use the same design for several different boards: $$\frac{\underset{b}{\overset{boards}{Max}}\left[(MaxAC_b-MinCK_b)-\underset{b}{\overset{boards}{Min}}(MaxAC_b-MinCK_b)\right]}{2}$$

Parameter	Description
Average delay difference between address and command and CK	A value equal to the average of the longest and smallest address and command signal delay values, minus the delay of the CK signal. The value can be positive or negative. Positive values represent address and command signals that are longer than CK signals; negative values represent address and command signals that are shorter than CK signals. The average delay difference between address and command and CK is expressed by the following equation: $$\frac{\sum\binom{n=n}{n=1}\left[\left(\frac{Longest\ AC\ PathDelay + Shortest\ AC\ PathDelay}{2}\right) - CK_n PathDelay\right]}{n}$$ where n is the number of memory clocks. For multi-rank or multiple CS configuration, the equation is: $$\frac{\sum\binom{r=r}{r=1}[Average\ delay\ difference\ between\ AC\ and\ CK\ in\ rank\ r]}{r}$$ The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins for DDR2 and DDR3 SDRAM interfaces. You should derive this value through board simulation. For multiple boards, the average delay difference between address and command and CK is expressed by the following equation, if you want to use the same design for several different boards: $$\underset{b}{\overset{boards}{Avg}}\left[\left(\frac{MaxAC_b + MinAC_b}{2}\right) - \left(\frac{MaxCK_b + MinCK_b}{2}\right)\right]$$

Related Information

- **Analizing Timing of Memory IP chapter, External Memory Interface Handbook**
  Provides more information about derating method and measuring eye reduction.
- **Board Skew Parameter Tool**

UniPHY Parameters—Controller Settings

There are four groups of options: **Avalon Interface**, **Low Power Mode**, **Efficiency**, and **Configuration, Status and Error Handling**.

Table 5-12: Controller Settings - Avalon Interface

Parameter	Descriptions
Generate power-of-2 data bus widths for Qsys or SOPC Builder	Rounds down the Avalon-MM side data bus to the nearest power of 2. You must enable this option for Qsys systems. If this option is enabled, the Avalon data buses are truncated to 256 bits wide. One Avalon read-write transaction of 256 bit width maps to four memory beat transactions, each of 72 bits (8 MSB bits are zero, while 64 LSB bits carry useful content). The four memory beats may comprise an entire burst length-of-4 transaction, or part of a burst-length-of-8 transaction.
Generate SOPC Builder compatible resets	This option is not required when using the MegaWizard Plug-in Manager or Qsys.
Maximum Avalon-MM burst length	Specifies the maximum burst length on the Avalon-MM bus. Affects the AVL_SIZE_WIDTH parameter.
Enable Avalon-MM byte-enable signal	When you turn on this option, the controller adds the byte enable signal (avl_be) for the Avalon-MM bus to control the data mask (mem_dm) pins going to the memory interface. You must also turn on **Enable DM pins** if you are turning on this option. When you turn off this option, the byte enable signal (avl_be) is not enabled for the Avalon-MM bus, and by default all bytes are enabled. However, if you turn on **Enable DM pins** with this option turned off, all write words are written.
Avalon interface address width	The address width on the Avalon-MM interface.
Avalon interface data width	The data width on the Avalon-MM interface.

Table 5-13: Controller Settings - Low Power Mode

Parameter	Description
Enable Self-Refresh Controls	Enables the self-refresh signals on the controller top-level design. These controls allow you to control when the memory is placed into self-refresh mode.
Enable Deep Power-Down Controls	Enables the Deep-Powerdown signals on the controller top level. These controls allow you to control when the memory is placed in Deep-Powerdown mode. This option is available only for LPDDR2 SDRAM.
Enable Auto Power-Down	Allows the controller to automatically place the memory into power-down mode after a specified number of idle cycles. Specifies the number of idle cycles after which the controller powers down the memory in the auto-power down cycles parameter.

Parameter	Description
Auto Power-Down Cycles	The number of idle controller clock cycles after which the controller automatically powers down the memory. The legal range is from 1 to 65,535 controller clock cycles.

Table 5-14: Controller Settings - Efficiency

Parameter	Description
Enable User Auto-Refresh Controls	Enables the user auto-refresh control signals on the controller top level. These controller signals allow you to control when the controller issues memory autorefresh commands.
Enable Auto-Precharge Control	Enables the autoprecharge control on the controller top level. Asserting the autoprecharge control signal while requesting a read or write burst allows you to specify whether the controller should close (autoprecharge) the currently open page at the end of the read or write burst.
Local-to-Memory Address Mapping	Allows you to control the mapping between the address bits on the Avalon-MM interface and the chip, row, bank, and column bits on the memory: • **Chip-Row-Bank-Col**—improves efficiency with sequential traffic. • **Chip-Bank-Row-Col**—improves efficiency with random traffic. • **Row-Chip-Bank-Col**—improves efficiency with multiple chip select and sequential traffic.
Command Queue Look-Ahead Depth	Selects a look-ahead depth value to control how many read or writes requests the look-ahead bank management logic examines. Larger numbers are likely to increase the efficiency of the bank management, but at the cost of higher resource usage. Smaller values may be less efficient, but also use fewer resources. The valid range is from 1 to 16.
Enable Reordering	Allows the controller to perform command and data reordering that reduces bus turnaround time and row/bank switching time to improve controller efficiency.
Starvation limit for each command	Specifies the number of commands that can be served before a waiting command is served. The valid range is from 1 to 63.

Table 5-15: Controller Settings - Configuration, Status and Error Handling

Parameter	Description
Enable Configuration and Status Register Interface	Enables run-time configuration and status interface for the memory controller. This option adds an additional Avalon-MM slave port to the memory controller top level, which you can use to change or read out the memory timing parameters, memory address sizes, mode register settings and controller status. If Error Detection and Correction Logic is enabled, the same slave port also allows you to control and retrieve the status of this logic.
CSR port host interface	Specifies the type of connection to the CSR port. The port can be exported, internally connected to a JTAG Avalon Master, or both: • **Internal (JTAG)**—connects the CSR port to a JTAG Avalon Master. • **Avalon-MM Slave** —exports the CSR port. • **Shared**—exports and connects the CSR port to a JTAG Avalon Master.
Enable Error Detection and Correction Logic	Enables ECC for single-bit error correction and double-bit error detection. MAX 10 devices supports ECC only for 16 bits + 8 bits ECC memory configuration.
Enable Auto Error Correction	Allows the controller to perform auto correction when a single-bit error is detected by the ECC logic. To turn this on, you must first turn on **Enable Error Detection and Correction Logic**.

UniPHY Parameters—Diagnostics

There is one option group supported for MAX 10 devices: **Simulation Options**.

Table 5-16: Diagnostics - Simulation Options

Parameter	Description
Enable verbose memory model output	Turn on this option to display more detailed information about each memory access during simulation.

✉ **Subscribe** 💬 **Send Feedback**

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 Exteral Memory Interface User Guide**
15.0	**MAX 10 Exteral Memory Interface User Guide**
14.1	**MAX 10 Exteral Memory Interface User Guide**

**ISO
9001:2008
Registered**

ALTERA
now part of Intel

Document Revision History for MAX 10 External Memory Interface User Guide

Date	Version	Changes
May 2016	2016.05.02	• Updated UniPHY IP core parameter settings for LPDDR2, DDR2 and DDR3. • Updated Supported Maximum External Memory Interface Width in MAX 10 Device Packages table. • Added MAX 10 External Memory Interface User Guide Archives table. • Updated DDR2, DDR3 and LPDDR2 can use only user I/O pins from banks 5 and 6 of MAX 10 devices to generate address and control or command signals..
November 2015	2015.11.02	• Added links to MAX 10 DDR3 UniPHY IP core reference design. • Added topic that lists the maximum external memory interface widths supported for different MAX 10 device packages. • Removed the topics about the IP catalog and parameter editor, generating IP cores, and the files generated by the IP core, and added a link to *Introduction to Altera IP Cores*. • Changed instances of *Quartus II* to *Quartus Prime*.
May 2015	2015.05.11	Added on board termination recommendation for DDR3 component.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Date	Version	Changes
May 2015	2015.05.04	• Updated the footnote in the topic about external memoy interface support and performance to specify that the default maximum frequency for LPDDR2 is 167 MHz. • Removed the F672 package from the 10M25 device. • Removed the note about contacting Altera for DDR3, DDR3L, DDR2, and LPDDR2 external memory interface support. The Quartus Prime software supports these external memory interfaces from version 15.0. • Added a topic about the PHYCLK network. • Moved information about recommended LPDDR2 termination scheme into a new topic under LPDDR2 design considerations section. The information was previously in the topic about recommended DDR2/DDR3 termination schemes. • Updated the guidelines about board design requirement to improve clarity. • Updated and added related information links to relevant information. • Added a topic about the low power feature available from version 15.0 of the Quartus Prime software. • Updated the topic about the phase detector to add a figure showing the VT tracking system overview.
December 2014	2014.12.15	• Changed Altera MAX 10 EMIF IP core to UniPHY IP core. • Removed reference to DIMM in a footnote under the table that lists the termination recommendations for DDR2 component. The UniPHY IP core for MAX 10 does not support DIMM. • Added a list of the MAX 10 memory controller features. • Added "Preliminary" tag to the table that lists the I/Os unavailable in certain MAX 10 packages while implementing DDR3 or LPDDR2 external memory interfaces. • Updated the board design requirement with additional guidelines. • Added information for the MAX 10 external memory interface UniPHY IP core. This addition includes the chapters about external memory interface implementation and IP core references. • Edited texts and added related information links to improve clarity.

Date	Version	Changes
September 2014	2014.09.22	Initial release.

MAX 10 Analog to Digital Converter User Guide

Contents

UG-M10ADC ✉ Subscribe 💬 Send Feedback

MAX® 10 devices feature up to two analog-to-digital converters (ADC). The ADCs provide the MAX 10 devices with built-in capability for on-die temperature monitoring and external analog signal conversion.

The ADC solution consists of hard IP blocks in the MAX 10 device periphery and soft logic through the Altera Modular ADC IP core.

The ADC solution provides you with built-in capability to translate analog quantities to digital data for information processing, computing, data transmission, and control systems. The basic function is to provide a 12 bit digital representation of the analog signal being observed.

The ADC solution works in two modes:

- Normal mode—monitors single-ended external inputs with a cumulative sampling rate of up to 1 million samples per second (MSPS):
  - Single ADC devices—up to 17 single-ended external inputs (one dedicated analog and 16 dual function input pins)
  - Dual ADC devices—up to 18 single-ended external inputs (one dedicated analog and eight dual function input pins in each ADC block)
- Temperature sensing mode—monitors external temperature data input with a sampling rate of up to 50 kilosamples per second. In dual ADC devices, only the first ADC block supports this mode.

Related Information

- **MAX 10 ADC Architecture and Features** on page 2-1
- **MAX 10 ADC Design Considerations** on page 3-1
- **MAX 10 ADC Implementation Guides** on page 4-1
- **Altera Modular ADC and Altera Modular Dual ADC IP Cores References** on page 5-1
- **How to Create ADC Design in MAX 10 Device Using Qsys Tool**
  Provides video instruction that demonstrates how to create the ADC design in MAX 10 devices using the Qsys system integration tool within the Quartus® Prime software and how to use the ADC toolkit to view the measured analog signal.
- **How to Create Simultaneous Measurement with MAX 10 ADC, Part 1**
  Provides the first part of video instruction series that explains the differences between the MAX 10 Altera Modular ADC and Altera Modular Dual ADC IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.

ALTERA

now part of Intel

- **How to Create Simultaneous Measurement with MAX 10 ADC, Part 2**
  Provides the second part of video instruction series that explains the differences between the MAX 10 Altera Modular ADC and Altera Modular Dual ADC IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.

ADC Block Counts in MAX 10 Devices

The ADC block is available in single and dual supply MAX 10 devices.

Table 1-1: Number of ADC Blocks in MAX 10 Devices and Packages

For more information about the device part numbers that feature ADC blocks, refer to the device overview.

Package	Power Supply	Device					
		10M04	10M08	10M16	10M25	10M40	10M50
M153	Single	1	1	—	—	—	—
U169	Single	1	1	1	—	—	—
U324	Dual	1	1	1	—	—	—
F256	Dual	1	1	1	2	2	2
E144	Single	1	1	1	1	1	1
F484	Dual	—	1	1	2	2	2
F672	Dual	—	—	—	—	2	2

Related Information
MAX 10 FPGA Device Overview

ADC Channel Counts in MAX 10 Devices

Different MAX 10 devices support different number of ADC channels.

Table 1-2: ADC Channel Counts in MAX 10 Devices

- Devices with two ADC blocks have two dedicated analog inputs and each ADC block has 8 dual function pins. You can use the dual function pins in an ADC block as general purpose I/O (GPIO) pins if you do not use the ADC.
- For more information about the device part numbers that feature ADC blocks, refer to the device overview.

Package	Pin Type	ADC Channel Counts Per Device					
		10M04	10M08	10M16	10M25	10M40	10M50
M153	Dedicated	1	1	—	—	—	—
	Dual function	8	8	—	—	—	—
U169	Dedicated	1	1	1	—	—	—
	Dual function	8	8	8	—	—	—
U324	Dedicated	1	1	1	—	—	—
	Dual function	16	16	16	—	—	—
F256	Dedicated	1	1	1	2	2	2
	Dual function	16	16	16	16	16	16
E144	Dedicated	1	1	1	1	1	1
	Dual function	8	8	8	8	8	8
F484	Dedicated	—	1	1	2	2	2
	Dual function	—	16	16	16	16	16
F672	Dedicated	—	—	—	—	2	2
	Dual function	—	—	—	—	16	16

Related Information

- **MAX 10 FPGA Device Overview**
- **MAX 10 ADC Vertical Migration Support** on page 1-4

MAX 10 ADC Vertical Migration Support

Figure 1-1: ADC Vertical Migration Across MAX 10 Devices

The arrows indicate the ADC migration paths. The devices included in each vertical migration path are shaded.

Device	Package						
	M153	U169	U324	F256	E144	F484	F672
10M04	↕	↕	↕	↕	↕		
10M08							↕
10M16							
10M25							
10M40							↕
10M50				↕	↕	↕	

■ **Dual ADC Device:** Each ADC (ADC1 and ADC2) supports 1 dedicated analog input pin and 8 dual function pins.

■ **Single ADC Device:** Single ADC that supports 1 dedicated analog input pin and 16 dual function pins.

■ **Single ADC Device:** Single ADC that supports 1 dedicated analog input pin and 8 dual function pins.

Table 1-3: Pin Migration Conditions for ADC Migration

Source	Target	Migratable Pins
Single ADC device	Single ADC device	You can migrate all ADC input pins
Dual ADC device	Dual ADC device	
Single ADC device	Dual ADC device	• One dedicated analog input pin.
Dual ADC device	Single ADC device	• Eight dual function pins from the ADC1 block of the source device to the ADC1 block of the target device.

Related Information

ADC Channel Counts in MAX 10 Devices on page 1-3

MAX 10 Single or Dual Supply Devices

MAX 10 devices are available in single or dual supply packages.

- For devices with single power supply:
  - Use on chip regulator to power up the digital supply.
  - Use V_{CCA} to power up the ADC analog.
- For dual power supply devices, you must provide external power supplies of 1.2 V and 2.5 V to power up the ADC.

To choose the correct device, refer to the MAX 10 device overview.

For more information about the ADC parameter, refer to the device datasheet.

Related Information
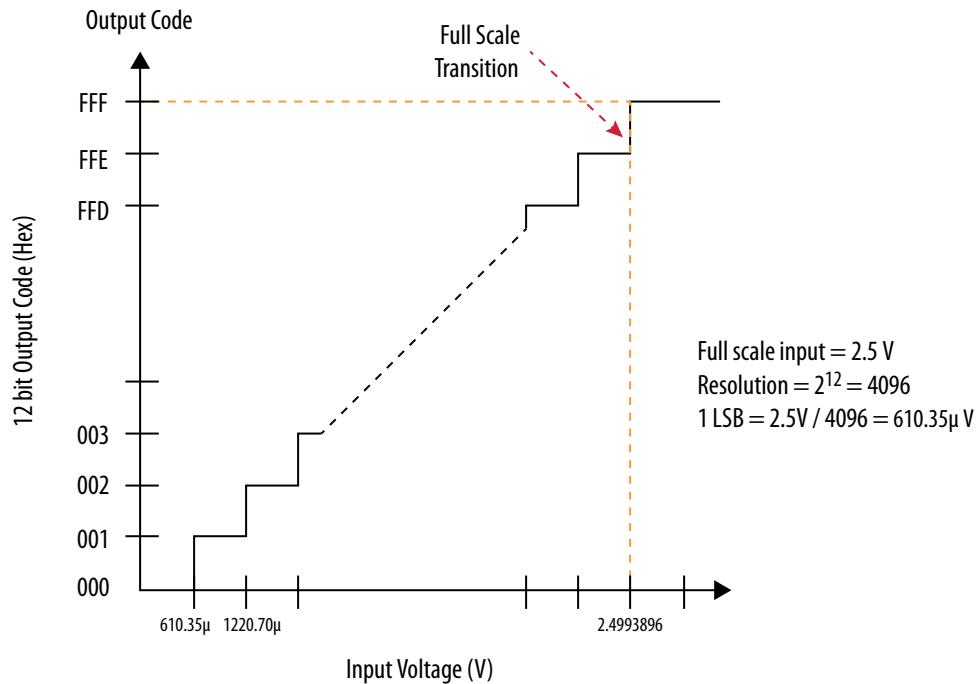
- **MAX 10 Device Datasheet**
- **MAX 10 FPGA Device Overview**

MAX 10 ADC Conversion

The ADC in dual supply MAX 10 devices can measure from 0 V to 2.5 V. In single supply MAX 10 devices, it can measure up to 3.0 V or 3.3 V, depending on your power supply voltage.

- In prescaler mode, the analog input can measure up to 3.0 V in dual supply MAX 10 devices and up to 3.6 V in single supply MAX 10 devices.
- The analog input scale has full scale code from `000h` to `FFFh`. However, the measurement can only display up to *full scale – 1 LSB*.
- For the 12 bits corresponding value calculation, use unipolar straight binary coding scheme.

Figure 1-2: ADC Measurement Display for 2.5 V

The MAX 10 ADC is a 1 MHz successive approximation register (SAR) ADC. If you set up the PLL and Altera Modular ADC IP core correctly, the ADC operates at up to 1 MHz during normal sampling and 50 kHz during temperature sensing.

Note: The analog value represented by the all-ones code is not full scale but *full scale – 1 LSB*. This is a common convention in data conversion notation and applies to ADCs.

Related Information

- **Creating MAX 10 ADC Design** on page 4-2
- **Altera Modular ADC Parameters Settings** on page 5-2
- **Altera Modular Dual ADC Parameters Settings** on page 5-8

In MAX 10 devices, the ADC is a 12 bits SAR ADC that provides the following features:

- Sampling rate of up to 1 MSPS
- Up to 18 channels for analog measurement: 16 dual function channels and two dedicated analog input channels in dual ADC devices
- Single-ended measurement capability
- Simultaneous measurement capability at the dedicated analog input pins for dual ADC devices
- Soft logic sequencer
- On-chip temperature sensor with sampling rate of 50 kilosamples per second
- Internal or external voltage references usage. The source of the internal voltage reference is the ADC analog supply; the ADC conversion result is ratiometric.

Related Information

- **MAX 10 Analog to Digital Converter Overview** on page 1-1
- **MAX 10 Analog to Digital Converter User Guide Archives** on page 6-1
  Provides a list of user guides for previous versions of the Altera Modular ADC and Altera Modular Dual ADC IP cores.

MAX 10 ADC Hard IP Block

The MAX 10 ADC is a successive approximation register (SAR) ADC that converts one analog sample in one clock cycle.

Each ADC block supports one dedicated analog input pin and up to 16 channels of dual function pins.
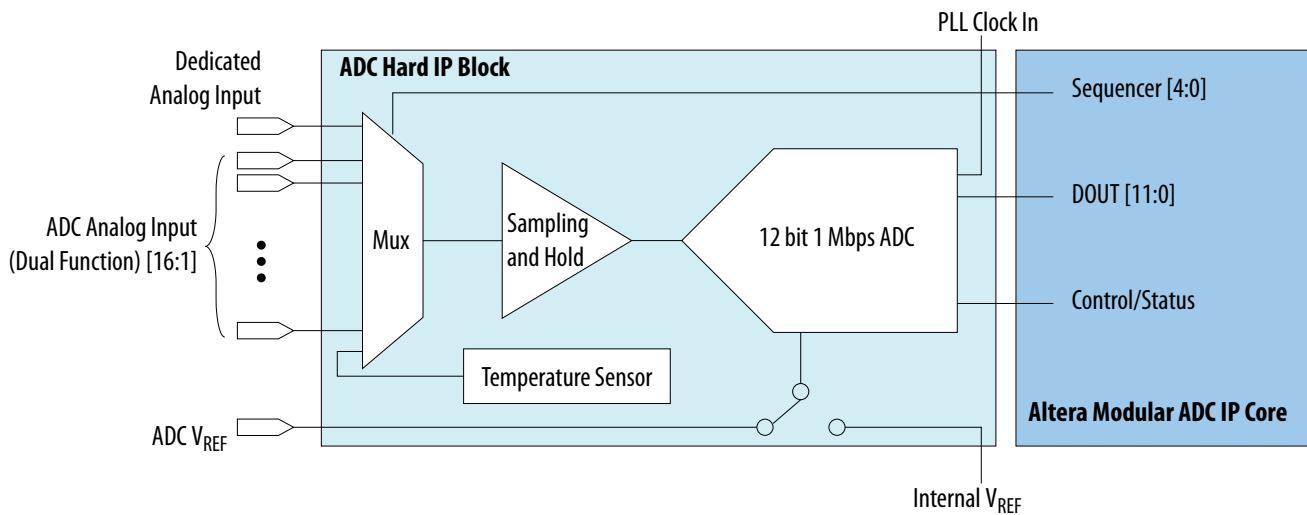
You can use the built-in temperature sensing diode (TSD) to perform on-chip temperature measurement.

now part of Intel

Figure 2-1: ADC Hard IP Block in MAX 10 Devices

Note: In dual ADC devices, the temperature sensor is available only in ADC1.

ADC Block Locations

The ADC blocks are located at the top left corner of the MAX 10 device periphery.
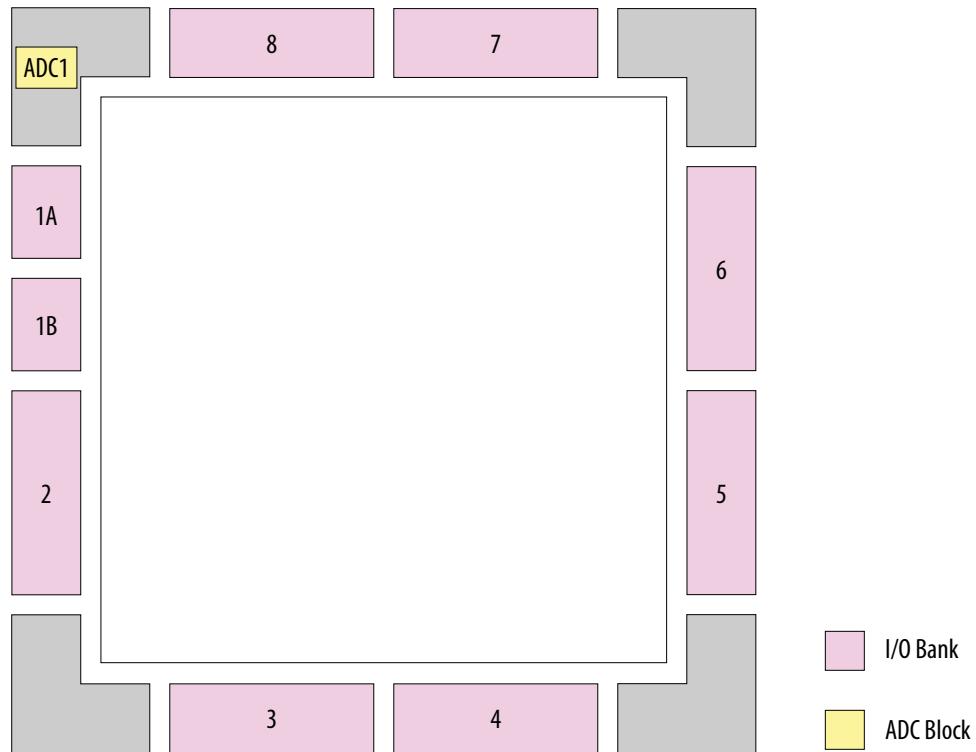
Figure 2-2: ADC Block Location in MAX 10 04 and 08 Devices

Figure 2-3: ADC Block Location in MAX 10 16 Devices
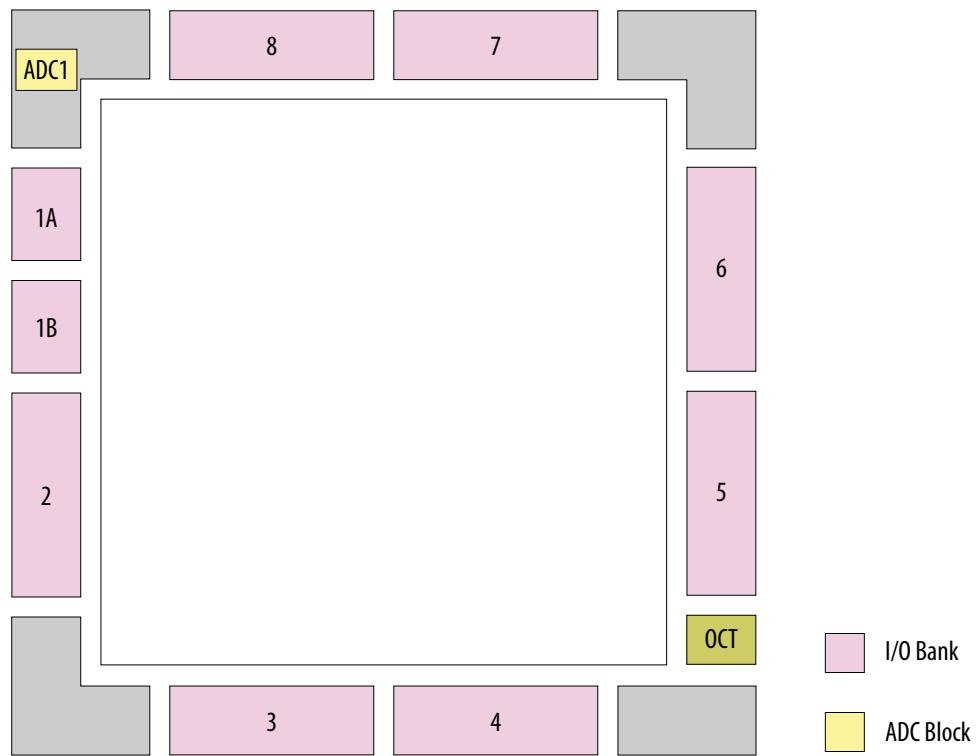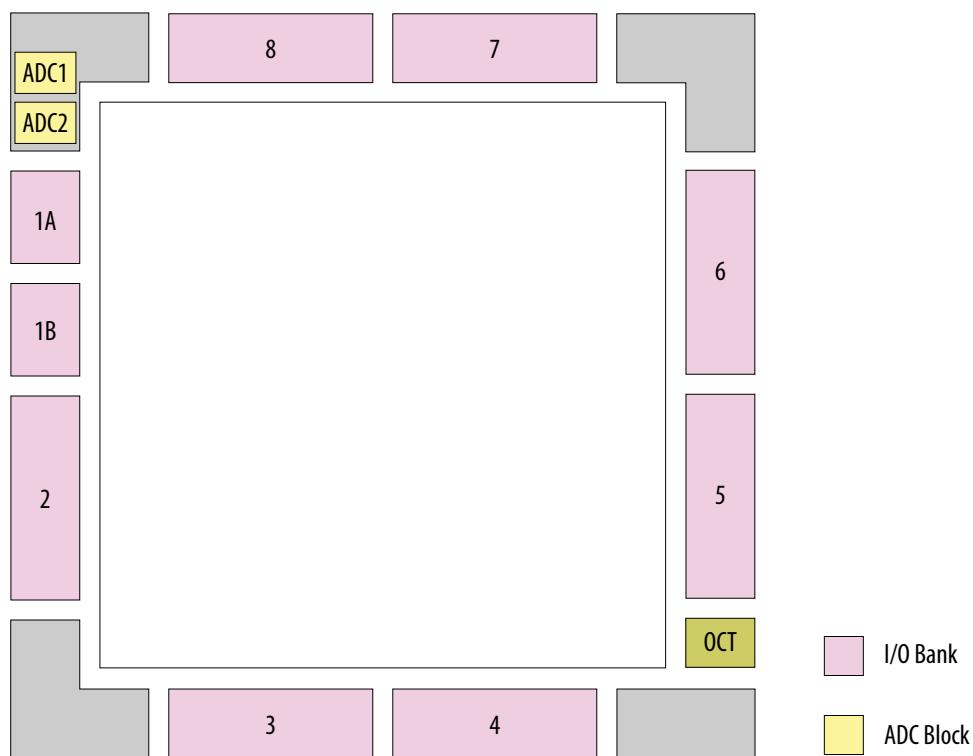
Figure 2-4: ADC Block Location in MAX 10 25, 40, and 50 Devices

Package E144 of these devices have only one ADC block.

Single or Dual ADC Devices

MAX 10 devices are available with single or dual ADC blocks.

For devices with one ADC block, you can use up to 17 ADC channels:

- These channels include one dedicated analog input and up to 16 dual function pins.
- You can use the dual function pins as GPIO pins when you do not use the ADC.

Note: MAX 10 devices in the E144 package have only 8 dual function ADC pins.

For devices with two ADC blocks, you can use up to 18 ADC channels:

- For dual ADC devices, each ADC block can support one dedicated analog input pin and up to 8 dual function pins.
- If you use both ADC blocks in dual ADC devices, you can use up to two dedicated analog input pins and 16 dual function pins.
- For simultaneous measurement, you can use only dedicated analog input pins in both ADC blocks because the package routing of both dedicated analog pins are matched. For dual function pins, the routing latency between two ADC blocks may cause data mismatch in simultaneous measurement.
- For simultaneous measurement, use the Altera Modular Dual ADC IP core.

To choose the correct device, refer to the MAX 10 device overview.

Related Information

- **MAX 10 FPGA Device Overview**

ADC Analog Input Pins

The analog input pins support single-ended and unipolar measurements.

The ADC block in MAX 10 devices contains two types of ADC analog input pins:

- Dedicated ADC analog input pin—pins with dedicated routing that ensures both dedicated analog input pins in a dual ADC device has the same trace length.
- Dual function ADC analog input pin—pins that share the pad with GPIO pins.

If you use bank 1A for ADC, you cannot use the bank for GPIO.
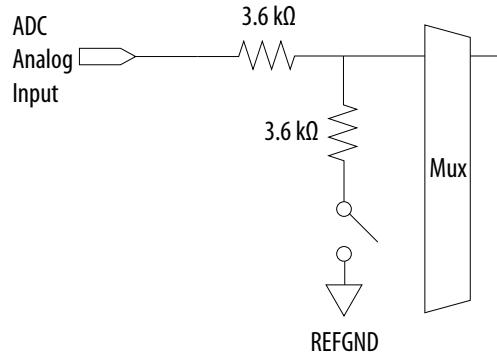
Each analog input pin in the ADC block is protected by electrostatic discharge (ESD) cell.

ADC Prescaler

The ADC block in MAX 10 devices contains a prescaler function.

The prescaler function divides the analog input voltage by half. Using this function, you can measure analog input greater than 2.5 V. In prescaler mode, the analog input can handle up to 3 V input for the dual supply MAX 10 devices and 3.6 V for the single supply MAX 10 devices.

Figure 2-5: ADC Prescaler Block Diagram

The prescaler feature is available on these channels in each ADC block:

- Single ADC device—channels 8 and 16 (if available)
- Dual ADC device:
  - Using Altera Modular ADC IP core—channel 8 of first or second ADC
  - Using Altera Modular Dual ADC IP core—channel 8 of ADC1 and channel 17 of ADC2

ADC Clock Sources

The ADC block uses the device PLL as the clock source. The ADC clock path is a dedicated clock path. You cannot change this clock path.

Depending on the device package, the MAX 10 devices support one or two PLLs—PLL1 only, or PLL1 and PLL3.

For devices that support two PLLs, you can select which PLL to connect to the ADC. You can configure the ADC blocks with one of the following schemes:

- Both ADC blocks share the same clock source for synchronization.
- Both ADC blocks use different PLLs for redundancy.

If each ADC block in your design uses its own PLL, the Quartus® Prime Fitter automatically selects the clock source scheme based on the PLL clock input source:

- If each PLL that clocks its respective ADC block uses different PLL input clock source, the Quartus Prime Fitter follows your design (two PLLs).
- If both PLLs that clock their respective ADC block uses the same PLL input clock source, the Quartus Prime Fitter merges both PLLs as one.

In dual ADC mode, both ADC instance must share the same ADC clock setting.

ADC Voltage Reference

Each ADC block in MAX 10 devices can independently use an internal or external voltage reference. In dual ADC devices, you can assign an internal voltage reference to one ADC block and an external voltage reference to the other ADC block.

There is only one external VREF pin in each MAX 10 device. Therefore, if you want to assign external voltage reference for both ADC blocks in dual ADC devices, share the same external voltage reference for both ADC blocks.

Altera recommends that you use external voltage reference for the ADC blocks. If the ADC block uses an internal voltage reference, the ADC block is tied to its analog voltage and the conversion result is ratiometric.

ADC Temperature Sensing Diode

The ADC block in MAX 10 devices has built-in TSD. You can use the built-in TSD to monitor the internal temperature of the MAX 10 device.

- While using the temperature sensing mode, the ADC sampling rate is 50 kilosamples per second during temperature measurement.
- After the temperature measurement completes, if the next conversion in the sequence is normal sampling mode, the Altera Modular ADC IP core automatically switches the ADC back to normal sampling mode. The maximum cumulative sampling rate in normal sampling mode is 1 MSPS.
- When the ADC switches from normal sensing mode to temperature sensing mode, and vice versa, calibration is run automatically for the changed clock frequency. The calibration incurs at least six clock calibration cycles from the new sampling rate.
- The ADC TSD measurement uses a 64-samples running average method. For example:

  - The first measured temperature value is the average of samples 1 to 64.
  - The second measured temperature value is the average of samples 2 to 65.
  - The third measured temperature value is the average of samples 3 to 66.
  - The subsequent temperature measurements follow the same method.

For dual ADC devices, the temperature sensor is available in ADC1 only.

Temperature Measurement Code Conversion

Use the temperature measurement code conversion table to convert the values measured by the ADC TSD to actual temperature.

Table 2-1: Temperature Code Conversion Table

Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code
-40	3798	-6	3738	28	3670	62	3593	96	3510
-39	3796	-5	3736	29	3667	63	3592	97	3507
-38	3795	-4	3733	30	3666	64	3591	98	3504
-37	3793	-3	3732	31	3664	65	3590	99	3501
-36	3792	-2	3731	32	3662	66	3589	100	3500
-35	3790	-1	3730	33	3660	67	3585	101	3498
-34	3788	0	3727	34	3658	68	3582	102	3496
-33	3786	1	3725	35	3656	69	3579	103	3494
-32	3785	2	3721	36	3654	70	3576	104	3492
-31	3782	3	3720	37	3651	71	3573	105	3490
-30	3781	4	3719	38	3648	72	3570	106	3489
-29	3780	5	3717	39	3645	73	3567	107	3486
-28	3779	6	3715	40	3643	74	3564	108	3483
-27	3777	7	3713	41	3642	75	3561	109	3480
-26	3775	8	3711	42	3641	76	3558	110	3477
-25	3773	9	3709	43	3640	77	3555	111	3474
-24	3771	10	3707	44	3638	78	3552	112	3471
-23	3770	11	3704	45	3636	79	3551	113	3468
-22	3768	12	3703	46	3634	80	3550	114	3465
-21	3766	13	3702	47	3632	81	3549	115	3461
-20	3765	14	3700	48	3630	82	3548	116	3460
-19	3764	15	3699	49	3628	83	3547	117	3459
-18	3762	16	3698	50	3625	84	3546	118	3456
-17	3759	17	3697	51	3622	85	3542	119	3451
-16	3756	18	3696	52	3619	86	3538	120	3450
-15	3754	19	3695	53	3616	87	3534	121	3449
-14	3752	20	3688	54	3613	88	3530	122	3445
-13	3751	21	3684	55	3610	89	3526	123	3440
-12	3750	22	3682	56	3607	90	3525	124	3432
-11	3748	23	3680	57	3604	91	3524	125	3431

Send Feedback

Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code
-10	3746	24	3678	58	3601	92	3522	—	—
-9	3744	25	3677	59	3598	93	3519	—	—
-8	3742	26	3676	60	3595	94	3516	—	—
-7	3740	27	3673	61	3594	95	3513	—	—

ADC Sequencer

The Altera Modular ADC and Altera Modular Dual ADC IP cores implement the sequencer. Use the Altera Modular ADC or Altera Modular Dual ADC parameter editor to define the ADC channel acquisition sequence and generate the HDL code.

The sequencer can support sequences of up to 64 ADC measurement slots. While configuring the Altera Modular ADC or Altera Modular Dual ADC IP core, you can select which channel, including the TSD channel, to sample in each sequencer slot. During runtime, you cannot change the channel sequence but you can configure the conversion mode using the Nios® II HAL driver API.

You can specify up to 64 slots and assign the channel for each slot. You can repeat the same channel number several times if required.

Related Information

Guidelines: ADC Sequencer in Altera Modular Dual ADC IP Core on page 2-8

Guidelines: ADC Sequencer in Altera Modular Dual ADC IP Core

Follow these sequencer guidelines if you use dual ADC blocks with the Altera Modular Dual ADC IP core.

- The conversion sequence length of both ADC blocks must be the same.
- You can configure independent patterns for the conversion sequence of each ADC blocks.
- You can set a sequencer slot in ADC2 to NULL. If you set the slot to NULL, ADC2 will perform a dummy conversion for the slot with output of "0". The NULL option is available only for ADC2.
- The temperature sensor is available only in ADC1. If you configure a sequencer slot in ADC1 for temperature sensing, you must set the same sequencer slot number in ADC2 to NULL.
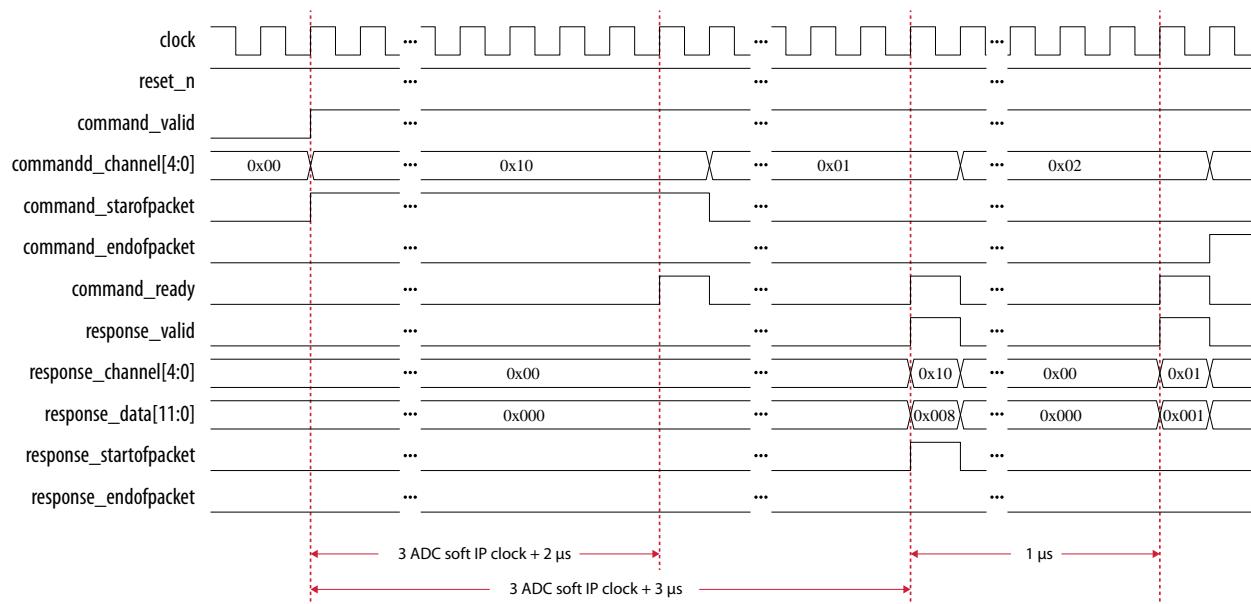
Related Information

ADC Sequencer on page 2-8

ADC Timing

Figure 2-6: MAX 10 ADC Timing Diagram

- This figure shows the timing diagram for the command and response interface of the Altera Modular ADC control core.
- The timing diagram shows the latency of the first valid response data, and the latency between the first acknowledgment of the first command request and the back-to-back response data.

The timing diagram shows an example where:

- The conversion sequence is channel 16 → channel 1 → channel 2
- The response data for channel 16 is 8
- The response data for channel 1 is 1

Altera Modular ADC and Altera Modular Dual ADC IP Cores

You can use the Altera Modular ADC and Altera Modular Dual ADC IP cores to generate soft IP controllers for the ADC hard IP blocks in MAX 10 devices.

There are two ADC IP cores:

- Altera Modular ADC IP core—each instance can control one ADC hard IP block. In a dual ADC device, you can instantiate one Altera Modular ADC IP core instance for each ADC block. However, both instances will be asynchronous to each other.
- Altera Modular Dual ADC IP core—you can control both ADC hard IP block with a single IP instance.

  - For the analog input pins (ANAIN1 and ANAIN2) in both ADC hard IP blocks, the measurement is synchronous.
  - For the dual function input pins, there are some measurement timing differences caused by the routing latency.

You can perform the following functions with the Altera Modular ADC or Altera Modular Dual ADC IP core parameter editor:

- Configure the ADC clock, sampling rate, and reference voltage.
- Select which analog input channels that the ADC block samples.
- Configure the threshold value to trigger a threshold violation notification.
- Set up a conversion sequence to determine which channel requires more frequent attention.

Related Information

- **Altera Modular ADC and Altera Modular Dual ADC IP Cores References** on page 5-1
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Altera Modular ADC IP Core Configuration Variants

The Altera Modular ADC IP core provides four configuration variants that target different ADC use cases. These configuration variants support usages from basic system monitoring to high performance ADC data streaming.

Configuration 1: Standard Sequencer with Avalon-MM Sample Storage on page 2-10
In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples.

Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection on page 2-11
In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples with the additional capability of detecting threshold violation.

Configuration 3: Standard Sequencer with External Sample Storage on page 2-13
In this configuration variant, you can use the standard sequencer micro core and store the ADC samples in external storage.

Configuration 4: ADC Control Core Only on page 2-13
In this configuration variant, the Altera Modular ADC generates only the ADC control core.
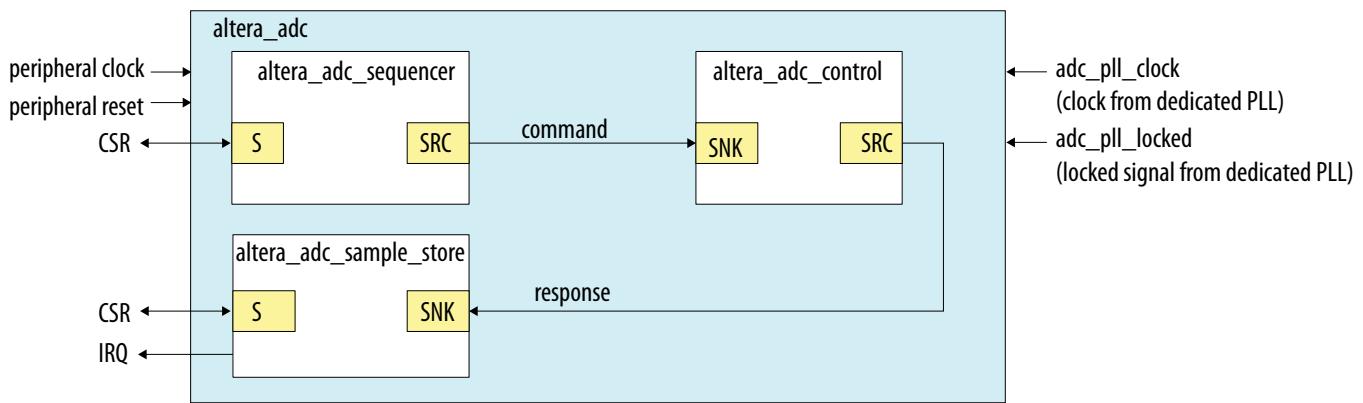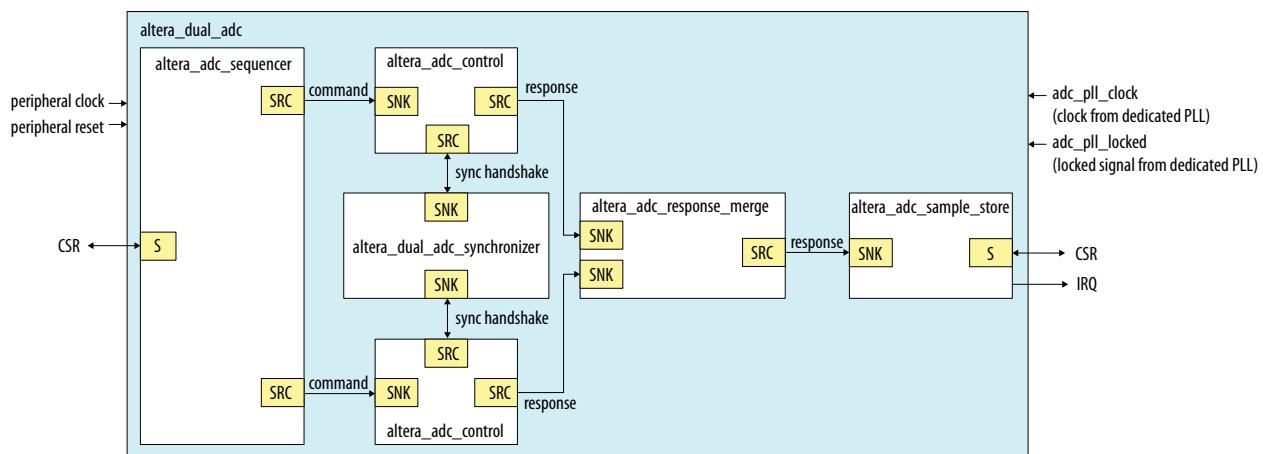
Related Information
Altera Modular ADC and Altera Modular Dual ADC IP Cores References on page 5-1

Configuration 1: Standard Sequencer with Avalon-MM Sample Storage

In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples. This configuration is useful for basic system monitoring application.

In a system monitoring application, the ADC captures a block of samples data and stores them in the on-chip RAM. The host processor retrieves the data before triggering another block of ADC data sample request. The speed of the host processor in servicing the interrupt determines the interval between each block sample request.

UG-M10ADC
2016.05.02

Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and...

2-11

Figure 2-7: Standard Sequencer with Avalon-MM Sample Storage (Altera Modular ADC IP Core)

Figure 2-8: Standard Sequencer with Avalon-MM Sample Storage (Altera Modular Dual ADC IP Core)

Related Information
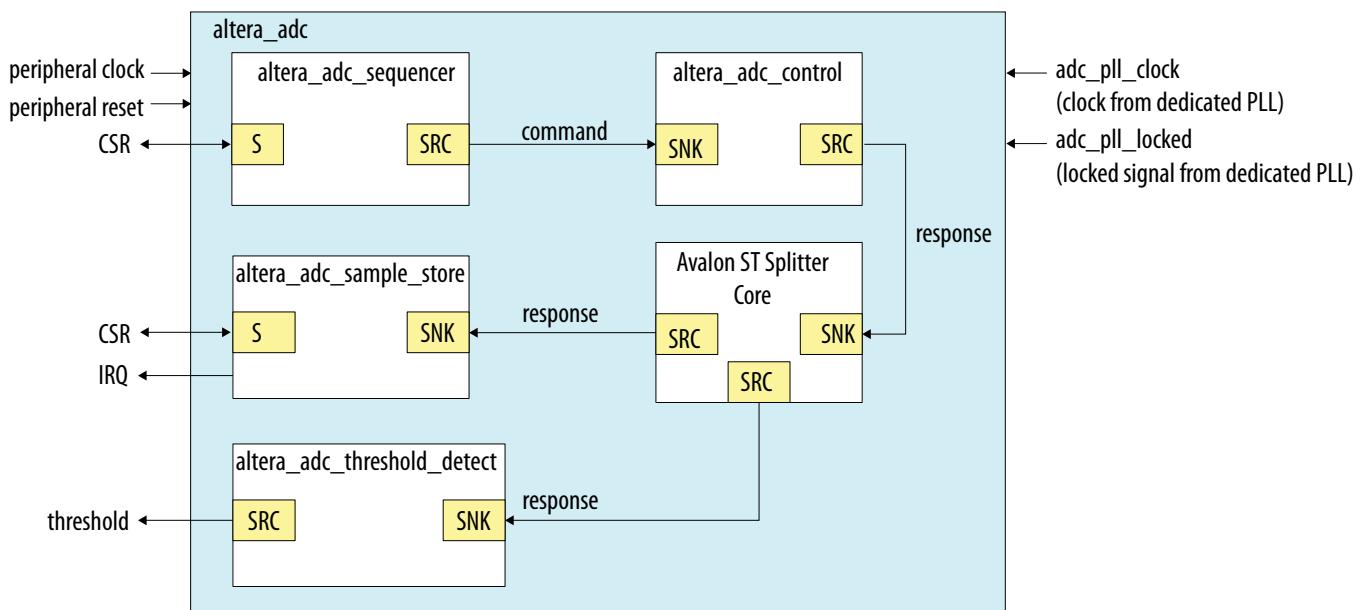
- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Completing ADC Design** on page 4-8

Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection

In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples with the additional capability of detecting threshold violation. This configuration is useful for system monitoring application where you want to know whether the ADC samples value fall outside the maximum or minimum threshold value.
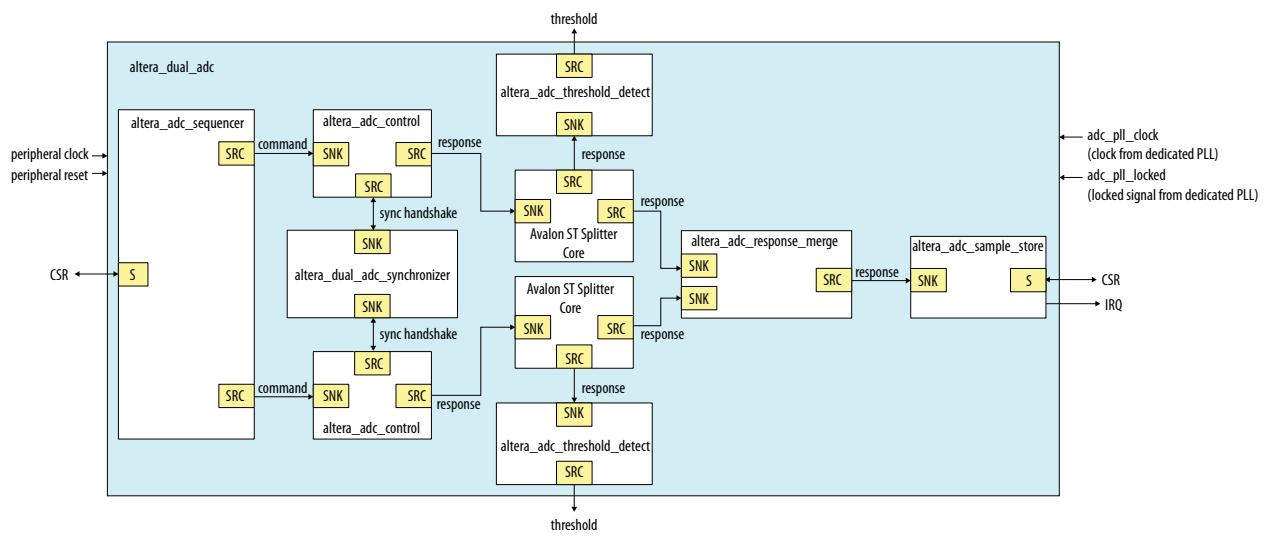
When the threshold value is violated, the Altera Modular ADC or Altera Modular Dual ADC IP core notifies the discrete logic component. The discrete component then triggers system recovery action. For example, the system can increase the fan speed in a temperature control system.

2-12 Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and...

UG-M10ADC
2016.05.02

Figure 2-9: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection (Altera Modular ADC IP Core)

In dual ADC mode, you can configure the threshold detection of each ADC instance independently of each other. This capability is available because each ADC instance measures different analog metrics.

Figure 2-10: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection (Altera Modular Dual ADC IP Core)


Related Information

- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Completing ADC Design** on page 4-8

Configuration 3: Standard Sequencer with External Sample Storage

In this configuration variant, you can use the standard sequencer micro core and store the ADC samples in external storage.

You need to design your own logic to interface with the external storage.

Figure 2-11: Standard Sequencer with External Sample Storage (Altera Modular ADC IP Core)
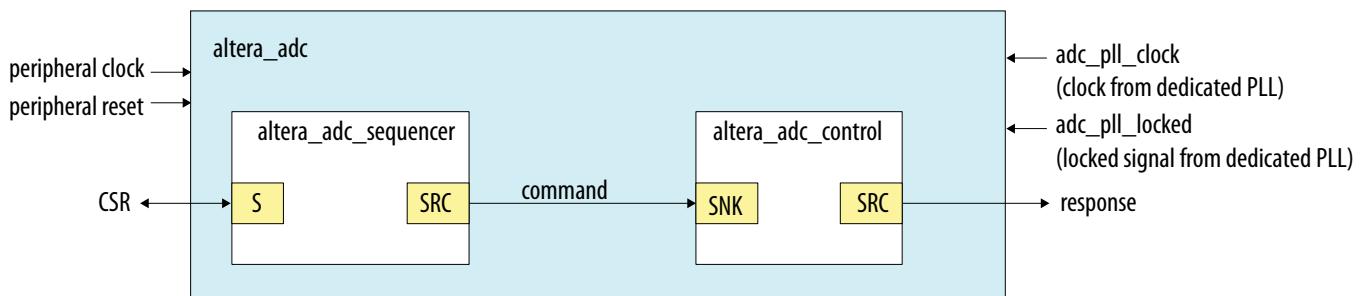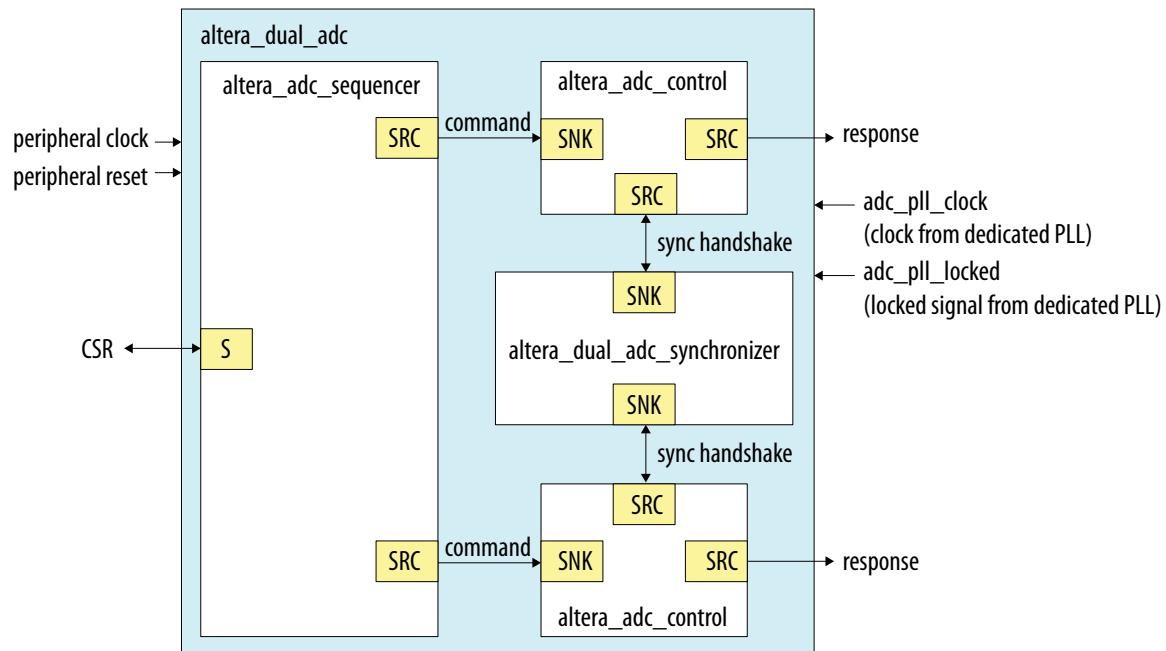
Figure 2-12: Standard Sequencer with External Sample Storage (Altera Modular Dual ADC IP Core)

Related Information

- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Completing ADC Design** on page 4-8

Configuration 4: ADC Control Core Only

In this configuration variant, the Altera Modular ADC generates only the ADC control core. You have full flexibility to design your own application-specific sequencer and use your own way to manage the ADC samples.

Figure 2-13: ADC Control Core Only (Altera Modular ADC IP Core)
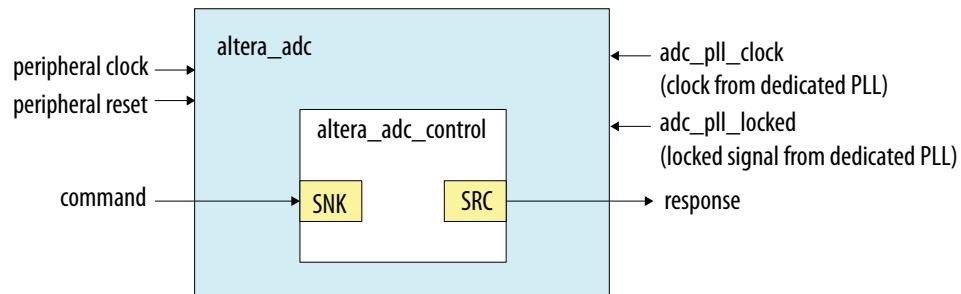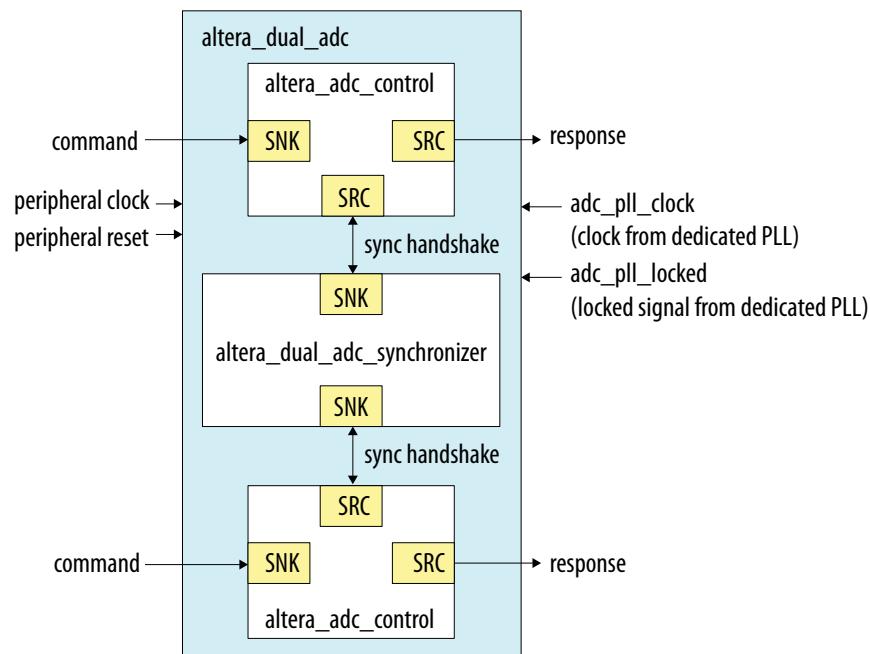
**Figure 2-14: ADC Control Core Only (Altera Modular Dual ADC IP Core)**

Related Information

- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Completing ADC Design** on page 4-8

Altera Modular ADC and Altera Modular Dual ADC IP Cores Architecture

The Altera Modular ADC IP core consists of six micro cores.

Table 2-2: Altera Modular ADC Micro Cores

Micro Core	Description
ADC control	This core interacts with the ADC hard IP block. The ADC control core uses Avalon ST interface to receive commands from upstream cores, decodes, and drives the ADC hard IP block accordingly.
Sequencer	This core contains command register and static conversion sequence data. The sequencer core issues commands for downstream cores to execute. • You can use the command register to configure the intended conversion mode. • You can configure the length and content of the conversion sequence data only when generating the IP core. • You can access the register of the sequencer core through the Avalon-MM slave interface. • The command information to the downstream core goes through the Avalon ST interface.
Sample storage	This core stores the ADC samples that are received through the Avalon ST interface. • The samples are stored in the on-chip RAM. You can retrieve the samples through the Avalon-MM slave interface. • With this core, you have the option to generate interrupt when the ADC receives a block of ADC samples (one full round of conversion sequence).
Response merge	This core merges simultaneous responses from two ADC control cores into a single response packet to send to the sample storage core. This core is available only if you use the Altera Modular Dual ADC IP core in the following configurations: • Standard Sequencer with Avalon-MM Sample Storage • Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection
Dual ADC synchronizer core	This core performs synchronization handshakes between two ADC control cores. This core is available only if you use the Altera Modular Dual ADC IP core.

Micro Core	Description
Threshold detection	• This core supports fault detection. The threshold detection core receives ADC samples through the Avalon ST interface and checks whether the samples value exceeds the maximum or falls below the minimum threshold value. • The threshold detection core conveys threshold value violation information through the Avalon ST interface. • You can configure which channel to enable for maximum and minimum threshold detection and the threshold values only during IP core generation.

ADC Control Core

The ADC control core drives the ADC hard IP according to the command it receives. The control core also maps the channels from the Altera Modular ADC IP core to the channels in the ADC hard IP block.

The ADC control core of the Altera Modular ADC IP core implements only the functions that are related to ADC hard IP block operations. For example:

• Power up
• Power down
• Analog to digital conversion on analog pins
• Analog to digital conversion on on-chip temperature sensor

The ADC control core has two clock domains:

• One clock domain for clocking the ADC control core soft logic
• Another clock domain for the ADC hard IP block

The ADC control core does not have run-time configurable options.

**Figure 2-15: ADC Control Core High-Level Block Diagram**
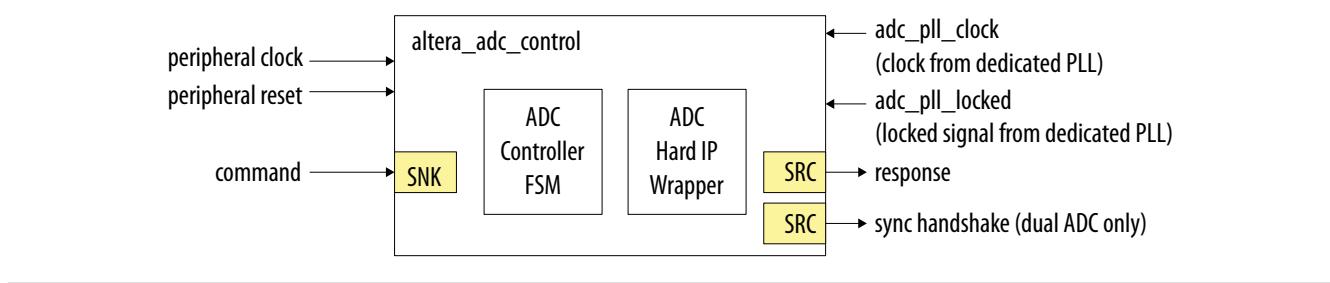
Table 2-3: ADC Control Core Backpressure Behavior
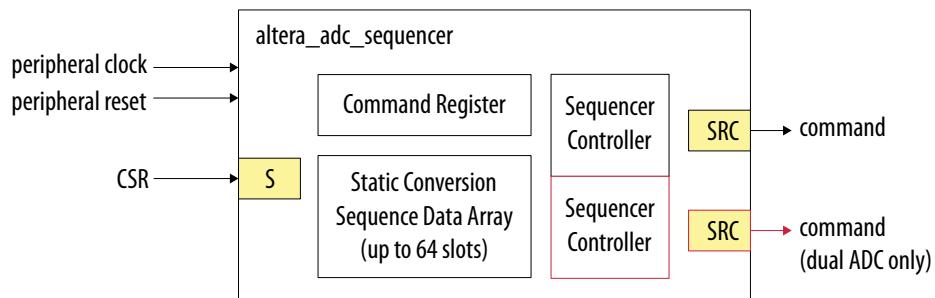
Interface	Backpressure Behavior
Command	The ADC control core asserts ready when it is ready to perform a sample conversion.
	The ADC control core only accepts one command at a time. The control core releases ready when it completes processing current command and prepares to perform the next command.
	Once the ADC control core asserts "cmd_ready=1" to acknowledge the current command, the Sequencer core provides the next valid request within two clock cycles. If the next valid request comes after two clock cycles, the ADC control core perform non-continuous sampling.
Response	The ADC control core does not support backpressure in the response interface. The fastest back-to-back assertion of valid request is 1 μs.

Sequencer Core

The sequencer core controls the type of conversion sequence performed by the ADC hard IP. You can configure the conversion mode during run time using the sequencer core registers.

During Altera Modular ADC or Altera Modular Dual ADC IP core configuration, the sequencer core provides up to 64 configurable slots. You can define the sequence that the ADC channels are sampled by selecting the ADC channel for each sequencer slot.

The sequencer core has a single clock domain.

Figure 2-16: Sequencer Core High-Level Block Diagram

Table 2-4: Sequencer Core Conversion Modes

Conversion Mode	Description
Single cycle ADC conversion	• In this mode, when the run bit is set, ADC conversion starts from the channel that you specify in the first slot. • The conversion continues onwards with the channel that you specify in each sequencer slot. • Once the conversion finishes with the last sequencer slot, the conversion cycle stops and the ADC hard IP block clears the run bit.

Conversion Mode	Description
Continuous ADC conversion	• In this mode, when the run bit is set, ADC conversion starts from the channel that you specify in the first slot. • The conversion continues onwards with the channel that you specify in each sequencer slot. • Once the conversion finishes with the last sequencer slot, the conversion begins again from the first slot of the sequence. • To stop the continuous conversion, clear the run bit. The sequencer core continues the conversion sequence until it reaches the last slot and then stops the conversion cycle.

Related Information

- **Altera Modular ADC Parameters Settings** on page 5-2
  Lists the parameters available during Altera Modular ADC IP core configuration.
- **Altera Modular Dual ADC Parameters Settings** on page 5-8
  Lists the parameters available during Altera Modular Dual ADC IP core configuration.
- **Sequencer Core Registers** on page 5-18
  Lists the registers for run-time control of the sequencer core.

Sample Storage Core
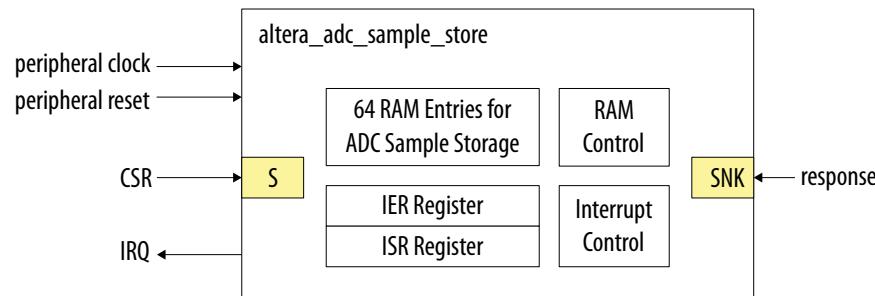
The sample storage core stores the ADC sampling data in the on-chip RAM. The sample storage core stores the ADC samples data based on conversion sequence slots instead of ADC channels.

For example, if you sample a sequence of CH1, CH2, CH1, CH3, CH1, and then CH4, the ADC sample storage core stores the channel sample data in the same RAM entry sequence. This means that CH1 sample data will be in the first, third, and fifth RAM entries; one for each sequence slot.

The sample storage core asserts IRQ when it completes receipt of a sample block. You can disable the IRQ assertion during run time using the interrupt enable register (IER) of the sample storage core. If you disable IRQ assertion, you must create polling methods in your design to determine the complete receipt of a sample block.

The sample storage core has a single clock domain.

Figure 2-17: Sample Storage Core High-Level Block Diagram

Related Information

**Sample Storage Core Registers** on page 5-19
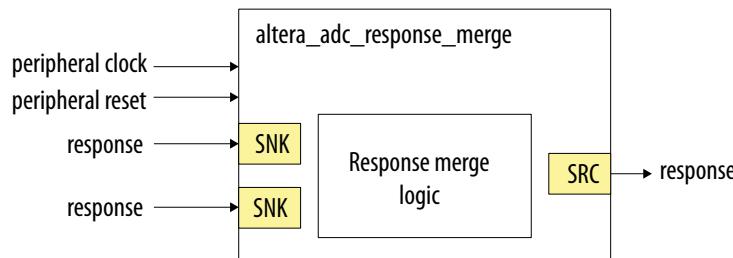
Response Merge Core

The response merge core merges simultaneous responses from two ADC control cores in the Altera Modular Dual ADC IP core.

The Altera Modular Dual ADC IP core uses the response merge core if you use the following configurations:

- Standard Sequencer with Avalon-MM Sample Storage
- Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection

Figure 2-18: Response Merge Core High-Level Block Diagram


Dual ADC Synchronizer Core
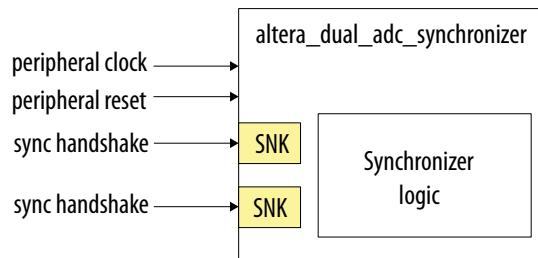
The dual ADC synchronizer core performs synchronization handshakes between two ADC control cores in the Altera Modular Dual ADC IP core.

The peripheral clock domain is asynchronous to the ADC PLL clock domain in the ADC control core. Control event from the ADC hard IP block can appear at the peripheral clock domain at the same time, or by a difference of one peripheral clock between ADC1 and ADC2 control cores. Both ADC hard IP cores communicate with the dual ADC synchronizer core through the Avalon-ST interface.

For example, although a new command valid event from the sequencer arrives at both ADC control cores at the same peripheral clock cycle, the end of conversion signals arrive at one peripheral clock cycle difference between ADC1 and ADC2. To avoid the condition where ADC1 begins conversion earlier or later than ADC2, the ADC control core performs synchronization handshake using the dual ADC synchronizer core.

An ADC control core asserts a `sync_valid` signal when it detects an ADC PLL clock domain event. The dual ADC synchronizer core asserts the `sync_ready` signal after it receives `sync_valid` signals from both ADC control cores. After the `sync_ready` signal is asserted, both ADC control cores proceed to their next internal state.

**Figure 2-19: Dual ADC Synchronizer Core High-Level Block Diagram**
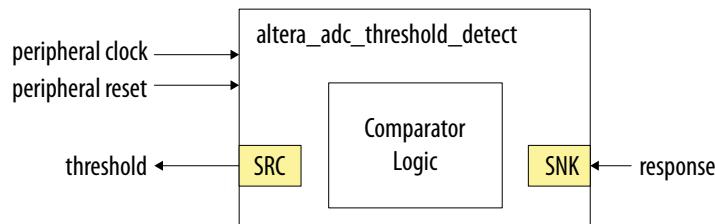
Threshold Detection Core

The threshold detection core compares the sample value that the ADC block receives to the threshold value that you define during Altera Modular ADC IP core configuration. This core does not have run-time configurable options.

If the ADC sample value is beyond the maximum or minimum threshold limit, the threshold detection core issues a violation notification through the Avalon-ST interface.

The threshold detection core has a single clock domain.

Figure 2-20: Threshold Detection Core High-Level Block Diagram

Altera ADC HAL Driver

The Altera ADC HAL driver supports the following features:

- Read ADC channel data.
- Enable maximum or minimum threshold and return a user callback when the interrupt is triggered.
- Command the control of the ADC (run, stop, and recalibrate).

ADC Toolkit for Testing ADC Performance

You can use the ADC Toolkit provided with the Quartus Prime software to understand the performance of the analog signal chain as seen by the MAX 10 ADC blocks.

The ADC Toolkit supports monitoring the ADC whether you use the Altera Modular ADC or Altera Modular Dual ADC IP core. However, the ADC Toolkit can only monitor one ADC block at a time. If you are using the Altera Modular Dual ADC IP core, configure the **Debug Path** parameter in the IP core to select which ADC block you want to hook up to the ADC Toolkit.

Related Information

ADC Toolkit
Provides more information about the ADC Toolkit.

ADC Logic Simulation Output

By default, the ADC logic simulation outputs a fixed unique value for each ADC channel. If you want to specify your own expected output for each channel, you can enable the option in the Altera Modular ADC or Altera Modular Dual ADC IP core.

The ADC simulation model for MAX 10 devices supports the standard digital logic simulators that the Quartus Prime software supports.

Related Information

Quartus Prime Simulator Support

Fixed ADC Logic Simulation Output

In the default option (**Disabled**), the ADC simulation always output a fixed value for each ADC channel, including the analog and TSD channels. The values are different for single and dual ADC devices.

Table 2-5: Fixed Expected Output Data for Single ADC Device Simulation

Channel	Expected Output Data (Decimal Value)
CH0	0
CH1	1
CH2	2
CH3	3
CH4	4
CH5	5
CH6	6
CH7	7
CH8	8
CH9	9
CH10	10
CH11	11
CH12	12
CH13	13
CH14	14
CH15	15
CH16	16
TSD	3615

Table 2-6: Fixed Expected Output Data for Dual ADC Device Simulation

Channel	Expected Output Data (Decimal Value)	
	ADC1	ADC2
CH0	10	20
CH1	11	21

Channel	Expected Output Data (Decimal Value)	
	ADC1	ADC2
CH2	12	22
CH3	13	23
CH4	14	24
CH5	15	25
CH6	16	26
CH7	17	27
CH8	18	28
TSD	3615	— (No TSD in ADC2)

User-Specified ADC Logic Simulation Output

You can configure the Altera Modular ADC or Altera Modular Dual ADC IP core to use expected output files that you provide for the logic simulation output for each ADC channel. The user-specified expected output file is not supported on the TSD channel.

If you enable this option in the IP core, you must specify a text file for each ADC channel that you enable in the IP core. The simulation model reads the text file for each channel and outputs the value of the current sequence in the text file. Once the model reaches the end of the text file, it loops back to the beginning of the sequence.

The text file contains two columns of numbers consistent with industry simulation text file format. The first column represents time and the second column represents the expected output values in voltage. The ADC simulation model does not use the time value but uses the text rows for the data sequence. The ADC IP core automatically converts the voltage values in the second column to 12 bit digital value based on the reference voltage you specify in the IP core parameter settings.

**Figure 2-21: Simulation Output Example, One Channel Enabled**
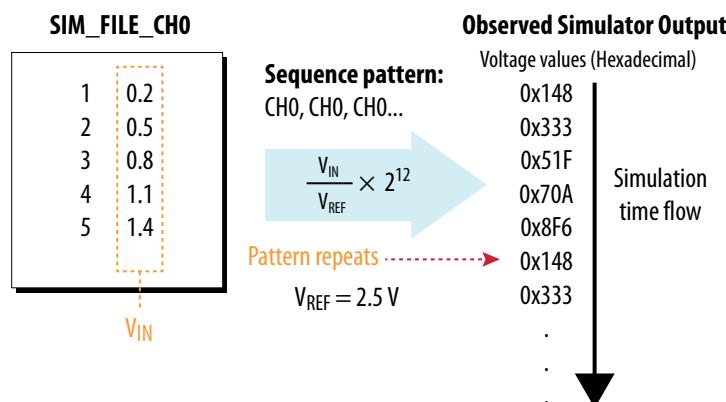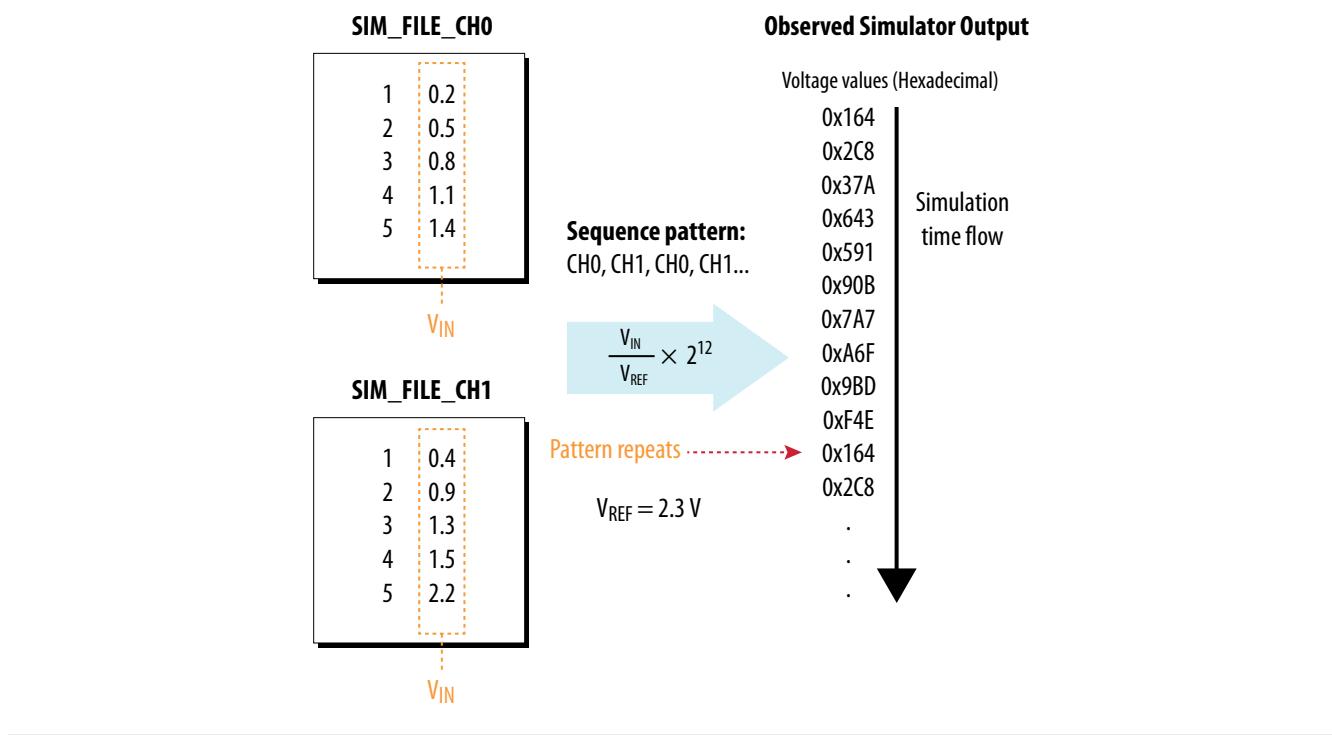
Figure 2-22: Simulation Output Example, Two Channels Enabled

SIM_FILE_CH0

1	0.2
2	0.5
3	0.8
4	1.1
5	1.4

V_{IN}

Sequence pattern:
CH0, CH1, CH0, CH1...

SIM_FILE_CH1

1	0.4
2	0.9
3	1.3
4	1.5
5	2.2

V_{IN}

$$\frac{V_{IN}}{V_{REF}} \times 2^{12}$$

Pattern repeats

$V_{REF} = 2.3\ V$

Observed Simulator Output

Voltage values (Hexadecimal)

0x164
0x2C8
0x37A
0x643
0x591
0x90B
0x7A7
0xA6F
0x9BD
0xF4E
0x164
0x2C8
.
.
.

Simulation
time flow

UG-M10ADC ✉ Subscribe 💬 **Send Feedback**

There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Related Information

MAX 10 Analog to Digital Converter Overview on page 1-1

Guidelines: ADC Ground Plane Connection

For power pins, use the GND pin. For the ADC and V_{REF} pins, use the REFGND pin.

Related Information

MAX 10 FPGA Device Family Pin Connection Guidelines
Provides more information about pin connections including pin names and connection guidelines.
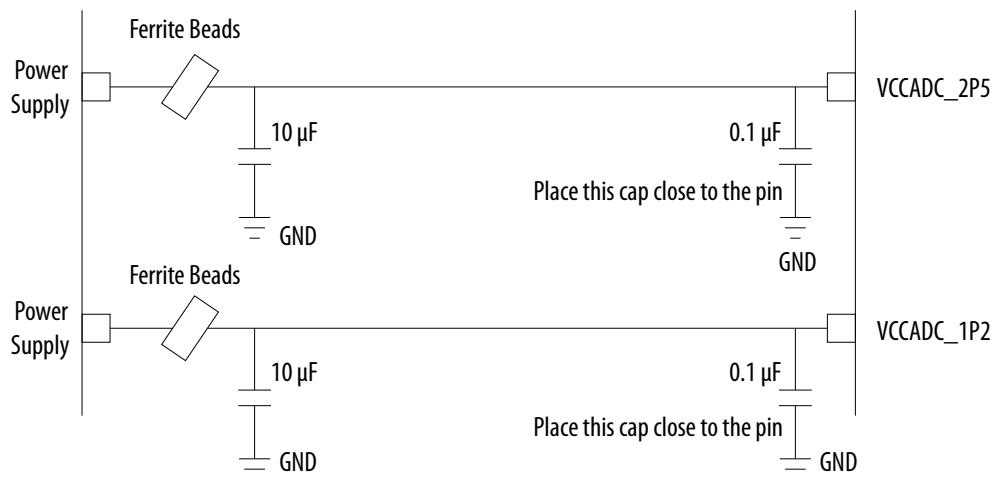
Guidelines: Board Design for Power Supply Pin and ADC Ground (REFGND)

The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. There must be no parallel routing between power, ground, and I/O traces. If a power plane is not possible, route the power and ground traces as wide as possible.

- To reduce IR drop and switching noise, keep the impedance as low as possible for the ADC power and ground. The maximum DC resistance for power is 1.5 Ω.
- The power supplies connected to the ADC should have ferrite beads in series followed by a 10 µF capacitor to the ground. This setup ensures that no external noise goes into the device power supply pins.
- Decouple each of the device power supply pin with a 0.1 µF capacitor. Place the capacitor as close as possible to the device pin.

ALTERA
now part of Intel

Figure 3-1: Recommended RC Filter for Power Traces

There is no impedance requirement for the REFGND. Altera recommends that you use the lowest impedance with the most minimum DC resistance possible. Typical resistance is less than 1 Ω.

Altera recommends that you set a REFGND plane that extends as close as possible to the corresponding decoupling capacitor and FPGA:

- If possible, define a complete REFGND plane in the layout.
- Otherwise, route the REFGND using a trace that is as wide as possible from the island to the FPGA pins and decoupling capacitor.
- The REFGND ground is the reference ground plane for the ADC V_{REF} and analog input.
- Connect REFGND ground to the system digital ground through ferrite beads. You can also evaluate the ferrite bead option by comparing the impedance with the frequency specifications.

Guidelines: Board Design for Analog Input

The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. There must be no parallel routing between analog input signals and I/O traces, and between analog input signals and FPGA I/O signal traces.

- The total RC constant, including package, trace, and parasitic driver must be less than 42.4 ns. This consideration is to ensure that the input signal is fully settled during the sampling phase.
- If you reduce the total sampling rate, you can calculate the required settling time as
  $$0.45 \div F_S > 10.62 \times RC\ constant.$$
- To gain more total RC margin, Altera recommends that you set the driver source impedance as low as possible:

  - For prescaler-disabled channel—less than 1 kΩ
  - For prescaler-enabled channel—less than 11 Ω

Trace Routing

- If possible, route the switching I/O traces on different layers.
- There is no specific requirement for input signal trace impedance. However, the DC resistance for the input trace must be as low as possible.
- Route the analog input signal traces as adjacent as possible to REFGND if there is no REFGND plane.
- Use REFGND as ground reference for the ADC input signal.
- For prescaler-enabled input signal, set the ground reference to REFGND. Performance degrades if the ground reference of prescaler-enabled input signal is set to common ground (GND).

Input Low Pass Filter Selection

- A low pass RC filter can reduce the trace spacing between analog input signal and digital I/O signal to meet -100 dB crosstalk requirement. Altera recommends that you place an active low pass filter to help in meeting the required settling time.
- Place the low pass filter as close as possible to the analog input signals.
- The cut off frequency depends on the analog input frequency. Altera recommends that the $F_{cutoff \, @ \, -3dB}$ is five times the input frequency.
- You can download the ADC input SPICE model for ADC front end board design simulation from the Altera website.

Table 3-1: RC Constant and Filter Value

This table is an example of the method to quantify the RC constant and identify the RC filter value.

Total RC Constant = $(R_{DRIVER} + R_{BOARD} + R_{PACKAGE} + R_{FILTER}) \times (C_{DRIVER} + C_{BOARD} + C_{PACKAGE} + C_{FILTER} + C_{PIN})$

Driver		Board		Package		Pin Capacitance (pF)	RC Filter		$F_{cutoff \, @ \, -3dB}$ (MHz)	Total RC Constant (ns)	Settling Time (ns)
R (Ω)	C (pF)	R (Ω)	C (pF)	R (Ω)	C (pF)		R (Ω)	C (pF)			
5	2	5	17	3	5	6	60	550	4.82	42.34	42.4
10	2	5	17	3	5	6	50	580	5.49	41.48	42.4

**Figure 3-2: Passive Low Pass Filter Example**
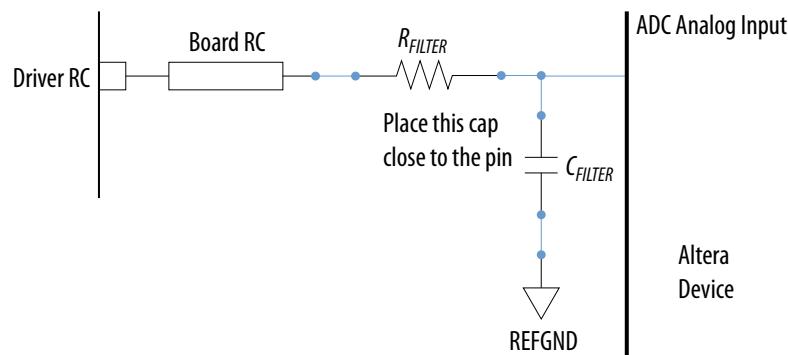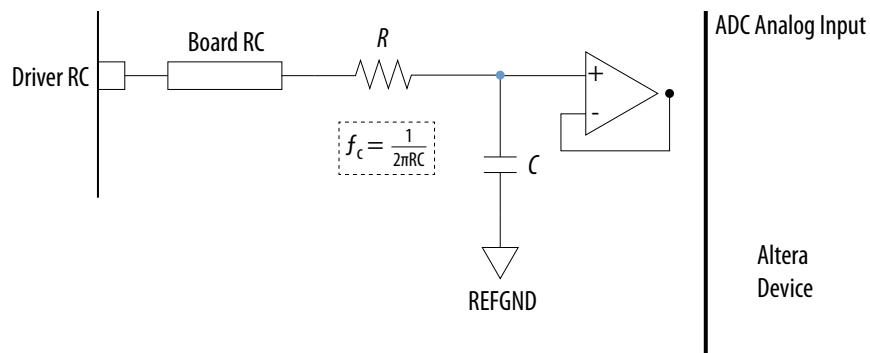
Figure 3-3: First Order Active Low Pass Filter Example

This figure is an example. You can design *n*th order active low pass filter.

Related Information

- **Parameters Settings for Generating Altera Modular ADC or Altera Modular Dual ADC IP Core** on page 4-5
- **Altera Modular ADC Parameters Settings** on page 5-2
- **Altera Modular Dual ADC Parameters Settings** on page 5-8
- **SPICE Models for Altera Devices**
  Provides the MAX 10 ADC spice model download.

# Guidelines: Board Design for ADC Reference Voltage Pin
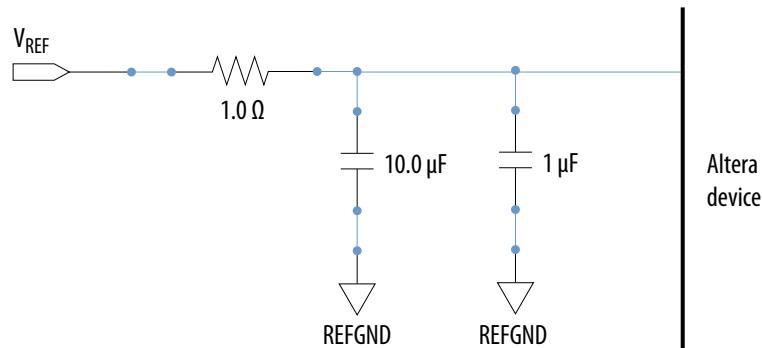
The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. There is no parallel routing between analog input signals and I/O traces. Route the V_{REF} traces as adjacent as possible to REFGND.

If a REFGND plane is not possible, route the analog input signal as adjacent as possible to REFGND.

There is one ADC reference voltage pin in each MAX 10 device. This pin uses REFGND as ground reference. Keep the trace resistance less than 0.8 Ω.

Figure 3-4: RC Filter Design Example for Reference Voltage Pin

Place the RC filter as close as possible to the analog input pin.
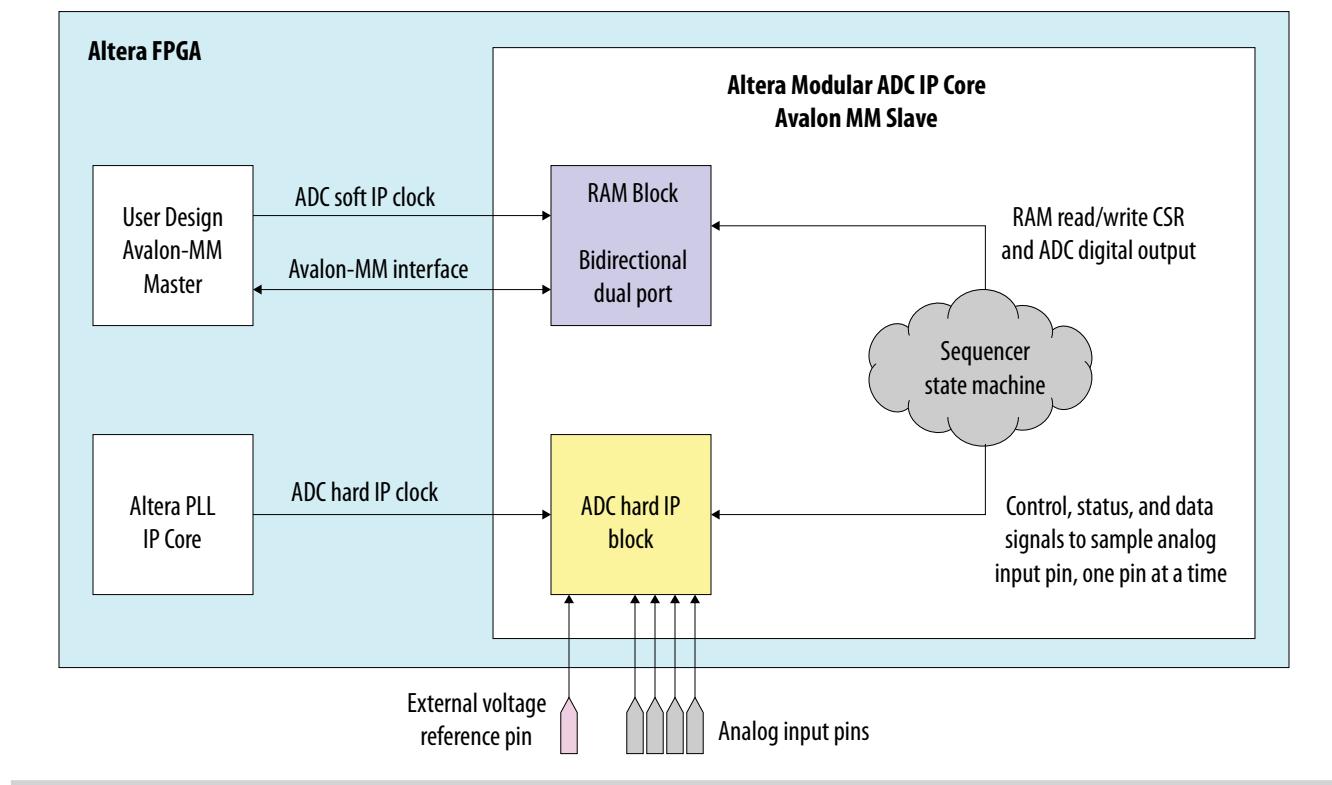
You can implement your ADC design in the Quartus Prime software. The software contains tools for you to create and compile your design, and configure your device.

The Quartus Prime software allows you to set up the parameters and generate your Altera Modular ADC IP core. To understand the ADC signal performance, you can use the Quartus Prime ADC Toolkit. For more information about using the Quartus Prime software and the ADC toolkit, refer to the related information.

Figure 4-1: High Level Block Diagram of the MAX 10 ADC Solution

Related Information

- **MAX 10 Analog to Digital Converter Overview** on page 1-1

- **Quartus II Handbook, Volume 1: Design and Synthesis**
  Provides more information about using IP cores in the Quartus II software.
- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.
- **ADC Toolkit**
  Provides more information about the ADC Toolkit.
- **ADC Toolkit for Testing ADC Performance** on page 2-20

Creating MAX 10 ADC Design

To create your ADC design, you must customize and generate the ALTPLL and Altera Modular ADC IP cores.

The ALTPLL IP core provides the clock for the Altera Modular ADC IP core.

1. Customize and generate the ALTPLL IP core.
2. Customize and generate the Altera Modular ADC IP core.
3. Connect the ALTPLL IP core to the Altera Modular ADC IP core.
4. Create ADC Avalon slave interface to start the ADC.

Related Information

- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Parameters Settings for Generating Altera Modular ADC or Altera Modular Dual ADC IP Core** on page 4-5
- **Parameters Settings for Generating ALTPLL IP Core** on page 4-3
- **Completing ADC Design** on page 4-8
- **How to Create ADC Design in MAX 10 Device Using Qsys Tool**
  Provides video instruction that demonstrates how to create the ADC design in MAX 10 devices using the Qsys system integration tool within the Quartus Prime software and how to use the ADC toolkit to view the measured analog signal.
- **How to Create Simultaneous Measurement with MAX 10 ADC, Part 1**
  Provides the first part of video instruction series that explains the differences between the MAX 10 Altera Modular ADC and Altera Modular Dual ADC IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.
- **How to Create Simultaneous Measurement with MAX 10 ADC, Part 2**
  Provides the second part of video instruction series that explains the differences between the MAX 10 Altera Modular ADC and Altera Modular Dual ADC IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.

Customizing and Generating Altera Modular ADC IP Core

Altera recommends that you use the Altera Modular ADC IP core with a Nios II processor, which supports the ADC HAL driver.

1. Create a new project in the Quartus Prime software.

    While creating the project, select a device that has one or two ADC blocks.
2. In the Quartus Prime software, select **Tools** > **Qsys**.
3. In the **Qsys** window, select **File** > **New System**.
    A clock source block is automatically added under the **System Contents** tab.
4. In the **System Contents** tab, double click the clock name.
5. In the **Parameters** tab for the clock source, set the **Clock frequency**.
6. In the **IP Catalog** tab in the **Qsys** window, double click **Processors and Peripherals** > **Peripherals** > **Altera Modular ADC**.
    The Altera Modular ADC appears in the **System Contents** tab and the Altera Modular ADC parameter editor opens.
7. In the Altera Modular ADC parameter editor, specify the parameter settings and channel sampling sequence for your application.
8. In the **System Contents** tab in the **Qsys** window, double click the **Export** column of the `adc_pll_clock` and `adc_pll_locked` interfaces to export them.
9. Connect the `clock`, `reset_sink`, `sample_store_csr`, and `sample_store_irq` signals. Optionally, you can use the Nios II Processor, On-Chip Memory, and JTAG UART IP cores to form a working ADC system that uses the Altera ADC HAL drivers.
10. In the **Qsys** window, select **File** > **Save**.

You can copy an example HDL code to declare an instance of your ADC system. In the **Qsys** window, select **Generate** > **HDL Example**.

Related Information

- **Creating MAX 10 ADC Design** on page 4-2
- **Parameters Settings for Generating ALTPLL IP Core** on page 4-3
- **Parameters Settings for Generating Altera Modular ADC or Altera Modular Dual ADC IP Core** on page 4-5
- **Configuration 1: Standard Sequencer with Avalon-MM Sample Storage** on page 2-10
- **Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection** on page 2-11
- **Configuration 3: Standard Sequencer with External Sample Storage** on page 2-13
- **Configuration 4: ADC Control Core Only** on page 2-13
- **ADC PLL Clock Interface of Altera Modular ADC and Altera Modular Dual ADC** on page 5-17
- **ADC PLL Locked Interface of Altera Modular ADC and Altera Modular Dual ADC** on page 5-18

Parameters Settings for Generating ALTPLL IP Core

Navigate through the ALTPLL IP core parameter editor and specify the settings required for your design. After you have specified all options as listed in the following table, you can generate the HDL files and the optional simulation files.

For more information about all ALTPLL parameters, refer to the related information.

Table 4-1: ALTPLL Parameters Settings

To generate the PLL for the ADC, use the following settings.

Tab	Parameter	Setting
Parameter Settings > General/ Modes	What is the frequency of the inclk0 input?	Specify the input frequency to the PLL.
Parameter Settings > Inputs/ Lock	Create an 'areset' input to asynchronously reset the PLL	Turn off this option.
	Create 'locked' output	Turn on this option. You need to connect this signal to the `adc_pll_locked` port of the Altera Modular ADC or Altera Modular Dual ADC IP core.
Output Clocks > clk c0	Use this clock	Turn on this option.
	Enter output clock frequency	Specify an output frequency of 2, 10, 20, 40, or 80 MHz. You can specify any of these frequencies. The ADC block runs at 1 MHz internally but it contains a clock divider that can further divide the clock by a factor of 2, 10, 20, 40, and 80.
		Use this same frequency value in your Altera Modular ADC or Altera Modular Dual ADC IP core. You need to connect this signal to the `adc_pll_clock` port of the Altera Modular ADC or Altera Modular Dual ADC IP core.
		Different ADC sampling rates support different clock frequencies. For a valid sampling rate and clock frequency combination, refer to the related information.

Related Information

- **Creating MAX 10 ADC Design** on page 4-2
- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Completing ADC Design** on page 4-8
- **MAX 10 Clock Networks and PLLs User Guide**
- **ADC PLL Clock Interface of Altera Modular ADC and Altera Modular Dual ADC** on page 5-17
- **ADC PLL Locked Interface of Altera Modular ADC and Altera Modular Dual ADC** on page 5-18
- **Valid ADC Sample Rate and Input Clock Combination** on page 5-13

Parameters Settings for Generating Altera Modular ADC or Altera Modular Dual ADC IP Core

Navigate through the Altera Modular ADC IP core parameter editor and specify the settings required for your design. After you have specified all options as listed in the following tables, you can generate the HDL files and the optional simulation files.

Altera recommends that you save the generated files in the design file directory (default setting).

For more information about each Altera Modular ADC or Altera Modular Dual ADC parameter, refer to the related information section.

Table 4-2: Parameter Settings in General Group

Parameter	Setting
Core Variant	There are four configuration variants of the Altera Modular ADC IP core. Select the core variant that meets your requirement. For more information, refer to the related information.
Debug Path	Turn this on to enable the debug path for the selected core variant. You can use the ADC Toolkit to monitor the ADC performance.
Generate IP for which ADCs of this device?	For devices with two ADC blocks, select the ADC block for which you are generating the IP core. There are feature differences between the two ADC blocks. The temperature sensor is available only in the first ADC block. There are also different channel counts in both ADC blocks.
ADC Sample Rate	Select the predefined sampling rate for the ADC from 25 kHz to 1 MHz. A lower sampling rate allows you greater flexibility in designing your ADC front end driver circuit. For example, using a lower sampling rate gives you a wider settling time margin for your filter design. The sampling rate you select affects which ADC input clock frequencies are available. Refer to the related information for more details about the sampling rate and the required settling time.
ADC Input Clock	Select the same frequency that you set for the ALTPLL IP core that clocks the Altera Modular ADC IP core. When configuring the ALTPLL IP core, specify a clock frequency that is supported by the ADC sampling rate. For more details, refer to the related information.
Reference Voltage Source	Select whether you want to use external or internal reference voltage. There is only one VREF pin. For dual ADC blocks, you can use one external V_{REF} source for both ADC blocks, or external V_{REF} for one ADC block and internal V_{REF} for the other ADC block.
External Reference Voltage	If you use external V_{REF} source in your design, specify the V_{REF} level.

4-6 Parameters Settings for Generating Altera Modular ADC or Altera Modular...

UG-M10ADC
2016.05.02

Parameter	Setting
Enable user created expected output file	If you want to use your own expected output file to simulate the ADC, enable this function and specify the file for the specific ADC channel. For more information about this feature and the file format, refer to the related information.

Table 4-3: Parameters Settings in Channels Group

You can navigate through the tabs for all the available channels and turn on the channel you want to use. In each channel (and **TSD**) tab, you can specify the settings in this table.

Parameter	Setting
Use Channel 0 (Dedicated analog input pin - ANAIN)	This option is available in the **CH0** tab. CH0 is the dedicated analog input channel. If you want to use the dedicated analog input, turn on this option.
User created expected output file	If you enabled the option to use your own expected output file, click **Browse** and select the file for each enabled channel. This option is available in all channel tabs except the TSD tab.
Use Channel *N*	You can select which dual-function ADC channels to turn on or off. There are 16 channels (CH1 to CH16) for single ADC devices and 8 channels (CH1 to CH8) for each ADC block in dual ADC devices.
Use on-chip TSD	This option is available in the **TSD** tab. The TSD channel is the temperature sensing channel. Turn on this option if you want the IP core to read the built-in temperature sensor in the ADC block. The sampling rate of the ADC block falls to 50 kHz when it reads the temperature measurement. After it completes the temperature reading, the ADC sampling rate returns to 1 MHz. For the Altera Modular Dual ADC IP core, if you specify the TSD in a sequencer slot for ADC1, specify NULL in the same sequencer slot number for ADC2.
Enable Maximum threshold for Channel *N*	Turn on this option if you want to set a maximum threshold value for the channel.
Enter Maximum Threshold for Channel *N*	Enter the maximum threshold voltage for the channel. The IP core will generate a threshold violation notification signal to indicate that the sampled data is over the threshold value that you specify.
Enable Maximum threshold for on-chip TSD (TSD tab)	Enter the maximum threshold temperature for the temperature sensor in Celsius. The IP core will generate a threshold violation notification signal to indicate that the sampled temperature is over the temperature that you specify.

Parameter	Setting
Enable Minimum threshold for Channel *N*	Turn on this option if you want to set a minimum threshold value for the channel.
Enter Minimum Threshold for Channel *N*	Enter the minimum threshold voltage for the channel. The IP core will generate a threshold violation notification signal to indicate that the sampled data is below the threshold value that you specify.
Enter Minimum Threshold for on-chip TSD (TSD tab)	Enter the maximum threshold temperature for the temperature sensor in Celsius. The IP core will generate a threshold violation notification signal to indicate that the sampled temperature is below the temperature that you specify.

Table 4-4: Parameters Settings in Sequencer Group

Parameter	Setting
Number of slot used	Select the number of channels to use for conversion. The parameter editor displays the number of slots available in the **Conversion Sequence Channels** based on your selection.
Slot *N*	For each available slot, select the channel to sample in the sequence. The available channels depend on the channels that you turned on in the **Channels** parameters group. If you turned on a channel but do not select the channel in any of the sequencer slots, the unselected channel is not measured during the ADC sampling sequence. The ADC block samples the measurements in the sequence you specify. After it reaches the last slot in the sequence, it repeats the sampling from the first slot.

Related Information

- **Creating MAX 10 ADC Design** on page 4-2
- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Completing ADC Design** on page 4-8
- **Altera Modular ADC Parameters Settings** on page 5-2
- **Altera Modular Dual ADC Parameters Settings** on page 5-8
- **Altera Modular ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping** on page 5-6
- **Altera Modular Dual ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping** on page 5-12
- **Valid ADC Sample Rate and Input Clock Combination** on page 5-13
- **User-Specified ADC Logic Simulation Output** on page 2-22
  Provides more information and examples about the user-specified output file format.
- **Guidelines: Board Design for Analog Input** on page 3-2
  Provides more information about the sampling rate and settling time.
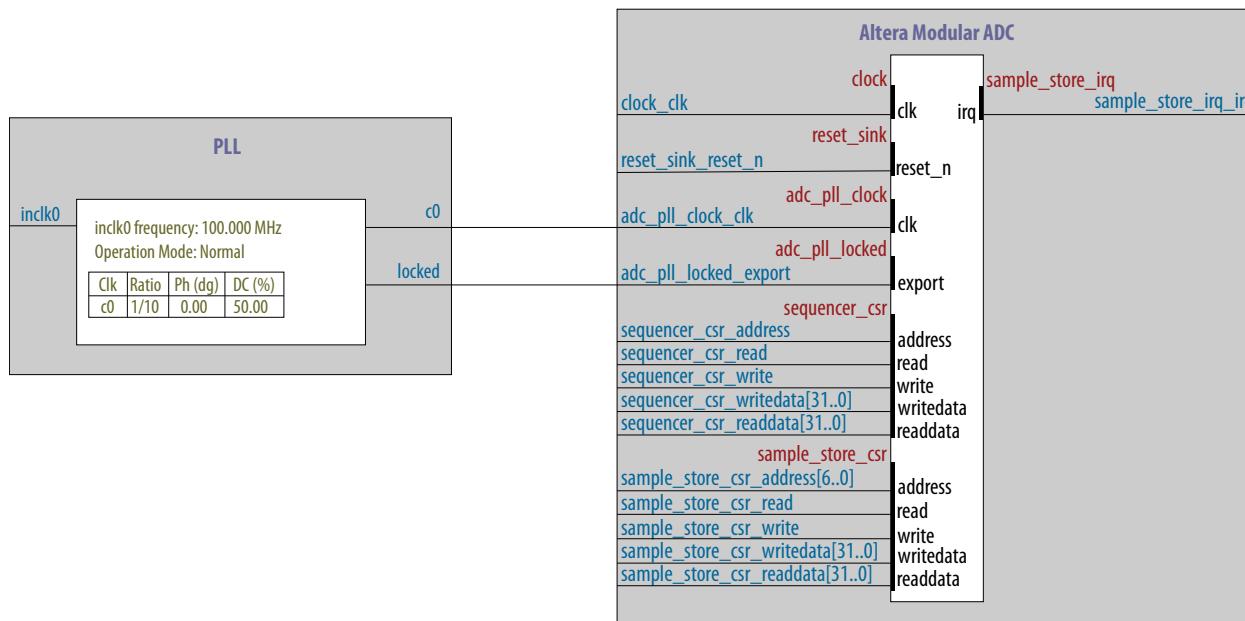
Completing ADC Design

The ADC design requires that the ALTPLL IP core clocks the Altera Modular ADC IP core.

Before you begin

Generate the ALTPLL and Altera Modular ADC IP cores with the settings in the related information.

Figure 4-2: Basic MAX 10 ADC Design

1. Create the design as shown in the preceding figure.
2. Connect the `c0` signal from the ALTPLL IP core to the `adc_pll_clock_clk` port of the Altera Modular ADC IP core.
3. Connect the `locked` signal from the ALTPLL IP core to the `adc_pll_locked_export` port of the Altera Modular ADC IP core.
4. Create the ADC Avalon slave interface to start the ADC.

Related Information

- **Creating MAX 10 ADC Design** on page 4-2
- **Parameters Settings for Generating ALTPLL IP Core** on page 4-3
- **Parameters Settings for Generating Altera Modular ADC or Altera Modular Dual ADC IP Core** on page 4-5
- **Configuration 1: Standard Sequencer with Avalon-MM Sample Storage** on page 2-10
- **Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection** on page 2-11
- **Configuration 3: Standard Sequencer with External Sample Storage** on page 2-13
- **Configuration 4: ADC Control Core Only** on page 2-13

2016.05.02

The Altera Modular ADC or Altera Modular Dual ADC IP core is a soft controller for the ADC hard IP blocks. You can generate soft IPs to instantiate the on-chip ADC blocks. With this IP core, you can configure the ADCs and abstract the low level handshake with the ADC hard IP blocks.

The Quartus Prime software generates your customized Altera Modular ADC or Altera Modular Dual ADC IP core according to the parameter options that you set in the parameter editor.

Related Information

ISO
9001:2008
Registered

now part of Intel

Altera Modular ADC Parameters Settings

There are three groups of options: **General**, **Channels**, and **Sequencer**.

Table 5-1: Altera Modular ADC Parameters - General

Parameter	Allowed Values	Description
Core Variant	- Standard sequencer with Avalon-MM sample storage - Standard sequencer with Avalon-MM sample storage and threshold violation detection - Standard sequencer with external sample storage - ADC control core only	Selects the core configuration for the Altera Modular ADC IP core.
Debug Path	- Disabled - Enabled	Enables the debug path.
Generate IP for which ADCs of this device?	1st ADC 2nd ADC	For devices that have two ADC blocks, specifies which ADC block you want to instantiate using the IP core.
ADC Sample Rate	25, 50, 100, 200, 250, and 500 kHz, and 1 MHz.	Specifies the ADC sampling rate. The sampling rate you select affects which ADC input clock frequencies are available. Refer to the related information for more details about the sampling rate and the required settling time.

Parameter	Allowed Values	Description
ADC Input Clock	2, 10, 20, 40, and 80 MHz	Specifies the frequency of the PLL clock counter zero (`c0`) clock supply for the ADC core clock. • You must configure the `c0` of the first ALTPLL IP core that you instantiate to output one of the frequencies in the allowed values list. • Connect the ALTPLL `c0` output signal to the Altera Modular ADC clk_in_pll_c0 input signal. For valid ADC sampling rate and input clock frequencies combinations, refer to the related information.
Reference Voltage Source	• External • Internal	Specifies the source of voltage reference for the ADC: • External—uses `VCCVREF` pin as the voltage reference source. • Internal—uses the on-chip 2.5 V (3.0/3.3V on voltage-regulated devices) as the voltage reference source.
External Reference Voltage	• Dual supply devices: up to 2.5 V • Single supply devices: up to 3.63 V	Specifies the voltage of `VCCVREF` pin if you use it as reference voltage to the ADC.
Enable user created expected output file	• Enabled • Disabled	Specifies the source of output data for ADC logic simulation: • Enabled—uses the expected output data you provide for each ADC channel except the TSD channel. • Disabled—uses fixed expected output data for all ADC channels. This is the default setting. For more information about this feature and the file format, refer to the related information.

Table 5-2: Altera Modular ADC Parameters - Channels

This group of parameters is divided into several tabs—one for each channel, and one tab for the TSD.

Parameter	Allowed Values	Description
Use Channel 0 (Dedicated analog input pin - ANAIN) (CH0 tab)	• On • Off	Enables the dedicated analog input pin.

Parameter	Allowed Values	Description
User created expected output file	—	Specifies user-created expected output file for ADC logic simulation for the channel. This option is available for each enabled channel except the TSD if you select **Enable user created expected output file**.
Use Channel *N* (Each channel in its own tab)	• On • Off	Enables the dual-function analog input, where *N* is: • 1 to 16 channels for single ADC devices • 1 to 8 channels for dual ADC devices
Use on-chip TSD (TSD tab)	• On • Off	Specifies that the IP core reads the built-in temperature sensor in the ADC. If you turn on this option, the ADC sampling rate is up to 50 kHz when it reads the temperature measurement. After it completes the temperature reading, the ADC sampling rate is up to 1 MHz.
Enable Maximum threshold for Channel *N* (Each channel in its own tab)	• On • Off	Enables the maximum threshold feature for the channel. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enable Maximum threshold for on-chip TSD (TSD tab)	• On • Off	Enables the maximum threshold feature for the TSD. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Maximum Threshold for Channel *N* (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the maximum threshold value in Volts. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Maximum Threshold for on-chip TSD (TSD tab)	—	Specifies the maximum threshold value in Celsius. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enable Minimum threshold for Channel *N* (Each channel in its own tab, including channel 0)	• On • Off	Enables the minimum threshold feature for the channel. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.

Parameter	Allowed Values	Description
Enable Minimum threshold for on-chip TSD (TSD tab)	• On • Off	Enables the minimum threshold feature for the TSD. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Minimum Threshold for Channel N (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the minimum threshold value in Volts. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Minimum Threshold for on-chip TSD (TSD tab)	—	Specifies the minimum threshold value in Celsius. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enable Prescaler for Channel N	• On • Off	Enables the prescaler function, where N is: • Channels 8 and 16 (if available) for single ADC devices. • Channel 8 of ADC1 or ADC2 for dual ADC devices.

Table 5-3: Altera Modular ADC Parameters - Sequencer

Parameter	Allowed Values	Description
Number of slot used	1 to 64	Specifies the number of conversion sequence slots to use. The **Conversion Sequence Channels** section displays the slots available according to the number of slots you select here.
Slot N	Enabled channel number (CH N)	Specifies which enabled ADC channel to use for the slot in the sequence. The selection option lists the ADC channels that you turned on in the **Channels** parameter group.

Related Information

- **Sequencer Core** on page 2-17
- **Configuration 1: Standard Sequencer with Avalon-MM Sample Storage** on page 2-10
- **Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection** on page 2-11
- **Configuration 3: Standard Sequencer with External Sample Storage** on page 2-13
- **Configuration 4: ADC Control Core Only** on page 2-13
- **Altera Modular ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping** on page 5-6

5-6 Altera Modular ADC IP Core Channel Name to MAX 10 Device Pin Name...

UG-M10ADC
2016.05.02

Altera Modular ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping

Each ADC channel in the Altera Modular ADC IP core corresponds to different device pin name for single and dual ADC devices.

Table 5-4: Altera Modular ADC IP Core Channel to Pin Mapping for Single ADC Devices

Channel Name	Pin Name
CH0	ANAIN1
CH1	ADC1IN1
CH2	ADC1IN2
CH3	ADC1IN3
CH4	ADC1IN4
CH5	ADC1IN5
CH6	ADC1IN6
CH7	ADC1IN7
CH8	ADC1IN8
CH9	ADC1IN9
CH10	ADC1IN10
CH11	ADC1IN11
CH12	ADC1IN12
CH13	ADC1IN13
CH14	ADC1IN14
CH15	ADC1IN15
CH16	ADC1IN16

Table 5-5: Altera Modular ADC IP Core Channel to Pin Mapping for Dual ADC Devices

ADC Block	Channel Name	Pin Name
First ADC	CH0	ANAIN1
	CH1	ADC1IN1
	CH2	ADC1IN2
	CH3	ADC1IN3
	CH4	ADC1IN4
	CH5	ADC1IN5
	CH6	ADC1IN6
	CH7	ADC1IN7
	CH8	ADC1IN8
Second ADC	CH0	ANAIN2
	CH1	ADC2IN1
	CH2	ADC2IN2
	CH3	ADC2IN3
	CH4	ADC2IN4
	CH5	ADC2IN5
	CH6	ADC2IN6
	CH7	ADC2IN7
	CH8	ADC2IN8

Altera Modular Dual ADC Parameters Settings

There are three groups of options: **General**, **Channels**, and **Sequencer**.

Table 5-6: Altera Modular Dual ADC Parameters - General

Parameter	Allowed Values	Description
Core Variant	• Standard sequencer with Avalon-MM sample storage • Standard sequencer with Avalon-MM sample storage and threshold violation detection • Standard sequencer with external sample storage • ADC control core only	Selects the core configuration for the Altera Modular Dual ADC IP core.
ADC Sample Rate	25, 50, 100, 200, 250, and 500 kHz, and 1 MHz.	Specifies the ADC sampling rate. The sampling rate you select affects which ADC input clock frequencies are available. Refer to the related information for more details about the sampling rate and the required settling time.
ADC Input Clock	2, 10, 20, 40, and 80 MHz	Specifies the frequency of the PLL clock counter zero (c0) clock supply for the ADC core clock. • You must configure the c0 of the first ALTPLL IP core that you instantiate to output one of the frequencies in the allowed values list. • Connect the ALTPLL c0 output signal to the Altera Modular Dual ADC clk_in_pll_c0 input signal. For valid ADC sampling rate and input clock frequencies combinations, refer to the related information.

Parameter	Allowed Values	Description
Reference Voltage (ADC1 or ADC2)	• External • Internal	Specifies the source of voltage reference for the ADC: • External—uses VCCVREF pin as the voltage reference source. • Internal—uses the on-chip 2.5 V (3.0/3.3V on voltage-regulated devices) as the voltage reference source.
External Reference Voltage	• Dual supply devices: up to 2.5 V • Single supply devices: up to 3.63 V	Specifies the voltage of VCCVREF pin if you use it as reference voltage to the ADC.
Enable user created expected output file	• Enabled • Disabled	Specifies the source of output data for ADC logic simulation: • Enabled—uses the expected output data you provide for each ADC channel except the TSD channel. • Disabled—uses fixed expected output data for all ADC channels. This is the default setting. For more information about this feature and the file format, refer to the related information.

Table 5-7: Altera Modular Dual ADC Parameters - Channels

This group of parameters is divided into two main tabs for ADC1 and ADC2. For each tab, there are several channel tabs—one for each channel, and one tab for the TSD in ADC1.

Parameter	Allowed Values	Description
Use Channel 0 or 9 (Dedicated analog input pin - ANAIN) (CH0 tab for ADC1 or CH9 tab for ADC2)	• On • Off	Enables the dedicated analog input pin for ADC1 or ADC2.
User created expected output file	—	Specifies user-created expected output file for ADC logic simulation for the channel. This option is available for each enabled channel except the TSD if you select **Enable user created expected output file**.

Parameter	Allowed Values	Description
Use Channel *N* (Each channel in its own tab)	• On • Off	Enables the dual-function analog input, where *N* is: • Channels 1 to 8 for ADC1 • Channels 10 to 17 for ADC2
Use on-chip TSD (TSD tab in ADC1 only)	• On • Off	Specifies that the IP core reads the built-in temperature sensor in ADC1. If you turn on this option, the ADC sampling rate is up to 50 kHz when it reads the temperature measurement. After it completes the temperature reading, the ADC sampling rate is up to 1 MHz. **Note:** If you select the TSD for a sequencer slot in ADC1, select NULL for the same sequencer slot number in ADC2.
Enable Maximum threshold for Channel *N* (Each channel in its own tab)	• On • Off	Enables the maximum threshold feature for the channel. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enable Maximum threshold for on-chip TSD (TSD tab)	• On • Off	Enables the maximum threshold feature for the TSD. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Maximum Threshold for Channel *N* (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the maximum threshold value in Volts. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Maximum Threshold for on-chip TSD (TSD tab)	—	Specifies the maximum threshold value in Celsius. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enable Minimum threshold for Channel *N* (Each channel in its own tab, including channel 0)	• On • Off	Enables the minimum threshold feature for the channel. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.

Parameter	Allowed Values	Description
Enable Minimum threshold for on-chip TSD (TSD tab)	• On • Off	Enables the minimum threshold feature for the TSD. This option is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Minimum Threshold for Channel *N* (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the minimum threshold value in Volts. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enter Minimum Threshold for on-chip TSD (TSD tab)	—	Specifies the minimum threshold value in Celsius. This setting is available only if you select the **Standard sequencer with Avalon-MM sample storage and threshold violation detection** core variant.
Enable Prescaler for Channel *N*	• On • Off	Enables the prescaler function, where *N* is: • Channel 8 in ADC1 • Channel 17 in ADC2

Table 5-8: Altera Modular Dual ADC Parameters - Sequencer

Parameter	Allowed Values	Description
Number of slot used	1 to 64	Specifies the number of conversion sequence slots to use for both ADC1 and ADC2. The **Conversion Sequence Channels** section displays the slots available for ADC1 and ADC2 according to the number of slots you select here.
Slot *N*	Enabled channel number (CH *N*)	Specifies which enabled ADC channel to use for the slot in the sequence. The selection option lists the ADC channels that you turned on in the **Channels** parameter group for ADC1 and ADC2. **Note:** If you select the TSD for a sequencer slot in ADC1, select NULL for the same sequencer slot number in ADC2.

Related Information

- **Sequencer Core** on page 2-17
- **Configuration 1: Standard Sequencer with Avalon-MM Sample Storage** on page 2-10

5-12 Altera Modular Dual ADC IP Core Channel Name to MAX 10 Device Pin Name...

UG-M10ADC
2016.05.02

Altera Modular Dual ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping

Each ADC channel in the Altera Modular Dual ADC IP core corresponds to different device pin name.

Table 5-9: Altera Modular Dual ADC IP Core Channel to Pin Mapping

ADC Block	Channel Name	Pin Name
ADC1	CH0	ANAIN1
	CH1	ADC1IN1
	CH2	ADC1IN2
	CH3	ADC1IN3
	CH4	ADC1IN4
	CH5	ADC1IN5
	CH6	ADC1IN6
	CH7	ADC1IN7
	CH8	ADC1IN8
ADC2	CH9	ANAIN2
	CH10	ADC2IN1
	CH11	ADC2IN2
	CH12	ADC2IN3
	CH13	ADC2IN4
	CH14	ADC2IN5
	CH15	ADC2IN6
	CH16	ADC2IN7
	CH17	ADC2IN8

Valid ADC Sample Rate and Input Clock Combination

Each predefined ADC sampling rate supports a list of input clock frequencies. When you configure the ALTPLL IP core to clock the ADC, use an ADC input clock frequency supported by your ADC sampling rate.

The ability to specify the ADC sampling rate allows you more design flexibility. If you are not using the maximum MAX 10 ADC sampling rate, you get a wider settling time margin.

Table 5-10: Valid Combination of ADC Sampling Rate and Input Clock

Total ADC Sampling Rate (kHz)	ADC Input Clock Frequency (MHz)				
	2	10	20	40	80
1000	Yes	Yes	Yes	Yes	Yes
500	—	Yes	Yes	Yes	—
250	—	Yes	Yes	—	—
200	Yes	—	—	—	—
125	—	Yes	—	—	—
100	Yes	—	—	—	—
50	Yes	—	—	—	—
25	Yes	—	—	—	—

Related Information

- **Parameters Settings for Generating ALTPLL IP Core** on page 4-3
- **Parameters Settings for Generating Altera Modular ADC or Altera Modular Dual ADC IP Core** on page 4-5
- **Altera Modular ADC Parameters Settings** on page 5-2
- **Altera Modular Dual ADC Parameters Settings** on page 5-8

Altera Modular ADC and Altera Modular Dual ADC Interface Signals

Depending on parameter settings you specify, different signals are available for the Altera Modular ADC or Altera Modular Dual ADC IP core.

Command Interface of Altera Modular ADC and Altera Modular Dual ADC

The command interface is an Avalon-ST type interface that supports a ready latency of 0.

Table 5-11: Command Interface Signals

Signal	Width (Bit)	Description
valid	1	Indication from the source port that current transfer is valid.
ready	1	Indication from the sink port that it is ready for current transfer.

Signal	Width (Bit)	Description
channel	5	Indicates the channel that the ADC hard block samples from for current command. • 31—recalibration request • 30:18—not used • 17—temperature sensor • 16:0—channels 16 to 0; where channel 0 is the dedicated analog input pin and channels 1 to 16 are the dual purpose analog input pins
startofpacket	1	Indication from the source port that current transfer is the start of packet. • For altera_adc_sequencer core implementation, the IP core asserts this signal during the first slot of conversion sequence data array. • For altera_adc_control core implementation, this signal is ignored. The IP core just passes the received information back to the corresponding response interface.
endofpacket	1	Indication from the source port that current transfer is the end of packet. • For altera_adc_sequencer core implementation, IP core asserts this signal during the final slot of conversion sequence data array. • For altera_adc_control core implementation, this signal is ignored. The IP core just passes the received information back to the corresponding response interface.

Related Information

Response Interface of Altera Modular ADC and Altera Modular Dual ADC

The response interface is an Avalon-ST type interface that does not support backpressure. To avoid overflow condition at the source port, implement sink ports with response data process time that is fast enough, or with enough buffers storage.

Table 5-12: Response Interface Signals

Signal	Width (Bit)	Description
valid	1	Indication from the source port that current transfer is valid.

Signal	Width (Bit)	Description
channel	5	Indicates the ADC channel to which the ADC sampling data corresponds for the current response. • 31:18—not used • 17—temperature sensor • 16:0—channels 16 to 0; where channel 0 is the dedicated analog input pin and channels 1 to 16 are the dual purpose analog input pins
data	12 or 24	ADC sampling data: • 12 bit width for Altera Modular ADC • 24 bit width for Altera Modular Dual ADC
startofpacket	1	Indication from the source port that current transfer is the start of packet. For altera_adc_control core implementation, the source of this signal is from the corresponding command interface.
endofpacket	1	Indication from the source port that current transfer is the end of packet. For altera_adc_control core implementation, the source of this signal is from the corresponding command interface.

Threshold Interface of Altera Modular ADC and Altera Modular Dual ADC

The threshold interface is an Avalon-ST type interface that does not support backpressure.

Table 5-13: Threshold Interface Signals

Signal	Width (Bit)	Description
valid	1	Indication from the source port that current transfer is valid.
channel	5	Indicates the ADC channel for which the threshold value has been violated. • 31:18—not used • 17—temperature sensor • 16:0—channels 16 to 0; where channel 0 is the dedicated analog input pin and channels 1 to 16 are the dual purpose analog input pins

Send Feedback

Signal	Width (Bit)	Description
data	1	Indicates the type of threshold violation: • 1—Exceeds maximum threshold value • 0—Below minimum threshold value

Related Information

- **Altera Modular ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping** on page 5-6
- **Altera Modular Dual ADC IP Core Channel Name to MAX 10 Device Pin Name Mapping** on page 5-12

CSR Interface of Altera Modular ADC and Altera Modular Dual ADC

The CSR interface is an Avalon-MM slave interface.

Table 5-14: CSR Interface Signals

Signal	Width (Bit)	Description
address	1 or 7	Avalon-MM address bus. The address bus width is in the unit of word addressing: • altera_adc_sample_store core—address width is seven • altera_adc_sequencer core—address width is one
read	1	Avalon-MM read request.
write	1	Avalon-MM write request.
writedata	32	Avalon-MM write data bus.
readdata	32	Avalon-MM read data bus.

IRQ Interface of Altera Modular ADC and Altera Modular Dual ADC

The IRQ interface is an interrupt interface type.

Table 5-15: IRQ Interface Signals

Signal	Width (Bit)	Description
irq	1	Interrupt request.

Peripheral Clock Interface of Altera Modular ADC and Altera Modular Dual ADC

The peripheral clock interface is a clock sink interface type.

Table 5-16: Peripheral Clock Interface Signals

Signal	Width (Bit)	Description
clock	1	Single clock that clocks all Altera Modular ADC or Altera Modular Dual ADC micro cores.

Peripheral Reset Interface of Altera Modular ADC and Altera Modular Dual ADC

The peripheral reset interface is a reset sink interface type.

Table 5-17: Peripheral Reset Interface Signals

Signal	Width (Bit)	Description
reset_n	1	Single reset source that that resets all Altera Modular ADC or Altera Modular Dual ADC micro cores.

ADC PLL Clock Interface of Altera Modular ADC and Altera Modular Dual ADC

The ADC PLL clock interface is a clock sink interface type.

Table 5-18: ADC PLL Clock Interface Signals

Signal	Width (Bit)	Description
clock	1	ADC hard IP clock source from c0 output of dedicated PLL1 or PLL3. Export this interface from the Qsys system.

Related Information

- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Parameters Settings for Generating ALTPLL IP Core** on page 4-3

5-18 ADC PLL Locked Interface of Altera Modular ADC and Altera Modular Dual...

UG-M10ADC
2016.05.02

ADC PLL Locked Interface of Altera Modular ADC and Altera Modular Dual ADC

The ADC PLL locked interface is a conduit end interface type.

Table 5-19: ADC PLL Locked Interface Signals

Signal	Width (Bit)	Description
`conduit`	1	ADC hard IP locked signal output of dedicated PLL1 or PLL3. Export this interface from the Qsys system.

Related Information

- **Customizing and Generating Altera Modular ADC IP Core** on page 4-3
- **Parameters Settings for Generating ALTPLL IP Core** on page 4-3

Altera Modular ADC Register Definitions

The registers in the generated Altera Modular ADC IP core provide the IP core with the control and settings during operation.

Sequencer Core Registers

Table 5-20: Command Register (CMD)

Address Offset: `0x0`

Bit	Name	Attribute	Description	Value	Default
31:4	Reserved	Read	Reserved	—	0
3:1	Mode	Read-Write	Indicates the operation mode of the sequencer core. This register is ignored when the run bit (bit 0) is set. In continuous conversion, the data will be overwritten after a complete sampling sequence.	• 7—Recalibrate the ADC • 6 to 2—Reserved • 1—Single cycle ADC conversion • 0—Continuous ADC conversion	0
0	Run	Read-Write	Use this control bit to trigger the sequencer core operation. The Altera Modular ADC IP core waits until the sequencer core completes its current operation before writing to this register bit.	• 1—Run • 0—Stop	0

Related Information

Sample Storage Core Registers

Table 5-21: ADC Sample Register (ADC_SAMPLE) of Altera Modular ADC

Address Offset: `0x3F` (slot 63)—`0x0` (slot 0)

Bit	Name	Attribute	Description	Value	Default
31:12	Reserved	Read	Reserved	—	0
11:0	Sample	Read	The data sampled by the ADC for the corresponding slot.	Sampled data	0

Table 5-22: ADC Sample Register (ADC_SAMPLE) of Altera Modular Dual ADC

Address Offset: `0x3F` (slot 63)—`0x0` (slot 0)

Bit	Name	Attribute	Description	Value	Default
31:28	Reserved	Read	Reserved	—	0
27:16	Sample	Read	The data sampled by ADC2 for the corresponding slot.	Sampled data	0
15:12	Reserved	Read	Reserved	—	0
11:0	Sample	Read	The data sampled by ADC1 for the corresponding slot.	Sampled data	0

Table 5-23: Interrupt Enable Register (IER)

Address Offset: `0x40`

Clear the enable bit to prevent the corresponding interrupt status bit from causing interrupt output assertion (IRQ). The enable bit does not stop the interrupt status bit value from showing in the interrupt status register (ISR).

Bit	Name	Attribute	Description	Value	Default
31:1	Reserved	Read	Reserved	—	0
0	M_EOP	Read-Write	The enable bit for the end of packet (EOP) interrupt.	• 1—Enables the corresponding interrupt • 0—Disables the corresponding interrupt	1

Table 5-24: Interrupt Status Register (ISR)

Address Offset: `0x41`

Bit	Name	Attribute	Description	Value	Default
31:1	Reserved	Read	Reserved	—	0
0	EOP	Read-Write (one cycle)	EOP interrupt.	• 1—Indicates complete receipt of a sample block • 0—Automatically clears to 0 after indication of complete receipt	0

Related Information

Sample Storage Core on page 2-18

ADC HAL Device Driver for Nios II Gen 2

The Altera Modular ADC IP core provides a HAL device driver. You can integrate the device driver into the HAL system library for Nios II Gen 2 systems.

The Altera Modular ADC IP core provides software files that define low-level access to the hardware. You can use the macros definition and functions in the software files to initialize the Altera Modular ADC core.

- **altera_modular_adc_sequencer_regs.h**—this file defines the register map for the sequencer core. It provides symbolic constants to access the low-level hardware.
- **altera_modular_adc_sample_store_regs.h**—this file defines the register for sample storage core. It provides symbolic constants to access the low-level hardware.
- **altera_modular_adc.h**—include this file into your application. It automatically includes the other header files and defines additional functions.
- **altera_modular_adc.c**—this file implements helper functions that are defined in the header file.

2016.05.02

UG-M10ADC   ✉ **Subscribe**   💬 **Send Feedback**

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 Analog to Digital Converter User Guide**
15.0	**MAX 10 Analog to Digital Converter User Guide**

ISO 9001:2008 Registered

ALTERA
now part of Intel

2016.05.02

UG-M10ADC ✉ **Subscribe** 💬 **Send Feedback**

Date	Version	Changes
May 2016	2016.05.02	• Removed all preliminary marks. • Added new function to specify predefined ADC sampling rate up to 1 MSPS. Previously, the ADC always operate at the maximum sampling rate. • Removed link to a workaround to reduce the sampling rate. Now you can set the sampling rate in the IP core parameter editor. • Added the ADC Toolkit that supports the Altera Modular ADC and Altera Modular Dual ADC IP cores. • Added feature to simulate ADC output using your own expected output files for each ADC channel except the TSD channel. • Corrected the description for bits 11:0 and bits 27:16 of the ADC sample register for Altera Modular ADC and Altera Modular Dual ADC IP cores. Bits 11:0 and bits 27:16 hold the actual 12 bit sampled data for the storage slot instead of the slot number. • Corrected the default value for bit 0 of the interrupt enable register (IER) and interrupt status register (ISR). The default value for `M_EOP` is 1 and for `EOP` is 0.
November 2015	2015.11.02	• Added related information link to *Introduction to Altera IP Cores*. • Added links to instructional videos that demonstrate how to create ADC designs in MAX 10 devices. • Changed instances of *Quartus II* to *Quartus Prime*.
June 2015	2015.06.11	Updated the board design guidelines for analog input.

ISO 9001:2008 Registered

ALTERA
now part of Intel

B-2 Document Revision History for MAX 10 Analog to Digital Converter User...

UG-M10ADC
2016.05.02

Date	Version	Changes
May 2015	2015.05.04	- Added the Altera Modular Dual ADC IP core. - Removed F672 from the 10M25 device and added ADC information for the E144 package of the 10M04 device: - Updated the ADC block counts. - Updated the ADC vertical migration support. - Updated the ADC channel counts. - Updated the table that lists the ADC channel count to list only 8 dual function pins (instead of 16) for the M153 and U169 packages. - Updated the ADC vertical migration diagram to clarify that there are single ADC devices with eight and 16 dual function pins. - Updated the topic about ADC conversion to specify that in prescaler mode, the analog input in dual and single supply devices can measure up to 3.0 V and 3.6 V, respectively. - Updated the ADC IP core architecture figures to include features for the dual ADC IP core. - Added information and topics about the response merge and dual ADC synchronizer micro cores. - Removed notes about contacting Altera for the ADC pin RLC filter design. - Updated the ADC prescaler topic to change the ADC2 channel that supports prescaler from channel 16 to channel 17. - Updated the diagram that shows the ADC timing: - To clarify that the numbers are hexadecimal numbers. - Relabeled the signals to match the command and response interface signal names. - Updated the RC constant and filter value and the filter design example figure to clarify the source of the example values. - Added guidelines for setting up the sequencer in dual ADC mode. - Added topics that list the mapping of Altera Modular ADC and Altera Modular Dual ADC IP cores channel names to MAX 10 device pin names. - Corrected the address offset of the interrupt enable register (from 0x41 to 0x40) and interrupt status register (from 0x40 to 0x41) for the sample storage core. - Updated the sample storage core registers table to include registers for Altera Modular Dual ADC. - Removed statements about availability of the threshold trigger feature in a future version of the Quartus Prime software. The feature is now available from version 15.0 of the software.

UG-M10ADC
2016.05.02

Document Revision History for MAX 10 Analog to Digital Converter User...

B-3

Date	Version	Changes
December 2014	2014.12.15	• Added ADC prescaler block diagram. • Replaced the ADC continuous conversion timing diagram with the ADC timing diagram. • Corrected a minor error in the example in the topic about the sample storage core. • Added information that the ADC TSD measures the temperature using a 64-samples running average method. • Updated majority of the temperature codes in the table that lists the temperature code conversion. • Added chapter that provides the ADC design considerations. • Removed mention of value "0" for values allowed for the number of sequencer slots used in Altera Modular ADC IP core parameter editor. Only values 1 to 64 are allowed. • Removed the statement about enabling and disabling additional ADC response interface or debugging in the topic about the Altera Modular ADC IP core configuration variants. You can enable or disable the debug path in the parameter editor. • Removed the debug paths diagrams for each ADC core configuration. • Removed the statement about using the sequencer core to trigger recalibration. The ADC is automatically recalibrated when it switches from normal sensing mode to temperature sensing mode. • Edited text to clarify about routing power or ground traces if power or ground plane is not possible. • Updated the total RC constant values in the table that shows the RC constant and filter values calculation. • Corrected spelling for "prescaler".
September 2014	2014.09.22	Initial release.

MAX 10 FPGA Configuration User Guide

Contents

UG-M10CONFIG ✉ Subscribe 💬 Send Feedback

You can configure MAX® 10 configuration RAM (CRAM) using the following configuration schemes:

- JTAG configuration—using JTAG interface.
- Internal configuration—using internal flash.

Supported Configuration Features

Table 1-1: Configuration Schemes and Features Supported by MAX 10 Devices

Configuration Scheme	Remote System Upgrade	Compression	Design Security	SEU Mitigation
JTAG configuration	—	—	—	Yes
Internal configuration	Yes	Yes	Yes	Yes

Related IP Cores

- Altera Dual Configuration IP Core—used in the remote system upgrade feature.
- Altera Unique Chip ID IP Core—retrieves the chip ID of MAX 10 devices.

Related Information

- **MAX 10 FPGA Configuration Schemes and Features** on page 2-1
  Provides information about the configuration schemes and features.
- **MAX 10 FPGA Configuration Design Guidelines** on page 3-1
  Provides information about using the configuration schemes and features.
- **Altera Unique Chip ID IP Core** on page 2-18
- **Altera Dual Configuration IP Core** on page 2-16

ISO 9001:2008 Registered

ALTERA
now part of Intel

Configuration Schemes

Figure 2-1: High-Level Overview of JTAG Configuration and Internal Configuration for MAX 10 Devices

JTAG Configuration

In MAX 10 devices, JTAG instructions take precedence over the internal configuration scheme.

Using the JTAG configuration scheme, you can directly configure the device CRAM through the JTAG interface—`TDI`, `TDO`, `TMS`, and `TCK` pins. The Quartus® Prime software automatically generates an SRAM Object File (**.sof**). You can program the **.sof** using a download cable with the Quartus Prime software programmer.

Related Information

Configuring MAX 10 Devices using JTAG Configuration on page 3-2
Provides more information about JTAG configuration using download cable with Quartus Prime software programmer.

**ISO
9001:2008
Registered**

now part of Intel

JTAG Pins

Table 2-1: JTAG Pin

Pin	Function	Description
TDI	Serial input pin for: • instructions • boundary-scan test data • programming data	• TDI is sampled on the rising edge of TCK • TDI pins have internal weak pull-up resistors.
TDO	Serial output pin for: • instructions • boundary-scan test data • programming data	• TDO is sampled on the falling edge of TCK • The pin is tri-stated if data is not shifted out of the device.
TMS	Input pin that provides the control signal to determine the transitions of the TAP controller state machine.	• TMS is sampled on the rising edge of TCK • TMS pins have internal weak pull-up resistors.
TCK	Clock input to the BST circuitry.	—

All the JTAG pins are powered by the V_{CCIO} 1B. In JTAG mode, the I/O pins support the LVTTL/LVCMOS 3.3-1.5V standards.

Related Information

- **MAX 10 Device Datasheet**
  Provides more information about supported I/O standards in MAX 10 devices.
- **Guidelines: Dual-Purpose Configuration Pin** on page 3-1
- **Enabling Dual-purpose Pin** on page 3-2

Internal Configuration

You need to program the configuration data into the configuration flash memory (CFM) before internal configuration can take place. The configuration data to be written to CFM will be part of the programmer object file (**.pof**). Using JTAG In-System Programming (ISP), you can program the **.pof** into the internal flash.

During internal configuration, MAX 10 devices load the CRAM with configuration data from the CFM.

Internal Configuration Modes

Table 2-2: Supported Internal Configuration Modes Based on MAX 10 Feature Options

MAX 10 Feature Options	Supported Internal Configuration Mode
Compact	• Single Compressed Image • Single Uncompressed Image

MAX 10 Feature Options	Supported Internal Configuration Mode
Flash and Analog	• Dual Compressed Images • Single Compressed Image • Single Compressed Image with Memory Initialization • Single Uncompressed Image • Single Uncompressed Image with Memory Initialization

Note: In dual compressed images mode, you can use the CONFIG_SEL pin to select the configuration image.

Related Information

- **Configuring MAX 10 Devices using Internal Configuration** on page 3-5
- **Remote System Upgrade** on page 2-9

Configuration Flash Memory

The CFM is a non-volatile internal flash that is used to store configuration images. The CFM may store up to two compressed configuration images, depending on the compression and the MAX 10 devices. The compression ratio for the configuration image should be at least 30% for the device to be able store two configuration images.

Related Information

Configuration Flash Memory Permissions on page 2-20

Configuration Flash Memory Sectors

All CFM in MAX 10 devices consist of three sectors, CFM0, CFM1, and CFM2 except for the 10M02. The sectors are programmed differently depending on the internal configuration mode you select.

The 10M02 device consists of only CFM0. The CFM0 sector in 10M02 devices is programmed similarly when you select single compressed image or single uncompressed image.

Figure 2-2: Configuration Flash Memory Sectors Utilization for all MAX 10 with Analog and Flash Feature Options

Unutilized CFM1 and CFM2 sectors can be used for additional user flash memory (UFM).

Internal Configuration Mode	Configuration Flash Memory Sectors		
	CFM2	CFM1	CFM0
Dual Compressed Image		Compressed Image 1	Compressed Image 0
Single Uncompressed Image	Additional UFM	Uncompressed Image 0	
Single Uncompressed Image with Memory Initialization	Uncompressed Image 0 with Memory Initialization		
Single Compressed Image with Memory Initialization	Compressed Image 0 with Memory Initialization		
Single Compressed Image	Additional UFM		Compressed Image 0

Related Information

CFM and UFM Array Size

Configuration Flash Memory Programming Time

Table 2-3: Configuration Flash Memory Programming Time for Sectors in MAX 10 Devices

Note: The programming time reflects JTAG interface programming time only without any system overhead. It does not reflect the actual programming time that you face. To compensate the system overhead, Quartus Prime Programmer is enhanced to utilize flash parallel mode during device programming for MAX 10 10M04/08/16/25/40/50 devices. The 10M02 device does not support flash parallel mode, you may experience a relatively slow programming time if compare to other device.

Device	In-System Programming Time (s)		
	CFM2	CFM1	CFM0
10M02	—	—	5.4
10M04 and 10M08	6.5	4.6	11.1
10M16	12.0	8.9	20.8
10M25	16.4	12.6	29.0
10M40 and 10M50	30.2	22.7	52.9

In-System Programming

You can program the internal flash including the CFM of MAX 10 devices with ISP through industry standard IEEE 1149.1 JTAG interface. ISP offers the capability to program, erase, and verify the CFM. The

JTAG circuitry and ISP instructions for MAX 10 devices are compliant to the IEEE-1532-2002 programming specification.

During ISP, the MAX 10 receives the IEEE Std. 1532 instructions, addresses, and data through the TDI input pin. Data is shifted out through the TDO output pin and compared with the expected data.

The following are the generic flow of an ISP operation:

1. Check ID—the JTAG ID is checked before any program or verify process. The time required to read this JTAG ID is relatively small compared to the overall programming time.
2. Enter ISP—ensures the I/O pins transition smoothly from user mode to the ISP mode.
3. Sector Erase—shifting in the address and instruction to erase the device and applying erase pulses.
4. Program—shifting in the address, data, and program instructions and generating the program pulse to program the flash cells. This process is repeated for each address in the internal flash sector.
5. Verify—shifting in addresses, applying the verify instruction to generate the read pulse, and shifting out the data for comparison. This process is repeated for each internal flash address.
6. Exit ISP—ensures that the I/O pins transition smoothly from the ISP mode to the user mode.

You can also use the Quartus Prime Programmer to program the CFM.

Related Information

Programming .pof into Internal Flash on page 3-8
Provides the steps to program the .pof using Quartus Prime Programmer.

ISP Clamp

When a normal ISP operation begins, all I/O pins are tri-stated. For situations when the I/O pins of the device should not be tri-stated when the device is in ISP operation, you can use the ISP clamp feature.

The ISP clamp feature allows you to use the Quartus Prime software to hold each MAX 10 I/O pins to a static state when you program the device. After you successfully program the device in ISP clamp mode, the I/O pins are released and the device functions according to the new design.

You can set the ISP clamp through **Device and Pin Option**, or **Pin Assignment tool**.

Real-Time ISP

In a normal ISP operation, to update the internal flash with a new design image, the device exits from user mode and all I/O pins remain tri-stated. After the device completes programing the new design image, it resets and enters user mode.

The real-time ISP feature updates the internal flash with a new design image while operating in user mode. During the internal flash programming, the device continues to operate using the existing design. After the new design image programming process completes, the device will not reset. The new design image update only takes effect in the next reconfiguration cycle.

ISP and Real-Time ISP Instructions

Table 2-4: ISP and Real-Time ISP Instructions for MAX 10 Devices

Instruction	Instruction Code	Description
CONFIG_IO	00 0000 1101	• Allows I/O reconfiguration through JTAG ports using the IOCSR for JTAG testing. This is executed after or during configurations. • nSTATUS pin must go high before you can issue the CONFIG_IO instruction.
PULSE_NCONFIG	00 0000 0001	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected.
ISC_ENABLE_HIZ [1]	10 1100 1100	• Puts the device in ISP mode, tri-states all I/O pins, and drives all core drivers, logic, and registers. • Device remains in the ISP mode until the ISC_DISABLE instruction is loaded and updated. • The ISC_ENABLE instruction is a mandatory instruction. This requirement is met by the ISC_ENABLE_CLAMP or ISC_ENABLE_HIZ instruction.
ISC_ENABLE_CLAMP [1]	10 0011 0011	• Puts the device in ISP mode and forces all I/O pins to follow the contents of the JTAG boundary-scan register. • When this instruction is activated, all core drivers, logics, and registers are frozen. The I/O pins remain clamped until the device exits ISP mode successfully.
ISC_DISABLE	10 0000 0001	• Brings the device out of ISP mode. • Successful completion of the ISC_DISABLE instruction happens immediately after waiting 200 μs in the Run-Test/Idle state.
ISC_PROGRAM[2]	10 1111 0100	Sets the device up for in-system programming. Programming occurs in the run-test or idle state.

ISC_NOOP[2]	10 0001 0000	• Sets the device to a no-operation mode without leaving the ISP mode and targets the ISC_Default register.
		• Use when:
		  • two or more ISP-compliant devices are being accessed in ISP mode and;
		  • a subset of the devices perform some instructions while other more complex devices are completing extra steps in a given process.

[1] Do not issue the ISC_ENABLE_HIZ and ISC_ENABLE_CLAMP instructions from the core logic.
[2] All ISP and real-time ISP instructions are disabled when the device is not in the ISP or real-time ISP mode, except for the enabling and disabling instructions.

Instruction	Instruction Code	Description
ISC_ADDRESS_SHIFT[2]	10 0000 0011	Sets the device up to load the flash address. It targets the ISC_Address register, which is the flash address register.
ISC_ERASE[2]	10 1111 0010	• Sets the device up to erase the internal flash. • Issue after ISC_ADDRESS_SHIFT instruction.
ISC_READ[2]	10 0000 0101	• Sets the device up for verifying the internal flash under normal user bias conditions. • The ISC_READ instruction supports explicit addressing and auto-increment, also known as the Burst mode.
BGP_ENABLE	01 1001 1001	• Sets the device to the real-time ISP mode. • Allows access to the internal flash configuration sector while the device is still in user mode.
BGP_DISABLE	01 0110 0110	• Brings the device out of the real-time ISP mode. • The device has to exit the real-time ISP mode using the BGP_DISABLE instruction after it is interrupted by reconfiguration.

Caution: Do not use unsupported JTAG instructions. It will put the device into an unknown state and requires a power cycle to recover the operation.

Initialization Configuration Bits

Initialization Configuration Bits (ICB) stores the configuration feature settings of the MAX 10 device. You can set the ICB settings in the **Convert Programming File** tool.

Table 2-5: ICB Values and Descriptions for MAX 10 Devices

Configuration Settings	Description	Default State/Value
Set I/O to weak pull-up prior usermode	• Enable: Sets I/O to weak pull-up during device configuration. • Disable: Tri-states I/O	Enable
Configure device from CFM0 only.	Enable: • CONFIG_SEL pin setting is disabled. • Device automatically loads image 0. • Device does not load image 1 if image 0 fails. Disable: • Device automatically loads secondary image if initial image fails.	Disable

Configuration Settings	Description	Default State/Value
Use secondary image ISP data as default setting when available.	Select ISP data from initial or secondary image to include in the POF. • Disable: Use ISP data from initial image • Enable: Use ISP data from secondary image ISP data contains the information about state of the pin during ISP. This can be either tri-state with weak pull-up or clamp the I/O state. You can set the ISP clamp through **Device and Pin Option**, or **Pin Assignment** tool.	Disable
Verify Protect	To disable or enable the Verify Protect feature.	Disable
Allow encrypted POF only	If enabled, configuration error will occur if unencrypted **.pof** is used.	Disable
JTAG Secure[3]	To disable or enable the JTAG Secure feature.	Disable
Enable Watchdog	To disable or enable the watchdog timer for remote system upgrade.	Enable
Watchdog value	To set the watchdog timer value for remote system upgrade.	0x1FFF[4]

Related Information

- **.pof and ICB Settings** on page 3-6
- **Verify Protect** on page 2-19
- **JTAG Secure Mode** on page 2-18
- **ISP and Real-Time ISP Instructions** on page 2-6
- **User Watchdog Timer** on page 2-15
- **Generating .pof using Convert Programming Files** on page 3-7
  Provides more information about setting the ICB during .pof generation using Convert Programming File.

Internal Configuration Time

The internal configuration time measurement is from the rising edge of nSTATUS signal to the rising edge of CONF_DONE signal.

---

[3] The JTAG Secure feature will be disabled by default in Quartus Prime. If you are interested in using the JTAG Secure feature, contact Altera for support.

[4] The watchdog timer value depends on the MAX 10 you are using. Refer to the Watchdog Timer section for more information.

Table 2-6: Internal Configuration Time for MAX 10 Devices (Uncompressed .rbf)

Device	Internal Configuration Time (ms)							
	Unencrypted				Encrypted			
	Without Memory Initialization		With Memory Initialization		Without Memory Initialization		With Memory Initialization	
	Min	Max	Min	Max	Min	Max	Min	Max
10M02	0.3	1.7	—	—	1.7	5.4	—	—
10M04	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M08	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M16	1.1	3.7	1.4	4.5	9.3	25.3	11.7	31.5
10M25	1.0	3.7	1.3	4.4	14.0	38.1	16.9	45.7
10M40	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6
10M50	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6

Table 2-7: Internal Configuration Time for MAX 10 Devices (Compressed .rbf)

Compression ratio depends on design complexity. The minimum value is based on the best case (25% of original **.rbf** sizes) and the maximum value is based on the typical case (70% of original **.rbf** sizes).

Device	Internal Configuration Time (ms)			
	Unencrypted/Encrypted			
	Without Memory Initialization		With Memory Initialization	
	Min	Max	Min	Max
10M02	0.3	5.2	—	—
10M04	0.6	10.7	1.0	13.9
10M08	0.6	10.7	1.0	13.9
10M16	1.1	17.9	1.4	22.3
10M25	1.1	26.9	1.4	32.2
10M40	2.6	66.1	3.2	82.2
10M50	2.6	66.1	3.2	82.2

Configuration Features

Remote System Upgrade

MAX 10 devices support the remote system upgrade feature. By default, the remote system upgrade feature is enabled when you select the dual compressed image internal configuration mode.

The remote system upgrade feature in MAX 10 devices offers the following capabilities:

- Manages remote configuration
- Provides error detection, recovery, and information
- Supports direct-to-application configuration image
- Supports compressed and encrypted **.pof**

There are two methods to access remote system upgrade in MAX 10 devices:

- Altera Dual Configuration IP core
- User interface

Related Information
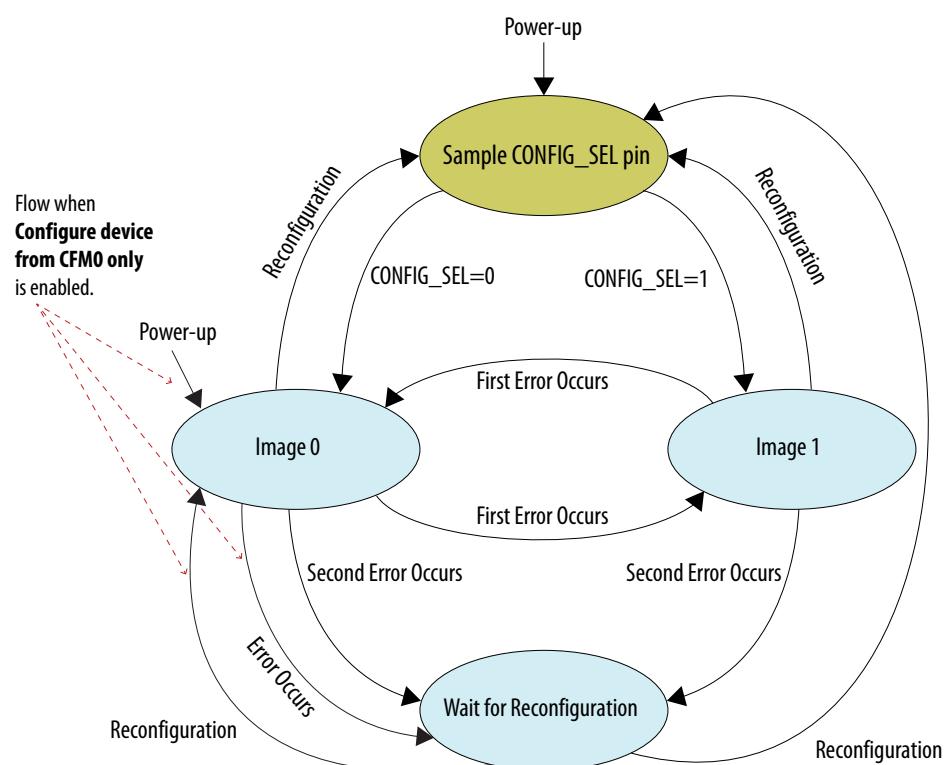
- **Altera Dual Configuration IP Core** on page 2-16
- **Accessing Remote System Upgrade through User Logic** on page 3-8
- **AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor**
  Provides reference design for remote system upgrade in MAX 10 FPGA devices.
- **I2C Remote System Update Example**
  This example demonstrates a remote system upgrade using the I2C protocol.

Remote System Upgrade Flow

Both the application configuration images, image 0 and image 1, are stored in the CFM. The MAX 10 device loads either one of the application configuration image from the CFM.

Figure 2-3: Remote System Upgrade Flow for MAX 10 Devices
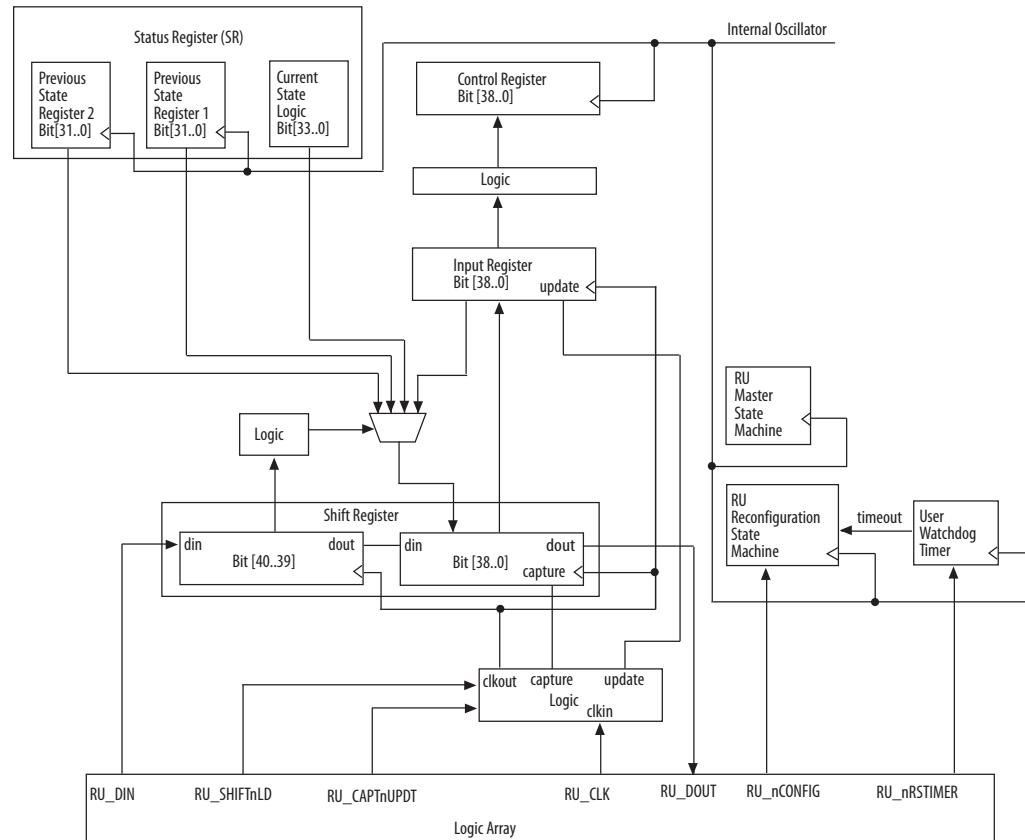
The remote system upgrade feature detects errors in the following sequence:

1. After power-up, the device samples the CONFIG_SEL pin to determine which application configuration image to load. The CONFIG_SEL pin setting can be overwritten by the input register of the remote system upgrade circuitry for the subsequent reconfiguration.

2. If an error occurs, the remote system upgrade feature reverts by loading the other application configuration image. These errors cause the remote system upgrade feature to load another application configuration image:

   - Internal CRC error
   - User watchdog timer time-out

3. Once the revert configuration completes and the device is in user mode, you can use the remote system upgrade circuitry to query the cause of error and which application image failed.

4. If a second error occurs, the device waits for a reconfiguration source. If the **Auto-restart configuration after error** is enabled, the device will reconfigure without waiting for any reconfiguration source.

5. Reconfiguration is triggered by the following actions:

   - Driving the nSTATUS low externally.
   - Driving the nCONFIG low externally.
   - Driving RU_nCONFIG low.

Remote System Upgrade Circuitry

Figure 2-4: Remote System Upgrade Circuitry

The remote system upgrade circuitry does the following functions:

- Tracks the current state of configuration
- Monitors all reconfiguration sources
- Provides access to set up the application configuration image
- Returns the device to fallback configuration if an error occurs
- Provides access to the information on the failed application configuration image

Remote System Upgrade Circuitry Signals

Table 2-8: Remote System Upgrade Circuitry Signals for MAX 10 Devices

Core Signal Name	Logical Signal Name	Input/Output	Description
RU_DIN	regin	Input	Use this signal to write data to the shift register on the rising edge of RU_CLK. To load data to the shift register, assert RU_SHIFTnLD.
RU_DOUT	regout	Output	Use this signal to get output data from the shift register. Data is clocked out on each rising edge of RU_CLK if RU_SHIFTnLD is asserted.
RU_nRSTIMER	rsttimer	Input	• Use this signal to reset the user watchdog timer. A falling edge of this signal triggers a reset of the user watchdog timer. • To reset the timer, pulse the RU_nRSTIMER signal for a minimum of 250 ns.
RU_nCONFIG	rconfig	Input	Use this signal to reconfigure the device. Driving this signal low triggers the device to reconfigure if you enable the remote system upgrade feature.
RU_CLK	clk	Input	The clock to the remote system upgrade circuitry. All registers in this clock domain are enabled in user mode if you enable the remote system upgrade. Shift register and input register are positive edge flip-flops.
RU_SHIFTnLD	shiftnld	Input	Control signals that determine the mode of remote system upgrade circuitry.
RU_CAPTnUPDT	captnupdt	Input	• When RU_SHIFTnLD is driven low and RU_CAPTnUPDT is driven low, the input register is loaded with the contents of the shift register on the rising edge of RU_CLK. • When RU_SHIFTnLD is driven low and RU_CAPTnUPDT is driven high, the shift register captures values from the input_cs_ps module on the rising edge of RU_CLK. • When RU_SHIFTnLD is driven high, the RU_CAPTnUPDT will be ignored and the shift register shifts data on each rising edge of RU_CLK.

Related Information

MAX 10 Device Datasheet

Provides more information about Remote System Upgrade timing specifications.

Remote System Upgrade Circuitry Input Control

The remote system upgrade circuitry has three modes of operation.

- Update—loads the values in the shift register into the input register.
- Capture—loads the shift register with data to be shifted out.
- Shift—shifts out data to the user logic.

Table 2-9: Control Inputs to the Remote System Upgrade Circuitry

Remote System Upgrade Circuitry Control Inputs				Operation Mode	Input Settings for Registers	
RU_SHIFTnLD	RU_CAPTnUPDT	Shift register [40]	Shift register [39]		Shift Register[38:0]	Input Register[38:0]
0	0	Don't Care	Don't Care	Update	Shift Register [38:0]	Shift Register [38:0]
0	1	0	0	Capture	Current State	Input Register[38:0]
0	1	0	1	Capture	{8'b0, Previous State Application1}	Input Register[38:0]
0	1	1	0	Capture	{8'b0, Previous State Application2}	Input Register[38:0]
0	1	1	1	Capture	Input Register[38:0]	Input Register[38:0]
1	Don't Care	Don't Care	Don't Care	Shift	{ru_din, Shift Register [38:1]}	Input Register[38:0]

The following shows examples of driving the control inputs in the remote system upgrade circuitry:

- When you drive RU_SHIFTnLD high to 1'b1, the shift register shifts data on each rising edge of RU_CLK and RU_CAPTnUPDT has no function.
- When you drive both RU_SHIFTnLD and RU_CAPTnUPDT low to 1'b0, the input register is loaded with the contents of the shift register on the rising edge of RU_CLK.
- When you drive RU_SHIFTnLD low to 1'b0 and RU_CAPTnUPDT high to 1'b1, the shift register captures values on the rising edge of RU_DCLK.

Remote System Upgrade Input Register

Table 2-10: Remote System Upgrade Input Register for MAX 10 Devices

Bits	Name	Description
38:14	Reserved	Reserved—set to `0`.
13	`ru_config_sel`	• 0: Load configuration image 0 • 1: Load configuration image 1 This bit will only work if the `ru_config_sel_overwrite` bit is set to `1`.
12	`ru_config_sel_overwrite`	• 0: Disable overwrite `CONFIG_SEL` pin • 1: Enable overwrite `CONFIG_SEL` pin
11:0	Reserved	Reserved—set to `0`.

Remote System Upgrade Status Registers

Table 2-11: Remote System Upgrade Status Register—Current State Logic Bit for MAX 10 Devices

Bits	Name	Description
33:30	`msm_cs`	The current state of the master state machine (MSM).
29	`ru_wd_en`	The current state of the enabled user watchdog timer. The default state is active high.
28:0	`wd_timeout_value`	The current, entire 29-bit watchdog time-out value.

Table 2-12: Remote System Upgrade Status Register—Previous State Bit for MAX 10 Devices

Bits	Name	Description
31	`nconfig`	An active high field that describes the reconfiguration sources which caused the MAX 10 device to leave the previous application configuration. In the event of a tie, the higher bit order takes precedence. For example, if the `nconfig` and the `ru_nconfig` triggered at the same time, the `nconfig` takes precedence over the `ru_nconfig`.
30	`crcerror`	
29	`nstatus`	
28	`wdtimer`	
27:26	Reserved	Reserved—set to `0`.
25:22	`msm_cs`	The state of the MSM when a reconfiguration event occurred. The reconfiguration will cause the device to leave the previous application configuration.
21:0	Reserved	Reserved—set to `0`.

Related Information

[**Altera Dual Configuration IP Core Avalon-MM Address Map**](#) on page 5-1

Master State Machine

The master state machine (MSM) tracks current configuration mode and enables the user watchdog timer.

Table 2-13: Remote System Upgrade Master State Machine Current State Descriptions for MAX 10 Devices

`msm_cs` Values	State Description
0010	Image 0 is being loaded.
0011	Image 1 is being loaded after a revert in application image happens.
0100	Image 1 is being loaded.
0101	Image 0 is being loaded after a revert in application image happens.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. You can use the timer to detect functional errors when an application configuration is successfully loaded into the device.

The counter is 29 bits wide and has a maximum count value of 2^{29}. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the user watchdog timer internal oscillator. Depending on the counter and the internal oscillator of the device, you can set the cycle time from 9ms to 244s.

Figure 2-5: Watchdog Timer Formula for MAX 10 Devices

$$\text{Watchdog timer time-out (seconds)} = \frac{\text{Watchdog timer value (decimal)}}{\text{Watchdog timer frequency}}$$

The timer begins counting as soon as the application configuration enters user mode. When the timer expires, the remote system upgrade circuitry generates a time-out signal, updates the status register, and triggers the loading of the revert configuration image. To reset the timer, pulse the `RU_NRSTIMER` for a minimum of 250 ns.

When you enable the watchdog timer, the setting will apply to all images, all images should contain the soft logic configuration to reset the timer. Application Configuration will reset the control block registers.

Related Information
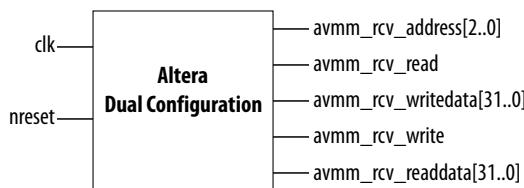
- **User Watchdog Internal Circuitry Timing Specifications**
  Provides more information about the user watchdog frequency.
- **Initialization Configuration Bits** on page 2-7

Altera Dual Configuration IP Core

The Altera Dual Configuration IP core offers the following capabilities through Avalon-MM interface:

- Asserts `RU_nCONFIG` to trigger reconfiguration.
- Asserts `RU_nRSTIMER` to reset watchdog timer if the watchdog timer is enabled.
- Writes configuration setting to the input register of the remote system upgrade circuitry.
- Reads information from the remote system upgrade circuitry.

Figure 2-6: Altera Dual Configuration IP Core Block Diagram

Related Information

- **Altera Dual Configuration IP Core Avalon-MM Address Map** on page 5-1
- **Avalon Interface Specifications**
  Provides more information about the Avalon-MM interface specifications applied in Altera Dual Configuration IP Core.
- **Instantiating the Altera Dual Configuration IP Core** on page 4-2
- **Altera Dual Configuration IP Core References** on page 5-1
- **Remote System Upgrade** on page 2-9
- **AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor**
  Provides reference design for remote system upgrade in MAX 10 FPGA devices.
- **I2C Remote System Update Example**
  This example demonstrates a remote system upgrade using the I2C protocol.

Configuration Design Security

The MAX 10 design security feature supports the following capabilities:

- Encryption—Built-in encryption standard (AES) to support 128-bit key industry-standard design security algorithm
- Chip ID—Unique device identification
- JTAG secure mode—limits access to JTAG instructions
- Verify Protect—allows optional disabling of CFM content read-back

AES Encryption Protection

The MAX 10 design security feature provides the following security protection for your designs:

- Security against copying—the non-volatile key is securely stored in the MAX 10 devices and cannot be read through any interface. Without this key, attacker will not be able to decrypt the encrypted configuration image.
- Security against reverse engineering—reverse engineering from an encrypted configuration file is very difficult and time consuming because the file requires decryption.
- Security against tampering—after you enable the JTAG Secure and Encrypted POF (EPOF) only, the MAX 10 device can only accept configuration files encrypted with the same key. Additionally, configuration through the JTAG interface is blocked.

Related Information

Generating .pof using Convert Programming Files on page 3-7

Encryption and Decryption

MAX 10 supports AES encryption. Programming bitstream is encrypted based on the encryption key that is specified by you. In MAX 10 devices, the key is part of the ICB settings stored in the internal flash. Hence, the key will be non-volatile but you can clear/delete the key by a full chip erase the device.

When you use compression with encryption, the configuration file is first compressed, and then encrypted using the Quartus Prime software. During configuration, the device first decrypts, and then decompresses the configuration file.

The header and I/O configuration shift register (IOCSR) data will not be encrypted. The decryption block is activated after the IOCSR chain is programmed. The decryption block only decrypts core data and postamble.

Related Information

JTAG Instruction Availability on page 2-19

Unique Chip ID

Unique chip ID provides the following features:

- Identifies your device in your design as part of a security feature to protect your design from an unauthorized device.
- Provides non-volatile 64-bits unique ID for each MAX 10 device with write protection.
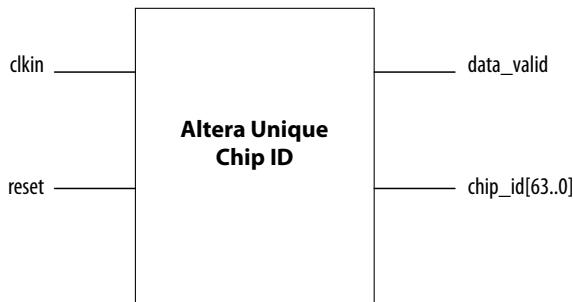
You can use the Altera Unique Chip ID IP core to acquire the chip ID of your MAX 10 device.

Related Information

- **Altera Unique Chip ID IP Core** on page 4-1
- **Altera Unique Chip ID IP Core Ports** on page 6-1

Altera Unique Chip ID IP Core

Figure 2-7: Altera Unique Chip ID IP Core Block Diagram

At the initial state, the `data_valid` signal is low because no data is read from the unique chip ID block. After feeding a clock signal to the `clkin` input port, the Altera Unique Chip ID IP core begins to acquire the chip ID of your device through the unique chip ID block. After acquiring the chip ID of your device, the Altera Unique Chip ID IP core asserts the `data_valid` signal to indicate that the chip ID value at the output port is ready for retrieval.

The operation repeats only when you provide another clock signal when the `data_valid` signal is low. If the `data_valid` signal is high when you provide another clock signal, the operation stops because the `chip_id[63..0]` output holds the chip ID of your device.

A minimum of 67 clock cycles are required for the `data_valid` signal to go high.

The `chip_id[63:0]` output port holds the value of chip ID of your device until you reconfigure the device or reset the Altera Unique Chip ID IP core.

JTAG Secure Mode

In JTAG Secure mode, the device only allows mandatory IEEE 1149.1 JTAG instructions to be exercised.

You can enable the JTAG secure when generating the **.pof** in the **Convert Programming Files**. To exit JTAG secure mode, issue the `UNLOCK` JTAG instruction. The `LOCK` JTAG instruction puts the device in the JTAG secure mode again. The `LOCK` and `UNLOCK` JTAG instructions can only be issued through the JTAG core access. Refer to **Table 2-15** for list of available instructions.

Related Information

JTAG Secure Mode Instructions

Table 2-14: JTAG Secure Mode Instructions for MAX 10 Devices

JTAG Instruction	Instruction Code	Description
LOCK	10 0000 0010	• Activates the JTAG secure mode. • Blocks access from both external pins and core to JTAG.
UNLOCK	10 0000 1000	Deactivates the JTAG secure mode.

Verify Protect

Verify Protect is a security feature to enhance CFM security. When you enable the **Verify Protect**, only program and erase operation are allowed on the CFM. This capability protects the CFM contents from being copied.

You can turn on the **Verify Protect** feature when converting the **.sof** file to **.pof** file in the Quartus Prime Convert Programming File tool.

Related Information

- **Configuration Flash Memory Permissions** on page 2-20
- **Generating .pof using Convert Programming Files** on page 3-7

JTAG Instruction Availability

Table 2-15: JTAG Instruction Availability Based on JTAG Secure Mode and Encryption Settings

JTAG Secure Mode	Encryption	Description
Disabled	Disabled	All JTAG Instructions enabled
Disabled	Enabled	All JTAG Instructions are enabled except: • CONFIGURE
Enabled	Disabled	All non-mandatory IEEE 1149.1 JTAG instructions are disabled except:
Enabled	Enabled	• SAMPLE/PRELOAD • BYPASS • EXTEST • IDCODE • UNLOCK • LOCK

Related Information

- **JTAG Secure Mode** on page 2-18
- **MAX 10 JTAG Secure Design Example** on page 3-20
- **JTAG Secure Design Example**

- **Encryption and Decryption** on page 2-17

Configuration Flash Memory Permissions

The JTAG secure mode and verify protect features determines the CFM operation permission.. The table list the operations permitted based on the security settings.

Table 2-16: CFM Permissions for MAX 10 Devices

Operation	JTAG Secure Mode Disabled		JTAG Secure Mode Enabled	
	Verify Protect Disabled	Verify Protect Enabled	Verify Protect Disabled	Verify Protect Enabled
ISP through core	Illegal operation	Illegal operation	Illegal operation	Illegal operation
ISP through JTAG pins	Full access	Program and erase only	No access	No access
Real-time ISP through core	Full access	Program and erase only	No access	No access
Real-time ISP through JTAG pins	Full access	Program and erase only	No access	No access
UFM interface through core[5]	Full access	Full access	Full access	Full access

Related Information

- **JTAG Secure Mode** on page 2-18
- **MAX 10 JTAG Secure Design Example** on page 3-20
- **JTAG Secure Design Example**
- **Verify Protect** on page 2-19
- **Generating .pof using Convert Programming Files** on page 3-7

SEU Mitigation and Configuration Error Detection

The dedicated circuitry built in MAX 10 devices consists of an error detection cyclic redundancy check (EDCRC) feature. You can use this feature to mitigate single-event upset (SEU) or soft errors.

The hardened on-chip EDCRC circuitry allows you to perform the following operations without any impact on the fitting of the device:

- Auto-detection of cyclic redundancy check (CRC) errors during configuration.
- Identification of SEU in user mode with the optional CRC error detection.
- Testing of error detection by error detection verification through the JTAG interface.

Related Information

- **Verifying Error Detection Functionality** on page 3-10
- **Enabling Error Detection** on page 3-11

---

[5] The UFM interface through core is available if you select the dual compressed image mode.

Configuration Error Detection

In configuration mode, a frame-based CRC is stored in the configuration data and contains the CRC value for each data frame.

During configuration, the MAX 10 device calculates the CRC value based on the frame of data that is received and compares it against the frame CRC value in the data stream. Configuration continues until the device detects an error or when all the values are calculated.

For MAX 10 devices, the CRC is computed by the Quartus Prime software and downloaded into the device as part of the configuration bit stream. These devices store the CRC in the 32-bit storage register at the end of the configuration mode.

User Mode Error Detection

SEUs are changes in a CRAM bit state due to an ionizing particle. MAX 10 devices have built-in error detection circuitry to detect data corruption in the CRAM cells.
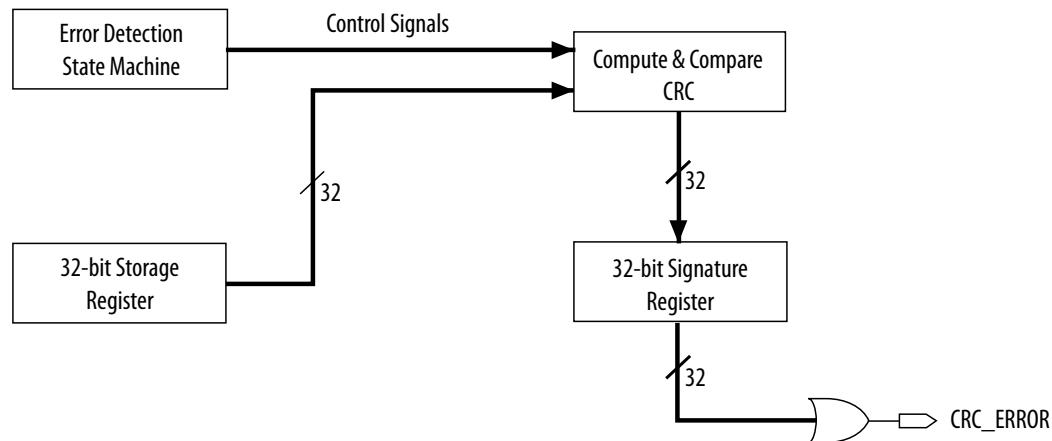
This error detection capability continuously computes the CRC of the configured CRAM bits. The CRC of the contents of the device are compared with the pre-calculated CRC value obtained at the end of the configuration. If the CRC values match, there is no error in the current configuration CRAM bits. The process of error detection continues until the device is reset—by setting nCONFIG to low.

The error detection circuitry in MAX 10 device uses a 32-bit CRC IEEE Std. 802 and a 32-bit polynomial as the CRC generator. Therefore, the device performs a single 32-bit CRC calculation. If an SEU does not occur, the resulting 32-bit signature value is 0x000000, which results in a 0 on the output signal CRC_ERROR. If an SEU occurs in the device, the resulting signature value is non-zero and the CRC_ERROR output signal is 1. You must decide whether to reconfigure the FPGA by strobing the nCONFIG pin low or ignore the error.

Error Detection Block

Figure 2-8: Error Detection Block Diagram

Error detection block diagram including the two related 32-bit registers—the signature register and the storage register.

There are two sets of 32-bit registers in the error detection circuitry that store the computed CRC signature and pre-calculated CRC value. A non-zero value on the signature register causes the CRC_ERROR pin to go high.

Table 2-17: Error Detection Registers for MAX 10 Devices

Register	Description
32-bit signature register	This register contains the CRC signature. The signature register contains the result of the user mode calculated CRC value compared against the pre-calculated CRC value. If no errors are detected, the signature register is all zeroes. A non-zero signature register indicates an error in the configuration CRAM contents. The CRC_ERROR signal is derived from the contents of this register.
32-bit storage register	This register is loaded with the 32-bit pre-computed CRC signature at the end of the configuration stage. The signature is then loaded into the 32-bit Compute and Compare CRC block during user mode to calculate the CRC error. This register forms a 32-bit scan chain during execution of the CHANGE_EDREG JTAG instruction. The CHANGE_EDREG JTAG instruction can change the content of the storage register. Therefore, the functionality of the error detection CRC circuitry is checked in-system by executing the instruction to inject an error during the operation. The operation of the device is not halted when issuing the CHANGE_EDREG JTAG instruction.

CHANGE_EDREG JTAG Instruction

Table 2-18: CHANGE_EDREG JTAG Instruction Description

JTAG Instruction	Instruction Code	Description
CHANGE_EDREG	00 0001 0101	This instruction connects the 32-bit CRC storage register between TDI and TDO. Any precomputed CRC is loaded into the CRC storage register to test the operation of the error detection CRC circuitry at the CRC_ERROR pin.

Error Detection Timing

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

The CRC_ERROR pin will remain low until the error detection circuitry has detected a corrupted bit in the previous CRC calculation. After the pin goes high, it remains high during the next CRC calculation. This pin does not log the previous CRC calculation. If the new CRC calculation does not contain any corrupted bits, the CRC_ERROR pin is driven low. The error detection runs until the device is reset.

The error detection circuitry is clocked by an internal configuration oscillator with a divisor that sets the maximum frequency. The CRC calculation time depends on the device and the error detection clock frequency.

Related Information
Enabling Error Detection on page 3-11

Error Detection Frequency

You can set a lower clock frequency by specifying a division factor in the Quartus Prime software.

Table 2-19: Minimum and Maximum Error Detection Frequencies for MAX 10 Devices

Device	Error Detection Frequency	Maximum Error Detection Frequency (MHz)	Minimum Error Detection Frequency (kHz)	Valid Values for n
10M02				
10M04				
10M08	$55\ \text{MHz}/2^n$ to $116\ \text{MHz}/2^n$	58	214.8	2, 3, 4, 5, 6, 7, 8
10M16				
10M25				
10M40	$35\ \text{MHz}/2^n$ to $77\ \text{MHz}/2^n$	38.5	136.7	
10M50				

Cyclic Redundancy Check Calculation Timing

Table 2-20: Cyclic Redundancy Check Calculation Time for MAX 10 Devices

Device	Divisor Value (n = 2)	
	Minimum Time (ms)	Maximum Time (ms)
10M02	2	6.6
10M04	6	15.7
10M08	6	15.7
10M16	10	25.5
10M25	14	34.7
10M40	43	106.7
10M50	43	106.7

Figure 2-9: CRC Calculation Formula

You can use this formula to calculate the CRC calculation time for divisor other than 2.

$$CRC\ Calculation\ Time_{Divisor\ n} = CRC\ Calculation\ Time_{Divisor\ 2} \times \frac{n}{2}$$

Example 2-1: CRC Calcualtion Example

For 10M16 device with divisor value of 256:

Minimum CRC calculation time for divisor 256 = 10 x (256/2) = 1280 ms

Recovering from CRC Errors

The system that MAX 10 resides in must control device reconfiguration. After detecting an error on the `CRC_ERROR` pin, strobing the `nCONFIG` pin low directs the system to perform reconfiguration at a time when it is safe for the system to reconfigure the MAX 10 device.

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

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

Configuration Data Compression

MAX 10 devices can receive compressed configuration bitstream and decompress the data in real-time during configuration. This feature helps to reduce the configuration image size stored in the CFM. Data indicates that compression typically reduces the configuration file size by at least 30% depending on the design.
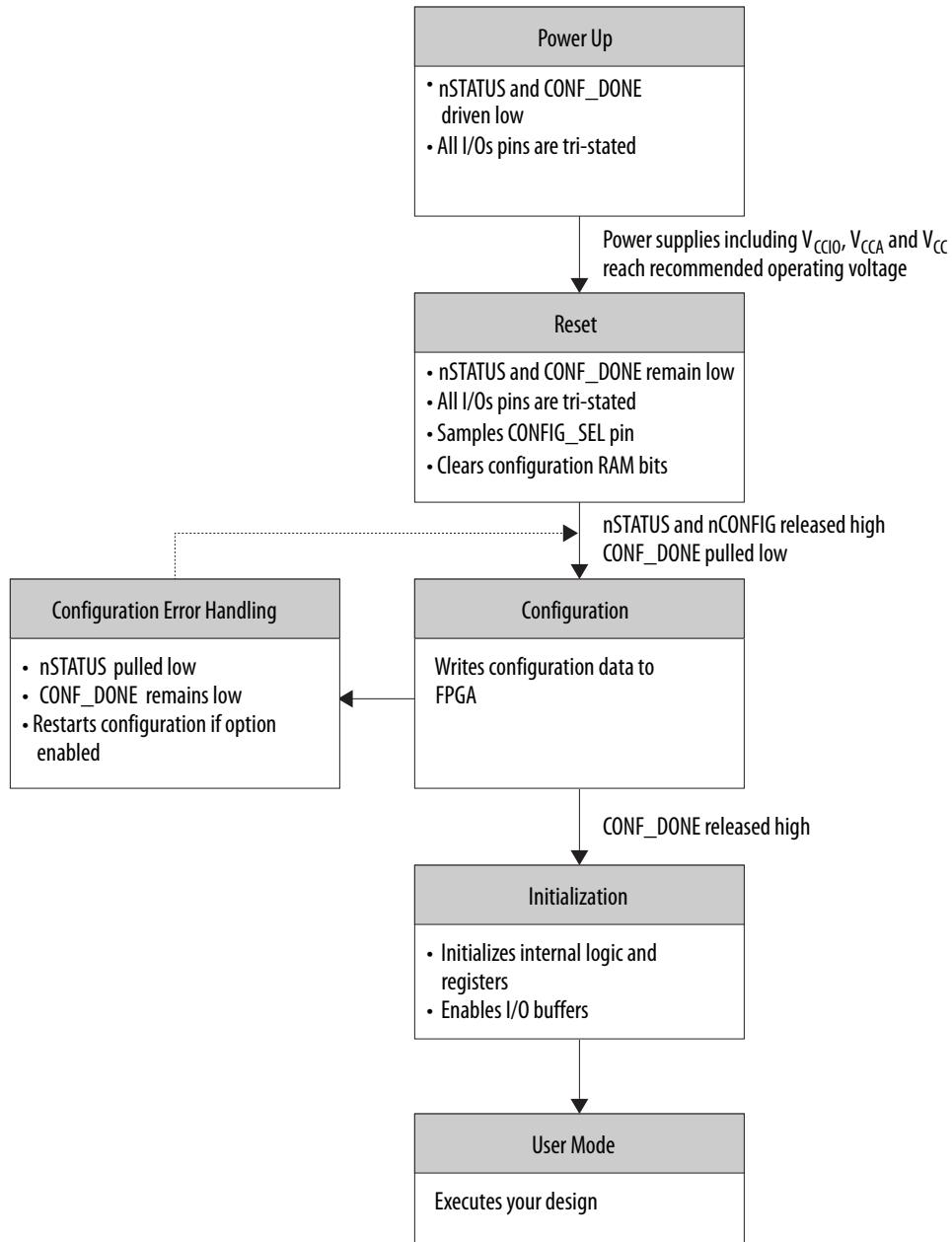
Related Information

Configuration Details

Configuration Sequence

Figure 2-10: Configuration Sequence for MAX 10 Devices

```
                    ┌─────────────────────────────┐
                    │          Power Up            │
                    ├─────────────────────────────┤
                    │ • nSTATUS and CONF_DONE      │
                    │   driven low                 │
                    │ • All I/Os pins are tri-stated│
                    └─────────────────────────────┘
                             │  Power supplies including V_CCIO, V_CCA and V_CC
                             │  reach recommended operating voltage
                             ▼
                    ┌─────────────────────────────┐
                    │           Reset              │
                    ├─────────────────────────────┤
                    │ • nSTATUS and CONF_DONE remain low │
                    │ • All I/Os pins are tri-stated│
                    │ • Samples CONFIG_SEL pin     │
                    │ • Clears configuration RAM bits│
                    └─────────────────────────────┘
                             │  nSTATUS and nCONFIG released high
                             │  CONF_DONE pulled low
  ┌──────────────────────────┐        ▼
  │ Configuration Error Handling │   ┌─────────────────────────────┐
  ├──────────────────────────┤    │        Configuration         │
  │ • nSTATUS pulled low      │◄───├─────────────────────────────┤
  │ • CONF_DONE remains low   │    │ Writes configuration data to │
  │ • Restarts configuration if option│ │ FPGA                      │
  │   enabled                 │    └─────────────────────────────┘
  └──────────────────────────┘             │  CONF_DONE released high
                                           ▼
                    ┌─────────────────────────────┐
                    │        Initialization        │
                    ├─────────────────────────────┤
                    │ • Initializes internal logic and │
                    │   registers                  │
                    │ • Enables I/O buffers        │
                    └─────────────────────────────┘
                                 │
                                 ▼
                    ┌─────────────────────────────┐
                    │         User Mode            │
                    ├─────────────────────────────┤
                    │ Executes your design         │
                    └─────────────────────────────┘
```

You can initiate reconfiguration by pulling the nCONFIG pin low to at least the minimum $t_{RU\_nCONFIG}$ low-pulse width. When this pin is pulled low, the nSTATUS and CONF_DONE pins are pulled low and all I/O pins are either tied to an internal weak pull-up or tri-stated based on the ICB settings.

Related Information

Generating .pof using Convert Programming Files on page 3-7
Provides more information about how to set the weak pull-up during configuration.

Power-up

If you power-up a device from the power-down state, you need to power the V_{CCIO} for bank 1B (bank 1 for 10M02 devices), bank 8 and the core to the appropriate level for the device to exit POR. The MAX 10 device enters the configuration stage after exiting the power-up stage with a small POR delay.

Related Information

- **MAX 10 Power Management User Guide**
  Provides more information about power supply modes in MAX 10 devices
- **MAX 10 Device Datasheet**
  Provides more information about the ramp-up time specifications.
- **MAX 10 FPGA Device Family Pin Connection Guideline**
  Provides more information about configuration pin connections.

POR Monitored Voltage Rails for Single-supply and Dual-supply MAX 10 Devices

To begin configuration, the required voltages must be powered up to the appropriate voltage levels as shown in the following table. The V_{CCIO} for bank 1B (bank 1 for 10M02 devices) and bank 8 must be powered up to a voltage between 1.5V – 3.3V during configuration.

Table 2-21: POR Monitored Voltage Rails for Single-supply and Dual-supply MAX 10 Devices

There is no power-up sequence required when powering-up the voltages.

Power Supply Device Options	Power Supply Monitored by POR
Single-supply	Regulated V_{CC_ONE}
	V_{CCA}
	V_{CCIO} bank 1B[6] and bank 8
Dual-supply	V_{CC}
	V_{CCA}
	V_{CCIO} bank 1B[6] and bank 8

---

[6] Bank 1 for 10M02 devices

Monitored Power Supplies Ramp Time Requirement for MAX 10 Devices

Figure 2-11: Monitored Power Supplies Ramp Time Requirement Diagram for MAX 10 Devices
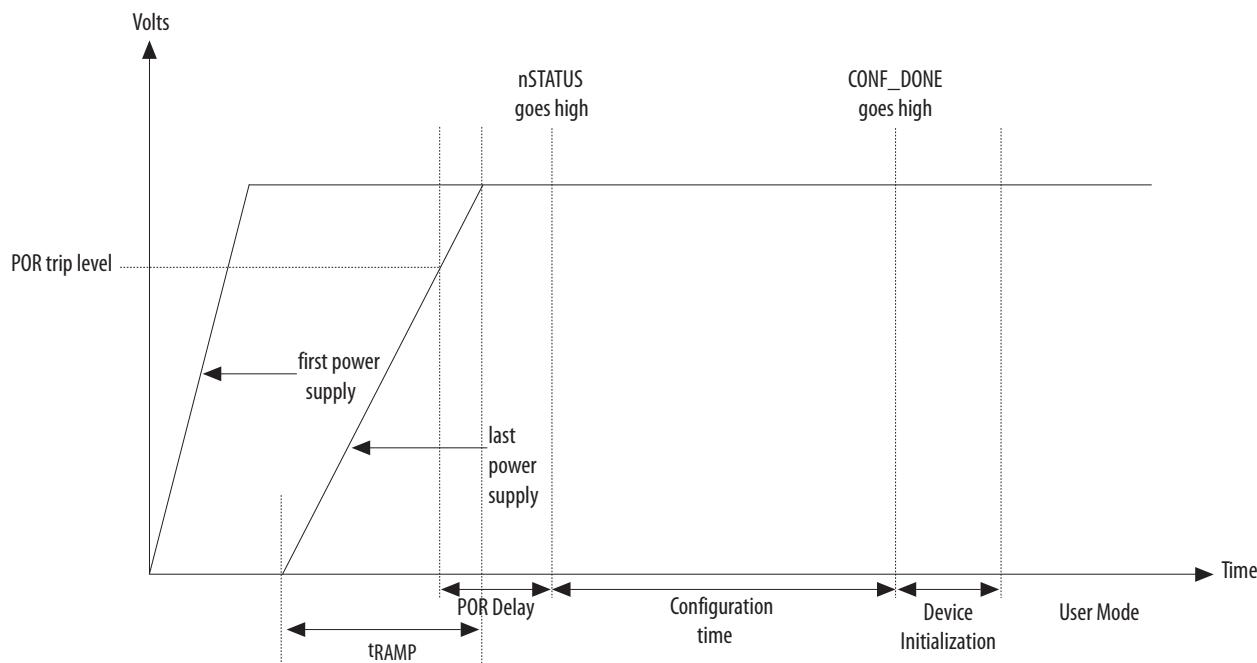
Table 2-22: Monitored Power Supplies Ramp Time Requirement for MAX 10 Devices

Symbol	Parameter	Minimum	Maximum	Unit
t_{RAMP}	Power Supply Ramp Time[7]	—[8]	10	ms

Configuration

During configuration, configuration data is read from the internal flash and written to the CRAM.

Configuration Error Handling

To restart configuration automatically, turn on the **Auto-restart configuration after error** option in the **General** page of the **Device and Pin Options** dialog box in the Quartus Prime software.

If you do not turn on this option, you can monitor the nSTATUS pin to detect errors. To restart configuration, pull the nCONFIG pin low for at least the duration of $t_{RU\_nCONFIG}$.

---

[7] Ensure that all V_{CCIO} power supply reaches full rail before configuration completes. See **Internal Configuration Time** on page 2-8.

[8] There is no absolute minimum value for the ramp rate requirement. Altera characterized the minimum t_{RAMP} of 200μs.

Initialization

The initialization sequence begins after the CONF_DONE pin goes high. The initialization clock source is from the internal oscillator and the MAX 10 device will receive enough clock cycles for proper initialization.

User Mode

After the initialization completes, your design starts executing. The user I/O pins will then function as specified by your design.

MAX 10 Configuration Pins

All configuration pins and JTAG pins in MAX 10 devices are dual-purpose pins. The configuration pins function as configuration pins prior to user mode. When the device is in user mode, they function as user I/O pins or remain as configuration pins.

Table 2-23: Configuration Pin Summary for MAX 10 Devices

All pins are powered by V_{CCIO} Bank 1B (bank 1 for 10M02 devices) and 8.

Configuration Pin	Input/Output	Configuration Scheme
CRC_ERROR	Output only, open-drain	Optional, JTAG and internal configurations
CONFIG_SEL	Input only	Internal configuration
DEV_CLRn	Input only	Optional, JTAG and internal configurations
DEV_OE	Input only	Optional, JTAG and internal configurations
CONF_DONE	Bidirectional, open-drain	JTAG and internal configurations
nCONFIG	Input only	JTAG and internal configurations
nSTATUS	Bidirectional, open-drain	JTAG and internal configurations
JTAGEN	Input only	Optional, JTAG configuration
TCK	Input only	JTAG configuration
TDO	Output only	JTAG configuration
TMS	Input only	JTAG configuration
TDI	Input only	JTAG configuration

Related Information

Dual-Purpose Configuration Pins

Guidelines: Dual-Purpose Configuration Pin

To use configuration pins as user I/O pins in user mode, you have to adhere to the following guidelines.

Table 3-1: Dual-Purpose Configuration Pin Guidelines for MAX 10 Devices

Guidelines	Pins
Configuration pins during initialization: • Tri-state the external I/O driver and drive an external pull-up resistor[9] or • Use the external I/O driver to drive the pins to the state same as the external weak pull-up resistor	• `nCONFIG` • `nSTATUS` • `CONF_DONE`
JTAG pins: • If you intend to switch back and forth between user I/O pins and JTAG pin functions using the `JTAGEN` pin, all JTAG pins must be assigned as single-ended I/O pins or voltage-referenced I/O pins. Schmitt trigger input is the recommended input buffer. • JTAG pins cannot perform as JTAG pins in user mode if you assign any of the JTAG pin as a differential I/O pin. • You must use the JTAG pins as dedicated pins and not as user I/O pins during JTAG programming. • Do not toggle JTAG pin during the initialization stage. • Put the test access port (TAP) controller in reset state by driving the `TDI` and `TMS` pins high and `TCK` pin low for at least 5 clock cycles before the initialization.	• `TDO` • `TMS` • `TCK` • `TDI`

Attention: Assign all JTAG pins as single-ended I/O pins or voltage-referenced I/O pins if you enable JTAG pin sharing feature.

---

[9] If you intend to remove the external weak pull-up resistor, Altera recommends that you remove it after the device enters user mode.

---

now part of Intel

Related Information

- **MAX 10 FPGA Device Family Pin Connection Guidelines**
  Provides more information about recommended resistor values.
- **MAX 10 Configuration Pins** on page 2-28
- **JTAG Pins** on page 2-2

JTAG Pin Sharing Behavior

Table 3-2: JTAG Pin Sharing Behavior for MAX 10 Devices

Configuration Stage	JTAG Pin Sharing	JTAGEN Pin	JTAG Pins (TDO, TDI, TCK, TMS)
User mode	Disabled	User I/O pin	Dedicated JTAG pins.
	Enabled	Driven low	User I/O pins.
		Driven high	Dedicated JTAG pins.
Configuration	Don't Care	Not used	Dedicated JTAG pins.

Note: You have to set the pins according to **Table 3-1** and with correct pin direction (input, output or bidirectional) for the JTAG pins work correctly.

Enabling Dual-purpose Pin

To use the configuration and JTAG pins as user I/O in user mode, you must do the following in the Quartus Prime software:

1. On the **Assignments** menu, click **Device**.
2. Click **Device and Pin Options**.
3. Select the **General** tab of **Device and Pin Options**.
4. In the **General Options** list, do the following:

   - Check the **Enable JTAG pin sharing**.
   - Uncheck the **Enable nCONFIG, nSTATUS, and CONF_DONE pins**.

Related Information

- **MAX 10 Configuration Pins** on page 2-28
- **JTAG Pins** on page 2-2

Configuring MAX 10 Devices using JTAG Configuration

The Quartus Prime software generates a **.sof** that can used for JTAG configuration. You can directly configure the MAX 10 device by using a download cable with the Quartus Prime software programmer.

Alternatively, you can use the JAM Standard Test and Programming Language (STAPL) Format File (**.jam**) or JAM Byte Code File (**.jbc**) with other third-party programmer tools.
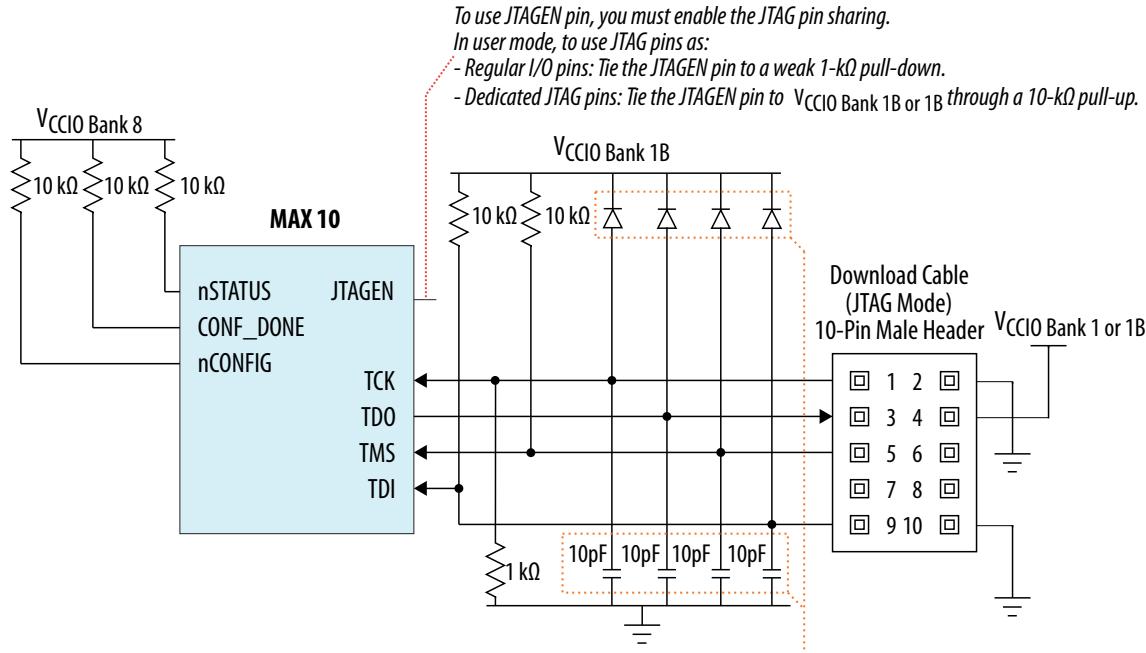
Related Information
AN 425: Using the Command-Line Jam STAPL Solution for Device Programming

JTAG Configuration Setup

Figure 3-1: Connection Setup for JTAG Single-Device Configuration using Download Cable
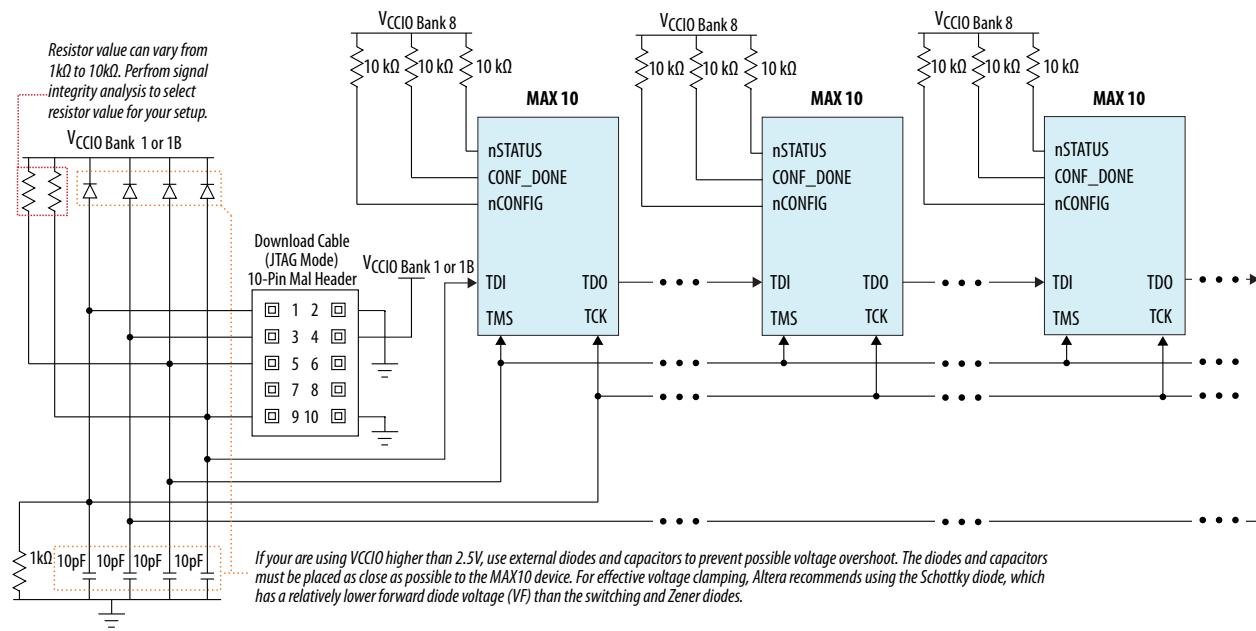
Connect to V_{CCIO} Bank 1 for 10M02 devices or V_{CCIO} Bank 1B for all other MAX 10 devices.

To use JTAGEN pin, you must enable the JTAG pin sharing.
In user mode, to use JTAG pins as:
- Regular I/O pins: Tie the JTAGEN pin to a weak 1-kΩ pull-down.
- Dedicated JTAG pins: Tie the JTAGEN pin to V_{CCIO} Bank 1B or 1B through a 10-kΩ pull-up.

If your are using VCCIO higher than 2.5V, use external diodes and capacitors to prevent possible voltage overshoot. The diodes and capacitors must be placed as close as possible to the MAX10 device. For effective voltage clamping, Altera recommends using the Schottky diode, which has a relatively lower forward diode voltage (VF) than the switching and Zener diodes.

Figure 3-2: Connection Setup for JTAG Multi-Device Configuration using Download Cable

Connect to V_{CCIO} Bank 1 for 10M02 devices or V_{CCIO} Bank 1B for all other MAX 10 devices.

To configure a device in a JTAG chain, the programming software sets the other devices to bypass mode. A device in bypass mode transfers the programming data from the `TDI` pin to the `TDO` pin through a single bypass register. The configuration data is available on the `TDO` pin one clock cycle later.

The Quartus Prime software uses the `CONF_DONE` pin to verify the completion of the configuration process through the JTAG port:

- `CONF_DONE` pin is low—indicates that the configuration has failed.
- `CONF_DONE` pin is high—indicates that the configuration was successful.

After the configuration data is transmitted serially using the JTAG `TDI` port, the `TCK` port is clocked to perform device initialization.

Voltage Overshoot Prevention

To prevent voltage overshoot, power up the download cable to 2.5 V when VCCIO of the JTAG pins are 2.5 V to 3.3 V. Tie the `TCK` pin to ground. If the VCCIO of the JTAG pins are using 1.5 V or 1.8 V, the download cable should be powered by the same VCCIO. For single-supply device which has to power-up the download cable within the range of 3.0 V to 3.3 V, Altera recommends you to add external capacitor or diode as shown in **Figure 3-1**.

JTAGEN

If you use the `JTAGEN` pin, Altera recommends the following settings:

- Once you entered user mode and JTAG pins are regular I/O pins—connect the `JTAGEN` pin to a weak pull-down (1 kΩ).
- Once you entered user mode and JTAG pins are dedicated pins—connect the `JTAGEN` pin to a weak pull-up (10 kΩ).

> **Note:** Altera recommends that you use three-pin header with a jumper or other switching mechanism to change the JTAG pins behavior.

ICB Settings in JTAG Configuration

The ICB settings are loaded into the device during **.pof** programming of the internal configuration scheme. The **.sof** used during JTAG configuration programs the CRAM only and does not contain ICB settings. The Quartus Prime Programmer will make the necessary setting based on the following:

- Device without ICB settings—ICB settings cleared from the internal flash or new device
- Device with ICB settings—prior ICB settings programmed using **.pof**

Devices without ICB Settings

For devices without ICB settings, the default value will be used. However, Quartus Prime Programmer disables the user watchdog timer by setting the Watchdog Timer Enable bit to 0. This step is to avoid any unwanted reconfiguration from occuring due to user watchdog timeout.

If the default ICB setting is undesired, you can program the desirable ICB setting first by using **.pof** programming before doing the JTAG configuration.

Devices with ICB Settings

For device with ICB settings, the settings will be preserved until the internal flash is erased. You can refer to the **.map** file to view the preserved ICB settings. JTAG configuration will follow the preserved ICB setting and behave accordingly.

If the prior ICB setting is undesired, you can program the desirable ICB setting first by using **.pof** programming before doing the JTAG configuration.

Related Information

- **.pof and ICB Settings** on page 3-6
- **Verify Protect** on page 2-19
- **JTAG Secure Mode** on page 2-18
- **ISP and Real-Time ISP Instructions** on page 2-6
- **User Watchdog Timer** on page 2-15
- **Generating .pof using Convert Programming Files** on page 3-7
  Provides more information about setting the ICB during .pof generation using Convert Programming File.

Configuring MAX 10 Devices using Internal Configuration

There are three main steps for using internal configuration scheme for MAX 10 devices:

1. Selecting the internal configuration scheme.
2. Generating the **.pof** with ICB settings
3. Programming the **.pof** into the internal flash

Related Information

- **Internal Configuration Modes** on page 2-2
- **Remote System Upgrade** on page 2-9

Selecting Internal Configuration Modes

To select the configuration mode, follow these steps:

1. Open the Quartus Prime software and load a project using a MAX 10 device.
2. On the **Assignments** menu, click **Settings**. The **Settings** dialog box appears.
3. In the **Category** list, select **Device**. The **Device** page appears.
4. Click **Device and Pin Options**.
5. In the **Device and Pin Options** dialog box, click the **Configuration** tab.
6. In the **Configuration Scheme** list, select **Internal Configuration**.
7. In the **Configuration Mode** list, select 1 out of 5 configuration modes available. The 10M02 devices has only 2 modes available.
8. Turn on **Generate compressed bitstreams** if needed.
9. Click **OK**.

.pof and ICB Settings

There are two methods which the **.pof** will be generated and setting-up the ICB. The internal configuration mode you selected will determine the corresponding method.

Table 3-3: .pof Generation and ICB Setting Method for Internal Configuration Modes

Internal Configuration Mode	ICB Setting	Description	.pof Generation Method to Use
Single Compressed Image	ICB can be set in **Device and Pin Options**	Quartus Prime software automatically generates the **.pof** during project compilation.	Auto-generated **.pof**[10]
Single Uncompressed Image			
Single Compressed Image with Memory Initialization.	ICB can be set during **Convert Programming Files** task.	You need to generate the **.pof** using **Convert Programming Files**.	Generating **.pof** using **Convert Programming Files**
Single Uncompressed Image with Memory Initialization			
Dual Compressed Images			

Auto-Generated .pof

To set the ICB for the auto-generated .pof, follow these steps:

1. On the **Assignments** menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Device**. The **Device** page appears.
3. Click **Device and Pin Options**.
4. In the **Device and Pin Options** dialog box, select the **Configuration** from the category pane.
5. Click the **Device Options …** button.
6. The **Max 10 Device Options** dialog box allows you to set the following:

---

[10] Auto-generated .pof does not allow encryption. To enable the encryption feature in Single Compressed and Single Uncompressed mode, use the Convert Programming Files method.

    a. User I/Os weak pull up during configuration.

    b. Verify Protect.

7. Click **OK** once setting is completed.

Generating .pof using Convert Programming Files

To convert **.sof** files to **.pof** files and to set the ICB, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.

2. Under **Output programming file**, select **Programmer Object File** (**.pof**) in the **Programming file type** list.

3. In the **Mode** list, select **Internal Configuration**.

4. To set the ICB settings, click **Option/Boot Info** and the **ICB setting** dialog box will appear. The **ICB setting** dialog box allows you to set the following:

    a. User I/Os weak pull up during configuration.

    b. Auto-recofigure from secondary image when initial image fails (enabled by default).

> **Note:** When you disable this feature, the device will always load the configuration image 0 without sampling the physical CONFIG_SEL pin. After successfully load the configuration image 0, you can switch between configuration image using the `config_sel_overwrite` bit of the input register. Refer to related information for details about Altera Dual Configuration IP core input register.

    c. Use secondary image ISP data as default setting when available.

    d. JTAG Secure.

> **Note:** The JTAG Secure feature will be disabled by default in Quartus Prime. If you are interested in using the JTAG Secure feature, contact Altera for support.

> **Caution:** MAX 10 FPGA device would become permanently locked if you enabled JTAG secure mode in the POF file and POF is encrypted with the wrong key. You must instantiate the internal JTAG interface for you unlock the external JTAG when the device is in JTAG Secure mode.

    e. Verify Protect.

    f. Allow encrypted POF only.

    g. Watchdog timer for dual configuration and watchdog timer value (Enabled after adding 2 **.sof** page with two designs that compiled with Dual Compressed Internal Images).

    h. User Flash Memory settings.

    i. RPD File Endianness

5. In the **File name** box, specify the file name for the programming file you want to create.

6. To generate a Memory Map File (**.map**), turn on **Create Memory Map File** (Auto generate output_file.map). The **.map** contains the address of the CFM and UFM with the ICB setting that you set through the **Option/Boot Info** option.

7. To generate a Raw Programming Data (**.rpd**), turn on **Create config data RPD** (Generate output_file_auto.rpd).

Separate Raw Programming Data (.rpd) for each configuration flash memory and user flash memory (CFM0, CFM1, UFM) section will be generated together for remote system upgrade purpose.

8. The **.sof** can be added through **Input files to convert** list and you can add up to two **.sof** files.

For remote system upgrade purpose, you can retain the original page 0 data in the **.pof**, and replaces page 1 data with new **.sof** file. To perform this, you must to add the **.pof** file in page 0, then add **.sof** page, then add the new **.sof** file to page 1.

9. After all settings are set, click **Generate** to generate related programming file.

Related Information

Encryption in Internal Configuration on page 3-18

Provides more information about internal configuration image loaded based on various settings.

Programming .pof into Internal Flash

You can use the Quartus Prime Programmer to program the **.pof** into the CFM through JTAG interface. The Quartus Prime Programmer also allows you to program the UFM part of the internal flash.

To program the **.pof** into the flash, follow these steps:

1. In the **Programmer** window, click **Hardware Setup** and select **USB Blaster**.
2. In the **Mode** list, select **JTAG**.
3. Click **Auto Detect** button on the left pane.
4. Select the device to be programmed, and click **Add File**.
5. Select the **.pof** to be programmed to the selected device.
6. There are several options in programming the internal flash:

   - To program any of the CFM0/CFM1/CFM2 only, select the corresponding CFM in the Program/ Configure column.
   - To program the UFM only, select the UFM in the Program/Configure column.
   - To program the CFM and UFM only, select the CFM and UFM in the Program/Configure column.

     Note: ICB setting is preserved in this option. However, before the programming starts, Quartus Prime Programmer will make sure the ICB setting in the device and the ICB setting in the selected **.pof** are the same. If the ICB settings are different, Quartus Prime Programmer will overwrite the ICB setting.

   - To program the whole internal flash including the ICB settings, select the **<yourpoffile.pof>** in the Program/Configure column.
7. To enable the real-time ISP mode, turn-on the **Enable real-time ISP to allow background program-ming**.
8. After all settings are set, click **Start** to start programming.

Accessing Remote System Upgrade through User Logic

The following example shows how the input and output ports of a WYSIWYG atom are defined in the MAX 10 device.

```
fiftyfivenm_rublock <rublock_name>
(
    .clk(<clock source>),
    .shiftnld(<shiftnld source>),
    .captnupdt(<captnupdt source>),
    .regin(<regin input source from the core>),
    .rsttimer(<input signal to reset the watchdog timer>),
    .rconfig(<input signal to initiate configuration>),
    .regout(<data output destination to core>)
);
```

```
defparam <rublock_name>.sim_init_config = <initial configuration for simulation
only>;
defparam <rublock_name>.sim_init_watchdog_value = <initial watchdog value for
simulation only>;
defparam <rublock_name>.sim_init_config = <initial status register value for
simulation only>;
```

Table 3-4: Port Definitions

Port	Input/Output	Definition
`<rublock_name>`	-	Unique identifier for the RSU Block. This is any identifier name which is legal for the given description language (e.g. Verilog, VHDL, AHDL, etc.). This field is required
`.clk(<clock source>)`	Input	This signal designates the clock input of this cell. All operation of this cell are with respect to the rising edge of this clock. Whether it is the loading of the data into the cell or data out of the cell, it always occurs on the rising edge. This field is required
`.shiftnld(<shiftnld source>)`	Input	This signal is an input into the remote system upgrade block. If shiftnld = 1, then data gets shifted from the internal shift registers to the regout at each rising edge of clk and it gets shifted into the internal shift registers from regin. This field is required
`.captnupdt(<captnupdt source>)`	Input	This signal is an input into the remote system upgrade 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>)`	Input	This signal is an input into the remote system upgrade block for all data being loaded into the core. The data is shifted into the internal registers at the rising edge of clk. This field is required
`.rsttimer(<input signal to reset the watchdog timer>)`	Input	This signal is an input into the watchdog timer of the remote update block. When this is high, it resets the watchdog timer. This field is required
`.rconfig(<input signal to initiate configuration>)`	Input	This signal is an input into the configuration section of the remote update block. When this signal goes high, it initiates a reconfiguration. This field is required
`.regout(<data output destination to core>)`	Output	This is a 1 bit output which is the output of the internal shift register updated every rising edge of .clk. The data coming out depends on the control signals. This field is required.

Related Information

- **Altera Dual Configuration IP Core References** on page 5-1
- **Remote System Upgrade** on page 2-9

3-10

- **AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor**
  Provides reference design for remote system upgrade in MAX 10 FPGA devices.
- **I2C Remote System Update Example**
  This example demonstrates a remote system upgrade using the I2C protocol.

Error Detection

Verifying Error Detection Functionality

You can inject a soft error by changing the 32-bit CRC storage register in the CRC circuitry. After verifying the failure induced, you can restore the 32-bit CRC value to the correct CRC value using the same instruction and inserting the correct value. Be sure to read out the correct value before updating it with a known bad value.

In user mode, MAX 10 devices support the CHANGE_EDREG JTAG instruction, which allows you to write to the 32-bit storage register. You can use **.jam** to automate the testing and verification process. You can only execute this instruction when the device is in user mode. This instruction enables you to dynamically verify the CRC functionality in-system without having to reconfigure the device. You can then switch to use the CRC circuit to check for real errors induced by an SEU.

After the test completes, to clear the CRC error and restore the original CRC value, power cycle the device or follow these steps:

1. After the configuration completes, use CHANGE_EDREG JTAG instruction to shift out the correct precomputed CRC value and load the wrong CRC value to the CRC storage register. When an error is detected, the CRC_ERROR pin will be asserted.
2. Use CHANGE_EDREG JTAG instruction to shift in the correct precomputed CRC value. The CRC_ERROR pin is de-asserted to show that the error detection CRC circuitry is working.

Example 3-1: JAM File

```
'EDCRC_ERROR_INJECT

ACTION ERROR_INJECT = EXECUTE;
DATA DEVICE_DATA;
BOOLEAN out[32];
BOOLEAN in[32] = $02040608;     'shift in any wrong CRC value
ENDDATA;
PROCEDURE EXECUTE USES DEVICE_DATA;
BOOLEAN X = 0;
DRSTOP IDLE;
IRSTOP IDLE;
STATE IDLE;
IRSCAN 10, $015;                'shift in CHANGE_EDREG instruction
WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;
DRSCAN 32, in[31..0], CAPTURE out[31..0];
WAIT IDLE, 10 CYCLES, 50 USEC, IDLE;
PRINT " ";
PRINT "Data read out from the Storage Register: "out[31], out[30], out[29],
out[28], out[27],
out[26], out[25], out[24], out[23], out[22], out[21], out[20], out[19],
out[18], out[17], out[16], out[15], out[14], out[13], out[12], out[11],
out[10], out[9], out[8], out[7], out[6], out[5], out[4], out[3],
out[2], out[1], out[0];          'Read out correct precomputed CRC value
PRINT " ";
```

```
        STATE IDLE;
        EXIT 0;
        ENDPROC;
```

You can run the **.jam** file using **quartus_jli** executable with the following command line:

```
quartus_jli -c<cable index> -a<action name> <filename>.jam
```

Related Information

- **SEU Mitigation and Configuration Error Detection** on page 2-20
- **AN 425: Using the Command-Line Jam STAPL Solution for Device Programming**
  Provides more information about quartus_jli command line executable.

Enabling Error Detection

The CRC error detection feature in the Quartus Prime software generates the `CRC_ERROR` output to the optional dual-purpose `CRC_ERROR` pin.

To enable the error detection feature using CRC, follow these steps:

1. Open the Quartus Prime software and load a project using MAX 10 device family.
2. On the **Assignments** menu, click **Settings**. The **Settings** dialog box appears.
3. In the **Category** list, select **Device**.
4. Click **Device and Pin Options**.
5. In the **Device and Pin Options** dialog box, click the **Error Detection CRC** tab.
6. Turn on **Enable error detection CRC**.
7. In the **Divide error check frequency by** field, enter a valid divisor.

   The divisor value divides down the frequency of the configuration oscillator output clock. This output clock is used as the clock source for the error detection process.
8. Click **OK**.
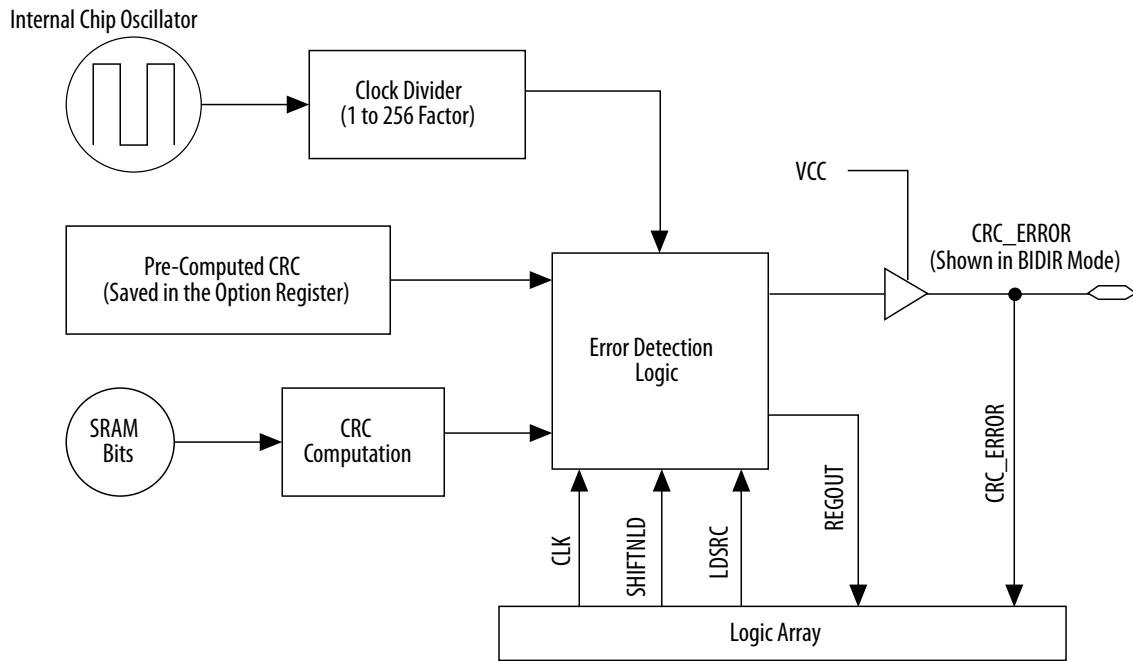
Related Information
SEU Mitigation and Configuration Error Detection on page 2-20

Accessing Error Detection Block Through User Logic

The error detection circuit stores the computed 32-bit CRC signature in a 32-bit register. The user logic from the core reads out this signature. The `fiftyfivenm_crcblock` primitive is a WYSIWYG component used to establish the interface from the user logic to the error detection circuit. The `fiftyfivenm_crcblock` primitive atom contains the input and output ports that must be included in the atom. To access the logic array,you must insert the `fiftyfivenm_crcblock` WYSIWYG atom into your design. The recommended clock frequency of .clk port is to follow the clock frequency of EDCRC block.

Figure 3-3: Error Detection Block Diagram with Interfaces for MAX 10 Devices

The following example shows how the input and output ports of a WYSIWYG atom are defined in the MAX 10 device.

```
fiftyfivenm_crcblock <name>
(
    .clk(<ED_CLK clock source>),
    .shiftnld(<ED_SHIFTNLD source>),
    .ldsrc (<LDSRC source>),
    .crcerror(<CRCERROR_CORE out destination>),
    .regout(<output destination>)
);
defparam <crcblock_name>.oscillator_divider = <internal oscillator division (1 to
256)>;
```

Table 3-5: Port Definitions

Port	Input/ Output	Definition
<crcblock_name>	—	Unique identifier for the CRC block and represents any identifier name that is legal for the given description language such as Verilog HDL, VHDL, AHDL. This field is required.
.clk(<clock source>	Input	This signal designates the clock input of this cell. All operations of this cell are with respect to the rising edge of the clock. Whether it is the loading of the data into the cell or data out of the cell, it always occurs on the rising edge. This port is required.

Port	Input/Output	Definition
.shiftnld (<shiftnld source>)	Input	This signal is an input into the error detection block. If shiftnld=1, the data is shifted from the internal shift register to the regout at each rising edge of clk. If shiftnld=0, the shift register parallel loads either the pre-calculated CRC value or the update register contents depending on the ldsrc port input. This port is required.
.ldsrc (<ldsrc source>)	Input	This signal is an input into the error detection block. If ldsrc=0, the pre-computed CRC register is selected for loading into the 32-bit shift register at the rising edge of clk when shiftnld=0. If ldsrc=1, the signature register (result of the CRC calculation) is selected for loading into the shift register at the rising edge of clk when shiftnld=0. This port is ignored when shiftnld=1. This port is required.
.crcerror (<crcerror out destination>)	Output	This signal is the output of the cell that is synchronized to the internal oscillator of the device (100-MHz or 80-MHz internal oscillator) and not to the clk port. It asserts automatically high if the error block detects that a SRAM bit has flipped and the internal CRC computation has shown a difference with respect to the pre-computed value. This signal must be connected either to an output pin or a bidirectional pin. If it is connected to an output pin, you can only monitor the CRC_ERROR pin (the core cannot access this output). If the CRC_ERROR signal is used by core logic to read error detection logic, this signal must be connected to a BIDIR pin. The signal is fed to the core indirectly by feeding a BIDIR pin that has its oe port connected to V_{CC}.
.regout (<output destination>)	Output	This signal is the output of the error detection shift register synchronized to the clk port, to be read by core logic. It shifts one bit at each cycle. User should clock the clk signal 31 cycles to read out the 32 bits of the shift register. The values at the .regout port are an inversion of the actual values.

Related Information

Enabling Data Compression

When you enable compression, the Quartus Prime software generates configuration files with compressed configuration data.

A compressed configuration file is needed to use the dual configuration mode in the internal configuration scheme. This compressed file reduces the storage requirements in internal flash memory, and decreases the time needed to send the bitstream to the MAX 10 device family. There are two methods to enable compression for the MAX 10 device family bitstreams in the Quartus Prime software:

- Before design compilation—using the **Compiler Settings** menu.
- After design compilation—using the **Convert Programming Files** option.

Enabling Compression Before Design Compilation

To enable compression before design compilation, follow these steps:

1. On the **Assignments** menu, click **Device**. The **Settings** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. Click the **Configuration** tab.
4. Turn on **Generate compressed bitstreams**.
5. Click **OK**.
6. In the **Settings** dialog box, click **OK**.

Related Information
[Configuration Data Compression](#) on page 2-24

Enabling Compression After Design Compilation

To enable compression after design compilation, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.
2. Under **Output programming file**, from the pull-down menu, select your desired file type.
3. If you select the Programmer Object File (**.pof**), you must specify a configuration device, directly under the file type.
4. In the **Input files to convert** box, select **SOF Data**.
5. Click **Add File** to browse to the MAX 10 device family **.sof**.
6. In the **Convert Programming Files** dialog box, select the **.pof** you added to **SOF Data** and click **Properties**.
7. In the **SOF Properties** dialog box, turn on the **Compression** option.

Related Information
[Configuration Data Compression](#) on page 2-24

AES Encryption

This section covers detailed guidelines on applying AES Encryption for design security. There are two main steps in applying design security in MAX 10 devices. First is to generate the encryption key programming (**.ekp**) file and second is to program the **.ekp** file into the device.

The **.ekp** file has other different formats, depending on the hardware and system used for programming. There are three file formats supported by the Quartus Prime software:

- JAM Byte Code (**.jbc**) file
- JAM™ Standard Test and Programming Language (STAPL) Format (**.jam**) file
- Serial Vector Format (**.svf**) file

Only the **.ekp** file type generated automatically from the Quartus Prime software. You must create the **.jbc**, **.jam** and **.svf** files using the Quartus Prime software if these files are required in the key programming.

Note: Altera recommends that you keep the **.ekp** file confidential.

Generating .ekp File and Encrypt Configuration File

To generate the **.ekp** file and encrypt your configuration file, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.
2. Under **Output programming file**, select **Programmer Object File** (**.pof**) in the **Programming file type** list.
3. In the **Mode** list, select **Internal Configuration**.
4. Click **Option/Boot Info** and the **ICB setting** dialog box will appear.
5. You can enable the enable the **Allow encrypted POF** only option. Click **OK** once ICB setting is set.

    The device will only accept encrypted bitstream during internal configuration if this option is enabled. If you encrypt one of CFM0, CFM1 or CFM2 only, the Programmer will post a warning.
6. Type the file name in the **File name** field, or **browse** to and select the file.
7. Under the **Input files to convert** section, click **SOF Data**.
8. Click **Add File** to open the **Select Input File** dialog box.
9. Browse to the unencrypted **.sof** and click **Open**.
10. Under the **Input files to convert** section, click on the added **.sof**.
11. Click **Properties** and the **SOF Files Properties: Bitstream Encryption** dialog box will appear.
12. Turn on **Generate encrypted bitstream**.
13. Turn on **Generate key programming file** and type the **.ekp** file path and file name in the text area, or browse to and select **<filename>.ekp**.
14. You can the key with either a **.key** file or entering the key manually.

    Note: MAX 10 devices require the entry of 128-bit keys.

- Adding key with a **.key** file.

  The **.key** file is a plain text file in which each line represents a key unless the line starts with "#". The "#" symbol is used to denote comments. Each valid key line has the following format:

  ```
  <key identity><white space><128-bit hexadecimal key>
  # This is an example key file
  key1 0123456789ABCDEF0123456789ABCDEF
  ```

  1. Enable the **Use key file** checkbox.
  2. Click **Open** and add the desired **.key** file and click **Open** again.
  3. Under **Key entry** part, the key contained in the **.key** file will be selected in the drop-down list.
  4. Click **OK**.

- Entering your key manually.

  1. Under **Key entry** part, click the **Add** button.
  2. Select the **Key Entry Method** to enter the encryption key either with the **On-screen Keypad** or **Keyboard**.
  3. Enter a key name in the **Key Name (alphanumeric)** field.
  4. Key in the desired key in the **Key (128-bit hexadecimal)** field and repeat in the **Confirm Key** field below it.
  5. Click **OK**.

15. Read the design security feature disclaimer. If you agree, turn on the **acknowledgment** box and click **OK**.
16. In the **Convert Programming Files** dialog box, click **OK**. The **<filename>.ekp** and encrypted configuration file will be generated in the same project directory.

   Note: For dual configuration **.pof** file, both **.sof** file need to be encrypted with the same key. The generation of key file and encrypted configuration file will not be successful if different keys are used.

Generating .jam/.jbc/.svf file from .ekp file

To generate **.jam**/**.jbc**/**.svf** file from **.ekp** file, follow these steps:

1. On the **Tools** menu, click **Programmer** and the **Programmer** dialog box will appear.
2. In the **Mode** list, select **JTAG** as the programming mode.
3. Click **Hardware Setup**. The **Hardware Setup** dialog box will appear.
4. Select **USBBlaster** as the programming hardware in the **currently selected hardware list** and click **Done**.
5. Click **Add File** and the **Select Programmer File** dialog box will appear.
6. Type `<filename>.ekp` in the **File name** field and click **Open**.
7. Select the .ekp file you added and click **Program/Configure**.
8. On the **File** menu, point to **Create/Update** and click **Create JAM, SVF, or ISC File**. The **Create JAM, SVF, or ISC File** dialog box will appear.
9. Select the file format required for the **.ekp** file in the **File format** field.

- JEDEC STAPL Format (**.jam**)
- Jam STAPL Byte Code (**.jbc**)
- Serial Vector Format (**.svf**)

10. Type the file name in the **File name** field, or browse to and select the file.
11. Click **OK** to generate the **.jam**, **.jbc** or **.svf** file.

Programming .ekp File and Encrypted POF File

There are two methods to program the encrypted **.pof** and **.ekp** files:

- Program the **.ekp** and **.pof** separately.

    Note: You only can program the **.ekp** and **.pof** separately when **Allow encrypted POF only** option is disabled.

- Integrate the **.ekp** into **.pof** and program both altogether.

Programming .ekp File and Encrypted .pof Separately

To program the **.ekp** and encrypted **.pof** separately using the Quartus Prime software, follow these steps:

1. In the Quartus Prime Programmer, under the **Mode** list, select **JTAG** as the programming mode.
2. Click **Hardware Setup** and the **Hardware Setup** dialog box will appear.
3. Select **USBBlaster** as the programming hardware in the **Currently selected hardware** list and click **Done**.
4. Click **Add File** and the **Select Programmer File** dialog box will appear.
5. Type `<filename>.ekp` in the **File name** field and click **Open**.
6. Select the **.ekp** file you added and click **Program/Configure**.
7. Click **Start** to program the key.

    Note: The Quartus Prime software message window provides information about the success or failure of the key programming operation. Once the **.ekp** is programmed, **.pof** can be programmed separately. To retain the security key in the internal flash that had been programmed through the **.ekp**, continue with the following steps.

8. Select the **.pof** to be programmed to the selected device.
9. Check only the functional block that need to be updated at child level for CFM and UFM. Do not check operation at the parent level when using Programmer GUI.
10. After all settings are set, click **Start** to start programming.

Integrate the .ekp into .pof Programming

To integrate the **.ekp** into **.pof** and program both altogether using the Quartus Prime software, follow these steps:

1. In the Quartus Prime Programmer, under the **Mode** list, select **JTAG** as the programming mode.
2. Click **Hardware Setup** and the **Hardware Setup** dialog box will appear.
3. Select **USBBlaster** as the programming hardware in the **Currently selected hardware** list and click **Done**.
4. Click the **Auto Detect** button on the left pane.
5. Select the **.pof** you wish to program to the device.

6. Select the **<yourpoffile.pof>**, `right click` and select **Add EKP File** to integrate **.ekp** file with the **.pof** file.

   Once the **.ekp** is integrated into the **.pof**, you can to save the integrated **.pof** into a new **.pof**. This newly saved file will have original **.pof** integrated with **.ekp** information.

7. Select the **<yourpoffile.pof>** in the **Program/Configure** column.

8. After all settings are set, click **Start** to start programming

Encryption in Internal Configuration

During internal configuration, the FPGA decrypts the **.pof** with the stored key and uses the decrypted data to configure itself. The configuration image loaded during configuration is also affected by the encryption settings and the **Configure device from CFM0 only** setting.

Table 3-6: Configuration Image Outcome Based on Encryption Settings, Encryption Key and CONFIG_SEL Pin Settings

Table shows the scenario when you disable the **Configure device from CFM0 only**.

Configura-tion Image Mode	CFM0 (image 0) Encryption Key	CFM1 (image 1) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	CONFIG_SEL pin	Design Loaded After Power-up
Single	Not Encrypted	Not Available	No key	Disabled	0	image 0
Single	Not Encrypted	Not Available	No key	Disabled	1	image 0
Single	Not Encrypted	Not Available	Key X	Disabled	0	image 0
Single	Not Encrypted	Not Available	Key X	Disabled	1	image 0
Single	Not Encrypted	Not Available	Key X	Enabled	0	Configuration Fail
Single	Not Encrypted	Not Available	Key X	Enabled	1	Configuration Fail
Single	Key X	Not Available	No key	Enabled	0	Configuration Fail
Single	Key X	Not Available	No key	Enabled	1	Configuration Fail
Single	Key X	Not Available	Key X	Enabled	0	image 0
Single	Key X	Not Available	Key X	Enabled	1	image 0
Single	Key X	Not Available	Key Y	Enabled	0	Configuration Fail
Single	Key X	Not Available	Key Y	Enabled	1	Configuration Fail
Dual	Not Encrypted	Not Encrypted	No key	Disabled	0	image 0
Dual	Not Encrypted	Not Encrypted	No key	Disabled	1	image 1
Dual	Key X	Not Encrypted	No key	Disabled	0	image 1[11]
Dual	Key X	Not Encrypted	No key	Disabled	1	image 1
Dual	Key X	Not Encrypted	Key X	Disabled	0	image 0
Dual	Key X	Not Encrypted	Key X	Disabled	1	image 1
Dual	Key X	Not Encrypted	Key X	Enabled	0	image 0

[11] After image 0 configuration failed, device will automatically load image 1.

Configura-tion Image Mode	CFM0 (image 0) Encryption Key	CFM1 (image 1) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	CONFIG_SEL pin	Design Loaded After Power-up
Dual	Key X	Not Encrypted	Key X	Enabled	1	image 0
Dual	Key X	Not Encrypted	Key Y	Enabled	0	Configuration Fail
Dual	Key X	Not Encrypted	Key Y	Enabled	1	Configuration Fail
Dual	Key X	Key X	No key	Enabled	0	Configuration Fail
Dual	Key X	Key X	No key	Enabled	1	Configuration Fail
Dual	Key X	Key X	Key X	Enabled	0	image 0
Dual	Key X	Key X	Key X	Enabled	1	image 1
Dual	Key X	Key Y	Key X	Enabled	0	image 0
Dual	Key X	Key Y	Key X	Enabled	1	image 0[12]
Dual	Key Y	Key Y	Key Y	Enabled	0	image 0
Dual	Key Y	Key Y	Key Y	Enabled	1	image 1
Dual	Key X	Key Y	Key Y	Enabled	0	image 1[11]
Dual	Key X	Key Y	Key Y	Enabled	1	image 1

Table 3-7: Configuration Image Outcome Based on Encryption Settings and Encryption Key

Table shows the scenario when you enable the **Configure device from CFM0 only**.

CFM0 (image 0) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	Design Loaded After Power-up
Not Encrypted	No key	Disabled	image 0
Not Encrypted	Key X	Disabled	image 0
Not Encrypted	Key Y	Disabled	image 0
Not Encrypted	No key	Enabled	Configuration Fail
Not Encrypted	Key X	Enabled	Configuration Fail
Not Encrypted	Key Y	Enabled	Configuration Fail
Key X	No key	Disabled	Configuration Fail
Key X	Key X	Disabled	image 0
Key X	Key Y	Disabled	Configuration Fail
Key X	No key	Enabled	Configuration Fail
Key X	Key X	Enabled	image 0
Key X	Key Y	Enabled	Configuration Fail
Key Y	No key	Disabled	Configuration Fail
Key Y	Key X	Disabled	Configuration Fail

[12] After image 1 configuration failed, device will automatically load image 0.

CFM0 (image 0) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	Design Loaded After Power-up
Key Y	Key Y	Disabled	image 0
Key Y	No key	Enabled	Configuration Fail
Key Y	Key X	Enabled	Configuration Fail
Key Y	Key Y	Enabled	image 0

Related Information

MAX 10 JTAG Secure Design Example

This design example demonstrates the instantiation of internal JTAG WYSIWYG atom and the example of user logic implementation in the Quartus Prime software to execute the `LOCK` and `UNLOCK` JTAG instructions. This design example is targeted for MAX 10 devices with the JTAG Secure Mode enabled.

Related Information
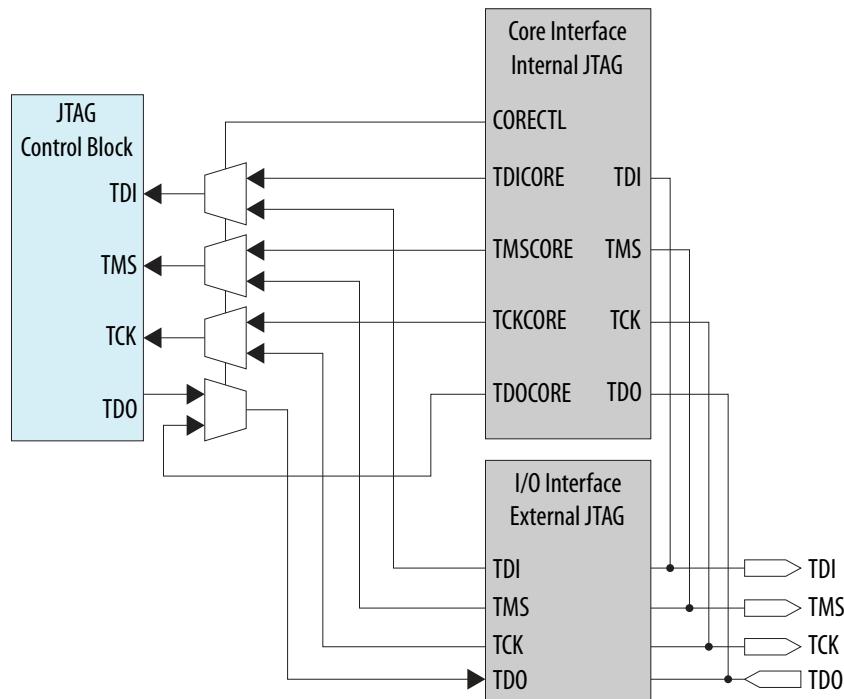
-
-
- **JTAG Secure Design Example**

Internal JTAG Interface

There are two interfaces to access the JTAG control block in MAX 10 devices:

- External JTAG interface—connection of the JTAG control block from the physical JTAG pins; `TCK`, `TDI`, `TDO`, and `TMS`.
- Internal JTAG interface—connection of the JTAG control block from the internal FPGA core fabric.

You can only access the JTAG control block using either external or internal JTAG interface one at a time. External JTAG interface are commonly used for JTAG configuration using programming cable. To access the internal JTAG interface, you must include the WYSIWYG atom in your Quartus Prime software design.

Figure 3-4: Internal and External JTAG Interface Connections

Note: To ensure the internal JTAG of MAX 10 devices function correctly, all four JTAG signals (TCK, TDI, TMS and TDO) in the JTAG WYSIWYG atom need to be routed out. The Quartus Prime software will automatically assign the ports to their corresponding dedicated JTAG pins.

WYSIWYG Atom for Internal JTAG Block Access

The following example shows how the input and output ports of a WYSIWYG atom are defined in the MAX 10 device.

```
fiftyfivenm_jtag <name>
(
    .tms(),
    .tck(),
    .tdi(),
    .tdoutap(),
    .tdouser(),
    .tdicore(),
    .tmscore(),
    .tckcore(),
    .corectl(),
    .tdo(),
    .tmsutap(),
    .tckutap(),
    .tdiutap(),
    .shiftuser(),
    .clkdruser(),
    .updateuser(),
    .runidleuser(),
    .usr1user(),
    .tdocore(),
```

```
        .ntdopinena()
    );
```

Table 3-8: Port Description

Ports	Input/Output	Functions
`<name>`	—	Identifier for the MAX 10 JTAG WYSIWYG atom and represents any identifier name that is legal for the given description language, such as Verilog HDL, VHDL, and AHDL.
`.corectl()`	Input	Active high input to the JTAG control block to enable the internal JTAG access from core interface. When the FPGA enters user mode after configuration, this port is low by default. Pulling this port to logic high will enable the internal JTAG interface (with external JTAG interface disabled at the same time) and pulling this port to logic low will disable the internal JTAG interface (with external JTAG interface enabled at the same time).
`.tckcore()`	Input	Core `tck` signal
`.tdicore()`	Input	Core `tdi` signal
`.tmscore()`	Input	Core `tms` signal
`.tdocore()`	Output	Core `tdo` signal
`.tck()`	Input	Pin `tck` signal
`.tdi()`	Input	Pin `tdi` signal
`.tms()`	Input	Pin `tms` signal
`.tdo()`	Output	Pin `tdo` signal
`.clkdruser()`	Input/Output	These ports are not used for enabling the JTAG Secure mode using the internal JTAG interface, you can leave them unconnected.
`.runidleuser()`		
`.shiftuser()`		
`.tckutap()`		
`.tdiutap()`		
`.tdouser()`		
`.tdoutap()`		
`.tmsutap()`		
`.updateuser()`		
`.usr1user()`		
`.ntdopinena()`		

Executing LOCK and UNLOCK JTAG Instructions

When you configure this reference design into a MAX 10 device with the JTAG Secure mode enabled, the device is in JTAG Secure mode after power-up and configuration.

To disable the JTAG Secure mode, trigger the `start_unlock` port of the user logic to issue the UNLOCK JTAG instruction. After the UNLOCK JTAG instruction is issued, the device exits from JTAG secure mode. When the the JTAG Secure mode is disabled, you can choose to full-chip erase the internal flash of MAX 10 device to disable the JTAG Secure mode permanently.

The `start_lock` port in the user logic triggers the execution of the LOCK JTAG instruction. Executing this instruction enables the JTAG Secure mode of the MAX 10 device.

Figure 3-5: LOCK or UNLOCK JTAG Instruction Execution
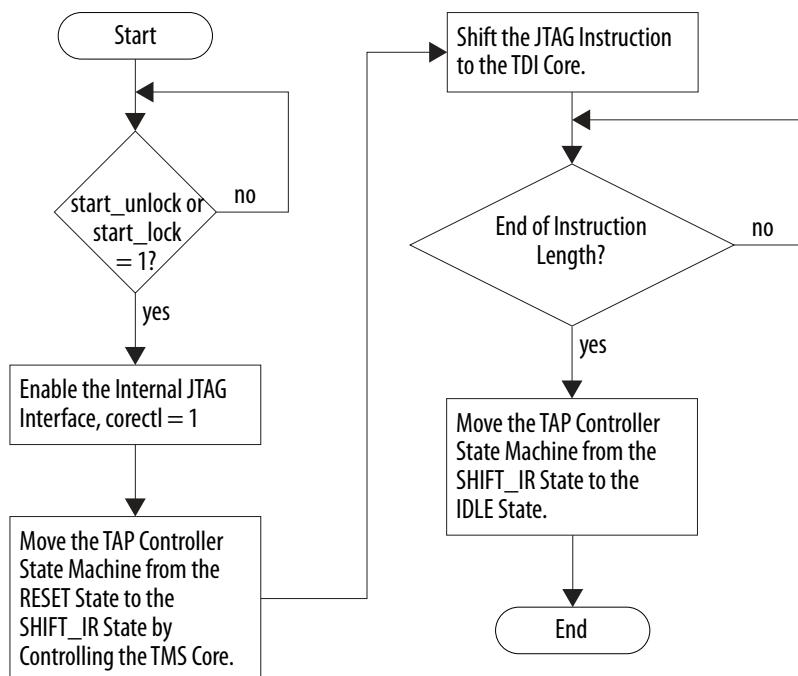
Table 3-9: Input and Output Port of the User Logic

Port	Input/ Output	Function
clk_in	Input	Clock source for the user logic. The f_{MAX} of the user logic depends on the timing closure analysis. You need to apply timing constraint and perform timing analysis on the path to determine the f_{MAX}.
start_lock	Input	Triggers the execution of the LOCK JTAG instruction to the internal JTAG interface. Pulse signal high for at least 1 clock cycle to trigger.

Port	Input/Output	Function
start_unlock	Input	Triggers the execution of the UNLOCK JTAG instruction to the internal JTAG interface. Pulse signal high for at least 1 clock cycle to trigger.
jtag_core_en_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the corectl port of the JTAG WYSIWYG atom to enable the internal JTAG interface.
tck_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the tck_core port of the JTAG WYSIWYG atom.
tdi_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the tdi_core port of the JTAG WYSIWYG atom.
tms_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the tms_core port of the JTAG WYSIWYG atom.
indicator	Output	Logic high of this output pin indicates the completion of the LOCK or UNLOCK JTAG instruction execution.

Verifying the JTAG Secure Mode

You can verify whether your device has successfully entered or exited JTAG secure mode by executing a non-mandatory JTAG instruction.

Note: You must instantiate the internal JTAG interface for you unlock the external JTAG when the device is in JTAG Secure mode.

When you enable the JTAG Secure option, the MAX 10 device will be in the JTAG Secure mode after power-up. To validate the JTAG Secure feature in your design example, perform these steps:

1. Configure the reference design **.pof** file into the device with JTAG Secure mode enabled. After power cycle, the device should be in JTAG Secure mode.
2. You can ensure that the device enters user mode successfully by observing one of the following:

   - CONFDONE pin goes high
   - counter_output pin starts toggling

3. Issue the PULSE_NCONFIG JTAG instruction using the external JTAG pins to reconfigure the device. You can use the **pulse_ncfg.jam** file attached in the design example. To execute the **pulse_ncfg.jam** file, you can use the quartus_jli or the JAM player. You can ensure that the device does not reconfigure by observing one of the following:

   - CONFDONE pin stays high
   - counter_output pin continues toggling

   Unsuccessful reconfiguration verifies that the device is currently in JTAG Secure mode.
4. Pull the start_unlock port of the user logic to logic high to execute the UNLOCK JTAG instruction.

   The indicator port goes high after the UNLOCK JTAG instruction is complete.
5. Issue the PULSE_NCONFIG JTAG instruction using the external JTAG pins to reconfigure the device. You can ensure that the device reconfigures successfully by observing one of the following:

- `CONFDONE` pin is low
- `counter_output` pin stops toggling

Successful reconfiguration verifies that the device is currently not in JTAG Secure mode.

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.

Altera Unique Chip ID IP Core

This section provides the guideline to implement the Altera Unique Chip ID IP Core.

Related Information

Instantiating the Altera Unique Chip ID IP Core

To instantiate the Altera Unique Chip ID IP Core, follow these steps:

1. On the Tools menu of the Quartus Prime software, click **IP Catalog.**
2. Under the Library category, expand the Basic Functions and Configuration Programming.
3. Select **Altera Unique Chip ID** and click **Add**, and enter your desired output file name
4. In the Save IP Variation dialog box:

   - Set your IP variation filename and directory.
   - Select IP variation file type.

5. Click **Finish**.

Resetting the Altera Unique Chip ID IP Core

To reset the Altera Unique Chip ID IP core, you must assert high to the `reset` signal for at least one clock cycle. After you de-assert the `reset` signal, the Altera Unique Chip ID IP core re-reads the unique chip ID

ALTERA
now part of Intel

of your device from the fuse ID block. The Altera Unique Chip ID IP core asserts the `data_valid` signal after completing the operation.

Altera Dual Configuration IP Core

This section provides the guideline to implement the Altera Dual Configuration IP Core.

Instantiating the Altera Dual Configuration IP Core

To instantiate the Altera Dual Configuration IP Core, follow these steps:

1. On the Tools menu of the Quartus Prime software, click **IP Catalog.**
2. Under the Library category, expand the Basic Functions and Configuration Programming.
3. Select **Altera Dual Configuration** and after clicking **Add**, the IP Parameter Editor appears.
4. In the New IP Instance dialog box:

   - Set the top-level name of your IP.
   - Select the Device family.
   - Select the Device

5. Click **OK**.

Related Information

- **Altera Dual Configuration IP Core** on page 2-16
- **Accessing Remote System Upgrade through User Logic** on page 3-8
- **AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor**
  Provides reference design for remote system upgrade in MAX 10 FPGA devices.
- **I2C Remote System Update Example**
  This example demonstrates a remote system upgrade using the I2C protocol.

Altera Dual Configuration IP Core Avalon-MM Address Map

Table 5-1: Altera Dual Configuration IP Core Avalon-MM Address Map for MAX 10 Devices

- Altera recommends you to set the reserve bits to `0` for write operations. For read operations, the IP core will always generate `0` as the output.
- Write `1` to trigger any operation stated in the description.
- You need to trigger the desired operation from offset 2 before any read operation of offset 4, 5, 6 and 7.

Offset	R/W	Width (Bits)	Description
0	W	32	- Bit 0—trigger reconfiguration. - Bit 1—reset the watchdog timer. - Bit 31:2—reserved. Signals are triggered at the same write cycle on Avalon.

Offset	R/W	Width (Bits)	Description
1	W	32	• Bit 0—trigger `config_sel_overwrite` to the input register. • Bit 1—writes `config_sel` to the input register. Set 0 or 1 to load from configuration image 0 or 1 respectively. • Bit 31:2—reserved. The `busy` signal is generated right after the write cycle, while the configuration image information is registered. Once `busy` signal is high, writing to this address is ignored until the process is completed and the `busy` signal is de-asserted.
2	W	32	• Bit 0—trigger read operation from the user watchdog. • Bit 1—trigger read operation from the previous state application 2 register. • Bit 2—trigger read operation from the previous state application 1 register. • Bit 3—trigger read operation from the input register. • Bit 31:4—reserved. The `busy` signal is generated right after the write cycle.
3	R	32	• Bit 0—IP `busy` signal. • Bit 31:1—reserved. The `busy` signal indicates the Dual Configuration IP core is in the writing or reading process. In this state, all write operation to the remote system upgrade block registers operation request are ignored except for triggering the reset timer. Altera recommends you to pull this `busy` signal once you triggered any read or write process.
4	R	32	• Bit 11:0—user watchdog value.[13] • Bit 12—current state of the user watchdog. • Bit 16:13—`msm_cs` value of the current state. • Bit 31:17—reserved.
5	R	32	• Bit 3:0—previous state application 1 reconfiguration source value from the *Remote System Upgrade Status Register—Previous StateBit for MAX 10 Devices* table. • Bit 7:4—`msm_cs` value of the previous state application 1. • Bit 31:8—reserved.

[13] You can only read the 12 most significant bit of the 29 bit user watchdog value using Dual Configuration IP Core.

Offset	R/W	Width (Bits)	Description
6	R	32	• Bit 3:0—previous state application 2 reconfiguration source value from the *Remote System Upgrade Status Register— Previous StateBit for MAX 10 Devices* table. • Bit 7:4—`msm_cs` value of the previous state application 2. • Bit 31:8—reserved.
7	R	32	• Bit 0—`config_sel_overwrite` value from the input register. • Bit 1—`config_sel` value of the input register.[14] • Bit 31:2—reserved.

Related Information

- **Altera Dual Configuration IP Core** on page 2-16
- **Avalon Interface Specifications**
  Provides more information about the Avalon-MM interface specifications applied in Altera Dual Configuration IP Core.
- **Instantiating the Altera Dual Configuration IP Core** on page 4-2
- **Remote System Upgrade Status Registers** on page 2-14
  The Remote System Upgrade Status Register—Previous StateBit for MAX 10 Devices table provides more information about previous state applications reconfiguration sources.

Altera Dual Configuration IP Core Parameters

Table 5-2: Altera Dual Configuration IP Core Parameter for MAX 10

Parameter	Value	Description
Clock frequency	Up to 80MHz	Specifies the number of cycle to assert `RU_nRSTIMER` and `RU_nCONFIG` signals. Note that maximum `RU_CLK` is 40Mhz, Altera Dual Configuration IP Core has restriction to run at 80Mhz maximum, which is twice faster than hardware limitation. This is because Altera Dual Configuration IP Core generate `RU_CLK` at half rate of the input frequency.

---

[14] Reads the `config_sel` of the input register only. It will not reflect the physical CONFIG_SEL pin setting.

Altera Unique Chip ID IP Core Ports

Table 6-1: Altera Unique Chip ID IP Core Ports

Port	Input/Output	Width (Bits)	Description
clkin	Input	1	• Feeds clock signal to the unique chip ID block. The maximum supported frequency is 100 MHz. • When you provide a clock signal, the IP core reads the value of the unique chip ID and sends the value to the `chip_id` output port.
reset	Input	1	• Resets the IP core when you assert the `reset` signal to high for at least one clock cycle. • The `chip_id [63:0]` output port holds the value of the unique chip ID until you reconfigure the device or reset the IP core.
data_valid	Output	1	• Indicates that the unique chip ID is ready for retrieval. If the signal is low, the IP core is in initial state or in progress to load data from a fuse ID. • After the IP core asserts the signal, the data is ready for retrieval at the `chip_id[63..0]` output port.
chip_id	Output	64	• Indicates the unique chip ID according to its respective fuse ID location. The data is only valid after the IP core asserts the `data_valid` signal. • The value at power-up resets to `0`.

ISO 9001:2008 Registered

now part of Intel

Document Revision History for MAX 10 FPGA Configuration User Guide

Date	Version	Changes
May 2016	2016.05.13	• Changed instances of Standard POR to Slow POR to reflect Quartus Prime GUI.
		• Updated t_{CFG} to $t_{RU\_nCONFIG}$.
		• Corrected file type from **.ekp** to **.pof** in Step 8 of *Programming .ekp File and Encrypted .pof Separately*.
		• Corrected Use secondary image ISP data as default setting when available description in *ICB Values and Descriptions for MAX 10 Devices* table.
		• Corrected CFM programming time.
		• Added note on JTAG pin requirements when using JTAG pin sharing.
		• Moved *JTAG Pin Sharing Behavior* under *Guidelines: Dual-Purpose Configuration Pin*.
		• Updated configuration sequence diagram by moving 'Clears configuration RAM bits from Power-up state to Reset state.
		• Corrected error detection port input and output for `<crcblock_name>` from input to none.
		• Added example of remote system upgrade access through user interface and port definitions.
		• Removed preliminary terms for *Error Detection Frequency* and *Cyclic Redundancy Check Calculation Timing*.
		• Added *Connection Setup for JTAG Multi-Device Configuration using Download Cable* diagram.
		• Updated *Connection Setup for JTAG Single-Device Configuration using Download Cable* diagram.
		• Added new JTAG Secure design example.
		• Edited Remote System Upgrade section title by removing in Dual Image Configuration.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Date	Version	Changes
		• Updated *Monitored Power Supplies Ramp Time Requirement for MAX 10 Devices* table. • Added *Internal Configuration Time*. • Removed Instant ON feature. • Updated User Flash Memory instances to additional UFM in *Configuration Flash Memory Sectors Utilization for all MAX 10 with Analog and Flash Feature Options* figure.
December	2015.12.14	• Updated ICB setting description for *Set I/O to weak pull-up prior usermode* option to state the weak pull-up is enabled during configuration. • Removed *Accessing the Remote System Upgrade Block Through User Interface.* • Added input and output port definition for error detection WYSIWYG atom. • Updated the I/O pin state to be dependent on ICB bit setting during reconfiguration.
November 2015	2015.11.02	• Removed JRunner support for JTAG configuration and link to AN 414. • Updated differences in supported internal configuration mode supported based on device feature options in a table. • Removed maximum number of compressed configuration image table do to redundancy. • Updated Initialization Configuration Bits setting and description to reflect Quartus Prime 15.1 update. • Updated **Enable JTAG pin sharing** and **Enable nCONFIG, nSTATUS, and CONF_DONE pins** to reflect Quartus II 15.1 update. • Added information about ISP clamp feature. • Updated information about steps to generate Raw Programming Data (.rpd). • Renamed section title from *Configuration Total Flash Memory Programming Time* to *Configuration Flash Memory Programming Time.* • Renamed table title from *Configuration Total Flash Memory Programming Time for Sectors in MAX 10 Devices* to *Configuration Flash Memory Programming Time for Sectors in MAX 10 Devices.* • Added note to *Configuration Flash Memory Programming Time for Sectors in MAX 10 Devices* table. • Added information about internal JTAG interface and accessing internal JTAG block through user interface. • Added MAX 10 JTAG Secure design example.

Date	Version	Changes
June 2015	2015.06.15	• Added related information link to AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor in *Altera Dual Configuration IP Core References* and *Remote System Upgrade in Dual Compressed Images.* • Added pulse holding requirement time for `RU_nRSTIMER` in *Remote System Upgrade Circuitry Signals for MAX 10 Devices* table. • Added link to *Remote System Upgrade Status Register—Previous State Bit for MAX 10 Devices* table for related entries in *Altera Dual Configuration IP Core Avalon-MM Address Map for MAX 10 Devices* table.

Date	Version	Changes
May 2015	2015.05.04	• Rearranged and updated Configuration Setting names 'Initialization Configuration Bits for MAX 10 Devices' table. • Updated 'High-Level Overview of Internal Configuration for MAX 10 Devices' figure with JTAG configuration and moved the figure to 'Configuration Schemes' section. • Added link to corresponding description of configuration settings in 'Initialization Configuration Bits for MAX 10 Devices' table. • Updated the default watchdog time value from hexadecimal to decimal value in 'Initialization Configuration Bits for MAX 10 Devices' table. • Updated the ISP data description in 'Initialization Configuration Bits for MAX 10 Devices' table. • Updated 'User Watchdog Timer' by adding time-out formula. • Added link to 'User Watchdog Internal Circuitry Timing Specifications' in MAX 10 FPGA Device Datasheet. • Added footnote to indicate that JTAG secure is disabled by default and require Altera support to enable in 'Initialization Configuration Bits for MAX 10 Devices' table. • Updated minimum and maximum CRC calculation time for divisor 2. • Updated remote system upgrade flow diagram. • Updated 'Encryption in Internal Configuration' table by adding 'Key' terms and changed Image 1 and Image 2 to Image 0 and Image 1 respectively. • Added footnote to 'Encryption in Internal Configuration' to indicate auto-reconfiguration when image fails. • Added formula to calculate minimum and maximum CRC calculation time for other than divisor 2. • Added caution when JTAG Secure is turned on. • Added information about auto-generated .pof for certain type of internal configuration modes. • Added .pof and ICB setting guide through Device and Pin Options and convert programming file. • Added configuration RAM (CRAM) in 'Overview' • Editorial changes.
December 2014	2014.12.15	• Rename BOOT_SEL pin to CONFIG_SEL pin. • Update Altera IP Core name from Dual Boot IP Core to Altera Dual Configuration IP Core. • Added information about the AES encryption key part of ICB. • Added encryption feature guidelines. • Updated ICB settings options available in 14.1 release. • Updated Programmer options on CFM programming available in 14.1 release.

Date	Version	Changes
September 2014	2014.09.22	Initial release.

MAX 10 User Flash Memory User Guide

Contents

MAX 10 User Flash Memory Overview

2016.05.02

UG-M10UFM   ✉ **Subscribe**   💬 **Send Feedback**

Altera® MAX® 10 FPGAs offer a user flash memory (UFM) block that stores non-volatile information.

The UFM provides an ideal storage solution that you can access using the Avalon Memory Mapped (Avalon-MM) slave interface to UFM.

The UFM block also offers the following features.

Features	Capacity
Endurance	Counts to at least 10,000 program/erase cycles
Data retention (after 10,000 program/erase cycles)	• 20 years at 85 °C • 10 years at 100 °C

Maximum operating frequency	• Serial interface
	  • 10M02,10M04, 10M08, 10M16, 10M25: 7.25 MHz
	  • 10M40, 10M50: 4.81 MHz
	• Parallel interface
	  • 10M02: 7.25 MHz
	  • 10M04, 10M08, 10M16, 10M25, 10M40, 10M50: 116 MHz

Data length	Stores data of up to 32 bits length in parallel

Related Information

- **Utilizing the User Flash Memory (UFM) on Max 10 Devices with a Nios II Processor**
- **Putting Altera MAX Series in Hibernation Mode Using User Flash Memory**
- **MAX 10 User Flash Memory User Guide Archive** on page 6-1
  Provides a list of user guides for previous versions of the Altera On-Chip Memory IP core.

ISO 9001:2008 Registered

ALTERA
now part of Intel

The UFM architecture of MAX 10 devices is a combination of soft and hard IPs. You can only access the UFM using the Altera On-Chip Flash IP core in the Quartus® Prime software.

UFM and CFM Array Size

Each array is organized as various sectors.

A page is the smallest amount of flash memory that you can erase at one time. A sector contains a number of pages. You can erase each page or sector independently.

The Altera On-Chip Flash IP core also gives you access to configuration flash memory (CFM) based on your specification in the parameter editor.

Table 2-1: UFM and CFM Array Size

This table lists the dimensions of the UFM and CFM arrays.

Device	Pages per Sector					Page Size (Kb)	Total User Flash Memory Size (Kb) [1]	Total Configuration Memory Size (Kb) [1]
	UFM1	UFM0	CFM2 (Image 2)	CFM1 (Image 2)	CFM0 (Image 1)			
10M02	3	3	0	0	34	16	96	544
10M04	0	8	41	29	70	16	1,248	2,240
10M08	8	8	41	29	70	16	1,376	2,240
10M16	4	4	38	28	66	32	2,368	4,224
10M25	4	4	52	40	92	32	3,200	5,888
10M40	4	4	48	36	84	64	5,888	10,752
10M50	4	4	48	36	84	64	5,888	10,752

[1] The maximum possible value, which is dependent on the mode you select.

UFM Memory Organization Map

The address scheme changes based on the configuration mode you specify in the Altera On-Chip Flash parameter editor.

The following tables show the dynamic UFM support based on different configuration mode and MAX 10 variant.

Table 2-2: Dynamic Flash Size Support: Flash and Analog Variants

Configuration	UFM1	UFM0	CFM2 (Image 2)	CFM1 (Image 2)	CFM0 (Image 1)
Dual compressed images	UFM space	UFM space	—	—	—
Single uncompressed image	UFM space	UFM space	UFM space	—	—
Single compressed image	UFM space	UFM space	UFM space	UFM space	—
Single uncompressed image with memory initialization	UFM space	UFM space	—	—	—
Single compressed image with memory initialization	UFM space	UFM space	—	—	—
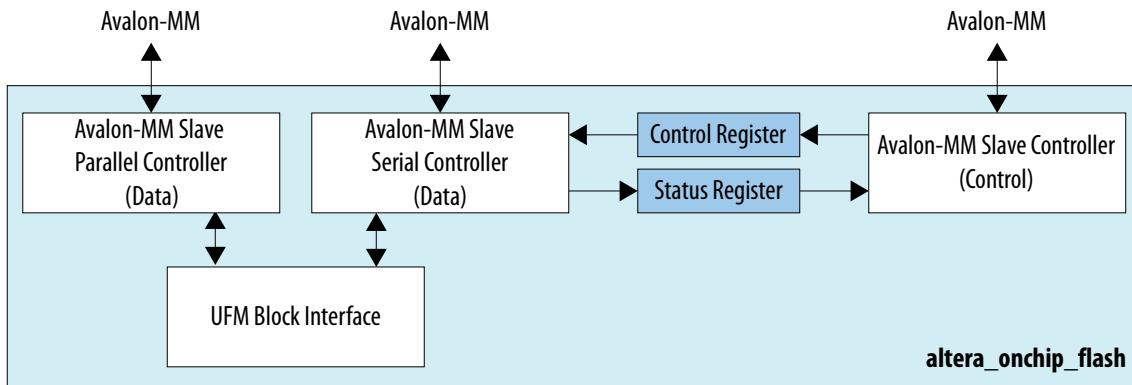
Table 2-3: Dynamic Flash Size Support: Compact Variant

Configuration	UFM1	UFM0	CFM2 (Image 2)	CFM1 (Image 2)	CFM0 (Image 1)
Dual compressed images	Not available				
Single uncompressed image	UFM space	UFM space	—	—	—
Single compressed image	UFM space	UFM space	—	—	—
Single uncompressed image with memory initialization	Not available				
Single compressed image with memory initialization	Not available				

UFM Block Diagrams

This figure shows the top level view of the Altera On-Chip Flash IP core block diagram. The Altera On-Chip Flash IP core supports both parallel and serial interfaces for MAX 10 FPGAs.

Figure 2-1: Altera On-Chip Flash IP Core Block Diagram
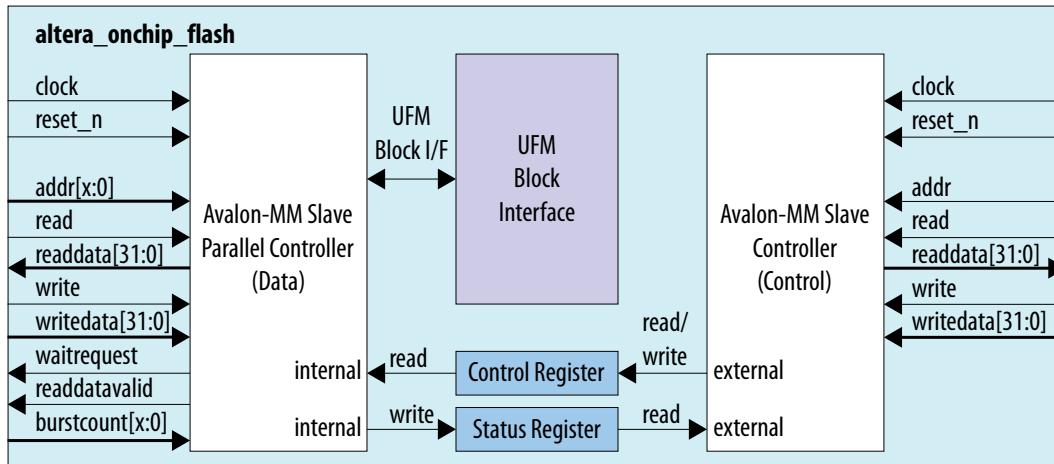
This IP block has two Avalon-MM slave controllers:

- Data—a wrapper of the UFM block that provides read and program accesses to the flash.
- Control—the CSR and status register for the flash, which is required only for program and erase operations.

These figures show the detailed overview of the Avalon-MM interface during read and program (write) operation.

Figure 2-2: Altera On-Chip Flash IP Core Avalon-MM Slave Read and Program (Write) Operation in Parallel Mode
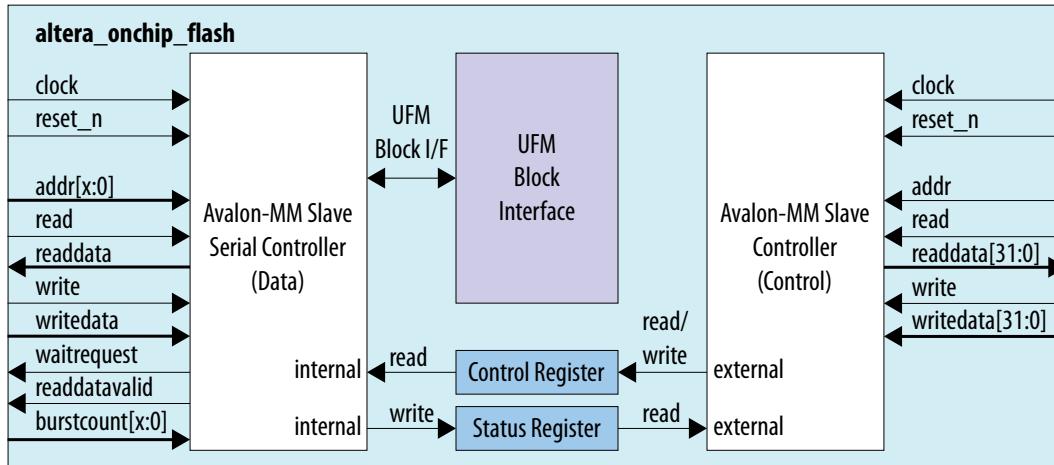
This figure shows the standard interface for MAX 10 devices in parallel mode.

Note: The maximum frequency for all devices in parallel mode, except for 10M02, is 116 MHz. The maximum frequency for 10M02 devices is 7.25 MHz.

Figure 2-3: Altera On-Chip Flash IP Core Avalon-MM Slave Read and Program (Write) Operation in Serial Mode

This figure shows the standard interface for MAX 10 devices in serial mode.

These figures show the detailed overview of the Avalon-MM interface during read only operation.

Figure 2-4: Altera On-Chip Flash IP Core Avalon-MM Slave Read Only Operation in Parallel Mode
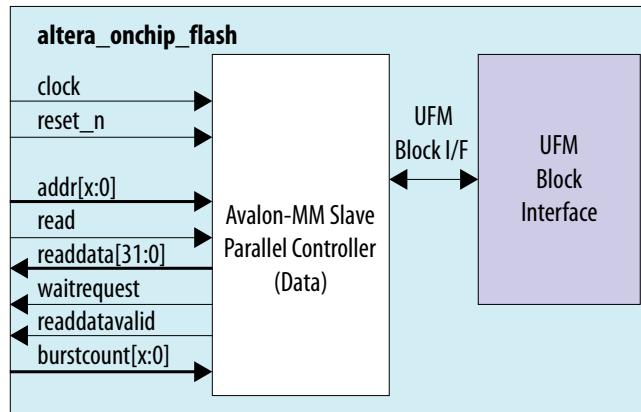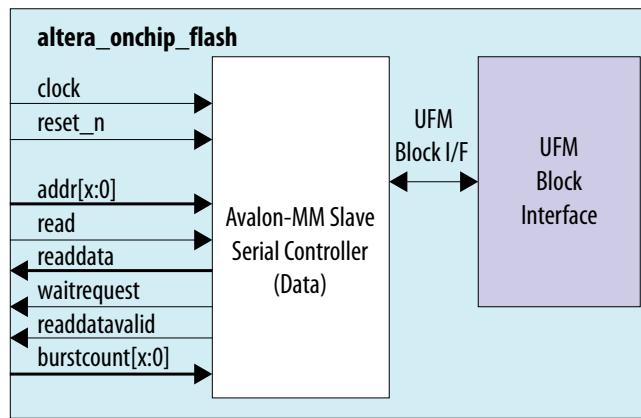
Figure 2-5: Altera On-Chip Flash IP Core Avalon-MM Slave Read Only Operation in Serial Mode

UFM Operating Modes

The UFM block offers the following operating modes:

- Read
- Burst read
- Program (Write)
- Sector erase
- Page erase
- Sector write protection

You can choose one of the following access modes in the Altera On-Chip Flash parameter editor to read and control the operations.

- Read and program mode—this mode allows both data and control slave interface. This mode is applicable for both UFM and CFM sectors.
- Read only mode—this mode allows only data slave interface, and restricted to only read operations. This mode is applicable for both UFM and CFM sectors.
- Hidden—this mode does not allow any read or program (write) operations. This mode is applicable only for CFM sectors.

The following table shows the comparison between parallel and serial modes.

Table 2-4: Comparison between Parallel Mode and Serial Mode

Feature	Parallel Mode	Serial Mode
Avalon-MM Data Interface	Parallel mode with 32-bit data bus	Serial mode with 32 bits based burst count
Access Mode	• Read and program • Read only • Hidden	• Read and program • Read only • Hidden
Read Mode	• Incrementing burst read • Wrapping burst read	Incrementing burst read only
Program (Write) Operation	Single 32-bit parallel program operation	Single 32-bit serial program operation

✉ **Subscribe** 💬 **Send Feedback**
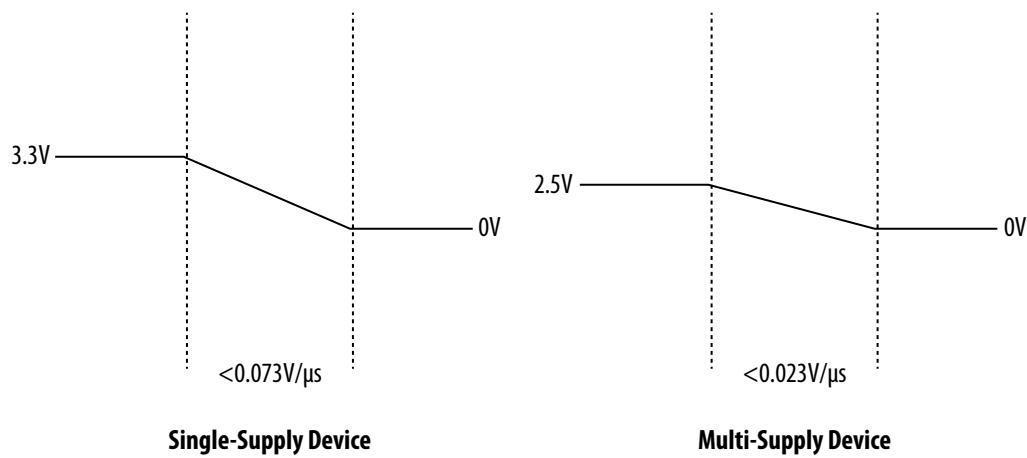
There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Guideline: UFM Power Supply Requirement

During UFM and CFM operations, make sure to follow the maximum slew rate requirement for power supply ramp down. This setting prevents device damage in case of power loss.

Table 3-1: Maximum Slew Rate Requirement

Device	Maximum Slew Rate
Single-supply device	0.073V/µs
Multi-supply device	0.023V/µs

Guideline: Program and Read UFM with JTAG

You can program UFM using JTAG interface version IEEE Standard 1149.1.

now part of Intel

The JTAG interface supports Jam™ Standard Test and Programming Language (STAPL) Format File (**.jam**), Programmer Object File (**.pof**), and JAM Byte Code File (**.jbc**).

You can use the Quartus Prime Programmer to program **.pof** through the JTAG interface. To program **.pof**, into the flash, follow these steps:

1. In the **Programmer** window, click **Hardware Setup**, and select **USB Blaster**.
2. In the **Mode** list, select **JTAG**.
3. Click **Auto Detect** on the left pane.
4. Select the device to be programmed, and click **Add File**.
5. Select the **.pof** to be programmed to the selected device.
6. Select the **UFM** in the **Program/Configure** column.
7. Click **Start** to start programming.

To program through **.jam** or **.jbc** files, refer to the *Using the Command-Line Jam STAPL Solution for Device Programming* application note.

Related Information
AN 425: Using the Command-Line Jam STAPL Solution for Device Programming

Guideline: UFM Content Initialization

You can initialize the UFM content using Altera software.

The initial memory content supports Memory Initialization File (**.mif**), and Hexadecimal (Intel-Format) File (**.hex**).

You can initialize the UFM content using either one of the following ways:

- Set the initial memory content through the Altera On-Chip Flash IP core.
- Set the initial memory content through the **Convert Programming File** tool in the Quartus Prime software when you convert **.sof** to **.pof**.

Guideline: Erase Before Program

Make sure to erase the flash location before you perform a program (write) operation.

Related Information

- **Utilizing the User Flash Memory (UFM) on Max 10 Devices with a Nios II Processor**
- **Putting Altera MAX Series in Hibernation Mode Using User Flash Memory**

Altera On-Chip Flash IP Core

The Altera IP core design flow helps you get started with any Altera IP core.

The Altera On-Chip Flash IP core is installed as part of the Quartus Prime installation process. You can select and parameterize any Altera IP core from the library. Altera provides an integrated parameter editor that allows you to customize the Altera On-Chip Flash IP core to support a wide variety of applications. The parameter editor guides you through the setting of parameter values and selection of optional ports.

Related Information

Introduction to Altera IP Cores
Provides more information about Altera IP cores.

UFM Avalon-MM Operating Modes

The UFM operating modes use Avalon-MM interface.

UFM Read Status and Control Register
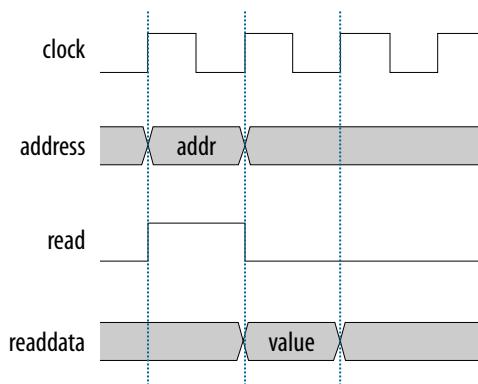
You can access the control register value through the Avalon-MM control slave interface.

Figure 4-1: Read Status and Control Register

The figure below shows the timing diagram for the read status and control register.

To use the control register, assert the `read` signal and send the control register address to the control slave address.
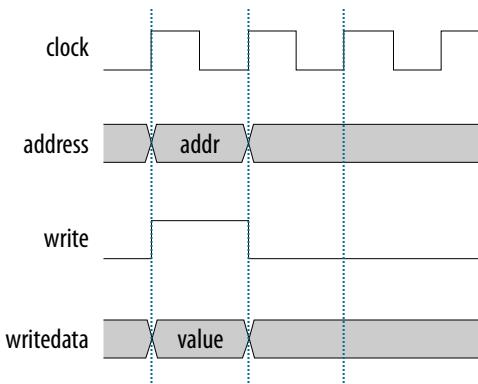
The flash IP core then sends the register value through the `readdata` bus.

UFM Write Control Register

You can program (write) the control register value through Avalon-MM control slave interface.

Figure 4-2: Program (Write) Control Register

The figure below shows the timing diagram for the program control register.

To program the control register, assert the `write` signal.

The flash IP core then sends address `0×01` (control register) and `writedata` (register value) to control the slave interface.

UFM Program (Write) Operation

The UFM offers a single 32-bit program (write) operation.

To perform a UFM program operation, follow these steps:

1. Disable the write protection mode. Write 0 into the write protection register for the sector of the given data through the Avalon-MM control interface.
2. Program the following data into flash through the Avalon-MM data interface.

   - Address: legal address (from Avalon-MM address map)
   - Data: user data

   Set burst count to 1 (parallel mode) or 32 (serial mode).
3. The flash IP core sets the `busy` field in the status register to `2'b10` when the program operation is in progress.
4. If the operation goes well, the flash IP core sets the write successful field in the status register to `1'b1` or write successful. The flash IP core sets the write successful field in the status register to `1'b0` (failed) if one of the following conditions takes place:

   - The burst count is not equal to 1 (parallel mode) or 32 (serial mode).
   - The given address is out of range.
   - The sector protection mode or write protection mode of the corresponding sector is not clear (the value is not `1'b0`).
5. Repeat the earlier steps if you want to perform another program operation.
6. You have to enable back the write protection mode when the program operation completes. Write 1 into the write protection register for the corresponding sector through the Avalon-MM control interface.

   Note: Check the status register after each write to make sure the program operation is successful (write successful).

Figure 4-3: Program Operation in Parallel Mode

The figure below shows the write data timing diagram in parallel mode.
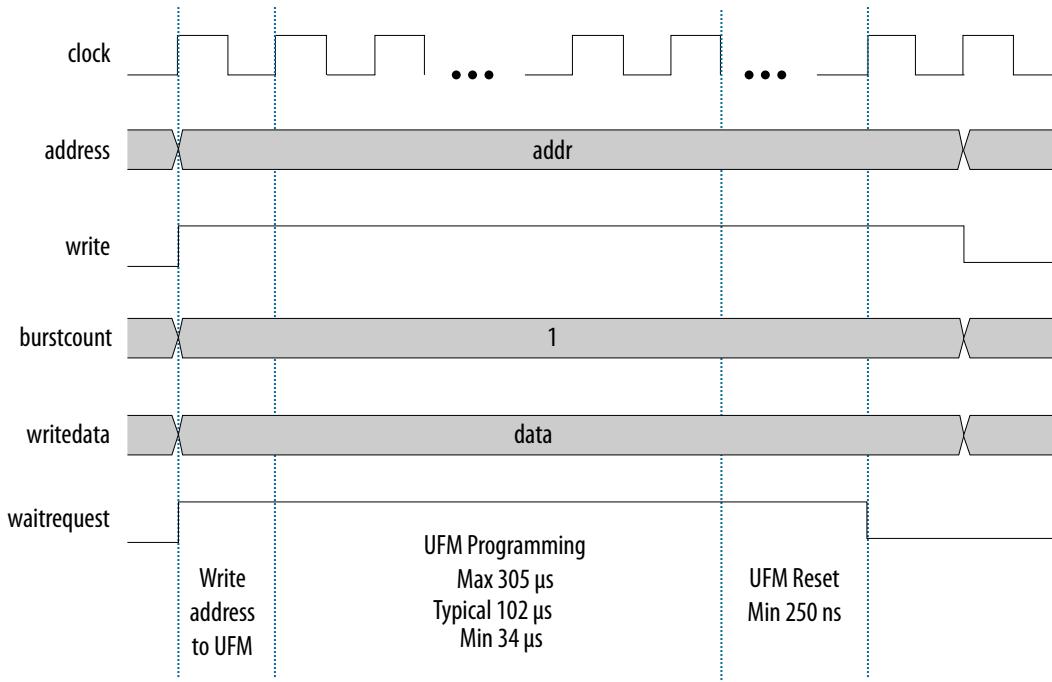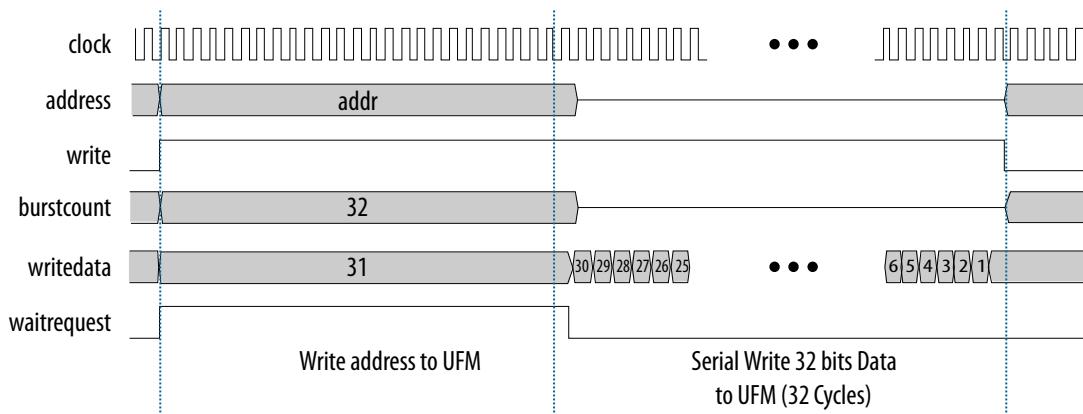
Figure 4-4: Program Operation in Serial Mode

The figure below shows the write data timing diagram in serial mode.

UFM Sector Erase Operation

The sector erase operation allows the UFM to erase by sectors.

To perform a UFM sector erase operation, follow these steps:

1. Disable the write protection mode. Write 0 into the write protection register for the sector through the Avalon-MM control interface.
2. Write the appropriate bits into the control register to select the sector erase location. The flash IP core stores the sector erase address and initiates the sector erase operation.

   Note: The IP core only accepts the sector erase address when it is in IDLE state; busy field at status register is 2'b00. If the IP core is busy, it will ignore the sector erase address.

3. The flash IP core sets the busy field in the status register to 2'b01 when the erase operation is in progress.
4. The flash IP core then asserts the waitrequest signal if there are any new incoming read or write commands from the data interface.
5. The flash IP core erases the sector. It stores the physical flash erase result in the erase successful field in the status register when the sector erase operation completes.

   Note: The maximum erase time is 350 ms.

6. The flash IP core sets the erase successful field in the status register to 1'b0 (failed) if one of the following conditions takes place:

   - You send an illegal sector number.
   - The sector protection mode or write protection mode of the corresponding sector is not clear (the value is not 1'b0).

7. Repeat the earlier steps if you want to perform another sector erase operation.
8. You have to enable back the write protection mode when the sector erase operation completes. Write 1 into the write protection register for the corresponding sector through the Avalon-MM control interface.

   Note: Check the status register after each erase to make sure the erase operation is successful (erase successful).

UFM Page Erase Operation

The page erase operation allows the UFM to erase by pages.

To perform a UFM page erase operation, follow these steps:

1. Disable the write protection mode. Write 0 into the write protection register for the sector through the Avalon-MM control interface.
2. Write the appropriate bits into the control register to select the page erase location. The flash IP core stores the page erase address and initiates the page erase operation.

    Note: The IP core only accepts the page erase address when the IP is in IDLE state; `busy` field at status register is `2'b00`. If the IP core is busy, it will ignore the page erase address.

3. The flash IP core sets the `busy` field in the status register to `2'b01` when the erase operation is in progress.
4. The flash IP core then asserts the `waitrequest` signal if there are any new incoming read or write commands from the data interface.
5. The flash IP core erases the page. It stores the physical flash erase result in the erase successful field in the status register when the page erase operation completes.

    Note: The maximum erase time is 350 ms.

6. The flash IP core sets the erase successful field in the status register to `1b'0` (failed) if you send an illegal address.
7. Repeat the earlier steps if you want to perform another page erase operation.
8. You have to enable back the write protection mode when the page erase operation completes. Write 1 into the write protection register for the corresponding page through the Avalon-MM control interface.

    Note: Check the status register after each erase to make sure the erase operation is successful (erase successful).

UFM Read Operation

The UFM offers a single 32-bit read operation.

To perform a read operation, the address register must be loaded with the reference address where the data is or is going to be located in the UFM.

To perform a UFM read operation, follow these steps:

1. Assert the `read` signal to send the legal data address to the data slave interface.
2. Set the burst count to 1 (parallel mode) or 32 (serial mode).
3. The flash IP core asserts the `waitrequest` signal when it is busy.
4. The flash IP core asserts the `readdatavalid` signal and sends the data through the `readdata` bus.
5. The flash IP core sets the `busy` field in the status register to `2'b11` when the read operation is in progress.
6. If the operation goes well, the flash IP core sets the read successful field in the status register to `1'b1` or read successful. It sets the read successful field in the status register to `1'b0` (failed) and returns empty flash if you try to read from an illegal address or protected sector.

The following figures show the timing diagrams for the read operations for the different MAX 10 devices in parallel and serial modes.
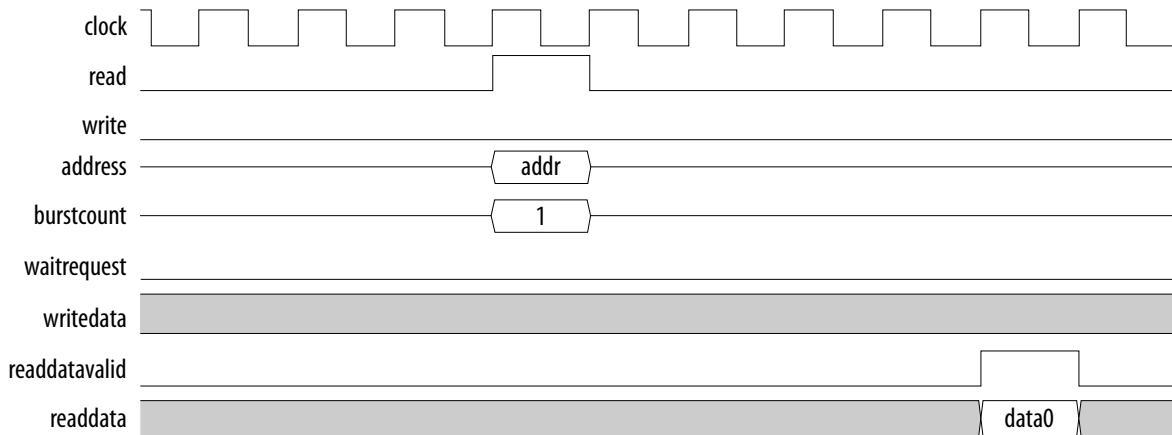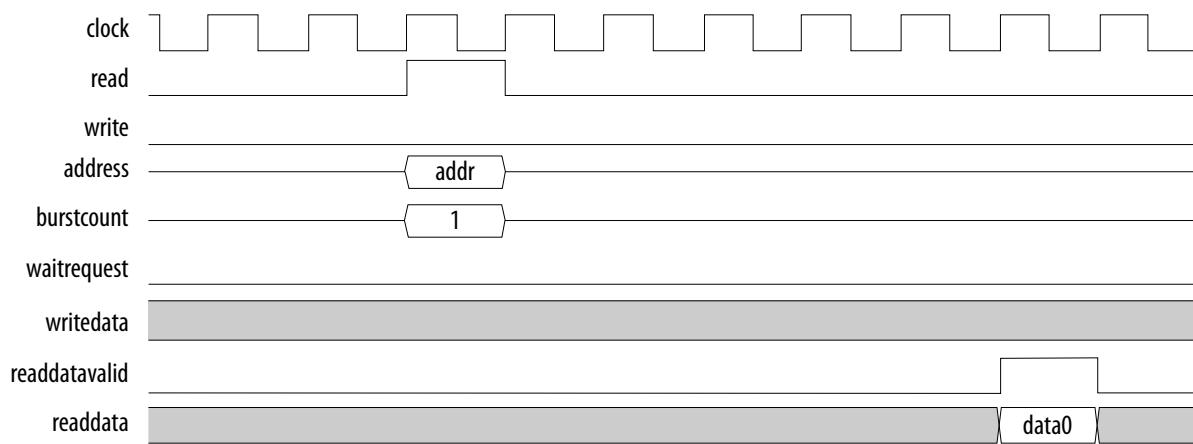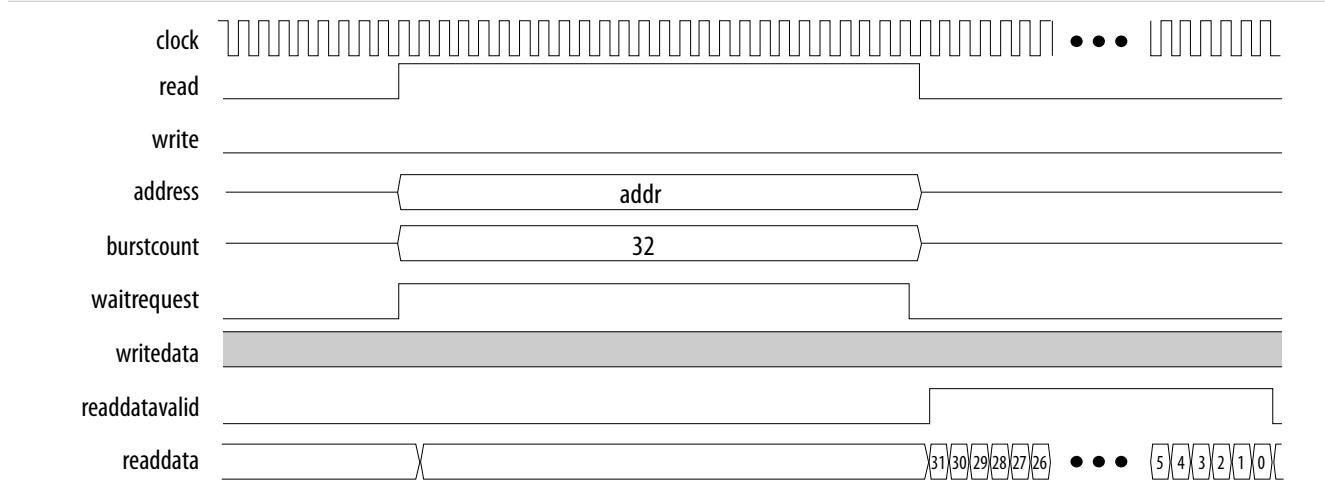
Figure 4-5: Read Operation for 10M04, 10M08, 10M16 and 10M25 Devices in Parallel Mode

Figure 4-6: Read Operation for 10M40 and 10M50 Devices in Parallel Mode

Figure 4-7: Read Operation for MAX 10 Devices in Serial Mode

UFM Burst Read Operation

The burst read operation is a streaming 32-bit read operation.

The burst read operation offers the following modes:

- Data incrementing burst read—allows a maximum of 128 burst counts.
- Data wrapping burst read—has fixed burst counts of 2 (10M04/08) and 4 (10M16/25/40/50)

To perform a UFM burst read operation, follow these steps:

1. Assert the `read` signal and send the legal burst count and legal data addresses to the data interface.
2. The flash IP core asserts the `waitrequest` signal when it is busy.
3. The flash IP core then asserts the `readdatavalid` signal and sends the data through the `readdata` bus.

   Note: For data wrapping burst read operation, if the address reaches the end of the flash, it wraps back to the beginning of the flash and continues reading.
4. The flash IP core sets the `busy` field in the status register to `2'b11` or `busy_read` when the read operation is in progress.
5. If the operation goes well, the flash IP core sets the read successful field in the status register to `1'b1` or read successful. It sets the read successful field in the status register to `1'b0` (failed) and changes empty flash to 1 if you try to read from an illegal address or protected sector.

UFM Data Incrementing Burst Read

The following figures show the timing diagrams for the data incrementing burst read operations for the different MAX 10 devices.
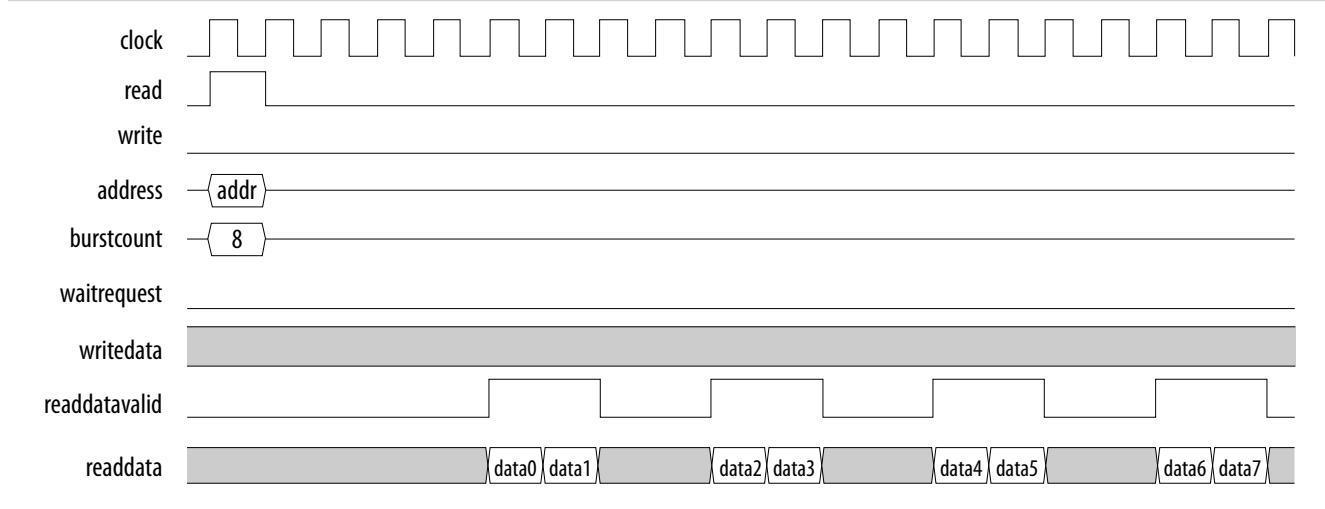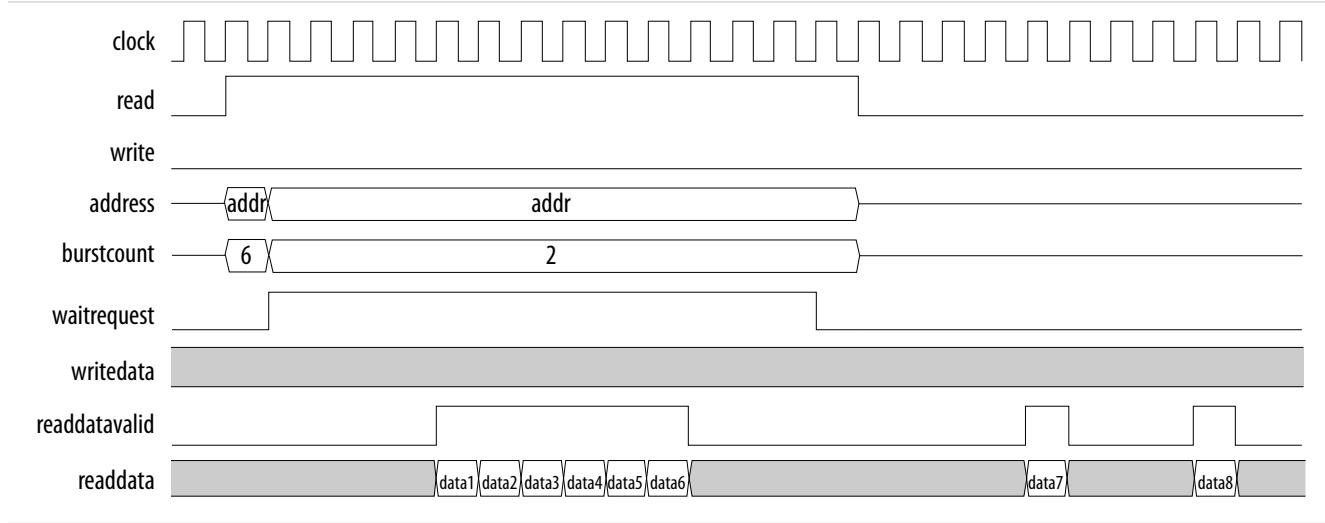
Figure 4-8: Incrementing Burst Read Operation for 10M04 and 10M08 Devices in Parallel Mode

Figure 4-9: Incrementing Burst Read Operation for 10M16 and 10M25 Devices in Parallel Mode
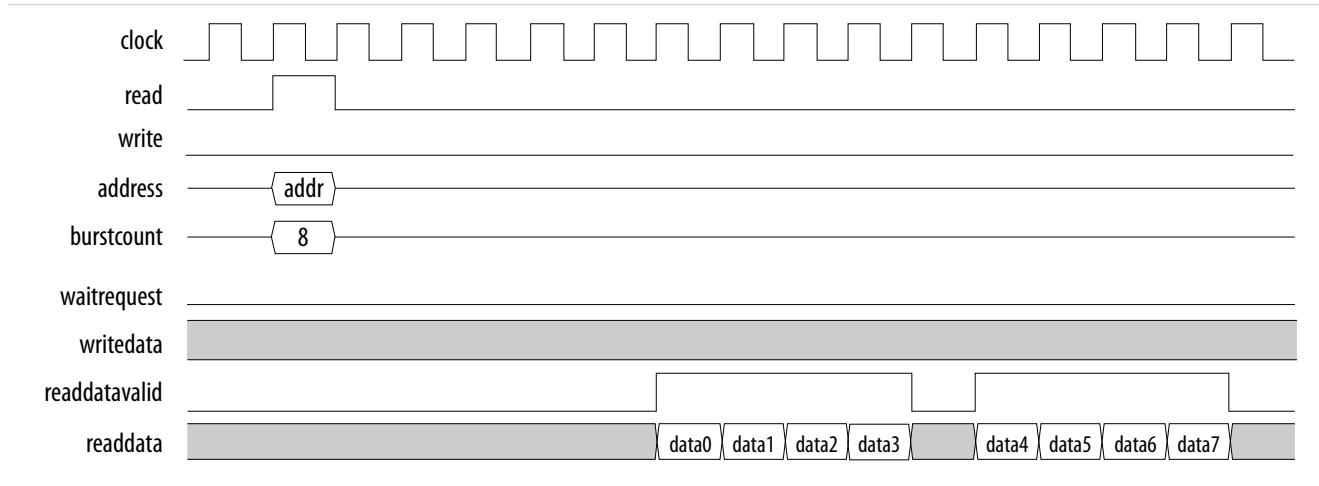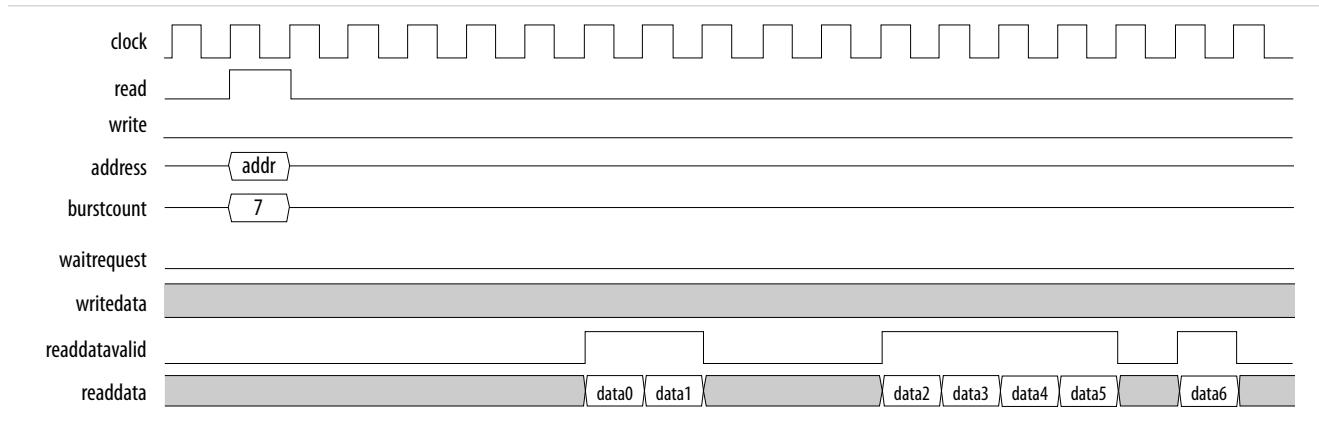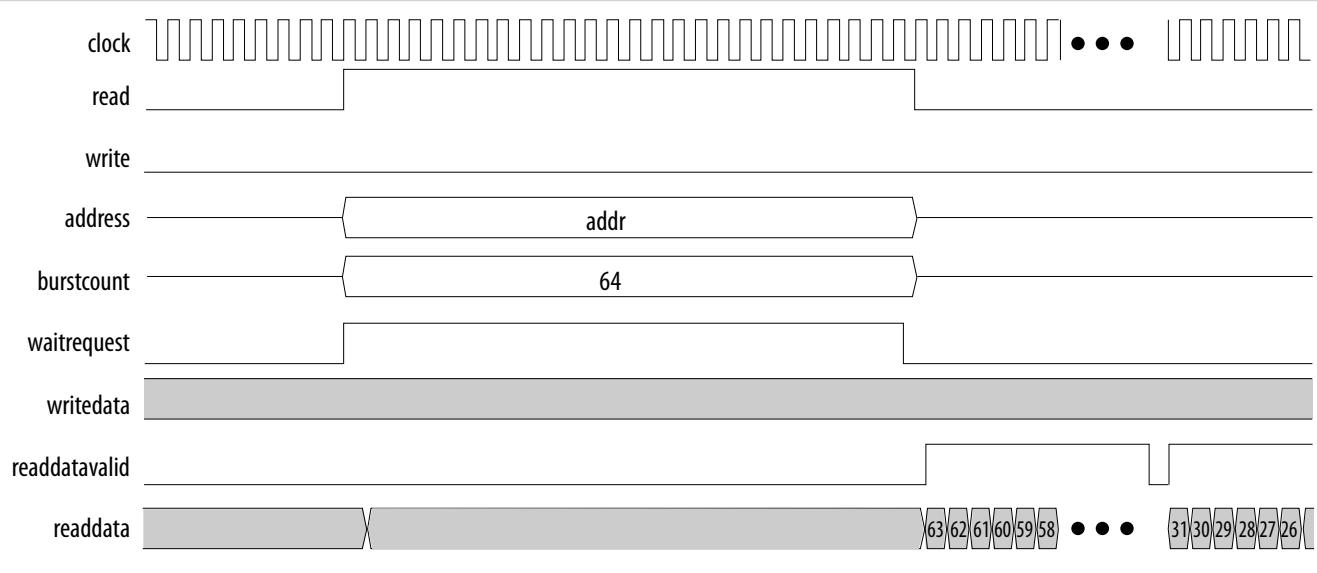
Figure 4-10: Incrementing Burst Read Operation for 10M50 Devices in Parallel Mode

Figure 4-11: Unaligned Address Incrementing Burst Read Operation for 10M50 Devices in Parallel Mode

Send Feedback

Figure 4-12: Incrementing Burst Read Operation for MAX 10 Devices in Serial Mode

UFM Data Wrapping Burst Read

The UFM supports data wrapping when it receives an unaligned address.

Note: Wrapping burst read is available only for parallel interface.

Table 4-1: Data Wrapping Support for MAX 10 Devices

Device	Data Register Length	Flash IP Data Bus Width	Fixed Supported Burst Count	Data Wrapping
10M04, or 10M08	32	64	2	The address wraps back to the previous boundary after 64 bits or 2 cycles. For example, for a wrapping in a 32-bit data interface: 1. Start address is 0×01 2. Address sequence will be 0×01, then back to address 0×00
10M16, 10M25, 10M40, or 10M50	32	128	4	The address wraps back to the previous boundary after 128 bits or 4 cycles. For example, for a wrapping in a 32-bit data interface: 1. Start address is 0×02 2. Address sequence will be 0×02 and 0×03, then back to address 0×00 and 0×01

The following figures show the timing diagrams for the data wrapping burst read operations for the different MAX 10 devices.

Figure 4-13: Wrapping Burst Read Operation for 10M04 and 10M08 Devices
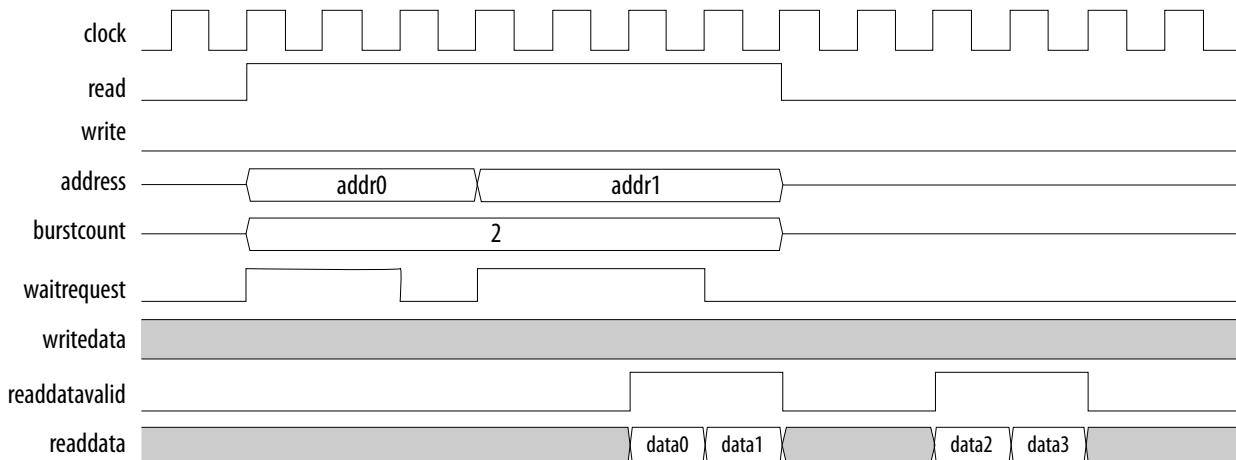
Figure 4-14: Wrapping Burst Read Operation for 10M16 and 10M25 Devices
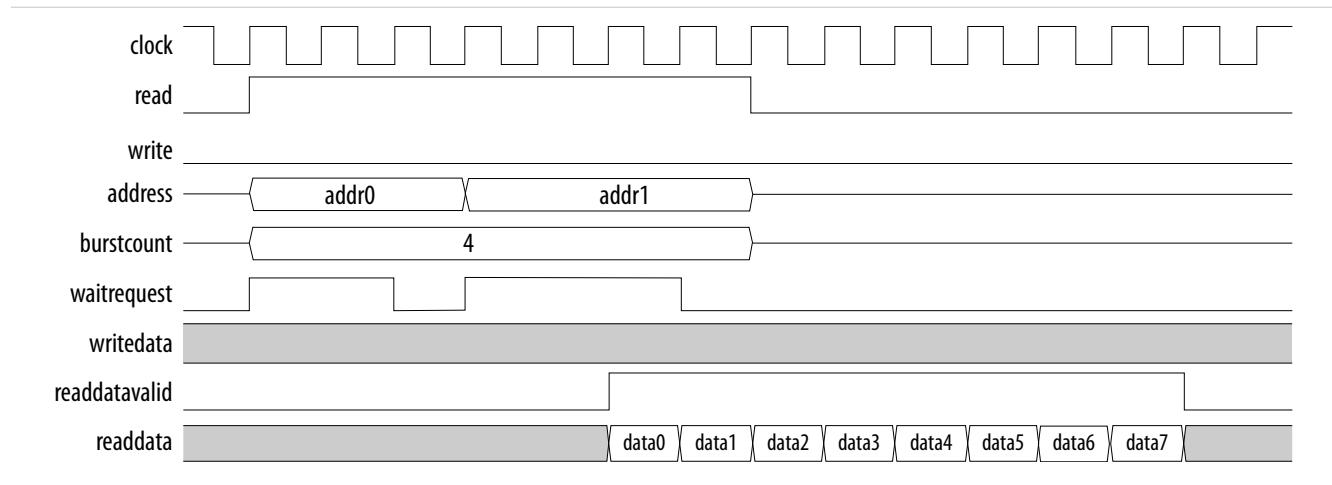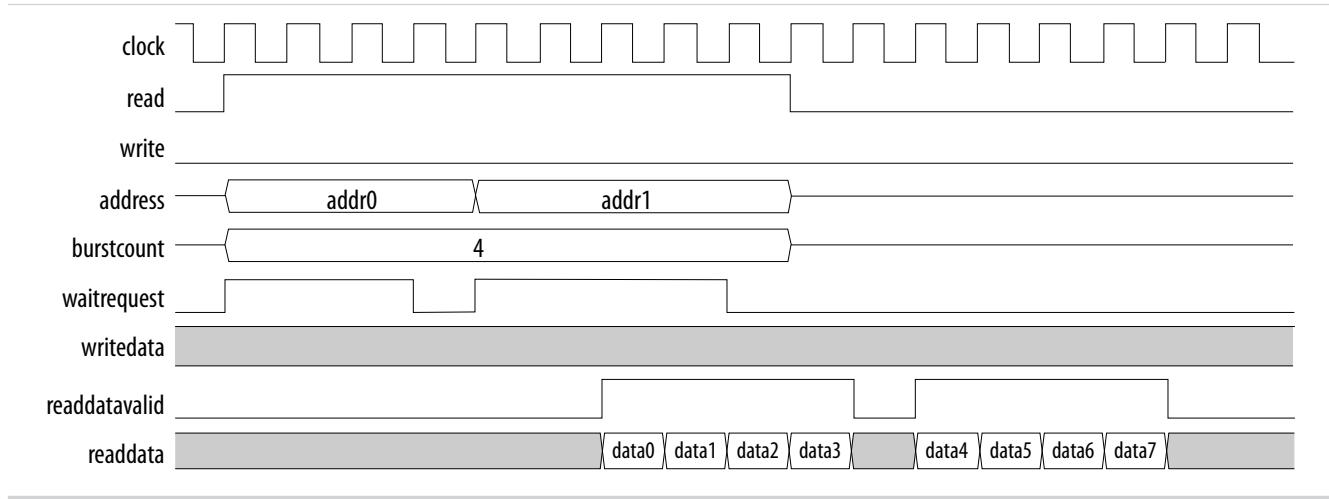
Figure 4-15: Wrapping Burst Read Operation for 10M40 and 10M50 Devices

Flash Initialization Files

The On-Chip Flash IP core supports the **.hex**, **.mif**, and **.dat** files.

If the total data size in the initialization file is less the maximum UFM size, the IP core retains blank data (all 1's).

If the total data size in the initialization file is larger than the maximum UFM size, the IP core ignores the extra data.

Table 4-2: Types of Flash Initialization File Supported

File Type	Format	Notes
.hex	Standard Intel hexadecimal file—uses byte addressing.	For flash initialization in actual hardware.
.mif	Standard Altera memory initialization file—uses word addressing.	For flash initialization in actual hardware.
.dat	32-bit data width file—uses word addressing.	For flash initialization in simulation model.

This section provides information about the Altera On-Chip Flash IP Core parameters, signals, and registers.

Altera On-Chip Flash Parameters

The following table lists the parameters for the Altera On-Chip Flash IP core.

Table 5-1: Altera On-Chip Flash IP Core Parameters

Parameters	Default Value	Description
Data interface	Parallel	Allows you to select the type of interface. You can choose parallel or serial.
Read burst mode	Incrementing	Allows you to select the type of read burst mode. You can choose incrementing or wrapping. | Incrementing mode	Read burst count is 2, 4, 8, … 128
Wrapping mode	Burst count fixed to 2 or 4	 **Note:** Serial interface supports only incrementing mode. Parallel interface does not support wrapping mode for 10M02 devices.
Read burst count	2	Allows you the flexibility to adjust the maximum burst count bus width. • Parallel mode: This setting represents the maximum burst count number. • Serial mode: This setting supports stream read and represents the words to be read for each read operation. The Avalon-MM interface burst count bus width is equal to 32*read burst count.

Parameters	Default Value	Description
Configuration mode	Single uncompressed image	Allows you to select the configuration mode. You can choose one of these options: • Dual compressed images • Single uncompressed image: Accesses CFM2 sector as UFM • Single compressed image: Accesses CFM2 and CFM1 sectors as UFM • Single uncompressed image with memory initialization • Single compressed image with memory initialization
Flash Memory	—	The sector ID, address range value, and flash type are generated dynamically by hardware **.tcl** based on the device and configuration mode you select. Indicates the address mapping for each sector and adjusts the **Access Mode** for each sector individually. **Note:** Only CFM sectors support **Hidden** access mode.
Clock frequency	116.0 MHz	Key in the appropriate clock frequency in MHz. The maximum frequency is 116.0 MHz for parallel interface and 7.25 MHz for serial interface. **Note:** If you use 10M02 devices, the maximum frequency for parallel interface is 7.25 MHz.
Initialize flash content	Off	Turn on this option to initialize the flash content.
Enable non-default initialization file	Off	Turn on this option to enable your preferred initialization file. If you choose to have a non-default file, type the filename or select the **.hex** or **.mif** file using the browse button.
User created hex or mif file	—	This option is only available if you turn on **Enable non-default initialization file**. Assign your own **.hex** or **.mif** filename.
User created dat file for simulation	—	This option is only available if you turn on **Enable non-default initialization file**. Assign your own simulation filename.

Altera On-Chip Flash Signals

The following table lists the signals for the Altera On-Chip Flash IP core.

Table 5-2: Avalon-MM Slave Input and Output Signals for Parallel and Serial Modes.

Signal	Width	Direction	Description
Clock and Reset			
clock	1	Input	System clock signal that clocks the entire peripheral.
reset_n	1	Input	System synchronous reset signal that resets the entire peripheral. The IP core asserts this signal asynchronously. This signal becomes synchronous in the IP core after the rising edge of the clock.
Control			
avmm_csr_addr	1	Input	Avalon-MM address bus that decodes registers.
avmm_csr_read	1	Input	Avalon-MM read control signal. The IP core asserts this signal to indicate a read transfer. If present, the readdata signal is required.
avmm_csr_readdata	32	Output	Avalon-MM read back data signal. The IP core asserts this signal during read cycles.
avmm_csr_write	1	Input	Avalon-MM write control signal. The IP core asserts this signal to indicate a write transfer. If present, the writedata signal is required.
avmm_csr_writedata	32	Input	Avalon-MM write data bus. The bus master asserts this bus during write cycles.
Data			
avmm_data_addr	User-defined	Input	Avalon-MM address bus that indicates the flash data address. The width of this address depends on your selection of device and configuration mode.
avmm_data_read	1	Input	Avalon-MM read control signal. The IP core asserts this signal to indicate a read transfer. If present, the readdata signal is required.
avmm_data_readdata	• Parallel mode: 32 • Serial mode: 1	Output	Avalon-MM read back data signal. The IP core asserts this signal during read cycles.
avmm_data_write	1	Input	Avalon-MM write control signal. The IP core asserts this signal to indicate a write transfer. If present, the writedata signal is required.
avmm_data_writedata	• Parallel mode: 32 • Serial mode: 1	Input	Avalon-MM write data bus. The bus master asserts this bus during write cycles.
avmm_data_ waitrequest	1	Output	The IP core asserts this bus to pause the master when the IP core is busy during read or write operations.

Send Feedback

Signal	Width	Direction	Description
`avmm_data_readdata-valid`	1	Output	The IP core asserts this signal when the `readdata` signal is valid during read cycles.
`avmm_data_burstcount`	User-defined	Input	The bus master asserts this signal to initiate a burst read operation. • In write operations, the burst count is always fixed to 1 for parallel mode and 32 for serial mode. • In incrementing burst read mode, the supported read burst count range: | Parallel mode	$1\text{-}2^{(\text{burstcount width-1})}$
Serial mode	1-128*32	 • In wrapping burst read mode (parallel mode only), the supported read burst count is fixed to 2 and 4. 	10M04, and 10M08	1–2
10M16, 10M25, 10M40 and 10M50	1–4	

Altera On-Chip Flash Registers

The following table lists the address mapping and registers for the Altera On-Chip Flash IP core.

Table 5-3: Altera On-Chip Flash Control Address Mapping

Register	Address	Access	Description
Status Register	0×00	Read only	Stores the status and result of recent operations and sector protection mode.
Control Register	0×01	Read/Program	Stores the following information: • Page erase address • Sector erase address • Sector write protection mode

Table 5-4: Altera On-Chip Flash Status Register

Bit Offset	Field	Default Value	Description
1–0	busy	2'b00	2'b00 IDLE 2'b01 BUSY_ERASE 2'b10 BUSY_WRITE 2'b11 BUSY_READ

Bit Offset	Field	Default Value	Description
2	rs (read successful)	1'b0	1'b0 Read failed 1'b1 Read successful
3	ws (write successful)	1'b0	1'b0 Write failed 1'b1 Write successful
4	es (erase successful)	1'b0	1'b0 Erase failed 1'b1 Erase successful
5	sp (UFM1 protection bit)	—	The IP core sets these bits based on the specified device and configuration mode. If the IP core sets one of these bits, you cannot read or program on the specified sector.
6	sp (UFM0 protection bit)	—	
7	sp (CFM2 protection bit)	—	
8	sp (CFM1 protection bit)	—	
9	sp (CFM0 protection bit)	—	
31–10	dummy (padding)	—	All of these bits are set to 1.

Table 5-5: Altera On-Chip Flash Control Register

Bit Offset	Field	Default Value	Description
19–0	pe (page erase address)	All 1's	Sets the page erase address to initiate a page erase operation. The IP core only accepts the page erase address when it is in IDLE state. Otherwise, the page address will be ignored. The legal value is any available address. The IP core erases the corresponding page of the given address.

Bit Offset	Field	Default Value	Description
22–20	se (sector erase address)	3'b111	Sets the sector erase address to initiate a sector erase operation. The IP core only accepts the sector erase address when it is in IDLE state. Otherwise, the page address will be ignored. <table><tr><td>3'b001</td><td>UFM1</td></tr><tr><td>3'b010</td><td>UFM0</td></tr><tr><td>3'b011</td><td>CFM2</td></tr><tr><td>3'b100</td><td>CFM1</td></tr><tr><td>3'b101</td><td>CFM0</td></tr><tr><td>3'b111</td><td>Not set</td></tr><tr><td>Other values</td><td>Illegal address</td></tr></table> **Note:** If you set both sector address and page address at the same time, the sector erase address gets the priority. The IP core accepts and executes the sector erase address and ignores the page erase address.
23	wp (UFM1 write protection)	1	The IP core uses these bits to protect the sector from write and erase operation. You must clear the corresponding sector write protection bit before your program or erase the sector. <table><tr><td>1'b0</td><td>Disable write protected mode</td></tr><tr><td>1'b1</td><td>Enable write protected mode</td></tr></table>
24	wp (UFM0 write protection)	1	
25	wp (CFM2 write protection)	1	
26	wp (CFM1 write protection)	1	
27	wp (CFM0 write protection)	1	
31–28	dummy (padding)	—	All of these bits are set to 1.

MAX 10 User Flash Memory User Guide Archive

A

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 User Flash Memory User Guide**
15.0	**MAX 10 User Flash Memory User Guide**
14.1	**MAX 10 User Flash Memory User Guide**

ISO 9001:2008 Registered

ALTERA
now part of Intel

2016.05.02

UG-M10UFM ✉ **Subscribe** 💬 **Send Feedback**

Date	Version	Changes
May 2016	2016.05.02	• Added the typical and minimum UFM programming time in parallel mode. • Corrected the minimum UFM reset time in parallel mode to 250 ns. • Added links to archived versions of the *MAX 10 User Flash Memory User Guide*.
November 2015	2015.11.02	• Added information about the supported flash initialization files. • Added serial interface support for 10M40 and 10M50 devices. The maximum frequency for MAX 10 devices is 7.25 MHz, except for 10M40 and 10M50 devices, which is 4.81 MHz. • Added parallel interface support for 10M02 devices. The maximum frequency for MAX 10 devices is 116 MHz, except for 10M02 devices, which is 7.25 MHz. • Changed instances of *Quartus II* to *Quartus Prime*.
May 2015	2015.05.04	• Changed *write* to industry-standard term *program*. • Added a note to the *UFM and CFM Array Size* section that the total UFM size is the maximum possible value, which is dependent on the selected mode. • Added design consideration information about the maximum slew rate requirement for power supply ramp down. • Added design consideration information about erasing the flash location before performing a program operation.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Date	Version	Changes

December 2014	2014.12.15	• Added support for serial interface.
		• Added maximum operating frequency of 7.25 MHz for serial interface.
		• Updated the UFM block diagram to include serial interface.
		• Added design consideration information about creating initial memory content using the IP core, and programming UFM using JTAG interface version IEEE Standard 1149.1.
		• Added new timing diagrams for read and write operations in serial mode.
		• Added information for the new serial interface related GUI parameters, signals, and registers.
		• Added information for the following new Avalon-MM slave interface signals for serial mode: `addr`, `read`, `readdata`, `write`, `writedata`, `waitrequest`, `readdatavalid`, and `burstcount`.
		• Added information for the following new parameters:
		
		  • **Data Interface** that allows you to choose between **Parallel** and **Serial** interface.
		  • **Configuration Scheme** and **Configuration Mode** that replace **Dual Images**. The new parameters include all supported configuration modes.
		  • **Read Burst Count** that allows the burst count width to be auto-adjusted.

September 2014	2014.09.22	Initial release.

MAX 10 JTAG Boundary-Scan Testing
User Guide

ALTERA®

Contents

2015.05.04

MAX® 10 devices support the IEEE Std.1149.1 (JTAG) boundary-scan testing (BST).

When you perform BST, you can test pin connections without using physical test probes and capture functional data during normal operation. The boundary-scan cells (BSCs) in a device can force signals onto pins, or capture data from pin or core logic signals. Forced test data is serially shifted into the BSCs. Captured data is serially shifted out and externally compared to expected results.

Note: You can perform BST on MAX 10 devices before, after, and during configuration.

Related Information

- **MAX 10 FPGA Configuration User Guide**
  Provides more information about JTAG in-system programming.
- **JTAG BST Architecture** on page 2-1
- **JTAG Boundary-Scan Register** on page 2-2
- **JTAG BST Operation Control** on page 3-1
- **I/O Voltage Support in the JTAG Chain** on page 4-1
- **Enabling and Disabling JTAG BST Circuitry** on page 5-1
- **Guidelines for JTAG BST** on page 6-1
- **Boundary-Scan Description Language Support** on page 7-1

ISO 9001:2008 Registered

MAX 10 JTAG interface uses four pins, TDI, TDO, TMS, and TCK.

JTAG Pins

Table 2-1: JTAG Pin Descriptions

Pin	Function	Description
TDI	Serial input pin for: • instructions • test data • programming data	• TDI is sampled on the rising edge of TCK • TDI pins have internal weak pull-up resistors.
TDO	Serial output pin for: • instructions • test data • programming data	• TDO is sampled on the falling edge of TCK • The pin is tri-stated if data is not being shifted out of the device.
TMS	Input pin that provides the control signal to determine the transitions of the TAP controller state machine.	• TMS is sampled on the rising edge of TCK • TMS pins have internal weak pull-up resistors.
TCK	The clock input to the BST circuitry.	—

All the JTAG pins are powered by the V_{CCIO} 1B. In JTAG mode, the I/O pins support the LVTTL/LVCMOS 3.3-1.5V standards.

JTAG Circuitry Functional Model

The JTAG BST circuitry requires the following registers:
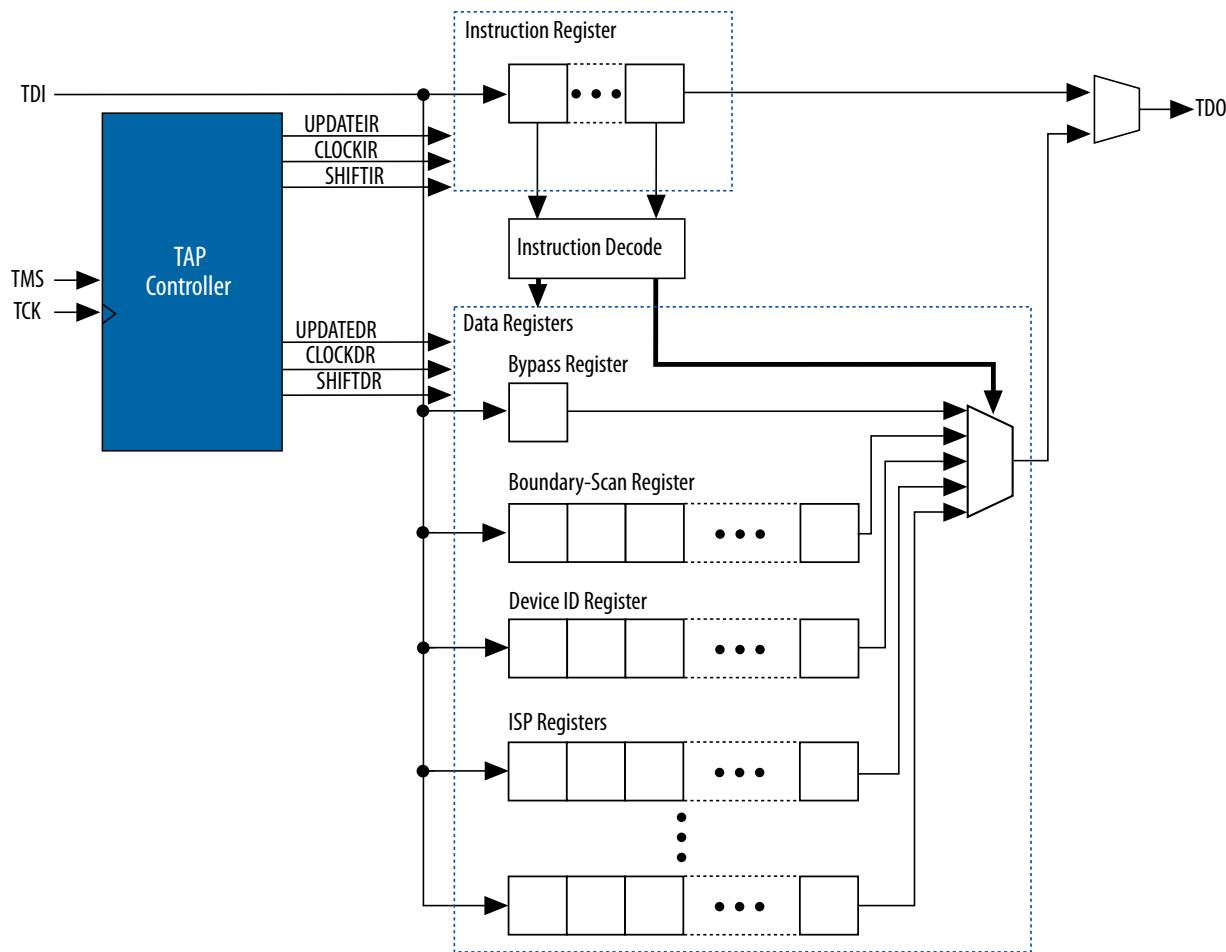
- Instruction register—determines which action to perform and which data register to access.
- Bypass register (1-bit long data register)—provides a minimum-length serial path between the TDI and TDO pins.
- Boundary-scan register—shift register composed of all the BSCs of the device.

Figure 2-1: JTAG Circuitry Functional Model

- Test access port (TAP) controller—controls the JTAG BST.
- TMS and TCK pins—operate the TAP controller.
- TDI and TDO pins—provide the serial path for the data registers.
- The TDI pin also provides data to the instruction register to generate the control logic for the data registers.

JTAG Boundary-Scan Register

You can use the boundary-scan register to test external pin connections or to capture internal data. The boundary-scan register is a large serial shift register that uses the TDI pin as an input and the TDO pin as an output. The boundary-scan register consists of 3-bit peripheral elements that are associated with MAX 10 I/O pins.

Boundary-Scan Cells in MAX 10 I/O Pin

The MAX 10 3-bit BSC contains the following registers:

- Capture registers—connect to internal device data through OUTJ, OEJ, and PIN_IN signals.
- Update registers—connect to external data through PIN_OUT and PIN_OE signals.

Figure 2-2: User I/O BSC with JTAG BST Circuitry for MAX 10 Devices

The TAP controller generates the global control signals internally for the JTAG BST registers, shift, clock, and update. The instruction register generates the MODE signal.

The data signal path for the boundary-scan register runs from the serial data in (SDI) signal to the serial data out (SDO) signal. The scan register begins at the TDI pin and ends at the TDO pin of the device.
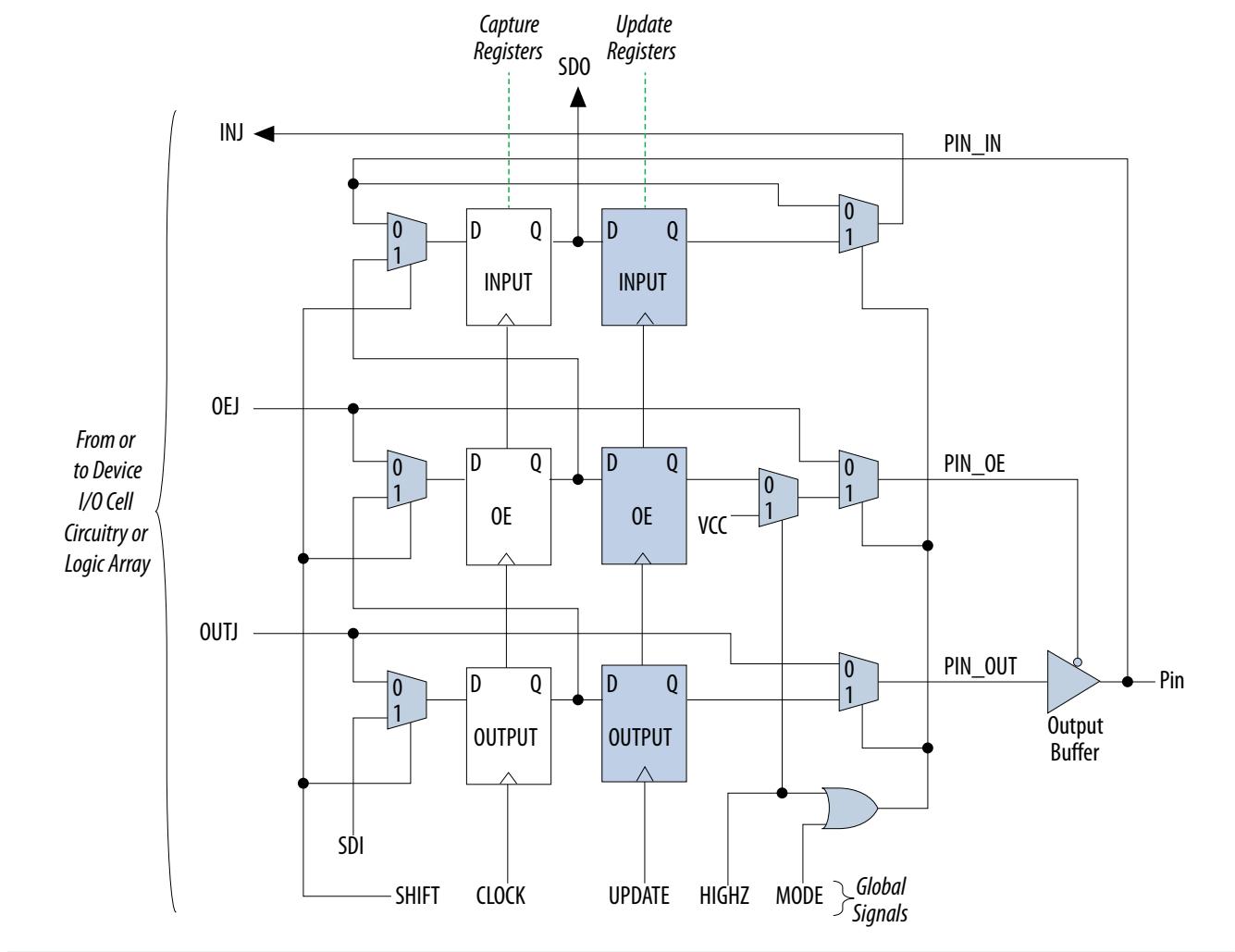
Table 2-2: BSC Capture and Update Register for MAX 10 Devices

Pin Type	Captures			Drives		
	Output Capture Register	OE Capture Register	Input Capture Register	Output Update Register	OE Update Register	Input Update Register
User I/O	OUTJ	OEJ	PIN_IN	PIN_OUT	PIN_OE	INJ

Note: All VCC and GND pin types do not have BSCs.

JTAG IDCODE

The `IDCODE` is unique for each MAX 10 device. Use this code to identify the devices in a JTAG chain.

Table 3-1: IDCODE Information for MAX 10 Devices

Supply Option	Device	Device			
		Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit)
Single-supply	10M02	0000	0011 0001 1000 0001	000 0110 1110	1
	10M04	0000	0011 0001 1000 1010	000 0110 1110	1
	10M08	0000	0011 0001 1000 0010	000 0110 1110	1
	10M16	0000	0011 0001 1000 0011	000 0110 1110	1
	10M25	0000	0011 0001 1000 0100	000 0110 1110	1
	10M40	0000	0011 0001 1000 1101	000 0110 1110	1
	10M50	0000	0011 0001 1000 0101	000 0110 1110	1
Dual-supply	10M02	0000	0011 0001 0000 0001	000 0110 1110	1
	10M04	0000	0011 0001 0000 1010	000 0110 1110	1
	10M08	0000	0011 0001 0000 0010	000 0110 1110	1
	10M16	0000	0011 0001 0000 0011	000 0110 1110	1
	10M25	0000	0011 0001 0000 0100	000 0110 1110	1
	10M40	0000	0011 0001 0000 1101	000 0110 1110	1
	10M50	0000	0011 0001 0000 0101	000 0110 1110	1

JTAG Secure Mode

In JTAG secure mode, the device only allow SAMPLE/PRELOAD, BYPASS, EXTEST, and IDCODE JTAG instructions.

Related Information

MAX 10 FPGA Configuration User Guide
Provides more information about the JTAG Secure Mode.

JTAG Private instruction

Caution: Never invoke the following instruction codes. These instructions can damage the device and render it unusable:

- 10 0100 0000
- 10 0011 0000
- 10 1110 0000
- 10 0011 0001

JTAG Instructions

Instruction Name	Instruction Binary	Description
SAMPLE/PRELOAD	00 0000 0101	• Permits an initial data pattern to be an output at the device pins. • Allows you to capture and examine a snapshot of signals at the device pins if the device is operating in normal mode.
EXTEST [1]	00 0000 1111	• Forces test pattern at the output pins and capture the test results at the input pins. • Allows you to test the external circuitry and board-level interconnects.
BYPASS	11 1111 1111	• Places the 1-bit bypass register between the TDI and TDO pins. • Allows the BST data to pass synchronously through target devices to adjacent devices during normal device operation.
USERCODE	00 0000 0111	• Places the 1-bit bypass register between the TDI and TDO pins. • Allows you to shift the USERCODE register out of the TDO pin serially.
IDCODE	00 0000 0110	• Selects the IDCODE register and places it between the TDI and TDO pins. • Allows you to shift the IDCODE register out of the TDO pin serially.

---

[1] HIGHZ, CLAMP, and EXTEST instructions do not disable weak pull-up resistors or bus hold features.

Instruction Name	Instruction Binary	Description
HIGHZ [1]	00 0000 1011	• Places the 1-bit bypass register between the TDI and TDO pins. The 1-bit bypass register tri-states all the I/O pins. • Allow the BST data to pass synchronously through target devices to adjacent devices if device is operating in normal mode.
CLAMP [1]	00 0000 1010	• Places the 1-bit bypass register between the TDI and TDO pins. The 1-bit bypass register holds I/O pins to a state defined by the data in the boundary-scan register. • Allow the BST data to pass synchronously through target devices to adjacent devices if device is operating in normal mode.
USER0	00 0000 1100	• Allows you to define the scan chain between the TDI and TDO pins in the MAX 10 logic array. • Use this instruction for custom logic and JTAG interfaces.
USER1	00 0000 1110	• Allows you to define the scan chain between the TDI and TDO pins in the MAX 10 logic array. • Use this instruction for custom logic and JTAG interfaces.

A JTAG chain can contain several Altera and non-Altera devices.
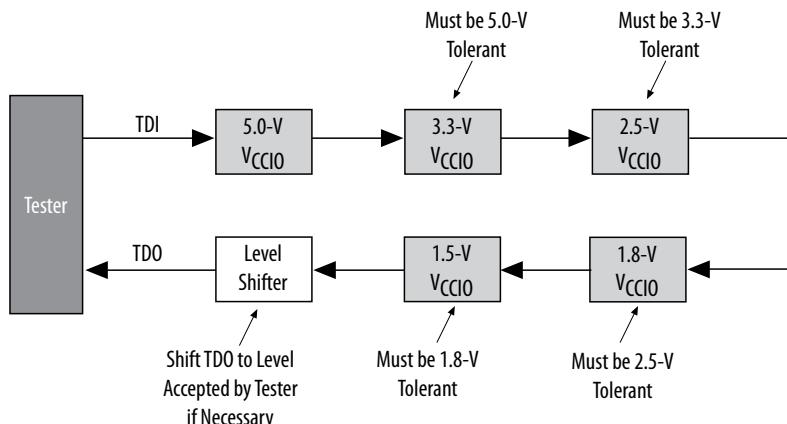
The TDO pin of a device drives out at the voltage level according to the V_{CCIO} of the device. The devices can interface with each other although the devices may have different V_{CCIO} levels.

For example, a device with 3.3-V V_{CCIO} can drive to a device with 5.0-V V_{CCIO} because 3.3 V meets the minimum VIH on transistor-to-transistor logic (TTL)-level input for the 5.0-V V_{CCIO} device.

MAX 10 devices can support 1.5-, 1.8-, 2.5-, or 3.3-V input levels, depending on the V_{CCIO} voltage of I/O Bank 1B.

To interface the TDI and TDO lines of the JTAG pins of devices that have different V_{CCIO} levels, insert a level shifter between the devices. If possible, construct the JTAG chain where device with a higher V_{CCIO} level drives to a device with an equal or lower V_{CCIO} level. In this setup, you only require a level shifter for shifting the TDO level to a level JTAG tester accept.

Figure 4-1: JTAG Chain of Mixed Voltages and Level Shifters

The JTAG BST circuitry in MAX 10 devices is automatically enabled after the power-up.

To ensure that you do not inadvertently enable the JTAG BST circuitry when it is not required, disable the circuitry permanently with pin connections as listed in the following table.

Table 5-1: Pin Connections to Permanently Disable the JTAG BST Circuitry in MAX 10 Devices

JTAG Pins	Connection to Disable
TMS	V_{CCIO} supply of Bank 1B
TCK	GND
TDI	V_{CCIO} supply of Bank 1B
TDO	Leave open

You must enable this circuitry only if you use the BST or ISP features.

Consider the following guidelines when you perform BST with the device:

- If the "10..." pattern does not shift out of the instruction register through the `TDO` pin during the first clock cycle of the `SHIFT_IR` state, the TAP controller did not reach the proper state. To solve this problem, try one of the following procedures:

  - Verify that the TAP controller has reached the `SHIFT_IR` state correctly. To advance the TAP controller to the `SHIFT_IR` state, return TAP controller to the `RESET` state and send the `01100` code to the `TMS` pin.
  - Check the connections to the `VCC`, `GND`, `JTAG`, and dedicated configuration pins on the device.

- Perform a `SAMPLE/PRELOAD` test cycle before the first `EXTEST` test cycle to ensure that known data is present at the device pins when you enter `EXTEST` mode. If the `OEJ` update register contains 0, the data in the `OUTJ` update register is driven out. The state must be known and correct to avoid contention with other devices in the system.

- To perform testing before configuration, hold the `nCONGFIG` pin low.

UG-M10JTAG ✉ Subscribe 💬 Send Feedback

The BSDL—a subset of VHDL—provides a syntax that allows you to describe the features of an IEEE Std. 1149.1 BST-capable device that can be tested. Test software development systems then use the BSDL files for test generation, analysis, failure diagnostics, and in-system programming.

Related Information

IEEE 1149.1 BSDL Files

Provides more information about BSC group definitions.

ISO 9001:2008 Registered

Document Revision History for MAX 10 JTAG Boundary-Scan Testing User Guide

Date	Version	Changes
May 2015	2015.05.04	Added note on about performing the boundary-scan testing in 'Overview'.
September 2014	2014.09.22	Initial release.

ALTERA®

MAX 10 Power Management User Guide

Contents

MAX 10 Power Management Overview

2016.05.02

UG-M10PWR ✉ **Subscribe** 💬 **Send Feedback**

MAX® 10 devices offer the following power supply device options:

- Single-supply device—requires 1 external power supply of 3.0 V or 3.3 V while offering maximum convenience and board simplicity.
- Dual-supply device—requires 2 external power supplies of 1.2 V and 2.5 V while offering the most features, highest performance, and when coupled with high-efficiency Enpirion® PowerSoCs, the lowest power solution.

Related Information

- **MAX 10 Power Management Features and Architecture** on page 2-1
  Provides information about power management features and architecture
- **MAX 10 Power Management User Guide Archives** on page 4-1
  Provides a list of user guides for previous versions of the SmartVID IP core.

**ISO
9001:2008
Registered**

ALTERA®
now part of Intel

MAX 10 power optimization features are as follows:

- Single-supply or dual-supply device options
- Power-on reset (POR) circuitry
- Power management controller scheme
- Hot socketing

Power Supply Device Options

Single-Supply Device

MAX 10 single-supply devices only need either a 3.0- or 3.3-V external power supply. The external power supply serves as an input to the MAX 10 device VCC_ONE and VCCA power pins. This external power supply is then regulated by an internal voltage regulator in the MAX 10 single-supply device to 1.2 V. The 1.2-V voltage level is required by core logic operation.

Figure 2-1: MAX 10 Single-Supply Device

Dual-Supply Device

MAX 10 dual-supply devices require 1.2 V and 2.5 V for the device core logics and periphery operations.

ISO 9001:2008 Registered

ALTERA
now part of Intel

Figure 2-2: MAX 10 Dual-Supply Device

Comparison of the MAX 10 Power Supply Device Options

Table 2-1: Comparison of the MAX 10 Power Supply Device Options

Characteristics	Single-Supply Device	Dual-Supply Device
Voltage regulator count [1]	1	2
Core and I/O performance	Low	High

For MAX 10 single-supply devices, only one power supply is required—3.0 V or 3.3 V to power the core of the FPGA. The same power supply can be used to power the I/O if the same 3.0 V or 3.3 V voltage is required. If different I/O voltage is used, then additional voltage regulators will be needed.

For MAX 10 dual-supply devices, two power supplies are required to supply power to the device core, periphery, phase-locked loop (PLL), and analog-to-digital converters (ADC) blocks—1.2 V and 2.5 V. Depending on the I/O standard voltage requirement, you may use two or more voltage regulators.

As the power rails for the FPGA core are supplied externally in the MAX 10 dual-supply devices, the design can be optimized for power by using high efficiency switching power supplies on the board. The power savings will be equal to the increased efficiency of the regulators used compared to the internal linear regulators of the MAX 10 single-supply devices. If linear regulators are used to power the MAX 10 dual-supply devices, the power consumption of the MAX 10 dual-supply devices will be approximately equal to the MAX 10 single-supply devices.

The device performance of the single-supply device is lower than that of the dual-supply device. For the performance difference in terms of LVDS, pseudo-LVDS, digital signal processing (DSP), and internal memory performance, refer to the MAX 10 FPGA device datasheet.

Related Information
MAX 10 FPGA Device Datasheet
Provides details about the MAX 10 performance difference in terms of LVDS, pseudo-LVDS, DSP, and internal memory performance.

---

[1] This shows the number of power supplies required by the core and periphery of the MAX 10 devices. You may need additional voltage regulators to supply power to the VCCIO if the VCCIO does not have the same voltage level as the core and periphery supply.

Power Supply Design

Designing a power tree for a MAX 10 single- or dual-supply device will vary depending on the static and dynamic power, as well as I/O and other feature utilization, for each specific use case.

The Altera® Enpirion portfolio of power management solutions, combined with comprehensive design tools, enable optimized MAX 10 device power supply design. The Enpirion portfolio includes power management solutions that are compatible with all MAX 10 variants.

The *MAX 10 FPGA Device Family Pin Connection Guidelines* provides a more detailed recommendation about how to group inputs to power a MAX 10 device. The PowerPlay Early Power Estimators (EPE) tool for MAX 10 devices provides input rail power requirements and specific device recommendations based on each specific MAX 10 use case.

Individual input rail voltage and current requirements are summarized on the **Report** tab while input rail groupings and specific power supply recommendations can be found on the **Main** and **Enpirion** tabs, respectively.

Warning: MAX 10 single-supply devices have maximum power consumption of V_{CC_ONE}, as listed in the following table. Running a design that goes beyond the maximum power consumption of V_{CC_ONE} of the MAX 10 single-supply device may cause functional issue on the device. Therefore, ensure that your device does not exceed the maximum power consumption of V_{CC_ONE} when you analyze the power consumption of your design using the PowerPlay EPE spreadsheet.

Table 2-2: Maximum Power Consumption of V_{CC_ONE} for MAX 10 Single-Supply Devices

Device	Maximum Power Consumption (W)
10M02S	0.778
10M04S	1.362
10M08S	1.362
10M16S	2.270
10M25S	2.943
10M40S	5.267
10M50S	5.267

Related Information

- **Enpirion Power Management Solutions**
  Provides more information about Altera's Power Management IC and PowerSoC solutions designed for powering FPGAs.
- **MAX 10 FPGA Device Family Pin Connection Guidelines**
  Provides a more detailed recommendation about how to group inputs in order to power a MAX 10 device.
- **PowerPlay Early Power Estimators (EPE) and Power Analyzer**

Transient Current

You may observe a transient current at the V_{CCIO} power supply when powering up the MAX 10 devices. The transient current of V_{CCIO} applies to all V_{CCIO} voltage levels supported by the MAX 10 device.

Table 2-3: Maximum V$_{CCIO}$ Power Supply Transient Current for MAX 10 Devices

Device	Maximum Power Supply Transient Current (mA)	Duration (s)
10M02	220	
10M04	290	
10M08	300	
10M16	430	25% of the ramp time
10M25	510	
10M40	670	
10M50	680	

Note: The value of the transient current is based on the zero decoupling capacitance on the characterization board. The observed value will be less than the published value after adding the decoupling capacitance on your design board. Altera recommends using a soft start regulator that is able to reduce the transient current when the device is powered.

Power-On Reset Circuitry

The POR circuitry keeps the MAX 10 device in the reset state until the POR monitored power supply outputs are within the recommended operating range of the maximum power supply ramp time, t_{RAMP}.

If the ramp time, t_{RAMP}, is not met, the MAX 10 device I/O pins and programming registers remain tri-stated, during which device configuration could fail.

The MAX 10 device POR circuit monitors the following power rails during power up regardless of the power supply device options:

- V_{CC} or regulated V_{CC_ONE}
- V_{CCIO} of banks 1B and 8 [2]
- V_{CCA}

The POR circuitry also ensures V_{CCIO} level of I/O banks 1B and 8[2] that contain configuration pins reach an acceptable level before configuration is triggered.

[2] V_{CCIO} of banks 1 and 8 for the 10M02 device.
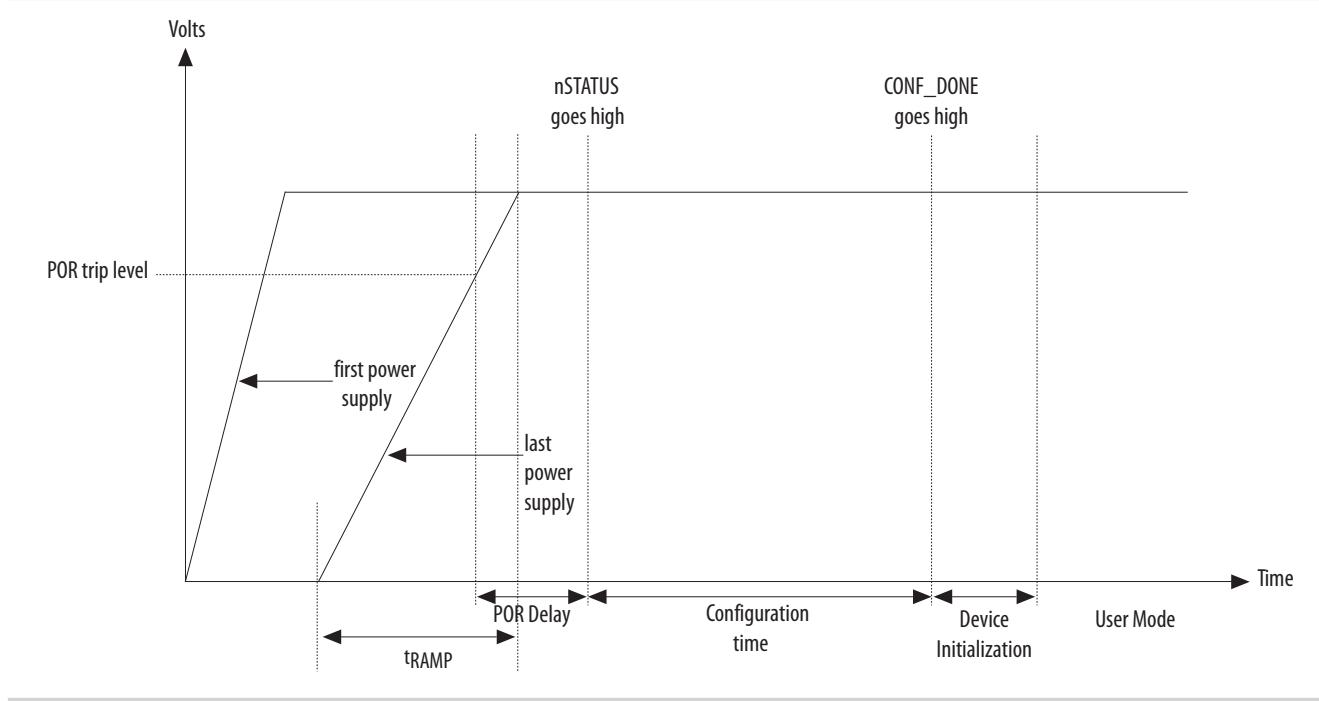
Power Supplies Monitored and Not Monitored by the POR Circuitry

Table 2-4: Power Supplies Monitored and Not Monitored by the POR Circuitry

Power Supply Device Options	Power Supplies Monitored	Power Supplies Not Monitored
Single-supply device	• Regulated V_{CC_ONE} • V_{CCA} • V_{CCIO} [(3)]	—
Dual-supply device	• V_{CC} • V_{CCA} • V_{CCIO}[(3)]	• V_{CCD_PLL} • V_{CCA_ADC} • V_{CCINT}

The MAX 10 POR circuitry uses an individual POR-detecting circuitry to monitor each of the configuration-related power supplies independently. The outputs of all the individual POR detectors gate the main POR circuitry. The main POR circuitry waits for all individual POR circuitries to release the POR signal before allowing the control block to start configuring the device. The main POR is released after the last ramp-up power reaches the POR trip level followed by a POR delay.

Figure 2-3: Monitored Power Supplies Ramp Up
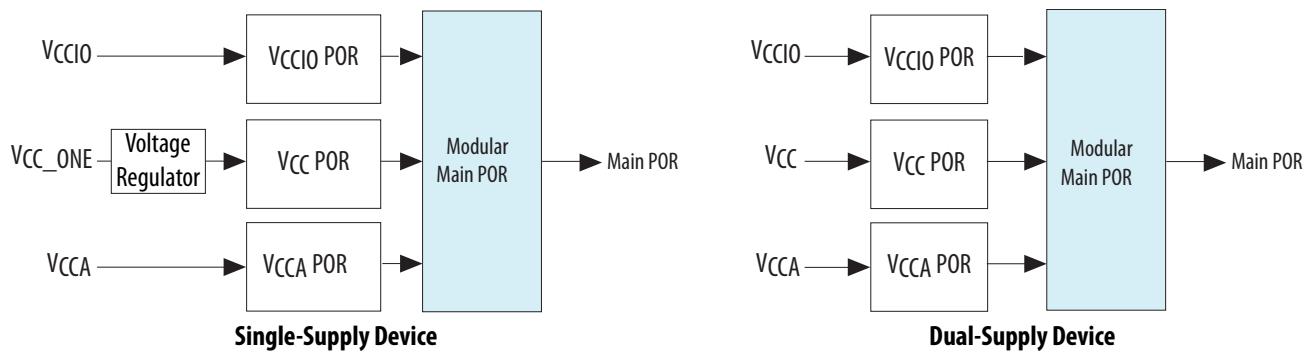
Note: Each individual power supply must reach the recommended operating range within the specified t_{RAMP}.

Note: All VCCIO banks must reach the recommended operating level before configuration completes.

Note: The typical value of POR delay is 2.5 ms for MAX 10 devices.

[(3)] For banks 1B and 8 for all MAX 10 devices and banks 1 and 8 for the 10M02 device.

Figure 2-4: Simplified POR Diagram for MAX 10 Devices

Single-Supply Device

Dual-Supply Device

After the MAX 10 device enters user mode, the POR circuit continues to monitor the V_{CCA} and V_{CC} power supplies. This is to detect a brown-out condition during user mode. If either the V_{CCA} or V_{CC} voltages go below the POR trip point during user mode, the main POR signal is asserted. When the main POR signal is asserted, the device is forced into reset state. V_{CCIO}[3] is monitored by the POR circuitry. In the event of the V_{CCIO}[3] voltage drops during user mode, the POR circuit does not reset the device. However, the POR circuit does monitor the VCCIO voltage drop for up to 9 ms after the last power rail reaches its trip point.
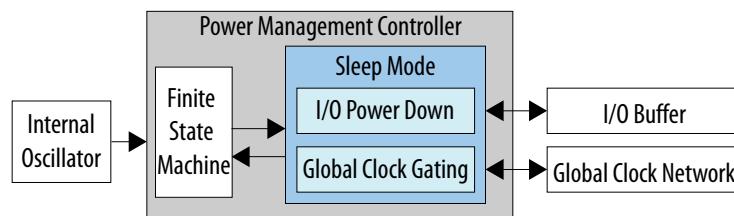
Instant-On Support

In some applications, it is necessary for a device to wake up very quickly to begin operation. The MAX 10 device offers the instant-on feature to support fast wake-up time applications. With the instant-on feature, MAX 10 devices can directly enter configuration mode with a short delay time after the POR trips for the monitored power supplies.

Power Management Controller Scheme

The power management controller scheme allows you to allocate some applications in sleep mode during runtime. This enables you to to turn off portions of the design, thus reducing dynamic power consumption. You can re-enable your application with a fast wake-up time of less than 1 ms.

Power Management Controller Architecture

Figure 2-5: Power Management Controller Architecture

The MAX 10 device contains hardware features that enable I/O power down and global clock (GCLK) gating to manage low-power state during sleep mode. You can power down the I/O buffer dynamically

when your application is in idle or sleep mode. One example is the digital single lens reflex DSLR camera application where the LVDS I/O needs to be powered down during the idle condition. Without touching any buttons, the screen turns off while the camera is still powered on.

Altera provides a soft power management controller as reference design utilizing low-power features implemented in the MAX 10 devices. You can modify the reference design based on your application. The soft power management controller includes a simple finite state machine (FSM) to manage the low-power state mode by powering down the I/O buffer and GCLK gating during sleep mode.

All MAX 10 devices contain hardware features for clock gating. The 10M16, 10M25, 10M40, and 10M50 devices contain hardware features for I/O power down. With hardware features, you can manage the low-power state during sleep mode by using the soft power management controller that you define.

You can implement the power management controller in FPGA core fabric with a minimum of one I/O port reserved for sleep mode enter and exit signals.

Internal Oscillator

The internal oscillator clocks the power management controller operation. The internal oscillator is routed from flash to the core. The internal oscillator enables the power management controller to detect the wake-up event and the sleep mode event. In order to enable the internal oscillator clock when the power management controller is enabled, you have to set oscena to 1. For the clock frequency of the internal oscillator, refer to the *MAX 10 FPGA Device Datasheet*.

Related Information
MAX 10 FPGA Device Datasheet
Provides details about the MAX 10 ramp time requirements, internal oscillator clock frequency, and hot-socketing specifications.

I/O Buffer Power Down

The MAX 10 device has a dynamic power-down feature on some of the I/O buffers that have high-static power consumption. The dynamic power-down feature is only applicable for the I/O buffers that have been programmed for the I/O standards in the following table.

Table 2-5: I/O Buffer Power Down

I/O Buffer	I/O Standards	Control Port	Control Signal Capability
Input	SSTL, HSTL, HSUL, and LVDS	nsleep	1 per I/O bank[4]
Output	All I/O standards	oe	1 per I/O buffer

During power-up and configuration modes, the soft power management controller is not yet configured and the control signals are forced to 1 (inactive). After configuration mode, when the power management controller is activated, the power management controller will default the control signals to 1. When control signals are 0, the power management controller powers down or tri-states the I/O buffers. Subsequently the I/O is put into the sleep mode.

---

[4] I/O banks 1A and 1B share one control signal.

The MAX 10 device I/O buffers need to maintain the previous states during the sleep mode operation. The previous states in your core logics remain upon exiting the sleep mode.
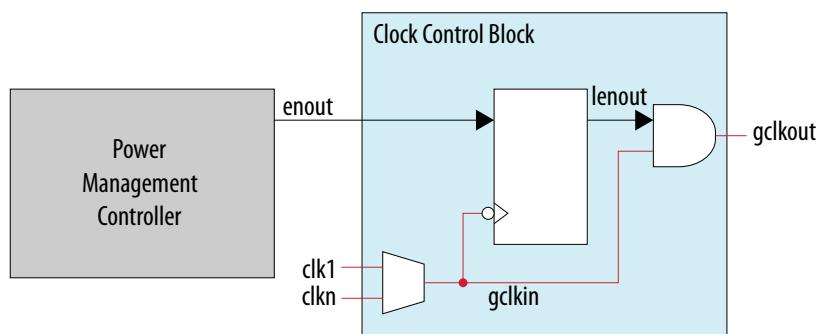
Global Clock Gating

The dynamic power-down feature is available in GCLK networks only. You can use the power management controller for the dynamic power-down of a GCLK network by controlling the active high `enout` signal. The GCLK networks serve as low-skew clock sources for functional blocks such as logic array blocks (LABs), DSP, embedded memory, and PLLs.

When a GCLK network is gated, all the logics fed by the GCLK network are in off-state. This reduces the overall power consumption of the device. The dynamic power-down feature allows core logics to control the following power-up and power-down conditions of the GCLK networks:

- Power down synchronously or asynchronously
- Power up asynchronously

Figure 2-6: GCLK Gating

Hot Socketing

The MAX 10 device offers hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove the MAX 10 device on a board in a system during system operation. This does not affect the running system bus or the board that is inserted into the system.

The hot-socketing feature removes some encountered difficulties when using the MAX 10 device on a PCB that contains a mixture of devices with different voltage levels.

With the MAX 10 device hot-socketing feature, you no longer need to ensure a proper power-up sequence for each device on the board. MAX 10 device hot-socketing feature provides:

- Board or device insertion and removal without external components or board manipulation
- Support for any power-up sequence
- Non-intrusive I/O buffers to system buses during hot insertion

Hot-Socketing Specifications

The MAX 10 device is a hot-socketing compliant device that does not need any external components or special design requirements. Hot-socketing support in the MAX 10 device has the following advantages:

- You can drive the devices before power up without damaging the device.
- I/O pins remain tri-stated during power up. The device does not drive out before or during power up, therefore not affecting other buses in operation.

Drive MAX 10 Devices Before Power Up

Before or during power up or power down, you can drive signals into I/O pins, dedicated input pins, and dedicated clock pins without damaging the MAX 10 devices.

The MAX 10 device supports any power-up or power-down sequence to simplify system-level design.

I/O Pins Remain Tri-stated During Power up

The output buffers of the MAX 10 device are turned off during system power up or power down. The MAX 10 device family does not drive out until the device is configured and working in recommended operating conditions. The I/O pins are tri-stated during power up or power down.

A possible concern for semiconductor devices in general regarding hot-socketing is the potential for latch up. Latch up can occur when electrical subsystems are hot-socketed into an active system. During hot-socketing, the signal pins may be connected and driven by the active system. This occurs before the power supply can provide current to the V_{CC} of the device and ground planes. This condition can lead to latch up and cause a low-impedance path from V_{CC} to ground in the device. As a result, the device extends a large amount of current, possibly causing electrical damage.

The design of the I/O buffers and hot-socketing circuitry ensures that the MAX 10 device family is immune to latch up during hot-socketing.
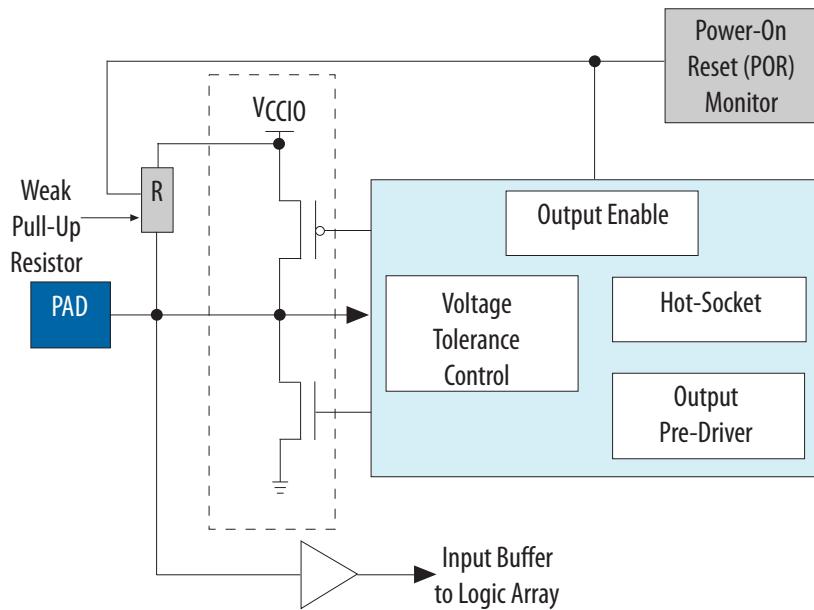
Related Information
MAX 10 FPGA Device Datasheet
Provides details about the MAX 10 ramp time requirements, internal oscillator clock frequency, and hot-socketing specifications.

Hot-Socketing Feature Implementation

The hot-socketing feature tri-states the output buffer during the power-up (V_{CCIO} or V_{CC} power supplies) or power-down event. The hot-socketing circuitry generates an internal HOTSCKT signal when V_{CCIO} or V_{CC} is below the threshold voltage during power up or power down. The HOTSCKT signal cuts off the output buffer to ensure that no DC current leaks through the pin (except for weak pull-up leaking). Each I/O pin has the circuitry shown in the following figure. The hot-socketing circuit does not include CONF_DONE and nSTATUS pins to ensure that these pins are able to operate during configuration. Thus, it is an expected behavior for these pins to drive out during power-up and power-down sequences.

Figure 2-7: Hot-Socketing Circuitry for MAX 10 Devices

The POR circuit monitors the voltage level of power supplies and keeps the I/O pins tri-stated during power up. The weak pull-up resistor in MAX 10 device I/O elements (IOE) keeps the I/O pins from floating. The voltage tolerance control circuit protects the I/O pins from being driven before V_{CCIO} and V_{CC} supplies are powered up. This prevents the I/O pins from driving out when the device is not in user mode.

Altera uses GND as reference for hot-socketing operation and I/O buffer designs. To ensure proper operation, Altera recommends connecting the GND between boards before connecting the power supplies. This prevents the GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND can cause an out-of-specification I/O voltage or current condition with the Altera device.

This reference design utilizes the low-power feature supported in MAX 10 devices. The following figure shows the related block diagrams in the power management controller reference design.

Figure 3-1: Power Management Controller Block Diagram
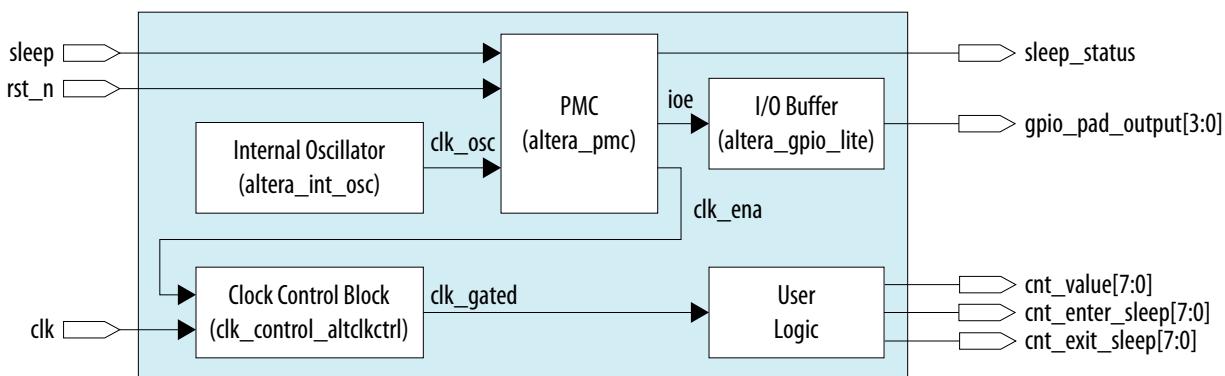
Table 3-1: Input and Output Ports of the Power Management Controller Reference Design

Port Name	Input/Output	Description
sleep	Input	Sleep control.
rst_n	Input	Active low reset signal.
clk	Input	Clock signal.
sleep_status	Output	Sleep status of the system. This signal is asserted high when the system is entering the sleep mode condition. This signal is de-asserted when the system exits the sleep mode condition completely.
gpio_pad_output[3:0]	Output	General-purpose I/O (GPIO) output ports.
cnt_value[7:0]	Output	Free-running counter value in user logic.
cnt_enter_sleep[7:0]	Output	Counter value when the system is entering sleep mode condition.

Port Name	Input/Output	Description
`cnt_exit_sleep[7:0]`	Output	Counter value when the system is exiting sleep mode condition.

The power management controller design is a FSM showing the state of powering down and powering up global clocks (GCLKs) and I/O buffers. The internal oscillator, clock control block, and I/O buffer are intellectual property (IP) that are supported by the Quartus® Prime software and you can instantiate the IPs from the IP catalog. The user logic can be any logical circuitry that are implemented using logic element (LE) and an embedded component such as DSP and internal memory in your design. In this reference design, the user logic used is a free-running 8-bit counter. The `cnt_enter_sleep` and `cnt_exit_sleep` ports are used to ensure user logic can enter and exit sleep mode without data corruption. It is expected for that `cnt_enter_sleep[7:0]` and `cnt_exit_sleep[7:0]` are at the same value after the user logic enter and exit sleep mode. `gpio_pad_output` ports demonstrate tri-stated state of the GPIO when the system is in sleep mode.

Related Information

Power Management Controller Reference Design

Clock Control Block

The clock control IP core (`clk_control_altclkctrl`) is an IP provided in the Quartus Prime software. This IP is used to control the clock system in the device. The GCLKs that drive through the device can be dynamically powered down by controlling the active high `ena` signal. The `ena` port is an input to the clock control IP block. When this IP is instantiated, select the `ena` port to enable the controls of GCLKs.

Related Information

Clock Control Block (ALTCLKCTRL) Megafunction User Guide

I/O Buffer

Altera GPIO Lite IP core (`altera_gpio_lite`) is implemented as an input, output, or bidirectional I/O buffer. You can control the power down of these I/O buffers by enabling the `nsleep` port of the input buffer and the `oe` port of the output buffer. The `oe` and `nsleep` ports are pulled low by the power management controller design to power down the I/O buffers during sleep mode. Altera recommends using a separate Altera GPIO Lite IP when some of the I/O buffer is not required to be powered down.

Related Information

Altera GPIO Lite IP Core References

Internal Oscillator

Internal Oscillator IP core (`altera_in_osc`) is a free-running oscillator once you enable it. This oscillator runs throughout the entire power management controller design.
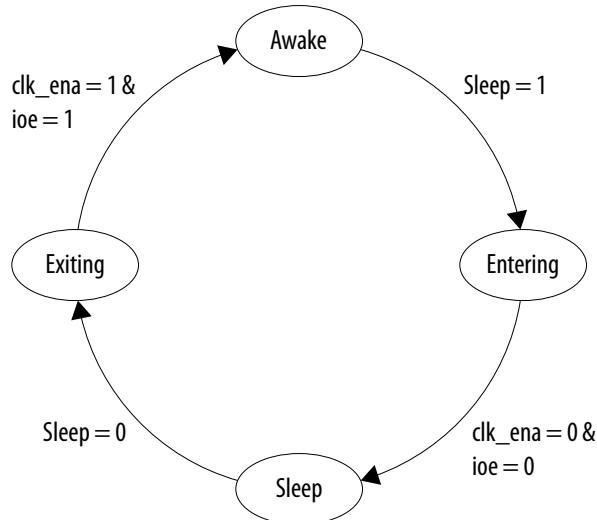
Related Information

Internal Oscillator IP Core

Power Management Controller

The power management controller implements a simple FSM to control the power-up and power-down sequences of the GCLK networks and I/O buffer.

Figure 3-2: FSM of the Power Management Controller

Entering State

When the power management controller detects a sleep event, the FSM transitions to the Entering state and performs power-down operation on I/O buffers and GCLK networks. A sleep event is detected when the `sleep` signal is asserted. A sleep event could be triggered by an internal or external request.

Sleep State

After the power-down operation on I/O buffers and GCLK networks, the FSM transitions to the Sleep state and waits for the wake-up event. This state is the sleep mode state.

Exiting State

When the power management controller detects a wake-up event, the FSM transitions to the Exiting state and performs power-up operation on I/O buffers and GCLK networks. A wake-up event is detected when the `sleep` signal is de-asserted. A wake-up event could be triggered by an internal or external request such as interruption or time-out on some counters.

Awake State
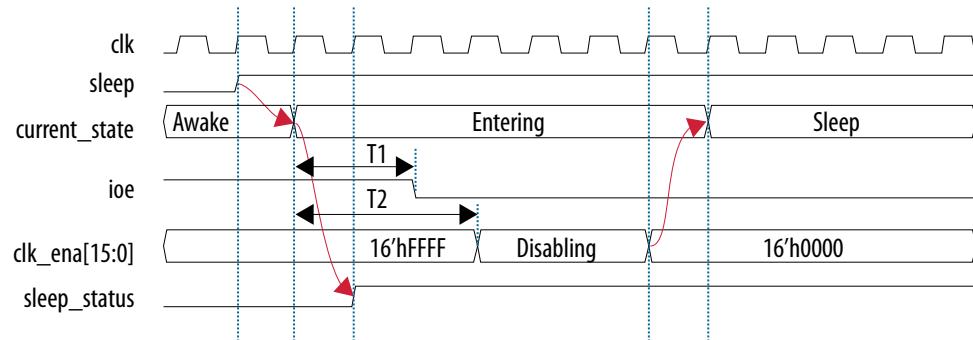
After the power-up operation on I/O buffers and GCLK networks, the FSM transitions to the Awake state.

This process repeats when a sleep event is initiated again.

Entering or Exiting Sleep Mode

During power-up and configuration modes, the sleep signal must be low. When the sleep signal is asserted, the device immediately enters sleep mode. Upon entering sleep mode, the functionality of the device such as GCLK networks and I/O buffers are dynamically powered down—to minimize dynamic power dissipation. All configuration data is retained when the device is in the sleep mode.

## Entering Sleep Mode

Figure 3-3: Entering Sleep Mode Timing Diagram
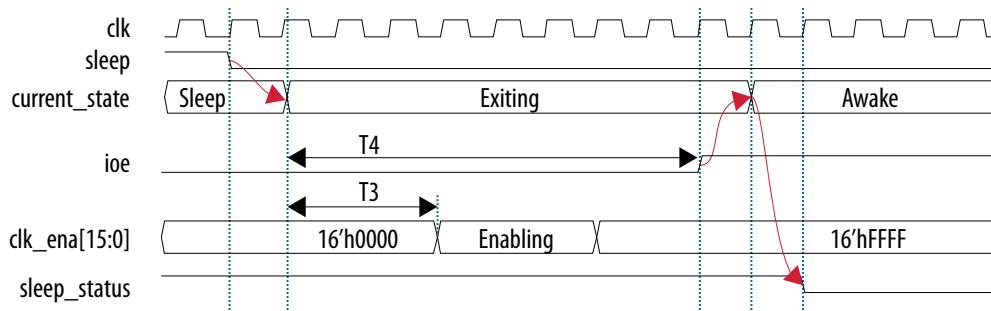
The following sequence occurs when the device enters sleep mode:

1. An internal or external request drives the sleep signal high, forcing the device to go into sleep mode.
2. After a delay of T1, the power management controller powers down all the I/O buffers by de-asserting ioe signal that connects to oe and nsleep ports of the I/O buffers.
3. After a delay of T2, the power management controller turns off all GCLK networks by disabling clk_ena[15:0] signal from LSB to MSB. After three clock cycles, the clk_ena[15:0] signal is fully disabled and transits into the sleep state.
4. The power management controller remains in sleep state until the sleep signal is de-asserted.
5. User logic will latch the running counter value before entering the sleep state and output to cnt_sleep_enter port. The running counter is then frozen.
6. gpio_pad_output (GPIO) is tri-stated when ioe is de-asserted.

Exiting Sleep Mode

Figure 3-4: Exiting Sleep Mode Timing Diagram

The following sequence occurs when the device exits sleep mode:

1. An internal or external request drives the `sleep` signal low, forcing the device to exit sleep mode.
2. After a delay of T3, the power management controller turns on all GCLK networks by enabling `clk_ena[15:0]` signal from LSB to MSB. After three clock cycles, the `clk_ena[15:0]` signal is fully enabled and all GCLK networks are turned on.
3. After a delay of T4, the power management controller powers up all the I/O buffers by asserting the `ioe` signal.
4. The power management controller remains in awake state until the `sleep` signal is asserted.
5. User logic will latch the running counter value before the awake state and output to `cnt_sleep_exit` port. The running counter is then release from freeze.
6. `gpio_pad_output` (GPIO) is driving its output value when `ioe` is asserted.

Timing Parameters

The following table lists the definition and minimum value of the T1, T2, T3, and T4 parameters in the entering sleep mode timing diagram and exiting sleep mode timing diagram, respectively.

Table 3-2: T1, T2, T3, and T4 Parameters Minimum Value and Definition

Parameter	Width (bits)	Minimum Value (Clock Cycle)	Description
T1	6	1	`ioe` disable timing.
T2	6	11	`clk_ena` disable timing.
T3	6	1	`clk_ena` enable timing.
T4	6	40	`ioe` enable timing.

T1, T2, T3, and T4 can be increased based on your system requirement.

Hardware Implementation and Current Measurement

This design is implemented using the 10M50DAF484C6 device. You can implement this design using any MAX 10 device. This design runs on the MAX 10 Development Kit Board to show current and power relative between user mode and sleep mode.

The resource utilization of this design is as follows:

- 42,000 LEs (84% of total LEs)—gray counter top module utilizes most of the LEs in the device
- 33 I/O pins (9% of total pins)—covering 3 input pins and 30 output pins
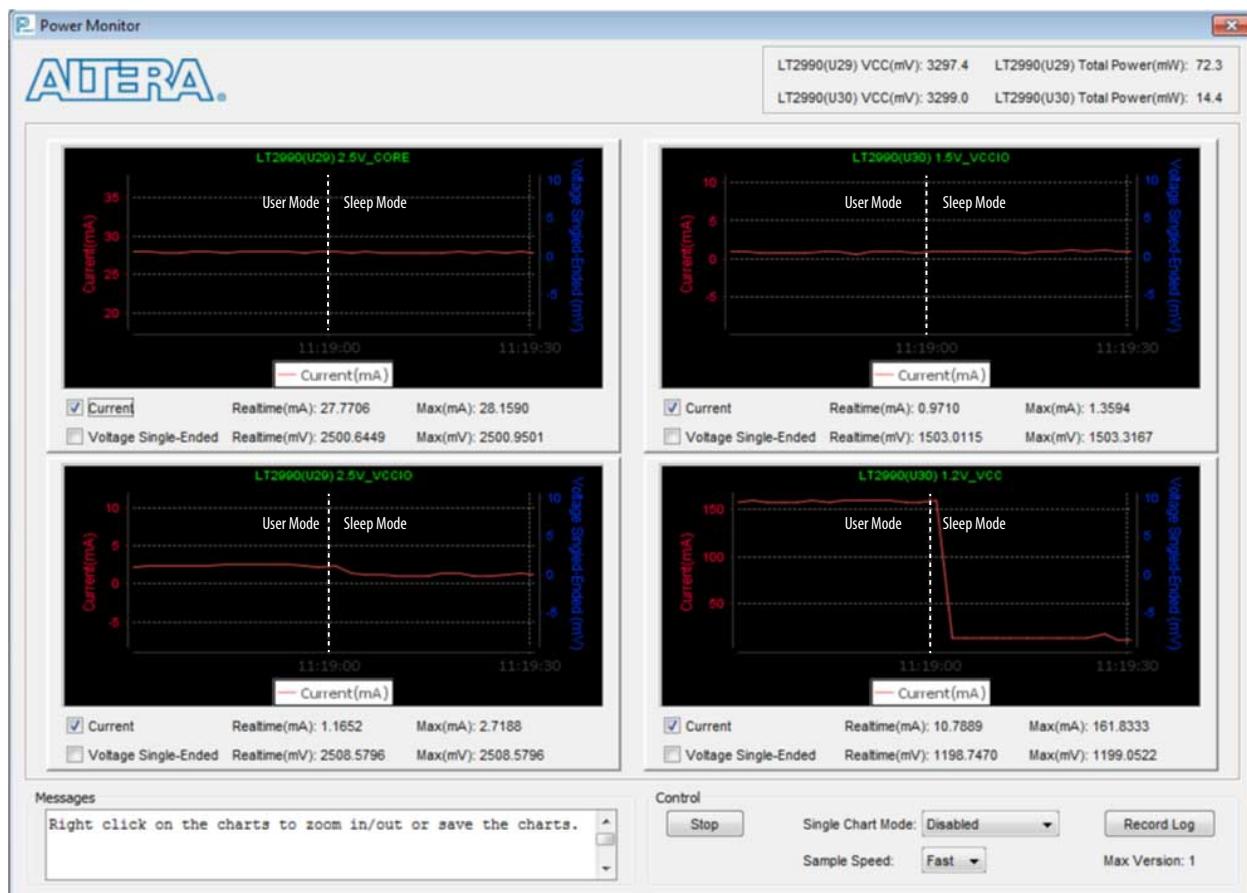
The current in this design is measured using a current monitor component (the Linear Technologies LTC 2990). The measured current is further processed by a pre-programmed design in a MAX II device. The

measured current is shown on Altera power monitor GUI when the PowerMonitor.exe is launched. You will see a current monitor for each of the main supplies to the MAX 10 device as follows:

- 2.5V_CORE[5]
- 2.5V_VCCIO
- 1.5V_VCCIO
- 1.2V_VCC

For design demonstration purpose, the push button is used for sleep control and the LEDs are used for sleep status. Thus, these signals have been inverted on the pin level. To enter sleep mode, press and hold the push button USER_PB0. To release the design to user mode, release the push button USER_PB0. LED0 indicates the sleep status of the device. LED0 is turned on when the device enters sleep mode and is turned off when the device is in user mode. During sleep mode, gpio_pad_output ports connecting to LED1–LED4 are tri-stated and then turned off.

Figure 3-5: Current Monitor for Each Supply

In sleep mode, all GCLK networks are gated and all output buffers are disabled.

---

[5] This is 2.5V_VCCA.

Table 3-3: Comparison of Current and Power Consumption

Current and Power	User Mode	Sleep Mode
1.2V_ICC (mA)	160	11
2.5V_ICCA (mA)	28	28
1.5V_ICCIO (mA)	1.3	1.0
2.5V_ICCIO (mA)	2.7	1.2
Total power (mW)	270	88

The results show an approximate 93% reduction in the core current (1.2V_ICC) consumption and an approximate 56% reduction in I/O current (2.5V_ICCIO) consumption in sleep mode relative to user mode. The total power consumption reduction in this design in sleep mode is about 68%.

2016.05.02

UG-M10PWR ✉ **Subscribe** 💬 **Send Feedback**

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	**MAX 10 Power Management User Guide**
15.0	**MAX 10 Power Management User Guide**
14.1	**MAX 10 Power Management User Guide**

ALTERA
now part of Intel

Additional Information for MAX 10 Power Management User Guide **B**

Document Revision History for MAX 10 Power Management User Guide

Date	Version	Changes
May 2016	2016.05.02	• Updated the I/O Pins Remain Tri-stated During Power Up section. • Updated the Power Supplies Monitored and Not Monitored by the POR Circuitry section. • Updated the information for the single-supply device in the Instant-On Power-Up Sequence Requirement table.
November 2015	2015.11.02	• Added the Transient Current section. • Changed instances of Quartus II to Quartus Prime.
February 2015	2015.02.09	Added the MAX 10 Power Management Controller Reference Design.
December 2014	2014.12.15	• Updated the MAX 10 Power Management Overview section. • Updated the Dual-Supply Device section to update details on power consumption for dual-supply devices. • Updated the Power Supply Design section to include the maximum power consumption for each MAX 10 single-supply device. • Updated the Power Management Controller Scheme section to include updates on sleep mode.
September 2014	2014.09.22	Initial release.

ISO 9001:2008 Registered

ALTERA now part of Intel