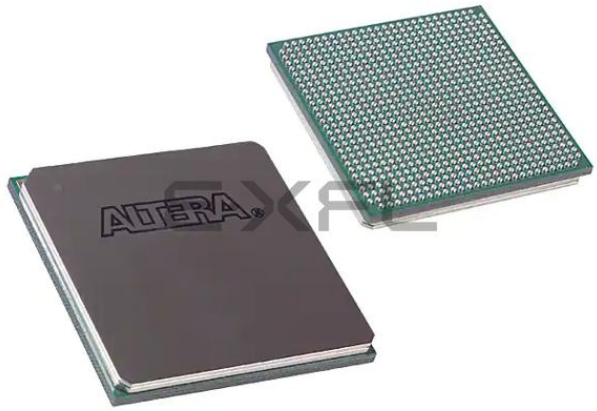


Welcome to E-XFL.COM

Understanding Embedded - FPGAs (Field Programmable Gate Array)

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

Applications of Embedded - FPGAs

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

Details	
Product Status	Active
Number of LABs/CLBs	1803
Number of Logic Elements/Cells	28848
Total RAM Bits	608256
Number of I/O	532
Number of Gates	-
Voltage - Supply	0.97V ~ 1.03V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	780-BGA
Supplier Device Package	780-FBGA (29x29)
Purchase URL	https://www.e-xfl.com/product-detail/intel/ep4ce30f29c8ln

This chapter provides additional information about the document and Altera.

About this Handbook

This handbook provides comprehensive information about the Altera® Cyclone® IV family of devices.

How to Contact Altera

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

Contact (1)	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Nontechnical support (general)	Email	nacomp@altera.com
(software licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in a document and titles of Quartus II Help topics. For example, "Typographic Conventions."

2. Logic Elements and Logic Array Blocks in Cyclone IV Devices

CYIV-51002-1.0

This chapter contains feature definitions for logic elements (LEs) and logic array blocks (LABs). Details are provided on how LEs work, how LABs contain groups of LEs, and how LABs interface with the other blocks in Cyclone® IV devices.

Logic Elements

Logic elements (LEs) are the smallest units of logic in the Cyclone IV device 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

Figure 3–3 and Figure 3–4 show the address clock enable waveform during read and write cycles, respectively.

Figure 3–3. Cyclone IV Devices Address Clock Enable During Read Cycle Waveform

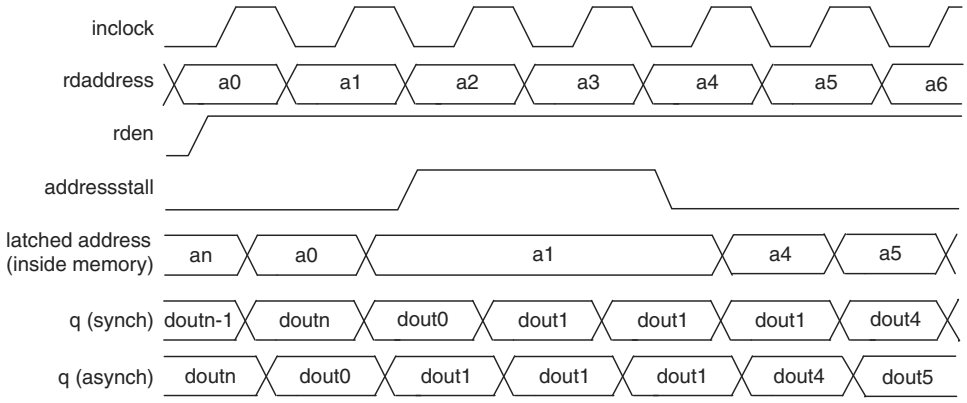
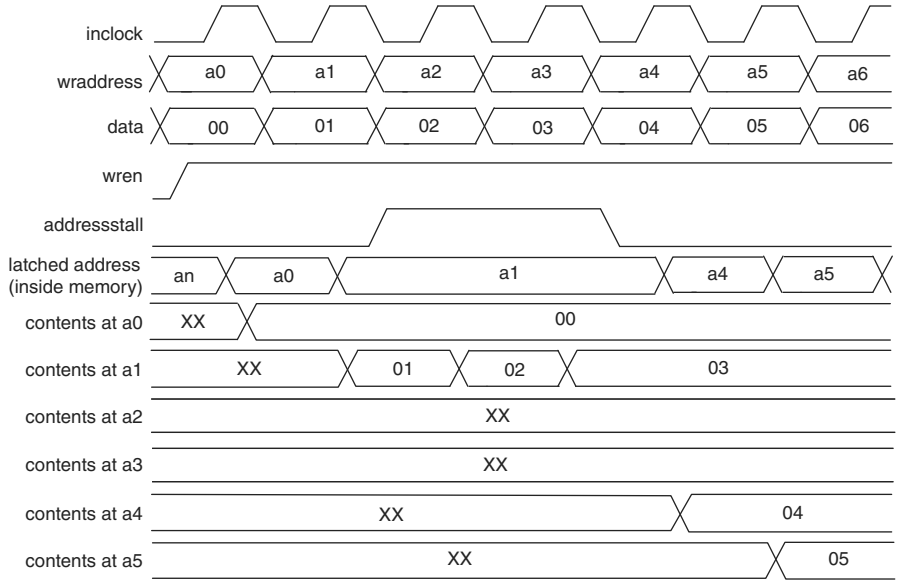


Figure 3–4. Cyclone IV Devices Address Clock Enable During Write Cycle Waveform



Mixed-Width Support

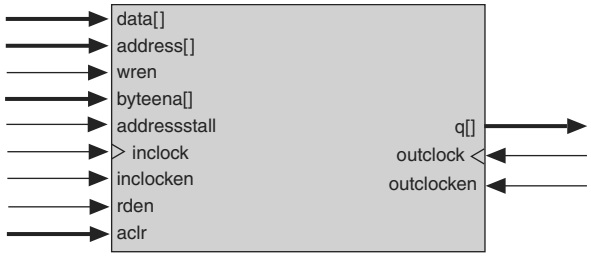
M9K memory blocks support mixed data widths. When using simple dual-port, true dual-port, or FIFO modes, mixed width support allows you to read and write different data widths to an M9K memory block. For more information about the different widths supported per memory mode, refer to "Memory Modes" on page 3–7.

☞ 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.

Single-Port Mode

Single-port mode supports non-simultaneous read and write operations from a single address. Figure 3–6 shows the single-port memory configuration for Cyclone IV devices M9K memory blocks.

Figure 3–6. Single-Port Memory (1), (2)



Notes to Figure 3–6:

(1) You can implement two single-port memory blocks in a single M9K block.
(2) For more information, refer to "Packed Mode Support" on page 3–4.

During a write operation, the behavior of the RAM outputs is configurable. If you activate `rden` during a write operation, the RAM outputs show either the new data being written or the old data at that address. If you perform a write operation with `rden` deactivated, the RAM outputs retain the values they held during the most recent active `rden` signal.

To choose the desired behavior, set the **Read-During-Write** option to either **New Data** or **Old Data** in the RAM MegaWizard Plug-In Manager in the Quartus II software. For more information about read-during-write mode, refer to "Read-During-Write Operations" on page 3–15.

The port width configurations for M9K blocks in single-port mode are as follow:

- 8192 × 1
- 4096 × 2
- 2048 × 4
- 1024 × 8
- 1024 × 9
- 512 × 16
- 512 × 18
- 256 × 32
- 256 × 36

Table 5–1. GCLK Network Connections for EP4CGX15, EP4CGX22, and EP4CGX30 (1), (2) (Part 2 of 2)

GCLK Network Clock Sources	GCLK Networks																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
PLL_3_C1	—	—	—	—	—	—	✓	—	—	✓	—	—	—	—	—	—	✓	—	—	✓
PLL_3_C2	—	—	—	—	—	✓	—	✓	—	—	—	—	—	—	—	✓	—	✓	—	—
PLL_3_C3	—	—	—	—	—	—	✓	—	✓	—	—	—	—	—	—	—	✓	—	✓	—
PLL_3_C4	—	—	—	—	—	—	—	✓	—	✓	—	—	—	—	—	—	—	✓	—	✓
PLL_4_C0 (3)	—	—	—	—	—	✓	—	—	✓	—	✓	—	—	✓	—	—	—	—	—	—
PLL_4_C1 (3)	—	—	—	—	—	—	✓	—	—	✓	—	✓	—	—	✓	—	—	—	—	—
PLL_4_C2 (3)	—	—	—	—	—	✓	—	✓	—	—	✓	—	✓	—	—	—	—	—	—	—
PLL_4_C3 (3)	—	—	—	—	—	—	✓	—	✓	—	—	✓	—	✓	—	—	—	—	—	—
PLL_4_C4 (3)	—	—	—	—	—	—	—	✓	—	✓	—	—	✓	—	✓	—	—	—	—	—
DPCLK2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—
DPCLK3 (4)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—
DPCLK4 (4)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—
DPCLK5	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓
DPCLK6 (4)	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—
DPCLK7	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—	—
DPCLK8	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—
DPCLK9 (4)	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—
DPCLK10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—
DPCLK11 (4)	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—
DPCLK12 (4)	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—
DPCLK13	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—

Notes to Table 5–1:

(1) EP4CGX30 information in this table refers to all EP4CGX30 packages except F484 package.

(2) PLL_1 and PLL_2 are multipurpose PLLs while PLL_3 and PLL_4 are general purpose PLLs.

(3) PLL_4 is only available in EP4CGX22 and EP4CGX30 devices in F324 package.

(4) This pin applies to EP4CGX22 and EP4CGX30 devices.

Table 5–2. GCLK Network Connections for EP4CGX30, EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150 Devices (1), (2) (Part 4 of 4)

GCLK Network Clock Sources	GCLK Networks																												
	0	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
DPCLK17	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	—	—	—	—	—	—	—	—	—

Notes to Table 5–2:

(1) EP4CGX30 information in this table refers to only EP4CGX30 device in F484 package.

(2) PLL_1, PLL_2, PLL_3, and PLL_4 are general purpose PLLs while PLL_5, PLL_6, PLL_7, and PLL_8 are multipurpose PLLs.

(3) PLL_7 and PLL_8 are not available in EP4CXGX30, EP4CGX50, EP4CGX75, EP4CGX110, and EP4CGX150 devices in F484 package.

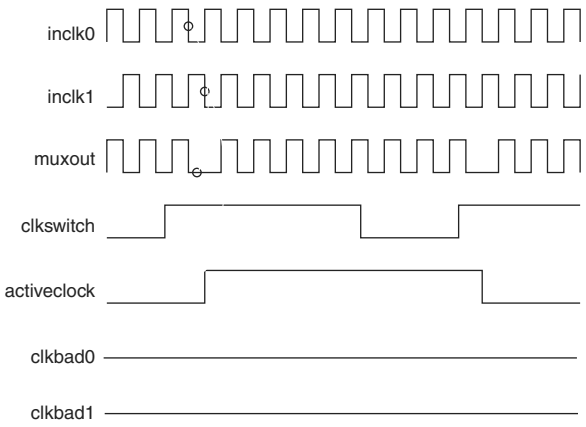
Table 5–3. GCLK Network Connections for Cyclone IV E Devices (1) (Part 1 of 3)

GCLK Network Clock Sources	GCLK Networks																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
CLK1	—	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
CLK2/DIFFCLK_1p	—	✓	—	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
CLK3/DIFFCLK_1n	✓	—	—	✓	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
CLK4/DIFFCLK_2p	—	—	—	—	—	✓	—	✓	—	✓	—	—	—	—	—	—	—	—	—	—
CLK5/DIFFCLK_2n	—	—	—	—	—	—	✓	✓	—	—	—	—	—	—	—	—	—	—	—	—
CLK6/DIFFCLK_3p	—	—	—	—	—	—	✓	—	✓	✓	—	—	—	—	—	—	—	—	—	—
CLK7/DIFFCLK_3n	—	—	—	—	—	✓	—	—	✓	—	—	—	—	—	—	—	—	—	—	—
CLK8/DIFFCLK_5n (2)	—	—	—	—	—	—	—	—	—	—	✓	—	✓	—	✓	—	—	—	—	—
CLK9/DIFFCLK_5p (2)	—	—	—	—	—	—	—	—	—	—	—	✓	✓	—	—	—	—	—	—	—
CLK10/DIFFCLK_4n (2)	—	—	—	—	—	—	—	—	—	—	—	✓	—	✓	✓	—	—	—	—	—
CLK11/DIFFCLK_4p (2)	—	—	—	—	—	—	—	—	—	—	✓	—	—	✓	—	—	—	—	—	—
CLK12/DIFFCLK_7n (2)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	✓	—	✓
CLK13/DIFFCLK_7p (2)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	✓	—	—
CLK14/DIFFCLK_6n (2)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	✓	—	✓	✓

In this mode, the activeclock signal mirrors the clkswitch signal. As both blocks are still functional during the manual switch, neither clkbad signals go 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. The clkswitch signal and the automatic switch only works depending on the availability of the clock that is switched to. If the clock is unavailable, the state machine waits until the clock is available.

When CLKSWITCH = 1, it overrides the automatic switch-over function. As long as clkswitch signal is high, further switch-over action is blocked.

Figure 5–19. Clock Switchover Using the clkswitch Control *(1)*



Note to Figure 5–19:

(1) Both inclk0 and inclk1 must be running when the clkswitch signal goes high to start a manual clock switchover event.

Manual Clock Switchover

PLLs of Cyclone IV devices support manual switchover, in which the clkswitch signal controls whether inclk0 or inclk1 is the input clock to the PLL. The characteristics of a manual switchover are similar to the manual override feature in an automatic clock switchover, in which the switchover circuit is edge-sensitive. When the clkswitch signal goes high, the switchover sequence starts. The falling edge of the clkswitch signal does not cause the circuit to switch back to the previous input clock.

For more information about PLL software support in the Quartus II software, refer to the *ALTPLL Megafunction User Guide*.

Guidelines

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

- Clock loss detection and automatic clock switchover require the inclk0 and inclk1 frequencies be within 20% of each other. Failing to meet this requirement causes the clkbad0 and clkbad1 signals to function improperly.

Table 5–12. Dynamic Phase Shifting Control Signals (Part 2 of 2)

Signal Name	Description	Source	Destination
scanclk	Free running clock from core used in combination with phasestep to enable or disable dynamic phase shifting. Shared with scanclk for dynamic reconfiguration.	GCLK or I/O pins	PLL reconfiguration circuit
phasedone	When asserted, it indicates to core logic that the phase adjustment is complete and PLL is ready to act on a possible second adjustment pulse. Asserts based on internal PLL timing. De-asserts on the rising edge of scanclk.	PLL reconfiguration circuit	Logic array or I/O pins

Table 5–13 lists the PLL counter selection based on the corresponding PHASECOUNTERSELECT setting.

Table 5–13. Phase Counter Select Mapping

phasecounterselect			Selects
[2]	[1]	[0]	
0	0	0	All Output Counters
0	0	1	M Counter
0	1	0	C0 Counter
0	1	1	C1 Counter
1	0	0	C2 Counter
1	0	1	C3 Counter
1	1	0	C4 Counter

To perform one dynamic phase-shift, follow these 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 180°, in other words, a phase shift of 5 ns.

Table 6–7. Differential I/O Standards Supported in Cyclone IV GX I/O Banks

Differential I/O Standards	I/O Bank Location	External Resistor Network at Transmitter	Transmitter (TX)	Receiver (RX)
LVDS	5,6	Not Required	✓	✓
	3,4,5,6,7,8	Three Resistors		
RSDS	5,6	Not Required	✓	—
	3,4,7,8	Three Resistors		
	3,4,5,6,7,8	Single Resistor		
mini-LVDS	5,6	Not Required	✓	—
	3,4,5,6,7,8	Three Resistors		
PPDS	5,6	Not Required	✓	—
	3,4,5,6,7,8	Three Resistors		
BLVDS (1)	3,4,5,6,7,8	Single Resistor	✓	✓
LVPECL (2)	3,4,5,6,7,8	—	—	✓
Differential SSTL-2 (3)	3,4,5,6,7,8	—	✓	✓
Differential SSTL-18 (3)	3,4,5,6,7,8	—	✓	✓
Differential HSTL-18 (3)	3,4,5,6,7,8	—	✓	✓
Differential HSTL-15 (3)	3,4,5,6,7,8	—	✓	✓
Differential HSTL-12 (3)	4,5,6,7,8	—	✓	✓

Notes to Table 6–7:

(1) Transmitter and Receiver f_{MAX} depend on system topology and performance requirement.

(2) The LVPECL I/O standard is only supported on dedicated clock input pins.

(3) The differential SSTL-2, SSTL-18, HSTL-18, HSTL-15, and HSTL-12 I/O standards are only supported on clock input pins and PLL output clock pins. PLL output clock pins do not support Class II interface type of differential SSTL-18, HSTL-18, HSTL-15, and HSTL-12 I/O standards.

You can use I/O pins and internal logic to implement a high-speed differential interface in Cyclone IV devices. Cyclone IV devices do not contain dedicated serialization or deserialization circuitry. Therefore, shift registers, internal phase-locked loops (PLLs), and I/O cells are used to perform serial-to-parallel conversions on incoming data and parallel-to-serial conversion on outgoing data. The differential interface data serializers and deserializers (SERDES) are automatically constructed in the core logic elements (LEs) with the Quartus II software ALTLVDS megafunction.

Figure 6–16. RSDS, Mini-LVDS, or PPDS Interface with External Resistor Network on the Top and Bottom I/O Banks *(1)*

Note to Figure 6–16:

(1) R_S and R_P values are pending characterization.

A resistor network is required to attenuate the output voltage swing to meet RSDS, mini-LVDS, and PPDS specifications when using emulated transmitters. You can modify the resistor network values to reduce power or improve the noise margin.

The resistor values chosen must satisfy Equation 6–1.

Equation 6–1. Resistor Network

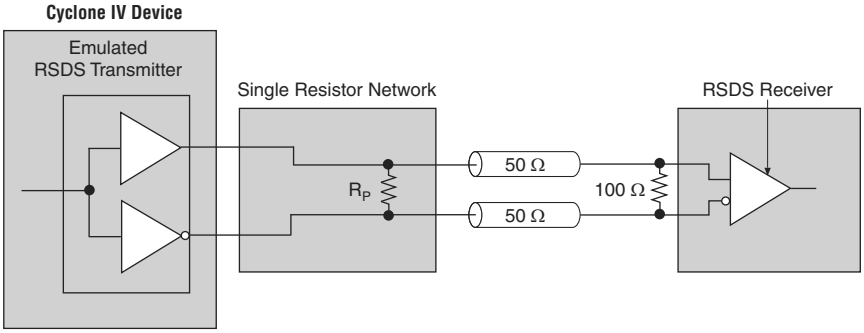
$$\frac{R_S \times \frac{R_P}{2}}{R_S + \frac{R_P}{2}} = 50\ \Omega$$

Altera recommends that you perform simulations using Cyclone IV devices IBIS models to validate that custom resistor values meet the RSDS, mini-LVDS, or PPDS requirements.

It is possible to use a single external resistor instead of using three resistors in the resistor network for an RSDS interface, as shown in Figure 6–17. The external single-resistor solution reduces the external resistor count while still achieving the required signaling level for RSDS. However, the performance of the single-resistor solution is lower than the performance with the three-resistor network.

Figure 6–17 shows the RSDS interface with a single resistor network on the top and bottom I/O banks.

Figure 6–17. RSDS Interface with Single Resistor Network on the Top and Bottom I/O Banks *(1)*



Note to Figure 6–17:

(1) R_P value is pending characterization.

Use the ACTIVE_DISENGAGE instruction with the CONFIG_IO instruction to interrupt configuration. Table 8–16 lists the sequence of instructions to use for various CONFIG_IO usage scenarios.

Table 8–16. JTAG CONFIG_IO (without JTAG_PROGRAM) Instruction Flows *(1)*

JTAG Instruction	Configuration Scheme and Current State of the Cyclone IV Device											
	Prior to User Mode (Interrupting Configuration)				User Mode				Power Up			
	PS	FPP	AS	AP	PS	FPP	AS	AP	PS	FPP	AS	AP
ACTIVE_DISENGAGE	O	O	O	O	O	O	O	O	—	—	—	—
CONFIG_IO	R	R	R	R	R	R	R	R	NA	NA	NA	NA
JTAG Boundary Scan Instructions (no JTAG_PROGRAM)	O	O	O	O	O	O	O	O	—	—	—	—
ACTIVE_ENGAGE	A	A	R *(2)*	R *(2)*	A	A	R *(2)*	R *(2)*	—	—	—	—
PULSE_NCONFIG			A *(3)*	A *(3)*			O	O	—	—	—	—
Pulse nCONFIG pin			A *(3)*	A *(3)*			O	O	—	—	—	—
JTAG TAP Reset	R	R	R	R	R	R	R	R	—	—	—	—

Notes to Table 8–16:

(1) You must execute "R" indicates that the instruction before the next instruction, "O" indicates the optional instruction, "A" indicates that the instruction must be executed, and "NA" indicates that the instruction is not allowed in this mode.

(2) Required if you use ACTIVE_DISENGAGE.

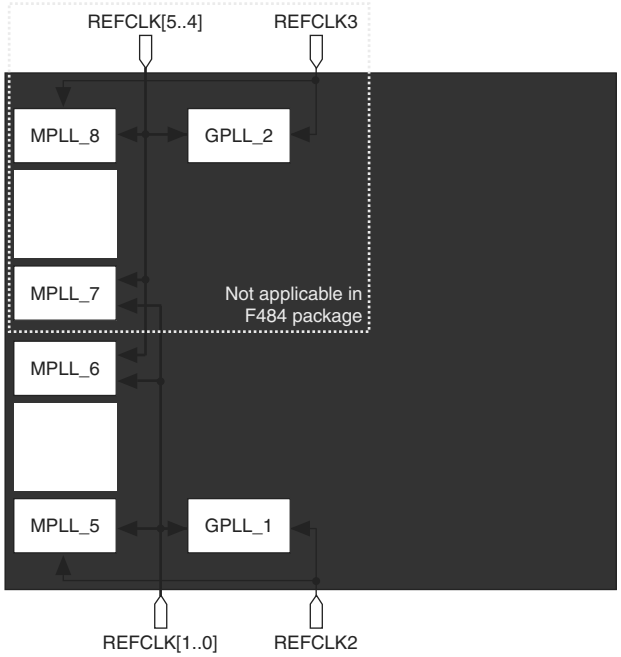
(3) Neither of the instruction is required if you use ACTIVE_ENGAGE.

The CONFIG_IO instruction does not hold nSTATUS low until reconfiguration. You must disengage the AS or AP configuration controller by issuing the ACTIVE_DISENGAGE and ACTIVE_ENGAGE instructions when active configuration is interrupted. You must issue the ACTIVE_DISENGAGE instruction alone or prior to the CONFIG_IO instruction if the JTAG_PROGRAM instruction is to be issued later (Table 8–17). This puts the active configuration controllers into the idle state. The active configuration controller is re-engaged after user mode is reached through JTAG programming (Table 8–17).

While executing the CONFIG_IO instruction, all user I/Os are tri-stated.

If reconfiguration after interruption is performed using configuration modes (rather than using JTAG_PROGRAM), it is not necessary to issue the ACTIVE_DISENGAGE instruction prior to CONFIG_IO. You can start reconfiguration by either pulling nCONFIG low for at least 500 ns or issuing the PULSE_NCONFIG instruction. If the ACTIVE_DISENGAGE instruction was issued and the JTAG_PROGRAM instruction fails to enter user mode, you must issue the ACTIVE_ENGAGE instruction to reactivate the active configuration controller. Issuing the ACTIVE_ENGAGE instruction also triggers reconfiguration in configuration modes; therefore, it is not necessary to pull nCONFIG low or issue the PULSE_NCONFIG instruction.

Figure 1–26. PLL Input Reference Clocks in Transceiver Operation for F484 and Larger Packages (1), (2), (3)



Notes to Figure 1–26:

(1) The REFCLK2 and REFCLK3 pins are dual-purpose CLKIO, REFCLK, or DIFFCLK pins that reside in banks 3A and 8A respectively.

(2) The REFCLK[1..0] and REFCLK[5..4] pins are dual-purpose differential REFCLK or DIFFCLK pins that reside in banks 3B and 8B respectively. These clock input pins do not have access to the clock control blocks and GCLK networks. For more details, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter.

(3) Using any clock input pins other than the designated REFCLK pins as shown here to drive the MPLLs and GPLLs may have reduced jitter performance.

The input reference clocks reside in banks 3A, 3B, 8A, and 8B have dedicated $V_{CC\_CLKIN3A}$, $V_{CC\_CLKIN3B}$, $V_{CC\_CLKIN8A}$, and $V_{CC\_CLKIN8B}$ power supplies separately in their respective I/O banks to avoid the different power level requirements in the same bank for general purpose I/Os (GPIOs). Table 1–6 lists the supported I/O standard for the REFCLK pins.

Table 1–6. REFCLK I/O Standard Support

I/O Standard	HSSI Protocol	Coupling	Termination	VCC_CLKIN Level		I/O Pin Type		
				Input	Output	Column I/O	Row I/O	Supported Banks
LVDS	ALL	Differential AC (Needs off-chip resistor to restore V_{CM})	Off-chip	2.5 V	Not Supported	Yes	No	3A, 3B, 8A, 8B
LVPECL	ALL		Off-chip	2.5 V	Not Supported	Yes	No	3A, 3B, 8A, 8B
1.2 V, 1.5 V, 3.3 V PCML	ALL		Off-chip	2.5 V	Not Supported	Yes	No	3A, 3B, 8A, 8B
	ALL		Off-chip	2.5 V	Not Supported	Yes	No	3A, 3B, 8A, 8B
	ALL		Off-chip	2.5 V	Not Supported	Yes	No	3A, 3B, 8A, 8B
HCSL	PCIe	Differential DC	Off-chip	2.5 V	Not Supported	Yes	No	3A, 3B, 8A, 8B

- transmitter in electrical idle
- receiver signal detect
- receiver spread spectrum clocking

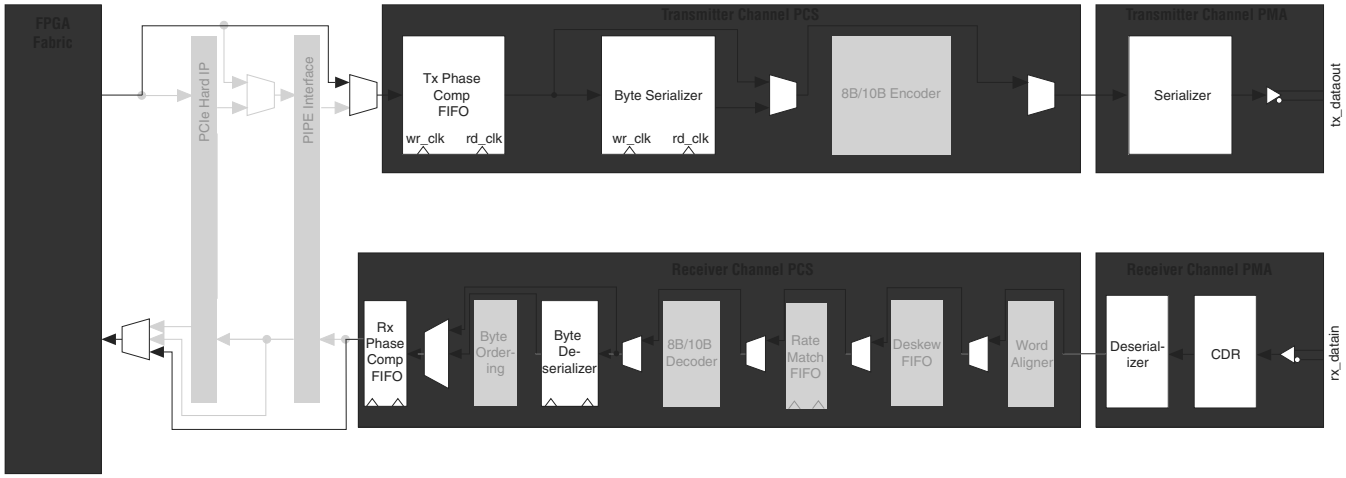
Low-Latency PCS Operation

When configured in low-latency PCS operation, the following blocks in the transceiver PCS are bypassed, resulting in a lower latency PCS datapath:

- 8B/10B encoder and decoder
- word aligner
- rate match FIFO
- byte ordering

Figure 1–47 shows the transceiver channel datapath in Basic mode with low-latency PCS operation.

Figure 1–47. Transceiver Channel Datapath in Basic Mode with Low-Latency PCS Operation



Transmitter in Electrical Idle

The transmitter buffer supports electrical idle state, where when enabled, the differential output buffer driver is tri-stated. During electrical idle, the output buffer assumes the common mode output voltage levels. For details about the electrical idle features, refer to "PCI Express (PIPE) Mode" on page 1–52.

The transmitter in electrical idle feature is required for compliance to the version 2.00 of PHY Interface for the PCI Express (PIPE) Architecture specification for PCIe protocol implementation.

Signal Detect at Receiver

Signal detect at receiver is only supported when 8B/10B encoder/decoder block is enabled.

All Supported Functional Modes Except the PCIe Functional Mode

This section describes reset sequences for transceiver channels in bonded and non-bonded configurations. Timing diagrams of some typical configurations are shown to facilitate proper reset sequence implementation. In these functional modes, you can set the receiver CDR either in automatic lock or manual lock mode.

In manual lock mode, the receiver CDR locks to the reference clock (lock-to-reference) or the incoming serial data (lock-to-data), depending on the logic levels on the `rx_locktorefclk` and `rx_locktodata` signals. With the receiver CDR in manual lock mode, you can either configure the transceiver channels in the Cyclone IV GX device in a non-bonded configuration or a bonded configuration. In a bonded configuration, for example in XAUI mode, four channels are bonded together.

Table 2–4 lists the lock-to-reference (LTR) and lock-to-data (LTD) controller lock modes for the `rx_locktorefclk` and `rx_locktodata` signals.

Table 2–4. Lock-To-Reference and Lock-To-Data Modes

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual, LTR Mode
—	1	Manual, LTD Mode
0	0	Automatic Lock Mode

Bonded Channel Configuration

In a bonded channel configuration, you can reset all the bonded channels simultaneously. Examples of bonded channel configurations are the XAUI, PCIe Gen1 ×2 and ×4, and Basic ×2 and ×4 functional modes. In Basic ×2 and ×4 functional mode, you can bond **Transmitter Only** channels together.

In XAUI mode, the receiver and transmitter channels are bonded. Each of the receiver channels in this mode has its own `rx_freqlocked` output status signals. You must consider the timing of these signals in the reset sequence.

Table 2–5 lists the reset and power-down sequences for bonded configurations under the stated functional modes.

Table 2–5. Reset and Power-Down Sequences for Bonded Channel Configurations

Channel Set Up	Receiver CDR Mode	Refer to
Transmitter Only	Basic ×2 and ×4	"Transmitter Only Channel" on page 2–7
Receiver and Transmitter	Automatic lock mode for XAUI functional mode	"Receiver and Transmitter Channel—Receiver CDR in Automatic Lock Mode" on page 2–8
Receiver and Transmitter	Manual lock mode for XAUI functional mode	"Receiver and Transmitter Channel—Receiver CDR in Manual Lock Mode" on page 2–9

The **Offset cancellation for Receiver channels** option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver and Transmitter** and **Receiver only** configurations. It is not available for **Transmitter only** configurations. For **Receiver and Transmitter** and **Receiver only** configurations, you must connect the necessary interface signals between the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

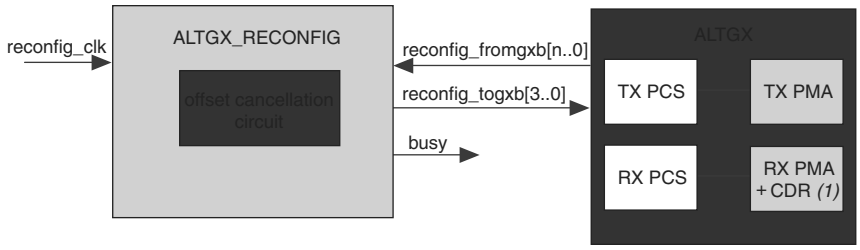
Offset cancellation is automatically executed once every time the device is powered on. The control logic for offset cancellation is integrated into the dynamic reconfiguration controller. You must connect the ALTGX_RECONFIG instance to the ALTGX instances (with receiver channels) in your design. You must connect the `reconfig_fromgxb`, `reconfig_togxb`, and necessary clock signals to both the ALTGX_RECONFIG and ALTGX (with receiver channels) instances.

When the device powers up, the dynamic reconfiguration controller initiates offset cancellation on the receiver channel by disconnecting the receiver input pins from the receiver data path. Subsequently, the offset cancellation process goes through different states and culminates in the offset cancellation of the receiver buffer.

☞ Offset cancellation process only occurs one time after power up and does not occur when subsequent `reconfig_reset` is asserted. If you assert `reconfig_reset` after the offset cancellation process is completed, the offset cancellation process will not run again.
If you assert `reconfig_reset` upon power up; offset cancellation will not begin until `reconfig_reset` is deasserted. If you assert `reconfig_reset` after power up but before offset cancellation process is completed; offset cancellation will not complete and restart only when `reconfig_reset` is deasserted.

Figure 3–2 shows the connection for offset cancellation mode.

Figure 3–2. ALTGX and ALTGX_RECONFIG Connection for the Offset Cancellation Process



Note to Figure 3–2:

(1) This block is active during the offset cancellation process.

☞ The dynamic reconfiguration controller sends and receives data to the transceiver channel through the `reconfig_togxb` and `reconfig_fromgxb` signals.

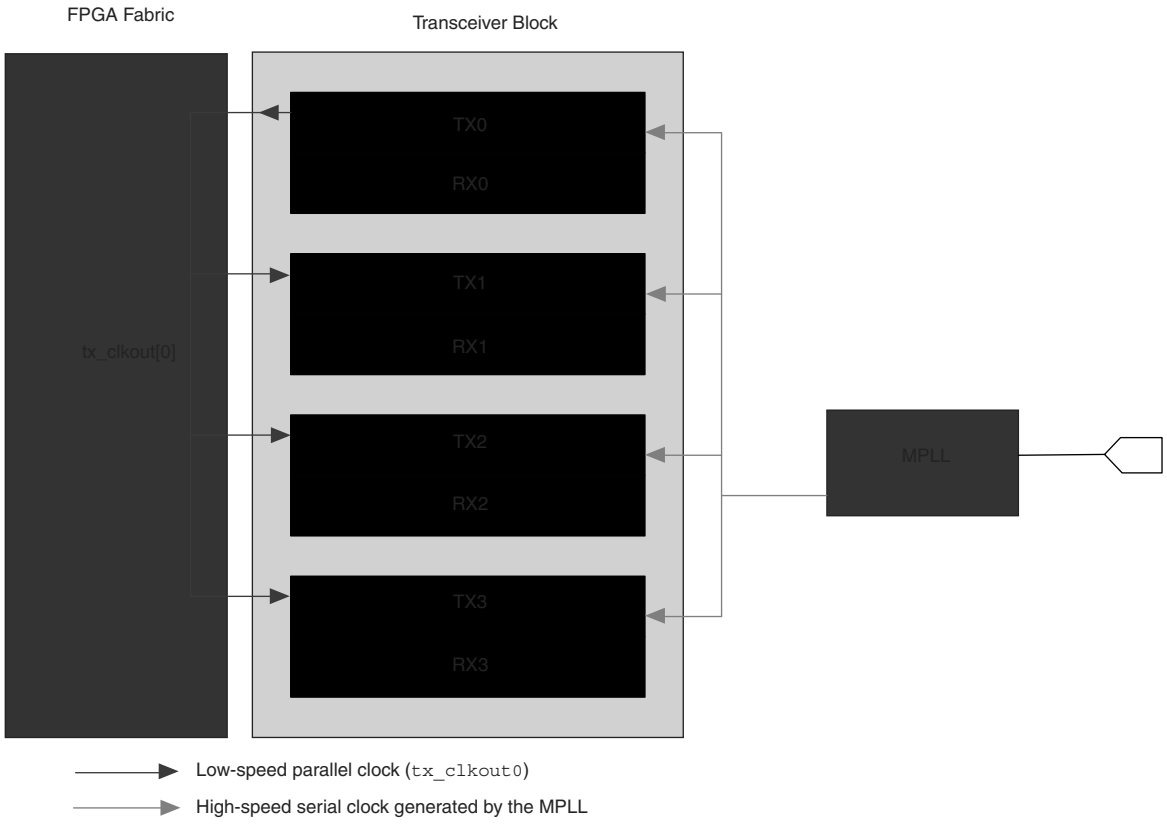
☞ The `gxb_powerdown` signal must not be asserted during the offset cancellation sequence.

Option 1: Share a Single Transmitter Core Clock Between Transmitters

- Enable this option if you want `tx_clkout` of the first channel (channel 0) of the transceiver block to provide the write clock to the Transmitter Phase Compensation FIFOs of the remaining channels in the transceiver block.
- This option is typically enabled when all the channels of a transceiver block have the same functional mode and data rate and are reconfigured to the identical functional mode and data rate.

Figure 3–11 shows the sharing of channel 0's `tx_clkout` between all four regular channels of a transceiver block.

Figure 3–11. Option 1 for Transmitter Core Clocking (Channel Reconfiguration Mode)

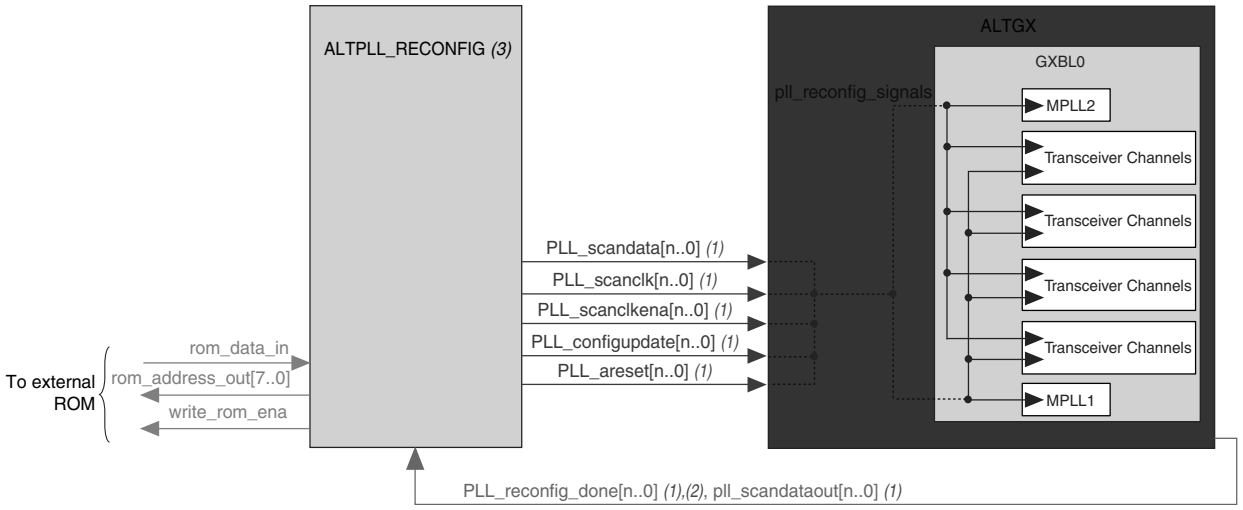


The .**mif** files carries the reconfiguration information that will be used to reconfigure the multipurpose PLL or general purpose PLL dynamically. The .**mif** contents is generated automatically when you select the **Enable PLL Reconfiguration** option in the **Reconfiguration Setting** in ALTGX instances. The **.mif** files will be generated based on the data rate and input reference clock setting in the ALTGX MegaWizard. You must use the external ROM and feed its content to the ALTPLL_RECONFIG megafunction to reconfigure the multipurpose PLL setting.

For more information about instantiating the ALTPLL_Reconfig, refer to the *AN 609: Implementing Dynamic Reconfiguration in Cyclone IV GX Devices*.

Figure 3–16 shows the connection for PLL reconfiguration mode.

Figure 3–16. ALTGX and ALTPLL_RECONFIG Connection for PLL Reconfiguration Mode



Notes to Figure 3–16:

(1) *<n>* = (number of transceiver PLLs configured in the ALTGX MegaWizard) - 1.

(2) You must connect the `pll_reconfig_done` signal from the ALTGX to the `pll_scandone` port from ALTPLL_RECONFIG.

(3) You need two ALTPLL_RECONFIG controllers if you have two separate ALTGX instances with transceiver PLL instantiated in each ALTGX instance.

For more information about connecting the ALTPLL_RECONFIG and ALTGX instances, refer to the *AN 609: Implementing Dynamic Reconfiguration in Cyclone IV GX Devices*.

Table 1–42 and Table 1–43 list the IOE programmable delay for Cyclone IV E 1.2 V core voltage devices.

Table 1–42. IOE Programmable Delay on Column Pins for Cyclone IV E 1.2 V Core Voltage Devices *(1), (2)*

Parameter	Paths Affected	Number of Setting	Min Offset	Max Offset								Unit
				Fast Corner			Slow Corner					
				C6	I7	A7	C6	C7	C8	I7	A7	
Input delay from pin to internal cells	Pad to I/O dataout to core	7	0	1.314	1.211	1.211	2.177	2.340	2.433	2.388	2.508	ns
Input delay from pin to input register	Pad to I/O input register	8	0	1.307	1.203	1.203	2.19	2.387	2.540	2.430	2.545	ns
Delay from output register to output pin	I/O output register to pad	2	0	0.437	0.402	0.402	0.747	0.820	0.880	0.834	0.873	ns
Input delay from dual-purpose clock pin to fan-out destinations	Pad to global clock network	12	0	0.693	0.665	0.665	1.200	1.379	1.532	1.393	1.441	ns

Notes to Table 1–42:

(1) The incremental values for the settings are generally linear. For the exact values for each setting, use the latest version of the Quartus II software.

(2) The minimum and maximum offset timing numbers are in reference to setting **0** as available in the Quartus II software.

Table 1–43. IOE Programmable Delay on Row Pins for Cyclone IV E 1.2 V Core Voltage Devices *(1), (2)*

Parameter	Paths Affected	Number of Setting	Min Offset	Max Offset								Unit
				Fast Corner			Slow Corner					
				C6	I7	A7	C6	C7	C8	I7	A7	
Input delay from pin to internal cells	Pad to I/O dataout to core	7	0	1.314	1.209	1.209	2.201	2.386	2.510	2.429	2.548	ns
Input delay from pin to input register	Pad to I/O input register	8	0	1.312	1.207	1.207	2.202	2.402	2.558	2.447	2.557	ns
Delay from output register to output pin	I/O output register to pad	2	0	0.458	0.419	0.419	0.783	0.861	0.924	0.875	0.915	ns
Input delay from dual-purpose clock pin to fan-out destinations	Pad to global clock network	12	0	0.686	0.657	0.657	1.185	1.360	1.506	1.376	1.422	ns

Notes to Table 1–43:

(1) The incremental values for the settings are generally linear. For the exact values for each setting, use the latest version of the Quartus II software.

(2) The minimum and maximum offset timing numbers are in reference to setting **0** as available in the Quartus II software.