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	2475
Number of Logic Elements/Cells	39600
Total RAM Bits	1161216
Number of I/O	328
Number of Gates	-
Voltage - Supply	1.15V ~ 1.25V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	484-BGA
Supplier Device Package	484-FBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/intel/ep4ce40f23c8

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.

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

Parity Bit Support

Parity checking for error detection is possible with the parity bit along with internal logic resources. Cyclone IV devices 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.

Byte Enable Support

Cyclone IV devices M9K memory blocks support byte enables that mask the input data so that only specific bytes of data are written. The unwritten bytes retain the previous written value. The `wren` signals, along with the byte-enable (`byteena`) signals, control the write operations of the RAM block. The default value of the `byteena` signals is high (enabled), in which case writing is controlled only by the `wren` signals. There is no clear port to the `byteena` registers. M9K blocks support byte enables when the write port has a data width of ×16, ×18, ×32, or ×36 bits.

Byte enables operate in one-hot manner, with the LSB of the `byteena` signal corresponding to the least significant byte 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.

Table 3–2 lists the byte selection.

Table 3–2. byteena for Cyclone IV Devices M9K Blocks *(1)*

byteena[3..0]	Affected Bytes			
	datain ×16	datain ×18	datain ×32	datain ×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]

Note to Table 3–2:

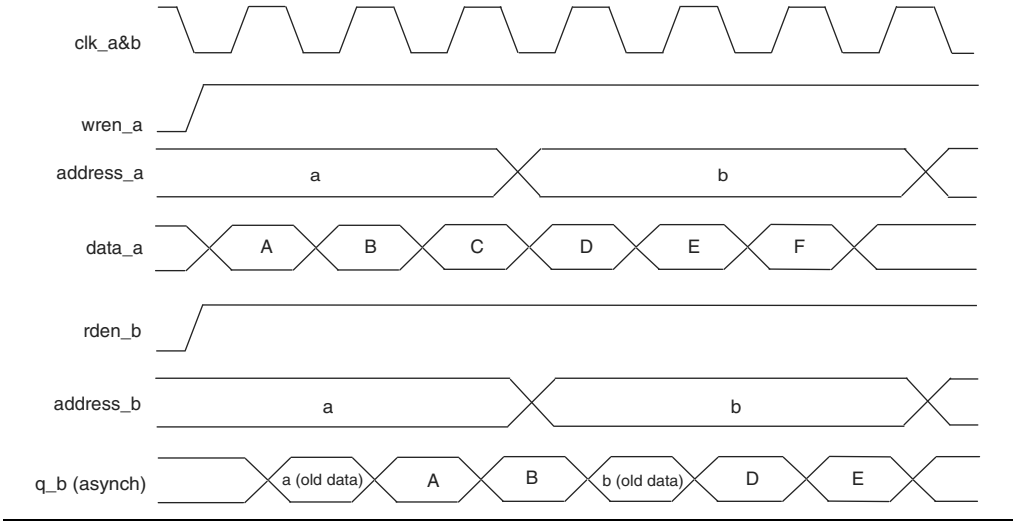
(1) Any combination of byte enables is possible.

In this mode, you also have two output choices: **Old Data** mode or **Don't Care** mode. In **Old Data** mode, a read-during-write operation to different ports causes the RAM outputs to reflect the old data at that address location. In **Don't Care** mode, the same operation results in a "Don't Care" or unknown value on the RAM outputs.

For more information about how to implement the desired behavior, refer to the *RAM Megafunction User Guide*.

Figure 3–16 shows a sample functional waveform of mixed port read-during-write behavior for **Old Data** mode. In **Don't Care** mode, the old data is replaced with "Don't Care".

Figure 3–16. Mixed Port Read-During-Write: Old Data Mode

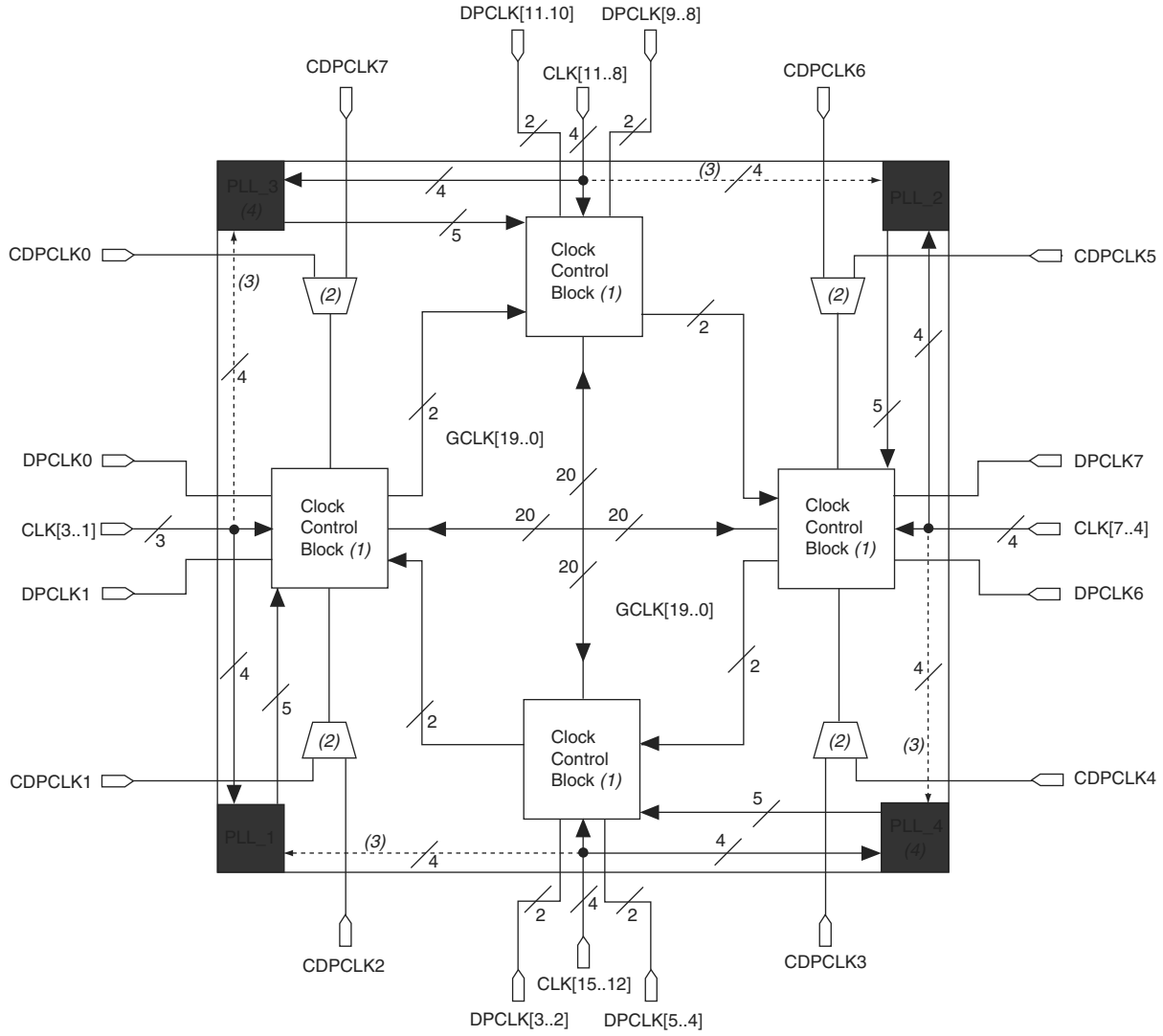


For mixed-port read-during-write operation with dual clocks, the relationship between the clocks determines the output behavior of the memory. If you use the same clock for the two clocks, the output is the old data from the address location. However, if you use different clocks, the output is unknown 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.

Conflict Resolution

When you are using M9K memory blocks in true dual-port mode, it is possible to attempt two write operations to the same memory location (address). Because there is no conflict resolution circuitry built into M9K memory blocks, this results in unknown data being written to that location. Therefore, you must implement conflict-resolution logic external to the M9K memory block.

Figure 5–4. Clock Networks and Clock Control Block Locations in Cyclone IV E Devices



Notes to Figure 5–4:

(1) There are five clock control blocks on each side.

(2) Only one of the corner CDPCLK pins in each corner can feed the clock control block at a time. You can use the other CDPCLK pins as general-purpose I/O (GPIO) pins.

(3) Dedicated clock pins can feed into this PLL. However, these paths are not fully compensated.

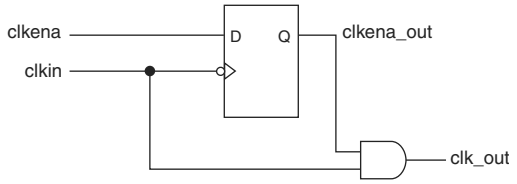
(4) PLL_3 and PLL_4 are not available in EP4CE6 and EP4CE10 devices.

The inputs to the clock control blocks on each side of the Cyclone IV GX device must be chosen from among the following clock sources:

- Four clock input pins
- Ten PLL counter outputs (five from each adjacent PLLs)
- Two, four, or six DPCLK pins from the top, bottom, and right sides of the device
- Five signals from internal logic

Figure 5–7 shows how to implement the clkena signal with a single register.

Figure 5–7. clkena Implementation

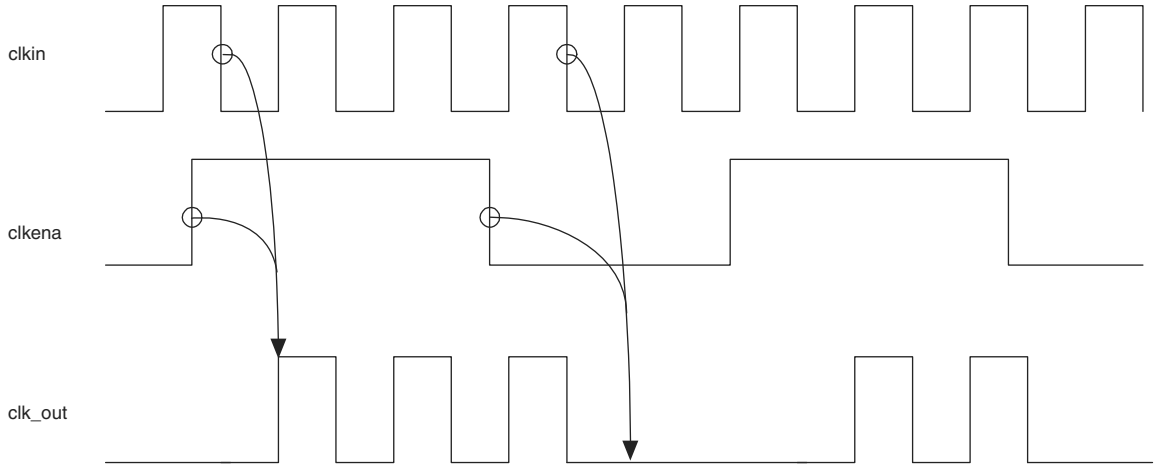


The clkena circuitry controlling the output C0 of the PLL to an output pin is implemented with two registers instead of a single register, as shown in Figure 5–7.

Figure 5–8 shows the waveform example for a clock 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.

Figure 5–8. clkena Implementation: Output Enable



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

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

1. Disable the primary output clock by de-asserting 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 you must wait before enabling the secondary clock is design-dependent. You can build custom logic to ensure glitch-free transition when switching between different clock sources.

Deterministic Latency Compensation Mode

The deterministic latency mode compensates for the delay of the multipurpose PLLs through the clock network and serializer in Common Public Radio Interface (CPRI) applications. In this mode, the PLL PFD feedback path compensates the latency uncertainty in `Tx dataout` and `Tx clkout` paths relative to the reference clock.

Hardware Features

Cyclone IV PLLs support several features for general-purpose clock management. This section discusses clock multiplication and division implementation, phase shifting implementations, and programmable duty cycles.

Clock Multiplication and Division

Each Cyclone IV PLL provides clock synthesis for PLL output ports using M/(N*post-scale counter) scaling factors. The input clock is divided by a pre-scale factor, N, and is then multiplied by the M feedback factor. The control loop drives the VCO to match f_{IN} (M/N). Each output port has a unique post-scale counter that divides down the high-frequency VCO. For multiple PLL outputs with different frequencies, the VCO value is the least common multiple of the output frequencies that meets its frequency specifications. For example, if output frequencies required from one PLL are 33 and 66 MHz, the Quartus II software sets the VCO to 660 MHz (the least common multiple of 33 and 66 MHz in the VCO range). Then, the post-scale counters scale down the VCO frequency for each output port.

There is one pre-scale counter, N, and one multiply counter, M, per PLL, with a range of 1 to 512 for both M and N. The N counter does not use duty cycle control because the purpose of this counter is only to calculate frequency division. There are five generic post-scale counters per PLL that can feed GCLKs or external clock outputs. These post-scale counters range from 1 to 512 with a 50% duty cycle setting. The post-scale counters range from 1 to 256 with any non-50% duty cycle setting. The sum of the high/low count values chosen for a design selects the divide value for a given counter.

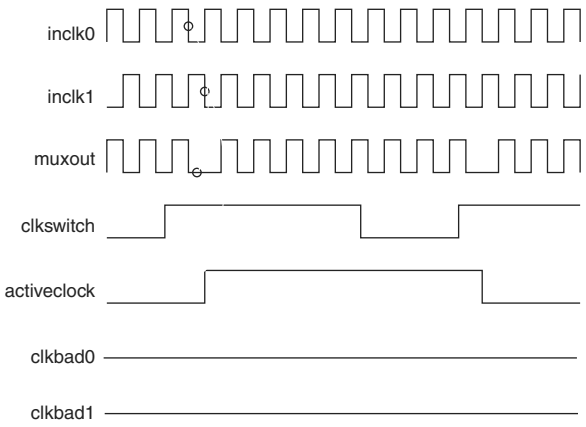
The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered into the ALTPLL megafunction.

Phase alignment between output counters is determined using the t_{PLL_PSERR} specification.

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*.

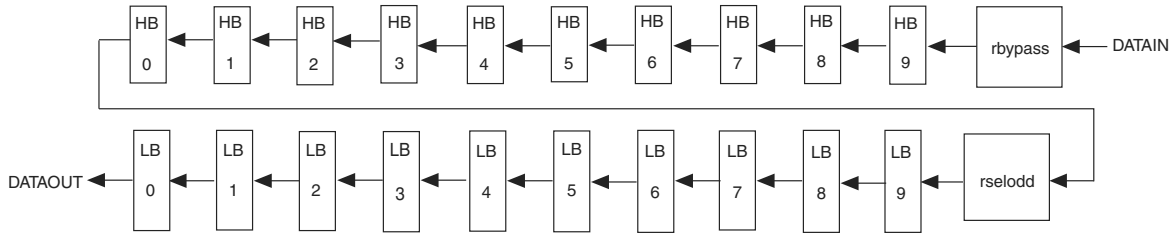
Guidelines

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.

Figure 5–25 shows the scan chain bit order sequence for one PLL post-scale counter in PLLs of Cyclone IV devices.

Figure 5–25. Scan Chain Bit Order



Charge Pump and Loop Filter

You can reconfigure the charge pump and loop filter settings to update the PLL bandwidth in real time. Table 5–8 through Table 5–10 list the possible settings for charge pump current (I_{CP}), loop filter resistor (R), and capacitor (C) values for PLLs of Cyclone IV devices.

Table 5–8. Charge Pump Bit Control

CP[2]	CP[1]	CP[0]	Setting (Decimal)
0	0	0	0
1	0	0	1
1	1	0	3
1	1	1	7

Table 5–9. 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 6–3. Cyclone IV Devices Supported I/O Standards and Constraints (Part 3 of 3)

I/O Standard	Type	Standard Support	V_{CCIO} Level (in V)		Column I/O Pins			Row I/O Pins (1)	
			Input	Output	CLK, DQS	PLL_OUT	User I/O Pins	CLK, DQS	User I/O Pins
LVPECL (7)	Differential	—	2.5	—	✓	—	—	✓	—

Notes to Table 6–3:

(1) Cyclone IV GX devices only support right I/O pins.

(2) The PCI-clamp diode must be enabled for 3.3-V/3.0-V LVTTL/LVCMOS.

(3) The Cyclone IV architecture supports the MultiVolt I/O interface feature that allows Cyclone IV devices in all packages to interface with I/O systems that have different supply voltages.

(4) Cyclone IV GX devices do not support 1.2-V V_{CCIO} in banks 3 and 9. I/O pins in bank 9 are dual-purpose I/O pins that are used as configuration or GPIO pins. Configuration scheme is not support at 1.2 V, therefore bank 9 can not be powered up at 1.2-V V_{CCIO}.

(5) Differential HSTL and SSTL outputs use two single-ended outputs with the second output programmed as inverted. Differential HSTL and SSTL inputs treat differential inputs as two single-ended HSTL and SSTL inputs and only decode one of them. Differential HSTL and SSTL are only supported on CLK pins.

(6) PPDS, mini-LVDS, and RSDS are only supported on output pins.

(7) LVPECL is only supported on clock inputs.

(8) Bus LVDS (BLVDS) output uses two single-ended outputs with the second output programmed as inverted. BLVDS input uses LVDS input buffer.

(9) 1.2-V HSTL input is supported at both column and row I/Os regardless of Class I or Class II.

(10) True LVDS, RSDS, and mini-LVDS I/O standards are supported in left and right I/O pins, while emulated LVDS, RSDS, and mini-LVDS I/O standards are supported in the top, bottom, and right I/O pins.

Cyclone IV devices support PCI and PCI-X I/O standards at 3.0-V V_{CCIO}. The 3.0-V PCI and PCI-X I/O are fully compatible for direct interfacing with 3.3-V PCI systems without requiring any additional components. The 3.0-V PCI and PCI-X outputs meet the V_{IH} and V_{IL} requirements of 3.3-V PCI and PCI-X inputs with sufficient noise margin.

For more information about the 3.3/3.0/2.5-V LVTTL & LVCMOS multivolt I/O support, refer to *AN 447: Interfacing Cyclone III and Cyclone IV Devices with 3.3/3.0/2.5-V LVTTL/LVCMOS I/O Systems*.

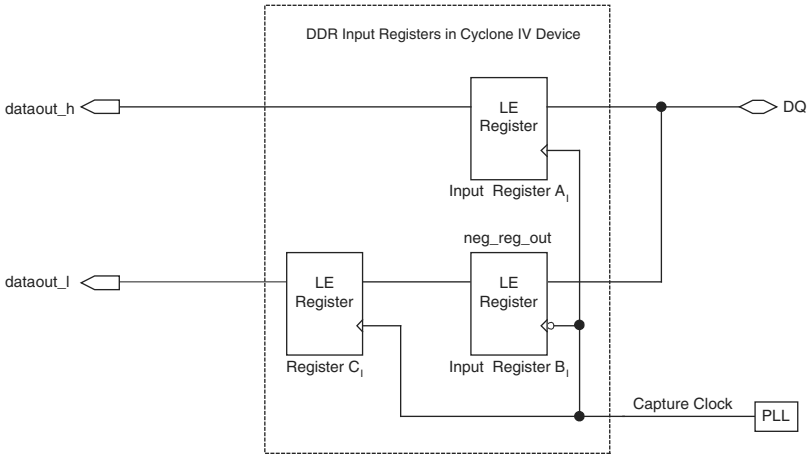
Termination Scheme for I/O Standards

This section describes recommended termination schemes for voltage-referenced and differential I/O standards.

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, 3.0-V PCI, and PCI-X I/O standards do not specify a recommended termination scheme per the JEDEC standard

Figure 7–7 illustrates Cyclone IV DDR input registers.

Figure 7–7. Cyclone IV DDR Input Registers



These DDR input registers are implemented in the core of devices. The DDR data is first fed to two registers, input register A_I and input register B_I.

- Input register A_I captures the DDR data present during the rising edge of the clock
- Input register B_I captures the DDR data present during the falling edge of the clock
- Register C_I aligns the data before it is synchronized with the system clock

The data from the DDR input register is fed to two registers, `sync_reg_h` and `sync_reg_l`, then the data is typically transferred to a FIFO block to synchronize the two data streams to the rising edge of the system clock. Because the read-capture clock is generated by the PLL, the read-data strobe signal (`DQS` or CQ) is not used during read operation in Cyclone IV devices; hence, postamble is not a concern in this case.

The **.rbf** used by the JRunner software driver cannot be a compressed **.rbf** because the JRunner software driver uses JTAG-based configuration. During JTAG-based configuration, the real-time decompression feature is not available.

For more information about the JRunner software driver, refer to *AN 414: JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration* and the source files on the Altera website at (www.altera.com).

Combining JTAG and AS Configuration Schemes

You can combine the AS configuration scheme with the JTAG-based configuration (Figure 8–28). This setup uses two 10-pin download cable headers on the board. One download cable is used in JTAG mode to configure the Cyclone IV device directly through the JTAG interface. The other download cable is used in AS mode to program the serial configuration device in-system through the AS programming interface. If you try configuring the device using both schemes simultaneously, JTAG configuration takes precedence and AS configuration terminates.

Table 8–25 lists the contents of previous state register 1 and previous state register 2 in the status register. The status register bit in Table 8–25 shows the bit positions in a 3-bit register. The previous state register 1 and previous state register 2 have the same bit definitions. The previous state register 1 reflects the current application configuration and the previous state register 2 reflects the previous application configuration.

Table 8–25. Remote System Upgrade Previous State Register 1 and Previous State Register 2 Contents in Status Register

Status Register Bit	Definition	Description
30	nCONFIG source	One-hot, active-high field that describes the reconfiguration source that caused the Cyclone IV device to leave the previous application configuration. If there is a tie, the higher bit order indicates precedence. For example, if nCONFIG and remote system upgrade nCONFIG reach the reconfiguration state machine at the same time, the nCONFIG precedes the remote system upgrade nCONFIG.
29	CRC error source	
28	nSTATUS source	
27	User watchdog timer source	
26	Remote system upgrade nCONFIG source	
25:24	Master state machine current state	The state of the master state machine during reconfiguration causes the Cyclone IV device to leave the previous application configuration.
23:0	Boot address	The address used by the configuration scheme to load the previous application configuration.

If a capture is inappropriately done while capturing a previous state before the system has entered remote update application configuration for the first time, a value outputs from the shift register to indicate that the capture is incorrectly called.

Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (Table 8–22 on page 8–75). While both registers can only be updated when the device is loaded with a factory configuration image, the update register writes are controlled by the user logic, and the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic should send the option bits (Cd_early and Osc_int), the configuration address, and watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes high, the remote system upgrade state machine updates the control register with the contents of the update register and starts system reconfiguration from the new application page.

☞ To ensure the successful reconfiguration between the pages, assert the RU_nCONFIG signal for a minimum of 250 ns. This is equivalent to strobing the reconfig input of the ALTREMOTE_UPDATE megafunction high for a minimum of 250 ns.

If there is an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (based on mode and error condition) by setting the control register accordingly.

Table 8–26 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

☞ WYSIWYG is an optimization technique that performs optimization on a VQM (Verilog Quartus Mapping) netlist in the Quartus II software.

Error Detection Block

Table 9–3 lists the types of CRC detection to check the configuration bits.

Table 9–3. Types of CRC Detection to Check the Configuration Bits

First Type of CRC Detection	Second Type of CRC Detection
■ CRAM error checking ability (32-bit CRC) during user mode, for use by the CRC_ERROR pin. ■ There is only one 32-bit CRC value. This value covers all the CRAM data.	■ 16-bit CRC embedded in every configuration data frame. ■ During configuration, after a frame of data is loaded into the device, the pre-computed CRC is shifted into the CRC circuitry. ■ Simultaneously, the CRC value for the data frame shifted-in is calculated. If the pre-computed CRC and calculated CRC values do not match, nSTATUS is set low. ■ Every data frame has a 16-bit CRC. Therefore, there are many 16-bit CRC values for the whole configuration bit stream. ■ Every device has a different length of configuration data frame.

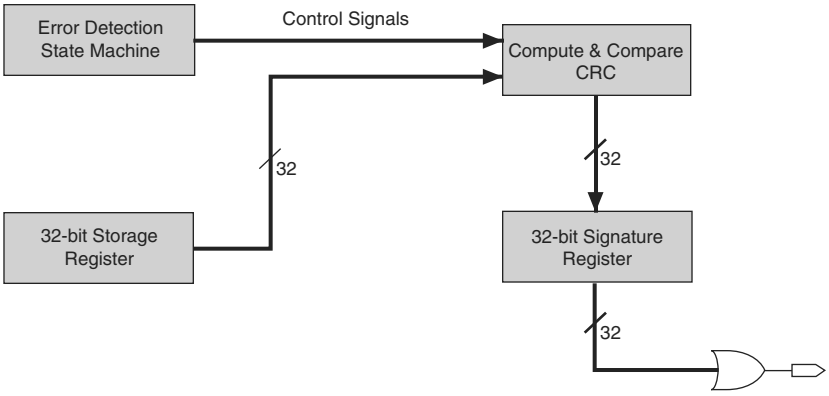
This section focuses on the first type—the 32-bit CRC when the device is in user mode.

Error Detection Registers

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 set high.

Figure 9–1 shows the block diagram of the error detection block and the two related 32-bit registers: the signature register and the storage register.

Figure 9–1. Error Detection Block Diagram



Word Aligner

Figure 1–16 shows the word aligner block diagram. The word aligner receives parallel data from the deserializer and restores the word boundary based on a pre-defined alignment pattern that must be received during link synchronization. The word aligner supports three operational modes as listed in Table 1–3.

Figure 1–16. Word Aligner Block Diagram

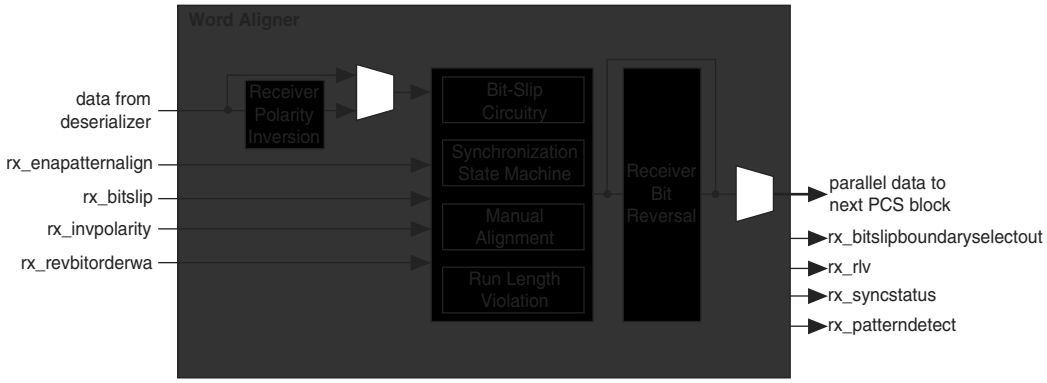


Table 1–3. Word Aligner Modes

Modes	PMA-PCS Interface Widths	Allowed Word Alignment Pattern Lengths
Manual Alignment	8-bit	16 bits
	10-bit	7 or 10 bits
Bit-Slip	8-bit	16 bits
	10-bit	7 or 10 bits
Automatic Synchronization State Machine	10-bit	7 or 10 bits

Manual Alignment Mode

In manual alignment mode, the `rx_enapatternalign` port controls the word aligner with either an 8- or 10-bit data width setting.

The 8-bit word aligner is edge-sensitive to the `rx_enapatternalign` signal. A rising edge on `rx_enapatternalign` signal after deassertion of the `rx_digitalreset` signal triggers the word aligner to look for the word alignment pattern in the received data stream. It updates the word boundary if it finds the word alignment pattern in a new word boundary. Any word alignment pattern received thereafter in a different word boundary causes the word aligner to re-align to the new word boundary only if there is a rising edge in the `rx_enapatternalign` signal.

The 10-bit word aligner is level-sensitive to the `rx_enapatternalign` signal. The word aligner looks for the programmed 7-bit or 10-bit word alignment pattern or its complement in the received data stream, if the `rx_enapatternalign` signal is held high. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If the `rx_enapatternalign` signal is deasserted, the word aligner maintains the current word boundary even when it receives the word alignment pattern in a new word boundary.

Table 1–4 lists the synchronization state machine parameters for the word aligner in this mode.

Table 1–4. Synchronization State Machine Parameters

Parameter	Allowed Values
Number of erroneous code groups received to lose synchronization	1–64
Number of continuous good code groups received to reduce the error count by one	1–256

After deassertion of the rx_digitalreset signal in automatic synchronization state machine mode, the word aligner starts looking for the synchronization code groups, word alignment pattern or its complement in the received data stream. When the programmed number of valid synchronization code groups or ordered sets are received, the rx_syncstatus signal is driven high to indicate that synchronization is acquired. The rx_syncstatus signal is constantly driven high until the programmed number of erroneous code groups are received without receiving intermediate good groups; after which the rx_syncstatus signal is driven low. The word aligner indicates loss of synchronization (rx_syncstatus signal remains low) until the programmed number of valid synchronization code groups are received again.

In addition to restoring word boundaries, the word aligner supports the following features:

- Programmable run length violation detection—detects consecutive 1s or 0s in the data stream, and asserts run length violation signal (rx_rlv) when a preset run length threshold (maximum number of consecutive 1s or 0s) is detected. The rx_rlv signal in each channel is clocked by its parallel recovered clock and is asserted for a minimum of two recovered clock cycles to ensure that the FPGA fabric clock can latch the rx_rlv signal reliably because the FPGA fabric clock might have phase differences, ppm differences (in asynchronous systems), or both, with the recovered clock. Table 1–5 lists the run length violation circuit detection capabilities.

Table 1–5. Run Length Violation Circuit Detection Capabilities

Supported Data Width	Detector Range		Increment Step Settings
	Minimum	Maximum	
8-bit	4	128	4
10-bit	5	160	5

Table 3–2. Dynamic Reconfiguration Controller Port List (ALTGX_RECONFIG Instance) (Part 5 of 7)

<table>
<tr><th>Port Name</th><th>Input/ Output</th><th>Description</th></tr>
<tr><td>tx_preemp[4..0] (1)</td><td>Input</td><td>This is an optional pre-emphasis write control for the transmit buffer. Depending on what value you set at this input, the controller dynamically writes the value to the pre-emphasis control register of the transmit buffer.
The width of this signal is fixed to 5 bits if you enable either the Use 'logical_channel_address' port for Analog controls reconfiguration option or the Use same control signal for all the channels option in the Analog controls screen. Otherwise, the width of this signal is 5 bits per channel.
<table>
<tr><th>tx_preemp[4..0]</th><th>Corresponding ALTGX instance settings</th><th>Corresponding pre-emphasis setting (mA)</th></tr>
<tr><td>00000</td><td>0</td><td>Disabled</td></tr>
<tr><td>00001</td><td>1</td><td>0.5</td></tr>
<tr><td>00101</td><td>5</td><td>1.0</td></tr>
<tr><td>01001</td><td>9</td><td>1.5</td></tr>
<tr><td>01101</td><td>13</td><td>2.0</td></tr>
<tr><td>10000</td><td>16</td><td>2.375</td></tr>
<tr><td>10001</td><td>17</td><td>2.5</td></tr>
<tr><td>10010</td><td>18</td><td>2.625</td></tr>
<tr><td>10011</td><td>19</td><td>2.75</td></tr>
<tr><td>10100</td><td>20</td><td>2.875</td></tr>
<tr><td>10101</td><td>21</td><td>3.0</td></tr>
</table>
All other values => N/A</td></tr>
<tr><td>rx_eqctrl[3..0] (1)</td><td>Input</td><td>This is an optional write control to write an equalization control value for the receive side of the PMA.
The width of this signal is fixed to 4 bits if you enable either the Use 'logical_channel_address' port for Analog controls reconfiguration option or the Use same control signal for all the channels option in the Analog controls screen. Otherwise, the width of this signal is 4 bits per channel.
<table>
<tr><th>rx_eqctrl[3..0]</th><th>Corresponding ALTGX instance settings</th></tr>
<tr><td>0001</td><td>Low</td></tr>
<tr><td>0101</td><td>Medium Low</td></tr>
<tr><td>0100</td><td>Medium High</td></tr>
<tr><td>0111</td><td>High</td></tr>
</table>
All other values => N/A</td></tr>
</table>

There are three methods that you can use to dynamically reconfigure the PMA controls of a transceiver channel:

- "Method 1: Using logical_channel_address to Reconfigure Specific Transceiver Channels" on page 3–14
- "Method 2: Writing the Same Control Signals to Control All the Transceiver Channels" on page 3–16
- "Method 3: Writing Different Control Signals for all the Transceiver Channels at the Same Time" on page 3–19

Method 1: Using logical_channel_address to Reconfigure Specific Transceiver Channels

Enable the `logical_channel_address` port by selecting the **Use 'logical_channel_address' port** option on the **Analog controls** tab. This method is applicable only for a design where the dynamic reconfiguration controller controls more than one channel.

You can additionally reconfigure either the receiver portion, transmitter portion, or both the receiver and transmitter portions of the transceiver channel by setting the corresponding value on the `rx_tx_duplex_sel` input port. For more information, refer to Table 3–2 on page 3–4.

Connecting the PMA Control Ports

The selected PMA control ports remain fixed in width, regardless of the number of channels controlled by the ALTGX_RECONFIG instance:

- `tx_vodctrl` and `tx_vodctrl_out` are fixed to 3 bits
- `tx_preemp` and `tx_preemp_out` are fixed to 5 bits
- `rx_eqdcgain` and `rx_eqdcgain_out` are fixed to 2 bits
- `rx_eqctrl` and `rx_eqctrl_out` are fixed to 4 bits

Write Transaction

To complete a write transaction, perform the following steps:

1. Set the selected PMA control ports to the desired settings (for example, `tx_vodctrl` = **3'b001**).
2. Set the `logical_channel_address` input port to the logical channel address of the transceiver channel whose PMA controls you want to reconfigure.
3. Set the `rx_tx_duplex_sel` port to **2'b10** so that only the transmit PMA controls are written to the transceiver channel.
4. Ensure that the `busy` signal is low before you start a write transaction.
5. Assert the `write_all` signal for one `reconfig_clk` clock cycle.

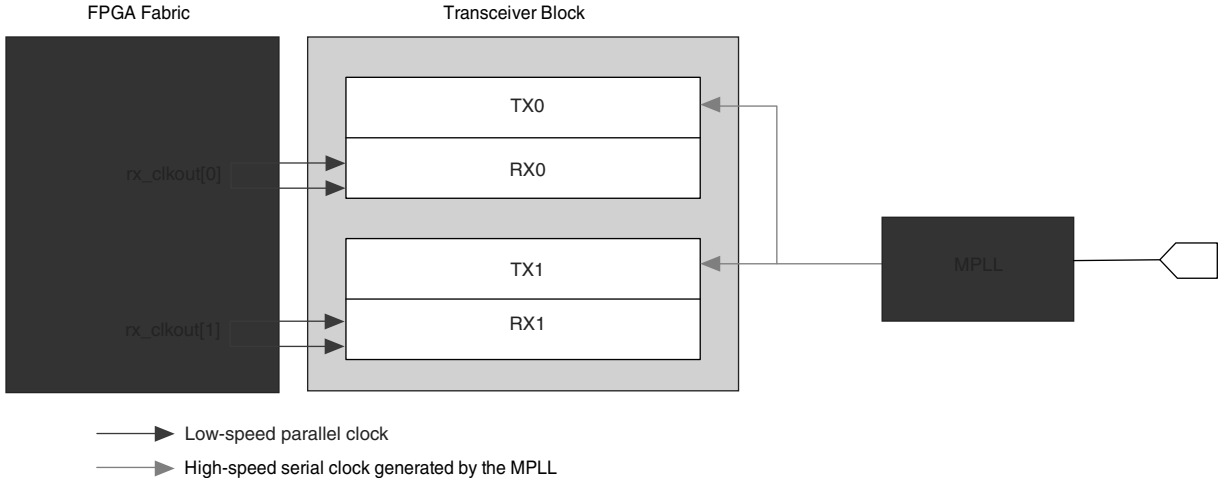
The `busy` output status signal is asserted high to indicate that the dynamic reconfiguration controller is busy writing the PMA control values. When the write transaction has completed, the `busy` signal goes low.

Option 3: Use the Respective Channel Receiver Core Clocks

- Enable this option if you want the individual channel's `rx_clkout` signal to provide the read clock to its respective Receive Phase Compensation FIFO.
- This option is typically enabled when the channel is reconfigured from a Basic or Protocol configuration with or without rate matching to another Basic or Protocol configuration with or without rate matching.

Figure 3–15 shows the respective `rx_clkout` of each channel clocking the respective receiver channels of a transceiver block.

Figure 3–15. Option 3 for Receiver Core Clocking (Channel Reconfiguration Mode)



PLL Reconfiguration Mode

Cyclone IV GX device support the PLL reconfiguration support through the ALTPLL_RECONFIG MegaWizard. You can use this mode to reconfigure the multipurpose PLL or general purpose PLL used to clock the transceiver channel without affecting the remaining blocks of the channel. When you reconfigure the multipurpose PLL or general purpose PLL of a transceiver block to run at a different data rate, all the transceiver channels listening to this multipurpose PLL or general purpose PLL also get reconfigured to the new data rate. Channel settings are not affected. When you reconfigure the multipurpose PLL or general purpose PLL to support a different data rate, you must ensure that the functional mode of the transceiver channel supports the reconfigured data rate.

The PLL reconfiguration mode can be enabled by selecting the **Enable PLL Reconfiguration** option in the ALTGX MegaWizard under **Reconfiguration Setting** tab. For multipurpose PLL or general purpose PLL reconfiguration, **.mif** files are required to dynamically reconfigure the PLL setting in order to change the output frequency of the transceiver PLL to support different data rates.

Table 1–21. Transceiver Specification for Cyclone IV GX Devices (Part 2 of 4)

Symbol/ Description	Conditions	C6			C7, I7			C8			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Receiver											
Supported I/O Standards	1.4 V PCML, 1.5 V PCML, 2.5 V PCML, LVPECL, LVDS										
Data rate (F324 and smaller package) *(15)*	—	600	—	2500	600	—	2500	600	—	2500	Mbps
Data rate (F484 and larger package) *(15)*	—	600	—	3125	600	—	3125	600	—	2500	Mbps
Absolute V_{MAX} for a receiver pin *(3)*	—	—	—	1.6	—	—	1.6	—	—	1.6	V
Operational V_{MAX} for a receiver pin	—	—	—	1.5	—	—	1.5	—	—	1.5	V
Absolute V_{MIN} for a receiver pin	—	–0.4	—	—	–0.4	—	—	–0.4	—	—	V
Peak-to-peak differential input voltage V_{ID} (diff p-p)	V_{ICM} = 0.82 V setting, Data Rate = 600 Mbps to 3.125 Gbps	0.1	—	2.7	0.1	—	2.7	0.1	—	2.7	V
V_{ICM}	V_{ICM} = 0.82 V setting	—	820 ± 10%	—	—	820 ± 10%	—	—	820 ± 10%	—	mV
Differential on-chip termination resistors	100–Ω setting	—	100	—	—	100	—	—	100	—	Ω
	150–Ω setting	—	150	—	—	150	—	—	150	—	Ω
Differential and common mode return loss	PIPE, Serial Rapid I/O SR, SATA, CPRI LV, SDI, XAUI	Compliant									—
Programmable ppm detector *(4)*	—	± 62.5, 100, 125, 200, 250, 300									ppm
Clock data recovery (CDR) ppm tolerance (without spread-spectrum clocking enabled)	—	—	—	±300 *(5)*, ±350 *(6)*, *(7)*	—	—	±300 *(5)*, ±350 *(6)*, *(7)*	—	—	±300 *(5)*, ±350 *(6)*, *(7)*	ppm
CDR ppm tolerance (with synchronous spread-spectrum clocking enabled) *(8)*	—	—	—	350 to –5350 *(7)*, *(9)*	—	—	350 to –5350 *(7)*, *(9)*	—	—	350 to –5350 *(7)*, *(9)*	ppm
Run length	—	—	80	—	—	80	—	—	80	—	UI
Programmable equalization	No Equalization	—	—	1.5	—	—	1.5	—	—	1.5	dB
	Medium Low	—	—	4.5	—	—	4.5	—	—	4.5	dB
	Medium High	—	—	5.5	—	—	5.5	—	—	5.5	dB
	High	—	—	7	—	—	7	—	—	7	dB

Table 1–47. Document Revision History

Date	Version	Changes
February 2010	1.1	■ Updated Table 1–3 through Table 1–44 to include information for Cyclone IV E devices and Cyclone IV GX devices for Quartus II software version 9.1 SP1 release. ■ Minor text edits.
November 2009	1.0	Initial release.