Intel - EP4CE15F23I7N Datasheet





Welcome to <u>E-XFL.COM</u>

Understanding <u>Embedded - FPGAs (Field</u> <u>Programmable Gate Array)</u>

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	963
Number of Logic Elements/Cells	15408
Total RAM Bits	516096
Number of I/O	343
Number of Gates	-
Voltage - Supply	1.15V ~ 1.25V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	484-BGA
Supplier Device Package	484-FBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/intel/ep4ce15f23i7n

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong



Figure 2–1 shows the LEs for Cyclone IV devices.

Figure 2–1. Cyclone IV Device LEs

LE Features

You can configure the programmable register of each LE for D, T, JK, or SR flipflop operation. Each register has data, clock, clock enable, and clear inputs. Signals that use the global clock network, general-purpose I/O pins, or any internal logic can drive the clock and clear control signals of the register. Either general-purpose I/O pins or the internal logic can drive the clock enable. For combinational functions, the LUT output bypasses the register and drives directly to the LE outputs.

Each LE has three outputs that drive the local, row, and column routing resources. The LUT or register output independently drives these three outputs. Two LE outputs drive the column or row and direct link routing connections, while one LE drives the local interconnect resources. This allows the LUT to drive one output while the register drives another output. This feature, called register packing, improves device utilization because the device can use the register and the LUT for unrelated functions. The LAB-wide synchronous load control signal is not available when using register packing. For more information about the synchronous load control signal, refer to "LAB Control Signals" on page 2–6.

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

Figure 4–2 shows the multiplier block architecture.



Figure 4–2. Multiplier Block Architecture

Input Registers

You can send each multiplier input signal into an input register or directly into the multiplier in 9- or 18-bit sections, depending on the operational mode of the multiplier. You can send each multiplier input signal 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 for 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, as well as other multipliers 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. For multiplier information, refer to "Operational Modes" on page 4–4.

Each multiplier operand is a unique signed or unsigned number. The signa and signb signals 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.

GCLK Network Clock	GCLK Networks																			
Sources	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
CLK15/DIFFCLK_6p (2)	—	—	—	—	—		—	—		_	_		_		_	~	_	_	~	—
PLL_1_C0 (3)	\checkmark	—	—	\checkmark	—	—	—	—	—	—	—	—	—	—	_	—	_	_	—	—
PLL_1_C1 (3)	—	\checkmark	—	—	\checkmark	_	—	—	—	_	_	_	_	_	_	_	_	_	_	—
PLL_1_C2 (3)	\checkmark	—	\checkmark	—	—	—	—	—	—	_	—	_	—	_	_	_			—	—
PLL_1_C3 (3)		~	_	~			_		_											—
PLL_1_C4 (3)	-	-	~	-	~		_	-											-	—
PLL_2_C0 (3)	—	—	—	—		\checkmark	—	—	\checkmark	_		_	—	_		_			—	—
PLL_2_C1 (3)	_	_	_	_		_	\checkmark	_	_	>										—
PLL_2_C2 (3)	_	_		_		\checkmark		~												—
PLL_2_C3 (3)	—	—	—	—	—	—	\checkmark	—	\checkmark	—	—	—	—	—	—	—	—	—	—	—
PLL_2_C4 (3)	—	—	—	—	—	—	—	\checkmark	—	\checkmark	—	—	—	—	—	—	—	—	—	—
PLL_3_C0	—	—	—	—	—	—	—	—	—	—	\checkmark	—	—	\checkmark	—	—	—	—	—	—
PLL_3_C1	—	—	—	—	—	—	—	—	—	—	—	\checkmark	—	—	\checkmark	—	—	—	—	—
PLL_3_C2	—	—	—	—	—	—	—	—	—	—	\checkmark	—	\checkmark	—	—	—	—	—	—	—
PLL_3_C3	—	—	—	—	—	—	—	—	—	—	—	\checkmark	—	\checkmark	—	—	—	—	—	—
PLL_3_C4	—	—	—	—	—	—	—	—	—	—	—	—	\checkmark	—	\checkmark	—	—	—	—	—
PLL_4_C0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	\checkmark	—	—	\checkmark	—
PLL_4_C1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	\checkmark	—	—	\checkmark
PLL_4_C2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	\checkmark	—	\checkmark	—	—
PLL_4_C3	—	—	—	—		—	—	—	—	_	_	_		_		_	\checkmark		\checkmark	—
PLL_4_C4	—	—	—	—		—	—	—	—	_	_	_		_		_		\checkmark	—	\checkmark
DPCLK0	\checkmark	—	—	—	—	—	—	—	—	—	—	—	—	—	_	—	—	—	—	—
DPCLK1	—	\checkmark	_	—	_	—	—	—	—	—	_	_	_	—	_	—	_	_	—	—
DPCLK7 (4)																				
CDPCLK0, Or	—	—	\checkmark	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
CDPCLK7 (2), (5)																				

 Table 5–3.
 GCLK Network Connections for Cyclone IV E Devices (1)
 (Part 2 of 3)

Figure 7–2 shows the location and numbering of the DQS, DQ, or CQ# pins in the Cyclone IV GX I/O banks.



Figure 7–2. DQS, CQ, or CQ# Pins in Cyclone IV GX I/O Banks ⁽¹⁾

Note to Figure 7–2:

(1) The DQS, CQ, or CQ# pin locations in this diagram apply to all packages in Cyclone IV GX devices except devices in 169-pin FBGA and 324-pin FBGA.





Notes to Figure 8-9:

- (1) Connect the pull-up resistors to the V_{CCIO} supply of the bank in which the pin resides.
- (2) Connect the pull-up resistor to the V_{CCIO} supply voltage of the I/O bank in which the nCE pin resides.
- (3) The nCEO pin is left unconnected or used as a user I/O pin when it does not feed the nCE pin of another device.
- (4) The MSEL pin settings vary for different configuration voltage standards and POR time. You must set the master device in AP mode and the slave devices in FPP mode. To connect MSEL [3..0] for the master device in AP mode and the slave devices in FPP mode, refer to Table 8–5 on page 8–9. Connect the MSEL pins directly to V_{CCA} or GND.
- (5) The AP configuration ignores the WAIT signal during configuration mode. However, if you are accessing flash during user mode with user logic, you can optionally use the normal I/O pin to monitor the WAIT signal from the Micron P30 or P33 flash.
- (6) Connect the repeater buffers between the Cyclone IV E master device and slave devices for DATA [15..0] and DCLK. All I/O inputs must maintain a maximum AC voltage of 4.1 V. The output resistance of the repeater buffers must fit the maximum overshoot equation outlined in "Configuration and JTAG Pin I/O Requirements" on page 8–5.

In a multi-device AP configuration, the board trace length between the parallel flash and the master device must follow the recommendations listed in Table 8–11.

- 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.
- **C** 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.

Figure 9–3 shows the error detection block diagram in FPGA devices and shows the interface that the WYSIWYG atom enables in your design.



Figure 9–3. Error Detection Block Diagram

The user logic is affected by the soft error failure, so reading out the 32-bit CRC signature through the regout should not be relied upon to detect a soft error. You should rely on the CRC_ERROR output signal itself, because this CRC_ERROR output signal cannot be affected by a soft error.

To enable the cycloneiv_crcblock WYSIWYG atom, you must name the atom for each Cyclone IV device accordingly.

Example 9–1 shows an example of how to define the input and output ports of a WYSIWYG atom in a Cyclone IV device.

Example 9–1. Error Detection Block Diagram

```
cycloneiv_crcblock<crcblock_name>
(
.clk(<clock source>),
.shiftnld(<shiftnld source>),
.ldsrc(<ldsrc source>),
.crcerror(<crcerror out destination>),
.regout(<output destination>),
);
```

Transmitter Channel Datapath

The following sections describe the Cyclone IV GX transmitter channel datapath architecture as shown in Figure 1–3:

- TX Phase Compensation FIFO
- Byte Serializer
- 8B/10B Encoder
- Serializer
- Transmitter Output Buffer

TX Phase Compensation FIFO

The TX phase compensation FIFO compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock, when interfacing the transmitter channel to the FPGA fabric (directly or through the PIPE and PCIe hard IP). The FIFO is four words deep, with latency between two to three parallel clock cycles. Figure 1–4 shows the TX phase compensation FIFO block diagram.

Figure 1–4. TX Phase Compensation FIFO Block Diagram



Note to Figure 1-4:

(1) The x refers to the supported 8-, 10-, 16-, or 20-bits transceiver channel width.

- The FIFO can operate in registered mode, contributing to only one parallel clock cycle of latency in Deterministic Latency functional mode. For more information, refer to "Deterministic Latency Mode" on page 1–73.
- **To** For more information about FIFO clocking, refer to "FPGA Fabric-Transceiver Interface Clocking" on page 1–43.

Byte Serializer

The byte serializer divides the input datapath width by two to allow transmitter channel operation at higher data rates while meeting the maximum FPGA fabric frequency limit. This module is required in configurations that exceed the maximum FPGA fabric-transceiver interface clock frequency limit and optional in configurations that do not.

To For the FPGA fabric-transceiver interface frequency specifications, refer to the *Cyclone IV Device Data Sheet*.

at time n + 2 is encoded as a positive disparity code group. In the same example, the current running disparity at time n + 5 indicates that the K28.5 in time n + 6 should be encoded with a positive disparity. Because tx_forcedisp is high at time n + 6, and tx_dispval is high, the K28.5 at time n + 6 is encoded as a negative disparity code group.

Miscellaneous Transmitter PCS Features

The transmitter PCS supports the following additional features:

Polarity inversion—corrects accidentally swapped positive and negative signals from the serial differential link during board layout by inverting the polarity of each bit. An optional tx_invpolarity port is available to dynamically invert the polarity of every bit of the 8-bit or 10-bit input data to the serializer in the transmitter datapath. Figure 1–9 shows the transmitter polarity inversion feature.

Figure 1–9. Transmitter Polarity Inversion



tx_invpolarity is a dynamic signal and might cause initial disparity errors at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

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.





Table 1-3. Word Aligner Modes

Modes	PMA-PCS Interface Widths	Allowed Word Alignment Pattern Lengths
Manual Alignment	8-bit	16 bits
Manual Algument	10-bit	7 or 10 bits
Rit Clin	8-bit	16 bits
Bit-Silp	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 alignment pattern maintains the current word boundary even when it receives the word alignment pattern in a new word boundary.

After updating the word boundary, word aligner status signals (rx_syncstatus and rx_patterndetect) are driven high for one parallel clock cycle synchronous to the most significant byte of the word alignment pattern. The rx_syncstatus and rx_patterndetect signals have the same latency as the datapath and are forwarded to the FPGA fabric to indicate the word aligner status. Any word alignment pattern received thereafter in the same word boundary causes only the rx_patterndetect signal to go high for one clock cycle.

Figure 1–17 shows the manual alignment mode word aligner operation in 10-bit data width mode. In this example, a /K28.5/ (10'b0101111100) is specified as the word alignment pattern.

The word aligner aligns to the /K28.5/ alignment pattern (red) in cycle *n* because the rx_enapatternalign signal is asserted high. The rx_syncstatus signal goes high for one clock cycle indicating alignment to a new word boundary. The rx_patterndetect signal also goes high for one clock cycle to indicate initial word alignment.

At time n + 1, the rx_enapatternalign signal is deasserted to instruct the word aligner to lock the current word boundary.

The alignment pattern is detected again (green) in a new word boundary across cycles n + 2 and n + 3. The word aligner does not align to this new word boundary because the rx_enapatternalign signal is held low.

The /K28.5/ word alignment pattern is detected again (blue) in the current word boundary during cycle n + 5 causing the rx_patterndetect signal to go high for one parallel clock cycle.



Figure 1–17. Word Aligner in 10-bit Manual Alignment Mode

If the word alignment pattern is known to be unique and does not appear between word boundaries, you can hold the rx_enapatternalign signal constantly high because there is no possibility of false word alignment. If there is a possibility of the word alignment pattern occurring across word boundaries, you must control the rx_enapatternalign signal to lock the word boundary after the desired word alignment is achieved to avoid re-alignment to an incorrect word boundary.

Rate Match FIFO

In asynchronous systems, the upstream transmitter and local receiver can be clocked with independent reference clocks. Frequency differences in the order of a few hundred ppm can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference clock) to the local receiver reference clock domain. Figure 1–21 shows the rate match FIFO block diagram.

Figure 1–21. Rate Match FIFO Block Diagram



The rate match FIFO compensates for small clock frequency differences of up to ± 300 ppm (600 ppm total) between the upstream transmitter and the local receiver clocks by performing the following functions:

- Insert skip symbols when the local receiver reference clock frequency is greater than the upstream transmitter reference clock frequency
- Delete skip symbols when the local receiver reference clock frequency is less than the upstream transmitter reference clock frequency

The 20-word deep rate match FIFO and logics control insertion and deletion of skip symbols, depending on the ppm difference. The operation begins after the word aligner synchronization status (rx_syncstatus) is asserted.

P

Rate match FIFO is only supported with 8B/10B encoded data and the word aligner in automatic synchronization state machine mode.

8B/10B Decoder

The 8B/10B decoder receives 10-bit data and decodes it into an 8-bit data and a 1-bit control identifier. The decoder is compliant with Clause 36 of the IEEE 802.3 specification.

Figure 1–22 shows the 8B/10B decoder block diagram.

Figure 1–22. 8B/10B Decoder Block Diagram



When the byte serializer is enabled, the low-speed clock frequency is halved before feeding into the read clock of TX phase compensation FIFO. The low-speed clock is available in the FPGA fabric as tx_clkout port, which can be used in the FPGA fabric to send transmitter data and control signals.

Figure 1-33. Transmitter Only Datapath Clocking in Non-Bonded Channel Configuration



Figure 1–34 shows the datapath clocking in receiver only operation. In this mode, the receiver PCS supports configuration without the rate match FIFO. The CDR unit in the channel recovers the clock from the received serial data and generates the high-speed recovered clock for the deserializer, and low-speed recovered clock for forwarding to the receiver PCS. The low-speed recovered clock feeds to the following blocks in the receiver PCS:

- word aligner
- 8B/10B decoder
- write clock of byte deserializer
- byte ordering
- write clock of RX phase compensation FIFO

When the byte deserializer is enabled, the low-speed recovered clock frequency is halved before feeding into the write clock of the RX phase compensation FIFO. The low-speed recovered clock is available in the FPGA fabric as rx_clkout port, which can be used in the FPGA fabric to capture receiver data and status signals.

Figure 1-34. Receiver Only Datapath Clocking without Rate Match FIFO in Non-Bonded Channel Configuration



Note to Figure 1-34:

(1) High-speed recovered clock.

When the transceiver is configured for transmitter and receiver operation in non-bonded channel configuration, the receiver PCS supports configuration with and without the rate match FIFO. The difference is only at the receiver datapath clocking. The transmitter datapath clocking is identical to transmitter only operation mode as shown in Figure 1–33.

Receiver Spread Spectrum Clocking

Asynchronous SSC is not supported in Cyclone IV devices. You can implement only synchronous SSC for SATA, V-by-One, and Display Port protocols in Basic mode.

PCI Express (PIPE) Mode

PIPE mode provides the transceiver channel datapath configuration that supports ×1, ×2, and ×4 initial lane width for PCIe Gen1 signaling rate with PIPE interface implementation. The Cyclone IV GX transceiver provides following features in PIPE mode:

- PIPE interface
- receiver detection circuitry
- electrical idle control
- signal detect at receiver
- lane synchronization with compliant state machine
- clock rate compensation with rate match FIFO
- Low-Latency Synchronous PCIe
- fast recovery from P0s state
- electrical idle inference
- compliance pattern transmission
- reset requirement

Figure 1–48 shows the transceiver channel datapath and clocking when configured in PIPE mode with ×1 channel configuration.

Figure 1–48. Transceiver Channel Datapath and Clocking when Configured in PIPE Mode with ×1 Channel Configuration



Notes to Figure 1-48:

- (1) Low-speed recovered clock.
- (2) High-speed recovered clock.

Block	Port Name	Input/ Output	Clock Domain	Description			
	rx_coreclk	Output Clock signal		Optional read clock port for the RX phase compensation FIFO.			
RX PCS	rx_phase_comp_fifo _error	Output	Synchronous to tx_clkout (non-bonded modes) or coreclkout (bonded modes)	RX phase compensation FIFO full or empty indicator.A high level indicates FIFO is either full or empty.			
	rx_bitslipboundarys electout	Output	Asynchronous signal.	Indicate the number of bits slipped in the word aligner configured in manual alignment mode.			
	rx datain	Input	N/A	Receiver serial data input port.			
	rx_freqlocked	Output	Asynchronous signal	 Receiver CDR lock state indicator A high level indicates the CDR is in LTD state. A low level indicates the CDR is in LTR state. 			
RX PMA	rx_locktodata	Input	Asynchronous signal	 Receiver CDR LTD state control signal A high level forces the CDR to LTD state When deasserted, the receiver CDR lock state depends on the rx_locktorefclk signal level. 			
	rx_locktorefclk Inpu		Asynchronous signal	 Receiver CDR LTR state control signal. The rx_locktorefclk and rx_locktodata signals control whether the receiver CDR states as follows: [rx_locktodata:rx_locktorefclk] 2'b00—receiver CDR is in automatic lock mode 2b'01—receiver CDR is in manual lock mode (LTR state) 2b'1x—receiver CDR is in manual lock mode (LTD state) 			
	rx_signaldetect	rx_signaldetect Output Asynchronous signal		 Signal threshold detect indicator. Available in Basic mode when 8B/10B encoder/decoder is used, and in PIPE mode. A high level indicates that the signal present at the receiver input buffer is above the programmed signal detection threshold value. 			
	rx_recovclkout Outpu		Clock signal	 CDR low-speed recovered clock Only available in the GIGE mode for applications such as Synchronous Ethernet. 			

Table 1-27	. Receiver Ports	n ALTGX Megafunction	for Cyclone IV GX	(Part 3 of 3)
------------	------------------	----------------------	-------------------	---------------

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.



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





Notes to Figure 3-16:

- (1) $\langle n \rangle =$ (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.

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

• For more information about the supported maximum clock rate, device and pin planning, IP implementation, and device termination, refer to *Section III: System Performance Specifications* of the *External Memory Interface Handbook*.

Table 1-37 lists the memory output clock jitter specifications for Cyclone IV devices.

Table 1–37. Memory Output Clock Jitter Specifications for Cyclone IV Devices (1), (2)

Parameter	Symbol	Min	Max	Unit
Clock period jitter	t _{JIT(per)}	-125	125	ps
Cycle-to-cycle period jitter	t _{JIT(cc)}	-200	200	ps
Duty cycle jitter	t _{JIT(duty)}	-150	150	ps

Notes to Table 1-37:

- (1) Memory output clock jitter measurements are for 200 consecutive clock cycles, as specified in the JEDEC DDR2 standard.
- (2) The clock jitter specification applies to memory output clock pins generated using DDIO circuits clocked by a PLL output routed on a global clock (GCLK) network.

Duty Cycle Distortion Specifications

Table 1–38 lists the worst case duty cycle distortion for Cyclone IV devices.

Table 1–38. Duty Cycle Distortion on Cyclone IV Devices I/O Pins (1), (2), (3)

Symbol	C	6	C7	, 17	C8, I8	BL, A7	C9L		llait	
Symbol	Min	Max	Min	Max	Min	Max	Min	Max	UIII	
Output Duty Cycle	45	55	45	55	45	55	45	55	%	

Notes to Table 1-38:

(1) The duty cycle distortion specification applies to clock outputs from the PLLs, global clock tree, and IOE driving the dedicated and general purpose I/O pins.

(2) Cyclone IV devices meet the specified duty cycle distortion at the maximum output toggle rate for each combination of I/O standard and current strength.

(3) Cyclone IV E 1.0 V core voltage devices only support C8L, C9L, and I8L speed grades. Cyclone IV E 1.2 V core voltage devices only support C6, C7, C8, I7, and A7 speed grades. Cyclone IV GX devices only support C6, C7, C8, and I7 speed grades.

OCT Calibration Timing Specification

Table 1–39 lists the duration of calibration for series OCT with calibration at device power-up for Cyclone IV devices.

Table 1–39. Timing Specification for Series OCT with Calibration at Device Power-Up for Cyclone IV Devices $^{(1)}$

Symbol	Description	Maximum	Units
t _{octcal}	Duration of series OCT with calibration at device power-up	20	μs

Note to Table 1-39:

(1) OCT calibration takes place after device configuration and before entering user mode.

I/O Timing

Use the following methods to determine I/O timing:

- the Excel-based I/O Timing
- the Quartus II timing analyzer

The Excel-based I/O timing provides pin timing performance for each device density and speed grade. The data is typically used prior to designing the FPGA to get a timing budget estimation as part of the link timing analysis. The Quartus II timing analyzer provides a more accurate and precise I/O timing data based on the specifics of the design after place-and-route is complete.

The Excel-based I/O Timing spreadsheet is downloadable from Cyclone IV Devices Literature website.

Glossary

Table 1–46 lists the glossary for this chapter.

Letter	Term	Definitions
Α	—	—
В	—	_
C	—	_
D	—	_
E	—	_
F	f _{HSCLK}	High-speed I/O block: High-speed receiver/transmitter input and output clock frequency.
C	GCLK	Input pin directly to Global Clock network.
u	GCLK PLL	Input pin to Global Clock network through the PLL.
Н	HSIODR	High-speed I/O block: Maximum/minimum LVDS data transfer rate (HSIODR = 1/TUI).
I	Input Waveforms for the SSTL Differential I/O Standard	Vswing Vswing V _{REF} V _{IL}

Table 1-46. Glossary (Part 1 of 5)