



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	216
Number of Logic Elements/Cells	-
Total RAM Bits	-
Number of I/O	102
Number of Gates	-
Voltage - Supply	3V ~ 3.6V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 70°C (TA)
Package / Case	144-LQFP
Supplier Device Package	144-TQFP (20x20)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=epf10k30atc144-2



For more information, see the following documents:

- *Configuration Devices for APEX & FLEX Devices Data Sheet*
- *BitBlaster Serial Download Cable Data Sheet*
- *ByteBlasterMV Parallel Port Download Cable Data Sheet*
- *Application Note 116 (Configuring APEX 20K, FLEX 10K & FLEX 6000 Devices)*

FLEX 10K devices are supported by Altera development systems; single, integrated packages that offer schematic, text (including AHDL), and waveform design entry, compilation and logic synthesis, full simulation and worst-case timing analysis, and device configuration. The Altera software provides EDIF 2.0 and 3.0, LPM, VHDL, Verilog HDL, and other interfaces for additional design entry and simulation support from other industry-standard PC- and UNIX workstation-based EDA tools.

The Altera software works easily with common gate array EDA tools for synthesis and simulation. For example, the Altera software can generate Verilog HDL files for simulation with tools such as Cadence Verilog-XL. Additionally, the Altera software contains EDA libraries that use device-specific features such as carry chains which are used for fast counter and arithmetic functions. For instance, the Synopsys Design Compiler library supplied with the Altera development systems include DesignWare functions that are optimized for the FLEX 10K architecture.

The Altera development systems run on Windows-based PCs and Sun SPARCstation, and HP 9000 Series 700/800 workstations.



See the *MAX+PLUS II Programmable Logic Development System & Software Data Sheet* for more information.

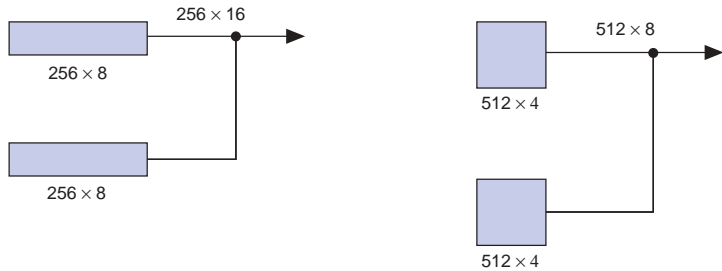
Functional Description

Each FLEX 10K device contains an embedded array to implement memory and specialized logic functions, and a logic array to implement general logic.

The embedded array consists of a series of EABs. When implementing memory functions, each EAB provides 2,048 bits, which can be used to create RAM, ROM, dual-port RAM, or first-in first-out (FIFO) functions. When implementing logic, each EAB can contribute 100 to 600 gates towards complex logic functions, such as multipliers, microcontrollers, state machines, and DSP functions. EABs can be used independently, or multiple EABs can be combined to implement larger functions.

Larger blocks of RAM are created by combining multiple EABs. For example, two 256×8 RAM blocks can be combined to form a 256×16 RAM block; two 512×4 blocks of RAM can be combined to form a 512×8 RAM block. See [Figure 3](#).

Figure 3. Examples of Combining EABs



If necessary, all EABs in a device can be cascaded to form a single RAM block. EABs can be cascaded to form RAM blocks of up to 2,048 words without impacting timing. Altera's software automatically combines EABs to meet a designer's RAM specifications.

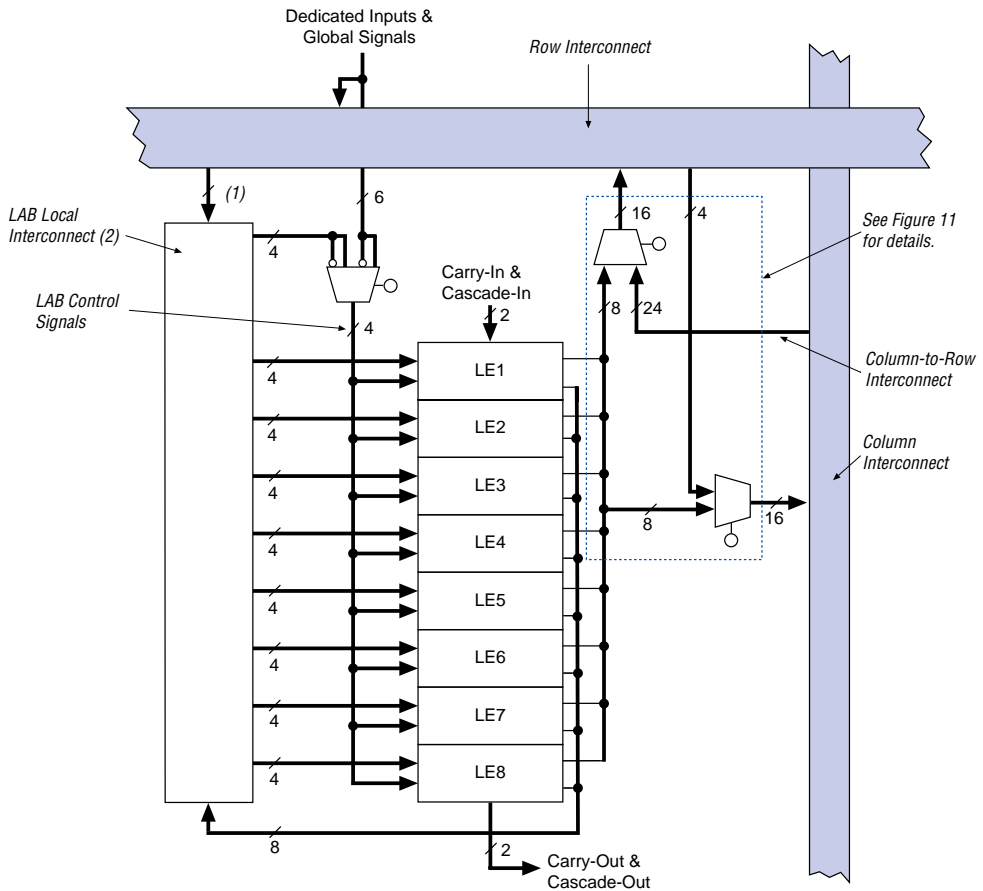
EABs provide flexible options for driving and controlling clock signals. Different clocks can be used for the EAB inputs and outputs. Registers can be independently inserted on the data input, EAB output, or the address and WE inputs. The global signals and the EAB local interconnect can drive the WE signal. The global signals, dedicated clock pins, and EAB local interconnect can drive the EAB clock signals. Because the LEs drive the EAB local interconnect, the LEs can control the WE signal or the EAB clock signals.

Each EAB is fed by a row interconnect and can drive out to row and column interconnects. Each EAB output can drive up to two row channels and up to two column channels; the unused row channel can be driven by other LEs. This feature increases the routing resources available for EAB outputs. See [Figure 4](#).

Logic Array Block

Each LAB consists of eight LEs, their associated carry and cascade chains, LAB control signals, and the LAB local interconnect. The LAB provides the coarse-grained structure to the FLEX 10K architecture, facilitating efficient routing with optimum device utilization and high performance. See [Figure 5](#).

Figure 5. FLEX 10K LAB



Notes:

- (1) EPF10K10, EPF10K10A, EPF10K20, EPF10K30, EPF10K30A, EPF10K40, EPF10K50, and EPF10K50V devices have 22 inputs to the LAB local interconnect channel from the row; EPF10K70, EPF10K100, EPF10K100A, EPF10K130V, and EPF10K250A devices have 26.
- (2) EPF10K10, EPF10K10A, EPF10K20, EPF10K30, EPF10K30A, EPF10K40, EPF10K50, and EPF10K50V devices have 30 LAB local interconnect channels; EPF10K70, EPF10K100, EPF10K100A, EPF10K130V, and EPF10K250A devices have 34 LABs.

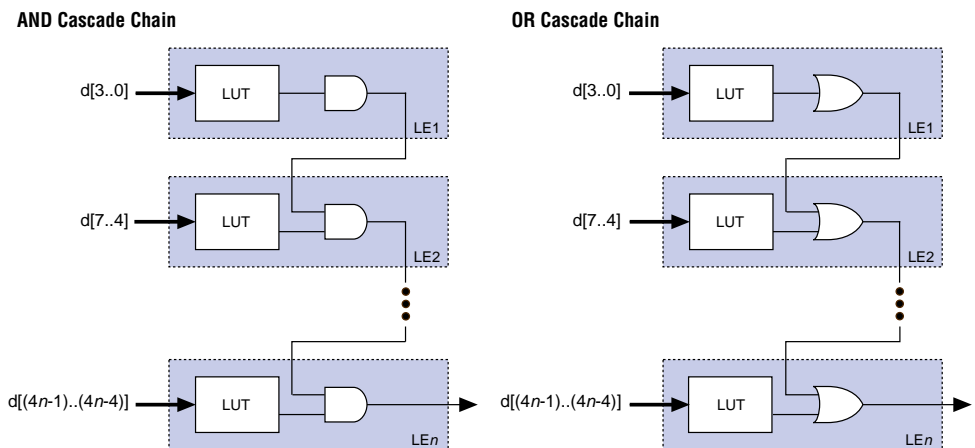
Cascade Chain

With the cascade chain, the FLEX 10K architecture can implement functions that have a very wide fan-in. Adjacent LUTs can be used to compute portions of the function in parallel; the cascade chain serially connects the intermediate values. The cascade chain can use a logical AND or logical OR (via De Morgan's inversion) to connect the outputs of adjacent LEs. Each additional LE provides four more inputs to the effective width of a function, with a delay as low as 0.7 ns per LE. Cascade chain logic can be created automatically by the Compiler during design processing, or manually by the designer during design entry.

Cascade chains longer than eight bits are implemented automatically by linking several LABs together. For easier routing, a long cascade chain skips every other LAB in a row. A cascade chain longer than one LAB skips either from even-numbered LAB to even-numbered LAB, or from odd-numbered LAB to odd-numbered LAB (e.g., the last LE of the first LAB in a row cascades to the first LE of the third LAB). The cascade chain does not cross the center of the row (e.g., in the EPF10K50 device, the cascade chain stops at the eighteenth LAB and a new one begins at the nineteenth LAB). This break is due to the EAB's placement in the middle of the row.

Figure 8 shows how the cascade function can connect adjacent LEs to form functions with a wide fan-in. These examples show functions of $4n$ variables implemented with n LEs. The LE delay is as low as 1.6 ns; the cascade chain delay is as low as 0.7 ns. With the cascade chain, 3.7 ns is needed to decode a 16-bit address.

Figure 8. Cascade Chain Operation



Normal Mode

The normal mode is suitable for general logic applications and wide decoding functions that can take advantage of a cascade chain. In normal mode, four data inputs from the LAB local interconnect and the carry-in are inputs to a four-input LUT. The Compiler automatically selects the carry-in or the DATA3 signal as one of the inputs to the LUT. The LUT output can be combined with the cascade-in signal to form a cascade chain through the cascade-out signal. Either the register or the LUT can be used to drive both the local interconnect and the FastTrack Interconnect at the same time.

The LUT and the register in the LE can be used independently; this feature is known as register packing. To support register packing, the LE has two outputs; one drives the local interconnect and the other drives the FastTrack Interconnect. The DATA4 signal can drive the register directly, allowing the LUT to compute a function that is independent of the registered signal; a three-input function can be computed in the LUT, and a fourth independent signal can be registered. Alternatively, a four-input function can be generated, and one of the inputs to this function can be used to drive the register. The register in a packed LE can still use the clock enable, clear, and preset signals in the LE. In a packed LE, the register can drive the FastTrack Interconnect while the LUT drives the local interconnect, or vice versa.

Arithmetic Mode

The arithmetic mode offers 2 three-input LUTs that are ideal for implementing adders, accumulators, and comparators. One LUT computes a three-input function, and the other generates a carry output. As shown in [Figure 9](#) on page 19, the first LUT uses the carry-in signal and two data inputs from the LAB local interconnect to generate a combinatorial or registered output. For example, in an adder, this output is the sum of three signals: a, b, and carry-in. The second LUT uses the same three signals to generate a carry-out signal, thereby creating a carry chain. The arithmetic mode also supports simultaneous use of the cascade chain.

FastTrack Interconnect

In the FLEX 10K architecture, connections between LEs and device I/O pins are provided by the FastTrack Interconnect, which is a series of continuous horizontal and vertical routing channels that traverse the device. This global routing structure provides predictable performance, even in complex designs. In contrast, the segmented routing in FPGAs requires switch matrices to connect a variable number of routing paths, increasing the delays between logic resources and reducing performance.

The FastTrack Interconnect consists of row and column interconnect channels that span the entire device. Each row of LABs is served by a dedicated row interconnect. The row interconnect can drive I/O pins and feed other LABs in the device. The column interconnect routes signals between rows and can drive I/O pins.

A row channel can be driven by an LE or by one of three column channels. These four signals feed dual 4-to-1 multiplexers that connect to two specific row channels. These multiplexers, which are connected to each LE, allow column channels to drive row channels even when all eight LEs in an LAB drive the row interconnect.

Each column of LABs is served by a dedicated column interconnect. The column interconnect can then drive I/O pins or another row's interconnect to route the signals to other LABs in the device. A signal from the column interconnect, which can be either the output of an LE or an input from an I/O pin, must be routed to the row interconnect before it can enter an LAB or EAB. Each row channel that is driven by an IOE or EAB can drive one specific column channel.

Access to row and column channels can be switched between LEs in adjacent pairs of LABs. For example, an LE in one LAB can drive the row and column channels normally driven by a particular LE in the adjacent LAB in the same row, and vice versa. This routing flexibility enables routing resources to be used more efficiently. See [Figure 11](#).

I/O Element

An I/O element (IOE) contains a bidirectional I/O buffer and a register that can be used either as an input register for external data that requires a fast setup time, or as an output register for data that requires fast clock-to-output performance. In some cases, using an LE register for an input register will result in a faster setup time than using an IOE register. IOEs can be used as input, output, or bidirectional pins. For bidirectional registered I/O implementation, the output register should be in the IOE and, the data input and output enable register should be LE registers placed adjacent to the bidirectional pin. The Compiler uses the programmable inversion option to invert signals from the row and column interconnect automatically where appropriate. [Figure 13](#) shows the bidirectional I/O registers.

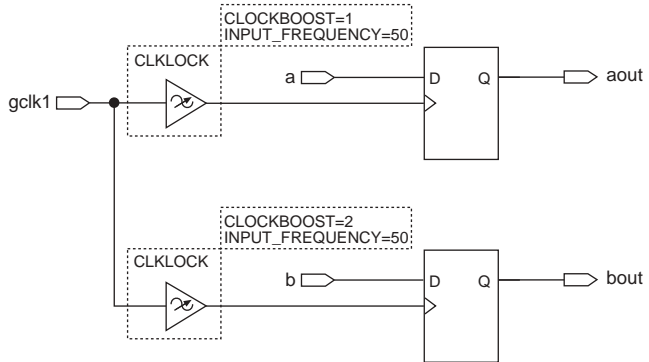
Each IOE selects the clock, clear, clock enable, and output enable controls from a network of I/O control signals called the peripheral control bus. The peripheral control bus uses high-speed drivers to minimize signal skew across devices; it provides up to 12 peripheral control signals that can be allocated as follows:

- Up to eight output enable signals
- Up to six clock enable signals
- Up to two clock signals
- Up to two clear signals

If more than six clock enable or eight output enable signals are required, each IOE on the device can be controlled by clock enable and output enable signals driven by specific LEs. In addition to the two clock signals available on the peripheral control bus, each IOE can use one of two dedicated clock pins. Each peripheral control signal can be driven by any of the dedicated input pins or the first LE of each LAB in a particular row. In addition, an LE in a different row can drive a column interconnect, which causes a row interconnect to drive the peripheral control signal. The chip-wide reset signal will reset all IOE registers, overriding any other control signals.

Tables 8 and 9 list the sources for each peripheral control signal, and the rows that can drive global signals. These tables also show how the output enable, clock enable, clock, and clear signals share 12 peripheral control signals.

Figure 17. Enabling ClockLock & ClockBoost in the Same Design



To use both the ClockLock and ClockBoost circuits in the same design, designers must use Revision C EPF10K100GC503-3DX devices and MAX+PLUS II software versions 7.2 or higher. The die revision is indicated by the third digit of the nine-digit code on the top side of the device.

Output Configuration

This section discusses the peripheral component interconnect (PCI) pull-up clamping diode option, slew-rate control, open-drain output option, MultiVolt I/O interface, and power sequencing for FLEX 10K devices. The PCI pull-up clamping diode, slew-rate control, and open-drain output options are controlled pin-by-pin via Altera logic options. The MultiVolt I/O interface is controlled by connecting V_{CCIO} to a different voltage than V_{CCINT} . Its effect can be simulated in the Altera software via the **Global Project Device Options** dialog box (Assign menu).

PCI Clamping Diodes

The EPF10K10A and EPF10K30A devices have a pull-up clamping diode on every I/O, dedicated input, and dedicated clock pin. PCI clamping diodes clamp the transient overshoot caused by reflected waves to the V_{CCIO} value and are required for 3.3-V PCI compliance. Clamping diodes can also be used to limit overshoot in other systems.

Clamping diodes are controlled on a pin-by-pin basis via a logic option in the Altera software. When V_{CCIO} is 3.3 V, a pin that has the clamping diode turned on can be driven by a 2.5-V or 3.3-V signal, but not a 5.0-V signal. When V_{CCIO} is 2.5 V, a pin that has the clamping diode turned on can be driven by a 2.5-V signal, but not a 3.3-V or 5.0-V signal. However, a clamping diode can be turned on for a subset of pins, which allows devices to bridge between a 3.3-V PCI bus and a 5.0-V device.

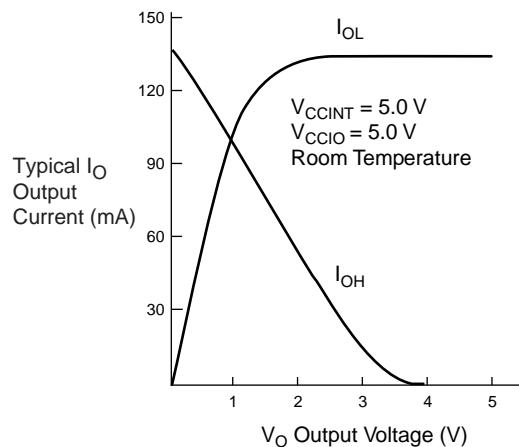
Notes to tables:

- (1) See the *Operating Requirements for Altera Devices Data Sheet*.
- (2) Minimum DC input voltage is -0.5 V . During transitions, the inputs may undershoot to -2.0 V for input currents less than 100 mA and periods shorter than 20 ns .
- (3) Numbers in parentheses are for industrial-temperature-range devices.
- (4) Maximum V_{CC} rise time is 100 ms . V_{CC} must rise monotonically.
- (5) Typical values are for $T_A = 25^\circ\text{ C}$ and $V_{CC} = 5.0\text{ V}$.
- (6) These values are specified under the Recommended Operation Condition shown in [Table 18](#) on page 45.
- (7) The I_{OH} parameter refers to high-level TTL or CMOS output current.
- (8) The I_{OL} parameter refers to low-level TTL or CMOS output current. This parameter applies to open-drain pins as well as output pins.
- (9) This value is specified for normal device operation. The value may vary during power-up.
- (10) Capacitance is sample-tested only.

Figure 20 shows the typical output drive characteristics of FLEX 10K devices with 5.0-V and 3.3-V V_{CCIO} . The output driver is compliant with the 5.0-V *PCI Local Bus Specification, Revision 2.2* (for 5.0-V V_{CCIO}).

Figure 20. Output Drive Characteristics of FLEX 10K Devices

5.0-V



3.3-V

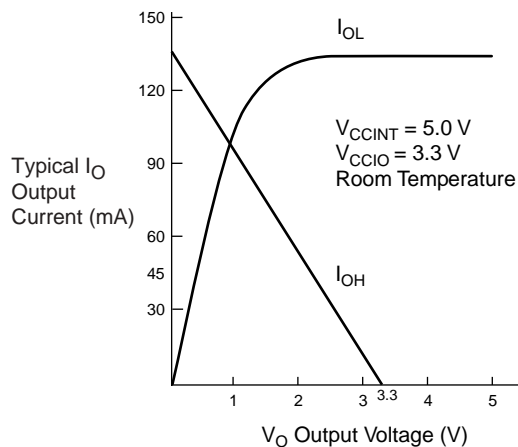
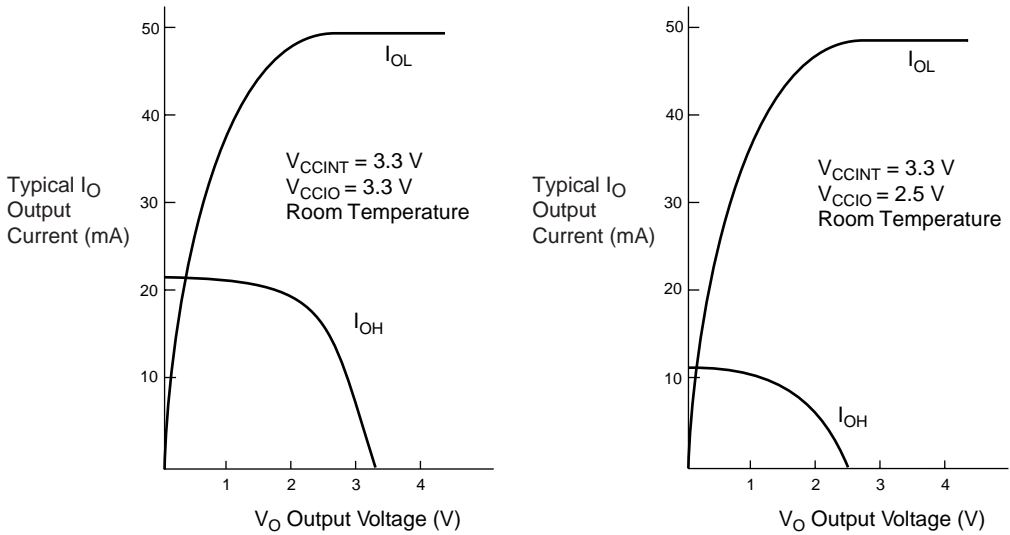


Table 28. FLEX 10KA 3.3-V Device DC Operating Conditions *Notes (6), (7)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit	
V_{IH}	High-level input voltage		1.7 or $0.5 \times V_{CCINT}$, whichever is lower		5.75	V	
V_{IL}	Low-level input voltage		-0.5		$0.3 \times V_{CCINT}$	V	
V_{OH}	3.3-V high-level TTL output voltage	$I_{OH} = -11$ mA DC, $V_{CCIO} = 3.00$ V (8)	2.4			V	
	3.3-V high-level CMOS output voltage	$I_{OH} = -0.1$ mA DC, $V_{CCIO} = 3.00$ V (8)	$V_{CCIO} - 0.2$			V	
	3.3-V high-level PCI output voltage	$I_{OH} = -0.5$ mA DC, $V_{CCIO} = 3.00$ to 3.60 V (8)	$0.9 \times V_{CCIO}$			V	
	2.5-V high-level output voltage	$I_{OH} = -0.1$ mA DC, $V_{CCIO} = 2.30$ V (8)	2.1			V	
		$I_{OH} = -1$ mA DC, $V_{CCIO} = 2.30$ V (8)	2.0			V	
		$I_{OH} = -2$ mA DC, $V_{CCIO} = 2.30$ V (8)	1.7			V	
V_{OL}	3.3-V low-level TTL output voltage	$I_{OL} = 9$ mA DC, $V_{CCIO} = 3.00$ V (9)			0.45	V	
	3.3-V low-level CMOS output voltage	$I_{OL} = 0.1$ mA DC, $V_{CCIO} = 3.00$ V (9)			0.2	V	
	3.3-V low-level PCI output voltage	$I_{OL} = 1.5$ mA DC, $V_{CCIO} = 3.00$ to 3.60 V (9)			$0.1 \times V_{CCIO}$	V	
	2.5-V low-level output voltage	$I_{OL} = 0.1$ mA DC, $V_{CCIO} = 2.30$ V (9)				0.2	V
		$I_{OL} = 1$ mA DC, $V_{CCIO} = 2.30$ V (9)				0.4	V
		$I_{OL} = 2$ mA DC, $V_{CCIO} = 2.30$ V (9)				0.7	V
I_I	Input pin leakage current	$V_I = 5.3$ V to -0.3 V (10)	-10		10	μ A	
I_{OZ}	Tri-stated I/O pin leakage current	$V_O = 5.3$ V to -0.3 V (10)	-10		10	μ A	
I_{CC0}	V_{CC} supply current (standby)	$V_I =$ ground, no load		0.3	10	mA	
		$V_I =$ ground, no load (11)		10		mA	

Figure 23. Output Drive Characteristics for EPF10K250A Device



Timing Model

The continuous, high-performance FastTrack Interconnect routing resources ensure predictable performance and accurate simulation and timing analysis. This predictable performance contrasts with that of FPGAs, which use a segmented connection scheme and therefore have unpredictable performance.

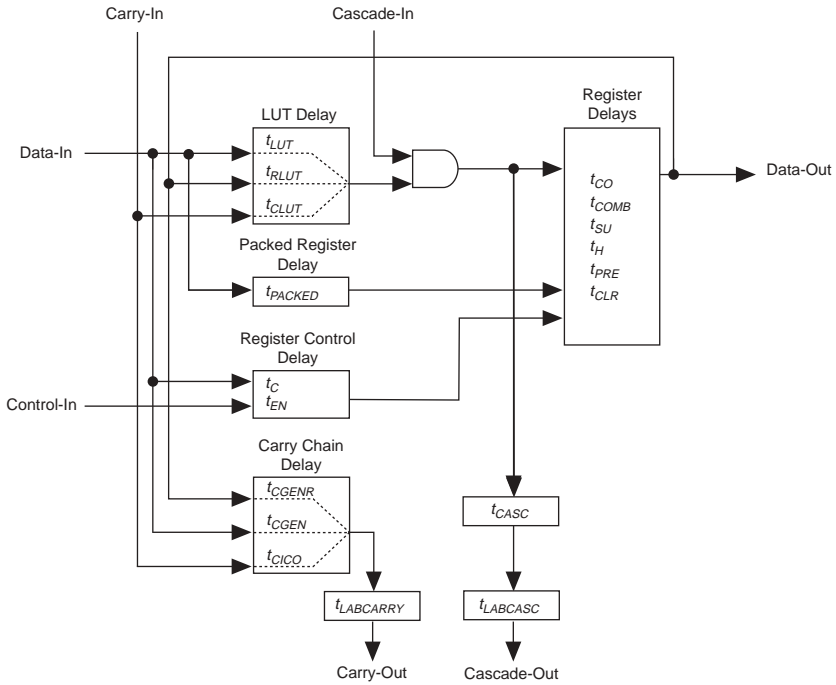
Device performance can be estimated by following the signal path from a source, through the interconnect, to the destination. For example, the registered performance between two LEs on the same row can be calculated by adding the following parameters:

- LE register clock-to-output delay (t_{CO})
- Interconnect delay ($t_{SAMEROW}$)
- LE look-up table delay (t_{LUT})
- LE register setup time (t_{SU})

The routing delay depends on the placement of the source and destination LEs. A more complex registered path may involve multiple combinatorial LEs between the source and destination LEs.

Figures 25 through 27 show the delays that correspond to various paths and functions within the LE, IOE, and EAB timing models.

Figure 25. FLEX 10K Device LE Timing Model



Tables 39 through 47 show EPF10K10 and EPF10K20 device internal and external timing parameters.

Table 39. EPF10K10 & EPF10K20 Device LE Timing Microparameters *Note (1)*

Symbol	-3 Speed Grade		-4 Speed Grade		Unit
	Min	Max	Min	Max	
t_{LUT}		1.4		1.7	ns
t_{CLUT}		0.6		0.7	ns
t_{RLUT}		1.5		1.9	ns
t_{PACKED}		0.6		0.9	ns
t_{EN}		1.0		1.2	ns
t_{CICO}		0.2		0.3	ns
t_{CGEN}		0.9		1.2	ns
t_{CGENR}		0.9		1.2	ns
t_{CASC}		0.8		0.9	ns
t_C		1.3		1.5	ns
t_{CO}		0.9		1.1	ns
t_{COMB}		0.5		0.6	ns
t_{SU}	1.3		2.5		ns
t_H	1.4		1.6		ns
t_{PRE}		1.0		1.2	ns
t_{CLR}		1.0		1.2	ns
t_{CH}	4.0		4.0		ns
t_{CL}	4.0		4.0		ns

Table 52. EPF10K30 Device Interconnect Timing Microparameters *Note (1)*

Symbol	-3 Speed Grade		-4 Speed Grade		Unit
	Min	Max	Min	Max	
$t_{DIN2IOE}$		6.9		8.7	ns
t_{DIN2LE}		3.6		4.8	ns
$t_{DIN2DATA}$		5.5		7.2	ns
$t_{DCLK2IOE}$		4.6		6.2	ns
$t_{DCLK2LE}$		3.6		4.8	ns
$t_{SAMELAB}$		0.3		0.3	ns
$t_{SAMEROW}$		3.3		3.7	ns
$t_{SAMECOLUMN}$		2.5		2.7	ns
$t_{DIFFROW}$		5.8		6.4	ns
$t_{TWOROWS}$		9.1		10.1	ns
$t_{LEPERIPH}$		6.2		7.1	ns
$t_{LABCARRY}$		0.4		0.6	ns
$t_{LABCASC}$		2.4		3.0	ns

Table 53. EPF10K40 Device Interconnect Timing Microparameters *Note (1)*

Symbol	-3 Speed Grade		-4 Speed Grade		Unit
	Min	Max	Min	Max	
$t_{DIN2IOE}$		7.6		9.4	ns
t_{DIN2LE}		3.6		4.8	ns
$t_{DIN2DATA}$		5.5		7.2	ns
$t_{DCLK2IOE}$		4.6		6.2	ns
$t_{DCLK2LE}$		3.6		4.8	ns
$t_{SAMELAB}$		0.3		0.3	ns
$t_{SAMEROW}$		3.3		3.7	ns
$t_{SAMECOLUMN}$		3.1		3.2	ns
$t_{DIFFROW}$		6.4		6.4	ns
$t_{TWOROWS}$		9.7		10.6	ns
$t_{LEPERIPH}$		6.4		7.1	ns
$t_{LABCARRY}$		0.4		0.6	ns
$t_{LABCASC}$		2.4		3.0	ns

Table 60. EPF10K70 Device EAB Internal Timing Macroparameters *Note (1)*

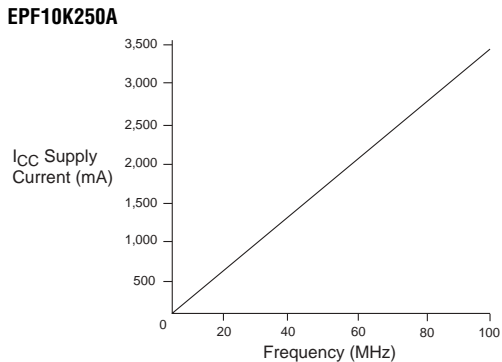
Symbol	-2 Speed Grade		-3 Speed Grade		-4 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{EABAA}		12.1		13.7		17.0	ns
$t_{EABRCCOMB}$	12.1		13.7		17.0		ns
$t_{EABRCREG}$	8.6		9.7		11.9		ns
t_{EABWP}	5.2		5.8		7.2		ns
$t_{EABWCCOMB}$	6.5		7.3		9.0		ns
$t_{EABWCREG}$	11.6		13.0		16.0		ns
t_{EABDD}		8.8		10.0		12.5	ns
$t_{EABDATACO}$		1.7		2.0		3.4	ns
$t_{EABDATASU}$	4.7		5.3		5.6		ns
$t_{EABDATAH}$	0.0		0.0		0.0		ns
$t_{EABWESU}$	4.9		5.5		5.8		ns
t_{EABWEH}	0.0		0.0		0.0		ns
$t_{EABWDSU}$	1.8		2.1		2.7		ns
t_{EABWDH}	0.0		0.0		0.0		ns
$t_{EABWASU}$	4.1		4.7		5.8		ns
t_{EABWAH}	0.0		0.0		0.0		ns
t_{EABWO}		8.4		9.5		11.8	ns

Table 88. EPF10K10A Device EAB Internal Timing Macroparameters *Note (1)*

Symbol	-1 Speed Grade		-2 Speed Grade		-3 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{EABAA}		8.1		9.8		13.1	ns
$t_{EABRCCOMB}$	8.1		9.8		13.1		ns
$t_{EABRCREG}$	5.8		6.9		9.3		ns
t_{EABWP}	2.0		2.4		3.2		ns
$t_{EABWCCOMB}$	3.5		4.2		5.6		ns
$t_{EABWCREG}$	9.4		11.2		14.8		ns
t_{EABDD}		6.9		8.3		11.0	ns
$t_{EABDATACO}$		1.3		1.5		2.0	ns
$t_{EABDATASU}$	2.4		3.0		3.9		ns
$t_{EABDATAH}$	0.0		0.0		0.0		ns
$t_{EABWESU}$	4.1		4.9		6.5		ns
t_{EABWEH}	0.0		0.0		0.0		ns
$t_{EABWDSU}$	1.4		1.6		2.2		ns
t_{EABWDH}	0.0		0.0		0.0		ns
$t_{EABWASU}$	2.5		3.0		4.1		ns
t_{EABWAH}	0.0		0.0		0.0		ns
t_{EABWO}		6.2		7.5		9.9	ns

Table 95. EPF10K30A Device EAB Internal Timing Macroparameters *Note (1)*

Symbol	-1 Speed Grade		-2 Speed Grade		-3 Speed Grade		Unit
	Min	Max	Min	Max	Min	Max	
t_{EABAA}		9.7		11.6		16.2	ns
$t_{EABRCCOMB}$	9.7		11.6		16.2		ns
$t_{EABRCREG}$	5.9		7.1		9.7		ns
t_{EABWP}	3.8		4.5		5.9		ns
$t_{EABWCCOMB}$	4.0		4.7		6.3		ns
$t_{EABWCREG}$	9.8		11.6		16.6		ns
t_{EABDD}		9.2		11.0		16.1	ns
$t_{EABDATACO}$		1.7		2.1		3.4	ns
$t_{EABDATASU}$	2.3		2.7		3.5		ns
$t_{EABDATAH}$	0.0		0.0		0.0		ns
$t_{EABWESU}$	3.3		3.9		4.9		ns
t_{EABWEH}	0.0		0.0		0.0		ns
$t_{EABWDSU}$	3.2		3.8		5.0		ns
t_{EABWDH}	0.0		0.0		0.0		ns
$t_{EABWASU}$	3.7		4.4		5.1		ns
t_{EABWAH}	0.0		0.0		0.0		ns
t_{EABWO}		6.1		7.3		11.3	ns

Figure 32. $I_{CCACTIVE}$ vs. Operating Frequency (Part 3 of 3)

Configuration & Operation



The FLEX 10K architecture supports several configuration schemes. This section summarizes the device operating modes and available device configuration schemes.

See *Application Note 116 (Configuring APEX 20K, FLEX 10K & FLEX 6000 Devices)* for detailed descriptions of device configuration options, device configuration pins, and for information on configuring FLEX 10K devices, including sample schematics, timing diagrams, and configuration parameters.

Operating Modes

The FLEX 10K architecture uses SRAM configuration elements that require configuration data to be loaded every time the circuit powers up. The process of physically loading the SRAM data into the device is called *configuration*. Before configuration, as VCC rises, the device initiates a Power-On Reset (POR). This POR event clears the device and prepares it for configuration. The FLEX 10K POR time does not exceed 50 μ s.

During initialization, which occurs immediately after configuration, the device resets registers, enables I/O pins, and begins to operate as a logic device. The I/O pins are tri-stated during power-up, and before and during configuration. Together, the configuration and initialization processes are called *command mode*; normal device operation is called *user mode*.



Notes: