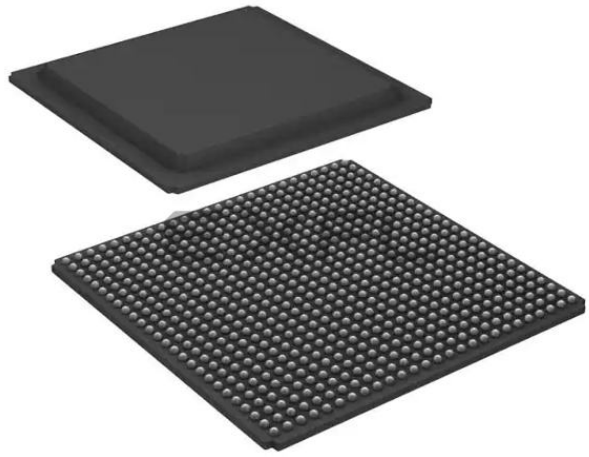

Welcome to E-XFL.COM

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	Obsolete
Number of LABs/CLBs	2400
Number of Logic Elements/Cells	10800
Total RAM Bits	573440
Number of I/O	404
Number of Gates	129600
Voltage - Supply	1.71V ~ 1.89V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	676-BGA
Supplier Device Package	676-FBGA (27x27)
Purchase URL	https://www.e-xfl.com/product-detail/xilinx/xcv405e-8fg676c

Table 1: **Virtex-E Extended Memory Field-Programmable Gate Array Family Members**

Device	Logic Gates	CLB Array	Logic Cells	Differential I/O Pairs	User I/O	BlockRAM Bits	Distributed RAM Bits
XCV405E	129,600	40 x 60	10,800	183	404	573,440	153,600
XCV812E	254,016	56 x 84	21,168	201	556	1,146,880	301,056

Virtex-E Compared to Virtex Devices

The Virtex-E family offers up to 43,200 logic cells in devices up to 30% faster than the Virtex family.

I/O performance is increased to 622 Mb/s using Source Synchronous data transmission architectures and synchronous system performance up to 240 MHz using singled-ended SelectI/O technology. Additional I/O standards are supported, notably LVPECL, LVDS, and BLVDS, which use two pins per signal. Almost all signal pins can be used for these new standards.

Virtex-E devices have up to 640 Kb of faster (250MHz) block SelectRAM, but the individual RAMs are the same size and structure as in the Virtex family. They also have eight DLLs instead of the four in Virtex devices. Each individual DLL is slightly improved with easier clock mirroring and 4x frequency multiplication.

V_{CCINT}, the supply voltage for the internal logic and memory, is 1.8 V, instead of 2.5 V for Virtex devices. Advanced processing and 0.18 µm design rules have resulted in smaller dice, faster speed, and lower power consumption.

I/O pins are 3 V tolerant, and can be 5 V tolerant with an external 100 Ω resistor. PCI 5 V is not supported. With the addition of appropriate external resistors, any pin can tolerate any voltage desired.

Banking rules are different. With Virtex devices, all input buffers are powered by V_{CCINT}. With Virtex-E devices, the LVTTL, LVCMOS2, and PCI input buffers are powered by the I/O supply voltage V_{CCO}.

The Virtex-E family is not bitstream-compatible with the Virtex family, but Virtex designs can be compiled into equivalent Virtex-E devices.

The same device in the same package for the Virtex-E and Virtex families are pin-compatible with some minor exceptions. See the data sheet pinout section for details.

General Description

The Virtex-E FPGA family delivers high-performance, high-capacity programmable logic solutions. Dramatic increases in silicon efficiency result from optimizing the new architecture for place-and-route efficiency and exploiting an aggressive 6-layer metal 0.18 µm CMOS process. These advances make Virtex-E FPGAs powerful and flexible alternatives to mask-programmed gate arrays. The Virtex-E family includes the nine members in Table 1.

Building on experience gained from Virtex FPGAs, the Virtex-E family is an evolutionary step forward in programmable logic design. Combining a wide variety of programmable system features, a rich hierarchy of fast, flexible interconnect resources, and advanced process technology, the Virtex-E family delivers a high-speed and high-capacity programmable logic solution that enhances design flexibility while reducing time-to-market.

Virtex-E Architecture

Virtex-E devices feature a flexible, regular architecture that comprises an array of configurable logic blocks (CLBs) surrounded by programmable input/output blocks (IOBs), all interconnected by a rich hierarchy of fast, versatile routing resources. The abundance of routing resources permits the Virtex-E family to accommodate even the largest and most complex designs.

Virtex-E FPGAs are SRAM-based, and are customized by loading configuration data into internal memory cells. Configuration data can be read from an external SPROM (master serial mode), or can be written into the FPGA (SelectMAP™, slave serial, and JTAG modes).

The standard Xilinx Foundation Series™ and Alliance Series™ Development systems deliver complete design support for Virtex-E, covering every aspect from behavioral and schematic entry, through simulation, automatic design translation and implementation, to the creation and downloading of a configuration bit stream.

Higher Performance

Virtex-E devices provide better performance than previous generations of FPGAs. Designs can achieve synchronous system clock rates up to 240 MHz including I/O or 622 Mb/s using Source Synchronous data transmission architechtures. Virtex-E I/Os comply fully with 3.3 V PCI specifications, and interfaces can be implemented that operate at 33 MHz or 66 MHz.

While performance is design-dependent, many designs operate internally at speeds in excess of 133 MHz and can achieve over 311 MHz. Table 2, page 3, shows performance data for representative circuits, using worst-case timing parameters.

Table 2: **Performance for Common Circuit Functions**

Function	Bits	Virtex-E -7
Register-to-Register		
Adder	16 64	4.3 ns 6.3 ns
Pipelined Multiplier	8 x 8 16 x 16	4.4 ns 5.1 ns
Address Decoder	16 64	3.8 ns 5.5 ns
16:1 Multiplexer		4.6 ns
Parity Tree	9 18 36	3.5 ns 4.3 ns 5.9 ns
Chip-to-Chip		
HSTL Class IV		
LVTTL,16mA, fast slew		
LVDS		
LVPECL		

Virtex-E Extended Memory Device/Package Combinations and Maximum I/O

Table 3: **Virtex-EM Family Maximum User I/O by Device/Package (Excluding Dedicated Clock Pins)**

Package	XCV405E	XCV812E
BG560	404	404
FG676	404	
FG900		556

Virtex-E Extended Memory Ordering Information

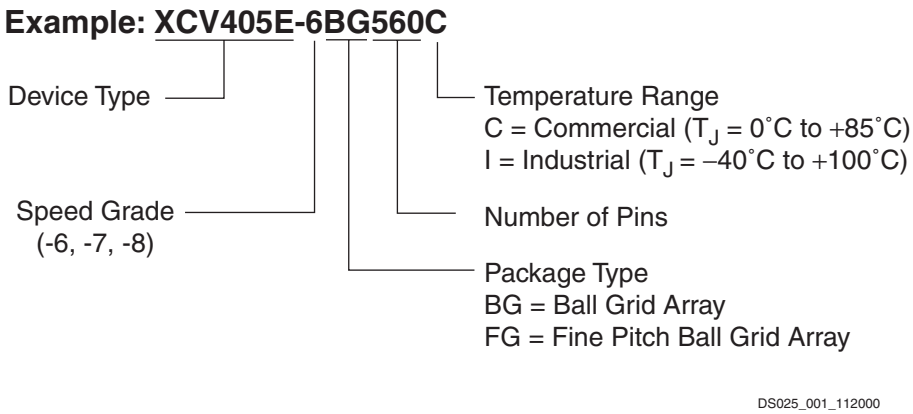

Figure 1: **Virtex Ordering Information**

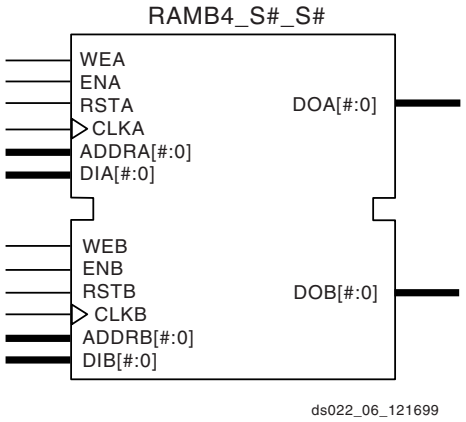

Figure 6: **Dual-Port Block SelectRAM**

Table 5 shows the depth and width aspect ratios for the block SelectRAM. The Virtex-E block SelectRAM also includes dedicated routing to provide an efficient interface with both CLBs and other block SelectRAM modules. Refer to XAPP130 for block SelectRAM timing waveforms.

Table 5: Block SelectRAM Port Aspect Ratios

Width	Depth	ADDR Bus	Data Bus
1	4096	ADDR<11:0>	DATA<0>
2	2048	ADDR<10:0>	DATA<1:0>
4	1024	ADDR<9:0>	DATA<3:0>
8	512	ADDR<8:0>	DATA<7:0>
16	256	ADDR<7:0>	DATA<15:0>

Programmable Routing Matrix

It is the longest delay path that limits the speed of any worst-case design. Consequently, the Virtex-E routing architecture and its place-and-route software were defined in a joint optimization process. This joint optimization minimizes long-path delays, and consequently, yields the best system performance.

The joint optimization also reduces design compilation times because the architecture is software-friendly. Design cycles are correspondingly reduced due to shorter design iteration times.

Local Routing

The VersaBlock, shown in Figure 7, provides local routing resources with the following types of connections:

- Interconnections among the LUTs, flip-flops, and GRM
- Internal CLB feedback paths that provide high-speed connections to LUTs within the same CLB, chaining them together with minimal routing delay
- Direct paths that provide high-speed connections between horizontally adjacent CLBs, eliminating the delay of the GRM

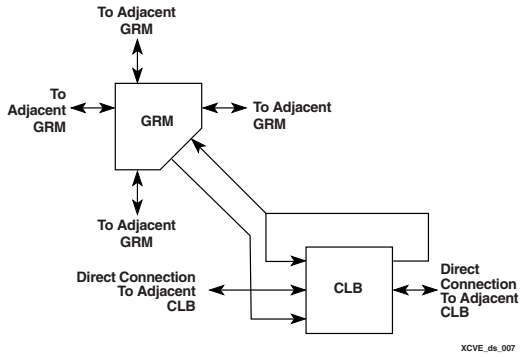

Figure 7: **Virtex-E Local Routing**

General Purpose Routing

Most Virtex-E signals are routed on the general purpose routing, and consequently, the majority of interconnect resources are associated with this level of the routing hierarchy. The general routing resources are located in horizontal and vertical routing channels associated with the CLB rows and columns. The general-purpose routing resources are listed below.

- Adjacent to each CLB is a General Routing Matrix (GRM). The GRM is the switch matrix through which horizontal and vertical routing resources connect, and is also the means by which the CLB gains access to the general purpose routing.
- 24 single-length lines route GRM signals to adjacent GRMs in each of the four directions.
- 72 buffered Hex lines route GRM signals to another GRMs six-blocks away in each one of the four directions. Organized in a staggered pattern, Hex lines are driven only at their endpoints. Hex-line signals can be accessed either at the endpoints or at the midpoint (three blocks from the source). One third of the Hex lines are bidirectional, while the remaining ones are uni-directional.
- 12 Longlines are buffered, bidirectional wires that distribute signals across the device quickly and efficiently. Vertical Longlines span the full height of the device, and horizontal ones span the full width of the device.

I/O Routing

Virtex-E devices have additional routing resources around their periphery that form an interface between the CLB array and the IOBs. This additional routing, called the VersaRing, facilitates pin-swapping and pin-locking, such that logic redesigns can adapt to existing PCB layouts. Time-to-market is reduced, since PCBs and other system components can be manufactured while the logic design is still in progress.

Identification Registers

The IDCODE register is supported. By using the IDCODE, the device connected to the JTAG port can be determined.

The IDCODE register has the following binary format:

vvvv:ffff:fffa:aaaa:aaaa:cccc:cccc:ccc1

where

v = the die version number

f = the family code (05 for Virtex-E family)

a = the number of CLB rows (ranges from 16 for XCV50E to 104 for XCV3200E)

c = the company code (49h for Xilinx)

The USERCODE register is supported. By using the USERCODE, a user-programmable identification code can be loaded and shifted out for examination. The identification code (see Table 7) is embedded in the bitstream during bitstream generation and is valid only after configuration.

Table 7: **IDCODEs Assigned to Virtex-E FPGAs**

FPGA	IDCODE
XCV405EM	v0C28093h
XCV812EM	v0C38093h

Note:
Attempting to load an incorrect bitstream causes configuration to fail and can damage the device.

Including Boundary Scan in a Design

Since the boundary scan pins are dedicated, no special element needs to be added to the design unless an internal data register (USER1 or USER2) is desired.

If an internal data register is used, insert the boundary scan symbol and connect the necessary pins as appropriate.

Development System

Virtex-E FPGAs are supported by the Xilinx Foundation and Alliance Series CAE tools. The basic methodology for Virtex-E design consists of three interrelated steps: design entry, implementation, and verification. Industry-standard tools are used for design entry and simulation (for example, Synopsys FPGA Express), while Xilinx provides proprietary architecture-specific tools for implementation.

The Xilinx development system is integrated under the Xilinx Design Manager (XDM™) software, providing designers with a common user interface regardless of their choice of entry and verification tools. The XDM software simplifies the selection of implementation options with pull-down menus and on-line help.

Application programs ranging from schematic capture to Placement and Routing (PAR) can be accessed through the XDM software. The program command sequence is generated prior to execution, and stored for documentation.

Several advanced software features facilitate Virtex-E design. RPMs, for example, are schematic-based macros with relative location constraints to guide their placement. They help ensure optimal implementation of common functions.

For HDL design entry, the Xilinx FPGA Foundation development system provides interfaces to the following synthesis design environments.

- Synopsys (FPGA Compiler, FPGA Express)
- Exemplar (Spectrum)
- Synplicity (Synplify)

For schematic design entry, the Xilinx FPGA Foundation and Alliance development system provides interfaces to the following schematic-capture design environments.

- Mentor Graphics V8 (Design Architect, QuickSim II)
- Viewlogic Systems (Viewdraw)

Third-party vendors support many other environments.

A standard interface-file specification, Electronic Design Interchange Format (EDIF), simplifies file transfers into and out of the development system.

Virtex-E FPGAs are supported by a unified library of standard functions. This library contains over 400 primitives and macros, ranging from 2-input AND gates to 16-bit accumulators, and includes arithmetic functions, comparators, counters, data registers, decoders, encoders, I/O functions, latches, Boolean functions, multiplexers, shift registers, and barrel shifters.

The “soft macro” portion of the library contains detailed descriptions of common logic functions, but does not contain any partitioning or placement information. The performance of these macros depends, therefore, on the partitioning and placement obtained during implementation.

RPMs, on the other hand, do contain predetermined partitioning and placement information that permits optimal implementation of these functions. Users can create their own library of soft macros or RPMs based on the macros and primitives in the standard library.

The design environment supports hierarchical design entry, with high-level schematics that comprise major functional blocks, while lower-level schematics define the logic in these blocks. These hierarchical design elements are automatically combined by the implementation tools. Different design entry tools can be combined within a hierarchical design, thus allowing the most convenient entry method to be used for each portion of the design.

Design Implementation

The place-and-route tools (PAR) automatically provide the implementation flow described in this section. The partitioner takes the EDIF net list for the design and maps the logic into the architectural resources of the FPGA (CLBs and IOBs, for example). The placer then determines the best locations for these blocks based on their interconnec-

Table 8: **Configuration Codes**

Configuration Mode	M2	M1	M0	CCLK Direction	Data Width	Serial D_{out}	Configuration Pull-ups
Boundary-scan mode	0	0	1	N/A	1	No	Yes
SelectMAP mode	0	1	0	In	8	No	Yes
Slave-serial mode	0	1	1	In	1	Yes	Yes

Table 9 lists the total number of bits required to configure each device.

Table 9: **Virtex-E Bitstream Lengths**

Device	# of Configuration Bits
XCV405E	3,430,400
XCV812E	6,519,648

Slave-Serial Mode

In slave-serial mode, the FPGA receives configuration data in bit-serial form from a serial PROM or other source of serial configuration data. The serial bitstream must be set up at the DIN input pin a short time before each rising edge of an externally generated CCLK.

For more detailed information on serial PROMs see the PROM data sheet at http://www.xilinx.com/bvdocs/publications/ds026.pdf.

Multiple FPGAs can be daisy-chained for configuration from a single source. After a particular FPGA has been configured, the data for the next device is routed to the DOUT pin. Data on the DOUT pin changes on the rising edge of CCLK.

The change of DOUT on the rising edge of CCLK differs from previous families but does not cause a problem for mixed configuration chains. This change was made to improve serial configuration rates for Virtex and Virtex-E only chains.

Figure 13 shows a full master/slave system. A Virtex-E device in slave-serial mode should be connected as shown in the right-most device.

Slave-serial mode is selected by applying <111> or <011> to the mode pins (M2, M1, M0). A weak pull-up on the mode pins makes slave-serial the default mode if the pins are left unconnected. However, it is recommended to drive the configuration mode pins externally. Figure 14 shows slave-serial mode programming switching characteristics.

Table 10 provides more detail about the characteristics shown in Figure 14. Configuration must be delayed until the $\overline{INIT}$ pins of all daisy-chained FPGAs are High.

Table 10: **Master/Slave Serial Mode Programming Switching**

	Description	Figure References	Symbol	Values	Units
CCLK	DIN setup/hold, slave mode	1/2	T_{DCC}/T_{CCD}	5.0/0.0	ns, min
	DIN setup/hold, master mode	1/2	T_{DSCK}/T_{CKDS}	5.0/0.0	ns, min
	DOUT	3	T_{CCO}	12.0	ns, max
	High time	4	T_{CCH}	5.0	ns, min
	Low time	5	T_{CCL}	5.0	ns, min
	Maximum Frequency		F_{CC}	66	MHz, max
	Frequency Tolerance, master mode with respect to nominal			+45% –30%	

The sequence of operations necessary to configure a Virtex-E FPGA serially appears in Figure 15.

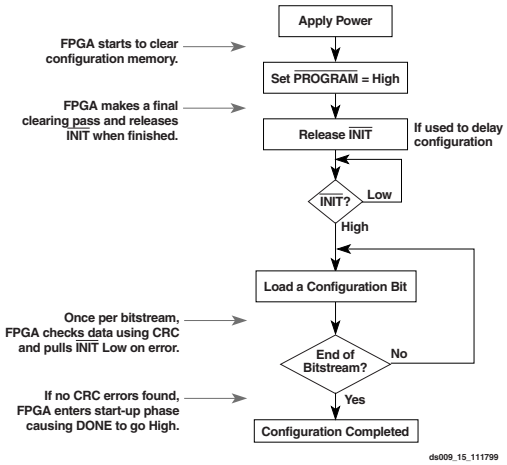

Figure 15: **Serial Configuration Flowchart**

Figure 16 shows the timing of master-serial configuration. Master-serial mode is selected by a <000> or <100> on the mode pins (M2, M1, M0). Table 10 shows the timing information for Figure 16

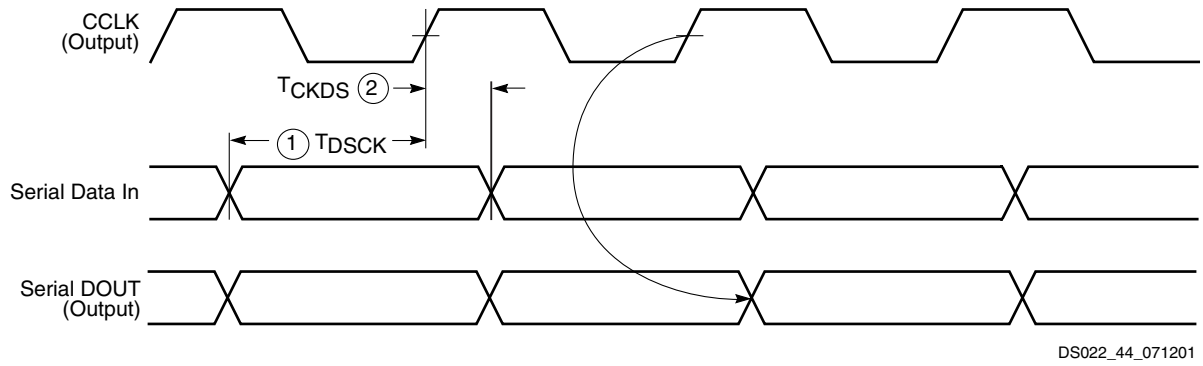

Figure 16: **Master-Serial Mode Programming Switching Characteristics**

At power-up, V_{CC} must rise from 1.0 V to V_{CC} min in less than 50 ms, otherwise delay configuration by pulling $\overline{\text{PROGRAM}}$ Low until V_{CC} is valid.

SelectMAP Mode

The SelectMAP mode is the fastest configuration option. Byte-wide data is written into the FPGA with a BUSY flag controlling the flow of data.

An external data source provides a byte stream, CCLK, a Chip Select ($\overline{\text{CS}}$) signal and a Write signal ($\overline{\text{WRITE}}$). If BUSY is asserted (High) by the FPGA, the data must be held until BUSY goes Low.

Data can also be read using the SelectMAP mode. If $\overline{\text{WRITE}}$ is not asserted, configuration data is read out of the FPGA as part of a readback operation.

After configuration, the pins of the SelectMAP port can be used as additional user I/O. Alternatively, the port can be retained to permit high-speed 8-bit readback.

Retention of the SelectMAP port is selectable on a design-by-design basis when the bitstream is generated. If retention is selected, PROHIBIT constraints are required to prevent SelectMAP-port pins from being used as user I/O.

Multiple Virtex-E FPGAs can be configured using the SelectMAP mode, and be made to start-up simultaneously. To configure multiple devices in this way, wire the individual CCLK, Data, $\overline{\text{WRITE}}$, and BUSY pins of all the devices in parallel. The individual devices are loaded separately by asserting the $\overline{\text{CS}}$ pin of each device in turn and writing the appropriate data. See Table 11 for SelectMAP Write Timing Characteristics.

Write

Write operations send packets of configuration data into the FPGA. The sequence of operations for a multi-cycle write operation is shown below. Note that a configuration packet can be split into many such sequences. The packet does not have to complete within one assertion of $\overline{\text{CS}}$, illustrated in Figure 17.

1. Assert $\overline{\text{WRITE}}$ and $\overline{\text{CS}}$ Low. Note that when $\overline{\text{CS}}$ is asserted on successive CCLKs, $\overline{\text{WRITE}}$ must remain

Configuration through the TAP uses the CFG_IN instruction. This instruction allows data input on TDI to be converted into data packets for the internal configuration bus.

The following steps are required to configure the FPGA through the boundary-scan port (when using TCK as a start-up clock).

1. Load the CFG_IN instruction into the boundary-scan instruction register (IR)
2. Enter the Shift-DR (SDR) state
3. Shift a configuration bitstream into TDI
4. Return to Run-Test-Idle (RTI)
5. Load the JSTART instruction into IR
6. Enter the SDR state
7. Clock TCK through the startup sequence
8. Return to RTI

Configuration and readback via the TAP is always available. The boundary-scan mode is selected by a <101> or <001> on the mode pins (M2, M1, M0). For details on TAP characteristics, refer to XAPP139.

Configuration Sequence

The configuration of Virtex-E devices is a three-phase process. First, the configuration memory is cleared. Next, configuration data is loaded into the memory, and finally, the logic is activated by a start-up process.

Configuration is automatically initiated on power-up unless it is delayed by the user, as described below. The configuration process can also be initiated by asserting $\overline{\text{PROGRAM}}$. The end of the memory-clearing phase is signalled by $\overline{\text{INIT}}$ going High, and the completion of the entire process is signalled by DONE going High.

The power-up timing of configuration signals is shown in Figure 20.

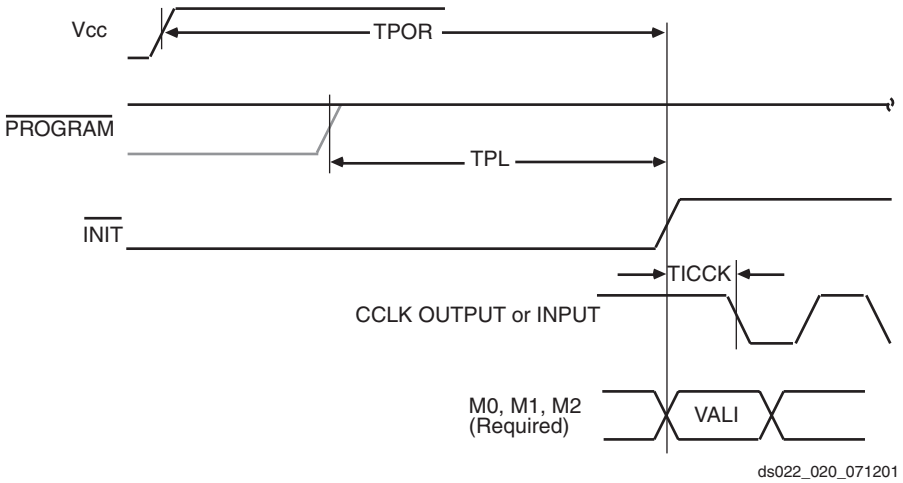

Figure 20: **Power-Up Timing Configuration Signals**

The corresponding timing characteristics are listed in Table 12.

Table 12: **Power-up Timing Characteristics**

Description	Symbol	Value	Units
Power-on Reset[1]	T_{POR}	2.0	ms, max
Program Latency	T_{PL}	100.0	µs, max
CCLK (output) Delay	T_{ICCK}	0.5	µs, min
		4.0	µs, max
Program Pulse Width	$T_{PROGRAM}$	300	ns, min

Notes:
1. T_{POR} delay is the initialization time required after V_{CCINT} reaches the recommended operating voltage.

Delaying Configuration

$\overline{\text{INIT}}$ can be held Low using an open-drain driver. An open-drain is required since $\overline{\text{INIT}}$ is a bidirectional open-drain pin that is held Low by the FPGA while the configuration memory is being cleared. Extending the time that the pin is Low causes the configuration sequencer to wait. Thus, configuration is delayed by preventing entry into the phase where data is loaded.

Start-Up Sequence

The default Start-up sequence is that one CCLK cycle after DONE goes High, the global 3-state signal (GTS) is released. This permits device outputs to turn on as necessary.

One CCLK cycle later, the Global Set/Reset (GSR) and Global Write Enable (GWE) signals are released. This permits

1x Clock Outputs — CLK[0|90|180|270]

The 1x clock output pin CLK0 represents a delay-compensated version of the source clock (CLKIN) signal. The CLKDLL primitive provides three phase-shifted versions of the CLK0 signal while CLKDLLHF provides only the 180 phase-shifted version. The relationship between phase shift and the corresponding period shift appears in Table 13.

Table 13: **Relationship of Phase-Shifted Output Clock to Period Shift**

Phase (degrees)	Period Shift (percent)
0	0%
90	25%
180	50%
270	75%

The timing diagrams in Figure 25 illustrate the DLL clock output characteristics.

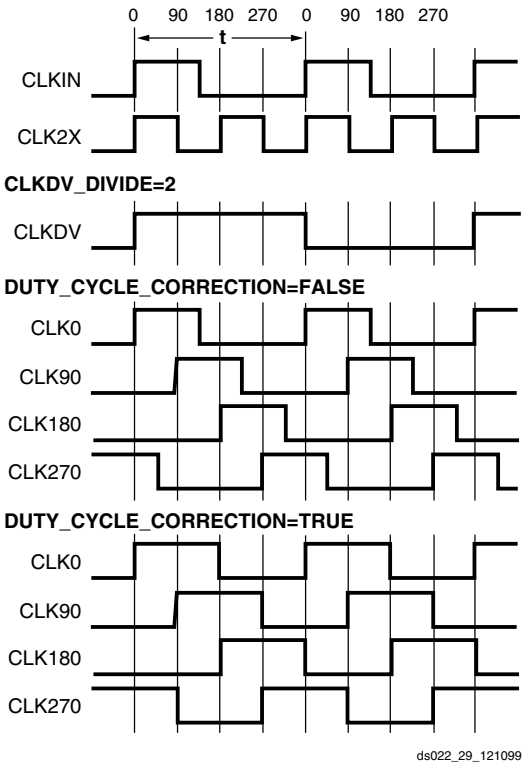

Figure 25: **DLL Output Characteristics**

The DLL provides duty cycle correction on all 1x clock outputs such that all 1x clock outputs by default have a 50/50 duty cycle. The DUTY_CYCLE_CORRECTION property (TRUE by default), controls this feature. In order to deactivate the DLL duty cycle correction, attach the DUTY_CYCLE_CORRECTION=FALSE property to the DLL symbol. When duty cycle correction deactivates, the output clock has the same duty cycle as the source clock.

The DLL clock outputs can drive an OBUF, a BUFG, or they can route directly to destination clock pins. The DLL clock outputs can only drive the BUFGs that reside on the same edge (top or bottom).

Locked Output — LOCKED

To achieve lock, the DLL might need to sample several thousand clock cycles. After the DLL achieves lock, the LOCKED signal activates. The DLL timing parameter section of the data sheet provides estimates for locking times.

To guarantee that the system clock is established prior to the device "waking up," the DLL can delay the completion of the device configuration process until after the DLL locks. The STARTUP_WAIT property activates this feature.

Until the LOCKED signal activates, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular the CLK2X output appears as a 1x clock with a 25/75 duty cycle.

DLL Properties

Properties provide access to some of the Virtex-E series DLL features, (for example, clock division and duty cycle correction).

Duty Cycle Correction Property

The 1x clock outputs, CLK0, CLK90, CLK180, and CLK270, use the duty-cycle corrected default, exhibiting a 50/50 duty cycle. The DUTY_CYCLE_CORRECTION property (by default TRUE) controls this feature. To deactivate the DLL duty-cycle correction for the 1x clock outputs, attach the DUTY_CYCLE_CORRECTION=FALSE property to the DLL symbol.

Clock Divide Property

The CLKDV_DIVIDE property specifies how the signal on the CLKDV pin is frequency divided with respect to the CLK0 pin. The values allowed for this property are 1.5, 2, 2.5, 3, 4, 5, 8, or 16; the default value is 2.

Startup Delay Property

This property, STARTUP_WAIT, takes on a value of TRUE or FALSE (the default value). When TRUE the device configuration DONE signal waits until the DLL locks before going to High.

Virtex-E DLL Location Constraints

As shown in Figure 26, there are four additional DLLs in the Virtex-E devices, for a total of eight per Virtex-E device. These DLLs are located in silicon, at the top and bottom of the two innermost block SelectRAM columns. The location constraint LOC, attached to the DLL symbol with the identifier DLL0S, DLL0P, DLL1S, DLL1P, DLL2S, DLL2P, DLL3S, or DLL3P, controls the DLL location.

The LOC property uses the following form:

LOC = DLL0P

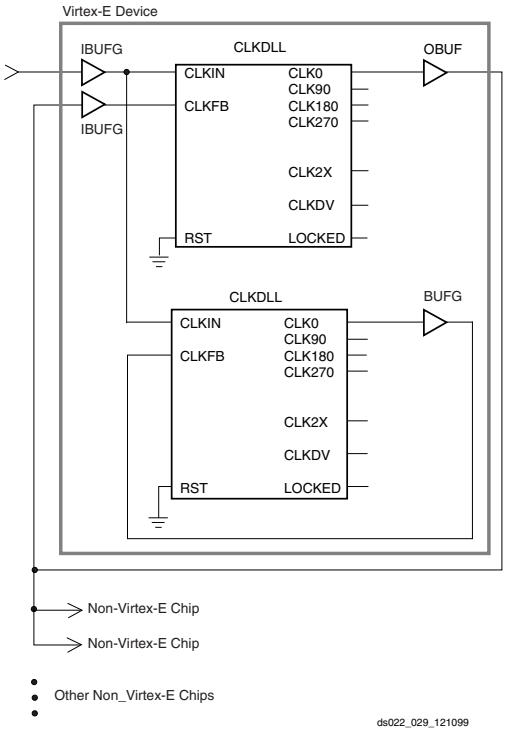

Figure 28: **DLL De-skew of Board Level Clock**

Board-level de-skew is not required for low-fanout clock networks. It is recommended for systems that have fanout limitations on the clock network, or if the clock distribution chip cannot handle the load.

Do not use the DLL output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

The dll_mirror_1 files in the xapp132.zip file show the VHDL and Verilog implementation of this circuit.

De-Skew of Clock and Its 2x Multiple

The circuit shown in Figure 29 implements a 2x clock multiplier and also uses the CLK0 clock output with zero ns skew between registers on the same chip. A clock divider circuit could alternatively be implemented using similar connections.

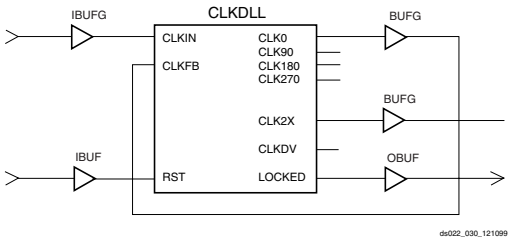

Figure 29: **DLL De-skew of Clock and 2x Multiple**

Because any single DLL can access only two BUFGs at most, any additional output clock signals must be routed from the DLL in this example on the high speed backbone routing.

The dll_2x files in the xapp132.zip file show the VHDL and Verilog implementation of this circuit.

Virtex-E 4x Clock

Two DLLs located in the same half-edge (top-left, top-right, bottom-right, bottom-left) can be connected together, without using a BUFG between the CLKDLLs, to generate a 4x clock as shown in Figure 30. Virtex-E devices, like the Virtex devices, have four clock networks that are available for internal de-skewing of the clock. Each of the eight DLLs have access to two of the four clock networks. Although all the DLLs can be used for internal de-skewing, the presence of two GCLKBUFs on the top and two on the bottom indicate that only two of the four DLLs on the top (and two of the four DLLs on the bottom) can be used for this purpose.

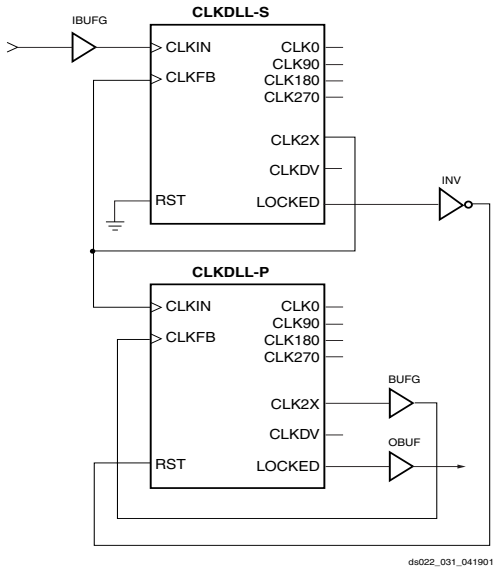

Figure 30: **DLL Generation of 4x Clock in Virtex-E Devices**

The dll_4xe files in the xapp 32.zip file show the DLL implementation in Verilog for Virtex-E devices. These files can be found at:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp132.zip

Using Block SelectRAM+ Features

The Virtex FPGA Series provides dedicated blocks of on-chip, true dual-read/write port synchronous RAM, with 4096 memory cells. Each port of the block SelectRAM+ memory can be independently configured as a read/write port, a read port, a write port, and can be configured to a specific data width. block SelectRAM+ memory offers new capabilities, allowing FPGA designers to simplify designs.

The voltage reference signal is “banked” within the Virtex-E device on a half-edge basis such that for all packages there are eight independent V_{REF} banks internally. See Figure 38 for a representation of the Virtex-E I/O banks. Within each bank approximately one of every six I/O pins is automatically configured as a V_{REF} input. After placing a differential amplifier input signal within a given V_{REF} bank, the same external source must drive all I/O pins configured as a V_{REF} input.

IBUF placement restrictions require that any differential amplifier input signals within a bank be of the same standard. How to specify a specific location for the IBUF via the LOC property is described below. Table 19 summarizes the Virtex-E input standards compatibility requirements.

An optional delay element is associated with each IBUF. When the IBUF drives a flip-flop within the IOB, the delay element by default activates to ensure a zero hold-time requirement. The NODELAY=TRUE property overrides this default.

When the IBUF does not drive a flip-flop within the IOB, the delay element de-activates by default to provide higher performance. To delay the input signal, activate the delay element with the DELAY=TRUE property.

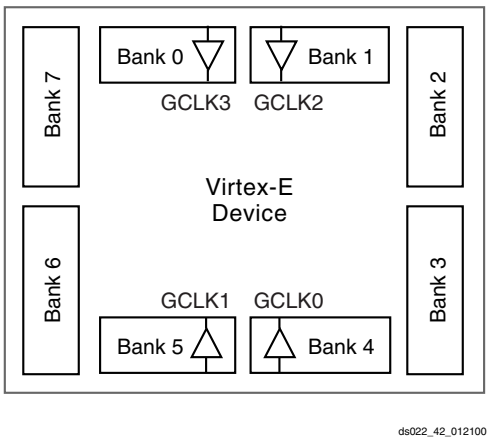

Figure 38: **Virtex-E I/O Banks**

Table 19: **Xilinx Input Standards Compatibility Requirements**

Rule 1	Standards with the same input V_{CCO}, output V_{CCO}, and V_{REF} can be placed within the same bank.

IBUFG

Signals used as high fanout clock inputs to the Virtex-E device should drive a global clock input buffer (IBUFG) via an external input port in order to take advantage of one of the four dedicated global clock distribution networks. The output of the IBUFG should only drive a CLKDLL, CLKDLLHF, or a BUFG symbol. The generic Virtex-E IBUFG symbol appears in Figure 39.

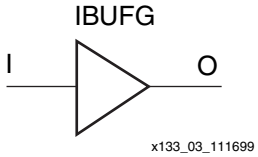

Figure 39: **Virtex-E Global Clock Input Buffer (IBUFG) Symbol**

The extension to the base name determines which I/O standard is used by the IBUFG. With no extension specified for the generic IBUFG symbol, the assumed standard is LVTTL.

The following list details variations of the IBUFG symbol.

- IBUFG
- IBUFG_LVCMOS2
- IBUFG_PCI33_3
- IBUFG_PCI66_3
- IBUFG_GTL
- IBUFG_GTLP
- IBUFG_HSTL_I
- IBUFG_HSTL_III
- IBUFG_HSTL_IV
- IBUFG_SSTL3_I
- IBUFG_SSTL3_II
- IBUFG_SSTL2_I
- IBUFG_SSTL2_II
- IBUFG_CTT
- IBUFG_AGP
- IBUFG_LVCMOS18
- IBUFG_LVDS
- IBUFG_LVPECL

When the IBUFG symbol supports an I/O standard that requires a differential amplifier input, the IBUFG automatically configures as a differential amplifier input buffer. The low-voltage I/O standards with a differential amplifier input require an external reference voltage input V_{REF}.

The voltage reference signal is “banked” within the Virtex-E device on a half-edge basis such that for all packages there are eight independent V_{REF} banks internally. See Figure 38 for a representation of the Virtex-E I/O banks. Within each bank approximately one of every six I/O pins is automatically configured as a V_{REF} input. After placing a differential amplifier input signal within a given V_{REF} bank, the same external source must drive all I/O pins configured as a V_{REF} input.

IBUFG placement restrictions require any differential amplifier input signals within a bank be of the same standard. The LOC property can specify a location for the IBUFG.

Application Examples

Creating a design with the SelectI/O features requires the instantiation of the desired library symbol within the design code. At the board level, designers need to know the termination techniques required for each I/O standard.

This section describes some common application examples illustrating the termination techniques recommended by each of the standards supported by the SelectI/O features.

Termination Examples

Circuit examples involving typical termination techniques for each of the SelectI/O standards follow. For a full range of accepted values for the DC voltage specifications for each standard, refer to the table associated with each figure.

The resistors used in each termination technique example and the transmission lines depicted represent board level components and are not meant to represent components on the device.

GTL

A sample circuit illustrating a valid termination technique for GTL is shown in Figure 44. Table 23 lists DC voltage specifications.

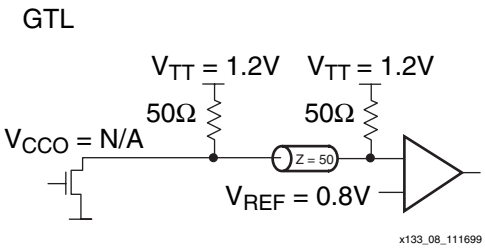

Figure 44: **Terminated GTL**

Table 23: **GTL Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	-	N/A	-
$V_{REF} = N \times V_{TT}$[1]	0.74	0.8	0.86
V_{TT}	1.14	1.2	1.26
$V_{IH} = V_{REF} + 0.05$	0.79	0.85	-
$V_{IL} = V_{REF} - 0.05$	-	0.75	0.81
V_{OH}	-	-	-
V_{OL}	-	0.2	0.4
I_{OH} at V_{OH}(mA)	-	-	-
I_{OL}at V_{OL}(mA) at 0.4V	32	-	-
I_{OL}at V_{OL}(mA) at 0.2V	-	-	40

Note: N must be greater than or equal to 0.653 and less than or equal to 0.68.

GTL+

A sample circuit illustrating a valid termination technique for GTL+ appears in Figure 45. DC voltage specifications appear in Table 24.

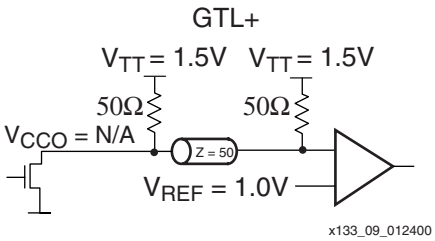

Figure 45: **Terminated GTL+**

Table 24: **GTL+ Voltage Specifications**

Parameter	Min	Typ	Max
V_{CCO}	-	-	-
$V_{REF} = N \times V_{TT}$[1]	0.88	1.0	1.12
V_{TT}	1.35	1.5	1.65
$V_{IH} = V_{REF} + 0.1$	0.98	1.1	-
$V_{IL} = V_{REF} - 0.1$	-	0.9	1.02
V_{OH}	-	-	-
V_{OL}	0.3	0.45	0.6
I_{OH} at V_{OH} (mA)	-	-	-
I_{OL}at V_{OL} (mA) at 0.6V	36	-	-
I_{OL}at V_{OL} (mA) at 0.3V	-	-	48

Note: N must be greater than or equal to 0.653 and less than or equal to 0.68.

HSTL

A sample circuit illustrating a valid termination technique for HSTL_I appears in Figure 46. A sample circuit illustrating a valid termination technique for HSTL_III appears in Figure 47.

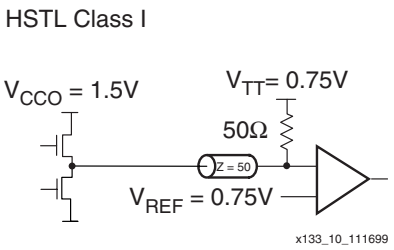

Figure 46: **Terminated HSTL Class I**

Table 25: **HSTL Class I Voltage Specification**

Parameter	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF}	0.68	0.75	0.90
V_{TT}	-	$V_{CCO} \times 0.5$	-
V_{IH}	V_{REF} + 0.1	-	-
V_{IL}	-	-	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	-	-
V_{OL}			0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	8	-	-

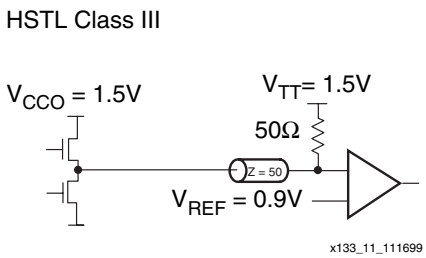

Figure 47: **Terminated HSTL Class III**

Table 26: **HSTL Class III Voltage Specification**

Parameter	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF} (1)	-	0.90	-
V_{TT}	-	V_{CCO}	-
V_{IH}	V_{REF} + 0.1	-	-
V_{IL}	-	-	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	24	-	-

Note: Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user."

A sample circuit illustrating a valid termination technique for HSTL_IV appears in Figure 48.

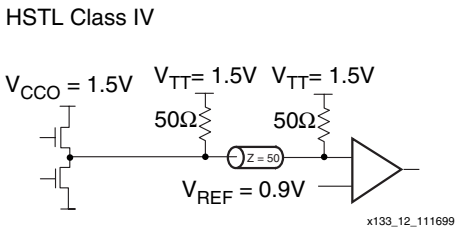

Figure 48: **Terminated HSTL Class IV**

Table 27: **HSTL Class IV Voltage Specification**

Parameter	Min	Typ	Max
V_{CCO}	1.40	1.50	1.60
V_{REF}	-	0.90	-
V_{TT}	-	V_{CCO}	-
V_{IH}	V_{REF} + 0.1	-	-
V_{IL}	-	-	V_{REF} – 0.1
V_{OH}	V_{CCO} – 0.4	-	-
V_{OL}	-	-	0.4
I_{OH} at V_{OH} (mA)	–8	-	-
I_{OL}at V_{OL} (mA)	48	-	-

Note: Per EIA/JESD8-6, "The value of V_{REF} is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.

Creating LVDS Output Buffers

LVDS output buffer can be placed in wide number of IOB locations. The exact location are dependent on the package that is used. The Virtex-E package information lists the possible locations as IO_L#P for the P-side and IO_L#N for the N-side where # is the pair number.

HDL Instantiation

Both output buffers are required to be instantiated in the design and placed on the correct IO_L#P and IO_L#N locations. The IOB must have the same net source the following pins, clock (C), set/reset (SR), output (O), output clock enable (OCE). In addition, the output (O) pins must be inverted with respect to each other, and if output registers are used, the INIT states must be opposite values (one HIGH and one LOW). Failure to follow these rules leads to DRC errors in software.

VHDL Instantiation

```
data0_p  : OBUF_LVDS port map
(I=>data_int(0),   O=>data_p(0));

data0_inv: INV       port map
(I=>data_int(0),   O=>data_n_int(0));

data0_n  : OBUF_LVDS port map
(I=>data_n_int(0), O=>data_n(0));
```

Verilog Instantiation

```
OBUF_LVDS data0_p   (.I(data_int[0]),
.O(data_p[0]));

INV       data0_inv (.I(data_int[0],
.O(data_n_int[0]);

OBUF_LVDS data0_n   (.I(data_n_int[0]),
.O(data_n[0]));
```

Location Constraints

All LVDS buffers must be explicitly placed on a device. For the output buffers this can be done with the following constraint in the UCF or NCF file.

```
NET data_p<0> LOC = D28; # IO_L0P

NET data_n<0> LOC = B29; # IO_L0N
```

Synchronous vs. Asynchronous Outputs

If the outputs are synchronous (registered in the IOB), then any IO_L#PIN pair can be used. If the outputs are asynchronous (no output register), then they must use one of the pairs that are part of the same IOB group at the end of a ROW or at the top/bottom of a COLUMN in the device.

The LVDS pairs that can be used as asynchronous outputs are listed in the Virtex-E pinout tables. Some pairs are marked as asynchronous-capable for all devices in that package, and others are marked as available only for that device in the package. If the device size might change at some point in the product lifetime, then only the common pairs for all packages should be used.

Adding an Output Register

All LVDS buffers can have an output register in the IOB. The output registers must be in both the P-side and N-side IOBs. All the normal IOB register options are available (FD, FDE, FDC, FDCE, FDP, FDPE, FDR, FDRE, FDS, FDSE, LD, LDE, LDC, LDCE, LDP, LDPE). The register elements can be inferred or explicitly instantiated in the HDL code.

Special care must be taken to insure that the D pins of the registers are inverted and that the INIT states of the registers are opposite. The clock pin (C), clock enable (CE) and set/reset (CLR/PRE or S/R) pins must connect to the same source. Failure to do this leads to a DRC error in the software.

The register elements can be packed in the IOB using the IOB property to TRUE on the register or by using the “map -pr [i|o|b]” where “i” is inputs only, “o” is outputs only and “b” is both inputs and outputs.

To improve design coding times VHDL and Verilog synthesis macro libraries have been developed to explicitly create these structures. The output library macros are listed in Table 43. The O and OB inputs to the macros are the external net connections.

Table 43: **Output Library Macros**

Name	Inputs	Outputs
OBUFDS_FD_LVDS	D, C	O, OB
OBUFDS_FDE_LVDS	DD, CE, C	O, OB
OBUFDS_FDC_LVDS	D, C, CLR	O, OB
OBUFDS_FDCE_LVDS	D, CE, C, CLR	O, OB
OBUFDS_FDP_LVDS	D, C, PRE	O, OB
OBUFDS_FDPE_LVDS	D, CE, C, PRE	O, OB
OBUFDS_FDR_LVDS	D, C, R	O, OB
OBUFDS_FDRE_LVDS	D, CE, C, R	O, OB
OBUFDS_FDS_LVDS	D, C, S	O, OB
OBUFDS_FDSE_LVDS	D, CE, C, S	O, OB
OBUFDS_LD_LVDS	D, G	O, OB
OBUFDS_LDE_LVDS	D, GE, G	O, OB
OBUFDS_LDC_LVDS	D, G, CLR	O, OB
OBUFDS_LDCE_LVDS	D, GE, G, CLR	O, OB
OBUFDS_LDP_LVDS	D, G, PRE	O, OB
OBUFDS_LDPE_LVDS	D, GE, G, PRE	O, OB

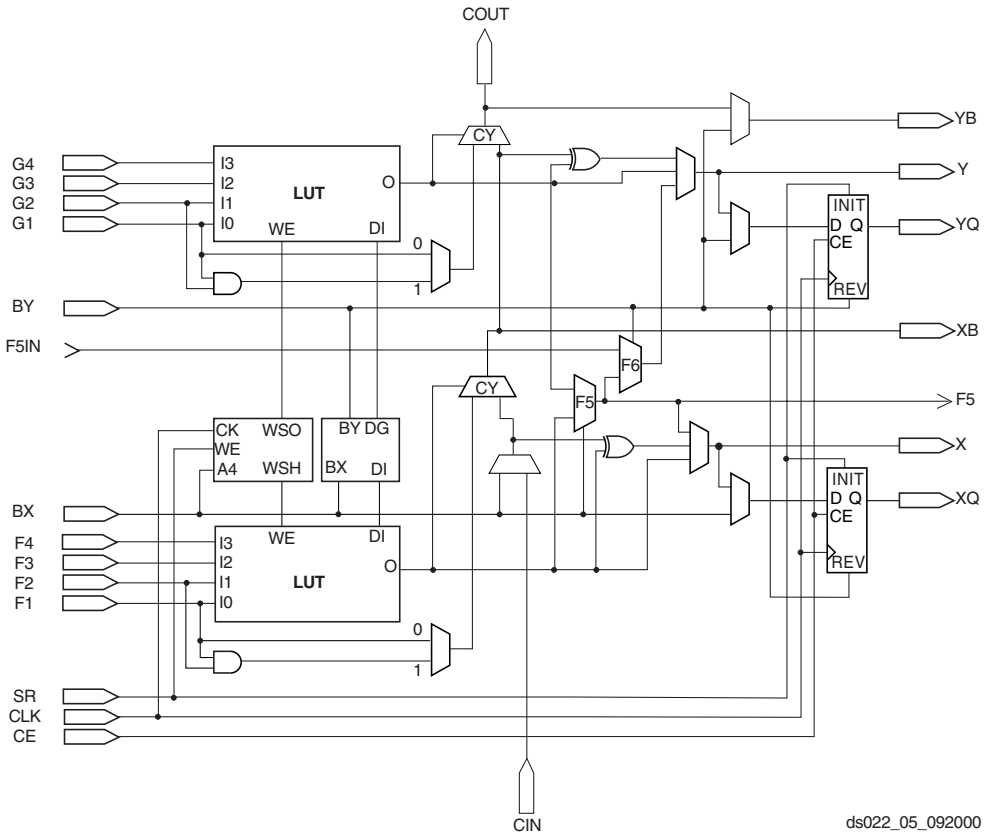

Figure 2: **Detailed View of Virtex-E Slice**

Table 1: **BG560 BGA — XCV405E and XCV812E**

Bank	Pin Description	Pin#
1	IO_L27N_YY	C14
1	IO_L27P_YY	D14
1	IO_L28N_Y	A13
1	IO_L28P_Y	E14
1	IO_L29N_YY	C13
1	IO_VREF_L29P_YY	D13[1]
1	IO_L30N_YY	C12
1	IO_L30P_YY	E13
1	IO_L31N	A11
1	IO_L31P	D12
1	IO_L32N_YY	B11
1	IO_L32P_YY	C11
1	IO_L33N_YY	B10
1	IO_VREF_L33P_YY	D11
1	IO_L34N	C10
1	IO_L34P	A9
1	IO_L35N_YY	C9
1	IO_VREF_L35P_YY	D10[1]
1	IO_L36N_YY	A8
1	IO_L36P_YY	B8
1	IO_L37N_Y	E10
1	IO_L37P_Y	C8
1	IO_L38N_YY	B7
1	IO_VREF_L38P_YY	A6
1	IO_L39N_YY	C7
1	IO_L39P_YY	D8
1	IO_L40N	A5
1	IO_L40P	B5
1	IO_L41N_YY	C6
1	IO_VREF_L41P_YY	D7
1	IO_L42N_YY	A4
1	IO_L42P_YY	B4
1	IO_L43N_Y	C5
1	IO_L43P_Y	E7
1	IO_WRITE_L44N_YY	D6
1	IO_CS_L44P_YY	A2

Table 1: **BG560 BGA — XCV405E and XCV812E**

Bank	Pin Description	Pin#
2	IO	D3
2	IO	F3
2	IO	G1
2	IO	J2
2	IO_DOUT_BUSY_L45P_YY	D4
2	IO_DIN_D0_L45N_YY	E4
2	IO_L46P_Y	F5
2	IO_L46N_Y	B3
2	IO_L47P	F4
2	IO_L47N	C1
2	IO_VREF_L48P_Y	G5
2	IO_L48N_Y	E3
2	IO_L49P_Y	D2
2	IO_L49N_Y	G4
2	IO_L50P_Y	H5
2	IO_L50N_Y	E2
2	IO_VREF_L51P_YY	H4
2	IO_L51N_YY	G3
2	IO_L52P_Y	J5
2	IO_L52N_Y	F1
2	IO_L53P	J4
2	IO_L53N	H3
2	IO_VREF_L54P_YY	K5[1]
2	IO_L54N_YY	H2
2	IO_L55P_Y	J3
2	IO_L55N_Y	K4
2	IO_VREF_L56P_YY	L5
2	IO_D1_L56N_YY	K3
2	IO_D2_L57P_YY	L4
2	IO_L57N_YY	K2
2	IO_L58P_Y	M5
2	IO_L58N_Y	L3
2	IO_L59P	L1
2	IO_L59N	M4
2	IO_VREF_L60P_Y	N5[1]
2	IO_L60N_Y	M2
2	IO_L61P_Y	N4

Table 1: **BG560 BGA — XCV405E and XCV812E**

Bank	Pin Description	Pin#
NA	VCCINT	B18
NA	VCCINT	B28
NA	VCCINT	C22
NA	VCCINT	C24
NA	VCCINT	E9
NA	VCCINT	E12
NA	VCCINT	F2
NA	VCCINT	H30
NA	VCCINT	J1
NA	VCCINT	K32
NA	VCCINT	M3
NA	VCCINT	N1
NA	VCCINT	N29
NA	VCCINT	N33
NA	VCCINT	U5
NA	VCCINT	U30
NA	VCCINT	Y2
NA	VCCINT	Y31
NA	VCCINT	AB2
NA	VCCINT	AB32
NA	VCCINT	AD2
NA	VCCINT	AD32
NA	VCCINT	AG3
NA	VCCINT	AG31
NA	VCCINT	AJ13
NA	VCCINT	AK8
NA	VCCINT	AK11
NA	VCCINT	AK17
NA	VCCINT	AK20
NA	VCCINT	AL14
NA	VCCINT	AL22
NA	VCCINT	AL27
NA	VCCINT	AN25
0	VCCO	A22
0	VCCO	A26
0	VCCO	A30

Table 1: **BG560 BGA — XCV405E and XCV812E**

Bank	Pin Description	Pin#
0	VCCO	B19
0	VCCO	B32
1	VCCO	A10
1	VCCO	A16
1	VCCO	B13
1	VCCO	C3
1	VCCO	E5
2	VCCO	B2
2	VCCO	D1
2	VCCO	H1
2	VCCO	M1
2	VCCO	R2
3	VCCO	V1
3	VCCO	AA2
3	VCCO	AD1
3	VCCO	AK1
3	VCCO	AL2
4	VCCO	AN4
4	VCCO	AN8
4	VCCO	AN12
4	VCCO	AM2
4	VCCO	AM15
5	VCCO	AL31
5	VCCO	AM21
5	VCCO	AN18
5	VCCO	AN24
5	VCCO	AN30
6	VCCO	W32
6	VCCO	AB33
6	VCCO	AF33
6	VCCO	AK33
6	VCCO	AM32
7	VCCO	C32
7	VCCO	D33
7	VCCO	K33
7	VCCO	N32
7	VCCO	T33

Table 2: **BG560 Package Differential Pin Pair Summary XCV405E and XCV812E**

50	2	H5	E2	NA	-
51	2	H4	G3	√	VREF_2
52	2	J5	F1	NA	-
53	2	J4	H3	2	-
54	2	K5	H2	NA	VREF_2
55	2	J3	K4	NA	-
56	2	L5	K3	√	D1
57	2	L4	K2	√	D2
58	2	M5	L3	2	-
59	2	L1	M4	NA	-
60	2	N5	M2	1	VREF_2
61	2	N4	N3	1	-
62	2	N2	P5	NA	-
63	2	P4	P3	√	D3
64	2	P2	R5	2	-
65	2	R4	R3	NA	-
66	2	R1	T4	NA	VREF_2
67	2	T5	T3	NA	-
68	2	T2	U3	√	IRDY
69	3	U1	U2	NA	-
70	3	V2	V4	NA	VREF_3
71	3	V5	V3	NA	-
72	3	W1	W3	2	-
73	3	W4	W5	√	VREF_3
74	3	Y3	Y4	NA	-
75	3	AA1	Y5	1	-
76	3	AA3	AA4	1	VREF_3
77	3	AB3	AA5	NA	-
78	3	AC1	AB4	2	-
79	3	AC3	AB5	√	D5
80	3	AC4	AD3	√	VREF_3
81	3	AE1	AC5	1	-
82	3	AD4	AF1	NA	VREF_3
83	3	AF2	AD5	NA	-

Table 2: **BG560 Package Differential Pin Pair Summary XCV405E and XCV812E**

84	3	AG2	AE4	NA	-
85	3	AH1	AE5	√	VREF_3
86	3	AF4	AJ1	NA	-
87	3	AJ2	AF5	2	-
88	3	AG4	AK2	1	VREF_3
89	3	AJ3	AG5	NA	-
90	3	AL1	AH4	NA	-
91	3	AJ4	AH5	√	INIT
92	4	AL4	AJ6	√	-
93	4	AK5	AN3	NA	-
94	4	AL5	AJ7	√	-
95	4	AM4	AM5	√	VREF_4
96	4	AK7	AL6	1	-
97	4	AM6	AN6	√	-
98	4	AL7	AJ9	√	VREF_4
99	4	AN7	AL8	NA	-
100	4	AM8	AJ10	1	-
101	4	AL9	AM9	1	VREF_4
102	4	AK10	AN9	1	-
103	4	AL10	AM10	√	VREF_4
104	4	AL11	AJ12	√	-
105	4	AN11	AK12	NA	-
106	4	AL12	AM12	√	-
107	4	AK13	AL13	√	VREF_4
108	4	AM13	AN13	1	-
109	4	AJ14	AK14	√	-
110	4	AM14	AN15	√	VREF_4
111	4	AJ15	AK15	NA	-
112	4	AL15	AM16	1	-
113	4	AL16	AJ16	1	VREF_4
114	4	AK16	AN17	1	-
115	5	AM17	AM18	NA	GCLK LVDS 1/0
116	5	AK18	AJ18	1	VREF_5
117	5	AN19	AL19	1	-

Table 3: **FG676 Fine-Pitch BGA — XCV405E**

Bank	Pin Description	Pin #
3	IO_L90P_YY	W20
3	IO_L90N_YY	AC24
3	IO_D7_L91P_YY	AB23
3	IO_INIT_L91N_YY	Y21
4	GCK0	AA14
4	IO	AC18
4	IO	AE20
4	IO	AE23
4	IO	AF21
4	IO_L92P_YY	AC22
4	IO_L92N_YY	AD26
4	IO_L93P_Y	AD23
4	IO_L93N_Y	AA20
4	IO_L94P_YY	Y19
4	IO_L94N_YY	AC21
4	IO_VREF_L95P_YY	AD22
4	IO_L95N_YY	AB20
4	IO_L96P	AE22
4	IO_L96N	Y18
4	IO_L97P	AF22
4	IO_L97N	AA19
4	IO_VREF_L98P_YY	AD21
4	IO_L98N_YY	AB19
4	IO_L99P_YY	AC20
4	IO_L99N_YY	AA18
4	IO_L100P_Y	AC19
4	IO_L100N_Y	AD20
4	IO_L101P_Y	AF20
4	IO_L101N_Y	AB18
4	IO_L102P	AD19
4	IO_L102N	Y17
4	IO_L103P	AE19
4	IO_VREF_L103N	AD18
4	IO_L104P_YY	AF19

Table 3: **FG676 Fine-Pitch BGA — XCV405E**

Bank	Pin Description	Pin #
4	IO_L104N_YY	AA17
4	IO_L105P_Y	AC17
4	IO_L105N_Y	AB17
4	IO_L106P_YY	Y16
4	IO_L106N_YY	AE17
4	IO_L107P_YY	AF17
4	IO_L107N_YY	AA16
4	IO_L108P	AD17
4	IO_L108N	AB16
4	IO_L109P_YY	AC16
4	IO_L109N_YY	AD16
4	IO_VREF_L110P_YY	AC15
4	IO_L110N_YY	Y15
4	IO_L111P_YY	AD15
4	IO_L111N_YY	AA15
4	IO_L112P_Y	W14
4	IO_L112N_Y	AB15
4	IO_VREF_L113P_Y	AF15
4	IO_L113N_Y	Y14
4	IO_L114P	AD14
4	IO_L114N	AB14
4	IO_LVDS_DLL_L115P	AC14
5	GCK1	AB13
5	IO	AD7
5	IO	AD13
5	IO	AE4
5	IO	AE7
5	IO	AF5
5	IO_LVDS_DLL_L115N	AF13
5	IO_L116P_Y	AA13
5	IO_VREF_L116N_Y	AF12
5	IO_L117P_Y	AC13
5	IO_L117N_Y	W13
5	IO_L118P_YY	AA12

Table 3: **FG676 Fine-Pitch BGA — XCV405E**

Bank	Pin Description	Pin #
1	VCCO	H15
2	VCCO	N18
2	VCCO	M19
2	VCCO	M18
2	VCCO	L19
2	VCCO	K19
2	VCCO	J19
3	VCCO	V19
3	VCCO	U19
3	VCCO	T19
3	VCCO	R19
3	VCCO	R18
3	VCCO	P18
4	VCCO	W18
4	VCCO	W17
4	VCCO	W16
4	VCCO	W15
4	VCCO	V15
4	VCCO	V14
5	VCCO	W9
5	VCCO	W12
5	VCCO	W11
5	VCCO	W10
5	VCCO	V13
5	VCCO	V12
6	VCCO	V8
6	VCCO	U8
6	VCCO	T8
6	VCCO	R9
6	VCCO	R8
6	VCCO	P9
7	VCCO	N9
7	VCCO	M9
7	VCCO	M8
7	VCCO	L8

Table 3: **FG676 Fine-Pitch BGA — XCV405E**

Bank	Pin Description	Pin #
7	VCCO	K8
7	VCCO	J8
NA	GND	V25
NA	GND	V2
NA	GND	U17
NA	GND	U16
NA	GND	U15
NA	GND	U14
NA	GND	U13
NA	GND	U12
NA	GND	U11
NA	GND	U10
NA	GND	T17
NA	GND	T16
NA	GND	T15
NA	GND	T14
NA	GND	T13
NA	GND	T12
NA	GND	T11
NA	GND	T10
NA	GND	R17
NA	GND	R16
NA	GND	R15
NA	GND	R14
NA	GND	R13
NA	GND	R12
NA	GND	R11
NA	GND	R10
NA	GND	P25
NA	GND	P17
NA	GND	P16
NA	GND	P15
NA	GND	P14
NA	GND	P13

Table 5: **FG900 Fine-Pitch BGA Package — XCV812E**

Bank	Description	Pin
4	IO_L165N_YY	AE19
4	IO_VREF_4_L166P_YY	AK22
4	IO_L166N_YY	AH20
4	IO_L167P	AG19
4	IO_L167N	AB17
4	IO_L168P	AJ19
4	IO_L168N	AD17
4	IO_L169P_YY	AA16
4	IO_L169N_YY	AA17
4	IO_VREF_4_L170P_YY	AK21
4	IO_L170N_YY	AB16
4	IO_L171P	AG18
4	IO_L171N	AK20
4	IO_L172P	AK19
4	IO_L172N	AD16
4	IO_L173P_YY	AE16
4	IO_L173N_YY	AE17
4	IO_VREF_4_L174P_YY	AG17
4	IO_L174N_YY	AJ17
4	IO_L176P	AG16
4	IO_L176N	AK17
4	IO_LVDS_DLL_L177P	AF16
5	GCK1	AK16
5	IO	AD8
5	IO	AD14
5	IO	AE10
5	IO	AE12
5	IO	AG15
5	IO	AH5
5	IO	AH8
5	IO	AK12
5	IO_LVDS_DLL_L177N	AH16
5	IO_L179P	AB15

Table 5: **FG900 Fine-Pitch BGA Package — XCV812E**

Bank	Description	Pin
5	IO_L179N	AF15
5	IO_L180P_YY	AA15
5	IO_VREF_5_L180N_YY	AF14
5	IO_L181P_YY	AH15
5	IO_L181N_YY	AK15
5	IO_L182P	AB14
5	IO_L182N	AF13
5	IO_L183P	AH14
5	IO_L183N	AJ14
5	IO_L184P_YY	AE14
5	IO_VREF_5_L184N_YY	AG13
5	IO_L185P_YY	AK13
5	IO_L185N_YY	AD13
5	IO_L186P	AE13
5	IO_L186N	AF12
5	IO_L187P	AC13
5	IO_L187N	AA13
5	IO_L188P_YY	AA12
5	IO_VREF_5_L188N_YY	AJ12
5	IO_L189P_YY	AB12
5	IO_L189N_YY	AE11
5	IO_L191P_Y	AG11
5	IO_L191N_Y	AF11
5	IO_L192P_Y	AH11
5	IO_L192N_Y	AJ11
5	IO_L194P_YY	AD12
5	IO_L194N_YY	AK11
5	IO_L195P_YY	AJ10
5	IO_VREF_5_L195N_YY	AC12
5	IO_L196P	AK10
5	IO_L196N	AD11
5	IO_L197P	AJ9
5	IO_L197N	AE9
5	IO_L198P_YY	AH10