University of Belgrade
ETF – School of Electrical Engineering
www.etf.bg.ac.yu

FPGA Comparative Analysis

Author: **Ognjen Šćekić**
ogi@cg.yu

Mentor: **prof. dr Veljko Milutinović**
vm@etf.bg.ac.yu

Belgrade
2005.

Introduction

This paper aims to give the reader basic understanding of currently available FPGA architectures and solutions from leading vendors – Xilinx and Altera.

FPGA vs. ASIC. Applications of FPGAs.

Widely used computer architectures have a fixed central processing unit (CPU) operating on data stored in a memory. Programs determine the sequence of single instructions executed by the CPU. This is a disadvantage for algorithms which can be executed in parallel.

In contrast, FPGA computers have no given processor structure but offer large amounts of logic gates, registers, RAM and routing resources. These can be used for performing logical and arithmetical operations, for variable storage and to transfer data between different parts of the system. Programs do not determine the sequence of execution but the logic structure of the reconfigurable machine. Therefore, algorithms are not only executable in parallel but are executed using a minimum amount of hardware. A single bit operation, for instance, is mapped on a single logical block of an FPGA (typically less than 0.01% of the machine size for currently existing architectures) instead of using about 3% of a complete 32-bit ALU like in a general purpose processor. No register-register transfers are needed to bring operands to the logical element or store the result. Typically thousands of operations can be performed in parallel on an FPGA computer during every clock cycle.

ASICs (Application Specific Integrated Circuits), as the name suggests, are tailor-made on demand for specific applications, rather than intended for general-purpose use. (e.g. a chip designed solely to run a cell phone is an ASIC). In fact, often the same individual or company that designed the chip is the end user and the device is not available commercially.

It is clear that this way of designing and producing circuits is extremely expensive and time-consuming, but inevitable for certain high-end applications. However, for smaller designs and/or lower production volumes, ASICs have started to become a less attractive solution, as FPGAs grow larger, faster and more capable.

Many companies nowadays use FPGAs during the early design phase and preproduction phases and then switch later to ASIC for volume production. For applications whose future commercial success is unknown the FPGA route offers lower risk.

Table below presents a qualitative comparison between FPGAs and ASICs:

CHARACTERISTIC	FPGA	ASIC
Time-to-market	Short	Long
High volume unit cost	High	Low
Flexibility after manufacturing	High	None
Performance	Medium	Very high
Density	Medium	Very high
Power consumption	High	Low
Minimum order quantities	None	High
Design flow complexity	Medium	Very high
Complexity of test	Low	High
Turnaround Time	Hours	Months

Figure 1 – ASIC and FPGA comparison

The following table lists common application areas for FPGAs today:

End Markets	Subsegments	Application
Communications	Wireless	Cellular Base Stations
		Wireless LAN
	Networking	Metro Area Networks
		Optical Networks
		DSL Modems
		Switches
		Routers
		Mass Storage
Storage		Storage Area Networks
		Network Attached Storage
		High Speed Servers
		Computer Peripherals
		Mass Storage
Office Automation		Copiers, printers
Consumer, Industrial And Other	Consumers	Plasma Displays
		DVRs
		Set Top Boxes
		MP3 Players
		Digital Cameras
	Industrial	Factory Automation
		Medical Imaging
		Test Equipment
	Automotive	Multimedia Systems
		GPS Navigation Systems
		Voice Recognition
	Military	Satellite Surveillance
		Radar and Sonar System
		Secure Communication

Figure 2 – Major FPGA application areas

Market overview

As we will see later in the text, the competition in the FPGA market is bitter. The reason becomes more than obvious when we look at the current market figures of the key players: ***Xilinx***, ***Altera***, ***Lattice*** and ***Actel***.

Xilinx, the industry leader in FPGAs, estimates the total logic market at $57 billion composed of ASIC $14.0B, FPGA $2.8B, other PLD (Programmable Logic Devices) at $0.5B and other general purpose logic at $8.5 B.

Obviously, both Xilinx and Altera would like to find ways to expand the market for FPGAs. There are two general approaches:

- First approach is to lower the per-unit-production-cost of FPGA to attack the low-end market.
- Second approach is to increase its capability to compete at the high end for high-performance DSP (Digital Signal Processing units), high-speed I/O, embedded processing and next generation applications.

Unit production costs can be lowered by using 90 nm technology and 300 mm wafers. Costs can also be lowered by designing FPGAs with less processing power, less memory, smaller feature sets, etc.

Table below shows total revenues for key industrial players in 2003 and first two quarters of 2004 expressed in millions of dollars.

	1Q03	2Q03	3Q03	4Q03	1Q04	2Q04
Xilinx	306	313	316	366	403	424
Altera	195	205	209	217	242	268
Lattice	57	56	43	53	59	61
Actel	34	37	38	40	42	44
TOTAL	**592**	**611**	**606**	**676**	**746**	**797**

Figure 3 – Revenues of PLD manufacturers in 2003 and first half of 2004

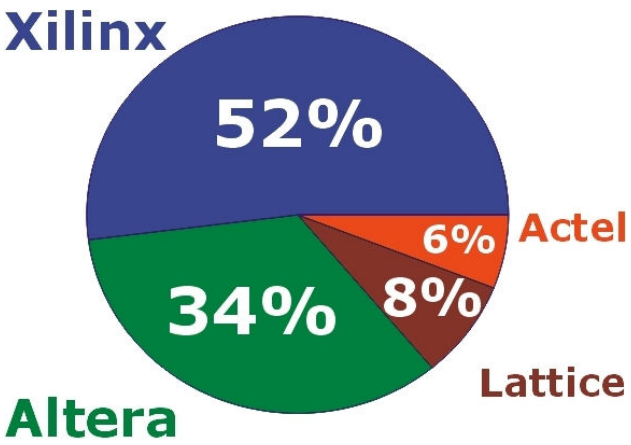


Figure 4 - PLD market share

What is even more important is the fact that FPGAs constitute ever bigger part of the total revenue, and its share is rapidly growing – Altera announced that FPGA accounted for 69% of total revenue while growing 73% year over year.

Xilinx – (pronounced "zylinks") was founded in 1984 and shipped its first commercial product in 1985. Today Xilinx employs around 2,600 people. Xilinx claims more than 7,500 customers worldwide and more than 50,000 design starts. This amounts to more than half the world demand for FPGAs. Xilinx partners with leading semiconductor manufacturers such as IBM Microelectronics, UMC (United Microelectronics Corporation) and Seiko. Xilinx is the net market leader (at the moment at least).

Altera – founded in 1983. The company developed the first reprogrammable logic device (PLD) in 1984. Altera expanded its technology leadership in 1988 with the MAX architecture and, in 1992, with the look-up table (LUT)-based FLEX architecture. It recently strengthened even further with the introduction of newer, more powerful and efficient architectures, the Quartus II development system, and an extensive IP offering. Altera's strategic partner is TSMC.

Both companies offer an extensive range of FPGAs, with approximately equivalent capabilities, both for low-end and for high-end market, as well as complete software development kits, along with many IP blocks ("intellectual property") for specific needs of other companies.

Recent FPGA design timeline

Xilinx offers its **Virtex** family at the high end and **Spartan** at the low end.
Altera offers **Stratix** at the high end and **Cyclone** at the low end.

	Altera	Xilinx
high-end FPGA family	Stratix	Virtex
low-end FPGA family	Cyclone	Spartan

	Altera	Xilinx
1997	APEX	
1998		Virtex
2000		Spartan II
2001	APEX II	Virtex II
2002	Stratix & Cyclone	Virtex II Pro
2003		Spartan-3
2004	Stratix II & Cyclone II	Virtex-4

Figure 5 – FPGA design timeline

Virtex and Stratix families are direct opponents, as are Spartan and Cyclone, respectively.

This paper will cover the design basics of Cyclone II, Stratix II, Spartan-3 and Virtex-4 families, comparing the solutions offered by the two companies.

Key factors for comparing FPGAs

Before we can proceed with comparing actual FPGAs, it is important to visualize key factors that distinguish one FPGA from another:

- **Fabrication process**
- **Logic density**
- **Clock management**
- **On-chip memory**
- **DSP capabilities**
- **I/O compatibility**
- **Software support & other design services**

Fabrication process

It is clear that more advanced fabrication process brings higher integration, and thus higher density and/or reduced size of chips.

The Xilinx-IBM partnership resulted in **90nm** technology (previously 0.13µm) first used in Spartan-3, and later in Virtex-4 FPGA family. It gave Xilinx a competitive edge on the high-end FPGA market in terms of density, because it took their main opponent Altera-TSMC another year to achieve the same technology. In fact, it was only in 2004 that Altera released its first 90nm FPGAs – Stratix II and Cyclone II, allowing Xilinx a year of unchallenged market expansion.

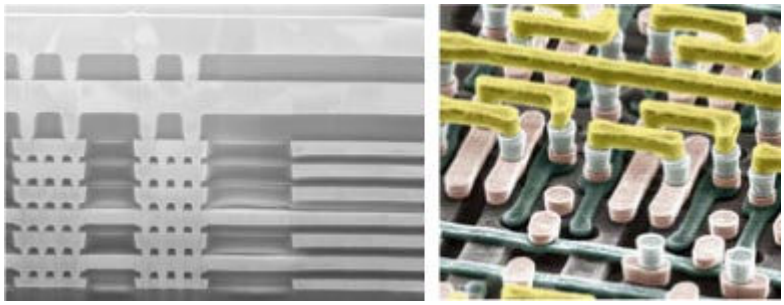


Figure 6 - Part of Cyclone II 90nm structure

Logic density

Although the term "density" is generally closely related to the fabrication process, used here it refers to "logic density" rather than physical density. As we can see from **Insert 1**, the terminology used for expressing the logic density of FPGAs is quite confusing. The point is, however, clear: we need a unit to express the logic capability of our FPGA. The problem is how to define this unit. By introducing new features into a logic block, its functionality increases, and cannot be easily expressed in terms of LCs.

The best example of this is Altera's ALM structure: depending on the selected configuration each ALM can act as if it had 1 or 2 LUTs, with 3,4,5,6 or 7 inputs. We will explain this later. Lacking a universal unit we still use LEs and ELCs to express the density of FPGAs, but in order for these figures to have a proper meaning we must describe the way the basic logic block of our FPGA works.

Insert 1 – logic structures naming

For the logic structure consisting of a 4-input look-up table (LUT), a D-flip-flop and some additional circuitry Xilinx uses the term **LC – Logical Cell**.
Altera uses the term **LE – Logical Element**.

These structures used to be main building blocks in early designs and were used to express the complexity of FPGA structure. As FPGA architecture became more and more complex, manufacturers started calling their main building blocks differently.

These "new" building blocks typically contain more than one LUT, more than one D-FF, and a mix of combinational, arithmetic, and register logic. The improved functionality of these "new" blocks earned them a new name. However, since they do contain the same elements as a simple LC (or LE), both manufacturers still list the equivalent number of LCs(LEs) as an important attribute in their datasheets.

Altera still uses the term LE for describing its Cyclone II family, but they have adopted a new term **ALM – Adaptive Logic Module** for describing Stratix II family. **(1 ALM = 2.5 LEs)**

Xilinx uses the term **CLB – Configurable Logic Block** to name the basic logic block of all its FPGAs. Each CLB has **8** LCs. But since these 8 LCs provide a grater functionality than if they were separate, Xilinx now uses the unit **ELC – Equivalent Logic Cell (1 ELC = 1.125 LC)** to state the complexity of its FPGAs.
To make it all more complicated, Xilinx introduced the term **ASMBL – Advanced Silicon Modular Block** (pronounced like "assemble") to describe the new feature-rich architecture of their Virtex-4 building blocks.

Clock management

Clock management comprises two basic functions:

- Remove clock skew[1] and propagation delay
- Generate new clock signals with different frequencies/phases

Removing clock skew and propagation delay

All parts of a digital circuit need to be synchronized to a desired clock signal. If the circuit is large, complex, and operating at high frequencies, the clock propagation delay and clock skew have a great impact on its performance.

Therefore, providing a clock signal with zero-delay in all parts of an FPGA becomes crucial.

Generally, this can be done using either **DLL**s – **Delay Locked Loops**, or **PLL**s – **Phase Locked Loops**.

Both of these 2 types of circuits yield the same result – they compensate for the delay generated on the routing network inside the FPGA, providing zero-delay clock signal (with respect to a user source clock) to different parts of FPGA.

They only differ in method they use to achieve this:

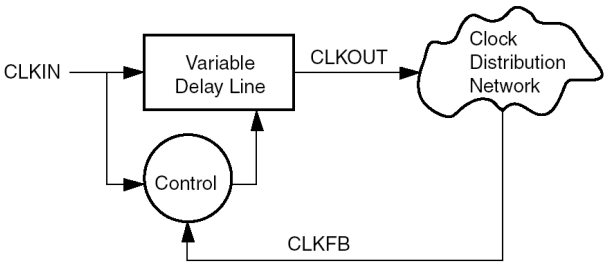


Figure 7a - DLL block diagram

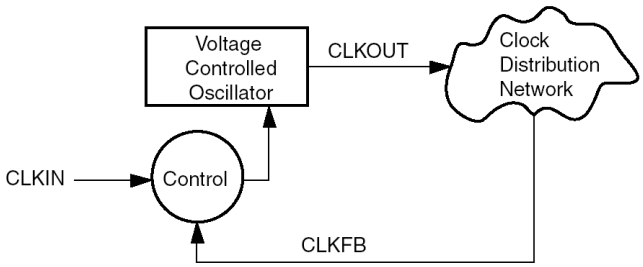


Figure 7b - PLL block diagram

A DLL in its simplest form consists of a variable delay-line and control logic. The delay-line produces a delayed version of the input clock CLKIN. The clock distribution network routes the clock to all internal registers and to the clock feedback CLKFB pin. The control logic must sample the input clock as well as the feedback clock in order to adjust the delay line. The delay-line consists on an array of delay elements, typically CMOS voltage-controlled inverters connected in series.
DLL works by inserting delay between the input clock and the feedback clock until the two rising edges align, putting the two clocks 360° out of phase (meaning they are in phase). After the edges from the input clock line up with the edges from the feedback clock, the DLL "locks". Thus, the DLL output clock compensates for the delay in the clock distribution network.

A PLL uses a different architecture to accomplish the same task. The fundamental difference between the PLL and DLL is that instead of a delay line, the PLL uses a voltage controlled oscillator which generates a clock signal that approximates the input clock CLKIN. The control logic, consisting of a phase detector and filter, adjusts the oscillator frequency and phase to compensate for the clock distribution delay.

[1] Clock skew is a phenomenon in synchronous circuits in which the clock signal (sent from the clock source) arrives at different components in a circuit at different times, due to different distances it has to travel.

The PLL control logic compares the input clock to the feedback clock CLKFB and adjusts the oscillator clock until the rising edge of the input clock aligns with the feedback clock. The PLL then "locks."

What are the advantages and drawbacks?
The oscillator used in the PLL introduces instability and an accumulation of phase error. This in turn degrades the performance of the PLL when attempting to compensate for the delay of the clock distribution network, whereas the unconditionally stable DLL architecture does not accumulate phase error. On the other hand, the PLL typically has an advantage when it comes to frequency synthesis.

As we will see later, Altera uses PLLs and Xilinx uses DLLs.

Clock generation and phase shifting

Since the whole point of FPGAs lies in their configurability, having the option to make different parts (called **clock domains**) of the same FPGA work at different frequencies dramatically simplifies the design, at the same time improving the performance.

Clock skew elimination is only the basic feature of DLL and PLL circuits. In addition to this, they both act as source clock frequency multipliers/dividers, duty-cycle regulators as well as phase shifters.

Clock multiplication gives the designer a number of design alternatives. For instance, a 50 MHz source clock multiplied 4X by the DLL/PLL can drive an FPGA design operating at 200 MHz. This technique can simplify board design because the clock path on the board no longer distributes such a high-speed signal.

A multiplied clock also provides designers the option of time-domain multiplexing – e.g. using one circuit twice per clock cycle, consuming less area than two copies of the same circuit.

The DLL/PLL can also act as a clock mirror. By driving the DLL/PLL output out of chip and then back in again, the DLL/PLL can be used to de-skew a board level clock between multiple devices.

High-end FPGAs have sophisticated clock managers which provide additional features, and are more resistant to temperature and voltage variations.

On-chip memory

As FPGA applications grow in complexity so does their need for memory. Using Look-Up Tables as registers for storing data couldn't possibly provide enough space for serious applications. Especially if these applications require numerous arithmetical computations to be performed, and are time dependent. As this is often the case, the outside memory could not produce desired efficiency. This is why, with every new generation of FPGAs, more and more memory gets embedded into FPGA.
The main advantages of embedded (built-in) memory are:

- Short access time
- High bandwidth
- Great versatility

The first two points are self-explanatory. The third point means that the embedded memory can behave like various memory forms, and implement some of the most commonly used memory functions, including:

- ✓ RAM (synchronous/asynchronous)
- ✓ ROM
- ✓ FIFO
- ✓ Buffers
- ✓ Cache
- ✓ Shift registers
- ✓ etc…

DSP capabilities

As we can see from Fig. 2, the majority of FPGA applications require some sort of **Digital Signal Processing (DSP).** DSP requires many computations to take place in short periods of time. In order to reduce the time these computations take, and to increase efficiency, computations are executed in parallel (pipelining).
FPGAs are ideal for implementing this pipeline mode of DSP, thanks to their adaptable structure.
FPGA manufacturers have over years developed special DSP units to help designers fully exploit the FPGA possibilities. These units are designed to optimize execution of most commonly used DSP algorithms (filtering, compression, encoding/decoding, equalization, digital conversion, FFT, modulation, etc.) They usually contain a great number of multipliers (in parallel), accumulators, shift registers, adders…

I/O compatibility

As FPGAs continue to grow in size and capacity, the larger and more complex systems designed for them demand an increased variety of **I/O standards**. Furthermore, as system-clock speeds continue to increase, the need for high-performance I/O becomes more important.

Modern bus applications, pioneered by the largest and most influential companies in the digital electronics industry, are commonly introduced with a new I/O standard tailored specifically to the needs of that application. The bus I/O standards provide specifications to other vendors who create products designed to interface with these applications.
Each standard often has its own specifications for current, voltage, I/O buffering, and termination techniques.

If an I/O standard implementation requires two I/O pins (differential input/output) it is called **differential**. If it requires only one pin, it is called **single-ended**. Some standards need additional reference-voltage to be supplied.

Insert 2 – most commonly used I/O standards

LVTTL— Low-Voltage TTL - A general purpose standard that uses an LVTTL input buffer and a Push-Pull output buffer. This standard requires an output source voltage (V_{CCO}), but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

LVCMOS — Low-Voltage CMOS - A general purpose standard. This standard requires an output source voltage (V_{CCO}), but does not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}).

PCI — Peripheral Component Interface - This standard specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses a LVTTL input buffer and a push-pull output buffer. This standard does not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}), however, it does require a 3.3V output source voltage (V_{CCO}).

AGP — Advanced Graphics Port - A 3.3V standard used for graphics applications. This standard requires a Push-Pull output buffer and a Differential Amplifier input buffer.

GTL — Gunning Transceiver Logic - A high-speed bus standard invented by Xerox. This standard requires a differential amplifier input buffer and an open-drain output buffer.

SSTL — Stub Series Terminated Logic – A general purpose memory bus standard sponsored by Hitachi and IBM This standard has two classes, I and II. This standard requires a Differential Amplifier input buffer and an Push-Pull output buffer.

HSTL — High-Speed Transceiver Logic – A general purpose high-speed, 1.5V bus standard sponsored by IBM. This standard has four variations or classes. This standard requires a Differential Amplifier input buffer and a Push-Pull output buffer.

Interfaces for these and other I/O standards are implemented in **I/O blocks** - parts of FPGA internal architecture positioned peripherally and connected to I/O pins and to internal interconnects.

An I/O block usually contains: programmable input and output buffers, D-FFs, pull-up and/or pull-down resistors, a delay array, bus-hold (keeper) circuit, etc.

Buffers are configurable, so they could adjust to various standards (in some standards, the user needs to supply reference-voltage). D-flipflops are used as optional delay elements. Pull-up and pull-down resistors are used to assert or de-assert a pin that would otherwise be left floating. Bus-hold (keeper) circuit keeps the last known logic levels on a bus if all other devices connected to the bus are in high-Z state. Circuitry for protection from electrostatic charges is also usually found in I/O blocks.

I/O blocks are divided into **banks**. A bank is a group of neighboring pins which use the same or compatible I/O standard. This division is useful when implementing several I/O standards at the same time. In this way different banks can be configured to use different standards.

Software support & other design services

Developing an FPGA-based hardware system is a complex process. Different manufacturers divide this process into different stages, providing a complete software solution for each of them.
The coarsest of these divisions lists 3 stages:

- System design & synthesis
- Design implementation
- On-chip verification

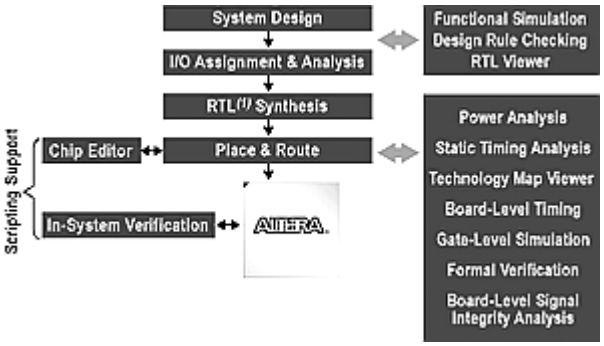


Figure 8a - Altera FPGA design flow diagram

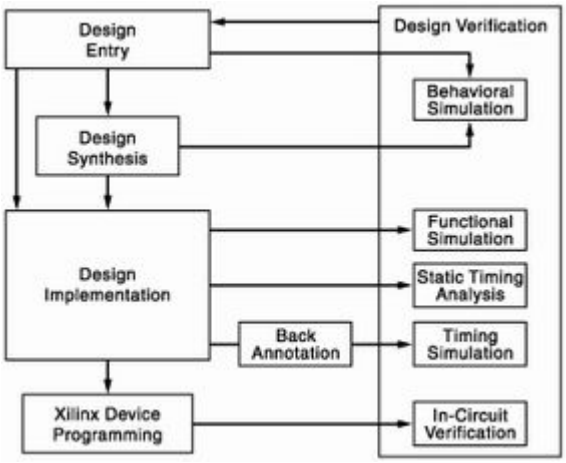


Figure 8b - Xilinx FPGA design flow diagram

System design stage

System design stage begins with design entry, which can be done either using a **HDL - hardware description language** (*VHDL* or *Verilog*), or a schematic editor, or, in majority of cases, using a combination of these two. Software solutions offer complete integrated environments, with tools for guiding users through each step of this process. A wide variety of FPGA-ready component libraries are available, ranging from simple processors, peripheral components, controllers, down to general logic (gates, counters, decoders, etc).

These design environments support 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.
This software also provides several analysis and simulation tools for early-stage debugging.

Once the hardware design is complete it is **synthesized** – a process that transforms it from HDL form into a low-level gate form called **RTL - Register transfer level description.**

The system design stage is platform independent, meaning that the resulting RTL description of our system can be fitted into any FPGA. This is why companies other than FPGA manufacturers develop their own software environments hoping their more user-friendly interface, and better built-in tools would attract customers. A good example of this is *Synplify Pro* tool, from Synplicity Inc. – a third-party developer given the best ratings in 2005 by "FPGA and Structured ASIC Journal".

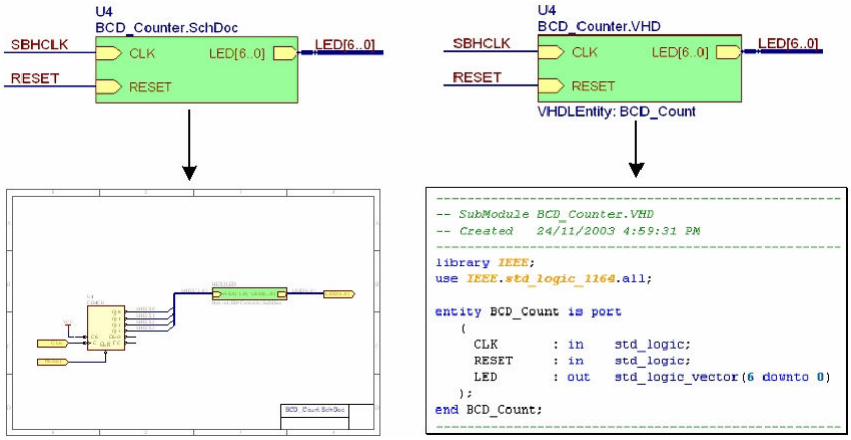


Figure 9 - HDL and schematic representation of a BCD counter

Design implementation

The design implementation stage is commonly called the **Place-And-Route** stage. The Place-And-Route tools take the input RTL netlist for the design and map the logic into the architectural resources of the FPGA (LCs and I/O blocks, for example). Then, the best location for these blocks is found, based on their interconnections and desired performance. Finally, the interconnects are routed, and pins assigned.

This stage is clearly platform-dependent, since our design is implemented in an actual FPGA architecture. Therefore, place-and-route tools are developed by the FPGA manufacturers. The place-and-route tools are developed to take full advantage of FPGA architecture, and to provide optimum performance for a given design. Many analysis and simulation tools are provided for this stage as well.

The result of this stage is a **configuration file** which is loaded into FPGA at startup.

On-chip verification

The last stage – On-chip verification, is executed once the design has been loaded into the FPGA. It gives developer the possibility for real-world debugging. Special cables are supplied with FPGA development kits by the vendors, for connecting FPGAs to a PC or a workstation. This way, we can read the contents of built-in memory, observe the states of flipflops, and monitor our entire system.

Xilinx as well as Altera provide integrated software development tools.

Altera's development tool is called ***Quartus II***, and that of Xilinx is called ***ISE***.

In this paper we will not discuss the features of these development tools.

It is worth noticing that both Altera and Xilinx provide **"intellectual property" blocks**, i.e. complete design of a complex system, written in a hardware description language, optimized to run on their FPGAs, (like microcontrollers, microprocessors, etc.)

Fig. 10 shows the block diagram of Altera's embedded soft-core RISC 16/32-bit processors ***Nios/NiosII***, written completely in HDL, and available within Quartus software:

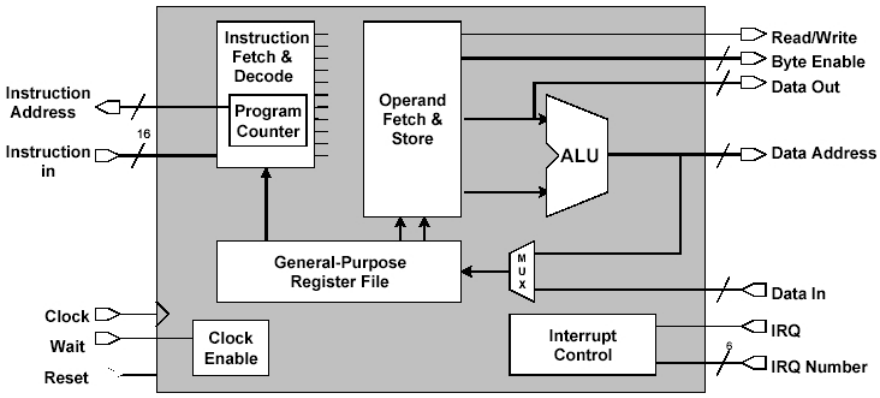


Figure 10 - Altera's embedded soft-core Nios processor

Xilinx offers its own IP microprocessors: 8-bit ***PicoBlaze***, and 32-bit ***MicroBlaze***.

List of available intellectual property blocks can be found at manufacturers' websites.

Another very important issue has to be addressed here:

When FPGA based designs move in volume production the main issue is cost reduction. The greatest difference between Xilinx and Altera is the approach the two vendors take to reduce costs.

Xilinx offers a service called ***EasyPath*** which helps customer develop a design that will run faster and generate a greater yield for the FPGA. Once the clients have developed their system on FPGA, they send it over to Xilinx. After 8 weeks they get back the optimized FPGAs with exactly the same functionality. These optimized FPGAs are 30%-80% less expensive when mass produced, and they represent **replacements** for structured ASICs, and take less time to be completed.

Selection Criteria	Structured ASICs*	EasyPath FPGAs
Time to Prototype Samples	4 – 8 weeks	0 weeks
Total Time to Volume Production	12 – 15 weeks	8 weeks
Vendor NRE/Mask Costs	$100K – $200K	$75K
Design Costs for Conversion	$250K – $300K	$0
Additional Cost of Tools for Conversion	$100K – $200K	$0
Unit Costs	Low	Low
Risk	High	Low
Flexibility to Make Changes In-System	Inflexible	Flexible
Design Conversion from Prototype to Production	Additional Engineering	Conversion Free

** Xilinx market analysis*

Figure 11 - EasyPath advantages over ASIC migration (source Xilinx)

Altera offers a service called ***HardCopy***. It is a **migration path** from FPGA to structured ASIC. Much like "traditional" way of passing to ASIC.
However, Altera developed a fine-grained cell structure (*HCells*) ASICs which perfectly match the LEs of Altera's FPGAs. That way Stratix logic elements (LEs) are mapped to equivalent logic elements in the corresponding HardCopy device. Quartus software maps the utilized portion of each Stratix FPGA. If a Stratix LE is not used in the FPGA design, then it is not mapped to the HardCopy device, yielding a more efficient mapping of the prototyped design.

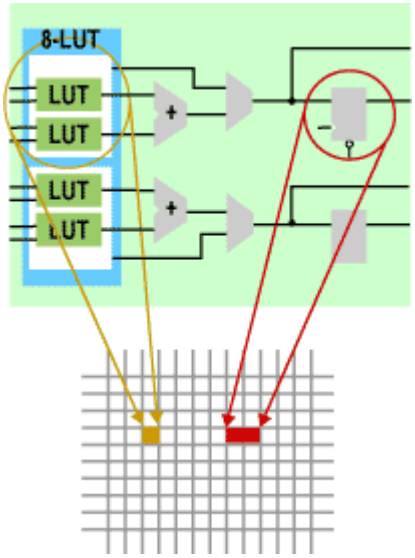


Figure 12 - Stratix II ALM mapping to HCells in a HardCopy II device

FPGA Overviews & Comparisons

NOTE: More detailed overview will be given for Cyclone II and Spartan-3 families only. Stratix II and Virtex-4 features will only be listed, without getting into details.

Cyclone II family

Cyclone II is the most recent Altera's low-end FPGA family, introduced in 2004, but first shipped in February 2005.

Cyclone II FPGA family is based on a 1.2V, 90nm process with densities over 68K logic elements (LEs) [2] and up to 1.1 Mbits of embedded RAM, with features like embedded 18 × 18 multipliers to support high-performance DSP applications, phase-locked loops (PLLs) for system clock management, and high-speed external memory interface support for SRAM and DRAM devices, Cyclone II devices support differential and single-ended I/O standards, including 64-bit, 66-MHz PCI and PCI-X for interfacing with processors and ASIC devices. The Cyclone II FPGA family offers commercial grade and industrial grade devices[3].

Table 1–1. Cyclone II FPGA Family Features

Feature	EP2C5	EP2C8	EP2C20	EP2C35	EP2C50	EP2C70
LEs	4,608	8,256	18,752	33,216	50,528	68,416
M4K RAM blocks (4 Kbits plus 512 parity bits	26	36	52	105	129	250
Total RAM bits	119,808	165,888	239,616	483,840	594,432	1,152,000
Embedded multipliers *(1)*	13	18	26	35	86	150
PLLs	2	2	4	4	4	4
Maximum user I/O pins	158	182	315	475	450	622

Note to Table 1–1:
(1) This is the total number of 18 × 18 multipliers. For the total number of 9 × 9 multipliers per device, multiply the total number of 18 × 18 multipliers by 2.

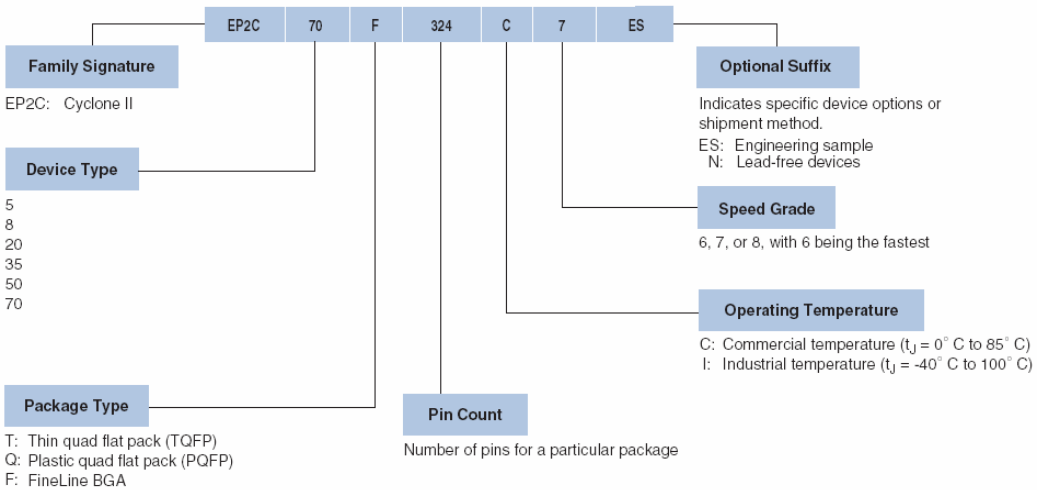


Figure 13 - Cyclone II family features (high) and packaging information (low)

[2] To avoid confusion about the logic structure naming conventions used by Xilinx and Altera see Insert 1. For the complete list of all abbreviations used see page 47 – Abbreviations.
[3] Commercial grade (range): Tc = 0°C to +85°C Industrial grade (range): Ti = -40°C to +100°C

Functional description

Cyclone II devices contain a two-dimensional row- and column-based architecture to implement custom logic. Column and row interconnects of varying speeds provide signal interconnects between **logic array blocks (LABs)**, embedded memory blocks, and embedded multipliers.
The logic array consists of LABs, with 16 logic elements (LEs) in each LAB. LABs are grouped into rows and columns across the device. Cyclone II devices range in density from 4,608 to 68,416 LEs.
Cyclone II devices provide a global clock network and up to four phase-locked-loops (PLLs). The global clock network consists of up to 16 global clock lines that drive throughout the entire device. The global clock network can provide clocks for all resources within the device, such as **input/output elements (IOEs)**, LEs, embedded multipliers, and embedded memory blocks. The global clock lines can also be used for other high fan-out signals.

Cyclone II PLLs provide general-purpose clocking with clock synthesis and phase shifting as well as external outputs for high-speed differential I/O support.

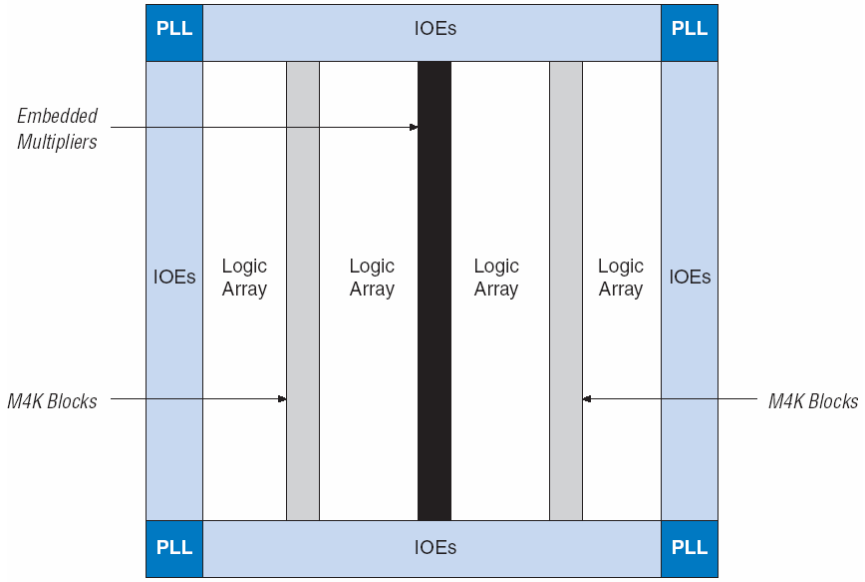


Figure 14 - Cyclone II floorplan

M4K memory blocks are true dual-port memory blocks with 4K bits of memory plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual-port, or single-port memory up to 36-bits wide at up to 260 MHz. These blocks are arranged in columns across the device in between certain LABs. Cyclone II devices offer between 119 to 1,152 Kbits of embedded memory.

Each embedded multiplier block can implement either two 9 × 9-bit multipliers, or one 18 × 18-bit multiplier with up to 250-MHz performance. Embedded multipliers are arranged in columns across the device.

Each Cyclone II device I/O pin is fed by an IOE located at the ends of LAB rows and columns around the periphery of the device. I/O pins support various single-ended and differential I/O standards, such as the 66- and 33-MHz, 64- and 32-bit PCI standard, PCI-X, and the LVDS I/O standard at a maximum data rate of 805 megabits per second (Mbps) for inputs and 640 Mbps for outputs. Each IOE contains a bidirectional I/O buffer and three registers for registering input, output, and output-enable signals.

LE units

The smallest unit of logic in the Cyclone II architecture, the LE, is compact and provides advanced features with efficient logic utilization. Each LE features:

- ✓ Four-input look-up table (LUT), which is a function generator that can implement any function of four variables
- ✓ Programmable register
- ✓ Carry-chain connection
- ✓ Register-chain connection
- ✓ Ability to drive all types of interconnects: local, row, column, register chain, and direct link interconnects

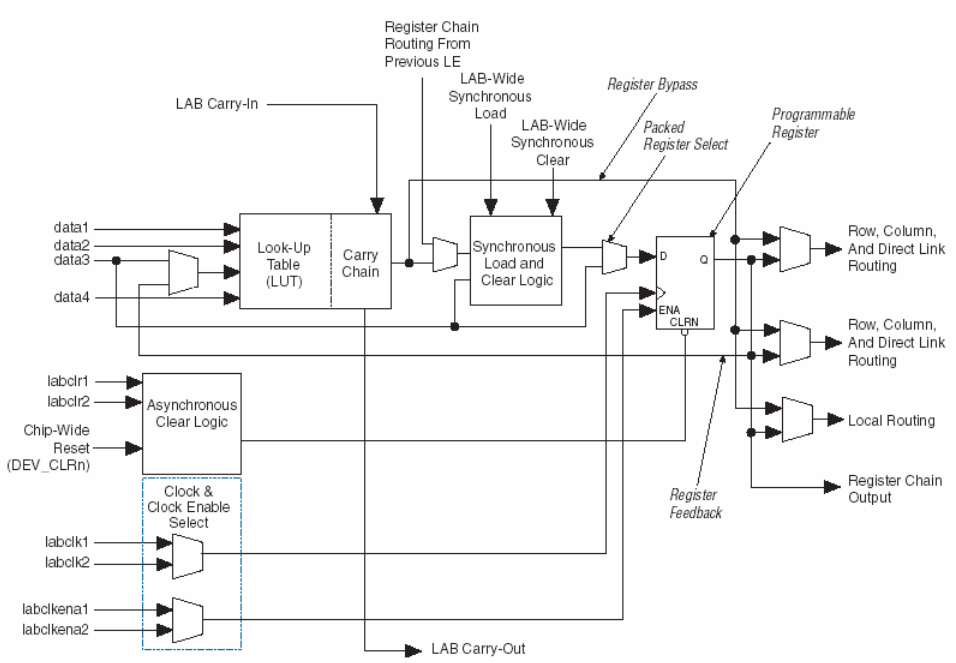


Figure 15 - Cyclone II logical element (LE)

Each LE's programmable register can be configured for D, T, JK, or SR 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 register's clock and clear control signals. Either general-purpose I/O pins or internal logic can drive the clock enable. For combinational functions, the LUT output bypasses the register and drives directly to the LE outputs.

In addition to the three general routing outputs, the LEs within an LAB have register chain outputs. Register chain outputs allow registers within the same LAB to cascade together. The register chain output allows an LAB to use LUTs for a single combinational function and the registers to be used for an unrelated shift register implementation. These resources speed up connections between LABs while saving local interconnect resources.

The Cyclone II LE operates in one of the following modes:

- Normal mode
- Arithmetic mode

Each mode uses LE resources differently. The Quartus II software automatically chooses the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions. If required, you can also create special-purpose functions that specify which LE operating mode to use for optimal performance.

The normal mode is suitable for general logic applications and combinational functions. In normal mode, four data inputs from the LAB local interconnect are inputs to a four-input LUT.

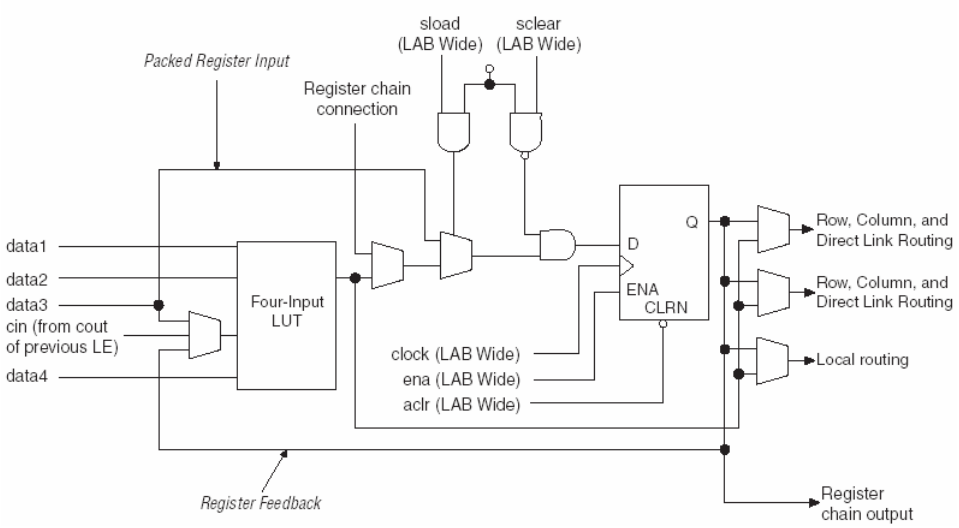


Figure 16 - Cyclone II LE in Normal mode

The arithmetic mode is ideal for implementing adders, counters, accumulators, and comparators. An LE in arithmetic mode implements a 2-bit full adder and basic carry chain.

The Quartus II Compiler automatically creates carry chain logic during design processing, or you can create it manually during design entry. It also creates carry chains longer than 16 LEs by automatically linking LABs in the same column if necessary.

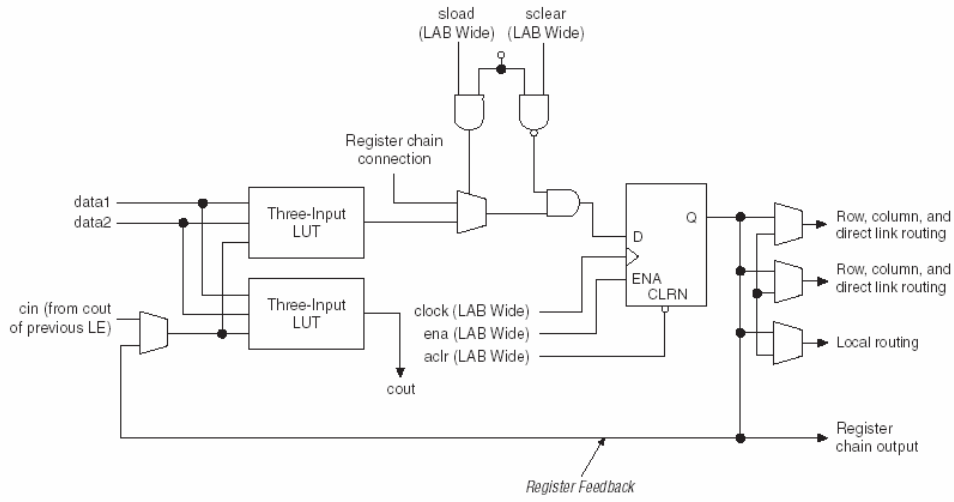


Figure 17 - Cyclone II LE in Arithmetic mode

Logic Array Blocks (LAB) and Interconnects

Each LAB consists of the following:

- ✓ 16 LEs
- ✓ LAB control signals
- ✓ LE carry chains
- ✓ Register chains
- ✓ Local interconnect

The local interconnect transfers signals between LEs in the same LAB. Register chain connections transfer the output of one LE's register to the adjacent LE's register within an LAB. The Quartus II Compiler places associated logic within an LAB or adjacent LABs, allowing the use of local, and register chain connections for performance and area efficiency.

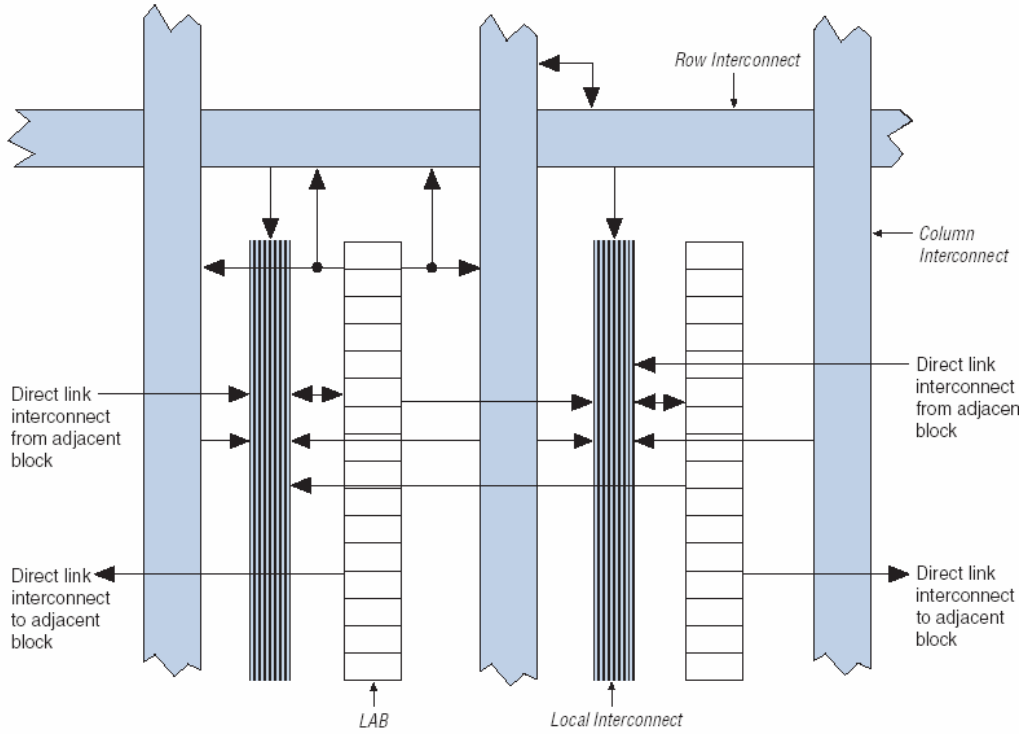


Figure 18 - Cyclone II LAB structure

The LAB local interconnect connects LEs within the same LAB. The LAB local interconnect is driven by column and row interconnects and LE outputs within the same LAB. Neighboring LABs, PLLs, M4K RAM blocks, and embedded multipliers from the left and right can also drive an LAB's local interconnect through the direct link connection. Direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility.

Row interconnects route signals to and from LABs, PLLs, M4K memory blocks, and embedded multipliers within the same row.
These row resources include:

- ✓ Direct link interconnects between LABs and adjacent blocks
- ✓ R4 interconnects traversing four blocks to the right or left
- ✓ R24 interconnects for high-speed access across the length of the device

Column interconnect operates similar to the row interconnect. Each column of LABs is served by a dedicated column interconnect, which vertically routes signals to and from LABs, M4K memory blocks, embedded multipliers, and row and column IOEs.

These column resources include:

- ✓ Register chain interconnects within an LAB
- ✓ C4 interconnects traversing a distance of four blocks in an up and down direction
- ✓ C16 interconnects for high-speed vertical routing through the device

Clock management

Cyclone II devices provide global clock networks and up to four PLLs for a complete clock management solution. Cyclone II clock network features include:

- ✓ Up to 16 global clock networks
- ✓ Up to four PLLs
- ✓ Global clock network dynamic clock source selection
- ✓ Clock network dynamic enable and disable

Sixteen or eight (depends on device) global clock networks drive throughout the entire device. Dedicated clock pins, PLL outputs, the logic array, and dual-purpose clock pins can also drive the global clock network. The global clock network can provide clocks for all resources within the device, such as IOEs, LEs, memory blocks, and embedded multipliers. The global clock lines can also be used for control signals, such as clock enables and synchronous or asynchronous clears fed from the external pin. Internal logic can also drive the global clock network for internally generated global clocks and asynchronous clears, clock enables, or other control signals with large fan-out.

There is a clock control block for each global clock network available in Cyclone II devices. The clock control blocks are arranged on the device periphery. Clock control blocks are used to control global clock networks – to select a global clock network source and to enable/disable it.

Multiplexers are used with these clocks to form six-bit buses to drive column IOE clocks, LAB row clocks, or row IOE clocks. Another multiplexer at the LAB level selects two of the six LAB row clocks to feed the LE registers within the LAB.

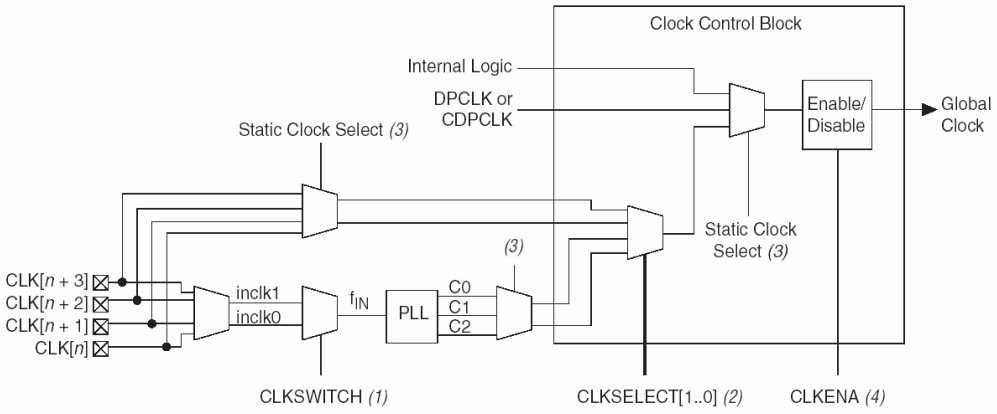


Figure 19 - Global clock network control block

Cyclone II PLLs provide general-purpose clocking as well as support for the following features:

- ✓ Clock multiplication and division
  Ranges form x(1/128) up to x32

- ✓ Phase shifting
  Programmable phase shifts in increments of at least 45°.

- ✓ Programmable duty cycle
  Generate clock outputs with a variable duty cycle.

- ✓ Manual clock switchover
  Enables you to switch between two reference input clocks for applications that may require support for clocks with two different frequencies.

- ✓ etc.

Embedded memory

The Cyclone II embedded memory consists of columns of M4K memory blocks. The M4K memory blocks include input registers that synchronize writes and output registers to pipeline designs and improve system performance. The output registers can be bypassed, but input registers cannot.

Each M4K block can implement various types of memory with or without parity, including true dual-port, simple dual-port, and single-port RAM, ROM, and first-in first-out (FIFO) buffers. The M4K blocks support the following features:

- ✓ 4,608 RAM bits (including parity bits – one for each byte)
- ✓ 250-MHz performance
- ✓ True dual-port memory
  Supports any combination of two-port operations: two reads, two writes, or one read and one write at two different clock frequencies.
- ✓ Simple dual-port memory
  Simultaneous reads and writes are supported.
- ✓ Single-port memory
  Simultaneous reads and writes are not allowed.
- ✓ Shift register
  Memory blocks are used to implement shift registers. Data is written into each address location at the falling edge of the clock and read from the address at the rising edge of the clock.
- ✓ FIFO buffer
  Memory blocks are used to implement FIFO buffers. Simultaneous read and write from an empty FIFO buffer is not supported.
- ✓ ROM
  When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.
- ✓ Byte enable
  The "byte enable" allows the input data to be masked so the device can write to specific bytes. The unwritten bytes retain the previous written value.
- ✓ Address clock enable
  Used to hold the previous address value for as long as the signal is enabled. This feature is useful in handling misses in cache applications.
- ✓ Content Addressable memory (CAM) ⇔ Associative memory
- ✓ etc.

Embedded multipliers

Cyclone II devices have embedded multiplier blocks optimized for multiplier-intensive digital signal processing (DSP) functions, such as finite impulse response (FIR) filters, fast Fourier transform (FFT) functions, and discrete cosine transform (DCT) functions. You can use the embedded multiplier in one of two basic operational modes, depending on the application needs:

- ✓ One 18-bit multiplier
- ✓ Up to two independent 9-bit multipliers

Embedded multipliers can operate at up to 250 MHz.

Device	Embedded Multiplier Columns	Embedded Multipliers
EP2C5	1	13
EP2C8	1	18
EP2C20	1	26
EP2C35	1	35
EP2C50	2	86
EP2C70	3	150

Figure 20 - Multiplier count in Cyclone II family

Each Cyclone II device has one to three columns of embedded multipliers that efficiently implement multiplication functions. An embedded multiplier spans the height of one LAB row.

The embedded multiplier consists of the following elements:

- ✓ Multiplier block
- ✓ Input and output registers
- ✓ Input and output interfaces

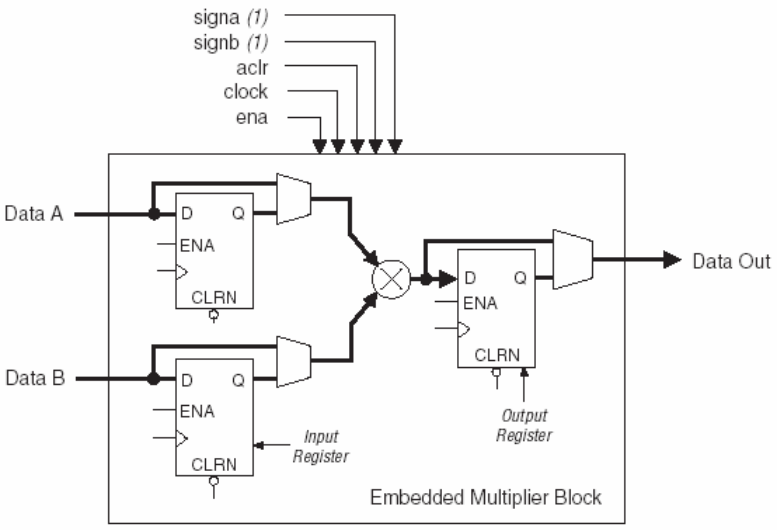


Figure 21 - Multiplier block architecture

Each multiplier operand can be a unique signed or unsigned number. Two signals, *signa* and *signb*, control the representation of each operand respectively. A logic "1" value on the *signa* signal indicates that data A is a signed number while a logic "0" value indicates an unsigned number. The result of the multiplication is signed if any one of the operands is a signed value.

An embedded multiplier can be configured to support a single 18 × 18 multiplier for operand widths up to 18 bits or two independent 9 × 9 multipliers.

I/O structure and features

The IOEs are located in I/O blocks around the periphery of the Cyclone II device. There are up to five IOEs per row I/O block and up to four IOEs per column I/O block.

IOEs support many features, including:

- ✓ Leading differential and single-ended I/O standards
- ✓ Weak pull-up resistors during configuration
- ✓ Programmable pull-up resistors in user mode
- ✓ 3-state buffers
- ✓ Programmable input and output delays
- ✓ Bus-hold circuitry
- ✓ Joint Test Action Group (JTAG) boundary-scan test (BST) support

Cyclone II device IOEs contain a bidirectional I/O buffer and three registers for complete embedded bidirectional single data rate transfer. The IOE contains one input register, one output register, and one output enable register. You can use IOEs as input, output, or bidirectional pins.

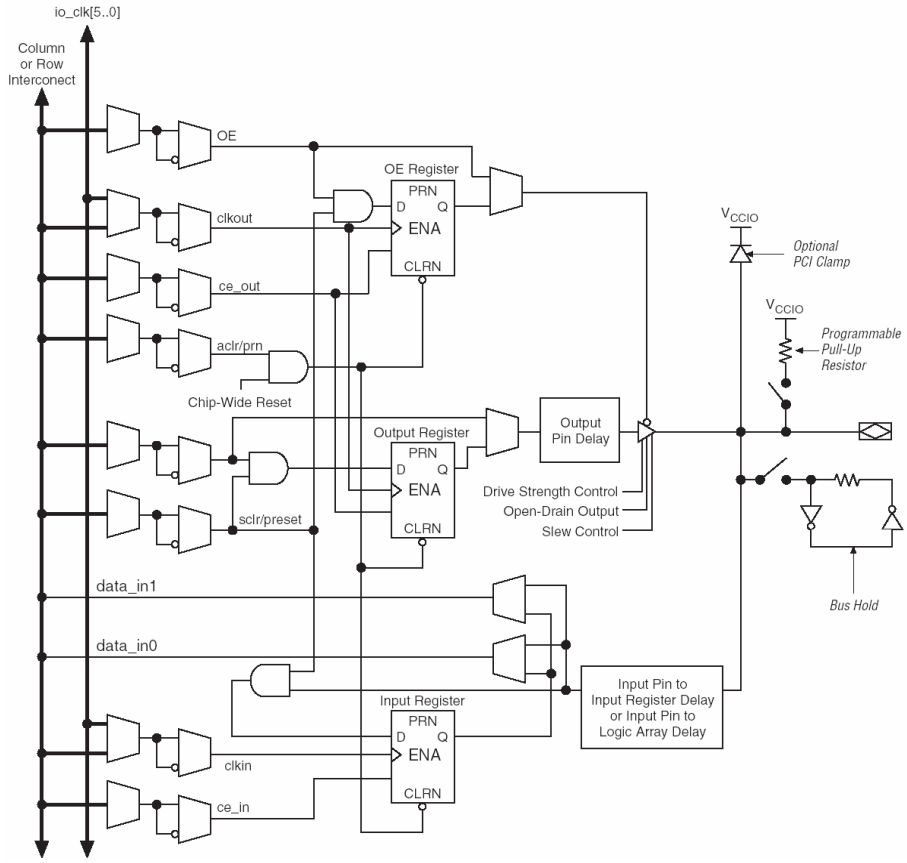


Figure 22 - Cyclone II I/O element (IOE)

There are two paths available for combinational or registered inputs to the logic array. Each path contains a unique programmable delay chain, shown in Fig. 22.
Programmable delays are used to ensure zero hold times and minimize setup times. The Quartus II Compiler can program these delays to automatically minimize setup time while providing a zero hold time.

The IOE registers in each I/O block share the same source for clear or preset. You can program preset or clear for each individual IOE, but both features cannot be used simultaneously. You can also program the registers to power up high or low after configuration is complete.

Cyclone II devices support a broad range of external memory interfaces such as SDR SDRAM, DDR SDRAM, DDR2 SDRAM, and QDRII SRAM external memories. Cyclone II devices feature dedicated high-speed interfaces that transfer data between external memory devices at up to 167 MHz/333 Mbps for DDR and DDR2 SDRAM devices and 167 MHz/667 Mbps for QDRII SRAM devices.

IOEs support most conventional and high-speed I/O protocols:

- ✓ LVTTL (3.3V, 2.5V, 1.8V)
- ✓ LVCMOS (3.3V, 2.5V, 1.8V, 1.5V)
- ✓ SSTL (classes I, II) and differential
- ✓ HSTL (classes I, II) and differential
- ✓ PCI and PCI-X
- ✓ etc.

The I/O pins on Cyclone II devices are grouped together into I/O banks and each bank has a separate power bus. EP2C5 and EP2C8 devices have four I/O banks, while EP2C20, EP2C35, EP2C50, and EP2C70 devices have eight I/O banks. Each device I/O pin is associated with one I/O bank. To accommodate voltage-referenced I/O standards, each Cyclone II I/O bank has a VREF bus.

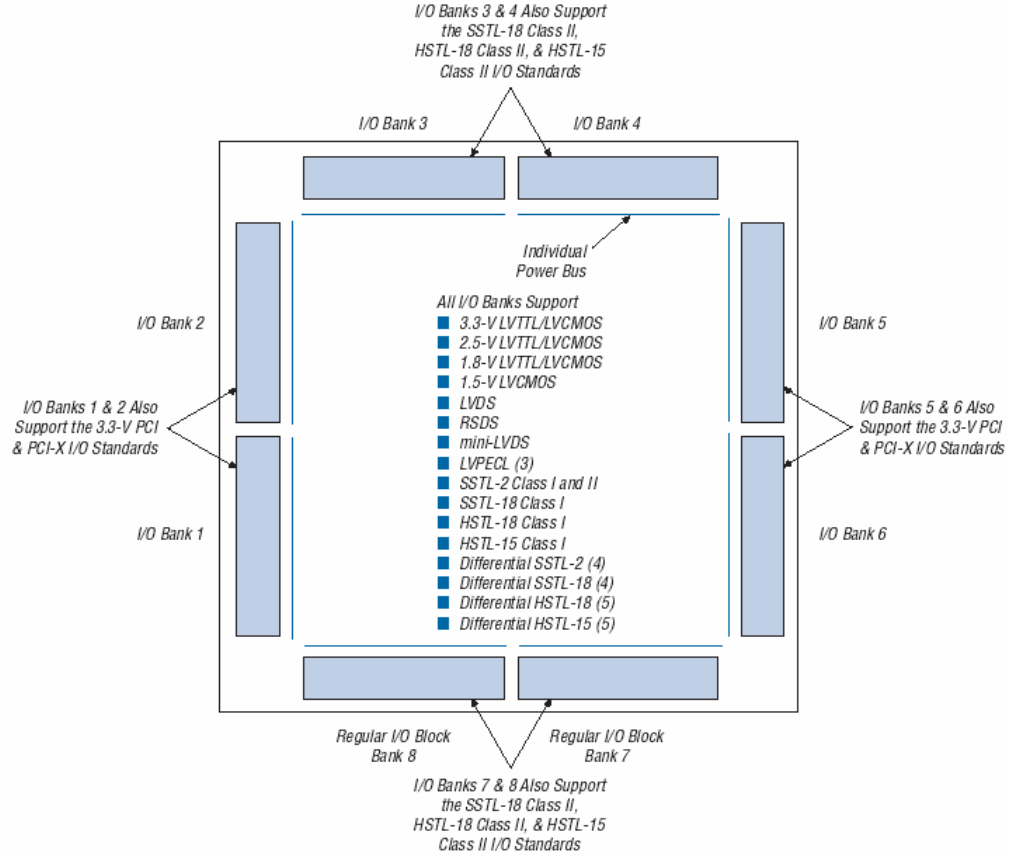


Figure 23 - Cyclone II I/O banks

Each I/O bank can support multiple standards with the same V_{CCIO} for input and output pins. For example, when V_{CCIO} is 3.3-V, a bank can support LVTTL, LVCMOS, and 3.3-V PCI for inputs and outputs. Voltage referenced standards can be supported in an I/O bank using any number of single-ended or differential standards as long as they use the same V_{REF} and a compatible V_{CCIO} value.
The Cyclone II architecture supports the *MultiVolt* I/O interface feature, which allows Cyclone II devices in all packages to interface with systems of different supply voltages. They have one set of V_{CC} pins (V_{CCINT}) that power the internal device logic array.

Start-up Configuration

The logic, circuitry, and interconnects are configured with CMOS SRAM elements that require configuration data to be loaded each time the circuit powers up. The process of physically loading the SRAM data into the device is called **configuration**. During initialization, which occurs immediately after configuration, the device resets registers, enables I/O pins, and begins to operate as a logic device. Together, the configuration and initialization processes are called **command mode**. Normal device operation is called **user mode**.

Cyclone II devices can be configured automatically at system power-up with data stored in an Altera **configuration device** or provided by a system controller. The Cyclone II device's optimized interface allows it to act as controller in an Active serial (AS) configuration scheme.

In addition to these low-cost configuration devices, Cyclone II devices can be configured in real-time via a serial data stream using the Passive serial (PS) configuration mode. The PS interface also enables microprocessors to treat Cyclone II devices as memory and configure them by writing to a virtual memory location, simplifying reconfiguration.

After a Cyclone II device has been configured, it can be reconfigured in-circuit by resetting the device and loading new configuration data. With real-time reconfiguration, the device is forced into command mode with the nCONFIG pin. The configuration process loads different configuration data, reinitializes the device, and resumes user-mode operation. These real-time changes that can be made during system operation enable creation of innovative **reconfigurable systems**[4].

The configuration pins support 1.5-V/1.8-V or 2.5-V/3.3-V I/O standards. The voltage level of the configuration output pins is determined by the V_{CCIO} of the bank where the pins reside.

Configuration Scheme	Data Source
Active serial (AS)	Low-cost serial configuration device
Passive serial (PS)	Enhanced or EPC2 configuration device, MasterBlaster, ByteBlasterMV, ByteBlaster II or USB Blaster download cable, or serial data source
JTAG	MasterBlaster, ByteBlasterMV, ByteBlaster II or USB Blaster download cable or a microprocessor with a Jam or JBC file

Figure 24 – Cyclone II configuration schemes

[4] Reconfigurable systems are seen today as a way future systems might look like. The idea is to have a system which could reconfigure itself to best fit the problem at hand. Instead of working with fixed "blocks" of computing power, scientists will have flexible "clay" that can be molded to perfectly fit their codes. Extensive research is being done in this direction.

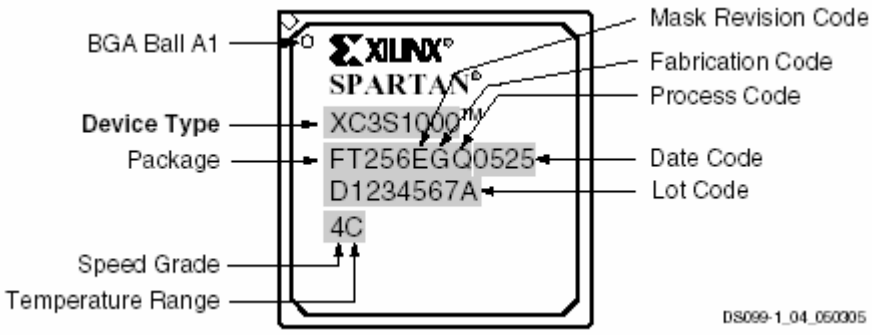
Spartan-3 family

Spartan-3 was first announced in April 2003. Its latest version is called Spartan-3E family.

Spartan-3 family is based on a **90nm,** eight layer metal process, with densities up to 74K logic cells (LCs) and up to 1.8 Mbits of embedded RAM, with dedicated 18 × 18 multipliers, up to 4 digital clock managers (DCM), and high-speed, versatile external RAM interface. Spartan-3 devices support differential and single-ended I/O standards. A great range of commercial grade and industrial grade devices is offered.

Device	System Gates	Equivalent Logic Cells[1]	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Maximum User I/O
			Rows	Columns	Total CLBs					
XC3S50[2]	50K	1,728	16	12	192	12K	72K	4	2	124
XC3S200[2]	200K	4,320	24	20	480	30K	216K	12	4	173
XC3S400[2]	400K	8,064	32	28	896	56K	288K	16	4	264
XC3S1000[2, 3]	1M	17,280	48	40	1,920	120K	432K	24	4	391
XC3S1500[3]	1.5M	29,952	64	52	3,328	208K	576K	32	4	487
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565
XC3S4000[3]	4M	62,208	96	72	6,912	432K	1,728K	96	4	712
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784

Figure 25 - Spartan-3 family features



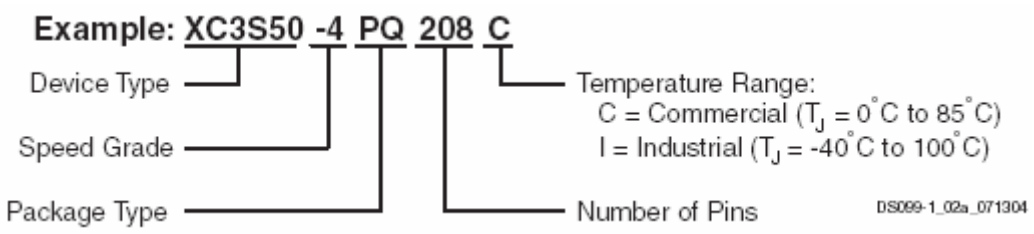


Figure 26 - Spartan-3 family packaging information

Functional description

The Spartan-3 family architecture consists of five fundamental programmable functional elements:

- Configurable Logic Blocks (CLBs)
  Contain RAM-based Look-Up Tables (LUTs) to implement logic and storage elements that can be used as flip-flops or latches. CLBs can be programmed to perform a wide variety of logical functions as well as to store data.

- Digital Clock Manager (DCM) blocks
  Provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase shifting clock signals.

- Block RAM
  Provides data storage in the form of 18-Kbit dual-port blocks.

- Multiplier blocks
  Accept two 18-bit binary numbers as inputs and calculate the product.

- Input/Output Blocks (IOBs)
  Control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow, plus 3-state operation. Twenty-four different I/O standards, including seven high-performance differential standards, are available. Double Data-Rate (DDR) registers are included. The Digitally Controlled Impedance (DCI) feature provides automatic on-chip terminations, simplifying board designs.

These elements are organized as shown below. A ring of IOBs surrounds a regular array of CLBs. The XC3S50 has a single column of block RAM embedded in the array. Those devices ranging from the XC3S200 to the XC3S2000 have two columns of block RAM. The XC3S4000 and XC3S5000 devices have four RAM columns. Each column is made up of several 18-Kbit RAM blocks; each block is associated with a dedicated multiplier. The DCMs are positioned at the ends of the outer block RAM columns. The Spartan-3 family features a rich network of switches that interconnect all five functional elements, transmitting signals among them. Each functional element has an associated switch matrix that permits multiple connections to the routing.

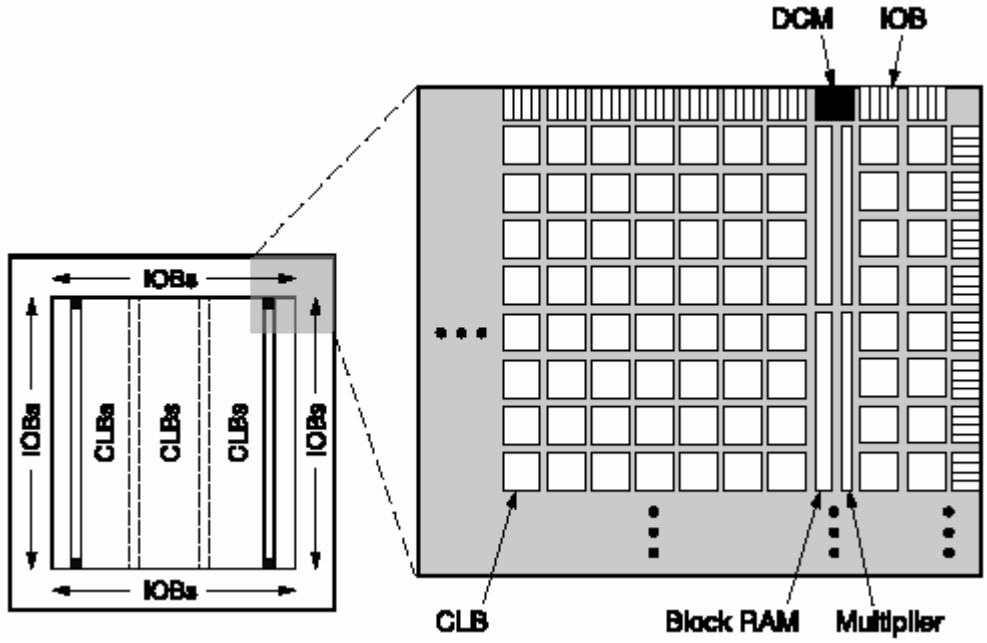


Figure 27 - Spartan-3 floorplan

Spartan-3 devices offer embedded *XtremeDSP* functionality with dedicated 18x18 multipliers and up to 330 billion multiply and accumulates (MACs) per second.

The Spartan-3 FPGA memory architecture provides efficient area utilization:

Each CLB **4-input LUT** (Look-Up Table) works as a 16-bit fast, compact shift register. LUTs can be cascaded to build longer shift registers, or implement pipeline registers and buffers. LUTs can also work as a single-port or dual-port RAM/ROM, and can be cascaded to build larger memories, FIFOs and buffers.

Embedded block-RAM (up to 1.87 Mbits) is made up of synchronous, cascadable 18 Kb blocks.

Digital Clock Managers deliver sophisticated digital clock management impervious to system jitter, temperature and voltage variations.

CLB overview

The Configurable Logic Blocks (CLBs) constitute the main logic resource for implementing synchronous as well as combinatorial circuits. Each CLB comprises four interconnected **slices**, as shown below. These slices are grouped in pairs. Each pair is organized as a column with an independent carry chain.

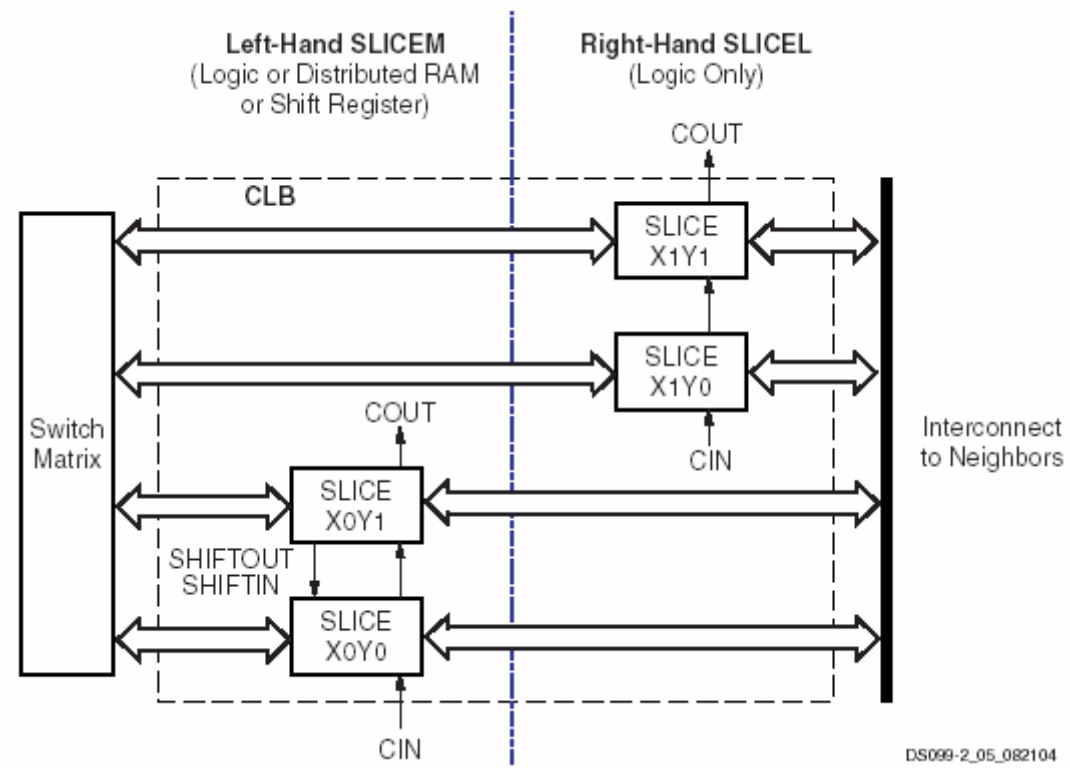


Figure 28 - Spartan-3 CLB structure

All four slices have the following elements in common:

- ✓ 2 logic function generators (4-input LUTs)
- ✓ 2 storage elements
- ✓ Wide-function multiplexers
- ✓ Carry logic
- ✓ Arithmetic gates

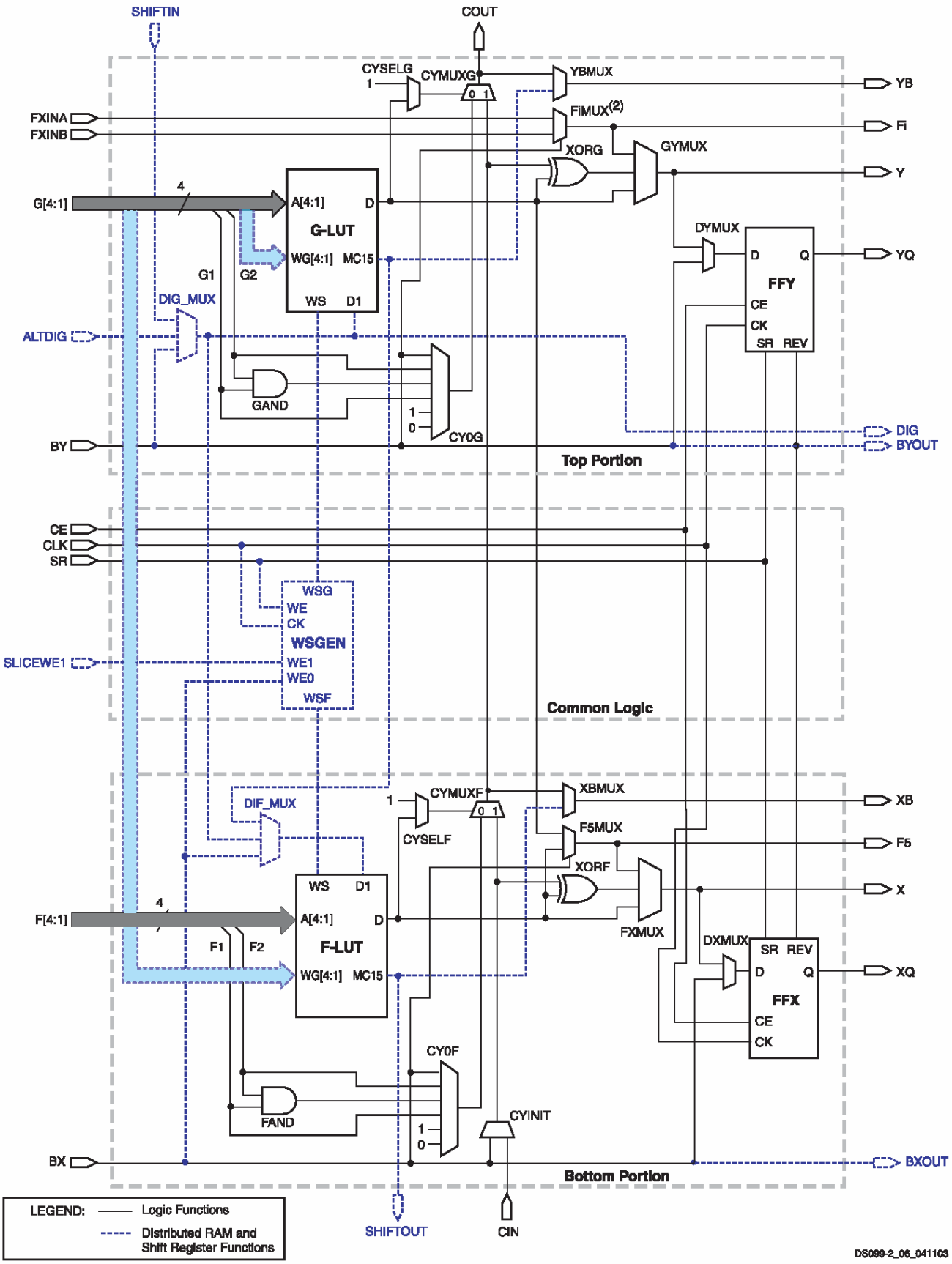


Figure 29 - Left-hand slice of Spartan-3 CLB

Both the left-hand and the right-hand slice pairs use these elements to provide logic, arithmetic, and ROM functions. The function generators located in upper and lower portions of the slice are referred to as the "G" and "F", respectively.

Each of the two LUTs (F and G) in a slice has four logic inputs (A1-A4) and a single output (D). This permits any four-variable Boolean logic operation to be programmed into them.

The **carry chain**, together with various dedicated arithmetic logic gates, support fast and efficient implementations of math operations. The carry chain enters the slice as CIN and exits as COUT. Five multiplexers control the chain: CYINIT, CY0F, and CYMUXF in the lower portion as well as CY0G and CYMUXG in the upper portion. The dedicated arithmetic logic includes the exclusive-OR gates XORG and XORF (upper and lower portions of the slice, respectively) as well as AND gates - GAND and FAND (upper and lower portions, respectively).

The LUTs not only support the logic functions described above, but also can function as ROM that is initialized with data at the time of configuration, as well as distributed RAM. This type of memory affords moderate amounts of data buffering anywhere along a data path. One left-hand LUT stores 16 bits.

The **storage element**, which is programmable as either a D-type flip-flop or a level-sensitive latch, provides a means for synchronizing data to a clock signal, among other uses. The storage elements in the upper and lower portions of the slice are called FFY and FFX, respectively.

Wide-function multiplexers effectively combine LUTs in order to permit more complex logic operations. Each slice has two of these multiplexers with F5MUX in the lower portion of the slice and F_iMUX (i=6,7,8) in the upper portion.

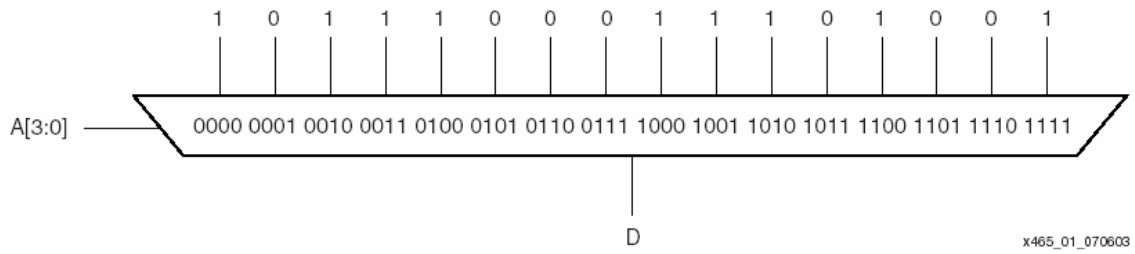


Figure 30 - LUT modeled as a multiplexer when configured to work as a logic function generator

Beside already mentioned functions, the left-hand pair of slices supports two additional functions:

- ✓ Storing data using LUTs
- ✓ Shifting data with LUTs configured as 16-bit registers

Each LUT in left-hand CLBs can be configured as a 16-bit **shift register** without using the flip-flops available in each slice. Shift operations are synchronous with the clock, and output length is dynamically selectable. A separate dedicated output allows the cascading of any number of 16-bit shift registers to create whatever size shift register is needed. Each CLB resource can be configured using four of the eight LUTs as a 64-bit shift register.

When configured to work as a shift register the LUT can be described as a 16:1 multiplexer with the four inputs serving as binary select lines, and the values programmed into the Look-Up Table serving as the data being selected.
Bits are shifted from either LSB to MSB or MSB to LSB according to application.

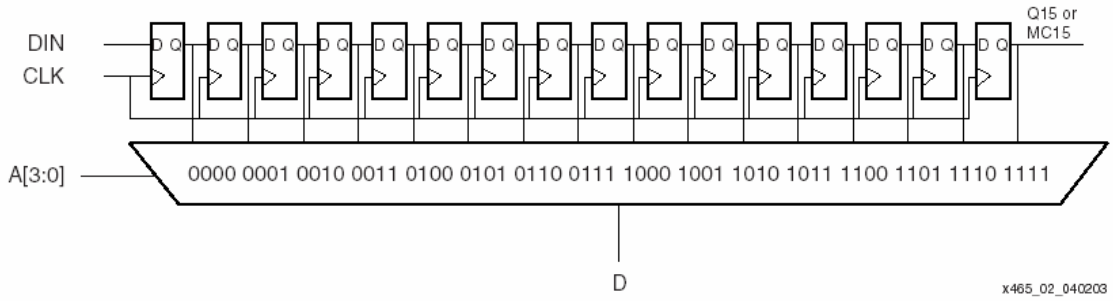


Figure 31 - LUT configured as a 16-bit shift register

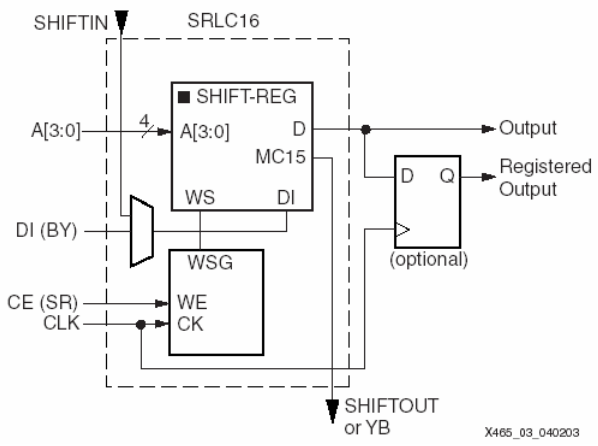


Figure 32a - Logic cell (LC) with LUT as shift register and storage element

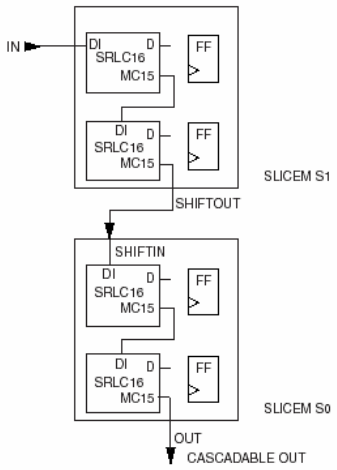


Figure 32b - Cascading LUTs to form 64-bit shift registers

Interconnects

Interconnects pass signals among various functional elements of Spartan-3 devices. There are four kinds of interconnects:

- ✓ Long lines
  Connect every sixth CLB in a row/column. Because of their low capacitance, these lines are well-suited for carrying high-frequency signals with minimal skew. They can serve as replacements for global clock lines as well.
- ✓ Hex lines
  Connect every third CLB in a row/column.
- ✓ Double lines
  Connect every other CLB in a row/column.
- ✓ Direct lines
  Afford any CLB direct access to neighboring CLBs.

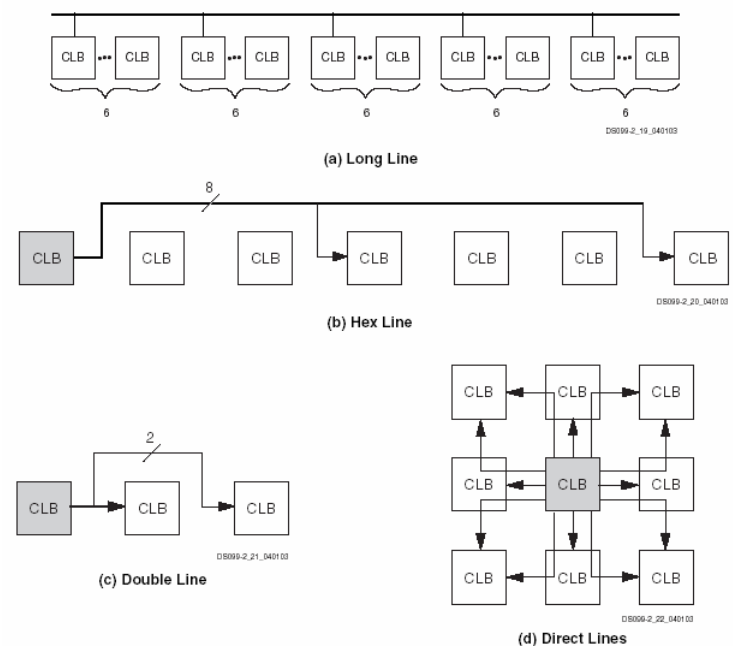


Figure 33 - Types of interconnects in Spartan-3 family

Clock management

Spartan-3 devices provide flexible, complete control over clock frequency, phase shift and clock-skew through the use of the **Digital Clock Manager (DCM)** feature. To accomplish this, the DCM employs a Delay-Locked Loop (DLL), a fully digital control system that uses feedback to maintain clock signal characteristics with a high degree of precision despite normal variations in operating temperature and voltage.

The DCM supports three major functions:

- ✓ Clock-skew elimination
- ✓ Frequency synthesis
- ✓ Phase shifting

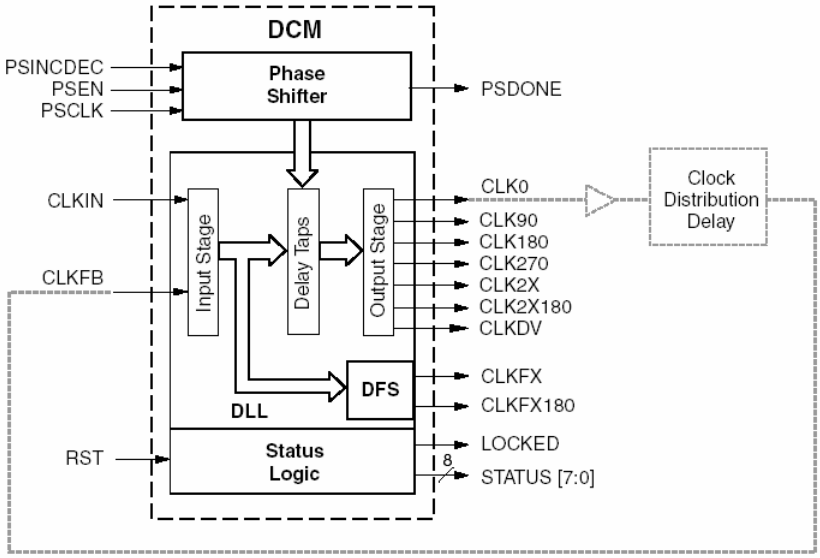


Figure 34 - DCM functional blocks and signals

The DCM has four functional components:

- ✓ Delay-locked loop (DLL)
- ✓ Digital frequency synthesizer (DFS)
- ✓ Phase shifter (PS)
- ✓ Status logic

One of the basic functions of a DLL component is to eliminate clock skew. The main signal path of the DLL consists of an input stage, followed by a series of discrete delay elements, which in turn leads to an output stage.

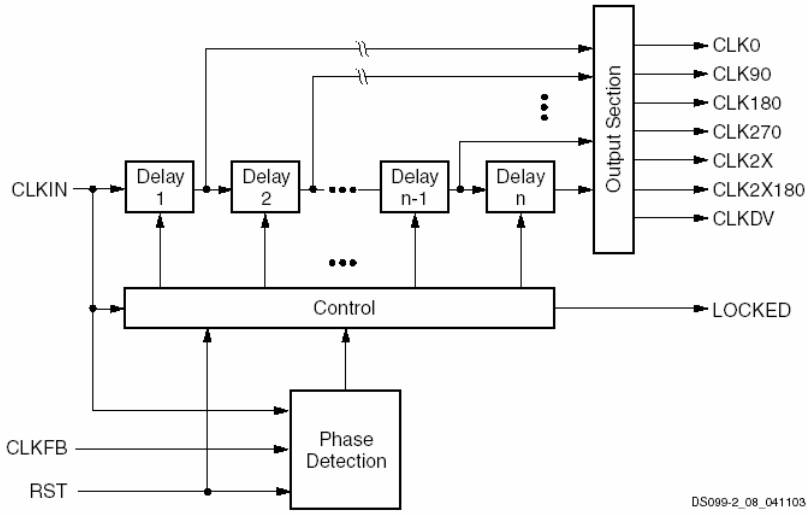


Figure 35 - Spartan-3 Delay-Locked Loop (DLL)

The DLL component has two clock inputs, CLKIN and CLKFB, as well as seven clock outputs. The clock signal supplied to the CLKIN input serves as a reference waveform, with which the DLL seeks to align the feedback signal at the CLKFB input. When eliminating clock skew, the common approach to using the DLL is as follows:

> *The CLK0 signal is passed through the clock distribution network to all the registers it synchronizes. These registers are either internal or external to the FPGA. After passing through the clock distribution network, the clock signal returns to the DLL via a feedback line called CLKFB. The control block inside the DLL measures the phase error between CLKFB and CLKIN. This phase error is a measure of the clock skew that the clock distribution network introduces. The control block activates the appropriate number of delay elements to cancel out the clock skew. Once the DLL has brought the CLK0 signal in phase with the CLKIN signal, it asserts the LOCKED output, indicating a "lock" on to the CLKIN signal.*

The DLL supports two distinct operating modes: High Frequency and Low Frequency. The Low Frequency mode permits all seven DLL clock outputs to operate over a low-to-moderate frequency range. The High Frequency mode allows the CLK0, CLK180 and CLKDV outputs to operate at the highest possible frequencies. The remaining DLL clock outputs are not available for use in High Frequency mode.

The DFS component generates clock signals the frequency of which is a product of the clock frequency at the CLKIN input and a ratio of two user-determined integers. This user-defined ratio permits a wide range of possible output frequencies.

$$f_{CLKFX} = f_{CLKIN} \cdot \frac{C_{MUL}}{C_{DIV}} \quad ; \quad C_{MUL} \in [2,32] \ , \ C_{DIV} \in [1,32]$$

This allows multiplication range from x(1/16) up to x32.

In addition to CLK0 for zero-phase alignment to the CLKIN signal, the DLL also provides the CLK90, CLK180 and CLK270 outputs for 90°, 180° and 270° phase-shifted signals, respectively. These outputs afford "coarse" phase control.

The second approach to **phase-shifting** uses the PS component to provide a still finer degree of control. The PS component is only available when the DLL is operating in its Low-frequency mode. The PS component phase shifts the DCM output clocks by introducing a "fine phase shift" between the CLKFB and CLKIN signals inside the DLL component. The user can control this fine phase shift down to a resolution of 1/256 of a CLKIN cycle.

Spartan-3 devices have 8 global clock inputs called GCLK0 – GCLK7. These inputs provide access to a low-capacitance, low-skew network that is well-suited to carrying high-frequency signals.

Eight global clock multiplexers (also called BUFGMUX elements) are provided that accept signals from global clock inputs and route them to the internal clock network as well as DCMs.

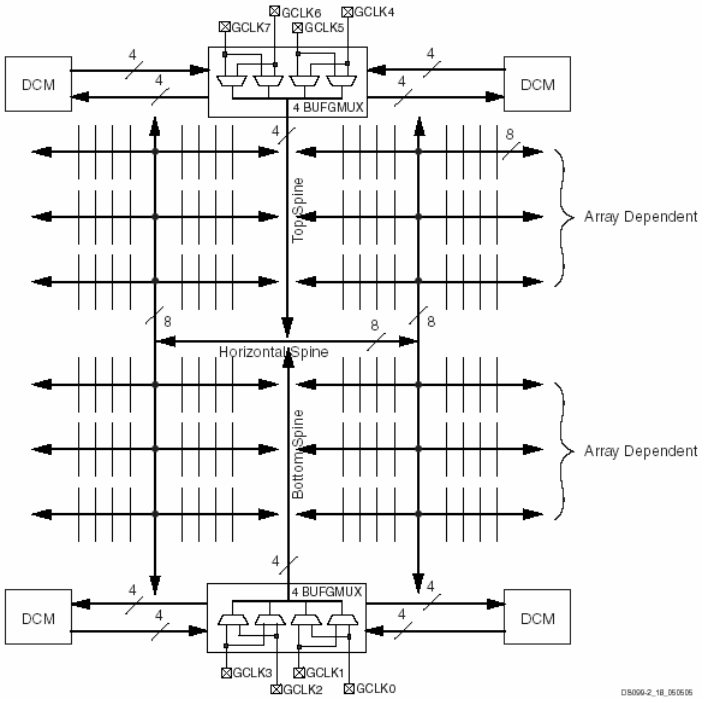


Figure 36 - Spartan-3 clock network

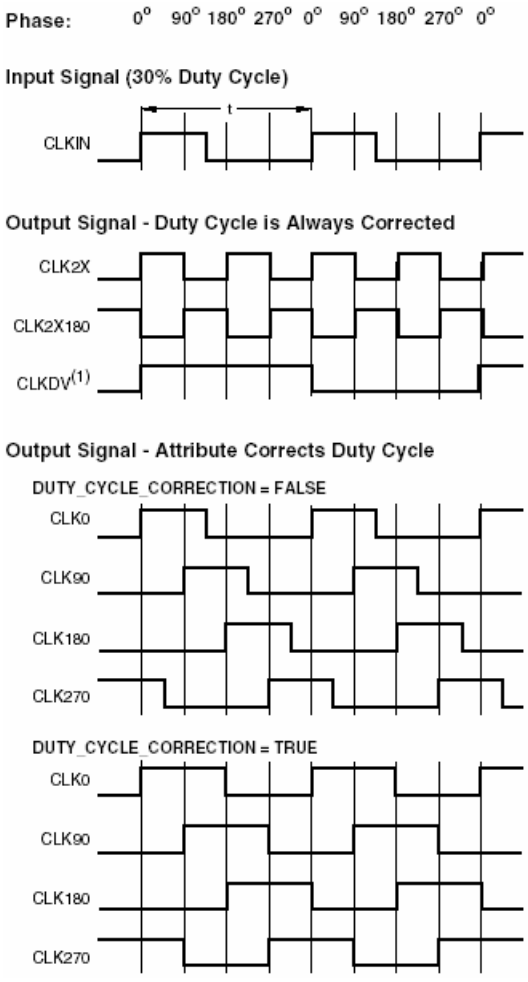


Figure 37 - Characteristics of DLL outputs

Embedded memory (block RAM)

All Spartan-3 devices support 200 MHz block RAM, which is organized as configurable, synchronous 18Kbit blocks, in up to 4 columns.

Each block RAM contains 18,432 bits of fast static RAM, 16Kbits of which is allocated to data storage and, in some memory configurations, an additional 2Kbits allocated to parity bits. Physically, the block RAM memory has two completely independent access ports, labeled Port A and Port B (dual port memory).

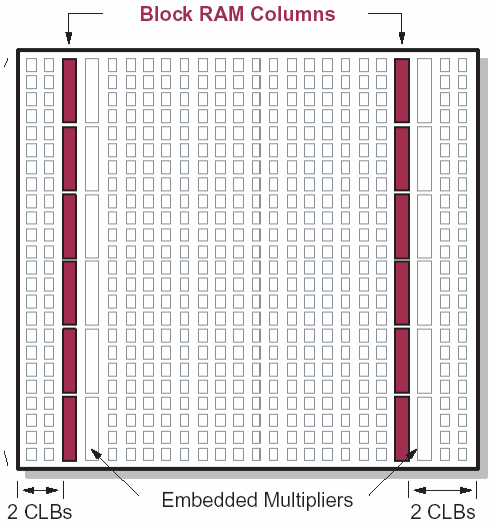


Figure 38 - XC3S200 floorplan

The structure is fully symmetrical. Both ports are interchangeable and both ports support data read and write operations. Each memory port is synchronous, with its own clock, clock enable, and write enable. Read operations are also synchronous and require a clock edge and clock enable.

Though physically a **dual-port memory**, block RAM can simulate a single-port memory in an application. Furthermore, each block memory supports multiple configurations.

Possible use of block RAM include:
local data storage, FIFOs, buffers, stacks, associative memories, state machines, program storage for embedded processor(s), etc…

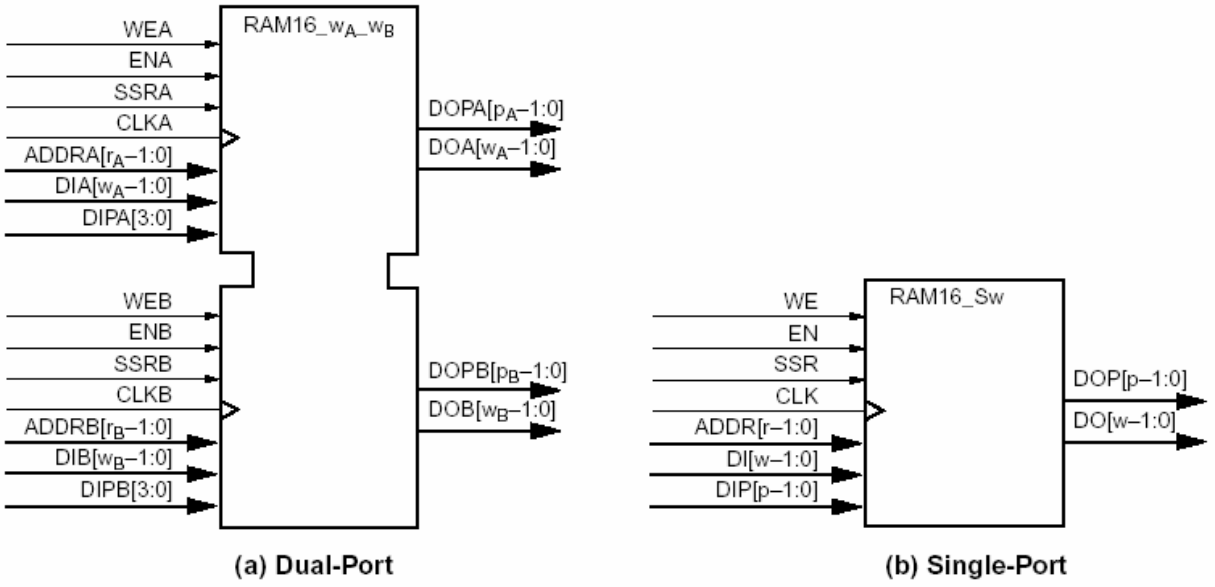


Figure 39 - Spartan-3 block memory module, although physically dual-port RAM, can be configured as single-port RAM

Embedded multipliers

All Spartan-3 devices provide embedded multipliers that accept two 18-bit words as inputs to produce a 36-bit product.

With 4 to 104 of these dedicated multipliers in each device, fast arithmetic functions can be implemented with minimal use of the general-purpose resources. In addition to the performance advantage, dedicated multipliers require less power than CLB-based multipliers.

The input buses to the multiplier accept data in two's-complement form (either 18-bit signed or 17-bit unsigned). One such multiplier is matched to each block RAM. The close physical proximity of the two ensures efficient data handling.

Cascading multipliers permits multiplicands more than three in number as well as wider than 18-bits.

Multiplication using inputs with more than 18 bits is possible by decomposing the multiplication process into smaller sub-processes. The binary representation of either input can be split at any point, provided the proper weighting and sign of the MSBs is taken into account. Splitting off the 18 MSBs of the input makes the best use of the 18-bit signed multipliers.

For example, Fig. 40 shows how a 22x16 multiplier could be implemented. The 22-bit value is decomposed into an 18-bit signed value and a 4-bit unsigned value from the LSBs. Two partial products are formed. The first is a 20-bit signed product, which is the result of multiplying the 16-bit signed value by the 4-bit unsigned section. The second is a 34-bit signed product, formed by multiplying the 16-bit signed value by the 18-bit signed section. The addition process restores the weighting of the products (note the least significant bits of the first product bypass the addition) and forms the final 38-bit product. Since the first product is signed, the 20-bit value needs to be sign-extended before addition. The adder itself only needs to be 34 bits, requiring 17 slices.

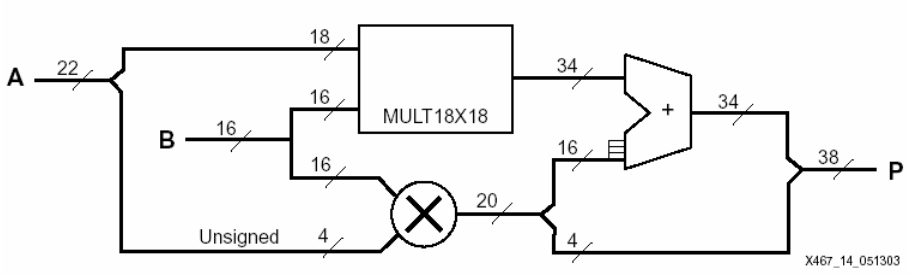


Figure 40 - 22x16 multiplier implementation

I/O structure and features

The Input/Output Block (IOB) provides a programmable, bidirectional interface between an I/O pin and the FPGA's internal logic.

A Spartan-3 IOB is shown in Fig. 41.

There are three main signal paths within the IOB (each path has its own pair of storage elements that can act as either registers or latches):

- ✓ Input path

  The input path carries data from the pad, which is connected to a package pin, through an optional programmable delay element directly to the I line. There are alternate routes through a pair of storage elements to the IQ1 and IQ2 lines. The IOB outputs - I, IQ1, and IQ2 all lead to the FPGA's internal logic. The delay element can be set to ensure a hold time of zero.

- ✓ Output path

  The output path, starting with the O1 and O2 lines, carries data from the FPGA's internal logic through a multiplexer and then a three-state driver to the IOB pad. In addition to this direct path, the multiplexer provides the option to insert a pair of storage elements.

- ✓ 3-state path

  The 3-state path determines when the output driver is high impedance. The T1 and T2 lines carry data from the FPGA's internal logic through a multiplexer to the output driver. In addition to this direct path, the multiplexer provides the option to insert a pair of storage elements. When the T1 or T2 lines are asserted High, the output driver is high-impedance (floating, Hi-Z).

The optional **pull-up** and **pull-down resistors** are intended to establish High (V_{CCO}) and Low (GND) levels, respectively, at unused I/O pins.

Each I/O block has an optional **keeper circuit** that retains the last logic level on a line after all drivers have been turned off. This is useful to keep bus lines from floating when all connected drivers are in a high-impedance state.

Clamp diodes protect all device pads against damage from electrostatic discharges.

In certain conditions it is common practice to add termination resistors to the line carrying the signal. These resistors effectively match the impedance of a device's I/O to the characteristic impedance of the transmission line, thereby preventing reflections that adversely affect signal integrity. However, adding resistors to the modern devices requires too much area, and too much components. Furthermore, for some packages it is not always possible to place resistors close to the pins. Cyclone-3 devices answer these concerns by providing **Digitally Controlled Impedance (DCI)**.

Spartan-3 devices support 18 single-ended **I/O standards** and 6 I/O differential standards. Differential standards are implemented by using a pair of IOBs.

All single-ended standards except the LVCMOS, LVTTL, and PCI varieties require a reference Voltage (V_{REF}) to determine the input-switching threshold. The V_{CCO} lines provide current to the output driver. The voltage on these lines determines the output voltage swing for majority of standards.

The need to supply V_{REF} and V_{CCO} imposes constraints on which standards can be used in the same bank.

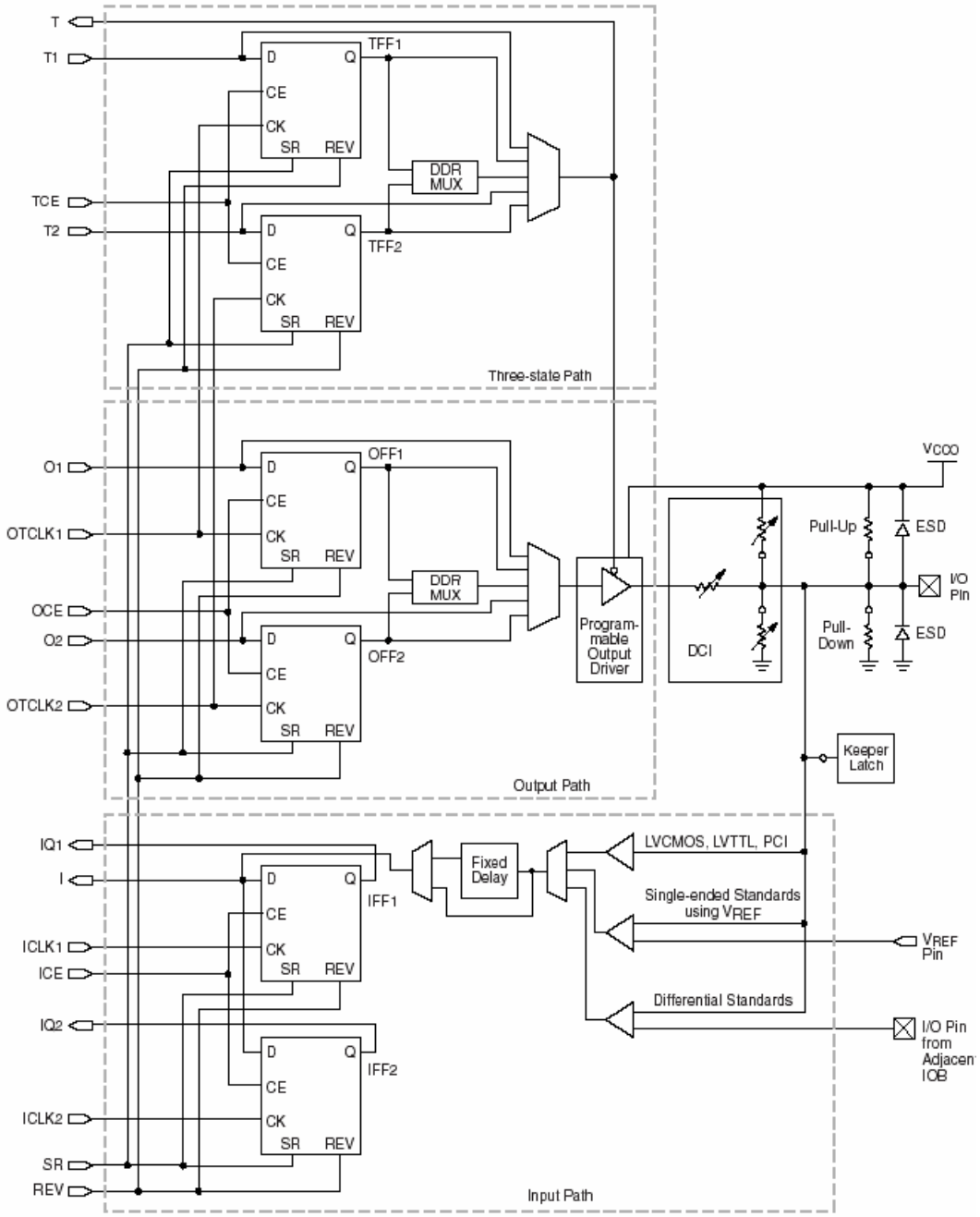


Figure 41 - Spartan-3 I/O block (IOB)

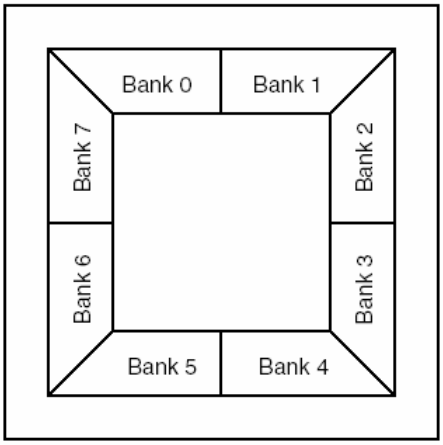


Figure 42 - Spartan-3 banks

Supported I/O standards include:

- ✓ LVTTL (3.3V)
- ✓ LVCMOS (3.3V, 2.5V, 1.8V, 1.5V)
- ✓ SSTL (classes I, II) and differential
- ✓ HSTL (classes I, II, III) and differential
- ✓ PCI 3.0V
- ✓ etc.

Start-up configuration

Spartan-3 devices are configured by loading application specific configuration data into internal configuration memory. Configuration is carried out using a subset of device pins, some of which are "dedicated" to one function only, while others, indicated by the term "dual-purpose", can be re-used as general-purpose User I/Os, once configuration is complete.

Configuration Mode(1)	M0	M1	M2
Master Serial	0	0	0
Slave Serial	1	1	1
Master Parallel	1	1	0
Slave Parallel	0	1	1
JTAG	1	0	1

Figure 43 - Spartan-3 configuration modes

Depending on the system design, several configuration modes are supported, selectable via mode pins. The mode pins M0, M1, and M2 are dedicated pins.

The table on the left list the possible configuration modes.

In **Slave Serial mode**, the FPGA receives configuration data in bit-serial form from a serial PROM or other serial source of configuration data. The FPGA on the far right of Fig. 44 is set for the Slave Serial mode.

The CCLK pin on the FPGA is an input in this mode. The serial bitstream must be set up at the DIN input pin a short time before each rising edge of the externally generated CCLK.

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 internally to the DOUT pin.

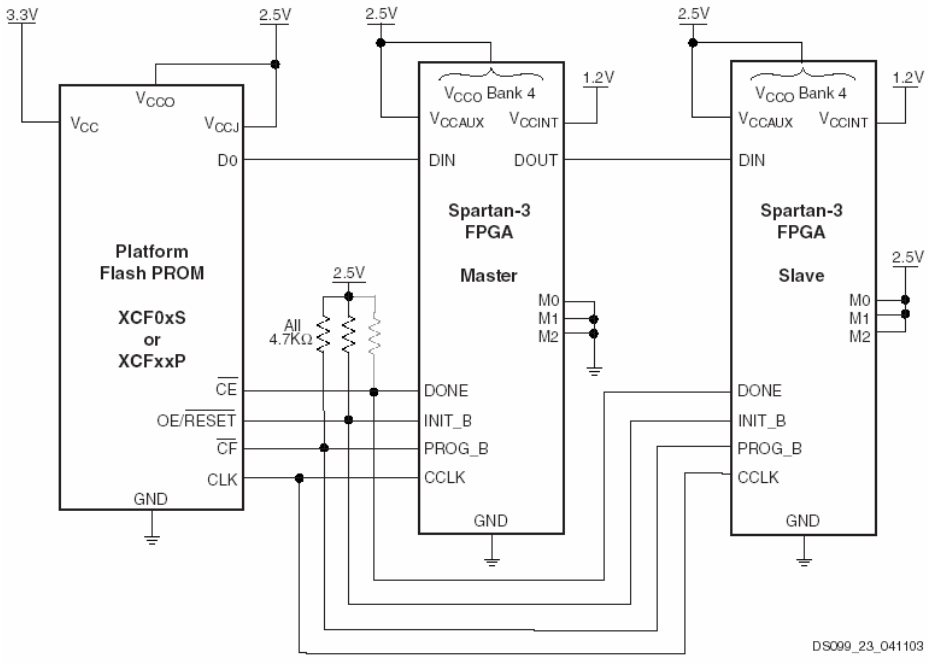


Figure 44 - Spartans in Master-Serial (center) and Slave-Serial (right) configuration modes

In **Master Serial mode**, the FPGA drives CCLK pin, which behaves as a bidirectional I/O pin. The FPGA in the center of Fig. 44 is set for Master Serial mode and connects to the serial configuration PROM and to the CCLK inputs of any slave FPGAs in a configuration daisy-chain.

The master FPGA drives the configuration clock on the CCLK pin to the Xilinx Serial PROM, which, in response, provides bit-serial data to the FPGA's DIN input. The FPGA accepts this data on each rising CCLK edge. After the master FPGA finishes configuring, it passes data on its DOUT pin to the next FPGA device in a daisy-chain. The DOUT data appears after the falling CCLK clock edge. The Master Serial mode interface is identical to Slave Serial except that an internal oscillator generates the configuration clock (CCLK).

With **Slave Parallel mode** byte-wide data is written into the FPGA with a BUSY flag controlling the flow of data. An external source provides 8-bit-wide data, CCLK, a Chip Select (CS_B) signal and a Write signal (RDWR_B).

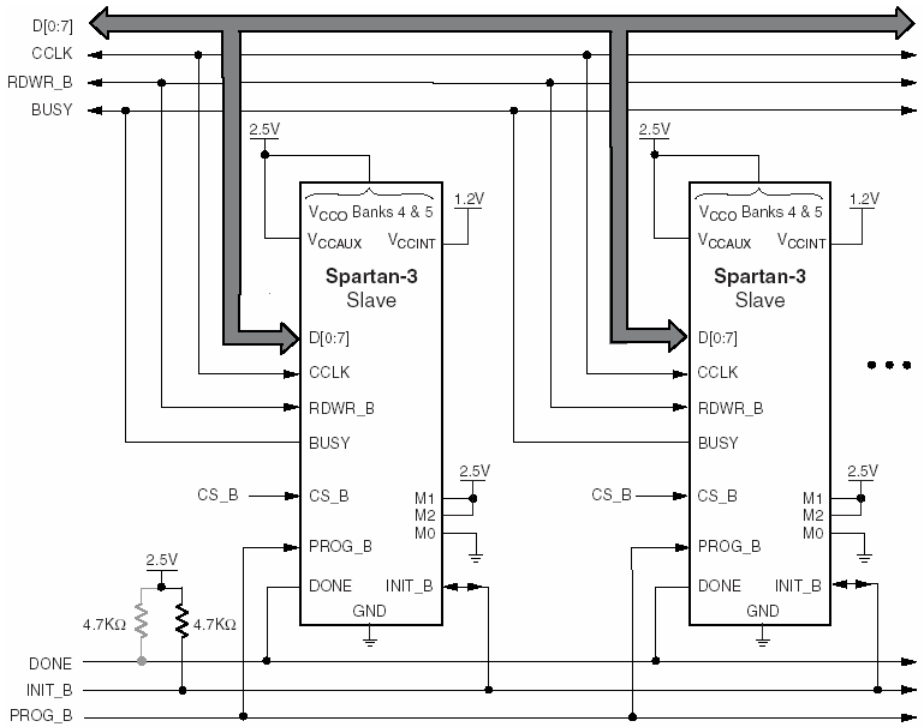


Figure 45 - Two Spartans in Slave-Parallel configuration mode

In **Master Parallel mode**, the FPGA configures from byte-wide data, and the FPGA supplies the CCLK configuration clock. CCLK behaves as a bidirectional I/O pin.

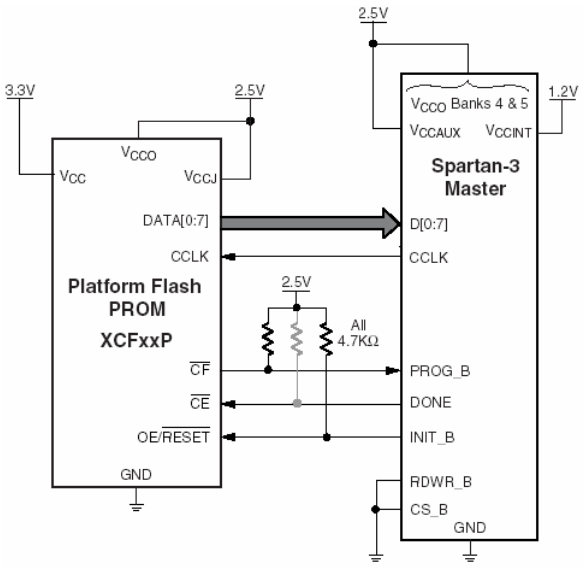


Figure 46 - Master-Parallel configuration mode

In **Boundary-Scan (JTAG) mode**, dedicated pins are used for configuring the FPGA. The configuration is done entirely through the IEEE 1149.1 Test Access Port (TAP).
FPGA configuration using the Boundary-Scan mode is compliant with the newest IEEE standards for In-System Configurable (ISC) devices.

Stratix II family

A quick overview

In February 2004 Altera launched newly architected, high-density Stratix II family of FPGAs.

The Stratix II FPGA family is based on a 1.2-V, 90-nm, all-layer copper SRAM process and features a new logic structure that maximizes performance, and enables device densities approaching 180,000 equivalent logic elements (LEs). Stratix II devices offer up to 9 Mbits of on-chip, *TriMatrix* memory for memory-demanding applications and has up to 96 DSP blocks with up to 384 (18-bit × 18-bit) multipliers for efficient implementation of high performance filters and other DSP functions. Various high-speed external memory interfaces are supported, including double data rate (DDR) SDRAM and DDR2 SDRAM, RLDRAM II, quad data rate (QDR) II SRAM, and single data rate (SDR) SDRAM. Stratix II devices support various I/O standards along with support for 1-gigabit per second (Gbps) source synchronous signaling.

Stratix II devices offer a complete clock management solution with internal clock frequency of up to 550 MHz and up to 12 phase-locked loops (PLLs). Stratix II devices are also the industry's first FPGAs with the ability to decrypt a configuration bitstream using the Advanced Encryption Standard (AES) algorithm to protect designs.

System designers requiring a low-risk cost-reduction path for high-volume production can easily migrate their Stratix II FPGA designs to structured-ASIC production with HardCopy II devices. HardCopy II devices significantly minimize migration risk because they are generated directly from a Stratix II FPGA and preserve the Stratix II architecture's high density, high performance, high functionality, and enhanced timing features.

Feature	EP2S15	EP2S30	EP2S60	EP2S90	EP2S130	EP2S180
ALMs	6,240	13,552	24,176	36,384	53,016	71,760
Adaptive look-up tables (ALUTs) *(1)*	12,480	27,104	48,352	72,768	106,032	143,520
Equivalent LEs *(2)*	15,600	33,880	60,440	90,960	132,540	179,400
M512 RAM blocks	104	202	329	488	699	930
M4K RAM blocks	78	144	255	408	609	768
M-RAM blocks	0	1	2	4	6	9
Total RAM bits	419,328	1,369,728	2,544,192	4,520,488	6,747,840	9,383,040
DSP blocks	12	16	36	48	63	96
18-bit × 18-bit multipliers *(3)*	48	64	144	192	252	384
Enhanced PLLs	2	2	4	4	4	4
Fast PLLs	4	4	8	8	8	8
Maximum user I/O pins	366	500	718	902	1,126	1,170

Figure 47 - Stratix II family features (* 1 ALM = 2.5 LEs)

One of the greatest innovations and improvements is certainly represented by the ALM architecture, allowing it to be configured in various modes.

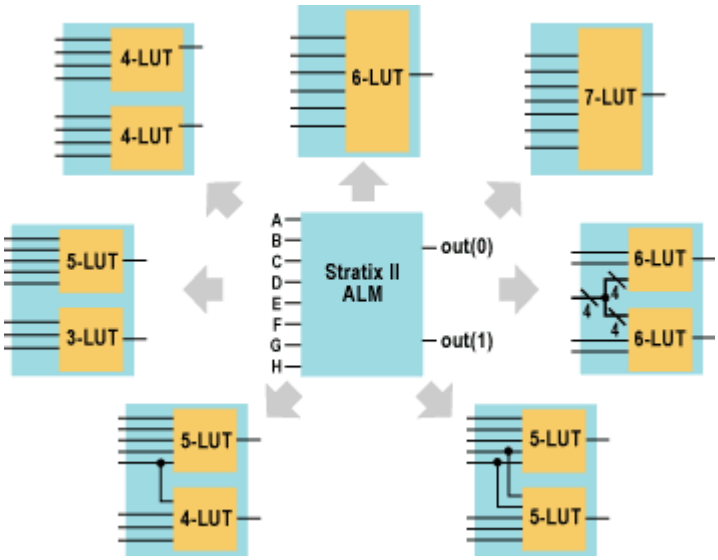


Figure 48 - Adaptive Logic Module (ALM) in Normal Mode

Virtex-4 family

A quick overview

The Virtex-4 Family is the newest generation of FPGAs from Xilinx. It was introduced in 2004. The innovative *ASMBL (Advanced Silicon Modular Block)* column-based architecture is unique in the programmable logic industry.
Three versions (platforms) of Virtex-4 devices are available: LX, FX, and SX. Each platform is optimized for particular necessities. A wide array of IP blocks complete the system solution.

- ✓ 90-nm copper CMOS process
- ✓ 1.2V core voltage
- ✓ Three high-performance platforms LX/SX/FX
  - *Virtex-4 LX: Logic applications solution*
  - *Virtex-4 FX: Full-featured solution for embedded platform applications*
  - *Virtex-4 SX: Solution for Digital Signal Processing (DSP) applications*
- ✓ Flexible logic resources
  - *Up to 40% speed improvement over previous generation devices*
  - *Up to 200,000 logic cells*
  - *Cascadable shift registers or distributed memory capability*
- ✓ *Xesium* Clock Technology
  - *Up to 20 Digital Clock Manager (DCM) blocks*
  - *Additional Phase-Matched Clock Dividers (PMCD)*
  - *32 Global Clock networks*

- ✓ *XtremeDSP* Slice
  - *18x18 signed multipliers*
  - *Optional pipeline stages*
  - *Built-In Accumulator (48-bits) & Adder/Subtracter*
  - *Up to 100% speed improvement over previous generation devices*

- ✓ *SmartRAM* Memory Hierarchy
  - *Distributed RAM*
  - *Up to 10Mb of integrated block memory operating at 500MHz*
  - *Dual-Port 18-Kbit RAM blocks*
  - *Optional pipeline stages*
  - *Optional programmable FIFO logic – Automatically remaps RAM signals as FIFO signals*
  - *High-speed memory interface support: DDR and DDR-2 SDRAM, QDR-II, etc.*

- ✓ *SelectIO* Technology
  - *1.5 to 3.3 V I/O Operation*
  - *Up to 960 user I/Os*
  - *Digitally-controlled impedance (DCI) active termination*
  - *Fine grained I/O banking (configuration in one bank)*
  - *All important high-speed I/O standards supported*

- ✓ IBM *PowerPC* RISC Processor Core (FX only)
  - *PowerPC 405 (PPC405) Core*
  - *Auxiliary Processor Unit Interface (User Co-processor)*

- ✓ *RocketIO* 622 Mb/s to 10+ Gb/s Multi-Gigabit Transceivers (FX only)

- ✓ Secure Chip AES 256-bit Encryption for Intellectual Property protection

At the heart of the Virtex-4 family is ASMBL architecture. It is a modular framework of silicon subsystems, enabling a new FPGA development methodology for rapid and cost-effective deployment of platforms targeted to different application domains.

The new highly modular ASMBL architecture makes use of advanced packaging technology and eliminates geometric layout constraints associated with traditional chip design (such as hard dependencies between I/O count and fabric array size). The ASMBL architecture also addresses the increasingly more demanding requirements for on-chip power and ground distribution by allowing power and ground to be placed anywhere on the chip.

Thanks to ASMBL, Xilinx designers can vary the number and ratio of different functional columns to create a platform or family of different sized devices, each best suited for a certain domain of applications depending on the desired type of functional attributes. Platforms can be created with different mixtures of features: logic, memory, DSP, processing and high-speed I/O. This approach enables the right feature mix at the lowest cost, and resulted in 3 platforms of Virtex-4 FPGAs – LX, FX, SX.

Altera vs. Xilinx

Deciding which of the two leading manufacturers is currently better, on basis of described features, is an impossible task:

- Both of them offer a vast range of FPGAs, at different prices, guaranteed to satisfy any user's needs.

- If we make feature-to-feature comparison of same rank FPGAs we will find that they offer the same or very similar things at very similar prices:

  - ✓ 90nm process
  - ✓ Low-voltage cores
  - ✓ Up to 200,000[5] LC (LEs)
  - ✓ Maximum internal frequency around 500 MHz
  - ✓ Embedded 18x18 multipliers and enhanced DSP features
  - ✓ Great amounts of multi-purpose embedded RAM - up to 10Mbits
  - ✓ Support for leading I/O standards and external memory interfaces
  - ✓ Numerous IP blocks (Nios II, MicroBlaze, etc.)
  - ✓ Complete software systems (ISE and Quartus II)

  As far as clock management is concerned, it seems that Xilinx is leading the way for now with higher count of digital clock managers and global clock networks.

  As we have mentioned before, one of the greatest differences between the two companies is their solution for volume production – Xilinx offers EasyPath FPGAs, Altera offers HardCopy migration path to structured ASICs. However, both of them claim that their solution is the most affordable one.

- Benchmarking also yields controversial results. All the benchmarks are performed either by Xilinx/Altera, or their associate companies. Both companies issue whitepapers claiming their FPGAs considerably outperform the opponent's ones:

  Quote:

  > *"… Our benchmark results show that for high-density 90-nm FPGAs, the Altera Stratix II family commands an average of 39% performance lead over Xilinx Virtex-4 family. For low-cost FPGAs, the Altera 90-nm Cyclone II family provides an average 60% higher performance than the Xilinx 90-nm Spartan-3 family…"*
  >
  > **Altera whitepaper, "FPGA Performance Benchmarking Methodology"**

  Quote:

  > *"… Cyclone II performance as demonstrated by a suite of customer designs using the most cost effective speed grade has degraded almost a full speed grade from Quartus II v4.1 to v4.2, and further degradation is indicated for the new v5.0. Spartan-3 design performance is now slightly faster than Cyclone II when comparing the most cost effective speed grade in each device…"*
  >
  > **Xilinx whitepaper, "Spartan-3 vs. Cyclone II Performance Analysis"**

[5] With new architectures like ALM, expressing the "logic" density of FPGAs in terms of LCs (LEs) is not accurate.

So, is there a right way to find out?

Let us ask the customers:

Quote:

"… Xilinx swept all FPGA vendor categories in a survey of more than 350 design teams worldwide in which respondents were asked to rate their experience with FPGA and EDA companies' products and services.

FPGA designers ranked Xilinx highest in reader/customer satisfaction for devices, design tools, service and support, including:

Virtex and Spartan FPGAs - *"Xilinx continues to lead the pack in performance and features, and goes the extra mile in explaining how to use their devices for particular class of application."*

ISE design tools *- "Xilinx has made significant improvements to their tool suite over the past year, particularly in the DSP and embedded design areas."*

Support staff, application engineers, and documentation- *"Xilinx consistently sets the standard for support staff and resources, particularly with their robust website and responsible and knowledgeable application engineers."*

***FPGA Journal*, October 2004**

Conclusion

It seems that for now Xilinx is the winner. However, as more and more money is in game the competition slowly starts to close the gaps. A careful reader will notice that the stated reasons for Xilinx winning the readers' award have more to do with client relations than with a great difference in performance.

The competition will undoubtedly remain fierce, giving birth to the new state-of-the-art FPGA architectures, to the delight of FPGA users.

References

1. *Cyclone II Device Family Data Sheet*, Altera Inc., July 2005
2. *Stratix II Device Handbook*, vol 1, Altera Inc., July 2005
3. *Spartan-3 FPGA Family: Complete Data Sheet*, Xilinx Inc., August 2005
4. *Spartan-3 Application notes*, Xilinx Inc., 2003-2005
5. *Virtex-4 Family Overview*, Xilinx Inc., June 2005
6. *FPGA Designer Quickstart Guide*, whitepaper, Altium Inc., July 2005
7. *FPGA: The chip that flip-flops*, lecture notes, J. A. Zubairi, October 2004
8. *Reconfigurable Computing - FPGA structures*, lecture notes, J. Morris, University of Auckland, NZ
9. *Phase-Locked Loops, lecture notes*, B. Nikolic, February 2004
10. *FPGA Performance Benchmarking Methodology*, whitepaper, Altera Inc., 2004
11. *Spartan-3 vs. Cyclone II Performance Analysis*, whitepaper, Xilinx Inc., May 2005
12. *Stratix II vs. Virtex-4 Performance Comparison*, whitepaper, Altera Inc., 2005
13. *The Stratix Routing and Logic Architecture*, D. Lewis et al., FPGA'03 Monetrey, February 2003
14. *The Roles of FPGAs in Reprogrammable Systems*, S. Hauck, Northwestern University Illinois, Proceedings of the IEEE, 1997
15. *Overview of Programmable Logic Devices*, J.P. Davis, lecture notes, University of South Carolina, 2004.
16. *Reconfigurable Computing*, lecture notes, University of Massachusets, 2004
17. *FPGA Direction*, J. Horgan, article, EDAcafe.com weekly, August 2004
18. *Virtex-4 Gets Real, K. Morris*, article, FPGA and Programmable Logic Journal, September 2004
19. *Synplifying Physical Synthesis*, K. Morris, article, FPGA and Programmable Logic Journal, October 2005
20. *Annual FPGA Journal Reader's Choice Awards*, FPGA Journal, October 2004
21. *VLSI lecture notes*, ETF, University of Belgrade
22. www.altera.com online resources
23. www.xilinx.com online resources

Abbreviations

AES - Advanced Encryption Standard
AGP - Advanced Graphics Port
ALM - Adaptive Logic Module
ASIC - Application Specific Integrated Circuit
ASMBL – Advanced Silicon Modular Block

BST - Boundary-Scan Test

CLB – Configurable Logic Block

DCI - Digitally Controlled Impedance
DCM - Digital Clock Manager
DDR - Double Data-Rate
DLL – Delay Locked Loop
DSP - Digital Signal Processing

ELC - Equivalent Logic Cell
EDA - Electronic Design Automation

FFT - Fast Fourier Transform
FPGA - Field Programmable Gate Array

GTL - Gunning Transceiver Logic

HDL - Hardware Description Language
HSTL - High-Speed Transceiver Logic

IOB - Input/Output Block
IOE - Input/Output Element
IP - Intellectual Property
ISE - Xilinx development software

JTAG - Joint Test Action Group

LAB - Logic Array Block
LC - Logic Cell
LE - Logic Element
LSB - Least Significant Bit
LUT - Look-Up Table
LVCMOS - Low-Voltage CMOS
LVTTL - Low-Voltage TTL

MAC - Multiply and Accumulate
MSB - Most Significant Bit

PCI - Peripheral Component Interface
PLD - Programmable Logic Device
PLL - Phase Locked Loop

RTL - Register Transfer Level

SSTL - Stub Series Terminated Logic

Contents