Lattice Semiconductor Corporation
Bringing the Best Together

OPTIMIZING FPGAs FOR HIGH-VOLUME APPLICATIONS

A Lattice Semiconductor White Paper

June 2004

Revised January 2005

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Optimized Logic Solutions For High-Volume Applications

For a number of years designers of moderate-volume equipment, such as that typically found in equipment intended for core networking, industrial and high-performance computing, have enjoyed the low risk and fast time-to-market advantages of FPGA devices.   High-volume equipment designs, such as consumer electronics, wireless communications and network access equipment, with their associated cost sensitivity, have typically used ASICs for logic implementation. However, decreasing product life cycles and increasing minimum economic volumes for ASICs make this approach increasingly challenging.

The increases in the minimum economic volumes for ASICs stems from the fact that mask sets become increasingly expensive with each new generation of ASIC, and that the number of die per wafer for a given design complexity increases, due to both smaller geometries and larger wafer sizes.

These same trends work in favor of programmable logic solutions. Smaller geometries and larger wafers lead to lower programmable device costs. As the device costs for FPGAs continue to improve and the NRE costs for ASICs escalate, an increasing number of high-volume designs can take advantage of FPGA technology, as illustrated in Figure 1.

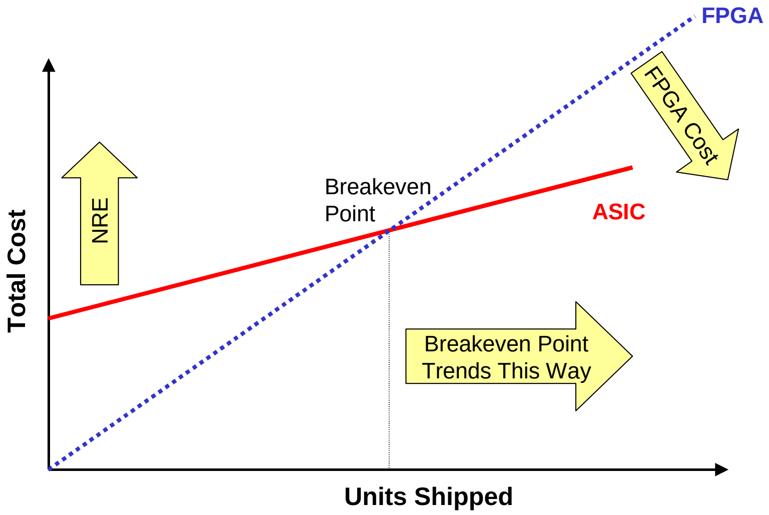


Figure 1 – FPGA versus ASIC Breakeven Point

Achieving Optimized Programmable Logic Solutions

In order to maximize the applicability of FPGA architecture to high-volume design, it is necessary to optimize cost, features and performance. This optimization begins with developing a clear understanding of the features that are required for FPGAs in this space. Then, at every step of the development process, from architectural definition and process choice to silicon design, activities have to be centered on this understanding.

Initial attempts by some vendors did not achieve the required optimization. One approach has been to define the pad ring and then backfill with logic. While achieving low cost, this approach does little to ensure that the correct feature set is delivered to designers. Another approach has been to begin with a high-performance architecture, and then strip out features. Unfortunately, this approach has not yielded the cost savings demanded by designers in high-volume markets.

An additional opportunity for optimization that is critical for high-volume FPGA design is optimization of overall system cost. Availability of low-cost configuration memory, the capability to simplify the interface to common components such as DDR memory, and devices that utilize LVDS all impact overall equipment cost.

This whitepaper next examines specific factors impacting the optimization of an FPGA architecture in terms of the logic block, I/O support, external memory interfacing and configuration memory support. It also details the architectural decisions that Lattice Semiconductor made based on these factors when architecting its LatticeECP^{TM} (ECconomy Plus) and LatticeEC^{TM} (ECconomy) device families. The paper concludes with an overview of the LatticeECP/EC architecture.

Logic Block and Routing

The logic block and its associated routing are of great importance during device optimization because they are the largest contributor to die area. For Look Up Table (LUT) based architectures, one area of considerable importance for this optimization is the support of distributed memory: the capability to use LUTs as small memories. This is because support for this feature increases logic block size up to 20%. Interviews with designers indicate that most designers prefer to have a small amount of distributed memory to allow the efficient implementation of scratch pad memories and register files. However, very few designs required more than a small percentage of the LUTs to support this feature. This is shown conceptually in Figure 2.

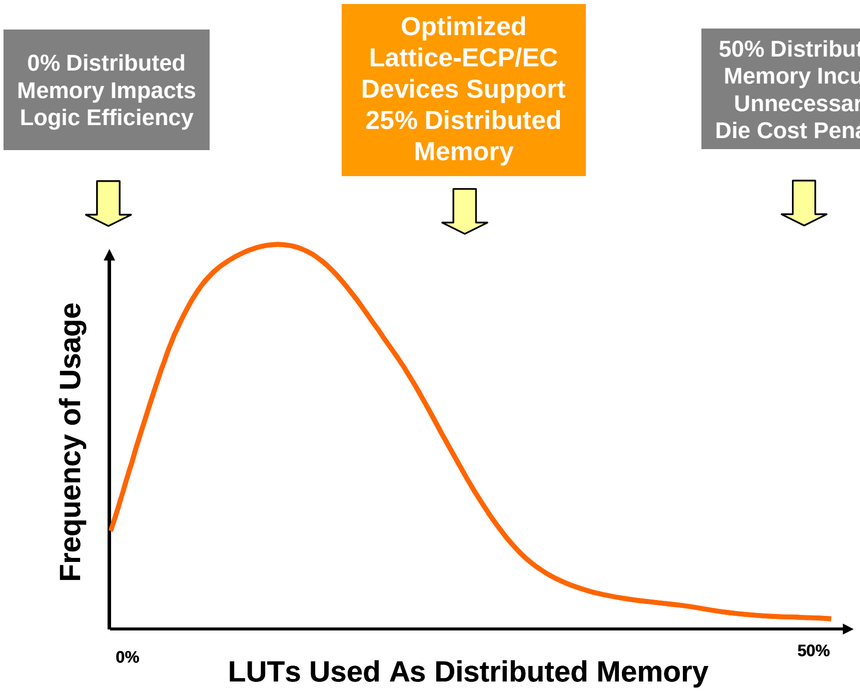


Figure 2 – Distributed Memory Requirements For High-Volume Designs

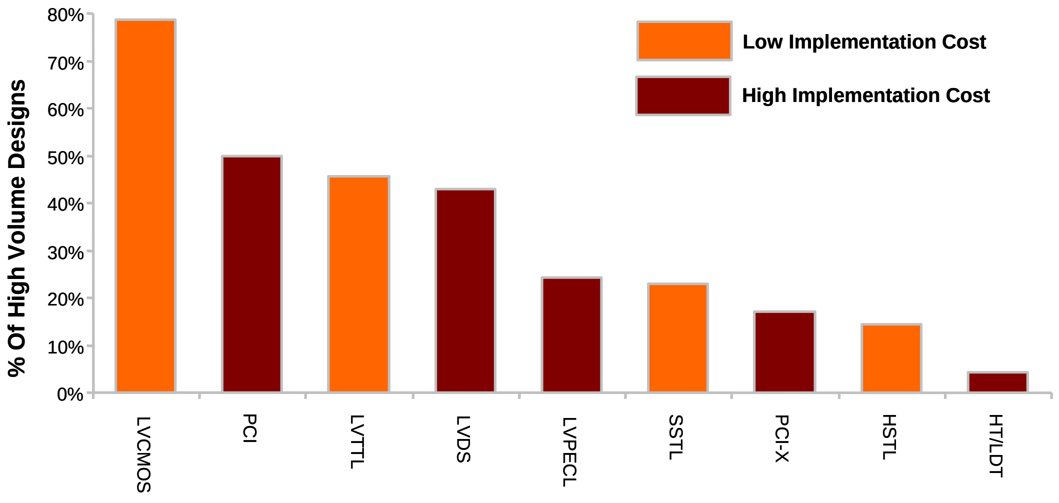
To best meet the desire for distributed memory while controlling costs, Lattice chose to support distributed memory in 25% of the logic blocks within the LatticeECP/EC devices. This contrasts sharply with other architectures intended for high-volume applications that either provide no distributed memory, forcing designers to use many LUTs to build small memories, or 50% distributed memory, requiring designers to pay an additional overhead for a feature they are unlikely to use.

The amount of routing an architecture contains has a significant impact on its cost, performance and device usability. The choice of process has a strong influence on the ability to balance the cost, performance and usability variables. For its LatticeECP/EC devices, Lattice chose to use a 130nm Low-k dielectric process from Fujitsu that provides 9 layers of metal. The availability of a large number of high-performance metal layers provided an excellent basis for building a cost effective routing fabric. Utilizing this process, a mix of x1, x2, and x6 routing resources was developed that provides excellent routability while balancing performance and cost. This routing is fully buffered for consistent high-speed, independent of loading.

I/O Standard Support

The I/O buffers and related interface logic represent the second largest contributor to die area (after the logic block routing combination), so this area also deserves careful examination during architectural optimization. The I/O standards supported by a device are the primary driver of its costs in the I/O area. There are a wide variety of I/O standards, each with its own popularity. Figure 3 shows, based on Lattice Semiconductor research, the popularity of nine common I/O standards, as well as an indication of the relative cost of implementing each I/O type, categorized as either high or low.

Based on these inputs, Lattice tailored the I/O support for its LatticeECP/EC devices. The most popular I/Os (LVCMOS, PCI, LVTTL, and LVDS) were implemented in the LatticeECP/EC devices. However, those popular I/O types having a high implementation cost (PCI and LVDS) were implemented only on half the device, as system designers generally indicated that they required only a limited number of I/Os to support these standards. Those I/O standards with a lower popularity (LVPECL, SSTL, PCI-X, HSTL, and HT/LDT) were implemented only in cases in which they did not significantly increase the implementation costs (SSTL and HSTL.) The devices can also emulate LVPECL with the addition of external resistors.



Percentage of ECP/EC I/O Supporting	**100%**	**50%**	**100%**	**50%**	**0%***	**100%**	**0%**	**100%**	**0%**

* Can be supported through emulation

Figure 3 – I/O Standard Support Requirements For High-Volume Designs

Other low-cost devices either fail to provide on-chip support for all of the most popular standards, most notably missing true LVDS support, or implement a plethora of I/O standards, unnecessarily increasing the cost associated with the I/O.

External Memory Interface Support

Designers of cost sensitive equipment naturally select the memory technology that provides the lowest cost per bit given their technical requirements; traditionally, this has often meant Synchronous DRAM (SDRAM). However, in the last few years volume shipments of Double Data Rate DRAM have been growing to the point where DDR DRAM is expected to represent over 50% of the bits shipped in 2004. Increasingly, designers are finding that DDR DRAM provides a lower cost per bit than SDRAM.

Although DDR DRAM in many cases provides cheaper storage, it is significantly more difficult to interface to than SDRAM. The design challenges include aligning data (DQ) with data strobe (DQS) signals, splitting a stream of data with transitions on both edges of the clock into multiple streams transitioning on one edge of the clock, and managing data transfer from the DQS clock domain to the system clock domain. The alignment of

DQ and DQS is made all the more challenging by the bi-directional nature of the DQS signal. Figure 4 shows a typical DDR memory interface.

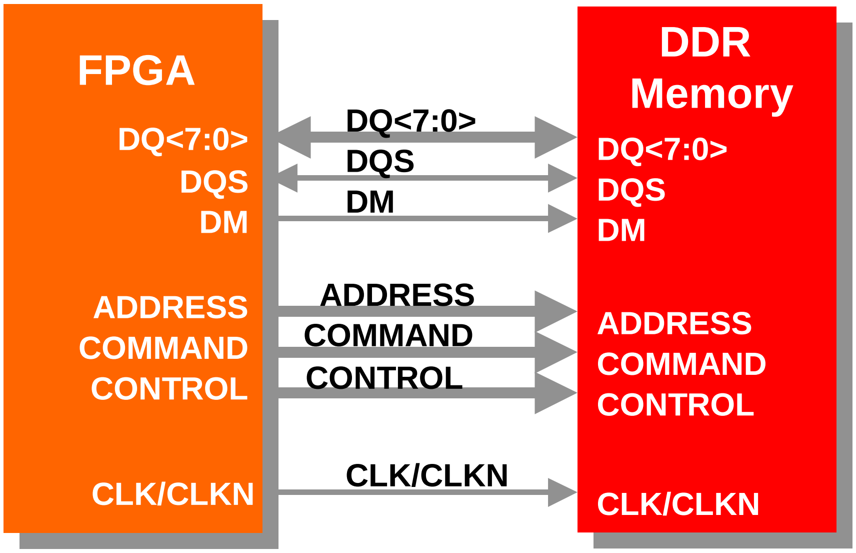


Figure 4 – Typical FPGA to DDR memory Interface

Implementing DDR interfaces in traditional FPGA logic is both challenging, especially as performance increases, and consumes a significant number of resources. The LatticeECP and EC devices provide dedicated resources to align DQ and DQS signals, multiplex to and from double data rate, and transfer data from the DQS clock domain to the system clock domain. Figure 5 indicates some of the DDR support resources available in the LatticeECP/EC devices. The approach taken by the Lattice ECP/EC devices contrasts with other low-cost FPGAs that provide either no DDR support or only limited support. The impact can be dramatic; through dedicated DDR support resources the LatticeECP/EC devices save, for general-purpose use, between 500 and 1000 registers when implementing a 64-bit wide memory interface. This represents a considerable portion of the 1.5K general-purpose registers in the smallest devices. Performance is also improved by 25% over other low-cost FPGAs, allowing faster operation, more margin to specifications or the use of a slower speed grade device.

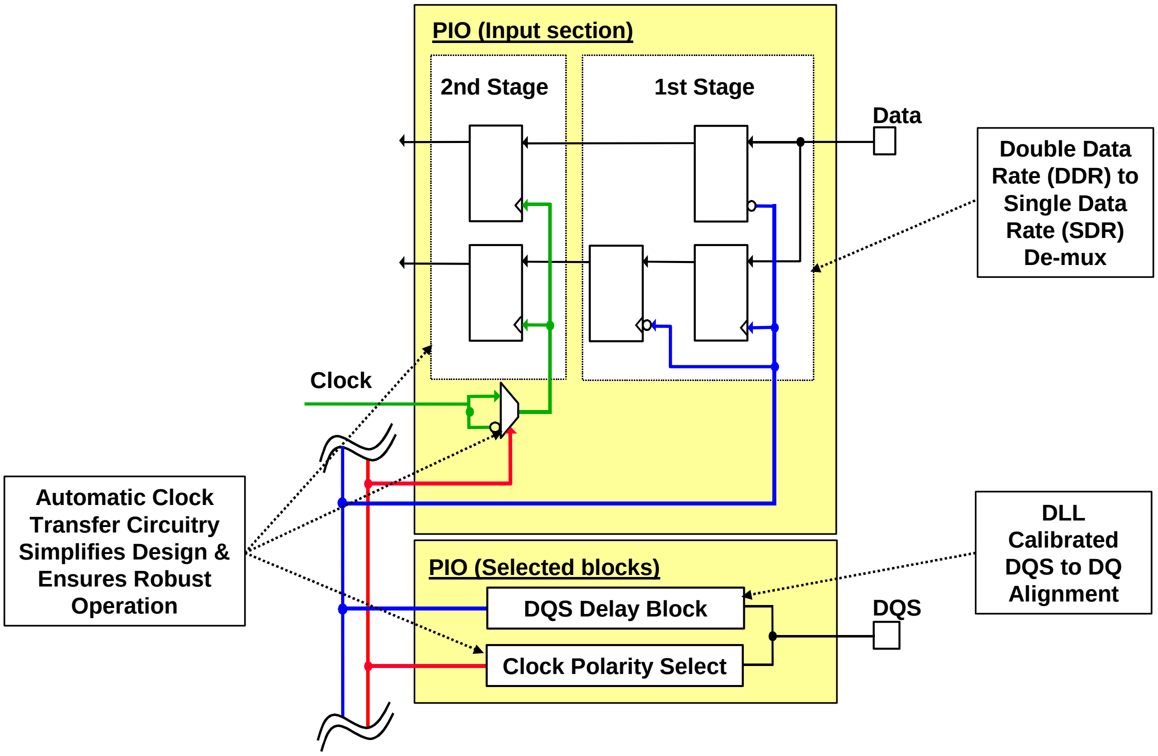

PIO (Input section)
2nd Stage
1st Stage
Data
Double Data Rate (DDR) to Single Data Rate (SDR) De-mux
Clock
Automatic Clock Transfer Circuitry Simplifies Design & Ensures Robust Operation
DLL Calibrated DQS to DQ Alignment
PIO (Selected blocks)
DQS Delay Block
Clock Polarity Select
DQS

Figure 5 – LatticeECP/EC DDR Input Support Circuitry

Configuration Memory Cost

Most FPGAs utilize SRAM cells to store their configuration; as such they need to be reloaded at power-up. Traditionally, this has been accomplished through the use of a proprietary configuration storage PROM purchased from the FPGA supplier. These configuration memories have tended to be relatively expensive, up to a third of the FPGA cost. To reduce overall costs, many designers configure their FPGAs with a microprocessor, storing the configuration in the much cheaper FLASH memory that is used for code storage. While reducing cost, this approach poses two significant challenges. As the FPGA cannot be used for microprocessor support logic, a second smaller programmable device is often required on the board, resulting in higher costs and part count. By including the FPGA configuration data in the microprocessor code store, hardware configuration and software become intertwined, in some cases requiring a full software release to implement a minor change in the hardware.

Over the last few years multiple suppliers have developed small footprint Flash memories that utilize the Serial Peripheral Interface (SPI) standard. Vigorous competition among suppliers has keep prices for these parts at a reasonable level and up to four times lower per bit than the proprietary boot memories supplied by FPGA manufacturers. In order to optimize overall system costs, Lattice Semiconductor architected the LatticeECP/EC devices to support configuration directly from SPI memory (Figure 6) in addition to a variety of other Parallel, Serial, and JTAG-based modes.

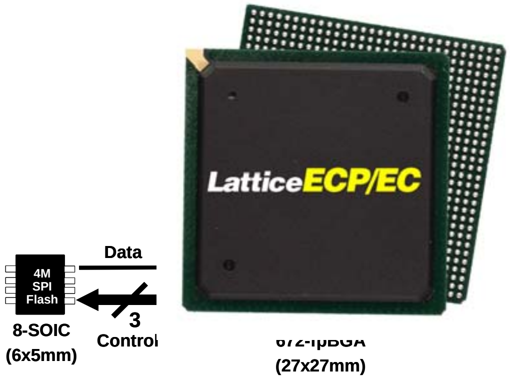


Figure 6 – LatticeECP/EC to SPI Flash Interface

<u>Description of the Architecture</u>

The LatticeECP concept combines an optimized FPGA fabric with high-speed dedicated functions. The first Lattice family to be implemented with this approach is the LatticeECP-DSP (<u>EC</u>onomy <u>P</u>lus <u>D</u>igital <u>S</u>ignal <u>P</u>rocessing), which provides high-performance on-chip DSP blocks. To achieve even lower costs, the LatticeEC family supports all of the general-purpose features of the LatticeECP devices without dedicated functional blocks. Figure 7 illustrates the LatticeECP concept.

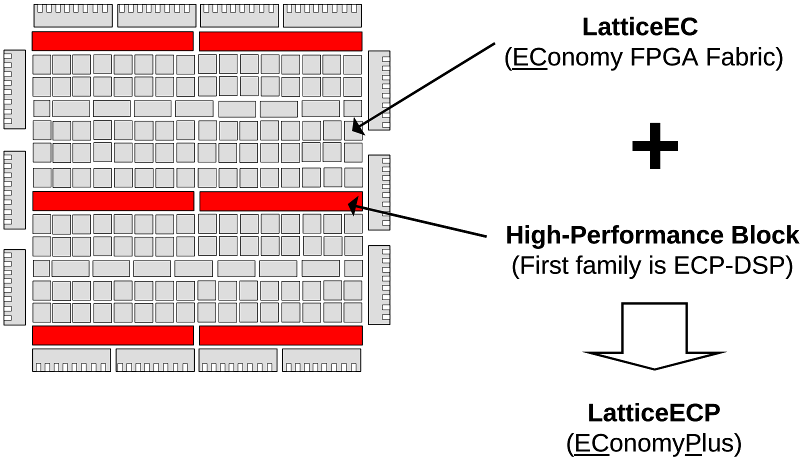


Figure 7 – The LatticeECP Concept

The LatticeECP-DSP and EC devices contain an array of logic blocks surrounded by Programmable I/O Cells (PIC). Interspersed between the rows of logic blocks are rows of sysMEM^{TM} Embedded Block RAM (EBR). The LatticeECP-DSP supports a row of sysDSP^{TM} blocks, allowing it to efficiently implement high performance DSP functions up to 10,000 multiply accumulates per second (MMAC/s).[i] The sysDSP capability is removed in the LatticeEC devices to allow a further reduction in cost for users who do not implement DSP functions. Figures 8 and 9 are the block diagrams of the LatticeECP-DSP and EC devices.

There are two kinds of logic blocks, the Programmable Functional Unit (PFU) and Programmable Functional unit without RAM (PFF). The PFU contains the building blocks for logic, arithmetic, RAM, ROM and register functions. The PFF block contains building blocks for logic, arithmetic and ROM functions. Both PFU and PFF blocks are optimized for flexibility, allowing complex designs to be implemented quickly and efficiently. Logic Blocks are arranged in a two-dimensional array. Only one type of block is used per row. The PFU blocks are used on the outside rows. The rest of the core consists of rows of PFF blocks interspersed with rows of PFU blocks. For every three rows of PFF blocks there is a row of PFU blocks.

Each PIC block encompasses two PIOs (PIO pairs) with their respective sysIO^{TM} interfaces. All PIOs support LVTTL, LVCMOS, HSTL and SSTL input and output as well as differential receivers for LVDS, BLVDS and LVPECL. The PIO pairs on the left and right edges of the device can be configured as LVDS drivers. The PIO pairs on the top and bottom of the device also provide optional PCI clamp diodes. The PIOs also contain a variety of circuitry to simplify the implementation of high performance DDR memory interfaces.

The sysMEM EBRs are large, dedicated, fast memory blocks. They provide 9Kbits of memory that can be configured as Dual-port, Pseudo Dual-Port, Single-Port or ROM. Users can also specify FIFOs in their designs and the design tools construct them out of EBRs and a small amount of PFU logic. Memory width is programmable from 1 to 36.

The PFU, PFF, PIC and EBR Blocks are arranged in a two-dimensional grid with rows and columns as shown in Figures 8 and 9. The blocks are connected with many vertical and horizontal routing channel resources. The place and route software tool automatically allocates these routing resources.

At the end of the rows containing the sysMEM Blocks are the sysCLOCK™ Phase Locked Loop (PLL) Blocks. These PLLs have multiply, divide and phase shifting capability; they are used to manage the phase relationship of the clocks. The LatticeECP/EC architecture provides up to four PLLs per device.

Every device in the family has a JTAG Port with internal Logic Analyzer (ispTRACY™) capability. The sysCONFIG™ port allows for serial or parallel device configuration and includes support for low-cost industry standard Serial Peripheral Interface (SPI) memory. The LatticeECP/EC devices use 1.2V as their core voltage. Table 1 shows the density, I/O and other statistics associated with the ECP-DSP and EC families.

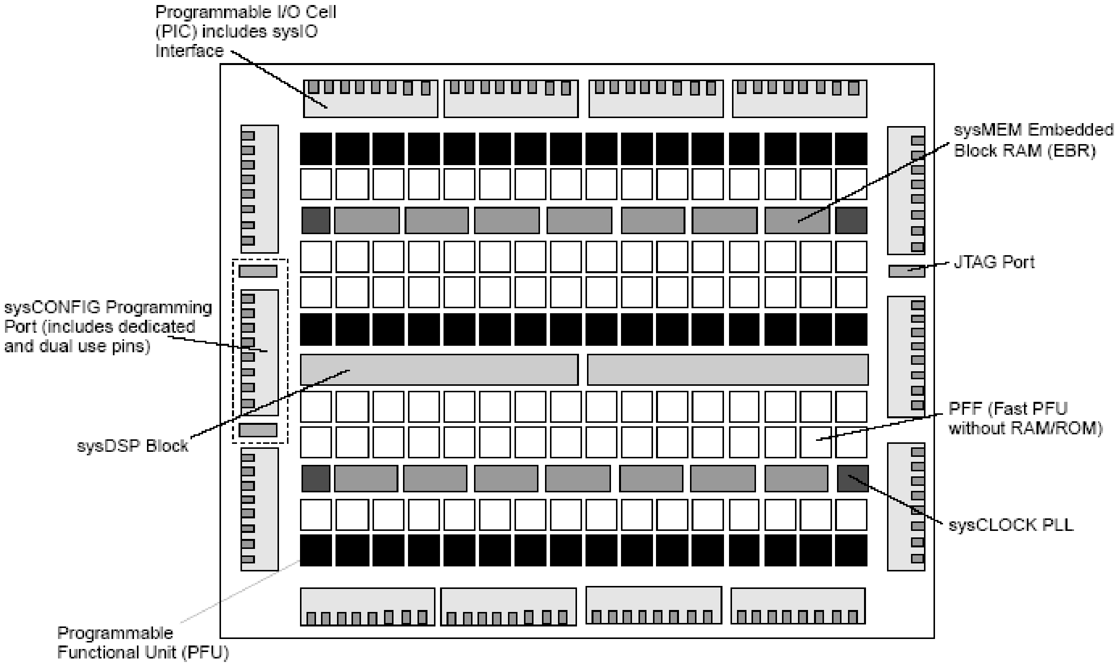

Programmable I/O Cell (PIC) includes sysIO Interface
sysMEM Embedded Block RAM (EBR)
JTAG Port
sysCONFIG Programming Port (includes dedicated and dual use pins)
sysDSP Block
PFF (Fast PFU without RAM/ROM)
sysCLOCK PLL
Programmable Functional Unit (PFU)

Figure 8 – LatticeECP-DSP Block Diagram

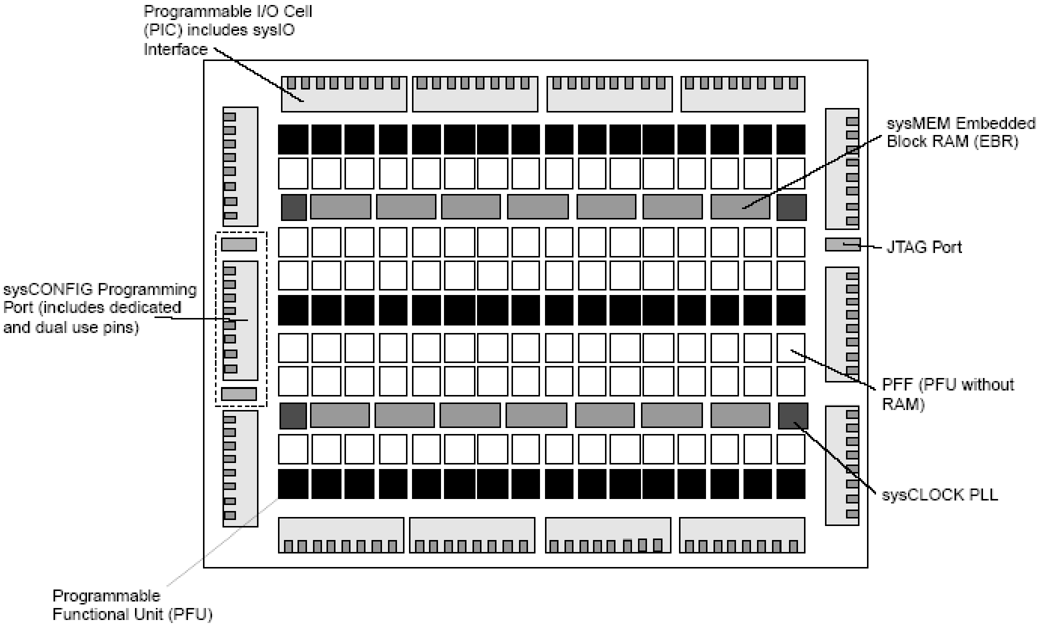

Programmable I/O Cell (PIC) includes sysIO Interface
sysMEM Embedded Block RAM (EBR)
JTAG Port
sysCONFIG Programming Port (includes dedicated and dual use pins)
PFF (PFU without RAM)
sysCLOCK PLL
Programmable Functional Unit (PFU)

Figure 9 – LatticeEC Block Diagram

Device	EC1	EC3	EC6 ECP6	EC10 ECP10	EC15 ECP15	EC20 ECP20	EC33 ECP33
LUTs (K)	1.5	3.1	6.1	10.2	15.4	19.7	32.8
sysMEM Blocks	2	6	10	30	38	46	58
sysMEM (Kbits)	18	55	92	276	350	424	535
Distributed RAM (Kbits)	6	12	25	41	61	79	131
Voltage (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2
sysDSP Blocks[1]			4	5	6	7	8
18x18 Multipliers[1]			16	20	24	28	32
PLLs	2	2	2	4	4	4	4
Max I/Os	112	160	224	288	352	400	496

1. ECP-DSP devices.

Table 1 – ECP-DSP & EC Family Members

Summary

With the ASIC approach becoming more challenging and FPGA cost effectiveness improving, utilizing FPGAs is becoming more attractive for high-volume designs. An acceleration of this trend requires that FPGAs be precisely targeted to the requirements of high-volume designers. While every aspect of architecture optimization is important, the logic block, I/O, external memory interfaces and configuration support represent the most critical areas. The LatticeECP/EC devices have been carefully optimized in these and other areas to deliver the exact feature mix required by high-volume designs at an attractive price.

###

[i] Please reference the Lattice White Paper, “High-Performance DSP Capability within an Optimized Low-Cost FPGA Architecture” for more details on this block.