REVIEW OF ADVANCED FPGA ARCHITECTURES AND TECHNOLOGIES[1]

Yang Haigang[*,**] Zhang Jia[*,**] Sun Jiabin[*] Yu Le[*]

[*](*Institute of Electronics*, *Chinese Academy of Sciences*, *Beijing 100190*, *China*)

[**](*University of Chinese Academy of Sciences*, *Beijing 100049*, *China*)

Abstract Field Programmable Gate Array (FPGA) is an efficient reconfigurable integrated circuit platform and has become a core signal processing microchip device of digital systems over the last decade. With the rapid development of semiconductor technology, the performance and system integration of FPGA devices have been significantly progressed, and at the same time new challenges arise. The design of FPGA architecture is required to evolve to meet these challenges, while also taking advantage of ever increased microchip density. This survey reviews the recent development of advanced FPGA architectures, including improvement of the programming technologies, logic blocks, interconnects, and embedded resources. Moreover, some important emerging design issues of FPGA architectures, such as novel memory based FPGAs and 3D FPGAs, are also presented to provide an outlook for future FPGA development.

Key words Field Programmable Gate Array (FPGA); Microchip architecture; Programmable logic device; System-on-Chip (SoC)

CLC index TN47

DOI 10.1007/s11767-014-4090-x

I. Introduction

Field Programmable Gate Arrays (FPGAs) are pre-fabricated silicon devices that can be electrically programmed to implement almost any kind of digital circuit or system. As illustrated in Fig. 1, the basic structure of an FPGA consists of a sea of Logic Blocks (LBs), an interconnection network, and configurable I/O blocks. Because of very high level integration, the recent FPGA devices also include memory blocks, hardwired DSP blocks, clock management blocks, and high speed data transceiver blocks, all embedded monolithically[1−3]. Logic blocks are the main digital processing resources, and each of them is configured to perform combinational as well as sequential operations depending on the required function to implement. For the combinatorial operations, a set of LookUp Tables (LUTs) are employed as arbitrary logic-function generators, and for the sequential operations a set of D-Flip-Flops are involved. Moreover, some evolved forms of LB are optimized to be able to support additional functions, such as local storage (distributed RAM memory), Shift Register (SR), multiplexer, and adder/subtractor operations. The interconnection network is programmable by the user so as to link as many LBs as necessary[4−7].

In order to optimize FPGA performance, hardwired DSP blocks are included, which allow complex arithmetic operations to be performed. Internal memories, such as RAM, ROM, flash RAM, and shift registers, are optionally integrated to increase the processing speed and simplify the board level design of the system. The integrated clock management blocks are used to provide system synchronization synthesizable. They are usually based on Phase-Locked-Loops (PLLs), which support features such as frequency multiplication and division, propagation delay compensation and phase shift correction. The current FPGA devices also include high speed data transceiver blocks that generally consist of transmission and reception buffers. Various data communication protocols are

[1] Manuscript received date: April 9, 2014; revised date: August 23, 2014.
Supported by National Natural Science Foundation of China (No. 61271149), National High Technology Research and Development Program of China (No. 2012AA012301), and National Science and Technology Major Project of China (No. 2013ZX03006004).
Corresponding author: Yang Haigang, born in 1960, male, Professor. Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China,
Email: yanghg@mail.ie.ac.cn.

supported, including USB, Ethernet, CAN, PCI, SPI, I2C, *etc.* Furthermore, a few complex FPGA architectures include embedded microprocessors and peripherals for additional SoC support features[8–10].

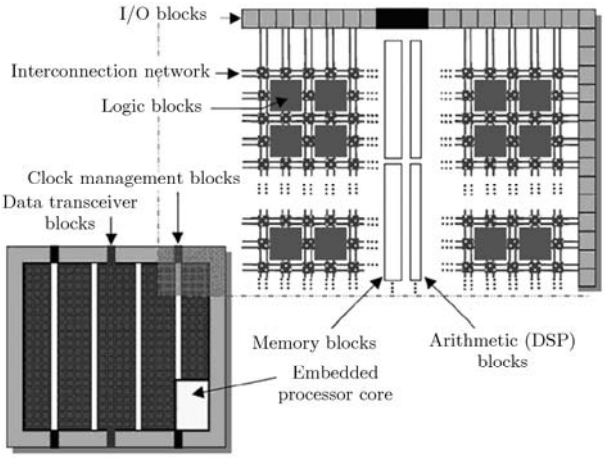


Fig. 1 Generic structure of an FPGA[11]

To provide high integration density, high-speed and low-power consumption, FPGAs have been the subject of a considerable progress in terms of advanced semiconductor process technology. Recent commercial FPGA devices, such as Altera's Arria 10 and Xilinx' Virtex UltraScale, have reached down to 20 nm process[12,13]. Moreover, Xilinx is aiming to offer Virtex UltraScale all programmable devices built on TSMC's 16 nm FinFET process technology, and Altera and Intel Corporation have jointly announced that the next generation of Altera's highest performance FPGA products (Stratix 10) would be produced using Intel's 14 nm 3D Tri-Gate transistor technology[14,15]. The development of FPGA has been greatly heightened by implementation of the advanced process. Thus unprecedented levels of performance, system integration and bandwidth can be expected in the next generations of FPGA to come.

The goal of this survey is to present the state-of-the-art FPGA architecture and to foresee future trends in FPGA design progress. The rest of this paper is organized as follows. Section II introduces the main approaches to programming technologies of FPGA. Section III addresses the design issues with regard to logic block architecture, while programmable routing architecture is described in Section IV. Section V discusses arrangement of the hardware resources embedded in FPGAs and explains about a convergence towards System-on-Chip (SoC) FPGAs. Section VI tries to explore the future technology trends of FPGAs, and a conclusion of the paper is given in Section VII.

II. Programming Technologies

An FPGA is programmed using electrically programmable switches. The properties of these programmable switches, such as size, on-resistance, and capacitance, have a significant impact on programmable logic architecture. The approaches widely used in modern FPGAs include Static Random Access Memory (SRAM), flash memory, and anti-fuse. Of these approaches, SRAM-based FPGAs are most commonly used and get a dominant position in the market, mainly due to its scalability with CMOS process technology. In this section, all these approaches of programming technologies will be reviewed, and their advantages and disadvantages are compared to provide a complete understanding of the programming technologies.

1. SRAM based programming

The basis for SRAM programming technology is the static memory cell, which is shown in Fig. 2. Access to the cell is enabled by the word line which controls the two access transistors M_5 and M_6 for both READ and WRITE operations. If the word line is not asserted, the access transistors M_5 and M_6 disconnect the cell from the bit lines, and the two cross-coupled inverters formed by M_1~M_4 will continue to reinforce each other as long as they are connected to the supply[3].

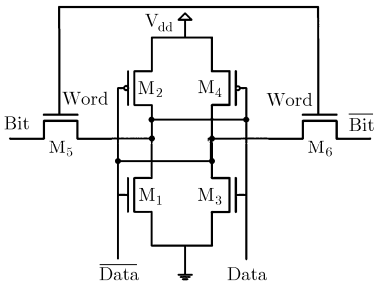


Fig. 2 Static memory cell

SRAM-based FPGAs utilize SRAM to provide

configurability for routing and computational functions, typically through the use of LUTs and multiplexers. There are two primary advantages associated, namely re-programmability by design and compatibility to standard CMOS process technology. As a result, SRAM-based FPGAs can catch up with the latest CMOS technology available and, therefore, benefit from the increased integration level, the higher speeds and the lower dynamic power consumption with smaller minimum geometries offered[16].

2. Flash/E^{2}PROM memory based programming

In the case of flash memory technology, the configuration is based on flash connections that keep the configuration state when the device is powered down. Each connection contains two transistors that share a floating gate and store the programming bit information, as Fig. 3 shows. The conductivity of the access transistor can be controlled by injecting charge onto the floating gate. Because the floating gate is electrically isolated by its insulating layer, any electrons placed on it are trapped there and, under normal conditions, will not discharge for many years.

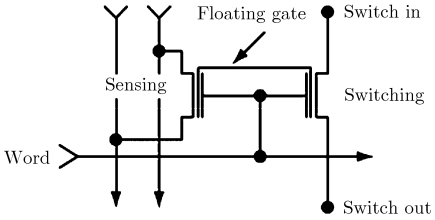


Fig. 3 Floating gate transistor[17]

Unlike CMOS based SRAM, non-volatility is one of the most important advantages for flash memory based programming technology. As a result, flash memory based FPGAs do not have to use external resources to store and load configuration data. Additionally, as all configuration and routing data retain their states when the power is off, a flash-based device can function immediately upon power-up instead of having to wait for reconfiguration. The flash approach is also more area efficient than SRAM-based technology which requires 5 or 6 transistors to implement the programmable bit storage.

The main disadvantages of flash-based devices are the limited numbers of reprogramming runs and the need for a non-standard CMOS process. Current devices such as the Microsemi IGLOO2 are only rated for 500~1000 programming cycles[18].

3. Anti-fuse based one time programming

An alternative to SRAM and floating gate-based technologies is anti-fuse programming technology. This technology is based on structures which exhibit very high-resistance under normal circumstances but can be "blown" (in reality, connected) to create a low resistance link. In contrast to SRAM or floating gate programming technologies, this link is permanent. Therefore, FPGAs based on anti-fuse programming technology can be programmed only once.

The primary advantage of anti-fuse programming technology is its small footprint[16]. With metal-to-metal anti-fuses (Fig. 4), no silicon area is required to make connections, decreasing the area needed for programmability. For the same reason, anti-fuses have lower on resistances and parasitic capacitances than in other programming technologies. Moreover, as the programming of anti-fuse based FPGAs needs only to be done once, the security for the hardware description bit stream downloaded to the FPGAs are greatly improved.

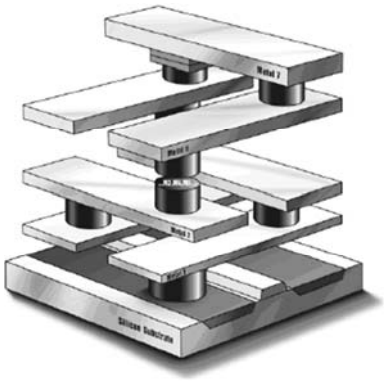


Fig. 4 Anti-fuse interconnect elements (metal- to-metal)[19]

However, there exist some disadvantages. In particular, since anti-fuse-based FPGAs require a nonstandard CMOS process, they are typically several generations behind in the manufacturing processes compared to SRAM-based FPGAs (see Tab. 1 below). Furthermore, the fundamental me-

Tab. 1 Comparison of different programming technologies

Parameters	Programming type		
	SRAM	Flash	Anti-fuse
Nonvolatile	×	√	√
Reprogrammable	√	√	×
Cell components	5 or 6 transistors	1 or 2 transistors	Metal-to-metal
Read/Write speed	1 ns/1 ns	100 ns/200 ns[20]	-
State-of-the-art process	20 nm	65 nm[18]	130 nm
Endurance (Cycles)	Infinite	500~1000[18]	1
Programming yield	100%	100%	>90%

chanism of programming, which involves significant changes to the properties of the materials in the fuse, leads to scaling challenges when new IC fabrication processes are phased in. The inability to reprogram also restricts the scope of applications. Unlike other technologies, in-system programming is not possible, which requires special programming system used to program a device before it is mounted on a final product. Finally, the one-time programmability of anti-fuses makes it impossible for manufacturing tests to detect all possible faults[16].

4. Summary

The three programming technologies reviewed in this section have different application domains in modern FPGA markets. SRAM-based programming technology has become the most widely used because of its compatibility to standard CMOS process. Flash memory based FPGAs are mainly used in some specific applications where both nonvolatile and reprogrammable technologies are required. Due to highest reliability at expenses of the flexibility, anti-fuse FPGAs are mainly used in aviation and aerospace systems. Tab. 1 summarizes the pros and cons among the different programming technologies.

III. Logic Block Architecture

FPGAs consist of logic blocks to implement logic functions, programmable routing to interconnect these functions and I/O blocks to make off-chip connections. The logic block architecture is extremely important as it lies at the heart with respect to the design of FPGAs for optimized performance and logic density[21−23]. In this section, we will discuss the basic issues and trade-offs in logic block architecture design, with a revisit to the evolution of the commercial FPGA architectures.

1. Design fundamentals

The purpose of a logic block in an FPGA is to provide the basic computation and storage elements used in digital logic systems. In this section, three most cited logic blocks, which are respectively based on LUT, multiplexer, and And-Inverter Cone (AIC), will be discussed.

(1) LUT

LUT is considered to be an array of memories which has only 1 bit of output. The logic block architecture with one LUT and DFF is shown in Fig. 5. In most cases, the truth table for a K-input logic function is stored in a $(2^K \times 1)$ SRAM. The address lines of the SRAM function as inputs and the output of the SRAM provides the value of the logic function. However, the high functionality of LUT does not come for free. They tend to be roughly large and slow, as their area grows exponentially and delay grows linearly both with the number of inputs. Also, the number of outputs is intrinsically only one, limiting their flexibility[24,25].

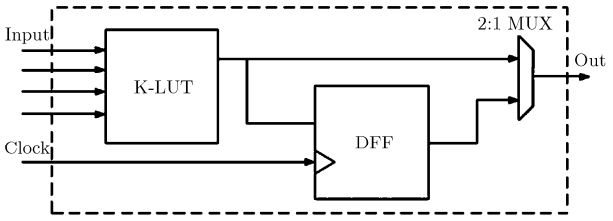


Fig. 5 LB architecture with one LUT and DFF

The mainly used commercial FPGAs are always based on clusters, such as in the Altera Stratix, Cyclone, and the Xilinx Virtex series[26−35]. A cluster is a group of Basic Logic Elements (BLEs) that are fully connected by a MUX-based cross

bar[36], as illustrated in Fig. 6.

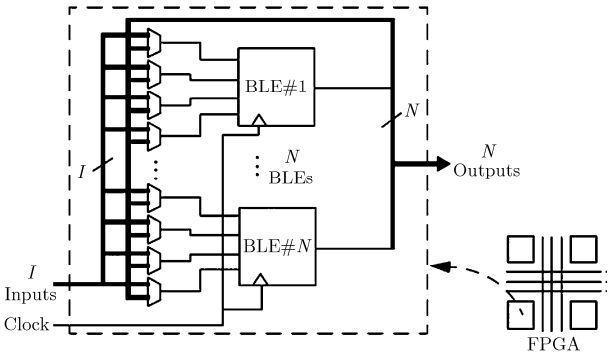
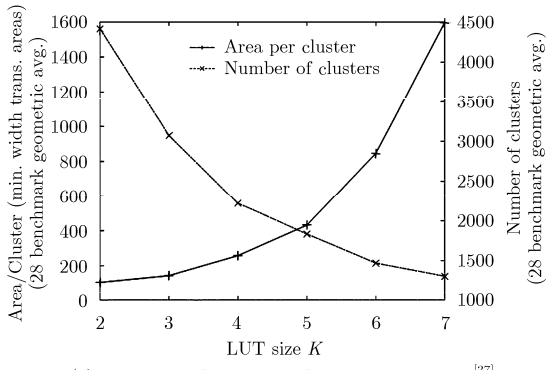
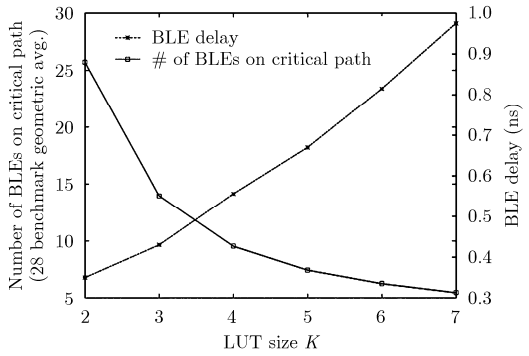


Fig. 6 Structure of Logic Cluster[36]

In most industrial FPGAs, the granularity of logic block is adjustable by using of LUTs and flip-flops. The LUT size (K, number of inputs) and cluster size (N, number of LUTs per cluster) have important effect on the area and speed performance of FPGAs. The work of Ref. [37] first explored the effect of K and N on FPGA architecture, as Fig. 7 demonstrated.



(a) Number of clusters and cluster area versus K[37]



(b) Number of BLEs on critical path and BLE delay versus K[37]

Fig. 7 Illustration the effect of LUT and BLE Characteristics

Fig. 7(a) illustrates the effect of LUT size on the number of clusters and cluster area, and Fig. 7(b) shows the trade-off between BLE delay and number of BLEs on critical path. It is understood that as the LUT and cluster size increases, the total number of logic blocks needed to implement a given function is decreased, and the number of such blocks on the critical path is decreased too. However, both the size and delay of the logic block increase with K and N. Furthermore, the area dedicated to routing outside the block will change as a function of K and N, and this has a strong effect on the results. The choice of the logic block granularity, which produces the best area-delay product, lies in somewhere between these two extremes[38-40].

(2) MUX

The functions that can be implemented by a single two-input MUX are shown in Fig. 8. In MUX based FPGAs, two-input MUXs are used to implement different logic functions by connecting each of its inputs to a constant or to a signal. Using such methods, a logic block capable of implementing a large number of functions can be constructed by grouping a number of MUXs and basic logic gates.

Multiplexer-based logic blocks have the advantage of providing a large degree of functionality for a relatively small number of transistors. This is, however, achieved at the expense of a large number of inputs, which place high demands on the routing resources. Such blocks are, therefore, more suited to FPGA's that use programmable switches of small size such as flash and anti-fuse.

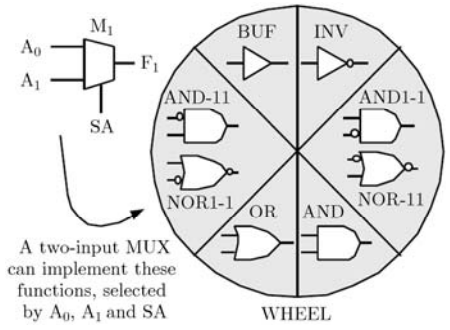


Fig. 8 Functions implemented by a two-input MUX[41]

(3) AIC

The And-Inverter Cone (AIC) is a full binary tree of cells which can be configured as 2-input

NAND or AND gates. The architecture of the AICs is inspired from the And-Inverter Graphs (AIGs)[42], where all nodes are 2-input AND gates with an optional inversion at the output. However, in this original AIC architecture, inverters are not available at the inputs of the cells. Therefore, nodes whose fan-outs include both inverted and non-inverted edges cannot be represented as-is. To accommodate these cases, some AIG transformations are required, such as duplicating the node that has the fan-out or adding an inverter node at its output. This provides more flexibility and allows the AIC to map a larger subset of functions, and the architecture of the AIC is shown in Fig. 9.

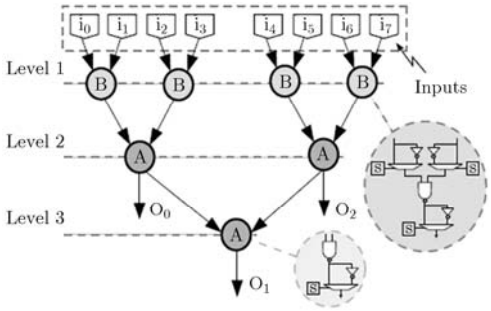


Fig. 9 AIC architecture[43]

Though unable to implement all possible functions of its inputs, the AIC logic block is considered to be less versatile but more efficient compared to LUT-based block. AICs can be richer in terms of input and output bandwidth, because their area grows linearly with the number of inputs. Also their delay grows logarithimically with the broad input and intermediate output are easier to implement.

2. Commercial product examples

In general, published research work on logic block architecture tends to model and explore relatively simple basic logic elements, such as the pure k-input LUT. In contrast, commercial logic blocks have undergone an evolution that typically has led to the development of more complex blocks in an attempt to gain more functionality.

(1) Altera

In Altera FPGA product families, Stratix and Cyclone based architectures use Logic Elements (LEs) of different types arranged into Logic Array Blocks (LABs). In Stratix I FPGA, each LAB contains some number of LEs, which is composed of a 4- input LUT, a programmable register, carry chain with carry select capability, and some other control logics, as Fig. 10 shows. With the 4-input LUT, each LE can implement any function of four variables, and it also supports single bit addition or subtraction mode dynamically selectable by a LAB-wide control signal. This device has a simple architecture which is easily mapped and synthesized in software, and is considered to be suitable for low cost FPGAs. This architecture continues to be used in Cyclone 1-4 FPGA families[27−30].

To get advanced features with efficient logic utilization, the architectures in Stratix II have evolved in great extent. Adaptive Logic Module (ALM) becomes the smallest unit of logic instead of LE, and it keeps as the main feature of LABs in the following Stratix device families. For example, the ALM in Stratix V devices consists of combinational logic, four registers, and two adders, as shown in Fig. 11. The combinational logic portion has eight inputs including two adaptive LUTs using Altera's patented LUT technology. An entire ALM is needed to implement an arbitrary six-input function. Because it has eight inputs to the combinational logic block, one ALM support implementing various combinations of two individual functions. In addition to implementing a full 6-input LUT, the ALM can, for example, implement 2 independent 4-input functions or a 5-input and a 3-input functions with independent inputs[31]. Because 4 registers and 2 adders are available, the ALM has the flexibility to implement 2.5 logic elements (LEs) of a classic 4-input LUT (4-LUT) architecture, consisting of a 4-LUT, carry logic, and a register[32]. Compared to LEs with fixed 4-input LUTs, the architecture of ALM exhibits much more functionality as well as flexibility, providing the performance superiority of larger LUTs and the area efficiency of smaller LUTs[44].

The design of the ALM shows possible trade-offs between FPGA speed performance and area cost. Altera's research results indicated that a basic 6-LUT could yield a 14% performance improvement by reducing the number of levels of logic elements on the critical paths of circuits, while this performance increase also had a large area penalty,

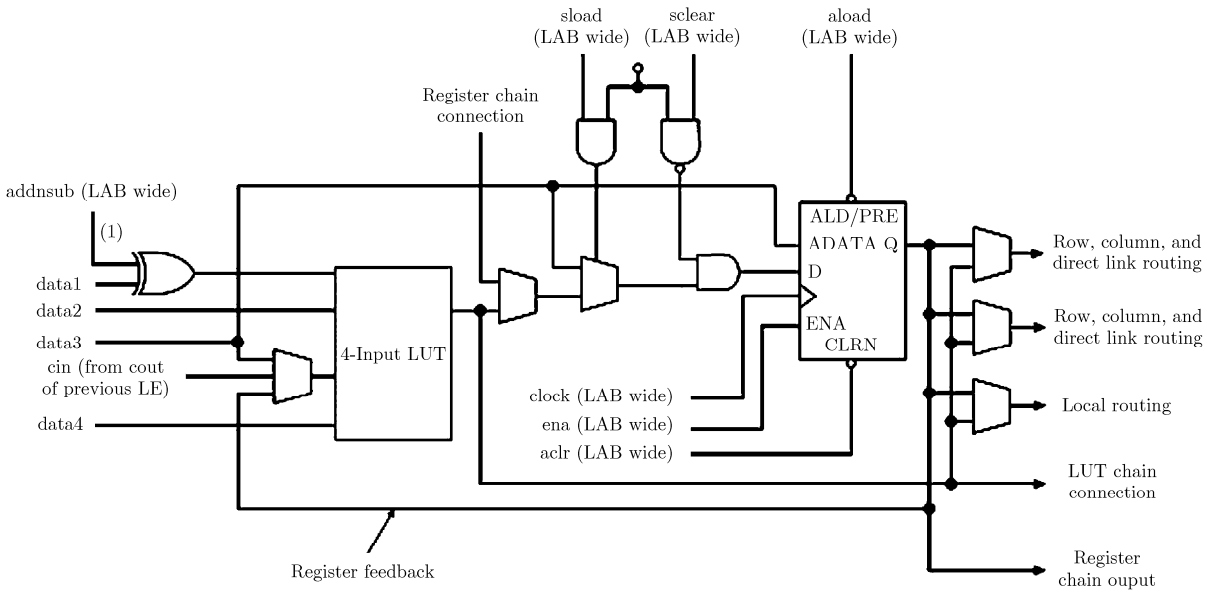


Fig. 10 LE architecture in Stratix I[26]

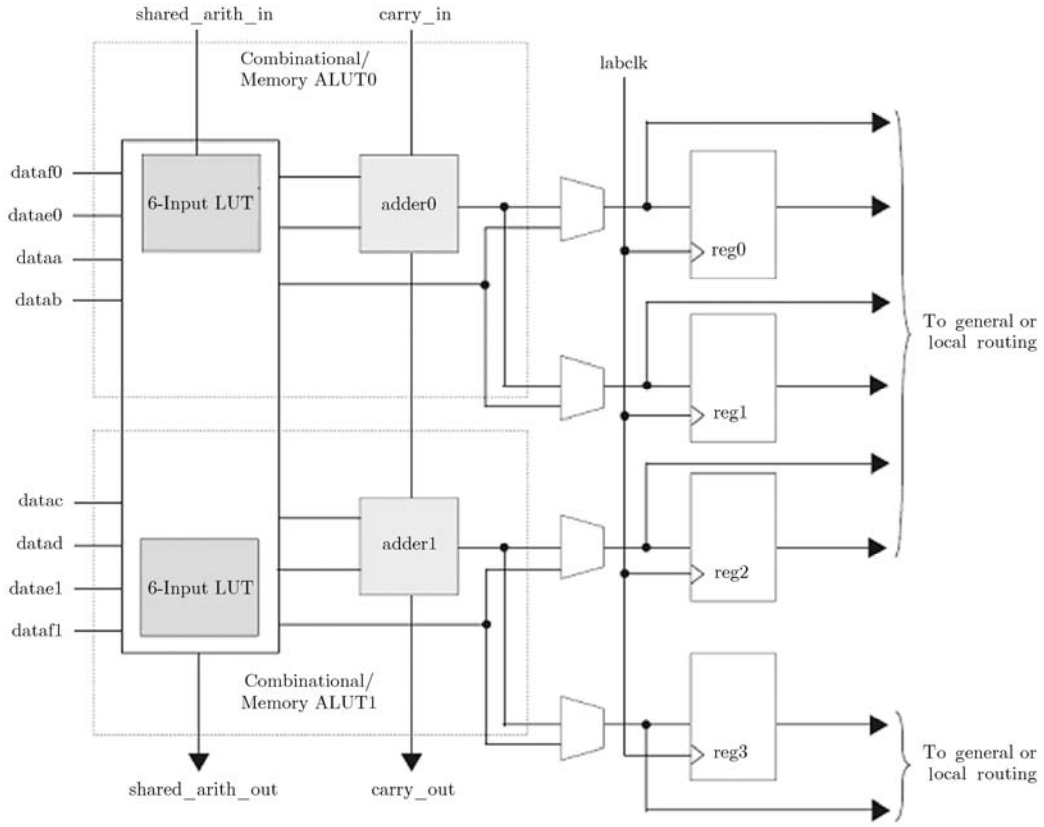


Fig. 11 ALM block diagram in Stratix V[31]

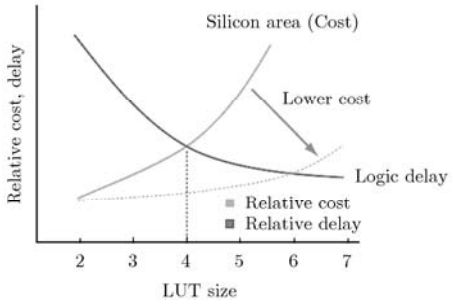


Fig. 12 Delay-cost tradeoff with LUT size in ALM design[32]

a 17% area increase resulting from a larger LUT-mask and more inputs for the LUT[32]. Fig. 12 illustrates the tradeoff between area and delay for different sizes of LUTs. The basic approach in designing the ALM was not only to investigate building a larger LUT to reduce levels of logic and hence increase performance, but also to avoid the area increase by efficiently dividing the larger LUT into smaller LUTs when appropriate, as illustrated by the dashed line. The ability to divide an LUT is what makes it "adaptive."

(2) Xinlix

In most Xinlix FPGA devices, the Basic Logic Element (BLE) is based on LUTs with fixed-inputs. Take the early device XC3000 as an example (Fig. 13), each BLE contains a 5-inputs LUT, which can implement any arbitrarily defined 5-input Boolean function, or two arbitrarily defined 4-input Boolean functions, as long as these two functions share common inputs. However, at that time (around 1995~2000) the EDA tools was not sophisticated for highly efficient synthesis and placement of this complex function generator, and it was replaced by architectures based on 4-input LUTs in Virtex 1-4 FPGAs. This change could be seen as a compromise between area and performance.

Xilinx proposed Virtex-5 family FPGAs in 2007, in which Configurable Logic Blocks (CLBs) based on 6-input LUTs started to be used as basic elements. The CLB contains two slices, each organized as a column, and contains four 6-input LUTs, four storage elements, some multiplexers, and carry logic. These elements are used by all slices to provide logic, arithmetic, and ROM functions. The function generators are implemented as six-input LUTs, and each can be configured as either a 6-input LUT with one output, or as two 5-input

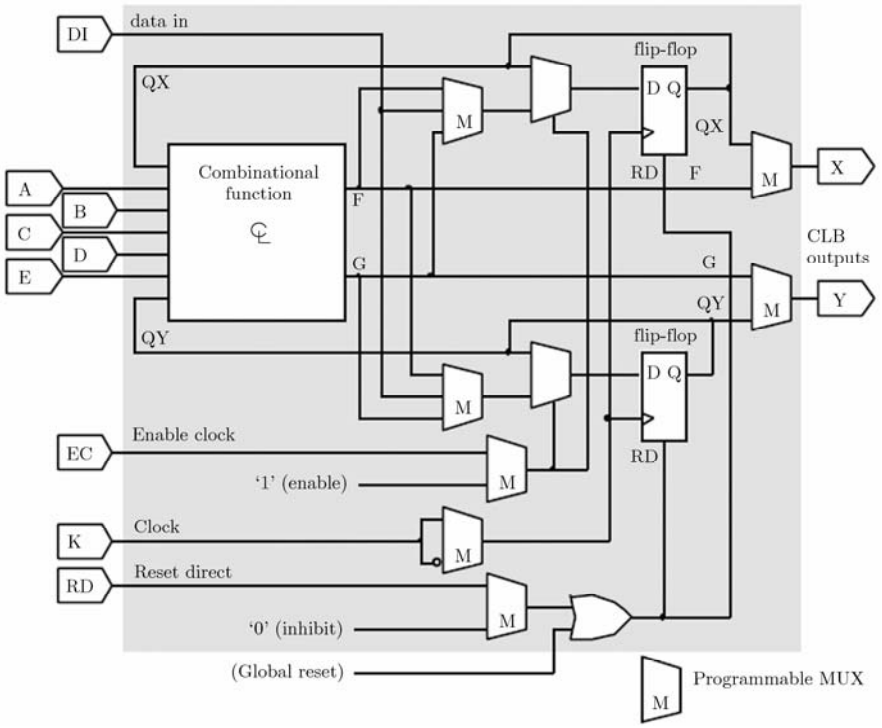


Fig. 13 Diagram of basic logic element of XC3000[41]

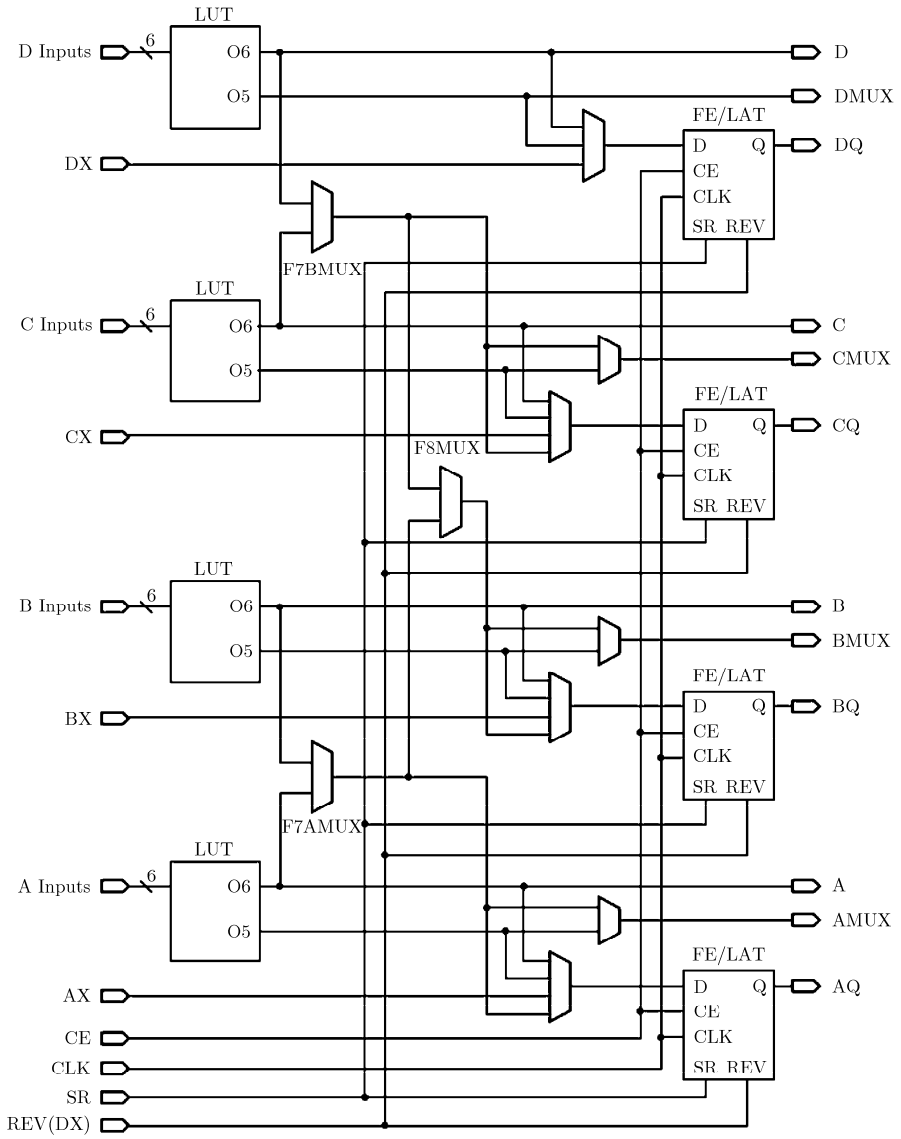


Fig. 14 Diagram of slice in Virtex-5 FPGA[33]

LUTs with separate outputs but common addresses or logic inputs. Each 5-input LUT output can optionally be registered in a flip-flop, and 4 flip-flops in each slice can optionally be configured as latches. The diagram of slice in Virtex-5 is shown in Fig. 14.

In addition, some slices in Virtex-5 support extra functions such as storing data using LUTs and shifting data with 32-bit registers[33]. These slices are called SLICEM, in which the function generators (LUTs) can be implemented as synchronous RAM resource called distributed RAM element. Multiple LUTs in a SLICEM can be combined to store larger amount of data, up to 256 bits. In general, distributed RAM is more efficient in terms of resources and performance to implement memories that consist of 64 bits or less, thus provide a trade-off between using storage elements for very small arrays and block RAM for larger arrays[34]. A similar function can be found in the Memory LAB (MLAB) of Stratix IV, where each ALM can be configured as 64 bits memory block[7].

The comparison between the different LUT architectures of the two main FPGA vendors has caused a series of arguments. It is announced by Altera that the flexible ALM architecture is advanced compared to the fixed 6-inputs LUT ar-

chitecture of Xilinx's Virtex-5 devices, and each ALM could get equivalent performance to 1.8 logic elements based on 6-inpus LUT[45]. However, Xilinx pronounced that the performance of ALM only equal to 1.2 times of the architecture based on fixed 6-inputs LUT, while taking much more cost in terms of area. As a result, the Virtex-5 devices still contain a higher capacity of logic resources than their counterpart Stratix III devices[46]. Whatever the truth of these arguments present, taking LUTs with multiple inputs as the basic unit of FPGA has became the mainstream in the design of FPGA architectures.

IV. Routing Architecture

FPGA architecture consists of programmable logic elements and a programmable routing fabric. The programmable routing provides connections among logic blocks and I/O blocks to complete a user-designed circuit. It consists of wires and programmable switches that form the desired connections. To accommodate a wide variety of circuits, the interconnect structure must be flexible enough to support widely varying local and distant routing demands together with the design goals of speed performance and power consumption. In commercial architectures, the routing consumes most of the chip area, and is responsible for most of the circuit delay. As FPGAs are migrated to more advanced technologies, the routing fabric becomes even more important. Thus, there has been a great deal of interest in developing efficient FPGA routing architectures. In this section, the discussion to be given will focus exclusively on the FPGA's general purpose routing.

1. Overall routing

A basic issue in FPGA design is the organization of the global routing architecture, which is the macroscopic allocation of wires. Based on the arrangement of the logic and interconnect resources, the routing architecture of FPGAs are broadly categorized into the following three main types: island-style routing architecture, channel-style routing architecture and hierarchical routing architecture. Nowadays, island-style is the mainstream of the routing architectures used in commercial FPGAs. Channel-style and hierarchical routing architecture can be found in some early FPGA devices, such as Microsemi ACT, SX, and MX family FPGAs for channel-style[47], and Altera Flex10K, Apex, and Apex II FPGAs for hierarchical routing architecture[48−50]. For the rest of this section, we will focus on the design issues of island-style routing architecture.

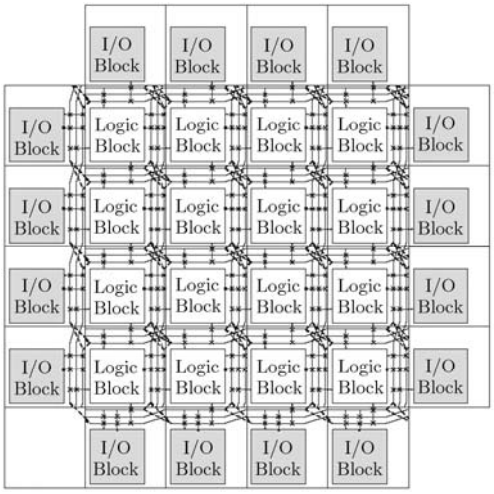


Fig. 15 Example of island style FPGA[51]

Island style routing architecture consists of an array of programmable logic blocks connected via vertical and horizontal programmable routing channels, as Fig. 15 shows[51]. It typically has routing channels on all four sides of the logic blocks, and the input or output of one logic block can connect to the routing channels with the connection blocks. Island-style routing architectures generally employ wire segments of different lengths for flexible interconnection planning, and the horizontal and vertical routing channels are connected at every intersection with programmable switch block.

2. Detailed routing

The detailed routing architecture of FPGA defines the logical structure of interconnection between each other of wire segments and I/O blocks. Switch blocks are used to form connections between these wire segments at every intersection of the channels. In the rest of this section, the discussion will focus on optimizing switch patterns, routing channel segmentation and bi-directional routing design in FPGAs.

(1) Switch box

Switch block is a programmable interconnect block at the intersection of each of the horizontal and vertical channels, which programmably connects incoming track to a number of outgoing tracks. Clearly, the flexibility of each switch block is essential to the overall flexibility and routability of the device. Since the transistors in the switch block add capacitance loading to each track, the switch block has a significant effect on the speed of each routable connection, and hence the speed of the FPGA as a whole. In addition, since such a large portion of an FPGA is devoted to routing, the chip area required by each switch block will have a large effect on the achievable logic density of the device. Thus, the design of a good switch block is of the up-most importance.

Fig. 16 shows three previous switch block architectures that have been proposed[52−55]. In each block, an incoming track can be connected to three outgoing tracks, but the topology of each block is different. The disjoint switchbox has been used in a number of commercial FPGAs, such as the Xilinx XC4000 family[41]. As seen in Fig. 16(a), the connection pattern is "symmetric" in the disjoint block, which means a wire entering a disjoint switch block can only connect to other wires with the same numerical designation via programmable switches. As a result, routes in the FPGA are isolated into distinct routing domains, limiting routing flexibility. The Wilton switch block in Fig. 16(b) is similar to the disjoint switch block, except that each diagonal connection has been "rotated" by one track[56]. This eliminates the domains problem of disjoint switch block, and results in many more routing choices for each connection. In addition to the Wilton and disjoint switch blocks, a number of alternative designs, such as the Universal switch block shown in Fig. 16(c), have also been suggested[57]. The design focus of the Universal block is on maximizing the number of simultaneous connections that can be made using this block, and it does not taking into account interactions between neighboring switch blocks. A full review of additional island-style switch blocks optimized for length 1 wire segments can be found in Ref. [58].

Each of the blocks in Fig. 16 was developed and evaluated assuming the architecture with only single-length wires (*i.e.* wires that only connect neighboring switch blocks). In reality, FPGAs, however, typically have longer wires which connect distant switch blocks. Such a routing architecture is called a segmented architecture, and it is known that such architectures lead to a higher density and speed than architecture with only single-length wires[59].

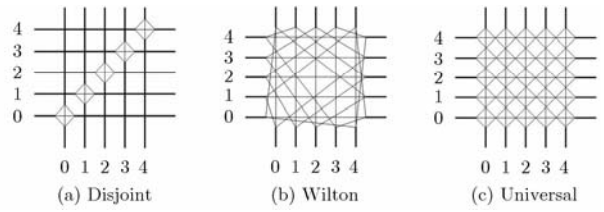


Fig. 16 Previous switch blocks[52−55]

The majority of recent switch block designs only allow switch connections between wire endpoints or between wire midpoints, but not between endpoints and midpoints. Midpoint-to-midpoint connections are made using single disjoint connections. Examples include the Imran[56] and shifty[60] switch blocks, as shown in Fig. 17 and Fig. 18. The Imran switch block uses a Wilton switch block to connect endpoints of wires and single-transistor disjoint connections to connect midpoints. This switch block has been shown to be more area efficient than disjoint, universal, or Wilton switch blocks[56]. The shifty switch block similarly allows for routing domain changes on endpoint turns and disjoint connections at midpoints. Experimentation has shown that shifty and Imran switch blocks give similar area and delay results[60]. Both switch blocks are superior to disjoint switch blocks in area and delay performance due to their ability to allow for diverse routing paths.

(2) Channel segmentation

In the design of FPGA routing architecture, wire segment is defined as the number of logic blocks that a routing wire spans before terminating. All FPGAs use routing channel segmentation, whereby each routing track is divided into wire segments with different length. Studies have shown that the mix of segment lengths used in different routing tracks can have a significant impact on interconnect performance[6,58], and hence the overall FPGA performance.

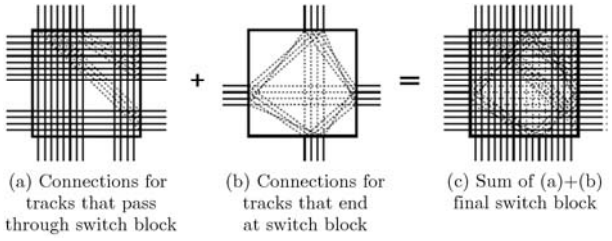


Fig. 17 Imran switch block[60]

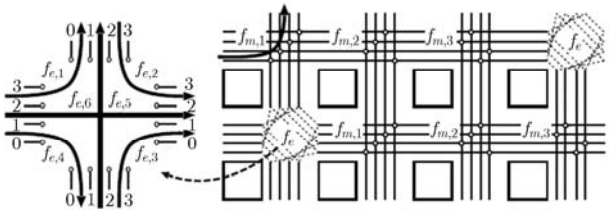


Fig. 18 Shifty switch block[60]

Conceptually, routing architecture with shorter segments results in better routability and lower excess net loading, which means higher logic density and lower power consumption. However, in an FPGA with too many short wires, some long connections will have to be constructed using several short wire segments connected in series, resulting in longer delays. On the other hand, the speed performance of the routing architecture can be improved by increasing longer segments, but at the expense of lower logic density and higher power consumption. Moreover, it was observed in Ref. [61] that with CMOS technology scaling, the average segment length should decrease because of the increase in wire parasitics relative to device parasitic. Given these tradeoffs and observations, the routing channel segmentation should be chosen carefully to optimize the overall FPGA performance.

In Ref. [62], the design of segmented routing channel was first discussed for row-based FPGAs. The approach of constructing a segmented routing channel to get high routability is shown, assuming both random origination points and geometrically distributed connection lengths. In Ref. [63], this statistical approach was extended to island-style FPGAs, where empirical distributions for segment lengths were first determined by statistically analyzing placed and routed designs, and then separate horizontal and vertical channel segmentations were found according to the demand for each segment length.

Empirical methods were also attempted. In Ref. [59], Betz *et al.* used a contemporary FPGA router which combines global and detailed routing into one step to evaluate segmentation. It is shown in Fig. 19 that among channels of equal length segments, a channel with only length-4 segments achieves the lowest routing area and shortest critical path delay. Moreover, a routing channel with a mixture of length-4 and 8 segments can outperform a channel arrangement with only length-4 segments. Similarly in Ref. [64], optimal uniform segmentation was investigated experimentally for 100 nm process FPGA, and their results have shown that using length-3 segments leads to the lowest energy consumption as well as energy-delay-area product. These studies verified the importance of including significant medium length segments which span between several logic blocks in an island-style routing architecture, and it is validated during the development of the Stratix architecture[65], which contains significant length 4 and length 8 segments.

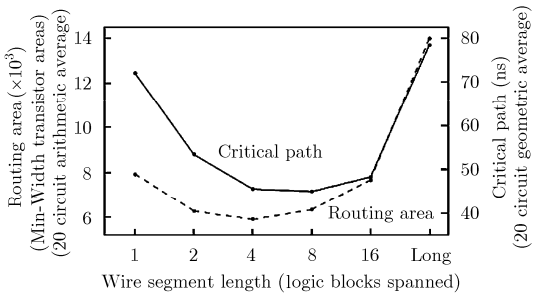


Fig. 19 Speed and area of FPGAs vs. routing wire segment length[59]

In addition to connection pattern and quantification parameters, FPGA detailed routing architecture performance is governed by the types of switches used to make connections, the size of transistors used to build programmable switches and the metal width and spacing of interconnection wires[59]. Routing switches are typically made from collections of basic transistor structures including pass transistors, buffers, and multiplexers.

(3) Bi-directional or unidirectional

Fig. 20 shows the basic structures of bi-directional and unidirectional routing. In bi-directional routing, the output of the cluster tile is connected via a buffer, and alternatively, in unidirectional

routing the output is connected to a multiplexer with other wires (from both cluster outputs and other wires in the channel), and one buffer then drives the output of this multiplexer.

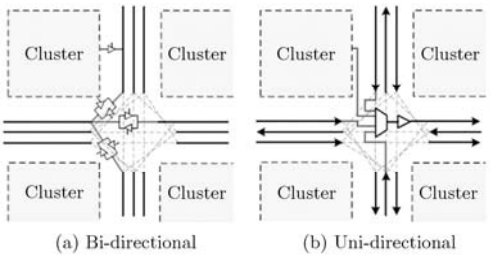


Fig. 20 Bi-directional and uni-directional basic structure[66]

Based on the routing structure just discussed, some contrast between uni-directional and bi-directional routing architectures can be observed. First, in terms of area, uni-directional routing consumes less area than bi-directional routing due to buffer sharing facilitated by the multiplexer/driver routing switch. This effect is shown by Lemieux[67] and Lewis[68], and in their works, it is indicated that the overall area required to implement a uni-directional routing architecture was found to be about 20% less than the corresponding bidirectional equivalent. Second, the uni-directional routing nets connect to fewer switching points, resulting in less capacitive load. But for the situation that a routing path goes through at least one switch box, bi-directional routing architecture has fewer programmable switches compared to uni-directional routing. In modern FPGA routing design, uni-directional is used for most of the tracks, and bi-directional routing is only used for the long interconnects across several switch blocks to restrain from expanding with regard to the number of such long interconnects.

3. Asynchronous interconnect

As the logic size of FPGA grows, there are some challenges faced by conventional FPGA architectures. First, the delays of the long interconnect wires can easily dominate all other delays. Second, to evenly distribute the global clock signals all over the FPGA area requires great design efforts because of the clock skew. Third, FPGAs are more likely to contain a multitude of modules running at different clock frequencies since they have grown to sufficient die sizes. Data signals appear to be asynchronous in the new clock domain when moving data across modules[69].

Introducing asynchronous concept into the FPGA architecture is a possible solution to the named challenges. In terms of interconnect delays, performance is dictated by the average of the interconnect delays rather than the worst-case delay. By adopting asynchronous design, FPGAs can provide architectural supports for communications across different clock domains. Different modules running at different clock frequencies can be easily glued together.

Achronix's FPGA is typically implemented with asynchronous interconnect, and Fig. 21 illustrates the principle of Achronix FPGA's asynchronous

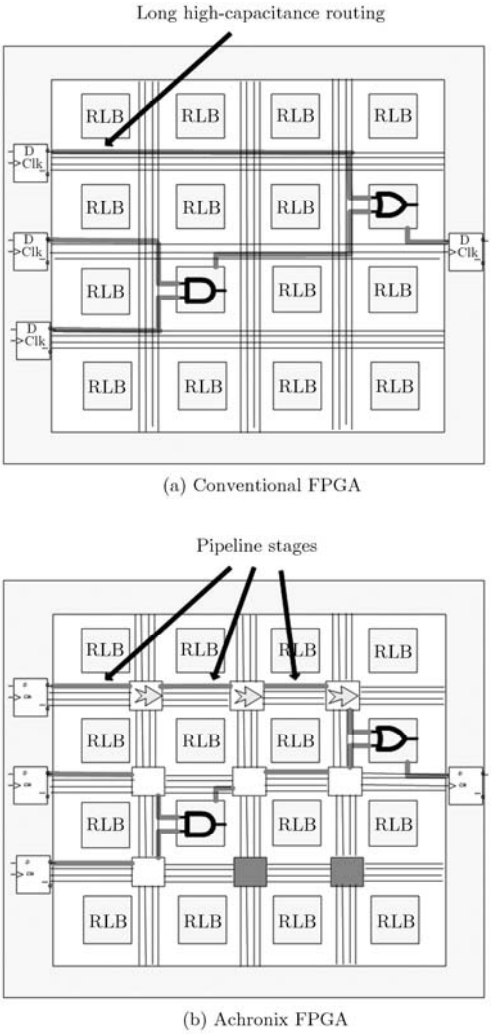


Fig. 21 Conventional implementation vs. asynchronous implementation[71]

routing architecture. Within a traditional FPGA, signals which travel on long routing tracks suffer from a high capacitive load, and things get worse when the FPGA become larger in size. Within asynchronous FPGAs, the built-in pipelining ensures that signals only ever need to travel on short routing tracks, decreasing the capacitance load at each stage. Moreover, asynchronous architecture ensures there is only one logic level per pipeline stage, allowing a much faster rate of data tokens[70].

4. 2.5 dimensional interconnect

The Xilinx Stacked Silicon Interconnect (SSI) technology is a 2.5 dimensional integration of multiple active devices on a passive interposer to form a single device, as illustrated in Fig. 22. It is implemented to answer the challenges that obstructed attempts to combine the interconnect logic of two or more FPGAs to create a larger FPGA for implementing a complex design. These challenges mainly include: the amount of available I/O is insufficient for connecting; the latency of signals passing between FPGAs limits performance; and power consumption is increased by using standard device I/O to create logical connections between multiple FPGAs.

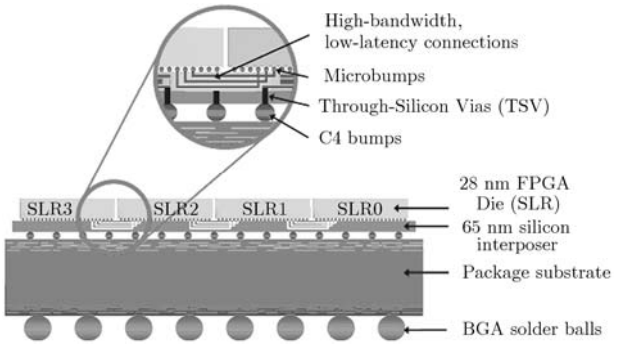


Fig. 22 Stacked Silicon Interconnect cross section[72]

To rise to the challenges, the SSI technology uses passive silicon interposers with microbumps and Through-Silicon Vias (TSVs) to combine multiple FPGA die in a single package, and enables high- bandwidth connectivity between multiple dies by providing a great number of connections. It takes lower latency and consumes lower power than the multiple FPGA approach, while enabling the integration of massive quantities of interconnect logic, transceivers, and on-chip resources. In addition, it is of extra importance at the early stages of a new process node, as the defect densities are still quite high and die yield declines dramatically with the increasing of die size. This also helps in maximizing the functional performance of each chip .

Fig. 22 shows the side view of the die stack-up with four FPGA slices, silicon interposer, and package substrate. The silicon interposer acts as an interconnect vehicle based on a silicon manufacturing process on which multiple dies are set side by side and interconnected. The key innovation is to augment the standard I/Os with thousands of die-to-die connections through passive traces fabricated on the silicon interposer, which provides high connectivity and low latency without incurring the power penalty of traditional I/O structures. Besides this, SSI technology avoids the power and reliability issues that result from stacking multiple FPGA dies on top of each other. Compared to organic or ceramic substrates, silicon interposers are considered to offer better geometries of interconnection (approximately 20X denser wire pitch) to provide device-scale interconnect hierarchy that enables, in theory, up to 10,000 die-to-die connections[73].

V. Embedded Resources

With the increasing amount and improved performance of functional modules, such as DSP, memory, clock manager, high speed transceiver, and so on, FPGA has become an imperative system-enable device. The inclusion of these embedded resources have great influence on the performance of modern FPGAs. Moreover, the main commercial vendors tend to provide a series of FPGA platforms rather than a single FPGA product, with varying feature mix optimized for different application domains[74]. How to get a rapid and cost-effective assembly of FPGA platforms with different functional modules, has become a sought after issue of FPGA design. In this section we will discuss this problem and give a broad view on the design features of Altera and Xilinx.

Xilinx created the so-called Advanced Silicon Modular Block (ASMBL) architecture to enable rapid and cost-effective assembly of FPGA plat-

forms with different features[75]. A high-level description of this architecture is illustrated in Fig. 23. In this architecture, logic resources, such as the Configurable Logic Block (CLB), DSP, memory and so on, are arranged in columns and can be selected for targeted applications. Customers can choose FPGA platforms with the right mix of features and capabilities for their specific design. The main advantage of ASMBL architecture is the elimination of geometric layout constraints, which means any kind of hard IP blocks can be scaled independent of surrounding resources, and power and ground can be placed anywhere on the chip[76].

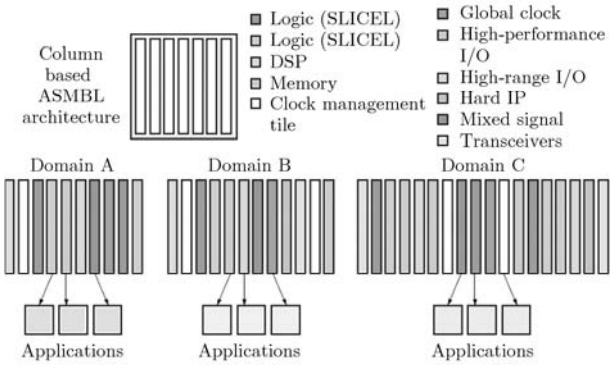


Fig. 23 Column-based resources of ASMBL architecture[77]

Altera devices comprise an array of LABs interconnected by rows and columns of routing wires. The LAB architecture naturally creates a tall and narrow layout. As Fig. 24 shows, functional modules, such as MegaRAMs, are organized as rectangle blocks and have a quite larger width, rather than columns in Virtex. These modules could not be assembled and removed directly like the ASMBL architecture, and restart of synthesis and placement is needed to carry out different FPGA platforms. For the customers, this design flow means widely-expanded choice with "personalized" FPGA platforms, while for the FPGA designers, more experience of customization is needed, as well as much more powerful software and EDA tools to support rapid synthesis and placement[65].

Another important architecture of the embedded resources can be found in SoC FPGAs, where both microprocessor and FPGA are integrated into a single device. Melding the two technologies provides a variety of benefits including higher integration, lower power, smaller board size, and higher bandwidth communication between them. Besides this, SoC FPGA system has complete flexibility to select any combination of peripherals and controllers, and it shows good ability to make tradeoffs between hardware and software to maximize efficiency and performance. Altera Arria family devices and Xilinx Zynq family devices are typical examples of this architecture[8,9].

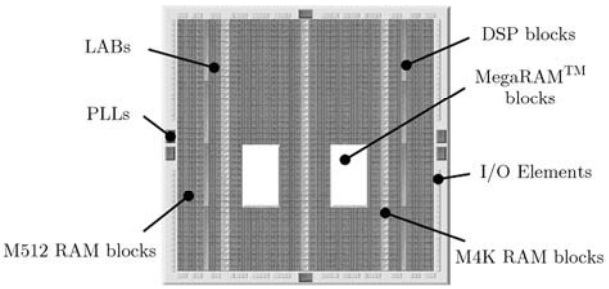


Fig. 24 Overview of Stratix floorplan[65]

VI. Future Trends of FPGA

In the field of FPGA development, persistent concerns are placed on area, speed and power, which are considered to be ever-existing design issues for FPGAs. Besides, there are some other emerging technologies that may have impact on the future trends of FPGA, such as novel memory based FPGAs and 3D FPGAs. This section will present a brief introduction.

1. Novel memory technologies

Over the past decade, the semiconductor industry has experienced a resurgence of interest in the search for highly scalable memory technologies. As novel nonvolatile memory technologies are fast progressing, there has been a growing interest on investigating their use in future FPGAs.

Lots of research efforts have been done to develop a novel memory technology for FPGAs which combines the desired merits of nonvolatility and high performance. Kryder introduced 13 alternative (NVM) technologies which are evaluated with respect to density, device performance, and likelihood of success in 2020[20]. Among these new memory technologies, Phase-Change RAM (PCRAM), Spin-Torque Transfer RAM (STTRAM) and Resistive RAM (RRAM) are most promising candidates and hence have received a lot of atten-

tions[78−83]. This section will discuss these alternative memory technologies as well as their prospect for future FPGAs.

(1) PCRAM

PCRAMs utilize a reversible phase change between the amorphous and the crystalline states of a chalcogenide glass to produce a reversible resistance change in the cell[84]. Fig. 25 shows the schematic cross-sectional view of the PCRAM cells. The structure consists of the top and bottom electrodes of TiW material, the SiO_2 dielectric layer as the isolator, and $Ge_2Sb_2Te_5$ phase change material as the active layer. The crystalline and amorphous states have low and high resistance, respectively, for data 0 and 1, and a heater element is used to generate heat to switch between states.

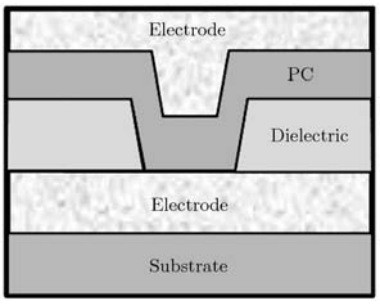
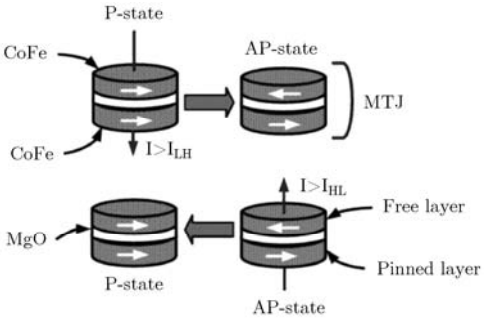


Fig. 25 Schematic cross-section of PCRAM cell[85]

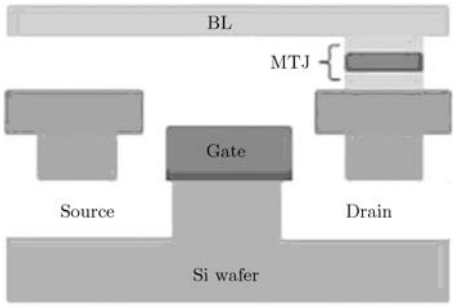
PCRAM is a nonvolatile memory technology with speed comparable to SRAM, capacity comparable to DRAM, almost unlimited endurance, and write speed faster than flash by several orders of magnitude, and yet they require only three to four additional masks[78]. It is considered to be one of the most promising candidates for future FPGAs. The main challenge in PCRAMs is controlling the resistance variations during manufacturing and after read/write cycles.

(2) STTRAM

The basic building block of an STTRAM cell is the Magnetic Tunneling Junction (MTJ) (see Fig. 26). Each MTJ consists of two ferromagnetic layers separated by a very thin tunneling dielectric film. Switching MTJ states from antiparallel or "1" to parallel or "0" and vice versa is performed by running a polarized electron current from the top to the bottom of the MTJ and vice versa[86]. The direction of magnetization of free layer can be controlled by the injection of spin-polarized electrons. Hence, the MTJ can be switched between two stable magnetic states with high or low resistances.



(a) Change in magnetization states for an MTJ



(b) Cell structures of STTRAM[87]

Fig. 26 Schematic of STTRAM

A scheme for sensing the resistance of the MTJs in case of FPGAs has been proposed in Refs. [88] and [89]. The principle idea is to use two MTJ elements for storing one-bit information, and then, employing a sense amplifier for reading out the stored configuration. Just like PCRAM, STTRAM also has the advantage of nonvolatility and small cell size. Besides this, compared to PCRAM, STTRAM has faster read/write speed and a larger number of write cycles. The key challenge in an STTRAM-based reconfigurable framework is how to sense the resistance state for each configuration bit during normal functioning of the FPGA. Sense-amplifier-based data read-out mechanisms, commonly employed in embedded STTRAM memories, will incur large overhead when used in an FPGA framework.

(3) RRAM

RRAM cells are capacitor-like structures that exhibit a resistive switching phenomenon in transition metal oxides. This memory conception is

based on electric pulse-induced reversible resistance change effect, and has a kind of metal-insulator-metal sandwich architecture, which is shown in Fig. 27. The metallic ions in the insulator are solid state electrolyte, which can be collected into filaments when the electrical bias provided, inducing a very low resistance. When the electrode bias is reversed, the filaments shrink or disappear, exhibiting a high resistance. The high resistance value can be obtained more than 1000 times than the low resistance. Furthermore, the resistance value can be maintained constant even if removing the applied voltage.

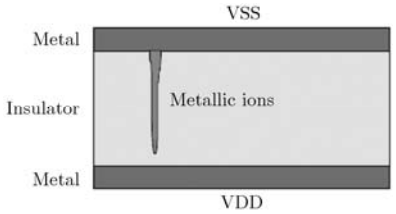


Fig. 27 RRAM structure

RRAM architecture has been proposed to realize FPGA. M. Liu proposed a new FPGA conception called rFPGA[90]. In rFPGA, the conventional architectures, such as Switch Block (SB), Connection Block (CB) and block memory, are substituted by RRAM. As shown in Fig. 28, the 2T1R (two transistors and one resistor) RRAM are used to substitute the conventional programmable switches in CB and SB, and the 1T1R RRAM are used to substitute the 6T SRAM cell in block memory. Simulation results show that the routing resource complexity can be decreased and the channel delay will be reduced 2~3 times by using RRAM to realize CB and SB. The block memory's area will be reduced 5~6 times by using RRAM. The improved architecture can also reduce power consumption because it employs less number of transistors. Other research on improving the integration level is focused on substituting 2-D RRAM with 3D RRAM[91].

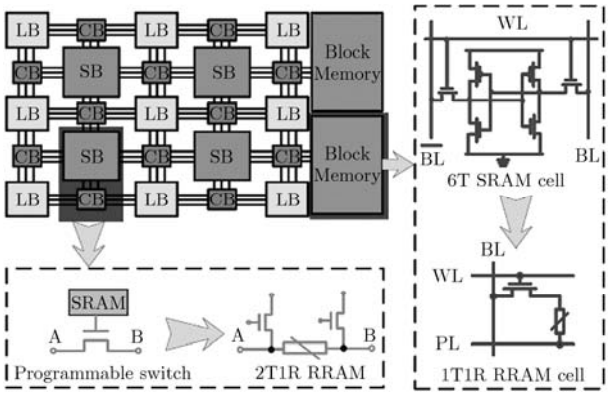


Fig. 28 FPGA utilizing RRAM components as memory and routing elements[92]

(4) Summary

Tab. 2 summarizes features of the proposed emerging memory technologies along with SRAM and Flash, which are included for comparison purposes. The cell sizes of all memory technologies in units of minimum feature size F were projected based upon the Emerging Research Devices (ERD) chapter of the 2007 International Technology Roadmap for Semiconductors (ITRS), which contains a tabulation of the recent experimental values as reported in literature.

2. 3D FPGA

3D FPGA, which represents another important FPGA developing trend, is based on three dimensions integrated circuit technologies. As FPGA chips grow in complexity, the problems such as interconnect delay, clock synchronization and IR

Tab. 2 Comparison of embedded memory technologies[20,78,87,93]

Device type	Cell structure	Cell components*	Minimum cell size (F^2, F=feature size)	Incremental masks	Read/write speed (ns)	Endurance (Cycles)
SRAM	Latch	6T	24	0	1/1	Infinite
Flash	Floating	1T-2T	8-12	6-8	25/200	10^4~10^5
PCRAM	Phase	1T1R	6-10	3-4	20/50	$>10^{12}$
STTRAM	Magneto	1T1MTJ	6-10	3-4	10/10	$>10^{15}$
RRAM	Memristor	1T1R	4	3-4	10/20	10^5~10^6

*T = transistor; C = capacitor; R = resistance/phase-change element

drop in power grid have become more severe than ever. 3D technologies based on Through Silicon Vias (TSVs) is considered to be the most promising candidate to solve such problems.

M. Lin from Stanford University has proposed an optimal structure of 3D FPGA based on chips stacked and utilized Through Silicone Vias (TSVs) as interconnect channels. Fig. 29 shows a typical 3D FPGA structure, which extends traditional 2-D FPGA to 3 layers: memory on the top, switch in the middle and Logic on the substrate layer. The advantage of this structure is easy for integrating logic and SRAM IP. Compared with Virtex-II, this stacked 3D FPGA has 3.2 times logic density integration with area reduced to 31%.

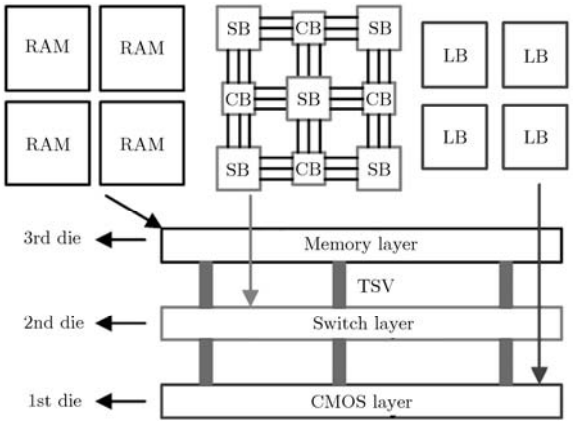
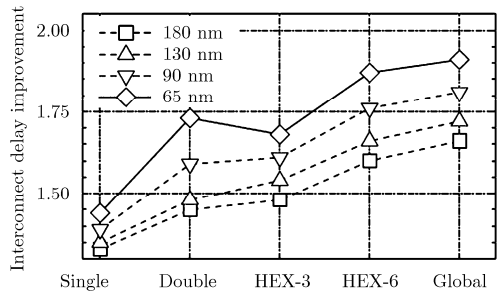


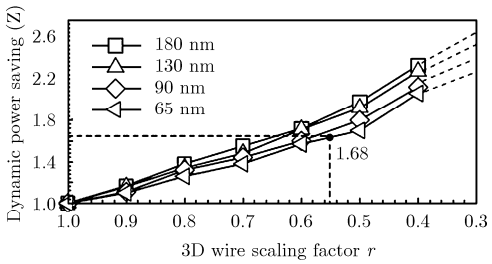
Fig. 29 Monolithically stacked 3D-FPGA

Estimation of the 3D FPGA for each of the four technology nodes (180 nm, 130 nm, 90 nm, and 65 nm) in ITRS[93] using versatile (VPR), see Fig. 30, shows that the geometric average of the pin-to-pin delay has improved range between 1.7 and 2.05, and between 1.31 and 2.14 for the critical path delay, and 1.68 dynamic power consumption saving over a conventional 2D FPGA.

Although 3D FPGA has overwhelming advantage compared with conventional 2-D FPGA, less of stability in 3D fabrication process and lack of supported EDA tool hampers its application. Fortunately, along with the 3D technique being developed and place and route tool in 3D FPGA being reported[94], it has been envisaged that 3D FPGA could probably be industrialized in the coming 5 years.



(a) Interconnect delay (One (referred to as Single), two (Double), three (HEX-3), and six (HEX-6) are sets of FPGA tile width segments in addition to interconnects that span the entire array width (Global))



(b) Relative power saving (Factor $0<r<1$ represents the ratio of the 3D-FPGA tile width to the baseline 2D-FPGA)

Fig. 30 Improvements for different submicron technology nodes[95]

VII. Brief discussions on Current Development of FPGA Research in China

Generally speaking, research and development of FPGA in China is in the phase of fully catching up[96−102]. With respect to the architecture design, some endeavor is worthily mentioned. For example, Fudan University has developed an evaluation system for FPGA architecture based on the tools of VPR, to estimate the power, area and timing information[103]. A delay-estimated method based on statistical model was also proposed, help optimizing FPGA interconnect architecture[104]. To improve the flexibility of the routing resources, a novel switch box, called Minimum-Loop Maximization (MLM) switch box was developed by maximizing the minimum loop size in the routing-resource graph[105].

The "Comet" serials FPGAs[106,107] have been developed by System on Programmable Chip Research Department of Institute of Electronics, Chinese Academy of Sciences, and comprehensive research into FPGA architectures were performed onto this platform. Through empirical methods, the relationship between Flexibility of connection (Fc) of a generic FPGA and its performance was

analyzed, optimizing the flexibility of FPGA architecture[108]. Moreover, the routability of switch box and connection box were modeled, and simulated annealing method was employed to maximize the information entropy of the switch distribution in order to improve routability[109]. These research achievements have been successfully applied to "Comet 02" mega-gates FPGA chip, which was the first domestic large scale FPGA product launched into space with satellites.

VIII. Conclusion

The aim of this paper is to provide a comprehensive insight into advanced FPGA architectural design. FPGA technology has been greatly progressed by the rapid development of semiconductor technology, while at the same time the designers face new challenges due to the undesired effects caused by deep submicron and nano-meter process. To keep pace with today's high-end market demands, FPGA devices are being developed towards high levels of performance, system integration and bandwidth. In addition, with the increasing amount and improved performance of functional modules, FPGA has become an imperative system-enable device, and microprocessors were integrated into FPGA devices to provide a comprehensive platform base for next-generation systems. For the outlook into the future, it is quite promising that the design of FPGAs will remain an exciting and dynamic technology for the years to come, and the emerging technologies might be hybrid together to provide tremendous opportunities to develop the future System on Chip FPGA platforms.

References

[1] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes. Features, design tools, and application domains of FPGAs. *IEEE Transactions on Industrial Electronics*, **54**(2007)4, 1810–1823.

[2] M. Slimane-Kadi, D. Brasen, and G. Saucier. A fast-FPGA prototyping system that uses inexpensive high-performance FPIC. In Proceeding of 2nd Annual Workshop on FPGAs, Berkeley, CA, USA, 1994, 1–6.

[3] S. M. Trimberger. Field-Programmable Gate Array Technology. USA, Kluwer, 1994, Chapters 1–4.

[4] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, **12**(2004)3, 288–298.

[5] Xilinx. Virtex-6 Family Overview. DS150 (v1.0), February 2, 2009.

[6] V. Betz, J. Rose, and A. Marquardt. Architecture and CAD for Deep-Submicron FPGAs. Berlin, Kluwer Academic Publishers, 1999, Chapters 1–5.

[7] Altera Corporation. Stratix IV Device Handbook. March 2009.

[8] Altera Corporation. Architecture Matters: Choosing the Right SoC FPGA for Your Application. November 2013.

[9] Xilinx. Zynq-7000 All Programmable SoC Overview. March 2013.

[10] Microsemi. SmartFusion2 System-on-Chip FPGAs. January 2014, Revision 5.

[11] E. Monmasson, L. Idkhajine, and M. Cirstea, *et al*.. FPGAs in industrial control applications. *IEEE Transactions on Industrial Informatics*, **7**(2011)2, 224–243.

[12] Altera Corporation. Arria 10 Device Handbook. June 2013.

[13] Xilinx. Xilinx UltraScale Architecture for High-Performance, Smarter Systems. December 2013.

[14] Xilinx. Xilinx UltraScale: The Next-Generation Architecture for Your Next-Generation Architecture. July 2013.

[15] Altera Corporation. Expect a Breakthrough Advantage In Next Generation FPGAs. June 2013.

[16] I. Kuon, R. Tessier, and J. Rose. FPGA architecture: survey and challenges. *Electronic Design Automation*, **2**(2007)2, 135–253.

[17] Microsemi. ProASIC3 Flash Family FPGAs with Optional Soft ARM Support. January 2013, Revision 13.

[18] Microsemi. IGLOO2 FPGAs. June 2013.

[19] Microsemi. Axcelerator Family FPGAs. Revision 18, March 2012.

[20] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**(2007)2, 203–215.

[21] Yang Hai-gang, Sun Jia-bin, and Wang Wei. An overview to FPGA device design technologies. *Journal of Electronics and Information Technology*, **32**(2010)3, 714–727.

[22] A. M. Smith, G. A. Constantinides, and P. Y. K. Cheung. FPGA architecture optimization using geometric programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **29**(2010)8, 1163–1176.

[23] F. Barranco, M. Tomasi, J. Diaz, et al.. Architecture for hierarchical optical flow estimation based on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(2012)6, 1058–1067.

[24] E. Ahmed and J. Rose. The effect of logic block granularity on deep-submicron FPGA performance and density. Master's thesis, University of Toronto, Department of Electrical and Computer Engineering, 2001.

[25] R. Tessier, V. Betz, D. Neto, and T. Gopalsamy. Power-aware RAM mapping for FPGA embedded memory blocks. In Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, 2006, 189–198.

[26] Altera Corporation. Stratix Device Handbook. January 2006.

[27] Altera Corporation. Cyclone Device Handbook. May 2008, ver. 3.1.

[28] Altera Corporation. Cyclone II Device Handbook. February 2008, ver. 4.0.

[29] Altera Corporation. Cyclone III Device Handbook. August 2012, ver. 4.2.

[30] Altera Corporation. Cyclone IV Device Handbook. February 2013, ver. 1.8.

[31] Altera Corporation. Stratix V Device Handbook. June 2013.

[32] Altera Corporation. FPGA Architecture. July 2006, ver. 1.0.

[33] Xilinx. Virtex-5 FPGA User Guide. March 2012, ver. 5.4.

[34] Xilinx. UltraScale Architecture Configurable Logic Block. December, 2013, ver. 1.0.

[35] Xilinx. 7 Series FPGAs Configurable Logic Block. August 2013, ver. 1.5.

[36] V. Betz and J. Rose. Cluster-based logic blocks for FPGAs: area-efficiency vs. input sharing and size. IEEE Custom Integrated Circuits Conference, Santa Clara, CA, USA, 1997, 551–554.

[37] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. In Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays, USA, ACM Press, 2000, 3–12.

[38] R. Lysecky and F. Vahid. A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning. Design, Automation and Test in Europe, Munich, Germany, 2005, 18–23.

[39] P. Biswas, S. Banerjee, and N. Dutt. Performance and energy benefits of instruction set extensions in an FPGA soft core. 19th International Conference on VLSI Design and Held jointly with 5th International Conference on Embedded Systems and Design, India, 2006, 1–5.

[40] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**(2007)2, 203–215.

[41] Michael John and Sebastian Smith. Application-specific integrated circuits. US, Electronic Industry Press, January 2003.

[42] A. Mishchenko, S. Chatterjee, and R. Brayton. DAG-aware AIG rewriting: a fresh look at combinational logic synthesis. In Proceedings of the 43rd Design Automation Conference, San Francisco, CA, USA, July 2006, 532–536.

[43] Hadi Parandeh-Afshar, Hind Benbihi, *et al.*. Rethinking FPGAs: elude the flexibility excess of LUTs with and-inverter cones. ACM/SIGDA 20th International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, February 2012, 119–128.

[44] D. Lewis, D. Cashman, Mark Chan, *et al.*. Architectural enhancements in Stratix-V™. ACM/SIGDA 21st International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, February 2013, 147– 156.

[45] Altera Corporation. Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison. September 2007, ver. 2.0

[46] Xilinx. Advantages of the Virtex-5 FPGA 6-Input LUT Architecture. December 2007, ver. 1.0.

[47] V. George and J. M. Rabaey. Low-Energy FPGAs: Architecture and Design.US, Kluwer Academic Publishers, US, 2001.

[48] Altera Corporation. FLEX 10K embedded programmable logic device family. DS-F10K-4.2 January 2003.

[49] Altera Corporation. APEX 20K programmable logic device family data sheet. DS-APEX20K-5.1, March 2004.

[50] Altera Corporation. APEX II programmable logic device family. DSAPEXII-3.0, August 2002.

[51] V. Betz and J. Rose. FPGA routing architecture: Segmentation and buffering to optimize speed and density. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, February 1999, 140–149.

[52] Xilinx Inc.. The Programmable Logic Data Book. 1994.

[53] G. G. Lemieux and S. D. Brown. A detailed router for

allocating wire segments in field-programmable gate arrays. ACM Physical Design Workshop, Reston Sheraton, US, April 1993, 1–8.

[54] Y. W. Chang, D. Wong, and C.Wong. Universal switch modules for FPGA design. *ACM Transactions on Design Automation of Electronic Systems*, **1**(1996)1, 80–101.

[55] S. J. E. Wilton. Architectures and algorithms for field-programmable gate arrays with embedded memory. Ph.D. thesis, University of Toronto, 1997.

[56] M. I. Masud and S. Wilton. A new switch block for segmented FPGAs. International Workshop on Field Programmable Logic and Applications, US, August 1999, 274–281.

[57] H. Fan, J. Liu, Y. L. Wu, and C. C. Cheung. On optimal hyperuniversal and rearrangeable switch box designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **22**(2003)12, 1637–1649.

[58] G. Lemieux and D. Lewis. Design and Interconnection Networks for Programmable Logic. Boston, MA, US, Kluwer Academic Publishers, 2004.

[59] D. Marple and L. Cooke. An MPGA compatible FPGA architecture. IEEE Custom Integrated Circuits Conference, Boston, MA, US, 1992, 421–424.

[60] G. Lemieux and D. Lewis. Analytical framework for switch block design. International Symposium on Field Programmable Logic and Applications, Monterey, CA, US, September 2002, 122–131.

[61] M. Lin, A. El Gamal, Y. C. Lu, and S. Wong. Performance benefits of monolithically stacked 3D FPGA. *IEEE Transaction on Computer -Aided Design Integrated Circuits and System*, **26**(2007)2, 216–229.

[62] A. El Gamal, J. Greene, V. Roychowdhury, *et al.*. Segmented channel routing in nearly as efficient as channel routing (and just as hard). University of California/Santa Cruz Conference, Berkeley, CA, US, 1991, 192–211.

[63] W. K. Mak and D. F. Wong. Channel segmentation design for symmetrical FPGAs. International Conference on Computer-aided Design, Kasuga, Japan, 1997, 496–501.

[64] C. G. Wong, A. J. Martin, and P. Thomas. An architecture for asynchronous FPGAs. IEEE International Conference on Field-Programmable Technology (FPT), HK SAR China, 2003, 170–177.

[65] David Lewis, Vaughn Betz, *et al.*. The StratixTM routing and logic architecture. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, February 2003, 41–48.

[66] P. Jamieson, W. Luk, *et al.*. An energy and power consumption analysis of FPGA routing architectures. International Conference on Field-Programmable Technology (FPT), Sydney, Australia, December 2009, 324–327.

[67] G. Lemieux and D. Lewis. Directional and single-driver wires in FPGA interconnect. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, December 2004, 41–48.

[68] D. Lewis, *et al.*. The Stratix II logic and routing architecture. In Proceeding: ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, Februery 2005, 14–20.

[69] John Teifel and Rajit Manohar. An Asynchronous Dataflow FPGA Architecture. *IEEE Transactions on Computers*, **53**(2004)11, 1376–1392.

[70] Achronix Corporation. ACE User Guide. December 2012, ver. 5.0.

[71] Achronix Corporation. Introduction to Achronix FPGAs. August 2008, Rev. 1.6.

[72] Xilinx. Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency. December 2012, ver. 1.2.

[73] L. Madden, E. Wu, *et al.*. Advancing high performance heterogeneous integration through die stacking. European Solid-State Circuits Conference (ESSCIRC), Bordeaux, France, September 2012, 18–24.

[74] Kurt Keutzer, Sharad Malik, A. Richard Newton, *et al.*. System-level design: orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **19**(2000)12, 1523–1543.

[75] Xilinx. Virtex-4 FPGA User Guide. UG070 (v2.6), December 2008.

[76] Xilinx. Virtex-4: Breakthrough Performance at the Lowest Cost. December 2004.

[77] Xilinx. Virtex-5 Platform FPGA Family Technical Backgrounder. May 2006.

[78] S. Natarajan, S. Chung, L. Paris, and A. Keshavarzi. Searching for the dream embedded memory. *IEEE Solid-State Circuits Magazine*, **1**(2009)3, 34–44.

[79] Y. Guillemenet, L. Torres, and G. Sassatelli. Non-volatile run-time field-programmable gate arrays structures using thermally assisted switching magnetic random access memories. *IET Computer Digital Technology*, **4**(2010) 3, 211–226.

[80] S. Paul, S. Mukhopadhyay, and S. Bhunia. Circuit and architecture co-design approach for hybrid CMOS-STTRAM non-volatile FPGA. *IEEE Trans-*

actions on. Nanotechnology, **10**(2011)3, 385–394.

[81] Jason Cong and Bingjun Xiao. FPGA-RPI: A novel FPGA architecture with rram-based programmable interconnects. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, **22**(2014)4, 864–877.

[82] S. Tanachutiwat, M. Liu, and W. Wang. FPGA based on integration of CMOS and RRAM. *IEEE Transactions on Very Large Scale Integrity System*, **19** (2011)11, 2023–2032.

[83] O. Turkyilmaz, J. Figueras, and Y. Zorian. RRAM-based FPGA for ACM normally off, instantly on ACM applications. International Symposium on Nanoscale Architecture, US, 2012, 101–108.

[84] A. Pirovano. Electronic switching effect in phase-change memory cells. IEEE International Electron Devices Meeting, Washington, D.C., US, 2005, 923–926.

[85] R. Zhao and L.P. Shi. Study of phase change random access memory (PCRAM) at the nano-scale. Non-Volatile Memory Technology Symposium, Albuquerque, New Mexico, US, 2007, 36–39.

[86] M. Hosomi. A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM. International Electron Device Meeting, Washington, D.C., US, 2005, 473–476.

[87] Somnath Paul and Swarup Bhunia. A circuit and architecture codesign approach for a hybrid CMOS-STTRAM nonvolatile FPGA. *IEEE Transactions on Nanotechnology*, **10**(2011)3, 385–394.

[88] W. Zhao, E. Belhaire, Q. Mistral, E. Nicolle, T. Devolder, and C. Chappert. Integration of spin-RAM technology in FPGA circuits. International Conference Solid-State Integrated Circuit. Technology, US, 2006, 799–802.

[89] N. Bruchon, L. Torres, G. Sassatelli, *et al.*. New non-volatile FPGA concept using magnetic tunneling junction. IEEE Computing Society Annie Symposium Emery on VLSI Technology Architecture, US, 2006, 269–276.

[90] M. Liu and W. Wang. Rfpga: CMOS-Nano hybrid FPGA using RRAM components. IEEE International Symposium on Nanoscale Architectures, Anaheim, CA, US, June 2008, 93–98.

[91] J. Zhang, Y. Q. Ding, *et al.*. A 3D RRAM using stackable 1TXR memory all for high density application. International Conference on Communications, Circuits and Systems, CA, US, July 2009, 917–920.

[92] Hai-Gang Yang. Overview: Emerging technologies on giga-scale FPGA implementation, 2010 IEEE International Symposium on Circuits and Systems (ISCAS), France, July 2010, 1428–1431.

[93] G. Hariharan, R. Chaware, L. Yip. *et al.*. Assembly process qualification and reliability evaluations for heterogeneous 2.5D FPGA with HiCTE ceramic. Electronic Components & Technology Conference, Las Vegas, NV, USA, November 2013, 904–908.

[94] C. Ababei, H. Mogal, and K. Bazargan. Three-dimensional place and route for FPGAs. IEEE Asia and South Pacific Design Automation Conference, Shanghai, China, January 2005, 773–778.

[95] Mingjie Lin, A. El Gamal, Yi-Chang, *et al.*. Performance benefits of monolithically stacked 3D FPGA. *IEEE Transactions on Computer Aided Design of Integrated Circuit and Systems*, **26**(2007)2, 216–229.

[96] Xie Ding, Lai Jinmei, and Tong Jiarong. Research of efficient utilization routing algorithm for Current FPGA. *Chinese Journal of Electronics*, **19**(2010)1, 48–52.

[97] Zhu Limin, Bian Jinian, Zhou Qiang, *et al.*. Integer programming based routing algorithm fro hierarchical FPGAs. *Journal of Computer-Aided Design & Computer Graphics*, **22**(2010)10, 1687–1694.

[98] Zhang Kun, Zhou Huabing, Stanley L. Chen, *et al.*. Technology mapping for FPGA with multi-mode logic cell. *Journal of Computer-Aided Design & Computer Graphics*, **21**(2009)10, 1375–1380.

[99] Zhang Huiguo, Tang Yulan, Yu Zongguang, and Tao Yufeng. High performance FPGA LUT design and implementation. *Research & Progress of SSE*, **29** (2009)4, 584–588.

[100] Xu Hanyang, Wang Jian, and Lai Jinmei. A FPGA prototype design emphasis on low power technique. In Proceedings of the 2014 ACM/SIGDA 22nd International Symposium on Field Programmable Gate Arrays, Monterey, CA, US, February 2014, 147–150.

[101] Gao Haixia, Yang Yintang, and Dong Gang. Theoretical analysis of effect of LUT size on area and delay of FPGA. *Chinese Journal of Semiconductors*, **26** (2005)5, 893–898.

[102] Gao Haixia, Ma Xiaohua, and Yang Yintang. Accurate interconnection length and routing channel width estimates for FPGAs. *Chinese Journal of Semiconductors*, **27**(2006)7, 1196–1200.

[103] Chen Yuanfeng, Tang Pushan, Lai Jinmei, *et al.*. Evaluation System for FPGA. *Journal of Fudan University* (*Natural Science*), **45**(2006)4, 523–528.

[104] Zhen Wang, Ding Xie, Jinmei Lai, *et al.*. FPGA interconnect architecture exploration based on a statistical model. International Conference on Field-Pro-

grammable Logic and Applications, Greece, 2011, 447–452.

[105] Yu Jiande, Xie Ding, Shao Yun, *et al*.. Mini-loop maximization method on FPGA routing resources architecture design. *Journal of Computer-Aided Design & Computer Graphics*, **22**(2010)6, 934–942.

[106] System on Programmable Chip Research Department of Institute of Electronics, Chinese Academy of Science. COMET02 Device Handbook. January 2010, Ver. 1.1.

[107] System on Programmable Chip Research Department of Institute of Electronics, Chinese Academy of Science. ER2C1000-G Handbook. June 2013, Ver. 1.0.

[108] Xingzheng Li, Haigang Yang, and Hua Zhong. Use of VPR in design of FPGA architecture. International Conference on Solid-State and Integrated Circuit Technology, Shanghai, China, 2006, 1880–1882.

[109] Li Wei, Yang Haigang, and Gong Xiao. Optimal design of topological structure for FPGA connection box based on information entropy. *Journal of Computer-Aided Design & Computer Graphics*, **21**(2009)2, 203–208.