FPGA Architectural Research: A Survey

STEPHEN BROWN
University of Toronto

Over the past ten years, new types of field-programmable devices, especially sophisticated architectures such as those of FPGAs and CPLDs, have greatly influenced digital circuit design. The author evaluates currently proposed architectures for area and speed performance.

TODAY'S FIELD-PROGRAMMABLE gate array products are extremely sophisticated, to the point where state-of-the-art FPGAs are among the largest integrated circuits currently on the market. For example, the Altera Flex 10K100 has about 10 million transistors and measures about 1.8 cm×1.5 cm in 0.5-micron technology. Such advances are the result of intensive research and development in both industry and academia.

A recent article in *IEEE Design & Test of Computers*[1] summarized the classes of field-programmable devices currently available and described many of the most important commercial devices. Here we describe current research studies, evaluating the enhancements to FPGA architecture each recommends and how these architectures affect the two most important metrics: total chip area and speed performance. We also note examples of commercial products possessing features consistent with the recommendations of the research studies.

Overall approach

The studies that we discuss follow the general approach shown in Figure 1. The methodology is an experimental one in which researchers propose and then study a new FPGA architecture. To experiment on the architecture, researchers must also develop computer-aided design tools to map circuits into the proposed chips. They determine the architectural parameters (logic-block complexity, interconnect flexibility, and so on), create the CAD tools, map benchmark circuits into the hypothetical chips, and then evaluate the architecture's performance. As Figure 1 shows, researchers carry out the experiments iteratively, continually adjusting the architecture and CAD tools.

Recent FPGA research developments

Some of the results we discuss next find implementation in current FPGA products, and others may influence future architectures.

Logic-block complexity. The first FPGA architecture studies reported in research publications concerned the complexity of an FPGA's logic block.[2,3] The basic idea was to study how much circuitry a single logic block should be able to implement. For practical reasons, these studies assumed a logic block to be a lookup table (LUT) memory and defined logic-block complexity as simply the number of inputs K to the LUT.

The Rose et al. study[2] then determined the effect of K on both the area required and the speed performance of implemented circuits.

0740-7475/96/$05.00 © 1996 IEEE

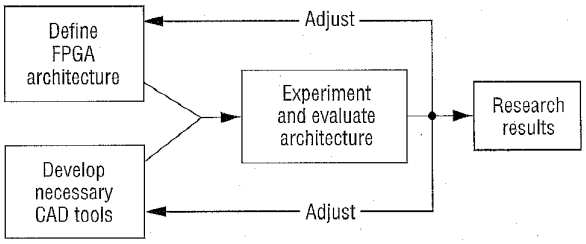

Figure 1. Approach to FPGA research.

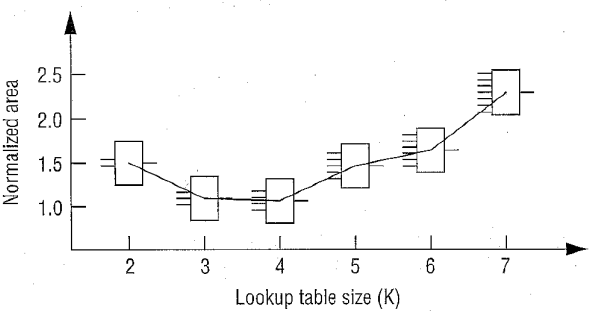

Figure 2. Effect of logic-block functionality on FPGA area.

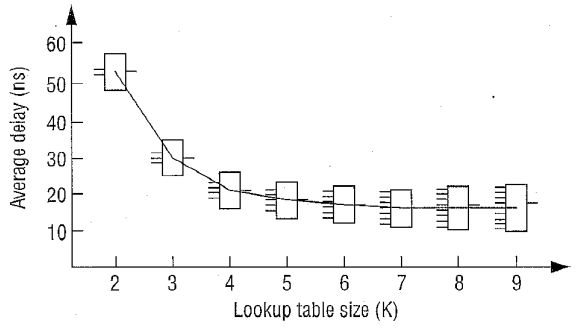

Figure 3. Effect of logic-block functionality on speed.

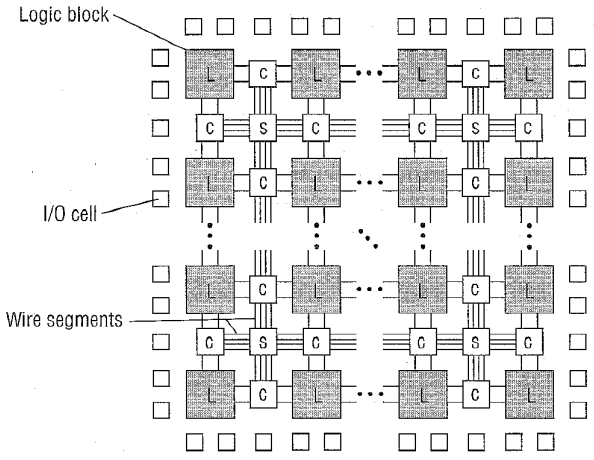

Figure 4. Model of FPGA routing structures.

Figures 2 and 3 summarize results for area and speed performance. Figure 2 shows the relative total area needed in an FPGA for different values of K, assuming identical logic blocks. The FPGA had a minimum area for a K of 4, and so the other data points of Figure 2 are normalized to those results. For instance, the figure shows that the $K = 2$ implementation requires 1.5 times more area; for $K = 7$, twice as much. These results reflect two factors:[2] As K increases, the total number of LUTs needed decreases; at the same time, the area per LUT increases. At a given point, the number of LUTs times the area per LUT is a minimum.

Figure 3 shows how different values of K affect speed performance. The vertical axis represents critical-path delays averaged over a set of circuits. Very small LUTs perform poorly, and larger logic blocks result in better performance, up to a point: Improvements diminish beyond Ks of 5 or 6. To summarize, the results of this logic-block complexity study indicate that LUTs should have about 4 inputs to optimize area. For speed performance, about 5 inputs is best. The Altera Flex 8000 is a commercial FPGA with the recommended value of $K = 4$.

FPGA interconnect flexibility. Besides its logic blocks, the other fundamental parameter that determines an FPGA's architecture is the interconnect structure. Rose and Brown[4] first studied interconnects in FPGAs. The study assumed an FPGA to have the structure illustrated in Figure 4 and sought to determine the number and organization of routing switches and wire segments in the FPGA's interconnect. Interconnect wires exist in both horizontal and vertical routing channels between rows and columns of logic blocks. Routing switches appear in two places in this architecture. The C blocks house switches to connect the logic-block pins to the routing wires, and the S block switches connect one wire segment to another. This study assumed that all wire segments span only a single logic block and that joining two wires together at S blocks forms longer connections.

The study's main experiments concerned determining how many programmable routing switches (a measure of the area needed for the FPGA) to place in the C and S blocks.[4] The researchers investigated this by defining two parameters. Fc sets the number of wire segments that each logic-block pin can connect to in a C block, and Fs determines how many other wire segments a wire segment entering an S block can connect to.

Figure 5 shows the results of experiments with these parameters. Each curve in the figure corresponds to a specific value of Fs from 2 (lowest curve) to 6 (highest). The

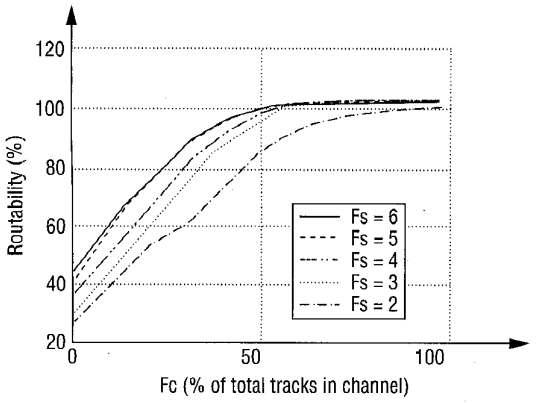

Figure 5. *Results of routing-flexibility studies.*

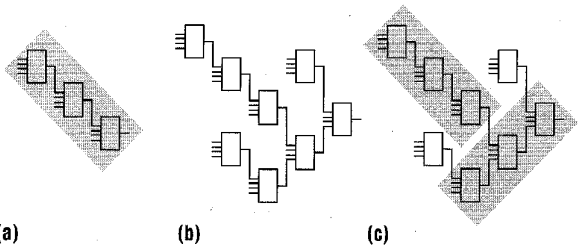

Figure 6. *Example of hardwired logic block* ***(a)****. A typical circuit mapped into eight 4-input LUTs* ***(b)*** *has four connections in its critical path; the same circuit mapped into hardwired logic block* ***(c)*** *has only one connection in the critical path.*

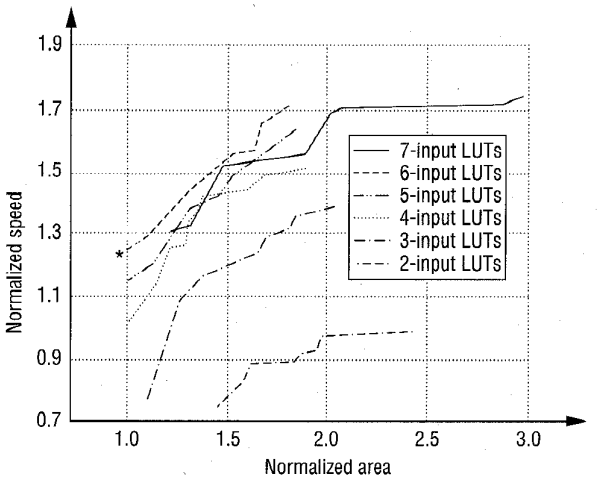

Figure 7. *Effect of hardwired logic blocks on speed and area.*

horizontal axis represents *Fc* as a percentage. At the left side of the graph, each logic-block pin cannot connect to any tracks (0%). On the right, we can programmably connect each pin to all tracks (100%). The vertical axis corresponds to the percentage of required connections (routability) in benchmark circuits that the CAD tools could successfully complete for the given values of *Fc* and *Fs*.

Clearly, for low values of *Fc*, routing circuits is difficult, but if *Fc* is at least 50% of available tracks, routability is good. Also, for all values of *Fs* except 2, as long as *Fc* is greater than 50%, routability is high. Thus, the basic conclusions reached in the study are that *Fc* should be high, and *Fs* can be greater than or equal to 3. An example of a commercial product with these characteristics is the Xilinx XC4000 series.[5]

Hardwired logic blocks. Routing delays comprise 40 to 60% of the total signal propagation time in FPGAs.[6] Thus, it is pragmatic to design FPGA architectures that avoid routing delays whenever possible. A recent study showed one way to do this using hardwired logic blocks.[7] Figure 6 illustrates the premise of the research by showing an example of a hardwired logic block and its use. Figure 6a shows three 4-input LUTs hardwired together in a cascade. These LUTs form a single hardwired logic block. Thus, circuits mapped into it can take advantage of the hardwired connections and thereby travel through fewer programmable switches. An example of this appears in Figures 6b and 6c. Figure 6b is an example circuit that we might map into eight normal 4-input LUTs. In this case, the longest path through the circuit passes through four programmable connections, and hence the circuit would traverse four switches in series.

In contrast, Figure 6c shows the same circuit mapped into hardwired logic blocks. This case would require only one programmable switch, and the circuit would be considerably faster. The main potential drawback of hardwired logic blocks is the added complexity that CAD tools would need to deal with when mapping circuits into the blocks. To quantify the effects on speed performance, we compared the hardwired speed performance of benchmark circuits to the results achievable using normal 4-input LUTs (Figure 7). The vertical axis corresponds to relative speed performance; the horizontal axis, relative area. We show a set of curves for hardwired logic blocks based on 2- to 7-input LUTs. Also, each curve includes results averaged over many hardwired arrangements and not just the simple cascade of Figure 6a.[7]

Hardwired logic blocks can provide a significant win in terms of speed performance. For example, hardwired 6-input LUTs can provide 25% higher speed at no cost in area (marked with an asterisk in the figure). A commercial product with a simple variant of hardwired logic blocks is the Xilinx XC4000; its blocks comprise two 4-input LUTs connected to a 3-input LUT.

Hierarchical FPGAs. The previous section illustrated one way of avoiding programmable switches with hardwired con-

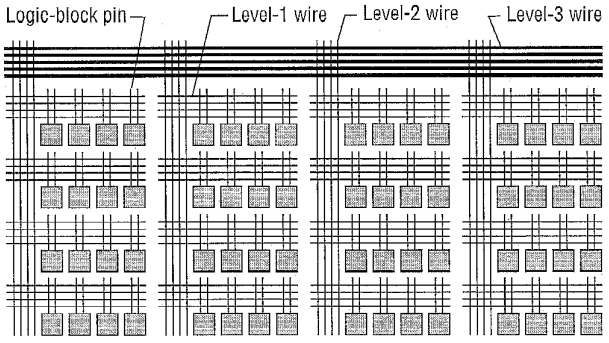

Figure 8. *Example of a hierarchical FPGA.*

nections. Aggarwal and Lewis described another way.[8] They divide an FPGA into sections and provide a set of wire segments that span each section. Logic blocks within the section can connect using a single wire segment. Longer connections use a higher level of interconnect to reach one section from another. An FPGA with this type of structure is called hierarchical. We've illustrated the concept in Figure 8, which shows an FPGA with three levels of hierarchy. At level 1 (the lowest level), four 2-input LUTs connect via level-1 wire segments. Each "section" of the chip at level 2 contains four level-1 blocks and wire segments to connect them. Level 3 comprises four level-2 blocks and appropriately long wire segments.

The most significant potential drawback of a hierarchical structure of this sort is that it introduces a "quantization problem." That is, the circuitry mapped into the chip might not partition well into the hierarchically repeated units. Despite this, hierarchical FPGAs, as opposed to flat LUT-based architectures, offer considerable savings in both area and speed performance.[8]

Altera's Flex 8000 has three levels. The lowest level, a logic array block, consists of eight 4-input LUTs. This FPGA's second level is called a row and contains a group of logic array blocks. Its highest level comprises a collection of rows.

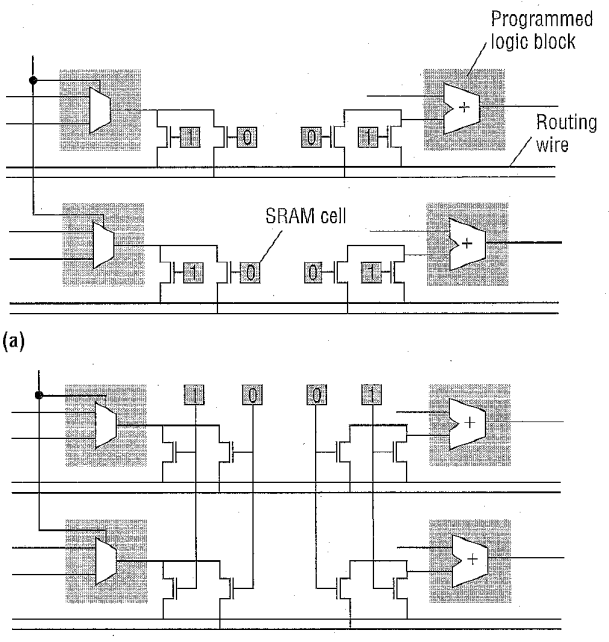

Figure 9. *Sharing of routing bits in data path FPGAs. Implementation without sharing* ***(a)*** *uses eight SRAM cells; with sharing* ***(b)****, four SRAM cells.*

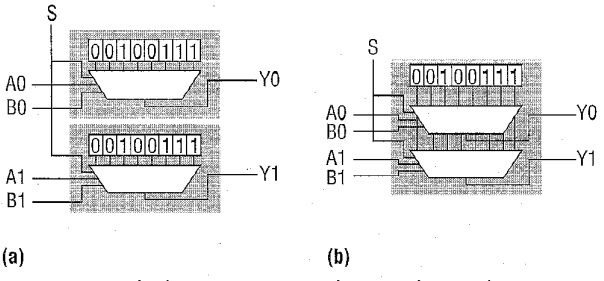

Figure 10. *Multiplexer-programmed LUT in data path FPGAs without* ***(a)*** *and with sharing* ***(b)*** *of bits used to program the LUTs.*

Data path FPGAs. Designers have traditionally designed FPGAs for general-purpose use. A different approach is to optimize a chip for a specific class of circuits. An example of this philosophy is the FPGA of Cherepacha and Lewis,[9] who proposed an architecture that takes advantage of data path circuit properties. A data path circuit manipulates a set of bits—as opposed to a single bit—by performing arithmetic, multiplexing, and other operations. We can optimize an architecture for data path circuits in the routing structures and the logic blocks.

Figure 9 shows how data paths can share routing bits. In the figure, the data paths manipulate two bits of data in the same way, passing them through (LUTs programmed as) a multiplexer and then an adder. Figure 9a shows the traditional approach, in which eight programmable switches (SRAM-controlled pass transistors in this example) route the two bits through the interconnect. (For simplicity, we assume that this FPGA has only two tracks per channel.) In contrast, Figure 9b illustrates that since the circuit manipulates both the upper and lower data bits in exactly the same way, the data paths can share the SRAM cells associated with each bit. This requires only four SRAM bits instead of eight, saving chip area.

Figure 10 illustrates another way to share programming bits. This figure corresponds to the same circuit as in Figure

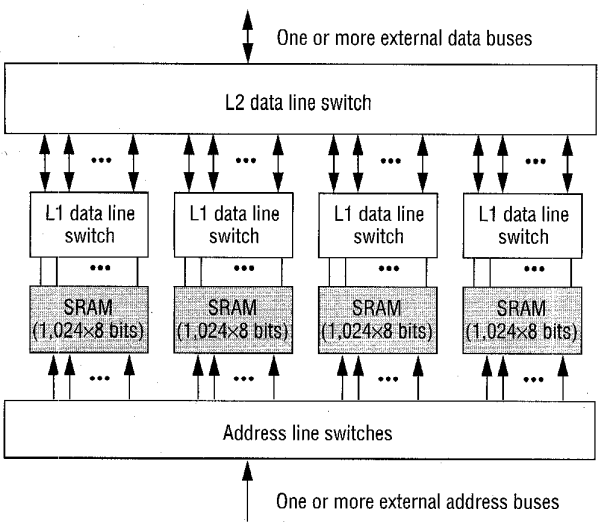

Figure 11. *Field-programmable memory architecture.*

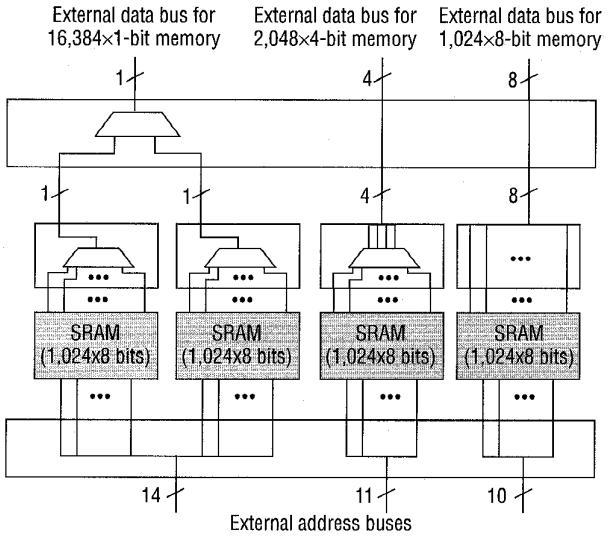

Figure 12. *Example configuration of field-programmable memory.*

9. It shows that since both LUTs for the two data bits have identical configurations, they can share their programming bits. (Figure 10 shows only the LUT programmed as a multiplexer, but the same would apply for the LUT programmed as an adder.) Note that I have depicted the 3-input LUT in the figure as a set of eight SRAM cells that pass through an 8-to-1 multiplexer to produce the LUT's output. This is how LUTs are usually constructed in FPGAs. The only change required in the logic-cell design is that we must build the LUT as a dual-ported memory, since it must serve two separate sets of inputs (address lines to the memory).

For implementation of data path circuits, the study's results showed a decrease in required chip area by about a factor of two for data path FPGAs as compared to traditional architectures.[9] Of course, this result depends on the data path properties of the circuits implemented, and area savings may vary considerably in practice.

FPGA memory structures. In the previous section, we described how to design an FPGA to take advantage of features in a specific class of circuit. Wilton, Rose, and Vranesic tackled a similar situation,[10] showing how to build a circuit block optimized for a specific class of applications. That circuit block is only part of the FPGA, not the whole chip. The motivation for this idea is twofold: Circuits implemented in FPGAs often require blocks of read/write memory, and the size of the required memories changes from circuit to circuit. The article shows how to realize a field-programmable memory (FPM) configurable both in the depth and width of the memory.

An example FPM appears in Figure 11. A set of address lines, which we can think of as one or more address buses of variable size, enter the FPM. The address lines enter a switching block (labeled address line switches) that directs the address bits to one or more of the FPM's SRAM blocks. In the figure, the FPM has four 1,024×8-bit memory blocks, but in general an FPM might have any number of SRAM blocks of any aspect ratio. The data lines from each SRAM block connect to a switch block labeled L1 data line switch, which multiplexes the data line bits and produces various aspect ratios. For example, we could multiplex the lower and upper nibbles in one of the 1,024×8-bit SRAM blocks to yield a 4-bit-wide memory with 2,048 entries (2,048×4 bits).

Each L1 data line switch connects to an L2 data line switch, which allows combining (multiplexing) data from multiple blocks to form larger memories. For instance, multiplexing two 1,024×8-bit SRAMs would yield a 2,048×8-bit block. Finally, the output (actually I/Os) of the FPM is a set of data lines connected to the L2 data line switch. We can think of these data bits as one or more external, variable-sized data buses.

An example of Figure 11's FPM configuration appears in Figure 12. The two 1,024×8-bit SRAM blocks on the left are each configured by their L1 data line switch to implement 8,192×1-bit memories. These memories are further multiplexed in the L2 data line switch to realize a 16,384×1-bit memory. The configuration of the next SRAM block to the right uses only the L1 data line switch to implement a 2,048×4-bit memory. Finally, the SRAM block on the far right remains unaltered by the data line switches and provides a 1,024×8-bit SRAM block.

To evaluate chip area and speed performance, the researchers compared the FPM architecture with the alternative of implementing memory blocks using LUTs.[10] They used the Xilinx XC4000 series, which allows configuration of its logic

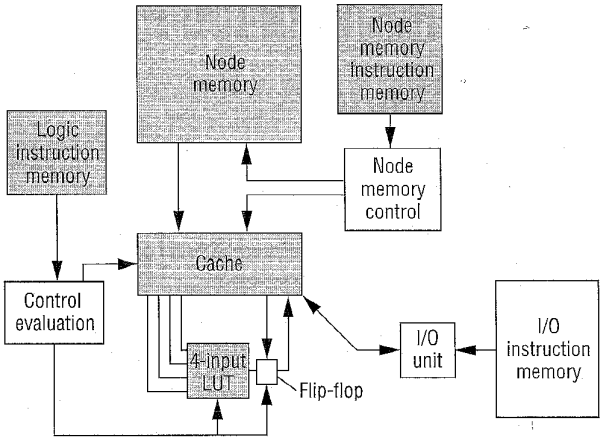

Figure 13. Architecture of virtual element gate array (VEGA).

blocks into variable-sized read/write memory blocks. The results show between 16 and 23 times higher memory density for the FPM compared to the XC4000 series, and two to three times better speed performance (memory access time).

FPGA for logic emulation. A study by Jones and Lewis[11] described an application-specific FPGA architecture designed for logic emulation. Researchers have traditionally used either slow software approaches or very expensive hardware accelerators for logic emulation. An obvious approach to emulating a logic circuit is simply to implement it by programming the circuit into an FPGA. This should be functionally correct, but will have different timing than the real circuit if built using a different technology. In fact, logic emulators are available for this purpose, but to attain useful logic capacity, they consist of many FPGAs connected together and cost about $100,000.

The key idea of Jones and Lewis is that FPGAs do not have sufficient capacity to emulate large circuits because arrays of LUTs with programmable interconnects inherently result in a low number of gates per unit area. To improve upon this logic density, the study showed how to multiplex a single LUT over time to serve as the equivalent of an entire array of LUTs. The approach required extra hardware—a set of SRAM memories, which themselves have very high density—to hold the different results generated over time by the multiplexed LUT.

Figure 13 depicts the architecture of the time-multiplexed FPGA or virtual element gate array (VEGA). This architecture emulates a circuit by representing each node in the circuit (that is, each output of a LUT) as a location in the SRAM block called the node memory. The single 4-input LUT computes the value over time as the circuit operates for each node. It executes a "program" stored in the SRAM block called the logic instruction memory.

A simple analogy for this concept is that the 4-input LUT is like a CPU, and the logic instruction memory stores the program that the CPU executes. Each "instruction" executed by the LUT specifies several parameters: a truth table function for the LUT, the addresses of four nodes to read as inputs to the 4-input LUT, and an address in which to store the generated output. Also, the 4-input LUT can optionally feed a flip-flop; each instruction includes configuration information for the flip-flop (such as the clock node).

Figure 13 also shows additional blocks such as a cache and a third memory block. These units serve to improve the efficiency of the architecture, much as a fast cache memory does in a computer system. The VEGA architecture also includes an I/O unit that allows us to build larger emulators by interconnecting multiple VEGAs.

Jones and Lewis evaluate VEGA efficiency by comparing its logic density to that in a typical commercial FPGA series, the Xilinx XC4000. VEGA achieved about 25 times greater logic density for the implemented circuit. The price paid for this improvement is lower speed performance. With a maximum operating frequency of about 50 kHz, VEGA is about 20 times slower than an XC4000-based emulator. However, this may be acceptable because VEGA offers a much cheaper solution in terms of gates per unit area, and the speed performance achieved is much greater than that of software simulations, the only other inexpensive solution.

THE AIM OF THIS ARTICLE has been to provide insight into FPGA architectural design. FPGA technology will remain an exciting and dynamic technology for at least the next several years as industry and academia continue to develop increasingly sophisticated devices through innovative research studies. D&T

References

1. S. Brown and J. Rose, "FPGA and CPLD Architectures: A Tutorial," *IEEE Design & Test of Computers*, Vol. 13, No. 2, pp. 42-57.
2. J.S. Rose et al., "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE J. Solid-State Circuits*, Vol. 25, No. 5, Oct. 1990, pp. 1217-1225.
3. S. Singh et al., "The Effect of Logic Block Architecture on FPGA Performance," *IEEE J. Solid-State Circuits*, Vol. 27, No. 3, Mar. 1992, pp. 281-287.
4. J.S. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *IEEE J. Solid-State Circuits*, Vol. 26, No. 3, Mar. 1991, pp. 277-282.
5. H. Hsieh et al., "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays," *Proc. 1990 Custom Integrated Circuits Conf.*, IEEE, Piscataway, N.J., 1990,

pp. 31.2.1-31.2.7.
6. J. Frankle, "Iterative and Adaptive Slack Allocation for Performance-Driven Layout and FPGA Routing," *Proc. 29th ACM/IEEE Design Automation Conf.*, IEEE Computer Society Press, Los Alamitos, Calif., 1992, pp. 536-542.
7. K. Chung et al., "Using Hierarchical Logic Blocks to Improve the Speed of Field-Programmable Gate Arrays," *Proc. Int'l Workshop on Field Programmable Logic and Applications*, Oxford, UK, 1991.
8. A. Aggarwal and D. Lewis, "Routing Architectures for Hierarchical Field-Programmable Gate Arrays," *Proc. Custom Integrated Circuits Conf.*, IEEE, 1994, pp. 475-478.
9. D. Cherepacha and D. Lewis, "DP-FPGA: An FPGA Architecture Optimized for Datapaths," *VLSI Design*, Vol. 4, No. 4, 1996, pp. 329-343.
10. S. Wilton, J. Rose, and Z. Vranesic, "Architecture of Centralized Field-Configurable Memory," *Proc. Third ACM Int'l Symp. on Field-Programmable Gate Arrays*, Assoc. for Computing Machinery, New York, 1995, pp. 97-103.
11. D. Jones and D.M. Lewis, "A Time Multiplexed FPGA Architecture for Logic Emulation," *Proc. Third ACM Int'l Symp. on Field-Programmable Gate Arrays*, ACM, 1995.

Stephen Brown is an assistant professor of electrical and computer engineering at the University of Toronto. He holds a PhD in electrical engineering from that university; his dissertation (on architecture and CAD for FPGAs) won the Canadian NSERC's 1992 prize for the best doctoral thesis in Canada. In 1990, the International Conference on Computer-Aided Design awarded him and coauthor Jonathan Rose a Best Paper award. A coauthor of the book, *Field-Programmable Gate Arrays*, he has also won four awards for excellence in teaching electrical engineering, computer engineering, and computer science courses. Brown is the general and program chair for the Fourth Canadian Workshop on Field-Programmable Devices (FPD 96) and is on the Technical Program Committee for the Sixth International Workshop on Field-Programmable Logic (FPL 96). He is a member of the IEEE and the Computer Society.

Address questions or comments about this article to Stephen Brown, Department of Electrical and Computer Engineering, University of Toronto, 10 Kings College Road, Toronto, Ontario, Canada, M5S 3G4; brown@eecg.toronto.edu.