The Design of an SRAM-Based Field-Programmable Gate Array—Part I: Architecture

Paul Chow, *Member, IEEE,* Soon Ong Seo, Jonathan Rose, *Member, IEEE,*
Kevin Chung, Gerard Páez-Monzón, and Immanuel Rahardja

***Abstract*—Field-programmable gate arrays (FPGA's) are now widely used for the implementation of digital systems, and many commercial architectures are available. Although the literature and data books contain detailed descriptions of these architectures, there is very little information on how the high-level architecture was chosen, and no information on the circuit-level or physical design of the devices. This paper describes the high-level architectural design of a static-random-access memory programmable FPGA. A forthcoming Part II will address the circuit design issues through to the physical layout. The logic block and routing architecture of the FPGA was determined through experimentation with benchmark circuits and custom-built computer-aided design tools. The resulting logic block is an asymmetric tree of four-input lookup tables that are hard-wired together and a segmented routing architecture with a carefully chosen segment length distribution.**

***Index Terms*—Field-programmable gate arrays, FPGA architecture, SRAM programmable.**

I. Introduction

FIELD-PROGRAMMABLE gate array (FPGA) technology permits the design of many different complex digital circuits using a single off-the-shelf device [1]. The time-to-market pressures and low financial risk has made FPGA's and complex programmable logic devices (CPLD's) an increasingly popular vehicle for prototyping and, in many cases, actual production. There are many different architectures now available from over 15 silicon vendors. Although the databooks and related literature usually describe the architecture in detail, there is little information on *how* the architecture was chosen and no information on the circuit-level or physical-layout level of the design. The contribution of this paper is to describe how a suitable architecture was chosen for a static random-access memory (SRAM) programmable FPGA. A forthcoming paper [2] examines the circuit and layout issues encountered when implementing that FPGA. Our primary goal here is to produce a high-speed architecture and circuit with a reasonable logic density.

FPGA's were first introduced in 1986 by Xilinx Inc., San Jose, CA, using a memory-based programming technology [3]. Since then, there have been many new commercial architectures [4]–[8] and several new programming technologies, including two types of antifuse [9]–[11] and floating-gate transistors, which are ultraviolet (UV) and electrically erasable [12]–[14].

A few noncommercial FPGA architectures have been reported for which the design details are more readily available. The Triptych FPGA [15], [16] matches the physical structure of the routing architecture to the fanin/fanout nature of the structure of digital logic by using short connections to the nearest neighbors. Segmented routing channels are used between the columns to provide for nets with fanout greater than one. No discussion is given about how the segmentation length distribution was selected. This routing architecture does not allow the arbitrary point-to-point routing available in general FPGA structures. The logic block implements logical functions using a multiplexer-based three-input lookup table followed by a master-slave D-latch and can also be used for routing. Initial results show potential implementation efficiencies in terms of area using this structure. The Montage FPGA [17], [16] is a version of the Triptych architecture, which is modified to support asynchronous circuits and interfacing separately clocked synchronous circuits. This is achieved by the addition of an arbiter unit and a clocking scheme that allows two possible clocks or makes the latches transparent.

Earlier work at the University of Toronto, Toronto, Ont., Canada, resulted in the implementation of an architecture (UTFPGA1) using three cascaded four-input logic blocks and segmented routing [18]. UTFPGA1 used information from previous architectural studies, but there was very little transistor-level optimization (for speed), and little time was spent on layout optimization. This was a first attempt that provided some insight into the problems faced in the design and layout of an FPGA. An earlier version of the work presented here appeared in [19].

Modern trends in computer architecture have shown the importance of considering both the compiler technology and the hardware technology at the same time, when designing for high performance [20]. This is also true in the design of an FPGA, where it is important that the computer-aided design (CAD) tools collaborate with the architecture of the FPGA. At the University of Toronto, we have developed

Manuscript received May 31, 1996; revised September 8, 1998. This work was supported under a MICRONET Network of Excellence Grant.

P. Chow and J. Rose are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont., Canada M5S 3G4 (e-mail: pc@eecg.toronto.edu).

S. O. Seo is with ATI Technologies, Thornhill, Ont., Canada L3T 7N6.

K. Chung is with Xilinx Toronto Development Centre, Toronto, Ont., Canada M5S 2T9.

G. Páez-Monzón is with National Semiconductor Corporation/Cyrix-West, Santa Clara, CA 95052-8090 USA, on leave from CEMISID, Universidad de Los Andes–Venezuela, Mérida-Mérida 5101, Venezuela.

I. Rahardja is with Aristo Technology Inc., Cupertino, CA 95104 (e-mail: pc@eecg.toronto.edu).

Publisher Item Identifier S 1063-8210(99)03341-7.

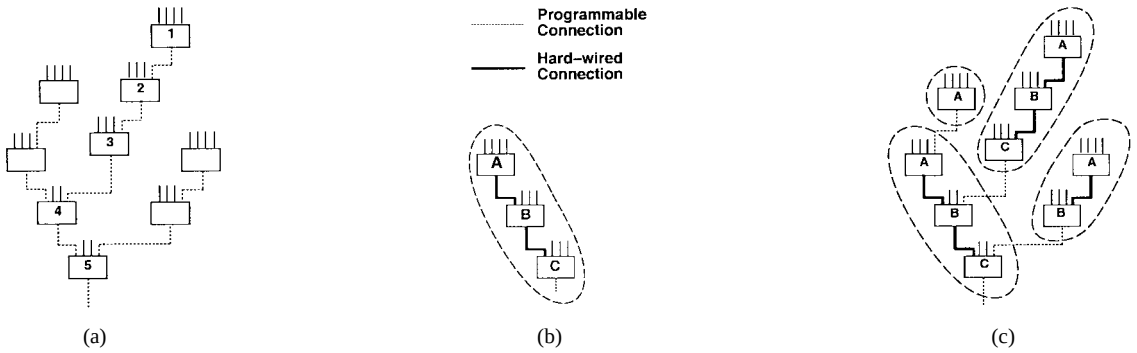


Fig. 1. Hard-wired connections and logic blocks. (a) Network of basic blocks. (b) Hard-wired connections and logic blocks. (c) Faster HLB network.

a number of custom CAD tools that have been used to perform *experimental* architectural studies. In this paper, we will describe two important architectural innovations that lead to higher speed FPGA's and influenced the architecture of our second-generation FPGA. In the forthcoming Part II [2], we will also introduce a novel layout style for FPGA's that builds the basic tile using a set of identical minitiles and customization by the addition of vias. This allows us to still achieve reasonable density while significantly reducing the time spent doing custom layout. These features have all been implemented in our test chip, called **L**ogic that's **E**rasable and **G**reatly **O**ptimized (LEGO).

In Section II, we show how various hard-wired interconnect topologies between lookup tables can form larger logic blocks and influence delay and how to select an appropriate topology. In Section III, the routing architecture is derived and we show that longer dedicated wire lengths in the routing architecture are beneficial. Section IV provides a summary of our results and some conclusions.

II. Logic Block Architecture

The speed and density gap between FPGA's and mask-programmable gate arrays (MPGA's) is mostly due to the routing structures used to connect logic components in each technology. In MPGA's, the logic elements are connected with mask-programmed metal wires. In FPGA's, logic block pins are connected using field-programmable switches. Regardless of the type of programmable switch used in the FPGA (e.g., SRAM controlled pass transistors [4] or antifuses [21], or floating-gate-based switching [14]), the capacitance, resistance, and size of programmable connections makes them much slower and larger than a simple metal wire.

One way to improve the speed and density of FPGA's is to replace some of the programmable connections between basic logic blocks with hard-wired connections, which are simple metal wires [22]. By using hard-wired links to construct more coarse-grained logic blocks [called hard-wired logic blocks (HLB's)] from several basic blocks, the delay and size of circuits can be reduced. For example, Fig. 1(a) shows a network of basic blocks with five programmable connections in the routing along the critical path (which we define as the combinational path with the largest number of logic levels) from block 1 to block 5 and nine programmable connections in total. Suppose that three of the basic blocks are hard-wired together to create an HLB with a fast three-block path, as shown in Fig. 1(b). The resulting circuit in Fig. 1(c) has only two slow programmable links instead of five along the critical path. This is a sizable reduction in routing delay. Also, the total number of programmable connections has been reduced from 9 to 4 and this may lead to a reduction in routing area.

A. The HLB Design Space

Given this general notion of HLB's, it is clear that there are many possible ways to implement it. Assuming that there is only one type of basic block that is connected in a given HLB, the choices lie in the topology of the interconnection between the blocks and size of the basic block itself.

Each choice will provide different delay and area for a given circuit. In general, the more hard-wired links in an HLB, the faster the circuits implemented using that logic block. However, a greater number of hard-wired connections provide less connection flexibility and may lead to lower density because basic blocks are wasted. In this section, we give a brief description of the experiments performed to determine a good choice for the HLB topology. We omit the work done to determine the choice of the basic block, which the interested reader can find in [23], along with a more detailed description of these experiments.

The basic block we choose to work with is the four-input lookup table (4-LUT), as previous studies have indicated that this is a reasonable choice for both speed and density [24]–[28] and as such has been adopted by several commercial vendors [4]–[6].

We restricted the choice of topology to be a tree, as circuits often are treelike in nature. Fig. 2 illustrates several of the topologies that were investigated. The naming convention of these structures is as follows: it begins with the letter "L," then the height of the HLB (in basic blocks), then a dash ("–") followed by a listing of sizes of the subtrees from a preorder traversal of the canonical HLB tree. Each subtree size is separated by a dot (".") with the restriction that leaf inputs and single-LUT subtrees are not listed.

In addition to the HLB topologies illustrated in Fig. 2, all possible tree topologies with nine or fewer 4-LUT's and three or fewer LUT levels were explored. Initial experiments indicated that although HLB's with more than nine 4-LUT's provide greater speedups, they require too much area to be considered practical.

TABLE I
SPEED AND AREA OF HLB'S-BASED FPGA'S, SPEED-OPTIMIZED CIRCUITS

Number of LUTs in HLB	HLB Topology	Speed-Optimized Circuits		Area-Optimized Circuits	
		Speed Factor	Area Factor	Speed Factor	Area Factor
1	L1	1 (35 MHz)	1 ($15.3 \times 106\ \mu m^2$)	1 (22 MHz)	1 ($6.7 \times 106\ \mu m^2$)
2	L2-2	1.14	1.19	1.17	0.93
3	L2-3	1.26	1.21	1.17	0.96
4	L2-4	1.34	1.32	1.09	1.00
4	L3-4.2	1.28	1.31	1.27	0.93
5	L2-5	1.42	1.41	1.03	1.10
5	L3-5.2	1.37	1.37	1.24	1.04
6	L3-6.5	1.43	1.65	1.04	1.23
7	L3-7.3	1.43	1.66	1.19	1.28
8	L3-8.3.2	1.47	1.61	1.25	1.32
9	L3-9.4.2	1.50	1.70	1.24	1.42

An important architectural assumption is that each HLB has a buffer on the output of each LUT basic block that is accessible to the programmable routing. This direct access has two important advantages: delay is reduced because an output can be accessed without propagating it through downstream logic blocks and density is increased because unrelated pieces of logic can be packed together in a multioutput HLB.

The goal of the experiments described below is to select from among the many possible hard-wired 4-LUT architectures one that produces high system speeds at a reasonable cost in area.

B. Experimental Method

To evaluate the HLB topologies, a set of 15 benchmark combinational circuits from the Microelectronics Centre of North Carolina (MCNC) suite[1] were each "implemented" as a set of FPGA's with each HLB topology. The area and delay of each implemented circuit is then calculated using area and delay models, and the results are averaged over all circuits for each HLB topology. By "implementation," we mean that each circuit is processed through a suite of CAD tools that take it from the logic level (equations) through to the physical placement upon the hypothetical FPGA and the global routing on that FPGA. Note that a custom logic synthesis tool, called TEMPT, was developed to allow synthesis into the hard-wired structures [29].

The focus of this paper is not the details of these experiments, but the results. As such, the description of the area and delay models and the CAD tools needed to provide the implementation are omitted, but can be found in [23]. The only detail necessary for the discussion below is the fact that the CAD implementation stream has two modes: one in which the resulting FPGA circuit is optimized for speed (sometimes at great cost in area) and the other in which the circuit is optimized for area at the cost of some speed. Our method of selecting an appropriate logic block for LEGO is to inspect the results of both experiments using both optimizations and to deduce a reasonable compromise.

[1] MCNC, "Logic synthesis and optimization benchmarks user guide, Version 3.0," Jan. 1991.

C. Experimental Results

Table I provides the summarized results of the implementation of the benchmark circuits for those HLB topologies that appear the most promising. Table I gives the highest speed (lowest critical path delay) topologies for each *size* of HLB, where size is measured by the number of 4-LUT's in the HLB, when the circuits are optimized for both speed and area.

The first column of Table I lists the number of 4-LUT's in the HLB. If there are two rows for a given number of 4-LUT's, the second entry is an alternative HLB that shows a slower speed, but at a lower area cost. The second column gives the label of the HLB topology, as described above. The third and fifth columns give the normalized speedup of the HLB with respect to L1 (a 4-LUT FPGA with no hard-wired links), for the speed and area-optimized implementations. The speed is calculated as the geometric average of the inverse of the critical path delay for each circuit. It is then normalized with respect to L1. The fourth and sixth columns of Table I give the normalized area, with respect to L1, for both the speed and area cases. The unnormalized absolute values for speed and area are given in the L1 row, assuming a 1.2-μm complimentary metal–oxide–semiconductor (CMOS) process. Absolute areas come from actual layouts of cells used in an area model. Absolute times come from a delay model, which incorporates delay data derived from the simulation of the cell layouts.

The speed-optimized results show that the hard-wired links can increase system speed from 14% up to 50% compared to a flat nonhard-wired 4-LUT when the circuits are speed optimized. However, the area cost of this speed is an increase of 19%–70% in circuit size. Some of this increase is caused by wasteful logic synthesis tools that sacrifice area for speed, but it also comes from the fundamental speed-area tradeoff with hard-wired links, as discussed above.

The area-optimized results show that several topologies are actually *more* area efficient than the flat four-input lookup table. All topologies provide a gain in speed, although less than that of the speed optimized case.

Our criteria in choosing a block is to select one that gives a good combination of speed and good density. From Table I, we reject those blocks that have too high an area penalty: those in the last four rows. Of the remaining blocks, L3-5.2, L2-5,

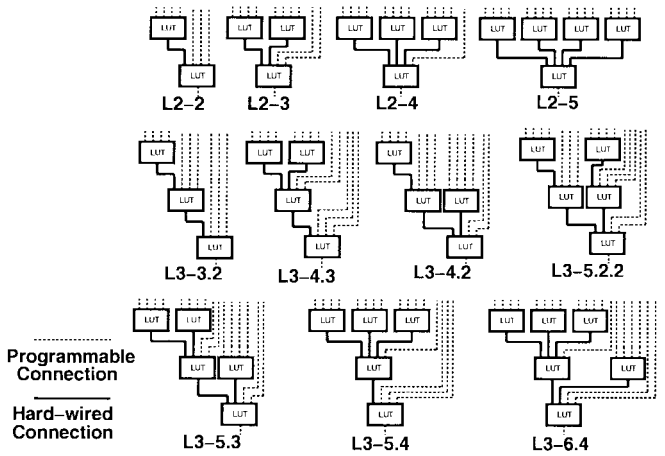


Fig. 2. Example of some of the tree topologies investigated.

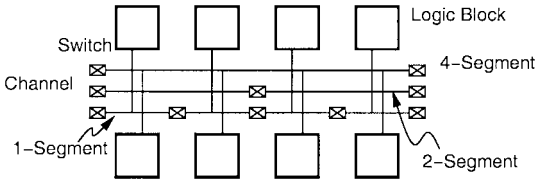


Fig. 3. Segmented routing architecture.

L3-4.2, and L2-4 provide the best speed up. However, the L3-4.2 block proved superior to the others in the area-optimized case, and so it was selected. Of note is that this block provided the best logic density overall, while also providing significant speed up. The L3-4.2 HLB is illustrated in Fig. 2.

It is tempting to use programmable multiplexers in place of the hard-wired connections to allow either the direct link or to access an input. We do not consider their use in this study for two reasons: first, we are interested in finding the fastest topologies and the multiplexers do not improve the speed, secondly, each extra input to the logic block is expensive in terms of area and part of the purpose of the hard-wired connection is to save that area.

III. Routing Architecture

The FPGA routing architecture is the most important determining factor of system speed and logic density because the programmable switches, which are pass transistors driven by SRAM cells, have significant resistance and capacitance and require large area. The routing architecture is the manner in which wire segments and switches are organized.

In the original Xilinx 2000 and 3000 architectures [3], [30] a very simple architecture was employed: most wire segments spanned only the length (or width) of one logic block and had switches at each end. One way to improve the speed of connections that travel long distances is to provide segments that span multiple logic blocks without being switched, as illustrated in Fig. 3. In this way, a segment with appropriate length for a connection can be selected, which results in less resistance along the path. The general notion of segmented routing was first introduced in [9].

A key architectural question is the determination of the number of segments of each length, called the *segmentation length distribution.* The greater number of longer segments, the greater the likelihood that long and otherwise time-consuming connections will be routed with fewer series switches, resulting in a faster circuit. An excessive number of longer segments, however, will mean that some of the long segments will have to be used for short connections, resulting in the waste of part of the segment. This waste leads to a decrease in logic density. Thus, it is important to find a segmentation distribution that improves speed, but does not waste too much area.

There is a second important architectural feature of segmented routing architectures besides the distribution: a segment is called internally *populated* if it is possible to make connections from the middle of a segment to logic blocks or other routing segments. The advantage of unpopulated segments is that it has less parasitic switch capacitance connected to the segment, which makes it faster. The disadvantage is that the reduction in routing flexibility (without population there cannot be internal fanout) may result in the need for more tracks and, thus, a loss of logic density.

In this section, we briefly summarize a study on segmented routing architecture distribution and population that was used as the basis for decisions on the routing architecture for LEGO. The basic approach was experimental, similar to the one described in the previous section: several benchmark circuits are "implemented" as FPGA's, each with a different segmentation distribution. These experiments required several CAD tools, including a detailed router specifically designed for FPGA's [31]. The number of tracks needed to complete the route using segments of only length one is called the minimum. When the channel includes segments of other lengths, the number of tracks required above the minimum is measured.

A. Definitions and Experimental Method

Fig. 4 illustrates the broad architectural features of the LEGO FPGA. The *logic block* (L) has pins that are connected to routing channels in a structure called a *connection block* (C). The *switch block* (S) provides connectivity among its attached channels. A channel consists of C and S blocks. Each channel contains W routing tracks. We assume that each track consists of only one type of segment length. We will consider only segments of length one, two, and three, which will be referred to as 1-, 2-, and 3-segments.

Let D_1 be the fraction of the W tracks that contain 1-segments. Similarly, D_2 and D_3 are the fractions for 2- and 3-segments so that $D_1 + D_2 + D_3 = 1$.

Concerning the issue of the internal population of the segments, we divide it into two parts: whether or not the connection blocks internal to the segment (i.e., those not at its ends) should be populated and whether or not the internal switch blocks should be populated. The following notation indicates the level of population: CB_u indicates that all unpopulatable connection blocks are unpopulated, while CB_p means that they are populated. Similarly, for switch block population, we use SB_u and SB_p. Thus, there are four combinations of population. Notice that for 2-segments,

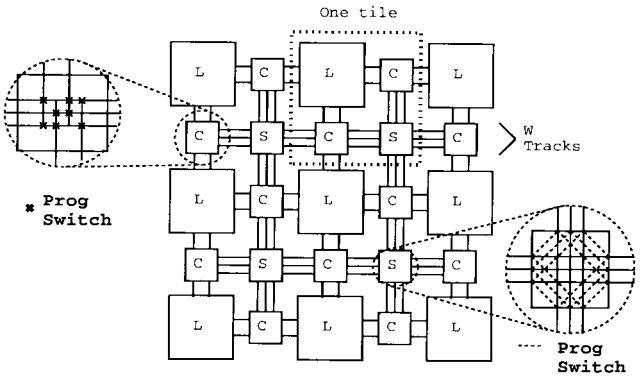


Fig. 4. Architectural definitions.

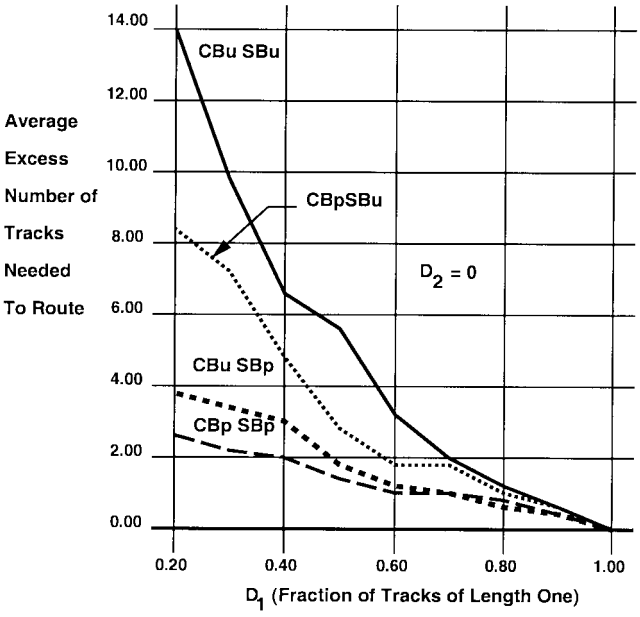


Fig. 5. Population comparison.

only the middle switch block can be depopulated and, for 3-segments, there are two switch blocks and one connection block that can be depopulated.

B. Experimental Results and Decisions

The following experiments were performed: for all possible values of D_1, D_2, and D_3, using all four population combinations, the number of excess tracks needed over the minimum (for $D_1 = 1$) were measured and averaged for five benchmark circuits. We addressed the questions of population and distribution using these experiments. Note that the logic block used in these experiments was a single 4-input LUT, with a D flip-flop. While this was not the block chosen for LEGO, we believe the general results discussed below will hold for most logic blocks.

1) Population of Segments: The first question we address is that of population of the S and C blocks. Fig. 5 is a plot, for $D_2 = 0$, of the average number of excess tracks versus D_1 for the five circuits. The four curves are the different

TABLE II
AVERAGE EXCESS TRACKS FOR ALL DISTRIBUTIONS, POPULATION CB_uSB_p. ABSOLUTE AVERAGE NUMBER OF TRACKS = 12, STANDARD DEVIATION = 2.2

D_2 \ D_3	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
0.0	0.0	0.4	0.6	1.0	1.2	1.8	3.0	3.4	3.8
0.1	0.2	0.6	1.0	1.2	1.6	2.2	2.8	3.4	
0.2	0.6	0.8	1.0	1.0	1.8	3.0	3.2		
0.3	0.6	0.6	1.2	1.8	2.2	2.4			
0.4	0.6	1.0	1.0	2.0	2.6				
0.5	0.8	1.2	1.4	2.0					
0.6	1.2	1.6	2.0						
0.7	1.4	1.8							
0.8	1.8								

combinations of population for the switch and connection block. Notice that as D_1 decreases from one, D_3 increases from zero ($D_3 = 1 - D_1$). Since this provides a greater number of longer segments with lower flexibility, we expect the number of tracks needed to successfully route will increase, and this is borne out in the figure.

The data show that both cases in which the switch block is unpopulated (the upper two curves) result in significantly more excess tracks. For this reason we decided to populate the switch block in LEGO. On the other hand, the bottom two curves (with the switch block populated) illustrate that only a minor increase in tracks is experienced when the connection block is depopulated. Since there is an advantage in speed for depopulation (less capacitance on the track), we decided to depopulate the connection block in LEGO.

2) Segmentation Distribution: Table II gives the average number of excess tracks for many possible values of D_1, D_2, and D_3. Note that, since $D_1 + D_2 + D_3 = 1$, there are only two independent variables, which we chose as D_2 and D_3 in the table. The rows vary D_2 and the columns vary D_3. This table is for architectures with both the C blocks unpopulated and the S blocks populated, as discussed above.

Table II illustrates that, even with 80% length (three segments) that only about four extra tracks on average are required. (The minimum number of absolute tracks required for the five circuits ranged from 9 to 15.) We decided that we were willing to tolerate only about one extra track per channel above the minimum so that the area cost of the higher speed tracks would be small. Thus, architectures with D_2 in the range of 0.0–0.6 and D_3 in the range of 0.1–0.4 (those shown in boxes in Table II) would reflect such a choice. The faster architectures are those with higher values of D_3.

In Part II [2], we give the reasons that lead to the choice of 16 tracks per channel in LEGO. For the segmentation distribution, there is an additional constraint that arises from the one-tile style of layout that was used: there must be an even number of 2-segment tracks and the number of 3-segment tracks must be a multiple of three. Taking this constraint and the above distribution into account, the following segmentation distribution was used in LEGO: nine of the 16 tracks are 1-segments, four tracks are 2-segments, and three are 3-segments. This corresponds to a segmentation distribution in which $D_1 = 0.56$, $D_2 = 0.25$, and $D_3 = 0.19$, which fits within the architectural range indicated above.

IV. Conclusions

We have described the high-level architectural decisions used to select the logic block and routing architecture for the design of a high-performance FPGA. We developed the architecture using an experimental process in which custom-built CAD tools are used to implement benchmark circuits on candidate architectures. This method has the danger that the tools may have unfair algorithmic biases toward certain architectures, resulting in "incorrect" architectural decisions. We have made an honest effort, however, to use the highest quality algorithms possible. This method is the only practical way of dealing with the enormous complexity of FPGA architecture development because theoretical analysis can never account for all of the conflicting constraints that must be considered in a real design. In addition, this kind of approach produces an architecture that is tuned to the capabilities of the tools, which is key to the success of an FPGA.

The architecture will be a symmetric array of the L3-4.2 HLB logic blocks with a segmented routing architecture employing the distribution and population described in Section III. Fig. 4 illustrates such an array. An FPGA with these characteristics was designed and fabricated. The circuit and layout issues are described in the forthcoming Part II [2].

As technology continues to scale, the area of the logic will decrease, but the relative effect of the delays due to wiring will increase. This will create further demands on the routing architecture, especially as the amount of logic available on a single FPGA will also increase, resulting in larger systems being built. Architecturally, there will continue to be a need for innovation in new logic and routing structures, and any new studies must always match the capabilities of the design and implementation tools to the architecture.

It is particularly important that a good set of benchmark circuits be available, especially to the academic community. With the trend toward larger systems-on-a-chip, studies like ours can only have merit when they use circuits that reflect this trend. We hope that the community will work toward improving the publicly available benchmark suites such as the one provided by MCNC.[1] It is essential that the industrial community participate in this effort as they possess such circuits and stand to gain the most from the research that uses them.

In the future, we will explore other aspects in architecture and design, including the incorporation of large blocks of memory, more complex routing structures for high-speed interconnect, and the global distribution of routing tracks.

Acknowledgment

The authors acknowledge the valuable feedback provided by this paper's referees.

References

[1] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field-Programmable Gate Arrays.* Norwell, MA: Kluwer, 1992.

[2] P. Chow, S. O. Seo, J. Rose, K. Chung, G. Páez-Monzón, and I. Rahardja, "The design of an SRAM-based field-programmable gate array—Part II: Circuit design and layout," *IEEE Trans. VLSI Syst.,* to be published.

[3] W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sze, "A user programmable reconfigurable gate array," in *Proc. Custom Integrated Circuits Conf.,* May 1986, pp. 233–235.

[4] H. Hsieh, W. Carter, J. Y. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, and L. Tinkey, "Third-generation architecture boosts speed and density of field-programmable gate arrays," in *Proc. Custom Integrated Circuits Conf.,* 1990, pp. 31.2.1–31.2.7.

[5] B. K. Britton, D. D. Hill, W. Oswald, N.-S. Woo, and S. Singh, "Optimized reconfigurable cell array architecture for high-performance field-programmable gate arrays," in *Proc. Custom Integrated Circuits Conf.,* 1993, pp. 7.2.1–7.2.5.

[6] R. Cliff *et al.,* "A dual granularity and globally interconnected architecture for a programmable logic device," in *Proc. Custom Integrated Circuits Conf.,* 1993, pp. 7.3.1–7.3.5.

[7] D. E. Smith, "Intel's FLEXlogic FPGA architecture," in *Compcon Spring'93,* pp. 378–384.

[8] D. Tavana, W. Yee, S. Young, and B. Fawcett, "Logic block and routing considerations for a new SRAM-based FPGA architecture," in *Proc. Custom Integrated Circuits Conf.,* 1995, pp. 511–514.

[9] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. A. El-ayat, and A. Mohsen, "An architecture for electrically configurable gate arrays," *IEEE J. Solid-State Circuits,* vol. 24, pp. 394–398, Apr. 1989.

[10] J. Birkner *et al.,* "A very-high-speed field-programmable gate array using metal-to-metal antifuse programmable elements," *Microelectron. J.,* vol. 23, pp. 561–568, 1992.

[11] D. Marple and L. Cooke, "An MPGA compatible FPGA architecture," in *Proc. Custom Integrated Circuits Conf.,* 1992, pp. 4.2.1–4.2.4.

[12] S. C. Wong, H. C. So, J. H. Ou, and J. Costello, "A 5000-gate CMOS EPLD with multiple logic and interconnect arrays," in *Proc. Custom Integrated Circuits Conf.,* 1989, pp. 5.8.1–5.8.4.

[13] A. Gupta, V. Aggarwal, R. Patel, P. Chalasani, D. Chu, P. Seeni, P. Liu, J. Wu, and G. Kaat, "A user configurable gate array using CMOS-EPROM technology," in *Proc. Custom Integrated Circuits Conf.,* 1990, pp. 31.7.1–31.7.4.

[14] R. Patel *et al.,* "A 90.7 MHz-2.5 million transistors CMOS PLD with JTAG boundary scan and in-system programmability," in *Proc. Custom Integrated Circuits Conf.,* 1995, pp. 507–510.

[15] C. Ebeling, G. Borriello, S. A. Hauck, D. Song, and E. A. Walkup, "TRIPTYCH: A new FPGA architecture," in *FPGA's,* W. Moore and W. Luk, Eds.. Abingdon, U.K.: Abingdon, 1991, ch. 3.1, pp. 75–90.

[16] G. Borriello, C. Ebeling, S. A. Hauck, and S. Burns, "The Triptych FPGA architecture," *IEEE Trans. VLSI Syst.,* vol. 3, pp. 491–500, Dec. 1995.

[17] S. Hauck, G. Borriello, S. Burns, and C. Ebeling, "MONTAGE: An FPGA for synchronous and asynchronous circuits," in *Proc. 2nd Int. Workshop Field-Programmable Logic Applicat.,* Vienna, Austria, Sept. 1992.

[18] P. Chow, S. O. Seo, D. Au, T. Choy, B. Fallah, D. Lewis, C. Li, and J. Rose, "A 1.2 μm CMOS FPGA using cascaded logic blocks and segmented routing," in *FPGA's,* W. Moore and W. Luk, Eds. Abingdon, U.K.: Abingdon, 1991, ch. 3.2, pp. 91–102.

[19] P. Chow, S. O. Seo, K. Chung, G. Paez, and J. Rose, "A high-speed FPGA using programmable mini-tiles," in *Symp. Integrated Syst.,* Mar. 1993, pp. 103–122.

[20] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach,* 2nd ed. New York: Morgan Kaufmann, 1995.

[21] M. Ahrens, A. El Gamal, D. Galbraith, J. Greene, and S. Kaptanoglu, *et al.,* "An FPGA family optimized for high densities and reduced routing delay," in *Proc. Custom Integrated Circuits Conf.,* 1990, pp. 31.5.1–31.5.4.

[22] K. Chung, S. Singh, J. Rose, and P. Chow, "Using hierarchical logic blocks to improve the speed of field-programmable gate arrays," in *FPGA's,* W. M. and W. Luk, Eds. Abingdon, U.K.: Abingdon, 1991, ch. 3.3, pp. 103–113.

[23] K. Chung, "Architecture and synthesis of field-programmable gate arrays with hard-wired connections," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, Ont., Canada, 1994.

[24] J. Rose, R. J. Francis, P. Chow, and D. Lewis, "The effect of logic block complexity on area of programmable gate arrays," in *Custom Integrated Circuits Conf.,* May 1989, pp. 5.3.1–5.3.5.

[25] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: The effect of logic block functionality on area efficiency," *IEEE J. Solid-State Circuits,* vol. 25, pp. 1217–1225, Oct. 1990.

[26] S. Singh, "The effect of logic block architecture on the speed of field-programmable gate arrays," M.A.Sc. thesis, Dept. Elect. Eng., Univ. Toronto, Toronto, Ont., Canada, 1991.

[27] S. Singh, J. Rose, P. Chow, and D. Lewis, "The effect of logic block architecture on FPGA performance," *IEEE J. Solid-State Circuits,* vol.

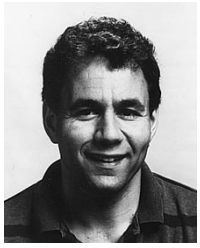
27, pp. 281–287, Mar. 1992.
[28] J. L. Kouloheris and A. El Gamal, "PLA-based FPGA area versus cell granularity," in *Proc. Custom Integrated Circuits Conf.*, 1992, pp. 4.3.1–4.3.4.
[29] K. Chung and J. Rose, "TEMPT: Technology mapping for the exploration of FPGA architectures with hard-wired connections," in *Proc. 29th Design Automation Conf.*, 1992, pp. 361–367.
[30] H.-C. Hsieh, K. Duong, J. Y. Ja, R. Kanazawa, L. T. Ngo, L. G. Tinkey, W. S. Carter, and R. H. Freeman, "A second generation user-programmable gate array," in *Custom Integrated Circuits Conf.*, 1987, pp. 515–521.
[31] S. Brown, J. S. Rose, and Z. Vranesic, "A detailed router for field programmable gate arrays," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 620–628, May 1992.

Paul Chow (S'79–M'83) received the B.A.Sc. degree with honors in engineering science, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, Ont., Canada, in 1977, 1979, and 1984, respectively.

In 1984, he joined the Computer Systems Laboratory, Stanford University, Stanford, CA, as a Research Associate, where he was a major contributor to the MIPS-X project. In January 1988, he joined the Department of Electrical and Computer Engineering, University of Toronto, where he is currently an Associate Professor. He spent the 1995–1996 year at ATI Technologies Inc., Thornhill, Ont., Canada, where he was involved with integrated circuit (IC) design and new CAD flows. His research interests include high-performance computer architectures, architectures for programmable digital-signal processors, VLSI systems design, and FPGA architectures and applications. He is currently the chair of the Technical Advisory Committee for the Canadian Microelectronics Corporation.

Soon Ong Seo received the B.A.Sc. degree with honors in electrical engineering from the University of Ottawa, Ottawa, Ont., Canada, in 1985, and the M.A.Sc. degree from the University of Toronto, Toronto, Ont., Canada, in 1994.

In 1992, he joined ATI Technologies Inc., Thornhill, Ont., Canada, as a Design Engineer, where he has been involved in the design and support of numerous graphics chips. His interests include high-performance architecture and design, graphics, and FPGA design and architecture.



Jonathan Rose (S'79–M'80) received the Ph.D. degree in electrical engineering from the University of Toronto, Toronto, Ont., Canada, in 1986.

He is currently a Professor of electrical and computer engineering at the University of Toronto, and an NSERC University Research Fellow. From 1986 to 1989, he was a Research Associate in the Computer Systems Laboratory, Stanford University. In 1989, he joined the faculty of the University of Toronto. He spent the 1995–1996 year as a Senior Research Scientist at Xilinx Inc., San Jose, CA, where he worked on a next-generation FPGA architecture. He has worked for Bell-Northern Research and a number of FPGA companies on a consulting basis. His research covers all aspects of FPGA's, including architecture, CAD, field-programmable systems, and graphics and vision applications of rapid prototyping systems. He is the co-founder of the ACM FPGA Symposium, and remains part of that Symposium on its Steering and Program Committees.

Kevin Chung received the B.A.Sc., M.A.Sc., and Ph.D. degrees from the University of Toronto, Toronto, Ont., Canada, in 1986, 1988, and 1994, respectively.

From 1993 to 1994, he was a Software Engineer at Data I/O Corporation. Since 1994, he has been with Xilinx Inc, Toronto, Ont., Canada, where he is currently a Senior Software Engineer in the Xilinx Toronto Development Centre. His main interests are in FPGA architectures and CAD algorithms for FPGA's.

Gerard Páez-Monzón received the B.E.E. degree from Villanova University, Villanova, PA, in 1979, and the D.E.A. degree in computer systems and Ph.D. degree in computer science from the Université Pierre et Marie Curie, Paris, France, in 1983 and 1986, respectively.

He is currently a Professor in the Engineering School, Universidad de Los Andes–Venezuela, Los Andes, Venezuela. He is currently on leave at the National Semiconductor Corporation/Cyrix-West, Santa Clara, CA, as a Microprocessor Architect, where he is involved with the Architecture Group in $\times 86$ architecture designs. He was a Visiting Scholar at the University of Toronto, Toronto, Ont., Canada, from 1991 to 1992, where he worked with the FPGA Group. He is a Principal Member of the Research Center CEMISID, Universidad de Los Andes, in the areas of microprocessor architecture, microprogramming, floating-point units design, FPGA, and VLSI design. He has authored a computer architecture book and a number of computer science research papers.

Immanuel Rahardja received the bachelor degree in electrical engineering from the University of Toronto, Toronto, Ont., Canada, in 1992.

From 1992 to 1996, he was with Xilinx Inc., San Jose, CA, where he was involved with various projects exploring and evaluating different FPGA architectures. He is currently with Aristo Technology Inc., Cupertino, CA, where he is developing the graphical user interface software for a set of block-level system IC planning and implementation design tools.