The Design of a SRAM-Based Field-Programmable Gate Array—Part II: Circuit Design and Layout

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. In Part I of this paper, we described the high-level architectural design of a static random-access memory programmable FPGA. This paper will address the circuit-design issues through to the physical layout. We address area-speed tradeoffs in the design of the logic block circuits and in the connections between the logic and the routing structure. All commercial FPGA designs are done using full-custom hand layout to obtain absolute minimum die sizes. This is both labor and time intensive. We propose a design style with a *minitile* that contains a portion of all the components in the logic tile, resulting in less full-custom effort. The minitile is replicated in a 4 × 4 array to create a macro tile. The minitile is optimized for layout density and speed, and is customized in the array by adding appropriate vias. This technique also permits easy changing of the hard-wired connections in the logic block architecture and the segmentation length distribution in the routing architecture.**

***Index Terms*—FPGA, FPGA architecture, FPGA circuit design, field-programmable gate arrays, SRAM programmable.**

I. Introduction

THE design and implementation of field-programmable gate array (FPGA) technology is rarely described in the literature because much of the information is proprietary. In [1], we describe some of the prior work and show the high-level architectural decisions used to select the logic block and routing architecture for the design of a high-performance FPGA. A symmetric array of hard-wired four-input logic blocks with a segmented routing architecture was selected. Fig. 1 illustrates such an array.

Manuscript received May 31, 1996; revised December 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.

Publisher Item Identifier S 1063-8210(99)04561-8.

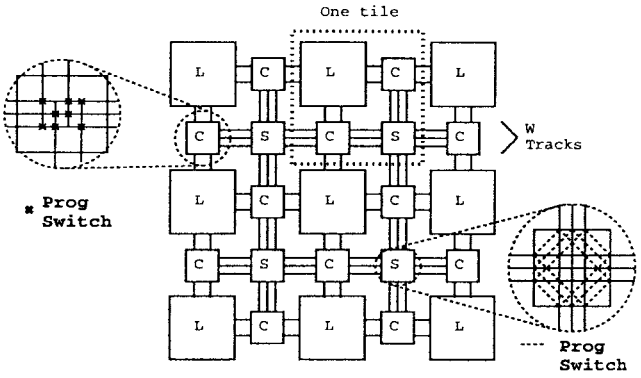


Fig. 1. Architectural definitions.

This paper reports on the circuit design and layout considerations that were made during the implementation of our prototype chip. In Section II, we discuss the various circuit-design issues that must be considered. A novel layout style for FPGA's is presented in Section III. The area, speed, and performance of the chip are given in Section IV. Finally, Section V provides some conclusions.

II. Circuit Design

In this section, we discuss the circuit-level design of the FPGA architecture described in [1]. The technology used in this design is a 1.2-μm two-level metal n-well complimentary metal–oxide–semiconductor (CMOS) process.

Fig. 2 illustrates the signal path starting at the input of an L3-4.2 lookup table (LUT), which was selected in Section II of [1]. Following the LUT is an output stage that contains a multiplexer to select the latched or unlatched version of the output before the signal is driven through a connection block (C block) onto the routing tracks. The signal then goes through one switch block (S block), or a number of S blocks, before it enters another C block where it will reach the input of the next logic block.

In the following sections, we discuss issues that arose in the design of this path. Simulation with HSPICE[1] was used to evaluate different circuit options.

[1] *HSPICE User's Manual,* Meta-Software Inc., Campbell, CA 95008 USA.

TABLE I
AREA COMPARISON FOR THE TWO POSSIBLE C-BLOCK TECHNIQUES

Method	(Programming Bits) × (Transistors/Bit)	Inverters × 2	(Connection Transistors) × Scale	Total Transistors
Pass transistor	(8 × 16) × 5	0	(8 × 16) × 3	1024
Multiplexer	(8 × 4) × 5	32 × 2	(8 × 30) × 3	944

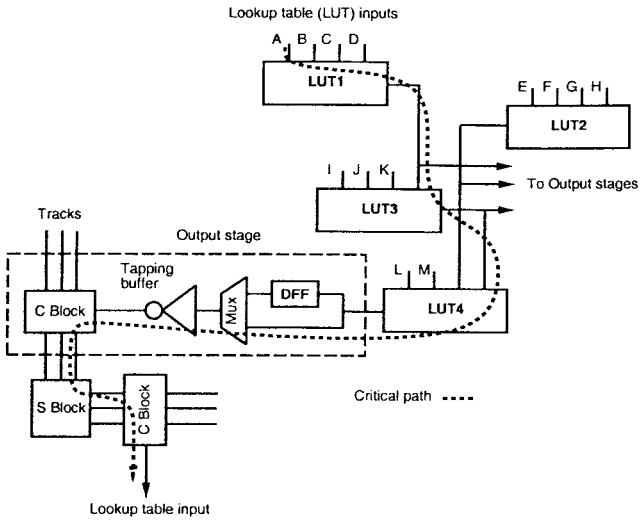


Fig. 2. Architectural definitions.

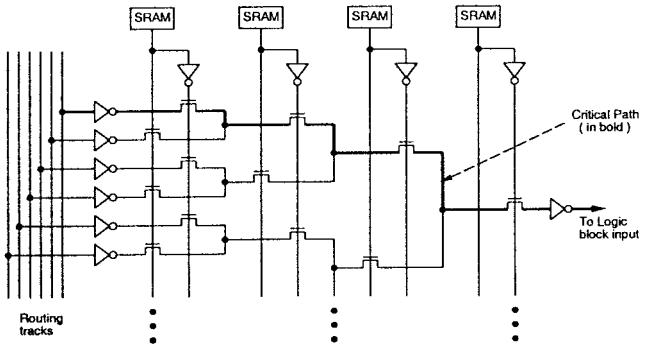


Fig. 3. Connection block input using a multiplexer.

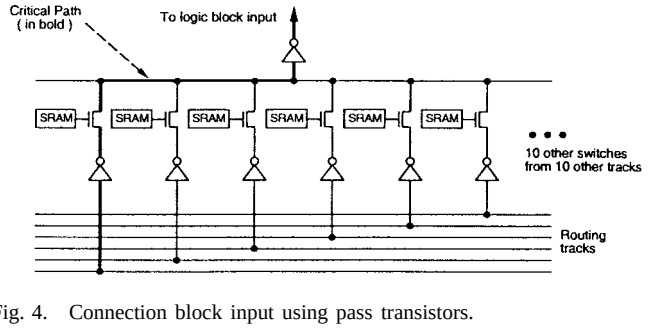


Fig. 4. Connection block input using pass transistors.

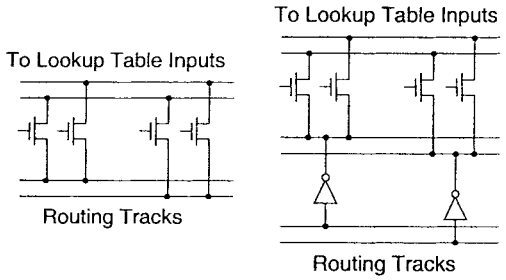


Fig. 5. C-block input isolation options. (a) Inputs with no receivers. (b) Inputs with isolation.

A. Multiplexers Versus Switches

The C block is used to connect signals in the routing tracks to the input pins of the logic blocks, and can be implemented either using multiplexers or point-to-point switches. Fig. 3 illustrates the use of a multiplexer and Fig. 4 shows a connection block using one switch for each track-to-pin connection. The boxes labeled static random-access memory (SRAM) are the programming bits. In UTFPGA1 [2], we used multiplexers for the inputs and switches for the outputs. The reasoning was that since only one wire in the channel can be connected to an input pin, there is no need for the flexibility provided by switches. This reduces the number of bits needed to select the input connection and the total number of transistors needed. For the output, having a switch from a pin to each track provides fanout capability.

Table I compares the two possibilities on the basis of transistor count. The buffer inverters from the routing tracks are not counted. In our test chip, called **L**ogic that's **E**rasable and **G**reatly **O**ptimized (LEGO), every C block is connected to two logic blocks with four inputs on each side. For the multiplexer technique, eight multiplexers with 30 transistors each are required to connect to the 16 tracks. This assumes that n-channel switches are used instead of full complementary switches, and that every track can be connected to every logic block pin. Four programming bits are needed to program each multiplexer as well as 32 inverters (4 inverters/mux × 8 muxes) to generate the complementary inputs. For the pass transistor technique, eight sets of 16 switches are required to connect to the 16 tracks. Each switch will require one programming bit. For either technique, each programming bit will require five transistors. The circuit for the programming cells is discussed in Section III-A. Unit-sized transistors are used in the programming cells and the transistors that would be used in the switch or multiplexers are about three unit transistors in area. The result is that the pass-transistor technique is about 1024 unit transistors in area, compared to 944 for the multiplexer.

Although using a multiplexer results in less area, it has a significant performance impact because the signal must go through a number of series transistors. In LEGO, we chose to use switches at both the inputs and outputs of the logic block. The price is that a significant fraction of the area is devoted to the C blocks.

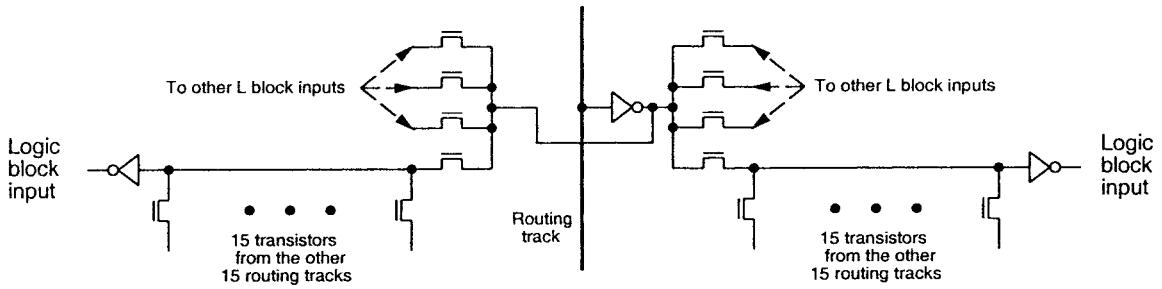


Fig. 6. Track isolation using shared buffer.

B. Isolation

Another important issue is the capacitance at the C-block inputs. With switches connected as shown in Fig. 5(a), the capacitance seen from the routing track depends on whether the switch is on or off. If the switch is off, the contribution to the capacitance is due to the switch transistor itself. If the switch is on, then any of the circuitry on the other side of switch will contribute to the load seen by the driving signal. This means that the loading will be a function of the number of C-block switches that are turned on. This is related to the problem noted in the design of the Triptych [3] logic block, and their solution was to use gate logic instead of switch logic. The problem in LEGO can be reduced by adding a buffer between the track and each switch input, which makes the load independent of how many switches are turned on. However, this still has one load per switch input. Further isolation can be provided by using the configuration shown in Fig. 5(b). In LEGO, each connection block connects the tracks to two sets of logic block inputs, so an additional optimization can be done, as shown in Fig. 6, which reduces the number of buffers required by one-half, also reducing the capacitance on the track.

C. LUT Design

For the LUT design, we have shown in [1, Sec. II] that hard-wired connections can significantly improve performance. At the implementation level, the actual placement of the hard-wired connections is also important.

Fig. 7 shows a part of the multiplexer tree in the LUT. It shows that the load of input S is only one transistor and one inverter, while the load of input P is eight transistors and an inverter. Input S is the least loaded input and also the input that determines the final output of the multiplexer. Therefore, when connecting the output of a LUT to the input of the next LUT, it is best to use the inputs that are closest to the output. In this way, the delay through the LUT's can be partially overlapped. Fig. 2 shows the hard-wired connections used in LEGO. Note that the output of LUT2 is connected to input R of LUT4 (using the input labels of Fig. 7), while the output of LUT3 is connected to input S. This is because the output of LUT3 will arrive the latest, so it should be connected to the fastest input, which is input S.

D. Full Complementary Versus n-Channel-Only Switches

Throughout the array, there are switches in the LUT, C block, and S block, thus, the design of the switches will

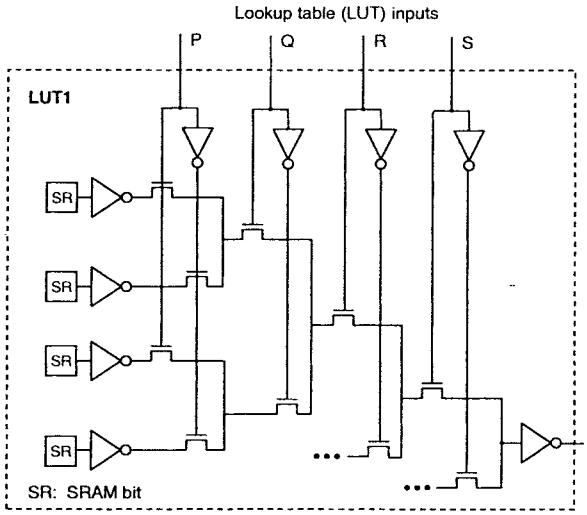


Fig. 7. Top quarter of LUT multiplexer tree.

have a significant impact on the area and speed of the array. The most obvious area gain can be made by using a single n-channel pass transistor instead of a full transmission gate. In this section, we summarize the important considerations when deciding between the two alternatives.

By using a pass transistor, the area benefit is not just because there are half the transistors, but also because of a savings in well areas, which can be significant depending on whether the switches are scattered enough that individual wells must be provided for each transmission gate. In addition, internal nodes of n-channel pass transistor circuits may not require contacts, saving area and reducing junction capacitance. There is also a savings in routing for the complementary clock or providing the local clock inverter if that technique is used.

For signal integrity, the pass transistor causes a V_t voltage drop when passing highs and has a slower rise time toward the end of a low-to-high transition. These effects can be partially compensated by lowering the switching threshold of the succeeding gates. Unfortunately, the reduced high also means that the pull-up transistors in succeeding gates will not be fully turned off, resulting in static power dissipation. The problems become even greater as supply voltages move toward 3 V because the worst case V_t voltage drop could still be around 1.2–1.3 V ($V_t = 0.7$ V nominal at room temperature), significantly degrading the noise margin. This means that pass transistors are not viable unless the gate voltage of the pass transistor is bootstrapped to above the supply.

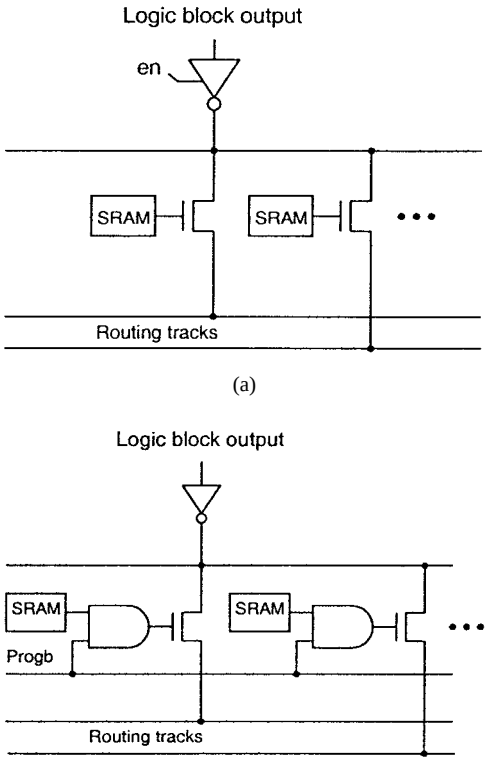


Fig. 8. Hazard prevention alternatives. (a) Using a tristate buffer. (b) Using a buffer and the **progb** signal.

As for speed, consider the construction of a transmission gate by the addition of a p-channel transistor that is 2.5 times the size of the n-channel transistor. The resistance changes by 0.5, but the junction capacitance increases by 3.5 or more, depending on the use of contacts. This means that the *RC* delay is at least $0.5 \times 3.5 = 1.75$ times larger, so that the n-channel pass transistor would normally be faster, although careful simulations would always be required to make sure that this is true.

In LEGO, which was designed for 5-V operation, we chose to use pass transistors because we could get a better propagation delay and reduced area. To compensate for the V_t drop, we used simulations to adjust the switching point of the gates following the switches and to make sure that the propagation delay for rising and falling signals was balanced.

E. Power-Up and Programming Protection

A difficult problem occurs between the time after chip power-up, but before the programming has been downloaded. At this point, the configuration bits have not been programmed and it is possible that there are conflicts between drivers on the routing tracks. This problem has been considered previously. In UTFPGA1 [2], the output drivers are disabled using a tristate buffer, as shown in Fig. 8(a). The buffers will be disabled during power up and programming. This design, however, impacts the speed because the disabling transistors of the tristate buffer will be in the critical path. Fig. 8(b) shows the method used in LEGO. The output routing switch is controlled not only by a configuration bit, but also by another signal, labeled **Progb**, which is a global signal activated during power up that disables all connections to the track, preventing the hazard.

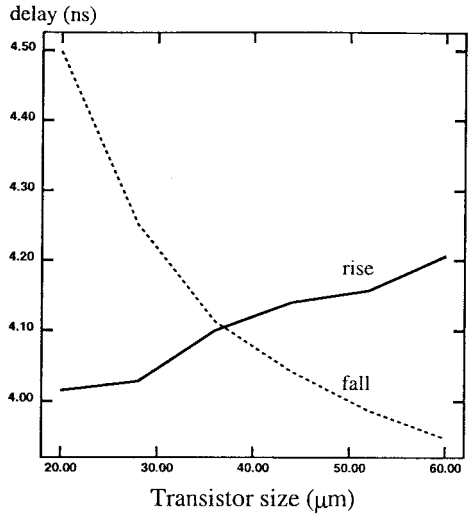


Fig. 9. An example of varying switch sizes for transistor sizing.

F. Transistor Sizing and Signal Transition Balancing

Transistor sizing is a major concern for achieving high performance. Small transistors limit the current, while large transistors have a higher capacitive load. Therefore, some optimization is required. In Triptych [3], the method of logical effort [4] was used to assist in transistor sizing and buffer insertion for speed. We chose to use actual simulation because we felt it was easier to balance the rise and fall times. We began by simulating the components of the critical path individually, but later tuned the design by simulating the entire critical path at once.

Our initial assumption was that the n-channel pass transistors used for the S-block switches and C-block outputs had to be at least as large as the transistors of the drivers so as to reduce the effects of the switch sizes on the critical path delay. This allowed us to choose an initial driver size. We then tuned the design by simulating with different switch sizes. An example result is shown in Fig. 9, which shows the effect of varying the logic block output driver size on the total delay of the critical path. It can be seen that the rise and fall times are equal at about 37 μm for the size of the n-channel driver pulldown. In LEGO, we actually used 34 μm because, in our initial simulations, we used nine tracks in the channels, which was later changed to 16 tracks, and thus a higher load.

It should also be noted that the tuning was done by starting at the end of the critical path. This gives a better idea of the loads to be driven as sizes are fixed.

Fig. 10 shows the critical path through the logic block section, and Fig. 11 shows the critical path in routing using the tuned transistor sizes. From simulation, the total delay is about 4.1 ns along the critical path, broken into about 2.7 ns for the logic block and 1.4 ns for the routing.

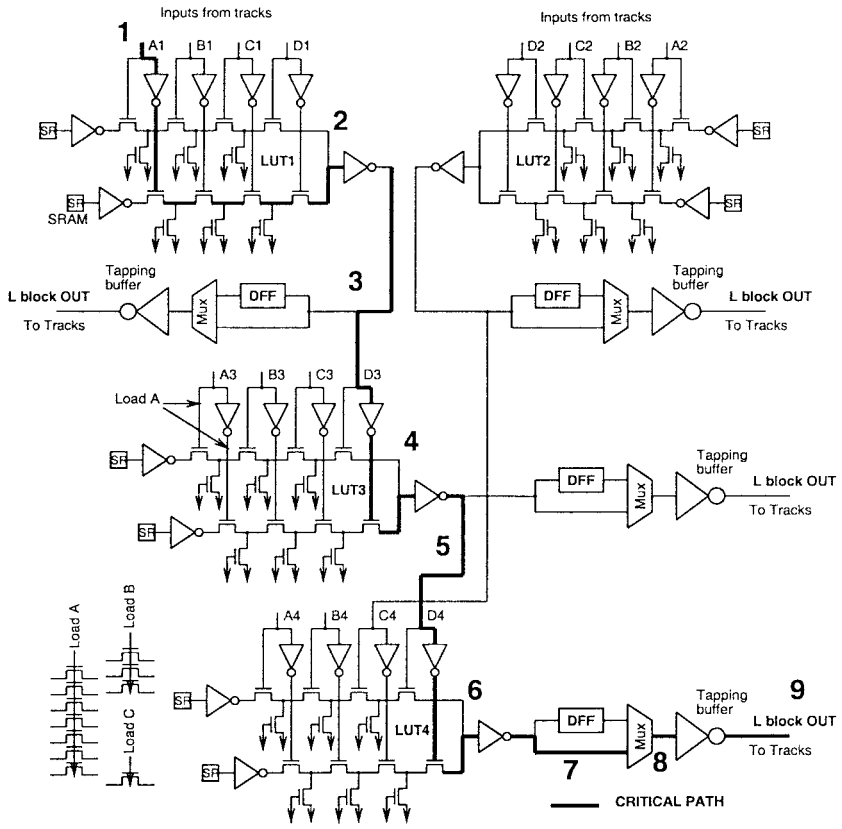


Fig. 10. Details of critical path through the logic block.

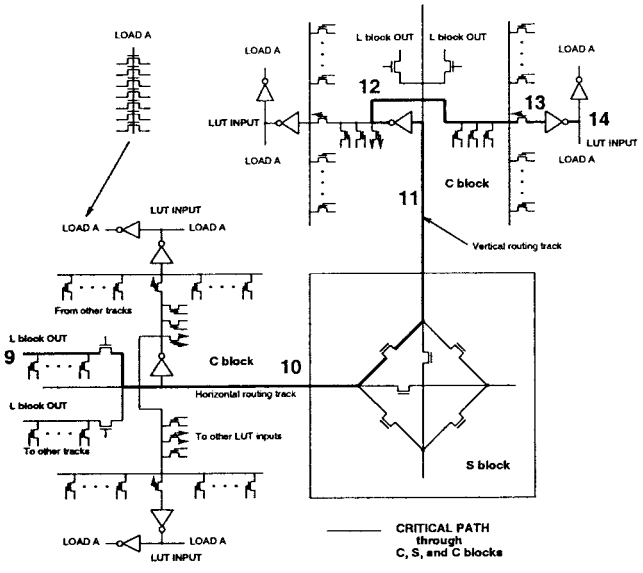


Fig. 11. Details of critical path through the routing network.

III. Physical Design and Layout

Besides the concerns for optimizing the speed and area of the circuits, we also considered some of the issues related to the overall physical design and layout of the chip that would be used to implement the FPGA circuit. In this section, we will describe the programming architecture of LEGO and some of the layout issues.

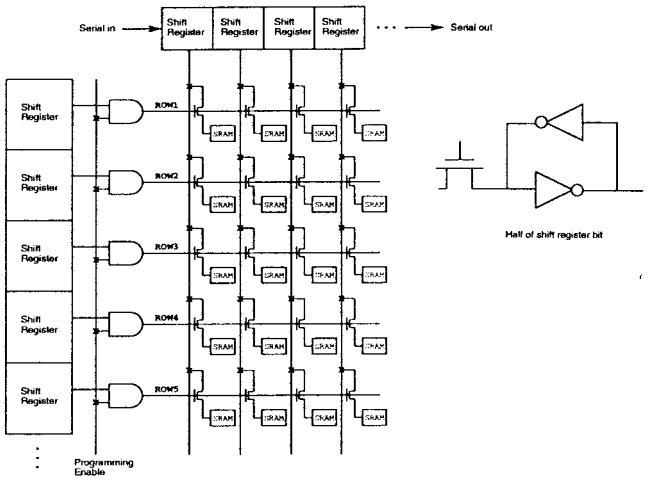


Fig. 12. Serial–parallel programming of the configuration bits.

A. Programming Architecture

Three techniques for programming the configuration bits were identified. The simplest is to connect all of the bits into a long shift register. This requires only one pin to input the programming data. Although this is the slowest method, the speed of programming was not of concern in this project. Of greater concern is that each bit must contain a master–slave section because it behaves as a shift register and, hence, each bit is very large. An alternative, which requires smaller bit cells, is to organize the programming bits into

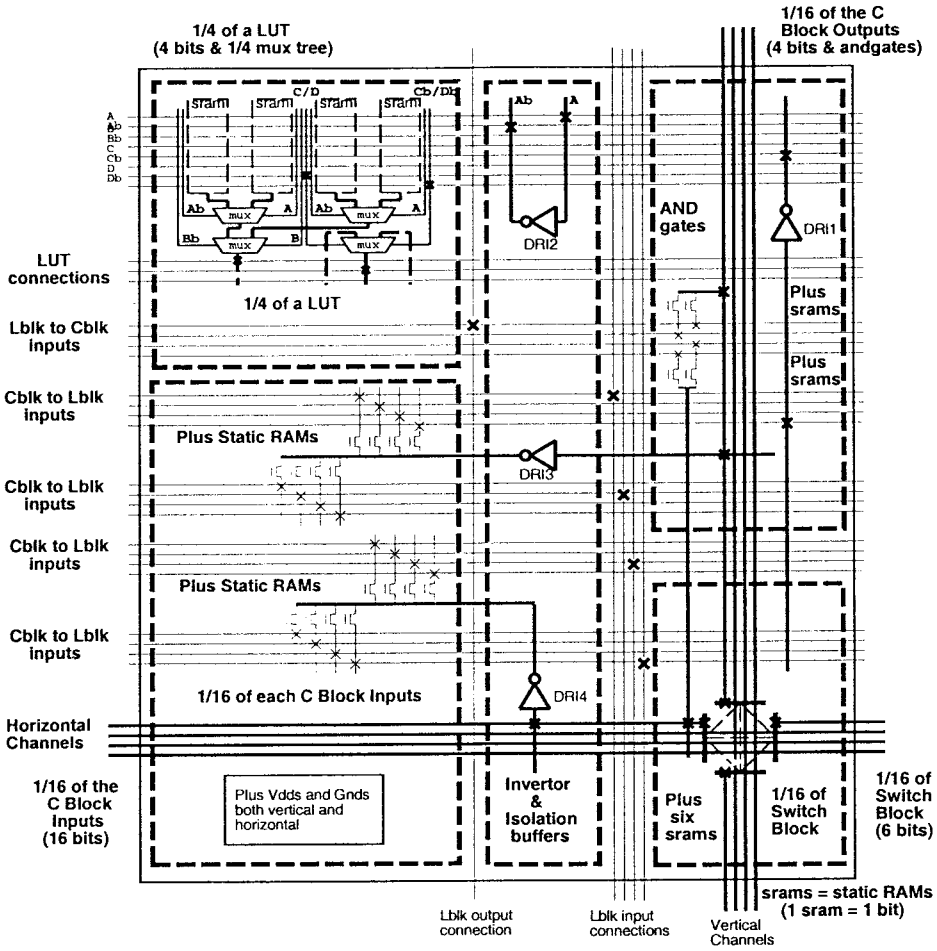


Fig. 13. Details of the minitile (1/16 of a logic tile).

rows and columns, as in a regular memory. This will require significantly more pins for the parallel data as well as for selecting the word to be programmed.

A compromise of the serial and parallel techniques was used in LEGO and is shown in Fig. 12. This serial–parallel technique uses a master–slave shift register to serially load a row of bits. The vertical shift register is initialized with the top bit on and the remainder off. This bit is shifted after each row is programmed to select the next row. After a row of bits has been loaded in the top shift register, the whole row of data is copied into the selected row in one clock pulse. RAM's that store the data bits do not have to be master–slave because the data is not shifted beyond this point. This halves the size of a programming bit as compared to the scheme that uses a single shift chain. This technique still leaves the option of programming any row, as in the fully parallel method, but it requires fewer programming pins.

The regular structure of the serial–parallel architecture also fits well with the layout methodology used for LEGO, which is described below.

B. The Minitile Layout Style

We wanted to find a better layout style than just trying to attack the problem purely as a full-custom design. When doing very large scale integration (VLSI) layout, it is desirable to find some regularity in the topology to make the task easier. In an FPGA, the first level of regularity is the arraying of the tiles, as shown in Fig. 1, where a tile includes an L block, an S block, and two C blocks. This partitioning was used for UTFPGA1 and in LEGO. However, even the full custom layout of a tile is a significant task, and it is desirable to find even more regularity. In UTFPGA1 [2], we attempted to use the same C-block layout for both the vertical and horizontal channels, but this resulted in a very poor tile floor plan because it was difficult to pack the pieces together. For LEGO, we wanted a solution that had a finer granularity.

Instead, we divided the tile into 16 *minitiles*. Each minitile contains 1/16 of the L block, S block, and C blocks. The tile[2] is implemented by creating a 4×4 array of minitiles, and adding vias at the appropriate locations to customize the local function of each minitile. The flip-flops, muxes to select the latched or combinational outputs, and output tapping buffers are at the edge of the array since there are only four of each. These are shown as part of the output stage in Fig. 2.

The motivation for this topology comes from the topology used for programmable logic arrays (PLA's) [5], where the goal is to make the overall structure regular and have irregularity restricted to the final programming. Fig. 13 shows the details of the minitile.

[2]We will use *tile* to denote the complete tile, and *minitile* to denote the smaller building-block piece.

The major problem with dividing the functionality of the tile across 16 minitiles is the routing needed to interconnect the minitiles. It is difficult to organize the routes correctly, and it is likely that the extra wires will cost more in area than a nonminitile approach. The area issue is addressed further in Section IV. Fig. 13 shows all the necessary wiring. Examples of via placement are shown by the X's. The horizontal and vertical tracks of Fig. 1 are shown in heavier lines intersecting at the switch block (S block).

The choice of how many tracks to use depends on the actual size of the overall array of tiles in a chip and on the number of minitiles. For larger FPGA's, more tracks would be needed and, in the case of LEGO, the number will be a multiple of four because of the minitile array size.

Each minitile contains one-quarter of a mux tree, which is really a 4-to-1 mux that has the SRAM bits at its inputs and a separate 2-to-1 mux. The three mux interconnect tracks going through the quarter mux-tree are used to connect between the minitiles to form the full four-input LUT's. These tracks are segmented in the minitile to allow sharing of the tracks and can be connected with a piece of metal as needed during final configuration. One of the 2-to-1 muxes is not required when hooking up the full LUT.

In addition, the outputs of the LUT's are routed on the *LUT connection* tracks to the edge of the tile for connection to the D flip-flops and to the driver to the tracks. The output of the driver is routed back into the minitiles along an *L blk to C blk inputs* track for connection to the C block for output. The remaining vertical wires are used to distribute signals to the minitiles along the vertical dimension of the array.

Driver *DRI2* is the inverter shown at the inputs to the LUT's in Fig. 7. Driver *DRI1* drives the input from the C block to the LUT inputs.

The drivers marked *DRI3* and *DRI4* are the track isolation drivers, as shown in Fig. 6. The transistors at the outputs of these drivers are the switches described in Section II-A. In this partitioning, the C block includes connections to the south- and east-neighbor tiles so some of the wires in each minitile are used to route signals to the LUT inputs in those neighbors.

Using the minitile methodology, we sacrifice the potential for global layout optimization across the whole tile in exchange for a simpler full-custom layout task. However, we could afford to spend significant effort on the layout of the minitile. The floor plan of the minitile is shown in Fig. 14. The vertical dimension is limited in pitch by the number of horizontal metal two tracks. In the horizontal dimension, there are still some visible internal vertical routing channels that do not go over any active area. If we had a third metal routing layer so that the vertical routing could go over active area, then the horizontal dimension could be reduced by about 25%. With this area removed, the layout efficiency would be quite good, as most of the metal routing would be on top of active area, which cannot be compacted much. Since we did not attempt a full custom layout of the tile, we do not have a quantitative measure of our layout efficiency. However, we feel that by using this methodology, there was not a great sacrifice in overall density, and shortening the time to do the layout was more important.

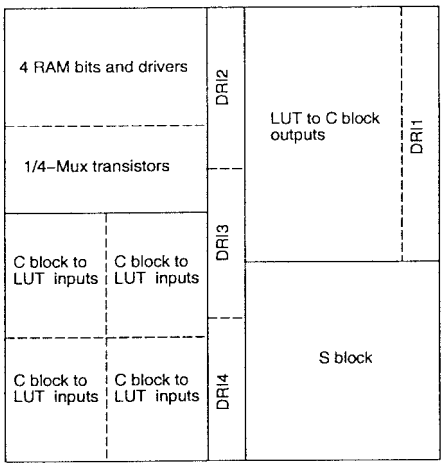


Fig. 14. Floor plan of minitile.

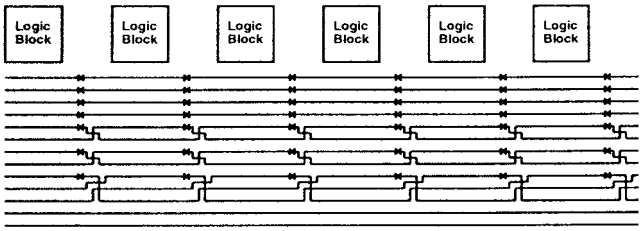


Fig. 15. Segmentation scheme for use in a tile layout style.

TABLE II
PERFORMANCE COMPARISONS FOR THE ATN BENCHMARK

FPGA	Published (ns)	Adjusted (ns)
Crosspoint CP20420-1	37.4	12.4
Crosspoint CP20420-0	54.2	29.2
Actel A1280-1	70.0	45.0
Xilinx XC4005-5	41.4	16.4
LEGO	n/a	11.5

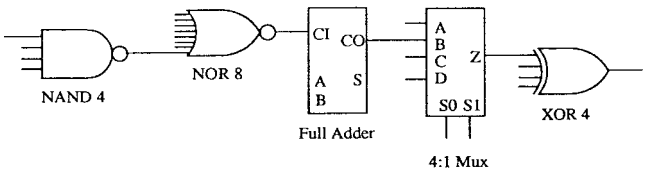


Fig. 16. The ATN benchmark circuit.

An added benefit that has not been explored at this point is that, with this ability to configure the tile after the hard layout work is done, it is possible to try other hard-wired interconnections for the LUT's and other channel segmentation distributions. This may be of benefit if our work in nonhomogeneous logic blocks suggests that a mix of tiles with different types of hard-wired connections is beneficial.

C. Other Layout Issues

Our experience with UTFPGA1 showed that doing a full custom layout of a complete tile is extremely difficult, and

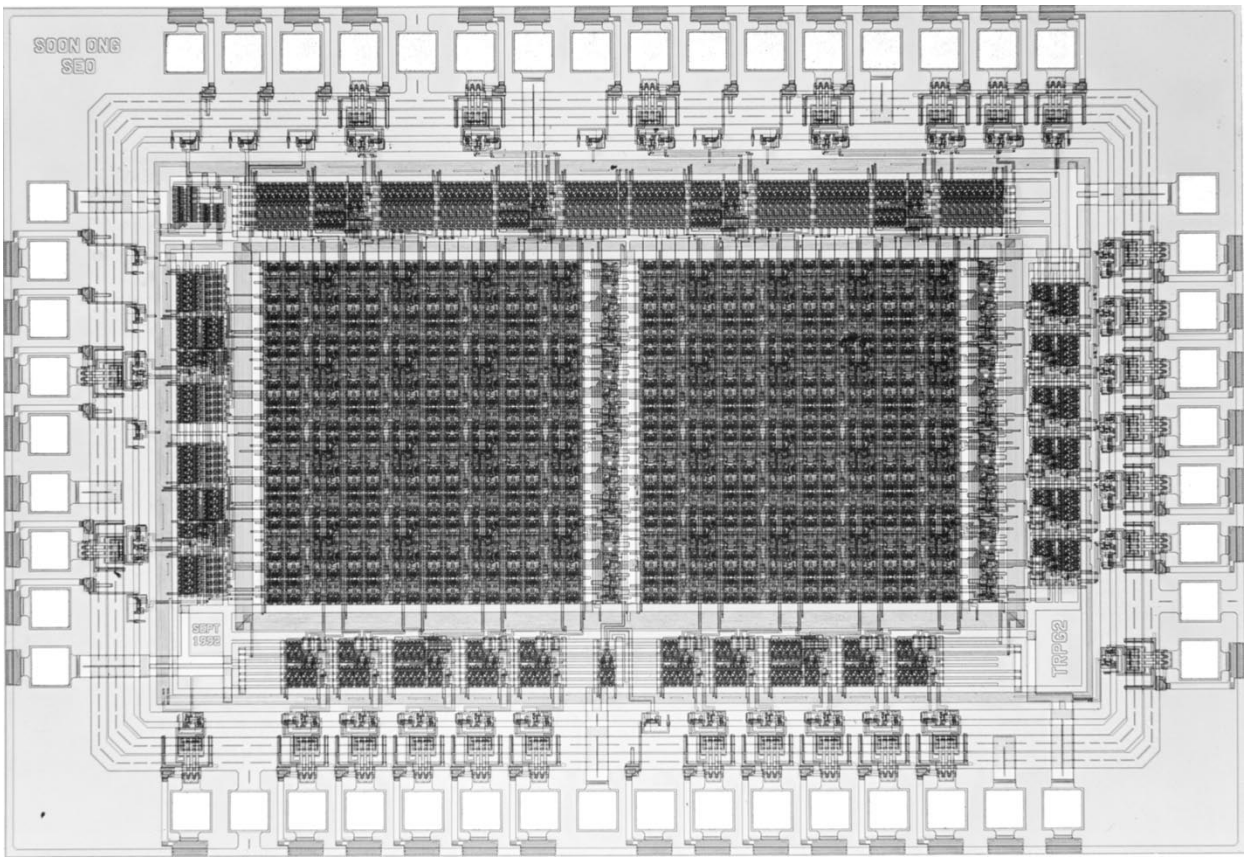


Fig. 17. Photomicrograph of the LEGO test chip.

distributing power was one of the major problems. With the minitile methodology, it is easy to have a rectangular floor plan for the tile, and this makes it easier to distribute power throughout the array.

As described in [1, Sec. III], LEGO uses a segmented routing scheme. The segments repeat at intervals of one, two, and three tiles. To achieve this with a single tile and without requiring customization of the interconnect between the tiles, the scheme shown in Fig. 15 was used. This scheme will work as long as the number of tracks for a repeat interval is the same as the repeat interval. In LEGO, there are four tracks of length two and three tracks of length three.

IV. Results and Performance

The goal of this project was to design an FPGA with high performance without greatly sacrificing area. In this section, we analyze how well this goal was met. We first provide area measurements and speed measurements from simulation. Subsequently, measured results from a fabricated test chip are provided.

A. Area and Speed

The size of each tile, which contains four four-input LUT's, four D flip-flops, and connections for 16 tracks, is 1240 μm by 1184 μm. By way of comparison, the Xilinx 3000 tile in the same feature size technology (1.2-μm CMOS), which contains the equivalent of two four-input LUT's, two D flip-flops and connections for eight tracks (about half the functionality of the LEGO tile) is roughly 500 μm by 400 μm. This is about a factor of four smaller, which comes from the fact that there is twice as much logic and routing in the LEGO tile, minitile strategy, and design decisions that sacrificed area for speed. In particular, the large size drivers and the switches (rather than a multiplexer) used in the C block take up significant area. Each LEGO tile requires 492 programming bits.

From simulation, we measure a delay of 4.1 ns for the critical path shown in Figs. 10 and 11. The measured delay for LEGO is simulated time for nominal conditions—not worst case.

An attempt to compare LEGO with commercial FPGA's is shown in Table II. It is only provided as a point of reference since doing more complicated circuits across a number of architectures would be difficult. The circuit used is shown in Fig. 16. The figure and some of the numbers in Table II are taken from some performance benchmarks published by Crosspoint Solutions, Santa Clara, CA.[3] The numbers in the *published* column are the reported numbers assuming that the output goes to a pad driving a 50-pF load. The *adjusted* numbers subtract 25 ns as an attempt to take out the effect of the output pad.[4] This is a gross approximation, as we do not have the numbers for the delays

[3] "Crosspoint solutions, performance benchmarks, rev 1.01," Aug. 1992.

[4] The Crosspoint benchmarks erroneously used data for a Xilinx 4005-7 part, which does not exist. The Xilinx data has been multiplied by 5/7 to reflect the speed of a 4005-5.

due to only the logic. However, it can be seen that LEGO compares well even though the others are implemented in higher performance technologies. Much of this is because of the larger functionality of the LEGO logic block. The LEGO number is from an HSPICE simulation under nominal conditions. The other numbers are for worst-case process conditions.

B. Fabrication Results

A chip comprising two LEGO tiles was fabricated and the photomicrograph is shown in Fig. 17. The extra wires that exit the column and rows were either attached to pads or "wrapped" around the tiles to connect to the other side of the channel. This chip has 50 pads, including 12 power and ground, 23 signal pins into the FPGA core, eight programming pins, five control pins, and two test pins.

The testing of the chip showed that the logic blocks were not fully functional, and the cause has not been determined, though we suspect that having only nominal parameters to simulate the pass transistor logic may have lead to some threshold problems. However, we were able to make reliable connections using the programmable routing.

A smaller test chip was also fabricated. It contained only four of the minitiles, which was enough to build one of the four-input LUT's. From this chip, we were able to make delay measurements that included one part of the logic block. Based on these measurements, we observed a pad-to-pad delay through one S-block switch transistor of 19 ns, where the output was unloaded. The pad-to-pad delay through part of the critical path, shown in Figs. 10 and 11, was measured. The measured path exits the logic block through the tapping buffer at node 3 of Fig. 11, rather than node 6. The measured delay was 26 ns, giving a delay of about 7 ns when the pad delay is eliminated. The simulated delay for this path is about 3 ns. Speed measurements were done using a tester and have an error of ± 1 ns. We attribute most of the difference to the fact that the simulations are based on nominal parameters.

V. Conclusions

We have described the low-level circuit design and layout methodology used in the design of a high-performance FPGA. The important issues in circuit design that affect the critical path delay have been shown. We have focused on achieving a high-speed design, while keeping in mind the area tradeoffs. The most critical issues are the design of the switches and minimizing the capacitance of the routing network. As well as the circuit issues, we have also presented a layout methodology for FPGA's that reduces the amount of custom layout required without sacrificing excessive density.

Our results have shown that the LEGO design compares favorably with existing commercial FPGA's. The most important contribution of our work is that we have gained a greater understanding of the real design issues in an FPGA circuit.

As technology continues to scale, supply voltages will decrease, and the number of metal layers will increase. The electrical properties of the new technologies will require that new tradeoffs be considered in the circuits and layouts of FPGA's. These properties will also impact the architectures of both the logic blocks and routing. It is clear that both the architectural and circuit-level aspects of FPGA design must continue to evolve.

Acknowledgment

The authors wish to thank J. Pristupa for his support of their comptuer-aided design (CAD) and VLSI Laboratories, P. Chow for help with the testing, and K. Martin for his insights into circuit design using pass transistors. The Canadian Microelectronic Corporation provided fabrication access using the facilities of Nortel. Their help is always appreciated. The authors also acknowledge the valuable feedback provided by the referees.

References

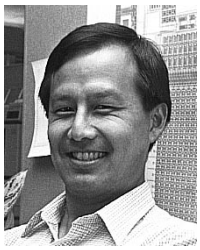
[1] P. Chow, S. O. Seo, J. Rose, K. Chung, G. Páez-Monzón, and I. Rahardja, "The design of an SRAM-based field-programmable gate array—Part I: Architecture," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 191–197, June 1999.

[2] 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. Nashville, TN: Abingdon, 1991, ch. 3.2, pp. 91–102.

[3] 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. Nashville, TN: Abingdon, 1991, ch. 3.1, pp. 75–90.

[4] I. E. Sutherland and R. F. Sproull, "Logical effort: Designing for speed on the back of an envelope," in *Proc. Univ. California at Santa Cruz Conf. Advanced Res. VLSI*, Mar. 1991, pp. 1–16.

[5] C. Mead and L. Conway, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.

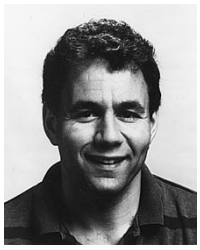


Paul Chow (S'79–M'83) received the B.A.Sc. degree with honors in engineering science, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, Ont., Canada, in 1977, 1979, and 1984, respectively.

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.