A 1.2μm CMOS FPGA Using Cascaded Logic Blocks and Segmented Routing

**Paul Chow, Soon Ong Seo, Dennis Au, Terrence Choy,
Bahram Fallah, David Lewis, Cherry Li and Jonathan Rose**

Department of Electrical Engineering
University of Toronto
Toronto, Ontario, Canada M5S 1A4

This paper describes the architecture and implementation issues of a field-programmable gate array designed at the University of Toronto. The chip is designed in a 1.2 μm double-metal CMOS technology, and was recently sent for fabrication. The logic block is a cascade of three four-input lookup tables hard-wired together for high performance. A segmented routing architecture with segments that span one, two and three logic blocks in both the vertical and the horizontal directions is used. Several design issues including power-on and programming safety, connectivity alternatives, and layout of the basic tile are discussed.

OVERVIEW

Field-Programmable Gate Array (FPGA) technology has grown rapidly since it was first described by Xilinx (Carter *et al* 1986). Since then, there have been newer versions from Xilinx (Hsieh *et al* 1990), and many other commercial efforts (Ahrens *et al* 1990, Algotronix 1989, Wong *et al* 1989, AMD 1990, Concurrent 1991, Plessey 1989, Plus 1989, Muroga *et al* 1991). All of the reported work focuses on architectural features or programming technology, but very little has been discussed about implementation and circuit design issues. We describe here the architecture and implementation of an FPGA at the University of Toronto (UTFPGA1) that uses a 1.2 μm double-metal CMOS technology. The work is part of our research into logic block and routing architectures (Rose *et al* 1990, Rose and Brown 1991, Singh *et al* 1991).

The goal of designing UTFPGA1 is to aid our understanding of FPGA implementation issues so that we can design better models for our architectural studies. The effort of doing a real implementation has given us insight into many design tradeoffs and considerations. We begin in the next section by giving an overview of the architecture. This is followed by a more detailed discussion of the design issues and the circuits used. Some statistics about the chip are then given, and we conclude with an analysis of the design and our future directions.

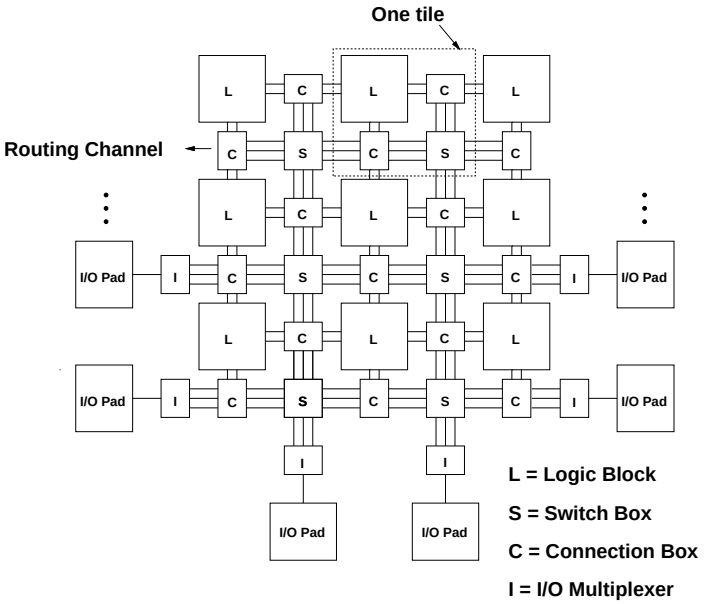

Figure 1 General architecture of UTFPGA1.

ARCHITECTURE

The general architecture of UTFPGA1 is shown in Figure 1. The logic block (L) contains the functionality of the circuit while the connection boxes (C) connect the logic block pins into the neighbouring channel. The switch box (S) makes connections between adjacent horizontal and vertical channel segments. Connections to the I/O pads are done through I/O blocks (I), which connect to the routing channels. Configuration is done by programming static memory configured as shift registers. We have designed a single tile that contains one logic block, two connection boxes and one switch box. This tile can then be arrayed to any size.

The logic block contains three cascaded four-to-one lookup tables as shown in Figure 2. This configuration was chosen because recent results (Singh *et al* 1991) have shown that significant gains in optimizing for delay can be achieved by having some hardwired connections between logic blocks. The block also contains a resettable D flip-flop.

The routing architecture has tracks segmented into lengths of one, two, and three tiles. Such an architecture provides fast paths for longer connections, improving FPGA performance. Figure 3 illustrates the routing architecture. An 'X' indicates a break in the track from which turns or straight connections can be made. Note that the crossovers in the channel allow all tiles to be identical and still have the required segmentation. The identical structure is used in the vertical channels. The segmentation is discussed further when tile floorplanning is examined below.

DESIGN ISSUES

Many design issues must be solved to build a useful FPGA circuit. This section describes the trade-offs considered, and some of the circuits used.

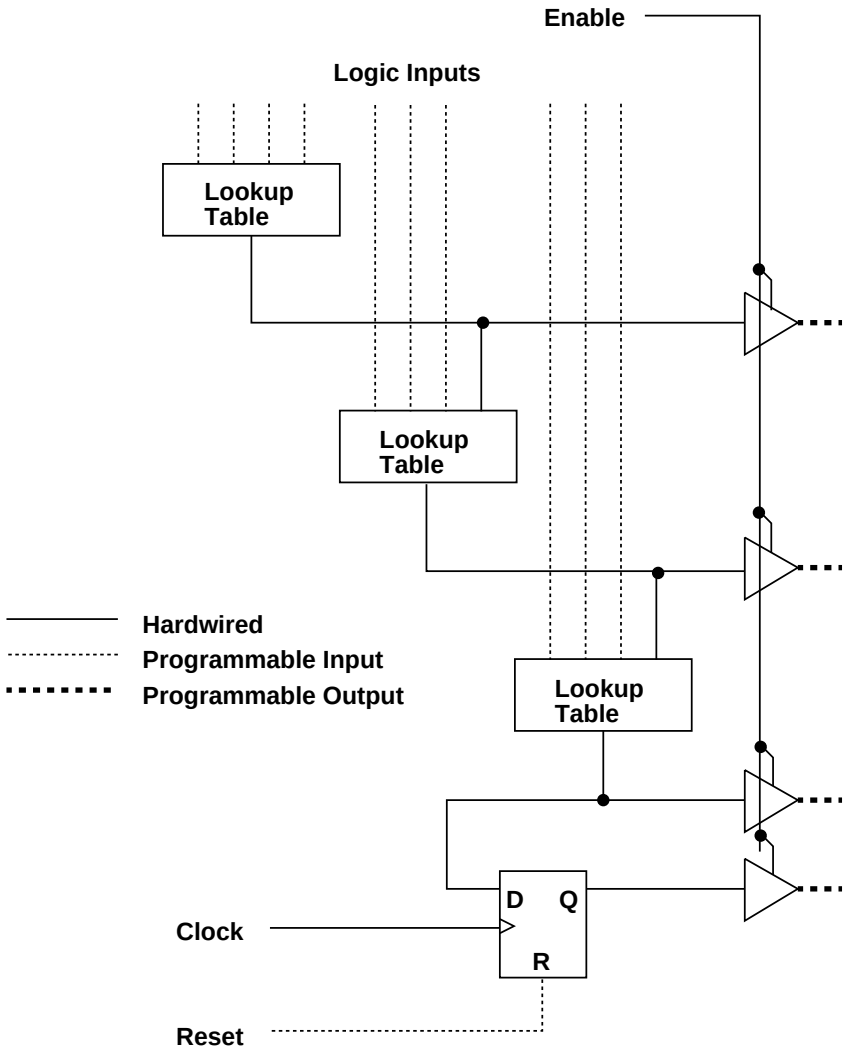

Figure 2 Logic block architecture.

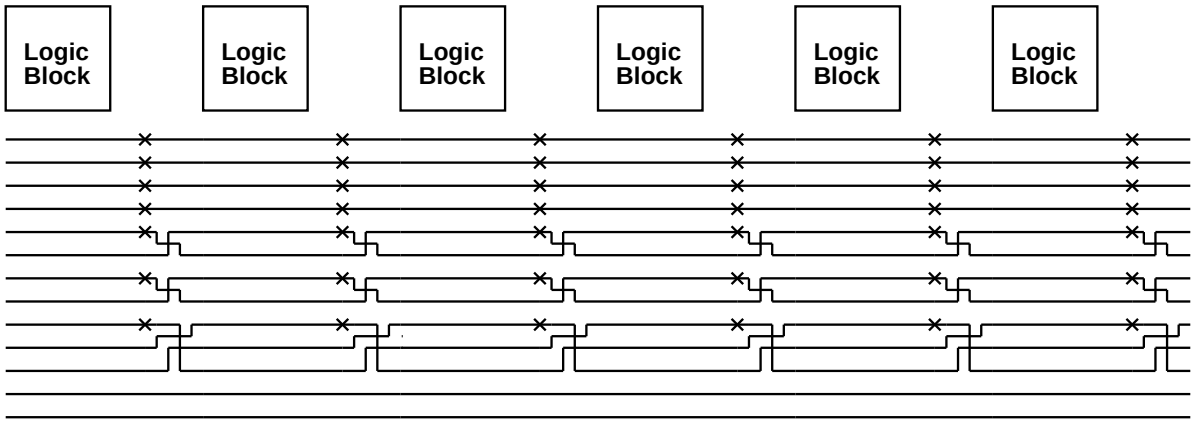

Figure 3 Segmented channel structure.

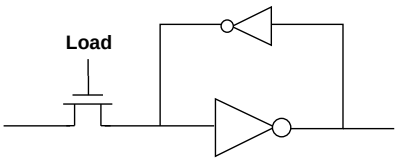

Figure 4 Memory cell circuit.

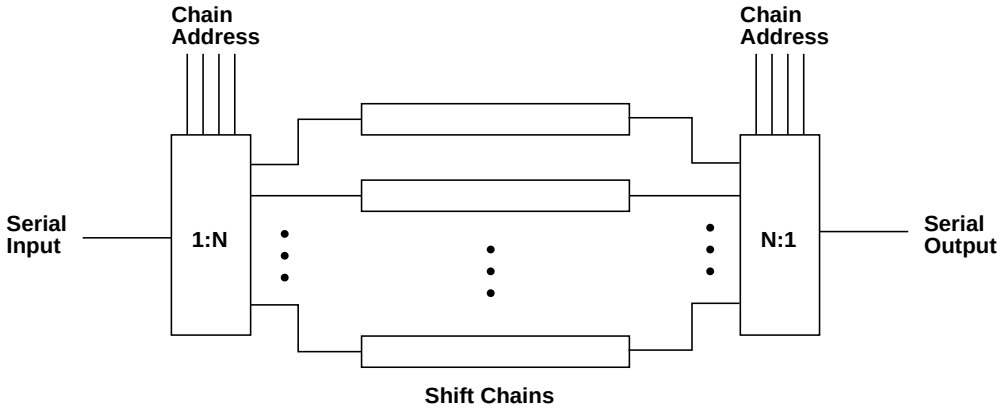

Figure 5 Shift chain organization.

Programming

The use of static memory as the programming technology was the only choice available because we had no way to build the antifuses or EPROM cells used by some commercial FPGAs (Gamal *et al* 1989, Wong *et al* 1989). This memory is used to describe the logic functions to be implemented, and to set the positions of the switches in the routing network. This is the approach taken by several commercial designs (Hsieh *et al* 1990, Plessey 1989, Muroga *et al* 1991, Concurrent 1991, Algotronix 1989). It also provides us with several advantages: since we only receive a limited number of chips, it is better that they can be reused many times, and testing of the chip is easier since we can take advantage of the programming method to allow us to shift out the contents of the configuration memory.

The memory cell is a static flow-through latch with a single n-channel access transistor. Figure 4 shows the logic for the cell. A shift register stage uses two of these cells in a master-slave configuration. The feedback inverter is weakened so that the input can overdrive it. This circuit is similar to the one used by Xilinx (Xilinx 1991) and in MIPS-X (Chow 1989).

We connect the memory cells into shift chains using a two-phase clock, and separately address the shift chain in each tile during programming. Figure 5 shows this organization. By dividing the shift chain into many parts, we achieve better testability, and reduce the possibility that a manufacturing defect will cause the entire chip to be nonfunctioning. If a single shift chain is used, a defect in one bit would mean that a significant portion of the chip could not be programmed. The output multiplexer in Figure 5 allows the contents of the shift chains to be read.

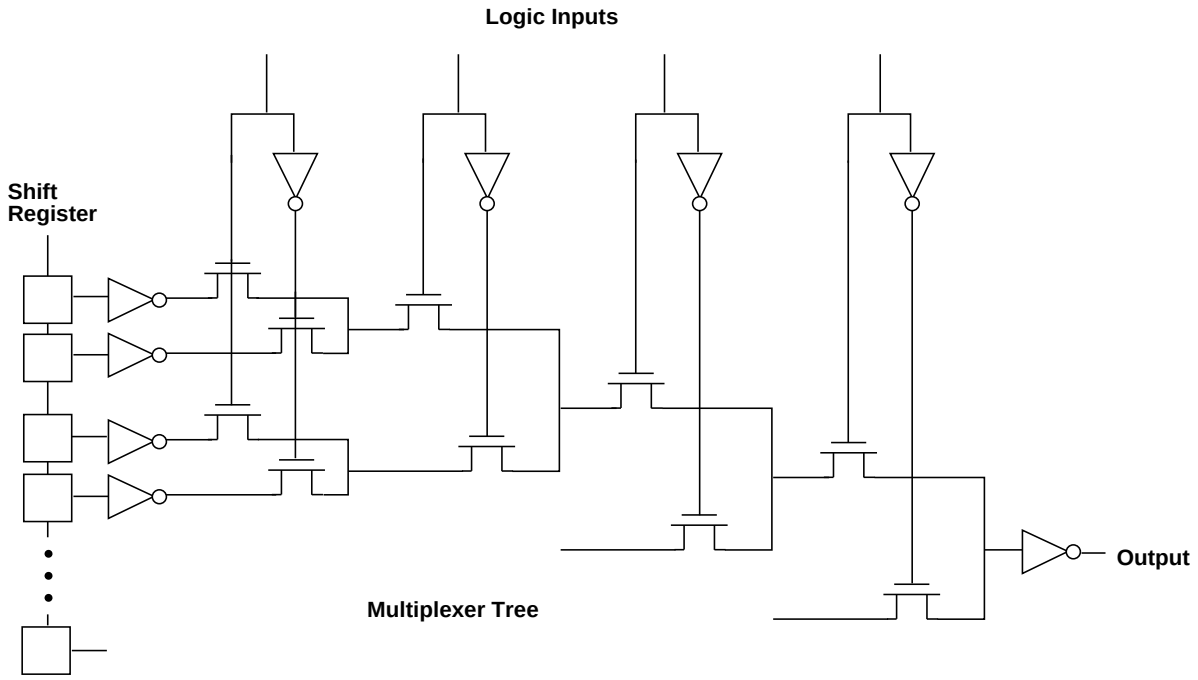

Figure 6 Lookup table circuit.

Logic Blocks

Our research into logic block architectures (Singh *et al* 1991) has shown that using a hierarchical structure can increase the performance of the resulting circuit. From our early results we chose to implement a cascade of three lookup tables as shown in Figure 2. The actual lookup table is implemented as a shift register chain storing the data bits, and an n-channel pass transistor multiplexer tree to select the output. Figure 6 shows a portion of the circuit. The inverters at the output of the shift registers prevent charge sharing, which could disturb the state of the memory cells. They also provide some extra drive through the pass transistors. The output of the multiplexer is sensed by an inverter. The inverter transistor ratios could be adjusted to lower the switching threshold and compensate for the degraded highs, but this was not done in this implementation. Xilinx uses an extra mask step to adjust the threshold of the p transistor in the inverter (Xilinx 1989).

The cascading of the lookup tables is done by connecting the output of the preceding lookup table to the last stage of the multiplexer tree in the next stage. This reduces the critical path through the cascade to the delay through one lookup table and the final stages of the last two lookup tables. To reduce the delay for functions that do not require all the lookup tables, the outputs of the intermediate lookup tables are also available external to the logic block.

Switch Boxes

The switch boxes (S boxes in Figure 1) consist of the n-channel pass transistors that are used to connect segments of tracks together, and the shift register cells used to configure the switches. Of the 13 lines entering each side of a switch block, six pass directly through because they are the longer segments that do not terminate at that box. The remaining seven segments can switch to any

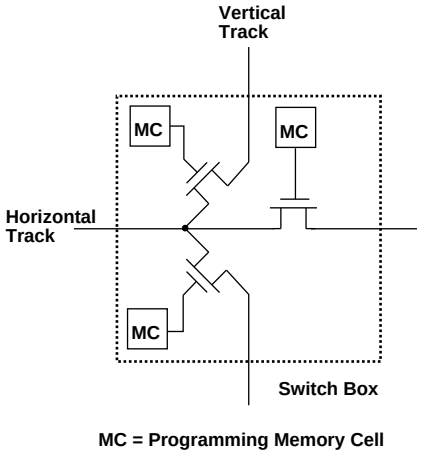

Figure 7 Switch box configuration.

of the remaining three sides of the box. Figure 7 shows the switch box configuration for a track entering from the left. This level of flexibility in the switch box is sufficient to achieve complete routability with only slightly more than the minimum number of tracks (Rose and Brown 1991), assuming sufficient flexibility in the connection boxes.

Connection Boxes

The connection boxes connect the inputs and outputs of the logic block to the channel. In this implementation, each input can select from five of the 13 possible track segments. There are four outputs from the logic block, with one output per side. Each output can also connect to five possible tracks. Experience from routing this chip has shown that this is insufficient flexibility and this is predicted by our architectural studies (Rose and Brown 1991) on routing flexibility. A better choice would have been to have at least eight possible track segments for each input or output port.

Two choices for the implementation of the connection boxes were considered. All input connections could be done using a multiplexer from N tracks to 1 logical pin, or a single switch for each track to pin connection. Using a multiplexer will reduce the number of programming bits needed because the bits are encoded instead of having one bit per switch. This reduces the area requirements. For inputs to the logic blocks, using a multiplexer makes sense because usually no more than one wire in the channel is connected to any input. However, it may be desirable to have an output from the logic block connect to more than one wire in the channel to provide fanout of the signal. To achieve this flexibility, having a switch to each wire in the track would be required, meaning that there would be one bit per switch. Any attempts to reduce the number of bits by encoding them and using demultiplexers would prevent any fanout possibility. For these reasons, the inputs are multiplexed and the outputs are individually switched.

The delay measurements show that there is significant delay incurred in the connection blocks, and this is mostly due to the input multiplexers. Using simple switches would require more area but improve the speed. More details of the circuit delays are given below.

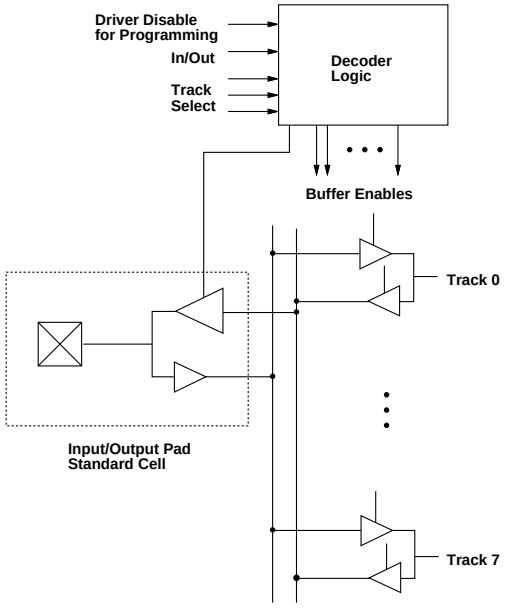

Figure 8 I/O block logic.

I/O Blocks

The I/O block is a multiplexer that connects the pad to eight tracks in the channel. The design of the I/O blocks is based around the input/output standard cell pad from CMC (Canadian Microelectronics Corporation 1990). Figure 8 shows the logic of the pad and the multiplexer. In retrospect, it might have been sufficient to use an n-channel device, as in the switch boxes, instead of the output buffer to the pad. This would require much less area but may increase the delay through the block. The input buffer is required to provide some drive onto the track segment and to provide a means for disabling the drive onto the bus during configuration of the device.

Floorplanning a Tile

In focusing the design on a single tile, the goal is to design a tile that can be arrayed in both the horizontal and vertical directions. There are several physical design issues that must be considered: track segmentation, power distribution, and the organization of the tile components.

It was not immediately obvious that any repeatable pattern existed that would allow some form of track segmentation, and this is probably true if an arbitrary number of different track lengths are required. However, with some restrictions on how many of each length are to be used, the pattern shown in Figure 3 can be used. Segments of length two must come in pairs; segments of length three must come in triplets, and so on.

UTFPGA1 only has two tiles, so getting power to the tiles is simple. For larger arrays careful thought must be given to power distribution. In our next version we plan to use a grid style of distribution, with power lines running in the routing channels so that they will butt to the neighbouring tiles, and for easy access within the tiles.

Figure 9 shows a plot of one tile and Figure 10 gives the floorplan. It can be seen that the physical layout closely approximates the logical layout shown in Figure 2. This was certainly the easiest way

Figure 9 Layout of a tile.

to do the design, but there is much room for improvement. About 15% of the area is taken up by the routing channels and about 15% is for the wiring channels in the lookup table. This means that about 30% of this tile is wiring with no active devices below it. Significant area can be saved if more of the active devices can be placed under the routing and wiring channels. About 35% of the tile area is used by the memory cells, so compacting these cells can also improve density.

Power-on and Programming Safety

When power is first applied to the circuit, and during programming, great care must be taken to prevent a random configuration that results in drivers turning on and fighting each other. It is also important that the pads are not configured in a way that they conflict with the external circuitry. In UTFPGA1 we tristate the pads, disable the drive of the I/O blocks onto the routing tracks, and disable the output of the logic blocks. An external pin is provided to do this, and this must be asserted during power-on and programming.

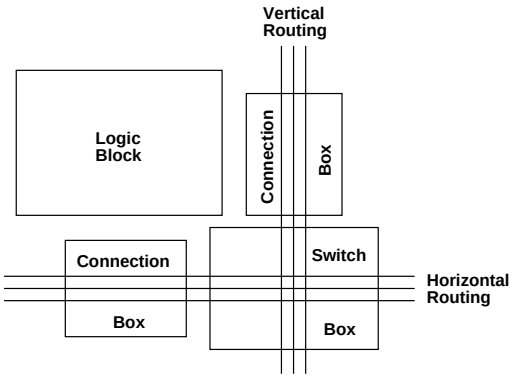

Figure 10 Floorplan of a tile.

Clock Distribution

A very important issue is clock distribution. The most general scheme would be to have the clock line be like any other signal, but this would cause severe problems with delay and skew. A better scheme would be to provide a direct connection into each logic block. UTFPGA1 uses a simple scheme with one clock line that originates from an input pad and is connected to all of the clock inputs of the D flip flops in the logic blocks. This allows us to test simple synchronous designs, but it does not allow circuits that might require qualified clock signals. More work needs to be done in this area.

DESIGN STATISTICS

UTFPGA1 was designed as a graduate course project in VLSI design, and submitted for fabrication in July of 1991. Five students worked on the design over a period of 15 weeks spending a total of 2000 hours. This time included several weeks of design study before any circuit design was initiated. The design contains only two tiles so that the basic functionality can be tested before attempting a more ambitious array of cells.

With the limited time available, very little effort was spent on optimizing the design for speed. However, recent simulations have shown promising results, and we expect to achieve significant improvements in the next version with better transistor sizing and circuit design. Table 1 gives the delays through various parts of the chip for a path from the input to a logic block, out through a connection box, through one switch box, into the connection box of the next tile, and to the input of a logic block. The two values depend on whether highs or lows are driven. The slower delays are due to highs being degraded as they are driven through n-channel switches or multiplexer networks. The I/O mux delay is the delay from the pad to a routing channel loaded with only one tile (no fanout to other tiles).

UTFPGA1 has 26 pads, six of which are I/O blocks. The remainder are used for power, programming, and other control functions. Other statistics for UTFPGA1 are shown in Table 2.

Since the chip was submitted for fabrication, one logic bug has been discovered. When the drivers are disabled during power-on and programming, one of the drivers in the I/O muxes continues to drive one of the routing segments. This will cause some unnecessary power dissipation,

Table 1 Circuit delays measured from simulation.

	Delay
Logic block	1.9 to 4.7 ns
Connection box output	0.9 ns
Routing delay through 1 switch box	0.2 ns
Connection box input	1.6 to 3.4 ns
Total path delay	6.4 to 7.4 ns
I/O mux	1.3 to 1.7 ns

Table 2 Design statistics.

Programming bits in	Connection box (C)	28
	Switch box (S)	42
	Logic block (L)	48
	I/O block (I)	4
	Tile (T=L+S+2C)	146
	UTFPGA1 (2T+6I)	316
Dimension of	Programming bit	33x50 μm
	Logic block	540x400 μm
	Switch box	500x400 μm
	Connection box	400x360 μm
	I/O block	600x400 μm
	Tile	900x800 μm
	UTFPGA1	3170 x 2340 μm

but we do not anticipate that any damage will result.

CONCLUSIONS AND FUTURE WORK

The design of UTFPGA1 was an exercise to understand the issues in designing a real FPGA, and in particular, the topology problems when doing the actual layouts. With this experience we are confident that we can design a faster, more dense circuit next time.

Although we did not have the time to focus on the speed aspects of the circuit design, we did do some experiments with transistor sizing for the n-channel pass transistors used in the switches and the multiplexers. Our initial intuition was to use minimum size devices ($\frac{W}{L} = \frac{3.2}{1.2}$) to keep the area small and the capacitance on the routing tracks low. We found that a larger transistor ($\frac{W}{L} = \frac{12.8}{1.2}$) gave much better delay characteristics. With the smaller transistors, the high resistance dominates the capacitance effects.

We have found that significant delay is encountered due to the degraded highs and the resistance through the multiplexer inputs of the connection blocks and the multiplexers in the logic blocks. For increased speed, it may be better to use switches at the inputs to the connection blocks, even though this will use more area. The degraded highs also have implications on noise margins and unbalanced switching times. These effects can be lessened by better design of the circuits used to sense the degraded levels.

The plot of UTFPGA1 shows clearly that a lot of active device area is being wasted beneath the routing channels and the wiring channels in the logic blocks. More thought about layout topology should be able to reduce significantly the area required to implement this circuit.

Important issues that have not been considered sufficiently in this project are the clock distribution network, the connectivity in the logic block to the D flip flop and its clocks to the combinational logic, the number of D flip flops required, power distribution, and I/O block design.

Work has already begun on the next design, and we will be focusing on higher speed circuits and more area-efficient layout. We are also investigating other logic block architectures to reduce overall circuit delay and different switch box topologies that require fewer transistors but allow the same routability.

ACKNOWLEDGEMENTS

We gratefully acknowledge the help of Jaro Pristupa of the VLSI Research Group for the work he does to support our CAD environment, Ben Tseng for working on a CAD tool to help program the chip, and the Canadian Microelectronics Corporation for providing fabrication support.

This work is supported by a grant from Micronet, a Network of Centres of Excellence in Canada, and by NSERC Operating Grants URF0043298, A4029, and OGP0036648.

REFERENCES

Ahrens, M., Gamal, A. E., Galbraith, D., Greene, J., Kaptanoglu, S., K.R. Dharmarajan, L. H., Ku, S., McGibney, P., McGowan, J., Samie, A., Shaw, K., Stiawalt, N., Whitney, T., Wong, T., Wong, W., and Wu, B. (1990). An FPGA Family Optimized for High Densities and Reduced Routing Delay. In *Custom Integrated Circuits Conference*, pp. 31.5.1–31.5.4. IEEE.

Algotronix (1989). *CAL 1024 Datasheet*. Algotronix Ltd., Edinburgh, Scotland.

AMD (1990). *Mach Devices High Density EE Programmable Logic Data Book*. AMD, 901 Thompson Place, P.O. Box 3453, Sunnyvale, CA 94088-3453.

Canadian Microelectronics Corporation (1990). The CMOS4S Standard Cell Library. Report ICI-021R0, VLSI Implementation Centre, Caruthers Hall, Queen's University, Kingston, Ontario, Canada K7L 3N6.

Carter, W. S., Duong, K., Freeman, R. H., Hsieh, H.-C., Ja, J. Y., Mahoney, J. E., Ngo, L. T., and Sze, S. L. (1986). A User Programmable Reconfigurable Logic Array. In *Custom Integrated Circuits Conference*, pp. 233–235. IEEE.

Chow, P., editor (1989). *The MIPS-X RISC Microprocessor*. Kluwer Academic Publishers.

Concurrent (1991). *CFA6006 Field-Programmable Gate Array Data Sheet*. Concurrent Logic, Sunnyvale, CA.

Gamal, A. E., Greene, J., Reyneri, J., Rogoyski, E., El-ayat, K. A., and Mohsen, A. (1989). An Architecture for Electrically Configurable Gate Arrays. *IEEE Journal of Solid-State Circuits*, 24(2):394–398.

Hsieh, H.-C., Carter, W. S., Ja, J., Cheung, E., Schreifels, S., Erickson, C., Freidin, P., and Tinkey, L. (1990). Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays. In *Custom Integrated Circuits Conference*, pp. 31.2.1–31.2.7. IEEE.

Muroga, H., Murata, H., Saeki, Y., Hibi, T., Ohashi, Y., and Noguchi, T. (1991). A Large Scale FPGA with 10K Core Cells with CMOS 0.8μm 3-Layered Metal Process. In *Custom Integrated Circuits Conference*, pp. 6.4.1–6.4.4. IEEE.

Plessey (1989). *ERA60100 Preliminary Datasheet*. Plessey Semiconductor, Swindon, England.

Plus (1989). *FPGA2040 Field-Programmable Gate Array Data Sheet*. Plus Logic, San Jose, CA.

Rose, J. and Brown, S. (1991). Flexibility of Interconnection Structures for Field-Programmable Gate Arrays. *IEEE Journal of Solid-State Circuits*, 26(3):277–282.

Rose, J., Francis, R. J., Lewis, D., and Chow, P. (1990). Architecture of Field Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency. *IEEE Journal of Solid-State Circuits*, 25(5):1217–1225.

Singh, S., Rose, J., Lewis, D., Chung, K., and Chow, P. (1991). Optimization of Field-Programmable Gate Array Logic Block Architecture for Speed. In *Custom Integrated Circuits Conference*, pp. 6.1.1–6.1.6. IEEE.

Wong, S. C., So, H. C., Ou, J. H., and Costello, J. (1989). A 5000-Gate CMOS EPLD with Multiple Logic and Interconnect Arrays. In *Custom Integrated Circuits Conference*, pp. 5.8.1–5.8.4. IEEE.

Xilinx (1989). Presentation by Xilinx at the ITRC Field Programmable Gate Arrays Symposium, University of Toronto.

Xilinx (1991). *The Programmable Gate Array Data Book*. Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124.