Optimization of Field-Programmable Gate Array Logic Block Architecture for Speed

Satwant Singh, Jonathan Rose, David Lewis, Kevin Chung, Paul Chow
Dept. Electrical Engineering, University of Toronto, Ontario, Canada M5S 1A4

Abstract

This paper explores the effect of the choice of logic block on the speed of a Field-Programmable Gate Array (FPGA). A set of logic circuits are implemented as FPGAs each using a different logic block, and the speed of the implementation is measured. While the result depends on the delay of programmable routing, experiments indicate that wide input PLA-style AND-OR gates, four and five-input lookup tables and certain multiplexor configurations produce the lowest total delay over the important values of routing delay. Furthermore, significant gains in performance (from 10% to 41% reduction in total delay) can be achieved by connecting a small number of logic blocks together using hard-wired connections.

1 Introduction

The Field-Programmable Gate Array (FPGA) is a new ASIC medium that provides instant manufacturing turnaround and extremely low manufacturing costs. An FPGA can be designed like a Gate Array but is user-programmed like a PLD. However, an FPGA has both lower logic density and performance than a Gate Array that is made in the same process technology. These deficiencies can be addressed by improving the *architecture* of the FPGA. The architecture of an FPGA consists of its logic block function, interconnection structure, and I/O block design. In previous work, we have investigated the effect of logic block functionality on the area of FPGAs [Rose90a], and the effect of switching block flexibility on routability [Rose90b]. In this paper we look at ways to improve the *speed* of the FPGA by architectural changes to the logic block.

The FPGA was introduced in [Cart86] and newer versions have been presented in [Hsie90] [Ahre90] and [Wong89]. An FPGA consists of an array of logic blocks surrounded by a programmable interconnection structure. There are many different kinds of interconnection structures, such as those articulated in the commercial architectures and [Rose90b]. It is universally true, however, that the delay of the routing is significantly greater than that of a simple metal wire in the same process technology because programmable interconnects contain significant resistance and capacitance. Connection delays often exceed the delay of the logic block, and this is one of the fundamental limitations on FPGA speed.

The performance of an FPGA can be increased by reducing the number of stages of programmable routing used in the critical paths. One way to do this is to choose logic blocks with high functionality so that the number of logic block levels in the critical path is minimized, as illustrated in Figure 1. Figure 1a gives the implementation of the logic function $f=ab\bar{d} + abc + ac\bar{d}$ using a 2-input nand gate as the logic block. It requires four levels of the logic block in the critical path. Figure 1b shows an implementation of the same function using three-input lookup tables, which require only two levels. Since the latter avoids two levels of slow programmable interconnect, this will likely lead to a significant decrease in delay. Increasing the functionality of the logic block, however, is likely to increase its combinational delay. The increase is only profitable if the reduction in routing delay more than offsets the increase in total delay due to the logic block.

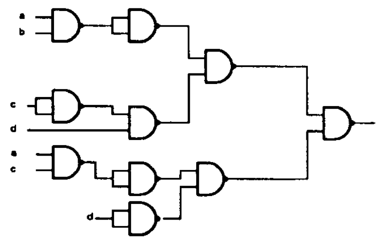
A second way to reduce routing delay is to connect basic logic blocks with hard-wired non-programmable connections (a wire with near-zero delay) so that programmable interconnect is avoided. The questions addressed in this paper are:

1. What is a good choice for a logic block that minimizes the total delay in an FPGA? Experiments indicate that wide input PLA-style AND-OR gates, four and five-input lookup tables and the Actel [ElGa89] logic block are roughly equivalent in terms of total delay over a set of circuits using each of those blocks.

2. Which hard-wired interconnection topology of four-input lookup tables minimizes the total delay in an FPGA? Experiments indicate that a major gain can be had with a small number of blocks hard-wired in two-level and three-level configurations - from 10% to 41% lower delay, depending on the routing delay, as compared to the case where no hard-wired connections are used.

This paper is concerned only with the the speed of the FPGA and so we ignore issues that affect the logic density.

This work was supported by NSERC Operating Grants #URF0043298, #A4029, #OGP0036648, a MICRONET research grant, a research grant from Bell-Northern Research, and ITRC.


a) Logic Block = 2-Input NAND Gate, Number of Logic Blocks in Critical Path = 4

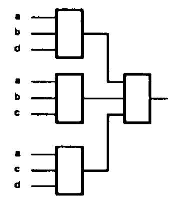

b) Logic Block = 3-Input LOOKUP TABLE, Number of Logic Blocks in Critical Path = 2

Figure 1 - *Implementation of* $f = ab\bar{d} + abc + ac\bar{d}$

2 Experimental Choices, Process and Model

To answer the above questions, our approach is to implement a set of circuits in many FPGAs each of which uses a different logic block, and then measure the resulting delay of the circuit. The following section discusses the selection of logic blocks. The implementation procedure, described in Section 2.2, takes a set of benchmark circuits and applies logic synthesis techniques to produce a network of interconnected logic blocks. The delay measuring process requires the determination of the critical path length in each FPGA implementation and a model for calculating the total delay given the logic block delay and the routing delay. This process is discussed in Section 2.3.

2.1 Logic Block Selection

Thirteen different logic blocks of varying functionality were selected for comparison, and are named and described in Table 1. These blocks were chosen to represent a wide cross-section of the possible blocks:

- The nand2pi and nand3pi are simple nand gates that have a *programmable invert* capability, which allows the inputs to be passed in true or complement form.
- The mux21 and mux41 logic blocks can implement all possible logic functions of a multiplexor using the primary inputs and the selector inputs.
- The actel logic block is the one used by Actel Corp in their ACT-1 series of FPGAs [ElGa89].
- The K2, K3, K4 and K5 logic blocks are lookup tables with 2 to 5 inputs. They are implemented as one large multiplexor with the appropriate number of inputs.
- The A2O2pi gate implements the logic function $f = AB + CD$ where A,B,C and D are the inputs. It has the programmable invert capability.
- The A8O3pi, A16O3pi, and A32O5pi are wide AND-OR gates. The gate A8O3pi has a total of eight inputs, each of which can be selected to form three separate product terms which are OR'd together. The gate A16O3pi has 16 inputs and three product terms, and the gate A32O5pi has 32 inputs and five product terms. These three gates have the programmable invert capability.

Block Name	Logic Function	Delay (ns) 1.2 μm CMOS
nand2pi	2-input nand with prog inv	1.26
nand3pi	3-input nand with prog inv	1.42
mux21	2 to 1 mux	1.08
mux41	4 to 1 mux	1.31
actel	Actel logic block [ElGa89]	1.31
K2	2-input 1-output lookup table	1.39
K3	3-input 1-output lookup table	1.44
K4	4-input 1-output lookup table	1.71
K5	5-input 1-output lookup table	2.03
A2O2pi	f = AB + CD with prog inv	1.58
A8O3pi	3-wide OR of 8-wide AND	2.69
A16O3pi	3-wide OR of 16-wide AND	3.77
A32O5pi	5-wide OR of 32-wide AND	5.95

Table 1 - *Logic Block Selection and Delay*

Table 1 also gives the worst-case delay of each logic block determined using the Spice 2G6 circuit simulator, in a 1.2µm CMOS process. The simulation includes a small buffer following the logic function, but no loading or delay due to routing.

2.2 Logic Synthesis Procedure

Each test circuit is converted into a network of logic blocks using the procedure below. This procedure deals only with combinational circuits.

1. Collapse the logic circuit into a two-level representation, using the misII [Bray87] *collapse* command so that each output is only a function of its primary inputs. Optimize the two-level expression using Espresso [Bray84].

2. Factor each output separately into a multi-level logic expression using the misII *decompose* command [Bray87]. Remove any fanout created by the decompose by re-substituting logic that has fanout back into the original logic expression. This step is necessary because the technology mapping in step 3 does a poor job across fanout. By removing fanout the delay is reduced but the area is increased. As such the results presented below are optimistic. Note also that without fanout some circuits, such as a parity tree, have a size that is exponential in the number of inputs. It is not possible to run this class of circuits in these experiments.

3. Perform the technology mapping of the boolean network. This converts the logic into a network of logic blocks. This was done two ways, depending on the logic block:

   i. For all of the logic blocks *except* A8O3pi, A16O3pi and A32O5pi, technology mapping was done using the MIS II technology mapping program [Detj87]. The mapper is set to optimize the critical path delay. This mapper was used for all the logic blocks, (even though superior mappers were available for some of the blocks) so as to create a "level" playing field.

   ii. For logic blocks A8O3pi, A16O3pi and A32O5pi, the MIS II mapper couldn't be used because these gates are too big for it to handle. For these blocks, step 2 was omitted. The two-level expressions are mapped into a minimum-depth tree of gates in the following way: Consider the general case where the logic block has p inputs to the AND gates and can OR together s product terms. For these three gates p is one of 8, 16 or 32, and s is either 3 or 5. If the logic expression to be mapped has a product term with v variables in it then the number of levels in the logic tree to implement the product term is $\lceil \log_p v \rceil$. Similarly if the number of product terms in the expression is t then the depth of the OR tree would be $\lceil \log_s t \rceil$. When the two trees are combined, there is a saving of one logic level and so the total depth in logic blocks is $\lceil \log_p v \rceil + \lceil \log_s t \rceil - 1$.

2.3 Model for Measuring Delay

The speed of a circuit implemented in an FPGA with a given logic block is a function of the delay of the logic block (D_{LB}), the number of logic blocks in the critical path (N_L), and the delay incurred in the routing between each logic block, (D_R). Assuming that each stage of logic block incurs one routing delay and one logic block delay, then the total delay (D_{TOT}) can be calculated as:

$$D_{TOT} = N_L \times (D_{LB} + D_R) \qquad (1)$$

The value of N_L can be measured for each circuit after it is mapped into a logic block using the procedure described above. The value of D_{LB} was determined as described in Section 2.1.

The value of D_R is much more difficult to determine. It is a function of the routing architecture, the fanout of a connection, the length of the connection, the process technology and the programming technology. Since our purpose is to understand general architectural principles, it is important not to fix any of these parameters. As such, most of the results below will be given as a function of D_R, rather than choosing a specific D_R.

3 Experimental Results

The experimental circuits that were used are a selection of 15 logic synthesis benchmarks provided by the Microelectronics Center of North Carolina (MCNC) and one standard cell-based circuit from Bell-Northern Research. They range in size from 28 to over 700 two-input nand gate equivalents. Each circuit was passed through the implementation procedure described in Section 2.2 once for every logic block listed in Table 1.

Table 2 gives the average and standard deviation of the number of logic blocks in the critical path (N_L) over all 16 circuits for each logic block, and the calculation of D_{TOT} from equation (1). The D_{TOT} calculation uses N_L from Table 2, D_{LB} from Table 1 and values of 0, 2, 4 and 10ns for D_R. These values of D_R were chosen because they range from the minimum possible to the point where the rankings of the logic blocks based on D_{TOT} don't change. Figure 2 is a plot of the same data: D_{TOT} versus D_R for all the different logic blocks.

A number of conclusions arise from this data. If the routing is very fast, such as $D_R = 0$, the data from Table 2 indicate that the Actel logic block is the fastest, followed closely by the three- and four-input lookup tables. This can be explained by the fact that the Actel block has slightly more logic levels than the others, but its delay is significantly less, and so with no penalty

Logic Block	$\overline{N_L}$	St. Dv.	$D_{TOT} = N_L \times (D_{LB} + D_R)$			
			D_R=0 (ns)	D_R=2 (ns)	D_R=4 (ns)	D_R=10 (ns)
mux21	11.4	4.6	12.3	35.1	57.9	126.3
K2	10.8	4.5	15.0	36.6	58.2	123.0
nand2pi	10.8	4.5	13.6	35.2	56.8	121.6
nand3pi	9.3	3.8	13.2	31.8	50.4	106.2
mux41	7.3	2.2	9.6	24.2	38.8	82.6
A2O2pi	6.9	2.6	10.9	24.7	38.5	79.9
K3	6.2	2.4	8.9	21.3	33.7	70.9
actel	6.2	1.8	8.1	20.5	32.9	70.1
K4	5.2	1.8	8.9	19.3	29.7	60.9
K5	4.8	1.9	9.7	19.3	28.9	57.7
A32O5pi	3.3	1.3	19.6	26.2	32.8	52.6
A8O3pi	4.0	1.6	10.8	18.8	26.8	50.8
A16O3pi	3.6	1.5	13.6	20.8	28.0	49.6

Table 2 - *Average Critical Path Length and Total Delay*

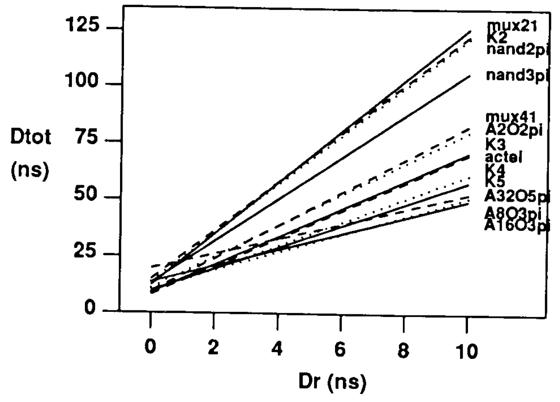


Figure 2 - *D_{TOT} vs D_R for all Blocks*

for routing its total delay is the lowest.

As the routing delays increase (D_R = 2ns and 4ns), the fastest block becomes A8O3pi followed closely by the four- and five-input lookup tables. The Actel logic block and the three-input lookup table are also close to the minimum. Increasing the routing delays reduces the advantage of the faster logic blocks so that the total number of logic block levels becomes the more important factor. Within the accuracy of these experiments the delay for the K3, K4, K5, Actel, A8O3pi, A16O3pi and A32O5pi gates are roughly equivalent.

When the routing delay is very large (D_R = 10ns), the wide AND-OR gates with programmable inversion (A8O3pi, A16O3pi, A32O5pi) dominate the field. The four- and five-input lookup tables are roughly 20% slower, and the Actel and three-input lookup table blocks are about 40% slower. At this point the small N_L values of the wide AND-OR gates dominates all the other effects, and these logic blocks win out.

It should be noted that these results depend heavily on the quality of the logic synthesis tools. As indicated in the following section, a technology mapper tuned for lookup tables [Fran91] produces significantly better values of N_L for the lookup tables. If that mapper was used in these experiments, the lookup tables would have better D_{TOT} across all values of D_R. It is possible that a mapper tuned specifically for the other logic blocks would greatly improve their D_{TOT} as well.

4 Increasing Speed with Hard-Wired Connections

Even using the A8O3pi gate or four- and five-input lookup tables as logic blocks, FPGAs cannot achieve speeds comparable to a gate array in the same process technology. This is due to the large propagation delay of each routing stage. An architectural technique to reduce the number of general purpose routing stages is to create hard-wired (non-programmable) connections between two or more basic logic blocks. In this section we answer the question posed in the introduction: what topology of hard-wired connections will result in the lowest total delay?

For this experiment, we focused on the four-input lookup table since it achieved close to the minimum total delay for values of D_R ranging from 0ns to 4ns, and also because it has a small number of inputs. Figure 3 illustrates the eight different configurations of four-input lookup tables that were investigated, and gives the associated name. Each block is named in the form Lx.y. The x value gives the number of levels in the tree, and y gives the number of lookup tables.

A two-level hard-wired structure that forms a complete tree (block L2.5 in Figure 3) will reduce the number of programmable interconnect levels by up to a factor of two, since a fanout-free network will trivially map onto these blocks. The important question, then, is how close will a smaller group of blocks (such as L2.2 or L2.3) come to reducing the number of programmable interconnects (N_R) by two. Note that if the

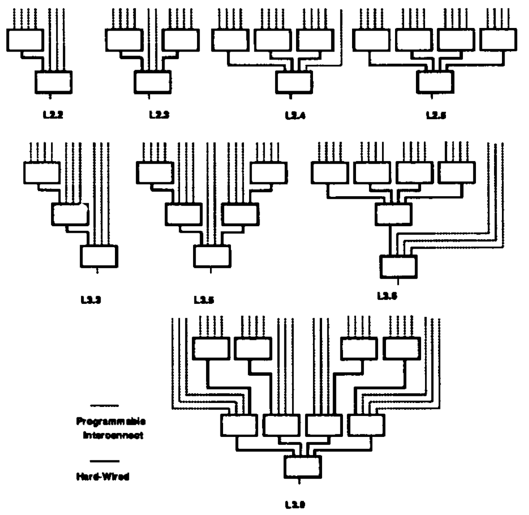


Figure 3 - *Hierarchical Logic Block Selection*

number of programmable connections in the original circuit is three, then this can only be reduced to two. A similar argument can be made about three, four and five level hard-wired logic blocks: they can at most reduce the number of programmable connections by three, four or five. The maximum reduction in average N_R over all 16 circuits can be calculated by dividing N_L in each 4-input lookup table circuit by the number of levels in the logic block, and using a ceiling function for the division. For the 16 test circuits the minimum possible average N_R for the two, three and four level blocks is 2.25, 1.63 and 1.44 respectively.

4.1 Hierarchical Logic Synthesis Procedure

The logic synthesis procedure described in Section 2.2 above was modified to use the Chortle-d technology mapping program [Fran91] for lookup tables instead of the MIS II mapping program, because Chortle-d is tuned to optimized delay for lookup tables. Note that the average depth is significantly smaller using Chortle-d.

To map into the hierarchical logic blocks shown in Figure 3, an additional mapping step is needed. A new mapping tool, called *Tempt* [Chun91] was developed. The input to Tempt is a network of lookup tables and the output is a network of hierarchical logic blocks optimized for delay. The resulting mapping is simply a set of groupings of logic blocks that fit into the topology of the selected hierarchical logic block; it does not change the function within the lookup table.

Note also that degenerate cases of the larger blocks are also used. For example, the L2.2 structure appears within the L3.3 structure, and we assume that a "tap-off" from the second logic block in L3.3 will provide the output of that logic block to the external routing with no delay penalty.

4.2 Hierarchical Delay Model

The delay model described in Section 2.3 must be modified to take into account the fact that the hard-wired (internal) connections in Figure 3 have effectively zero delay. Thus the total delay becomes a function of the same factors in equation (1) (N_L, D_{LB}, and D_R) as well as an additional variable: N_R, the number of programmable interconnect connections in the critical path. Since only those connections incur the D_R delay, the total delay is now given by:

$$D_{TOT} = N_L \times D_{LB} + N_R \times D_R \qquad (2)$$

5 Results for Hard-Wired Blocks

Table 3 gives the measured average values of N_L and N_R for each of the hierarchical blocks in Figure 3, and includes the results for a single four-input lookup table. These values are averaged over all 16 circuits. As mentioned in Section 4.1, the N_L value for the four-input lookup table smaller than in Table 2 because the Chortle-d technology mapper is tuned for delay optimization of lookup tables [Fran91].

As mentioned in Section 4, the two-level blocks can achieve a theoretical minimum average N_R of 2.25, and this is achieved by the complete two-level tree, L2.5, in the experiments. The two-level blocks L2.2, L2.3, L2.4 and L2.5 attain progressively lower average N_R values, from 3.38 to 2.25. The most significant gains, per added block, are achieved by L2.2 and L2.3.

Of all three-level blocks that were used (including many not illustrated in Figure 3) none obtained an N_R lower than 2.13. The 2.13 value was achieved by the nine-block L3.9. In all our experiments with two, three, four and five level blocks, we observed that the hierarchical blocks with symmetry attained the best performance. This is likely due to the Chortle-d Technology mapping program [Fran91], which typically produces very balanced trees of logic, in the effort to minimize delay.

Table 3 also gives the total delay for FPGAs using the hierarchical logic blocks for D_R = 0,2,4 and 10 nanoseconds. For D_R = 0 all delays are the same, since the programmable interconnect is the same speed as the hard-wired interconnect. For D_R = 2 the two-level blocks achieve a decrease in delay

Logic Block	$\overline{N_L}$	N_R		$D_{TOT} = N_L \times D_{LB} + N_R \times D_R$			
		Avg	St. Dv	D_R=0 (ns)	D_R=2 (ns)	D_R=4 (ns)	D_R=10 (ns)
K4	4.1	4.13	2.1	7.1	15.3	23.6	48.4
L2.2	4.1	3.38	1.2	7.1	13.8	20.6	40.9
L2.3	4.1	2.56	1.1	7.1	12.2	17.3	32.7
L2.4	4.1	2.44	1.0	7.1	11.9	16.8	31.5
L2.5	4.1	2.25	0.8	7.1	11.6	16.1	29.6
L3.3	4.1	3.31	1.1	7.1	13.7	20.3	40.2
L3.5	4.1	2.25	0.9	7.1	11.6	16.1	29.6
L3.6	4.1	2.19	0.8	7.1	11.4	15.8	29.0
L3.9	4.1	2.13	0.7	7.1	11.3	15.6	28.4

Table 3 - *Programmable Connection Length and Total Delay*

from 10% (L2.2) to 24% (L2.5) relative to the single four-input lookup table (K4). The three-level blocks decrease the delay from 10% to 26%. Experiments with higher level blocks did not achieve any further reduction - those experiments were limited to use fewer than 10 four-input lookup tables in the hierarchical logic block. For D_R = 4 the two-level blocks achieve a decrease in delay from 13% to 32%. The three-level blocks decrease the delay from 14% to 34%. For D_R = 10 the two-level blocks achieve a decrease in delay from 16% to 39%. The three-level blocks decrease the delay from 17% to 41%.

6 Conclusions and Future Work

This paper has explored the relationship between logic block architecture and the speed of the resulting FPGA. The principal conclusions are, first that Wide-input PLA-style AND-OR gates, four and five-input lookup tables and the the Actel [ElGa89] logic block are all good choices for a logic block for mid-range values of programmable routing delay. These results depend heavily on the quality of the logic synthesis tools, particularly how well the tools optimize delay. Second, significant gains in performance can be had by hard-wiring several logic blocks together, with reductions in delay ranging from 10% to 41% depending on the hard-wired configuration and the routing delay. Most of these gains in performance can be made with only a few logic blocks hard-wired together. Configurations that are symmetric are superior to asymmetric configurations.

In the future, we will adapt these experiments and tools to account for the area cost of the gain in delay.

7 References

[Ahre90]
M. Ahrens, et. al, "An FPGA Family Optimized for High Densities and Reduced Routing Delay," Proc. 1990 CICC, May 1990, pp. 31.5.1 - 31.5.4.

[Bray84]
R.K. Brayton et. al, **Logic Minimization Algorithms for VLSI Synthesis**, Kluwer Academic Publishers, 1984.

[Bray87]
R.K. Brayton et. al, "MIS: A Multiple-Level Logic Optimization System," IEEE Trans. on CAD, Vol CAD-6, No. 6, Nov 1987, pp. 1062-1081.

[Cart86]
W. Carter et. al, "A User Programmable Reconfigurable Gate Array," Proc. 1986 CICC, May 1986, pp. 233-235.

[Chun91]
K. Chung, "Mapping Hierarchical Logic Blocks," Ph.D. dissertation, in progress.

[Detj87]
E. Detjens et. al, "Technology Mapping in MIS", Proc. ICCAD 87, Nov 1987, pp. 116-119.

[ElGa89]
A. El Gamal, et. al, "An Architecture for Electrically Configurable Gate Arrays," IEEE JSSC Vol. 24, No. 2, April 1989, pp. 394-398.

[Fran91]
RJ Francis, J. Rose, Z. Vranesic "Technology Mapping for Delay Optimization of Lookup Table-Based FPGAs," submitted to MCNC Intl Workshop on Logic Synthesis, 1991.

[Hsie90]
H. Hsieh, et. al "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays," Proc. 1990 CICC, May 1990, pp. 31.2.1 - 31.2.7.

[Rose90a]
J.S. 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 JSSC, Vol. 25 No. 5, October 1990, pp. 1217-1225.

[Rose90b]
J. Rose, S. Brown, "The Effect of Switch Box Flexibility on Routability of Field Programmable Gate Arrays," Proc. 1990 CICC, May 1990, pp. 27.5.1 - 27.5.4.

[Wong89]
S.C. Wong, H.C. So, J.H. Ou, J. Costello, "A 5000-Gate CMOS EPLD with Multiple Logic and Interconnect Arrays," Proc. 1989 CICC, May 1989, pp. 5.8.1 - 5.8.4.