Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency

JONATHAN ROSE, MEMBER, IEEE, ROBERT J. FRANCIS, DAVID LEWIS, MEMBER, IEEE, AND PAUL CHOW, MEMBER, IEEE

Abstract—**This paper examines the relationship between the functionality of a field-programmable gate array (FPGA) logic block and the area required to implement digital circuits using that logic block. This investigation is done experimentally by implementing a set of industrial circuits as FPGA's using CAD tools for technology mapping, placement, and routing. Using a simple model of the interconnection and logic block area, a range of programming technologies (the method of FPGA customization) is explored. The experiments are based on logic blocks that use lookup tables for implementing combinational logic. Results indicate that the best number of inputs to use (a measure of the block's functionality) is between three and four, and that a *D* flip-flop should be included in the logic block. These results are largely independent of the programming technology.**

More generally, it was observed that the area efficiency of a logic block depends not only on its functionality but on the average number of pins connected per logic block. It is shown that as the number of connected pins per block increases, the number of wiring tracks required to route those blocks also increases. Since adding functionality to a block will lead to an increase in the number of connected pins, it follows that an increase in functionality of the block is only beneficial if the total number of blocks is reduced to more than compensate for the increased wiring area. This notion leads to the conclusion that the most area-efficient logic blocks are those with a high amount of functionality per pin.

I. Introduction

THE field-programmable gate array (FPGA) is a revolutionary idea in semicustom integrated circuits that reduces the IC manufacturing time from months to minutes and prototype cost more than three decades. The FPGA was introduced in [1] and newer versions have been presented in [2]–[9]. It is similar to a gate array in structure, but can be field-programmed to specify the function of the logic blocks and their interconnection. As a result of the programmability, the architecture of an FPGA is more complex than that of a conventional gate array.

Manuscript received September 18, 1989; revised January 18, 1990. This work was supported by NSERC Operating Grants URF0043298, A4029, and OGP0036648, a research grant from Bell-Northern Research, and DARPA Contract N00014-87-K-0828.

The authors are with the Department of Electrical Engineering, University of Toronto, Toronto, Ont., Canada M5S 1A4.

IEEE Log Number 9037794.

In this paper we focus on the logic block architecture, and study the effect of logic block functionality on FPGA area. We ignore speed considerations, even though they are very important, because we need first to determine the plausible architectures from an area perspective. In later studies we can use this information to know the cost, in area, of obtaining an FPGA with a particular performance. An associated study investigates the effect of routing structure flexibility on routability of FPGA's [10].

The logic block functionality is an important factor in the FPGA architecture. It can be loosely defined as the number of logic blocks required to implement an (unspecified) set of circuits. A precise and usable definition of functionality remains an open question. If the block has insufficient functionality then too much area must be devoted to the interconnection. If the block has excess functionality then it may suffer from underutilization and wasted active area.

In this paper we focus on a simple logic block architecture that contains a K-input lookup table to implement combinational logic and a D flip-flop. We address two questions concerning this block with the following results:

1) What is the best number of inputs, K, to use for the combinational function? Note that K is direct measure of functionality. Our results show that the best number of inputs is consistently between three and four, and is almost the same whether or not the block contains a D flip-flop.
2) Should the logic block contain a D flip-flop? Experiments indicate that the presence of a D flip-flop in the logic block always reduces chip area.

The explanation of these results lead to an important insight into the functionality–area trade-off of FPGA's. First, we find that interconnection area completely dominates active area so that the trade-off between routing area and logic block functionality is the key one. Second, as intuition suggests, when the functionality of the logic block increases, the total number of logic blocks required to implement a circuit decreases. However, when functionality increases, so does the number of pins on the

0018-9200/90/1000-1217$01.00

logic block. We have found that the total routing area is a direct function of the number of connected pins on the logic block: as the number of pins increases, the routing area increases significantly. Therefore, a beneficial increase in functionality of the logic must reduce the total number of blocks to more than compensate for the increased routing area. This point further implies that a good choice for an area-efficient block is one that has high functionality per connected pin.

The architectural choices that affect the area of an FPGA depend on the programming technology, which is the underlying method by which logic functions are configured and routing connections are made. For example, the programming technology used in [3] creates logic functions using static RAM lookup tables, and performs routing using pass transistors and multiplexors. The FPGA described in [4] uses an antifuse for both logic configuration and interconnection that, when blown, causes two metal tracks to be electrically joined. The FPGA described in [7] uses an EPROM programming technology. The experiments described in this paper account for the area requirements of differing programming technologies.

Previous work on FPGA's has taken the form of descriptions of a specific architecture and implementation. The first device that can be considered a FPGA was introduced in 1986 [1], and subsequent devices are described in [2] and [3]. The programming technology is based on static RAM bits that are loaded into the chip when the system is turned on. The most recent logic block architecture consists of a combinational block followed by two resettable D flip-flops. The combinational block is a static RAM lookup table that can be configured to realize any five-input one-output logic function, or any two four-input one-output logic functions in which the inputs must be selected from the same set of signals.

The global routing architecture is similar to a channeled gate array. Connections are made from the logic block to the channel via multiplexors controlled by static RAM. At the intersection of horizontal and vertical channels (the switchbox), connections are made by transistors that are turned on or off using static RAM bits. There are dedicated local interconnects between neighboring blocks that are not switched, as well as "long" lines that traverse entire channels. Dedicated lines for global clock distribution are also present.

An FPGA based on antifuse programming technology [11] was introduced in [4]–[6]. The logic block consists of three multiplexors and a logic OR gate, which can be connected in various ways using the antifuse to perform a wide range of combinational and sequential logic functions. The interconnection architecture consists of horizontal channels with staggered segments of wires that can be joined by antifuses, and vertical connections across the logic blocks. A global clock distribution scheme is also included.

A third FPGA architecture, based on an EPROM programming technology, was introduced in [7]. The logic block architecture is very similar to that in a single PLD: two-level AND–OR logic followed by a single D flip-flop. An additional level of logic is provided by an expander product-term array associated with a group of logic blocks. Inputs are wired-OR selected from a bus using programmed transistors. The interconnection scheme is a two-level hierarchy. A bus structure connects logic blocks within one level of the hierarchy and between groups of logic blocks at the second level of the hierarchy.

Two more FPGA architectures were recently introduced. The first, based on a static RAM programming technology [8], has a logic block consisting of a two-input NAND gate, a multiplexor, and a latch. Routing is performed using the same logic element so that routing and logic are directly traded off. The second recent FPGA [9] uses an EPROM programming technology and a logic block based on AND–OR gates similar to a PLD. Interconnection is done with a two-level hierarchy of buses.

Gray and Kean have also proposed an FPGA-like structure [12]. It uses a static RAM programming technology and a logic block based on interconnections of five multiplexors. The interconnection scheme is two connections to each nearest neighbor, one in each of the four orthogonal directions. This chip has been used to design an encryption algorithm and a fluid flow simulator.

In this paper we explore a range of FPGA logic block architectures rather than defining a single complete architecture as above. This provides insight into the trade-offs involved in choosing a logic block, rather than a "point" experience. To make this first exploration plausible, we investigate only the effect of architectural decisions on the area efficiency. To our knowledge, this is the first such study. An early version of this work appeared in [13].

This paper is organized as follows. Section II details our experimental approach to answering the questions raised above, and gives the architectural model used in the experiments. Section III presents the results of the experiments and the detailed reasoning, both theoretical and experimental, for these results. Section IV draws more general conclusions from the specific results of the experiments.

II. General Approach and Experimental Procedure

Our purpose is to determine the area efficiency of a range of logic blocks and to find a logic block that results in a FPGA with the most functionality per unit area. This will include both the active and routing area. The approach is to implement a set of circuits using different logic blocks and programming technologies, and to determine the area required for each.

The global interconnection architecture of the FPGA used in these experiments is shown in Fig. 1. It is a regular array of identical logic blocks, separated by horizontal and vertical routing channels. The number of tracks in all of the routing channels, W, is the same. This model does not cover such general approaches such as nonho-

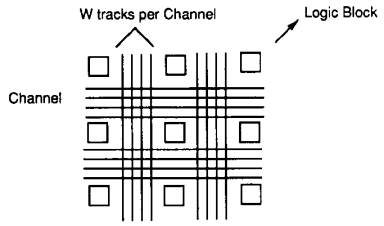


Fig. 1. Global interconnection model of the FPGA.

mogeneous logic blocks or hierarchical routing structures nor can it account for detailed techniques such as over-the-cell routing. It is general enough that most conceivable architectures can map into it and thus benefit from conclusions that are derived.

The implementation procedure described below transforms each circuit into an FPGA. The input to the procedure is a logic circuit, the functional description of the logic block (the general form of the logic blocks we consider is given in Section II-B), and a programming technology, characterized by the area it requires, as described in Section II-C. The output of the procedure is the area required to implement the circuit for the given logic block and programming technology.

The implementation procedure for each circuit, logic block, and programming technology is as follows.

1) Partition the circuit into the current logic block, in a step called technology mapping [14], [15]. This determines N, the number of logic blocks required to implement the input circuit. This is a difficult problem for FPGA blocks with table-lookup logic functions because each block may be able to implement tens of thousands of combinational functions, and conventional mappers can only handle on the order of hundreds of library elements. The *Chortle* program was developed to do this mapping. It uses a greedy algorithm that tries to collapse as many of the original logic cells as it can into each logic block. The result of this step is a new netlist that interconnects only logic blocks and is functionally equivalent to the original circuit.

2) Perform the placement of the resulting netlist. This is done using the Altor placement program [16], which is based on the min-cut placement algorithm [17]. Altor makes the array as square as possible.

3) Perform the global routing of the circuit. Global routing determines the path of channels that each wire is to take. This determines W, the largest number of tracks required in all of the channels. The approach used is similar to the LocusRoute standard cell global routing algorithm described in [18], but is changed to fit the model in Fig. 1. Note that W and the number of logic blocks are determined by the implementation procedure (and the circuit itself) and are not fixed beforehand. The effect of this approach is discussed in Section II-A below.

4) Using W, the placement dimensions, and the model for logic block area and routing pitch described in Section II-C, the area of the field-programmable gate array required to implement the circuit using the given logic block and programming technology is calculated.

A. Fixed Versus Floating Array Size and Channel Width

In this section we discuss the assumption, implied in the implementation procedure, that the realized FPGA can take on any number of tracks per channel W and any number of logic blocks N. An FPGA, however, has fixed dimensions, i.e., W and N are set at fabrication. For an exact determination of area efficiency, each circuit should be implemented in an FPGA with a fixed N and W, giving the area of the circuit implemented using each logic block as a function of these two variables. The best block is the one that minimizes area for all W and N. If the circuit does not fit into W tracks or N blocks, then its "area" would be infinity.

Unfortunately, to explore the space of possible logic blocks in this manner would require many experiments and sophisticated CAD tools. To reduce the number of experiments and the complexity of the CAD tools, we allow the number of blocks and the number of tracks per channel to "float," that is, to be a function of the circuit, and not prespecified (or "fixed") in every experiment.

The floating number of blocks can be justified because area efficiency is only an issue when the circuit just fits into the gate array. There is an error, however, in allowing the number of tracks to float. Suppose that two circuits (of the same size) achieve a minimum area using the same logic block, but using different values of W. When one value of W has to be chosen (presumably between the two) it is possible that a different logic block would achieve better total area over the two circuits. Our experimental data, however, show that circuits of the same size tend to exhibit about the same W when implemented using the same logic block. This is due to the fact that the circuits used are generally similar in type—random logic with only small amount of data path. Thus we are confident that conclusions drawn from data using the floating W routing model are close to those that would be drawn from the more exact method of experimentation using the fixed W model.

B. General Logic Block Architecture

FPGA's have lower density and speed than conventional gate arrays because connections are made through large switches that have higher resistance and capacitance than regular metal wiring. FPGA speed can be increased by grouping together greater amounts of logic in one block to reduce the amount of slower wiring. Area efficiency can also be improved by grouping more logic into a single block and reducing the number of programmable connections. Thus the logic block contains many more transistors than the basic unit of a conventional gate array, typically between 100 and 1000 transistors, and so there are many choices of logic block architecture.

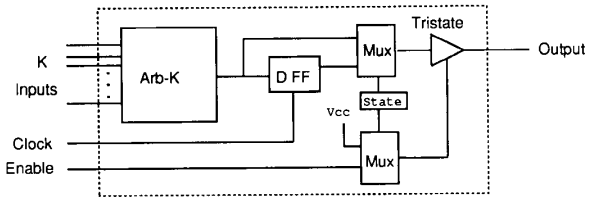


Fig. 2. General model of logic block.

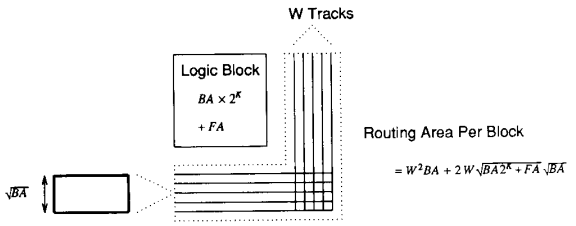


Fig. 3. Area models.

For this initial study, we chose a general architecture that contains only the essential parts of a logic block, and includes the ability to vary the amount of functionality in the block. Fig. 2 depicts the architectural model of the logic block. It consists of a K-input combinational logic function (referred to as "Arb-K"), which can implement any K to 1 combinational logic function. The lookup-table style of logic block was chosen because it is easy to vary the functionality of the logic block by changing the number of inputs to the lookup table. Note that we can consider implementation of a lookup table using any kind of programming technology, including the antifuse and EPROM.

The Arb-K is connected to a D flip-flop, which is included because sequential logic is a fundamental component of digital logic. To determine if the D flip-flop is beneficial, two versions of this basic model will be considered: one that contains the D flip-flop, and one that does not.

The flip-flop is followed by a multiplexor that selects either the flip-flop output or the Arb-K output. The multiplexor output is passed to a tristate driver that can be enabled by another input or set permanently on. The tristate is included because many of the test circuits contain tristate logic, and the circuits would have to be redesigned in fundamental ways if there were no tristate mechanism.

C. Area Models

The area calculation in step 4 of the implementation procedure requires a model that gives the logic block area and routing wire width as a function of programming technology. To create a simple model of these quantities, the programming technology is represented by one parameter: the area required to store one bit, called the bit area BA. For example, in the FPGA of [3], the bit area is the area of a static RAM bit. In the FPGA of [4] the bit area is much smaller—the size of an antifuse—which is the area of a via [11].

The area of a logic block of the form shown in Fig. 2 is a function of the number of its inputs, and the amount of fixed hardware it contains. An Arb-K block, because it can implement any K to 1 logic function, requires 2^K bits of information to be stored in a lookup table, and so must have area proportional to 2^K. The routing and circuitry required to access the Arb-K block, the area required by the D flip-flop (if it is present), and all other interconnection hardware are represented by a second parameter, called the fixed area, FA. Using BA and FA, we have the following expression for logic block area:

$$\text{logic block area} = BA \times 2^K + FA \qquad (1)$$

In a 1.25-μm CMOS technology, FA is estimated as 2100 μm^2 for logic blocks without a D flip-flop and 5100 μm^2 for logic blocks that contain a D flip-flop. The bit area for a SRAM programming technology is about 400 μm^2 and for an antifuse technology it is roughly 40 μm^2. In our experiments, we will vary the bit area both in and outside this range of values, to represent programming technologies based on EPROM [7] or ferroelectric cells [19], as well as other potential technologies that may require more area.

An estimate of the area required by wiring is important in determining the logic block because routing area can take up from 50% to over 90% of the total area, depending on the programming technology. To determine routing area the pitch of the routing track as a function of programming technology is required. Each routing track will need at least one bit of information in it, and probably several, to determine if a set of switches or fuses is open or closed. Since it is difficult to physically design a bit with highly non-square aspect ratios, the pitch of a routing track is approximated as the square root of the area required by a bit, i.e., routing pitch $= \sqrt{BA}$. Fig. 3 summarizes the area models.

III. Experimental Results

Twelve circuits were used in these experiments: five standard-cell circuits from Manufacturer A, four standard cell circuits from Manufacturer B, one standard cell circuit designed at the University of Toronto, and two PAL-based circuits designed at the University of Toronto. Table I gives a description of each circuit including its size, source, and type.

In these experiments we considered 16 different logic blocks of the type shown in Fig. 2: eight blocks with the number of inputs to the lookup table (K) ranging from two to nine including a D flip-flop, and another eight with the same range of K excluding the D flip-flop. We consider five different programming technologies, with the bit area ranging over 40, 100, 415, 800, and 1600 μm^2. This gives a total of 12 circuits $\times$ 16 logic blocks $\times$ 5 programming technologies = 960 different implementations.

TABLE I
EXPERIMENTAL CIRCUIT CHARACTERISTICS

Circuit	#Standard Cells	Source	Type
BNRA	1681	BNR	Random/Some DP
BNRB	1339	BNR	Random/Some DP
BNRC	1073	BNR	Random/Some DP
BNRD	742	BNR	Random/Some DP
BNRE	420	BNR	Random/Some DP
Z01	2266	Zymos	Telecom
Z02	3501	Zymos	Controller
Z03	560	Zymos	8-bit Multiplier
Z07	768	Zymos	Random Logic
DMA	429	UTSC	DMA Controller
BUSC	220*	UTPAL	Bus Controller
DFSM	710*	UTPAL	DRAM State Mach.

*These numbers are approximate equivalents, since the source circuits are PAL-based, not standard cells.

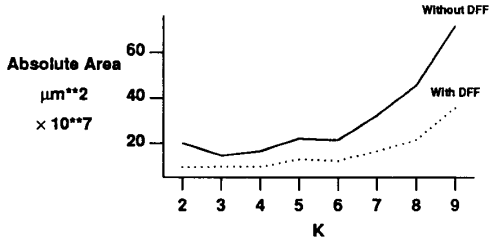


Fig. 4. Area versus K for 1073-cell circuit; bit area = 415 μm^2.

For each circuit we obtain the area of implementation required for each logic block and each programming technology. Fig. 4 gives sample results for one 1073-standard-cell circuit. It is a plot of the absolute area required to implement an FPGA versus the number of inputs to its arbitrary combinational logic block K. There are two curves, one for a logic block with a D flip-flop, and one without. The programming technology, $BA = 415$ μm^2, corresponds to a SRAM-based approach [3]. Using similar data for all of the circuits, with a range of programming technology sizes, the questions raised in the introduction were addressed.

A. Number of Inputs to Logic Block

Fig. 5 is a plot of the average normalized area over all the circuits versus K for logic blocks that contain a D flip-flop. Normalized area is defined as follows. Let the area required to implement original circuit number i in an FPGA using a logic block with K inputs to the combinational block be A^i_K. The normalized area for that circuit, N^i_K, is given by

$$N^i_K = \frac{A^i_K}{\min_{\text{all } s}\{A^i_s\}}.$$

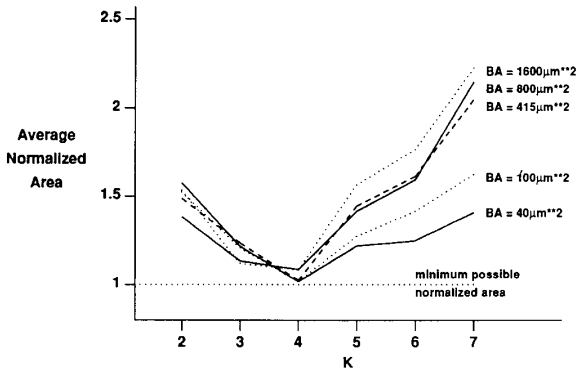


Fig. 5. Average normalized area versus K, including D flip-flop.

This normalized metric is used so that each circuit is given equal weight in the average.

Fig. 5 gives several curves for different bit areas (programming technologies). It indicates that logic blocks with K between three and four consistently achieve the lowest total area. Surprisingly, this minimum occurs with very little dependence on the programming technology. Fig. 6 is a plot of the average (over all of the circuits) of the lowest area K achieved for each circuit versus the programming technology size. The plot includes standard deviation bars of plus and minus one standard deviation. It shows the minor dependence on programming technology.

In the following sections we explain the reasons for this minimum and the minor dependence on programming technology by using empirical data bolstered with theoretical results. These explanations lead to insights into good choices for area-efficient logic blocks.

We first show the relationship between K and the total number of logic blocks. The logic block that minimizes the total active area (excluding routing area) is dependent on the programming technology. However, because the routing area dominates the total active area it is the trade-off between routing area and logic block functionality that is the key one. We then show the relationship between the routing area per block and K, and this reveals that the logic block which minimizes total routing area is technology-independent. This behavior is due to fundamental properties of circuit connectivity, and we review the relevant theoretical results in this area.

1) Active-Area Minimum K: In this section we explain the behavior of the active-area minimum K: that is, the K which minimizes the total active area. Total active area is the product of the number of logic blocks and the area of the logic block. Fig. 7 is a plot of the number of logic blocks and block area versus K using experimental and model data for circuit $A3$, a typical circuit. The product of these two curves gives the total active area as a function of K. The number of logic blocks is a decreasing function of K because a more functional logic block can implement more of the original circuit. The logic block area increases exponentially in K, as modeled by (1).

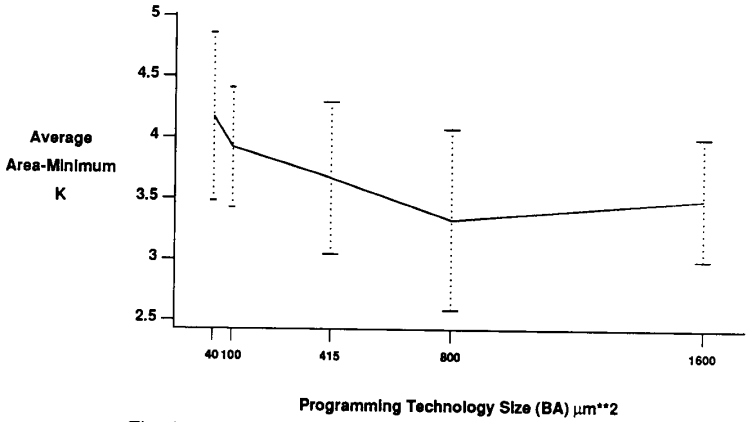


Fig. 6. Best-area K's versus programming technology size.

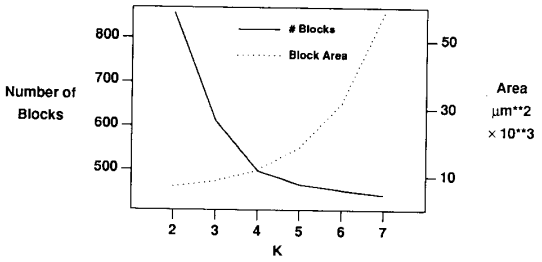


Fig. 7. Number of blocks and block area versus K.

From the shape of the two curves and the positions of their asymptotes, it is clear that their product will exhibit a minimum. This minimum is a function of the programming technology size (BA). As BA increases the dotted curve in Fig. 7 rises according to (1). If BA is larger then this causes logic blocks with larger K to become more expensive in terms of area, and reduces the active-area minimum K. For all the experimental circuits, the active-area minimum K was 2, 3, and 4 as the bit area was varied from 1600 μm^2 to 415 μm^2 to 40 μm^2, respectively. However, because the active area takes up a small amount of the total area, this technology dependence is almost completely masked by the routing area, as discussed below.

2) Relative Size of Routing Versus Active Area: For the values of K surrounding the minimum total-area K (K ranging from 2 to 6), the experimental results show that the routing area dominates the active area. Table II gives the absolute active and routing areas and their ratio for the 1073-cell circuit $A3$. The given data are for the smallest and midsize programming technologies. The data are typical of all the circuits, and show that the routing area is at least a factor of 3 greater than active area for the small technology, and as much as 15 times greater for the midsize technology. The ratio increases with programming technology size.

3) Routing-Area Minimum K: Routing area is the product of the number of logic blocks and the routing area per logic block. The routing area per block is defined to be

TABLE II
RELATIVE SIZE OF ACTIVE AND ROUTING AREA FOR CIRCUIT $A3$

K	Small Prog Tech, BA= 40μm^2			Middle Prog Tech, BA= 415μm^2		
	Active Area $\mu m^2 \times 10^7$	Route Area $\mu m^2 \times 10^7$	Ratio	Active Area $\mu m^2 \times 10^7$	Route Area $\mu m^2 \times 10^7$	Ratio
2	0.39	1.3	**3.3**	0.63	10.	**15.**
3	0.33	1.3	**3.8**	0.67	8.9	**13.**
4	0.27	1.2	**4.6**	0.57	9.0	**15.**
5	0.29	1.5	**4.9**	1.0	12.	**12.**
6	0.33	1.2	**3.8**	1.6	11.	**6.7**

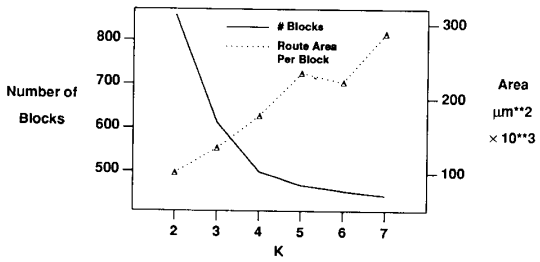


Fig. 8. Number of blocks and route area per block versus K.

the space taken by the routing tracks on two of the four sides of the logic block, as shown in Fig. 3. Fig. 8 is a plot of the number of logic blocks and the routing area per block versus K, for circuit $A3$. Note that the routing area per block curve is from experimental data; the "jog" in that curve is due to experimental variation. The solid curve is the same as in Fig. 7. The routing area per block was observed to be an increasing function of K, as W was an increasing function of K. This effect, which is not intuitively obvious, is observed consistently in all of the experiments and has been derived theoretically by El Gamal [20]. It is the underlying effect that causes most

of the results presented in this paper, and so we explore it in greater detail in Section III-A-4.

As in the case of the active area, the shape of the two curves indicates a minimum of their product—the routing-area minimum K. The technology dependence, however, is much less than for the active-area minimum K. This can be explained by analyzing the routing area surrounding each logic block. It is a function of W, the programming technology size (BA) and the size of the logic block, and can be derived by inspection of Fig. 1:

$$\text{route area per block} = W^2BA + 2WS\sqrt{BA} \qquad (2)$$

where $S = \sqrt{\text{logic block area}}$. The dominant term of the above equation is W^2BA, and it is proportional to BA. Hence the dotted curve in Fig. 8 rises proportionally to the programming technology. The minimum of the product of the two curves will not change due to BA because the minimum of any function $f(X)$ is the same when multiplied by a constant term, $C \times f(X)$. Hence the routing-area minimum K is largely independent of the programming technology. This is different from the active-area minimum K in which the term proportional to BA in (1) is of the same order as the term not proportional to BA. The K that gives the minimum routing area ranges between three and four, and on average for the data in Fig. 5, the minimum K is closest to 4.

The total-area minimum K is a function of the minimum K for the active and routing areas. Since the routing area dominates the active area as shown in Section III-A-1, it is the routing-area minimum K that dominates. Hence, the total-area minimum K is near four, and varies little with programming technology.

4) Effect of Number of Pins and Average Wire Length on W: The routing-area minimum K occurs because of the shape of the curves in Fig. 8: the number of logic blocks is a decreasing function of K, while the routing area per block is an increasing function of K. The relationship between the routing area per block and K (the dotted curve in Fig. 8) is not intuitively apparent. Since most of the results in this paper are derived from this fact, it is worth exploring in greater detail. In this section we review a relevant theoretical result that explains this relationship.

El Gamal [20] derives an expression for the expected value of the number of tracks in a gate-array channel under the following assumptions:

1) The average number of wires (connected pins) emanating from a block is λ.
2) The average wire length of point-to-point connections is $\bar{R}$, where distance is measured in the manhattan number of logic blocks the wire must traverse. No assumption is made about the wire length distribution.

TABLE III
CIRCUIT *A*3 STATISTICS: THEORETICAL AND OBSERVED CHANNEL WIDTHS

	Observed		Calculated	Observed	
K	λ	$\bar{R}$	$\frac{\lambda\bar{R}}{2}$	W_{avg}	W
2	3.8	4.1	7.8	9	13
3	3.7	3.7	6.8	9	14
4	4.2	3.4	7.2	11	16
5	4.7	3.1	7.3	11	18
6	5.1	2.9	7.4	12	16
7	5.3	3.1	8.2	12	18
8	5.5	2.9	8.0	13	17
9	6.0	2.9	8.7	15	21

3) Wires emanating from a block are uniformly "spread out" from the logic block (see [20] for a precise definition of the wire trajectories).

Using these assumptions El Gamal shows that the expected value of the width of any channel, W_{avg}, is given by

$$W_{avg} = \frac{\lambda\bar{R}}{2}. \qquad (3)$$

The expected maximum channel width W will certainly be at least this value and most probably larger.

In the context of the experiments presented in this paper, (3) says that if the number of connected pins λ increases faster than the average wire length $\bar{R}$ decreases, then the channel width will increase. We observe this to be the case for our implementations for all K greater than 2. While $\bar{R}$ does decrease as K increases, it does so more slowly than the average number of used pins increases. Note that λ is a function of K. Table III gives sample data for the 1073-cell circuit $A3$, and is typical of all of the circuits. For each K, Table III gives the measured λ, the measured average wire length $\bar{R}$, the theoretical expected average channel width $\lambda\bar{R}/2$, the observed average channel width W_{avg}, and the observed maximum channel width W. While the calculated $\lambda\bar{R}/2$ and the observed W_{avg} are somewhat different (and this is probably due to assumption 3 above and the properties of the router), there is a correspondence between the trends of the two columns. Note that between $K = 2$ and $K = 3$, $\bar{R}$ does decrease faster than λ increases, and a drop in average channel width is predicted by the theory. The measured value, however, stays the same but does not decrease. From these calculations we are confident that the observed increase in W is not an experimental aberration but a consistent property of gate array-style circuits.

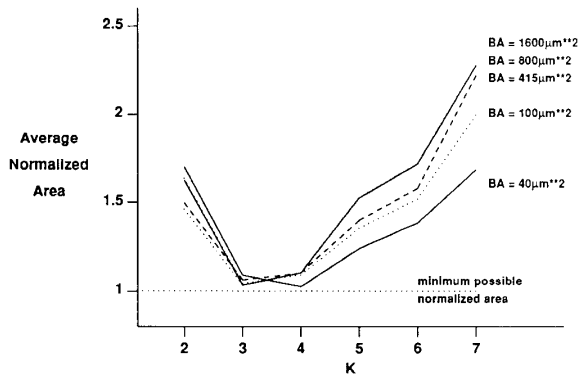


Fig. 9. Average normalized area versus K, without D flip-flop.

B. Logic Block Without D Flip-Flop

The same experiments as described in Section III-A were run using logic blocks that do not contain a flip-flop. Flip-flops were implemented using combinations of several logic blocks. Fig. 9 is a plot of the average normalized area over all the circuits versus K, using a logic block that does not contain a D flip-flop. This figure indicates that the best choice for K is in the same range (three to four) when a D flip-flop is included, but is slightly lower. This slight lowering can be explained by the theory described in Section III-A-4: the total number of logic blocks required to implement a circuit without a D flip-flop in the logic block is greater than the number required if there is a D flip-flop. This causes the average wire length $\bar{R}$ to increase because there are more blocks for each wire to traverse. From (3), and from the fact that experimental data indicate that the average number of pins emerging from a logic block (λ) is only slightly changed by the presence of a D flip-flop, we find that the number of tracks per channel increases: $W_{NoD} > W_D$, where W_{NoD} is the number of tracks per channel for a circuit implemented with logic blocks without a D flip-flop and W_D is the number when the flip-flop is included. Experimental data confirm this. From (2) the routing area per block increases as W^2. The cost, then, in routing area per block of increasing K (i.e., λ in (3)) for blocks without a D flip-flop is more than for blocks with a D flip-flop. Therefore, the K that minimizes total routing area is lower when the logic block does not include a D flip-flop.

C. Utility of the D Flip-Flop

We sought to determine if having a D flip-flop in the logic block was beneficial. An FPGA implemented using logic blocks without an embedded flip-flip requires more blocks than if there is an embedded flip-flop because each flip-flop must be implemented using several logic blocks. Experimental results show that the number of logic blocks needed to implement each circuit increased between 1.4 and 2.3 times (for the test circuits) when the flip-flop was removed from the logic block, depending on the number of flip-flops in the original circuit. The logic block size without a D flip-flop, however, is about 2.1 to 2.5 times smaller depending on the programming technology for K in the range of 3 to 4. This means that the active area without using a D flip-flop is roughly the same, but because there are about twice as many blocks, the routing area will at least double. Furthermore, as discussed above, the channels are wider because of the larger average wire length. Hence the routing area more than doubles. Since routing area dominates the overall area, this indicates that it is always better to include a D flip-flop.

TABLE IV
RATIO OF AREA WITHOUT FLIP-FLOP TO AREA WITH FLIP-FLOP

Circuit	$\frac{\text{Area Without Flip-Flop}}{\text{Area With Flip-Flop}}$	% Flip-Flops
BNRA	2.8	25%
BNRB	1.5	18%
BNRC	1.3	9%
BNRD	1.8	17%
BNRE	1.4	18%
Z01	2.1	7%
Z02	1.8	6%
Z03	0.7	0%
Z07	1.2	11%
DMA	1.4	12%
BUSC	2.1	11%
DFSM	1.9	6%

Table IV gives the ratio of area without flip-flop to area with flip-flop for all of the circuits. Since all of these numbers are greater than 1 (except for Z03, which is purely combinational), we see experimental confirmation of the above argument. The table also gives the percentage of standard cells that contain flip-flops. The ratios in Table IV are for the smallest possible programming technology. The ratio remains nearly constant over the range of programming technologies. This occurs because the routing area dominates the total area, and routing area is predominantly a linear function of the bit area. Thus, the change in programming technology cancels out in the ratio calculation.

IV. Conclusions and Future Work

This paper has explored the trade-off between the functionality of a field-programmable gate array logic block and the area required to implement circuits using that block. Our key observation is that while increasing the functionality of the logic block reduces the total number of logic blocks, this can be more than offset by an increase in routing area due to a larger number of pin connections per block. We have presented both experimental and theoretical confirmation of this effect. Since the routing area dominates the active area even for small

programming technologies, this consideration is of fundamental importance. It also implies that logic blocks which provide the most amount of functionality per connected pin will result in the best area efficiency. A lookup-table-based logic block is one good choice because each pin can be "connected" to any logic function, giving high functionality per pin.

For the specific logic block architecture that was considered we have shown that the best number of inputs to the lookup-table combinational logic block is between three and four, and that it is always beneficial to include a *D* flip-flop in the logic block. These results have little dependence on the programming technology.

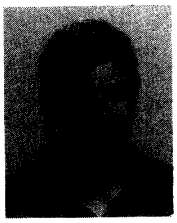
In the future, we will examine other classes of logic blocks to determine their area efficiency. We will also explore interconnection architectures with the aim of improving the density and speed of field-programmable gate arrays. These projects will also require research on CAD tools for technology mapping, placement, and routing.

ACKNOWLEDGMENT

The authors are grateful to G. Martin of Bell-Northern Research, and T. Young of Zymos for supplying the circuits and cell functional descriptions.

REFERENCES

[1] W. Carter *et al.*, "A user programmable reconfigurable gate array," in *Proc. CICC*, May 1986, pp. 233–235.
[2] H. Hsieh *et al.*, "A second generation user programmable gate array," in *Proc. CICC*, May 1987, pp. 515–521.
[3] H. Hsieh *et al.*, "A 9000-gate user-programmable gate array," in *Proc. CICC*, May 1988, pp. 15.3.1–15.3.7.
[4] A. El Gamal *et al.*, "An architecture for electrically configurable gate arrays," in *Proc. CICC*, May 1988, pp. 15.4.1–15.4.4.
[5] K. El-Ayat *et al.*, "A CMOS electrically configurable gate array," in *ISSCC Dig. Tech. Papers*, 1988, pp. 76–77.
[6] A. El Gamal *et al.*, "An architecture for electrically configurable gate arrays," *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 394–398, Apr. 1989.
[7] 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. CICC*, May 1989, pp. 5.8.1–5.8.4.
[8] Plessey Semiconductor, Swindon, England, ERA60100 preliminary data sheet, 1989.
[9] C. Marr, "Logic array beats development time blues," *Electron. Syst. Des. Mag.*, pp. 38–42, Nov. 1989.
[10] J. Rose and S. Brown, "The effect of switch box flexibility on routability of field programmable gate arrays," in *Proc. CICC*, May 1990, pp. 27.5.1–27.5.4.
[11] E. Hamdy *et al.*, "Dielectric based antifuse for logic and memory ICs," in *IEDM Tech. Dig.*, Dec. 1988, pp. 786–789.
[12] J. P. Gray and T. A. Kean, "Configurable hardware: A new paradigm for computation," in *Proc. Decennial Caltech Conf. VLSI*, C. L. Seitz, Ed., Mar. 1989, pp. 279–295.
[13] J. Rose, R. J. Francis, P. Chow, and D. Lewis, "The effect of logic block complexity on area of programmable gate arrays," in *Proc. CICC*, May 1989, pp. 5.3.1–5.3.5.
[14] K. Keutzer, "DAGON: Technology binding and local optimization by DAG matching," in *Proc. 24th Design Automation Conf.*, June 1987, pp. 341–347.
[15] E. Detjens *et al.*, "Technology mapping in MIS," in *Proc. ICCAD*, Nov. 1987, pp. 116–119.
[16] J. Rose, Z. Vranesic, and W. M. Snelgrove, "ALTOR: An automatic standard cell layout program," in *Proc. Can. Conf. VLSI*, Nov. 1985, pp. 168–173.
[17] M. A. Breuer, "Min-cut placement," *J. Design Automation Fault-Tolerant Computing*, pp. 343–362, Oct. 1977.
[18] J. Rose, "LocusRoute: A parallel global router for standard cells," in *Proc. 25th Design Automation Conf.*, June 1988, pp. 189–195.
[19] J. T. Evans and R. Womack, "An experimental 512-bit nonvolatile memory with ferroelectric storage cell," *IEEE J. Solids-State Circuits*, vol. 23, no. 5, pp. 1171–1175, Oct. 1988.
[20] A. El Gamal, "Two-dimensional stochastic model for interconnections in master slice integrated circuits," *IEEE Trans. Circuits Syst.*, vol. CAS-28, no. 2, pp. 127–138, Feb. 1981.



Jonathan Rose (S'79–M'86) received the B.A.Sc. degree with honors in engineering science in 1980, and the M.A.Sc. and Ph.D. degrees in electrical engineering in 1982 and 1986, respectively, from the University of Toronto, Toronto, Ont., Canada.

During the summer of 1983, he was with Bell-Northern Research Ltd., Ottawa, Ont., in the Integrated Circuits CAD/CAM group. From 1986 to 1989 he was a Research Associate in the Computer Systems Laboratory at Stanford University, Stanford, CA. In 1989 he joined the faculty of the University of Toronto, where he is currently an Assistant Professor of Electrical Engineering. His research interests include CAD and architecture for field-programmable gate arrays, automatic layout, and parallel CAD algorithms.

Robert J. Francis received the B.A.Sc. and M.A.Sc. degrees in electrical engineering from the University of Toronto, Ont., Canada, in 1982 and 1984, respectively.

From 1985 to 1987 he worked at the Computer Systems Research Institute, Toronto, Ont., providing system software support. He is currently a Ph.D. candidate at the University of Toronto. His research interests are in the areas of CAD tools and architectures for field-programmable gate arrays.

David Lewis (M'86) received the B.A.Sc. degree from the University of Toronto, Toronto, Ont., Canada, in 1977, and the Ph.D. degree in 1985.

From 1982 to 1985 he was employed as a research associate on the Hubnet project at the University of Toronto, and developed custom integrated circuits for a 50-Mb/s local area network. He has been an Assistant Professor at the university since 1985. His research interests include hardware accelerators for simulation, computer architecture for high-level languages, logarithmic arithmetic, and VLSI architecture.

Dr. Lewis is a member of the ACM.

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 at Stanford University, Stanford, CA, as a Postdoctoral Fellow and later as a Research Associate. He was a major contributor to the MIPS-X project. Since January 1988 he has been an Assistant Professor in the Department of Electrical Engineering at the University of Toronto. His current research interests include computer architecture, VLSI systems design and CAD tools for systems design, and field-programmable gate array architectures and applications.