The Effect of Logic Block Complexity on Area of Programmable Gate Arrays

Jonathan Rose
Computer Systems Laboratory, Stanford University, Stanford, CA 94305

Robert J. Francis, Paul Chow, David Lewis
Dept. of Electrical Engineering, University of Toronto, Ontario, Canada M5S 1A4

Abstract

This paper explores the tradeoff between the area of a Programmable Gate Array (PGA) and the functionality of its logic block. A set of industrial circuits are implemented as PGAs using tools for technology mapping, placement and routing. A simple model allows the exploration of a range of programming technologies, and accounts for the area required by wiring. Experiments indicate that for combinational logic blocks implemented using lookup tables, the best number of inputs to use is between three and four, and that a D flip-flop should always be included in the logic block. These results are independent of the programming technology.

1 Introduction

The Programmable Gate Array is an exciting new idea in semi-custom integrated circuits that reduces the IC manufacturing time from months to *minutes* and prototype cost from tens of kilodollars to under $100. The PGA was introduced in [Cart86] and newer versions have been presented in [Hsie87,Hsie88,ElGa88,ElAy88]. It is similar to a gate array in structure, but can be field-programmed to specify the function of the basic logic blocks and their interconnection. The architecture of a PGA consists of its logic block function, interconnection scheme, and I/O block design. In this paper we focus on the logic block design, and study the effect of logic block complexity on PGA area. We ignore speed considerations, even though they are very important, because we need first to determine the plausible architectures from an area perspective.

The architectural choices that affect the area of a PGA depend on the *programming technology*, which is the underlying method by which the logic function is configured and connections are made. For example, the programming technology used in [Hsie88] creates logic functions using static RAM lookup tables, and performs routing using pass transistors and multiplexors. The PGA described in [ElGa88] uses an *anti-fuse* for both logic and interconnection that, when *blown*, causes two metal tracks to be electrically joined.

This work was supported by DARPA Contract #N00014-87-K-0828, and NSERC Operating Grants #A4029 and #OGP0036648.

The logic block design is an important factor in the PGA architecture. If it has insufficient functionality then too much area must be devoted to the interconnection. If the block has excess functionality then it may suffer from under-utilization and wasted active area. We address two questions concerning the logic block design: First, should the basic logic block contain a D flip-flop? Our experiments indicate that the presence of a D flip-flop in the logic block is always desirable, regardless of the programming technology. Second, if the logic block contains an arbitrary K to 1 combinational function, what is the best number (K) of inputs to use? Our results show that the best number of inputs remains nearly constant over a wide range of programming technologies and was almost the same whether or not the block contained a D flip-flop.

2 Experimental Procedure

To answer these questions, our approach is to implement a set of circuits in a variety of logic blocks and programming technologies, and determine the area required for each.

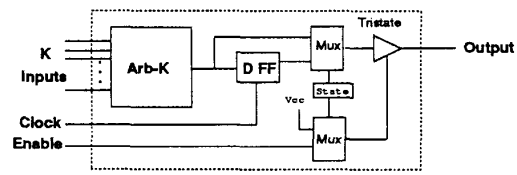

Figure 1 - *General Model of Logic Block*

Figure 1 depicts the general architectural model used for the logic block. It consists of a K-input arbitrary combinational logic function (referred to as "Arb-K"), connected to a D flip-flop 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. 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.

IEEE 1989 CUSTOM INTEGRATED CIRCUITS CONFERENCE CH2671-6/89/0000-0015 $1.00 © 1989 IEEE

The global architecture of the PGA under consideration is shown in Figure 2. 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.

The aim of the implementation procedure is to determine the area of the PGA required to implement each original circuit. A crucial notion in this method is that the number of logic blocks required for each circuit in the realized PGA is *determined* by the logic partitioning (step 1 below), rather than being pre-specified as is normally the case with any gate array. Also, W is determined by the placement and routing steps. With this approach, we learn the kind of logic block that most naturally fits each circuit.

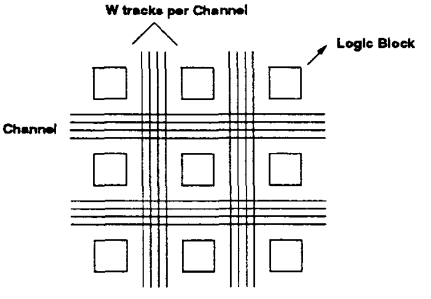

Figure 2 - *Routing Model of PGA*

The procedure described below transforms each circuit (originally in standard cell form) into a Programmable Gate Array. It takes as input the following:

1. A logic circuit, in the form of a netlist of interconnected cells.

2. A description of the logic block of the form described in Figure 1 - a value of K indicating how many inputs to the Arb-K block, and whether or not it has a D flip-flop.

3. A programming technology, parameterized by the area it requires, as described in Section 3.

The output of the procedure is the area required to implement the circuit for the specified logic block and programming technology.

Procedure: For each logic block type and programming technology:

1. Partition the original circuit into the current logic block. This is sometimes called *technology mapping* [Detj87], and is a more difficult problem for PGA logic blocks with table-lookup logic functions. This is because each logic block can contain many combinational logic functions. 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. It is functionally equivalent to the original circuit.

2. Perform the placement of the resulting netlist. This is done using the Altor placement program [Rose85], which is based on the min-cut placement algorithm [Breu77]. 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, and then determines the maximum number of tracks required in each channel, W. The approach used is similar to the LocusRoute standard cell global routing algorithm described in [Rose88], but is changed to fit the model in Figure 2.

4. Using W, the placement dimensions, and the model for logic block area and routing pitch described in Section 3, the area of the PGA required to implement the original circuit is calculated.

The above procedure makes the approximation that the global routing track count determines the number of tracks required in a channel. This is generally accepted as true for unconstrained channel routers, but may not be true for more constrained switch-based interconnection schemes. The work of [ElGa88] points out that the error in this assumption is only a few tracks.

3 Architecture Model

The area calculation in step 4 above 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, or the Bit Area (BA). For example, in the Xilinx PGA [Hsie88], the Bit Area is the area of a static RAM bit. In the Actel PGA [ElGa88] the bit area is much smaller, the size of an anti-fuse, which is the minimum square area of a metal wire [ElAy88].

The area of a logic block of the form shown in Figure 1 is also 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 is represented by a second parameter, called the Fixed Area (FA). Using BA and FA, we have the following expression for logic block area:

$$Logic\ Block\ Area = BA \times 2^K + FA \quad (1)$$

In a 1.25μm CMOS process technology, FA has been estimated to be $2100\mu m^2$ for logic blocks without a D flip-flop and $5100\mu m^2$ for logic blocks that contain a D flip-flop. The Bit Area for an SRAM programming technology is about $400\mu m^2$ and for an anti-fuse technology is roughly $40\mu m^2$. In our experiments, we will vary the Bit Area between and above these two values, to represent programming technologies based on EPROM or Ferroelectric cells [Evan88], as well as potentially faster technologies that may take much 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. $Routing\ Pitch = \sqrt{BA}$.

4 Experimental Results

The circuits used in these experiments are five standard-cell circuits obtained from Bell-Northern Research. They range in size from 420 to 1681 standard cells, and consist of a mix of random logic and data path circuits.

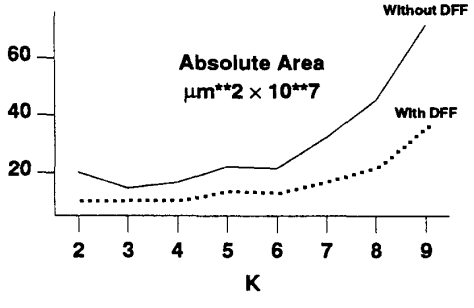

Figure 3 - *Area versus K for 1073-Cell Circuit*

Figure 3 gives example results for a 1073-cell circuit. It is a plot of the absolute area required to implement a PGA 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 = $415\mu m^2$, corresponds to an SRAM-based approach [Hsie88]. Using similar data for all of the circuits, with a range of programming technology sizes, the questions raised in the introduction were addressed.

4.1 Number of Inputs to Logic Block

I. Logic Block Contains D Flip-Flop. Figure 4 is a plot of the sum of the normalized area of all the circuits versus K, where normalized area is defined as follows: Let the area required to implement original circuit number i in a PGA using a logic block with k inputs to the combinational block be A_k^i. The normalized area for that circuit, N_k^i, is given by:

$$N_k^i = \frac{A_k^i}{\min_{\text{all } s}\{A_s^i\}}$$

The sum of the normalized areas over all five circuits for a given k is given by $\sum_{i=1}^{5} N_k^i$.

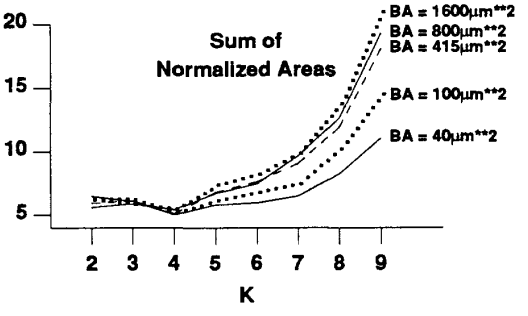

Figure 4 - *Sum of Normalized Areas versus K Using DFF*

The data in Figure 4 are for PGAs with logic blocks that contain a D flip-flop. The figure gives several curves for different bit areas (programming technologies). It is clear, from the dip at K = 4, that a 4-input arbitrary logic block consistently achieves the lowest area. This minimum can be explained by the separate effect of K on active area and routing area, as follows:

Active area is the product of the number of logic blocks and the area of each logic block. Figure 5 is a plot of the number of logic blocks and block size versus K using experimental and model data for a 1073-cell 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 with larger K, a logic block can consume more standard cells, reducing the total number of blocks. The logic block area increases exponentially in K, as modeled by equation 1 in Section 3.

Since the two curves are monotonically increasing and decreasing in K, their product exhibits a clear minimum. This minimum is a function of the Bit Area. As *BA* increases the dotted curve in Figure 5 rises according to equation 1. 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 1600μm^2 to 415 μm^2 to 40 μm^2 respectively.

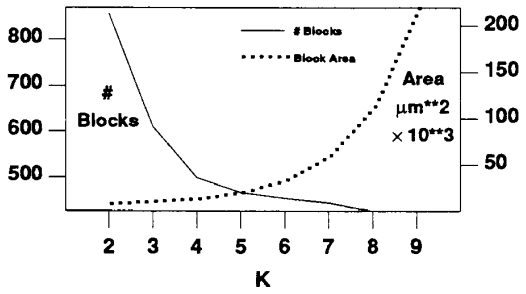

Figure 5 - *#Blocks and Block Area vs. K*

Routing area is the product of the number of logic blocks and the routing area per logic block. The routing area per block is the space taken by the routing tracks on two of the four sides of the logic block. Figure 6 is a plot of number of logic blocks and the routing area per logic block versus K, for the 1073-cell circuit. The routing area per block was observed to be an increasing function of K, as W was an increasing function of K. This effect occurs because as the number of pins in each logic block goes up, congestion (which is measured by W) increases because more wiring has to occur in a smaller area.

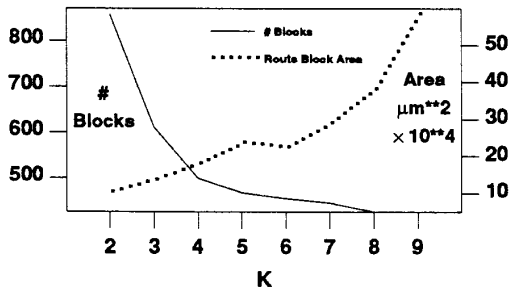

Figure 6 - *#Blocks and Route Area per Block vs. K*

The routing area surrounding each logic block is a function of W, the number of tracks per channel, the programming technology size (BA) and the size of the logic block itself. It can be derived by inspection of Figure 2 and is given by:

$$Route\ Area\ Per\ Block = W^2 BA + 2 \times WS\sqrt{BA}$$

where $S = \sqrt{Logic\ Block\ Area}$. Note that the routing-area minimum K is highly dependent on the accuracy of W - because the routing area is dominated by a term proportional to W^2. The routing-area minimum K also varies due to programming technology, but the number tracks per channel, W, is the stronger influence. Because the global router is not accurate to an exact number of tracks, we can only extract general trends from the data. The trend is clear, however - the K that gives the minimum routing area ranges between three and four, and on average for the data in Figure 4, the minimum K is closer to 4.

The total-area minimum K is a combination of the minimum K for the active and routing areas. However, because the routing area takes up from about 70% of the total area (for small bit areas) to over 95% (for large bit areas), it is the routing area that dominates. Hence, the total-area minimum K is near four, and varies little with programming technology.

II. Logic Block Without D Flip-Flop. Figure 7 is a plot of the sum of the normalized area over all circuits, 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), as for PGAs with logic blocks that contain a D flip-flop. The minimum-area K's for active area and routing area exhibit the same behavior as described above.

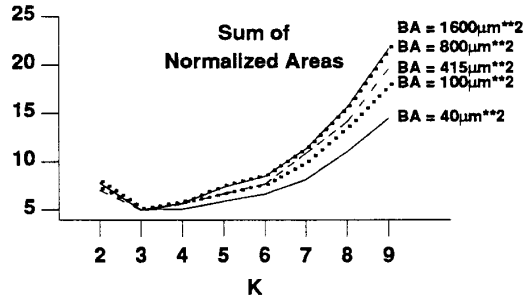

Figure 7 - *Sum of Normalized Areas versus K Without DFF*

4.2 Utility of the D Flip-Flop

We also sought to determine if having a D flip-flop in the logic block was beneficial. A PGA implemented using logic blocks without flip-flops requires more blocks than the same PGA implemented using logic blocks that have flip-flops. When there is no flip-flop each memory element must be implemented by a combination of several logic blocks. The technology mapping program, Chortle, showed that the number of logic blocks needed to implement each circuit increased between 1.9 and 2.5 times when the flip-flop was removed from the logic block. The logic block size without a D flip-flop, however, is

about 2.1 to 2.5 times smaller depending on the programming technology. This means that the active area without using a D flip-flop is about the same, but because there are about twice as many blocks, the routing area will roughly double. Since routing area dominates the overall area, this indicates that it is always better to include a D flip-flop.

Figure 8 is a plot of $\frac{Area\ Without\ Flip\text{-}Flop}{Area\ With\ Flip\text{-}Flop}$ versus Bit Area for each of the five circuits. It indicates when it is advantageous to use a flip-flop — if this quantity is greater than one, then it is better to use a flip-flop in terms of total area. The numbers at the end of each line in the figure are the number of standard cells in the original circuit. For all of the circuits, in varying degrees depending on what proportion of flip-flops they contain, it is clearly advantageous to include a flip-flop. 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.

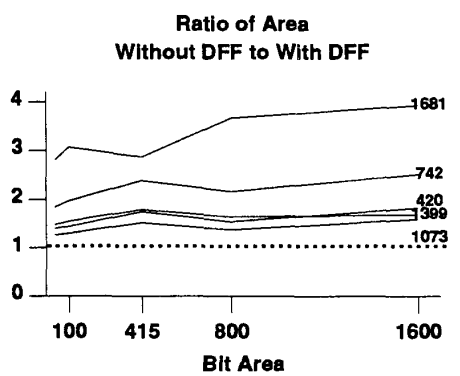

Figure 8 - *Without DFF:With DFF versus Bit Area*

5 Conclusions and Future Work

We have presented a procedure and a model for evaluating the effect of the complexity and functionality of the logic block on the area of Programmable Gate Arrays. The approach allows the investigation of a range of programming technologies, and takes the routing area requirements into account. Using this method, we have demonstrated that a good number of inputs to use for the arbitrary combinational logic block is between three and four, independent of programming technology. In addition, we have demonstrated that it is always advantageous to include a D flip-flop as part of the logic block, regardless of the programming technology.

In the future, we will examine logic blocks that do not use lookup tables, such as AND-OR structures, to determine if more functionality per unit area can be obtained. We will also explore interconnection architectures with the aim of improving the density and speed of Programmable Gate Arrays. These projects will also require research in CAD tools for technology mapping, placement and routing.

6 Acknowledgements

The authors are grateful to Grant Martin of Bell-Northern Research for supplying the circuits and cell functional descriptions.

7 References

[Breu77]
M.A. Breuer, "Min-Cut Placement," Journal of Design Automation and Fault-Tolerant Computing, pp. 343-362, Oct 1977.

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

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

[ElAy88]
K. El-Ayat, et. al, "A CMOS Electrically Configurable Gate Array," Proc. 1988 ISSCC, pp. 76-77.

[ElGa88]
A. El Gamal, et. al, "An Architecture for Electrically Configurable Gate Arrays," Proc. 1988 CICC, May 1988, pp. 15.4.1 - 15.4.4.

[Evan88]
"An Experimental 512-bit Nonvolatile Memory with Ferroelectric Storage Cell," J.T. Evans, R. Womack, IEEE JSSC, Vol 23, No. 5, Oct. 1988, pp. 1171-1175.

[Hsie87]
H. Hsieh et. al, "A Second Generation User Programmable Gate Array," Proc. 1987 CICC, May 1987, pp. 515-521.

[Hsie88]
H. Hsieh, et. al "A 9000-Gate User-Programmable Gate Array," Proc. 1988 CICC, May 1988, pp. 15.3.1 - 15.3.7.

[Rose85]
J. Rose, Z. Vranesic, W. M. Snelgrove, "ALTOR: An Automatic Standard Cell Layout Program," Proc. Can. Conf. on VLSI, Nov. 1985, pp. 168-173.

[Rose88]
J. Rose, "LocusRoute: A Parallel Global Router for Standard Cells," Proc. 25th DAC, June 1988, pp. 189-195.