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, Toronto, Ont. Canada

1987

1 Introduction

The Programmable Gate Array (PGA) 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,ElGa88a,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. This paper studies the effect of logic block complexity on total circuit area for PGAs.

The *architecture* of a PGA consists of its logic block function, interconnection scheme, I/O block design and the global structure. There are many tradeoffs between architecture, area, and speed, each of which depends heavily on the *programming technology*. Programming technology is the underlying method by which the logic function is set and the connections are implemented at program time. For example, the programming technology used in [Hsie88] is based on static RAM and pass transistors, while that of [ElGa88a] uses an anti-fuse. In this paper we focus on the effect of logic block complexity on PGA area, ignoring speed considerations. While circuit speed is very important, this work represents an initial exploration into plausible architectures from an area perspective.

We address two questions: First, should the basic logic block contain a significant amount of fixed hardware, such as a D flip-flop? Our experimental results indicate that a D flip-flop is desirable for large programming technologies (like SRAM [Hsie88]) but that it is inefficient for smaller technologies such as the anti-fuse. Second, for logic blocks containing arbitrary combinational logic functions, (i.e. *any* K to 1 logic function) what is the best number (K) of inputs to use? Surprisingly, 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.

This work was supported by DARPA Contract #N00014-87-K-0828, and NSERC Operating Grants #A4029 and #A4053.

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. This data will indicate an appropriate choice of logic block, in terms of area, for a given technology.

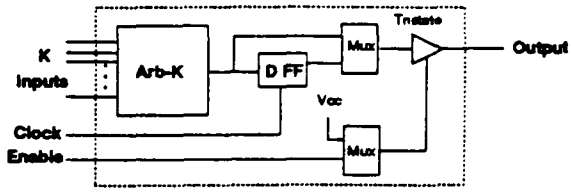

Figure 1 - *General Model of Logic Block*

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 an "Arb-K"), connected to a D flip-flop followed by a multiplexer that selects either the flip-flop output or the Arb-K output. Its output is passed to a tristate driver that can be enabled by another input or left permanently on. To determine if the D flip-flop is beneficial, two variations of this basic model will be considered: one that contains the D flip-flop, and one that does not.

The global architecture of the PGA under consideration is shown in Figure 2. It is a regular array of logic blocks, separated by horizontal and vertical routing channels. The number of tracks in all of the routing channels, W, is the same. Since we want to know the area requirements of a logic block architecture, a crucial concept in this procedure is that W is *determined* by the placement and routing for each circuit.

The following procedure performs the circuit implementation:

Input: a logic circuit, a range of K's indicating how many inputs on the Arb-K block, and a set of programming technologies.

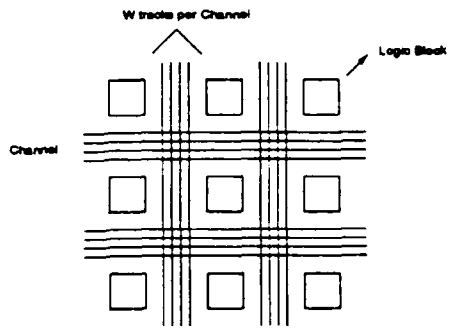

Figure 2 - *Routing Model of PGA*

Output: for each (K, programming technology) pair the area required to implement the circuit with a logic block that contains a D flip-flop, and the area for a logic block that does not.

Procedure: For each logic block type:

1. Partition the original circuit into the current logic block. This is sometimes called *technology mapping* [Detj87], but is a more difficult problem for PGAs because each logic block can collapse many combinational logic functions. The *Chortle* program was developed to do this mapping [Fran88]. It uses a greedy algorithm that tries to collapse as many standard cells as it can into each logic block.

2. Perform the placement of the resulting circuit. 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 hence determines the maximum number of tracks required in each channel, W. The algorithm used is similar to the one described in [Rose88], but is changed to fit the routing model pictured in Figure 2.

4. Section 3 describes a model for the logic block area and routing area as a function of K and programming technology. With this model, W, and the placement dimensions, the circuit area for a range of programming technologies 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 switch-based routing schemes. We have reason to believe, however, that the error in this assumption is only a few tracks [ElGa88b].

3 Architecture Model

The area of a logic block is a function of the number of its inputs, the amount of fixed hardware it contains, and the programming technology. The pitch of the routing track can be approximately modeled as a function of the programming technology.

The programming technology is represented by one parameter: the area required to store one bit in the technology, or 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 [ElGa88a] the bit area is much smaller, close to the space required by an anti-fuse. The overhead required to access the Arb-K block and the area required by the D flip-flop (if it is present) and all other non-arbitrary logic function hardware is represented by a second parameter, called the Fixed Overhead Area (FA).

An Arb-K block, because it can implement *any* K to 1 logic function, requires 2^K bits of information to be stored and so must have area proportional to 2^K. Using this, we can derive the following expression for logic block area:

$$Logic\ Block\ Area = BA \times 2^K + FA$$

where BA is the bit area in the programming technology and FA is the fixed overhead.

The basic technology is assumed to be 1.25μm CMOS. FA has been calibrated using data acquired from Xilinx [Cart88], giving FA = $1200\mu m^2$ for logic blocks without a D flip-flop and $16000\mu m^2$ for logic blocks with a D flip-flop. The corresponding Bit Area for an SRAM programming technology is $400\mu m^2$ and is roughly $40\mu m^2$ for an anti-fuse technology. In our experiments, we will vary the Bit Area between and above these two values.

Though the PGA interconnection scheme is not addressed directly in this paper, the *area* required by routing is an important factor in determining the logic block. We need to know the *pitch* of the routing track as a function of programming technology. 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 lay out 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 [Mart88],

ranging in size from 420 to 1681 standard cells. They consist of a mix of random logic and data path circuits. Figure 3 is a plot of absolute area for the PGA versus number of inputs to the arbitrary combinational logic block, K, for a 1073 standard cell circuit.

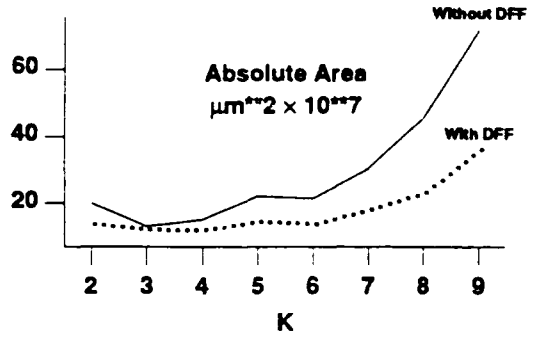

Figure 3 - *Area for versus K for One Circuit*

There are two curves - one giving area when the logic block contains a D flip-flop, and one without. The programming technology, BA = 415μm^2, corresponds to an SRAM-based approach [Hsie88]. Using similar data for all of the circuits, with more programming technologies, the questions raised in the introduction were addressed.

4.1 Number of Inputs to Logic Block

Figure 4 shows the sum of the normalized areas over all of the circuits, versus K. The normalized area for a circuit is determined by dividing the area using logic block K by the best area over all K. The logic block used in this data contains a D flip-flop. Figure 4 gives several plots for different bit areas (programming technologies).

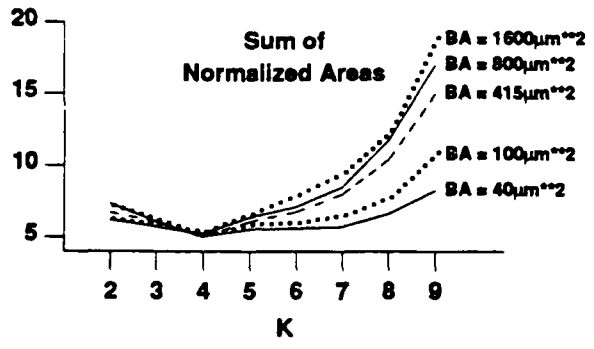

Figure 4 - *Sum of Normalized Areas versus K Using DFF*

It is clear, from the dip at K = 4, that a 4-input arbitrary logic block consistently achieves the lowest area. Surprisingly, this number is constant over a wide range of bit areas. It is due to the fact that, for a given K, the area is predominantly a linear function of the bit area.

The number of logic blocks increases when the logic block has no flip-flop because the D flip-flops must then be implemented in combinational gates. Since the size of each logic block is less, the final area may or may not be smaller. Figure 5 is a normalized area plot for logic blocks that do not contain D flip-flops. The best number of inputs in this case is three, only slightly different than the D flip-flop case. Again, this number is independent of programming technology.

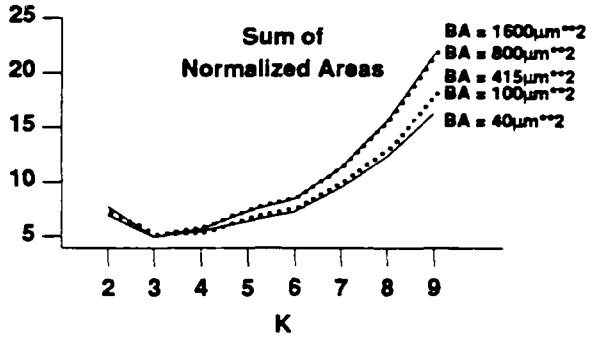

Figure 5 - *Sum of Normalized Areas versus K Without DFF*

In both cases, the best K is low (3 or 4) primarily because the circuits cannot make effective use of the larger K blocks. because the increasing functionality comes at the cost of a much greater active area, which exponentially increases in K, it doesn't pay to use the larger logic blocks.

4.2 Utility of the D Flip-Flop

Figure 6 is plot of circuit area using and not using a D flip-flop versus Bit Area for a 1073 standard cell circuit. The circuit area used is the one obtained with the lowest area K.

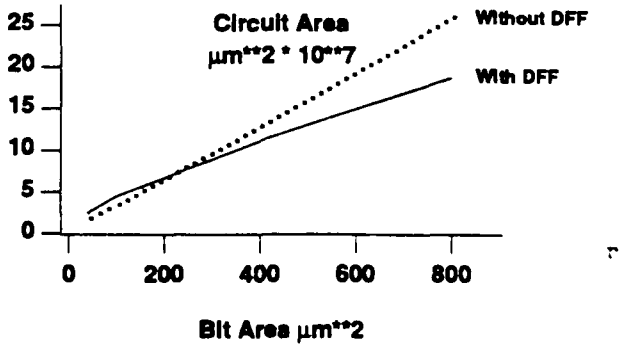

Figure 6 - *Area versus Bit Area 1 Circuit*

This figure shows that, for very small bit areas, it is advantageous not to use a D flip flop, but larger bit areas

perform better by using a D flip flop. This is true for all of the circuits, but the cross-over point is different for each.

Figure 7 is a plot of $\frac{Area\ Without\ Flip\text{-}Flop}{Area\ With\ Flip\text{-}Flop}$ versus Bit Area for each circuit, indicating when it is advantageous to use a flip-flop. In the smaller bit areas, corresponding to an anti-fuse programming technology, the use of a D flip flop is unprofitable. This is the case in the Actel PGA [ElGa88]. The middle and larger bit areas, corresponding to the SRAM program technology, can benefit by including a D flip-flop, and in fact the Xilinx PGA [Hsie88] uses two D flip-flops.

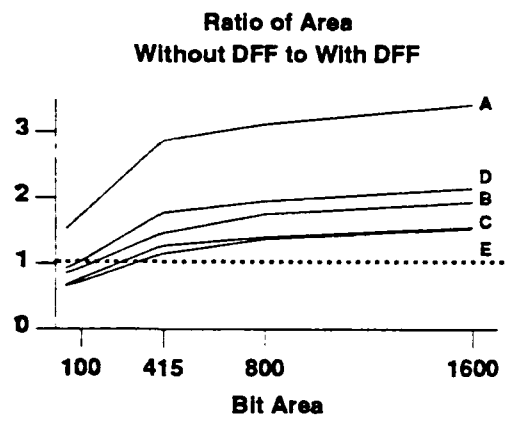

Figure 7 - *Without DFF:With DFF versus Bit Area*

5 Conclusions and Future Work

We have presented a procedure and model to evaluate different logic block architectures for Programmable Gate Arrays, on the basis of circuit area. Using this method, for a particular set of circuits, we have demonstrated a good number of inputs to use for the arbitrary combinational logic block. In addition, we have displayed the trade-off between programming technology area and utility of a D flip-flop in the logic block.

There is an enormous amount of future work to be done in this field. We would like to investigate more kinds of logic blocks - in particular those with less arbitrary logic functions. Other work will directly address questions dealing with circuit speed. This relates to the specific architecture of the interconnection scheme. All of this work needs to be implemented on a wider range of circuits. New CAD algorithms need to be developed for PGAs. Our technology mapper needs more development, and the placement and routing needs to address the specific needs of PGAs. PGAs, because they promise such enormous economic advantages, are a fertile and growing field of research and development.

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.

[Cart88]
W. Carter, Private Communication.

[Detj87]
E.Detjeus 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.

[ElGa88a]
A. El Gamal, et. al, "An Architecture for Electrically Configurable Gate Arrays," Proc. 1988 CICC, May 1988, pp. 15.4.1 - 15.4.4.

[ElGa88b]
A. El Gamal, Private Communication.

[Fran88]
R.J. Francis, "Chortle: A Technology Mapping Algorithm for Programmable Gate Arrays," in preparation.

[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.

[Mart88]
Grant Martin, Bell-Northern Research, private communication.

[Rose85]
J. Rose, et. al, "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.