The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density

Elias Ahmed
Dept. of Electrical & Computer Engineering
University of Toronto
Toronto, Canada
elias@eecg.toronto.edu

Jonathan Rose
Dept. of Electrical & Computer Engineering
University of Toronto
Toronto, Canada
jayar@eecg.toronto.edu

ABSTRACT

In this paper we revisit the FPGA architectural issue of the effect of logic block functionality on FPGA performance and density. In particular, in the context of lookup table, cluster-based island-style FPGAs [4] we look at the effect of lookup table (LUT) size and cluster size (number of LUTs per cluster) on the speed and logic density of an FPGA.

Although this question was addressed some time ago in [17] [18] [12] [13] [10] and [22], several reasons compelled us to revisit the issue. First, prior work focused on non-clustered logic blocks, which are known to have a significant impact on the area and delay [16]. Second, most prior studies tended to look at area or delay, but not both as we will here. Third, prior results were based on IC process generations that are several factors larger than current process generations, and so do not take deep-submicron electrical effects into account. In the present work, we perform detailed spice-level simulations of circuits and perform appropriate buffer and transistor sizing for all the logic and routing elements, in the manner of [4]. Fourth, the CAD tools available today for experimentation are significantly better than those available 10 years ago, when this question was first raised. Our new results show that the superior tools give rise to different trends in the explanation of the results. Finally, a recent publication [11] has suggested that a more fine-grained logic block (smaller LUT size) is a better choice than was previously thought.

We use a fully timing-driven experimental flow [4] [15] in which a set of benchmark circuits are synthesized into different cluster-based [2] [3] [15] logic block architectures, which contain groups of LUTs and flip-flops. We look across all architectures with LUT sizes in the range of 2 inputs to 7 inputs, and cluster size from 1 to 10 LUTs. In order to judge the quality of the architecture we do both detailed circuit level design and measure the demand of routing resources for every circuit in each architecture.

These experiments have resulted in several key contributions. First, we have experimentally determined the relationship between the number of inputs required for a cluster as a function of the LUT size (K) and cluster size (N). Second, contrary to previous results, we have shown that when the cluster size is greater than four, that smaller LUTs (size 2 and 3) are almost as area efficient as 4-input LUTs, as suggested in [11]. However, our results also show that the performance of FPGAs with these small LUT sizes is significantly worse (by almost a factor of 2) than larger LUTs. Hence, as measured by area-delay product, or by performance, these would be a bad choice. Also, we have discovered that LUT sizes of 5 and 6 produce much better area results than were previously believed. Finally, our results show that a LUT size of 4 to 6 and cluster size of between 4 and 10 provides the best area-delay product for an FPGA.

FPGA 2000 Monterey CA USA

1. INTRODUCTION

Several studies in the past have examined the effect of logic block functionality on the area and performance of FPGAs. The work in [18] and [14] showed that a LUT size of 4 is the most area efficient in a non-clustered context. In addition, it was demonstrated in [21] [22] and [12] that using a LUT size of 5 to 6 gave the best performance. A recent publication [11] has suggested that using a heterogeneous mixture of LUT sizes of 2 and 3 was equivalent in area efficiency to a LUT size of 4, and hence could be a good choice. In addition [1] states that a logic structure using two 3-input LUTs was most beneficial in terms of area and speed. However, it must be noted that both these last two papers did not perform a full area or delay study where a range of LUT sizes was examined.

These newer suggestions and the fact that many things have changed (as listed in the abstract above) since the original studies on the subject compel us to revisit the issue. In addition, careful analysis in this kind of study may well lead to suggestions for better architectures. It is also worth emphasizing that the study presented in this paper employs significantly better CAD tools than before, and vastly more detailed area and delay models for the underlying FPGA circuits. Furthermore, the experimental flow continues to its proper end, at the detailed routing level on "reasonable" [4] routing architectures, which previous studies could not afford to do.

The focus of this paper is to determine the effect of the number of inputs to the LUT (K) (in a homogeneous architecture) and the number of such LUTs in a cluster (N) on the performance and density of an FPGA. A cluster [2] [3] is group of basic logic elements (BLEs) that are fully connected by a mux-based cross bar as illustrated in Figure 2. The Altera Flex 6K, 8K, 10K, and Xilinx 5200 and Virtex are commercial examples of such clusters (although the Xilinx logic clusters are not fully connected).

Increasing either LUT size (K) or cluster (N) increases the functionality of the logic block, which has two positive effects: it decreases the total number of logic blocks needed to implement a given function, and it decreases the number of such blocks on the critical path, typically improving performance. Working against these positive

effects is that the size of the logic block increases with both K and N. The size of the LUT is exponential in K [18] and the size of the cluster is quadratic in N [2]. Furthermore, the area devoted to routing outside the block will change as a function of K and N, and this effect (since routing area typically is a large percentage of total area) has a strong effect on the results. The choice of the logic block granularity which produces the best area-delay product lies in between these two extremes. In exploring these trade-offs we seek to answer the following questions:

- For a cluster-based logic block with N LUTs of size K and I inputs to the cluster, what should the value of I be so that 98 % of the LUTs in the cluster can be fully utilized? (Certainly setting I=K×N will do this, but a value less than this, which is cheaper, may also suffice).
- What is the effect of K and N on FPGA area?
- What is the effect of K and N on FPGA delay?
- Which values of K and N give the best area-delay product?

More crucially, we would like to clearly explain the results and thus perhaps leading to better architectures.
This paper is outlined as follows: Section 2 describes the global architecture of the FPGA we employ, as well as the internal structure of the clustered logic blocks used throughout this paper. Section 3 details the experimental CAD flow and steps that were performed to produce the results. Section 4 describes the logic and routing architectures, and some details of the area and delay modeling. Section 5 presents the key results from these experiments. Finally, we conclude in Section 6.

2. GLOBAL ARCHITECTURE AND INTERNAL STRUCTURE OF CLUSTERS

The basic FPGA architecture we employ is an "island-style" structure where an array of logic blocks are surrounded by routing channels as shown in Figure 1. The I/O pads are evenly distributed around the perimeter of the FPGA.

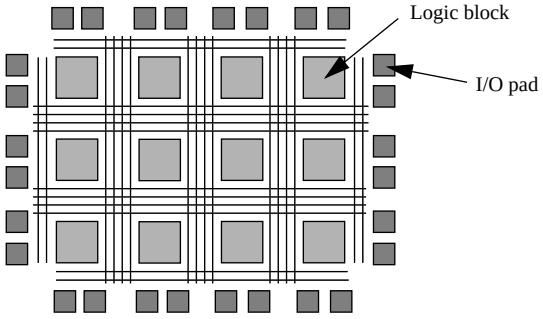

Figure 1: Island-Style FPGA [4]

The structure of the cluster-based logic block used in our experiments is illustrated in Figure 2b. Each cluster contains N basic logic elements (BLEs) fed by I cluster inputs. The BLE, illustrated in Figure 2a, consists of a K-input lookup table (LUT) and register, which feed a two-input mux that determines whether the registered or unregistered LUT output drives the BLE output.

For clusters containing more than one BLE, we assume a "fully connected" [2] approach; this means that all I cluster inputs and N outputs can be programmably connected to each of the K inputs on every LUT. These are implemented using the multiplexers shown in the Figure, which are not necessary for clusters of size N=1.

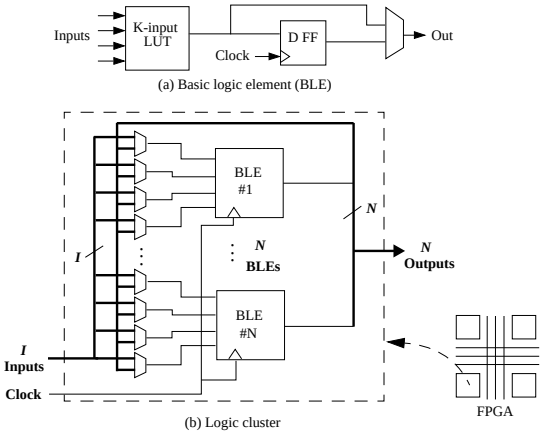

Figure 2: Structure of (a) Basic Logic Element (BLE) and (b) Logic Cluster [4]

3. EXPERIMENTAL METHODOLOGY

The best-known and most believable method of determining the answers to the questions posed in the introduction is to experimentally synthesize real circuits using a CAD flow into the different FPGA architectures of interest, and then measure the resulting area and delay [5] [4] [12]. Figure 3 illustrates the CAD flow that we employ. First, each circuit passes through technology-independent logic optimization using the SIS program [20]. It is worth noting that, from this point on, the entire CAD flow is fully timing-driven. Technology mapping (which converts the logic expressions into at netlist of K-input LUTs), was performed using the FlowMap and FlowPack tools [8]. Then, all the registers and LUTs were packed into logic clusters using the timing driven packing algorithm (T-VPACK) [16]. This was followed by timing-driven placement using a timing-enhanced version [16] of VPR [4]. Then full path-based and timing-driven routing is performed using VPR [4].
In our approach to modeling the area of an FPGA required by any given circuit, we determine the minimum number of tracks needed to successfully route each circuit, W_{min}. Clearly this isn't possible in real FPGAs, but we believe this is meaningful as part of a logic density metric for an architecture. The area model which makes use of this minimum track count is described more fully in Section 4. In order to determine the minimum number of tracks per channel to route each circuit we continuously route each circuit, removing tracks from the architecture until it fails to route. We call the situation where the FPGA has the minimum number of tracks needed to route a given circuit a "high stress" routing since the circuit is barely routable. We believe that measuring the performance of a circuit under these high-stress conditions is unreasonable and atypical, because FPGA designers don't like working just on the edge of routability. They will typically change something to avoid it, such as using a larger device, or removing part of the circuit.
For this reason, we add 30% more tracks to the minimum track count and then perform final "low stress" routing and use that to measure the critical path delay.

From the output of the router, and using the area and delay models described in the next section, we can compare different architectures.

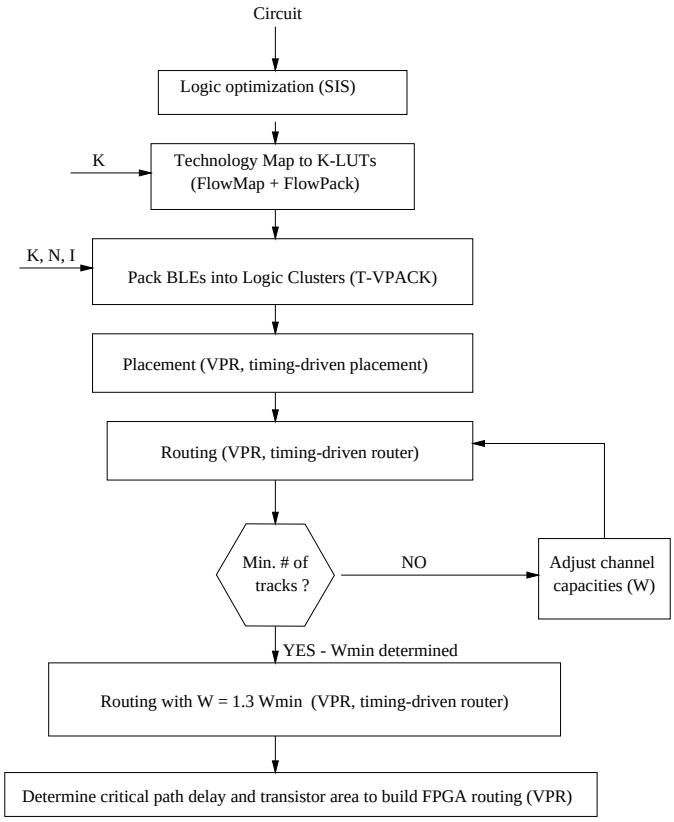

Figure 3: Architecture Evaluation Flow

4. FPGA ARCHITECTURE MODELING

In this section we give a brief description of the area and delay modeling developed by Betz [4]. The level of detail present in these models goes far beyond any modeling previously used in this kind of experimental analysis. All device parameters and circuits are modeled using SPICE simulations of a 0.35 µm CMOS process.
We make the following assumptions about the basic island-style architecture:

- The number of routing tracks in each channel between logic blocks is uniform throughout the FPGA.
- All metal routing wires are placed on metal layer 3 with minimum width and spacing.
- Each circuit is mapped into the smallest square (MxM) grid possible given the number of logic clusters it requires.

  However, it is important to note that the area metric we count is not the total area required by the square M x M block on the FPGA. Rather, we use the exact number of clusters required to implement the circuit. For example, a circuit which requires 800 logic blocks will be routed in 29 x 29 FPGA grid which results in 841 blocks. We use the area of the logic and routing surrounding 800 clusters as opposed to 841.

4.1 Area Model

Betz' area modeling procedure [4] was to create the detailed, transistor-level circuit design of all of the logic and routing circuitry in the FPGA. This includes circuits for the LUTs, flip-flops, intra-cluster muxes, inter-cluster routing muxes and switches and all of the associated programming bits. His basic assumption was that the total area of the FPGA was active-area limited, which tends to be true when there are many layers of metal. Two commercial PLD vendors have confirmed this assumption.
The design process includes proper sizing of all of the gates and buffers, including the pass-transistors in the routing. Betz uses the number of "minimum-width transistor areas" as his area metric. The definition of a minimum-width transistor area is the smallest possible layout area of a transistor that can be processed for a specific technology plus the minimum spacing surrounding the transistor as shown in Figure 4. The spacing is dictated by the design rules for that particular technology. Any transistors in the circuit design that are sized larger than minimum are counted as a greater number of minimum-width transistors, taking into account the fact that a double size transistor takes less than twice the layout area. One advantage of this metric is that it is a somewhat process-independent estimate of the FPGA area.

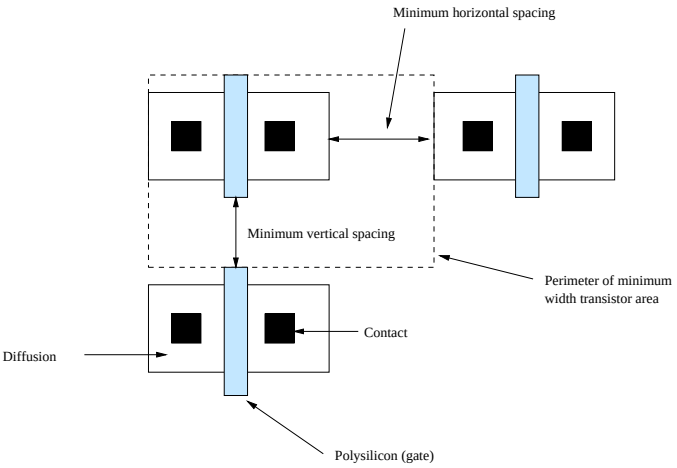

Figure 4: Definition of a Minimum-Width Transistor Area [4]

4.2 Logic Circuit Design and Delay Model

The circuit design process described above is also necessary to determine accurate delay measurements of the final placed and routed circuit. In deep-submicron IC design processes, the effect of wire resistance and capacitance becomes more prevalent. We account for these effects in this delay modeling. Figure 5 shows the detailed logic block circuit. The timing values given are based on SPICE simulations of a 0.35 µm, 3.3 V CMOS process. The paths have been simulated with their actual loads in place and the input driven by what would actually be driving it in a real FPGA.
As the cluster size increases, the buffers shown in Figure 5 must be sized larger because of larger loading from the internal muxes, which results in an increase in the basic BLE delay. This is shown in Table 1 which gives the logic delays as the cluster size increases for the paths indicated in Figure 5 for a BLE based on a 4-input LUT.
Similarly, the design of the larger LUTs must be done carefully, with proper buffer sizing and, in some cases, insertion of buffers within

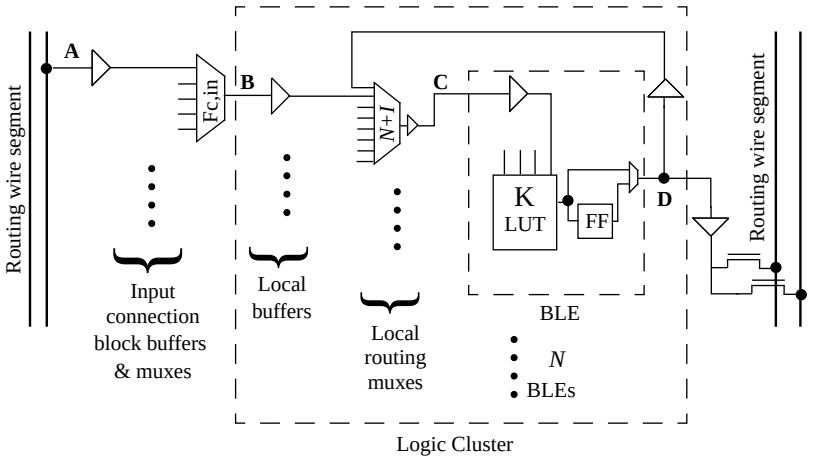

Figure 5: Structure and Speed Paths of a Logic Cluster [4]

Table 1: Logic Cluster Delays for 4-input LUT Using 0.35 µm CMOS process

Cluster Size (N)	A to B (ps)	B to C and D to C (ps)	C to D (ps)	B to D (ps)
1 (No local routing muxes)	760	140	438	578
2	760	649	438	1087
4	760	761	438	1199
6	760	849	438	1287
8	760	892	438	1330
10	760	912	438	1350

the tree of pass-transistors. Table 2 presents the LUT delay as a function of the LUT size.

Table 2: LUT Delays Using 0.35 µm CMOS process

LUT Size (K)	C to D (ps)
2	100
3	294
4	438
5	562
6	707
7	862

4.3 Routing Architecture

The target routing architecture of the CAD flow used in these experiments is one that Betz et. al [4] indicate is a good choice. This architecture has the following parameters:

- Routing segments have a logical length of four (the logical length of a segment is defined as the number of logic block clusters that it spans)
- 50% of these segments use tri-state buffers as the programmable switch and 50% use pass transistors

The experiments conducted in [4] were based on a LUT size of four and a cluster size of four. We will assume these results are valid for all the LUT sizes and cluster sizes that we are comparing.
However, the LUT and cluster size does affect the sizing of the buffers used to drive the programmable routing, both from the block itself and the tri-state buffers internal to the programmable routing. As the logic block cluster increases in size, the size of each logic tile is larger, and therefore the length of the wires being driven by each buffer increases. Since this increases the capacitive loading of each wire, the buffers must be sized appropriately. Betz [4] indicates that for a cluster size of four and a LUT size of four, the best routing pass transistor width was ten times the minimum width, while the best tri-state buffer size was only five times the minimum. We size our buffers in direct proportion to the length of this tile. That is, if the tile length has doubled, then we double the size of the routing buffers.

5. EXPERIMENTAL RESULTS

In this section we present the experimental results of synthesizing benchmark circuits through the CAD flow described in Section 3 with the area delay modeled as described in Section 4. The benchmark circuits used in these experiments were the twenty largest from MCNC [24]. Table 3 gives a description of the circuits, including the name, number of 4 input-LUTs and number of nets.
Each circuit was mapped, placed and routed with LUT size varying from 2 to 7 and cluster sizes from 1 to 10. With 6 different LUT sizes and 10 different cluster sizes this gives a total of 60 distinct architectures.

5.1 Cluster Inputs Required vs. LUT and Cluster Size

Before answering the principal questions raised in the introduction, we need to determine an appropriate value for I, the number of logic block cluster inputs (see Section 2 for a definition of I). The value of I should be a function of K (the LUT size) and N (the number of LUTs in a cluster). This is of concern since the larger the number of inputs the larger and slower the multiplexers feeding the LUT inputs

Table 3: MCNC Benchmark Circuit Descriptions

Circuit	# of 4-Input BLEs	Number of Nets
alu4	1522	1536
apex2	1878	1916
apex4	1262	1271
bigkey	1707	1936
clma	8383	8445
des	1591	1847
diffeq	1497	1561
dsip	1370	1599
elliptic	3604	3735
ex1010	4598	4608
ex5p	1064	1072
frisc	3556	3576
misex3	1397	1411
pdc	4575	4591
s298	1931	1935
s38417	6406	6435
s38584.1	6447	6485
seq	1750	1791
spla	3690	3706
tseng	1047	1099

will be, and more programmable switches will be needed to connect externally to the logic block. Indeed, one of the principal advantages of fully-connected clusters is that they require fewer than the full number of inputs (K $\times$ N) to achieve high logic utilization. There are several reasons for this:

- Some of the inputs are feedbacks from the outputs of LUTs within the same clusters, saving inputs.
- Some inputs are shared by multiple LUTs in the cluster
- Some of the LUTs do not require all of their K-inputs to be used. Indeed this is often the case, as pointed out in [11].

Betz and Rose [2] [3] showed that when K=4 and I is set to the value 2N+2, then 98% of all of the 4-LUTs in a cluster would typically be used. We would like to find a similar relation, but one that includes the variable K.

To determine this relation, we ran several experiments, using only the first three steps illustrated in Figure 3: logic synthesis, technology mapping and packing. For each possible value of N and K, we ran experiments varying the value of I (the maximum number of inputs to the cluster allowed by the packer) from 1 to K $\times$ N. Following [2] we chose the lowest value of I that provided 98% utilization of all of the BLEs present in the circuit. Figure 6 is a plot of the relationship between the number of inputs (I) required to achieve 98% utilization and the cluster size (N) and the LUT size (K). Typically, the value of I must be between 50 and 60% of the total possible BLE inputs, I = K $\times$N.

By inspection we have generalized the relationship as:

$$I = \frac{K}{2} \times (N+1)$$

This equation provides a close fit to the results in Figure 6. The average percentage error across all possible data points is only 10.1 % with a standard deviation of 7.6 %.

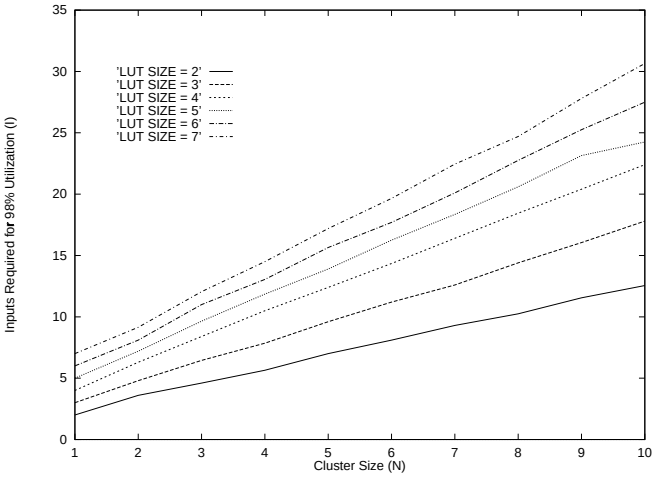

Figure 6: Number of Inputs Required for 98% Logic Block Utilization

5.2 Area as a Function of N and K

In this section we present and discuss the experimental results that show the area of an FPGA as a function of N and K. Note that I was set to the value determined in the previous section. These results are for the 20 benchmark circuits. Area, as discussed above, is measured in terms of the total number of minimum-width transistors required to implement all of the logic and routing.

5.2.1 Total Area

Figures 7 and 8 give a plot of the geometric average (across all 20 circuits) of the total area required as a function of cluster size and LUT size. Several observations can be made from this data:

- For clusters of size 1, LUT sizes of 4, 5 and 6 are the most area-efficient.
- For larger clusters ($N > 3$) the area-efficiency differences between LUT sizes is not very large, with the exception of 7-input LUTs. The data shows that all these areas are within about 10 % of each other. This includes LUTs of size 2, which is somewhat surprising. It appears that the clustering of 2-input LUTs ameliorates the usual high wiring requirements of fine-grain blocks, as is apparent with cluster size N=1 and K=2.

It is instructive to break out the components of the data in Figures 7 and 8 in order to achieve both insight and inspiration on how to make more area-efficient FPGAs. The total area can be broken into two parts, the logic block area (including the muxes inside the clusters) and the routing area, which is the programmable routing external to the clusters. Throughout the rest of this paper, these will be referred to as the intra-cluster area and inter-cluster area respectively.

We will first explore the intra-cluster area. Figure 9 shows the total intra-cluster area component of the total area (again, geometrically averaged over the 20 circuits) as a function of the LUT size. The data shows that the intra-cluster area increases as K increases. This area is the product of the total number of clusters times the area per cluster. A plot of these two components for a cluster size of 1 is given in Figure 10.

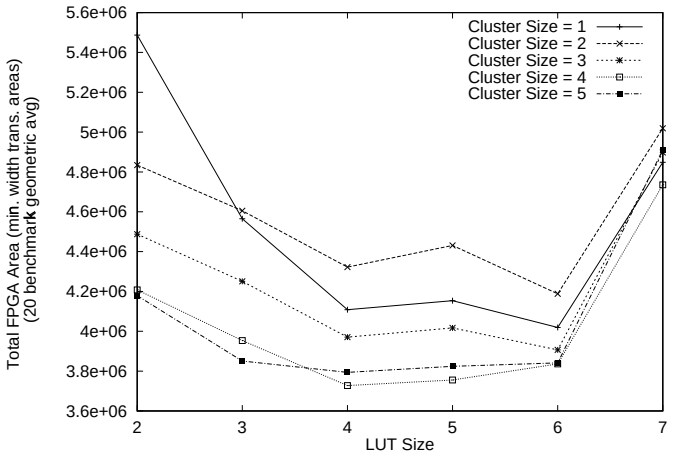

Figure 7: Total Area for Clusters of Size 1 to 5

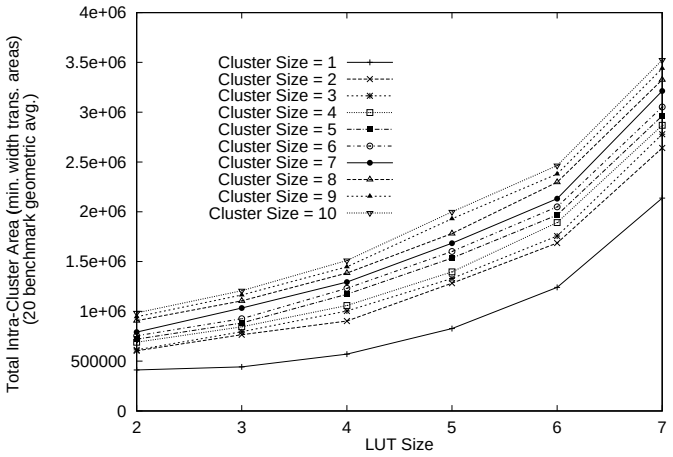

Figure 9: Total Logic Block Cluster Area

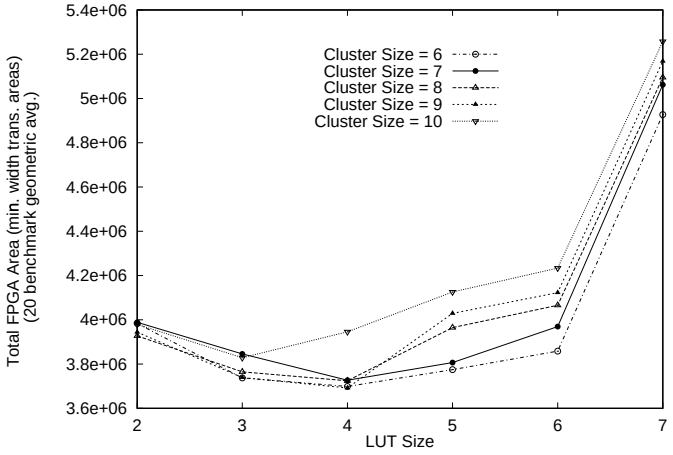

Figure 8: Total Area for Clusters of Size 6 to 10

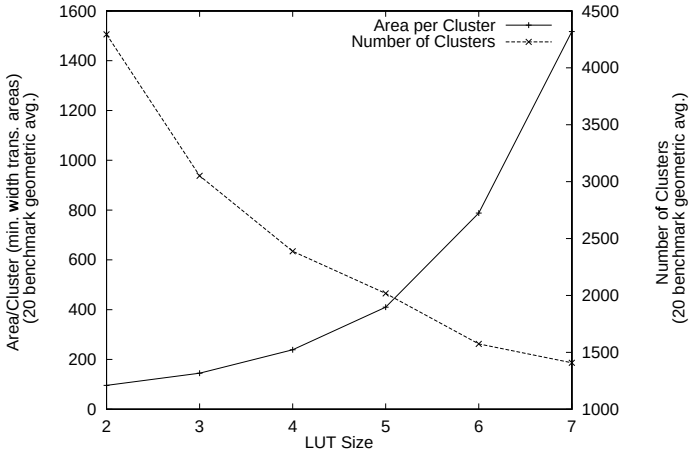

Figure 10: Number of Clusters and Cluster Area Versus K (for N=1)

The logic block area grows exponentially with LUT size as there are 2^K bits in a K-input LUT. In addition, larger LUT sizes require larger intra-cluster multiplexers because the size of each multiplexer is (I + N) = (K/2 (N+1) + N). As K increases, though, the number of clusters decreases (because each LUT can implement more of the logic function) as shown by the downward curve in Figure 10. However, the rate of decrease in the number of logic blocks is far outweighed by the increase in the size of the block as K increases, and hence the upward trend in Figure 9.

Observing the absolute values in Figures 9 and 7, we see that the intra-cluster area typically takes up about only 25% to 35% of the total area, except when the LUT size reaches 6 and 7, at which point intra-cluster area becomes a dominant factor.

The key effect, as always in FPGAs, is with the routing area. Figure 11 is a plot of the total inter-cluster routing area as a function of the LUT size and cluster size. The Figure shows that the routing area decreases in a linear fashion with increasing LUT size. This particular result is interesting since previous work from [18] has shown that the routing area achieved a minimum between K=3 and K=4, and increased for values of K beyond this.

To explain this observed behavior, observe Figure 12 which decomposes the total routing area into two separate components: the number of clusters and the (external) routing area per cluster. These curves are given for a cluster size of 1, but are representative for all cluster sizes. The product of these two curves gives the total inter-cluster routing area. The reason why the routing area decreases linearly with LUT size is that as we increase the LUT size, the number of clusters decreases much faster than the rate at which the routing area per cluster increases. The difference in results from [18] and our current results can be attributed to the fact that we are now using better CAD tools with more sophisticated algorithms; in particular the quality of the placement tool and the routing tool is significantly better, and uses significantly less wiring. In addition, for clustered logic blocks, more of the routing is being implemented within the cluster itself.

5.3 Performance as a Function of N and K

The second key metric for FPGAs is critical path delay, or performance. The total critical path delay is defined as the total delay due

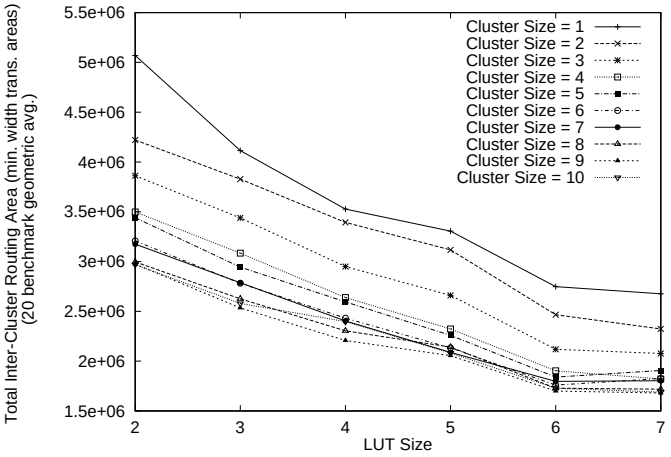

Figure 11: Routing Area

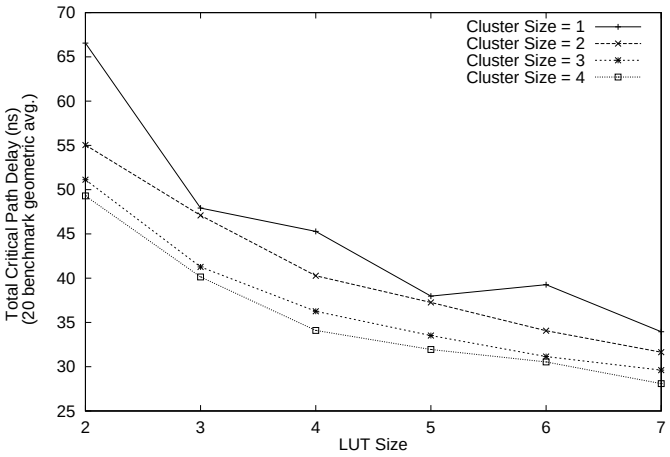

Figure 13: Total Delay for Clusters of Size 1 to 4

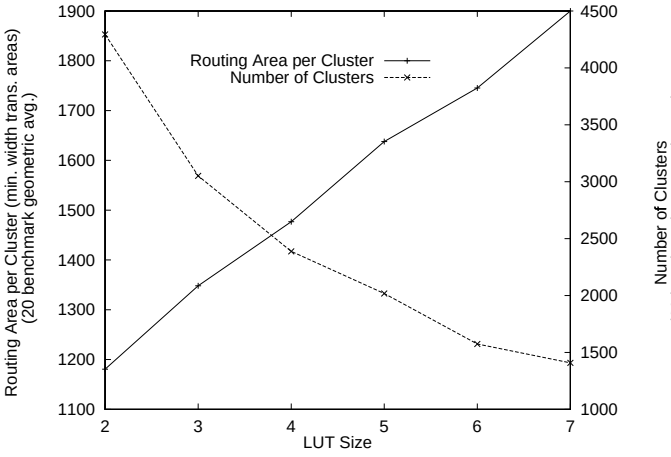

Figure 12: Number of Clusters and Routing Area Per Cluster Versus K (for N=1)

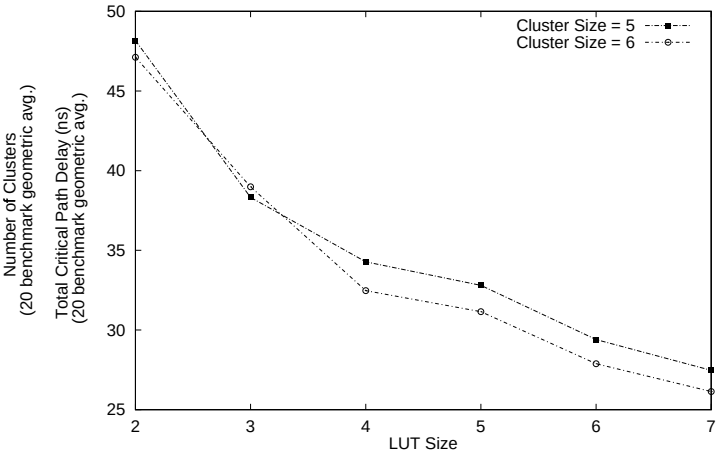

Figure 14: Total Delay for Clusters of Size 5 and 6

to the logic cluster combined with the routing delay. Figures 13 to 15 show the geometric average of the total critical path delay across all 20 circuits as a function of the cluster size and LUT size. Observing the Figures, it is clear that increasing N or K decreases the critical path delay. These decreases are significant: an architecture with N=1 and K=2 has an average delay of 66 ns while K=7 and N=10 has an average critical path delay of just 26 ns. There are two trends that explain this behavior. As the LUT and cluster size increases:

- the delay of the LUT and the delay through a cluster increases
- the number of LUTs and clusters in series on the critical path decreases

We will discuss these effects in more detail below.

It is instructive to break the total delay into two components: intra-cluster delay (which includes the delay of the muxes and LUTs), and inter-cluster delay.

Figure 16 shows the portion of the critical path delay that comes from the intra-cluster delay as a function of K and N. For N=1, the total intra-cluster delay increases with K while for all other cluster size the delay decreases. The reason is that for the non-clustered architecture there are no input multiplexers feeding the LUT inputs. Hence, the percentage increase for any subsequent LUT size increment will appear to be much greater than for the clustered architecture where there are input multiplexers. As we'll see below, though, this increasing effect is outweighed by the decrease in the inter-cluster routing delay.

Notice that the portion of the delay within the cluster increases as the cluster size increases. This is because the intra-cluster muxes get larger and therefore slower. However, the delay through these muxes is still much faster than the inter-cluster delay, as shown below.

Figures 17 and 18 show the portion of the critical path delay that comes from the inter-cluster routing delay as a function of K and N. As K increases there are fewer LUTs on the critical path, and this translates into fewer inter-cluster routing links, thus decreasing the inter-cluster routing delay. Similarly, as N is increased, more connections are captured within a cluster, and again, the inter-cluster

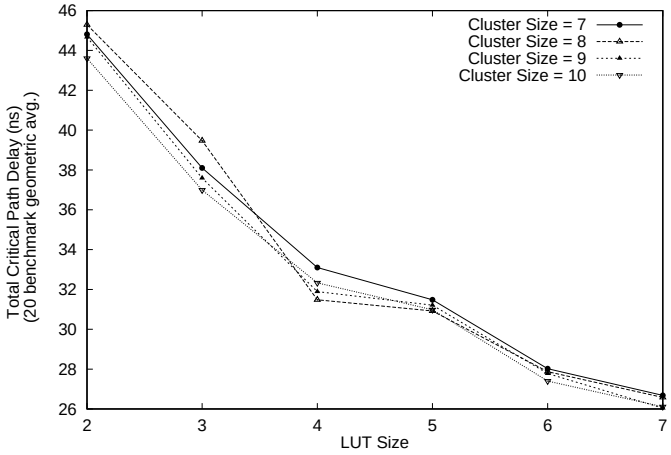

Figure 15: Total Delay for Clusters of Size 7 to 10

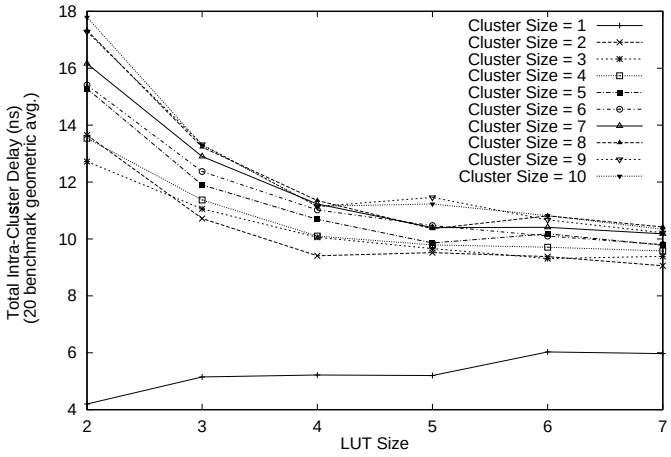

Figure 16: Total Intra-Cluster Delay for Clusters of Sizes 1 to 10

routing delay decreases.

In discussing these trade-offs, it's useful to work an explicit example: Table 4 shows how the delay through one BLE and multiplexer stage (delay from B to D on Figure 5) doubles from 0.578 ns to 1.087 ns when going from K=4 and N=1 to K=4 and N=2. Although the number of levels on the critical path remains fairly constant since we have not modified K, the total logic delay increases from 4.58 ns to 8.77 ns. However, since there are now 2 BLEs in every cluster as opposed to a single BLE, more logic is implemented internally within the clusters. Nets that normally would have been routed externally are now internal to the clusters. This translates in a reduction in the routing delay from 39.5 ns when using K=4, N=1 to 30.4 ns for K=4 and N=2. The total critical path delay decreases from 44.1 ns to 39.2 ns as originally shown in Figure 13.

In general, inter-cluster routing delay is much larger than the intra-cluster delay, and hence the value of increasing the cluster or LUT size.

Figure 19 illustrates this concept at the BLE level: it is a plot of BLE delay and number of BLEs on the critical path versus LUT size for a cluster size of 1. The BLE delay increases with K, but the number of levels decreases. For N=1, the BLE delay increases much faster than the decrease in the number of levels and hence the increase in the logic delay from figure 16. For all other cluster sizes, the number of levels decreases quicker than the increase in BLE delay.

Table 4: Critical Path Delay Comparison for K=4

	N=1	N=2
BLE + Mux Delay	0.578 ns	1.087 ns
Avg # of BLEs on Critical Path	7.94	8.07
Total Intra-Cluster Delay	4.58 ns	8.77 ns
Total Routing Delay	39.5 ns	30.39 ns
Total Delay	44.08 ns	39.16 ns

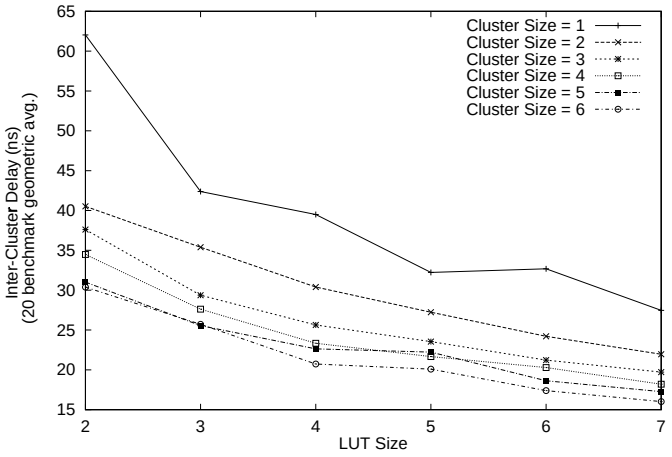

Figure 17: Total Inter-Cluster Delay for Clusters of Size 1 to 6

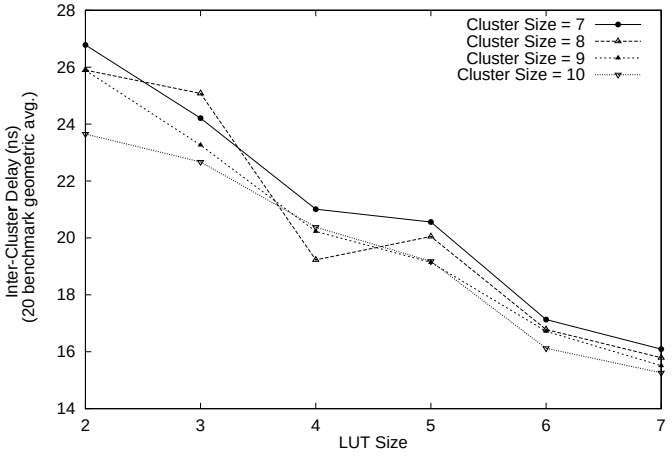

Figure 18: Total Inter-Cluster Delay for Clusters of Size 7 to 10

5.4 Area-Delay Product

So far, we have examined the effect of K and N on area and performance of FPGAs. As area can often be traded for delay, it is in-

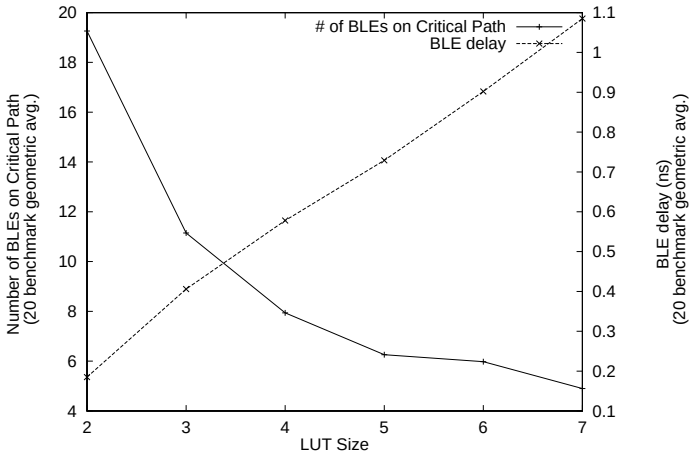

Figure 19: Number of BLEs on Critical Path and BLE delay vs K (for N=1)

structive to look at the area-delay product. Figure 20 displays the area-delay product versus K and N. This plot clearly shows that using a LUT size of between 4 and 6 and clusters of 4 to 10 appear to give the best area-delay results.

Notice that area-delay decreases significantly as the LUT size is increased from 2 to 4. This is because, even though clustered 2-input and 3-input lookup tables achieve good area, their delay is poor, and so they are a bad choice.

The area-delay product jumps for K=7 principally because the huge area cost for 7-input LUT outweighs the modest performance gains it achieves.

This latter observation suggests that, if there was a way to achieve the depth properties of a 7-input LUT without paying the heavy area price, then such a 7-input input function may well be a good choice.

We have also observed that, for large clusters, a large portion of the delay is taken up by the intra-cluster muxes. If this delay could be reduced somehow, then significant speed wins could be achieved.

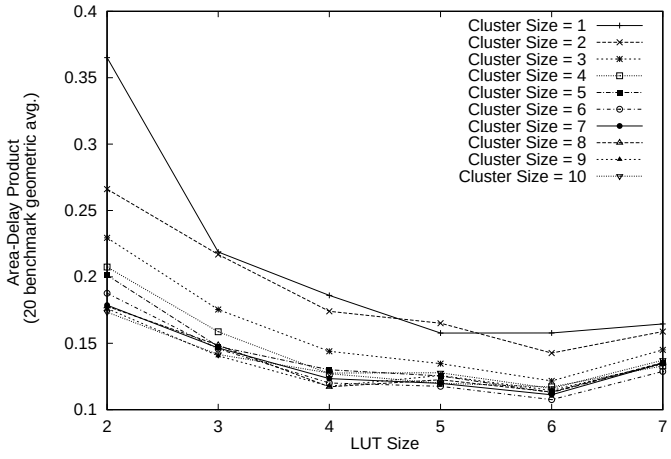

Figure 20: Area-Delay Product for Clusters of Size 1 to 10

6. CONCLUSION

We have studied the effect that different logic block architectures have on FPGA area and performance. The main results are summarized in table 5. In addition, we experimentally derived a relationship between the number of cluster logic block inputs required to achieve 98% utilization as a function of the LUT size, K and the cluster size, N. This is $I = \frac{K}{2} \times (N+1)$, where I is the number of distinct cluster inputs.

Secondly, we have shown that although small LUT sizes may be area efficient in mid-sized ($>$ 3 BLEs) clusters, their performance characteristics are very poor. If area-delay is the main criteria, then the use of clusters of between 4 and 10 and LUT sizes of 4 to 6 will produce the best overall results.

Finally, our work suggests two future directions: finding ways to reduce the number of levels of logic without the expense of large LUTs, and reducing the delay of intra-cluster multiplexers.

Table 5: Summary of Best Area, Delay, and Area-Delay Results

Criteria	LUT Size (K)	Cluster Size (N)
Area	3 to 4	6 to 10
Delay	7	4 to 10
Area-Delay	4 to 6	4 to 10

7. ACKNOWLEDGMENTS

The authors are grateful to Vaughn Betz and Alexander Marquardt for providing the foundation upon which this work is built, and for advice and help throughout this effort.

8. REFERENCES

[1] O. Agrawal, H. Chang, B. Sharpe-Geisler, N. Schmitz, B. Nguyen, J. Wong, G. Tran, F. Fontana and B. Harding, “An Innovative, Segmented High Performance FPGA Family with Variable-Grain-Architecture and Wide-gating Functions”, FPGA’99, Monterey, CA, 1999.

[2] V. Betz and J. Rose, “Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size”, IEEE Custom Integrated Circuits Conference, Santa Clara, CA, 1997, pp. 551-554.

[3] V. Betz and J. Rose, “How Much Logic Should Go in an FPGA Logic Block?”, IEEE Design and Test Magazine, Spring 1998, pp. 10-15.

[4] V. Betz, J. Rose and A. Marquardt, “Architecture and CAD for Deep-Submicron FPGAs”, Kluwer Academic Publishers, 1999.

[5] S. Brown, R. Francis, J. Rose and Z. Vranesic, “Field-Programmable Gate Arrays”, Kluwer Academic Publishers, 1992.

[6] S. Brown and J. Rose, “FPGA and CPLD Architectures: A Tutorial”, IEEE Design & Test of Computers, Summer 1996, pp.42-57.

[7] Kevin Chung, PhD Thesis: “Architecture and Synthesis of Field-Programmable Gate Arrays with Hardwired Connections”, University of Toronto, 1994.

[8] J. Cong and Y. Ding, “FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs”, IEEE Trans. on CAD, Jan. 1994, pp.1-12.

[9] J. Cong and Y. Hwang, “Boolean Matching for Complex PLBs in LUT-based FPGAs with Application to Architecture Evaluation”, FPGA 98, Monterey, CA, 1998.

[10] D. Hill and N-S Woo, “The Benefits of Flexibility in Look-up Table FPGAs”, in FPGAs, W. Moore and W. Luk Eds., Abingdon 1991, edited from the Oxford 1991 International Workshop on FPGAs, pp. 127-136.

[11] S. Kaptanoglu, G. Bakker, A. Kundu and I. Corneillet “A new high density and very low cost reprogrammable FPGA architecture”, FPGA’99, Monterey, CA, 1999.

[12] J. Kouloheris and A.El Gamal, “FPGA Performance vs. Cell Granularity”, Proc. of Custom Integrated Circuits Conference, May 1991, pp. 6.2.1 - 6.2.4.

[13] J. Kouloheris and A.El Gamal, “FPGA Area vs. Cell Granularity - Lookup Tables and PLA Cells”, First ACM Workshop on FPGA’s, FPGA ‘92, Berkeley, CA, February 1992.

[14] J. Kouloheris and A.El Gamal, “FPGA Area vs. Cell Granularity - PLA Cells”, Proc. of Custom Integrated Circuits Conference, May 1992.

[15] A. Marquardt, “M.A.Sc Thesis: Cluster-Based Architecture, Timing-Driven Packing, and Timing-Driven Placement for FPGAs”, University of Toronto, 1999.

[16] A. Marquardt, V. Betz and J. Rose, “Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density”, ACM/SIGDA FPGA 99, 1999.

[17] J. Rose, R.J. Francis, P. Chow and D. Lewis, “The Effect of Logic Block Complexity on Area of Programmable Arrays”, Proc. 1989 Custom Integrated Circuits Conference, May 1989, pp. 5.3.1-5.3.5.

[18] J. Rose, R.J. Francis, D. Lewis and P. Chow, “Architecture of Field-Programmable Gate Arrays: The Effect of Logic Functionality on Area Efficiency”, IEEE Journal of Solid-State Circuits, 1990.

[19] A. Sedra and K. Smith, “Microelectronic Circuits: Third Edition”, Oxford University Press, 1991.

[20] E.M. Sentovich et al, “SIS: A System for Sequential Circuit Analysis”, Tech. Report No. UCB/ERL M92/41, University of California, Berkeley, 1990.

[21] S. Singh, “The Effect of Logic Block Architecture on FPGA Performance”, M.A.Sc. Thesis, University of Toronto, 1991.

[22] S. Singh, J. Rose, P. Chow and D. Lewis, “The Effect of Logic Block Architecture on FPGA Performance”, IEEE Journal of Solid-State Circuits, 1992.

[23] N. West and K. Eshraghian, “Principles of CMOS VLSI Design; A System Perspective; Second Edition”, Addison Wesley, 1993.

[24] S. Yang, “Logic Synthesis and Optimization Benchmarks, Version 3.0”, Tech. Report, Microelectronics Centre of North Carolina, 1991.