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

Elias Ahmed and Jonathan Rose

***Abstract*—In this paper, we revisit the field-programmable gate-array (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 (Betz *et al.* 1997) 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. We use a fully timing-driven experimental flow (Betz *et al.* 1997), (Marquardt, 1999) in which a set of benchmark circuits are synthesized into different cluster-based (Betz and Rose, 1997, 1998) and (Marquardt, 1999) logic block architectures, which contain groups of LUTs and flip-flops. Across all architectures with LUT sizes in the range of 2 to 7 inputs, and cluster size from 1 to 10 LUTs, 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 clustering small LUTs (sizes 2 and 3) produces better area results than what was presented in the past. 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 3–10 provides the best area-delay product for an FPGA.**

***Index Terms*—Architecture, clusters, computer-aided design (CAD), field-programmable gate-array (FPGA), look-up table (LUT), very large scale integration (VLSI).**

## I. Introduction

SEVERAL studies in the past have examined the effect of logic block functionality on the area and performance of field-programmable gate-arrays (FPGAs). The work in [15] and [23], showed that a look-up table (LUT) size of 4 is the most area efficient in a nonclustered context. In addition, it was demonstrated in [13], [25], and [27] that using a LUT size of 5 to 6 gave the best performance. The work in [12] 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 three-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.

Although these questions were addressed some time ago in [11], [13], [14], [22], [23], and [27], several reasons compelled us to revisit the issue. First, prior work focused on nonclustered logic blocks, which are known to have a significant impact on the area and delay [21]. 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 transistor-level design of circuits and perform appropriate buffer and transistor sizing for all the logic and routing elements, in the manner of [6]. Fourth, the computer-aided design (CAD) tools available today for experimentation are significantly better than those available over a decade 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. In addition to these reasons, we believe that careful analysis in this kind of study, may well lead to suggestions for better architectures. The work in [7] and [8] provides a more detailed general background and overview of the issues affecting FPGA architectures.

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 [4], [5] is group of basic logic elements (BLEs) that are fully connected by a mux-based cross bar as illustrated in Fig. 2. The Altera Flex 6 K, 8 K, 10 K, 20 K, Stratix, Cyclone and Xilinx 5200, Virtex and VirtexII are commercial examples of such clusters (although not all of these are 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 [23] and the size of the cluster is quadratic in $N$ [4]. Furthermore, the area devoted to routing outside the block will change as a function of $K$ and $N$, and this effect (since routing area

Manuscript received December 10, 2002; revised May 25, 2003.

E. Ahmed was with the Department of Electrical and Computer Engineering (ECE), University of Toronto, Toronto, ON M5S 3G4, Canada and is now with Altera Toronto Technology Center, Toronto, ON, Canada (e-mail: eahmed@altera.com).

J. Rose is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: jayar@eecg.toronto.edu).

Digital Object Identifier 10.1109/TVLSI.2004.824300

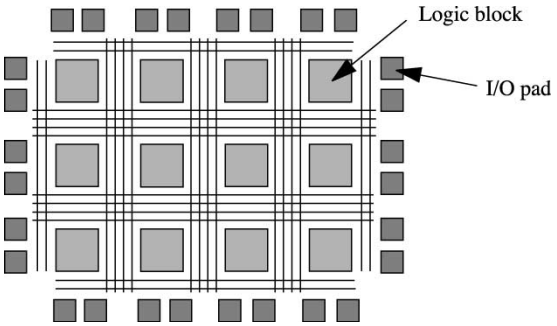


Fig. 1. Island-style FPGA [6].

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 tradeoffs, we seek to answer the following questions.

1) An expensive part of a clustered architecture is the number of inputs to the cluster, which we will refer to as $I$. 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 $I = K \times N$ will do this, but a value less than this, which is cheaper, may also suffice.)
2) What is the effect of $K$and $N$ on FPGA area?
3) What is the effect of $K$and $N$ on FPGA delay?
4) Which values of $K$and $N$ give the best area-delay product?

More crucially, we would like to clearly explain the results, which may give rise to insights that lead to better architectures.

An earlier version of this work appeared in [2]. That work targeted a 0.35-$\mu$m CMOS process whereas this work is based on a 0.18-$\mu$m process. The work presented here also includes more extensive analysis of the results as well as incorporating new benchmark circuits. This paper is outlined as follows. Section II 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 III details the experimental CAD flow and steps that were performed to produce the results. Section IV describes the logic and routing architectures, and some details of the area and delay modeling. Section V presents the results from these experiments. Finally, we conclude in Section VII.

## II. 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 Fig. 1. The I/O pads are evenly distributed around the perimeter of the FPGA.

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

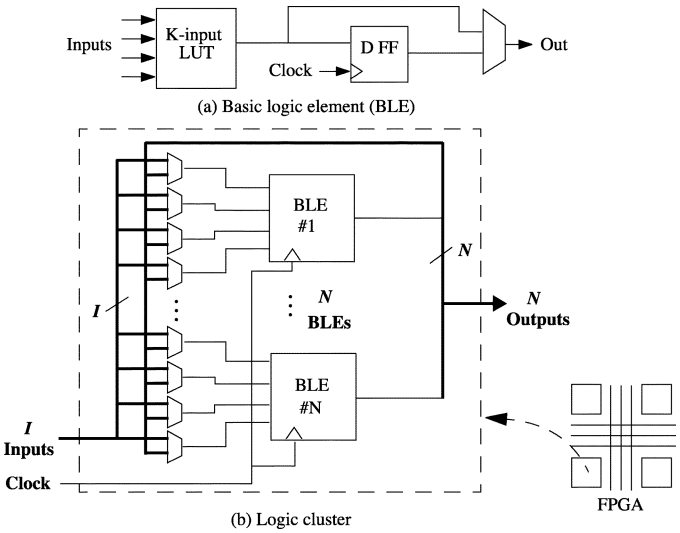


Fig. 2. (a) Structure of the basic logic element (BLE) and (b) logic cluster assuming a LUT size ($K$) of 4 [6].

For clusters containing more than one BLE, we assume a "fully connected" [4] 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 Fig. 2(b), which are not necessary for clusters of size $N = 1$.[1] It should be mentioned that a fully connected logic cluster is not the only approach, and recent work [16], [17] has explored the effects of a depopulated cluster. As stated earlier, the Stratix FPGA [18], as well as Virtex and Virtex II, are commercial examples of architectures which employ depopulated logic clusters.

## III. 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 [6], [7], [13]. Fig. 3 illustrates the CAD flow that we employ. First, each circuit passes through technology independent logic optimization using the SIS program [24]. It is worth noting that, from this point on, the entire CAD flow is fully timing driven. Timing-driven tools are necessary because our goal is to make conclusions about the speed performance of the architectures we will explore. Technology mapping (which converts the logic expressions into a netlist of $K$-input LUTs), was performed using the FlowMap and FlowPack tools [10]. Then, all the registers and LUTs were packed into logic clusters using the timing driven packing algorithm (T-VPACK) [21]. This was followed by timing-driven placement using a timing-enhanced version [21] of VPR [6]. Then full path-based and timing-driven routing is performed using VPR [6].

In our approach to modeling the area of an FPGA required by any given circuit, we determine the minimum number of

[1]In this case, every LUT input must be accessible to the channel. Note that Fig. 2 does not show the channel tracks to cluster input multiplexers—these can be seen in Fig. 5.

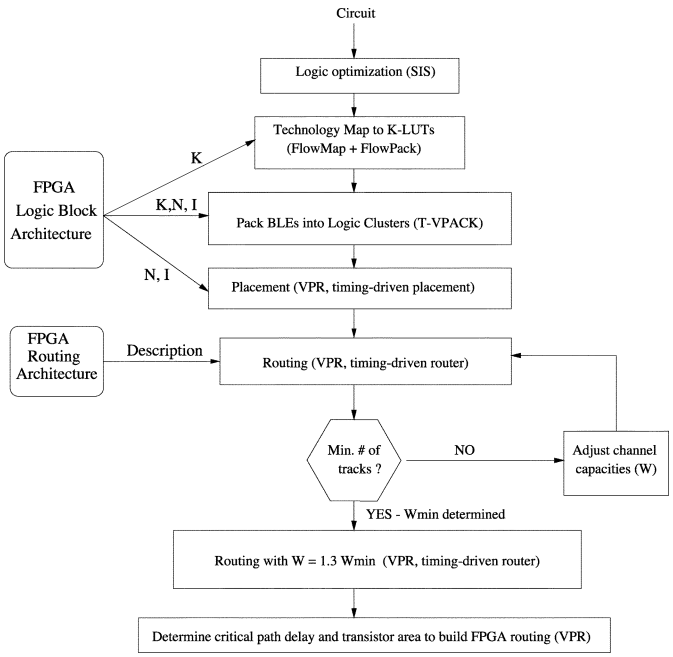


Fig. 3. Architecture evaluation flow.

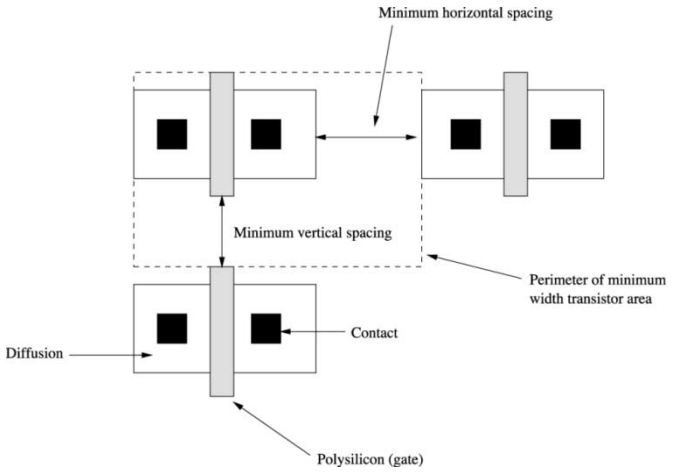


Fig. 4. Definition of a minimum-width transistor area [6].

tracks needed to successfully route each circuit, $W_{\min}$. Clearly this is not possible in real FPGAs, but we believe this is meaningful as part of a logic density metric for an architecture because it measures the routing demand of each circuit in each architecture. The area model, which makes use of this minimum track count, is described more fully in Section IV. 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 do not 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.

## IV. FPGA Architecture Modeling

In this section, we give a brief description of the area and delay modeling developed by Betz *et al.* [6]. The level of detail present in these models goes far beyond any modeling previously used in this kind of experimental analysis. Research done previous to Betz *et al.* [6] has modeled the area by parameterizing the size of wires [23] or counting routed pins [11] on the logic blocks, and then guessing scaling factors between logic and routing area. Prior research has modeled delay by counting the number of programmable switches present in a path, or by using a simple unit delay model for each level of LUT [27]. In this paper, all device parameters and circuits are modeled using SPICE simulations of a 0.18-$\mu$m CMOS process. We make the following assumptions about the basic island-style architecture.

1) The number of routing tracks in each channel between logic blocks is uniform throughout the FPGA.
2) All wire parameters were derived assuming minimum width and spacing on metal layer 3.
3) Each circuit is mapped into the smallest square $(M \times M)$ 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 $M \times M$ 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 $29 \times 29$ FPGA grid which results in 841 blocks. We use the area of the logic and routing surrounding 800 clusters as opposed to 841.

### A. Area Model

The area modeling procedure used by Betz *et al.* [6] 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, intracluster muxes, intercluster routing muxes and switches, and all of the associated programming bits. The basic assumption was that the total area of the FPGA was determined by the transistors, which tends to be true when there are many layers of metal. Two commercial programmable logic device (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 *et al.* use the number of "minimum-width transistor areas" as the 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 Fig. 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,

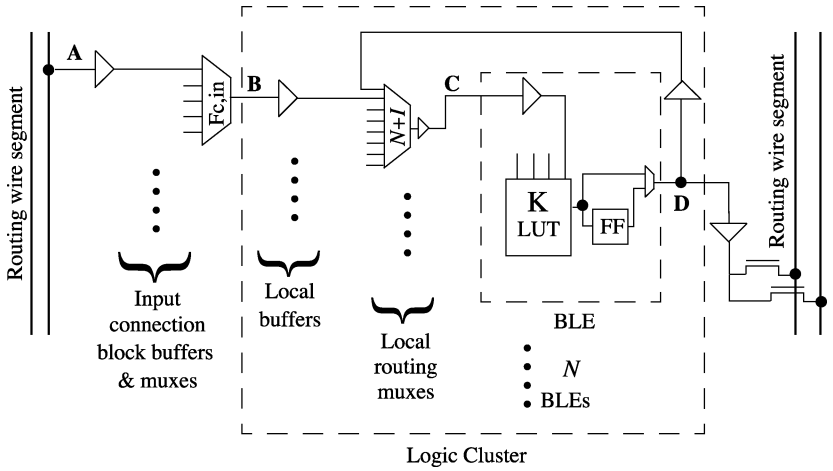


Fig. 5. Structure and speed paths of a logic cluster [6].

TABLE I
LOGIC CLUSTER DELAYS FOR FOUR-INPUT LUT USING 0.18-$\mu$m 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) | 377 | 180 | 376 | 556 |
| 2 | 377 | 221 | 385 | 606 |
| 4 | 377 | 301 | 401 | 702 |
| 6 | 377 | 332 | 397 | 729 |
| 8 | 377 | 331 | 396 | 727 |
| 10 | 377 | 337 | 387 | 724 |

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.

### B. 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. Fig. 5 shows the detailed logic block circuit. The timing values given are based on SPICE simulations of a 0.18-$\mu$m 1.8-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 Fig. 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 I which gives the logic delays as the cluster size increases for the paths indicated in Fig. 5 for a BLE based on a four-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 the tree of pass-transistors. Table II presents the LUT delay as a function of the LUT size.[2]

[2]The small variations in BLE delay (from C to D) is due to the fact that the buffers for each LUT and cluster size have been sized differently to optimize the delay for each given architecture. The result is that a small amount of noise is introduced in the SPICE simulations for each LUT.

TABLE II
LUT DELAYS USING 0.18-$\mu$m CMOS PROCESS

| LUT Size (K) | C to D (ps) |
|---|---|
| 2 | 199 |
| 3 | 283 |
| 4 | 401 |
| 5 | 534 |
| 6 | 662 |
| 7 | 816 |

### C. Routing Architecture

The target routing architecture of the CAD flow used in these experiments is one that Betz *et al.* [6] indicate is a good choice. This architecture has the following parameters.

1) 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).
2) 50% of these segments use tri-state buffers as the programmable switch and 50% use pass transistors.

The experiments conducted in [6], which were used to determine these routing architecture settings were based on logic clusters of size $N = 4$ and LUT size $K = 4$. We will assume that the above architecture choices are reasonable for all of the values of $K$ and $N$ were presented in this paper. It is possible that this assumption is not correct, but time and space constraints limit the extent of the architectural space we can explore.

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,

TABLE III
MCNC BENCHMARK CIRCUIT DESCRIPTIONS

| Circuit | # of 4-Input BLEs | # 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 |
| display_chip | 1794 | 2419 |
| img_calc | 10141 | 10180 |
| img_interp | 2727 | 2769 |
| input_chip | 807 | 841 |
| peak_chip | 809 | 840 |
| scale125_chip | 2632 | 2654 |
| scale2_chip | 1189 | 1202 |
| warping | 1353 | 1394 |

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 *et al.* [6] indicate 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.

## V. EXPERIMENTAL RESULTS

In this section, we present the experimental results of synthesizing benchmark circuits through the CAD flow described in Section III with the area delay modeled as described in Section IV. The benchmark circuits used in these experiments were the twenty largest from MCNC [30] along with eight new benchmarks.[3] Table III gives a description of the circuits, including the name, number of four-input LUTs, and number of nets.[4]

Each circuit was mapped, placed, and routed with the LUT size varying from 2 to 7 and cluster sizes from 1 to 10. With six different LUT sizes and ten different cluster sizes this gives a total of 60 distinct architectures.

### A. Cluster Inputs Required Versus 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 II 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 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.

1) Some of the inputs are feedbacks from the outputs of LUTs within the same clusters, saving inputs.
2) Some inputs are shared by multiple LUTs in the cluster
3) 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 [12].

Betz *et al.* [4], [5] 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. Other unpublished work has shown that the value of 98% is a reasonable choice for total area efficiency. Higher utilization values require more inputs that increase the area of the cluster without a commensurate decrease in total number of clusters. Lower values increase the number of clusters and the total area. 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 Fig. 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 [4], we chose the lowest value of $I$ that provided 98% utilization of all of the BLEs present in the circuit. Fig. 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

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

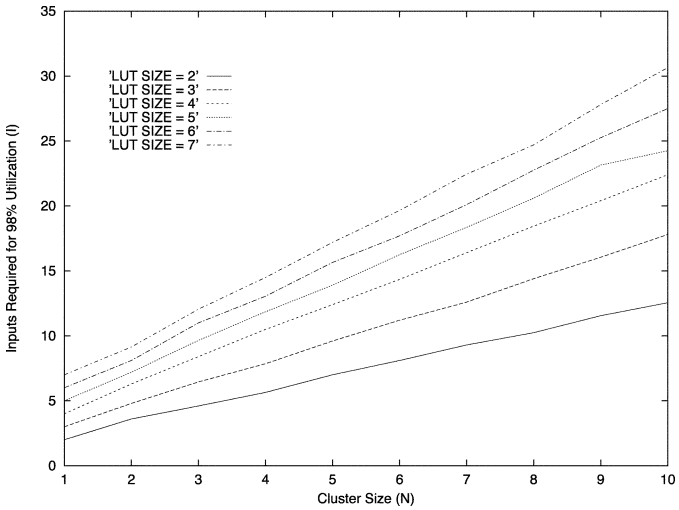


Fig. 6. Number of inputs required for 98% logic block utilization.

[3]The eight new benchmarks are from the University of Toronto, Toronto, ON, Canada, from two computer vision applications. The eight benchmarks are {display_hip, img_alc, img_nterp, input_hip, peak_hip, scale125_hip, scale2_hip, warping}.

[4]Nets are electrically equivalent signals within a circuit.

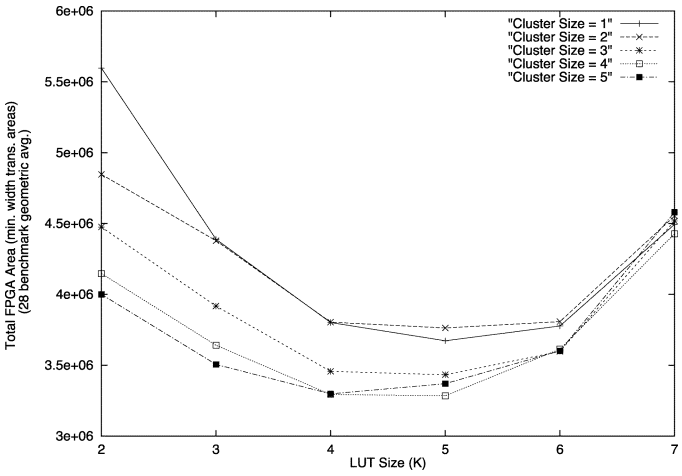


Fig. 7. Total area for clusters of size 1–5.

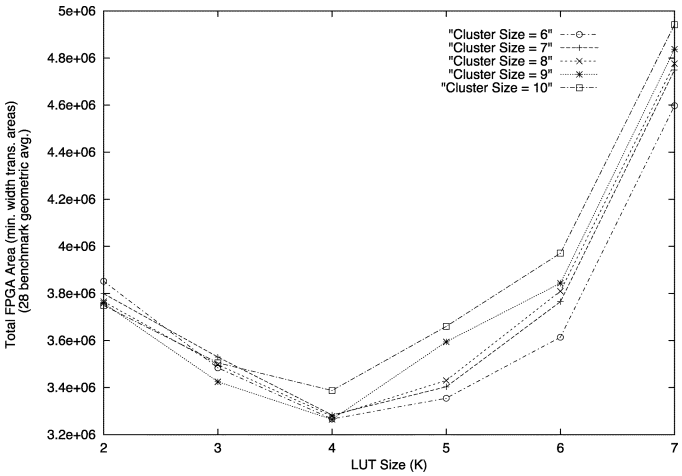


Fig. 8. Total area for clusters of size 6–10.

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

### *B. 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 (1). These results are for the 28 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.

*1) Total Area:* Figs. 7 and 8 give a plot of the geometric average (across all 28 circuits) of the total area required as a function of cluster size and LUT size. Several observations can be made from this data:

- LUT sizes of 4 and 5 are the most area efficient for all cluster sizes.
- There is a reduction in total area as the cluster size is increased from 1 to 3 for all LUT sizes. However, as clusters are made larger $(N > 4)$ there is very little impact on total FPGA area.

Fig. 8 illustrates this last observation very well as all the curves are quite close to each other for any given LUT size. Increasing cluster size results in more BLEs being added to a cluster and connections that normally would have been routed externally are now absorbed internal to the cluster. This reduces the intercluster area which is usually much larger than the intracluster area and thus has a positive impact on total area. However, the reason the total FPGA area does not decrease is because increasing logic capacity within a cluster generally translates into a direct increase in the number of input and output pins on the cluster. The effect of this is an increase in track count. It must also be remembered that the logic block area is also increasing due to the LUT area and the multiplexer area.

It is instructive to break out the components of the data in Figs. 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 intracluster area and intercluster area respectively.

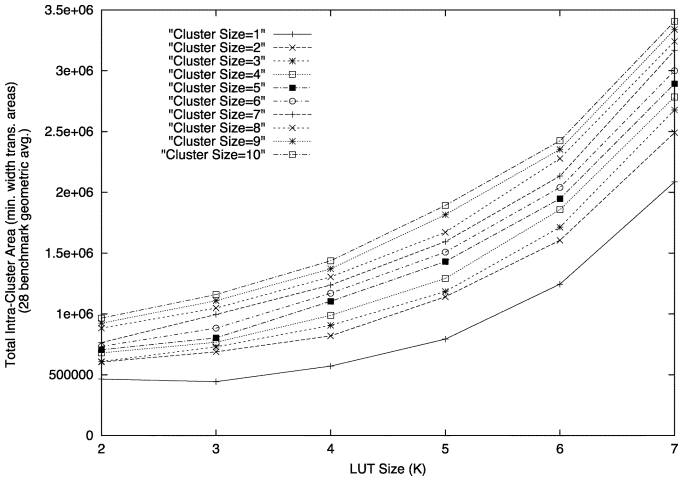


Fig. 9. Total logic block cluster area.

We will first explore the intracluster area. Fig. 9 shows the total intracluster area component of the total area (again, geometrically averaged over the 28 circuits) as a function of the LUT size. The data shows that the intracluster 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 Fig. 10.

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 intracluster 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 Fig. 10. However, the rate of decrease in the number of clusters is far outweighed by the increase in the size of the cluster as $K$ increases, and hence the upward trend in Fig. 9.

Fig. 11 decomposes the intracluster area into two parts: a) intracluster multiplexer area and b) LUT area. The results illustrate that the local intracluster routing area cannot be ignored

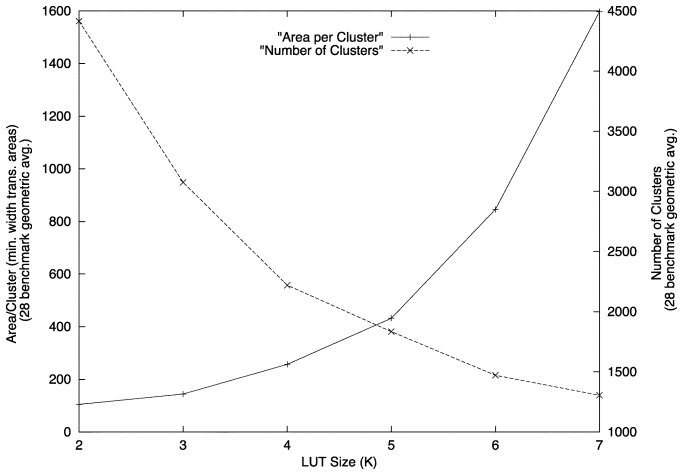


Fig. 10. Number of clusters and cluster area versus $K$ (for $N = 1$).

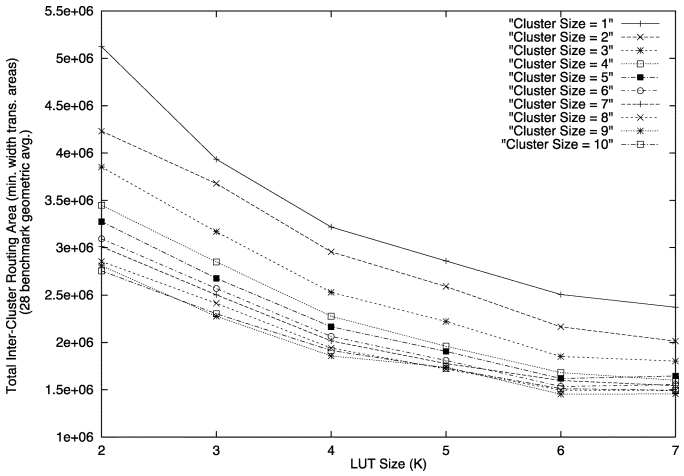


Fig. 12. Intercluster routing area.

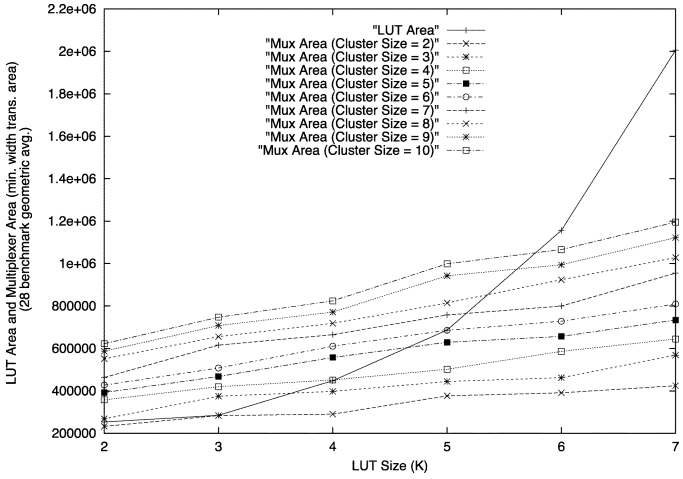


Fig. 11. Intracluster multiplexer area and LUT size.

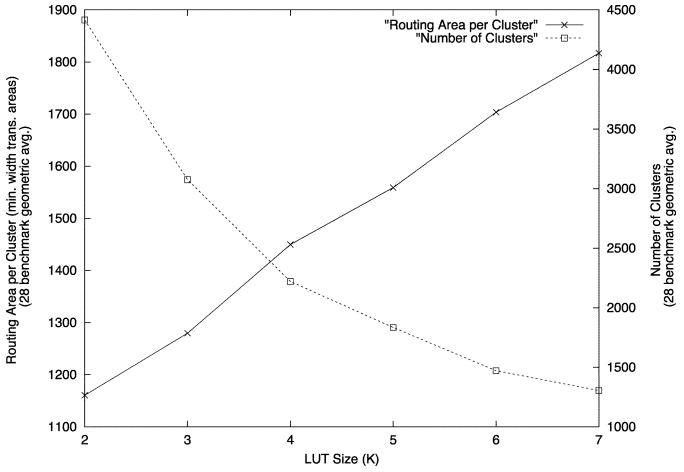


Fig. 13. Number of clusters and routing area per cluster versus $K$ (for $N = 1$).

and can be quite significant for larger clusters. Observing the absolute values in Figs. 7 to 9, we see that the intracluster routing 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 intracluster area becomes a dominant factor.

The key effect, as always in FPGAs, is with the intercluster area. Fig. 12 is a plot of the total intercluster routing area as a function of the LUT size and cluster size, and shows that the intercluster routing area decreases in a linear fashion with increasing LUT size. This particular result is interesting since previous work from [23] 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 Fig. 13 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 intercluster 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 [23] 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.

### C. 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 to the logic cluster combined with the routing delay. Fig. 14 shows the geometric average of the total critical path delay across all 28 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 45 ns while $K = 7$ and $N = 10$ has an average critical path delay of just 14 ns. There are two trends that explain this behavior. As the LUT and cluster size increases

1) the delay of the LUT and the delay through a single cluster increases;
2) the number of LUTs and clusters in series on the critical path decreases.

We will discuss these effects in more detail below.

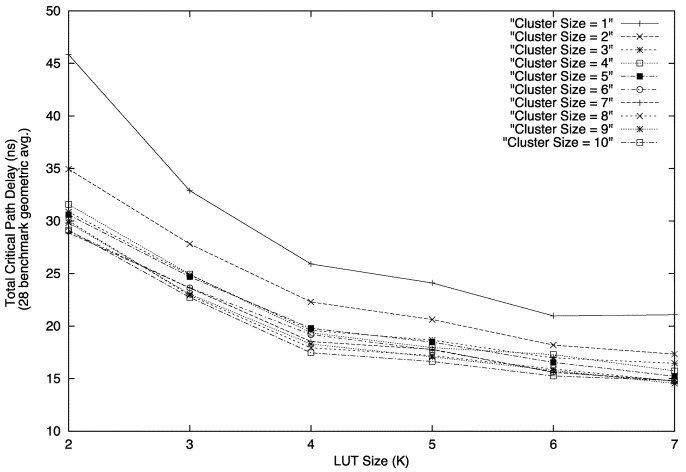


Fig. 14. Total delay for clusters of sizes 1–10.

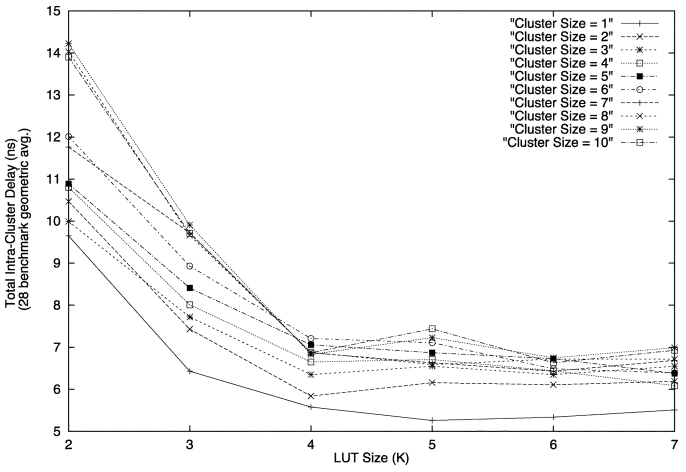


Fig. 15. Total intracluster delay for clusters of sizes 1–10.

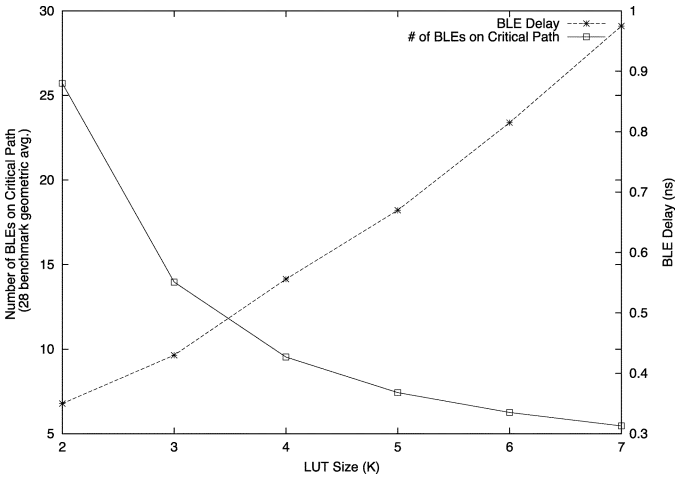


Fig. 16. Number of BLEs on critical path and BLE delay versus $K$ (for $N = 1$).

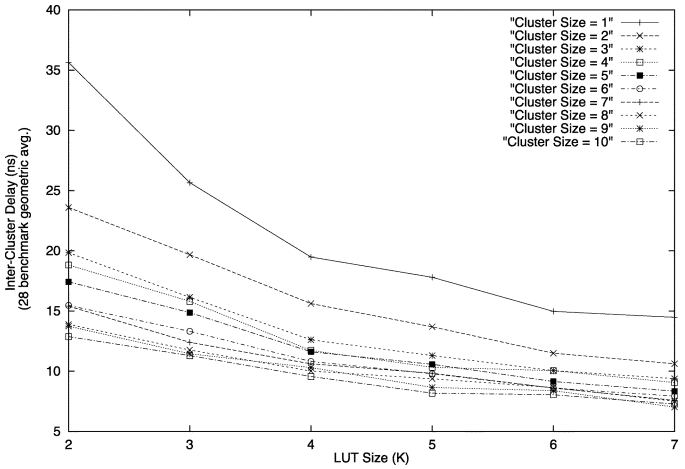


Fig. 17. Total intercluster delay for clusters of sizes 1–10.

It is instructive to break the total delay into two components: intracluster delay (which includes the delay of the muxes and LUTs), and intercluster delay. Fig. 15 shows the portion of the critical path delay that comes from the intracluster delay as a function of $K$ and $N$. There are two key points to observe here. First, the intracluster delay decreases as the LUT size increases. This is due to the fact that there is a reduction in the number of BLE levels on the critical path and hence there will be fewer logic levels to implement. This will translate into a reduction in intracluster delay. Fig. 16 illustrates this concept more clearly 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.[5] The number of BLE levels decreases quicker than the increase in BLE delay and hence the decrease in logic delay. The second behaviour that should be noticed is that the intracluster delay increases for any given LUT size as the cluster size is increased. This is because the intracluster muxes get larger and, therefore, slower. However, the delay through these muxes is still much faster than the intercluster delay, as shown in Fig. 15.

Fig. 17 shows the portion of the critical path delay that comes from the intercluster 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 intercluster routing links, thus decreasing the intercluster routing delay. Similarly, as $N$ is increased, more connections are captured within a cluster, and again, the intercluster routing delay decreases.

[5]The BLE delay is defined as the delay from point B to D in Fig. 5.

TABLE IV
CRITICAL PATH DELAY COMPARISON FOR $K = 4$

| | N=1 | N=4 |
|---|---|---|
| BLE + Mux Delay | 0.556 ns | 0.702 ns |
| Avg # BLEs on Crit. Path | 9.53 | 9.13 |
| Total Intra-Cluster Delay | 5.58 ns | 6.65 ns |
| Total Inter-Cluster Delay | 19.4 ns | 11.7 ns |
| Total Delay | 25.9 ns | 19.4 ns |

In discussing these tradeoffs, it is useful to follow an explicit example: Table IV shows how the delay through one BLE and multiplexer stage (delay from B to D on Fig. 5) rises from 0.556 to 0.702 ns when going from $K = 4$ and $N = 1$ to $K = 4$ and $N = 4$. Although the number of BLE levels on the critical path remains fairly constant since we have not modified $K$, the total logic delay increases from 5.58 to 6.65 ns due to the increase in the local cluster routing multiplexers. However, since there are now four BLEs in every cluster as opposed to a single BLE, more logic is implemented internally within the clusters.

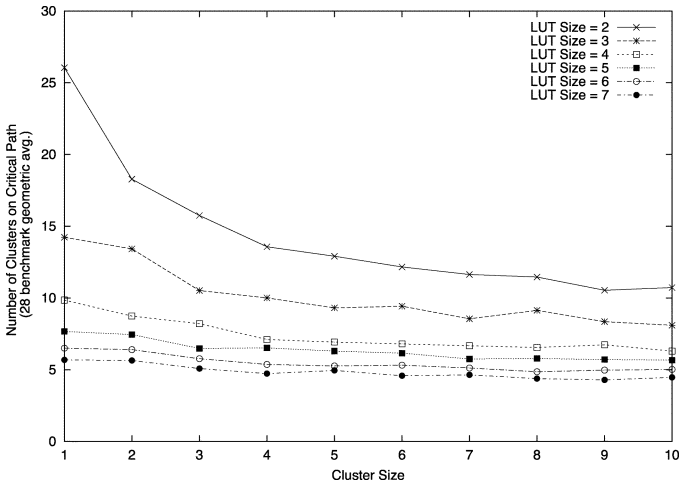


Fig. 18. Number of cluster levels on critical path.

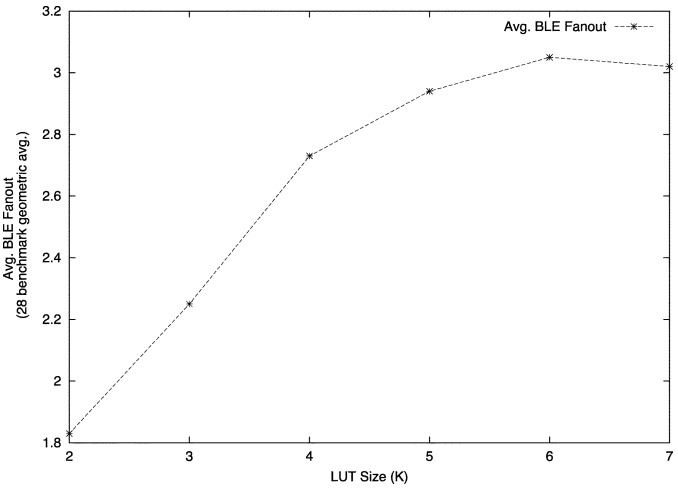


Fig. 19. Average BLE fanout.

Nets that normally would have been routed externally are now internal to the clusters. This translates in a reduction in the intercluster routing delay from 19.4 ns when using $K = 4, N = 1$ to 11.7 ns for $K = 4$ and $N = 4$. The total critical path delay decreases from 25.9 to 19.4 ns as originally shown in Fig. 14.

In general, intercluster routing delay is much larger than the intracluster delay and, hence, the value of increasing the cluster or LUT size. However, it is interesting that increasing cluster size has little impact after a certain point (for $N > 3$). Fig. 14 shows this clearly where for any fixed LUT size, the majority of the improvement in critical path delay occurs as the cluster size is increased from 1 to 3. Any further increases in cluster size results in a very minimum delay improvement. This behaviour suggests that clustering has little effect after a certain point. This is counter intuitive to what we expect. That is, employing larger clusters should always reduce the critical path. Although, the total delay results from Fig. 14 do not contradict this, what was surprising was how little of an improvement in total delay that was achieved with larger clusters. To better understand this situation, it is interesting to examine the number of intercluster "hops" on the critical path. Fig. 18 shows the number of cluster levels as a function of cluster and LUT size. The results clearly show the number of levels decreasing with increasing cluster and LUT sizes. But, for any given LUT size it can be seen that most of the reduction in the number of levels occurs as the cluster size is increased from 1 to 3. Also, recall that the majority of the critical path delay was reduced in this range.

Another interesting trend to observe from Fig. 18, is that increasing the cluster size has less of an effect for architectures composed of larger LUTs. For example, increasing the cluster size from 1 to 10 for a two-input LUT architecture results in a 60% reduction in the number of cluster levels on the critical path. Conversely, employing BLEs with a seven-input LUT and varying the cluster size from 1 to 10 results in only a 22% reduction in logic levels. Hence, clustering proves to be more effective for smaller LUTs. To understand this more clearly, we should examine the average BLE fanout for every LUT size. Fig. 19 shows this and as we can see larger LUTs correlate to larger average fanout. The reason smaller LUTs had a better response to larger cluster sizes was due to the fact that each LUT had a relatively small fanout and hence adding an extra BLE to a cluster usually guaranteed some reduction in the number of logic levels. The same cannot be said about larger LUTs since they have a much larger average block fanout and it becomes much more difficult to ensure that any subsequent BLE addition will result in fewer cluster levels on the critical path.

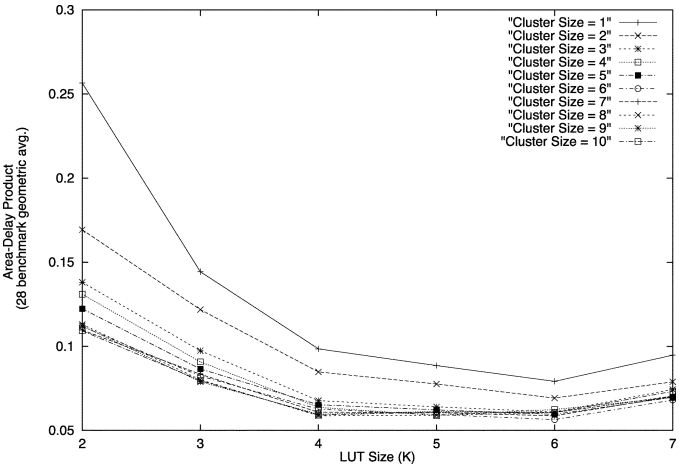


Fig. 20. Area-delay product for clusters of sizes 1–10.

### D. 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 instructive to look at the area-delay product. This is simply the product of the area in minimum width transistor areas times delay in nanoseconds. Fig. 20 displays the area-delay product versus $K$ and $N$. This plot clearly shows that using a LUT size of between 4–6 and clusters of 3–10 appear to give the best area-delay results.

Notice that area-delay decreases significantly as the LUT size is increased from 2 to 4 for all cluster sizes. This is because their delay is poor due to the large amount of BLE levels on the critical path combined with the fact that the total area requirements are slightly larger (by about 20%), and so they are a bad choice. The area-delay product jumps for $K = 7$ principally

TABLE V
SUMMARY OF BEST AREA, DELAY, AND AREA-DELAY RESULTS

| Criteria | LUT Size (K) | Cluster Size (N) |
|---|---|---|
| Area | 4 to 5 | 4 to 9 |
| Delay | 7 | 3 to 10 |
| Area-Delay | 4 to 6 | 3 to 10 |

because the huge area cost for seven-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 seven-input LUT without paying the heavy area price, then such a seven-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 intracluster muxes. If this delay could be reduced somehow, then significant speed wins could be achieved. Also, Fig. 19 suggests that a heterogenous mixture of larger LUTs and smaller LUTs could be beneficial.

### E. Experimental Data

This discussion is only one of many ways to look at a large quantity of data involving 60 architectures, 28 benchmark circuits and three results (area, delay, and area/delay product). The raw data for these experiments can be found in [3].

## VI. CAVEATS ON EMPIRICAL RESULTS

This section outlines some of the key underlying assumptions that have been made in this research that could affect the results. First, the architectural results presented in this paper are dependant on the quality of the CAD tools used in the implementation flow. While one of the tools comes with a nice optimality result (the Flowmap Technology mapper guarantees a depth-optimal LUT mapping) the majority have heuristic algorithms with no guarantees. Secondly, the results depend directly on the nature of the benchmark circuits we employ. We used the commonly-used MCNC [30] and some locally produced benchmarks. It is possible that other circuits from specialized application domains would produce different results. Yan *et al.* [29] showed that some results could vary depending on the tools and benchmarks used. However, some results were shown to be quite consistent across tools and benchmarks. Third, some key routing architecture parameters that were used in our experiments were based on values that were derived from the cluster size $N = 4$ and LUT size $K = 4$ architecture. These routing parameters were then used for all values of $N$ and $K$ in our experiments. It is possible that exploring the space of routing architectures for each of the different and LUT and cluster sizes would produce different results on the relative goodness of these architectures. Time and space constraints prevented this broader exploration.

## VII. CONCLUSION

We have studied the effect that different logic block architectures have on FPGA area and performance. The main results are summarized in Table V. 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 $I = (K/2) \times (N + 1)$, where I is the number of distinct cluster inputs.

Secondly, we have shown that small LUT sizes (two-input and three-input LUTs) are not as area efficient as the four and five-input LUTs and their performance characteristics are very poor. If area delay is the main criteria, then the use of clusters of between 3–10, and LUT sizes of 4–6 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 intracluster multiplexers.

## 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," in *Proc. IEEE Field Programmable Gate Arrays (FPGA)*, Monterey, CA, 1999, pp. 17–26.

[2] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-Submicron FPGA performance and density," in *Proc. IEEE Field Programmable Gate Arrays (FPGA)*, Monterey, CA, 2000, pp. 3–12.

[3] E. Ahmed, "The effect of logic block granularity on deep-submicron FPGA performance and density," M.A.Sc thesis, Univ. Toronto, Toronto, ON, Canada, 2001.

[4] V. Betz and J. Rose, "Cluster-based logic blocks for FPGAs: area-efficiency versus input sharing and size," in *IEEE Custom Integrated Circuits Conf.*, Santa Clara, CA, 1997, pp. 551–554.

[5] ——, "How much logic should go in an FPGA logic block?," *IEEE Des. Test Mag.*, pp. 10–15, Spring 1998.

[6] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, 1999.

[7] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field-Programmable Gate Arrays*. Norwell, MA: Kluwer, 1992.

[8] S. Brown and J. Rose, "FPGA and CPLD architectures: a tutorial," *IEEE Des. Test Comput.*, vol. 13, pp. 42–57, Summer 1996.

[9] K. Chung, "Architecture and synthesis of field-programmable gate arrays with hardwired connections," Ph.D. dissertation, Univ. Toronto, Toronto, Canada, 1994.

[10] J. Cong and Y. Ding, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1–12, Jan. 1994.

[11] D. Hill and N.-S. Woo, "The Benefits of flexibility in look-up table FPGAs," in *Proc. Oxford Int. Workshop FPGAs*, W. Moore and W. Luk, Eds., Abingdon, Oxfordshire, , UK, 1991, pp. 127–136.

[12] S. Kaptanoglu, G. Bakker, A. Kundu, and I. Corneillet, "A new high density and very low cost reprogrammable FPGA architecture," in *IEEE Field Programmable Gate Arrays*, Monterey, CA, 1999, pp. 3–12.

[13] J. Kouloheris and A. El Gamal, "FPGA Performance Versus Cell Granularity," in *Proc. Custom Integrated Circuits Conf.*, May 1991, pp. 6.2.1–6.2.4.

[14] ——, "FPGA area vs. cell granularity—Lookup tables and PLA cells," in *Proc. 1st. ACM Workshop FPGAs*, Berkeley, CA, Feb. 1992, pp. 9–14.

[15] ——, "FPGA area vs. cell granularity—PLA cells," in *Proc. Custom Integrated Circuits Conf.*, May 1992, pp. 4.3.1–4.3.4.

[16] G. Lemieux, "Efficient interconnection network components for programmable logic devices," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Toronto , Toronto, ON, Canada, 2003.

[17] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT clusters," in *Proc. ACM/SIGDA Int. Symp. FPGAs*, Monterey, CA, 2001, pp. 59–68.

[18] D. Lewis *et al.*, "The stratix routing and logic architecture," in *Proc. IEEE FPGAs*, Monterey, CA, 2003, pp. 12–20.

[19] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in *Proc. FPGA*, Monterey, CA, 2003, pp. 175–184.

[20] A. Marquardt, "Cluster-based architecture, timing-driven packing, and timing-driven placement for FPGAs," M.A.Sc thesis, Univ. Toronto, Toronto, ON, Canada, 1999.

[21] A. Marquardt, V. Betz, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," in *Proc. ACM/SIGDA FPGA*, 1999, pp. 37–46.

[22] J. Rose, R. J. Francis, P. Chow, and D. Lewis, "The effect of logic block complexity on area of programmable arrays," in *Proc. Custom Integrated Circuits Conf.*, May 1989, pp. 5.3.1–5.3.5.

[23] 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 J. Solid-State Circuits*, pp. 1217–1225, Oct. 1990.

[24] E. M. Sentovich *et al.*. (1990) SIS: A system for sequential circuit analysis. Univ. Calif., Berkeley, CA. [Online] http://www–cad.eecs.berkeley.edu/~ellen/Html/sis/html Tech. Rep. UCB/ERL M92/41

[25] S. Singh, "The effect of logic block architecture on FPGA performance," M.A.Sc. thesis, Univ. Toronto, Toronto, ON, Canada, 1991.
[26] A. Singh and M. Marek-Sadowska, "Efficient circuit clustering for area and power reduction," in *Proc. FPGA*, Monterey, CA, 2002, pp. 59–66.
[27] S. Singh, J. Rose, P. Chow, and D. Lewis, "The effect of logic block architecture on FPGA performance," *IEEE J. Solid-State Circuits*, vol. 27, pp. 281–287, 1992.
[28] M. G. Wrighton and A. M. DeHon, "Hardware-assisted simulated annealing with application for fast FPGA placement," in *Proc. FPGA*, Monterey, CA, 2003, pp. 33–42.
[29] A. Yan, R. Cheng, and S. J. E. Wilton, "On the sensitivity of FPGA architectural conclusions to experimental assumptions, tools and techniques," in *Proc. FPGA 2002*, Monterey, CA, 2002, pp. 147–156.
[30] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectron. Centre of North Carolina, Research Triangle Park, NC, 1991.

**Elias Ahmed** received the B.Eng. degreein computer systems engineering from Carleton University, Ottawa, ON, Canada, in 1998, and the M.A.Sc. in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2001.

In 1996 and 1997, he held summer research positions at Cadabra Design Libraries Inc., Ottawa, Canada, before joining the company full-time as a Member of the Technical Staff in 1998. He is currently a Senior Software Engineer with the Altera Toronto Technology Center, Toronto, ON, Canada. His research interests include CAD algorithms, place and route, and architecture for field programmable gate arrays (FPGAs).

**Jonathan Rose** received the Ph.D. degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1986.

He is currently a Professor of Electrical and Computer Engineering, University of Toronto, and a Senior Research Scientist at the Altera Toronto Technology Center, Canada. From 1986 to 1989, he was a Post-Doctoral Scholar and then Research Associate in the Computer Systems Laboratory, Stanford University, Stanford, CA. In 1989, he joined the Faculty at the University of Toronto. From 1995 to 1996, he was a Senior Research Scientist at Xilinx, San Jose, CA, working on the Virtex FPGA architecture. In October 1998, he cofounded the Right Track CAD Corporation, which delivered architecture for FPGAs and packing, placement and routing software for FPGAs to FPGA device vendors. He was President and CEO of Right Track until May 2000. Right Track was purchased by Altera Toronto, and became part of the Altera Toronto Technology Center, where Rose was Senior Director until April 2003. His group at Altera Toronto shared responsibility for the development of the architecture for the Altera Stratix, Stratix GX and Cyclone FPGAs. His group was also responsible for placement, routing, and delay annotation software for these devices, and for the placement and routing software for the Altera Apex 20 K and Flex 10 K FPGAs. He currently holds the part-time position of Senior Research Scientist at Altera Toronto.

Dr. Rose is Cofounder, and Member of the Steering Committee, of the ACM FPGA Symposium.