SPEED AND AREA TRADE-OFFS IN CLUSTER-BASED FPGA ARCHITECTURES

Alexander (Sandy) Marquardt, Vaughn Betz, and Jonathan Rose

Right Track CAD Corp.
#313 - 720 Spadina Ave.
Toronto, ON, Canada M5S 2T9
{arm, vaughn, jayar}@rtrack.com

ABSTRACT

One way to reduce the delay and area of FPGAs is to employ logic-cluster-based architectures, where a logic cluster is a group of logic elements connected with high-speed local interconnections. In this paper we empirically evaluate FPGA architectures with logic clusters ranging in size from 1 to 20, and show that compared to architectures with size 1 clusters, architectures with size 8 clusters have 23% less delay (30% faster clock speed), and require 14% less area. We also show that FPGA architectures with large cluster sizes can significantly reduce design compile time — an increasingly important concern as the logic capacity of FPGAs rises. For example, an architecture that uses size 20 clusters requires 7 times less compile time than an architecture with size 1 clusters.

INDEX TERMS

Clustering, design, gate-array, high-speed-interconnect, performance-trade-offs, high-performance.

1. INTRODUCTION

Field-Programmable Gate Arrays (FPGAs) have become one of the most popular implementation media for digital circuits, and since their introduction in 1984 FPGAs have become a multi-billion dollar industry. The key to the success of FPGAs is their programmability, which allows any circuit to be instantly realized by appropriately programming an FPGA.

FPGAs have some compelling advantages over Standard Cells or Mask-Programmed Gate Arrays (MPGAs): faster time-to-market, lower non-recurring engineering costs (NRE), and easier debugging. Additionally, FPGAs offer designers the ability to fix errors or to add features to systems that have already been manufactured. FPGAs are also useful for implementing designs that are low volume or are required immediately, since they do not require design specific manufacturing like Standard Cells or MPGAs.

The benefits offered by FPGAs come at a price — FPGAs are at least three times slower, and require at least ten times the area of MPGAs [1]. This loss in speed is mainly due to the fact that logic in FPGAs is connected via programmable switches, while in Standard Cells or MPGAs, logic is connected with metal wires. The programmable switches in FPGAs have high resistance and capacitance compared to the metal wiring in Standard Cells or MPGAs, and therefore reduce circuit speed. Interconnect delay is more significant (typically well over 50% of the total circuit delay) in FPGAs than it is in MPGAs or Standard Cells, and consequently it is more important to minimize the interconnect delay in FPGAs than it is in MPGAs or Standard Cells.

Another important factor affecting circuit delay is the process used in the manufacture of an FPGA. As process geometries shrink into the deep-submicron region, interconnect resistance and capacitance become increasingly significant — smaller processes which result in improvements in logic speed do not result in similar improvements in interconnect speed. The result of this is that as processes shrink, interconnect delay accounts for an increasing proportion of total circuit delay. Clearly, interconnect delay must be minimized in order to achieve the best possible circuit performance.

The design of an FPGA's architecture can have a significant impact on the performance of circuits implemented in the FPGA. In this work we explore FPGA architectures composed of *logic clusters*, where a logic cluster is a grouping of logic elements connected with high-speed local interconnections. Altera's FLEX 6K, 8K and 10K [2], the Xilinx 5200 and Virtex [3][4], the newest Actel [5], and the Vantis VF1 [6] parts all employ some form of cluster-based logic blocks, so research in this area is of clear commercial relevance.

It is also of interest to see how cluster size affects design compile time. An FPGA composed of large clusters requires fewer logic blocks to implement a circuit than an FPGA using small clusters. This reduces the size of the placement and routing problem, and hence reduces design compile time.

Previous work on logic-cluster based architectures considered only area [7][8]. An earlier version of this work considered circuit speed [10] but used a CAD flow that was not completely timing-driven. In this work we make use of a completely timing-driven CAD flow (including a new timing-driven placement algorithm) to evaluate the effect of cluster size on FPGA speed and area.

This paper is organized as follows. Section 2 introduces the structure of cluster-based logic blocks. In Section 3 we outline the experimental methodology used to evaluate the utility of different cluster sizes. Section 4 discusses how area and delay are modeled. Section 5 discusses the area-delay product metric, and why we believe it is a useful tool for comparing different architectures. Section 6 describes how we selected various parameters for the different cluster sizes. Section 7 presents area and delay results for various cluster sizes, while Section 8 shows how design compile time is affected by different cluster sizes. Section 9 discuses potential sources of inaccuracies. Finally, in Section 10 we present our conclusions.

2. FPGA ARCHITECTURE AND CLUSTER-BASED LOGIC BLOCKS

In general, an FPGA consists of logic blocks, I/O blocks, and programmable routing, as shown in Figure 1. To implement a circuit in an FPGA, each of the logic blocks in the FPGA are appropriately programmed to implement a small part of the functionality of the desired circuit, and each of the I/O blocks is programmed to be an input pad or an output pad. These functional portions and I/Os are then connected through the programmable routing.

The logic block used in an FPGA has a large impact on the performance of the FPGA. We are interested in determining the effects and trade-offs of cluster-based logic blocks, which we describe below.

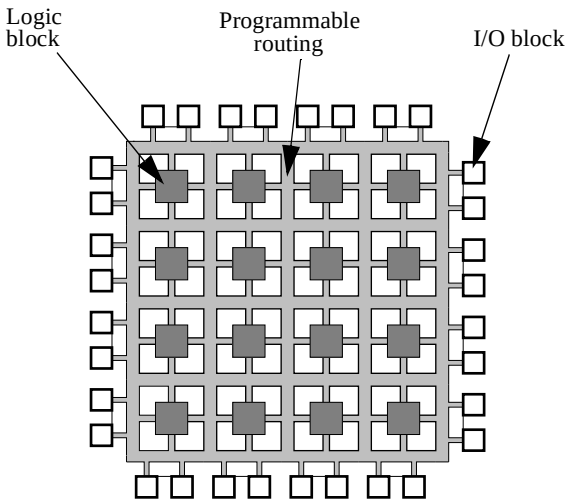

Figure 1 A generic FPGA [1]

2.1 CLUSTER-BASED LOGIC BLOCKS

We are interested in studying logic blocks which consist of a grouping of *basic logic elements* (BLEs) connected with fast local interconnect. In general, a BLE is a small indivisible unit combining sequential and combinational logic; the BLE that we study consists of a 4-LUT and a flip-flop as shown in Figure 2-a. A logic block combining one[1] or more BLEs is known as a *logic cluster* [8][9]. Figure 2-b shows the structure of a logic cluster consisting of N BLEs and the routing required to connect them together.

The clusters that we study are *fully-connected*, meaning that any BLE input can connect to any cluster input or any BLE output. Since the cluster is fully connected it is possible to bring a net into the cluster on a single cluster input, and route this net to many BLEs within the cluster via the local routing. This allows the number of nets brought into the cluster (number of cluster inputs used) to be less than the total number of BLE inputs within the cluster. Another benefit of fully connected clusters is that CAD tools are simplified since all BLEs within the cluster are logically equivalent.

A logic cluster consisting of BLEs is described with the following four parameters [8][9]:

1. The size of (number of inputs to) a LUT (*K*),
2. Cluster size (*N*) — the number of BLEs in a cluster,
3. The number of inputs to the cluster for use as inputs by the LUTs (*I*), and
4. The number of clock inputs to a cluster (for use by the registers), *Mclk*.

This work focuses on logic clusters in which the LUT size, *K*, is 4 and the number of clock pins on a cluster, M_{clk}, is 1 — this is the case shown in Figure 2, and was also the case used in [8][9], and [10]. Note that the total number of BLE inputs is $K \cdot N$; however, only I inputs are brought into the cluster.

[1] A logic cluster that consists of only one BLE has no local routing.

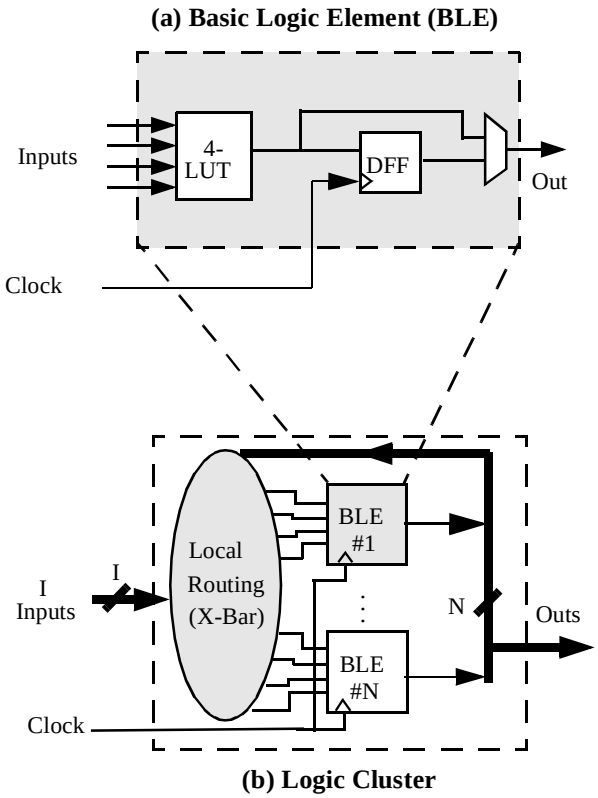

Figure 2 Logic cluster and basic logic element (BLE)

[7] [8] showed that FPGAs composed of logic clusters of size 1-10 BLEs have the best area efficiency. That research did not consider the effect of cluster size on circuit speed, however, it did speculate that larger cluster sizes would have a positive impact on FPGA performance.

3. EXPERIMENTAL METHODOLOGY

We use an empirical method to explore different FPGA architectures. This involves technology-mapping, packing, placing, and routing benchmark circuits[2] into realistic architectures with clusters of size 1 through 20. The area and delay of each circuit implementation is then computed using sophisticated models, and we are able to judge the quality of each architecture.

3.1 CAD FLOW

The CAD flow that we use to evaluate different FPGA architectures is basically the same as in [8][9], and is given in Figure 3. First each circuit is logic-optimized by SIS [12] and technology mapped into 4-LUTs by FlowMap [13]. Then the timing-driven packing algorithm, T-VPack [10][14] is used to group the LUTs and registers into logic clusters of the desired size with the desired number of inputs. After this each circuit is mapped onto an FPGA by a timing-driven placement tool called T-VPlace [14] that we

[2] Our benchmarks consist of the 20 largest MCNC circuits [11]. The circuits range in size from 1047 to 8383 4-LUTs. The circuits used are: alu4, apex2, apex4, bigkey, clma, des, diffeq, dsip, elliptic, ex1010, ex5p, frisc, misex3, pdc, s298, s38417, s38584.1, seq, spla, and tseng.

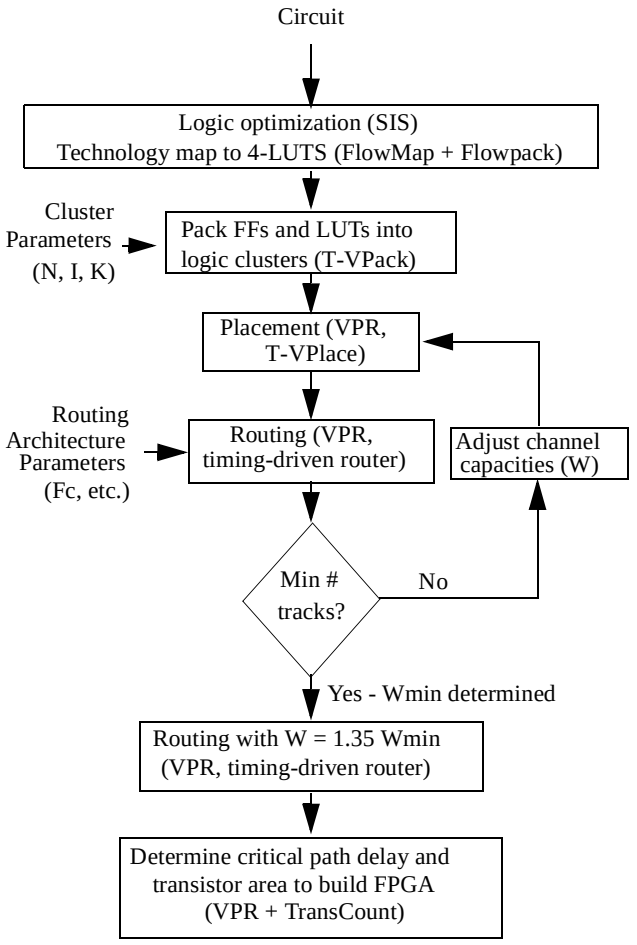

Figure 3 Architecture evaluation CAD flow [8][9].

have incorporated into VPR. Finally, VPR's timing-driven router is used to connect all of the wiring.

T-VPack [10][14] is a timing-driven version of the VPack algorithm developed in [8]. The T-VPack algorithm takes a netlist of LUTs and registers (BLEs), and produces a netlist of logic clusters as shown in Figure 4. T-VPack maps BLEs into clusters so that physical constraints on the number of inputs (*I*), number of BLEs (*N*), and the number of clocks (*Mclk*) are satisfied. In addition to meeting physical constraints, T-VPack has two optimization goals

1. To minimize the number of cluster inputs used, which minimizes the number of point-to-point connections in the post-clustering circuit.
2. To pack BLEs along the critical path into as few clusters as possible so that many critical connections use the fast routing inside the logic clusters.

After packing is complete, the next stage in the CAD flow is placement which is done with T-VPlace [14]. The T-VPlace algorithm is an extension to the VPlace algorithm developed in [8][9]. It is given a netlist of circuit blocks (I/Os or logic clusters), and maps each circuit block into a physical location in the FPGA. This algorithm is simulated annealing [15][16][17] based and optimizes the

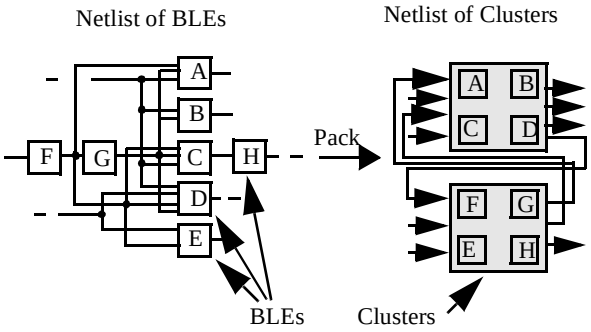

Figure 4 Packing example

final placement to minimize the required routing area as well as minimizing the critical path delay.

The next stage in the CAD flow is routing. The router in VPR [8] [9] is fully timing-driven and attempts to minimize the critical path delay (given the current placement).

Figure 3 shows how VPR computes the minimum number of tracks in which a circuit will route, which we refer to as a *high-stress* routing. Basically VPR repeatedly routes each circuit with different channel widths (number of tracks per channel), scaling the FPGA's architecture until it finds the minimum number of tracks in which the circuit will route. We define a *low-stress* routing to occur when an FPGA has 35% more routing resources than the minimum required to route a given circuit. We feel that low-stress routings are indicative of how an FPGA will generally be used (it is rare that a user will utilize 100% of all routing and logic resources), so our delay results are based on low-stress routings.

By allowing the channel width to vary, and searching for the minimum routable width, we can detect small improvements in FPGA architectures or CAD algorithms that might otherwise go unnoticed. Compare this to mapping a circuit into a fixed size FPGA — this would only tell us if the circuit fit or not. A "binary" result like this makes it is difficult to draw conclusions about new architectures.

After placement and routing, we know exactly how each benchmark circuit is embedded into the FPGA architecture under consideration. This allows us to apply the detailed area and delay models described in Section 4 to evaluate the area and delay of each implementation.

4. Architecture Modeling

In this section we first describe the area and delay models that we use to evaluate the various FPGA architectures. After this we describe the effect that varying cluster size has on segment lengths, and transistor sizing.

4.1 Area Model

The area model[1] that we use is based on counting the number of *minimum-width transistor areas* required to implement each FPGA architecture, which is the same model as was used in [8][9]. A minimum-width transistor area is simply the layout area occupied by the smallest transistor that can be contacted in a process, plus the

[1] Note that the area model is based on transistor-area rather than metal area, since transistor area determines the die size of current FPGAs.

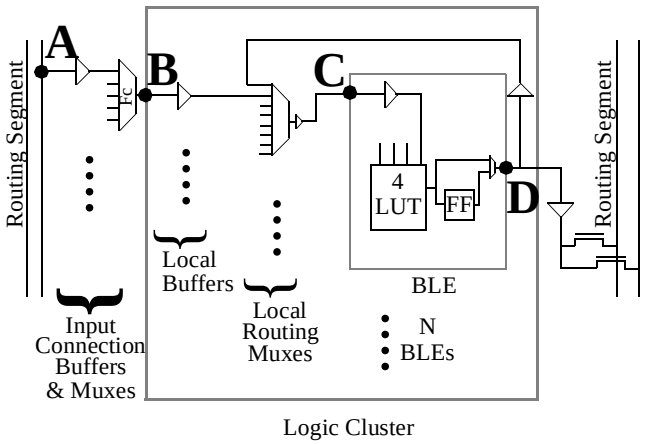

Figure 5 Detailed Logic Cluster Structure

minimum spacing to another transistor above it and to its right [8]. By counting the number of minimum-width transistor areas required to implement an FPGA, rather than the number of square microns which these transistors would occupy, we obtain a process-independent estimate of the FPGA area. The area model that we use is described in detail in [8][9].

We use a program called *TransCount* [9], to determine the area of a cluster-based logic block (including the local cluster routing) with any values of N, I, K, and M_{clk}. This program models such effects as buffer resizing as a function of the fanout of the connections within a logic block, and builds multi-stage buffers when high drive strengths are required. Since the area of an FPGA includes both logic block area and routing area, we use VPR to determine the transistor-count of the area taken by the routing for each FPGA of interest, and by adding this area to the logic block area we obtain the total FPGA area.

4.2 Delay Model

The delays of the connections within logic clusters were found by performing SPICE simulations using TSMC's 0.35 µm process for each structure in the cluster. Figure 5 shows the major structures and speed paths in a logic cluster. Important delay values through this cluster are shown in Table 1, however some delays cannot be listed because the process information is proprietary and was obtained under a non-disclosure agreement.

VPR has a built-in delay estimator that uses a *modified* Elmore delay [18] model to estimate the delay of each connection in the routing. The modifications to the Elmore delay are described in [19], and are such that it can be used to estimate delay of circuits containing buffers, resistors, and capacitors. After every connection's delay in the circuit has been computed, VPR performs a path-based timing analysis using these inter-cluster connection delay values (Elmore delay) and intra-cluster delay values (Table 1). A full description of the timing analyzer used in VPR is available in [8] or [9].

4.3 Effect of Cluster Size on the Physical Length of FPGA Routing Segments

As we increase the cluster size, both the logic area per cluster and routing area per cluster grow. Figure 6 demonstrates how a tile (a logic block plus its associated routing) grows as cluster size is increased. This increased tile size results in routing segments with

Table 1: Intra-cluster delays in TSMC's 0.35 µm CMOS process (letters correspond to points labelled in Figure 5).

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 (and no **D** to **C** path)	379	519
2	760	687	379	1066
4	760	761	379	1140
8	760	902	379	1281
16	760	1054	379	1433
20	760	1081	379	1460

the same "logical length" having different physical lengths for logic clusters of different sizes, where the logical length of a routing segment is the number of logic blocks that the segment spans.

We call the measured length of a routing segment its physical length. The resistance and capacitance of a routing segment grow linearly with the segment's physical length. We have experimentally determined the average rate at which the FPGA tiles grow with cluster size, and have used this information to appropriately scale the routing segment resistance and capacitance values for the various cluster sizes. The increase in the resistance and capacitance of routing segments as the size of the FPGA logic block increases is an important effect that has often been neglected in prior FPGA architecture research.

4.4 Sizing Routing Transistors to Compensate for Different Physical Segment Lengths

To compensate for differences in the capacitance and resistance of routing segments in FPGAs using different sizes of logic clusters, we scale the routing pass transistors and buffers. All of our pass transistor and buffer scaling is in relation to a base architecture that has been area-delay optimized for clusters of size four. From this base architecture, we linearly scale routing buffers and pass transistors depending on the relation between the new segment lengths and the base segment length. For example, in an FPGA with size

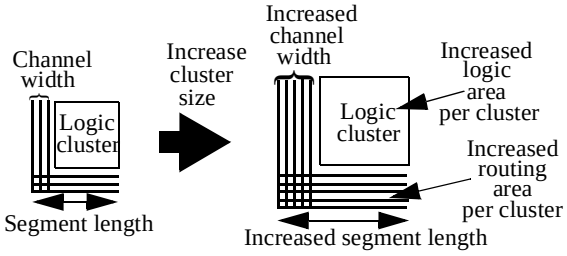

Figure 6 Effect of cluster size on physical length of routing

16 clusters, the physical segment length is approximately 2 times longer than in an architecture with size 4 clusters. To maintain roughly the same speed per routing segment, we increase the size of the routing switches connecting to each wire by a factor of 2. In Section 7.4 we verify that this linear scaling of buffers and pass-transistors with physical segment length provides good results.

VPR models the changes in delay caused by resizing buffers and pass-transistors in the routing, and it also accurately models the area required for different sizes of routing pass-transistors and buffers.

5. Architecture Evaluation — Area-Delay Product

One metric that we will use to evaluate the quality of different architectures is the area-delay product. We feel that there are two reasons that this metric makes sense:

1. Intuitively, we want to find the point at which we are sacrificing the least amount of area for the most improvement in speed. Given that we can always trade area for speed (see below), and speed for area, it makes sense to combine these two factors into one curve to see where the best trade-off occurs.
2. Much of the performance gain from using an FPGA is derived from parallelizing functional units, rather than raw clock speed. In this case, *throughput = number of functional units · clock rate*. Another way of looking at this is, *throughput = (1/ area per functional unit) · (1/delay)*. Therefore if we minimize the area-delay product, we will maximize throughput.

There are two main factors which can affect the area-delay product of an FPGA: transistor sizing, and the FPGA architecture. In general, the speed of an FPGA can be increased (to a point) by sizing up the buffers and transistors within the FPGA, but this increases area. Alternatively, the FPGA can be made smaller by sizing down the buffers and transistors, but this degrades the FPGA performance.

Throughout this paper, we will size the transistors in each FPGA architecture to minimize the FPGA's area-delay product. Only by resizing transistors appropriately for each architecture in this way can we fairly compute the speed and area-efficiency of FPGAs with different logic block architectures.

6. Architecture Parameters

To evaluate the speed and area of an FPGA employing logic clusters for its logic blocks, we must choose not only the logic block architecture and transistor sizes, but also a routing architecture and the flexibility of the logic block to routing interface. The following sections detail the architectural parameters used in our experiments.

6.1 Basic Architecture

We investigate island-style FPGAs in which each logic cluster is surrounded by routing channels on all four sides with the logic cluster input and output pins evenly distributed around the logic cluster perimeter. This basic architecture was shown in Figure 1. For our experiments each circuit is mapped to the smallest square FPGA with enough logic clusters and I/O pads to accommodate it.

In our experiments, we vary the number of I/O pads per row or column depending on the cluster size. Since a large cluster size requires fewer clusters to implement a given circuit, we require more I/O pads per row or column. We set the number of I/O pads per row or column to

$$Pads = \lceil 2 \cdot \sqrt{Cluster\_Size} \rceil \qquad \textbf{(1.1)}$$

Setting the number of I/O pads per row or column with the above equation keeps the total number of I/O pads roughly the same for each FPGA architecture, independent of the cluster size that is used.

Recall that we describe a logic cluster with four parameters: the number of logic inputs (I), the number of BLEs (LUTs and registers) in a cluster (N), the number of clock inputs (M_{clk}), and the number of inputs to each LUT (K). We fix the number of clocks per cluster at one for all our experiments, since the MCNC benchmark circuits we use to evaluate architectures all have only one clock. We set the number of inputs to each LUT, K, to 4, since previous research has shown LUTs of this size are the most area-efficient [20], and because this is the LUT size used in most commercial FPGAs. We describe how we set the number of inputs, I, in the next section.

6.2 Inputs Required vs. Cluster Size

Previous work [8] has examined the issue of how many cluster inputs are required for 98% utilization of the logic clusters, where utilization is defined as

$$utilization = \frac{\left\lceil \frac{num\ logic\ \ blocks}{cluster\ size} \right\rceil}{num\ clusters\ used} \qquad \textbf{(1.2)}$$

That research, however, used VPack to map logic into the clusters. Since we are using our new T-VPack algorithm for packing in our cluster-based logic block experiments, and because T-VPack has better utilization than VPack, it is prudent to re-run these experiments with T-VPack. Figure 7 shows the number of inputs required to achieve an average utilization of 98% vs. cluster size for both VPack and T-VPack[1]. We use the T-VPack results of this experiment to set the number of physical inputs per cluster for the remainder of our architecture studies.

6.3 Routing Architecture

Recall that we define the number of logic blocks which a routing segment spans as the logical length of that segment. In [8][9] it is shown that an architecture in which routing segments have a logical length of four, with 50% of the segments connected by tri-state buffers and 50% connected by pass-transistors, provides good area-efficiency and speed for FPGAs containing logic clusters of size four. This routing architecture is shown in Figure 8. We implicitly assume that this routing architecture is good for architectures containing logic clusters of all sizes, and we use this routing architecture in all of our experiments. Ideally, one would find the best routing architecture for each FPGA employing a different cluster size, but this would require a huge amount of effort. By basing all of our experiments on this routing architecture, we may slightly favor architectures with size four clusters over other architectures.

[1] This shows that T-VPack reduces the number of inputs required vs. T-VPack for 98% utilization at large cluster sizes. The fact that the two tools have different requirements for the number of inputs is an example of the dependencies between FPGA architecture and CAD.

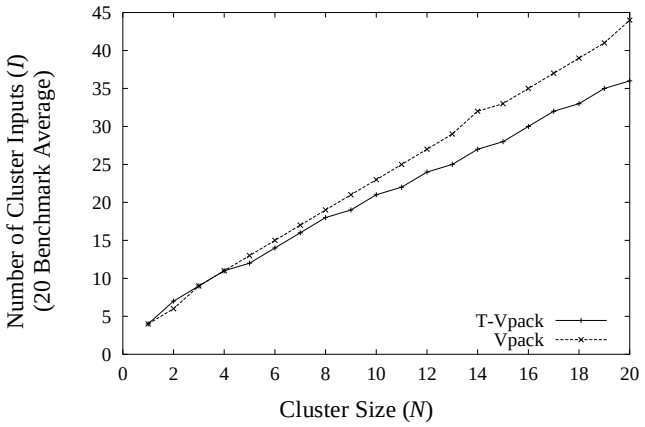

Figure 7 Inputs required for 98% utilization vs. cluster size

6.4 Flexibility of Logic Block to Routing Interconnect vs. Cluster Size

For a cluster of size 1 [21] showed that a good value of F_c (the number of routing tracks to which each logic block pin can connect) is W (the total number of tracks in a channel); This value of F_c means that each logic block pin can connect to any routing track in an adjacent channel. However, for large clusters, setting F_c to W provides far more routing flexibility than is required, wasting area.

[8] found that a more appropriate level of routing flexibility results when the F_c value for logic block output pins, $F_{c,output}$ is set to W/N, so all the experiments in the next section use this value. This choice of $F_{c,output}$ ensures that all the routing tracks in each channel can be driven by at least one output from each cluster.

Choosing the appropriate value for $F_{c,input}$ involves finding the best trade-off between track width and area per track as follows

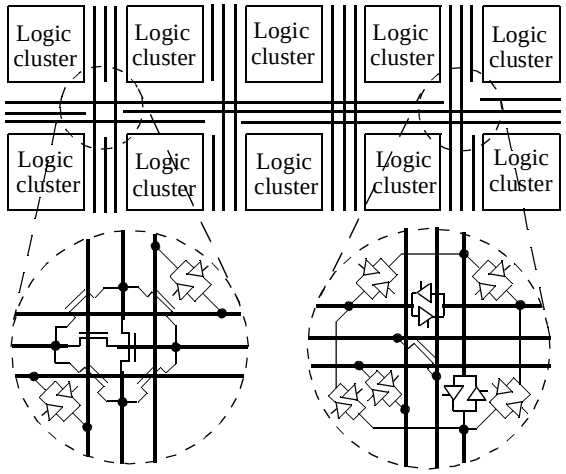

Figure 8 FPGA with length 4 segments, 50% buffered and 50% pass transistor switches.

1. As $F_{c,input}$ is increased, fewer tracks are required to implement a given circuit since the router has more choices of which track each input can connect to.
2. Each track takes more area as $F_{c,input}$ is increased since there are more switches on each track (recall that routing area is determined by transistor area, not wiring area [8][9]).

Therefore, we must determine the point at which the best trade-off occurs. We have run experiments on size 4, 8, 14, and 20 clusters to determine the best $F_{c,input}$ values as shown in Table 2, and have linearly interpolated between these results for other cluster sizes.

Table 2: Routing area vs. $F_{c,\ input}$ for various cluster sizes[a]

$F_{c,\ input}$	Routing Area for various cluster sizes (in millions of minimum-width transistors)			
	4	**8**	**14**	**20**
0.1·W	—	—	—	1.51
0.2·W	—	—	1.38	1.41
0.3·W	—	1.29	1.34	1.41
0.4·W	1.47	1.27	1.34	1.42
0.5·W	1.45	1.28	1.37	1.46
0.6·W	1.44	1.30	—	—
0.7·W	1.45	—	—	—
0.8·W	1.49	—	—	—
0.9·W	1.50	—	—	—
Best $F_{c,input}$ value	0.6·W	0.4·W	0.3·W	0.2·W

a. The MCNC circuits used for these experiments are the 10 smallest of the 20 benchmark circuits that we use in all of our other experiments.

Note, for these experiments, we have noticed that the critical path is not affected by the $F_{c,,input}$ values chosen, so we choose the $F_{c,input}$ value based only on the area results.

7. Experimental Results: Area and Delay at Various Cluster Sizes

Recall that our goal in this work is to determine the effect of cluster size on the area and speed of FPGAs that use a cluster-based architecture. The CAD flow of Figure 3 is used to obtain area and critical-path delay estimations for the 20 benchmark circuits implemented in architectures with clusters of size one through twenty. This involves packing, placing, and routing the benchmark circuits and comparing the resulting FPGA areas and critical path delays. The results that we present are based on low-stress routings

(described in Section 3.1). The total area of each circuit (logic plus routing) is given in terms of the equivalent number of minimum-width transistor areas. Critical path delay is given in seconds.

Note that the CAD tools we use are heuristic and there is variability in the quality of the solutions that the CAD tools obtain. This causes the area and delay curves to be somewhat "jagged". We use the average of 20 circuits to minimize the imperfections of the CAD tool results, but this does not completely smooth the resulting curves. Even with these small imperfections, we believe that the overall trends are still quite visible.

7.1 AREA RESULTS

In Figure 9 we show the geometric average of the area required to implement the benchmarks vs. cluster size. Total area is affected by inter-cluster routing area (area taken by routing between clusters), and cluster area (area taken by BLEs and local cluster routing). We now discuss these two components.

As we increase cluster size up to about size 9, the amount of routing required between clusters is reduced since many connections are completely absorbed within the clusters. After size 9, the routing area begins to increase. We believe that the reason for this increase is because large clusters make it difficult for the placer to do a good job minimizing wirelength. This happens because larger clusters are connected to more nets, which increases the number of clusters that each cluster has nets in common with. It is therefore likely that when the placer moves a large cluster to improve the wire-length of some nets, this same move will increase the wire-length of many other nets.

The area taken by the logic clusters is shown as well. Notice that there is a jump in intra-cluster area between size 1 and size 2 clusters. This occurs because for size 1 clusters there is no need for local multiplexors. For clusters of size 2 through 20, as we increase cluster size (N), the total area taken by the multiplexers within each cluster grows quadratically, but the number of clusters required to implement a circuit is decreasing with 1/N. The overall result is a linear increase in the total area taken by the logic clusters. For sufficiently large clusters, the area reductions in the routing are overtaken by the increased area required to implement the larger clusters.

If one is trying to minimize the area of an FPGA architecture, a cluster of size 7 is the best, however, any cluster size between 1 and 15 requires within 20% of the area taken by size 7 clusters.

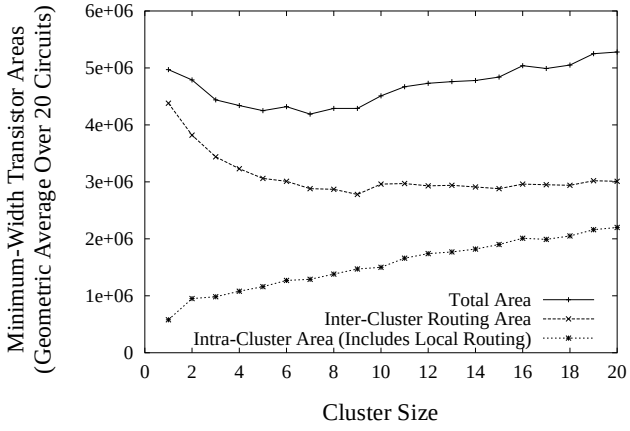

Figure 9 Area vs. Cluster Size

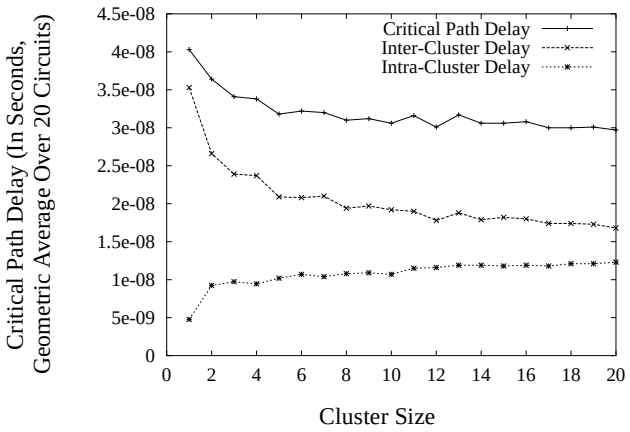

Figure 10 Critical Path Delay vs. Cluster Size

7.2 DELAY RESULTS

Figure 10 shows the geometric average of the critical path delay of the benchmarks vs. cluster size. This graph shows that the critical path delay is decreasing as cluster size is increased. An architecture with size 20 clusters is 33% faster (has 25% less delay) than an architecture with size 1 clusters.

In Figure 11 we show the relationship between the number of internal (intra-cluster — fast) and external (inter-cluster — slower) connections on the critical path. As cluster size is increased the number of internal connections on the critical path is increased, and the number of external connections is decreased. This provides a circuit speedup due to fact that internal connections are faster than external connections[1].

It is interesting to note that the number of external (inter-cluster) nets on the critical path (Figure 11) does not decrease as much with cluster size as the inter-cluster delay (Figure 10) decreases with

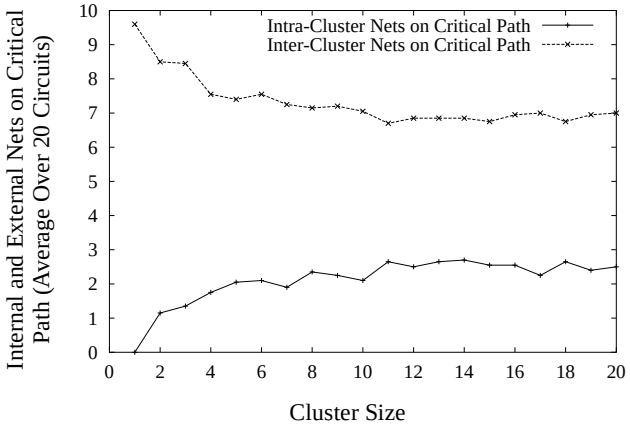

Figure 11 Internal and External Nets on the Critical Path

[1] As cluster size is increased, internal cluster multiplexor and wiring delays increase. If we were to keep increasing the cluster size indefinitely, this effect would eventually result in internal delays becoming large enough that any gains obtained from making connections local to the cluster would be lost.

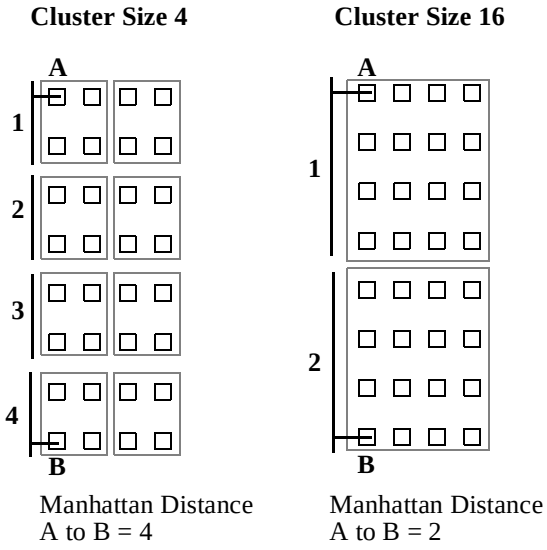

Figure 12 Decreased Manhattan Distance as Cluster Size Increases

cluster size. From size one to size twenty we have a reduction in the number of external nets on the critical path of about 28%; compare this to the inter-cluster component of the critical path delay which has been reduced by 51% over this same range. This means that the circuit speedup visible in Figure 10 for larger cluster sizes is not only caused by a reduction in the number of external nets on the critical path but it is also *caused by inter-cluster connections on the critical path becoming faster.* This is explained below

The improvement in inter-cluster delay with increased cluster size is caused in part by a reduction in the "logical" manhattan distance between connections in the FPGA as shown in Figure 12. By sizing buffers[1] to compensate for the increased physical length of routing wire segments associated with larger clusters, the delay of each routing segment has remained roughly constant. Since the total number of segments on the critical path has decreased due to the reduction in the "logical" manhattan distance, the result is a greater improvement in inter-cluster component of the critical path delay than the reduction in the number of external nets on the critical path would indicate.

7.3 Area-Delay Product

In Figure 13 we show the geometric average of the area-delay product of the benchmarks vs. cluster size. An important result is visible in this figure — clusters of size 3 through 20 provide the best trade-off between area and delay, with the best results occurring for a cluster of size 8. Compared to a cluster of size one, a cluster of size eight has an area-delay product that is 33.5% less.

7.4 Effect of Routing Transistor Sizing on Critical Path Delay and Area at Various Cluster Sizes

The purpose of this section is to provide a verification that the manner in which we sized routing buffers and transistors is acceptable, and did not favor one cluster size over another.

[1] Changes in delay and area due to different size routing buffers is accounted for in VPRs timing and area models.

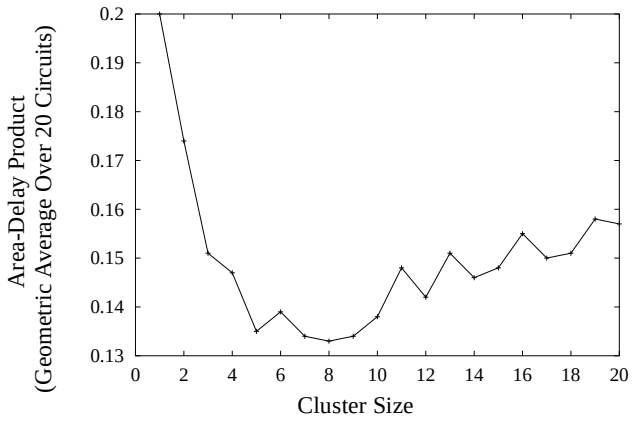

Figure 13 Area-Delay Product vs. Cluster Size

We have repeated the experiments described in Section 7 using transistor and buffer sizes of one-half and double the sizes used in Section 7. The results from these experiments are shown in Figures 14, 15, and 16. These experiments show that area can be traded for speed, and speed for area. Figure 16 shows that for large cluster sizes, our "regular" transistor sizing is too large for the best area-delay trade-off. Therefore, for large cluster sizes the "half" transistor size results are a better indicator of the architecture performance.

7.5 Summary

Any architecture with clusters in the range of size 3 to 20 is reasonable, with size 8 being the best. On average, circuits implemented in an FPGA with size eight clusters have 23% less delay (a 30% increase in speed) and use 14% less area than circuits implemented in an FPGA with size one clusters.

We also showed how the sizing of the routing transistors and buffers affects area and delay.

While we presented only 20 circuit average results in this paper, all of the individual benchmark circuits tracked these averages quite well (with minor variations, mostly at cluster sizes one and two).

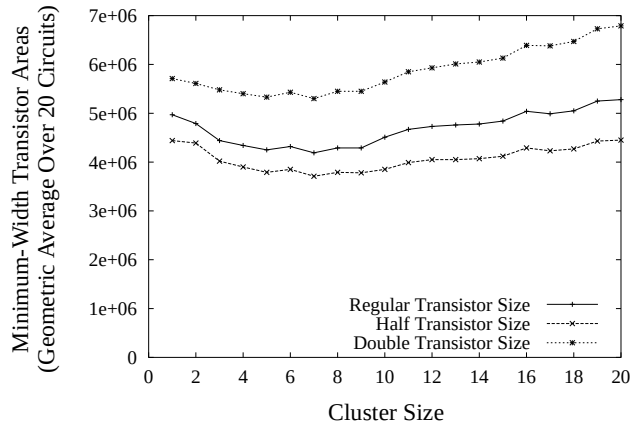

Figure 14 Area vs. Cluster Size for Various Transistor Sizings

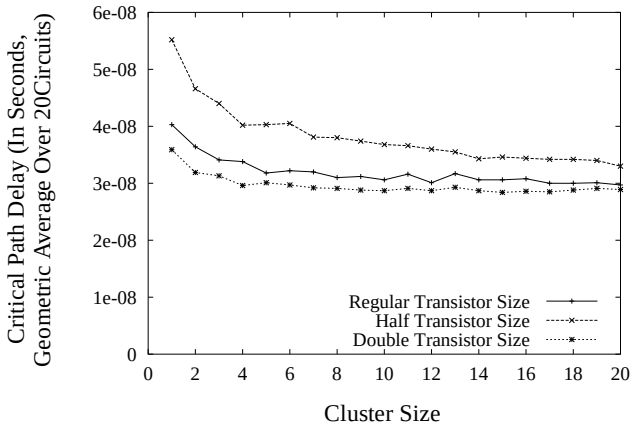

Figure 15 Critical Path Delay vs. Cluster Size for Various Transistor Sizings

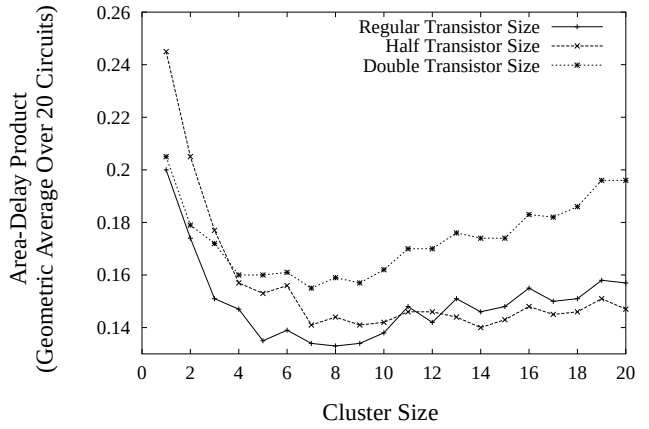

Figure 16 Area-Delay Product vs. Cluster Size for Various Transistor Sizings

8. Design Compile Time vs. Cluster Size

In this section we demonstrate that cluster-based FPGA architectures can significantly improve design compile time. Figure 17 shows how the average CPU time (on a 300 MHz UltraSparc workstation) required to implement the circuits varies with cluster size. The solid line in Figure 17 shows the total (packing, placement, and routing) compile time, while the three dashed lines show the individual components of this compile time. The routing time is taken from low-stress (minimum number of tracks per channel + 35%) routings. The packing time is insignificant for all cluster sizes compared to the placement and routing time.

As larger logic clusters are employed in an FPGA the time to compile circuits is dramatically reduced. As larger clusters are employed, fewer of these clusters are required to implement each circuit. Since the size of a placement problem is proportional to the number of logic clusters that a circuit is mapped to, this dramatically reduces placement time. In Figure 17, for example, one can see that the placement time is reduced by a factor of 8 times as the cluster size increases from 1 to 20. Larger logic clusters also reduce the routing time since large clusters result in fewer inter-cluster connections to route. For example, using a size 20 logic cluster reduces routing time by 3 times vs. using a size 1 cluster. Building

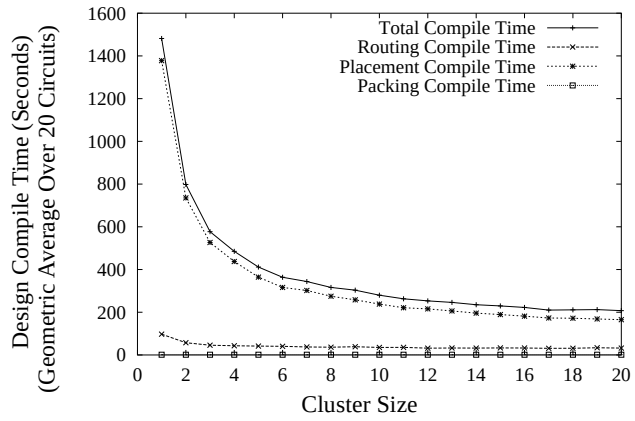

Figure 17 Design Compile Time vs. Cluster Size

an FPGA with size 20 logic clusters reduces the total CPU time required for placement and routing by 7 times vs. a size 1 logic cluster.

9. Potential Sources of Inaccuracies

Every effort has been made to ensure that our results are accurate, however, there are three potential sources of inaccuracies.

First, without actually laying out the various FPGA architectures, there is some estimation involved in determining how much area various FPGA implementations will require.

Second, VPR uses the Elmore delay model [18] to evaluate the routing delay of circuits implemented in the various FPGA architectures. Generally the routing delays calculated by VPR are within 9% of SPICE delays [8][9]. Since routing delay is only a portion of the total circuit delay, and because we use actual SPICE values to evaluate the intra-cluster component of the circuit delay, our overall delay numbers should deviate by less than 9% compared to SPICE.

Third, area and delay results are affected by the quality of the placement and routing software. The tools used for these experiments have been shown to produce high quality results [8][9][10][14], but it is always possible that the CAD software does a better job for certain architectures over others.

We have taken considerable care to minimize the effects of these potential sources of inaccuracies, and we believe that our results are of high quality.

10. Conclusions

Using the area-delay product evaluation metric, we have demonstrated that logic clusters containing between 3 and 20 BLEs all achieve good performance, so any cluster size in this range is a reasonable choice. Compared to FPGAs using a single BLE logic block, logic clusters in this size range achieve significant area and speed improvements. For example, an FPGA employing a size 8 logic cluster requires 14% less area, achieves 30% higher speed, and has an area-delay product 33.5% lower than an FPGA using a single BLE logic block.

We have also shown that larger cluster sizes can significantly improve design compile time. For example, an architecture with size 20 clusters requires 8 times less placement time and 3 times less routing time compared to an architecture with size 1 clusters.

11. REFERENCES

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

[2] Altera Inc., *Data Book*, 1998.

[3] Xilinx Inc., "XC5200 Series of FPGAs", *Data Book*, 1997.

[4] Xilinx Inc., "Virtex 2.5 V Field Programmable Gate Arrays", *Advance Product Data Sheet*, 1998.

[5] S. Kaptanoglu et. al., "A new high density and very low cost reprogrammable FPGA Architecture", *FPGA*, 1999, pp. 3 - 12.

[6] O. Agrawal et. al., "An Innovative, Segmented High Performance FPGA Family with Variable-Grain-Architecture and Wide-gating Functions," *FPGA*, 1999, pp. 17 - 26.

[7] 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.

[8] V. Betz, "Architecture and CAD for Speed and Area Optimization of FPGAs," *Ph. D. Dissertation, University of Toronto*, 1998.

[9] V. Betz, J. Rose, A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, February 1999.

[10] A. Marquardt, V. Betz, J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density", *FPGA*, 1999, pp 37-46.

[11] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," *Tech. Report*, Microelectronics Center of North Carolina, 1991.

[12] E. M. Sentovich et al, "SIS: A System for Sequential Circuit Analysis," *Tech. Report No. UCB/ERL M92/41*, University of California, Berkeley, 1992.

[13] 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.

[14] A. Marquardt, "Cluster-Based Architecture, Timing-Driven Packing, and Timing-Driven Placement for FPGAs," *M.A.Sc., University of Toronto*, 1999.

[15] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," *Science*, May 13, 1983, pp. 671 - 680.

[16] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *JSSC*, April 1985, pp. 510 - 522.

[17] C. Sechen and K. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement," *ICCAD*, 1987, pp. 478 - 481.

[18] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *J. Applied Physics*, Vol. 19, January 1948, pp. 55-63.

[19] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," *ICCAD*, 1996, pp. 44 - 49.

[20] J. Rose, R. J. Francis, D. Lewis and P. Chow, "Architecture of Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE Journal of Solid State Circuits*, Oct. 1990, pp. 1217 - 1225.

[21] J. Rose and S. Brown. "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *JSSC*, March 1991, pp. 277 - 282.