Speed and Area Tradeoffs in Cluster-Based FPGA Architectures

Alexander Marquardt, Vaughn Betz, and Jonathan Rose

***Abstract*—One way to reduce the delay and area of field-programmable gate arrays (FPGA's) 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 FPGA's rises. For example, an architecture that uses size 20 clusters requires seven times less compile time than an architecture with size 1 clusters.**

***Index Terms*—Clustering, design, gate-array, high-performance, high-speed interconnect, performance tradeoffs.**

I. Introduction

FIELD-programmable gate arrays (FPGA's) have become one of the most popular implementation media for digital circuits. Since their introduction in 1984, FPGA's have become a multibillion dollar industry. The key to the success of FPGA's is their programmability, which allows any circuit to be instantly realized by appropriately programming an FPGA.

FPGA's have some compelling advantages over standard cells or mask-programmed gate arrays (MPGA's): faster time-to-market, lower nonrecurring engineering (NRE) costs, and easier debugging. Additionally, FPGA's offer designers the ability to fix errors or to add features to systems that have already been manufactured. FPGA's 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 MPGA's.

The benefits offered by FPGA's come at a price—FPGA's are at least three times slower and require at least ten times the area of MPGA's [1]. This loss in speed is mainly due to the fact that logic in FPGA's is connected via programmable switches, while in standard cells or MPGA's, logic is connected with metal wires. The programmable switches in FPGA's have high resistance and capacitance compared to the metal wiring in standard cells or MPGA's, and therefore reduce circuit speed. Interconnect delay is more significant (typically well over 50% of the total circuit delay) in FPGA's than it is in MPGA's or standard cells, and consequently it is more important to minimize the interconnect delay in FPGA's than it is in MPGA's 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-submicrometer region, interconnect resistance and capacitance become increasingly significant—smaller processes that 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 paper, 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 6 K, 8 K, and 10 K [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 computer-aided design (CAD) flow that was not completely timing driven. In this paper, 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 II introduces the structure of cluster-based logic blocks. In Section III, we outline the experimental methodology used to evaluate the utility of different cluster sizes. Section IV discusses how area and delay are modeled. Section V discusses the area-delay product metric and why we believe it is a useful tool for comparing different architectures. Section VI describes how we selected various parameters for the different cluster sizes. SectionVII presents area and delay results for various cluster sizes, while Section VIII shows how design compile time is affected by different cluster sizes. Section IX discusses potential sources of inaccuracies. Last, in Section X, we present our conclusions.

Manuscript received April 26, 1999.

A. Marquardt and V. Betz are with Right Track CAD Corporation, Toronto, Ont., M5S 2T9, Canada (e-mail: arm@rtrack.com; vaughn@rtrack.com).

J. Rose is with Right Track CAD Corporation, Toronto, Ont., M5S 2T9, Canada, and the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont., M5S 3G4, Canada (e-mail: jayar@eecg.toronto.edu).

Publisher Item Identifier S 1063-8210(00)00756-3.

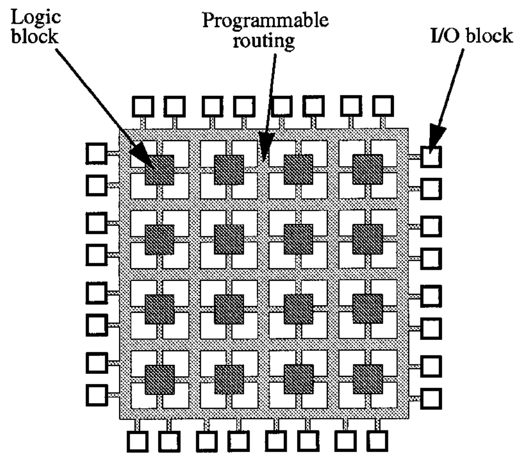

Fig. 1. A generic FPGA [1].

II. FPGA Architecture and Cluster-Based Logic Blocks

In general, an FPGA consists of logic blocks, I/O blocks, and programmable routing, as shown in Fig. 1. To implement a circuit in an FPGA, each of the logic blocks in the FPGA is 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/O's 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 tradeoffs of cluster-based logic blocks, which we describe below.

A. Cluster-Based Logic Blocks

We are interested in studying logic blocks that consist of a grouping of *basic logic elements* (BLE's) 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 four-input lookup table (LUT) and a flip-flop as shown in Fig. 2(a). A logic block combining one[1] or more BLE's is known as a *logic cluster* [8], [9]. Fig. 2(b) shows the structure of a logic cluster consisting of N BLE's 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 BLE's 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 BLE's within the cluster are logically equivalent.

A logic cluster consisting of BLE's is described with the following four parameters [8], [9]:

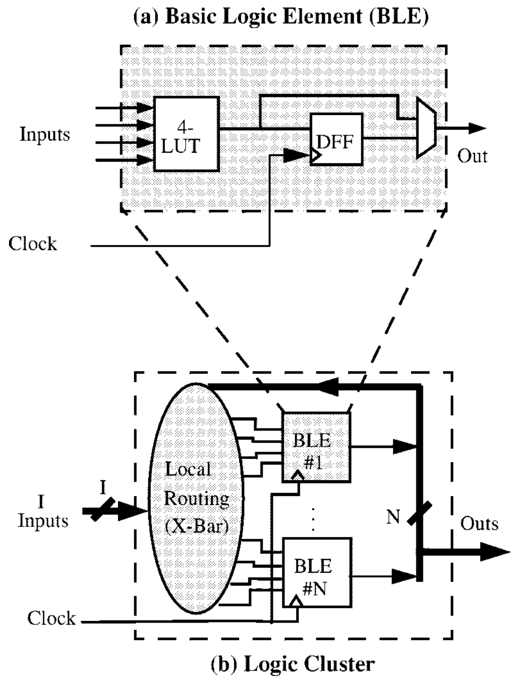

Fig. 2. Logic cluster and basic logic element.

1) size of (number of inputs to) an LUT (K);
2) cluster size (N)—the number of BLE's in a cluster;
3) number of inputs to the cluster for use as inputs by the LUT's (I);
4) number of clock inputs to a cluster (for use by the registers) M_{clk}.

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

References [7] and [8] showed that FPGA's composed of logic clusters of size 1–10 BLE's 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.

III. 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 1–20. The area and delay of each circuit implementation are then computed using sophisticated models, and we are able to judge the quality of each architecture.

A. CAD Flow

The CAD flow that we use to evaluate different FPGA architectures is basically the same as in [8] and [9] and is given

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

[2]Our benchmarks consist of the 20 largest MCNC circuits [11]. The circuits range in size from 1047 to 8383 4-LUT's. 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.

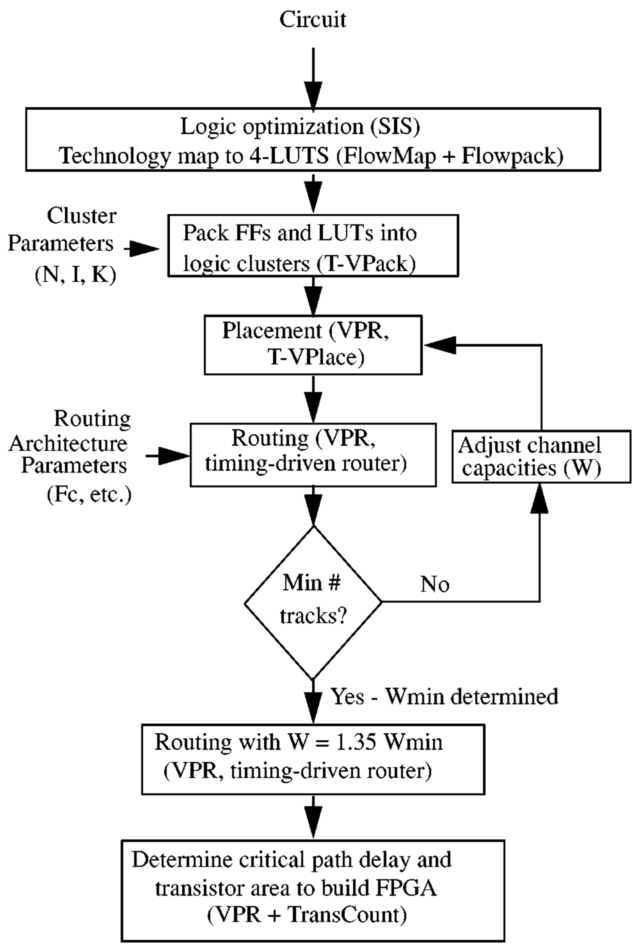
Circuit
Logic optimization (SIS)
Technology map to 4-LUTS (FlowMap + Flowpack)
Cluster Parameters (N, I, K)
Pack FFs and LUTs into logic clusters (T-VPack)
Placement (VPR, T-VPlace)
Routing Architecture Parameters (Fc, etc.)
Routing (VPR, timing-driven router)
Adjust channel capacities (W)
Min # tracks?
No
Yes - Wmin determined
Routing with W = 1.35 Wmin (VPR, timing-driven router)
Determine critical path delay and transistor area to build FPGA (VPR + TransCount)

Fig. 3. Architecture evaluation CAD flow [8], [9].

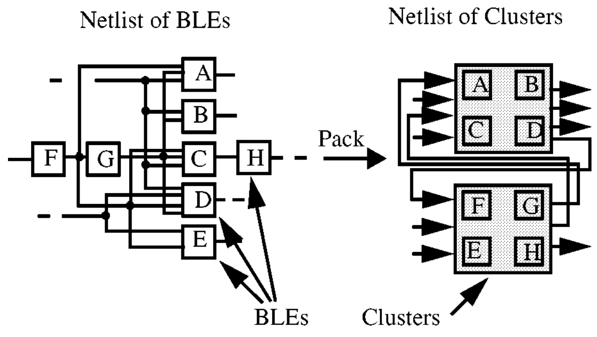

Fig. 4. Packing example.

in Fig. 3. First each circuit is logic optimized by SIS [12] and technology mapped into 4-LUT's by FlowMap [13]. Then the timing-driven packing algorithm, T-VPack [10], [14], is used to group the LUT's 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 have incorporated into VPR. Last, 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 LUT's and registers (BLE's) and produces a netlist of logic clusters as shown in Fig. 4. T-VPack maps BLE's into clusters so that physical constraints on the number of inputs (I), number of BLE's (N), and number of clocks (M_{clk}) 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 BLE's 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] and [9]. It is given a netlist of circuit blocks (I/O's or logic clusters) and maps each circuit block into a physical location in the FPGA. This algorithm is simulated annealing [15]–[17] based and optimizes the final placement to minimize the required routing area as well as minimize 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).

Fig. 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 IV to evaluate the area and delay of each implementation.

IV. 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.

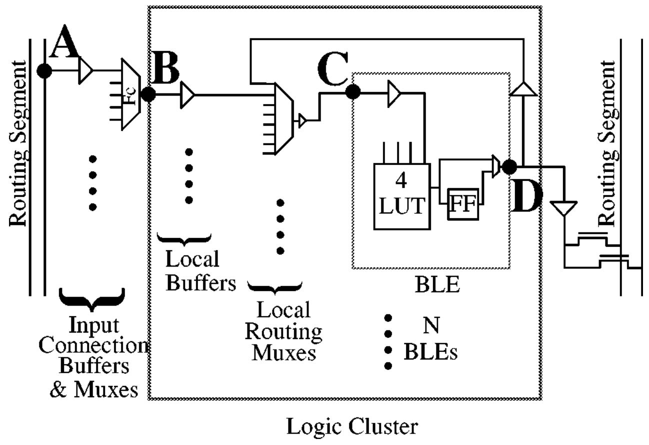

Fig. 5. Detailed logic cluster structure.

TABLE I
INTRACLUSTER DELAYS IN TSMC'S 0.35-µm CMOS PROCESS (LETTERS CORRESPOND TO POINTS LABELLED IN FIG. 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

A. Area Model

The area model[3] 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] and [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 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 that 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] and [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 multistage 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.

B. 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. Fig. 5 shows the major structures and speed paths in a logic cluster. Important delay values through this cluster are shown in Table I; however, some delays cannot be listed because the process information is proprietary and was obtained under a nondisclosure 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 intercluster connection delay values (Elmore delay) and intracluster delay values (Table I). A full description of the timing analyzer used in VPR is available in [8] or [9].

[3]Note that the area model is based on transistor area rather than metal area, since transistor area determines the die size of current FPGA's.

C. 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. Fig. 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 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.

D. Sizing Routing Transistors to Compensate for Different Physical Segment Lengths

To compensate for differences in the capacitance and resistance of routing segments in FPGA's 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 16 clusters, the physical segment length is approximately two times longer than in an archi-

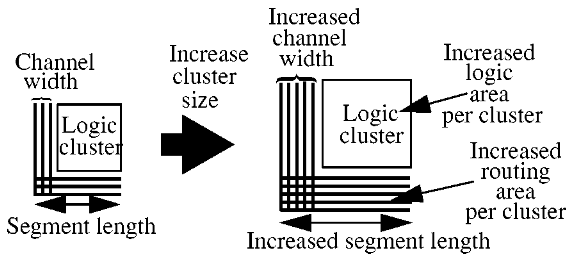

Fig. 6. Effect of cluster size on physical length of routing segments.

tecture 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 two. In Section VII-D, 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.

V. 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 tradeoff 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* $\cdot$ *clock rate*. Another way of looking at this is *throughput* $=$ (*1/area per functional unit*) $\cdot$ (*1/delay*). Therefore, if we minimize the area-delay product, we will maximize throughput.

There are two main factors that 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 FPGA’s with different logic block architectures.

VI. 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.

A. Basic Architecture

We investigate island-style FPGA’s 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 Fig. 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

$$\text{Pads} = \lceil 2 \cdot \sqrt{\text{Cluster\_Size}} \rceil. \tag{1}$$

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 BLE’s (LUT’s 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 four, since previous research has shown that LUT’s of this size are the most area efficient [20] and because this is the LUT size used in most commercial FPGA’s. We describe how we set the number of inputs I in the next section.

B. Inputs Required Versus 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

$$\text{utilization} = \frac{\left\lceil \dfrac{\text{num logic blocks}}{\text{cluster size}} \right\rceil}{\text{num cluster used}}. \tag{2}$$

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 rerun these experiments with T-VPack. Fig. 7 shows the number of inputs required to achieve an average utilization of 98% versus cluster size for both VPack and T-VPack.[4] 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.

[4]This shows that T-VPack reduces the number of inputs required versus 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.

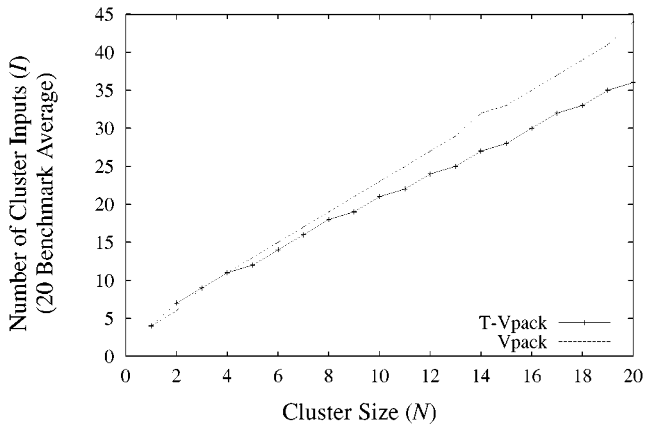

Fig. 7. Inputs required for 98% utilization versus cluster size.

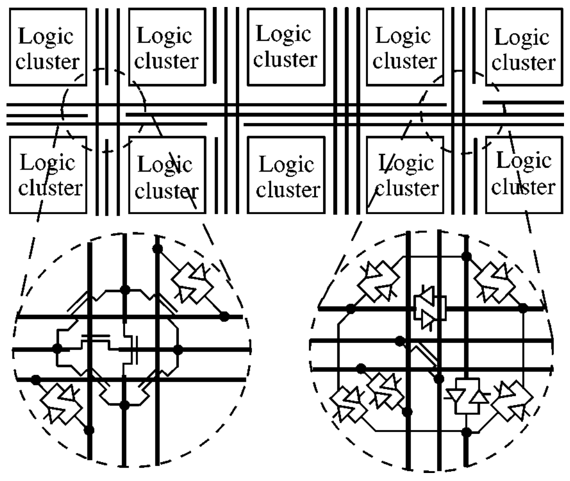

Fig. 8. FPGA with length 4 segments, 50% buffered, and 50% pass-transistor switches.

C. Routing Architecture

Recall that we define the number of logic blocks that a routing segment spans as the logical length of that segment. In [8] and [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 FPGA's containing logic clusters of size four. This routing architecture is shown in Fig. 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 4 clusters over other architectures.

D. Flexibility of Logic Block to Routing Interconnect Versus 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.

TABLE II
ROUTING AREA VERSUS $F_{c,\text{input}}$ FOR VARIOUS CLUSTER SIZES

	Routing Area for various cluster sizes (in millions of minimum-width transistors)			
$F_{c,\text{ input}}$	**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 $F_{c,\text{input}}$ value	0.6·W	0.4·W	0.3·W	0.2·W

Reference [8] found that a more appropriate level of routing flexibility results when the F_c value for logic block output pins $F_{c,\text{output}}$ is set to W/N, so all the experiments in the next section use this value. This choice of $F_{c,\text{output}}$ 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 $F_{c,\text{input}}$ involves finding the best tradeoff between track width and area per track as follows.

1) As $F_{c,\text{input}}$ 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 $F_{c,\text{input}}$ 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 tradeoff occurs. We have run experiments on size 4, 8, 14, and 20 clusters to determine the best $F_{c,\text{input}}$ values, as shown in Table II, and have linearly interpolated between these results for other cluster sizes. Note that for these experiments, we have noticed that the critical path is not affected by the $F_{c,\text{input}}$ values chosen, so we choose the $F_{c,\text{input}}$ value based only on the area results.

VII. 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 FPGA's that use a cluster-based architecture. The CAD flow of Fig. 3 is used to obtain area and critical-path delay estimations for the 20 benchmark circuits implemented in architectures with clusters of size 1–20.

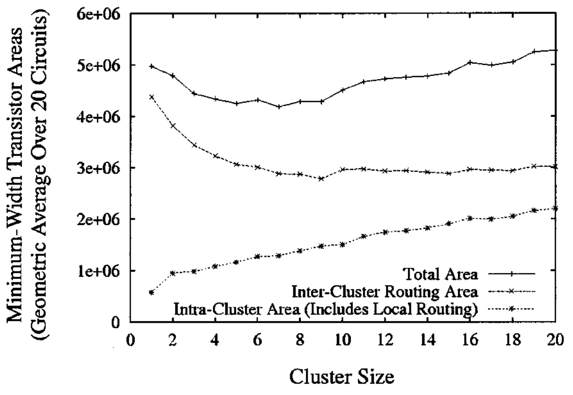

Fig. 9. Area versus cluster size.

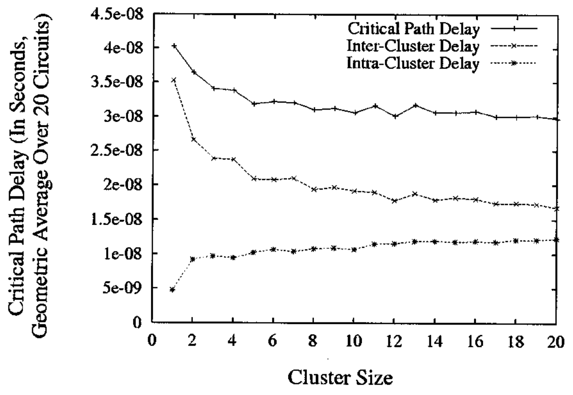

Fig. 10. Critical path delay versus cluster size.

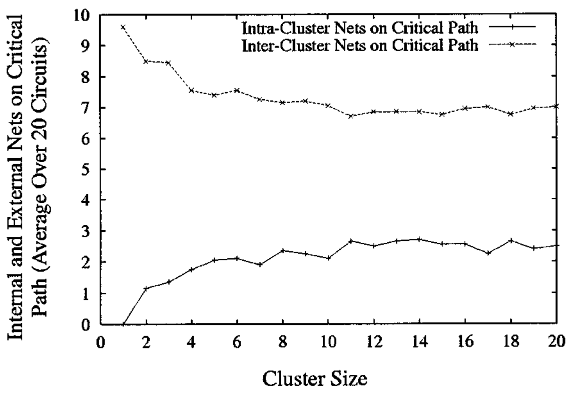

Fig. 11. Internal and external nets on the critical path.

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 III-A). 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 an 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.

A. Area Results

In Fig. 9, we show the geometric average of the area required to implement the benchmarks versus cluster size. Total area is affected by intercluster routing area (area taken by routing between clusters) and cluster area (area taken by BLE's 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 with which each cluster has nets in common. 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 intracluster area between size 1 and size 2 clusters. This occurs because for size 1 clusters, there is no need for local multiplexers. For clusters of size 2–20, 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 decreases 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.

B. Delay Results

Fig. 10 shows the geometric average of the critical path delay of the benchmarks versus 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 Fig. 11, we show the relationship between the number of internal (intracluster—fast) and external (intercluster—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.[5]

It is interesting to note that the number of external (intercluster) nets on the critical path (Fig. 11) does not decrease as much with cluster size as the intercluster delay (Fig. 10) decreases with cluster size. From size 1–20, we have a reduction in the number of external nets on the critical path of about 28%; compare this to the intercluster component of the critical path delay, which has been reduced by 51% over this same range.

[5]As cluster size is increased, internal cluster multiplexer 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.

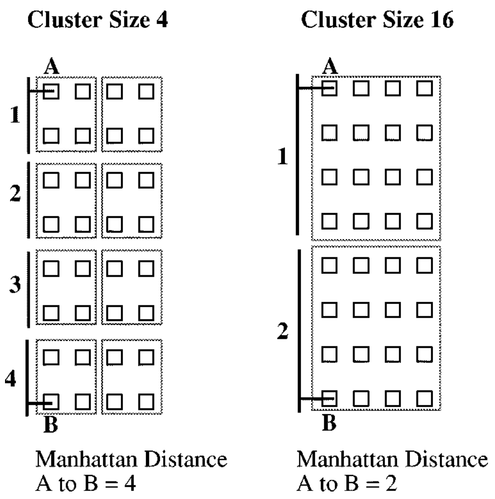

Fig. 12. Decreased Manhattan distance as cluster size increases.

This means that the circuit speedup visible in Fig. 10 for larger cluster sizes is not only caused by a reduction in the number of external nets on the critical path, but is also *caused by inter-cluster connections on the critical paths becoming faster*. This is explained below.

The improvement in intercluster 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 Fig. 12. By sizing buffers[6] 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 intercluster component of the critical path delay than the reduction in the number of external nets on the critical path would indicate.

C. Area-Delay Product

In Fig. 13, we show the geometric average of the area-delay product of the benchmarks versus cluster size. An important result is visible in this figure—clusters of size 3–20 provide the best tradeoff 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 8 has an area-delay product that is 33.5% less.

D. 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.

We have repeated the experiments described in SectionVII using transistor and buffer sizes of one-half and double the sizes used in Section VII. The results from these experiments are shown in Figs. 14–16. These experiments show that area can be traded for speed and speed for area. Fig. 16 shows that for large cluster sizes, our "regular" transistor sizing is too large for the best area-delay tradeoff. Therefore, for large cluster sizes,

[6]Changes in delay and area due to different size routing buffers are accounted for in VPR's timing and area models.

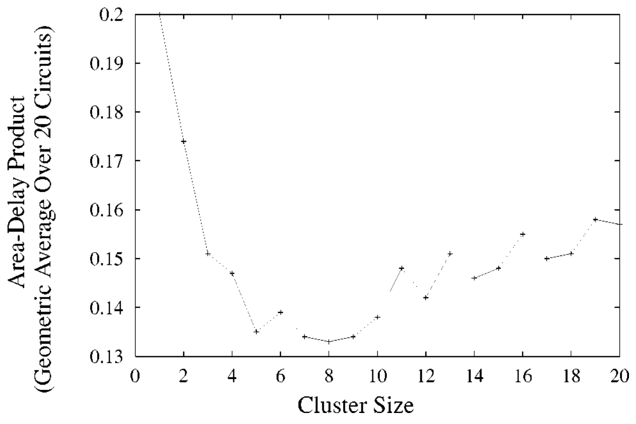

Fig. 13. Area-delay product versus cluster size.

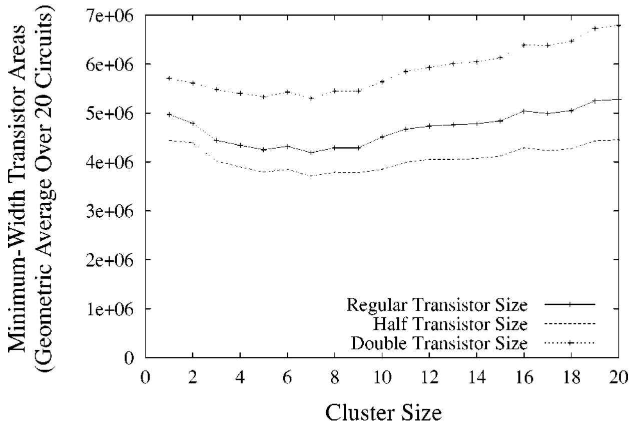

Fig. 14. Area versus cluster size for various transistor sizings.

the "half" transistor size results are a better indicator of the architecture performance.

E. Summary

Any architecture with clusters in the range of size 3–20 is reasonable, with size 8 being the best. On average, circuits implemented in an FPGA with size 8 clusters have 23% less delay (a 30% increase in speed) and use 14% less area than circuits implemented in an FPGA with size 1 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 1 and 2).

VIII. Design Compile Time Versus Cluster Size

In this section, we demonstrate that cluster-based FPGA architectures can significantly improve design compile time. Fig. 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 Fig. 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

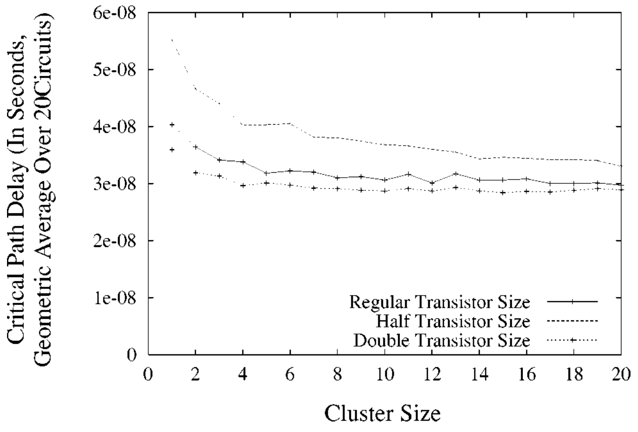

Fig. 15. Critical path delay versus cluster size for various transistor sizings.

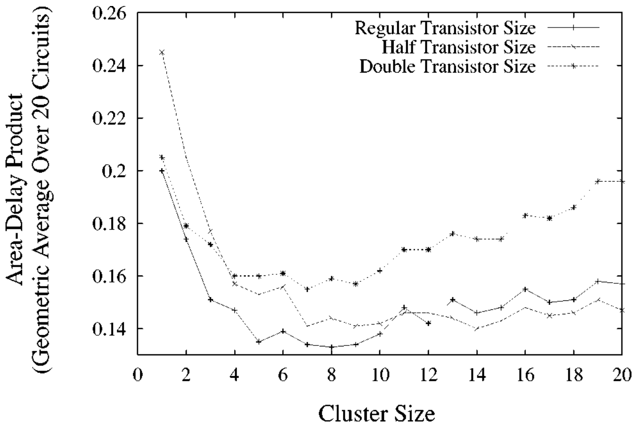

Fig. 16. Area-delay product versus cluster size for various transistor sizings.

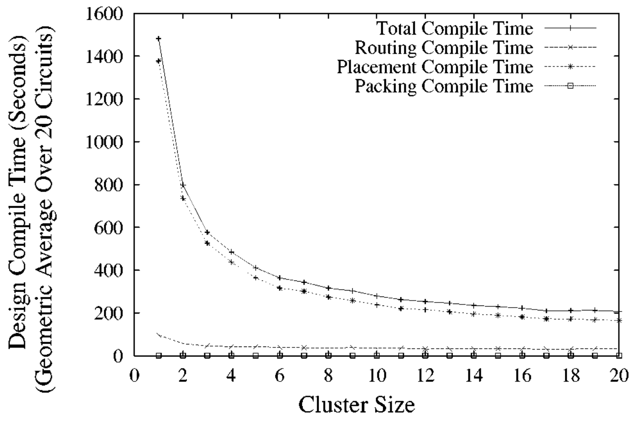

Fig. 17. Design compile time versus cluster size.

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 to which a circuit is mapped, this dramatically reduces placement time. In Fig. 17, for example, one can see that the placement time is reduced by a factor of eight times as the cluster size increases from 1 to 20. Larger logic clusters also reduce the routing time since large clusters result in fewer intercluster connections to route. For example, using a size 20 logic cluster reduces routing time by three times versus using a size 1 cluster. Building an FPGA with size 20 logic clusters reduces the total CPU time required for placement and routing by seven times versus a size 1 logic cluster.

IX. 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 intracluster 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]–[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.

X. Conclusion

Using the area-delay product evaluation metric, we have demonstrated that logic clusters containing between 3–20 BLE's all achieve good performance, so any cluster size in this range is a reasonable choice. Compared to FPGA's 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 eight times less placement time and three times less routing time compared to an architecture with size 1 clusters.

References

[1] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field-Programmable Gate Arrays*. Norwell, MA: Kluwer Academic, 1992.
[2] *Data Book*, Altera, Inc., 1998.
[3] *Data Book*, Xilinx, Inc., 1997.
[4] *Advance Product Data Sheet*, Xilinx, Inc., 1998.
[5] S. Kaptanoglu *et al.*, "A new high density and very low cost reprogrammable FPGA architecture," *FPGA*, pp. 3–12, 1999.
[6] O. Agrawal *et al.*, "An innovative, segmented high performance FPGA family with variable-grain-architecture and wide-gating functions," *FPGA*, pp. 17–26, 1999.
[7] V. Betz and J. Rose, "How much logic should go in an FPGA logic block?," *IEEE Design Test Mag.*, pp. 10–15, Spring 1998.
[8] V. Betz, "Architecture and CAD for speed and area optimization of FPGAs," Ph.D. dissertation, Univ. Toronto, 1998.

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

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

[11] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectronics Center of North Carolina, Tech. Rep., 1991.

[12] E. M. Sentovich *et al.*, "SIS: A system for sequential circuit analysis," Univ. Calif., Berkeley, CA, Tech. Rep. UCB/ERL M92/41, 1992.

[13] 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. 13, pp. 1–12, Jan. 1994.

[14] A. Marquardt, "Cluster-based architecture, timing-driven packing, and timing-driven placement for FPGAs," Master's thesis, Univ. Toronto, Toronto, Canada, 1999.

[15] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, pp. 671–680, May 13, 1983.

[16] C. Sechen and A. Sangiovanni-Vincentelli, "The timberwolf placement and routing package," *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 510–522, Apr. 1985.

[17] C. Sechen and K. Lee , "An improved simulated annealing algorithm for row-based placement," in *Proc. ICCAD*, 1987, pp. 478–481.

[18] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55–63, Jan. 1948.

[19] T. Okamoto and J. Cong, "Buffered Steiner tree construction with wire sizing for interconnect layout optimization," in *Proc. 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 J. Solid-State Circuits*, vol. 25, pp. 1217–1225, Oct. 1990.

[21] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE J. Solid-State Circuits*, vol. 26, pp. 277–282, Mar. 1991.

Alexander Marquardt received the undergraduate degree from the University of Manitoba, Winnipeg, Man., Canada, in 1995 and the M.A.Sc. degree from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont., Canada, in 1999.

Between degrees he worked for two years at Computing Devices Canada, Ottawa, Ont., developing software and hardware for embedded systems. He is currently with Right Track CAD Corporation, Toronto, where he is an FPGA architect and CAD developer.

Vaughn Betz received the B.Sc. degree from the University of Manitoba, Winnipeg, Man., Canada, in 1991, the M.S. degree from the University of Illinois at Urbana-Champaign in 1993, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont., Canada.

His research consisted of developing advanced CAD tools for placement and routing in FPGA's and devising routing architectures that improve the speed and density of FPGA's. He is now Vice President of Engineering at Right Track CAD Corporation, Toronto. Prior to moving into FPGA research, he spent several years writing electromagnetic field solvers for the analysis of various structures, including circuit boards and chip packaging at very high speeds.

Dr. Betz received both the Governor General's Gold Medal and Colton Award for Research Excellence for his Ph.D. work.

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

He is a Professor of Electrical and Computer Engineering at the University of Toronto and President and CEO of Right Track CAD Corporation, Toronto. From 1986 to 1989, he was a Research Associate in the Computer Systems Laboratory at Stanford University. In 1989, he joined the Faculty of the University of Toronto. He spent 1995–1996 as a Senior Research Scientist at Xilinx, Inc., San Jose, CA, working on a next-generation FPGA architecture. He is the Cofounder of the ACM FPGA Symposium and remains part of its steering and program committees. He has worked for Bell-Northern Research and a number of FPGA companies on a consulting basis. His research covers all aspects of FPGA's including architecture, CAD, field-programmable systems, and graphics and vision applications of rapid prototyping systems.

Dr. Rose was a corecipient of a distinguished paper award (with S. Brown) at the 1990 ICCAD Conference.