Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size

Vaughn Betz and Jonathan Rose

Department of Electrical and Computer Engineering, University of Toronto
10 King's College Road, Toronto, Ontario, CANADA M5S 3G4
{vaughn, jayar}@eecg.utoronto.ca

Abstract

While modern FPGAs often contain clusters of 4-input lookup tables and flip flops, little is known about good choices for two key architectural parameters: the number of these basic logic elements (BLEs) in each cluster, and the total number of distinct inputs that the programmable routing can provide to each cluster. In this paper we explore the effect of these parameters on FPGA area-efficiency. We show that a cluster containing N BLEs needs only 2N + 2 distinct inputs (vs. the 4N maximum) to achieve complete logic utilization. Secondly, we find that a cluster size of 4 is most area-efficient, and leads to an FPGA that is 5 - 10% more area-efficient than an FPGA based on a single BLE logic block.

1. Introduction

One of the key determinants of an FPGA's area-efficiency is the structure and granularity of its logic block. If a very simple, or fine-grained, logic block is employed, more logic blocks will be required to implement a given circuit, and the routing area required to interconnect the blocks may become excessive. On the other hand, if a very complex, or coarse-grained, logic block is used, much of the logic block functionality may be unused in most circuits, again wasting area.

Most commercial FPGAs use logic blocks based on look-up tables (LUTs) [1, 2, 3], and accordingly most prior research has focused on LUT-based logic blocks [4, 5, 6]. In [4], it is shown that a 4-input LUT is the most area-efficient LUT, chiefly because LUT complexity grows exponentially with the number of inputs. In this study, we investigate a logic block based on a *cluster* of 4-input LUTs. The complexity of this logic block cluster grows less than quadratically with cluster size, so it holds promise as a practical coarse-grained logic block.

We explore two questions concerning this cluster architecture. First, how many distinct inputs should be provided to a cluster of N 4-LUTs? Secondly, how many 4 LUTs should be included in a cluster to create the most area-efficient logic block? Recent FPGAs from Xilinx [7], Altera [1], Lucent Technologies [3] and Actel [8] have all grouped several LUTs together into a more coarse-grained logic block, but there has been little published work investigating the number of LUTs which should be included in a cluster.

The next section describes the cluster architecture in detail. Section 3 outlines the experimental method we used to evaluate each variant of the architecture. Section 4 describes the algorithms used in our logic cluster packing program. Section 5 presents results concerning the number of inputs that must be provided to a cluster of N 4-LUTs, while Section 6 evaluates the area-efficiency of clusters of different sizes. Finally, we summarize our results and conclusions.

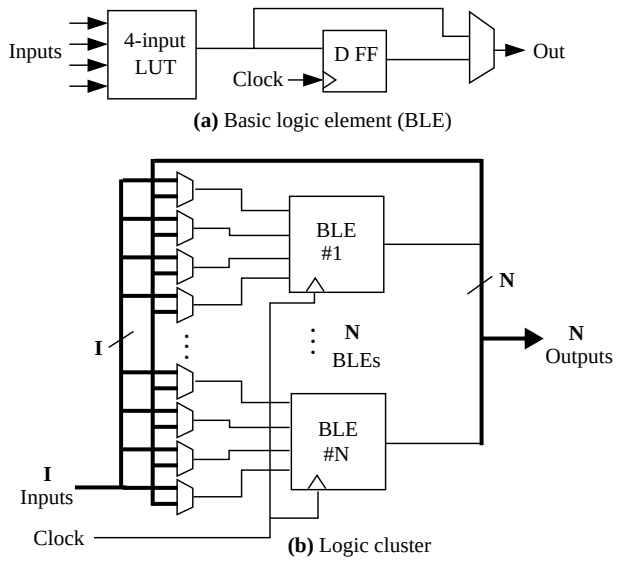
This research was supported by the Information Technology Centre of Ontario, the Walter C. Sumner Foundation, and an NSERC 1967 Scholarship.

2. Cluster-Based Logic Blocks

Fig. 1 shows the structure of a logic cluster. This logic block has a two-level hierarchy; the overall block is a collection of *basic logic elements* (BLEs). As shown in Fig. 1a, our basic logic element is composed of a 4-LUT and a register, and the BLE output can be either the registered or unregistered version of the LUT output. The complete logic block consists of N interconnected BLEs, as shown in Fig. 1b. We call the total logic block a *logic cluster*.

We describe a logic cluster via two parameters, N and I. N is the number of BLEs per cluster, while I is the number of inputs to the cluster. As Fig. 1 shows, not all 4N LUT inputs are accessible from outside the logic cluster. Instead, only I external inputs are provided to the logic cluster -- multiplexers allow arbitrary connections of these cluster inputs to the BLE inputs. The same multiplexers also connect to each of the N BLE outputs, allowing the output of any BLE within the cluster to be connected to any of the BLE inputs. All N outputs of the logic cluster can be connected to the FPGA routing for use by other logic clusters.

Notice that the logic cluster of Fig. 1 is *fully connect*ed; i.e. each of the 4N BLE inputs can be connected to any of the I cluster inputs or any of the N BLE outputs. It is simpler to write CAD tools that completely exploit logic clusters that are fully connected than those which are not. For example, determining if a group of BLEs can be implemented in a single cluster only requires counting the number of cluster inputs



(a) Basic logic element (BLE)

(b) Logic cluster

Fig. 1. Structure of basic logic element (BLE) and logic cluster.

and comparing it to I if a cluster is fully connected. As well, in a fully-connected logic cluster all the cluster inputs and all the cluster outputs are logically-equivalent, which gives the router a great deal of flexibility in how it routes inter-cluster nets. The logic block cluster used in the Altera 8K and 10K FPGAs is fully connected [1], and the logic block cluster used in the Xilinx 5200 FPGA is nearly fully connected [7].

While a strictly hierarchical FPGA was investigated in [9], to our knowledge this is the first work to investigate the use of logic blocks with a two-level hierarchy within an otherwise flat FPGA architecture.

3. Experimental Methodology

Our goal in this research is to determine the values of N and I that lead to the most area-efficient FPGA architecture. Our method is experimental -- we technology-map, place and route 20 of the largest MCNC benchmark circuits [10] into each architecture to determine the area used in each case. Nine of these benchmark circuits are sequential and eleven are combinational; they range in size from 500 to 3690 BLEs.

3.1. CAD Flow

Fig. 2 illustrates the CAD flow used in these experiments. First, the SIS [11] synthesis package is used to perform technology-independent logic optimization of each circuit. Next, each circuit is technology-mapped into 4-LUTs and flip flops by FlowMap [12]. Our VPACK program (described in Section 4) then maps this netlist of 4-LUTs and flip flops into logic clusters with the specified values of N and I. Finally, the VPR placement and routing tool [13] is used to place and globally route the circuit.

As Fig. 2 shows, the circuit is repeatedly routed with different channel capacities until VPR finds the minimum number of tracks per channel required to successfully globally route the circuit. At this point we have enough information to use our area model to evaluate the architecture's area-efficiency.

3.2. Area Model

The area model is based on counting the number of minimum width transistors required to implement a benchmark circuit in each FPGA architecture (larger transistors are counted as several minimum width transistors). To allow

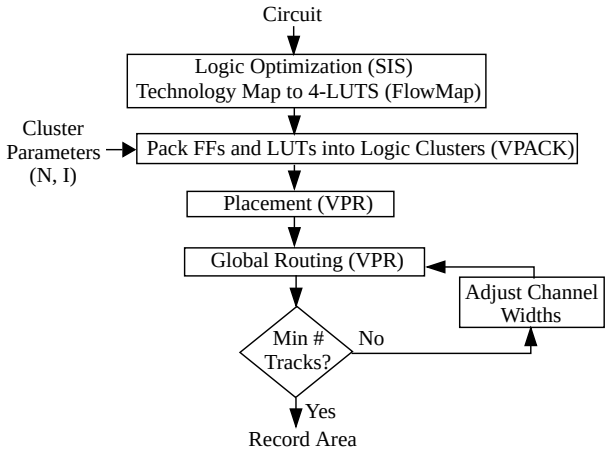


Fig. 2. Architecture Evaluation Flow.

averaging of results from circuits of different sizes, we use a normalized area metric: number of transistors used per BLE in a circuit. We have developed a detailed model of the number of transistors required to implement both logic clusters and FPGA routing in an SRAM-based FPGA [14]. This model tries to build an FPGA using as few transistors as possible without unduly compromising speed. The model takes as input: the logic cluster parameters (N and I); the number of routing tracks to which each logic cluster input or output can connect (F_c [15]); the number of routing track segments to which each segment can connect at a switch point (F_s [15]); and the channel width required after detailed routing ($W_{detailed}$).

N and I are the architectural parameters being varied in this study. We set the other parameters in the model as follows. F_s is set to 3, since this is the value used by most commercial FPGAs. The fourth parameter, $W_{detailed}$, can be determined after a circuit has been placed, globally routed and detailed routed. We do not currently have a detailed router capable of routing logic cluster based FPGAs, so we use the number of tracks required after global routing, W_{global}, to estimate $W_{detailed}$. It has been shown that $W_{detailed}$ is highly correlated to W_{global} in FPGAs with reasonable amounts of interconnection flexibility [15]. We have used the SEGA detailed router [16] to determine that for a "conventional" logic block consisting of 1 BLE, $W_{detailed}$ is approximately 1.35 times W_{global}. Throughout our experiments we assume that this relation holds true for other logic blocks, provided appropriate choices are made for the value of F_c.

We present results using two different assumptions, one pessimistic and one optimistic, about how $W_{detailed}$ depends on F_c. The pessimistic assumption sets F_c equal to 10 for all architectures, and assumes that $W_{detailed}$ is always 35% more than W_{global}. We set $F_c = 10$ because $F_c = W_{detailed}$ has been shown to be a reasonable choice when the logic block is a single BLE [15], and the average value of $W_{detailed}$ over our 20 benchmarks for this architecture is 10. This model is pessimistic because the full connectivity of a logic cluster means *all* the inputs are equivalent, and *all* the outputs are also equivalent. Therefore, with F_c set to 10 for all architectures, the number of choices the detailed router has to enter and leave logic blocks increases essentially linearly with cluster size, yet we assume it still takes 1.35 W_{global} tracks to detailed route a circuit.

Our optimistic assumption assumes that $W_{detailed} = 1.35\ W_{global}$ can be achieved with lower values of F_c as the cluster size increases. The optimistic area model sets $F_c = 10$ only for FPGAs using a cluster size of 1. As the cluster size increases, F_c is scaled by the number of pins on the cluster, relative to a cluster of size 1. In this way, the number of possible connections to a cluster is kept constant as the cluster size increases. This is an optimistic model because the number of nets that must be routed to a cluster increases with cluster size, so there is more competition for connections. This competition is not as severe as one might initially assume however, as the average number of nets input to a cluster increases relatively slowly with cluster size, and some cluster outputs will only be used internally. Note that the Altera Flex 8K and 10K FPGAs leverage the logical equivalence of cluster inputs and outputs

to allow the use of F_c values of 1 and 2, respectively [1].

Our transistor model assumes that, in addition to the circuitry shown in Fig. 1, each logic cluster has one set/reset signal of programmable polarity. We also assume that the clock and set/reset signals are routed on dedicated nets, as this is the usual case in commercial FPGAs. We have verified that our model accurately predicts the number of transistors used in several key structures in a commercial FPGA [17].

4. Logic Cluster Packing Algorithm

We have developed a tool, VPACK, that first packs flip flops and LUTs together into BLEs, and then packs multiple BLEs into logic clusters. VPACK takes as input a netlist of flip flops and LUTs. It first uses a simple and optimal pattern-matching algorithm to pack a flip flop and a LUT together into one BLE wherever possible.

The second step of VPACK packs these BLEs into logic clusters. The optimization goals are twofold: attempt to fill each cluster to its capacity, N, and minimize the number of used inputs to each cluster.

VPACK constructs each cluster sequentially. It begins by choosing a seed BLE for the cluster. We have found that the best way to choose this seed is to select the currently unclustered BLE with the most used inputs, as such BLEs use the most cluster inputs, which are a scarce resource. VPACK then greedily selects the BLE which shares the most inputs and outputs with the cluster being constructed; this tends to minimize the number of inputs that must be routed to each cluster. This procedure of greedily selecting a BLE to include in the cluster continues until either the cluster is full or until adding any unclustered BLE would cause the number of distinct inputs needed by the cluster to exceed I. If the cluster is full, we select a new seed BLE and begin packing BLEs into a new cluster. If, however, the cluster occupancy is still less than N but we cannot add any BLEs because of a lack of cluster inputs, a second, hill-climbing, phase of VPACK is invoked.

Since we know that any clusters that reach this second phase will be difficult to pack to capacity, VPACK now selects BLEs to add to the cluster in order to minimize the increase in the number of cluster inputs required. In this phase, VPACK also allows BLEs to be added to a cluster even if it results in an infeasible cluster (i.e the number of inputs required by the cluster exceeds I). Note that adding a BLE to a cluster in which all of its inputs are already present, and in which the output of the BLE is used by some other BLE already in the cluster *decreases* the number of distinct inputs to the cluster by 1. This is the key to the hill-climbing phase; while adding one BLE to a cluster may make it infeasible, it may become feasible again when additional BLEs are added. The hill-climbing phase terminates when the cluster is full; if the cluster is still infeasible VPACK backs up the last point at which the cluster was feasible. VPACK then selects a seed BLE for the next cluster and invokes the first phase again, as before.

This clustering algorithm is very efficient. None of the twenty benchmark circuits used in this study required more than 3 seconds to cluster on a 70 MHz Sparc 5. The complexity of the algorithm is O(kC), where C is the number of BLEs in a circuit, and k is the maximum fanout of any net.

5. Experimental Results: Relationship of I to N

As discussed in the introduction, the first question we wish to answer is how many distinct inputs, I, should be provided to a cluster of size N. Since the number of transistors required to implement each of the multiplexers shown in Fig. 1 grows linearly with I (for large I), we would like to make I as small as possible. On the other hand, if I is made too small, many of the BLEs in a logic cluster may become essentially unusable, reducing logic utilization and wasting area. We find the minimum value of I that allows good cluster utilization by running benchmark circuits through the first two steps shown in Fig. 2, technology-mapping and cluster packing, and measuring the resulting logic utilization for different values of I.

Fig. 3 shows how the average logic utilization of our 20 benchmarks varies with I for three different logic cluster sizes. The vertical axis is the fraction of BLEs in a cluster that VPACK is able to use, while the horizontal axis is the number of distinct inputs to the cluster relative to the total number of BLE inputs in a cluster (i.e. I/4N). For very low values of I, the logic utilization is very low, as one would expect. It is interesting, however, that when I is only 50 to 60% of the total number of BLE inputs, the logic utilization is essentially 100%. Clearly it is possible to pack BLEs together so that they have many common inputs and can reuse locally generated outputs. The relative amount of input sharing and output reuse increases slightly with logic cluster size, causing the curves in Fig. 3 to shift to the left as cluster size increases.

The solid line in Fig. 4 shows the value of I required to achieve 98% logic utilization as the cluster size, N, is varied. The dashed line in Fig. 4 shows how the average number of logic cluster inputs that are actually used varies with cluster size. Although there are 4N BLE inputs in a logic cluster of size N, the number of inputs required to achieve 98% logic utilization is only about 2N + 2. Furthermore, the average number of logic cluster inputs that are actually used grows even more slowly. On average, a cluster of size 1 uses 3.5 of its inputs, while an cluster of size 16 uses only 19.7 of its inputs. In other words, while the logic per cluster has increased by a factor of 16, the average number of connections that must be routed to each cluster has increased by a factor of only 5.6.

Our results indicate that commercial FPGAs can be more aggressive in reducing the value of I. For example, the Altera Flex 8K FPGAs use logic clusters with N = 8 and I = 24 [1],

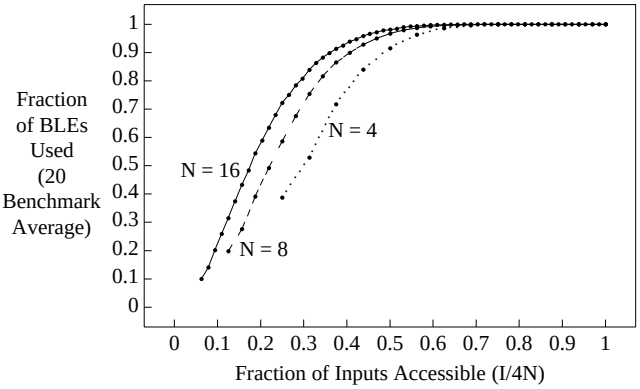


Fig. 3. Logic utilization vs. number of logic cluster inputs.

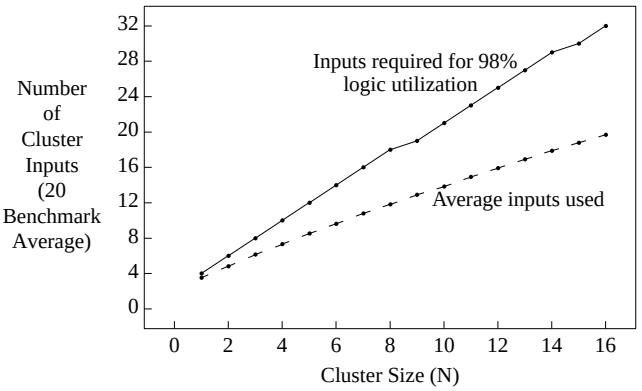


Fig. 4. Variation in inputs required and inputs used with cluster size.

while our results indicate that I = 18 suffices for a cluster of this size. Similarly, the Xilinx 5200 FPGA uses a logic cluster with N = 4, and makes all 16 LUT inputs accessible [7], while our results suggest 10 inputs are sufficient.

6. Experimental Results: Best Logic Cluster Size

We are now in a position to examine which cluster size leads to the most area-efficient FPGA. Throughout this section, the number of inputs, I, to a cluster of size N is chosen to be the minimum value that allows VPACK to achieve 98% logic utilization. This value of I allows our logic clusters to be essentially fully utilized, while minimizing the complexity of the cluster input multiplexers.

We ran 20 benchmark circuits through the experimental flow described in Section 3, and determined the area they required after placement and routing in each architecture. Fig. 5 shows how our area metric, number of transistors required per BLE, varies with cluster size under both the pessimistic and optimistic area models of Section 3. The pessimistic area model predicts that clusters of size 1, i.e. the traditional single BLE logic block, and clusters of size 4 are essentially tied as the most area-efficient logic blocks. The optimistic area model, on the other hand, predicts that a cluster size of 4 is best, and leads to an FPGA that is 12% more area-efficient than one based on a cluster of size 1. Since we know the truth is somewhere between these two models, we can conclude that a cluster of size 4 is most area-efficient, and will lead to an FPGA that is approximately 5 - 10% more area-efficient than one using a single BLE as its logic block.

Cluster-based logic blocks have two other advantages

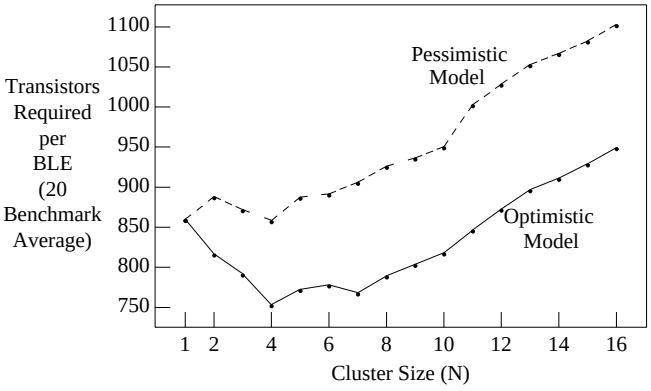


Fig. 5. Area-efficiency versus cluster size.

over single BLE logic blocks. First, in an FPGA composed of logic clusters many nets will be completely contained with a logic cluster. These nets will be routed using only the multiplexers within a cluster; as the delay of these multiplexers is less than that of the main FPGA routing this will tend to increase the FPGA speed. Secondly, by clustering N BLEs into each logic cluster before placement we reduce the number of blocks to be placed by a factor of N. This greatly reduces the placement time, which is of increasing concern in today's large FPGAs.

7. Conclusions

There are two main conclusions to be drawn from this work. First, the number of distinct inputs required by a cluster grows fairly slowly with cluster size, N. A cluster of size N requires approximately 2N + 2 distinct inputs. Secondly, some cluster-based logic blocks lead to better area-efficiency than the traditional single BLE (4-LUT plus flip flop) logic block. Specifically, we found that a cluster of size 4 with 10 distinct inputs is the most area-efficient, and leads to an FPGA that is approximately 5 - 10% more area-efficient than one based on a single BLE logic block. Since cluster-based logic blocks also lead to reduced placement times and a faster FPGA overall, the advantages of cluster-based logic blocks over a single-BLE logic block are significant.

References

[1] Altera Inc., *Data Book*, 1996.

[2] Xilinx Inc., *The Programmable Logic Data Book*, 1994.

[3] AT & T Inc., *ORCA Datasheet*, 1994.

[4] 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.

[5] J. Kouloheris and A. El Gamal, "FPGA Area vs. Cell Granularity -- Lookup Tables and PLA Cells," *ACM Workshop on Field-Programmable Gate Arrays*, 1992, pp. 9 - 14.

[6] D. Hill and N-S Woo, "The Benefits of Flexibility in Look-up Table FPGAs," in *FPGAs*, W. Moore and W. Luk, Eds., Abingdon: 1991, pp. 127 - 136.

[7] D. Tavana, W. Yee, S. Young, and B. Fawcett, "Logic Block and Routing Considerations for a New SRAM-Based FPGA Architecture," *CICC*, 1995, pp. 24.6.1 - 24.6.4.

[8] D. Bursky, "Programmable Arrays Mix FPGA and ASIC Blocks," *Electronic Design*, Oct. 14, 1996, pp. 69 - 74.

[9] A. Aggarwal and D. Lewis, "Routing Architectures for Hierarchical Field Programmable Gate Arrays," *ICCD*, 1994, pp. 475 - 478.

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

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

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

[13] V. Betz and J. Rose, "Directional Bias and Non-Uniformity in FPGA Global Routing Architectures," *ICCAD*, 1996, pp. 652 - 659.

[14] V. Betz, *PhD Dissertation*, in preparation, University of Toronto.

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

[16] S. Brown, G. Lemieux, and M. Khellah, "Segmented Routing for Speed-Performance and Routability in Field-Programmable Gate Arrays," *Journal of VLSI Design*, Vol. 4, No. 4, pp. 275 - 291

[17] F. Heile, Altera Corp., San Jose, CA, *Personal Communication*.