Using Architectural "Families" to Increase FPGA Speed and Density

Vaughn Betz and Jonathan Rose
University of Toronto
Toronto, ON, Canada
M5S 1A4
vaughn@eecg.toronto.edu

Abstract

In order to narrow the speed and density gap between FPGAs and MPGAs we propose the development of "families" of FPGAs. Each FPGA family is targeted at a single maximum logic capacity, and consists of several "siblings", or FPGAs of different yet complementary architectures. Any given application circuit is implemented in the sibling with the most appropriate architecture. With properly chosen siblings, one can develop a family of FPGAs which will have better speed and density than any single FPGA. We apply this concept to create two different FPGA families, one composed of architectures with different types of hard-wired logic blocks and the other created from architectures with different types of heterogeneous logic blocks. We found that a family composed of eight chips with different hard-wired logic block architectures simultaneously improves density by 12 to 14% and speed by 18 to 20% over the best single hard-wired FPGA.

1 Introduction

Designing hardware with Field-Programmable Gate Arrays (FPGAs) avoids the higher non-recurring engineering costs and longer development times of Mask-Programmed Gate Array (MPGA) solutions. These two key advantages, shorter design cycles and lower development costs, have made FPGAs an extremely popular technology for prototyping and low-volume production runs. However, the reprogrammability that allows such rapid and inexpensive development exacts a cost; typically the speed and density of logic implemented in an FPGA is an order of magnitude lower than that of logic implemented in an MPGA [1]. This speed disadvantage makes FPGAs unsuitable for a large portion of hardware projects, while the area penalty makes FPGAs more expensive than MPGAs for the high-volume implementation of a hardware component.

We propose a method by which these speed and density disadvantages can be reduced. An FPGA architecture is normally selected so that it can implement the largest possible class of application circuits as efficiently as possible. This means that the FPGA must contain enough programmable switches and routing resources to give *any* application circuit a reasonable chance of successfully routing. The vast majority of circuits will utilize only a small portion of these routing switches, but since different circuits use different switches, reducing the flexibility of the FPGA by replacing some of these switches with metal links is difficult. Some application circuits would fit into this less-flexible FPGA very well, and would be smaller and faster than they would have been on the original chip. Other circuits, however, might have made good use of the programmable resources that have been removed, and will now be larger and/or slower than they would have been on the original chip.

If, however, one manufactures a *family* of related FPGA architectures, one can have the best of both worlds. The FPGA *family* is a group of chips, each of which is based on a somewhat different FPGA architecture, and each individual chip in this family is called a *sibling*. All siblings have equivalent maximum logic capacities. Instead of attempting to implement all application circuits in one very flexible FPGA chip, we use the most suitable sibling for each application circuit. Each sibling is tailored to a certain class of application circuits in some way -- say by replacing many programmable switches with hard-wired links between logic blocks and by using longer routing segments. This sibling implements certain circuits very efficiently, but its reduced flexibility means that some circuits may no longer fit into it at all. We overcome this reduced flexibility by choosing the architecture of the remaining siblings so that they can efficiently implement any circuit which will not fit into this chip well. With good choices for the architecture of each chip, a small number of siblings will be able to implement any application circuit more efficiently than a single highly flexible FPGA.

Since a single FPGA is sufficient for prototyping but even a small production run may require 50 chips, FPGA revenues come primarily from sales of chips intended for

This work was supported by the Natural Sciences and Engineering Research Council of Canada and MICRONET.

use in production hardware. This is precisely where the siblings concept would be of the greatest use. As part of the synthesis procedure, the CAD software would determine not only the technology mapping, placement and routing of the circuit, but also the best sibling to use. Since each sibling should be smaller (and hence cheaper) than a general purpose FPGA, the production volume at which it is cost-effective to switch to mask programmed logic will be increased. As well, the higher speed of siblings will allow the FPGA implementation of circuits which previously could not meet performance specifications without using custom or semi-custom logic.

The structure of this paper is as follows. Section 2 examines the relationship between FPGAs and MPGAs and shows that the sibling concept lies between these two extrema in the architectural spectrum. In Section 3 the siblings concept is applied to create a family of FPGAs composed of different types of hard-wired logic block architectures, and the performance gains are assessed. Section 4 conducts a similar experiment in which the architectures in the family contain different types of heterogeneous logic blocks. Finally, we summarize our findings and draw some conclusions in Section 5.

2 The Siblings Concept

The basic idea of siblings is to create FPGAs which have higher performance and are smaller than current FPGAs by removing some of their flexibility, i.e. some of their programmable switches. The flexibility removed from each individual sibling is recovered by having the choice of several different siblings in which to implement an application circuit.

It is instructive to consider the differences between MPGAs and FPGAs in order to see how the siblings concept fits into the spectrum of circuit implementation technologies. A single FPGA can implement any application circuit (subject only to size constraints), so only one type of FPGA is needed. A fully-fabricated MPGA, on the other hand, implements exactly one circuit, so a new MPGA is required for each new application. The MPGA achieves its higher speed and smaller area by using small, low-delay wires rather than larger and slower programmable interconnect, and by only laying out the interconnect resources required by this circuit. We view these two solutions as extremes in a spectrum of possible implementation choices, as Figure 1 shows. The idea of an FPGA family allows us to choose other points on the architectural axis of Figure 1.

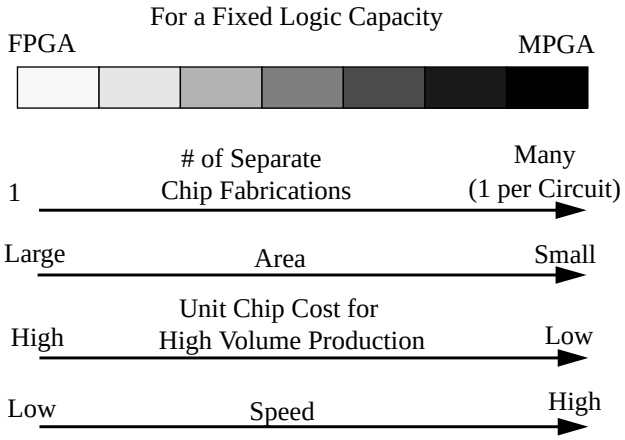


Figure 1: Speed and Density Variation with Number of Distinct Chips Fabricated.

The best choice for the number of siblings in a family must be determined experimentally; it depends on the area and speed advantage each new sibling confers and on the cost of developing each new sibling. Clearly it is desirable to have the smallest number of siblings which provides the necessary performance improvement, since smaller inventories will then have to be maintained by the vendor or user, and the family development costs will be lower. The architectural differences between siblings are also crucial. A poor set of choices will result in little or no gain in speed and density, or will require a large number of siblings to meet the performance goals. A good set of choices will provide larger speed and density gains with a smaller number of siblings.

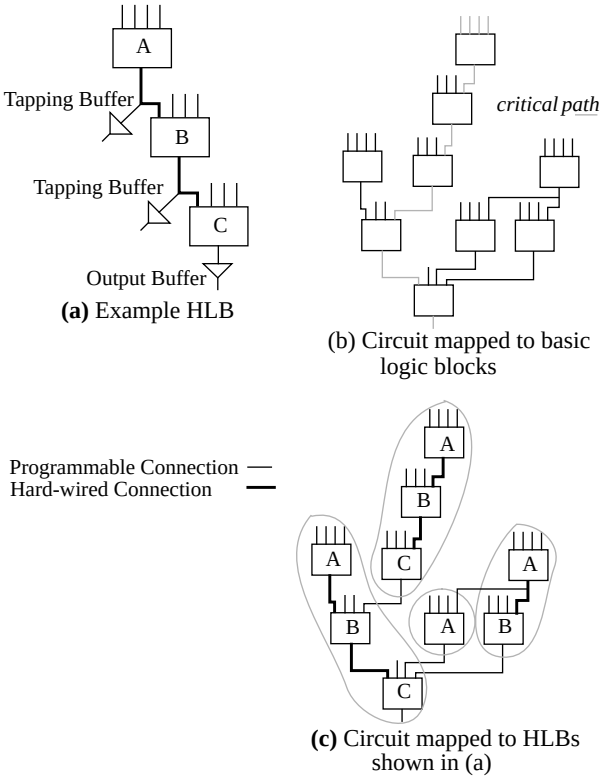
Another way of viewing the siblings concept is to observe that along any FPGA architectural axis, for example the granularity of the logic block, different circuits will have different speed and density. While there may be a good single choice based on the average behaviour over all circuits, previous experimental work has shown a significant circuit-dependent fluctuation in speed and density for any particular choice. Providing more implementation choices will reduce this variation and improve both speed and density.

3 An FPGA Family Based on Hard-Wired Logic Block Architectures

3.1 Hard-Wired Logic Blocks

A hard-wired logic block (HLB) is created by replacing some programmable connections with simple metal wires, or hard-wired connections, between logic blocks [2, 3]. Since a metal wire incurs a much smaller delay and requires less area than a programmable connection, FPGAs

built using HLBs have the potential to be both smaller and faster than FPGAs in which all connections are programmable. In this work, as in [2,3], we will consider only HLBs composed of look-up tables (LUTs). Figure 2 illustrates both an example HLB composed of three four-input LUTs (4-LUTs) hard-wired in a chain and a circuit implemented with this HLB.



(a) Example HLB

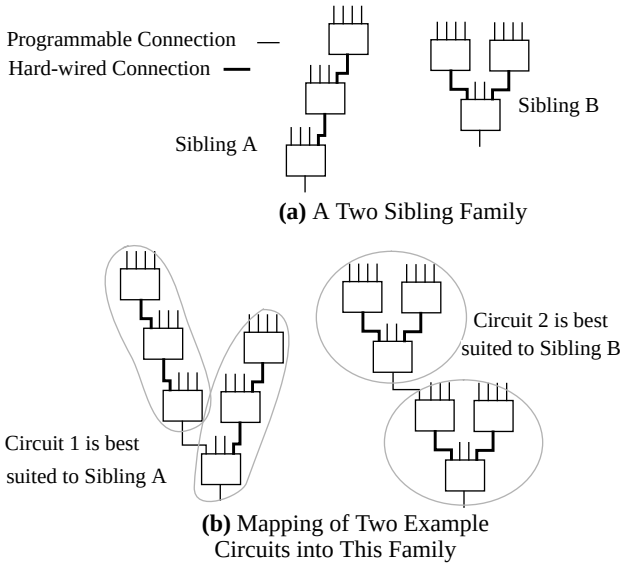
(b) Circuit mapped to basic logic blocks

(c) Circuit mapped to HLBs shown in (a)

Figure 2: An Example HLB and Circuit Implementation.

In the HLBs we study, the inputs of some LUTs are permanently hard-wired to the outputs of others. The output of each LUT in an HLB can also be directed to the programmable interconnect via the output and tapping buffers shown in the figure. Figure 2b shows an example circuit implemented with a (nonhard-wired) 4-LUT logic block, while Figure 2c shows the same circuit implemented via the HLB of Figure 2a. The circuit implemented with HLBs will be faster, since it has replaced many of the slow programmable connections on the critical path with fast hard-wired ones. The effect of HLBs on area is more difficult to predict. Since many programmable connections are replaced with simple metal wires, HLB circuit implementations can be smaller than standard LUT realizations. On the other hand, since HLBs are more coarse-grained, it is more difficult to utilize them fully, so the area required by many circuits will increase. In practice, one finds that HLBs with a large number of hard-wired connections generally cause an increase in circuit area, while HLBs with only a few hard-wired connections can reduce the circuit area slightly compared to nonhard-wired implementations [2,3].

We create families of FPGAs by simply choosing a different HLB architecture for each sibling. Figure 3a shows an example family with two siblings, and Figure 3b illustrates the fact that an application circuit is always implemented in the most suitable sibling.



(a) A Two Sibling Family

(b) Mapping of Two Example Circuits into This Family

Figure 3: An Example Family and Circuit Implementations.

3.2 Experimental Methodology

We use the results of a previous study [3] to evaluate the density and speed improvements attainable by applying the siblings concept to HLBs. The results of technology mapping a set of fifteen mcnc benchmark circuits into 209 different HLB architectures are used to compare the "goodness" of each family. Each architecture is defined by the size of its LUTs (from 2 to 7 inputs) and by the topology of the hard-wired connections between these LUTs. For each architecture we compute area and delay estimates for each circuit and normalize them to those of the same circuit implemented in a 4-LUT FPGA with no hard-wired connections. The 4-LUT has previously been shown to be a good choice for an FPGA logic block architecture [4], and by normalizing our area and delay results to it we can average results from circuits with a wide variance of sizes and logic depths in a meaningful way. All averaging is done with geometric averages, since taking the arithmetic average of normalized numbers can lead to misleading results [5].

The delay and area of a family with only one sibling are taken to the geometric averages of that FPGA's delay and area metrics over our fifteen benchmark circuits, respectively. The speed of an FPGA is simply the reciprocal of delay. In a family with more than one sibling, we define the "family score" on a circuit as the best area and/or delay

metric achieved by *any* of the siblings on that circuit. Averaging this "family score" over all the benchmark circuits yields the performance of this family.

To assess the improvement in the performance of a family as the number of siblings is increased, we must find the best family with the given number of siblings that can be constructed from our pool of possible architectures. For families with three or fewer siblings we exhaustively check the performance of all the possible families, so we are guaranteed to find the best family. With 209 architectures to choose from the number of possible families grows very rapidly as the family size increases. Hence for families with four or more siblings we limit the search space by assuming that a family of size n consists of the best family of size n-1, plus one more sibling. Therefore for four or more siblings we may not have the absolute best family that can be constructed, but our investigations have shown that the family we obtain is either the best or one with only slightly lower performance.

3.3 Area and Delay Models

The area and delay models used are deliberately kept simple in order to allow the evaluation of the FPGA architectures after technology mapping; i.e. no place and route step was undertaken. Note that only relative delay and area metrics are necessary in this study, since we are simply comparing architectures. The critical path delay consists of the logic block delay plus the delay incurred by programmable interconnect, since the delay of a hard-wired connection is essentially zero [3].

$$D_{Tot} = N_L \times D_{LB} + N_R \times D_R$$

D_{Tot} is the total delay, while N_R and N_L are the number of programmable connections and the number of logic blocks on the critical path, respectively. D_{LB} and D_R are the delays of a logic block and of a programmable connection, respectively. The architectures studied were all based on LUTs with between 2 and 7 inputs. The delays of these logic blocks have been found from SPICE simulations of 1.2 μm CMOS implementations [6], and are listed in Table 1.

TABLE 1. Lookup Table Delays in a 1.2 μm CMOS process.

# Inputs to LUT	D_{LB} (ns)
2	1.39
3	1.44
4	1.71
5	2.03
6	2.38
7	2.85

Choosing a value for D_R is more problematic, since the delay of a programmable connection varies widely depending on its fanout and the number of routing switches through which it passes. We set D_R to 4 ns, since in our experience this is a reasonable value for the type of FPGA architectures we are studying implemented in 1.2 μm CMOS. To ensure that the value chosen for D_R does not affect our conclusions, we also conduct experiments with D_R as low as 1 ns and as high as 10 ns.

Instead of referring to delay directly, we will often refer to the speedup of one FPGA with respect to another. The speedup of $FPGA_A$ with respect to $FPGA_B$ is defined as

$$Speedup_{AB} = \frac{Delay_B}{Delay_A}.$$

The area of the logic blocks in a LUT-based FPGA is mostly SRAM bits, while the routing area correlates well with the total number of pins on the logic blocks. We therefore take the area of a circuit to be proportional to [7, 8]

$$Area \propto N_{HLB}(HLB_{Bits} + Pinfac \times HLB_{Pins} + FA \times HLB_{LUTs})$$

where N_{HLB} is the number of hard-wired logic blocks required to implement the circuit and HLB_{Bits}, HLB_{Pins}, and HLB_{LUTs} are the number of bits, pins and LUTs per HLB, respectively. Pinfac is the number of logic bit equivalent area units consumed by each pin, while FA is the number of logic bit equivalent area units required by the fixed resources (a D flip flop and output buffer) associated with each LUT. In a 1.2 μm CMOS process FA is 3.4 [3], while Pinfac is approximately 14 for the type of FPGA architectures we are studying. As with the D_R parameter, however, we also conduct experiments with Pinfac set as low as 4 and as high as 30, in order to ensure that its value does not greatly influence our results.

3.4 Experimental Results

The technology mapping procedure could be set to produce either area or delay-optimized circuits, and we tested the performance of siblings on both types of circuits. For the area-mapped case, we chose the sibling architectures to minimize the area of the benchmark circuits. In the case of delay-mapped circuits, however, choosing the sibling architectures to minimize delay alone produces a family with unacceptably large areas -- typically about four times the area of a nonhard-wired 4 LUT implementation. Consequently we chose siblings to minimize the sum of the area and delay for the delay-mapped circuits. This slows down the circuits only slightly, while reducing their area by a factor of approximately 4. Figure 4 shows the speed and area improvements possible with siblings for both types of circuit mappings.

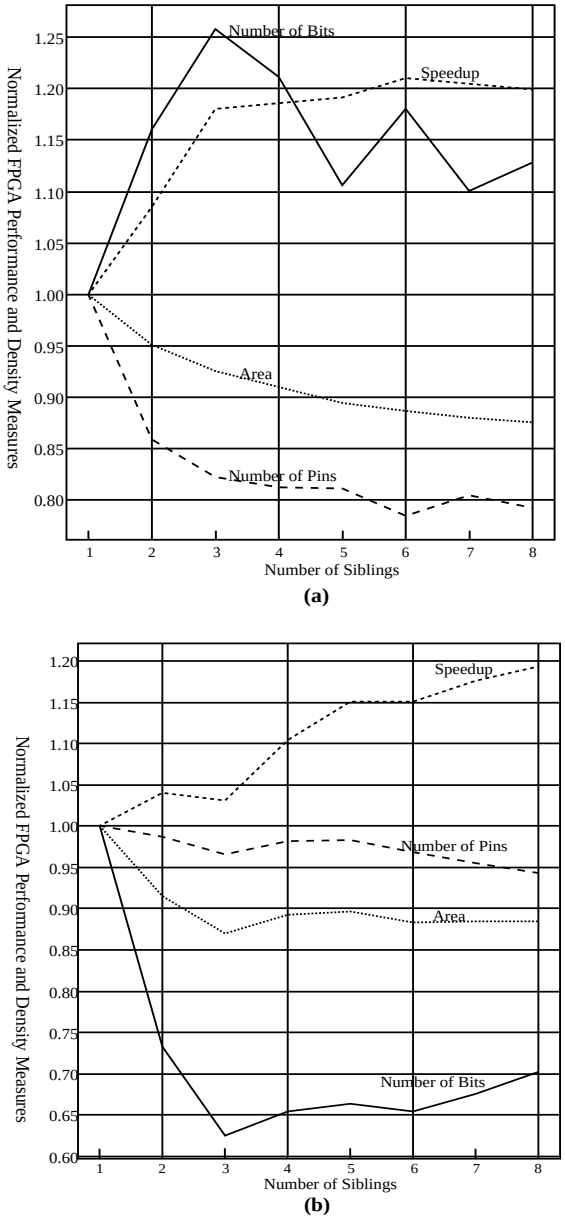


Figure 4: Improvements in FPGA Performance and Density Measures as a Function of Number of Siblings for (a) Circuits Technology Mapped to Minimize Area, and (b) Circuits Technology Mapped to Minimize Delay.

In Figure 4 we have normalized all our results to those obtained by the best single FPGA architecture (i.e. the 1 sibling case) so that the performance improvement due to increasing the size of the family is immediately apparent. From Figure 4, one sees that using a family with 8 siblings would result in FPGAs that are 12.5% smaller and 20% faster than any single FPGA for the area-mapped case, and 11.5% smaller and 19% faster for the delay-mapped case.

Figures 5 and 6 show which FPGA architectures formed the best families with between one and three siblings for the area-mapped and delay-mapped circuits, respectively. The label on a LUT denotes its size (number of inputs), heavy lines between LUTs denote hard-wired connections from the output of one to the input of the next, and the lighter lines indicate programmable connections.

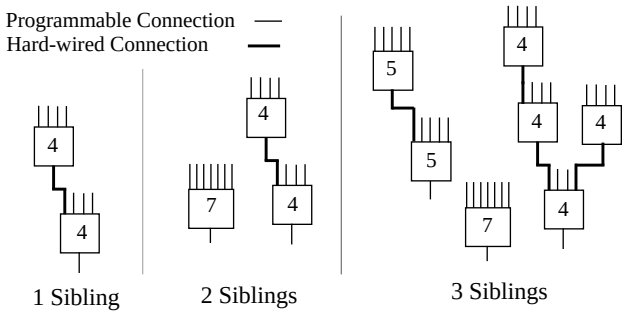


Figure 5: Architectures Constituting the Best Families for the Area-Mapped Circuits.

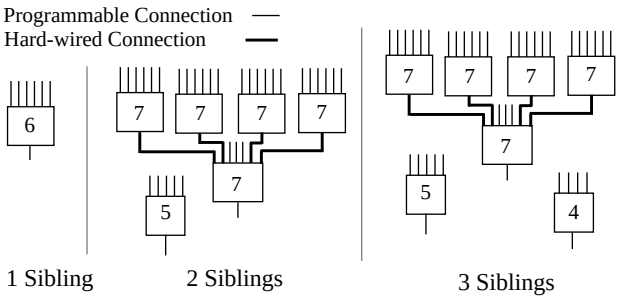


Figure 6: Architectures Constituting the Best Families for the Delay-Mapped Circuits.

The most area-efficient single HLB architecture is shown in the 1 sibling case of Figure 5. Notice that it consists of two 4-LUTs, which have previously been shown to be very area efficient [4], connected by 1 hard-wired connection. Since every LUT must fan out, the technology mapper can usually make good use of this hard-wire, so the area efficiency of this architecture is not surprising. As the number of siblings increases to 2 and 3, larger LUTs (7-LUTs and 5-LUTs) tend to be selected, and the 1 sibling 4-LUT HLB is eventually replaced by a 4-LUT HLB with 3 hard-wired connections. This is an expected result; the choice of siblings in which to implement a circuit helps to make up for the flexibility lost as we move toward coarser-grained logic blocks. Hence, the utilization of these larger logic blocks is high, and their lower routing area requirements translate into area-efficient circuits.

Figure 6 shows that a single 6-LUT is the best architecture for simultaneously minimizing the area and delay of the delay-mapped circuits. While a 6-LUT is less area efficient than a 4-LUT, it leads to faster circuits, and hence is a good choice when both area and delay must be minimized.

In Figure 6, notice that when one goes from one to two siblings, a single six-input LUT is replaced by a large 7-LUT HLB and a single five-input LUT. The six-input LUT has been replaced by one more coarse-grained HLB and one more fine-grained HLB. Clearly, when forced to use only one general-purpose FPGA architecture, we had to compromise between the two and use the six-input LUT. Similarly, one can see from Figure 5 that the single best FPGA architecture for area-mapped circuits (1 sibling) is not one of the architectures chosen for the three-sibling family; again the single architecture was a compromise choice.

Tables 2 and 3 show the effect of varying D_R and Pinfac over a wide range. Notice that the parameter we are optimizing, the area reduction for area-mapped circuits and the sum of the area and delay reductions for the delay-mapped circuits, is fairly constant regardless of the values of D_R and Pinfac. The sibling architectures chosen for each family change somewhat when the extreme values of D_R and Pinfac are used in our delay and area models, but the trends observed as the family size increases are the same.

TABLE 2. Effect of Varying D_R and Pinfac Parameters for Area-Mapped Circuits

D_R (ns)	Pin-fac	5 Sibling Area Red.	8 Sibling Area Red.	5 Sibling Speedup	8 Sibling Speedup
4.0	14	10.6%	12.5%	1.19	1.20
1.0	14	10.6%	12.5%	1.16	1.15
10.	14	10.6%	12.5%	1.21	1.23
4.0	4	9.4%	10.4%	1.02	1.01
4.0	30	14.2%	15.6%	1.09	1.07

TABLE 3. Effect of Varying D_R and Pinfac Parameters for Delay-Mapped Circuits

D_R (ns)	Pin-fac	5 Sibling Area Red.	8 Sibling Area Red.	5 Sibling Speedup	8 Sibling Speedup
4.0	14	10.3%	11.5%	1.15	1.19
1.0	14	11.7%	12.0%	1.07	1.09
10.	14	6.0%	8.7%	1.30	1.32
4.0	4	-0.5%	-0.9%	1.23	1.31
4.0	30	2.9%	6.1%	1.24	1.25

We also conducted experiments in which *both* the area-mapped and delay-mapped circuits were implemented in a single family. This case is of interest because some applications may be limited by the capacity of an FPGA, while others are limited by its speed. Consequently, manufacturers desire FPGA families capable of efficiently implementing circuits that have been optimized for either area or delay. In this case we chose the siblings to minimize the sum of area and delay, and found that a family with 8 siblings was 13.7% smaller and 18% faster than the best single HLB FPGA architecture for our benchmark circuits.

4 Performance of an FPGA Family Composed of Heterogeneous Logic Block Architectures

As described in Section 3, we construct an FPGA family by choosing an appropriate number of siblings from a pool of possible architectures. The architectural "pool" used in this section consisted of 45 different FPGAs with a heterogeneous mixture of two different types of logic blocks. As outlined in [9, 10], each FPGA has logic blocks of two different sizes. The defining parameters for each FPGA are the size of the small LUT, s, the size of the large LUT, p, and the ratio of the number of small logic blocks to large logic blocks, $r = N_S/N_P$. Figure 7 shows an example of a heterogeneous FPGA with p = 4, s = 2, and r = 2.

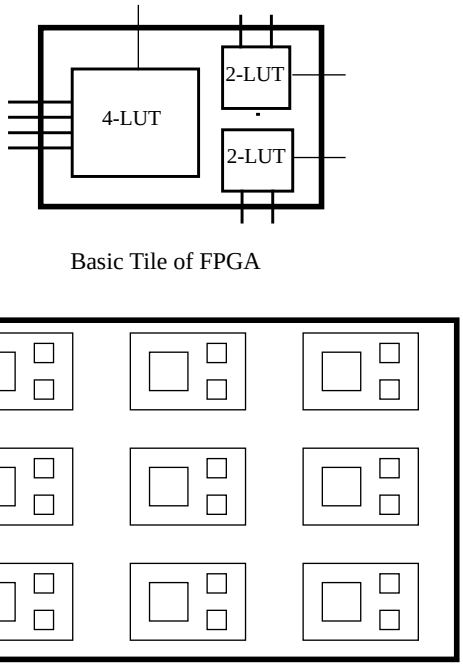


Figure 7: Heterogeneous FPGA architecture with p=4, s = 2 and r = 2.

The architectures in the "pool" used to create families have all possible heterogeneous logic block architectures with p between 3 and 7, s between 2 and 6, and r equal to 0.5, 1, or 2. The experimental procedure is very similar to that described in Section 3.1, with two exceptions. First, delay statistics were not computed in [10], so we select our families to minimize area alone, and we cannot determine the speedup. Secondly, since we are using a pool of only 45

different architectures, it is feasible to exhaustively search all possible families with up to 8 siblings in order to be sure of finding the best family.

The gains provided by applying the siblings concept to this architectural axis are not as great as those obtained from hard-wired architectures. With 8 siblings, we obtain only a 6% area reduction, even though we are choosing our families strictly to minimize area. Clearly, choosing the architectural differences between siblings wisely is crucial to obtaining large density and performance gains as the family size is increased. In order to assess the influence of Pinfac on these results, we let it vary from 4 to 30. Table 4 shows that our results are not very sensitive to this parameter.

TABLE 4. Influence of Pinfac on results.

Pinfac	5 Sibling Area Improvement	8 Sibling Area Improvement
4	4.9%	5.1%
14	5.1%	6.0%
30	5.6%	6.9%

5 Conclusions

The concept of an FPGA family allows one to explore points in the architectural spectrum between the usual extremes of general-purpose FPGAs and nonprogrammable MPGAs. We have run experiments with two families of FPGAs, and found that a family consisting of different types of hard-wired FPGA architectures outperforms one created from different heterogeneous logic block architectures. We found that a family of eight different HLB FPGAs can implement circuits in 12 to 14% less area and with 18 to 20% faster critical paths than the best single HLB architecture. While this improvement may not be enough to justify the higher development and inventory costs associated with marketing eight distinct FPGAs, we believe that greater gains can be realized by clever choices of the sibling architectures, and are pursuing further research in this direction. One promising direction of research involves varying not only the logic block architecture of the siblings, but also the routing architecture in order to increase the differences between siblings.

Acknowledgments

The authors wish to thank Dr. Kevin Chung and Mr. Jianshe He for providing and explaining the use of their data and CAD software.

References

[1] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of Field-Programmable Gate Arrays," *Proceedings of the IEEE*, Vol. 81, No. 7, pp. 1013-1029, July 1993.

[2] K. Chung, S. Singh, J. Rose, and P. Chow, "Using Hierarchical Logic Blocks to Improve the Speed of FPGAs," in **FPGAs**, W. Moore and W. Luk Eds., Abingdon 1991, pp. 103-113.

[3] K. Chung, "Architecture and Synthesis of Field-Programmable Gate Arrays with Hard-wired Connections," *Phd Dissertation*, University of Toronto, 1994.

[4] J. S. 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*, Vol. 26, No. 3, pp. 277-282, March 1991.

[5] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Communications of the ACM*, Vol. 29, No. 3, pp. 218-221, March 1986.

[6] S. Singh, J. Rose, D. Lewis, K. Chung, and P. Chow, "The Effect of Logic Block Architecture on FPGA Performance," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 3, March 1992, pp. 281-287.

[7] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, **Field-Programmable Gate Arrays**, Kluwer Academic Publishers 1992, pp. 93-96.

[8] 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.

[9] J. He, J. Rose, "Advantages of Heterogeneous Logic Block Architectures for FPGAs," *Custom Integrated Circuits Conference 1993*, pp. 7.4.1-7.4.5, May 1993.

[10] J. He, "Technology Mapping and Architecture of Heterogeneous Field-Programmable Gate Arrays," *M.A.Sc. Thesis*, University of Toronto, 1994.