On the Sensitivity of FPGA Architectural Conclusions to Experimental Assumptions, Tools, and Techniques

Andy Yan, Rebecca Cheng, Steven J.E. Wilton
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, B.C., Canada
stevew@ece.ubc.ca

ABSTRACT

Recent years have seen a tremendous increase in the capacities and capabilities of Field-Programmable Gate Arrays (FPGA's). Much of this dramatic improvement has been the result of changes to the FPGAs' internal architectures. New architectural proposals are routinely generated in both academia and industry. For FPGA's to continue to grow, it is important that these new architectural ideas are fairly and accurately evaluated, so that those worthy ideas can be included in future chips. Typically, this evaluation is done using experimentation. However, the use of experimentation is dangerous, since it requires making assumptions regarding the tools and architecture of the device in question. If these assumptions are not accurate, the conclusions from the experiments may not be meaningful. In this paper, we investigate the sensitivity of FPGA architectural conclusions to experimental variations. To make our study concrete, we evaluate the sensitivity of four previously published and well-known FPGA architectural results: lookup-table size, switch block topology, cluster size, and memory size. It is shown that these experiments are significantly affected by the assumptions, tools, and techniques used in the experiments.

1. INTRODUCTION

Since their introduction in 1985, Field-Programmable Gate Arrays (FPGA's) have seen a phenomenal growth in their ability to implement large complex digital circuits. Originally used primarily for prototyping and small glue logic replacement, FPGA's are now used to implement entire systems containing memory, embedded processors, and other embedded functionality. A 1994 databook quotes a maximum gate count of 25,000; in July 2001, a part that can implement circuits containing six million system gates was announced. The achievable clock frequency has increased over the years as well.

Much of this dramatic improvement has been the result of architectural improvements. There have been numerous academic and industrial investigations including logic block studies [1,2,5,6], routing architecture studies [7,11,14], and memory block studies [9,10]. In general, each of these studies considers one or a handful of architectural parameters in isolation, and finds "good" values for those parameters using experimentation. During the experiments, a handful of realistic benchmark circuits are typically fed through a representative CAD tool. Detailed models are then used to measure the area or delay of the circuit, and, based on these results, one of the architectures is deemed "the best". This is summarized in Figure 1.

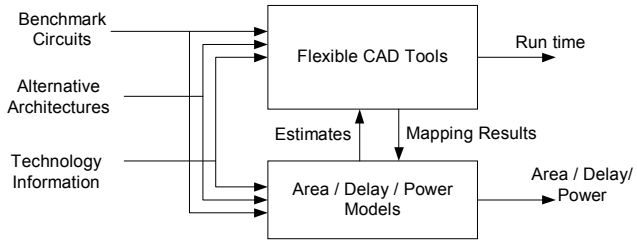


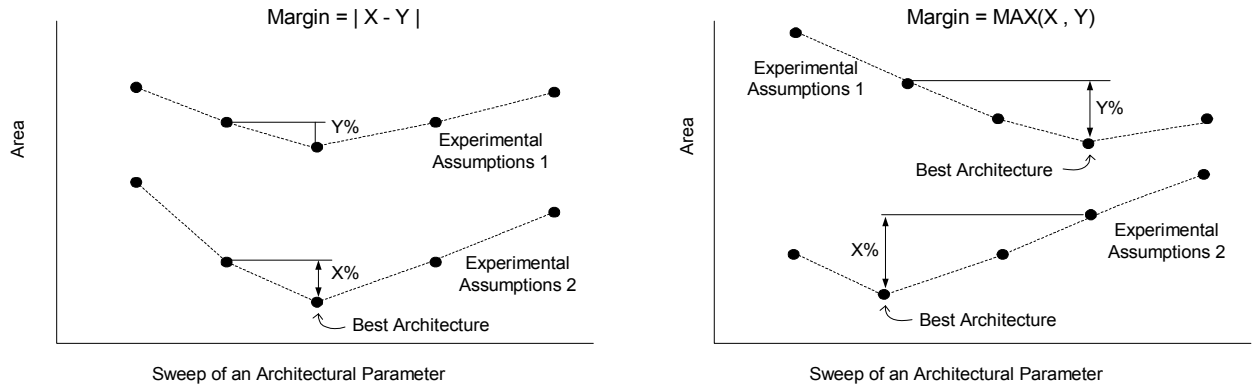
Figure 1: Experimental Framework

Relying on the results of this sort of experimentation is dangerous. No matter how careful a researcher is, assumptions and approximations must be made. In some cases, these assumptions and approximations may affect the results of the experiments, and possibly even change the conclusions of the experiments. Some of these assumptions can be categorized as follows:

CAD Tools: Clearly, the CAD tools employed for the architectural study will have a significant impact on the results. This includes not only placement and routing tools, but also the optimization and technology-mapping algorithms. In some cases, companies will run experiments using a pre-release experimental tool flow. The intention is that the final release software will be similar, but there will likely be some changes, and these changes may affect the architectural results. In academic studies, representative tools, such as Flowmap [3] and VPR [11] are often used to try to make the results as vendor-neutral as possible. Yet, these tools could lead to results that would not be seen had commercial tools been employed.

CAD Tool Settings: Most tools have numerous settings that can be used to guide the optimization algorithms. The documentation that accompanies VPR and T-VPACK has over six pages describing the run-time switches available; many of these switches will significantly affect the results of the optimization, and perhaps the conclusions of architectural experiments.

FPGA'02, February 24-26, 2002, Monterey, California, USA.



a) Case 1: If the best architecture in each experiment is the same

b) Case 2: If the best architecture in each experiment is different

Figure 2: Illustration of Margin Metric used in this paper

Experimental Techniques: There are several ways to use a CAD tool to evaluate an architecture. As an example, many researchers allow the number of tracks in each FPGA channel to "float" [11]. That is, they find the minimum number of tracks needed in each channel to successfully route a circuit, and use an FPGA with exactly that number (or a fixed multiple of that number) in comparisons. On the other hand, many commercial studies (in which the researchers have a fixed device in mind) assume a fixed number of tracks per channel. Each of these techniques may lead to different results, and perhaps different conclusions. As another example, many experiments are performed assuming the I/O connections to each benchmark circuit can be assigned to any I/O pin; others assume the pin assignment is predetermined and fixed.

Orthogonal Architecture Assumptions: When investigating the effects of one architectural parameter, it is usually necessary to fix several other parameters. As an example, when performing logic block studies, the routing fabric architecture is often fixed. Yet, it is conceivable that later changes in the routing fabric may influence the optimum logic block architecture.

In this work, we examine the sensitivity of FPGA architectural research to experimental variations. In order to make our study concrete, we focus on four previously-published fundamental FPGA architectural experiments:

1. What is the optimum lookup-table (LUT) size? [1,2,5]
2. What sort of switch block works well? [7,9,11]
3. How many lookup-tables should be included in a logic block or cluster? [2]
4. How large should the memory arrays in an FPGA be? [10]

For each of these experiments, we investigate how sensitive the conclusions are to experimental variations. It is important to note that we are not setting out to actually answer these questions; they have been answered well in the previous works, and in most cases, the conclusions are well known. Our goal is to determine how sensitive these conclusions are to experimental variations. Also note that it is the conclusions we care about; in this paper, we will see many cases when the raw data changes significantly, but the overall conclusions of the study are the same.

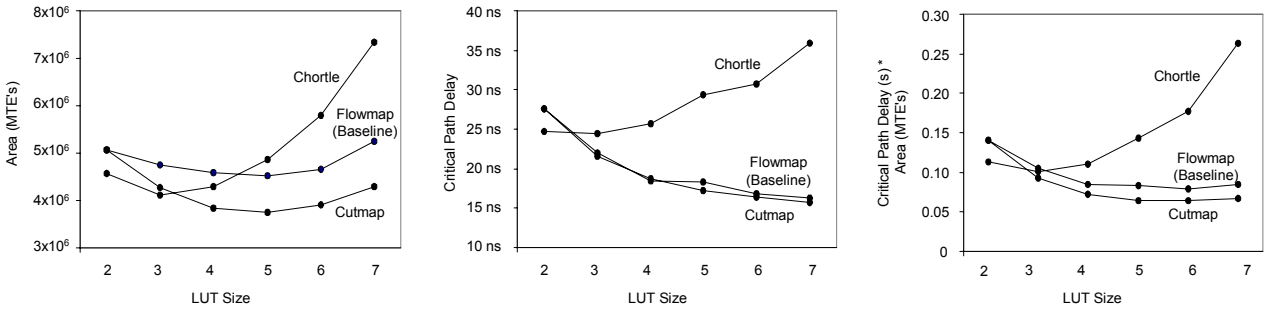
In this paper, we will focus on the first two questions. These questions speak to the very basic architecture elements within an FPGA (lookup-tables and routing). Results for Questions 3 and 4 will be summarized, but details will not be presented.

2. EVALUATION METRICS

Before focusing on each experiment in detail, this section describes how we will evaluate the sensitivity of an experiment on the assumptions, tools, and techniques. Consider a company which is considering increasing the size of the memory arrays on a given architecture. Suppose that experiments have shown that the larger memory array size will lead to better packing density. The fact that the new architecture would be better is not enough – the company also cares about how much better the new architecture is. Redesigning the memory arrays would require a significant engineering effort, and is only justified if the expected gains are significant.

This example illustrates the need to examine the effects of the experimental assumptions, tools, and techniques on not only which architecture is deemed the best, but also the margin by which that architecture is better than the others. Thus, in this paper, for each experiment, we will present graphs which show how the selection of the best architecture depends on experimental parameters, as well as measurements indicating how the margin is affected by the experimental parameters. We will quantify the latter as follows:

(1) First consider experiments in which the best architecture remains the same for different experimental assumptions, tools, and techniques. As an example, consider the fictitious area-optimization example in Figure 2(a). This figure shows a sweep of an architectural parameter on the horizontal axis, with measured area results on the vertical axis. Two experiments are shown; the experiments differ in the experimental assumptions that were made (perhaps one experiment uses the VPR routing tool, and one uses a different routing tool, for example). For several values of



a) Area Results b) Delay Results c) Area*Delay Results

Figure 3: LUT Size Experimental Results for three different technology-mappers

the architectural parameter, area measurements are made and are represented by dots and connected by dotted lines in the graph. In this example, the best architecture is in the same location for each experiment. For each experiment, we measure the percentage difference between the area of the best result and the next-best area result is measured (labeled X% and Y% in the diagram). The margin is then defined as the absolute value of the difference between X and Y. Note that a margin for the delay results can be defined similarly.

(2) Now consider an experiments in which the best architecture is not the same as the experimental assumptions are changed. Figure 2(b) shows a fictitious example. Again, we have two experiments, this time giving different conclusions. Suppose the two best areas are at points A (for experiment 1) and B (for experiment 2). For experiment 1, we work out the percentage difference between the area at points A and B (this is labeled X% in the diagram). We then do the same for experiment 2 (labeled Y%). The margin, in this case, is the maximum of X and Y. Again, note that a margin for the delay results can be defined similarly.

Intuitively, this definition leads to a high margin if the experimental conclusions are significantly affected by a change in the experimental assumptions, and a low margin if the conclusions are not significantly affected. The definition suffers from the fact that the margin depends on the number of values considered for the architectural parameter (the spacing between points in Figures 2(a) and 2(b)). Despite this, we can still draw significant conclusions from the margin metric, and thus will use it in this paper.

3. LOOKUP-TABLE SIZE

Most FPGA's use lookup-tables as their basic logic units. One of the fundamental decisions an FPGA architect must make is what size these lookup tables should be (size is usually measured in terms of the number of inputs to each lookup-table). In this section, we consider an experiment to find the best lookup-table size for an FPGA. Such experiments have been reported in [6] and later [2]. Intuitively, a smaller lookup table consumes less chip area and is faster, however, more lookup-tables (and the associated routing) are required to implement a circuit. Previous experiments have suggested that lookup-tables with 4-6 inputs provide the best balance between these competing factors. In this section, we seek to determine how sensitive these conclusions are to various experimental assumptions, tools, and techniques.

In this section, we consider the following "baseline" experiment. Twenty circuits were optimized using SIS (choosing the best of script.rugged and script.algebraic) and technology-mapped to LUT's using Flowmap and Flowpack [3]. The circuits were then placed and routed on an FPGA using VPR [11]. An FPGA with four lookup-tables per cluster, and routing segments of length 4 was targeted. For each circuit, the minimum number of tracks per channel was found, this number was increased by 30%, and the routing repeated. The critical path delay and the area, in terms of Minimum Transistor Equivalents (MTE's), was measured. This flow is similar to that used in many previous architecture studies [2,10,11].

3.1 CAD Tool Effects

There are two sets of CAD tools used in the baseline experiments described above. First, consider the role of the technology-mapper. This tool packs logic into lookup-tables. We repeated the above baseline experiment, but replaced Flowmap with two other technology mappers [4][16]. Figure 3 shows the area, critical path delay, and the product of the area and the critical path delay as a function of LUT size, averaged (geometric average) over twenty large circuits, for each of the technology-mappers. The margin metric, as described in Section 2, is summarized in Table 1 and will be discussed in Section 3.5.

As stated in the introduction, the purpose of the data in Figure 3 is not to compare the quality of the technology-mapping tools, nor is it to actually determine the best lookup-table size. These have been well studied in previous work. Instead, the purpose of the data in Figure 3 is to determine whether or not the choice of LUT-size (the conclusions of the experiments) would be influenced by the technology-mapper used in the experimentation. As the data shows, the choice of LUT size is significantly affected by the technology-mapper. If Chortle is used, the most area-efficient LUT has 3 inputs, while if Flowmap or Cutmap is used, the most area-efficient LUT has 5 inputs. In terms of delay, the conclusions are also very different: if Chortle is used, a smaller LUT is preferred, while if Flowmap or Cutmap are used, a larger LUT is a better choice. Although Chortle has been around for several years, and we would not expect it to perform as well as Flowmap or Cutmap, it is still available, and thus it is conceivable

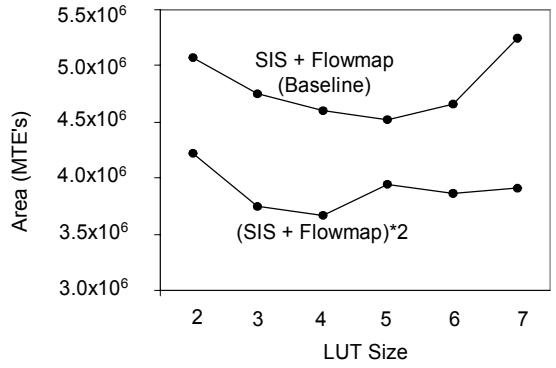


Figure 4: LUT Size Experimental Results for two different circuit optimization schemes

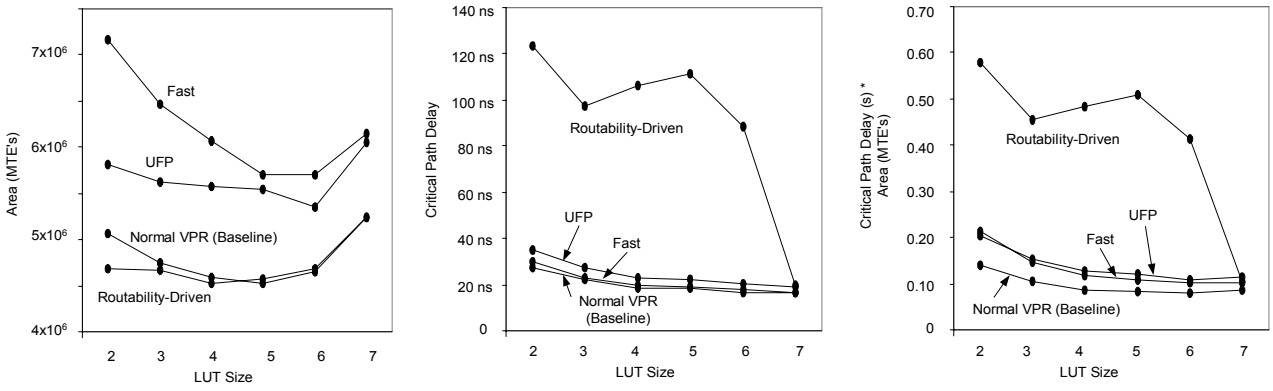
that we would see experimental results gathered using this CAD tool. The above graphs suggest that such architectural conclusions should be viewed with suspicion.

Figure 4 shows another interesting comparison. In the baseline experiment, we optimized each circuit using SIS and then technology-mapped the circuit with Flowmap. We repeated this experiment, but after running Flowmap, we re-optimized the Flowmap'ed circuits (using the same SIS scripts) and re-technology-mapped each circuit using Flowmap (in other words, each circuit was optimized and mapped twice). As shown in Figure 4, this has a significant effect on the area conclusions: in the baseline experiment, the most area-efficient LUT has 5 inputs, while if the circuits are optimized twice, the most area-efficient LUT has 4 inputs (and a 5-input LUT is a particularly bad choice). Delay results are virtually the same for both experiments, and are thus not shown. Very few published architectural results make any more than a brief mention of how the benchmark circuits were optimized; the results in Figure 4 show that this optimization is important, and must be considered carefully.

A place and route tool is also an integral part of the experiment. We repeated the experiment using three alternative place and route algorithms (in addition to VPR run in it's "normal" mode): (1) VPR in "fast" mode, in which fewer placement and routing iterations are performed, (2) VPR in "routability-driven" mode, in which timing is not one of the primary optimization goals, and (3) the Ultra-Fast Placer (UFP) described in [13] which places circuits using a constructive algorithm followed by a low-temperature anneal followed by a standard timing-driven VPR routing algorithm. Figure 5 shows the results. The most area-efficient LUT size is 6 if VPR in "fast mode" or the Ultra-fast placer is used, while the most area-efficient LUT-size is 5 if normal VPR is used, and 4 if the routability-driven router is used. The delay results show a dramatic difference between the routability-driven results and the results from the other place and route tools. This illustrates the danger when using routability-driven tools and measuring timing results.

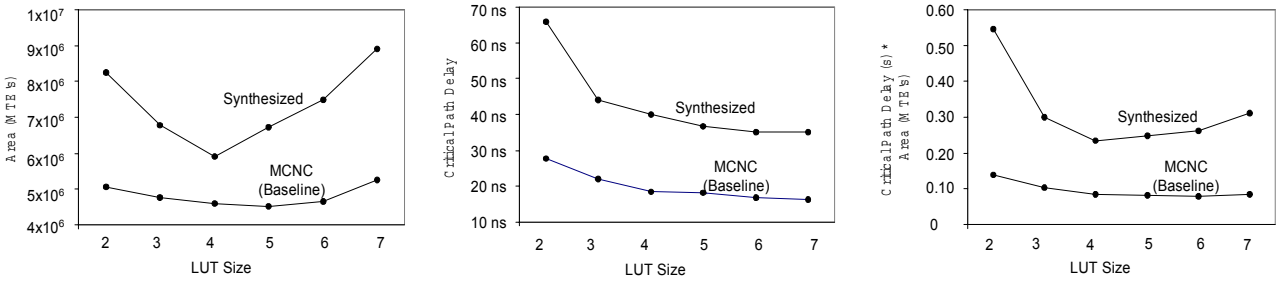
3.2 Benchmark Circuits

The architectural conclusions are also dependent on the circuits employed. The data presented in the previous subsection was gathered using 20 large combinational and sequential benchmark circuits obtained from the Microelectronics Center of North Carolina (MCNC). We repeated the experiments, but used 8 large benchmark circuits synthesized directly from VHDL or Verilog. As shown in Figure 6, the synthesized circuits show the same trends, although the area results show a significantly steeper slope below and above the best area architecture. In many cases, architectural decisions are made based on the product of area and delay results; the third graph in Figure 6 shows that if the synthesized circuits were used in experimentation, a LUT size of 4 would likely be chosen, while if the MCNC circuits were used, a larger LUT size would appear to be a better choice. This may indicate why commercial FPGAs typically have small (3 or 4-input) lookup tables, even though academic studies predict that larger lookup-tables would be better; most FPGA companies have access to a large number of user circuits, beyond the MCNC circuits. This highlights the need for a new suite of benchmark circuits that better reflects the types of circuits used by today's FPGA customers.



a) Area Results b) Delay Results c) Area*Delay Results

Figure 5: LUT Size Results for four different placement and routing tools



a) Area Results b) Delay Results c) Area*Delay Results

Figure 6: LUT Size results for two different benchmark suite

3.3 Experimental Method

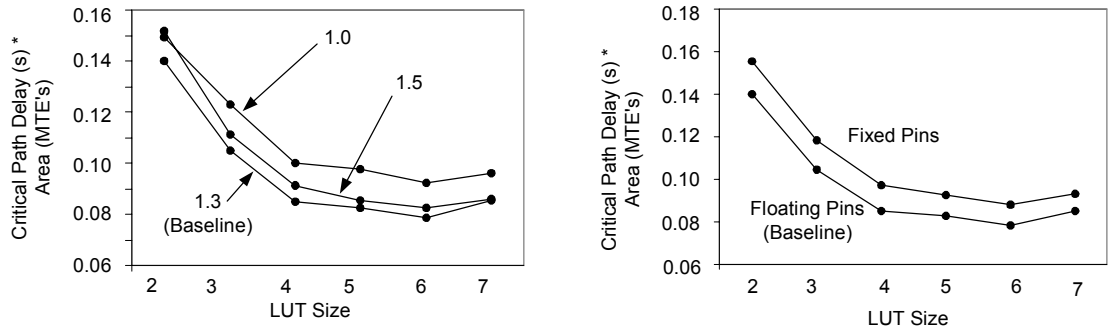
Equally important as the CAD tools and the benchmark circuits is the manner in which these tools and circuits are used in the experimentation. We investigated two modifications to the above baseline experimental flow. In the baseline flow, we find the minimum number of tracks needed to route each circuit, multiply this number by 1.3, and re-route the circuit to obtain timing numbers. In Figure 7(a), we show the area*delay results for several values of this multiplier. As the graph shows, this multiplier has little effect on the architectural conclusions. Figure 7(b) shows the area delay results if we repeat the baseline experiments, but fix the pins randomly before place and route. Again, there is little effect on the architectural conclusions.

3.4 Orthogonal Architecture Assumptions

Intuitively, the best architecture for a logic block will depend on the routing fabric. A routing fabric that is flexible means smaller lookup-tables are a better choice, since cascading them to implement larger functions is easier. On the other hand, the larger and slower the routing fabric, the larger the best LUT size, since more logic can be packed into each logic block. We repeated the baseline experiment, first varying the number of accessible tracks per logic block pin (Fc in the terminology of [11]), and then varying the length of each wiring segment (the length of a wiring segment is the number of logic blocks spanned by the segment). Figure 8 shows the area results for both sets of experiments. Clearly, the choice of the most area-efficient LUT does depend on the value of Fc and the segment length. The baseline experiment indicates that a LUT size of 4 or 5. On the other hand, the most area-efficient choice is 6 if Fc is 1.0 (meaning every track in a neighbouring channel is accessible by every logic block pin), or if Fc is 0.3 (meaning only 30% of the tracks in a neighbouring channel are accessible to each logic block pin). The best choice is also very slightly affected by the choice of segment length; if the segment length is 8, the most area-efficient LUT size is 4, while if the segment length is 1, the most area efficient LUT size is 6. Note that the vertical scale on these graphs is relatively small; Section 3.5 will show that the margin (as defined in Section 2) is small for these experiments. The delay results are not shown; the delay conclusions show little sensitivity to either the value of Fc or the segment length.

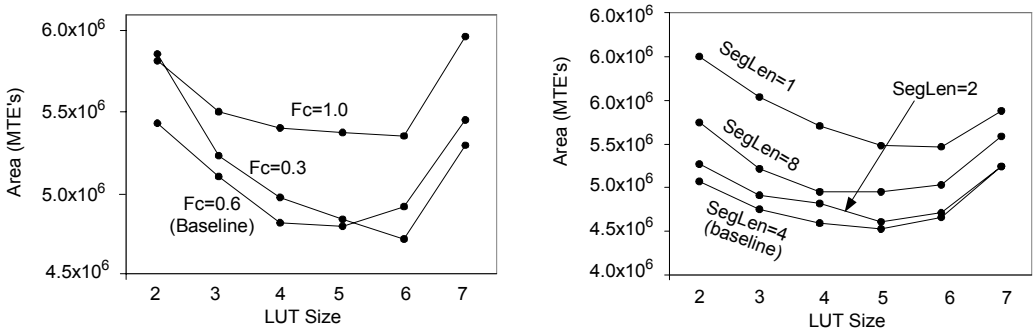
3.5 Summary: Quantitative Measurements

Table 1 summarizes the margin (as defined in Section 2) for each experiment. Each experiment is categorized as "not sensitive" (margin less than 2%), "slightly sensitive" (margin between 2% and 5%), "sensitive" (margin between 5% and 10%), "very sensitive" (margin between 10% and 100%), or "extremely sensitive" (margin more than 100%) based on the area*delay margin measurement. Clearly, the boundaries between these categories is subjective, however, the categories do help give an intuitive feel for how sensitive the conclusions are to the various experimental assumptions. It is interesting that no significant trend is seen: experiments labeled "very sensitive" appeared when the CAD tool was varied, the benchmark circuits were varied, the experimental techniques were varied, and the orthogonal architecture assumptions were varied. The large number of these "very sensitive" experiments clearly indicates that the LUT size conclusions are quite sensitive to the architectural assumptions, tools, and techniques.



a) Channel Multiplier b) Fixed vs. Floating Pins

Figure 7: LUT Size Results for several different experimental methodology changes



a) Connection Block Flexibility b) Segment Length

Figure 8: LUT Size Results for several different orthogonal architecture assumptions

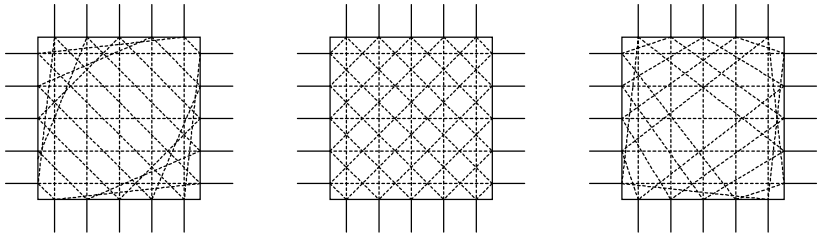
Modifications to Experimental Assumptions	Margin (compared to Baseline Experiment)			Qualitative Comment
	Area	Delay	Area*Delay	
Use Chortle instead of Flowmap	18 %	47 %	76 %	Very sensitive
Use Cutmap instead of Flowmap	1.1 %	0.57 %	4.2 %	Slightly sensitive
Optimize and Technology Map Circuits Twice	7.3 %	4.4 %	8.5 %	Sensitive
Use Fast Option of VPR	2.9 %	2.4 %	3.6 %	Slightly sensitive
Use Ultra-Fast Placer [13]	3.3 %	3.6 %	0.08 %	Not sensitive
Use Routability-Driven Place and Route	1.5 %	344 %	301 %	Extremely sensitive
Used Synthesized Circuits rather than MCNC	14 %	3.8 %	11 %	Very Sensitive
Measure results using minimum channel width	1.5 %	7.3 %	1.0 %	Not Sensitive
Multiply minimum channel width by 1.1	0.25 %	1.3 %	5.4 %	Sensitive
Multiply minimum channel width by 1.2	0.77 %	2.1 %	4.8 %	Slightly Sensitive
Multiply minimum channel width by 1.4	0.39 %	1.4 %	2.1 %	Slightly Sensitive
Multiply minimum channel width by 1.5	0.21 %	3.4 %	1.4 %	Not Sensitive
Use Fixed, Predetermined Pin Locations	0.70 %	3.9 %	0.22 %	Not Sensitive
Use F_c=0.3 rather than F_c=0.6	2.7 %	0.46 %	5.7 %	Sensitive
Use F_c=0.4 rather than F_c=0.6	23 %	2.9 %	11 %	Very Sensitive
Use F_c=0.5 rather than F_c=0.6	2.4 %	1.7 %	3.7 %	Slightly Sensitive
Use F_c=0.7 rather than F_c=0.6	0.26 %	2.0 %	5.5 %	Sensitive
Use F_c=0.8 rather than F_c=0.6	1.0 %	6.6 %	11 %	Very Sensitive
Use F_c=0.9 rather than F_c=0.6	0.37 %	4.4 %	2.7 %	Slightly Sensitive
Use F_c=1.0 rather than F_c=0.6	2.4 %	2.6 %	4.8 %	Slightly Sensitive
Use Segments of length 1 rather than length 4	2.9 %	4.6 %	8.5 %	Sensitive
Use Segments of lengt 2 rather than length 4	0.58 %	2.0 %	0.028 %	Not Sensitive
Use Segments of length 8 rather than length 4	1.5 %	3.4 %	4.3 %	Slightly Sensitive

Table 1: Margin Results for LUT size experiments

4. SWITCH BLOCK

Another fundamental question when designing an FPGA is how the logic blocks should be connected. The development of a flexible, yet fast and small routing fabric is important, and has been well studied [7,9,11,14]. A key question is what sort of switch block works well. A switch block is a flexible interconnect block that lies at the intersection of every horizontal and vertical channel [11]. The switch block can be configured to connect each incoming track to some number (typically three) of outgoing tracks. The topology of the switch block, ie. exactly which three output tracks are accessible from a given input track, has a significant effect on the routability of the chip, and hence the area and delay of circuits implemented on the FPGA.

Four switch blocks have been proposed in previous literature: the Disjoint switch block [12], the Wilton switch block [9], the Universal switch block [14], and the Masud switch block [7]. The first three switch block topologies are summarized in Figure 9; a dotted line represents a potential programmable connection between incident tracks. The Masud block uses the Disjoint pattern for all segments that pass through a switch block and the Wilton pattern for all segments that terminate at a switch block see [7] for details).



a) Disjoint b) Universal c) Wilton

Figure 9: Switch Block Types

The first three switch block patterns are well compared in [11]. That work concluded that the Wilton switch block worked well for architectures which contained only single-length segments (ie. routing segments that span only one logic block), however, for FPGAs with larger segments (which are the norm), the Disjoint block works better. The Masud block was compared to the other blocks in [7]; that paper concluded that the Masud block provided an improvement in density without any significant effect on speed. In this section, we seek to determine how well these conclusions hold for a variety of architectural tools, techniques, and assumptions.

The baseline experiment consists of placing and routing 20 large benchmark circuits using timing-driven VPR. The circuits were optimized as in Section 3.0. For each pattern and for each circuit, the minimum channel width required to route the circuit was found. For each circuit, the minimum channel width was then increased by 30%, and the routing repeated. Detailed area and delay models were used to evaluate each implementation. A routing fabric consisting of segments of length four was assumed, and it was assumed that 50% of the routing switches contain re-powering buffers, and 50% are simply pass transistors (this was shown to work well in [11]). Fc, the proportion of the tracks in an adjacent channel to which each logic block pin can be connected was set to 60%. This is the same methodology used in [11] and [7].

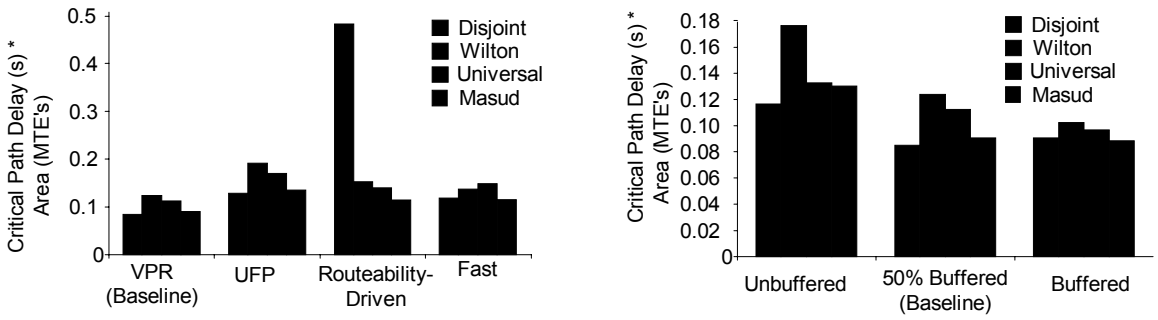
Figure 10(a) shows the area*delay results for four different placement and routing tools (the same tools used in Section 3.1). As the data shows, the choice of the placement and routing tool has little impact on the conclusions of the experiment, with one notable exception. If the routability-driven placement and routing tool is used, the area*delay of the Disjoint switch block is well over twice that of any of the other switch blocks, a behaviour not seen when using any of the other tools. As explained in [11], the pattern of the disjoint block is such the routing fabric is divided into "domains"; each connection between logic blocks can only use tracks within a single domain. As explained earlier, we assumed an architecture with 50% pass transistors and 50% re-powering buffers. The architecture generator in VPR is such that all switches within a given domain are either all pass transistors or all re-powering buffers. The routability-driven router does not understand the difference between pass transistors and re-powering buffers, and hence may choose to use a domain consisting of only pass transistors for a long wire, leading to very slow circuits. The other three switch blocks do not divide the routing fabric into segments, however, so this behaviour is not seen (in those cases, there will be some pass-transistors and some re-powering buffers on any long path between logic blocks). The other tools don't show this kind of behaviour, even with the Disjoint block is used, since they are intelligent enough to not use routing domains consisting only of pass transistors for long connections. We repeated the experiment, but for different mixes of pass transistors and re-powering buffers, and found that the behaviour illustrated in Figure 10(a) disappears. This is an excellent example of the main thesis of this paper: small changes in the experimental tools can significantly effect the conclusions of an architecture study.

Another interesting observation can be made by comparing the results of the baseline experiment in Figure 10(a) to the conclusions in [7]. Although it is difficult to deduce from the graph, the baseline experiment shows that the Disjoint switch block is slightly better than the Masud block, while [7] concluded the opposite. The difference is, again, due to the assumption regarding the mix of pass-transistors and buffers. In [7], it was assumed all segments are buffered. Figure 10(b) shows the delay results if we repeat our baseline experiment (a) when all switches are unbuffered and (b) all switches are buffered. The rightmost set of bars (the buffered results) matches those in [7]. The fact that the other two sets of bars lead to different conclusions strengthens our position that the experimental results in this particular experiment can be affected by small changes in the experimental assumptions – in this case, small changes in the assumptions regarding buffered/unbuffered switches.

We also investigated the impact of using different experimental methodologies (as in Section 3.2) and different orthogonal assumptions (including values of Fc) but found that the conclusions were not strongly affected by these results. The graphs are not shown here, but are summarized in Table 2 which shows the margin for the experimental variations that we investigated. Again, each experimental variation was labeled as "not sensitive", "slightly sensitive", "sensitive", "very sensitive" and "extremely sensitive", depending on the area*delay margin. Note that most changes were deemed "not sensitive" or "slightly sensitive". The only entry labeled "extremely sensitive" was when the routability-driven place and route tool is used instead of the timing-driven VPR, as was explained above.

5. CLUSTER SIZE

In most FPGA's, lookup-tables are grouped into clusters (called CLB's in the Xilinx parts and LAB's in Altera parts). Connections between LUT's within a cluster are significantly faster than connections between clusters. Intuitively, the larger the cluster, the fewer cluster-to-cluster connections required, leading to a more area-efficient and faster architecture. On the



a) Place and Route Tool

b) Buffered Assumptions

Figure 10: Switch Block Results

Modifications to Experimental Assumptions	Margin (compared to Baseline)			Qualitative Comment
	Area	Delay	Area*Delay	
Use Fast Option of VPR	9.3 %	3.4 %	6.8 %	Sensitive
Use Ultra-Fast Placer [13]	1.5 %	13 %	1.2 %	Not Sensitive
Use Routability-Driven Packing, Place & Route	1.7 %	330 %	320 %	Extremely Sensitive
Used Synthesized Circuits rather than MCNC	1.0 %	7.9 %	7.5 %	Sensitive
Measure results using minimum channel width	0.21 %	16 %	2.0 %	Slightly Sensitive
Multiply minimum channel width by 1.5	0.23 %	2.4 %	2.2 %	Slightly Sensitive
Implement on a double-sized FPGA	0.03 %	2.5 %	7.5 %	Sensitive
Use F_c=0.3 rather than F_c=0.6	1.9 %	1.2 %	1.8 %	Not Sensitive
Use F_c=0.5 rather than F_c=0.6	0.5 %	2.3 %	1.8 %	Not Sensitive
Use F_c=0.7 rather than F_c=0.6	1.0 %	1.2 %	1.1 %	Not Sensitive
Use F_c=0.9 rather than F_c=0.6	1.0 %	3.0 %	1.0 %	Not Sensitive
Use Segments of length 1 rather than length 4	6.3 %	18 %	33 %	Very Sensitive
Use Segments of length 2 rather than length 4	2.2 %	4.6 %	1.2 %	Not Sensitive
Use Segments of length 8 rather than length 4	0.64 %	4.5 %	3.8 %	Slightly Sensitive
Assume all switches buffered	4.6 %	5.3 %	6.8 %	Sensitive
Assume all switches unbuffered	9.6 %	3.2 %	4.5 %	Slightly Sensitive

Table 2: Margin Results for Switch Block Experiments

other hand, if the cluster is too large, the local connections within a cluster will become slow. Previous work has found that a good choice for the cluster size is between 4 and 10 [2]. In this work, we revisited this conclusion to find out how well it holds for a range of experimental assumptions.

We considered the following "baseline" experiment. Twenty circuits were optimized using SIS (choosing the best of script.rugged and script.algebraic) and technology-mapped to 4-input LUT's using Flowmap [3]. The circuits were then packed into clusters using T-VPACK and placed and routed on an FPGA using VPR [11]. An FPGA with routing segments of length 4 was targeted. The Disjoint switch block was assumed, and it was assumed that each logic block pin can connect to 60% of the tracks in an adjacent channel (Fc=0.6 using the terminology of [11]). For each circuit, the minimum number of tracks per channel was found, this number was increased by 30%, and the routing repeated. The critical path delay and the area, in terms of Minimum Transistor Equivalents (MTE's) [11], was measured, for several values of the cluster size. The number of inputs to each cluster was scaled up with the cluster size. From this data, the cluster sizes that resulted in the best area and delay implementations were deemed "the best".

Table 3 summarizes our results. As before, each experiment is categorized as "not sensitive" (margin less than 2%), "slightly sensitive" (margin between 2% and 5%), "sensitive" (margin between 5% and 10%), and "very sensitive" (margin more than 10%). Note that, in all but two cases, the experiments are classified as "not sensitive" or "slightly sensitive". This is contrast to the LUT size results in Section 3, in which significantly more experiments are classified as "very sensitive" (or even "extremely sensitive"). Thus, we conclude that, overall, the cluster size experiments are not nearly as sensitive to the experimental tools, techniques, and assumptions, compared to the LUT size experiments.

6. MEMORY ARRAY SIZE

On-chip storage has become an essential part of all modern FPGA's. Typically, current FPGA's contain large memory arrays which provide a dense implementation of storage (compared to implementing storage in the flip-flops within each logic element). However, the use of embedded memory arrays require the FPGA vendor to partition the chip area into memory regions and logic regions when the chip is designed. Since circuits have widely-varying memory requirements, this "average case" partitioning may result in poor device utilizations for logic-

Modifications to Experimental Assumptions	Margin (compared to Baseline Experiment)			Qualitative Comment
	Area	Delay	Area*Delay	
Use Chortle instead of Flowmap	2.6 %	2.3 %	1.5 %	Not Sensitive
Use Cutmap instead of Flowmap	6.5 %	3.4 %	2.6 %	Slightly Sensitive
Use Fast Option of VPR	9.5 %	3.7 %	1.5 %	Not Sensitive
Use Ultra-Fast Placer [13]	2.6 %	0.9 %	3.1 %	Slightly Sensitive
Use Routability-Driven Packing, Place & Route	2.6 %	2.8 %	9.8 %	Sensitive
Used Synthesized Circuits rather than MCNC	4.6 %	4.1 %	0.21 %	Not Sensitive
Measure results using minimum channel width	0.0036 %	2.7 %	3.7 %	Slightly Sensitive
Multiply minimum channel width by 1.5	0.30 %	1.9 %	2.2 %	Slightly Sensitive
Use Fixed, Predetermined Pin Locations	2.6 %	0.10 %	4.4 %	Slightly Sensitive
Use F_c=0.3 rather than F_c=0.6	4.9 %	4.6 %	5 %	Slightly Sensitive
Use F_c=0.4 rather than F_c=0.6	2.4 %	4.6 %	1.4 %	Not Sensitive
Use F_c=0.5 rather than F_c=0.6	0.69 %	4.6 %	1.7 %	Not Sensitive
Use F_c=0.7 rather than F_c=0.6	2.4 %	4.6 %	0.31 %	Not Sensitive
Use F_c=0.8 rather than F_c=0.6	4.4 %	3.4 %	1.7 %	Not Sensitive
Use F_c=0.9 rather than F_c=0.6	6.8 %	4.6 %	2.6 %	Slightly Sensitive
Use F_c=1.0 rather than F_c=0.6	5.7 %	4.6 %	1.5 %	Not Sensitive
Use Segments of length 1 rather than length 4	0.92 %	5.8 %	19 %	Very Sensitive
Use Segments of length 2 rather than length 4	0.48 %	2.1 %	3.6 %	Slightly Sensitive
Use Segments of length 8 rather than length 4	1.8 %	1.6 %	2.7 %	Slightly Sensitive

Table 3: Results for Cluster Size Experiments

Modifications to Experimental Assumptions	Margin (compared to Baseline Experiment)	Qualitative Comment
Use SMAP-d rather than SMAP	0.16 %	Not Sensitive
Use EMBPACK rather than SMAP	53 %	Very Sensitive
Use Blocking Factor 2 rather than 1	0.65 %	Not Sensitive
Use Blocking Factor 4 rather than 2	2.8 %	Slightly Sensitive
Use Blocking Factor 8 rather than 4	2.8 %	Slightly Sensitive
Use Chortle instead of Flowmap	17 %	Very Sensitive
Optimize and Technology Map Circuits Twice	1.4 %	Not Sensitive
Assume FPGA has 3-LUTs rather than 4-LUTs	0.81 %	Not Sensitive
Assume FPGA has 5-LUTs rather than 4-LUTs	1.1 %	Not Sensitive

Table 4: Results for Memory Size Experiments

intensive or memory-intensive circuits. In particular, if a circuit does not use all the available memory arrays to implement storage, the chip area devoted to the unused arrays is wasted.

This chip area need not be wasted, however, if the unused memory arrays are configured as ROM's and used to implement logic. Two tools have been published that map logic into unused memory arrays: SMAP [8] and EMBPACK [15]. Regardless of the tool used, the architecture of each memory array (in particular, the number of bits in each array) will have a significant impact on the ability of the tools to pack logic into the memories. If a memory array is too large, the mapping tool may be unable to effectively fill the memory array with logic. On the other hand, if a memory array is too small, the area overhead due to the decoders, sense amplifiers, etc., becomes significant.

In [10], a study was presented which seeks to find the best size of a memory array used to implement logic. That paper concluded that the best memory array size was 2Kbits. In this work, we revisited this experiment and investigate how sensitive that conclusion is to experimental techniques, tools, and assumptions.

Table 4 summarizes our results. As before, each experimental modification is classified according to how sensitive the conclusions are on that experimental modification. Overall, two modifications were shown to be "Very Sensitive": the use of EMBPACK rather than SMAP, and the use of Chortle rather than Flowmap. Thus, we conclude that if this experiment is used to choose a memory array size for a commercial chip, it is important that the CAD tool used in this experiment closely match the CAD tool that will be used in the final production software.

7. CONCLUSIONS

The main message of this paper is this: experimental assumptions, tools, and techniques can have a significant impact on the conclusion of FPGA architectural experiments, and need to be considered carefully when conclusions are presented. We have shown, through several examples in this paper that some of the "traditional", well known architectural conclusions can be significantly changed, just by changing some of the assumptions, tools, and techniques used in the experimentation. A study that

presents an optimum architecture is not enough; there must be some notion of how sensitive the results are.

In this paper, we have illustrated this using four well-known architecture results. First, we examined how sensitive the lookup-table size is to various experimental variations. Overall, we found that the optimum LUT-size did depend on several factors: in particular, we found that the CAD tools employed (both the technology-mapper and the placement and routing tool) could significantly skew the conclusions. The best LUT size could range from three to seven, depending on the CAD tools used. It was also determined that conclusions can be influenced by the benchmark circuits used and the architecture of the FPGA's routing fabric.

We also examined how the choice of switch block could be influenced by the experimental assumptions, tools, and techniques. Overall, the conclusions of this experiment held up better than the LUT size conclusions as various experimental assumptions were changed, however, we did see an example of how the experimental results could be severely impacted by using a routability-driven tool rather than a timing-driven tool.

Finally, we investigated how the choice of the optimum cluster size and memory array size is impacted by experimental assumptions. The cluster size experiment was deemed to be not as sensitive as the others, however the memory size experiment was found to be very sensitive to the packing tools employed.

8. ACKNOWLEDGEMENTS

Funding was provided by the Natural Sciences and Engineering Research Council of Canada, Altera, and Micronet. The authors wish to thank Jason Cong for providing the RASP package, Yaska Sankar for providing the ultra-fast placement tool, Bob Francis for providing Chortle, and Vaughn Betz and Jonathan Rose for providing VPR.

REFERENCES

[1] A. Marquardt, V. Betz, and J. Rose, Speed and Area Trade-offs in Cluster-Based FPGA Architectures, IEEE Transactions on VLSI Systems, vol. 8, pp. 84-93, Feb, 2000.

[2] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," in ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Feb. 2000, pp. 3-12.

[3] J. Cong and Y. Ding, Flowmap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, pp. 1-12, Jan, 1994.

[4] J. Cong and Y. Hwang, "Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping," Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 1995, pp. 68-74.

[5] J. Rose, R.J. Francis, D. Lewis, and P. Chow, Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency IEEE Journal of Solid-State Circuits, vol. 25, pp. 1217-1225, Oct, 1990.

[6] J. Rose, R.J. Francis, P. Chow, and D. Lewis, "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays," IEEE Custom Integrated Circuits Conference, May, 1989, pp. 5.3.1-5.3.5.

[7] M.I. Masud and S.J.E. Wilton, "A New Switch Block for Segmented FPGAs," International Workshop on Field Programmable Logic and Applications, August 1999.

[8] S.J.E. Wilton, Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 56-68, Jan, 2000.

[9] S.J.E. Wilton, Architecture and Algorithms for Field-Programmable Gate Arrays with Embedded Memory, PhD Thesis, University of Toronto, 1997.

[10] S.J.E. Wilton, "Implementing Logic in FPGA Embedded Memory Arrays: Architectural Implications," IEEE Custom Integrated Circuits Conference, May 1998.

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

[12] Xilinx, Inc. XC4000E and XC4000X Field-Programmable Gate Arrays Datasheet, v. 1.6. 1999.

[13] Y. Sankar and J. Rose, "Trading Quality for Compile Time: Ultra-Fast Placement for FPGAs," ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Feb. 2000, pp. 157-166.

[14] Y. W. Chang , D. Wong , and C. Wong, Universal Switch Modules for FPGA Design ACM Transactions on Design Automation of Electronic Systems, vol. 1, pp. 80-101, Jan, 1996.

[15] J. Cong and S. Xu, "Technology Mapping for FPGAs with Embedded Memory Blocks," *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Feb 1998, pp. 179-187.

[16] R.J Francis, J. Rose, Z. Vranesic, "Technology Mapping Lookup Table-Based FPGAs for Performance" Proc. 1991 IEEE International Conference on Computer-Aided Design (ICCAD), November 1991, pp. 568-571.