

Automated Transistor Sizing for FPGA Architecture Exploration

Ian Kuon and Jonathan Rose

The Edward S. Rogers Sr. Department of Electrical and Computer Engineering,
University of Toronto, Toronto, Ontario, Canada
{ikuon,jayar}@eecg.utoronto.ca

ABSTRACT

The creation of an FPGA requires extensive transistor-level design. This is necessary for both the final design, and during architecture exploration, when many different logic and routing architectures are considered. For such explorations, it is not feasible to spend significant amounts of time on transistor-level design. This paper presents an automated transistor sizing tool for FPGA architecture exploration that uses a two-phased approach - a coarse rapid phase with simple modeling followed by refinement with much more accurate models. The output of the system is a design optimized towards a specific area-delay criterion. We compare the quality of our results to prior manual and partially automated approaches. Also, our tool has been used to produce hundreds of candidate architectures which we are releasing to support future high quality explorations.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids

General Terms: Algorithms, Design, Measurement

Keywords: FPGA, Transistor Sizing, Optimization

1. INTRODUCTION

Field-programmable gate arrays (FPGAs) are increasingly used to avoid the significant implementation challenges that accompany the latest process technologies. However, the design of the FPGAs themselves is a huge undertaking in which significant time is spent finalizing the circuit topology, optimizing the transistor sizes and creating the physical layout. While this level of effort is acceptable for creating a product, architects developing a new FPGA cannot expend such effort for every new idea.

However, the experimental procedure used to assess FPGA architectures requires accurate transistor-level design. The assessment process, shown in Figure 1, takes benchmark circuits and synthesizes, places and routes them on to each FPGA design. This process is necessary because the programmable nature of FPGAs means delay, area and power can vary between applications implemented on the same FPGA. Two attributes define the FPGA design: the logical architecture and its transistor-level design. The *logical architecture* defines the logical behavior of the FPGA including the number and size of the lookup tables (LUTs) grouped together and the structure of the routing segments that connect those clusters of LUTs. The transistor-level design defines the area, delay and power of the FPGA's components. With these inputs, the *effective* area, speed and power of the FPGA can

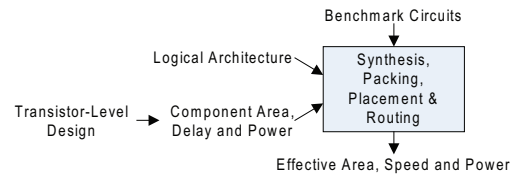


Figure 1: FPGA Architecture Assessment

be determined from the experimental procedure. One of the significant challenges in this process is the transistor-level design as this has historically required months of manual effort [1, 3, 7].

Due to the massive manual effort required, thorough exploration of the wide range of design possibilities was not possible. At best, manual transistor sizing was performed for a few sample logical architectures and the results were assumed to apply to a wide range of different architectures. As well, the need for FPGAs that make different area and delay trade-offs at the transistor-level was generally ignored. These trade-offs have become important as the market for FPGAs has expanded beyond that which can be effectively served by a single design. This has necessitated the development of FPGAs that are more heavily tailored to a market's area or speed requirements and logical architecture changes alone are not able to meet those needs.

To enable more accurate and thorough architecture exploration, we have developed a transistor sizing tool for the routing and LUT-based logic of an FPGA. As inputs, a user provides the FPGA's logical architecture, a few circuit structure parameters (such as the multiplexer topologies to use) and a setting indicating the desired balance between area and performance. Transistor-level optimization is then performed. While transistor sizing for custom circuits is a well-studied problem [4, 5, 11], the programmable nature of FPGAs adds unique challenges and opportunities which we will describe. The main goal of our optimizer is to enable the architectural exploration process shown in Figure 1 but our tool could also be useful to new entrants to the market trying to quickly create new FPGA or FPGA-like devices.

The remainder of this paper is organized as follows: The optimization problem and its inputs are detailed in Section 2. Then, in Section 3 the optimization process is described and in Section 4 the quality of its results is examined. Finally, Section 5 highlights the architectural explorations this tool enables and Section 6 concludes.

2. PROBLEM DEFINITION

Before describing our transistor sizing tool for FPGAs, we examine the unique features of this problem and describe how we handle these issues. The broad architecture explorations we aim to enable require a great deal of flexibility in the optimizer and, in this section, we will also review the inputs necessary to provide this flexibility. To keep the scope of this work tractable, we will focus exclusively on area and delay optimization.

2.1 Uniqueness of FPGA Transistor Sizing

The transistor sizing optimization problem for FPGAs is on the surface similar to the problem faced by any custom circuit designer. It involves minimizing some objective function such as

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2008, June 8–13, 2008, Anaheim, California, USA

Copyright 2008 ACM 978-1-60558-115-6/08/0006...5.00

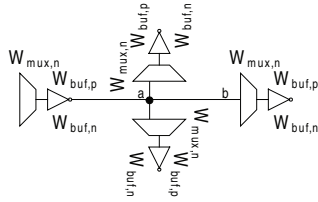


Figure 2: Repeated Equivalent Parameters

area, delay or a product of area and delay, subject to a number of area, delay or size constraints. While this is a standard optimization problem, the unique features of programmable circuit design change both the problem and its complexity.

The most unique feature of FPGA design optimization is that there is no well-defined critical path. Different circuits will have different critical paths and, therefore, there is no standard or useful definition for the “delay” of an FPGA. This is the reason the experimental process shown in Figure 1 is necessary and, in Section 2.4, we describe how we handle this issue.

The other feature of FPGA design is the significant number of logically equivalent components whose size and electrical properties must be kept the same. An example of this, illustrated in Figure 2, is a routing track that contains a number of logically equivalent switches. This requirement greatly reduces the number of parameters that require optimization which enables us to consider approaches that would not normally be possible when optimizing a design containing hundreds of millions of transistors.

2.2 Inputs

The area and delay estimation process requires three inputs: the logical architecture of the FPGA, some electrical architecture parameters and an optimization objective.

2.2.1 Logical Architecture Parameters

As described earlier, the logical architecture defines the behavior of the FPGA that is visible to a user including the size of the lookup tables, the number of LUTs grouped together to form a cluster, the connections available into each cluster, and the structure, the lengths and the flexibility of the routing segments used to connect these clusters. Modern FPGAs often also contain hard blocks such as multipliers and memories which we do not consider in this work. Ignoring these hard blocks is acceptable as the LUT-based logic still comprises a large portion of the FPGA’s area and it continues to significantly impact the overall performance of an FPGA. Also, the design of the routing within the LUT-based blocks can be reused with any hard blocks.

2.2.2 Electrical Architecture Parameters

While the logical architecture parameters define the functionality of the circuitry, such as the size of any required multiplexers, there is a multitude of different possible electrical implementations for each logical architecture. For large multiplexers, the approaches used range from fully encoded multiplexers [3], to two-level partially-decoded structures [8, 6]. Similarly, the placement of buffers has also varied between placement at the input to multiplexers (in addition to the output) [1] or simply at the output [8, 6]. These implementation choices are left as inputs to allow architects to explore their impact. From these few electrical architecture inputs, a detailed circuit-level implementation is created assuming standard circuit design practices.

2.2.3 Optimization Objective

Finally, in addition to the logical and electrical parameters that must be specified, it is also necessary to provide an objective function that will determine the trade-offs to make between performance and area. While past architectural studies focused exclusively on minimizing the area-delay product of the circuit [3, 7], we believe a wider range of possibilities must be considered. Different trade-offs are made by altering the function that the optimizer aims to minimize. We focus on functions of the form, $\text{Area}^b \text{Delay}^c$. Such functions have been commonly used

in the past for optimization [5, 11]. By varying b or c different trade-offs between area and performance can be achieved. The following sections describe the methods used to measure the area and delay for this objective function.

2.3 Area Modeling

Simple area models such as the sum of all the transistor widths have been used in the past for circuit optimization; however, since we aim to explore a wide range of architectures and area and delay trade-offs with our optimizer, more accuracy is needed because unrealistic area modeling could cause the scope of these trade-offs to be measured incorrectly.

The most accurate area estimate would be to create complete layouts of each design but that is clearly not feasible for the large number of designs that will be created. Instead, we developed a modified version of the minimum-width transistor area model typically used in FPGA architectural studies [3, 1]. Multiple improvements were made to the model including: adjusting the multiplicative constants in the minimum-width transistor area model to match our process rules, separate modeling of the area for the densely laid out configuration SRAM cells and calibration of the estimated areas against manually laid out portions of the design.

2.4 Optimization Delay Model

As described previously, one challenge in the optimization of programmable circuitry is that the eventual critical path is not known at design time and, in fact, the path will vary between application circuits. To handle this issue, we create a *representative* path which will be used during optimization. Unlike with traditional circuit optimization in which all the paths through a combinational block must be optimized to achieve a desired delay, we instead create a single path containing all the unique components within the FPGA. This path is the shortest register to register path within an FPGA that uses every unique resource. The delay for optimization purposes is taken as a weighted sum of the component delays of this path with the weights set based on the frequency with which each component occurred in the critical paths of a set of benchmark circuits.

3. AUTOMATED TRANSISTOR SIZING

With the inputs and the area and delay models defined, the optimization process can now be described. In the first phase, delay estimation using simple linear device models is used to obtain a near optimal sizing given those models. Linear device models have long been known to be inaccurate [9] and, therefore, to account for the true behavior of transistors, a second phase refines the sizes using a simple algorithm combined with simulation using accurate device models. The complete process is described below.

3.1 Phase 1 – Linear-based Models

In this first phase of optimization, transistors are treated as simple linear resistors and capacitors. The delay of the circuit is computed using the standard Penfield-Rubinstein RC model [10]. The optimization of this delay given the linear components is a well-studied problem [11, 5] and it has been recognized that, for the objective functions we use in this work, the optimization problem can be mapped to a convex one [5] which implies that any local minimum obtained is in fact the global minimum. Since getting trapped in sub-optimal local minima is not a concern, a relatively simple algorithm can be used and, for this work, a modified version of the TILOS algorithm was used [5]. We describe the algorithm as changing *parameter* values not specific transistor sizes to emphasize that transistors are not sized independently since preserving logical equivalency requires groups of transistors to have matching sizes.

The algorithm starts with all the transistors minimum sized. For each parameter, the change in the objective function per change in area is measured. After testing all the parameters, the parameter with the greatest reduction in the objective function (which is to be minimized) per area is increased in size, as was done in TILOS. In addition to this, the parameter with the greatest increase in the objective function per area is decreased in

size, which will reduce the objective function. This latter feature addresses one potential limitation of the TILOS algorithm but it is applied selectively to avoid oscillatory changes (i.e. alternating between increasing and decreasing a parameter value). The process repeats until no parameters require adjustment. As with the original TILOS implementation, there is the possibility that the algorithm will terminate with a non-optimal solution. We do not believe this is a concern because, regardless of the solution obtained, refinement of the designs is necessary to account for the non-linear behavior of the devices. This refinement is done in the next phase of the algorithm.

3.2 Phase 2 – Sizing with Accurate Models

The sizes determined using the linear models in the previous phase are adjusted using an algorithm that relies on accurate device models and full simulation with Synopsys HSPICE. This improves the accuracy of the delay measurements but means that the problem is no longer convex. The use of a single representative path as described in Section 2.4 means that interrelated circuit paths do not have to be considered which simplifies the problem considerably and allows complex approaches such as those in [4] to be avoided. Furthermore, even though FPGAs contain millions of transistors, the representative path contains only thousands of transistors which makes optimization using HSPICE feasible. We build on top of HSPICE using a simple iterative greedy algorithm. Every parameter is considered in turn and is simulated across a range of values. The value that gives the best result is selected and the process then repeats for the next parameter. The algorithm terminates if no parameter values required adjustment or repeats over all the parameters if any values were changed.

Such a greedy algorithm is not well-suited to adjusting the sizes of transistors individually because closely connected transistors must typically be sized together to realize the full benefit of size increases. We address this challenge by also operating on small groups of related transistors. For example, in a two stage buffer, one adjustment considered by the optimizer is the adjustment of all four transistors together (i.e. increasing the sizes of all the transistors by a common factor). However, we retain the freedom to adjust each transistor size individually as well to enable improvements such as those possible by skewing the pn ratios to offset the slow rise times of multiplexers with NMOS pass transistors.

This phase of the optimization dominates the run time. The overall run time depends on the parameters of the design being optimized with the LUT size being the most significant parameter. For a 4-LUT architecture with clusters of size 10, the run time on an Intel Xeon 5160 is 4.3 hours. Our focus in this work was not run time optimization and, hence, this time could be easily reduced through the use of fast SPICE simulators such as NanoSim or parallelization of this latter phase of the algorithm as it is readily parallelizable.

4. QUALITY OF RESULTS

The goal of our tool is to produce transistor sizings that can enable architectural explorations of FPGA. In this section, we compare the quality of results we obtain to past work. With our automated approach we gain the ability to search a larger design space and this can enable significant performance improvements. To provide further confidence in our results, we demonstrate that for a problem in which exhaustive sizing is possible, we obtain comparable results.

4.1 Past Interconnect Optimizers

The transistor sizing of buffers for the routing interconnect, similar to that shown in Figure 2, was considered in [6] for 180 nm CMOS. The sizes and test circuit structure used in [6] were simulated¹ and the delay results were compared to those obtained when our optimizer was used. Table 1 summarizes this comparison across the full range of interconnect lengths considered in

¹Due to different interconnect modeling methods and HSPICE versions slightly different delay results were obtained when we replicated the simulation from [6]. This difference is minor and less than 11 ps at worst.

Table 1: Interconnect Buffer Sizing Optimization

Wirelength (mm)	Delay (ps/mm)		
	From [6]	This Work	With Mux Optimization
0.5	410	409	262
1	258	257	203
2	189	193	159
3	178	179	160
4	181	176	168

Table 2: Logic Block Delays

Cluster Size	Delay (ps)		% Improvement
	[1]	This Work	
4	1079	1060	1.8%
6	1106	1095	1.0%
8	1104	1138	-3.0%
10	1101	1174	-6.2%

[6]. Clearly, we are able to achieve performance matching that obtained in [6].

In [6], the size of the transistors within the routing multiplexers was fixed at minimum size. With our tool more thorough optimization and design space exploration is possible. When we allow the sizes of the multiplexer transistors to be optimized significant delay improvements were possible as shown in column 4. For the shortest interconnect segment, an improvement of 36 % was obtained.

4.2 Past Manual Transistor Sizing

While the work in [6] focused on routing circuit design, [1] performed detailed sizing of the logic block. Delays for various segments within the logic block were optimized for a range of cluster sizes and LUT sizes using 180 nm CMOS.

We used the same technology and the same optimization objective (area-delay) with our optimization tool.² The results obtained are compared with the results from [1] in Table 2. The table lists the delay from the input of the logic block to the output across a range of cluster sizes. The results obtained using our optimizer closely match the previously reported delays but the current results were obtained with a fraction of the effort. While our work produces slightly slower designs in some cases, these differences may be due to the vastly different area models used in these works and that our work adjusted the routing circuit sizes as well. This latter factor is significant because with different cluster sizes the ratio of logic area to routing area will vary. Together, both these factors can significantly affect the results since the overall area-delay product was optimized. We believe that, since we considered both the logic and routing, our sizing is in fact more realistic.

4.3 Comparison to Exhaustive Search

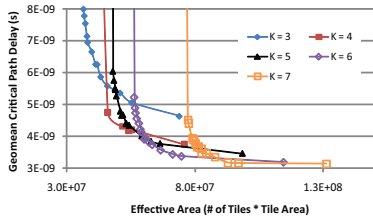
We also compared the results obtained from our optimizer to the best results obtained from an exhaustive search with both optimizing for delay. It is only possible to optimize some of the sizes of a loaded routing segment, similar to that shown in Figure 2, for this comparison because the number of test cases quickly grows unreasonably large for the exhaustive search.

We varied the specific transistors whose size could be adjusted. The delay for the routing segment from our sizer and the exhaustive search were compared for four different combinations of adjustable transistors. The results were consistently within 1.2 % of each other. Table 3 lists all the comparison results. For these cases, our optimization tool was 30.6X times faster than the exhaustive search.

²There are differences in circuit structure between these works as [1] assumed that the selection signals to the multiplexers were driven at voltages above the nominal supply voltage and [1] used a fully-encoded multiplexer structure. These two differences likely partially cancel each other out.

Table 3: Exhaustive Search Comparison

Test Case	Segment Delay (ps)		%
	Exhaustive	This Work	
A	221.4	221.9	-0.2%
B	226.9	230.1	-1.4%
C	224.9	224.9	0.0%
D	226.4	226.4	0.0%

**Figure 3: Trade-offs for Varied LUT Sizes**

5. ARCHITECTURE EXPLORATION

In this section, we show how our tool can be applied to various aspects of FPGA architecture development.

5.1 Enabling Architecture Exploration

A significant contribution of this tool is that it makes it easy to perform the transistor sizing for a large number of different designs which, as shown in Figure 1, is necessary for accurate architectural studies. To enable future FPGA architecture research, we are publishing, at <http://www.eecg.utoronto.ca/vpr/architectures/>, the areas and delays for over a hundred different combinations of logical architecture, optimization objective and process technology. We believe this is the first time that accurate and complete component areas and delays have been made available publicly.

5.2 Architecture Exploration Examples

We now perform some architectural and circuit structure investigations that demonstrate the significant benefits of using our automated transistor sizer to obtain the component area and delay measurements required to assess an FPGA design's quality.

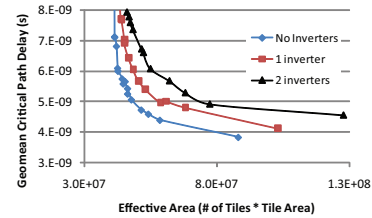
5.2.1 LUT Size

One frequently investigated issue is the selection of LUT size. We examine which LUT sizes in 90 nm CMOS are best when different area-delay trade-offs are sought. For each LUT size, we use our transistor sizer with a range of different optimization objectives. The designs are evaluated using the experimental procedure shown in Figure 1. For that process, an updated version of VPR [2] (available at <http://www.eecg.utoronto.ca/vpr>) is used with the standard MCNC benchmark circuits. The overall delay is the geometric mean critical path delay, as measured with HSPICE, of the benchmark circuits.

We plot the delay versus area curves obtained using this procedure in Figure 3 for LUTs ranging in size (K) from 3–7. The figure indicates that, as expected [1], for the best performance large 7-LUTs are best and for the smallest design, smaller LUT sizes are best. However, previously the trade-offs between area and delay could not be fully appreciated. We now see that a 6-LUT architecture can achieve nearly the same performance as a 7-LUT design when the same amount of area is used.

5.2.2 Multiplexer Buffer Placement

As discussed in Section 2.2.2, one circuit question that has not been fully resolved is whether to use buffers prior to multiplexers in the routing structures. For example, in Figure 2, a buffer could be placed at positions a and b to isolate the routing track from the multiplexers. The potential advantage of the pre-multiplexer buffer is that it reduces the load on the routing tracks because

**Figure 4: Trade-offs with Varied Pre-Multiplexer Inverter Usage**

only a single buffer can be used to drive the multiple multiplexers that connect to the track in a given region. (For example, at position a in Figure 2.) Both logical architecture, which affects the number of multiplexers connecting to a track, and electrical design, which determines the size (and hence load) of the transistors in the multiplexers relative to the size of the transistors in the buffer, may impact the decision to use the pre-multiplexer buffers.

We investigated this issue for one architecture consisting of 4-LUT clusters of size 10 with length 4 routing tracks in 90 nm CMOS. Using the procedure described above, the effective area and delay was determined using the full experimental flow without a buffer, with a single inverter, and with a two-inverter buffer. Figure 4 plots the area delay curves for each of these cases. It is interesting to consider the full area-delay space because it might be possible that for different transistor sizings the buffers might become useful. However, in Figure 4, we see that across the range of the design space the fastest delay for any given area is obtained without using the buffers. For this architecture no pre-multiplexer buffering is appropriate. Similar results were obtained for other cluster sizes as well.

6. CONCLUSION

In this paper, we have presented a tool that performs transistor sizing for FPGAs. This tool delivers results that are comparable to past approaches but requires significantly less effort than was previously required which will enable architects to more easily explore different FPGA architectures and area and delay trade-offs.

7. REFERENCES

- [1] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *TVLSI*, 12(3):288–298, March 2004.
- [2] V. Betz and J. Rose. VPR: A new packing, placement and routing tool for FPGA research. In *FPL*, 1997.
- [3] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.
- [4] A. R. Conn et al. Gradient-based optimization of custom circuits using a static-timing formulation. In *DAC*, pages 452–459, 1999.
- [5] J. P. Fishburn and A. Dunlop. TILOS: A posynomial programming approach to transistor sizing. In *ICCAD*, pages 326–328, Nov. 1985.
- [6] E. Lee et al. Interconnect driver design for long wires in FPGAs. In *FPT*, pages 89–96, Dec. 2006.
- [7] G. Lemieux et al. Directional and single-driver wires in FPGA interconnect. In *FPT*, pages 41–48, Dec. 2004.
- [8] D. Lewis et al. The Stratix II logic and routing architecture. In *FPGA*, pages 14–20, 2005.
- [9] J. K. Ousterhout. Switch-level delay models for digital MOS VLSI. In *DAC*, pages 542–548, 1984.
- [10] J. Rubinstein et al. Signal delay in RC tree networks. *TCAD*, 2(3):202–211, July 1983.
- [11] S. S. Sapatnekar et al. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *TCAD*, 12(11):1621–1634, 1993.