Characterization and Parameterization of a Pipeline Reconfigurable FPGA

Matthew Moe, Herman Schmit, Seth Copen Goldstein
Carnegie Mellon University
Pittsburgh, PA 15213
{moe, herman, seth}@ece.cmu.edu

Abstract

This extended abstract defines a class of architectures for pipeline reconfigurable FPGAs by parameterizing a generic model. This class of architectures is sufficiently general to allow exploration of the most important design trade-offs. The parameters include the word size and LUT size, the number of global busses and registers associated with each logic block, and the horizontal interconnect within each stripe. We have developed an area model for the architecture that allows us to quickly estimate the area of an instance of the architectural class as a function of the parameter values. We compare the estimates generated by this model to one instance of the architecture that we have designed and fabricated.

1.0 Introduction

FPGA architectures are created by balancing the features desired by a compiler and the constraints of silicon design. It is therefore desirable to identify the design decisions that need to be made, parameterize the architecture so that a complete architecture is defined by a choice of those parameters, and evaluate how choices for the different parameters affect both silicon area and the ability of the compiler to generate efficient configurations quickly and robustly. This paper defines and parameterizes a large class of pipeline-reconfigurable FPGA architectures. We also briefly describe two area models developed for this class of architectures. We have created one instance of this architecture, and compare the results of our area model against the real silicon area required by our actual design.

2.0 Parameterized Architecture

Two stripes of an extremely general pipeline reconfigurable architecture are shown in Figure 1. We have modified this generic model to include some parameters that help us to determine desirable attributes for our FPGA. A complete picture of the parameters of this model is shown in Figure 2

As in [2], it was decided that the most efficient means of implementing combinational logic was through the use of LUT's. B bits worth of LUT's grouped together form a logic block. This decreased by a factor of B the number of configuration bits necessary to configure these LUT's. N of these logic blocks were grouped together to form a stripe. This makes each stripe N*B bits wide.

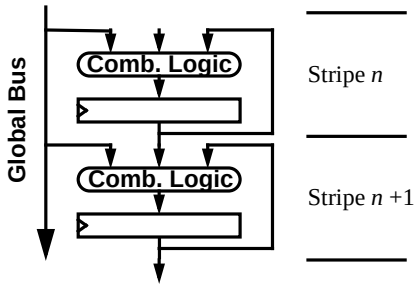

Figure 1. Generalized Pipeline Reconfigurable FPGA

There are three types of the D inputs to each logic block: D_g is the number of logic block inputs that can connect to the horizontal interconnect network or the G global busses. These global busses are B bits wide and are the only way to get data in and out of the reconfigurable fabric[1]. $D_p$is the number of logic block inputs that can connect to the horizontal interconnect network or the P pass registers of the previous stripe. Pass registers are a means of passing multiple registered values from one stripe to the next. This allows P*N*B bits of data to be passed from one stripe to the next. $D_c$is the number of control inputs to the LUT. The logic block therefore contains B LUTs, each of which has $D = D_p + D_g + D_c$ inputs.

3.0 Area Model

We have developed two models that allow us to generate silicon area estimates of any instance of the architecture. The first model uses wire density to determine the lower bound of area for any set of parameters, and a silicon technology with any number of metal layers and any metal routing pitch. The second model makes some specific design decisions for a three metal layer technology, and includes estimates of non-interconnect areas.

In the first model we need to determine the number of wires that travel horizontally and vertically through the logic block. In total, there are 2B(P+G+N) + 4C + 22 wires that need to cross the cell boundary. C is the number of configuration bits needed for each logic block and is based on the other parameters. Our methodology for using the wire-bound model is to assume that the first layer of metal in any technology will be completely utilized by power and ground distribution as well as contact area for devices and intra-cell routing. The remaining layers of metal can be dedicated to signal routing in either a horizontal or vertical

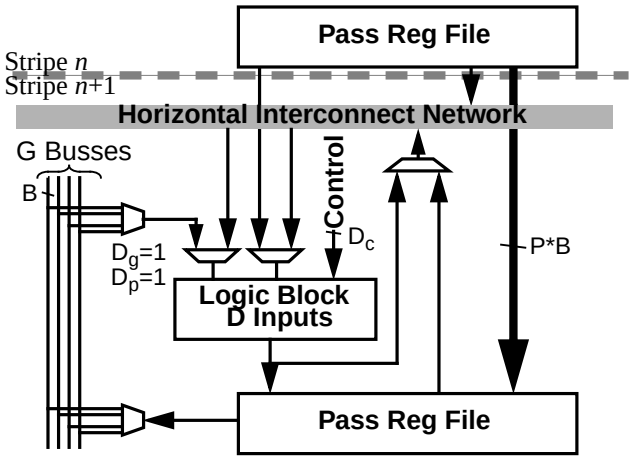

Figure 2. Parameterized Cell Architecture

direction. By computing the number of wires crossing the cell boundary, and the maximum wires that can cross a unit of length in the chosen technology, we can determine the perimeter of the cell. If we assume that the cell is square, we can derive the area by computing the square of one quarter of the perimeter.

We have designed and fabricated an instance of the architecture, and we use the area of this design to justify and refine our model. The parameter settings for the architecture we fabricated are as follows:

$$(B,G,N,P,D_c,D_p,D_g) = (4,4,4,1,1,1,1) \quad \text{(EQ 1)}$$

The real dimensions of this design, as well as the wire-bound estimate are shown in Table 1. If we make the technology-specific decisions to use metal 3 to route vertical signals (global busses, pass registers, and configuration bits) and use metal 2 for horizontal signals, we can determine the height and width of the cell using the wire-bound model. This decision increases the error of the model, but it shows that the source of error is primarily in estimating the height of the cell.

The width of the actual cell is determined not by the wiring, but by the width of SRAM cells. This is not true in general, because if there are many more global bus bits than configuration bits, the width will be wire-bound. Our augmented width model takes the maximum of the SRAM pitch and the wiring width:

$$Width = \max(Width_{SRAM} \cdot C, Pitch_{M3}(BG + BP + 2C)) \quad \text{(EQ 2)}$$

The primary cause of error in this model was the fact that the LUTs, the configuration SRAMs, the registers and the input multiplexors required metal 2 routing, therefore making those wiring channels useless for horizontal routing. In order to increase the accuracy of the height model, we can include the area of these elements in the height estimate using the term A_{Fixed}. We've developed models for these components based on the parameters, which we will not discuss here. The augmented cell height model is:

$$Height = Pitch_{M2} \cdot (BN + 11) + (A_{Fixed})/(Width) \quad \text{(EQ 3)}$$

The results of the augmented width and height models are shown in Table 1.

We have accounted for most of the remaining disparity between the augmented model and the real design. A height of 114λ was required by the interconnect network and not accounted for in the model. A future refinement of the model will include this height. Another 42λ of height went into other constant and control parts, these are implementation dependent and are left to the layout designer's discretion. Adding these two pieces to our model leaves 13% of the height and 23% of the area unaccounted for. This area can be accounted for by many small pieces too numerous to classify and by suboptimal design.

4.0 Related work

Early work in evaluating FPGA architectures [4] explored using architectural parameters like LUT size and interconnect network to optimize FPGA architecture for gate replacement. By developing CAD tools that could retarget to models of variably sized LUTs and interconnect, they determined optimal parameters for FPGA architectures. DeHon [3] broke down FPGA area into area for interconnect, configuration storage, data storage, and control and configuration storage.

5.0 Conclusions and Future Work

Using the augmented model, the area of a tile is 65% that of the actual area of the tile. The width of the model is 88% of the width of the actual tile. The height of the cell is 75% of the height of the actual tile. The error in the width can be attributed to the fact that the SRAMs were used as a guide for the width of the cell in the actual design, but not as an absolute maximum. The height difference in the cell can be attributed to various random wiring needed for intra-cell connections not added into the model. In addition, our layout is sub-optimal. The model has helped us discover places where the layout could be improved.

We plan to continue increasing the descriptive power of our parameterized architecture and improve the accuracy of our area model. In addition, we will create more instances of the architecture to determine the accuracy of the model across the parameter space. The major hurdle at this point is the parametrization of interconnect structures and the determination of which ones are worthy of investigation. We plan to try many different interconnect structures, and we hope to leverage previous research on well-characterized and parameterized network architectures.

Table 1: Area Comparison

Model	Width (λ)	Height (λ)	Area (λ^2)
Wire bound (square)	333	333	111,000
Wire Bound (3-layer)	560	162	90,700
Augmented	725	955	692,000
Actual	810	1280	1,040,000

6.0 Acknowledgments

The authors wish to thank DARPA (under contract DABT63-96-C-0083) for funding this research.

7.0 References

[1] S. Cadambi, J. Weener, H. Schmit, S. C. Goldstein, and D. E. Thomas, "Managing Pipeline Reconfigurable FPGAs," In *Sixth International ACM/SIGDA Symposium on Field-Programmable Gate Arrays*. ACM, February 1997.

[2] D. Cherepacha and D. Lewis. "A Datapath Oriented Architecture for FPGAs." In *Second International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*. ACM, February 1994.

[3] A. DeHon, *Reconfigurable Architectures for General-Purpose Computing.* Ph.D. Thesis, AI Technical Report 1586, MIT Artificial Intelligence Laboratory, September 1996.

[4] S. Singh, J. Rose, P. Chow, D. Lewis, "The Effect of Logic Block Architecture on FPGA Performance," *IEEE JSSC*, Vol. 27 No. 3, pp. 281-287, March 1992.

[5] Schmit, H. "Incremental Reconfiguration for Pipelined Applications," *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 47-55, 1997.