FPGA Performance versus Cell Granularity

Jack L. Kouloheris* Abbas El Gamal†

Dept. of Electrical Engineering
Stanford University
Stanford, Ca. 94305-4055
(415) 723-3473
FAX: (415) 723-8473

Abstract

An experimental approach is used to investigate the relationship between the performance of an FPGA and its basic cell granularity. Over a large set of design examples it is found that a 4 or 5 input cell achieves minimum average critical path delay for a wide range of programmable switch time constant, τ_s. As expected, the "optimal" cell granularity is found to gradually increase as τ_s increases.

Introduction

A crucial choice in the design of FPGAs [3, 4, 5, 6] is the type and granularity of the basic cell. Employing a large basic cell would require less routing resource and result in less overall routing delay than for a smaller cell. On the other hand, a large basic cell would be slower and less utilized than a smaller one. It is, therefore, plausible that an "optimal" granularity may indeed exist. Rose *et al.* [1] used an experimental approach to examine the effect of cell granularity on area. In this paper we use a similar approach with a slightly different model to show the effect of cell granularity on performance.

FPGA Model

The FPGA model we assume in this paper is similar to [6], comprising rows of identical K-input one-output basic cells interspersed with segmented routing channels. In order to simplify the logic mapping problem and to provide an easily parameterizable cell, we assume as in [1] that each basic cell can implement an arbitrary boolean function of K variables (e.g. a K-input RAM or ROM). The basic cell is also assumed to have a D flip-flop optionally in series with the cell output. The cell inputs are connected to vertical wire segments that extend over the routing channel (see Figure 1.) A switch (such as an antifuse [6] or RAM-controlled pass transistor [3]) is located at each intersection of the input segments with the horizontal wire segments. The cell outputs are handled in a similar fashion, but extend over multiple routing channels. We additionally assume single segment routing [7] between cell outputs and cell inputs. This implies that each cell output passes through exactly two switches to reach any input that is connected to it. We define the switch time constant, $\tau_s = R_{sw}C_{sw}$, where R_{sw} is the switch"on" resistance, and C_{sw} is the switch loading capacitance (including parasitics) seen by a wire segment. Interconnect delays are computed using the method of Penfield *et al.* [8] (see Figure 2). Cell delay was modeled as shown in the formula in Figure 2, and was determined via Spice simulations of an implementation in a 0.8 micron CMOS technology.

*Supported by an IBM Resident Study Fellowship and DARPA contract J-FBI-89-101

†Research supported by DARPA contract J-FBI-89-101

Experimental Procedure

We use a set of benchmark designs representing a mix of design styles and sources, including the MCNC [9] and ISCAS benchmark sets as well as real FPGA designs (see Table 1). Our experimental procedure is as follows:

For each design in benchmark set
  For K = 2 to 10
    1. Remap logic design onto K-input cells (Chortle[2])
    2. Run placement (GDT AutoCells)
    3. Run global routing to determine horizontal and vertical segment lengths and channel densities
    4. Calculate net delays
    5. Back annotate netlist with delays and find critical path delays (Ceres[10])
  End
End

For each design and for each K we find the number of cells, net delays, critical path delay, as well as horizontal

IEEE 1991 CUSTOM INTEGRATED CIRCUITS CONFERENCE CH2994-2/91/0000/0020 $1.00 © 1991 IEEE

and vertical track requirement W and V. We use the average of all the combinational path delays (input to output or latch to latch) as a measure of the critical path delay. The baseline implementation for comparing the designs is the implementation for K=2. The delay for each design is normalized by dividing the values for K=3 ... 10 by the values for K=2. This allows us to combine the data from the different benchmarks, and show the relative change over the K=2 case.

Results

Our results are shown in Figures 3 through 6. The error bars in all figures indicate the 95% confidence interval of the mean over all benchmarks.

Figure 3 shows how the total number of basic cells in a design decreases as the granularity of the basic cell is increased. As K increases from 2 to 4, the number of cells required to implement a design decreases substantially, requiring less than half as many cells as for K=2. For K greater than about 5 or 6 there is little improvement in the number of cells required. We found that the average number of cells in the critical path decreased in direct proportion to the decrease in the total number of cells in the design.

The average net delay (Figure 4) is an increasing function of K as the number of switches capacitively loading each net is increasing.

Figure 5 shows how the critical path delay varies with K and with τ_s. The top curve, $\tau_s = 0$, shows the critical path delay with zero interconnect delay. This is an increasing function of K as the delay per cell is increasing faster than the number of blocks in the critical path is reduced. For non-zero values of τ_s, the critical path delay first decreases, as the savings from having fewer cells in the critical path exceeds the increase in cell and net delay, then increases as the critical path ceases to be significantly improved with increasing K.

Figure 6 plots the median best K vs τ_s. As expected, technologies with small τ_s favor small granularity while technologies with larger τ_s favor larger granularity (see Fig. 6). The curve flattens out for large τ_s as the mapping program is not able to effectively use more than 6 inputs per output for most designs. However, a few designs are able to effectively utilize large cells, resulting in a large variance in the best K for large τ_s.

An analysis of variance (ANOVA) was performed on the data for the curves in Figures 3–5, to see if the effect due to varying the granularity was significant compared to the variability between benchmarks. For all three cases, the effects were significant with $p < .0001$. The non-parametric Kruskal-Wallis test was used to test the variation of the best K vs τ_s shown in Figure 6, as the distributions were highly non-normal. In this case also , the effect was significant with $p < 0.0001$. No significant differences in behavior were found for large designs (> 1000 cells) vs. small designs.

Limitations of Study

The experiments in this study have been limited to one routing scheme, one basic cell type, and one mapping program in order to isolate the effect we wanted to study. The density achieved by the simple single-segment routing scheme used has been previously shown [7] to differ by only a small constant factor from unrestricted channel routing, hence we do not believe it would change the basic tradeoff we observed. However other routing schemes, such as hierarchical routing, could change the results. The basic cell we used has a delay that is linearly dependent on the number of inputs; other cells with different delay dependencies could produce different results. The logic mapping and placement tools also exert an influence on the results. The tools we used optimized for area. Performance-directed CAD tools might perform differently. In all of these cases the same experimental methodology could still be used to examine the tradeoffs.

Conclusions

Our initial work indicates that, for the single-output RAM-based FPGA, 4 to 5 inputs yield the best delay for a range of programming technologies. Surprisingly, [1] showed that this size cell was optimal for area as well. Our experiments indicate that single output basic cells are increasingly underutilized as K increases above about 5 or 6 (see Figure 3). This is because for larger K the input to output ratio of the cell becomes unbalanced. In a related paper [12] we investigate other cell types (e.g. multiple-output RAMs, PLAs) to determine if area-efficient multi-output structures can be found that improve basic cell utilization.

Acknowledgements

We would like to thank Jonathan Rose and Bob Francis for supplying us with the Chortle [2] technology mapping program, Frederic Mailhot for enhancements to his Ceres [10] and Mercury[11] tools, and Mentor Graphics (Silicon Design Division) for the use of their GDT software.

References

[1] Jonathan Rose *et al.*, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 5, October 1990, pp. 1217–1225.

[2] R. J. Francis *et al.*, "Chortle: A Technology Mapping Program for Lookup Table-Based Field Programmable Gate Arrays," *Proceedings of the 27th Design Automation Conference*, June 1990, pp. 613–619.

[3] W. Carter *et al.*, "A User Programmable Reconfigurable Gate Array," *Proceedings of the 1986 Custom Integrated Circuits Conference*, 1986, pp. 233–235.

[4] H. Hsieh *et al.*, "A Second Generation User Programmable Gate Array," *Proceedings of the 1987 Custom Integrated Circuits Conference*, 1987, pp. 515-521.

[5] H. Hsieh *et al.*, "A 9000-Gate User Programmable Gate Array," *Proceedings of the 1988 Custom Integrated Circuits Conference*, 1988, pp. 15.3.1–7.

[6] Abbas El Gamal *et al.*, "An Architecture for Electrically Configurable Gate Arrays," *Proceedings of the 1988 Custom Integrated Circuits Conference*, 1988, pp. 15.4.1–4.

[7] Abbas El Gamal, Jonathan Greene, and Vwani Roychowdhury, "Segmented Channel Routing," *Proceedings of the 27th Design Automation Conference*, June 1990, pp. 567–572.

[8] Jorge Rubinstein, Paul Penfield, Jr., and Mark Horowitz, "Signal Delay in RC Tree Networks," *IEEE Transactions on Computer Aided Design*, Vol. 2, No. 3, July 1983

[9] R. Lisanke , "Logic Synthesis and Optimization Benchmarks, User Guide, Version 2.0," *Microelectronics Center of North Carolina* , 1988

[10] F. Mailhot and G. De Micheli, "Technology Mapping Using Boolean Matching and Don't Care Sets", *Proc. European Design Automation Conf.*, March 1990, pp. 212-216.

[11] G. De Micheli, D. Ku, F. Mailhot and T. Truong, "The Olympus Synthesis System", *IEEE Design & Test of Computers*, Vol. 7 No. 5, October 1990, pp. 37-53.

[12] "FPGA Cell Utilization vs. Granularity", in preparation.

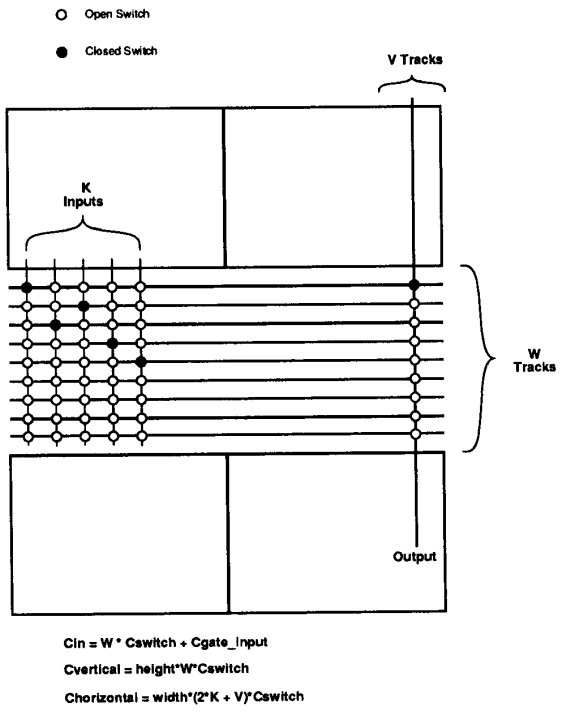

Figure 1: FPGA Model

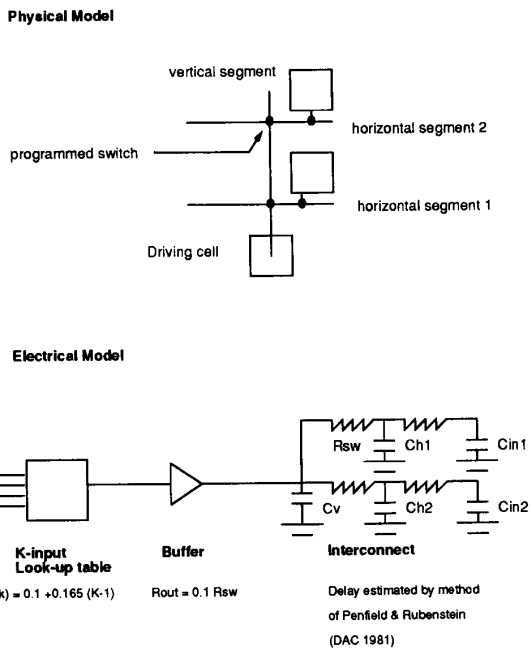

Figure 2: Delay Model

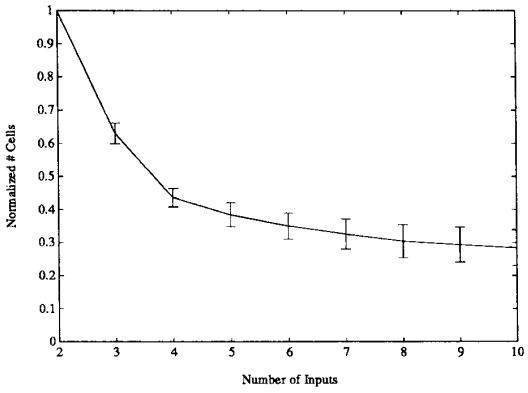

Figure 3: Number of Basic Cells vs. Number of Inputs.

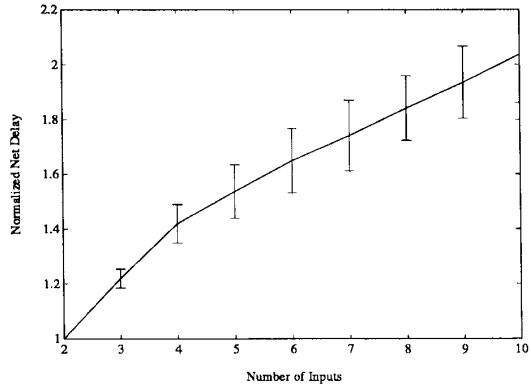

Figure 4: Average Net Delay vs. K.

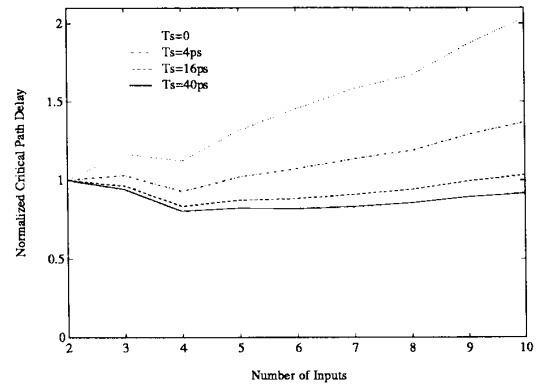

Figure 5: Critical Path Delay vs. Number of Inputs vs. τ_s. The top curve, $\tau_s = 0$, shows the critical path delay with zero interconnect delay. The curves for $\tau_s = 4$ps. and $\tau_s = 16$ps. correspond to programming technologies such as the antifuse and pass transistor, respectively.

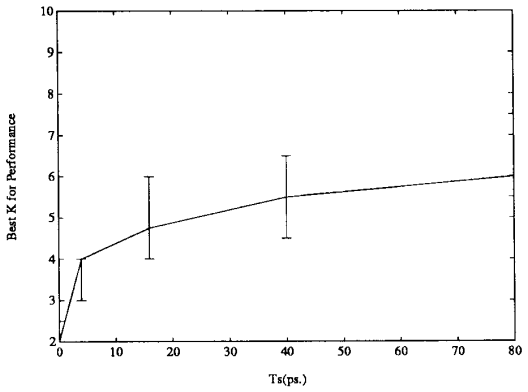

Figure 6: Best K vs. τ_s. Low τ_s architectures (close to a mask-programmed gate array) favor small granularity while for larger τ_s larger granularity yields higher performance. The curve flattens out and exhibits large variance for large τ_s as the mapping program is not able to effectively use more than 6 inputs per output for most designs. A few designs are able to effectively utilize large cells.

Benchmark name	**2-input cells**	**Best K for τ_s**			
		0	**4**	**16**	**40**
C499	392	4	4	4	4
apex6	671	2	2	4	4
decbig	3247	2	4	5	8
rot	591	2	2	6	6
des	3218	2	4	4	7
seq	3338	3	5	5	10
wmm	586	2	4	4	8
alu	384	2	4	8	8
bmk3	82	2	4	8	8
hsim	440	2	4	4	4
C880	348	5	5	5	7
C1908	429	2	4	4	4
C6288	2400	2	2	4	4
C7552	3283	2	2	4	4
afmt	551	2	4	5	5
bam	612	2	2	8	8
graphic	8940	2	7	7	7
k2	2596	2	4	5	9
sampler	605	2	4	4	6
soar	510	2	3	4	4
Mode		2	4	4	4
Mean		2.3	3.7	5.1	5.7
Total	33,223				

Table 1: Best K vs. τ_s for each benchmark