A Comparing Study of Technology Mapping for FPGA*

Hans-Georg Martin, Wolfgang Rosenstiel
University of Tübingen - Department of Computer Engineering
Sand 13, D-72076 Tübingen, Germany
email: <martinh,rosen>@informatik.uni-tuebingen.de

Abstract

This paper investigates some design flows to obtain final designs on Xilinx XC4000 FPGAs. The examples generated by high level synthesis were mapped including placement and routing. This reveals that the common criteria of area optimal or delay-optimal circuits should be enlarged by routability and computing time.

1. Introduction

Since FPGA circuits are increasingly used in many fields, a lot of research was done to obtain a good FPGA design. While placement and routing is strongly connected with the detailed architecture inside of the chip and mostly managed by the commercial FPGA software, the optimization and mapping can be more influenced by the user. Our aim is to compare some design flows to see what is the real effect of logic optimization and LUT-based technology mapping with respect to the final design.

The usual design goals of reducing area and/or circuit delay are supported by a large number of algorithms, but our examples show, that there are two other questions of increasing significance: computing time and routability. We show that for some examples state-of-the-art-tools don‘t generate final designs, while a simpler mapping approach is faster or makes a design possible at all.

2. LUT-based Technology Mapping: Overview

The plenty of methods for the LUT-based technology mapping can be grouped in algorithms for area reduction (e.g. [3]), delay reduction (e.g. [2] [4] [6] [7] [8]), combined methods [5] and procedures with other aspects [9].

Mostly the methods start with a decomposition step for the big nodes using Roth-Karp-decomposition, cofactoring [3], or multiple-output decomposition [6]. Afterwards some tools perform a strictly technology mapping (node merging, [2] [5] [7]), while other include also an expensive logical restructuring ([3] [8]). Allowing a register retiming ([7] [8]) offers a dramatic increase of optimization potential. But less was researched on mapping for routability: [9] influences the pins-per-net-ratio [10], merely it can be only a first step. Section 4 underlines that routability is not only a question of good placement, but also a matter of appropriate technology mapping.

3. Four Design Alternatives and Examples

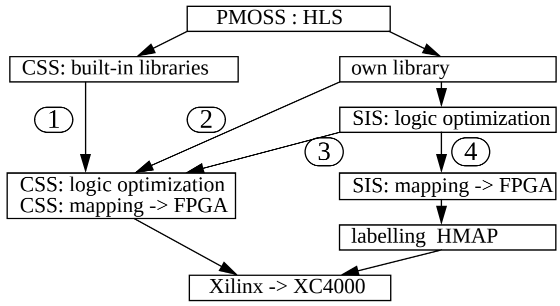

Figure 1. Four design flows from high level to FPGA

We investigate four design alternatives (fig. 1).

`Design Flow 1:` The high level synthesis (HLS) is performed with PMOSS [1]; then the adders and other functional units are replaced by (fast) elements of built-in libraries within a commercial synthesis system (CSS); thereafter the CSS optimizes and maps this network to FPGAs, and finally Xilinx software places and routes this onto a real XC4000 Xilinx FPGA.

`Design Flow 2:` The difference to design flow 1 is the usage of an own library with ripple carry adders etc.

`Design Flow 3` is flow 2 plus a logical optimization step with the Berkley tool SIS [12] (script.algebraic).

`Design Flow 4`, as a first promissing try, uses SIS also for mapping to 4-input LUTs including collapsing, cofactoring, and binate covering (script bases on [3], [12]). Then, assigning some LUTs to the H-function-generator of the configurable logic blocks (CLB) of Xilinx XC4000 series reduces both the number of CLBs and the delay.

4. Results for the Design Methods

In table 1 and 2 some results for the four design flows are presented, regarding three C-programs (ascii-to-integer, elliptical wave filter, and differential equation), which were synthesized with PMOSS for several word lengths (4-32, column 1). The other columns of table 1 depict the FPGA type, the number of CLBs, and the final circuit delay for each design flow. The computational effort for these experi-

* This work was supported by the Deutsche Forschungsgemeinschaft.

ments is stated in table 2 (sparc 10, 64 MB main memory, pure CPU time only).

Table 1: Results for 4 design flows [a]: #CLB + delay

expl.	#i/o	flow 1 type	flow 1 #clb	flow 1 ns	flow 2 [a] type	flow 2 [a] #clb	flow 2 [a] ns	flow 4 type	flow 4 #clb	flow 4 ns
atoi08	33	4002A	44	49.3	4002A	29	73.5	4002A	53	83.2
atoi16	42	4003	63	47.3	4002A	39	122.5	4003	83	110.4
atoi32	58	4004A	103	70.9	4004A	66	234.8	4004A	141	226.5
ellip04	67	4004A	104	102.5	4004A	109	90.6	4005 [b]	137	108.9
ellip08	131	4005H	178	119.0	4008 [b]	184	111.5	4008	236	131.6
ellip15	243	4025 [d]	288	162.7	4025	313	180.2	4025	397	182.0
diffeq04	35	4002A	52	68.9	4003 [b]	55	66.4	4003	76	81.9
diffeq08	67	4004A	101	115.6	4005 [b]	104	130.8	4005	159	130.4
diffeq16	131	4008	248	247.6	4010 [b]	267	268.7	4013 [b]	399	284.8
diffeq24	195	4025	458	-- [c]	4025	486	-- [c]	4025 [d]	759	420.0
diffeq31	251	4025	695	-- [c]	4025	713	-- [c]	4025	1155	-- [e]

a. design flow 3 omitted here delivers slightly bigger designs then flow 2
b. the next smaller FPGA allows no placement or no routing
c. routing is not possible
d. Xilinx internal error e. too many CLBs for XC4025

Table 2: CPU times in hours:minutes at a sparc10

expl.	flow 1 CSS	flow 1 xilinx	flow 2 CSS	flow 2 xilinx	flow 3 SIS	flow 3 CSS	flow 3 xilinx	flow 4 SIS	flow 4 xilinx
atoi08	4	7	6	4	0	6	5	2	3
atoi16	5	7	8	6	0	8	8	3	6
atoi32	7	15	14	12	1	14	16	4	10
ellip04	9	23	10	29	1	10	1:29	4	13
ellip08	16	2:11	16	37	2	18	37	9	30
ellip15	29	34:07	37	5:21	3	39	25:37	17	2:20
diffeq04	7	11	7	11	0	7	7	3	5
diffeq08	12	1:43	11	25	1	12	18	6	15
diffeq16	30	6:17	28	1:24	3	30	1:41	18	58
diffeq24	1:17	[a] 9:59	1:12	[a] 9:30	10	1:04	57:31	40	5:12
diffeq31	2:46	[a] 9:15	1:53	[a] 8:56	22	2:01	[a] 7:26	1:12	--

a. time until the message: „routing is not possible“

Table 3: Ranking of the design flows

Type of example	without a multiplier (atoi)				with a multiplier (ellip, diffeq)			
Design Flow	1	2	3	4	1	2	3	4
#CLB	+	+++	++	-	+++	++	+	-
Delay	+++	++	+	+	++	++	+	+
CPU-Time	++	+	+	+++	---	+	--	+++
Design is possible	++	++	++	++	-	-	+	+

Design flow 1 is the best, but not always possible. For circuits without a multiplier (atoi) it delivers very fast designs because only design flow 1 uses the carry-logic.

Sometimes design flow 2 produces designs with less CLBs than flow 1. Design flow 3 (omitted in table 1) yields to slightly bigger and slower results than design flow 2. So, a logic minimization added in the flow is not profitable.

Design flow 4 gives the biggest (and often the slowest) design, but the advantage of this flow is the fact that it produces results, even if the other methods cannot route the design (diffeq24). Furthermore, flow 4 is the fastest way to obtain a design (table 2). For the big circuits (ellip15, diffeq24, diffeq31) the design flows 1-3 are very time consuming, even if the routing is not possible.

A summary is given in table 3.

5. Conclusion

Commercial tools can handle small as well as large networks and they produce good results concerning area and timing, but they need very long computation time. For some big examples we find an unacceptable CPU time of 1-2 days, or the message „circuit is unroutable“ after 9 hours.

The advantage of our simple mapping approach (design flow 4) is that it can manage some big examples in a much shorter period of time (up to 13 times faster) with slightly worse results. Though there exist more sophisticated tools, design flow 4 is a real possibility for FPGA design, especially for rapid prototyping.

6. Outlook

In the future it will be worth to combine the usage of carry-logic in the FPGAs with design flow 4 or with a more advanced LUT-based technology mapping for obtaining fast and suitable designs. Also we want to investigate the reasons of unwirability of the big examples in detail and try to find better mapping criteria for routability.

References

[1] H.-J. Eikerling, W. Hardt, J. Gerlach, W. Rosenstiel. „A Methodology for Rapid Analysis and Optimization of Embedded Systems.“ In Symposium on Engineering of Computer Based Systems, Friedrichshafen, Germany, March, 1996.

[2] R.J.Francis, J.Rose, and Z.Vranesic. „Technology Mapping for Lookup Table-Based FPGAs for Performance“. ICCAD 91, pp. 568-571.

[3] Rajeev Murgai, N. Shenoy, R.K. Brayton, A. Sangiovanni-Vincentelli. „Improved Logic Synthesis Algorithms for Table Look Up Architectures“. ICCAD, 1991, pp. 564-567.

[4] J. Cong and Y. Ding. „FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs“. IEEE Trans. on Computer-Aided Design, 13:1-12, 1994.

[5] J. Cong and Y.-Y. Hwang. „Simultanious Depth and Area Minimization in LUT-Based FPGA Mapping“. Proc. ACM 3rd Int‘l Symp. on FPGA, Feb. 1995, pp. 68-74.

[6] C. Legl, B. Wurth, and E. Eckl. „A Boolean Approach to Performance-Directed Technology Mapping for LUT-Based FPGA Design“. 33rd DAC 1996, pp. 730-733.

[7] Peichen Pan and C.L.Liu. „Optimal Clock Period FPGA Technology Mapping for Sequential Circuits“. 33. DAC 1996, paper 45.1.

[8] J. Cong and C. Wu. „FPGA Synthesis with Retiming and Pipelining for Clock Period Minimization of Sequential Circuits“. DAC 1997, pp. 644-649.

[9] M.Schlag, J. Cong, and P.K. Chan. „Routability-driven technology mapping for lookup table-based FPGA‘s. IEEE Trans. on CAD, 13:13-26, 1994.

[10] P.K. Chan, M.D.F. Schlag, and J.Y. Zien. „On Routability Prediction for Field-Programmable Gate Arrays“. 30. DAC 1993, pp. 326-330.

[11] Xilinx Inc., San Jose, CA-95125. „The Programmable Logic Data Book“, 1994.

[12] E.M. Sentowich, K.J. Singh et al. „SIS: A System for Sequential Circuit Synthesis“. Documentation included in the SIS package.