Full-Custom vs. Standard-Cell Design Flow – A Quantitative Adder Comparison

Henrik Eriksson, Tomas Henriksson, and Christer Svensson
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden
{hener,tomhe,chs}@isy.liu.se

Per Larsson-Edefors
Department of Computer Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
perla@ce.chalmers.se

Abstract

Full-custom design techniques are considered superior to standard-cell design techniques when a high-performance circuit is requested. The structured routing of critical wires is considered to be the most important contributor to the performance gap. This is true for bit-sliced designs with interconnections only to neighboring bitslices, such as register files and ripple-carry adders, but not for designs with inter-bitslice interconnections spanning several bitslices, such as tree adders, barrel shifters and reduction-tree multipliers. Standard-cell design techniques scale better with the data width than full-custom bitsliced layouts for designs dominated by inter-bitslice interconnections.

1. Introduction

To speed-up the design phase for circuits with high performance requirements, standard-cell based approaches were invented. In standard-cell design basic gates or building blocks, e.g. multiplexers, full adders, flip-flops, and basic logic functions, are provided by a chip vendor. Designs can now be synthesized using these cells and appropriate CAD tools. The drawback in early process technologies was the limited number of metal layers giving the need for special routing channels for connecting the cells. This is usually not a problem in modern technologies which can have up to eight metal layers.

In full-custom design, all circuits are hand-crafted by a designer which gives the possibility to use special circuit styles and arbitrary sizing of the transistors. Since the design time is much longer than in standard-cell based design, full-custom design is mainly used in critical parts.

It is common knowledge that there is a substantial performance gap between full-custom and standard-cell based techniques, although contradictory examples exist [1]. Recently, initiatives to compare and combine the techniques have been launched [2, 3]. Dally and Chang [3] conclude that the structured floorplanning in full-custom design is the largest contributor to the performance gap for regular components such as register files.

The aim of this paper is to quantify the performance gap between the two techniques for two types of adder structures, i.e. a regular ripple-carry adder and a less regular, but faster, tree adder. The rest of this paper is organized as follows: in Section 2 the two adder structures are described, then our simulation environments are presented in Section 3. Section 4 provides the results and finally we draw our conclusions in Section 5.

2. Adders Used in the Comparison

The adders used in the comparison are one 64-bit ordinary ripple-carry adder (RCA) and one 64-bit Han-Carlson tree adder (HCA) [4]. These adder topologies were chosen since they represent two extremes of adder implementations. The RCA has almost no interconnect between the cells which comes at the cost of poor performance. The HCA, on the other hand, has as all tree adders [5] complex interconnect, i.e. inter-bitslice interconnections, and very high performance.

2.1. The Ripple-Carry Adder

For the full-custom design a state-of-the-art 16-transistor full-adder (FA) cell was chosen [6], see Fig. 1. To optimize the delay, a buffering inverter had to be inserted in every second cell in the carry path. In the case of standard cells, a Verilog netlist was written using full-adder cells from the standard-cell library.

2.2. The Han-Carlson Adder

The Han-Carlson adder is shown in Fig. 2. As can be seen in the figure, three different types of cells are needed represented by black squares, white squares, and black dots (or circles). The black squares perform the following functions:

$$G_N = A_N B_N,\ P_N = A_N + B_N,\ X_N = A_N \oplus B_N \tag{1}$$

the black dot the following:

$$G_{N+1} = G_N + P_N G_X,\ P_{N+1} = P_N P_X \tag{2}$$

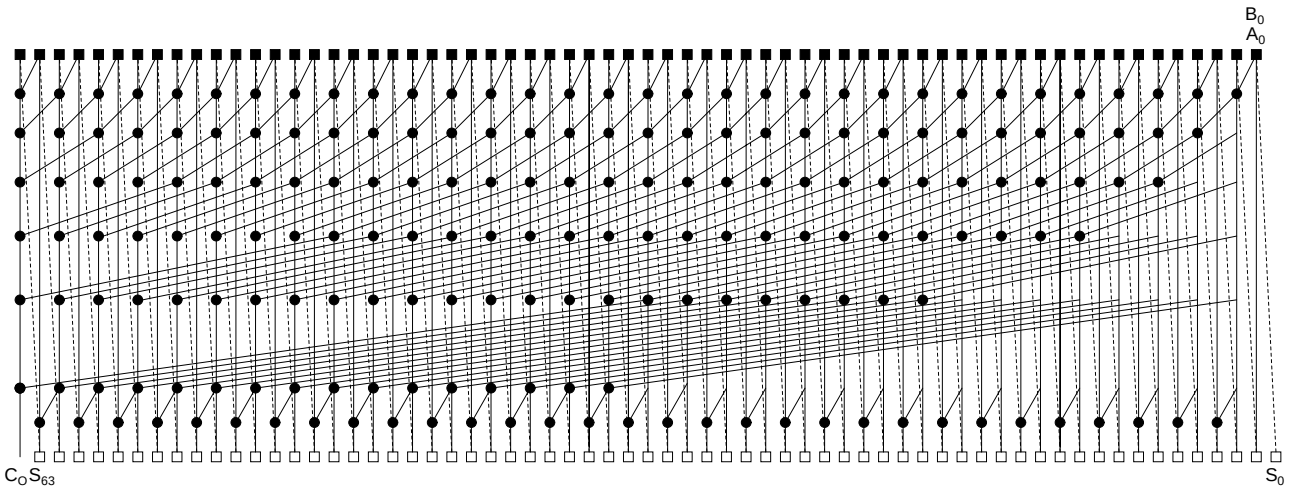

Figure 2. 64-bit Han-Carlson adder.

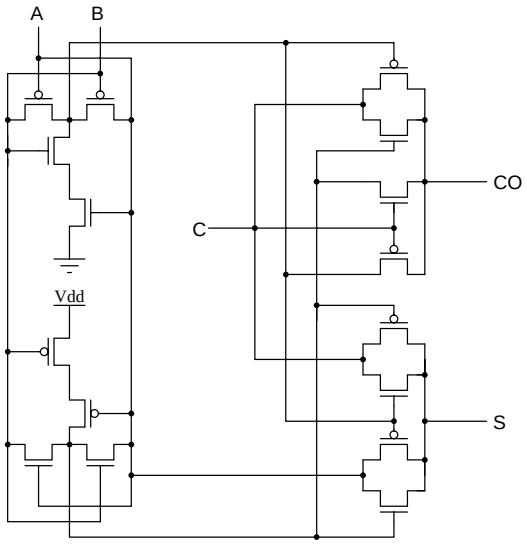

Figure 1. 16-transistor full-adder cell.

and finally the white squares:

$$S_N = X_N \oplus G_{Nmax}. \tag{3}$$

In the full-custom implementation, domino gates without footers were used throughout the design except for the XOR gates where the single-ended one from [7] was used. An example of a domino gate is shown in Fig. 3.

Basic gates like: AND, OR, EXOR, and AND-OR from the standard-cell library were used in the Verilog netlist that was used for the standard-cell design.

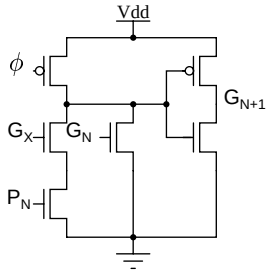

Figure 3. Domino gate without footer (part of the dot operator).

3. Simulation Environments

In this section the simulation environments and platforms for the two design flows are described. For all implementations, regardless of design flow, the AMS triple-metal 0.35-μm process technology has been used.

3.1. Standard-Cell Design Flow

Layouts of the standard-cell designs have been implemented using Silicon Ensemble for place & route (P&R) from Cadence and delays have been obtained from static-timing analysis (STA) with backannotated parasitics. For all designs, in-place optimization (IPO) has been used in order to resize the drive strength and/or introduce new buffers after routing. This allows for individual optimization of the drive strengths dependent on the actual wire parasitics.

For each design, several layouts have been made with different row utilization ratios. (Row utilization ratio is the accumulated cell length divided by the accumulated row length.) In Table 1, the layouts with shortest delay are

Table 1. Comparison of adders

Adder	W/o wires		With wires	
	Delay [ns]	Area [mm^2]	Delay [ns]	Area [mm^2]
RCA 64 (org)	33.94	0.022	32.10	0.044
RCA 64 (reb)	29.53	0.021	30.34	0.045
RCA 64 (opt)	3.52	0.118	3.75	0.196
RCA 64 (fcs)	31.46	0.011	31.46	0.011
RCA 64 (fchp)	23.54	0.015	23.54	0.015
HCA 64 (org)	3.76	0.059	4.04	0.091
HCA 64 (reb-a-o)	3.28	0.075	3.86	0.123
HCA 64 (reb)	3.22	0.070	3.92	0.110
HCA 64 (opt)	2.10	0.131	2.60	0.212
HCA 64 (fcs)	1.98	0.358	2.13	0.416
HCA 64 (fchp)	1.07	0.546	1.17	0.618
Simple (opt)	2.42	0.120	3.34	0.270

presented. For RCA64 and HCA64, *org* means the original hand-written Verilog netlist according to the structures in Section 2. Standard cells with minimum drive strengths have been used. A slight improvement can be achieved by using a synthesis tool (BuildGates from Cadence) which resizes the drive strengths and introduces buffers based on estimated wire parasitics. These designs are referred to as *reb* in Table 1.

The HCA64 is based on an AND-OR (AO21) standard cell. In the standard-cell library that was used, this cell is only available in one drive strength. Therefore, a second version of HCA64 was implemented where the AO21 cells were replaced by separate AND and OR gates. This design is referred to as *reb-a-o* in Table 1.

Three optimized designs were also implemented referred to as *opt* in Table 1. By optimized we mean letting the synthesis tool use its full capability to select adequate cells and sizes of these cells by manipulating the original netlists. The design named *Simple* is based on a carry-lookahead adder that was generated by logic synthesis of a plus sign in VHDL.

3.2. Full-Custom Design Flow

The layouts have been implemented using an ordinary layout editor (Led from Mentor Graphics) and spice netlists with lumped capacitances have been extracted. The extracted netlists have been simulated using *HSpice*™ (level 49 and process parameters provided by technology). To get delays without wires, the capacitances that represent those wires have been removed from the spice netlist.

Each kind of adder has been implemented using two different circuit techniques, one ordinary static, *fcs* and one individually selected high-performance circuit technique *fchp*. The inter-bitslice interconnections are routed in a similar fashion regardless of circuit techniques.

Table 2. Comparison of cells

Cell	Area [μm^2]	Power [μW/MHz]	Delay [ns]
FA	364	1.28	0.53
FA (fcs)	180	0.599	0.55
FA (fchp)	230	0.769	0.37
AO21	109	0.65	0.39
AO21 (fcs)	366	0.761	0.42
AO21 (fchp)	555	1.736	0.12

4. Results and Discussion

There are three interesting results that can be extracted from Table 1. First we compare the RCA64 implementations then we compare the HCA64 implementations, and finally we compare the optimized adders. Unfortunately, no power figures could be provided for the standard-cell implementations due to incompatible power libraries. In Table 2, a cell comparison between the two flows is presented. The delays are the critical ones, e.g. input carry to output carry for the FA. All cells have been characterized in an application-like environment, i.e. driving and loading gates are the same as in the adder structures.

4.1. Ripple-Carry Adder Comparison

In the full-custom implementations of the RCA64, inter-bitslice interconnections are negligible and therefore do not contribute to the total adder delay. The high-performance implementation of the RCA (fchp) is still a static one, therefore it would be possible to reduce the delay even further by employing a dynamic circuit technique.

The standard-cell implementation, on the other hand, suffers from an unstructured layout which significantly increases wire parasitics. However, since the drive strengths are increased during rebuffering and P&R, the standard-cell RCA64 (reb) can compete with RCA (fcs) in terms of delay but requires a much larger area. The small area of the full-custom RCAs are due to an exclusive use of minimum-sized transistors. The superiority of the full-custom designs was expected and in accordance with the results in [3].

The shorter delay with wires for RCA (org) is caused by the IPO which has introduced buffers and consequently reduced the delay.

4.2. Han-Carlson Adder Comparison

In the full-custom implementations of the HCA, there are inter-bitslice interconnections spanning as many as 32 bitslices. These inter-bitslice wires contribute to the total adder delay and pose a challenge for the full-custom designer.

The standard-cell implementations present an unstructured layout like for the RCA cases and suffer from similar types of wire parasitics. The HCA64 (reb-a-o) is slightly

better because it makes use of standard cells that are available with many different drive strengths. As a consequence, no extra buffers have to be inserted.

When comparing the delays of the best standard-cell based adder and the full-custom adders in Table 1, we see that the P&R tool produces layout with about the same delay as the ordinary static full-custom design, whereas the design using high-performance circuit techniques is considerably faster. Our interpretation is that the tool performs place and route as efficiently as a designer for designs with inter-bitslice interconnections, but that full-custom designs reach higher performance through the use of advanced circuit techniques.

If we consider a 128-bit adder, having wires that span 64 bitslices, an extrapolation of our results indicates that the P&R tool would produce a layout which is better than the structured bitsliced full-custom layout. This is due to the fact that the length of the critical wires in full-custom bitsliced layouts is proportional to width of adder, but in standard-cell based layouts the length is proportional to the square root of the width. For small tree adders full-custom layouts are preferable, but for wide adders standard-cell based layouts are better since the routing-dependent delay scales better with the adder width.

Other data path components, such as barrel shifters and reduction-tree multipliers, also suffer from similar problems and the same analysis pertains to them.

Having only three metal layers has been a limitation for both the full-custom designer and the P&R tool. More modern process technologies supporting up to eight metal layers would probably shift the outcome of our study. We expect the P&R tool to be better at exploiting the freedom and complexity that comes with these extra metal layers.

4.3. Optimized Adder Comparison

Another interesting result, although it is expected, is the fact that the performance of the optimized adders depends to a large extent on the initial specification. When supplying a Han-Carlson adder netlist, the synthesis tool manages to construct an adder which is 22% faster than the adder which is synthesized outgoing from the plus sign in VHDL. We also note that when we let the synthesis tool optimize the ripple-carry adder, the delay reduction is huge and a netlist quite different from that of a ripple-carry adder has been generated.

4.4. Estimated Design Times

Since the design times have not been explicitly measured, it is hard to give any precise figures. However, a rough estimate would be: 2 hours for a synthesized RCA and 4 hours for a full-custom one. For the HCAs, on the other hand, the estimated time would be 6 hours for the synthesized and 24 hours for the full-custom. All these figures are based on design flows that are up and running.

5. Conclusion

In this paper standard-cell and full-custom implementations have been compared. Two kinds of 64-bit adder topologies have been used as test vehicles, namely the ripple-carry and Han-Carlson adders. The adders have been designed using a 0.35 μm technology.

We find that full-custom design outperforms standard-cell design mainly through a more aggressive circuit design (domino technique and detailed sizing). The more aggressive circuit technique exhibits delay reductions of 50-70% compared to the ordinary static technique. Using an ordinary static circuit technique in full-custom design flow, makes the adder delays comparable. For the small regular ripple-carry adder, the full-custom design is about 3 times smaller than the standard-cell design, indicating that full-custom is preferable for very regular designs. For the less regular Han-Carlson adder the routing of the standard-cell design is as good as the full-custom design showing less or little need for manual routing for designs with inter-bitslice interconnections. With data paths exceeding 64 bits, tools will probably outperform bitsliced full-custom routing, which is commonly used in modern processors.

6. Acknowledgments

This work was financially supported by the INTELECT programme of the Swedish Foundation for Strategic Research (SSF).

[1] C.A. Gray, M.J.S. Smith, J. Rowson, and M. O'Brian, "A Comparison of an ASIC Synthesized Design to a Schematic Entry Design for a Viterbi Decoder", In *Proceedings of the 1991 Custom Integrated Circuits Conference*, 1991, pp. 11.1.1-11.1.4.

[2] D.G. Chinnery and K. Keutzer, "Closing the Gap Between ASIC and Custom: An ASIC Perspective", In *Proceedings of the 2000 Design Automation Conference*, 2000, pp. 637-642.

[3] W.J. Dally and A. Chang, "The Role of Custom Design in ASIC Chips", In *Proceedings of the 2000 Design Automation Conference*, 2000, pp. 643-647.

[4] T. Han and D.A. Carlson "Fast Area-Efficient VLSI Adders", In *Proceedings of the 8th IEEE Symposium on Computer Arithmetic*, 1987, pp. 49-56.

[5] S. Knowles, "A Family of Adders", In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 277-281.

[6] A.M. Shams and M.A. Bayoumi, "A Novel High-Performance CMOS 1-Bit Full Adder Cell", *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 478-481, 2000.

[7] S.K. Mathew, R.K. Krishnamurthy, M.A. Anders, R. Rios, K.R. Mistry, and K. Soumyanath, "Sub-500-ps 64-b ALUs in 0.18-μm SOI/Bulk CMOS: Design and Scaling Trends", *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1636-1646, 2001.