IEICE TRANS. FUNDAMENTALS, VOL.E86–A, NO.12 DECEMBER 2003

LETTER *Special Section on VLSI Design and CAD Algorithms*

Experimental Study on Cell-Base High-Performance Datapath Design

Masanori HASHIMOTO[†a)], ***Regular Member,*** **Yoshiteru HAYASHI**[†], ***Nonmember,*** ***and*** **Hidetoshi ONODERA**[†], ***Regular Member***

SUMMARY This paper experimentally investigates the effectiveness of regularly-placed bit-slice layout and transistor-level optimization to datapath circuit performance. We focus on cell-base design flows with transistor-level circuit optimization. We examine the effectiveness through design experiments of 32-bit carry select adder and 16-bit tree-style multiplier in a 0.35 μm technology. From the experimental results, we can scarcely observe that manual cell placement contributes to improve circuit performance. On the other hand, transistor-level circuit optimization is so effective that circuit delay is reduced by 11–20% and power dissipation decreases to 42–62%. We can see that, in the case of cell-base design, transistor-level optimization is also important as well as in the case of custom design, whereas cell-base bit-slice layout has less importance to circuit performance.

key words: *datapath design, bit-slice layout, transistor sizing, cell-base design*

1. Introduction

In SoC/ASIC design, cell-base design with automatic CAD tools is widely used. However performance of circuits designed automatically by CAD tools is generally low compared with that of custom-designed circuits. Traditionally arithmetic execution units, such as adder and multiplier, are designed by datapath design method [1]. The regularity of circuit structure is utilized for efficient layout design, and both compact and high-performance layouts are realized. However, extracting circuit regularity and devise an efficient layout strategy is not suitable for design automation. Then datapath circuits are, in most cases, designed manually in full-custom design style. Therefore datapath design usually requires a long design time. Circuit designers eagerly demand highly-automated short-time design methodologies for high-performance circuits whose performance is close to that of full-custom design.

Recently cell-base design is partially enhanced to realize transistor-level optimization with minimum additional efforts [2], [3]. These design methodologies exploit well-established design automation framework for cell base design, i.e logic synthesizer generates gate-level netlist and P&R tool places cells and routes interconnects. Transistor-level optimization is realized by replacing cells. References [2], [3] generate plenty of varieties in transistor sizes for each logical function according to the optimization results.

In this paper, we experimentally examine design methods for high-performance datapath circuits exploiting cell-base design framework. Generally, performance of datapath circuits is thought to be improved by transistor-level circuit optimization and regular datapath layout that considers signal flows [4]. Reference [5] reports that regular bit-slice cell placement reduces layout area by 64%. However the evaluated circuit includes large register file. The effectiveness for arithmetic execution units is not clear because the layout efficiency of array-style register file is so different from that of automatically-placed register file. We evaluate the effectiveness of these two techniques for arithmetic execution unit design, that is how much performance of adder and multiplier is improved by regularly-placed layouts and transistor sizing.

2. Regularly-Placed Bit-Slice Layout

We here examine the performance improvement gained by regularly-placed layout that considers signal flows. We take up a 32-bit adder and a 16-bit multiplier as evaluation targets. We design two layouts for each circuit: (1) place cells manually considering signal flows; (2) place cells automatically by CAD tools. We then compare the circuit performance of both the layouts.

2.1 Design Circuits

2.1.1 32-Bit Carry Select Adder

We design a 32-bit carry select adder (CSA). Figure 1 shows the structure of a carry select adder. A, B are two data inputs, C is the carry, and S is the sum. RCA is a ripple-carry adder, and MUX is a multiplexer. In a carry selector adder, the partitioned block sizes, i.e. the bit sizes of RCA, are important for reducing propagation delay. The delay of well-

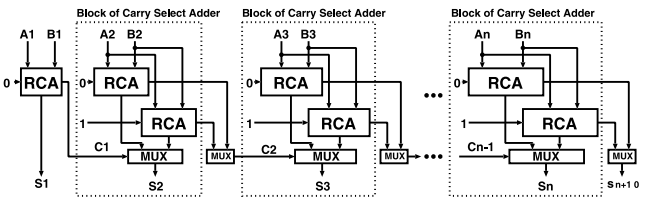

Fig. 1 Carry select adder.

Manuscript received March 13, 2003.
Manuscript revised June 9, 2003.
Final manuscript received July 16, 2003.

[†]The authors are with the Department of Communications and Computer Engineering, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: hasimoto@i.kyoto-u.ac.jp

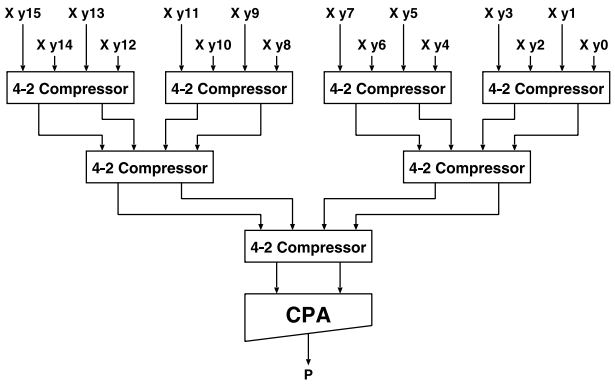

Fig. 2 Organization of 16-bit tree-style multiplier using 4-2 compressors.

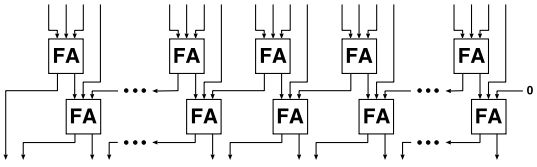

Fig. 3 4-2 compressor.

designed CSA becomes the sum of the propagation delay through the first block and the delay times of the multiplexers in the other blocks. In our design, the number of blocks is 9, and the bit sizes are 2, 2, 2, 3, 4, 4, 4, 5, 6. This configuration is verified to be the best by our simple analysis. The number of used cells is 62 of FA (full adder) and 38 of MUX.

2.1.2 16-Bit Tree-Style Multiplier Using 4-2 Compressors

We design a 16-bit tree-style multiplier using 4-2 compressors [6]. This multiplier has both the merits of speed and regularity of circuit structure. Figure 2 shows the organization of the designed multiplier with 4-2 compressors. In this paper, 4-2 compressor is constructed with serially-connected full adders (Fig. 3). CPA in Fig. 2 represents a carry propagation adder. We here use the carry select adder designed in Sect. 2.1.1 as CPA. The number of cells is 256 of 2-input AND, 322 of FA, and 40 of MUX.

2.2 Layout Design

We design a carry select adder and a multiplier with 4-2 compressors in a cell-base design style. In order to evaluate the effectiveness of the bit-slice layout that considers the flow of data signals and control signals, these circuits are designed in the following two ways:

- Cells and external IO pins are placed manually considering signal flows and regularity of circuit structure. Interconnects are routed automatically by CAD tools.
- Cell placement and routing are executed by automatic CAD tools. External IO pins are placed manually.

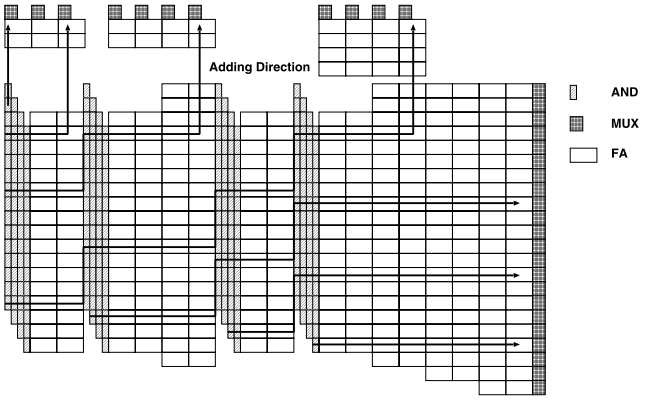

Fig. 4 Partially-bent layout of 16-bit multiplier.

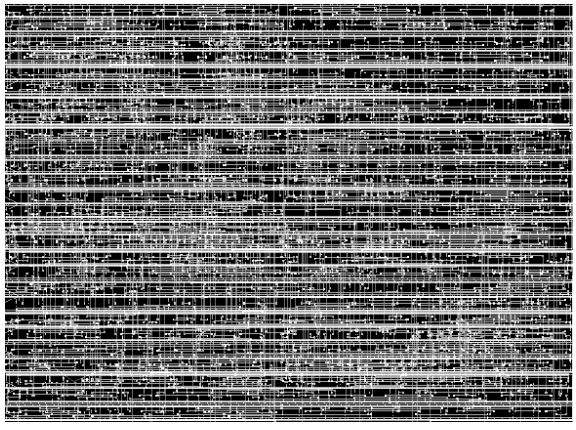

Fig. 5 Automatically-placed layout of 16-bit multiplier with 4-2 compressors.

In the automatic cell placement, we do not use a timing-driven placement option. We experimentally confirmed that the timing-driven placement scarcely, or rather never contributed to performance improvement in the experimental circuits. We compare the circuits designed in the above two ways from the point of circuit delay and wire length. We assume a $0.35\,\mu$m process technology with three metal layers.

In the case of multiplier, several layout strategies are proposed [7], [8]. We here adopt a layout strategy shown in Fig. 4. This layout is partially bent for reducing dead space. The layout area assigned for manually-placed layout and automatically-placed layout is the same. The layout area of 32-bit CSA is $84.0\,\mu\mathrm{m} \times 403.2\,\mu\mathrm{m}$, and that of 16-bit multiplier is $515.2\,\mu\mathrm{m} \times 378.0\,\mu\mathrm{m}$. The designed layouts of 16-bit multiplier are shown in Figs. 5, 6. The signal flows in the manually-placed layout are systematic, whereas the interconnects in the automatically-placed layout are disorderly.

2.3 Performance Evaluation

We evaluate the performance of the designed circuits. Table 1 shows the total wire length and the longest path delay. The path delay times are evaluated using a transistor-level

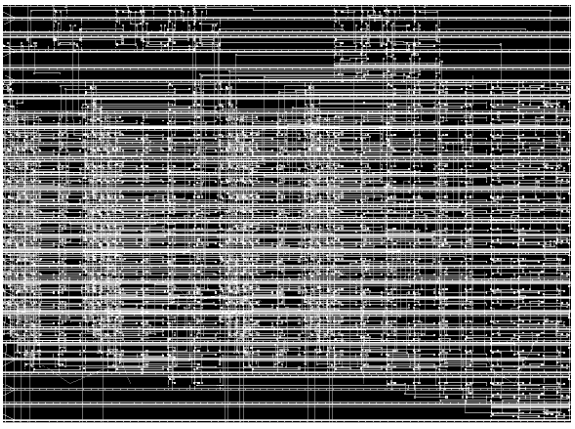

Fig. 6 Manually-placed layout of 16-bit multiplier with 4-2 compressors.

Table 1 Performance comparison.

Circuit		Wire Length [μm]	Delay [ns]
CSA	Auto	9283	4.52
	Manual	10422	4.54
Multi-plier	Auto	80395	7.52
	Manual	71941	7.57

Table 2 Relationship between performance and layout area.

Area [μm$\times\mu$m]	Core Utilization	Wire Length [μm]	Delay [ns]
515.2×378.0 (1.00)	0.65	80395	7.52
474.6×378.0 (0.92)	0.71	74656	7.60
449.4×378.0 (0.87)	0.75	71886	7.47
424.2×378.0 (0.82)	0.79	71782	7.59
399.0×378.0 (0.77)	0.84	68373	7.70

timing analyzer [9]. As you see, the delay times of both the layout styles are almost the same. The difference of wire length is only about 10%. We can not find a distinct superiority of manually-placed datapath layout that compensates a long design time.

When we place cells manually considering signal flows and circuit regularity, dead space tends to become large. It is difficult to devise an efficient layout strategy both for regularity and compact area. In the case of our designed multiplier, the core utilization ratio, which is defined as (cell area)/(layout area), is 0.65. 35% of layout area is dead space even though we adopt the partially-bent layout to decrease dead space. On the other hand, in the case of automatic P&R tools, layout area is reduced easily by trial and error. We evaluate the performance of the automatically-placed layouts in the case that layout area is reduced as far as routing can be completed. Table 2 shows the performance when layout area decreases. The layout area listed in the bottom line is the minimum area limited by routing. As you see, P&R tool realizes more compact and shorter-interconnect layout. Also circuit delay does not change so much. Without a sophisticated layout strategy, automatic P&R tools, in most cases, provide compact layout.

These results imply that regularity extraction from synthesized netlists hardly helps to increase circuit performance whereas its problem complexity is very high. We can obtain layouts whose performance is close or rather superior to regularly-placed bit-slice layout by the following easy way; generating gate-level netlists by logic synthesizer or module generator and making layouts by automatic P&R tools.

3. Transistor-Level Optimization

We examine the effectiveness of transistor-level optimization to datapath circuit performance. We here assume the semi-custom design methodology proposed in Ref. [3]. Various driving-strength cells are generated on the fly according to the optimization result, and newly-generated cells replace pre-optimized cells. Transistor-level optimization hence can be executed in cell-base design. Reference [3] can reduce transistor sizes while keeping interconnects unchanged, thanks to the preserved pin positions inside cells. We then do not consider interconnect modifications after transistor-level optimization. Even if other design methodologies with ECO (Engineering Change Order) technique partially modify interconnects, the effectiveness of transistor sizing is not expected to change considerably.

We use a cell library whose cell height is nine interconnect pitches, and the transistor sizes inside cells are relatively small. The initial netlists consist of standard-size (1x) cells. This is because 1) the interconnects are not so long, 2) there are not high fanout instances, and 3) weak cells such as 0.5x are not included so commonly in usual standard cell libraries except buffer and inverter cells. In this paper, we basically evaluate continuous transistor sizing. The difference in performance improvement between the continuous sizing and the discrete sizing with some weak cells is reported in Ref. [12].

3.1 32-Bit Carry Select Adder

We optimize the carry select adder designed in Sect. 2.2. When all transistors in the circuit are resized independently, the number of variables is tremendous and a sophisticated optimizer with high-speed transistor-level timing analysis is necessary. As a first-step trial, we optimize the circuits in the following primitive way. We here select all 28 transistors in a full adder cell as variables. All the full adder cells in the 32-bit carry select adder are the same. In this circuit, there are two types of multiplexer; a multiplexer in each block of carry select adder and multiplexers between blocks (Fig. 1). The multiplexer cells for each type are the same circuit. Therefore the number of variable is 52(=28+12+12). We construct a transistor size optimizer with a non-linear optimization package (FSQP) [10] and a transistor-level timing analyzer [9]. We first minimize the longest path delay. We then reduce the total transistor width under the condition that the circuit delay is allowed to increase by 3%.

The optimization results are shown in Table 3. The column of "Tr. Width" means the sum of all transistor width

Table 3 Transistor-level optimization results in 32-bit carry select adder.

Layout	Tr.-level Opt.	Tr. Width [μm]	Power [mW]	Delay [ns]
Auto	Initial	4830	6.07	4.52
	Optimized	2268 (−53%)	3.10 (−49%)	3.62 (−20%)
Manual	Initial	4830	6.07	4.54
	Optimized	2052 (−58%)	2.54 (−58%)	3.85 (−15%)

Table 4 Transistor-level optimization results in 16-bit multiplier.

Layout	Tr.-level Opt.	Tr. Width [μm]	Power [mW]	Delay [ns]
Auto	Initial	17964	39.50	7.52
	Optimized	12684 (−29%)	24.48 (−38%)	6.44 (−14%)
Manual	Initial	17964	39.60	7.57
	Optimized	13023 (−28%)	23.36 (−41%)	6.76 (−11%)

in the circuit. "Power" is the average power dissipation evaluated by a transistor-level power simulator [11]. The input patterns are generated randomly, and the cycle time is 10 ns. The number of the applied patterns is 100. The row of "Auto" corresponds to the optimization result of the automatically-placed layout. "Manual" is the result of the manually-placed layout. The numbers inside parentheses represent the reduction from the initial circuits. The circuit delay and power dissipation are reduced similarly in the manually-placed layout and the automatically-placed layout. The circuit delay is reduced by 15 to 20%, and the power dissipation is reduced by 50%. Thus transistor-level circuit tuning is indispensable for high-performance datapath circuit design.

3.2 16-Bit Tree-Style Multiplier Using 4-2 Compressors

We optimize the multiplier designed in Sect. 2.2. A 1-bit 4-2 compressor consists of two full adder cells, and there are 28 transistors in each full adder cell. We choose 28 transistor sizes in a full adder as variables. We also add all transistor sizes in a 2-input AND cells that generate partial products. The total number of variables is 34. All the full adders used inside 4-2 compressors in the multiplier are the same. The 2-input AND cells are also the same. The 32-bit carry select adder optimized in Sect. 3.1 is imported as the carry propagation adder in the final stage. Table 4 shows the optimization results. The circuit delay of the manual layout and of the automatic layout is reduced by 11% and 14% respectively. The power consumption decreases to 59% and 62%. The evaluation in this section is preliminary because the number of variables are small and hence the true optimum solution, we think, is not found. Nevertheless the effectiveness of transistor sizing considerably. We will enhance our transistor sizing framework [12] so that not only simple leaf cells but also over twenty-transistor cells, such as carry select adder and 4-2 compressor, can be handled in order to design high-performance datapath circuits.

4. Conclusion

We study a high-performance datapath design methodology exploiting well-established cell-base design framework. We experimentally investigates the effectiveness of bit-slice layout that considers flows of data signal and control signal. 32-bit carry select adder and 16-bit tree-style multiplier using 4-2 compressors are designed in a 0.35 μm process. We can not find a significant performance difference between the manually-placed layout and the automatically-placed layout, as far as arithmetic execution units we investigate. The performance improvement by manual cell placement is small although the design time required for manually-placed layout is much larger. Transistor-level optimization is commonly thought to improve circuit performance. We observe that transistor sizing contributes to reduce both circuit delay and power dissipation considerably. Our future work includes developing an effective and practical transistor-level optimization technique.

Acknowledgement

This work is supported in part by the 21st Century COE Program (Grand No. 14213201).

References

[1] M.J.S. Smith, Application-Specific Integrated Circuits, Addison Wesley Longman, 1997.
[2] G.A. Northrop and P.-F. Lu, "A semi-custom design flow in high-performance microprocessor design," Proc. DAC, pp.426–431, 2001.
[3] H. Onodera, M. Hashimoto, and T. Hashimoto, "ASIC design methodology with on-demand library generation," Proc. Symposium on VLSI Circuits, pp.57–60, 2001.
[4] D.G. Chinnery and K. Keutzer, "Closing the gap between ASIC and custom: An ASIC perspective," Proc. DAC, pp.637–642, 2000.
[5] W.J. Dally and A. Chang, "The role of custom design in ASIC chips," Proc. DAC, pp.643–647, 2000.
[6] A. Chandrakasan, W.J. Bowhill, and F. Fox, Design of High-Performance Microprocessor Circuits, IEEE Press, 2001.
[7] N. Itoh, Y. Naemura, N. Makino, Y. Nakase, T. Yoshihara, and Y. Horiba, "A 600-MHz 54×54-bit multiplier with rectangular-styled Wallace tree," IEEE J. Solid-State Circuits, vol.36, no.2, pp.249–257, Feb. 2001.
[8] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54×54-b multiplier using pass-transistor multiplexer," IEEE J. Solid-State Circuits, vol.30, no.3, pp.251–257, March 1995.
[9] PathMill Reference Manual, Synopsys, CA, 1999.
[10] C. Laurence, J.L. Zhou, and A.L. Tits, "User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints," April 1997.
[11] PowerMill Reference Manual, Synopsys, CA, 1999.
[12] M. Hashimoto and H. Onodera, "Post-layout transistor sizing for power reduction in cell-base design," IEICE Trans. Fundamentals, vol.E84-A, no.11, pp.2769–2777, Nov. 2001.