An Area-Efficient Iterative Modified-Booth Multiplier Based on Self-Timed Clocking

Myoung-Cheol Shin, Se-Hyeon Kang, and In-Cheol Park
Dept. of EECS, Korea Advanced Institute of Science and Technology, Taejon, Korea
{mcshin,shkang}@ics.kaist.ac.kr, icpark@ee.kaist.ac.kr

Abstract

A new iterative multiplier based on a self-timed clocking scheme is presented. To reduce the area required for the multiplier, only two CSA rows are iteratively used to complete a multiplication. The partial CSA array is controlled by a fast internal clock generated using a self-timed technique. Compared with the array implementation, the proposed multiplier yields an 86.6% area reduction at the expense of 18.8% slow down for 64×64-bit multiplication.

1. Introduction

Area-efficient and fast multipliers are essential building blocks for high-performance computing and digital signal processing systems. In those applications multipliers should be small enough so that a large number of them can be integrated on a single chip. The iterative multipliers using a partial CSA array are regarded as the most suitable architecture for those applications.

This paper describes a new iterative modified-Booth multiplier that works at a self-timed clock. It uses a very fine-grained iteration, *i.e.* it contains only two CSA rows, which leads to a large reduction in area. In order to provide the fast clock and control signals required to the fine-grained architecture, previous designs such as Stanford Pipeline Iterative Multiplier (SPIM) used internal clocks generated by inverter chains [1], the length of which should be adjusted manually. We implemented the internal clock using self-timed clocking, which does not require manual clock tuning, minimizes timing waste, and allows the multiplier to be integrated with any system regardless of the system clock.

2. Multiplier architecture

In the proposed multiplier, a partial CSA array that consists of two CSA rows runs at the clock internally generated by a self-timed technique. The multiplier therefore looks like a combinational logic from the outside. We selected the structure with two rows rather than one because it reduces register delay by half and enables a technology-independent self-timed clock generator with dummy cells.

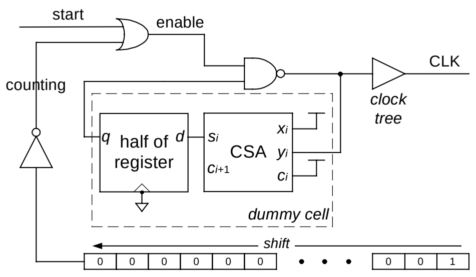

Figure 1. The self-timed clock generator

Fig. 1 shows the clock generator used in this design. It contains a dummy CSA cell and a half of a register cell, *e.g.* the master latch of a master-slave flip-flop, which are connected as a ring oscillator. Because the generated clock period is twice as long as the delay of the ring oscillator, it is closely matched to the delay of two CSA's and one register in the critical path. This minimizes the timing waste of the iterative multiplier. Since the dummy CSA cell is connected so as to yield a maximum delay and the NAND gate adds extra delay, the clock period is guaranteed to be no shorter than the longest path delay of the entire design. If the CSA's and registers in the datapath slow down due to the process variation, the clock generator cells slow down by the same rate.

The clock generator operates as follows. When an external start signal is issued, the enable signal goes high, and the internal clock starts. The clock signal is buffered by a balanced clock tree and distributed throughout the datapath. A one-hot encoded shift register is used to count the clock cycles and generate control signals for the datapath. When the iterative operation is completed, the '1' in the shift register arrives at the end and the clock then stops and stays low until the next start signal arrives. The multiplier consumes much less power when it is idle since the clock of the entire multiplier is cut off.

The pipeline architecture of the $n\times n$-bit proposed multiplier is shown in Fig. 2. The shaded elements are pipeline registers. Because slow static CMOS registers may yield a large portion of the delay, all the registers used in our design are much faster true single-phase clock (TSPC) registers [2].

A weighted carry-save adder (WCSA) [3] in Fig. 2 processes the least significant two bits at once to avoid increasing number of CSA's necessary to modified Booth recoding. The detailed description on the operation and array structure including WCSA's can be found in [3]. We used variations of a transmission gate adder for the CSA's and the WCSA's, all of the outputs of which are no slower than the 'sum' output of a CSA.

The pipeline progresses as follows. Multiplicand and Multiplier operands are latched at the beginning of a multiplication. At the first stage, the shift register shifts Multiplier to the right by four bits, Booth recoders recode the least significant five bits, and Booth multiplexers select two partial products.

The iterations of a partial CSA array are performed at the second stage. Two rows of CSA's add the feedback carry and sum outputs from the previous iteration and two partial products from the first stage. Two WCSA's calculate the least significant four bits of the CSA output.

At the third stage, a shift register receives the WCSA output and shifts its content to the right by four bits at every iteration cycle. When the iteration finishes, the carry and sum outputs of the CSA array are added at the CPA. We get (n+2) most significant bits from the CPA output, 2 bits from the upper WCSA, and (n–4) least significant bits from the shift register to form the final 2n-bit product.

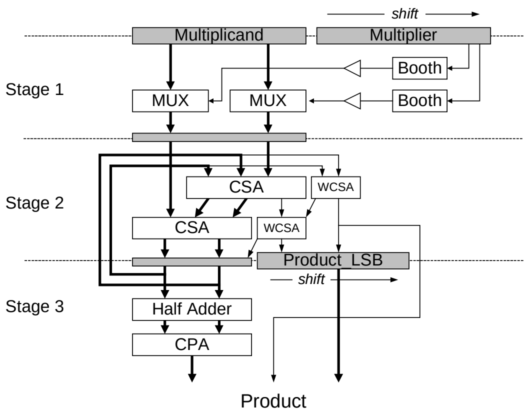

Figure 2. The pipeline architecture of the proposed multiplier

Table 1. The area of the layout (unit: mm^2)

	proposed	array	add & shift
16×16	0.0542	0.1190	0.0297
32×32	0.1064	0.4153	0.0581
64×64	0.2108	1.569	0.1119

Table 2. The worst-case latency (unit: ns)

	proposed	array	add & shift
16×16	9.194	6.308	22.52
32×32	14.52	11.01	51.47
64×64	24.64	20.00	141.5

3. Experimental results

To compare the area and the performance, we implemented and simulated the proposed multipliers, conventional array multipliers, and classical add-and-shift multipliers of three different sizes. We applied the modified Booth recoding to all of them and implemented them with the same datapath cells in a 0.35μm CMOS technology.

The area comparison of the surrounding rectangles of the layouts is given in Table 1. The proposed multiplier occupies much smaller area than the array implementation and is about twice as large as the add-and-shift multiplier. As the operand size increases, the difference between the proposed one and the array multiplier becomes more significant. For the 64×64-bit implementations, the proposed one occupies only 13.4% area of the array multiplier.

We measured the worst-case latencies using the HSPICE simulation with the worst-case input patterns at 3.3V power supply and the junction temperature of 85°C. The results are shown in Table 2. The latency of the proposed multiplier is close to the array multiplier. In the case of 64×64-bit multiplication, it shows 18.8% performance degradation due to the pipeline register delay and clock margin. The add-and-shift multiplier is much slower than the others. The proposed one has the smallest product of 'area×latency' in any case.

4. Conclusion

We have proposed a new area-efficient iterative modified-Booth multiplier. A clock generated by a self-timed technique enables the best performance by minimizing the clock timing waste. The clock generation circuit requires no adjustment, and the validity of the clock is insensitive to process variation. Other techniques such as TSPC registers and modified Booth structure with WCSA's are used to reduce the area and to improve the performance.

Experimental results show that it reduces the area drastically while maintaining the performance. The silicon area reduction becomes more significant as the bit width increases. We believe that it is suitable for use in large systems that require a lot of small multipliers because the proposed iterative multiplier has small area, reasonable performance, and independent self-timed control.

References

[1] M. R. Santoro and M. A. Horowitz, "SPIM: a pipelined 64×64-bit iterative multiplier," *IEEE J. Solid-State Circuits*, vol.24, no. 2, pp. 487–493, Apr. 1989.

[2] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 62–70, Feb. 1989.

[3] B-I. Park, I-C. Park, and C-M. Kyung, "A regular layout structured multiplier based on weighted carry-save adders," *Proc. IEEE ICCD '99*, pp. 243–248, Oct. 1999.