High-Speed and Low-Power Multipliers Using the Baugh-Wooley Algorithm and HPM Reduction Tree

Magnus Själander and Per Larsson-Edefors
Department of Computer Science and Engineering
Chalmers University of Technology, SE-412 96 Göteborg, Sweden

***Abstract*— The modified-Booth algorithm is extensively used for high-speed multiplier circuits. Once, when array multipliers were used, the reduced number of generated partial products significantly improved multiplier performance. In designs based on reduction trees with logarithmic logic depth, however, the reduced number of partial products has a limited impact on overall performance. The Baugh-Wooley algorithm is a different scheme for signed multiplication, but is not so widely adopted because it may be complicated to deploy on irregular reduction trees. We use the Baugh-Wooley algorithm in our High Performance Multiplier (HPM) tree, which combines a regular layout with a logarithmic logic depth. We show for a range of operator bit-widths that, when implemented in 130-nm and 65-nm process technologies, the Baugh-Wooley multipliers exhibit comparable delay, less power dissipation and smaller area foot-print than modified-Booth multipliers.**

I. Introduction

Multiplication is an important arithmetic operation and multiplier implementations date several decades back in time. Multiplications were originally performed by iteratively utilizing the ALU's adder. As timing constraints became stricter with increasing clock rates, dedicated multiplier hardware implementations such as the array multiplier were introduced. Since then ever more sophisticated methods on how to implement multiplications have been proposed. One of the more popular implementations is that of the modified-Booth recoding scheme together with a logarithmic-depth reduction tree and a fast final adder. Modified-Booth recoding has the advantage of reducing the number of generated partial products by half, compared to partial-product generation based on 2-input AND-gates. This fact decreases the size of the reduction circuitry, which commonly is a logarithmic-depth reduction tree, e.g. Wallace, Dadda or TDM. Since such reduction trees are infamous for their irregular structures, which make them difficult to place and route during the physical layout of a multiplier, a decreased size of the reduction circuit eases the implementation and improves the performance of the multiplier.

Modified-Booth implementation strategies are commonly motivated by the need for fast multipliers. However, with the ongoing integration trend for which power dissipation is an ever pressing concern, modified Booth is no longer the obvious implementation choice. Already in 1997 Callaway *et al.* showed that, for a 2 μm process technology, a Wallace multiplier is more energy efficient than a modified-Booth multiplier [1]. Even though power has become a bigger concern since then, the modified-Booth multiplier is still prevailing as the main implementation choice for high-speed multipliers.

In this paper we compare a multiplier based on the modified-Booth algorithm against one based on the less used Baugh-Wooley algorithm. As will be shown in Sec. V, a Baugh-Wooley multiplier implemented in a 130-nm and 65-nm process technology is more power and energy efficient than a modified-Booth multiplier of equal bit-width. This efficiency comes with only an insignificant increase in delay. We will subsequently show that the high power dissipation makes modified Booth a poor implementation choice for high-speed multipliers.

II. Modified-Booth Multiplication

The modified-Booth (MB) algorithm [2] guarantees that only half the number of partial products will be generated, compared to a conventional partial-product generation using 2-input AND gates. The reduction of partial-product rows is achieved by encoding the multiplier into $\{-2,-1,0,1,2\}$, which is then multiplied with the multiplicand. An MB multiplier therefore works, internally, with a two's complement representation of the partial products. To avoid having to sign extend the partial products, we are using the scheme presented by Fadavi-Ardekani [3]. In the two's complement representation, a change of sign includes the insertion of a '1' at the least significant bit (LSB) position (henceforth called LSB insertion). To avoid an irregular implementation of the partial-product reduction circuitry, we draw on the idea called modified partial-product array [4]. Here, the impact of LSB insertion on the *two* least significant bit positions of the partial product is pre-computed. The pre-computation redefines the LSB of the partial product (Eq. 1 in [4]) and moves the potential '1', which results from the LSB insertion, to the second least significant position (Eq. 2). Note that our Eq. 2 is different from the corresponding equation used in [4]. Fig. 1 illustrates an 8-bit MB multiplication using sign-extension prevention and the modified partial-product array scheme.

$$p_{LSBi} = y_0(x_{2i-1} \oplus x_{2i}) \tag{1}$$

$$a_i = x_{2i+1}(\overline{x_{2i-1} + x_{2i}} + \overline{y_{LSB} + x_{2i}} + \overline{y_{LSB} + x_{2i-1}}) \tag{2}$$

The MB recoding logic can be designed in many different ways. For the subsequent comparison we have chosen to implement two different, recent recoding schemes. The first recoding scheme is the one presented by Yeh *et al.* [4], since

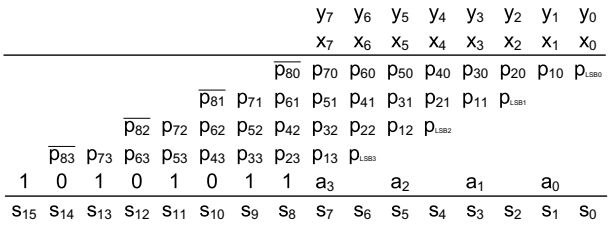

Fig. 1. Illustration of an 8-bit modified-Booth multiplication.

this is claimed to be faster than many previous schemes. We have also chosen to implement the recoding scheme of Hsu *et al.* [5]. We chose to implement this recoding scheme because it is a recent one, showing promising results. To reduce the number of partial-product bits entering the inputs of the final adder, the regular logarithmic reduction tree of the High Performance Multiplier (HPM) [6] has been used.

III. Baugh-Wooley Multiplication

The Baugh-Wooley (BW) algorithm [7] is a relatively straightforward way of doing signed multiplications; Fig. 2 illustrates the algorithm for an 8-bit case, where the partial-product bits have been reorganized according to Hatamian's scheme [8]. The creation of the reorganized partial-product array of an N-bit wide multiplier comprises three steps: *i)* The most significant bit (MSB) of the first $N-1$ partial-product rows and all bits of the last partial-product row, except its MSB, are inverted. *ii)* A '1' is added to the Nth column. *iii)* The MSB of the final result is inverted.

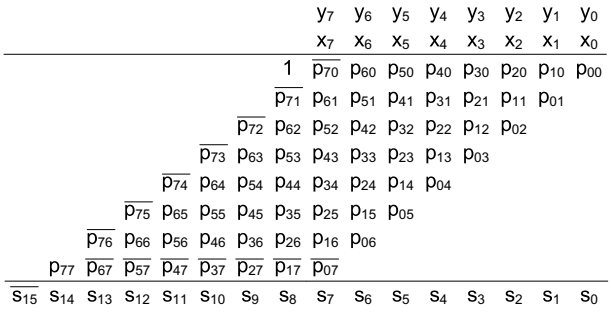

Fig. 2. Illustration of an 8-bit Baugh-Wooley multiplication.

To the best of our knowledge, performance investigations in the open literature concerning BW implementations have exclusively been based on reduction arrays, in the spirit of the original paper [7]. In this investigation, however, we have chosen to implement the BW algorithm on the HPM reduction tree [6] instead.

IV. Multiplier Evaluation Setup

To evaluate the different multiplier designs, a multiplier generator [9] was created. This generator is capable of generating gate-level VHDL netlist descriptions of modified-Booth (MB) and Baugh-Wooley (BW) multipliers of various operand bit-widths, according to the schemes presented in the previous sections. Both the MB recoding scheme of Yeh *et al.* and Hsu *et al.* have been implemented. All netlists are based on the regular HPM reduction tree [6], based on 3:2 adder (full adder) cells. A Kogge-Stone [10] adder is used as final adder, placed on the reduction tree outputs.

By exhaustively simulating all input patterns and verifying the result using Cadence NC-VHDL, the VHDL generator was verified to generate functionally correct multiplier netlists of sizes up to at least 16 bits. For multipliers larger than 16 bits, the functionality of each multiplier size was verified for a random pattern of one million vectors.

The VHDL descriptions were synthesized using Synopsys Design Compiler together with a commercially available 1.2 V 130-nm process technology. Each synthesized netlist was taken through place and route using Cadence Encounter. To create a common input and output interface to the multipliers and to enable higher place and route quality from the EDA tools, which do not handle purely combinatorial circuits well, registers were placed on the inputs and outputs of the multipliers. Delay estimates were obtained after resistance and capacitance extraction from the placed and routed netlists. Power estimates were derived from the same netlists, using value change dump (VCD) data from simulations of 10,000 random input vectors. All delay, power, and area figures that we present, include the input and output registers on the multipliers and are for the worst case corner at 1.08 V.

Initially, the timing constraints were systematically set so as to push the limits of the timing that can be achieved for each generated VHDL description. This guarantees that a very fast implementation is obtained; however, the power dissipation becomes prohibitively high, due to excessive buffering and resizing of gates. Therefore, for each implementation, the timing was subsequently relaxed with 100, 300, and 600 ps, respectively, relative to the fastest timing obtained. Correspondingly, power estimates were obtained for each timing constraint.

V. Comparison of Baugh-Wooley and Modified-Booth Multipliers

An extensive evaluation, see [11], of the two modified-Booth (MB) recoding circuits by Yeh *et al.* and Hsu *et al.* showed that the recoding circuit by Yeh *et al.* is both faster and more energy efficient. Thus, in the following comparisons, the recoding circuits of Yeh *et al.* will be used in the MB implementations.

We begin the comparison of Baugh-Wooley (BW) and MB multipliers by considering their efficiency in terms of delay and power. Fig. 3 shows the delay and power for 16-, 32-, 48-, and 64-bit multipliers. The figure shows that the MB implementation can be up to 150 ps faster than the BW implementation. However, for 32-bit operand-width the BW implementation outperforms the MB implementation. In contrast to the timing, the power dissipation is consistently and significantly lower for the BW implementations. The power dissipation of the MB implementations is 25–40% higher than that of a BW multiplier of the same bit-width, which operates at the same speed.

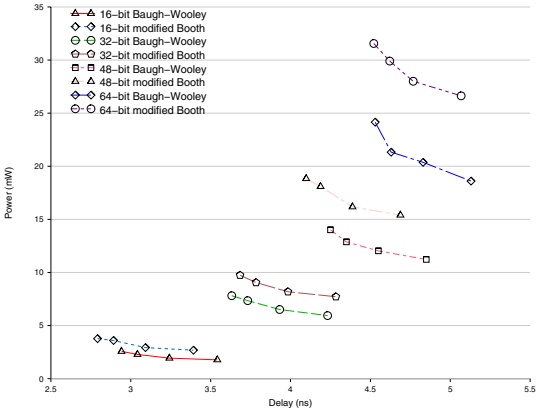

Fig. 3. Delay and corresponding power dissipation for 16-, 32-, 48-, and 64-bit instances of modified-Booth and Baugh-Wooley multipliers.

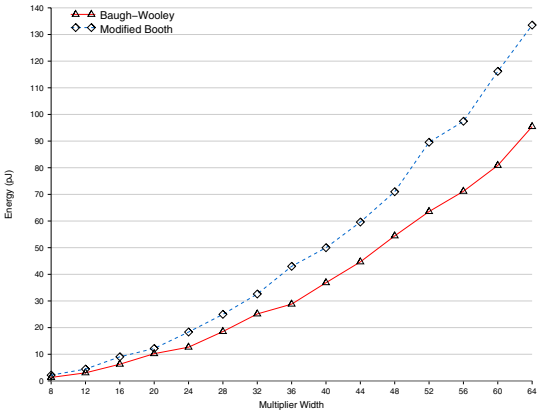

Fig. 4. The energy-per-operation for the modified-Booth and the Baugh-Wooley multiplier.

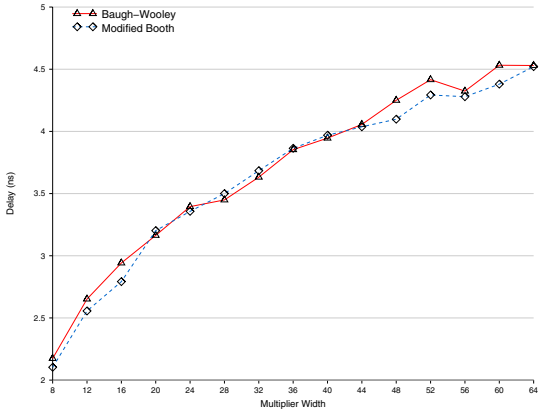

Fig. 5. The shortest delay for the modified-Booth and the Baugh-Wooley multiplier.

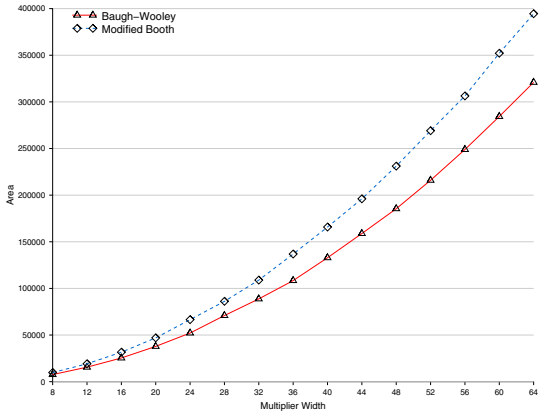

Fig. 6. The area (μm^2) for the modified-Booth and the Baugh-Wooley multiplier.

A useful metric when considering both delay and power is the energy that is required to complete a single multiplication operation. Since we are now dealing with single-cycle multipliers, the energy-per-operation is obtained as the product of power and delay for each point in the graph of Fig. 3. Fig. 4 shows the result of such a calculation for bit-widths from 8 to 64 bits: Among the four timing constraints, only the smallest energy-per-operation obtained for each bit-width is plotted. The lowest energy-per-operation is consistently obtained at a timing that is 100 ps to 300 ps slower than the fastest possible. From the graph it is clear that the BW multiplier is a much more efficient implementation in terms of energy.

Fig. 5 shows the result of the shortest delay that can be achieved, for bit-widths of 8 to 64 bits. The graph shows that the BW multiplier can match the performance of the MB multiplier for bit-widths of up to 44 bits. Furthermore, even for wider multipliers the difference is no more than 150 ps, i.e. less than 4%.

If we consider area, which is an important factor for efficient multiplier implementations, the BW multiplier also in this respect outperforms the MB multiplier. Fig. 6 shows the area of the different multiplier implementations and for all bit-widths the BW implementation consistently is the smallest. The BW implementation is about 20% smaller than that of a MB implementation of the same operand bit-width.

A. *Dissecting the Timing*

When we take a closer look at the timing contribution of *i)* the partial-product generation, *ii)* the reduction tree, and *iii)* the final adder to the total delay for MB and BW multipliers, we see that the delay that is gained by having a smaller reduction tree for the MB implementation is canceled by the large delay of the MB recoding circuitry. Table I shows that the partial-product generation of the MB multiplier is about two times as slow as the generation in the BW multiplier. This comes as no surprise, since for BW there is only one 2-input AND-gate in the critical path, while for the MB multiplier the critical path goes through the encoder followed by the decoder. Taking gate fanout into consideration, the primary inputs of a BW multiplier need to drive only N minimum-sized 2-input AND-gates, while for the MB multiplier the x primary inputs first have to drive $N/2$ encoders, which in turn drive N decoders. The delay difference of the reduction tree is not as large as

one first could expect, considering that the modified-Booth algorithm reduces the partial product rows with 50%. The small delay difference is due to the logarithmic behavior of the reduction tree that reduces the benefit of a reduced number of partial product rows. For the final adder the delay is, as expected, similar for the two implementation choices. This is due to that the critical path through the reduction tree is in the middle of the tree, for both the BW and MB implementation.

TABLE I

THE DELAY FOR THE PARTIAL-PRODUCT GENERATION, THE REDUCTION TREE, AND THE FINAL ADDER FOR TWO 32-BIT MULTIPLIERS.

	Baugh-Wooley		Modified Booth	
	Increment	Total	Increment	Total
Partial Product	0.459 (ns)	0.459 (ns)	0.953 (ns)	0.953 (ns)
Reduction Tree	2.092 (ns)	2.551 (ns)	1.679 (ns)	2.632 (ns)
Final Adder	1.081 (ns)	3.632 (ns)	1.052 (ns)	3.684 (ns)

VI. IMPLEMENTATION IN A 65-NM PROCESS TECHNOLOGY

We had limited access to a commercial 65-nm low-power process technology and used it to implement a 32-bit BW and MB multiplier with standard threshold voltage cells. For the 65-nm design flow, Cadence Encounter is used for synthesis, placement, and routing. Timing and power estimates are for the worst-case 125 °C corner at 1.1 V. The timing is for lowest delay possible and the power dissipation was estimated using value change dump (VCD) data from simulations with 10,000 random input vectors, as for the 130-nm process.

TABLE II

DELAY, POWER, ENERGY, AND AREA FOR 32-BIT BAUGH-WOOLEY AND MODIFIED-BOOTH MULTIPLIERS IN A 65-NM PROCESS.

	Delay (ns)	Power (mW)	Energy (pJ)	Area (μm^2)
Baugh	2.59 (100%)	23.4 (100%)	60.6 (100%)	48.1k (100%)
Booth	2.50 (97%)	37.5 (160%)	93.8 (155%)	52.1k (108%)

TABLE III

DELAY, POWER, ENERGY, AND AREA FOR 32-BIT BAUGH-WOOLEY AND MODIFIED-BOOTH MULTIPLIERS IN A 130-NM PROCESS.

	Delay (ns)	Power (mW)	Energy (pJ)	Area (μm^2)
Baugh	3.63 (100%)	7.81 (100%)	28.4 (100%)	88.8k (100%)
Booth	3.68 (101%)	9.74 (125%)	35.8 (126%)	108.9k (123%)

The results for 32-bit BW and MB multipliers in the 65-nm process are shown in Table II. The high power and energy dissipation of the MB multiplier does not justify the small performance advantage; 90 ps (3%) faster than the BW multiplier. Furthermore, the MB multiplier is also 8% larger in terms of area. For reference, the corresponding data (those for shortest delay) for the 32-bit 130-nm implementations are shown in Table III. The relationship between the BW and MB multiplier does not change much in terms of shortest delay. However, in relation to BW, the power and energy for the MB multiplier increase significantly with technology scaling. Regarding area, the area penalty for the MB implementation is reduced from 23% in 130-nm to 8% in 65-nm.

Considering the results for the 65-nm and 130-nm it is clear that at least for a 32-bit multiplier, a MB implementation has higher power dissipation, larger area and only a negligible delay improvement compared to a BW implementation.

VII. CONCLUSION

The modified-Booth algorithm, which is commonly used today, makes multiplier design complex and a significant design effort is needed to obtain an efficient implementation. The design decision of using the modified-Booth algorithm is most likely based on the notion that a smaller reduction circuitry achieves a better implementation. This was once true, but the introduction of logarithmic-depth reduction trees, and particularly the regular structure of the HPM reduction tree, has made the size of the reduction circuit less of a concern when designing a multiplier. The logic depth through the HPM reduction tree differs by only one or two full adders for a modified-Booth and Baugh-Wooley implementation of the same operand bit-width. Considering that the critical path of a modified-Booth multiplier is located in its encoder and decoder, it is difficult to envision a modified-Booth implementation that can be much faster than a Baugh-Wooley implementation, regardless of the recoding scheme used. Taking power, energy per operation, and area into consideration, it is clear that the gain by reducing the reduction circuitry is lost in the recoding circuitry, making a modified-Booth implementation perform worse than a Baugh-Wooley implementation.

REFERENCES

[1] T. K. Callaway and J. Earl E. Swartzlander, "Power-Delay Characteristics of CMOS Multipliers," in *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, June 1997, pp. 26–32.

[2] O.L.MacSorley, "High Speed Arithmetic in Binary Computers," in *Proceedings of the IRE*, vol. 49, no. 1, January 1961, pp. 67–97.

[3] J. Fadavi-Ardekani, "MxN Booth Encoded Multiplier Generator Using Optimized Wallace trees," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, pp. 120–125, 1993.

[4] W.-C. Yeh and C.-W. Jen, "High-Speed Booth Encoded Parallel Multiplier Design," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 692–701, July 2000.

[5] S. K. Hsu, S. K. Mathew, M. A. Anders, B. R. Zeydel, V. G. Oklobdzija, R. K. Krishnamurthy, and S. Y. Borkar, "A 110 GOPS/W 16-bit Multiplier and Reconfigurable PLA Loop in 90-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 256–264, January 2006.

[6] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, "Multiplier Reduction Tree with Logarithmic Logic Depth and Regular Connectivity," in *IEEE International Symposium on Circuits and Systems*, May 2006.

[7] C. R. Baugh and B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Transactions on Computers*, vol. 22, pp. 1045–1047, December 1973.

[8] M. Hatamian, "A 70-MHz 8-bit x 8-bit Parallel Pipelined Multiplier in 2.5-μm CMOS," *IEEE Journal on Solid-State Circuits*, vol. 21, no. 4, pp. 505–513, August 1986.

[9] M. Själander, "HMS Multiplier Generator," http://www.sjalander.com/research/multiplier, February 2008.

[10] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Transactions on Computers*, vol. 22, no. 8, pp. 786–793, August 1973.

[11] M. Själander and P. Larsson-Edefors, "The Case for HPM-Based Baugh-Wooley Multipliers," Department of Computer Science and Engineering, Chalmers University of Technology, Tech. Rep. 08-8, March 2008.