Hindawi Publishing Corporation
VLSI Design
Volume 2014, Article ID 343960, 6 pages
http://dx.doi.org/10.1155/2014/343960

Research Article

Low-Area Wallace Multiplier

Shahzad Asif and Yinan Kong

Department of Engineering, Macquarie University, Sydney, NSW 2109, Australia

Correspondence should be addressed to Shahzad Asif; shahzad.asif@mq.edu.au

Received 18 March 2014; Accepted 23 April 2014; Published 12 May 2014

Academic Editor: Yu-Cheng Fan

Multiplication is one of the most commonly used operations in the arithmetic. Multipliers based on Wallace reduction tree provide an area-efficient strategy for high speed multiplication. A number of modifications are proposed in the literature to optimize the area of the Wallace multiplier. This paper proposed a reduced-area Wallace multiplier without compromising on the speed of the original Wallace multiplier. Designs are synthesized using Synopsys Design Compiler in 90 nm process technology. Synthesis results show that the proposed multiplier has the lowest area as compared to other tree-based multipliers. The speed of the proposed and reference multipliers is almost the same.

1. Introduction

Multiplication is one of the most widely used arithmetic operations. Due to this a wide range of multiplier architectures are reported in the literature providing flexible choices for various applications. Among them the simplest is array multiplier [1] which is also the slowest. Some high performance multipliers are presented in [2–5]. The focus of this paper is Wallace multiplier [6]. Wallace multiplier uses full adders and half adders to reduce the partial product tree to two rows, and then a final adder is used to add these two rows of partial products. We call this design "TW (traditional Wallace) multiplier" in this text. TW multiplier performs its operation in three steps. (1) Generate all the partial products. (2) The partial product tree is reduced using full adders and half adders until it is reduced to two terms. (3) Finally, a fast adder is used to add these two terms.

Waters and Swartzlander [7] presented a reduced complexity Wallace multiplier by reducing the number of half adders in the reduction process. We call this design "RCW (reduced complexity Wallace) multiplier" from now on. The speed of the RCW multiplier is expected to be the same as of TW multiplier due to the equal number of reduction stages in both multipliers. The RCW uses a larger final adder as compared to the TW multiplier. A number of strategies are reported in [8–10] to improve the speed of the RCW. However, the focus of their research is to reduce the delay by using a faster final adder while still using the same reduction tree as RCW. As a result, the final adder size for the multipliers in [8–10] is the same as that of RCW.

The focus of this paper is to optimize the reduction tree in a way that can reduce the size of the final adder. The reduced size of the final adder resulted in low area of the multiplier without incurring any extra delay. We call our design "PW (Proposed Wallace) multiplier." We also considered Dadda multiplier [11] for comparison due to its similarity with the Wallace multiplier.

This paper makes a contribution in the design of Wallace treed based multipliers by proposing a strategy to reduce the area of reduced complexity Wallace (RCW) multiplier. This innovative method allows for an effective utilization of half adders in such a way that the size of the final adder is reduced. It also provides a more regular structure of the reduction tree and the final adder.

The rest of the paper is organized as follows. Section 2 discusses some previous approaches for partial product tree reduction. In Section 3, the proposed Wallace multiplier is presented. In Section 4, the choice of final adder is discussed. Section 5 evaluates the results for all the designs synthesized in Synopsys. The work is concluded in Section 6.

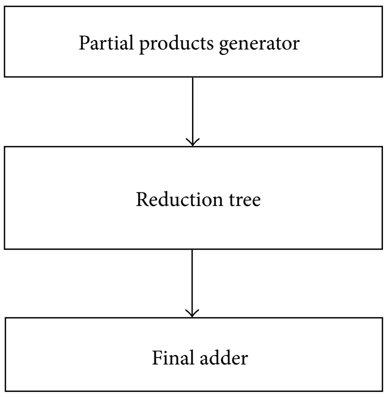

FIGURE 1: Block diagram of tree-based multipliers.

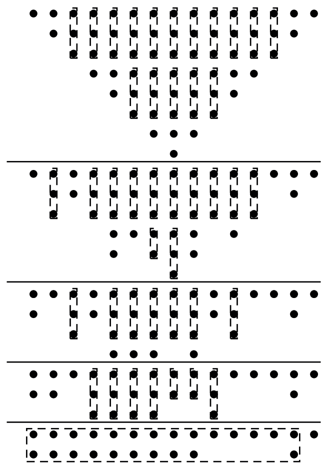

FIGURE 3: 8-bit reduced complexity Wallace reduction.

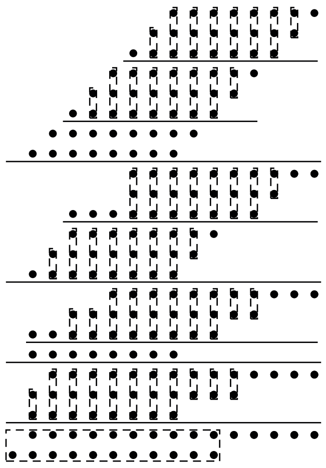

FIGURE 2: 8-bit traditional Wallace reduction.

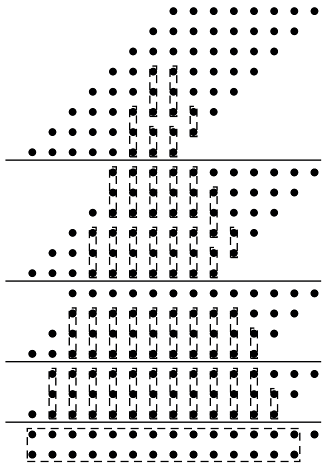

FIGURE 4: 8-bit Dadda reduction.

2. Previous Architectures

This section discusses some previous Wallace tree-based multiplier architectures. The general block diagram of tree-based multipliers is shown in Figure 1.

The dot notation [11] is used to represent the partial product tree in all the architectures discussed in this section as shown from Figures 2 to 5. The full adders and half adders are represented by boxes around the dot products. The box which encloses three dot products represents a full adder, whereas the box containing only two dot products is used to represent a half adder. The stages are separated by a thick horizontal line.

2.1. Traditional Wallace (TW) Multiplier. In TW multiplier architecture, the partial product tree is divided into groups [6]. Each stage can have one or more groups as shown in the 8-bit TW reduction process in Figure 2.

The groups in a stage are separated by a thin horizontal line. Each group consists of three rows except the last group where the number of rows can be less than three. Equation (1) calculates the number of rows in the last group for each stage as

$$\text{Last\_Group}_i = r_i \bmod 3. \tag{1}$$

An N-bit multiplier has N rows in the first stage. The number of rows in remaining stages can be calculated by using

$$r_i = \left\lfloor \frac{2r_{i-1}}{3} \right\rfloor + r_{i-1} \bmod 3. \tag{2}$$

Reduction is performed using a full adder or a half adder depending on the number of elements in that particular column of the group. If a column has only one element then that is passed on to the next stage without any reduction. If the last group of a stage contains less than three rows then no reduction is performed on that group as shown in stage 1 of Figure 2.

The size of the final adder for an N-bit TW multiplier with S stages can be calculated by

$$\text{FinalAdder}_{\text{TW}} = (2N - 1) - S. \tag{3}$$

Table 1: Final adder size for different multipliers.

N	Final adder size				Logic levels
	TW	RCW	Dadda	PW	
8	11	14	14	10	4
16	25	30	30	24	5
24	40	46	46	39	6
32	55	62	62	54	6
64	117	126	126	116	7

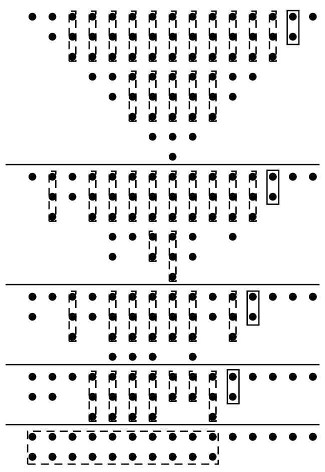

Figure 5: 8-bit proposed Wallace reduction.

2.2. Reduced Complexity Wallace (RCW) Multiplier. Waters and Swartzlander [7] presented a modification in the TW multiplier to reduce the complexity of the reduction tree. An 8-bit RCW reduction process is shown in Figure 3.

The partial products are readjusted in a reverse pyramid style which makes it easy to analyse the tree for efficient reduction. The number of stages for RCW multiplier remains the same as that of TW multiplier. RCW tries to reduce the partial product tree using only full adders. Half adders are used only where they are necessary to satisfy the number of rows in a stage according to (2). This approach allows RCW multiplier to reduce the area of the reduction process. However, RCW multiplier uses a much larger final adder as compared to TW multiplier. The size of the final adder for an N-bit RCW multiplier can be computed by

$$\text{FinalAdder}_{\text{RCW}} = 2N - 2. \quad (4)$$

2.3. Dadda Multiplier. Dadda multiplier [11] tries to reduce the number of full adders and half adders by performing the reduction only where it is essential to satisfy (2). An 8-bit Dadda reduction process is shown in Figure 4.

Dadda has the same number of stages as that of TW and RCW. We can see from Figure 4 that Dadda performed reduction only on four columns in stage 1. This is because the other columns already satisfy (2). The same approach is used in all stages to reduce the tree until we achieve a tree of only two rows. The size of the final adder is the same for Dadda and RCW multiplier as computed in (4).

3. Proposed Wallace (PW) Multiplier

In this section, we proposed a modification in the RCW multiplier to further reduce its area by reducing the size of the final adder. PW multiplier has the same number of stages and the same rule for maximum number of rows in a stage as in the other multipliers discussed in this paper.

An 8-bit PW reduction process is shown in Figure 5. PW uses an additional half adder in each stage in order to reduce the size of the final adder. The algorithm scans from the right side and starts the reduction by using a half adder when it finds the first column where the number of elements is greater than one. The additional half adders are shown in solid boxes at each stage of PW multiplier in Figure 5.

Compared with RCW in Figure 3, the introduction of half adders in Figure 5 makes the final adder in PW "less wide", namely, a smaller size. This is because, in each stage, the half adder that we introduced computes the final product bit for that particular column of the partial product tree. Therefore, the size of the required final adder is decreased by one in each stage. The least significant bit (LSB) of the product, P_0, is produced by the partial product generation block by computing $A_0 \times B_0$. In the first stage of the reduction process, product bit P_1 is computed by using the additional half adder. In the second stage, P_2 is computed. Similarly, stage 3 and stage 4 compute the product bits P_3 and P_4, respectively. Thus, when the partial product tree is reduced to two rows, five LSBs ($P_4 - P_0$) of the product are already computed as shown in Figure 5. Therefore, the size of the final adder in 8-bit PW is reduced by four as compared to the final adder in RCW multiplier. The size of the final adder for an N-bit PW multiplier with S stages can be computed by

$$\text{FinalAdder}_{\text{PW}} = (2N - 2) - S. \quad (5)$$

The comparison of (4) and (5) shows a reduction of S in the size of final adder from RCW to PW. This is achieved at the expense of an increased area for reduction process due to the insertion of additional half adders in the PW. However, the effect of additional half adders is very small as compared to the area saved by reducing the final adder size. Therefore, the overall area of the PW is less than that of the RCW.

The size of the final adder and their required logic levels for different multipliers are given in Table 1. The PW has the smallest final adder as compared to all other multipliers. All the multipliers need the same number of logic levels to implement the final adder, which means that all the multipliers will have almost the same delay. The architecture of the final adder is discussed in Section 4.

Table 2: Synthesis parameters for Synopsys Design Vision.

Technology	90 nm CMOS
Supply voltage	1.2 V
Temperature	25°C
Process model	Typical
Interconnect model	Balanced tree

4. Final Adder Design

The third step of the Wallace tree-based multipliers is to add the remaining two rows using a fast adder. Some of the most widely used parallel-prefix adders used for high speed operations are Kogge-Stone [12], Sklansky [13], and Brent-Kung [14]. These adders use the same tree topology but differ in terms of logic levels, fanout, and interconnect wires. We used Kogge-Stone adder in all the multipliers discussed in this paper. The logic levels for implementation of an N-bit Kogge-Stone adder can be calculated by using

$$\text{LogicLevels} = \lceil \log_2 (N) \rceil . \qquad (6)$$

5. Results

In this section, we will discuss the verification of designs for correct operation, synthesis tool, and the results.

5.1. Functional Verification. The multipliers are implemented in VHDL with the test programs to verify the designs. All the possible input combinations are applied to thoroughly test the 8-bit multipliers. Since an exhaustive testing of bigger multipliers was not practical, they are tested with random inputs applied. The Galois-type linear feedback shift registers (LFSRs) are designed to generate pseudorandom binary sequence (PRBS) of maximum cycle for the multipliers under test [15]. All the designs are compiled and simulated using Synopsys VCS.

5.2. Synthesis Tool. All the multipliers are synthesized in Synopsys Design Compiler (DC) using 90 nm technology. The designs can be optimized for delay, power, and area by setting the appropriate options in the DC. The designer has the option of setting the various synthesis parameters such as fanout, wire load models, interconnect strategy, and PVT (process, voltage, and temperature).

The scripts are written to synthesize the TW, RCW, Dadda, and PW multipliers for optimized area. In order to have a fair comparison, the same synthesis parameters are specified for all the designs. Table 2 shows different parameters from SAED 90 nm library used for synthesis.

5.3. Synthesis Results. The detailed synthesis reports are generated by Design Compiler for area and timing. The area report includes number of cells, the area used by cells, and the interconnect area. The timing report shows the complete critical path along with the delay associated with each cell in the path. Table 3 shows the synthesis results for delay and area for different multipliers.

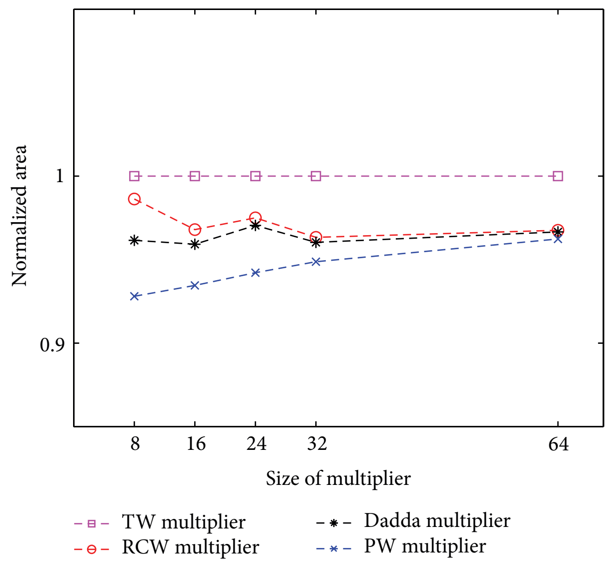

Figure 6: Area of different multipliers on Synopsys 90 nm technology.

Figure 6 shows the normalized area for each multiplier. The area of all multipliers is normalized with respect to TW multiplier by using

$$\text{Norm\_Value}_{\text{Mult\_XX}} = \frac{\text{Original\_Value}_{\text{Mult\_XX}}}{\text{Original\_Value}_{\text{TW\_Mult}}}. \qquad (7)$$

It is clear from Figure 6 that the TW has the largest area as expected. Areas of RCW and Dadda are almost the same which also conforms to the results of [7]. PW has the lowest area for all the multiplier configurations. The reduction in area is more prominent when the size of the multiplier is small. As the size of the multiplier increases, the area of PW tends to asymptotically approach the area of RCW and Dadda multiplier. Therefore, the PW is particularly useful when the final adder has a significant area in the multiplier, while the area advantage might decrease in larger multipliers.

Figure 7 shows the normalized delay for each multiplier. The delays are normalized according to (7). All the multipliers use the same number of reduction stages and the same logic levels in the final adder. Therefore their delays are expected to be the same. However, the Design Compiler uses different cells to optimize the area in each design due to their different architectures and different final adder sizes. This can result in larger delays for the designs where the synthesizer can optimize the area by using relatively slower standard cells. It can be seen in Figure 7 that the PW has the least delay in 24-bit multiplier. In the rest of the multiplier sizes, PW has almost the same delay as of RCW which is less than the Dadda and TW. One exception to this is the 16-bit multiplier where PW has larger delay than RCW multiplier.

Figure 8 shows the normalized power consumption for each multiplier. The power consumption of all multipliers is normalized with respect to TW multiplier by using (7).

Table 3: Synthesis results from Synopsys DC on 90 nm technology.

Size	Delay (ns)				Area (μm^2)				Power (mW)			
	TW	RCW	Dadda	PW	TW	RCW	Dadda	PW	TW	RCW	Dadda	PW
8	2.81	2.64	2.64	2.66	3392	3346	3262	3148	1.92	2.04	1.94	1.96
16	4.13	3.65	3.80	3.75	14847	14372	14242	13876	11.09	11.41	11.22	11.20
24	4.64	4.58	4.62	4.44	34337	33479	33323	32352	27.69	28.24	28.32	28.04
32	14.83	14.62	14.82	14.62	61526	59271	59086	58375	51.85	52.74	53.11	52.48
64	22.88	21.57	21.98	21.62	246842	238843	238597	237553	216.50	219.17	222.64	218.92

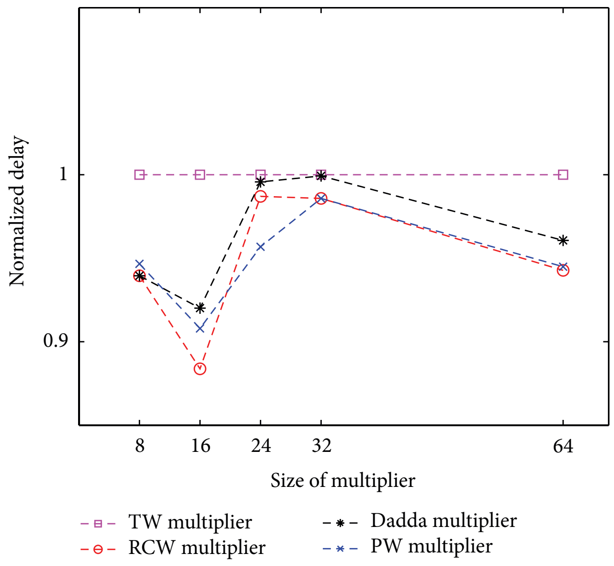

Figure 7: Delay of different multipliers on Synopsys 90 nm technology.

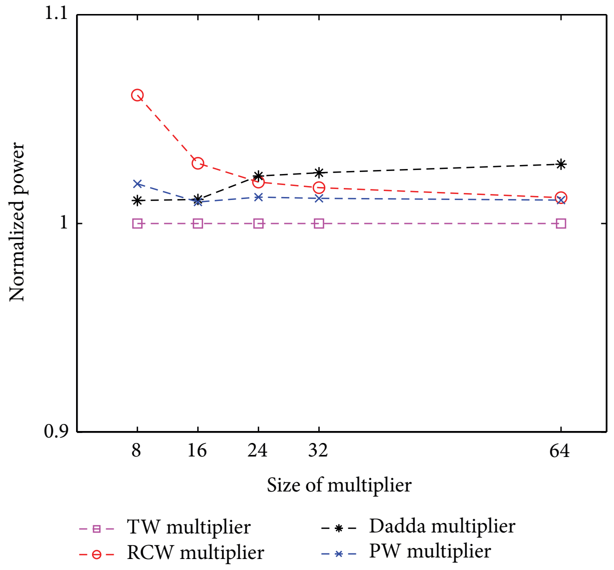

Figure 8: Power consumption of different multipliers on Synopsys 90 nm technology.

It is clear from Figure 8 that the TW multiplier has the lowest power consumption as compared to the other multipliers. One reason for this could be that the Design Compiler was able to find the low-power cells to synthesize the TW multiplier. The regular structure of TW multiplier could also be a reason of its low-power consumption. The power consumption of PW is less than that of the RCW multiplier due to the smaller final adder used in PW multiplier. It can be noted that the difference in power consumption of PW and RCW is very little for large multipliers, as expected, due to the small difference in their area.

6. Conclusion and Future Work

This paper presents a method to reduce the area of the Wallace multiplier. The proposed architecture, named as PW (proposed Wallace) multiplier, uses a smaller final adder to reduce the area of a multiplier. The designs are synthesized in Synopsys Design Compiler using 90 nm process technology. The synthesis results verify that the PW multiplier, as expected, has the smallest area as compared to the other Wallace based multipliers. The speed of the PW multiplier is almost the same as of other multipliers.

As our future work, we plan to implement the designs using Synopsys IC Compiler to analyze the postlayout results for area and delay. Synopsys Prime Time can be used to analyze the multipliers for their power consumption.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] N. H. E. Weste and D. M. Harris, *Integrated Circuit Design*, Pearson, 2010.

[2] J.-Y. Kang and J.-L. Gaudiot, "A fast and well-structured multiplier," in *Proceedings of the EUROMICRO Systems on Digital System Design (DSD '04)*, pp. 508–515, September 2004.

[3] C. R. Baugh and B. A. Wooley, "A twos complement parallel array multiplication algorithm," *IEEE Transactions on Computers*, vol. C-22, no. 12, pp. 1045–1047, 1973.

[4] S.-R. Kuang, J.-P. Wang, and C.-Y. Guo, "Modified booth multipliers with a regular partial product array," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 5, pp. 404–408, 2009.

[5] B. C. Paul, S. Fujita, and M. Okajima, "ROM-based logic (RBL) Design: a low-power 16 bit multiplier," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 11, pp. 2935–2942, 2009.

[6] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, 1964.

[7] R. S. Waters and E. E. Swartzlander, "A reduced complexity wallace multiplier reduction," *IEEE Transactions on Computers*, vol. 59, no. 8, pp. 1134–1137, 2010.

[8] S. Rajaram and K. Vanithamani, "Improvement of Wallace multipliers using parallel prefix adders," in *Proceedings of the International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN '11)*, pp. 781–784, July 2011.

[9] P. Jagadeesh, S. Ravi, and K. H. Mallikarjun, "Design of high performance 64 bit mac unit," in *Proceedings of the International Conference on Circuits, Power and Computing Technologies (ICCPCT '13)*, pp. 782–786, March 2013.

[10] M. Kumaran and M. Kamarajan, "Multicore embedded system using parallel processing technique," *International Journal of Emerging Trands in Electrical and Electronics*, vol. 5, no. 3, 2013.

[11] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[12] 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. C-22, no. 8, pp. 786–793, 1973.

[13] J. Sklansky, "Conditional-sum addition logic," *IRE Transactions on Electronic Computers*, vol. EC-9, pp. 226–231, 1960.

[14] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, 1982.

[15] R. Ward and T. Molteno, "Table of linear feedback shift registers," Datasheet, Department of Physics, University of Otago, 2007.