Performance Analysis of Wallace and Radix-4 Booth-Wallace Multipliers

Shahzad Asif, Yinan Kong

Department of Engineering, Macquarie University, NSW 2109, Australia

shahzad.asif@mq.edu.au

***Abstract*—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. In the previous years the Booth encoding is widely used in the tree multipliers to increase the speed of the multiplier. However, the efficiency of the Booth encoders decreases with the technology scale down. In this paper we showed that the use of Booth encoders in fact increases the delay and power of the Wallace multiplier in the deep submicron technology. The radix-4 Booth-Wallace and the Wallace multipliers are implemented for various sizes and synthesized using Synopsys Design Compiler in 90nm process technology. The synthesis results show that the Wallace multiplier has up to 17% less delay and 70% less power consumption as compared to the radix-4 Booth-Wallace multipliers. The Power-Delay Product (PDP) of the Wallace multiplier is up to 68% lower than the Booth-Wallace multiplier.**

***Index Terms*—Wallace multiplier, Booth encoder, Booth-Wallace, high speed, low power, radix-4**

I. Introduction

The high speed arithmetic circuits are desirable for all the computing applications specifically in Digital Signal Processing (DSP) algorithms. Multiplication is one of the most extensively used operation in the DSP algorithms, therefore speed optimization of the multipliers is critical in order to speed up the DSP algorithms.

A large number of high speed multiplier architectures are proposed in the literature among which the tree multipliers are considered to be one of the fastest. Fig. 1 shows the general block diagram of the tree based multipliers.

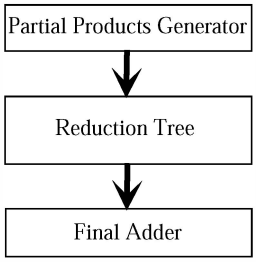


Fig. 1: Block Diagram of Tree Based Multipliers

As can be seen in the Fig. 1, the operation of the tree multipliers is divided in three steps.

1) **Partial product generation:** The simplest circuit for generating the partial product tree consists of AND gates. An N-bit multiplier requires N^2 AND gates to generate the partial product tree of N rows. Another method to generate partial product tree is by Booth encoders [1] which will be discussed in detail in Section III.
2) **Reduction Tree:** In this step, Full Adders (FAs) and Half Adders (HAs) are used to add all the columns of the partial product tree in parallel fashion. The partial product tree is reduced stage by stage until only two rows are left.
3) **Final Adder:** In this step, the remaining two rows of the partial product tree are added by a fast adder to produce the final product.

The tree multipliers were first proposed by Wallace in 1964 [2] in which the partial product tree is divided into groups where each group consists of three rows. Then the addition is performed in every column using full adders and half adders. This process is repeated until the tree is reduced to two rows. Another tree based multiplier was proposed by Dadda in 1965 [3] and hence named as Dadda multiplier. The Dadda multiplier also uses full adders and half adders to reduce the partial product tree. However, unlike Wallace, the Dadda multiplier optimises the hardware by performing minimum reduction in a stage.

Over the past few decades, a number of researchers used Booth encoders to reduce the delay of the tree multipliers due to their smaller size of partial product tree. Some of the Booth encoder based high speed multipliers are reported in [4]–[9]. A detailed analysis of high radix Booth multipliers is presented in [10] which proves that the radix-4 Booth encoder has higher performance as compared to the radix-8, radix-16, and radix-32 Booth multipliers. However the author in [10] did not compare the performance of Wallace multiplier with the Booth multiplier.

It is however not clear that the use of Booth encoders always result in a reduced delay of the multiplier. Specifically in the deep submicron CMOS technology the use of Booth encoders might actually increases the delay of the multiplier due to their complex encoding circuity. It is shown in the [11] that the performance of the Booth multipliers is lower as compared to the Bough-Wooley multipliers [12]. The results in [13]

also shows that the Wallace multiplier has slightly less delay than the radix-4 Booth encoder for the 32-bit multiplier. In order to prove that the Wallace multiplier performs better as compared to the Booth-Wallace a more thorough analysis is required. We present the detailed analysis of Booth-Wallace and Wallace multipliers of different sizes (8-bit to 256-bit) to show that the performance of the Booth-Wallace multipliers is much lower than the Wallace multipliers for all multiplier sizes.

The rest of this paper is organized as follows. In Section II the design of a modified Wallace multiplier is presented. Section III describes the design of the radix-4 Booth Wallace multiplier. The choice of the final adder is discussed in the Section IV. Section V compares the performance of the Wallace and Booth-Wallace multipliers. The work is concluded in the Section VI.

II. Reduced Complexity Wallace (RCW) Multiplier

In this section we will discuss the architecture of the Reduced Complexity Wallace (RCW) which was presented in [14] in one of the high quality transactions. RCW is an area improved architecture of the Wallace multiplier which has the same speed as of traditional Wallace due to the same reduction stages. The dot diagram of a 16× 16 RCW multiplier is shown in Fig. 2.

In the Fig. 2 the reduction stages are separated by thick horizontal lines. The Full Adders (FAs) and Half Adders (HAs) in each stage are represented by the boxes around the dot products. The right most column is called column 0. The partial product tree is readjusted in the form of a reverse pyramid which makes it easier to analyse the tree for reduction. The RCW reduces the area of the multiplier by removing unnecessary half adders in the reduction as shown in the Fig. 2. The 16×16 RCW uses only 9 half adders which are much less than the traditional Wallace which uses 52 half adders [14].

The maximum rows in a stage of RCW multiplier can be computed by Equation (1).

$$R_i = 2 \times \left\lfloor \frac{R_{i-1}}{3} \right\rfloor + R_{i-1} \bmod 3 \tag{1}$$

The total stages for an $N \times N$ RCW multiplier can be calculated by Algorithm 1.

Algorithm 1 Stages for $N \times N$ RCW Multiplier

Require: $Stages \leftarrow 0, rows \leftarrow N$
  while $rows > 2$ **do**
    $Stages \leftarrow Stages + 1$
    $rows \leftarrow 2 \times \lfloor rows/3 \rfloor + rows \bmod 3$
  end while

III. Radix-4 Booth-RCW Multiplier

Booth encoding is widely used in multipliers for generating the partial product tree due to its smaller tree size. Booth encoders are categorized in radices with respect to the number of bits used to encode the data. The Booth encoder originally proposed by Booth [1] uses two bits to encode and thus called radix-2 Booth encoder. A number of variants have been proposed in the literature to modify the Booth encoder to suit different applications. The work in [10] present a detailed analysis of radix-4, radix-8, radix-16, and radix-32 Booth multipliers. The radix-4 Booth encoder is the most efficient encoder as compared to the other high radix Booth encoders [10] due to its simple structure. The block diagram of the radix-4 Booth encoder is shown in the Fig. 3.

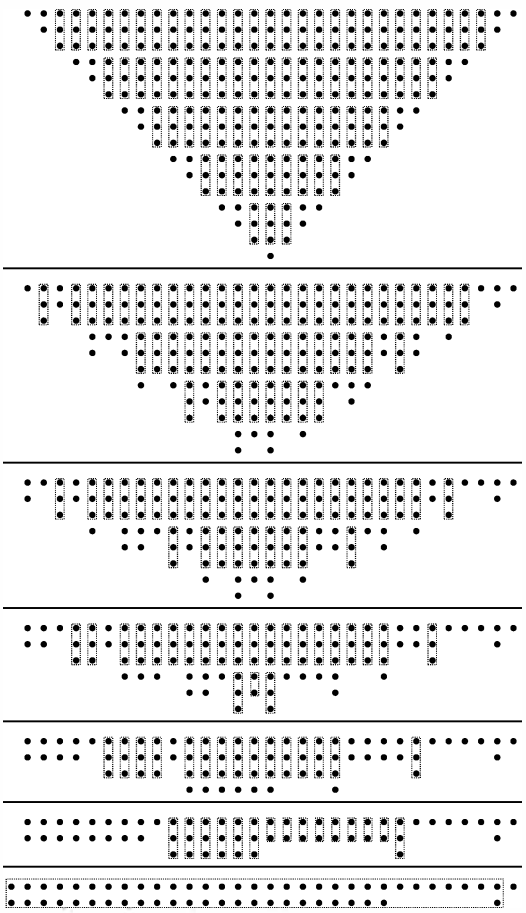


Fig. 2: Dot Diagram of 16×16 RCW Multiplier

The radix-4 Booth encoder uses the 3-bit combinations of the multiplier to perform the encoding [15]. In order to use the Booth encoding for the unsigned multiplier an extra sign bit is need to be calculated in the partial product generation. The Table I shows the encoding scheme used by the radix-4 Booth encoder.

It can be seen from the Table I that even though the multiplier is unsigned the partial products can be negative. The negative partial products require inversion and addition of '1' at Least Significant Bit (LSB) - known as the LSB insertion - which would result in an irregular layout of the partial product tree. In order to have a regular layout of the partial product tree we used the idea presented in [11] which is a modified form of the [5]. According to this scheme the

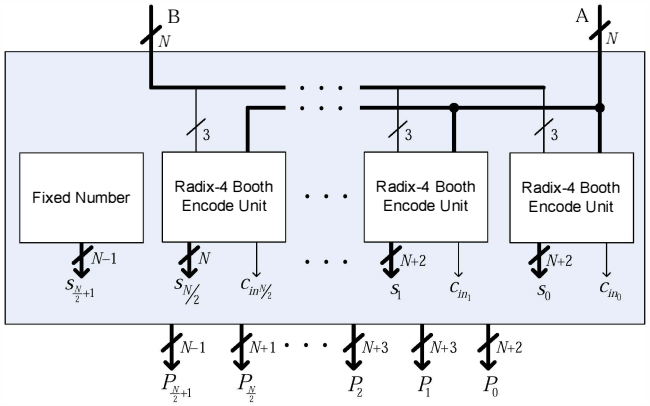


Fig. 3: Block diagram of radix-4 Booth encoding

TABLE I: Radix-4 Booth Encoding

En_{2i+1}	En_{2i}	En_{2i-1}	*Partial product*	*Sign bit*
0	0	0	0	0
0	0	1	Y	0
0	1	0	Y	0
0	1	1	2Y	0
1	0	0	−2Y	1
1	0	1	−Y	1
1	1	0	−Y	1
1	1	1	−0=0	0

impact of LSB insertion on the least significant bit position of the partial product tree is pre-computed (Eq. 2) and the potential '1' is shifted to the second least significant position (Eq. 3). Furthermore, the negative partial products need to be sign-extended in the Booth encoding. This sign-extension of the negative partial products is avoided by using the scheme proposed by Fadavi-Ardekani [4].

$$P_{LSB} = A_0.(En_1 \oplus En_0) \tag{2}$$

$$C_{in} = En_2.\left(\overline{En_1 + En_0} + \overline{A_0 + En_1} + \overline{A_0 + En_0}\right) \tag{3}$$

The partial product tree for the radix-4 Booth encoder is shown in the Fig. 4. Note that the partial product in Fig. 4 is slightly different from the [11] because the design in [11] is for a signed multiplier.

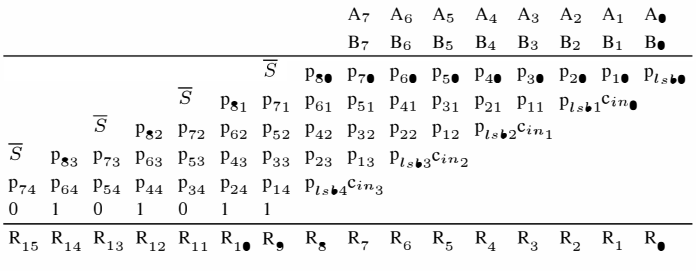


Fig. 4. Illustration of an 8-bit radix-4 Booth encoder

The partial product tree generated by radix-4 Booth encoder consists of $\frac{N}{2} + 2$ rows. The partial product tree is readjusted in the form of reverse pyramid and RCW multiplier is used to perform the reduction until we are left with only two rows.

IV. 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 [16], Sklansky [17], Brent-Kung [18], [19], [20], and [21]. These adders use same tree topology but differ in terms of logic levels, fanout, and interconnect wires. The thorough analysis of the parallel-prefix adders can be found in [22]. 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 Equation (4).

$$LogicLevels = \lceil log_2(N) \rceil \tag{4}$$

V. Performance Comparison

A. Implementation Platform and Synthesis Setup

The choice of implementation platform and synthesis tool plays a major role in the performance of a design. The VHDL designs can be synthesized for implementation on FPGA (Field Programmable Gated Array) or ASIC (Application Specific Integrated Circuit). In FPGAs, the designs are implemented in the form of LUTs (Look-Up Tables), whereas ASICs allow the standard cells based flow for implementation of designs. The standard cells based implementation provides better performance in terms of area, delay, and power consumption. The FPGA is more suitable for synthesis of behavioural VHDL code whereas ASIC is more suitable for the designs which uses basic gates in the VHDL code therefore we chose standard cells for the implementation of all the designs.

All the multipliers are synthesized in Synopsys Design Compiler (DC) using 90nm technology. The designs can be optimized for delay, power, and area by setting the appropriate options in the Design Compiler. The designer has the option of setting the various synthesis parameters such as fanout, wire load models, interconnect strategy, and PVT (Process, Voltage, Temperature), etc. The synthesis process is divided in three steps: 1) Analyze: Design Compiler reads all the VHDL files in the design and analyze them for syntax errors. 2) Elaborate: The top level design is converted into technology independent RTL model. 3) Compile: In this step, the RTL model of the design is synthesized into standard cells provided by the design library. The designs are optimized according to the strategy and constraints set by the designer.

The scripts are written to synthesize the RCW and Booth-RCW multipliers for minimum delay. In order to have a fair comparison, the same synthesis parameters are specified for all the designs. Table II shows different parameters from SAED 90 nm library used for synthesis.

The Design Compiler generates the reports for delay, power, and area of the designs. However the power reports generated by Design Compiler are not accurate because it is calculated

TABLE II: Synthesis Parameters for Synopsys DC

Technology	90 nm CMOS
Libraries used	saed90nm_typ_lvt saed90nm_typ_hvt
Supply Voltage	1.2 V
Temperature	$25°C$
Output load	1.5 pF
Interconnect Model	Balanced-Tree
Compile effort	Medium
Wire load model	Automatic

without any information of the switching activity of the circuit. In order to obtain more accurate power estimation the designs are simulated in Modelsim SE with their maximum frequency and switching activity is stored in the Value Change Dump (VCD) file. The VCD file is then used with the Synopsys Prime Time (PT) to obtain the power dissipation of the designs.

B. Synthesis Results

This section presents and discusses the synthesis results of the RCW and Booth-RCW multipliers. The number of reduction stages for the multipliers are given in the Table III.

TABLE III: Reduction Stages

	Tree Reduction Stages	
Size	**RCW**	**Radix-4 Booth-RCW**
8	4	3
16	6	5
32	8	6
64	10	8
128	11	10
256	13	11

As can be seen in the Table III that the difference in reduction stages of RCW and Booth-RCW is very little (up to 2) which suggests that the speed improvement in Booth-RCW due to smaller reduction tree would be less than expected. This small decrease in delay is expected to be cancelled out due to the large delay of encoding circuitry in the Booth-RCW multiplier. Hence the Booth-RCW is expected to have more delay as compared to the RCW.

The delay and power estimates are obtained from Design Compiler reports as well as from Prime Time to analyse the accuracy of the two tools. The delay and power results are reported in the Table. IV.

The Fig. 4 shows the comparison of RCW and Booth-RCW multipliers. The values in the graphs are normalized with respect to RCW by using the Equation (5).

$$Norm\_Value_{Mult\_X} = \frac{Original\_Value_{Mult\_X}}{Original\_Value_{RCW}} \quad (5)$$

It can be seen from the Fig. 4a and Fig. 4b that the Booth-RCW multiplier has larger delay as compared to the RCW

TABLE IV: Synthesis Results of RCW and Booth-RCW Multipliers from Synopsys tools on 90 nm Technology

	Results from Synopsys DC				**Results from Synopsys PT**			
	Delay (ns)		**Power** (mW)		**Delay** (ns)		**Power** (nW)	
Size	**RCW**	**R4B RCW**	**RCW**	**R4B RCW**	**RCW**	**R4B RCW**	**RCW**	**R4B RCW**
8	1.87	1.96	0.2	0.3	1.85	1.93	0.3	1.8
16	2.45	2.65	1.3	1.5	2.44	2.63	9.8	11.3
32	7.9	6.91	6.8	7.4	7.73	7.12	9.8	17.1
64	13.28	12.33	26.1	27.1	13.11	12.16	5.2	17.5
128	22.28	26.67	100.7	95.9	22.4	26.73	8.4	14.9
256	31.37	37.69	387.1	371.4	31.37	37.8	15.1	28.3

* R4B refers to Radix-4 Booth Encoding

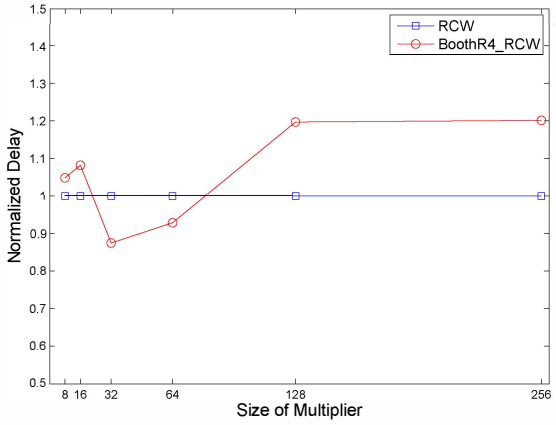
with the exception of 32-bit and 64-bit multiplier where the delay of Booth-RCW is slightly less than the RCW. The delay of Booth-RCW is about 17% more for the 128-bit and 256-bit multiplier as compared to the RCW. The reason of this is the large delay of partial product generator circuitry in the Booth-RCW multiplier as compared to the RCW. This confirms that the use of Booth encoding in Wallace multiplier can reduce the speed of the multiplication.

The Fig. 4c and Fig. 4d shows the normalized power consumption of the two designs obtained from the Design Compiler and Prime Time, respectively. There is a large difference in the power estimates calculated by DC and PT (See Table IV for actual values). Since the PT calculates the power consumption based on the switching activity therefore it provides more accurate results (Fig. 4d). The power consumption of the Booth-RCW is up to 70% higher as compared to the RCW. Therefore Booth multiplier is not suitable for low power designs.

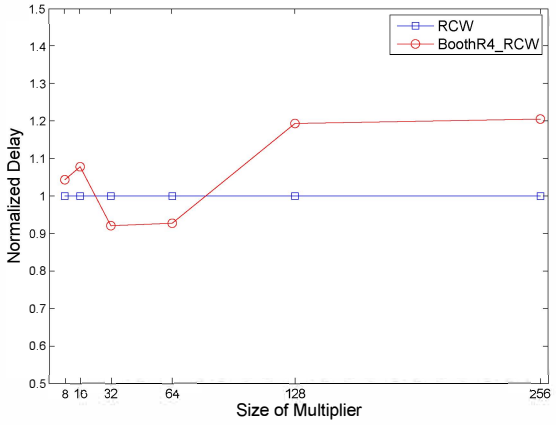
Another commonly used performance parameter is Power-Delay Product (PDP) which shows the energy required for one operation. The normalized PDP of the Booth-RCW and RCW is shown in the Fig. 5a. The PDP of the RCW is up to 68% lower than the Booth-RCW. Based on these results we can safely state that the use of Booth encoding results in a decrease in performance of the Wallace multiplier in deep submicron technologies. The area of the Booth-RCW is slightly lower for large multipliers as compared to the RCW as shown in the Fig. 5b

VI. Conclusion

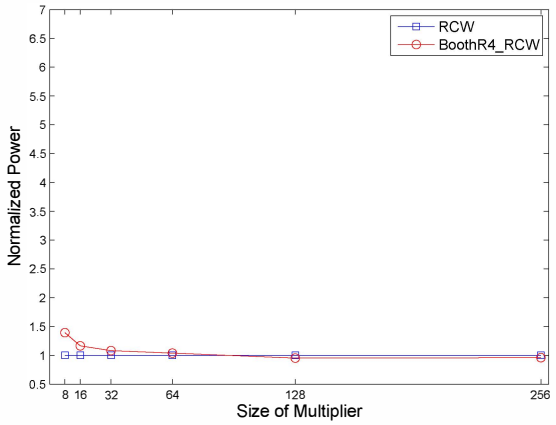
In this work, we compared the performance of Wallace and radix-4 Booth-Wallace multipliers. The work was motivated by the ever increasing use of Booth encoding in the literature to reduce the delay of the multiplier. The high speed architecture of radix-4 Booth-Wallace multiplier was implemented to be compared with the Wallace multiplier. In order to perform a detailed analysis we implemented multipliers of different sizes (8-bit to 256-bit) using the both architectures. The designs are synthesized using Synopsys Design Compiler in 90nm CMOS technology. The results for delay and power are obtained using the Synopsys Prime Time which shows that the RCW



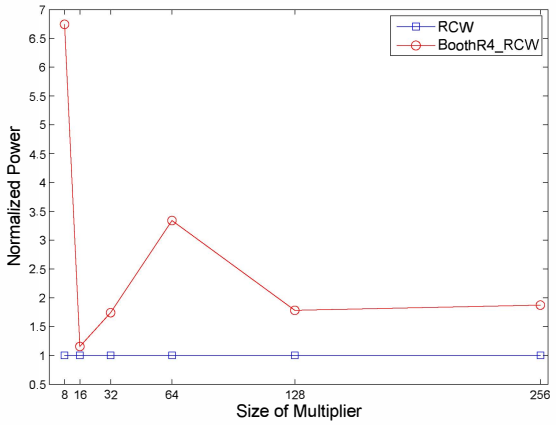
(a) Delay results from Synopsys Design Compiler



(b) Delay results from Synopsys Prime Time

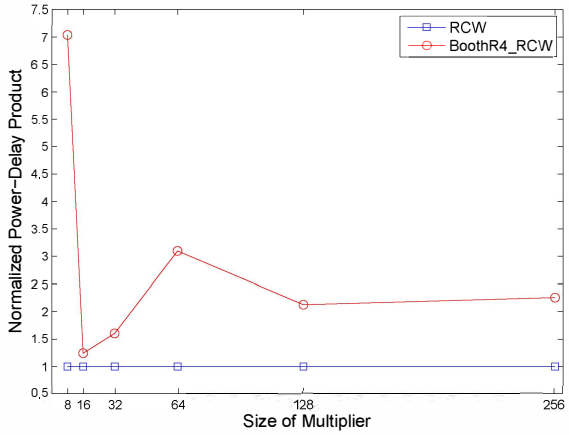


(c) Power results from Synopsys Design Compiler

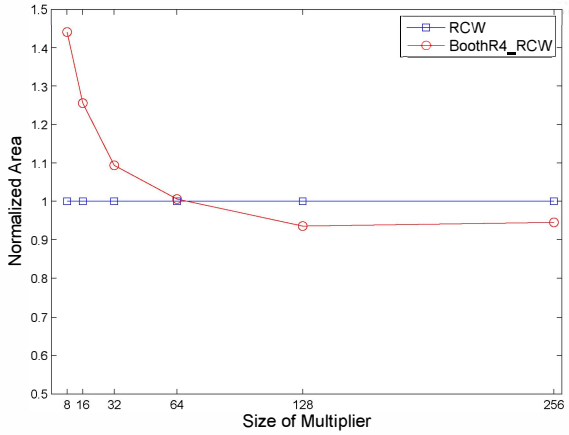


(d) Power results from Synopsys Prime Time

Fig. 4: Comparison of delay and power of RCW and Booth-RCW Multiplier from Synopsys tools on 90 nm Technology



(a) PDP results from Synopsys Prime Time



(b) Area results from Synopsys Design Compiler

Fig. 5: Comparison of Area and PDP of RCW and Booth-RCW from Synopsys tools on 90 nm Technology

performs significantly better in terms of both delay and power as compared to the Booth-RCW. The RCW has up to 17% lower delay and 70% lower power as compared to the Booth-RCW. The Power-Delay Product (PDP) of the RCW is up to 68% lower than the Booth-RCW. Based on these results we can conclude that the use of Booth encoding reduces the performance of any tree based multiplier.

We have also compared the consistency between the delay and power results obtained from the Design Compiler and Prime Time. The delay estimates are similar in both the tools but the power results of Design Compiler have large deviation from the more accurate results generated by Prime Time. This is because the Design Compiler generates the power report in the absence of any switching information of the circuit.

REFERENCES

[1] A. D. Booth, "A signed binary multiplication technique," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236–240, 1951. [Online]. Available: http://qjmam.oxfordjournals.org/content/4/2/236.abstract

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

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

[4] J. Fadavi-Ardekani, "M*n 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.

[5] 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, 2000.

[6] R. Ismail and R. Hussin, "High performance complex number multiplier using booth-wallace algorithm," in *Semiconductor Electronics, 2006. ICSE '06. IEEE International Conference on*, Oct 2006, pp. 786–790.

[7] M. Bansal, S. Nakhate, and A. Somkuwar, "High performance pipelined signed 64x64-bit multiplier using radix-32 modified booth algorithm and wallace structure," in *Computational Intelligence and Communication Networks (CICN), 2011 International Conference on*, Oct 2011, pp. 411–415.

[8] M. Rao and S. Dubey, "A high speed and area efficient booth recoded wallace tree multiplier for fast arithmetic circuits," in *Microelectronics and Electronics (PrimeAsia), 2012 Asia Pacific Conference on Postgraduate Research in*, Dec 2012, pp. 220–223.

[9] R. Kshirsagar, E. Aishwarya, A. Vishwanath, and P. Jayakrishnan, "Implementation of pipelined booth encoded wallace tree multiplier architecture," in *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on*, Dec 2013, pp. 199–204.

[10] K. Swee and L. H. Hiung, "Performance comparison review of 32-bit multiplier designs," in *Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on*, vol. 2, June 2012, pp. 836–841.

[11] M. Sjalander and P. Larsson-Edefors, "High-speed and low-power multipliers using the baugh-wooley algorithm and hpm reduction tree," in *15th IEEE Int. Conf. Electronics, Circuits and Systems, ICECS*, St. Julien's, Aug 2008, pp. 33–36.

[12] 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, December 1973.

[13] K. Swee and L. H. Hiung, "Performance comparison review of 32-bit multiplier designs," in *Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on*, vol. 2, June 2012, pp. 836–841.

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

[15] O. L. Macsorley, "High-speed arithmetic in binary computers," *Proceedings of the IRE*, vol. 49, no. 1, pp. 67–91, 1961.

[16] 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, August 1973.

[17] J. Sklansky, "Conditional-sum addition logic," *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, 1960.

[18] 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, March 1982.

[19] T. Han and D. A. Carlson, "Fast area-efficient vlsi adders," in *IEEE 8th Symposium on Computer Arithmetic*, May 1987, pp. 49–56.

[20] S. Knowles, "A family of adders," in *Proc. 15th IEEE Symposium on Computer Arithmetic*, Adelaide, SA, April 2001, pp. 277–281.

[21] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *Journal of the AssoclaUon for Computing Machinery*, vol. 27, no. 4, pp. 831–838, October 1980.

[22] D. Harris, "A taxonomy of parallel prefix networks," in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2, November 2003, pp. 2213–2217.