Radix-16 Booth multiplier using novel weighted 2-stage Booth algorithm

Hyunpil Kim[1], **Sangook Moon**[2], **and Yongsurk Lee**[1a)]

[1] *School of Electrical & Electronic Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, Korea*

[2] *Department of Electronic Engineering, Mokwon University, 88 Doanbuk-ro, Seo-gu, Daejeon 302–729, Republic of Korea*

a) *yonglee@yonsei.ac.kr*

Abstract: In this study, we propose a radix-16 Booth multiplier using a novel weighted 2-stage Booth algorithm. Most conventional multipliers utilize radix-4 Booth encoding because a higher radix increases encoder complexity. To resolve this problem, we propose the weighted 2-stage Booth algorithm. The synthesis results show that the multiplier using the proposed algorithm achieves better power-delay products than those achieved by conventional Booth multipliers. We believe that the proposed Booth algorithm can be broadly utilized in general processors as well as digital signal processors, mobile application processors, and various arithmetic units that use Booth encoding.

Keywords: Booth algorithm, multiplier, weighted 2-stage Booth algorithm, Booth encoding

Classification: Integrated circuits

References

[1] K. L. S. Swee and H. Lo Hai: ICIAS (2012) 854. DOI:10.1109/ICIAS.2012.6306134

[2] R. Riedlinger, R. Arnold, L. Biro, B. Bowhill, J. Crop, K. Duda, E. S. Fetzer, O. Franza, T. Grutkowski, C. Little, C. Morganti, G. Moyer, A. Munch, M. Nagarajan, C. Parks, C. Poirier, B. Repasky, E. Royman, T. Singh and M. W. Stefaniw: IEEE J. Solid-State Circuits **47** (2012) 177. DOI:10.1109/JSSC.2011.2167809

[3] L. Li, J. Hu and Y. Chen: IEICE Electron. Express **9** (2012) 352. DOI:10.1587/elex.9.352

[4] R. Muralidharan and C. Chip-Hong: IEEE Trans. Circuits Syst. I **60** (2013) 2940. DOI:10.1109/TCSI.2013.2252642

[5] C. Chip-Hong, G. Jiangmin and Z. Mingyan: IEEE Trans. Circuits Syst. I **51** (2004) 1985. DOI:10.1109/TCSI.2004.835683

[6] D. R. Gandhi and N. N. Shah: ISSP (2013) 1. DOI:10.1109/ISSP.2013.6526864

[7] Y. Wen-Chang and J. Chein-Wei: IEEE Trans. Comput. **49** (2000) 692. DOI:10.1109/12.863039

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

1 Introduction

A multiplier is one of the most commonly used arithmetic units, and until recently, its performance has been the most significant feature considered during design. However, with the ever-increasing demand for low-power consumption of processors used in servers and mobile equipment, power consumption is gradually becoming an important design feature. Despite such changes in the requirement, most conventional multipliers still utilize the radix-4 Booth algorithm for high-performance processors and arithmetic units.

According to Swee and Hai, the radix-4 Booth multiplier offers the best performance and area features [1]. However, the advantage of a higher radix Booth multiplier is that the number of partial products is reduced. The small number of partial products is especially favorable when implementing a pipeline multiplier. Another study [2] proposes an Itanium processor that uses four clock cycles for the pipeline multiplier. In this case, large pipeline registers are required because the Wallace (or carry-save adder) tree is divided by the pipeline registers.

Although the higher radix multiplier has the advantages, there are two main problems related to its implementation. First, extra adder logics are required for a higher radix unlike the radix 2 and the radix 4. Second, larger input multiplexers (MUXs) are required in the higher radix multiplier. These problems result in low performance and high consumption of resources and power.

Most multiplier-related research has focused on other areas such as modulo multipliers with a residue number system for encryption at the architectural level [3, 4] and compressors for optimization of the Wallace tree at the circuit level [5, 6]. Therefore, in this study, we propose a novel Booth encoding algorithm to resolve the two key problems of the higher radix multiplier.

2 Conventional Booth encoding

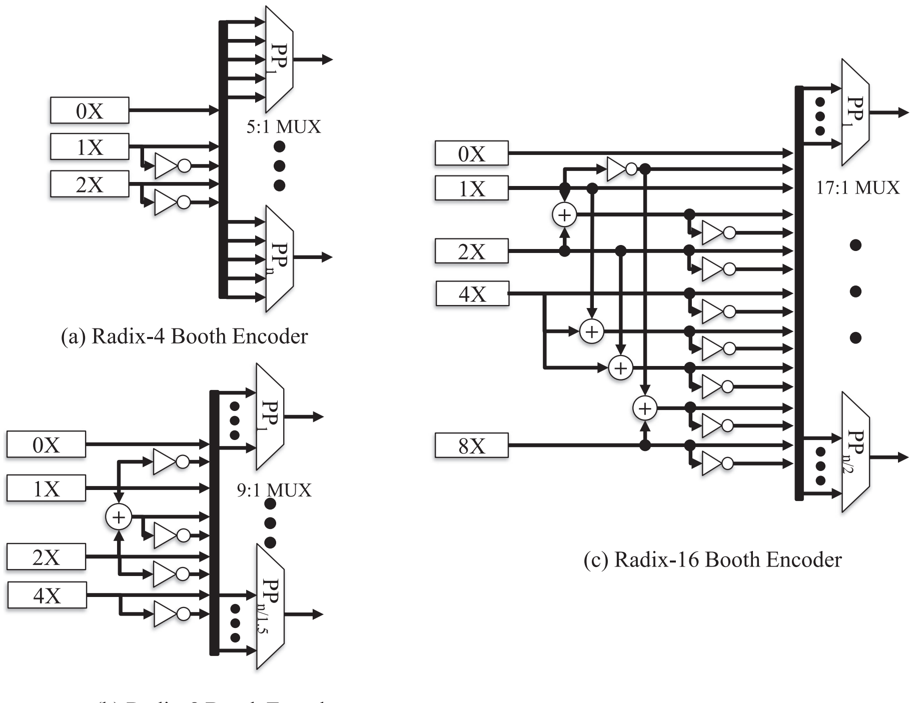
A Booth multiplier consists of a Booth encoder, carry-save adder (CSA) tree to add partial products at a time, and final adder for the results of the CSA tree. A most significant feature of the Booth multiplier is that the number of partial products is proportional to the radix N of Booth encoding by $\log_2$ (N). It is an advantage that the number of partial products is halved when the increase in the radix is four-fold. In the 32-bit multiplier, for example, the number of partial products is 16 in the case of radix 4, but 8 in the case of radix 16. However, a higher radix Booth multiplier has the disadvantage of performance reduction. Besides, both area and power consumption increase because of the overhead in the higher radix Booth encoder.

Fig. 1 shows a conventional radix-4 (a), radix-8 (b), and radix-16 Booth encoder (c). As shown Fig. 1, unlike the radix-4 Booth encoders, the higher radix Booth encoders require additional adders. For example, adders are required for 3X, 5X, 6X, and 7X in radix-16 Booth encoding, where X is

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

the multiplicand. In addition, higher radix Booth encoders require larger input MUXs. For example, 5-to-1 MUXs are used in radix-4 Booth encoders, but 9-to-1 MUXs and 17-to-1 MUXs are used, respectively, in radix-8 and radix-16 Booth encoders.

Of late, most Booth multipliers use radix-4 Booth encoding because the overhead due to the higher radix in the encoder is recognized as a more important problem than the increase in the number of partial products.



(b) Radix-8 Booth Encoder

Fig. 1. Schematics of conventional Booth encoders of (a) radix-4, (b) radix-8, and (c) radix-16 multipliers.

Multipliers use the Wallace tree for fast summation of the partial products. Most high-performance processors adopt a super-pipeline technique to increase throughput. The multipliers used in high-performance processors are also pipelined, and currently, pipeline registers are inserted at the middle points of the Wallace tree. The conventional radix-4 multiplier is difficult to manage in a pipeline structure because of the greater number of partial products compared to those of a higher radix multiplier with large word size.

3 Weighted 2-stage Booth algorithm

We propose that using a higher radix to reduce the number of partial products is the most beneficial strategy, and that a new Booth encoding algorithm should be developed to reduce the burden of the encoder due to the higher radix. Here, we propose the weighted 2-stage Booth algorithm.

First, the partial products of a given radix can be defined by the following operation:

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

$$
\begin{aligned}
&[[x_{i+k+n+1}, x_{i+k+n}, \ldots, x_{i+k-1}]] \\
&\quad = -2^{k+n+1}x_{i+k+n+1} + 2^{k+n}x_{i+k+n} + \cdots + 2^{k}x_{i+k} + 2^{k}x_{i+k-1}, \qquad (1)
\end{aligned}
$$

where n represents a 2^{n+1} base of radix, k indicates the kth encoding bit, and x_i is binary. For example, the radix-4, radix-8, and radix-16 partial products are derived by Eq. (2), (3), and (4), respectively, as follows:

$$[[x_{i+1}, x_i, x_{i-1}]] = -2^1x_{i+1} + 2^0x_i + 2^0x_{i-1}, \qquad (2)$$

$$[[x_{i+2}, x_{i+1}, x_i, x_{i-1}]] = -2^2x_{i+2} + 2^1x_{i+1} + 2^0x_i + 2^0x_{i-1}, \qquad (3)$$

$$[[x_{i+3}, x_{i+2}, x_{i+1}, x_i, x_{i-1}]] = -2^3x_{i+3} + 2^2x_{i+2} + 2^1x_{i+1} + 2^0x_i + 2^0x_{i-1}. \qquad (4)$$

One of the most significant characteristics of this operation is that the law of separation is adopted. For example, the radix-16 partial products are derived by a summation of the 2-stage radix-4 operations, that is,

$$
\begin{aligned}
&[[x_{i+3}, x_{i+2}, x_{i+1}]] + [[x_{i+1}, x_i, x_{i-1}]] \\
&\quad = -2^3x_{i+3} + 2^2x_{i+2} + 2^2x_{i+1} + (-2^1x_{i+1} + 2^0x_i + 2^0x_{i-1}) \\
&\quad = -2^3x_{i+3} + 2^2x_{i+2} + 2^1x_{i+1} + 2^0x_i + 2^0x_{i-1}, \qquad (5) \\
&[[x_{i+3}, x_{i+2}, x_{i+1}, x_i, x_{i-1}]] = [[x_{i+3}, x_{i+2}, x_{i+1}]] + [[x_{i+1}, x_i, x_{i-1}]]. \qquad (6)
\end{aligned}
$$

The operation described by Eq. (6) is presented in Table I.

As shown in Table I, the three least significant bits (x_{i+1}, x_i, x_{i-1}) are calculated in the same manner as in conventional radix-4 Booth encoding, whereas the three most significant bits (x_{i+3}, x_{i+2}, x_{i+1}) are calculated as radix-4 Booth encoding weighted four times.

The proof of the general formula for a higher radix Eq. (7) is derived as follows (where n is odd if $s = \frac{n+1}{2}$, and even if $s = \frac{n}{2}$):

$$
\begin{aligned}
&[[x_{i+n-1}, x_{i+n-2}, \ldots, x_{i+s+1}, x_{i+s}]] + [[x_{i+s}, \ldots, x_i, x_{i-1}]] \\
&\quad = -2^{n-1}x_{i+n-1} + 2^{n-2}x_{i+n-2} + \cdots + 2^{s+1}x_{i+s+1} + 2^{s}x_{i+s} \\
&\qquad + \{-2^{s}x_{i+s} + \cdots + 2^0x_i + 2^0x_{i-1}\} \\
&\quad = -2^{n-1}x_{i+n-1} + 2^{n-2}x_{i+n-2} + \cdots + 2^0x_i + 2^0x_{i-1} \\
&\quad = [[x_{i+k+n+1}, x_{i+k+n}, \ldots, x_{i+k-1}]]. \qquad (7)
\end{aligned}
$$

4 Methodology

The proposed weighted 2-stage Booth algorithm can be simply designed as shown in Fig. 2. The Booth encoder using the proposed algorithm is expected to be smaller and faster than a general radix-16 Booth encoder because the large input MUX used for the general radix-16 Booth encoder are substituted by two simpler and smaller input MUXs in parallel. For example, for implementing a 32-bit multiplier, a 17-to-1 MUX is used in the conventional Booth encoder whereas two 5-to-1 MUXs are used in the proposed encoder.

We designed both conventional Booth multipliers and the proposed multiplier using Verilog HDL and verified the functionality for all the cases. The conventional Booth multiplier was designed in accordance with the study

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

Table I. Weighted 2-stage Booth encoding truth table for radix 16

X_{i+3}	X_{i+2}	X_{i+1}	X_i	X_{i-1}	PP	X_{i+1}	X_i	X_{i-1}	Base PP	X_{i+3}	X_{i+2}	X_{i+1}	Weighted PP	2-stage PP (B.PP + W.PP)
0	0	0	0	0	0Y	0	0	0	0F	0	0	0	0F	0F
0	0	0	0	1	1Y	0	0	1	1F					1F
0	0	0	1	0	1Y	0	1	0	1F					1F
0	0	0	1	1	2Y	0	1	1	2F					2F
0	0	1	0	0	2Y	1	0	0	-2F	0	0	1	4F	2F
0	0	1	0	1	3Y	1	0	1	-1F					3F
0	0	1	1	0	3Y	1	1	0	-1F					3F
0	0	1	1	1	4Y	1	1	1	0F					4F
0	1	0	0	0	4Y	0	0	0	0F	0	1	0	4F	4F
0	1	0	0	1	5Y	0	0	1	1F					5F
0	1	0	1	0	5Y	0	1	0	1F					5F
0	1	0	1	1	6Y	0	1	1	2F					6F
0	1	1	0	0	6Y	1	0	0	-2F	0	1	1	8F	6F
0	1	1	0	1	7Y	1	0	1	-1F					7F
0	1	1	1	0	7Y	1	1	0	-1F					7F
0	1	1	1	1	8Y	1	1	1	0F					8F
1	0	0	0	0	-8Y	0	0	0	0F	1	0	0	-8F	-8F
1	0	0	0	1	-7Y	0	0	1	1F					-7F
1	0	0	1	0	-7Y	0	1	0	1F					-7F
1	0	0	1	1	-6Y	0	1	1	2F					-6F
1	0	1	0	0	-6Y	1	0	0	-2F	1	0	1	-4F	-6F
1	0	1	0	1	-5Y	1	0	1	-1F					-5F
1	0	1	1	0	-5Y	1	1	0	-1F					-5F
1	0	1	1	1	-4Y	1	1	1	0F					-4F
1	1	0	0	0	-4Y	0	0	0	0F	1	1	0	-4F	-4F
1	1	0	0	1	-3Y	0	0	1	1F					-3F
1	1	0	1	0	-3Y	0	1	0	1F					-3F
1	1	0	1	1	-2Y	0	1	1	2F					-2F
1	1	1	0	0	-2Y	1	0	0	-2F	1	1	1	0F	-2F
1	1	1	0	1	-1Y	1	0	1	-1F					-1F
1	1	1	1	0	-1Y	1	1	0	-1F					-1F
1	1	1	1	1	0Y	1	1	1	0F					0F

performed by Wen-Chang and Chein-Wei [7]. Each multiplier was built and optimized by Synopsys Design Compiler using CMOS libraries with constraints of maximum frequency in 32-nm processes by the Synopsys Armenia Educational Department.

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

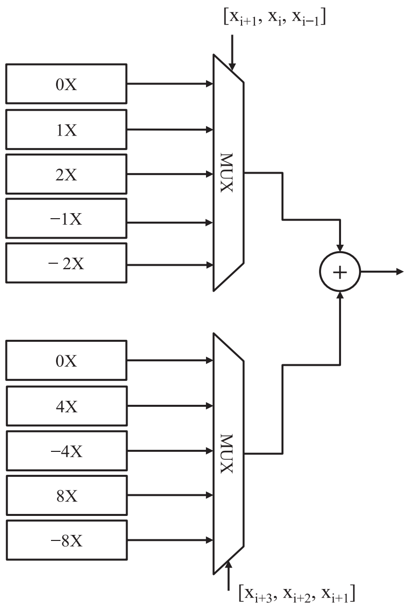


Fig. 2. Schematic of the proposed weighted 2-stage Booth encoder for radix 16.

5 Results

Fig. 3 depicts the power-delay (PD) products, which are normalized to the radix-4 value in 16-bit, 32-bit, and 64-bit word size to make it easy to compare the features of the multipliers at a glance. The PD product is a figure of merit related to the energy efficiency of a logic block and is calculated by multiplying the power consumed and the input/output delay. Generally, an increase in the logic block area causes the latency to reduce and the power consumed to increase. In other words, there is a trade-off between the latency and power consumption. Therefore, a lower PD product indicates a more efficient performance under identical power conditions.

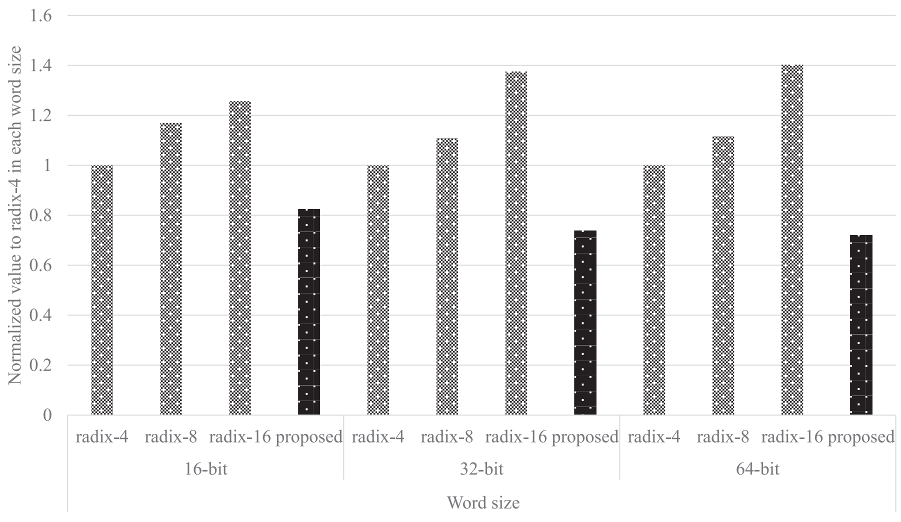


Fig. 3. Power-delay products normalized to radix 4 in each word.

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

Similar to the results reported in [1], conventional Booth multipliers exhibit poor performance characteristics with increasing radix and word size. On the other hand, the proposed radix-16 multiplier with weighted 2-stage Booth algorithm achieves lower PD products than those of conventional Booth multipliers and exhibits better performance characteristics with increasing radix and word size. We presume that these improved results are due to the substitution of the 17-to-1 MUX of the radix-16 encoding by two 5-to-1 MUXs when using the proposed weighted 2-stage Booth algorithm.

Table II lists the total area and the power consumption of each encoder and the ratio of the extra adders for each radix and word size. Each encoder consists of MUXs, buffers, and extra adders. The areas and the power consumption of MUXs and buffers can be calculated by subtracting those corresponding to the extra adders from the overall area and the power consumed. The MUXs and buffers consume the most hardware resources as well as power in conventional Booth multipliers. On the other hand, the additional adders consume more than half the power and the area in the proposed multiplier because the 17-to-1 MUXs have been replaced with the 5-to-1 MUXs and adders when using the proposed weighted 2-stage Booth algorithm. From these observations, we infer two points. First, the area and the power consumption of the MUXs are more critical than those of the adders in the 32-nm process. Second, the proposed weighted 2-stage Booth algorithm is more favorable in cases where the radix is high and the word size large.

Table II. Total area and power consumption of each encoder and the ratio of extra adders for each radix and word size.

		Area (μm^2)			Power (μW)		
		64 bit	32 bit	16 bit	64 bit	32 bit	16 bit
Radix 4	Total	32715	7559	2053	3490	1010	285
	Adder (%)	0	0	0	0	0	0
Radix 8	Total	36468	10034	3237	4290	1380	555
	Adder (%)	2182 (5.98)	977 (9.74)	301 (9.3)	208 (4.85)	131 (9.49)	52 (9.37)
Radix 16	Total	58162	17011	5540	7100	2310	843
	Adder (%)	8477 (14.57)	4192 (24.64)	1352 (24.4)	962 (13.55)	522 (22.6)	224 (26.57)
Proposed	Total	40021	10378	3065	4479	1690	597
	Adder (%)	23654 (59.1)	6081 (58.6)	1581 (51.58)	3435 (76.69)	1068 (63.2)	367 (61.47)

6 Conclusions

The multiplier is one of the most commonly used arithmetic units. Generally, multipliers are implemented with radix-4 Booth encoders. Clearly, the higher radix Booth multiplier has the advantage of less partial products. However,

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014

the higher radix Booth multiplier requires extra adders and large input MUXs. A novel weighted 2-stage Booth algorithm was proposed to resolve this problems. The proposed encoding algorithm facilitates the use of large input MUXs of the higher radix instead of small input MUXs and adders. In the case of the study performed with a 32-nm process, the synthesis results indicate that the multiplier encoded by the proposed weighted 2-stage algorithm achieves better PD products than those achieved with conventional multipliers. Therefore, we expect that the proposed algorithm can be broadly utilized for high-performance processors as well as digital signal processors, mobile applications, and various arithmetic units that use Booth encoding.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2013035233).

© IEICE 2014
DOI: 10.1587/elex.11.20140407
Received May 1, 2014
Accepted May 28, 2014
Publicized June 12, 2014
Copyedited July 10, 2014