Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation

Honglan Jiang, *Student Member, IEEE,* Jie Han, *Member, IEEE,* Fei Qiao, and Fabrizio Lombardi, *Fellow, IEEE*

Abstract—The Booth multiplier has been widely used for high performance signed multiplication by encoding and thereby reducing the number of partial products. A multiplier using the radix-4 (or modified Booth) algorithm is very efficient due to the ease of partial product generation, whereas the radix-8 Booth multiplier is slow due to the complexity of generating the odd multiples of the multiplicand. In this paper, this issue is alleviated by the application of approximate designs. An approximate 2-bit adder is deliberately designed for calculating the sum of $1\times$ and $2\times$ of a binary number. This adder requires a small area, a low power and a short critical path delay. Subsequently, the 2-bit adder is employed to implement the less significant section of a recoding adder for generating the triple multiplicand with no carry propagation. In the pursuit of a trade-off between accuracy and power consumption, two signed 16×16 bit approximate radix-8 Booth multipliers are designed using the approximate recoding adder with and without the truncation of a number of less significant bits in the partial products. The proposed approximate multipliers are faster and more power efficient than the accurate Booth multiplier; moreover, the multiplier with 15-bit truncation achieves the best overall performance in terms of hardware and accuracy when compared to other approximate Booth multiplier designs. Finally, the approximate multipliers are applied to the design of a low-pass FIR filter and they show better performance than other approximate Booth multipliers.

Index Terms—multiplier, approximate computing, radix-8, Wallace tree, truncation.

1 Introduction

Computer arithmetic is extensively used in many digital signal processing (DSP) applications. Multipliers are more complex than adders and subtractors, so the speed of a multiplier usually determines the operating speed of a DSP system. High precision is often looked as a strict requirement with other considerations such as delay, hardware complexity and power dissipation of a design. Some applications such as media processing, recognition and data mining are error-tolerant, so an approximate arithmetic unit can be employed [1].

The design of an approximate multiplier usually deals with the accumulation of partial products, which is a bottleneck in its operation. Truncation of the lower part of the partial products is a simple approximation scheme to reduce the delay and hardware overhead; such a scheme is referred to as fixed-width multiplier design. Several error compensation strategies have been recently proposed to improve the accuracy of fixed-width multipliers [2], [3], [4], [5], [6]. In the scheme of [2], the outputs of the Booth encoder are used to compensate the error generated by the truncated modified Booth multiplier; this method reduces the error by nearly 50% compared with a truncated design without error compensation. Another fixed-width modified Booth multiplier is presented in [3] by utilizing a simplified sorting network as a main part of the error compensation circuit; a high accuracy has been achieved too. Moreover, a probabilistic approach has been proposed to compensate the quantization error in a fixed-width Booth multiplier [4], [5], [6]. An adaptive conditional-probability estimator is also presented in [6]. The estimator is derived from a probabilistic analysis rather than a time-consuming exhaustive simulation as in [2] and [3].

An approximate 2×2 bit multiplier has been proposed in [7] by modifying one entry of the Karnaugh Map (K-Map). Three bits (rather than four) are used for the output of the 2×2 multiplier; the approximate 2×2 bit multiplier is then used as a basic block for designing multipliers for larger operands. This design achieves 31.78% to 45.4% reductions in power dissipation with only 1.39% to 3.32% errors on average. A different method for designing approximate multipliers consists of utilizing counters or compressors, as used in Wallace or Dadda trees [8], [9]. In [8], an inaccurate $4:2$ counter is applied to an approximate 4×4 bit Wallace multiplier. Multipliers of larger sizes have been built by using the approximate 4×4 bit multiplier for a shorter delay, lower power consumption and higher accuracy. Two approximate $4-2$ compressors are designed in [9] by changing several output values in the truth table; four designs are then proposed by using the approximate $4-2$ compressors in the Dadda tree of an 8×8 multiplier. Significant reductions in power dissipation and delay have been accomplished in [9]. A novel approximate multiplier has been proposed in [10] by using an approximate adder that omits carry propagation between partial products. A 20% reduction in delay and up to 69% reduction in power have been achieved for an 8×8 bit multiplier.

- *H. Jiang and J. Han are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada. Email: honglan@ualberta.ca, jhan8@ualberta.ca*
- *F. Qiao is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. Email: qiaofei@tsinghua.edu.cn*
- *F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, USA. E-mail: lombardi@ece.neu.edu*

Manuscript received October 8, 2015; revised October 11, 2015.

Most existing approximate multiplier designs consider the partial product accumulation stage and nearly all of them are for operations of unsigned numbers [7], [8], [9], [10]. The Booth algorithm is commonly used for a signed multiplication; therefore, Booth multipliers are discussed in this paper. The radix-4 recoded Booth algorithm is mostly utilized for high speed operations [2], [3], [4], [5], [6]. In contrast, the radix-8 algorithm generates fewer partial products than radix-4 and hence fewer adders are required for accumulating the partial products. However, the hardware-efficient radix-8 recoding algorithm is seldom used due to the extra time incurred for the operation of odd multiples of the multiplicand. Specifically, the step for computing three times the multiplicand requires a preliminary processing by an additional adder (with possibly a long carry propagation). This adder contributes to an increase of 10-20% in delay compared to the radix-4 algorithm (that generates multiples of the multiplicand simply by shifting) [11].

In this paper, approximate designs of a Booth multiplier are proposed; they are based on an approximation scheme that deals not only with the partial product accumulation, but also with the generation of recoded multiplicands. A 2-bit approximate recoding adder is initially designed to reduce the additional delay encountered in previous radix-8 schemes, thereby increasing the speed of the radix-8 Booth algorithm. A Wallace tree is leveraged to compute the sum of partial products and to further reduce the addition time. A truncation technique is then applied to the least significant partial products to reduce the power and delay. Two signed 16×16 bit approximate radix-8 Booth multipliers are then proposed; they are referred to as approximate Booth multipliers 1 and 2 (or ABM1 and ABM2). Simulation results show that ABM1 is 20% faster than the accurate radix-8 Booth multiplier. ABM2 saves as much as 44% in power dissipation compared to the accurate multiplier. This approximate multiplier is also 31% faster and requires 43% less circuit area. Finally, the ABMs are applied to a low-pass FIR filter; this application shows that the proposed approximate multipliers outperform other approximate multipliers found in the technical literature.

This paper is organized as follows. Section 2 presents the design of the approximate recoding adder. The designs of the approximate multiplier are described in Section 3. Section 4 shows the simulation results in comparison with the accurate and other approximate Booth multipliers. In Section 5, the approximate multiplier designs are applied to an FIR filter operation to show the applicability of the proposed designs. Finally, the paper is concluded in Section 6.

2 Proposed Approximate Recoding Adder

The partial products in the radix-2 and radix-4 algorithms can be easily generated by shifting or 2's complementing; 2's complementing is implemented by inverting each bit and then adding a '1' in the partial product accumulation stage. However, in the radix-8 algorithm, an odd multiple of the multiplicand Y (i.e., $(3Y)$) is required and must be calculated. A preliminary addition is required to calculate $3Y$ by implementing $Y + 2Y$, which incurs additional delay and power cost. Therefore, a high speed approximate recoding adder is designed for performing $Y + 2Y$ in this section.

2.1 Design of the Approximate Recoding Adder

Consider a 16-bit signed multiplier, the preliminary addition is shown in Fig. 1 (sign bits are shown in bold) [12]. The least significant bit of $3Y(S_0)$ is the same as y_0, and the sign bit of $3Y$ is given by y_{15}, because the sign does not change when the multiplicand is multiplied by 3. Therefore, only the 16 bits in the middle are processed. The carry propagation in a 16-bit adder takes a significant time compared with shifting.

To reduce the ripple carry propagation, two adjacent bits are added (instead of adding just one bit each time as in a conventional scheme) to take advantage of the duplication of the same bit, as shown in the box in Fig. 1. Take any 2-bit addition ($y_{i+1}y_i + y_iy_{i-1}$, where i is $1, 3, \cdots, 15$) as an example; the addition result is given by

$$\begin{aligned} &2^{i+2}C_{out} + 2^{i+1}S_{i+1} + 2^iS_i \\ &= 2^iC_{in} + 2^iy_{i-1} + 3 \times 2^iy_i + 2^{i+1}y_{i+1}, \end{aligned} \tag{1}$$

where y_{i-1}, y_i and y_{i+1} are the 3 bits of the multiplicand, y_i is the duplicated bit, C_{in} is the carry-in from the previous addition, S_i and S_{i+1} are the first and second sum bits of the 2-bit addition, and C_{out} is the carry-out of the 2-bit adder. The accurate truth table is shown in Table 1. Fig. 2 shows the K-Maps of these three outputs. The following functions can be obtained

$$\begin{aligned} C_{out} = &((y_{i-1} \vee y_{i+1} \vee C_{in}) \wedge y_i) \\ &\vee (C_{in} \wedge y_{i+1} \wedge y_{i-1}), \end{aligned} \tag{2}$$

$$\begin{aligned} S_{i+1} = &(((\overline{y_i \vee y_{i-1}}) \vee (\overline{C_{in}} \wedge y_{i-1})) \wedge y_{i+1}) \vee \\ &(((C_{in} \wedge \overline{y_i} \wedge y_{i-1}) \vee (\overline{C_{in}} \wedge y_i \wedge \overline{y_{i-1}})) \wedge \overline{y_{i+1}}), \end{aligned} \tag{3}$$

$$S_i = C_{in} \oplus y_i \oplus y_{i-1}, \tag{4}$$

where "$\vee$" and "$\wedge$" are OR and AND operations, respectively.

The circuit implementations of (2) and (3) are rather complex, so some approximations are made on S_{i+1} and C_{out}. As shown in Fig. 2, C_{out} becomes the same as y_i if 2 out of its 16 outputs are changed (shown in bold in Fig. 3). Hence, the computation and propagation of the carry are ignored. Similarly, S_{i+1} can also be simplified to y_{i+1} when 4 output values are changed (Fig. 3). Thus, the accurate truth table is changed to Table 2 for the approximation scheme. The output functions of the approximate 2-bit adder are simplified to

$$C_{out} = y_i, \tag{5}$$

$$S_{i+1} = y_{i+1}, \tag{6}$$

$$S_i = C_{in} \oplus y_i \oplus y_{i-1}. \tag{7}$$

The approximate 2-bit adder can be implemented by just one 3-input XOR gate as shown in Fig. 4. The probability of generating an error in this approximate 2-bit adder is given by

$$\begin{aligned} P(error) = &P(C_{in}y_{i+1}y_iy_{i-1} = 0010) \\ &+P(C_{in}y_{i+1}y_iy_{i-1} = 0110) \\ &+P(C_{in}y_{i+1}y_iy_{i-1} = 1101) \\ &+P(C_{in}y_{i+1}y_iy_{i-1} = 1001) \end{aligned} \tag{8}$$

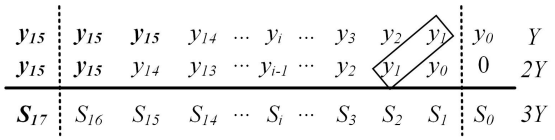

Fig. 1. 16-bit preliminary addition.

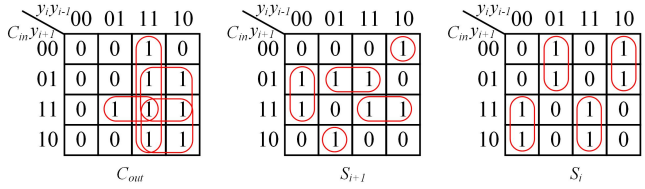

Fig. 2. K-Maps of the 2-bit addition.

Assume that any input of an adder is equally likely to occur, i.e., the occurrence probability of '1' or '0' at the input is 1/2; then, the probability of obtaining any value of $C_{in}y_{i+1}y_iy_{i-1}$ is 1/16. Therefore, the error rate of the approximate 2-bit adder is 1/4.

Due to the approximation, a +2 error (i.e., the difference between the approximate output and the accurate output) occurs when the input of the 2-bit adder is either "0010" or "0110"; the error is -2 when the input is either "1101" or "1001". These four errors are detected by the circuit in Fig. 5(a) (e_i is '1' when errors are detected). As revealed in Table 1 and Table 2, an error can be partially compensated by +1 or -1 when the least significant output bit S_i is flipped on the condition that e_i is '1'. This is accomplished by using an XOR gate as shown in Fig. 5(b), i.e., S_i is inverted when e_i is '1', otherwise S_i does not change. To fully correct these errors, C_{out} must be the same with y_{i+1}, and S_{i+1} must be inverted when error is detected. The error recovery circuit is shown in Fig. 5(c).

The approximate 2-bit adder cannot be used to add the entire 16 bits in the operands, because a large error would occur when the partial product $3Y$ is required in the multiplier's most significant part. However, the approximate adder can be used to implement the less significant part of the recoding adder and the most significant part can be implemented by a precise adder. Fig. 6 shows the circuit of the approximate recoding adder with 8 approximated bits; 4 approximate 2-bit adders and a 7-bit precise adder are utilized in the lower and higher parts, respectively. For the 16^{th} bit (S_{16}), $S_{16} = y_{15} \oplus y_{15} \oplus C_o = C_o$, where C_o is the carry-out of the 7-bit precise adder. In total, four XOR gates and a 7-bit adder are used in the approximate design; this is simpler than the circuit of a ripple-carry adder; moreover, the critical path delay is given only by the

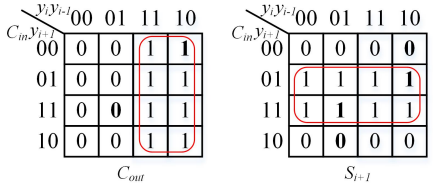

Fig. 3. K-Maps of the approximate 2-bit addition.

TABLE 1. TRUTH TABLE OF THE 2-BIT ADDER

$C_{out}S_{i+1}S_i$		y_iy_{i-1}			
		00	01	11	10
$C_{in}y_{i+1}$	00	000	001	100	011
	01	010	011	110	101
	11	011	100	111	110
	10	001	010	101	100

TABLE 2. TRUTH TABLE OF THE APPROXIMATE 2-BIT ADDER

$C_{out}S_{i+1}S_i$		y_iy_{i-1}			
		00	01	11	10
$C_{in}y_{i+1}$	00	000	001	100	**10**1
	01	010	011	110	1**1**1
	11	011	**01**0	111	110
	10	001	0**0**0	101	100

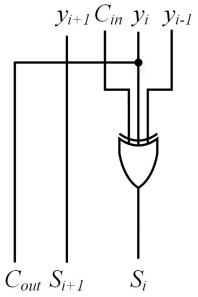

Fig. 4. Circuit of the proposed approximate 2-bit adder.

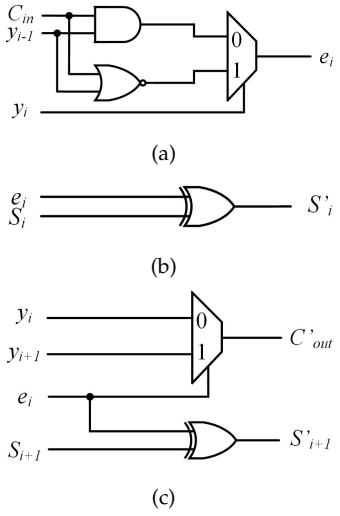

Fig. 5. (a) Error detection, (b) Partial error compensation and (c) Full error recovery circuits for the approximate 2-bit adder.

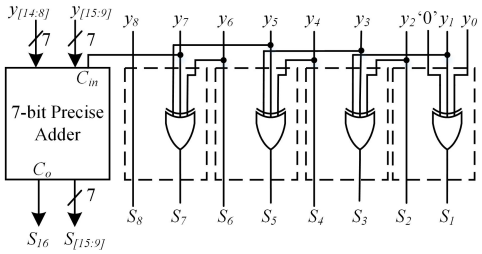

Fig. 6. Approximate recoding adder with 8 approximated bits.

delay of the 7-bit adder. The pass rate of the approximate adder (defined as the probability of obtaining a correct result [8]) is $(1-1/4)^4 = 31.64\%$.

2.2 Simulation Results for the Approximate Recoding Adder

Six types of approximate recoding adders are implemented in VHDL and synthesized by the Synopsys Design Compiler in STM 28nm CMOS process. Simulations are performed at a temperature of 25°C and a supply voltage of 1V. Critical path delay and area are then reported by the Synopsys Design Compiler. The power dissipation is estimated by the PrimeTime-PX tool using the value change dump file. The inputs for each design are 10 million random 16-bit binary numbers, and the clock period is 2 ns. Additionally, the values of power-delay product (PDP) and area-delay product (ADP) are calculated as comprehensive metrics.

To quantify the quality of the approximate designs, different metrics have been proposed, such as the error and pass rate, the error distance (ED), the mean error distance (MED) and the normalized mean error distance (NMED) [13]. ED is the arithmetic distance between an erroneous result and the corresponding correct result. MED is the mean value of the EDs for all possible inputs. The accuracy characteristics are obtained in MATLAB by simulating the functions of the approximate designs.

For assessing the best $3Y$ calculation, the following approximate recoding adders are considered: the approximate recoding adder with 8 approximated bits (ARA8), ARA8 with error compensation (using Fig. 5(b)) for the most significant approximate 2-bit adder (ARA8-2C), ARA8 with error recovery (using Fig. 5(c)) for the most significant approximate 2-bit adder (ARA8-2R), and the approximate recoding adder with 6 approximated bits (ARA6). According to [14], the lower part OR adder (LOA) [15] and the error tolerant adder type II (ETAII) [16] outperform the other approximate adders when an acceptable accuracy loss is considered. Moreover, the input pre-processing approximate adder (IPPA) in [10] shows good performance in accumulating partial products. Therefore, the following approximate adders are simulated for comparison: LOA whose lower bits are implemented by OR gates , ETAII, the modified ETAII [16] (ETAIIM), IPPA and the truncated ripple-carry adder (TRCA). Fig. 7 shows a cell of IPPA, where A_i and B_i are the i^{th} least bits of the adder's operands, S_i is the sum, and E_i is the error signal required for error compensation. No carry propagation is needed for IPPA unless an error compensation is considered. ETAII is shown in Fig. 8, where a conventional adder is divided into several segments. Hence, the carry propagation chain is significantly reduced by operating different segments in parallel.

TABLE 3. ACRONYMS OF APPROXIMATE ADDERS

Acronym	Meaning
ARA8	Proposed approximate recoding adder
ARA8-2C	ARA8 with error compensation
ARA8-2R	ARA8 with error recovery
IPPA [10]	Input pre-processing adder
LOA [15]	Lower part OR adder
TRCA	Truncated ripple-carry adder
ETAII [16]	Error tolerant adder type II
ETAIIM [16]	Modified ETAII

TABLE 4. COMPARISON RESULTS OF APPROXIMATE ADDERS OPERATING AS A RECODING ADDER

Adder Type	Delay (ns)	Area (um^2)	Power (μW)	PDP (fJ)	ADP ($um^2.ns$)	Pass Rate (%)	MED
ARA8	0.73	32	18.37	13.41	23.36	31.61	73.41
ARA8-2C	0.73	35	20.96	15.30	25.55	31.56	41.79
ARA8-2R	0.90	37	20.42	18.38	33.30	42.19	18.37
ARA6	0.94	37	22.63	21.27	34.78	42.19	18.37
IPPA8	0.94	59	34.35	32.29	55.46	34.36	31.52
IPPA7	0.83	55	31.18	25.88	45.65	27.89	63.53
IPPA6	0.72	51	27.96	20.13	36.72	22.64	127.44
LOA8	0.80	31	18.40	14.72	24.8	17.37	79.78
LOA7	0.90	33	20.50	18.45	29.70	21.41	39.81
LOA6	1.01	36	22.61	22.84	36.36	26.55	19.75
TRCA7	0.77	26	16.83	12.96	20.02	0.39	254.01
TRCA6	0.88	29	19.29	16.98	25.52	0.78	126.06
TRCA5	0.98	33	21.71	21.28	32.34	1.56	62.01
TRCA4	1.09	36	24.16	24.40	39.24	3.15	30.01
ETAII4	0.54	57	35.77	19.32	30.78	94.13	256.22
ETAII5	0.70	58	38.51	26.96	40.60	97.71	62.81
ETAII6	0.80	58	38.01	30.41	46.40	99.28	14.71
ETAIIM	0.70	60	38.69	27.08	42.00	97.08	14.98

The acronyms of these approximate adders used hereafter are shown in Table 3. As the recoding adder is only utilized for calculating the value of $3\times$ multiplicand (i.e., $2Y+Y$), all other approximate adders are simulated for the same function. The circuit and accuracy characteristics of the approximate adders are shown in Table 4. The overheads of the error detection, compensation and recovery circuits for all designs are included in the reported results. The numbers following the labels of the adders identify the parameters in the schemes as follows.

- It is the number of the most significant bits used for error correction for IPPA.
- It is the number of the lower approximated bits implemented by OR gates for LOA.
- In TRCA, it is the number of truncated lower bits.
- The parameter is the segment length in ETAII.
- In the ETAIIM, 4 bits are used for generating the carries of the less significant segments and 8 bits for the most significant segment.

Among all approximate adders, ETAII4 is the fastest, and IPPA6, ARA8 and ARA8-2C are also very fast schemes, while LOA6 and TRCA4 are rather slow. ETAII and ETAIIM incur the largest power dissipation and circuit area due to

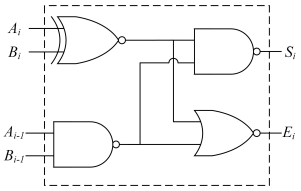

Fig. 7. The circuit of the input pre-processing adder cell [10].

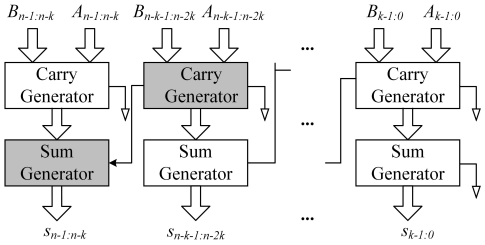

Fig. 8. The diagram of the error tolerant type II adder [14].

the segmented structure, while TRCA is the most power and area efficient design due to truncation. For the same reasons, ETAII and ETAIIM show a high pass rate (more than 90%), while TRCA has the lowest pass rate (less than 2%). The high pass rates of ETAII4 and ETAII5 cannot guarantee a high MED, because the EDs are rather significant, though errors seldom occur. Although IPPA performs well in [10] for partial product accumulation, it is not suitable for operating as a recoding adder. This occurs because IPPA is designed for an iterative addition in the accumulation of partial products for a multiplier. In this operation, error signals can be accumulated efficiently (e.g., by using OR gates), and then the error is compensated at the final stage using an accurate adder. For a recoding adder, however, the error must be compensated immediately, which makes IPPA less efficient. The power dissipation and area of the proposed approximate adders and LOA are very close. However, the proposed approximate adders have higher pass rates and lower MEDs than LOA.

As the values of ADP show the same trend as the PDP for all approximate adders, PDP is selected as the metric for hardware comparison. The MEDs and PDPs of all approximate adders are shown in a 2D plot (Fig. 9); the adders with MEDs larger than 80 are not included since they are not sufficiently accurate for a recoding adder. At a similar values of MED, the proposed approximate recoding adder always has a smaller PDP than the other approximate adders, e.g., the values of MED for ARA8, LOA8, TRCA5, IPPA7 and ETAII5 are nearly 70, ARA8 has the lowest PDP (13.41 fJ). Likewise, ARA8-2R and LOA7 have close values of PDP (about 20 fJ), ARA8-2R shows smaller MED (18.37). Due to the error recovery circuit, ARA8-2R has the same accuracy characteristics as ARA6 (both of them utilize 6 approximated bits). Nevertheless, ARA8-2R is faster and more power efficient than ARA6. Therefore, ARA8, ARA8-2C and ARA8-2R show the best tradeoff in hardware and accuracy in the implementation of the recoding adders for an approximate radix-8 Booth multiplier.

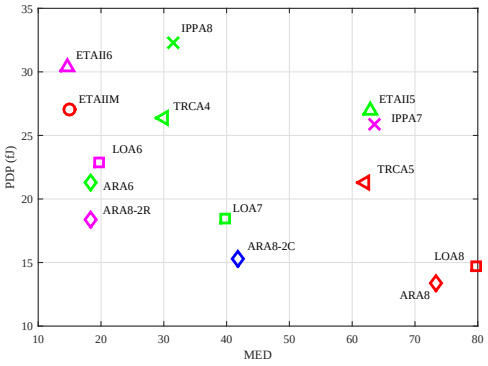

Fig. 9. The MEDs and PDPs of the approximate adders as recoding adders.

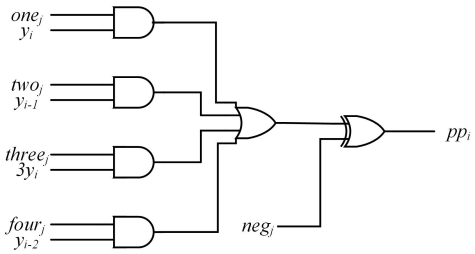

Fig. 10. Partial product generator.

3 Approximate Multiplier Designs

In this section, two approximate 16-bit signed multipliers based on the radix-8 Booth algorithm are proposed. A Booth multiplier consists of stages of multiplier encoding, partial product generation, partial product accumulation and the final addition. In the radix-8 Booth algorithm, nine types of partial products $(-4Y, -3Y, -2Y, -Y, 0, Y, 2Y, 3Y, 4Y)$ are generated by the multiplier encoder and the partial product generator. Moreover, a Wallace tree is used to implement the sum of the partial products to reduce the total multiplication time. The selection of the partial products as inputs to the Wallace tree is controlled by the partial product generator and is ultimately determined by the multiplier encoder. Fig. 10 shows the 1-bit partial product generator. The input signals of one_j, two_j, $three_j$, $four_j$ and neg_j are the multiplier recoding results according to the radix-8 algorithm [11]. y_i is one bit of the multiplicand, and $3y_i$ is the corresponding bit of $3Y$ calculated by the recoding adder. The AND gates are used to select the partial products and perform shift operation, while the XOR gate completes the inversion of the positive multiple of the multiplicand for a negative recoding factor.

For the 16-bit multiplier, the radix-8 recoding algorithm generates six signed digits. Hence, six partial products are generated. The dot-notation of the partial products for the 16-bit multiplier is shown in Fig. 11, in which sign extension elimination technique is used [3]. In Fig. 11, a dot represents a bit of a partial product, a square is the sign of a recoding factor (neg_j in Fig. 10). The sign bit of each partial product is in gray, the bars on the top of them mean the inverting operation. The partial products are accumulated by Wallace tree in this paper.

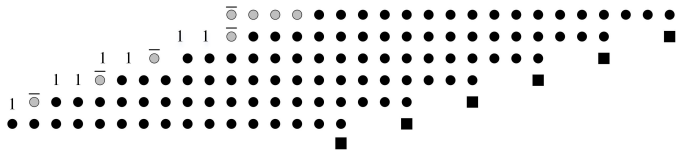

Fig. 11. Partial product tree of a 16-bit radix-8 Booth multiplier. ●: a partial product; ◉: the sign bit; $\overline{◉}$: the inverted sign bit; ■ : the sign of the recoding factor.

Besides the approximation of the recoding adder of the 16-bit Booth multiplier, two approximate designs are further proposed in the accumulation stage.

- In the first approximate Booth multiplier (ABM1), the accumulation of the partial products is accurate. Therefore, the most accurate approximate recoding adder is utilized, i.e. ARA8-2R.
- As some of the partial products are imprecise due to the approximate recoding adder, it may not be necessary to accumulate them accurately. Consequently, a few lower bits of the partial products are truncated in the second approximate Booth multiplier (ABM2) to save additional power and reduce the delay. Nine and fifteen bit truncations are used in ABM2. For the nine bit truncation, ARA8-2C (in ABM2-C9) and ARA8-2R (in ABM2-R9) are used as the recoding adders. Three configurations of the approximate recording adder are used in the fifteen bit truncation scheme: ARA8 (in ABM2-15), ARA8-2C (in ABM2-C15) and ARA8-2R (in ABM2-R15). An additional '1' (average error) is finally added to the 17^{th} bit of the 15-bit truncation multiplier to compensate the error generated by the truncated lower part. Moreover, the 15-bit truncation multiplier can also be used as a fixed-width multiplier.

4 Simulation Results of Multiplier

In this section, the hardware overheads and accuracy of the proposed approximate 16-bit multiplier designs are assessed. Meanwhile, the performance of the accurate and several other approximate Booth multipliers are compared with the proposed designs.

4.1 Hardware Measures

The proposed approximate multipliers, the accurate radix-8 (AcBM) and the other approximate Booth multipliers in [2] (BM04), [3] (BM11) and [6] (PEBM) are implemented and simulated by Synopsys Design Compiler in STM 28nm CMOS process with a supply voltage of 1V at 25°C. The critical path delay and area are reported by Synopsys Design Compiler, and the power is measured by the PrimeTime-PX tool. 10 million random input combinations are used for the power analysis and the clock cycle is 4 ns. Moreover, the fixed-width radix-8 Booth multiplier using the probability estimation theory in [6] (denoted as PEBM8) is also compared. The critical path delay, area and power consumption are reported in Table 6. The acronyms used for the multipliers are shown in Table 5. The structure of

TABLE 5. ACRONYMS OF APPROXIMATE MULTIPLIERS

Acronym	Booth Algorithm	Truncation (bits)	Recoding Adder
AcBM	radix-8	0	accurate
ABM1	radix-8	0	RAR8
ABM2-C9	radix-8	9	RAR8-2C
ABM2-R9	radix-8	9	RAR8-2R
ABM2-15	radix-8	15	RAR8
ABM2-C15	radix-8	15	RAR8-2C
ABM2-R15	radix-8	15	RAR8-2R
BM04 [2]	radix-4	15	no
BM11 [3]	radix-4	15	no
PEBM [6]	radix-4	15	no
PEBM8	radix-8	15	accurate

AcBM is the same as ABM1, except for the recoding adder. Table 6 shows that the approximate recoding adder causes ABM1 to have a speed improvement of nearly 20%, thus, the recoding adder in the radix-8 algorithm results in a significant difference in the Booth multiplier. The critical path delay of ABM2-15 and ABM2-C15 improve by 31% compared with the accurate radix-8 design; moreover, the circuit areas of ABM2 with 9-bit truncation (T9) and 15-bit truncation (T15) are roughly 18% and 43% smaller than the accurate design. The power consumption of ABM2 (T9) and ABM2 (T15) are 18% and 44% less than AcBM, therefore their PDPs and ADPs are also smaller.

PEBM8 is much slower than the proposed approximate multipliers due to the accurate recoding adder. BM04 and BM11 have longer delays and smaller area than PEBM although they are all based on the radix-4 Booth algorithm. This is mainly because that the partial product accumulation of BM04 and BM11 are performed by an array structure while it is performed by a carry-save adder tree for PEBM. As for area, ABM1 requires the largest area (724 um^2), BM04, ABM2-15, ABM2-C15 and ABM2-R15 require approximately an area of 420 um^2, and the area for ABM2-C9 and ABM2-R9 is nearly 600 um^2. For power dissipation, ABM2-15 shows the best performance, ABM2-C15 and ABM2-R15 are better than the other approximate multipliers. The area and power dissipation of ABM1, ABM2-C9 and ABM2-R9 are larger than the fixed-width multipliers, hence truncation is an effective scheme to reduce hardware overheads. The proposed fixed-width multipliers have smaller values of area and power consumption than the other fixed-width multipliers (except that BM04 has the smallest area) due to the hardware-efficient radix-8 algorithm, thereby their PDPs and ADPs are also very small.

4.2 Accuracy Measures

MED cannot be used for a fair comparison of two approximate designs with different number of bits; the definition of NMED is proposed in [13] to overcome this limitation. The NMED is defined as the normalization of MED by the maximum output of the accurate design. The maximum absolute error (MAE) of the approximate multipliers is measured to evaluate their error magnitude. Additionally, the relative error distance (RED, defined as the error distance over the absolute accurate result) is used to assess the error distribution of approximate multipliers.

TABLE 6. HARDWARE COMPARISON RESULTS OF THE APPROXIMATE BOOTH MULTIPLIERS

Multiplier Type	Delay (ns)	Area (um^2)	Power (uW)	PDP (fJ)	ADP ($um^2.ns$)
AcBM	2.99	737	371.9	1111.98	2203.63
ABM1	2.38	724	363.3	865.84	1723.12
ABM2-C9	2.23	604	305.2	680.60	1346.92
ABM2-R9	2.41	606	305.5	736.26	1460.46
ABM2-15	2.07	419	206.8	428.08	867.33
ABM2-C15	2.07	422	208.1	430.77	873.54
ABM2-R15	2.25	424	208.0	468.00	954.00
BM04	1.93	407	226.5	437.15	785.51
BM11	1.96	475	258.1	505.88	931.00
PEBM	1.83	528	264.3	483.67	966.24
PEBM8	2.86	452	221.6	633.78	1292.72

The functions of the proposed and previous approximate multipliers are simulated in MATLAB, their values of pass rate, NMED, MAE and P_{RED} (the probability of getting a RED smaller than 2%) are reported in Table 7. ABM1 gives the highest pass rate (55.937%), while the pass rates of the other approximate multipliers are nearly 0 due to truncation. In terms of NMED, ABM1 has the smallest value, while ABM2-15 has the largest value. ABM1 and ABM2-R9 have similar NMEDs; i.e., the error due to the recoding adder is more significant than the one due to the truncation. The NMEDs of all proposed approximate multipliers using ARA8-2R as recoding adders (ABM1, ABM2-R9 and ABM2-R15) are smaller than PEBM8 and BM04. The proposed approximate designs have relatively high MAEs because $3Y$ is required in the most significant partial product and hence, the error due to the approximate recoding adder incurs in a rather large magnitude for the final multiplication result. However, as per their small values of NMED and high values of P_{RED}, the probability of occurrence of a large MAE is extremely small in these schemes. In terms of P_{RED}, ABM1 incurs the largest value, and ABM2-C9 and ABM2-R9 also have higher values of P_{RED} than fixed-width multipliers. The accuracy of PEBM is higher than that of PEBM8 in terms of NMED and P_{RED}; this indicates that the probability estimator based error compensation is more suited for the radix-4 Booth algorithm than for the radix-8. In summary, ABM1 shows the best performance in terms of pass rate, NMED and P_{RED}, while ABM2-15 has the highest NMED and MAE.

Fig. 12 shows a comprehensive comparison of all approximate Booth multipliers by considering both NMED and PDP. Among the fixed-width multipliers, ABM2-R15 is the most efficient design with moderate values of NMED and PDP. BM11 and PEBM show better NMED but larger PDP, while BM04 is opposite. ABM1 and ABM2-R9 have nearly the same small value of NMED, but ABM2-R9 has a better PDP. Therefore, ABM2-R9 should be selected if the accuracy is required within the reported range. ABM2-15 and ABM2-C15 have similar small values of PDP, but ABM2-C15 has a smaller NMED. Thus, ABM2-C15 should be considered at a PDP value of 430 fJ.

5 FIR Filter Application

In this section, the proposed multipliers are applied to a 30-tap low-pass equiripple Finite Impulse Response (FIR) filter to assess the viability of these designs. The FIR filter is designed by the Filter Design & Analysis Tool (FDATOOL) in MATLAB. The pass-band and stop-band frequencies of the filter are 8 kHz and 15 kHz, respectively. The input of the FIR filter is the sum of three sinusoidal variables $x_1(n)$, $x_2(n)$ and $x_3(n)$ with 1 kHz, 15 kHz, and 20 kHz frequencies, respectively, and a White Gaussian Noise $\eta(n)$ with $-30dBW$ power, i.e., $x(n) = x_1(n) + x_2(n) + x_3(n) + \eta(n)$. The White Gaussian Noise is used to simulate the random effects found in nature.

TABLE 7. ACCURACY COMPARISON RESULTS OF THE APPROXIMATE BOOTH MULTIPLIERS

Multiplier Type	Pass Rate (%)	NMED (10^{-5})	MAE (10^5)	P_{RED} (%)
ABM1	55.937	1.92	3.18	99.77
ABM2-C9	0.257	4.43	6.75	99.43
ABM2-R9	0.274	1.97	3.19	99.74
ABM2-15	0.006	9.07	13.23	98.40
ABM2-C15	0.003	5.73	7.25	98.79
ABM2-R15	0.011	3.41	3.67	99.08
BM04	0.006	5.31	2.29	97.72
BM11	0.014	2.18	1.25	99.17
PEBM	0.011	2.26	1.38	99.10
PEBM8	0.009	3.50	1.29	98.98

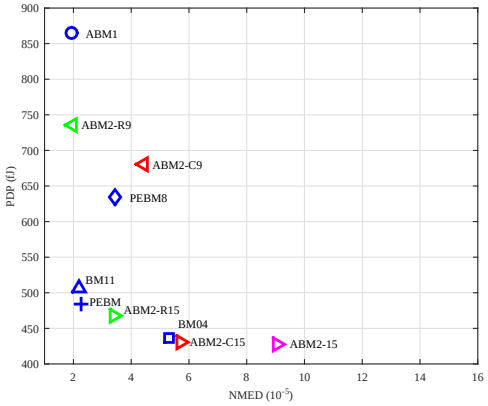

Fig. 12. NMEDs and PDPs of the approximate Booth multipliers.

The approximate 16×16 bit multipliers are applied to compute the output of the filter, while the adders used here are accurate. To assess the performance of the approximate multipliers for the FIR filter operation, the input signal-to-noise ratio (SNR_{in}) and output signal-to-noise ratio (SNR_{out}) are used. The same input signal with an SNR_{in} of 3.89 dB (due to a randomly generated White Gaussian Noise) is utilized for all operations.

The simulation results of the FIR filter operation are shown in Fig. 13, in which the output signal-to-noise ratios (SNR_{out}) are sorted in descending order. ABM1, ABM2-C9 and ABM2-R9 obtain nearly the same SNR_{out} (about 27 dB), and this value is higher than those of the fixed-width multipliers. The minor loss in SNR occurs mainly due to the small relative error distance of these multipliers is acceptable for FIR operation. BM11 achieves the highest SNR_{out} among all fixed-width multipliers. The values of SNR_{out} for ABM2-C15 and ABM2-R15 are slightly lower, around 25 dB. The output signal-to-noise ratios of PEBM8

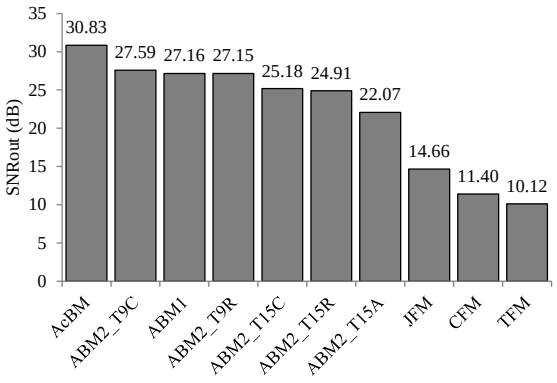

Fig. 13. Sorted SNR_{out} for the accurate and approximate multipliers.

and BM04 are much lower than the proposed multipliers. Overall, the multipliers with no truncation (ABM1) and with 9-bit truncation (ABM2-C9 and ABM2-R9) perform better than those with 15-bit truncation for this application. Higher bit truncation results in a larger error distance and hence, the effect is more pronounced in the final filter result. Specifically, the performance of ABM2-C9 in the filter application is better than ABM1 and ABM2-R9 although its NMED is more than two times larger. These results indicate that the NMED, while useful in evaluating the quality of an approximate design in general, is not always accurate when assessing the design for a specific application. In this case, the filter operation is inherently error resilient, because some noise could be filtered out. For the same reason, the error caused by an approximate multiplier could be mitigated in some cases. Finally, the NMED and MAE of ABM2-15 are larger than those of PEBM8 and BM04. However, its filter application result is better, which is consistent with the RED results in Table 7 (given as P_{RED}). This indicates that the RED is reliable in predicting the performance of an approximate design in the filter application, because it considers the specific input in an evaluation. Overall, the proposed approximate multipliers are the best schemes for the FIR filter application.

6 Conclusion

In this paper, different signed 16×16 bit approximate radix-8 Booth multiplier designs have been proposed. Initially, an approximate 2-bit adder consisting of a 3-input XOR gate has been proposed to calculate the triple of binary numbers. The error detection, compensation and recovery circuits of the approximate 2-bit adder have also been presented. The 2-bit adder is then employed to implement the lower part of an approximate recoding adder for generating a triple multiplicand without carry propagation; it overcomes the issue commonly found in a radix-8 scheme. In the proposed signed approximate radix-8 Booth multipliers, referred to as ABM1 and ABM2, a truncation technique has been employed to further save power and time. The parallel processing by a Wallace tree is then employed to speed up the addition of partial products.

The simulation results have shown that the proposed approximate recoding adders (ARA8, ARA8-2C and ARA8-2R) are more suitable (in terms of hardware efficiency and accuracy) for a radix-8 Booth multiplier than other approximate adders. The recoding adder is very important for the critical path delay of the multiplier. However, the error due to the recoding adder is more significant than the one caused by truncation (provided the truncation number of the partial products is less than or equal to 9 for a 16×16 bit multiplier). The simulation results in an FIR filter application have shown that the proposed ABM1, ABM2-C9 and ABM2-R9 perform well with only a 3 dB drop in output signal-to-noise ratio. With similar values of PDP, the proposed designs outperform the other approximate multipliers in the FIR filter operation, thus these designs may be useful for low-power and imprecise operations in error-resilient systems.

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *ETS*, 2013, pp. 1–6.

[2] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Transactions on VLSI Systems*, vol. 12, no. 5, pp. 522–531, 2004.

[3] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified booth multipliers for lossy applications," *IEEE Transactions on VLSI Systems*, vol. 19, no. 1, pp. 52–60, 2011.

[4] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic estimation bias circuit for fixed-width booth multiplier and its dct applications," *IEEE Transactions on Circuits and Systems II*, vol. 58, no. 4, pp. 215–219, 2011.

[5] Y.-H. Chen, C.-Y. Li, and T.-Y. Chang, "Area-effective and power-efficient fixed-width booth multipliers using generalized probabilistic estimation bias," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 277–288, 2011.

[6] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers," *IEEE Transactions on Circuits and Systems I*, vol. 59, no. 3, pp. 594–603, 2012.

[7] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *2011 24th International Conference on VLSI Design*, 2011, pp. 346–351.

[8] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in *ICCD,*. IEEE, 2013, pp. 33–38.

[9] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, 2015.

[10] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *DATE*, 2014, p. 95.

[11] B. Millar, P. E. Madrid, and E. Swartzlander, "A fast hybrid multiplier combining booth and wallace/dadda algorithms," in *Proceedings of the 35th Midwest Symposium on Circuits and Systems*, 1992, pp. 158–165.

[12] J. Hidalgo, V. Moreno-Vergara, O. Oballe, A. Daza, M. Martín-Vázquez, and A. Gago, "A radix-8 multiplier unit design for specific purpose," in *XIII Conference of Design of Circuits and Integrated Systems*, vol. 10, 1998, pp. 1535–1546.

[13] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.

[14] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*. ACM, 2015, pp. 343–348.

[15] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems*, vol. 57, no. 4, pp. 850–862, 2010.

[16] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *ISIC*. IEEE, 2009, pp. 69–72.