Microelectronics Journal 101 (2020) 104816

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Approximate radix-8 Booth multiplier for low power and high speed applications

Bipul Boro, K. Manikantta Reddy *, Y.B. Nithin Kumar, M.H. Vasantha

Department of Electronics and Communications Engineering, National Institute of Technology Goa, India

A R T I C L E I N F O

Keywords:
Approximate computing
Booth multiplier
Booth encoder
Approximate multiplier

A B S T R A C T

Approximate computing is an emerging circuit design technique which reduce the energy consumption with acceptable degradation in accuracy. Three approximate radix-8 Booth multipliers are proposed in this paper to explore the advantages of approximate computing. These multipliers are designed by using two proposed approximate Booth encoders for the generation of approximate partial products. Approximate partial products are introduced into a few number of least significant columns (AC) of the partial product matrix. The proposed multipliers with 16-bit inputs are simulated using a 45-nm CMOS technology library. For AC = 16, the results indicate that the first proposed multiplier reduces power delay product (PDP) and Area by 33% and 23% respectively as compared to conventional radix-8 Booth multiplier. Moreover, it has a Normalized Mean Error Distance (NMED) of 1.43E-5. The second proposed multiplier shows 43% and 31% reduction in PDP and Area respectively with an NMED of 1.664E-5. Similarly, the third proposed multiplier shows 37% and 25% reduction in PDP and Area respectively with an NMED of 1.512E-5. The accuracy of proposed multipliers are verified with real-time applications in image processing and deep learning.

1. Introduction

Approximate computing is emerging as a new direction in VLSI design, which addresses the well-known challenges such as power consumption and delay [1]. In approximate computing, errors are intentionally introduced into the system for simplifying the circuit logic or system architecture. This method can be used in error-tolerant applications like multimedia processing [1], image filtering [2], data mining, and machine learning [3]. Hardware accelerators for such applications emerged as cost effective and energy efficient alternative to CPUs and GPUs. Hence, the recent research on approximate computing has been concentrated on developing application-specific approximate hardware accelerators and automation frameworks to develop such accelerators.

An Iris recognition application is proposed in Refs. [4] with approximation methodologies applied at algorithm (software) and hardware accelerator stages to minimize the runtime of the application. Architectural space exploration methodologies are proposed in Refs. [5,6] to efficiently enable approximate computing in application specific hardware accelerators. These methodologies search, select and combine most optimal approximate circuits from the available approximate circuit libraries [7,8]. Since elementary arithmetic circuits such as adders and multipliers are the basic building blocks of hardware accelerators, selection of these circuits with proper bit-width combination are considered as trade-off points against accuracy and electrical performance.

Approximate multipliers are broadly analyzed in the literature, since multiplier is the most fundamental and energy-hungry block in microprocessors, digital signal processors and ASIC/FPGA based hardware accelerators. A multiplier performs multiplication in three steps: partial products generation, partial products accumulation and carry propagate addition. In case of Dadda/Wallace multipliers, partial products are generated by bit-wise multiplication of multiplicand and multiplier using AND logic gates. Later, adders and compressors are used to accumulate (add) these partial products.

In high speed applications, the Booth multiplier based on modified Booth encoding (MBE) [9] is widely employed. It is the most popular multiplier for signed multiplication. For the multiplication of N-bit operands X and Y, the radix-2^r ($r > 0$) MBE encodes the bits of multiplier (Y) in groups of ($r + 1$)-bits with an overlap of 1-bit. The radix -2^r MBE reduces the height of the partial product matrix from N to $\lceil N/r \rceil$ through encoding Y. This reduces the number of adders required

* Corresponding author.
E-mail addresses: bipulboro567@gmail.com (B. Boro), reddy78922@gmail.com (K.M. Reddy), nithin.shastri@nitgoa.ac.in (Y.B.N. Kumar), vasanthmh@nitgoa.ac.in (M.H. Vasantha).

https://doi.org/10.1016/j.mejo.2020.104816
Received 7 November 2019; Received in revised form 8 April 2020; Accepted 14 May 2020
Available online 22 May 2020
0026-2692/© 2020 Elsevier Ltd. All rights reserved.

to accumulate the partial products and results in a faster multiplication as compared to Dadda/Wallace multipliers. As the value of r increases, the hardware requirement in the accumulation stage decreases, but the complexity of MBE increases. For example, radix-4 MBE reduces the height of the partial product matrix from N to $\lceil N/2 \rceil$ and requires 0, ± 1 and ± 2 multiples of multiplicand (X) to generate partial products. All these multiples can be easily generated by applying shift and negation operations on X. Similarly, radix-8 MBE reduces the height of partial product matrix from N to $\lceil N/3 \rceil$ and requires 0, ± 1, ± 2, ± 3, and ± 4 multiples of X. Out of these terms, $3X$ is a non-power of two multiple called as *hard multiple*, which cannot be obtained via shifting X. The $3X$ term is generally obtained by adding X and $2X$ terms using an N-bit adder called as *recoding adder*. Later, X, $3X$ and Y terms are passed as inputs to the multiplier. The recoding adder increases overall critical path delay, which is the main drawback of the radix-8 Booth multiplier [10].

To reduce the complexity of partial product generation in the radix-8 Booth multiplier, this paper proposes three approximate Booth encoders. The first design generates $3X$ term using approximate half adders, which simplifies the partial product generation. The second design approximates a $4X$ term with $3X$ to further simplify the partial product generation. The third design uses a combination of these two designs to achieve optimal performance. The proposed multipliers are described in Verilog HDL and simulated using a 45-nm CMOS technology library. Accuracy of the proposed multipliers are analyzed with some of the well-known error metrics such as Normalized Mean Error Distance (NMED), Mean Relative Error Distance (MRED), Pass Rate, and P_{RED}. The results indicate that the proposed multipliers provide better trade-off for electrical performance and accuracy, compared to existing approximate Booth multipliers. Finally, the effectiveness of the proposed multipliers are verified with image processing and deep learning applications.

This paper is organized as follows: A brief review of available approximate multipliers is presented in Section 2. Section 3 reviews the conventional radix-8 Booth multiplication. The design of proposed approximate MBEs and multipliers are presented in Section 4. Simulation results and error metrics of proposed multipliers are analyzed in Section 5. The performance of proposed multipliers are verified with real-time applications in Section 6. Finally, Section 7 concludes the paper.

2. Related work

A brief review of the available approximate multipliers with approximations at various stages of multiplication process is presented in this section.

Approximation of input operands: An error tolerant multiplier (ETM) proposed in Ref. [11] splits the input operands into two parts. Then partial products are generated only through higher order bits of the input operands. The error introduced due to the elimination of lower order partial products is compensated based on the lower order input bits. A dynamic range unbiased multiplier (DRUM) proposed in Ref. [12] uses only k-significant bits of the input operands starting from a leading logic-1 bit. These k-bit operands are then multiplied using $k \times k$ accurate multipliers. Approximate logarithmic multipliers proposed in Ref. [13,14] are based on Mitchell logarithmic multiplier that converts input operands into approximate logarithmic numbers. Approximate log based multiplier with minimal error bias [15] reduces the successive accumulation of errors when the approximate multipliers are cascaded. An approximate multiplier using piece-wise linear approximation with a guaranteed worst-case error is proposed in Ref. [16]. Although approximation of input operands reduces the hardware complexity of a conventional multiplier, these methods introduce large errors into the output.

Approximation of partial products: This method is adopted in literature [17–20] to simplify the partial product generation in Booth multipliers. Two radix-4 approximate MBEs (R4ABE1 and R4ABE2) are proposed in Ref. [17] by introducing 4 and 8 errors respectively in the five variable K-map that generates the logical expression of partial products. Approximate radix-4 Booth multiplier (ABM-M1) [18] introduces 4 errors into the K-map of partial product generator. Two more multipliers (ABM-M2 and ABM-M3) are also proposed by further simplifying the partial product generator based on LSBs of the input operands. Approximate hybrid Booth multiplier [19] encodes r LSBs of the input operand with approximate radix-2^r encoding and remaining MSBs with radix-4 encoding. The non-power of two multiples of the multiplicand in radix-2^r encoding are approximated with the nearest power of two values. An approximate recoding adder is proposed in Ref. [20] to generate the hard multiple of the multiplicand ($3X$) in radix-8 Booth multipliers.

Approximation of partial product tree: This method is considered in the design of fixed with multipliers (FWM) [21,22] that receives N-bit input and produces a N-bit product instead of $2N$-bit. The partial products corresponding to truncated output bits are divided into two parts referred as LP_{major} and LP_{minor}, which are then used to compensate for the truncation error. A single formula based compensation circuit for FWM using conditional probability estimation is proposed in Ref. [23]. Generally, FWMs achieve significant improvement in electrical performance but introduce large errors into the output. An efficient design methodology with proper utilization of LUTs is proposed in Ref. [24] for the design of FPGA based approximate multipliers.

Approximate accumulation of partial products: This method is widely studied in the recent literature by using approximate adders and compressors for the accumulation of partial products. Few approximate full adders are proposed in Refs. [1,25–27] by simplifying the logical expressions of adder outputs or by predicting carry signals. A dynamic accuracy configurable adder which achieves performance-quality trade-off is proposed in Ref. [28]. Various approximate 4:2 compressors are proposed [29–34] and efficiently used in the design of approximate Dadda multipliers. Approximate square-accumulate and multiply-accumulate architectures with self-healing (SquASH and MACISH) methodologies [35,36] introduce similar error with opposite polarities into a pair of squarers or multipliers. Some of these errors are subsequently canceled (healed) in the accumulation stage.

The available works on approximate Booth multipliers have mainly concentrated on radix-4 MBE due to the straightforward implementation. The performance of high radix Booth multipliers is limited by the requirement of non-power of two multiples of the multiplicand. Therefore, this paper proposes two approximate radix-8 MBEs to simplify the partial product generation of the radix-8 Booth multiplier. It can be combined with any of the available approximate accumulation techniques to further reduce the hardware complexity. The operation of conventional radix-8 Booth multiplier is elaborated in the following section.

3. Conventional Radix-8 Booth multiplier

Let X and Y are two N-bit signed binary numbers, which represent multiplicand and multiplier respectively. Then X and Y can be represented in 2's complement notation with the following equations:

$$X = -x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i \tag{1}$$

$$Y = -y_{N-1}2^{N-1} + \sum_{i=0}^{N-2} y_i 2^i \tag{2}$$

To obtain the product of X and Y, the radix-8 modified Booth encoding (MBE) [9,10] groups four bits of Y into a set $\{y_{3j+2}, y_{3j+1}, y_{3j}, y_{3j-1}\}$ with one bit overlapping as shown in Fig. 1. While grouping, the y_{-1} bit is assumed to be logic-0 and the sign bit y_{N-1} can be extended to higher bits. The Booth encoded digit of each group is given by the equation:

$$Y_j = -4y_{3j+2} + 2y_{3j+1} + y_{3j} + y_{3j-1} \text{ for } 0 \le j \le \lceil N/3 \rceil - 1 \tag{3}$$

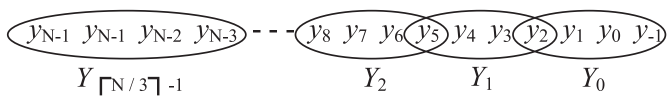
y_{N-1} y_{N-1} y_{N-2} y_{N-3}
y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0 y_{-1}
$Y_{\lceil N/3 \rceil -1}$
Y_2
Y_1
Y_0

Fig. 1. Grouping of multiplier bits in radix-8 Booth encoding.

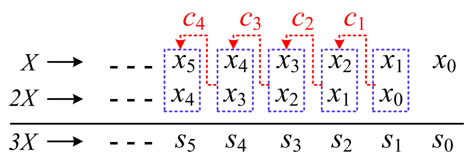
c_4 c_3 c_2 c_1
X
x_5 x_4 x_3 x_2 x_1 x_0
$2X$
x_4 x_3 x_2 x_1 x_0
$3X$
s_5 s_4 s_3 s_2 s_1 s_0

Fig. 2. Recoding adder design.

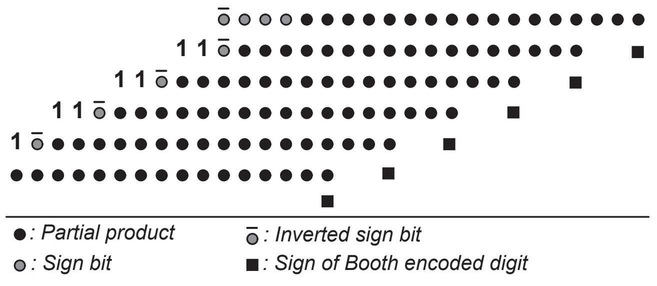

Fig. 3. Architecture of radix-8 Booth multiplier.

Here, Y_j can be equal to $0, \pm 1, \pm 2, \pm 3$ or $\pm$ 4 as shown in Table 1. Then, the partial product (PP) rows of multiplier are obtained by the equation:

$$PP_j = X * Y_j \text{ for } 0 \leq j \leq \lceil N/3 \rceil - 1 \tag{4}$$

Here, PP_j represents jth partial product row. From Eq. (3) and Eq. (4), the PP_j can be equal to 0, $\pm X$, $\pm 2X$, $\pm 3X$, $\pm 4X$ as shown in Table 1. When sign bit of a Booth encoded digit $y_{3j+2} = 0$, the PP_j takes only positive values. In these values, $2X$ and $4X$ can be obtained by left shifting X by one bit and two bit positions respectively. However, the $3X$ term need to be implemented as $X + 2X$ using a recoding adder as shown in Fig. 2. When $y_{3j+2} = 1$, a negative partial product row is obtained by 2's complement of the corresponding positive partial product row. Thus, it is implemented by complementing each individual bit of the positive partial product row and by adding logic-1 (y_{3j+2}) at the LSB. The partial product rows of radix-8 Booth encoded multiplier are shown in Fig. 3 using dot notation, where each dot denotes a partial product bit. The partial product bit PP_{ij} at the ith bit position in the jth row of the partial product matrix is shown Table 1, where s_i represents output bits of recoding adder. The simplified logic expression for PP_{ij} is given by:

$$\begin{aligned} PP_{ij} = \Big[& x_i \overline{(y_{3j+2} \oplus y_{3j+1})} (y_{3j} \oplus y_{3j-1}) \\ & + x_{i-1} (\overline{y}_{3j+1} y_{3j} y_{3j-1} + y_{3j+1} \overline{y}_{3j} \overline{y}_{3j-1}) \\ & + s_i (y_{3j+2} \oplus y_{3j+1})(y_{3j} \oplus y_{3j-1}) \\ & + x_{i-2} (\overline{y}_{3j+2} y_{3j+1} y_{3j} y_{3j-1} + y_{3j+2} \overline{y}_{3j+1} \overline{y}_{3j} \overline{y}_{3j-1}) \Big] \\ \oplus \, & y_{3j+2} \end{aligned} \tag{5}$$

From Eq. (5), when $y_{3j+2} = 1$, each individual bit of corresponding PP_j row is complemented through an XOR operation with y_{3j+2}. A signed multiplier requires few sign extension bits at the MSBs of each partial product row for the proper accumulation of partial products as shown in Fig. 3. The implementation of conventional MBE that generate PP_{ij} bits is shown in Fig. 4. The following section proposes two approximate radix-8 MBEs to simplify the generation of partial products.

4. Proposed approximate Booth multipliers

The presence of s_i bit in Eq. (5) increases the critical path delay of the multiplier, since the s_i bits are generated by recoding adder as shown in Fig. 2. Therefore, to generate the partial products in parallel,

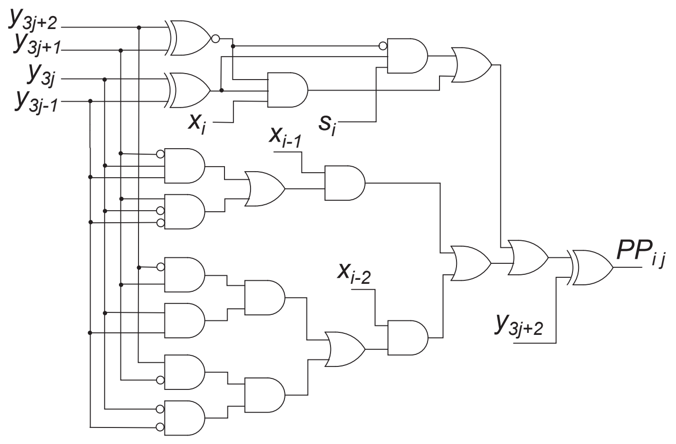

Fig. 4. Implementation of conventional MBE.

Table 1
Truth table of conventional and proposed approximate MBEs.

y_{3j+2}	y_{3j+1}	y_{3j}	y_{3j-1}	Y_j	PP_j	PP_{ij}	PP'_{ij}	PP''_{ij}
0	0	0	0	0	0	0	0	0
0	0	0	1	+1	X	x_i	x_i	x_i
0	0	1	0	+1	X	x_i	x_i	x_i
0	0	1	1	+2	2X	x_{i-1}	x_{i-1}	x_{i-1}
0	1	0	0	+2	2X	x_{i-1}	x_{i-1}	x_{i-1}
0	1	0	1	+3	3X	s_i	$x_i + x_{i-1}$	$x_i + x_{i-1}$
0	1	1	0	+3	3X	s_i	$x_i + x_{i-1}$	$x_i + x_{i-1}$
0	1	1	1	+4	4X	x_{i-2}	x_{i-2}	$x_i + x_{i-1}$
1	0	0	0	−4	-4X	$\overline{x_{i-2}}$	$\overline{x_{i-2}}$	$\overline{x_i + x_{i-1}}$
1	0	0	1	−3	-3X	$\overline{s_i}$	$\overline{x_i + x_{i-1}}$	$\overline{x_i + x_{i-1}}$
1	0	1	0	−3	-3X	$\overline{s_i}$	$\overline{x_i + x_{i-1}}$	$\overline{x_i + x_{i-1}}$
1	0	1	1	−2	-2X	$\overline{x_{i-1}}$	$\overline{x_{i-1}}$	$\overline{x_{i-1}}$
1	1	0	0	−2	-2X	$\overline{x_{i-1}}$	$\overline{x_{i-1}}$	$\overline{x_{i-1}}$
1	1	0	1	−1	-X	$\overline{x_i}$	$\overline{x_i}$	$\overline{x_i}$
1	1	1	0	−1	-X	$\overline{x_i}$	$\overline{x_i}$	$\overline{x_i}$
1	1	1	1	0	0	0	0	0

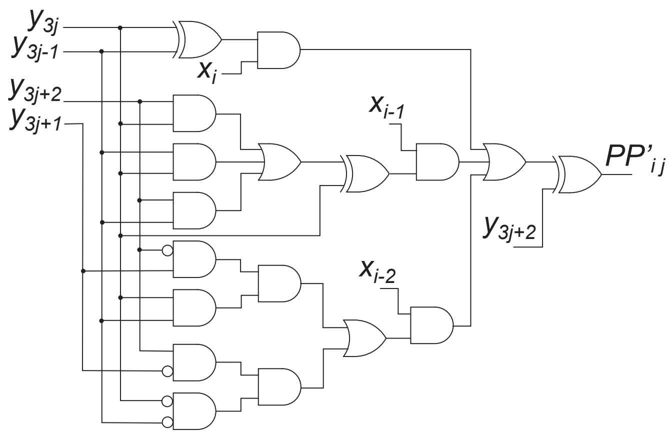

Fig. 5. Implementation of proposed R8ABE1.

two approximate MBEs are proposed in this section by approximating the s_i bit in Eq. (5).

4.1. Radix-8 approximate Booth encoder 1 (R8ABE1)

The addition of X and $2X$ starts with the addition of x_0 and x_1 bits using a half adder as shown in Fig. 2. The carry bit $c_1 = 1$ for only one out of four possible combinations of x_1x_0. Therefore, the value of c_1 can be approximated to logic-0 for all the combinations. To reduce the absolute difference between accurate and approximate result for the input combination $x_1x_0 = 11$, the value of sum bit (s_1) is approximated to logic-1 for this combination. Hence, operation of half adder can be approximated with a simple two input OR gate. As the carry bit is eliminated, all the bits of X and $2X$ can approximately added using OR gates with $s_i = x_i + x_{i-1}$. Therefore, Eq. (5) is simplified by replacing s_i bit with $x_i + x_{i-1}$. Hence, the proposed R8ABE1 generates approximate partial products using the simplified logical expression given by Eq. (6).

$$\begin{aligned}PP'_{ij} = \big[&x_i(y_{3j} \oplus y_{3j-1})\\ &+ x_{i-1}\left(y_{3j} \oplus \left(y_{3j+2}y_{3j} + y_{3j-1}y_{3j} + y_{3j+2}y_{3j-1}\right)\right)\\ &+ x_{i-2}(\overline{y}_{3j+2}y_{3j+1}y_{3j}y_{3j-1} + y_{3j+2}\overline{y}_{3j+1}\overline{y}_{3j}\overline{y}_{3j-1})\big]\\ &\oplus y_{3j+2} \end{aligned} \tag{6}$$

The truth table to generate PP'_{ij} is shown in Table 1. The proposed R8ABE1 introduces error when $Y_j = \pm 3$, which has a probability of 4/16 since 4 out of 16 combinations of inputs produce $Y_j = \pm 3$. Moreover, with $Y_j = \pm 3$, error is introduced into approximate partial product bits PP'_{ij} when $x_ix_{i-1} = 11$, which has probability of 1/4. Therefore, the overall probability of an approximate partial product PP'_{ij} to be erroneous is 1/16 only. The implementation of proposed R8ABE1 is shown in Fig. 5. It can be observed that the R8ABE1 does not need s_i input bits and eliminates the use of recoding adder.

4.2. Radix-8 approximate Booth encoder 2 (R8ABE2)

As shown in Table 1, $Y_j = \pm 4$ for only two out of sixteen possible combinations. Therefore, to further simplify the partial products generation, $4X$ term is approximated to $3X$ and implemented using OR gates. By substituting these approximations in Eq. (5), the proposed R8ABE2 generates approximate partial products using the simplified logical expression given by Eq. (7).

$$\begin{aligned}PP''_{ij} = \big[&x_i\left((y_{3j} \oplus y_{3j-1}) + \overline{y}_{3j+2}y_{3j+1}y_{3j-1}\right.\\ &\left.+ y_{3j+2}\overline{y}_{3j+1}\overline{y}_{3j}\right) + x_{i-1}\left((y_{3j+2} \oplus y_{3j+1})\right.\\ &\left.+ \overline{y}_{3j+2}y_{3j}y_{3j-1} + y_{3j+1}\overline{y}_{3j}\overline{y}_{3j-1}\right)\big] \oplus y_{3j+2}\end{aligned} \tag{7}$$

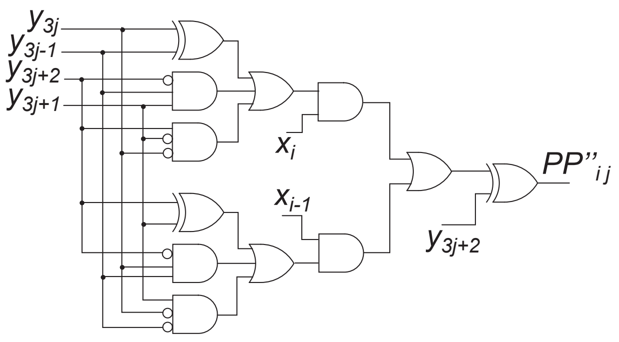

Fig. 6. Implementation of proposed R8ABE2.

The truth table to generate PP''_{ij} is shown in Table 1. As shown in Fig. 6, the proposed R8ABE2 further reduces hardware complexity as compared to R8ABE1.

4.3. Approximate multiplier design

Two approximate multipliers R8ABM1 and R8ABM2 are designed by generating partial products using proposed R8ABE1 and R8ABE2 respectively. The arrangement of partial products in approximate multiplier is similar to the conventional radix-8 Booth multiplier architecture shown in Fig. 3. The accumulation of partial products is performed using full adders and 4:2 compressors [37]. Introducing approximate partial products in all the columns of the multiplier causes significant errors in the output of the multiplier. Therefore, a chosen number of the less significant columns of the multiplier are generated using the proposed Booth encoders, and the remaining columns are generated using exact Booth encoders. The selection of number of columns that contain approximate partial products shows a trade-off between electrical performance and accuracy of multiplier. Hence, a parameter named as *Approximate Columns* (AC) is considered, which represents the number of columns (from LSB) with approximate partial products.

A third approximate multiplier R8ABM3 is also proposed by utilizing both the proposed Booth encoders. The R8ABM3 generates partial products in (AC/2) number of LSB columns using R8ABE2, next (AC/2) number of columns using R8ABE1 and the remaining MSB columns using conventional encoders. To find the optimized value of AC, the proposed multipliers are simulated with various values of AC. The simulation results and error analysis of three proposed multipliers are discussed in the following section.

5. Simulation results and error analysis

The three proposed multipliers R8ABM1, R8ABM2 and R8ABM3 along with conventional radix-8 Booth multiplier for $N = 16$ are described at gate level using verilog HDL. The multipliers are then synthesized using the Cadence Genus synthesis tool with a 45-nm CMOS technology library (gpdk045) provided by Cadence. Later, placement and routing are carried out using the Cadence Innovus implementation system. A testbench is created with one million randomly generated 16-bit input pairs. All the simulations are performed at 25°C operating temperature with a supply voltage of 1 V and a clock frequency of 200 MHz.

5.1. Simulation results

The simulation results of proposed 16-bit multipliers for AC = 12, 14, 16, 18, and 20 are shown in Table 2. Power, delay, power delay product (PDP), and layout area are considered as metrics to analyze the electrical performance and hardware complexity of proposed designs.

Table 2
Simulation results and Error metrics of proposed 16-bit approximate radix-8 Booth multipliers.

Multiplier	AC	Power (μW)	Delay (ns)	PDP (pJ)	Area (μm^2)	NMED ($\times 10^{-5}$)	MRED ($\times 10^{-2}$)	Pass rate (%)	P_{RED} (%)
Conventional Radix-8	0	586.4	0.922	0.541	2486	0	0	100.0	100.0
Proposed R8ABM1	12	516.6	0.809	0.418	2114	0.0531	1.227	54.81	99.54
	14	501.2	0.778	0.390	2031	0.672	6.015	51.68	98.38
	16	485.6	0.742	0.361	1924	1.430	14.264	47.22	97.16
	18	470.4	0.707	0.333	1848	6.812	67.378	41.74	92.12
	20	456.4	0.675	0.308	1786	28.244	386.42	33.56	85.64
Proposed R8ABM2	12	479.8	0.761	0.365	1942	0.161	3.416	40.15	99.01
	14	462.5	0.736	0.340	1829	0.912	10.270	36.52	97.89
	16	442.1	0.702	0.310	1715	1.664	22.512	32.14	95.23
	18	423.6	0.668	0.283	1631	12.071	111.48	26.72	89.55
	20	407.4	0.641	0.261	1574	51.520	572.24	19.46	80.37
Proposed R8ABM3	12	507.4	0.797	0.404	2071	0.080	1.774	51.15	99.41
	14	489.6	0.765	0.375	1970	0.744	7.292	47.13	98.23
	16	470.6	0.728	0.343	1852	1.512	17.151	41.94	96.48
	18	451.7	0.691	0.312	1764	8.916	85.022	35.73	91.09
	20	434.4	0.660	0.287	1688	38.583	471.23	27.22	83.27

5.1.1. Power consumption

The hardware required for the generation of partial products in proposed approximate multipliers is less than that of conventional multiplier as shown in Figs. 4–6. This reduces the overall power consumption of proposed multipliers. From Table 2, as compared to conventional multiplier, the power consumption of proposed R8ABM1 is reduced by 12% and 22% with AC = 12 and 20 respectively. For R8ABM2, the power consumption is reduced by 18% with AC = 12 and reduced by 30% with AC = 20. Similarly, the power consumption of R8ABM3 is reduced by 14% and 26% with AC = 12 and AC = 20 respectively.

5.1.2. Delay

From Eqs. (5)–(7), elimination of s_i bit in the partial product generation of proposed multipliers reduces the critical path delay. As shown in Table 2, the critical path delays of the proposed R8ABM1 with AC = 12 and 20 are 12% and 26% lesser respectively than that of conventional multiplier. Similarly, the delay of proposed R8ABM2 is reduced by 17% with AC = 12 and reduced by 30% with AC = 20. For R8ABM3, delay is reduced by 14% and 28% with AC = 12 and 20 respectively.

5.1.3. PDP

The simultaneous reduction in power consumption and delay of the proposed multipliers reduce the PDP. As compared to conventional multiplier, the PDP of proposed R8ABM1, R8ABM2 and R8ABM3 are reduced by 23%, 32% and 25% respectively with AC = 12, where as it is reduced by 43%, 52% and 47% respectively with AC = 20.

5.1.4. Area

Elimination of recoding adders and simplified logic expressions of proposed approximate Booth encoders reduce the overall area of proposed multipliers. The layout area savings of the proposed R8ABM1, R8ABM2 and R8ABM3 are 15%, 22% and 17% respectively with AC = 12, where as it is 28%, 36% and 32% respectively with AC = 20.

5.2. Error analysis

The error introduced into the output of the multiplier due to approximations is analyzed using well-known error metrics such as *normalized mean error distance* (NMED), *mean relative error distance* (MRED), *pass rate*, and P_{RED}. This error analysis is performed by exhaustively applying all possible input combinations and the results are shown in Table 2. The definition of considered error metrics and the performance of the proposed multipliers in terms of these error metrics are discussed in this subsection. These metrics depend on a parameter referred to as *error distance* (ED), which is defined as the difference between accurate and approximate output.

5.2.1. NMED

NMED is defined [38] as the mean value of the EDs for all possible inputs normalized with maximum possible ED.

$$\mathrm{NMED} = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} \frac{|ED_i|}{|ED_{\max}|} \tag{8}$$

where, N is the bit-length of the multiplier inputs. ED_i is the ED of approximate multipliers for the ith input combination. $ED_{\max}$ is the maximum possible ED at the output of approximate multiplier. The ED at the output of approximate multiplier increases with increase in number of columns (AC) that contain approximate partial products. As shown in Table 2, the NMED of proposed R8ABM1 increased from 0.0531×10^{-5} with AC = 12 to 28.244×10^{-5} with AC = 20. The NMED of proposed R8ABM2 increased from 0.161×10^{-5} with AC = 12 to 51.52×10^{-5} with AC = 20. Similarly, the NMED of proposed R8ABM3 also increased from 0.08×10^{-5} with AC = 12 to 38.583×10^{-5} with AC = 20.

5.2.2. MRED

MRED is mean value of the EDs for all possible inputs relative to the corresponding *accurate output* (AcO), which is expressed by Ref. [30]:

$$\mathrm{MRED} = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} \frac{|ED_i|}{|AcO_i|} \tag{9}$$

where AcO_i is accurate output of the multiplier for ith input combination. From Table 2, the MRED of proposed R8ABM1 increased from 1.227×10^{-2} with AC = 12 to 386.4×10^{-2} with AC = 20. The MRED of proposed R8ABM2 increased from 3.416×10^{-2} with AC = 12 to 572.2×10^{-2} with AC = 20. Similarly, MRED of proposed R8ABM3 is also increased from 1.774×10^{-2} with AC = 12 to 471.23×10^{-2} with AC = 20.

5.2.3. Pass rate

Pass rate [20] is the percentage of number of correct outputs produced by an approximate circuit with respect to the total number of outputs.

$$\text{Pass rate}(\%) = \frac{\text{Number of correct outputs}}{\text{Total number of outputs}} \times 100 \tag{10}$$

The pass rate of approximate multiplier increases with increase in number of correct outputs produced by the multiplier. The proposed R8ABE1 introduces error when $Y_j = \pm 3$, where as R8ABE2 introduces error when $Y_j = \pm 3$ and ± 4. Therefore, the overall probability for the output of the proposed R8ABM1 to become erroneous is less than that of the proposed R8ABM2 and R8ABM3. As shown in Table 2, the pass rates of R8ABM1, R8ABM2 and R8ABM3 are 54.81%, 40.15% and 51.15% respectively with AC = 12. Similarly, the pass rates of R8ABM1, R8ABM2 and R8ABM3 are 33.56%, 19.46% and 27.22% respectively with AC = 20.

5.2.4. P_{RED}

P_{RED} [17,20] indicates percentage of approximate outputs which are equal to accurate output or having a relative error < 2%, which is given by the equation:

$$P_{RED} = \frac{\text{Number of outputs with RED } < \text{ 2\%}}{\text{Total number of outputs}} \times 100 \tag{11}$$

where *relative error distance* (RED) of output for *i*th input combination is given by $|ED_i/AcO_i|$. From Table 2, the P_{RED} of R8ABM1, R8ABM2 and R8ABM3 are 99.54%, 99.01% and 99.41% respectively with AC = 12, where as 85.64%, 80.37% and 83.27% respectively with AC = 20.

From Table 2, it can be observed that increase in number of columns (AC) that contains approximate partial products reduce the hardware complexity but also reduce the accuracy of the output. Out of the three proposed designs, R8ABM3 shows better trade-off for both electrical performance and accuracy. To optimize the electrical performance and accuracy of N-bit approximate multipliers, AC = N is considered in the literature [17,18,29,31]. Therefore, the performance of the proposed approximate Booth multipliers are compared with existing approximate Booth multipliers by considering AC = N in the following subsection.

5.3. Comparison with existing multipliers

The three proposed radix-8 Booth multipliers along with conventional radix-4 Booth multiplier, conventional radix-8 Booth multiplier, existing approximate radix-4 [17,18], radix-8 [20] and hybrid high radix [19] Booth multipliers are simulated with N = 16. In the proposed multipliers, approximate partial products are introduced into 16 LSB columns (AC = 16) of partial product matrix. The existing radix-4 approximate multipliers R4ABM1, R4ABM2 [17], and ABM-M1 [18] approximate 4, 8 and 4 entries respectively in the K-map that generates partial products. For better comparison, approximate partial products are introduced into 16 LSB columns in these multipliers. The radix-4 ABM-M2 proposed in Ref. [18] generates only one approximate partial product corresponding to 8 LSBs in every partial product row. Moreover, ABM-M3 [18] generates approximate partial product in only 16th column corresponding to all the partial products which are less significant than 16th bit position of partial product matrix. The hybrid high radix multiplier proposed in Refs. [19] encodes r LSBs of the input operand with radix-2^r encoding and remaining MSBs with radix-4 encoding. Based on this criteria, three approximate multipliers RAD64, RAD256 and RAD1024 are proposed in Ref. [19] by encoding 6, 8 and 10 LSBs of the input operand with approximate radix-64, radix-256 and radix-1024 respectively. In case of radix-8 approximate Booth multiplier (ABM) proposed in Ref. [20], 8 LSBs of each recoding adder corresponding to each partial product row are considered to be approximate. The simulation results of proposed approximate multipliers along with all the considered existing approximate multipliers are shown in Table 3.

As compared to most of the existing approximate Booth multipliers, the proposed R8ABM1, R8ABM2 and R8ABM3 multipliers achieved better accuracy. The proposed multipliers produce output with less NMED compared to all other multipliers. The MRED and P_{RED} of the approximate radix-8 Booth multiplier in Ref. [20] are better than the proposed multipliers. It is observed that this multiplier is producing more number of correct outputs there by achieving highest pass rate among the considered approximate multipliers. The elimination of hard multiple 3X in proposed radix-8 multipliers significantly reduced the critical path delay while maintaining high accuracy at the output. The power consumption and area of ABM-M2 and ABM-M3 are less than the proposed

Table 3
Performance comparison of proposed multipliers with conventional and existing approximate Booth multipliers.

Multiplier Design	Power (μW)	Delay (ns)	PDP (pJ)	Area (μm^2)	NMED ($\times 10^{-5}$)	MRED ($\times 10^{-2}$)	Pass rate (%)	P_{RED} (%)
Conventional radix-4	614.0	1.04	0.639	2656	0	0	100.0	100.0
R4ABM1 [17]	479.9	0.94	0.451	2066	3.103	134.72	6.02	90.61
R4ABM2 [17]	448.7	0.92	0.412	1875	3.017	586.11	0.14	86.07
ABM-M1 [18]	472.5	0.94	0.444	1994	2.726	96.172	5.87	91.35
ABM-M2 [18]	408.2	0.74	0.302	1632	103.4	842.16	0.06	80.14
ABM-M3 [18]	421.4	0.79	0.335	1560	8.641	157.46	0.12	88.02
RAD64 [19]	482.6	0.92	0.445	1894	2.342	52.26	8.44	92.42
RAD256 [19]	401.4	0.81	0.325	1564	6.667	122.37	1.52	90.53
RAD1024 [19]	387.8	0.68	0.264	1492	52.176	514.72	0.36	85.71
Conventional radix-8	586.4	0.92	0.541	2486	0	0	100.0	100.0
Radix-8 ABM [20]	557.6	0.85	0.474	2236	1.922	3.126	55.94	99.72
Proposed R8ABM1	485.6	0.74	0.361	1924	1.430	14.264	47.22	97.16
Proposed R8ABM2	442.1	0.70	0.310	1715	1.664	22.512	32.14	95.23
Proposed R8ABM3	470.6	0.73	0.343	1852	1.512	17.151	41.94	96.48

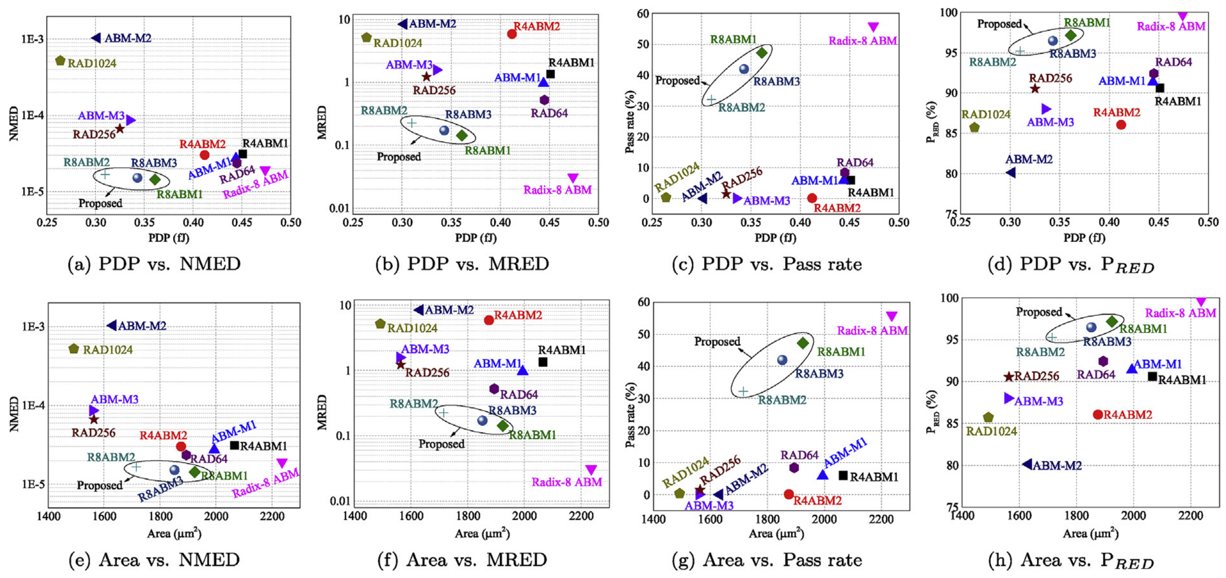

Fig. 7. Hardware versus Error metrics of approximate multipliers.

multipliers but their pass rate and P_{RED} are remarkably low due to truncation of LSBs in the partial product rows. The approximate high radix encoding of LSBs in RAD256 and RAD1024 reduced the power consumption and layout area but increased the NMED and MRED due to propagation of errors into MSBs.

As there is a trade-off between electrical performance, hardware complexity and accuracy of approximate multipliers, hardware verses error metrics graphs are plotted as shown in Fig. 7. The NMED and MRED values are plotted on log scale due to the exponential difference in values. For any considered approximate multiplier, the values of PDP, Area, NMED, and MRED should be as less as possible, where as Pass rate and P_{RED} should be as high as possible. From Fig. 7, the proposed multipliers performs better than the other approximate Booth multipliers in terms of both hardware and accuracy. Even though the multiplier in Ref. [20] performs well in terms of accuracy, it consumes significantly higher PDP and Area compared to proposed multipliers.

6. Case studies

In this section, the effectiveness of proposed 16-bit approximate radix-8 Booth multipliers along with considered multipliers from literature are tested with image processing and deep learning applications.

6.1. JPEG compression

Joint Photographic Experts Group (JPEG) standard is a very well known ISO standard used for lossy image compression. JPEG compression eliminates the least important information in a given input image and reduce the file size of the image. Discrete Cosine Transform (DCT) [39] is a popular linear transform used in JPEG compression to compress the sparse image data into smaller number of coefficients. The compression process starts with rearranging the input data. For an RGB image with 16-bit channel width, each pixel takes a value in the range [0,65535]. Initially, the input pixel data is made to be symmetrical around zero by subtracting 32768 from each value. The resulting pixel array is divided into blocks of 8 × 8 pixels. Then DCT is applied on each block, which is essentially a matrix multiplication of two matrices of sizes 8 × 8. Therefore, a total of 512 multiplications are required to compute DCT of each block.

To analyze the efficacy of approximate multipliers, matrix multiplication operation is implemented with 16-bit approximate multipliers and used to apply DCT transformation in JPEG compression. The proposed multipliers with AC = 16 and the existing multipliers with features as discussed in Section 5.3 are considered for this analysis. An input image with 16-bit channel width is considered from MIT Adobe FiveK Dataset [40], which is shown in Fig. 8a. The result of JPEG compression with accurate multiplier and three proposed approximate multipliers are shown in Fig. 8b–c. The quality degradation in a compressed image generated with approximate multipliers is difficult to recognize by human eye due to persistence of vision. Therefore, the accuracy of approximate output images compared to accurate output image is estimated with well-known metrics such as peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [41]. The PSNR (in decibels, dB) is defined as:

$$\mathrm{PSNR} = 10\,\log_{10}\left(\frac{\mathrm{MP}^2}{\mathrm{MSE}}\right) \tag{12}$$

where MP is maximum possible pixel value in the image. The Mean Square Error (MSE) of $p \times q$ pixels accurate output image (X) and

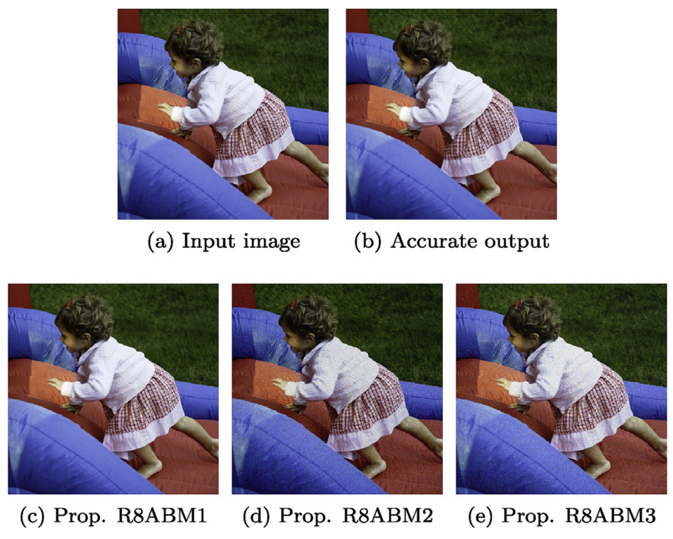

Fig. 8. Image compression with proposed multipliers.

Table 4
Accuracy metrics for image compression and HDR applications.

Multiplier	PSNR (dB)	SSIM	HDR Accuracy (%)
Prop. R8ABM1	65.82	0.996	97.32
Prop. R8ABM2	59.24	0.988	96.84
Prop. R8ABM3	63.76	0.993	97.18
R4ABM1 [17]	57.04	0.977	95.71
R4ABM2 [17]	52.82	0.969	95.68
ABM-M1 [18]	59.33	0.982	95.83
ABM-M2 [18]	48.75	0.956	93.68
ABM-M3 [18]	55.11	0.971	95.07
RAD64 [19]	57.66	0.980	95.81
RAD256 [19]	55.34	0.964	94.76
RAD1024 [19]	50.25	0.948	93.41
Radix-8 ABM [20]	65.91	0.996	97.36

approximate output image (Y) is given by:

$$\text{MSE} = \frac{1}{pq}\sum_{i=0}^{p-1}\sum_{j=0}^{q-1}[X(i,j) - Y(i,j)]^2 \tag{13}$$

The SSIM [41] parameter is given by:

$$\text{SSIM} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{14}$$

Where μ_x and σ_x are mean and variance of accurate output image pixels respectively. Similarly, μ_y and σ_y are mean and variance of approximate output image pixels respectively. σ_{xy} represents correlation coefficient. C_1 and C_2 are constants used to avoid instabilities. Table 4 shows PSNR and SSIM of the compressed images using proposed and other existing approximate Booth multipliers. It can be observed that the proposed multipliers along with multiplier in Refs. [20] are producing output images with highest PSNR and SSIM.

6.2. Handwritten digit recognition (HDR)

To verify the effectiveness of proposed multipliers in deep learning applications, the Convolution Neural Network (CNN) based LeNet-5 [42] model is implemented for handwritten digit recognition (HDR). The LeNet model is shown in Fig. 9, which contains two CNN layers and two sub-sampling (average pool) layers for feature mapping and three fully connected networks for classification. The model is trained and validated with the MNIST database [43], which contains 60,000 examples of 28 × 28 pixel handwritten digit images for training. These images are zero-padded to obtain the 30 × 30 pixel input images. The model is trained with 10 epochs and 128 batch size to achieve a test accuracy of 98.92%. Similar to Fig. 9, in most of the machine learning and deep learning applications, the last layer is a softmax activation function that converts input values to probability distributions in range of 0–1. The highest input values to softmax layer get convert to highest probabilities. The output of the softmax layer with highest probability is used to predict the final output of the model. Therefore, the input values to the last softmax layer need not be necessarily exact.

For inference, all the matrix multiplications and convolutions in the model are carried out with approximate multipliers in INT16 precision. The trained weights and biases are scaled to 16-bit integer values. The proposed approximate Booth multiplier along with approximate multipliers from literature are tested on LeNet-5 with 1000 random test images from the MNIST testing data set. The test accuracy of HDR application implemented using approximate multipliers are shown in Table 4. It can be observed that the proposed multipliers along with multiplier in Ref. [20] are achieving highest accuracy. Even though multiplier in Ref. [20] is performing similar to proposed multipliers in both the case studies, it is consuming significantly more PDP and area as compared to proposed multipliers.

7. Conclusion

This paper has proposed three designs of approximate radix-8 Booth multiplier using two novel designs of approximate radix-8 Booth encoders. The proposed multipliers along with conventional and other existing approximate Booth multipliers are simulated for power, delay and area. The results indicated that the proposed multipliers achieved best optimized performance for both hardware and error metrics. Finally, the effectiveness of proposed multipliers are verified with image compression and handwritten digit recognition applications. The proposed multipliers have produced outputs with highest values for accuracy metrics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Bipul Boro: Investigation, Validation, Visualization. **K. Manikantta Reddy:** Conceptualization, Methodology, Software, Investigation, Visualization, Writing - original draft, Writing - review & editing. **Y.B. Nithin Kumar:** Supervision, Project administration, Funding acquisition. **M.H. Vasantha:** Supervision, Project administration.

Acknowledgement

The authors would like to thank Ministry of Electronics and Information Technology, Government of India and Digital India Corporation for funding this work under Visvesvaraya Ph.D scheme.

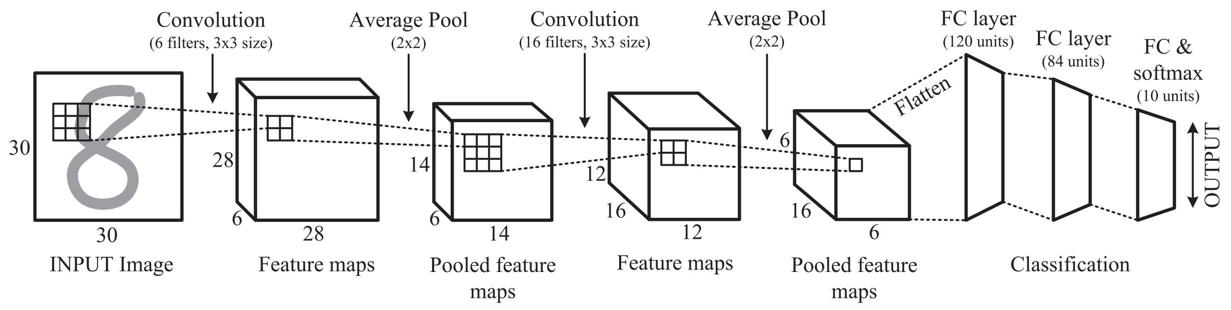

Fig. 9. LeNet-5 architecture [42] with two CNN, two average pool, and three fully connected layers.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.mejo.2020.104816.

References

[1] V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, Low-power digital signal processing using approximate adders, IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 32 (1) (Jan. 2013) 124–137.

[2] S. Mazahir, O. Hasan, M. Shafique, Self-compensating accelerators for efficient approximate computing, Microelectron. J. 88 (Jun. 2019) 9–17.

[3] B. Liu, Z. Wang, S. Guo, H. Yu, Y. Gong, J. Yang, L. Shi, An energy-efficient voice activity detector using deep neural networks and approximate computing, Microelectron. J. 87 (May 2019) 12–21.

[4] S. Hashemi, H. Tann, F. Buttafuoco, S. Reda, Approximate computing for biometric security systems: a case study on Iris scanning, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2018, pp. 319–324.

[5] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, J. Henkel, J. Henkel, Architectural-space exploration of approximate multipliers, in: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, 2016, pp. 1–8.

[6] V. Mrazek, M.A. Hanif, Z. Vasicek, L. Sekanina, M. Shafique, autoAx: an automatic design space exploration and circuit building methodology utilizing libraries of approximate components, in: Proc. of the 56th Annual Design Automation Conference (DAC), Las Vegas, 2019, pp. 1–6.

[7] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, J. Henkel, Invited: cross-layer approximate computing: from logic to architectures, in: Proc. of the 53rd Annual Design Automation Conference (DAC), Austin, TX, 2016, pp. 1–6.

[8] V. Mrazek, R. Hrbacek, Z. Vasicek, L. Sekanina, EvoApprox8b: library of approximate adders and multipliers for circuit design and benchmarking of approximation methods, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, 2017, pp. 258–261.

[9] A.D. Booth, A signed binary multiplication technique, Q. J. Mech. Appl. Math. 4 (2) (1951) 236–240.

[10] A.A. Del Barrio, R. Hermida, S. Ogrenci-Memik, A combined arithmetic-high-level synthesis solution to deploy partial carry-save radix-8 booth multipliers in datapaths, IEEE Trans. Circuits Syst. I: Reg. Pap. 66 (2) (Feb. 2019) 742–755.

[11] K.Y. Kyaw, W.L. Goh, K.S. Yeo, Low-power high-speed multiplier for error-tolerant application, in: IEEE Int. Conf. of Electron Devices and Solid-State Circuits (EDSSC), Hong Kong, 2010, pp. 1–4.

[12] S. Hashemi, R.I. Bahar, S. Reda, DRUM: a dynamic range unbiased multiplier for approximate applications, in: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, 2015, pp. 418–425.

[13] J.Y.L. Low, C.C. Jong, Unified Mitchell-based approximation for efficient logarithmic conversion circuit, IEEE Trans. Comput. 64 (6) (June 2015) 1783–1797.

[14] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, F. Lombardi, Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications, IEEE Trans. Circ. Syst. I: Reg. Pap. 65 (9) (Sept. 2018) 2856–2868.

[15] H. Saadat, H. Bokhari, S. Parameswaran, Minimally biased multipliers for approximate integer and floating-point multiplication, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 37 (11) (Nov. 2018) 2623–2635.

[16] M. Loukrakpam, M. Choudhury, Implementation of energy-efficient approximate multiplier with guaranteed worst case relative error, Microelectron. J. 88 (Jun. 2019) 1–8.

[17] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, F. Lombardi, Design of approximate radix-4 booth multipliers for error-tolerant computing, IEEE Trans. Comput. 66 (8) (Aug. 2017) 1435–1441.

[18] S. Venkatachalam, E. Adams, H.J. Lee, S. Ko, Design and analysis of area and power efficient approximate booth multipliers, IEEE Trans. Comput. 68 (11) (Nov. 2019) 1697–1703.

[19] V. Leon, G. Zervakis, D. Soudris, K. Pekmestzi, Approximate hybrid high radix encoding for energy-efficient inexact multipliers, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 26 (3) (Mar. 2018) 421–430.

[20] H. Jiang, J. Han, F. Qiao, F. Lombardi, Approximate radix-8 booth multipliers for low-power and high-performance operation, IEEE Trans. Comput. 65 (8) (Aug. 2016) 2638–2644.

[21] K.J. Cho, K.C. Lee, J.G. Chung, K.K. Parhi, Design of low-error fixed-width modified booth multiplier, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 12 (5) (May 2004) 522–531.

[22] J. Wang, S. Kuang, S. Liang, High-accuracy fixed-width modified booth multipliers for lossy applications, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 19 (1) (Jan. 2011) 52–60.

[23] Y. Chen, T. Chang, A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers, IEEE Trans. Circ. Syst. I: Reg. Pap. 59 (3) (Mar. 2012) 594–603.

[24] S. Ullah, et al., Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators, in: 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, 2018, pp. 1–6.

[25] W. Xu, S.S. Sapatnekar, J. Hu, A simple yet efficient accuracy-configurable adder design, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 26 (6) (Jun. 2018) 1112–1125.

[26] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, Block-based carry speculative approximate adder for energy-efficient applications, in: IEEE Trans. on Circuits and Systems II: Express Briefs, 1st, 67, 137–141, Jan. 2020.

[27] Li Luo, Zhe Chen, Xinghua Yang, Fei Qiao, Wei Qi, Huazhong Yang, A single clock cycle approximate adder with hybrid prediction and error compensation methods, Microelectronics J. 87 (May 2019) 45–50.

[28] Bharat Garg, Sunil Dutt, G.K. Sharma, Bit-width-aware constant-delay run-time Accuracy Programmable Adder for error-resilient applications, Microelectronics J. 50 (Apr. 2016) 1–7.

[29] A. Momeni, J. Han, P. Montuschi, F. Lombardi, Design and analysis of approximate compressors for multiplication, IEEE Trans. Comput. 64 (4) (Apr. 2015) 984–994.

[30] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 25 (4) (Apr. 2017) 1352–1361.

[31] S. Venkatachalam, S. Ko, Design of power and area efficient approximate multipliers, IEEE Trans. Very Large Scale Integrat. (VLSI) Syst. 25 (5) (May 2017) 1782–1786.

[32] M. Ha, S. Lee, Multipliers with approximate 4-2 compressors and error recovery modules, IEEE Embedded Syst. Lett. 10 (1) (Mar. 2018) 6–9.

[33] M.S. Ansari, H. Jiang, B.F. Cockburn, J. Han, Low-power approximate multipliers using encoded partial products and approximate compressors, IEEE J. Emerg. Sel. Top. Circ. Syst. 8 (3) (Sept. 2018) 404–416.

[34] K.M. Reddy, M.H. Vasantha, Y.B. Nithin Kumar, D. Dwivedi, Design and analysis of multiplier using approximate 4-2 compressor, AEU - Int. J. Electron. Commun. 107 (Jul. 2019) 89–97.

[35] G.A. Gillani, M.A. Hanif, M. Krone, S.H. Gerez, M. Shafique, A.B.J. Kokkeler, SquASH: approximate square-accumulate with self-healing, IEEE Access 6 (2018) 49112–49128.

[36] G.A. Gillani, M.A. Hanif, B. Verstoep, S.H. Gerez, M. Shafique, A.B.J. Kokkeler, MACISH: designing approximate MAC accelerators with internal-self-healing, IEEE Access 7 (2019) 77142–77160.

[37] C. Chang, J. Gu, M. Zhang, Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits, IEEE Trans. Circ.Syst. 51 (10) (Oct. 2004) 1985–1997.

[38] J. Liang, J. Han, F. Lombardi, New metrics for the reliability of approximate and probabilistic adders, IEEE Trans. Comput. 62 (9) (Sept. 2013) 1760–1771.

[39] N. Ahmed, T. Natarajan, K.R. Rao, Discrete cosine transform, IEEE Trans. Comput. C-23 (1) (Jan. 1974) 90–93.

[40] V. Bychkovsky, S. Paris, E. Chan, F. Durand, Learning photographic global tonal adjustment with a database of input/output image pairs, in: Proc. 24th IEEE Conf. on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 2011, pp. 97–104.

[41] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (Apr. 2004) 600–612.

[42] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (Nov. 1998) 2278–2324.

[43] Y. Lecun, C. Cortes, C.J.C. Burges, The MNIST Database of Handwritten Digits, 2010. [Online] Available: http://yann.lecun.com/exdb/mnist/.