Design of Majority Logic-based Approximate Booth Multipliers for Error-Tolerant Applications

Tingting Zhang, *Student Member, IEEE,* Honglan Jiang, *Member, IEEE,* Hai Mo, Weiqiang Liu, *Senior Member, IEEE,* Fabrizio Lombardi, *Life Fellow, IEEE,* Leibo Liu, *Senior Member, IEEE,* and Jie Han, *Senior Member, IEEE*

***Abstract*—Approximate computing at the nanoscale provides sufficiently accurate and often adaptive results to improve hardware efficiency for error-tolerant applications. Differently from conventional Boolean logic-based designs, many emerging nanotechnologies extensively assemble circuits using the voter-based majority logic (ML). In this paper, we investigate designs of approximate radix-4 Booth multipliers based on ML. Initially, we propose two new radix-4 Booth partial product (PP) generation methods by exploiting the characteristics of ML. Based on these methods, approximate PP generators are designed to produce single-sided or double-sided errors. The PPs are then reduced by using the features of errors to construct approximate multipliers. Specifically, complementary strategies guided by an analysis of error effects are developed to compensate for the accuracy loss and to reduce the hardware overhead during the PP reduction. The reduced PPs are then compressed by using full adders. Four approximate multipliers are proposed to offer various accuracy requirements for different applications. These designs show superior performance in power and area for emerging nanotechnologies. As case studies, image processing, a multiple-layer perceptron and a multi-task convolutional neural network are presented to show the validity and advantages of the proposed designs.**

***Index Terms*—Booth encoding, partial product generator, approximate multiplier, majority logic, nanotechnologies**

I. Introduction

As alternatives to CMOS, emerging nanotechnologies have been considered for digital circuit design by exploiting their characteristics of high density and low energy dissipation. Substantially different from conventional Boolean logic, these nanotechnologies are often based on majority logic (ML) [1], such as nanomagnetic logic (NML) [2], and magnetic tunnel junction (MTJ) devices [3]. An ML function uses an odd number of inputs; the logic expression of a 3-input majority gate is given by $M(A, B, C) = AB + BC + AC$. Due to its ability of realizing more complex functions, fewer gates are required when building circuits using majority gates.

As an emerging paradigm of low-power design, approximate computing can provide a reduction in hardware at a reasonable accuracy loss without significant negative effects for error-tolerant applications, such as neural networks (NNs). A major hurdle in NNs is the energy overhead; so, approximate computing can be used, especially for multiplication, which is one of the most hardware intensive arithmetic operations [4].

Since both emerging nanotechnologies and approximate computing benefit from low power and high performance, approximate designs based on these nanotechnologies can be symbiotically devised [5]. Unlike approximate designs for CMOS circuits, approximate ML-based designs have not been extensively studied. A direct mapping of previous approximate designs to ML-based nanotechnologies usually does not fully utilize the specific properties of ML; approximate designs for ML must be adapted for these new technologies.

Prior research includes the designs of approximate full adders (AFAs) [6-7], approximate compressors for multipliers [5, 8-9], array multipliers using approximate partial product (PP) generation and compressors [10], and an approximate PP generator (PPG) based on the modified Booth encoding (MBE) with single-sided errors [11]. Only unsigned approximate multiplier designs are considered in [10]. Moreover, approximate circuits with single-sided errors [11] lead to catastrophic results in some applications with accumulative operations, including NNs [4].

For approximate designs, the errors that make the approximate results always smaller or larger than the exact results are referred to as negative or positive single-sided errors. Otherwise, whether the number of negative errors equal to that of positive errors or not for a particular design, determines if the errors are called unbiased or biased double-sided errors.

In this paper, radix-4 Booth multipliers based on ML are designed with approximate PP generation, reduction, and compression. Initially, the ML implementations of two new MBE-based PPGs are proposed. Then, different approximate PPGs are designed: some produce negative or positive single-sided errors and others produce biased or unbiased double-sided errors. In the PP reduction, complementary strategies are considered

Manuscript received June 15, 2021; revised September 12, 2021; 1; accepted January 12, 2022. (Corresponding authors: Weiqiang Liu and Jie Han.)

T. Zhang and J. Han are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (email: ttzhang@ualberta.ca, jhan8@ualberta.ca).

H. Jiang is with the Department of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: honglan@sjtu.edu.cn).

H. Mo and L. Liu are with the School of Integrated Circuits, Tsinghua University, Beijing 100084, China (e-mail: moh19@mails.tsinghua.edu.cn; liulb@tsinghua.edu.cn).

W. Liu is with College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: liuweiqiang@nuaa.edu.cn).

F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston MA 02115, USA (e-mail: lombardi@ece.neu.edu).

by analyzing the properties of the introduced errors to control accuracy. To assess the proposed approximate multipliers, we perform an analytical investigation of hardware complexity, errors, and the estimate of power dissipation based on nanotechnologies. Applications with various accuracy requirements are also provided to show the validity of the proposed designs.

This paper is a significant extension of [11]. The contributions of this paper are as follows:

(1) PP generation: Based on the MBE, in addition to the classical PP generator (CPPG) [11], the new PPG (NPPG) [12] and two newly proposed PPGs are considered for efficient ML implementations.

(2) Approximate PPGs: In addition to the approximate PPG with positive single-sided errors in [11], seven other approximate PPGs are designed based on various Booth PP generation methods by introducing errors towards different directions.

(3) PP reduction and compression: The PPs are reduced by using complementary strategies based on different error features and then efficiently compressed.

(4) Various applications: Apart from the image multiplication considered in [11], the proposed multipliers at different levels of accuracy are applied to edge detection, classification, joint face detection and alignment.

The remainder of this paper is organized as follows. Section II reviews the preliminaries. Section III proposes new PP generation methods and discusses the ML implementations. Section IV presents the approximate PPG designs. Section V and Section VI discuss the PP reduction and compression strategies, and then evaluate the required hardware. Applications are presented in Section VII. Section VIII concludes the paper.

II. Preliminaries

A. Radix-4 Booth Encoding

The modified Booth algorithm (also known as the radix-4 Booth algorithm) [13] has been widely utilized to solve the sign correction issue of signed multiplication and reduce the number of PPs. For an $n \times n$ multiplier, let $A = a_{n-1}a_{n-2}...a_2a_1a_0$ denote the multiplicand and $B = b_{n-1}b_{n-2}...b_2b_1b_0$ denote the multiplier. The most significant bits of A and B are the sign bits. In the MBE, the multiplier bits are first grouped into sets of three adjacent bits and the bits on the two sides overlap with the neighboring sets, except for the first set. Then, the multiplicand is encoded into $-2A$, $-A$, 0, A, or $2A$ to generate the PP array, as per Table I [13], where $0 \leq i \leq \frac{n}{2} - 1$, $0 \leq j \leq n - 1$. As the first step in a Booth multiplier design, the PP generation plays an important role in the overall circuit design. Compared with the naive encoding and shift operations, a dedicated PPG can be more efficient in performance and power consumption. The CPPG is a common implementation of the MBE [14]. Let pp_{ij} be the PP bit in the ith row and the jth column, as

$$\begin{aligned} pp_{ij} &= (b_{2i} \oplus b_{2i-1})(b_{2i+1} \oplus a_j) \\ &\quad + \overline{(b_{2i} \oplus b_{2i-1})}(b_{2i+1} \oplus b_{2i})(b_{2i+1} \oplus a_{j-1}). \quad (1) \end{aligned}$$

The negation operation is performed by inverting every bit of

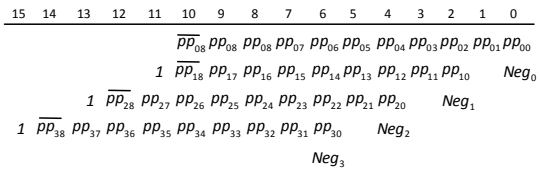


Fig. 1: PP and sign bit generation for an 8-bit Booth multiplier [13].

A (realized by pp_{ij}) and adding a '1' to the LSB (implemented by Neg_i, given by (2)) to get the 2's complement.

$$Neg_i = b_{2i+1}\overline{b}_{2i} + b_{2i+1}\overline{b}_{2i-1}. \quad (2)$$

The Baugh Wooley algorithm is used to avoid the sign bit extension. All Booth PP generation methods discussed here are developed based on the structure of Fig. 1 [13].

Based on the MBE, the NPPG modifies the PP "0" when $b_{2i+1}b_{2i}b_{2i-1} = 111$ in Table I by performing the negation operation on it: pp_{ij} is inverted to '1'and Neg_i is '1' [12]. This enables a simplified Neg_i by directly using b_{2i+1} as

$$Neg_i = b_{2i+1}, \quad (3)$$

with a slight increase in the complexity for pp_{ij} as

$$\begin{aligned} pp_{ij} &= (b_{2i} \oplus b_{2i-1})(b_{2i+1} \oplus a_j) + b_{2i+1}b_{2i}b_{2i-1} \\ &\quad + \overline{(b_{2i} \oplus b_{2i-1})}(b_{2i+1} \oplus b_{2i})(b_{2i+1} \oplus a_{j-1}). \quad (4) \end{aligned}$$

B. Error Metrics

We consider error metrics to evaluate the approximate designs [15]. The error rate (ER) is the percentage of input assignments that result in errors. The error distance (ED) is defined as the absolute difference between the approximate and the exact results, while the relative error distance (RED) is the difference divided by the exact result. The mean error distance (MED) and mean relative error distance (MRED) are the averages of the ED and the RED, respectively. The NMED is the normalized MED by the maximum exact result.

C. Transformation of Majority Logic Functions

Shannon's decomposition can be used to reformulate a logic function for simplifying an ML-based expression [16], by

$$F = xF_x + \overline{x}F_{\overline{x}}. \quad (5)$$

TABLE I: Implementations of the MBE Scheme using CPPG [14] and NPPG [12]

$b_{2i+1}b_{2i}b_{2i-1}$	PP	CPPG [14]		NPPG [12]	
		pp_{ij}	Neg_i	pp_{ij}	Neg_i
000	0	0	0	0	0
001	+A	a_j	0	a_j	0
010	+A	a_j	0	a_j	0
011	+2A	a_{j-1}	0	a_{j-1}	0
100	-2A	$\overline{a}_{j-1}$	1	$\overline{a}_{j-1}$	1
101	-A	$\overline{a}_j$	1	$\overline{a}_j$	1
110	-A	$\overline{a}_j$	1	$\overline{a}_j$	1
111	0	0	0	1	1

where F is a Boolean function of a variable x, $\overline{x}$ is the complement of x, and F_x and $F_{\overline{x}}$ are the positive and negative Shannon cofactors with x set to '1' and '0', respectively.

The following transformation rules are considered to simplify ML-based implementations using a reduced number of majority gates and a shorter critical path: inverter propagation $\Omega.I$. $M(\overline{x},\overline{y},\overline{z}) = \overline{M}(x,y,z)$, distributivity $\Omega.D$. $M(x,y,M(u,v,z)) = M(M(x,y,u),M(x,y,v),z)$ and complementary associativity $\Psi.C$. $M(x,y,M(y,\overline{u},z)) = M(x,u,M(y,x,z))$ [16]. In addition, if the appearance of the minterms or maxterms in the K-map looks like the letter "T", it is said to have a "T shape", which can be implemented by using a 3-input majority gate [17].

For the ease of reading, the acronyms of the PPG designs are listed in Table II.

III. Formulation and Design of ML-based PPGs

This section first formulates the CPPG and NPPG for ML implementations, referred to as MLCG and MLNG, respectively, and then discusses the ML implementations of two new PPGs based on the MBE, namely MLGA and MLGB.

A. ML Implementation of Partial Product Generators

1) MLCG: AND or OR gates can be directly mapped into ML by setting one input of the 3-input majority gate to '0' or '1'. pp_{ij} and Neg_i are obtained by using sixteen and three majority gates, respectively [11]. However, such a naive mapping does not fully leverage the capabilities of ML. Initially, the expression of pp_{ij} can be divided into two sections using Shannon's decomposition at b_{2i+1} [16]. There is a T shape for the maxterms (eight out of ten cases) in the K-map when $b_{2i+1}=0$ and 1, respectively (see Table A1 [14] in the supplementary document; the same for other tables indexed with "A" unless otherwise noted). The terms in T shapes can be implemented by using $\overline{M}(b_{2i},b_{2i-1},a_j)$ for b_{2i+1} and $M(b_{2i},b_{2i-1},a_j)$ for $\overline{b}_{2i+1}$, as

$$\begin{aligned} pp_{ij} &= \left\{b_{2i+1}\overline{M}(b_{2i},b_{2i-1},a_j)\right\}(b_{2i}+b_{2i-1}+\overline{a}_{j-1}) + \\ &\quad \left\{\overline{b}_{2i+1}M(b_{2i},b_{2i-1},a_j)\right\}(\overline{b}_{2i}+\overline{b}_{2i-1}+a_{j-1}). \end{aligned} \quad (6)$$

TABLE II: Acronyms of Partial Product Generator Designs

Category	Acronyms	Meaning
Boolean logic based PPGs	CPPG	classical partial product generator in [14]
	NPPG	new partial product generator in [12]
Majority logic based PPG designs	MLCG	majority logic based classical partial product generator design
	MLNG	majority logic based new partial product generator design
	MLGA	majority logic based partial product generator design A
	MLGB	majority logic based partial product generator design B
Approximate majority logic based PPG designs	AMLCG	approximate majority logic based classical partial product generator design
	AMLNG	approximate majority logic based new partial product generator design
	AMLGA	approximate majority logic based partial product generator type A
	AMLGB	approximate majority logic based partial product generator type B

Consider $\Omega.D$, Neg_i can be simplified to

$$\begin{aligned} Neg_i &= M(M(b_{2i+1},\overline{b}_{2i},0),M(b_{2i+1},\overline{b}_{2i-1},0),1) \\ &= M(b_{2i+1},0,M(\overline{b}_{2i},\overline{b}_{2i-1},1)). \end{aligned} \quad (7)$$

Consider $\Omega.I$, when $j=0$, $\overline{M}(b_{2i},b_{2i-1},a_j)$ in (6) is equal to $M(\overline{b}_{2i},\overline{b}_{2i-1},1)$ that can be reused in (7).

2) MLNG: MLNG can be interpreted as a variant of MLCG. Neg_i is given by (3). For pp_{ij}, the minterms for MLNG are the same as for MLCG when $b_{2i+1}=0$; thus the expression is the same as (6). When $b_{2i+1}=1$, there is a T shape for eight out of ten minterms in the K-map (see Table A2 [12]) which can be implemented using $M(b_{2i},b_{2i-1},\overline{a}_j)$ as

$$\begin{aligned} pp_{ij} &= b_{2i+1}\left\{M(b_{2i},b_{2i-1},\overline{a}_j)+\overline{b}_{2i}\overline{b}_{2i-1}\overline{a}_{j-1}\right\} + \\ &\quad \left\{\overline{b}_{2i+1}M(b_{2i},b_{2i-1},a_j)\right\}(\overline{b}_{2i}+\overline{b}_{2i-1}+a_{j-1}) \end{aligned} \quad (8)$$

3) MLGA: Consider that PP is also encoded to "0" when $b_{2i+1}b_{2i}b_{2i-1}=000$ in Table I. Like MLNG, MLGA modifies the PP "0" when $b_{2i+1}b_{2i}b_{2i-1}=000$. The results of $b_{2i+1}b_{2i}b_{2i-1}=000$ are then changed to '1's (as per Tables A3 and A4) to realize the negation operation. For pp_{ij}, the expression is the same as (6) when $b_{2i+1}=1$ and $M(\overline{b}_{2i},\overline{b}_{2i-1},a_j)$ can cover eight out of ten minterms (see Table A3) when $b_{2i+1}=0$, as

$$\begin{aligned} pp_{ij} &= (b_{2i}\oplus b_{2i-1})(b_{2i+1}\oplus a_j)+\overline{b}_{2i+1}\overline{b}_{2i}\overline{b}_{2i-1} \\ &\quad +\overline{(b_{2i}\oplus b_{2i-1})}(b_{2i+1}\oplus b_{2i})(b_{2i+1}\oplus a_{j-1}) \\ &= \left\{b_{2i+1}M(\overline{b}_{2i},\overline{b}_{2i-1},\overline{a}_j)\right\}(b_{2i}+b_{2i-1}+\overline{a}_{j-1}) + \\ &\quad \left\{\overline{b}_{2i+1}M(\overline{b}_{2i},\overline{b}_{2i-1},a_j)+\overline{b}_{2i+1}b_{2i}b_{2i-1}a_{j-1}\right\} \end{aligned} \quad (9)$$

In this case, a T shape occurs for the minterms in the K-map of Neg_i (as per Table A4). Thus, Neg_i is given by

$$\begin{aligned} Neg_i &= \overline{b_{2i},\overline{b}_{2i+1}+b_{2i-1}\overline{b}_{2i+1}+b_{2i-1}b_{2i}} \\ &= \overline{M}(\overline{b}_{2i+1},b_{2i},b_{2i-1}) = M(b_{2i+1},\overline{b}_{2i},\overline{b}_{2i-1}) \end{aligned} \quad (10)$$

4) MLGB: MLGB can be considered as a combination of MLNG and MLGA to modify the PP "0" when $b_{2i+1}b_{2i}b_{2i-1}=000$ and 111 in Table I. Thus, the results in both cases are set to '1's (see Tables A5 and A6). As discussed above, pp_{ij} employs $b_{2i+1}M(b_{2i},b_{2i-1},\overline{a}_j)$ and $\overline{b}_{2i+1}\overline{M}(b_{2i},b_{2i-1},\overline{a}_j)$ to include sixteen out of all twenty true cases, as given by

$$\begin{aligned} pp_{ij} &= (b_{2i}\oplus b_{2i-1})(b_{2i+1}\oplus a_j)+\overline{(b_{2i}\oplus b_{2i-1})}(b_{2i+1}\oplus b_{2i}) \\ &\quad (b_{2i+1}\oplus a_{j-1})+\overline{b}_{2i+1}\overline{b}_{2i}\overline{b}_{2i-1}+b_{2i+1}b_{2i}b_{2i-1} \\ &= b_{2i+1}M(b_{2i},b_{2i-1},\overline{a}_j)+\overline{b}_{2i+1}\overline{M}(b_{2i},b_{2i-1},\overline{a}_j) + \\ &\quad \left\{\overline{b}_{2i}\overline{b}_{2i-1}\overline{a}_{j-1}+b_{2i}b_{2i-1}a_{j-1}\right\}. \end{aligned} \quad (11)$$

Consider $\Psi.C$, for Neg_i, the fully-utilized majority gate $M(b_{2i+1},\overline{b}_{2i},\overline{b}_{2i-1})$ is converted into a partially-utilized majority gate $M(\overline{b}_{2i},\overline{b}_{2i-1},0)$, as described in

$$\begin{aligned} Neg_i &= M(b_{2i+1},\overline{b}_{2i},\overline{b}_{2i-1})+b_{2i+1} \\ &= M(\overline{b}_{2i},\overline{b}_{2i-1},0)+b_{2i+1}. \end{aligned} \quad (12)$$

Moreover, consider $\Omega.I$, $M(\overline{b}_{2i},\overline{b}_{2i-1},0)$ in (12) is equal to $\overline{M}(b_{2i},b_{2i-1},\overline{a}_j)$ when $j=0$, which can be reused in (11).

TABLE III: Comparison of ML Implementations for Different Partial Product Generation Methods

PPGs	Equations	MV	INV	D
MLCG	$pp_{ij} = M(M(M(b_{2i+1}, 0, \overline{M}(b_{2i}, b_{2i-1}, a_j)), M(b_{2i}, 1, M(b_{2i-1}, \overline{a}_{j-1}, 1)), 0), M(\overline{b}_{2i+1}, 0, M(b_{2i}, b_{2i-1}, a_j)), M(a_{j-1}, 1, M(\overline{b}_{2i-1}, \overline{b}_{2i}, 1)), 0), 1)$	10	6	4
	$Neg_i = M(b_{2i+1}, 0, M(b_{2i}, \overline{b}_{2i-1}, 1))$	2	2	2
MLNG	$pp_{ij} = M(M(b_{2i+1}, M(M(b_{2i}, b_{2i-1}, \overline{a}_j), M(\overline{b}_{2i}, M(\overline{b}_{2i-1}, \overline{a}_{j-1}, 0), 0), 1), 0), M(M(\overline{b}_{2i+1}, M(b_{2i}, b_{2i-1}, a_j), 0), M(\overline{b}_{2i}, M(\overline{b}_{2i-1}, a_{j-1}, 1), 1), 0), 1)$	11	5	5
	$Neg_i = b_{2i+1}$	0	0	0
MLGA	$pp_{ij} = M(M(M(b_{2i+1}, 0, M(\overline{b}_{2i}, \overline{b}_{2i-1}, \overline{a}_j)), M(b_{2i}, 1, M(b_{2i-1}, \overline{a}_{j-1}, 1)), 0), M(M(\overline{b}_{2i+1}, 0, M(\overline{b}_{2i}, \overline{b}_{2i-1}, a_j)), M(M(\overline{b}_{2i+1}, b_{2i}, 0), M(b_{2i-1}, a_{j-1}, 0), 0), 1), 1)$	12	5	4
	$Neg_i = M(b_{2i+1}, \overline{b}_{2i}, \overline{b}_{2i-1})$	1	2	1
MLGB	$pp_{ij} = M(M(M(M(b_{2i}, b_{2i-1}, \overline{a}_j), b_{2i+1}, 0), M(\overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j), \overline{b}_{2i+1}, 0), 1), M(M(\overline{a}_{j-1}, M(\overline{b}_{2i-1}, \overline{b}_{2i}, 0), 0), M(b_{2i}, M(b_{2i-1}, a_{j-1}, 0), 0), 1), 1)$	10	6	4
	$Neg_i = M(b_{2i+1}, 1, M(\overline{b}_{2i}, \overline{b}_{2i-1}, 0))$	2	2	2

B. Comparison of Different Partial Product Generators

Table III presents the ML-based equations and the required logic gates for different PP generation methods evaluated by the number of utilized majority voters (MVs), the number of utilized inverters (INVs), and the critical path delay (D). The critical path delay is measured by the number of majority gates on the critical path since the delay for inverters is often very small for ML-based nanotechnologies [18].

All PP generation methods use the inversion of every input operand bit. Moreover, MLCG and MLGB use one more inverter when computing pp_{ij} due to the reuse of a majority gate. Compared with MLCG and MLGB, MLNG requires one more majority gate and thus, it incurs into a longer critical path to generate pp_{ij}; however, it does not need any logic gate to implement Neg_i. MLGA uses two more majority gates to generate pp_{ij} and one fewer majority gate to generate Neg_i.

For an $n \times n$ multiplier, $\frac{n}{2}$ Neg_i and approximately $\frac{n^2}{2}$ PPs are required. Therefore, the efficiency to generate pp_{ij} has a more significant effect on the multiplier design. Compared with MLNG and MLGA, although MLCG and MLGB have a higher complexity in generating Neg_i, they need fewer majority gates to implement pp_{ij} and a shorter critical path. Thus, MLCG and MLGB are more efficient for exact PP generation.

IV. Design of ML-based Approximate PPGs

A. Design of Approximate MLCG (AMLCG)

Initially, (1) can be divided into two parts as shown in (13)-(15): $sub1_{ij}$ relates to b_{2i+1} and $sub2_{ij}$ relates to $\overline{b}_{2i+1}$.

$$pp_{ij} = sub1_{ij} + sub2_{ij}, \tag{13}$$

$$sub1_{ij} = \left\{b_{2i+1}\overline{M}(b_{2i}, b_{2i-1}, a_j)\right\}(b_{2i} + b_{2i-1} + \overline{a}_{j-1}), \tag{14}$$

$$sub2_{ij} = \left\{\overline{b}_{2i+1}M(b_{2i}, b_{2i-1}, a_j)\right\}(\overline{b}_{2i} + \overline{b}_{2i-1} + a_{j-1}). \tag{15}$$

1) Introducing single-sided errors [11]: To simplify $sub1_{ij}$ and $sub2_{ij}$, (14) and (15) are approximately obtained by using only the terms in the curved brackets, resulting in four positive errors (see Table A7). An approximate pp_{ij} (app_{ij}) is given by

$$\begin{aligned} app_{ij} &= b_{2i+1}\overline{M}(b_{2i}, b_{2i-1}, a_j) + \overline{b}_{2i+1}M(b_{2i}, b_{2i-1}, a_j) \\ &= M(M(b_{2i+1}, \overline{M}(b_{2i}, b_{2i-1}, a_j), 0), M(\overline{b}_{2i+1}, \\ &\quad M(b_{2i}, b_{2i-1}, a_j), 0), 1). \end{aligned} \tag{16}$$

2) Introducing double-sided errors: If only single-sided errors are introduced, it leads to relatively large errors, especially used for building a large circuit. To reduce the number of required majority gates in the PPGs and to control errors, we consider leveraging different characteristics of the input operands and introduce unbiased double-sided errors. Therefore, two approximate PPGs are designed for pp_{ij}: the first one is for the PPs except the first row, denoted as $app1_{ij}$, where $1 \le i \le n-1$, $0 \le j \le \frac{n}{2} - 1$; the second one is for the PPs in the first row, denoted as $app2_{0j}$, where $0 \le j \le \frac{n}{2} - 1$.

For $app1_{ij}$, four unbiased errors are introduced (as per Table A8) by replacing $b_{2i} + b_{2i-1} + \overline{a}_{j-1}$ in (14) and $\overline{b}_{2i} + \overline{b}_{2i-1} + a_{j-1}$ in (15) with $\overline{a}_j$ and a_j, respectively. $app1_{ij}$ is given by

$$\begin{aligned} app1_{ij} = & \quad b_{2i+1}\overline{a}_j\overline{M}(b_{2i}, b_{2i-1}, a_j) \\ & + \overline{b}_{2i+1}a_jM(b_{2i}, b_{2i-1}, a_j) \\ = & \quad M(M(M(b_{2i+1}, \overline{a}_j, 0), \overline{M}(b_{2i}, b_{2i-1}, a_j), 0), \\ & M(M(\overline{b}_{2i+1}, a_j, 0), M(b_{2i}, b_{2i-1}, a_j), 0), 1). \end{aligned} \tag{17}$$

Consider $app2_{0j}$ ($j \neq 0$), b_{2i-1} is equal to 0. Using (17), two unbiased errors are introduced (as per Table A9). Consider $\Psi.C$, it is further simplified to

$$\begin{aligned} app2_{0j} = & \quad b_1\overline{a}_j\overline{M}(b_0, 0, a_j) + \overline{b}_1a_jM(b_0, 0, a_j) \\ = & \quad M(M(\overline{b}_1, M(b_0, 0, a_j), 0), M(b_1, \overline{a}_j, 0), 1). \end{aligned} \tag{18}$$

For app_{00}, (18) causes single-sided errors; therefore, (16) is used to generate an exact pp_{00}. app_{i0} ($i \neq 0$) is expressed by (17) with unbiased errors. Thus, (18), (16) and (17) are used to generate app_{0j} ($j \neq 0$), app_{00} and all other app_{ij}.

B. Design of Approximate MLNG (AMLNG)

Similarly, (8) is divided into two sections: $sub1_{ij}$ as

$$sub1_{ij} = b_{2i+1}\left\{M(b_{2i}, b_{2i-1}, \overline{a}_j) + \overline{b}_{2i}\overline{b}_{2i-1}\overline{a}_{j-1}\right\}, \tag{19}$$

and $sub2_{ij}$ as shown in (15).

Consider app_{ij}, by ignoring the second term in the curved bracket in (19), two negative errors are introduced, and $sub1_{ij}$ can be simplified to the first term in (20). To obtain the unbiased double-sided errors, two positive errors are further introduced as the second term in (20), similar to the approximation for the MLCG-based $sub2_{ij}$. Thus, by introducing four unbiased errors (as per Table A10), (8) is approximately simplified to

$$\begin{aligned} app_{ij} &= b_{2i+1}M(b_{2i}, b_{2i-1}, \overline{a}_j) + \overline{b}_{2i+1}M(b_{2i}, b_{2i-1}, a_j) \\ &= M(M(b_{2i+1}, M(b_{2i}, b_{2i-1}, \overline{a}_j), 0), M(\overline{b}_{2i+1}, \\ &\quad M(b_{2i}, b_{2i-1}, a_j), 0), 1). \end{aligned} \tag{20}$$

Consider app_{0j} ($j \neq 0$). Since the representation for the cases of $\overline{b}_1$ has no error using (20) when $i = 0$, two negative errors are introduced. The equation of MLNG-based pp_{0j} is the same as for MLCG-based pp_{0j}. To obtain unbiased double-sided errors, (18) is employed. Consider app_{00}, (18) or (20) causes single-sided errors. Therefore, (16) is used to generate pp_{00} with no error. Hence, (18), (16) and (20) are employed to generate app_{0j} ($j \neq 0$), app_{00} and all other app_{ij}.

C. Design of Approximate MLGA (AMLGA)

The MLGA-based pp_{ij} in (9) is divided into (14) and

$$sub2_{ij} = \overline{b}_{2i+1} M(\overline{b}_{2i}, \overline{b}_{2i-1}, a_j) + \overline{b}_{2i+1} b_{2i} b_{2i-1} a_{j-1}. \quad (21)$$

For $app1_{ij}$, similar to the AMLNG-based app_{ij}, two negative errors are introduced by disregarding the second term in (21); thus $sub2_{ij}$ is simplified to the second term in (22). $sub1_{ij}$ in (14) is approximately implemented by the first term in (22) with two positive errors. Thus, $app1_{ij}$ is implemented by (22) with four unbiased errors (as per Table A11).

$$\begin{aligned} app1_{ij} &= b_{2i+1} M(\overline{b}_{2i}, \overline{b}_{2i-1}, \overline{a}_j) + \overline{b}_{2i+1} M(\overline{b}_{2i}, \overline{b}_{2i-1}, a_j) \\ &= M(M(b_{2i+1}, M(\overline{b}_{2i}, \overline{b}_{2i-1}, \overline{a}_j), 0), M(\overline{b}_{2i+1}, \\ &\quad M(\overline{b}_{2i}, \overline{b}_{2i-1}, a_j), 0), 1). \end{aligned} \quad (22)$$

As discussed, using (22) to generate app_{0j} ($j \neq 0$) introduces single-sided errors. The AMLGA-based $app2_{0j}$ is expressed by (23) with two unbiased errors (as per Table A12).

$$\begin{aligned} app2_{0j} &= \overline{b}_1 M(\overline{b}_1, \overline{b}_0, a_j) + b_1 \overline{a}_j \\ &= M(M(\overline{b}_1, M(\overline{b}_1, \overline{b}_0, a_j), 0), M(b_1, \overline{a}_j, 0), 1). \end{aligned} \quad (23)$$

For app_{00}, (22) is applied with no error. Thus, (23) and (22) are employed to generate app_{0j} ($j \neq 0$) and all other app_{ij}.

D. Design of Approximate MLGB (AMLGB)

1) Introducing single-sided errors: $sub1_{ij}$ and $sub2_{ij}$ to generate pp_{ij} are given in (19) and (21), respectively.

By ignoring the second terms in (19) and (21), app_{ij} is generated by (24) with four negative errors (as per Table A13).

$$\begin{aligned} app_{ij} &= b_{2i+1} M(b_{2i}, b_{2i-1}, \overline{a}_j) + \overline{b}_{2i+1} \overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j) \\ &= M(M(b_{2i+1}, M(b_{2i}, b_{2i-1}, \overline{a}_j), 0), M(\overline{b}_{2i+1}, \\ &\quad \overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j), 0), 1). \end{aligned} \quad (24)$$

This scheme can be considered as a combination of the designs of AMLNG- and AMLGA-based app_{ij}.

2) Introducing double-sided errors: By replacing $b_{2i+1}\overline{b}_{2i}\overline{b}_{2i-1}\overline{a}_{j-1}$ in (19) and $\overline{b}_{2i+1} b_{2i} b_{2i-1} a_{j-1}$ in (21) by $b_{2i+1}\overline{a}_j$ and $\overline{b}_{2i+1} a_j$, respectively, $app1_{ij}$ is obtained by introducing four unbiased errors (see Table A14), as

$$\begin{aligned} app1_{ij} &= b_{2i+1} M(b_{2i}, b_{2i-1}, \overline{a}_j) + \overline{b}_{2i+1} \overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j) \\ &\quad + b_{2i+1} \overline{a}_j + \overline{b}_{2i+1} a_j \\ &= M(M(b_{2i+1}, M(M(b_{2i}, b_{2i-1}, \overline{a}_j), \overline{a}_j, 1), 0), \\ &\quad M(\overline{b}_{2i+1}, M(\overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j), a_j, 1), 0), 1). \end{aligned} \quad (25)$$

TABLE IV: Comparison of Different Approximate Booth Partial Product Generation Methods

PP Generations	approximate pp_{ij}						
	Type	Eq.	Errors	MV	INV	D	ER(%)
AMLCG	app_{ij}	(16)	positive	4	2	3	12.5
	app_{0j} $(i = 0, j \neq 0)$	(18)	unbiased	4	2	3	12.5
	app_{00} $(i = 0, j = 0)$	(16)		4	2	3	0
	other app_{ij}	(17)		6	3	3	12.5
AMLNG	app_{0j} $(i = 0, j \neq 0)$	(18)	unbiased	4	2	3	12.5
	app_{00} $(i = 0, j = 0)$	(16)		4	2	3	0
	other app_{ij}	(20)		5	2	3	12.5
AMLGA	app_{0j} $(i = 0, j \neq 0)$	(23)	unbiased	4	3	3	12.5
	app_{00} $(i = 0, j = 0)$	(22)		5	4	3	0
	other app_{ij}	(22)		5	4	3	12.5
AMLGB	app_{ij}	(24)	negative	4	3	3	12.5
	app_{0j} $(i = 0, j \neq 0)$	(23)	unbiased	4	3	3	12.5
	app_{00} $(i = 0, j = 0)$	(22)		5	4	3	0
	other app_{ij}	(25)		6	3	4	12.5

When $i = 0$, pp_{0j} for MLGB has the same logic expression as MLGA. (23) and (22) are employed to represent the AMLGB-based app_{0j} ($j \neq 0$) and app_{00}, respectively. Therefore, for AMLGB, (23), (22) and (25) are employed to generate app_{0j} ($j \neq 0$), app_{00} and all other app_{ij}.

E. Hardware Comparison and Error Analysis

Table IV shows the comparison of the different approximate PPGs. The AMLCG and AMLGB require a similar hardware complexity. Compared to introducing single-sided errors, introducing unbiased errors in AMLCG and AMLGB results in the increase of hardware complexity. With unbiased errors, AMLNG and AMLGA save two more majority gates than AMLCG and AMLGB. Compared to their exact counterparts in Table III, AMLNG and AMLGA save up to 58% of utilized majority gates, while AMLCG and AMLGB save 40% and 60% of utilized majority gates with unbiased and single-sided errors, respectively.

The errors are analyzed for 8×8 Booth multipliers. As defined in [11], the approximation factor p denotes the number of columns with approximate PPs in the PP array. The approximation is applied from the PPs of least significance.

The AMLCG- or AMLGB-based multipliers with unbiased errors lead to the same error characteristics. With an increase of p, the improvement in errors increases up to 56.9% in NMED and 81.1% in MRED compared to their counterparts with single-sided errors. Compared with previous designs in [11], the AMLGB-based designs with unbiased errors show an improvement of up to 56% and 81% in NMED and MRED, respectively. For unbiased errors, when $p < 5$, the AMLGA-based designs show a higher accuracy than other PPG-based multipliers; when $p > 5$, the AMLCG- or AMLGB-based designs are better choices for higher accuracy; when $p = 5$, the AMLGA- and AMLGB-based designs are superior in MRED, respectively. Detailed error analysis are presented in Tables A15 and A16.

V. Proposed Partial Product Reduction

Depending on the different error characteristics introduced by the proposed approximate PPGs, complementary strategies

are utilized in the PP reduction based on a probability analysis. The 8×8 Booth multiplier is presented as a case study.

A. PP Reduction for the Designs with Single-sided Errors

1) Positive single-sided errors: The AMLCG-based design with positive single-sided errors results in larger results than the exact results. As a common approximation technique, truncation can significantly reduce hardware complexity to generate output values that are smaller than the exact results. Therefore, truncation is considered as a complementary strategy for the positive errors due to the approximate PP generations.

Assume that the number of truncated columns from the least significant column in the PP array is $t+1$ ($0 \leq t < p$, and $t = 0$ means that Column 0 including Neg_i is truncated). So, $p-t-1$ columns remain using the approximate PPGs. To select the number of columns in the PP array for truncation under different values of p to maximize the effects of error compensation, we assume the probability of generating effective carries from the truncated columns be P_G and the probability of introducing positive single-sided errors by the remaining approximate PPs after truncation be P_{CP}.

For a specific p, P_G and P_{CP} are calculated using a probabilistic analysis (see the supplementary document) and compared for different values of t. The selection of t is determined to ensure the difference between P_G and P_{CP} the smallest. Combined with truncation, the improvement in accuracy increases with p, up to 27.7% in NMED and 8.4% in MRED (as per Table A17).

2) Negative single-sided errors: The AMLGB-based design with negative single-sided errors makes the results smaller than the exact ones. Instead of using truncation, the PPs of lower significance are replaced by '1's to make the approximate results larger to compensate for the errors.

Assume the number of columns in which the PPs are set to '1's (from the least significance in the PP array) is $l + 1$ ($0 \leq l < p$, *e.g.* $l = 0$ means that the PP in Column 0 is set as '1's, except for Neg_0); then, the remaining $p - l - 1$ columns of PPs are generated by the approximate PPGs. A similar probabilistic analysis is then performed to decide the pairs of l and p (see the supplementary material).

The use of the complementary strategy achieves an improvement up to 27.7% in NMED and 9.4% in MRED (as per Table A17). These two PP generation methods with complementary strategies result in multipliers with similar error characteristics. By using the probability analysis, it can be concluded that excluding the exact PPs, it is better to keep two columns of higher significance for approximation while the other columns complement the errors. As verified by exhaustive simulation, the selected pairs of t and p, and l and p (as per Table A17) are the best values. For $p = 5$ and 8, the complementary strategy for an AMLGB-based design is slightly more effective. For an AMLCG-based design with PP reduction, the complementary strategy (truncation) can significantly reduce the bit-width of the final results.

B. PP Reduction for the Designs with Unbiased Errors

Truncation is used as a complementary strategy for designs with a relatively high error tolerance ($p \geq 5$). As discussed in Sections III and IV, AMLCG- and AMLGB-based multipliers with unbiased errors are considered due to their smaller hardware overhead to generate exact PPs and the higher accuracy to generate approximate PPs. From the experiments, although truncation increases the NMED, it can improve the MRED (see Table A18). For an improvement in MRED and to maximize the advantages of truncation, AMLGB-based multipliers with $t = 0, 1, 1$ for $p = 5, 6, 7$ are preferred; while the AMLCG-based multiplier with $t = 2$ is preferred for $p = 8$.

VI. Proposed Approximate Booth Multipliers

Prior to compression, the PP array is preprocessed, as discussed in Section V. For compression, exact full adders are used to preserve the accuracy; however, they can be replaced by approximate adders to further reduce hardware at a loss of accuracy. As a case study, the 8×8 radix-4 Booth multiplier is presented to show the compression process.

A. Designs for the Compression of PP Arrays

In this section, we consider accuracy, at $p = 4$, $p = 6$ and $p = 8$. As discussed in Sections IV and V, with unbiased double-sided errors, the AMLGA-based design ($p = 4$), the AMLGB-based design ($p = 6$, $t = 1$) and the AMLCG-based design ($p = 8$, $t = 2$) are considered.

For AMLCG- and AMLGB-based multipliers with single-sided errors, the PP array can be significantly reduced by using the complementary strategies, but with a high accuracy loss. Therefore, they are suitable for the low accuracy cases (i.e. $p = 6$ and 8) to reduce the hardware. The AMLCG-based design (positive, $p = 6$, $t = 3$) and the AMLGB-based design (negative, $p = 8$, $l = 5$) are considered. If the smallest bit-width of the output result is desired, then the AMLCG-based design is preferred.

The reduced PP array are compressed by leveraging the different characteristics of PPs. The compressed design for the AMLGB-based PP array (negative, $p = 8$, $l = 5$) requires thirteen full adders and a 9-bit ripple carry adder (RCA) (see Appendix A.E). The AMLCG (positive, $p = 6$, $t = 3$)-based PP array is compressed by using fifteen full adders and a 11-bit RCA. To simplify the compression process of the AMLGA (unbiased, $p = 4$)-based PP array, Neg_0 is discarded resulting in a 28% increase in NMED but a 2.8% decrease in MRED; thus, the compression requires sixteen full adders and a 12-bit RCA. The compression of the AMLGB (unbiased, $p = 6$, $t = 1$)-based PP array requires sixteen full adders and a 12-bit RCA; the compression of the AMLCG (unbiased, $p = 8$, $t = 2$)-based PP array requires seventeen full adders and a 10-bit RCA.

B. Hardware Analysis

Table V presents the numbers of MVs and INVs, D, the area-delay product (ADP, MV $\times$ D), the error metrics (NMED and

MRED) and the bit width (BW) of the final results for the exact and proposed approximate Booth multipliers. ADP is used to assess the overall hardware efficiency. For the reduction of PPs in the exact multipliers, the schemes of Section VI are used, i.e., only full adders are applied in parallel prior to obtaining the two rows of PPs. Thus, thirty full adders and a 13-bit RCA are required in the compression process. Since no other ML-based approximate signed multipliers were previously proposed, only exact Booth multipliers are considered for comparison.

TABLE V: Hardware and Error Evaluation of Exact and Approximate 8×8 Booth Multipliers based on Different PPGs

Multiplier	Errors	p	t	l	MV	INV	D	NMED (10^{-3})	MRED (10^{-3})	ADP	BW	Energy(fJ)	
												NML	STT-MTJ
MLCG	-	-	-	-	442	138	24	0	0	10608	16	2.29	817
MLNG	-	-	-	-	479	106	25	0	0	11975	16	2.49	886
MLGA	-	-	-	-	497	106	24	0	0	11928	16	2.58	919
MLGB	-	-	-	-	444	138	24	0	0	10656	16	2.30	821
[11]	positive	6	-	-	638	142	25	1.3	14.2	15950	16	3.31	1180
		8	-	-	518	142	25	7.2	95.6	12950	16	2.69	958
AMLCG	positive	6	3	-	329	102	20	1.1	12	6580	12	1.71	608
	unbiased	8	2	-	325	106	20	3.3	17	6500	13	1.69	601
AMLGA	unbiased	4	-	-	435	76	21	0.14	0.61	9135	16	2.26	804
AMLGB	negative	8	-	5	258	84	19	5.2	86	4902	16	1.34	477
	unbiased	6	1	-	363	106	21	0.71	3.5	7623	14	1.88	671

With no approximation, the MLCG- and MLGB-based exact multipliers show good performance, reducing the number of majority gates by up to 11%, and the ADP by up to 12% compared with the MLNG- and MLGA-based exact schemes. Using approximate PP generation methods, the hardware decreases with a reduced accuracy (in NMED or MRED). Compared with the MLCG-based exact multiplier, those using approximate PPGs can obtain an improvement from 13% to 53% in the ADP and from 12% to 20% in delay. The AMLGA-based multiplier achieves only a reduction of 1.5% in the number of majority gates, 45% of the inverters and 12% of the delay. The AMLGB-based multiplier with negative single-sided errors ($p = 8$) reduces up to 41% of the majority gates, 39% of the inverters, 20% of the delay and 53% of the ADP at a relatively large decrease in accuracy. Moreover, with truncation, the bit-width of the final result can be reduced by up to 12 bits.

The proposed multipliers with positive single-sided and unbiased double-sided errors introduced by AMLCG perform similarly in hardware metrics. The AMLCG (positive, $p = 6$, $t = 3$)-based multiplier design is likely to be preferred, due to the higher accuracy. Due to the specific ML optimizations of PPGs, even the exact multiplier designs in Table V are superior in hardware than the approximate multiplier design in [11]. Moreover, the additional use of truncation can improve accuracy and reduce the bit-width of the final result.

Finally, some ML-based technologies are considered for power dissipation, including NML and STT-MTJ. For simplicity the power consumption and critical path delay of a majority gate are considered as a unit under different technologies or devices to estimate the energy consumption of the considered designs. The energy for the majority gate based on NML or STT-MTJ is approximately 5.2×10^{-3} fJ [2] or 1.85 fJ [3], respectively. As shown in Table V, the proposed AMLGB-based multiplier (negative, $p = 8$, $l = 5$) leverages the low-power nature of emerging nanotechnologies, consuming only 1.34 fJ and 6.71×10^2 fJ for NML and STT-MTJ, respectively. These results show the power efficiency of the proposed ML-based multipliers in emerging nanotechnologies.

In Fig. 2, it is shown that compared with exact multiplier designs, the AMLGA-based design achieves at least a reduction of 13% in ADP. With a further decrease in accuracy, the AMLGB-based design (unbiased, $p = 6$, $t = 1$) reduces the ADP by 28%. Although the AMLCG-based designs have a similar ADP, the one with positive single-sided errors is superior in NMED. The AMLGB-based design (negative, $p = 8$, $l = 5$) saves 53% of ADP with a large accuracy loss. Therefore, four levels of accuracy can be established for different requirements of applications: high accuracy (AMLGA, unbiased, $p = 4$), good accuracy (AMLGB, unbiased, $p = 6$, $t = 1$), moderate accuracy (AMLCG, positive, $p = 6$, $t = 3$), and low accuracy (AMLGB, negative, $p = 8$, $l = 5$).

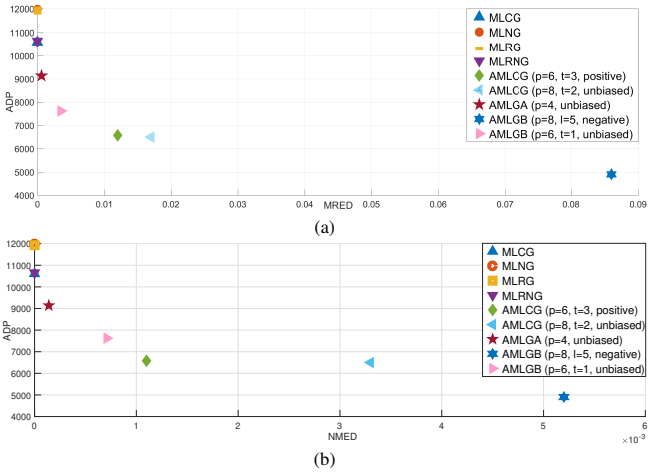


Fig. 2: Errors (in NMED and MRED) vs. hardware complexity for exact and approximate 8×8 Booth multipliers: (a) ADP vs NMED and (b) ADP vs MRED.

VII. Application

Several applications are considered for the proposed approximate multipliers at four different accuracy levels: low and moderate accuracy for image multiplication and edge detection using the Sobel operator, good and high accuracy for a multilayer perceptron (MLP) for classification and a multi-task CNN for joint face detection and alignment, respectively.

A. Image Processing

Image multiplication [10] and edge detection using the Sobel operator [19] are considered for the proposed approximate multipliers with low accuracy (AMLGB, negative, $p = 8$, $l = 5$) and the ones with moderate accuracy (AMLCG, positive, $p = 6$, $t = 3$), respectively. The structural similarity index measure (SSIM) and the peak signal-to-noise ratio (PSNR) are used to evaluate the quality of the image processing results.

The experiments on ten different images return on average a PSNR of 59.34 dB and an SSIM of 0.9993 for the image multiplication, and a PSNR of 71.09 dB and an SSIM of 0.9998 for the edge detection, respectively.

B. Classification

The proposed approximate 8×8 radix-4 Booth multipliers (AMLGB, unbiased, $p = 6$, $t = 1$) with good accuracy are used in an MLP of three layers (an input layer of 784 neurons, one hidden layer of 100 neurons and an output layer of 10 neurons) to classify the MNIST dataset. The performance of the MLP is assessed by the accuracy of the classification.

The 8×8 approximate Booth multipliers are utilized in the inference. The trained weight matrix, the inputs and biases are mapped into [-128, 127] for the inference. Using MATLAB, 94.41% and 94.34% are obtained as classification accuracies when using the exact and approximate multipliers, respectively. Moreover, compared with the exact design, the proposed design reduces the ADP of the multipliers in inference by up to 28% along with a reduced hardware for the accumulation operations due to the truncation in each multiplication.

C. Face Detection and Alignment

A multi-task CNN (MTCNN) is considered for joint face detection and alignment, using the Face Detection Data Set and Benchmark (FDDB) and Annotated Facial Landmarks in the Wild (AFLW) datasets. The MTCNN consists of three cascaded CNNs: the so-called "proposal network" (PNet), "refine network" (RNet) and "output network" (ONet) [4]. The true positive rate (TPR) and the normalized mean error (NME) indicate the accuracy in face detection and alignment, respectively. Different numbers of multiply-and-accumulate (MAC) units are required when the accuracy varies because of the face detection algorithm. As a MTCNN is usually executed by using data pipelines due to the limitation of the hardware resources, we report the average number of MACs required for detecting the faces in an image on the FDDB dataset to assess the hardware overhead related to performance and energy consumption.

Due to the large number of multiplications required for a MTCNN, approximate 8×8 multipliers with a relatively high accuracy (AMLGA, unbiased, $p = 4$) are used to replace the exact circuits. Fig. 3 shows an example for the results of face detection and alignment. Compared with the exact multiplier, the use of the proposed approximate multipliers achieves a 5.1% reduction in the number of MACs due to the acceleration of inference (0.48 billion vs 0.51 billion), thus approximately saving 18.9% in ADP for the MACs at a 7.4% accuracy loss with a TPR of 80.24%; however, the NME is increased from 3.58% to 11.01%. A higher accuracy can be achieved by using approximate multipliers for the less error-sensitive MACs and exact designs for the more error-sensitive components. The proposed approximate multiplier performs better in face detection than alignment, while the approximate multipliers [11] for $p = 4$ show an unacceptable TPR of 0.23% and an NME of 32.88%.

(a) (b)

Fig. 3: Results of face detection and alignment using 8×8 approximate Booth multipliers with high accuracy (AMLGA, unbiased, $p = 4$): (a) face alignment and (b) face detection.

VIII. Conclusion

In this paper, approximate Booth multipliers are proposed based on ML that are applicable to emerging nanotechnologies. New partial product generation methods are first developed for efficient ML implementations. Approximate partial product generators are then designed with different types of errors: positive single-sided, negative single-sided, biased double-sided and unbiased double-sided errors. Improvements up to 56% and 81% in NMED and MRED are achieved in comparison to previous designs. For the reduction of the PP array, complementary strategies are proposed by considering the features of the errors. Various schemes for PP compression are then designed to efficiently calculate the final results. Finally, four approximate multipliers with different levels of accuracy are proposed to meet various design requirements in area, delay, and accuracy. A hardware evaluation at the gate level shows a saving of 28% in ADP compared to the exact design and very low NMED and MRED of 0.00071 and 0.0035 for ML-based nanotechnologies (NML and MTJ). Case studies of image processing and neural networks show the great potential of the proposed designs for ML-based emerging technologies.

Acknowledgment

This work was supported by NSERC (RES0048688), and NSFC (62022041 and 62104127). The research of Fabrizio Lombardi was supported by NSF under CCF-1953961 and 1812467 grants. T. Zhang was supported by a Ph.D. scholarship from the China Scholarship Council (CSC).

References

[1] B. Parhami, D. Abedi, and G. Jaberipur, "Majority-Logic, its applications, and atomic-scale embodiments," Comput. Elect. Eng., vol. 83, no.106562, 2020.

[2] M. Vacca, M. Graziano and M. Zamboni, "Majority voter full characterization for nanomagnet logic circuits," IEEE Trans. Nanotechnol., vol. 11, no. 5, pp. 940-947, 2012.

[3] V. Jamshidi, "A vlsi majority-logic device based on spin transfer torque mechanism for brain-inspired computing architecture," IEEE Trans. Very Large Scale Integration Syst., vol. 28, no. 8, pp. 1858-1866, 2020.

[4] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: a survey, characterization and recent applications," Proc. IEEE, vol. 108, no. 12, pp. 2108-2135, 2020.

[5] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," IEEE Trans. Circuits Syst. I, Regular Papers, vol. 66, no. 11, 2019.

[6] C. Labrado, H. Thapliyal and F. Lombardi, "Design of majority logic based approximate arithmetic circuits," in Proc. IEEE Int. Symp. Circuits Syst., pp. 2122-2125, 2017.

[7] T. Zhang, W. Liu, E. McLarnon, M. O'Neill and F. Lombardi, "Design of majority logic (ML) based approximate full adders," in Proc. IEEE Int. Symp. Circuits Syst.,, pp. 1-5, 2018.
[8] M. H. Moaiyeri, F. Sabetzadeh, and S. Angizi, "An efficient majority-based compressor for approximate computing in the nano era," Microsyst. Technol., vol. 4, pp. 1-13, 2017.
[9] S. Angizi, H. Jiang, R. F. DeMara, J. Han and D. Fan, "Majority-based spin-CMOS primitives for approximate computing," IEEE Trans. Nanotechnol., vol. 17, no. 4, pp. 795-806, 2018.
[10] W. Liu, T. Zhang, E. McLarnon, M. O'Neill, P. Montuschi and F. Lombardi, "Design and analysis of majority logic based approximate adders and multipliers," IEEE Trans. Emerg. Top. Comput., 2019.
[11] T. Zhang, W. Liu, J. Han and F. Lombardi, "Design and analysis of majority Logic based approximate radix-4 booth encoders," in Proc. IEEE/ACM Int. Symp. Nanoscale Architecturess, pp. 1-6, 2019.
[12] X. Cui, W. Liu, E. Swartzlander and F. Lombardi, "A modified partial product generator for redundant binary multipliers," IEEE Trans. Comput., vol. 65, no. 4, pp. 1165-1171, 2016.
[13] O. MacSorley, "High-speed arithmetic in binary computers," Proc. IRE, vol. 49, pp. 67–91, 1961.
[14] W. Yeh and C. Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, no. 7, pp. 692-701, 2000.
[15] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Comput., vol. 62, no. 9, pp. 1760-1771, 2013.
[16] L. Amaru, E. Testa, M. Couceir, O. Zografos, G. D. Micheli and M. Soeken, "Majority logic synthesis," In Proc. Int. Conf. Comput.-Aided Des., no. 79, pp. 1-6, 2018.
[17] R. Zhang, P. Gupta and N. K. Jha, "Majority and minority network synthesis with application to QCA-, SET-, and TPL-based nanotechnologies," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 26, no. 7, pp. 1233-1245, 2007.
[18] V. Pudi, K. Sridharan, and F. Lombardi, "Majority logic formulations for parallel adder designs at reduced delay and circuit complexity," IEEE Trans. Comput., vol. 66, no. 10, pp. 1824-1830, 2017.
[19] L. Vasileios, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," IEEE Trans. Very Large Scale Integration Syst., vol. 26, no. 3, pp. 421-430, 2017.

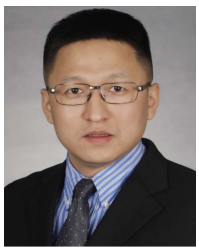
Tingting Zhang (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in the College of Electronic and Information Engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2016 and 2019, respectively. She is working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada, since Sep. 2019. Her research interests include computer arithmetic, high-performance computing, approximate computing, and emerging technologies in computing systems.

Honglan Jiang (Member, IEEE) received the B.Sc. and Master's degrees in instrument science and technology from the Harbin Institute of Technology, Harbin, China, in 2011 and 2013, respectively, and the Ph.D. degree from the University of Alberta, Edmonton, AB, Canada, in 2018. From 2018 to 2021, she was a Postdoctoral Fellow with the School of Integrated Circuits, Tsinghua University, Beijing, China. She is currently an associate professor in the Department of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai, China. Her research interests include approximate computing, reconfigurable computing, and stochastic computing.

Hai Mo received the B.S. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2019. He is working toward the M.S. degree at the School of Integrated Circuits, Tsinghua University, Beijing, China, on approximation algorithm and highperformance computing. His research interests include architecture design, in-memory computing, and hardware implementation for deep learning accelerators.

Weiqiang Liu (Senior Member, IEEE) received the B.Sc. degree in information engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2006, and the Ph.D. degree in electronic engineering from Queen's University Belfast (QUB), Belfast, U.K., in 2012. In December 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently a Professor and the Vice Dean. His research interests include approximate computing, hardware security, and VLSI design for digital signal processing and cryptography.

Fabrizio Lombardi (Life Fellow, IEEE) received the B.S. degree (Hons.) in electronic engineering from the University of Essex, U.K., in 1977, the master's degree in microwaves and modern optics, the Diploma degree in microwave engineering from the Microwave Research Unit, University College London, in 1978, and the Ph.D. degree from the University of London in 1982. He is currently the International Test Conference (ITC) Endowed Chair Professorship with Northeastern University, Boston, USA. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems.



Leibo Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering and the Ph.D. degree from the Institute of Microelectronics, Tsinghua University, Beijing, China, in 1999 and 2004, respectively. He is currently a Full Professor with the School of Integrated Circuits, Tsinghua University. His current research interests include reconfigurable computing, mobile computing, and very large-scale integration digital signal processing.

Jie Han (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Delft University of Technology, Delft, The Netherlands, in 2004. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, and novel computational models for nanoscale and biological applications.

APPENDIX A
K-MAPS, ERROR ANALYSIS, COMPLEMENTARY STRATEGIES, COMPRESSION

A. K-maps for exact and approximate PPGs

1) K-maps for exact PPGs: The K-maps for exact PP generations are given in Tables A1-A6, including MLCG, MLNG and the newly proposed MLGA and MLGB. The maxterms in light gray and the minterms in dark gray can be represented using 3-input majority gates using K-map based ML optimization.

TABLE A1: K-Map of the MLCG-based pp_{ij} [14]

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	0	1	1
01	0	0	1	0	1	0	1	0
11	0	1	1	1	0	0	0	0
10	0	1	0	1	0	0	0	1

TABLE A2: K-Map of the MLNG-based pp_{ij} [12]

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	1	1	1
01	0	0	1	0	1	1	1	0
11	0	1	1	1	0	1	0	0
10	0	1	0	1	0	1	0	1

TABLE A3: K-Map of the MLGA-based pp_{ij}

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	1	0	0	0	1	0	1	1
01	1	0	1	0	1	0	1	0
11	1	1	1	1	0	0	0	0
10	1	1	0	1	0	0	0	1

TABLE A4: K-Map of the MLGA-based Neg_i

b_{2i+1} \ $b_{2i}\ b_{2i-1}$	00	01	11	10
0	1	0	0	0
1	1	1	0	1

TABLE A5: K-Map of the MLGB-based pp_{ij}

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	1	0	0	0	1	1	1	1
01	1	0	1	0	1	1	1	0
11	1	1	1	1	0	1	0	0
10	1	1	0	1	0	1	0	1

TABLE A6: K-Map of the MLGB-based Neg_i

b_{2i+1} \ $b_{2i}\ b_{2i-1}$	00	01	11	10
0	1	0	0	0
1	1	1	1	1

TABLE A7: K-Map of the AMLCG-based app_{ij}

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	(1)	0	1	0	1	1
01	0	0	1	0	1	0	1	(1)
11	0	1	1	1	0	0	0	(1)
10	0	1	(1)	1	0	0	0	1

TABLE A8: K-Map of the AMLCG-based $app1_{ij}$

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	0	0	1	0	1	1
01	0	0	(0)	0	1	0	1	(1)
11	0	1	1	1	0	0	0	0
10	0	1	(1)	1	0	0	0	(0)

TABLE A9: K-Map of the AMLCG-based $app2_{0j}$

$a_j\ a_{j-1}$ \ $b_1\ b_0$	00	01	11	10
00	0	0	1	1
01	0	0	1	(1)
11	0	1	0	0
10	0	1	0	(0)

TABLE A10: K-Map of the AMLNG-based $app1_{ij}$

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	0	0	(1)	0	1	1	1	(0)
01	0	0	1	0	1	1	1	0
11	0	1	1	1	0	1	0	0
10	0	1	(1)	1	0	1	0	(0)

TABLE A11: K-Map of the AMLGA-based $app1_{ij}$

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	1	0	0	0	1	0	1	1
01	1	0	(0)	0	1	0	1	(1)
11	1	1	(0)	1	0	0	0	(1)
10	1	1	0	1	0	0	0	1

TABLE A12: K-Map of the AMLGA-based $app2_{0j}$

$a_j\ a_{j-1}$ \ $b_1\ b_0$	00	01	11	10
00	1	0	1	1
01	1	0	1	(1)
11	1	1	0	0
10	1	1	0	(0)

TABLE A13: K-Map of the AMLGB-based app_{ij}

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	1	0	0	0	1	1	1	(0)
01	1	0	(0)	0	1	1	1	0
11	1	1	(0)	1	0	1	0	0
10	1	1	0	1	0	1	0	(0)

TABLE A14: K-Map of the AMLGB-based $app1_{ij}$

$a_j\ a_{j-1}$ \ $b_{2i+1}\ b_{2i}\ b_{2i-1}$	000	001	011	010	110	111	101	100
00	1	0	0	0	1	1	1	1
01	1	0	⓪	0	1	1	1	①
11	1	1	1	1	0	1	0	0
10	1	1	①	1	0	1	0	⓪

2) K-maps for approximate PPGs: Tables A7-A14 present the K-maps of approximate pp_{ij} based on different PPGs.

B. Error analysis of the proposed approximate PPGs

The approximate designs are measured with respect to the introduced error directions, i.e., single-sided vs unbiased double-sided errors, and biased vs unbiased doublesided errors.

1) Single-sided vs unbiased double-sided errors: Table A15 shows the NMED/MRED of the AMLCG- and AMLGB-based designs with single-sided and unbiased double-sided errors.

With single-sided errors, the use of AMLGB shows a slightly higher accuracy (in NMED and MRED) than for AMLCG when $p \leq 5$. When generating approximate PPs with single-sided errors, the errors accumulate, resulting in a substantial difference from the exact result. Therefore, approximate PPGs with unbiased errors show lower NMEDs and MREDs than for single-sided errors in the approximate multiplier.

2) Biased or unbiased double-sided errors: The comparison of errors due to the use of AMLNG and AMLGA with biased and unbiased double-sided errors is given in Table A16. Unlike designs with unbiased errors, the AMLNG- and AMLGA-based designs with biased errors ignore the different characteristics of PPs, only using (20) and (22) to compute the PPs, respectively.

Generally, approximate PPGs with unbiased errors result in a more accurate approximate multiplier (with lower NMEDs and MREDs) than for biased errors. Independently of biased or unbiased errors, the AMLGA-based designs show better accuracy than AMLNG. Moreover, the NMED for AMLGA-based designs with biased errors is comparable to that for AMLNG-based ones with unbiased errors.

C. Complementary strategies for the PP reduction

1) Positive single-sided errors: Define C_i and N_i as the carry-out and the number of '1's in Column i. C_i can be larger than '1' for multiple PPs in a column. The probability of the carry values for each column in the PP array is analyzed using Bayes theory, i.e., computing the prior probability as per the posterior knowledge of the relationship between input operands and output results.

For example, in Column 0, there are two operands (pp_{00} and Neg_0), so C_0 has two possible values: 0 and 1. Hence:

$$\begin{aligned} P(C_0 = 1) &= P(N_0 = 2) = P(pp_{00} = 1 \cap neg_0 = 1) \\ &= P(pp_{00} = 1 | neg_0 = 1) P(neg_0 = 1) \\ &= \frac{3}{4} \times \frac{1}{2} = \frac{3}{8}. \end{aligned} \quad (1)$$

The probabilities of the carry values for the other columns in the PP array can be similarly derived. Then, P_G can be derived as approximately:

$$P_G = \sum_{i=1}^{max} 2^{i-1} \times P(C_t = i). \quad (2)$$

where max is the largest possible value of the carry generated by the truncated columns.

The proposed AMLCG-based app_{ij} with single-sided errors has a probability of 50% to introduce positive errors when $b_{2i+1}b_{2i}b_{2i-1} = 011$ and 100 (as per Table A7). Since different PPs may be correlated, the conditional probability is considered in this analysis. Based on the analysis of the carries, $P(b_{2i+1}b_{2i}b_{2i-1} = 011 \text{ or } 100 | P(C_n > 0))$ can be easily obtained. When carries are generated for the truncated columns, the errors introduced in the remaining PPs of higher significance can mitigate the accuracy loss. Assume the probability of introducing positive single-sided errors for each remaining approximate PP is given by ${P_{CP}}^{app_{ij}}$ ($0 \leq i \leq \lceil \frac{p}{2} - 1 \rceil$, $t + 1 - 2i \leq j \leq p - 2i - 1$). ${P_{CP}}^{app_{ij}}$ takes into consideration the distance from the truncated column with the highest significance to app_{ij}, as given by

$$\begin{aligned} {P_{CP}}^{app_{ij}} &= 2^{j-(t+1-2i)} \times \frac{1}{2} \\ &\quad \times P(b_{2i+1}b_{2i}b_{2i-1} = 011/100, P(C_t > 0)) \\ &= 2^{j-(t+2-2i)} \times P(C_t > 0) \times \\ &\quad P(b_{2i+1}b_{2i}b_{2i-1} = 011/100 | P(C_t > 0)) \end{aligned} \quad (3)$$

As the sum of all ${P_{CP}}^{app_{ij}}$, P_{CP} is given by

$$P_{CP} = \sum_{i=0}^{\lceil \frac{p}{2} - 1 \rceil} \sum_{j=t+1-2i}^{p-2i-1} {P_{CP}}^{app_{ij}}. \quad (4)$$

The selected pair of t and p depends on the distance between P_G and P_{CP} to make their values the closest.

2) Negative single-sided errors: Let $P_E^{pp_{ij}}$ ($0 \leq i \leq \lceil \frac{l+1}{2} - 1 \rceil$, $0 \leq j \leq l - 2i$) and P_E be the probability of introducing positive errors resulting from using '1' to replace each pp_{ij} and the original PPs in all $l + 1$ columns, respectively. For each pp_{ij}, there is a probability of $\frac{5}{8}$ to replace the original '0's, thus introducing positive errors. The calculation of $P_E^{app_{ij}}$ considers the difference between the significances of pp_{ij} and Column $l + 1$, as

$${P_E}^{pp_{ij}} = 2^{j-(l+1-2i)} \times \frac{5}{8} = \frac{5}{2^{l+4-2i-j}}. \quad (5)$$

To simplify the analysis, assume that the generation of each PP is independent. Thus, P_E is the sum of all $P_E^{pp_{ij}}$:

$$P_E = \sum_{i=0}^{\lceil \frac{l+1}{2} - 1 \rceil} \sum_{j=0}^{l-2i} {P_E}^{pp_{ij}}. \quad (6)$$

Similarly, ${P_{CN}}^{app_{ij}}$ ($0 \leq i \leq \lceil \frac{p}{2} - 1 \rceil$, $l + 1 - 2i \leq j \leq p - 2i - 1$) is defined as the probability of introducing negative errors due to each remaining app_{ij} encoded by AMLGB on Column $l + 1$. Each app_{ij} using AMLGB has a probability of $\frac{1}{8}$ to introduce a negative error. ${P_{CN}}^{app_{ij}}$ also depends on the difference between the significances of app_{ij} and Column $l+1$, as

$${P_{CN}}^{app_{ij}} = 2^{j-(l+1-2i)} \times \frac{1}{8} = 2^{j-(l+4-2i)}. \quad (7)$$

TABLE A15: Error Measurements for an 8×8 Booth Multiplier using AMLCG and AMLGB with Single-sized or Unbiased Double-sided Errors in PP Generation

Encoders	AMLCG [11] using (16) (single-sized)		AMLCG using (18) & (16) & (17) (unbiased)		AMLGB using (24) (single-sized)		AMLGB using (23) & (22) & (25) (unbiased)	
	NMED	MRED	NMED	MRED	NMED	MRED	NMED	MRED
$p=1$	0	0	0	0	1.52×10^{-5}	6.10×10^{-5}	0	0
$p=2$	1.52×10^{-5}	6.10×10^{-5}	1.52×10^{-5}	9.15×10^{-5}	3.05×10^{-5}	1.22×10^{-4}	1.52×10^{-5}	9.15×10^{-5}
$p=3$	7.62×10^{-5}	4.27×10^{-4}	5.91×10^{-5}	3.50×10^{-4}	9.15×10^{-5}	4.57×10^{-4}	5.91×10^{-5}	3.50×10^{-4}
$p=4$	1.98×10^{-4}	1.4×10^{-3}	1.13×10^{-4}	6.71×10^{-4}	2.13×10^{-4}	1.5×10^{-3}	1.13×10^{-4}	6.71×10^{-4}
$p=5$	5.64×10^{-4}	4.9×10^{-3}	3.21×10^{-4}	1.9×10^{-3}	5.79×10^{-4}	5.0×10^{-3}	3.21×10^{-4}	1.9×10^{-3}
$p=6$	1.3×10^{-3}	1.4×10^{-2}	6.32×10^{-4}	3.7×10^{-3}	1.3×10^{-3}	1.4×10^{-2}	6.32×10^{-4}	3.7×10^{-3}
$p=7$	3.3×10^{-3}	3.9×10^{-2}	1.6×10^{-3}	9.2×10^{-3}	3.3×10^{-3}	3.9×10^{-2}	1.6×10^{-3}	9.2×10^{-3}
$p=8$	7.2×10^{-3}	9.5×10^{-2}	3.1×10^{-3}	1.8×10^{-2}	7.2×10^{-3}	9.5×10^{-2}	3.1×10^{-3}	1.8×10^{-2}

TABLE A16: Error Measurements for an 8×8 Booth Multiplier using AMLNG and AMLGA with Biased or Unbiased Double-sided Errors in PP Generation

Encoders	AMLNG using (20) (biased double-sized)		AMLNG using (18) & (16) & (20) (unbiased double-sized)		AMLGA using (22) (biased double-sized)		AMLGA using (23) & (22) & (22) (unbiased double-sized)	
	NMED	MRED	NMED	MRED	NMED	MRED	NMED	MRED
$p=1$	1.52×10^{-5}	6.10×10^{-5}	0	0	0	0	0	0
$p=2$	3.05×10^{-5}	1.22×10^{-4}	1.52×10^{-5}	9.15×10^{-5}	1.52×10^{-5}	6.10×10^{-5}	1.52×10^{-5}	9.15×10^{-5}
$p=3$	9.92×10^{-5}	4.73×10^{-4}	8.39×10^{-5}	3.66×10^{-4}	4.57×10^{-5}	3.05×10^{-4}	3.05×10^{-5}	1.83×10^{-4}
$p=4$	2.00×10^{-4}	1.3×10^{-3}	1.68×10^{-4}	8.08×10^{-4}	1.44×10^{-4}	1.2×10^{-3}	1.12×10^{-4}	6.25×10^{-4}
$p=5$	5.22×10^{-4}	4.1×10^{-3}	4.69×10^{-4}	2.5×10^{-3}	3.45×10^{-4}	3.6×10^{-3}	2.82×10^{-4}	2.0×10^{-3}
$p=6$	1.1×10^{-3}	1.1×10^{-2}	9.53×10^{-4}	6.6×10^{-3}	8.68×10^{-4}	1.0×10^{-2}	7.52×10^{-4}	6.3×10^{-3}
$p=7$	2.5×10^{-3}	2.9×10^{-2}	2.3×10^{-3}	1.8×10^{-2}	1.9×10^{-3}	2.7×10^{-2}	1.7×10^{-3}	1.7×10^{-2}
$p=8$	5.1×10^{-3}	6.9×10^{-2}	4.7×10^{-3}	4.4×10^{-2}	4.5×10^{-3}	6.7×10^{-2}	4.0×10^{-3}	4.3×10^{-2}

P_{CN} includes all ${P_{CN}}^{app_{ij}}$, as given by

$$P_{CN} = \sum_{i=0}^{\lceil \frac{p}{2}-1 \rceil} \sum_{j=l+1-2i}^{p-2i-1} {P_{CN}}^{app_{ij}}. \quad (8)$$

The selected pair of l and p depends on when the distance between P_E and P_{CN} is smallest.

Table A17 shows the analytical results of the values of t (and l) for p ($p \in [4, 8]$) and the error results for 8×8 multipliers.

3) Unbiased double-sided errors: Error results of AMLCG and AMLGB (unbiased errors) based 8×8 Radix-4 Booth multiplier combined with truncation when $p \geq 5$ is presented in Table A18. Only those cases when the MRED of the approximate multiplier has an improvement after combining with truncation are included.

From the experimental results, despite truncation increases the values of NMED, it could improve the accuracy in MRED. In Table A15, the AMLCG- and AMLGB-based multipliers without truncation share the same error characteristics. Truncation increases the error distances more when the accurate results are larger, but it decreases the error distances more when the accurate results are smaller. Compared with the error results presented in Table A15, using truncation in the AMLGB-based designs when $5 \leq p \leq 7$ can achieve a significant improvement in MRED but with a larger increase in NMED than in the AMLCG-based designs. When $p = 8$, if three columns of PPs are truncated ($t = 2$), the AMLCG-based design can still get an improvement in MRED but it is not applicable to the AMLGB-based one.

TABLE A17: Error Measurements for an 8×8 Booth Multiplier using the Proposed PP Reduction Schemes, AMLCG and AMLGB with Single-sided Errors for PP Generations

PPGs	AMLCG (positive)			AMLGB (negative)		
p	t	NMED	MRED	l	NMED	MRED
4	1	1.9×10^{-4}	1.4×10^{-3}	1	2.0×10^{-4}	1.4×10^{-3}
5	2	5.0×10^{-4}	4.3×10^{-3}	2	5.0×10^{-4}	4.1×10^{-3}
6	3	1.1×10^{-3}	1.3×10^{-2}	3	1.1×10^{-3}	1.3×10^{-2}
7	4	2.4×10^{-3}	3.4×10^{-2}	4	2.5×10^{-3}	3.4×10^{-2}
8	5	5.2×10^{-3}	8.7×10^{-2}	5	5.2×10^{-3}	8.6×10^{-2}

TABLE A18: Error Measurements for an 8×8 Booth Multiplier using Truncation for the PP Reduction, AMLCG and AMLGB with Unbiased-sided Errors for PP Generations

Value of p	Value of t	AMLCG		AMLGB	
		NMED	MRED	NMED	MRED
5	0	3.41×10^{-4}	1.8×10^{-3}	3.63×10^{-4}	1.7×10^{-3}
6	0	6.51×10^{-4}	3.3×10^{-3}	6.71×10^{-4}	3.1×10^{-3}
	1	6.72×10^{-4}	3.6×10^{-3}	7.11×10^{-4}	3.5×10^{-3}
7	0	1.6×10^{-3}	8.2×10^{-3}	1.6×10^{-3}	7.7×10^{-3}
	1	1.6×10^{-3}	8.5×10^{-3}	1.7×10^{-3}	8.0×10^{-3}
8	0	3.2×10^{-3}	1.61×10^{-2}	3.2×10^{-3}	1.48×10^{-2}
	1	3.2×10^{-3}	1.64×10^{-2}	3.2×10^{-3}	1.52×10^{-2}
	2	3.3×10^{-3}	1.70×10^{-2}	-	-

D. PP compression

The compressed design for the AMLGB-based PP array (negative, $p = 8$, $l = 5$) is shown in Fig. A1. At Stage 1, the initial PP array is generated using Columns 0 to 5 to complement errors. The constants '1's in Columns 2 to 5 are

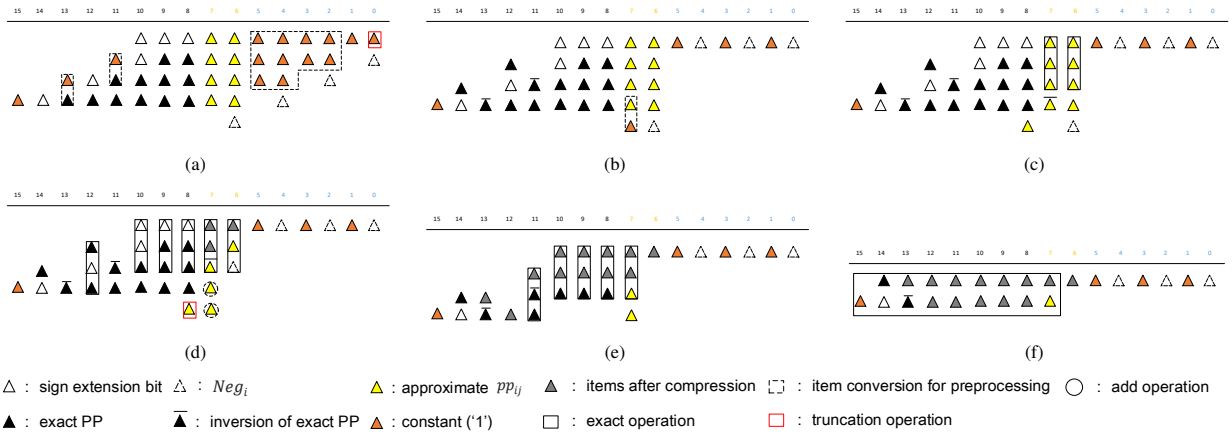


Fig. A1: Design of PP Compression in an 8×8 Booth multiplier using AMLGB (negative, $p = 8$, $l = 5$): (a) Stage 1 (preprocessing), (b) Stage 2 (preprocessing), (c) Stage 3, (d) Stage 4, (e) Stage 5, and (f) Stage 6.

preprocessed into "101010". To further simplify computation, the constant '1' in Column 0 is truncated. Consider that in binary computation, adding '1' to an operand results in a carry equal to the operand and a sum equal to the inversion of the operand, Columns 11 and 13 at Stage 1 and Column 7 at Stage 2 are simplified. The ML-based adder design requires two units of gate delay [12]. Stages 3 to 5 reduce the PP array to two rows with one additional term at the least significant position using full adders. Stage 3 compresses the approximate PPs in Columns 6 and 7 to keep the same execution time for the exact PP generation. At Stage 4, a circled PP in Column 8 is replaced by two terms in Column 7 to simplify the further compression. Finally, a 9-bit ML-based ripple carry adder (RCA) is required. The compressed PP designs for the other cases are similar.

E. Overall approximate Booth multiplier designs

Table A19 extends Table V to give the overall designs of the Booth multipliers.

TABLE A19: Extension of Table V

Multiplier	PP Generation		Errors	p	t	l	MV_{AM}	INV_{AM}	D	NMED (10^{-3})	MRED (10^{-3})	ADP	BW	Energy(fJ)	
	Equations	MV												NML	STT-MTJ
MLCG	$pp_{ij} = M(M(M(b_{2i+1}, 0, \overline{M}(b_{2i}, b_{2i-1}, a_j)), M(b_{2i}, 1, M(b_{2i-1}, \overline{a}_{j-1}, 1)), 0), M(\overline{b}_{2i+1}, 0, M(b_{2i}, b_{2i-1}, a_j)), M(a_{j-1}, 1, M(\overline{b}_{2i-1}, \overline{b}_{2i}, 1)), 0), 1)$ $Neg_i = M(b_{2i+1}, 0, M(\overline{b}_{2i}, \overline{b}_{2i-1}, 1))$	10 2	-	-	-	-	442	138	24	0	0	10608	16	2.29	817
MLNG	$pp_{ij} = M(M(b_{2i+1}, M(M(b_{2i}, b_{2i-1}, \overline{a}_j), M(\overline{b}_{2i}, M(\overline{b}_{2i-1}, \overline{a}_{j-1}, 0), 0), 1), 0), M(M(\overline{b}_{2i+1}, M(b_{2i}, b_{2i-1}, a_j), 0), M(\overline{b}_{2i}, M(\overline{b}_{2i-1}, a_{j-1}, 1), 1), 0), 1)$ $Neg_i = b_{2i+1}$	11 0	-	-	-	-	479	106	25	0	0	11975	16	2.49	886
MLGA	$pp_{ij} = M(M(M(b_{2i+1}, 0, M(\overline{b}_{2i}, \overline{b}_{2i-1}, \overline{a}_j)), M(b_{2i}, 1, M(b_{2i-1}, \overline{a}_{j-1}, 1)), 0), M(M(\overline{b}_{2i+1}, 0, M(\overline{b}_{2i}, \overline{b}_{2i-1}, a_j)), M(M(\overline{b}_{2i+1}, b_{2i}, 0), M(b_{2i-1}, a_{j-1}, 0), 0), 1), 1)$ $Neg_i = M(b_{2i+1}, \overline{b}_{2i}, \overline{b}_{2i-1})$	12 1	-	-	-	-	497	106	24	0	0	11928	16	2.58	919
MLGB	$pp_{ij} = M(M(M(M(b_{2i}, b_{2i-1}, \overline{a}_j), b_{2i+1}, 0), M(\overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j), \overline{b}_{2i+1}, 0), 1), M(M(\overline{a}_{j-1}, M(\overline{b}_{2i-1}, \overline{b}_{2i}, 0), 0), M(b_{2i}, M(b_{2i-1}, a_{j-1}, 0), 0), 1), 1)$ $Neg_i = M(b_{2i+1}, 1, M(\overline{b}_{2i}, \overline{b}_{2i-1}, 0))$	10 2	-	-	-	-	444	138	24	0	0	10656	16	2.30	821
[11]	$app_{ij} = M(M(b_{2i+1}, \overline{M}(b_{2i}, b_{2i-1}, a_j), 0), M(\overline{b}_{2i+1}, M(b_{2i}, b_{2i-1}, a_j), 0), 1)$	4	positive	6	-	-	638	142	25	1.3	14.2	15950	16	3.31	1180
				8	-	-	518	142	25	7.2	95.6	12950	16	2.69	958
AMLCG	$app_{ij} = M(M(b_{2i+1}, \overline{M}(b_{2i}, \overline{b}_{2i-1}, a_j), 0), M(\overline{b}_{2i+1}, M(b_{2i}, b_{2i-1}, a_j), 0), 1)$	4	positive	6	3	-	329	102	20	1.1	12	6580	12	1.71	608
	$app_{0j} = M(M(\overline{b}_1, M(b_0, 0, a_j), 0), M(b_1, \overline{a}_j, 0), 1)$	4	unbiased	8	2	-	325	106	20	3.3	17	6500	13	1.69	601
	$app_{00} = M(M(b_1, \overline{M}(b_0, 0, a_0), 0), M(\overline{b}_1, M(b_0, 0, a_0), 0), 1)$	4													
	$app_{ij} = M(M(M(b_{2i+1}, \overline{a}_j, 0), \overline{M}(b_{2i}, b_{2i-1}, a_j), 0), M(M(\overline{b}_{2i+1}, a_j, 0), M(b_{2i}, b_{2i-1}, a_j), 0), 1)$	6													
AMLGA	$app_{0j} = M(M(\overline{b}_1, M(\overline{b}_1, b_0, a_j), 0), M(b_1, \overline{a}_j, 0), 1)$	4	unbiased	4	-	-	435	76	21	0.14	0.61	9135	16	2.26	804
	$app_{00} = M(M(b_1, M(\overline{b}_0, 1, \overline{a}_0), 0), M(\overline{b}_1, M(\overline{b}_0, 1, a_0), 0), 1)$	5													
	$app_{ij} = M(M(b_{2i+1}, M(\overline{b}_{2i}, \overline{b}_{2i-1}, \overline{a}_j), 0), M(\overline{b}_{2i+1}, M(\overline{b}_{2i}, \overline{b}_{2i-1}, a_j), 0), 1)$	5													
AMLGB	$app_{ij} = M(M(b_{2i+1}, M(b_{2i}, b_{2i-1}, \overline{a}_j), 0), M(\overline{b}_{2i+1}, \overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j), 0), 1)$	4	negative	8	-	5	258	84	19	5.2	86	4902	16	1.34	477
	$app_{0j} = M(M(\overline{b}_1, M(\overline{b}_1, b_0, a_j), 0), M(b_1, \overline{a}_j, 0), 1)$	4	unbiased	6	1	-	363	106	21	0.71	3.5	7623	14	1.88	671
	$app_{00} = M(M(b_1, M(\overline{b}_0, 1, \overline{a}_0), 0), M(\overline{b}_1, M(\overline{b}_0, 1, a_0), 0), 1)$	5													
	$app_{ij} = M(M(b_{2i+1}, M(M(b_{2i}, b_{2i-1}, \overline{a}_j), \overline{a}_j, 1), 0), M(\overline{b}_{2i+1}, M(\overline{M}(b_{2i}, b_{2i-1}, \overline{a}_j), a_j, 1), 0), 1)$	6													

MV: the number of required majority voters for a single pp_{ij} or Neg_i; MV_{AM}: the number of required majority voters in an approximate multiplier; INV_{AM}: the number of required inverters in an approximate multiplier; D: the delay for an approximate multiplier; ADP: MV $\times$ D; BW: bit-width of the final result.