OPACT: Optimization of Approximate Compressor Tree for Approximate Multiplier

Weihua Xiao[1], Cheng Zhuo[2], Weikang Qian[1,3,*]
[1]University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, China
[2]College of Information Science and Electronic Engineering, Zhejiang University, China
[3]MoE Key Laboratory of Artificial Intelligence, Shanghai Jiao Tong University, China
Email: 019370910014@sjtu.edu.cn, czhuo@zju.edu.cn, qianwk@sjtu.edu.cn; *corresponding author

***Abstract*—Approximate multipliers have attracted significant attention of researchers for designing low-power systems. The most area-consuming part of a multiplier is its compressor tree (CT). Hence, the prior works proposed various approximate compressors to reduce the area of the CT. However, the compression strategy for the approximate compressors has not been systematically studied: Most of the prior works apply their ad hoc strategies to arrange approximate compressors. In this work, we propose OPACT, a method for optimizing approximate compressor tree for approximate multiplier. An integer linear programming problem is first formulated to co-optimize CT's area and error. Moreover, since different connection orders of the approximate compressors can affect the error of an approximate multiplier, we formulate another mixed-integer programming problem for optimizing the connection order. The experimental results showed that OPACT can produce approximate multipliers with an average reduction of 24.4% and 8.4% in power-delay product and mean error distance, respectively, compared to the best existing designs with the same types of approximate compressors used.**

***Index Terms*—Approximate Multipliers, Compressor Tree, Approximate Compressors, Integer Programming, Optimization**

I. Introduction

Approximate computing, an emerging computing paradigm, designs highly energy-efficient computing systems at the cost of slight accuracy drop [1], [2]. This property well matches many recent error-tolerant but power-hungry applications, such as data mining and deep learning [3], [4]. As multipliers play an important role in many computation-intensive applications, many works focus on designing approximate multipliers.

A multiplier has three main parts: partial product generator (PPG), compressor tree (CT), and carry propagation adder (CPA). PPG's output is a *bit matrix (BM)*, each entry of which is a partial product. CT reduces each column in BM to 1 or 2 remaining bits. For generating the final product, CPA is further applied to sum up CT's final output BM. Among the existing works on approximate multiplier design, some approximate the PPG part [5]–[7], while most of the others approximate the CT since it takes the most area in a multiplier.

Compressors are the basic component of a CT. The prior works on approximating CT proposed various approximate compressors, which can be divided into two types, those with unequally weighted outputs [8]–[10] and those with equally weighted outputs [11], [12]. However, the compression strategy for the approximate compressors has not been systematically studied: to build the CT, existing methods often arrange the approximate compressors in an ad hoc way. Moreover, the connection order of the approximate compressors in a CT has an impact on the final error, while this point is often overlooked [10].

In this paper, to address the above issues, we propose OPACT, a systematic method for optimizing approximate compressor tree for approximate multiplier. First, an integer linear programming (ILP) problem is set up to allocate compressors, aimed at co-optimizing the area and error of the CT (see Section III-A). Then, a mixed-integer programming (MIP) formulation is proposed to optimize the connection order for those compressors obtained from the previous ILP problem (see Section III-B). Due to space limit, we only present the methodology on approximate compressors with equally weighted outputs, which have smaller hardware cost but larger error than those with unequally weighted outputs. However, OPACT can be easily adapted to those with unequally weighted outputs. The experimental results showed that OPACT can produce approximate multipliers with an average reduction of 12.4%, 24.4%, and 8.4% in area-delay product (ADP), power-delay product (PDP), and mean error distance (MED), respectively, compared to the best existing designs using the same types of approximate compressors.

II. Preliminaries and Related Works

In this section, we discuss preliminaries and related works.

A. Error Metrics

We describe three typical error metrics used for measuring the error for approximate computing: error rate (ER), MED, and mean relative error distance (MRED) [13].

Suppose that the given accurate circuit has M input combinations in total and the i-th ($1 \leq i \leq M$) one occurs with probability p_i. Denote the approximate and the accurate outputs based on the binary radix encoding under the i-th input combination as $\hat{Y}_i$ and Y_i, respectively. ER gives the probability that the output of an approximate circuit is incorrect, calculated as $ER = \sum_{1 \leq i \leq M: \hat{Y}_i \neq Y_i} p_i$. MED and MRED are measures on the average error magnitude, calculated as $MED = \sum_{i=1}^{M} |\hat{Y}_i - Y_i| p_i$ and $MRED = \sum_{i=1}^{M} \frac{|\hat{Y}_i - Y_i|}{Y_i} p_i$, respectively.

B. Compressors

Compressors are the basic modules to construct a CT. In this work, exact 2:2 and 3:2 compressors and approximate 3:2 and 4:2 compressors with equally weighted outputs from [11] are

used.[1] An exact 2:2 (resp. 3:2) compressor, shown in Fig. 1(a) (resp. Fig. 1(b)), is a 1-bit half (resp. full) adder, which takes 2 (resp. 3) bits as inputs and outputs 2 bits, a sum bit S and a carry-out bit C.

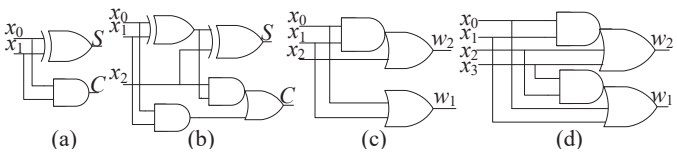


Fig. 1. Schematics of (a) exact 2:2 compressor, (b) exact 3:2 compressor, (c) approximate 3:2 compressor, and (d) approximate 4:2 compressor.

The approximate 3:2 and 4:2 compressors from [11] are shown in Figs. 1(c) and (d), respectively. Their outputs w_1 and w_2 have the same weights as their inputs. Hence, we call them *approximate compressors with equally weighted outputs*.

Assume that the *probability of being a* 1 (hereafter simply referred to as *probability*) for input x_i is p_i and that all the inputs are independent. Then, we can obtain from the truth table the output probabilities of the exact 2:2, exact 3:2, approximate 3:2, and approximate 4:2 compressors as shown in Eqs. (1), (2), (3), and (4), respectively.

$$\begin{aligned} &P(S=1) = p_0 + p_1 - 2p_0p_1, \\ &P(C=1) = p_0p_1. \end{aligned} \tag{1}$$

$$\begin{aligned} &P(S=1) = (p_0+p_1+p_2) - 2(p_0p_1+p_0p_2+p_1p_2) + 4p_0p_1p_2, \\ &P(C=1) = p_0p_1 + p_0p_2 + p_1p_2 - 2p_0p_1p_2. \end{aligned} \tag{2}$$

$$\begin{aligned} &P(w_1=1) = p_0 + p_1 - p_1p_0, \\ &P(w_2=1) = p_2 + p_1p_0 - p_2p_1p_0. \end{aligned} \tag{3}$$

$$\begin{aligned} &P(w_1=1) = p_1 + p_0 + p_3p_2 + 2p_3p_0 \\ &\quad + p_1p_0 - 2p_3p_1p_0 - p_3p_2p_1 - p_3p_2p_0 + p_3p_2p_1p_0, \\ &P(w_2=1) = p_2 + p_3 + p_1p_0 - p_3p_2 - p_2p_1p_0 - p_3p_1p_0 \\ &\quad + p_3p_2p_1p_0. \end{aligned} \tag{4}$$

For the approximate 3:2 and 4:2 compressors, we also list their MEDs in Eqs. (5) and (6), respectively.

$$MED_{3:2} = p_0p_1p_2, \tag{5}$$

$$MED_{4:2} = p_0p_1p_2 + p_1p_2p_3 + 2p_0p_1p_3(1-p_2). \tag{6}$$

C. Partial Product Generator and Compressor Tree

There are two major types of PPGs for a multiplier, AND gate-based and modified Booth encoding-based. In this paper, we only use AND gate-based PPG for illustration due to space limit. PPG's output is a BM. In Fig. 2, the BM BM_0 is the output BM of an 8-bit PPG. Its height is 8. A PPG's output BM is compressed to a BM with two rows by a CT. The *CT of an approximate multiplier* (hereafter simply referred to as *approximate CT*) consists of exact compressors and approximate compressors as shown in Fig. 2. If a 2:2 (resp. 3:2) compressor is applied at column j, then the bit number in column j reduces by 1 (resp. 2) and that in column $(j+1)$ increases by 1. If an approximate 3:2 (resp. 4:2) compressor is applied at column j, then only the bit number in column j reduces by 1 (resp. 2). An approximate CT has multiple compression stages. Fig. 2 shows an example of a compression process for an 8-bit approximate multiplier, which has two stages. According to the convention used in this paper, compression stage i compresses BM BM_{i-1} and produces BM BM_i, as shown in Fig. 2.

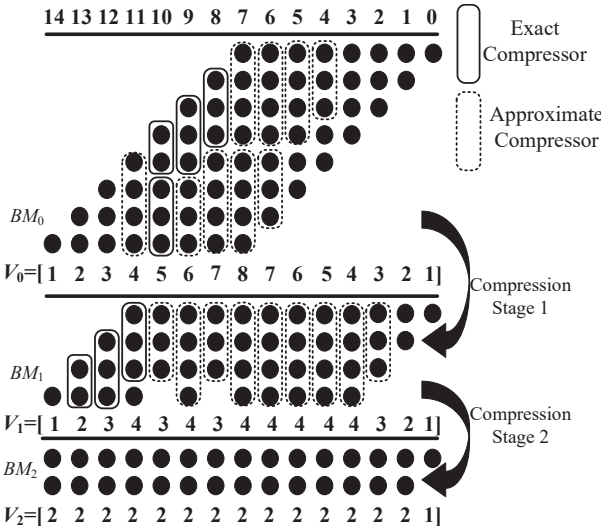


Fig. 2. A compression process of an 8-bit approximate multiplier.

D. Related Works

We describe several previous works on designing better approximate compression schemes. For an n-bit approximate multiplier, the work [11] divides the PPG's output BM into two parts: the least significant part (LSP) from the 0-th column to the $(n-1)$-th column and the most significant part (MSP) from the n-th column to the $(2n-2)$-th column. Each column of the LSP with more than 2 bits is compressed by an approximate compressor with input size equal to the bit count of the column. The MSP is compressed by the Dadda scheme but hybridly uses the approximate and the exact compressors. The work [12] empirically finds that hybridly using approximate and exact compressors at one column would increase the final error. Hence, it improves the scheme from [11] by using only exact or approximate compressors at one column. Moreover, it increases the number of columns in the LSP. To the best of our knowledge, the work [10] is the first one to claim that the connection order of the compressors in an approximate CT can affect the error and proposes a heuristic approach to optimize the connection order. However, it is manually designed and only applicable to its own compressors.

III. Optimization of Approximate Compressor Tree

In this section, we will present our proposed optimization method OPACT, aimed at co-optimizing the area of the approximate CT and the error of the approximate multiplier. The error metric we consider here is MED. OPACT has two stages. The first stage optimizes the allocation of the compressors at each column of each compression stage by co-optimizing the area and the MED of the approximate CT based on a

[1]Exact 2:2 and 3:2 compressors are the basic ones to build other larger exact compressors and thus give finer granularity for potentially building a better CT.

rough MED model (see Section III-A). We remark that since our approximate multiplier uses an accurate CPA, the MED of the approximate CT is just the MED of the approximate multiplier. After this stage, we determine the numbers of all types of compressors applied at each column of all BMs during the compression process. The second stage optimizes the connection order of all the compressors obtained from the first stage by minimizing an improved MED estimation of the approximate CT (see Section III-B).

A. Optimization of Compressor Allocation

In this section, we propose an ILP-based formulation to optimize the compressor allocation. Before describing our method, some definitions are first given. We define the word length of the multiplier as m. According to the description in Section II-C, CT reduces the PPG's output BM into a BM with two rows through multiple compression stages. We define the number of compression stages as s. In order to facilitate subsequent modeling and compare with other works fairly, s is fixed as the number of compression stages in [11], [12].

Same as [14], we model each BM by a *bit count vector (BCV)* $V = [x_{l-1}, x_{l-2}, \ldots, x_0]$, where l is the number of columns in this BM and x_j $(0 \leq j \leq l-1)$ is the number of bits in column j of the BM. We use V_0 to denote the BCV for the output BM BM_0 of the PPG; see Fig. 2 for an example. Generally, the BCV V_0 for an m-bit AND-gate based PPG is:

$$V_0 = (1, 2, \ldots, m-1, m, m-1, \ldots, 1).$$

During the compression process, the BM changes after each compression stage and the corresponding BCV changes. We denote the BCV for BM_i $(1 \leq i \leq s)$ as V_i. Fig. 2 shows the BCVs for BM_1 and BM_2, i.e., V_1 and V_2.

Generally, exact compressors can be applied at the left-most column of a BM, which will produce a new column of bits before the left-most one due to the generated carry signals. However, in our work, we enforce a constraint that no exact compressor is applied at the left-most column. This simplifies the formulation of the optimization problem, as the lengths of all BCVs remain with the same value of $(2m-1)$. Although this constraint limits the optimization space, the application of compressors at each BM's left-most column during the compression process is very rare. Hence, it has very little impact on the optimality of our solution.

We denote the numbers of exact 2:2, exact 3:2, approximate 3:2, and approximate 4:2 compressors applied at column j $(0 \leq j \leq 2m-2)$ of BM_i $(0 \leq i \leq s-1)$ as $h_{i,j}$, $f_{i,j}$, $p_{i,j}$, $q_{i,j}$, respectively. These four types of variables are unknowns to be solved in order to determine how to allocate compressors at each column of each compression stage. Additionally, we also takes the MED introduced by the approximate compressors into consideration. We denote the MEDs introduced by an approximate 3:2 and an approximate 4:2 compressor as e and E, respectively. At this stage, for simplicity, we assume that all bits in each BM during the compression process have a probability of 0.25, which equals the probability of each bit in BM_0 when the inputs to the multiplier are uniformly and independently distributed. Thus, e and E are fixed as $\frac{1}{64}$ and $\frac{7}{128}$ by Eqs. (5) and (6), respectively. Clearly, the above assumption is not necessarily true. We will improve the MED estimation of the approximate compressors in the next section.

Our proposed ILP formulation is as follows:

$$\min \quad \alpha H + \beta F + \gamma P + \delta Q + wD \tag{7}$$

$$s.t. \quad H = \sum_{i=0}^{s-1} \sum_{j=0}^{2m-2} h_{i,j}, \quad F = \sum_{i=0}^{s-1} \sum_{j=0}^{2m-2} f_{i,j}, \tag{8}$$

$$P = \sum_{i=0}^{s-1} \sum_{j=0}^{2m-2} p_{i,j}, \quad Q = \sum_{i=0}^{s-1} \sum_{j=0}^{2m-2} q_{i,j}, \tag{9}$$

$$d_{i,j} = p_{i,j} e + q_{i,j} E, \text{for } 0 \leq i \leq s-1, 0 \leq j \leq 2m-2, \tag{10}$$

$$D = \sum_{i=0}^{s-1} \sum_{j=0}^{2m-2} 2^j d_{i,j}, \tag{11}$$

$$h_{i,2m-2} = f_{i,2m-2} = 0, \text{ for } 0 \leq i \leq s-1, \tag{12}$$

$$h_{i,j} \geq 0, f_{i,j} \geq 0, \text{ for } 0 \leq i \leq s-1, 0 \leq j \leq 2m-3, \tag{13}$$

$$p_{i,j} \geq 0, \quad q_{i,j} \geq 0, \text{ for } 0 \leq i \leq s-1, 0 \leq j \leq 2m-2, \tag{14}$$

$$2h_{i,j} + 3f_{i,j} + 3p_{i,j} + 4q_{i,j} \leq V_i[j], \quad \text{for } 0 \leq i \leq s-1, 0 \leq j \leq 2m-2, \tag{15}$$

$$V_{i+1}[j] = V_i[j] - (h_{i,j} + 2f_{i,j} + p_{i,j} + 2q_{i,j}) + (h_{i,j-1} + f_{i,j-1}), \quad \text{for } 0 \leq i \leq s-1, 1 \leq j \leq 2m-2, \tag{16}$$

$$V_{i+1}[0] = V_i[0] - (h_{i,0} + 2f_{i,0} + p_{i,0} + 2q_{i,0}), \text{ for } 0 \leq i \leq s-1, \tag{17}$$

$$0 \leq V_s[j] \leq 2, \text{ for } 0 \leq j \leq 2m-2. \tag{18}$$

We aim to co-minimize the area and the introduced MED of all the compressors. Thus, the optimization target is given by Eq. (7). It is a linear combination of the areas and the MEDs of all the compressors. It has five constants α, β, γ, δ, and w. The first four constants represent the real areas of the exact 2:2, exact 3:2, approximate 3:2, and approximate 4:2 compressors synthesized in Nangate 45nm technology [15] by Synopsys Design Compiler [16], respectively. The last constant w is a parameter controlling the weight of MED in the optimization target. H, F, P, and Q are the total numbers of exact 2:2, exact 3:2, approximate 3:2, and approximate 4:2 compressors, respectively, calculated by Eqs. (8) and (9). D is the total MED introduced by all the compressors. To get D, we first calculate $d_{i,j}$, which is the MED introduced at column j of BM_i. It is calculated by directly summing up the MEDs of all the approximate compressors at the given column, as shown in Eq. (10). Then, the total MED D is calculated as a weighted sum of each column's MED, as shown in Eq. (11).

The constraint mentioned above that no exact compressor should be applied at each BM's leftmost column is implemented by Eq. (12). Eq. (15) corresponds to the requirement that the number of inputs of all the compressors applied at column j of BM_i should be no more than the bit count at this column, i.e., $V_i[j]$.

The bit count in column j of BM_i should satisfy Eq. (16), since by Section II-B, each exact 2:2, exact 3:2, approximate 3:2, and approximate 4:2 compressor applied at column j reduces the bit count of column j by 1, 2, 1, and 2, respectively, while each exact compressor applied at column $(j-1)$ increases the bit count of column j by 1. For the special case of column 0, the situation reduces to the one shown in Eq. (17). Finally, the bit count in each column of the BM produced by the final compression stage should be non-negative and no more than 2, which is modeled by Eq. (18).

We call this formulation of co-optimizing the area and MED of the approximate CT the optimization mode *CoOpt*. By changing the formulation slightly, we can also obtain two other optimization modes: (i) *AreaOpt*, which sets the MED D as a

constraint and minimizes the area; (ii) *ErrOpt*, which sets the area as a constraint and minimizes the MED D.

B. Optimization of Connection Order

Given the optimized compressor allocation from Section III-A, we further minimize the MED of the approximate CT by optimizing the connection order of all the allocated compressors.

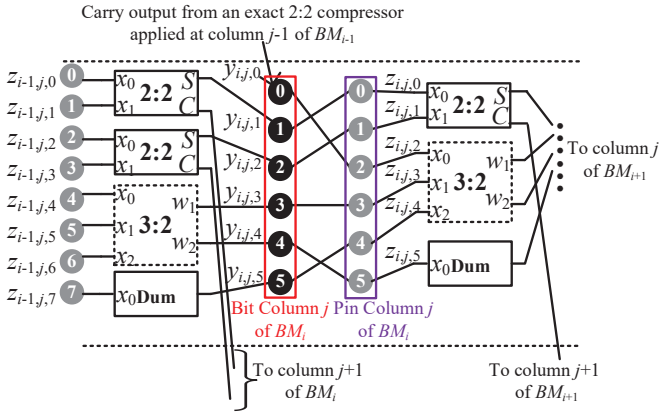


Fig. 3. Example of connection order.

The problem of optimizing the connection order determines how the bits in the column j of BM_i should be connected to the compressors applied at the column, which are determined from the allocation method described in Section III-A, for any i and j. We use Fig. 3 to illustrate the problem and our basic modelling. Assume that the column of 6 bits in black circles are those in the column j of BM_i. We call the column *bit column*. Assume that 1 exact 2:2 compressor (represented by a solid rectangle) and 1 approximate 3:2 compressor (represented by a dotted rectangle) are applied at this column. Since these two compressors only handle 5 bits at the column, 1 remaining bit at the column will be directly passed to the column j of BM_{i+1}. For convenience, we introduce a *dummy compressor*. In general, a k-bit dummy compressor directly forwards its k inputs to its k outputs. For this example, the dummy compressor, which is represented by a box "Dum" in the figure, has a single input and output. It passes bit 4 of the bit column to column j of BM_{i+1}. Including the dummy one, all the compressors applied at the bit column have 6 input pins in total. They are shown as a column of grey circles in Fig. 3. We call the column of all the input pins the *pin column*. The connection order problem decides how the bits of the bit column should connect to the pins of the pin column. Fig. 3 shows a possible connection order. It can also be viewed as a bijection between the bit column and the pin column.

Next, we will present our solution in detail. To formalize the connection order problem, the order of the bits in the bit column and that of the pins of the pin column should be specified. We first describe these orders in Sections III-B1 and III-B2, respectively. Then, we present the modeling of the bijection in Section III-B3. As the optimization target is MED, it is critical to improve the MED estimation, which we present in Section III-B4. Finally, we summarize the optimization problem in Section III-B5.

1) Order of the Bit Column: Consider an arbitrary bit column. Assume that it is column j of BM_i. There are seven sources for the bits in the bit column: Sources 1 and 2 are the carry outputs from the exact 2:2 and 3:2 compressors applied at column $(j-1)$ of BM_{i-1}, respectively; Sources 3 and 4 are the sum outputs from the exact 2:2 and 3:2 compressors applied at column j of BM_{i-1}, respectively; Sources 5 and 6 are the outputs from the approximate 3:2 and 4:2 compressors applied at column j of BM_{i-1}, respectively; Source 7 is the outputs of the dummy compressor applied at column j of BM_{i-1}. From the previous step of optimizing the compressor allocation, the number of compressors in Source k ($1 \le k \le 6$) is known. For example, the number of compressors in Source 1 is $h_{i-1,j-1}$.

The bits in a bit column are arranged in the order from Source 1 to Source 7. For example, Fig. 3 also shows the detailed sources for those bits in the bit column. Bit 0 is the carry output from an exact 2:2 compressor applied at column $(j-1)$ of BM_{i-1} (i.e., Source 1). Bits 1 and 2 are the sum outputs of two exact 2:2 compressors applied at column j of BM_{i-1} (i.e., Source 3). Bits 3 and 4 are the outputs of an approximate 3:2 compressor applied at column j of BM_{i-1} (i.e., Source 5). Finally, Bit 5 is the output of a dummy compressor applied at column j of BM_{i-1} (i.e., Source 7).

2) Order of the Pin Column: Consider column j of BM_i for an arbitrary i and j. Generally, there are five types of compressors applied at the column: Types 1 and 2 are the exact 2:2 and 3:2 compressors, respectively; Types 3 and 4 are the approximate 3:2 and 4:2 compressors, respectively; Type 5 is the dummy compressor. From the previous stage of optimizing the compressor allocation, the numbers of compressors in Types 1 to 4 are known, which are $h_{i,j}$, $f_{i,j}$, $p_{i,j}$, and $q_{i,j}$, respectively. The compressors are arranged in the order from Type 1 to Type 5. Then, the order of the input pins of each compressor is kept. This gives the order of all the pins of the pin column. For example, the pins of the pin column shown in Fig. 3 are ordered following the above rule.

3) Modeling of the Bijection: As we stated above, the connection order optimization is essentially to determine the best bijection between a bit column and its corresponding pin column. Assume that the bit column is column j of BM_i. Then, the sizes of both columns equal $V_i[j]$. We introduce a binary *connection variable* $b_{i,j,r,c}$ ($0 \le r, c \le V_i[j]-1$) to model the bijection. It is 1 if and only if bit r of the bit column connects to pin c of the pin column. Thus, the connection order optimization essentially solves for $b_{i,j,r,c}$'s to minimize the MED of the approximate CT. As each bit of the bit column connects to exactly one pin of the pin column and vice versa, $b_{i,j,r,c}$'s should satisfy the following two constraints:

$$\sum_{c=0}^{V_i[j]-1} b_{i,j,r,c} = 1, \text{ for } 0 \le r \le V_i[j]-1, \quad (19)$$

$$\sum_{r=0}^{V_i[j]-1} b_{i,j,r,c} = 1, \text{ for } 0 \le c \le V_i[j]-1. \quad (20)$$

4) Improved MED Estimation: In the previous step of optimizing the compressor allocation, we estimate the MED of each approximate compressor by Eqs. (5) and (6), assuming that the input probabilities p_i's of each compressor are 0.25. This leads to a very rough estimation. In this step of optimizing

the connection order, we try to improve the MED estimation of each approximate compressor. This requires an improved estimation of the input probabilities of each compressor. Fortunately, with the help of the connection variables and the knowledge of the compressor allocation, it is possible. In this section, we first show a method to improve the estimation to the input probabilities of the compressors. The basic idea is to propagate the probabilities from one compression stage to the next.

For column j ($0 \leq j \leq 2m-2$) of BM_i ($0 \leq i \leq s-1$), denote the probability of bit k ($0 \leq k \leq V_i[j]-1$) of the column and that of pin k of the corresponding pin column as $y_{i,j,k}$ and $z_{i,j,k}$, respectively. Our target is to get $z_{i,j,k}$'s. They are obtained recursively from $z_{i',j,k}$'s, where $i' < i$. Specifically, we first get $y_{i,j,k}$'s from the related $z_{i-1,j,k}$'s and $z_{i-1,j-1,k}$'s by applying Eqs. (1)–(4). Below is an example.

Example 1. *Consider the compressor allocation shown in Fig. 3. By Eq.* (1)*, the probability of the output S of the first exact $2:2$ compressor is $z_{i-1,j,0}+z_{i-1,j,1}-2z_{i-1,j,0}z_{i-1,j,1}$. By the ordering of the bit column defined in Section III-B1, the output S is bit 1 of the bit column. Thus, we have*

$$y_{i,j,1} = z_{i-1,j,0} + z_{i-1,j,1} - 2z_{i-1,j,0}z_{i-1,j,1}. \tag{21}$$

Note that for $1 \leq k \leq 5$ $y_{i,j,k}$ is calculated from the related $z_{i-1,j,k}$'s. For $y_{i,j,0}$, since it corresponds to the carry output of an exact $2:2$ compressor applied at column $(j-1)$ of BM_{i-1}, it is calculated from the related $z_{i-1,j-1,k}$'s.

Then, with the help of the connection variables $b_{i,j,r,c}$'s, we can obtain $z_{i,j,k}$'s from $y_{i,j,k}$'s as

$$z_{i,j,k} = \sum_{r=0}^{V_i[j]-1} y_{i,j,r} \cdot b_{i,j,r,k}. \tag{22}$$

Once $z_{i,j,k}$'s are obtained, we can apply Eqs. (5) and (6) to get the MED of each approximate compressor applied at the column. Then, by a calculation similar to Eqs. (10) and (11), we can get an improved MED estimation of the entire approximate CT.

5) Optimization Problem and Solution: The entire optimization formulation builds upon the previous components. It solves the binary connection variables $b_{i,j,r,c}$ to minimize the improved MED estimation of the approximate CT. Note the key variables involve the binary variables $b_{i,j,r,c}$ and the real variables $y_{i,j,k}$ and $z_{i,j,k}$. The key constraints involve those shown in the form of Eqs. (5), (6), (19)–(22). The details are omitted due to space limit. Since Eqs. (5), (6), and (21) are non-linear, the formulation is an MIP, which can be solved by an MIP solver.

IV. Experimental Results

In this section, we show the experimental results on OPACT. The MIP solver Gurobi Optimizer 9 [17] was used as the ILP/MIP solver. We generated the Verilog HDL code for approximate CTs according to the solutions of our ILP/MIP problems by MyHDL [18], which was further combined with the AND gate-based PPGs and the exact CPAs to build the approximate multipliers. We synthesized the approximate multipliers by Synopsys Design compiler [16] using Nangate 45nm technology [15] and measured their areas, delays, and dynamic powers. To measure their accuracy, the error metrics ER, MED, and MRED described in Section II-A were used. We set the inputs as uniformly distributed. For the multipliers with word length $m \leq 12$, we enumerated all the input patterns to obtain the error. Otherwise, we did Monte Carlo simulation with 2^{24} patterns.

A. Comparison of Single Design

In this section, we compared our proposed compression scheme with those heuristic ones [11], [12], which give the state-of-the-art designs using the same approximate compressors with equally weighted outputs as ours. We selected Kogge-Stone adder as the CPA following [11]. Four word lengths were considered: $m = 8, 12, 16, 20$. As shown in Section III-A, the optimization of compressor allocation has three modes *CoOpt*, *AreaOpt* and *ErrOpt* with different controlling parameters w, MED bound, and area bound, respectively. Different designs can be obtained for each mode by tuning their parameters. Since our proposed *OPACT* takes a long runtime for larger designs (e.g., when $m = 20$), we set a runtime bound for optimization as 600s.

For comparison with [11], we first used the design *1StepFull* from [11] as a reference. We generated three approximate multipliers with MED similar to it by the above three modes, respectively, and then selected the one with the minimal ADP. We call the design *OPACT1*. We also produced a design called *OPACT2* by OPACT in the same way using the design *2StepsFull* from [11] as a reference.[2] Furthermore, the *AM* designs from [12] are reported to be better than *2StepsFull*. Thus, we also compared *OPACT2* with *AM*. Specifically, we compared to *AM8-5* and *AM12-7* from [12], which are 8 and 12-bit approximate multipliers, respectively. Following the method from [12], we also designed *AM16-10* and *AM20-11*, which are 16 and 20-bit approximate multipliers, respectively.

Table I shows the comparison results on error and hardware cost, where *OPACT1* and *1StepFull* are put in one group for comparison, while *OPACT2*, *2StepsFull*, and *AM* are put in another. In each group, we highlight the best design in bold for each metric. From the table, we can see that for the groups with *OPACT1*, *OPACT1* has comparable hardware cost as *1StepFull*, while its MED is usually smaller. For all the groups with *OPACT2*, *OPACT2* are better than the other two existing designs in MED, area, delay, and power. Based on the data from Table I, Table II further lists the average improvement of an OPACT design to the one(s) in the same group over the 4 word lengths in MED, MRED, ADP, and PDP. Generally speaking, an OPACT design is better than its counterpart(s) in both error and hardware cost. The reason is because our proposed optimized compression scheme typically utilizes fewer exact compressors and more approximate ones, while maintaining a higher accuracy.

B. Comparison of Multiple Designs

Our proposed OPACT is able to produce multiple designs for one word length by tuning the parameters or changing

[2]Note that besides *1StepFull* and *2StepsFull*, two other approximate multipliers are proposed in [11]. However, they apply the approximate compressors to a truncated multiplier, while we do not. Thus, for a fair comparison, we did not compare to these two designs.

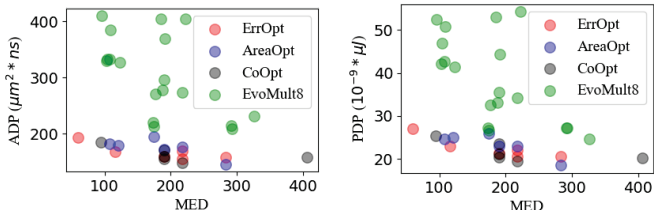
TABLE I
ERROR AND HARDWARE COST COMPARISON OF APPROXIMATE MULTIPLIERS.

size	multiplier	ER/%	MED	MRED	area/μm^2	delay/ns	power/μW
8*8	OPACT1	23.0	**14.6**	**0.0010**	**287.0**	0.84	**43.0**
	1StepFull	**22.0**	22.5	0.0021	292.0	**0.82**	44.7
	OPACT2	50.1	**283**	**0.0163**	**214.9**	**0.68**	**27.3**
	2StepsFull	**49.0**	352	0.0190	237.0	0.75	31.4
	AM8-5	50.1	**283**	0.0165	216.8	0.81	27.4
12*12	OPACT1	64.3	670	0.0006	**719.5**	**1.10**	**112.6**
	1StepFull	**36.0**	**650**	**0.0005**	730.7	**1.10**	128.6
	OPACT2	86.0	**2.89E4**	0.01	**528.0**	**0.92**	**65.0**
	2StepsFull	**76.3**	3.60E4	**0.0088**	595.6	0.99	88.1
	AM12-7	78.3	2.95E4	0.0097	535.2	0.94	76.7
16*16	OPACT1	89.5	**2.34E4**	**0.00010**	**1302.8**	1.20	**211.9**
	1StepFull	**54.0**	2.60E4	0.00011	1311.9	**1.19**	253.9
	OPACT2	97.3	**1.07E6**	0.0028	**950.2**	**1**	**125.1**
	2StepsFull	91.0	2.60E6	0.0036	1040.3	1.05	165.8
	AM16-10	**89.0**	1.29E6	**0.0027**	987.1	1.13	155.3
20*20	OPACT1	96.7	**3.14E5**	**0.00001**	**2095.5**	1.45	**356.3**
	1StepFull	**67.0**	8.70E5	0.00002	2116.6	**1.33**	439.3
	OPACT2	99.0	**7.37E7**	**0.001**	**1522.6**	**1.24**	**189.6**
	2StepsFull	99.0	3.53E8	0.002	1648.9	1.31	280.1
	AM20-11	**96.2**	8.64E7	**0.001**	1586.4	1.39	262.3

TABLE II
AVERAGE IMPROVEMENT OF OPACT DESIGNS IN MED, MRED, ADP, AND PDP.

	MED/%	MRED/%	ADP/%	PDP/%
OPACT1 over 1StepFull	26.5	23.2	−0.7	11.3
OPACT2 over 2StepsFull	44.4	18.2	15.3	29.2
OPACT2 over AM	8.4	−1.4	12.4	24.4

the optimization mode. In this section, we considered the word length $m = 8$ and compared a set of designs optimized by OPACT to that of *EvoMult8* [19] in a two-dimensional space defined using ADP-MED or PDP-MED. *EvoMult8* is a collection of 8-bit approximate multipliers synthesized by an approximate logic synthesis technique based on an evolutionary algorithm. For the three optimization modes *AreaOpt*, *ErrOpt*, and *CoOpt* of OPACT, we selected 6 designs for each of them by tuning their corresponding controlling parameters. For the *EvoMult8*, we selected 18 top designs within the similar MED range as our selected OPACT designs.



(a) (b)

Fig. 4. Comparison in (a) the ADP-MED space and (b) the PDP-MED space between 8-bit OPACT-optimized designs and *EvoMult8* designs.

Figs. 4(a) and 4(b) show the comparison between the 8-bit OPACT designs and the *EvoMult8* designs in the ADP-MED and the PDP-MED spaces, respectively. Clearly, our 8-bit OPACT designs dominate the *EvoMult8* designs in both the ADP-MED space and the PDP-MED space. Moreover, the Pareto frontier of the designs obtained by different optimization modes of OPACT is dense, which indicates that the tuning of the controlling parameters is able to make a trade-off between the accuracy and the hardware cost at a fine granularity.

V. CONCLUSION

In this work, we propose OPACT, a novel technique for optimizing the approximate compressor tree for approximate multipliers. It consists of two stages: (i) optimizing the compressor allocation in the approximate CT; (ii) optimizing the connection order of these allocated compressors, aimed at minimizing the MED. Approximate multipliers optimized by OPACT have a lower PDP than the state-of-the-art approximate multipliers, while maintaining a smaller MED. We note that in the current MED estimation, it is based on an assumption that the inputs to a compressor are independent, which is not necessarily true. A future direction is to further take the correlation among the inputs into account.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China Grant 2020YFB2205501 and National Natural Science Foundation of China Grants 62034007 and 61974133.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *ETS*, 2013, pp. 1–6.
[2] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing [point of view]," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 394–399, 2020.
[3] H. Jiang *et al.*, "Approximate arithmetic circuits: A survey, characterization and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.
[4] M. S. Ansari *et al.*, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE TC*, vol. 70, no. 4, pp. 614–625, 2021.
[5] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSID*, 2011, pp. 346–351.
[6] G. Zervakis *et al.*, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE TVLSI*, vol. 24, no. 10, pp. 3105–3117, 2016.
[7] M. S. Ansari *et al.*, "Low-power approximate multipliers using encoded partial products and approximate compressors," *IEEE JETCAS*, vol. 8, no. 3, pp. 404–416, 2018.
[8] R. Marimuthu, Y. E. Rezinold, and P. S. Mallick, "Design and analysis of multiplier using approximate 15-4 compressor," *IEEE Access*, vol. 5, pp. 1027–1036, 2017.
[9] Y. Guo *et al.*, "Low-cost approximate multiplier design using probability-driven inexact compressors," in *APCCAS*, 2018, pp. 291–294.
[10] A. G. M. Strollo *et al.*, "Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers," *IEEE TCAS-I*, vol. 67, no. 9, pp. 3021–3034, 2020.
[11] D. Esposito *et al.*, "Approximate multipliers based on new approximate compressors," *IEEE TCAS-I*, vol. 65, no. 12, pp. 4169–4182, 2018.
[12] M. Wang *et al.*, "An optimized compression strategy for compressor-based approximate multiplier," in *ISCAS*, 2020, pp. 1–5.
[13] I. Qiqieh *et al.*, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *DATE*, 2017, pp. 7–12.
[14] W. Xiao, W. Qian, and W. Liu, "GOMIL: Global optimization of multiplier by integer linear programming," in *DATE*, 2021, pp. 374–379.
[15] "Nangate 45nm open cell library," 2008. [Online]. Available: http://www.nangate.com/
[16] "Synopsys design compiler," 2012. [Online]. Available: http://www.synopsys.com
[17] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com
[18] "MyHDL," 2018. [Online]. Available: https://www.myhdl.org/
[19] V. Mrazek *et al.*, "EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *DATE*, 2017, pp. 258–261.