A Time Efficient Comprehensive Model of Approximate Multipliers for Design Space Exploration

Ziying Cui, Ke Chen, Bi Wu, Chenggang Yan, Yu Gong and Weiqiang Liu
Nanjing University of Aeronautics and Astronautics, Nanjing, China
{ziying.cui, chen.ke, wubi_sl, yancg, gongyu, liuweiqiang}@nuaa.edu.com

***Abstract*—Multipliers play an essential role in various data processing applications and have garnered significant attention in approximate computing (AxC) for their energy-efficient features. However, formulating a precise error model for approximate data processing algorithms in conjunction with hardware metrics presents a challenge, leading to substantial time consumption in the design space exploration. This paper introduces an analytical model for approximate multipliers while considering input patterns. This model furnishes accurate error metrics, along with high-precision hardware metrics for various approximate multiplier configurations, impervious to variations in input data distribution. The proposed error model reduces the runtime by an average factor of 120.85 and, in some instances, by as much as 2,500 times, when contrasted with simulation-based methods. The design space exploration is performed on a 3×3 convolution circuit, revealing a comparable Pareto-optimal set and substantial reductions of up to 79.46% in the Power-Delay-Product (PDP) and 71.98% in area compared to the accurate counterpart. Additionally, the result of the Gaussian Blur application experiment demonstrates a 68.59% reduction in PDP and a 56.21% reduction in area, all while maintaining a PSNR of 30 dB.**

***Index Terms*—Approximate multipliers, Analytical error modeling, Design space exploration**

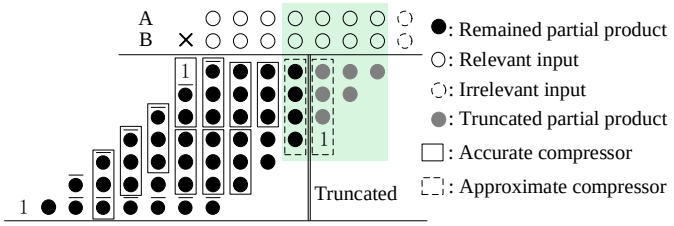
I. Introduction

Approximate computing technology, relying on a mature CMOS manufacturing process, has been proven to be a practical approach to improving energy efficiency for fault-tolerant applications such as artificial intelligence and computer vision. As a fundamental computational element, the multiplier is widely utilized in computationally intensive applications and contributes significantly to VLSI circuits' power and area consumption. Consequently, recent years have witnessed a proliferation of research works dedicated to the development of approximate multipliers aimed at substantial reductions in power consumption and silicon footprint across diverse design contexts [1]–[4].

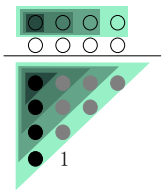
Therefore, in the pursuit of identifying the most suitable approximate multiplier for a specific application among the various available design alternatives, the presence of a comprehensive model encompassing both accuracy metrics and hardware metrics holds significant importance in Electronics Design Automation (EDA). While the commonly employed Monte Carlo simulation technique has gained widespread acceptance, it presents certain challenges when determining the optimal number of iterations due to the inherently pseudo-random nature of computer-generated random numbers. Additionally, this technique fails to account for the intricate relationship between error and input distribution, as well as the various modes of approximation that could provide valuable insights for making hardware design decisions.

Conversely, the exhaustive approach, which involves considering every conceivable input case, is only feasible for small-scale circuits, as its execution time experiences exponential growth with increasing input bit-width. To address these formidable challenges, the research community has introduced various analytical or semi-analytical error models in recent years. However, these models often exhibit specialization for specific applications or are burdened by computationally intensive requirements, rendering them less practical for integration into the actual hardware design process. Moreover, formal methods and Bayesian network models are employed for the assessment of the quality of approximate circuits within a reasonable time frame. It should be noted that formal methods necessitate more specialized input representations in comparison to high-level descriptions, as opposed to analytical methods. Conversely, the Bayesian network model is constrained by several error metrics.

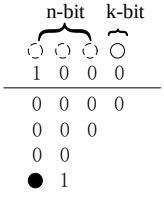
This paper presents a precise input-aware analytical method to address the prevailing issue. This method offers the flexibility to extend its application for modeling approximation errors arising from partial product generation, compression, and accumulation. Moreover, it capitalizes on the one-to-many relationships that exist between various relevant partial products and input combinations to derive precise metrics, thereby yielding a significant reduction in computational runtime compared to conventional simulation-based methods. The proposed model is employed in the context of design space exploration for a 3×3 convolution circuit to demonstrate its viability and effectiveness. Quality assessment is performed by replacing the conventional Monte Carlo simulation with the analytical error model. To avoid excessive computational time, the design space is pruned using the Monte Carlo Tree Search (MCTS) technique. In addition, the hardware metrics such as circuit area, power consumption and latency are also estimated. As a result, the optimal design selection from the Pareto set, guided by the proposed error model, yields an impressive 79.46% reduction in Power-Delay-Product (PDP) and 71.98%



(a) Partial product generation and first-stage compression



(b) Demonstration of relevance



(c) Example when k=1

Fig. 1: Demo of an approximate multiplier with 8-bit inputs, using 1-bit input-truncation, 3-bit partial-product-truncation and compensation and approximate compressors.

reduction in area compared to its accurate counterpart when MSE is constrained to 1×10^7. Additionally, it achieves a similar Pareto-optimal set compared to that using the Monte Carlo simulation, with a tenfold decrease in running time. The proposed method is also applied to Gaussian Blur, using a constant coefficient multiplier compared to the previous method. When constraining the accuracy to 30 dB, 68.59% of the PDP and 56.21% of the area are reduced compared to the exact replica, indicating a favorable trade-off.

II. Proposed Comprehensive Model for Approximate Multipliers

In this section, the proposed analytical input-aware model of configurable approximate multipliers, including error and hardware metrics, is presented in detail. The model is mainly applicable to the deliberately designed approximate multiplier, which is divided into partial product generation, partial product compression and final accumulation to approximate. The approximate methods comprise approximate Booth encoding and decoding [2], input truncation and compensation [5], partial product truncation and compensation [1], and approximate compressors [3], which are configurable parameters of the model. Fig. 1a shows a typical representation of this kind of multiplier. In design space exploration, the range and diversity of errors generated by approximate multipliers are more important than the method of approximation itself. For other types of multiplier [6], only one traversal is needed to obtain the error metrics corresponding to any input distribution. In certain scenarios, the specific characteristics of the input distribution for the multiplier may either be unknown or exhibit substantial variations across different use cases. A uniform distribution assumption is considered during the design phase for this situation. As such, an explicit error model for uniform input distributions is introduced, offering enhanced computational speed with full accuracy.

A. Proposed Model for Accuracy Metrics

1) MED: Mean Error Distance. In conventional error calculation procedures, large amounts of multiply-accumulate are used, which means a significant amount of runtime in software implementation. Consider a multiplier using it-bit input-truncation with two N-bit operands, A and B, and their exact output $E = A \times B$. To improve clarity, A is divided into the truncated part A_0 and the remaining part A_{21}, having $A = A_{21}2^{it} + A_0$, and so do B. The error distance (ED) can be written in

$$\begin{aligned} ED &= E - A_{21}2^{it} \times B_{21}2^{it} \\ &= 2^{it} \times (A_{21}B_0 + B_{21}A_0) + A_0B_0, \end{aligned} \tag{1}$$

so MED is

$$\begin{aligned} MED = \sum_{a_{21}=-Ed_{it}}^{Ed_{it}-1} \sum_{b_{21}=-Ed_{it}}^{Ed_{it}-1} \sum_{a_0=0}^{2^{it}-1} \sum_{b_0=0}^{2^{it}-1} \big(2^{it} \times (a_{21}b_0 + b_{21}a_0) \\ + a_0b_0) \times p_a p_b \\ Ed_{it} = 2^{N-it-1}, \end{aligned} \tag{2}$$

where p_a is the probability of A to be a and so do p_b. Input-aware is achieved by using the probability of each input case instead of that of a single bit, which effectively avoids errors caused by bit-to-bit correlation (except when the inputs adhere to a uniform distribution). Since the inputs are independent of each other, the equation can be modified into

$$\begin{aligned} MED =& 2^{it} \times \left(\sum_{a_{21}=-Ed_{it}}^{Ed_{it}-1} a_{21} \times p_{a_{21}} \sum_{b_0=0}^{2^{it}-1} b_0 \times p_{b_0} \right. \\ &\left. + \sum_{b_{21}=-Ed_{it}}^{Ed_{it}-1} b_{21} \times p_{b_{21}} \sum_{a_0=0}^{2^{it}-1} a_0 \times p_{a_0} \right) \\ &+ \sum_{a_0=0}^{2^{it}-1} a_0 \times p_{a_0} \sum_{b_0=0}^{2^{it}-1} b_0 \times p_{b_0} \\ p_{a_{21}} =& P\left(A_{21} = a_{21}\right) = \sum_{a_0=0}^{2^{it}-1} P(A = a_{21}2^{it} + a_0) \\ p_{a_0} =& P\left(A_0 = a_0\right) = \sum_{a_{21}=-Ed_{it}}^{Ed_{it}-1} P(A = a_{21}2^{it} + a_0), \end{aligned} \tag{3}$$

as the treatment of B is the same as that of A, it will not be reiterated here and will be maintained consistently throughout. Time is greatly reduced due to the reduction in loop iterations from 4 to 1. For uniform distribution, the equation can be further simplified into

$$p_{a_{21}} = 2^{it-N}, p_{a_0} = 2^{-it}. \tag{4}$$

By separating the constant probability from the loop, the summation equation is applied, eliminating the loop entirely.

Considering other approximate methods concerning partial products, such as partial product truncation, which depends on both inputs, a new paradigm needs to be proposed to compute

metrics quickly and accurately. Regarding this part, we focus on the partial products processed approximately, avoiding time-consuming exhaustion. Taking fig. 1a as an example, where four columns of partial products are approximated, the corresponding error only relates to the 4 least significant bits (LSBs) of both inputs after input-truncation, resulting in 2^8 considered combinations rather than 2^{16}. In other words, the ED is the same regardless of the other 4 bits when the $1 \sim 4th$ LSBs of the input are fixed. Therefore, it is sufficient to consider only one of these cases. Furthermore, it is evident from fig. 1b that only the high n bits of the four LSBs of A are relevant to the first n rows of partial products, as indicated in the colored box. However, if the n LSBs of B are all zero, the first n rows of partial products are also zero and cut the connection to the relevant high n bits of A, as shown in Fig. 1c. We have $n + k = 4$ here. To reduce the number of cases to be considered, classification is done based on the value of k ($0 < k \leq 4$, where all partial products are zero when k is 0). To clearly categorize, set the nth LSB of B to 1, while the lower bits are all 0. This classification reduces the number of cases to be considered from 256 to 170. The relevant EDs are calculated according to the input situations and pre-stored in a Look-up Table (LUT) and represented by ed_x^y, where x and y are index values and the use case is shown in II-A3. Hence, the proposed method can be expanded to any approximate methods related to partial products including approximate Booth coder/decoder, approximate compressors, approximate accumulation, etc.

For an approximate design using combinatorial methods, MED can be calculated separately and finally summed.

2) $MRED$: Mean Relative Error Distance. With a division added to MED, the computational complexity of $MRED$ is improved. Taking the design with input-truncation and partial-product-truncation for example, having

$$RED = \frac{ED^{it}}{E} + \frac{ED^{t}}{E}, \tag{5}$$

where ED^{it} and ED^t are EDs caused by input-truncation and partial-product-truncation. When $E = 0$, we add ED instead of RED, which will not be discussed here. Focusing on the $MRED$ caused by input-truncation, namely $MRED^{it}$:

$$\begin{aligned} MRED^{it} &= \sum_{a_{21}=-Ed_{it}}^{Ed_{it}-1} \sum_{b_{21}=-Ed_{it}}^{Ed_{it}-1} \sum_{a_0=0}^{2^{it}-1} \sum_{b_0=0}^{2^{it}-1} (1 \\ &\quad - \frac{a_{21} b_{21} 2^{2it}}{(a_{21}2^{it} + a_0)(b_{21}2^{it} + b_0)}) \times p_a p_b \\ &= 1 - \sum_{\substack{a_{21}=-Ed_{it} \\ a_{21}\neq 0}}^{Ed_{it}-1} \sum_{a_0=0}^{2^{it}-1} \frac{p_a}{\left(1 + \frac{a_0}{a_{21}2^{it}}\right)} \\ &\quad \times \sum_{\substack{b_{21}=-Ed_{it} \\ b_{21}\neq 0}}^{Ed_{it}-1} \sum_{b_0=0}^{2^{it}-1} \frac{p_b}{\left(1 + \frac{b_0}{b_{21}2^{it}}\right)} \end{aligned} \tag{6}$$

The equation is presented in 1's complement for simplicity. When it comes to $MRED^t$, a simplification is performed, which reduces the fourfold cycle to two or three.

3) $MAED$: Mean Absolute Error Distance. Different from MED and $MRED$, the error caused by different methods cannot be calculated separately with no special treatment during the calculation of $MAED$. Therefore, the error distance is separated into positive and negative ones, targeting different combinations of approximate means. Considering an approximate design using input-truncation and partial-product-truncation, it can be figured out by

$$AED = \left| ED^{it} + ED^{t} \right| \tag{7}$$

As ED^t is always non-negative due to sign bit handling in partial product generation and ED^{it} can be positive or negative, both should be taken into account. When both inputs are positive, ED^{it} is positive too. Therefore, we can divide $MAED$ into two parts, $MAED^+$ and $MAED^{\pm}$. The first part can be worked out easily as MED. The remaining part can be further written as

$$\begin{aligned} MAED^{\pm} &= \sum_{k=1}^{t} \sum_{m=0}^{2^{k-1}-1} \sum_{a_1=0}^{2^k-1} \sum_{a_0=0}^{2^{it}-1} \sum_{b_0=0}^{2^{it}-1} M^{AED} \\ M^{AED} &= \sum_{b_2=-Ed_{it+t}}^{Ed_{it+t}-1} \sum_{a_2=-Ed_{it+k}}^{Ed_{it+k}-1} \mid 2^{it} \times (a_{21}b_0 + b_{21}a_0) \\ &\quad + a_0 b_0 + 2^{2it} \times ed_{a_1}^m \mid \times p_a p_b \\ b_1 &= \mathrm{m}2^{t-k+1} + 2^{t-k} \\ a_{21} &= a_2 2^k + a_1, b_{21} = b_2 2^t + b_1 \\ Ed_{it+t} &= 2^{N-it-t-1}, Ed_{it+k} = 2^{N-it-k-1}, \end{aligned} \tag{8}$$

where a_{21} is separated into a_2 and a_1 for the sake of establishing contact with ED^t, so do b_{21}. M^{AED} is an intermediate variable and the situation where both A and B are positive needs to be eliminated, which is omitted here for simplicity. The $(t-k)th$ bit of b_1 is always 1 in each category, resulting in 2^{k-1} possibilities. As the sign of each error result must be judged before calculation, at most one cycle can be eliminated in software by figuring out the positive and negative critical value due to monotonicity. The more iterations of the untied loop, the greater the payoff. Therefore, A_2 is chosen. The critical value comes from

$$2^{it}\left(\left(A_2 2^k + A_1\right) \times B_0 + B_{21}A_0\right) + A_0 B_0 + 2^{2it} ed_{A_1}^m > 0 \tag{9}$$

After the absolute value is removed, it is treated in the same way as MED.

4) RMS_{ed}: Root Mean Square of error distance. To break down the terms, the square should be expanded in the equation. Similarly, a design using input-truncation and partial-product-truncation is taken as an example. The squared error is:

$$\begin{aligned} SED &= (ED^{it} + ED^{t})^2 \\ &= {ED^{it}}^2 + {ED^{t}}^2 + 2 \times ED^{it} \times ED^{t} \end{aligned} \tag{10}$$

After expanding the first term, which is a square of eq. (1), it is clear that it contains a factor $A_{21}A_0{B_0}^2$. Taking it as an example:

$$\begin{aligned}&\sum_{a_{21}=-Ed_{it}}^{Ed_{it}-1}\sum_{b_{21}=-Ed_{it}}^{Ed_{it}-1}\sum_{a_0=0}^{2^{it}-1}\sum_{b_0=0}^{2^{it}-1}a_{21}a_0{b_0}^2\times p_a p_b\\ =&\sum_{a_{21}=-Ed_{it}}^{Ed_{it}-1}\sum_{a_0=0}^{2^{it}-1}a_{21}a_0\times p_a\sum_{b_0=0}^{2^{it}-1}{b_0}^2\times p_{b_0}\end{aligned}\tag{11}$$

The quadratic sum equation is used here to avoid the cycle. When it comes to the last term of eq. (10), separating the loop related to ED^t first, and the rest can be treated in the same way as MED.

Since Var_{ed} (Variance of error distance) can be derived from MED and RMS_{ed} easily, it will not be explained here.

B. Proposed Model for Hardware Estimation of Approximate Multipliers

Within the context of design space exploration, the comprehensive evaluation of hardware parameters such as area, power consumption, and delay, alongside the accuracy of the design, holds paramount importance. This paper introduces a specialized hardware analytical model tailored to a specific multiplier, with the primary objective of mitigating time-related expenses.

1) Modelling for Circuit Area: The modeling of the area in this study employs partition counting due to the multiplier's composition of distinct modules with varying partitions and substantial reuse. Similar to the error model, the approximate multiplier is divided into three primary components: partial product generation, compression, and final accumulation. In the initial stage of partial product generation, the determination of the number of partial products to be generated relies on factors such as the input-truncation and partial-product-truncation bit counts. Accurate categorization of partial products necessitates consideration of factors such as the chosen algorithm, the spatial placement of partial products, and the potential application of compensation techniques. These factors collectively influence the spatial requirements of the partial product generation circuit. Moving to the compression stage, the accurate assessment of the requisite number of compressors for each type is feasible, given the consistent utilization of the same compression algorithm. In the accumulation stage, the calculation of the necessary quantities of half adders and full adders hinges on the characteristics of the last two partial products and whether truncated partial product compensation is incorporated into the design. The combinational area of an array multiplier can be represented as:

$$A_{com} = C_{ppg}A_{ppg} + C_{ppc}A_c + C_a A_a, \tag{12}$$

where A_{com} is the combinational area, C_{ppg} is the count of partial product generation circuits, A_{ppg} is the area of that, C_{ppc} is the count of partial product compressors, A_c is the area of compressors, C_a is the count of accumulation bits, A_a is the area of adders. Here, A_{ppg} is the area of an AND-gate. The paper proposes an algorithm that focuses on counting the number of individual basic constructs. The range of applications for the basic constructs can be expanded by adjusting them based on technology, approximate Booth encoding algorithm, and approximate compressor.

2) Modelling for Power Consumption: The research described in reference [7] showcases a strong correlation coefficient of 1 between the power consumption and the area of specific multiplier designs. Consequently, a statistical approach is employed to estimate power consumption based on the derived area values. A set of representative approximate multipliers is chosen for hardware description, followed by the synthesis process. Therefore, the curve fitting is performed to elucidate the relationship between power consumption and the area of these approximate multipliers. Notably, this relationship is separately modeled and fitted for combinational circuits, sequential circuits, and distinct coding algorithms due to their notable distinctions. The empirical data highlights a linear association between power consumption and the corresponding area for these designs. For example, the combination power consumption of an array multiplier is:

$$P_{com} = 0.0001768A_{com} - 0.004179, \tag{13}$$

where P_{com} is the power of combinational circuit respectively.

3) Modelling for Latency: To characterize the delay characteristics of the multiplier, it is necessary to partition it into three sequential stages, while also identifying the area and the longest propagation path within each stage. Additionally, it is imperative to determine the requisite number of compression steps for the partial product compression stage. In the final accumulation stage, it is important to note that counting should not initiate from the least significant bit. This is due to the presence of a non-rectangular array structure, where the compression of lower-order bits may conclude ahead of others. Moreover, it is crucial to consider that the counting result obtained during the partial product compression step is multiplied by the cumulative delay associated with the corresponding compressor. In contrast, the delay in the final accumulation stage must be scaled by the carry delay introduced by the half-adder or the full-adder. Taking an array multiplier as an example, the combinational delay is:

$$T_{com} = T_{ppg} + C_{ppct}T_{c\_s} + C_{ac}T_{a\_c}, \tag{14}$$

where T_{com} is the combinational delay, T_{ppg} is the delay of partial product generation, C_{ppct} is the count of partial product compression stages, T_{c_s} is the sum delay of compressors, C_{ac} is the count of accumulation bits done after the completion of compression, T_{a_c} is the carry delay of adders. Here, T_{ppg} is the delay of an AND-gate.

III. Evaluation and Analysis

To evaluate the performance and quality of the proposed model, several approximate multipliers are assessed using both analytical and Monte Carlo simulation methods, some of which also use exhaustive simulation, along with synthesis.

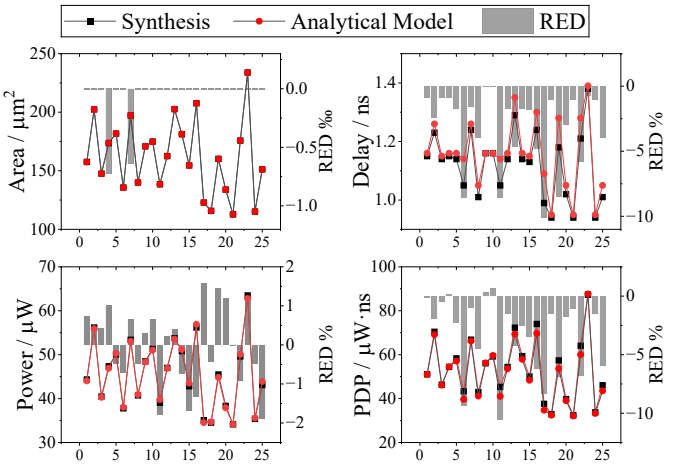


Fig. 2: Hardware evaluation using 25 representative multipliers for area, delay, power consumption and PDP.

The proposed model is implemented in Python on a computer with the following configurations: Intel(R) Core(TM) i7-8700 CPU, at 3.20 GHz, with 6 cores, 12 threads, 16GB RAM. To evaluate the hardware model, the selected approximate multipliers are described in Verilog HDL and synthesized using Synopsys' Design Compiler tool at 28nm technology.

A. Evaluation on Model Precision

1) Accuracy Model: Since software-implemented Monte Carlo simulation uses pseudo-random numbers as inputs and it is difficult to evaluate its accuracy, exhaustive simulation is used as the benchmark. Three representative approximate multipliers are selected to validate the model's accuracy and performance. The accuracy and runtime of the deduced four error metrics are shown in Table. I. As Var_{ED} is introduced by the other two metrics, it will not be analyzed here separately. Note that the experiments use uniform and Gaussian distributions for descriptive purposes only. The model proposed can be applied to cases with arbitrary input distributions.

The Table. I reveals that the model's error magnitude is exceptionally small, on the order of 10^{-13} or even smaller. It is crucial to emphasize that this minute error primarily stems from limitations in Python's computational precision rather than computational inaccuracies, thereby affirming the high precision of the error model. Notably, an observable trend in the operational time becomes evident as the multiplier bit-width increases. Specifically, the proposed model exhibits a consistent and linear increase in execution time, while the exhaustive simulation experiences a marked exponential growth. Comparing the three different multipliers presented, the model's computational speed demonstrates the potential to exceed that of the exhaustive simulation by a factor of 3000 or more for inputs with specific distributions. In the case of uniform distribution inputs, the model's efficiency can surge to be as much as 20000 times faster or even greater, thanks to the additional optimization incorporated within the model.

2) Hardware Model: To evaluate the hardware model, 25 representative multipliers are selected for hardware description and synthesis. The outcomes and those obtained using the analytical model are shown in fig. 2. The accuracy of the area modeling is evident, with a MRED of -0.055‰, and 23 out of 25 multipliers are exact. This is because the counting of the individual components of the approximate multiplier is completely accurate and the area is simple compared to the power and delay algorithms, which are less affected by fan-in and fan-out. Additionally, the area of the same components remains constant. As power consumption is determined by the area and has a strong correlation with it, the power consumption model is highly accurate, with a mean value of -0.28‰ and a maximum absolute RED of no more than 1.9%. The delay error primarily arises from the fact that reused parts within the same multiplier may have varying delays due to the effect of different fan-in and fan-out. This becomes particularly evident when the parts are complex and their number increases. The delayed MRED is -3.18% and the absolute RED is no more than 10%. As PDP is derived from power consumption and delay, its error is mainly caused by delay, with a MRED of 3.2% and a maximum of 10.5%. The error in the application algorithm is acceptable because the result is only used as a reference for iteration, and the final pareto set passes synthesis verification.

B. Evaluation on Model Performance

Since exhaustive simulation can be incredibly time-consuming, particularly for large bit-width, Monte Carlo simulation methods are typically preferred when analytical methods are not available. Therefore, to demonstrate the superiority of the proposed model, a comparison with Monte Carlo simulations will be conducted.

To demonstrate the general performance of the proposed model, the simulation time and accuracy are averaged for designs. The number of designs corresponding to the bit-width is 14, 28, and 35 for 8bit, 9bit or 10bit, and 12bit or 16bit separately, covering all kinds of configurable approximate multipliers. The approximate bit-width is 7, equal to the sum of input-truncation, partial-product-truncation, and bit-width using approximate compressors.

The evaluation results are shown in Fig. 3a to Fig. 3d. $MRED$ is used as an error metric for the Monte Carlo simulation method. Also, the logarithm of the ratio of the runtime for the Monte Carlo simulation and the error model is done to make it clear. The multiplier bit-width for all metrics are 8, 12, and 16 except for $MAED$, which uses 8, 9, and 10 due to the long simulation time. The figure also shows the time ratios between different bit-width multipliers when the number of Monte Carlo simulations is 5000, which is due to the fact that in most cases the multiplicity relation remains unchanged for different numbers of simulations.

As shown in fig. 3a, to obtain an error of MED of less than 10%, the 8bit and 12bit multipliers require about 50,000 iterations of Monte Carlo simulation, while the 16bit requires 500,000, which means hundreds of times the runtime of the proposed error model. In fig. 3b, comparing with MED, using Monte Carlo simulation to get $MRED$ with acceptable

TABLE I: The Accuracy and Runtime Compared with Exhaustive Simulation of MED, $MRED$, $MAED$, and RMS_{ed}

Input distribution	MED			$MRED$			$MAED$			RMS_{ed}		
	Diff /10^{-13}	ex_time/ m_time(s)	runtime ratio	Diff /10^{-15}	ex_time/ m_time(s)	runtime ratio	Diff /10^{-13}	ex_time/ m_time(s)	runtime ratio	Diff /10^{-13}	ex_time/ m_time(s)	runtime ratio
8-bit array multiplier with 2-bit input-truncation, 3-bit partial-product-truncation and compensation												
uniform	0	1.40 / 0.0012	1166.67	0	1.48 / 0.0031	477.42	0	1.47 / 0.088	16.70	0	1.45 / 0.0016	906.25
μ = 0 σ =100	4.09	1.51 / 0.0047	321.28	-0.059	1.57 / 0.0075	209.33	-3.41	1.61 / 0.34	4.74	-6.82	1.56 / 0.016	97.50
μ = 50 σ =200	8.6	1.5 / 0.0047	319.16	-0.247	1.6 / 0.0075	213.33	7.67	1.56 / 0.32	4.88	0.568	1.55 / 0.017	91.18
10-bit radix_4 booth multiplier with 3-bit input-truncation and compensation, 2-bit partial-product-truncation and compensation												
uniform	0	45.16 / 0.0021	21504.76	0	46.14 / 0.0077	5992.21	0	45.21 / 0.66	68.50	0	44.89 / 0.0029	15479.31
μ = 0 σ =100	3.77	27.34 / 0.008	3417.50	26	47.84 / 0.025	1913.60	23.3	46.88 / 3.77	12.44	3.41	47.54 / 0.038	1251.05
μ = 50 σ =200	2.82	46.64 / 0.0087	5360.92	1.11	47.72 / 0.025	1908.80	-235	47.92 / 3.83	12.51	-5.68	49.69 / 0.036	1380.28
12-bit radix_8 booth multiplier with 1-bit input-truncation and compensation, 4-bit partial-product-truncation												
uniform	0	377.84 / 0.018	20991.11	-141	388.10 / 0.087	4460.92	0	402.45 / 1.33	302.59	0	390.2 / 0.021	18580.95
μ = 0 σ =100	24.1	411.9 / 0.13	3168.46	-241	421.95 / 0.35	1205.57	1550	414.22 / 17.15	24.15	0.426	414.63 / 0.58	714.88
μ = 50 σ =200	75	407.36 / 0.13	3133.54	-17.3	419.75 / 0.31	1354.03	12000	426.53 / 17.28	24.68	-1.71	421.83 / 0.58	727.29

[1] μ and σ are the mean and variance of Gaussian distribution. Diff means the difference between exhaustive result and the model result in scientific notation. ex_time/m_time means the runtime of exhaustive simulation and the model.

accuracy needs more iterations and more runtime, up to 2,500 times slower than the error model. In fig. 3c, $MAED$ has little advantage over Monte Carlo simulations in terms of time due to the complexity of the equation and limited space available for optimization. In fig. 3d, For approximate multipliers with similar approximate bit-width and multiplier bit-width, using the proposed error model demonstrated a clear advantage in the evaluation of RMS_{ed}. Considering a 12-bit multiplier with an approximate bit-width of 7 and limiting the $MRED$ of the four error metrics to less than 10%, using the error model is on average 120.85 times faster than the Monte Carlo simulation. For the 8-bit multiplier with 7 approximate bits, the error of $MRED$ can be reduced to less than 10% until the simulation time reaches 2549.8 times the error model.

In summary, the error model and Monte Carlo simulation exhibit distinct characteristics when assessing various metrics across differing bit-width configurations. Specifically, when appraising $MRED$ and MED, the proposed model consistently demonstrates a pronounced advantage. Furthermore, the error model emerges as the preferred choice across all metrics, particularly in scenarios necessitating high precision or in situations where a comprehensive understanding of the interplay between iteration count and precision is lacking.

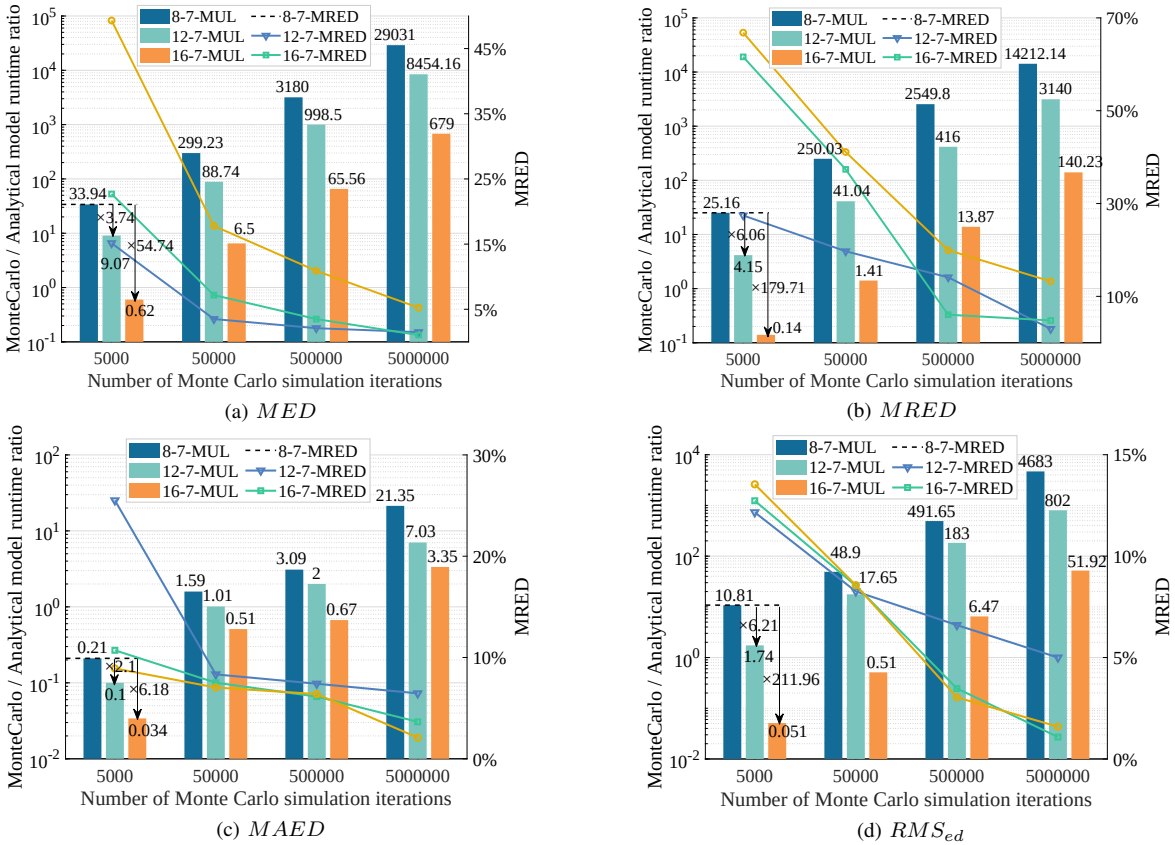
IV. Error Model Verification

The proposed model is utilized in the design space exploration [8] for a 3×3 convolution circuit to verify its effectiveness. The convolution circuit consists of nine parallel 8-bit approximate multipliers and an accurate adder tree. To showcase the universality of the error model, a Gaussian Blur application is also presented here. It follows the same structure as the MAC, except for the utilization of fixed coefficient multipliers. All designs among the Pareto-optimal set are described in Verilog HDL and synthesized using Synopsys' Design Compiler tool at 28nm technology for confirming.

To get the precision of the convolution circuit, the error computation method verified in [9] is employed. The proposed model replaces the MSE modeling of individual components in the algorithm. The mean squared error of the convolution can be modeled as the sum of the MSEs of all approximate multipliers. The hardware evaluation approach presented in [10] is adopted. Similarly, the hardware modeling of individual components is replaced with that proposed in this paper.

Next, the model is applied to the design space exploration of a 3×3 convolution circuit. The Monte Carlo Tree Search (MCTS) explained in [11] will be employed as the search technique. The optimization objectives are to minimize the power-delay product (PDP) and area respectively. For the derivation of errors in a single multiplier, a Monte Carlo simulation method using 5000 sets of inputs is used except for the model proposed for comparison. To guarantee reliability, the final results undergo verification through Monte Carlo simulation with 5 million sets of random numbers and synthesis using Synopsys' Design Compiler tool. The design space comprises 20 representative approximate multipliers, with a total design space size of 794 billion. Fig. 4 displays the Pareto-optimal set and the close solutions obtained after several iterations. The figure illustrates that the method proposed in this paper can generate Pareto-optimal sets of comparable quality to the Monte Carlo method while being 10 times faster, which comes from the runtime ratio of 5000 Monte Carlo simulations and the proposed model when calculating the RMS_{ed} of 8-bit approximate multipliers.

To demonstrate the practicality, the optimal values are selected by iterating the algorithm under four accuracy constraints. The value closest to the constraints is chosen from the



(a) MED (b) $MRED$

(c) $MAED$ (d) RMS_{ed}

Fig. 3: The time ratio of the Monte Carlo simulation method to the proposed method and the errors in the form of $MRED$ for (a) MED, (b) $MRED$, (c) $MAED$ and (d) RMS_{ed}.

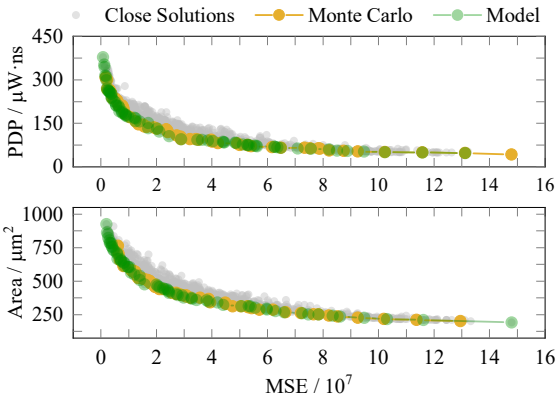


Fig. 4: The Pareto-optimal values using the proposed model and Monte Carlo method for PDP-minimization and area-minimization purposes along with close solutions separately.

TABLE II: The Optimal Designs for Different Constraints regard to PDP and Area

MSE_set /10^7	MSE_ex /10^7	PSNR_ex /dB	Synthesis	Analytial Model
PDP-optimal			PDP/$\mu W \cdot ns$	
0	0	$+\infty$	856.96	/
1	0.98	33.45	176.02	173.05
2	2.01	30.34	131.06	131.00
6	5.98	25.61	74.28	73.02
10	9.98	23.38	50.50	52.94
Area-optimal			Area/μm^2	
0	0	$+\infty$	2158.76	/
1	0.98	33.45	604.93	604.95
2	2.00	30.36	475.27	474.61
6	5.98	25.61	287.28	288.46
10	9.96	23.39	225.41	226.20

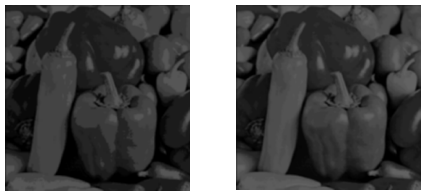
[1] MSE_set is the accuracy constraint. MSE_ex and PSNR_ex are the exact value of the optimal design.

set of Pareto-optimal values. Details are shown in Table. II. For the sake of fairness, the exact copy employs an 8-bit fully accurate multiplier, but rounds off the lower 7 bits before entering the addition tree, in accordance with the approximate MAC addition tree. The MRED of the PDP model error is below 2.1%, while that of the model error of the area is below 2.3‰. These results are consistent with the evaluation in III-A2, and the errors fall within an acceptable range. When the MSE is constrained to 1×10^7, the design space exploration yields an optimal design that achieves a 79.46% reduction of PDP in comparison to the exact counterpart when the optimal PDP is required. Similarly, the area can be reduced by 71.98% when the optimal area is required. This remarkable outcome is attributed to the high precision and speed offered by the recommended model, which facilitates the efficient utilization of redundancy while upholding the requisite accuracy constraints.

To further demonstrate the extensibility of the error model, the design space exploration methodology is applied to Gaus-

sian Blur. The chosen convolution kernel size is 3×3 with 8-bit precision. It is worth noting that Gaussian Blur uses fixed coefficient multipliers, whereas the error model can be adapted by changing the input distribution of the multipliers. For instance, if the weight is 120, the input probability is set to 1 at 120, and the probability of the other 255 numbers is 0. As the hardware model in this paper cannot be applied to the fixed coefficient multiplier, the synthesized results are used instead. The 25 commonly used 8-bit grayscale maps serve as a reference for obtaining another input distribution of multipliers. The objectives are PDP optimization and area optimization, with design accuracy limits of 20dB and 30dB, respectively. The final results are shown in Fig. 5. It is evident that when the PSNR reaches 30dB, the difference between the approximate image and the actual image perceived by the human eye is minimal. When limiting precision to 30 dB, the design space exploration algorithm yielded an optimal design with 68.59% less PDP and 56.21% less area compared to the exact counterpart. Compared to a MAC composed of regular multipliers, an exact fixed coefficient MAC is simpler and therefore has less room for optimization. However, it is still possible to achieve a satisfactory design with a good balance of accuracy and hardware overhead using the proposed approach.

(a) Area = 951.05 μm^2 PDP = 294.98 $\mu W \cdot ns$ (b) PDP, 20.37dB PDP = 34.32 $\mu W \cdot ns$ (c) PDP, 30.00dB PDP = 92.66 $\mu W \cdot ns$



(d) Area, 20.37dB Area = 221.89 μm^2 (e) Area, 30.04dB Area = 416.43 μm^2

Fig. 5: Resulting images of generated optimal solutions implementing Gaussian blur for different targets, where (a) is the exact result.

V. Conclusion

This paper presents an analytical model tailored to configurable approximate multipliers, which takes into account input distribution while retaining full precision for the error component. The proposed model offers adaptability, accommodating various approximate techniques. Through rigorous mathematical derivations, we have established five distinct quality metrics, namely, MED, $MRED$, $MAED$, RMS_{ed}, and Var_{ED}, in addition to four hardware metrics, each customized to suit different approximate techniques. In comparison to the Monte Carlo simulation method, our proposed model demonstrates a remarkable reduction in runtime, with an average decrease of 120.85, and in specific instances, as low as 2,500. This performance gain is particularly pronounced when precision constraints are set at 10% or higher. To emphasize the practical utility of our model, we apply it in the design space exploration of a 3×3 convolution circuit and Gaussian Blur, employing normal multipliers and fixed coefficient multipliers, respectively. Impressively, this utilization results in a design achieving a substantial 79.46% reduction in Power-Delay Product (PDP) and a 71.98% reduction in area when compared to the accurate counterpart in Multiply-Accumulate (MAC) operations. Additionally, in the context of Gaussian Blur, it becomes feasible to reduce the PDP by 68.59% and the area by 56.21% compared to the exact counterpart, while still maintaining an accuracy level of 30dB. Furthermore, it is worth noting that a similar Pareto-optimal set is generated, but with a runtime that is ten times faster.

Acknowledgment

This work is supported by grants from the National Key R&D Programme of China (2022YFB4500200), National Nature Science Foundation of China (62101252, 92364201).

References

[1] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant [for DSP]," Proceedings of IEEE Workshop on VLSI Signal Processing, Veldhoven, Netherlands, 1993, pp. 388-396.

[2] C. Xu, Z. Cui, K. Chen, and W. Liu, "Design and analysis of energy-efficient approximate Booth-folding squarers with precision recovery," Electronics Letters, vol. 58, no. 9, pp. 349–351, 2022.

[3] K. Chen, C. Xu, H. Waris, W. Liu, P. Montuschi and F. Lombardi, "Exact and Approximate Squarers for Error-Tolerant Applications," in IEEE Transactions on Computers, vol. 72, no. 7, pp. 2120-2126, 1 July 2023.

[4] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi and F. Lombardi, "Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 9, pp. 2856-2868, Sept. 2018.

[5] H. Jiang, L. Liu, P. P. Jonker, D. G. Elliott, F. Lombardi and J. Han, "A High-Performance and Energy-Efficient FIR Adaptive Filter Using Approximate Distributed Arithmetic Circuits," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 1, pp. 313-326, Jan. 2019.

[6] H. Kim, M. S. Kim, A. A. Del Barrio and N. Bagherzadeh, "A Cost-Efficient Iterative Truncated Logarithmic Multiplication for Convolutional Neural Networks," IEEE 26th Symposium on Computer Arithmetic (ARITH), Kyoto, Japan, 2019, pp. 108-111.

[7] Z. Vasicek, "Formal Methods for Exact Analysis of Approximate Circuits," in IEEE Access, vol. 7, pp. 177309-177331, 2019.

[8] S. Coward, T. Drane and Y. Harel, "Automatic Design Space Exploration for an Error Tolerant Application," IEEE 27th Symposium on Computer Arithmetic (ARITH), Portland, OR, USA, 2020, pp. 117-120.

[9] D. Sengupta, F. S. Snigdha, J. Hu, and S. S. Sapatnekar, "SABER: Selection of Approximate Bits for the Design of Error Tolerant Circuits," in Proceedings of the 54th Annual Design Automation Conference, 2017, pp. 1–6.

[10] J. Castro-Godínez, J. Mateus-Vargas, M. Shafique, and J. Henkel, "AxHLS: Design space exploration and high-level synthesis of approximate accelerators using approximate functional units and analytical models," in Proceedings of the 39th International Conference on Computer-Aided Design, 2020, pp. 1–9.

[11] M. Awais, H. G. Mohammadi and M. Platzner, "An MCTS-based Framework for Synthesis of Approximate Circuits," IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Verona, Italy, 2018, pp. 219-224.