IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 29, 2020, accepted January 10, 2021, date of publication January 22, 2021, date of current version May 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053578

Metrics, Noise Propagation Models, and Design Framework for Floating-Point Approximate Computing

YIYAO XIANG[1], LEI LI[1], SHIWEI YUAN[1], WANTING ZHOU[1], AND BENQING GUO[2], (Member, IEEE)

[1]University of Electronic Science and Technology of China, Chengdu 611731, China
[2]College of Communication Engineering, Chengdu University of Information Technology, Chengdu 610225, China

Corresponding author: Lei Li (lilei_uestc@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China NSAF Joint Fund under Grant U1630133 and in part by the National Defense Science and Technology Key Laboratory Fund under Grant 6142103190310.

ABSTRACT Approximate computing has emerged as an efficient solution for energy saving at the expense of calculation accuracy, especially for floating-point operation intensive applications, which have urgent demands for some uniform design frameworks for floating-point approximate computing combining the approximate computing techniques with the metrics of applications. In this paper, a simple approximate method with a zero-mean noise for the mantissa was introduced firstly, called PAM. Secondly, based on the proposed approximate method, the corresponding noise propagation models for floating-point operations were built, including floating-point addition, subtraction, and multiplication. Thirdly, a uniform design framework, which is only related to the operational-level topology of applications, was presented. The presented design framework can be used to evaluate the quality of data produced by applications before the circuit design is completed, and the efficient bit width of the mantissa can be obtained under specific requirements, which is also suitable for truncation. Finally, we studied the feasibility of the proposed design framework through two typical applications of image processing, edge detection and Gaussian filtering. The experimental results of edge detection have shown that our proposed design framework could effectively predict efficient bit width under the specific peak signal-to-noise ratio, with a difference of 1-2 bits in extreme situations. The Gaussian filtering experiment has demonstrated that the proposed design framework could apply to applications with complex calculations and structures.

INDEX TERMS Approximate computing, efficient bit width, noise propagation models, floating-point, mantissa, truncation.

I. INTRODUCTION

With the ever-increasing quest for performance and high integration, power consumption has become a crucial issue. Approximate computing (AC) has emerged as an efficient solution to save energy and improve performance by relaxing the requirement of accuracy and making the full advantage of the accepted error tolerance [1], [2]. This technique can be used in a wide range of power-hungry applications with inherent error resilience, including multimedia, data mining, pattern recognition, machine learning, and artificial intelligence [3]. To exploiting the corresponding margin through AC, energy efficiency can be improved aggressively [2].

Previous works have developed various approximate techniques to make the full advantage of inherent error resilience. Voltage over scaling (VOS)-centered methods employ low voltage supply to achieve high energy efficiency at the cost of computing accuracy, including algorithmic noise tolerance (ANT) [4], significance-driven computation (SDC) [5], and nonuniform voltage over scaling [6]. Several methods based on logic complexity reduction were proposed, covering algorithms [7], [8], logic [9], [10], gate [11], and transistor levels [12]. The majority of these works focus on fixed-point

The associate editor coordinating the review of this manuscript and approving it for publication was Qi Zhou.

computing. Actually, floating-point (FP) computing has been applied to a lot of power-hungry applications for its large dynamic range. The documentary [13] demonstrates that the fixed-point design would be almost 5 times larger and 40% slower than the corresponding FP unit with the same dynamic range maintained. Up to now, FP computing is used in many applications, ranging from climate modeling, electromagnetic scattering theory, image and signal processing, and the internet of things (IoT) [14], [15], especially those IoT applications that severely limited in their energy budget. Generally, with the corresponding complex processes taken into account, FP arithmetic units are very power-hungry. Therefore, it is significant to develop FP AP techniques.

Truncation has become a prevalent FP AP technique due to its simplicity and effectiveness. In the past, some new FP AP techniques were derived on the basis of truncation. Although some approximate approaches for FP adders and FP multipliers were proposed [16]- [19], it still lacks some design frameworks for FP AC to guide our design and help us to estimate the quality of data during the design phase, especially those uniform design frameworks that can combine the AC techniques with the metrics of applications. Such a framework is crucial for designers in project planning, and the framework that can be applied to truncation is more critical. If a framework applies to truncation, likely, it is also applicable to other FP AP techniques that are related to truncation.

Based on the tuning technique, Imani et.al [19] presented a framework for CFPU, which relies on real test cases. The fundamental significance of this paper is to establish a uniform framework that independent of data through a new FP AC technology. The contributions of this paper can be summarized as follows:

1) We proposed a simple approximate method like truncation, which enables voltage scaling. We called this approximate method ''Preapproximation'' (PAM), which can induce a zero-mean noise with a uniform distribution. It is convenient to build noise propagation models for approximate calculation units, which is of great significance to the establishment of the framework.
2) Based on the proposed approximate method and the occurrence probability, the corresponding noise propagation models (NPMs) for FP operations were derived. They are mantissa-independent and exponent-independent.
3) According to the derived NPMs for FP operations, we defined the noise propagation model of applications ANPM) and metrics to estimate the quality of applications. Integrating the proposed approximate method and the derived models with the metrics of applications, the design framework could be established. According to the operation-level topology, the requirement of an application can determine the needed efficient bit width of mantissa and vice versa. The proposed design framework is also suitable for truncation.
4) We studied the feasibility of the proposed design framework through two typical applications of image processing, edge detection and Gaussian filtering. The experimental results demonstrated that our proposed design framework can effectively predict efficient bit width under the specific peak signal-to-noise ratio, with a difference of 1-2 bits in extreme situations.

II. RELATED WORK

A. FP APPROXIMATE APPROACHES

Truncation is an energy-efficient and straightforward technique, which is often used in FP units design and approximate FP estimation, especially for bit-width optimization [20]–[23]. Yan *et al.* [24] proposed a configurable floating-point multiplier based on the K-nearest neighbor algorithm, called kNN-CAM, which lost 4.86% accuracy in exchange for 66.875% area-saving and 19.134% acceleration. Their method was carried out based on truncation, which could predict the efficient bit width needed by the new data to reduce the consumption and area at the largest range. Unfortunately, the circuit design is depended on the input data, and only the approximate multiplier was designed. Thus it is difficult to predict the efficient bit width needed by applications containing other operations. Mitchell's algorithm was used in [25] to design a configurable floating-point multiplier which achieved more than 25X power reduction by truncating 19 bits in the mantissa. In [26], the Mitchell algorithm was combined with truncation to propose an approximate log multiplier to save energy for the convolutional neural network (CNNs).

Almurib *et al.* [27] designed inaccurate adders by reducing the number of transistors. Camus *et al.* [17] combined two approximate circuit design techniques, namely gate-level pruning and inexact speculative adder, and completed three approximate 32-bit floating-point units (FPU). Test results demonstrated up to 27% power, 36% area and 53% power-area product saving, compared with the IEEE-754 single-precision FPU using 65nm process technology.

[19] proposed a configurable floating-point multiplier (CFPU) by introducing a tuning technique and declared that the CFPU could achieve 2.4X energy-delay product improvement compared with other approximate multipliers, with maintaining the same accuracy. The CFPU in [28] avoided the multiplication by finding and discarding one mantissa and directly using the other one. A fixed-point adder called reconfigurable approximate carry look-ahead adder (RAP-CLA) was used in [29] to design an approximate floating-point adder. Compared with the existing advanced floating-point adder, its power consumption and delay are increased by 7% and 21% respectively.

Although many approximate computing methods were proposed, truncation is still the most widely used method because it is simple and easy to implement and extend, especially for bit-width optimization. However, truncation cannot

offer a zero-mean error distribution [12] which can simplify the derivation of noise propagation model. We would make a detailed study in III-B2.

B. FP METRICS OF ARITHMETIC UNIT

In [30], the authors presented some metrics for fixed-point approximate adders, including error distance (ED), and mean error distance (MED). ED was defined as $ED(a, b) = |a - b|$, where a and b were the erroneous and correct results, respectively. MED was defined as the average value of EDs, $MED = E(ED)$. [30] normalized the error distance to the maximum value of error produced by an approximate adder.

[16] studied inexact FP adder design. Liu *et al.* [16] considered the inexact operations on both mantissa and exponent, extended the metrics in [30] and proposed a relative error distance (RED) to analyze the relationship between errors of inexact mantissa adder and inexact exponent adder. RED was defined as $RED = log_2(ED)$.

In [17], the authors took the relative error as metrics, which was defined as $|(a - b)/b|$, where a and b were the erroneous and correct results, respectively.

These metrics were proposed to analyze the errors of a single operation unit. However, it is limited to deal with the operation-level noise propagation and to analyze the errors of applications.

C. FP OPERATION-LEVEL NOISE PROPAGATION MODELS

Despite some works [31]- [33] focused on operation-level noise propagation, they just focused on fixed-point AC. The noise propagation of FP operations is different from that of fixed-point operations. The possible shift would cause lots of issues for analyzing the noise propagation of FP operations. To the best of our knowledge, no work addressing operation-level noise propagation of FP AC was found. In this paper, based on the occurrence probability, we proposed an estimation method.

In this paper, a floating-point approximation method similar to truncation would be proposed in section III, called ''Preapproximation'', which could facilitate us to build NPMs for multiplier, adder, and subtractor. Then the models and the overall design framework will be introduced. The feasibility of the whole framework would be proved in section IV through two experiments which are edge detection and Gaussian filtering.

III. NOISE PROPAGATION MODELS FOR FP OPERATIONS

A. NEW METRICS AND MODELS FOR FP OPERATIONS

In this paper, the metrics were defined in a different way to address the NPMs. ED was defined as

$$ED(a, b) = \text{a-b} \tag{1}$$

which is the relative error, whereas ED in [30] is the absolute value. Similar to [30], a and b in (1) are the erroneous and correct results, respectively. Instead of using ED directly, we treated ED as a variable and used its distribution in this paper.

Herein, we normalized three times of the standard variance of the final error distance distribution to the minimum mantissa value of the final results of the corresponding application instead of each ED to the maximum value of error in [30]. Thus, we combined all the noises with the requirement of the corresponding application. For some approximate applications, [34] presented the corresponding average error rates, which range from 3% to 10%.

B. APPROXIMATE SCHEMES AND THE INTRODUCED NOISES

1) PREAPPROXIMATION FOR THE INPUT OPERANDS

Herein, we introduced a new approximate approach to deal with the input operands. We assumed that the input operands are in floating-point number format, which consists of three parts: the sign bit, the exponent bits and the mantissa bits. Since the mantissa operation takes most of the energy of the corresponding operation, especially FP multiplication [18], our process is just on the mantissa. As full precision is not indispensable to approximate computation, the computation based on the more significant bits is sufficient to reach the requirements. The efficient bit width of the mantissa is defined as the needed more significant bits for FP operations. Herein, w denotes the efficient bit width of mantissa.

Once the efficient bit width of the mantissa is given, the less significant bits should be taken care. Without loss of generality, let us take single-precision as an example. $H.M_{[22:0]}$ denotes the mantissa of single-precision, where H is the hidden bit. In the analysis of this paper, the hidden bit was always taken as $1'b1$. PAM can be done like this: Retain $M_{[22:22-w+2]}$ bits, assign $M_{[22-w+1]}$ to be 1 and delete $M_{[22-w:0]}$, and this process as shown in Fig. 1. Thus, we have

$$H.M_{[22:22-w+2]}\#1 = H.M_{[22:0]} + U(\mu, \sigma^2) \tag{2}$$

where # is used for concatenating bits. $U(\mu, \sigma^2)$ is a discrete uniform distribution with $\mu = 0$ and $\sigma^2 = 1/(3 * 2^{2w})$. The corresponding explanation of (2) is given next. Without loss of generality, $M_{[n:0]}$ is in a discrete uniform distribution from 0 to $2^{n+1} - 1$. Assigning $M_{[n]} = 1$ means moving the mean value of this discrete uniform distribution from $(2^{n+1} - 1)/2$ to close to 0. Thus, an approximate discrete uniform distribution with $\mu = 0$ is introduced. When $H.M_{[22:0]} = H.M_{[22:22-w+2]}\#0\#0\cdots0$, the introduced noise is the largest

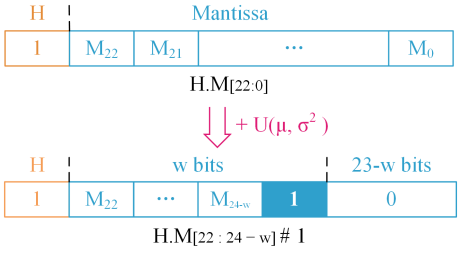

FIGURE 1. The process of PAM for the input operand, which is to set $M_{[22-w+1]}$ to 1 and delete $M_{[22-w:0]}$. This process is equivalent to introducing a zero-mean noise with a uniform distribution.

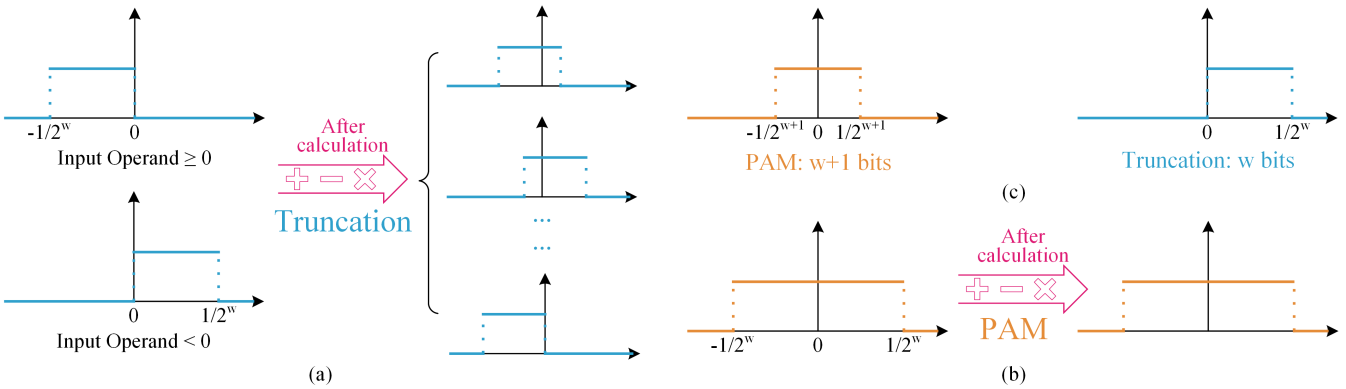

FIGURE 2. The noise distribution induced by truncation or PAM. Blue is correlated with truncation, and orange is associated with PAM. (a)The left part is the noise distributions induced by truncation, where the negative operand means that the positive noise distribution was induced. The right part is the possible noise distributions after approximate calculations. (b)The noise distribution induced by PAM. No matter what kind of operation, the mean of noise is zero. (c) The noise distributions of PAM with $w+1$ efficient bits and truncation with w efficient bits.

positive value, that is $ED = 1/2^w$. When $H.M_{[22:0]} = H.M_{[22:22-w+2]}\#1\#1\cdots1$, the introduced noise is the largest negative value, that is $ED = -(2^{23-w}-1)/2^{23}$, which is almost close to $-1/2^w$. Accordingly, PAM not only completes the approximate process, but also introduced a noise of $U(0, 1/(3*2^{2w}))$. $H.M_{[22:22-w+2]}\#1$ is used in approximate computation.

2) COMPARISON OF TRUNCATION AND PAM

It is well known that truncation is the widely used energy-efficient technique with enabling voltage scaling [12]. And it can offer the most energy-efficiency compared with other approximate techniques. But truncation has a non-zero mean noise, which will influence the final noises on the results.

Compared with PAM, truncation process is to delete $M_{[22-w:0]}$, and $H.M_{[22:23-w]}$ is used in approximate computation. The noise distributions induced by truncation need to be divided into two cases. When the operand is positive, the negative noise distribution is induced, and vice versa. In both cases, the mean of noise, μ, is non-zero. μ will participate in the calculations and then affect the final noise distribution. Therefore, operations such as addition, subtraction, and multiplication will make the noise distribution more complicated. Part of the noise distributions were plotted in Fig. 2 (a). The possible noise distributions generated by applications are shown in the figure. Therefore, it is challenging to establish the noise propagation model for truncation due to the complexity of the noise distribution.

To address this issue, we proposed the PAM to derive the noise propagation model. The most significant of PAM is that it could induce zero-mean noise. Therefore, the value of μ is always zero after any calculation, which can simplify the derivation of noise propagation model. It is only necessary to pay attention to the range of the final noise distribution. The noise distribution induced by PAM and the noise distribution after calculation can be represented in Fig. 2 (b). It should be noted that although PAM could induce zero-mean noise, the range of noise induced by PAM is twice of truncation under the same efficient bit width. As shown in the left graph of both Fig. 2 (a) and Fig. 2 (b). When the efficient bit width is w, the range of noise induced by PAM is $[-1/2^w, 1/2^w]$, and the range of noise induced by truncation is $[-1/2^w, 0]$ or $[0, 1/2^w]$. However, when the efficient bit width of PAM is $w+1$, the noise distribution is like the left graph in Fig. 2 (c). The range of noise distribution could be reduced to $[-1/2^{w+1}, 1/2^{w+1}]$, which is better than the noise caused by truncation with w efficient bits. Therefore, the relationship between truncation and PAM can be established by the Sandwich Theorem, which was represented by (3).

$$A_{PAM.w} < A_{T.w} < A_{PAM.w+1} \tag{3}$$

where $A_{PAM.w}$ and $A_{T.w}$ respectively represent the accuracy of PAM and truncation. In general, to achieve higher accuracy, PAM is required 1 bit more than truncation. However, the significant purpose of PAM is to establish the noise propagation model, which is of significant importance for approximate computing. The problem, which is challenging to establish the noise propagation model suitable for truncation, could be solved through PAM. Thus, PAM was proposed in this paper. The above analysis has shown that it is feasible to use PAM to indirectly establish the noise propagation model suitable for truncation.

This section presented how to deal with the input operands. In the next section, the approximate scheme used in the arithmetic units based on similar thinking would be provided.

3) THE APPROXIMATE SCHEME USED IN THE ARITHMETIC UNITS

As is well known, in FPU, integer adders are often employed to produce the final result, such as FP multiplication and FP subtraction. Herein, we presented an approximate scheme to deal with the approximation in FP multiplication, FP addition, and FP subtraction.

For addition, $Y_{[n:0]} = A_{[n-1:0]} + B_{[n-1:0]}$. For approximate computation, Y can be computed as $Y_{[n:f]} = A_{[n-1:f]} + B_{[n-1:f]} + 2^f$. Thus, some noise in a triangular distribution is introduced by replacing $A_{[f-1:0]} + B_{[f-1:0]}$ with 2^f. For simplifying our models, a discrete uniform distribution is used instead of this triangular distribution by assuming $Z_{[f:0]} = A_{[f-1:0]} + B_{[f-1:0]}$ is independent.

For FP multiplication, given the efficient bit width of the mantissa of w, based on our assumption, this approximation on the final adder will introduce a noise of $U(0, 1/(3*2^{2w}))$.

For FP addition and FP subtraction, there are some exponent differences. We fixed the position of the mantissa with the larger exponent and then aligned the position of the mantissa with the less exponent. When the exponent difference is less than and equals to 23, the total effect of approximation equals to introducing a noise of $U(0, 1/(3*2^{2w}))$. When the exponent difference is larger than 23, the operation will converge to PAM. Without loss of generality, sweeping all the possible exponent differences, the total effect of approximation can be seen as a noise of $U(0, 1/(3*2^{2w}))$.

C. FLOATING-POINT OPERATION NOISE PROPAGATION MODELS

Before establishing NPMs for FP operations, some assumptions should be made:

1) All the noises are independent.
2) The above proposed approximate schemes were employed.
3) We assumed that the final noise mainly comes from two parts, the input noise (IO) and the approximate noise (AO). IO refers to the PAM noise or the noise from the former stage. AO refers to the introduced noise by an approximate process in the corresponding operation.
4) We assumed the final noise of an application was with a normal distribution, according to the central limit theorem [44].

1) FP MULTIPLICATION

For FP multipliers, we assumed two operands, $H_A.M_{A[22:0]}$ and $H_B.M_{B[22:0]}$. After PAM, they become

$$\begin{cases} H_A.M_{A[22:24-w]}\#1 = H_A.M_{A[22:0]} + U(\mu, \sigma^2) \\ H_B.M_{B[22:24-w]}\#1 = H_B.M_{B[22:0]} + U(\mu, \sigma^2) \end{cases} \tag{4}$$

Combined with (4), FP multiplication can be written as:

$$\begin{aligned} &PI_{[1:0]}.PF_{[2w-1:0]} \\ &\quad = (H_A.M_{A[22:24-w]}\#1) * (H_B.M_{B[22:24-w]}\#1) \\ &\quad = (H_A.M_{A[22:0]}) * (H_B.M_{B[22:0]}) \\ &\qquad + (H_A.M_{A[22:0]} + H_B.M_{B[22:0]}) * U(\mu, \sigma^2) \\ &\qquad + U(\mu, \sigma^2) * U(\mu, \sigma^2) \\ &\quad \approx (H_A.M_{A[22:0]}) * (H_B.M_{B[22:0]}) \\ &\qquad + (H_A.M_{A[22:0]} + H_B.M_{B[22:0]}) * U(\mu, \sigma^2) \end{aligned} \tag{5}$$

where PI denotes the integer part of the product, and PF denotes the fraction part. It is well known that multipliers always use an adder to obtain the final results [35] and [36]. $PI_{[1:0]}.PF_{[2w-1:0]}$ will produce more bits than what the approximate multiplier needs. Another approximation is needed. From the above analysis, this approximation will introduce another noise, $U(\mu, \sigma^2)$, which is AO. After approximation, we have

$$\begin{aligned} &PI_{[1:0]}.PF_{[w-1:0]} \\ &\quad \approx (H_A.M_{A[22:0]}) * (H_B.M_{B[22:0]}) \\ &\qquad + (H_A.M_{A[22:0]} + H_B.M_{B[22:0]} + 1) * U(\mu, \sigma^2) \end{aligned} \tag{6}$$

Now we focus on the discussion about noise, which is $(H_A.M_{A[22:0]} + H_B.M_{B[22:0]} + 1) * U(\mu, \sigma^2)$, and denote it as N_{mul}. There are two cases:

1) $PI_{[1:0]} = 2'b01$
   When one of $H_A.M_{A[22:0]}$ and $H_B.M_{B[22:0]}$ is $1.1\cdots1$ and the other one is $1.0\cdots0$, N_{mul} gets the largest value. For simplicity, $H_A.M_{A[22:0]}+H_B.M_{B[22:0]}+1 = 4$ is chosen, therefore,
   $$N_{mul} \le 4 * U(\mu, \sigma^2) \tag{7}$$
2) $PI_{[1]} = 1'b1$
   For this case, one more right shift is needed for normalization, which equals to being divided by two. During the process of right shift, the noise is reduced as well. When $H_A.M_{A[22:0]}$ and $H_B.M_{B[22:0]}$ both are $1.1\cdots1$, N_{mul} gets the largest value.
   $$N_{mul} \le (4/2 + 1) * U(\mu, \sigma^2) = 3 * U(\mu, \sigma^2) \tag{8}$$

For FP multipliers, without loss of the generality, we assumed two operands, $H_A.M_{A[22:0]}$ and $H_B.M_{B[22:0]}$ with different noises, $\alpha * U(\mu, \sigma^2)$ and $\beta * U(\mu, \sigma^2)$, respectively. Based on the above derivation process, the noise on the final result should be

$$CN_{(mult,\alpha,\beta)} = \begin{cases} (H_A.M_{A[22:0]} * \alpha \\ +H_B.M_{B[22:0]} * \beta \\ +1) * U(\mu, \sigma^2), & PI_{[1:0]} = 2'b01 \\ (H_A.M_{A[22:0]} * \alpha/2 \\ +H_B.M_{B[22:0]} * \beta/2 \\ +1) * U(\mu, \sigma^2), & PI[1] = 1'b1 \end{cases} \tag{9}$$

where $CN_{(mult,\alpha,\beta)}$ is the cumulative noise. The first parameter in $CN_{(mult,\alpha,\beta)}$ denotes the operation name or application name. The second denotes the noise factor of the first operand. The last is the noise factor of the second operand.

By sweeping $\{M_{A[22:21]}, M_{B[22:21]}\}$, the probability of $PI_{[1]} = 1'b1$ is 3/8 and $PI_{[1:0]} = 2'b01$ is 5/8. Thus, the corresponding $CN_{(mult,\alpha,\beta)}$ can be given as

$$\begin{aligned} &CN_{(mult,\alpha,\beta),PI_{[1:0]}=2'b01} \\ &\quad = H_A.M_{A[22:0]} * \alpha + H_B.M_{B[22:0]} * \beta + 1 \\ &\quad \approx (1 + \frac{3}{10} * 0.5 + \frac{4}{10} * 0.25 + 0.5 * 0.25) * \alpha \\ &\qquad + (1 + \frac{3}{10} * 0.5 + \frac{4}{10} * 0.25 + 0.5 * 0.25) * \beta + 1 \\ &\quad = 1.375(\alpha + \beta) + 1 \end{aligned} \tag{10}$$

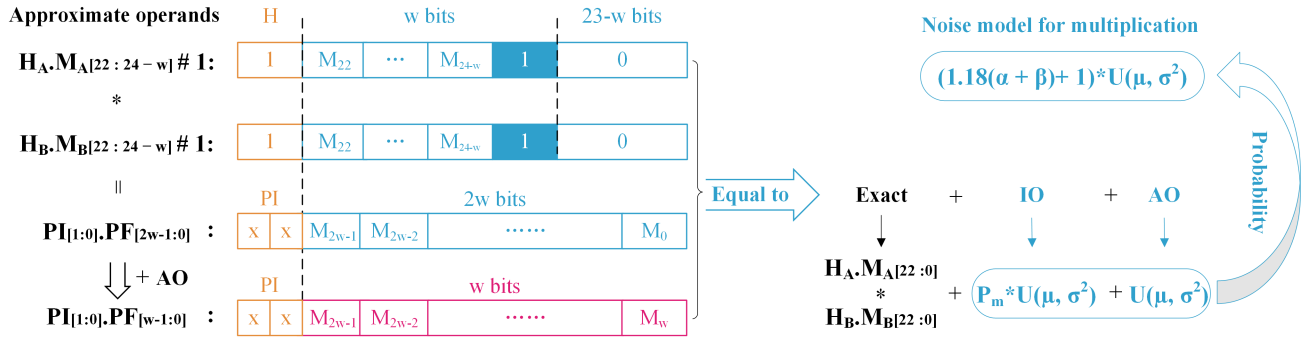

FIGURE 3. Noise propagation model of floating-point multiplier. They are mantissa independent and exponent independent. The left side of the figure is a schematic diagram of the multiplication. It can be equivalent to the right part of the figure, that is, the exact plus IO and AO, and finally evolved into a mantissa independent and exponential independent model according to the probability of occurrence, the upper right part.

and

$$\begin{aligned} &CN_{(mult,\alpha,\beta),PI_{[1]}=1'b1} \\ &= H_A.M_{A[22:0]} * \alpha/2 + H_B.M_{B[22:0]} * \beta/2 + 1 \\ &\approx (1 + \frac{5}{6} * 0.5 + \frac{4}{6} * 0.25 + 0.5 * 0.25)/2 * \alpha \\ &\quad + (1 + \frac{5}{6} * 0.5 + \frac{4}{6} * 0.25 + 0.5 * 0.25)/2 * \beta + 1 \\ &\approx 0.85 * (\alpha + \beta) + 1 \end{aligned} \tag{11}$$

Thus,

$$\begin{aligned} CN(mult, \alpha, \beta) &= CN_{(mult,\alpha,\beta),PI_{[1:0]}=2'b01} * P_{(PI_{[1:0]}=2'b01)} \\ &\quad + CN_{(mult,\alpha,\beta),PI_{[1]}=1'b1} * P_{(PI_{[1]}=1'b1)} \\ &= (1.375 * (\alpha + \beta) + 1) * 5/8 \\ &\quad + (0.85 * (\alpha + \beta) + 1) * 3/8 \\ &\approx (1.18 * (\alpha + \beta) + 1) \end{aligned} \tag{12}$$

Thus, a general model for FP multipliers is built, which is mantissa-independent and only care about the input noises, as shown in Fig. 3.

2) FP ADDITION

We also assume two operands, $H_A.M_{A[22:0]}$ and $H_B.M_{B[22:0]}$ with different noises, $\alpha * U(\mu, \sigma^2)$ and $\beta * U(\mu, \sigma^2)$, respectively. There are three cases according to the exponents.

1) $E_A > E_B$.
After alignment, the addition of mantissa can be expressed as

$$\begin{aligned} &PI_{[1:0]}.PF_{[w:0]} \\ &= H_A.M_{A[22:0]} + \alpha * U(\mu, \sigma^2) \\ &\quad + (H_B.M_{B[22:0]} + \beta * U(\mu, \sigma^2))/2^p \end{aligned} \tag{13}$$

where p is the exponent difference, $p = E_A - E_B$. From all possible FP additions in the used final adder, the total effect of approximation can be seen as a noise of $U(\mu, \sigma^2)$. Therefore, we have

$$CN_{(add,\alpha,\beta)} = (\alpha + \frac{\beta}{2^p} + 1) * U(\mu, \sigma^2) \tag{14}$$

Herein, for this case, possibly there is a carry-out. But we ignore this situation since the noises will be reduced by a 1-bit right-left. Moreover, this situation takes place in a low probability.

2) $E_A = E_B$
Regarding this case, the addition of mantises can be written as

$$\begin{aligned} PI_{[1:0]}.PF_{[w:0]} &= (H_A.M_{A[22:0]} + H_B.M_{B[22:0]}) \\ &\quad + (\alpha + \beta) * U(\mu, \sigma^2) \end{aligned} \tag{15}$$

A 1-bit right shift is needed for this case since there is a carry-out. A 1-bit right shift for noises means that they are divided by 2. Considering the total effect of approximation, $U(\mu, \sigma^2)$, we have

$$CN_{(add,\alpha,\beta)} = (\frac{\alpha + \beta}{2} + 1) * U(\mu, \sigma^2) \tag{16}$$

3) $E_A < E_B$
This case is similar to case 1). Therefore,

$$CN_{(add,\alpha,\beta)} = (\frac{\alpha}{2^p} + \beta + 1) * U(\mu, \sigma^2) \tag{17}$$

In total,

$$CN_{(add,\alpha,\beta)} = \begin{cases} (\alpha + \frac{\beta}{2^p} + 1) * U(\mu, \sigma^2), & E_A > E_B \\ (\frac{\alpha + \beta}{2} + 1) * U(\mu, \sigma^2), & E_A = E_B \\ (\frac{\alpha}{2^p} + \beta + 1) * U(\mu, \sigma^2), & E_A < E_B \end{cases} \tag{18}$$

which is exponent-dependent. Based on the occurrence probability, the final model can be obtained as

$$\begin{aligned} CN_{(add,\alpha,\beta)} &= CN_{(add,\alpha,\beta),E_A>E_B} * P_{(E_A>E_B)} \\ &\quad + CN_{(add,\alpha,\beta),E_A=E_B} * P_{(E_A=E_B)} \\ &\quad + CN_{(add,\alpha,\beta),E_A<E_B} * P_{(E_A<E_B)} \\ &< \frac{\alpha}{2} + \frac{\beta}{254} + \frac{\alpha + \beta}{508} + \frac{\alpha}{254} + \frac{\beta}{2} + 1 \\ &\approx \frac{\alpha + \beta}{2} + 1 \end{aligned} \tag{19}$$

which is exponent-independent, as shown in Fig. 4.

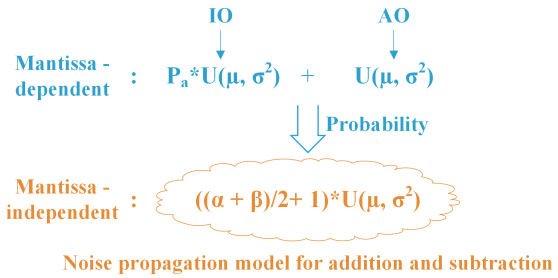

FIGURE 4. The noise propagation model of the floating-point adder, which is the same as the subtraction.

3) FP SUBTRACTION

We still used the same assumptions in FP addition. The model of FP subtraction is more complicated than that of FP addition. There are three cases.

1) $E_A > E_B$.
After alignment, the subtraction of mantissa can be expressed as

$$\begin{aligned} &PI_{[1:0]}.PF_{[w:0]} \\ &= H_A.M_{A[22:0]} + \alpha * U(\mu, \sigma^2) \\ &\quad - (H_B.M_{B[22:0]} + \beta * U(\mu, \sigma^2))/2^p \\ &= H_A.M_{A[22:0]} + \alpha * U(\mu, \sigma^2) \\ &\quad + (\overline{H_B.M_{B[22:0]}} + 1 + \beta * U(\mu, \sigma^2))/2^p \end{aligned} \quad (20)$$

where $\overline{H_B.M_{B[22:0]}}$ represents the inverse code of $H_B.M_{B[22:0]}$, Same to FP addition, the approximation will introduce another noise of $U(\mu, \sigma^2)$. Therefore, we have

$$CN_{(sub,\alpha,\beta)} = (\alpha + \frac{\beta}{2^p} + 1) * U(\mu, \sigma^2) \quad (21)$$

When 1-bit left shift is needed,

$$CN_{(sub,\alpha,\beta)} = (2\alpha + \frac{\beta}{2^{p-1}} + 1) * U(\mu, \sigma^2) \quad (22)$$

2) $E_A = E_B$
For this case, the operation can be written as

$$\begin{aligned} &PI_{[1:0]}.PF_{[w:0]} \\ &= H_A.M_{A[22:0]} + \alpha * U(\mu, \sigma^2) \\ &\quad + (\overline{H_B.M_{B[22:0]}} + 1 + \beta * U(\mu, \sigma^2)) \end{aligned} \quad (23)$$

For this case, several bits of left shift is needed. We denoted it as t-bits left shift. In our implementations, the maximum value of t is restricted by the requirements of applications, $1/2^t < requirement$. After t-bits left shift,

$$CN_{(sub,\alpha,\beta)} = (2^t * (\alpha + \beta) + 1) * U(\mu, \sigma^2) \quad (24)$$

3) $E_A < E_B$
This case is similar to the case 1). Therefore,

$$CN_{(sub,\alpha,\beta)} = (\frac{\alpha}{2^p} + \beta + 1) * U(\mu, \sigma^2) \quad (25)$$

Based on the occurrence probability, the model can be further written as

$$\begin{aligned} CN_{(sub,\alpha,\beta)} &= CN_{(sub,\alpha,\beta),E_A>E_B} * P_{(E_A>E_B)} \\ &\quad + CN_{(sub,\alpha,\beta),E_A=E_B} * P_{(E_A=E_B)} \\ &\quad + CN_{(sub,\alpha,\beta),E_A<E_B} * P_{(E_A<E_B)} \\ &\approx \frac{\alpha + \beta}{2} + 1 \end{aligned} \quad (26)$$

Same to FP addition, the model of FP subtraction is exponent-independent as well, as shown in Fig. 4.

D. NOISE PROPAGATION MODEL OF APPLICATIONS AND DESIGN FRAMEWORK

As mentioned above, we presented three NPMs for FP multiplication, FP addition, and FP subtraction. Based on these NPMs, the ANPM could be obtained. According to the central limit theorem [44], the final noise would converge to a normal distribution. To characterize the error of an application, the critical path for an application was introduced. The critical path for an application was defined as the path to get the largest noise on the final result. Thus, we could achieve the final error distance distribution of the final result by

$$\begin{aligned} &\alpha_0 = 1, \\ &\beta_0 = 1 \\ &\ldots \\ &\alpha_i = CN_{(ope_i,\alpha_{Left,i-1},\beta_{Left,i-1})}, \\ &\beta_i = CN_{(ope_i,\alpha_{Right,i-1},\beta_{Right,i-1})} \\ &\ldots \\ &CN_{(app)} = CN_{(ope_n,\alpha_{n-1},\beta_{n-1})} \end{aligned} \quad (27)$$

where n is the number of operations on the critical path. *Left* and *Right* just are used to differentiate the operations of the former stage. From the derived process of (27), it can be obtained that $CN_{(app)}$ is only dependent of the operation-level topology. Only the operation-level topology is given, $CN_{(app)}$ is fixed. Thus, we could get the final noise distribution.

$$N(0, CN_{(app)} * \sigma^2) = N(0, CN_{(app)}/(3 * 2^{2w})) \quad (28)$$

where $N(0, CN_{(app)} * \sigma^2)$ is a normal distribution. It can be used to estimate the quality of the final results. Without loss of generality, we defined our metrics by normalizing three times of the standard variance of the final error distance distribution to the minimum mantissa value of the final results of the corresponding application. Herein, we take $1.0\cdots0$ as the minimum mantissa value of the final results of the corresponding application. Thus, we have

$$3 * sqrt(CN_{(app)}/(3 * 2^{2w})) <= requirement \quad (29)$$

Since $CN_{(app)}$ only depends on operation-level topology, once the topology is fixed, the requirement can determine the needed efficient bit width directly. The confidence interval of the model is 99.7% in theory, but there may be some difference in actual applications.

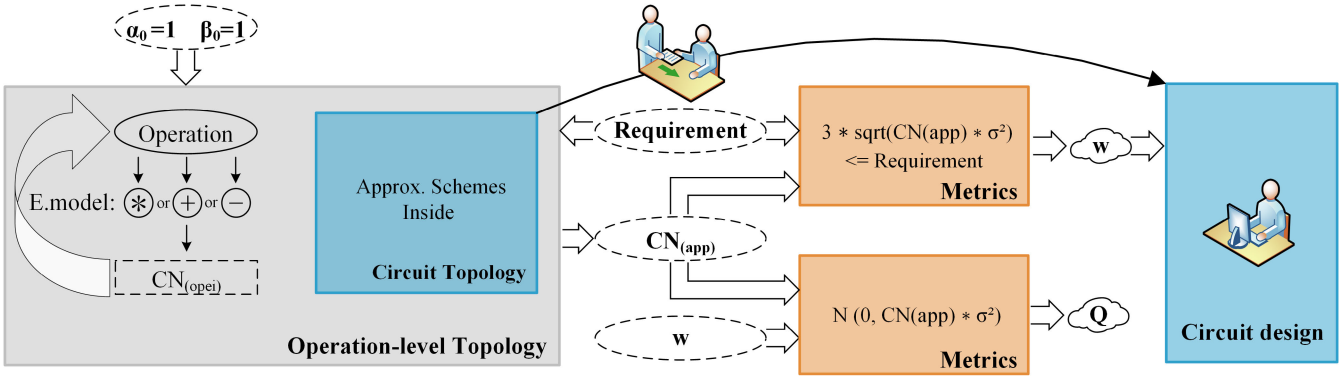

FIGURE 5. The design framework. The left part of the figure is mainly an operating-level topology which obtains the application noise propagation models. The upper orange square is used to generate the efficient bit width that meet the specific requirement and then to guide circuit designers, and the lower one provides a measure to evaluate the quality of the data.

After preparing the above NPMs, ANPM, and metrics, the design framework can be obtained. The circuit topology is the organization of the node hardware executing units, such as approximate FP adder and FP multiplier. The proposed approximate schemes are included in each node hardware executing unit. Operation-level topology is how to realize all needed operations, using the corresponding hardware based on a given circuit topology. The corresponding $CN_{(app)}$ can be obtained through (27). Combining the requirement and the $CN_{(app)}$ with the metrics defined in (29), the corresponding efficient bit width of mantissa, w, can be achieved. Thus, three key factors(approximation schemes, circuit topology and the efficient bit width of mantissa) of circuit design have been obtained, as shown in Fig. 5. Once $CN_{(app)}$ is obtained, combined with the efficient bit width of mantissa, the noise distribution will be easily got. Further, the quality of data can be estimated through (29) or other metrics, Q in Fig. 5.

It is important to note that our framework was obtained based on PAM, and our purpose is to use it for truncation. The accuracy of both PAM and truncation under different efficient bit width could be expressed by (3). When the efficient bit width is the same, the accuracy of truncation would be higher than PAM, which means that the quality of the data generated by truncation is better than that predicted by the framework and the efficient bit width required for truncation is less than that the predicted results. Thus, our proposed framework is equally applicable to truncation. Many approximate calculation methods have derived from truncation which is representative. Thus, our following work would focus on analyzing the relationship between the design framework and truncation.

A commonly used field of approximate computing is image processing, and we would use it to explain the design framework proposed above detailedly. Approximate computing is inseparable from accuracy. When the user gives a specific requirement, it is necessary to consider the operations needed by the application and how to achieve the requirements. And then the efficient bit width should be determined by requirements, which is very troublesome, especially for floating point operations. In this paper, a general framework was proposed to address this problem. The designer can get the noises of the application according to the operation-level topology structure by (27), which is $CN_{(app)}$. And then based on the metrics shown in the upper orange box of Fig. 5, the efficient bit width, w, can be obtained. Thus, the proposed framework provides designers a method for approximate computing by combing applications with specific accuracy requirements. In addition, combined $CN_{(app)}$ with w, as shown in the other orange box of Fig. 5, the noise distribution of the application can be evaluated, Q.

IV. EXPERIMENTAL RESULTS

In section III, we had studied noises generated by the approximate adder, subtractor, and multiplier with the probability method. Finally, NPMs and ANPM were established to achieve the framework. Through the framework, we can quickly evaluate the quality of data generated by applications. More importantly, the efficient bit widths of the mantissa under the specific requirements can be obtained. In this section, we experimented with two typical applications of the image, edge detection and Gaussian filtering, to verify the proposed framework. The data with a zero-mean error is defined as general data. Peak signal-to-noise ratio (PSNR) would be used in this paper to evaluate the image quality.

A. EDGE DETECTION

Like the general image edge detection experiment, we used operation units designed in section III to implement approximate edge detection computing. The process of the experiments was the same as Fig. 5. After the multiplier, adder, and subtractor with configurable efficient bit width were designed, the hardware circuit of the edge detection experiment could be obtained. By analyzing the operation-level topology of edge detection algorithm and combining it with (27), the $CN_{(app)}$ could be obtained. According to (28), the quality of the data generated by the application could be evaluated. The specific efficient bit width could be obtained by combining users' requirements with (29).

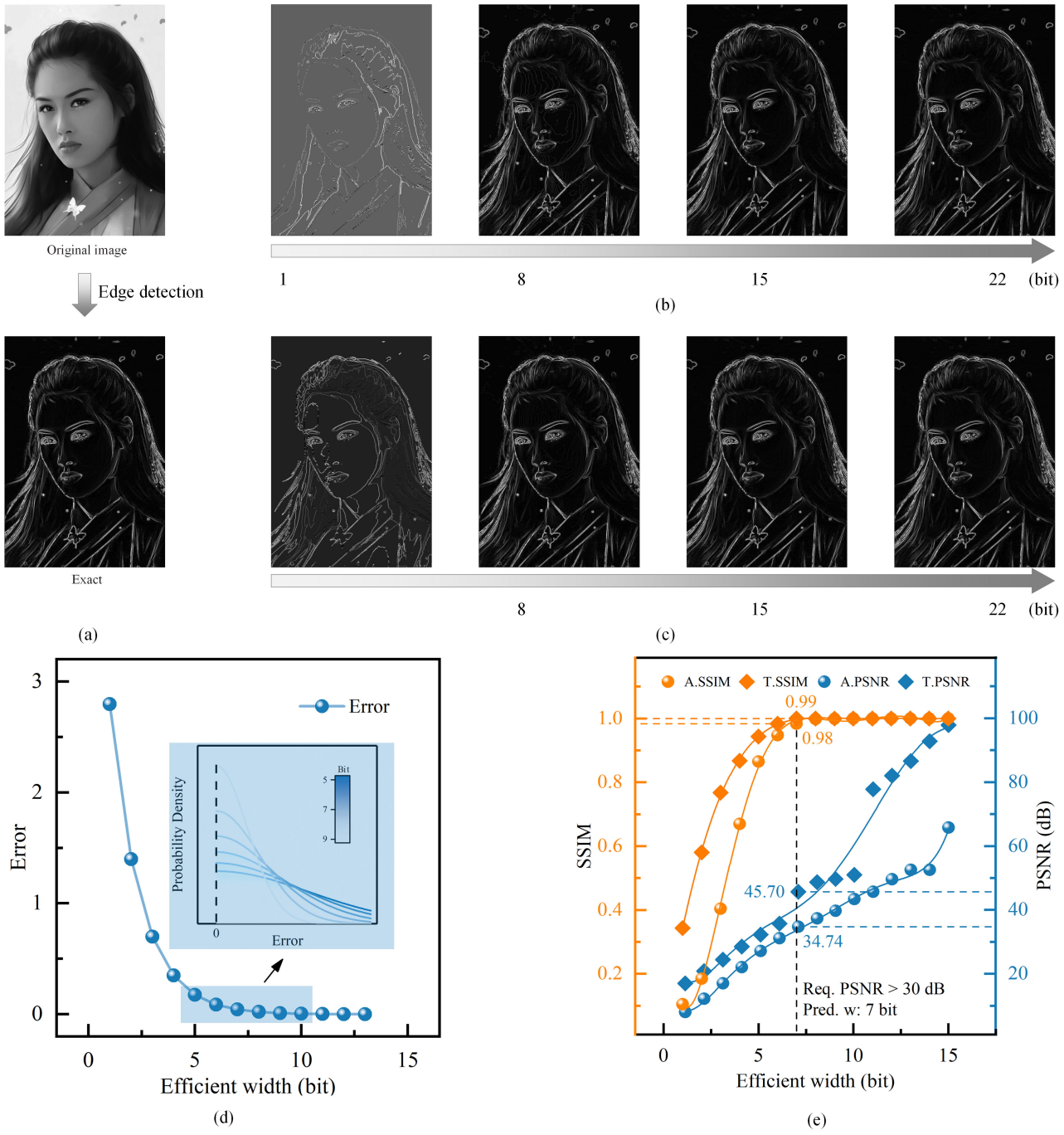

FIGURE 6. Results of edge detection experiments. (a) Accurate result of image edge detection. The result of edge detection after approximate computing of different efficient bit width by PAM (b) and truncation (c). (d) Noise caused by application under different efficient bit width, blue sphere. Each sphere represents the noise distribution. The blue shading in the upper right corner magnifies the error distribution under part of the efficient bit width. (e) SSIM and PSNR corresponding to PAM and truncation. A. and T. are the abbreviation for PAM and truncation respectively.

Following the steps above, three different sets of edge detection experiments for one specific image were carried out: accurate calculation, truncation, and the PAM. Firstly, the experiment was performed with accurate units, and the experimental result was used as a standard reference for the other two inaccurate consequences. The accurate experimental result was shown in Fig. 6 (a). Then the others two approximate computing experiments, PAM and truncation, were performed, and the corresponding results were obtained when the efficient bit width changed from 1 bit to 22 bits. In the Fig. 6 (b) and (c), the images above the gray gradient arrow are the experimental results corresponding to the different efficient bit width. The following information can be obtained from these two sets of diagrams:

1) With the increase of efficient bit width, the difference between two approximate results and the standard reference could no longer be distinguished by human eyes. This phenomenon appeared in many previous literature pieces, which also explains why some applications suitable for approximate calculation.
2) When the efficient bit width is the same, PAM would produce larger noise than truncation. It is consistent with (3). In actual applications, each of these two approaches has its advantages due to different input data. For example, in an extreme case, when all the mantissas are $23'b111\ldots111$, the noise induced by PAM would be much smaller than truncation. But the more important significance of PAM is that it provides convenience for us to build noise propagation models.

By analyzing the operation-level topology of the application and combining (28), the error distributions generated by the application consisting of approximate units were obtained. The value of each distribution at three times the standard deviation was plotted as the blue sphere in Fig. 6 (d). As we expected, it shows that the higher bit of the mantissa is, the more significant impact will be.

Structural similarity (SSIM) and PSNR are often used to evaluate image quality. Fig. 6 (e) shows the SSIM and PSNR obtained after processing the current image with two approximate methods under different efficient bit widths. PAM and truncation are represented sphere and square, respectively, and they are abbreviated as A. and T.. For example, A.SSIM represents the SSIM obtained by PAM processing. The following conclusions could be drawn from Fig. 6 (e):

1) The higher the values of SSIM and PSNR, the more accurate the result is. When the efficient bit width is the same, all the squares are above the corresponding color spheres, which means that the noise induced by PAM is larger than that induced by truncation. And it is similar to the results reflected by the Fig. 6 (b) and Fig.6 (c). The reason for the phenomenon is that most of the mantissa bits of the current image data are 0. PAM sets the lowest bit of the valid bits to 1 is equivalent to induced other noise into the current data. And thus the noise induced by truncation would be smaller for the current image. For different data, two methods may have their own advantages. For example, when the mantissas are 23'b111…111, the noise induced by PAM will be smaller than that induced by truncation.
2) With the same efficient bit width, PSNR is only about 40 dB when SSIM approaches the limit. When the efficient bit width is 7, the SSIM is 0.99 and 0.98, respectively. At this time, the corresponding PSNR is only about 40 dB, which means a PSNR of about 40 dB may become a common requirement of the image quality. Generally, if the PSNR of two images exceeds 37dB, the difference between them may not be distinguished by human eyes [45]. Therefore, the range of PSNR studied in this paper focused on 20-50 dB.

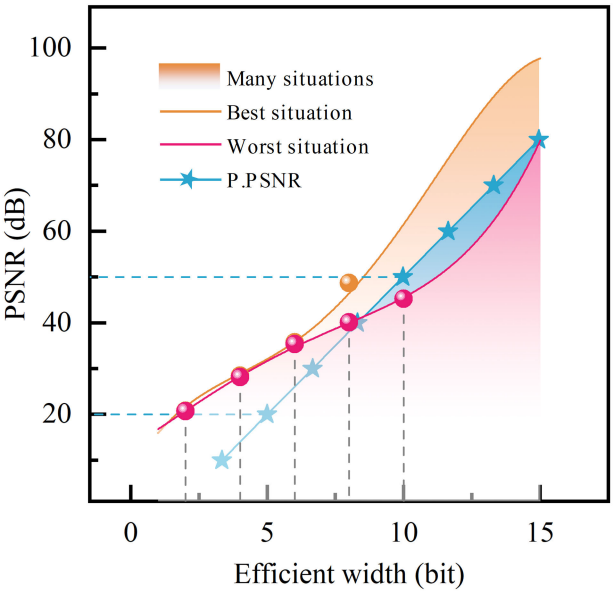

FIGURE 7. The corresponding PSNR of the image after truncation in two extreme situations. The orange and pink curves represent the best and worst situations, respectively. The blue curve with stars indicates the predicted result, represented by P.PSNR. The orange shading represents the range that the framework can successfully predict. The blue shading represents the area that the framework may not cover. 20-50 dB is the focus range of our research.

3) When the efficient bit width used by PAM is one bit more than truncation, PAM is not necessarily more accurate than truncation. For example, when the efficient bit width of truncation is 7, the experimental result is 45.70 dB. However, all of the experimental results are lower than 45.70 dB when the efficient bit widths of PAM are 8, 9, or even 10. This seems to be different from (3). It should be noted that (3) is suitable for the general situation and it might be slight deviations from the specific data. The noise induced by truncation to the current image is smaller, and thus a smaller efficient bit width can achieve a larger PSNR.
4) It is reasonable that the efficient bit width predicted by the framework will have a difference of 1-2 bits compared with the actual need. When a PSNR requirement of 30-40 dB was required, the result of our framework prediction was 7 bits. However, for the current image, only 5 bits were needed for truncation and 6 bits for PAM. The prediction was successful with a difference of 1-2 bits. The proposed framework is independent of the input data, and it is a uniform framework suitable for all data. Therefore, to cover all the data and situations, it is reasonable to have a difference of 1-2 bits for specific data. The current image is processed with truncation will produce relatively small noise. Therefore, fewer bits are required than general data. Similarly, after the image was processed by the PAM, it would produce larger noise than truncation. Thus, PAM would take more bits to achieve the same PSNR than truncation.

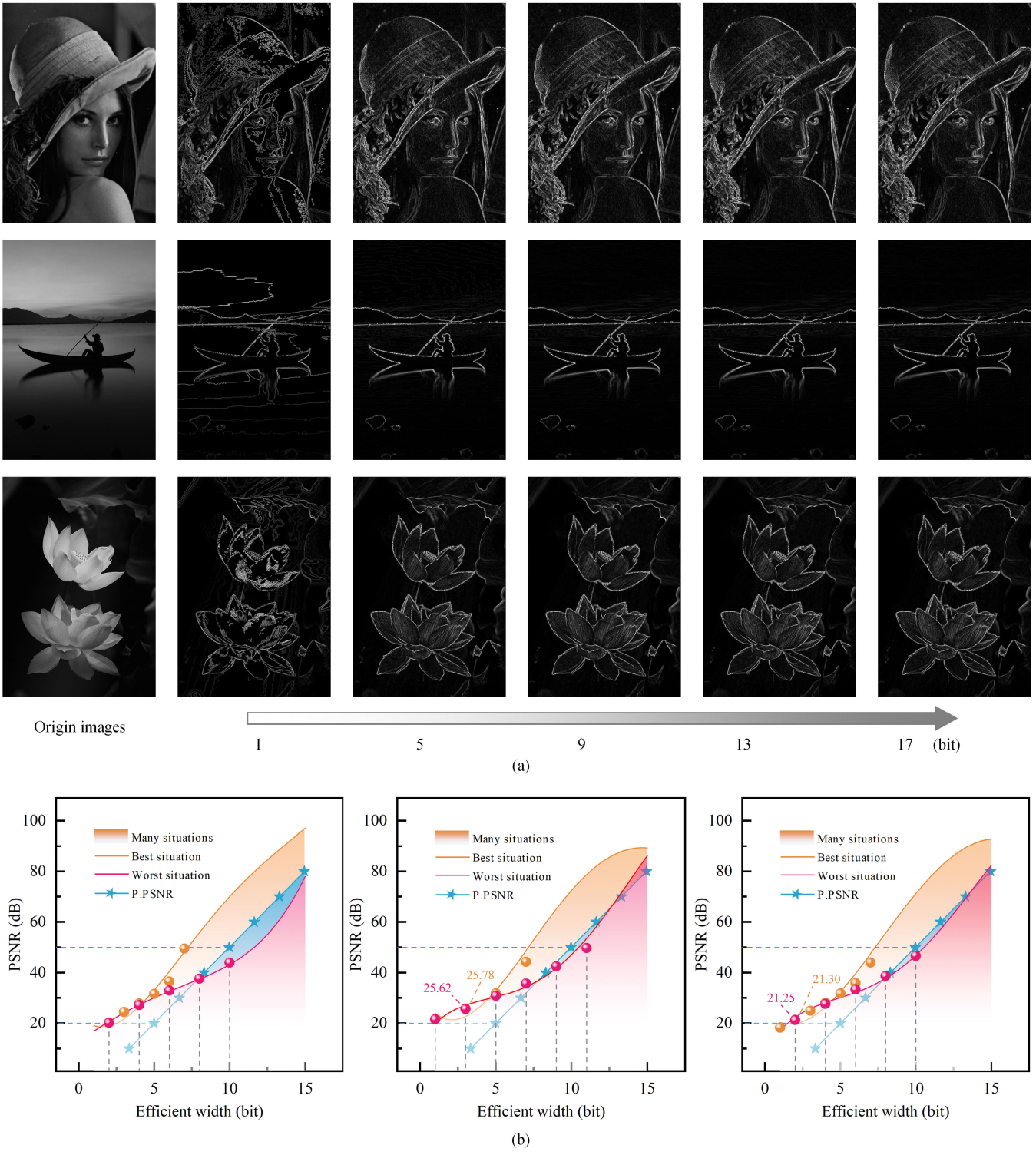

FIGURE 8. The corresponding PSNR of three classes of images after truncation in two extreme situations. (a)Three groups of images were produced after edge detection under different efficient bit widths. Three data graphs from left to right in (b) correspond to three experiments from top to bottom in (a).

In the above work, only one image was analyzed. The feature of this image is that the noise generated by truncation is relatively small. The results show that our framework could successfully predict the efficient bit width.

Our proposed framework is unified for all data. To verify whether the framework applies to other data, we need to verify the worst situation and the best situation. The worst situation is the image that would produce the largest noise

after truncation. In other words, all mantissas of pixels in the image are $23'b111\cdots11$. In the best situation, all mantissas are 0, which means each mantissa is $23'b000\cdots00$. Each efficient bit width could achieve the higher requirements in the best situation. In this paper, we took the previous experimental results as the best situation. And then, we prepared the worst image for a similar experiment according to the previous operation-level topology. Fig. 7 shows the experimental results. In the figure, the best situation and the worst situation were represented by the orange curve and the pink curve. These two curves represent two extreme situations, and thus all data were included in the area between them (orange and blue shading). In other words, the orange and blue shading contains all the innumerable images with the same exponent and arbitrary mantissa. Therefore, Fig. 7 represents results of a class of images, and their exponents are the same as the image in Fig. 6, and the mantissa is any number from $23'b000\ldots000$ to $23'b111\ldots111$. If the predicted PSNR (blue curve) is lower than the actual PSNR (orange or pink curve), it can be considered as a successful prediction. The blue shading represents some extreme situations in which a small part of these images could not be successfully covered. In order to further verify that our framework is independent of the data, we conducted edge detection experiments on the other three classes of images, and the experimental results are shown in Fig. 8. The following points can be obtained from the figure:

1) The blue shadings in Fig. 7 and Fig. 8 (b) are very small when the range of PSNR is 20-50 dB, which mean that most of the results predicted by the framework are proper.
2) For some part of blue shadings, the prediction results of the framework are 1 bit less than the actual needed. The first reason is that the proposed framework is derived based on the occurrence probability and data-independent, while the real image is specific. Secondly, the used metric normalizes three times of the standard variance of the final error distance distribution to the minimum mantissa value of the final results of the corresponding application. Thus it is acceptable considering the models are based on the occurrence probability. Actually, these extreme cases fluctuate in a small range around the predicted curve.
3) Although these four class of input data are different, it could be seen from Fig. 7 and Fig. 8 (b) that their respective trends are roughly the same. It is shows that the framework is independent of data and only related to the application of operation-level topology.

B. GAUSS FLITTER

It could be concluded that our framework can be successfully used for edge detection. The previous experiments were only carried out on an application, although it applies to various types of data. We also need to test the performance of the framework in other applications. In this section, we used

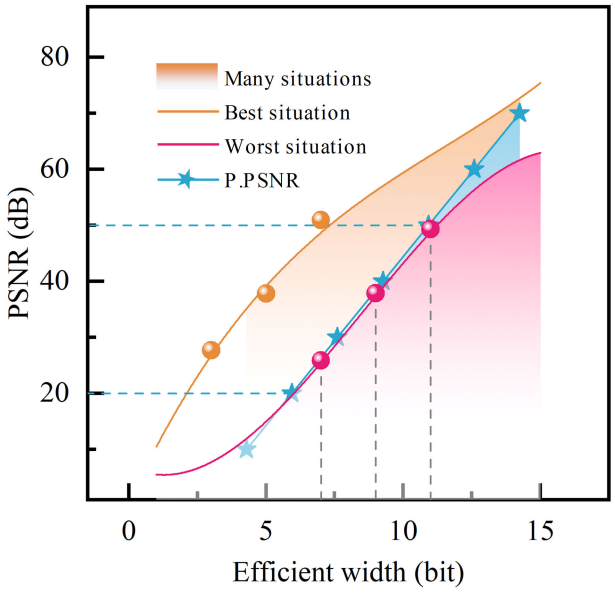

FIGURE 9. The corresponding PSNR of the image after being truncated in the Gaussian filtering experiment.

approximate units to implement the Gaussian filtering algorithm. It should be noted that the Gaussian filtering algorithm in this paper is not the general one, and it adds extra steps based on ordinary circuits. This is to verify whether our framework can be successfully applied to more complex applications.

Similar to the previous experimental process, we obtained these two extreme results of data processing with truncation under the new circuit topology and also used PSNR as the quality evaluation indicator. Finally, we compared the efficient bit width of different PSNR in two extreme situations with those predicted by the framework, as shown in Fig. 9. It can be seen from the figure that some parts of the blue curve were slightly higher than the pink curve, which means that there may be some differences in the prediction results. But the differences may not exist in reality. For example, the predicted efficient bit width was 10.9 bits, and the actual required was 11 bits, but both of them were 11 bits actually. Therefore, in extreme situations, the framework prediction experiments were utterly consistent with the actual when the range of PSNR is 20-50 dB. The following two conclusions could be drawn from the figure:

1) The results of the framework prediction are proper in most situations, and a few extreme cases with some differences. This is similar to the experimental results of edge detection.
2) The framework is also applicable to complex applications.

Combining the above two experiments, it can be concluded that the proposed framework could be successfully used to evaluate data quality, and more importantly, can successfully predict the efficient bit width needed by applications under specific requirement. It is a uniform framework that does

not depend on data, but only related to the operation-level topology of the application.

V. CONCLUSION

An approximate computing method of floating-point, similar to truncation, called PAM, was proposed in this paper. We introduced a noise distribution with zero-mean into the method, through which noise propagation models for multiplier, adder, and subtracter were established. Under such conditions, a new metric $3 * sqrt(CN_{(app)}/(3 * 2^{2w}))$ was proposed to put forward a design framework, which can be applied to truncation. Edge detection and Gaussian filtering experiments were conducted to test the results when the efficient bit width stepping from one to fifteen bits. We focused on the analysis of the experimental results when the range of PSNR is 20-50 dB. Edge detection experiments show that the framework can successfully used for most images. In the extreme situation, there might be a difference of 1-2 bits. Considering that the corresponding models were derived based on the occurrence probability, it is rational and can be accepted. As for Gaussian filtering experiments, the predicted results of the framework remained entirely proper for all situations.

The proposed framework in this paper can be used for transprecision design [46]. Configurable transprecision units are designed and employed for CPU. During the program compilation phase, the compiler can be used to define the used efficient bit width based on the defined metrics in this paper. Further, the corresponding instructions are employed to set the efficient bit width. Thus, an energy-efficient design can be built for specific approximate computing applications.

ACKNOWLEDGMENT

The authors would like to appreciate professor L. Benini and F. K. Gurkaynak from ETH, Zurich for their valuable advice and discussion.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18TH IEEE Eur. TEST Symp. (ETS)*, Avignon, France, May 2013, pp. 1–6.

[2] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, p. 62, Mar. 2016.

[3] M. K. Ayub, O. Hasan, and M. Shafique, "Statistical error analysis for low power approximate adders," in *Proc. 54th Annu. Design Autom. Conf.*, Austin, TX, USA, Jun. 2017, pp. 1–6.

[4] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proc. Int. Symp. Low Power Electron. Design*, San Diego, CA, USA, Aug. 1999, pp. 30–35.

[5] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: A voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, San Fancisco, CA, USA, Aug. 2009, pp. 195–200.

[6] L. N. B. Chakrapani, K. K. Muntimadugu, and K. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, Atlanta, GA, USA, 2008, pp. 187–196.

[7] M. Shafique, L. Bauer, and J. Henkel, "EnBudget: A run-time adaptive predictive energy-budgeting scheme for energy-aware motion estimation in H.264/MPEG-4 AVC video encoder," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2010, pp. 1725–1730.

[8] Y. V. Ivanov and C. J. Bleakley, "Real-time H.264 video encoding in software with fast mode decision and dynamic complexity control," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 6, no. 1, pp. 1–21, Feb. 2010.

[9] B. J. Phillips, D. R. Kelly, and B. W. Ng, "Estimating adders for a low density parity check decoder," *Proc. SPIE*, vol. 6313, Aug. 2006, Art. no. 631302.

[10] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2010, pp. 957–960.

[11] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[12] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[13] A. A. Gaffar, W. Luk, P. Y. Cheung, N. Shirazi, and J. Hwang, "Automating customisation of floating-point designs," in *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream* (Lecture Notes in Computer Science). 2002, pp. 523–533.

[14] V. Patil, A. Raveendran, P. M. Sobha, A. David Selvakumar, and D. Vivian, "Out of order floating point coprocessor for RISC v ISA," in *Proc. 19th Int. Symp. VLSI Design Test*, Ahmedabad, India, Jun. 2015, pp. 1–7.

[15] L. Li, M. Gautschi, and L. Benini, "Approximate DIV and SQRT instructions for the RISC-V ISA: An efficiency vs. Accuracy analysis," in *Proc. 27th Int. Symp. Power Timing Modeling, Optim. Simulation (PATMOS)*, Thessaloniki, Greece, Sep. 2017, pp. 1–8.

[16] W. Liu, L. Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and analysis of inexact floating-point adders," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 308–314, Jan. 2016.

[17] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," in *Proc. ESSCIRC Conf.: 42nd Eur. Solid-State Circuits Conf.*, Lausanne, Switzerland, Sep. 2016, pp. 465–468.

[18] P. Yin, C. Wang, W. Liu, and F. Lombardi, "Design and performance evaluation of approximate floating-point multipliers," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Pittsburgh, PA, USA, Jul. 2016, pp. 296–301.

[19] M. Imani, D. Peroni, and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," in *Proc. 54th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Austin, TX, USA, Jun. 2017, pp. 1–6.

[20] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 3, pp. 273–286, Jun. 2000.

[21] A. Gupta, S. Mandavalli, V. J. Mooney, K.-V. Ling, A. Basu, H. Johan, and B. Tandianus, "Low power probabilistic floating point multiplier design," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Chennai, India, Jul. 2011, pp. 182–187.

[22] J. Eilert, A. Ehliar, and D. Liu, "Using low precision floating point numbers to reduce memory cost for MP3 decoding," in *Proc. IEEE 6th Workshop Multimedia Signal Process.*, Siena, Italy, Sep./Oct. 2004, pp. 119–122.

[23] F. Fang, T. Chen, and R. A. Rutenbar, "Floating-point bit-width optimization for low-power signal processing applications," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Orlando, FL, USA, May 2002, pp. III–3208.

[24] M. Yan, Y. Song, Y. Feng, G. Pasandi, M. Pedram, and S. Nazarian, "KNN-CAM: A k-Nearest neighbors-based configurable approximate floating point multiplier," in *Proc. 20th Int. Symp. Qual. Electron. Design (ISQED)*, Santa Clara, CA, USA, Mar. 2019, pp. 1–7.

[25] H. Zhang, W. Zhang, and J. Lach, "A low-power accuracy-configurable floating point multiplier," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Seoul, South Korea, Oct. 2014, pp. 48–54.

[26] M. S. Kim, A. A. D. Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell's approximate log multipliers for convolutional neural networks," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 660–675, May 2019, doi: 10.1109/TC.2018.2880742.

[27] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Approximate DCT image compression using inexact computing," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 149–159, Feb. 2018, doi: 10.1109/TC.2017.2731770.

[28] D. Peroni, M. Imani, and T. S. Rosing, "Runtime efficiency-accuracy tradeoff using configurable floating point multiplier," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 2, pp. 346–358, Feb. 2020, doi: 10.1109/TCAD.2018.2885317.
[29] R. Omidi and S. Sharifzadeh, "Design of low power approximate floating-point adders," *Int. J. Circuit Theory Appl.*, vol. 49, no. 1, pp. 185–195, 2020.
[30] 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, Sep. 2013.
[31] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, 2014, p. 361.
[32] C. Li, W. Luo, S. S. Sapatnekar, and J. Hu, "Joint precision optimization and high level synthesis for approximate computing," in *Proc. 52nd Annu. Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2015, pp. 1–6.
[33] D. Sengupta, F. S. Snigdha, J. Hu, and S. S. Sapatnekar, "SABER: Selection of approximate bits for the design of error tolerant circuits," in *Proc. 54th Annu. Design Automat. Conf.*, Austin, TX, USA, 2017, p. 72.
[34] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Vancouver, BC, Canada, Dec. 2012, pp. 449–460.
[35] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[36] S.-R. Kuang, J.-P. Wang, and C.-Y. Guo, "Modified booth multipliers with a regular partial product array," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 5, pp. 404–408, May 2009.
[37] W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 692–701, Jul. 2000.
[38] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate Radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.
[39] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, 2014, pp. 1–4.
[40] H. Zhang, M. Putic, and J. Lach, "Low power GPGPU computation with imprecise hardware," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2014, pp. 1–6.
[41] M. Gautschi, M. Schaffner, F. K. Gurkaynak, and L. Benini, "4.6 a 65nm CMOS 6.4-to-29.2pJ/FLOP@0.8 V shared logarithmic floating point unit for acceleration of nonlinear function kernels in a tightly coupled processor cluster," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Jan. 2016, pp. 82–83.
[42] M. Schaffner, M. Gautschi, F. K. Gürkaynak, and L. Benini, "Accuracy and performance trade-offs of logarithmic number units in multi-core clusters," in *Proc. IEEE 23rd Symp. Comput. Arithmetic (ARITH)*, Santa Clara, CA, USA, Jul. 2016, pp. 95–103.
[43] M. Gautschi, M. Schaffner, F. K. Gurkaynak, and L. Benini, "An extended shared logarithmic unit for nonlinear function kernel acceleration in a 65-nm CMOS multicore cluster," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 98–112, Jan. 2017.
[44] A. Lyon, "Why are normal distributions normal?" *Brit. J. Philosophy Sci.*, vol. 65, no. 3, pp. 621–649, Sep. 2014.
[45] H. Makkar and O. S. Lamba, "An improved VBM3D filtering technique for removal noise in video signals," *Eur. J. Adv. Eng. Technol.*, vol. 4, no. 8, pp. 584–591, 2017.
[46] *OPRECOMP*. [Online]. Available: http://oprecomp.eu

YIYAO XIANG received the B.S. degree from Chengdu University, Chengdu, China, in 2017. He is currently pursuing the M.S. degree with the University of Electronic Science and Technology of China(UESTC), Chengdu, China.

His current research interests include hardware circuit design, machine learning, hardware security, and approximate computing.

LEI LI received the Ph.D. degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2010.

He has worked at D-ITET, ETH Zurich, from 2016 to 2017, as a Senior Scientist. He is currently an Associate Professor with the University of Electronic Science and Technology of China. His current research interests include integrated circuits, machine learning, and hardware security.

SHIWEI YUAN received the B.S. degree from the University of South China, Hengyang, China, in 2019. He is currently pursuing the M.S. degree with the University of Electronic Science and Technology of China (UESTC), Chengdu, China.

His current research interests include hardware circuit design, machine learning, hardware security, and approximate computing.

WANTING ZHOU received the M.Eng. degree and the Ph.D. degree in communication from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2008 and 2014, respectively.

Since 2008, she has been with the UESTC. Her current research interests include SoC design, and hardware trojan detection.

BENQING GUO (Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2005 and 2011, respectively.

He had multiple years of industry experience in the development of hardware systems and integrated circuits. Since 2014, he conducted research work of integrated circuits in the UESTC. Then he worked at the Microelectronics Group, Pavia University, Italy, from 2017 to 2018, as a Visiting Scholar. He joined the Chengdu University of Information Technology in 2020 and is currently a Researcher. His research interests include integrated circuits and techniques.