Securing Approximate Computing Systems via Obfuscating Approximate-Precise Boundary

Pruthvy Yellu, *Student Member, IEEE,* and Qiaoyan Yu, *Senior Member, IEEE*

***Abstract*—Approximate computing (AC) techniques have been leveraged to improve computing performance and energy efficiency with minor degradation in accuracy. Recent literature indicates that some AC mechanisms could be exploited by attackers to implement new attack surfaces. To address the emerging attacks in AC systems, we propose to obfuscate the approximate-precise boundary (APB) with entry-blurring and boundary-broadening schemes. The proposed entry-blurring scheme leverages a hidden quality metric, which has a strong correlation with approximation errors, to obscure the entrance of APB and eliminate the explicit transition between approximate and precise modes, thus improving AC systems' resilience against APB attacks. The proposed boundary-broadening scheme enlarges the transition zone between approximate and precise modes by expanding a single APB threshold to two comparison thresholds, and it further enables a random selection of approximate computing modules in the candidate library. The protection mechanisms provided by our obfuscation method strengthens AC systems' resilience against APB attacks. Our case studies show that the proposed entry-blurring scheme improves the application quality by up to 168% over the baseline and successfully achieves the desired accuracy. The latency overhead of our method is negligible and the increase on area and power cost can be minimized to 6% and 8%, respectively.**

I. Introduction

Approximate computing (AC) techniques trade precision with better processing speed and energy efficiency, benefiting the applications such as signal processing, data mining, and machine learning [1]. In the era of big data, advanced approximation mechanisms will play an increasingly important role. However, as reported in recent articles [2]–[5], attackers could leverage various approximation mechanisms to create new attacks and then compromise AC systems. For example, approximate computation and storage modules could be manipulated to create a covert channel, through which critical information is leaked [2], [4], [6]. Attackers can also tamper with the built-in error resilience mechanisms to accumulate more errors than what the system can tolerate [7], [8]. If the errors induced by approximation are collected and propagated throughout the entire system maliciously, an attack in the AC module could lead to a catastrophic effect in the entire computing system [2], [4], [9]. Consequently, it is imperative to address the new security threats brought by the utilization of approximation techniques.

P. Yellu, and Q. Yu are with the Department of Electrical and Computer Engineering, University of New Hampshire, Durham, NH, 03824 USA. e-mail: qiaoyan.yu@unh.edu.

This work is partially supported by National Science Foundation Awards CNS-1652474 and CNS-2022279.

Manuscript received on December 8, 2021, revised on Feb. 20, 2022.

To facilitate the development of countermeasures against various attacks in AC systems, different models are proposed to characterize those attacks. The work [10] highlights the security threats from a malicious modification on the control logic of approximate memory. The attacks in approximate arithmetic modules are demonstrated in [7]. The survey [2] introduces the attacks that build covert channels, compensate approximation errors, terminate error resilience mechanisms, and propagate errors.

Although the existing efforts [2], [7], [10] have comprehensively demonstrated the key features of the attacks conducted in AC systems, those works mainly focus on individual approximate arithmetic/memory modules. There still lacks the investigation that examines the interaction between approximate and precise modules from a system point of view. To fill this gap, this work studies the security threats from the unprotected boundary between approximate and precise modules, explores the feasible defense methods to strengthen the boundary and thus secure AC systems, and assesses the efficiency of proposed countermeasures in practical applications. More specifically, we make the following contributions in this work:

- We propose the concept of approximate-precise boundary (APB) and demonstrate three kinds of APBs that can be commonly observed in AC systems. Practical applications are provided to demonstrate the pressing need of protecting AC systems from APB attacks.
- While existing countermeasures protect the approximate modules, we propose high-level obfuscation mechanisms to secure the transition zone between precise and approximate operations. Our countermeasure is composed of entry-blurring and boundary-broadening schemes, which harden APBs and mitigate the attack that breaches the AC system via APBs.
- The proposed countermeasures are evaluated in two case studies —approximate image edge detection and approximate memory. Our experimental results confirm that our method can effectively mitigate APB attacks and meet the desired quality requirement.

The rest of this work is organized as follows. Section II summarizes the related work. Section III introduces the attack model interested in this work. In Section IV, we define APBs and provide an example for each APB. In Section V, we propose two obfuscation schemes to harden APBs. Experimental results are available in Section VI. We conclude this work in Section VII.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

II. Related Work

A. Approximate Computing Techniques

The main principle of approximate computing is trading accuracy with performance and power consumption. The approximate computing techniques can be applied at various levels: circuit, microarchitecture, algorithm, compiler, system, and application [11]–[13].

At circuit level, approximation techniques reduce the computation precision or shorten the critical delay path to improve performance and energy efficiency [14], [15]. Approximate techniques deployed in memory include reducing the refresh interval [16], [17], overscaling the supply voltage [18], and narrowing the guard band [19]. To save power consumption of memory, researchers [20] also propose to discard the least significant bits (LSB) and store error correcting codes in the LSB positions. In an approximate computing system, there exists both approximate and precise instructions [21]. To execute both types of instructions, the instruction set architecture (ISA) and compiler need to be changed accordingly [22], [23]. For instance, certain bits are added to the ISA to facilitate the differentiation of the instruction type and guide the precise or approximate execution [22]. At algorithm level, approximate computation can be realized by the techniques such as reducing the loop, replacing the repeatedly executed arithmetic functions with look-up tables (LUTs) [24], and skipping the execution of the functions in a predicted part of the code [25], [26]. The approximation techniques employed at software level mainly aim at reducing computational power [27].

Approximate computing has been widely applied to image processing, where the application can tolerate errors. Many Tools are developed to automate the process of selecting an appropriate approximate level for the desired quality requirement. The tool in the work [28] developed an algorithm to choose the best approximate computing intellectual property (IP) for image edge detection.

B. Attacks in Approximate Computing Systems

In spite of facilitating to improve performance and energy efficiency, approximation techniques could be manipulated to conduct new attacks. Recent studies [2]–[6], [29], [30] report that approximation mechanisms can be exploited to perform stealthy attacks, which suddenly cause significant degradation in accuracy and lead to unpredictable outputs. The works [4], [7] predict that a misused approximation mechanism could exaggerate the error propagation or leak some critical information. The work [10] projects that attackers could (1) modify the original memory allocation map for precise and approximate memory blocks and then transfer the confidential data to the approximate memory block, (2) exploit the relaxed guard band in multi-bit memory cells to cause quantization errors, and (3) tamper with the memory controller to alter the DRAM refresh rate. The follow-up work [6] proposes some new vulnerabilities in approximate data. Two practical attacks on the AC systems for multi-layer perceptron and Sobel algorithm based image edge detection are presented in [31].

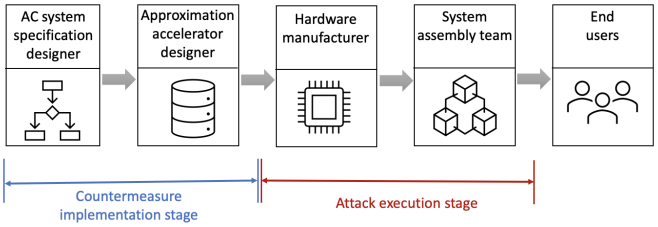


Fig. 1: Multiple entities involved in AC system design and deployment.

The works aforementioned urge us to investigate effective countermeasures against various attacks in AC systems.

C. Attack Detection and Mitigation for Approximate Computing Systems

Some researchers [3], [4], [6] exploit approximate computing mechanisms to address the hardware security issues and they also envision that there is an urgent need of managing the security threats in AC systems. The work [7] analyzes the impact of malicious interconnect and Boolean logic on approximate arithmetic units. The work [7] also presents a logic defense mechanism to thwart the attack that tampers with the input and output of approximate arithmetic units. However, that defense mechanism only works for small approximate arithmetic modules, instead of protecting the computing system from a higher level. That approach will not be able to thwart the attack that replaces the approximate IP with a malicious one.

To secure the data in AC systems, the recent work [6] introduces some information hiding mechanisms. To save the power consumption, some of the approximation techniques neglect some LSBs of the data. The positions for those LSBs are utilized to hide the critical information or implement abnormal checking mechanisms. That approach may lead to a new attack surface. As indicated in the works [2], [4], the approximate techniques can be manipulated to create a covert channel. The unused LSB positions could be leveraged to form a covert channel to leak information.

The follow-up work [31] indicates that the clear boundary between the precise and approximate modules makes approximate systems vulnerable to attacks. The framework proposed in [31] suggests obfuscating the time, location, and the mean of approximation. However, that work only uses simple examples to prove the concept.

III. Attack Model

The design and deployment of AC systems involve multiple entities as shown in Fig. 1. We assume that attacks are conducted in the stages of hardware manufacturing and system assembly and initialization. Although hardware manufacturers have access to the physical implementation, it is difficult for them to fully recover all the design details. We assume that manufacturing attackers can recognize the instances of major functional modules. The system assembly team has the knowledge of system specification and limited test cases for verification. The AC system will be a gray box to an adversary in the assembly and testing team, who could manipulate the control knobs for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE I: Definition of APB and its applications.

Category of APBs	Applications
APB-1: Positions for approximate bits	Arithmetic unit, Memory
APB-2: Data block for approximate computation	Memory, Communication network
APB-3: Mode-switching threshold for approximation	Adaptive computation, Adaptive storage

system configuration. In this work, we abstract the attack as high-level design tampering, instead of specific hardware modification. The primary attack goal is to degrade the AC system's computation accuracy. To address the attacks, we apply the proposed countermeasures to the early design stages of AC systems.

IV. New Attack Surface: Approximate-Precise Boundary (APB)

A. Definition of APB

High-performance computing systems are highly demanded in computation-intensive applications. An AC system leverages its inherent error resilience to trade reduced accuracy with higher speed and lower power consumption. Typically, a computing system will integrate precise and approximate operations together to obtain the degraded (but still acceptable) accuracy/precision. Regardless of the approximation mechanism at different abstraction levels, we can observe an explicit boundary between precise and approximate operations/logic/storage zones. The definition of APB is explicitly established in the system specification phase. Based on the existing literature [11], we categorize the boundaries between precise and approximate operations in Table I. In the following section, we introduce the specific attack that exploits the APB as a new attack surface.

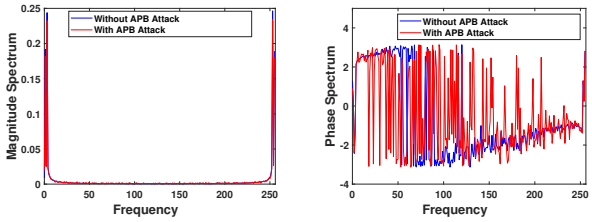
B. Attacks on APBs

1) APB-1 Attack: Altering Approximate Bit Positions

In precision-scaling based approximation techniques, users need to decide how many bits and which bits will be processed approximately. **The first type of APB refers to the dedicated bit position for approximation**. As precision scaling is one of the approximation techniques often applied in Fast Fourier Transform (FFT), we demonstrate the impact of precision loss due to APB-1 attack in a 256-point approximate FFT. In our case study, we applied approximate additions to the FFT, where several LSBs in the mantissa of the addition operands were discarded. The APB-1 attack ignored more LSBs than what were allowed in the original approximate FFT. As an example, we performed the approximate FFT on a sinusoidal signal. The APB-1 attack muted 50 mantissa bits in the addition operation at the first stage of FFT. As shown in Figs. 2(a) and (b), although the APB-1 attack does not cause a significant change on the magnitude, it results in numerous spikes in the phase spectrum.

2) APB-2 Attack: Sabotaging Approximate Data Blocks

Dynamic approximation techniques adapt the precise and approximate operations based on the nature of data.



(a) (b)

Fig. 2: Effect of the attack on APB-1: (a) magnitude and (b) phase spectra for a sinusoidal signal processed by a tampered approximate FFT.

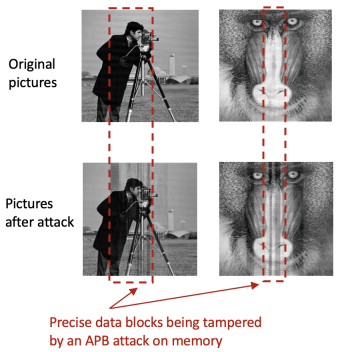


Fig. 3: Effect of the attack on APB-2: critical data blocks of images stored in a compromised approximate memory.

Critical data that carries important features will be processed by precise computation and stored in precise memory blocks; on the contrary, non-critical data will be handled by approximate arithmetic and saved in an approximate memory. **The second type of APB separates the data blocks for critical and non-critical information**. Attacks on APB-2 could happen during the data blocks being transferred over a hybrid precise/approximate storage media. We assume that two images shown on the first row in Fig. 3 are compressed and decompressed in an application of image processing, in which a hybrid precise/approximate memory is used to store the intermediate files. If the APB-2 attack alters the storage location of the critical pixels (in the highlighted zone) from precise memory to approximate memory, the feature (e.g., darkness) of those images will be changed. If image classification is further performed on those images, the classification error could increase due to the APB attack. Thus, the explicit indication of approximate-precise data boundary will become an attack target and ease the attack in AC systems.

3) APB-3 Attack: Modifying Mode Switching Threshold

An AC system tunes its configurations (e.g., precision parameters, control knobs to switch the precise/approximate operation mode) to achieve its desired accuracy. Although varying with applications, the common system-configuration options include voltage, frequency, bit-width for precision control, and the number of repetitive iterations. In an AC system, there may exist one or more thresholds to determine to what extent the approximate operations can be deployed in the system. Unlike other two APBs, **the third type of APB is a threshold** in the format of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

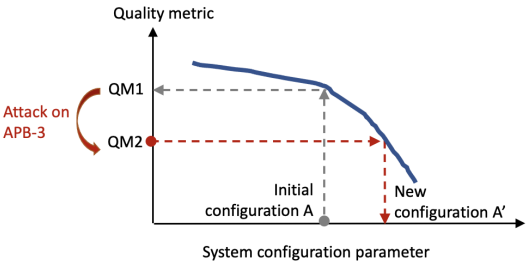
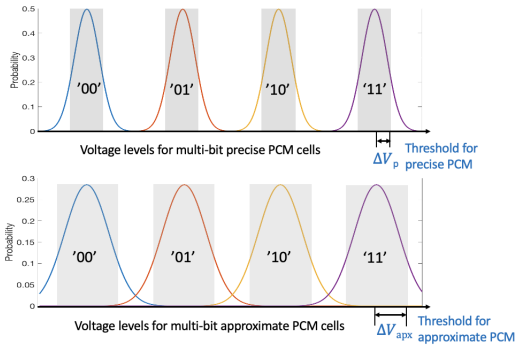
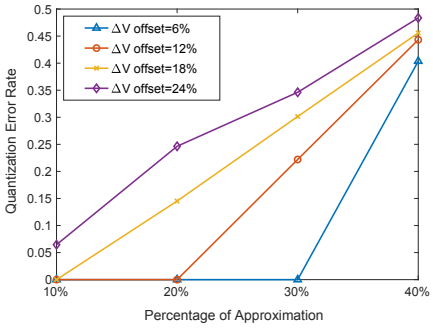


Fig. 4: Abstract view of the attack on APB-3.



(a)



(b)

Fig. 5: Effect of the attack on APB-3: (a) voltage threshold for precise/approximate PCM, and (b) quantization error rate of the compromised PCM cells.

either a cut-off value for the aforementioned configuration parameters or a quality requirement for the application. A conceptual attack on APB-3 is illustrated in Fig. 4. Assume an AC system is initialized with *configuration A* to achieve the performance quantified by the quality metric *QM1*. When an attack alters QM1 deployed in a self-tuning AC system, the lower quality *QM2* will lead the system switch to a new configuration option *A'*.

We use an approximate Phase Change Memory (PCM) as an example to demonstrate the impact of the attack on APB-3 on the quantization error rate of PCM. Depending on the number of read and write iterations, the accumulated voltage on the PCM cell will turn the PCM material into either amorphous or crystalline and thus tune its resistance. As shown in Fig. 5(a), the voltage presented by the PCM material can be quantized into four levels, which means each PCM cell carries two bits. An approximate PCM narrows the guard band (i.e., $\Delta V_{apx} > \Delta V_p$) and adapts the

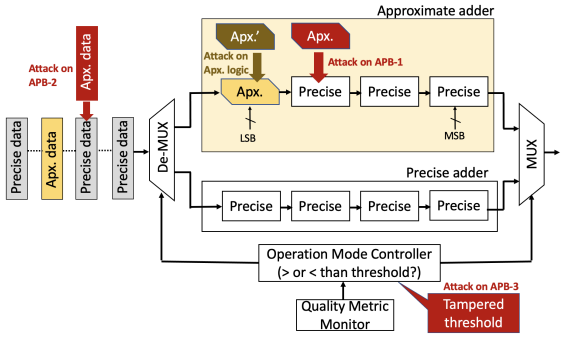


Fig. 6: Demonstration of three APB attacks in a hybrid approximate/precise adder.

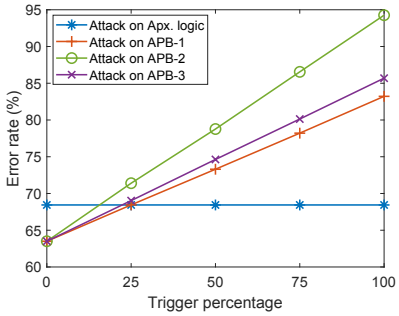


Fig. 7: Impact of different attacks on the error rate of approximate computing systems.

number of iterations in the read and write process. If ΔV_p is compromised by the APB-3 attack in the initialization stage, less write iterations will be applied to the PCM block. As shown in Fig. 5(b), a larger ΔV offset due to the attack leads to a higher quantization error rate. More approximation in PCM will result in more precision loss.

C. Significance of Protecting APB vs. Approximate Module

Section IV-B illustrates the impact of the attacks on APBs on the system output and application quality. In this section, we use an approximate adder and its application to compare the consequence of tampering with three APBs and the outcome of altering the approximate algorithm. Assume that a complete 16-bit approximate adder is composed of one approximate 4-bit adder and three precise 4-bit full adders, as shown in Fig. 6. A general attack on the approximate module *Apx.* will change the logic function into *Apx.'*. In contrast, the APB attacks will occur at three attack surfaces. The attack on APB-1 will cause multiple 4-bit full adders to compute approximately. The attack on APB-2 will make more data blocks to be processed by the approximate adders than what is originally planned. The attack on APB-3 will alter the original threshold, which determines when to use approximate or precise mode, to manipulate the output quality (e.g., error rate of approximation). Our preliminary assessment shown in Fig. 7 indicates that the error rate increase caused by the attacks on APBs is more significant than that originated from the attack on the approximate module itself. This trend grows steadily as the triggering probability of the APB attack increases further. Based on this case study, we believe that it is crucial to protect the APB from various attacks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

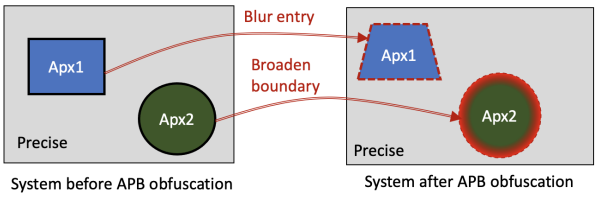


Fig. 8: Overview of proposed APB obfuscation method.

V. Proposed APB Obfuscation Method

A. Method Overview

Typically, an AC system integrates precise and approximate operations together to obtain the degraded but still acceptable accuracy/precision. Regardless of the approximation mechanism at different abstraction level, there is an explicit APB between precise and approximate logic/storage zones. APBs are defined in the specification phase of AC systems' design flow. To address the APB attacks, we propose to secure AC systems by obfuscating the APBs mentioned in Section IV. More specifically, our method is composed of (1) entry-blurring and (2) boundary-broadening schemes. As shown in Fig. 8, the proposed method alters the appearance of the approximation modules at the top level of a computing system. The entry-blurring scheme obfuscates the true entrance of the approximate modules. The boundary-broadening scheme expands the transition zone between the precise and approximate modules. As a result, these two schemes harden the APB from a system point of view and reduce the APB attack success rate. In the following subsections, we explain each scheme in detail.

B. Entry-Blurring Scheme

1) Conceptual View

The traditional sharp transition from precise to approximation mode will lead to a drop on the quality metric λ. Depending on the instant λ obtained from various input and external interference, traditional AC systems compare λ with the mode-transition threshold Λ_C to switch between approximate and precise modes. If the threshold Λ_C is compromised by the APB attack, the application quality will decrease, as shown in Fig. 9. To thwart the APB attack, we propose an entry-blurring scheme, which leverages a hidden quality metric ϵ to obfuscate the mode transition. The hidden quality metric has a stronger correlation with the errors caused by the APB attack than the original (explicit) quality metric, thus providing a higher sensitivity to the attack. Moreover, the hidden quality metric is transparent to users and does not have a direct-access interface available to the external world, making the system less vulnerable to APB attacks than the system only using an explicit quality metric. If an attack leads to a large decrease in λ, the AC system protected by our countermeasure will choose proper precise and approximate modules to perform the reliable computation and approximately maintain the expected application quality. The non-linear transition between the precise and approximate modes shown in Fig. 9 is designed to thwart the APB attacks, either tampering with or reverse engineering the APB.

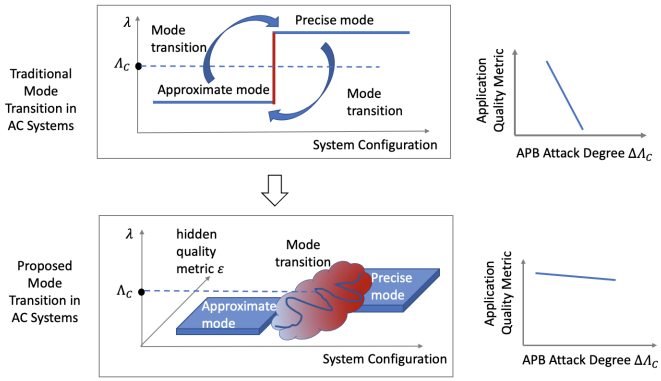


Fig. 9: Conceptual view of proposed entry-blurring scheme.

2) Mathematical Modeling

The quality metrics interested in applications direct the selection of approximate or precise computation. To determine the hidden quality metric, one needs to examine the dependent factors of the errors in the approximation process. It varies with applications where AC mechanisms are applied to. For instance, the most common quality metric used in image processing applications is Peak Signal-to-Noise Ratio (PSNR). If the application using approximate computation achieves a PSNR below the threshold, the application needs to switch to precise computation. Assume that the function $\Lambda(\cdot)$ is the metric to assess the quality of deployed computing system, X_{in} is the application input, and Out_{ref} is the reference output. We define the overall configuration function $h(\cdot)$ in Eq. (1). The functions $F_{apx}(\cdot)$ and $F_{precise}(\cdot)$ configure the parameters in AC systems.

$$h(X_{in}) = \begin{cases} F_{apx}(X_{in}) & \lambda > \Lambda_c \\ F_{precise}(X_{in}) & \lambda \leq \Lambda_c \end{cases} \quad (1)$$

In which, Λ_c is the cutoff threshold for the quality metric explicitly declared in the user manual and λ is the instant quality assessed by Eq.(2).

$$\lambda = \Lambda\left(h(X_{in}), Out_{ref}\right) \quad (2)$$

The proposed obfuscated transition method changes $h(\cdot)$ in Eq. (1) to $H(\cdot)$ in Eq. (3).

$$H(X_{in}, \Lambda_c, \epsilon) = \begin{cases} F_{apx}(X_{in}) & \lambda > \Lambda_c \\ \Psi\left(X_{in}, G_{apx}, \epsilon\right) & \lambda \leq \Lambda_c, \epsilon < E_{hidden} \\ F_{precise}(X_{in}) & \lambda \leq \Lambda_c, \epsilon \geqslant E_{hidden} \end{cases} \quad (3)$$

Where the function $\Psi(\cdot)$ describes the obfuscated mode transition method, which searches for an alternative approximate function G_{apx} from an implicit function array $\{\mathbf{G}|G_{apx} \in [G_{apx_1}, G_{apx_2}, ..., G_{apx_n}]\}$. The function $\Psi(\cdot)$ stands for an iterative searching process to find the best approximation function to satisfy the expectation on Λ_c and the hidden quality metric E_{hidden}. The selection of the explicit and hidden metrics (λ and ϵ) and the target quality (Λ_c and E_{hidden}) depends on the specific AC application.

We assume that a candidate list of precise and approximate computation modules is available in the IP library, in which diverse IPs either use different approximation mechanisms or have tunable parameters to achieve different

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

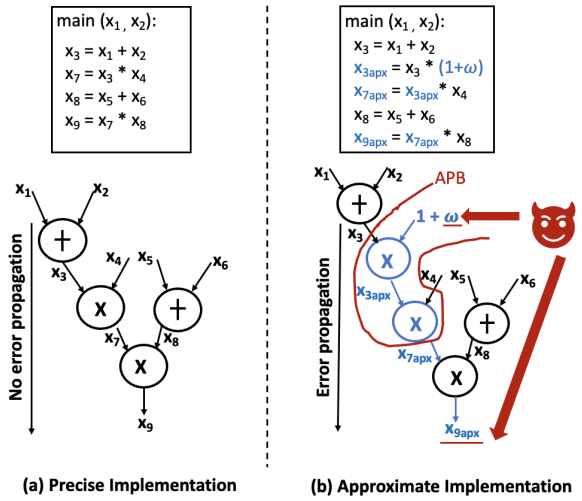


Fig. 10: Illustration of an attack surface in the DFG for an approximate computing system.

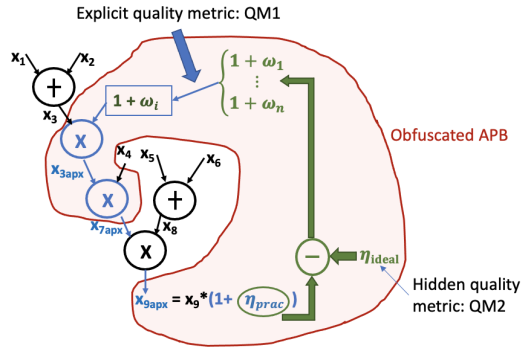


Fig. 11: The APB in the DFG example protected with proposed entry-blurring scheme.

Algorithm 1: Proposed entry-blurring scheme deployed in an image processing application.

Data: Application A; $QM_{spec} = PSNR_{spec}$;
Approximate IPs from library $Apx[1:N]$;
Image size: Row, Col;
Result: Apx-IP-index

```
Run the application A without any approximation to obtain Out_pre;
Calculate η_ideal;
i=1;
while j ≤ N do
    Run the application A with approximate setting Apx[j] to obtain Out_apx;
    Calculate PSNR;
    if Attack Free then
        PSNR'_spec = PSNR_spec
    else
        PSNR'_spec = attacker assigned arbitary PSNR'
    end
    Calculate η_prac;
    if (PSNR>= PSNR'_spec) and (η_ideal >= η_prac) then
        Use Apx[j] in Application A;
        array[i] = Apx[j];
        PSNR_cal[i] = PSNR;
        i++;
    else
        Discard Apx[j];
    end
    j++;
end
Return the index of minimum value from the array PSNR_cal as Apx-IP-index;
Note: PSNR, η_ideal, and η_prac can be obtained from Eq. (10, 15,16), respectively, in Appendix.
```

performance and accuracy. In the phase of deploying the AC algorithm in an application, we initialize the explicit quality metric and its intended goal. The proposed $\Psi(\cdot)$ function enables the mitigation of the attacks on APB.

3) Deploying Entry-blurring Scheme in An Application

We take an approximate arithmetic function as a subject to demonstrate the attack surface and the deployment of our entry-blurring scheme. The main function and its data flow graph (DFG) are described in Fig. 10. We assume that the sum of x_1 and x_2 is computed by an approximate adder. Following the work [32], we model the effect of approximation by introducing an error characteristic ω, which represents system configuration of an approximate IP, to one of the multiplier inputs, x_3. Thus, the approximation error propagated from the approximate adder to other nodes in DFG is expressed in Eq. (4).

$$x_{7apx} = [x_3 * (1+\omega)] * x_4 \tag{4}$$

The error due to the approximate adder will lead x_{7apx} to deviate from the precise product of x_3 and x_4. The deviation will be further propagated to the final output x_9. We define the deviation on the final output as a ratio η_{prac} and thus obtain x_{9apx} as denoted in Eq. (5).

$$x_{9apx} = x_9 * (1+\eta_{prac}) \tag{5}$$

In this example, the attack surface is the entrance of selecting an approximate adder for x_1 and x_2. Theoretically, the diverse choices for the approximate adder can be modelled as introducing x_{3apx} with a different ω. The attack on the APB leads a difference between x_{9apx} and x_9, which exceeds the range that can be tolerated by the target application.

The proposed entry-blurring scheme will obfuscate the APB highlighted in Fig. 10. The key idea of our obfuscation is to collaboratively use an explicit quality metric (QM1) and a hidden quality metric (QM2) to select the best-fit approximate IP, rather than only relying on QM1 to configure the approximate system. In the application of image processing, QM1 is PSNR and QM2 is average relative error (ARE). ARE is a specific quality metric that compares the ideal (η_{ideal}) and practically (η_{prac}) obtained primary outputs. The APB attack on the approximate adder selection will

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

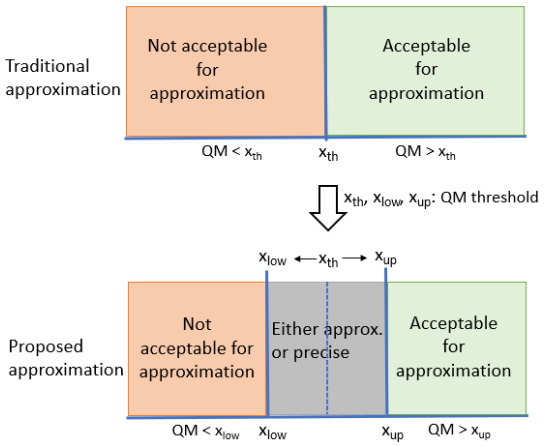
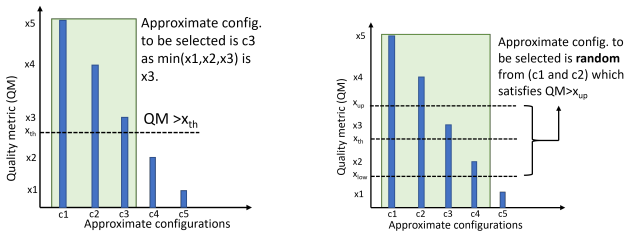


Fig. 12: Conceptual view of boundary-broadening scheme.



(a) (b)

Fig. 13: The selection of approximation configuration using (a) baseline algorithm [28] and (b) proposed boundary-broadening scheme.

lead to an unexpected ARE. As the hidden quality metric ARE is directly and strongly tied with the approximation errors, we can use ARE to detect the presence of attacks. The derivation of ARE is introduced through Eq. (10) to Eq. (15) in the appendix. As shown in Fig. 11, our scheme incorporates QM1 (PSNR) and QM2 (η_{ideal}) to tune the system configuration ω (i.e., selecting a proper approximate adder). The hidden quality metric QM2 is an interpretation of QM1. The detailed selection procedure for an image processing application is presented in Algorithm 1.

C. Boundary-Broadening Scheme

1) Conceptual view

The proposed boundary-broadening scheme is another way to mitigate APB attacks. Our scheme replaces an explicit cut-off threshold with a vague transition zone, which acts as a buffer for approximate and precise mode switching. The concept of boundary-broadening scheme is depicted in Fig. 12. In traditional AC systems, the proper computing mode (i.e., approximate or precise) is determined based on the quality metric (QM) of interest. Only when the QM at the check point is greater than the QM threshold x_{th}, the adopted approximate IP will guarantee to meet the minimum quality requirement of the application. In our scheme, we expand the threshold x_{th} to two thresholds, x_{low} and x_{up}. The buffered area between x_{low} and x_{up} is considered as an obfuscated transition zone.

In a real scenario such as the one shown in Fig. 13, there will be multiple approximate configurations to enable the computing system reach the desired quality metric. Assume there are five approximate configurations c1~c5, which results in its quality metric x1~x5. If the instant computation quality is higher than x_{th}, the baseline scheme selects c3 out of the group of c1, c2 and c3 to minimize the hardware cost; otherwise the system opts out the approximate mode. In contrast, the proposed boundary-broadening scheme sets two thresholds x_{up} and x_{low} and adds c4 approximation configuration to the group of selection candidates. When the instant computation quality QM is between x_{up} and x_{low}, the boundary-broadening scheme randomly selects a configuration from the group of c1 and c2. From this example we can see, our boundary-broadening scheme (1) introduces some randomness to the selection of approximate configurations, and (2) expands the number of configuration choices that can eventually enable the system to achieve the desired quality.

2) Mathematical Modeling

The two thresholds x_{low} and x_{up} depend on the specific application. To simplify the analysis, we assume that the single threshold x_{th} is located in the middle of x_{low} and x_{up}, as expressed in Eq. (6).

$$x_{th} = \frac{(x_{low} + x_{up})}{2} \tag{6}$$

We continue to use PSNR as the quality metric to demonstrate the process of deriving x_{low} and x_{up} from the original single threshold x_{th}. Equation (7) expresses the relation among these three thresholds.

$$PSNR_{th} = PSNR_{spec} = \frac{(PSNR_{low} + PSNR_{up})}{2} \tag{7}$$

Based on the derivation in Appendix, we have $PSNR_{low}$ and $PSNR_{up}$ expressed in Eqs. (8) and (9), respectively, where α_1 stands for the deviation of MSE_{low} from MSE_{spec} .

$$PSNR_{low} = 10log_{10}\left(\frac{(2^{NB}-1)^2}{MSE_{spec}+\alpha_1}\right) \tag{8}$$

$$PSNR_{up} = (2 * PSNR_{spec}) - PSNR_{low} \tag{9}$$

3) Deploying Boundary-Broadening Scheme in An Application

We apply the proposed boundary-broaden scheme to an image processing application, Sobel edge detection. The entire process of the deployment is described in Algorithm 2. Readers can follow the same procedure (except replace PSNR with other quality metrics of interest) to apply our APB obfuscation method to other applications. In our case study, the quality metric is PSNR. The $PSNR_{cal}$ is obtained by running the Sobel edge detection application with approximate IPs. The $PSNR_{spec}$ is provided by the designer during the design specification. The $PSNR_{cal}$ is

Algorithm 2: Proposed boundary-broadening scheme deployed in an image processing application.

Data: Application A; Approximate IPs from library Apx[1:K]; $QM_{spec} = PSNR_{spec}$; $QM_{cal}(= PSNR_{cal})$[j] corresponding to each $Apx[j], j \in (1, K)$, where K is the number of approx. IPs that satisfies the APB condition;

Result: Apx-IP-index

```
Calculate MSE_spec from PSNR_spec ;
Assume a value for α_1;
Derive PSNR_up and PSNR_low , respectively;
Arrange the PSNR_cal array in ascending order;
Change the order of Apx array according to PSNR_cal;
j =1;
while j ≤ K do
    if PSNR_cal[j]>= PSNR_up then
        assign Apx-IP-index = j;
    else
        if PSNR_low < PSNR_cal[j] < PSNR_up then
            Randomly select a value from array whose index is n, where n ∈ (1, K);
            assign Apx-IP-index = n;
            break;
        else
            Discard Apx[j];
        end
    end
    j++;
end
Return Apx-IP-index;
Note: MSE_spec, PSNR_low, and PSNR_up are derived in Eqs. (18, 21, 22), respectively, shown in Appendix.
```

compared with the $PSNR_{spec}$ to check if the approximate IP used by the application satisfies the minimum quality requirement of the system. If the $PSNR_{cal}$ is no less than $PSNR_{spec}$, that approximate IP will be adopted by the application; otherwise, it will be discarded. We assume that the attacker changes the $PSNR_{spec}$ to $PSNR'_{spec}$ to conduct the APB-3 attack. To protect the system from such an attack, we replace the single threshold with two thresholds $PSNR_{low}$ and $PSNR_{up}$ derived by Eqs. (8) and (9). Now, when the approximate IP used in the Sobel edge detection produces an instant $PSNR_{cal}$ that is in the range between $PSNR_{up}$ and $PSNR_{low}$, the proper approximate IP will be a random one from the library $Apx[1:K]$, rather than the approximate IP providing $PSNR_{cal}$.

VI. Experimental Results

A. Experimental Setup

1) Baseline and Proposed Scheme Deployment

As approximate computing has been widely applied in image processing, we evaluated the impact of APB attacks and proposed attack mitigation method on an image processing application—approximate Sobel edge detection.

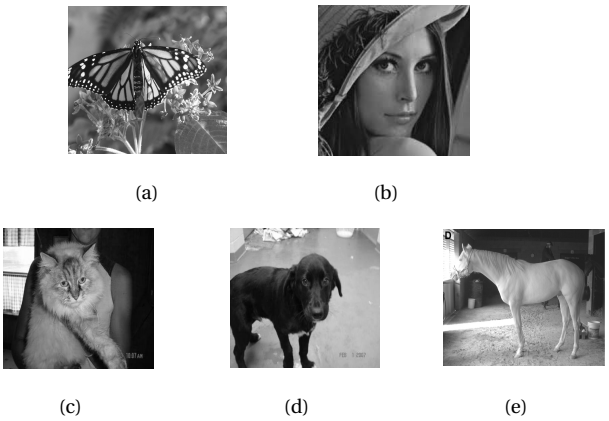


Fig. 14: Original images used in our evaluation: (a) butterfly, (b) Lena, (c) cat, (d) dog, and (e) horse.

Five images shown in Fig. 14 were utilized in our assessment. We adopted the approximate edge detection presented in [28] as our baseline, in which the quality metric PSNR directed the selection of proper approximate computing IPs for the application. The baseline and our methods were implemented and simulated in Matlab.

2) Attack Scenario in Our Evaluation

In our following experiments, the APB is the logic that determines whether the AC system uses approximate or precise operations. The quality threshold used in the APB is a user-specified PSNR. We assume that attackers have access to the quality control knob and can introduce malicious approximate IPs to the library of IP candidates for the AC system. We further assumed that the attacker has access to the row indicator and column indicator of the image being processed. We have designed two types of triggering logic: (1) trigger is always-on and (2) trigger is activated only when a particular row or column is under processing. When the triggering condition arrives, the specific APB attacks we applied to the case studies include (1) adding malicious approximate IPs to the IP library to cause unexpected precision reduction at unique trigger conditions (APB-1 and APB-2 attacks) and (2) tampering with the PSNR threshold during the AC system's initialization phase to degrade the overall computation accuracy (APB-3 attack). The occurrence frequency of all attacks were managed by an external triggering controller, which was independent with the application inputs.

B. Impact of APB Attacks on Image Quality

Assume that an attack sabotages the APB in the process of AC system initialization by changing the original PSNR ($PSNR_{spec}$) from 30db to a lower value ($PSNR'_{spec}$), 20db or 10db. Consequently, the APB attack will lead the AC system to overuse the approximate operations. We compared the PSNRs achieved at the end of the approximate Sobel edge detection without and with the APB attack. Figure 15 illustrates one example of the outputs of the approximate edge detection application suffering from the different-degree APB attacks. More images were used in our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

(a) (b) (c)

Fig. 15: Output of approximate edge detection without/with APB attacks: (a) $PSNR_{spec}$=30db, (b) $PSNR'_{spec}$=20db, and (c) $PSNR'_{spec}$=10db.

TABLE II: Obtained Application Quality (PSNR) of Approximate Image Edge Detection with/without APB Attacks.

Images	Without APB attack ($PSNR_{spec}$=30db)	With APB attack ($PSNR'_{spec}$=20db)	With APB attack ($PSNR'_{spec}$=10db)
Butterfly	34.60db	21.00db	14.49db
Horse	32.93db	26.25db	13.21db
Lena	30.11db	23.78db	17.38db
Cat	35.04db	21.74db	15.25db
Dog	32.94db	25.89db	12.28db
Average	33.12db	23.73db (79% $PSNR_{spec}$)	14.52db (48% $PSNR_{spec}$)

assessment. As shown in the second column of Table II, if no APB attack is conducted, the real PSNR is higher than the desired quality $PSNR_{spec}$. When the APB is tampered by the attacker, the final quality of the application cannot reach the target PSNR. As listed in the third and fourth columns of Table II, the two APB attacks reduce the average real PSNR to 79% and 48% of $PSNR_{spec}$. This assessment indicates that the APB attack indeed leads to a significant quality drop and needs effective attack mitigation.

C. Evaluation of Entry-Blurring Scheme

1) Sensitivity of Quality Metrics to APB Attacks

The errors induced by approximate operations may not have a strong correlation with the final quality interested in the application. When exploring a hidden quality metric for the purpose of attack detection/mitigation, we customize the metric to enlarge the correlation between the output deviation and the system configuration choices. Assume that the APB attack alters $PSNR_{spec}$ from 30db to 20db. Figure 16 shows that the hidden quality metric ARE we selected for attack mitigation is more sensitive to the APB attack than the explicit quality metric PSNR, which is measured from the primary output of the edge detection on the butterfly image. When the triggering probability of the APB attack increases from 20% to 100%, the change in the quality metric increases from 2× to 31×.

Next, we examine the sensitivity of the hidden quality metric to the severeness of the APB attack. Figure 16 only shows the sensitivity of the hidden quality metric to the APB attack altering $PSNR_{spec}$ by 33.3%. Now, we made the APB attack reduce the original $PSNR_{spec}$ by 20% to 100%. As shown in Fig. 17, the hidden quality metrics for the horse image without attack (η_{ideal}) and with attack ($\eta_{practical}$) have significant difference, varying from 99.5% to 306.3%. A larger increase in the hidden quality metric will provide higher confidence in the detection of APB attacks.

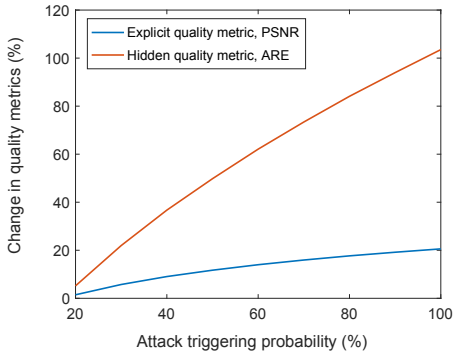


Fig. 16: The sensitivity of explicit and hidden quality metrics to APB attacks.

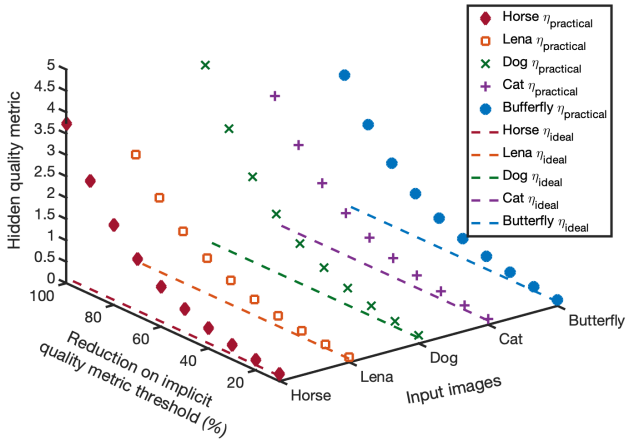


Fig. 17: Sensitivity of hidden quality metric to the APB attack that reduces the quality threshold applied in the APB.

We repeated the same experiment for the other four images listed in Section VI-A1. The results shown in Fig. 17 confirm that the sensitivity of our hidden quality metric to the APB attack is approximately the same for different images.

2) Efficiency of Proposed APB Attack Mitigation

The baseline [28] provides four system configurations 0, 0.004, 0.056 and 0.12, each of which represents the relative error yielded by a combination of the deployed approximate IPs [32], [33]. We further introduce two new configurations, 0.25 and 0.5, to consider the APB attack that results in exchanging the approximate and precise bit positions or accelerating the propagation of approximate errors throughout the AC system [7]. As shown in Fig. 18, each approximation configuration yields a unique value for the hidden quality metric we selected for attack mitigation. The characteristic of hidden quality metric versus different approximate configurations varies with the input image, but the general trend remains the same.

The entry-blurring algorithm was applied to the approximate Sobel edge detection. The minimum PSNR requirement was set to 30db. For each image, we measured the corresponding ARE for this given PSNR requirement and denoted it as a reference. As shown in Fig. 19, the baseline without using any protection mechanisms suffers from the APB attack, resulting in an average increase on ARE by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

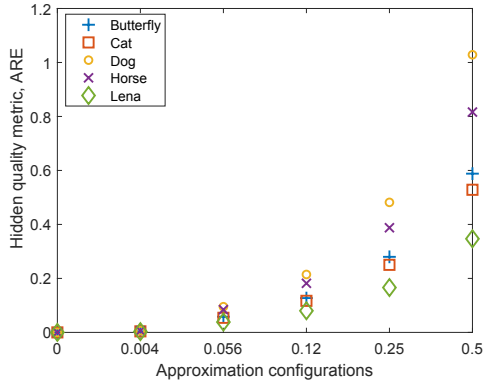


Fig. 18: Variation on the hidden quality metric due to different approximate computing configurations.

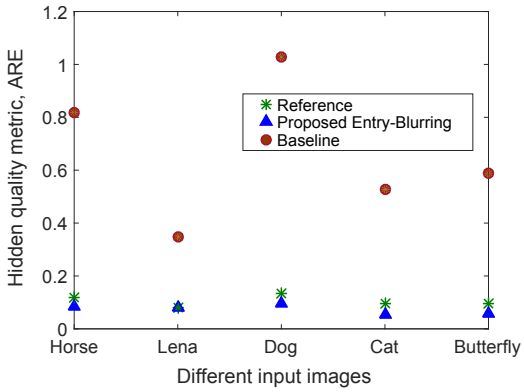


Fig. 19: Successful APB attack mitigation by proposed entry-blurring scheme.

82.9% over the reference. In contrast, the ARE achieved by the proposed entry-blurring scheme is always less than the reference, which indicates that our approximate computing results met the minimum PSNR requirement and thus the APB attack was mitigated successfully.

The result shown in Fig. 19 is for the scenario that the APB attack is always triggered. In practical applications, the APB attack may happen occasionally. Thus, we further examine the final quality of the AC application upon the attack with different triggering probabilities. As shown in Fig. 20, the proposed entry-blurring scheme assures that the final quality is always above $PSNR_{spec}$, regardless of the attack triggering probability and the input images. If the attack is always triggered, our scheme improves the final PSNR by 149%, 73%, 168%, 130% and 139% for the image horse, Lena, dog, cat, and butterfly, respectively, compared to baseline. As each image has its unique pixel patterns, the tampered approximation in the edge detection causes distinctive error propagation and thus the final PSNR achieved by our scheme slightly varies with the input images. As the attack triggering probability reduces to 20%, our attack mitigation scheme still obtains the average improvement of 53% over the baseline without entry blurring.

3) Overhead on Latency

We analyzed the computation complexity overhead of the proposed entry-blurring scheme applied in the Sobel edge detection application. As the total number of arithmetic operations in the application depends on the size of the input image, the computation complexity reported in this section is for an image composed of 256x256 pixels. The baseline is the method that only uses a single explicit quality metric in the APB. In contrast, our scheme needs to calculate both explicit and hidden quality metrics. That is why more computations are required in the application. The overall arithmetic operations required for the baseline and proposed entry-blurring scheme are summarized in Table III. We assume that the image processing application is run in an Intel Ivy Bridge processor, in which the latency of basic operations is reported in [34]. Given an image with 256x256 pixels, the overall latency overhead incurred by the proposed method is 0.0013%, which is negligible.

TABLE III: Latency Overhead. Unit: clock cycles (cc).

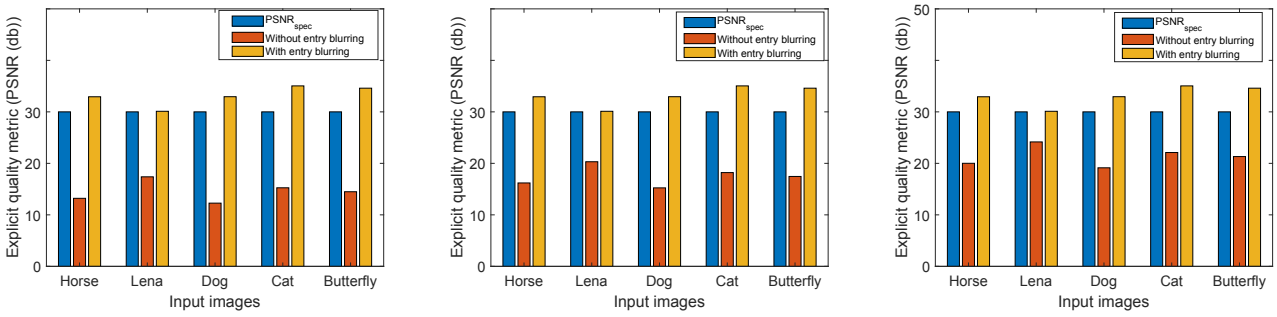
Arithmetic operations	Operation overhead	Unit latency (cc)	Baseline (cc)	Proposed scheme (cc)
Addition	0	1	2258060	2258060
Multiplication	0	3	7161276	7161276
Division	38	19	19	57
Subtraction	2	1	65536	65538
Square	0	3	1167411	1167411
Square-root	10	10	1290320	1290330
Mean	40	20	40	80
Transpose	0	19	19	19
Log	60	30	30	90
Total	*0.0013%*	—	*11942711*	*11942861*

D. Evaluation of Boundary-Broadening Scheme

1) Improved Attack Resilience

We also applied the proposed boundary-broadening scheme to the approximate Sobel edge detection and implemented Algorithm 2 in Matlab. We set α_1 in Eq. (8) to $2^{*}MSE_{spec}$. The parameter MSE_{spec} can be obtained from the given $PSNR_{spec}$. Assume that the desired application quality $PSNR_{spec}$ is 30db. According to Eqs. (8) and (9), the two thresholds used in our APB-attack-mitigation mechanism are 26.36db and 33.64db. The baseline [28] uses a single threshold (30db) to determine which approximate IP will obtain the minimum acceptable quality on accuracy. In contrast, the proposed boundary-broadening scheme leverages the two thresholds to randomly choose one approximate configuration from the candidate list. Table IV compares the possible options and the final choice for the horse image in the process of approximate edge detection. If an attacker alters the APB threshold in the baseline from 30db to 20db, the final approximate configuration adopted by the baseline will be shifted from 0.056 to 0.25. Due to the broadened APB boundary, our attack mitigation method will make more configuration options available for the system, as shown in the most bottom right of Table IV. Figures 21(a) and (b) indicate that our mitigation scheme will provide more system configuration options for more severe modifications on the APB threshold. The standard deviation of the final PSNRs achieved by all options is over 22. The various configuration options offered by our method will contribute to the resilience against reverse engineering attack on the deployed attack mitigation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

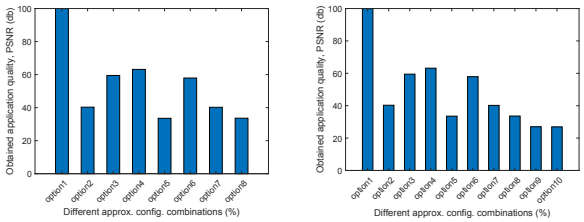


(a) (b) (c)

Fig. 20: The final PSNR achieved by different approximate edge detection algorithms under the APB attack with a triggering probability of (a) 100%, (b) 50% and (c) 20%.

TABLE IV: The Approximate Configuration Options Available in Our Experiments.

Selection scheme	Approx. config. to meet target quality (no attack)	Approx. config. to meet target quality (with attack)	Final selection of approx. config. (no attack, with attack)
Baseline	0, 0.004, 0.056	0, 0.004, 0.056, 0.12, 0.25	(0.056, 0.25)
Proposed boundary-broadening	0, 0.004	0, 0.004, 0.056, 0.12	Randomly select from: (0, 0); (0, 0.004); (0, 0.056); (0, 0.12); (0.004, 0); (0.004, 0.004); (0.004, 0.056); (0.004, 0.12)



(a) (b)

Fig. 21: The random options available in the proposed boundary-broadening scheme when the attack reduces the APB threshold by (a) 33% and (b) 66%.

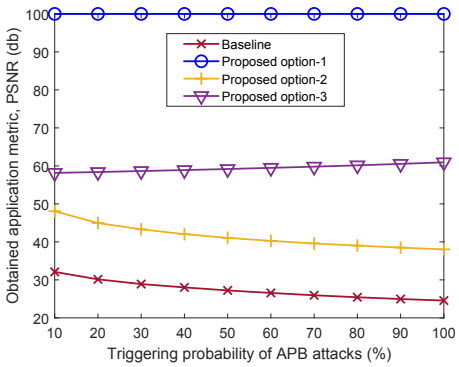


Fig. 22: PSNR improvement achieved by proposed boundary-broadening scheme.

The broadened APB boundary improves the system's resilience against the APB attack that sabotages the threshold for precise and approximate boundary. As shown in Fig. 22, when the single threshold for APB is used [28] and the APB of the AC system is under attack, the final quality represented by PSNR is decreased from 30db to 25db. In contrast, the deployment of two APB thresholds improves the PSNR, regardless how often the APB attack

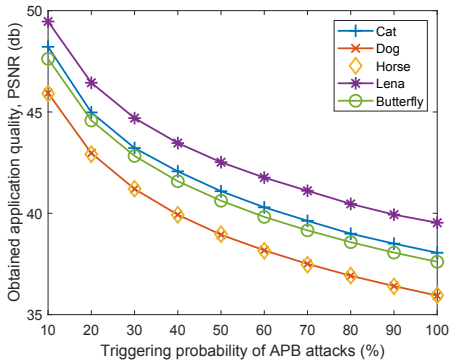


Fig. 23: Our PSNR improvement for different input images.

is triggered. The random configuration options built in our scheme makes the relation of PSNR and attack triggering probability unique and unpredictable. As shown in Fig. 22, the proposed boundary-broadening improves the application quality by 49.7%, 81.2%, and 211.5% for the random options 1, 2, 3, respectively, when the APB attack triggering probability is 10%. If the attack happens more often, our PSNR improvement will be increased to 170% on average. More interestingly, the trend of three two-threshold options could decrease, increase, or remain flat with the increasing APB attack probability. The option3 in Fig. 22 represents the precise IP, and other two options (1 and 2) are approximate IPs with different system configurations. Each option enables the AC system to achieve a PSNR that is higher than the user-specified requirement. Although our random choice in protection incurs more power consumption than a static approximate configuration, it still consumes less power than a fully precise computing configuration. Next, we evaluate whether the PSNR improvement achieved by our method is input dependent or not. We assume that all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

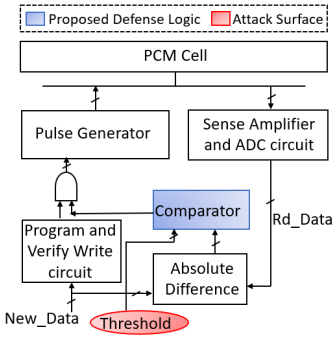


Fig. 24: The attack surface and the location of defense mechanism in approximate PCM.

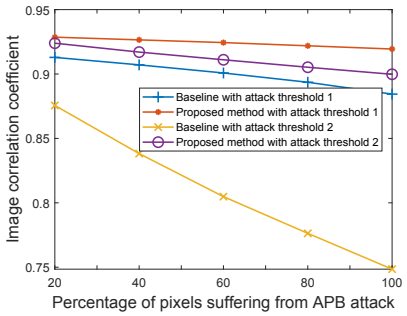


Fig. 26: Attack mitigation achieved by proposed boundary-broadening scheme.

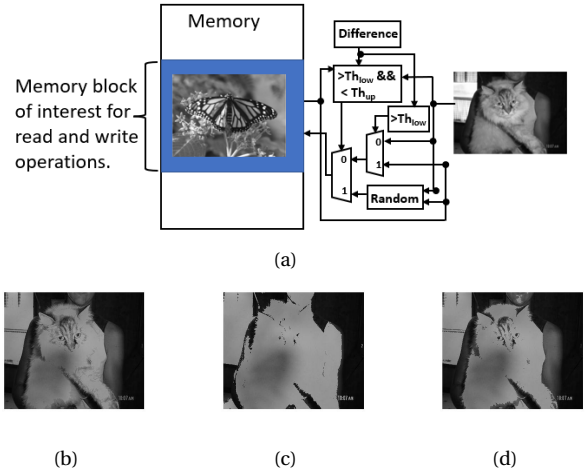


Fig. 25: APB attack mitigation in approximate PCM. (a) An example of write operation, (b) the original approximate PCM without attack, (c) the original approximate PCM with APB attack, and (d) the boundary-broadening protected approximate PCM with APB attack.

input images require the same approximate configuration in the initialization stage. As shown in Fig. 23, different inputs do not affect the trend of how PSNR changes with the attack triggering probability.

2) Improved Attack Resilience in Approximate PCM

Approximate PCM adapts the number of iterations for its write operation to save power consumption and improve performance [19], [35], [36]. If the difference between the new data and the current content saved in the target memory cell is lower than the voltage threshold, the writing operation is shortened. Here, the voltage threshold is the APB in this particular approximation mechanism. Figure 24 highlights the attack surface in an approximate PCM [37]. We applied the proposed boundary-broadening scheme to the comparator so that the write operation of the PCM is resilient to the attack on APB-3. As our boundary-blurring method is implemented in the comparator (hardware), the protection mechanism will protect the system from attacks at run-time, rather than at an initial stage.

We implemented the write circuit for approximate PCM presented in the work [37] as our baseline in this section. The flowchart in our evaluation is depicted in Fig. 25(a). We stored an butterfly image in the dedicated approximate PCM cells. A new cat image was the new data to be written to the same memory location. Without the APB attack, the cat image will be successfully overwritten to the PCM cells and the new image will be read out as the one shown in Fig. 25(b). When the attack changes the APB threshold used in the comparator, the new image stored in the PCM is distorted as illustrated in Fig. 25(c). From this example, we can conclude that the quality of the approximate write operation is reduced by the APB attack. However, once the proposed boundary-broadening mechanism is employed to the approximate PCM, the comparator logic checks if the difference between the cat image and butterfly image is in the range of Th_{low} and Th_{up}, less than Th_{low}, or greater than Th_{up}. Because of the modified comparator logic highlighted in the middle of Fig. 25(a), the image in Fig. 25(b) for without attack and the image in Fig. 25(d) for with attack are visualized almost the same. If the attacker makes more increase on the APB threshold of PCM, less new data will be written to the memory. We assume the attacker increases the APB threshold from 30 to 60 (i.e., attack threshold 1) and 100 (i.e., attack threshold 2). We used the Pearson correlation coefficient to quantify the image similarity before and after the attack. As shown in Fig. 26, the proposed scheme facilities the approximate PCM to write the new image with a correlation coefficient of over 0.9 (if two images are identical, the correlation coefficient is 1). In contrast, the baseline that uses a single threshold results in the image having a lower image correlation coefficient. As the percentage of attacks increases, the proposed method successfully mitigates the attack and only incurs 2.7% drop on the image correlation, but the baseline does not have the attack resilience and leads the image correlation coefficient drop by 16.7% in our case study.

The logic for approximate write operation and our defense scheme was written in Verilog HDL and synthesized by Synopsys Design Compiler with a TSMC 65nm technology. The hardware cost for the proposed scheme is listed in Table V. The boundary-broadening scheme costs 16.7% more area and 16.4% more power compared to the PCM approximate write mechanism with one threshold. The area and power overhead is induced by the calculation of the expanded thresholds Th_{low} and Th_{up}. Alternatively,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3168261, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE V: Area and Power Consumption of Approximate PCM Write Logic Module.

Design	Area (nm^2)	Dynamic Power (μW)	Leakage Power (μW)
Baseline	1230.84	479.0475	4.9477
Proposed Method, (two thresholds from hardware)	1436.76	557.8565	5.7897
Proposed Method, (two thresholds from software)	1309.32	521.57	5.2867

these two thresholds can be pre-computed and read from software. Then, our overhead on area and power is reduced to 6% and 8%, respectively.

VII. Conclusion

Approximate computing has become increasingly popular because it is promising to further improve performance and energy efficiency. However, the utilization of approximate techniques also make AC systems vulnerable to new attacks. We propose three APB boundaries observed in the common AC systems and demonstrate some practical attack examples, which have the potential to inspire more researchers developing unique countermeasures for AC systems. To mitigate those attacks, we further introduce entry blurring scheme and boundary broadening scheme to obfuscate the APB. Our experimental results show that the proposed entry blurring scheme improves the PSNR by up to 53% over the baseline when the attack triggering probability is 20%. When the APB attack becomes always triggered, our method achieves 168% more PSNR than the baseline. The latency overhead for the entry-blurring scheme is negligible. The proposed boundary-broadening scheme also enables the AC system to mitigate the APB attack and meet the quality requirement of an approximate image processing application. Our case study on approximate PCM further confirms that the attack resilience is improved by our APB obfuscation method. The overhead on area and power induced by the boundary-broadening scheme is around 16%. We can leverage software support to reduce the area and power overhead to 6% and 8%.

Appendix

The quality metric Peak-Signal-Noise-Ratio (PSNR) is defined in Eq. (10).

$$PSNR = 10log_{10}\frac{\left(2^{NB}-1\right)^2}{MSE} \tag{10}$$

in which, NB is the number of the bits representing each pixel. The Mean Square Error (MSE) for the comparison of original and processed images is denoted by Eq. (11).

$$MSE = \frac{\sum_{i=1}^{i=Row}\sum_{j=1}^{j=Col}\left(Out_{ij.apx}-Out_{ij.pre}\right)^2}{Row*Col} \tag{11}$$

Where $Out_{ij.apx}$ is an approximate output, $Out_{ij.pre}$ is an precise output for a general image pixel in row i and column j, and the maximum range of i and j are *Row* and *Col*, the dimension of an image in processing. We define the ARE for a general pixel in the image as η_{ij} and use η_{ideal} to stand for the expected average η_{ij} for the selected approximate configuration.

$$\eta_{ij} = \frac{Out_{ij.apx}}{Out_{ij.pre}} - 1 \tag{12}$$

$$\eta_{ideal} = \frac{\sum_{i=1}^{i=Row}\sum_{j=1}^{j=Col}\eta_{ij}}{Row*Col} \tag{13}$$

By substituting MSE and $Out_{ij.apx}$ in Eq. (10) with Eqs. (11,12,13), we can rewrite $Out_{ij.apx}$ and then rearrange the PSNR in a format shown in Eq. (14).

$$\begin{aligned} PSNR = {} & 20log_{10}\left(2^{NB}-1\right) - 20log_{10}\left(\eta_{ideal}\right) \\ & -10log_{10}\left(\frac{\sum_{i=1}^{i=Row}\sum_{j=1}^{j=Col}Out_{ij.pre}^2}{Row*Col}\right) \end{aligned} \tag{14}$$

As a result, given a PSNR required by the application, we are able to derive the ideal value of the hidden quality metric. After rearranging Eq.(14), we can obtain the theoretical ARE on the final output $x9$ as expressed in Eq. (15).

$$\eta_{ideal} = 10^{log_{10}\left(2^{NB}-1\right)-\frac{PSNR}{20}-\frac{1}{2}log_{10}\left(\frac{\sum_{i=1}^{i=Row}\sum_{j=1}^{j=Col}Out_{ij.pre}^2}{Row*Col}\right)} \tag{15}$$

In the boundary-broadening scheme, we measure the Average Relative Error (ARE) at run-time using Eq. 16. The comparison between η_{ideal} and η_{prac} will indicate if the IP is legitimate or not.

$$\eta_{prac} = \sqrt{\frac{MSE}{\left(\frac{\sum_{i=1}^{i=Row}\sum_{j=1}^{j=Col}Out_{ij.pre}^2}{Row*Col}\right)}} \tag{16}$$

As defined in Eq. (10), PSNR is a function of MSE. We can also obtain the relation between MSE_{spec} and MSE_{low} and MSE_{up} from Eq. (17). The $\alpha 1$ and $\alpha 2$ are the deviation values of MSE_{low} and MSE_{up} from MSE_{spec} value respectively.

$$\begin{aligned} MSE_{spec} &= \sqrt{MSE_{low}*MSE_{up}} \\ &= \sqrt{\left(MSE_{spec}-\alpha_1\right)*\left(MSE_{spec}+\alpha_2\right)} \end{aligned} \tag{17}$$

In which, MSE_{low} and MSE_{up} are the shifted mean square error for $PSNR_{low}$ and $PSNR_{up}$, respectively. The variables α_1 and α_2 can have a minimum value of 2MSE and they stand for the offset from the mean square error (MSE) due to the proposed boundary-broadening scheme. Solving Eq. (17), we can see that MSE_{spec} determines the offset as described in Eq. (18).

$$MSE_{spec} = \frac{\alpha_1*\alpha_2}{\alpha_1+\alpha_2} \tag{18}$$

In Algorithm 2, we continue to use PSNR as the quality metric to demonstrate the process of deriving two thresholds x_{low} and x_{up} from the original single threshold x_{th} defined in Eq. (19).

$$x_{th} = PSNR_{spec} \tag{19}$$

Thus, we can also substitute x_{low} and x_{up} with $PSNR_{low}$ and $PSNR_{up}$, respectively. Thus, the average of the two quality metric thresholds will be equal to the target quality metric $PSNR_{spec}$ as expressed in Eq. (20).

$$PSNR_{th} = PSNR_{spec} = \frac{(PSNR_{low} + PSNR_{up})}{2} \quad (20)$$

When we deploy this scheme in a specific application, we can assume either α_1 or α_2 and derive another one from Eq. (18). Next, we can derive $PSNR_{low}$ and $PSNR_{up}$ using the relation expressed in Eqs. (21) and (22), respectively.

$$PSNR_{low} = 10log_{10}\left(\frac{(2^{NB}-1)^2}{MSE_{spec} + \alpha_1}\right) \quad (21)$$

$$PSNR_{up} = (2 * PSNR_{spec}) - PSNR_{low} \quad (22)$$

REFERENCES

[1] L. Sekanina, "Introduction to approximate computing: Embedded tutorial," in *Proc. DDECS*, 2016, pp. 1–6.

[2] P. Yellu, L. Buell, M. Mark, M. A. Kinsy, D. Xu, and Q. Yu, "Security threat analyses and attack models for approximate computing systems: From hardware and micro-architecture perspectives," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, no. 4, apr 2021.

[3] W. Liu, C. Gu, M. O'Neill, G. Qu, P. Montuschi, and F. Lombardi, "Security in approximate computing and approximate computing for security: Challenges and opportunities," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2214–2231, 2020.

[4] F. Regazzoni, C. Alippi, and I. Polian, "Security: The dark side of approximate computing?" in *Proc. ICCAD*, 2018, pp. 1–6.

[5] Y. Wang, Q. Xu, G. Qu, and J. Dong, "Information hiding behind approximate computation," in *Proc. GLSVLSI*, 2019, p. 405–410.

[6] Y. Wang, J. Dong, Q. Xu, Z. Lu, and G. Qu, "Is it approximate computing or malicious computing?" in *Proc. GLSVLSI*, 2020, p. 333–338.

[7] P. Yellu, M. R. Monjur, T. Kammerer, D. Xu, and Q. Yu, "Security threats and countermeasures for approximate arithmetic computing," in *Proc. ASP-DAC*, 2020, pp. 259–264.

[8] P. Yellu and Q. Yu, "Can we securely use approximate computing?" in *Proc. ISCAS*, 2020, pp. 1–5.

[9] Y. Dou, S. Yu, C. Gu, M. O'Neill, C. Wang, and W. Liu, "Security analysis of hardware trojans on approximate circuits," in *Proc. GLSVLSI*, 2020, p. 315–320.

[10] P. Yellu, N. Boskov, M. A. Kinsy, and Q. Yu, "Security threats in approximate computing systems," in *Proc. GLSVLSI*, 2019, p. 387–392.

[11] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb 2016.

[12] M. Sparsh, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, Mar. 2016.

[13] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited: Cross-layer approximate computing: From logic to architectures," in *Proc. DAC*, 2016, pp. 1–6.

[14] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *TCAD'13*, vol. 32, no. 1, pp. 124–137, Jan 2013.

[15] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *TVLSI*, vol. 18, no. 8, pp. 1225–1229, 2010.

[16] S. Liu, K. Pattabiraman, T. Moscibroda, and B. Zorn, "Flikker: Saving dram refresh-power through critical data partitionin," *ACM SIGPLAN Notices*, vol. 47, pp. 213–224, 04 2011.

[17] I. Bhati, Z. Chishti, S. Lu, and B. Jacob, "Flexible auto-refresh: Enabling scalable and energy-efficient dram refresh reductions," in *Proc. ISCA*, June 2015, pp. 235–246.

[18] C. B. Kushwah and S. K. Vishvakarma, "A sub-threshold eight transistor (8t) sram cell design for stability improvement," in *Proc. ICICDT*, May 2014, pp. 1–4.

[19] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate storage in solid-state memories," in *Proc. 2013 MICRO*, Dec 2013, pp. 25–36.

[20] F. Frustaci, D. Blaauw, D. Sylvester, and M. Alioto, "Better-than-voltage scaling energy reduction in approximate srams via bit dropping and bit reuse," in *Proc. PATMOS*, Sep. 2015, pp. 132–139.

[21] E. Hadi, S. Adrian, C. Luis, and B. Doug, "Architecture support for disciplined approximate programming," *SIGARCH Comput. Archit. News*, vol. 40, no. 1, p. 301–312, Mar. 2012.

[22] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," *SIGPLAN Not.*, vol. 46, no. 6, p. 164–174, Jun. 2011.

[23] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proc. MICRO*, 2012, pp. 449–460.

[24] A. Rahimi, L. Benini, and R. K. Gupta, "Spatial memoization: Concurrent instruction reuse to correct timing errors in simd architectures," *IEEE TCSII: Express Briefs*, vol. 60, no. 12, pp. 847–851, 2013.

[25] D. Palomino, M. Shafique, A. Susin, and J. Henkel, "Thermal optimization using adaptive approximate computing for video coding," in *Proc. DATE*, March 2016, pp. 1207–1212.

[26] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. C. Rinard, "Using code perforation to improve performance, reduce energy consumption, and respond to failures," 2009.

[27] J. Mengte, A. Raghunathan, S. Chakradhar, and S. Byna, "Exploiting the forgiving nature of applications for scalable parallel execution," in *Proc. IPDPS*, April 2010, pp. 1–12.

[28] D. Ma, R. Thapa, X. Wang, X. Jiao, and C. Hao, "Workload-aware approximate computing configuration," in *Proc. DATE*, 2021, pp. 920–925.

[29] W. Liu, C. Gu, G. Qu, and M. O'Neill, *Approximate Computing and Its Application to Hardware Security*. Cham: Springer International Publishing, 2018, pp. 43–67.

[30] F. Regazzoni and I. Polian, "Side channel attacks vs approximate computing," in *Proc. GLSVLSI*, 2020, p. 321–326.

[31] P. Yellu, L. Buell, D. Xu, and Q. Yu, "Blurring boundaries: A new way to secure approximate computing systems," in *Proc. GLSVLSI*, 2020, p. 327–332.

[32] V. Akhlaghi, S. Gao, and R. K. Gupta, "Lemax: Learning-based energy consumption minimization in approximate computing with quality guarantee," in *Proc. DAC*, 2018, pp. 1–6.

[33] Z. Kedem. *et al.*, "Optimizing energy to minimize errors in dataflow graphs using approximate adders," in *Proc. CASES*, 2010, p. 177–186.

[34] "Instruction tables," https://www.agner.org/optimize/instruction_tables.pdf, accessed: 2021-12-1.

[35] K.-J. Lee *et al.*, "A 90 nm 1.8 v 512 mb diode-switch pram with 266 mb/s read throughput," *IEEE JSSC*, vol. 43, no. 1, pp. 150–162, 2008.

[36] M. T. Teimoori, M. A. Hanif, A. Ejlali, and M. Shafique, "Adam: Adaptive approximation management for the non-volatile memory hierarchies," in *Proc. DATE*, 2018, pp. 785–790.

[37] Y. Fang, H. Li, and X. Li, "Softpcm: Enhancing energy efficiency and lifetime of phase change memory in video applications via approximate write," in *proc. IEEE 21st ATS*, 2012, pp. 131–136.

Pruthvy Yellu is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of New Hampshire. Her current research focuses on hardware Trojan detection, design obfuscation, and approximate computing.

Qiaoyan Yu (S'03-M'11-SM'17) received the Ph.D. degree in Electrical Engineering from University of Rochester, USA in 2011. Dr. Yu is currently a Professor with the Department of Electrical and Computer Engineering, University of New Hampshire. Her research interests include hardware security and trust, embedded system security, error control for networks-on-chip. Dr. Yu is the recipient of National Science Foundation CAREER award in 2017. She has served on the technical program committees of HOST, DAC, FDTC, Asian HOST, ISVLSI, DFT, ASP-DAC, GLSVLSI, and ISCAS.