Approximate Computing and Its Application to Hardware Security

Weiqiang Liu, Chongyan Gu, Gang Qu, and Máire O'Neill

Abstract The demand for high speed and low power in nanoscale integrated circuits (ICs) for many applications, such as image and multimedia data processing, artificial intelligence, and machine learning, where results of the highest accuracy may not be needed, has motivated the development of approximate computing. Approximate circuits, in particular approximate arithmetic units, have been studied extensively and made significant impact on the power performance of such systems. The first goal of this chapter is to review both the existing approximate arithmetic circuitries, which include adders, multipliers, and dividers, and popular approximate algorithms. The second goal of this chapter is to explore broader applications of approximate computing. As an example, we review two case studies, one on a lightweight device authentication scheme based on erroneous adders and the other one on information hiding behind a newly proposed approximate data format. This approach of applying approximate computing in security is interesting and promising in the Internet of things (IoT) domain where the devices are extremely resource constrained and cannot afford conventional cryptographic solutions to provide data security and user privacy. We also discuss the potential of approximate computing in building hardware security primitives for cyber physical system (CPS) and IoT devices.

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

C. Gu · M. O'Neill
Centre for Secure Information Technologies (CSIT), Institute of Electronics, Communications & Information Technology (ECIT), Queen's University Belfast (QUB), Belfast, UK
e-mail: cgu01@qub.ac.uk; m.oneill@ecit.qub.ac.uk

G. Qu
Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD, USA
e-mail: gangqu@umd.edu

Ç. K. Koç (ed.), *Cyber-Physical Systems Security*,
https://doi.org/10.1007/978-3-319-98935-8_3

1 Introduction

The performance of various computing systems, from sensors, smartphones, and other mobile devices to servers, supercomputers, and cloud computing data centers, has been increasing dramatically in the past several decades in line with the advances in IC design according to the famous Moore's Law. However, as Moore's Law is approaching its limit [34], the conventional techniques are unable to further improve the computing performance of systems with limited power budget, i.e., the power consumption restricts the performance of computing systems. It becomes challenging to continue improving system performance by conventional CMOS technologies. One of the major concerns is the increasing on-chip power density and the power consumption requirements by the application. Chip designs at the nanoscale urgently require new approaches and paradigms to reduce low-power and high-performance computing systems.

Dynamically adjusting the supply voltage and clock frequency is one of the most effective low-power design methods [32]. However, as we push the supply voltage closer and closer to the threshold voltage, the circuit delay increases and may malfunction [19]. This coupled with the high integration density makes it very challenging to test and verify the design. Indeed, due to the lower-power supply voltage and the higher integration density at the nanoscale of a circuit design, ensuring fully correct computation results from ICs will result in a dramatic increase in cost. The International Technology Roadmap for Semiconductors (ITRS) states that the cost of manufacturing verification and testing can be greatly reduced by tolerating errors for devices [39]. Therefore, without affecting the usage and perception, acceptable reduction of the computing accuracy can effectively reduce both the power consumption and test/verification cost.

Due to the error-resilient and fault-tolerant ability of the human brain, visual and auditory systems, certain level of processing errors will not affect the quality of human perception and recognition of the processed data [14, 59]. Examples have been reported in artificial intelligence (AI), machine learning, data mining, multimedia signal processing [14, 35, 36, 59] etc. In these applications, the data includes noisy or redundant information, and therefore it makes little sense to compute the precise result based on erroneous data or perform redundant computation.

Motivated by the above challenges, approximate computing (also known as inexact computing) has attracted significant attention from both academia and industry in recent years [25, 41, 80]. Approximate computing can reduce power consumption and improve system performance by introducing acceptable errors. Therefore, we can introduce computation accuracy as a third design metrics in addition to delay and power consumption as shown in Fig. 1. It depicts a three-dimension (3D) design space by taking into account the computational accuracy, performance, and power consumption of approximate computing circuits.

Not surprisingly, some of the early research results have also made their impact on industry. Google's deep learning (DL) chip, the tensor processing unit (TPU), achieves a significant improvement in processing performance by using

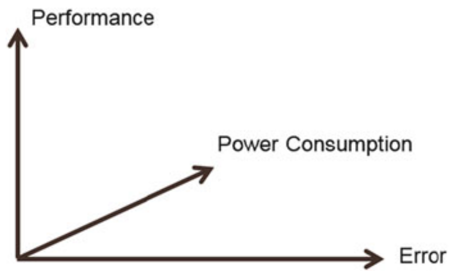


Fig. 1 A 3D design relationship of performance, power, and accuracy for approximate computing

approximate computing techniques [42]. The performance of TPU outperforms over traditional GPU and CPU processors by 15–30 times. It is a crucial component in AlphaGo which has defeated human Go champion. As another example, with the support of the Defense Advanced Research Projects Agency (DARPA), Bates developed an approximate computing chip based on an approximate arithmetic unit and founded a company known as Singular Computing [72]. This chip is used in DARPA's UPSIDE project to enable real-time video target tracking on drones. Compared to traditional processors, it can increase the speed of video processing by 100 times and consumes less than 2% of a traditional processor power by using a Singular Computing chip. Finally, we mention that both IBM [8] and ARM [65] have investigated heavily on approximate computing. This evidence shows that approximate computing is already making significant impact on the design of today's application-specific processors, and it will have higher potential in the design for future systems.

Speaking of future systems, the emerging IoT are perhaps the one that will have the most influence on our lives. The IoT era has already arrived with billions of electronics devices surrounding us, and it is predicted that there will be more than 50 billion connected IoT devices by 2020 [62]. They will have a large impact on a wide range of markets, from wearable health-care devices to embedded systems in smart cars, many of which will be underpinned by devices which are limited with regard to computation and power consumption. This has led to a high demand for cryptographic devices that can provide authentication to protect user privacy and data security. Conventional cryptographic approaches, which involve complex cryptographic algorithms, are unsuitable to be implemented on IoT devices as they incur significant timing, energy, and area overhead [66]. This opens the opportunity for developing low-cost lightweight security primitives based on approximate computing. For example, information could be hidden into the process and results of the approximate computing to protect design intellectual protection (IP) as watermark, fingerprint, or lightweight encryption [19].

Approximate computing has also been used to implement deep neural network (DNN) algorithms which have found applications in solving hardware security problems such as side-channel analysis (SCA)-based attacks [20], attacks on physical unclonable function (PUF) [38], Hardware Trojan (HT) detection [28], etc. Hence, an approximate DNN design could benefit and revolutionize hardware security-related applications.

Previously, there are several excellent surveys on approximate computing. Jiang et al. [41] reviewed and classified current designs of approximate arithmetic circuits. A complete survey of existing approximate computing work is presented in [80]. Unlike this work, we focus our discussion on the implementation of approximate arithmetic circuits and their applications in cybersecurity. Specifically, this chapter contributes in the following ways:

- A detailed classification and review of current approximate circuits, in particular approximate arithmetic circuits, including adders, multipliers, and dividers are introduced.
- Current approximate error-tolerant algorithms are briefly reviewed, and their applications are discussed.
- Two case studies demonstrating lightweight authentication and security primitives using approximate computing are presented.
- Future works on applying approximate computing into different cyber-security scenarios, including SCA techniques, PUFs, and logic obfuscation techniques, are also discussed.

2 Approximate Circuit

Arithmetic units including adders, multipliers, and dividers play important roles in processors, which significantly influence the performance and the power consumption of the whole computing system. It is expected to achieve higher speed and power efficiency as well as error tolerance for cognitive applications, e.g., recognition, data analysis, and computer vision. These motivated the fast development of approximate arithmetic designs. The design of approximate computing circuits mainly uses voltage-based probability CMOS techniques and logic reduction and pruning methods. Probability CMOS technique reduces energy consumption by allocating higher supply voltages to important areas to ensure the accuracy of most significant bits (MSBs) while appropriately reducing the supply voltage of least significant bits (LSBs) that have a less effect on the result. Cheemalavagu et al. [9] proposed a probabilistic adder that uses a conventional precision adder structure by providing various supply voltages for different bits depending on the degree of importance. However, this technique requires a higher implementation cost and generates uncontrollable errors, which restrict its subsequent applications. Therefore, most of the approximate computing circuits are based on the logic reduction and pruning methods. In cognitive computing applications, e.g., image recognition, machine learning, and pattern recognition, the key arithmetic units mainly include adders and multipliers. Therefore, high-performance and low-power adders and multipliers have been extensively studied.

Table 1 An overview of approximate adder circuits

Type			Previous works
Speculative adders	Non-segmented		SSA [54], ACA [78], SHCA [18]
	Segmented	Non-MUX	ESA [60], ETAII [88], ACAA [43], GeArA [71]
		MUX	SCSA [17], ACSA [44], GDA [83], CCBA [6]
Transistor-based approximate full adders			LOA [57], AMAs [24], AXAs [81], InXAs [2]

2.1 Approximate Adders

An overview and classification of current approximate adders are listed in Table 1. The concept of an approximate adder was first proposed for asynchronous adders [63], while the first synchronous speculative adder was proposed by Intel [54]. It has been found that full adders have a shorter carry propagation length for random operands than the length of a full carry chain. Hence, it gets faster and more energy-efficient adders by designing shorter carry chains using some specific bits. Similar as this idea, the researchers designed a family of speculative approximate adders, including non-segmented speculative approximate adders and segmented speculative approximate adders.

The non-segmented speculative approximate adder includes synchronous speculative adder (SSA) [54], almost correct adder (ACA) [78], speculative Han-Carlson adder (SHCA) [18], etc. The segmented approximate adder is a type of speculative approximate adder. The main difference is that the segmented adder divides the adder into several sub-adders and the carry propagation is computed in parallel in each sub-adder. Based on whether they have a multiplexer (MUX) or not, the segmented approximate adder can be divided into two categories, MUX-based segmented approximate adder and non-MUX-based segmented adder. The non-MUX-based segmented approximate adder includes equal segmentation adder (ESA) [60], error tolerant adder type II (ETAII) [88], accuracy configurable approximate adder (ACAA) [43], and generalized accuracy configurable approximate adder (GeArA) [71]. The MUX-based segmented approximate adder is mainly based on a carry skip or carry-select adder, including speculative carry select adder (SCSA) [17], approximate carry skip adder (ACSA) [44], gracefully-degrading adder (GDA) [83], and carry cut-back adder (CCBA) [6].

The speculative approximate adder is primarily targeted at increasing the speed and performance, while the transistor-based approximate full adder can significantly reduce power consumption. By reducing the number of transistors and basic gates from the exact full adder, an energy-efficient approximate full adder can be achieved. The first approximate full adder is a bio-inspired LOA [57], in which the MSB is implemented by approximate full adders and the LSB uses OR gates. An AND gate is used for carry propagation and the critical path delay is determined by the MSBs, which consumes very little power due to its simple structure. Gupta et al. [24] proposed five approximate mirror adders (AMAs) based on the traditional

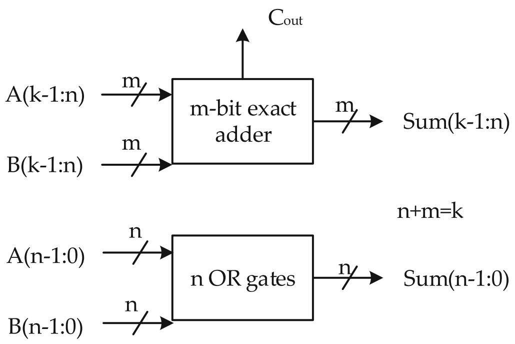


Fig. 2 The revised LOA adder structure

mirror adder. The approximate full adder also includes approximate XOR-/NXOR-based full adders (AXAs) [81] and Inexact Adder cells (InXAs) [2].

The research in [41] shows that SCSA and ACA adders present better accuracy, while ESA has the lowest accuracy and LOA exhibits medium accuracy. In terms of hardware performance, SCSA has higher power consumption. The speed of speculative approximate adder is faster; however it consumes more power. Although the speed of approximate full adder is slower, it demonstrates low power consumption and consumes less hardware resources.

The LOA design is chosen in this chapter as an example to illustrate the approximate adder. For an approximate floating-point adder, a revised LOA adder is used, as it significantly reduces the critical path by ignoring the lower carry bits [51]. A k-bit LOA consists of two parts as shown in Fig. 2, an m-bit exact adder and an n-bit inexact adder. The m-bit adder is used for the m MSBs of the sum, while the n-bit adder consists of OR gates to compute the addition of n LSBs, i.e., the lower n-bit adder is an array of n 2-input OR gates. In the original LOA design, an additional AND gate is used for generating the most significant carry bit of the n-bit adder; all carry bits in the n-bit inexact adder are ignored to further reduce the critical path.

2.2 *Approximate Multipliers*

The approximate multipliers shown in Table 2 can be classified based on the approximate design of different components. The idea of approximating operands, known as logarithmic multiplier (LM), has been proposed by Mitchell in the 1960s [58]. The LM transforms multiplication operation into additions in the logarithm domain to achieve low power consumption. However, its accuracy is low. An approximate logarithmic multiplier (ALM) and an iterative approximation logarithmic multiplier (IALM) have been proposed in [53]. Compared to the traditional LM, ALM achieves higher accuracy and lower power consumption by introducing

Table 2 An overview of approximate multiplier circuits

Type of approximate multipliers		Previous works
Approximate operand	Logarithmic	LM [58], ALM [53], IALM [53]
	Non-logarithmic	ETM [48], DRUM [29]
Approximate partial product generation	Non-booth encoding	UDM [46]
	Booth encoding	R4ABMs [52], R8ABMs [40]
Approximate partial product tree	Truncated	BAM [57]
	Untruncated	PPPM [86], R4ABMs [52]
Approximate counters or compressors	Normal binary	ANBCs [61]
	Redundant binary	ARBCs [7]

an approximate mantissa adder. IALM significantly improves the performance of the LM by introducing an iterative mechanism; however, its power consumption is relatively higher. Recently, the design of approximate multipliers based on the dynamic scaling of operands has been proposed, including fault tolerant multipliers (ETM) [48] and dynamic range multipliers (DRUM) [29]. They have very low power consumption; however, their accuracy is also lower than others [53].

The state-of-the-art high-performance multipliers normally include three parts: partial product generation, partial product accumulation, and final addition. Much research has been conducted on the approximate design of each part. Kulkarni et al. [46] proposed an approximate 2×2 multiplier, which can be used to construct larger sized underdesigned multipliers (UDMs). Approximate Booth multipliers, a radix-4 approximate Booth multiplier (R4ABM) and a radix-8 approximate Booth multiplier (R8ABM), based on approximate radix-4 modified Booth encoding (MBE) algorithms and a regular partial product array that employs an approximate Wallace tree, have been proposed in [52] and [40]. The R4ABM multiplier with an approximate factor of 14 is the most efficient design when considering both power-delay product and the error metric. Traditional Booth multipliers, e.g., broken-array multiplier (BAM) [57], truncate partial product compression trees; however, this design has a lower accuracy. Zervakis et al. [86] proposed a partial product perforation (PPP) technique that reduces the number of partial products.

The approximate radix-4 Booth multiplier is further illustrated as an example in this chapter to show the design of approximate multipliers. A Booth multiplier consists of three parts: partial product generation using a Booth encoder, partial product accumulation using compressors, and final product generation using a fast adder.

The Booth encoder plays an important role in the Booth multiplier, which reduces the number of partial product rows by half. Consider the multiplication of two N-bit integers, i.e., a multiplicand $\mathbf{A}$ and a multiplier $\mathbf{B}$ in two's complement, which is given as follows:

$$A = -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i \tag{1}$$

$$B = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i \tag{2}$$

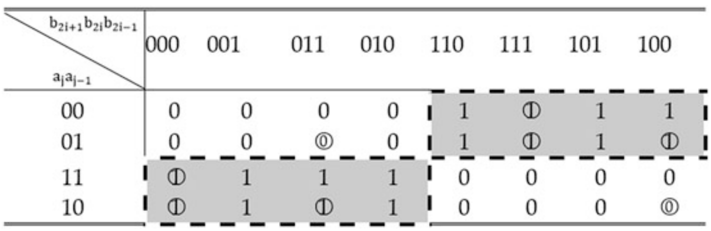
In a Booth encoder, each group is decoded by selecting the partial products as $-2\mathbf{A}$, $-\mathbf{A}$, 0, $\mathbf{A}$, or $2\mathbf{A}$. The negation operation is performed by inverting each bit of $\mathbf{A}$ and adding a "1" (defined as Neg) to the LSB [45, 84].

The circuit diagrams of the radix-4 Booth encoder and decoder are provided in [84]. The output, i.e., the partial product pp_{ij}, of the Booth encoder is given as follows:

$$pp_{ij} = (b_{2i} \bigoplus b_{2i-1})(b_{2i} \bigoplus a_j) + \overline{(b_{2i} \bigoplus b_{2i-1})}(b_{2i+1} \bigoplus b_{2i})(b_{2i+1} \bigoplus a_{j-1}) \tag{3}$$

The first R4ABM, which uses radix-4 approximate Booth encoding-2 (R4ABE2) and the regular approximate partial product array, has been proposed in [52]. The truth table of the R4ABE2 method is shown in Fig. 3, where ① denotes a "0" entry that has been replaced by a "1"; eight entries in the K-map are modified to simplify the logic of the Booth encoding. The strategy for R4ABE2 is that in addition to having a symmetric truth table with a small error, the number of prime implicants (identified by rectangle) should be as small as possible.

The gate-level circuit of R4ABE2 is shown in Fig. 4. R4ABE2 only requires one XOR-2 gate by using transmission gates, so the transistor count of R4ABE2 is 4.



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

Fig. 3 K-map of R4ABE2

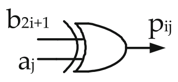


Fig. 4 The gate-level circuit of R4ABE2

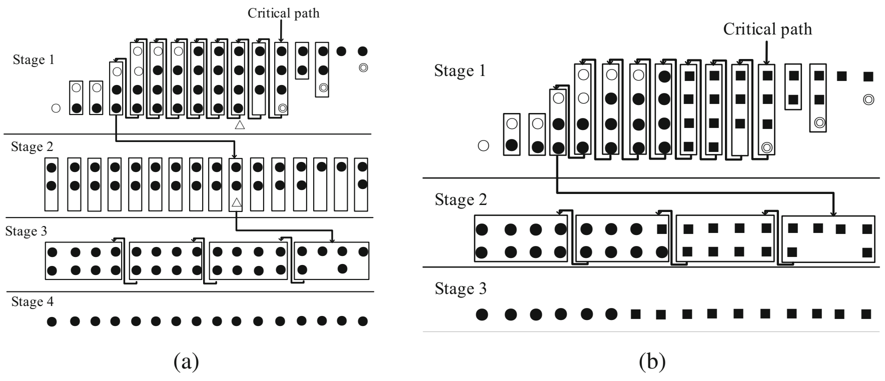


Fig. 5 The 8 × 8 Booth multiplier: (**a**) Exact irregular partial product array, (**b**) Approximate regular partial product array by ignoring the Neg term in the fifth partial product row. The exact partial product term is represented by filled circle, while the approximate partial product term is represented by filled square. Open circle and circle within circle represent the sign extension bit and the Neg term

R4ABE2 reduces the complexity of the Booth encoder by over 88% and improves the delay by 60% compared with MBE.

For a more regular partial product array (requiring a smaller reduction stage), the Neg term in the $(N/2 + 1)$th row of the approximate design of a Booth multiplier can be ignored (shown as $\triangle$ in Fig. 5a). For an N-bit radix-4 Booth multiplier when N is a power of 2, removing the extra Neg term significantly reduces the critical path, area, and power when the 4-2 compressor is used for the partial product accumulation. In the approximate partial product array (Fig. 5b), one reduction stage is saved; this significantly reduces the complexity and critical path delay. The error rate of the approximate partial product array with the ignored Neg bit is 37.5%, and its logic function is given as follows:

$$\mathrm{Neg}_{\frac{N}{2}-1} = (b_{2N+1}\overline{b_{2N}} + b_{2N+1})\overline{b_{2N-1}} = b_{2N+1}\overline{b_{2N}b_{2N-1}} \tag{4}$$

2.3 Approximate Dividers

As mentioned above, both approximate adders and approximate multipliers have been studied quite extensively. However, the design of approximate arithmetic division has not been fully analyzed. The computation of division is different from multiplication; division is mostly a sequential process, while multiplication can be executed as a multi-operand parallel addition. Thus, when considering approximate computing for division, an approach targeting the sequential nature of division must be developed; for example, when calculating the quotient, the error introduced previously will affect the next iteration. Therefore, a proper approximate design has to mitigate error propagation.

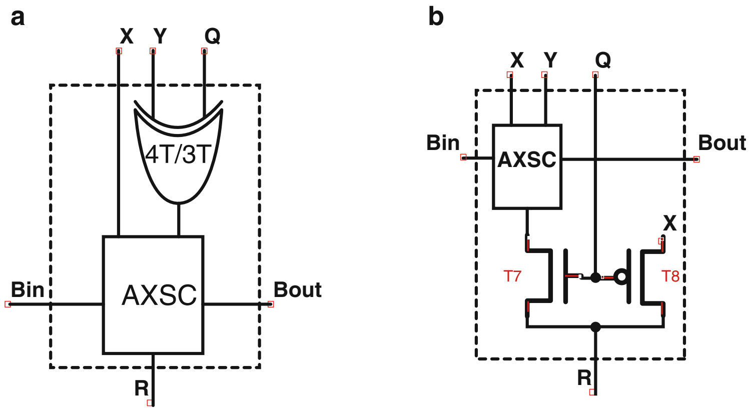
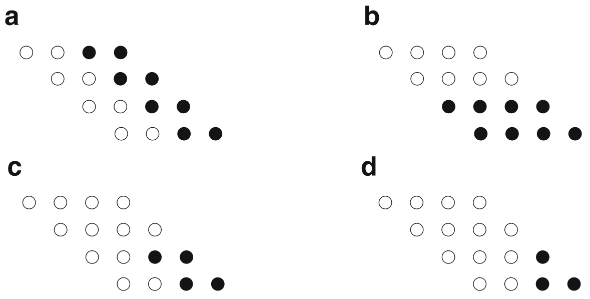


Fig. 6 Examples of restoring and non-restoring divider cells: (**a**) Non-restoring divider cells, AXDnr [64], (**b**) Restoring divider cells, AXDr [12]

Chen et al. [11] have proposed the design of an AXDnr, shown in Fig. 6a; different AXDnr designs have been proposed by replacing the logic primitives with approximate subtractors. Chen et al. [13] have proposed designs of an approximate high-radix divider, in which an approximate signed-digit adder cell is utilized to replace the exact signed-digit adder cell. A type of dynamic approximate divider has been investigated in [30], in which, for different lengths of input operands, leading-one detectors and a barrel shifter are utilized to reduce the inaccuracy. Chen et al. [11] have proposed a few inexact subtractor cells inexact subtractor cells (AXSCs) at transistor level for the design of an AXDnr. As different types of divider, restoring and non-restoring dividers have been analyzed for approximate computing; [12] has shown that an AXDr has better performance than AXDnr with respect to power consumption while also introducing a small degradation in accuracy.

The AXDr is shown in Fig. 6b. A non-restoring divider needs a remainder correction circuit for adjusting the sign of the remainder to be consistent with the dividend, thus incurring additional circuit complexity and power consumption. This can be improved by utilizing a restoring array divider [64]. As shown in Fig. 7, four

Fig. 7 Four division replacement schemes used in approximate array dividers [12]: (**a**) vertical replacement, (**b**) horizontal replacement, (**c**) square replacement, and (**d**) triangle replacement



types of replacement schemes, including vertical, horizontal, square, and triangle replacements, are used for the division operation.

3 Approximate Software/Algorithm

The main techniques used in the design of approximate algorithms include precision scaling [85], loop perforation [74], task skipping [70], and task dropping [21]. Accuracy scaling techniques reduce computational and storage requirements by varying the precision or length of the operation. Yeh et al. [85] proposed an architecture with a hierarchical floating-point unit that leverages dynamic precision reduction to enable efficient float-point unit sharing among multiple cores. This technique can gradually reduce the accuracy of the run time until the minimum accuracy of the value is reached. Tian et al. [74] proposed a precision-scaled off-chip data access technique for clustering problems to reduce energy consumption. The loop perforation technique reduces computations by skipping some iterations of the loop. An example of code without the loop perforation technique that involves skipping iterations is shown in Fig. 8 (Table 3).

The application of approximate computing, e.g., using the precision scaling technique, in DNN algorithms has already been widely studied. Since the training is more sensitive to accuracy, to reduce the cost of storage and the computational requirements, the precision scaling technique mainly focuses on the precise reduction of operands and operations, e.g., dynamic fixed-point technique [55], weight

```
// Original code without loop perforation
for ( int i = 0; i < N; i++ ) {
    // ...
}

// Modified code with skipping n iterations each time
for ( int i = 0; i < N; i++ ) {
    // ...
    i = i + skipping_factor;
}
```

Fig. 8 An example of loop perforation technique

Table 3 An overview of approximate algorithms

Approximate algorithms	Previous works
Precision scaling	[85]
Loop perforation	[74]
Task skipping/dropping	[21, 70]
Low-precision DNN	[10, 15, 16, 55, 87]
Sparsity and pruning	[1, 26]

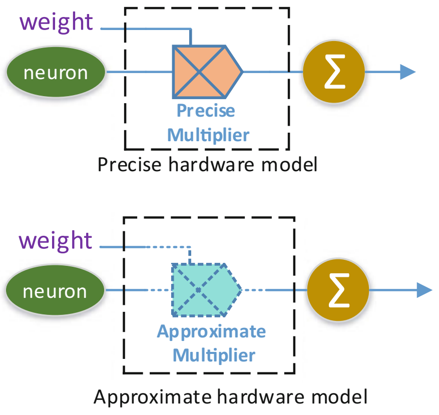


Fig. 9 The application of approximate computing to neural networks

reduction [15], activation reduction function [16], nonlinear quantization [87], and weight sharing [10]. In addition, DNNs also utilize other techniques, including the sparsity of activation functions [1] and network pruning techniques [26], to reduce computations and the size of network models.

Venkataramani et al. [77] comprehensively studies various applications for approximate computing, including image searching, recognition and detection, image segmentation, as well as data classification. Yazdanbakhsh et al. [82] presented a set of approximate computing benchmarks for different platforms. Figure 9 shows an example of the application of approximate computing to energy-efficient machine learning implementation. Since the approximate circuit could reduce the cost of storage and the computational requirements, an approximate circuit is utilized to replace the precise circuit. Then, to accelerate the computing, machine learning algorithms are involved by setting neuron and weight as parameters.

4 Approximate Computing for Hardware Security

4.1 Security Primitives Based on Approximate Computing

To minimize the power cost of IoT devices while still providing a practical security solution, Gao et al. proposed a security primitive in [19], based on basic arithmetic operations carried out by approximate function units, to embed information for authentication and other security-related applications.

4.1.1 Floating-Point Format with Embedding Security

In the work [19], it has been shown that floating-point-based approximate arithmetic computing can be employed for embedding security as shown in Fig. 10. The IEEE

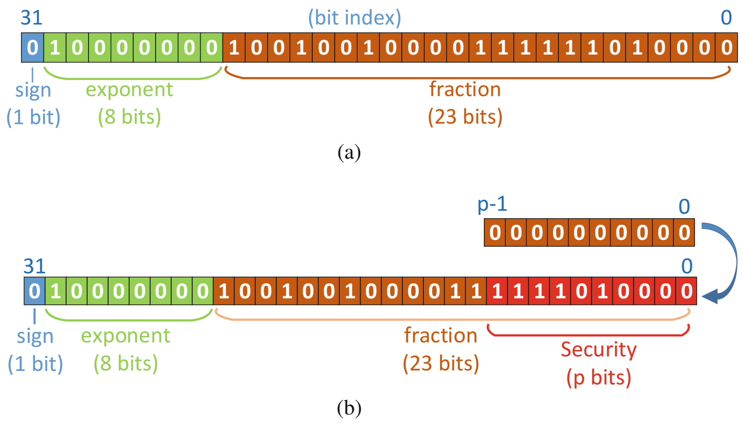


Fig. 10 The application of approximate computing to extract security: (**a**) IEEE 754 single-precision floating-point format for 32-bit data and (**b**) approximate format with security extraction. The last p LSB bits can be used as security bits to embed information

754 standard [37] specifies a binary floating-point format as having 1 sign bit, 8 exponent bits, and 23 fraction bits as shown in Fig. 10a. The sign bit determines the sign of the number, and it represents 1 or -1 if the leading bit is 0 or 1, respectively. The exponent is either an 8-bit signed integer from -128 to 127 or an 8-bit unsigned integer from 0 to 255. The significand includes 23 fraction bits to the right of the binary point.

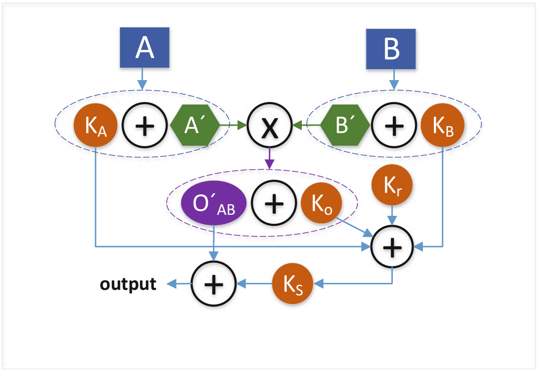
The value of IEEE 754-formatted data is computed using Eq. (5) by a given 32-bit binary data with a given biased sign, exponent e (the 8-bit unsigned integer), and a 23-bit fraction. For the example of Fig. 10a, the value is equal to 3.14159 in decimal format using Eq. (5):

$$\text{value} = (-1)^{b_{31}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{e-127} \tag{5}$$

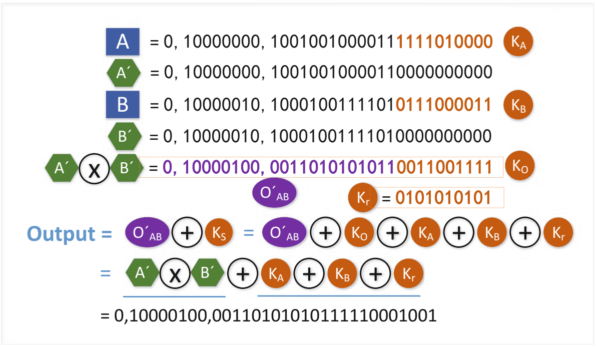
Since the LSB p bits in the fraction have little impact on the value, they can be directly used as *security* bits, as shown in Fig. 10b, to embed information without impacting the other $32-p$ bits. In this example, the approximate value is 3.1413574 by setting the last 10 bits ($p = 9$) to 0. The error introduced to the precision value is 0.0074%, which means the last p bits introduce less than 2^{p-24} error compared to the precision format.

4.1.2 Approximate Computing with Embedded Security Information

Figure 11 shows the process and an example of applying approximate computing to information hiding. Two real numbers A and B can be written as $A = A' \oplus K_A$ and



(a)


A = 0, 10000000, 10010010000111111010000 K_A
A´ = 0, 10000000, 10010010000110000000000
B = 0, 10000010, 10001001111010111000011 K_B
B´ = 0, 10000010, 10001001111010000000000
A´ X B´ = 0, 10000100, 00110101010110011001111 K_O
O´_AB
K_r = 0101010101
Output = O´_AB + K_S = O´_AB + K_O + K_A + K_B + K_r
= A´ X B´ + K_A + K_B + K_r
= 0,10000100,00110101010111110001001

(b)

Fig. 11 An example of the application of approximate computing to information embedding: (**a**) Flowchart of approximate computing with information embedding proposed by Gao et al. [19] and (**b**) an example of approximate computing with information hiding

$B = B' \oplus K_B$ using the approximate format introduced in Sect. 4.1.1, where A' and B' are the numbers A and B in approximate format that the last p bits are replaced by 0s; K_A and K_B are the last p bits of A and B. $\oplus$ is an XOR operation.

The process of executing information-embedded approximate computing proposed in [19] mainly includes the following steps. A multiplication operation of A and B, $A \times B$, is demonstrated in this example:

- Represent A and B in the approximate format: $A = A' \oplus K_A$ and $B = B' \oplus K_B$, respectively.
- Calculate and represent $A' \times B'$ in the approximate format: $A' \times B' = O'_{AB} \oplus K_O$.
- Generate $K_S = K_A \oplus K_B \oplus K_O \oplus K_r$, where K_r is a random key.
- Calculate the result $O'_{AB} \oplus K_S$ as the result of $A \times B$.

An example of the process of hiding information into approximate computing is shown in Fig. 11a. The numbers A and B are 3.14159 and 12.31, respectively. $A \times B = 3.14159 \times 12.31 = 38.6729729$ is obtained for the precise computation;

$O'_{AB} = A' \times B' = 3.1413574 \times 12.30957 = 38.6687588$ is calculated for the approximate computation with $p = 10$. The final result with security information embedded is computed as $O'_{AB} \oplus K_S = 38.6729729$, with only a 0.00448 percentage accuracy loss over the accurate result. Hence, compared to the straight approximate computing, this approach achieves approximate computing and information hiding at the same time, which can significantly reduce power and hardware resource consumption. Moreover, K_S can be used as a function of K_A, K_B, K_O, and K_r, e.g., $F(K_A, K_B, K_O, K_r)$, for the application of IP watermarking, digital fingerprinting, and lightweight encryption. For example, the IP owner's digital signature can be used as the key K_r to enable information embedding for the application of IP watermarking. Similarly, for digital fingerprinting, a unique fingerprint of each device can be utilized and embedded in the p LSBs. For the same operands of approximate computing, different key K_r values can be embedded and used to differentiate individual devices.

4.2 A Low-Voltage Approximate Computing Adder for Authentication

Due to the ubiquitous nature of IoT devices, lightweight authentication of an entity is one of the most fundamental problems in providing IoT security. A novel voltage over-scaling (VOS)-based lightweight authentication approach is presented in [3] to address this challenge. By utilizing the VOS technique, commonly employed in approximate computing to reduce the power, to exacerbate the effects of process variation and extract information related to its variation, it can be used for security purpose. Digital circuits and systems are normally operated under the nominal voltage to guarantee correct outputs. Properly reducing the operating voltage under the prescribed margin can considerably save power consumption. However, over scaling voltage can generate timing errors and thus sacrifice the output quality. The errors are related to the process variation and could be tolerated by certain applications such as image processing. Hence, a two-factor authentication scheme that uses passwords and hardware properties is proposed to achieve lightweight authentication for IoT.

The authentication protocol, shown in Fig. 12, utilizes a VOS computation unit that can generate process variation-dependent errors. The authentication protocol is divided into two stages, enrollment and authentication. For the enrollment, device i has a password $\mathbf{K}$, composed of two keys $\mathbf{K} = (k1, k2)$, and enrolled in a server's database. Moreover, the error pattern of an adder unit in device i is derived and stored in the server. For the authentication, a random string $\mathbf{R}$ is generated by the server and sent to device i. Device i calculates $\mathbf{L}$ according to the equation $\mathbf{L} = \mathbf{R} + k1$ using the adder unit and then computes $\mathbf{Y}$, where $\mathbf{Y} = \mathbf{L} \oplus k2$. $\mathbf{Y}$ is sent back to the server. The server calculates $\mathbf{L}$ and **L'**, where $\mathbf{L}' = \mathbf{M}(\mathbf{R}, k1)$. If the hamming distance of $\mathbf{L}$ and **L'** is smaller than τ, the threshold of error tolerance, the authentication succeeds. Otherwise, the authentication event aborts.

Device i		Server
	Enrollment $(1\times)$	
$\mathsf{K} = (k1, k2)$	$\longleftrightarrow$	$\mathsf{K} = (k1, k2)$
		error pattern $\mathsf{M}(\cdots)$
	Authentication $(d\times)$	
	$\xleftarrow{\mathsf{R}}$	
$\mathsf{L} = \mathsf{R} + k1$		
$\mathsf{Y} = \mathsf{L} \oplus k2 = (\mathsf{R} + k1) \oplus k2$	$\longrightarrow$	
		$\mathsf{L} = \mathsf{Y} \oplus k2$
		$\mathsf{L}' = \mathsf{M}(\mathsf{R}, k1)$
		Abort if $\mathsf{HD}(\mathsf{L}, \mathsf{L}') > \tau$

Fig. 12 The lightweight authentication protocol based on approximate computation unit [3]

5 Future Research Directions

Accelerating machine learning using approximate computing can be generally applied to side-channel attacks (SCAs), physical unclonable function (PUF) modeling attacks, and the detection of Hardware Trojans, which will be discussed in details as follows.

5.1 PUFs and SCAs

A PUF is a security primitive which utilizes the inherent process variations present during manufacturing in order to generate a unique digital fingerprint that is intrinsic to the device itself. As this natural variation between the silicon dies is out of the manufacturer's control, they are inherently difficult to clone, as well as providing additional tamper-evident properties [22]. PUFs also offer improved security as they can produce unique keys on the fly without the need for storage in non-volatile memory (NVM) on the device which reduces the risk of physical attack and saves hardware resources. These properties have a number of advantages over current state-of-the-art alternatives, opening up interesting opportunities for higher-level security protocols such as key storage and device authentication for both application specific integrated circuit (ASIC) and field programmable gate array (FPGA)-based devices.

PUF architectures can be broadly classified into Weak PUF and Strong PUF (SPUF) types as discussed in [23]. Weak PUFs have a limited challenge response pair (CRP) space and, in the extreme case, only have a single response. Therefore, they are more suited to applications such as key storage or for seeding a pseudo random number generator (PRNG), where the response never leaves the chip and is only accessed as required. In contrast, SPUFs have a large number of possible

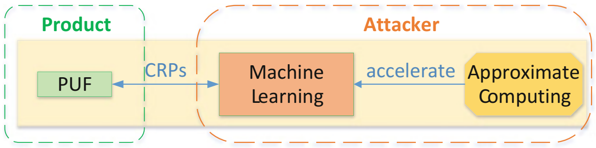


Fig. 13 An example of the application of machine learning to PUFs. The approximate computing is presented to accelerate/improve the efficiency of machine learning attacks

CRPs, whereby a large number of random challenges will return a random response unique to each challenge, as well as the physical device. By design, this implies that the requirement for a much larger entropy pool such that related challenges should not lead to related responses on the same device. Hence, SPUFs have been proposed for applications such as lightweight mutual authentication.

However, most SPUF architectures based on linear and additive functions have been shown to be vulnerable to machine learning (ML) attacks. To date, linear regression (LR), support vector machine (SVM), and evolutionary strategies (ES)-based ML methods have been widely utilized to attack PUFs [4, 5, 68, 69, 75].

In order to prevent modeling attacks, SPUF designs have been enhanced by increasing their complexity to raise the bar of attacking efforts of the adversaries. Figure 13 shows an example of the application of machine learning to SPUFs. Since approximate computing can be used to improve significantly the effectiveness of machine learning attacks, applying approximate computing-based modeling attacks to break SPUF designs could dramatically increase the attack success rate and how to mitigate this will be a more interesting and challenging problem.

5.2 SCAs

Machine learning techniques have also been used for improving SCAs attacks. A relatively new approach to profiling attacks involves the application of machine learning techniques to improve their efficiency and success. It has been shown that these attacks can be even more powerful than template attacks in practice, as less assumptions are required on the distribution of the underlying trace data [49, 56]. Much of the research to date has centered on the use of SVMs [31, 33] and random forests [50]. Research by Lerman et al. [49] showed how such approaches can be used to uncover the key of a protected (masked) advanced encryption standard (AES) implementation. A general process illustration of this idea is shown in Fig. 14. Gilmore et al. in [20] improved upon this research by investigating the novel application of a neuron network (NN)-based attack against a masked AES design. This two-stage attack first uses a NN model to recover the mask, with a second NN model built to recover the masked secret data. Combining the knowledge recovered from both attacks allows subsequent key recovery with only a single trace. Parallel work has shown how to recover the secret key with only a single model and no

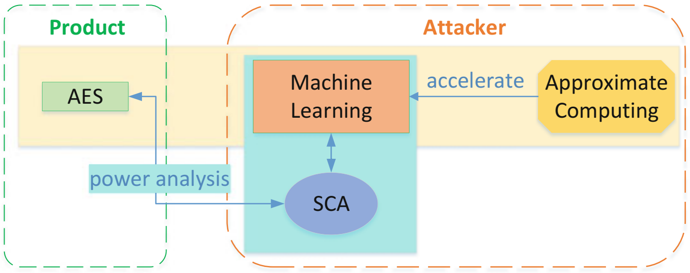


Fig. 14 An example of the application of machine learning to SCAs. The approximate computing is presented to accelerate/improve the efficiency of machine learning attacks

mask knowledge requirements at a cost of additional traces in the attack stage [56]. As shown in Fig. 14, approximate computing can be also applied for accelerating the machine learning algorithms for side channel attacks.

5.3 Hardware Trojans (HTs)

Resulting from the globalization of the semiconductor supply chain, the design and fabrication of ICs are now distributed worldwide. It brings great benefit to IC companies, which means a lower design cost and a shorter time-to-market window [47]. However, it also raises serious concern about IC trustworthiness triggered by the use of third-party vendors. As a result, it is becoming very difficult to ensure the integrity and authenticity of devices. A hardware trojan (HT) can be inserted into IC products at any untrusted phase of the IC production chain by third-party vendors or adversaries with an ulterior motive [79].

DL is a data-driven approach, where the goal is to ensure the learning algorithm is agnostic to the problem at hand; only the data changes [73]. This type of approach is often based on NN-type architectures with multiple hidden layers. With advances in training algorithms and computational power, it is now possible to train vast amounts of data leading to today's rapid advancements and adoption.

Hasegawa et al. [27] proposed a Trojan classification method for gate-level netlists using SVMs. By analyzing the netlists from the Trust-HUB benchmark suite [76], they identify several features strongly related to HTs. Trained by these features, their SVM approach results in high true positive rates, but relatively poor true negative rates when applied to the benchmark suite. Very recently, it was proposed to use DL in HT detection on gate-level netlists [27].

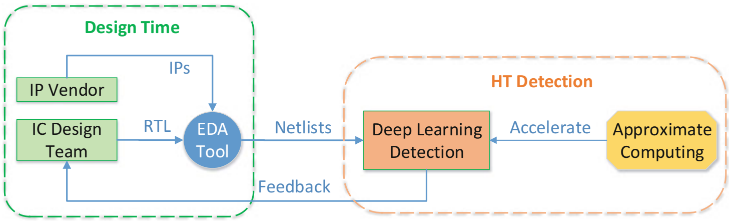


Fig. 15 The application of approximate computing to accelerate the detection of HTs

Figure 15 shows an approach using approximate computing to accelerate DL algorithms for HT detection. According to the effectiveness of the approximate circuit and algorithm development, the efficiency of the HT detection will be significantly improved.

5.4 *Approximate Arithmetic Circuit for Logic Obfuscation*

Logic obfuscation involves hiding important information, e.g., functionality and implementation, related to a circuit design by inserting additional logic components into the original design so that reverse engineering will not work without authorization. In order to execute its valid functionality to generate correct outputs, a secret key is implemented to the logic obfuscated circuit. If a wrong key is applied, the functionality will be incorrect and wrong outputs are generated by the obfuscated circuit. Logic obfuscation techniques have been utilized to protect IP and evaluate the trust of hardware [3]. However, an attacker can decipher the key by sensitizing the key values to the output or isolating the key-related gates since the logic obfuscation circuit, additionally added, can be removed from the original circuit [67].

To counter this, Fig. 16 shows a potential application of approximate arithmetic circuits in logic obfuscation. If the underlying design to be obfuscated is an approximate arithmetic circuit, logic obfuscation can be applied to the MSB or LSB

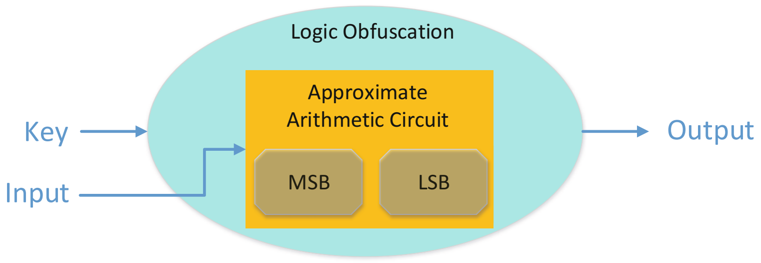


Fig. 16 A potential application of approximate arithmetic circuit to logic obfuscation

of an approximate arithmetic circuit that can only be used correctly by applying the key of the logic obfuscation circuit. Otherwise, the computation results will be too erroneous to use.

6 Conclusion

In this chapter, current approximate hardware approaches, in particular approximate arithmetic circuits, including adders, multipliers, and dividers as well as approximate software/algorithms are briefly reviewed. Two case studies, a security primitive based on approximate arithmetic circuits and a low-voltage approximate computing adder for authentication, are presented. Possible research directions for the application of approximate computing in hardware security scenarios, including SCAs, PUFs, and logic obfuscation techniques, are introduced and discussed. The goal of this chapter is to inspire future research on applying approximate computing techniques to hardware security applications.

Acknowledgements This work was partly supported by the National Natural Science Foundation of China (61871216 and 61771239), by Nature Science Foundation of Jiangsu Province (BK20151477), by Six Talent Peaks Project in Jiangsu Province (2018XYDXX-009), by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korean government (MSIT) (No. 2016-0-00399, Study on secure key hiding technology for IoT devices [KeyHAS Project]), and by the Engineering and Physical Sciences Research Council (EPSRC) (EP/N508664/-CSIT2).

References

1. J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N.E. Jerger, A. Moshovos, Cnvlutin: ineffectual-neuron-free deep neural network computing, in *Proceedings of ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA'16)* (IEEE, New York, 2016), pp. 1–13
2. H. Almurib, N. Kumar, F. Lombardi, Inexact designs for approximate low power addition by cell replacement, in *Proceedings of IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2016), pp. 660–665
3. M. Arafin, M. Gao, G. Qu, VOLtA: voltage over-scaling based lightweight authentication for IoT applications, in *Proceedings of 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)* (2017), pp. 336–341. https://doi.org/10.1109/ASPDAC.2017.7858345
4. G.T. Becker, On the pitfalls of using arbiter-PUFs as building blocks. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(8), 1295–1307 (2015)
5. G.T. Becker, The gap between promise and reality: on the insecurity of XOR arbiter PUFs, in *Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems (CHES'15)* (Springer, Berlin, 2015), pp. 535–555
6. V. Camus, J. Schlachter, C. Enz, A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision, in *Proceedings of 53rd Annual Design Automation Conference (DAC)* (2016), p. 127

7. T. Cao, W. Liu, C. Wang, X. Cui, F. Lombardi, Design of approximate redundant binary multipliers, in *Proceedings of IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (2016), pp. 31–36. https://doi.org/10.1145/2950067.2950094
8. L. Chang, Cognitive data-centric systems, in *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)* (2017), pp 1–1. http://doi.acm.org/10.1145/3060403.3060491
9. S. Cheemalavagu, P. Korkmaz, K. Palem, B. Akgul, L. Chakrapani, A probabilistic CMOS switch and its realization by exploiting noise, in *Proceeding of IFIP International Conference on VLSI* (2005), pp. 535–541
10. W. Chen, J. Wilson, S. Tyree, K. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, in *Proceeding of International Conference on Machine Learning* (2015), pp. 2285–2294
11. L. Chen, J. Han, W. Liu, F. Lombardi, Design of approximate unsigned integer non-restoring divider for inexact computing, in *Proceedings of ACM 25th Edition on Great Lakes Symposium on VLSI (GLSVLSI)* (2015), pp. 51–56. http://doi.acm.org/10.1145/2742060.2742063
12. L. Chen, J. Han, W. Liu, F. Lombardi, On the design of approximate restoring dividers for error-tolerant applications. IEEE Trans. Comput. **65**(8), 2522–2533 (2016). https://doi.org/10.1109/TC.2015.2494005
13. L. Chen, F. Lombardi, P. Montuschi, J. Han, W. Liu, Design of approximate high-radix dividers by inexact binary signed-digit addition, in *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)*, New York (2017), pp. 293–298. http://doi.acm.org/10.1145/3060403.3060404
14. V. Chippa, S. Chakradhar, K. Roy, A. Raghunathan, Analysis and characterization of inherent application resilience for approximate computing, in *Proceedings of 50th Annual Design Automation Conference (DAC)* (2013), p. 113
15. M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: training deep neural networks with binary weights during propagations, in *Advances in Neural Information Processing Systems* (2015), pp. 3123–3131
16. M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1 (2016). http://arxiv.org/abs/1602.02830. 1602.02830
17. K. Du, P. Varman, K. Mohanram, High performance reliable variable latency carry select addition, in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE, New York, 2012), pp. 1257–1262
18. D. Esposito, D.D. Caro, E. Napoli, N. Petra, A.G.M. Strollo, Variable latency speculative Han-Carlson adder. IEEE Trans. Circuits Syst. Regul. Pap. **62**(5), 1353–1361 (2015). https://doi.org/10.1109/TCSI.2015.2403036
19. M. Gao, Q. Wang, M.T. Arafin, Y. Lyu, G. Qu, Approximate computing for low power and security in the internet of things. Computer **50**(6), 27–34 (2017). https://doi.org/10.1109/MC.2017.176
20. R. Gilmore, N. Hanley, M. O'Neill, Neural network based attack on a masked implementation of AES, in *Proceedings of IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (2015), pp. 106–111. https://doi.org/10.1109/HST.2015.7140247
21. I. Goiri, R. Bianchini, S. Nagarakatte, T. Nguyen, ApproxHadoop: bringing approximations to mapreduce frameworks, in *Proceedings of ACM SIGARCH Computer Architecture News*, vol. 43 (ACM, New York, 2015), pp. 383–397
22. C. Gu, N. Hanley, M. O'Neill, Improved reliability of FPGA-based PUF identification generator design. ACM Trans. Reconfig. Technol. Syst. **10**(3), 20:1–20:23 (2017). http://doi.acm.org/10.1145/3053681
23. J. Guajardo, S.S. Kumar, G.J. Schrijen, P. Tuyls FPGA intrinsic PUFs and their use for IP protection. Vienna (2007)
24. V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, Low-power digital signal processing using approximate adders. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(1), 124–137 (2013)

25. J. Han, M. Orshansky, Approximate computing: an emerging paradigm for energy-efficient design, in *Proceedings of the 18th IEEE European Test Symposium (ETS)* (2013), pp. 1–6. https://doi.org/10.1109/ETS.2013.6569370
26. S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in *Proceedings of Advances in Neural Information Processing Systems* (2015), pp. 1135–1143
27. K. Hasegawa, M. Oya, M. Yanagisawa, N. Togawa, Hardware trojans classification for gate-level netlists based on machine learning, in *Proceedings of IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)* (2016), pp. 203–206. https://doi.org/10.1109/IOLTS.2016.7604700
28. K. Hasegawa, M. Yanagisawa, N. Togawa, Hardware trojans classification for gate-level netlists using multi-layer neural networks, in *Proceedings of IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)* (2017), pp. 227–232. https://doi.org/10.1109/IOLTS.2017.8046227
29. S. Hashemi, R. Bahar, S. Reda, Drum: a dynamic range unbiased multiplier for approximate applications, in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design* (IEEE Press, New York, 2015), pp. 418–425
30. S. Hashemi, R.I. Bahar, S. Reda, A low-power dynamic divider for approximate applications, in *Proceedings of 53rd Annual Design Automation Conference (DAC)*, New York (2016), pp. 105:1–105:6. http://doi.acm.org/10.1145/2897937.2897965
31. A. Heuser, M. Zohner, Intelligent machine homicide, in *Proceeding of International Workshop on Constructive Side-Channel Analysis and Secure Design* (Springer, New York, 2012), pp. 249–264
32. I. Hong, D. Kirovski, G. Qu, M. Potkonjak, M.B. Srivastava, Power optimization of variable-voltage core-based systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **18**(12), 1702–1714 (1999). https://doi.org/10.1109/43.811318
33. G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, J. Vandewalle, Machine learning in side-channel analysis: a first study. J. Cryptogr. Eng. **1**(4), 293 (2011). https://doi.org/10.1007/s13389-011-0023-x
34. J. Hruska, Nvidia's CEO declares Moore's law dead (2017). https://www.extremetech.com/computing/256558-nvidias-ceo-declares-moores-law-dead
35. S. Hua, G. Qu, S.S. Bhattacharyya, An energy reduction technique for multimedia application with tolerance to deadline misses, in *Proceedings of Design Automation Conference* (IEEE, New York, 2003), pp. 131–136
36. S. Hua, G. Qu, S.S. Bhattacharyya, Probabilistic design of multimedia embedded systems. ACM Trans. Embed. Comput. Syst. **6**(3) (2007). http://doi.acm.org/10.1145/1275986.1275987
37. IEEE Standard for Floating-Point Arithmetic (2008). IEEE Std 754-2008, pp. 1–70. https://doi.org/10.1109/IEEESTD.2008.4610935
38. Y. Ikezaki, Y. Nozaki, M. Yoshikawa, Deep learning attack for physical unclonable function, in *2016 IEEE 5th Global Conference on Consumer Electronics* (2016), pp. 1–2. https://doi.org/10.1109/GCCE.2016.7800478
39. ITRS 2.0 home page (Last accessed 16 January 2018). http://www.itrs2.net/
40. H. Jiang, J. Han, F. Qiao, F. Lombardi, Approximate radix-8 booth multipliers for low-power and high-performance operation. IEEE Trans. Comput. **65**(8), 2638–2644 (2016)
41. H. Jiang, C. Liu, L. Liu, F. Lombardi, J. Han, A review, classification, and comparative evaluation of approximate arithmetic circuits. Emerg. Technol. Comput. Syst. **13**(4), 60:1–60:34 (2017). http://doi.acm.org/10.1145/3094124
42. N. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, In-datacenter performance analysis of a tensor processing unit, in *Proceeding of 44th Annual International Symposium on Computer Architecture* (ACM, New York, 2017), pp. 1–12
43. A.B. Kahng, S. Kang, Accuracy-configurable adder for approximate arithmetic designs, in *Proceedings of 49th Annual Design Automation Conference (DAC)* (2012), pp. 820–825

44. Y. Kim, Y. Zhang, P. Li, An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems, in *Proceedings of the International Conference on Computer-Aided Design* (IEEE Press, New York, 2013), pp. 130–137
45. S.R. Kuang, J.P. Wang, C.Y. Guo, Modified booth multipliers with a regular partial product array. IEEE Trans. Circuits Syst. Express Briefs **56**(5), 404–408 (2009). https://doi.org/10.1109/TCSII.2009.2019334
46. P. Kulkarni, P. Gupta, M. Ercegovac, Trading accuracy for power with an underdesigned multiplier architecture, in *Proceedings of 24th IEEE International Conference on VLSI Design* (2011), pp. 346–351
47. A. Kulkarni, Y. Pino, T. Mohsenin, SVM-based real-time hardware trojan detection for many-core platform, in *Proceedings of 17th International Symposium on Quality Electronic Design (ISQED)* (2016), pp. 362–367. https://doi.org/10.1109/ISQED.2016.7479228
48. K. Kyaw, W. Goh, K. Yeo, Low-power high-speed multiplier for error-tolerant application, in *Proceedings of IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)* (IEEE, New York, 2010), pp. 1–4
49. L. Lerman, G. Bontempi, O. Markowitch, A machine learning approach against a masked aes. J. Cryptogr. Eng. **5**(2), 123–139 (2015). https://doi.org/10.1007/s13389-014-0089-3
50. L. Lerman, R. Poussier, O. Markowitch, F.X. Standaert, Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. J. Cryptogr. Eng. (2017). https://doi.org/10.1007/s13389-017-0162-9
51. W. Liu, L. Chen, C. Wang, M. O'Neill, F. Lombardi, Design and analysis of inexact floating-point adders. IEEE Trans. Comput. **65**(1), 308–314 (2016). https://doi.org/10.1109/TC.2015.2417549
52. W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, F. Lombardi, Design of approximate radix-4 booth multipliers for error-tolerant computing. IEEE Trans. Comput. **66**(8), 1435–1441 (2017). https://doi.org/10.1109/TC.2017.2672976
53. W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, F. Lombardi, Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications. IEEE Trans. Circuits Syst. Regul. Pap. **65**(9), 2856–2868 (2018)
54. S.L. Lu, Speeding up processing with approximation circuits. Computer **37**(3), 67–73 (2004)
55. Y. Ma, N. Suda, Y. Cao, J. Seo, S. Vrudhula, Scalable and modularized RTL compilation of convolutional neural networks onto FPGA, in *Proceedings of 26th International Conference on Field Programmable Logic and Applications (FPL)* (IEEE, New York, 2016), pp. 1–8
56. H. Maghrebi, T. Portigliatti, E. Prouff, Breaking cryptographic implementations using deep learning techniques, in *Proceedings of International Conference on Security, Privacy, and Applied Cryptography Engineering* (Springer, Berlin, 2016), pp. 3–26
57. H. Mahdiani, A. Ahmadi, S. Fakhraie, C. Lucas, Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. IEEE Trans. Circuits Syst. Regul. Pap. **57**(4), 850–862 (2010)
58. J. Mitchell, Computer multiplication and division using binary logarithms. IRE Trans. Electron. Comput. **EC-11**(4), 512–517 (1962)
59. D. Modha, R. Ananthanarayanan, S. Esser, A. Ndirango, A. Sherbondy, R. Singh, Cognitive computing. Commun. ACM **54**(8), 62–71 (2011)
60. D. Mohapatra, V. Chippa, A. Raghunathan, K. Roy, Design of voltage-scalable meta-functions for approximate computing, in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE, New York, 2011), pp. 1–6
61. A. Momeni, J. Han, P. Montuschi, F. Lombardi, Design and analysis of approximate compressors for multiplication. IEEE Trans. Comput. **64**(4), 984–994 (2015)
62. A. Nordrum, Popular internet of things forecast of 50 billion devices by 2020 is outdated (2016). https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-/billion-devices-by-2020-is-outdated
63. S. Nowick, Design of a low-latency asynchronous adder using speculative completion. IEEE Proc. Comput. Digital Technol. **143**(5), 301–307 (1996)

64. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs* (Oxford University Press, Oxford, 2000)
65. C. Peter, ARM's lead engineer discusses inexact processing EE Times (2013). https://www.eetimes.com/author.asp?section_id=36&doc_id=1318829
66. G. Qu, L. Yuan, Design things for the internet of things: an eda perspective, in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2014), pp. 411–416. https://doi.org/10.1109/ICCAD.2014.7001384
67. J. Rajendran, Y. Pino, O. Sinanoglu, R. Karri, Security analysis of logic obfuscation, in *Proceedings of Design Automation Conference (DAC)* (2012), pp. 83–89. https://doi.org/10.1145/2228360.2228377
68. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber, Modeling attacks on physical unclonable functions, in *Proceedings of 17th ACM Conference on Computer and Communications Security(CCS'10)*, Chicago (2010), pp. 237–249
69. U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas, PUF modeling attacks on simulated and silicon data. IEEE Trans. Inf. Forensics Secur. **8**(11), 1876–1891 (2013)
70. M. Samadi, S. Mahlke, CPU-GPU collaboration for output quality monitoring, in *Proceedings of 1st Workshop on Approximate Computing Across the System Stack* (2014), pp. 1–3
71. M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, A low latency generic accuracy configurable adder, in *Proceedings of 52nd Design Automation Conference (DAC)* (2015), pp 1–6. https://doi.org/10.1145/2744769.2744778
72. T. Simonite, (Last accessed 12 January 2018) This chip is hardwired to make mistakes but could help computers understand the world. https://www.technologyreview.com/s/601263/why-a-chip-thats-bad-at-math-can-help-computers/-tackle-harder-problems/
73. V. Sze, Y. Chen, T. Yang, J. Emer, Efficient processing of deep neural networks: a tutorial and survey. Proc. IEEE **105**(12), 2295–2329 (2017). https://doi.org/10.1109/JPROC.2017.2761740
74. Y. Tian, Q. Zhang, T. Wang, F. Yuan, Q. Xu, ApproxMA: approximate memory access for dynamic precision scaling, in *Proceedings of 25th Edition on Great Lakes Symposium on VLSI (GLSVLSI)* (2015), pp. 337–342. http://doi.acm.org/10.1145/2742060.2743759
75. J. Tobisch, G.T. Becker, On the scaling of machine learning attacks on PUFs with application to noise bifurcation, in *Proceedings of International Workshop on Radio Frequency Identification: Security and Privacy Issues* (Springer, Berlin, 2015), pp. 17–31
76. TrustHub (Last accessed 12 January 2018) Trusthub.org. http://trust-hub.org/
77. S. Venkataramani, S.T. Chakradhar, K. Roy, A. Raghunathan, Approximate computing and the quest for computing efficiency, in *Proceedings of 52nd Annual Design Automation Conference (DAC)* (2015), pp. 120:1–120:6. http://doi.acm.org/10.1145/2744769.2751163
78. A. Verma, P. Brisk, P. Ienne, Variable latency speculative addition: a new paradigm for arithmetic circuit design, in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, pp. 1250–1255 (2008)
79. X. Xie, Y. Sun, H. Chen, Y. Ding, Hardware trojans classification based on controllability and observability in gate-level netlist. IEICE Electronics Express (2017). https://doi.org/10.1587/elex.14.20170682
80. Q. Xu, T. Mytkowicz, N. Kim Approximate computing: a survey. IEEE Design Test **33**(1), 8–22 (2016). https://doi.org/10.1109/MDAT.2015.2505723
81. Z. Yang, A. Jain, J. Liang, J. Han, F. Lombardi, Approximate XOR/XNOR-based adders for inexact computing, in *Proceedings of 13th IEEE Conference on Nanotechnology (IEEE-NANO)* (IEEE, New York, 2013), pp 690–693
82. A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, P. Lotfi-Kamran AxBench: a multiplatform benchmark suite for approximate computing. IEEE Design Test **34**(2), 60–68 (2017). https://doi.org/10.1109/MDAT.2016.2630270
83. R. Ye, T. Wang, F. Yuan, R. Kumar, Q. Xu, On reconfiguration-oriented approximate adder design and its application, in *Proceedings of IEEE International Conference on Computer-Aided Design* (IEEE Press, New York, 2013), pp. 48–54
84. W.C. Yeh, C.W. Jen, High-speed booth encoded parallel multiplier design. IEEE Trans. Comput. **49**(7), 692–701 (2000). https://doi.org/10.1109/12.863039

85. T. Yeh, P. Faloutsos, M. Ercegovac, S. Patel, G. Reinman, The art of deception: adaptive precision reduction for area efficient physics acceleration, in *Proceedings of 40th Annual IEEE/ACM International Symposium on Microarchitecture* (IEEE, New York, 2007), pp. 394–406
86. G. Zervakis, K. Tsoumanis, S. Xydis, N. Axelos, K. Pekmestzi, Approximate multiplier architectures through partial product perforation: power-area tradeoffs analysis, in *Proceedings of 25th Edition on Great Lakes Symposium on VLSI (GLSVLSI)* (ACM, New York, 2015), pp. 229–232
87. A. Zhou, A. Yao, Y. Guo, L. Xu, Y. Chen, Incremental network quantization: towards lossless CNNs with low-precision weights. CoRR abs/1702.03044 (2017). http://arxiv.org/abs/1702.03044. 1702.03044
88. N. Zhu, W.L. Goh, K.S. Yeo, An enhanced low-power high-speed adder for error-tolerant application, in *Proceedings of 12th IEEE International Symposium on Integrated Circuits* (IEEE, New York, 2009), pp. 69–72