Input-Conscious Approximate Multiply-Accumulate (MAC) Unit for Energy-Efficiency

MAHMOUD MASADEH (Student Member, IEEE), OSMAN HASAN, (Senior Member, IEEE), AND SOFIÈNE TAHAR (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada
Corresponding author: Mahmoud Masadeh (m_masa@ece.concordia.ca)

ABSTRACT The Multiply-Accumulate Unit (MAC) is an integral computational component of all digital signal processing (DSP) architectures and thus has a significant impact on their speed and power dissipation. Due to an extraordinary explosion in the number of battery-powered “Internet of Things” (IoT) devices, the need for reducing the power consumption of DSP architectures has tremendously increased. Approximate computing (AxC) has been proposed as a potential solution for this problem targeting error-resilient applications. In this paper, we present a novel FPGA implementation for input-aware energy-efficient 8-bit approximate MAC (AxMAC) unit that reduces its power consumption by: performing multiplication operation approximately, or approximating the input operands then replacing multiplication by a simple shift operation. We propose an input-aware conditional block to bypass operands multiplication by (1) zero forwarding for zero-value operands, (2) judiciously approximating 43.8% of inputs into power-of-2 values, and (3) replacing the multiplication of power-of-2 operands by a simple shift operation. Experimental results show that these simplification techniques reduce delay, power and energy consumption with an acceptable quality degradation. We evaluate the effectiveness of the proposed AxMAC units on two image processing applications, i.e., image blending and filtering, and a logistic regression classification application. These applications demonstrate a negligible quality loss, with 66.6% energy reduction and 5% area overhead.

INDEX TERMS Approximate computing, approximate multiplier, approximate multiple-accumulate unit (AxMAC), input-aware approximation, image processing, FPGA.

I. INTRODUCTION
The nascent approximate computing (AxC) design paradigm is a promising approach for designing power- and area-efficient digital circuits, which are quite suitable for battery-powered devices. AxC has been used previously in loopy compression and numeric computation [1]. AxC relies on the principle of replacing traditional data processing elements by less-complex and energy-efficient ones, while compromising on the accuracy of the results. There are many applications (e.g., [2], [3]) such as machine learning, robotics, networking, multimedia processing and big-data processing, i.e., data recognition, mining and synthesis that can tolerate computation imprecision. Thus, AxC can be leveraged upon as an opportunity for designing energy-efficient error-tolerant systems suitable for error-resilient applications.

Multiply-accumulate (MAC) computation is an important and expensive operation in a variety of applications. It is extensively used in the ubiquitous operations related to digital signal processing (DSP), digital filtering, correlation, convolution, speech processing, video coding, and communication. For example, the convolution operation, which consists about 90% of MAC based computations [4], is used in various image processing related tasks, such as smoothing, sharpening and edge detection. Similarly, filtering an image of size (512 × 512) pixels, based on (3 × 3) kernel, requires 262144 multiply-accumulate operations each with 9 multiplication operations and 8 addition operations.

The ever-increasing demand for ultra low power consumption, small footprint, and high performance computing systems, necessitates designing a fast MAC unit, with low-power...
consumption, that is suitable for portable battery-powered DSP systems. For example, the CEVA-NP4000 DSP processor includes 4096 $8 \times 8$ MAC units [5], which are used in various high-performance energy-constrained edge processing applications, i.e., IoT, smart-phones and enterprise surveillance. We believe that these requirements can be attained by designing energy-efficient approximate MAC (AxMAC) units that are suitable for error-resilient applications.

MAC units with a parallel architecture are suitable for high performance applications, i.e., netpeds and laptops, however, they have a large design area and power consumption. On the other hand, MAC units with a recursive architecture are suitable for small and low-power IoT nodes, which often have a limited memory storage and operand size. Recursive MAC units, which we target in this work, are embedded within various digital systems used in our daily life. For example, CEVA-NeuPro is a dedicated low power artificial intelligence (AI) processor family for deep learning at the edge [5]. Table 1 shows a list of processors of the NeuPro family, with their corresponding design configurations and target markets. For instance, the NP4000 processor, which includes 4096 $8 \times 8$ MAC units, is ubiquitous and used in automotive and surveillance. Thus, energy-efficient MAC designs can have a significant impact on the power consumption of the overall processor.

| Product | Number of MAC Units | Target Market |
|---------|---------------------|---------------|
| NP4000  | 4096                | Automotive, Surveillance |
| NP2000  | 2048                | Surveillance, Drones |
| NP1000  | 1024                | Smart-phones |
| NP500   | 512                 | IoT, Smart-phones |

A conventional recursive MAC unit consists of four components, i.e., multiplier, adder, accumulator and controller. Therefore, any of these components, can be approximated. In this paper, we target the multiplier and controller blocks as we propose an energy-efficient input-aware unsigned 8-bit approximate MAC unit. The proposed approximate MAC is applicable to both Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA) designs. However, in this work we target FPGAs which are more suitable for run-time adaptive AxMAC unit design. The proposed design is primarily based on: (1) FPGA implementation of our previously designed approximate array multipliers [6], (2) a novel input-aware condition block, which is integrated into the controller to avoid using the multiplier by approximating input operands then performing a simple shift operation. Inspired by our previous work [6], [7], the main contributions of the paper are as follows:

- We propose an FPGA implementation for five designs of an energy-efficient unsigned 8-bit approximate MAC (AxMAC) units.
- Since approximation is application-dependent, we propose a novel input-aware conditional block to replace multiplication by a simple shift left, which is applicable to 43.8% of the operands input space.
- We demonstrate the prototype design of 8-bit approximate MAC units, and their synthesis on electronic design automation (EDA) tools to analyze their area-power-delay trade-offs. Additionally, we implement three established multipliers/MAC to compare their area, power, delay and energy with our proposed designs.
- Two real image processing applications and a logistic regression classification demonstrated the qualitative advantages of the proposed designs.

Note that the above input-aware approximation design methods are applicable to diversely scaled MAC units, regardless of the multiplier size, i.e., $8 \times 16, 16 \times 16$ or $32 \times 32$, and whether it is signed or unsigned multiplication. The rest of the paper is organized as follows: Section II provides a brief background about the structure of the conventional MAC unit and its building blocks. Section III describes related work about approximate MAC units. Section IV explains preliminaries to the proposed AxMAC units. Section V introduces the proposed approximate 8-bit MAC unit, and its building blocks, with experimental results based on the Virtex-6 FPGA. The accuracy evaluation of two image processing applications, i.e., image blending and smoothing, and logistic regression classification based on the proposed AxMAC units, are given in Section VI. Finally, Section VII concludes the paper.

**II. CONVENTIONAL MAC UNIT**

Multiply-accumulate is a standard operation that computes the product of a sequence of two numbers, i.e., $X$ and $Y$, of length $N$, and then adds that product to an accumulator ($A$), i.e., $A = \sum_{i=1}^{N} X_i \times Y_i$. The multiplicands $X$ and multiplier $Y$ are assumed to be $n$-bit wide, while the size of the adder and accumulator should be at least $2n + \log_2 N$, to accommodate the final result. In this work, as shown in Figure 1, we consider $N = 16$, and the size of the adder and the accumulator as 20-bit. The design of a MAC unit has several configurations according to the various settings of its components. Next, we describe each component in a conventional 8-bit MAC unit.

(1) **8-bit Binary Multiplier**: An $n$-bit array multiplier is composed of $n^2$ AND gates for partial products generation, and $n - 1$ $n$-bit adders for partial products accumulation. Array multipliers have periodic structures and thus lead to
a compact hardware due to short wiring which allows for efficient pipelining. This advantage makes array multipliers some of the most used in embedded System on Chip. We design an exact 8-bit MAC unit using an exact 8-bit array multiplier as the basic multiplication building block. This in turn can be used for our proposed AxMAC units while considering various approximate array multipliers as it will be explained in Section IV-C.

(2) 20-bit Binary Adder: Targeting a low-power high-speed MAC design, we design and evaluate various 20-bit binary adders, i.e., ripple carry adder (RCA), carry look ahead adder (CLA) and carry select adder (CSA), then we evaluate the performance analysis for each adder, including its area, delay and power consumption based on its RTL implementation. RCA is known for its large delay while CLA has a large area for carry prediction [8]. Without loss of generality, based on the analysis results of the 20-bit adders, we choose to adapt the carry select adder (CSA) based on 4-bit (CLA) as basic building blocks (CSA-CLA) for the MAC units design.

(3) 20-bit Accumulator: For multiplying and accumulating $N$ n-bit numbers, the accumulator size should be at least $2n + \log_2 N$ to accommodate the final result. Thus, we use a 20-bit accumulator to multiply 16 numbers each of size 8-bit.

(4) Controller: For energy efficiency, we propose a novel input-aware conditional block to approximate input operands then replace the multiplication operation with a simple shift for almost 43.8% of the inputs, as will be explained in Section V-A. The proposed conditional block is applicable to MAC units with various operands, i.e., 8 or 16-bit width and signed or unsigned polarity.

III. RELATED WORK

There is a considerable amount of literature, available on designing MAC units due to their great significance in DSP applications. For example, [9], [10] and [11] target exact designs without considering approximation for power reduction. Existing work on approximate MAC unit is very scarce compared to other functional units. For example, Dutt et al. [12] proposed an approximate radix-2 hybrid redundant MAC unit based on a redundant number system. The proposed design exploits an approximate hybrid redundant adder as the basic building block for both addition and multiplication operations in the MAC unit. However, a significant energy gain mandates approximating 40 out of the 64 bits of the results, which degrades the output quality significantly. An approximate MAC unit based on partial products compression and elimination, was proposed in [13]. The height of the partial products is reduced by using the OR gate for compression [14], [15], and the selected columns of partial products are not formed [16], [17]. Moreover, a compensation term is required to reduce the error, and improve output quality.

Several approaches for approximate computing have been reported, namely voltage overscaling (VOS), use of approximate hardware components and use of approximate memory units. These approaches can also be utilized for approximating the MAC units. VOS involves reducing the voltage applied to the hardware for energy reduction. A voltage-scalable meta-function was proposed in [18], where the supply voltage represents a quality tuning knob. However, the computations of the most significant bits are usually based on the critical paths and thus VOS is likely to lead to large errors. On the other hand, our proposed designs include approximate functional units as well as input approximation.

As an example of approximate memory units based approaches, [19] and [20] propose lookup table-based techniques. Raha and Raghunathan [19] proposed a quantized lookup table (LUT) for approximating meta-functions, which form the core computational kernels of error-resilient applications. However, for complex functions with a large number of inputs, the LUT size can become extremely large. Similarly, Alvarez [20] presented a fuzzy memoization LUT-based technique to store already computed values for specific input operands. These internal values can then be used to estimate the output for similar inputs. However, this strategy did not lead to a perceived hardware reduction.

There exists a considerable amount of efforts to design approximate hardware components. For instance, the works [21], [22] replace constant multiplications with multiple operations of shift-and-add. However, such techniques are applicable only when the operands are represented in the fixed point format and one of them is known in advance. Raha et al. [23] designed a dual-mode reconfigurable adder block, which is able to adapt the approximation degree based on the applied inputs. However, having multiple designs to meet the required quality, mandate a complex controller with a large area overhead.

Ranjan et al. [24] proposed an automatic methodology for sequential logic approximation (ASLAN), which is considered as the first effort towards the synthesis of approximate sequential circuits. Such an approach can naturally leverage upon the approximate arithmetic operators for their use within the synthesis algorithm, and thus our proposed approximate MAC units can be useful for such approximate synthesis tools.

Hashemi et al. [25] designed an approximate multiplier, called DRUM, which finds the first leading ‘1’ from the most significant position in both multiplication operands, and then prunes the size of the required multiplier to reduce the approximation error. However, this design cannot be integrated on general purpose processors as it needs to set the multiplier size for each application offline. The work in [26] proposed a configurable floating-point multiplier (CFPU), which multiplies input operands depending on the input mantissa. However, such CFPU gives poor accuracy with coarse grain tuning capability. This limits the number of applications that could benefit from approximation. The work in [27] proposed a runtime configurable floating-point multiplier (RMAC) by approximating the mantissa multiplication to a simple addition between the mantissa of input operands. Despite being runtime configurable with low
energy consumption, both designs in [26] and [27] are still unable to completely remove the need for exact multiplication from their designs.

In this paper, we propose a hybrid approach, in the sense that it aims to simplify the MAC design by: 1) utilizing an approximate multiplier, or 2) approximating the input operands to the multiplier by reducing them to smaller power-of-2 values and then using the well-known technique of shifting rather than multiplication. Thus, approximate multipliers, e.g., [19]–[27], can be easily integrated with our proposed approximate MAC units. We compare our work with two approximate MAC units based on approximate multipliers proposed by Kulkarni et al. [28] and Kyaw et al. [29] that have similar structures to our designed multipliers and their results are still competitive. Moreover, we compare our work with a third approximate MAC unit based on the approximate tree compressor multiplier (ATCM), proposed by Yang et al. [30], which is a Wallace tree multiplier. Kulkarni et al. [28] construct a large (8 × 8) multiplier—which we call 2 × 2-based multiplier—using smaller (2 × 2) approximate multipliers as building blocks. The work in [29] designs an (8 × 8) error tolerant multiplier (ETM) based on the truncation principle by dividing the multiplier into two parts, i.e., accurate and approximate. For an 8-bit multiplier, the most significant 8-bits of the result are generated based on exact multiplication while the least significant 8-bits of the result are generated based on probabilistic bit manipulation. ATCM utilizes a 4-to-2 compressor proposed by [31] which equally partitions the rows of the partial product tree array to reduce power and delay. In the rest of the paper, we call the above three MAC units: Kul-MAC, ETM-MAC and ATCM-MAC, respectively.

IV. PRELIMINARIES
In this work, we target the MAC unit with 8-bit input operands. Without loss of generality, this work can be extended to any operand size, e.g., 16 × 8 and 16 × 16. The proposed AxMAC unit, approximately computes the product of two numbers, i.e., X and Y, then accurately adds the product to the accumulator (A). The basic approximate full adders and our previously designed approximate multipliers were originally designed for ASIC (Application Specific Integrated Circuit). Thus, because of the architectural differences between ASICs and FPGAs, we reevaluated the characteristics of our basic building blocks on Virtex-6 FPGA in this paper. Next, we briefly discuss preliminaries to the proposed AxMAC units.

A. FPGA SYNTHESIS
To analyze the metrics of approximate designs, we utilize the XC6VLX75T FPGA, which belongs to the Virtex-6 family, and the FF484 package [32]. For functionality verification, we use VHDL simulation based on Mentor Graphics Modelsim [33]. We use Xilinx XPower Analyser for power calculation based on exhaustive design simulation [34]. For logic synthesis, we use the Xilinx Integrated Synthesis Environment (ISE 14.7) tool suite [35]. In FPGAs, look-up-table (LUT) are a small asynchronous SRAMs that are used to implement combinational logic circuits, while flip-flops are single-bit memory cells that are used to hold a state. Also, slices are the basic building block components containing a number of LUT’s, flip-flops, and carry logic elements of the design before mapping. Any slice that is used even partially is counted in the occupied slices in the map report. Design delay represent the maximum combinational path delay.

B. APPROXIMATE MIRROR ADDERS
Approximate computing relies on the principle of fail small or fail rare, where approximation should introduce a small error magnitude or error rate. Based on that and utilizing the mirror adder, five approximate Full Adders (FAs) are proposed in [36]. Several approximate FAs designed in [36] at register transfer level (RTL) with reduced design complexity. We use such designs as building blocks for developing low-power approximate 8-bit array multipliers [6], [7]. Table 2 shows the area, power and delay (period) of different designs of FAs at RTL.

As shown in Table 2, all designs occupy one LUT and one slice. However, AMA5 has 4 I/Os and it does not occupy any LUT or slice. The delay of AMA1 – AMA3 is similar to the exact FA delay, i.e., 755 ps where logic delay is 43 ps and routing delay is 712 ps. AMA4 design has a negligible delay reduction, i.e., 0.5%, while AMA5 exhibits 63% delay reduction due to its design simplicity where it has no logic delay and its routing delay is 279 ps. Regarding power consumption, all designs have a reduced power consumption with an average of 37.1% compared to the exact full adder. Thus, these designs clearly satisfy the goal of approximation by saving power, where power reduction ranges from 28.6% for AMA1 to 52.5% for AMA5. In the sequel of this paper, we use these five approximate FAs, i.e., AMA1 – AMA5, for designing an 8-bit approximate array multipliers at RTL.

C. APPROXIMATE 8-BIT ARRAY MULTIPLIERS
As avowed by [37], multiplier components utilize 46% of chip area in most MAC modules. Thus, an energy-efficient multiplier design can play a significance role in low-power VLSI system design. To control error significance, the higher order multiplication operations are performed accurately while only the lowest significant part of the result is approximated in the proposed approach. Thus, based on quality evaluation of various approximate multipliers with a different number of approximated bits [6], [7], here, we propose to
use approximate array multipliers with 9-bit of the result approximated out of 16-bit in the approximate MAC units as shown in Figure 3. Approximating more than 9-bits gains more area, power and delay reduction. However, the quality degradation is also quite significant.

Table 3 shows the area, power and delay of different designs of the 8-bit approximate multiplier at RTL. Clearly, all designs have a reduced area and power consumption, where the design area is represented as the number of slice LUTs and occupied slices. The design delay (maximum combinational path delay) consists of the delay of two components, i.e., logic delay and routing delay. All approximate designs have a reduced logic delay, but AMA1 – AMA3 have a slightly longer routing delay. Thus, their total design delay is more than the exact design delay.

As shown in Figure 4, the area reduction of our approximate multipliers varies from 10.3% to 60.7% with an average of 37.6%. Similarly, power reduction of the approximate multipliers ranges from 63.6% to 71.7% with an average of 68.8%, where dynamic power varies from 129.6 mW to 166.6 mW compared to the exact design, which is 457.7 mW. Due to the simplicity of the design, multipliers based on AMA4 and AMA5 have shorter critical paths with delay reduction of 32.7% and 48.1%, respectively. The energy reduction for the approximate multipliers varies between 60.5% and 85.3% with an average of 71.8%. Noticeably, designs based on AMA4 and AMA5 always exhibit more approximation benefits for all design metrics compared to others. The 2x2-based multiplier [28], ETM multiplier [29] and ATCM [30] have a power reduction of 49.3%, 68.8% and 70.3%, respectively.

It is important to note that all full adders and 8-bit approximate multipliers, given in Tables 2 and 3, are combinational designs. Thus, the proposed design can be integrated directly into the MAC unit without the need for registers. Therefore, design synthesis on FPGA adds I/O buffers to the top level design ports. Thus, the maximum combinational delay will include the I/O buffer delay, i.e., maximum delay from FPGA input pin to the FPGA output pin. Moreover, a considerable amount of dynamic power consumption is due to such added I/O buffers. For sequential designs, where registers are added to buffer the inputs and outputs, the path no longer includes any I/O elements, and the maximum path delay and dynamic power are thus reduced substantially.

Building a (8 x 8) approximate multiplier based on smaller blocks, as proposed in [28], requires 16 (2 x 2) blocks and 60 FAs for partial results accumulation. Thus, as shown in Table 3, the 2 x 2-based multiplier [28] has an area close to the area of the exact array multiplier with 49.3% power reduction. The ETM multiplier [29] relies on constructing just only a quarter of the partial products, based on the most significant 4-bits of the two operands. Thus, it exhibits a reduced area, power and delay as shown in Figure 4, with relatively large errors. ATCM [30], based on Wallace tree architecture, employs an approximate 4-to-2 compressor designed in [31]. It has a competitive power reduction.

D. ACCURACY METRICS
There are several application dependent error metrics used in approximate computing to quantify approximation errors and evaluate design accuracy [38], [39]. For example, considering an approximate arithmetic design with two inputs, i.e., \(X\) and \(Y\), of \(n\)-bit each, where the exact result is \((P)\) and the approximate result is \((P')\), these error metrics include:

- Error Rate (ER): Also called error probability, is the percentage of erroneous outputs among all outputs.
- Error Distance (ED): The arithmetic difference between the exact output and the approximate output for a given
input. ED can be given by:

\[ ED = |P' - P| \] (1)

- Mean Error Distance (MED): The average of ED values for a set of outputs obtained by applying a set of inputs. MED is an effective metric for measuring the implementation accuracy of a multiple-bit circuit design, and obtained as:

\[ MED = \frac{1}{2^n} \sum_{i=1}^{2^n} |ED_i| \] (2)

- Normalized Error Distance (NED): The normalization of MED by the maximum result that an exact design can have \((P_{max})\). NED is an invariant metric independent of the size of the circuit, therefore, it is used for comparing circuits of different sizes, and it is expressed as:

\[ NED = \frac{MED}{P_{max}} \] (3)

- Relative Error Distance (RED): The ratio of ED to the accurate output, given by:

\[ RED = \frac{ED}{P} = \frac{|P' - P|}{P} \] (4)

- Mean Relative Error Distance (MRED): The average value of all possible relative error distances (RED):

\[ MRED = \frac{1}{2^n} \sum_{i=1}^{2^n} |RED_i| \] (5)

- Mean Square Error (MSE): It is defined as the average of the squared ED values:

\[ MSE = \frac{1}{2^n} \sum_{i=1}^{2^n} |P'_i - P_i|^2 = \frac{1}{2^n} \sum_{i=1}^{2^n} |ED_i|^2 \] (6)

- Peak Signal-to-Noise Ratio (PSNR): The peak signal-to-noise ratio is a fidelity metric used to measure the quality of the output images:

\[ PSNR = 10 \cdot \log_{10} \left( \frac{255^2}{MSE} \right) \] (7)

These error metrics will be used to evaluate the accuracy of various proposed approximate MAC units.

V. PROPOSED APPROXIMATE MAC (AXMAC) UNIT

We use the five approximate array multipliers [6] and build five AxMAC units in order to explore the quality-power trade-offs. It is worth noting that the static power evaluated for the approximate multipliers is almost the same as the one for the exact designs, since it depends on \(V_{dd}\) and \(I_{dd}\) [40]. However, the dynamic power consumption, which depends on the internal capacitance \((C_L)\) of the design, and the switching factor \((\alpha)\) is a factor of the value of the input operands [40]. We propose to reduce the dynamic power consumption by avoiding multiplication and using simple shift operations.

A. CONDITIONAL OPERANDS BLOCK

The multiplier treats all operands similarly irrespective of their magnitude. However, the power consumption of the multiplier depends on the value of its operands. The source of CMOS power consumption, such as binary adders and multipliers, is primarily composed of static and dynamic power as given in Equations 8 and 9, respectively.

\[ P_s = V_{dd} \cdot I_{dd} \] (8)

\[ P_d = \alpha \cdot F \cdot C_L \cdot V_{dd} \] (9)

Static power \((P_s)\) depends on the supply voltage \((V_{dd})\) and leakage current \((I_{dd})\), while the dynamic power \((P_d)\) depends on the supply voltage \((V_{dd})\), operation frequency \((F)\) and the capacitive load \((C_L)\), which in turn depend on the structure of the underlying circuit and the switching \((\alpha)\) that depends on the operand’s values. The major portion of power consumption in the MAC unit is attributed to the dynamic power \((P_d)\). Thus, we target reducing such input-dependent dynamic power consumption through minimizing circuit switching activity \((\alpha)\).

For an n-bit array multiplier, \(n^2\) partial products are always generated then summed up irrespective of the operands value. However, based on the operands value, partial product generations and summations may be avoided in specific cases to
decrease power consumption. For that, we propose an input-aware conditional block to check the input operands for the following three cases:

1) if any of the operands is zero, then disable the multiplier and directly pass a value of zero to the adder. This case covers \(2 \times 2^n = 512\) different inputs for the 8-bit approximate multiplier.

2) if any of the operands has a magnitude \((M)\) which is a power-of-2, then the other operand is shifted left by \(\log_2 M\) positions and passed to the adder. A list of such applicable operators is shown in Table 4. It includes \(2 \times n \times 2^n = 4096\) different inputs for our approximate multipliers.

3) if any of the input operands has a magnitude \((M)\) that can be approximated to \((M')\) which is a power-of-2, then the other operand is shifted left by \(\log_2 M'\) positions and passed to the adder. Table 5 shows the proposed list of the approximable operands and their respective approximated values. The list includes \(2 \times 47 \times 2^n = 24064\) different inputs for the 8-bit approximate multiplier.

Algorithm 1 explains the work flow for the proposed conditional block. It accepts the two operands to be multiplied, i.e., \(X\) and \(Y\), and the number of times \((N)\) to multiply and accumulate. The output flag Multiply-or-Add \((MA)\), indicates whether to multiply \(X\) and \(Y\) and forward the result to the adder or not. Initially, it is reset (Line 4) to indicate that the output of the multiplier is forwarded to the adder. Whenever, any of the operands has a special value, i.e., zero, power-of-2 or approximable to power-of-2, the \(MA\) flag is set to 1 (Lines 9, 13, 17, 22 and 27) to indicate that the output of the conditional block, i.e., conditional result \((CR)\), is forwarded to the adder and the multiplier is bypassed. The conditional result is set to a suitable value (Lines 8, 12, 16, 21 and 26), based on the magnitude of the input operands.

Table 4 explains different possible cases for the operation of the input-aware conditional block embedded within an 8-bit approximate MAC unit. The conditional block checks for 9 cases for each operand of the 8-bit multiplier. \(\bar{X}\) and \(\bar{Y}\) are operand values which are not power-of-2 nor approximable to power-of-2. The shift left operation, which is equivalent to multiplication by two, is denoted by the symbol \(\langle\langle\cdot\rangle\rangle\), and \(CR\) is the conditional result to be used instead of the multiplier result.

### Table 4. Power-of-2 operands of the conditional block.

| Operand X | Operand Y | Conditional Result (CR) | Operand X | Operand Y | Conditional Result (CR) |
|-----------|-----------|-------------------------|-----------|-----------|-------------------------|
| 0         | \(\bar{Y}\) | 0                       | \(\bar{X}\) | 8         | \(X \langle\langle 3\rangle\rangle\) |
| \(\bar{X}\) | 0         | 0                       | 16        | \(\bar{Y}\) | \(Y \langle\langle 4\rangle\rangle\) |
| 1         | \(\bar{Y}\) | 32                      | \(X\)     | 32        | \(Y \langle\langle 5\rangle\rangle\) |
| \(\bar{X}\) | 2         | 32                      | \(X\)     | 32        | \(Y \langle\langle 5\rangle\rangle\) |
| 4         | \(\bar{X}\) | 64                      | \(X\)     | 64        | \(Y \langle\langle 6\rangle\rangle\) |
| 8         | \(\bar{Y}\) | \(\langle\langle 2\rangle\rangle\) | 138       | \(\bar{Y}\) | \(Y \langle\langle 7\rangle\rangle\) |
| 8         | \(\bar{Y}\) | \(\langle\langle 3\rangle\rangle\) | \(X\)     | 128       | \(Y \langle\langle 7\rangle\rangle\) |

### Table 5. List of approximable operands.

| M | 147 | 135 | 124 | 128 | 140 | 136 | 132 | 134 | 133 |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
| 2 | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
| 4 | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
| 8 | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   |

Algorithm 1 Input-Aware Energy-Efficient Conditional Block \((CR, MA) = CB(X, Y)\)

**Input:**

1. (1) \(X\): Multiplicand; (2) \(Y\): Multiplier;
2. (3) \(A\): Accumulator; (4) \(N\): Number of operands

**Output:**

3. (1) \(CR\): Conditional Result; (2) \(MA\): Multiply-or-Add
4. \(MA \leftarrow 0\)
5. **while** \(i \leq N\) **do**
6. Read \(X_i, Y_i\)
7. **if** \(X_i \equiv 0\) **or** \(Y_i \equiv 0\) **then**  // Check for zero operand
8. \(CR_i \leftarrow 0\)
9. \(MA \leftarrow 1\)
10. **end if**
11. **if** \(X_i\) is Power-of-2 **then**
12. \(CR_i \leftarrow (Y_i \langle\langle \log_2 (X_i) \rangle\rangle\)  // Shift-left operand \(Y_i\)
13. \(MA \leftarrow 1\)  // by \(\log_2 X_i\) times
14. **end if**
15. **if** \(Y_i\) is Power-of-2 **then**
16. \(CR_i \leftarrow (X_i \langle\langle \log_2 (Y_i) \rangle\rangle\)  // Shift-left operand \(X_i\)
17. \(MA \leftarrow 1\)  // by \(\log_2 Y_i\) times
18. **end if**
19. **if** \(X_i\) is Not Power-of-2 **and** \(X_i\) is approximable **then**
20. \(X'_i \leftarrow \text{Approximate}(X_i)\)
21. \(CR_i \leftarrow (Y_i \langle\langle \log_2 (X'_i) \rangle\rangle\)  // Shift-left operand \(Y_i\)
22. \(MA \leftarrow 1\)  // by \(\log_2 X'_i\) times
23. **end if**
24. **if** \(Y_i\) is Not Power-of-2 **and** \(Y_i\) is approximable **then**
25. \(Y'_i \leftarrow \text{Approximate}(Y_i)\)
26. \(CR_i \leftarrow (X_i \langle\langle \log_2 (Y'_i) \rangle\rangle\)  // Shift-left operand \(X_i\)
27. \(MA \leftarrow 1\)  // by \(\log_2 Y'_i\) times
28. **end if**
29. **end while**

Table 5 shows 47 different values of each operand, which are approximable to power-of-2, and their approximated value. For example, if the operand has a value between 124 and 135, it will be approximated to 128, then as shown in Table 4, the multiplication is replaced by shifting the other operand 7 positions to the left. Moreover, if the operand has a value which is not power-of-2 nor approximable to power-of-2, e.g., 140, the shifting process is not applicable, and the result is evaluated by approximate multiplication. For the 8-bit MAC unit, these special cases cover \(2 \times (9 + 47) \times 2^n\) out of \(2^n \times 2^n\) cases which equals 43.8% of the operands input space. Thus, the proposed conditional block would be beneficial in power saving, especially for applications that exhibit a high probability of having such operands, i.e., with
Figure 5. Histogram of Error Distance (ED) for various Approximate MAC Units.

Table 5. Approximable operands of the conditional block and their approximated value.

| Approximable Operand | Approximated Operand |
|----------------------|----------------------|
| Binary | Decimal | Approximated Value |
| 0000 0000 | 0000 0000 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0000 0001 | 0000 0001 | 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0000 0010 | 0000 0011 | 2 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0000 0100 | 0000 0101 | 4 5 4 4 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0000 0110 | 0000 0111 | 6 7 6 6 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0000 1000 | 0000 1001 | 12 19 16 16 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0001 0000 | 0001 0001 | 28 39 28 28 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0011 0000 | 0011 0001 | 80 71 64 64 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0111 1000 | 0111 1001 | 128 135 128 128 0 0 0 0 0 0 0 0 0 0 0 0 |

B. APPROXIMATE 8-BIT MAC UNIT

Algorithm 2 explains the operation flow of the proposed AxMAC unit, based on its previously explained components. Initially, the contents of the accumulator register are reset (Line 4), i.e., $A \leftarrow 0$, and it is reset again after processing the sequence of $N$ operands. For each pair of operands, $X_i$ and $Y_i$, a partial result ($PR_i$) is set to a specific value based on: 1) the conditional block evaluation result for special case operands (Line 9); or 2) the multiplier result (Line 11). Then, the partial result is accumulated (Line 13) and the final result is returned in the accumulator (Line 16), i.e., $A \approx \sum_{i=1}^{N} X_i \ast Y_i$. Figure 6 depicts an equivalent graphical representation for the RTL implementation of the AxMAC unit design including the approximate multiplier and conditional block.

1) ACCURACY ANALYSIS

To meet AxC requirements, we are mandated to analyze the error metrics of the proposed AxMAC units, as well as their area, power, and delay on a Virtex-6 FPGA. We evaluate various accuracy metrics for the proposed AxMAC designs which all have the full adders (FAs) that contribute to the 9 least significant bits (LSBs) are being approximated as shown in Figure 3. Moreover, we evaluate the accuracy metrics for

Figure 6. RTL of approximate mac unit with conditional block.
Ax3MAC has the highest MSE. All proposed AxMAC units have a MSE less than Kul–MAC and ETM–MAC. Kul–MAC has a low ER and NED, which are 0.4673 and 2.86%, respectively, with a high MSE which is $6.46 \times 10^6$. ER and NED for ETM–MAC are 0.996 and 14.29%, respectively, and MSE equals $4.14 \times 10^6$. ATCM-MAC [30] has the minimal NED, i.e., 1.57%, where its ER is 0.6028 and MSE is $2.62 \times 10^3$.

Figure 5 shows the histogram distribution for the ED of the proposed AxMAC units. The zoomed-in (top-right) figure is the histogram for Ax1MAC – Ax5MAC, because it is invisible due to ETM-MAC. The maximum ED for Ax1MAC is 2820 with average of 538. Ax2MAC with maximum of 1452 has an average of 464, while Ax3MAC with maximum of 1876 has an average of 1010. Ax4MAC and Ax5MAC have the lowest values, with average of 185 for both, and their maximum value is 796 and 756, respectively. The bottom-left corner of Figure 5 clearly shows histogram distribution for the ED of Kul–MAC, ETM-MAC and ATCM-MAC units. The maximum ED for Kul-MAC is 14450 with an average of 903, where small error values occur frequently and large error values occur rarely. The maximum ED for ETM-MAC is 7170 with an average of 1645. Moreover, the maximum ED for ATCM-MAC is 7748 with a low average of 224.

In addition to the above-mentioned accuracy evaluation of AxMAC units with different settings, we also evaluate the output quality of two image processing applications and logistic regression classifier, utilizing our approximate MAC units with 9-bit of the result being inexact, Kul–MAC, ETM–MAC and ATCM–MAC units. More details on this application-dependent quality evaluation can be found in Section VI.

2) AREA, POWER/ENERGY AND DELAY ANALYSIS

Now, we evaluate the proposed AxMAC units in terms of area, power, delay and energy. Table 7 shows the obtained design metrics including area (Slice Register, Slice LUTs and Occupied Slices), dynamic power consumption, minimum operating delay, maximum operating frequency and energy of different MAC units. The evaluated designs are:

- Conventional MAC unit, where its design characteristics are used as a baseline for evaluating other units;
- Five proposed approximate MAC units, i.e., Ax1MAC – Ax5MAC;
- Three related work, i.e., Kul–MAC, ETM–MAC and ATCM–MAC;

All above designs are with two settings, i.e., without and with the conditional block (w/o CB and w/CB). The comparison results for the different AxMAC units are shown in Figure 7, where the x-axis contains the names of the evaluated AxMAC units. The y-axis is the percentage in reduction from the conventional design. Note that larger values represent better results. The four bars (from left to right) in each design show the percentage of reduction in area, power consumption, critical path delay and design energy. As shown in Figure 7 (left side), our proposed designs without

| Type of Design | ER | MED | NED | MRED | MSE |
|----------------|----|-----|-----|------|-----|
| Ax1MAC         | 0.977 | 538 | 0.0758 | 0.1879 | $4.53 \times 10^6$ |
| Ax2MAC         | 0.996 | 464 | 0.0976 | 4.2929 | $2.97 \times 10^6$ |
| Ax3MAC         | 1.000 | 1010 | 0.2200 | 13.4248 | $1.71 \times 10^6$ |
| Ax4MAC         | 0.977 | 185 | 0.0286 | 0.0582 | $5.27 \times 10^4$ |
| Ax5MAC         | 0.952 | 185 | 0.0261 | 0.0488 | $5.34 \times 10^4$ |

As shown in Table 6, the average ER for all proposed designs is quite high, i.e., around 98%, which is expected due to approximating 9-bits of the 16-bits of the results. Thus, the error magnitude should be reduced to compensate for the high error probability. The proposed AxMAC designs have a tolerable error, e.g., NED ranges from 2.61% to 22.8%. The proposed Ax4MAC and Ax5MAC have the lowest MSE while the previously mentioned Kul–MAC [28], ETM–MAC [29] and ATCM-MAC [30] units.

Table 6 shows the accuracy metrics for these eight different AxMAC units. We evaluated the accuracy in MATLAB by implementing equivalent behavioral models and performing an exhaustive testing. Thus, the total number of test patterns was 65536.

As shown in Table 6, the average ER for all proposed designs is quite high, i.e., around 98%, which is expected due to approximating 9-bits of the 16-bits of the results. Thus, the error magnitude should be reduced to compensate for the high error probability. The proposed AxMAC designs have a tolerable error, e.g., NED ranges from 2.61% to 22.8%. The proposed Ax4MAC and Ax5MAC have the lowest MSE while the previously mentioned Kul–MAC [28], ETM–MAC [29] and ATCM–MAC [30] units.

Table 6. Error metrics for approximate MAC units with 9-bits of the result being approximated.
the conditional-block exhibit a reduced area compared to the exact one, with an average of 35.6% reduction, where it ranges from 26.2% to 41.3%. Ax4MAC has the highest area and delay reduction followed by Ax5MAC. Also, we were able to obtain an average of 13.4% power reduction, which ranges from 2.6% to 26.3%. Ax5MAC has the greatest power saving and energy reduction followed by Ax4MAC. Clearly, Ax4MAC and Ax5MAC are superior designs compared to the related work [28]–[30]. Adding the input-aware conditional block for more power saving introduces an area overhead, as shown in Figure 7 (right side), where the proposed designs have larger area with an average of 20%. Moreover, the conditional block lengthen the critical path, where the average delay reduction drops from 24.1% to 5.6%. However, the average power (energy) reduction of the approximate designs enhanced to 48.9% (51.3%) rather than 13.4% (32.9%). Comparing Figure 4 with Figure 7 (left-side), shows that the design characteristics of approximate multipliers are not shown directly on AxMAC units. This is due to the I/O buffers added to the top level ports when synthesizing the approximate multipliers on FPGA.

Figure 8 shows a comparative analysis for the best AxMAC units, i.e., Ax4MAC and Ax5MAC, compared with Kul–MAC, ETM–MAC and ATCM–MAC units. Without the conditional block, Ax4MAC and Ax5MAC designs have 41.3% and 40.1% area reduction, respectively. Kul–MAC has a large area with 17% increase over the conventional MAC unit, while ETM–MAC has a 37.2% area reduction. The area reduction of ATCM–MAC is 12.9%. Ax4MAC and Ax5MAC designs have 19.7% and 26.3% power reduction, respectively. Kul–MAC has a higher power consumption with 1.3% increase over the conventional MAC unit. ETM–MAC has a high power reduction while ATCM–MAC has an insignificant reduction, i.e., 22.4% and 1.3%, respectively. ETM–MAC has the highest delay reduction followed by Ax4MAC with 52.1% and 48.2%, respectively, while the other three designs are close in their values. Regarding energy consumption, Ax4MAC and Ax5MAC have 58.4% and 58.6% energy reduction, respectively, while ETM–MAC has a slightly higher energy reduction, i.e., 62.9%. Kul–MAC and ATCM–MAC exhibit about 42% energy reduction.

As shown in Figure 9, adding the conditional block achieves better power and energy efficiency at the cost of an increased area and delay. For example, Ax4MAC with 5% increased area has a 52.6% and 66.6% power and energy reduction, respectively. Similarly, Ax5MAC has 7.9% more area has 51.3% and 65.7% power and energy reduction, respectively. The power reduction for Kul–MAC, ETM–MAC and ATCM–MAC is 42.1%, 48.7% and 42.1%, respectively. Moreover, the energy reduction for these designs is 58.9%, 63.8% and 59.2%, respectively. However, their area is increased by an average of 15.3%.
FIGURE 9. Comparison of the Design Characteristics for Ax4MAC, Ax5MAC, Kul-MAC, ETM-MAC and ATCM-MAC with the Conditional-Block.

As a whole, there is no single design which is superior in all design metrics. However, all proposed designs exhibit a great design characteristics, in terms of area, power, delay and energy. For instance, with merely a 5% area overhead, Ax4MAC achieves 52.6% and 66.6% reduction in power and energy, respectively. Similarly, with a 7.9% area overhead, Ax5MAC achieves 51.3% and 65.7% reduction in power and energy, respectively.

VI. APPLICATIONS
This section describes logistic regression classifier and two image processing applications that we use to evaluate the proposed AxMAC designs, and validate their quality.

A. LOGISTIC REGRESSION
Prediction problems can be solved using logistic regression which is a classification method where its dependent variables have only two values, e.g., pass or fail. Logistic regression models are used to predict the probability of the binary response whenever a linear regression is not applicable. We evaluate our MAC units against a logistic regression classifier to predict whether a student gets admitted into a university [41]. We evaluate the model against 100 two-dimensional points that belong to two classes: Admitted or Not Admitted, based on applicant scores on two exams, as shown in Figure 11. We build a classification model that estimates an applicant probability of admission based on his scores.

The training accuracy of our classifiers proves how well the learned model predicts based on our training set as shown in Table 8. The training accuracy of the model based on the exact MAC unit is 90%. Models based on Ax1MAC – Ax3MAC, provide the same accuracy as the exact design while models based on Ax4MAC, Ax5MAC and ATCM-MAC have an accuracy almost identical to the exact design. The accuracy of models based on Kul-MAC and ETM-MAC is acceptable, i.e., 80% and 68%, respectively. Thus, our proposed AxMAC units provide high-quality results for the considered classification models which is higher than the related work.

B. IMAGE BLENDING
We evaluate and compare the accuracy of the proposed approximate MAC units utilizing an image blending application based on image multiplication mode, where two images are multiplied pixel-by-pixel, for all components, i.e., red, green and blue. The value of the pixels ranges from 0 to 255. We used MATLAB to evaluate the error metrics for image processing. To this end, we have modeled all approximate MAC units in MATLAB and used them in blending. We use the peak signal to noise ratio (PSNR), described in Equation 7, to measure the image quality.

Figure 10 shows a comparison of the obtained PSNR for different approximate MAC units. The PSNR for

TABLE 8. Training accuracy of classification models utilizing exact MAC and AxMAC units.

| Design     | Accuracy (%) |
|------------|--------------|
| Exact MAC  | 90           |
| Kul-MAC    | 80           |
| ETM-MAC    | 68           |
| ATCM-MAC   | 89           |
| Ax1MAC     | 90           |
| Ax2MAC     | 90           |
| Ax3MAC     | 90           |
| Ax4MAC     | 89           |
| Ax5MAC     | 89           |
Ax1MAC – Ax5MAC are 39.3dB, 41.2dB, 35.7dB, 48.7db and 48.3dB, respectively. Moreover, for Kul–MAC, ETM–MAC and ATCM–MAC designs, PSNR is 29.2, 29.1 and 43.7, respectively. Visually, all approximately processed images are non-distinguishable compared with the accurately processed ones. Designs based on Ax4MAC and Ax5MAC have the best output image quality with 66.6% and 65.7% energy reduction, respectively. This is consistent with the obtained results in Section V-B and the ones given in Table 7. Moreover, other designs have an acceptable PSNR, e.g., ATCM–MAC with 43.7dB. Generally, all proposed approximate designs have an acceptable quality degradation, each with different area, power and energy efficiency.

C. GAUSSIAN SMOOTHING

We also evaluate the efficiency of the proposed AxMAC units on a Gaussian smoothing filter application, which reduces image noise and details through attenuating high frequency signals, by working as a low-pass filter. Gaussian blur filter is widely used in mobile applications, such as Instagram and Snapchat, which involve image processing. Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a circularly symmetric 2-D Gaussian function, given in Equation 10 [42].

\[
G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \quad \text{(10)}
\]

\[
\text{Kernel} = \frac{1}{254} \begin{bmatrix} 24 & 30 & 24 \\ 30 & 38 & 30 \\ 24 & 30 & 24 \end{bmatrix} \quad \text{(11)}
\]

Theoretically, the Gaussian distribution is non-zero everywhere. However, in practice, it is effectively zero more than about three standard deviations (\(\sigma\)) from the mean (\(\mu\)). The Gaussian smoothed output is a weighted-average of each pixel's neighborhood. We use a (3 x 3) kernel, shown in Equation 11, based on \(\sigma = 1.5\), where the kernel average weight is more towards the value of the central pixels.

To illustrate the usefulness of the proposed AxMAC unit, we use its different versions within the Gaussian blur algorithm. All models are implemented in MATLAB. The Gaussian kernel, given in Equation 10, is applied to 8-bit gray-scale input images of size (512 x 512) pixels. Our input image is the benchmark lena image with added zero-mean, Gaussian white noise with a variance of 0.01 is added to the original grayscale image. The required 2D-convolution with (3 x 3) kernel requires 9 multiplication operations and 8 addition operations per pixel, which are expensive operations. Therefore, we approximated the Gaussian blur filtering by replacing MAC units with the proposed AxMAC units. In order to measure the quality of the output images, we use PSNR as given in Equation 7.

Figure 12 provides a pictorial representation of applying the Gaussian filter to the lena image, with different proposed designs of AxMAC units as well as Kul–MAC, ETM–MAC and ATCM–MAC, utilizing the (3 x 3) kernel. For referencing, the lena image with Gaussian-noise added and the noisy image filtered with exact MAC unit, are shown in Figure 12. The resulted PSNR values are computed with
respect to the image obtained by applying Gaussian filter with exact MAC unit on the noisy image. The PSNRs for the Ax1MAC – Ax5MAC are 32dB, 32.8dB, 30.3dB, 38.5dB, and 39dB, respectively. Moreover, for Kulp – MAC, ETM – MAC and ATCM – MAC designs, PSNR is 36.9, 37.1 and 38.2, respectively. As in image blending, the Ax4MAC and Ax5MAC units exhibit the best output quality with maximum energy reduction. Overall, all proposed designs have insignificant quality degradation, i.e., the PSNR ranges from 30.3dB to 39dB, with significant energy reduction.

VII. CONCLUSION
In this paper, we proposed a novel energy-efficient approximate MAC unit based on input awareness, which is suitable for error-resilient applications. The proposed AxMAC units are based on: 1) approximate multipliers for power reduction, and 2) an input-aware conditional block to approximate input operands then replace multiplication with a simple shift operation. The proposed input-aware MAC units are applicable to multipliers regardless of their size or being signed or unsigned. Proposed designs achieve an energy reduction of 66.6% and 65.7% with an overhead of 5% and 7.9% more area for Ax4MAC and Ax5MAC, respectively. These proposed designs are quite competitive compared to the state-of-the-art. The experimental results of logistic regression classifier and image processing applications show that the proposed approximate designs result in non-distinguishable outputs compared to the accurately processed ones. The output quality of approximate computing is highly input dependent, i.e., for some inputs, the output errors may reach unacceptable levels. Therefore, a run-time adaptive AXML unit design is being considered for future research as well as investigating multipliers with different settings, i.e., larger and/or signed operands.

REFERENCES
[1] J. von Neumann, “Probabilistic logic and the synthesis of reliable organisms from unreliable components,” in Automata Studies, vol. 34. Princeton, NJ, USA: Princeton Univ. Press, 1956, pp. 43–98.
[2] A. K. Mishra, R. Barik, and S. Paul, “iaACT: A software-hardware framework for understanding the scope of approximate computing,” in Proc. Workshop Approx. Comput. Across Syst. Stack, 2014, p. 52.
[3] J. Bornholt, T. Mytkowicz, and K. McKinley, “Uncertain \( < T > : \) A first-order type for uncertain data,” SIGPLAN Notices, vol. 49, no. 4, pp. 51–66, 2014.
[4] H. Nakahara and T. Sasao, “A deep convolutional neural network based on nested residue number system,” in Proc. Int. Conf. Field Program. Logic Appl., 2015, pp. 1–6.
[5] (2019). CEVA NeuPro a Family of AI Processors for Deep Learning at the Edge. Accessed: Aug. 6, 2019. [Online]. Available: https://www.ceva-dsp.com/product/ceva-neupro/
[6] M. Masadeh, O. Hasan, and S. Tahar, “Comparative study of approximate multipliers,” in Proc. ACM Great Lakes Symp. VLSI, 2018, pp. 415–418.
[7] M. Masadeh, O. Hasan, and S. Tahar, “Comparative study of approximate multipliers,” CoRR, vol. abs/1803.06587, pp. 1–23, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.06587
[8] R. P. P. Singh, P. Kumar, and B. Singha, “Performance analysis of fast adders using VHDL,” in Proc. Int. Conf. Adv. Recent Technol. Commun. Comput., 2009, pp. 189–193.
[9] L.-H. Chen, O. T.-C. Chen, T.-Y. Wang, and Y.-C. Ma, “A multiplication-accumulation computation unit with optimized compressors and minimized switching activities,” in Proc. Int. Symp. Circuits Syst., 2005, pp. 6118–6121.
[10] J.-K. Chung, H. Lee, and C.-S. Choi, “A power-aware variable-precision multiply-accumulate unit,” in Proc. Int. Symp. Commun. Inf. Technol., 2009, pp. 1336–1339.
[11] M. S. Kumar, D. A. Kumar, and P. Samundiswary, “Design and performance analysis of multiply-accumulate (MAC) unit,” in Proc. Int. Conf. Circuits, Power Comput. Technol., 2014, pp. 1084–1089.
[12] S. Datt, A. Chauhan, R. Bhadoriya, S. Nandi, and G. Trivedi, “A high-performance energy-efficient hybrid redundant MAC for error-resilient applications,” in Proc. Int. Conf. VLSI Design, 2015, pp. 351–356.
[13] D. Esposito, A. G. M. Strollo, and M. Alioto, “Low-power approximate MAC unit,” in Proc. 13th Conf. Ph.D. Res. Microelectron. Electron., 2017, pp. 81–84.
[14] A. Cilardo, D. D. Caro, N. Petra, F. Caserta, N. Mazzecco, E. Napoli, and A. G. M. Strollo, “High speed speculative multipliers based on speculative carry-save tree,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 12, pp. 3426–3435, Dec. 2014.
[15] I. Quieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, “Energy-efficiency approximate multiplier design using bit significance-driven logic compression,” in Proc. Design, Autom. Test Eur. Conf., 2017, pp. 7–12.
[16] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, “Design of fixed-width multipliers with linear compensation function,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 947–960, May 2011.
[17] M. de la Guia Solaz, W. Han, and R. Conway, “A flexible low power DSP with a programmable truncated multiplier,” IEEE Trans. Circuits Syst., vol. 59, no. 11, pp. 2555–2568, Nov. 2012.
[18] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, “Design of voltage-scalable meta-functions for approximate computing,” in Proc. Design, Autom. Test Eur., 2011, pp. 1–6.
[19] A. Raha and V. Raghunathan, “qLUT: Input-aware quantized table lookup for energy-efficient approximate accelerators,” ACM Trans. Embedded Comput. Syst., vol. 16, no. 5s, 2017, Art. no. 130.
[20] C. Alvarez, J. Corbal, and M. Valero, “Fuzzy memoization for floating-point multimedia applications,” IEEE Trans. Comput., vol. 54, no. 7, pp. 922–927, Jul. 2005.
[21] Y. Voronenko and M. Püschel, “Multiplierless multiple constant multiplication,” ACM Trans. Algorithms, vol. 3, no. 2, May 2007, Art. no. 11.
[22] H. T. Nguyen and A. Chattejee, “Number-splitting with shift-and-add decomposition for power and hardware optimization in linear DSP synthesis,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 8, no. 4, pp. 419–424, Aug. 2000.
[23] A. Raha, H. Jayakumar, and V. Raghunathan, “Input-based dynamic reconfiguration of approximate arithmetic units for video encoding,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 5, pp. 846–857, Mar. 2016.
[24] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan, “ASLAN: Synthesis of approximate sequential circuits,” in Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE), 2014, pp. 1–6.
[25] S. Hashemi, R. I. Bahar, and S. Reda, “DRUM: A dynamic range unbiased multiplier for approximate applications,” in Proc. Int. Conf. Computer-Aided Design (ICCAD), 2015, pp. 418–425.
[26] M. Imani, D. Peroni, and T. Rosing, “CPU$: $CPU$: Configurable floating point multiplier for energy-efficient computing,” in Proc. Design Autom. Conf., 2017, pp. 1–6.
[27] M. Imani, R. Garcia, S. Gupta, and T. Rosing, “RMAC: Runtime configurable floating point multiplier for approximate computing,” in Proc. Int. Symp. Low Power Electron. Design, 2018, pp. 12–1–12–6.
[28] P. Kulkarni, P. Gupta, and M. Ercogovac, “Trading accuracy for power with an underdesigned multiplier architecture,” in Proc. Int. Conf. VLSI Design, 2011, pp. 346–351.
[29] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, “Low-power high-speed multiplier for error-tolerant application,” in Proc. Int. Conf. Electron Devices Solid-State Circuits, 2010, pp. 1–4.
[30] T. Yang, T. Ukezono, and T. Sato, “Low-power and high-speed approximate multiplier design with a tree compressor,” in Proc. Int. Conf. Int. Conf. Design, 2017, pp. 89–96.
[31] Z. Yang, J. Han, and F. Lombardi, “Approximate compressors for error-resilient multiplier design,” in Proc. Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst., 2015, pp. 183–186.
[32] (2019). Virtex-6 XC6VLX75T FPGA. Accessed: Aug. 6, 2019. [Online]. Available: https://www.digilentinc.com/product-detail/en/xilinx/xc6vlx75t-1ff484t-xc6vlx75t-1ff484t-nd/2500879
[33] (2019). Mentor Graphics Modelsim. Accessed: Aug. 6, 2019. [Online]. Available: https://www.mentor.com/company/higher_ed/modelsim-student-edition

[34] (2019). Xilinx XPower Analyser. Accessed: Aug. 6, 2019. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xu733.pdf

[35] (2019). Xilinx Integrated Synthesis Environment. Accessed: Aug. 6, 2019. [Online]. Available: https://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html

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

[37] W.-C. Yeh and C.-W. Jen, “High-speed and low-power split-radix FFT,” IEEE Trans. Signal Process., vol. 51, no. 3, pp. 864–874, Mar. 2003.

[38] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, “Inexact designs for approximate low power addition by cell replacement,” in Proc. Design, Autom. Test Eur., 2016, pp. 660–665.

[39] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” IEEE Trans. Comput., vol. 62, no. 9, pp. 1760–1771, Sep. 2013.

[40] J. M. Rabaey, Digital Integrated Circuits: A Design Perspective. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[41] (2019). Logistic Regression. Accessed: Aug. 6, 2019. [Online]. Available: https://www.coursera.org/learn/machine-learning

[42] C. Solomon, Fundamentals of Digital Image Processing: A Practical Approach with Examples in MATLAB. Hoboken, NJ, USA: Wiley, 2011.

MAHMOUD MASADEH (S’18) received the B.Sc. degree in computer engineering from Yarmouk University, Irbid, Jordan, in 2003, the M.Sc. degree in computer engineering from the Delft University of Technology, The Netherlands, and the M.Sc. degree in management information system (MIS) from the Arabic Academy for Banking and Financial Sciences (AABFS), Jordan. He is currently pursuing the Ph.D. degree with Concordia University. From 2005 to 2011, he was a full-time TA, and from 2013 to 2016, as a full-time Instructor with Yarmouk University. He is currently a Research Assistant, under the supervision of Prof. S. Tahar, with Concordia University. His current research interests include approximate computing and energy-efficient VLSI circuit design. He is a Student Member of ACM and the Jordanian Engineering Association.

OSMAN HASAN (M’07–SM’14) received the B.Eng. degree (Hons.) from the University of Engineering and Technology at Peshawar, Pakistan, in 1997, and the M.Eng. and Ph.D. degrees from Concordia University, Montreal, Canada, in 2001 and 2008, respectively. From 2001 to 2003, he was an ASIC Design Engineer with LSI Logic Corporation, Ottawa, Canada. He is currently an Associate Professor with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Pakistan. He is also an Adjunct Professor with Concordia University. He is also a Founder and the Director of the SA Ve Lab, NUST, which mainly focuses on the design and formal verification of embedded systems. He was a member of the Association for Automated Reasoning (AAR) and a Member of the Pakistan Engineering Council (PEC). He has received several awards and distinctions, including the Pakistan’s Higher Education Commission’s Best University Teacher Award, in 2010, the Best Young Researcher Award, in 2011, and the President’s Gold Medal Award for the Best Teacher of the University from NUST, in 2015.

SOFIÈNE TAHAR (M’96–SM’07) received the Diploma degree in computer engineering from the Technische Universität Darmstadt, Darmstadt, Germany, in 1990, and the Ph.D. degree (Hons.) in computer science from the University of Karlsruhe, Karlsruhe, Germany, in 1994. He is currently a Professor and the Research Chair in formal hardware verification of systems-on-chip with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada, where he is also a Founder and the Director of the Hardware Verification Group. His current research interests include formal hardware verification, system-on-chip verification, AMS circuit verification, and probabilistic, statistical, and the reliability analysis of systems. He is a Senior Member of ACM and a Professional Engineer in the Province of Quebec. He received the title of the University Research Fellow upon receiving the Concordia University’s Senior Research Award, in 2007.