Run-time Configurable Approximate Multiplier using Significance-Driven Logic Compression

Ibrahim Haddadi[†ψ], Issa Qiqieh[‡], Rishad Shafik[†], Fei Xia[†], Mohammed Al-hayanni[∓], Alex Yakovlev[†]
[†]Microsystems Group, School of Engineering, Newcastle University, NE1 7RU, UK.
[ψ]Department of Computer Engineering, TAIBAH University, Kingdom of Saudi Arabia
[‡]Faculty of Engineering, Al-Balqa Applied University, Salt 19117, Jordan.
[∓]Department of Electrical Engineering, The University of Technology, Iraq.
E-mail: I.Haddadi2@ncl.ac.uk; i.qiqieh@bau.edu.jo; Rishad.Shafik@ncl.ac.uk; Alex.Yakovlev@ncl.ac.uk

***Abstract*—Designing energy-efficient hardware continues to be challenging due to arithmetic complexities. The problem is further exacerbated in systems powered by energy harvesters as variable power levels can limit their computation capabilities. In this work, we propose a run-time configurable adaptive approximation method for multiplication that is capable of managing the energy and performance tradeoffs — ideally suited in these systems. Central to our approach is a Significance-Driven Logic Compression (SDLC) multiplier architecture that can dynamically adjust the level of approximation depending on the run-time power/accuracy constraints. The architecture can be configured to operate in the exact mode (no approximation) or in progressively higher approximation modes (i.e. 2 to 4-bit SDLC). Our method is implemented in both ASIC and FPGA. The implementation results indicate that our design has only a 2.3% silicon overhead, on top of what is required by a traditional exact multiplier. We evaluate the efficiency of the proposed design through a number of case studies. We show that our method achieves similar image fidelity as in the existing approximate methods, without a delay penalty. Further, the inclusion of the dynamic approximation techniques is justified by up to 62.6% energy savings when processing an image with a multiplier using 4-bit SDLC and 35% energy savings when using 2-bit SDLC. In addition, case study results show that the proposed approach incurs negligible loss in output quality with the worst PSNR of 30dB when using the 4-bit SDLC multiplier.**

***Index Terms*—Energy efficiency, approximate computing, low power design, multiplier design.**

I. Introduction

The concept of approximate arithmetic involves replacing system components of normal degrees of complexity with less complex components, which may provide reduced accuracy in one way or another. This aims to achieve lower power consumption, higher performance and lower area, with acceptable levels of accuracy [1], [2], [3].

Over the years, approximate hardware designs have been extensively researched. Many studies consider the pruning of arithmetic complexity leveraged by the inherent error-resilience of certain real-world applications. Examples span across various design strategies, such as accuracy scaling [4], approximation of parallel logical patterns [5], [6] and hardware/software approximations for NNs [7]. The commonality among these examples is to achieve power savings by employing static-configuration methods for specifying fixed approximate processing units, including approximate multipliers and adders [8], [9], based on design-time predictions of environment and data conditions.

In approaches that are aware of the bit-level precision [10], the more significant blocks are processed using exact arithmetic units, whilst non-significant blocks are processed using approximate and low-complexity ones. Recently, a Significance-Driven Logic Compression (SDLC) approximation method has been proposed for multipliers, where complex logic operations are replaced by low-complexity logic gates for a group of partial product terms depending on progressive bit significance [1]. This method allows the computations to operate at different logic compression levels, designed in different multiplier units. By suitably choosing the multiplier logic compression level, the method offers a varying degree of energy efficiency and performance. To enable this a number of multipliers are designed with selection circuitry, which is expensive in terms of area and idle/leakage power.

Emerging ubiquitous systems, in particular systems based on energy-harvesting, represent a paradigm shift from traditional systems. The energy supply of such systems can vary temporally and spatially within a dynamic range, essentially making computation extremely challenging [5], [11], [12], [13]. Adaptive hardware approximations with tunable energy and accuracy trade-offs can present opportunities in these systems for elastically continuing computation under varying power levels. Certain real-world applications rely on approximate computing due to its inherent error resilience. For instance, around 82% of run-time is spent on unnecessary computations, which can alternatively be executed in approximate modes [14] to reduce energy consumption.

To date, there has been limited research on configurable hardware designs to operate in different approximation modes in response to environmental and operating conditions such as energy availability and data accuracy requirements. This paper is an attempt to address this need.

A. Contributions

For this work, we extend the SDLC approximation method further as follows. We design a new configurable multiplier using adaptive SDLC multiplier architecture. The architecture leverages a tunable approximation method to specify the logic compression level depending on the model-driven energy and accuracy trade-offs. Thus, by allocating the appropriate configurations during run-time, we can adjust the computation capability under variable power or energy budgets. Specifically, in this work, we make the following *contributions*:

- propose a new multiplier architecture using variable approximation, which supports run-time configuration;
- evaluate the impact of the configurability on the output quality and non-functional metrics;
- show implementations of the proposed architecture in three case studies, including the design of Energy-Aware Configuration Algorithm (EACA) that can determine the right configuration under variable power or energy budgets during run-time.

B. Paper Organization

The rest of the paper is organized as follows: Section II describes the SDLC method and the motivation of this work. Section III explains in detail the proposed design method. Section IV presents error analysis and validates the configurable architecture by comparing with individual approximate multipliers. Section V compares the non-functional metrics such as area, speed and energy with competing designs. Section VI presents the EACA algorithm and two cases of real-world problem solving. Finally, Section VII concludes the paper, highlighting our future work.

II. Existing SDLC Method and Motivation

Recently, Qiqieh *et al.* [6] proposed an approximate multiplier design with different levels of logic compression depending on bit significance, called significance-driven logic compression (SDLC). Their investigation highlighted energy-accuracy trade-offs corresponding to these levels. The principle idea is to combine the partial product terms and compress them progressively based on significance (i.e. least significant bits are more compressed compared to more significant bits) through replacing the AND gates by the OR gates. Architecturally, this leads to reduction of the carry propagation chain length and thereby energy efficiency at the cost of minor accuracy loss.

Figure 1 illustrates the SDLC approach using a dot notation for (8×8) parallel multiplier. This can be explained in three steps, as follows. To begin, (8×8) AND logic gates are used to generate the partial product matrix (PPM). The first step aims to form a cluster of several rows in the PPM. The depth of the clusters may be 2 or 4 bits (2- or 4-bit SDLC). The idea is to utilize array of OR logic gates to reduce the number of product terms within each cluster. The next step describes how OR logic array is used to compress the terms into a single row of bits leading to a reduction of the total number of product terms within each cluster in the PPM to decrease, as seen in step 2. A commutative remapping of the bit sequence is applied in the third step to form a decreased number of the partial product rows after applying the logic compression. The number of the rows is depending on the depth of the clusters selected in the first step. The dotted rectangles indicate the critical column's height, which is reduced by half in case of 2-bit SDLC and by quarter for the 4-bit SDLC in step 3 compared to the exact accumulation tree in step 1.

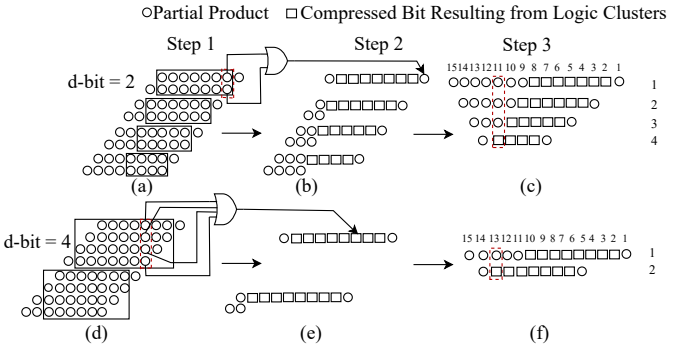

Fig. 1. Two different sizes of logic clusters are used to compress partial products based on their progressive bit-significance in an (8 x 8) parallel multiplier architecture. (a) clustering a group of bits within two successive rows in the partial product bit-matrix after bitwise multiplication; (b) generating a reduced set of product terms after targeting the depth of 2-row logic compression; (c) ordered matrix after applying commutative remapping of the bit sequence resulting from the SDLC approach; (d), (e), and (f) the same process when applying 4-bit depth of logic clusters. The d-bit indicates the depth of logic cluster.

After these three steps the result accuracy varies depending on the significant bit sizes. More compression (i.e. clustering of rows) is performed for bits of lower significance (compression happens for the square cells) and progressively less compression is performed for higher significance bits (no compression for the round cells). The final product is then generally approximate with its loss of accuracy under control. Because of shorter carry chain paths and a reduction of the number of single-bit adders, the energy is reduced drastically and speed is improved.

Although the SDLC method provides substantial energy reduction, it lacks run-time configurability. In other words, the accuracy cannot be adjusted dynamically with variable compression levels. As a result, the multiplier models produced can provide only a specific range of performance, energy and power due to static design.

In certain fields of application, such as artificial intelligence and signal processing, variable accuracy can be leveraged in favor of energy savings opportunistically depending on the power or energy availability. For example, when power delivery is high it may be possible to tune the accuracy mode to a higher level, while in low power situations the accuracy can be scaled down. This is done with the aim of longer operating lifetime and survivability of the execution. Section VI includes an exemplar of similar

applications. To facilitate this, we are keen to re-design the SDLC based multiplier for run-time reconfigurability with power awareness. Our key hypothesis is as follows: by designing variable SDLC knobs in a multiplier, it is possible to leverage the SDLC levels in response to power levels dynamically and improve adaptability and survivability under variable power situations. To corroborate our hypothesis, we design a new configurable and energy-aware multiplier using three different modes: an exact and two different approximate levels of 2- and 4-bit SDLC multipliers.

III. Proposed Configurable Approximation Hardware

Whilst it is possible to construct a system with copies of separate exact and approximate (e.g. SDLC) multipliers and select which one to use depending on the environmental conditions, such a design is wasteful of silicon and may cause substantial energy loss through leakage. This method is therefore not suitable for systems that have to operate under power and energy uncertainty. In the following subsections, we present a novel, run-time configurable multiplier design that allows dynamic approximation needs in such applications.

A. Configurable Multiplier Architecture

We propose a configurable multiplier, which provides exact and approximate versions for use under appropriate conditions. This multiplier only requires an insignificant additional amount of silicon compared to a regular exact multiplier. This is realized by maximally re-using the single-bit adders for different configurations of the multiplier.

Figure 2 shows the difference between the exact and the proposed multiplier. Generally, the final product of exact multiplication can be generated after following three main steps: 1) PPM is formed. 2) PPM is reduced to a height of two rows using any accumulation method such as Wallace or Dadda trees, then 3) these two rows are combined by using a carry propagation adder (see Figure 2 (a)). In the proposed multiplier, the SDLC approach is included after PPM formation (as shown in Figure 2 (b)), to minimize the number of rows in PPM using logic compression (see Figure 1). Therefore, this decreases the delay needed for PPM to be reduced to a height of two rows (i.e. during stage 2). Then, the minimized PPM is accumulated to a height of two rows and summed up using CPA similarly in steps 2 and 3 of the exact multiplier. In this paper, we demonstrate our approach based on the Wallace tree structure, which, similar to other multi-stage tree multipliers, achieves high speed due to a lower logic depth compared to more conventional designs [15][16].

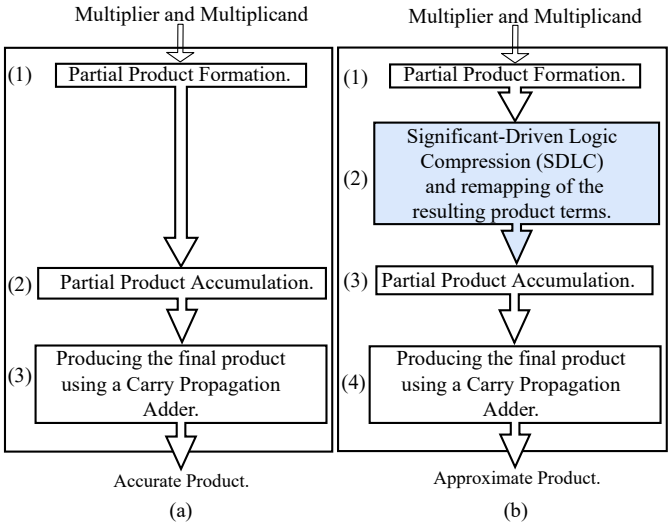

Fig. 2. Process chart explaining the difference between the main stages in (a) exact multiplication and (b) the proposed multiplication using SDLC approach.

For this work the Wallace tree serves as an example. Our method can be applied based on any exact multiplier design including multi-stage tree structures such as Wallace and Dadda trees. Fundamentally, adders in the multiplier serving as the exact configuration are re-purposed through re-wiring to implement the approximate configurations.

Figure 3 illustrates how the proposed configurable (8 x 8) multiplier design performs the exact and approximate multiplication. In this design, each circle represents one partial-product bit. The necessary half adders are marked by rectangles spanning columns of two partial product bits and full adders are marked by rectangles spanning columns of three bits. The exact configuration in (a) represents a traditional Wallace tree multiplier which uses the largest number of half and full-adder units within the reduction stages. The 2-bit SDLC configuration presented in (b) is considerably smaller, and the 4-bit SDLC in (c) is the smallest. It is worth noting that the proposed configurable design requires a few additional number of adders on top of the already existing adders needed by the exact configuration.

By investigating the proposed configurable multiplier design with its exact and approximate (2 and 4-bit SDLC) configurations, we identify structural similarities in the shapes of the SDLC's and parts of the Wallace tree multiplier. For instance, in Figure 3, the 2-bit SDLC's reduction stages and the carry-propagate addition (CPA) parts in (b) can be mapped onto stages 3 and 4 and the CPA of the exact reduction tree in (a), with only ten additional partial-product bits required (shown in black and diameter-line circles). Also, the 4-bit SDLC in (c) can be mapped onto the CPA in (a) with just a single additional bit. This bit is also shared with the 2-bit SDLC.

Table I may be read in conjunction with Figure 3 to represent the required single-bit adders in all parts of the proposed configurable multiplier design. As can be seen from the table, the exact configuration requires the majority of half and full adders, and only a very small number of adders are needed in addition to the exact configuration to accommodate the SDLC configurations. In Section V the additional silicon area required by the SDLC's is calculated.

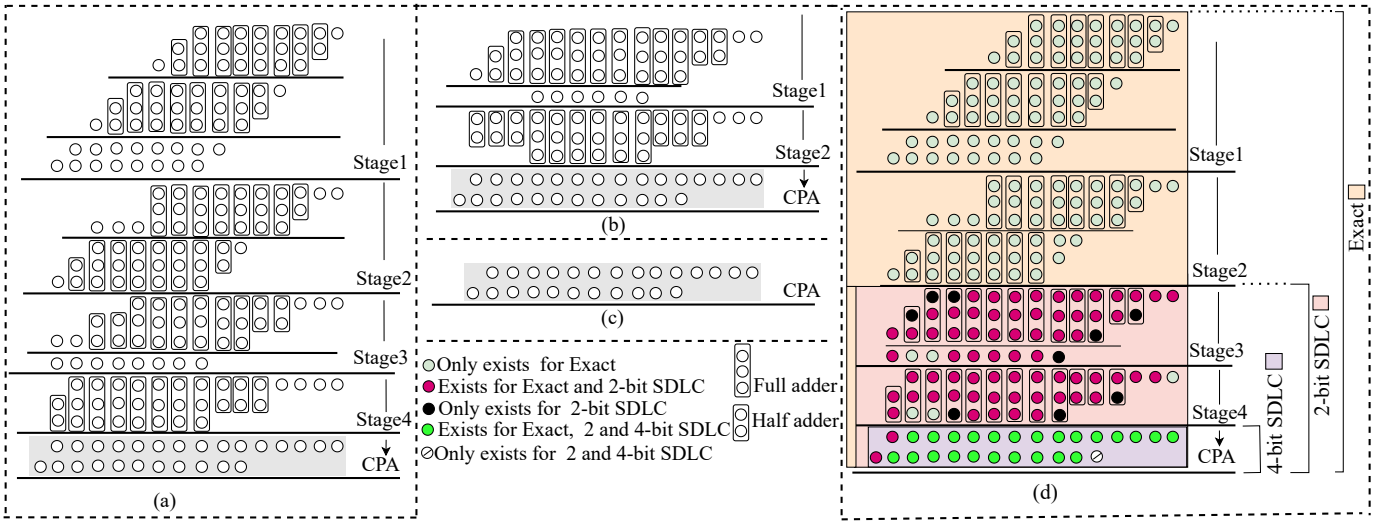

Fig. 3. The reduction stages of an (8 × 8) Wallace tree multiplication illustrate the accumulation method for the PPM formed from exact and two different sizes of logic clusters (shown in Figure 1) (a) four reduction stages are required in case of (8 × 8) traditional Wallace tree multiplier(WTM); (b) two reduction stages are required by Wallace accumulation method to reduce the PPM generated by 2-bit SDLC(see Figure1 (c); (c) no reduction stages for 4-bit SDLC as the height of the PPM is only two rows(see Figure 1 (f)), and (d) configurable (8 × 8) Wallace tree multiplication includes the common similarities and variations shown in (a), (b) and (c).

TABLE I
THE NUMBER OF FULL AND HALF ADDERS USED BY THE ACCUMULATION STAGES AND CPA FOR DIFFERENT EXACT, APPROXIMATE MULTIPLIERS AND THE PROPOSED CONFIGURABLE (8 × 8) WALLACE TREE.

Multiplier →	Exact		2-bit SDLC		4-bit SDLC		Proposed	
Adder type	HA	FA	HA	FA	HA	FA	HA	FA
Stage 1	4	12	3	9	-	-	4	12
Stage 2	3	13	6	6	-	-	3	13
Stage 3	4	6	-	-	-	-	3	9
Stage 4	4	7	-	-	-	-	4	8
CPA	10		11		11		11	

B. Hardware Knobs for Run-Time Configuration

In the case of the (8 x 8) multiplier, a 3-to-1 Multiplexer (MUX) is needed for top-level configuration selection. This switch provides the actuation facility for controlling the configuration according to whatever rules the designer sets for the configuration strategy. A configuration signal is received from the controller and fed to the MUX to select one of the three configurations accordingly.

The configuration procedure is low-overhead, no more than a couple of layers of parallel switches. Even after including the two-bit to three-wire one-hot signal decoding logic, the entire procedure should fit comfortably within a single clock cycle for any reasonably modern technology.

In this work, we synthesized the hardware into FPGA to characterize the energy and power required for executing each configuration (see Section V). This is because we eventually implement on FPGA. If other technologies are used in the implementation, the same characterization can be performed on the relevant technology. From these characterization data, it is possible to derive energy thresholds that an energy-aware configuration algorithm may use to make control decisions.

If the rules of configuration are not determined by energy availability, different characterization experiments may be carried out to produce appropriate threshold values in the alternative physical parameters that are important for the configuration control. In this paper, we concentrate on energy-aware configuration. For instance, one of our case studies, the Energy-Aware Configuration Algorithm (EACA), which will be presented in Section VI-A, concentrates on efficient energy usage. As a result, all our experimental work is focused on energy being the crucial physical parameter. The hardware facilities that provide the hooks for configuration described in this section target energy-aware designs, but are generic enough to need no adjustment or little adjustment for control based on other parameters.

With three different configurations in total, a configuration signal produced by the control module consists of two binary bits. We set 00 for the exact configuration, 01 for the 2-bit SDLC configuration, and 1x (with x representing "don't care", i.e. can be either 0 or 1) for the 4-bit SDLC configuration. This is shown in Figure 4. Giving the don't care to the 4-bit SDLC configuration shows our energy-centric design priority, as we use the smallest circuit to select the smallest configuration, which is likely chosen when energy supply is low. Based on the configuration selection input signal, the control hooks perform the following actuation:

- the right groups of adders are included in the configuration - for the example in Section III-A, the three selectable groups roughly correspond with the three parts of the exact Wallace tree: Stages 1 and 2, Stages 3 and 4, and the CPA. The exact configuration includes all three groups, the 2-bit SDLC configuration includes Stages 3 and 4 and the CPA, and the 4-bit SDLC includes the CPA.

- additional adders are included as appropriate for the SDLC configurations.
- appropriate re-wiring of the full and half adders (growing some half adders to full adders and shrinking some full adders to half adders) for the appropriate SDLC configurations.

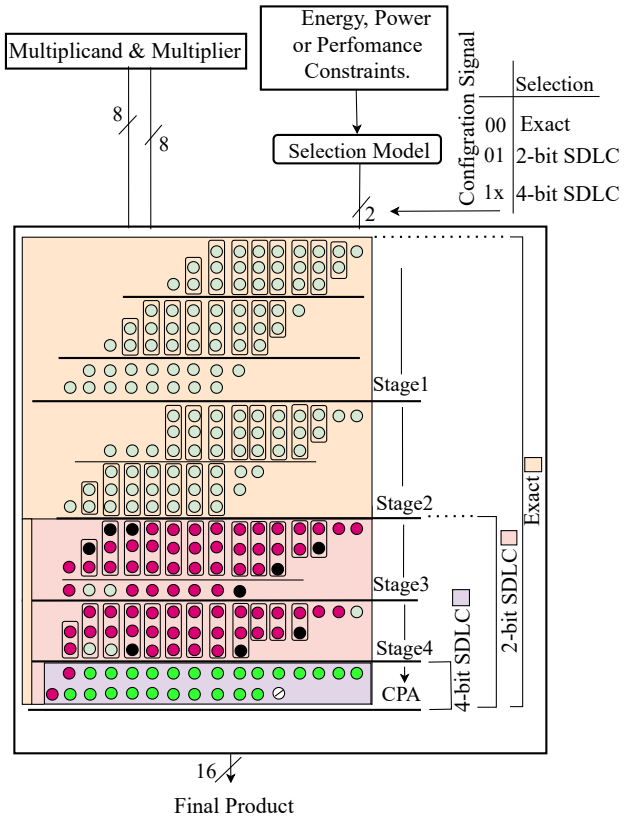

Fig. 4. Diagrammatic sketch of the proposed hardware architecture of the configurable (8 × 8) multiplier with exact, 2-bit and 4-bit SDLC modes.

IV. Error Analysis

In this section, the impact on accuracy of the proposed approach is investigated in the form of error analysis.

Several error metrics have been discussed in [17] and [18] for evaluating and quantifying errors. The error distance (ED) is defined as the arithmetic difference between the accurate (exact) product (P) and erroneous (approximate) product (P'). The relative error distance (RED) is the ratio of ED over the accurate output, and it is defined as in [19]:

$$RED = \frac{ED}{P} = \frac{|P - P'|}{P}. \tag{1}$$

Note that we assume that the products are positive, as the multipliers are for unsigned integers.

The mean RED (MRED) is one more error metric for any (N × N) approximate multiplier, it is defined as:

$$MRED = \frac{\sum_{i=0}^{2^{2N-1}} RED_i}{2^{2N}}. \tag{2}$$

MRED is the mean value of RED across all possible different operand pairs of the multiplication. Without knowing the application, here we assume that each unique pair of operand values has exactly the same probability of happening (uniform distribution).

For comparing multipliers of different degrees and methods of approximation, we use the normalized MRED (NMED):

$$NMED = \frac{MRED}{P_{max}} = \frac{\frac{\sum_{i=0}^{2^{2N-1}} RED_i}{2^{2N}}}{P_{max}}, \tag{3}$$

where P_{max} is the maximum product that can be obtained from an (N x N) accurate multiplier, i.e. $P_{max} = (2^{N-1})^2$. We perform simulation studies in Matlab by incorporating a functional model of the proposed multiplier using different logic clusters with (8 × 8) Wallace tree accumulation.

TABLE II
MRED and NMED for different approximate configurations in an (8 × 8) configurable multiplier.

Multiplier ↓	MRED (%)	NMED(%)	Difference from [6]
2-bit SDLC	1.9883	0.3527	0 (%)
4-bit SDLC	10.5836	3.2723	0 (%)

Table II lists the error trade-off when changes between the two degrees of approximation (2- and 4-bit SDLC) in the new design. As can be seen, the MRED is only minimal for the 2-bit SDLC type. The 4-bit SDLC also records a small error. A similar observation can be made in the case of the NMED metrics. Moreover, the table includes the difference between the approximation parts in this new design and the previous work [6], in order to investigate if by moving to a configurable design, a price is paid in errors. The results show that the differences are 0%, and there is no error overhead.

For more extensive analysis involving error metrics, for instance in the case study in Section VI-B, different configurations of the proposed configurable multiplier are used in calculating the Gaussian blur algorithm. We use such metrics as Peak Signal to Noise Ratio (PSNR). PSNR is a fidelity metric used to measure the quality of the output images. PSNR is expressed as:

$$PSNR = 20log_{10}\frac{255}{\sqrt{MSE}}, \tag{4}$$

where MSE is the mean squared-error measured with respect to the reference pixel [20]. We can find the total MSE of the image as the sum of all sub-image MSE values as follows:

$$MSE_B^k = \frac{1}{m \cdot n}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(B_k(i,j) - B'_k(i,j)\right)^2. \tag{5}$$

These PSNR analysis methods are used in Section VI-B.

V. Comparative Evaluations

In this section, we compare the area, delay and power tradeoffs of our multiplier design with recently proposed approaches, considering two hardware implementations: ASIC and FPGA. These implementations are included in order to achieve a wider insight of comparative evaluations

as some of the compared implementations are in ASIC, while others are in FPGA.

A. Area, Delay & Power Tradeoffs in ASIC Implementations

For ASIC comparisons, a generic System-Verilog code is used to generate synthesizable modules for the proposed configurable multiplier. Mentor Graphics Questa Sim is used to compile the System-Verilog code and run the associated testbenches. Synopsys Design Compiler is used for synthesizing the multiplier configurations, and the circuits are implemented in the Faraday 90*nm* technology library. The compared methods are implemented in exactly the same way, so that comparisons can be performed on the same implementation technology node and library.

Table III presents area, delay and power tradeoff figures when compared with [6]. As can be seen, the configurable hardware is larger in terms of area than the exact multiplier alone. A delay overhead is also expected in the configurable design because of the increased number of adders.

TABLE III
COMPARING EXISTING MULTIPLIER DESIGNS AND THE PROPOSED CONFIGURABLE DESIGN IN TERMS OF POWER(P), AREA (A) DELAY(DL) AND POWER-DELAY PRODUCT (PDP).

Multiplier ↓	$P(\mu W)$	$A(um^2)$	DL(ns)	$PDP(fJ)$
Fixed Configs [6]	158.39	3495.71	7.82	1238.6
Proposed Exact	66.20	1450.40	2.66	176.1
Exact [1]	62.42	1417.47	2.63	164.2
Proposed 2-bit SDLC	39.21	904.56	2.11	82.7
Proposed 4-bit SDLC	25.42	501.37	1.35	34.32

It can be seen from Table III that the area overhead, comparing the configurable multiplier (Proposed Exact) to the exact multiplier on which it is based (Exact [1]), is 2.3%. The power overhead is 6%, the latency overhead is 1.1%, and the power-delay product overhead is 7.2%. However, the proposed model saves more than 82.73% of area compared with the solution proposed in [6], which includes separate exact and SDLC multipliers for runtime selection. This large area reduction implies significant power savings because of leakage power, leading to large PDP reductions.

The competitive figures obtained on these non-functional metrics mean that this configurable design would also compare favourably against the other designs in [1] and [6].

B. Area, Delay & Power Tradeoffs in FPGA Implementations

For a more flexible design, the configurable multiplier is also implemented in FPGA using Xilinx Vivado Design Suite for the Ultra96-V2 platform [21]. The compared existing designs, originally also on FPGA, are re-implemented on this same platform for fair comparisons.

A previous study includes different designs of exact multipliers on FPGA (see Table IV), focusing on performance [22]. These are compared with our three configurations for non-functional parameters. Table IV lists the results for each design in terms of area, delay, and power. It can be seen that the proposed configurable multiplier, in its different configurations (between exact to 4-bit), is competitive in terms of area, delay, and power compared to Modified Radix2 Booth Multiplier (MRBM) and WTM. It is noteworthy that our exact configuration is competitive in delay with these multipliers, which were designed for speed.

TABLE IV
COMPARING NON-FUNCTIONAL METRICS WITH KUMAR ET AL [22].

Multiplier ↓	Area (LUT's)	Power (W)	Delay (ns)
MRBM [22]	137	1.010	6.721
WTM [22]	96	1.061	6.102
Proposed Exact	91	1.22	6.102
Proposed 2-bit SDLC	65	0.32	4.1
Proposed 4-bit SDLC	42	0.23	3.8

VI. CASE STUDIES

In this section, first we illustrate an example configuration selection algorithm. This algorithm attempts to find the maximum usage of available energy by selecting the least approximate configuration. Later, we present two more case studies for the configurable multiplier in real-world problems. These case studies demonstrate the capabilities of the proposed configurable multiplier as well as the validity of the configuration selection algorithm.

A. Energy-Aware Configuration Algorithm (EACA)

In order to have a design that can survive under unreliable power supply (non-deterministic fluctuations in power levels) and guarantee reliable computation, we need to build a system with survival instincts. We design a configuration controller taking advantage of the three-mode configurable multiplier architecture to implement energy-aware execution. The EACA model, which fits into the "Selection Model" box in Figure 4, is shown in Algorithm 1.

Algorithm 1 Energy-aware configuration algorithm

```
Initialize with energy figures
const: k = number of different approximation configurations
for i = 1 to k do
    E (i) = energy required for the i^th configuration
end for        ▷ i = 1 is exact and i = k is the min config
begin EACA
while true do
    Eia = 0                          ▷ instantaneous available energy
    j=1                                          ▷ initialize index
    wait control cycle time length
    while Eia < E(k) do
        obtain Eia from energy supply
    end while          ▷ until enough energy for min config
    while Eia < E(j) do
        j = j+1
    end while          ▷ find the least approx config for Eia
    select the j^th configuration
end while
end EACA
```

EACA assumes the existence of energy supply information at run-time, which is not uncommon among well-designed energy harvesting systems [23],[24],[25]. The available

energy is then compared with the required energy to execute the least energy-hungry configuration. If the available energy is insufficient, the multiplier is not executed whilst the monitoring of incoming energy continues. Once the available energy is confirmed to be enough to execute at least the configuration with the smallest energy requirement (in the (8 × 8) multiplier example in this work, this is the 4-bit SDLC configuration), EACA continues to the configuration selection stage. In this stage, EACA progressively tests the available energy against the energy required by the different configurations one by one, starting from the heaviest configuration with the smallest approximation, and moving towards the lightest configuration, which has the greatest degree of approximation.

In principle, EACA attempts the following optimization problem: Treating the available energy as a constraint, maximize the multiplication accuracy (or minimize the multiplication approximation) by selecting the correct configuration that fits the available energy.

B. Gaussian Blur Filter

In this case study, we evaluate the efficacy of the proposed technique with a real life image processing application, which consists of additions and multiplications using our three multiplier configurations. Our analysis considers the Gaussian blur filter [26] since it is widely used in graphics software, typically to reduce image noise and artifacts (e.g. Moiré effects) by acting as a low-pass filter.

We constructed the multiplier in Matlab covering 2-, and 4-bit depth clustering in the case of (8 × 8) multiplication. The Gaussian kernel is (3 × 3) with a 1.5 standard deviation value, and it uses 8-bit fixed point arithmetic and is applied to 8-bit gray-scale input images of a size of (500 × 500) pixels. We approximate Gaussian blur by replacing the standard multiplication in the Gaussian filter with the aforementioned approximate multipliers.

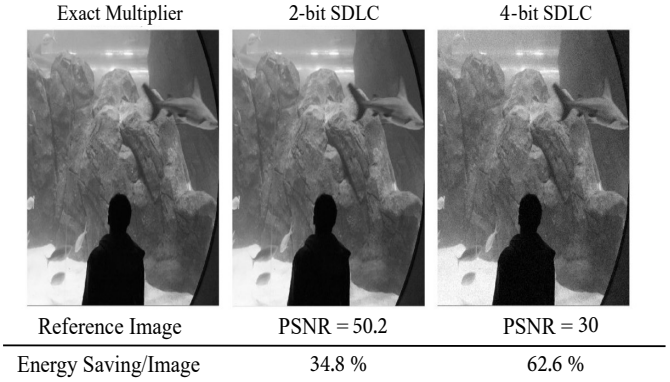

Fig. 5. Output quality and energy consumption for Gaussian blur filtering using the three different configurations of the proposed (8 × 8) multiplier.

Figure 5 illustrates the impact of different configurations on the image quality after applying the Gaussian blur filter. As shown in Figure 5 the use of the SDLC approach can yield reasonable results.

The PSNR values for the case of 2-, and 4-bit logic clustering for (8 × 8) SDLC are 50.2dB and 30dB respectively. The values of PSNR are computed compared to the reference image, obtained from applying Gaussian blur filtering using the exact configuration, according to (4).

To calculate the consumed energy in the multiplier when processing an input image, we follow

$$Energy = Power * Delay * N, \tag{6}$$

where Power and Delay are obtained for one multiplier design from the synthesis tool. N is the number of multiplications necessary to treat the input image by Gaussian filter. The energy savings are then calculated compared to the conventional exact multiplier.

The energy savings and PSNR results obtained, shown in Figure 5, demonstrate that the proposed approach can provide significant dynamic energy savings with acceptable image qualities. The quality-energy tradeoff is clear across the three configurations.

C. Energy-Aware Approximation

Figure 6 shows an example execution trace where the supply energy is variable within a wide range with time. The highest amount of energy available is shown to be 250 fJ and the lowest amount of energy available is shown to be 35 fJ. EACA's energy-aware configuration helps the system achieve reasonable PSNR figures in the execution for the available energy at any time. The priority is survival of the continuous execution by optimizing PSNR whilst satisfying energy constraints. This trace confirms that this configurable multiplier supports EACA's sustainability focus. In this experiment, the extra overheads used in the configuration circuits prove to be negligible compared to the multiplier's costs. EACA itself is assumed to be run on an outside processor whose energy is external to the shown energy budget, without losing generality. A future work would be the simplified realization of EACA on low-cost hardware.

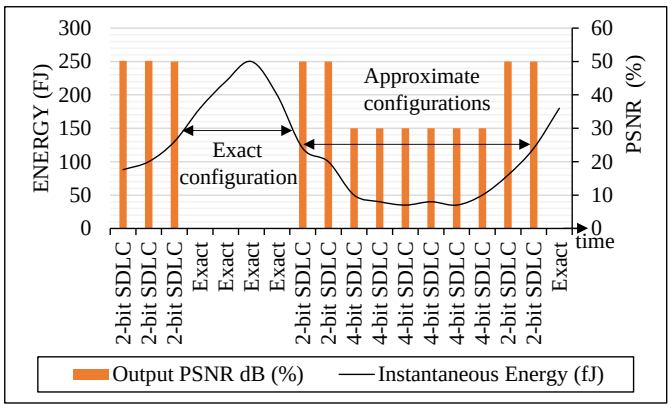

Fig. 6. Different scenarios for the EACA model operate at run-time under highly variable energy conditions while sustaining execution.

VII. Conclusions

We present a configurable multiplier design with the ability to dynamically tune the approximation in the multiplier through logic compression control. We implemented the System-Verilog design on an ASIC, using Synopsys Design Compiler and Xilinx Vivado Design Suite. The synthesized results show that up to 43.75% of energy savings and almost 38.03% of reduction in critical path delay can be achieved. We also implement the proposed multiplier on the Ultra96v2 Evaluation FPGA Platform. The results show that the proposed approach can provide significant energy and area savings with negligible loss in output quality (the worst PSNR is 30dB for the 4-bit SDLC multiplier).

We demonstrate the capabilities of the configurable multiplier by introducing the EACA method of finding the optimal configuration of the multiplier depending on the available energy. The results show that the EACA model allows the proposed design to operate at run-time under highly variable energy conditions while sustaining execution.

The highlight of this configurable multiplier is the sharing of the same adders between different configurations, saving both silicon and leakage energy.

We believe that the proposed multiplier architecture leverages arithmetic approximation to support energy-efficiency for applications where detailed accuracy may not be important, such as machine learning.

However, scaling up the current architecture to a higher level of complexity is a non-trivial topic and promising direction for future work.

Acknowledgement

The authors would like to acknowledge the funding support from EPSRC IAA Project Whisperable (Newcastle University, UK) and also thank the College of Computer Science and Engineering (CCSE), TAIBAH University, Kingdom of Saudi Arabia for funding postgraduate research of the lead author.

References

[1] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 7–12.

[2] L. Sekanina, "Introduction to approximate computing: Embedded tutorial," in *2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2016, pp. 1–6.

[3] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, Mar. 2016. [Online]. Available: https://doi.org/10.1145/2893356

[4] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y. Cheung, and G. A. Constantinides, "Deep neural network approximation for custom hardware: Where we've been, where we're going," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, p. 40, 2019.

[5] R. Shafik, A. Yakovlev, and S. Das, "Real-power computing," *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1445–1461, Oct 2018.

[6] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, S. Das, and A. Yakovlev, "Significance-driven logic compression for energy-efficient multiplier design," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 417–430, 2018.

[7] G. Zhong, A. Dubey, C. Tan, and T. Mitra, "Synergy: A HW/SW framework for high throughput cnns on embedded heterogeneous SoC," *CoRR*, vol. abs/1804.00706, 2018.

[8] K. Al-Maaitah, G. Tarawneh, A. Soltan, I. Qiqieh, and A. Yakovlev, "Approximate adder segmentation technique and significance-driven error correction," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–6.

[9] K. Al-Maaitah, I. Qiqieh, A. Soltan, and A. Yakovlev, "Configurable-accuracy approximate adder design with light-weight fast convergence error recovery circuit," in *2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2017, pp. 1–6.

[10] D. Burke, D. Jenkus, I. Qiqieh, R. Shafik, S. Das, and A. Yakovlev, "Special session paper: significance-driven adaptive approximate computing for energy-e!icient image processing applications," in *2017 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2017, pp. 1–2.

[11] S. P. Beeby, R. Torah, M. Tudor, P. Glynne-Jones, T. O'donnell, C. Saha, and S. Roy, "A micro electromagnetic generator for vibration energy harvesting," *Journal of Micromechanics and microengineering*, vol. 17, no. 7, p. 1257, 2007.

[12] J. A. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," *IEEE Pervasive computing*, vol. 4, no. 1, pp. 18–27, 2005.

[13] D. Dondi, A. Bertacchini, D. Brunelli, L. Larcher, and L. Benini, "Modeling and optimization of a solar energy harvester system for self-powered wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 7, pp. 2759–2766, 2008.

[14] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–9.

[15] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, 1964.

[16] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Fifteenth International Symposium on Quality Electronic Design*, 2014, pp. 263–269.

[17] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on computers*, vol. 62, no. 9, pp. 1760–1771, 2012.

[18] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[19] J. M. Phillips and W. M. Tai, "The gaussiansketch for almost relative error kernel distance," *arXiv preprint arXiv:1811.04136*, 2018.

[20] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 258–261.

[21] J. A. Belloch, G. León, J. M. Badía, A. Lindoso, and E. San Millan, "Evaluating the computational performance of the xilinx ultrascale+ eg heterogeneous mpsoc," *The Journal of Supercomputing*, vol. 77, pp. 2124–2137, 2021.

[22] M. S. Kumar, D. A. Kumar, and P. Samundiswary, "Design and performance analysis of multiply-accumulate (mac) unit," in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*. IEEE, 2014, pp. 1084–1089.

[23] P. D. Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, "Energy harvesting from human and machine motion for wireless electronic devices," *Proceedings of the IEEE*, vol. 96, no. 9, pp. 1457–1486, 2008.

[24] A. S. Weddell, M. Magno, G. V. Merrett, D. Brunelli, B. M. Al-Hashimi, and L. Benini, "A survey of multi-source energy harvesting systems," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 905–908.

[25] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2014.

[26] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.