Microelectronics Journal 145 (2024) 106135

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Logic cloning based approximate signed multiplication circuits for FPGA ☆

Abhinav Kulkarni *, Messaoud Ahmed Ouameur, Daniel Massicotte

Electrical and Computer Engineering Department, Université du Québec à Trois-Rivières, Trois-Rivières, QC G9A 5H7, Canada

ARTICLE INFO

Dataset link: https://github.com/abhinav333/Logic-Cloning

Keywords:
Approximate computing
Reconfigurable computing
Logic cloning
Arithmetic multiplication
Massive MIMO

ABSTRACT

As hardware circuits become larger and more intricate, there is a growing need for approximate circuit techniques. These approaches offer a trade-off, sacrificing some system accuracy in exchange for greater hardware resource efficiency and energy conservation. In the context of FPGA-based computation-intensive arithmetic multiplication, Logic Cloning (LC) is introduced to systematically induce controlled approximation. LC-Baugh Wooley (BW) circuits deliver exceptional error performance with precise approximation, while LC-Booth circuits are characterized by reduced Look-Up Table (LUT) resource consumption. In the case of 16-bit operands, LC methods effectively reduce LUT resource consumption by 31.05% for Booth and 36.85% for BW. Additionally, compared to their accurate counterparts, they lower the Power Delay Product (PDP) by 34% for Booth and 35% for BW. When it comes to symbol error-rate performance for Zero Forcing (ZF) Multiple Input Multiple Output (MIMO) uplink detection, these LC approximate multiplication circuits exhibit robust performance, particularly LC-BW circuits, which closely match the accuracy of ZF detection, followed by LC-Booth circuits.

1. Introduction

Modern applications involving multimedia and communication systems require extensive data processing and therefore need high computing resources. Also, energy dissipation has become a fundamental barrier to scale computing performance across multiple hardware computing platforms. To accommodate the requirement for next-generation computing applications, several techniques are put forth that leverage the existing hardware capability to provide demanding application performance, while simultaneously optimizing resource and energy efficiency. Approximate computing [1,2] is one such emerging technology that enables explicit control over the energy and hardware resource consumption for the intended application by incurring penalties in the system accuracy. The philosophy behind approximate computing is that application systems do not always have to provide the most accurate results for acceptable system performance [3].

In a logic circuit, the system functionality of the circuit is represented by a Boolean logic function. Logic circuits have been conventionally designed using deterministic logic gates with the input/output deterministic bit signals being logic '1' or '0'. The work [4] explores a pre-synthesis iterative approach to Application Specific Integrated Circuit (ASIC) approximation by pruning the logic gates based on the probability of active logic. Probabilistic bit signals are transformed to intended probabilities with deterministic combinational logic [5]. The characteristic of Boolean functions to output signal logic '1' and '0' with certain probability is exploited in [6] to form Probabilistic Boolean Logic (PBL) gate models. PBL employs implicit probabilistic logic gates. However, randomized circuits [7] use random input bits with deterministic gates for circuit output approximation, where the circuit operates with random inputs. Furthermore, gate-level methodologies are suited for ASIC implementation and cannot be ported for FPGA implementation because of architectural differences between both [8].

In reconfigurable computing, FPGA hardware implementation is characterized by adaptable hardware reconfiguration with uniform fundamental hardware structures, unlike ASIC hardware implementation. Resource and energy optimization of Register Transfer Level (RTL) circuits is a strategic task for both ASIC and FPGA implementations. At the RTL stage, ASIC circuits are composed of logic gates, while the FPGA circuits are composed of Configurable Logic Block (CLB), primarily comprising of LUTs, which are spread across the FPGA fabric. For a specific RTL architecture, ASIC hardware implementation is inherently about 35× efficient in terms of silicon area utilization, 14× efficient in terms of dynamic power consumption and about 4× faster than FPGA hardware implementation [8]. Hence, the ASIC implementations show asymmetric gains when they are ported to FPGA and vice versa. The

☆ This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), in part by Prompt, in part by the Canadian Foundation for Innovation (CFI), in part by CMC Microsystems, in part by OPAL-RT Technologies Inc. and in part by Hydro-Québec. Laboratoire des signaux et systèmes intégrés and Chaire de recherche sur les signaux et l'intelligence des systèmes haute performance (www.uqtr.ca/lssi).

* Corresponding author.
E-mail addresses: Abhinav.Kulkarni@uqtr.ca (A. Kulkarni), Messaoud.Ahmed.Ouameur@uqtr.ca (M.A. Ouameur), daniel.massicotte@uqtr.ca (D. Massicotte).

https://doi.org/10.1016/j.mejo.2024.106135
Received 30 October 2023; Received in revised form 14 January 2024; Accepted 18 February 2024
Available online 19 February 2024

proposed work is focused on the discussion of comparative techniques for FPGA hardware implementation. Therefore, *can the probabilistic nature of Boolean logic be utilized for inducing approximation in specific* FPGA *circuits?*

For the computationally intensive operation of arithmetic multiplication, optimal utilization of FPGA resources and power can be achieved either by efficiently implementing RTL of the circuit or by introducing approximation in the accurate RTL [9]. Dedicated heterogeneous ASIC DSP blocks are embedded in the FPGA fabric for efficient multiplication operation. However, these blocks improve the area and power consumption efficiency without providing any further scope for approximation based on the FPGA application [8]. Also, applications have to stringently adhere to the operand bit-width of the DSP block for significant efficiency gains. Hence, research on soft-core FPGA multiplication circuits is gaining ground.

1.1. Preliminary

The multiplication operation of multiplicand and multiplier generates Partial Product (PP)s, which are accumulated to produce the multiplication product. Carry Save Adder (CSA) structure is employed with a configuration like Wallace [10] or Dadda [11] for accumulation of PPs. Booth algorithm encodes the circuit operand bits which effectively reduces the total number of PPs required for the computation of the multiplication product. Unsigned multiplication involves unsigned operands with unsigned multiplication product, whereas signed multiplication is characterized by signed operands with signed multiplication product. Multiplier architectures have been adapted for FPGA by utilizing LUT and Carry Chain (CC) (comprising several Carry Chain Unit (CCU)s) primitives for the accurate multiplication operation. Guidelines for efficient implementation of accurate unsigned multiplication for FPGA implementation [12] are suggested in the form of efficient mapping and arithmetic transformation for improvement in the logic density of the implementation. However, this approach requires an explicit compressor tree for the accumulation of PPs and the implementation is suited for a 3-input LUT [13]. In the Multiply And Accumulate (MAC) architecture of FPGA multiplication circuits, the PP bits are simultaneously generated and accumulated using LUTs and CCUs. An accurate unsigned multiplication circuit [14] using Booth encoding eliminates the need for an explicit compressor tree for PPs accumulation by employing MAC operation and utilizes a contemporary 6-input LUT structure for FPGA implementation. However, its RTL implementation suffers from the under-utilization of LUT inputs for PP bit generation and is specifically intended for the unsigned multiplication operation.

For achieving signed multiplication using unsigned circuit implementation, the operands in two's complement format are segregated into sign and magnitude using sign-converters. The unsigned multiplication is performed with the magnitude of these operands and the multiplication product is converted back to two's complement by the sign-converter using signs of operands. Signed multiplication operation using sign-converters adds computational overhead for multiplication operation [15,16] in terms of increased LUT resource consumption. A smart approach for signed multiplication is by modifying the PPs generation by employing sign-extension for PPs using BW method, which avoids any kind of explicit sign conversion of operands. Also, the Booth encoding technique is modified for PPs generation [17], further reducing the number of PPs for signed multiplication.

1.2. Relevant work

Efficient implementation of an accurate signed multiplication circuit employing the Booth algorithm [16] reduced the CC by 3 CCUs for the generation of PPs, thereby optimizing the hardware resource consumption. However, an external hardware entity like an adder or a compressor was required for PPs accumulation and the accuracy of the circuit thus depends on the PP accumulation methodology utilized. Using the FPGA implementation of an accurate unsigned multiplication circuit [14], an architecture using LUTs and CCs for accurate signed multiplication circuit based on Booth encoding [18] known as Booth-Opt saved LUTs by embedding the logic of the carry generating rightmost LUT and the leftmost LUT into adjacent LUT in a PP row, thereby achieving a reduction in CC. An improvement in critical path delay of the work in [18] was presented in the further work [19] by optimizing the PPs generation and employing PP reduction tree, however, it was characterized by increased LUT resource consumption.

Performance gains were obtained by introducing approximations in accurate multiplication architectures. Approximations can be employed in PP accumulation by using FPGA implementation of approximate adder [20–22] or by approximate compressor [23], while the PPs are generated accurately. Configurable signed multiplication circuit architectures [24] were built using a combination of accurate and four approximate compressors for PPs accumulation. As these circuit architectures utilized explicit PP accumulation method, the LUT resource consumption was variable with circuit accuracy for a particular bit-width operation.

The functional approximation was introduced in the accurate circuit while generation of PPs. The Booth-Opt circuit was further modified for Booth-Approx [18] by approximating its PPs generation, specifically by approximating the carry signal generation at the rightmost LUT for all PPs, except the last PP. Booth-Approx was characterized by a reduction in CC chain and LUT resource consumption. However, Booth-Approx was non-configurable to achieve a trade-off with multiplication accuracy and energy efficiency. AxBM circuits [25] functionally modified the accurate Radix-8 Booth encoding for FPGA implementation to alleviate the generation of challenging 3×Multiplicand. The 3×Multiplicand was approximated by 4×Multiplicand in AxBM1 and AxBM2, while −3×Multiplicand was approximated by −4×Multiplicand in AxBM1 and −2×Multiplicand in AxBM2 respectively. For generating multiples of multiplicand for AxBM1 and AxBM2, 4× signal was generated by implicit XNOR operation of 1× and 2× signals in LUT of PP bit generation, thereby mapping the Booth encoder to a two output 5-input LUT configuration. However, AxBM1 and AxBM2 were characterized by non-MAC PPs accumulation and very poor multiplication product accuracy.

The presented work introduces a heuristic methodology designed to induce approximation in arithmetic multiplication circuits suitable for FPGA implementation at the pre-synthesis stage. The study distinguishes the logic structure from hardware entities and analyzes the circuit logic. The methodology relies on Truth Probability (TP) values, allowing for the creation of modular approximate multiplication circuits based on a single configurable parameter, in contrast to static approximate circuits in the works [18,25]. Additionally, the proposed approach employs a MAC architecture for PP accumulation, eliminating the need for explicit PP accumulation found in works [24,25]. In the work [24], the reduction stages for PP accumulation increase with the operand bit-width and have under-utilization of inputs for approximate compressors in certain instances, which is not the case for MAC architecture implementations.

1.3. Contributions

The following contributions are presented in the current work:

- Devise LC methodology for heuristically approximating the accurate multiplication circuits for FPGA implementation. In this pioneering work, LC methodology systematically utilizes the probability of logic '1' of the LUT logic structure output to induce approximation in the accurate circuit.
- Proposition of novel LC-BW and LC-Booth approximate circuits for signed multiplication. LC-BW circuits are characterized by high multiplication accuracy, their approximation being controlled with finer granularity, while LC-Booth circuits are characterized by low LUT resource consumption.

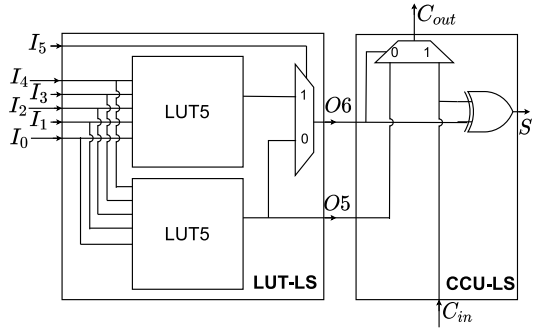

Fig. 1. LUT-LS and CCU-LS for Xilinx FPGA device [26].

- Analysis of proposed LC approximate circuits for LUT consumption and energy efficiency for 8-bit and 16-bit configurations with competitive signed approximate multiplication circuits. The approximation induced by LC methodology in accurate multiplication circuits is effective in reducing the LUT consumption and PDP.
- Evaluation of approximate signed circuit for symbol error-rate performance of MIMO uplink detection using ZF algorithm for 16-QAM and 64-QAM configuration. LC approximate signed circuits provides close to accurate ZF detection.

The paper is organized as follows; Section 1 discusses related work on approximate circuits techniques, motivating the need for softcore FPGA circuits. Section 2 explains the proposed methodology for the approximation of FPGA implementation of multiplication circuits in detail. Section 3 explains the application of the proposed methodology for signed circuits. Section 4 presents an error analysis of the proposed work with contemporary signed approximate multiplication circuits, while Section 5 discusses FPGA implementation analysis. Section 6 evaluates approximate signed multiplication circuits for MIMO uplink detection and discusses gains of the LC circuits for various MIMO configurations. Section 7 concludes the discussion by summarizing the LC methodology for FPGA implementation and its further potential.

2. Proposed methodology

2.1. Notations

$\mathcal{P}\{\mathcal{G}\}$ denotes the probability of occurrence of any event $\mathcal{G}$. The notation $|\mathbb{G}|$ denotes the cardinality of any finite set $\mathbb{G}$. Operators $\vee$ and $\wedge$ denote logical OR and AND operations. The symbol $\widehat{\ }$ denotes the approximate evaluation of the signal or value. Terms 'LUT', 'CC' and 'CCU' are used to denote the respective hardware entities henceforth, while the term with the suffix '-LS' denotes the logic structure of the corresponding hardware entity. The row of LUTs and CC which performs the MAC operation involving PP is denoted as MAC PP (MPP) for dexterity. $(.)^{(i)}$ denotes evaluation of any signal or value at the ith simulation step. $[.]_2$ denotes integer values represented in two's complement form.

2.2. FPGA hardware architecture

A FPGA device consists of CLBs spread across the FPGA fabric. For implementing arithmetic circuits, the carry logic is implemented using a dedicated CC chain comprising multiple CCUs. A CLB structure is comprised of slice units and each slice unit comprises four 6-input LUTs, a CC chain, and an associated circuit [26]. Every LUT is physically connected to a single CCU as shown in Fig. 1. For the Xilinx FPGA device, every LUT-LS can implement either two 5-input logic functions with shared inputs or a 6-input and 5-input logic function with shared inputs and shared logic values [27]. Every LUT is configured with a 64-bit hexadecimal value defining its characteristic LUT-LS.

In MAC architecture, MPPs are produced by rows of LUTs and CCs. The LUT-LS of the LUTs of MPP performs the MAC operation and the intermediate value produced is denoted as $\beta-$MPP value. The associated CC-LS of the CC chain of MPP transforms the $\beta-$MPP value into MPP value. The final MPP value is considered as the product of the multiplication operation. The $O6$ signal of the LUT-LS contributes a specific bit of the $\beta-$MPP value, while the S signal of the CCU-LS contributes a specific bit of the MPP value. A CCU-LS XORs the signal from LUT-LS with a prior carry signal and also computes the carry signal to be propagated to the next CCU-LS in the CC-LS chain. The CCU-LS as shown in Fig. 1 computes the following Boolean functions:

$$C_{out} = (\overline{O6} \wedge O5) \vee (C_{in} \wedge O6) \tag{1}$$

$$S = (C_{in} \wedge \overline{O6}) \vee (\overline{C_{in}} \wedge O6) \tag{2}$$

2.3. Probabilistic analysis

Signals of a FPGA multiplication circuit comprise of the inputs, interconnects and outputs. To investigate the potential for functional approximation in the circuit architecture, it is essential to develop a mathematical model to perform the probabilistic analysis of the circuit signals. The accurate multiplication circuit has inputs comprising of two N-bit signed operands and outputs forming one $2N$-bit multiplication product. Let the finite set $\mathbb{B}$ be defined such that $\mathbb{B} = \{0, 1\}$. Consider a FPGA multiplication circuit with $2N$ inputs, where each input is represented as $\psi_i \in \mathbb{B}$ for $i = 0, 1, \ldots, 2N$. The exhaustive simulation of the circuit comprises 2^{2N} unique operational states based on the input combinations. Let a particular simulation step in exhaustive simulation be denoted as ζ such that $0 \leq \zeta \leq 2^{2N} - 1$. A specific combination of $\psi_i^{(\zeta)}$ for a particular simulation step ζ is represented using a finite set $\mathbb{P}_\zeta = \{\psi_i^{(\zeta)} : 0 \leq i \leq 2N\}$. For the exhaustive simulation with $\mathbb{P}_\zeta$ for all ζ, the sample space is represented using a finite set $\mathbb{S} = \{\mathbb{P}_\zeta : 0 \leq \zeta \leq 2^{2N} - 1\}$.

Let $X \in \mathbb{B}$ represent a Boolean variable [28] for any signal of the circuit. The set of all such values of X due to exhaustive simulation is denoted by the finite set $\mathbb{X} = \{X^{(\zeta)} : 0 \leq \zeta \leq 2^{2N} - 1\}$. The number of logic '1' in the set $\mathbb{X}$ is $\mathbb{X}_1 = \{X^{(\zeta)} : 0 \leq \zeta \leq 2^{2N} - 1, X^{(\zeta)} = 1\}$. Let $[Y]_2$ be any arithmetic value computed in the circuit, which comprises of arithmetic combination of different signals. Then, $\mathbb{Y}$ denotes the finite set of values generated by $[Y]_2$ such that $\mathbb{Y} = \{[Y]_2^{(\zeta)} : 0 \leq \zeta \leq 2^{2N} - 1\}$.

The probability that X is logic '1' for the exhaustive simulation of the circuit is called the TP of X and evaluated as $\mathcal{P}\{X = 1\}$. The expression $\mathcal{T}(X)$ is used to imply $\mathcal{P}\{X = 1\}$ for dexterity. TP of X is computed as:

$$\mathcal{T}(X) = \frac{|\mathbb{X}_1|}{|\mathbb{X}|} = \frac{|\mathbb{X}_1|}{2^{2N}} \tag{3}$$

The arithmetic mean of X is computed as:

$$\mathcal{A}(X) = \sum_{X^{(\zeta)} \in \mathbb{X}} X^{(\zeta)} / |\mathbb{X}| \tag{4}$$

Similarly, the arithmetic mean of $[Y]_2$ is computed as:

$$\mathcal{A}([Y]_2) = \sum_{[Y]_2^{(\zeta)} \in \mathbb{Y}} [Y]_2^{(\zeta)} / |\mathbb{Y}| \tag{5}$$

But, since the sum of elements of $\mathbb{X}$ is equal to cardinality of $\mathbb{X}_1$:

$$\mathcal{T}(X) = \frac{|\mathbb{X}_1|}{|\mathbb{X}|} = \frac{\sum_{\zeta=0}^{2^{2N}-1} X^{(\zeta)}}{|\mathbb{X}|} = \mathcal{A}(X) \tag{6}$$

During circuit simulation, the calculation of the average value for any arithmetic value within the circuit is facilitated by Eq. (5). Eq. (6) allows for the computation of the TP value for any signal in the circuit.

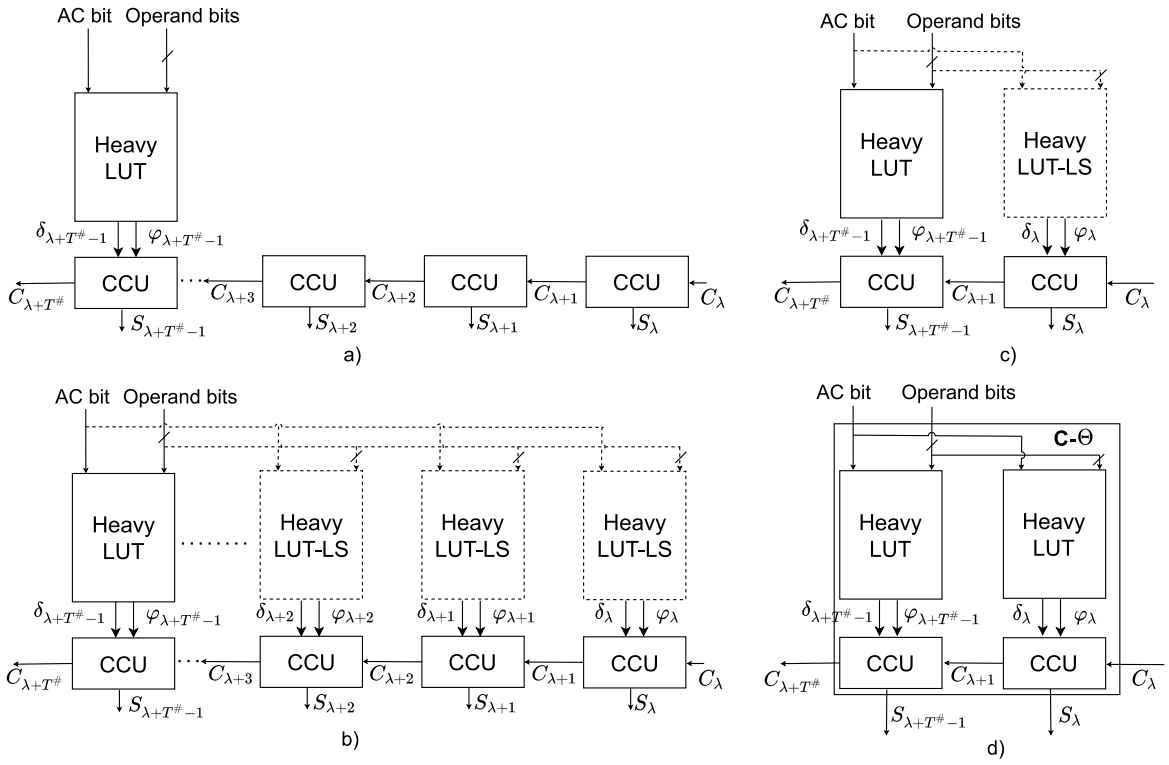

Fig. 2. (a) LUTs removed from MPP based on parameter $T^{\#}$ (b) Logic cloning of LUT-LS of Heavy LUT (c) CC chain reduction (d) C-Θ structure, where Θ represents LUT-LS configuration of Heavy LUT.

2.4. Error metrics

For evaluating approximate multiplication circuits, error metrics [29] capture the relative accuracy of the accurate and approximate multiplication product. For operands $[A]_2$ and $[B]_2$, $[\Phi]_2$ and $[\widehat{\Phi}]_2$ represent the accurate and approximate multiplication product respectively. $\Delta_\zeta = [\Phi]_2^{(\zeta)} - [\widehat{\Phi}]_2^{(\zeta)}$ computes the arithmetic difference between accurate and approximate multiplication product for the ζth simulation step. The sum of absolute errors in terms of the arithmetic mean of the multiplication product is computed using triangle inequality as:

$$\begin{aligned}\sum_{\zeta=0}^{2^{2N}-1} |\Delta_\zeta| &\geq |\sum_{\zeta=0}^{2^{2N}-1} \Delta_\zeta| \\ &= |\sum_{\zeta=0}^{2^{2N}-1} \Delta_\zeta| + Y \\ &= |\sum_{\zeta=0}^{2^{2N}-1} [\Phi]_2^{(\zeta)} - \sum_{\zeta=0}^{2^{2N}-1} [\widehat{\Phi}]_2^{(\zeta)}| + Y \\ &= |\mathcal{A}([\Phi]_2) - \mathcal{A}([\widehat{\Phi}]_2)| 2^{2N} + Y \end{aligned} \tag{7}$$

where $Y \geq 0$ is an arbitrary constant.

Mean Error Distance (MED) considers the averaging effect of the sum of absolute errors over the complete operand range, which is useful in measuring the implementation accuracy of the approximate circuit.

$$MED = \frac{1}{2N} \sum_{\zeta=0}^{2^{2N}-1} |\Delta_\zeta| \tag{8}$$

Relative Error Distance (RED) computes the relative error distance of an approximate multiplication product with respect to its accurate multiplication product (RED is valid only for non-zero accurate multiplication output). Mean Relative Error Distance (MRED) measures the arithmetic mean of RED for all valid accurate multiplication products.

$$MRED = \frac{1}{2N} \sum_{\zeta=0}^{2^{2N}-1} \frac{|\Delta_\zeta|}{|[\Phi]_2^{(\zeta)}|} \tag{9}$$

RED and MRED are useful in evaluating approximate circuit designs with varying operand ranges. Normalized Mean Error Distance (NMED) is a metric that normalizes MED with the maximum value of the accurate multiplication product over the operand range. Also, the maximum relative error (Max Rel) of an approximate multiplier circuit is a useful metric that pinpoints the worst-case operation of the approximate circuit design.

2.5. Logic cloning

Exhaustive simulation involves simulating the accurate multiplication circuit with a range of 2^{2N} distinct input combinations. Consider bit signals $a_i, b_i, \phi_i \in \mathbb{B}$ for $i = 0, 1, \ldots, N-1$. For signed multiplication, N bit operand multiplicand and circuit are represented in base-2 notation as $[A]_2 = -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i$ and $[B]_2 = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$ respectively, while the multiplication product is denoted as $[\Phi]_2$. The accurate multiplication circuit is prototyped and verified for hardware implementation by mapping the multiplication architecture to the FPGA architecture. The input bit signals of the circuit architecture are mapped such that $\psi_i \leftarrow a_i$ when $0 < i \leq N-1$ and $\psi_i \leftarrow b_i$ when $N < i \leq 2N-1$. The verified FPGA circuit is modeled in the C environment to compute the TP value of the $O6$ signal of the LUT-LSs of the accurate multiplication circuit. The LUT-LS function blocks are constructed to represent the Boolean logic function by using LUT-LS hexadecimal configuration value. The CCU-LS function blocks are constructed using the Eqs. (1) and (2). The FPGA interconnects are represented with an indexed 2-dimensional array structure. During exhaustive simulation, the occurrence of logic '1' at every $O6$ signal is recorded to compute its TP using Eq. (3).

For any nth β-MPP of the accurate circuit, the $O6$ signal of the LUT-LS is mapped to φ signal such that $\varphi \leftarrow O6$. The $O5$ signal of the

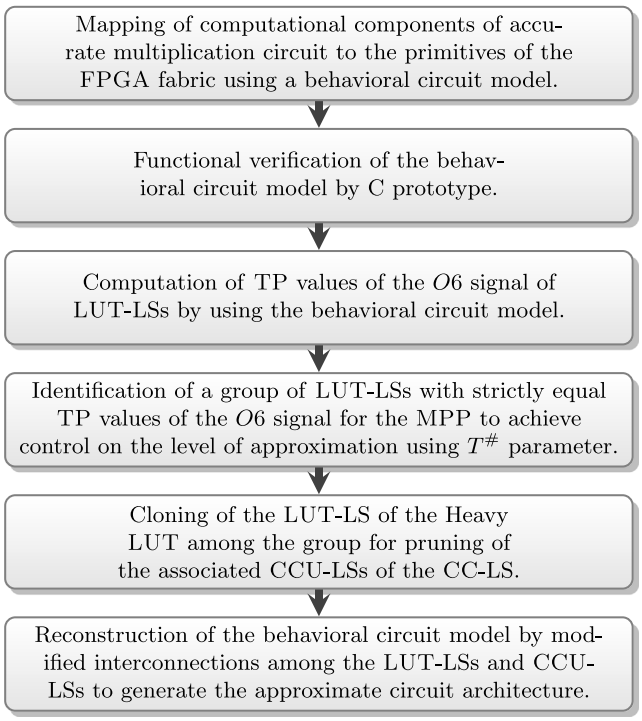

Fig. 3. LC methodology for approximation in FPGA implementation of multiplication circuit as explained in Section *2.6*.

LUT-LS is mapped to δ signal such that $\delta \leftarrow O5$, whose only function is to provide the Accumulator (AC) bit supplied to the LUT-LS for MAC operation to the CCU-LS. For any nth $\beta-$MPP of the circuit, let $\mathcal{T}(\varphi_\lambda) = \mathcal{T}(\varphi_{\lambda+1}) = \cdots \mathcal{T}(\varphi_{\lambda+T-1})$ for any $\lambda > 0$ for T LUT-LSs. Out of T LUT-LSs, with a group of $T^{\#}$ LUT-LSs such that $0 \leq T^{\#} \leq T$, the arithmetic mean value of $\beta-$MPP can be effectively approximated with only LUT-LS corresponding to $\varphi_{\lambda+T^{\#}-1}$ signal as given in Lemma. Thus, the $O6$ signal value of $(T^{\#}-1)$ LUT-LSs are redundant to approximate the arithmetic mean of $\beta-$MPP. The LUT corresponding to $\varphi_{\lambda+T^{\#}-1}$ signal is termed as Heavy LUT and $(T^{\#}-1)$ LUTs are removed in the MPP row. Parameter $T^{\#}$ controls the amount of approximation of the arithmetic mean of $\beta-$MPP, with $T^{\#} = 0$ resulting in no approximation, while $T^{\#} = T$ resulting in highest amount of approximation. Approximation in every nth $\beta-$MPP contributes to inducing approximation in the arithmetic mean value of the final multiplication product.

The approximation of the nth $\beta-$MPP makes several LUTs redundant thereby leaving CCU-LSs with dangling input signals as shown in Fig. 2a, which are interlinked to propagate the carry signal. To analyze the redundancy in the CC-LS of the nth $\beta-$MPP, the Heavy LUT-LS is cloned as shown in Fig. 2b. Consider $T^{\#}$ CCU-LSs of the corresponding $T^{\#}$ LUT-LSs with equal TP. δ_i and φ_i represent inputs to the ith CCU, while C_i and C_{i+1} represent the C_{in} and C_{out} respectively. Due to logic cloning procedure, $\varphi_\lambda = \varphi_{\lambda+1} = \cdots = \varphi_{\lambda+T^{\#}-1} \equiv \varphi_{\#}$. For nth MPP, if $\delta_\lambda = \delta_{\lambda+1} = \cdots = \delta_{\lambda+T^{\#}-1} \equiv \delta_{\#}$, then in such case, the Eqs. (1) and (2) for λth CCU-LS become:

$$C_{\lambda+1} = (\overline{\varphi_{\#}} \wedge \delta_{\#}) \vee (C_\lambda \wedge \varphi_{\#}) \tag{10}$$

$$S_\lambda = C_\lambda \oplus \varphi_{\#} \tag{11}$$

For evaluating the $(\lambda+2)$th carry signal, the Eq. (10) becomes:

$$\begin{aligned} C_{\lambda+2} &= (\overline{\varphi_{\#}} \wedge \delta_{\#}) \vee (C_{\lambda+1} \wedge \varphi_{\#}) \\ &= (\overline{\varphi_{\#}} \wedge \delta_{\#}) \vee (((\overline{\varphi_{\#}} \wedge \delta_{\#}) \vee (C_\lambda \wedge \varphi_{\#})) \wedge \varphi_{\#}) \\ &= (\overline{\varphi_{\#}} \wedge \delta_{\#}) \vee (C_\lambda \wedge \varphi_{\#}) \end{aligned} \tag{12}$$

It is inferred from Eqs. (10) and (12) that $C_{\lambda+1} = C_{\lambda+2} = \cdots = C_{\lambda+T^{\#}}$ and $C_{\lambda+T^{\#}}$ can be equated with $C_{\lambda+1}$. Hence, from Eq. (11), it is also inferred that $S_{\lambda+1} = S_{\lambda+2} = \cdots = S_{\lambda+T^{\#}}$. Thus, CCU-LS for computing C_{i+1} and S_i for $\lambda+1 \leq i \leq \lambda+T^{\#}-2$ can be deemed redundant and the corresponding CCUs are pruned as shown in Fig. 2c. However, every LUT is physically connected to a single CCU, hence an additional LUT is required to supply the δ_λ and φ_λ signals as shown in Fig. 2d.

2.6. Overview of LC methodology

The LC methodology for approximating FPGA implementation of multiplication circuits is outlined in Fig. 3. In the initial phases of the multiplication algorithm to architecture mapping, an apt behavioral circuit model intended for implementation on an FPGA is developed, integrating LUT and CC primitives. The behavioral circuit model is prototyped in C by utilizing the logic structure of the primitives. The model undergoes functional verification to guarantee accurate behavior as intended by the multiplication algorithm. Through this careful verification, it is ensured that the chosen combination of primitives effectively captures the desired computational logic of the multiplication algorithm. Subsequently, the approximation process commences by calculating TP values for the output signal $O6$ within the LUT-LSs. Specifically, the TP values are computed by using circuit inputs through analysis as outlined in Section 2.3 using the behavioral circuit model. Upon analyzing the TP values of the MPPs of the behavioral circuit model, a set of LUT-LSs with identical TP values is identified for each MPP. Using a single parameter $T^{\#}$, a selective removal of a specified number of LUT-LS is employed. This results in corresponding CCU-LSs dangling without any input signals. Hence, a redundancy analysis is initiated as the Heavy LUT-LS is cloned, enabling a detailed examination of redundancy in the CC-LS. This process results in the subsequent removal of redundant CCU-LSs.

The behavioral circuit model obtained at this stage comprises LUT-LS and CCU-LS barring the redundant ones. The interconnects obliterated by redundant logic structures are modified, ensuring adherence to the physical constraints imposed by the FPGA architecture. The methodology generates various approximate circuit versions for multiplication based on the $T^{\#}$ parameter, providing a range of trade-offs between computational accuracy and resource utilization.

3. Logic cloning for signed multiplication circuits

3.1. Logic cloning for BW circuit

For the computation of the signed multiplication of two operands without the requirement of sign converters, the BW method is an efficient method for generating N PPs, which are accumulated to generate the final product $[\Phi]_2$. For BW architecture, LUT-LSs are configured for implementing two 5-input Boolean functions as shown in Fig. 4. LUT-LS of L configuration as shown in Fig. 5a performs the AND logic operation to produce a PP output using operand bits and performs its XOR logic operation with the AC bit to produce the $O6$ signal, comprising the $\beta-$MPP bit.

The functionality of controlled negation logic operation of the AC bit and its XOR logic operation with the output of NAND logic operation of the operand bits is configured for LUT-LS of M configuration as shown in Fig. 5b. Constant logic '1' signal is provided by LUT-LS of J configuration as shown in Fig. 5c. Using LUT-LS configurations of L, M and J, FPGA implementation of BW circuit for 4-bit and 8-bit multiplication operation as shown in Figs. 6 and 7 respectively are constructed. Considering $O6$ signal of LUT-LS of the BW multiplication circuit as X, TP value of this signal for every LUT-LS is evaluated using Eq. (6).

For the application of LC methodology for approximation in BW circuit, TP of LUT-LS of L, M and J configuration for 4-bit and 8-bit BW circuit are shown in Fig. 6 and Fig. 7 respectively. For a particular MPP row, all LUT-LSs with L configuration with an equal TP pose potential for the LC methodology. For every MPP_i where $0 \leq i < N-1$,

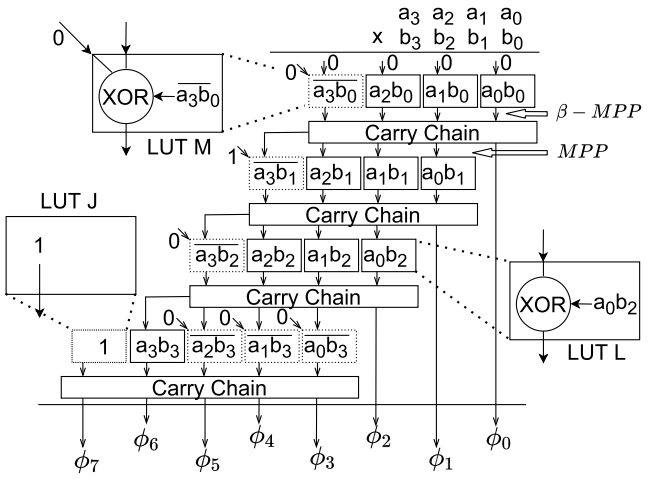

Fig. 4. Mapping of MPPs of signed multiplication circuit with the LUT-LS configurations using BW method for 4-bit operands. PP output is added using XOR operation in LUT-LS of L configuration. LUT-LS of M configuration employs MAC operation for negated PP output and controlled negation of AC bit. LUT J is utilized for supplying logic '1' signal for the most significant MPP bit of the last MPP.

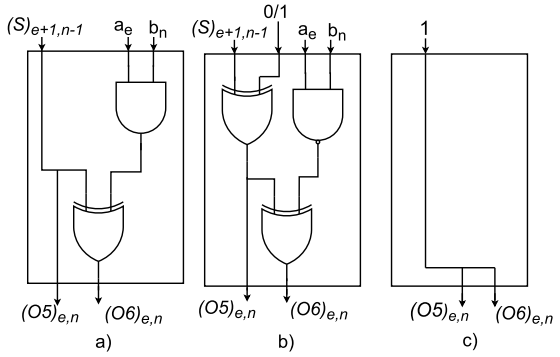

Fig. 5. LUT configuration for BW circuit with (a) L (b) M and (c) J.

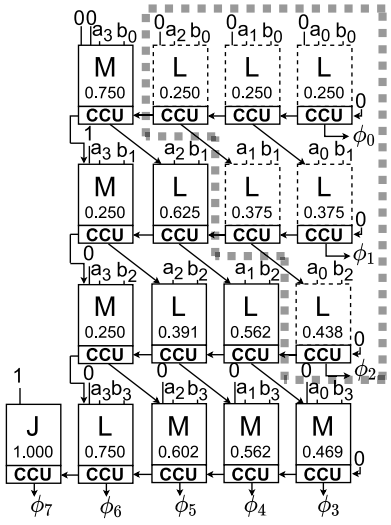

Fig. 6. Signed multiplication circuit of BW circuit for 4-bit operands. TP for $O6$ signal of LUT-LS of every LUT shown on its depiction.

$T = N - i - 1$ LUT-LSs of L configuration have equal TP with $\lambda = 0$. In version 1 of the approximate BW circuit, the LC methodology is applied by considering $T^{\#} = T$, resulting in saving of $(T-2)$ LUTs per MPP of the

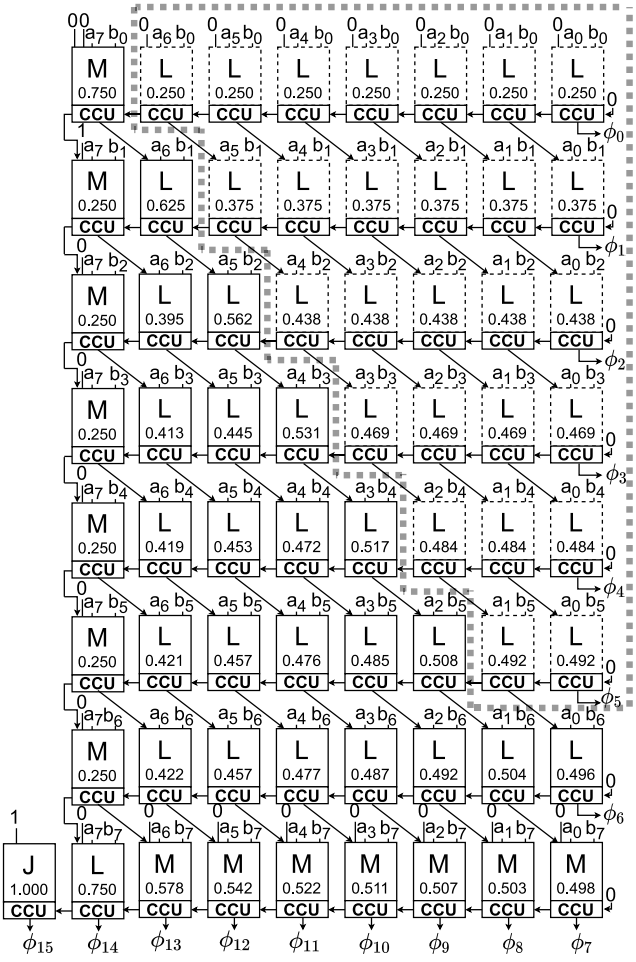

Fig. 7. BW circuit for signed multiplication with 8-bit operands.

accurate BW circuit, while the $(T-1)$th LUT is considered as the Heavy LUT. Version 1 of the approximate BW circuit is represented as LC-BW-1, the 8-bit LC-BW-1 shown in Fig. 8a. In version 2 of the approximate BW circuit, LC methodology is applied by considering $T^{\#} = T - 1$, thereby saving $(T-3)$ LUTs per MPP of the accurate BW circuit, while the $(T-2)$th LUT is considered as the Heavy LUT. Version 2 of the approximate BW circuit is represented as LC-BW-2, the 8-bit LC-BW-2 shown in Fig. 8b.

3.2. Logic cloning for Booth circuit

In BW multiplication, N PPs of the multiplier are computed for operands with N bits. It is challenging to efficiently utilize LUT inputs for FPGA implementation of the Boolean functions involved in the BW circuit. A more efficient method of signed multiplication circuit, the Booth algorithm [17] encodes the operand multiplier bits, such that the number of PPs required for multiplication product is reduced. In Radix-4 Booth encoding [17], the operand circuit is encoded as follows:

$$[B]_2 = \sum_{i=0}^{N/2-1} [B_i']_2 2^{(2i)} \tag{13}$$

where $[B_i']_2$ for the ith bit position is represented as:

$$[B_i']_2 = -2b_{2i+1} + b_{2i} + b_{2i-1} \tag{14}$$

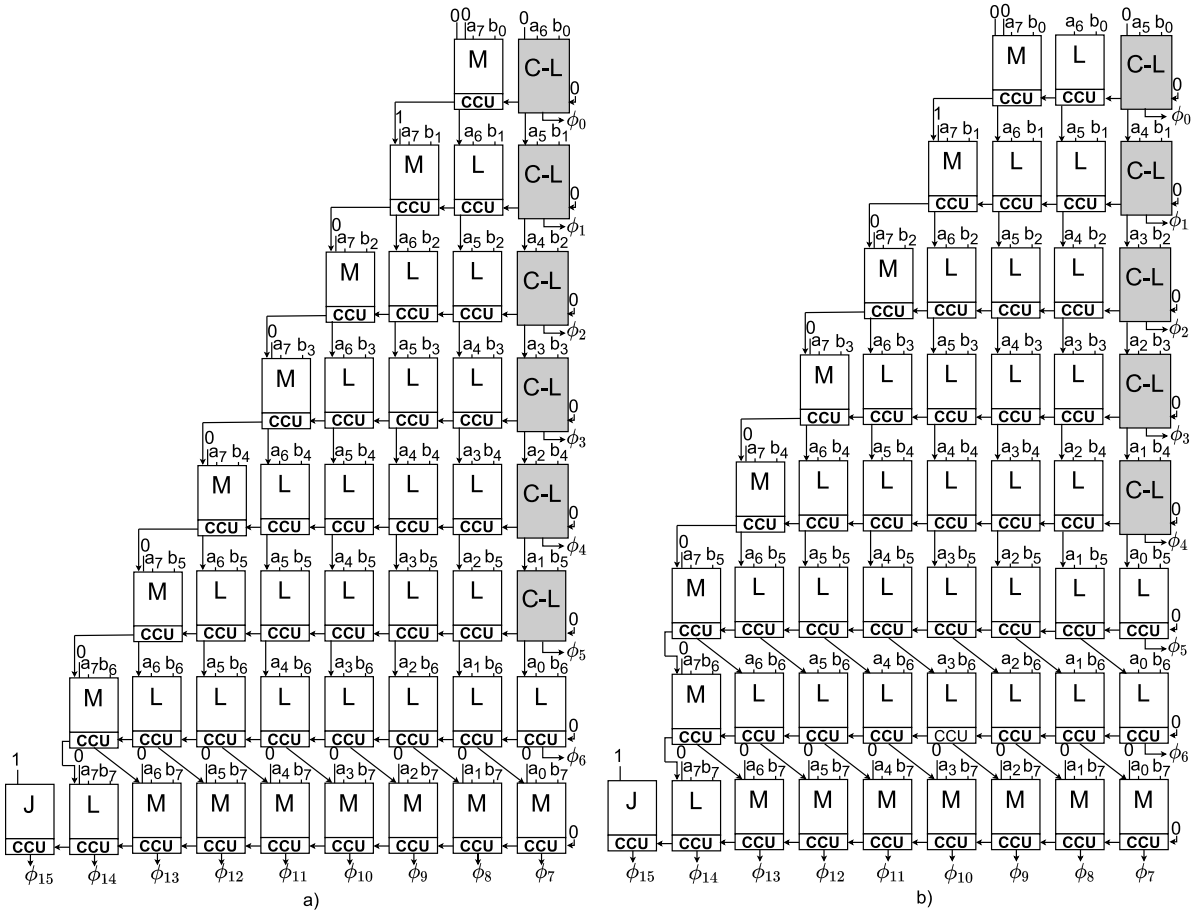

Fig. 8. LC-BW circuits for 8-bit operands (a) LC-BW-1 and (b) LC-BW-2.

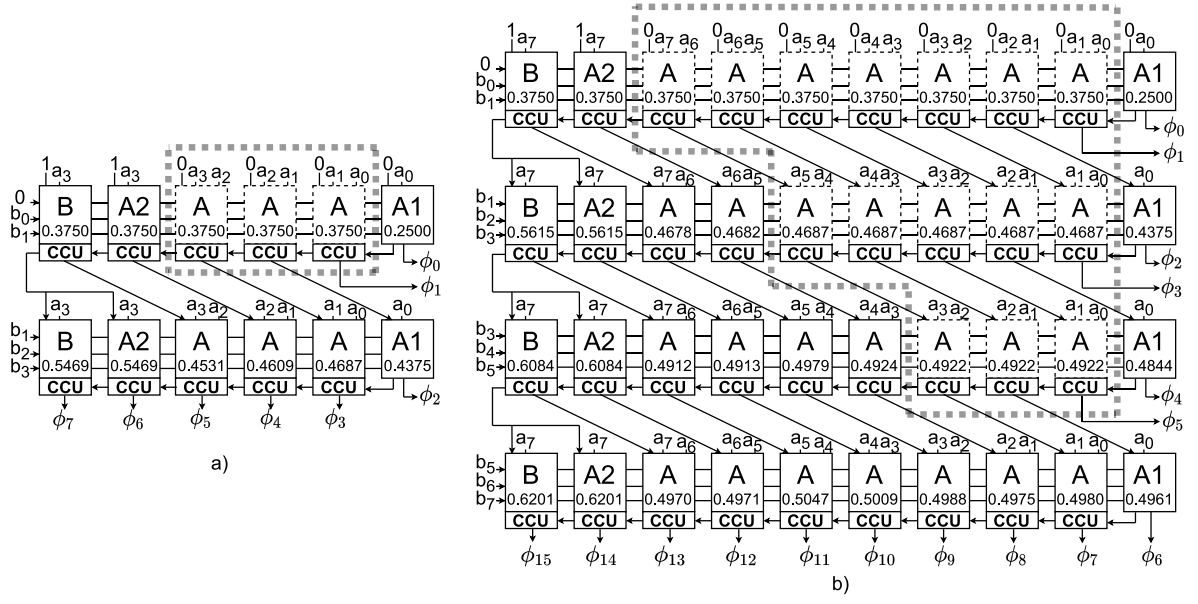

Fig. 9. Signed multiplication circuit of Booth-Opt [18] circuit with TP of $O6$ signal of LUT-LSs evaluated for (a) 4-bit (b) 8-bit operation.

The product is calculated as :

$$[\Phi]_2 = \sum_{i=0}^{N/2-1} [A]_2[B_i']_2 2^{(2i)} \tag{15}$$

Booth-Opt [18] is utilized as FPGA implementation of a signed multiplication circuit utilizing the Booth algorithm to demonstrate the applicability of LC methodology.

For every LUT-LS of the Booth-Opt circuit, TP value of the $O6$ signals are computed using Eq. (6) by considering $O6$ signal as X for analysis.

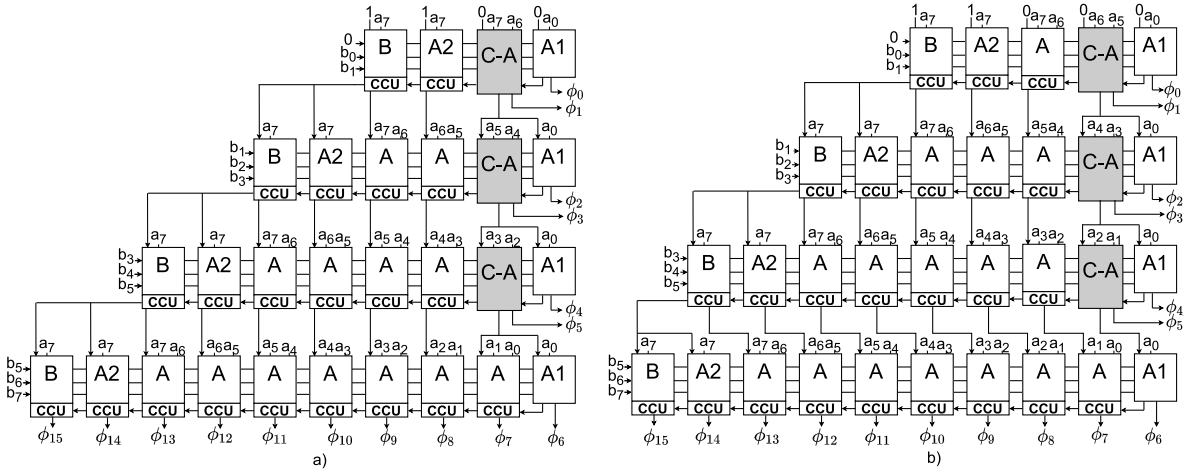

Fig. 10. LC-Booth circuits for 8-bit operands (a) LC-Booth-1 (b) LC-Booth-2.

Table 1
Evaluation of approximate circuits for multiplication product accuracy.

Multiplier	$N = 8$				$N = 16$			
	MED	MRED	NMED	Max Rel	MED	MRED	NMED	Max Rel
MUL_14_4[a] [24]	512	0.9393	0.0313	2118	7.007+e6	0.0301	652.569−e5	1.000
MUL_14_6[a] [24]	100	0.2317	0.0061	1094	2.765+e6	0.0169	257.543−e5	1.000
MUL_13_4[a] [24]	647	1.2014	0.0395	2118	7.518+e6	0.0333	700.189−e5	1.000
MUL_13_6[a] [24]	112	0.2486	0.0068	838	3.270+e6	0.0200	304.518−e5	1.000
Booth-Approx [18]	89	0.0949	0.0054	6	0.023+e6	0.0010	2.146−e5	0.097+e6
AxBM1 [25]	1229	2.2104	0.0750	72	294.880+e6	11.4910	27 470.000−e5	0.037+e6
AxBM2[b] [25]	1200	2.1893	0.0732	72	286.675+e6	11.4473	26 698.000−e5	0.037+e6
LC-BW-1	37	0.0584	0.0022	63	0.015+e6	0.0011	1.357−e5	0.016+e6
LC-BW-2	16	0.0279	0.0010	31	0.007+e6	0.0058	0.655−e5	0.008+e6
LC-Booth-1	82	0.0830	0.0050	2	0.034+e6	0.0016	3.184−e5	3.000
LC-Booth-2	37	0.0451	0.0023	2	0.017+e6	0.0009	1.543−e5	3.000

[a] 16-bit variant is built using a combination of M8s variants.
[b] For 16-bit, nine LSBs are truncated and '1' is added to the tenth bit for error compensation.
Proposed approximate multiplication circuits are shown in boldface.

TP values for LUT-LSs are evaluated for Booth-Opt circuit as shown in Fig. 9a and Fig. 9b for 4-bit and 8-bit multiplication operations respectively. The TP for $O6$ signal is constant across a chunk of LUT-LSs with A configuration for every MPP row. Precisely, for MPP_i with $i = 0, 1, \ldots N/2 - 1$ for $\lambda = 1$, about $T = N - 1 - 2i$ LUT-LSs have equal TP. For version 1 of the approximate Booth circuit, LC methodology is applied to the accurate Booth circuit by considering $T^{\#} = T$ LUT-LSs of A configuration for every MPP, thereby saving $(T - 2)$ LUTs and CCUs per MPP. The Tth LUT is considered as the Heavy LUT for each MPP. Version 1 of the approximate Booth circuit is represented as LC-Booth-1. For version 2 of the approximate Booth circuit, LC methodology is applied to the accurate Booth circuit by considering $T^{\#} = (T - 1)$ LUT-LSs of configuration A for every MPP, thereby saving $(T - 3)$ LUTs and CCUs per MPP. The $(T - 1)$th LUT of MPP is considered as the Heavy LUT. Version 2 of the approximate Booth circuit is represented as LC-Booth-2. Both versions of the 8-bit LC based circuits are shown in Fig. 10a and Fig. 10b.

4. Simulation results

Competitive circuit designs are evaluated as per the error metrics in Table 1. For signed multiplication of N bit operands, the operand range is $[-2^{N-1}, 2^{N-1} - 1]$, while the multiplication product occupies $2N$ bits. As the error metrics are dependent on the operand range of the circuit according to Eqs. (8) and (9), the error performance of the approximate multiplication circuits is also dependent on the operand range. For the 8-bit signed configuration of approximate compressor based multiplication circuits [24], PPs are generated using the BW method. For 16-bit signed configuration, four 8-bit unsigned approximate multiplication circuits (built using the approximate compressors) are utilized while the sign computation is managed by the additional 17th bit of the PP. The signal ordering of the approximate compressors affects the error performance of the circuit. Booth-Approx [18] is approximated from the FPGA implementation of the Booth-Opt. For AxBM [25], the Booth encoding of accurate Radix-8 is modified for AxBM1 and AxBM2 encoders.

Table 2
LUT resource consumption of accurate and proposed LC circuits for N bit operands (N is even).

Multiplier	#LUT
BW	$(N^2 + 1)$
BOOTH-OPT	$(N^2 + 2N)/2$
LC-BW-1	$(N^2 + 5N - 4)/2$
LC-BW-2	$(N^2 + 7N - 10)/2$
LC-Booth-1	$(N^2 + 8N - 4)/4$
LC-Booth-2	$(N^2 + 10N - 8)/4$

For the 8-bit configuration, the LC-BW circuits have the lowest MED, MRED and NMED among all the circuits, with LC-BW-2 of LC-BW being the lowest than LC-BW-1. LC-Booth circuits show the best error performance after LC-BW circuits, where LC-BW-2 performs better than LC-BW-1. However, the maximum relative error is lower for the LC-Booth circuits than that for the LC-BW circuits. Booth-Approx performs best after LC-BW and LC-Booth circuits for the MED, MRED and NMED error metrics, its maximum relative error lies in between that of LC-BW and LC-Booth. Approximate compressor based multiplication circuits

(MUL_xy_k; where x and y denote the approximate compressors used and k denotes the approximation factor) perform better after Booth-Approx, LC-BW and LC-Booth circuits. AxBM1 and AxBM2 have the lowest error performance, AxBM2 performs better than AxBM1. For circuits with 16-bit operands; MED, MRED and NMED follow a similar pattern in error performance as the circuits with 8-bit operands. The maximum relative error varies unevenly among the circuits. It decreases with an increase in bit-width for multiplication circuits using approximate compressors. However, it increases for AxBM1, AxBM2, Booth-Approx, LC-BW and LC-Booth circuits.

5. Hardware implementation

For demonstrating the hardware evaluation of approximate circuits, Xilinx Vivado 2021.1 tool is used for synthesis and implementation, with XILINX VIRTEX-7 FPGA device xc7vx690tffg1930-3. FPGA circuits are prototyped in VHSIC Hardware Description Language (VHDL). For RTL synthesis, LUT6_2 and CARRY4 primitives are used as components. Critical Path Delay (CPD) is the longest path delay in the circuit, measured between the input and output signal of the circuit. The signal propagation delay of LUT is independent of its LUT-LS configuration.

Conventional multiplication circuits are combinational in nature, hence the logic delay of the circuit is clock invariant and the circuit delay must be lesser than the clock cycle period. However, for estimating the routing delay, the design tool optimizes the critical path based on the timing constraint of the inputs. Hence, the timing constraint is adjusted to get the best possible delay value of the critical path for a particular circuit configuration. To get the precise CPD value, initially, the timing slack value is nullified at the synthesis level by calibrating the timing constraint. In the next stage, implementation is initiated with the timing constraint obtained at the synthesis level for nullifying the timing slack. At the subsequent iterations of the synthesis/implementation, the timing constraint is updated with the data path delay from the previous iteration of synthesis/implementation. This process is repeated until the design tool provides a relatively minimum value for the data path delay, to be considered as CPD.

For power calculation, simulation configuration with a supply voltage of 1 V with Linear Feet per Minute (LFM) of 250 along with a heat sink is considered. Precise dynamic power values are computed by implementing multiple instances of RTL circuit design. The logic delay time of the Xilinx IP [30] in the area-optimized mode is chosen as the clock period to evaluate the power consumption of circuits.

The LUT consumption of non-MAC circuits of AxBM and approximate compressor based multiplication circuits not only depends on the PP generation but also on the PP accumulation method. However, the MAC circuits have a comprehensive architecture with implicit PP accumulation. LUT resource consumption of accurate BW and Booth circuits and their derived accurate LC approximate circuits for N bit operands are depicted in Table 2. The efficiency of LUT resource consumption of MAC approximate multiplication circuits is shown in Fig. 11. In terms of LUT savings, LC-BW circuits show a logistic growth, while the Booth-Approx shows an exponential decay with an increase in the circuit bit-width N. LC-BW circuits achieve more LUT savings as compared to LC-Booth circuits. Also, version 1 of LC-BW and LC-Booth shows more LUT saving than version 2.

To evaluate the performance of LC methodology, circuit configurations are shown in Table 3. As compared to the accurate BW circuit, the gains in LUT consumption for LC-BW-1 are 5.88% for 4-bit, 23.07% for 8-bit and 35.40% for 16-bit configuration. LUT consumption of LC-BW-2 for 4-bit configuration is similar to that of accurate BW circuit, while the gains are 15.38% for 8-bit and 30.40% for 16-bit configuration. LUT consumption for LC-BW-1 is lower than that for LC-BW-2 for similar configurations. Similarly, as compared to the accurate Booth circuit, gains in LUT consumption for LC-Booth-1 are 8.33% for 4-bit, 32.5% for 8-bit and 34.02% for 16-bit configuration. LUT consumption of LC-Booth-2 for 4-bit configuration is equal to that of the accurate Booth

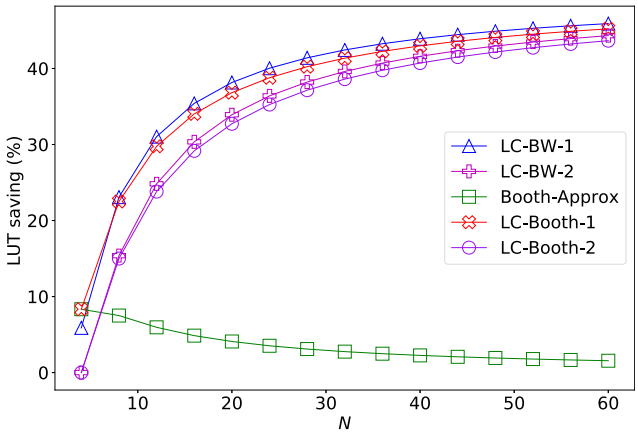

Fig. 11. LUT saving due to induced approximation of MAC signed approximate circuits.

circuit, while gains in LUT consumption are 15% for 8-bit and 29.17% for 16-bit configuration. Gains in LUT resource consumption increase with an increase in bit-width for all approximate circuits.

The Power Delay Product (PDP) of a circuit is the product of its CPD and power consumption. For 4-bit circuit configurations, in terms of PDP efficiency, LC approximate circuits have similar PDP efficiency compared to the accurate BW circuit, while LC-Booth-1 and LC-Booth-2 circuits are 20% and 3% more efficient than the accurate Booth circuit. For 8-bit circuit configuration, LC-BW-1 and LC-BW-2 show an efficiency of 23% and 4% in PDP respectively as compared to accurate BW circuit, while LC-Booth-1 and LC-Booth-2 show an average of 18% efficiency in PDP as compared to accurate Booth circuit. For 16-bit circuit configuration, LC-BW-1 and LC-BW-2 show an average of 36.5% PDP efficiency as compared to the accurate BW circuit, while LC-Booth-1 and LC-Booth-2 show an average of 29% PDP efficiency as compared to accurate Booth circuit. Efficiency in PDP increases with an increase in bit-width for all circuit configurations.

On comparing LUT consumption for competitive approximate multiplication circuits for 8-bit and 16-bit configurations, it is lowest for LC-Booth circuits, followed by Booth-Approx. AxBM1 and AxBM2 have comparable LUT consumption with LC-BW circuits, for 16-bit circuit configuration. For the 8-bit circuit configuration, the CPD of multiplication circuits with approximate compressors is higher than that of Booth-Approx and does not increase drastically with an increase in the bit-width of the operands. CPD performance of LC-circuits is comparable to multiplication circuits utilizing approximate compressors, however, the former's CPD increases more drastically as compared to the latter on increasing the bit-width. For the 16-bit circuit configuration, the low power consumption is more critical than low CPD in lowering the PDP of multiplication circuits with approximate compressors as compared to the PDP of AxBM1 and AxBM2 circuits. PDP of LC-Booth circuits lies in between that of multiplication circuits with approximate compressors and Booth-Approx. LC-BW circuits are characterized by the highest CPD, compensated by their better error performance.

Fig. 12 captures the relationship between PDP, NMED and LUT resource consumption of approximate multiplication circuits. LC-Booth circuits have the lowest LUT consumption with average PDP and low NMED. LC-BW circuits have the lowest NMED, average LUT consumption and highest PDP. LUT consumption and PDP increase while NMED decreases with the version of approximation for LC-Booth and LC-BW circuits. While having a comparable NMED with an LC-based approximate circuit, Booth-Approx has an average LUT consumption and PDP. Multiplication circuits with approximate circuits (MUL_xy_k) are characterized by the lowest PDP and high LUT consumption. AxBM1 and AxBM2 circuits have an overall average PDP and LUT consumption, however, have an extremely high NMED for 16-bit operands.

Table 3
Hardware implementation analysis of signed approximate multiplication circuits on XILINX VIRTEX-7 FPGA device. CPD in ns, Power in W and PDP in pJ.

	Multiplication circuits	$N = 4$				$N = 8$				$N = 16$			
		#LUT	CPD	Power	PDP	#LUT	CPD	Power	PDP	#LUT	CPD	Power	PDP
Accurate	Xilinx(Speed) [30]	19	1.85	0.34	0.63	73	2.76	1.35	3.73	281	3.73	3.70	13.78
	Xilinx(Area) [30]	25	2.82	0.60	1.69	86	3.41	1.95	6.66	314	5.03	5.50	27.69
	BW	17	2.71	0.23	0.63	65	5.67	1.20	6.80	257	11.64	3.55	41.31
	Booth-Sign-1 [16]	18	1.65	0.68	1.13	66	2.80	2.36	6.60	243	4.48	5.92	26.52
	Booth-Sign-2 [19]	14	2.59	0.51	1.31	54	4.37	1.66	7.26	208	5.28	5.90	31.14
	Booth-Opt [18]	12	2.31	0.26	0.60	40	4.13	1.00	4.13	144	7.96	2.75	21.90
Approximate	MUL_14_4[a] [24]	–	–	–	–	48	4.15	0.30	1.25	208	5.85	1.30	7.60
	MUL_14_6[a] [24]	–	–	–	–	59	4.65	0.43	2.01	212	5.85	1.32	7.72
	MUL_13_4[a] [24]	–	–	–	–	45	3.92	0.26	1.02	163	5.62	1.00	5.62
	MUL_13_6[a] [24]	–	–	–	–	57	4.57	0.35	1.60	176	5.63	1.04	5.86
	Booth-Approx [18]	–	–	–	–	37	3.41	1.24	4.23	137	6.88	2.78	19.13
	AxBM1 [25]	–	–	–	–	–	–	–	–	194	3.68	4.90	18.03
	AxBM2 [25]	–	–	–	–	–	–	–	–	161	3.45	4.12	14.21
	LC-BW-1	16	2.76	0.23	0.64	50	5.20	1.00	5.20	166	10.65	2.45	26.10
	LC-BW-2	17	2.72	0.23	0.63	55	5.39	1.00	5.39	179	10.84	2.45	26.55
	LC-Booth-1	11	2.08	0.23	0.48	27	4.04	0.85	3.43	95	7.95	1.90	15.10
	LC-Booth-2	12	2.23	0.26	0.58	34	3.93	0.85	3.34	102	8.22	1.90	15.61

[a] Clock frequency of 100 MHz.

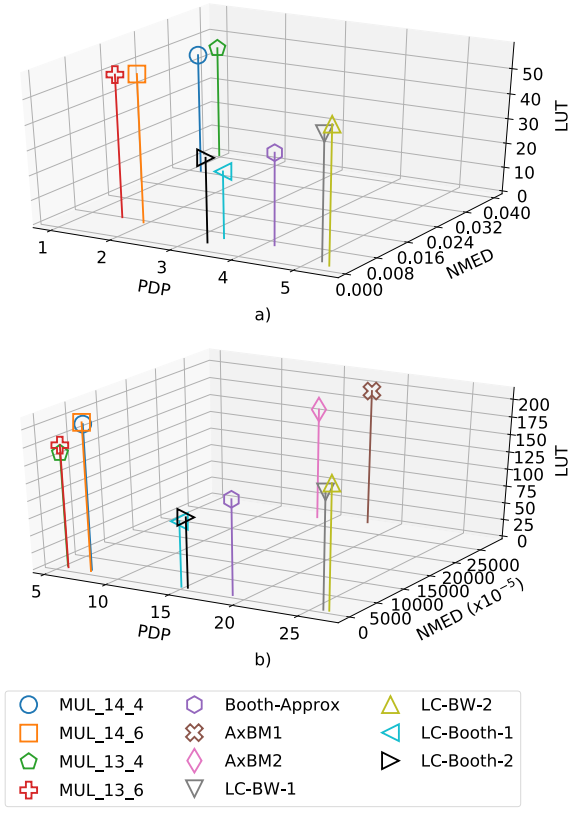

Fig. 12. Comparative analysis of PDP, NMED and LUT resource consumption for (a) 8-bit (b) 16-bit configuration. PDP is given in pJ.

6. Case study: Massive MIMO detection

Being a promising concept for future cellular networks, massive MIMO has been at the forefront in substantially improving both spectral and energy efficiencies of communication systems. A MIMO system is characterized by several Base Station (BS) antennas interacting with several User Equipment (UE)s over a wireless communication channel by spatial multiplexing [31]. The power dissipation at MIMO BS is significant when the system has to be scaled on a massive level for servicing an increasing number of UEs [32]. MIMO uplink detection with the ZF algorithm is explored to evaluate the impact of approximate signed multiplication circuits on symbol error-rate performance.

Approximate circuits are evaluated for the ZF MIMO uplink detection algorithm. For a generic BS model, a BS with B antennas servicing U number of UE is considered. Every UE has a single antenna. Accordingly, $\mathbf{y} = \mathbf{Hx} + \mathbf{n}$ represents MIMO uplink signal at BS, where $\mathbf{y} \in \mathbb{C}^{B\times 1}$ is the vector representing receive signal over B antennas of BS, $\mathbf{x} \in \mathbb{C}^{U\times 1}$ is transmit signal estimate from UEs to BS. $\mathbf{H} \in \mathbb{C}^{B\times U}$ is the channel model whose entries follow identical distribution (i.i.d). $\mathbf{n} \in \mathbb{C}^{B\times 1}$ is the channel noise. The ZF MIMO uplink signal is obtained as:

$$\hat{\mathbf{x}} = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{y} \tag{16}$$

Typically, the more critical case is when the number of transmit antennas is negligible to number of receive antennas i.e. $B \gg U$. In such scenarios, statistical methods of optimization are used to estimate the transmit vector $\hat{\mathbf{x}}$ from receive signal $\mathbf{y}$. Symbol error rate performance of MIMO uplink signal detector improves with increasing the number of BS antennas as the single-user channels become more decorrelated and approaches optimal uplink detection performance when the number of BS antennas is infinite [33]. However, this scenario is practically not possible to engineer due to resource constraints, signal processing complexity and energy requirements. Hence, optimization is introduced in the MIMO uplink signal detection to achieve optimal detection with bounded resource constraints.

Approximate multiplication circuits functionally implemented in the C environment are compiled into a dynamic library for accelerated simulation, which is linked to high-level Python implementation of the MIMO uplink detection, channel matrix being randomly generated for simulation. The circuit implementation in the C environment is interfaced with the Python environment using ctypes library. The 64-bit data type *unsigned long long* is used in C to handle multiplication operands and the multiplication product is interfaced with *c_int64* data type provided by ctypes library. To compute Eq. (16), matrix inverse operation is performed using the state-of-art Gauss-Jordan elimination method and accurate multiplication operations are replaced with approximate multiplication operations. To perform a multiplication operation, the operands are left bit shifted by $N-1$ and truncated to the nearest signed integer since all the operands are in the range (−1,1). After the multiplication operation, the product is right bit-shifted by $2(N-1)$.

Simulation result with 5000 randomly generated symbols for ZF MIMO uplink detection using various approximate multipliers circuits is presented in Fig. 13. For the 8-bit simulation of 16-QAM and 64-QAM based MIMO detection, circuits with approximate compressors,

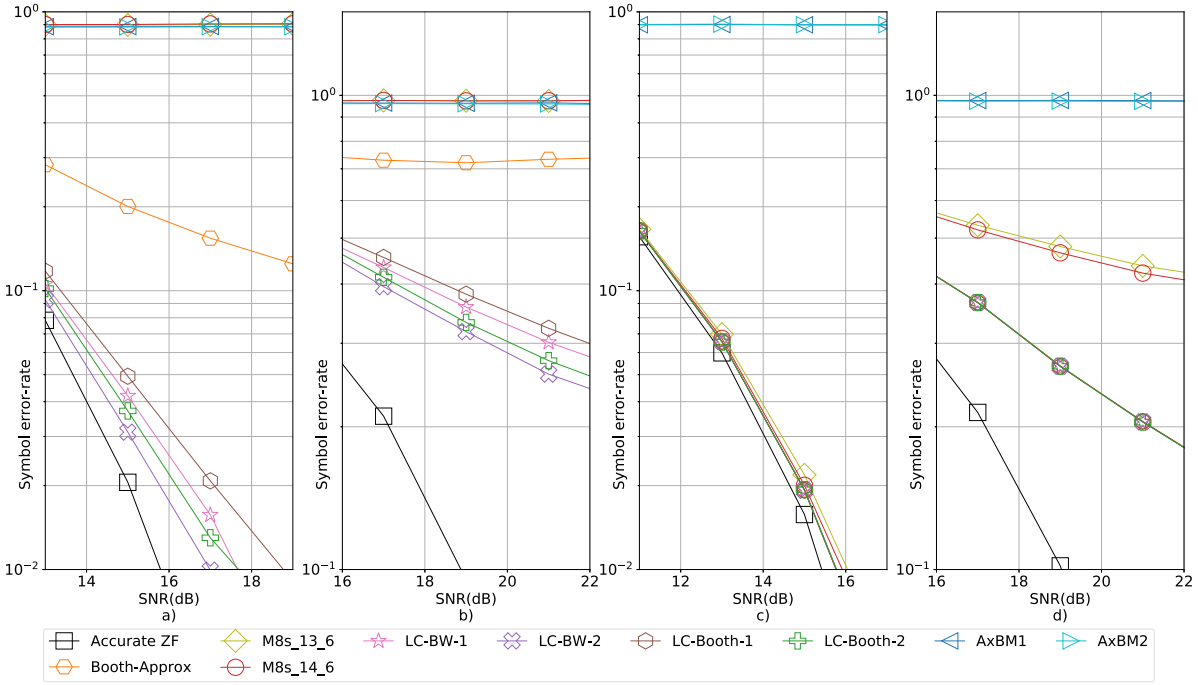

Fig. 13. Symbol error rate vs. SNR simulation for ZF MIMO uplink detection using 8-bit circuits for MIMO configuration $B = 128$, $U = 4$ given as (a) 16-QAM (b) 64-QAM and 16-bit circuits for MIMO configuration $B = 128$, $U = 4$ given as (c) 16-QAM (d) 64-QAM.

AxBM1 and AxBM1 show the worst error-rate performance. Booth-Approx diverges for higher Signal-to-Noise Ratio (SNR) for 16-QAM and 64-QAM configurations. LC based circuits converge towards a 1% symbol error-rate at a difference of less than 3 dB for 16-QAM as compared to the accurate ZF MIMO uplink detector. Symbol error-rate performance of version 2 of both LC-BW and LC-Booth is better than that of version 1 of both circuits respectively. ZF detector with LC-BW circuits provides more optimal detection than ZF detector with LC-Booth circuits. For 16-bit ZF MIMO uplink detection, detectors of all approximate circuits except AxBM1 and AxBM2, achieve a 1% symbol error-rate at an SNR difference of less than 1 dB for 16-QAM. Approximation in circuits increases inter-symbol interference which is prominently sensitive to higher QAM modulations.

ZF detection with LC-BW circuits shows close to accurate ZF detection performance, specifically at 16-QAM configuration for both 8-bit and 16-bit ZF detection, but the LC-BW circuits are characterized by relatively high PDP. ZF detection with LC-Booth circuits show close to that of ZF detection with LC-BW and the LC-Booth circuits have a relatively lower PDP than LC-BW circuits. Multiplication circuits with approximate compressors have the lowest PDP, however, their ZF detection stringently requires 16-bit operation for convergence. LC-Booth circuits are more advantageous than Booth-Approx for ZF detection symbol error rate performance as well as PDP efficiency.

7. Conclusion

LC methodology is an approximation technique employed on the accurate multiplication circuit for FPGA implementation to harness gains in energy and resource consumption by penalizing the accuracy of the multiplication product. The application of LC methodology for approximate signed multiplication circuits based on BW and Booth provides a systematic control over LUT resource consumption and PDP. LC-BW circuits show the best performance in terms of multiplication accuracy by cutting down on the PDP of the accurate BW, however, have more LUT consumption than other competitive approximate circuits. LC-Booth circuits require fewer LUTs and have lower PDP as compared to LC-BW circuits. However, since the number of MPPs is less in LC-Booth circuits, the granularity of approximation is lower for them and hence they incur more penalty in error performance as compared to LC-BW circuits. LC methodology provides explicit control over the approximation for LUT consumption, energy efficiency and accuracy for LC-BW and LC-Booth circuits, by selectively choosing the eligible LUTs of a MPP row for the particular circuit by using $T^{\#}$ parameter. Massive MIMO detection involves computationally intensive matrix-matrix and matrix–vector multiplication operations, which are fundamentally optimized by substituting accurate multiplication with approximate multiplication. For the application of approximate circuits for MIMO uplink detection, the ZF detector with LC-BW circuits achieves close to accurate ZF detection performance. It is also observed that the error performance of approximate multiplication circuits is critical for ZF detection for high QAM and low bit-width configurations. It can also be inferred that no single error metric can be used to optimally choose the approximate multiplication circuit for the ZF MIMO detection. CPD for circuits increase with bit-width and can prove a critical factor in configuring the circuit with high clock frequency. The proposed work presents a heuristic methodology for approximating arithmetic multiplication circuits for FPGA and such analysis for higher bit-widths is a topic of further research exploration with future computing capabilities.

CRediT authorship contribution statement

Abhinav Kulkarni: Conceptualization, Methodology, Investigation, Visualization, Writing – Original Draft. **Messaoud Ahmed Ouameur:** Writing – review & editing, Supervision, Resources, Project administration. **Daniel Massicotte:** Writing – review & editing, Supervision, Funding acquisition, Resources.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Daniel Massicotte reports financial support was provided by Natural Sciences and Engineering Research Council of Canada (NSERC), Prompt, Canadian Foundation for Innovation (CFI), CMC Microsystems,

Opal-RT Technologies Inc. and Hydro-Québec. The other authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The entire code employed for this work is accessible as open-source on GitHub at: https://github.com/abhinav333/Logic-Cloning.

Appendix A

A.1. Lemma

The following derivation explains that if a group of LUT-LSs of the MPP have equal TP value of $O6$ signal, then the arithmetic mean of β-MPP can be approximated without using all LUT-LSs of the MPP. Let the multiplication product $[\Phi]_2$ have N_R MPPs, each comprised of N_C bits. The nth MPP denoted as $[\Phi_n]_2$ is comprised of the nth β-MPP denoted as $\sum_{i=0}^{n-1}[\Phi_i]_2$ and the CC offset denoted as C_n. Therefore, $[\Phi_n]_2 = [C_n]_2 + \sum_{i=0}^{n-1}[\Phi_i]_2$ with $[\Phi_0]_2 = 0$ and the final product given as $[\Phi]_2 = [\Phi_{n=N_R}]_2$. ρ_i represents the power values of radix 2 such that $\rho_0 < \rho_1 < \cdots < \rho_{N_C-1}$. Hence, nth β-MPP is represented as $\sum_{i=0}^{n-1}[\Phi_i]_2 = -\varphi_{N_C-1}2^{\rho_{N_C-1}} + \sum_{i=0}^{N_C-2}\varphi_i 2^{\rho_i}$. The value of the nth $\beta-$MPP for the ζth simulation step is computed as:

$$\sum_{i=0}^{n-1}[\Phi_i^{(\zeta)}]_2 = -\varphi_{N_C-1}^{(\zeta)}2^{\rho_{N_C-1}} + \sum_{i=0}^{N_C-2}\varphi_i^{(\zeta)}2^{\rho_i} \tag{17}$$

The arithmetic mean of nth $\beta-$MPP evaluated over the exhaustive simulation range:

$$\mathcal{A}\left(\sum_{i=0}^{n-1}[\Phi_i]_2\right) = -\mathcal{A}(\varphi_{N_C-1})2^{\rho_{N_C-1}} + \sum_{i=0}^{N_C-2}\mathcal{A}(\varphi_i)2^{\rho_i} \tag{18}$$

From Eq. (6), as $\mathcal{T}(\varphi_\lambda) = \cdots = \mathcal{T}(\varphi_{\lambda+T^\#-1}) \implies \mathcal{A}(\varphi_\lambda) = \cdots = \mathcal{A}(\varphi_{\lambda+T^\#-1})$

Hence, Eq. (18) is transformed as:

$$\mathcal{A}\left(\sum_{i=0}^{n-1}[\Phi_i]_2\right) = -\mathcal{A}(\varphi_{N_C-1})2^{\rho_{N_C-1}} + \sum_{i=\lambda+T^\#}^{N_C-2}\mathcal{A}(\varphi_i)2^{\rho_i} \cdots + \underbrace{\mathcal{A}(\varphi_{\lambda+T^\#-1})\sum_{i=\lambda}^{\lambda+T^\#-1}2^{\rho_i}}_{\text{critical term}} + \sum_{i=0}^{\lambda-1}\mathcal{A}(\varphi_i)2^{\rho_i} \tag{19}$$

The critical term in Eq. (19) is approximated as:

$$\mathcal{A}(\varphi_{\lambda+T^\#-1})\sum_{i=\lambda}^{\lambda+T^\#-1}2^{\rho_i} \approx \mathcal{A}(\varphi_{\lambda+T^\#-1})2^{\rho_{\lambda+T^\#-1}} \tag{20}$$

Using Eq. (20) in Eq. (19):

$$\mathcal{A}\left(\sum_{i=0}^{n-1}[\hat{\Phi}_i]_2\right) = -\mathcal{A}(\varphi_{N_C-1})2^{\rho_{N_C-1}} + \sum_{i=\lambda+T^\#}^{N_C-2}\mathcal{A}(\varphi_i)2^{\rho_i} \cdots + \underbrace{\mathcal{A}(\varphi_{\lambda+T^\#-1})2^{\rho_{\lambda+T^\#-1}}}_{\text{approximated term}} + \sum_{i=0}^{\lambda-1}\mathcal{A}(\varphi_i)2^{\rho_i} \tag{21}$$

Thus for every nth MPP, by using the approximation in Eq. (20), an error is generated in the $\beta-$MPP value with arithmetic mean given as:

$$\mathcal{A}\left(\sum_{i=0}^{n-1}[\Phi_i]_2\right) - \mathcal{A}\left(\sum_{i=0}^{n-1}[\hat{\Phi}_i]_2\right) = \mathcal{A}(\varphi_{\lambda+T^\#-1})\sum_{i=\lambda}^{\lambda+T^\#-2}2^{\rho_i} \tag{22}$$

The arithmetic mean error in Eq. (22) is accumulated at every $\beta-$MPP till the final multiplication product value is computed, provided $\mathcal{T}(\varphi_\lambda) = \cdots = \mathcal{T}(\varphi_{\lambda+T^\#-1})$ for every MPP.

References

[1] J. Han, M. Orshansky, Approximate computing: an emerging paradigm for energy-efficient design, in: 2013 18th IEEE European Test Symposium, ETS, 2013, pp. 1–6, http://dx.doi.org/10.1109/ETS.2013.6569370.

[2] H. Jiang, F.J.H. Santiago, H. Mo, L. Liu, J. Han, Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications, Proc. IEEE 108 (12) (2020) 2108–2135, http://dx.doi.org/10.1109/JPROC.2020.3006451.

[3] V.K. Chippa, S.T. Chakradhar, K. Roy, A. Raghunathan, Analysis and characterization of inherent application resilience for approximate computing, in: 2013 50th ACM/EDAC/IEEE Design Automation Conference, DAC, 2013, pp. 1–9, http://dx.doi.org/10.1145/2463209.2488873.

[4] A. Lingamneni, C. Enz, K. Palem, C. Piguet, Synthesizing Parsimonious Inexact Circuits through Probabilistic Design Techniques, ACM Trans. Embed. Comput. Syst. 12 (2s) (2013) http://dx.doi.org/10.1145/2465787.2465795.

[5] W. Qian, M.D. Riedel, H. Zhou, J. Bruck, Transforming Probabilities With Combinational Logic, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 30 (9) (2011) 1279–1292, http://dx.doi.org/10.1109/TCAD.2011.2144630.

[6] L.N. Chakrapani, K.V. Palem, A Probabilistic Boolean Logic and Its Meaning, Rice University, Department of Computer Science Technical Report, 2008.

[7] D. Gardy, Random boolean expressions, in: Discrete Mathematics and Theoretical Computer Science, Discrete Mathematics and Theoretical Computer Science, 2005, pp. 1–36.

[8] I. Kuon, J. Rose, Measuring the Gap Between FPGAs and ASICs, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 26 (2) (2007) 203–215, http://dx.doi.org/10.1109/TCAD.2006.884574.

[9] S. Mittal, A Survey of Techniques for Approximate Computing, ACM Comput. Surv. 48 (4) (2016) http://dx.doi.org/10.1145/2893356.

[10] C.S. Wallace, A Suggestion for a Fast Multiplier, IEEE Trans. Electron. Comput. EC-13 (1) (1964) 14–17, http://dx.doi.org/10.1109/PGEC.1964.263830.

[11] L. Dadda, Some schemes for parallel multipliers, Alta Freq. 34 (1965) 349–356.

[12] H. Parandeh-Afshar, P. Ienne, Measuring and Reducing the Performance Gap between Embedded and Soft Multipliers on FPGAs, in: 2011 21st International Conference on Field Programmable Logic and Applications, 2011, pp. 225–231, http://dx.doi.org/10.1109/FPL.2011.48.

[13] Intel-Altera, Stratix III Device Handbook, volume I, 2.2, 2011.

[14] M. Kumm, S. Abbas, P. Zipf, An Efficient Softcore Multiplier Architecture for Xilinx FPGAs, in: 2015 IEEE 22nd Symposium on Computer Arithmetic, 2015, pp. 18–25, http://dx.doi.org/10.1109/ARITH.2015.17.

[15] J. Hu, W. Qian, A new approximate adder with low relative error and correct sign calculation, in: 2015 Design, Automation Test in Europe Conference Exhibition, DATE, 2015, pp. 1449–1454.

[16] S. Ullah, T.D.A. Nguyen, A. Kumar, Energy-Efficient Low-Latency Signed Multiplier for FPGA-Based Hardware Accelerators, IEEE Embed. Syst. Lett. 13 (2) (2021) 41–44, http://dx.doi.org/10.1109/LES.2020.2995053.

[17] G.W. Bewick, Fast Multiplication: Algorithms and Implementation (Ph.D. thesis), Stanford University, 1994.

[18] S. Ullah, H. Schmidl, S.S. Sahoo, S. Rehman, A. Kumar, Area-Optimized Accurate and Approximate Softcore Signed Multiplier Architectures, IEEE Trans. Comput. 70 (3) (2021) 384–392, http://dx.doi.org/10.1109/TC.2020.2988404.

[19] S. Ullah, S. Rehman, M. Shafique, A. Kumar, High-Performance Accurate and Approximate Multipliers for FPGA-Based Hardware Accelerators, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 41 (2) (2022) 211–224, http://dx.doi.org/10.1109/TCAD.2021.3056337.

[20] B.S. Prabakaran, S. Rehman, M.A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, M. Shafique, DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems, in: 2018 Design, Automation Test in Europe Conference Exhibition, DATE, 2018, pp. 917–920, http://dx.doi.org/10.23919/DATE.2018.8342140.

[21] S. Boroumand, H.P. Afshar, P. Brisk, Approximate quaternary addition with the fast carry chains of FPGAs, in: 2018 Design, Automation Test in Europe Conference Exhibition, DATE, 2018, pp. 577–580, http://dx.doi.org/10.23919/DATE.2018.8342073.

[22] S. Balasubramani, U. Jagadeeshan, U. Krishnamoorthy, Performance optimized approximate multiplier architecture ST-AxM - based on statistical analysis and static compensation, Microelectron. Reliab. 151 (2023) 115277, http://dx.doi.org/10.1016/j.microrel.2023.115277.

[23] S. Venkatachalam, S.-B. Ko, Design of Power and Area Efficient Approximate Multipliers, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (5) (2017) 1782–1786, http://dx.doi.org/10.1109/TVLSI.2016.2643639.

[24] N. Van Toan, J.-G. Lee, FPGA-based multi-level approximate multipliers for high-performance error-resilient applications, IEEE Access 8 (2020) 25481–25497.

[25] H. Waris, C. Wang, W. Liu, F. Lombardi, AxBMs: Approximate radix-8 Booth Multipliers for High-Performance FPGA-Based Accelerators, IEEE Trans. Circuits Syst. II 68 (5) (2021) 1566–1570, http://dx.doi.org/10.1109/TCSII.2021.3065333.

[26] Xilinx, 7 Series FPGAs Configurable Logic Block, User Guide, UG474 (v1.8) , 2016.

[27] Xilinx, Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC, Libraries Guide, UG953 (v2021.2), 2021.

[28] D.V. Hall, Digital Circuits and Systems, Glencoe/McGraw-Hill School Publishing Company, 1989.

[29] J. Liang, J. Han, F. Lombardi, New Metrics for the Reliability of Approximate and Probabilistic Adders, IEEE Trans. Comput. 62 (9) (2013) 1760–1771, http://dx.doi.org/10.1109/TC.2012.146.
[30] Xilinx, Multiplier, LogiCORE IP Product Guide, PG108, v12.0 , 2015.
[31] A. Kulkarni, M.A. Ouameur, D. Massicotte, Hardware Topologies for Decentralized Large-Scale MIMO Detection Using Newton Method, IEEE Trans. Circuits Syst. I. Regul. Pap. 68 (9) (2021) 3732–3745, http://dx.doi.org/10.1109/TCSI.2021.3097042.
[32] N. Rajatheva, I. Atzeni, E. Bjornson, A. Bourdoux, S. Buzzi, J.-B. Dore, S. Erkucuk, M. Fuentes, K. Guan, Y. Hu, et al., White paper on broadband connectivity in 6G, 2020, arXiv preprint arXiv:2004.14247.
[33] E. Björnson, L. Sanguinetti, H. Wymeersch, J. Hoydis, T.L. Marzetta, Massive MIMO is a reality—What is next?: Five promising research directions for antenna arrays, Digit. Signal Process. 94 (2019) 3–20, http://dx.doi.org/10.1016/j.dsp.2019.06.007, Special Issue on Source Localization in Massive MIMO.