JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

DESIGN OF HIGH PERFORMANCE DYNAMICALLY TRUNCATED APPROXIMATE PIPELINED MULTIPLIER

[1]Dr. Jayanthi K Murthy, [2]Sneha Bhusaraddi
[1]Associate Professor, [2]Student
[1]Department of Electronics and Communication Engineering,
[1]BMS College of Engineering, Bengaluru, India

Abstract : In the realm of computational arithmetic units, the multiplier stands as one of the most widely utilized functional components. Across numerous applications, the frequent deployment of multipliers comes at a notable cost with the consumption of substantial power resources. To address this challenge, a cutting-edge approach has emerged that is the use of approximate multipliers. This innovative method seeks to curtail energy consumption and alleviate critical path latency, primarily within applications where a certain margin for error is acceptable. In return for reduced energy usage and enhanced performance, the approximate multiplier willingly sacrifices precision. This paper not only introduces the concept of an approximate 4:2 compressor but also pioneers an adjustable approximate multiplier. This versatile multiplier possesses the capability to dynamically truncate partial products, aligning its accuracy with the variable needs of the application. Furthermore, in pursuit of latency reduction, the paper puts forward the concept of pipelining. By segmenting the multiplication process into stages, this approach optimizes the processing speed. A standout feature of the suggested approximate multiplier is its adaptability. It can tailor its precision and processing power in real-time, responding to the specific requirements of the end-users. Simulation results, conducted using the Xilinx Vivado simulation software, attest to the efficacy of the proposed approximate multiplier. In comparison to the standard multiplier, it exhibits the potential to reduce latency by a substantial 58.620% and lower average power consumption by an impressive 25.438%.

Index Terms **- Approximate computing, approximate multiplier, dynamic truncation, compressor, pipelining, reconfigurable approximate design.**

I. INTRODUCTION

In many diverse applications, such as digital signal processing (DSP), computer vision, multimedia processing, image recognition, and artificial intelligence, multipliers play a central role as key arithmetic units. These applications often demand a multitude of multiplication operations, which, in turn, engender substantial power consumption. This elevated power demand poses a considerable challenge, particularly in the context of resource-constrained mobile devices. Consequently, numerous research endeavors have explored innovative strategies to mitigate the power consumption associated with multiplier circuits. For applications where a certain degree of error tolerance is permissible, one notable technique for curtailing the power requirements of multipliers is approximate multiplication. In such scenarios, the absolute precision of computational outcomes becomes less priority.The adoption of approximate multipliers comes with a deliberate compromise on the accuracy of results, but it yields substantial benefits in terms of reduced silicon real estate utilization, diminished timing delays, and a more efficient power profile.

There are two distinct categories of approximate multipliers, each offering unique strategies for power optimization. The first category involves dynamic voltage scaling, a technique that regulates the timing characteristics of the multiplier. By supplying the multiplier with a lower voltage, it effectively alters the critical path, causing delays. This deliberate voltage reduction can lead to timing violations, resulting in errors and the generation of approximate results. The second type of approximate multipliers focuses on the actual redesign of precision multiplier circuits. Prominent examples include the Wallace tree multiplier and Dadda tree multiplier. Through such redesigns, the fundamental functional behavior of multipliers can be altered to achieve power savings while accepting a certain degree of imprecision in the results.

Historically, a substantial portion of research in multiplier redesign has proposed sub-optimal m-n compressors with m inputs and n outputs. These less reliable compressors were frequently used to condense partial results generated during the multiplication process, consuming a substantial share of the multiplier's energy and introducing significant delays in the critical path. Essentially, these earlier approximation multipliers primarily traded output precision for predefined power consumption levels. Nonetheless, in contrast, certain applications, such as artificial intelligence, demonstrated dynamic requirements that fluctuated over time, underscoring the value of adaptability in fine-tuning accuracy and power consumption as needed.

In the context of our research, we propose the utilization of a high-precision 4:2 compressor as a pivotal component. Building upon this, we proceed to construct an approximate multiplier. To further enhance flexibility in terms of precision and power consumption, we introduce a dynamic input truncation method. Additionally, we implement a pipelining concept to mitigate latency and enhance overall performance.

This paper makes several notable contributions:

1. We introduce a high-precision approximate 4:2 compressor, which serves as a fundamental building block for the envisioned approximate multiplier.
2. We put forth a dynamic input truncation technique, tailored to offer adaptability in terms of both accuracy and power consumption in the context of multiplication. This approach proves particularly advantageous for Convolutional Neural Networks (CNNs), enabling straightforward adjustments of power utilization to meet the diverse requirements of individual network layers.
3. We advocate for the adoption of a pipelined multiplier architecture to curtail latency and enhance the overall circuit's performance.
4. Leveraging the innovative 4:2 compressor, dynamic input truncation method, and pipelining concept, we present a high-performance, reconfigurable approximate multiplier with the potential to cater to a wide array of computational needs.

Based on our simulation results, when compared to a traditional reconfigurable multiplier, the proposed approximate multiplier, featuring dynamic input truncation and pipelining, exhibits a remarkable reduction in latency by approximately 58.620%. Additionally, it achieves an average power consumption decrease of approximately 25.438%.

The remaining part of this paper has been categorized, as follows: Section II introduces previous approximate multipliers and the metric for comparing different approximate multipliers. Section III introduces our proposed approximate multiplier. Section IV Simulation results and discussion. Section V compares the results of the conventional multiplier and the proposed approximate multiplier. Section VI concludes the paper.

II. LITERATURE SURVEY

Various techniques are available for constructing approximate multipliers, offering different avenues for achieving power efficiency. These methods encompass approaches like voltage scaling to adjust the supply voltage, selective truncation of certain partial product rows, designing equations aimed at simplifying multiplier operations, and the strategic use of approximate compressors to reduce the number of required partial product rows. Each of these methods offers a unique approach to address the power consumption challenges associated with multipliers in different applications.

In previous studies, researchers have implemented the voltage scaling technique as described in references [1] and [2]. This method involves the adjustment of the supply voltage to logic gates, effectively reducing power consumption. However, if the supply voltage drops below the required nominal level, it can lead to timing violations and the generation of approximate results. The magnitude of the error introduced in such cases can be notably significant, especially when timing violations affect critical pathways.

To minimize the carry propagation distance, researchers in references [3] and [4] introduced approximate multipliers. They achieved this by selectively truncating partial product columns that were in close proximity to the least significant bit (LSB) column. As one moved further away from the LSB column, the contribution of these partial products diminished due to their reduced significance. This strategic truncation of partial products with relatively low weights helped mitigate the potential for error propagation over longer distances.

In the work detailed in reference [5], introduces an approximate multiplier by modifying the original multiplication equation represented as (1). In this revised approach, they replaced the rounded input values A and B, denoted as Ar and Br. To achieve hardware cost reductions, they approximated Ar and Br by selecting the nearest values of 2n. This approximation allowed them to compute the products Ar x B, Br x A, and Ar x Br efficiently using shift operations. The updated multiplication equation, depicted in equation (2), no longer incorporates the (Ar - A) (Br - B) term, signifying the shift towards an approximate multiplication method.

$$A \times B = (Ar - A) \times (Br - B) + Ar \times B + Br \times A - Ar \times Br \quad (1)$$

$$A \times B = Ar \times B + Br \times A - Ar \times Br \quad (2)$$

As described in reference [6], the author introduces the Wallace tree multiplier, an architecture renowned for its speedy multiplication capabilities. This multiplier leverages Wallace tree structures and exact 2:2 and 3:2 compressors, often referred to as half-adders, to effectively reduce the number of rows of partial products. It's also possible to integrate precise 4:2 compressors into Wallace engines for enhanced functionality. The Tree Multiplier, outlined in reference [7], further refines this arrangement to create a more regular and efficient structure. The end product is formed by employing a carry-propagating adder to sum up the reduced partial products. In the realm of past research, many efforts were directed towards developing approximation compressors that were adaptations of the widely used exact 4:2 compressor.

A precise 4:2 compressor can be created by the combination of two full adders, as illustrated in Figure 1. Here, Cin represents the carry-in from the preceding column compressor in a multiplication operation, and X1 to X4 denote the partial products within the same column. This precise 4:2 compressor yields three distinct outputs: Cout, sum, and carry, as delineated in the figure.

In contrast, most approximate 4:2 compressors, as depicted in the block diagram of Figure 2, deviate from this pattern. They neither utilize Cin nor generate Cout, diverging from the exact 4:2 compressor featured in Figure 1. The shift in design leads to an increase in the total number of inputs to four, while the overall number of outputs decreases to two. This transformation can significantly simplify the compression of partial products. However, it's crucial to note that this simplification introduces the possibility of errors, particularly when all four inputs are set to 1. In such cases, errors are inevitable due to the binary output '100,' which necessitates a minimum of three output ports for representation.

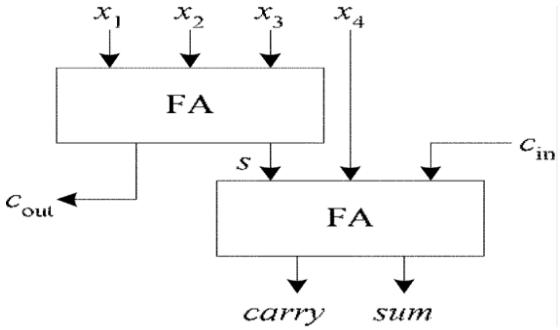

FIGURE 1. Accurate 4-2 compressor.

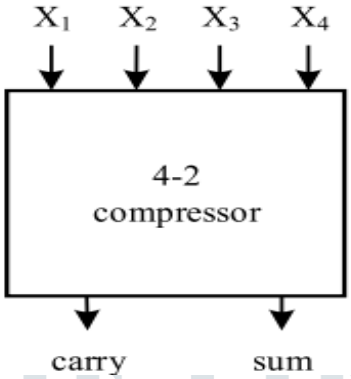

FIGURE 2. Approximate 4:2 compressor

In the paper [8] author introduced two variations of approximate 4:2 compressors. The first, Momeni_acc, deviates from the exact 4:2 compressor by having five inputs and three outputs. On the other hand, Momeni_fast, akin to many other approximate multipliers, employs four inputs and yields two outputs, aligning it with a more compact design.
In a similar vein, the author in paper [9] proposed three approximate 4:2 compressors, ACCI1, ACCI2, and ACCI3, all derived from the structure of the exact 4:2 compressor. These compressors employed different formulas to calculate the sum bit, offering various approaches to approximation.
The author in reference [3] took a step further by enhancing ACCI3, addressing its error-prone conditions. Additionally, they designed a straightforward error recovery circuit for this improved compressor. However, it's worth noting that when both X3 and X4 were set to 1, ACCI3 exhibited errors, necessitating further refinements.
However, the author modification in reference [3] introduced a noteworthy change to the generated carry's Boolean function, ensuring that the error value consistently remained at -1. This adjustment, in turn, simplified the associated error recovery circuit.
In a different approach, in reference [10] sought to minimize the latency of their proposed approximate 4:2 compressor by replacing the XOR gate with a MUX gate. This design optimization aimed at enhancing efficiency.
Nonetheless, an issue emerged when dealing with ACCI1 and the scenario where X1, X2, X3, and X4 were all set to 1. In this context, Lin's compressor yielded a value of two for the error distance instead of one. The author in paper [11] put forth an innovative dual-stage 4-2 compressor. This unique design allowed for the error distance to be either positive or negative, adapting to different situations. In cases where faults occurred during the process, they divided the partial product reduction into multiple stages and cascaded the dual-stage 4-2 compressor across various phases to address the issues comprehensively.

In the work detailed in reference [12], introduces a 4:2 compressor that relied on a majority-based approach. The carry bit was determined by a 3-input majority gate, while the sum bit was consistently set to 1. In a distinct approach, as described in reference [13], crafted an approximate 4:2 compressor employing a stacking circuit technique that quantified the number of ones in the input. This ingenious method effectively reduced the four inputs to three, with both the sum and carry bits generated by a complete adder.
Seeking a balanced trade-off. [14] factored in the uneven data distribution observed in Convolutional Neural Networks (CNNs) where activation's and weights followed Gaussian-like distributions. Their work weighed the interplay between precision, power consumption, and latency. When compared to precise 4:2 compressors, the approximate 4:2 compressors mentioned above excelled in rapidly reducing partial products, exhibited lower power usage, and boasted shorter timing delays.
The paper [15] presents four configurable approximate 4:2 compressors, each of which consists of an approximate portion and an extra part with a distinctive construction. Both accurate and approximate operation modes are available. When operating in accurate mode, the supplemental portion is active and modifies the approximation part's output to produce accurate results.
The additional section is shut down with power gating while in the approximation mode the approximation result is obtained directly as the final result. The precise 2-2 compressor (half-adder) and the 3-2 compressor (full adder) were updated by in the paper [17]. The updated compressors feature a mask signal to regulate whether the result is exact or approximate.
The paper [19] creates an configurable approximate multiplier that enables multiplications with a range of input bit widths by splitting a multiplier into two sub-multipliers.
The author in paper [4] proposed a programmable truncated multiplier. They replaced the 2-input AND gates with 3-input AND gates to generate the partial product elements with the ability to truncate. During runtime, some partial product columns are shortened when high precision is not necessary. The precision of approximate multipliers for CNN inference should be predicted and dynamically changed, according to author of paper [18] examination of the impact of bit-width precision on accuracy.

In the paper [20], the author proposed dependency-resolving intra-unit pipeline architectures for high-throughput multipliers. Briefly speaking, the architectures generate partial results in the intra unit pipeline stages and forward them to their earlier pipeline stages so that dependent instructions can be injected into the execution unit with a minimum wait time.
In the paper [21], the author proposed RAPID, the first pipelined approximate multiplier and divider architectures, customized for FPGAs. The proposed units efficiently utilize 6-input Look-up Tables (6-LUTs) and fast carry chains to implement Mitchell's approximate algorithms The pipelined stages of the combinational data channel of multiplier and divider should be divided for almost uniform latency to reduce the latency overhead. To achieve this, they have adopted the following steps: first, each step of the multiplication or division is synthesized in isolation to get an estimation of the delay for each stage. Afterwards, re-synthesis has been performed to assess whether a marginal fine-tuning for the adopted partitioning can produce better end-to-end latency. Furthermore, a pipeline register's delay overhead is minimal, especially due to the fact that in most cases, the divider is on the critical route.

III. PROPOSED METHOD

The conventional and our suggested multiplication flows are initially contrasted in this section. The introduction of our proposed 4:2 compressor is as follows. The process then moves on to dynamic input truncation, which is employed to build the adjustable multiplier.

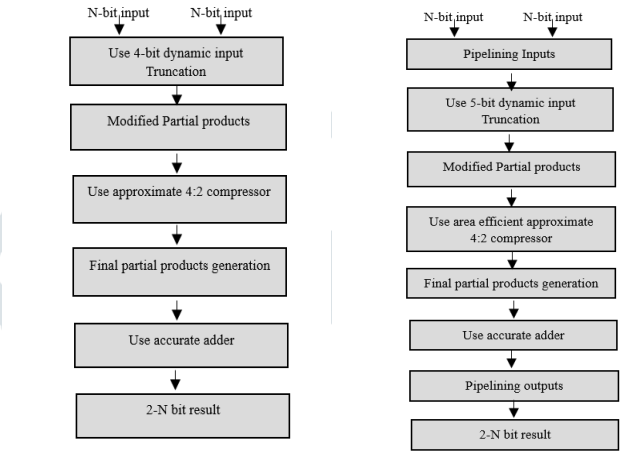

FIGURE 3: Multiplication Flow of a) conventional and b) proposed approximate multipliers

A. Proposed Flow And Approximate Multiplier

The overall operation of the traditional multiplication flow, which produces results in 2-N bits, is presented in Figure 3(a). Dynamic input truncation is achieved using two-input AND gates, and accurate partial products are first produced before being compressed using approximate compressors. Using precise adders, the compressed partial products are added to generate the final product. Figure 3(b) demonstrates our proposed approach for the indicated appropriate multipliers. The proposed multiplication distinguishes from traditional multiplication through the processes of creating partial products and compressing the partial products. When constructing partial products that provide 2-N bit outputs, we use dynamic input truncation, a pipelining approach, and an effective compressor to create the modified partial products.

B. Proposed Efficient 4-2 Compressor

In this work, a high-efficiency and low-power approximate 4:2 compressor is proposed. The design of the proposed 4:2 approximate compressor is described as follows. The proposed high speed area-efficient 4:2 approximate compressor has been demonstrated in Figure 4. The compressor inputs are A1, A2, A3 and A4, outputs are carry and sum. A multiplexer (MUX) based design approach is employed to produce sum. Output of XOR gate serves as the select line for the MUX. When select line goes high, (A3A4) is selected and when it goes low, (A3 + A4) is selected. The suggested 4: 2 compressor can reduce carry generation logic to an OR gate by including an error with error distance 1 in the accurate compressor's truth table. Below are the logical formulations for accomplishing SUM and CARRY.

$$SUM = (A1 \oplus A2)A3A4 + (A1 \oplus A2)(A3 + A4) \quad (1)$$
$$CARRY = A1 + A2 \quad (2)$$

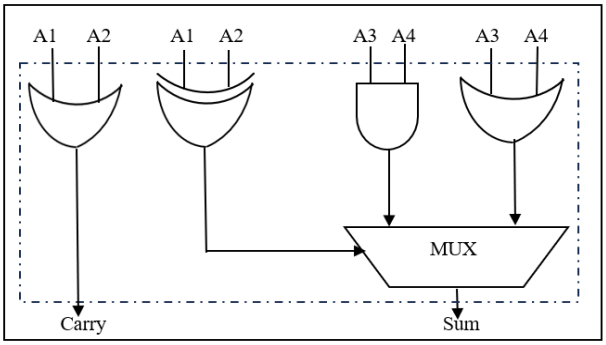

FIGURE 4. Proposed area-efficient 4:2 compressor.

TABLE 1. Truth table for proposed area efficient 4:2 compressor.

A1	A2	A3	A4	Carry	Sum	Diff
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

C. Error detection circuit

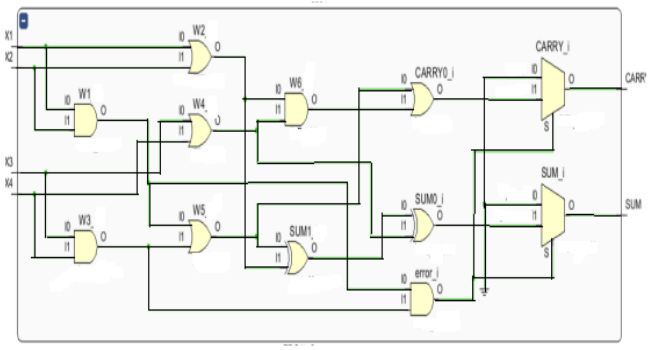

Figure 5: Error Detection Circuit

Equation (3)-(11) are used to construct W1–W4 from four inputs, X1–X4. The carry bit in the proposed compressor is designed to always be correctly generated since an approximately calculated carry bit has a greater error distance (ED) than the sum bit, i.e., an inaccurate carry bit generates two times the error distance (ED) of that produced by an incorrect sum bit. The equations of generating carry bit are shown in (3)-(11). In three situations, the carry bit will change to 1. X1 and X2 both make up one. Additionally, X3 and X4 are also 1. The third condition is either X1 or X2 equals 1 and either X3 or X4 equals 1. Eqn (7) examines the first two scenarios, while (8) examines the third scenario. The last carry bit is produced by Eqn (9). Eqn (17) illustrates the suggested equation to produce the sum bit. (18) displays the error detection circuit's (EDC) equation.

$$W1=X1 \text{ AND } X2 \quad (3)$$

$$W2=X1 \text{ OR } X2 \quad (4)$$

$$W3= X3 \text{ AND } X4 \quad (5)$$

$$W4=X3 \text{ OR } X4 \quad (6)$$

W5=W1 OR W3 (7)

W6= W2 AND W4 (8)

Carry=W5 OR W6 (9)

Sum=W5 XOR W2 XOR W4 (10)

ERROR= W1 AND W3 (11)

In approximate multipliers, error detection circuits are employed to identify and flag potential errors or inaccuracies that may occur due to the approximate nature of the multiplication operation. These circuits help ensure that any errors introduced by the approximation process are detected and can be accounted for or corrected.

D. Dynamic Input Truncation

We propose a dynamic input truncation strategy, as shown in Figure 6, to create a partial product, whose equation is given in (12), where A is the multiplicand and B is the multiplier, so as to come up with an adjustable approximation multiplier at runtime. To determine whether to end the partial product PPD, the Trunc signal is applied. The partial product is reduced to 0 if the Trunc value is 1. Power-saving Trunc signals accurately set the PPDs in the multiplications to zeros. In other words, when we notice the Trunc signals, we can envisage the hardware units in the corresponding columns being disabled.

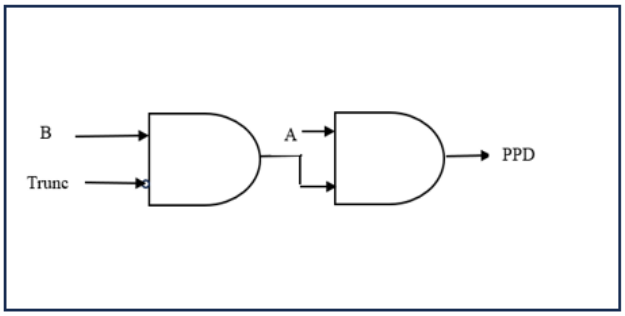

FIGURE 6. Modified partial product.

PPDij= (~ Trunc AND Bi) AND Aj (12)

Since each multiplier bit corresponds to 8 bits of the multiplicand, we recommend sharing gates with an additional AND gate for an 8x 8 multiplier to reduce hardware expenses. For example, PPD00 and PPD01 are equivalent to Trunc0B0A0 and Trunc0B0A1, respectively. Three 2-input AND gates are required in this scenario, as shown in Figure 7, and Trunc0B0 can be computed beforehand to construct a mask.

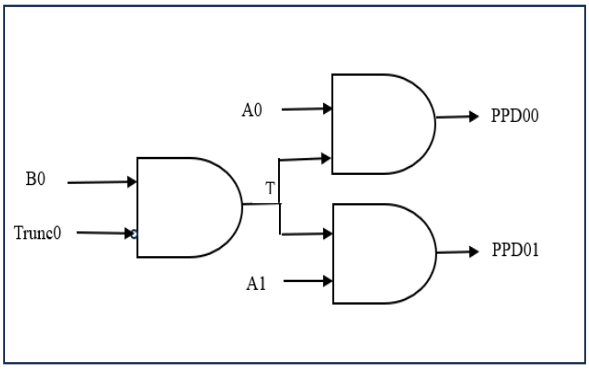

FIGURE 7. Gate sharing technique

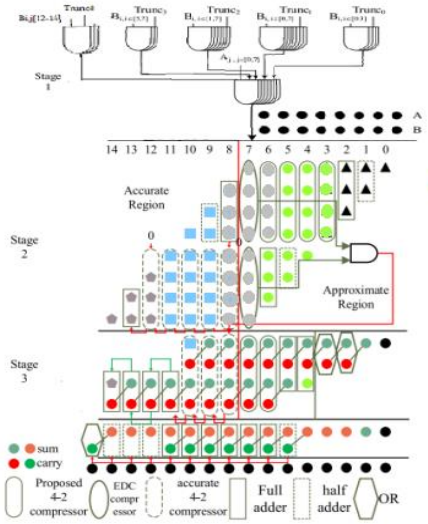

Figure 8: Approximate multiplier with 5-bit dynamic truncation using approximate 4:2 compressor

E. The Proposed Approximate Multiplier

FIGURE 8 displays a approximate multiplier using the suggested method. The proposed method can still be applied to larger multipliers even though the multiplier's input width is only intended to be 8 bits. There are three steps in the approximation multiplier that is proposed. Each partial product is produced in the initial stage using two 2-input AND gates, as previously demonstrated in Figure 6, with the gate sharing technique being used in Figure 7 to further minimize hardware costs.

Depending on the requirements, it is possible to assess the generated partial product's accuracy using the Trunc signal. Specifically, from MSB to LSB every bit controls column 14th-12th, 11th-9th, 8th-6th, 5th-3rd, and 2nd-0th respectively, corresponding to the shades of khaki, sky blue, green, and black in Stage 2 of Figure 8. This 5-bit Trunc signal is used in our proposed approximate multiplier to increase control efficiency and decrease hardware costs. For example, if the Trunc (4-0) is (00101)2, the columns 14th-12, 11th-9th, and 5th-3rd are accurate, whereas the columns 6 th -8th, and 2nd-0th are truncated.

Users can modify the recommended multiplier to suit their needs because the partitioning of the columns offers a variety of options for controlling Trunc signals.

The 3-4-4-4 and 3-3-3-3-3 partitions both keep a balance between power savings, accuracy, and area overhead, on the basis of the findings of our numerous research that tested various partitions. With more precise partitioning, there is more flexibility in determining amount of power that can be saved and how much accuracy is lost. But in return, it will encounter a significant amount of area overhead.

The process for compressing the partial products is illustrated in the second phase. The two regions of the resulting partial products are as follows: the accurate region is in columns 14th–8th, while the approximate region is in columns 7th–0th. Using the most logically sound half-half division, the actual and approximate zones are divided. We can execute a 30-70 split, for instance, with the approximative multiplier performing extra computations, but there will be a significant accuracy loss.

The approximate computing for power savings will not be impacted by a 70-30 split, though. Since their weight and significance have risen there, we make use of accurate 4:2 compressors to compress the partial products there. Because these columns are close to the LSB, where errors are less relevant, we create results in the third stage using OR gates in columns 3rd – 0th and omit carry propagation. We compress the partial products in the remaining columns using the suggested accurate 4:2 compressors, full adders, and half adders. Following the completion of the third stage, the final two partial product rows are obtained, and these two rows are then precisely joined to provide the outcomes.

The proposed approximate multiplier, as shown in FIGURE 8, contains high-accuracy 4-2 compressors, a simple error compensation circuit, and dynamic input truncation. There are twenty-five distinct configurations in our recommended approximate multiplier because the Trunc signal has a 5-bit bit width. In this case, the proposed_00001 indicates that Trunc = 00001, and the proposed_00100 indicates that Trunc = 00100.

However, only three configurations, proposed_00001, proposed_00100, and proposed_00111, can be adopted for their more accurate results resulting from the untruncated eight leftmost partial product columns.

We can modify our suggested approximation multiplier to choose the bits that are permitted to not be accurately computed at run-time, making it appropriate for quantized CNNs. As illustrated in FIGURE 8, we employ a 5-bit truncated configuration parameter called Trunc in each convolution layer to specify whether the columns of the partial product are disregarded. We can reduce the power consumption from many MACs by dynamically adjusting our suggested multiplier based on the decimal point positions of the quantized weights by employing the truncation option.

F. Pipelining Concept

Pipelining is the method of obtaining orders from the processor using a pipeline. It permits the orderly storing and carrying out of instructions. It is also known as processing in a pipeline. Simple pipelining can increase concurrency in systems without feedback loops. Pipelining is a technique that overlaps several instructions as they are being executed. A structure resembling a pipe is created by connecting the various phases of the pipeline to one another. Instructions come in one end and leave the other Pipelines boost the overall flow of instructions. A combinational circuit is followed by an input register in a pipeline system segment. The combinational circuit operates on the register, which is used to store data. The input register of the following section receives the combinational circuit's output. Pipelining can lead to a reduction in the critical path by placing latches at appropriate feed-forward locations. This reduction can be exploited to operate the system with higher speed. In other words, pipelining transforms a topology (containing no feedback loops) to an equivalent form that is now suitable for a high-speed application while the original topology cannot meet the speed demands of the application. Pipelining reduces the critical path at the expense of an increase in latches and the input-output delay referred to as system latency.

IV. SIMULATION RESULTS AND DISCUSSION

The experimental setup that will be utilized to assess the approximative multipliers is initially described in this section.
Then, it compares the approximate multipliers under different evaluation metrics. Finally, it compares the critical path delay, cell area, power consumption.

Figure 9: Output Waveform of Approximate Multiplier with 4-bit dynamic Truncation technique using approximate 4:2 compressor.

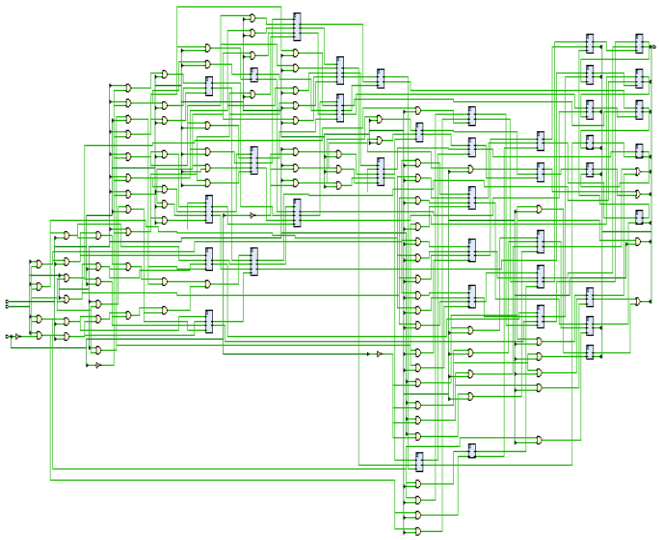

Figure 10: Schematic Diagram of Approximate Multiplier with 4 bit dynamic Truncation technique using approximate 4:2 compressor

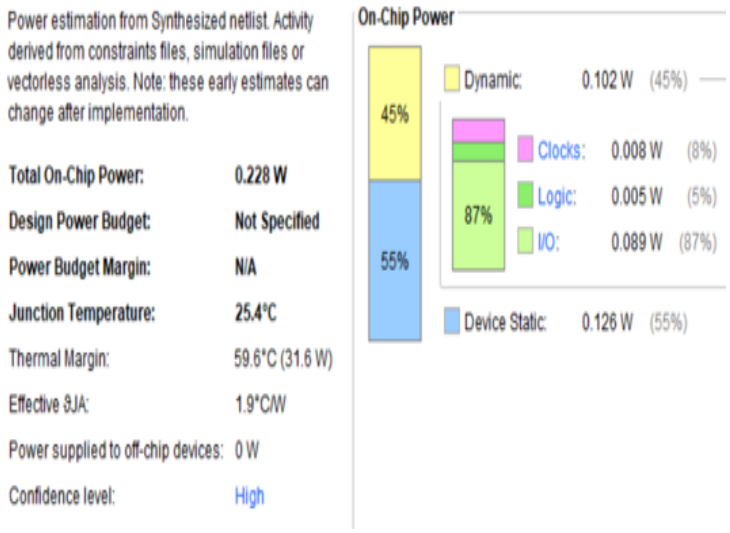

Figure 11: Power Report of Approximate Multiplier with 4-bit dynamic Truncation technique using approximate 4:2 compressor

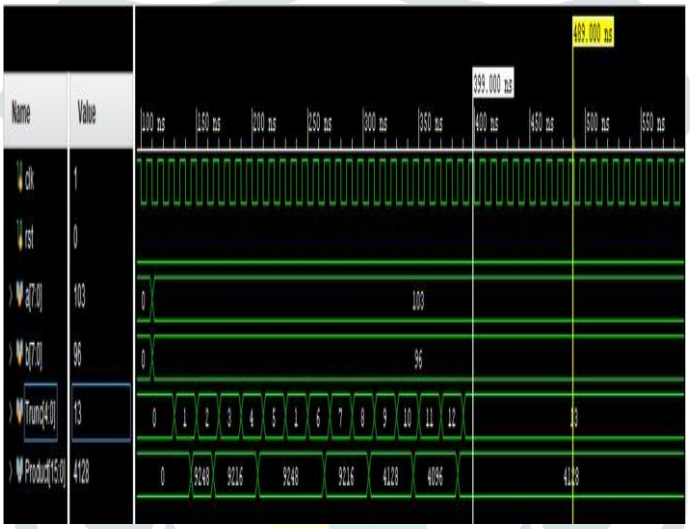

Figure 12: Output Waveform of Approximate Pipelined Multiplier with 5-bit dynamic Truncation technique using approximate efficient compressor.

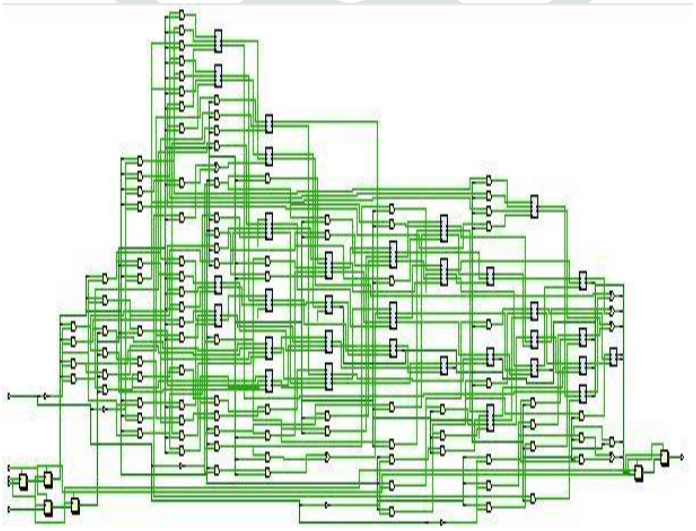

Figure 13: Schematic Diagram of Approximate pipelined Multiplier with 5-bit dynamic Truncation technique using approximate efficient 4:2 compressor

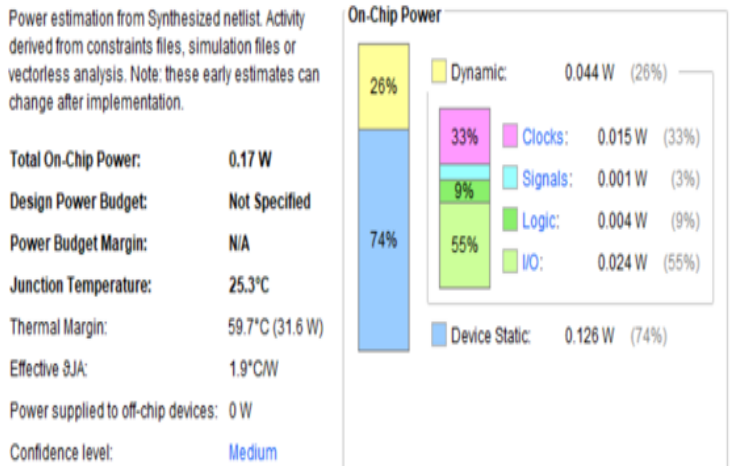

Figure 14: Power Report of Approximate pipelined Multiplier with 5-bit dynamic Truncation technique using approximate area efficient 4:2 compressor

V. COMPARISON OF RESULTS

Approximate Multipliers	Power(mW)	Delay(ns)	Area (LUT's)
8-bit Approximate multiplier using 4-bit dynamic Truncation with 3-4-4-4 partition and using 4:2 compressor with non pipelined architecture.	228	8.509	86
8-bit Approximate multiplier using 5-bit dynamic truncation with 3-3-3-3-3 partition and using area efficient compressor with pipelined architecture.	170	3.521	87

VI. CONCLUSION AND FUTURE SCOPE

In this study, we present an innovative approximate 4:2 compressor that offers a remarkable balance between accuracy and efficiency, serving as a fundamental building block for our proposed approximate multiplier. This novel approximate multiplier dynamically adapts by truncating partial products to tailor its accuracy according to specific application needs.

In comparison to traditional approximate multipliers, our adjustable approximate multiplier showcases substantial improvements, reducing both delay and average power consumption by 58.620% and 25.438%, respectively. Furthermore, our proposed pipelined multiplier excels in terms of minimal delay and the most efficient power utilization when benchmarked against other approximation multipliers.Future research directions will delve deeper into the analysis of diverse partitioning methods to establish more robust and concrete relationships between hardware costs, accuracy, and power consumption. In particular, we aim to address the variation in Trunc signals required for different networks or specific convolutional layers to achieve optimal outcomes. Our forthcoming work will prioritize investigating the distinctive features and characteristics of various convolutional layers to identify an optimal set of parameters for different layer types. This approach will empower users to readily apply our proposed multipliers across a spectrum of Convolutional Neural Networks, streamlining the implementation process and enhancing versatility.

REFERENCES

[1] B. Moons and M. Verhelst, ‘‘DVAS: Dynamic voltage accuracy scaling for increased energy-effificiency in approximate computing,’’ in Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED), Jul. 2018, pp. 237–242.
[2] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, ‘‘Design of voltage-scalable meta-functions for approximate computing,’’ in Proc.Design, Autom. Test Eur., Mar. 2019, pp. 1–6.
[3] K. Yin Kyaw, W. Ling Goh, and K. Seng Yeo, ‘‘Low-power high-speed multiplier for error-tolerant application,’’ in Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC), Dec. 2018, pp. 1–4.
[4] M. de la Guia Solaz, W. Han, and R. Conway, ‘‘A flexible low power DSP with a programmable truncated multiplier,’’ IEEE Trans. Circuits Syst., vol. 59, no. 11, pp. 2555–2568, Nov. 2019.
[5] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, ‘‘RoBa multiplier: A rounding-based approximate multiplier for highspeed yet energy-effificient digital signal processing,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 2, pp. 393–401, Feb. 2017.
[6] C. S. Wallace, ‘‘A suggestion for a fast multiplier,’’ IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 2015.
[7] A. Weinberger, ‘‘4:2 carry-save adder module,’’ IBM Tech. Discl. Bull., vol. 23, no. 8, pp. 3811–3814, 2017.
[8] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, ‘‘Design and analysis of approximate compressors for multiplication,’’ IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2019.
[9] Z. Yang, J. Han, and F. Lombardi, ‘‘Approximate compressors for error resilient multiplier design,’’ in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFTS), Oct. 2019, pp. 183–186.
[10] C.-H. Lin and I.-C. Lin, ‘‘High accuracy approximate multiplier with error correction,’’ in Proc. IEEE 31st Int. Conf. Comput. Design (ICCD), Oct. 2019, pp. 33–38.
[11] P. J. Edavoor, S. Raveendran, and A. D. Rahulkar, ‘‘Approximate multiplier design using novel dual-stage 4:2 compressors,’’ IEEE Access, vol. 8, pp. 48337–48351, 2020.
[12] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, ‘‘A majority-based imprecise multiplier for ultra-effificient approximate image multiplication,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 66, no. 11, pp. 4200–4208, Nov. 2019.
[13] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. D. Meo, ‘‘Comparison and extension of approximate 4–2 compressors for lowpower approximate multipliers,’’ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 9, pp. 3021–3034, Sep. 2020.
[14] H. Xiao, H. Xu, X. Chen, Y. Wang, and Y. Han, ‘‘Fast and high-accuracy approximate MAC unit design for CNN computing,’’ IEEE Embedded Syst. Lett., early access, Dec. 21, 2021, doi: 10.1109/LES.2021.3137335.
[15] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, ‘‘Dual-quality 4:2 compressors for utilizing in dynamic accuracy confifigurable multipliers,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4,pp. 1352–1361, Apr. 2021.
[16] F.-Y. Gu, I. -C. Lin and J. -W. Lin, “A Low-Power and High-AccuracyApproximate Multiplier with Reconfigurable Truncation”,in IEEE Access, vol.10,pp.60447-60458, 2022, doi: 10.1109/ACCESS.2022.3179113.
[17] T. Yang, T. Ukezono, and T. Sato, ‘‘A low-power high-speed accuracy controllable approximate multiplier design,’’ in Proc. 23rd Asia South Pacifific Design Autom. Conf. (ASP-DAC), Jan. 2018, pp. 605–610.
[18] I. Hammad, L. Li, K. El-Sankary, and W. M. Snelgrove, ‘‘CNN inference using a preprocessing precision controller and approximate multipliers with various precisions,’’ *IEEE Access*, vol. 9, pp. 7220–7232, 2021.
[19] C. Guo, L. Zhang, X. Zhou, W. Qian, and C. Zhuo, ‘‘A reconfifigurable approximate multiplier for quantized CNN applications,’’ in Proc.25th Asia South Pacifific Design Autom. Conf. (ASP-DAC), Jan. 2020, pp. 235–240.
[20] Jihee Seo and Dae Hyun Kim, “Dependency-Resolving Intra-Unit Pipeline Architecture for High-Throughput Multipliers”,IEEE Trans. 978-3-9819263-2-3/DATE19/c 2020 EDAA.
[21] Zahra Ebrahimi, Muhammad Zaid, Mark Wijtvliet, Akash Kumar, “RAPID: ApproximAte Pipelined Soft MultIpliers and Dividers for High-Throughput and Energy-Efficiency, IEEE Trans. rxiv:2206.13970v1 [cs.AR] 28 Jun 2022.