Received 30 January 2024, accepted 16 February 2024, date of publication 26 February 2024, date of current version 1 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3369695

RESEARCH ARTICLE

ADEPNET: A Dynamic-Precision Efficient Posit Multiplier for Neural Networks

ADITYA ANIRUDH JONNALAGADDA[1], UPPUGUNDURU ANIL KUMAR[2], RISHI THOTLI[1], SATVIK SARDESAI[1], SREEHARI VEERAMACHANENI[3], (Senior Member, IEEE), AND SYED ERSHAD AHMED[1],(Member, IEEE)

[1]Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science, Pilani, Hyderabad Campus, Hyderabad, Telangana 500078, India
[2]Department of Electronics and Communication Engineering, Faculty of Science and Technology (IcfaiTech), The ICFAI Foundation for Higher Education (Deemed to be University), Hyderabad, Telangana 501203, India
[3]Department of Electronics and Communication Engineering, Gokaraju Rangaraju Institute of Engineering and Technology (GRIET), Hyderabad, Telangana 500090, India

Corresponding author: Uppugunduru Anil Kumar (anilkumar.u@ifheindia.org)

ABSTRACT The posit number system aims to be a drop-in replacement of the existing IEEE floating-point standard. Its properties- tapered precision and high dynamic range, allow a smaller size posit to almost match the performance of a much larger size floating-point in representing decimals. This becomes especially useful for performing error-tolerant tasks like deep learning inference computation where low latency and area are a priority. Recent research has found that the performance of deep neural network models saturates beyond a certain level of accuracy of multipliers used for convolutions. Therefore, the extra hardware cost of developing precise arithmetic circuits for such applications becomes an unnecessary overhead. This paper explores approximate posit multipliers in the convolutional layers of deep neural networks and attempts to find an ideal balance between hardware utilization and inference accuracy. Posit multiplication involves several steps, with the mantissa multiplication step utilizing maximum hardware resources. To mitigate this, a posit multiplier circuit using an approximate hybrid-radix Booth encoding for mantissa multiplication and techniques such as truncation and bit masking based on input regime size are proposed. In addition, a novel Booth encoding control scheme to prevent unnecessary bits from switching has been devised to reduce dynamic power dissipation. Compared to existing literature, these optimizations have contributed to a 23% decrease in power dissipation in the mantissa multiplication stage. Further, a novel area and energy-efficient decoder architecture have also been developed with an 11% reduction in dynamic power dissipation and area compared to existing decoders. Overall, the proposed $< 16, 2 >$ posit multiplier offers a 14% reduction in the PDP over the existing approximate posit multiplier designs. The proposed $< 16, 2 >$ multiplier also achieves over 90% accuracy in inferencing deep learning models such as ResNet20, VGG-19 and DenseNet.

INDEX TERMS Approximate posit multipliers, deep neural networks, energy-efficient.

I. INTRODUCTION

The posit format, proposed in [1] is a Type III unum designed as a replacement for the existing IEEE 754 Floating Point Standard [2]. The posit format consists of variable bit-widths with no fixed size specified for the regime and mantissa fields. These regime bits act along with the exponent bits to produce an effective exponent that amplifies the dynamic range of posits in comparison to floats. The posit format displays an important property called tapered precision where smaller numbers can be more accurately represented as compared to larger numbers where precision is traded-off with greater dynamic range. This is again a consequence of the variable bit-widths of the posit format which allows it to represent smaller numbers more accurately as well as allowing the possibility of representing larger values compared to floats. The larger numbers utilize more regime bits than smaller numbers leaving a smaller number of mantissa bits for more precise representation. Though the tapered precision is favorable in most applications,

The associate editor coordinating the review of this manuscript and approving it for publication was Alex James.

the variable bit-widths also pose additional complexity for hardware design as considerable hardware is required to decode each field of the posit. Consequently, a posit arithmetic block comes at a greater cost than its floating-point analogue as shown in [3], [4] and [5]. This drawback has reduced interest in designing dedicated hardware units for posits.

Deep learning applications are known to have tolerance to lower numerical precision. High-precision computations are not especially useful in inferencing neural networks as the additional precision brings no advantage while at the same time being slower and less power efficient. The effects of lower precision can be nullified by regularization effects in these neural networks as per [6]. This observation is corroborated by the data presented in [7] where approximation techniques like truncation were utilized, inducing error into the multiplier. The inexact multiplier was utilized for inference computation in popular deep learning models like AlexNet [8], ResNet18 [9] and a loss of less than 1% accuracy was observed in inference computation compared to the exact multipliers.

As shown in the results section in this paper, the exact multipliers have an overhead of over 2.5× and 1.5× in power and area compared to the inexact multipliers in the mantissa multiplication process. For deep learning applications, the inexact multipliers have almost no loss in accuracy and also lead to significant reduction in power and area requirements. Therefore, the design of approximate posit multipliers has been undertaken in this paper.

The highlights of the proposed approximate multiplier include the design of

- A novel energy-efficient posit decoder architecture with intermediate control signals to reduce dynamic power dissipation.
- A hybrid-radix Booth encoded partial product array structure for efficient mantissa multiplication.
- A high-speed mask generation circuit for variable precision depending on mantissa size of posit.
- A Booth encoding control scheme in the partial product generation (PPG) to reduce dynamic power dissipation.

This paper is organized as follows. Section II presents the related work describing the posit standard and posit multipliers in existing literature. The proposed work highlighted in Section III details the contributions of the developed multiplier. Section IV describes the results consisting of the hardware synthesis results and error metrics of the proposed posit multiplier. Section V presents the applications section where the proposed multiplier is deployed in the convolutional layers of popular deep neural network models to test its effectiveness. Finally, Section VI concludes the paper.

II. RELATED WORK

The related work has been organized into 2 sections- posit number system and posit multipliers in which the posit format and posit multipliers in existing literature have been briefly highlighted.

A. POSIT FORMAT

A posit number consists of sign, regime, exponent and mantissa bits. The major difference between IEEE 754 floating-point standard and the posit standard is the variable bit-widths of posits.

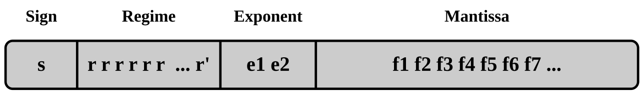


FIGURE 1. Representation of a general $< N, 2 >$ posit.

A general n-bit posit representation is shown in Fig.1. The sign bit at the MSB is a single bit which is 1 when the posit is negative and 0 when it is positive. The regime field is a string of identical bits terminated by their inverted bit. In instances where the regime vector consists of ‘r’ consecutive zeros, the corresponding regime value (rg) is denoted as ‘-r’. Conversely, when the regime vector exhibits ‘r’ consecutive ones, the regime value (rg) is represented as ‘r-1’. The exponent field generally consists of ‘es’ number of bits, however this has been fixed to 2 bits as per the New Posit Standard of 2022 [10]. The regime and the exponent field together form an effective exponent calculated as rg $\times$ 2^{es}+exp where ‘rg’ is the regime value, exp is the exponent value and ‘es’ is the number of bits in the exponent field. The fraction field fills the remaining bits of the n-bit posit.

The posit value is obtained from the following equation.

$$value = (-1)^s * useed^{rg} * 2^{exp} * (1 + frac) \qquad (1)$$

where useed $= 2^{2^{es}}$

B. POSIT MULTIPLICATION

In the existing literature, there has been a lot of emphasis on reducing the hardware cost of posit arithmetic units to make them more area and energy efficient. The mantissa or fraction multiplier is often the most expensive in terms of area and power and thus most of the prevailing designs have prioritized the optimization of this process. The existing literature has been classified into exact and inexact multipliers.

The exact posit multiplier proposed in [11] utilizes the same mantissa multiplication process as a standard posit multiplier with an optimized leading one detector circuit in the decoder stage to reduce the dynamic power dissipation. The exact posit multiplier (PEfPM) designed in [12] uses control logic in the PPG stage to decompose the mantissa multiplier circuit into 16 smaller regions using 4 vertical and 4 horizontal control signals. The control logic enabled only the portions of the circuit required for computation based on the regime size of the multiplier and multiplicand in an attempt to minimize dynamic power dissipation in the circuit. The multipliers (EFPM P1 and P2) proposed in [13]

TABLE 1. Comparison of various existing posit multipliers.

Design	Size	Type	Algorithm	Approximation	Power	Area	Delay	Accuracy
PEfPM [12]	8,16,32	Exact	Booth	NA	High	High	High	Fully Accurate
EFPM [13]	16	Exact	Booth	NA	High	High	High	Fully Accurate
IPM [14]	32	Inexact	Iterative	Truncation	Moderate	Low	High	Moderate
PLAM [15]	32	Inexact	Logarithm	Logarithm	Low	Low	Low	Low
Posit-VAR [7]	8,16,32	Inexact	Booth	Variable Truncation	Low	Low	Moderate	Moderate

were precise multipliers with more optimized control logic schemes with advantages over the multiplier in [12].

The multiplier designed in [14] was an iterative shift-based posit multiplier (IPM) and was the first to explore truncation as an approximation technique in posit multiplication. The proposed 32-bit multiplier utilized 12 mantissa bits, discarding the remaining bits to save area. The design in [15] was an approximate logarithmic multiplier (PLAM) based on Mitchell's Algorithm [16]. The PLAM design was hardware efficient but also induced considerable error. The Posit-VAR [7] employs a radix-8 Booth algorithm for approximate multiplication, featuring a unique variable precision control module that adapts based on the regime size of the product posit, enhancing precision as needed. Additionally, it incorporates unconditional truncation of partial product bits of lower significance. The work proposed in the paper highlighted that inexact designs offer significant advantages in area, power and delay with very minimal overhead in accuracy of inference computation of common deep learning models. This observation has led to increased research in approximating the posit multiplication process and has provided the motivation for the proposed work. The properties of the existing exact and inexact posit multipliers in literature have been highlighted in Table 1.

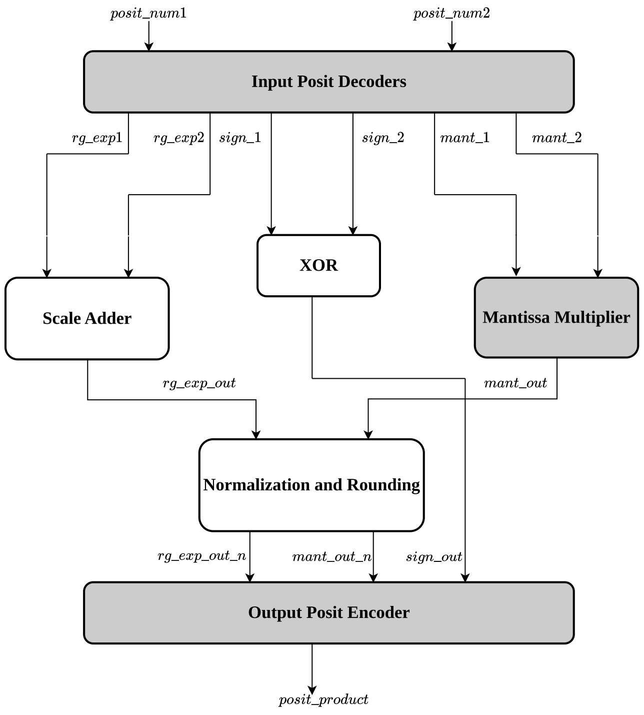


FIGURE 2. Proposed posit multiplier datapath.

III. PROPOSED WORK

The proposed work involves the design of an approximate posit multiplier (ADEPNET) circuit, specifically for the $< 16, 2 >$ posit format. Any mathematical operation on a posit involves decoding and encoding in conjunction with the specific arithmetic operation itself. In the case of posit multiplication, in addition to the standard mantissa multiplication, effective exponent addition and sign calculation, posit decoding and encoding are also performed. These decoding and encoding circuits are an inherent feature of posit arithmetic and are absent in floating-point operations. These extra steps are a result of the variable bit-widths of the different fields in a posit, unlike a floating-point number where each field occupies a fixed number of bits.

The entire posit multiplication datapath highlighting all the steps from initial decoding to final encoding is shown in Fig.2. The proposed posit decoder circuit and mantissa multiplication circuit have been highlighted in grey. A posit decoder circuit is used to break an $< n, es >$ posit into its constituent parts namely sign, regime, exponent and mantissa. The regime in conjunction with the exponent field form an effective exponent which is calculated as rg $\times$ 2^{es}+exp where rg is the regime value, exp is the exponent value and es is the number of bits in the exponent field. Once the effective exponent has been calculated, the effective exponents of the 2 posits are added to produce the resultant effective exponent. The individual sign bits are XORed to obtain the resultant sign bit and the mantissas are multiplied to form the resultant mantissa. The mantissa and exponent undergo normalization and finally the resultant mantissa, effective exponent and sign bits are packed together by the encoder to generate the resultant posit.

A. PROPOSED POSIT DECODER

The posit decoder shown in Fig. 3 stands as a fundamental requisite for the execution of any posit-based arithmetic operation. In contrast to floating-point operations which do not necessitate this step, the optimization of decoding overhead becomes important in order for posit arithmetic circuits to match or exceed their floating-point counterparts in performance metrics.

In prevailing design paradigms, the input undergoes a twos complement operation when the sign bit is denoted as 1. Following this, subsequent processing involving an

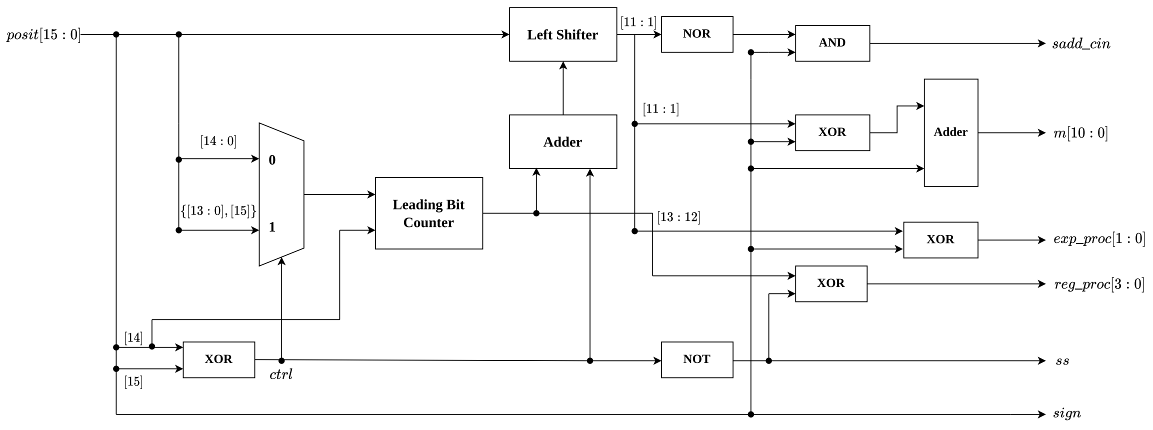


FIGURE 3. **Proposed < 16, 2 > posit decoder architecture.**

additional XOR operation with the MSB of the regime field is done to extract the effective exponent and mantissa values. The effective exponent refers to the total exponent of the input posit from both the regime and exponent fields and not just the contribution of the exponent field. In the proposed design framework, a portion of the processing is done before performing the twos complement operation where intermediate control signals are integrated to account for all input scenarios. This approach leads to a reduction in the complementing adder width and increasing the overlap between the exponent and mantissa calculation.

The proposed approach also extends its utility by reducing the overall delay of the multiplier. This is accomplished by making the regime value available sooner, which the mantissa multiplier requires as an input. When decoding the regime field of posits, there are four cases where the sign bit and first regime bit are (0,0), (0,1), (1,0), and (1,1), respectively. In cases (0,0) and (1,1), the regime field contributes a factor of '-p' to the effective exponent, where 'p' represents the number of times the first regime bit repeats before being terminated by the opposite bit. Alternatively, in cases (0,1) and (1,0), it contributes a factor of 'p - 1'. For hardware efficiency, the control signal 'ctrl' is triggered upon detection of (0,1) or (1,0) cases, eliminating the need for an extra adder. Upon setting the 'ctrl' bit, the input to the leading bit counter is shifted left by one bit, decrementing the count by one, and the sign bit is then appended to the end of the modified input. Since the sign bit in these cases is always dissimilar to the first regime bit, it can act as the terminating bit when the rest of the input is all ones or zeros.

This intermediary control signal 'ctrl' is generated by XORing the sign bit and the most significant regime field bit, which allows for the change in order of processing the input. The intermediate control signal derived from the sign bit and first regime bit acts as a select input to the multiplexer to determine the appropriate signal to route to the leading bit counter. When the sign bit and the first regime bit are different, the left-shifted posit is taken as the input. Conversely, when the two bits are identical, the unmodified posit bits are selected. Moreover, the significance of the intermediate control signal extends beyond the multiplexer; it also serves as an input to an adder. The intermediate control signal, coupled with the leading bit counter output, serves as inputs to the adder. Together, they determine the precise shift amount crucial for accurately decoding the exponent and mantissa fields in the input posit. After the shifting operation, the exponent and mantissa fields are extracted and processed depending on the sign bit. A new signal 'sadd_cin' is generated by performing NOR reduction on the pre-processed mantissa bits, which is used as the carry-in to the adder in the exponent processing step later. The signal 'ss' serves as the sign of the effective exponent.

B. PROPOSED MANTISSA MULTIPLIER

The mantissa multiplier circuit has been designed for a 12-bit operand size, as this is the maximum mantissa size of a $< 16, 2 >$ posit. The same methodology has been utilized for the $< 8, 0 >$ posit as well. In the past, there has been a lot of emphasis on reducing the hardware cost of the mantissa multiplication stage as it contributes to the maximum area, power, and delay of the circuit. This paper explores several approximation techniques and power optimizations to be applied in the mantissa multiplication process. The overall mantissa multiplication process has been divided into the following sections with different contributions in each section:

1. Hybrid-Radix Booth Encoded Partial Product Array
2. Mask Generation Module
3. Booth Encoding Control

1) HYBRID-RADIX BOOTH ENCODED PARTIAL PRODUCT ARRAY

The mantissa multiplication process utilizes the Booth multiplication algorithm, however unlike existing posit multipliers it does not use a single radix encoding scheme. As a result, a hybrid Booth encoding has been proposed with the aim

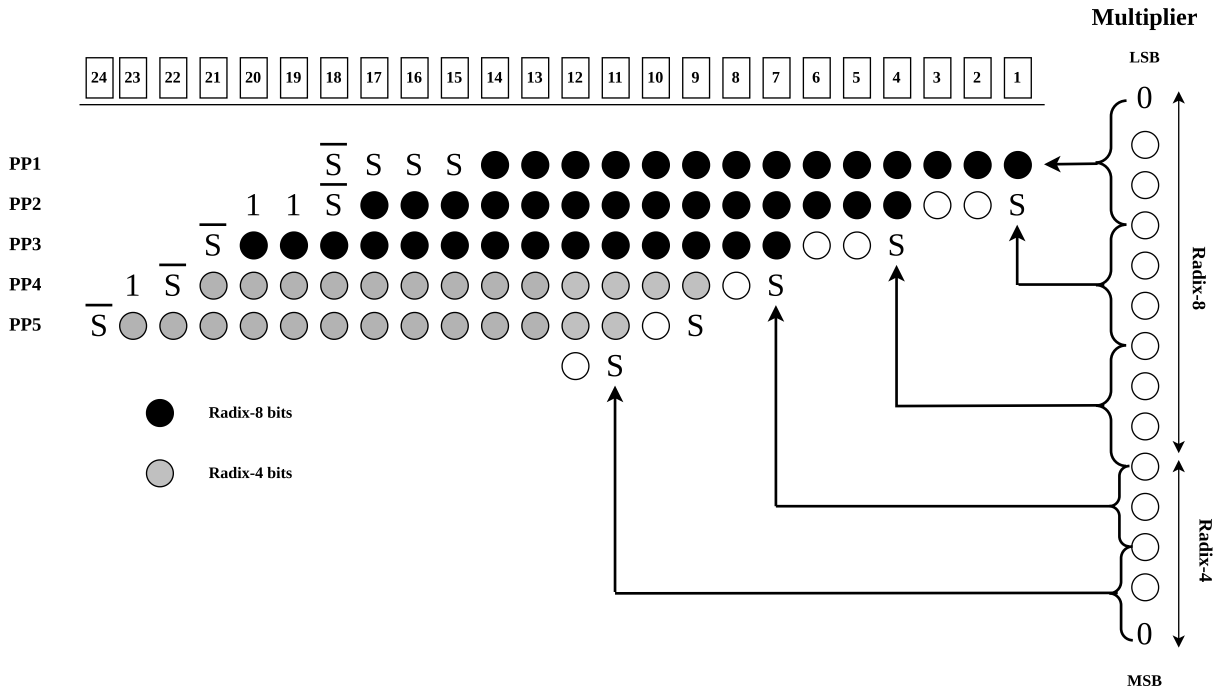


FIGURE 4. Hybrid-radix encoded partial product array.

to combine the individual advantages of the radix-4 and radix-8 scheme as well as overcome their individual inherent limitations.

The radix-8 encoding allows for a simpler reduction of partial product rows compared to radix-4 owing to a smaller number of partial product rows. However, it also involves the computation of three times the multiplicand, which can only be achieved by an addition process. While the advantage of fewer partial products exists, the delay introduced by addition in radix-8, which is not present in radix-4 encoding, offsets this benefit. Approximating the addition process in a radix-8 can reduce the impact of addition. However, this also induces significant errors in the system. Therefore, instead of a purely radix-8 encoding, a hybrid Booth encoding scheme has been devised with the first three partial product rows encoded using radix-8 and the next two partial product rows encoded using an exact radix-4 scheme. The hybrid encoding scheme shown in Fig.4 ensures that the more significant partial product rows are exactly computed to reduce the system's error without increasing the number of partial products compared to a purely radix-8 design.

The extended sign bits and '1' bits observed in each partial product row in Fig.4 compose the sign extension of that particular row. In the traditional Booth process, with the multiplicand being n-bits wide, all the partial products are sign extended such that the final bit position of each partial product row corresponds to a column number of '2n'. Since this extension is not area-efficient, reduced sign extensions for different partial product rows have been developed for both radix-4 and radix-8 Booth multipliers, as shown in Fig.4.

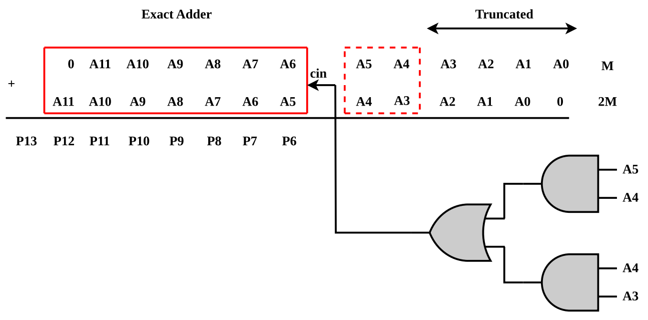

Exact Adder
Truncated
cin
+
0 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 M
A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 0 2M
P13 P12 P11 P10 P9 P8 P7 P6
A5
A4
A4
A3

FIGURE 5. Approximation for 3M partial product calculation with carry prediction scheme.

The radix-8 scheme involves the calculation of three times the multiplicand (3M) which is not as straightforward as computing twice the multiplicand (2M) and can easily be accomplished with a left shift operation. As a result, the 3M partial product calculation has been approximated rather than computing the exact sum. Instead of a 12-bit addition between M and 2M, a 7-bit exact addition on the MSB side of M and 2M has been performed with a carry-in prediction circuit as shown in Fig.5. The approximation brings about sizable reduction in overall power, delay and area of the circuit. The P6 bit in the 3M partial product calculation corresponds to the least significant bit which is left untruncated in the partial product array. The multiplicand bits A11-A5 are the bits used for the sum calculation. The error in the 3M partial product calculation arises because the carry-in calculation has not been performed. To overcome this, a carry-in prediction

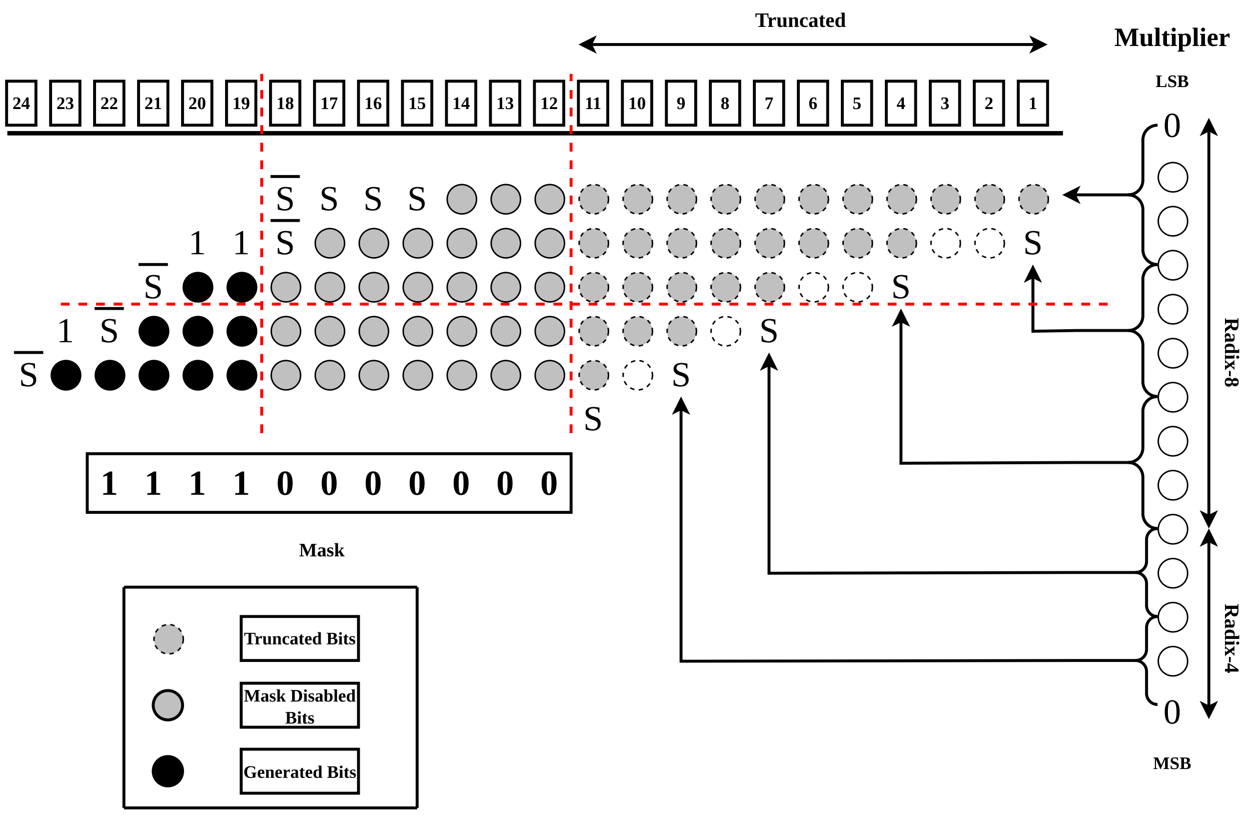


FIGURE 6. **Proposed mantissa multiplier with bit masking.**

circuit has been proposed to predict the carry-in to the 7-bit addition utilizing the A5, A4, A3 bits of the multiplicand. If any combination of the following pairs- A5, A4 and A4, A3 are both at a logic high, the carry-in prediction circuit predicts a logic one.

2) MASK GENERATION CIRCUIT

For a $< 16, 2 >$ posit, the maximum number of bits in the mantissa field, including the implied bit, is 12. When the mantissa multiplication operation is performed, the mantissa of the product is 24 bits as the size of the individual mantissa operands are 12 each. Since the mantissa size is limited to 12 bits, the entire multiplication process is unnecessary and partial product bits can be truncated such that only the partial product bits corresponding to the 13 most significant columns are generated. Even though the posit mantissa size is limited to a size of 12-bits, 13 columns of the partial product array have been generated for normalization and rounding purposes as shown in Fig.6.

In the case of posits, unconditional truncation of the columns on the LSB side is not the only optimization to reduce dynamic power. The size of the mantissa field in a posit decreases with an increase in the size of the regime field. Therefore, when the regime size is at its minimum value, the size of the mantissa field is at its maximum. The unconditional truncation only accounts for when the maximum size of the mantissa is utilized. When the size of the mantissa field of the product mantissa is smaller, additional columns of the partial product array can be masked in addition to the existing truncation as shown in Fig.6.

This brings about a dynamic precision with varying levels of approximation depending on the final regime size of the product posit. This is accomplished using a mask generation circuit which is used to generate a signal in the partial product generation stage to disable the partial product bits based on the width of the regime field. In a $< 16, 2 >$ posit, the regime width can vary from 2-bits to 15-bits. Since the exponent size is fixed to 2-bits, the number of mantissa bits depends solely on the width of the regime field. Taking an example when the regime value is 1, the regime field width is 3 and the number of mantissa bits in the posit is 10 (excluding the implied bit). The maximum mantissa size is 11 and thus the least significant mantissa bit in this case can be masked to save power. The relation between the regime field value, regime field width, mantissa field width and corresponding mask value is highlighted in Table 2.

The generated mask values are sent as column-wise enable signals to the partial product array to reduce dynamic power dissipation. The partial product array shown in Fig.6 combines the unconditional truncation of the columns of lower columns and mask based enabling and disabling of other columns based on the size of the regime field. In the example highlighted in Fig.6, the regime value of the product is calculated as 7 which corresponds to a regime field size

TABLE 2. Mask signal generation truth table.

Reg_Val [3:0]		Reg Width	Mant Width	Mask String
Reg_Val[4] = 0	Reg_Val[4] =1			
0000	1111	2	11	11111111111
0001	1110	3	10	11111111110
0010	1101	4	9	11111111100
0011	1100	5	8	11111111000
0100	1011	6	7	11111110000
0101	1010	7	6	11111100000
0110	1001	8	5	11111000000
0111	1000	9	4	11110000000
1000	0111	10	3	11100000000
1001	0110	11	2	11000000000
1010	0101	12	1	10000000000
1011	0100	13	0	00000000000
1100	0011	14	0	00000000000
1101	0010	15	0	00000000000
1110	0001	15	0	00000000000
1111	0000	15	0	00000000000

of 9-bits and a mantissa size of 4-bits excluding the implied bit. In this case, the mask enables the 4-bits of the mantissa product on the MSB side and disables the 7-bits towards the LSB side. The most significant column is always left unmasked as it corresponds to the implied bit of the mantissa.

From Table 2, it is evident that the mask can be derived by performing a simple left shift operation on a string consisting of '11' ones with the shift amount dependent upon the regime value of the product posit. If the regime value is positive, then the 4 bits on the LSB side of the regime value field are set as the shift amount for the left shifter circuit. In the case that the regime value is negative, the one's complement of the 4 bits on the LSB side of the regime value are utilized as the shift amount. A 11-bit barrel shifter circuit has been utilized for the mask generation. The mask generation differs from the circuit proposed in [7], where the circuit has been developed from Boolean logic equations derived from the truth table. The proposed circuit utilizes a simple left-shifter based architecture which can easily be scaled appropriately for any general size posit. The proposed mask generation circuit has comparable area and power to the circuit in [7], however it brings about an improvement in delay compared to the circuit in [7].

3) BOOTH ENCODING CONTROL

The mask circuit has been used to enable and disable partial product bits after they have been generated which reduces dynamic power dissipation. However, there is further scope to reduce the dynamic power even further by generating enable signals from the mask generation circuit to the Booth encoding itself. In some cases, the mask disables all the partial product bits of a particular partial product row and thus these bits play no role in the calculation of the product. Generating these bits from the Booth encoding and PPG blocks is a waste of power as they will anyways be masked. If the Booth encoding for the entire row of partial products can be disabled, the output of the PPG blocks are constrained to a logic low, saving even more power.

For example, if the mask value is '11111111000', the columns 12, 13 and 14 are disabled by the mask circuit. Since the MSB of the first partial product row belongs to column 14, there is no partial product bit in the row that is enabled by the mask. If the Booth encoding which is common to a particular row of partial products is disabled, then all the partial product bits are set to a logic low. However, in case the mask value is '11111111110', the Booth encoding for the first partial product row cannot be disabled as the MSB of the row belonging to column 12 is required for the product calculation and cannot be disabled to a logic low. Therefore for particular mask values only, the Booth encoding can be gated off.

The mask bits corresponding to columns 14, 17, 20, 21 and 23 have been used as enable signals to the Booth encoding circuits of rows 1, 2, 3, 4 and 5 respectively. The mask bit corresponding to the column where the MSB of a particular row is present has been used as the enable signal for the Booth encoding for the particular row. This has been done as the mask is a thermometric string- if a particular bit is at a logic 0, every bit to its right is also at a logic 0. Therefore if the mask bit corresponding to the MSB of a particular row is 0, the mask bits corresponding to all the other bits in the row will also be 0. In this case the optimization of disabling the Booth encoding for the partial product row is possible.

A Booth encoder circuit followed by a PPG block has been shown in Fig. 7. Based on the multiplier bits 'x[2:0]', the encoding is decided. If all the signals- 'one', 'two' and 'neg' are at a logic low, the partial product bit is set to a logic low and cannot toggle. This reduces the dynamic power as instead of generating the actual partial product bit and then masking it, the partial product bit is disabled at the encoding stage itself. It is important to note that the extended sign bits of a particular partial product row are retained in the partial product array even though an entire row of partial product bits is disabled. Instead of using the 'neg' signal set to 0, the input bit 'x[2]' to the Booth encoder circuit, as shown in Fig.7, is taken as the sign bit. This ensures that the sign bits are still part of the partial product irrespective of the Booth encoding control.

IV. RESULTS

A. HARDWARE SYNTHESIS RESULTS

For a fair comparison between proposed and existing designs, Verilog HDL has been utilized. The area, power and delay reports of existing designs PEfPM [12], Posit [1], Posit-VAR [7] and the proposed ADEPNET have been generated using the Cadence RTL compiler v7.1 at a TSMC 90 nm process node. The synthesis has been performed specifically utilizing the slow-normal library under the operating conditions of 1.8V, 25 C and frequency 1 GHz.

A comprehensive examination of exact and approximate mantissa multipliers is presented in Table 3, alongside the findings from the proposed work. Given that mantissa multiplication constitutes the most resource-intensive operation in a posit multiplier, the focus in Table 3 is solely on the results

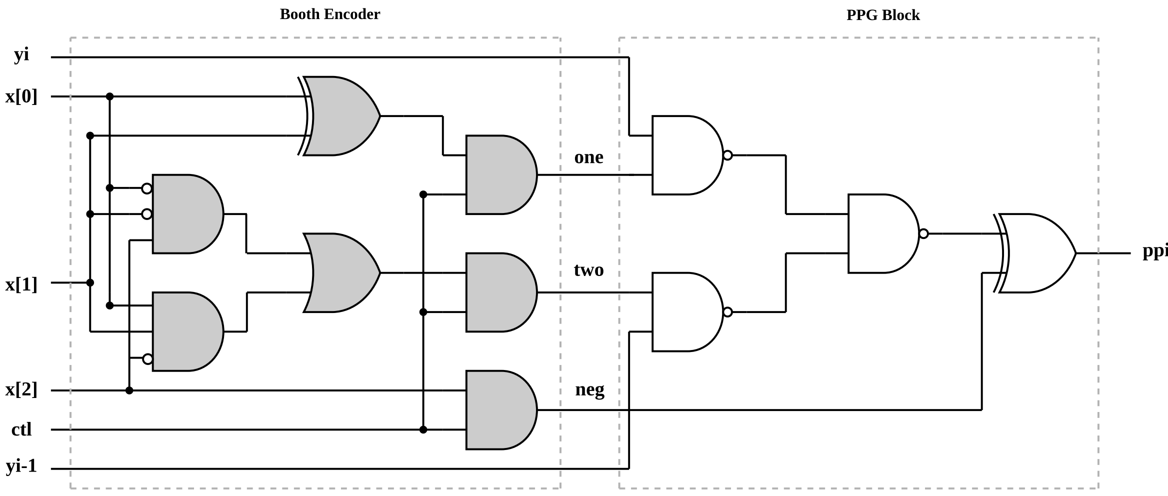


FIGURE 7. Booth encoding control.

TABLE 3. Existing and proposed mantissa multiplier synthesis results.

Posit Size	Multiplier Type	Design	Area(μm^2)	Area Ratio	Power(μ W)	Power Ratio	Delay(ns)	Delay Ratio	PDP Ratio
16-bit	Exact	PEfPM [12]	2771	1.78	145.678	3.41	4.461	1.37	4.671
	Exact	Posit [1]	2546	1.64	159.321	3.73	4.412	1.35	5.036
	Inexact	Posit-VAR [7]	1555	1	42.682	1	3.254	1	1
	Inexact	**ADEPNET(no BE ctrl)**	**1495**	**0.96**	**39.614**	**0.93**	**3.121**	**0.96**	**0.89**
	Inexact	**ADEPNET(BE ctrl)**	**1501**	**0.96**	**32.923**	**0.77**	**3.429**	**1.05**	**0.81**
8-bit	Exact	PEfPM [12]	806	1.62	28.032	2.44	2.456	1.25	3.05
	Exact	Posit [1]	756	1.53	26.271	2.29	2.382	1.22	2.79
	Inexact	Posit-VAR [7]	495	1	11.496	1	1.959	1	1
	Inexact	**ADEPNET(no BE ctrl)**	**500**	**1.02**	**10.613**	**0.92**	**1.892**	**0.97**	**0.89**
	Inexact	**ADEPNET(BE ctrl)**	**505**	**1.02**	**8.558**	**0.74**	**2.209**	**1.13**	**0.84**

of mantissa multiplication. This emphasizes the apparent reduction in hardware resource utilization achieved by the approximate multipliers when contrasted with their exact analogues. Among all the 16-bit posit multiplier designs PEfPM [12] and Posit [1] are exact posit multipliers and Posit-VAR [7], the proposed ADEPNET are approximate posit multipliers. It is evident that compared to the exact mantissa multipliers, the inexact mantissa multiplier of Posit-VAR [7] offers a minimum reduction in area, power, and delay of 39%, 71%, 26% respectively. These gains in power, area and delay are further extended by the proposed ADEPNET design. Both the ADEPNET and Posit-VAR utilize truncation where the size of the partial product array is nearly half that of the existing exact designs contributing to the significant reduction in area. The unconditional truncation along with the masking of unnecessary partial product bits depending on the regime size of the posit reduces the switching activity of the partial product bits in the circuit, resulting in a significant fall in dynamic power dissipation. Apart from the truncation and masking, the ADEPNET utilizes a hybrid radix encoded partial product array structure with an approximation in the calculation of the 3M partial product partial product and a Booth encoding control. These additions help in reducing the power dissipation of the ADEPNET design compared to the existing Posit-VAR design by 23% in the mantissa multiplication. However,the Booth encoding control brings an overhead of 5% in delay compared to the Posit-VAR [7]. This overhead has been compensated by the proposed decoder structure and the mask generation circuit. The introduction of the Booth encoding control in the proposed design reduces the power dissipation by 16.9%, however it also contributes to a 9.8% delay overhead as shown in Table 3. This is because the Booth encoding cannot take place until the enable signals from the mask generation circuit are available. There is no significant difference in the area requirements of the circuits with and without the encoding control. The 8-bit results follow a similar pattern to the 16-bit results.

The 16-bit synthesis results for the entire posit multiplication datapath of the existing exact PEfPM [12] design, existing inexact Posit-VAR [7] and the proposed ADEPNET along with a 32-bit floating-point multiplier have been depicted in Table 4. Since, 16-bit floating-points do not have the precision to be used for practical applications, the 16-bit posit multipliers have been compared with a 32-bit float. Since floating-point numbers do not require the extra steps

TABLE 4. Area, power and delay comparison between floating-point and posit multipliers.

Type	Design	Module	Area(μm^2)	Area Ratio	Power(μ W)	Power Ratio	Delay(ns)	Delay Ratio	PDP(fJ)	PDP Ratio
Exact	IEEE-FP32 [2]	Input Decoding	NA	NA	NA	NA	NA	NA		NA
		Control Circuit	NA	NA	NA	NA	NA	NA		NA
		Mantissa Multiplication	6942	2.51	653.372	4.49	7.524	1.69		7.59
		Exponent Addition	141	1.21	4.481	1.38	1.307			
		Encoder	NA	NA	NA	NA	NA	NA		NA
		Total	**7083**	**1.89**	**657.853**	**6.32**	**8.831**	**0.78**	**5809.49**	**4.93**
Exact	PEfPM [12]	Input Decoding	1521	1.08	44.113	1.1	3.551	1		1.1
		Control Circuit	26	NA	0.527	NA	0.134	NA		NA
		Mantissa Multiplication	2771	1.78	145.678	3.41	4.461	1.37		4.67
		Exponent Addition	117	1	3.256	1	0.978	1		1
		Encoder	811	1.35	19.175	1.16	3.165	1.04		1.21
		Total	**5246**	**1.39**	**212.749**	**2.04**	**12.289**	**1.08**	**2614.47**	**2.21**
Inexact	Posit-VAR [7]	Input Decoding	1404	1	40.024	1	3.518	1		1
		Mask Generation	72	1	1.587	1	0.55	1		1
		Mantissa	1555	1	42.682	1	3.254	1		1
		Exponent Addition	117	1	3.256	1	0.978	1		1
		Output Encoding	601	1	16.516	1	3.052	1		1
		Total	**3749**	**1**	**104.065**	**1**	**11.352**	**1**	**1181.35**	1
Inexact	**ADEPNET**	Input Decoding	1250	0.89	35.98	0.89	3.37	0.96		0.85
		Mask Generation	95	1.32	1.848	1.14	0.393	0.71		0.93
		Mantissa	1501	0.96	32.923	0.77	3.429	1.05		0.81
		Exponent Addition	117	1	3.256	1	0.978	1		1
		Output Encoding	601	1	16.516	1	3.052	1		1
		Total	**3564**	**0.95**	**90.523**	**0.87**	**11.222**	**0.99**	**1015.85**	**0.86**

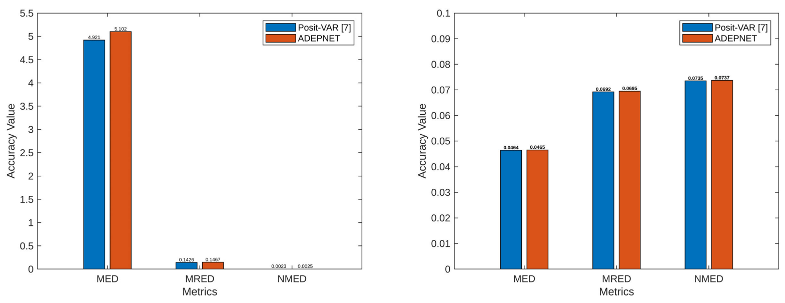
of decoding and encoding, floating-point multiplication is 23% faster than the proposed design. However, owing to a larger input size, they take up more than 7 times the power of the proposed design. The proposed decoder circuit has been optimized for power and area, offering a reduction of 11% in both power and area respectively compared to the Posit-VAR [7]. The proposed decoder also offers a minor advantage of 4.2% in delay compared to the Posit-VAR [7] design. The proposed mask generation circuit has been optimized for delay to compensate for the delay overhead of the proposed mantissa multiplication. The percentage improvement in delay of the mask generation circuit over the Posit-VAR design is over 28%. The lower delay of the mask generation circuit and the decoder circuit of the ADEPNET ensures that the delay of the overall design remains less than the delay of the Posit-VAR [7]. The encoder and exponent addition circuits that have been utilized in the ADEPNET are the same as the ones used in the Posit-VAR design [7]. Overall, the ADEPNET design provides a 5% reduction in area, 13% reduction in power and 0.8% reduction in delay compared to the Posit-VAR [7] design contributing to a 14% reduction in the overall PDP.

B. ERROR ANALYSIS

From Table 4 it is evident that the proposed ADEPNET design provides advantages in area, power and delay compared to the Posit-VAR, however there is also a slight degradation in accuracy of compared to the existing Posit-VAR design. This loss in accuracy compared to the Posit-VAR design is because of the extra approximation of the 3M partial product calculation for the radix-8 encoded partial products. The error metrics for the Posit-VAR and ADEPNET have been computed and highlighted in Fig.8. For the error metrics computation, over 4 million random combinations of multiplier and multiplicand $< 16, 2 >$ posits have been taken into consideration using the SoftPosit library in python. For the $< 8, 0 >$ posit configuration, comprehensive testing has been performed because of a smaller sample size compared to the $< 16, 2 >$ configuration. The Mean Error Distance(MED), Mean Relative Error Distance(MRED) and Normalized Mean Error Distance(NMED) values have been computed for all three designs. From Fig.8 it is evident that the loss in accuracy is very minimal, especially in comparison to the advantages gained in the area, power, and delay. In the 16-bit case, the accuracy degradation of the ADEPNET is 0.2%, 0.4% and 0.27% in the MED, MRED and NMED parameters respectively compared to the Posit-VAR design. The 8-bit design has an increase of 3%, 2.9% and 8.6% in the MED, MRED and NMED values respectively. It is apparent from Table 4 and Fig.8 that the drop off in accuracy is a rational trade-off for reduction of hardware complexity.

V. BENCHMARKING APPLICATIONS

Standard deep learning models such as AlexNet [8], ResNet18 [9], VGG-19 [18], LeNet5 [19], DenseNet [20] utilize the 32-bit single precision floating point [2] to represent the biases and weights in their convolutional layers. The 16-bit half precision floating-point has significantly lower precision compared to a 32-bit floating-point and thus is not used to perform inference. However, since the 16-bit floating-point reduces the memory utilization and latency a mixed approach of using both the formats is being explored as shown in [21]. Since it has been established that a smaller posit has similar precision compared to a larger



(a) $< 8, 0 >$ accuracy results (b) $< 16, 2 >$ accuracy results

FIGURE 8. **Accuracy results with different posit configurations.**

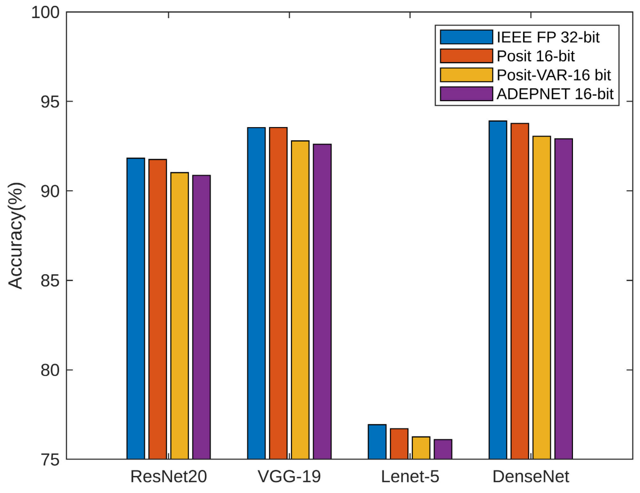


FIGURE 9. **Deep learning inference calculation accuracy.**

floating-point, all the weights and biases of popular deep learning models have been encoded into a $< 16, 2 >$ posit format and a 16-bit posit multiplier has been deployed in the convolutional layers of these models.

Since deep learning applications have inherent immunity against approximation errors, the effect of the error degradation of the ADEPNET versus the 32-bit floating point multiplier and the Posit-VAR design is expected to be reduced when both the designs are used for inference computation in deep learning applications. This expectation has been corroborated by the experimental data highlighted in Fig.9. The exact 32-bit floating-point multiplier, exact 16-bit posit multiplier, inexact Posit-VAR and the proposed ADEPNET designs are deployed in the deep learning models trained with the CIFAR-10 test set using the PyTorch framework and inference computation is performed. The results of the experiment are highlighted in Fig.9. From the data in Fig.9, it is clear that the 16-bit posit closely matches a 32-bit floating-point in inference accuracy and thus the proposed ADEPNET shows very similar performance in inference computation to that of the exact 16-bit posit multiplier and 32-bit floating point multiplier with the loss of accuracy in inference being less than 1% in all the deep learning models highlighted. Since the proposed design offers substantial improvements in the area, power and delay compared to the existing floating-point and posit multipliers without significant loss in accuracy while performing inference computation, it is an optimum design for deep learning applications.

VI. CONCLUSION

This paper explores approximate posit multipliers as an alternative to exact posit multipliers and floating-points for deep learning applications. The higher precision posits offers have been leveraged to develop an approximate multiplier circuit to reduce hardware resource consumption. By utilizing approximations like truncation, bit masking, and introducing a novel Booth encoding control scheme in the mantissa multiplication process, the proposed approximate design demonstrates a power dissipation of less than 23%, along with a sizeable 46% reduction in area compared to the exact PEfPM design in the mantissa multiplication stage. The proposed approximations lead to significant reduction in hardware complexity and do not considerably affect the accuracy of inference of deep learning models, as shown in the results section, where the approximate design had less than 1% accuracy degradation in inference accuracy compared to the exact posit multipliers. In the future, a mixed usage of $< 16, 2 >$ and $< 8, 0 >$ posits will be explored in an attempt to increase hardware efficiency of inference computation without degrading the accuracy significantly.

ACKNOWLEDGMENT

(Aditya Anirudh Jonnalagadda and Uppugunduru Anil Kumar contributed equally to this work.)

REFERENCES

[1] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, 2017.
[2] *IEEE Standard for Floating-Point Arithmetic—Redline*, Standard IEEE Std 754-2008, Aug. 2008, pp. 1–82, doi: 10.1109/IEEESTD.2008.5976968.
[3] A. Podobas and S. Matsuoka, "Hardware implementation of POSITs and their application in FPGAs," in *Proc. IPDPSW*, May 2018, pp. 138–145.
[4] Y. Uguen, L. Forget, and F. de Dinechin, "Evaluating the hardware cost of the posit number system," in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 106–113.

[5] H. Zhang, J. He, and S.-B. Ko, "Efficient posit multiply-accumulate unit generator for deep learning applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[6] Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, "Performance-efficiency trade-off of low-precision numerical formats in deep neural networks," in *Proc. Conf. Next Gener. Arithmetic*, 2019, pp. 1–9.

[7] H. Zhang and S.-B. Ko, "Efficient approximate posit multipliers for deep learning computation," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 1, pp. 201–211, Mar. 2023, doi: 10.1109/JETCAS.2022.3231642.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, vol. 1, 2012, pp. 1097–1105.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[10] Posit Working Group, *Posit Standard Documentation, Release 3.2-Draft*, 2018. Accessed: Sep. 24, 2020. [Online]. Available: https://posithub.org/docs/posit standard.pdf

[11] R. S. Haripriya, "Design of energy efficient and low delay posit multiplier," in *Proc. 36th Int. Conf. VLSI Des. 22nd Int. Conf. Embedded Syst. (VLSID)*, 2023, pp. 1–9.

[12] H. Zhang and S.-B. Ko, "Design of power efficient posit multiplier," *IEEE Trans. Circuits Syst. II, Exp. Papers*, vol. 67, no. 5, pp. 861–865, May 2020, doi: 10.1109/TCSII.2020.2980531.

[13] A. A. Jonnalagadda, A. K. Uppugunduru, S. Veeramachaneni, and S. E. Ahmed, "Design of energy efficient posit multiplier," in *Proc. Great Lakes Symp. VLSI*, Jun. 2023, pp. 1–6.

[14] C. J. Norris and S. Kim, "An approximate and iterative posit multiplier architecture for FPGAs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jul. 2021, pp. 1–5.

[15] R. Murillo, A. A. Del Barrio, G. Botella, M. S. Kim, H. Kim, and N. Bagherzadeh, "PLAM: A posit logarithm-approximate multiplier," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 2079–2085, Oct. 2022, doi: 10.1109/TETC.2021.3109127.

[16] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, vols. EC–11, no. 4, pp. 512–517, Aug. 1962.

[17] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, no. 2, pp. 236–240, 1951.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.

[21] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," 2017, *arXiv:1710.03740*.

ADITYA ANIRUDH JONNALAGADDA is currently pursuing the B.E. degree in electronics and communication engineering from the Birla Institute of Technology and Science, Pilani, Hyderabad Campus. He is passionate about computer architecture, FPGA-based system design, and deep learning. His research interests include approximate computing, low-power VLSI design, and hardware for machine learning.

UPPUGUNDURU ANIL KUMAR received the Ph.D. degree from the Birla Institute of Technology and Science, Pilani, India. He is currently an Assistant Professor with the Department of Electrical and Electronics Engineering, Faculty of Science and Technology (IcfaiTech), ICFAI Foundation for Higher Education, Hyderabad, Telangana, India. He has published more than 35 papers in international journals and conferences. His research interests include power-efficient VLSI design, approximate computing, and video coding. He received the Best Thesis Award from the IEEE CICT Conference, in 2022. He is a reviewer of several Springer/Elsevier journals.

RISHI THOTLI is currently pursuing the B.E. degree in electrical and electronics engineering with the Birla Institute of Technology and Science, Pilani, Hyderabad Campus, India. His research interests include computer architecture, energy-efficient circuit optimization, VLSI design, and approximate computing.

SATVIK SARDESAI is currently pursuing the B.E. degree in electronics and communication with the Birla Institute of Technology and Science, Pilani, Hyderabad Campus. He is passionate about microelectronics and RF circuits. His research interests include computer arithmetic and approximate computing.

SREEHARI VEERAMACHANENI (Senior Member, IEEE) received the Ph.D. degree from the International Institute of Information Technology Hyderabad. He is currently a Faculty Member with the Department of Electronics and Communication Engineering, Gokaraju Rangaraju Institute of Engineering and Technology (GRIET), Hyderabad, India. He has published more than 75 papers in international journals and conferences. His research interests include arithmetic circuits, approximate computing, hardware security, data converters, and in-memory computing. He is a Senior Member of ACM. He is a reviewer of several IEEE/Springer/Elsevier journals.

SYED ERSHAD AHMED (Member, IEEE) received the Ph.D. degree from the Birla Institute of Technology, India. He is currently an Assistant Professor with the Birla Institute of Technology and Science (BITS), Pilani, Hyderabad Campus, India. His research interests include arithmetic circuit design, low-power VLSI design, and approximate computing. He has published more than 40 international journals and conference articles.