iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** III **Month of publication:** March 2021

DOI: https://doi.org/10.22214/ijraset.2021.33372

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue III Mar 2021- Available at www.ijraset.com

Design of a 16-Bit Posit Multiplier with Power Efficiency

Monisha. R[1], Sivasakthi. V[2], Sivaprakasam. S[3], Poonguzhali. M[4]
[1, 2, 3, 4]*Narasu's Sarathy Institute of Technology, Salem. India.*

Abstract: Considering the many number of arithmetic functions or operations existing till now, we deem multiplication to be one of the most utilized function. They are implemented in various fields. Most notable applications are signal processing, graphics and scientific computation. Posit number system is widely used nowadays in replacement for IEEE number system. In this paper, we propose the architecture of a 16 –bit modified posit multiplier, which results in less power consumption. In the posit multiplier, our work relates to the mantissa multiplier. The multiplier used for mantissa is structured mainly for providing maximum bit-width that is possible, where the entire multiplier can be partitioned to produce smaller multipliers. Only the necessary partitioned multipliers will be used during the time of execution. Thus resulting in less power consumption.
Keywords: Posit multiplier, Mantissa multiplier, Less power consuming circuit, Modified booth encoding, 16-bit multiplier.

I. INTRODUCTION

Posits, a new data-type modeled to build as an alternative for IEEE floating point number. Posit number do not demand any operand to be of varying size (variables) because if they find any answer to be wrong, they do the rounding process. This behavior is very much unlike the universal numbers or also known as unum. The posit system yields many benefits, which may include their dynamic range being large, use of a simple hardware execution system, handling of exceptional cases are comparatively better, being greater in terms of accuracy. Posits do not take the values of infinity and zero. As enunciated before, posit processing system occupies less equipment compared to IEEE floats processing machine. Additionally, they have a irregular or inconsistent distribution of data that settles the need in certain applications, for example it may be useful in deep learning. The 8-bit or 16-bit posits are generally utilized in deep learning applications. 32-Bit posits arithmetic may be utilized scientifically computing fields.

The generalized pattern for the posit data-type or number system is shown in Fig. 1. Posit (*nb*, *es*) is the representation of posits where *nb* means total or absolute bit width and *es* means the bit width foe exponent. It constitutes 4 parts: 1.sign→**s** 2.regime→**rg** 3.exponent→**exp** 4.mantissa→**frac**. The component width of the bit is variable. The regime values are always varying. The rest of the positions for the bits shall be taken by mantissa ans also the exponent. This happens only when the regime does not reserve all the positions. A numeral illustrated in the format for posits arithmetic:

$$value = (-1)s \times useedrg \times 2exp \times (1 + frac) \quad (1)$$

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

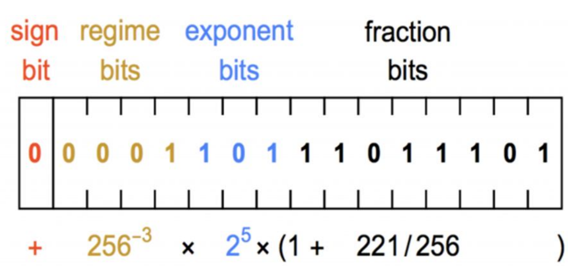

Fig. 1 Posit number system

We have dynamic ranging bit width here as shown in Fig. 1, exemption being the sign bit sized 1. The regime as well as the sign bit should always be present in the general structure. Exponent and mantissa sections occur if there any leftover positions for the bits. As such the fraction part which includes the implicit bit also has a bit-width ranging from 1 to *nb* – *es* value. Every time we do any operations, the value of mantissa need not be to its full extent always. Using a maximum bit multiplier all the times results in unwanted consumption of power.

We establish the successful implementation of a 16 – bit posit multiplier, where the mantissa (fraction) multiplier is split into several small ones and use them when required. Thus we efficiently design a low power consuming posit multiplier and reduce increasing power consumption. This architecture can also be used in various multipliers other than posit multipliers to improve power efficiency of the corresponding component or device. The upcoming sections are as follows: *Segment II* describes the existing multiplier method. *Segment III* briefs the proposed method. *Segment IV* presents the software generated result of our multiplier. Lastly *Segment V* concludes the paper.

II. EXISTING METHOD

Currently the working method of posit multiplier uses the modified booth's algorithm technique. This method is also known as bit-pair algorithm or radix-4 algorithm. There is a possibility to decrement the partial products number. Here we do not shift and add for all the columns of the multiplier and later doing multiplication with 0 or 1. But what we do here is to multiply every 2^{nd} column with 0 or ±1 or ±2. Both the methods yield similar results. Radix-4 booth encoder compares 3 bits at a time which is also known as the overlapping method. By adding a zero the the left end of the number, we start pairing them into a batch of 3 numbers together shown in Fig. 2.

Fig. 2 3-bit pairing for Booth recoding

Operating procedure for Radix-4 booth encoder is as shown in the Fig. 3. The results achieved from multiplying the different multiplier states with 0, ±1 and ±2 are described.

Multiplier Bits Block			Recoded 1-bit pair		2 bit booth	
i+1	i	i-1	i+1	i	Multiplier Value	Partial Product
0	0	0	0	0	0	Mx0
0	0	1	0	1	1	Mx1
0	1	0	1	-1	1	Mx1
0	1	0	1	0	2	Mx2
1	0	0	-1	0	-2	Mx-2
1	0	1	-1	1	-1	Mx-1
1	1	0	0	-1	-1	Mx-1
1	1	0	0	0	0	Mx0

Fig. 3 Radix-4 method booth recoding

Radix-4 Booth encoding procedure is as follows: (1) Make sure n is even, so if necessary create an extension using the sign bit. (2) Adjoin zero value to the LSB in our multiplier. (3) Depending upon each and every value, we form partial products as -y, +y, -2y, +2y or 0. Two's complement procedure is done to deal with negative values. After shifting the multiplier y bit one by one, multiplication process is thus proceeded. We obtain partial produced reduced by twice its size which is a main advantage. This reduction facilitates the decreased delay in propagation while the circuit is operating. The main disadvantage of the circuit mentionable must be the difficulty in the construction of the circuit hardware.

III. PROPOSED METHOD

The proposed posit multiplier; the design differs from the existing method in the mantissa multiplier. The mantissa multiplier uses the Modified booth multiplication, consisting of a (nb - es) bit mantissa multiplier. While doing multiplication, we do not always require a maximum bit-width mantissa multiplier. That is, there is no need to always use nb – es mantissa unit. Generally, the bits which are not used in multiplier and multiplicand will be assigned to zero normally. But those bits will be reverted to value one during recoding the multiplier when it is negative. Thus it results in an unwanted signal toggling. The unnecessary signal toggling must be avoided to reduce power consumption. Though we use the same radix-4 booth multiplication in the proposed system, the power efficiency can be achieved by splitting the multipliers into smaller ones and accessing or controlling it through a control signal. Such that this design also involves generating a control signal to enable the required smaller multiplier component when required during our run time. The design details of 16-bit multiplier is discussed in detail.

A. *Reduction of Multiplier*

Below given Fig. 4 shows the steps performed in the multiplier after the partial product generation. From this picture it is clear, that the number of bits used to generate an answer for 16-bit multiplication is very time, space and also memory consuming. Since we multiply all the 16 bit of the multiplier and multiplicand, an annoyingly long process takes place. Thus in the proposed system we have decided to divide or split the multiplier digits.

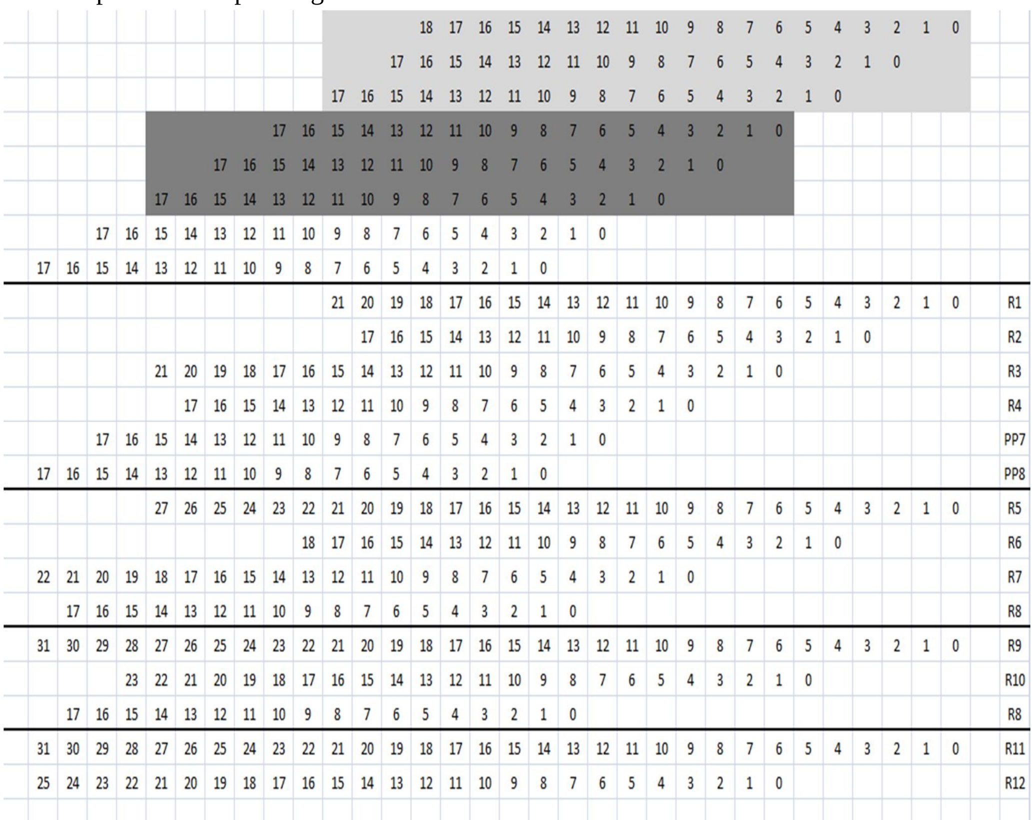

Fig. 4 Partial product addition after multiplying two 16-bit numbers in the existing method.

Here partial product is generated with the radix-4 booth sequence. Each PP has 16-bits. The multiplier can be classified into 4 batches: first 4-bits, first 8-bits, first 12-bits and the entire 16 bit. This type of classification or sub-dividing is done for both the multiplier and multiplicand input. As such, this process of dividing the inputs is done for the purpose of creating a control signal. The use of control signal will be discussed in the next section. Here in the proposed system, we have broken down the whole multiplier into16 small multiplier, since the design we have simulated is for a 16 digit multiplication. We name the smaller ones according to our opinion. During multiplication the no. of bits in the input may vary. For example, if we have found the position of 1 to be in the four digits for both the inputs then the control is given to the 1^{st} smaller multiplier namely PPG00. In that case the number of bits used during the procedure would be 4-bits for both the inputs.

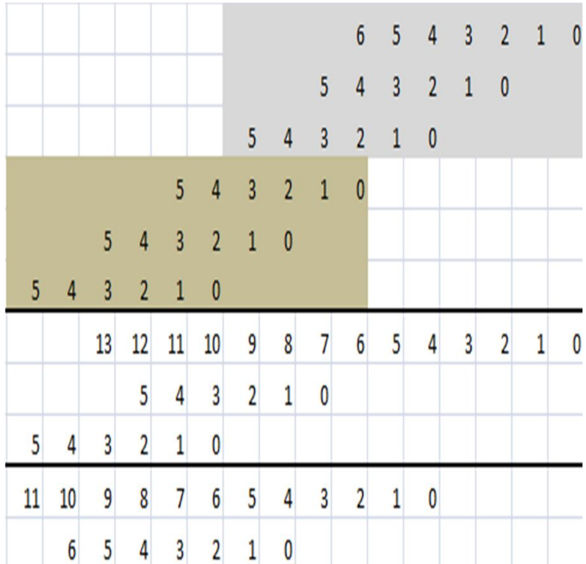

Fig. 5 Reduced partial products bit in proposed system

Fig 5 shows an example when both the multiplier and multiplicand are less than 7-bit.

B. Controlling Signal

For completing the design of our multiplier, we need a control signal. We divide the 16-bit digit input into four parts as already mentioned above. To generate Control Signal for both the inputs, one has to identify the position in where the binary digit ‘1’ makes its appearance. After discovering the position of 1, comparison of the divided subgroups and our input digit is performed. According to that grouping, we generate the control signal for both inputs. This in turn implies the selection of smaller multiplier which we need to use for the process. All in all we sum-up the use of control signal to select the smaller multiplier we have partitioned for the operation.

IV. SIMULATION RESULTS

The entire simulation and result obtaining using test vectors are done with the help of ModelSim software. We have used the ModelSim 6.3f version for its ease in finding errors.

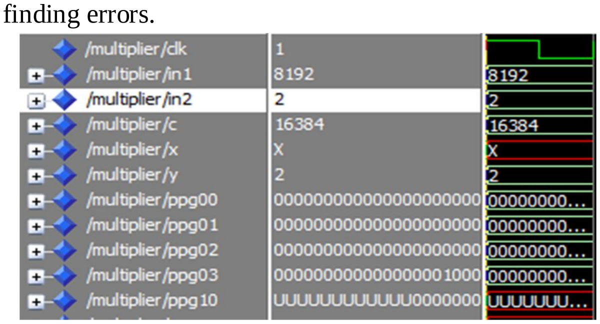

Fig. 6 Output sample 1 after simulation

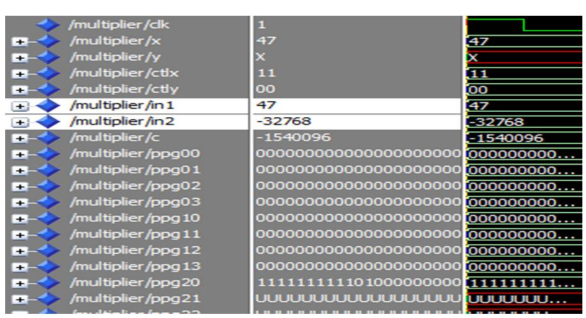

Fig. 7 Output sample 2 after simulation

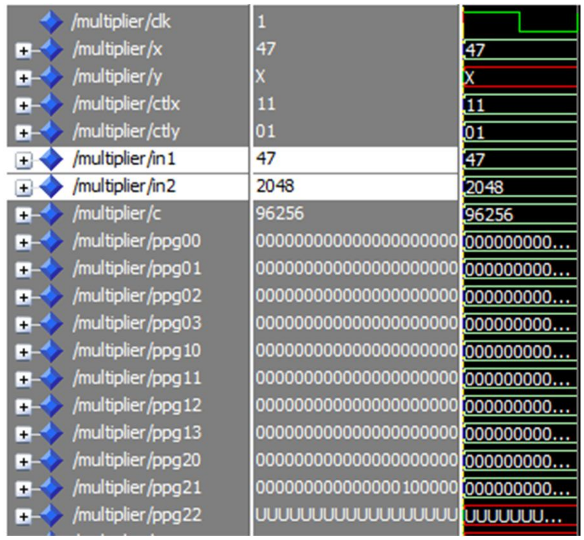

Fig. 8 Output sample 3 after simulation

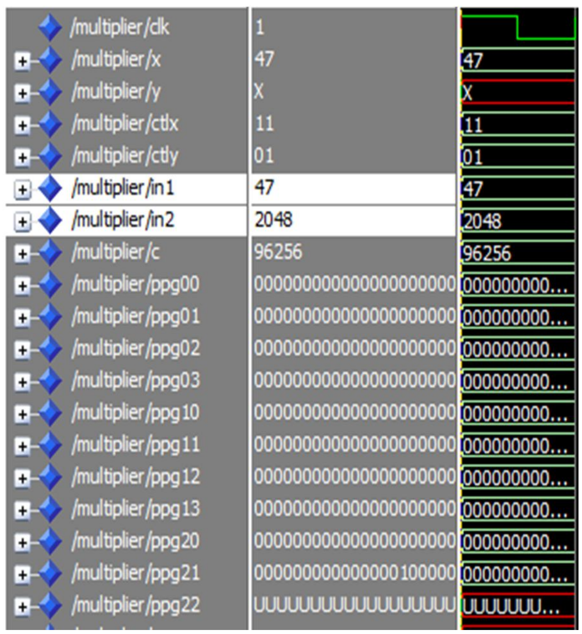

Fig. 9 Output sample 4 after simulation

In the above presented Fig.6 to Fig. 9, we have shown the simulation results after inputting test vectors. For the proposed design of Posit(16,1), we have achieved about a 20% cutback in power. The residue is left out in order to eliminate the toggling of the signal. We set the unwanted segments to 0, thus avoiding toggling in the last steps of rounding and adding. Such factors contribute to the lessened power consuming in our design structure.

V. CONCLUSION

The idea proposed in the paper is a 16-bit Posit multiplier architecture with power efficiency. Intrigued by the idea of reconstructing the multiplier unit for mantissa into smaller parts, because of the whole mantissa unit is not used entirely all the time, we have built the hardware. To limit the power consumption, we use only the necessary potion of the multiplier.The proposed method is evaluated to find that we can achieve about 16% reduction in power consumption. Our method is evaluated for 16-bit multiplier, whereas we can extend the work for 8-bit and 32-bit posit multipliers using the same technique. For futuristic purposes, more power reduction techniques for multiplier architecture can be developed. The work need not be necessarily limited to multipliers alone. Future works can be deployed also for Posits Adder or Posits Multiply Accumulate functions.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue III Mar 2021- Available at www.ijraset.com

REFERENCES

[1] Hao Zhang and Seok-Bum Ko , "Design of Power Efficient Posit Multiplier," IEEE on Circuits and Systems—ii: express briefs, vol. 67, no.5, May. 2020.
[2] J. L. Gustafson and I. Yonemoto, "Beating floating point at its own game: Posit arithmetic," Supercomput. Front. Innovat. Int. J., vol. 4, no. 2, pp. 71–86, Jun. 2017.
[3] Sneha Singh and Prachi Singh, "Implementation of Radix-4 Booth Multiplier by VHDL," IJRREEE, vol 4, issue 1, pp 1-11, Jan. 2017.
[4] Sukhmeet Kaur, Suman and Manpreet Singh Manna, "Implementation of Modified Booth Algorithm (radix-4) and its comparison with Booth Algorithm(radix-2)," Advances Electric and Electronic Engineering, ISSN 2231-1297, vol 3, no 6,pp 683-690, 2013.
[5] Bikash Chandra Sahoo, Sanjay Kumar Samant, "Design and Power Estimation of Booth Multiplier Using Different Adder Architectures,"2013.
[6] S.Shafiulla Basha, Syed, Jahangir Badashah, "Design and Implementation of Radix-4 Based High Speed Multiplier for ALU's using Minimal Partial Products," IJAET, ISSN 2231-1963, vol 4, issue 1, pp 314-325.