International Journal of LNCT
Volume-3 Issue-11 April -2019
ISSN: (Online):2456-9895

Review paper on Signed and Unsigned Multiplier by Using Radix-4 Process

Jagrati Nayak[1*], Prof. Prashant Purohit[2]
M. Tech. Scholar, Dept. of Electronics and Communication, LNCT, Bhopal[1]
Associate Professor, Dept. of Electronics and Communication, LNCT, Bhopal[2]

Abstract-

This paper presents the design and implementation of signed-unsigned Modified Booth Encoding (SUMBE) multiplier. The present Modified Booth Encoding (MBE) multiplier and the Baugh-Wooley multiplier perform multiplication operation on signed numbers only. The array multiplier and Braun array multipliers perform multiplication operation on unsigned numbers only. Thus, the requirement of the modern computer system is a dedicated and very high speed unique multiplier unit for signed and unsigned numbers. Therefore, this paper presents the design and implementation of SUMBE multiplier. The modified Booth Encoder circuit generates half the partial products in parallel. By extending sign bit of the operands and generating an additional partial product the SUMBE multiplier is obtained. The Carry Save Adder (CSA) tree and the final Carry Look-ahead (CLA) adder used to speed up the multiplier operation. Since signed and unsigned multiplication operation is performed by the same multiplier unit the required hardware and the chip area reduces and this in turn reduces power dissipation and cost of a system.

Keywords:-

***SUMBE, MBE, CSA, CLA,* Baugh-wooley multiplier**

1. INTRODUCTION

In advanced processing frameworks augmentation is a math activity. The increase activity comprises of delivering halfway items and after that including these incomplete items the last item is acquired. In this way the speed of the multiplier relies upon the quantity of incomplete item and the speed of the snake. Since the multipliers significantly affect the exhibition of the whole framework, some elite calculations and models have been proposed [1-2]. The rapid and committed multipliers are utilized in pipeline and vector PCs.

The fast Booth multipliers and pipelined Booth multipliers are utilized for advanced sign preparing (DSP) applications, for example, for sight and sound and correspondence frameworks. Rapid DSP calculation applications, for example, Fast Fourier transform (FFT) require augmentations and increases. The paper [3] presents a structure approach for fast Booth encoded parallel multiplier. For halfway item age, another Modified Booth encoding (MBE) plot is utilized to improve the presentation of customary MBE plans. In any case, this multiplier is just for marked number duplication task.

The regular changed Booth encoding (MBE) creates an unpredictable incomplete item exhibit in view of the additional fractional item bit at any rate huge piece position of every halfway item push.

Along these lines papers [4] presents a straightforward way to deal with create an ordinary incomplete item cluster with less halfway item pushes and immaterial overhead, in this manner bringing down the intricacy of fractional item decrease and lessening the territory, postponement, and intensity of MBE multipliers. Yet, the disadvantage of this multiplier is that it capacities just for marked number operands.

The changed Booth calculation is widely utilized for fast multiplier circuits. Once, when exhibit multipliers were utilized, the diminished number of created fractional items altogether improved multiplier execution. In structures dependent on decrease trees with logarithmic rationale profundity, be that as it may, the diminished number of halfway items limitedly affects generally speaking execution. The Baugh-Wooley calculation [5] is an alternate plan for marked increase, however isn't so broadly received in light of the fact that it might be confounded to send on sporadic decrease trees. Again the Baugh-Wooley calculation is for just marked number increase. The exhibit multipliers [6] and Braun cluster multipliers [7] works just on the unsigned numbers. In this manner, the necessity of the cutting edge PC framework is a committed and extremely rapid multiplier unit that can perform increase task on marked just as unsigned numbers. In this paper we structured and executed a devoted multiplier unit that can perform augmentation task on both marked and unsigned numbers, and this multiplier is called as SUMBE multiplier.

2. LITERATURE REVIEW

Increase could be one in everything about essential activity in cutting edge sign methodology and DSP framework. Speed execution of increment influents general execution of cutting edge sign procedure yet as in data processor frameworks [5, 7]. A couple of better figurings and models are needed than redesign and animate growth activities. Inside orchestrated Algorithm, factors of composing multiplier component are time of fragmented thing and summation of fragmentary thing. To improve speed

execution of increment, amounts of midway thing are decreased by abuse of radix-32 corner rule and for diminishing deferral of summation of inadequate thing Wallace tree structure has been used. This paper presents pipelined marked 64x64 piece multiplier segment. Choice of abuse of radix-32 as opposed to radix-16 is that it delivers less blended pack of midway thing and Wallace tree structure has been used as opposed to bunch structure as delayed consequence of in Wallace tree, blend of levels required are less considered of show structure.

Wallace tree is appreciated for his or her optimal preparing time, once adding different operands to 2 yields abuse of 3:2 or 4:2 blowers or each. Wallace tree affirmations total base general deferral figure1 exhibits 9 operands Wallace structure, any place 3:2 mechanical blower gadget pack information having 3 multi-bit sources of info and 2 multi-bit yields. 4:2 mechanical gadgets pack information having four multi-bit sources of info and 2 multi-bit yields. Abuse of each mechanical blower gadget, no. of levels has been decreased that conjointly causes overhauling speed of multiplier part. For 64x64 operands abuse of radix-32, total seven blowers part and four dimensions has been used in Wallace tree that is effectively less diverged from radix-16, inside which fourteen blowers and six dimensions are used in Wallace tree structure thus absolute execution of duplication has been reasonably upgraded inferable from less total entryway defer in pipelined marked 64x64 multiplier segment misuse of radix-32 stall Algorithm and Wallace tree structure.

In second pack, beginning piece is most fundamental piece of starting gathering and alternative bits are next 5 bit of multiplier component sum. In third bundle, starting piece is most basic piece of second gathering and choice bits are next 5 bit of multiplier component sum. This framework is continued. For each pack, incomplete thing is delivered abuse of number sum. For n bit multiplier variable there's n/5 or [n/5 + 1] gatherings and fragmentary thing in organized altered stall rule radix-32. Table I for organized radix-32 changed corner principle has been sketched out and radix-32 register exchange rationale level read of altered stall encoder conjointly has been arranged that is demonstrated in figure five. Accordingly it diminishes proportion of deficient thing stood out from radix-16, upgrades system intensity of multiplier segment, decline check delay. Computation of forefront multiplier component and re-encoding of multiplier variable is dead in parallel. Issue Fi is learned misuse of correlation and figure four. Figure three show diagram of 64x64-piece organized multiplier part.

3. DIFFERENT TYPES OF MULTIPLIER

These are the conventional multipliers having regular structures. Add and shift algorithm is used for its operation and hence its circuit is based on this algorithm. By direct mapping of the manual multiplication into hardware, an Array multiplier circuit can be developed [8]. An array of adder circuits can be used to accumulate partial products. The partial products are generated by multiplying the multiplicand with each bit of multiplier. The bit order decides the amount of shift of partial products. At the final stage, the partial products are added. The number of generated partial products is equal to that of multiplier bits. If multiplier length is equal to N, then N-1 numbers of adders are required to implement array multiplier [9].

Vedic Multiplier

Vedic Mathematics is an ancient system of mathematics existed in India. In this eminent approach, methods of basic arithmetic are simple, powerful and logical. Another advantage is its regularity. These advantages make Vedic Mathematics an important topic for research. Vedic Mathematics rules are mainly based on sixteen Sutras. Out of these sixteen Sutra's Urdhva Triyakbhyam sutras and Nikhilam sutras are used for multiplication. Vedic multipliers are considered to be the best compared with conventional multipliers and Urdhva Triyakbhyam Sutra based multiplication is more efficient compared to that of Nikhilam Sutra [10]. Implementation of Vedic mathematics on FPGA is easy due to its regularity and simplicity [4].

All partial products required for multiplication are calculated much before actual multiplication begins. This is the big advantage of this multiplication. Based on the Vedic Mathematics algorithm, these partial products are added to obtain final product which leads to a very high speed approach. Multiplier designs based on Vedic mathematics are with high speed and consume relatively low power. Multipliers are the basic and key blocks of a Digital Signal processor. Multiplication is the key process in improving the computational speed of Digital Signal Processors [6]. Convolution, Fast Fourier transforms and various other transforms make use of multiplier blocks [11]. Among various methods of multiplications in Vedic mathematics, Urdhva Tiryagbhyam is efficient. Urdhva Tiryagbhyam is a general multiplication formula applicable to all cases of multiplication.

Booth Series of Multipliers

There is no need to take the sign of the number into deliberation in dealing with unsigned multiplication. However in signed multiplication the process will be changed because the signed number is in a 2's compliment pattern which would give a wrong result if multiplied by using similar process for unsigned multiplication [6]. Booth's algorithm is used for this. Booth's algorithm preserves the sign of the result. Booth multiplication allows for smaller, faster

multiplication circuits through encoding the signed numbers to 2's complement, which is also a standard technique used in chip design, [6] and provides significant improvements by reducing the number of partial product to half over "long multiplication" techniques. Radix 2 is the conventional booth multiplier.

Radix 2

In booth multiplication, partial product generation is done based on recoding scheme e.g. radix 2 encoding. Bits of multiplicand (Y) are grouped from left to right and corresponding operation on multiplier (X) is done in order to generate the partial product [19]. In radix-2 booth multiplication partial product generation is done based on encoding which is as given by Table1. Parallel Recoding scheme used in radix-2 booth multiplier is shown in the Table 1.

Table 1: Booth recoding for radix 2

Q_n	Q_{n+1}	Recoded Booth	Operation
0	0	0	Shift
0	1	+1	Add x
1	0	-1	Subtract x
1	1	0	Shift

Radix 4

One of the solutions attaining high speed multipliers is to improve parallelism. It helps in decreasing the number of consecutive calculation stages [1]. The Original version of Booth's multiplier (Radix – 2) had two drawbacks [7]. The number of Add or Subtract operations became variable and hence became difficult while designing Parallel multipliers. The Algorithm becomes disorganized when there are isolated 1s. These problems are overthrown by using Radix 4 Booth's algorithm which can browse strings of three bits with the algorithm. The above recoding has the nice feature that they translate into the partial products shown in table 2.

Table 2: Booth recoding for radix 4

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

4. PROPOSED METHODOLOGY

According to this algorithm the multiplier term is encoded. For the coding, three successive bits of the multiplier operand are considered, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier; the first bit is the $(n-1)$th bit of the multiplier. The encoding process is shown as in the figure 1.

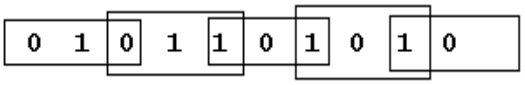

Figure 1: Grouping of bits from the multiplier term, for using Radix-4 Booth recoding

In modulo 2^n+1 multiplication, the result and one input use weighted representations, while the other input uses diminished-1 representation.

Let $d[A]=(a_n a_{n-}\dots\dots a_0)$ be the diminished-1 representation of the weighted binary number A , $B=(b_n b_{n-}\dots\dots b_0)$ and $P=|A\times B|_{2^n+1}$ all be weighted binary nmbers. The algorithm uses $d[A]$ and B to compute the value of P, that is

$$P=(A\times B)mod(2^n+1)=|A\times B|_{2^n+1}=F(d[A],B) \quad (1)$$

Where F() is a function that maps the inputs $d[A]$ and B to the output P.

The modulo 2^n+1 multipliers can be used for the situation where one operand being constant and another operand being variable. In that situation, the diminished-1 representation of the constant operand and can be pre-computed and be present in the hardware, so its conversion circuits are not required. The algorithm uses the radix-4 Booth recoding [11] where successive overlapping triplets of B are examined and encoded. The outputs of the encoder uses the $d[A]$ to form the partial products. The partial product reduction scheme uses the well-known inverted end-around-carry (IEAC) adder tree [12]. The final adder generates the product. To avoid (n+1)-bit circuits, we distinguish the following two cases:

When $\overline{a_n+b_n}=1$; and when $\overline{a_n+b_n}=0$

Table 3: Encoding for $-b_{n-1}+b_0-2b_1$ and $-b_{n-1}+b_0+2b_1$

b_1	b_0	b_{n-1}	Encoding	Formulas
0	0	0	0	$-b_{n-1}+b_0-2b_1$
0	1	0	1	
1	0	0	-2	
1	1	0	-1	
0	0	1	-1	$-b_{n-1}+b_0+2b_1$
0	1	1	0	
1	0	1	1	
1	1	1	1	

Booth Encoder and Booth Selector

Due to the encoding scheme accordant with the radix-4 Booth recoding, the partial product generator (PPG) can be constructed with the well-known Booth encoder (BE) and Booth selector (BS). Many implementations for BE block and BS block

were published, but they can be reduced to two categories: one having 4-bit bus and the other having 3-bit bus. Here, the 3-bit bus approach is used.
The BE block examines successive overlapping triplets of $b_{2i-1}+b_{2i}-2b_{2i+1}$ and encodes each to $0,\ \pm1,\ and\ \pm2$. Each BE block produces 3 bits: 1x, 2x and Sign. The 3 bits along with the multiplicand $d[A]$, are used to form partial products. The partial products are produced by the BS blocks. Each BS block takes as inputs two successive bits of $d[A]$. There are two types of BS blocks in the multipliers $BS^{+}\ and\ BS^{-}$ since the inverted multi-bit left-circular shifts of $d[A]$ and $d[-A]$ are different. For the i-th partial products, $0 \le i \le K$, there are $2i\ \ BS^{-}$ blocks and $(n-2i)\ \ BS^{+}$ blocks are required.

5. CONCLUSION

In all multiplication operation product is obtained by adding partial products. Thus the final speed of the multiplier circuit depends on the speed of the adder circuit and the number of partial products generated. If radix 4 Booth encoding technique is used then there are only 3 partial products and for that only one CSA and a CLA is required to produce the final product. Due to the encoding scheme accordant with the radix-4 Booth recoding, the partial product generator (PPG) can be constructed with the well-known Booth encoder (BE) and Booth selector (BS). In step second, due to one operand A is used as diminished-1 number representation $d[A]$, one correction is generated. In step third, the partial products and the correction term which are n-bits each are added through a inverted end around carry (IEAC) adder tree.

REFERENCES

[1] D. Kalaiyarasi and M. Saraswathi, “Design of an Efficient High Speed Radix-4 Booth Multiplier for both Signed and Unsigned Numbers”, 4th International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), IEEE 2018.

[2] Prof. S. B. Patil, Miss. Pritam H. Langade, “Design of Improved Systolic Array Multiplier and Its Implementation on FPGA”, International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015

[3] Elisardo Antelo, Paolo Montuschi and Alberto Nannarelli, “Improved 64-bit Radix-16 Booth Multiplier Based on Partial Product Array Height Reduction”, IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 64, No. 2, February 2017.

[4] Kavita and Jasbir Kaur, “Design and Implementation of an Efficient Modified Booth Multiplier using VHDL”, Special Issue: Proceedings of 2nd International Conference on Emerging Trends in Engineering and Management, ICETEM 2013.

[5] Shiann-Rong Kuang, Jiun-Ping Wang and Cang-Yuan Guo, “Modified Booth Multipliers With a Regular Partial Product Array”, IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 56, No. 5, May 2009.

[6] S. Vassiliadis, E. Schwarz, and D. Hanrahan, “A general proof for overlapped multiple-bit scanning multiplications,” IEEE Trans. Comput., vol. 38, no. 2, pp. 172–183, Feb. 1989.

[7] D. Dobberpuhl et al., “A 200-MHz 64-b dual-issue CMOS microprocessor,” IEEE J. Solid-State Circuits, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.

[8] E. M. Schwarz, R. M. A. III, and L. J. Sigal, “A radix-8 CMOS S/390 multiplier,” in Proc. 13th IEEE Symp. Comput. Arithmetic (ARITH), Jul. 1997, pp. 2–9.

[9] J.Clouser etal.,“A600-MHz superscalar floating-point processor,” IEEE J. Solid-State Circuits, vol. 34, no. 7, pp. 1026–1029, Jul. 1999.

[10] S. Oberman, “Floating point division and square root algorithms and implementation in the AMD-K7 microprocessor,” in Proc. 14th IEEE Symp. Comput. Arithmetic (ARITH), Apr. 1999, pp. 106–115.

[11] R. Senthinathan et al., “A 650-MHz, IA-32 microprocessor with enhanced data streaming for graphics and video,” IEEE J. Solid-State Circuits, vol. 34, no. 11, pp. 1454–1465, Nov. 1999.

[12] R. Riedlinger etal.,“A32nm,3.1billion transistor, 12wide issue itanium processor for mission-critical servers,” IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 177–193, Jan. 2012.