International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 03 | Mar 2020 www.irjet.net p-ISSN: 2395-0072

DESIGN OF A LOW POWER SERIAL- PARALLEL MULTIPLIER WITH LOW TRANSITION ADDITION

Muralidharan.V[1], Reigny Keziah.M[2], Sathish Kumar. N[3]

[1]*Assistant Professor, Dept. of ECE, Christ the king Engineering College, Coimbatore, TamilNadu, India*
[2]*UG Scholar, Dept. of ECE, Christ the king Engineering College, Coimbatore, TamilNadu, India*
[3]*Professor, Dept. of ECE, Sri Ramakrishna Engineering College, Coimbatore, TamilNadu, India*

Abstract - — *This paper proposes a low power and low transition addition for the serial parallel multiplier for digital signal processing (DSP) and machine learning (ML) applications, dictating the area, delay, and overall performance of parallel implementations. In the proposed work a radix-4, serial-parallel multiplier using modified booths algorithm for accelerating applications such as digital filters, artificial neural networks, and other machine learning algorithms is designed. The Multiplication process is done by the serial–parallel (SP) modified radix-4 Booth multiplier that adds only the nonzero Booth encodings and skips over the zero operations, making the latency dependent on the multiplier value. Our optimizations can result in an improvement over the standard serial parallel Booth multiplier in terms of area–delay and power.*

Key words— *Modified Booth multiplication, serial parallel multiplier*, field programmable gate array (FPGA*), digital signal processing (DSP), machine learning (ML)*

I. INTRODUCTION

Multiplication is conceivable the most important signal processing and machine learning application. They can be also able to find the area, delay and overall performance of the serial parallel implementation [5]. The work of the multiplication circuits has been extensive However the modified booths algorithm at higher radixes I combination with Wallace tree has generally been accepted as the high performance implementation for the general problems. In the digital circuits, multiplication can be performed in many ways. They are as follows 1) parallel-parallel, 2) serial-serial, 3) serial-parallel,4) parallel-serial. Using the booths modified algorithm we can be able to study about the serial parallel two speed multiplier (TSM) that conditionally adds the non zeros part of multiplications and skip over the zeros sections.

In the digital signal processing and the machine learning, reduces the description are often used to improve the performance of the design conation for the smallest possible bit width to obtain a required computational accuracy. The condition is actually fixed at designed time and hence changes in the requirements that the future scope involves the redesigning of the implementation. In case of the smaller bit width would be enough, the design runs of the lower efficiency because the unnecessary computation takes place. To reduce this mixed precision algorithm attempt to use the lower bit width some portion of the time and large bit width when necessary. These are normally developed with the data path operating with the different ways

This paper gives the idea about the dynamic control structure for the removed part of the computational completely during run time. This is done by the modified serial parallel multiplier [11]. The method of this multiplier will skip all the zeros and all one's computation. The multiplier takes all the bits both operand in parallel and design to primitive blocks which is easily get into the digital signal processing, central processing units and global positioning units. For some input sets inputs the multipliers achieves some contemplate improvements in the computational performance. The multiplier is tested using the field programmable gate array (FPGA) technology [9]. This also account for the four different process –voltage-temperature corners. The main definition of this paper are as follows:

1) The modified booth multiplier has data path which is divided into two sub circle each operating with a different critical path.
2) Explaining how the multiplier takes advantage of the particular bit pattern to perform less work. This result in reduced the discontinuations increase the throughput and the area and time performance than the normal multipliers.
3) This method estimate shows the working of the multiplier and evaluation of the utility of the proposed multiplier through an FPGA

2. MULTIPLICATION AND MULTIPLIERS

Multipliers play an important role in regular digital signal processing and many other applications [14]. In advancement of technologies many researchers have tried to design multipliers which offer either of the following design targets of high speed, low power consumption and

hence less area or even combination of them in one multiplier thus making them adaptable for various high speed, low power and compact very large scale integrated circuits implementation.

The common multiplication is "add and shift" algorithm. In parallel multipliers the partial products to be added in the main parameter because they can be able to determine the performance of the parallel multiplier. The main thing is to reduce the number of partial product. To reduce it here comes the solution of Modified Booth algorithm is one of the most popular algorithms. The speed can be achieved by the Wallace Tree algorithm which can be used to reduce the number of subsequent adding stages. In the advancement we are using Modified Booth algorithm technique advantage of the booths algorithms in a multiplier. If the parallelism increase, the number of shifts between the partial products and intermediate sums to be added will become large and increase they can have reduced speed, increase in silicon area due to asymmetrical of structure and also increased power observed due to increase in interlink resulting from composite routing. On the other hand, "serial-parallel" multipliers agreement speed to achieve better performance for area and power consumption [3]. The selection of a parallel and serial multiplier actually depends on the essence of application. In this we introduce the multiplication algorithms, architecture and compare them in terms of speed, area, power and combination of these metrics. The multiplication algorithm for an N bit multiplicand y by N bit multiplier x is shown below:

$Y = Y_{n-1}\ Y_{n-2}$ .....$Y_2\ Y_1\ Y_0$ Multiplicand (y)

$X = X_{n-1}\ X_{n-2}$ .....$X_2\ X_1\ X_0$ Multiplier (x)

1010 (multiplier)
x 1011 (multiplicand)

1010	(partial product)
1010	
0000	
1010	
1101110	

Fig-1: 4-bit multiplication

In this give Fig-1 the 1010 is given as 'y' which is multiplicand and the 1011 is given as 'x' which is multiplier [10]. all the given values such as 1010,1010,0000,1010 is said to be as the partial product. And the final answer which is obtained by adding all the partial product.

2.1 Multiplication algorithm

If the Least Significant Bit of Multiplier is '1', then add the multiplicand into a compiler. Shift the multiplier one bit towards right and multiplicand one bit towards left [10]. Stop when all bits of the multiplier are zero. From above it is clear that the multiplication has been changed to numbers of additions. If the Partial Products are added serially then a serial adder is used with minimum hardware [7]. It is possible to add all the partial products with one combinational circuit using a parallel multiplier [10]. However, it is possible, to use contraction technique then the number of partial products can be reduced before the performance of addition.

2.2 Serial Parallel Multiplier

The general architecture of the serial/parallel multiplier is shown I the below Fig-2. The one operand is fed to the circuit in parallel while the other is serial. N number of partial products are formed each cycle. On successive cycles, each cycle does the addition is done in column one then the multiplication table of M*N Partial products [3]. The final results are stored in the output register after N+M cycles. While the area required is N-1 for M=N. In a serial parallel multiplier, the multiplicand x, arrives bit serially while the multiplier 'a' is applied in a bit parallel format. A common approach used in such multipliers is to generate a row or diagonal of bit products in each time slot and perform the addition simultaneously [6]. If the data is positive when the condition 'x' is greater than zero.

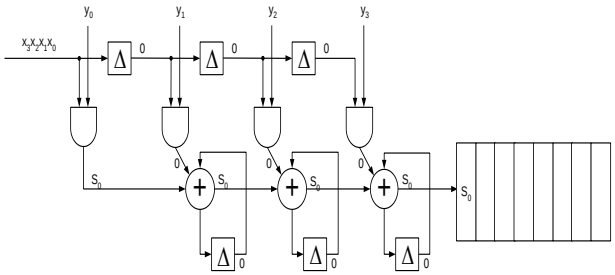

Fig-2: serial parallel multiplier

3. Modified booth multiplication

The modified booth algorithm is used to perform the high speed multiplication. The modified booth multiplier computational time and the logarithm of word length of operations are proportional to each other [4]. We can reduce the half the number of partial product. Radix 4-booth multiplication is used to increase the speed of the multipliers and reduce the area of the multiplier circuits.

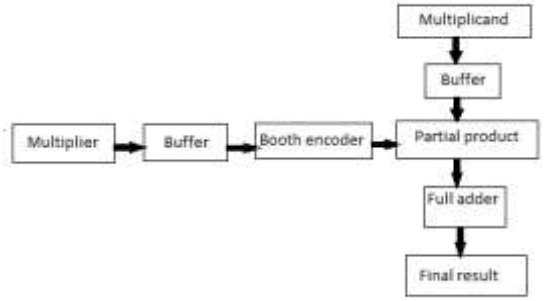

Fig-3: Block Diagram of the Modified Booths Algorithm

In every second column is taken and it is multiplied by 0 or+1 or+2 or -1 or -2 instead of multiplying with 1 or 0 after shifting and adding of every column of the booth multiplier. Thus half of the partial product can be reduced using this booth algorithm [7][6]. Based on the multiplier bits, the process of encoding the multiplicand is performed by radix-4-bit booth encoder

3.1 Booths encoder

The modified booths multiplier digits can be defined by following equation:

$$Zj = q2j + q2j-1 - 2q2j \text{ with } q-1 = 0 \quad (1)$$

The product of any digits of Z with multiplicand Y may be -2y, -y, 0, y, 2y. but by performing the left shift operation at the partial product generation stages, 2y may be generated. By taking 1`s complement to this 2y negativity is done, and the one is added in appropriate 4-2 compressor.

Table-1: Booth Recoding Table for Radix-4

BLOCK	PARTIAL PRODUCT
000	0
001	+1*multiplicand
010	+1*multiplicand
011	+2*multiplicand
100	-2*multiplicand
101	-1*multiplicand
110	-1*multiplicand
111	0

Here –2*multiplicand is actually the 2s complement of the multiplicand with an equivalent left shift of one-bit position [9]. Also, +2 *multiplicand is the multiplicand shifted left one-bit position which is equivalent to multiplying by 2.

To enter $\pm$ 2*multiplicand into the adder, an (n+1)-bit adder is required. In this case, the multiplicand is offset one bit to the left to enter into the adder while for the low-order multiplicand position a 0 is added [3]. Each time the partial product is shifted two bit positions to the right and the sign is extended to the left.

During each add-shift cycle, different versions of the multiplicand are added to the new partial product depends on the equation derived from the bit-pair recoding table above.

4. Radix-4 Booth Multiplication

The radix 4 booth recoding is simply a multiplexor that selects the correct shift and add operation based on the grouping of bits found in the product register [4]. The multiplicand and the 2's complement of the multiplicand and added based on the recording value.

$$N = \left\lfloor \frac{n+2}{2} \right\rfloor. \quad (2)$$

An equation describing the computation is given by

$$p = (Y_1 + Y_0)x + \sum_{i=1}^{N} 2^{2i-1}(Y_{2i+1} + Y_{2i} - 2Y_{2i-1})x. \quad (3)$$

Following the notation in Section II, Y denotes the Length-N digit vector of the multiplier y [1]. The radix-4 Booth algorithm considers three digits of the multiplier Y at a time to create an encoding e given by

$$e_i = Y_{2i+1} + Y_{2i} - 2Y_{2i-1} \quad (4)$$

where *i* denotes the *i* th digit. Apart from *Yi*+2*Yi*+1*Yi* = 000 and *Yi*+2*Yi*+1*Yi* = 111 which results in a 0, the multiplicand is scaled by either 1, 2, –2, or –1depending on the encoding.

This encoding *ei* is used to calculate a partial product Partial product by calculating

$$Partial\,Product_i = e_i x = (Y_{2i+1} + Y_{2i} - 2Y_{2i-1})x. \quad (5)$$

This Partial Product is aligned using a left shift (2^{2i-1}) and the summation is performed to calculate the final result *p[3]*. Since the *Y*–1 digit is nonexistent, the 0th partial product.

Partial Product 0 = *(Y*1 *+Y0) x.* A serial (sequential) version of the multiplication is performed by computing each partial product in *N* cycles.

$$p[0] = 2^{n-2}(Y_1 + Y_0)x$$
$$p[j+1] = 2^{-2}(p[j] + 2^n(Y_{2j+1} + Y_{2j} - 2Y_{2j-1})x), \quad j = 1, \ldots, N-1$$
$$p = p[N].$$

Thus the equation verifies the two-speed optimization of the multiplier. The two optimization can allow to do the better hardware utilization. The partial product to n most significant bits of the product p is (P[2*B-1]+= partial product) [2].

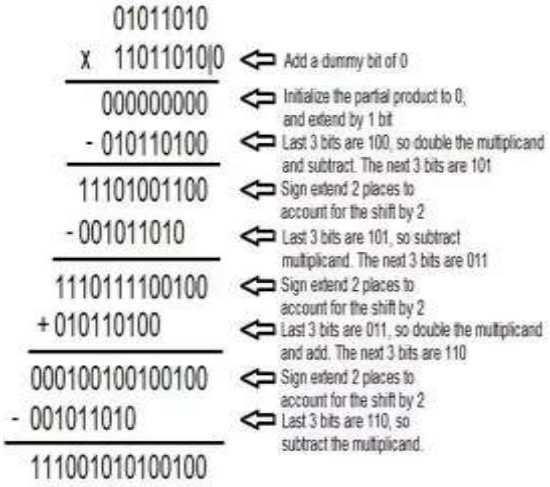

Fig-4: example for booths serial parallel multiplier

4.1. Block diagram of Radix 4 serial parallel Multiplier using booths multiplication

This block diagram represents radix 4 serial parallel multiplier using booths multiplication algorithm. The multiplicand and multiplier is given and both the input which is given both in serial and in parallel ways. The register in which the initial is set as 0. The shift and add control logic is to perform the all the condition of the modified booths multiplier [11]. The carry which is saved in the block c. This block diagram performs the radix 4 serial parallel multiplier using the modified booths multiplier.

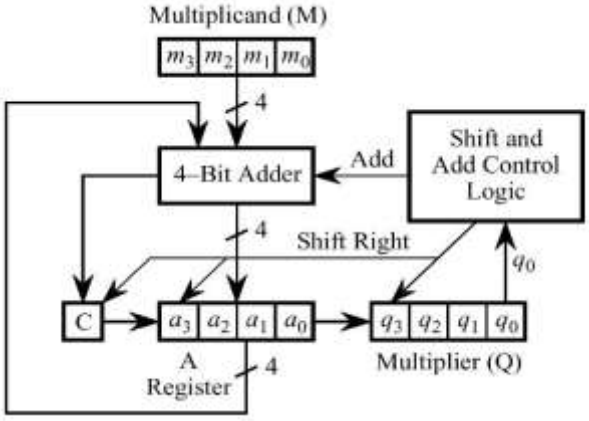

Fig-5: Block diagram of radix 4 serial parallel multiplier

5. Comparison Results

The comparison of the radix-2 and radix-4 shown that the difference between the characteristics of the multiplication speed, no of finding the radix 4 booths multiplier is better than radix-2 multiplier. By implementing both the multipliers radix 4 is more speed than radix 2.

Table-2: Comparision table of radix2 and radix 4 multipliers

DEVICE UTILIZATION SUMMARY	RADIX 2	RADIX 4
NUBMBER OF SLICES	397	71
NUMBER OF 4 INPUT LUT'S	184	100
NUMBER OF BOUNDED INPUTS	16	16
NUMBER OF BOUNDED OUTPUTS	16	16
MACRO STATISTICS		
LATCHES	24	12
8 BIT LATCH	24	12
XOR	71	23
1 BIT XOR 2	64	21
8 BIT XOR 2	7	2
TIMING SUMMARY		
MINIMUM PERIOD	5.45ns	4.750ns
MINIMUM INPUT ARRI-VAL TIME BEFORE	7.93ns	4.01ns
MINIMUM OUTPUT RE-QUIRED TIME AFTER	6.126ns	6.205ns

The simulated design for the radix4 serial parallel multiplier using booth multipliers same as radix 2 but the difference is that the synthesis report. Table-2, shows the comparison of the radix 4 and radix 2 multipliers and the

area, power and delay can be determined and the efficiency of the radix 4 multiplier. The following conditions which are compared is given in the comparison table.

5.1. Power Calculation

Parameters	Existing Method	Proposed Method
Gate Count	3279	2864
Power(mW)	197.28	177.42
Delay(ns)	8.664	7.626

Table-3: Area, power and delay of radix 4.

This comparison table shows about the area, power consumed and delay of the radix 2 and radix 4 serial parallel multiplier. The area which is given as the gate counting. In radix 2 the total number of gate count 3279 is used. Whereas in radix 4 only 2864 is consumed. In the case of power consumed in the radix 2 is 197.28 mW. Whereas in radix 4 the power consumed is given by 177.42 mW. The delay can be calculated for radix 2 is given by 8.664 ns. Whereas for the radix 4 the delay is given by 7.626 ns.

6. Conclusion

In this paper by using the radix 4 serial parallel multiplier by the help of modified booths multiplier we are able to get the required parameters such as area, power and delay. When comparing radix 2 to radix 4, the radix 4 is more efficient in terms of area, power and delay.

REFERENCES

[1] S. J. Schmidt and D. Boland, "Dynamic bit width assignment for efficient dot products," in Proc. Int. Conf. Field Program. Log. Appl., Sep. 2017, pp. 1–8.

[2] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang,"Hardware for machine learning: Challenges and opportunities," CoRR, vol. abs/1612.07625, Dec. 2016. [Online]

[3] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture, Oct. 2016, pp. 1-12.

[4] B. Dinesh, V. Venkateshwaran, P. Kavinmalar, and M. Kathirvelu, "Comparison of regular and tree based multiplier architectures with modified booth encoding for 4 bits on layout level using 45 nm technology, "in Proc. Int. Conf. Green Comput. Commun. Elect. Eng., Mar.2014, pp. 1–6.

[5] B. Rashidi, S. M. Sayedi, and R. R. Farashahi, "Design of a low-power and low-cost Booth-shift/add multiplexer-based multiplier,"in Proc. Iranian Conf. Elect. Eng. (ICEE), May 2014, pp. 14–19.

[6] B. Rashidi, "High performance and low-power finite impulse response filter based on ring topology with modified retiming serial multiplier on FPGA," IET Signal Process., vol. 7, no. 8, pp. 743–753, Oct. 2013.

[7] G. C. T. Chow, A. H. T. Tse, Q. Jin, W. Luk, P. H. Leong, and Thomas, "A mixed precision Monte Carlo methodology Reconfigurable accelerator systems,"in Proc ACM/SIGDA Int. Symp. Field Program. Gate Arrays, 2012, pp. 57–66.

[8] P. Devi, G. P. Singh, and B. Singh, "Low power optimized array multiplier with reduced area," in High Performance Architecture and Grid Computing, A. Mantri, S. Nandi, G. Kumar, and S. Kumar, Eds.Berlin, Germany: Springer, 2011, pp. 224–232.

[9] I. Kuon and J. Rose, "Area and delay trade-offs in the circuit and architecture design of FPGAs," in Proc.ACM/SIGDA 16th Int. Symp.Field Program. Gate Arrays, 2008, pp. 149–158.

[10] K. S. Trivedi and M. D. Ercegovac, "On-line algorithms for division and multiplication," IEEE Trans. Comput., vol. C-26, no. 7,pp. 681–687, Jul. 1977.

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

[12] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs.New York, NY, USA: Oxford Univ. Press, 2000.

[13] M. D. Ercegovac and T. Lang, Digital Arithmetic (Morgan Kaufmann Series in Computer Architecture and Design). San Mateo, CA, USA:Morgan Kaufmann, 2004.

[14] V.Muralidharan and N.Sathish kumar, Design and Implementation of low power and high speed Multiplier using Quaternary carry look-ahead adder, Microprocessor and Microsystems (Elsevier). https://doi.org/10.1016/j.micpro.2020.103054