International Journal of Engineering Research and General Science Volume 3, Issue 4, July-August, 2015
ISSN 2091-2730

VHDL Implementation of Advanced Booth Dadda Multiplier

Sumod Abraham[1], Sukhmeet Kaur[2], Sanjana Malhotra[3]

[1]Student, Manav Rachna College of Engineering, India, sumod11abraham@gmail.com

[2]Asst. Prof, Manav Rachna College of Engineering, India, sukhmeet@mru.edu.in

[3]Associate Prof, Manav Rachna College of Engineering, India, hodece@mru.edu.in

ABSTRACT- Every processor processes on the basis of arithmetic operations. Out of the basic arithmetic operations, it is multiplication which is complex. In other words, time consuming. So to speed up the processor we need a fast multiplier. The speed of the multiplier depends upon the generated partial products. The number of adders that are used in the process of multiplication makes it complex as well. The main focus of this work is partial product reduction using radix 4 Booth algorithm and also reduce the number of adders by using Dadda multiplier. Since two methods are clubbed together for different operations, this multiplier is called Advanced Booth Dadda multiplier.

Keywords: Adders, Booth multiplier, Dadda multiplier, Partial Products, Radix 4 multiplication.

1. Introduction

[1] Arithmetic circuits for example adders and multipliers are time consuming depending upon the bits and hardware complexity. [2] In signal processing, major operation of various filters like FIR, IIR filter is convolution, which can be implemented by using delay element and arithmetic operators. So multiplier must be fast. One such algorithm is Booth algorithm. [3] In 1950, A.D. Booth invented the Booth Algorithm during the study on crystallography. [4] By using this method, multiplication gets faster as the count of partial products (P.P.) gets reduced.

[5] Mac-Sorley proposed modified Booth algorithm, where three bits instead of two bits are scanned. So the number of partial products generated are less than that in Booth algorithm. [6] Luigi Dadda designed a hardware multiplier in 1965 which is known as Dadda multiplier. [7] Here the number of full adders used is (N2-4N+3) and half adders used is (N-1) where N is the number of bits of the multiplier. So by combining Advanced booth algorithm and Dadda multiplier, number of partial products as well as number of adders can be reduced.

2. Booth Multiplier

[8] Iteration steps are reduced by using Booth multiplier when compared with other conventional methods.

2.1. Radix 2 Multiplication

In this method, the multiplier is appended by '0' in the LSB and then it is grouped in overlapping groups of two. Then based on the operations mentioned in the recording table drawn below, multiplicand is altered to finally get the product. Here, we get n partial products for n bit multiplier. Now that the count of P.P. is equal to the number of multiplier bits, hence process is slow.

TABLE 1. Recording table of radix 2 multiplication

M (i+1)	M (i)	[3] OPERATIONS
0	0	No arithmetic operation is performed only shifting is done.
0	1	Add multiplicand to left half part of product and then shifting is done.
1	0	Subtract multiplicand from left half part of product and then shifting is done
1	1	No arithmetic operation is performed only shifting is done.

2.2. Radix-4 Multiplication

[9] It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. In this method, the multiplier is appended by '0' in the LSB and then it is grouped in overlapping groups of three. Then based on the operations mentioned in the recording table drawn below, multiplicand is altered to finally get the product. Here we get (n/2) partial products for n bit multiplier where n is even. When n is odd, we get ((n/2)+1) partial products.

So as we increase the number of overlapping bits in the multiplier (radix), partial products can be decreased. In this paper, partial products are formed using radix 4 multiplication process. When compared [10] Radix-4 Booth Multiplier gives higher speed when compared with Radix-2 Booth Multiplier. [11]The main disadvantage of the modified booth multiplier is its complexity to produce partial product.

TABLE 2. Recording table of radix 4 multiplication

M (i+1)	M (i)	M (i-1)	OPERATION
0	0	0	0
0	0	1	1 X Multiplicand
0	1	0	1 X Multiplicand
0	1	1	2 X Multiplicand
1	0	0	-2 X Multiplicand
1	0	1	-1 X Multiplicand
1	1	0	-1 X Multiplicand
1	1	1	0

3. Dadda Multiplier

It is a column compression multiplier. [13]Column compression multiplier continued to be studied due to their high speed performance.

In this method, partial products are arranged in a definite pattern shown below by shifting all the columns upwards. Then using (2,2) and (3,2) counters the height of the pattern is reduced to two. Then using carry select adder these two rows are added to get the final

product. [14] In order to realize the minimum number of reduction stages, the height of each intermediate matrix is limited to the least integer that is no more than 1.5 times the height of its successor.

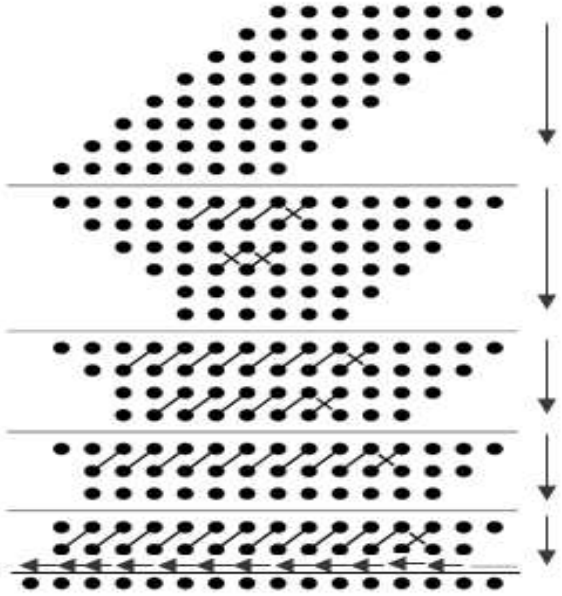


Fig. 1. [15] Pattern of Dadda Multiplier

4. Proposed Multiplier

In this paper, Booth Dadda multiplier is described where Radix 4 Booth Multiplier is used for partial product formation and Dadda Multiplier is used to add those partial products. When we simply use these two techniques, result may come incorrect. So we use sign template along with partial products formed after using Radix 4 multiplication and then arrange them into Dadda pattern.

4.1 Sign Template

In this, the first partial product that is formed is appended with S, S and S $\overline{\text{in}}$ the MSB where S is the sign bit and S is the complement of sign bit. R$\overline{\text{e}}$st in all partial products except in last and last but one partial product, 1 and S is appended at MSB. In last but one partial product, S is appended in the MSB and in the last partial product, nothing is appended, instead MSB is ignored.

4.2 Example

Consider two numbers where multiplicand is 15_{10} i.e. $(00001111)_2$ and multiplier is 30_{10} i.e. $(00011110)_2$.

Stage 1: Radix 4 multiplication:

Firstly multiplier is subdivided into overlapping groups of three after appending a zero (shown in bold) at LSB which is shown below:

0 0 0 1 1 1 1 0 **0**

Then we find all possible partial products that can be formed by multiplicand.

1 * multiplicand = 0 0 0 0 1 1 1 1 (multiplicand as it is)

-1 * multiplicand = 1 1 1 1 0 0 0 1 (2"s complement of (1*multiplicand))

2 * multiplicand = 0 0 0 1 1 1 1 0 (ignore MSB and append a zero to the LSB of (1* multiplicand))

-2 * multiplicand = 1 1 1 0 0 0 1 0 (2"s complement of (2*multiplicand))

0 = 0 0 0 0 0 0 0 0

Then based on the sign template we determine the operations to be performed. We start forming partial products from the LSB. So we get four partial products.

First partial product will be 1 1 1 0 0 0 1 0 as pair (100) has the operation (-2*multiplicand).

Second partial product will be 0 0 0 0 0 0 0 0 as pair (111) has the operation '0'.

Third partial product will be 0 0 0 1 1 1 1 0 as pair (011) has the operation (2*multiplicand).

Fourth partial product will be 0 0 0 0 0 0 0 0 as pair (000) has the operation '0'.

Stage 2: Sign template:

In this stage, sign bits are appended towards the MSB of each partial product. For pairs (000), (001), (010) and (011) S is '0' and S is '1' and for pairs ($\overline{100}$), (101), (110) and (111) S is '1' and S is '0'.

In this example, the first partial product becomes **0 1 1** 1 1 1 0 0 0 1 0. Appended sign bits are shown in bold. The second partial product becomes **1 1** 0 0 0 0 0 0 0 0. The third partial product becomes **1** 0 0 0 1 1 1 1 0. The fourth partial product becomes 0 0 0 0 0 0 0 0. After every partial product, two bits are left vacant from LSB.

The proper structure of the partial product format is shown below:

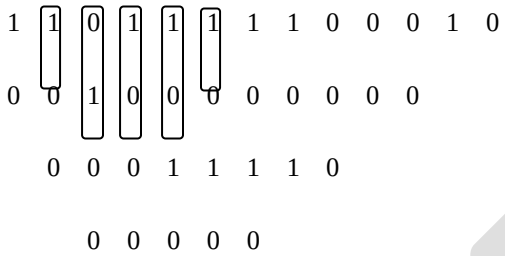
				0	**1**	**1**	1	1	1	0	0	0	1	0
			1	**1**	0	0	0	0	0	0	0	0		
	1	0	0	0	1	1	1	1	0					
	0	0	0	0	0	0	0							

Stage 3: Dadda Multiplier

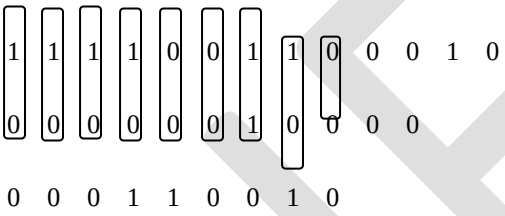
Now all the bits are shifted upwards column wise like the way shown below:

So the structure becomes like the way shown below:

Now reducing the number of rows to 3 by using (2,2) or (3,2) adders as shown below:



Sum of the encircled bits are placed first in that column followed by left over bits and carry of the previous encircled columns. The result is shown below:



Again counters are used to reduce the column height to two. The result is shown below:

1 1 1 0 1 0 0 0 0 0 0 1 0

0 0 1 0 0 1 1 0 0 0 0

Now add the two rows using carry select adder to get the final product as shown below:

$(0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0)_2$ which is equivalent to $(450)_{10}$.

5. Comparison

Proposed multiplier is compared with Radix-4 Booth multiplier and Radix-2 Booth multiplier and the result is shown below:

TABLE 3. Performance of proposed multiplier

PARAMETERS	PROPOSED MULTIPLIER	RADIX-4 BOOTH MULTIPLIER [12]	RADIX-2 BOOTH MULTIPLIER [12]
No. of slices	78	119	166
No. of LUTs	78	213	300
Path Delay	8.003 ns	29.198 ns	37.881 ns

6. Result

Result after multiplying two eight bit numbers in VHDL is shown below:

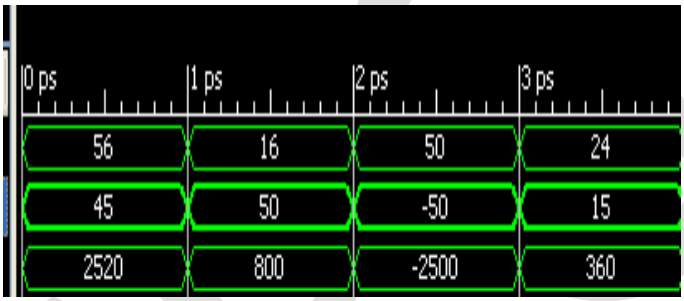


Fig. 2. Implementation of Advanced Booth Dadda in VHDL

7. Conclusion

It can be concluded that the Advance Booth Dadda multiplier uses less number of slice LUTs. Also it is a faster multiplier when compared to radix 4 Booth multiplier and radix 2 Booth multiplier. [16] It is known that, systems based on microprocessor have fast-growing demand of embedded microprocessors with high speed and low power features. Since, multiplier is an important unit of microprocessor, if this is made to work faster, entire embedded system will work in high speed. [9] A multiplier is an essential element of the digital signal processing such as filtering, convolution, and inner products. [17] Low-power multiplier design holds a significant part in low- power VLSI system design. So, some out of the many applications prove the importance of proposed multiplier. By using higher radix multiplication method (in the future), number of partial products gets further reduced thereby making the multiplier much faster than the proposed multiplier..

REFERENCES:

[1] Rakesh Kumar Saxena, Neelam Sharma and A. K Wadhwani "Design of Fast Pipelined Multiplier using Modified Redundant Adder," I.J. Intelligent Systems and Applications, 2012, 4, 47-53.

[2] Chen, O.T.-C., Wei-Lung Liu, Hsun-Chang Hsieh and Jeng-Yih Wang, "A highly-scaleable FIR using the radix-4 Booth algorithm," Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on , vol.3, no., pp.1765,1768 vol.3, 12-15 May 1998.

[3] Deepali Chandel , Gagan Kumawat , Pranay Lahoty, Vidhi Vart Chandrodaya and Shailendra Sharma, "Booth Multiplier: Ease of multiplication," International Journal of Emerging Technology and Advanced Engineering, (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 3, March 2013).

[4] Mohammad Tohidi, Yasin Amani, Mostafa Amirpour and Saeid Karamzadeh, "A New High Speed, Area Efficient 7-2 Compressor for Fast Arithmetic Operations," Internatıonal Journal of Natural and Engineering Sciences 7 (2): 72-77, 2013 ISSN: 1307-1149, E-ISSN: 2146-0086.

[5] K. Ramesh and M. Vaidehi, "Design Of High Speed Hardware Efficient 4-Bit SFQ Multiplier," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 2, February 2013.

[6] Addanki Purna Ramesh, "Implementation of Dadda and Array Multiplier Architectures Using Tanner Tool," International Journal of Computer Science & Engineering Technology (IJCSET) ISSN : 2229- 3345.

[7] Anju S and M Saravanan, "High Performance Dadda Multiplier Implementation Using High Speed Carry Select Adder," International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 3, March 2013 ISSN (Print) : 2319-5940.

[8] Sandeep Shrivastava, Jaikaran Singh and Mukesh Tiwari, "Implementation of Radix-2 Booth Multiplier and Comparison with Radix-4 Encoder Booth Multiplier," International Journal on Emerging Technologies 2(1): 14-16(2011) ISSN : 0975-8364.

[9] S. Jagadeesh and S.Venkata Chary, "Design of Parallel Multiplier– Accumulator Based on Radix-4 Modified Booth Algorithm with SPST," International Journal Of Engineering Research And Applications (IJERA) ISSN: 2248-9622.

[10] Sukhmeet Kaur, Suman and Manpreet Signh Manna, "Implementation of Modified Booth Algorithm (Radix 4) and its Comparison with Booth Algorithm (Radix-2)," Advance in Electronic and Electric Engineering. ISSN 2231-1297, Volume 3, Number 6 (2013), pp. 683-690.

[11] M. Jeevitha, R.Muthaiah and P.Swaminathan, "Review Article:Efficient Multiplier Architecture in VLSI Design," Journal of Theoretical and Applied Information Technology 30th April 2012. Vol. 38 No.2, ISSN:1992-8645.

[12] K. Babulu and G.Parasuram, "FPGA Realization of Radix-4 Booth Multiplication Algorithm for High Speed Arithmetic Logics," International Journal of Computer Science and Information Technologies, Vol. 2 (5) , 2011, 2102-2107 (ISSN:0975-9646).

[13] Vasudev G. and Rajendra Hegadi, "Design and Development of 8 -Bits Fast Multiplier for Low Power Applications," IACSIT International Journal of Engineering and Technology, Vol. 4, No. 6, December 2012.

[14] Sumod Abraham, Sukhmeet Kaur and Shivani Singh, "Study of Various High Speed Multipliers," 2015 International Conference on Computer Communication and Informatics (ICCCI -2015), Jan. 08 – 10, 2015, Coimbatore, INDIA.

[15] B. Jeevan, S. Narender, C.V.K. Reddy and K. Sivani, "A high speed binary floating point multiplier using Dadda algorithm," Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on , vol., no., pp.455,460, 22-23 March 2013.

[16] Na Tang, Jian-hui Jiang, and Lin, K., "A high-performance 32-bit parallel multiplier using modified Booth's algorithm and sign-deduction algorithm," ASIC, 2003. Proceedings. 5th International Conference on , vol.2, no., pp.1281,1284 Vol.2, 21-24 Oct. 2003.

[17] Sakshi Rajput , Priya Sharma , Gitanjali and Garima, "High Speed and Reduced Power – Radix-2 Booth Multiplier," IJCEM International Journal of Computational Engineering & Management, Vol. 16 Issue 2, March 2013 ISSN (Online): 2230-7893