International Conference on Information Engineering, Management and Security 2016 [ICIEMS]

ISBN	978-81-929866-4-7	**VOL**	01
Website	iciems.in	**eMail**	iciems@asdf.res.in
Received	02 – February – 2016	**Accepted**	15 - February – 2016
Article ID	ICIEMS014	**eAID**	ICIEMS.2016.014

Implementation of MAC using Modified Booth Algorithm

R P Meenaakshi Sundhari[1], M Karthickumar[2], S Pavithra[3] and E Madura[4]
[1, 2, 4] Assistant Professor and [3]PG Scholar
[1, 3, 4] Angel College of Engineering and Technology, Tirupur, India.
[2] Erode Sengunthar Engineering College, Erode, India.

Abstract- *The proposed system is an efficient processing of 16-bit Multiplier Accumulator using Radix-8 and Radix-16 modified Booth Algorithm and other adders (SPST adder, Carry select adder, Parallel Prefix adder) using VHDL (Very High Speed Integrated Circuit Hardware Description Language). This proposed system provides low power, high speed and fewer delays. In both booth multipliers, comparison between the power consumption (mw) and estimated delay (ns) are calculated. The application of digital signal processing like fast fourier transform, finite impulse response and convolution needs high speed and low power MAC (Multiplier and Accumulator) units to construct an added. By reducing the glitches (from 1 to 0 transition) and spikes (from 0 to 1 transition), the speed of operation is improved and dynamic power is reduced. The adder designed with SPST avoids the unwanted glitches and spikes, reduce the switching power dissipation and the dynamic power. The speed can be improved by reducing the number of partial products to half, by grouping of bits in the multiplier term. The proposed Radix-8 and Radix-16 Modified Booth Algorithm MAC with SPST reduces the delay and obtain low power consumption as compared to array MAC.*

Keywords: *Radix-8 modified booth algorithm Radix- 16 modified booth algorithm, Digital signal processing, VHDL (Very High Speed Integrated Circuit Hardware Description Language), Spurious Power Suppression Technique (SPST).*

I INTRODUCTION

Multiplication is a fundamental operation in digital signal processing application which consumes more power and area. Consequently, there is a need for designing low power Booth Algorithm. Booth algorithm is a standard technique which provides significant improvement in terms of chip area and power compared to other multiplication techniques. The implementation of the multiplier depends on the type of adder which is used in the MAC unit. By combining the multiplication with the accumulation the development of a hybrid type of adders like Parallel prefix adder and Carry save adder, the performance has improved. Several commercial processors have selected the Radix-8 multiplier architecture to increase the speed of operation, thereby reducing the number of partial products in the multiplication terms. The Radix-8 encoding reduces the digit number length in a signed digit representation as compared to Radix-2 multiplication. Its performance is bottleneck by the generation of the term 3X (Multiplicand), also referred to as hard multiple. The proposed MAC unit accumulates intermediate result in the terms of sum and carries bits instead of the output of the final adder, which optimize the pipeline system to improve the overall performance. The modified Booth's algorithm based on the Radix-8, generally called Booth-2, is the most popular approach for implementing the fast multipliers using parallel encoding. In general, multi-operand addition is the part of many complex arithmetic algorithms, such as multiplication and certain DSP algorithms. One of the most popular multi-operand adders is the carry-save adder which is capable of adding more than two operands at a time.

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. "Implementation of MAC using Modified Booth Algorithm". *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.

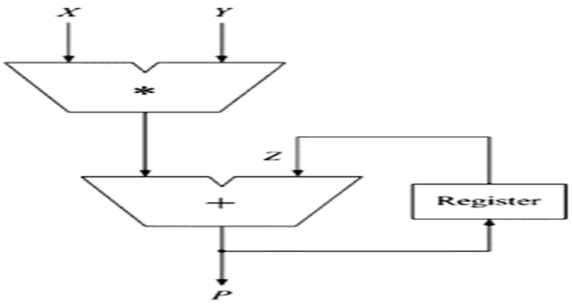


Figure 1. Hardware Architecture of General MAC Array Multiplier

The objective of this paper is to introduce the flexibility of adding three-input operands to a regular adder, thereby reducing the need of a special adder to the same process. General architecture of MAC is shown in Figure 1. The proposed approach is implemented using VHDL design with ModelSim 6.5c software. This executes the multiplication operation by multiplying the multiplier and the multiplicand. Multiplier is considered as X and multiplicand is Y which is added to the previous multiplication result Z as an accumulation step.

II Types of Adders

A. SPST Adder

In SPST Adder, the 16-bit adder /subtractor are divided into MSP (Most Significant Part) and LSP (Least Significant Part) between the 8th and 9th bits.

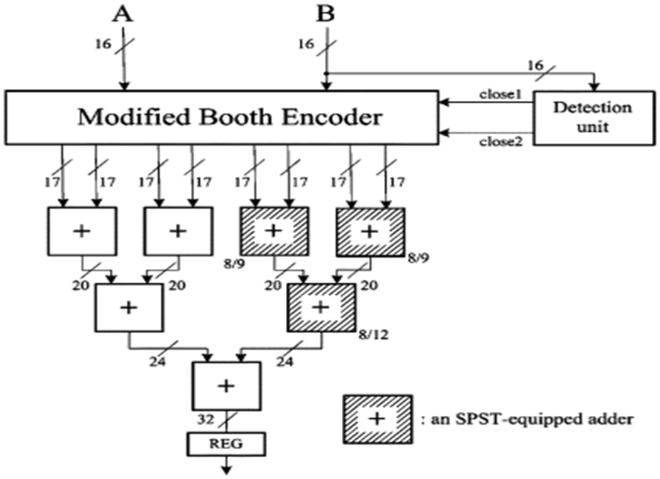


Figure 2. Proposed Low Power SPST Equipped Multiplier

The MSP of the original adder is modified to include the detection logical circuits, data controlling circuits, sign extension circuits, latch and clock circuits and logic for calculating carry-in and carry-out signals. Figure 2 shows the Proposed Low Power SPST Equipped Multiplier which consist of Latch, Detection logic and Sign extension logic.

B. Carry Select Adder

The 16-bit carry-select adder with a uniform block size of 4 can be created with three of these blocks and a 4- bit ripple carry adder is used. Since carry-in is known at the beginning of computation, a carry select block is not needed for first four bits. The delay of this adder will be four full adder delays, plus three MUX delays. The 16-bit carry-select adder with variable size can be similarly created shown in Figure 3. Here an adder with block sizes of 2-2-3-4-5 is used. This break-up is ideal when the full-adder delay is equal to the MUX delay, which is unlikely. The total delay is of two full adder delays and four multiplexer delays.

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. "Implementation of MAC using Modified Booth Algorithm". *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.

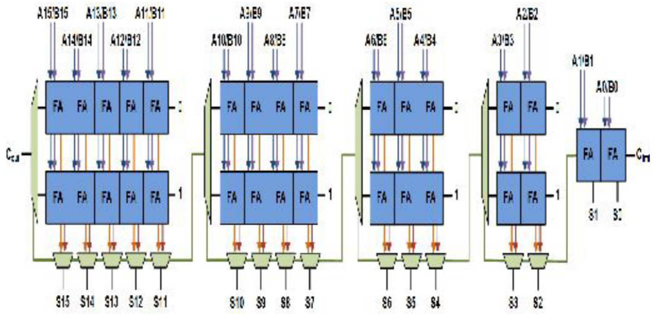


Figure 3. Variable Sized CSLA

C. Parallel Binary Adder

The goal of this paper is to present the architectures that provide the flexibility within a regular adder to augment/decrement the sum of two numbers by a constant which is considered in the addition process. This flexibility adds to the functionality of a regular adder, which achieves a comparable performance to conventional designs, therefore eliminating the need of having a dedicated adder unit to perform the same tasks. In this adder if the third operand is a constant, a design to accomplish three-input addition is required. These designs are called Enhanced Flagged Binary Adders (EFBA), shown in Figure 4. It also examines the performance of the adder when the operand size is expanded from 16 bits to 32 and 64 bits. Detailed analysis has been provided to compare the performance of the new designs with carry-save adders in terms of delay, power dissipated and area consumes.

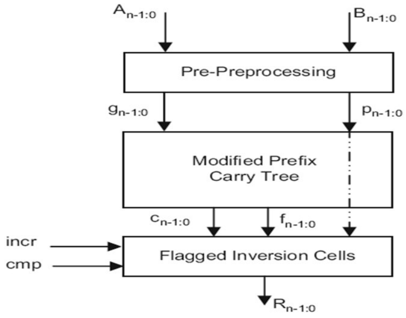


Figure 4. EFBA Block diagram

III Implementation

The Booth multiplication is a technique that allows faster multiplication by grouping the multiplier bits. The grouping of multiplier bits and Radix-8 Booth encoding reduce the number of partial products to half. The shifting and adding is for every column of the multiplier term and multiplying by 1 or 0 is commonly used. Here every second column is taken and multiplied by ±1, ±2, or 0. The advantage of this method is halving of the number of partial products. In Booth encoding the multiplier bits is formed in blocks of three, such that each block overlaps the previous block by only one bit. Grouping is started from the LSB side, and the first block only uses two bits of the multiplier term. Figure 5 below shows the grouping of bits from the multiplier term.

To obtain the correct partial product, each block is decoded from the grouped terms. Table 1 shows the encoding of the multiplier value Y, which uses the Modified Booth Algorithm and generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand X.

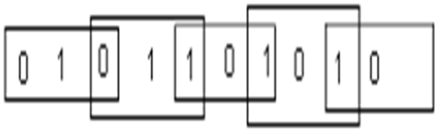
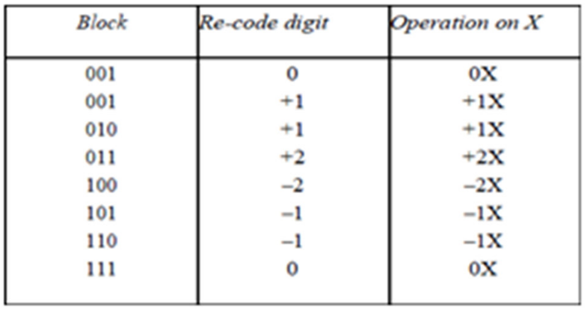


Figure 5. Grouping of bits from the multiplier term in the multiplication operation

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. "Implementation of MAC using Modified Booth Algorithm". *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.

Table I Encoded five signed digits



Block	*Re-code digit*	*Operation on X*
001	0	0X
001	+1	+1X
010	+1	+1X
011	+2	+2X
100	–2	–2X
101	–1	–1X
110	–1	–1X
111	0	0X

IV Modified Booth Algorithm for Radix-16

The numbers of subsequent calculation stages are decreased by enhancing the parallelism operation. So one of the solutions of realizing the high speed multipliers is to enhance parallelism operation. The Radix-4 Booth multiplier is the modified version of the conventional version of the Booth algorithm (Radix-2). The generation of Radix 2 and Radix 8 multiplication generally requires some kind of carry propagate adder, which increases the latency mainly due to the long wires that are required for propagating carries from the less significant to more significant bits.

High-speed modulo multipliers using the Booth encoding for partial product generation have been proposed in the Booth encoding technique which reduces the number of partial products to be generated and accumulated. In Radix-4 Booth encoding all modulo-reduced partial products can be generated by shifting and negation. The greater savings in area and dynamic power dissipation are feasible for large word-length multipliers by increasing the radix beyond four.

In Radix-8 Booth encoding method as shown in the Figure 6, the number of partial products is reduced by two- thirds. However, this reduction in the number of partial products leads to increased complexity in their generation. Compared with many other arithmetic operations multiplication is the time consuming and power hungry. Thus enhancing the performance of the circuit and reduction the power dissipation are the most important design challenges for all applications in which multiplier unit dominate the system performance and power dissipation.

The effective way to increase the speed of the multiplier is to reduce the number of the partial products. The number of partial products can be reduced with the higher radix Booth encoder, but the numbers of hard multiples are costly to generate and increases simultaneously. To increase the speed and performance, many parallel MAC architectures are proposed.

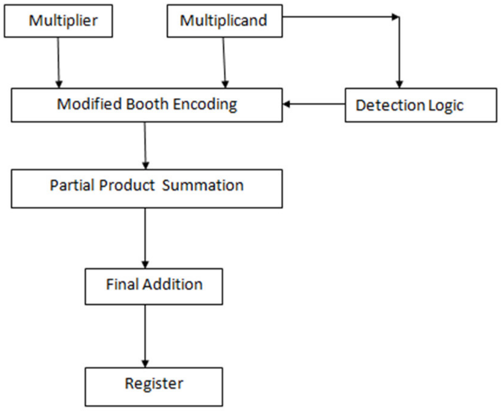


Figure 6. Block Diagram of Radix-8 MBA

There are two different common approaches that make use of parallelism to enhance the multiplication performance. The difference between the two is the latest one carries out accumulation by feeding back the final CSA (Carry Save Adder) output rather than the final adder results that are obtained. The entire process of parallel MAC is based on radix-8 booth encodings. Further the implementation result and the characteristics of parallel MAC based on both of the booth encodings is exposed.

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. “Implementation of MAC using Modified Booth Algorithm”. *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.

V Results

The simulation results for 16-bit Radix-2 and Radix-8 modified Booth algorithm with three different adders and MAC are shown below. Table II and III shows the synthesis report for array MAC, Radix-2 and Radix-8modified Booth algorithm with adders used in MAC. The code is dumped onto the target device Spartan 3E (Xc3s500eft2564), inputs (Set frequency of asynchronous nets as10MHz), signals (Set frequency for asynchronous nets as10MHz) and outputs (Set capacitive load of outputs as 28000 pf).

Table II shows the comparisons of power consumption and delay estimated of the Radix-2 Modified Booth Algorithm with three different adders in MAC. Table III shows the Radix-8 using that same adders used in the Radix-2 MAC. The design summary and simulation result also shown below.

Table II Comparison of radix-2 MBA

Device parameters	**Array Multiplier & accumulator**	**SPST adder**	**Parallel prefix adder**	**Parallel Binary adder**
Number of 4 input LUTs	636 out of 29504	1093 out of 29504	1083 out of 29504	549 out of 9312
Number of gate count for design	4209	5987	7167	3768
Estimated delay(ns)	217.8	39.69	24.93	53.084
Power consumption (mw)	154	144	138.80	16.533

Table III Comparison of Radix-8 MBA

Device parameters	**Array Multiplier & accumulator**	**SPST adder**	**Parallel prefix adder**	**Parallel Binary adder**
Number of 4 input LUTs	636 out of 29504	1093 out of 29504	1083 out of 29504	1222 out of 9312
Number of gate count for design	4209	5987	7167	7155
Estimated delay(ns)	217.8	39.69	24.936	66.10
Power consumption (mw)	154	144	138.80	19.93

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. “Implementation of MAC using Modified Booth Algorithm”. *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.

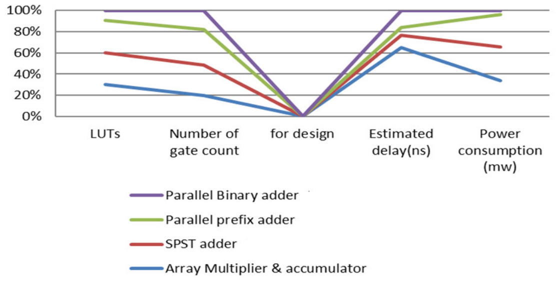
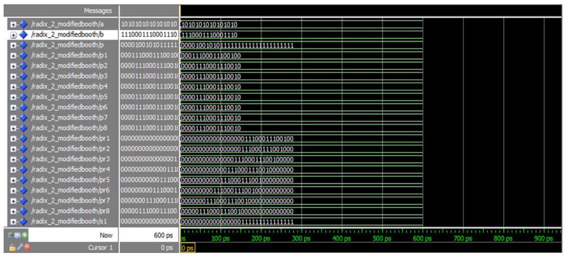
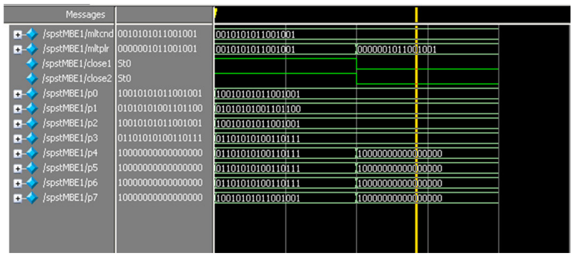


Figure 7. Graphical comparison of different parameter of the adders



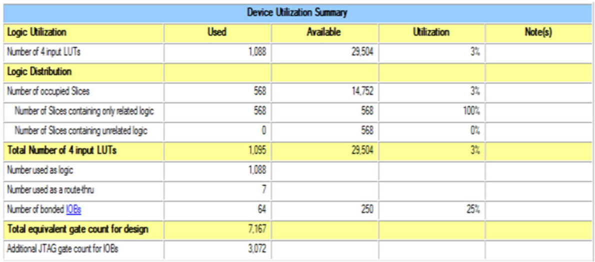
```
Messages
/radix_2_modifiedbooth/a     1010101010101010   1010101010101010
/radix_2_modifiedbooth/b     1110001110001110   1110001110001110
/radix_2_modifiedbooth/p     0000100101011111   00001001010111111111111111111111
/radix_2_modifiedbooth/p1    0001110001110010   0001110001110010 0
/radix_2_modifiedbooth/p2    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/p3    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/p4    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/p5    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/p6    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/p7    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/p8    000011100011100 1  00001110001110010
/radix_2_modifiedbooth/pr1   0000000000000000   00000000000000000111000111001 00
/radix_2_modifiedbooth/pr2   0000000000000000   00000000000000001110001110010 00
/radix_2_modifiedbooth/pr3   000000000000001 1  00000000000000011100011100100000
/radix_2_modifiedbooth/pr4   000000000000111 0  00000000000001110001110010000000
/radix_2_modifiedbooth/pr5   0000000000011100   00000000000111000111001000000000
/radix_2_modifiedbooth/pr6   000000000111000 1  00000000011100011100100000000000
/radix_2_modifiedbooth/pr7   0000000111000111   00000001110001110010000000000000
/radix_2_modifiedbooth/pr8   0000011100011100   00000111000111001000000000000000
/radix_2_modifiedbooth/s1    0000000000000000   00000000000000001111111111111111
Now                          600 ps             ps 100 ps 200 ps 300 ps 400 ps 500 ps 600 ps 700 ps 800 ps 900 ps
Cursor 1                     0 ps               0 ps
```

Figure 8. Simulation results for a 16-bit multiplier using radix-2 modified Booth algorithm with Parallel Prefix adder



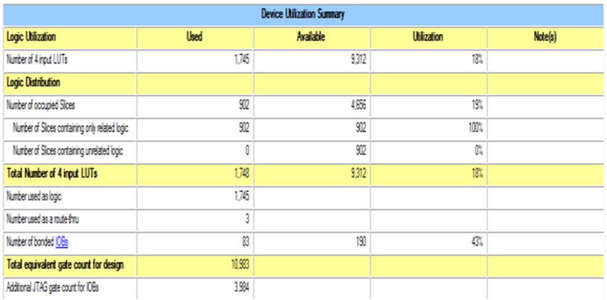
```
Messages
/spstMBE1/mltcnd  0010101011001001   0010101011001001
/spstMBE1/mltplr  0000001011001001   0010101011001001   0000001011001001
/spstMBE1/close1  St0
/spstMBE1/close2  St0
/spstMBE1/p0      10010101011001001  10010101011001001
/spstMBE1/p1      01010101001101100  01010101001101100
/spstMBE1/p2      10010101011001001  10010101011001001
/spstMBE1/p3      01101010100110111  01101010100110111
/spstMBE1/p4      10000000000000000  01101010100110111  10000000000000000
/spstMBE1/p5      10000000000000000  01101010100110111  10000000000000000
/spstMBE1/p6      10000000000000000  01101010100110111  10000000000000000
/spstMBE1/p7      10000000000000000  10010101011001001  10000000000000000
```

Figure 9. Simulation results for a 16-bit multiplier using radix-8 modified Booth algorithm with Parallel Prefix adder



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	1,088	29,504	3%	
Logic Distribution				
Number of occupied Slices	568	14,752	3%	
Number of Slices containing only related logic	568	568	100%	
Number of Slices containing unrelated logic	0	568	0%	
Total Number of 4 input LUTs	1,095	29,504	3%	
Number used as logic	1,088			
Number used as a route-thru	7			
Number of bonded IOBs	64	250	25%	
Total equivalent gate count for design	7,167			
Additional JTAG gate count for IOBs	3,072			

Figure 10. Design Summary of Radix-2 MBA for Parallel Prefix Adder



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	1,745	9,312	18%	
Logic Distribution				
Number of occupied Slices	902	4,656	19%	
Number of Slices containing only related logic	902	902	100%	
Number of Slices containing unrelated logic	0	902	0%	
Total Number of 4 input LUTs	1,748	9,312	18%	
Number used as logic	1,745			
Number used as a route-thru	3			
Number of bonded IOBs	83	190	43%	
Total equivalent gate count for design	10,983			
Additional JTAG gate count for IOBs	3,984			

Figure 11. Design Summary of Radix-8 MBA for Parallel Prefix Adder

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. "Implementation of MAC using Modified Booth Algorithm". *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.

VI Conclusion

The different adders are compared for various measures and works well either in power dissipation or in delay. So the performance of each adder is different from the other. The adder is to avoid the unwanted glitches and spikes therefore switching power dissipation is minimized. The Radix -2 modified booth algorithm reduces the number of partial products to half by grouping of bits from the multiplier term in the multiplication operation, which improves the speed.

VII Future Scope

The modified booth algorithm which is different from the existing booth algorithm are commonly used. The Radix-2 and Radix-8 Booth Algorithm is used for all multiplication process that reduces the number of critical path, and reduces the power consumption. In this paper, 16- bit Radix-8 Modified Booth Algorithm using spurious power suppression technique and Radix-16 MBA is also implemented from the designed Radix-8 MBA. The benefits of miniaturization are high packing densities, good circuit speed and low power consumption. A fixed-width multiplier is required to maintain a fixed format and minimum accuracy loss to output data.

References

1. Young-Ho Seo and Dong-Wook Kim, (February 2010) 'A new VLSI architecture of parallel multiplier-accumulator based on radix-2 modified Booth algorithm', in IEEE Trans. On Very Large Scale Integration (VLSI) Systems, vol. 18, no. 2, pp.201-208.
2. Z. Huang and M. D.Ercegovac, (March 2005), 'High-performance low-power left-to right array multiplier design', IEEE trans.Comput., vol.54, no.3, pp.272–283.
3. G.Lakshmi Narayanan and B. Venkataramani, (July 2005), 'Optimization techniques for FPGA-based wave pipelined DSP blocks', IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol.13, no. 7, pp.783-792.
4. H.K.Chen, K.C.Chao, J.I.Guo, J.S.Wang and Y.S. Chu, (2005), 'An efficient spurious power suppression technique (SPST) and its applications on MPEG-4AVC/H.264 transform coding design',Proc. IEEE Int. Symps. Low Power Electron. Des., pp.155–160.
5. H.Lee, (2004) 'A power-aware scalable pipelined Booth multiplier',Proc. IEEE Int. SOC Conf., , pp.123–126.
6. J.Choi, J.Jeon and K.Choi, (2000), 'Power minimization of functional units by partially guarded computation', Proc. IEEE Int. Symp. Low Power Electron. Des., pp.131–136.
7. J. Fadavi-Ardekani,(June1993), 'M*N Booth encoded multiplier generator using optimized Wallace trees', IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 1, no. 2, pp.120–125.
8. K.H. Chen, Y.M. Chen, and Y.S. Chu, 'A versatile multimedia functional unit design using the spurious power suppression technique', in Proc. IEEE Asian Solid -State circuits Conf., 2006.

Cite this article as: R P Meenaakshi Sundhari, M Karthickumar, S Pavithra, E Madura. "Implementation of MAC using Modified Booth Algorithm". *International Conference on Information Engineering, Management and Security 2016*: 70-76. Print.