International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 7, July-2014, pp: (429-435), Impact Factor: 1.252, Available online at: www.erpublications.com

Comparison of Multiplier Accumulator Architectures for DSP Applications

Sonali Singh[1], Pawan Sharma[2]
[1]Third Year, B.E. (Hons.), Dept. of Electrical and Electronics, Birla Institute of Technology and Science, Pilani, India
[2]Lecturer, Dept. of Electrical and Electronics, Birla Institute of Technology and Science, Pilani, India

Abstract: This paper presents several architectures and designs of 8-bit Multiplier Accumulator (MAC) for DSP applications. Modifications have been made to existing architectures and their performances compared for speed, area and power consumption. The designs have been coded and simulated in Verilog using ModelSim and synthesized using Cadence RC Compiler and UMC 90nm standard CMOS technology library. Their layouts were generated using Cadence SoC Encounter and were successfully re-simulated in ModelSim.

Keywords: Multiplier and Accumulator (MAC), Digital Signal Processing (DSP), Abacus Multiplier, Vedic Multiplier, Wallace Tree architecture, Booth Multiplier, Modified Booth Algorithm (MBA), Kogge Stone Adder, Conditional Sum Adder, Carry Save Adder (CSA), High Performance Multiplier (HPM) Reduction Tree.

I. INTRODUCTION

Real-time signal processing like large capacity data processing, audio signal processing, video/image processing are the most advanced techniques in present multimedia and communication systems. The multiplier Accumulator (MAC) [1] and multipliers are the essential elements of digital signal processing which involves filtering, inner products and convolution. Most digital signal processing methods use non-linear functions such as Discrete Cosine Transform (DCT) [2] or Discrete Wavelet Transform (DWT) [3] which are basically accomplished by repetitive application of multiplication and addition. Hence, the speed of the multiplication and addition circuits determines the speed and performance of the entire processor. The multiplier requires the longest delay among the basic operational blocks in a digital system and determines the critical path generally. The radix-4 Modified Booth's Algorithm (MBA) [4] is commonly used for high speed multiplication, however, it is the not the ultimate solution to the long critical path [5], [6] problem of multipliers. It, therefore, becomes imperative to modify existing architectures or design new ones which are not only faster but also optimized for minimum cost (in terms of Silicon area) and power consumption.

In order to achieve that end, this paper presents the design and implementation of 6 different MAC architectures based on popular algorithms such as Booth, Modified Booth, Wallace tree, Vedic, Abacus and Baugh−Wooley for multiplication of two 8-bit numbers and generation of their partial products. Carry Save Adder (CSA) and High Performance Multiplier (HPM) Reduction tree were used for reduction of the partial products. Various fast adders such as Carry Look Ahead (CLA), CSA, Kogge Stone and Conditional Sum have been combined with these multipliers and their performance compared in terms of speed, area and power delay product. Three of the designs are for unsigned numbers whereas the remaining three are for signed numbers.

This paper is organized as follows. Section II describes a general MAC operation, section III presents the different MAC implementations along with their characteristics, section IV draws a comparison of the performance of these architectures amongst themselves as well as with a few reference architectures and lastly, section V states the conclusion drawn from this work.

II. OVERVIEW OF MAC

In this section, basic MAC operation is introduced. A multiplier can be divided into three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand (X) and the multiplier (Y). The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry. If the process to accumulate the multiplied results is included, a MAC consists of four steps.

General hardware architecture of this MAC is shown in Fig.1. It executes the multiplication operation by multiplying input multiplier Y and the multiplicand X. This is added to the previous multiplication result Z as the accumulation step.

III. THE PROPOSED MAC ARCHITECTURES

In this section, the different MAC architectures are described in some detail.

A. Booth MAC

The multiplier in the Booth MAC is based on Booth's Algorithm [7] for partial product generation, which generates 8 partial products. The partial products are reduced using an 8-bit

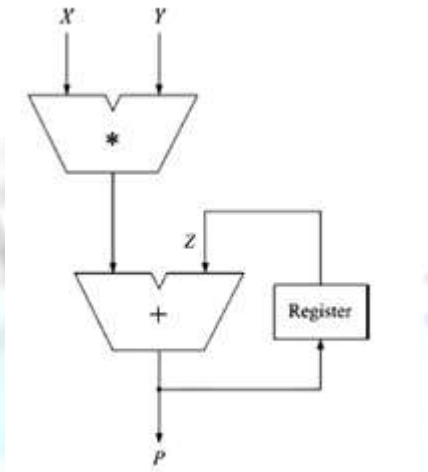

Fig. 1. Hardware Architecture of General MAC

CSA with sign-extension an illustration of which is given in fig. 2. The CSA outputs the product of the operands which is stored in a register and is added to the previous sum in the next clock cycle. An 8-bit Conditional Sum Adder is used for accumulation [15]. The designed MAC is pipelined to yield faster results and operates on signed numbers.

B. Wallace Tree MAC

In the Wallace Tree MAC, multiplier partial products were generated by simple and-ing of two 8-bit unsigned inputs and reduced using an 8-bit Wallace tree [8]. The Conditional Sum Adder [15] is used for accumulation. The two stages of the MAC (first, after product formation and second, after accumulation) are pipelined to give faster results.

C. Parallel Booth MAC

This architecture is based on the radix-2 Modified Booth Algorithm [4] which reduces the number of partial products to four from eight for an 8-bit signed multiplier. The Parallel Booth MAC is organized into 3 steps where accumulation is merged with the addition of the partial products into the CSA [9]. The CSA of the proposed MAC requires a total of 4 rows of full adders instead of 5 required in [9] and is based on the method given in fig. 2.

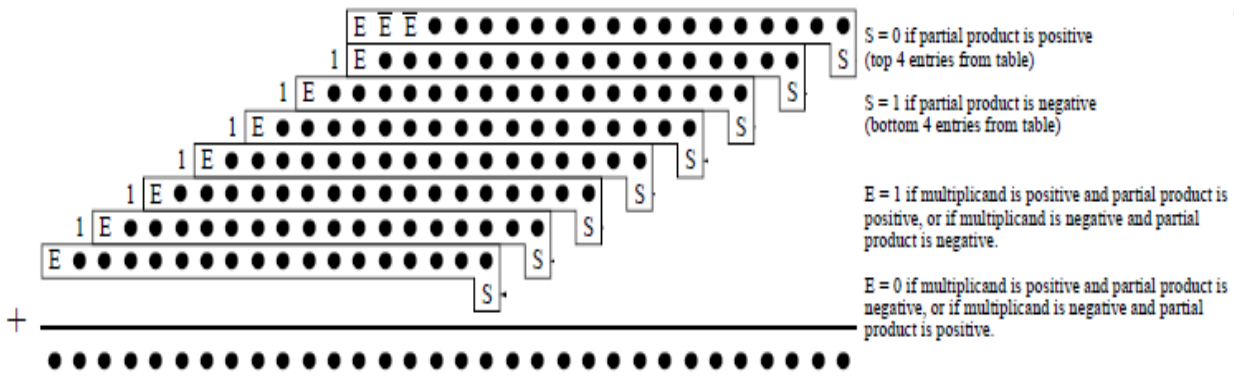

Fig. 2. Signed 16-bit CSA for Parallel MAC with sign extension

The reference MAC [9] does not make use of sign extension in the CSA whereas the proposed MAC does. The sign extension method has been elucidated in fig. 2. It finally generates two 16-bit outputs called sum and carry, the least significant 8 bits of which are added along with N (for conversion of 1's complement into 2's complement) using 2-bit CLAs. The most significant 8 bits are added using the 8-bit CLA whenever the MAC result is required. The three stages are pipelined to improve speed.

D. Low Power MAC with HPM Reduction Tree

This MAC architecture is composed of a partial product generator which uses the Baugh−Wooley Algorithm [10], [11] for generation of partial products of signed numbers, an HPM reduction tree [12] for their reduction into 15-bit sum and carry. These are added using a 15-bit Kogge stone Adder [13], [14]. A Conditional Sum Adder [15] is used for accumulation. The process requires three steps as illustrated in fig. 3.

E. Vedic MAC

This architecture is based on the vertical and crosswise Vedic multiplication principle [16] of two numbers. Based on this principle, the architectural implementation of a 2x2 multiplier is done. The Vedic multiplier has a modular nature and hence a 2x2 multiplier can be used to make a 4x4 multiplier and so on. The multiplier for the proposed MAC has been designed using four 4x4 Vedic multipliers. This was combined with a Conditional Sum Adder to form MAC and was pipelined to improve speed.

F. ABACUS MAC

This MAC architecture is designed for two 8-bit unsigned numbers and comprises of a multiplier called ABACUS [17] and a 16-bit Conditional Sum Adder for accumulation. ABACUS uses a particular threshold function to implement multiple fast carry operations in parallel through a cellular array, and therefore significantly deviates from the conventional approaches based on half/full adder or counter building blocks [17]. As per this algorithm, the partial products are first aligned as shown in fig. 4. The next part consists of two steps. First is the downward movement of a flag to replace a zero beneath it. This step can be visualized by the vertically downward movement of a bead through empty slots in a mechanical ABACUS. The compression of 1s to the bottom is a critical operation which quickly discards the non-contributing bits from the two dimensional array. The second step consists of the leftward movement of these compressed 1s and their transition follows the basic carry principle of a mechanical ABACUS. For a particular column having N flags, 2^i flags should be moved $\lfloor \log_2 N \rfloor$ columns to the left ($2^i \leq N$, $i \in \mathbb{N}$). This operation should be performed on all columns simultaneously in one clock cycle. The bit-compression and leftward carry steps should be repeated until only the last row is occupied with flags. Fig. 5 illustrates the ABACUS state after the first and fourth carry/compression cycle of multiplication. The number of clock cycles required for a MAC operation is different for different inputs. However, the proposed MAC been designed to produce results after 17 clock cycles as the worst case input X=FF, Y=FF requires 17 clock cycles to produce results.

IV. ASIC IMPLEMENTATION AND SIMULATION RESULTS

The ASIC implementation of the proposed design follows the Cadence design flow. The design has been developed using Verilog-HDL and synthesized in Encounter RTL Compiler using typical libraries of UMC 90 nm technology. The Cadence SoC Encounter is adopted for Placement & Routing (P&R) (Encounter User Guide 2008). It makes use of the Verilog net-list as well as the timing constraint file generated post-synthesis for creating the layout. Parasitic extraction was performed using Encounter Native RC extraction tool. All of the timing analysis was performed at the nominal voltage level 0.9 V, for the 90 nm process technology. Temperature was set at 125°C. The worst case delays of the multipliers were examined with back-annotation of parasitic resistances and capacitances extracted from the layouts. Each standard cell library used for this design includes LEF (Library Exchange Format) files and timing files. An LEF file contains the physical information for a process technology as

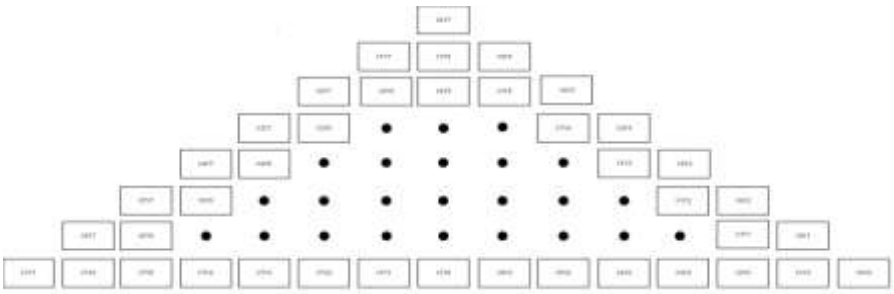

Fig. 3. Alignment of Partial Products in an 8-bit ABACUS Multiplier

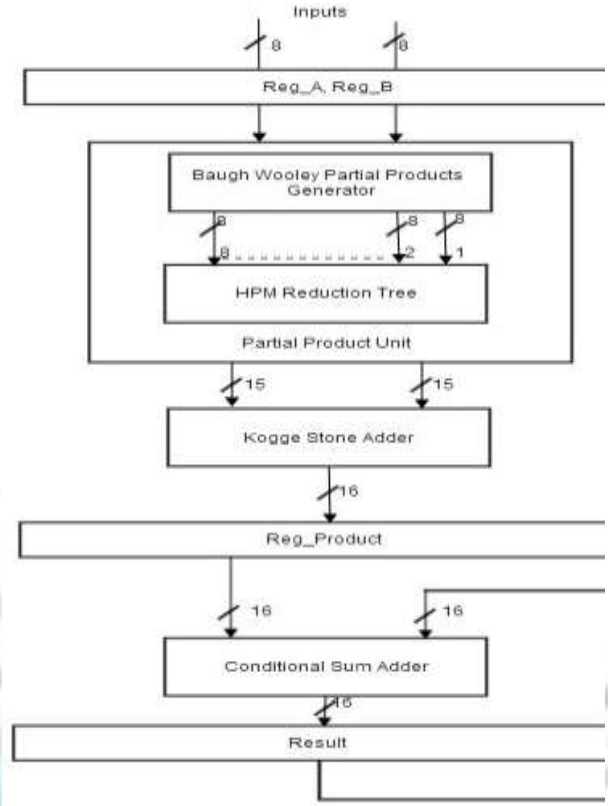

Fig. 4. Hardware Architecture of Low Power MAC.

well as geometric abstracts of all of the cells. All of the timing files used for this research is for the nominal temperature, voltage, and process corner, often named "typical.lib". Table 1 shows the final results obtained after Placement and Routing of each of the designs in Cadence SoC Encounter. The Verilog net-list generated post-simulation was back annotated and its functionality was successfully verified.

A. Comparison of Proposed Modified Booth MAC with existing Architectures

Table 2 shows a pre-layout performance comparison based on power, area, delay and power delay product between the proposed 8-bit Modified Booth MAC Architecture and some existing architectures [18] and [19].

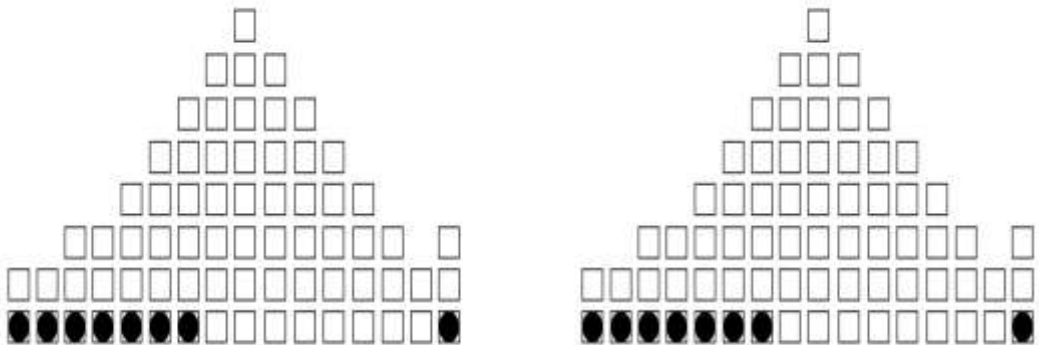

Fig. 5. ABACUS State after first and fourth carry/ compression cycle of the multiplication between X=0xFF and Y=0xFF.

TABLE I. FINAL RESULTS AFTER PLACEMENT AND ROUTING FROM CADENCE SoC ENCOUNTER

Type of MAC	Hardware Resource Consumption						
	Power (mW)				*Slack (ns)*	*Gate Count*	*Maximum Frequency (MHz)*
	Leakage	*Dynamic*	*Internal*	*Total*			
Booth	0.000123	0.03513	0.03723	0.07248	+11.255	439	46.51
Wallace	0.0002287	0.05563	0.04435	0.1002	+12.934	257	77.10
Parallel Booth	0.000093	0.044	0.0421	0.0862	+11.255	365	52.63
Low Power	0.000134	0.03501	0.05741	0.09255	+7.934	435	87.00
Vedic	0.000118	0.02643	0.02485	0.0514	+9.23	315	62.50
ABACUS	0.000431	0.1994	0.2053	0.4051	+6.856	1930	58.82

B. Comparison of Proposed Wallace Tree and Low Power MACs with existing Architectures

Table 3 shows a post-layout performance comparison based on power, delay and power delay product (PDP) of proposed 8-bit Wallace Tree (WT) and Low Power Multiplier Accumulators with some existing multiplier architectures [20]. Fig. 6 also shows a graphical comparison of the power delay product of the above mentioned architectures.

C. Comparison of Proposed Vedic Multiplier with existing Architectures

Table 4 shows a post-layout delay comparison of proposed 8-bit Vedic Multiplier with the existing architectures [16] and [21].

Table II. Pre-Layout Comparison Of Proposed And Existing Modified Booth Mac Architectures.

Type of Modified Booth Mac	Power (mW)	Area (μm^2)	Delay (ns)	Power Delay Product (pJ)
Proposed	0.35625	6006	5.30	1.8881
[18]	0.76000	6091	3.58	2.7208
[19]	0.15300	6603	5.50	0.8415

Table III. Post-Layout Comparison Of Proposed Macs With Existing Multiplier Architectures

	Type of MAC/Multiplier					
	Proposed		Reference [20]			
	WT MAC	Low Power	Slansky Adder	KSA	CSA	Modified CSA
Power (mW)	0.135	0.088	0.372	0.116	0.140	0.176
Delay (ns)	13.200	11.500	26.066	22.623	20.749	17.54
PDP (pJ)	1.782	1.012	9.7	2.6239	2.905	3.087

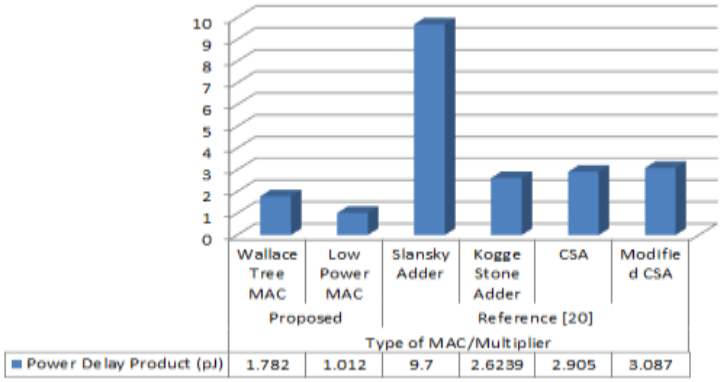

Fig. 6. Comparison of PDP of Wallace and Low Power MAC with reference multiplier architectures.

Table IV. Delay Comparison Of Proposed Vedic Multiplier With Existing Architectures

Type of Vedic Multiplier	Proposed	[16] (device1)	[16] (device2)	[21]
Delay (ns)	15.8	13.07	25.06	20.523
Reduction in delay		-20.88%	37%	23%

D. Comparison of Accumulator Architectures

Two accumulator architectures have been used in the MACs discussed above for fast addition. One is the Conditional Sum Adder and the other is the Kogge Stone Adder (KSA). Each of these being 16-bit wide, their performance in terms of power, delay, area and power delay product (PDP) and area are listed in table 5.

V. CONCLUSION

Six different MAC Architectures, along with their post-layout performance have been presented in this paper. The post-layout results of the proposed MAC architectures given in table 1 suggest that the Low Power Vedic using HPM Reduction is the fastest, operating at a maximum clock frequency of 87 MHz The Wallace Tree MAC has the minimum area consumption with a gate count of 257, whereas the Vedic MAC Architecture has the lowest power consumption at 51.4µW for 8-bit operands. The ABACUS Architecture, although designed for low power consumption does not yield the desired results because the proposed designed takes 17 clock cycles to produce the desired output which makes its dynamic power consumption very high as compared to others. The Modified Booth MAC has been compared with existing architectures in table 2. As we can see from the table, there is a 30% reduction in the power-delay-product of the proposed design and a reduction in area consumption of about 2% as compared to [18].

Table 3 shows a comparison between the proposed Wallace Tree MAC with four different multiplier architectures given in [20]. The post-layout results suggest that our MAC design consumes 64% less power as compared to the multiplier with the Slansky Adder, 16% more power as compared to the Kogge Stone Adder (KSA) multiplier, 4% less power as compared to the CSA multiplier and 13% less power as compared to the modified CSA multiplier. The delay of the proposed Wallace Tree (WT) MAC is 50% less than that of the Slansky Adder, 42% less than that of the Kogge Stone Adder, 36% less than that of the CSA and 25% less than that of the modified CSA. The power-delay-product of our Wallace Tree MAC is also the least. Also, the proposed Low Power MAC is the fastest as compared to the reference architectures [20] with the minimum power consumption.

Table 4 shows the delay comparison of 8-bit Vedic Multiplier with reference architectures [16] and [21]. We can see a reduction of 37% and 23% in the delay of the proposed Vedic Multiplier as compared to that described in [16] and [21] respectively. Lastly, a comparison has been drawn after Placement and Routing of the accumulators used for the various MAC Architectures proposed above. As we can see, the 16-bit Kogge Stone Adder has proven superior in terms of speed and power consumption whereas the 16-bit Conditional Sum Adder takes up a smaller area as is suggested by its lower gate count.

Table V. Post-Layout Performance Comparison Of Accumulators

Type of Accumulator	Power (mW)		
	Internal	Switching	Leakage
Conditional Sum	0.007122	0.01753	$3.478x10^{-5}$
Kogge Stone	0.005025	0.01409	$1.334x10^{-5}$

Type of Accumulator	Total Power(mW)	Gate Count	Maximum Frequency(Mhz)
Conditional Sum	0.02468	93	133.33
Kogge Stone	0.01913	125	166.66

ACKNOWLEDGMENT

The authors thank the Birla Institute of Technology and Sciences for providing effective software tools critical for the successful completion of this project.

REFERENCES

[1] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw- Hill, 1984.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
[2] Information Technology-Coding of Moving Picture and Associated Autio, MPEG-2 Draft International Standard, ISO/IEC 13818-1, 2, 3,1994.
[3] JPEG 2000 Part I Fina1119l Draft, ISO/IEC JTC1/SC29 WG1.
[4] O. L. MacSorley, "High speed arithmetic in binary computers," Proc. IRE, vol. 49, pp. 67–91, Jan. 1961.
[5] S. Waser and M. J. Flynn, Introduction to Arithmetic for Digital Systems Designers. New York: Holt, Rinehart and Winston, 1982.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
[6] A. R. Omondi, Computer Arithmetic Systems. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[7] A. D. Booth, "A signed binary multiplication technique," Quart. J. Math., vol. IV, pp. 236–240, 1952.
[8] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[9] Young-Ho Seo and Dong-Wook Kim , "A New VLSI Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm," IEEE transactions on very large scale integration (VLSI) systems, vol. 18, no. 2, Feb. 2010.
[10] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," IEEE Transactions on Computers, vol. 22, pp. 1045–1047, 1973.
[11] M. Själander and P. Larsson-Edefors, "The case for HPM based Baugh-Wooley multipliers," Tech. Rep. 08-8, Chalmers University of Technology, Goteborg, Sweden, 2008.
[12] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Själander, D. Johansson, and M. Schölin, "Multiplier reduction tree with logarithmic logic depth and regular connectivity," in Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 4–8, May 2006.
[13] J. Rabaey, "Digital Integrated Circuits: A Design Perspective", Prentice Hall, 1996.
[14] Swaroop Ghosh, Patrick Ndai and Kaushik Roy, "A Novel Low Overhead Fault Tolerant Kogge-Stone Adder Using Adaptive Clocking," 2008.
[15] J. Skylansky, "An evaluation of several two-summand binary adders," March, 1960.
[16] Devika Jaina, Kaviraj Sethi and Rutuparna Panda, "Vedic Mathematics Based Multiply Accumulate Unit," International Conference on Computational Intelligence and Communication Systems, 2011.
[17] Ali Muhtaroglu, "ABACUS: A Novel Array Multiplier-Accumulator Architecture for Low Energy Applications," 2010.
[18] F. Elguibaly, " A fast parallel multiplier-accumulator using the modified Booth algorithm," IEEE Trans. Circuits Syst., vol. 27, pp 902-908, Sept. 2000.
[19] J. Y. Yaswanth Babu and P. Dinesh Kumar, "A New Multiplier - Accumulator Architecture based on High Accuracy Modified Booth Algorithm," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 2, Issue 3, March 2013.
[20] E. Prakash, R. Raju and Dr. R. Varatharajan, "Effective Method for Implementation of Wallace Tree Multipliers Using Fast Adders," Journal of Innovative Research and Solutions (JIRAS), Dec. 2013.
[21] Yogita A. Navghade, Prof A.C.Kailuke, Prof N.G.Narole, "Design & Implementation of High Speed N- Bit Reconfigurable Multiplier Using Vedic Mathematics for DSP Applications," International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), 2013.