International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 14
October2016

Implementation and Design of High Performance 128 bit parallel prefix MAC unit

G Rupesh
M.Tech Student Scholar
Department of Electronics & Communication Engineering,
Narsimha Reddy Engineering College, Telangana, India.
rupeshchowdary9@gmail.com

D Kiran Kumar
Assistant professor
Department of Electronics & Communication Engineering,
Narsimha Reddy Engineering College, Telangana, India.
kirannrcm@gmail.com

Abstract: **Digital signal processing (DSP) applications, the critical operations usually involve many multiplications and/or accumulations. So, for real time signal processing applications, high throughput multiplier–accumulator (MAC) is always a key element to achieve a high-performance digital signal processing application. This is because speed and throughput rate are always the concerns of digital signal processing systems. This is because the limited battery energy of these portable products restricts the power consumption of the system. The multiplier is designed using single precision multiplier and the adder is done with parallel prefix adder. The total operation is coded with VERILOG-HDL, synthesized and simulated using Xillinx ISE 14.7.**

Key Words: Single Precision multiplier, Parallel prefix adder, multiplier and accumulator (MAC).

1. INTRODUCTION

Multiply-accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC, or MAC unit); the operation itself is also often called a MAC or a MAC operation. The MAC operation modifies an accumulator a.

$$\mathbf{a \longleftarrow a + (b \times c)}$$

MAC unit is an inevitable component in many digital signal processing (DSP) applications involving multiplications and/or accumulations. MAC unit is used for high performance digital signal processing systems. The DSP applications include filtering, convolution, and inner products. Most of digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) or discrete wavelet transforms (DWT). Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic determines the execution speed and performance of the entire calculation. Multiplication-and-accumulate operations are typical for digital filters. Therefore, the functionality of the MAC unit enables high-speed filtering and other processing typical for DSP applications. Since the MAC unit operates completely independent of the CPU, it can process data separately and thereby reduce CPU load. The application like optical communication systems which is based on DSP, require extremely fast processing of huge amount of digital data. The Fast Fourier Transform (FFT) also requires addition and multiplication. 128 bit can handle larger bits and have more memory.

A MAC unit consists of a multiplier and an accumulator containing the sum of the previous successive products. The MAC inputs are obtained from the memory location and given to the multiplier block. The design consists of 128 bit Single Precision multiplier, 256 bit Parallel prefix adder and a register. This paper is divided into six sections. In the first section the introduction about MAC unit is discussed. In the second section discuss about the detailed operation of MAC unit. The third and fourth section deals with the operation of Single Precision multiplier and Parallel prefix adder respectively. In the fifth section, the obtained result for the 128 bit MAC unit is discussed and finally the conclusion is made in the sixth section.

2. MAC OPERATION

The Multiplier-Accumulator (MAC) operation is the key operation not only in DSP applications but also in multimedia information processing and various other applications. As mentioned above, MAC unit consist of multiplier, adder and register/accumulator. In this paper, we used 128 bit Single Precision multiplier. The MAC inputs are obtained from the memory location and given to the multiplier block. This will be useful in 128 bit digital signal processor. The input which is being fed from the memory location is 128 bit. When the input is given to the multiplier it starts computing value for the given 128 bit input and hence the output will be 256 bits. The multiplier output is given as the input to parallel prefix adder which performs addition.

The function of the MAC unit is given by the following equation:

$$\textbf{Output} = \sum \textbf{Ai Bi}$$

The output of carry save adder is 257 bit i.e. one bit is for the carry (256bits+ 1 bit). Then, the output is given to the accumulator register. The accumulator register used in this design is parallel in parallel out (PIPO). Since the bits are huge and also parallel prefix adder produces all the output values in parallel, PIPO register is used where the input bits are taken in parallel and output is taken in parallel. The output of the accumulator register is taken out or fed back as one of the input to the parallel prefix adder.

The figure 1 shows the basic architecture of MAC unit.

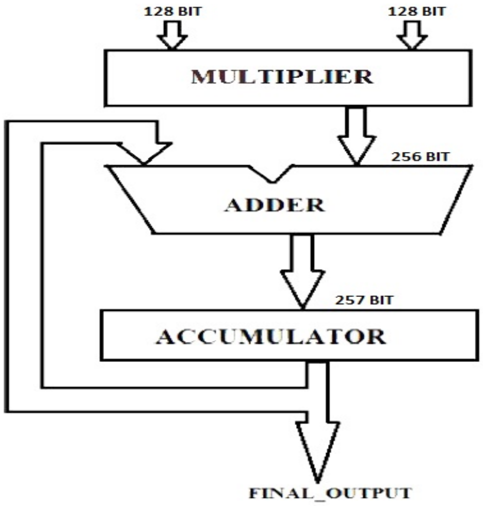


Figure 1: Basic architecture of MAC unit

3. EXISTING CONCEPT

Modified Wallace Multiplier

A modified Wallace multiplier is an efficient hardware implementation of digital circuit multiplying two integers. Generally in conventional Wallace multipliers many full adders and half adders are used in their reduction phase. Half adders do not reduce the number of partial product bits. Therefore, minimizing the number of half adders used in a multiplier reduction will reduce the complexity. Hence, a modification to the Wallace reduction is done in which the delay is the same as for the conventional Wallace reduction. The modified reduction method greatly reduces the number of half adders with a very slight increase in the number of full adders.

Reduced complexity Wallace multiplier reduction consists of three stages. First stage the N x N product matrix is formed and before the passing on to the second phase the product matrix is rearranged to take the shape of inverted pyramid. During the second phase the rearranged product matrix is grouped into non-overlapping group of three as shown in the figure 2, single bit and two bits in the group will be passed on to the next stage and three bits are given to a full adder. The number of rows in the in each stage of the reduction phase is calculated by the formula.

$$\mathbf{r_{i+1} = 2\ [r_{\ i/3}] + r_i\ mod3} \qquad (1)$$

$$\mathbf{r_i\ mod3 = 0,\ then\ r_{i+1} = 2\ [r_{\ i/3}]} \qquad (2)$$

If the value calculated from the above equation for number of rows in each stage in the second phase and the number of row that are formed in each stage of the second phase does not match, only then the half adder will be used. The final product of the second stage will be in the height of two bits and passed on to the third stage. During the third stage the output of the second stage is given to the carry propagation adder to generate the final output.

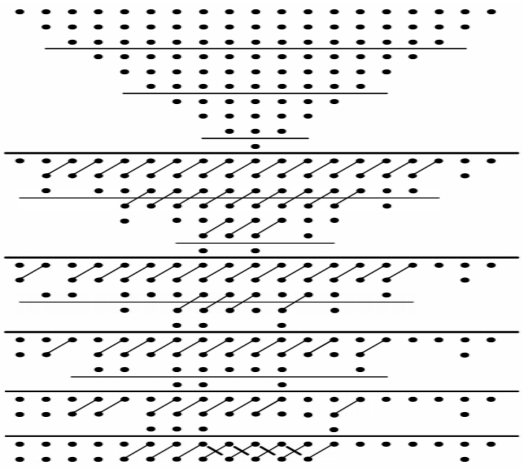


Figure 2: Modified Wallace IO-bit by IO-bit reduction

Thus 64 bit modified Wallace multiplier is constructed and the total number of stages in the second phase is 10. As per the equation the number of row in each of the 10 stages was calculated and the use of half adders was restricted only to the 10th stage. The total number of half adders used in the second phase is 8 and the total number of full adders that was used during the second phase is slightly increased that in the conventional Wallace multiplier. Since the 64 bit modified Wallace multiplier is difficult to represent, a typical l0-bit by 10-bit reduction shown in figure 2 for understanding. The modified Wallace tree shows better performance when carry save adder is used in final stage instead of ripple carry adder. The carry save adder which is used is considered to be the critical part in the multiplier because it is responsible for the largest amount of computation.

Carry save Adder

In this design 128 bit carry save adder is used since the output of the multiplier is 128 bits (2N). The carry save adder minimize the addition from 3 numbers to 2 numbers. The propagation delay is 3 gates despite of the number of bits. The carry save adder contains n full adders, computing a single sum and carries bit based mainly on the respective bits of the three input numbers. The entire sum can be calculated by shifting the carry sequence left by one place and then appending a 0 to most significant bit of the partial sum sequence. Now the partial sum sequence is added with ripple carry unit resulting in n + 1 bit value. The ripple carry unit refers to the process where the carryout of one stage is fed directly to the carry in of the next stage. This process is continued without adding any intermediate carry propagation.

Since the representation of 128 bit carry save adder is infeasible , hence a typical 8 bit carry save adder is shown in the figure 3.Here we are computing the sum of two 128 bit binary numbers, then 128 half adders at the first stage instead of 128 full adder. Therefore , carry save unit comprises of 128 half adders, each of which computes single sum and carry bit based only on the corresponding bits of the two input numbers. If x and y are supposed to be two 128 bit numbers then it produces the partial products and carry as S and C respectively.

$$\begin{aligned} s_i &= x_i \; \hat{} \; y_i \\ c_i &= x_i \;\&\; y_i \qquad (3) \end{aligned}$$

During the addition of two numbers using a half adder, two ripple carry adder is used. This is due the fact that ripple carry adder cannot compute a sum bit without waiting for the previous carry bit to be produced, and hence the delay will be equal to that of n full adders. However a carry-save adder produces all the output values in parallel, resulting in the total computation time less than ripple carry adders. So, Parallel in Parallel out (PIPO) is used as an accumulator in the final stage.

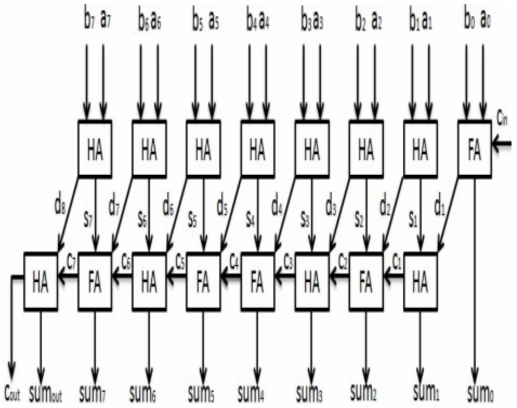
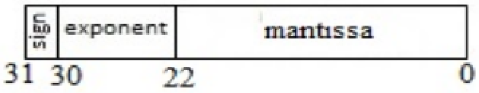


Figure 3: 8 bit carry save adder

4. PROPOSED CONCEPT

Single Precision Multiplier

In MAC unit multiplier block efficiency is increased using Single precision multiplier. Precision numbers have one sign bit, 34- bit exponent field and a 94-bit mantissa, for a total of 64 bits. one sign bit, 11- bit exponent field and a 52-bit mantissa, for a total of 64 bits. We have done floating point multiplication for Single precision numbers which include an one sign bit, a 8-bit exponent field and a 23-bit mantissa, for a total of 32 bits. The standard mandates binary floating point data be encoded on three fields: a one bit sign field, followed by exponent bits encoding the exponent offset by a numeric bias specific to each format, and bits encoding the significant



The scientific notation of IEEE 754 standard numbers is shown below and is stored in signed magnitude format.

$$\pm \textbf{mantissa} * 2^{\textbf{exponent}}$$

The sign bit is 0 for positive numbers and 1 for negative numbers

Architecture of Precision Multiplier

The Single Precision Unit Architecture comprised of two Operands A and B. The RM MODE is the round unit mode. For Normalization we are using two Pre Normalize block, one for Addition and Subtraction, other for Multiplication.

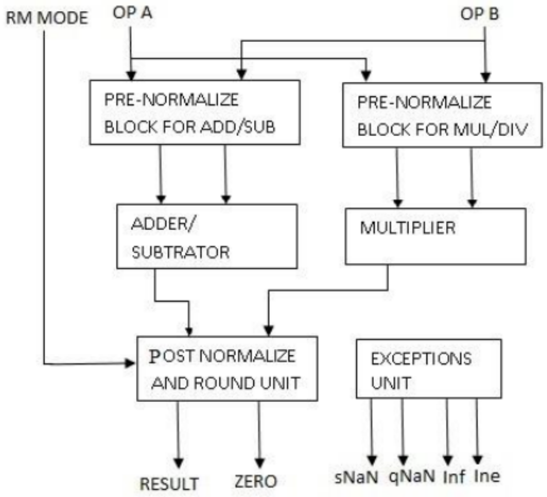

RM MODE
OP A
OP B
PRE-NORMALIZE BLOCK FOR ADD/SUB
PRE-NORMALIZE BLOCK FOR MUL/DIV
ADDER/ SUBTRATOR
MULTIPLIER
POST NORMALIZE AND ROUND UNIT
EXCEPTIONS UNIT
RESULT
ZERO
sNaN
qNaN
Inf
Ine

Figure 4: Architecture of Single Precision Unit

1. Pre Normalize Block for Adder and Subtractor Calculate the difference between the smaller and larger exponent. Adjust the smaller fraction by right shifting it, determine if the operation is an add or subtract after resolving the sign bits. Check for NaNs on inputs.

2. Pre Normalize Block for Multiplication: Computes the sum/difference of exponents, checks for exponent overflow, underflow condition and INF value on an input.

i. Add and Sub - 24 bit integer adder and Subtractor.

ii. Multiply - 2 cycle 24-bit Boolean integer multiplier

3. Post Normalize and Round Unit - Normalize fraction and exponent. Also do all the rounding's in parallel and then pick the output corresponding to the chosen rounding mode.

4. Exceptions Unit – This unit generates the exception signals like sNAN, qNAN, Inf and Ine The IEEE standard defines two classes of NaNs (non numbers):

i. quiet NaNs (qNaNs) : A qNaN is a NaN with the most significant fraction bit set.

ii. Signalling NaNs (sNaNs): A sNaN is a NaN with the most significant fraction bit clear.

The single precision floating point format is divided into three main parts corresponding to the sign, exponent and mantissa. Multiplication of the two operands is done in three parts and thereby obtaining the Product. The first part of the product which is the sign is determined by an exclusive OR function of the two input signs. The exponent of the product which is the second part is calculated by adding the two input exponents. The third part which is the significant of the product is determined by multiplying the two inputs significant each with a '1'concatenated to it.

A multiplication of two floating-point numbers is done in four steps:

- Non-signed multiplication of mantissas: it must take account of the integer part, implicit in normalization. The number of bits of the result is twice the size of the operands (48 bits).
- Normalization of the result, the exponent can be modified accordingly.
- Addition of the exponents, taking into account the bias.
- Calculation of the sign.

Parallel Prefix Adder

Parallel Prefix adder is that it is primarily fast when compared with carry save adders. Parallel Prefix adders (PPA) are family of adders derived from the commonly known carry look ahead adders. These adders are best suited for adders with wider word lengths. PPA circuits use a tree network to reduce the latency to O (log2 n) where "n" represents the number of bits. Parallel Prefix Adders (PPA) is variations of the well-known carry look ahead adder (CLA). The difference between a CLA and a PPA lies in the second stage which is responsible for the generation of the carry signals of the binary addition.

Parallel-prefix adder structure

Parallel-prefix structures are found to be common in high performance adders because of the delay is logarithmically proportional to the adder width. PPA's basically consists of 3 stages

- Pre computation
- Prefix stage
- Final computation

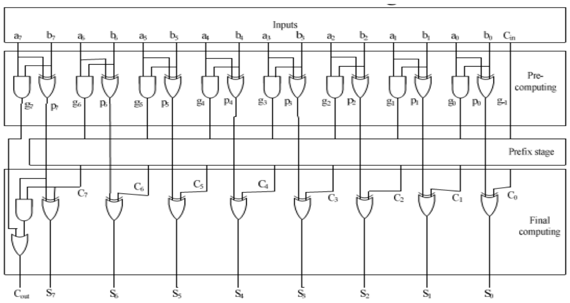

Inputs
a7 b7 a6 b6 a5 b5 a4 b4 a3 b3 a2 b2 a1 b1 a0 b0 Cin
Pre-computing
g7 p7 g6 p6 g5 p5 g4 p4 g3 p3 g2 p2 g1 p1 g0 p0 g-1
Prefix stage
C7 C6 C5 C4 C3 C2 C1 C0
Final computing
Cout S7 S6 S5 S4 S3 S2 S1 S0

Figure 5: Parallel-Prefix Adder 3 Stage Structure

Pre computation

In pre computation stage, propagates and generates are computed for the given inputs using the given equations.

Prefix stage

In the prefix stage, group generate/propagate signals are computed at each bit using the given equations. The black cell (BC) generates the ordered pair in equation, the gray cell (GC) generates only left signal, following.

$$\mathbf{G_{i:k} = G_{i:j} + P_{i:j} \,.\, G_{j-1:k}}$$
$$\mathbf{P_{i:k} = P_{i:j} \,.\, P_{j-1:k}} \qquad (4)$$

More practically, the equations (4) can be expressed using a symbol "o "denoted by Brent and Kung. Its function is exactly the same as that of a black cell i.e.

$$\mathbf{G_{i:k} : P_{i:k} = (G_{i:j} : P_{i:j})\ o\ (G_{j-1:k} : P_{j-1:k})} \qquad (5)$$

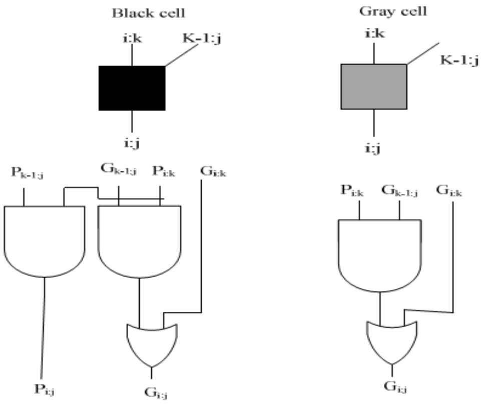


Figure 6: Black and Gray Cell logic Definitions

The "o" operation will help make the rules of building prefix structures.

Final computation

In the final computation, the sum and carryout are the final output.

$$S_i = P_i \cdot G_{i-1:-1}$$
$$C_{out} = G_{n:-1} \qquad (6)$$

Where "-1" is the position of carry-input. The generate/propagate signals can be grouped in different fashion to get the same correct carries. Based on different ways of grouping the generate/propagate signals, different prefix architectures can be created. Figure 3 shows the definitions of cells that are used in prefix structures, including BC and GC. For analysis of various parallel prefix structures. The 16 bit SKA uses black cells and gray cells as well as full adder blocks too. This adder computes the carries using the BC's and GC's and terminates with 4 bit RCA's. Totally it uses 16 full adders. The 16 bit SKA is shown in figure 4.4. In this adder, first the input bits (a, b) are converted as propagate and generate (p, g). Then propagate and generate terms are given to BC's and GC's. The carries are propagated in advance using these cells. Later these are given to full adder blocks.

Prefix Addition

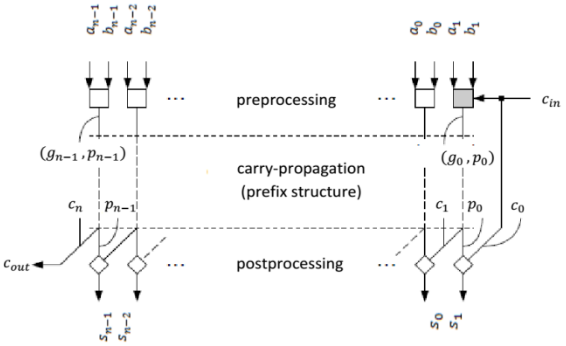


Figure 7: Prefix Addition structure

The structure of the prefix network determines the type of the prefix adder. Over the years several classical topologies have been proposed which optimize for one of the following parameters: area, logic-depth size, fan-out and interconnect count. The Kogge-Stone adder has minimum depth and fan-out but maximum area and interconnects count. Ladner and Fischer proposed a general way to construct the prefix network which includes the minimum depth case of the Sklansky topology with improved area requirement. The Brent-Kung adder has minimum area but maximum depth. The Han-Carlson adder combines the Brent-Kung and the Kogge-Stone structures in order to balance between logic depth and interconnect count. Knowles proposed a way to construct adders of minimum depth with all possibilities of fan-out count ranging from the minimum case seen in the Kogge-Stone to the maximum fan-out seen in the Sklansky adder.

5. EXPERMENTAL RESULTS

The Design is developed using Verilog and Synthesized using Xilinx 14.7 ISE. As a previous work different MAC Units were developed using different combination of multipliers and adders. Here in this implementation we selected modified Wallace multiplier with carry save adder to compare with our efficient method that is precision multiplier using parallel prefix adder and measure the performance parameters of MAC unit. The parameters are area, delay where area is measured in terms of number of slice and delay is measured in nanoseconds respectively.

When code is checked our multiplier is shown in schematic way as follow

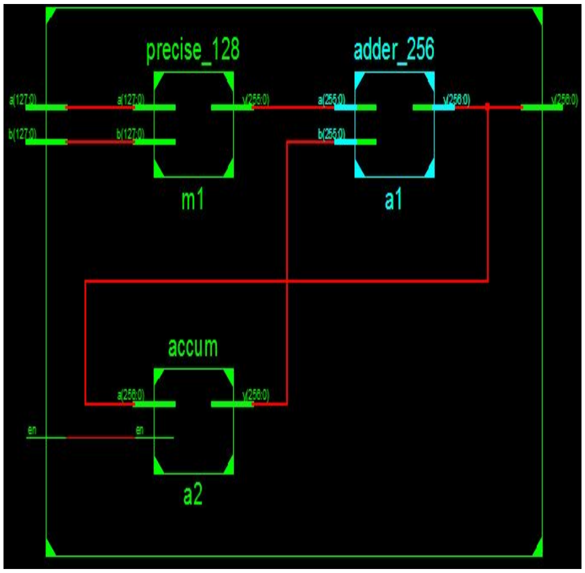


Figure 8: RTL view of Mix Column operation

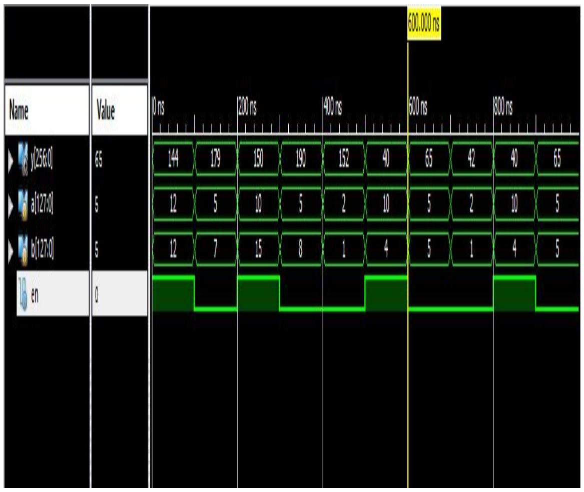


Figure 9: Simulation waveform of Mix Column Step

6. CONCLUSION

Hence, a High Performance 128 bit MAC Unit is designed and implemented using Precision Multiplier and Parallel Prefix Adder. When compared with modified Wallace multiplier with carry save adder MAC Unit which is developed and more efficient than earlier MAC units using different combinations of multipliers and adders the designed Precision Multiplier offers High Performance with Less Area, and Less Propagation Delay, which further increases the overall speed of MAC Unit. This MAC Unit is designed using Verilog - HDL and Synthesized using Xilinx 14.7 ISE.

7. REFERENCES

1. “*Implementation of high performance 64 bit MAC unit by modified Wallace multiplier*” IEEE Conference On Very Large Scale Integrated Circuits And Systems, January 2013.(Base paper).

2. Young-Ho seo and Dong wook kim, “*New vlsi architecture of parallelmultiplier-accumulator based radix 2modified booth algorithm*”IEEE Trans. Very Large Scale Integr. Syst., vol. 18, no. 2, pp. february 2010.

3. A. R. Cooper, “Parallel architecture modified Booth multiplier,”Proc.Inst. Electr. Eng. G, vol. 135, pp. 125–128, 1988.

4. D. Mohapatra, G. Karakonstantis, and K. Roy, “Reduced complexity wallace multiplier reduction” inProc. IEEE/ACM Int. Symp. Low Power Electron. Design,
Aug. 2009, pp. 195–200.

5. Jagadguru Swami Sri Bharath, Krsna Tirathji, “Vedic Mathematics or Sixteen Simple Sutras From The Vedas”, Motilal Banarsidas, Varanasi(India),1986.

6. A.P. Nicholas, K.R Williams, J. Pickles, “Application of Urdhava Sutra”, Spiritual Study Group, Roorkee
(India),1984.

7. Devika, K. Sethi and R.Panda, Vedic Mathematics Based Multiply Accumulate Unit, International Conference on Computational Intelligence and Communication
Systems, CICN 2011, pp.754-757, Nov. 2011.

8. B.Ramkumar, Harish M Kittur, P.Mahesh Kannan, “ASIC Implementation of Modified Faster Carry Save Adder”, European Journal of Scientific Research ISSN 1450-216X Vol.42 No.1, pp.53-58,2010.