Area and Power Efficient MAC using Variable Latency Carry Skip Adder

DasariSrikara[1], T.Jayanthi[2]
[1]*M.Tech(VLSI), Asst. Professor,*
[12]*Dept of E.C.E, CMR Engineering College*

Abstract- In this paper, we present an area efficient Multiply and Accumulator (MAC) structure using vedic multiplier and various carry skip adders. In this paper first we examine the area and delays of three types of carry skip adder designs. The first design is conventional CSKA using MUX. Second design which employs applying concatenation and incrementation schemes to improve the efficiency of the conventional CSKA (Conv-CSKA) structure. In addition, instead of utilizing multiplexer logic, the CSKA makes use of AND-OR-Invert (AOI) and OR-AND-Invert (OAI) compound gates for the skip logic (CI-CSKA). The third CSKA design which implements variable stage size(VSS) CSKA and with the 8-bit parallel prefix adder (PPA) in order to reduce the delay. These CSKA adders are used in MAC implementations along with the 16x16 vedic multiplier and the results are studied. Verilog HDL is used for designing the circuits. The synthesis and simulation results are obtained using Xilinx ISE 14.7.

Key Words- MAC units, Vedic Multiplier, Carry skip Adders.

I. INTRODUCTION

Due to the rapid growth of portable electronic systems like laptop, calculator, mobile etc, the low power devices have become very important in today‟s world. Low power and high-throughput circuitry design are playing the challenging role for VLSI designer. For real-time signal processing, a high speed and high throughput MAC unit is always a key to achieve a high performance digital signal processing system. The MAC unit performs multiplication and accumulation processes repeatedly in order to perform continuous and complex operations in digital signal processing. MAC unit also contains clock and reset in order to control its operation. Many researchers have been focusing on the design of advance MAC unit architectures. In the adder, the previous MAC output and the present output will added and it consists of Multiplier unit, one adder unit and both will get be combined by an accumulate unit. The major applications of Multiply-Accumulate (MAC) unit are microprocessors, logic units and digital signal processors, since it determines the speed of the overall system . The efficient designs by MAC unit are Nonlinear Computation like Discrete Cosine or wavelet Transform (DCT), FFT/IFFT. Since, they are basically executed by insistent application of multiplication and addition, the entire speed and performance can be compute by the speed of the addition and multiplication taking place in the system.

Generally the delay, critical delay, happens due to the long multiplication process and the propagation delay is observed because of parallel adders in the addition stage. High-speed multipliers are also desired as performance of DSP systems is limited by their performance to execute multiplication processes, which is due to the fact that multiplication dominates the execution time of a majority of DSP algorithms [mac3]. Designing of MAC unit that caters to both delay and power issues has, henceforth, become the need of the hour. Vedic Maths was formulated by Swami Bharati Krishna Tirthaji Maharaja from the ancient Indian scriptures (Vedas) after extensive research on the Vedas. Vedic Mathematics finds its base on the sixteen principles or „sutras‟. Thus, integration of multiplication with Vedic Maths can lead to wonders in various domains of engineering, such as Digital Signal Processing.

II. PRIOR WORK

Since the focus of this paper is on the MAC structure with vedic multiplier and various CSKA designs, first the related work to this MAC are reviewed.

A. Multiplier Unit

A multiplying function can be carried out in three ways: partial product Generation (PPG), partial product addition(PPA), and final conventional addition. The main bottle neck that should be considered in increasing the speed of MAC are partial product reduction. MAC design using conventional multiplier is replaced by vedic multiplier using UrdhavaTriyagbhayam sutra. Multiplication is the fundamental operation of MAC unit [12]. Power consumption, dissipation, area, speed and latency are the major issues in the multiplier unit. So, to avoid them, we go for fast multipliers in various applications of DSP, networking, etc. There are a major criterion that improve the speed of the MAC units which is reducing the partial products and because of that accumulator burden is getting reduced.

The basic operational block in MAC unit is the multiplier that determines the critical path and the delay. The (log2N + 1) partial products are produced by 2N-1 cross products of different widths for N*N. The partial products are generated by Urdhava sutra is by Criss Cross Method. The maximum number of bits in partial products will lead to Critical path.

The MAC unit with area efficiency at high speed processing with reasonable power consumption, for computation of squares is also studied[14]. This new architecture for MAC unit with low area and high speed was also studied. It was achieved by using Vedic Square [14] existing architecture. The principle of Vedic Square is based on the „Duplex D" property of UrdhvaTriyagbhyam which involves addition of twice the product of outermost pair of an n-bit number With odd number of bits, if one bit is left then its square is being taken in the result. Thus, computations decrease with Vedic square reduction in multiplication operations.

B. Addition Block

The existing architecture uses a RCA and the existing architecture, as shown in Fig.1, uses the combination of Vedic multiplier and Sparse Kogge Stone Adder for implementing high speed and low power consumption MAC unit [14].

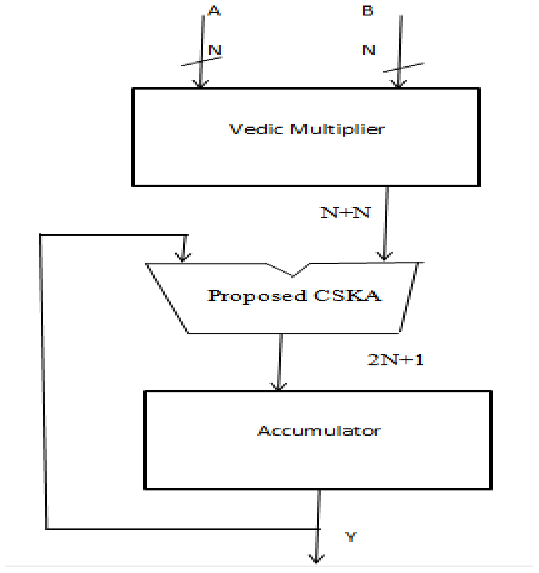


Fig.1: MAC unit

C. Accumulator

Accumulator has an important role in the DSP applications in various ranges and is a very basic and common method. The register designed in the accumulator is used to add the multiplied numbers. Multiplier, adder and an accumulator are forming the essential foundation for the MAC unit. The conventional MAC unit has a multiplier and multiplicand to do the basic multiplication and some parallel adders to add the partial products generated in the previous step. To get the final multiplication output we add the partial product to these results. Vedic Multiplier has put forward to intensify the action of the MAC Unit.

III. PROPOSED MAC STRUCTURE

The design of MAC architecture consists of 3 sub designs.
Design of 16×16 bit Vedic multiplier.
Design of adder using different types of CSKA logic.
Design of accumulator which integrates both multiplier and adder stages.

A. Vedic Multiplier

In this section we propose a Vedic multiplication technique called "Urdhva-Tiryakbhyam – Vertically and crosswise." Which can be used not only for decimal multiplication but also used for binary multiplication. This technique mainly consists of generation of partial products parallel and then we have to perform the addition operation simultaneously[10]. This algorithm can be used for 2x2, 4x4, 8x8....N×N bit multiplications. Since the sums and their partial products are calculated in parallel the Vedic multiplier does not depends upon the processor clock frequency. Hence there is no need of increasing the clock frequency and if the clock frequency increases it will automatically leads to the increase in the power dissipation. Hence by using this Vedic multiplier technique we can reduce the power dissipation. The main advantage of this Vedic multiplier is that it can reduce delay as well as area when compared with the other multipliers. Shift operation is not necessary because the partial product calculation will perform it in a single step, which in turn saves time and power. This is the main advantage of the Vedic multiplier.

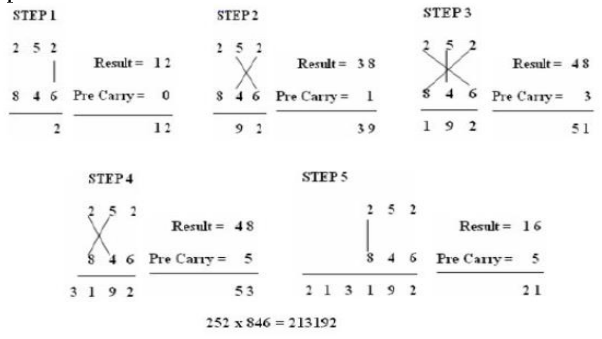


Fig.2: Multiplication of two decimal numbers

A. 2×2 Vedic Multiplier Block

To explain this method let us consider 2 numbers with 2 bits each and the numbers are A and B where A=a0a1 and B=b0b1 as shown in the below line diagram. First the least significant bit (LSB) bit of final product (vertical) is obtained by taking the product of two least significant bit (LSB) bits of A and B is a0b0. Second step is to take the products in a crosswise manner such as the least significant bit (LSB) of the first number A (multiplicand) is multiplied with the next higher bit of the multiplicand B in a crosswise manner. The output generated is 1-Carry bit and 1bit used in the result as shown below. Next step is to take product of 2 most significant bits (MSB) and for the obtained result previously obtained carry should be added. The result obtained is used as the fourth bit of the final result and final carry is the other bit.

$s0 = a0b0$ (1)
$c1s1 = a1b0 + a0b1$ (2)
$c2\ s2 = c1 + a1b1$ (3)
The obtained final result is given as c2s2s1s0. A 2×2 Vedic multiplier block is implemented by using two half adders and four two input and gates as shown in below Figure 8.

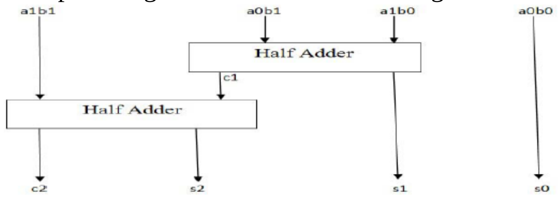


Fig.3: Block Diagram of 2×2 Vedic Multiplier

B. 4x4 Vedic Multiplier Block
In this section, now we will discuss about 4x4 bit Vedic multiplier. For explaining this multiplier let us consider two four bit numbers are A and B such that the individual bits can be represented as the A3A2A1A0 and B3B2B1B0. The procedure for multiplication can be explained in terms of line diagram shown in below figure. The final output can be obtained as the C6S6S5S4S3S2S1S0. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure 3. After performing all the steps the result (Sn) and Carry(Cn) is obtained and in the same way at each step the previous stage carry is forwarded to the next stage and the process goes on.
$S0 = A0B0$ (4)
$C1S1 = A1B0 + A0B1$ (5)
$C2S2 = C1 + A0B2 + A2B0 + A1B1$ (6)
$C3S3 = C2 + A0B3 + A3B0 + A1B2 + A2B1$ (7)
$C4S4 = C3 + A1B3 + A3B1 + A2B2$ (8)
$C5S5 = C4 + A3B2 + A2B3$ (9)
$C6S6 = C5 + A3B3$ (10)
For clear understanding, observe the block diagrams for 4x4 as shown below figure 9 and within the block diagram 4x4 totally there are four 2x2 Vedic multiplier modules, and three ripple carry adders which are of four bit size are used. The four bit ripple carry adders are used for addition of two four bits and likewise totally four are use at intermediate stages 3 of multiplier. The carry generated from the first ripple carry adder is passed on to the next ripple carry adder and there are two zero inputs for second ripple carry adder. The arrangement of the ripple carry adders are shown in below block diagram which can reduces the computational time such that the delay can be decrease.
Hence this is the general mathematical formula applicable to all cases of multiplication and its hardware architecture is shown in fig. 5. In order to multiply two 8-bit numbers using 4-bit multiplier we proceed as follows.Consider two 8 bit numbers denoted as AHAL and BHBL where AH and BH corresponds to the most significant 4 bits, AL and BL are the least significant 4 bits of an 8-bit number.

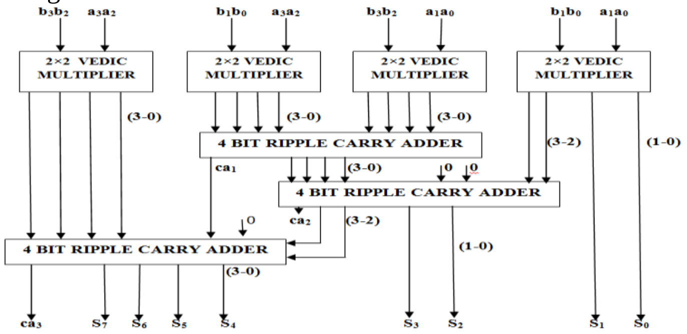


Fig.4: Block Diagram of 4x4 bit Vedic Multiplier

When the numbers are multiplied according to UrdhavaTiryakbhyam (vertically and crosswire) method, we get,
AH AL
BH BL
(AH x BH) + (AH x BL + BH x AL) + (AL x BL).
Thus we need four 4-bit multipliers and two adders to add the partial products and 4-bit intermediate carry generated. Since product of a 4 x 4 multiplier is 8 bits long, in every step the least significant 4 bits correspond to the product and the remaining 4 bits are carried to the next step. This process continues for 3 steps in this case. Similarly, 16 bit multiplier has four 8 x 8 multiplier and three 16 bit adders with 8 bit carry. Therefore we see that the multiplier is highly modular in nature. Hence it leads to regularity and scalability of the multiplier layout. The following fig. 5 shows the design of a 16×16 Vedic multiplier using an 8×8 Vedic multiplier and the design can be implemented using Verilog HDL.

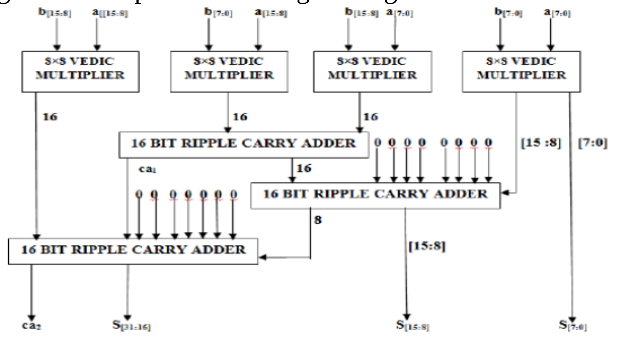


Fig.5: Block diagram of 16x16 vedic Multiplier

IV. Design of Various Carry Skip Adders

1) Designing a CSKA structure using the conventional CSKA (Conv-CSKA) structure.

2) Designing a modified CSKA structure by combining the concatenation and the incrementation schemes to the conventional CSKA (Conv-CSKA).
structure for enhancing the speed and energy efficiency of the adder. Themodification provides us with the ability to use simpler carry skip logics based on the AOI/OAI compound gates instead of the multiplexer.
3) Designing a hybrid variable latency CSKA structure based on the extension of the suggested CSKA, by replacing some of the middle stages in its structure with a PPA, which is modified in [4].

A. Modifying CSKAs for Improving Speed
The conventional structure of the CSKA consists of stages containing chain of full adders (FAs) (RCA block) and 2:1 multiplexer (carry skip logic). The RCA blocks are connected to each other through 2:1 multiplexers, which can be placed into one or more level structures [5]. The CSKA configuration (i.e., the number of the FAs per stage) has a great impact on the speed of this type of adder. Many methods have been suggested for finding the optimum number of the FAs. The techniques presented in [5] make use of VSSs to minimize the delay of adders based on a single level carry skip logic. These techniques, however, cause area and power increase considerably and less regular layout.
Alioto and Palumbo[5] propose a simple strategy for the design of a single-level CSKA. The method is based on the VSS technique where the near-optimal numbers of the FAs are determined based on the skip time (delay of the multiplexer), and the ripple time (the time required by a carry to ripple through a FA). The goal of this method is to decrease the critical path delay by considering a noninteger ratio of the skip time to the ripple time on contrary to most of the previous work. In all of the works reviewed so far, the focus was on the speed, while the power consumption and area usage of the CSKAs were not considered. Even for the speed, the delay of skip logics, which are based on multiplexers and form a large part of the adder critical path delay [5], has not been reduced.

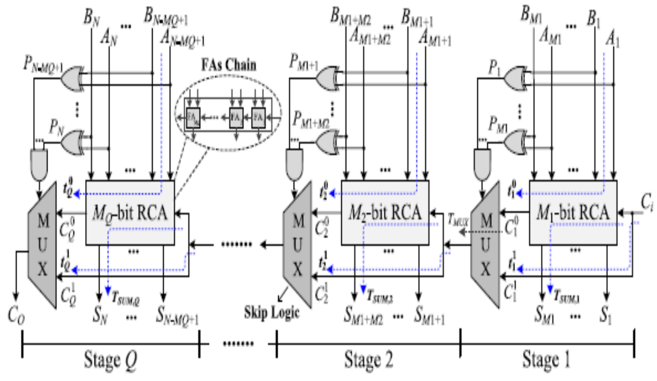


Fig.6: Conventional Carry Skip Adder[4]

B. Structure of modified CI-CSKA
The structure is based on combining the concatenation and the incrementation schemes [13] with the ConvCSKA structure, and hence, is denoted by CI-CSKA. It provides us with the ability to use simpler carry skip logics. The logic replaces 2:1 multiplexers by AOI/OAI compound gates (Fig. 7). The gates, which consist of fewer transistors, have lower delay, area, and smaller power consumption compared with those of the 2:1 multiplexer [37]. Note that, in this structure, as the carry propagates through the skip logics, it becomes complemented. Therefore, at the output of the skip logic of even stages, the complement of the carry is generated. The structure has a considerable lower propagation delay with a slightly smaller area compared with those of the conventional one. Note that while the power consumptions of the AOI (or OAI) gate are smaller than that of the multiplexer, the power consumption of the proposed CI-CSKA is a little more than that of the conventional one. This is due to the increase in the number of the gates, which imposes a higher wiring capacitance (in the noncritical paths).

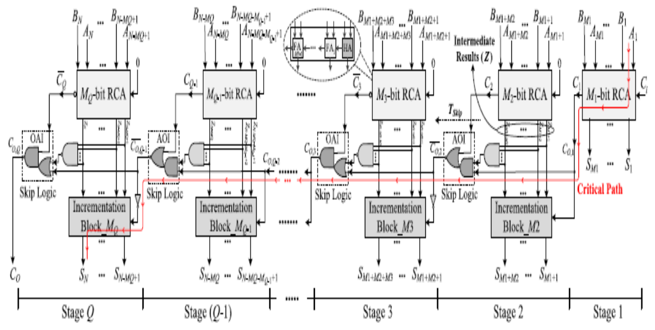
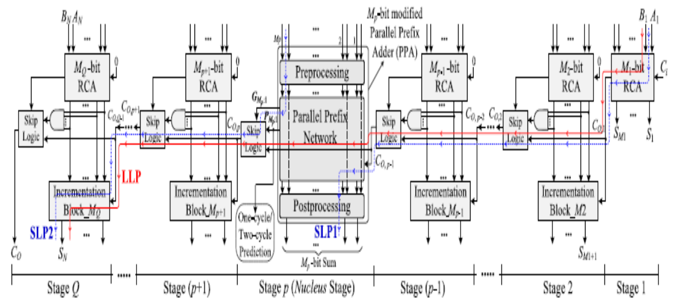


Fig.7: Modified CI-CSKA Structure

The reason for using both AOI and OAI compound gates as the skip logics is the inverting functions of these gates in standard cell libraries. This way the need for an inverter gate, which increases the power consumption and delay, is eliminated. As shown in Fig. 5, if an AOI is used as the skip logic, the next skip logic should use OAI gate. In addition, another point to mention is that the use of the proposed skipping structure in the Conv-CSKA structure increases the delay of the critical path considerably. This originates from the fact that, in the Conv-CSKA, the skip logic (AOI or OAI compound gates) is not able to bypass the zero carry input until the zero carry input propagates from the corresponding RCA block. To solve this problem, in the proposed structure, we have used an RCA block with a carry input of zero (using the concatenation approach). This way, since the RCA block of the stage does not need to wait for the carry output of the previous stage, the output carries of the blocks are calculated in parallel.

C. Hybrid Variable Latency CSKA Structure

The basic idea behind using VSS CSKA structures was based on almost balancing the delays of paths such that the delay of the critical path is minimized compared with that of the FSS structure . This deprives us from having the opportunity of using the slack time for the supply voltage scaling. To provide the variable latency feature for the VSS CSKA structure, we replace some of the middle stages in our proposed structure with a PPA modified in this paper. It should be noted that since the Conv-CSKA structure has a lower speed than that of the proposed one, in this section, we do not consider the conventional structure. The proposed hybrid variable latency CSKA structure is shown in Fig. 8 where an Mp-bit modified PPA is used for the pth stage (nucleus stage). Since the nucleus stage, which has the largest size (and delay) among the stages, is present in both SLP1 and SLP2, replacing it by the PPA reduces the delay of the longest off-critical paths. Thus, the use of the fast PPA helps increasing the available slack time in the variable latency structure. It should be mentioned that since the input bits of the PPA block are used in the predictor block, this block becomes parts of both SLP1 and SLP2.



M_p-bit modified Parallel Prefix Adder (PPA)
M_Q-bit RCA
M_{p+1}-bit RCA
M_{p-1}-bit RCA
M_2-bit RCA
M_1-bit RCA
Skip Logic
Preprocessing
Parallel Prefix Network
Postprocessing
Incrementation Block M_Q
Incrementation Block M_{p+1}
Incrementation Block M_{p-1}
Incrementation Block $M2$
One-cycle/ Two-cycle Prediction
LLP
SLP2
SLP1
M_p-bit Sum
Stage Q
Stage (p+1)
Stage p (Nucleus Stage)
Stage (p-1)
Stage 2
Stage 1

Fig.8: Hybrid variable size CI-CSKA

In the hybrid structure, the prefix network of the Brent–Kung adder is used for constructing the nucleus stage (Fig. 8). One the advantages of the this adder compared with other prefix adders is that in this structure, using forward paths, the longest carry is calculated sooner compared with the intermediate carries, which are computed by backward paths. In addition, the fan-out of adder is lesser than other parallel adders, while the length of its wiring is smaller [14]. Finally, it has a simple and regular layout.

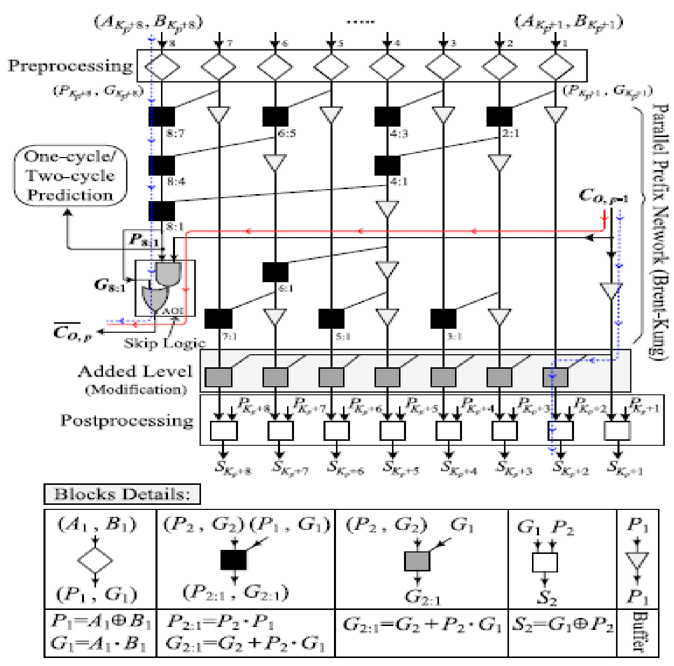


Fig.9: Internal structure of the pth stage of the proposed hybrid variable latency CSKA

As shown in the figure, in the preprocessing level, the propagate signals (Pi) and generate signals (Gi) for the inputs are calculated. In the next level, using Brent–Kung parallel prefix network, the longest carry (i.e., G8:1) of the prefix network along with P8:1, which is the product of the all propagate signals of the inputs, are calculated sooner than other intermediate signals in this network. The signal P8:1 is used in the skip logic to determine if the carry output of the previous stage (i.e., CO,p−1) should be skipped or not. In addition, this signal is exploited as the predictor signal in the variable latency adder.

It should be mentioned that all of these operations are performed in parallel with other stages. In the case, where P8:1 is one, CO,p−1 should skip this stage predicting that some critical paths are activated. On the other hand, when P8:1 is zero, CO,p is equal to the G8:1. In addition, no critical path will be activated in this case. After the parallel prefix network, the intermediate carries, which are functions of CO,p−1 and intermediate signals, are computed (Fig. 9). Finally, in the postprocessing level, the output sums of this stage are calculated. It should be noted that this implementation is based on the similar ideas of the concatenation and incrementation concepts used in the CI-CSKA discussed. It should be noted that the end part of the SPL1 path from CO,p−1 to final summation results of the PPA block and the beginning part of the SPL2 paths from inputs of this block to CO,p belong to the PPA block (Fig. 9). In addition, similar to the proposed CI-CSKA structure, the first point of SPL1 is the first inputbit of the first stage, and the last point of SPL2 is the last bit of the sum output of the incrementation block of the stage Q. Since the PPA structure is more efficient when its size is equal to an integer power of two, we can select a larger size for the nucleus stage accordingly [14]. The larger size (number of bits), compared with that of the nucleus stage in the original CI-CSKA structure, leads to the decrease in the number of stages as well smaller delays for SLP1 and SLP2.

D. Accumulator

The accumulator is designed to store cumulative addition of MAC unit, It is a group of registers which are designed for this. It has a reset pin which is used for resetting. When the reset value is high the content of the accumulator becomes zero and when reset is not equal to zero, the accumulator starts accumulating the summation. The inputs to the accumulator are output from the vedic multiplier and the previous content of the accumulator. Schematics and

V. SIMULATIONS RESULTS

In this section, first we will see the RTL Schematics of the designed MAC unit using vedic multiplier and various CSKA implementation and their simulation results. Then we can compare the delays and area utilized by the proposed structures. The vedic multiplier considered here are the 16x16-bit vedic multiplier and the adder designed here are 32-bit adders. These were designed using Verilog HDL and simulated using Xilinx ISE 14.4. The RTL schematics of the MAC unit and all the sub modules like vedic multiplier and all the CSKA structures are given below.

RTL Schematics

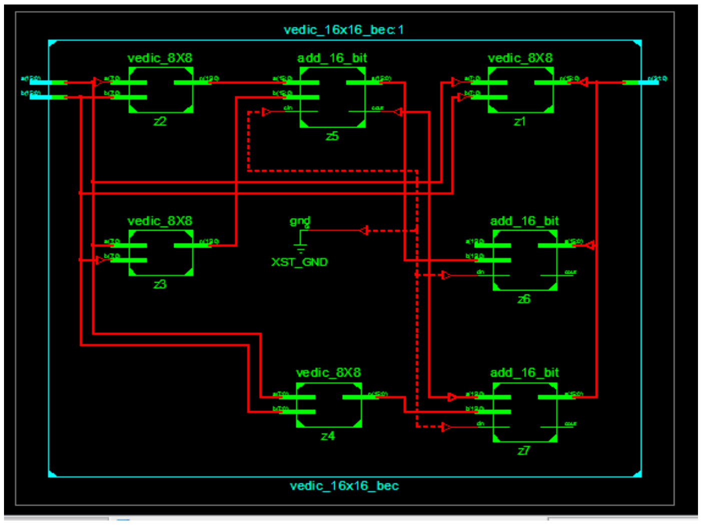


Fig.10: RTL Schematic of Vedic Multiplier

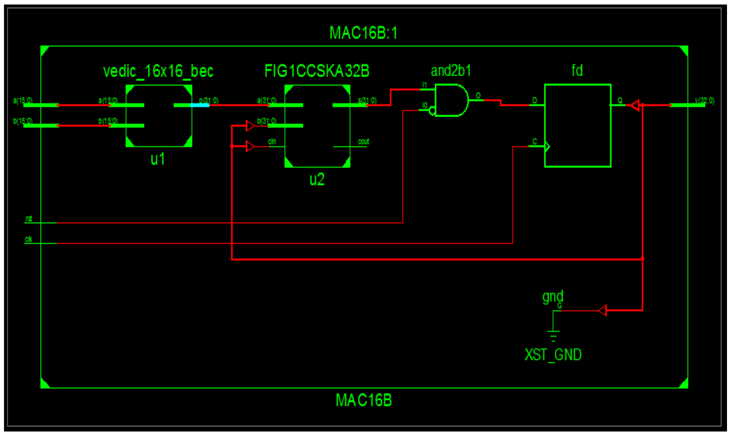


Fig.11: RTL Schematic of MAC with conv-CSKA

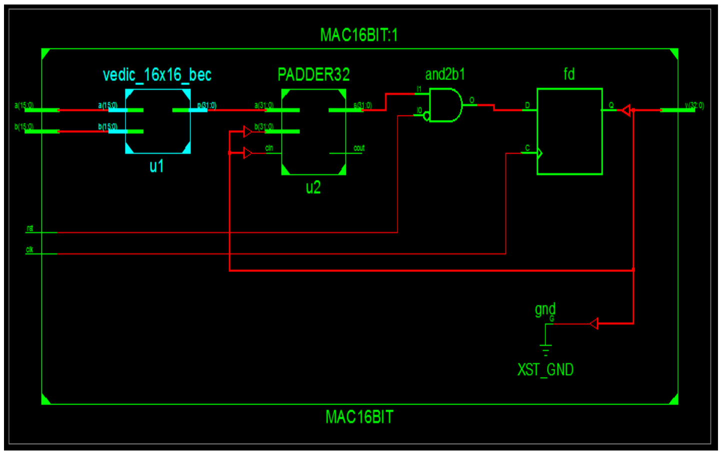


Fig.12: RTL Schematic of MAC with CI-CSKA

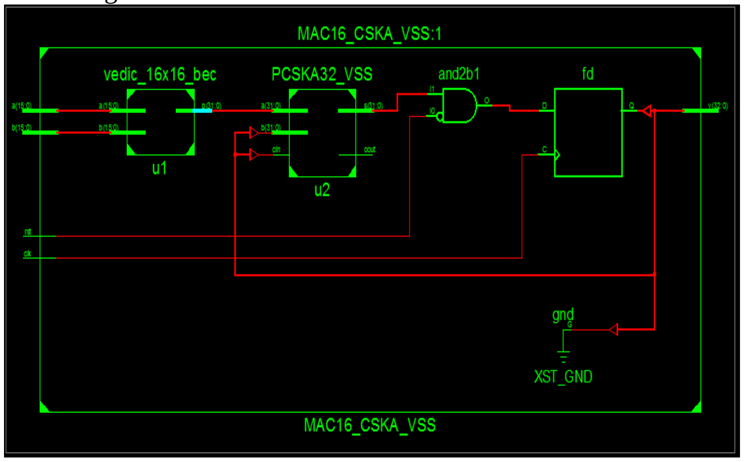


Fig.13: RTL Schematic of MAC with hybrid CSKA

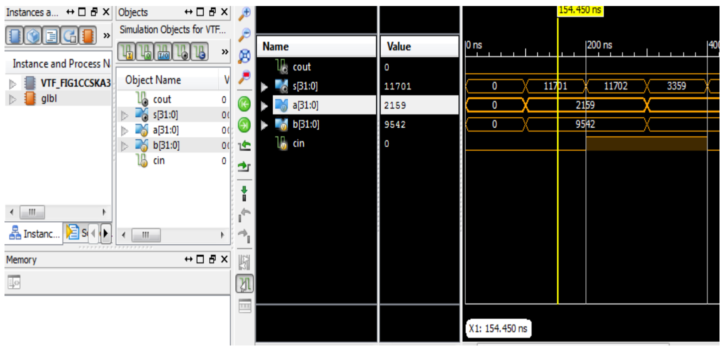


Fig.14: Simulation of Conv-CSKA

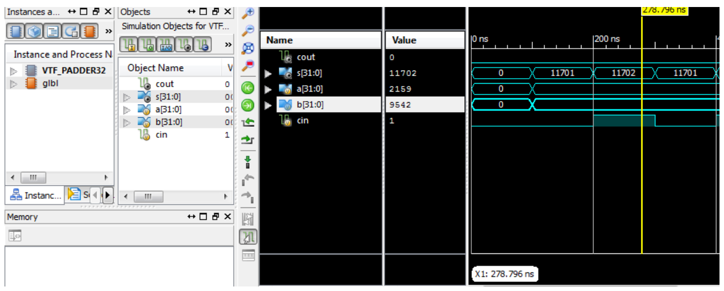


Fig.15: Simulation of CI-CSKA

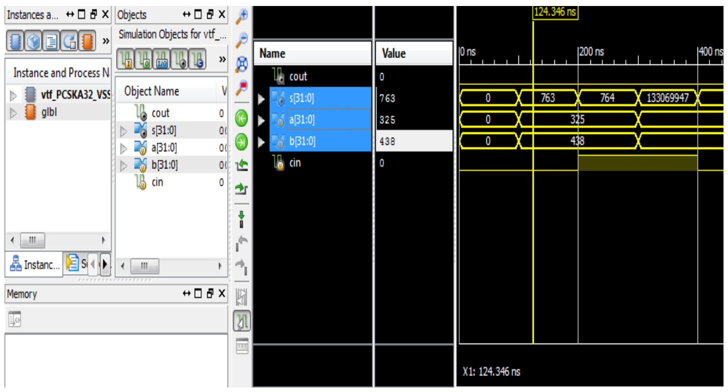


Fig.16: Simulation of CSKA

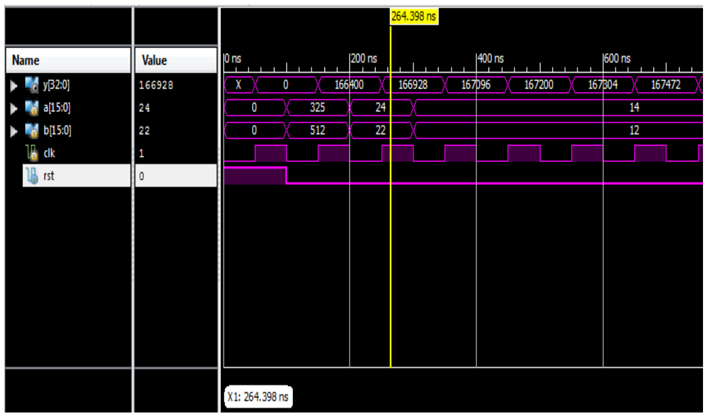


Fig.17: Simulation of MAC unit hybrid CSKA

Delay and Area Comparisons of CSKAs The delays of various CSKAs are compared below. It is noted that the conv- CSKA occupies less no LUTs but delay is more. In the CI-CSKA structure occupies slightly higher number of LUTs but the delay is considerably less. In the hybrid CSKA structure the delay is reduced and numbers of LUTs are also reduced.

Adder Design	No of slices	No of LUT's	Delay(ns)
Conventional [19]	51	88	54.94
Proposed [FSS]	57	101	34.67
Proposed [VSS]	55	98	21.80

VI. CONCLUSIONS

The MAC unit designed with vedic multiplier and different carry skip adders are analyzed and compared for parameters like Area, and Delay Consumption. In terms of delay, MAC with convCSKA and hybrid CSKA have almost same delay. But if you consider the area perspective the hybrid CSKA occupies less number of LUTs.

VII. REFERENCES

[1]. N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson–Addison-Wesley, 2011.
[2]. Koren, Computer Arithmetic Algorithms, 2nd ed. Natick, MA, USA: A K Peters, Ltd., 2002.
[3]. Samir Palinitkar,"Verilog HDL: A Guide to Digital Design and Synthesis".
[4]. MiladBahadori, Mehdi Kamal, Ali AfzaliKusha, and MassoudPedram,"Hi-speed and energy efficient carry skip adders operating under a wide range of supply voltage levels", IEEE Trans,2015.
[5]. M. Alioto and G. Palumbo, "A simple strategy for optimized design of one-level carry-skip adders," IEEE Trans. Circuits Syst. I, Fundam.Theory Appl., vol. 50, no. 1, pp. 141– 148, Jan. 2003.
[6]. David H.K.Hoe, Chris Martinez and Sri JyothsnaVundavalli", Design and Characterization of Parallel Prefix Adders using FPGAs", 2011 IEEE 43rd South eastern Symposium in pp. 168-172, 2011.
[7]. P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," IEEE Trans.Comput., vol. C-22, no. 8, pp. 786–793, Aug. 1973.
[8]. V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energydelay estimation technique for high performance microprocessor VLSI adders," in Proc. 16th IEEE Symp.Comput. Arithmetic, Jun. 2003, pp. 272–279.
[9]. Sudheer Kumar Yezerla, B RajendraNaik,Design and Estimation of Delay, Area and Power for Parallel Prefix Adders,

Proceedings of 2014 RAECS UIET Punjab University Chandigarh, 06 - 08 March, 2014.

[10]. BhavaniPrasad.Y, Ganesh Chokkakula, SrikanthReddy.P and Samhitha.N.R, Design of Low Power and High Speed Modified Carry Select Adder for 16 bit Vedic Multiplier, ISBN No.978-1-4799-3834- 6/14/$31.00©2014.

[11]. Premananda B.S. , Samarth S. Pai, Shashank B, Shashank S. Bhat, ”Design and Implementation of 8-Bit Vedic Multiplier” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 12, December 2013.

[12]. AvisekSen, ParthaMitra, DebarshiDatta, “Low Power MAC Unit for DSP Processor”, International Journal of Recent Technology and Engineering (IJRTE) Vol. 1, Issue-6, January 2013, pp. 93-95.

[13]. MarojuSai Kumar, D.Ashok Kumar, Dr. P. Samundiswary, ”Design and performance of analysis of Multiply and Accumulate Unit”,ICCPCT 2014.

[14]. Gitika Bhatia, Karanbir Singh Bhatia, OsheenChauhan, SoumyaChourasia, Pradeep Kuma5,”A efficient MAC unit with low area consumption” IEEE INDICON ,2015.

[15]. M.GaneshKumar ,I.V.Rameswar Reddy , K.Kameswara Reddy,” Design of Two Variable Multiplier Using Vedic Mathematics and ROM Approach”, IJERT 2013.