ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Design, Analysis and FPGA Implementation of N Bit Vedic Multiplier Based on Different Adder Architectures

S.Tamilselvan[1], V. Anil Kumar[2], V. Kamalkannan[3], CH.V.M.S.N.Pavan Kumar[4]

Assistant Professor, Department of ECE, Pondicherry Engg. College, Pondicherry, India [1]

Assistant Professor, Department of ECE, Tirumala Engg. College, Jonnalagadda, India [2]

Research Scholar, Department of ECE, Pondicherry Engg. College, Pondicherry, India [3,4]

Abstract: The speed of the multipliers depends on the speed of the adders which are used for addition of partial products. The papers main focus is on the time delay of the multiplication operation on multipliers based on the ancient Vedic mathematical Sutra called Urdhva Tiryakbhyam i.e. vertically and cross wise Sutra. This Vedic multiplier is implemented using adder which has lesser time delay among carry look ahead adder, ripple carry adder, carry skip adder and carry select adder. The Vedic multiplier is coded in Verilog HDL and simulated using Xilinx ISE 14.3 software. This multiplier is implemented on Spartan 6 FPGA devices. The Vedic multiplier is compared in terms of time delay with conventional multiplier and it is used in Fast Fourier Transform algorithm.

Index Terms: Ripple Carry Adder (RCA), Carry Look Ahead Adder (CLA), Carry Select Adder (CSA), Urdhva Tiryakbhyam, Fast Fourier Transform.

I. INTRODUCTION

Multipliers play a very important in today's computers, image processing and various applications. Many scientists and researchers has tried and are trying to design a multiplier which will offer the following design targets – high speed, less area, low power consumption and thus makes it suitable for various high speed, low power VLSI implementations. The multiplication commonly used is based on "shift and add" method [1]. In this multiplier, the number of partial products to be added is the main parameter that determines the performance of the multiplier. The number of partial products to be added is reduced using Modified Booth algorithm which is one of the most popular algorithms [2-5]. To improve speed, Wallace Tree algorithm can be used to reduce the number of sequential adding stages. Combination of both Modified Booth algorithm and Wallace Tree technique can provide advantage of both algorithms in one multiplier. But with increasing parallelism, the number of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed and increase in silicon area and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand "serial-parallel" multipliers compromise speed to achieve better performance for area and power consumption [18-19]. The selection of parallel or serial multiplier is actually depends on the nature of application. In this paper we introduce the multiplier using Vedic algorithm and its architecture and compare them in terms of time delay [6]. In microprocessor, DSP etc., addition and multiplication of two binary digits is the basic and most commonly used arithmetic operations. Statics shows that more than 70% instructions in microprocessor and most of DSP algorithms perform addition and multiplication [7]. So, addition and multiplication operations dominate the execution time. That's why; there is need of high speed multiplier [14-15]. The high speed processor demand has been increasing as a result of increasing computer and signal processing applications [16]. The consumption of power is also an important issue in multiplier design. In order to reduce power consumption, it is good to reduce the number of operation thereby reducing dynamic power which is a major part of total power consumption so the need of high speed and low power multiplier has increased [6]. Designer mainly concentrates on high speed and low power efficient circuit design. A good multiplier should be compactly packed and provide high speed and low power consumption unit [8].

II.URDHVA TIRYAKBHYAM SUTRA

The Vedic multiplier is based on the Vedic multiplication formulae called Sutra. This Urdhva Tiryakbhyam is for the multiplication of two digits [5]. In this paper, we will apply the same ideas to make the proposed multiplier compatible with the digital hardware.

Urdhva Tiryakbhyam Sutra is a general multiplication sutra applicable to all cases of multiplication. The meaning of Urdhva Tiryakbhyam is "Vertically and crosswise". The end digits of two lines are multiplied and the output is summed with the previous carry [6]. Initially the carry-in is taken to be as zero. When there are more lines in single step, all the outputs are summed to the previous carry. LSB of the number will be one of the final output digits and the rest wills act as the carry in for next step [20].

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

The multiplication of two 4-bit numbers is as shown in Fig. 1. Consider an example, which describes the multiplication of two decimal numbers 123x456. In First step, we take the product of LSBs of the multiplier and multiplicand, the LSB of the result i.e. 8 in this case, is stored and carry is generated for the next step i.e. 1. In step two, the two LSBs are multiplied crosswise, and their product is added with the previous carry.

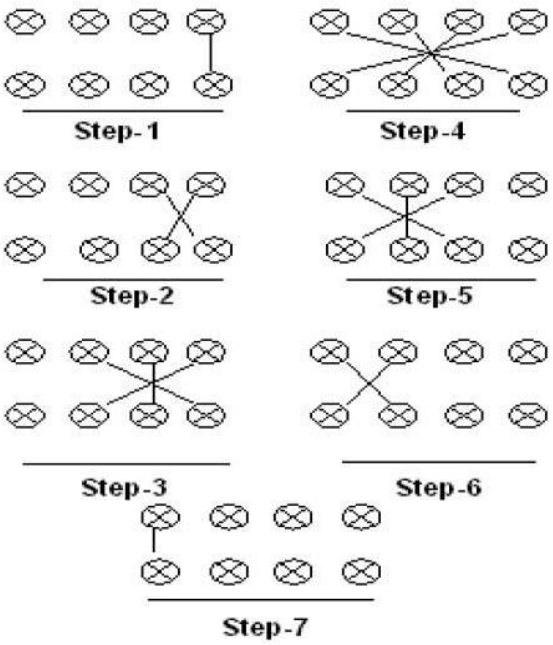

Fig. 1.Line diagram of Urdhva Tiryakbhyam

Likewise in the next step all the bits are multiplied crosswise and their products are summed up with the previous carry. Again In the next step, two MSBs are crosswise multiplied, and their results are added in the similar manner. And finally, the MSBs of the multiplier and multiplicand are multiplied, and the output is added with the previously generated carry to get the end result. We can apply this sutra to multiply binary numbers. As shown in Fig. 1 the bits of multiplier and multiplicand are crosswise multiplied and their result is added with previous carry to get the result of that step [10]. Then the final product is obtained by concatenating the results from each step and the carry in the last step. In Fig.1 bits are represented by circles [11].

A 2x2 Vedic Multiplier Architecture is obtained by using 2 half adders and four AND gates [9], as shown in Fig. 2. The product a0b0 is directly given to the output, a1b0 and a0Yb1 are added using first half adder and the half adder sum output is directly given to the output and the carry is added with product a1b1 using second half adder.

S0=a0xb0,
S1=a0xb1+a1xb0,
S2=a1xb1+C1,
Result= {C2S2S1S0}.

In this Vedic multiplier, different adders have been used and their performance has been analyzed based on time delay [6].

III. DIFFERENT ADDER ARCHITECTUERS

The different adders used for analysis and comparison in our project are

- Ripple Carry adder
- Carry Look ahead adder
- Carry Skip adder
- Carry Select adder

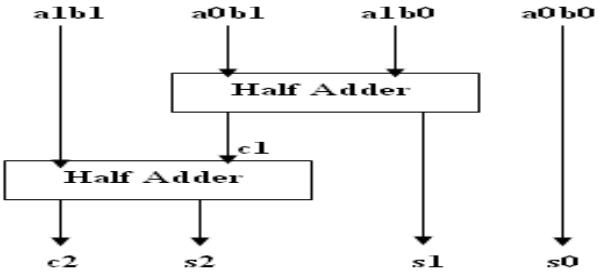

Fig.2 Architecture of a 2x2 multiplier

A. Ripple carry adder

Input: Binary stream A, Binary stream B, Carry in (C0)
Output: SUM (Sn), Carry OUT (Cout)

N-bit Ripple Carry Adder has N Full Adders cascaded serially where the carry ripples along the n-1 stages to generate the final Carry Out. Each full Adder (FA) generates the sum (Si) and carry (Ci) from the input bit stream and the sum output of each stage is calculated by Si= (Ai^Bi)^Ci-1 and the carry by
Ci= (Ai & Bi) + (Ai+Bi)Ci-1.

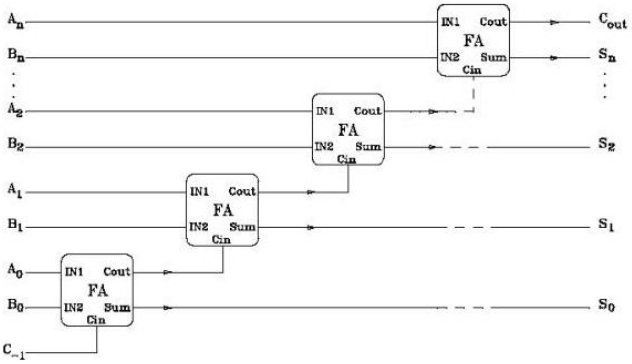

Fig.3 N-bit Ripple Carry Adder

Fig. 3 shows the block description of n-bit Ripple Carry Adder. Ripple-carry adders are the simplest and most compact adders but its performance is limited by the carry which must ripple from the LSB to MSB.

N full adder (FA) circuits are parallerley cascaded to add an N-bit number. N full adder circuits are required for N-bit addition. A ripple carry adder (RCA) is an adder circuit in which the full adders carry out is given as carry in of the succeeding full adder. It is so called ripple carry adder because each carry bit generated in a stage gets rippled into the next stage of full adder. In a ripple carry adder, the sum and carry out of a half adder circuit is not valid before the carry in of that half adder circuit occurs. The Propagation delays inside the RCA circuitry are the reason behind this which is the time delay between the

DOI 10.17148/IJARCCE.2015.4898

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

application of an input and occurrence of the corresponding output. In a NOT gate, when the input is "0", the output will be "1" and vice- versa. Propagation delay is the reason for the delay in NOT gate to get the output as "1" after the application of logic "0" to the input. Thus the propagation delay of carry is the time delay between the application of the carry in (Cin) signal and the occurrence of the carry out (Cout) signal.

B. Carry Look Ahead Adder
The below figure describes the block diagram of the CLA in two modules

A. Describes the Partial Full Adder
B. Describes the Carry Look ahead Logic.

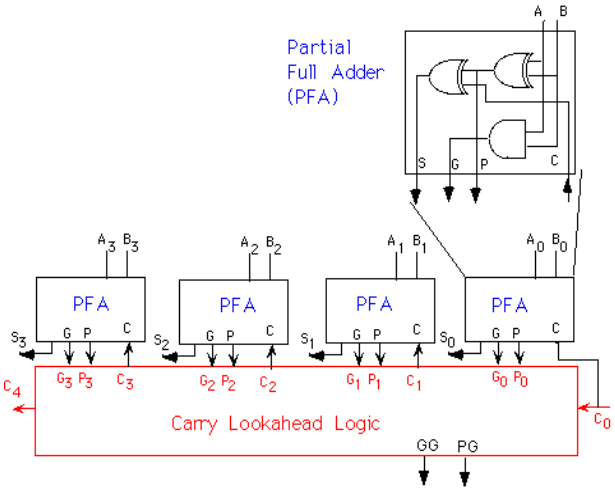

Fig. 4.4-bit Carry Look Ahead Adder

The Carry Look Ahead Adder (CLA) is the most widely used design for high-speed adders in modern Computers. The boon of using a carry look-ahead design over a ripple carry adder is that the Look-ahead is faster in computing the carry out. The carry-in values of each stage in a carry look-ahead design are calculated independent of each other through a series of logic circuits.
Process:

The equations of propagate and generate terms are

Gi= ai+ bi and Pi = ai^ bi

$C_1=G_0+P_0.C_0$ (1)

$C_2=G_1+P_1.C_1=G_1+P1.G_0+P_1.P_0.C_0$ (2)

$C_3=G_2+P_2.G_1+P_2.P_1.G_0+P_2.P_1.P_0.C_0$ (3)

$C_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3P_2.P_1.G_0 + P_3P_2.P_1.P_0.C_0$ (4)

The time delay difference between a ripple and a carry look-ahead adder increases as the size of the input increases because look-ahead generates carry out based on the initial carry in.
It is observed that the Carry-in's are independent of each other instead of rippling together. They are instead predicted by another logic device from A and B's inputs. The operational logic for this process is somewhat complicated and more involved than the ripple carry adder.

C. Carry select adder
Carry select Adder (CSA) is a better adder especially in the case of delay of carry. In a ripple-carry adder, every full adder stage has to wait for the incoming carry before a carry out of that stage can be generated. This limitation can be overcome by calculating sum and carry out for possible values of the carry input (0,1) in advance. Once the correct value of the incoming carry is known, the correct carry out result is easily selected with a simple multiplexer stage. The idea leads to the implementation of an adder called the linear carry select adder and the block diagram of the first four bit linear carry select adder is shown below.

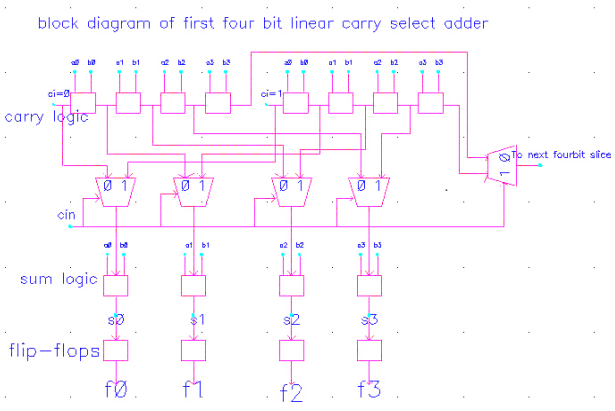

Fig. 5.Carry Select block diagram

In the above diagram, all the inputs are given at a time to both carry in Ci =0 and the Ci =1 carry logic. The carry circuits generate the appropriate carry-out and depending upon the original carry input the correct carry out and sum are selected from the multiplexer. This is the Basic 4 bit carry select adder. Thus for implementing the Higher order bit the carry out from the Multiplexer passes as the carry in for the next 4-bit, while the inputs are given at the same time. It is clear that the time delay is reduced to a large extent by performing the carry calculations in advance, but the disadvantage is that the hardware over head of the carry select adder is depends on the additional carry path and a multiplexer.

D. Carry skip adder
The carry skip adder provides a compromise between a ripple carry adder (RSA) and a CLA. The carry skip adder divides the words to be added into blocks. In each block, ripple carry is used to produce the sum bit and the carry. The Carry Skip Adder reduces the time delay due to the computation of carry i.e. by skipping over groups of consecutive adder stages. If Ai is not equal to Bi in a group, then there is need to compute the new value of Ci+1 for that block and the carry-in of the block can be propagated directly to the next block. If Ai = Bi = 1 for some i in the group, a new carry is generated which may be propagated up to the input of next group. If Ai = Bi = 0, a new carry will be generated and fed as input to next group. The basic idea behind the carry-skip adder is to detect if in each group all Ai is not equal to Bi, then enable

DOI 10.17148/IJARCCE.2015.4898

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

the block's carry-in to skip the block [12-13]. In general a block-skip delay can be different from the delay due to the propagation of a carry to the next bit position. In a carry skip adder, the growth of carry chain delay with the input digit size is reduced by making carry to skip the blocks of bits than rippling through them.

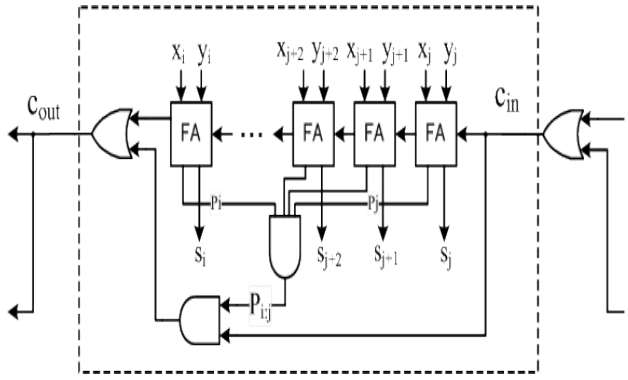

Fig.6 Carry skip adder

A carry-skip adder consists of ripple carry-adders (RCA) with a special speed up mechanism called a skip mechanism. Carry skip adder is a faster when compared to ripple carry adder. In an addition of large number of bits (n), carry skip adder has O (√n) delay which gives a better performance in terms of delay and layout of the circuit. Carry skip adder is based on carry skip mechanism which requires ripple carry blocks. A Carry skip adder is designed to increase the speed of addition by doing the skipping of a carry bit over a portion of the adder. Moreover the ripple carry adder (RCA) is faster for lesser number of bits. The industries of these days' uses desktop computers which use word lengths of 32 bits which in turn makes the carry skip structure more important. The crossover point between the carry skip adder and ripple-carry adder (RCA) is dependent on technology considerations. The carry-skip mechanism requires logic gate AND which accepts the propagate input of each full adder and if the output of the AND gate is 1 then the carry input is skipped and given as input to next block. The block diagram of carry skip adder is shown in figure 6.

IV. FAST FOURIER TRANSFORM

A Fast Fourier Transform (FFT) is an effective algorithm to calculate the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT). The FFT is based on method of decomposition of transform into smaller transforms and combining the smaller transforms to get the full transform. Fast Fourier Transform (FFT) efficiently performs Discrete Fourier transform and the performance is increased by a factor of 100 over direct computation of the DFT [17].

The algorithm of Fast Fourier Transform provides the two basic properties such as twiddle factor - the symmetry property and periodicity property**.** This properties reduces the number of complex multiplications involved to compute DFT. Calculating a N point DFT in the direct way requires O(N^2) operations, while an Fast Fourier Transform requires only O(N log N) operations to compute the same result [7].

Fast Fourier Transform (FFT) algorithms are based on the principle of decomposing a sequence of length N discrete Fourier Transform into successively smaller discrete Fourier transforms.

There are basically two classes of FFT algorithms.
Decimation In Time (DIT) algorithm
Decimation In Frequency (DIF) algorithm.

A. Decimation In Time (DIT) Algorithm

The Radix-2 Decimation In Time (DIT) algorithm divides the Discrete Fourier Transform of the function x(n) of length N into two terms: a sum over the indices of even-numbered n = 2m and a sum over indices of odd-numbered n = 2m + 1.

$$\mathbf{X_k} = \sum_{m=0}^{\frac{N}{2}-1} \mathbf{x_{2m}} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{\frac{N}{2}-1} \mathbf{x_{2m+1}} e^{-\frac{2\pi i}{N}(2m+1)k} \quad (5)$$

Since these smaller DFTs have a length of N/2, only N/2 outputs are need to be computed, DFT outputs for N/2 < k < N of length N/2 are identical to the DFT outputs for 0< k < N/2.So, Ek + N / 2 = Ek and Ok + N / 2 = Ok. The phase factor exp [− 2πik / N] is known as twiddle factor which obeys the relation: exp[− 2πi(k + N / 2) / N] = $e^{-\pi i}$exp[− 2πik / N] = − exp[− 2πik / N], turning over the sign of the Ok +N / 2 terms.

$$\mathbf{X_k} = \begin{cases} \mathbf{E_k} + e^{-\frac{2\pi i}{N}}\mathbf{O_k} & \text{if } k < N/2 \\ \mathbf{E_{k-N/2}} - e^{-\frac{2\pi i}{N}(k-N/2)}\mathbf{O_{k-N/2}} & \text{if } k \geq N/2 \end{cases} \quad (6)$$

If N/2 is even, we can further split the computation of each DFT of size N/2 into two computations of half size DFT. When $N=2^r$ this can be done until DFT of size 2 (i.e. butterfly diagram of two elements).

The Vedic multiplier is used in the FFT algorithm wherever multiplication is involved. According to the algorithm, in each stage the twiddle factor is multiplied with corresponding input and then added or subtracted to corresponding term to get output of that stage.

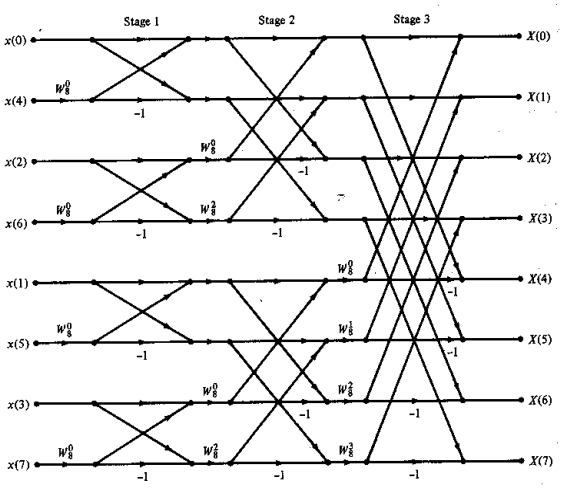

Fig. 7 8 point Radix 2 FFT general diagram

DOI 10.17148/IJARCCE.2015.4898

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

The multiplier used in the twiddle Factor multiplication is Vedic multiplier. In the addition part of the algorithm, carry select adder is used. In 8 point FFT 8, 16 and 32 bit carry select adders are used in the addition process of the algorithm and 8 and 16 bit multipliers are used in the multiplication process of the algorithm.

V. RESULTS AND COMPARISIONS

A. Simulation results of 4-bit adders

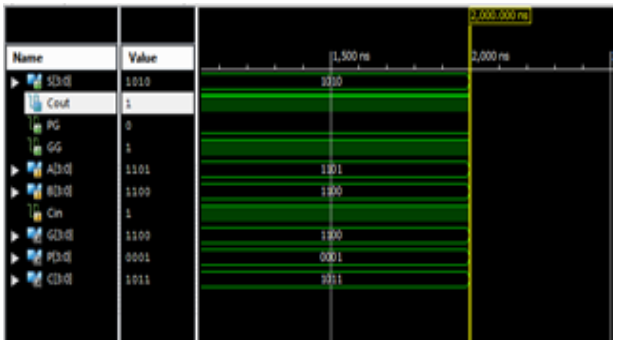

Fig. 8 Simulation of 4-bit carry look ahead adder

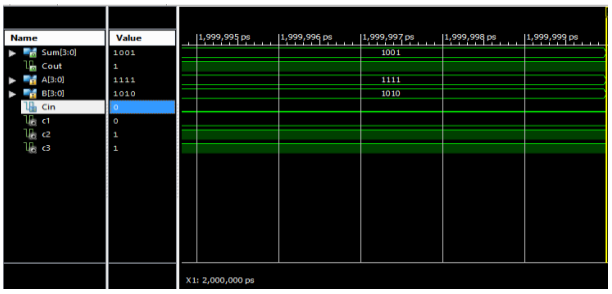

Fig.9. Simulation of 4-bit ripple carry adder

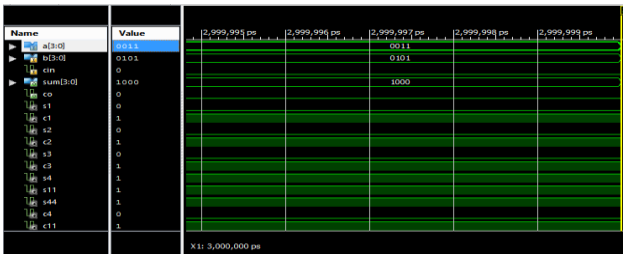

Fig. 10.Simulation of 4-bit carry select adder

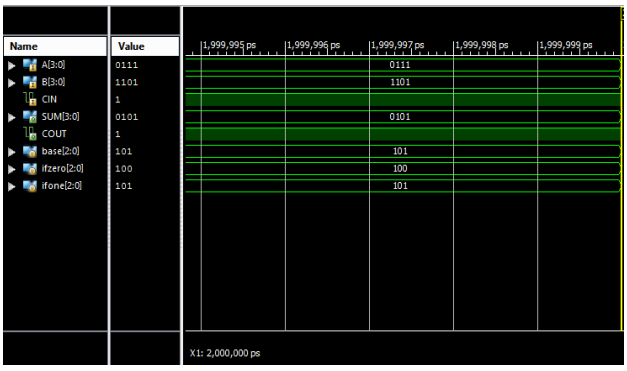

Fig.11 Simulation of 4-bit carry skip adder

Table I Comparison of 4 Bit Adders Based On Time Delay

Adders	Number of slice LUTs	Time delay(ns)
CLA	10	6.494
Ripple carry	6	7.682
Carry select	6	6.494
Carry skip	8	6.494

The time delay of carry select adder is lesser than that of the other adders.

B. Simulation Results of Vedic Multiplier Using 8, 16 Bit Carry Select Adder

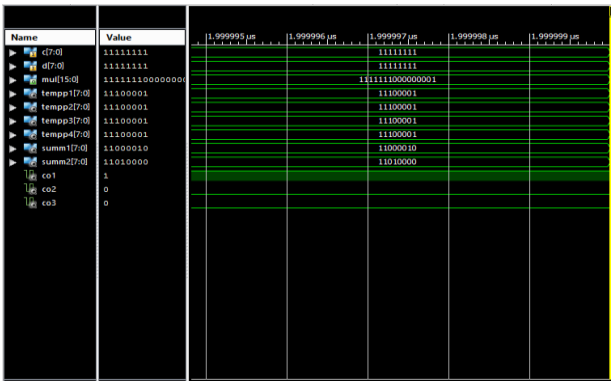

Fig.12 Vedic multiplier using 8-bit carry select adder

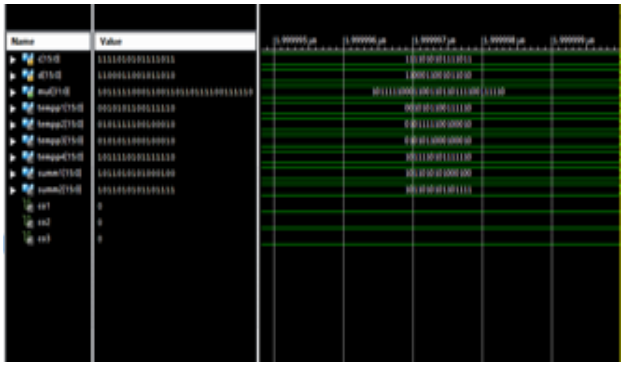

Fig.13 Vedic multiplier using 16-bit carry select adder

Table II Summary of Vedic Multiplier Using Carry Select Adder

Bit	Number of slice LUTs	Time delay(ns)
4bit	25	10.386
8bit	127	15.916
16bit	577	24.047

Since carry select adder had lesser time delay it was used in the Vedic multiplier for addition process and the above results was obtained.

C. Simulation results of Fast Fourier Transform using Vedic multiplier

Technology schematic

DOI 10.17148/IJARCCE.2015.4898

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

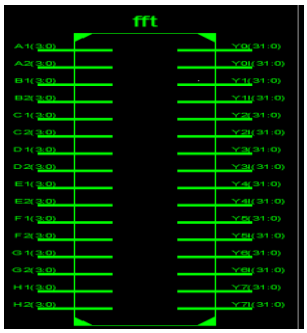

Fig.14 Technology schematic of FFT

Simulation Results

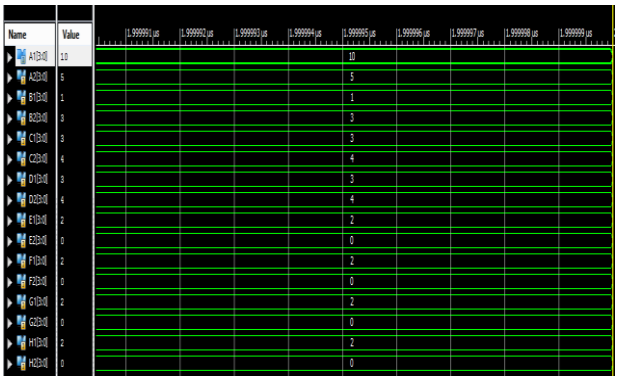

Fig.15 Input of 8 point FFT

1st stage output

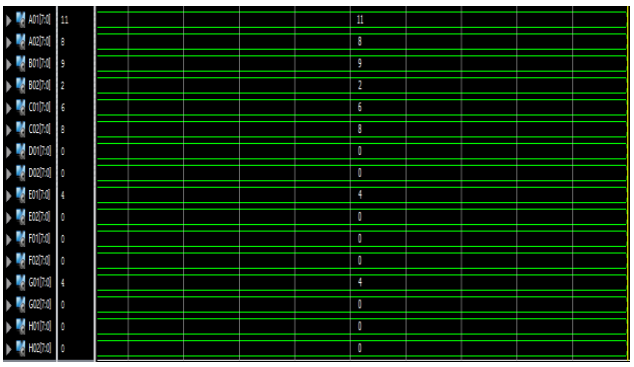

Fig.16 1st stage output of 8 point FFT

2nd stage output

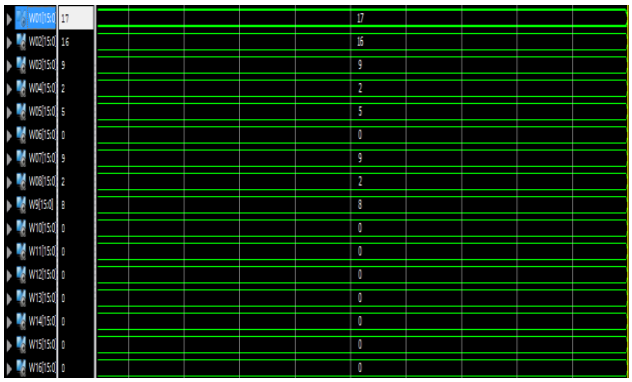

Fig.17 2nd stage output of 8 point FFT

Final stage output

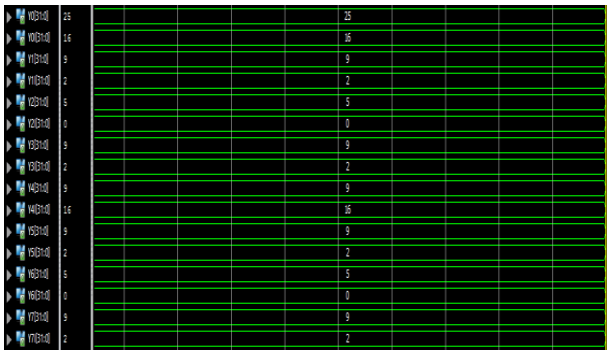

Fig.18 Final stage output of 8 point FFT

Table III Comparison of FFT Using Vedic and Conventional Multiplier

FFT using	Time delay(ns)
Vedic multiplier	23.47
conventional multiplier	41.055

FFT using Vedic multiplier has better time delay than that of FFT using conventional multiplier.

D. Implementation of Proposed Vedic Multiplier in Spartan 6 FPGA Board

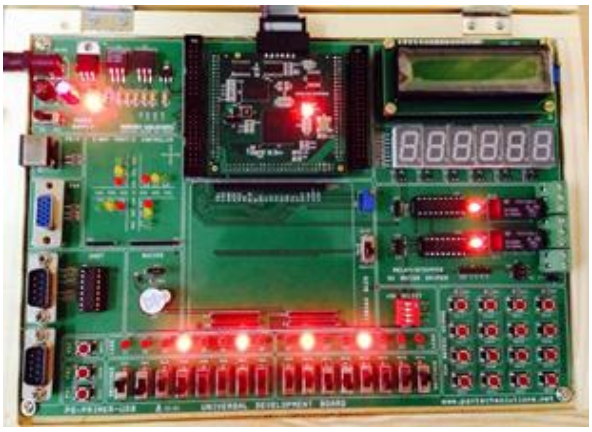

Fig.19. SPARTAN 6 FPGA BOARD

VI. CONCLUSION

A Conclusion

Carry Select Adder (CSA) had lesser time delay when compared to Ripple carry Adder, Carry look ahead adder and Carry skip ahead adder architectures from our analysis. The Vedic multiplier was designed using carry select adder. The proposed Vedic multiplier was used in the Fast Fourier Transform algorithm and its time delay was better compared to that of the Fast Fourier transform using conventional multiplier. The proposed Vedic multiplier was implemented in the Spartan 6 FPGA board.

B. Future work

Digital technology which are now in almost every engineering field. In Digital Signal Processing Faster

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

computation of additions and multiplications are required for faster computation of convolution, DFT etc. The basic computing process in any digital device is always a multiplication process. Therefore, Digital Signal Processing engineers are constantly looking for new efficient algorithms and hardware to implement them. The increasing demand on Digital Signal Processing application makes the researchers and engineers to pay attention on low cost, high speed, single chip implementation of DSP algorithm. Fast Fourier Transform (FFT) is the one of the basic algorithm that is primarily performed in any Digital Signal Processing system. The Fast Fourier Transform is widely used in digital signal processing (DSP) applications such as image processing, wireless communication, instrumentation and inspection of machines. The Vedic mathematics methods are complementary, direct and easy. Employing Vedic Mathematic sutra in the computation algorithms of the processor will decrease the execution time, area, power dissipated etc. Urdhva Tiryakbhyam, being a general multiplication sutra, is generally applicable to all types of multiplication. Another Vedic mathematics sutra called Nikhilam algorithm has the compatibility to different data types. This Nikhilam sutra can be used to build a high speed power efficient reconfigurable FFT.

REFERENCES

[1] Gaurav Sharma, Arjun Singh Chauhan, Himanshu Joshi and Satish Kumar Alaria, "Delay Comparison of 4 by 4 Vedic Multiplier based on Different Adder Architectures using VHDL", International Journal of IT, Engineering and Applied Sciences Research (IJIEASR), ISSN:2319-4413, Volume 2, No.6, pp: 28-32, June 2013.

[2] Bathija, R.K., Meena, R.S., Sarkar, S., Sahu, Rajesh, "Low Power High speed 16X16 bit Multiplier using Vedic Mathematics", International Journalof Computer Applications(IJCA), Vol. 59, Number 6, December 2012.

[3] 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, ISSN: 2320 – 3765 Vol. 2, Issue 12, December 2013.

[4] Anju and V.K. Agrawal, "FPGA Implementation of Low Power and High Speed Vedic Multiplier using Vedic Mathematics", IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), ISSN: 2319 – 4200, Volume 2, Issue No.5, pp: 51-57, June 2013.

[5] R. Sridevi, AnirudhPalakurthi, AkhilaSadhula, HafsaMahreen, "Design of a High Speed Multiplier (Ancient Vedic Mathematics Approach)", International Journal of Engineering Research, ISSN: 2319-6890 Volume No.2, Issue No.3, pp: 183-186, July 2013.

[6] Basant Kumar Mohanty and Sujit Kumar Patel, "Area–Delay–Power Efficient Carry-Select Adder", IEEE Transaction on Circuits and Systems—II: Express Briefs, VOL. 61, NO. 6, June 2014.

[7] Lakshmi Santhosh and Anoop Thomas, "Implementa- tion of Radix 2 and Radix 2^2 FFT Algorithms on Spartan6 FPGA", Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), IEEE, 6th July 2013.

[8] Wallace, C.S., "A suggestion for a fast multiplier," IEEE Trans. Elec. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[9] Booth, A.D., "A signed binary multiplication technique," Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236– 240, 1951.

[10] Jagadguru Swami Sri Bharath, Krsna Tirathji, "Vedic Mathematics or Sixteen Simple Sutras from the Vedas", Motilal Banarsidas, Varanasi (India), 1986.

[11] A.P. Nicholas, K.R Williams, J. Pickles, "Application of Urdhva Sutra", Spiritual Study Group, Roorkee (India), 1984.

[12] Neil H.E Weste, David Harris, Ayan anerjee,"CMOS VLSI Design, A Circuits and Systems Perspective", Third Edition, Published by Person Education, PP-327-328]

[13] Mrs. M. Ramalatha, Prof. D. Sridharan, "VLSI Based High Speed Karatsuba Multiplier for Cryptographic Applications Using Vedic Mathematics", IJSCI, 2007.

[14] Thapliyal H. and Srinivas M.B. "High Speed Efficient N x N Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics", Transactions on Engineering, Computing and Technology, 2004, Vol.2.

[15] "A Reduced-Bit Multiplication Algorithm for Digital Arithmetic" Harpreet Singh Dhilon and Abhijit Mitra, International Journal of Computational and Mathematical Sciences, Waset, Spring, 2008.

[16] D. Zuras, On squaring and multiplying large integers, In Proceedings of International Symposium on Computer Arithmetic, IEEE Computer Society Press, pp. 260-271, 1993.

[17] Shripad Kulkarni, "Discrete Fourier Transform (DFT) by using Vedic Mathematics"Papers on implementation of DSP algorithms/VLSI structures using Vedic Mathematics, 2006, www.edaindia.com, IC Design portal.

[18] M.C. Hanumantharaju, H. Jayalaxmi, R.K. Renuka, M. Ravishankar, "A High Speed Block Convolution Using Ancient Indian Vedic Mathematics", vol. 2, pp.169-173, International Conference on Computational Intelligence and Multimedia Applications, 2007.

[19] Himanshu Thapliyal, "Vedic Mathematics for Faster Mental Calculations and High Speed VLSI Arithmetic", Invited talk at IEEE Computer Society Student Chapter, University of South Florida, Tampa, FL, Nov 14 2008.

[20] Jeganathan Sriskandarajah, "Secrets of Ancient Maths: Vedic Mathematics", Journal of Indic Studies Foundation, California, pages 15 and 16.