VLSI Implementation of Split-radix FFT for High Speed Applications

Arunkumar P. Chavan
Department of Electronics and Communication
R. V. College of Engineering
Bengaluru-59

Sowmya Nag K.
Department of Electronics and Communication
R. V. College of Engineering
Bengaluru-59

Sujata Priyambada Mishra
Department of Electronics and Communication
R. V. College of Engineering
Bengaluru-59

ABSTRACT
Orthogonal Frequency Division Multiplexing (OFDM) is a method of encoding digital data on multiple carrier frequencies. It is a specialized form of Frequency Division Multiplexing (FDM) where the carrier frequencies are orthogonal to each other. It finds applications in wideband digital communication, DSL internet access and power line communication. Fast Fourier transform (FFT) processing is one of the key procedures in popular orthogonal frequency division multiplexing (OFDM) communication systems. Structured pipeline architectures, low power consumption, high speed and reduced chip area are the primary concerns in this VLSI and signal processing domain. A 16 point FFT processor is designed using Radix-2, Radix-4 and Split-Radix algorithms and compare their performances in terms of power, delay, and Power delay product (PDP)). Vedic Multiplier and Kogge Stone adder helps in performing high speed multiplication and addition operations. The processor is implemented in RTL using Verilog HDL. Cadence environment is utilized for performing synthesis and for generating the chip layout.

Keywords
Radix 2, Radix 4, Split radix, Vedic Mathematics, Urdhva Triyakhbhyam, Kogge Stone Adder

1. INTRODUCTION
Discrete Fourier Transform (DFT) is an important operation in the field of Digital signal processing. The DFT differs from Discrete-time Fourier transform (DTFT) as both the input and output sequences are of finite length. Since it deals with finite amount of data, it can be easily implemented in digital systems. Application of DFT includes spectral analysis, data compression, filtering, digital communication (OFDM), radar etc. The equations for DFT and inverse DFT are given below:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (1)$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \quad (2)$$

Direct computation of DFT is not efficient as it does not take into account the symmetry and periodicity properties of the twiddle factor. For an input sequence of length N, $\mathbf{N^2}$ complex multiplications and **N(N-1)** complex additions are involved in direct computation of DFT. The Fast Fourier Transform (FFT) is one of the most efficient algorithms for implementation of DFT as it reduces the number of arithmetic operations involved. The most commonly used FFT is the Cooley-Turkey algorithm. This algorithm uses a divide and conquers approach, recursively breaking down a larger DFT into several smaller DFTs. Other FFT algorithm includes Bruun's FFT [1], Rader's FFT [1], and Bluestein's FFT [1] etc.

Much new architecture for computation of FFT has been developed over VLSI platforms. In [2], a 16-point Radix-4 FFT core was implemented using New Distributed Arithmetic (NEDA), which requires less hardware. Design in [3] uses Radix-4 CORDIC approach for generating twiddle factors used in computation of FFT. The design in [4] focuses on the programmability aspect of FFT architectures for FPGA implementation.

This paper presents 16-point FFT architecture using radix-2, radix-4 and split radix algorithms. Vedic multiplier and Kogge Stone Adder are used for performing multiplication and addition operations with reduced latency, so that FFT computations are fast and feasible for real-time applications like Orthogonal Frequency Division Multiplexing (OFDM). The architecture are designed and implemented in cadence environment and analyzed with respect to number of gates, speed, power and PDP.

The outline of the paper is as follows. Section II gives the brief overview of Radix-2,Radix-4 and split radix algorithms. Section III provides design of multiplier and adder used in the FFT architecture. Section IV presents the measurement result and section V concludes the paper.

2. DESIGN AND WORKING OF DIFFERENT RADIX

2.1 Radix-2 DIT-FFT Algorithm
Consider the computation of an N-point DFT where $N=2^v$ and v is an integer. Applying the divide and conquer approach, the N point data sequence is split into two N/2 point data sequences f1[n] and f2[n] which correspond to the even-numbered and odd-numbered samples of the input x[n] respectively.

$$f1[n] = x[2n] \quad (3)$$
$$f2[n] = x[2n+1] \quad (4)$$

$$Where\ n = 0,1,2, \ldots\ldots, \frac{N}{2} - 1$$

The N-point DFT is represented in terms of DFTs of the decimated sequences as follows:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad \text{for } k=0, 1, \ldots, N-1 \quad (5)$$

$$= \sum_{n\ even} x[n]\, W_N^{kn} + \sum_{n\ odd} x[n] W_N^{kn}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{(2n+1)k} \quad (6)$$

After proper substitution, we finally get

$$X[k] = F1[k] + W_N^k F2[k] \qquad k = 0, 1, \ldots \frac{N}{2} - 1 \quad (7)$$

and

$$X\left(k + \frac{N}{2}\right) = F1(k) - W_N^k F2(K) \qquad k = 0, 1, \ldots \frac{N}{2} - 1 \quad (8)$$

Where F1[k] and F2[k] are DFTs of f1[n] and f2[n] respectively. This decimation process is performed for each of the sequences f1[n] and f2[n], and then repeated again and again till the resulting sequence is a two-point sequence. Thus, the total number of complex addition is reduced to N $\log_2$N. The number of complex multiplication is (N/2)$\log_2$N. There are N/2 butterflies per stage of computation and Log_2N stages.

The basic butterfly computation in the decimation in time (DIT) FFT algorithm is shown in Fig 1.

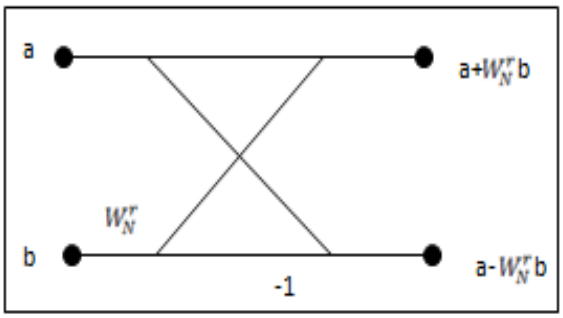

Fig 1: Butterfly structure for Radix-2 DIT- FFT

There are some important observations in this algorithm. Firstly, once a butterfly operation is performed, there is no need to store the inputs. Hence the same memory locations can be used to store the outputs. Therefore, for an N-point DFT, 2N storage locations are needed in order to store the results (N complex numbers). The same 2N locations are used throughout the computation and hence computations are said to be done in place [4]. Secondly, the input sequence x[n] is in bit reversed order while the resulting DFT X[k] is in normal order. If we do not consider the requirement that computations be done in place, then both the inputs and outputs can be in normal order [5].

2.2 Radix -4 FFT Algorithms

Computation of N-point DFT where $N=4^v$, is done using Radix-4 FFT algorithm. The N point DFT is divided into four N/4 point DFTs for implementation of radix 4 FFT [6]. The basic butterfly signal flow structure of radix 4 FFT is shown in fig2.

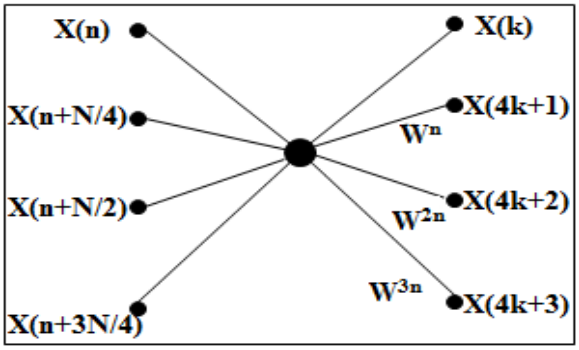

Fig 2: Basic butterfly structure for Radix-4 FFT

The Number of stages for radix 4 FFT is given by $\log_4$N. Thus, a 16-point radix-4 FFT requires 2 stages. Each stage consists of four radix-4 butterfly structures. Each butterfly consists of four inputs and four outputs. The inputs to the radix 4 FFT are given in the bit reversed order. The radix-4 butterfly structure requires 4 complex multiplication and 12 complex addition/ subtraction modules.

A 16-point radix-4 FFT requires four butterfly structures for the first stage of computation where the inputs are given in bit reversed order. The second stage of computation requires another four butterfly structures multiplied by appropriate twiddle factors and the outputs are taken in normal order. The inputs to first stage are x(n),x(n+4),x(n+8),x(n+12) where n=0,1,2,3 for 1st butterfly, 2nd butterfly ,3rd butterfly and 4th butterfly. These inputs are processed and their outputs are available as inputs for the second stage.

In the second stage, the output from each butterfly structure in the first stage is given as input to the first butterfly. For the second butterfly, second output from each butterfly structure in first stage is given as input and so on.

2.3 Split radix FFT Algorithm

Split-Radix FFT (SRFFT) algorithm is a modification of the Cooley-Turkey algorithm which uses both Radix-2 and Radix-4 decompositions in the same algorithm. In Radix-2 algorithm, the even numbered points and the odd numbered points of the DFT can be calculated independently. Thus, there is a possibility of using different methods for independent parts of the algorithm, to reduce the total number of arithmetic operations involved. Split-Radix FFT exploits this idea by using both Radix-2 and Radix-4 decompositions in the same algorithm [7]. It represents an N-point DFT in terms of one N/2-point DFT and two N/4 point DFTs, where $N=2^v$. It combines the simplicity of Radix-2 algorithm with the lesser computational complexity of Radix-4 algorithm to achieve lowest number of arithmetic operations.

Thus, the even numbered samples of the N-point DFT are computed using Radix-2 algorithm.

$$X[2k] = \sum_{n=0}^{\frac{N}{2}-1} (x[n] + x[n + \frac{N}{2}]) W_{N/2}^{nk}$$

(9) $Where\ k = 0,1,2, \ldots \ldots, \frac{N}{2} - 1$

Radix-4 algorithm is used for the odd-numbered samples, and the following N/4 point DFTs are obtained.

$$X[4k+1] = \sum_{n=0}^{\frac{N}{4}-1} \{(x[n] - x[n+N/2]) - j(x[n+N/4] - x[n+3N/4])\} W_N^n W_{N/4}^{kn} \quad (10)$$

$$X[4k+3] = \sum_{n=0}^{\frac{N}{4}-1}\{(x[n] - x[n+N/2]) + j(x[n+N/4] - x[n+3N/4])\}W_N^{3n}W_{N/4}^{kn} \quad (11)$$

The butterfly used in Split-Radix FFT is shown in fig.3

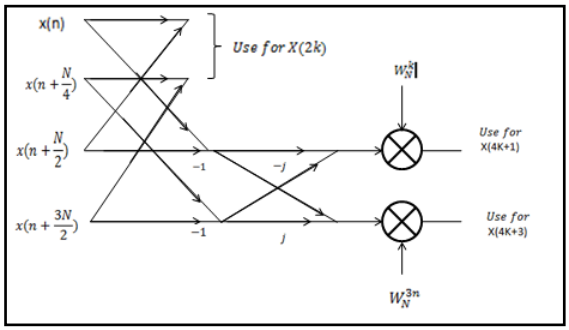

Fig 3:Basic butterfly Structure for Split Radix FFT

2.4. Number Representation and Complex Arithematic Operations

Consider two complex numbers (a+jb) and (c+jd). Each of the numbers has been represented using fixed point arithmetic. The first bit is the sign bit followed by 8 bits for the integer part and 8 bits for the fractional part. The equations for complex addition and subtraction are as shown:

$$(a + ib) + (c + id) = (a + c) + i(b + d) \quad (12)$$

$$(a + ib) - (c + id) = (a - c) + i(b - d) \quad (13)$$

The complex adder is designed using 2 adders. Equation for a complex multiplier is as shown:

$$(a + ib) * (c + id) = (ac - bd) + i(ad + bc) \quad (14)$$

Complex multiplier is designed using 4 multipliers, one adder and one substractor module.

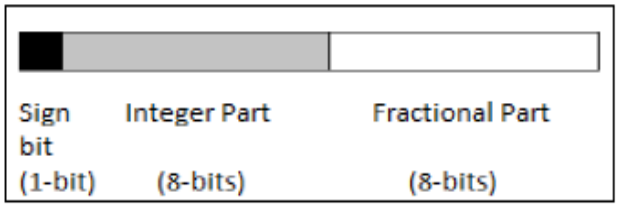

Fig 4: Fixed point number representation

3. DESIGN AND WORKING OF THE VEDIC MULTIPLIER AND KOGGE STONE ADDER USED IN PROPOSED DESIGNS

3.1 Vedic Multiplier

The term Vedic originates from the word "Veda" which means "Store House of Knowledge" [8]. Vedic mathematics can be used to optimize the algorithms used in conventional mathematics for faster operations. Vedic mathematics is described by 16 sutras. It finds application in various fields of mathematics. One such application of Vedic mathematics is design of a multiplier. The Vedic multiplier used in the proposed design utilizes Urdhva-tiryakbyham (UT) sutra. The literal meaning of UT is "vertical and crosswise" [8]. The operation of 2-bit Vedic multiplier is shown in Fig 5.

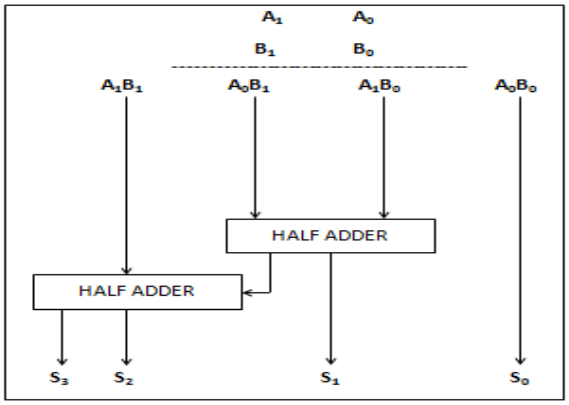

Fig 5: Architecture of a 2x2 Vedic Multiplier

Consider two numbers A and B of 2 bits each. The operation of multiplication using UT sutra is shown below.

$$S[0] = A[0]B[0] \quad (15)$$

$$C[1]S[1] = A[1]B[0] + A[0]B[1] \quad (16)$$

$$C[2]S[2] = C[1] + A[1]B[1] \quad (17)$$

The Same principle can be used for design of Vedic multiplier with increased bit length. 16-bit Vedic multiplier is used for realization of complex multiplication in FFT. The architectural view of 16 point Vedic multiplier is shown in Fig 6.

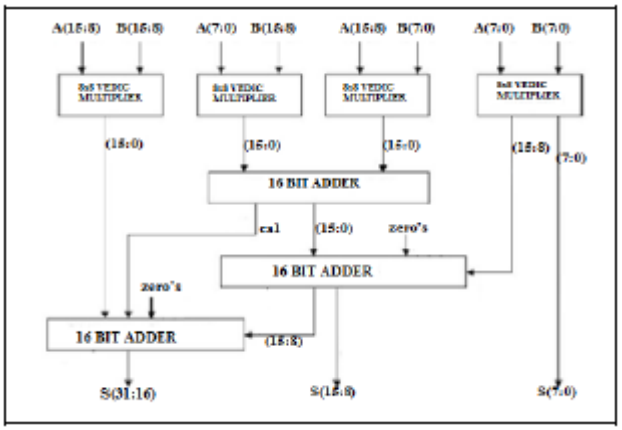

Fig 6: Architecture of 16 x 16 Vedic Multiplier

The 16-bit Vedic multiplier is realized using four 8-bit Vedic multiplier and three 16 bit Adders.

3.2 Kogge Stone Adder (Ksa)

Kogge stone adder, a parallel prefix form of carry look-ahead adder is incorporated for addition of partial products generated in vedic multiplier and complex addition in FFT. KSA is used as fastest adder in high speed design because of minimum logic depth and bounded fan-out. The main reason for high speed is that the carries are computed parallelly. The time for generation of carry signals is of the order of O(log n). The prefix network in KSA has built in redundancy which finds application in fault tolerant designs [9].

The operation of KSA is explained in three different stages:

i. Pre-processing (P and G generation)

ii. Look-ahead carry generation (CP_i and CG_i generation)

iii. Post processing (Computation of Sum)

1. Pre-Processing

The generate(Gi) and propagate(Pi) signals are computed depending on inputs by the Pre-Processing stage. The logic equations are as follows

$P_i=A_i$ XOR B_i

$G_i=A_i$ AND B_i

2. Look-ahead Carry generation(LCG)

High speed of the KSA is achieved by the LCG stage. LCG stage computes carry in parallel and hence achieves high speed. The carry propagate and generate signals are used as input to intermediate stage. The logical equations are shown below:

$CP_{i:j}= P_{i:k+1}$ and $P_{k:j}$ (18)

$CG_{i:j}= G_{i:k+1}$ or ($P_{i:k+1}$ and $G_{k:j}$) (19)

3. Post-Processing

Post processing stage is responsible for generation of sum bits. Post processing stage is incorporated by all carry look ahead adder. The logic equations are shown below:

$C_{i-1} = (P_i$ and $C_{in})$ or G_i (20)

$S_i = P_i$ x or C_{i-1} (21)

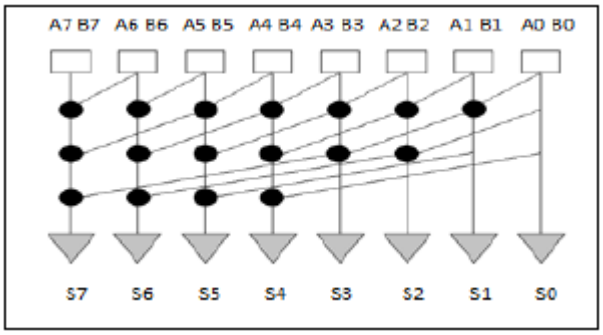

Fig 7: Structure of 8-bit Kogge Stone Adder

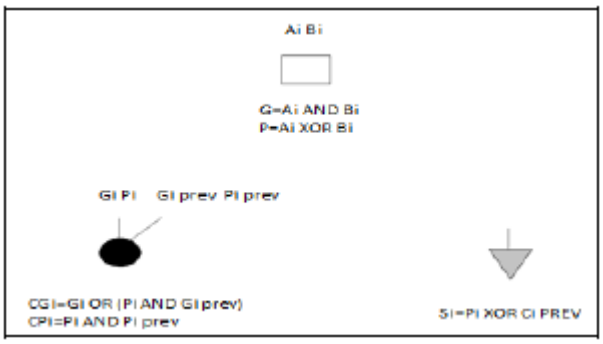

Fig 8: Components of Kogge-Stone Adder

4. LAYOUT AND MEASUREMENT RESULTS

Radix-2, Radix-4 and split Radix are designed, simulated and synthesized in Cadence environment.

Fig 9 shows the simulation waveform of the 16-point split Radix FFT.

Table1 shows the values of Area, Power and timing obtained for radix-2 radix-4 and split radix FFT algorithms. Fig 11 shows the plot for timing, power, and PDP for the three algorithms. From the timing comparison plot, we infer that split radix takes less computation time compared to radix-2 and radix 4. PDP is less for split radix algorithm compared to the Radix-2 and Radix-4 algorithm

Table1. The Experimental results of the proposed designs

Radix Design	Gates	Area (um^2)	Power (mw)	Timing (ns)	PDP (mwns)
Radix-2	32118	57470.364	3.350	2.9	9.716612
Radix-4	78891	139779.504	5.957	2.1	12.51054
Split-Radix	33221	54351.273	2.366	1.5	3.550118

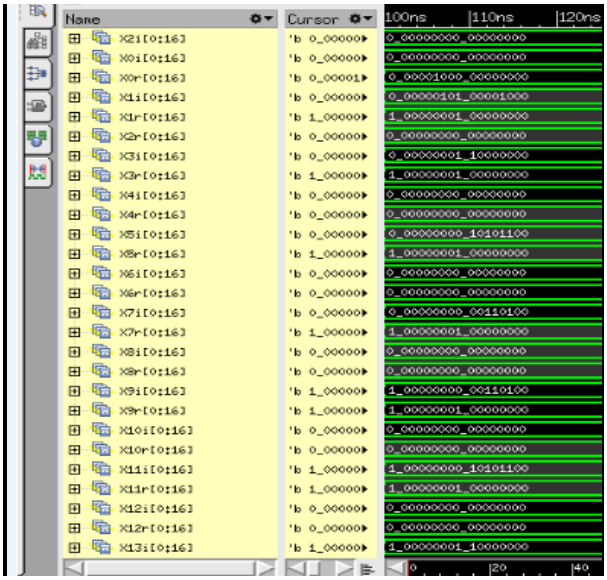

Fig 9:Simulation result for 16-point FFT

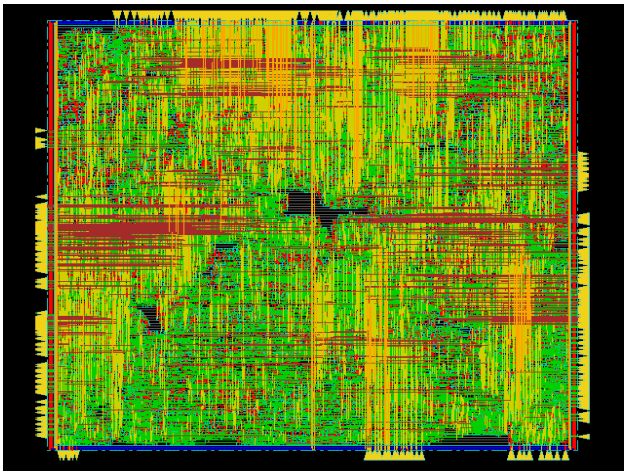

Fig 10:Chip Layout for FFT processor

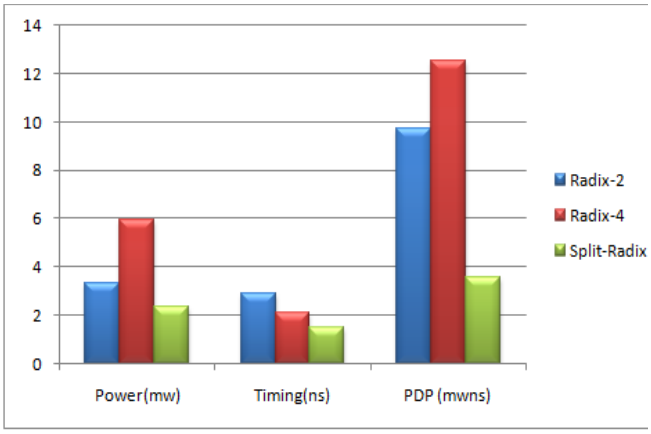

Fig 11: Delay, Timing and PDP comparison of FFT algorithms

5. CONCLUSION

The present paper reported architecture of 16-point split radix FFT core by using Vedic multiplier and kogge stone adder for multiplication and addition respectively.

Proposed design is efficient in terms of power, speed and power delay product compared with Radix-2 and Radix -4 algorithms. The proposed design occupies a chip area of 54351.273μm^2 for 45nm technology. Power consumption is 2.366mW.

6. REFERENCES

[1].R.Nevin,"Application Of The Rader-Brenner Fft Algorithm To Number-Theoretic Transforms" Ieee Transactions On Acoustics, Speech, And Signal Processing Volume 25 Issue 2, 1977.

[2] Abhishek Mankar, Ansuman Diptisankar Das And N Prasad,"Fpga Implementation Of 16-Point Radix-4 Complex Fft Core Using Neda", Students Conference On Engineering And Systems (Sces), 2013.

[3] Edwin Joseph, Rajagopal A, Karibasappa K,"Fpga Implementation Of Radix-2 Fft Processor Based On Radix-4 Cordic", ,Nirma University International Conference On Engineering (Nuicone),2012.

[4] Saikat Kumar Shome,Abhinav Ahesh, Durgesh Kr Gupta, Srk Vadali,"Architectural Design Of A Highly Programmable Radix-2 Fft Processor With Efficient Addressing Logic",International Conference On Devices, Circuits And Systems (Icdcs), 2012.

[5] Beard J,"An Inplace Self Recordering Fft", Ieee International Conference On Acoustics, Speech, And Signal Processing, Icassp '78,Volume:3.

[6] Zhijian Sun, Xuemei Liu , Zhongxing Ji ,"The Design Of Radix-4 Fft By Fpga",International Symposium On Intelligent Information Technology Application Workshops, 2008. Iitaw '08.

[7] Rashmi M J, G S Biradar, Meenakshi Patil,"Efficient Vlsi Architecture Using Dit-Fft Radix-2 And Split Radix Fft Algorithm",International Journal For Technological Research In Engineering ,Volume 1, Issue 10, June-2014.

[8] Honey Durga Tiwari, Ganzorig Gankhuyag, Chan Mo Kim, Yong Beom Cho,"Multiplier Design Based On Ancient Indian Vedic Mathematics", International Soc Design Conference, 2008.

[9]Mangesh B Kondalkar,Arunkumar P Chavan,P Narashimaraja ," Improved Fault Tolerant Sparse Kogge Stone Adder" International Journal Of Computer Applications (0975 – 8887) Volume 75– No.10, August 2013.