Design of High Speed FFT using Urdhva-Tiryagbhyam Algorithm and Karatsuba Algorithm

S. S. Kerur, Mustaq Kunnur

***Abstract:** In gift scenario each method has to be compelled to be quick, adept and simple. Fast Fourier transform (FFT) may be a competent algorithmic program to calculate the N purpose Discrete Fourier transform (DFT).It has huge applications in communication systems, signal processing and image processing and instrumentation. However the accomplishment of FFT needs immense range of complicated multiplications, therefore to create this method quick and simple. It's necessary for a number to be quick and power adept. To influence this problem the mixture of Urdhva Tiryagbhyam associate degreed Karatsuba algorithmic program offers is an adept technique of multiplication [1]. Vedic arithmetic is that the aboriginal system of arithmetic that includes a distinctive technique of calculation supported sixteen Sutras. Using these techniques within the calculation algorithms of the coprocessor can reduce the complexness, execution time, area, power etc. The distinctiveness during this project is Fast Fourier Transform (FFT) style methodology exploitation mixture of Urdhva Tiryagbhyam and Karatsuba algorithmic program based mostly floating point number. By combining these two approaches projected style methodology is time-area-power adept [1] [2]. The code writing is completed in verilog and also the FPGA synthesis on virtex 5 is completed using Xilinx ISE 14.5.*

***Keywords:** Fast Fourier Transform, Urdhva Tiryagbhyam algorithm, Karatsuba algorithm, IEEE 754 single preciseness floating point multiplier, booth multiplier.*

I. INTRODUCTION

Straight computation of Discrete Fourier Transform (DFT) needs of the order of N^2 advanced multiplication operations wherever N is that the rework size. The FFT rule, enduring a modern time in digital signal process by decreasing the order of complexness of DFT from N^2 to $N\log_2 N$, reduces the quantity of necessary advanced multiplications compared to a standard DFT [1].Since multipliers square measure extraordinarily power starved parts in VLSI styles they consequence in giant power consumption. Mixture of Urdhva Tiryagbhyam and Karatsuba rule based mostly floating point number purpose multiplier. The mixture of along algorithms provides a good technique of multiplication. Urdhva-Tiryagbhyam rule could be a greatest rule for binary multiplication in terms of area and delay. In addition, once input bits will increase, delay also will increase and therefore the partial product square measure summed during a ripple manner. Suppose for 8-bit multiplication, we'd like fourteen adders that square measure superimposed during a ripple manner. If we tend to attempt to pay compensation the delay then it'll cause enhance within the space or area. Consequently, Urdhva Tiryakbhyam rule doesn't holds smart, if the input bit length is giant. Hence, if we tend to apply Karatsuba rule at larger bit length and Urdhva-Tiryagbhyam rule at smaller bit length then we tend to presumably can pay compensation the restrictions of the algorithm and therefore the multiplier factor becomes good. This may reduce delay to an outsized quantity. Within the current scenario high rate digital telecommunication systems like Orthogonal Frequency-Division Multiplexing (OFDM) and Digital line (DSL) want period high-speed calculation of the Fast Fourier Transform. This planned project provides a good technique for IEEE 754 floating purpose multiplication which supplies a superior presentation in terms of delay and power. A combination or mixture of Karatsuba rule or algorithm and Urdhva-Tiryagbhyam algorithm is employed to accomplish unsigned binary multiplier factor for fraction multiplication that is also employed for eight point FFT [2]

II. METHODOLOGY

Floating point purpose numbers are sometimes utilized in digital applications like FIR filter, DSP's, Image transformation and Signal process applications. The floating point purpose numbers are described by exploitation IEEE 754 format. It provides two formats for the illustration of a floating point purpose range those are, Single preciseness and Double preciseness format. The single preciseness is of 32-bit (4 bytes) length and also the double preciseness is of 64-bit (8 bytes) length. The MSB bit of a floating purpose variety may be a sign bit. The exponent may be a signed/unsigned whole number. Just in case of single preciseness, the exponent is of 8-bit length and also the fraction is 23-bit. Because the exponent is 8-bit length therefore, it's a spread of - 127 to 128 within the same approach, in double preciseness, the exponent is 11-bit length and whereas fraction is 53-bit and additionally the exponent contains vary from -1023 to 1024.

Revised Manuscript Received on April 25, 2020.
* Correspondence Author

Dr. S. S. Kerur*, Assistant Professor, Department of Electronics and Communication Engineering at SDM College of Engineering and Technology, Dharwad, Karnataka, India.
Mr. Mustaq Kunnur, M.Tech student, Department of Electronics and Communication Engineering at SDM College of Engineering and Technology, Dharwad, Karnataka, India

A. Floating point purpose multiplier design

A floating point range has four elements specifically sign, exponent, significand or fraction and therefore the exponent base. A floating purpose range is diagrammatic represented in IEEE-754 format as significand × base $^{\text{exponent}}$. The exponent base supposed for binary format is 2. To attain multiplication of two floating purpose numbers $\mp s1\times b^{e1}$ and $\mp s2\times b^{e2}$, the significand or fraction elements are increased to urge the product fraction and exponents are additional to urge the product exponent [5].It is shown in figure no.1

i.e.; the product is $\mp(s1\times s2)\times b^{(e1+e2)}$. (1)

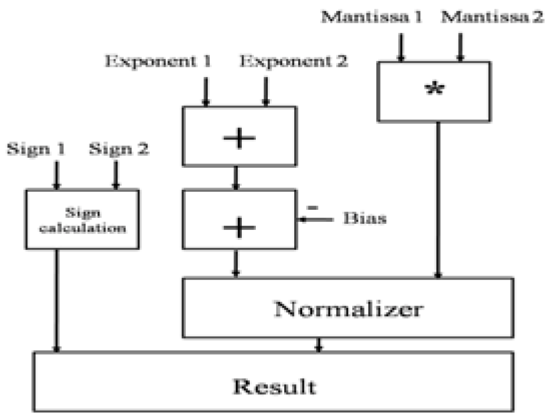

Figure:1 Floating point purpose multiplier

1. Sign calculation

The Most Significant Bit (MSB) of floating purpose range represents the sign bit. The sign of product are positive if each the numbers square measure similar sign and can be negative if numbers square measure of converse or opposite sign. So, to get the sign of product, we are able to use an easy X-OR circuit because the sign calculation.

2. Addition of exponents

To obtain the merchandise or summation result of exponents, the input exponents square measure additional at the same time. Since we have a tendency to use a bias in floating purpose format exponent, we have a tendency to need to work out the bias from the total of exponents to get the actual exponent. The worth of bias is 127_{10} and in binary (01111111_2) meant for single preciseness format and 1023_{10} and in binary (01111111111_2) meant for double preciseness format. The procedure time of fixed-point part multiplication operation is greatly additional that exponent addition. Thus a straightforward ripple borrow work out is perfect exponent addition.

3. Normalization of the end result

Normalization of the result floating purpose representations cover a hidden bit within the fixed-point part, that continually have the worth one and so it's the not hold on within the memory to save lots of one bit. Whereas leading one within the fixed-point part is taken into account to be the hidden bit, i.e. the one immediate to the left of percentage point. Typically standardization is ready by shifting, so the MSB of fixed-point part becomes non zero and in number a pair of, nonzero means that one. The percentage point within the fixed-point part multiplication is shifted left if the leading one isn't next to the immediate left of percentage point. And supposed for every left shift operation of the result, the exponent price is incremented by one. This is often called standardization of result. as a result of the worth of hidden bit is usually one, it's known as 'hidden 1'.

B. Mantissa multiplication of floating point numbers

As shown in figure no. 2, its accustomed build multiplication operation adept a combination of Karatsuba and Urdhva-Tiryagbhyam algorithms square measure applied. Karatsuba algorithmic program employs split and conquer technique within which it split a specified quantity into or

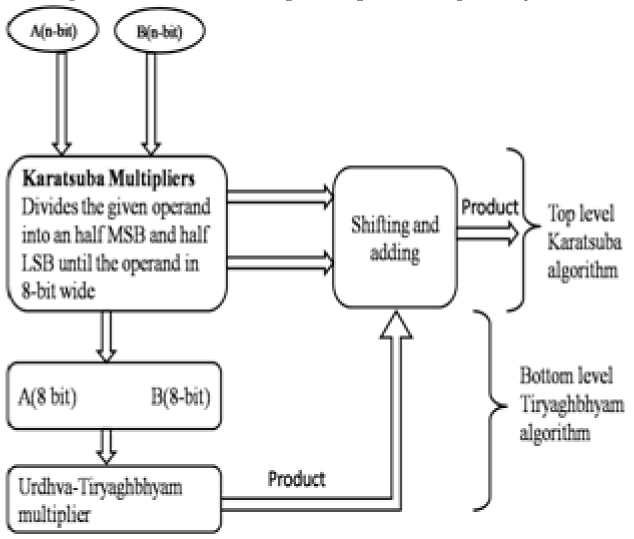
A(n-bit)
B(n-bit)
Karatsuba Multipliers
Divides the given operand into an half MSB and half LSB until the operand in 8-bit wide
Shifting and adding
Product
Top level Karatsuba algorithm
A(8 bit)
B(8-bit)
Bottom level Tiryaghbhyam algorithm
Urdhva-Tiryaghbhyam multiplier
Product

Figure: 2 Mantissa multiplications using Karatsuba and Urdhva-Tiryagbhyam algorithm.

Curious about most significant Bit (MSB) and Least significant Bit (LSB) and this continues till the quantity is 8-bit length. This algorithmic program is competent with larger bit lengths. However at lesser bit lengths it's not competent. Thus to surmount this, a combination of Karatsuba and Urdhva-Tiryagbhyam algorithms is employed. Urdhva-Tiryagbhyam algorithmic rule may be a finest algorithmic rule for binary multiplication in terms of space and delay. However, once input bits will increase, delay will increase and also the partial merchandise is summed during a ripple manner.

In case of associate degree 8-bit multiplication, we have a tendency to need fourteen adders that are extra during a ripple manner. If we have a tendency to try and pay compensation the delay then it'll cause the rise in space as a result, Urdhva Tiryakbhyam algorithmic rule doesn't match well if the input bit length is a lot of. Thus, if we have a tendency to use Karatsuba at larger bit length and Urdhva-Tiryagbhyam algorithmic rule at lesser bit length then we have a tendency to could compensate the restrictions of the algorithmic rule and also the multiplier factor becomes skillful. This can cut back delay to a good level. Figure no. two shows study blocks for Karatsuba and Urdhva-Tiryagbhyam algorithms

C. Fast Fourier Transform

Fast Fourier Transform (FFT) is a competent algorithm developed by Cooley & Tukey in 1965, used to calculate the DFT with condensed computation. Due to the competence of FFT, it is used for digital signal processing applications. FFT reduces the difficulty of calculating an N – point DFT to that of calculating many lesser sized DFTs [1].

Time domain to frequency domain conversion is done by (DFT). The X(k) is function in DFT, whereas x(n) is N point sequence is defined in Equation.1

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\left(\frac{2\pi}{N}\right)} kn, 0 \le k \le N-1 \quad (1)$$

The most popular as well as easy representation is shown in in Eq. 2,

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}, \qquad 0 \le k \le N-1 \quad (2)$$

Where $W_N = e^{-j\left(\frac{2\pi}{N}\right)}$, is the twiddle factor.

The N-point FFT will uses N/2 complex multiplications per stage by some power of W_N, except for the final 2-point DFT stage where no multiplication is needed. The radix-2 butterfly computation, seen in Figure no.3, is used to simplify the calculation [1].

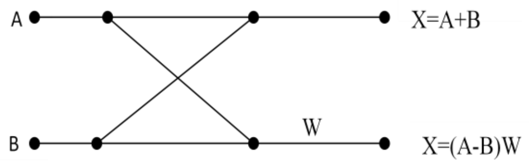

Figure:3 Two point DFT stage

Discrete Fourier Transform (DFT) or it's Inverse Discrete Fourier Transform (IDFT) sequences are computed by a Fast Fourier Transform (FFT) algorithm. Frequency domain from time domain conversion is also performed by Fourier Transform. A FFT promptly calculates this kind of transformation by dividing the DFT matrix into a product of spare factors. As a consequence, it manage to lessen the complexity of computing the DFT from $O(N^2)$ which arise if one just applies the definition of DFT, to O (N log N) , where N is the data size.

The methods like Decimation – In- Time (DIT) and Decimation – In – Frequency (DIF) FFT algorithms use the "divide – and – conquer" approach. This is achievable if the length of the sequence N is selected as $N = r^m$. here, r is known as the radix of the FFT algorithm.

The most sensibly implemented choice for r = 2 leads to radix – 2 FFT algorithms. So, with N = N = 2^m, the proficient computation is achieved by breaking the N – point DFT into two $\frac{N}{2}$- point DFTs, then breaking each $\frac{N}{2}$- point DFT into two $\frac{N}{4}$- point DFTs and continuing this course of action until 2 point DFTs are obtained. Figure 4 shows the three stages in calculation of an N=8 point DFT

III. RESULTS AND ANALYSIS

Urdhva Tiryagbhyam and Karatsuba algorithm based IEEE 754 single preciseness floating point number multiplier designed to implement the 8 point DIT- FFT and booth algorithm based IEEE 754 single preciseness floating point number multiplier is also designed to implement the 8 point DIT- FFT. Verilog HDL is used for is design and implementation. It is synthesized and simulated using Xilinx 14.5 synthesis tool (ISE 14.5) targeted on virtex-5. And 8-point FFT using Urdhva Tiryagbhyam and Karatsuba algorithm based IEEE754 single preciseness floating point number multiplier results are compared with 8-point FFT using booth algorithm based IEEE 754 single preciseness floating point multiplier. The summary of results on virtex-5 is also shown below. And performance of both FFT is compared in terms of area, delay and power

A. Synthesis results

Table: I Device utilization summary of Urdhva Tiryagbhyam and Karatsuba algorithm IEEE single preciseness floating point multiplier based 8-point FFT design

Device utilization summary			
Number of Slice Registers	32	207,360	1%
Number used as Latch-thrus	32		
Number of Slice LUTs	38,823	207,360	18%
Number used as logic	38,341	207,360	18%
Number using O6 output only	32,169		
Number using O5 output only	103		
Number using O5 and O6	6,069		
Number used as exclusive route-thru	482		
Number of route-thrus	649		
Number using O6 output only	553		

Table: II Delay summary of Urdhva Tiryagbhyam and Karatsuba algorithm IEEE single preciseness floating point multiplier based 8-point FFT design

Delay	
Gate delay	43.711ns
Net delay	133.951ns
Maximum combinational path delay	177.663ns

Table: III Power consumption summary of Urdhva Tiryagbhyam and Karatsuba algorithm IEEE single preciseness floating point multiplier based 8-point FFT design

Power consumption summary			
	Total	Dynamic power	Static power
Total power (mW)	3351.63	57.53	3294.10

B. Simulation results

Figure: 4 Simulation results of Urdhva Tiryagbhyam and Karatsuba algorithm IEEE single preciseness floating point multiplier based 8-point FFT design

C. Comparison between Urdhva Tiryagbhyam and Karatsuba algorithm and booth algorithm IEEE 754 single preciseness floating point multiplier based FFT

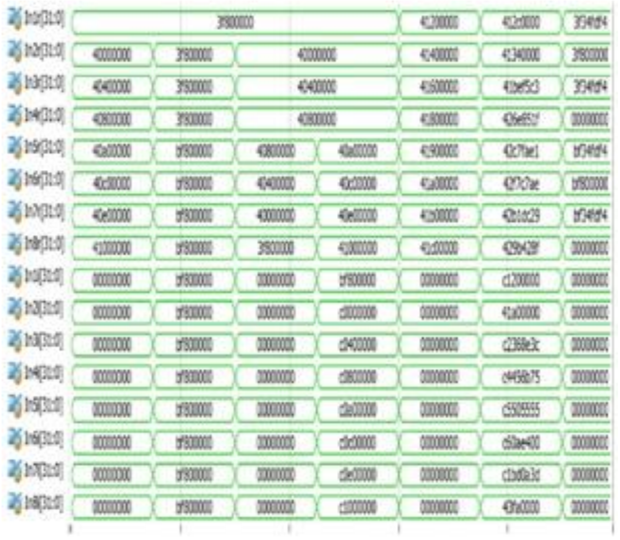

Table 4: Delay comparison between Urdhva Tiryagbhyam and Karatsuba algorithm and booth algorithm IEEE 754 single preciseness floating point multiplier based FFT

Vertex 5 XCVLX330 FF1760 -2	Delay in Nano seconds		
	Overall Delay	Gate Delay	Path Delay in ns
FFT Design based on proposed Multiplier	177.663	43.711	133.951
Design based on Booth Multiplier	180.338	39.955	140.383

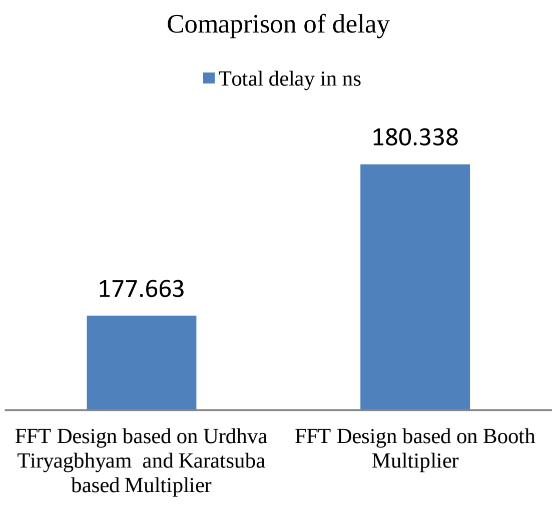

Figure 5: Area comparison between Urdhva Tiryagbhyam and Karatsuba algorithm and booth algorithm IEEE 754 single preciseness floating point multiplier based FFT

Table V: Area and power consumption comparison between Urdhva Tiryagbhyam and Karatsuba algorithm and booth algorithm IEEE 754 single preciseness floating point multiplier based FFT

Vertex 5 XCVLX330 FF1760 -2	Area	Power in milli Watts
	Slices	
FFT Design based on proposed Multiplier	177.663	57.53
Design based on Booth Multiplier	180.338	73.97

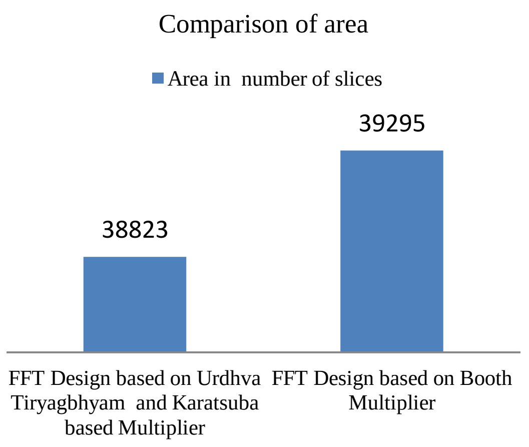

Figure 6: Area comparison between Urdhva Tiryagbhyam and Karatsuba algorithm and booth algorithm IEEE 754 single preciseness floating point number multiplier based FFT

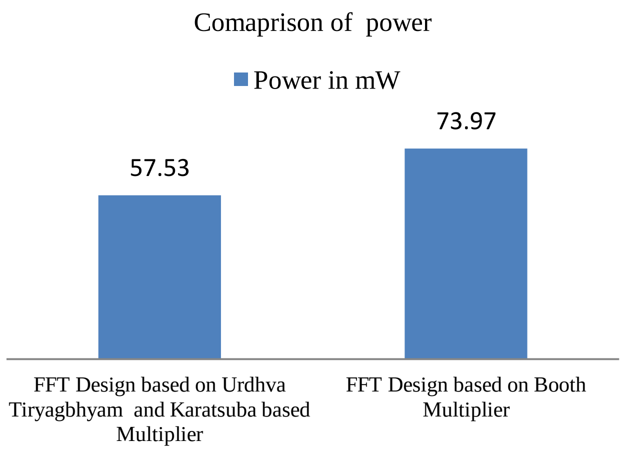

Figure 7: Power consumption comparison between Urdhva Tiryagbhyam and Karatsuba algorithm and booth algorithm IEEE 754 single preciseness floating point multiplier based FFT

IV. CONCLUSION

In any Very Large Scale Integration (VLSI) circuit design the performance is restricted with the ingredient parameter like area on chip, delay by circuit and power consumption. In applications like digital signal processing there is necessity of a proficient algorithm. Largest part of the power is consumed by Butterfly structure in FFT algorithm. So, implementation of a proficient butterfly is the most essential part of the FFT algorithm. Floating point number multipliers have been designed, implemented and simulated using Xilinx Integrated Synthesis Environment (ISE) 14.5 design suite while targeting virtex 5 Field Programmable Gate Array (FPGA) for synthesis. In this project using mixture Urdhva Tiryagbhyam and Karatsuba rule or algorithm is used for IEEE 745 single preciseness floating point number multiplier implementation because as compared to other multiplier proposed multiplier is far better in terms of delay, area and power as well.

The high speed multiplier circuit is implemented using this competent Urdhva Tiryagbhyam and Karatsuba algorithm it is also used to implement for progress of an efficient multiplier block. Butterfly structures for efficient FFT algorithm are designed using proficient adders and multipliers blocks. By results analysis we found that the performance of 8 point FFT based on proposed multiplier is much better in terms of delay, area and power as compared 8 point FFT based on booth multiplier. The verilog coding and project implementation is performed by using tool Xilinx ISE 14.5 design suite. Xpower analyzer is used to perform power consumption analysis of circuit. Future work is dedicated for further reducing the power consumption and area.

REFERENCES

1. R.K.Sarin ,Ashish Raman, "High Speed Reconfigurable FFT Design by Vedic Mathematics," journal of Computer Science and Engineering, vol. 1, pp. 59-63, May 2010.
2. Anagha V. Choudhari, Nivedita A. Pande, "Vedic Mathematics for Fast Multiplication in DSP," International Journal of Engineering and Innovative Technology (IJEIT), vol. 2, no. 8, pp. 245-247, February 2013.
3. Swami Sri Bharati Krsna Thirthaji Maharaja, Vedic mathematics.: Motilal Banarasidass Indological publishers and Book sellers, 1965.
4. S. Arish and R. K. Sharma, "An proficient floating point multiplier design for high and rapid applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm," International Conference on Signal Processing and Communication (ICSC), vol. 1, pp. 453-457, February 2015.
5. V Rajaraman, Computer Oriented Numerical Methods, 3rd ed.: PHI, January 2016.
6. Behrooz Parhami, Computer Arithmetic.: Oxford University Press, 2000.
7. V.P. Gejj,B.R.Pandurang,Kanchan A. Joshi Akshata R, "Design of high speed FFT using Vedic mathematics," International Journal of Modern Trends in Engineering and Research, vol. 02, no. 06, pp. 482-491, June 2015.

AUTHORS PROFILE

Dr. S. S. Kerur is an Assistant Professor in the department of Electronics and Communication Engineering at SDM College of Engineering and Technology, Dharwad, Karnataka, India. He obtained his Bachelor of Electronics & Communication Engineering from Basaveshwar Engineering College, Bagalkote. Master degree in Digital Electronics from SDMCET, Dharwad. Ph.D. from VTU, Belagavi, Karnataka, India. Guiding 04 no. of Ph.D. students at Visvesvaraya Technological University, Belagavi, Karnataka and guided 29 U.G. and 15 P.G. students. Conducted 2 National Conference and 04Workshop on during my tenure and achieved 02 awards. Published 05 no. of papers at International Journal and 07 at conferences and life member of ISTE,IE,IETE and VSI.

Mr. Mustaq Kunnur is M.Tech student in the department of Electronics and Communication Engineering at SDM College of Engineering and Technology, Dharwad, Karnataka, India. He obtained his Bachelor of Electronics & Communication Engineering from B. V. Bhoomaraddi College of Engineering Technology. Hubballi, Karnataka. He is pursuing Masters in Digital Electronics from SDMCET, and he has scored 9.76 CGPA till third semester. He also has one year industrial experience and 6 years of teaching experience as lecturer at Government Polytechnic Mundagod, Karnataka

.