ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 10, October 2016

VLSI Implementation of High Speed and Area Efficient Double-Precision Floating Point Multiplier

Ramireddy Venkata Suresh[1], K.Bala[2]

[1] M.Tech, Dept of ECE, Srinivasa Institute of Technology and Science, Ukkayapalli, kadapa, India

[2] Associate Professor, Dept of ECE, Srinivasa Institute of Technology and Science, Ukkayapalli, kadapa, India

***Abstract*—Floating-point arithmetic is ever-present in computer systems. All most all computer languages has supports a floating-point number types. Most of the computer compilers called upon floating-point algorithms from time to time for execution of the floating-point arithmetic operations and every operating system must be react virtually for floating-point exceptions like underflow and overflow. The double-precision floating arithmetic is mainly used in the digital signal processing (filters, FFTs) applications, numerical applications and scientic applications. The double-precision floating arithmetic operations are the addition, the subtraction, the multiplication, and the division. Among the all arithmetic operations, multiplication is widely used and most complex arithmetic operation. The double-precision (64-bit) floating point number is divide into three fields, Sign field, Exponent field and Mantissa field. The most significant bit of the number is a sign field and it is a 1-bit length, next 11-bits represents the exponent field of the number and remaining 52-bits are represents the mantissa field of the number. The double-precision floating-point multiplier requires a large 52x52 mantissa multiplications. The performance of the double-precision floating number multiplication mainly depends on the area and speed. The proposed work presents a novel approach to decrease this huge multiplication of mantissa. The Urdhva Tiryagbhyam technique permits to using a smaller number of multiplication hardware compared to the conventional method. In traditional method adding of the partial products are separately done and it takes more time in comparision with the proposed metdod. In proposed method the partial products are concurrently added with the multiplication operaton and it canreduce the time delay. The double-precision floating multiplier is implemented using Verilog HDL with Xilinx ISE tools on Virtex-5 FPGA.**

Keywords-Double-precision, Floating point, Multiplication, Vedic, Urdhva Tiryagbhyam, IEEE-754, Virtex-5 FPGA.

I. INTRODUCTION

The real numbers[2] are represented in binary format is called as floating point numbers. The floating-point numbers are IEEE-754 standards split into two types, first one is binary interchange format and second one decimal interchange format. The IEEE-754 standard[8] of floating-point numbers are the single precision (32-bit) format and the double-precision (64-bit) format.

The fig.1 represents single-precision (32-bit) floating point number format. In single-precision floating point number has 32-bits of length. The total 32-bits are divided into three fields, 31st bit of the number sign (1-bit) field, 30th bit to 23rd bit of the number is exponent (8-bits) field and 22nd bit to 0thbit of the number is mantissa (23-bits) field.

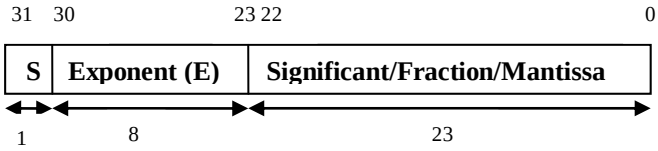


Fig. 1: Single Precision (32-bit) floating point number format representation.

The fig. 2 represents the double-precision (64-bit) floating point number format. The double-precision floating point number has 64-bits of length. The total 64-bits are divided into three fields, 63rd bit of the number is sign (1-bit) filed, 62nd to 52nd bit of the number is exponent (11-bits) filed and 51st to 0th bit of the number is mantissa (52-bits) filed.

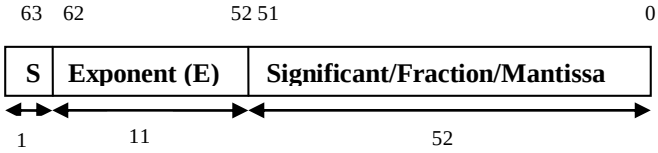


Fig. 2: Double Precision (64-bit) floating point number format representation.

The IEEE-754 standard of hardware implementations for floating point arithmetic operations are very important in all processors. The multiplication operation is crucial and very important among the all. The area of the application is main constraint which should be minimum and provides a greater speed. The area efficient implementation of floating point arithmetic operation and thus the efficient implementation of floating point multiplier are of a major concern.

Most of the high performance computations can be implemented by using the FPGAs (Field Programmable Gate Arrays)[7]. The FPGAs supports the high speed, large amount of logic resources i.e. large slice count and large LUTs and limited accessible on – board intellectual property (IP)[4]

cores builds a large set of applications. Now a days FPGAs can be used in tremendous applications like arithmetical and scientific computations, digital signal and image processing, communications, and cryptography computations. The proposed work is mainly focusing on the double-precision floating point multiplication implementation on FPGA platform.

The mantissa multiplication is most important and essential part for multiplication of the double precision floating point numbers. This is the main bottleneck of the performance. The double precision floating point number's mantissa is in 53-bits length, and this require a hardware implementation of 53x53 multipliers, which is very cost effective and expensive. In this work, one algorithm has proposed for the double precision floating numbers multiplication and it can be permits to using a reduced amount of multiplication accomplish a high speed at comparatively low hardware resources cost. The prosed method of multiplication results can be Comparing with Karatsuba algorithm[1] results. The implementation is mainly concerned only normalized numbers. The implementation can be done by using Xilinx ISE synthesis tool, ISIM simulator and Virtex-5[4](xc5vlx110t-1ff1136) speed grade-1 FPGA platform.

The paper contains the Introducation in section 1, section 2 has Proposed Approach of Design, section 3 has implement the design, section 4 has design results and finally conclusion of the paper is presenting section 5.

II. PROPOSED APPROACH OF DESIGN

ThenKaratsuba Multiplication Technique is using to designing proposed method of multplication. This multiplication technique is break up the two mantissas into three parts in fig. 3.

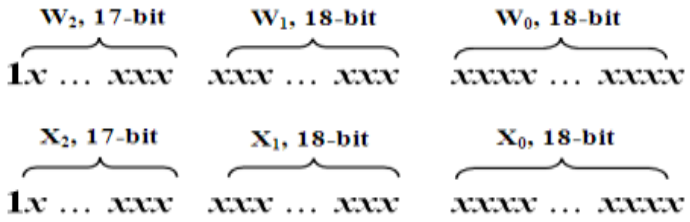


Fig. 3 53-bit mantissa is splitted into 17-bit and 18-bit fractions

The 53-bit mantissa[1] is splitted into 17-bit and 18-bit signed fractions. The terms w_0 and w_1 represents the 18-bit fraction and w_2 represents the 17-bit fraction of the one of the input operand. In the same way x_0 and $x_1$represents the 18-bit fractions and x_2 represents 17-bit fraction.

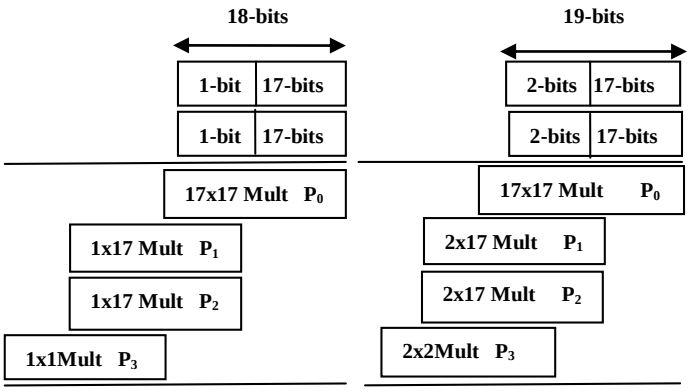


Fig.4 18-bit and 19-bit Multipliers

The designing of 53x53-bit mantissa multiplication is realized by using the 18-bit multiplier and 19-bit multipliers. The fig. 4 represents the 18-bit multiplier and 19-bit multipliers. The multiplier of 18-bit has 1-bit sign bit and 17-bits fractional bits, the first step is to multiply two 17-bits of the input operands and generate a partial product p_0 then generate a p_1,p_2, and p_3. In the same way 19-bit multiplier is also design.

The designing of Vedic multiplier is supports Vedic multiplication[5] principles (sutras). These principles are broadly used for the multiplication of two large decimal numbers. This same sutra can be used to perform multiplication[5] on two large binary numbers and it shown in fig. 5.. This proposed method is compatible to design digital hardware system. The Urdhva Tiryagbhyam technique[4] can take (2n-1) steps for designing the n-bit multiplier. In fig. 5 4-bit multiplier can take 7-steps for designing of the multiplier.

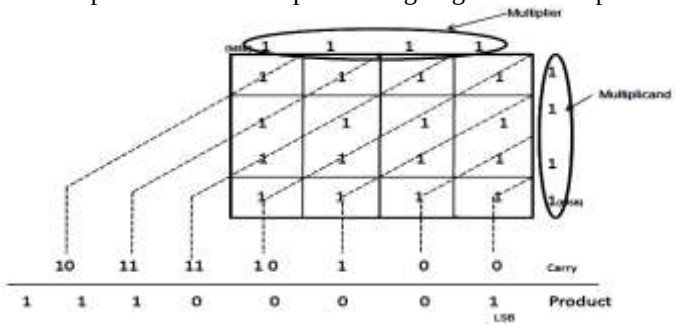


Fig.5 Steps involved for 4-bit binary numbers multiplication using Urdhva Tiryagbhyam Technique.

The 4-bit binary multiplier using Urdhva Tiryagbhyam technique is shown in fig. 5. The 4-bit multiplier has 4-bit multiplier as one of the operand and 4-bit multiplicand is the another operand. The carry resents the previous state generated carry and it can be added to the current to get the final product.

III. IMPLEMENTATION

Designing of the multiplier for floating-point numbers carries the operation separately by sign bit, exponent bits and mantissa bits.

A. Sign and Exponent operations.

The sign and exponent operations are performed in a straightforward manner. The operand 1 and operand 2 of sign-bits are carry out the operation of logical XOR.

$$Sign\_out = Sign\_in1 \oplus Sign\_in2$$

The resultant exponent field of the number is addition of operand 1's exponent and operand 2's exponent and subtracting the BIAS i.e.

$$Exp\_out = Exp\_in1 + Exp\_in2 - 1023$$

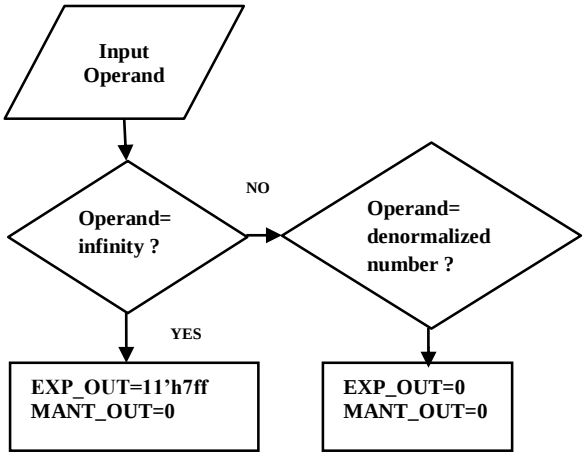
For double-precision floating number, the BIAS is equals to 1023 i.e. $(2^{11-1}-1)$.

The BIAS of the any floating point number is determined by $(2^{\text{exponent bits}-1}-1)$.

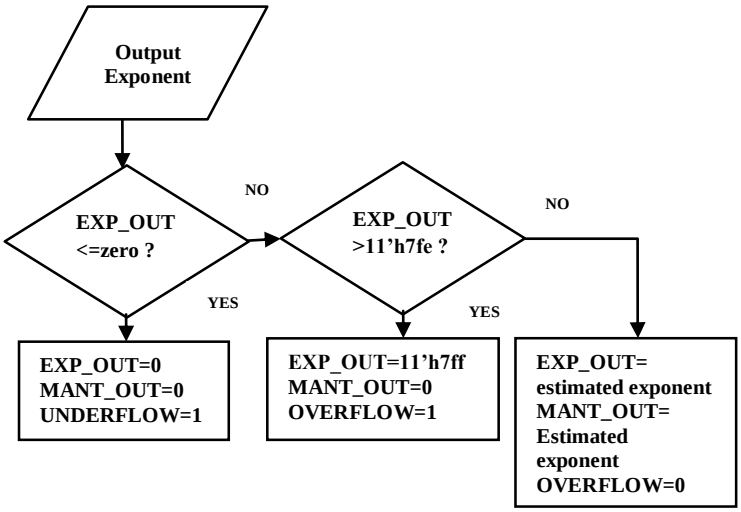
B. Exceptional Case Handling

The IEEE standard as defined by the many exceptional cases like NaN (Not a Number), INFINITE, ZERO, UNDERFLOW, OVERFLOW. These are appearing for all the floating point arithmetic operations. The main operation has been also combined with the detection of all the exceptional cases, and determining the final output as per standard. All the exceptional cases executions are in parallel with the IEEE-754 standard.

The fig. 6 represents the flowchart of the handling of exceptional cases.



(a)



(b)

Fig. 6 Flowchart for handling of exceptional cases (a) infinite case (b) OVERFLOW and UNDERFLOW cases

If one of the operand or both of the operands are infinity; then generate a INFINITY output (with calculate sign-bit). If denormalized number is one of the input operand, then the output is zero (w.r.to sign-bit). If the zero or below zero as output exponent, the displayed output is UNDERFLOW, and the 11'h7fe (2046 in decimal) is the output exponent, OVERFLOW as displayed output.

C. Normalization and Rounding

For getting of the the final result after mantissa multiplication the normalization is mandatory. After the mantissa multiplication, the resultant number has one extra bit in most significant bit before the decimal point. Similarly, in sometimes the same situation will arise after rounding. So, whenever the additional carry bit is produced after multiplication or rounding, the resultant product must be right-shifted for necessary shifts and the corresponding alterations should be made in exponent also to get the normalized result.

Rounding is necessary to get back the 106-bit mantissa multiplication result to 53-bit result only. In this paper only implemented Round to nearest rounding mode precised by the IEEE-754 standard and remaining modes can be implementing as per the application requirement.

IV. RESULTS

The FPGA implemenatation of double precision floating point multiplier using Urdhva Tiryagbhyam technique is divided into 18-bit multiplier, 19-bit multiplier and 53-bit mantissa multiplier. The 53-bit multiplier is designed by combining the 18-bit and 19-bit multipliers. The 18-bit multiplier, 19-bit multiplier and 53-bit mantissa multipliers are in followed figures 7-16.

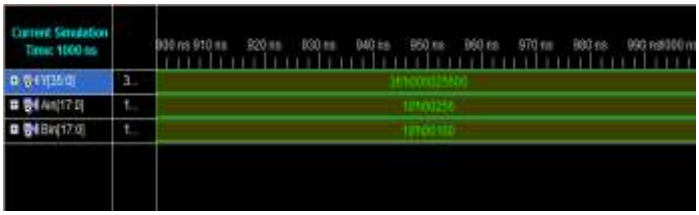


Fig. 7 Simulation result of 18-bit Urdhva Tiryagbhyam multiplier

The simulation result of 18-bit multiplier using Urdhva Tiryagbhyam multiplication technique is in fig. 7.Here A_{in} and B_{in} are the two input signals.In this A_{in} is one of the operand which is 18-bits in length and B_{in} is other operand which is also an 18-bits in length. These two operands are performing a multiplication operation and produce a 36-bit of result.
INPUTS: A_{in}[17:0]=18'h00256; B_{in}[17:0]=18'h00100;
OUTPUT: Y[35:0]=36'h000025600;

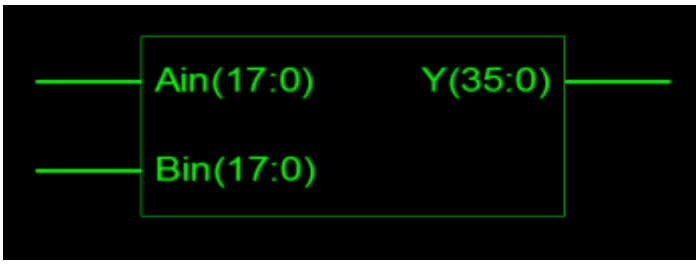


Fig. 8 RTL Schematic of 18-bit Urdhva Tiryagbhyam multiplier

The fig. 8 represents the RTL Schematic of 18-bit multiplier using Urdhva Tiryagbhyam multiplication technique. In this Ain and Bin are the input operand each one has the 18-bits in length and output Y is 36-bits in length.

Fig. 9 Simulation result of 19-bit Urdhva Tiryagbhyam multiplier

The simulation results of 19-bit multiplier using Urdhva Tiryagbhyam multiplication technique is in fig. 9. A_{in} and B_{in} are the two input operand of the 19-bit multiplier and Y is the output produced by the multiplier.
INPUTS: A_{in}[18:0]=19'h00019; B_{in}[18:0]=19'h0000F;
OUTPUT: Y[37:0]=38'h0000000177;

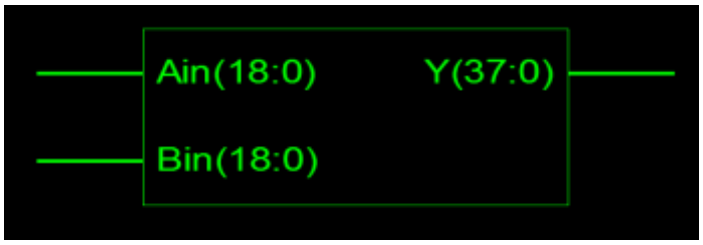


Fig. 10 RTL Schematic of 19-bit Urdhva Tiryagbhyam multiplier

The fig. 10 is the RTL Schematic of 19-bit multiplier using Urdhva Tiryagbhyam multiplication technique. A_{in} and B_{in} are two operands which are give to input to the multiplier and output of the multiplier displays as Y which is 38-bits in length.

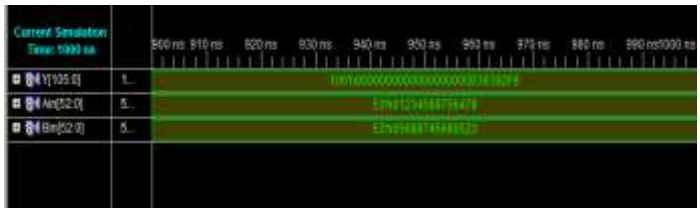


Fig. 11 Simulation result of 53-bits mantissa multiplier

Simulation result of 53x53-bit multiplier is in fig. 11. In this A_{in} and B_{in} are the inputs of the multiplier; each one has 53-bits in length and output Y has 106-bits in length.
INPUTS:A_{in}[52:0]=53'h01234589756478;
B_{in}[52:0]=53'h05689745689523;
OUTPUT: Y[105:0]=106'h0000000000000000005B36392F6;

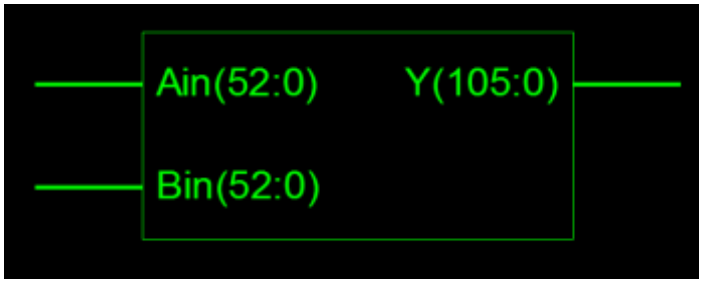


Fig. 12 RTL Schematic of 53-bit mantissa multiplier

The mantissa bits of the double-precision floating point multiplier have 53-bits. The 53-bit x 53-bit multiplier design is very difficult, the 53-bits are divided into 18-bits and 19-bits and design the multipliers. These multipliers are added to get the 53-bit mantissa multiplier.The fig.12 shows the RTL Schematic of 53x53-bits multiplier using Urdhva Tiryagbhyam multiplication technique. Here Ain and Bin are the input operands of the 53-bit multiplier; Y is the output of the multiplier.

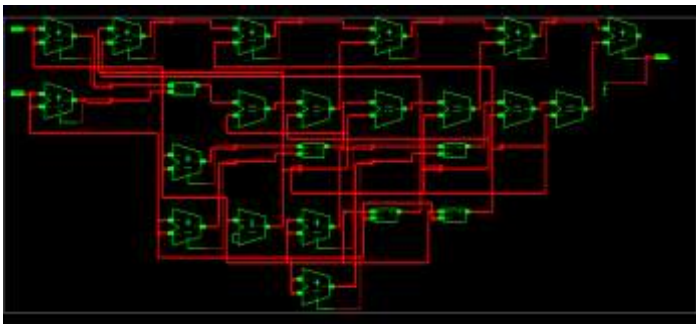


Fig. 13 Technology Schematic of 53-bit mantissa multiplier

The fig. 13 shows the technology schematic of the mantissa bits multiplier. In this the red color wires are interconnecting wiring network of the circuit. The technology schematic has adders, subtractors and multipliers. These components internally through wiring network to build the actual circuit.

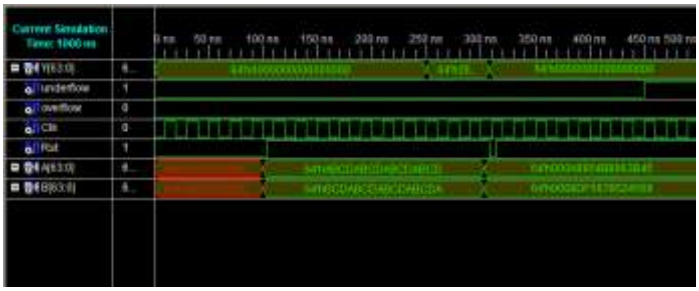


Fig. 14 Simulation result of double-precision Urdhva Tiryagbhyam multiplier

The resulting waveform of the double-precision Urdhva Tiryagbhyam multiplier has two inputs and each one has 64-bits wide. The two 64-bit input operands are performed multiplication operation; the resultant has 64-bits length.The fig. 14 shows the simulation resulting waveform of double-precision Urdhva Tiryagbhyam multiplier. In this the exceptional conditions like overflow and underflow are checked by applying the inputs A[63:0]=000a8954b8963b45 and B[63:0]=0008df5678524569. The output will display all zeros because of an exceptional case. The exceptional case is resultant exponent is less than or equal to zero.

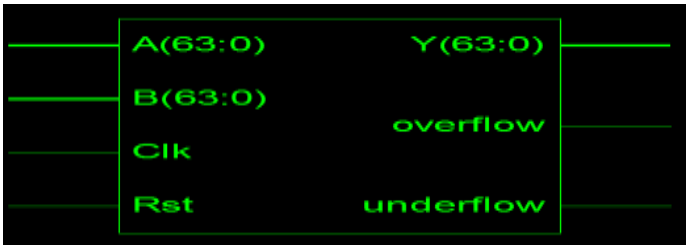


Fig. 15 RTL Schematic of double-precision Urdhva Tiryagbhyam multiplier

The RTL Schematic of double-precision Urdhva Tiryagbhyam multiplier is shown in fig. 15. The inputs are A[63:0], B[63:0], clock and reset and outputs areY[63:0], underflow and overflow. A[63:0] has 64-bit of floating point operand and B[63:0] has 64-bit of another floating point operand. These two operands are applied to inputs of floating point multiplier. The Y[63:0] is 64-bit floating point multiplier output. Overflow and underflow are exceptional conditions occur during the floating point multiplication.

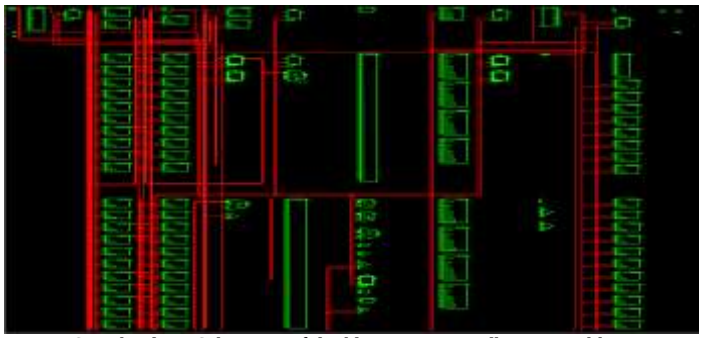


Fig.16 Technology Schematic of double-precision Urdhva Tiryagbhyam multiplier

The Technology schematic is in fig. 16 represents the how the components are internally connected. It also represents how the circuits are actually arranged in the FPGA.

Table 1: Device utilization summary.

Device	No. of Slice Registers		No. of Slice LUTs	
	Karatsuba Multiplier	Urdhva Tiryagbhyam Multiplier	Karatsuba Multiplier	Urdhva Tiryagbhyam Multiplier
Spartan-2 (xc2s15-6cs144)	1877	1970	3599	3607
Spartan-2 (xc2s200-6fg256)	1878	1970	3599	3607
Spartan-2e (xc2s600e-7fg676)	1933	1969	3599	3607
Spartan-3 (xc3s4000l-4fg900)	797	504	1384	900
Spartan-3a (xc3s1400a-5fg676)	798	526	1378	936
Spartan-3e (xc3s1600e-5fg484)	798	525	1378	936
Virtex-2 (xc2v6000-6ff1517)	796	506	1384	902
Virtex-2p (xc2vpx70-7ff1704)	796	506	1384	902
Virtex-4 (xc4vlx200-11ff1513)	798	441	1384	775
Virtex-5(xc5vlx110t-1ff1136)	390	373	1456	617

Table 1 represents the Device utilization summary. In this the comparision parameters are number slice registers and number of slice LUTs. These two parameters are comparision with the Karatsuba multiplier and Urdhva Tiryagbhyam multiplier. The devices are changed from Spartan to Virtex the number slice

registers used by the multipliers are reduced and also number of slice LUTs are reduced. In this the designinig of the Urdhva Tiryagbhyam multiplier requires minimum amount of logic resources to perform double-precision floating point multiplication operations.

The fig. 17 represents the delay representation of Karatsuba and Urdhva Tiryagbhyam multipliers. The delay of the both the multipliers are measured in nano seconds. The delay of the Karatsuba multiplier is 18.139ns and delay of the Urdhva Tiryagbhyam multiplier is 15.034ns. The delay of the Urdhva Tiryagbhyam multiplier is less compared to the Karatsuaba multiplier so Urdhva Tiryagbhyam multiplier is fast in comparison with the Karatsuba multiplier.

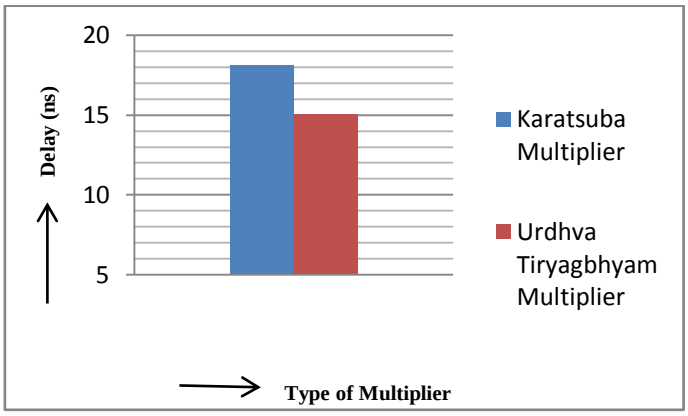


Fig. 17 Delay representation of Karatsuba and Urdhva Tiryagbhyam multipliers

The fig. 18 represents the Power consumption of the Karatsuba multiplier and Urdhva Tiryagbhyam multipliers. In this X-axis takes frequency in MHzs and Y- axis takes power in terms of watts. The frequency is from 10MHz to 600MHz and note down the power at the frequencies. In this fig. 18 observes the Karatsuba multiplier consumes more power in comparison with the Urdhva Tiryagbhyam multiplier.

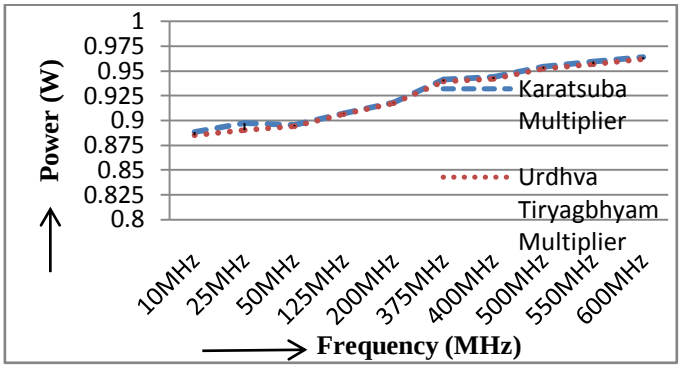


Fig. 18 Power consumption of Karatsuba and Urdhva Tiryagbhyam multipliers

V. CONCLUSION

The FPGA implementation of double-precision floating point multiplier using Urdhva Tiryagbhyam technique achieved the high speed and less area consumption compared to the Karatsuba multiplier and it is fully compatible with the binary interchange format of IEEE-754 standards.The design handles the various exceptional conditions like OVERFLOW and UNDERFLOW .

REFERENCES

[1] Manish Kumar Jaiswal, Ray C.C. Cheung "VLSI Implementation of Double-Precision Floating Multiplier Using Karatsuba Technique", Circuits syst signal process, Springer science business media, vol. 32, pp. 15-27, July 2013.

[2] Purna Ramesh addanki, Venkata Nagaratna Tilak Alapati and Mallikarjuna Prasad Avana "An FPGA Based High Speed IEEE-754 Double Precision Floating Point Adder/Subtractor and Multiplier Using Verilog" International Journal of Advanced Science and Technology, vol. 52, pp. 61-74, March 2013.

[3] M. al-Ashrafy, A. Salem, W. Anis, " An Efficient Implementation of Floating Point Multiplier", Saudi International Electronics, Communications and Photonics Conference(SIECPC), pp. 1-5, April 2011.

[4] Sandesh S. Saokar, R. M. Banakar, and Saroja Siddamal, "High Speed Signed Multiplier for Digital Signal Processing Applications", IEEE International Conference on Signal Processing, Computing and Control (ISPCC), pp. 1-6, March2012.

[5] Devika Jaina, Kabiraj Sethi, and Rutupama Pamda, "Vedic Mathematics Based Multiply Accumulate Unit ", International conference on computational intelligence and communication system, pp 754-757, 2011.

[6] Deena Dayalan, S.Deborah Priya, "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques", ACTEA IEEE pp 600-603, July 2009.

[7] P. Belanovic and M. Leeser, "A Library of Parameterized Floating-Point Modules and Their Use", in 12th International Conference on Field-Programmable Logic and Applications (FPL-02). London,UK: Springer Verilog, pp. 657-666, September 2002.

[8] Xilinx, Xilinx floating-point IP core. http://www.xilinx.com.

Author's Biography

RAMIREDDY VENKATA SURESH received his B.Tech degree from JNTUA Ananthapur Department of ECE. He is pursing M.Tech in Srinivasa Institute of Technology and Science, Ukkayapalli,Kadapa,AP

Mr. K. BALA is currently working as an associate professer In ECE Department, Srinivasa Instiute of Technology and science, Ukkayapalli, Kadapa, AP. He received his M.Tech from Sri Kottam Tulasi Reddy Memorial college of Engineering Kondair,Mahaboobnagar, AP.