Vol 05, Article 10470, October 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

DESIGN AND PERFORMANCE ANALYSIS OF DOUBLE-PRECISION FLOATING POINT MULTIPLIER USING URDHVA TIRYAGBHYAM SUTRA

Y SRINIVASA RAO[1], T SUBHASHINI[2], K RAMBABU[3]
P.G Student[1], Assistant Professor[2], Assistant Professor[3], Department of ECE, Gudlavalleru Engineering College,Gudlavalleru,Andhra Pradesh, INDIA
[1]srinivass.y25@gmail.com [2]subhashini.anagani@gmail.com [3]ksrk73@gmail.com

ABSTRACT

The arithmetic operations of floating point numbers in double- precision are addition, subtraction, multiplication and division. In these, the multiplication operation is a most preferably used arithmetic operation for designing various scientific and research applications. In general, the multiplication operation of floating point numbers in double-precision needs a very large mantissa multiplication, which is 53x53-bit mantissa multiplication to get the final result. This mantissa multiplication exits as a limit on both area and performance bounds of the multiplication operation. This paper presents a novel way to reduce this large multiplication. The proposed method in this paper allows using, less the amount of multiplication hardware compared to the traditional method. The multiplication is done by using Vedic sutra, which is Urdhva Tiryagbhyam. This multiplier is designing and targeting Virtex- 5 FPGA platform. The performance of the proposed multiplier is excellent with proficient utilization of resources. The design is completely compatible with the Standards of IEEE. The proposed multiplier's performance is comparison with Karatsuba multiplier..

Keywords:*floating point, FPGA, high performance, IEEE-754, multiplication, Vedic, Virtex-5, Urdhva Tiryagbhyam sutra*

1. INTRODUCTION

The binary number system is mostly used to representation of any digital system. The representations of real numbers in binary number system are called as floating point numbers. As defined by the IEEE-754 standard, the floating point numbers are divided into two categories, they are decimal interchange format and binary interchange format. The designing of floating point multipliers are very important in scientific computations, digital signal processing computations (like digital filters, FFT…) and numerical computations. As defined by the IEEE-754 standard, the floating point numbers are represented in two ways, they are Single precision (32-bit) and Double-precision (64-bit) format.

The notations of two different floating point number formats are shown in fig 1 and fig 2:

Floating point number representation in single-precision (32-bit) format:

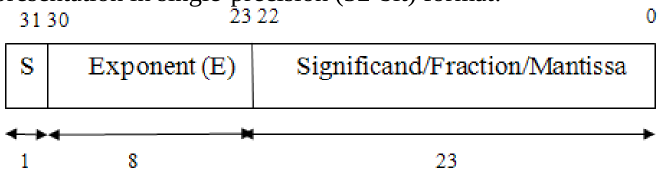

Fig.1. Floating point number representation of single-precision (32-bit).

Floating point number representation in double-precision (64-bit) format:

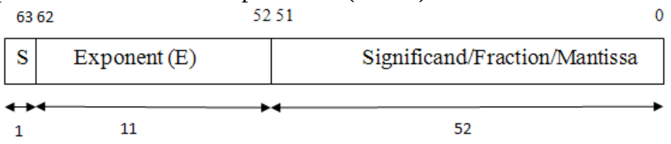

Fig.2. Floating point number representation of double-precision (64-bit) format.

The IEEE-754 standard defines various floating point arithmetic operations; they are addition, subtraction, multiplication, and division. These operations are very important for implementation of hardware of all processors. In this, the multiplication operation is crucial important and main operation for designing the processors and ASICs. Designing of any application, area is the main important constraint which should be minimized and provides greater performance. Our main motive is the required area of the multiplication should be minimized and efficient usage of resources then get the better performance.

There are two ways to improving the operational performance of floating point numbers. They are hardware implementation and algorithmic implementation. The floating point operations are also design by using the FPGA platforms.

FPGA (Field Programmable Gate Array) are preferably used to compute the high performance operations. The FPGAs are most suitable for designing a large set of applications, they require large amount of logic resources

means that the large area, large speed and intellectual property (IP) core. In now days, the FPGAs are used in tremendous applications like large floating point number computations, radar applications for tracking the accurate position of the object, communications, cryptography computations, digital image processing and digital signal processing applications. In now days, super-computers are designed by using the FPGAs. As a result, this work is primarily aimed for improved implementation of floating point number multiplication of double-precision (64-bit) operands on FPGA platform.
The mantissa multiplication is main important and critical part of the double precision floating point multiplication. This is the main important element of the area and performance bounds to designing the double-precision floating point multiplier. In double-precision floating point multiplier, the mantissa multiplication requires a 53-bits length of operands. These 53-bits of operands multiplication, the hardware implementation are very difficult and cost effective. In this work, the algorithm is proposed for the multiplication of 53 x 53 bit numbers and would be allows to using less amount of multiplication, achieving high performance at a relatively low hardware resources cost. Comparison of results has been done with Karatsuba algorithm and Vedic algorithm. The implementation is mainly concerned only normalized numbers, and designed for area and performance balanced. The design has been carried out using Xilinx ISE synthesis tool, ISIM simulator and Virtex-5(xc5vlx110t-1ff1136) speed grade-1 FPGA platform.
In this paper, the mantissa multiplication operations are carried out by using Urdhva Tiryagbhyam in binary. In section 2 represents proposed design approach, section 3 refers Implementation, section 4 refers results, section 5 refers conclusion, and section 6 refers future scope and references.

2. PROPOSED DESIGN APPROACH

The proposed design approach is taken from the Karatsuba Multiplication Technique. This multiplication technique, is divide the both 53-bits mantissas into three parts (including one hidden bit).

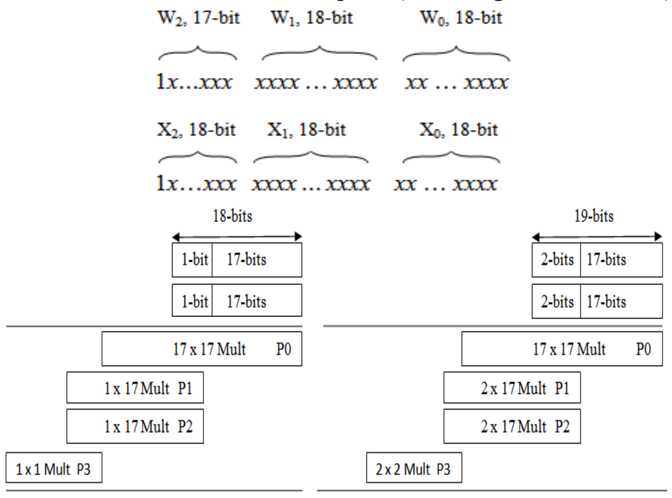

Fig.3. Representation of multipliers 18-bit and 19-bit

2.1 URDHVA TIRYAGBHYAM SUTRA FOR BINARY MULTIPLICATION

Vedic multiplication formulae (sutras) are commonly to desigining of Vedic multipliers. The Urdhva Tiryagbhyam sutra is widely used for the multiplication of two large decimal numbers. This same sutra can be used to perform multiplication on two large binary numbers. This proposed method is compatible to design digital hardware system.

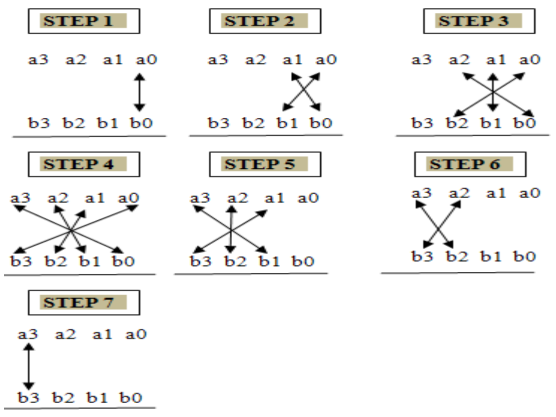

Fig.4. Multiplication of two 4-bit binary numbers using Urdhva Tiryagbhyam sutra

Vol 05, Article 10470, October 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

From using above method 18-bit and 19-bit multipliers can be designed. These multipliers are combined to get the 53 x 53 bit mantissa multiplication. By using this method partial products are concurrently added, there no need of special adders.

3. IMPLEMENTATION

The sign, exponent and mantissa operations of floating point multiplication are designing individually is represented in fig.5

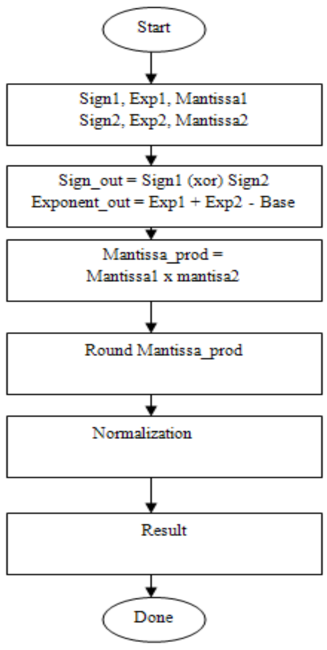

Fig.5. Flow Chart for Floating-Point Multiplication

3.1 SIGN AND EXPONENT OPERATION

The computation of sign and exponent bits are performed in simple way. The both operands of sign-bits are performed on logical XOR operation.

Sign_out = Sign1 (xor) Sign2

Addition of both input exponents (Exp1 and Exp2) and get the final exponent (Exponent_out) then subtracting the BASE (or) BIAS i.e.

Exponent_out=Exp1+ Exp2-1023

The BASE for the double-precision floating point number is calculated by (2^{11-1}-1) i.e. 1023.
The BASE of every floating point number is calculated by ($2^{\text{exponentbits-1}}$-1).

3.2 HANDLING OF EXCEPTIONAL CASES

As defined by the IEEE-754 standard, the floating point arithmetic operations support various exceptional cases. They are UNDERFLOW, OVERFLOW, ZERO, INFINITE and NaN (Not a Number). These will appear for all the floating point arithmetic operations. Thus, the most important operation is to determining the final resultant as per the standard and finding of all the exceptional conditions. All the exceptional conditions are executing in parallel with the standard of IEEE. For, example if one/both of the operands are infinite; the result will be displayed as INFINITY (with computed sign-bit). If any one of the operand is denormalized number, then the result will be displayed as ZERO (w.r.to sign-bit). If the resultant exponent is zero or below zero, the output will be shown as UNDERFLOW, and the exponent is above 11'h7fe (2046 in decimal), the output will be shown as OVERFLOW. These total operations are represented in Algorithm 1.

Algorithm 1Handling of Exceptional cases

```
If one of the operand infinite then
   EXPONENT_OUT = 11'h7ff;
   MANTISSA_OUT = 52'h0000000000000;
else if one operand isdenormalized then
   EXPONENT_OUT = 0;
   MANTISSA_OUT = 0;
else if exponent output ≤ZERO then
   UNDERFLOW = 1;
EXPONENT_OUT = 0;
   MANTISSA_OUT = 0;
```

```
else if exponent output > 11'h7fe then
OVERFLOW = 1;
   EXPONENT_OUT = 11'h7ff;
   MANTISSA_OUT = 0;
else
UNDERFLOW = 0;
   OVERFLOW = 0;
   EXPONENT_OUT = expected exponent;
   MANTISSA_OUT = expected mantissa;
end if
```

In addition, when one denormalized number other one is infinite number, the operation is shown as INVALID operation (Algorithm 2), and the result is NaN (Not a Number).

Algorithm 2 Invalid operation

```
If one of the operand is denormalized number and the another one is infinite number then
    EXPONENT_OUT = 11'h7ff;
    MANTISSA_OUT = 52'h8000000000000;
INVALID_OPERATION = 1;
Else
    INVALID_OPERATION = 0;
End if
```

3.3 NORMALIZATION AND ROUNDING

After multiplication of mantissa bits, the normalization is applied to final resultant for bringing back the 64-bit format i.e. sign-exponent-mantissa. Often the mantissa multiplication, the resultant has one additional bit is appear in the MSB position ahead of the decimal point. Likewise, in sometimes the similar condition will arise after rounding. The results obtained have to be set to the mandatory format. Each time an additional carry bit will be generated subsequent to the multiplication (or) rounding, the product is right-shifted for required shifts and the equivalent changes will be made in exponent also to get the normalized result.

Before rounding, the resultant of mantissa multiplication is 106-bit. Then rounding mode has been applied to get back the 106-bit product to 53-bit result. This paper, is going to implemente only round nearest mode précised by IEEE-754 standard. The remaining modes are also used depending on application.

4. RESULTS

By using the Urdhva Tiryagbhyam sutra, the floating point multiplier of double-precision (64-bit) was designed in Xilinx ISE 10.1 and simulated using ISIM simulator. The design is mapped on Virtex-5(xc5vlx-110t-1ff1136) FPGA. The fig: 6 show the simulation result of various test conditions of floating point multiplier of double-precision.

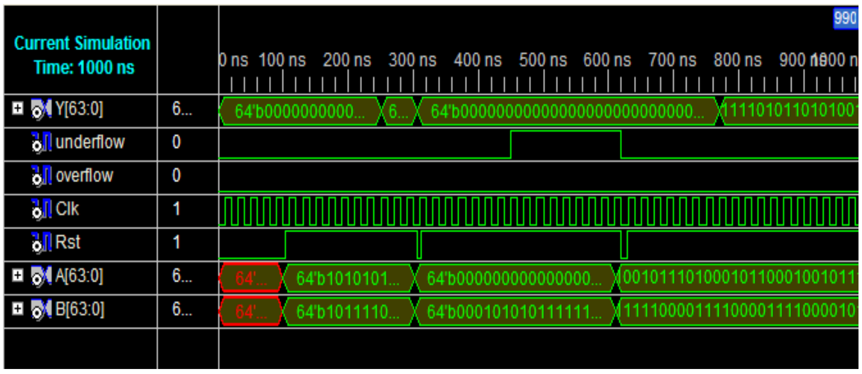

Fig.6 Urdhva Tiryagbhyam multiplier simulation result.

Table 1: Comparison table of two different Double-precision floating point multipliers

Slice Logic Utilization	Karatsuba Multiplier	Urdhva Tiryagbhyam Multiplier
No. of. Slice Registers	390	373
No. of. Slice LUTs	1456	617
No. of. Flip-Flop pairs	1613	684
No. of. IOs	196	196
No. of. IOBs	196	196

Table 2: Delay and Power Consumption of double precision floating point multipliers

S.NO	PARAMETERE OF COMPARISION	KARATSUBA MULTIPLIER	URDHVA TIRYAGBHYAM MULTIPLIER
1	DELAY (ns)	18.139	15.034
2	POWER CONSUMPTION (mw)	885	885

5. CONCLUSION

This paper concludes that, the proposed design is fully compatible with the binary interchange format of 754 IEEE standards. The design is targeted on Xilinx virtex-5 (xc5vlx-110t-1ff1136) FPGA. The design is achieved the better performance compared to Karatsuba multiplier. The design handles the various exceptional conditions like, overflow, underflow and the round nearest mode.

6. FUTURE SCOPE

The double precision floating multiplier using Urdhva Tiryagbhyam sutra can be implemented on ASIC flow to perform processor specified operations like square root, trigonometric functions.

REFERENCES

[1] Manish Kumar Jaiswal, Ray C.C. Cheung, *(*2013) "VLSI Implementation of Double-Precision Floating Multiplier Using Karatsuba Technique", *Circuits syst signal process, Springer science business media*,**32**,pp. 15-27.
[2] Purna Ramesh addanki, Venkata Nagaratna Tilak Alapati and Mallikarjuna Prasad Avana, *(*2013) March, "An FPGA Based High Speed IEEE-754 Double Precision Floating Point Adder'/Subtractor and Multiplier Using Verilog" *International Journal of Advanced Science and Technology*, **vol52**, pp. 61-74.
[3] M. al-Ashrafy, A. Salem, W. Anis, (2011) April,"An Efficient Implementation of Floating Point Multiplier", *Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, 24-26, pp. 1-5.
[4] Rudagi J. M. et. Al, (2011), "Design and Implementation of Efficient Multiplier Using Vedic Mathematics", *International Conference on Advances in Recent Technologies in Communication and Computing, pp 162-166.*
[5] Deena Dayalan, S.Deborah Priya, (2009), "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques", *ACTEA IEEE July 15-17*, pp 600-603.
[6] S. Venishetti, A. Akoglu, (2007), "A highly parallel FPGA based IEEE-754 compliant double-precision binary floating-point multiplication algorithm", in *International Conference on Field-Programmable Technology (ICFPT 2007)* pp. 145–152.