ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

Design and Analysis of Inexact Floating Point Multipliers

Jogadenu Vedavathi [1]
jogadenuvedavathi1993@gmail.com[1]

Mrs.T.Nagalaxmi[2]
tnagalaxmi@stanley.edu.in[2]

[1] PG Scholar, Dept of ECE, Stanley College of Engineering and Technology for Women, chapel road, abids, Ranga reddy,Telangana.

[2]Assistant Professor, Phd scholar,osmania university(OU),Stanley College of Engineering and Technology for Women, chapel road, abids, Ranga reddy,Telangana.

ABSTRACT: Power has become a key constraint in nano scale integrated circuit design due to the increasing demands for mobile computing and higher integration density in static and dynamic power dissipation. As an emerging computational paradigm, an inexact circuit offers a promising approach to significantly reduce both dynamic and static power dissipation for error-tolerant applications like image processing. For this, an inexact floating-point adder is implemented by approximately designing an exponent subtractor and mantissa adder. Related operations such as normalization and rounding are also dealt with in terms of inexact computing. An upper bound error analysis for the average case is presented to guide the inexact design; it shows that the inexact floating-point adder design is dependent on the application data range. The exact and inexact floating point adders has the disadvantage of occupying more area (8131.20 µ m2 and 5679.36µ m2). In order to reduce the area occupancy, the wallace tree multiplier and inexact floating point multiplier has been designed. These multipliers occupies the area of (5000 µ m2) and (3451.12 µ m2) respectively and improves the speed by (10%) and (5%) respectively. The Wallace tree multiplier and inexact floating point multiplier are designed and analyzed using Xilinx 13.2 for synthesis in verilog language.

Key Words—Inexact circuits, floating-point adder, low power, error analysis, high dynamic range image

I.INTRODUCTION

With advancement and development of innovative digital integrated circuits, power consumption has dramatically increased; power has become a key design constraint due to the high demand for mobile computing and higher integration density. Traditional designs apply fully accurate computing to all types of applications; however, error-tolerant applications involving human intervention (such as image processing) do not require full accuracy. So, it is possible to perform computation with inexact circuits; in these cases, inexact computing is an attractive approach to save power and area, while achieving improved performance compared to accurate designs. The arithmetic unit is the core of a processor, and its power largely determines the power of the whole processor. Recent research on inexact fixed-point adders has shown that inexact processing hardware with a relative error of 7.58 percent can be nearly 15 times more efficient in terms of speed, area and energy product than an accurate chip. Inexact chips are smaller, faster and consume less energy. Although fixed-point arithmetic circuits have been studied in terms of

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

inexact computing, floating-point (FP) arithmetic circuits are significantly more power hungry and they have not been fully considered for inexact computing. The FP format offers a high dynamic range for computationally intensive applications; FP adders and multipliers are commonly used in DSP systems However, its application to embedded DSP systems is limited due to the high power consumption. A low power design of an FP multiplier was investigated by Tong et al.; this design involves the truncation of hardware and a reduction of the bit width representation of the FP data. A probabilistic FP multiplier was proposed by Gupta et al. mostly as an energy efficient design. A lightweight FP design flow using bit-width optimization was proposed for low power signal processing applications. Low precision FP numbers have also been used for MP3 decoding to reduce memory utilization and power consumption. However, to the best of the authors' knowledge, there has been no research to date on an inexact FP adder design. In this paper, adder designs are studied as a starting point for inexact FP arithmetic; several inexact adder designs are proposed and assessed for application to high dynamic range images. The upper bound error due to the inexact design is analyzed for the average case to guide the design of inexact FP adders. A subjective visual difference predictor metric is used to measure the results of image addition; moreover, a procedure is introduced for designing inexact FP arithmetic circuits.

Fixed Point and Floating Point Representations

Every real number has an integer part and a fraction part; a radix point is used to differentiate between them. The number of binary digits assigned to the integer part may be different to the number of digits assigned to the fractional part. A generic binary representation with decimal conversion is shown in Figure 1.

	Integer part	Binary point	Fraction part
Binary	2^3 2^2 2^1 2^0	---	2^{-1} 2^{-2} 2^{-3}
Decimal	8 4 2 1	----	1/2 1/4 1/8

Figure1: Binary representation and conversion to decimal of a numeric

Basic Format

There are two basic formats described in IEEE 754 format, double-precision using 64-bits and single-precision using 32-bits. Table 1 shows the comparison between the important aspects of the two representations.

Format	Precision(P)	Emax	Emin	Exponent Bias	Exponent Width	Format Width
Single	24	+127	-126	127	8	32
Double	53	+1023	-1022	1023	11	64

Table 1: Single and double precision format summary

To evaluate different adder algorithms, we are only interested in single precision format. Single-precision format uses 1-bit for sign bit, 8-bits for exponent and 23-bits to represent the fraction as shown in Figure 2.

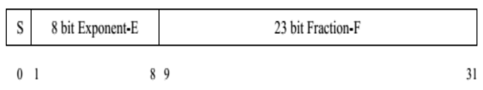


Figure 2: IEEE 754 single precision format

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

The single- precision floating-point number is calculated as (-1)S× 1.F × 2(E-127). The sign bit is either 0 for non-negative number or 1 for negative numbers. The exponent field represents both positive and negative exponents. To do this, a bias is added to the actual exponent. For IEEE single-precision format, this value is 127, for example, a stored value of 200 indicates an exponent of (200-127), or 73. The mantissa or significand is composed of an implicit leading bit and the fraction bits, and represents the precision bits of the number. Exponent values (hexadecimal)of 0xFF and 0x00 are reserved to encode special numbers such as zero, de normalized numbers, infinity, and NaNs.

The mapping from an encoding of a single-precision floating-point number to the number's value is summarized in Table 2.

sign	exponent	fraction	value	description
s	0*FF	0*00000 000	$(-1)^S$ ∞	Infinity
s	0*FF	F -/-0	NaN	Not a Number
s	0*00	0*00000 000	0	Zero
s	0*00	F -/-0	$(-1)^S$ $*0.F*2^{(E-126)}$	Denormalized Number
s	0*00<E< FF	F	$(-1)^S$ $*0.F*2^{(E-127)}$	Normalized Number

Table 2: IEEE 754 single precision floating-point encoding

Standard Floating Point Addition Algorithm

This section will review the standard floating point algorithm architecture, and the hardware modules designed as part of this algorithm, including their function, structure, and use. The standard architecture is the baseline algorithm for floating-point addition in any kind of hardware and software design.

II.LITERATURE SURVEY

Although the inexact design of fixed-point adders has been extensively studied, little research has been conducted on inexact floating-point arithmetic design. A low power design of a floating-point multiplier was investigated by Tong et al. which involves truncating hardware; the rounding unit was found to require almost half of the hardware of an exact floating-point multiplier. Therefore, the rounding unit is a candidate for removal to save power, similar to an inexact design. A probabilistic floating-point multiplier was proposed by Gupta et al. as an energy efficient design. However, to the best of the authors' knowledge, there has been no research to date on an inexact floating-point adder design, which has a more complex structure than a floating-point multiplier. An inherent problem of binary floating-point arithmetic used in financial calculations is that most decimal floating point numbers cannot be represented exactly in binary floating-point formats, and errors that are not acceptable may occur in the course of the computation. Decimal floating-point arithmetic addresses this problem, but a degradation in performance will occur compared to binary floating-point operations implemented in hardware. Despite its performance disadvantage, decimal floating-point arithmetic is required by certain applications that need results identical to those calculated by hand. This is true for currency conversion, banking, billing, and other financial applications. Sometimes, these requirements are mandated by law; other times, they are necessary to avoid large accounting discrepancies. Because of the importance of this problem a number of decimal solutions exist, both hardware and software. Software solutions include C#, COBOL, and XML, which provide decimal operations and data types. Also, Java and C/C++ both have packages, called Big Decimal and dec Number, respectively. Hardware solutions were more prominent earlier in the computer age with the ENIAC and UNIVAC. However, more recent examples include the CADAC, IBM's z900 and z9 architectures, and numerous other proposed hardware implementations. More hardware examples can be found, and a more in-depth discussion is found in Wang's work.

Design tradeoff analysis of floating-point adder in FPGAs[3]: Field Programmable Gate Arrays (FPGA) are increasingly being used to design high end computationally intense microprocessors capable of handling both fixed and floating point mathematical operations. Addition is the most complex operation in a floating-point unit and offers major delay while taking significant area. Over the years, the VLSI community has developed many floating-point adder algorithms mainly aimed to reduce the overall latency. An efficient design of floating-point adder onto an FPGA offers major area and performance overheads. With the recent advancement in FPGA architecture and area density, latency has been the main focus of attention in order to improve performance. Our research was oriented towards studying and implementing standard, Leading One Predictor (LOP), and far and close data-path floating-point addition algorithms. Each algorithm has complex sub-operations which lead significantly to overall latency of the design.

Design Of inexact floating-Point adders[4]:

The inexact design of an FP adder originates at an architectural level. It consists of designing both the mantissa adder and exponent subtractor by using approximate fixed-point adders. At the same time, related logic including the normalizer and the rounder should also be considered according to the inexact mantissa and exponent parts. The circuit level inexact designs are discussed in detail in the following sections.

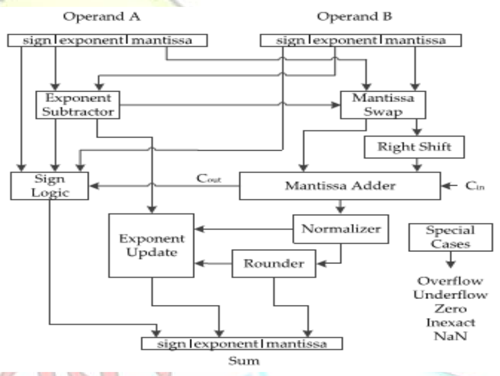


Fig.3. The accurate FP adder architecture

Exponent Subtractor

The exponent subtractor is used for exponent comparison and can be implemented as an adder. An inexact fixed-point adder has been extensively studied and can be used in the exponent adder; inexact adders such as lower-part-OR adders (LOA), approximate mirror adders, approximate XOR/XNOR-based adders, and equal segmentation adders can be found in the literature. For a fast FP adder, a revised LOA adder is used, because it significantly reduces the critical path by ignoring the lower carry bits. A k-bit LOA consists of two parts, i.e., an m-bit exact adder and an n-bit inexact adder. The m-bit adder is used for the m most significant bits of the sum, while then-bit adder consists of OR gates to compute the addition of the least significant n bits (i.e., the lower n-bit adder is an array of n two-input OR gates). In the original LOA design, an additional AND gate is used for generating the most significant carry bit of then-bit adder; in this work, all carry bits in then-bit inexact adder are ignored to further reduce the critical path. The exponent is dominant in the FP format, because it determines the dynamic range. The approximate design of the exponent subtractor must be carefully considered due to its importance in the number format. The results of the addition are significantly affected by applying an approximate design to only a few of the least significant bits of the exponent subtractor under a small data range

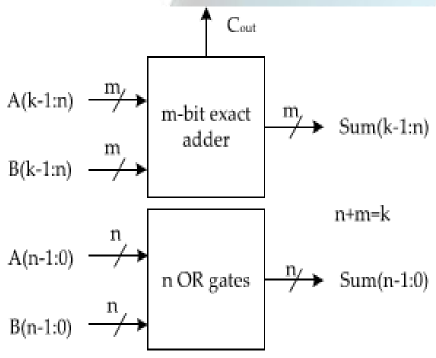


Fig.4.The revised LOA adder structure.

The revised LOA adder can also be used in the mantissa adder for an inexact design. Compared to an exponent subtractor, the mantissa adder offers a larger design space for inexact design, because the number of bits in the mantissa adder is significantly larger than the exponent subtractor. As shown in Table 1, the number of mantissa bits is larger than the number of exponent bits. For the IEEE single precision format, the exponent subtractor is an 8-bit adder, while the mantissa adder is a 25-bit adder (for two 24-bit significances). Furthermore, the inexact design in the mantissa adder has a lower impact on the error than its exponent counterpart in the lower data range, because the mantissa part is less significant than the exponent part. Therefore, an inexact design of a mantissa adder is more appropriate. A detailed analysis of errors introduced by each part is further discussed in the next section. [3] proposed a system in which the complex parallelism technique is used to involve the processing of Substitution Byte, Shift Row, Mix Column and Add Round Key. Using S- Box complex parallelism, the original text is converted into cipher text. From that, we have achieved a 96% energy efficiency in Complex Parallelism Encryption technique and recovering the delay 232 ns. The complex parallelism that merge with parallel mix column and the one task one processor techniques are used. In future, Complex Parallelism single loop technique is used for recovering the original message.

Normalizer

Normalization is required to ensure that the addition results fall in the correct range; the sum or difference may be too small and a multi-bit left shift process may be required. A reduction of the exponent is also necessary. The normalization is performed by a leading zeros counter that determines the required number of left shifts. As the mantissa adder is already not exact for then least significant bits, the detection of the leading zeros can also be simplified in the inexact design, i.e., approximate leading zero counting logic can be used.

Rounder

A rounding mode is required to accommodate the inexact number that an FP format can represent. A proper rounding maintains three extra bits (i.e., guard bit, round bit and sticky bit). The adder may

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

require a further normalization and exponent adjustment after the rounding step, therefore the hardware for rounding is significant. However, it does not affect the results of the inexact addition as the lower significant n bits are already inexact. Therefore, rounding can be ignored in the inexact design of an FP adder.

Overall Inexact FP Adder Architecture

Based on the previous discussion, an inexact FP adder can be designed by using approximate adders in the exponent subtractor and mantissa adders, an approximate leading zero counter in the normalizer and by ignoring the rounder. The inexact FP adder architecture is shown in Fig. 5.

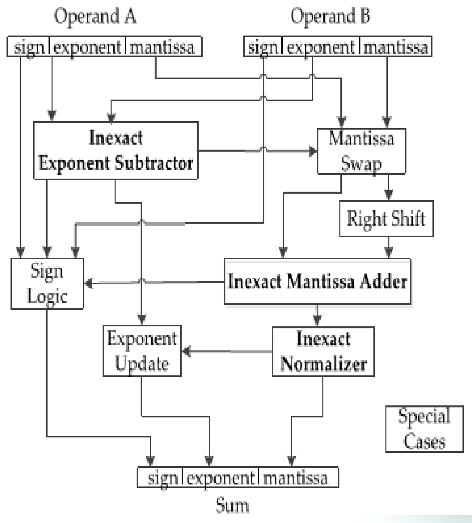


Fig5.The inexact FP adder architecture.

III.PROPOSED SYSTEM

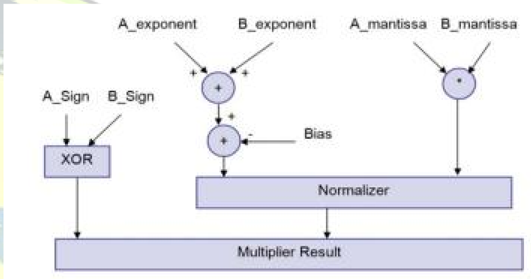
In floating point multipliers the area occupied by the rounding will be reduced by dadda multiplier. This is one of the best multiplier used this reduces the occupancy of area and power because it contains only three stepsProposed Dadda multiplier has 3 steps:

1. Perform Multiplication operation and get partial product matrix.

2. Reduce the number of partial products to two layers of full and half adders.

3. Group them into two numbers, and add them with a conventional adder.

Floating point Multiplier:



Two floating point numbers the following is done:

1. Multiplying the significand; i.e. (1.M1*1.M2)
2. Placing the decimal point in the result
3. Adding the exponents; i.e. (E1 + E2 – Bias)
4. Obtaining the sign; i.e. s1 xor s2
5. Normalizing the result; i.e. obtaining 1 at the MSB of the results' significand
6. Rounding the result to fit in the available bits
7. Checking for underflow/overflow occurrence

A. Sign bit calculation

Multiplying two numbers results in a negative sign number iff one of the multiplied numbers is of a negative value. By the aid of a truth table we find that this can be obtained by XORing the sign of two inputs.

B. Unsigned Adder (for exponent addition)

This unsigned adder is responsible for adding the exponent of the first input to the exponent of the second input and subtracting the Bias (127)

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

from the addition result (i.e. A_exponent + B_exponent - Bias). The result of this stage is called the intermediate exponent.. An 8-bit ripple carry adder is used to add the two input exponents. As shown in Fig. 3 a ripple carry adder is a chain of cascaded full adders and one half adder; each full adder has three inputs (A, B, Ci) and two outputs (S, Co). The carry out (Co) of each adder is fed to the next full adder (i.e. each carry bit "ripples" to the next full adder).

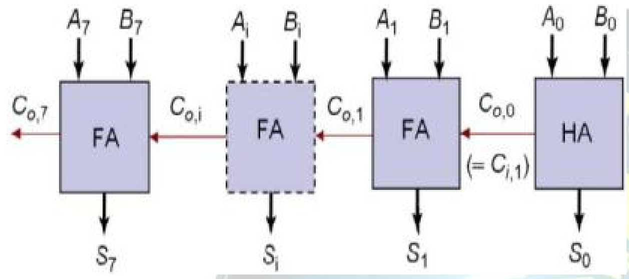


Fig 6: Ripple Carry Adder

The addition process produces an 8 bit sum (S7 to S0) and a carry bit (Co,7). These bits are concatenated to form a 9 bit addition result (S8 to S0) from which the Bias is subtracted. The Bias is subtracted using an array of ripple borrow subtractors.

A normal subtractor has three inputs (minuend (S), subtrahend (T), Borrow in (Bi)) and two outputs (Difference (R), Borrow out (Bo)).The subtractor logic can be optimized if one of its inputs is a constant value which is our case, where the Bias is constant (127|10 = 001111111|2). Table I shows the truth table for a 1-bit subtractor with the input T equal to 1 which we will call "one subtractor (OS)"

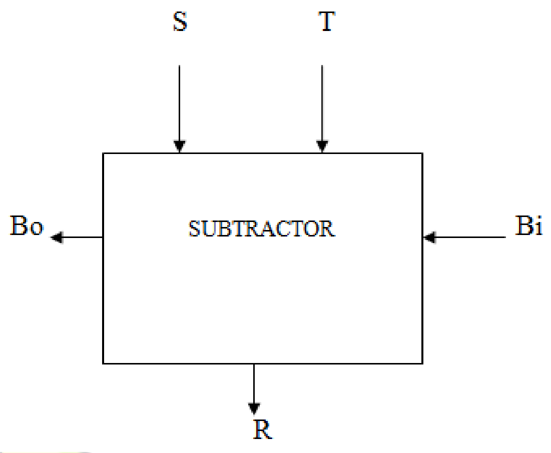


TABLE I. 1-BIT SUBTRACTOR WITH THE INPUT T = 1

S	T	Bi	Difference(R)	Bo
0	1	0	1	1
1	1	0	0	0
0	1	1	0	1
1	1	1	1	1

The Boolean equations (2) and (3) represent this subtractor:

$$\text{Difference(R)} = \overline{S \oplus B} \quad (2)$$

$$\text{Borrow(Bo)} = \bar{S} + Bi \quad (3)$$

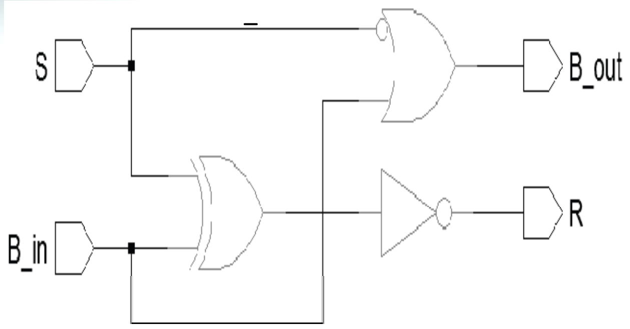


Fig 7:1-bit subtractor with the input T = 1

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

Table II shows the truth table for a 1-bit subtractor with the input T equal to 0 which we will call "zero subtractor (ZS)" TABLE II. 1-BIT SUBTRACTOR WITH THE INPUT T = 0

S	T	Bi	Difference(R)	Bo
0	0	0	0	0
1	0	0	1	0
0	0	1	1	1
1	0	1	0	0

The Boolean equations (4) and (5) represent this subtractor:

$$\text{Difference(R)} = S \oplus Bi \qquad (4)$$

$$\text{Borrow(Bo)} = \overline{S}\,.Bi \qquad (5)$$

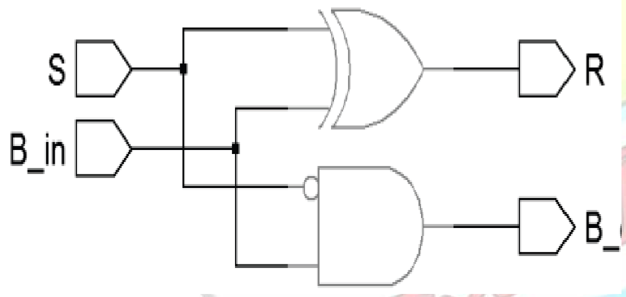


Fig 8:1-bit subtractor with the input T = 0

Fig. 6 shows the Bias subtractor which is a chain of 7 one subtractors (OS) followed by 2 zero subtractors (ZS); the borrow output of each subtractor is fed to the next subtractor. If an underflow occurs then Eresult < 0 and the number is out of the IEEE 754 single precision normalized numbers range; in this case the output is signaled to 0 and an underflow flag is asserted.

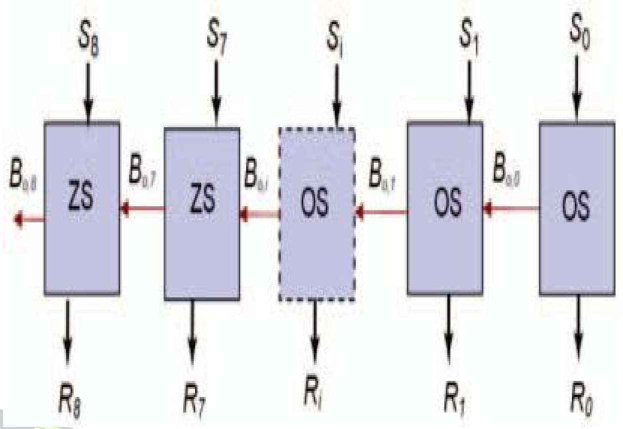


Fig 9:Ripple Borrow Subtractor

C. Unsigned Multiplier (for significand multiplication)

This unit is responsible for multiplying the unsigned significand and placing the decimal point in the multiplication product. The result of significand multiplication will be called the intermediate product (IP). The unsigned significand multiplication is done on 24 bit. Multiplier performance should be taken into consideration so as not to affect the whole multiplier's performance. A 24x24 bit carry save multiplier architecture is used as it has a moderate speed with a simple
architecture. In the carry save multiplier, the carry bits are passed diagonally downwards (i.e. the carry bit is propagated to the next stage). Partial products are made by ANDing the inputs together and passing them to the appropriate adder.
Carry save multiplier has three main stages:

1- The first stage is an array of half adders.
2- The middle stages are arrays of full adders. The number of middle stages is equal to the significand size minus two.
3- The last stage is an array of ripple carry adders. This stage is called the vector merging stage.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

The number of adders (Half adders and Full adders) in each stage is equal to the significand size minus one. For example, a 4x4 carry save multiplier is shown in Fig. 7 and it has the following stages:

1- The first stage consists of three half adders.

2- Two middle stages; each consists of three full adders.

3- The vector merging stage consists of one half adder and two full adders.

The decimal point is between bits 45 and 46 in the significand multiplier result. The multiplication time taken by the carry save multiplier is determined by its critical path. The critical path starts at the AND gate of the first partial products (i.e. a1b0 and a0b1), passes through the carry logic of the first half adder and the carry logic of the first full adder of the middle stages, then passes through all the vector merging adders. The critical path is marked in bold in F

D. Normalizer

The result of the significand multiplication (intermediate product) must be normalized to have a leading '1' just to the left of the decimal point (i.e. in the bit 46 in the intermediate product). Since the inputs are normalized numbers then the intermediate product has the leading one at bit 46 or 47

1- If the leading one is at bit 46 (i.e. to the left of the decimal point) then the intermediate product is already a normalized number and no shift is needed.

2- If the leading one is at bit 47 then the intermediate product is shifted to the right and the exponent is incremented by 1.

.

IV. UNDERFLOW/OVERFLOW DETECTION

Overflow/underflow means that the result's exponent is too large/small to be represented in the exponent field. The exponent of the result must be 8 bits in size, and must be between 1 and 254 otherwise the value is not a normalized one. An overflow may occur while adding the two exponents or during normalization. Overflow due to exponent addition may be compensated during subtraction of the bias; resulting in a normal output value (normal operation). An underflow may

occur while subtracting the bias to form the intermediate exponent. If the intermediate exponent < 0 then it's an underflow that can never be compensated; if the intermediate exponent = 0 then it's an underflow that may be compensated during normalization by adding 1 to it.

When an overflow occurs an overflow flag signal goes high and the result turns to ±Infinity (sign determined according to the sign of the floating point multiplier inputs). When an underflow occurs an underflow flag signal goes high and the result turns to ±Zero (sign determined according to the sign of the floating point multiplier inputs). Denormalized numbers are signaled to Zero with the appropriate sign calculated from the inputs and an underflow flag is raised. Assume that E1 and E2 are the exponents of the two numbers A and B respectively; the result's exponent is calculated by

$$E_{result} = E1 + E2 - 127$$

E1 and E2 can have the values from 1 to 254; resulting in Eresult having values from -125 (2-127) to 381 (508-127); but for normalized numbers, Eresult can only have the values from 1 to 254. Table III summarizes the Eresult different values and the effect of normalization on it.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

Eresult	Category	Comments
-125 ≤ Eresult < 0	Underflow	Can't be compensated during normalization
Eresult = 0	Zero	May turn to normalized numbe duringnormalization (by adding
1 < Eresult < 254	Normalized number	May result in overflow during normalization
255 ≤ Eresult	Overflow	Can't be compensated

IV.RESULTS

The floating point adder and multiplier Verilog HDL Modules have successfully simulated and verified using Xilinx ise13.2

SIMULATION RESULT OF INEXACT FLOATING POINT ADDER:

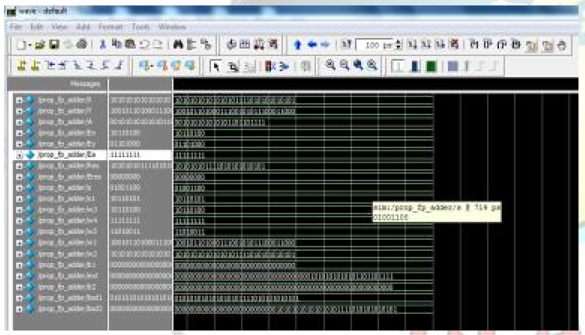


Fig 12 simulation results of inexact floating point adder

Clk=1,rst=0,input=11100000000011111,y=00011

RTL SCHEMATIC:

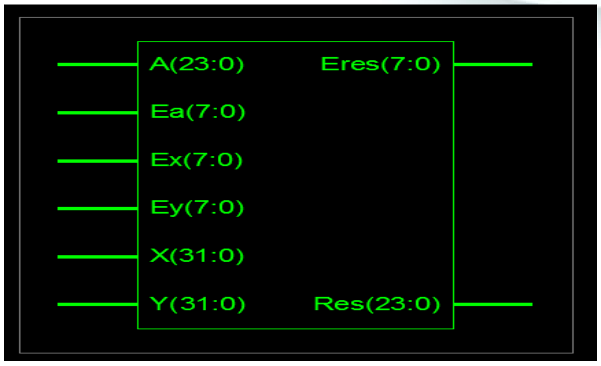


Fig 13 RTL schematic of inexact floating point adder

TECHNOLOGY SCHEMATIC:

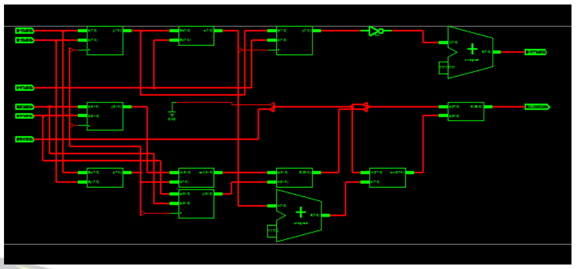
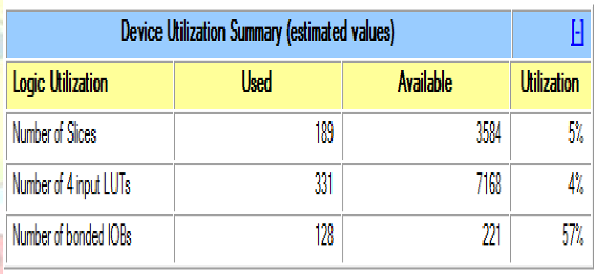


Fig 14 Technology schematic diagram of inexact floating point adder

DESIGN SUMMARY:



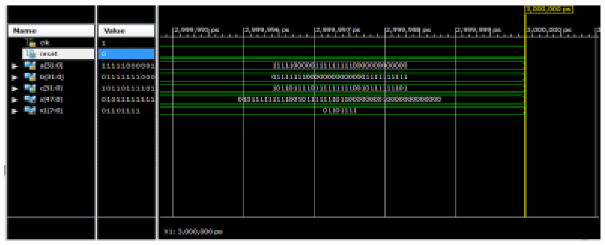
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	189	3584	5%
Number of 4 input LUTs	331	7168	4%
Number of bonded IOBs	128	221	57%

For the given inputs the number of slices occupied is 189 out of 3584 and number of 4 input LUTs is 331 out of 7168 and number of bonded IOBs is 128 out of 221

Fig 15 Design summary of inexact floating point adder

WALLACE TREE MULTIPLIER :

SIMULATION RESULTS:



ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

Fig 16 simulation results of Wallace tree multiplier.

When clk=1-0 at 100ps and reset=0 and intermediate results are stored in a1, a2 in full adders.

a=11111100001111110000000000000000

b= 01111100000000011111111111111111

c= 10110111011111110010111111111011

RTL SCHEMATIC:

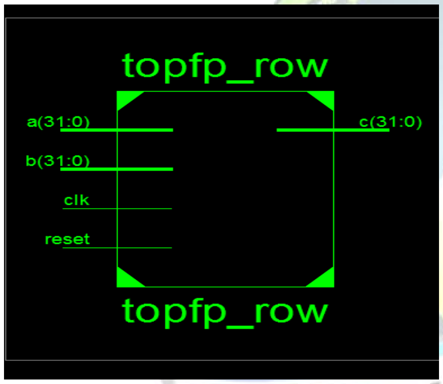


Fig 17 RTL schematic of Wallace tree multiplier

TECHNOLOGY SCHEMATIC:

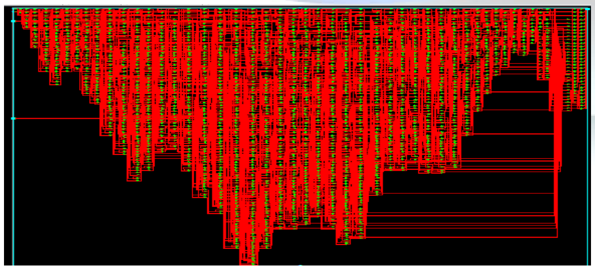
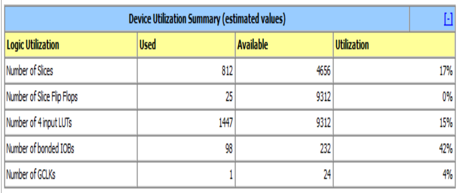


Fig 18 Technology schematic of Wallace tree multiplier

DESIGN SUMMARY:



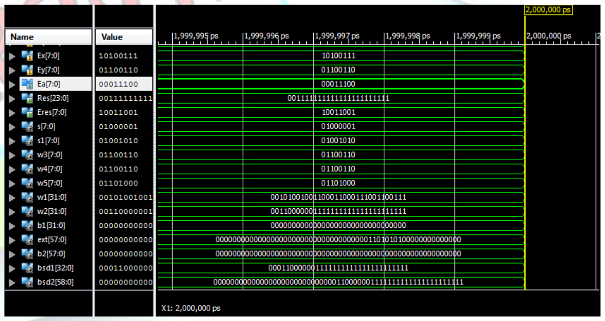
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	812	4656	17%
Number of Slice Flip Flops	25	9312	0%
Number of 4 input LUTs	1447	9312	15%
Number of bonded IOBs	98	232	42%
Number of GCLKs	1	24	4%

For the given input number of slices is 812 out of 4656 and the number of slice flip flops is 25 out of 9312 and the number of 4 input LUTs is 1447 out of 9312 and the number of bonded IOBs is 98 out of 232 and the number of GCLKs 1 out of 24.

Fig 19 Design summary of Wallace tree multiplier.

INEXACT FLOATING-POINT MULTIPLIER:

SIMULATION RESULTS:



```
Name          Value
Ex[7:0]       10100111     10100111
Ey[7:0]       01100110     01100110
Ea[7:0]       00011100     00011100
Res[23:0]     00111111111  001111111111111111111111
Eres[7:0]     10011001     10011001
s[7:0]        01000001     01000001
s1[7:0]       01001010     01001010
w3[7:0]       01100110     01100110
w4[7:0]       01100110     01100110
w5[7:0]       01101000     01101000
w1[31:0]      00101001001  00101001001100011000111001100111
w2[31:0]      00110000001  00110000001111111111111111111111
b1[31:0]      00000000000  00000000000000000000000000000000
ext[57:0]     00000000000  0000000000000000000000000000000000000011010101000000000000000
b2[57:0]      00000000000  0000000000000000000000000000000000000000000000000000000000
bsd1[32:0]    00011000000  000110000001111111111111111111111
bsd2[58:0]    00000000000  00000000000000000000000000000011000000111111111111111111111
X1: 2,000,000 ps
```

Clk=1,rst=0,input=1111111000000000,y=11000

Fig 20 simulation results of inexact floating point adder

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

RTL SCHEMATIC:

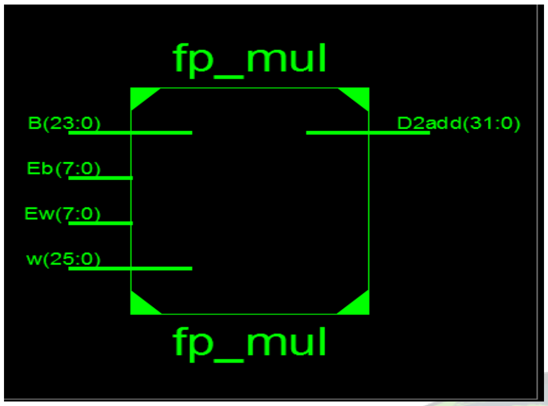


Fig 21 RTL schematic of inexact floating point multiplier

TECHNOLOGY SCHEMATIC:

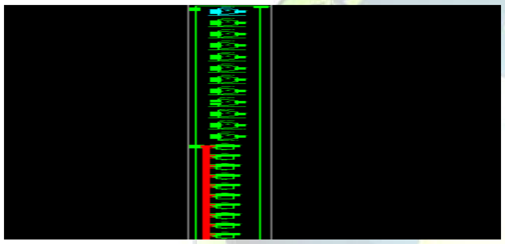
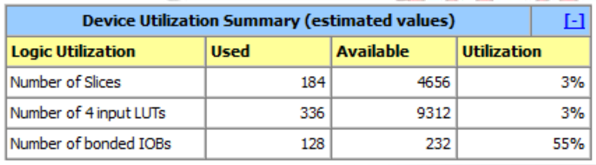


Fig 22 Technology schematic of inexact floating point multiplier.

DESIGN SUMMARY:



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	184	4656	3%
Number of 4 input LUTs	336	9312	3%
Number of bonded IOBs	128	232	55%

For the given inputs the number of slices occupied is 184 out of 4656 and number of 4 input LUTs is 336 out of 9312 and number of bonded IOBs is 128 out of 232

Fig 23 Design summary of inexact floating point multiplier

CONCLUSION

Dadda inexact multiplier is designed in the project. Due to the drawback of more space occupancy in exact and inexact adders the Wallace tree multiplier and floating point multiplier has been designed. The Wallace tree multiplier is also having drawbacks of area occupancy inorder to overcome this the dadda exact and inexact floating point multiplier has been designed. The dadda inexact multiplier, which occupies less space and area and other parameters has been compared with floating point adder and Wallace tree multiplier. The output of dadda inexact floating point multiplier results in less power consumption as 0.1999mW and area occupied is 2545.36µ m2 and the delay factor is 5.76ns.

FUTURE SCOPE

In the future work, pipelined double precision floating point multiplier operates on 64-bit operands. It can be designed for quadruple precision floating point multiplier operates on 128-bit operands to enhance precision. Future work can also further extend to increase the more speed and reduce area. It can be extended to have more mathematical operations like adder/ subtractor, divider and exponential functions.

REFERENCES

[1] K. Palem and A. Lingamneni, “Ten years of building broken chips: The physics and engineering of inexact computing,”ACM Trans. Embedded Comput. Syst., vol. 12, no. 2, article 87, 2013.

[2] A. Lingamneni, K. Muntimadugu, C. Enz, R. Karp, K. Palem, and C. Piguet, “Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling,” inProc. ACM Int. Conf. Comput. Frontiers, 2012, pp. 3–12.

[3] Christo Ananth, H.Anusuya Baby, “Encryption and Decryption in Complex

Parallelism", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 3, Issue 3, March 2014,pp 790-795

[4] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in Proc. Int. Symp. Low Power Electron. Des., 2011, pp. 1–3.

Authors Biography:

1.J.VEDAVATHI received B-Tech degree in Electronic and Communication Engineering from Turbomachinery Institute Of Science and Technology Engineering College, under JNTU Hyderabad, India. I am pursuing Masters in Embedded Systems from Stanley College of Engineering and Technology for Woman, Hyderabad, India.

e-mail id:jogadenuvedavathi1993@gmail.com

2.T.NAGALAXMI working as an Assistant professor, Phd scholar, osmania university(OU), STANLEY COLLEGE OF ENGINEERING AND TECHNOLOGY FOR WOMEN, Hyderabad till date.

She worked as Asst.Prof in Vidya Jyothi Engineering and Tech. And pursued M.TECH (Embedded systems), affiliated college by JNTUH. She is having nine years of teaching experience. Her areas of research interests are embedded systems, VLSI, embedded and real time systems, digital signal processing and architectures, Microprocessor & Micro controller, data communication systems.