International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 01 | Jan -2017 www.irjet.net p-ISSN: 2395-0072

SURVEY ON TWO-TERM DOT PRODUCT OF MULTIPLIER USING FLOATING POINT

[1]MukeshKrishna. R, [2]MohanaPriya.S, [3]ManickaVasagam.P

Department Electrical and Electronics Engineering

Students, Dr. Mahalingam College of Engineering and Technology, Udumalai road, Pollachi.

ABSTRACT-- ***The survey is based on the Floating Point in two-term Dot-Product of multiplier referred as discrete design. Floating Point is a wide variety for increasing accuracy, high speed, high performance and reducing delay, area and power consumption. This application of floating point is used for algorithms of Digital Signal Processing and Graphics. Many floating point application is to reduce area, from the survey the fused floating point gives better performance using both the single precision and the double precision in multiplication, addition and subtraction. The scientific notations sign bit, mantissa and exponent are used. The real numbers are divided into two, fixed component of significant range (lack of dynamic range) and exponential component in floating point (largest dynamic range). By converting from 24-bit to 48-bit fused floating point used to normalize the exponent part and rounding operation for latency reducing and the results are executed in the verilog hardware description Language.***

Index Terms-- Dot-Product Unit, fused floating point operations, normalization, rounding operations, latency, VHDL.

I.INTRODUCTION

The demand of floating point multiplier is more in Three-Dimensional (3D) array and also used in graphics and image processing. Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and Butterfly operations are needed floating point numbers [1]. Due to output data size is twice larger than the input data size so complexity, area and time are consumed by the multipliers. The best design challenge to get high speed working is in Field Programmable Gate Array (FPGA). The floating point shows the base, the location, the precision and it normalized or not. There are many models for multiplication floating point. Precision is the main role in floating point. We deal with both single and double precision floating point. The main significant of floating point number are (Sign bit * Mantissa * Base ^{Exponent}). The single precision has 24-bits which contain 0 to 31, left to right and double precision has 64-bits which contain 0 to 63, left to right [2]. The difference of these two precision is data, the double precision has twice the data of RAM, Cache and Band Width and reduce the performance. The result of sign bit by XOR and carry save adder used for two exponent components.

1.1 Data formats for single and double precision:

Sign bit	Biased Exponent	Unsigned fraction
S	8 bit – E	23 – bits P

a) Single Precision Data Format

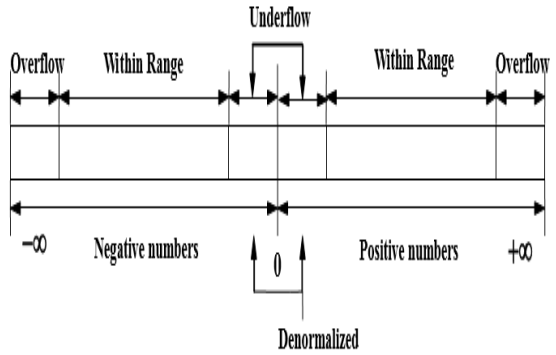
Sign bit	Biased Exponent	Unsigned fraction
S	11 bit – E	53– bits P

b) Double Precision Data Format

1.2 Format Parameters:

The implementation of hardware and software has basic IEEE format. In the standard IEEE format the floating points are in binary number. The binary floating point numbers are single precision and double precision. The single precision contains 32 bits and the precision which adds the fraction and hidden bits 23+1, exponent bit 8 is used. The maximum and the minimum values from +127 to -126. For the double precision, contain 64 bits and the precision which has 52+1, exponent bit is 11 is used. The maximum and the minimum values from -126 to -1022. For quadruple precision, hidden bits are 112+ 1 and the maximum and the minimum value from -16382 to + 16383 [3].

1.3 Representation of the floating point:



Denormalized:

Exponent part contain zeros and fraction or significand contain non-zeros denormalized is taken. Denormalized occur in zeros and lower normalized range [3]. Zero is a special value for exponent field all zeros and fraction zeroes.

Overflow:

Overflow occur limited range in smallest value and higher range in highest value. It indicate the range when reach extreme value. It doesn't show the indication when one operand is infinity. It must have the exact range [4]. When the result reaches extreme range, bias should adjust and a NaN is delivered instead.

Underflow:

Underflow takes place when floating point is smaller than the smallest value. It may be negative or positive exponent from -128 to 127, when lesser than -128 underflow occur. The result may be zero or denormal [4]. There is loss of accuracy after the denormalized numbers. Under flow adjust the result from overflow delivery.

Infinity:

The value of -infinity and +infinity used in exponent 0s and 1s. Sign bit for positive 0 and negative 1 are used. It denotes infinity as special value for operations to continue past overflow situations. It used undefined operations [5].

Not a Number:

It is an invalid value when does not show the real number representation. The exponent has 1s and the fraction has non-zeroes are taken in NaNs [6].

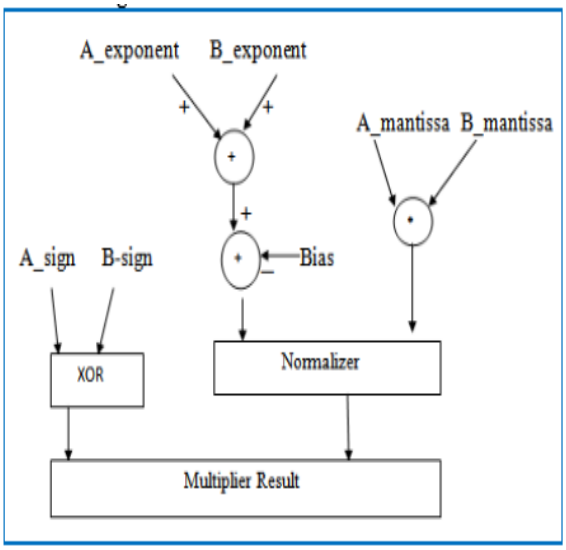
II. APPROACH

According to IEEE 754 2008 supports the floating point multiplier which has efficient carry saver. For the high performance of multiplier, pipe lining stages are used to increase operating frequency multiplier [5]. Here two approaches are seen for dot product unit, floating point adder and multiplier. The floating point 32 bit is in parallel operation is used for many application and minimizing the silicon area. A*(B+C) shows the parallel operation but in the fused floating point 3 cycles are reduced from 8 cycles instead of 2 cycles in 8 cycle. This greatly improves the performance. By rounding operation the addition cycle eliminated and directly performs the multiplication [6]. For the exact result of floating point operation rounding is needed. The different modes used are

Rounding Mode	Encoding	Unrounded	Rounded
Nearest Even	00	3.4	3
Zero	01	5.6	6
Positive Infinity	10	3.5	4
Negative Infinity	11	2.5	2

III. FLOATING POINT MULTIPLICATION OPERATION

The fused dot product derived from floating point add sub unit. They done separately and multiplexers choose add and sub with XOR process [7]. Converting decimal number to the floating point number, block diagram for floating point multiplier.



Block diagram of floating point multiplier

3.1 FLOATING POINT ALGORITHM:

1. Converting the value to binary, take fractional part for separating the integral value and fractional value. This fractional part is converting by multiplication [8]. Multiply by 2 repeatedly and harvest each one bit. It shows the decimal value to floating point.

For Example:

Convert 2.625 to floating Point format:

The integral part is $2_{10} = \mathbf{10_2}$

The fractional parts are

0.625*2= 1.25 – **1**

0.25 *2= 0.5 -- **0**

0.5 *2= 1.0 -- **1**

For $0.625_{10} = 0.101_2$ and $2.625_{10} = \mathbf{10.101_2}$

2. Adding an exponent part to binary number: The Product of Append and 2 power Exponent in the end of binary numbers.

$10.101_2 = 10.101_2 * \mathbf{2^0}$

3. Normalization: The value doesn't change when exponent adjust to one bit left in binary number.

$10.101_2*2^0 = 1.0101_2* \mathbf{2^1}$

5. Mantissa: Mantissa or significand is next to leading number which filled with zeroes on the right.

Mantissa – **0101**

6. Now add the bias to exponent of 2 in exponent field, for all biasing $2^{k-1} - 1$, k=number of bits in exponent field.

For example 8 – bit format $2^{3-1} -1 = 3$

32 –bit format $2^{8-1} -1 = 127$

Here the exponent power is 1, so $1 + 3 = 4 = 100_2$

7. The sign bit of negative is 1 and positive is 0 of given number.

For 8 bit and 32 bit

SIGN BIT	MANTISSA	EXPONENT
0	100	0101
0	10001001	01010000000000000000000

3.2 MULTIPLICATION OPERATION:

Take 2 floating point number A= -18.0 and B= 9.5

Their binary values are A= -10010.0 B= +1001.1 and [9] the normalization A= $-1.001*2^4$ B= $+1.0011*2^3$

1	10000011	00100000000000000000000
0	10000010	00110000000000000000000

3.3 Multiplication of Mantissa:

For normalization [10] adding 1 to the most significant bit is useful,

A	10010000000000000000000
B	10011000000000000000000

0 1	01010110000000000000000	00000000000000000000000

The result is in 48 bit: 0*558000000000

3.4 Adding the exponents:

For mantissa multiplication remove bias in two operands and add again the bias [10].

E result= (Ea-127) + (Eb-127) +127 E r= Ea+ Eb-127 then E r= 10000110

3.5 Calculation:

The Sign result Sr by EXOR of two operands Sa= 1 and Sb= 0 is

$Sr= Sa \oplus Sb \quad Sr= 1 \oplus 0 = 1$

The final result of sign, exponent and mantissa are

1	10000110	01010110000000000000000

A*B = -18.0* 9.5

$= -1.0101011*2^{134-127}$

$= -10101011.0$

$=-171.0_{10}$

IV. PROPOSED SYSTEM

The discrete design and floating point in 32 bit are the worst case for reducing the area and consumption. The proposed system is fused floating point dot product unit multiplication in double precision 48 bit reduce the delay and silicon area. It increases the speed faster than single precision floating point. It also decreases 40% of the worst errors. Floating points were implemented by conventional floating point and fused floating dot product unit multiplication in double precision. This A* B± C* D used in parallel operation is done by 2 cycles. For fused dot product is done in single cycle [11].

The design has been simulated in MODEL SIM, create design in work library as 45nm CMOS study cell and compile the design by go to simulation and start simulate to run, before that source code in VHDL [12]. VHDL is a high level language and used for digital circuits and systems. The system verilog input generates stimulus output. It is easy and similar to C programming and also used in Engineering Design Automation Tool. The fused floating points

V. CONCLUSION

The floating point multiplier is varied by 32 bit and 48 bit inputs. The simulation for both single precision and double precision design and implementation are analyzed [13]. The parallel operation is faster speed than the series single precision floating point. Comparing the fused floating point is higher performance than discrete values. The delay and silicon area are reduced and gives high speed more than 70%. Sometimes more performance in dual path reduction and pipe lining are used [14]. The more accurate result are performed by eliminating the rounding modes and normalization. The fused floating points are used for processors, system controllers and hardware.

VI. REFERENCE

[1] 32 bit Single Precision floating point Multiplier, Ms.Radhika Jumde, AVBIT, Pawnar, Wardha.

[2] IEEE Standard for floating point arithmetic, IEEE Standard 754-2008, New York, Inc., Aug.29, 2008.

[3] Saleh and E.Swartzlander, Jr., "A floating point fused dot product unit," in Proc. IEEE Int. Conf. Compute Design, 2008, pp. 427—431.

[4] Design and Simulation of Binary floating point multiplier using VHDL, U.V. Chaudhari and Prof.A.P.Dhande, Feb-2015.

[5] Simulation and Synthesis for multipliers using VHDL, Raj kumar singh, Shivanada Reddy, 2008.

[6] IEEE 754 floating point fused add sub unit, Sharmila Hemanandha, Siva Subramanian, Aug-2015.

[7] Floating point fused dot –Product Unit, kishore, Prakash, May-2015.

[8] Floating Point Adder and Multiplier, Eduardo Sanchez EPFL- HEIG- VD. Aug-2013.

[9] Normalization on floating point Multiplication using Verilog HDL, V.Narasimha, V.Swathi,

[10] Multiplication of floating point numbers using VHDL, Ms. Sobin Daniel, Sep-2014.

[11] A floating point multiplier, Concordia University.

[12] Design and Implementation of different multiplier using VHDL by Moumita Ghosh, Rourkela, 2007.

[13] Design of floating point Multiplier, Vamsi Krishna, Trivedi, Mar-2014.

[14] VHDL Modeling of floating point for VLSI designer Library, Wai-Leong Pang, Kah-Yoong Chan, Sew-Kin Wong, Choon- Siang Tan, July-2012.