ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Efficacious Architecture for Single Precision Floating Point Unit

Ch.Sreekanth
M.Tech,
Department of ECE,
Mother Teresa Institute of Science & Technology,
Sathupally, Khammam, Telangana, India.

G.Uday Kiran Bhargav
Assistant Prof,
Department of ECE,
Mother Teresa Institute of Science & Technology,
Sathupally, Khammam, Telangana, India.

ABSTRACT:

Floating is an important consideration for the many systems that are required high accuracy. That can be fulfilled with the IEEE 754 standard format. Base on the format the different operation can be performed but the area and power consumptions can be making tedious. In order to archive the effect of the power and speed we propose the novel methods of the partial product generation and also the advance multi-operand addition, and also the less architecture for the exponent addition. This paper emphasis advanced floating point ALU design. This can be done by using the unsigned Radix-4 multiplication method with novel addition strategies.

I.INTRODUCTION:

For DSP application, if an algorithm involves summation of a large number of products, it requires a large number of bits to represent the signal to allow for adequate signal growth over the summation. However, since a processor architecture will not allow for an unlimited number of bits ,some processors choose a floating point format for signal processing computationsIn order to perform the real number arithmetic operations IEEE 754 standard floating point only the possible way. This standard format refers the representation of the floating point number. This format which includes the sign, exponent and mantissa

Figure1. IEEE single precision floating point format

S = Sign
E = Exponent
M = Mantissa
The value represented by the data format in fig 1 given as

X=(-1)^s*2^(E-bias)*(1.F)

In the above sign bit is used to represent the sign of the particular numbers which is of one bit wide, (E) exponent is of 8-bit width which can be bias added with shifts. And the M- referred as the mantissa that can be 23 bits wide.

The bias depends upon the bits reserved for the exponent. In fig 1 the bias is 127, the largest positive number represented by 8 bits. The value of E can be from 0 to 255. floating point format, used to increase the range of numbers that can be represented, suffers from the problem of speed reduction for DSP computation.

Multiplying two numbers in floating point format is done by 1- adding the exponent of the two numbers then subtracting the bias from their result, 2- multiplying the significant of the two numbers, and 3- calculating the sign by XORing the sign of the two numbers.

In order to represent the multiplication result as a normalized number there should be 1 in the MSB of the result (leading one). This paper includes the Introduction and floating point unit description as the section II, proposed method as the section III followed by the results, conclusion and references.

II.FLOATING POINT UNIT:

Floating point can be designed for performing the operations like addition, subtraction, multiplication and division. There are different algorithms that are related with the individual operation of the floating point unit.

As we now the addition and subtraction algorithms are similar to each other except the subtraction of the significant. Whereas multiplication algorithm included steps like sign determination, exponent addition, significant multiplication, normalization and over flow/ under flow detection.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

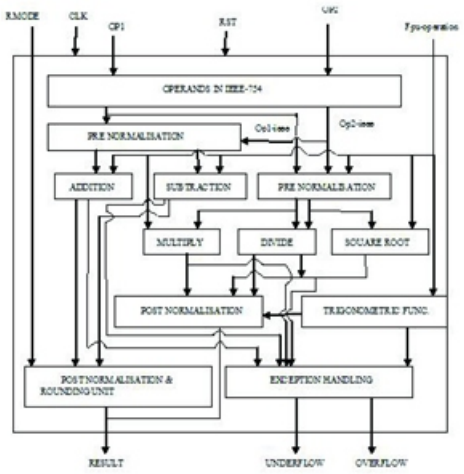


Figure2. Block Diagram of floating point arithmetic unit

Booth multiplication is a technique can make smaller, faster multiplication circuits, by recoding the

A)Partial product generator:

Rather than the traditional multiplication the multiplication can be carried out by the grouping the multiplicand and the multiplied by the redundant value of the particular group. In this the booth recoding can be of three bits each starts for either LSB to MSB or MSB to LSB by the over lapping each group numbers that are multiplied. By this we can reduce the number of partial products by half, by radix-Booth recoding technique can be considered as the partial product generated.

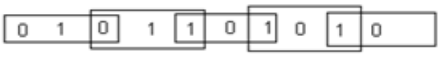


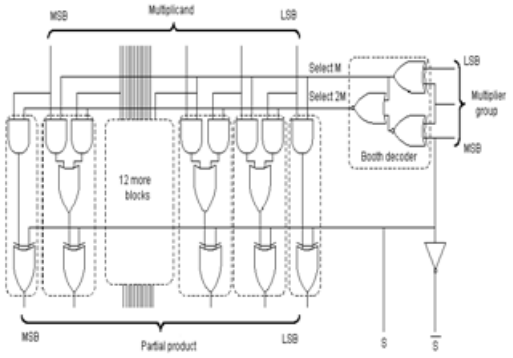
Fig3. Grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 1

TABLE 1

Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

The partial product generator it consist of the two blocks namely booth encoder and booth selector. Both encoder give the 1x or 2X and original and one left shifted output can be selected by the booth selector. For the compliment the xor tree is provide. In our architecture we are adopted the unsigned multiplication strategy for the design of the multiplication. There by we can reduce the area and the bit length of the partial products and the neg bits.



3.7 Booth partial product selector logic

B) Exponent Addition :

In general exponent addition
Etemp = Ea+Eb
Eout= Etemp – Bias
For exponent addition we uses ripple carry adder and where as for the subtraction from the bias, as we know the bias value we need not again to take two variable subtractions. In order to archive this we proposes the ONE and ZERO subtractors foe the bias subtraction

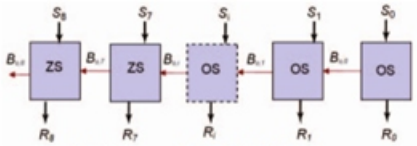


Figure 6. Ripple Borrow Subtractor

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

(127)10 = (00111111) 2
In one subtractor one input is contant one i.e., T which is in the bias where as in zero subtractor zero input is contant one i.e., T which is in the bias as shown in below fig.

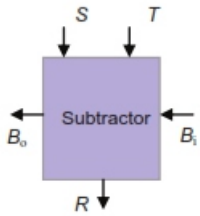


Fig: proposed on input constant subtractor

Table- One Subtractor:

S	T	B_i	Difference(R)	B_o
0	1	0	1	1
1	1	0	0	0
0	1	1	0	1
1	1	1	1	1

Boolean equation of one subtractor

$$Difference\ (R) = \overline{S \oplus B_i}$$

$$Borrow_{out}(B_o) = \overline{S} + B_i$$

Table-Zero Subtractor:

Boolean equation of zero subtractor

$$Difference\ (R) = S \oplus B_i$$

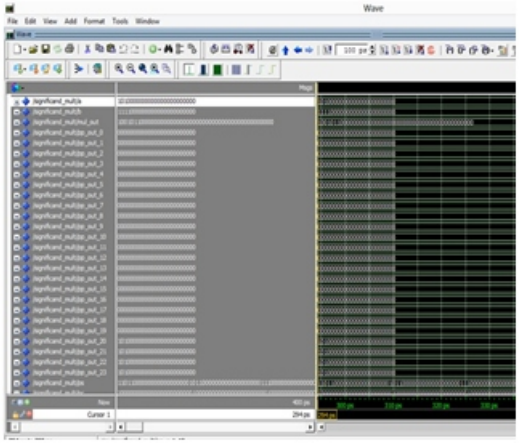
$$Borrow_{out}(B_o) = \overline{S} \cdot B_i$$

By this makes the effectve design of the floating point unit with the added advantage wit refers to the partial product generation and the exponent addtion.whic enables the high end performance of the systems in the three aspects bny considering the out the change made to the previous architecture.

IV.RESULTS AND DISCUSSION :
Significant multiplication:

Now the main part of this project is multiplication. The multiplication results are shown in the figure 5.5. here we are used braun multiplier the results are as follows
Let a=1.01000000000000000000000
B=1.11100000000000000000000
Here we use m.s.b preferable multiplication it is as shown below

1.0100
× 1.1110
00000
10100
10100
10100
10100
1001011000
We are Place the decimal point: 10.01011000



The above shows the partial products, which is 12 that means the partial products can be decreased by the factor of two.Over all top module which is of all results are shown below.

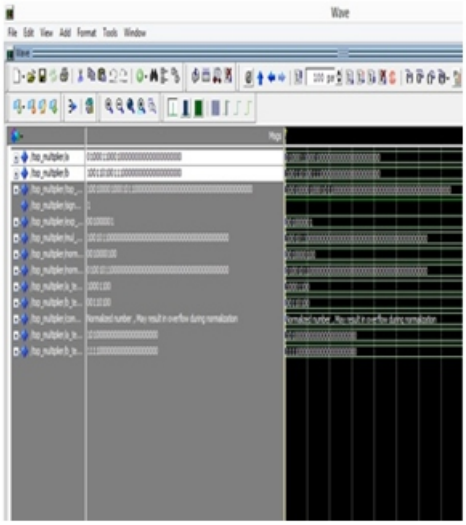


Fig.4 Result of top module

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

V.CONCLUSION:

The implementation of a high speed single precision FPU has been presented. The design has been synthesized with XININX. Strategies have been employed to realize optimal hardware and power efficient architecture. The layout generation of the presented architecture using the FPGA flow is an ongoing process and is being done using XININX RTL compiler with Spartan-6. Hence it can be concluded that this FPU can be effectively used for ASIC implementations which can show comparable efficiency and speed and if pipelined then higher throughput may be obtained.

REFERENCES:

[1] Rudolf Usselmann, "Open Floating Point Unit, The Free IP Cores Projects".

[2] Edvin Catovic, Revised by: Jan Andersson, "GRFPU – High Performance\ IEEE754 Floating Point Unit", Gaisler Research, Första Långatan 19, SE413 27 Göteborg, and Sweden.

[3] David Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic", ACM Computing Surveys, Vol 23, No 1, March 1991, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304.

[4] Yu-Ting Pai and Yu-Kumg Chen, "The Fastest Carry Lookahead Adder", Department of Electronic Engineering, Huafan University.

[5] Prof. Kris Gaj, Gaurav, Doshi, Hiren Shah, "Sine/Cosine using CORDIC Algorithm".

[6] S. F. Oberman and M. J. Flynn, "Division algorithms and implementations," IEEE Transactions on Computers, vol. 46, pp. 833–854, 1997.

[7] Milos D. Ercegovac and Tomas Lang, Division and Square Root: Digit- Recurrence Algorithms and Implementations, Boston: Kluwer Academic Publishers, 1994.

[8] ANSI/IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, 1985.

[9] Behrooz Parhami, Computer Arithmetic - Algorithms and Hardware Designs, Oxford: Oxford University Press, 2000.

[10] Steven Smith, (2003), Digital Signal Processing-A Practical guide for Engineers and Scientists, 3rd Edition, Elsevier Science, USA