VOL. 10, NO. 12, JULY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

ARCHITECTURAL DESIGN OF 8 BIT FLOATING POINT MULTIPLICATION UNIT

Usha S.[1] and Vijaya Kumar V.[2]
[1]VLSI Design, Sathyabama University, Chennai, India
[2]Department of Electronics and Communication Engineering, Sathyabama University, Chennai, India
E-Mail: ushass.sekar@gmail.com

ABSTRACT

In the recent high speed processor, floating point ALU (FP ALU) is one of the important units to perform the arithmetic and logical functions of the floating point number. Floating point multiplication is one of the arithmetic operations that take more computational time and when implemented in hardware, it requires more area due to the large number of component. However the advantage of implementing the floating point unit in the hardware system is to overcome the overflow and underflow conditions that occur in the logical operation. This paper presents a high speed 8 bit multiplier. To increase the speed and to reduce the power consumption due to the clock load, wave pipelining method has been used. The coding is written in VERILOG HDL and the design is analyzed in the Xilinx environment.

Keywords: FPU, floating point multiplication, wave pipelining, HDL.

INTRODUCTION

Multipliers are extensively used in Microprocessors, Digital Signal Processors and Communication applications. For the multiplication of larger numbers and higher order, a adders used are huge in number to perform the partial product addition. High speed multipliers are needed for increasing the speed of the processors. Higher throughput arithmetic operations are important to achieve the desired performance in many real time signal and image processing applications. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades [4] Reducing the time delay and power consumption are very essential requirements for many applications. In the past, multiplication was implemented generally with a sequence of addition, subtraction and shift operations. Two most common multiplication algorithms followed in the digital hardware are array multiplication algorithm and Booth multiplication algorithm. Since the importance of digital multipliers in DSP's is large, it has always been an active area of research. An 8 bit floating point number consists of signed, exponent and mantissa bits. All the bits are to be processed in order to perform the multiplication operation. Pipelining and parallel processing are the concepts that can be used in computational units to reduce the delay [1, 8].

Wave pipelining is the technique for pipelining digital systems that can increase the clock frequency of practical circuits without increasing the number of storage elements. In wave pipelining many coherent waves of data are sent through a block of combinational logic by applying new input faster than the delay through the logic ideally [3] If all parts from input and output have equal delay, when the circuit's clock frequency is limited by rise and fall times, clock skew, setup and hold times of the storage elements [2, 9]. In the ordinary pipelined system there is one wave of data between register stages. When a new set of values are clocked into one set of registers then the values are allowed to the next set of register before the first set is clocked again. In contrast wave pipelining is the use of multiple coherent waves of data between the storage elements.

8 BIT FLOATING POINT REPRESENTATION

Floating-point operations are used to simplify the addition, subtraction, multiplication, and division of fractional numbers. They are used when dealing with fractional numbers, such as 5.724 or a very large number and signed fractional numbers. When performing arithmetic operations involving fractions or very large numbers, it is necessary to know the location of the binary (radix) point and to properly align this point before the arithmetic operation. For floating-point operations, the location of the binary point will depend on the format of the computer. All numbers are placed in this format before carrying out the arithmetic operation. The fractional portion of the number is called the mantissa and the whole integer portion, indicating the scaled factor or exponent, is called the characteristic. The floating-point numbers are coded as reversing the sign-bit inverses the sign. Consequently the same operator performs as well addition or subtraction according to the two operand's signs. In the floating point 8 bit IEEE representation the first bit is assigned for sign of the number, the next three bits are assigned for the exponents and the last four bits are assigned to the mantissa. Figure-1 represents the 8 bit floating point number in IEEE standard.

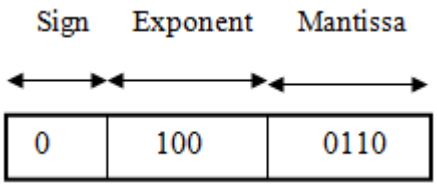
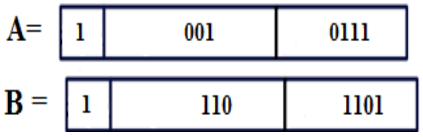


Figure-1. 8 Bit floating point representation.

8 BIT FLOATING POINT MULTIPLICATION

The multiplication entity has two 8 bit inputs A and B. The flow diagram of the multiplication process is shown in Figure-2.

The multiplication process of floating point numbers involves the following steps. Consider the following examples A and B.



- Obtain the input bits and assign the bits to separate wires. Sa and Sb are the signed bits, Ea and Eb are the exponent bits, Ma and Mb are the mantissa of the inputs A and B.
- Carry out the ex-or operation between Sa and Sb and obtain the resultant sign bit. Equation 1 provides the resultant sign bit.

$$\text{Sign} = \text{Sa}\text{\^{}}\text{Sb} \quad (1)$$

- Add the exponents and subtract the bias (100) from the result. Thus the resultant exponent can be obtained. Equation 3 represents the resultant exponent bit.

$$Ex = (Ea + Eb) \quad (2)$$

$$Ex = (001 + 110) = 111$$

$$Er = (Ex - 100) = (111 - 100)$$

$$Er = 011 \quad (3)$$

Multiply the Mantissa Ma and Mb using the conventional multiplication method and normalize the result.

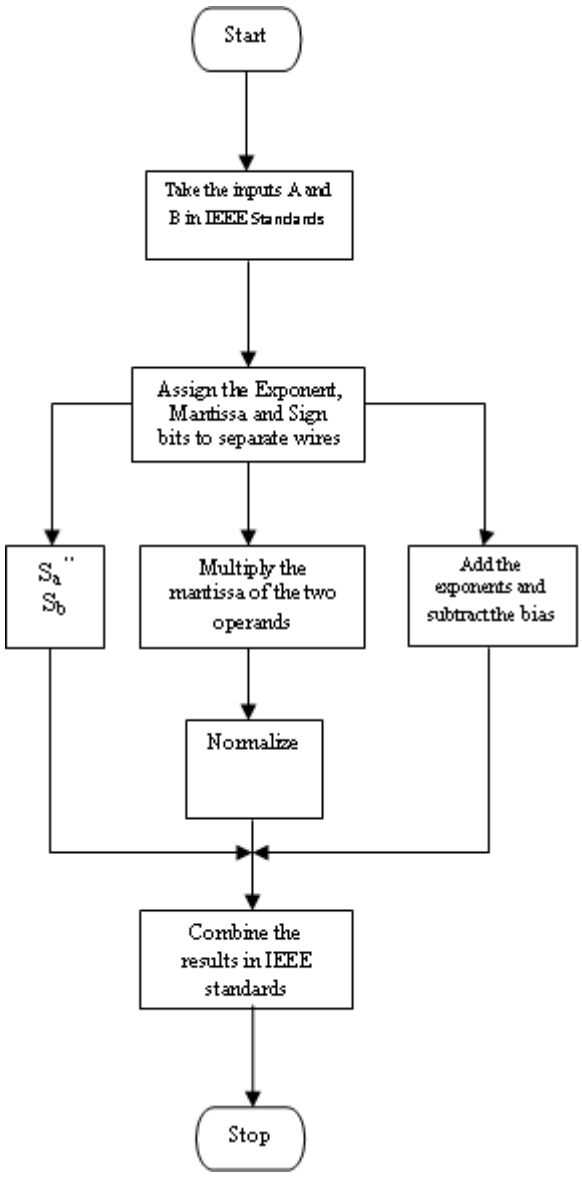


Figure-2. Flow diagram of floating point multiplication.

FLOATING POINT MULTIPLICATION UNIT

The architectural level helps in obtaining the exact result of the floating point multiplication. Hence the design does not require any status register for the overflow and underflow conditions. This has been separated into 3 modules like sign, exponent and mantissa. However for the exceptions like Not a Number (NAN) [7] and infinity the design provides the result to be 0. The different modules are explained in detail below. Figure-1 shows the block diagram of the floating point multiplication unit. The ex-or gate is used to produce the signed bit of the resultant bit. Sa and Sb are the inputs and sign is the out bit of the signed module. Ex is the output of the 3 bit ripple carry adder. Er is the output of exponent module after the bias is subtracted. The mantissa module is

designed for the conventional multiplication where the pipeline registers are added in between each stage.

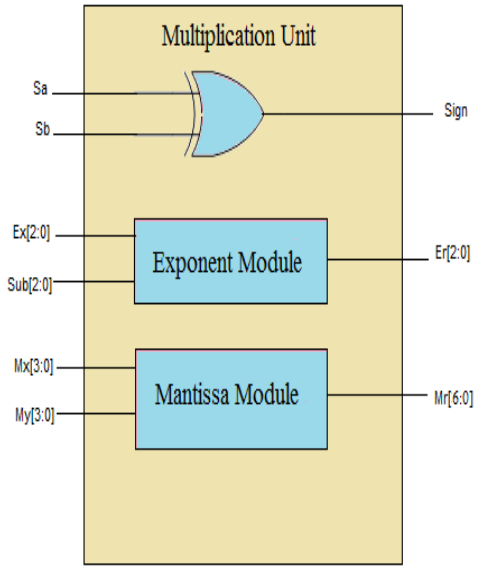


Figure-3. Floating point multiplication unit.

Sign module

In this module the resultant sign bit is obtained by performing ex-or operation between the inputs Sa and Sb. Figure-4 show the ex-or gate that gives out the output sign.

Figure-4. Ex-Or gate for resultant sign bit.

Exponent module

In this module processing steps for the exponents are carried out. The exponents are added and the bias number is being subtracted from the result.

Mantissa module

In this module multiplication of the mantissa is carried out. The sets M1 to M16 are obtained by conventional multiplication method by using an AND gate. By repeated addition the sum and carry of each element are obtained.

Thus from the Figure-7 the sum s0 to s7 are the added sum result and the carry c9 is also stored in the register. Figure-5 shows the circuit for mantissa multiplication. Thus when the pipelined registers are used in between these stages the speed of operation gets increased. Wave pipelining is the concept of finding the delay of each element and then making the delay to be same overall the circuits. Multiple signals with different timing delays are sent to trigger the transmission gates of the wave pipelined circuit. Hence they transfer the data from one stage to another.

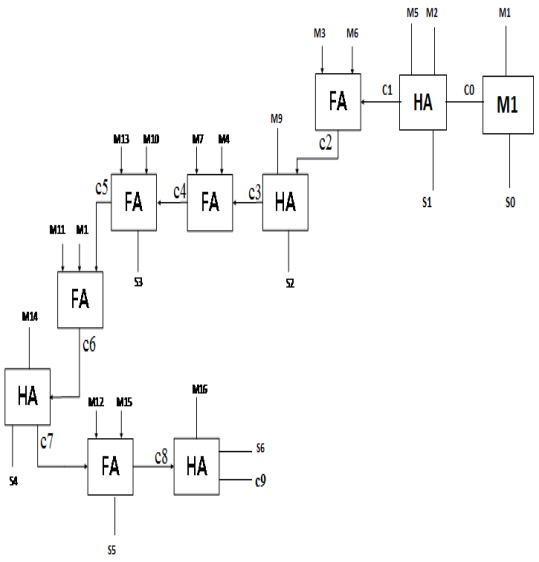


Figure-5. Circuit for mantissa multiplication.

The output of the AND gate M1 is taken directly as the LSB of the result as s0. Outputs of M2 and M5 are fed into the half adder. The sum output s1 is considered and the carry of which is given as the input to the full adder along with which M3 and M6 are fed in and the half adder which takes the carry along with the input M9 produces the sum s2. The same method is used to obtain the other outputs till s7 as in Figure-7.

SIMULATION RESULTS

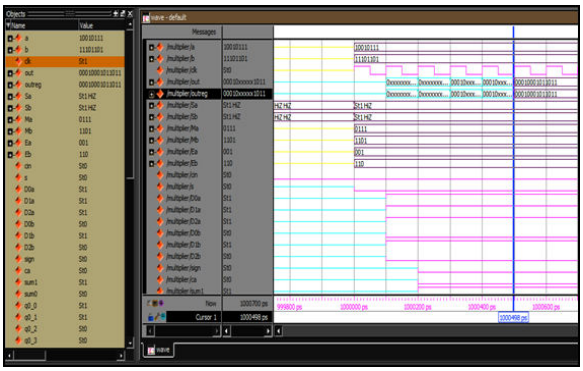


Figure-6. Simulation Result of an floating point multiplier for uneven exponents.

In the Figure-6 the simulation result for the multiplier with different exponents is shown. In this A, B, clk are the inputs and the output is out.

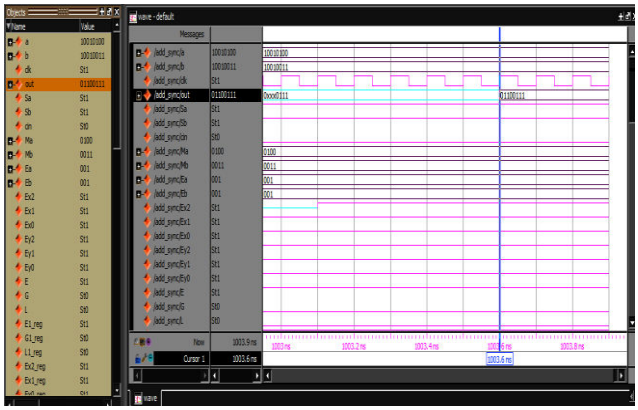


Figure-7. Simulation result for floating point multiplier with equal exponents.

In the Figure-7 the simulation result for the multiplier with equal exponents is shown. In this A, B, clk are the inputs and the output is out.

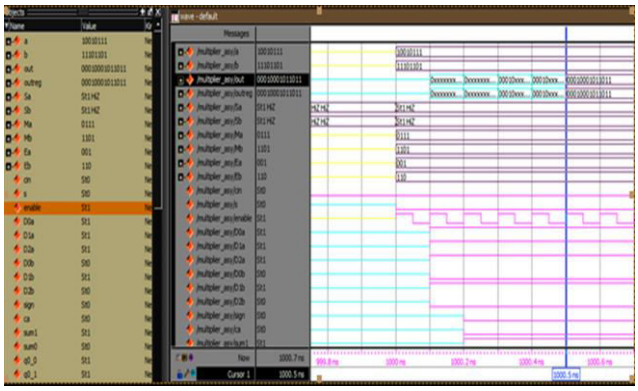


Figure-8. Simulation result for wave pipelined multiplier.

In the Figure-8 the simulation result for the multiplier with different exponents is shown. In this A, B, ena are the inputs and the output is out_reg.

Table-1. Performance analysis for synchronous multiplication unit.

Module	Synchronous multiplication unit		
Multiplication unit	**Power** (W)	**Delay** (s)	**PDP** (J)
	192m	0.20n	395.5p

Table-2. Performance analysis for wave pipelined multiplication unit.

Module	Wave pipelined multiplication unit		
Multiplication unit	**Power** (W)	**Delay** (s)	**PDP** (J)
	160m	0.62n	99.05p

Table-1 and Table-2 shows the performance analysis of the multiplication unit, where the performance is improved by 65% in wave pipelined based design.

CONCLUSIONS

Thus an efficient floating point multiplication unit has been designed. To increase the computation speed pipelined registers are used. This makes the hardware implementation of the design to be easier. It gives us method for hierarchical multiplier design and clearly indicates the computational advantages offered by pipelined stages. For the exponent part the 3 bit adders are used for the design and the sign bit is obtained by the ex-or operation of the input sign bits. The same design is also carried out using the wave pipelining method. The delay increased in the unit is negligible. However the power consumption of the wave pipelined multiplier gets reduced by 65%. The designed is analyzed in Xilinx environment and the result is obtained after simulation.

REFERENCES

[1] Dave Omkar. R. ASIC Implementation of 32 and 64 bit Floating Point ALU using Pipelining. International Journal for Computer Applications.

[2] Shuchi Bhagwani. Comparative Study of Various Network Processors Processing Elements Topologies. International Journal of Engineering Science and Innovative Technology (IJESIT). 2(1).

[3] Deck C. Wong *et al*. Designing High Performance Digital circuits using wave pipelining: Algorithms and Practical Experience. IEEE Transactions on computer aided design of Integrated circuits and Systems. 12(1).

[4] Sivarama Dandamudi. Guide to RISC ProcessorsFor Programmers And Engineers.

[5] M. Morris Mano. 1993. Computer System Architecture. 3rd edition, Prientice-Hall, New Jersey, USA. pp. 346-348.

[6] Premananda B .S. *et al*. Design and Implementation of 8-Bit Vedic Multiplier. International Journal of

Advanced Research in Electrical, Electronics and Instrumentation Engineering. 2(12).

[7] Shrivastava Purnima *et al*. VHDL environment for Floating point Arithmetic Logic Unit - ALU Design and Simulation. Research Journal of Engineering Sciences, ISSN 2278-9472, 1(2): 1-6.

[8] Ravi.T, Kannan. V. 2012. Design and Analysis of Low power CNTFET TSPC D Flip-Flop based shift Registers. Applied Mechanics and Materials journal.

[9] Ravi T *et al*. 2013. Ultra Low Power Single Edge Triggered Delay Flip Flop Based Shift Registers using 10-Nanometer Carbon Nano Tube Field Effect Transistor. In American Journal of Applied Sciences. 10(12): 1509-1520.