VOL. 12, NO. 1, JANUARY 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN IMPROVED FLOATING POINT ADDITION ALGORITHM

S. Subha
Department of Information Technology, SITE, Vellore Institute of Technology, Vellore, India
E-Mail: ssubha@rocketmail.com

ABSTRACT

Floating point addition/subtraction has been designed in literature. The methods involved two's complements add/subtract logic and XOR for counting leading zeroes. This paper proposes algorithm to perform add/subtract operation using one's complement and counting leading zeroes for normalization using NOR operation as reduction operator for two bits of mantissa at a time. The proposed model is simulated with Quartus 2 version with Cyclone II family processor. An improvement in area by 34% with increased performance of 47% with comparable power consumption is observed when compared with two's complement add/subtract, XOR operation for leading zeroes count.

Keywords: floating point add/subtract, leading zeroes count, NOR logic

1. INTRODUCTION

Floating point operations find applications in various computer fields. The floating points operations add, subtract, multiply and divide involve the corresponding operations to be performed on the various parts of floating point numbers. These operations are binary i.e. they require two operands and produce one result. The floating point representation as suggested by IEEE standards is shown in Figure-1. The number consists of sign bit, exponent and mantissa fields. The sign bit is for the mantissa field. The exponent field is 8 bits wide and is always positive. It uses excess notation. Exponents are expressed in excess-127 notation. The mantissa is 23 bits wide. It represents the number in binary number system. The mantissa is of the form 1.xxx…xx where x is 0 or 1. The leading one is assumed and is not represented in the number.The mantissa starts with one i.e. there are no leading zeroes.

	Sign	Exponent	Mantissa
#bit	1	8	23

Figure-1. IEEE 754 Single precision floating point notation.

The floating point operations are addition, subtraction, multiplication and division. For floating point add/subtract algorithms have been proposed in [1]. The authors use mainly two's complement for add/subtract operation in [1]. The power consumption of floating point add/subtract unit depends on the number of active components in the circuit. The number of active components is less in one's complement add/subtract operation than two's complement operation. The number of leading zeros is calculated in [1] using XOR of consecutive bits. The number of gate levels in XOR is more than in NOR gate. In view of these observations, this paper proposes floating point add/subtract operation using one's complement arithmetic and counting leading zeroes using NOR reduction operation. The proposed model was simulated in Quartus 2 Cyclone II environment. An improvement in area by 34% with increased performance of 47% with comparable power consumption is observed when compared with two's complement add/subtract, XOR operation for leading zeroes count.

The rest of paper is organized as follows. Section 2 gives the mathematical background, section 3 proposed model, section 4 simulations, section 5 conclusion followed by references.

2. MATHEMATICAL BACKGROUND

Adding of two numbers in binary can be done in two ways. One is one's complement notation and second is two's complement notation. Consider two numbers A and B. They are represented in sign magnitude notation. The steps in one's complement notation are given below [5].

A. Represent the numbers in one's complement notation. The one's complement notation is defined as follows for any binary number x.
a. One's complement of +x = x
b. One's complement of –x = 1111…11 – x
B. Perform the addition of the two numbers in step 1.
C. If there is carry out of the most significant bit (MSB), add it to the least significant bit(LSB). This is called end around carry.
D. If the result has zero in MSB the result is correct. If the result has one in MSB the correct result is the one's complement of the obtained result.
E. StopIn one's complement notation is that zero has two representations positive zero and negative zero.

The steps in two's complement notation add/subtract are given below [5].

1. Represent the numbers in two's complement notation. The two's complement notation is defined as follows.
a. Two's complement of +x = x
b. Two's complement of –x = One's complement of x + 1
2. Perform the addition of two numbers in step 1
3. Discard any carry of the result in step 2
4. If there is no end around in step 2 carry during subtraction, the result is the negative of two's complement of the result.

5. Stop.

There is only one representation of zero in two's complement notation.

The IEEE-754 floating point notation for single precision is given in Figure-1. A floating point number is denoted as sem where s is the sign of mantissa, e is the exponent and m is the mantissa without the leading one. Consider two floating point numbers A and B. The following steps are followed to perform addition of A and B.

Algorithm FP_ADD: Given two numbers A and B represented in IEEE-754 single precision format this algorithm performs the addition of the two numbers and represents the result in IEEE-754 single precision format. Assume the exponent in **B** is less than or equal to the exponent in **A**. Let the exponent of **B** be **b** and let the exponent of **A** be **a**.

a) First, convert the two representations to scientific notation. Thus, the hidden one is explicitly represented.

b) In order to add,the exponents of the two numbers need to be the same. The operand B is rewritten for this. This will result in B being not normalized, but value is equivalent to the normalized **B**. Add **a - b** to **B**'s exponent. Shift the radix point of the mantissa (signficand) **B** left by **a - b** to compensate for the change in exponent.

c) Add the two mantissas of **A**and the adjusted **B** together.

d) If the sum in the previous step does not have a single bit of value 1, left of the radix point, then adjust the radix point and exponent until it does.

e) Convert back to the one byte floating point representation.

f) Stop.

The above algorithm has been implemented with various algorithms at each step in[4]. The following is the implementation of the above algorithm in two's complement notation as proposed by the authors in [1].

Algorithm FP_ADD_two complement: Given two IEEE-754 single precision floating point numbers A and B this algorithm performs the addition/subtraction of these two numbers giving result in IEEE-754 single precision format.

a) Find the larger of two exponents by 2's complement subtraction.

b) Extend the mantissa of two numbers by adding the leading one.

c) Left shift the mantissa of two numbers by two bits. This results in the mantissa of 26 bits.

d) Adjust the mantissa of the lower exponent number by suitably right shifting by the difference of the exponents' value.

e) If operand is negative determine its two's complement. Perform addition operation of the operands in two's complement notation.

f) If the result carry out is equal to one get the two's complement of the result.

g) If the operation is addition display the result from step 5-6. If the operation is subtraction, display the two's complement of the result in step 5-6.

h) For the result obtained calculate the number of leading zeroes. This is done by XOR'ing the consecutive leading bits and incrementing the leading zeroes count if the result of XOR is zero. This operation is stopped as soon as the result of XOR is one.

i) Left shift the mantissa by the leading count number. Right shift the exponent by leading count number.

j) Left shift mantissa by one to remove the leading one. Left shift the exponent to adjust this.

k) Rounding is performed as mentioned in [1] after normalization. The mantissa M is 26 bits. If M[2] or M[3] is equal to zero, then if M[1:0] is zero then do nothing. If M[1:0] is not zero add one to LSB of M. The final result is M[25:3].

l) Stop.

The time taken for the above algorithm is dependent on the calculation of two's complement of mantissa, exponent and calculating the leading zeroes of the result.

The authors in [6,7] implemented the floating point add/subtract operation using one's complement. The authors in [3] implemented leading zero logic using OR, NOT, AND logic.The authors in [4] proposed leading zero counter using NOR planes. The authors in [8] propose floating point addition operation is obtained in one cycle 32% of time using variable latency algorithm. The author in [9,10] proposed novel design for leading zero detector using algorithm which finds the leading zeroes by considering two bits extending it to four bits at a time. The four bit LZD has NOR, OR, NOT gates. The authors in [2] mention that the leading zeroes in IBM RISC System/6000 floating point unit uses NOR gate on the inputs. The inputs with carry are used for NOR logic to count the number of zeroes.

3. PROPOSED MODEL

The algorithm proposed in [1] uses two's complement notation for determining the larger of two exponents and performing add/subtract operation on the mantissa. As two's complement operation requires the use

of adder to obtain the two's complement of a number, the absence of adder can reduce the hardware complexity. The author proposes to perform these operations using one's complement notation taking suitable steps to represent zero as positive zero. The model proposed in [1] also uses XOR operation to count the number of leading zeroes in the result of floating point mantissa operation. The XOR operation requires two gate levels in the simplest design. The author in this paper proposes to reduce the number of gate levels by using NOR gates. The proposed algorithm is shown below.

Algorithm FP_ADD_NOR: Given two IEEE-754 single precision floating point numbers A and B this algorithm performs the addition/subtraction of these two numbers giving result in IEEE-754 single precision format.

a) Convert the two numbers to be represented in scientific notation by including the hidden one.
b) Make the two exponents equal. This is done by choosing the larger of two exponents as the exponent of the result. The mantissa of smaller exponent number is shifted left for the appropriate number of times. The larger of two exponents is obtained by performing the subtraction of the two exponents using one's complement and making decision based on the sign of the result.
c) Perform the addition operation. For subtraction, perform the one's complement subtraction. The special case of zero is taken care of by making it positive zero.
d) Discard the leading zeroes in the result. Retain the count of number of leading zeroes that are discarded. The leading zeroes are calculated by applying the reduction NOR operator on two bits at a time starting from most significant bit in the mantissa. Let lzk be the leading zeroes counter. For any two consecutive bits k and l the following steps are performed.

1.Find temp = NOR(k,l)
2.If (temp =1) lzk = lzk + 2, stop
3.If (temp = 0) do steps 4-5
4.If (k=1) stop
5.If (l = 1) lzk = lzk+1, stop

e) Shift the exponent right by the count of leading zeroes obtained in step 4.
f) Normalize the number obtained in step (e) above. Apply rounding algorithm to this result.
g) Stop.

The hardware complexity is reduced in the proposed model.

4. SIMULATION

The algorithm proposed in [1] was simulated on Quartus 2 tool. The hardware configuration chosen is given in Table-1. The model was compiled, synthesized and power consumption calculated. The proposed model was simulated in same environment. It was compiled, synthesized and power consumption calculated. The simulation results are shown in Table-2. As seen from Table-2 an improvement in area by 34% with increased performance of 47% with comparable power consumption is observed when compared with two's complement add/subtract, XOR operation for leading zeroes count.

Table-1. Hardware configuration.

Parameter	Value
Processor Family	CycloneII
Device	EP2C15AF484C6
Package	FBGA
Speed	-Fastest

Table-2. Simulation results.

Parameter	Traditional	Proposed	%improvement
Area	#slices: 1234/14448	#slices : 809/14448	34.44%
Power	79.33mW	79.29mW	0.05%
Timing	107.348ns	56.678ns	47.20%

Example: The following gives example of the proposed logic. Let the two numbers be A and B. The numbers are

A = 0 0101 1000 0101 0100 1010 1001 0110 111
B = 0 0101 1000 0001 0010 0101 0110 1001 000

a) Both exponents are equal. The result exponent is 01011000.

b) The numbers after adding the leading one is calculated. The numbers are shifted left two times to accommodate for rounding. The operation A+B is performed as given below:

A= 1 0101 0100 1010 1001 0110 111 00
B=1 0001 0010 0101 0110 1001 000 00
A+B = 10 0110 0110 1111 1111 1111 111 00

c) The number of leading zeroes is two as the most significant one is discarded. The leading ones aer discarded. The result is
1100 1101 1111 1111 1111 1100 00. The exponent is shifted right twice equals to 0001 0110. .

d) Left shift mantissa by one to remove the leading one. This gives result as

1001 1011 1111 1111 1111 1000 00. The mantissa is denoted as M[25:0].

e) As M[2] is equal to zero and M[1:0] is zero no rounding is performed. The result is

mantissa: 1001 1011 1111 1111 1111 100
exponent: 0010 1100.
Sign : 0

5. CONCLUSIONS

Algorithms for floating point add/subtract operation is proposed in this paper. The model uses one's complement addition/subtraction for the exponent and mantissa add/subtract operation. The NOR reduction operator is used to determine the number of leading zeroes. The proposed model is simulated in Quartus 2 with Cyclone II processor. It is compared with the model proposed in [1] which uses two's complements add/subtract logic and XOR logic for leading zero count determination. The simulation results are shown in Table-2. As seen from Table-2 an improvement in area by 34% with increased performance of 47% with comparable power consumption is observed when compared with two's complement add/subtract, XOR operation for leading zeroes count.

REFERENCES

[1] Djodie Pesic, Ivan Ratkovic. 2015. An efficient FPGA Implementation of Floating Point Addition. Proceedings of 23rd Telecommunications Forum TENCOR. pp. 685-688.

[2] E. Hokenek, R.K. Montoye. 1990. Leading zero anticipator (LZA) in the IBM RISC System/6000 floating-point execution unit. IBM Journal of Research and Development. 34(1): pp. 71-77.

[3] H. Suzuki Y., Nakase H. Makino, H. Morinaka, K. Mashiko. 1995. Leading-zero Anticipatory Logic for high-speed floating point addition, Proceedings of IEEE Customized Integrated Circuits Conference. pp. 589-592.

[4] Kyung-Nam Han, Sang-Wook Han, Euisik Yoon. 2002. A new floating-point normalization scheme by bit parallel operation of leading one position value, Proceedings of ASIC. pp. 221-224.

[5] Morris Mano, Digital Logic and Computer Design, 4th edition.

[6] Nikhil Kikkeri, Peter-Micheal Seidel. 2007. An FPGA Implementation of a Fully Verified Double Precision IEEE Floating Point Adder, Proceedings of IEEE International Conference on Application Specific Systems, Architectures and Processors. pp. 83-88.

[7] Peter-Micheal Seidel and Guy Even. 2004. Delay-Optimized Implementation of Floating Point Addition. IEEE Transactions on Computers. 53(2): 97-113.

[8] Stuart F Oberman, Hesham Al-Twaijry and Micheal J. Flynn. 1997. The SNAP Project: Design of Floating Point Arithmetic Units, Proceedings of ARITH. pp. 156-165.

[9] Vojin G Oklobdzjia. 1992. An Implementation Algorithm and Design of a Novel Leading Zero Detector Circuit. Proceedings of 26th Asilomar Conference on Signals, Systems and Computers. pp. 391-395.

[10] Vojin G. Oklobdzjia. 1994. An Algoirthmi c and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis. IEEE Transactions on VLSI Systems. 2(1): 124-128.