High Speed Floating Point Units using Leading Zero Counting and Anticipation Logic for Addition and Subtraction

D. Jeevalakshmi(M.Tech)[1]
Pursuing M.Tech,

MD. Hayath Rajvee, M.Tech[2]
Asst.Professor,

QUBA College of Engineering and Technology[1,2],
Nellore.

ABSTRACT:- ***The floating point unit plays an important part in the modern microprocessors. The output of the floating point operation should be normalized. This paper mainly focuses on leading-zero counting (LZC) and leading-zero anticipatory (LZA) logic for high speed floating point addition and subtraction. New Boolean expressions are derived for the new leading zero counter. When compared with the known architectures the new circuits can be implemented effectively in both static and dynamic logic and also requires less energy per operation. The pre-decoding for normalization concurrently with addition for the significant is carried out in this logic. Shift operation for the normalization in parallel with the rounding operation is also performed. The use of simple Boolean algebra allows the proposed logic constructed from a simple cmos circuit. The proposed leading-zero counter can also be integrated with a proposed leading-zero anticipation logic for better and fast results.***

I. INTRODUCTION

The speed and accuracy of microprocessors was developed rapidly during the last decade. We have to consider optimizing both the delay and energy consumption of the modern microprocessors, in such a way that the floating point units play a dominant role. The main operation of the floating point data paths is the normalization. The output of the normalization will follow the IEEE-754 standard format i.e., 1.xxxxx.., x (0,1). The normalization process is done by involving the leading zero counting. The problem of normalization of the result involves counting the number of leading zeros in the result and then shifting the result in accordance to the outcome of the leading zero counter unit. To update the exponent part correctly the derived leading zero counter is also required. Almost all instruction sets of contemporary microprocessors include a count leading zeros (CLZ) instruction for fixed-point operands. In all the cases a leading zero anticipator (LZA) is employed to increase the computation speed.

This paper proposes a new leading zero counter and new leading zero anticipation logic which works effectively when compared with existing methods. In section 2 we discuss the functionality of the existing methods .Section 3 discuss about the proposed methods for leading zero counting and leading zero anticipation. Section 5 gives the results and section 6 concludes the paper followed by references.

II. FUNCTIONALITY OF EXISTING METHODS

A leading zero is any zero bit that leads a number string in positional notation. In binary representation consider the number of consecutive zeros that appear in a word before the first more significant bit that equal to one. The later is called leading digit. Leading zeroes occupy most significant digits, which could be left blank or omitted for the same numeric value. The process of encoding these leading zeros is called leading zero counting. The n bits are assumed as input $A=A_{N-1},A_{N-2}\ldots A_0$ in the LZC, where A_{N-1} is the most significant bit and produces $\log_2 n$ bits of the leading-zero count Z and a flag V
that denotes the all-zero case for the input A. The existing method is based on the two step encoding procedure. The position of the leading digit of the input is marked first and then the remaining bits are set to zeros.

For example, let us consider the input as 00011010 the position of the leading digit is determined as 00010000. An intermediate string S is produced to derive the one hot representation. The bits of S that follow the leading digit are set to one and the other more significant bits are set to zero. For the same input the value of S is equal to 00011111. The i^{th} bit of S is denoted as S_i , $0 \le i \le n\text{-}1$ is defined as follows

$$S_i=A_{n-1}+A_{n-2}+\ldots..+A_{i+1}+A_i.$$

Each bit in the equation reveals the existence of at least one bit equal to 1 between the MSB and the i^{th} bit position. The one-hot representation of the leading digit (L word) is determined by detecting the case (0,1) for two consecutive bits of S.

$$L_i=\underline{S}_{i+1}.A_i$$

The value of L_{s-1} is equal to the value of S_{n-1}. Here (.) represents the logical AND and x represents compliment operations respectively. The priority encoder [1] computes the value of L. The encoder translates the number of leading zeros to its weighted binary representation when the L word is given to it. In the case of an 8-bit input operand the number of leading zeros, Z bits, are given by

$$Z_2=L_3+L_2+L_1+L_0$$
$$Z_1=L_5+L_4+L_1+L_0$$
$$Z_0=L_6+L_4+L_2+L_0$$

The all-zero flag V is set to S_0 which shows that no bit is equal to one in A. In dynamic CMOS floating-point-unit implementations this approach is mostly preferred. The leading-zero count is computed in few logic stages by

employing wide dynamic OR gates for the computation of both the S and the Z bits.

When the circuit for detection of leading zeros does not need to consider cases where leading ones might result [4], then the leading zero indicator can be simplified to the equation below

$$F=T_iZ_{i+1},\ i\geq 0$$

Here a comparison of the operands is performed to ensure that only the smaller operand is complemented during subtraction. Other designs where this could be applied would be where separate adders are provided for use when the exponents are equal. One adder calculates A-B and the other calculates **B-A,** and the result from the adder producing a carry out is selected. Each adder then needs only a leading zero detector. The detection of the number of leading zeros to start in parallel with swapping, aligning and inverting the operands. When the exponents differ by one, the presumed smaller operand is shifted right one place and then inverted. Since the operands must be normalized, the function in the first bit must be G, and therefore the number of leading zeros is determined by the number of following bit positions that are Zs .Therefore, the leading zero indicator in each following bit position is $\mathbf{F} = \mathbf{Z_{i+I}}$.

III. PROPOSED LZC AND LZA METHODS

This section discusses about the proposed leading zero counter and the mathematical equations are derived for the proposed leading zero counter. The Boolean relations that describe the bits of the leading-zero count are simplified. The proposed method will be presented using an example of an 8-bit LZC unit. Then the value of the encoded bits Z_0,Z_1 and Z_2 are

$$Z_2=\overline{S_4}.S_3+\overline{S_3}.S_2+\overline{S_2}.S_1+\overline{S_1}.S_0$$
$$Z_1=\overline{S_6}.S_5+\overline{S_5}.S_4+\overline{S_2}.S_1+\overline{S_1}.S_0$$
$$Z_0=\overline{S_7}.S_6+\overline{S_5}.S_4+\overline{S_3}.S_2+\overline{S_1}.S_0$$

For the value i>j the pair (S_i,S_j) will never take the value of (1,0) as the string S is monotonically increasing. As the string S monotonically increases we have $S_i.S_j=S_i$ and Si+Sj=Sj for i>j. The tabular column below shows the reduction when i>j.

Table 1:- Reduction Table

S_i	S_j	$S_i+S_j=S_j$	$S_i.S_j=S_i$
0	0	0	0
0	1	1	0
1	1	1	1

By using these equations the value of the encoded bits can be reduced further as

$$Z_2=\overline{S_4}.S_0$$
$$Z_1=\overline{S_6}.S_4+\overline{S_2}.S_0$$
$$Z_0=\overline{S_7}.S_6+\overline{S_5}.S_4+\overline{S_3}.S_2+\overline{S_1}.S_0$$

When we consider the above equations, no normalization is required when the input is equal to zero, the Z bits are also set to zero. We can also further simplify the above equations to get the below mathematical equations

$$Z_2=\overline{S_4}$$
$$Z_1=\overline{S_6}.S_4+\overline{S_2}$$
$$Z_0=\overline{S_7}.S_6+S_5.S_4+S_3.S_2+\overline{S_1}$$

Instead of assuming the value of the string S we can directly compute the leading zero counting from the input operand A. So the expression of Si is represented by the value of A_i. The final equations are thus derived as follows

$\underline{V}= A_7+A_6+A_5+A_4+A_3+A_2+A_1+A_0$

$Z_2=A_7+A_6+A_5+A_4$

$Z_1=(\underline{A_7+A_4})[(\underline{A_3+A_2})(A_7+A_6+A_5+A_4)]$

$Z_0=[\underline{A_7}.(\underline{A_7+A_6}).\underline{A_5})][(\underline{A_7+A_6})(\underline{A_5+A_4})][\underline{A_3+(A_3+A_2)A_1}]$

The proposed leading zero counter is shown in the figure below.

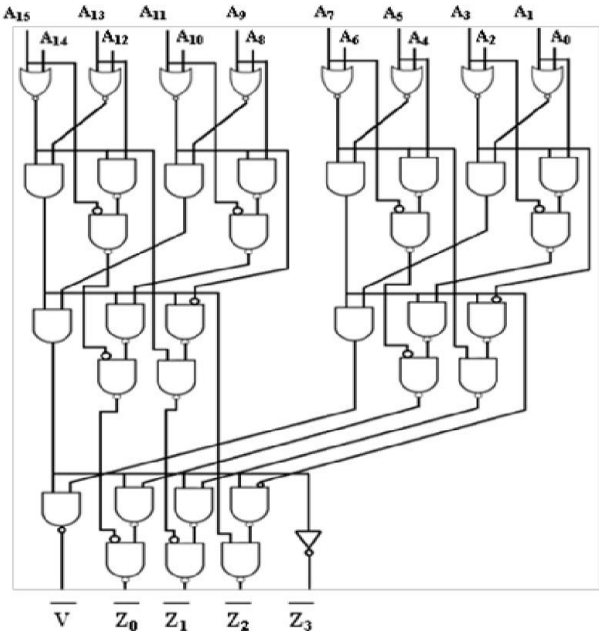


Figure 1: 16 bit im Fig: Implementation of proposed leading zero counter

The proposed LZA method is based on the restricted case that is A is always greater than or equal to B. According to this case A-B is always positive and A+B is also positive that is the result is always positive. Proposed Anticipation logic is based on generation of intermediate Difference and Borrow strings.

The proposed leading zero anticipation in addition can be explained by considering the addition example, consider A= 0010 0110 and B= 0000 1010, now the predicted string F_{add} is obtained as shown below.

A_{in} → 0010 0110

B_{in} → 0000 1010

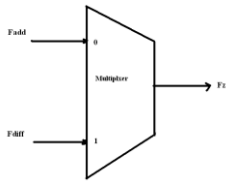
→ 0011 0000

By the above example we can find that number of leading zeros in F_{add} is equal to number of leading zeros in A, that is the number of leading zeros in result are always equal to or one less than operand A.

The subtraction in the proposed leading zero anticipation is explained considering the subtraction example, consider A= 0011 0110 and B= 0001 1010, now the propagate(P), borrow(B), difference(D) and the predicted string F_{sub} is obtained as shown below

A_{in}	=	0011 0110
B_{in}	=	0001 1010
P	=	0010 1100
B	=	0000 1000
D	=	0001 1100
F_{sub}	=	0011 1100

From the above example, the F_{sub} is proposed as $F_{sub}(0)$ is equal to P(0) and $F_{sub}[n]$ =P[n] ^ B[n-1] where B [n]=$\bar{A}$[n]·B[n]. The addition and subtraction are proposed by the Boolean algebra expressions as shown above. The proposed leading zero anticipation method is accurate and fast when compared with the existing leading zero anticipation method and overrides it in all aspects. To contain both the addition and subtraction a multiplexer is used to select the respective operation. The figure shows the proposed LZA.



Proposed LZA

The proposed LZA is more accurate than the existing LZA method this can be observed in following four cases. Any combination of input patterns falls into one of these patterns in the cases considered. In case 1 the number of leading zeros in the final result are 7 which is equal to the proposed LZA where the Existing LZA has 11 leading zeros in its result, in case 2 the number of leading zeros in the final result are 0 which is equal to the proposed LZA where the Existing LZA has 2 leading zeros in its result, in case 3 the number of leading zeros in the final result are 11, the proposed LZA has 10 leading zeros where the Existing LZA has 2 leading zeros in its result, in case 4 the number of leading zeros in the final result are 15 which is equal to the proposed LZA where the Existing LZA has 2 leading zeros in its result.

Case1:-

A_{in}	=	1000 0000 1010 0001
B_{in}	=	0111 1111 0110 1001
P	=	1111 1111 1100 1000
B	=	0111 1111 0100 1000
K	=	0000 0000 0001 0110
A- B	=	0000 0001 0011 1000
Proposed LZA	**=**	**0000 0001 0101 1000**
Existing LZA	**=**	**0000 0000 0001 001x**

Case 2:-

A_{in}	=	1001 1100 0010 0001
B_{in}	=	0001 1011 1100 1001
P	=	1000 0111 1110 1000
B	=	0000 0011 1100 1000
K	=	0110 0000 0001 0110
A+B	=	1011 0111 1110 1010
Proposed LZA	**=**	**1001 1100 0010 0001**
Existing LZA	**=**	**0011 1000 0001 001x**

Case 3:-

A_{in}	=	1001 1100 1010 0001
B_{in}	=	1001 1100 1000 1001
P	=	0000 0000 0010 1000
B	=	0000 0000 0000 1000
K	=	0110 0011 0101 0110
A-B	=	0000 0000 0001 1000
Proposed LZA	**=**	**0000 0000 0011 1000**
Existing LZA	**=**	**0011 1001 0101 001x**

Case 4:-

A_{in}	=	1001 1100 1010 1000
B_{in}	=	1001 1100 1010 0111
P	=	0000 0000 0000 1111
B	=	0000 0000 0000 0111
K	=	0110 0011 0101 0000
A-B	=	0000 0000 0000 0001
Proposed LZA	**=**	**0000 0000 0000 0001**
Existing LZA	**=**	**0011 1001 0101 000x**

Thus by the above mathematical derivations and the above cases show that the proposed methods of leading zero counting and leading zero anticipation logic can work effectively and the total delay is reduced compared to the

existing methods.

IV. IMPLEMENTATION RESULTS

The proposed methods of leading zero counting and leading zero anticipation are implemented on Spartan 3E family device XC3S250Epackage FT256 with speed -4 and the delay characteristics and the occupancy rates are compared with the existing method and tabulated below.

Sl.no.	Parameters	Existing Method	Proposed Method
1	No. of Slices	12/2448	10/2448
2	No. Of 4 Input LUTs	21/4896	17/4896
3	No. of bonded IOBs	21/172	21/172
4	Combinational Path Delay	14.293ns	11.189ns

.

Table1: Comparison of proposed LZC method with existing

Logic Utilization	**Proposed LZA**	**Available**
Number of Slices	32	4656
Number of 4 Input LUTS	32	9312
Number of bonded IOBs	97	232
Combinational Path Delay	6.347 ns	-

Table2: Comparison of proposed LZA method with existing

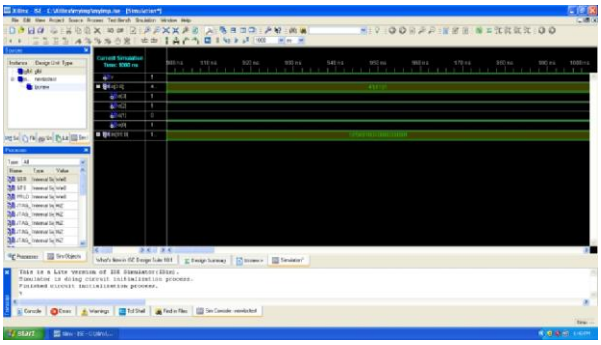


Figure 2:- screen showing the active low output of the proposed LZC for the given input 16 bit binary 0010000000000001

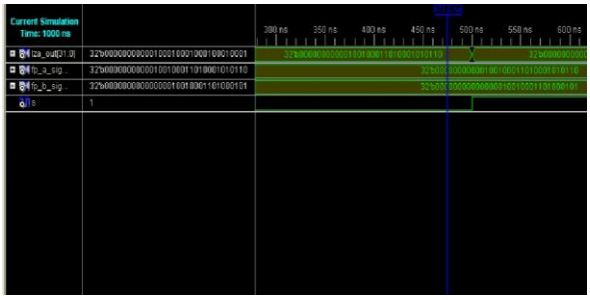


Fig2:- Simulation Result of Proposed LZA

The proposed methods are implemented using Veerilog HDL on Xilinx ISE 10.1 tool and the simulation results for both the proposed methods are displayed above.

V. CONCLUSION

Logic design for leading zero counter is proposed and the delay characteristics of the proposed LZC are significantly reduced when compared with the existing methods**.** The accuracy of the proposed LZA is significantly increased when compared with the existing methods. The design is implemented using Verilog HDL and verified using extensive directed-random vectors. The proposed leading zero counting and anticipation methods can be utilised in many application areas like RISC, CISC, Microprocessors and DSP. This proposed LZC and LZA can be effectively utilised in high speed floating point units.

REFERENCES

[1] Giorgos Dimitrakopoulos, *Member, IEEE*, Kostas Galanopoulos, Christos Mavrokefalidis, *Student Member, IEEE*,and Dimitris Nikolos, *Member, IEEE* "Low-Power Leading-Zero Counting and Anticipation Logic for High-Speed Floating Point Units" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 16, NO. 7, JULY 2009.

[2] E. Hokenek and R. Montoye, "Leading-zero anticipator (LZA) in the IBM RISC systed6000 floating-point execution unit," IBM **J.** Res. Develop., vol. 34, pp. 71-77, Jan. 1990.

[3] M. S. Schmookler and K. J. Nowka, "Leading zero anticipation and detection : A comparison of methods," in *Proc. 15th IEEE Symp. Comput. Arithmetic*, Jul. 2001, pp. 7–12.

[4] V. Oklobdzija, "An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis", IEEE Transactions on VLSI Systems, v. 2, no. 1, 1993.

[5] M. Schmookler and D. Mikan, "Two-state Leading Zero/One Anticipator (LZA), US Patent #5493520, Feb. 1996