International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 1, No. 1, March 2012, pp. 19~24
ISSN: 2089-4864

A Novel FPGA based Leading One Anticipation Algorithm for Floating Point Arithmetic Units

Ashwini S. Deshmukh
Department of Electronics and Telecommunication, Bhagawati Chaturvedi College of Engineering, Nagpur

Article Info

Article history:

Received Jan 22, 2012
Revised Mar 18, 2012
Accepted Mar 23, 2012

Keyword:

floating point arithmetic units
IEEE-754
leading one anticipation
multimedia
systems-on-Chips

ABSTRACT

In multimedia Systems-on-Chips, the design of specialized IEEE-754-compliant floating point arithmetic units (FPU) is critical with respect to both operating speed and silicon area demand. Leading one anticipation is a well-known issue in the implementation of high speed FPUs. We investigated a novel leading one anticipation algorithm allowing us to significantly reduce the anticipation failure rate with respect to the state-of the art. We embedded our technique into a complete FPU and compared its performance against existing solutions, definitely showing both area savings and total latency reduction.

Corresponding Author:

Miss. Ashwini S. Deshmukh
Department of Electronics and Telecommunication,
Bhagawati Chaturvedi College of Engineering, Nagpur
c/o Mr. Hitesh M. Tupkari, 148 Dattatraya Nagar, Nagpur- 24
Phone No: 8793662135
Email: as_deshmukh@rediffmail.com

1. INTRODUCTION

The increased interest in ASICs and SoCs for multimedia applications justifies the effort in the design of high-speed arithmetic units where either fixed or floating-point real number representations are exploited for calculations. In particular, high-throughput floating-point signed adders (FADD) lend themselves to a sophisticated design approach due to both the amount and complexity of the required operations (operands alignment, addition, renormalization and rounding). In case of effective subtraction (subtraction of operands weith the same sign or addition of operands with different signs), the number of leading digits in the output must be calculated to initiate the renormalization procedure. Since renormalization occupies the dominant portion of an FP addition, performing the leading digits calculation after the addition is ineffective and some reliable anticipation logic must be devised for parallel anticipation of the position of the first significant digit. For practical implementations, when both leading zeroes and ones may occur, either case can be handled by a separate anticipator [2][3].

In modern hardware applications, latency is the most important parameter for most operations. While designing for FPGA devices with millions of reconfigurable hardware gates, area can be given up to gain overall latency. Pipelining is another technique used to increase the overall throughput of the adder but doesn’t decrease the latency. In this chapter, pipelining and its effects on far and close data-path adder implementation will also be discussed.

2. LEADING ONE PREDICTOR ALGORITHM

In this section, LOP implementation and its use in floating-point addition is described in detail. LOP module is also used in far and close data-path algorithm implementation. LOP is used to predict the leading one in parallel with the adder computation. This decreases the number of logic levels in the critical path and results in an overall improvement in the latency.

2.1. Leading One Predictor

The LOP consists of two operations [4, 5]:

- Pre-encoding the inputs
- Leading one detection

Parallel, leading one detection and correction adds area to the design but decreases the overall latency of the module by reducing levels of logic. For a single-precision floating-point addition operation, a 24-bit LOP is needed. A and B are the two inputs, the leading one predictor designed is for A > B and/or A < B. This is needed because even after pre-normalization, the resulting addition can be negative as in case of A < B. When the effective operation is addition, the LOP result is not used and the shift value is always zero. The general structure is shown in Figure 1.

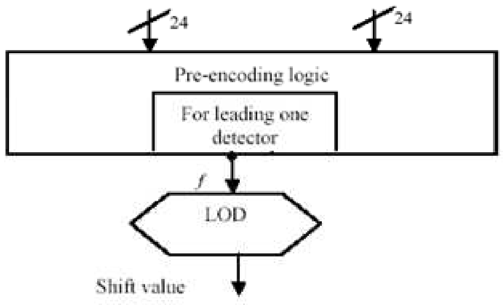


Figure 1. General structure of LOP

2.2. Pre-Encoding Logic

The pre-encoding module gives a string of 0s and 1s with the location of the first one being the leading one position. After this, the leading one detector already designed for standard algorithm is used to compute the position of the most significant bit. This is the amount we need to shift for normalization, in case of no correction. The pre-encoding module also gives strings of symbols, which are passed through the binary tree to determine if correction is needed. A one is added to the leading one position if a certain string of symbols is identified. In order to get the string of 0s and 1s, to pass through the LOD.

2.3. Leading One Detector

After the addition, the next step is to normalize the result. The first step is to identify the leading or first one in the result. This result is used to shift left the adder result by the number of zeros in front of the leading one. In order to perform this operation, special hardware, called Leading One Detector (LOD) or Leading Zero Counter (LZC), has to be implemented.

There are a number of ways of designing a complex and complicated circuit such as LOD. A combinational approach is a complex process because each bit of the result is dependant on all the inputs. This approach leads to large fan-in dependencies and the resulting design is slow and complicated. Another approach is using Boolean minimization and Karnaugh map, but the design is again cumbersome and unorganized.

The circuit can also be easily described behaviorally using VHDL and the rest can be left to QuartusII or any synthesis tool. In our floating-point adder design, we used the LOD design which identifies common modules and imposes hierarchy on the design. As compared to other options, this design has low fan-in and fan-out which leads to area and delay efficient design [6] first presented by Oklobdzija in 1994.

2.3.1. Oklobdzija's LOD

The first step in the design process is to examine two bits case shown in Table1.The module is named as LOD2. The pattern shows the possible combinations. If the left most bit is 1, the position bit is assigned 0 and the valid bit is assigned 1. The position bit is set to 1 if the second bit is 1 and the first bit is 0. The valid bit is set to 0 if both the bits are 0.

The logic for LOD2 is straightforward and shown in Figure 2. The same concept is used to implement LOD8, LOD16, and LOD32 modules. Using the above designed modules, the leading one detector takes the shape as shown in Figure 3. The input to the LOD for single-precision floating point adder is the first 24 bits of the result coming out of the integer adder. Twelve LOD2's, six LOD4's, three LOD8's, two LOD16's, and one LOD32 are implemented in parallel to get the position of the first leading one in the adder result.

Table 1. Truth table for LOD2

Pattern	Position Bit	Valid Bit
1x	0	1
01	1	1
00	x	0

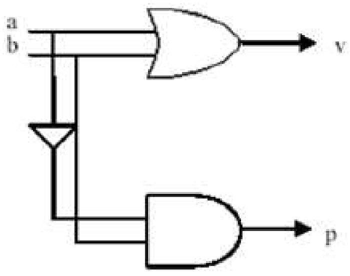


Figure 2: Hardware implementation for LOD2

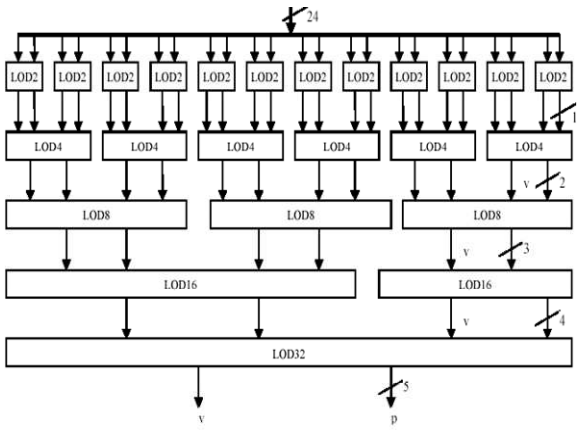


Figure 3: LOD implementation

The behavioral model was implemented using 'case' statements in VHDL defining each possibility behaviorally.

2.3.2. Left Shift Barrel Shifter

Using the results from the LOD, the result from the adder is shifted left to normalize the result. That means now the first bit is 1. This shifter can be implemented using "shl" operator in VHDL or by describing it behaviorally using 'case' statements.

The behavioral model had a negligibly smaller combinational delay, and smaller area, and is therefore used in our implementation. For a single precision floating-point adder the maximum amount of left shift needed is 27. The hardware for the behavioral left shifter is designed to only accommodate the maximum shift amount. As we have no control over the hardware implementation in VHDL shifter, it implements hardware for shift amounts greater than 27, thus resulting in bigger area and delay compared to behavioral shifter. In this case this operation is done in parallel to addition and thus decreases levels of logic in the critical path.

2.4. Post-normalization

In case of far path, only 1 bit left or right shift is needed. 1-bit right shift is done if the result from the adder is not normalized and the effective operation is addition. On the other hand, 1 bit left shift is needed if the most significant bit of the result is 0 and subtraction was carried out. This is a big reduction in the critical path in terms of latency when compared to standard or LOP algorithm implementations. In case of right shift, one is added to the larger exponent to obtain exponent out, on the other hand, one is subtracted from the larger exponent in case of left shift.

3. RTL VIEW:

The RTL View of the Leading One Predictor and Post–normalization block is shown in Figure 4. The Floating Point Adder is designed and implemented using QuartusII by Device: ‘Cyclone: EP1C4F400C8’

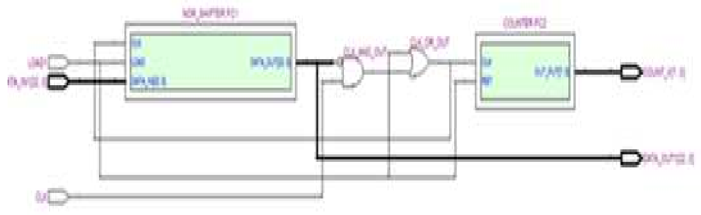


Figure 4. RTL View of the Leading One Predictor and Post–normalization block

4. TIMING AND AREA ANALYSIS

The LOP floating point adder is synthesized, placed and routed for Cyclone: EP1C4F400C8 FPGA device using QuartusII. The minimum clock period reported by the synthesis tool after placing and routing is 3.636 ns. The levels of logic cells reported are 32/4,000 (<1%). That means the maximum clock speed that can be achieved for this implementation is 275.03 MHz. A Timing and Area Analysis report for LOP algorithm implementation is shown in Table 2 and 3.

Table 2. Timing Analysis (Cyclone: EP1C4F400C8)

Entity	Clock Period (ns)	Clock Speed(MHz)
LOP Algorithm design	3.636 ns	275.03 MHz

Table 3. Area Analysis (Cyclone: EP1C4F400C8)

Entity	Logic Cells	LC Registers	Pins	Registors-Only LCs	LUT/Register LCs	Carry Chain LCs
LOP Algorithm design	32/4,000 (<1%)	32	56/301 (18%)	22	10(0)	8 (0)

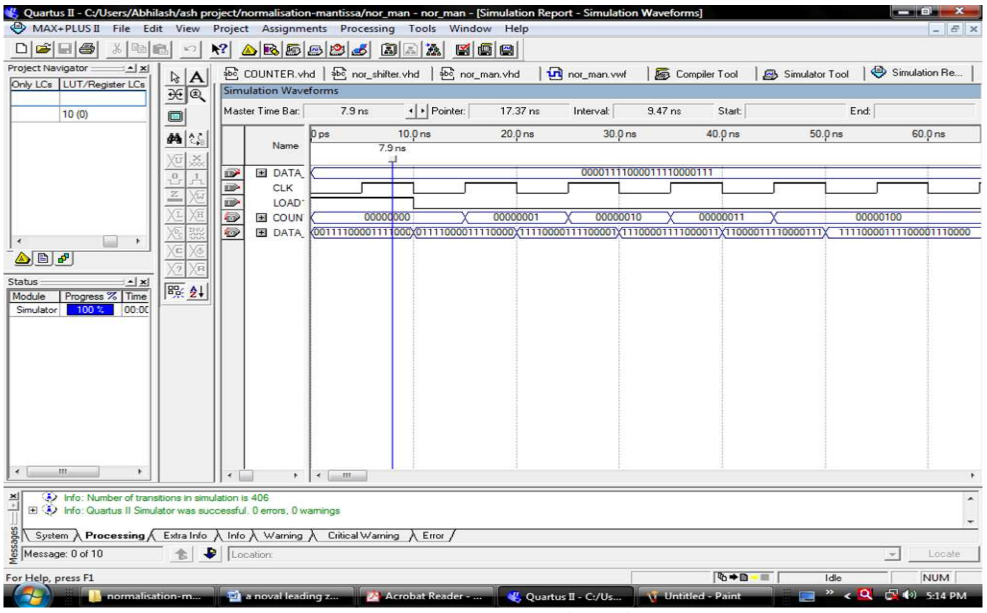
5. IMPLEMENTATION

The complete circuit of LOP is described in VHDL, as a structural component. Then hierarchical structure is created and the complete simulation can be observed. Figure 5 shows the waveform simulation result for the LOP algorithm. Wave form shows the leading one detection along with the normalization of the number. Also it gives count value of the number of shift for exponent that is the result of post normalization.

6. FIVE STAGE PIPELINE LOP FLOATING POINT ADDER IMPLEMENTATION

The leading one predictor algorithm is pipelined into five stages. Figure 6 shows the micro-architecture of the five stage pipeline implementation of LOP floating-point adder algorithm. The dotted lines show the registers used to induce stages between logic.

In the first stage of the implementation the two operands are compared to identify denormalization and infinity. Then the two exponents are subtracted to obtain the exponent difference and identify whether the operands need to be swapped using the exponent difference sign. In the second stage the right sifter is used to pre normalize the smaller mantissa. In the third stage the addition is done along with leading one prediction. In the fourth stage left shifter is used to post normalize the result. In the last stage the exponent out is calculated and rounding is done. The results are then compared to set overflow or underflow flags.



```
Quartus II - C:/Users/Abhilash/ash project/normalisation-mantissa/nor_man - nor_man - [Simulation Report - Simulation Waveforms]
Master Time Bar: 7.9 ns   Pointer: 17.37 ns   Interval: 9.47 ns   Start:   End:

DATA_   00001111000011110000111
CLK
LOAD
COUN    00000000 | 00000001 | 00000010 | 00000011 | 00000100
DATA_   0011110000111100 | 0111100001111000 | 1111000011110000 1 | 1110000111100001 1 | 1100001111000011 1 | 111100001111000011 10000

Info: Number of transitions in simulation is 406
Info: Quartus II Simulator was successful. 0 errors, 0 warnings
Message: 0 of 10
```

Figure 5. Simulation Waveform for LOP

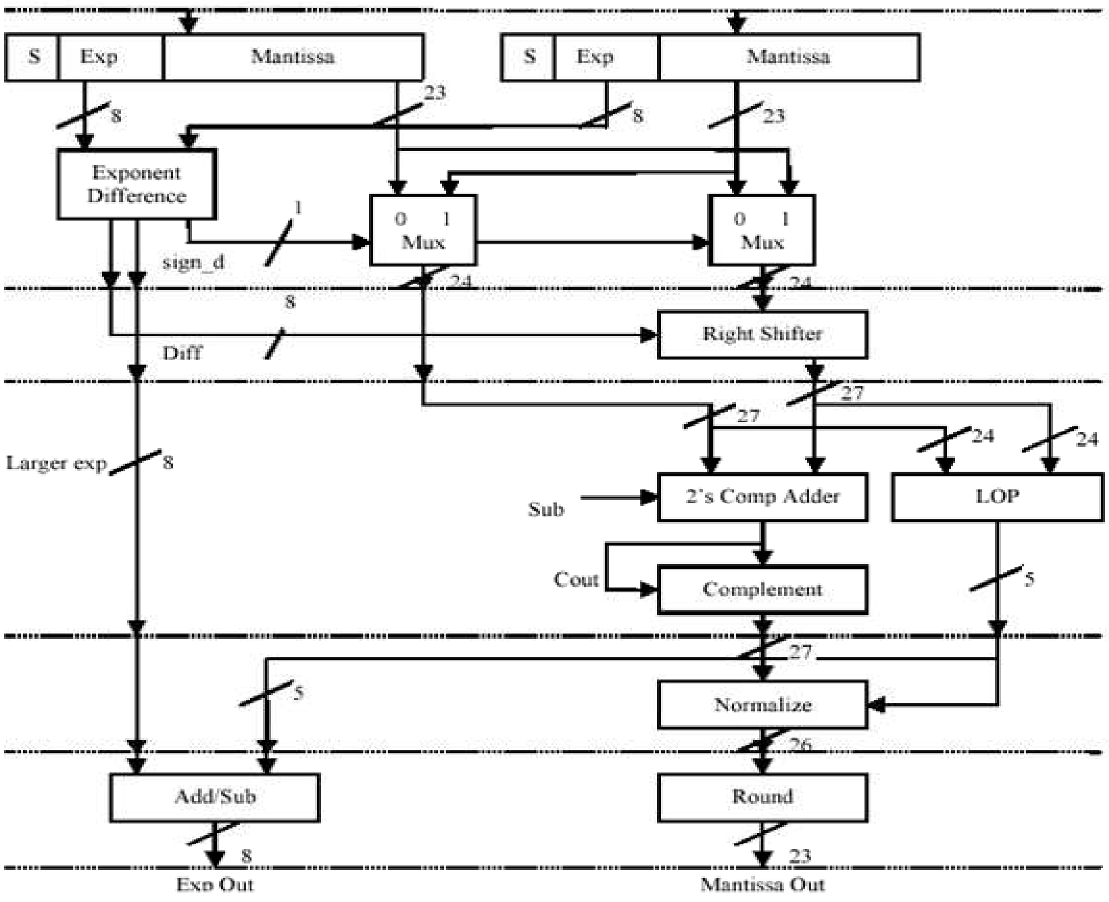


Figure 6. Micro-architecture of 5 stage pipeline LOP floating-point adder

7. CONCLUSION

In this paper, a Novel FPGA based Leading One Anticipation Algorithm for Floating Point Arithmetic Units is discussed. This algorithm gives designer the choice to select appropriate implementation according to their design needs. Standard floating-point addition algorithm is meant for area efficient designs where designers are working with small FPGA devices with very few equivalent logic gates to fit in their design. For latency efficient designs where the main aim is to have faster execution units, LOP algorithm is the best option. LOP is running at around 275 MHz for five pipeline stages is faster and better for implementation. A novel leading one anticipation algorithm allowing us to significantly reduce the anticipation failure rate with respect to the state-of the art.

REFERENCES

[1] Standard for binary floating-point arithmetic, IEEE Standard 754, 1985.

[2] S. Britton, R. Allmon, and S. Samudrala, “Leading one/zero bit detector for floating-point addition,” U.S. Patent 5317527, May 1994.

[3] M.S. Schmookler, and D. Mikan, “Two-state leading zero/one anticipator (LZA),” U.S. Patent 5493520, February 1996.

[4] J. D. Bruguera and T. Lang, “Leading-One Prediction with Concurrent Position Correction,” IEEE Transactions on Computers, pp. 1083–1097, 1999, vol. 48, no.10.

[5] M. J. Flynn, “Leading One Prediction -- Implementation, generalization, and application,” Technical Report: CSL-TR-91-463, March 1991.

[6] V. G. Oklobdzija, “An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis.” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 124-128, 1994, Vol. 2, No. 1.

[7] T. Chang, J. Huang, and S. Yang, “Leading-zero anticipatory logics for fast floating addition with carry propagation signal,” *Proc. 40th Midwest Symposium on Circuits and Systems*, 1997, vol. 1, pp. 385-388.

[8] H.P. Sit, et al., “Prenormalization for a floating-point adder,” U.S. Patent 5010508, April 1991

[9] S. Oberman, and M. Roberts, “Leading one prediction unit for normalizing near-path subtraction results within a floating point arithmetic unit,” U.S. Patent 6085208, July 2000.

[10] M.S. Schmookler, and K.J. Nowka, “Leading zero anticipation and detection – a comparison of methods,” *Proc. 15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 7-12.

[11] H. Sun, W. He, and M. Gao, “Designing leading zeros anticipatory logic based on production rules,” *Proc. 5th International Conference on ASIC*, 2003, vol. 2, pp. 1260-1264. H. Sun, and M. Gao, “Unified bit pattern for leading-zero anticipatory logic for high-speed floating-point addition,” *Proc. 3rd IEEE International Symposium on Signal Processing and Information Technology*, 2003, pp. 786-789.