# Double Precision Optimized Arithmetic Hardware Design for Binary & Floating Point Operands

Vikram Palodiya[#1], Hemant Ghayvat[#2], D.S Ajnar[#3], Pramod Kumar Jain[#4].
[#1, 2, 3, 4.] *Micro Electronics and VLSI design*
*** Electronics & Instrumentation Engineering department, SGSITS, Indore, MP*
*[#1] pal.vicky2008@rediffmail.com, [#2]ghayvat@gmail.com, [#3]dsajnar@gmail.com,*
*[#4]pramod22_in@yahoo.com*

***ABSTRACT - In today's scientific changes incident and rapid growth in financial, commercial, Internet-based applications, there is a huge demand for finding out the devices with low latency, power and area along with there is an growing need to allow computers to operate on both binary and decimal floating-point numbers. Accordingly, stipulation for decimal floating-point support is being added to the IEEE-754 Standard for Floating-Point Arithmetic. In this research work, we present the design and implementation of a decimal floating-point adder that is acquiescent with the current draft revision of this standard. The adder supports operations on 64-bit (16-digit) decimal floating-point operands [1] .We provide synthesis results indicating the estimated area and delay for our design when it is pipelined to various depths.***

***Keywords - Mantissa, Exponent, Sign bit, Operands, Latency, Underflow, and Overflow.***

## I. INTRODUCTION

Various high level or gate level programming languages have a potential for specifying floating -point numbers. The most frequent technique is to stipulate them by a real declaration statement as conflicting to fixed -point numbers, which are specified by an integer declaration statement. Any computer that has a compiler for handling floating point arithmetic operations, the operations are quite often included in the internal hardware. If no hardware resources available for that particular operation, the compiler must be designed with a package of floating point software subroutines (logic code which may be of a few line or thousands of lines, require repetitively).the hardware resource utilization method is more expensive, but it is so much more efficient than the software resource utilization method .the floating point hardware is incorporated approximately in all computer system and omitted only in very small ones [4]. Example of floating point hardware devices are Intel 8231, LPC3180 (it is an ARM9-based microcontroller for embedded applications requiring

High performance combined with low power dissipation.) and AMD's AM9512 floating point processor. This type of processor unit provides add, subtract, multiply, and divide operation for 32-bit and 64-bit operands. It can easily interface to enhance the computational capabilities of the host CPU because of this no need to change existing system resources, they have great adaptability.

The design performs addition and subtraction on 64-bit operands and can be pipelined to achieve substantial improvements in its critical delay path [5]. It can also be extended to support operations on 32-bit and 128-bit decimal floating-point numbers. Related work on decimal arithmetic includes designs for fixed-point decimal adders [6-7] and floating-point decimal processors [8-9]. Formats are specified for decimal floating-point numbers having widths of 32, 64, and 128 bits, which correspond to significant of 7, 16, and 34 decimal digits, respectively [5].

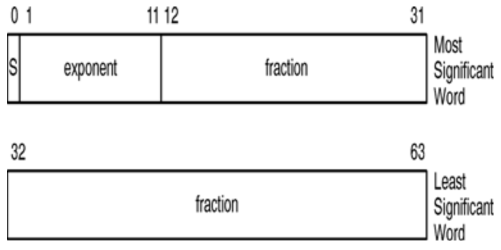


**Figure 1.1**

## II. METHODOLOGY

The register is the important term for these type of operation and their excecution,it is the group of flip flops where each one have storing potential of one bit at a time. For a particular logic of work the configuration of register is different .the register configuration for floating point operations is approximately similar to the arrengement for fixed point operations. They both excecute mantissa operation simillar way but the difference takes place in operations related to exponents ,the same ragisters and adder used for fixed point arithmetic are usd for processing the mantissas. The registere association for floating point operation as given in figure.it contains three different field: First for sign bit ,Second for mantissa in uppercase letter and Last one for exponent in lowercase letter.

The term A of AC represent mantissa,whose sign is given by $A_s$ and a magnitude that is in A.as we have taken exponent is in the part of the register denoted by the lowercase letter symbol **a.**.most significant bit of A ,labeled by $A_1$.bit in this position should be 1 for normalized.

Similarly role,register BR is association into $B_1$ ,$B_s$ and b , and QR into $Q_1$ ,$Q_s$ and q. a parralel adder perform the operation based on the value of the two mantissas and trasfers the resultant into A and the carry into E. A saprete paralle adder is used for the exponents.

Since the exponents are biased ,they do not have a distinct sign bit but are represented as a predisposed positive magnitude .it is understood that the floating point numbers are so large that the chance of an exponent overflow is very less, and for this reason the exponent overflow deserted . the exponents are also linked to a magnitude comparator that provide three binary outputs to designate their relative magnitude.

The numeral is in mantissa will be consider as a fraction, so the binary point is assumed to exist in to the left of the magnitude part.

The records in the registers are understood to be primarily normalized but after each arithmetic operation,the result will have to normalized. So the process of normalization require repeatetively after each and every operation.Thus all floating point operands coming from and going to the memory unit are always normalized [4].

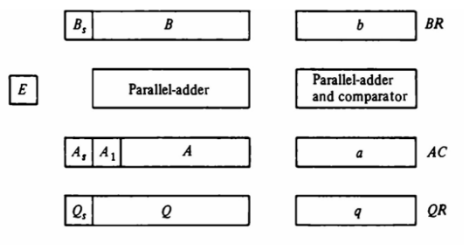


**Figure 1.2 1**

***Addition operation***

This operation is an addition of the two floating point operands data values (they are binary for machine). Two double-precision floating point inputs are added. Before the addition the exponent part of both the input will have to be equal in magnitude, then it is eligible for addition operation which takes place mainly between the mantissa parts of the floating point number is added to each other .After implementing this logic by the executing the instruction, result will be in the form of three parts. Sign, mantissa and exponent. The result here is shown in the sign, sum and exponent The operand A is in decimal value is 2.2700000000e-001 and the operand B in the decimal value is 3.4000000000e+001. The addition of two double-precision floating point number is 3.422700000000000e+001. This shows the resultant in the sign, sum, and exponent**.**

***Subtraction operation***

These two operations addition and subtraction are quiet same. Both the operations are very useful in the complex systems where complex addition and subtraction are being done. After executing the instruction, the result is generated in three different parts like sign, exponent, and mantissa.

The result here is shown in the sign, diff and exponent. The decimal value of the operand A is 4.6500000000e+002, and the operand B is 6.5000000000e+001. The subtraction of these two numbers will result in a floating point number. The number in the decimal format is 4.000000000000000e+002. The resultant output can be shown in the fig. is sign, diff, and exponent.

***RTL View of Simulation:***

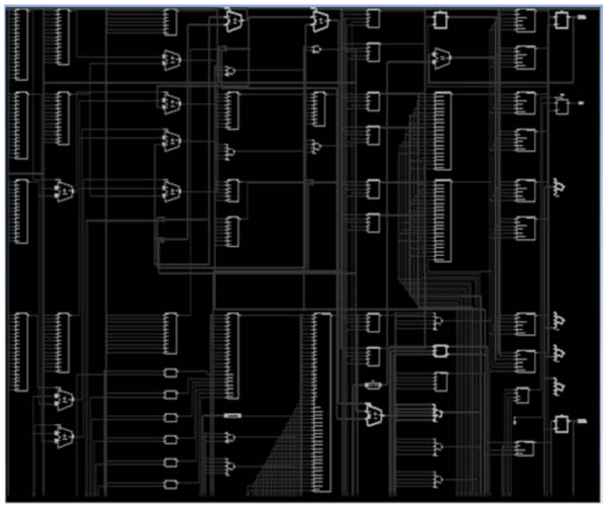


**Figure 1.3**

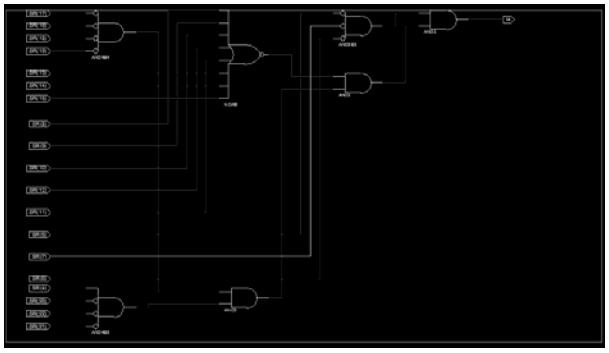


**Figure 1.4**

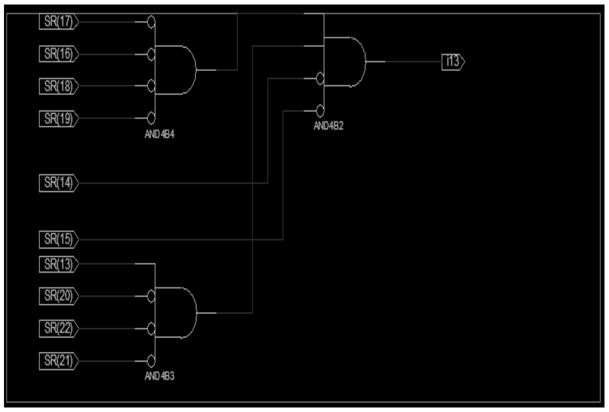
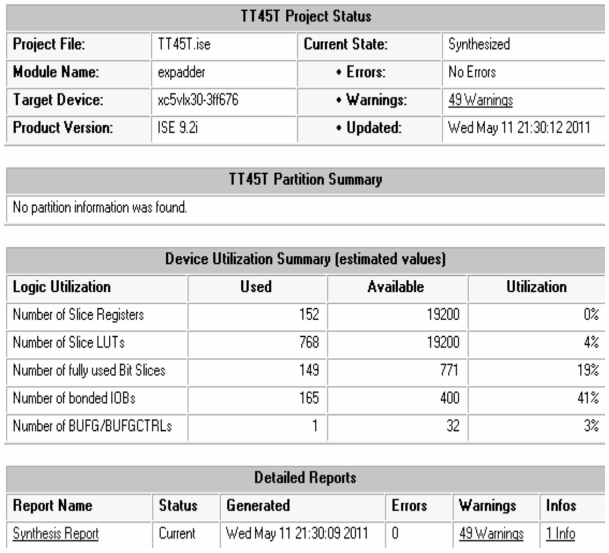


**Figure 1.5**



| TT45T Project Status | | | |
|---|---|---|---|
| **Project File:** | TT45T.ise | **Current State:** | Synthesized |
| **Module Name:** | expadder | **• Errors:** | No Errors |
| **Target Device:** | xc5vlx30-3ff676 | **• Warnings:** | 49 Warnings |
| **Product Version:** | ISE 9.2i | **• Updated:** | Wed May 11 21:30:12 2011 |

| TT45T Partition Summary |
|---|
| No partition information was found. |

| Device Utilization Summary (estimated values) | | | |
|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slice Registers | 152 | 19200 | 0% |
| Number of Slice LUTs | 768 | 19200 | 4% |
| Number of fully used Bit Slices | 149 | 771 | 19% |
| Number of bonded IOBs | 165 | 400 | 41% |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% |

| Detailed Reports | | | | | |
|---|---|---|---|---|---|
| **Report Name** | **Status** | **Generated** | **Errors** | **Warnings** | **Infos** |
| Synthesis Report | Current | Wed May 11 21:30:09 2011 | 0 | 49 Warnings | 1 Info |

**Figure 1.6**

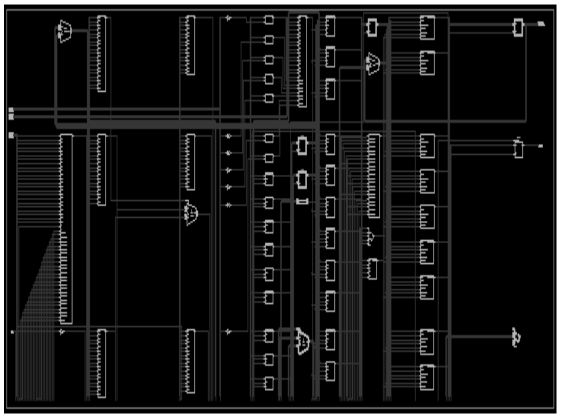


**Figure 1.7**

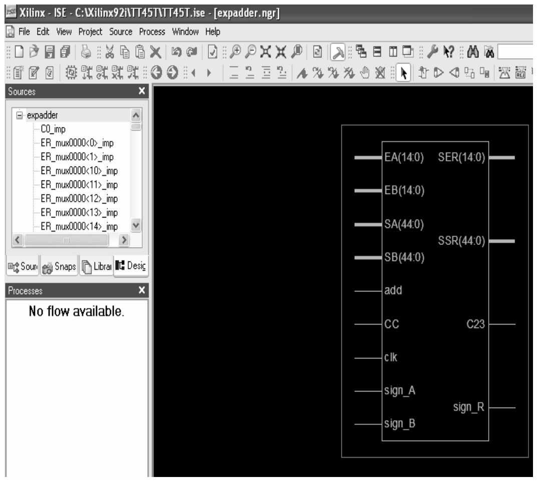


**Figure 1.8**

## Waveform Generation For Addition-Subtraction

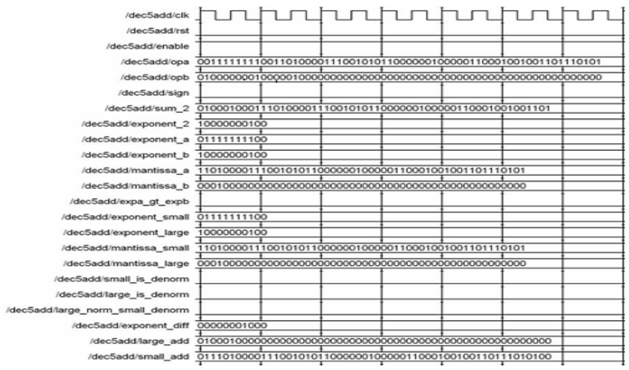


**Figure 1.9**

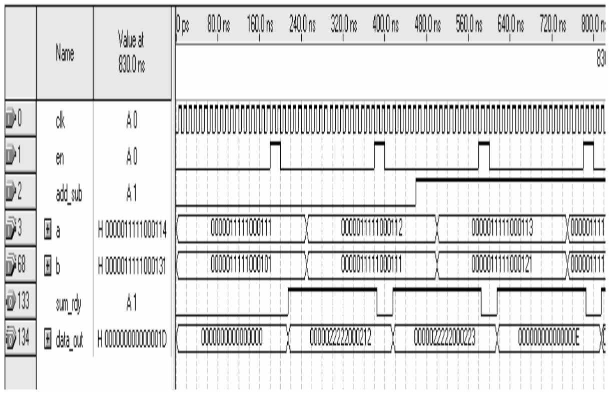


**Figure 1.10**

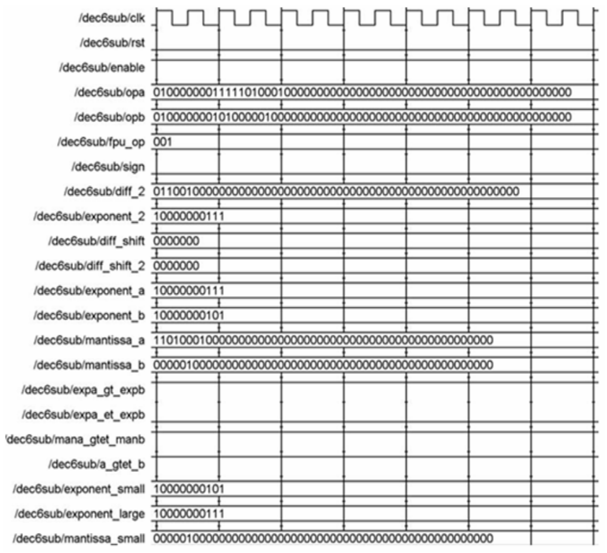

/dec6sub/clk
/dec6sub/rst
/dec6sub/enable
/dec6sub/opa 0100000001111101000100000000000000000000000000000000000000000000
/dec6sub/opb 0100000001010000010000000000000000000000000000000000000000000000
/dec6sub/fpu_op 001
/dec6sub/sign
/dec6sub/diff_2 0110010000000000000000000000000000000000000000000000000
/dec6sub/exponent_2 10000000111
/dec6sub/diff_shift 0000000
/dec6sub/diff_shift_2 0000000
/dec6sub/exponent_a 10000000111
/dec6sub/exponent_b 10000000101
/dec6sub/mantissa_a 11010001000000000000000000000000000000000000000000000
/dec6sub/mantissa_b 00000100000000000000000000000000000000000000000000000
/dec6sub/expa_gt_expb
/dec6sub/expa_et_expb
'dec6sub/mana_gtet_manb
/dec6sub/a_gtet_b
/dec6sub/exponent_small 10000000101
/dec6sub/exponent_large 10000000111
/dec6sub/mantissa_small 00000100000000000000000000000000000000000000000000000

**Figure 1.11**

***Subtraction***

## III. CONCLUSION

A floating-point arithmetic module with an optimized area and speed is presented. The effect of normalization on the area and speed has been examined experimentally. The design has been mapped on tools like Xilinx, Altera and optimized on tools like Microwind, Cadence. The presented Double -precision floating-point adder, and subtractor modules run at slightly faster clock speed with used area less than that used previously. . The proposed design has 11.2 ns delay; number of slice LUT used 936, number of bonded IOBs 187., Minimum Period 10.567 ns ,Maximum Frequency 168.236 MHz ,Minimum Input Arrival Time before Clock 11.097 ns,Maximum Output Required Time after Clock 12.536ns, Gate Delay (Logic) 3.937ns ,Net Delay (Route) 5.266 ns.

## IV. REFERENCES

[1] Thompson, Nandini Karra, and Michael J. Schulte “A 64-bit Decimal Floating-Point Adder” , IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design (ISVLSI’04)2004 IEEE.

[2] Hajime Kubosawa, Akira Katsuno, Hiromasa Takahashi, Tomio Sato, Atsuhiro Suga and Gensuke Goto, “A 64-bit Floating Point Processing Unit for a RISC Microprocessor”, Fujitsu Laboratories Ltd.10-1, Morinosato-Wakamiya, Atsugi 243-01, Japan 1992 IEEE

[3] HDL implementation and analysis of a residual register for A floating-point arithmetic unit By Akil kaveti March 25, 2008 Dr. William r. Dieter Director of thesis Dr. Yuming zhang Director of graduate studies

[4] Mano, Morris M., “COMPUTER SYSTEM ARCHITECTURE”.

[5] “Draft IEEE Standard for Floating-Point Arithmetic”, IEEE, inc., New York, 2003. Available from: Http://794r.ucbtest.org/drafts/794r.pdf.

[6] M.S .Schmookler and A.W. Weinderger, “High Speed Decimal Addition”, IEEE Transactions on. Computers, Vol. C-20, pp. 862-867, August 1971.

[7] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and Steven R. Carlough, “The IBM 900 Decimal Arithmetic Unit”, Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers, Vol. 2, pp. 1335-1339, IEEE, November 2001.

[8] G. Bohlender and T. Teufel, “A Decimal Floating-Point Processor for Optimal Arithmetic”, Computer arithmetic: Scientific Computation and Programming Languages, pp. 31-58, 1987.

[9] M. S. Cohen, T. E. Hull, and V. Carl Hamacher, “CADAC: A Controlled-Precision Decimal Arithmetic Unit”, IEEETransactions on Computers, Vol. 32, No. 4, pp. 370-377, IEEE, April 1983.

[10] RA. Kaivani A. Zaker aihosseini S. Gorgin M. Fazlali “Reversible Implementation of Densely-Packed-Decimal Converter to and from Binary-Coded-Decimal Format Using in IEEE-754” , Department of Electrical and Computer Engineering Shahid Beheshti University, Tehran, lran.

[l1] M. F. Cowlisha, “Decimal Floating-Point: Algorism for Computers”, Proceedings of the 16th IEEE Symposium on Computer Anathematic, pp. 104-1 1 1, June 2003.

[12] Fadi Y. Busaba et al., “The D3M 2900 Decimal Anthmetic Unit”, IEEE Trans. on Computers, Vol. 2, pp. 1335-1339, Nov.2001.

[13] Andre Guntoro and Manfred Glesner “High-Performance Fpga-Based Floating-Point Adder with Three Inputs” 2008 IEEE.

[14 ]Subhash Kumar Shrama, Himanshu Pandey, Shailendra Sahni and Vishal Kumar Srivastava “Implementation of IEEE-754 Addition and Subtraction for Floating Point Arithmetic Logic Unit”.