Singaporean Journal Scientific Research (SJSR)
ISSN: 2231 - 0061Vol.4, No.1pp. 218 - 220

available at: : http://www.iaaet.org/sjsr

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

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

***ABSTRACT* – In today's modern scientific world, technological changes happening with a very fast rate. The 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 increasing desire 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.We provide synthesis results indicating the estimated area and delay for our design when it is pipelined to various depths. High performance computing is strongly required in most applications which deal with floating point Numbers. Most workstations used in these fields are adopting a floating point processing unit to accelerate Performance.**

Keywords –Binary, Floating Point, Hardware design

I. INTRODUCTION

Various high level encoding languages have a capability 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 available for operation, the compiler must be designed with a package of floating point software subroutines (program or line of logic written once, uses more than once). Although the hardware method is more expensive, it is so much more efficient than the software method that floating point hardware is included in most computers and omitted only in very small ones[4]. Example of floating point hardware devices are Intel 8231, arithmetic processor and AMD's AM9512 floating point processor. The AM9512 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.

2. PROBLEM OVERVIEW

The user of computer prepares data with decimal numbers and receives results in decimal form. A CPU with an arithmetic logic unit can perform micro operations with binary data. To perform arithmetic operations with decimal data, it is necessary to convert the input numbers to binary, to perform all calculations with binary numbers, and to convert the results into decimal. This may be an efficient method in applications requiring a large number of calculations and relatively smaller amount of input and output data [14]. But user friendly format in which it allows input output operations in decimal but machine operation in binary, so the input decimal data have to convert in binary and in later part after operation converted back to decimal. But it is quite time consuming.

When the application calls for a large amount of input-output and a relatively smaller number of arithmetic calculations, it becomes convenient to do the internal arithmetic directly with the decimal numbers. Computers capable of performing decimal arithmetic must store the decimal data in binary coded form. The decimal numbers are then applied to a decimal arithmetic unit capable of executing decimal arithmetic micro operations.

3. PROBLEM FORMULATION

We implement divide and conquer approach, in which complex logic operations segmented or implementation into various multiple numbers of arithmetic logic blocks .These blocks works independently or dependently means, output of one is input of others. Then, we optimize whole unit, explore boundaries and tradeoff between speed, power, area. Electronic calculators invariably use an internal decimal arithmetic unit since inputs and outputs are frequent, not seem to be a reason for converting the displayed results to decimal. Since this process requires special circuits and also takes a longer time to execute. Many computers have hardware for arithmetic calculations with both binary and decimal data. Users can specify by programmed instructions whether they want the computer to perform calculations with binary or decimal data. The unit accepts coded decimal numbers and generates results in the same adopted binary code [14].

4. RESULT

Synthesis Reports : The corresponding circuit hardware realization is carried out by a synthesis tool.

Simulation Reports: The design descriptions are tested for their functionality at every level – behavioral, data flow, and gate. One has to verify here whether all the functions are carried out as expected and resolve them. All such performance is carried out by the simulation tool. The tool also has an editor to carry out any corrections to the source

code. Simulation involves testing the design for all its functions, functional sequences, timing constraints, and specifications.

Table 1: Arithmetic Unit for Double Precision Optimized Arithmetic Hardware Design for Binary & Floating Point Operands: Hardware Resource Utilization Summary targeting on xc4vlx40-12ff1148 device

SI. No	Device Parameter	Usage Number	Utilization %
1	Number of slices	320	1%
2	Number of slices Flip-Flop	450	1%
3	Number of 4-input LUTs	502	1%
4	Number of IOs	196	-
5	Number of bounded IOBs	196	30%
6	Area Constraint Ratio	5	-
7	Total memory usage	281640kb	-

Table 2 Arithmetic Unit for Double Precision Optimized Arithmetic Hardware Design for Binary & Floating Point Operand: Comparison in terms of Area & Delay.

Name of Design proposals	Number of bits	Delays (ns)	Area	
			Slices	LUTs
Sreehari [28]	32	8.9	-	523
Humberto[30]	32	12.1	256	495
Haller [29]	32	10.0	305	584
Hwang [27]	32	10.5	82	158
Fischer [31]	32	10.3	123	233
Subhash [14]	64	-	2325 (Addition) /2119(Subtraction)	-
Taher[32]	64	11.24	-	-
P.Karlstrom [18]	64	-	561	675
Our Proposal	64	11.2	320	502

5. CONCLUSION

This research purposed a mixture of hardware compilation, module generators, Floating point arithmetic and automatic interface generation to improve the efficiency, productivity and flexibility when implementing the floating point design on the FPGA. This dual representation is very valuable as allows for easy navigation over all the components of the units, which allows for a faster understanding of their interrelationships and the different aspects of a Floating Point operation. There are several possibilities for improvements to the system. It would be desirable if the coding strategy let the data path share hardware resources for some operation. This coding strategy thus can save area if it is critical for certain application. The parallelism must now be implemented by the user. It would be better if the compiler itself can detect the dependency to reorganize the data path in which the parallelism can be achieved automatically. Our result is high-quality in terms of area, power, speed and trade-off between parameter this is better explained in comparative view

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] Akil Kaveti Dr. William r. Dieter Director of thesis Dr. Yuming zhangDirector of graduate studies “HDL implementation and analysis of a residual register for A floating-point arithmetic unit” March 25, 2008 .

[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 onComputer 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, pp 37-40.

[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” ,International transactions in Mathematical Sciences and ComputerVolume3,No.1,2010,pp 131-140.

[15] Javier and Thomas “LOP for latency improvement in single data path floating point adder”.

[16] Gerald R. Morris and Viktor K. Prasanna “Pipelined Data path for an IEEE-754 64-Bit Floating-Point Jacobi Solver”, Supported by the United States National Science Foundation under award No. CCR-0311823 and in part by award No. ACI-0305763.

[17] GOVINDU G., ZHUOL. , CHOI S. , PRASANNA V. “Analysis of high performance floating-point arithmetic on FPGAS”. Pro c. 18th Int. Symp . Parallel and Distributed Processing, 2004, p. 1494.

[18] P. KA RLS T R OM P. , E H LIAR A. , LIU D. “High performance, low latency FPGA based floating point adder and multiplier units in a virtex 4 ”.IET Comput.Digit.Tech.,2008,Vol.2,No.4,pp.305-313/305.

[19] CATANZARO B., NELSON B. “Higher radix floating-point representations for fpga-based arithmetic”. Proc. 13th Annual IEEE Symp. Field-programmable Custom Computing Machines (FCCM’05), Washington, DC, USA, 2005, IEEE Computer Society, pp. 161 – 170.

[20] SCHWARZ E .M., SCHMOOKL ERM. , TRONG.D. “Hardware implementations of demoralized numbers”. Proc. 16th IEEE Symp. Computer Arithmetic, 2003, pp. 70 – 7 8.

[21] BRUNELLI C ., GARZ IA F. , NURM I J . , M UCCI C. , CAMPI F., RO SSI D. “A FPGA implementation of an open-source floating-point computation system”. Proc. 2005 I nt. Symp. System-on-Chip, 2005, pp. 29 – 32.

[22] SA NTORO M.R. , BEWICK G., HOROWI TZ M. A. “Rounding algorithms for IEEE multipliers”. Proc. 9t h Symp. Computer Arithmetic, 1989, pp. 176 – 183.

[23] NALLATECH ,“ Nallatech floating point cores” Nallate ch, 2002, available at: www.nallatech.com.

[24] DE T R EY J. , DE DI NEC HI N F. , “A parameterized floating - point exponential function for FPGAS”. IEEE In t. Conf. Field Programmable Technology, 2005, pp. 27 – 3 4

[25] XILINX: “Floating -point operator v3.0’ (Xilinx, 2006, 3rd edn.)”, available at: www.xilinx.com.

[26] ANDRAKA R .: ‘Supercharge your DSP with ultra-fa st floating point ffts’, DSP M a gaz in e , 2007, (3), pp. 42 – 4 4.

[27] S. Hwang. “High-Speed Binary and Decimal Arithmetic Logic Unit”, American Telephone and Telegraph Company, AT&T Bell Laboratories, US patent 4866656, pp. 1-11, Sep 1989.

[28] Sreehari Veeramachaneni, M, Kirthi Krishna; V, Prateek G, S. Subroto, S, Bharat, M.B.Srinivas, “A Novel Carry-Look Ahead Approach to a Unified BCD and Binary Adder/Subtractor”, 21st International Conference on VLSI Design 2008, pp. 547-552, Jan 2008.

[29] W. Haller, U. Krauch, and H. Wetter, “Combined Binary/Decimal Adder Unit,” International Business Machines Corporation, US patent 5928319, pp. 1 – 9, Jul 1999.

[30] D.R.Humberto Calderón, G. N. Gaydadjiev, S. Vassiliadis, “Reconfigurable Universal Adder”, Proc. of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP 07), pp. 186-191, July 2007.

[31] H. Fischer andW. Rohsaint. “Circuit Arrangement for Adding or Subtracting Operands Coded in BCD-Code or Binary-Code”, Siemens Aktiengesellschaft, US patent 5146423, pp. 1– 9, Sep 1992.

[32] M.Taher, M.Aboulwafa, A.Abelwahab, E.M.Saad “High –Speed,Area-Efficientfpga-Based Floating-Point Arithmetic Modules”.

[33] IP for floating point adder “http://hdlcores.com/dcdpdf/xil/dfpau-dp_ds.pdf” .

[34] Peter-Michael Seidel and Guy Even, “Delay-Optimized Implementation of IEEE Floating-Point Addition” Transactions on Computer”, Ieee Vol. 53, No. 2, February 2004.

[35] Nikhil Kikkeri and Peter-Michael Seidel, “Optimized Arithmetic Hardware Design based on Hierarchical Formal Verification” ,1-4244-0395-2/06/$20.00 ©2006 IEEE.