Vol 06, Article 01529; February 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

DECIMAL FLOATING POINT IMPLEMENTATION USING VERILOG HDL

VINOD KAPSE[1], PUNEET BHARADWAJ[2], Y. ARUNIKA RAO[3]
[1]Electronics and Communication, GGITS RGPV University, Jabalpur (M.P.), India
[2,3]Electronics and Communication, GGITS, Jabalpur (M.P.), India
[1]kapse.vinod@rediffmail.com, [2]puneet_bharadwaj@yahoo.com, [3]arunika.rao@gmail.com

ABSTRACT

Decimal Floating Point Multiplication is extensively used in many commercial application including financial analysis, banking, tax calculation, currency conversion, insurance and accounting. High performance, less circuitry and better overall characteristics are the main reason why binary Floating point hardware (BFP) is choosen over decimal floating point (DFP) hardware. However the binary floating point cannot precisely represent many common decimal values. Further, although binary arithmetic is well suited for the scientific community, it is quite different from manual calculation norms and does not meet legal requirements. Due to the shortcomings of BFP arithmetic, many application involving fractional decimal data are forced to perform their arithmetic either entirely in software or a combination of software and decimal fixed- point hardware. Providing decimal floating point improve the performance of such application. The proposed designs are compliant with IEEE754 format and handles overflow, underflow, rounding and various exception conditions. Floating Point multiplier are compared in terms of area, latency, throughput based on verified verilog HDL.

Keywords: *Decimal Floating Point Multiplication, Carry Save Adder, Wallace Tree, Latency, Throughput*

1. INTRODUCTION

Decimal arithmetic is necessary in many financial and commercial applications, which process decimal values and perform decimal rounding. Due to the importance of decimal arithmetic in commercial applications and the potential speed up achievable microprocessors supporting decimal floating-point (DFP) arithmetic are now available .A fundamental operation in DFP arithmetic is multiplication, which is integral to the decimal-dominant applications found in financial analysis, banking, tax calculation, currency conversion, insurance, and accounting. This paper presents the designs of an iterative and a parallel DFP multiplier. Our iterative DFP multiplier uses the iterative fixed point decimal multiplier presented in for significant multiplication. It features a reduced set of multiplicand multiples decimal CSAs for the iterative accumulation of partial products and a variation of direct decimal addition to implement decimal 4:2 compression. Our parallel DFP multiplier uses the radix-10 parallel fixed-point decimal multiplier proposed to perform significant multiplication.

2. IEEE FLOATING POINT STANDARD

IEEE / ANSI/ 754/854/ Standard

The IEEE has established a standard for floating -point numbers

Sign	Exponent	Significand

Figure 1-IEEE Floating Point Number

3. FLOATING POINT MULTIPLICATION ALGORITHM

1 Add exponents and subtract bias.
2. Multiply the significands.
3. Normalize the product.
4. Overflow or Underflow? If yes raise exception.
5 Round the significands to the appropriate number of bits.
6 If still not normalize go back to step 3.
7 Set the sign of the result.

3.1 Carry Save Addition- Carry Save Addition is the idea of utilizing addition without carries connected in series as in the Ripple Carry Adder but instead to count and hence avoid the ripple carry chain. In this way multi- operand addition can be carried out without the excessive delay resulting from long carry chains. The following example shows how a 4- bit numbers and generates a 4-bit partial sum and 4-bit carry vector, avoiding the connection of each bit adder's carry out to the carry – in of the next adder.

3.2 Vertical Carry propagate in Carry Save Adder

For adding more than two numbers the simple method first add two m numbers (all are n-bits wide) the resultant sum is add to the next vertically this process goes on. This requires full adder for adding three bit numbers. In case of intermediate partial stage sum carry save adder tree is used. This requires a total of n-1 addition and the total gate delay during this addition is O(nlgm).As studied from previous papers carry save

Vol 06, Article 01529; February 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

adder reduces this delay. In carry save adder three bit number x, y, z are added they generate sum (s) and carry out (c) as output. They do this is O (1) time because sum and carry is propagated vertically.

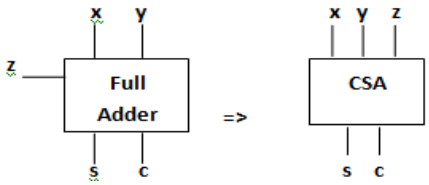


Figure 2: The carry save adder block is the same circuit as the full adder

A carry save adder is same as the full adder. It has also three input x, y, z and two output s and c but the difference c_{in} in full adder is renamed as z and the output c_{out} as c and sum is renamed as s. In carry save adder three input are added and carry generated is save in c and this carry is added to the final sum but in full adder carry is added directly to the next column. Carry save adder is faster than full adder. If its output sum and carry of one full adder goes to the input of another full adder this procedure is repeated and this form a carry save adder tree. The CSA block generate c1 not c0.CSA block are independent and the time required for all circuitry O (1) time. For getting final time the total delay is O(lg m) delay.

3.3 Applications of Carry save Adder Tree - Wallace Tree

Wallace Tree is known for their optimal computation time, when adding multiple operands to two outputs using carry save adders. They aim at offering high speed and low power consumption even when occupying small silicon area. The structure is optimized in such a way that the latency of the total circuit reduces considerably. The latency in the Wallace tree multiplier tree can be reduced by decreasing the number of adders in the partial products reduction stage . Multi –bit compressors are used for realizing the reduction in the number of partial product addition stages

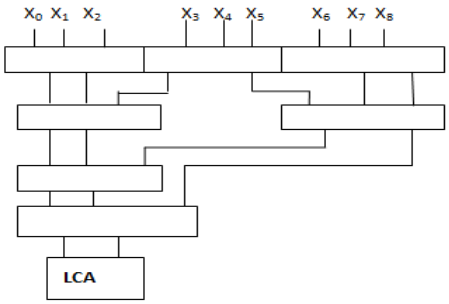


Figure 3 - A Wallace tree for adding m n-bit numbers

4. METHODOLOGY

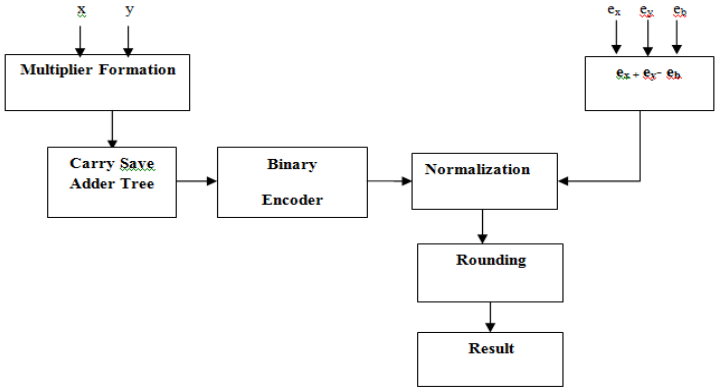


Figure 4- Design of Fast FMA Unit

- A multiplier consist of multi stage Partial Product Generation, Partial product addition and final addition. A multiplier consists of two operands, a multiplicand Y and multiplier X produces a product. In this method in place computation is used to multiply two numbers. In the simultaneous two row output are added using high speed adder such as carry save adder is used to compute the result.

4.1 Binary Encoder

- For leading 0s /1s prediction binary encoder is used.
- An encoder is a combinational circuit that performs the inverse operation of a decoder. The device output code has fewer bits than the input code has; the device is usually called an encoder. e.g. 2^n-to-n priority encoders.

Vol 06, Article 01529; February 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

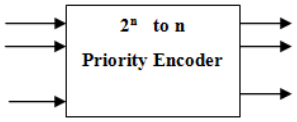


Figure 5- Binary Encoder

The simplest encoder is a 2^n-to-n binary encoder, where it has only one of 2^n inputs = 1 and the output is the n-bit binary number corresponding to the active input.

- **Example**
- **45.25 (base 10)= 101101.01(base 2)**
- **$1.0110101 * 2^5$**
- **E= 127+5**

0	**10000100**	**01101010000000000000**

Figure 6–IEEE Floating Point Format of operand A

- **35.75 (base 10)= 100011.11(base 2)**
- **1.00011×2^5**
- **E = 5+127**

1	**10000100**	**00011000000000000000000**

Figure 7– IEEE Floating Point Format of Operand B

4.2 Binary Multiplication

- **45.25×35.75**

45.25(base 10) = 10110101 (base 2)
35.75 (base 10)= 100011 (base 2)

```
 10110101
   ×100011
 _________
 10110101
 10110101×
 00000000 ××
 00000000 ×××
 00000000 ××××
 10110101 ××××
_____________
1100010011111
```

- **2^5=32 =100000**
- **Add exponent 100000+100000=1000000**
- **Subtract bias =127=0111111**
- **1000000**
- **- 0111111**

_____

- **0000001**

1	**0000001**	**11.000101111111**

Figure 8- IEEE Floating Format of Binary Multiplication

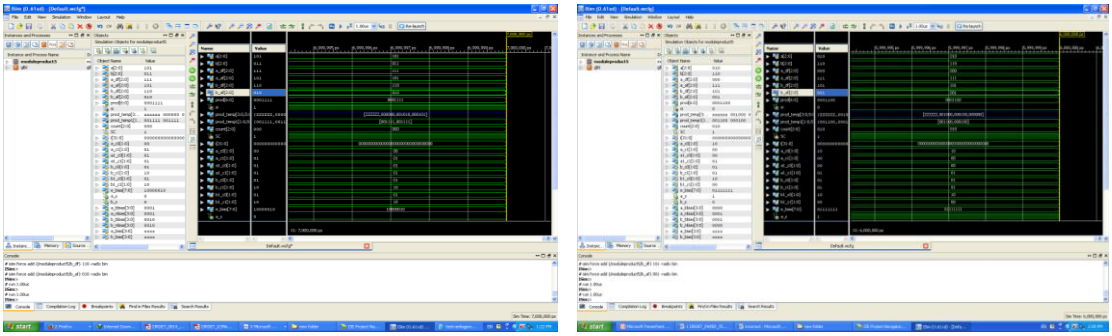
4.3 Horizontal Carry Propagate in Carry Save Adder

As in vertical carry propagate method if we add three number sum carry goes vertically to the next. In vertical carry propagate total delay is O (1) time. In horizontal method 3 bit in a row are added, and carry goes i into the next column or sum goes to the next full adder. This method reduces delay .For adding 5 bit number it requires 6 step means if no of bit is p it requires p+1 step whereas in horizontal carry propagate method for adding p-bit number it only requires p step. Hence horizontal carry propagate horizontal method is faster than vertical method as used in previous papers. We add two number in two rows and carry propagate in rows horizontally this reduces the delay by one as compared to vertical carry propagate method used in carry save adder and Wallace tree.

Vol 06, Article 01529; February 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

5. RESULT

Modulo Product



Conclusion and Future Scope

Conclusion

The goal of this research was to investigate hardware to reduce latency and increase throughput. By using horizontal carry propagate in carry save adder achieve this goal. DFP implementation using VERILOG may be beneficial for chip design. Horizontal Carry propagate in carry save adder is an attractive implementation. It reduces delay and latency.

Future Aspect

The future aspect of this thesis work will be implemented the design for purpose of banking, taxation, financial analysis.ALU can be designed for DSP application. Double precision floating point multiplier can also be implemented. This thesis work has been done using Verilog language. The proposed work is implemented on FPGA.

ACKNOWLEDGEMENTS

I Y. Arunika Rao (Mtech in Embedded System and VLSI Design) express my heartfelt thanks to my guide Mr. Dr. Vinod Kapse Mtech (Phd) HOD ECE DEPT, GGITS, Puneet Bharadwaj Jabalpur (M.P.) for their able guidance and useful suggestions which help me in the project work

REFERENCES

[1] Brian Hickmann, Andrew Krioukov, and Michael Schulte **"A Parallel IEEE P754 Decimal Floating-Point Multiplier"** Computer Design, 2007, ICCD 2007. 25th International Conference on DOI: 10.1109/ICCD.2007.4601916 Publication Year: 2007.

[2] Ramy Raafat, Amira M. Abdel-Majeed, Rodina Samy, Tarek ElDeeb, Yasmin Farouk, Mostafa Elkhouly, and Hossam A. H. Fahmy **" A Decimal Fully Parallel and Pipelined Floating Point Multiplier "** Signals, Systems and Computers, 2008 42nd Asilomar Conference on DOI: 10.1109/ACSSC.2008.5074737 Publication Year: 2008.

[3] Mark A. Erle, Michael J. Schulte "**Decimal multiplication via carry-save addition"** Application-Specific Systems, Architectures, and Processors, 2003. Proceedings. IEEE International Conference on DOI: 10.1109/ASAP.2003.1212858 Publication Year: 2003.

[4] Ramesh, A.P. ; Tilak, A.V.N. ; Prasad, A.M. **"An FPGA based high speed IEEE-754 double precision floating point multiplier using Verilog "**Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT), 2013 International Conference on DOI: 10.1109/ICEVENT.2013.6496575 Publication Year: 2013.

[5] V Narsimha Rao, V. Swathi **"Normalization of Floating Point Multiplication using Verilog HDL"** International Journal of VLSI and Embedded Systems-IJVES Vol 04, Article 07139; August 2013.

[6] Ms Anjana Sasidharan, Mr. M.K. Arun **"VHDL implementation of IEEE 754 Floating Point Unit"**