BEST: International Journal of Management, Information Technology and Engineering (BEST: IJMITE)
Vol. 1, Issue 3, Dec 2013, 199-204
© BEST Journals

IMPLEMENTATION OF FLOATING POINT MULTIPLIER USING VHDL

BHAGYASHREE HARDIYA, KRATI SHIV & MADHURI RAO

B. E. Students, Department of ECE, Institute of Engineering & Science, Indore Professional Studies, Indore, Madhya Pradesh, India

ABSTRACT

In this paper, multiplication of the floating point numbers described in IEEE 754 single precision valid. Floating point multiplier is done using VHDL .Implementation in VHDL(VHSIC Hardware Description Language) is used because it allow direct implementation on the hardware while in other language we have to convert them into HDL then only can be implemented on the hardware. In floating point multiplication, adding of the two numbers is done with the help of various types of adders but for multiplication some extra shifting is needed.This floating point multiplication handles various conditions like overflow, underflow, normalization, rounding. In this paper we use IEEE rounding method for perform the rounding of the resulted number.This paper reviews the implementation of an IEEE 754 single precision floating point multiplier developed by many researchers.

KEYWORDS: Floating Point Unit, XILINX ISE 8.1E as Synthesizer, MODEL-SIM as Simulator, Floating Point Arithmetic, Booth Multiplier, IEEE Rounding Method, Serial by Parallel Adder

INTRODUCTION

Floating point is the binary representation of the real numbers. There are many ways to represent the number system however non-integer representation has gained widespread use i.e. floating point. The typical floating point number can be represented exactly is of the form:

$$Significant\ digits \times base^{exponent}$$

FIXED POINT REPRESENTATION

A fixed point number representation is a real data type for a number that has a fixed number of digits after the decimal point ('.'). A value of a fixed-point data type is essentially an integer that is scaled by a specific factor determined by the type. The scaling factor is usually a power of 10 (for human convenience) or a power of 2 (for computational efficiency). However, other scaling factors may be used occasionally.

FLOATING POINT REPRESENTATION

The term floating point refers to the fact that a number's decimal point can "float"; that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated as the exponent component in the internal representation, and floating point can thus be thought of as a computer realization of scientific notation. As an example, an exponent of (−3) and significant of 1.5 might represent the number 1.5 × 2^3= 0.1875.

FLOATING POINT FORMAT

Sign Bit	Exponent	Mantissa
1 bit	8 bit	23 bits

Floating point number word length[2] may be of two types :

- Single precision floating point number consists of 32 bits
- Double precision floating point number consists of 64 bits.

The format for single precision floating point number is shown in figure above.

In this project we make use of only single precision floating point multiplier because of less complexity. The exponent is a signed number represented using the bias method with a bias of 127. The term biased exponent refers to the unsigned number contained in bits 1 through 8 and unbiased exponent means the actual power to which 2 is to be raised. The fraction represents a number less than 1, but the significand of the floating-point number is 1 plus the fraction part. In other words, if e is the biased exponent and f is the value of the fraction field, the number being represented as: 1.f *2e–127.

Serial by Parallel Booth Multiplier

The common multiplication method is “add and shift” algorithm. To reduce the number of partial products to be added, Modified Booth algorithm [3][4] is one of the most popular algorithms. The simple serial by parallel booth multiplier is particularly well suited for bit serial processors implemented in FPGAs without carry chains because all of its routing is to nearest neighbours with the exception of the input. The serial input must be sign extended to a length equal to the sum of the lengths of the serial input and parallel input to avoid overflow, which means this multiplier takes more clocks to complete than the scaling accumulator version. This is the structure used in the venerable TTL serial by parallel multiplier.

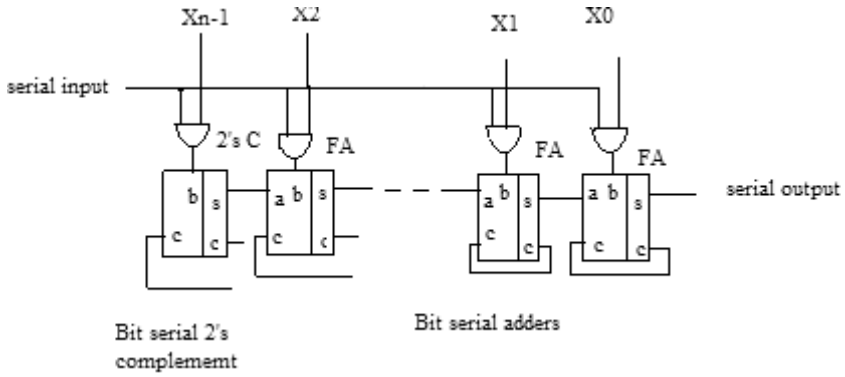

Figure 1: Serial by Parallel Booth Multiplier

The general architecture of the serial/parallel multiplier is shown in the figure above. One operand is fed to the circuit in parallel while the other is serial. N partial products are formed each cycle. On successive cycles, each cycle does the addition of one column of the multiplication table of M*N PPs. The final results are stored in the output register after N+M cycles. While the area required is N-1 for M=N.

Let us take an example:

Using Booth algorithm multiply A and B.

A= 20 & B=30

A= 0010100 Please note that both numbers are extended to cover 2A or 2B and the

B= 0011110 sign bit (whichever is larger).

A * B = A= 0 0 1 0 1 0 0

-0

B= 0 0 1 1 1 1 0 0

+2 -2

2A = 40 = 00101000

-2A = 11011000

Now performing the addition we have

1 1 1 1 1 1 1 0 1 1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 0

0 0 0 1 0 0 1 0 1 1 0 0 0

512 + 64 + 16 + 8 = (600)10

IEEE Rounding Method

Four rounding modes are dictated by the IEEE 754 standard:

- The round to nearest mode,
- The round to zero mode (RZ)
- The round to positive infinity (RP) mode,
- The round-to-minus infinity (RM) mode

The RP mode can be implemented as the RI mode for positive numbers and the RZ mode for negative numbers. Similarly, the RM mode can be implemented as the RZ mode for positive numbers and as the RI mode for negative numbers.

To perform IEEE rounding [6] using a conventional algorithm, we have to perform the following:

Step 1) C and S Computation: Compute C and S to a precision of 2N bits by generating the partial products and reducing them with a partial product reduction network.

Step 2) R Term Computation: Compute R by adding up C and S using a CPA.

Step 3) Pre-Round Normalization: Normalize R if R.m=1by right shifting R by 1-bit and adjusting the exponent appropriately.

Step 4) Rounding Bits Computation: Compute the g and s bits.

Step 5) Rounding: Denote the higher order N bits of R as Rh. Based on the rounding mode and rounding bits, add a rounding one unit-in-the-last-place (ulp) to Rh when necessary.

Step 6) Post-Round Normalization: If Rh.c=1 as a result of rounding, another 1-bit normalization right shift of Rh is needed, again with the exponent adjusted accordingly. The term Rh is the FCR SV.

Notation

The C and S are terms have 2N bits. The lower order N-1 bits are among the bits that the hardware needs to examine for rounding. The higher order N+1 bits are used to produce the FCR N bit significand; the extra bit is required to account for the 1-bit normalization right shift. For notational simplicity, we define the binary point to be at N-1th bit of C and S. As shown in Figure 1, the higher order-bit N+1 integer portions of S and C are denoted as Sf and Cf, respectively, and the lower order-bit fractions as and, respectively.

Note that Rh in this case is different from Ri and denotes the higher order N bits of R after the normalization step. Several implementation difficulties associated with the IEEE rounding specification can now be identified. First, and need to be computed to a precision of bits, requiring additional hard-ware. Second, correct rounding in the RN, RP, and RM modes depends on the bit, incurring additional delay because must first be computed and then rounded with an extra addition step. Finally, two normalization steps are potentially needed as rounding may cause the normalized significant of the result to again overflow, requiring a post-round normalization right shift.

LITERATURE REVIEW

Various researchers had contributed in the field of floating point multiplier using VHDL. Some of them are as follows:

- **Paper Title:** An Efficient Implementation of Floating Point Multiplier [1] [2]

  Publication: *International Journal of Engineering Research & Technology (IJERT)*

  Author Name: Mohamed Al-Ashrafy, Ashraf Salem, Wagdy Anis.

This paper presents a implementation of a floating point multiplier that supports the IEEE 754-2008 binary interchange format; the multiplier doesn't implement rounding and just presents the significant multiplication result as is (48 bits); this gives better precision if the whole 48 bits are utilized in another unit; i.e. a floating point adder to form a MAC unit.

The design has three pipelining stages and after implementation on a Xilinx Virtex5 FPGA it achieves 301 MFLOPs.

- **Paper Title:** Pipeline Floating Point ALU Design using VHDL[8][9]

  Publication: *ICSE2002 Proc. 2002, Penang, Malaysia*

  Author Name: Mamu Bin Ibne Reaz, Md. Shabiul Islam, Mohd. S. Sulaiman.

In this project, pipelined floating point multiplication is divided into three stages . Stage **1 checks** whether the operand is 0 and report the result accordingly. Stage 2 determines the product sign, add exponents and multiply fractions. Stage3 normalize and concatenate the product.

- **Paper Title:** Optimizing Floating Point Units in Hybrid FPGAs[10]

  Publication: IEEE transactions on very large scale integration (VLSI) systems, vol. 20, no. 7, July 2012.

  Author Name: ChiWai Yu, Alastair M. Smith, Wayne Luk, Fellow, IEEE, Philip H. W. Leong,

  Senior Member, IEEE, and Steven J. E. Wilton, Senior Member, IEEE

  This paper proposes a novel methodology to determine optimized coarse-grained FPUs in hybrid FPGAs, based

on common subgraph extraction. The effect of merging different coarse-grained types into larger FPUs is also studied. We observe that:

- The speed of the system is the highest for implementations involving only floating point adders and floating point multiplier.
- Higher density subgraphs produce greater reductionon area.
- They provide the best area-delay product.
- Merging of FPUs can improve the speed of hybrid FPGAs, but results in consuming more area.

CONCLUSIONS

In this paper, we have seen that the multipliers play an important role in today's digital signal processing and various other applications [11]. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. By using serial by parallel Booth multiplier we see that in parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm is one of the most popular algorithms. Rounding of the resulted number provide a precision multiplication of the numbers by using IEEE rounding method .

REFERENCES

1. IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.
2. ANSI/IEEE Std 754-1985, IEEE Standard for Binary Flooring-Point Arithmetic, IEEE, New York, 1985.
3. L. Song, K.K. Parhi, "Efficient Finite Field Serial/Parallel Multiplication", Proc. of International Conf. on Application Specific Systems, Architectures and Processors, pp. 72-82, Chicago, USA, 1996.
4. P. E. Madrid, B. Millar, and E. E. Swartzlander, "Modified Booth algorithm for high radix fixed- point multiplication," IEEE Trans. VLSI Syst., vol. 1, no. 2, pp. 164-167, June 1993.
5. A. Booth, "A signed binary multiplication technique," Q. J. Me& Appl. March., vol. 4, pp. 236-240, 19.51.
6. Nhon T. Quach, Member, IEEE, Naofumi Takagi, Senior Member, IEEE, and Michael J. Flynn, Fellow, IEEE" Systematic IEEE Rounding Method for High-Speed Floating-Point Multipliers" IEEE transactions on very large scale integration (vlsi) systems, vol. 12, no. 5, may 2004.
7. Report on Efficient Floating Point 32-bit single Precision Multipliers Design using VHDL by Dr. Raj Singh, Group Leader, VLSI Group, CEERI, Pilani.
8. Xianyang Jianga, Peng Xiaoa, Meikang Qiub, Gaofeng Wanga" Performance effects of pipeline architecture on an FPGA-based binary32 bit floating point multiplier "Microprocessors and Microsystems xxx (2013) xxx–xxx.
9. Mamu Bin Ibne Reaz, MEEE, Md. Shabiul Islam, MEEE, Mohd. S. Sulaiman, MEEE Faculty of Engineering, Multimedia University, 63 100 Cybejaya, Selangor, Malaysia "Pipeline Floating Point ALU Design using VHDL "ICSE2002 Proc. 2002, Penang, Malaysia.

10. C. W. Yu, A. M. Smith, W. Luk, P. H. W. Leong, and S. J. E. Wilton, “Optimizing coarse-grained units in floating point hybrid FPGA,” inProc. ICFPT, 2008, pp. 57–64.
11. John G. Proakis and Dimitris G. Manolakis (1996), “Digital Signal Processing :Principles, Algorithms and Applications”, Third Edition.