DESIGN OF DOUBLE PRECISION FLOATING POINT MULTIPLICATION ALGORITHM WITH VECTOR SUPPORT

T.Govinda Rao, P.Devi Pradeep, P.Kalyanchakravarthi

Assistant Professor, Department of ECE, GMRIT, RAJAM, AP, INDIA

ABSTRACT

This paper presents floating point multiplier capable of supporting wide range of application domains like scientific computing and multimedia applications and also describes an implementation of a floating point multiplier that supports the IEEE 754-2008 binary interchange format with methodology for estimating the power and speed has been developed. This Pipelined vectorized floating point multiplier supporting FP16, FP32, FP64 input data and reduces the area, power, latency and increases throughput. Precision can be implemented by taking the 128 bit input operands.The floating point units consumeless power and small part of total area. Graphic Processor Units (GPUS) are specially tuned for performing a set of operations on large sets of data. This paper also presents the design of a Double precision floating point multiplication algorithm with vector support. The single precision floating point multiplier is having a path delay of 72ns and also having the operating frequency of 13.58MHz.Finally this implementation is done in Verilog HDL using Xilinx ISE-14.2.

KEYWORDS

floating point multiplier, GPUS, operating frequency, HDL

1. INTRODUCTION

Floating Point numbers represented in IEEE 754 format is used in most of the DSP Processors. Floating point arithmetic is useful in applications where a large dynamic range is required or in rapid prototyping applications where the required number range has not been thoroughly investigated. A Floating point multiplier is the most common element in most digital applications such as digital filters, digital signal processors, data processors and control units.

There are two types of number formats present,

1. Fixed point representation
2. Floating point representation.

These refer to the format used to store and manipulate numbers within the devices. Fixed point DSPs usually represent each number with a minimum of 16 bits. In comparison, floating point DSPs use a minimum of 32 bits to store each value. This results in many more bit patterns than for fixed point. All floating point DSPs can also handle fixed point numbers, a necessary to implement counters, loops, and signals coming from the ADC and going to the DAC.In general purpose fixed point arithmetic is much faster than floating point arithmetic. However, with DSPs the speed is about the same, a result of the hardware being highly optimized for math operations.

The internal hardware of floating point DSP is much complicated than for a fixed device. Floating point has better precision and a higher dynamic range than fixed point. In addition, floating point programs often have a shorter development cycle, since the programmer doesn't generally need to

DOI:10.5121/Jmicro.2016.1201

worry about issues such as overflow, underflow and round-off error.Noise in signals is usually represented by its standard deviation. For here, the important fact is that the standard deviation of this quantization noise is about one-third of the gap size. This means that the signal-to-noise ratio for storing a floating point number is about 30 million to one, while for a fixed point number it is only about ten-thousand to one. In other words, floating point has roughly 30,000 times less quantization noise than fixed point.The important idea is that the fixed point programmer must understand dozens of ways to carry out the very basic task of multiplication. In contrast, the floating point programmer can spend is time concentrating on the algorithm the cost of the DSP is insignificant, but the performance is critical. In spite of the larger number of fixed point DSPs being used, the floating point market is the fastest growing segment. Verilog programming has been used to implement Floating Point Multiplier. Tool used for programming →XILINX ISE SUITE 14.2 Version.

2. IEEE754 FLOATING POINT REPRESENTATION

2.1 BASIC REPRESENTATION

IEEE floating point numbers have three basic components: the sign, the exponent, and the mantissa. The mantissa is composed of the fraction and an implicit leading digit. The exponent base 2 is implicit and need not be stored.

2.1.1 SINGLE PRECISION

The IEEE 754 single precision binary format representation

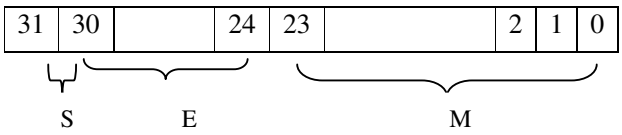


Figure 1: IEEE-754 Single Precision Bit Format

$$Z = (-1)^S \times 2^{(E-Bias)} \times (1 \cdot M) \qquad (2.1)$$

Where S = Sign Bit
E = Exponent
M = Mantissa

- The sign bit is as simple as it gets. 0 denotes a positive number; 1 denotes a negative number. Flipping the value of this bit flips the sign of the number.
- The exponent field needs to represent both positive and negative exponents. To do this, a bias is added to the actual exponent in order to get the stored exponent.
- For IEEE single-precision floats, this value is 127. Thus, an exponent of zero means that 127 is stored in the exponent field. A stored value of 200 indicates an exponent of (200-127), or 73. For reasons discussed later, exponents of -127 (all 0s) and +128 (all 1s) are reserved for special numbers.
- Significand is the mantissa with an extra MSB bit i.e.,1 which represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

$$M = m_{22}2^{-1} + m_{21}2^{-2} + \cdots\cdots\cdots + m_1 2^{-22} + m_0 2^{-23}$$

2.1.2 DOUBLE PRECISION

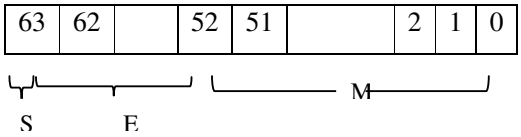


Figure 2 IEEE 754 Double Precision Bit Format

The sign bit is as simple as it gets. 0 denotes a positive number; 1 denotes a negative number. Flipping the value of this bit flips the sign of the number.

- The exponent field needs to represent both positive and negative exponents. To do this, a bias is added to the actual exponent in order to get the stored exponent.
- For IEEE single-precision floats, this value is 1023. Thus, an exponent of zero means that 1023 is stored in the exponent field. A stored value of 200 indicates an exponent of (1023-200), or 823. For reasons discussed later, exponents of -1023 (all 0s) and +1023 (all 1s) are reserved for special numbers.
- Significand is the mantissa with an extra MSB bit i.e., 1 which represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

$$M = m_{51}2^{-1} + m_{50}2^{-2} + m_{49}2^{-3} + \cdots\cdots\cdots + m_1 2^{-51} + m_0 2^{-52}$$

Parameter	Format			
	Single	**Single Extended**	**Double**	**Double Extended**
P	24	>=32	53	>=64
E_{max}	+127	>=+1023	+1023	>=+16383
E_{min}	-126	<=-1022	-1022	<=-16382
Exponent bias	+127	Unspecified	+1023	Unspecified
Exponent width in bits	8	>=11	11	>=15
Format width in bits	32	>=43	64	>=79

Table 1: Summary of Format Parameters

2.2 ROUNDING MODES

Rounding takes a number regarded as infinitely precise and, if necessary, modifies it to fit in the destination's format while signaling the inexact exception. The rounding modes affect all arithmetic operations except comparison and remainder. The IEEE-754 floating-point standard defines three rounding modes.

2.2.1 ROUND TO NEAREST

An implementation of this standard shall provide round to nearest as the default rounding mode. In this mode the representable value nearest to the infinitely precise result shall be delivered; if the two nearest representable values are equally near, the one with its least significant bit zero shall be delivered. However, an infinitely precise result with magnitude at least $2E\max(2 - 2{-}p)$ shall round to INFINITY with no change in sign.

2.2.2 DIRECTED ROUNDING

An implementation shall also provide three user-selectable directed rounding modes: round toward +INFINITY, round toward –INFINITY, and round toward 0. When rounding toward +INFINITY the result shall be the format's value (possibly +INFINITY) closest to and no less than the infinitely precise result. When rounding toward –INFINITY the result shall be the format's value (possibly –INFINITY) closest to and no greater than the infinitelyprecise result. When rounding toward 0 the result shall be the format's value closest to and no greater in magnitude than the infinitely precise result.

2.2.3 ROUNDING PRECISION

Normally, a result is rounded to the precision of its destination. However, some systems deliver results only to double or extended destinations. On such a system the user, which may be a high-level language compiler, shall be able to specify that a result be rounded instead to single precision, though it may be stored in the double or extended format with its wider exponent range. Similarly, a system that delivers results only to double extended destinations shall permit the user to specify rounding to single or double precision.

2.3 SPECIAL VALUES

The IEEE-754 floating-point standard specifies four kinds of special values

2.3.1 SIGNED ZERO

Since the floating-point number is a sign-magnitude representation, both positive and negative zeros exist. The two values are numerically equal, whereas some operations produce different results depending upon the sign bit.

$$\text{e.g., } 1 / (+0) = \infty \quad \text{and} \quad 1 / (-0) = -\infty$$

2.3.2 SUB NORMAL NUMBERS

A subnormal number represents a value of the magnitude which is smaller than the minimum normalized number by denormalizing the significand, which means the MSB of the significand is “0”. It improves the precision of the numbers that are close to zero so that the values can be represented when underflow occurs.

2.3.3 INFINITY ARITHMETIC

Infinity arithmetic shall be construed as the limiting case of real arithmetic with operands of arbitrarily large magnitude, when such a limit exists. Infinities shall be interpreted in the affine sense, that is, –INFINITY < (every finite number) < +INFINITY. The exceptions that do pertain to INFINITY are signaled only when

1. INFINITY is created from finite operands by overflow or division by zero, with corresponding trap disabled.
2. INFINITY is an invalid operand.

2.3.4 NANS (NOT-A-NUMBERS)

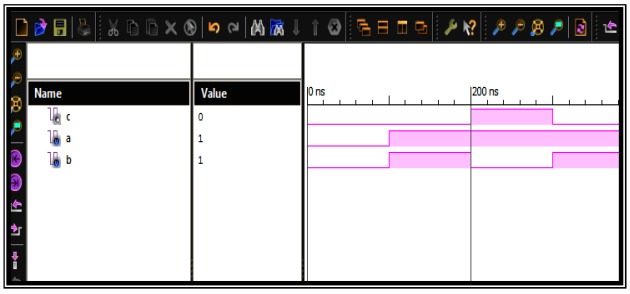
Two different kinds of NaN, signaling and quiet, shall be supported in all operations. SignalingNaNs afford values for uninitialized variables and arithmetic-like enhancements that are not the subject of the standard. Quiet NaNs should, by means left to the implementer's discretion, afford retrospective diagnostic information inherited from invalid or unavailable data and results. Propagation of the diagnostic information requires that information contained in the NaNs be preserved through arithmetic operations and floating-point format conversions.

2.4 EXCEPTIONS

2.4.1 INVALID OPERATION

The invalid operation exception is signaled if an operand is invalid for the operation to be performed. The result, when the exception occurs without a trap, shall be a quiet NaN provided the destination has a floating-point format. The invalid operations are

1. Any operation on a signaling NaN.
2. Addition or subtraction – magnitude subtraction of infinities such as (+INFINITY) + (–INFINITY)
3. Multiplication – 0 × INFINITY
4. Division – 0/0 or INFINITY/INFINITY
5. Remainder – *x* REM *y*, where *y* is zero or *x* is infinite.
6. Square root if the operand is less than zero



2.4.2 DIVISION BY ZERO

If the divisor is zero and the dividend is a finite nonzero number, then the division by zero exception shall be signaled. The result, when no trap occurs, shall be a correctly signed INFINITY.

2.4.3 OVERFLOW

The overflow exception shall be signaled whenever the destination format's largest finite number is exceeded in magnitude by what would have been the rounded floating-point result were the exponent range unbounded. The result, when no trap occurs, shall be determined by the rounding mode and the sign of the intermediate result as follows:

1. Round to nearest carries all overflows to INFINITY with the sign of the intermediate result
2. Round toward 0 carries all overflows to the format's largest finite number with the sign of the intermediate result
3. Round toward –INFINITY carries positive overflows to the format's largest finite number, and carries negative overflows to –INFINITY
4. Round toward +INFINITY carries negative overflows to the format's most negative finite number, and carries positive overflows to +INFINITY.

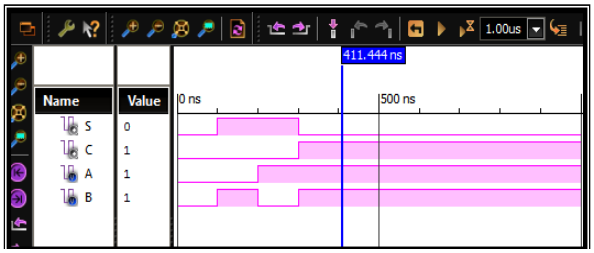
2.4.4 UNDERFLOW

Two correlated events contribute to underflow. One is the creation of a tiny nonzero result between ±2Emin which, because it is so tiny, may cause some other exception later such as overflow upon division. The other is extraordinary loss of accuracy during the approximation of such tiny numbers by denormalized numbers. The implementer may choose how these events are detected, but shall detect these events in the same way for all operations. Tininess may be detected either

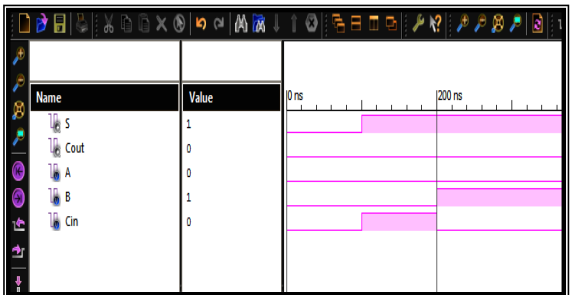
1. *After rounding* - when a nonzero result computed as though the exponent range were unbounded would lie strictly between ± 2Emin
2. *Before rounding* - when a nonzero result computed as though both the exponent range and the precision were unbounded would lie strictly between ± 2Emin.

3. SIMULATION RESULTS

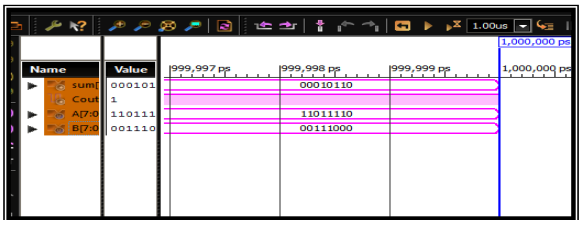
3.1 EXOR OUTPUT



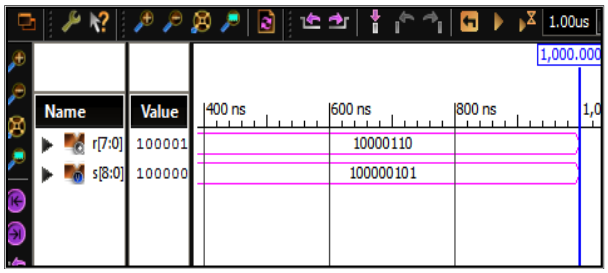
3.2 Half Adderoutput



3.3 Full Adder Output

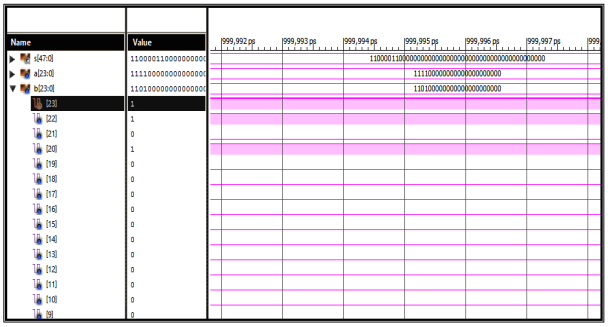


3.4 Ripple Carry Adder Output

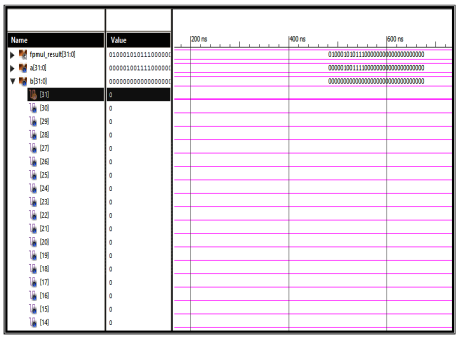


3.5 Ripple Borrow Subtractor Output

3.6 Normalizer Output



3.7 Unsigned Multiplier Output



3.8 Floating point Multiplier Output (60×-6.5)

DESIGN SUMMAR

fp_mul Project Status (03/15/2013 - 18:33:34)			
Project File:	FP_multiplier.xise	**Parser Errors:**	X 3 Errors
Module Name:	fp_mul	**Implementation State:**	Synthesized
Target Device:	xc3s100e-4tq144	**• Errors:**	No Errors
Product Version:	ISE 14.2	**• Warnings:**	6 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	649	960	67%
Number of 4 input LUTs	1128	1920	58%
Number of bonded IOBs	96	108	88%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Tue Apr 1 14:15:01 2014	0	6 Warnings (0 new)	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

3.9 SYNTHESIS result for single precision floating point multiplier

vec_fp_mult Project Status			
Project File:	vec_fp_mult.xise	**Parser Errors:**	No Errors
Module Name:	vec_fp_mult	**Implementation State:**	Synthesized
Target Device:	xa3s100e-4vqg100	**• Errors:**	No Errors
Product Version:	ISE 14.2	**• Warnings:**	26 Warnings (26 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	3107	960	323%
Number of Slice Flip Flops	593	1920	30%
Number of 4 input LUTs	5952	1920	310%
Number of bonded IOBs	296	66	448%
Number of MULT18X18SIOs	4	4	100%
Number of GCLKs	1	24	4%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Fri Apr 4 09:40:54 2014	0	26 Warnings (26 new)	0
Translation Report					
Map Report					

3.10 synthesis result for double precision floating point multiplier

4. CONCLUSION AND FUTURE SCOPE

This paper presented an implementation of a floating point multiplier that supports the IEEE 754-2008 binary interchange format. A methodology for estimating the power and speed has been developed. This Pipelined vectorized floating point multiplier supporting FP16, FP32, and FP64 input data and reduces the area, power, latency and increases throughput. Precision can be implemented by taking the 128 bit input operands. Register Transfer Logic has developed for Double precision Floating Point Multiplier further simulation results can be implemented. The performance of the Floating point multiplier can be increased by taking the 256 bit input bus instead of the 128 bit bus. The throughput and area optimization can be improved by using more general significant multipliers and exponent adders. Two 53 bit multipliers and two 24 bit multipliers are used to compute the significants of all supported Floating point formats

REFERENCES

[1] T.GovindaRao, D.Arun Kumar ”Design of Single Precision Floating Point Multiplication Algorithm with Vector Support”2015 IJSRSET-Volume1-Issue 1, themed Section: Engineering and Technology, Print ISSN: 2395-1990.

[2] ABaluni,FarhadMerchant, sk.nandy, s.Balakrishnan,”a Fully Pipelined Modular Multiple Precision Floating Point Multiplier With Vector Support” 2011 international symposium on Electronic system design(ISED)PP.45-50.

[3] K.Manopolous,D.Reises,V.A.Chouiaras“An Efficient Multiple Precision Floating Point Multiplier”2011, IEEE.PP 153-156.

[4] E.Stenersen,”Vectorized256-bit input fp16/fp32/fp64 floating point multiplier ”Norwegain University of Science and Technology,2007.

[5] I. Koren, Natick “Computer Arithmetic Algorithms” USA: A. K.Peters Ltd., 2001.

[6] S. T. Oskuli, Design of Low-Power Reduction-Trees in Parallel multipliers PhD thesis, Norwegian University of Science and Technology,2008.

[7] G. Even and P.-M. Seidel, “A comparison of three rounding algorithms for IEEEfloating-point multiplication,” IEEE Trans. Comput., vol. 49, no. 7, pp. 638–650,2006.

[8] N. T. Quach , N. Takagi, and M. J. Flynn, “Systematic IEEE rounding method for high Speed floating-point multipliers,” IEEE Trans. Very Large Scale Integr. Syst., vol. 12, no. 5, pp. 511–521, 2004.

[9] L. Wanhammar,” DSP Integrated Circuits”. Academic Press, 1999.

AUTHOR BIOGRAPHY

T.GovindaRao pursing PhD in Pondicherry Central University. He is an Assistant Professor in the Department of Electronics and Communication engineering, GMRIT, Rajam. His current research interests include the areas of VLSISD, Image processing, very large scale integration (VLSI) testing and fault-tolerant computing, video coding techniques, and Architectures design.

P. Devi Pradeep received M.Tech degree from National Institute of Technology, Warangal, Andhra Pradesh, India. He is joined as Assistant Professor in the Department. Of Electronics and Communication Engineering at GMR Institute of Technology, Rajam, Srikakulam District, Andhra Pradesh, India in 2009. Also he is currently as research scholar in KL Universtity.. Having Total teaching experience is 9 years. His research Interests are Video processing, Analog and digital VLSI Design. He is a life member of ISTE Since 2009.

P.Kalyanchakravarthy received M.Tech from NIT ,working as an assistant Professor in the Department of Electronics and Communication engineering, GMRIT, Rajam. His current research interests include the areas of signal and speech processing, very large scale integration (VLSI), image processing.