ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 2, Issue 11, November 2013

Asynchronous Single Precision Floating Point Multiplier Using Verilog HDL

Priyanka Koneru, M.Tech Student ECE, Sri Vasavi Engineering College. Tadepalligudem, INDIA
Tinnanti Sreenivasu, Assistant Professor ECE, Sri Vasavi Engineering College, Tadepalligudem, INDIA
Addanki Purna Ramesh, Associate Professor ECE, Sri Vasavi Engineering College, Tadepalligudem, INDIA

***Abstract*— A fast and energy efficient floating point unit is always needed in major applications like digital signal processing, image processing, and real time data processing and multimedia applications.**

As circuits get shrink, the synchronous design becomes a critical challenge in terms of clock skew and clock distribution. One attractive alternative is to use robust asynchronous circuits, which gracefully accommodate these timing discrepancies. In this paper, a single precision asynchronous floating point multiplier is implemented using VERILOG hardware description language.

***Index Terms*—: Asynchronous, clock skew, clock distribution, floating point multiplier, Single Precision, verilog.**

I. INTRODUCTION

Floating point arithmetic unit 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 [6]. In general, most of the applications represent these floating point numbers in IEEE 754 format. The 32-bit single precision floating point representation of IEEE 754 format consists of three fields.

- The most significant bit is the *sign bit* (S), with 0 for negative numbers and 1 for positive numbers.
- The following 8 bits represent *exponent* (E).
- The remaining 23 bits represents *fraction* (F).

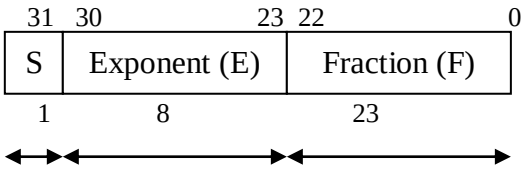

Figure 1: 32-bit Single Precision Floating-point Number

When two n-bit numbers are multiplied, a 2*n*-bit product is produced. The steps involved in that multiplication process are large and the addition of n partial products also becomes a complicated task.

For multiplication operation, to speed up the multiplication, use two alternatives may include

1. Reduce the number of partial products. Clearly, a smaller number of partial products reduces the complexity and as a result, reduces the partial products accumulation time.

2. Using asynchronous circuits in the design to reduce the delays due to the usage of global clock.

II. CHALLENGES IN SYNCHRONOUS CIRCUITS

Traditional synchronous FPGA architectures are facing challenges with the growing logic size of chips in terms of
1. A single slow component or logic slows down the whole chip.
2. Challenges with designing reusable components.
3. To evenly distribute global clock signals all over the FPGA area requires great efforts because of clock skew.
4. FPGAs are more likely to contain a multitude of modules running at different clock frequencies, with data signals appearing to be asynchronous in the new clock domain when moving data across modules.
6. Increased power consumption.
7. Improved noise and electromagnetic compatibility (EMC) properties.
8. Process variations seriously affect circuit designs.
9. Performance Overhead.

Hence, Asynchronous techniques have become more significant from past decade of years due to the continuous scaling of VLSI technologies. In an asynchronous circuit, the next computation step can start immediately after the previous step has been completed. There is no need to wait for a transition of the clock signal. This leads potentially to a fundamental performance advantage for asynchronous circuits, an advantage that increases with the variability in delays associated with these computation steps.

III. IMPLEMENTATION OF PROPOSED ARCHITECTURE

The Architecture has sign calculator, exponent calculator, mantissa calculator, which works parallel, and a normalization unit.

It takes two IEEE 754 format single precision floating point numbers and produces the multiplied output. It also supports the features like underflow, overflow and invalid operations.

The implementation of Floating point multiplier Unit consists of two stages of multiplication calculation and Normalization. First stage includes three blocks which work in parallel.

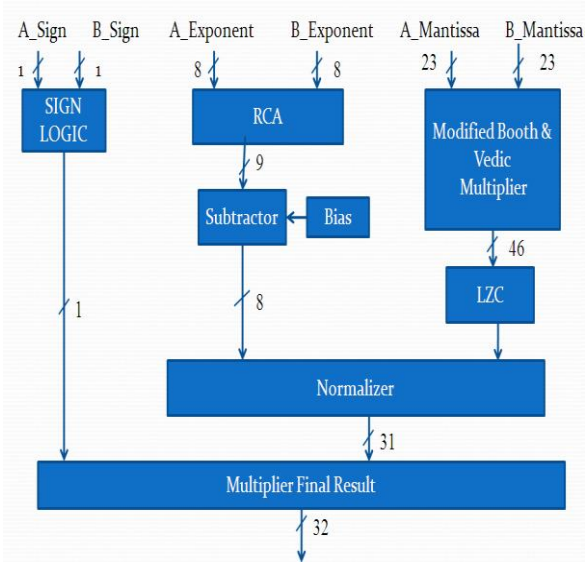

Figure 2: Floating Point Multiplier Architecture Using Booth and Vedic techniques

A. Sign Calculator: The Output Sign is the exclusive or of two sign bit inputs

B. Exponent Calculator: The input exponents are added using Ripple Carry Adder (RCA) and the bias is subtracted using Ripple Carry Subtractor (RCS) to produce the exponent of Output.

C. Mantissa Calculator: Output Mantissa is calculated by multiplying the mantissa's using multiplier (*Modified Booth/Vedic techniques*)

Second stage performs Normalization of the first stage output. It first calculates how much amount the mantissa needs to be left shifted using LZC (Leading Zero Counter) and finally produces the multiplier output.

IV. MULTIPLIERS USED IN THE ARCHITECTURE

A. Vedic multiplier:

For Mantissa calculations, Vedic and modified booth multipliers are used in the implementation. The design of Vedic Multiplier starts with 2x2 bit multiplier. Here, *"Urdhva Tiryakbhyam Sutra"* (Vertically and Crosswise Algorithm) has been used for multiplication to develop multiplier architecture.

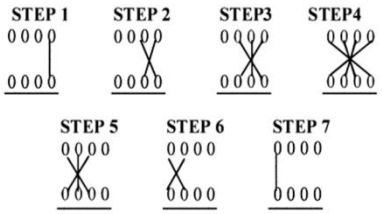

Figure 3: 4 X 4 bit Vedic multiplication

The expressions of the partial products obtained by multiplying $A=A_3A_2A_1A_0$ and $B= B_3B_2B_1B_0$ are

$P_1=A_0B_0$ and $Carry=C_0$

$P_2=A_1B_0+B_1A_0+C_0$ and $Carry=C_1$

$P_3=A_2B_0+B_2A_0+A_1B_1+C_1$ and $Carry=C_2$

$P_4=A_3B_0+A_0B_3+A_2B_1+A_1B_2+C_2$ &$Carry=C_3$

$P_5=A_3B_1+A_2B_2+A_1B_3+C_3$ and $Carry=C_4$

$P_6=A_3B_2+A_2B_3+C_4$ and $Carry=C_5$

$P_7=A_3B_3+C_5$

This Sutra shows how to handle multiplication of a larger number (N x N, of N bits each) by breaking it into smaller numbers of size (N/2 = n, say) and these smaller numbers can again be broken into smaller numbers (n/2 each) till 2 × 2 basic multiplier block. Hence, whole multiplication process is to be simplified.

First the basic block, 2×2 multipliers have been made then, using these blocks, 4×4 block and thereby using 4×4 block, 8×8 block and then finally 16×16 bit Multiplier has been made.

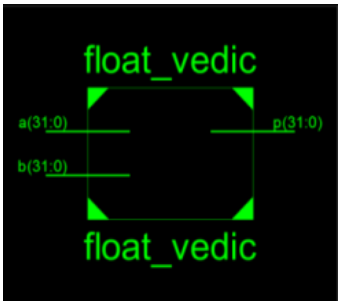

Figure 4: Black box view of single precision floating point vedic multiplier

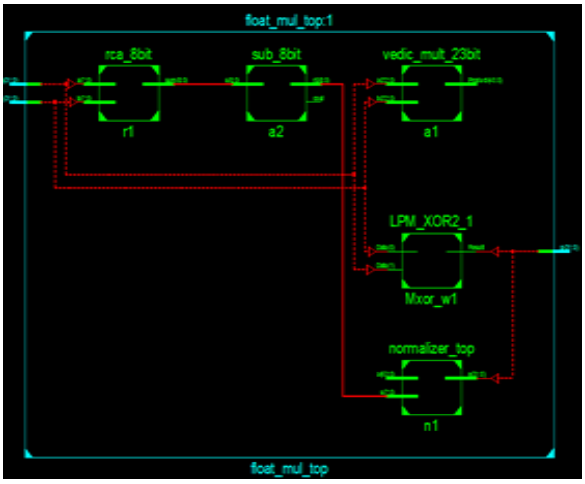

Figure 5: single precision floating point vedic multiplier

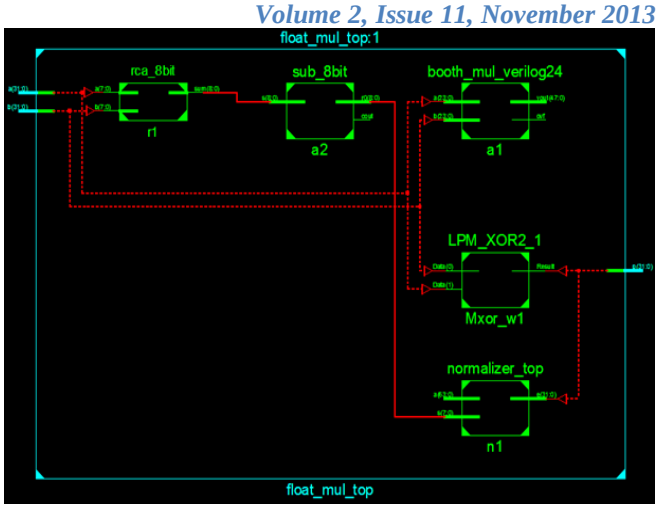

Figure 7: single precision floating point modified booth multiplier

B. *Modified Booth Multiplier*

Booth multiplication is smaller, faster multiplication algorithm through encoding the signed numbers to 2's complement, which is also a standard technique used in chip design, and provides significant improvements by reducing the number of partial product to half over "long multiplication" techniques.

Modified Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. Modified Booth algorithm reduces the number of partial products generated in a multiplication process through encoding the signed numbers to 2's complement according to the table shown below.

TABLE 1
Modified Booth Encoding Table

b2b1 b0	operation
000	All Zero's
001	Same Number
010	Same Number
011	Single Left Shift
100	2's complement & Left Shift
101	2's complement
110	2's complement
111	All zero's

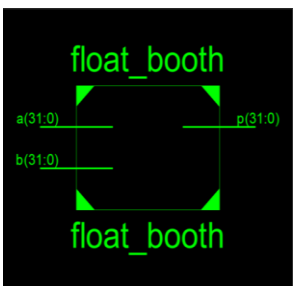

Figure 6: Black box view of single precision floating point modified booth multiplier

V. RESULTS:

Multiplier	Delay(ns)
Modified booth algorithm	87.476
Vedic multiplier	94.339

VI. CONCLUSION:

This paper presents the implementation of single precision floating point multiplier that supports the IEEE 754 binary interchange format. The whole design was captured in Verilog Hardware description language (HDL). The computation delays obtained for vedic and modified booth multipliers are 94.339ns and 87.476ns respectively. The design is implemented on a Xilinx ISE 9.2i tool targeting the Spartan 3E device xc3s1600e-5fg320.

VII. REFERENCES:

[1] An Asynchronous Floating-Point Multiplier by Basit Riaz Sheikh and Rajit Manohar,Computer Systems Laboratory, Cornell University
Ithaca, NY 14853, U.S.A

[2]Asynchronous FPGA Architecture with Distributed Control by Delong Shang, Fei Xia, Alex Yakovlev MSD Group, School of ECE, Newcastle University Newcastle upon Tyne, NE1 7RU, England, U.K.

[3]Scanning the Technology Applications of Asynchronous Circuits, C. H. (Kees) Van Berkel, Member, IEEE, Mark B. Josephs, And Steven M. Nowick, in proceedings of the IEEE, vol. 87, no. 2, February 1999.

[4]International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 8, February 2013 245 entitled "Vedic Mathematics for Fast Multiplication in DSP" by Nivedita A. Pande, VaishaliNiranjane, Anagha V. Choudhari, Lecturer, YCCE, Nagpur.

[5]Implementation of Vedic Multiplier using Different Architecture by ShikhaKaushik ,JavedAshraf ,Alfalah school of engineering andtechnology

[6]Floating Point Multiplier IP by SoftJin Technologies Pvt. Ltd.India, BANGALORE.