Synthesize of High Speed Floating-point Multipliers Based on Vedic Mathematics

Anjana S\textsuperscript{a,}* , Pradeep C\textsuperscript{a}, Philip Samuel\textsuperscript{b}

\textsuperscript{a}Department of ECE, SAINTGITS College of Engineering, Kottayam, Kerala, 686532, India

\textsuperscript{b}Division of IT, Cochin University of Science and Technology, Cochin, 682022, India

Abstract

This work proposes designing of high speed floating point multipliers. The multipliers are designed using Vedic Mathematics. The Vedic Multiplier (VM) has a regular structure therefore can be easily layout in a Silicon chip. Two different VMs are designed with different adder structures viz. Ripple Carry Adder (RCA) and Carry Look-ahead Adder (CLA). A single precision floating point multiplier is designed with these two VM structures. The multipliers are structurally modeled in Verilog HDL. The simulation and synthesis are done in Xilinx\textsuperscript{\textregistered} ISE 14.2. A comparison on the basis of area, power and delay is performed on Virtex-5 (xc5vlx110tf1136-1).

© 2015 The Authors. Published by Elsevier B.V.

Keywords: CLA; Floating Point Multiplier; FPU; RCA; Vedic Multiplier; Virtex-5;

1. Introduction

This paper discusses the designing of high speed, energy efficient floating point multiplier (FPM) which can be used for the designing of a reconfigurable multiplier capable of being configured as a FPM as well as an unsigned multiplier. Floating point numbers \textsuperscript{3} are one possible way of representing the real numbers in binary format. FPMs are the critical delay path components in DSP processors.

* Corresponding author. Tel: +91-9446414294
E-mail address: anjanas912@gmail.com
The floating point unit is a coprocessor that operates in parallel with the processor’s integer unit (Fig. 1). The FPU gets its instructions from the same instruction decoder and sequencer as the integer unit and shares the system bus with the integer unit. Other than these connections, the integer unit and FPU operate independently and in parallel. For single instruction processors, the operation of these two units depends upon the input signal data-type. Hence only one unit in two will be active at any instant of time. Or in other words, the other unit will simply reside on the FPGA wasting area, power and cost. Thus it will be advantageous if the multiplier unit reconfigures, depending on the input data-type, as an integer (unsigned) multiplier or a FPM.

For designing high speed multipliers, the Vedic Multipliers (VMs) are chosen. These multipliers provide high speed of operation while they are area efficient too. A single precision FPM contains a 24×24 unsigned multiplier, an unsigned adder, a sign bit calculator and a normalizer. The unsigned multiplier is designed using a 24×24 VM. The performance of VM improves when they constitute high speed adder.

A comparison on the basis of area, delay and power is also performed for floating point multiplier (using RCA and CLA) with the array multiplier and carry save multiplier. All multipliers are simulated and synthesized using Xilinx® ISE 14.2.

This paper consists of 5 sections: section 2 discusses various previous works; section 3 explains the proposed work. Section 4 is the experimental results and section 5 concludes the paper.

2. Previous Works

Implementation of floating point multiplier on FPGA is explained with a study of its effects on 4-input LUTs by Pardeep Sharma and Ajay Pal Singh. Floating point format along with the steps in conversion of decimal to floating point numbers is explained in this paper. The results of LUT utilization on Spartan-2, Spartan-2e, Spartan-3, Spartan-3e, Virtex, Virtex-2, Virtex-2p, Virtex-4, Virtex-E are provided and concluded that the implementation on Virtex-4 has the least LUT utilization.

Lokesh Bhardwaj and Sakshi Bajaj explained the multiplication algorithm for floating point numbers:

- Multiplying the significand (including the hidden bit)
- The decimal point is placed in the result
- The exponents are added (E1+E2-Bias)
- The sign bit of the product is obtained (S1 XOR S2)
- The result is normalized
- Normalizing is done such that there is a ‘1’ before the decimal point
- The result is rounded to nearest number

A new architecture for high speed 4x4 Vedic Multipliers based on two different sutras namely, Urdhva Tiryakbhyam and Nikhilam is presented by S. Karthika and Priyanka Udhayabhanu. Novel architectures for 2x2 and 4x4 Vedic Multipliers are introduced in this paper. This paper gives the idea that an N×N Vedic Multiplier can
be implemented using N/2×N/2 Vedic Multipliers and three adder structures. Also this paper explains that the delay of a Vedic Multiplier greatly depends on its adder structures. Mohammed Hasmat Ali and Anil Kumar Sahani presented a detailed study of different multipliers based on Array Multiplier, Constant Coefficient Multiplication (KCM) and multiplication based on Vedic Mathematics. The multipliers are compared based on LUTs and path delays. Results show that Vedic Urdhva Tiryakbhyam sutra is fastest multiplier with least path delay. Ch. Harish Kumar implemented Urdhva, Nikhilam Vedic Multipliers and Array Multiplier and analysed on the basis of power, area and delay. The multipliers are modelled using Verilog HDL and the simulation is done in Xilinx ISE 10.1. It is found that the Urdhva multiplier is the best multiplier compared to Array and Nikhilam multipliers. A high speed multiplier design which has a reconfigurable path for IEEE 754 single precision and two units- Add unit and Aligner-normalizer unit is also designed.

3. Proposed Works

FPMs are the key components in applications that involve large dynamic range, high precision and easy operation rules. Floating point numbers are one possible way of representing real numbers in binary format. The IEEE 754 standard for binary floating point arithmetic provides a precise specification of floating point number formats computation operations and exceptions and their handling.

The IEEE format specifies two main groups of floating point format:

- Single Precision
- Double Precision

The single precision format representation of floating point numbers has three fields:

- Sign bit (S)
- Mantissa (M)
- Exponent (E)

Fig. 2 shows the single precision format and it is represented as:

\[
(-1)^S \times 2^{(E-Bias)} \times M
\]  

3.1. Floating Point Multiplier Architecture

The single precision FPM is shown in Fig. 3 and it contains four different modules:

- Sign bit calculator: Multiplying two numbers results in a negative number if one of the multiplied numbers is of a negative value. A sign bit calculator is implemented using an XOR gate.
- Unsigned Adder: The unsigned adder is responsible for adding the exponent of the first input to the exponent of the second input and subtracting the bias (127\text{b}) from the addition result (i.e., \(A_{\text{Exponent}} + B_{\text{Exponent}} - \text{Bias}\)). An 8-bit Ripple Carry Adder (RCA) and a 9-bit Ripple Borrow Subtractor (RBS) are used here.
- Unsigned Multiplier: This unit is responsible for multiplying the unsigned multiplicands and placing the decimal point in the multiplication product. The unsigned significant multiplication is done on 24-bit. The multiplier
performance is chosen such that it does not affect the whole multiplier performance. A 24×24 VM is used. The result at this stage is called intermediate product.

- Normalizer: The result of the significand multiplication must be normalized to have a leading ‘1’ just to the left of the decimal point. Since the inputs are normalized numbers the intermediate product has the leading ‘1’ at bit 46 or 47.

![Floating point Multiplier Architecture](image1)

**Fig. 3. Floating point Multiplier Architecture.**

### 3.2. Vedic Multiplier

Vedic Multipliers\(^{14, 15, 16, 17, 18}\) are based on Vedic Mathematics. Vedic Mathematics is an ancient mathematics concept developed by Sri Bharati Krishna Tirthaji between 1911 & 1918. It includes sixteen sutras, dealing with arithmetic, algebra, geometry, trigonometry and calculus.

Vedic Multiplier discussed in this context is based on Urdhva Tiryakbhyam Sutra (literally- “Vertically and Crosswise”). Using this sutra the partial products are generated and summed in. This multiplier design has the advantage that as the number of bits increases, the gate delay and area increases slowly. Therefore it is time, space and area efficient.

![2×2 Vedic Multiplier](image2)

**Fig. 4. 2×2 Vedic Multiplier.**
The 2×2 Vedic Multiplier (Fig. 4) is implemented using four AND gates and two half adders. By using this sutra, the larger number (N×N) is broken down into smaller numbers (N/2×N/2) and these smaller numbers are again broken into still smaller numbers (N/4×N/4) till we reach multiplicand of size 2×2, thus simplifying the whole multiplication process. Hence 4×4 Vedic Multiplier is implemented using four 2×2 Vedic Multipliers and three 4-bit adders. Also an 8×8 Vedic Multiplier is implemented using four 4×4 Vedic Multipliers and three 8-bit adders. Thus an N×N Vedic Multiplier is implemented using four (N/2×N/2) Vedic Multipliers and three N-bit adders. A 32×32 VM is shown in Fig. 5.

- Vedic Multiplier using CLA: Carry Look-ahead algorithm helps in improving the speed of addition operation. Using this algorithm the carry bits for the successive stages can be calculated in advance depending on the input signals. CLA calculates the next carry by exploiting the fact that the carry generated at any bit-position depends on the three inputs to that position. A 32×32 VM using CLA is shown in Fig. 6. $P_i=X_i \oplus Y_i \text{ (Carry Propagation)}$, $G_i=X_i \land Y_i \text{ (Carry Generation)}$, $S_i=X_i \oplus Y_i \oplus C_i \text{ (Sum Generation)}$ and $C_{i+1}=G_i \lor (P_i \land C_i) \text{ (Next Carry)}$
Hence in Vedic Multiplier using CLA, the speed will be improved appreciably, since CLA unit produces output much faster than RCA unit.

- Vedic Multiplier using RCA: A ripple carry adder can be constructed with full adders connected in cascaded, with the carry output from each full adder connected to the carry input of the next full adder in the chain. Here the output is known only after the carry generated by the previous stage is produced. Thus, the sum of the most significant bits is only available after the carry signal has rippled through the adder from the least significant stage to the most significant stage. As a result, the final sum and carry bits will be valid after a considerable delay. A $32 \times 32$ VM using RCA is shown in Fig. 7.

![Fig. 7. 32×32 Vedic Multiplier using RCA.](image)

Therefore, the performance of the VM using RCA depends on how fast the RCA unit produces its output. As the size of the multiplier increases, the length of the RCA unit also increases, thus takes longer time to produce output. Hence the speed of this multiplier decreases as the number of bits increases.

4. Experimental Results

The multipliers are structurally modeled in Verilog HDL. The simulation and synthesis are done in Xilinx® ISE 14.2. A comparison on the basis of area, power and delay is performed on Virtex-5 (xc5vlx110tf1136-1). The entire structure is same for the four floating point multiplier designs except for the $24 \times 24$ multiplier block, which will be designed with separate integer multipliers. Hence area and speed of the floating point multipliers depends on the performance of the integer multipliers.

| Design                        | No. of LUTs (%) | No. of IOBs (%) |
|-------------------------------|----------------|-----------------|
| Vedic Multiplier using CLA    | 1              | 15              |
| Vedic Multiplier using RCA    | 2              | 15              |
| Carry Save Multiplier         | 1              | 15              |
| Array Multiplier              | 1              | 15              |

The Table 1 gives the device utilization summary of single precision floating point multipliers and Table 2 gives the device utilization of corresponding integer multipliers. The Vedic Multipliers consume more area than other multipliers. It is expected due to the presence of three adder structures and four $16 \times 16$ multipliers, each of which in turn contains $8 \times 8$ multiplier structures and so on. Carry Save
Multipliers and Array Multipliers use the same number of LUTs since both designs contain equal number of full adders.

| Integer Multiplier Design (32×32) | No. of LUTs (%) | No. of IOBs (%) |
|-----------------------------------|-----------------|-----------------|
| Vedic Multiplier using CLA        | 3               | 20              |
| Vedic Multiplier using RCA        | 3               | 20              |
| Carry Save Multiplier             | 2               | 20              |
| Array Multiplier                  | 2               | 20              |

Fig. 8 gives the device utilization summary of FPMs and Integer multipliers.

![Device Utilization Summary of FPMs and Integer Multipliers](image)

The Table 3 and Table 4 give the maximum combinational path delay and on-chip total power of single precision floating point multipliers and integer multipliers respectively.

| Single Precision Floating Point Multiplier Design | Maximum Combinational Path Delay (ns) | On-chip Total Power (W) |
|--------------------------------------------------|--------------------------------------|-------------------------|
| Vedic Multiplier using CLA                       | 31.832                               | 1.042                   |
| Vedic Multiplier using RCA                       | 33.246                               | 1.042                   |
| Carry Save Multiplier                            | 36.316                               | 1.042                   |
| Array Multiplier                                 | 56.481                               | 1.042                   |

The Vedic Multipliers are found to be having more speed than other multipliers. This is due to the partial products are generated and summed in parallel. Also the delay associated with the Vedic Multipliers is mainly due to the propagation of carry bits through adders. By using high speed adders like CLA, the performance of Vedic Multipliers can again be improved.

Carry Save Multiplier has much less path-delay than Array Multiplier because addition is done in parallel without waiting for the previous results. Array Multipliers suffers the highest path delay since addition operation can be performed only if the previous result is available.

Fig. 9 shows the delay and power of FPMs and Integer multipliers.
Thus the VM using CLA is found to be faster than other multipliers (43.64% speed trade-off as compared to Array Multiplier). This high speed compensates for the slight increase in its device utilization.

| Single Precision Floating Point Multiplier Design | Maximum Combinational Path Delay (ns) | On-chip Total Power (W) |
|--------------------------------------------------|--------------------------------------|-------------------------|
| Vedic Multiplier using CLA                       | 30.934                               | 1.042                   |
| Vedic Multiplier using RCA                       | 35.286                               | 1.042                   |
| Carry Save Multiplier                            | 70.064                               | 1.042                   |
| Array Multiplier                                 | 73.485                               | 1.042                   |

Fig. 9. Delay and Power of (a) FPMs; (b) Integer Multipliers.

5. Conclusion

A high speed FPM was designed. The multiplier was based on Vedic Mathematics. Vedic Multipliers were chosen due to its regular structure, gradually increasing delay and area. VMs also have the advantage that the delay in these multipliers is due to the time taken by the carry bits to propagate through the adders. Thus two different VMs were designed; one using CLA and the other using RCA. The design entry was done using CLA and the other using RCA. The design entry was done using Xilinx ISE 14.2 and was implemented on Virtex-5.

The VM designs were compared on the basis of area, power and delay with the Array Multiplier and the Carry Save Multiplier. The VMs were found to be having high speed as compared to other multipliers. This is because the partial products are generated and summed in parallel. The presence of CLA improved the speed of the VM. Although the area of these multipliers was slightly high, it compensates for the high speed of these multipliers (43.64% trade-off as compared to Array Multiplier).

Using these high speed multipliers the reconfigurable multiplier is to be designed. The reconfigurable module will be instantiated with the two instances: the integer multiplier and the FPM. This design will have two dynamic modules, integer multiplier and the FPM. The concept of Partial Reconfiguration will be used here.

The idea of Partial Reconfiguration is introduced by Xilinx® in late 1990s. Partial Reconfiguration (PR) is a feature of modern FPGAs that allows a subset of the logic fabric of a FPGA to dynamically reconfigure while the remaining logic continues to operate unperturbed. PR of FPGAs is a compelling design concept for general purpose reconfigurable systems for its flexibility and extensibility. PR is supported by the Xilinx Virtex series FPGAs since the Virtex-4. Xilinx PR extends the inherent flexibility of the FPGA by allowing specific regions of the FPGA to be programmed with new functionality while applications continue to run in the remainder of the device. PR
addresses three fundamental needs by enabling the designer to: Reduce cost and/or board space, change a design in the field and reduce power consumption.

References

1. Priyanka Koneru, Tinnanti Sreenivasu, Addanki Purna Ramesh. Asynchronous Single Precision Floating Point Multiplier using Verilog HDL, *I. J. of Advanced Research in Electronics and Communication Engineering*, Nov 2013.
2. Intel Architecture Software Developer’s Manual.
3. Pardeep Sharma, Ajay Pal Singh. Implementation of Floating Point Multiplier on Reconfigurable Hardware and Study its Effect on 4 input LUTs, *I. J. of Advanced Research in Computer Science and Software Engineering*, Jul 2012.
4. Lokesh Bhardwaj, Sakshi Bajaj. Design and Implementation of Efficient Adder based Floating Point Multiplier, *I. J. of VLSI and Embedded Systems* Jul 2014.
5. S Karthik, Priyanka Udayabhanu. FPGA Implementation of High Speed Vedic Multipliers, *I. J. of Engineering Research and Technology*, Dec 2012.
6. Mohammed Hasmat Ali, Anil Kumar Sahani. Study, Implementation and Comparison of Different Multipliers based on Array, KCM and Vedic Mathematics Using EDA Tools, *I. J. of Scientific and Research Publications*, Jun 2013.
7. Ch Harish Kumar. Implementation and Analysis of Power, Area and Delay of Array, Urdhva, Nikhilam Vedic Multipliers, *I. J. of Scientific and Research Publications*, Jan 2013.
8. Yogesh Kumar and RK Sharma. Clock-less Design of Reconfigurable Floating Point Multiplier, *International Association of CFOS and Corporate Treasurers* 2011.
9. Asiya Thapaswin Pattan, V Ramesh, C Md Aslam. An Efficient Implementation of Floating Point Multiplier, *I. J. of Engineering Research and Technology*, Sep 2012.
10. M Pravallika, V Vamsi Mohana Krishna. Design and Verification of High Speed and Efficient Asynchronous Floating Point Multiplier, *I. J. of Engineering Research and Technology*, Jul 2013.
11. Naresh Grover, MK Soni. Design of FPGA based 32-bit Floating Point Arithmetic Unit and Verification of its VHDL code using MATLAB, *I. J. Information Engineering and Electronics Business, MECS*, Feb 2014.
12. R Sai Siva Teja, A Madhusudhan. FPGA Implementation of Low- Area Floating Point Multiplier using Vedic Mathematics, *I. J. of Emerging Technology and Advanced Engineering*, Dec 2013.
13. IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.
14. P. Saha, A. Banerjee, A. Dandapat, P. Bhattacharyya. ASIC Design of A High Speed Low Power Circuit for Factorial Calculation Using Ancient Vedic Mathematics, *Elsevier, Microelectronics Journal 42*, 2011.
15. C Sheshavali, K Nirajan Kumar. Design and Implementation of Vedic Multiplier, *I. J. of Engineering Research and Development*, Sep 2013.
16. M Ganesh Kumar, IV Rameshwar Reddy, K Kameswara Reddy. Design of Two Variable Multiplier using Vedic Mathematics and ROM Approach, *I. J. of Engineering Research and Technology*, Sep 2013.
17. NG Nirmal, Dr. DT Ingle. Novel Delay Efficient Approach for Vedic Multiplier with Generic Adder Module, *I. J. of Engineering Research and Applications*, May-Jun 2013.
18. Gaurav Sharma, Arjun Singh Chauhan, Himanshu Joshi, Satish Kumar Alaria. Delay Comparison of 4 by 4 Vedic Multiplier based on Different Adder Architectures using VHDL, *I. J. of Engineering & Applied Science Research*, Jun 2013.
19. Raminder Preet Pal Singh, Praveen Kumar, Balwinder Singh. Performance Analysis of 32-Bit Array Multiplier with a Carry Save Adder and with a Carry Look-ahead Adder, *I. J. of Recent Technology and Engineering*, Nov 2009.
20. Xilinx, Inc. Virtex-5 FPGA User Guide, UG190 (V5.4) Mar 16, 2012.
21. Xilinx, Inc. Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite, WP374 (V1.2) May 30, 2012.
22. Xilinx, Inc. Partial Reconfiguration User Guide, UG702 (V14.5) Apr 26, 2013.
23. Xilinx, Inc. Vivado Design Suite User Guide: Partial Reconfiguration, UG909 (V2013.3) Oct 30, 2013.