FPGA Implementation of Vedic Floating Point Multiplier

Ravi Kishore Kodali, Lakshmi Boppana and Sai Sourabh Yenamachintala
Department of Electronics and Communication Engineering
National Institute of Technology, Warangal
WARANGAL 506004, INDIA
E-mail: ravikkodali@gmail.com

***Abstract*—Most of the scientific operation involve floating point computations. It is necessary to implement faster multipliers occupying less area and consuming less power. Multipliers play a critical role in any digital design. Even though various multiplication algorithms have been in use, the performance of Vedic multipliers has not drawn a wider attention. Vedic mathematics involves application of 16 sutras or algorithms. One among these, the *Urdhva tiryakbhyam sutra* for multiplication has been considered in this work. An IEEE-754 based Vedic multiplier has been developed to carry out both single precision and double precision format floating point operations and its performance has been compared with Booth and *Karatsuba* based floating point multipliers. Xilinx FPGA has been made use of while implementing these algorithms and a resource utilization and timing performance based comparison has also been made.**

Keywords:- Vedic multiplication, FPGA, Floating Point

I. Introduction

Fixed point and floating point number representations are being widely used by various applications as in the design of Digital Signal Processors (DSPs). High speed computation with high degree of accuracy are quite essential in a broad range of applications form basic consumer electronics to sophisticated industrial instrumentation. When compared to a fixed point representation, floating point can represent very small and very large numbers, thereby increasing the range of representation. Dynamic range and precision considerations determine whether fixed point or floating point representations are to be used for a specific application. An example, where dynamic range requirements demand the usage of floating point representation is matrix inversion. Various floating point arithmetic operations are extensively supported by all the microprocessors and computer systems. Among various floating point arithmetic operations, multiplication is more frequently used in many applications. The efficient FPGA implementation of complex floating point functions requires the use of efficient multiplication algorithms. An efficient multiplication algorithm, which facilitates optimized utilization of resources and minimum time delay must be used for effective implementation of floating point processors. A floating point multiplier is the most commonly used component in many digital applications such as digital filters, data processors and DSPs. About 37 % of the floating point instructions in benchmark applications constitute floating point multiplications [1].

Apart from high performance, energy efficient hardware for performing floating point multiplication is necessary for embedded systems. Crucial part of floating point multiplication involves multiplication of mantissa. Increase in the number of applications involving repeated use of multiplication and the need to complete multiplication faster with limited number of resources resulted in many multiplication algorithms. Among various algorithms, *Urdhva Triyakbhyam* originating from Vedic mathematics is found to be efficient in terms of area as well as time. Vedic mathematics put forth by Swami Bharati Krishna Tirtha as an embodiment of 16 sutras and 13 sub sutras provide algorithms for various arithmetic operations [2], [3]. The IEEE 754 standard specifies format for floating point representation of numbers. In addition, this format also specifies floating point arithmetic, inter conversion between floating point and integer formats, conversions among various floating point formats, and handling of exceptions. Another efficient algorithm is *Karatsuba* algorithm, which is based on divide and conquer approach [4], [5]. The rest of the paper is organized as follows: section II provides literature review, section III presents IEEE-754 floating point multiplier, section IV gives an overview of vedic sutras, section V presents results and simulation analysis and the final section concludes.

II. Literature Review

The floating point numbers in binary number system are represented in two formats namely, single and double precision. These formats are characterized by exponent, mantissa and sign fields. The single precision comprises of 32- bits with 23- bit mantissa, an 8- bit exponent and one sign bit. The double precision comprises of 64- bits with 52- bit mantissa, 11- bit exponent and one sign bit. The sign bit represents the sign of the number. A $'0'$ in the most significant bit (MSB) position denotes positive numbers, while $'1'$ represents negative numbers [6]. In IEEE-754 format, the floating point numbers are represented by:

Single Precision:

$$\underbrace{Sign}_{\text{1-bit}}\ \underbrace{Exponent}_{\text{8-bits}}\ \underbrace{Mantissa}_{\text{23-bits}}$$

Double Precision:

$$\underbrace{Sign}_{\text{1-bit}}\ \underbrace{Exponent}_{\text{11-bits}}\ \underbrace{Mantissa}_{\text{52-bits}}$$

978-1-4799-1823-2/15/$31.00 ©2015 IEEE

Floating point numbers are generally of the form $((1)^S)(F)(2^E)$. F represents the value in the fractional field, while E represents the value in the exponent field. In general, the mantissa part is normalized by adding 1 as MSB. When the exponent is too large to be represented in the exponent field then it indicates an overflow condition. When the negative exponent becomes too large to fit in the exponent field, then it indicates an underflow condition. IEEE-754 standard introduces a notation called NaN (Not a Number) as a result for invalid operations such as division of zero with zero, subtracting infinity from infinity. A detailed description on the use of *Karatsuba* algorithm recursively for the use in such applications is described in [4], [3]. Floating point multiplication utilizes a 32- bit and 53- bit multiplier for multiplication of its significand in single precision and double precision respectively. Different multiplication algorithms are used for multiplying the significands. The advantage of using Vedic multiplication algorithm over conventional methods is described in [6]. Floating point multiplication can also be done by using pipelining method wherein addition of exponents, multiplication of significands and calculation of sign bit can be done in parallel. Implementation of floating point multiplier using *Karatsuba* algorithm incorporating pipelining techniques with a latency of 8 cycles is implemented in [7]. Among arithmetic operations, multi-precision arithmetic is one of the most time consuming operations with $O(n^2)$ time complexity. Application of *Karatsuba* algorithm reduces the time complexity of multiplication from $O(n^2)$ to $O(n^{\log_2 3})$ [8].

Application of vedic multiplication algorithm to DSP operations reduces 40%-60% of time from the conventional procedures. A detailed description on implementation of various DSP operations using Vedic multiplication technique is described in [9]. The problem of finding the product of two numbers by successively forming the complement of the smaller of two operands is described in [10]. The algorithm utilized is proposed in Vedic mathematics. Various techniques described in Vedic mathematics can be used to design an ALU, which is compatible for co-processors. This Vedic co-processor will be more efficient than its conventional counterpart [11]. A performance comparison of array multiplier and vedic multiplier has been presented in [12].

III. FLOATING POINT MULTIPLICATION

Floating point multiplication of numbers represented using either single precision format or double precision format involve calculation of sign bit, exponent and mantissa of the product and then normalizing and rounding off the result. Sign bit is calculated by X-OR operation of the sign bits of two operands. The exponents of both the operands are added to obtain the product exponent. The resultant sum is subtracted from 127 in case of single precision and 1023 in case of double precision to obtain its biased exponent. The addition operation involves the use of 8- bit and 11- bit ripple carry adder for single and double precisions respectively. In order to improve the performance high speed adders can be used. The mantissa is multiplied using any of the multiplication algorithms [7].

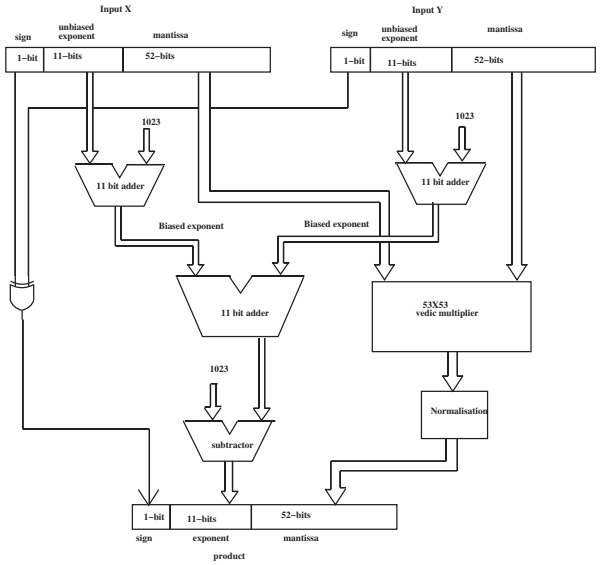


Fig. 1: IEEE-754 Double Precision Floating Point multiplier

When *Karatsuba* algorithm is implemented recursively, each operand is split into two parts with each part containing equal number of bits when the number of bits in the given number is even. When the number of bits are odd, the number is split into two parts with one part containing $\frac{(n+1)}{2}$- bits and the other part containing $\frac{(n-1)}{2}$- bits.

Algorithm 1 Floating Point Multiplication Algorithm

INPUT: Two floating point numbers a and b
OUTPUT: Floating point number c
Single Precision:
$c(31) \leftarrow a(31) xor b(31)$
$c(30:23) \leftarrow a(30:23) + b(30:23) - 1111111$
$Product(47:0) \leftarrow 1.a(22:0) * 1.b(22:0)$
Normalise and round off product to obtain $c(22:0)$
Double precision:
$c(63) \leftarrow a(63) xor b(63)$
$c(62:52) \leftarrow a(62:52) + b(62:52) - 111111111$
$Product(105:0) \leftarrow 1.a(51:0) * 1.b(51:0)$
Normalize and round off product to obtain $c(51:0)$

IV. AN OVERVIEW OF VEDIC ALGORITHM

The current work discusses Urdhva Tiryakbhyam sutra of Vedic mathematics and is applied to the multiplication of mantissa of floating point number. *Urdhva Tiryakbhyam* stands for vertical and crosswise multiplication. The individual bits of both the operands are subjected to vertical and cross wise multiplication. Vedic multiplier is proved to be more efficient in terms of area and time delay. An efficient application of this algorithm to various DSP operations is described in [9].

Vedic mathematics sutras can also be applied to perform the multiplication of numbers close to powers of 10 in a simple procedure avoiding the necessity to follow lengthy conventional procedure of multiplication [10]. Figure 2 shows

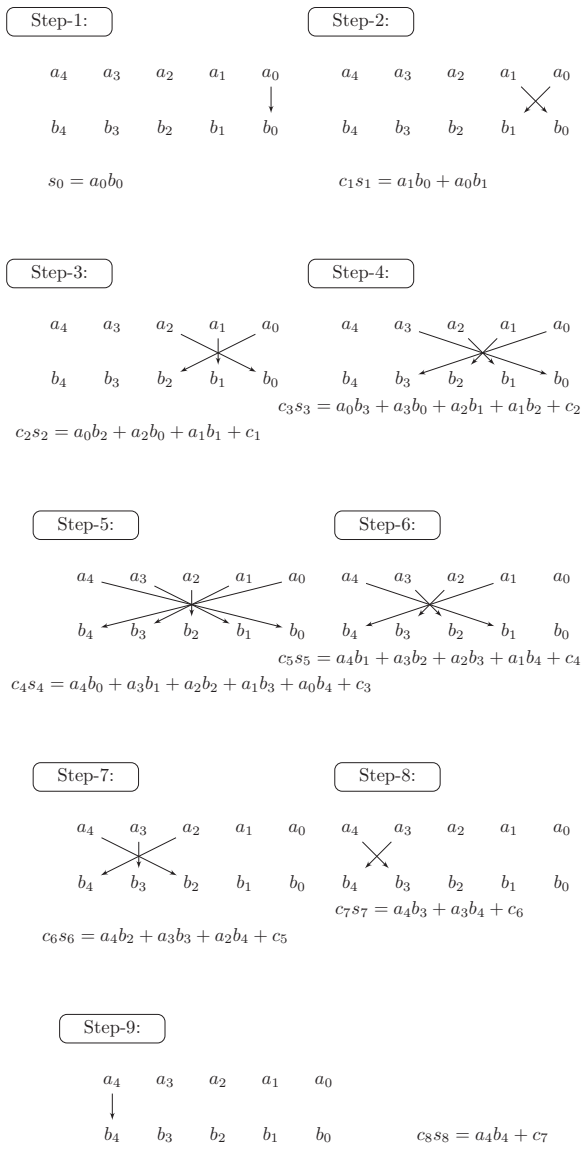


Fig. 2: Vedic multiplication illustration [3]

an illustration of Vedic multiplication involving two 5- bit numbers. In the case of single precision floating point multiplication, a 6- bit multiplier is designed using the vedic algorithm and utilizing this 6- bit multiplier as a basic component, the required 24- bit multiplier is obtained. In case of double precision floating point multiplication, a 7- bit multiplier is used as a basic component to construct the desired 53- bit multiplier to multiply the significands of both the operands as given in Algorithm -3. A comparison of time delays for various types of multipliers for different key lengths is presented in [11]. This comparison reveals the efficiency of Vedic multiplication algorithm.

V. RESULTS AND SIMULATION

The floating point multiplication in both single precision and double precision using Urdhva Tiryakbhyam algorithm

Algorithm 2 VEDIC ALGORITHM

INPUT: n- bit Multiplicand and Multiplier
OUTPUT: $2n$- bit product
$k \leftarrow 0$
S(k): $2n$- bit vector initialized to 0
for $i = 0$ **to** $(n-1)$ **do**
  for $j = 0$ **to** i **do**
    $S(k) = S(k) + a(i) \times b(i-j)$
  end for
  $k = k + 1$
end for
for $i = (n-1)$ **to** 1 **do**
  for $j = (n-1)$ **to** i **do**
    $S(k) = S(k) + a(i) \times b(n-(i-j))$
  end for
  $k = k + 1$
end for
for $i = 0$ **to** $(k-1)$ **do**
  $P = P + S(i)$
end for

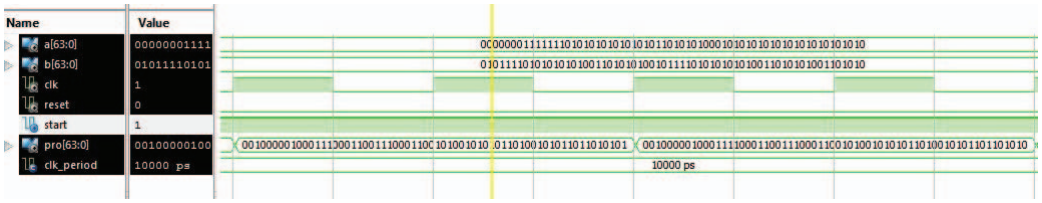
Algorithm 3 Floating point multiplication using Vedic algorithm

function $vedic(x[n], y[n])$
$M = 6$ Single Precision and $M = 7$ Double Precision
if $(n = M)$ **then**
  Perform Vedic multiplication {as given Figure -2}
else
  $A_1 \leftarrow vedic(x_L, y_L)$
  $\{x_L$ - lower $\frac{N}{2}$- bits and x_H - upper $\frac{N}{2}$- bits$\}$
  $A_2 \leftarrow vedic(x_H, y_H)$
  $A_3 \leftarrow vedic(x_L, y_H)$
  $A_4 \leftarrow vedic(x_H, y_L)$
  $S_1 + C_1 \leftarrow A_3 + A_4$
  $\{S_i, C_i$ indicate N- bit sum and the carry generated$\}$
  $P_1 \leftarrow A_{1H} \& A_{2L}$ {& denotes concatenation operation}
  $S_2 + C_2 \leftarrow P_1 + S_1$
  $S_3 + C_3 \leftarrow C_1 + C_2$
  $P_2 \leftarrow (\frac{N}{2} - 2)0s \; \& C_1 \; \& C_2$
  $S_4 \leftarrow P_2 + A_{2H}$
end if
return $S_4 \& S_2 \& A_{1L}$

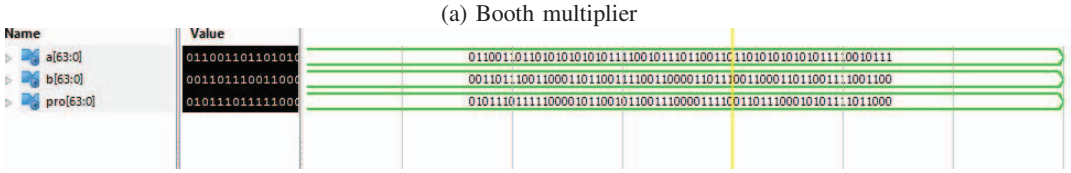
of vedic mathematics is synthesized using 7v2000tflg1925-2 device of Virtex-7 family. Its performance is compared with the floating point multipliers utilizing Booth and *Karatsuba* multiplication algorithms. Table -I provides the device utilization summary and time delay for both single precision and double precision floating point multiplications using three different algorithms and from this table it can be inferred that compared to *Karatsuba* multiplier, Vedic multiplier utilizes less number of resources and has shorter time delay. Even though Booth multiplier occupies less number of resources, due to its larger time delay making it inefficient for many applications. Booth multiplication algorithm can be efficiently

TABLE I: Comparison of Device Utilization and Timing Summaries of Booth, Karatsuba and Vedic IEEE-754 multipliers

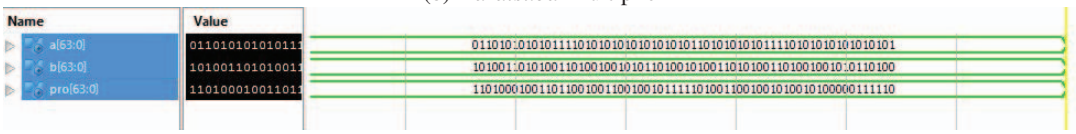
Logic Utilization	Single Precision			Double Precision		
	Booth Multiplication	*Karatsuba* Multiplication	Vedic Multiplication	Booth Multiplication	*Karatsuba* Multiplication	Vedic Multiplication
Number of Slice LUTs	163	2928(1%)	1121(0%)	324	15494 (10%)	9150(6%)
Number of LUT FF pairs Used	164	2928	1121	325	15494	9150
Number of IOs	99	96	96	195	192	192
Number of Bonded IOBs	99	96(16%)	96(16%)	195	192 (32%)	192 (32%)
Time Delay	47.33 ns	28.020 ns	27.760 ns	82.939 ns	34.081 ns	30.381 ns



(a) Booth multiplier



(b) *Karatsuba* Multiplier



(c) *Vedic* Multiplier

Fig. 3: Simulation Results for IEEE-754 Double Precision format multiplication

implemented for highly pipelined architectures, however it is a poor choice for single cycle multiplication [13].

VI. CONCLUSIONS

Both the IEEE-754 single precision and double precision floating point format multipliers using Booth, *Karatsuba* and Vedic algorithms have been synthesized using Xilinx Virtex-6 FPGA device. Based on the device utilization and performance comparison, Vedic multiplier outperforms *Karatsuba* multiplier both for single precision and double precision formats. Even though Booth multiplier occupies least resources, it is also the slowest.

REFERENCES

[1] G. Even and P.-M. Seidel, "A comparison of three rounding algorithms for ieee floating-point multiplication," *Computers, IEEE Transactions on*, vol. 49, no. 7, pp. 638–650, Jul 2000.

[2] Y. Bansal, C. Madhu, and P. Kaur, "High speed vedic multiplier designs-a review," in *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, March 2014, pp. 1–6.

[3] R. K. Kodali, S. S. Yenamachintala, and L. Boppana, "Fpga implementation of 160- bit vedic multiplier," in *Devices, Circuits and Communications (ICDCCom), 2014 International Conference on*, Sept 2014, pp. 1–5.

[4] X. Fang and L. Li, "On karatsuba multiplication algorithm," in *Data, Privacy, and E-Commerce, 2007. ISDPE 2007. The First International Symposium on*, Nov 2007, pp. 274–276.

[5] R. K. Kodali, S. K. Gundabathula, and L. Boppana, "Fpga implementation of ieee-754 floating point karatsuba multiplier," in *Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on*, July 2014, pp. 300–304.

[6] A. Kanhe, S. Das, and A. Singh, "Design and implementation of floating point multiplier based on vedic multiplication technique," in *Communication, Information Computing Technology (ICCICT), 2012 International Conference on*, Oct 2012, pp. 1–4.

[7] A. Mehta, C. Bidhul, S. Joseph, and P. Jayakrishnan, "Implementation of single precision floating point multiplier using karatsuba algorithm," in *Green Computing, Communication and Conservation of Energy (ICGCE), 2013 International Conference on*, Dec 2013, pp. 254–256.

[8] S. Erdem and C. Koc, "A less recursive variant of karatsuba-ofman algorithm for multiplying operands of size a power of two," in *Computer Arithmetic, 2003. Proceedings. 16th IEEE Symposium on*, June 2003, pp. 28–35.

[9] A. Itawadiya, R. Mahle, V. Patel, and D. Kumar, "Design a dsp operations using vedic mathematics," in *Communications and Signal Processing (ICCSP), 2013 International Conference on*, April 2013, pp. 897–902.

[10] V. Dave and C. Coulston, "Multiplication by complements," in *Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on*, Feb 2012, pp. 153–156.

[11] M. Ramalatha, K. Dayalan, P. Dharani, and S. Priya, "High speed energy efficient alu design using vedic multiplication techniques," in *Advances in Computational Tools for Engineering Applications, 2009. ACTEA '09. International Conference on*, July 2009, pp. 600–603.

[12] P. Saha, A. Banerjee, P. Bhattacharyya, and A. Dandapat, "High speed asic design of complex multiplier using vedic mathematics," in *Students' Technology Symposium (TechSym), 2011 IEEE*, Jan 2011, pp. 237–241.

[13] M. Paramasivam and R. Sabeenian, "An efficient bit reduction binary multiplication algorithm using vedic methods," in *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, Feb 2010, pp. 25–28.