An efficient binary multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm

Arish S
School of VLSI Design and Embedded Systems
National Institute of Technology Kurukshetra
Kurukshetra, India
arishsu@gmail.com

R.K.Sharma
School of VLSI Design and Embedded Systems
National Institute of Technology Kurukshetra
Kurukshetra, India
rksharama@nitkkr.ac.in

Abstract: Binary multiplication is an important operation in many high power computing applications and floating point multiplier designs. And also multiplication is the most time, area and power consuming operation. This paper proposes an efficient method for unsigned binary multiplication which gives a better implementation in terms of delay and area. A combination of Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm (Vedic Mathematics) is used to implement the proposed unsigned binary multiplier. Karatsuba algorithm is best suited for higher bits and Urdhva-Tiryagbhyam algorithm is best for lower bit multiplication. A new algorithm by combining both helps to reduce the drawbacks of both. The multiplier is implemented using Verilog HDL, targeted on Spartan-3E and Virtex-4 FPGA.

Keywords: fpga, binary multiplier, unsigned, Vedic mathematics, Urdhva-Tiryagbhyam, Karatsuba

I. INTRODUCTION

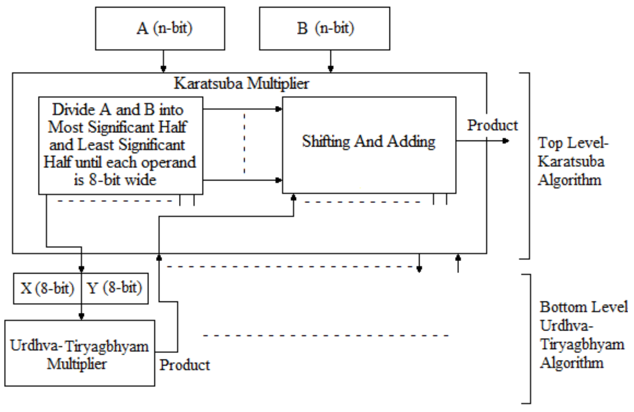
Binary multiplication is the most important operation in binary arithmetic. Applications such as High Power Computing (HPC), Image processing and Signal processing is based on binary multiplication. Area and delay requirements of conventional binary multipliers are much more and hence efficient multiplication algorithms are used to reduce delay and area requirement. Various multiplication algorithms are described in [1]. Vedic mathematics methods [2] are the most efficient in terms of delay and area. There are many literatures available on Vedic mathematics methods [3, 4, 5, 6, 7, 8 and 9]. Karatsuba algorithm [10, 11] is the most used algorithm for higher bit length multipliers. But every algorithm has its own advantages and disadvantages. In the proposed model, a combination of Urdhva-Tiryagbhyam algorithm (Vedic mathematics) and Karatsuba algorithm is used. By combining these two algorithms, best features of both the algorithms can be used in an efficient manner to reduce both delay and area.

II. KARATSUBA-URDHVA TIRYAGBHYAM BINARY MULTIPLIER

The proposed model, Karatsuba-Urdhva binary multiplier, uses a combination of Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm to make multiplication more efficient. Multiplication operation requires more time compared to addition. And as the number of bits increase, it consumes more area and time. In the design of floating point multiplier, unsigned binary multiplier is the important part. In double precision floating point format, we need a 53x53 bit multiplier and in single precision format we need 24x24 bit multiplier. It requires much time to perform these operations and it is the major contributor to the delay of the floating point multiplier. As the number of bits of operands increases, area and delay requirement increases drastically. The proposed model make the multiplication operation more area efficient and faster.

Karatsuba algorithm uses a divide and conquer approach where it breaks down the inputs into Most Significant half and Least Significant half and this process continues until the operands are of 8-bits wide. Karatsuba algorithm is best suited for operands of higher bit length. But at lower bit lengths, it is not as efficient as it is at higher bit lengths. To eliminate this problem, Urdhva Tiryagbhyam algorithm is used at the lower stages. The model of Urdhva-Tiryagbhyam algorithm is shown in Fig. 1.

Urdhva Tiryagbhyam algorithm is the best algorithm for binary multiplication in terms of area and delay. But as the number of bits increases, delay also increases as the partial products are added in a ripple manner. For example, for 4-bit multiplication, it requires 6 adders connected in a ripple manner. And 8-bit multiplication requires 14 adders and so on. Compensating the delay will cause increase in area. So Urdhva Tiryagbhyam algorithm is not that optimal if the number of bits is much more. If we use Karatsuba algorithm at higher stages and Urdhva Tiryagbhyam algorithm at lower stages, it can somewhat compensate the limitations in both the algorithms and hence the multiplier becomes more efficient. The circuit is further optimized by using carry select and carry save adders instead of ripple carry adders. This reduces the delay to a great extent with minimal increase in hardware. These two algorithms are explained in detail in the below sections.



A (n-bit)
B (n-bit)
Karatsuba Multiplier
Divide A and B into Most Significant Half and Least Significant Half until each operand is 8-bit wide
Shifting And Adding
Product
Top Level-Karatsuba Algorithm
X (8-bit)
Y (8-bit)
Urdhva-Tiryagbhyam Multiplier
Product
Bottom Level Urdhva-Tiryagbhyam Algorithm

Fig. 1 Karatsuba-Urdhva multiplier model

A. Urdhva Tiryagbhyam algorithm for multiplication

Urdhva-Tiryagbhyam sutra is an ancient Vedic mathematics method for multiplication [2, 3, 4 and 5]. It is a general formula applicable to all cases of multiplication. The formula is very short and consists of only one compound word and means 'Vertically and crosswise'. In Urdhva Tiryagbhyam algorithm, the number of steps required for multiplication can be reduced and hence the speed of multiplication is increased.

An illustration of steps for computing the product of two 4-bit numbers is shown below [6, 7]. The two input are $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$ and let the $p_7p_6p_5p_4p_3p_2p_1p_0$ be the product. And the temporary partial products are $t_0, t_1, t_2, \dots, t_6$. The partial products are obtained from the steps given below. The line notation of the steps is shown in Fig. 2.

Step1: $t_0(1bit) = a_0b_0$.
Step2: $t_1(2bit) = a_1b_0 + a_0b_1$.
Step3: $t_2(2bit) = a_2b_0 + a_1b_1 + a_0b_2$
Step4: $t_3(3bit) = a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3$.
Step5: $t_4(2bit) = a_3b_1 + a_2b_2 + a_1b_3$.
Step6: $t_5(2bit) = a_3b_2 + a_2b_3$.
Step7: $t_6(1bit) = a_3b_3$

The product is obtained by adding s_1, s_2 *and* s_3 as shown below, where s_1, s_2 *and* s_3 are the partial sum obtained.

$s_1 = t_6\, t_5[0]\, t_4[0]\, t_3[0]\, t_2[0]\, t_1[0]\, t_0$
$s_2 = t_5[1]\, t_4[1]\, t_3[1]\, t_2[1]\, t_1[1]$
$s_3 = t_3[2]$

$$\begin{array}{lcccccccc} \text{Product} = & t_6 & t_5[0] & t_4[0] & t_3[0] & t_2[0] & t_1[0] & t_0 & + \\ & & t_5[1] & t_4[1] & t_3[1] & t_2[1] & t_1[1] & 0 & + \\ & & & t_3[2] & 0 & 0 & 0 & 0 & \\ \hline p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 & \end{array}$$

This method can be further optimized to reduce the number of hardware. A more optimized hardware architecture [7, 9] is shown in Fig. 3. This model actually helps to eliminate the need for three operand 7-bit adder and hence reduces hardware

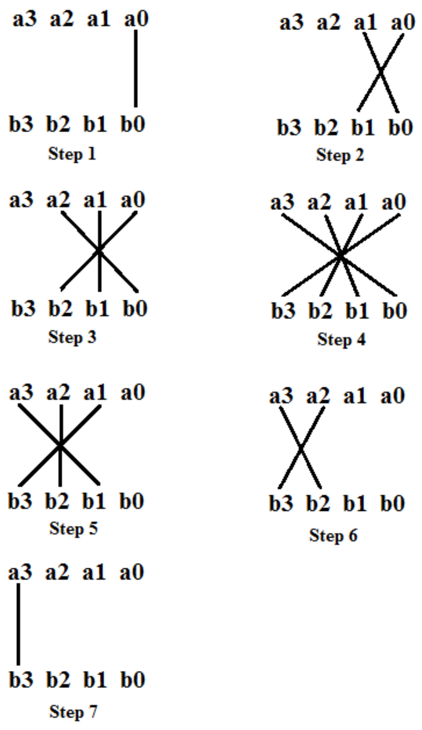


Fig. 2 Line notation of Urdhva Tiryagbhyam sutra

and delay. The adders are connected in ripple manner.

The expressions for product bits are as shown below.

$p_0 = a_0b_0$
$p_1 = LSB\ of\ (Sum(ADDER\ 1))$
$= LSB\ of\ (a_1b_0 + a_0b_1)$
$p_2 = LSB\ of\ (Sum(ADDER\ 2))$
$= LSB\ of\ (\text{MSB(ADDER1)}+a_2b_0 + a_1b_1 + a_0b_2)$
$p_3 = LSB\ of\ (Sum(ADDER\ 3))$
$= LSB\ of\ (\text{MSB(ADDER 2)}+a_3b_0 + a_2b_1 +$
a1b2+a0b3
$p_4 = LSB\ of\ (Sum(ADDER\ 4))$
$= LSB\ of\ (\text{MSB(ADDER1)}+a_3b_1 + a_2b_2 + a_1b_3)$
$p_5 = LSB\ of\ (Sum(ADDER\ 5))$
$= LSB\ of\ (\text{MSB(ADDER1)}+a_3b_2 + a_2b_3)$
$p_6 = LSB\ of\ (Sum(ADDER\ 6))$
$= LSB\ of\ (\text{MSB(ADDER1)}+a_3b_3)$
$p_7 = Carry\ of\ ADDER\ 6$

Since there are more than two operands in adders 2 to 5, we can use carry save addition to implement adders 2 to 5. This technique reduces the delay to a great extend compared to the ripple carry adder.

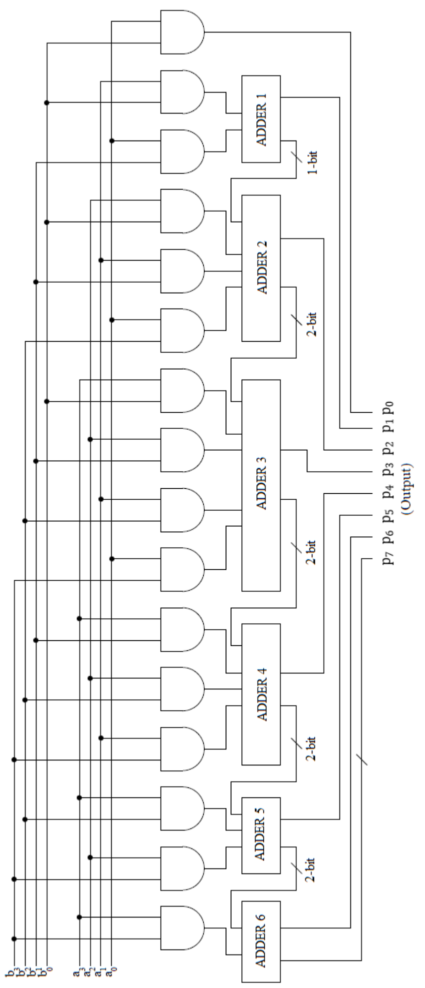


Fig. 3 Hardware architecture for 4x4 Urdhva Tiryagbhyam multiplier.

B. *Karatsuba Algorithm for multiplication*

Karatsuba multiplication algorithm [10, 11] is best suited for multiplying very large numbers. This method is discovered by Anatoli Karatsuba in 1962. It is a divide and conquer method, in which we divide the numbers into their Most Significant half and Least Significant half and then multiplication is performed. Karatsuba algorithm reduces the number of multipliers required by replacing multiplication operations by addition operations. Additions operations are faster than multiplications and hence the speed of multiplier is increased. As the number of bits of inputs increase, Karatsuba algorithm becomes more efficient. This algorithm is optimal if width of inputs is more than 16 bits. The hardware architecture of Karatsuba algorithm is shown in Fig. 4.

Product= $X.Y$

X and Y can be written as,

$$X = 2^{n/2}.\, \mathrm{X_l} + \mathrm{X_r} \qquad (1)$$

$$Y = 2^{n/2}.\, \mathrm{Y_l} + \mathrm{Y_r} \qquad (2)$$

Where $\mathrm{X_l}$, $\mathrm{Y_l}$ and $\mathrm{X_r}$, $\mathrm{Y_r}$ are the Most Significant half and Least Significant half of X and Y respectively, and n is the number of bits.

Then,

$$X.Y = \left(2^{\frac{n}{2}}.\, \mathrm{X_l} + \mathrm{X_r}\right).\left(2^{\frac{n}{2}}.\, \mathrm{Y_l} + \mathrm{Y_r}\right)$$
$$= 2^n.\, \mathrm{X_l\,Y_l} + 2^{n/2}\,(\mathrm{X_l\,Y_r} + \mathrm{X_r\,Y_l}) + \mathrm{X_r\,Y_r} \qquad (3)$$

The Second term in equation (3) can be optimized to reduce the number of multiplication operations.

i.e.; $\mathrm{X_l\,Y_r} + \mathrm{X_r\,Y_l} = (\mathrm{X_l} + \mathrm{X_r})(\mathrm{Y_l} + \mathrm{Y_r}) - \mathrm{X_l\,Y_l} - \mathrm{X_r\,Y_r}$ (4)

The equation (3) can be re-written as,

$$X.Y = 2^n.\, \mathrm{X_l\,Y_l} + \mathrm{X_r\,Y_r} + 2^{\frac{n}{2}}\,((\mathrm{X_l} + \mathrm{X_r})(\mathrm{Y_l} + \mathrm{Y_r}) - \mathrm{X_l\,Y_l} - \mathrm{X_r\,Y_r}) \qquad (5)$$

The recurrence of Karatsuba algorithm is,

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) \quad O(n^{1.585})$$

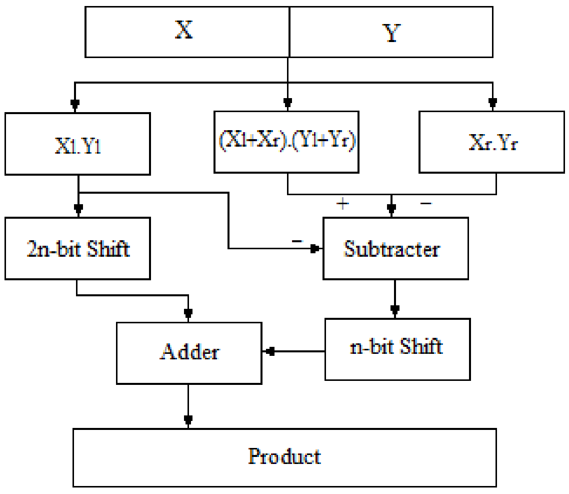


Fig. 4 Karatsuba multiplier

III. Implimentation and Results

The main objective of this paper is to design and implement an unsigned multiplier which must be efficient in its operation both in terms of delay and area. Since multiplication is the most important and complex operation in binary arithmetic, we designed a multiplier which can operate at high speed and increase in delay and area is significantly less with increase in number of bits. Proposed unsigned binary multiplier is implemented using Verilog HDL and tested. The multiplier units are further optimized by replacing simple adders with efficient adders like carry select adders and carry save adders. The model is synthesized and simulated using Xilinx Synthesis Tools (ISE 14.7) targeted on Saprtan-3E and Virtex-4 fpga. The summary of results on Virtex-4 fpga is given in table I.

Comparison with various multiplier units is given in tables II, III, IV and V

TABLE I
Performance analysis of Karatsuba-Urdhva multipliers

	8-bit multiplier	16-bit multiplier	24-bit multiplier	32-bit multiplier
Slices	113	410	972	1389
LUTs	120	451	1018	1545
IOBs	33	65	97	129
Delay	9.396ns	11.514ns	12.996ns	13.141ns
f_{max} (MHz)	274.469	248.964	226.508	209.606
Logic levels	14	22	31	39

TABLE II
Delay comparison of various 8-bit multipliers with proposed Karatsuba-Urdhva multiplier

	Ref. [6]	Ref. [7]	Ref. [8]	Proposed multiplier
Width	8-bit	8-bit	8-bit	8-bit
Delay	28.27ns	15.050ns	23.973ns	9.396ns

TABLE III
Delay comparison of various 16-bit multipliers with proposed Karatsuba-Urdhva multiplier

	Ref. [12]-vedic multiplier	Ref. [5]	Proposed multiplier
Width	16-bit	16-bit	16-bit
Delay	13.452ns	27.148ns	11.514ns

TABLE IV
Delay and area comparison of 24-bit multipliers with proposed Karatsuba-Urdhva multiplier

	Slices	LUTs	Delay
Ref. [13]	1306	2329	16.316ns
Proposed multiplier	972	1018	12.996ns

IV. Conclusion and Future Work

This paper shows how to effectively reduce the percentage increase in delay and area of a multiplier by using a very efficient combination of Karatsuba and Urdhva-Tiryagbhyam algorithms. The model can be further optimized in terms of delay by using pipelining methods and also by using efficient adders instead of ripple adders.

TABLE V
Delay and area comparison of 32-bit multipliers with proposed Karatsuba-Urdhva multiplier

	LUTs	Delay
Ref. [12]- Modified Booth multiplier (Radix-8)	2721	12.081ns
Ref. [12]- Modified Booth multiplier (Radix-16)	7161	11.564ns
Ref. [12]	2704	9.536ns
Proposed multiplier	1545	13.141ns

References

[1] Computer Arithmetic, Behrooz Parhami, Oxford University Press, 2000.

[2] “Vedic mathematics”, Swami Sri Bharati Krsna Thirthaji Maharaja, Motilal Banarasidass Indological publishers and Book sellers, 1965

[3] R. Sridevi, Anirudh Palakurthi, Akhila Sadhula, Hafsa Mahreen, “Design of a High Speed Multiplier (Ancient Vedic Mathematics Approach)”, International Journal of Engineering Research (ISSN : 2319-6890), Volume No.2, Issue No.3, pp : 183-186, July 2013

[4] Nivedita A. Pande, Vaishali Niranjane, Anagha V. Choudhari, “Vedic Mathematics for Fast Multiplication in DSP”, International Journal of Engineering and Innovative Technology (IJEIT), Volume 2, Issue 8, pp. 245-247, February 2013

[5] R.K. Bathija, R.S. Meena, S. Sarkar, Rajesh Sahu, “Low Power High Speed 16x16 bit Multiplier using Vedic Mathematics”, International Journal of Computer Applications (0975 – 8887), Volume 59– No.6, pp. 41-44, December 2012

[6] Poornima M, Shivaraj Kumar Patil, Shivukumar , Shridhar K P , Sanjay H, “Implementation of Multiplier using Vedic Algorithm”, International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-2, Issue-6, pp. 219-223, May 2013

[7] Premananda B.S., Samarth S. Pai, Shashank B., Shashank S. Bhat, “Design and Implementation of 8-Bit Vedic Multiplier”, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 2, Issue 12, pp. 5877-5882, December 2013

[8] R. Sai Siva Teja, A. Madhusudhan, "FPGA Implementation of Low-Area Floating Point Multiplier Using Vedic Mathematics", International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 3, Issue 12, pp. 362-366, December 2013.

[9] Harpreet Singh Dhillon, Abhijit Mitra, "A Reduced-Bit Multiplication Algorithm for Digital Arithmetic", World Academy of Science, Engineering and Technology, Vol 19, pp. 719-724, 2008

[10] N.Anane, H.Bessalah, M.Issad, K.Messaoudi, "Hardware implementation of Variable Precision Multiplication on FPGA", 4th International Conference on Design & Technology of Integrated Systems in Nanoscale Era, pp. 77-81, 2009

[11] Anand Mehta, C. B. Bidhul, Sajeevan Joseph, Jayakrishnan. P, "Implementation of Single Precision Floating Point Multiplier using Karatsuba Algorithm", 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), pp. 254-256, 2013

[12] Jagadeshwar Rao M, Sanjay Dubey, "A High Speed and Area Efficient Booth Recoded Wallace Tree Multiplier for fast Arithmetic Circuits", 2012 Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PRIMEASIA), pp. 220-223, 2012.

[13] Anna Jain, Baisakhy Dash, Ajit Kumar Panda, Muchharla Suresh, "FPGA Design of a Fast 32-bit Floating Point Multiplier Unit", International Conference on Devices, Circuits and Systems (ICDCS), pp. 545-547, 2012.