



OPTIMIZED NEURAL ARCHITECTURE 32 BIT VEDIC MULTIPLIER

¹KAKARLA LAKSHMI PRIYANKA, ²T.RAGHAVENDRA VISHNU M.TECH,PHD

¹STUDENT, ECE DEPARTMENT, PRIYADARSHINI INSTITUTE OF TECHNOLOGY & SCIENCE
FOR WOMEN

²ASSISTANCE PROFESSOR, PRIYADARSHINI INSTITUTE OF TECHNOLOGY & SCIENCE FOR
WOMEN

Abstract—The proposed research work specifies the modified version of binary vedic multiplier using vedic sutras of ancient vedic mathematics. It provides modification in preliminarily implemented vedic multiplier. The modified binary vedic multiplier is preferable has shown improvement in the terms of the time delay and also device utilization. The proposed technique was designed and implemented in Verilog HDL. For HDL simulation, modelsim tool is used and for circuit synthesis, Xilinx is used. The simulation has been done for 4 bit, 8 bit, 16 bit, 32 bit multiplication operation. Only for 32 bit binary vedic multiplier technique the simulation results are shown. This modified multiplication technique is extended for larger sizes. The outcomes of this multiplication technique is compared with existing vedic multiplier techniques.

1. INTRODUCTION

1.1 Overview:

BINARY multipliers are a widely used building block element in the design of microprocessors and embedded systems, and therefore, they are an important target for implementation optimization. Current implementations of binary multiplication follow the steps of :

- 1) recoding of the multiplier in digits in a certain number system;
- 2) digit multiplication of each digit by the multiplicand, resulting in a certain number of partial products;
- 3) reduction of the partial product array to two operands using multioperand addition techniques;

4) carry-propagate addition of the two operands to obtain the final result. The recoding type is a key issue, since it determines the number of partial products.

The usual recoding process recodes a binary operand into a signed-digit operand with digits in a minimally redundant digit set. Specifically, for radix- r ($r = 2^m$), the binary operand is composed of nonredundant radix- r digits (by just making groups of m bits), and these are recoded from the set $\{0, 1, \dots, r-1\}$ to the set $\{-r/2, \dots, -1, 0, 1, \dots, r/2\}$ to reduce the complexity of digit multiplications. For n -bit operands, a total of n/m partial products are generated for two's complement representation, and $(n+1)/m$ for unsigned representation. Radix-4 modified Booth is a widely used recoding method, that recodes a



binary operand into radix-4 signed digits in the set $\{-2, -1, 0, 1, 2\}$. This is a popular recoding since the digit multiplication step to generate the partial products only requires simple shifts and complementation.

The resulting number of partial products is about $n/2$. Higher radix signed recoding is less popular because the generation of the partial products requires odd multiples of the multiplicand which can not be achieved by means of simple shifts, but require carry-propagate additions. For instance, for radix-4 signed digit recoding [9] the digit set is $\{-8, -7, \dots, 0, \dots, 7, 8\}$, so that some odd multiples of the multiplicand have to be generated. Specifically, it is required to generate $\times 3$, $\times 5$, and $\times 7$ multiples ($\times 6$ is obtained by simple shift of $\times 3$). The generation of each of these odd multiples requires a two term addition or subtraction, yielding a total of three carry-propagate additions.

1.2 LITERATURE SURVEY:

However, the advantage of the high radix is that the number of partial products is further reduced. For instance, for radix-4 and n -bit operands, about $n/4$ partial products are generated. Although less popular than radix-4, there exist industrial instances of radix-8. and radix-4 multiplier in microprocessors implementations. The choice of these radices is related to area/delay/power optimization of pipelined multipliers (or fused multiplier adder as in the case of a Intel Itanium microprocessor), for balancing delay between stages and/or reduce the number of pipelining flip-flops.

A further consideration is that carry-propagate adders are today highly energy-

delay optimized, while partial product reductions trees suffer the increasingly serious problems related to a complex wiring and glitching due to unbalanced signal paths. It is recognized in the literature that a radix-8 recoding leads to lower power multipliers compared to radix-4 recoding at the cost of higher latency (as a combinational block, without considering pipelining). Moreover, although the radix-4 multiplier requires the generation of more odd multiples and has a more complex wiring for the generation of partial products, a recent microprocessor design considered it to be the best choice for low power (under the specific constraints for this microprocessor).

In some optimizations for radix-4 two's complement multipliers were introduced. Although for n -bit operands, a total of $n/2$ partial products are generated, the resulting maximum height of the partial product array is $n/2 + 1$ elements to be added (in just one of the columns). This extra height by a single-bit row is due to the $+1$ introduced in the bit array to make the two's complement of the most significant partial product (when the recoded most significant digit of the multiplier is negative). The maximum column height may determine the delay and complexity of the reduction tree, authors showed that this extra column of one bit could be assimilated (with just a simplified three bit addition) with the most significant part of the first partial product without increasing the critical path of the recoding and partial product generation stage. The result is that the partial product array has a maximum height of $n/2$. This reduction of one bit in the maximum height might be of interest

for high-performance short-bit width two's complement multipliers (small n) with tight cycle time constraints, that are very common in SIMD digital signal processing applications. Moreover, if n is a power of two, the optimization allows to use only 4-2 carry-save adders for the reduction tree, potentially leading to regular layouts. These kind of optimizations can become particularly important as they may add flexibility to the "optimal" design of the pipelined multiplier.

2. LITERATURE SURVEY

2.1 EXISTING METHODS-MULTIPLERS:

2.1.1 MULTIPLERS

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets

1. High speed,
2. Low power consumption,
3. Regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed,
4. Low power and compact VLSI implementation.

The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be

added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand "serial-parallel" multipliers compromise speed to achieve better performance for area and power consumption. The selection of a parallel or serial multiplier actually depends on the nature of application. In this lecture we introduce the multiplication algorithms and architecture and compare them in terms of speed, area, power and combination of these metrics. AND gates are used to generate the Partial Products (PP). If the multiplicand is N -bits and the Multiplier is M -bits then there is $N * M$ partial product.

2.1.2 HISTORY OF MULTIPLIERS

The early computer systems had what are known as Multiply and Accumulate units to perform multiplication between two binary unsigned numbers. The Multiply and Accumulate unit was the simplest implementation of a multiplier. The basic block diagram of such a system is given below.

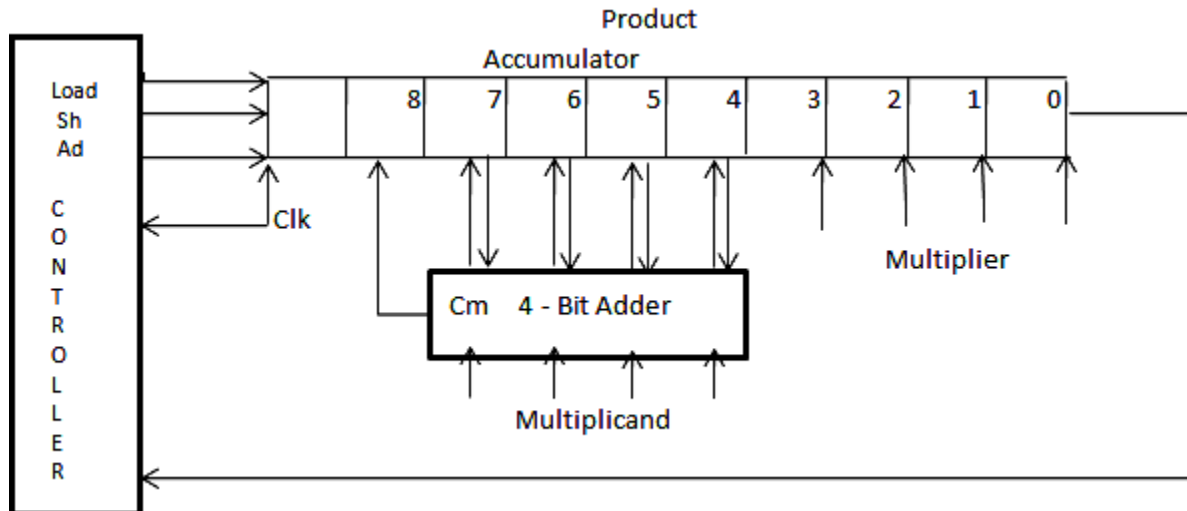


Fig.2.1 Multiplier Block Diagram

2.1.3 IMPLEMENTATION

The MAC unit requires a 4-bit multiplicand register, 4-bit multiplier register, a 4-bit full adder and an 8-bit accumulator to hold the product. In the figure above the product register holds the 8-bit result. In a typical binary multiplication, based on the multiplier bit being processed, either zero or the multiplicand is shifted and then added.

Following the same process would require an 8-bit adder. Instead, in the above design the contents of the product register are shifted right by one position and the multiplicand is added 5 to the contents. This multiply and accumulate block is also known by the name serial-parallel multiplier as the multiplier bits are processed serially but the addition takes place in parallel. The second type of multiplier is the parallel array multiplier.

The desire to speed up the rate at which the output is generated resulted in the development of this category of multiplier. In a serial-parallel multiplier discussed above, it takes one clock cycle to process one bit of the data input at any given time. Therefore, when working on an N-bit input it would take at least N clock cycles to generate the final output. In a parallel array multiplier the result is obtained as soon as inputs are presented to the multiplier. This is mainly because of the use of AND array structure to compute the partial product terms. Once the partial product terms are generated the only delay in generating the output is contributed by the adders which sum the partial product terms column wise to generate the result.



3. VEDIC MULTIPLICATION ALGORITHMS

3.1 HISTORY OF VEDIC MATHEMATICS:-

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya-Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. It covers explanation of several modern mathematical terms including arithmetic, geometry (plane, co-ordinate), trigonometry, quadratic equations, factorization and even calculus. His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swamiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. Obviously these formulae are not to be found in present text of Atharva Veda because these formulae were constructed by Swamiji himself. Vedic mathematics is not only a mathematical wonder but also it is logical. That's why VM has such a degree of eminence which cannot be disapproved. Due these phenomenal characteristic, VM has already crossed the boundaries of India and has become a leading topic of research abroad. VM deals with several basic as well as complex mathematical operations. Especially, methods of basic arithmetic are extremely simple and powerful.

3.2 ALGORITHMS OF VEDIC MATHEMATICS:-

3.2.1 VEDIC MULTIPLICATION

The proposed Vedic multiplier is based on the Vedic multiplication formulae (Sutras). These Sutras have been traditionally used for the multiplication of two numbers in the decimal number system. In this work, we apply the same ideas to the binary number system to make the proposed algorithm compatible with the digital hardware. Vedic multiplication based on some algorithms, some are discussed below:

3.2.1.1 Urdhva Tiryakbhyam sutra

The multiplier is based on an algorithm Urdhva Tiryakbhyam (Vertical & Crosswise) of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and crosswise". It is based on a novel concept through which the generation of all partial products can be done with the concurrent addition of these partial products. The parallelism in generation of partial products and their summation is obtained using Urdhva Tiryakbhyam explained in fig 2.1. The algorithm can be generalized for $n \times n$ bit number. Since the partial products and their sums are calculated in parallel, the multiplier is independent of the clock frequency of the processor. Thus the multiplier will require the same amount of time to calculate the product and hence is independent of the clock frequency. The net advantage is that it reduces the need of microprocessors to operate at increasingly

high clock frequencies. While a higher clock frequency generally results in increased processing power, its disadvantage is that it also increases power dissipation which results in higher device operating temperatures. By adopting the Vedic multiplier, microprocessors designers can easily circumvent these problems to avoid catastrophic device failures. The processing power of multiplier can easily be increased by increasing the input and output data bus widths since it has a quite a regular

structure. Due to its regular structure, it can be easily layout in a silicon chip. The Multiplier has the advantage that as the number of bits increases, gate delay and area increases very slowly as compared to other multipliers. Therefore it is time, space and power efficient. It is demonstrated that this architecture is quite efficient in terms of silicon area/speed.

4. EXISTING METHODS

4.1 Vedic Wallace Multiplier

In general, Wallace tree addition uses full adders to extensively reduce the partial products.

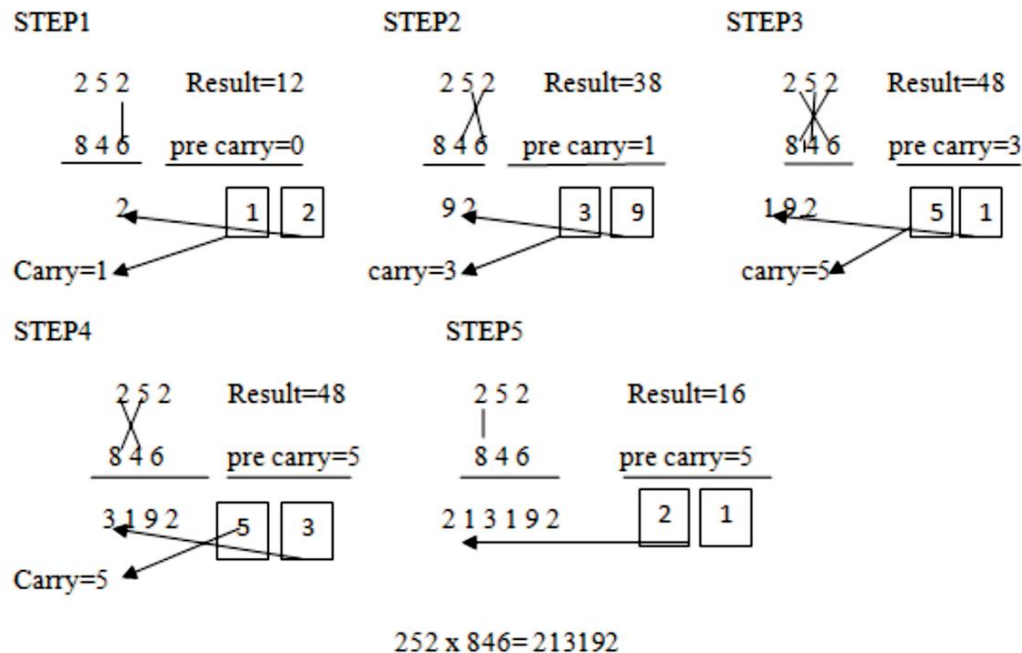


Figure 3.1 Multiplication of Two Decimal Numbers: 252x846

When the critical path is compared between the critical path in 4 bit conventional and Vedic multiplier, for a 4-bit multiplier, 4 partial products will be generated, as shown

in Figure 2 and are named as p0 to p3. For Wallace tree multiplier, a 3:2 reduction is used, so that the partial products are reduced from 4 to 3. The Delay in critical path is

given by the addition of 3 full adder sums, 2 full adder carry, and half adder carry. The critical path for Vedic mathematics as shown in Figure 3, is given by 2FAS is reduced by 3HAS and in terms of XOR gates, Vedic-Wallace uses 3XOR gates

instead of 4XOR i.e., less carry propagation delay than the conventional method. Hence, Vedic-Wallace has a variable improvement over design were depending upon the number of bits in multiplication

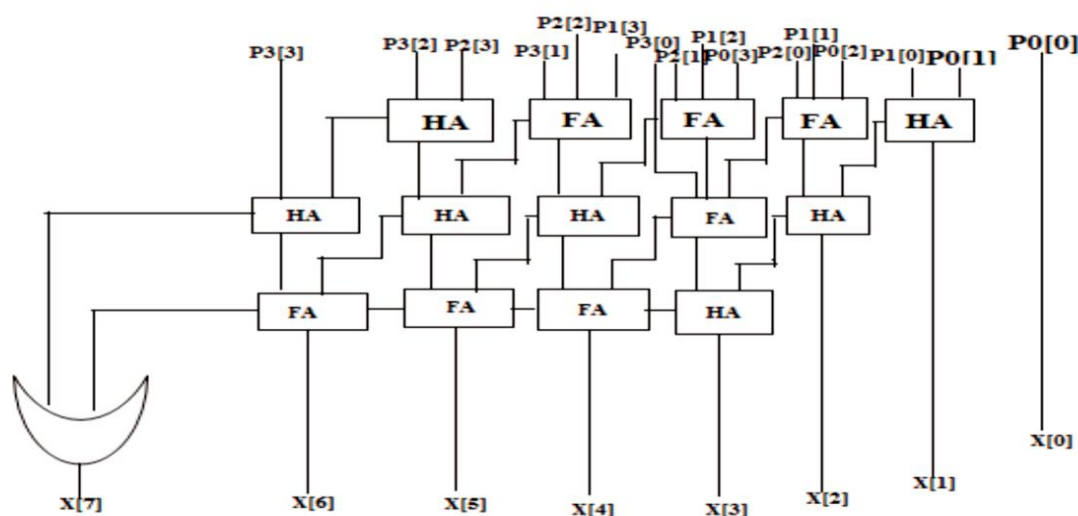


Figure 3.2 4x4 Multiplier using Wallace Tree

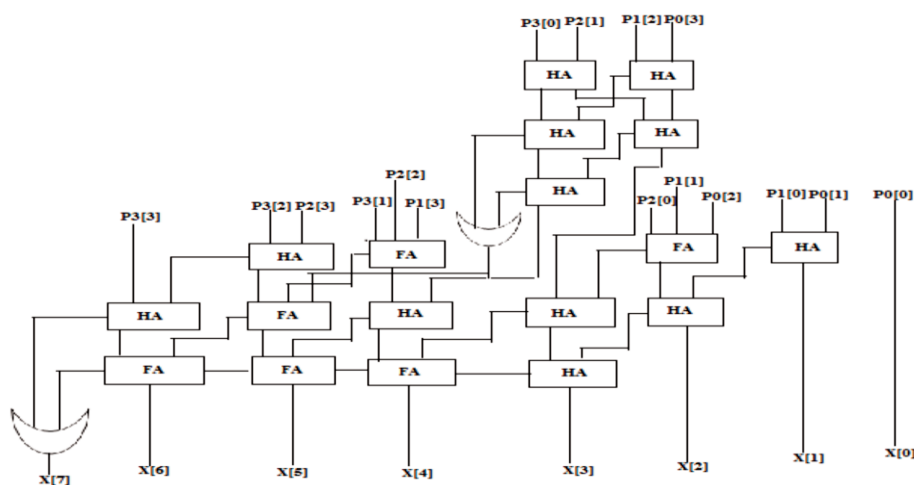


Figure 3. 4x4 Multiplier using Vedic Reduction

5. PROPOSED METHOD

5.1 . INTRODUCTION

Any proposed system must be efficient in terms of power, speed and size as per growing technology. In early days Vedic mathematics is based on 16 vedic sutras. By using Vedic methods the mathematical operations are fast and the processing speed to perform the operations can be improved. There has been many existing binary multipliers which are efficient.

II. MULTIPLIER

A binary multiplier [3] can be used in digital electronics as a electronic circuit, such as in computers to find the product of two binary numbers. Carbon-copy of normal multiplication technique is used by binary multiplier, the multiplicand is multiplied with each bit of the multiplier beginning from the least significant bit. Two half adder(HA) modules can be used in order to implement a 2-bit binary multiplier. A no of computer arithmetic calculations can be used to appliance digital multiplier. Among these techniques many imply computing a set of partial products, and then summing the generated partial products together. Fig. 1, shows 2x2 binary multiplier.

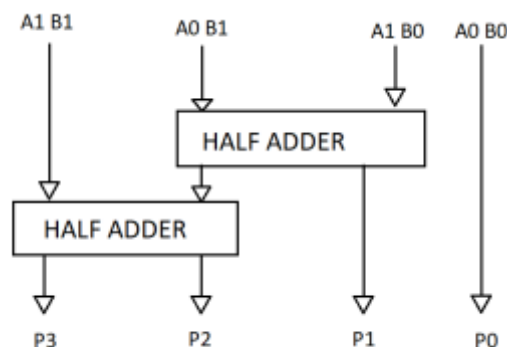


Fig. 1. 2x2 Binary Multiplier

A. Ripple Carry Adder(RCA)

In a multiplier number of Full adders are arranged in a manner to give the results of an addition operation of n-bit binary sequence. The input to next Full adder stage is obtained from the previous carry output of adder, it repeats until it reaches to the ending stage. Fig. 2 shows Four bit(RCA) Ripple Carry Adder [4].

III. VEDIC MULTIPLIER

The mode used by Vedic multiplier [6] is Vedic mathematics. By using this technique it will increase, and consumes fewer hardware elements. The sutra [6] used by Vedic multiplier is Urdhva Tiryakbhyam[3] which means Vertically as well as Crosswise. The Fig. 3 shows block diagram of 32 bit vedic multiplier circuit. The 2 input bits are separated into 2 similar parts the vertical and cross product calculations can be done as shown in Fig. 3, with inputs A[31:0] and B[31:0]. As shown in the Fig. 3, the 2 adders are used in the design of intermediate stages of the addition. The output carry Cout from these

two adders is given as input to another RCA. If bits are not of equal sizes concatenate them. For 32-bit Modified Vedic multiplier the outputs of parallel adder is given to OR gate and of the size of last RCA is reduced to half. Fig. 3, shows 32-bit Vedic multiplier.

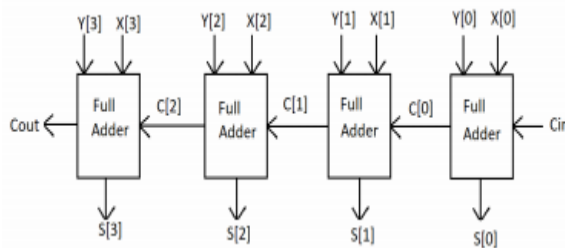


Fig. 2. 4-Bit Ripple Carry Adder

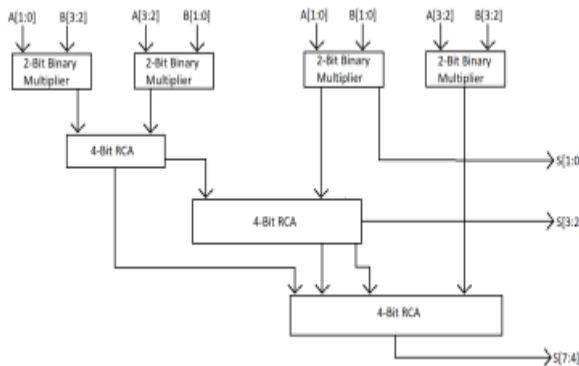


Fig. 3. 4-Bit Vedic Multiplier

IV. MODIFIED VEDIC MULTIPLIER

In the proposed paper, the two parallel adders are replaced by CSA [4] for the better execution of the multiplier architecture. The recommended modified Vedic multiplication methodology is done in the following for 4 bit inputs, A(A3 -A0) and B(B3 -B0) and 8 bit output S (S7 -S0).

$$\begin{array}{r}
 A_3A_2A_1A_0 \\
 \times B_3B_2B_1B_0 \\
 \hline
 (A_3A_2) \times (B_1B_0) \quad (A_1A_0) \times (B_1B_0) \\
 (A_3A_2) \times (B_3B_2) \quad (A_1A_0) \times (B_3B_2) \\
 \hline
 S_7 \ S_6 \quad S_5 \ S_4 \ S_3 \ S_2 \quad S_1 \ S_0
 \end{array}$$

A multiplier of 2 bit is used to calculate intermediate stage results, and the output is 4 bits. (A3A2)(B3B2) using 2 bit multiplier generates result: S3S3S2S3S30 (A3A2)(B1B0) using 2 bit multiplier generates result: S2S3S2S2S1S20 (A1A0)(B3B2) using 2 bit multiplier generates result: S1S3S1S2S1S10 (A1A0)(B1B0) using 2 bit multiplier generates result: S0S3S0S2S0S1S00

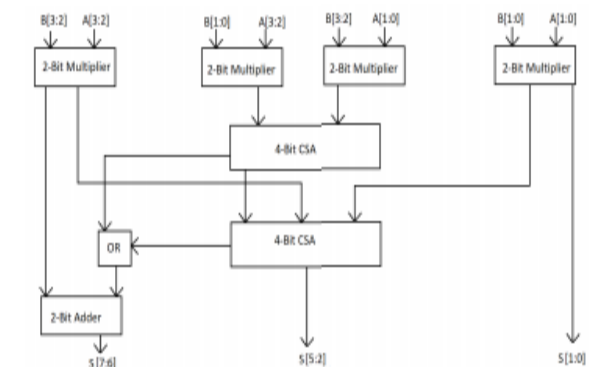


Fig. 4. Modified 4 Bit Vedic Multiplier

The 4 bit CSA Carry Save Adder [4] is used to add three 4 bit data inputs: S2S3S2S2S1S20, S1S3S1S2S1S10 and S3S1S30 S03 S02. The proposed 4 bit modified Vedic multiplier [7] is designed and the Fig. 4 shows it. The last two MSBs of CSA outputs are given as inputs to OR gate. In addition, the last stage 4 bit RCA is replaced by 2 bit adder circuit through which the output value of OR gate can be controlled. One of the input to last stage 2-

bit adder[6] is obtaining from the output of or gate.

SIMULATION RESULTS

First, a 4-bit Vedic multiplier is designed and in the same manner the size of Vedic multiplier is increased up to 32-bit i.e, 8-bit, 16-bit, and then 32-bit using RCA and then by using CSA [4][8], the modified 4-bit PROPOSED METHOD RESULTS

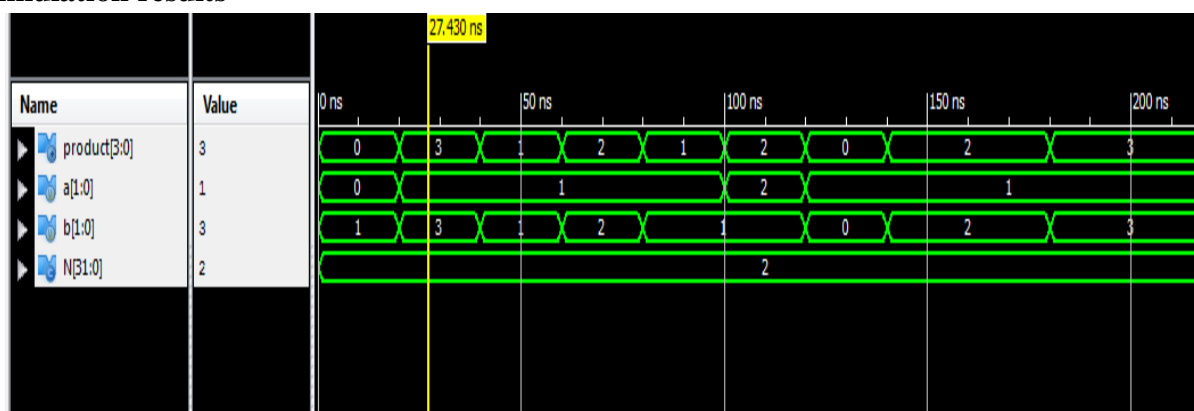
Vedic multiplier is implemented and in the same way the size of the modified vedic multiplier is increased upto 32-bit i.e, 8, 16, and 32- bit[7]. For the functionality verification it was done usingMODELSIM and the final synthesis is done by using XILINX ISE DESIGN SUITE.And also the working of the modelsim process is shown in the following:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	30	204000	0%
Number of fully used LUT-FF pairs	0	30	0%
Number of bonded IOBs	16	600	2%

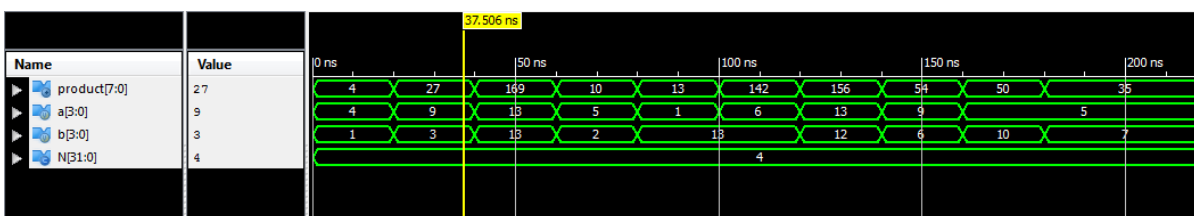
4x4

EXTENSION METHOD RESULTS

Simulation results



2x2





International Journal For Advanced Research In Science & Technology

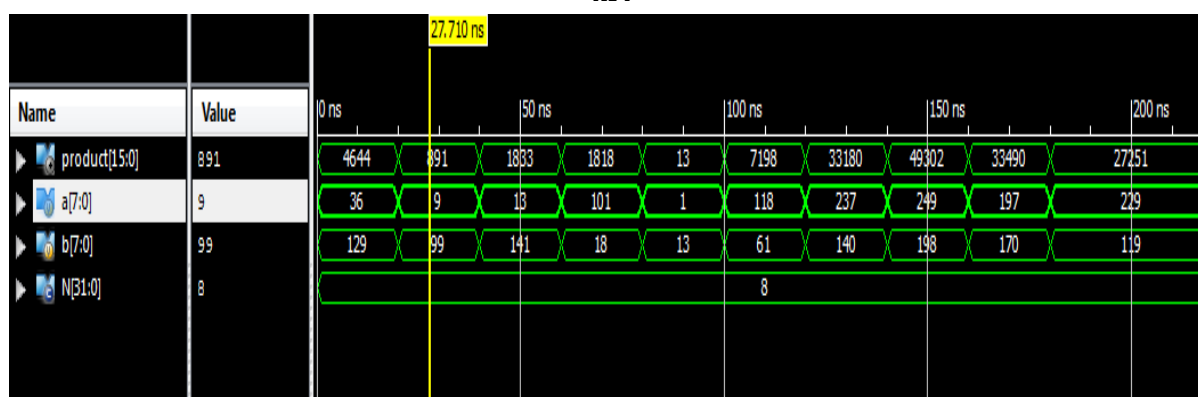
A peer reviewed international journal

www.ijarst.in

IJARST

ISSN: 2457-0362

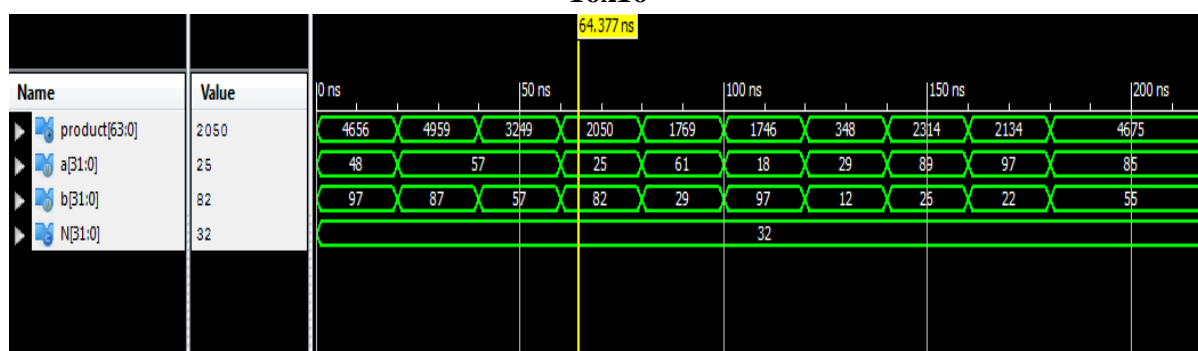
4x4



8x8



16x16



32x32

Design Summaries




International Journal For Advanced Research In Science & Technology

A peer reviewed international journal


www.ijarst.in

IJARST


ISSN: 2457-0362

Device Utilization Summary (estimated values)				
Logic Utilization	Used	Available	Utilization	
Number of Slice LUTs	4	204000	0%	
Number of fully used LUT-FF pairs	0	4	0%	
Number of bonded IOBs	8	600	1%	


2x2

Device Utilization Summary (estimated values)				
Logic Utilization	Used	Available	Utilization	
Number of Slice LUTs	24	204000	0%	
Number of fully used LUT-FF pairs	0	24	0%	
Number of bonded IOBs	16	600	2%	

4x4

Device Utilization Summary (estimated values)				
Logic Utilization	Used	Available	Utilization	
Number of Slice LUTs	124	204000	0%	
Number of fully used LUT-FF pairs	0	124	0%	
Number of bonded IOBs	32	600	5%	

8x8

Device Utilization Summary (estimated values)				
Logic Utilization	Used	Available	Utilization	
Number of Slice LUTs	554	204000	0%	
Number of fully used LUT-FF pairs	0	554	0%	
Number of bonded IOBs	64	600	10%	

16x16




International Journal For Advanced Research In Science & Technology

A peer reviewed international journal

www.ijarst.in

IJARST

ISSN: 2457-0362

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	2346	204000	1%
Number of fully used LUT-FF pairs	0	2346	0%
Number of bonded IOBs	128	600	21%

32x32

Time summaries

Timing constraint: Default path analysis

Total number of paths / destination ports: 14 / 4

Delay: 0.793ns (Levels of Logic = 3)

Source: a<0> (PAD)

Destination: product<3> (PAD)

Data Path: a<0> to product<3>

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	4	0.000	0.470	a_0_IBUF (a_0_IBUF)
LUT4:I0->O	1	0.043	0.279	HH2/cout1 (product_3_OBUF)
OBUF:I->O		0.000		product_3_OBUF (product<3>)
Total		0.793ns (0.043ns logic, 0.750ns route) (5.4% logic, 94.6% route)		

2x2



Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)
IBUF:I->O	6	0.000	0.479	a_3_IBUF (a_3_IBUF)
LUT4:I0->O	2	0.043	0.546	VV3/HH2/Mxor_s_xo<0>1 (product3<2>)
LUT5:I0->O	3	0.043	0.466	RC1/F3/Mxor_sum_xo<0>1 (RC1/sum1<2>)
LUT4:I0->O	4	0.043	0.442	RC1/F22/cout1 (RC1/cout_t1<1>)
LUT6:I3->O	2	0.043	0.461	RC2/F4/Mxor_sum_xo<0>1 (RC2/sum1<3>)
LUT5:I1->O	2	0.043	0.347	RC2/F23/cout1 (RC2/cout_t1<2>)
LUT6:I4->O	1	0.043	0.550	OO<1>1 (OO<1>)
LUT6:I0->O	1	0.043	0.279	RC3/F2/Mxor_sum_xo<0>1 (product_7_OBUF)
OBUF:I->O		0.000		product_7_OBUF (product<7>)
<hr/>				
Total		3.873ns (0.301ns logic, 3.572ns route)		
		(7.8% logic, 92.2% route)		

4x4

CONCLUSION

This work has presented a systematic method for binary multiplier[1] circuits which is based on Vedic mathematics. When it comes to the terms of time delay then the proposed system is more efficient than existing methods. Elongation for a higher bit size can be done with help of proposed technique. Moreover, adders of different architectures[5] can be used in the CSA Carry Save Adder design used in the proposed modified Vedic multiplier. Among many techniques modified architecture is used to increase and speed up the multiplication. In this technique hike in area occurred it is a drawback.

REFERENCES

- [1] I. Blake, G. Seroussi, and N.P. Smart, Elliptic Curves in Cryptography, ser. London Mathematical Society Lecture Note Series.. Cambridge, U.K.: Cambridge Univ. Press, 1999.
- [2] N. R. Murthy and M. N. S. Swamy, "Cryptographic applications of

brahmaqupta-bha skara equation," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 7, pp. 1565–1571, 2006.

- [3] L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," J. VLSI Digit. Process., vol. 19, pp. 149–C166, 1998.
- [4] P. K. Meher, "On efficient implementation of accumulation in finite field over $GF(2^m)$ and its applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 541–550, 2009.
- [5] L. Song, K. K. Parhi, I. Kuroda, and T. Nishitani, "Hardware/software codesign of finite field datapath for low-energy Reed-Solomon codecs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 8, no. 2, pp. 160–172, Apr. 2000.
- [6] G. Drolet, "A new representation of elements of finite fields $GF(2^m)$ yielding small complexity arithmetic circuits," IEEE Trans. Comput., vol. 47, no. 9, pp. 938–946, 1998.



- [7] C.-Y. Lee, J.-S. Horng, I.-C. Jou, and E.-H. Lu, "Low-complexity bit-parallel systolic montgomery multipliers for special classes of $GF(2^m)$," IEEE Trans. Comput., vol. 54, no. 9, pp. 1061–1070, Sep. 2005.
- [8] P. K. Meher, "Systolic and super-systolic multipliers for finite field $GF(2^m)$ based on irreducible trinomials," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 4, pp. 1031–1040, May 2008.
- [9] J. Xie, J. He, and P. K. Meher, "Low latency systolic montgomery multiplier for finite field $GF(2^m)$ based on pentanomials," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 2, pp. 385–389, Feb. 2013.
- [10] H. Wu, M. A. Hasan, I. F. Blake, and S. Gao, "Finite field multiplier using redundant representation," IEEE Trans. Comput., vol. 51, no. 11, pp. 1306–1316, Nov. 2002.
- [11] A. H. Namin, H. Wu, and M. Ahmadi, "Comb architectures for finite field multiplication in F_{2^m} ," IEEE Trans. Comput., vol. 56, no. 7, pp. 909–916, Jul. 2007.
- [12] A. H. Namin, H. Wu, and M. Ahmadi, "A new finite field multiplier using redundant representation," IEEE Trans. Comput., vol. 57, no. 5, pp. 716–720, May 2008.
- [13] A. H. Namin, H. Wu, and M. Ahmadi, "A high-speed word level finite field multiplier in F_{2^m} using redundant representation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 10, pp. 1546–1550, Oct. 2009.
- [14] A. H. Namin, H. Wu, and M. Ahmadi, "An efficient finite field multiplier using redundant representation," ACM Trans. Embedded Comput. Sys., vol. 11, no. 2, Jul. 2012, Art. 31.
- [15] North Carolina State University, 45 nm FreePDK wiki [Online]. Available: <http://www.eda.ncsu.edu/wiki/FreePDK45:Manual>