International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 8, August 2015)

Design of Low Power and High Speed Modified Carry Select Adder for 16 bit Vedic Multiplier

Yerra Manohar
M.Tech Student (VLSI-SD)
Dept.of.ECE, GNIT
manoharyarra@gmail.com

Mr. SK Saidulu,
Associate Professor
Dept.of.ECE, GNIT
sk.saidulu@gmail.com

Mrs.A Swetha
Associate Professor
Dept.of.ECE, GNIT
swethavlsi@yahoo.com

Prof B Kedarnath
HOD-ECE
Dept.of.ECE, GNIT
hodece.gnit@gniindia.org

Abstract: **In this paper the high speed and low power 16×16 Vedic Multiplier fundamental block is designed by using low power and high speed modified carry select adder. Modified Carry Select Adder employs a newly incremented circuit in the Carry Select Adder (CSA) which is known to be the fastest adder among the conventional adder structures. Multiplication namely Vedic multiplication has been introduced which is quite different from normal multiplication by shift and addition operations. Normally a multiplier is a key block major power dissipation source. This paper presents a new design methodology and less power efficient Vedic Multiplier based up on ancient Vedic Mathematic techniques. This paper presents a technique for N×N multiplication is implemented and gives very less delay for calculating area efficient Vedic multiplier based on the crosswise and vertical algorithm. Comparisons with existing conventional fast adder architectures have been made to prove its efficiency. The performance analysis shows that the proposed architecture achieves three fold advantages in terms of delay-area-power. The synthesis results of the Vedic multiplier has compared with the booth, array multiplier by different technologies.**

Keywords: **CSA, Multiplier, Vedic multiplier, LP-gate, high delay block.**

I. NTRODUCTION

Multiplication is one of the fundamental block in almost all the arithmetic logic units. This Vedic multiplication is mainly used in the fields of the Digital Signal Processing (DSP) and also in so many applications like Fast Fourier Transform, convolution, applications[2,3,9]. In most of the DSP algorithms multiplier is one of the key component and hence a high speed and area efficient multiplier is needed and multiplication time is also one of the predominant factor for DSP algorithms. The ancient mathematical techniques like Vedic mathematics used to reduce the computational time such that it can increases speed and also requires less hardware. There are sixteen sutras and sixteen sutras (sub formulae) constructed by swahiji. Vedic is a word obtained from the word "Veda" and its meaning is "store house of all knowledge". Vedic mathematics mainly consists of the 16 sutras which it can be related to the different branches of mathematics like algebra, arithmetic geometry.

II. ANCIENT VEDIC MATHEMATICAL ALGORITHMS

The Vedic mathematics mainly reduces the complex typical calculations in to simpler by applying sutras as stated above. These Vedic mathematic techniques are very efficient and take very less hardware to implement. These sutras are mainly used for multiplication of two decimal numbers and we extend these sutras for binary multiplications. Some of the techniques are discussed below.

A. Urdhva -Tiryagbhyam Sutra (Vertically and Crosswise):

Booth multipliers are generally used for multiplication purposes. Booth Encoder, Wallace Tree, Binary Adders and Partial Product Generator are the main components used for Booth multiplier architecture. Booth multiplier is mainly used for 2 applications are to increase the speed by reduction of the partial products and also by the way that the partial products to be added. In this section we propose a Vedic multiplication technique called "Urdhva-Tiryakbhyam – Vertically and crosswise." Which can be used not only for decimal multiplication but also used for binary multiplication? This technique mainly consists of generation of partial products parallel and then we have to perform the addition operation simultaneously[3]. This algorithm can be used for 2x2, 4x4, 8x8....N×N bit multiplications. Since the sums and their partial products are calculated in parallel the Vedic multiplier does not depends upon the processor clock frequency. Hence there is no need of increasing the clock frequency and if the clock frequency increases it will automatically leads to the increase in the power dissipation. Hence by using this Vedic multiplier technique we can reduces the power dissipation. The main advantage of this Vedic multiplier is that it can reduces delay as well as area when compared with the other multipliers.

B. Example for Decimal Multiplication Using Vedic Mathematics:

To illustrate this technique, let us consider two decimal numbers 252 and 846 and the multiplication of two decimal numbers 252×846 is explained by using the line diagram shown in below figure1. First multiply the both numbers present on the two sides of the line and then first digit is stored as the first digit of the result and remaining digit is stored as pre carry for the next coming step and the process goes on and when there is more than one line then calculate the product of end digits of first line and add the result to the product obtained from the other line and finally store it as a result and carry. The obtained carry can be used a carry for the further steps and finally we will get the required result which is the final product of two decimal numbers 252x846. Take the initial carry value as the zero. For clear understanding purpose we explained the complete algorithm in the below line diagram such that each bit represents a circle and number of bits equal to the number of circles present.

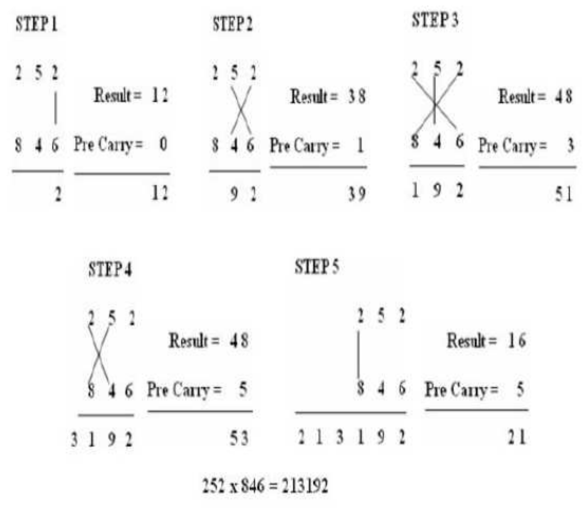


Figure 1. Multiplication of two decimal numbers

III. MODIFIED VEDIC MULTIPLIER ARCHITECTURE

The architectures for 2×2, 4×4, 8×8, 16×16. . .N×N bit modules are discussed in this section. In this section, the technique used is 'Urdhva-Tiryakbhyam' (Vertically and Crosswise) sutra which is a simple technique for multiplication with lesser number of steps and also in very less computational time. The main advantage of this Vedic multiplier is that we can calculate the partial products and summation to be done concurrently. Hence we are using this Vedic multiplier in almost all the ALU's.

A. 2×2 Vedic Multiplier Block

To explain this method let us consider 2 numbers with 2 bits each and the numbers are A and B where A=a0a1 and B=b0b1 as shown in the below line diagram. First the least significant bit (LSB) bit of final product (vertical) is obtained by taking the product of two least significant bit (LSB) bits of A and B is a0b0. Second step is to take the products in a crosswise manner such as the least significant bit (LSB) of the first number A (multiplicand) is multiplied with the next higher bit of the multiplicand B in a crosswise manner. The output generated is 1-Carry bit and 1bit used in the result as shown below. Next step is to take product of 2 most significant bits (MSB) and for the obtained result previously obtained carry should be added. The result obtained is used as the fourth bit of the final result and final carry is the other bit.

$$s0 = a0b0 \qquad (1)$$

$$c1s1 = a1b0 + a0b1 \qquad (2)$$

$$c2 \quad s2 = c1 + a1b1 \qquad (3)$$

The obtained final result is given as c2s2s1s0. A 2×2 Vedic multiplier block is implemented by using two half adders and four two input and gates as shown in below Figure 2.

B. 4x4 Vedic Multiplier Block

In this section, now we will discuss about 4x4 bit Vedic multiplier. For explaining this multiplier let us consider two four bit numbers are A and B such that the individual bits can be represented as the A3A2A1A0 and B3B2B1B0. The procedure for multiplication can be explained in terms of line diagram shown in below figure. The final output can be obtained as the C6S6S5S4S3S2S1S0. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure 3. After performing all the steps the result (Sn) and Carry(Cn) is obtained and in the same way at each step the previous stage carry is forwarded to the next stage and the process goes on.

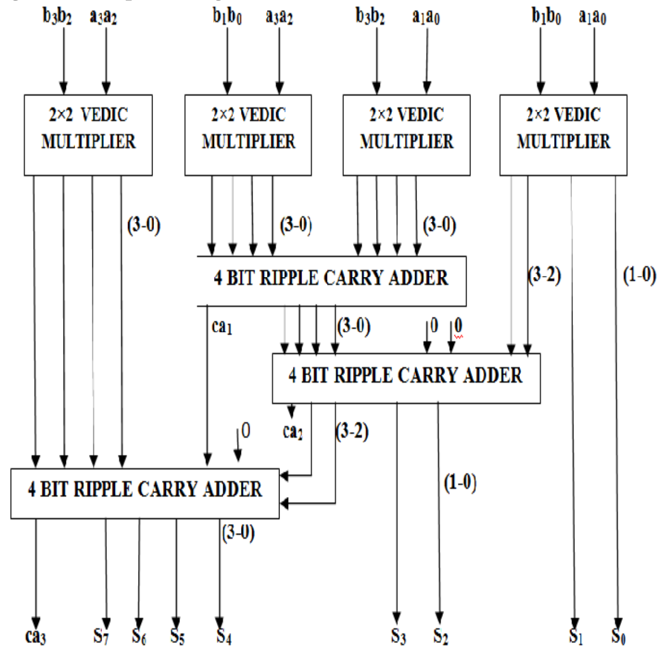


Figure 2. Block Diagram of 4x4 bit Vedic Multiplier

C. 8x8 Vedic Multiplier Block

In this section, now we will discuss about 4x4 bit Vedic multiplier. For explaining this multiplier let us consider two 8 bit numbers are A and B such that the individual bits can be represented as the A7A6A5A4A3A2A1A0 and B7B6B5B4B3B2B1B0. The procedure for multiplication can be explained in terms of line diagram shown in below figure 4. The final output can be obtained as the 16S15S14S13S12S11S10S9S8S7S6S5S4S3S2S1S0. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure 4. After performing all the steps the result (Sn) and Carry (Cn) is obtained and in the same way at each step the previous stage carry is forwarded to the next stage and the process goes on.

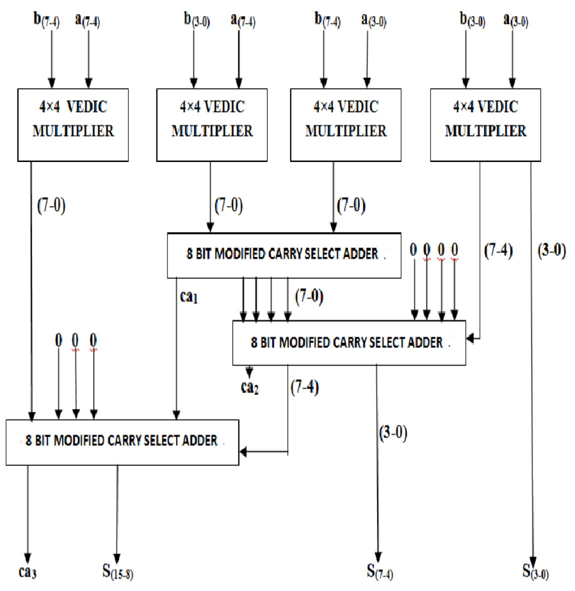


Figure 3. Block Diagram of 8x8 bit Vedic Multiplier

For clear understanding, observe the block diagrams for 8x8 as shown below and within the block diagram 8x8 totally there are four 4x4 Vedic multiplier modules, and three modified carry select adders which are of 8 bit size are used. The 8 bit modified carry select adders are used for addition of two 8 bits and likewise totally four are use at intermediate stages of multiplier. The carry generated from the first modified carry select adder is passed on to the next modified carry select adder and there are four zero inputs for second modified carry select adders. The arrangement of the modified carry select adders are shown in below block diagram which can reduces the computational time such that the delay can be decrease.

D. 16x16 Vedic Multiplier Block

In this section, now we will discuss about 4x4 bit Vedic multiplier[10]. For explaining this multiplier let us consider two 8 bit numbers are A and B such that the individual bits can be represented as the A [15:0] and B [15:0]. The procedure for multiplication can be explained in terms of line diagram shown in below figure 5. The final output can be obtained as the C16S[31:0]. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure. After performing all the steps the result (Sn) and carry (Cn) is obtained and in the same way at each step the previous stage carry is forwarded to the next stage and the process goes on. 16 Bit Modified Carry Select Adder: The Block diagram of the proposed Architecture consists of following parts

1) Ripple carry adder
2) Basic Unit (Binary to Excees-1 Converter)
3) Multiplexer

1) Ripple Carry Adder: Ripple carry adder is designed using multiple full adders to add 8-bit numbers. Each full adder inputs a Cin, which is the Cout of the previous adder. The adder is called a ripple-carry adder, since each carry bit "ripples" to the next full adder. The layout of a ripple-carry adder is simple, which allows for fast design time; however, the ripple-carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. Ripple adder is a combination of 4full adders in which output carry is used as input carry to the next full adder. RCA uses large number of AND, OR, NOT gates. It has the advantages of high speed and less delay.

2) Binary to Excess-1 Code Converter: In Binary to Excess one converter we are using XOR, AND, NOT gates by implementing these gates we are reducing the area, time delay, power consumption because of reduction in number of gates when compared to normal ripple carry adder. The 4 Bit Binary to Excess-1 Code Converter is shown in figure 5. The additionis achieved. Using BEC together with a multiplexer (mux) one input is the output of ripple carry adder gets as it input and another input of the mux is the BEC output. This gives the two partial results in parallel and the Mux are used to select either BEC output or the direct inputs according to the given control signal.

The Boolean Expressions of 4-bit BEC given below

$X0 = \sim B0$
$X1 = B0 \wedge B1$
$X2 = B2 \wedge (B0 \& B1)$
$X3 = B3 \wedge (B0 \& B1 \& B2)$

3) Multiplexer: Multiplexer is also called Universal element or Data Selector. A Multiplexer has of 2^n inputs have n select lines Basically MUX operation based on the select lines. Depending upon the select line the input is Send to the output. Multiplexers used to increase the amount of data that can be sent over the network. In this experiment, for each RCA we are using the five 2:1 Multiplexers and the outputs can be obtained as the five values and among them one can be used as the carry for the next block and remaining can be used for

the representation of the output. The 16 Bit Modified Carry Select Adder is shown in below figure 6. The values of 4 bit can be taken and remaining can be obtained from the next blocks. Like that we will obtain totally sixteen outputs and those are outputs of the sixteen bit addition. The five Multiplexers can give the outputs as the ripple carry adder or the binary to excess one converter output. Based up on the control signal we are selecting the output.

For clear understanding, observe the block diagrams for 16x16 as shown below and within the block diagram 16x16 totally there are four 8x8 Vedic multiplier modules, and three modified carry select adders which are of 16 bit size are used. The 16 bit modified carry select adders are used for addition of two 16 bits and likewise totally four are use at intermediate stages of multiplier. The carry generated from the first modified carry select adder is passed on to the next modified carry select adder and there are eight zero inputs for second modified carry select adders. The arrangement of the modified carry select adders is shown in below block diagram which an reduces the computational time such that the delay can be decrease.

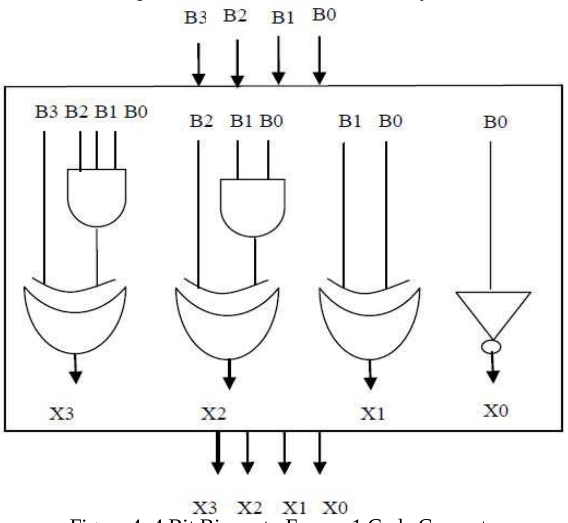


Figure 4. 4 Bit Binary to Excess-1 Code Converter

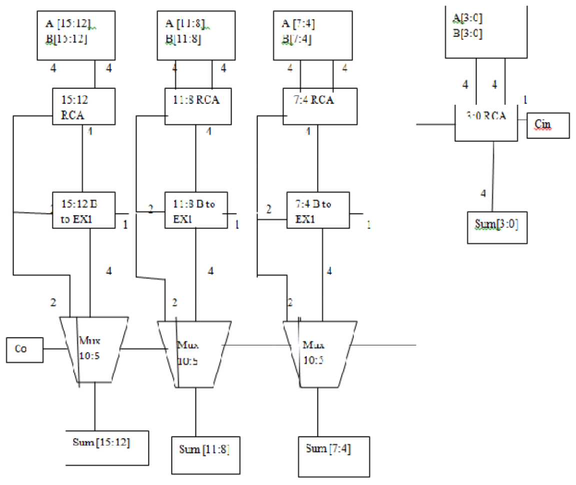


Figure.5 .16 Bit Modified Carry Select Adder

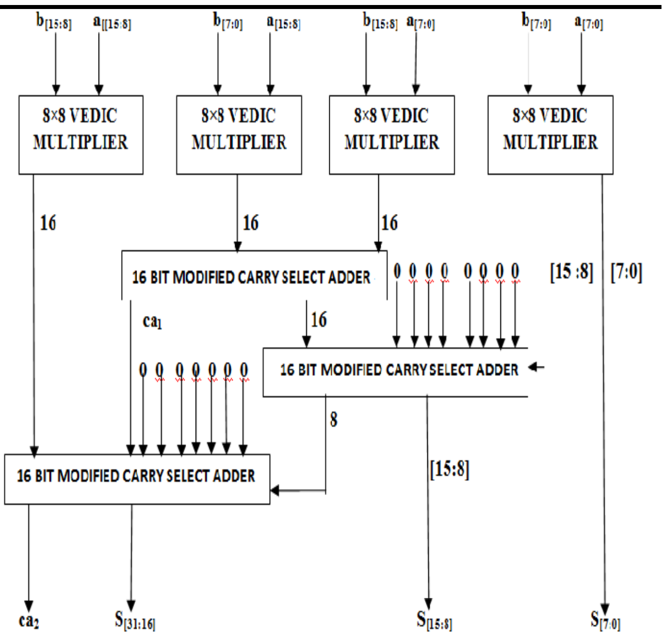
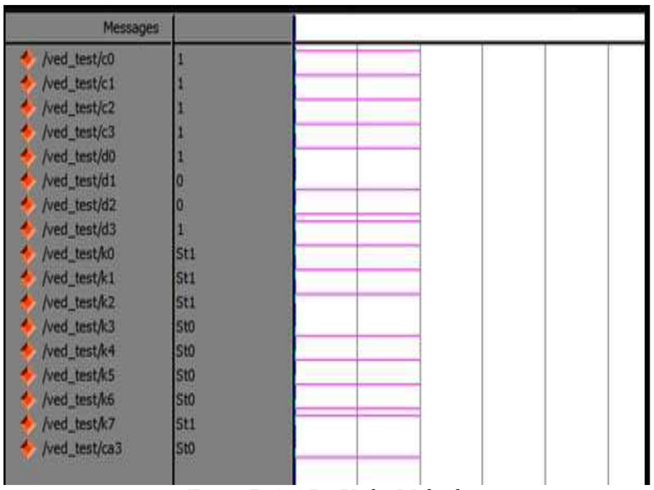


Figure 6. Block Diagram of 16x16 bit Vedic Multiplier

IV. RESULTS AND COMPARISON

In 2×2, 4×4, 8×8, 16×16 multiplication operations were designed using Verilog HDL and simulation is performed in Modelsim, NCLaunch. RTL compilation is performed in RC and physical design is carried out using Encounter RTL Compiler under Cadence environment. The comparison table for power, delay area is given in the below table. The comparison results of Vedic multiplier with the booth multiplier is given within the below table.



```
/ved_test/c0   1
/ved_test/c1   1
/ved_test/c2   1
/ved_test/c3   1
/ved_test/d0   1
/ved_test/d1   0
/ved_test/d2   0
/ved_test/d3   1
/ved_test/k0   St1
/ved_test/k1   St1
/ved_test/k2   St1
/ved_test/k3   St0
/ved_test/k4   St0
/ved_test/k5   St0
/ved_test/k6   St0
/ved_test/k7   St1
/ved_test/ca3  St0
```

Figure 7. 4×4 Bit Vedic Multiplier

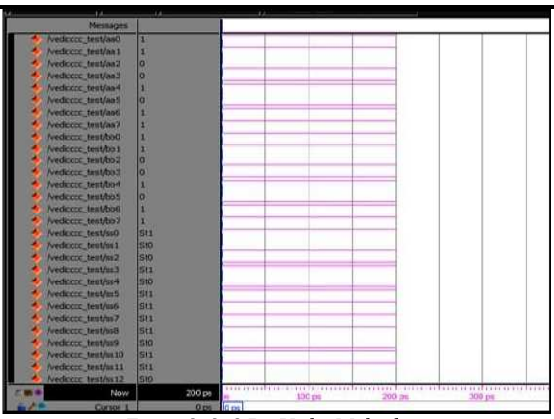


Figure 8. 8×8 Bit Vedic Multiplier

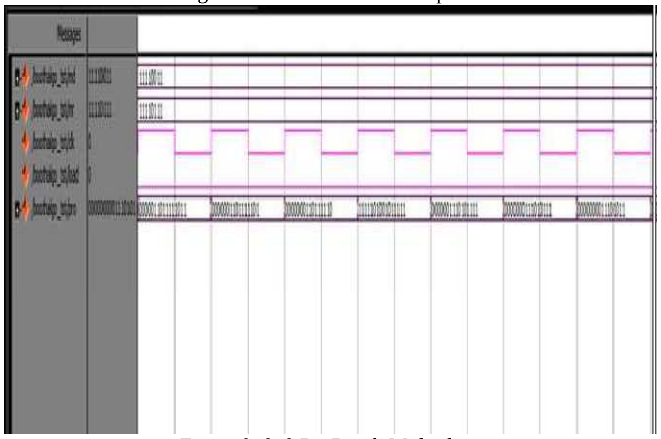


Figure 9. 8x8 Bit Booth Multiplier

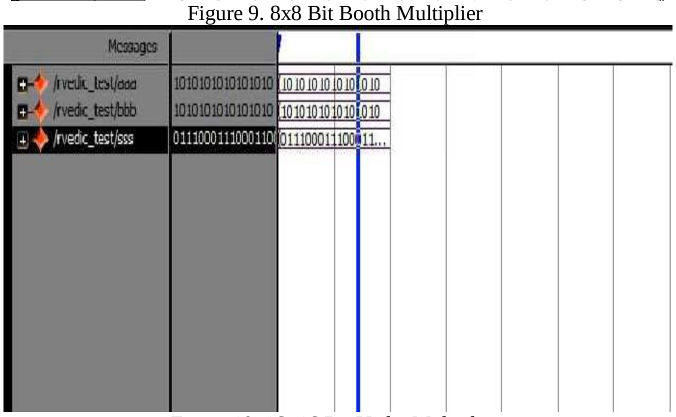

Messages
/rvedic_test/aaa 1010101010101010
/rvedic_test/bbb 1010101010101010
/rvedic_test/sss 0111000111000110

Figure 10. 16×16 Bit Vedic Multiplier

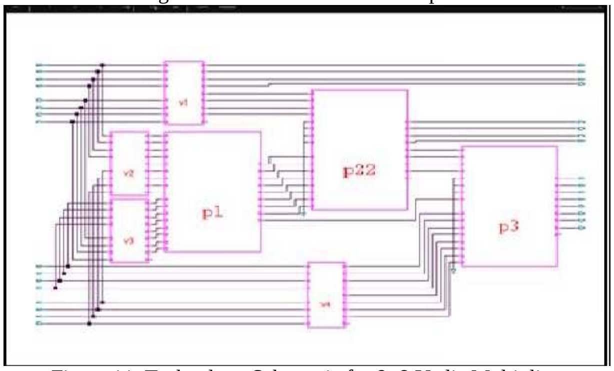


Figure 11. Technology Schematic for 8×8 Vedic Multiplier

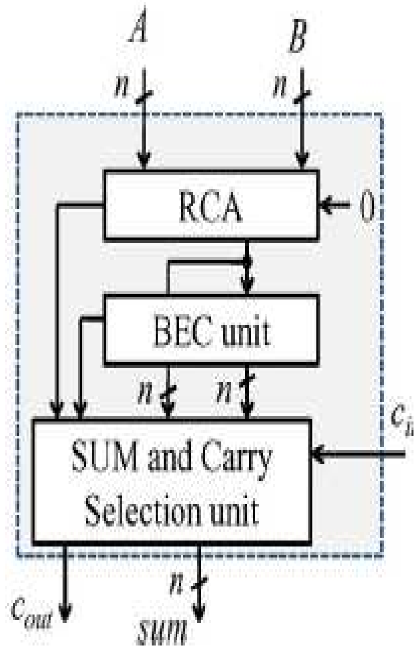


Figure.12. Structure of the BEC-based CSLA

A. Logic Expressions of the SCG Unit of the Conventional CSLA

As shown in Fig. 1(a), the SCG unit of the conventional CSLA [3] is composed of two n-bit RCAs, where n is the adder bit-width. The logic operation of the n-bit RCA is performed in four stages: 1) half-sum generation (HSG); 2) half-carry generation (HCG); 3) full-sum generation (FSG); and 4) full-carry generation (FCG). Suppose two n-bit operands are added in the conventional CSLA, then RCA-1 and RCA-2 generate n-bit sum (s0 and s1) and output-carry (c0out and c1out) corresponding to input-carry (cin =0and cin =1), respectively. Logic expressions of RCA-1 and RCA-2 of the SCG unit of the n-bit CSLA are given as

$$s_0^0(i) = A(i) \oplus B(i) \quad c_0^0(i) = A(i) \cdot B(i) \qquad \text{(1a)}$$

$$s_1^0(i) = s_0^0(i) \oplus c_1^0(i-1) \qquad \text{(1b)}$$

$$c_1^0(i) = c_0^0(i) + s_0^0(i) \cdot c_1^0(i-1) \quad c_{\text{out}}^0 = c_1^0(n-1) \qquad \text{(1c)}$$

$$s_0^1(i) = A(i) \oplus B(i) \quad c_0^1(i) = A(i) \cdot B(i) \qquad \text{(2a)}$$

$$s_1^1(i) = s_0^1(i) \oplus c_1^1(i-1) \qquad \text{(2b)}$$

$$c_1^1(i) = c_0^1(i) + s_0^1(i) \cdot c_1^1(i-1) \quad c_{\text{out}}^1 = c_1^1(n-1) \qquad \text{(2c)}$$

where $c_1^0(-1) = 0, c_1^1(-1) = 1, \text{and } 0 \leq i \leq n-1.$

As shown in (1a)–(1c) and (2a)–(2c), the logic expression of {s0 0(i),c0 0(i)} is identical to that of {s1 0(i),c1 0(i)}. These redundant logic operations can be removed to have an optimized design for RCA-2, in which the HSG and HCG of RCA-1 is shared to construct RCA-2. Based on this, [4] and [5] have used an add-one circuit instead of RCA-2 in the

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 8, August 2015)

CSLA, in which a BEC circuit is used in [6] for the same purpose. Since the BEC-based CSLA offers the best area–delay–power efficiency among the existing CSLAs, we discuss here the logic expressions of the SCG unit of the BEC-based CSLA as well.

B. Logic Expression of the SCG Unit of the BEC-Based CSLA As shown in Fig. 2, the RCA calculates n-bit sum s0 1 and c0out corresponding to cin =0. The BEC unit receives s0 1 and c0out from the RCA and generates (n + 1)-bit excess-1 code. The most significant bit (MSB) of BEC represents c1out, in which n least significant bits (LSBs) represent s1 1. The logic expressions of the RCA are the same as those given in (1a)–(1c). The logic expressions of the BEC unit of the n-bit BEC-based CSLA are given as

$$s_1^1(0) = \overline{s_1^0}(0) \quad c_1^1(0) = s_1^0(0) \qquad (3a)$$

$$s_1^1(i) = s_1^0(i) \oplus c_1^1(i-1) \qquad (3b)$$

$$c_1^1(i) = s_1^0(i) \cdot c_1^1(i-1) \qquad (3c)$$

$$c_{\text{out}}^1 = c_1^0(n-1) \oplus c_1^1(n-1) \qquad (3d)$$

$\text{for } 1 \leq i \leq n-1.$

We can find from (1a)–(1c) and (3a)–(3d) that, in the case of the BEC-based CSLA, c1 1 depends on s0 1, which otherwise has no dependence on s0 1 in the case of the conventional CSLA. The BEC method therefore increases data dependence in the CSLA. We have considered logic expressions of the conven- tional CSLA and made a further study on the data dependence to find an optimized logic expression for the CSLA. It is interesting to note from (1a)–(1c) and (2a)–(2c) that logic expressions of s0 1 and s1 1 are identical except the terms c0 1 and c1 1 since (s0 0 = s1 0 = s0). In addition, we find that c0 1 and c1 1 depend on {s0,c0,cin}, where c0 = c0 0 = c1 0. Since c0 1 and c1 1 have no dependence on s0 1 and s1 1, the logic operation of c0 1 and c1 1 can be scheduled before s0 1 and s1 1, and the select unit can select one from the set (s0 1,s1 1) for the final-sum of the CSLA. We find that a significant amount of logic resource is spent for calculating{s0 1,s1 1}, and it is not an efficient approach to reject one sum-word after the calculation. Instead, one can select the required carry word from the anticipated carry words {c0 and c1} to calculate the final-sum. The selected carry word is added with the half-sum (s0) to generate the final-sum (s). Using this method, one can have three design advantages:

1) Calculation of s0 1 is avoided in the SCG unit;
2) the n-bit select unit is required instead of the (n + 1)bit; and
3) small output-carry delay. All these features result in an area–delay and energy-efficient design for the CSLA. We have removed all the redundant logic operations of (1a)–(1c) and (2a)–(2c) and rearranged logic expressions of (1a)–(1c) and (2a)–(2c) based on their dependence. The proposed logic formulation for the CSLA is given as

$$s_0(i) = A(i) \oplus B(i) \quad c_0(i) = A(i) \cdot B(i) \qquad (4a)$$

$$c_1^0(i) = c_1^0(i-1) \cdot s_0(i) + c_0(i) \quad \text{for } \left(c_1^0(0) = 0\right) \qquad (4b)$$

$$c_1^1(i) = c_1^1(i-1) \cdot s_0(i) + c_0(i) \quad \text{for } \left(c_1^1(0) = 1\right) \qquad (4c)$$

$$c(i) = c_1^0(i) \quad \text{if } (c_{\text{in}} = 0) \qquad (4d)$$

$$c(i) = c_1^1(i) \quad \text{if } (c_{\text{in}} = 1) \qquad (4e)$$

$$c_{\text{out}} = c(n-1) \qquad (4f)$$

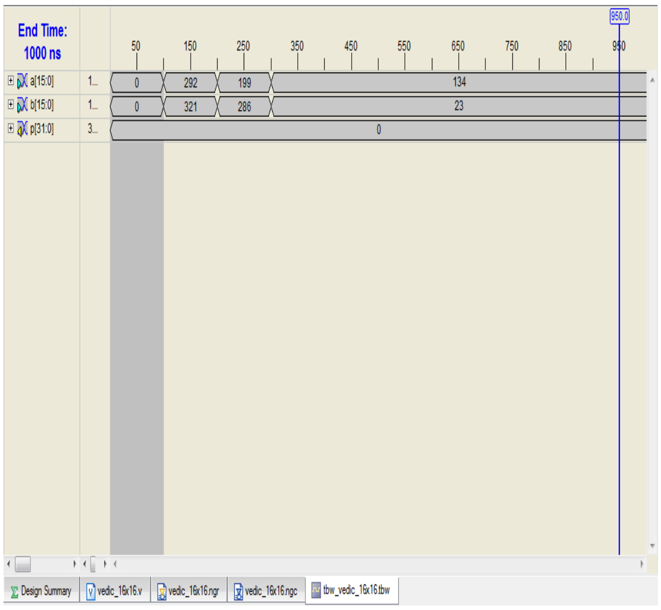
$$s(0) = s_0(0) \oplus c_{\text{in}} \quad s(i) = s_0(i) \oplus c(i-1). \qquad (4g)$$

V. PROPOSED ADDER DESIGN

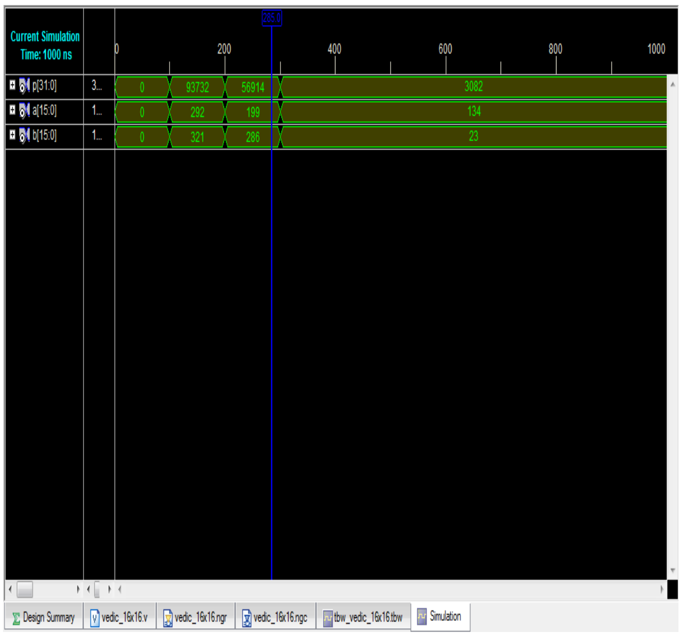
The proposed CSLA is based on the logic formulation given in (4a)–(4g), and its structure is shown in Fig. 3(a). It consists of one HSG unit, one FSG unit, one CG unit, and one CS unit. The CG unit is composed of two CGs (CG0 and CG1) corresponding to input-carry ‘0’ and ‘1’. The HSG receives two n-bit operands (A and B) and generate half-sum word s0 and half-carry word c0 of width n bits each. Both CG0 and CG1 receive s0 and c0 from the HSG unit and generate two n-bit full-carry words c0 1 and c1 1 corresponding to input-carry ‘0’ and ‘1’, respectively. The logic diagram of the HSG unit is shown in Fig. 3(b). The logic circuits of CG0 and CG1 are optimized to take advantage of the fixed input-carry bits. The optimized designs of CG0 and CG1 are shown in Fig. 3(c) and (d), respectively. The CS unit selects one final carry word from the two carry words available at its input line using the control signal cin. It selects c0 1 when cin =0; otherwise, it selects c1 1. The CS unit can be implemented using an n-bit 2-to-l MUX. However, we find from the truth table of the CS unit that carry words c0 1 and c1 1 follow a specific bit pattern. If c0 1(i) = ‘1’, then c1 1(i)=1 , irrespective of s0(i) and c0(i), for0 ≤ i ≤ n−1. This feature is used for logic optimization of the CS unit. The optimized design of the CS unit is shown in Fig. 3(e), which is composed of n AND–OR gates. The final carry word c is obtained from the CS unit. The MSB of c is sent to output as cout, and (n−1) LSBs are XORed with (n−1) MSBs of half-sum (s0) in the FSG [shown in Fig. 3(f)] to obtain (n−1) MSBs of final-sum (s). The LSB ofs0 is XORed with cin to obtain the LSB of s.

Final Results

```
RTL Top Level Output File Name     : vedic_16x16.ngr
Top Level Output File Name         : vedic_16x16
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO
Design Statistics

# IOs                              : 64

Cell Usage :
# BELS                             : 766
#      LUT2                        : 41
#      LUT3                        : 225
#      LUT4                        : 469
#      MUXF5                       : 31
# IO Buffers                    : 64
#     IBUF                    : 32
#     OBUF                     : 32

Selected Device : 3s500efg320-4
 Number of Slices:                    417  out of   4656     8%

 Number of 4 input LUTs:              735  out of   9312     7%

 Number of IOs:                       64
 Number of bonded IOBs:               64  out of    232    27%
.
```



Simulated output:



VI. CONCLUSION

The proposed Vedic multiplier gives less power consumption when compared to other multiplier techniques because the number of additions gets reduced by applying Urdhva-Tiryakbhyam which is a short approach form of multiplication. This multiplier has very less delay because of addition new module of modified carry select adder. This technique can be used well in DSP applications, as a squarer for the given number and cube also. Reducing the delay is very much advantage because it can increases speed and this Vedic multiplier is very useful for low power and high speed applications.

REFERENCES

[1]. Aniruddha Kanhe, Shishir Kumar Das and Ankit Kumar Singh, “Design and Implementation of Low Power Multiplier Using Vedic Multiplication Technique”, (IJCSC) International Journalof Computer Science and Communication Vol. 3, No. 1, January-June 2012, pp. 131- 132
[2]. H. Thapliyal and H.R Arbania. “A Time-Area-Power Efficient Multiplier and Square Architecture Based On Ancient Indian Vedic Mathematics”, Proceedings of the 2004, International Conference on VLSI (VLSI’04), Las Vegas, Nevada, June 2004, pp. 434-9.
[3]. Himanshu Thapliyal and M.B.Srinivas, “VLSI Implementation of RSA Encryption System Using Ancient Indian Vedic athematics”, Center for VLSI and Embedded System Technologies, International Institute of Information Technology Hyderabad-500019, India.
[4]. S. Hong, S. Kim, M.C. Papaefthymiou, and W.E.Stark, .Low power parallel multiplier design for DSP applications through coefficient optimization, in Proc. of Twelfth Annual IEEE Int. ASIC/SOConf. Sep. 1999, pp. 286-290.
[5]. R. Pushpangadan, V. Sukumaran, R.Innocent, D. Sasikumar, and V. Sundar, “High Speed Vedic Multiplier for Digital Signal Processors,” IETE Journal of Research, vol.55, pp.282- 286, 2009.
[6]. Devika, K. Sethi and R.Panda, “Vedic Mathematics Based Multiply Accumulate Unit,” 2011 International Conference on Computational Intelligence and Communication Systems, CICN 2011, pp.754-757, Nov. 2011.
[7]. Prabha S., Kasliwal, B.P. Patil and D.K. Gautam, “Performance Evaluation of Squaring Operation by Vedic Mathematics”, IETE Journal of Research, vol.57, Issue 1, Jan-Feb 2011.
[8]. P.D. Chidgupkar, and M.T. Karad, “The Implementation of Vedic Algorithms in Digital Signal Processing,” Global Journal of Engng. Educ., vol.8 , pp.153-158, 2004.
[9]. H.D. Tiwari, G. Gankhuyag, C.M. Kim, and Y.B. Cho, “Multiplier Design Based on Ancient Indian Vedic Mathematics”, Proc. Int SoC Design Conf., pp.65-68. 2008.
[10]. Umesh Akare, T.V. More and R.S. Lonkar, “Performance Evaluation and Synthesis of Vedic Multiplier”, National Conference on Innovative Paradigms in Engineering & Technology (NCIPET-2012), proceedings published by International Journal of Computer Applications (IJCA), 2012.