



IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 Issue: IX Month of publication: September 2022 DOI: https://doi.org/10.22214/ijraset.2022.46861

www.ijraset.com

Call: 🕥 08813907089 🔰 E-mail ID: ijraset@gmail.com

# Energy Efficient Approximate 8-bit Vedic Multiplier

Vijay Bhaskar Nittala<sup>1</sup>, Anisha Bomma<sup>2</sup>, M. Ramana Reddy<sup>3</sup>

<sup>1,2</sup>UG student, <sup>3</sup>Assistant Professor in Department of Electronics and Communication Engineering, Chaitanya Bharathi Institute of Technology, Gandipet, Hyderabad, India.

Abstract: The digital signal processing and its classification applications on the energy constrained devices should be supported based on efficiency. Because such applications must perform highly complex computations especially complex multiplication processes while exhibiting tolerance for a large amount of noise and for computational errors too. So, comparing all the arithmetic computations, improving the energy efficiency of multiplication is critical. In this project, an energy efficient approximate 8-bit Vedic multiplier is proposed which gives a tradeoff between computational accuracy and energy consumption. The proposed architecture has reduced area compared to other multiplier architectures which process same number of bits. The reduced architecture area reduces the power consumption. Also, the Vedic technology adopted for the multiplication reduces the delay further. But the approximate architecture output possesses a small amount of computational accuracy which is negligible for DSP applications.

Keywords: Digital Signal Processing, Energy Efficient, Vedic Multiplier, Approximate Multiplier Verilog HDL, and Simulations

## I. INTRODUCTION

For today's electronic world embedded system and mobiles, energy consumption is a critical design problem lot of efforts have already been taken at various levels for improving the energy efficiency. Among other arithmetic operations, multiplication is the most time and power consuming operation. It becomes more significant for large operands and complex multiplication. Considering these constraints, a new approximate multiplier is proposed. This new method can provide much more energy efficiency than the truncated methods. The error rate is low because it effectively captures the noteworthy lower bits. For DSP and its classification algorithm, generally one of two operands in the multiplication is stored in the memory. Here it is exploited to improve the energy efficiency of the approximate multiplier further. The area can be effectively reduced by the proposed approximate multiplier because many adders and gates can be excluded in this method as compared to other multipliers. Within the approximate multiplier architecture, a Vedic technology is used for the multiplication which reduces the delay in the multiplication. As the scale of integration keeps growing, more and more sophisticated signal processing systems are being implemented on a VLSI chip. These signal processing applications not only demand great computation capacity but also consume considerable amount of energy. While performance and Area remain to be the two major design tolls, power consumption has become a critical concern in today's VLSI system design. The need for low-power VLSI system arises from two main forces. First, with the steady growth of operating frequency and processing capacity per chip, large currents must be delivered and the heat due to large power consumption must be removed by proper cooling techniques. Second, battery life in portable electronic devices is limited. Low power design directly leads to prolonged operation time in these portable devices. Multiplication is a fundamental operation in most signal processing algorithms. Multipliers have large area, long latency and consume considerable power. Therefore, low-power multiplier design has been an important part in low power VLSI system design. There has been extensive work on low-power multipliers at technology, physical, circuit and logic levels. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly inlarger areas.

#### A. Vedic Multiplier

#### II. MULTIPLIER MODULE

The Booth multiplication algorithm and Array multiplication algorithm are associated with high propagation time. But the fast multiplication process is defined by Vedic Mathematics. The Veda means store of knowledge. The Vedic mathematics means storage of knowledge of mathematics related to the high-speed operations.



This Vedic mathematics provides the knowledge of varies area of mathematics like arithmetic, algebra, geometry using 16 important sutras. The method given in the Vedic math's is very simple. The Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja has combined all concepts together and given its explanation. Out of 16 sutras the Urdhva-Tiryagbhyam Vedic Sutra is very much useful multiplication operation. Present work is an effort to design and implement the multiplier using Urdhva-Tiryagbhyam Vedic Sutra.

# B. Urdhva-Triyagbhyam Sutra and its Implementation

The Urdhva Tiryagbhyam sutra is based on vertical and crosswise multiplication algorithm. This sutra enables the parallel generation of intermediate products and eliminates unwanted multiplication steps. This type of multiplier takes a form of one of the low powers and high-speed multiplier.

Design: Urdhva Triyagbhyam means "Vertically and Crosswise", which is the method of multiplication followed. For the Vedic multiplication consider a  $4\times4$  multiplier whose operands are A3A2A1A0 and B3B2B1B0.For the multiplication of a single bit a simple AND gate is used. So, the generalized equations for the output bits of 4X4 Vedicmultiplier can be written as:

- 1) S0=A0B0
- 2) S1=A0B1+A1B0
- 3) S2=A0B2+A1B1+A2B0+previous carry
- 4) S3=A0B3+A1B2+A2B1+A3B0+previous carry
- 5) S4=A1B3+A2B2+A3B1+previous carry
- 6) S5=A2B3+A3B2+previous carry
- 7) S6=A3B3

The S7 bit is the carry of the previous addition process. Thus, 8bit output is been obtained.

The block diagram of this sutra in being referred in the Fig. 1-

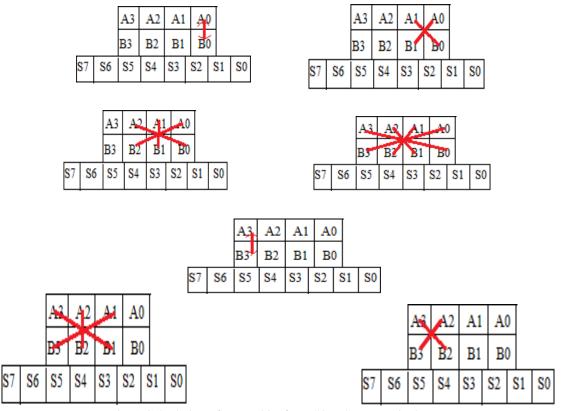


Fig. 1 Calculation of output bits from S0 to S7 respectively.



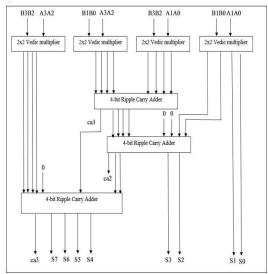


Fig.2Existing 4x4 Vedic Multiplier

Using four 2x2 bit Vedic multipliers and 3 adders we can build 4x4 bit Vedic multiplier as shown in Fig. 2 with proper instantiating of the 2x2 Vedic multipliers and adders. First 2x2 bit Vedic multiplier has inputs as A1A0 and B1B0. The last block is also 2x2 bit Vedic multiplier with inputs A3A2 and B3B2. The blocks in the middle are 2x2 bit multipliers with inputs A3A2 & B1B0 and A1A0 & B3B2. So, the result of multiplication will be of 8-bit as S7 S6 S5 S4 S3 S2 S1 S0. As compared to Array Multiplier, Vedic Multiplier is efficient in terms of delay and speed. The Vedic Multiplier can be used to reduce delay.

#### III. ARCHITECTURE OF APPROXIMATE 8-BIT MULTIPLIER

#### A. Design Process

The following steps are followed in the design of the multiplier

- 1) Design a basic 4-bit Adder which performs the function of 4-bit addition.
- 2) Design an efficient 4x4 bit Vedic multiplier using half adder, full adder and 4-bit adder.
- 3) Design 8x8 Vedic multiplier using 4x4 bit Vedic Multiplier and 8 Bit RippleCarry Adder.
- 4) Design 8-bit selection architecture using 2:1 Mux and design a 3:1 Mux toperform shifting operations on output bits.
- 5) Energy consumption and delay of the circuit is estimated.

#### B. Proposed 4 Bit Adder

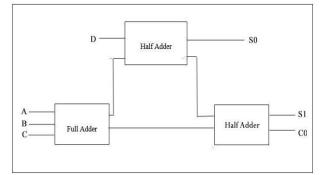


Fig. 3 A 4-bit adder is designed using half adders and full adder

Proposed 4-bit adder performs the function of 4-bit addition that gives two bits of sum and one carry as output. Its block diagram contains one full adder (FA) and two half adders (HA) as shown in the Fig.3. Here, A, B, C, D are four inputs. S0 and S1 are LSB and MSB of Sum outputs respectively and Sum is the sum of four inputs whereas C0 is the carry bit. The delay of the 4x4 Vedic Multiplier can be further decreased by using this proposed 4 Bit Adder.



C. 4X4 Vedic Multiplier

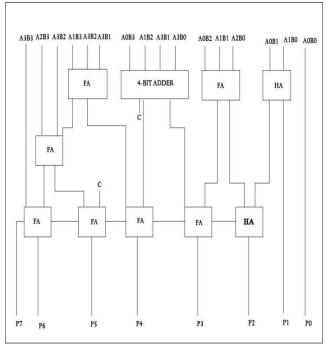


Fig. 4 4x4 Vedic Multiplier

To reduce the delay, a 4x4 Vedic multiplier is implemented using half adder, full adder and the proposed 4-bit adder as shown in the Fig. 4. Here, few full adders are replaced with the half adders so that the critical path delay is reduced to a greater extent. So, the designed multiplier is much more efficient than existing Vedic multiplier. The higher bit Vedic multiplier architectures like 8-bit, 16-bit, 32-bit etc. can be built using these 4-bit Vedic multipliers. The Fig.3.2 depicts two inputs consisting of 4-bit size that results by producing an output P resulting of 8-bit size.

# D. 8X8 Vedic Multiplier using 4X4 Vedic Multiplier

#### 1) 8-bit ripple carry adder

A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into next stage as shown in Fig. 5.

# The A and B are 8 bits each.A=A0 A1 A2 A3 A4 A5 A6 A7

#### B=B0 B1 B2 B3 B4 B5 B6 B7

A and B are added and the corresponding sum and carry is obtained. Here, S1, S2, S3, S4, S5, S6, S7, S8 is the sum. C8 is the carry out.

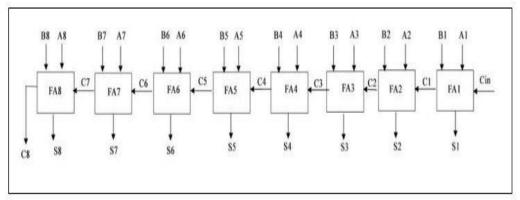


Fig. 5 8-bit Ripple Carry addder



# 2) 8X8 Vedic Multiplier

As shown in Fig. 6, the 8X8 Vedic Multiplier consists of 8-bit ripple carry adders and 4x4 Vedic Multipliers. So, 8bit inputs- a [0-7], b [0-7] are given and a 16-bit product s [15:0] is obtained.

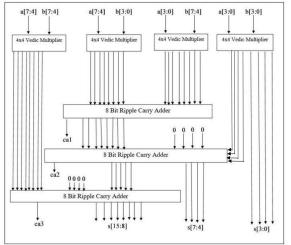
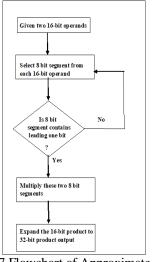


Fig. 6 8x8 Vedic Multiplier





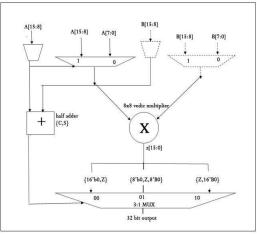


Fig. 8 8 Bit Approximate Vedic Multiplier



The structure of the proposed 8 Bit Selection Architecture of approximate multiplier is shown in Fig. 8. The architecture consists of two "8" bit selection units for each 16-bit operands, a multiplier unit, a 3 to 1 mux as output unit and a half adder which gives the selection bit for the output mux. Here the approximate multiplier takes only "8" consecutive bits for multiplication from each "16" bit operands. The 8-bit segment is selected only from one of two or three fixed bit positions of a positive number depending on where is its leading one bit. This technique gives a higher accuracy than simply truncating the LSBs. In this approach the 8-bit segments are selected from each operand from the leading one positions and it is then steered into an  $8 \times 8$  multiplier (which is here a Vedic multiplier) and then the 2\*8 output obtained from the multiplieris expanded to 2\*16 output bits.

## IV. DESIGN AND SIMULATION OF APPROXIMATE 8-BIT VEDIC MULTIPLIER

#### A. Simulation

The work deals with designing Approximate multiplier with 8-bit selection architecture and which also implements Vedic multiplication. So, the outputs are considered and simulated such that required operations of multiplier are verified. To obtain the delay report of project a simulating tool name XILINX ISE SUITE 14.7 is used, whereas to obtain the power report CADENCE simulating tool is used. Once all the required building blocks are designed and verified; these blocks are integrated to design the Approximate multiplier.

#### B. Simulation Waveforms

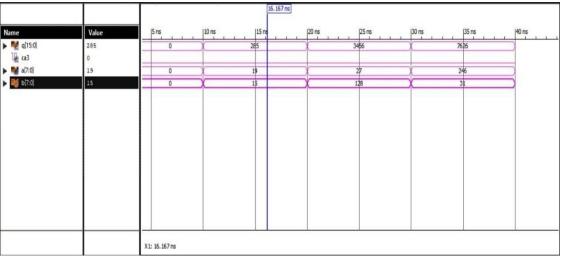


Fig. 9 Waveform of 8x8 Vedic Multiplier

| ISim R20         | )131013 (signa     | ture 0x8ef4fb         | (42)           |        |         |        |         |            |        |           |       |           |             |
|------------------|--------------------|-----------------------|----------------|--------|---------|--------|---------|------------|--------|-----------|-------|-----------|-------------|
| This is a l      | Full version of    | ISim.                 | 1216           |        |         |        |         |            |        |           |       |           |             |
| Time reso        | olution is 1 ps    |                       |                |        |         |        |         |            |        |           |       |           |             |
| Simulator        | r is doing circ    | uit initializatio     | n process.     |        |         |        |         |            |        |           |       |           |             |
| Finished         | circuit initializa | ation process         |                |        |         |        |         |            |        |           |       |           |             |
|                  | 0a= 0              | b= 0                  | ca3=0          | q=     | 0       |        |         |            |        |           |       |           |             |
|                  | 10a= 19            | b= 15                 | ca3=0          | q=     | 285     |        |         |            |        |           |       |           |             |
|                  | 20a= 27            | b=128                 | ca3=0          | q=     | 3456    |        |         |            |        |           |       |           |             |
|                  | 30a=246            | b= 31                 | ca3=0          | q=     | 7626    |        |         |            |        |           |       |           |             |
| Stopped<br>ISim> | at time : 40 ns    | s : in <u>File "C</u> | :/Users/anish/ | OneDri | ve/Desk | top/mp | 119/Pro | oject 4-2/ | bit8 v | adic/bit8 | vedic | tb.v" Lin | <u>e 60</u> |

From the simulation result, we can see the output of 8x8 bit Vedic multiplier for given 8-bit inputs a=19, b=15 is 285. Hence, the obtained output matches expected output.



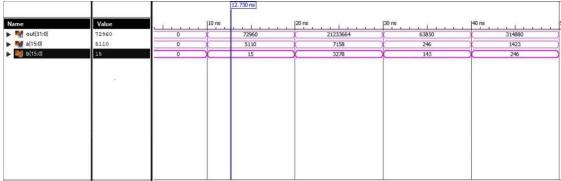


Fig. 10 Waveform of approximate 8-bit Vedic multiplier

| This is a Full version of ISi      | m.                           |                |  |    |
|------------------------------------|------------------------------|----------------|--|----|
| Time resolution is 1 ps            |                              |                |  |    |
| Simulator is doing circuit in      | nitialization process.       |                |  |    |
| Finished circuit initialization    | process.                     |                |  |    |
| 0a= 0                              | b= 0                         | out=           | 0  |    |
| 10a= 5110                          | b= 15                        | out=           | 72960                                      |    |
| 20a= 7158                          | b= 3278                      | out=           | 21233664                                   |    |
| 30a= 246                           | b= 143                       | out=           | 63850                                      |    |
| 40a= 1423                          | b= 246                       | out=           | 314880                                     |    |
| Stopped at time : 50 ns :<br>ISim> | in <u>File "C:/Users/ani</u> | ish/Desktop/ra | aspro/approx multi/approx multi tb.v" Line | 62 |

From the simulation result, we can see the output of the approximate 8-bit Vedic multiplier for the given input a=5110, b=15 is 72960 whereas the expected output is 76650. Hence, approximate multiplier produces approximate output instead of exact output and such a slight error in output is negligible for computation-intensive applications which are robust to small errors in computation.

A. Delay Report of 4x4 Vedic Multiplier

#### V. RESULTS AND DISCUSSIONS

| JESE Project Navigator (P.20131013) - G:\Project_4           ∑         File         Edit         View         Project         Source         Process           □         Image: Project and the pro  | s To | ools Window Layout Help  |   | ▶ ∑ <b>⊀</b>  | P   |               |  |  | - 8 x  |
|--|------|--|---|---|---|---------------|--|--|--|
| Start         → □ A ×          |      | Design Overview     Summary     Summary     OB Properties     Module Level Utilization     Timing Constraints     Prinout Report     Cock Report     Static Timing     Errors and Warning  | Summary     Module Level Utilization     Timing Constraints     Priout Report     Static Timing     Timing constraints     Cock Report     Static Timing     Timing | m combina<br>tails:<br>s display<br>nstraint:<br>umber of | red in nanos<br>Default pa<br>paths / des | th analy:     | ns)<br>Sis<br>ports:   | 342 / 8  |  |
| Image: Construction of the second | 24   | Synthesis Messages     Map Messages     Map Messages     Map Messages     Trining Messages     Bigler Messages     Detailed Roya Messages     Detailed Roya Messages     Detailed Roya Roya     July Messages     Detailed Roya Roya     Synthesis Roya     Synthesis Roya     Synthesis Roya     MD Parling     HDL Parling     HDL Parling | Delay:<br>Source:<br>Destina<br>Data Pa<br>Cell:<br><br>IBUF<br>LUT6<br>LUT6<br>LUT6<br>LUT4  |   | 10.072ns<br>b<0> (PA)<br>q<7> (PA)        | (Levels<br>D) | of Log:<br>Net<br>Delay<br>1.227<br>0.981<br>0.981<br>0.684<br>0.808 |  | (f4/Madd_n0003_Madd_lu<br>(f4/Madd_n0003_Madd_cy<br>(f7/Madd_n0003_Madd_cy |
|  |      | HDL Synthesis     HDL Synthesis Report     HDL Synthesis Report     Advanced HDL Synthesis Report     Low Level Synthesis  | Total   |   |   | 10.072n       |  | Rns logic, 5.260ns route)<br>& logic, 52.2% route) | -  |
|  | E    | Design Summ  | ary   |   | ×   |               |  |  |  |
| Errors   |      |  |   |   |   |               |  |  | +05×   |
| 4 m  |      |  |   |   |   |               |  |  |  |
| Console 🙁 Errors   |      |  |   |   |   |               |  |  |  |

Fig. 11 Delay Report of 4x4 Vedic Multiplier

Fig. 11 shows the delay report of 4x4 Vedic Multiplier. Delay obtained for 4x4 Vedic Multiplier is 10.072 ns. Hence it can be concluded that the performance of proposed Vedic multiplier seems to be highly efficient in terms of speed when compared to conventional multipliers and it is evident that the proposed design is more efficient than existing Vedic multiplier.



B. Delay Report of 8-bit Ripple Carry Adder

| ISE Project Navigator (P.20131013) - G:\Project_4-2\bit8_ripple_adder\bit8_ripple_adder.xise - [Design \$   |  |
|---|--|
| File Edit View Project Source Process Tools Window Layout Help  | - 8 ×  |
| 🗋 🍠 🗟 🕺 🕼 🗋 🗙 🐚 여 🗶 🖉 🖉 🖉 🔂 👘 🔂   |  |
| Start ↔ Π # x 👝 🗇 Design Overview   | Maximum combinational path delay: 8.317ns  |
| Design     ⊕ x     Implementation ● given and the set of the se                                | Timing Details:<br>All values displayed in nanoseconds (ns)<br>  |
| Construction of the sected.   | Delay: 8.317m (Levels of Logic = 6)<br>Source: a<2> (FAD)<br>Destination: s<7> (FAD)<br>Data Fath: a<2> to s<7><br>Cell:in->out fanout Delay Delay Logical Name (Net Name)<br>IBUF:I->O 2 1.222 0.981 a_2_IBUF (a_2_IBUF)  |
| Design Ubliftes     D | LUT5:I0->0 4 0.203 0.684 u2/Madd_n0003_Madd_cyc0>11 (u2/Madd_n0003_Madd_cy<br>LUT3:I4->0 2 0.205 0.681 u5/Madd_n003_Madd_cyc0>12 (u5/Madd_n003_Madd_cy<br>LUT3:I2->0 2 0.205 0.845 u5/Madd_n003_Madd_cyc0>12 (u5/Madd_n003_Madd_cy<br>LUT3:I2->0 1 0.205 0.579 u7/Madd_n003_Madd_cyc0>12 (u5/Madd_n003_Madd_cy<br>OBUF:I->0 2.571 a_7_0UF (s<7>)<br>Total 8.317hs (4.611ns logic, 3.706hs route)<br>(55.4% logic, 44.6% route) |
| Design Su   | mmary 🛛  |
| Errors  | + D & X  |
| 4   |  |
| Console 🔇 Errors  |  |

Fig. 12 Delay Report of 8-bit Ripple Carry Adder

The delay report of the 8-bit Ripple Carry Adder is shown in Fig. 12 and the delay obtained is 8.317ns. In ripple carry adder, each stage of the adder must wait until the previous stage has calculated and propagates its carry signal. Therefore, the total delay of the ripple carry adder is proportional to the number of bits. Faster adders are often required for bit widths of 16 or greater.

C. Delay Report of 8X8 Vedic Multiplier

| 🍃 ISE Project Navigator (P.20131013) - G:\Project_4 | 4-2\bi | it8_vedic\bit8_vedic.xise - [Design Summary] |                          |             |          |         |  |
|---|--------|--|--------------------------|-------------|----------|---------|--|
| Edit View Project Source Process                    | s To   | ools Window Layout Help                      |                          |             |          |         | - 8 ×  |
| o a  X @ @ X   0 0                                  | al     | » 🏓 🔑 🙉 🖉 🗟 🧖 🗟 🖽 🗖                          | 🖻 🔑 K? 🕨 🕱 📌             | <b>?</b>    |          |         |  |
| Start ↔ □ A X                                       | 1      | 🖻 Design Overview 🔺                          | Timing constraint:       | Default pat | h analys | sis     | A  |
| Design 🗗 🗶  | 0      | - 🖹 Summary                                  | Total number of p        | aths / dest | ination  | ports:  | 47898 / 17   |
| 📑 View: 🔿 🙀 Implementation 💿 🔜 Simulation           | G      | IOB Properties                               |                          |             |          |         |  |
|   |        | Module Level Utilization                     | Delay:                   | 18.489ns    |          | of Logi | .c = 14)   |
|   | -      | Timing Constraints     Pinout Report         | Source:                  | b<0> (PAI   |          |         |  |
| Hierarchy ^   | ç      | Pinout Report     Clock Report               | Destination:             | ca3 (PAD)   |          |         |  |
| 🔲 🦷 🗑 bit8_vedic                                    | 104    | Static Timing                                | Data Path: b<0> t        | 0 033       |          |         |  |
| aa 🖃 🛄 xc6slx45t-3fgg484                            | _      |  | baca racii. D(0) (       | 0 040       | Gate     | Net     |  |
| bit8_vedic_tb (G:\Project_4-2\k                     | 38     | Parser Messages                              | Cell:in->out             | fanout      | Delay    |         | Logical Name (Net Name)  |
| dur - bito_veuic (0.(Frojeci                        | M      | Synthesis Messages                           |                          |             |          | 1       |  |
| 56d b1 half addes (G)                               | (7)    | - 📓 Translation Messages                     | IBUF:I->0                | 24          | 1.222    | 1.517   | b 0 IBUF (b 0 IBUF)  |
| I f1 - full adder (G:\F                             |        | - 🖹 Map Messages                             | LUT6:I1->0               | 2           | 0.203    | 0.981   | u3/f1/Madd n0003 Madd cy<0>11 (u3/f1/Madd n0003 M  |
|   |        | Place and Route Messages                     | LUT6:I0->0               | 2           | 0.203    |         | u3/f4/Madd_n0003_Madd_lut<0>1 (u3/f4/Madd_n0003_M  |
| F2 - full_adder (G:\F                               |        | - 📄 Timing Messages                          | LUT5:I0->0               | 2           | 0.203    |         | u3/f4/Madd_n0003_Madd_xor<0>11 (w<19>)   |
| 4 <u> </u>  |        | - 🛐 Bitgen Messages                          | LUT6:I0->0               | 1           | 0.203    |         | f1/u3/Madd_n0003_Madd_lut<0>1 (f1/u3/Madd_n0003_M  |
| No Processes Running                                |        | - All Implementation Messages                | LUT6:I1->0               | 2           | 0.203    |         | f1/u3/Madd_n0003_Madd_xor<0>11 (t<3>)  |
| No Processes Running                                |        | Detailed Reports                             | LUT5:10->0               | 2           | 0.203    |         | f2/u3/Madd_n0003_Madd_lut<0>1 (f2/u3/Madd_n0003_M  |
| No single design module is selected.                |        | Synthesis Report                             | LUT6:I0->0<br>LUT5:I4->0 | 3           | 0.203    |         | f2/u3/Madd_n0003_Madd_cy<0>1 (f2/u3/Madd_n0003_Ma<br>f2/u4/Madd_n0003_Madd_cy<0>11 (f2/u4/Madd_n0003_M |
| 🖭 🗄 🎾 Design Utilities                              |        | Synthesis Report                             | LUT6:13->0               | 5           | 0.205    |         | f3/u1/Madd_n0003_Madd_cy<0>11 (f2/u4/Madd_n0003_M<br>f3/u1/Madd_n0003_Madd_lut<0>1 (f3/u1/Madd_n0003_M |
| <u></u>   |        | - Top of Report                              | LUT6:11->0               | 5           | 0.203    |         | f3/u2/Madd n0003 Madd cv<0>1 (f3/u2/Madd n0003 Ma  |
| 92  |        | Synthesis Options Summary                    | LUT6:14->0               | 2           | 0.203    |         | f3/u6/Madd n0003 Madd cy<0>1 (f3/u6/Madd n0003 Ma  |
|   |        | - HDL Parsing                                | LUT5: T0->0              | 1           | 0.203    |         | f3/u7/Madd n0003 Madd cy<0>11 (f3/u7/Madd n0003 M  |
| · • •   |        | HDL Elaboration                              | OBUF: I->0               |             | 2.571    |         | ca3 OBUF (ca3)   |
|   |        | 🖨 HDL Synthesis                              |                          |             |          |         |  |
|   |        | - HDL Synthesis Report                       | Total                    |             | 18.489ns | 6.233   | ins logic, 12.256ns route)   |
|   |        | Advanced HDL Synthesis                       |                          |             |          | (33.74  | : logic, 66.3% route)  |
|   |        | Advanced HDL Synthesis Report                | < m                      |             |          |         |  |
|   | Σ      | Low Level Sunthesis Topsian Sum              | nary                     |             |          |         |  |
|   | -      |  |                          |             |          | (1999)  |  |
| Errors  |        |  |                          |             |          |         | +□ 8 ×   |
| < <u> </u>  |        |  |                          |             |          |         | ,<br>,   |
| Console C Errors                                    |        |  |                          |             |          |         |  |

Fig. 13 Delay Report of 8X8 Vedic Multiplier

Fig. 13 shows the delay report of 8x8 Vedic Multiplier and Delay obtained is 18.489ns. Thus, proposed Vedic multiplier can be used to develop a high-speed complex number multiplier with reduced delay.



D. Delay Report of 8-bit Vedic Multiplier

| rt ↔□₽:<br>sian ₽:  |     | Design Overview     Summary  |      | Total numb  | er of pat                                    | hs / dest   | ination p  | ports:  | 1193304 / 32               |
|---|-----|--|------|---|--|---|--|---|----------------------------|
| View: () (() Implementation () (() () () () () () () () () () () ()   | 1 0 | DOB Properties     DOB Properties     DOB Properties     DoB Properties     DoB Properties     Did Level Utilization     Dinout Report     Cock Report     Cock Report     Cock Report     Did Did Did Did Did Did Did Did Did |      | lay:<br>Source:<br>Destinatic<br>Data Path:   | n:   | a<12> (PF<br>out<31> )  |  | of Logi<br>Net  | c = 17)                    |
| approx_multi     approx_multi     def      approx_multi     tb (G:\Project 4-   | 38  | Errors and Warnings  |      | Cell:in-  | >out   | fanout  | Delay  |   | Logical Name (Net Name)    |
| V approx.multi.b (G:)Project.<br>Vu approx.multi.b (G:)Project.<br>Vm and approx.multi.G:)Project.<br>Vm and approx.multi.S:)Project.<br>Vm and approx.multi.S:)Project.<br>Vm and ap |     |  |      | IBUF:I-<br>LUTS:IO<br>LUTS:IO<br>LUTS:IO<br>LUTS:IO<br>LUTG:I3<br>LUTG:I3<br>LUTG:I3<br>LUTG:I0<br>LUTG:I0<br>LUTG:I4<br>LUTG:I0<br>LUTG:I4<br>LUTS:IO<br>OBUF:I- | >0<br>>0<br>>0<br>>0<br>>0<br>>0<br>>0<br>>0 | 9<br>1<br>73<br>14<br>2<br>2<br>2<br>1<br>2<br>2<br>1<br>2<br>2<br>6<br>2<br>6<br>1<br>3<br>1 | 0.205<br>0.205<br>0.203<br>0.203<br>0.203<br>0.203<br>0.203<br>0.203<br>0.205<br>0.203 | 0.580<br>1.928<br>1.322<br>0.721<br>0.961<br>0.864<br>0.981<br>1.089<br>0.981<br>0.849<br>0.849<br>0.684<br>0.879 |                            |
|   |     | Advanced HDL Synthesis   |      | Total   |  |   | 22.093ns   | (6.848  | ins logic, 15.245ns route) |
|   |     | Advanced HDL Synthesis Report  |      |   |  |   |  | (31.0%  | logic, 69.0% route)        |
| Start h Files   | -   |  | •    | m   |  | -   |  |   |                            |
| Start C Files C Libraries   | E   | Design Summ  | mary |   |  |   |  |   |                            |
| ors   |     |  |      |   |  |   |  |   | + 🗆                        |

Fig. 14 Delay Report of 8-bit Vedic Multiplier

Delay obtained for the Approximate 8-bit Vedic Multiplier is 22.093ns. Thus, the bit reduction using 8 bit selection reduces the computation complexity which leads to a reduced delay and also the reduced delay Vedic architecture further reduces the delay.

#### E. Power Report of 4x4 Vedic Multiplier

| 0  | 1                              |              |               |          |                |
|--|--------------------------------|--------------|---------------|----------|----------------|
| Generated by: Encounter(R) RTL Compiler RC1<br>Generated on: Mar 14 2020 15:44:07<br>Module: bit4_vedic_multi<br>Technology library: tsmc18 1.0<br>Operating conditions: slow (balanced_tree)<br>Wireload mode: enclosed | 4.28 - v14.20-s067_1 (Jun 22.2 | 0016)        |               |          |                |
| Instance   | Cells                          | Leakage (nW) | Internal (nW) | Net (nW) | Switching (nW) |
| bit4_vedic_multi   | 112                            | 65.63        | 54353.84      | 35882.52 | 90236.3        |
| bit4_vedic_multi/f1  | 8                              | 3.92         | 4135.81       | 2512.32  | 6648.          |
| bit4_vedic_multi/f1/add_79_16  | 3                              | 1.68         | 1521.72       | 859.08   | 2380.          |
| bit4_vedic_multi/f1/add_79_18  | 5                              | 2.24         | 2614.09       | 1653.24  | 4267.3         |
| bit4_vedic_multi/f2  | 8                              | 3.92         | 3692.82       | 2083.80  | 5776.          |
| bit4_vedic_multi/f2/add_79_16  | 3                              | 1.68         | 1493.31       | 793.47   | 2286.          |
| bit4_vedic_multi/f2/add_79_18  | 5                              | 2.24         | 2199.51       | 1290.33  | 3489.8         |
| bit4_vedic_multi/f3  | 8                              | 3.92         | 4206.47       | 2051.68  | 6258.          |
| bit4_vedic_multi/f3/add_79_16  | 3                              | 1.68         | 1647.43       | 662.93   | 2310.0         |
| bit4_vedic_multi/f3/add_79_18  | 5                              | 2.24         | 2559.04       | 1388.75  | 3947.3         |
| bit4_vedic_multi/f4  | 8                              | 3.92         | 5850.67       | 2440.56  | 8291.2         |
| bit4_vedic_multi/f4/add_79_16  | 3                              | 1.68         | 2978.77       | 1321.08  | 4299.8         |
| bit4_vedic_multi/f4/add_79_18  | 5                              | 2.24         | 2871.90       | 1119.47  | 3991.0         |
| bit4_vedic_multi/f5  | 8                              | 3.92         | 5752.77       | 2899.83  | 8652.5         |
| bit4_vedic_multi/f5/add_79_16  | 3                              | 1.68         | 2400.45       | 1253.42  | 3653.8         |
| bit4_vedic_multi/f5/add_79_18  | 5                              | 2.24         | 3352.32       | 1646.40  |                |
| bit4 vedic multi/f6  | 8                              | 3.92         | 5264.89       | 2177.43  | 7442.3         |

Fig. 15 Power Report of 4x4 Vedic Multiplier

Fig. 15 shows the power report of a 4x4 Multiplier. The power consumed for 4x4 Multiplier is 0.180538mW. Here multiplication is based on Urdhva Triyakbhyam algorithm, which can reduce the power, delay, and hardware requirements for multiplication of numbers.



F. Power Report of 8-bit Ripple Adder

|   |                                    | Report Pow                           | er  |  |   |
|---|------------------------------------|--------------------------------------|---|--|---|
| Benerated by: Encounter(R) RTL Compiler RC<br>Senerated on: Mar 14 2020 15:53:20<br>Module: bit8_rtpple_adder<br>derbnology libray: Ismot 91.00<br>Operating conditions: slow (balanced_tree)<br>Vireload mode: enclosed  | 14.28 - v14.20:e067_1 (Jun 22 2016 | 9)                                   |   |  |   |
| Instance  | Cells                              | Leakage (nW)                         | Internal (nW)                                       | Net (nW)   | Switching (nW)  |
| it8_ripple_adder  | 64                                 | 31.32                                | 33274.84  | 22761.20   | 56036.04  |
| it8_ripple_adder/u0   | 8                                  | 3.92                                 | 2864.41   | 1286.91  | 4151.32   |
| l8_ripple_adder/u0/add_34_16  | 3                                  | 1.68                                 | 1336.86   | 727.18   | 2064.04   |
| t8_ripple_adder/u0/add_34_18  | 5                                  | 2.24                                 | 1527.55   | 559.74   | 2087.25   |
| t8_ripple_adder/u1  | 8                                  | 3.92                                 | 3500.91   | 1746.87  | 5247.70   |
| t8_ripple_adder/u1/add_34_16  | 3                                  | 1.68                                 | 1263.30   | 660.88   | 1924.1  |
| t8_ripple_adder/u1/add_34_18  | 5                                  | 2.24                                 | 2237.61   | 1085.98  | 3323.6  |
|   | 8                                  | 3.92                                 | 4118.45   | 2043.48  | 6161.9  |
| t8_ripple_adder/u2  |                                    |                                      |   |  |   |
|   | 3                                  | 1.68                                 | 1393.68   | 858.40   | 2252.0  |
| it8_ripple_adder/u2/add_34_16   | 3                                  | 1.68                                 | 1393.68   | 858.40<br>1185.08                                  |   |
| t8_ripple_adder/u2/add_34_16<br>t8_ripple_adder/u2/add_34_18  |                                    |                                      |   |  | 3909.8  |
| t8_ripple_adder/u2/add_34_16<br>t8_ripple_adder/u2/add_34_18<br>t8_ripple_adder/u3  | 5                                  | 2.24                                 | 2724.77   | 1185.08  | 3909.8<br>6793.4  |
| t8_ripple_adder/u2/add_34_16<br>t8_ripple_adder/u2/add_34_18<br>t8_ripple_adder/u3<br>t8_ripple_adder/u3/add_34_16  | 5                                  | 2.24<br>3.92                         | 2724.77 4486.82                                     | 1185.08<br>2306.60                                 | 3909.8<br>6793.4<br>2158.0  |
| 18_ripple_adder/u2/add_34_16<br>18_ripple_adder/u2/add_34_18<br>18_ripple_adder/u3<br>18_ripple_adder/u3/add_34_16<br>18_ripple_adder/u3/add_34_18  | 5<br>8<br>3                        | 2.24<br>3.92<br>1.68                 | 2724.77<br>4486.82<br>1365.27                       | 1185.08<br>2306.60<br>792.79                       | 3909.8<br>6793.4<br>2158.0<br>4635.3  |
| 18. r/pple_adder/u2/add_34_16<br>18. r/pple_adder/u2/add_34_18<br>18. r/pple_adder/u3/add_34_16<br>18. r/pple_adder/u3/add_34_16<br>18. r/pple_adder/u3/add_34_18<br>18. r/pple_adder/u4  | 5<br>8<br>3<br>5                   | 2.24<br>3.92<br>1.68<br>2.24         | 2724.77<br>4486.82<br>1365.27<br>3121.55            | 1185.08<br>2306.60<br>792.79<br>1513.81            | 3909.8<br>6793.4<br>2158.0<br>4635.3<br>7188.1  |
| 18. rpple_adder/u2           18. rpple_adder/u2/add_34_16           18. rpple_adder/u2/add_34_18           18. rpple_adder/u3/add_34_16           18. rpple_adder/u3/add_34_18           18. rpple_adder/u3/add_34_18           18. rpple_adder/u3/add_34_18           18. rpple_adder/u3/add_34_18           18. rpple_adder/u4/add_34_16           18. rpple_adder/u4/add_34_16           18. rpple_adder/u4/add_34_16           18. rpple_adder/u4/add_34_16 | 5<br>8<br>3<br>5<br>8              | 2.24<br>3.92<br>1.68<br>2.24<br>3.92 | 2724.77<br>4486.82<br>1365.27<br>3121.55<br>4716.17 | 1185.08<br>2306.60<br>792.79<br>1513.81<br>2471.99 | 2252 (0<br>3909.8<br>6733.4<br>2158.0<br>4435.3<br>7788.1<br>2158.0<br>2158.0<br>5030.1<br>7309.3 |

Fig. 16 Power Report of 8-bit Ripple Adder

Fig. 16 shows the power report of the 8-bit ripple adder. The power consumed for 8 Bit Ripple Carry Adder is 0.1121mW. The advantage of the ripple carry adder is low power consumption when compared to another adder

#### G. Power Report of 8x8 Vedic Multiplier

| Generated by: Encounter(R) RTL Compiler RC | 14.28 - v14.20-s067_1 (Jun 22 20 | 16)          |               |           |                |
|--|----------------------------------|--------------|---------------|-----------|----------------|
| Generated on: Mar 14 2020 16:28:43         |                                  | ,            |               |           |                |
| Module: bit8_vedic                         |                                  |              |               |           |                |
| Technology library: tsmc18 1.0             |                                  |              |               |           |                |
| Operating conditions: slow (balanced_tree) |                                  |              |               |           |                |
| Wireload mode: enclosed                    |                                  |              |               |           |                |
|  |                                  |              |               |           |                |
|  |                                  |              |               |           |                |
|  |                                  |              |               |           |                |
|  |                                  |              |               |           |                |
|  |                                  |              |               |           |                |
|  |                                  |              |               |           |                |
| Instance                                   | Cells                            | Leakage (nW) | Internal (nW) | Net (nW)  | Switching (nW) |
| bit8_vedic                                 | 640                              | 356.49       | 366128.07     | 214699.16 | 580827         |
| bit8_vedic/f1                              | 64                               | 31.32        | 56145.89      | 30147.80  | 86293          |
| bit8_vedic/f1/u0                           | 8                                | 3.92         | 2431.12       | 1257.53   | 3688           |
| bit8_vedic/f1/u0/add_106_16                | 3                                | 1.68         | 1152.00       | 595.27    | 1747           |
| bit8_vedic/f1/u0/add_106_18                | 5                                | 2.24         | 1279.12       | 662.25    | 1941           |
| bit8_vedic/f1/u1                           | 8                                | 3.92         | 4457.30       | 2216.39   | 6673           |
| bit8_vedic/f1/u1/add_106_16                | 3                                | 1.68         | 2045.56       | 992.35    | 3037           |
| bit8_vedic/f1/u1/add_106_18                | 5                                | 2.24         | 2411.73       | 1224.04   | 3635           |
| bit8_vedic/f1/u2                           | 8                                | 3.92         | 8539.06       | 4860.61   | 13399          |
| bit8_vedic/f1/u2/add_106_16                | 3                                | 1.68         | 3541.22       | 1982.65   | 5523           |
| bit8_vedic/f1/u2/add_106_18                | 5                                | 2.24         | 4997.84       | 2877.96   | 7875           |
| bit8_vedic/f1/u3                           | 8                                | 3.92         | 9702.29       | 5023.95   | 14726          |
| bit8_vedic/f1/u3/add_106_16                | 3                                | 1.68         | 3891.86       | 2114.56   | 6006           |
| bit8_vedic/f1/u3/add_106_18                | 5                                | 2.24         | 5810.43       | 2909.39   | 8719           |
| bit8_vedic/f1/u4                           | 8                                | 3.92         | 10446.83      | 5883.03   | 16329          |
| bit8_vedic/f1/u4/add_106_16                | 3                                | 1.68         | 4195.44       | 2313.44   | 6508           |
| bit8 vedic/f1/u4/add 106 18                |                                  | 2.24         | 6251.38       | 3569.59   | 9820           |

Fig. 17 Power Report of 8x8 Vedic Multiplier

Fig. 17 shows the report of the 8x8 Vedic Multiplier. The power consumed for 8X8 VedicMultiplier is 1.16201mW.



H. Power Report of Approximate 8 Bit Vedic Multiplier

|  | ۵ 🖉                               |   |  | 🕐 🕼   | 📑 Sat Mar 14, 4:40 PM 🛛 ro   |
|--|-----------------------------------|---|--|---|--|
|  |                                   | Report Pow  | er   |   | _ 0  |
| Samentad by: Encounter(F) RTL Compiler RC<br>Jamentad on: Wart 4 2020 16:38:57<br>Wodule: approx_multi<br>Technology library: Ismc18 1.0<br>Operating conditions: slow (balanced_tree)<br>Wireload mode: enclosed  | 14.28 - v14.20-6067_1 (Jun 22 201 | 6)  |  |   |  |
| Instance   | Cells                             | Leakage (nW)  | Internal (nW)  | Net (nW)  | Switching (nW)   |
| pprox_multi  | 770                               | 517.69  | 491757.35  | 275106.83   | 766864.1   |
| pprox_multi/h1   | 8                                 | 6.50  | 0.00   | 0.00  | 0.0  |
|  |                                   |   |  |   |  |
| pprox_multi/h1/mux_carry_130_9   | 1                                 | 1.31  | 0.00   | 0.00  | 0.0  |
|  | 1                                 | 1.31  | 0.00   | 0.00  |  |
| pprox_multi/h1/mux_carry_135_14  | 1                                 |   |  |   | 0.0  |
| pprox_multi/h1/mux_carry_135_14<br>pprox_multi/h1/mux_sum_130_9  | 1<br>1<br>1<br>1                  | 1.31  | 0.00   | 0.00  | 0.0  |
| pprox_multi/h1/mux_carry_135_14<br>pprox_multi/h1/mux_sum_130_9<br>pprox_multi/h1/mux_sum_135_14   | 1<br>1<br>1<br>1<br>640           | 1.31<br>1.31  | 0.00   | 0.00  | 0.0<br>0.0<br>0.0  |
| pprox_multl/h1/mux_cany_135_14<br>pprox_multl/h1/mux_sum_130_9<br>pprox_multl/h1/mux_sum_135_14<br>pprox_multl/h2  | 1<br>1<br>1<br>640<br>64          | 1.31<br>1.31<br>1.31  | 0.00<br>0.00<br>0.00   | 0.00<br>0.00<br>0.00  | 0.0<br>0.0<br>0.0<br>601023.4  |
| pprox_multi/h1/mux_carry_135_14<br>pprox_multi/h1/mux_sum_130_9<br>pprox_multi/h1/mux_sum_135_14<br>pprox_multi/h2<br>pprox_multi/h2/f1  |                                   | 1.31<br>1.31<br>1.31<br>356.49  | 0.00<br>0.00<br>0.00<br>380351.08  | 0.00<br>0.00<br>0.00<br>220672.40   | 0.0<br>0.0<br>0.0<br>601023.4<br>93880.9   |
| pprox_multi/h1/mux_cany_135_14<br>pprox_multi/h1/mux_sum_130_9<br>pprox_multi/h1/mux_sum_135_14<br>pprox_multi/h2/11<br>pprox_multi/h2/11  | 64                                | 1.31<br>1.31<br>1.31<br>356.49<br>31.32   | 0.00<br>0.00<br>0.00<br>380351.08<br>61451.16  | 0.00<br>0.00<br>0.00<br>220672.40<br>32429.79   | 0.0<br>0.0<br>601023.4<br>93880.9<br>3118.4  |
| pprox_multi/h1/mux_sum_130_9           pprox_multi/h1/mux_sum_136_14           pprox_multi/h1/mux_sum_136_14           pprox_multi/h2           pprox_multi/h2           pprox_multi/h2/11/00           pprox_multi/h2/11/00   | 64<br>8                           | 1.31<br>1.31<br>1.31<br>356.49<br>31.32<br>3.92   | 0.00<br>0.00<br>380351.08<br>61451.16<br>2190.35   | 0.00<br>0.00<br>220672.40<br>32429.79<br>\$28.11  | 0.0<br>0.0<br>0.0<br>601023.4<br>93880.9<br>3118.4<br>1559.2                         |
| pprox_multih1/mux_carry_135_14<br>pprox_multih1/mux_sum_130_9<br>pprox_multih1/mux_sum_136_14<br>pprox_multih20<br>pprox_multih2011<br>pprox_multih2011/u0/add_158_16<br>pprox_multih2111/u0/add_158_18  | 64<br>8                           | 1.31<br>1.31<br>1.31<br>356.49<br>31.32<br>3.92<br>1.68   | 0.00<br>0.00<br>380351.08<br>61451.16<br>2190.35<br>1095.18  | 0.00<br>0.00<br>220672.40<br>32429.79<br>928.11<br>464.05                                   | 0.0<br>0.0<br>0.0<br>601023.4<br>93880.9<br>3184.4<br>1558.2<br>1559.2               |
| pprox_multih1/mx_cary_135_14<br>pprox_multih1/mx_sum_136_9<br>pprox_multih2<br>pprox_multih2<br>pprox_multih211<br>pprox_multih211<br>pprox_multih211/u0/add_158_16<br>pprox_multih211/u0/add_158_16<br>pprox_multih211/u1<br>pprox_multih211/u1   | 64<br>8                           | 1.31<br>1.31<br>1.31<br>356.49<br>31.32<br>3.92<br>1.68<br>2.24   | 0.00<br>0.00<br>380351.08<br>61451.16<br>2190.35<br>1055.18<br>1095.18                                 | 0.00<br>0.00<br>220672.40<br>32429.79<br>928.11<br>464.05<br>464.05                         | 0.0<br>0.0<br>0.0<br>601023.4<br>93880.9<br>3118.4<br>1559.2<br>10298.6<br>10298.6   |
| pprox_multi/h1/mux_sum_130_9           pprox_multi/h1/mux_sum_130_9           pprox_multi/h1/mux_sum_130_9           pprox_multi/h21           pprox_multi/h211           pprox_multi/h211/u0/add_158_16           pprox_multi/h211/u0/add_158_16           pprox_multi/h211/u1           pprox_multi/h211/u1/add_158_16   | 64<br>8                           | 1.31<br>1.31<br>356.49<br>31.32<br>3.92<br>1.68<br>2.24<br>3.82   | 0.00<br>0.00<br>380351.08<br>61451.16<br>2190.35<br>1055.18<br>1095.18<br>6555.40                      | 0.00<br>0.00<br>220672.40<br>32429.79<br>928.11<br>464.05<br>3704.23                        | 0.0<br>0.0<br>601023.4<br>93980.9<br>3118.4<br>1559.2<br>1559.2<br>10298.6<br>4761.6 |
| pprox_multih1/mux_carry_135_14 pprox_multih1/mux_sum_130_9 pprox_multih1/mux_sum_136_14 pprox_multih2911 pprox_multih2911/u0 pprox_multih2911/u0 add_158_16 pprox_multih2911/u1 add_158_16 pprox_multih2911/u1add_158_16 pprox_multih2911/u1add_158_16   | 64<br>8<br>3<br>5<br>8<br>3       | 1.31<br>1.31<br>1.31<br>356.49<br>3.32<br>3.92<br>1.68<br>2.24<br>3.92<br>1.68<br>1.89<br>1.88  | 0.00<br>0.00<br>380351.08<br>61451.16<br>2190.35<br>1095.18<br>1095.18<br>66955.40<br>3043.45          | 0.00<br>0.00<br>220672.40<br>332429.79<br>2928.11<br>464.05<br>464.05<br>3704.23<br>1718.16 | 0.0<br>0.0<br>0.0<br>0.0<br>0.0<br>0.0<br>0.0<br>0.0<br>0.0<br>0.0                   |
| upprox_multihn/imux_carry_130_3           upprox_multihn/imux_carry_135_14           upprox_multihn/imux_sum_136_14           upprox_multihn/imux_sum_135_14           upprox_multihn/imux_sum_135_14           upprox_multihn/imux_sum_135_14           upprox_multihn/imux_sum_135_14           upprox_multihn/imux_sum_135_16           upprox_multihn/imux_sum_135_16           upprox_multihn/imux_sum_135_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16           upprox_multihn/imuti/usid_155_16 | 64<br>8<br>3<br>5<br>8<br>3       | 1.31<br>1.31<br>1.31<br>356.49<br>31.32<br>1.88<br>2.24<br>4<br>3.392<br>1.88<br>2.24<br>1.88<br>2.24<br>2.24<br>2.24<br>2.24<br>2.24 | 0.00<br>0.00<br>380031.00<br>61451.6<br>2190.35<br>1095.18<br>1095.18<br>6595.40<br>3043.45<br>3551.96 | 0.00<br>0.00<br>220672.40<br>38429.79<br>928.11<br>464.05<br>3704.23<br>1718.16<br>1986.07  | 0.00<br>0.00<br>0.00<br>0.00<br>0.01<br>0.01<br>0.01<br>0.01                         |

Fig.18 Power Report of Approximate 8 Bit Vedic Multiplier

The powers consumed for the Approximate 8-bit Vedic Multiplier is 1.5342mW. As the significant reduction in the proposed architecture offers a great advantage in the reduction of total power consumption

| Table                | 1- Synthesis Results |            |
|----------------------|----------------------|------------|
|                      | Maximum              | Total      |
| Module               | Combination          | Power(mW)  |
|                      | Delay(ns)            |            |
| Full adder           | 5.422                | 0.01368594 |
| 4-bit adder          | 5.473                | 0.03325859 |
| 4X4 Vedic            | 10.072               | 0.18053835 |
| Multiplier           |                      |            |
| 8-bit ripple carry   | 8.317                | 0.1121034  |
| adder                |                      |            |
| 8x8 Vedic Multiplier | 18.489               | 1.16201092 |
| Mux                  | 6.211                | 0.26936871 |
| Mux+8X8 Vedic        | 22.093               | 1.53424605 |
| Multiplier           |                      |            |

Table 1- Synthesis Results

The Table 1 shows the outcome of various stages of the of the proposed multiplier So finally, after the overall synthesis and simulation of the approximate 8-bit Vedic Multiplier. Maximum Combinational delay is 22.093 ns and the total power consumed is 1.5342mW.

| Table 2- Comparison of | the implemented desig | n with reference design |
|------------------------|-----------------------|-------------------------|
|                        |                       |                         |

| Design                                | Bits | Area (No of | Delay (ns) | Power (W) |
|---------------------------------------|------|-------------|------------|-----------|
|                                       |      | slices)     |            |           |
| Approximate 8-bit<br>Vedic multiplier | 16   | 153         | 22.093     | 0.001534  |
| Iterative logarithmic multiplier      | 16   | 381         | 41.555     | 0.15687   |



Here we compare our design for the number of slices, delay, and power consumption. The data for comparison is taken from the reference paper. Thus, the proposed approximate 8-bit Vedic multiplier is more energy efficient than the other conventional multipliers. Hence the potential applications of this approximate multiplier fall mainly in areas where there is no strict requirement on accuracy or wheresuper-low power consumption and high-speed performance are more important than theaccuracy.

#### VI. CONCLUSIONS

Area, power consumption and increased delay are the constituent factors in VLSI design that degrades the performance of any circuit. This proposed architecture of an energy efficient multiplier in which the area, delay and power consumption are reduced to a great extent on comparing with the present works. A technique for the multiplication of 8-bit operands with the help of Vedic multiplier is described. The proposed architecture is based on "Urdhva Triyakbhyam" sutra of Vedic mathematics which is a general multiplication technique for multiplication. This algorithm makes the parallel generation of partial products and removes unwanted multiplication steps and hence results in faster multiplication. Also, with this algorithm the increase in delay and power consumption is less with the increase in number of bits. This proposed technique works in two steps. In the first step the equation for each bit of the resultant is computed. In the second step that equation is executed with the required multipliers and Ripple Carry Adders. The reduction of number of bits which takes part in multiplication reduces the complexity of the multiplication process and thus improves the efficiency. The number of, half adders and full adders used in the proposed multiplier architecture is much less than the conventional multiplier architectures. The significant reduction in the proposed architecture offers a great advantage in the reduction of area and therefore the total power consumption. Thus, the proposed 8-bit multiplier is more energy efficient than the other conventional multipliers. Also, the bit reduction using 8 bit selection reduces the computational complexity which leads to a reduced delay and also the reduced delay Vedic architecture further reduces the delay. The potential applications of this approximate Multiplier fall mainly in areas where there is no strict requirement on accuracy or where super-low power consumption and highspeed performance are more important than the accuracy. The 8-bit approximate Vedic multiplier can be used for various digital signal processing applications where complex computations are to be performed.

#### VII. FUTURE SCOPE

Future scope of the work includes improving the performance of the approximate Vedic multiplier by using Enhanced Static Segment Method for segment-based multiplication, to provide good accuracy at notably low energy consumption considerable reduction in area overhead and use of carry look-ahead adder in design of Vedic multiplier which in turn reduces the propagation delay by introducing more complex hardware.

#### REFERENCES

- K. Y. Kyaw, W. L. Goh and K. S. Yeo, "Low power high-speed multiplier for error tolerant application", in IEEE Intl. Conf. Electron Devices and Solid-State Circuits (EDSSC), 2010, pp. 1-4.
- [2] Kaustubh Manikrao Gaikwad and Mahesh Shrikant Chavan," Analysis of Array Multiplier and Vedic Multiplier using Xilinx", in Communications on Applied Electronics (CAE), Volume 5 No.1, May 2016.
- [3] Mageshwari , G.A. Sathish Kumar," Design and Development of Approximate Multiplier Using Urdhva Algorithm", in International Journal of Innovative Research in Computer and Communication Engineering, Vol.5, Special Issue 3, April 2017.
- [4] B.Anish, Fathima, C.Vasanthanayaki," Design and Implementation of Energy Efficient Approximate Multiplier", International Journal of Computer Applications (0975 8887), NCICT (2015).
- [5] Sumant Mukherjee, Dushyant Kumar Soni, "Energy Efficient Multiplier forHigh-Speed DSP Application", in IJCSMC, Vol. 4, Issue. 6, June 2015.
- [6] Pushpalata Verma, "Design of 4x4 bit Vedic Multiplier using EDA Tool", in International Journal of Computer Applications (0975-888), Volume 48-No.20, June 2012
- [7] "Synthesis and Simulation Design Guide" available at www.xlinx.com
- [8] "Digital Design Flow" for Cadence
- [9] Z. Babi'c, A. Avramovi'c, and P. Bulic, "An Iterative Logarithmic Multiplier" in 2011



## **AUTHORS PROFILE**



Dr. M. Ramana Reddy has a research experience of more than 4 years. He is working as an Assistant professor in ECE Department at Chaitanya Bharathi Institute of Technology. His educational qualification is Doctor of Philosophy in VLSI. He has published 6 research papers in international journals. His area of specialization is CMOS VLSI.



Mr. Vijay Bhaskar Nittala is a young researcher pursued Bachelor of Engineering in Electronics and Communication at Chaitanya Bharathi Institute of Technology, Hyderabad. He has been trained in System Verilog, Computer Architecture and Discrete Mathematics. He has published 2 research papers in international journals. His research areas of interest are VLSI and Computer Engineering.



Mrs. Anisha Bomma is a young researcher currently pursuing Masters of Technology in Embedded Systems at Vellore Institute of Technology, Tamil Nadu. She completed Bachelor of Engineering in Electronics and Communication at Chaitanya Bharathi Institute of Technology, Hyderabad. She has been trained in System Verilog, Embedded Systems and System Engineering. Her research areas of interest are VLSI, Embedded Systems and Micro-Electronics.











45.98



IMPACT FACTOR: 7.129







# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 🕓 (24\*7 Support on Whatsapp)