An efficient high speed squaring and multiplier architecture using yavadunam sutra and bit reduction technique

Deepa A¹, Marimuthu C N² and Murugesan C³
¹Department of ECE, Assistant Professor, J.K.K.Munirajah College of Technology, Gobi, Erode, Tamil Nadu, India
² Department of ECE, Professor and Dean, Nandha Engineering College, Erode, Tamil Nadu, India
³ Department of ECE, J.K.K.Munirajah College of Technology, Gobi, Erode, Tamil Nadu, India

Email: adeepaeswaramoorthy@gmail.com

Abstract: Vedic Mathematics, an ancient Indian technique can be used to solve any arithmetic problems in an easy and simple way. A novel high speed Vedic squaring and multiplier unit is designed using the principles of Yavadunam sutra and the bit reduction technique is projected in this paper. The complexity of the multiplier is reduced as the bit reduction technique is employed and later the Yavadunam sutra is implemented for the calculation of the deficiency. The size of the proposed N bit multiplier is reduced to N-1 bit and also considerable speed improvement is achieved. The architecture is designed and realized using Xilinx Spartan FPGA and synthesized using 90nm and 180nm technology synopsys device.

Keywords—Vedic mathematics, Yavadunam Sutra, Vedic Squaring, Vedic multiplier, Bit reduction.

1. Introduction

The drastic growth in digital technology has led to the expansion of computer and DSP applications, leading to the increase in demand for the high speed processing. In most of the real time signal processing and image processing applications the desired performances can be achieved by the high throughput arithmetic operations. Multiplication is one of the key and arduous numerical operations employed in most digital signal processing applications [11]. The design of multiplier is a multifaceted task when compared to other binary operations. Consequently it is essential to design a multiplier with high speed and significantly less delay and less area with increase in bits number. In the design of a multiplier, optimization of the speed and area are the foremost issue [20]. On comparing the performances of the conventional multipliers with that of the Vedic multipliers, Vedic multipliers are faster and take up less area. A high speed Vedic multiplier using the principles of Yavadunam Sutra and bit reduction technique is proposed in this paper.

The book of wisdom, Vedas is four in numbers. Namely Rig, Sama, Yajur and Atharva. Also the up Vedas are Ayurveda, Gandharvaveda, Dhanurveda and Sthapatyaveda. Sthapatya Veda deals with the planning, designing and construction of houses, villages and cities. Arithmetic and Mathematics falls under Sthapatya Veda. Vedic Mathematics is a unique technique of calculations based on simple principles and rules with which any Mathematical problem – arithmetic, algebra, geometry, menstruations, trigonometry, etc., can be solved mentally in a logical way without much usage of pen-paper. Vedic Mathematics refers to a set of 16 Sutras or formulae and 13 up sutras [9], [10], [17], [30].

This paper embodies the design and realization of a novel algorithm that can multiply each and every binary number using the principles of Yavadunam sutra and weight reduction technique. Since there is no hardware or architecture currently available for the Yavadunam sutra, we have been working on...
this and have developed the architecture for the currently available sutra [9], [10], [30]. On using the principles of Yavadunam sutra the deficiencies are computed and the same has been used to develop a high speed Vedic binary multiplier [9], [10], [30]. Bit reduction method is used with a particular intention to accomplish improvement in the speed and area of the Yavadunam multiplier suggested in our prior work [30]. The number of parts and interconnects are decreased by reducing the number of bits fed to the multiplier. The area and delay will then be decreased and the speed increases. This paper is organized in an order that the concise reviews of some of the traditional existing multipliers are explained in section.2. Section.3 exposes about the design of the modified generic architecture for Yavadunam, a squaring sutra. Section.4 deals on the design of the modified Yavadunam multiplier architecture. Section.5 deals with the comparison of the performance of the proposed Yavadunam multiplier with that of some of the traditional multipliers reviewed in section.2. Section.6 deals on the conclusion of the proposed work with the extent for the enhancement of the work in future.

2. Related Work

Multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design. Multiplications are very expensive and slow the overall operation. The performance of many computational problems is often dominated by the speed at which a multiplication operation can be executed. In general multipliers are classified into the Serial and the Parallel multiplier. The parallel multipliers are preferred over the serial multipliers due to the fact that the parallel multipliers don’t make use of combinational circuit and does not have feedback unit [22]. The parallel multipliers are classified into array and tree multipliers. The basic array multiplier works by the add and shift method. They are not much preferred as they have long critical path, less speed and occupies more space [2], [29]. The delay generated by the carry propagation in each phase is considerably high [22], [6]. This can be reduced by restoring the carry propagation adder in each phase as a replacement for the carry save adder. This multiplier is a kind of array multiplier called Braun Multiplier. The main disadvantage of this multiplier is that for higher order bits it becomes inefficient as the number of parts increases quadratically for the increase in bit size [4]. Tree multipliers are more beneficial than the array multipliers. Wallace Tree multipliers expend less power and are of high speed, despite the fact that they have abnormal structure [1], [16], [28], [24]. Another form of the tree multiplier is Dadda. As in Wallace tree multiplier the Dadda multiplier does not reduce the partial products within each of its layers. It is somewhat faster and occupies less area compared to Wallace multiplier. It requires more number of interconnects as it transfers more number of bits and the design is complex [8], [2], [1], [16]. The current bang in Vedic Mathematics led to the progress of Vedic multipliers besides the traditional multipliers. Prior they have been used to solve problems of decimal numbers in a unique manner. Later calculations of binary numbers were carried out totally based on Vedic mathematics [12], [13], [14], [7], explains on the multipliers based on Urdhava Tiryagbhyam and Nikhilam Sutra, which are generally used. Vedic multipliers are superior to the conventional multipliers as they occupy much less area and less delay [5], [13], [18], [21]. The principles of Yavadunam Sutra and weight reduction technique to design a novel high speed binary Vedic square and multiplier are stated over here.

3. Modified Yavadunam architecture for squaring

Yavadunam sutra is used in this suggested technique to compute the deficiency from the closest base value. To optimize the area and speed, the method of weight reduction is employed; thereby the N bit size is reduced to N-1 bits. Based on the values of deficiencies there are two modes of operations: Mode-1 when the deficiency is positive and Mode-2 when the deficiency is negative.

Mode: 1

When the input is greater than the nearest base value $2^{N-1}$ then the deficiency ‘D’ is positive. When the input is, $A = 1110 = 10112$ the base nearest to the input is $2^{N-1} = 2^{4-1} = 2^3 = 810 = 10002$. Now the deficiency, $D = 112$. RHS is the square of the deficiency, $D^2 = 10012$. As the square module is N-1 bit
square module only the least N-1 bits are considered as RHS i.e. 0012 and the rest of the bit 12 is fed as the carry to the LHS. The LHS is the sum of the input, deviation and the carry from the RHS i.e. 1011 +0011+1=11112. Then the output is got by concatenating the LHS and the RHS part i.e.1111 0012. Figure 1 shows the proposed square architecture for the mode-1 operation.

![Proposed Modified Square Architecture for Mode-1](image1)

**Figure 1.** Proposed Modified Square Architecture for Mode-1.

**Mode: 2**

When the input is less than the nearest base value $2^{N-1}$ then the deficiency ‘D’ is negative. When the input is, $A = 610 = 0110_2$, the base nearest to the input is $2^{N-1} = 2^{4-1} = 2^3 = 8_{10} = 1000_2$. Now the deficiency, $D = A - 2^{N-1} = 0110 -1000 = -(0010_2)$. RHS is the square of the deficiency, $D^2 = 0100_2$. As the square module is N-1 bit square module only the least N-1 bits are considered as RHS i.e. 100 and the rest of the bits 00 are fed as the carry to the LHS. The LHS is the sum of the input, deficiency and the carry from the RHS i.e. 110-0010+0 = 01002. Then the output is obtained by concatenating the LHS and the RHS part i.e. 0100 1002. Figure 2 shows the proposed square architecture for the mode-2 operation.

![Proposed Modified Square Architecture for Mode-2](image2)

**Figure 2.** Proposed Modified Square Architecture for Mode-2.

**Algorithm for the proposed square architecture**

**Step: 1** For the input greater than the base value the deficiency (D) is calculated by discarding the MSB of A. For the input lesser than the base value the deficiency (D) is computed by taking two’s complement of A and discarding the MSB.

**Step: 2** The deficiency (D) is squared by the N-1 bit squarer.

**Step: 3** The RHS is derived by considering the least N-1 bits of the square of the deficiency and the rest of the bits is fed as the carry to the LHS.

**Step: 4** The input (A) is added to the deficiency if the deficiency is positive else the deficiency is subtracted from the input if the deficiency is negative.

**Step: 5** When the deficiency is positive.
LHS = (N bits of A) + ((N-1) bits of D) + (carry bits from RHS) When the deficiency is negative and the subtractor output is positive, LHS = [(N bits of A) - ((N-1) bits of D)] - (carry bits from RHS) When the deficiency is negative and the subtractor output is negative, LHS = (carry bits from RHS) - [(N bits of A) - ((N-1) bits of D)]

The table 1 represents the two modes i.e. when the input is greater than the base and when the input is less than the base (2^N-1 for different N values). This technique is well appropriate to the problem-free, easy and simple means of finding square of any number of any operand sizes. The suggested modified square architecture is shown in Figure 3. The mixed design of the two modes is depicted in Figure 3. The deficiency is computed based on the MSB of the input. If the MSB of the input is ‘0’, the input is two's complemented and the N-1 bit is neglected else if the MSB of the input is ‘1’ only the N-1 bit is neglected. Later the N-1 bit square module squares the deficiency. The RHS part of the output will be the N-1 bits of the square module and the other bits will be the carry bits which are taken to the LHS side. If the MSB of the input is 1 then the input and the defect will be added, else if the MSB of the input is 0 then the defect is subtracted from the input ‘Y’ is the output of the adder / subtractor unit.

![Combined Architecture of the Proposed Method](image_url)

Figure 3. Combined Architecture of the Proposed Method.

The output of the adder/ subtractor is added to carry bits of the RHS side when the value of ‘Y’ is positive. Whereas when the value of the ‘Y’ is negative the value of ‘Y’ is subtracted from the carry bits of the RHS side. The square of the given input the concatenated LHS and RHS parts.
### Table 1. Examples of square of any binary number using the proposed method.

| MODE | A   | N  | $2^N-1$ | $D=A-2^N-1$ | LHS=A+D+CARRY | RHS=D² | $A^2$ |
|------|-----|----|---------|-------------|---------------|-------|-------|
| MODE:1 | 1101 | 4  | 1000    | 101         | 1101+101+11   | 10 001 | 10101001 |
|      |     |    |         |             | = 10101      |       |       |
| MODE:2 | 0100 | 4  | 1000    | -(100)      | 0100-100+10   | 10000 | 10000 |
|      |     |    |         |             | = 10000100   |       |       |
| MODE:1 | 10000 | 8  | 10000000 | 0000010    | 10000010+10   | 000100 | 1000010000100 |
|      | 010   |    |         |             | = 10000100   |       |       |
| MODE:2 | 11001 | 8  | 10000000 | -(1100111)  | 11001-1100111 | 1010010 | 1001110001 |
|      |      |    |         |             | +1010010=100  |       |       |
| MODE:1 | 10001 | 16 | 100000000 | 000000000 | 10010111100000 | 10011000 | 100010010000100 |
|      | 00010 |    |         |             | +10011000    | 00010000 | 000010000000 |
|      | 11100 |    |         |             | = 100100100001000 | 000100000000 |
| MODE:2 | 110  | 16 | 100000000 | 000000000 | - (1111111111111010) | 1100100 | 1000100 |
|      |      |    |         |             | -1111111111111010 = 0 | 110100 |       |
|      |      |    |         |             | 1111111111111010 | 000000000000 |
| MODE:1 | 10110 | 32 | 100000000 | 000000000 | 110010110100000101 | 101000010 | 1111110011100110 |
|      | 01011 |    |         |             | 1110000000000 | 10100010001001 |
|      | 01000 |    |         |             | 100100000100000111 | 110001000 |
|      | 00101 |    |         |             | 0010100001 = 01 | 10101000110001001 |
|      | 11100 |    |         |             | 111111001101100110 | 110001010 |
|      | 00000 |    |         |             | 111111001101100110 | 000100000 |
|      | 000   |    |         |             | 111111001101100110 | 000000000000 |

4. Modified Yavadunam multiplier architecture

The Bit Reduction Technique is used for the Yavadunam Multiplier with a particular objective of achieving improvement in speed and area. The number of parts used is reduced by weight reducing the input fed to the multiplier. Thereby reducing the area and delay and consequently increasing the speed. Yavadunam sutra is used in this suggested technique to calculate the defect from the closest base value. Based on the value of the input there are three modes of operations. If the two inputs are greater than $2N-1$ then it is Mode: 1, if the two inputs are lesser than $2N-1$ then it is Mode: 2, if one of the input is greater than $2N-1$ and the other is lesser than $2N-1$ then it is Mode: 3.

**Mode: 1**

When the two inputs are greater than the base value $2N-1$ (i.e.) when both the deficiencies are positive it is mode: 1. Consider A and B as $(14)_{10}$ and $(12)_{10}$ respectively. The nearest base value to these two numbers are $2^{N-1} = 2^{4-1} = 2^3 = 8$. The deficiencies of A and B are $(6)_{10} = (110)_{2}$ and $(4)_{10} = (100)_{2}$. The deficiencies are first multiplied and the product of the deficiencies $(24)_{10} = (11000)_{2}$. The multiplier is a N-1 multiplier and hence the least N-1 bits of the multiplier output (i.e.) $(000)_{2}$ will be considered as the RHS part of the output and rest of the bits $(11)_{2}$ are fed as the carry to the LHS side. The nearest base value is subtracted from the sum of A and B i.e. $14+12-8= (18)_{10} = (10010)_{2}$. The carry from the RHS part is added to the subtractor output $[10010 + 11 = (10101)_{2}]$. This is considered
as the LHS part of the output. Concatenating the RHS (000)\textsubscript{2} and the LHS part (10101)\textsubscript{2} we get the product of (1110)\textsubscript{2} and (1100)\textsubscript{2} = (10101000)\textsubscript{2}. The figure 4 depicts the Proposed Modified Multiplier Architecture for Mode-1.

**Mode: 2**

When the two inputs are lesser than the base value $2^{N-1}$ (i.e.) when both the deficiencies are negative then it is the Mode: 2 operations. Consider the two inputs A and B, 7 and 3 respectively. The nearest base value to the two inputs is $2^{N-1} = 2^4 = 16$. The deficiencies of A and B are (7 - 8 = -1) and (3 - 8 = -5) respectively. The deficiencies are first multiplied by the N-1 multiplier. The product is \([-1 \times -5 = (5)_{10}] = (-1110)\textsubscript{2}\) and the least N-1 bits ‘101’ is considered as the RHS part of the output and the remaining bit ‘0’ is fed to the LHS as the carry. The nearest base value is subtracted from the sum of A and B [(i.e.) 0111+0011 = (1000)\textsubscript{2}]. The carry from the RHS part is added to the subtrahand output and it is considered as the LHS part of the output [(i.e.) 0+0010 = (0010)\textsubscript{2}]. Concatenating the LHS and RHS part we get (10101000)\textsubscript{2} which is product of the inputs 7 and 3. Figure 5 illustrates the Proposed Modified Multiplier Architecture for Mode-2.

**Mode: 3**

When one of the two inputs is lesser than the base value $2^{N-1}$ and the other input is greater than the base value $2^{N-1}$ (i.e.) when one of the deficiencies are negative and the other deficiencies are positive then it is the Mode:3 operations. Consider the two inputs A and B, 15 and 6 respectively. The nearest base value to the two inputs is $2^{N-1} = 2^4 = 16$. The deficiencies of A and B are (15 - 8 = 7) and (6 - 8 = -2) respectively. The deficiencies are first multiplied by the N-1 multiplier. The product is \([7 \times -2 = (14)\textsubscript{10}] = (-1110)\textsubscript{2}\) and the least N-1 bits ‘110’ is considered as the RHS part of the output and the remaining bits ‘-1’ are fed to the LHS as the carry. The nearest base value is subtracted from
the sum of A and B [(i.e.) 1111+0110 = (1101)\(_2\)]. The carry from the RHS part is subtracted from the above subtractor output and it is considered as the LHS part of the output [(i.e.) 1101 - (-) = 1101-1=(1100)\(_2\) and 1100-1=(1011)\(_2\)]. As the negative sign is fed as a carry to the LHS part, the RHS value is two’s complemented (i.e.) two’s complementing ‘110’ we get ‘010’. Concatenating the LHS and RHS part we get (1011010)\(_2\) = (90)\(_{10}\) which is product of the inputs 15 and 6. Figure 7 shows the Proposed Modified Multiplier Architecture for Mode-3.

Algorithm for the proposed modified multiplier

**Step: 1** The deficiency D\(_1\) (or D\(_2\)) is computed by taking two’s complement of A (or B) and discarding the MSB if A (or B) is lesser than 2\(^{N-1}\) else the deficiency D\(_1\) (or D\(_2\)) is computed by simply discarding the MSB of A (or B).

**Step: 2** Both the deficiencies (D\(_1\) or D\(_2\)) are multiplied by the N-1 bit multiplier.

**Step: 3** The RHS of the product (AB) is derived by considering the least N-1 bits of the product of the deficiencies D\(_1\) and D\(_2\). In case of both positive and both negative deficiencies,

\[
\text{RHS} = (\text{N-1 bits of } D_1 \times D_2) 
\]

In case of mixed deficiencies the multiplier output will be a negative value and to remove the negative sign the N-1 bits of the multiplier output is two’s complemented.

\[
\text{RHS} = (\text{N-1 bits of } \text{2scomplement of } (D_1 \times D_2)) \]

**Step: 4** The multiplier outputs other than the least N-1 bits are fed as carry to the LHS part.

**Step: 5** The LHS of the product is computed by adding the inputs A and B and subtracting the base value 2\(^{N-1}\) from the sum of A and B.

**Step: 6** The carry of the RHS is added to the LHS part. The sign of the adder changes based on the given inputs. If both the inputs are greater than 2\(^{N-1}\), then

\[
\text{LHS} = [A+B - 2^N] + \text{carry bits from RHS}. 
\]

If both the inputs are lesser than 2\(^{N-1}\), then \(\text{LHS} = [A+B - 2^N] + \text{carry bits from RHS}\). If both the inputs are mixed, then \(\text{LHS} = [A+B - 2^N] - \text{carry bits from RHS}\).

Combined Architecture of the Proposed Multiplier

Figure 6 shows the proposed multiplier’s architecture. Figure 8 depicts the combined architecture of all three methods. If the inputs are lesser than the base the deficiencies are computed by two’s complementing the inputs and discarding the MSB, otherwise the deficiencies are calculated by simply discarding the MSB of the input. By feeding the deficiencies to the N-1 bit multiplier, the deficiencies are multiplied. N-1 parts of the output of the multiplier will be the RHS and the rest of the bits will be offered to the LHS side. When the two inputs are greater than the base or lesser than the base, the multiplier’s N-1 bits will be the RHS. When the inputs are mixed, the output of the
multiplier will be negative and the N-1 bits of the multiplier output will be two's complemented to remove the negative sign. In the case of the mixed inputs, the two's complemented multiplier output will be the RHS part. Add the two inputs and subtract the base value from the output of the adder. If both inputs are positive or negative, the carry from the RHS will be added to the output of the subtractor. The carry from RHS is subtracted from the subtractor output in case of mixed inputs. This will be the LHS part. The table 2 portrays the three modes i.e. when the two inputs are greater than $2^{N-1}$, when the two inputs are less than $2^{N-1}$, when one of the inputs is greater than $2^{N-1}$ and the other input is less than $2^{N-1}$ for different N values. This technique is well suited for the simple and error-free multiplication of any number of any operand sizes.

5. Performance Analysis of the proposed Yavadunam multiplier

Using the Mentor Graphics tool, the VHDL codes for the modified Yavadunam multipliers were modelled and the same was realized in Xilinx Spartan 3e FPGA. Figure 9 shows the simulated output of a proposed 32-bit Yavadunam multiplier. The proposed architecture of multipliers is synthesized using the 90 nm and 180 nm technology Synopsys tool. The area, delay, power, Area Delay Product (ADP) and Power Delay Product (PDP) of the proposed multiplier architecture is shown in the Table 3. The IC layout of the proposed hybrid Vedic multiplier is derived from RTL to GDSII format using Cadence and it is shown in figure 10.
Table 2. Examples of multiplying two binary numbers using the proposed method.

| S. NO | A     | B     | N  | 2^N-1 | MODE | D1 | D2 | LHS | RHS | PRODUCT |
|-------|-------|-------|----|-------|------|----|----|-----|-----|---------|
|       | A-2^N-1 | B-2^N-1 |    |       |      |    |    |     |     |         |
| 1     | 110   | 01    | 4  | 1000  |      |    |    | (A+B-2^N-1) + CARRY | -0 | 100 \[\text{101000}\] |
|       | 0     | 11    |    |       |      |    |    | \[A\] | 100 | \[010\] |
|       |       |       |    |       |      |    |    | -0 | 100 | \[011\] |
|       |       |       |    |       |      |    |    | +1 | 100 |         |
| 2     | 100   | 10    | 8  | 1000  |      |    |    | (A+B-2^N-1) - CARRY | 101100 | 1001101001 |
|       | 110   | 00    | 00 | 0000  |      |    |    | -0 | 100 | \[010\] |
|       | 00    | 11    |    |       |      |    |    | +1 | 100 | \[011\] |
|       |       |       |    |       |      |    |    | -0 | 100 |         |
|       |       |       |    |       |      |    |    | +1 | 100 |         |
| 3     | 100   | 10    | 1  | 1000  |      |    |    | (A+B-2^N-1) + CARRY | 1100 | 1000111110 |
|       | 000   | 00    | 6  | 0000  |      |    |    | -0 | 100 | \[010\] |
|       | 001   | 01    |    | 0000  |      |    |    | +1 | 100 | \[011\] |
|       | 110   | 10    | 00 | 0000  |      |    |    | -0 | 100 | \[010\] |
|       | 100   | 11    |    |       |      |    |    | +1 | 100 | \[011\] |
|       | 10    | 00    |    |       |      |    |    | +0 | 100 | \[000\] |
|       | 01    | 00    |    |       |      |    |    | +0 | 100 | \[000\] |
| 4     | 010   | 01    | 3  | 1000  |      |    |    | (A+B-2^N-1) + CARRY | 100000000 | 1010001110 |
|       | 110   | 10    | 2  | 0000  |      |    |    | -0 | 100 | \[010\] |
|       | 010   | 10    | 00 | 0000  |      |    |    | +1 | 100 | \[011\] |
|       | 110   | 00    | 00 | 0000  |      |    |    | -0 | 100 | \[010\] |
|       | 100   | 01    | 00 | 0000  |      |    |    | +1 | 100 | \[011\] |
|       | 000   | 00    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
|       | 000   | 10    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
|       | 000   | 00    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
|       | 000   | 01    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
|       | 000   | 10    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
|       | 000   | 00    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
|       | 000   | 01    | 00 | 0000  |      |    |    | +0 | 100 | \[000\] |
Table 3. Experimental results of the modified Yavadunam multiplier.

| WIDTH          | 90nm TECHNOLOGY | 180nm TECHNOLOGY |
|----------------|-----------------|-------------------|
|                | 4 BIT           | 8 BIT            | 16 BIT         | 32 BIT         | 4 BIT    | 8 BIT    | 16 BIT         | 32 BIT         |
| DELAY(µs)      | 2595            | 2982             | 6104           | 10514          | 2958     | 4089     | 8628           | 14989          |
| AREA(µm²)      | 971             | 3171             | 13512          | 59654          | 2792     | 9780     | 42084          | 156847         |
| POWER(µW)      | 20.63           | 113.76           | 758.07         | 4102.38        | 94.57    | 518.23   | 3554.27        | 1943.05        |
| ADP(pSm²)      | 2.519           | 9.45             | 82.47          | 627.20         | 8.25     | 39.9     | 363.10         | 2350.97        |
| PDP(pSW)       | 0.053           | 0.339            | 4.627          | 43.13          | 0.27     | 2.11     | 30.66          | 29.12          |

Compared to conventional multipliers like Array [6], Shift and Add [19], Braun [4], Dadda [8], Wallace [16] and the Vedic multipliers like Urdhava [5] and Nikhilam [15], the performance of the proposed multiplier is evaluated. Array Multiplier is less economical as this multiplier delay is higher and requires more gates leading to increase in the area. Wallace tree multiplier design is quite hard but owing to its irregular structure it has high speed operation. Braun multiplier is far less beneficial as the delay is greater and as the number of components increases with the number of bits, the multiplier becomes inefficient incase of higher order bits. The efficiency of the shift and add multiplier depends on the clocks speed and it consumes more energy owing to high switching operation. For numbers of lower order bits Urdhava Tiryagbhayam is effective but the delay increases for numbers of higher operand size. Any delay-based compensation increases the area. Because of its constraints in the range of inputs, Nikhilam Navatashcaramam Dashatah multiplier is absolutely inefficient. In case of the lower-order bits Urdhava shows the improved performance. When compared to the other multipliers, the modified Yavadunam produces less delay incase of higher order bits. The output of the array and Braun multiplier is similar in case of area. Comparing with the other multipliers Wallace, Dadda and Urdhava are superior. The interpretation demonstrates that Wallace, Dadda, Urdhava, Array and Braun multipliers are advantageous over the other multipliers for designs of lower order bit. In the meantime, the modified Yavadunam gains more area saving for the bit sizes beyond 8 bits. Figure 10 and Figure 11 the graphical analysis of delay and area comparison of different traditional multipliers with that of the proposed multiplier for 180 nm technology multiplier respectively.

In case of power consumption Wallace, Dadda and Urdhava outperform the other multipliers. The analysis shows that for designs of lower order bit Wallace, Dadda and Urdhava multipliers are advantageous over the other multipliers. Urdhava multiplier is most advantageous than any other multipliers. Furthermore the modified Yavadunam produces more energy savings for the bit sizes beyond 8 bits Table 4 illustrates the leakage power, dynamic power and total power consumed by required 16 bit and 32 bit multipliers for 180 nm technology. Figure 12 shows the graphical representation for 180 nm technology of the detailed power analysis of different 32 bits multipliers.
Figure 10 Delay comparison of various multipliers for 180nm technology.

Figure 11. Area comparison of various multipliers for 180nm technology.

Table 4. Detailed Power analysis of various multipliers for 180nm Technology.

| Methods       | 16 Bit          | 32 Bit          |
|---------------|-----------------|-----------------|
|               | LP (nW)         | DP (nW)         | TP(nW) |
| Array         | 1498.83         | 3225490.493     | 3226989.306 |
| Braun         | 1498.813        | 313028.708      | 3134527.521 |
| Shift and add | 96572.023       | 12031383.266    | 4477597.992 |
| Wallace       | 1504.624        | 3560456.495     | 3561961.119 |
| Dadda         | 1437.605        | 3561339.265     | 3562776.869 |
| Urdhava       | 1631.327        | 2838220.367     | 2839851.693 |
| Nikhilam      | 1598.251        | 3958312.547     | 3959910.798 |
| Proposed      | 1364.72         | 3552911.364     | 3554275.08 |
| Modified Yavadunam | 1936911.021    | 1943055.993     | 1943055.993 |
6. Conclusion and scope of future work

The proposed modified Yavadunam multiplier is compared to some of the conventional and Vedic multipliers such as Array, Braun, Shift and Add, Wallace, Dadda, Urdhava and Nikhilam. The proposed Yavadunam multiplier is beneficial at greater order bits in area cutback and not more effective at lower order bits. For lower order bits, the proposed multiplier is not beneficial because they get through more power; indeed, they are excellent for higher order bits. The proposed multiplier’s leakage power is much less than the other multipliers. The proposed multiplier is thus only effective for higher order bits from 32 bits onwards. Since the proposed method is not much advantageous in case of area and consumes more power, their power delay product and the area delay product is optimized due to the decrease in delay. The proposed modified Yavadunam multiplier is advantageous of all the multipliers. In this work, the N bit multiplier is reduced to N-1 bit to optimize the speed and the area. The speed, area and power can still be optimized by further reducing the bit size.

References

[1] Abhilash R, Raju IBK, Chary G and Dubey S 2015 Area-Power Efficient Vedic Multiplier Using Compressors International Conference On Electrical, Electronics, Signals, Comm. Optim.1-5.

[2] Addanki Purna Ramesh 2011 Implementation of Dadda and Array multiplier Architecture using Tanner Tool IJCSET 2 28-41

[3] Akanksha Kant and Shobha Shama 2015 Applications of Vedic Multiplier Designs-A Review In proceedings of the 4th Int. Conf on Reliability, Infocom tech and optimization IEEE 1-6.

[4] Anitha R, Nelapati A, Lincy Jesima W and Bagyaveereswaran V 2012 Comparative Study of High performance Braun’s Multiplier using FPGAs IOSR J. Electron. Comm. Eng. 1 33-37.

[5] Arish S and Sharma R K 2015 An Efficient Binary Multiplier Design For High Speed

[6] Applications Using Karatsuba Algorithm and Urdhva-Tiryagbhyam Algorithm In Proceedings of the 2015 Global Conference On Communication Technologies (GCCT) IEEE 192-196.

[7] Arushi Somani, Dheeraj Jain, Sanjay Jaiswal and Kumkum Verma Swati Kasht 2012 Compare Vedic Multipliers with Conventional Hierarchical Array of Array Multiplier International Journal of Computer Technology and Electronics Engineering (IJCTEE) 2 52-55.

[8] Bengali SS. 2011 Vedic Mathematics and Its Applications in Computer Arithmetic Thesis M.Sc
Computer Science North Carolina state Univ Raleigh.

[9] Dadda L 1965 Some Schemes for Parallel multipliers Alta Frequenza 34 349-356.

[10] Deepa A and Marimuthu CN 2018 High Speed VLSI Architecture for Squaring Binary Numbers Using Yavadunam Sutra and Bit Reduction Technique in International Journal of Applied Engineering Research 13 4471-4474.

[11] Deepa A and Marimuthu CN 2017 A Modified Multiplier Architecture Using Yavadunam - A Squaring Algorithm of Vedic Mathematics in Perspectivas em ciencia da Informacao 22 322-332.

[12] Ganesh Kumar G, Venkata Sudhakar C and Naresh Babu M 2013 Design of High Speed Vedic Square by using Vedic Multiplication Techniques International Journal of Scientific and Engineering Research 04.

[13] Ghosh M 2007 Design and Implementation of Different Multipliers Using VHDL Thesis National Institute of Technology Rourkela.

[14] Goyal H and Akhter S 2015 An Advancement In The N×N Multiplier Architecture Realization Via the Ancient Indian Vedic Mathematic Int. J. Comput. Appl 127 24-27.

[15] Gupta A, Malviya U and Kapre V 2012 A Novel Approach to Design High Speed Arithmetic Logic Unit Based On Ancient Vedic Multiplication Technique Int. J. Mod. Eng. Res. 2 2695-2698.

[16] Harish Kumar Ch 2013 Implementation and Analysis of Power, Area and delay of Array, Urdhva, Nikilam Vedic Multipliers Int. J. of Scientific and Research Publications 3 1-5.

[17] Hussain and R K Sah 2015 Performance Comparision of Wallace Multiplier Architectures Int. Journal of Innovative Research in Science, Engineering and Technology 4 2347-6710.

[18] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaj 2009 Vedic Mathematics Motilal Banarsidas Publishers Pvt.Ltd Delhi.

[19] Jinesh S, Ramesh P and Josmin Thomas 2015 Implementation Of 64bit High Speed Multipliers For DSP Application-Based On Vedic Mathematics In Proceeding TENCON 2015 IEEE Region 10 Conference 1-5.

[20] Mottaghi-Dastjerdi M, Afzali-Kusha A and Pedram M 2014 BZ-FAD: A Low-Power Low- Area Multiplier Based on Shift-and-Add Architecture IEEE Trans On Very Large Scale Integration (VLSI) Syst. 17 267 386.

[21] Neeraj Mishra and Asmita Haveliya 2013 An Advancement In The N×N Multiplier Architecture Realization via the Ancient Indian Vedic Mathematics International Journal Of Electronics Communication and Computer Engineering (IJECCE) 4 544-548.

[22] Priyanka Brahmaiah V, Dhrma Teja L and Dr.Y.Padma Sai 2013 Study On Comparison Of Various Multipliers International Journal of Electronics and Communication Engineering And Technology 4 132-142.

[23] Richa S, Manjit K and Gurmohan S 2015 Design and FPGA Implementation of Optimized 32-Bit Vedic Multiplier and Square Architectures IEEE (ICIC) 960-964.
[24] Siridhara A L, Mahendra V and Ravinder T 2017 Performance Evaluation of Parallel Multipliers

[25] Journal of Engineering and Applied Sciences 12 5186-5189.

[26] Siriraman L, Saravana Kumar K and Prabakar T N 2013 Design and FPGA implementation of Binary Squarer using Vedic Mathematics IEEE (ICCCNT) 4-6.

[27] Vaijyanath K, Linganagouda K and Subhash K 2013 Low Power Square and Cube Architectures using Vedic Sutras IEEE (ICSIP) 354-358.

[28] Vidyashankari P and Lokesha B 2015 Design and Implementation of Square and Cube Architectures using Vedic Sutras on FPGA International Journal of Emerging Technology in Computer Science and Electronics (IJETCSE) 14 377-381.

[29] Wallace CS 1964 A Suggestion for a fast multiplier IEEE Trans Electron Comput EC 13 14-17.

[30] Wang J S, Kuo C N and Yang T H 2004 Low power fixed – width array multipliers In Proc. IEEE Symp. Low power Electron Des. 307 – 312.

[31] Deepa A and Marimuthu C N 2019 Design of High speed Vedic Square and Multiplier Architecture based on Yavadunam Sutra paper accepted and published online in Sadhana - Academy Proceedings in Engineering Sciences, http://doi.org/10.1007/s/12046-019-1180-3.