Design of a high speed Vedic multiplier and square architecture based on *Yavadunam Sutra*

A DEEPA^{1,*} and C N MARIMUTHU²

¹Department of Electronics and Communication Engineering, J.K.K. Munirajah College of Technology, Gobi, Erode, India

²Department of Electronics and Communication Engineering, Nandha Engineering College, Erode, India
e-mail: adeepaeaswaramoorthy@gmail.com; muthu_me2005@yahoo.co.in

MS received 15 June 2019; accepted 13 July 2019

Abstract. In current day situation we come across numerous mathematical challenges. This could be overwhelmed by the Vedic Mathematics. Vedic Mathematics is an ancient approach to solve problems in a rapid manner. In this paper the design of a novel Vedic square and multiplier architecture based on *Yavadunam Sutra* is proposed. Yavadunam is a squaring sutra of the Vedic Mathematics. We have designed a generic architecture for this squaring sutra and have designed a high speed Vedic binary multiplier architecture using the principles of Yavadunam sutra. The proposed multiplier offers significant improvement in speed. Xilinx Spartan FPGA is used to design and realize the architecture and the Synopsys device with 90 nm and 180 nm technology is used to synthesize the same.

Keywords. Vedic mathematics; Vedic multiplier; Vedic squaring; *Yavadunam Sutra*.

1. Introduction

Rapidly growing era has raised needs for quick and effective real time Digital signal processing programs like convolution, Fast Fourier Transform, Multiply and accumulate, and so on [1, 2]. With the advancement in technology, numerous researches are creating a wonderful attempt to design multipliers [3]. In case of the multiplication based functions the execution time distinctly depends on the rate of operation of the multiplier unit. Multiplication consumes more time when compared with the other basic operations in several DSP algorithms, so the critical path delay for the entire operation is decided with the aid of the delay required for the multiplication unit and it validates the performance of the algorithm. Moreover, the speed of the processor is predicted in phrases of the number of multiplications it can deal with in unit time. Therefore pace of the multiplier unit greatly affects the performance of a processor [4]. Multipliers reveal their applications in exclusive areas of engineering and technology like special effects, measurement and instrumentation, image enhancement, audio and video processing, Navigation, robotics, communications, animations and so on [5]. Henceforth numerous analysis are being completed in multiplier designs which are of rapid, low power utilization, less area with the goal that they are apt for different high speed, compact and low power applications [6]. Speed

upgradation and delay reduction in a multiplier is a noteworthy piece of satisfying the general outline. Multipliers operate at high system clock rate and are quite complicated circuits. The addition and multiplication oversees execution time even though they are the maximum often used binary mathematics operations. As of now we like to handle with a compact, handy and light weight gadgets. Consequently the principle aim of designing a circuit is to enhance the multipliers speed [7–9].

The word ‘Vedic’ in Vedic Mathematics means unrestricted storehouse of all knowledge [10]. When compared to the modern mathematics this Vedic Mathematics makes the calculations simple and easy. This ancient Indian Mathematics was revitalized by SriBharati Krishna Tirthaji Maharaj from Atharva Veda. The Vedic Mathematics engages precise sets of formulae or guidelines known as Sutras and up-sutras. Tirthaji created strategies and procedures for fortifying the standards enclosed in 16 sutras and 13 up-sutras and entitled it as Vedic Mathematics [10]. Some branches of Mathematics like calculus, trigonometry, and applied mathematics use these techniques of Vedic Mathematics [10]. This paper exemplifies the design and implementation of a high speed square and multiplier architectures. Most research has been done in Nikhilam and Urdhava sutra for designing multipliers. Yavadunam sutra does not possess any hardware or architecture till now. This made us to work on this squaring sutra and design architecture for the existing sutra [11, 12]. Here the deficiencies are obtained by using the principles of Yavadunam sutra

*For correspondence
Published online: 19 August 2019

Table 1. Examples of squaring decimal numbers based on Yavadunam Sutra of Vedic Mathematics.

Sl.NO.	Number to be squared (A)	Nearest base (B)	Deficiency (D)	LHS (A+D)	RHS (D ²)	A ² = LHS and RHS
1	9	10	$9 = 10 - 1, D = -1$	$9 - 1 = 8$	1	81
2	14	10	$14 = 10 + 4, D = 4$	$14 + 4 = 18$	16	196
3	97	100	$97 = 100 - 3, D = -3$	$97 - 3 = 94$	09	9409
4	104	100	$104 = 100 + 4, D = 4$	$104 + 4 = 108$	16	10816
5	998	1000	$998 = 1000 - 2, D = -2$	$998 - 2 = 996$	004	996004
6	1005	1000	$1005 = 1000 + 5, D = 5$	$1005 + 5 = 1010$	025	1010025

and the same have been implemented and high speed binary multiplier is designed. The proposed multiplier reduces layout complexity and can multiply binary numbers of any range. The organization of this paper is follows: section 2 explains in brief about the several traditional multipliers and the same has been compared with the performance of the proposed multiplier in section 5. Section 3 describes about Yavadunam, a squaring sutra with the help of which a generic architecture for a squaring unit is designed. Section 4 explains about the design of the proposed multiplier architecture using the principles of Yavadunam Sutra. Section 5 and 6 deals with the design of the proposed Yavadunam Vedic multiplier architecture, Section 7 analyses and compares the performances of the traditional multipliers with the proposed multiplier. In section 8, conclusions are provided with the scope for the future enhancement.

2. Related work

In general humans perform the mathematical calculations in a serial manner, while modifying it to the field of computation it very well suits for parallel processing. This segment gives a brief explanation about some of the traditional parallel unsigned multipliers. Wang *et al* [13] proposed a multiplier circuit based on the Add and Shift algorithm. Although this proposed array multiplier is easy to design, it is very slow because of its long critical path and it requires less area. Wallace [14] and Hussain *et al* [15] designed a multiplier using the Wallace tree architecture. Due to consumption of less power and high speed, it is more advantageous over the other multiplier architectures. Dadda [16] and Addanki Purna Ramesh [17] proposed a multiplier which is similar to Wallace multiplier but slightly faster and requires less area compared to Wallace multiplier. As there is transfer of more bits, it requires more interconnections and the design is complex. Anitha *et al* [18] designed a multiplier architecture which is a type of array multiplier. In case of higher order bits the number of components increases which leads to increase in area and makes the multiplier insufficient. Arish *et al* [19] proposed a finest algorithm for binary multiplication in terms of area and delay. But for higher order bits the area increases with increase in number of bits. Harish Kumar

[20] explains an architecture that is simple and faster than Urdhava but it is a special multiplier which can multiply the numbers that are only closer to the powers of 10.

3. Vedic squaring – Yavadunam

This segment brings in squaring operation using Vedic Methodology. Squaring is an exclusive case of multiplication [11, 12]. Yavadunam sutra has been used to square a number; it means that, “determine deficiency, lessen the deficiency from that number and write the square of the deficiency” [10–12, 21]. *Yavadunam Tavadunikritya Vargancha Yojayet* (YTVY) is up-sutra of the sutra Yavadunam. The specific condition of this formula is that it can be very effectively applied to find square of any number only if it is closer to the bases of powers of 10 i.e., 10, 100, 1000, 1000, The following are the steps to be followed to find the square of a number using Yavadunam sutra of the Vedic Mathematics [10–12].

Step 1: Find the deficiency with the nearest base.

Step 2: square the deficiency and place at the right side.

Step 3: Add or subtract the Deficiency from the number.

Step 4: Result = [Number – Deficiency] + carry over] & [Square of Deficiency].

Deficiency is the difference between the input to be squared and the base. For a number ‘8’ the nearest base of powers of 10 is ‘10’. Now the deficiency of 8 is $10 - 8 = 2$. For a number ‘996’ the nearest base of powers of 10 is ‘1000’. Now the deficiency of 996 is $1000 - 996 = 4$. The table 1 illustrates some of the examples of calculating square of binary numbers using the squaring sutra Yavadunam. Yavadunam is a precise method to square a number.

But this sutra is efficient if and only if the number to be squared is closer to the powers of 10. And due to this reason there is no existing standard architecture for Yavadunam sutra. As the sources of input are from a variety of ranges it is not an easy task to design a particular architecture.

4. Proposed square architecture

By extending the same view for binary this problem could be overcome. As per the sutra the deficiency is calculated from the nearest base of powers of 10 i.e., 10^N , similarly

Table 2. Examples of squaring binary numbers using the proposed method.

Sl.N O.	Number to be squared (A)	N	Base (2^N)	Deficiency (D = BASE - A)	LHS A-D + carry	RHS (D ²)	A ² = LHS and RHS
1	11	2	100	100 - 11 = 1	11 - 1 = 10	01	1001
2	1001	3	1000	1000 - 1001 = -1	1001 + 1 = 1010	001	1010 001
3	0010	4	10000	10000 - 10 = 1110	10-1110 = -1100+ 1100 ← =0	1100 0100	0000 0100
4	10000	5	100000	100000 - 10000 = 10000	10000 - 10000 = 0 + 1000 ← =1000	1000 00000	100000000
5	101110100 001100	16	1000000 0000000 000	1000000000000000- 0- 101110100001100 =10000101110100	101110100001100- 10000101110100= 111101000011000 +1000110000010= 1000101110011010	1000110000010 1011100010010 000	10001011100 11010101110 0010010000
6	1011001011 010	13	1000000 0000000	10000000000000- 1011001011010=10 0110100110	1011001011010- 100110100110+ 1011101000 ← 111110011100	1011101000 1011110100100	11111001110 01011110100 100

extending the same view for binary the base will be the powers of 2 i.e., 2^N , where N is the number of bits in the input. Table 2 illustrates some of the examples of calculating square of binary numbers using the proposed method. By this proposed method any number of any ranges can be squared precisely in a faster way. Based on the MSB of the input there are two modes to square any binary input. They are:

1. when the given input is greater than 2^{N-1}
2. when the given input is less than 2^{N-1}

4.1 Algorithm

Step 1: The deficiency (D) is computed by taking two's complement of 'A'.

Step 2: The deficiency (D) is squared by the N bit square unit.

Step 3: The RHS is derived by considering the least N bits of the square of the deficiency (D) and the rest of the bits are fed as carry to the LHS.

Step 4: The deficiency (D) is subtracted from the input (A)

Step 5: The output of the subtractor is added or subtracted to the extra bits fed as carry from the RHS based on the value of the input. If the input is greater than 2^{N-1} then,

$$\text{LHS} = [A-D] + [N \text{ bits from RHS}]$$

If the input is less than 2^{N-1} then,

$$\text{LHS} = [A-D] - [N \text{ bits from RHS}].$$

Figure 1 shows the combined architecture of the two modes. The input is two's complemented to obtain the deficiency. The N bit square module squares the deficiency. The least N bits of the squared output are the RHS. Rest of

the bits is fed as the carry to the LHS side. Subtractor subtracts the deficiency from the input. The subtractor output and the carry of the squared deficiency are summed up if the MSB of the number to be squared is 1 and this is the LHS, else subtract the subtractor output from the carry of the squared deficiency this is the LHS. Concatenate LHS and RHS and the concatenated output is the square of the given number.

5. Vedic multiplier – Yavadunam

As per the previous section $A \cdot A = [(A-D) \pm \text{carry}] \& D^2$. Squaring is a special case of multiplication. Hence a square architecture can be employed for multiplication operation. This segment conveys the multiplication operation using the Vedic sutra Yavadunam with examples and at last exemplifies the architecture of the n-bit multiplier. The principles of Yavadunam sutra is implemented to obtain the deficiency from the nearest base value. The following are the steps to be pursued to multiply two numbers using Yavadunam sutra [11, 12]. Let A and B be the two numbers to be multiplied.

Step 1: Find the deficiencies of the inputs A and B with the base value.

Step 2: The deficiencies of A and B are multiplied at the N bit multiplier and place the result at the right hand side.

Step 3: Add A and B. Subtract base value from the sum A and B, and place it at the left hand side.

Step 4: Result = $[(A+B) - \text{base}] + \text{carry}$ & [product of deficiencies].

Deficiencies are computed by two's complementing the input values. Based on the values of the deficiencies there are three modes of operations. If both the deficiencies are

Figure 1. Architecture of the proposed squaring circuit.

positive then it is Mode 1. If both the deficiencies are negative it is Mode 2. If the deficiencies are mixed i.e., one of the deficiencies is positive and the other is negative then

it is Mode 3. Table 3 illustrates some of the examples of calculating the product of two decimal numbers using the Vedic sutra Yavadunam. In example 3 the product of the two deficiencies is -40 . As N here is equal to 1 the least N bit of the product will be the RHS part and -4 will be fed to the LHS side. In example 6 the product of the two deficiencies is -30 . The negative sign is fed to the LHS side and the base value 100 is added to the RHS -30 to remove the negative sign and now the RHS is $100-30 = 70$. From the above examples it is implicit that in Mode1 both the deficiencies are positive, in Mode2 both the deficiencies are negative, in Mode 3 one of the deficiencies will be positive and the other will be negative. Incase of the mixed inputs, the product of the deficiencies in the RHS side will be negative. This negative sign is fed as the carry to the LHS side. This method will be successful only for the numbers that are closer to the base values.

6. Proposed Vedic multiplier architecture

By extending the same ideas for binary any, number can be multiplied using the Yavadunam sutra. In the binary implementation the base value will be the powers of 2 (i.e., 2^N), where N is the number of bits in the input. By this method any number of any ranges can be multiplied in a rapid way. In binary implementation the deficiency is

Table 3. Examples of multiplying two decimal numbers using the Yavadunam sutra of Vedic Mathematics.

Sl.N O.	MODES	A	B	BASE VALUE 10^N	D_1 $A-10^N$	D_2 $B-10^N$	LHS $(A+B)-10^N + \text{CARRY}$	RHS $D_1 * D_2$	AB LHS and RHS
1	MODE:1	12	13	10	$12-10 = 2$	$13-10 = 3$	$(12+13)-10=25-10 = 15$	$2*3 = 6$	156
2	MODE:2	9	4	10	$9-10 = -1$	$4-10 = -6$	$(9+4)-10=13-10 = 3$	$-1*-6 = 6$	36
3	MODE:3	15	2	10	$15-10 = 5$	$2-10 = -8$	$(15+2)-10=17-10 = 7-4 = 3$	$5*-8 = -40$ $=0$	30
4	MODE:1	104	112	100	$104-100 = 4$	$112-100 = 12$	$(104+112)-100 = 116$	$4*12 = 48$	11648
5	MODE:2	95	92	100	$95-100 = -5$	$92-100 = -8$	$(95+92)-100 = 187-100 = 87$	$-5*-8 = 40$	8740
6	MODE:3	110	97	100	$110-100 = 10$	$97-100 = -3$	$(110+97)-100 = 107-1 = 106$	$10*-3 = -30+100 = 70$	10670
7	MODE:1	1004	1009	1000	$1004-1000 = 4$	$1009-1000 = 9$	$(1004+1009)-1000 = 1013$	$4*9 = 36 = 036$	1013036
8	MODE:2	995	998	1000	$995-1000 = -5$	$998-1000 = -2$	$(995+998)-1000 = 993$	$-5*-2 = 10 = 010$	993010
9	MODE:3	998	1010	1000	$998-1000 = -2$	$1010-1000 = 10$	$(998+1010)-1000 = 1008-1 = 1007$	$-2*10 = -20+1000 = 980$	1007980

Figure 2. Proposed multiplier architecture.

computed based on the principles of Yavadunam sutra i.e., (D = the nearest base value of powers of 2 – the input) and the same is attained by taking the two's complement of the input. There are three modes of operations based on the input values A and B . If both A and B is greater than 2^{N-1} then it is Mode 1. If both A and B is less than 2^{N-1} then it is Mode 2. If the inputs are mixed then it is Mode 3.

Mixed inputs means if anyone of the inputs is greater than 2^{N-1} and the other is less than 2^{N-1} . The following are the algorithm for the proposed method for all the three modes. The multiplier used in the RHS side is an N bit multiplier hence the product of the deficiencies should be of N bit.

6.1 Algorithm

Step: 1 The deficiency D_1 (or D_2) is computed by taking two's complement of A (or B).

Step: 2 Both the deficiencies (D_1 or D_2) are multiplied using the N bit multiplier.

Step: 3 The RHS of the product (AB) is derived by considering the least N bits of the product of the deficiencies D_1 and D_2 .

Step: 4 MSB N bits of the multiplier is taken from the RHS

and it is added with deficiencies derived in Step 1.

Step: 5 The LHS of the product is computed by adding the deficiencies D_1 and D_2 . Depending on the value of the input the adder output is taken as such or two's complemented.

Step: 6 The carry of the RHS is added. The sign of the adder changes based on the given inputs.

If both the inputs are greater than 2^{N-1} ,
Then $LHS = [2^N - (D_1 + D_2)] + \text{carry bits from RHS}$
If both the inputs are lesser than 2^{N-1} ,
then $LHS = (D_1 + D_2) - \text{carry bits from RHS}$.
If both the inputs are mixed,
then $LHS = [2^N - (D_1 + D_2)] + \text{carry bits from RHS}$

Figure 2 depicts the architecture of the proposed multiplier. The combined architecture of all the three modes is portrayed in figure 2. The deficiencies are obtained by two's complementing the inputs. The deficiencies are multiplied by feeding them to the N bit multiplier. N bits of the multiplier output will be the RHS and the rest is fed as carry to the LHS side. When both the inputs are negative the two deficiencies will be added and the carry from RHS will be subtracted from the adder output. This will be the LHS part. When both the inputs are positive or when the inputs are mixed the two deficiencies are added and the adder output is two's complemented and summed up with the carry from the RHS. Table 4 illustrates some of the examples of calculating the product of two binary numbers using the proposed method. The table 4 exemplifies all the three modes. This method is well suitable to multiply any number of any bit size in a simple, easy and error free manner.

7. Result and discussion

The VHDL codes for the proposed Yavadunam multipliers were simulated using Mentor Graphics tool and the same were realized in Xilinx Spartan 3e FPGA. The proposed architectures are synthesized using the Synopsys device of

Figure 3. Simulation output of the proposed 32 bit Yavadunam multiplier.

90 nm and 180 nm technology. The area, delay, power, ADP and PDP of the proposed multiplier architecture is shown in table 5. The simulated output of the proposed 32 bit Vedic multiplier architecture using Yavadunam sutra is shown in figure 3.

The delay, power and area of the proposed Yavadunam multiplier are compared with some of the existing conventional and Vedic multipliers. Several multipliers of standard bit sizes 4, 8, 16, and 32 are compared. The performance of the proposed multiplier is compared with the

Table 4. Examples of multiplying two binary numbers using the proposed method.

S.N O	A	B	N	2 ^N	2 ^{N-1}	MODE	D ₁	D ₂	D ₁ + D ₂	D ₁ * D ₂	LHS	RHS D ₁ *D ₂	PRODUCT AB
1	100 1	010 0	4	1000 0	100 0	2 ^{N-1} < A 2 ^{N-1} > B MIXED	111	110 0	100 11	101 010 0	[2 ^N - (D ₁ +D ₂)]+CARRY -011+101 =10	101 0100 ↙ ↘ =0100	100100
2	100 1	110 1	1 6	1000 0000 0000 0000 0	100 000 000 000 0	2 ^{N-1} > A 2 ^{N-1} > B BOTH NEGATIVE	111 111 111 111 011 1	111 111 111 111 001 1	111 111 111 111 010 10	111 111 111 111 010 100 000 000 001 110 101	(D ₁ +D ₂) – CARRY 11111111111101010 - 11111111111101010 = 0	11111111 11110101 0 00000000 01110101 = 00000000 01110101	1110101
3	101 100 101 101 000 001 011 110 000 000 00	110 100 001 001 110 111 000 011 000 000 000	3 2	1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0	100 000 000 000 000 000 000 000 000 000 00	2 ^{N-1} < A 2 ^{N-1} < B BOTH POSITIVE	010 011 010 010 111 110 100 010 000 000 00	001 011 110 110 001 000 000 101 011 000 00	011 111 001 001 001 010 111 010 011 101 000 00 00	[2 ^N - (D ₁ +D ₂)]+CARRY 01000001101101110 0010000100000000 + 11100100100101011 10101011110 = 10010001101101110 111111001011110	11100100 10010101 11010101 1110 01011101 10011010 00000000 00000000 00000000 = 01011101 10011010 00000000 00000000	100100011 011011101 111110010 111100101 110110011 010000000 000000000 0	

Table 5. Experimental results of the proposed multiplier.

WIDTH	90 nm TECHNOLOGY				180 nm TECHNOLOGY			
	4 BIT	8 BIT	16 BIT	32 BITS	4 BIT	8 BIT	16 BIT	32 BIT
DELAY (Ps)	2592	2987	6120	10528	2951	4097	8654	15009
AREA (μm^2)	972	3175	13526	59684	2799	9784	42106	156871
POWER (μW)	20.64	113.82	758.15	4102.39	94.60	518.29	3554.35	1943.16
ADP (pSm^2)	2.519	9.48	82.78	628.35	8.26	40.08	364.39	2354.48
PDP (pSW)	0.053	0.339	4.639	43.19	0.26	2.16	31.38	339.12

Table 6. Cell Comparision of various multipliers.

Methods	90 nm, 180 nm Technology			
	4 Bit	8 Bit	16 Bit	32 Bit
Array	100	456	1936	7968
Braun	100	456	1936	7968
Shift-and-add	217	1303	6886	33264
Wallace	67	426	1927	8056
Dadda	85	421	1856	7842
Urdhava	85	453	2045	8653
Nikhilam	92	448	1958	8014
Proposed Yavadunam	92	429	1711	7746

conventional multipliers like Array [13], Shift and Add [21], Braun [18], Dadda [16], Wallace [14] and the Vedic multipliers like Urdhava [19] and Nikhilam [20]. Array multiplier is less economical as the delay of this multiplier is larger and uses more gates which increase the area. Wallace tree multiplier circuit layout is fairly difficult due to its irregular structure but possesses high speed operation. Braun multiplier is less advantageous as the delay is larger and the multiplier goes inefficient incase of higher order bits as the number of components increases with the increase in operand size. The performance of the shift and Add multiplier rely on the clock speed and due to high switching activity it consumes more power. Urdhava Tiryaqbhyam is an efficient multiplier for lower order bits but incase of higher order bits delay increases. Any

compensation in terms of delay increases the area. Nikhilam Navatashcaramam Dashatah multiplier is practically inefficient due to its limitations in the input range. Table 6 depicts the cell comparison of various multipliers for 90 nm and 180 nm technology.

Shift and Add multiplier design uses more number of cells. Dadda and the proposed Yavadunam require considerably less number of cells. The proposed Yavadunam multiplier is better amongst the three. For lower order bits Wallace is advantageous whereas incase of higher order bits the proposed multiplier is advantageous.

Table 7 shows the analysis of the area comparison of various multipliers.

Array and Braun multiplier performance are alike incase of area. Wallace, Dadda and Urdhava outperform the other multipliers. The analysis shows that for designs of lower order bit Wallace, Dadda, Urdhava, Array and Braun multipliers are advantageous over the other multipliers. Meanwhile the proposed Yavadunam yields more area saving for the bit sizes beyond 4 bit.

Table 8 shows the analysis of the delay comparison of various multipliers. Of all the multipliers the array multiplier is simple and linear but the carry propagation delay for the array multiplier is more when compared with the delay of other multipliers considered. Among all the multipliers considered, Vedic multipliers have minimum delay comparing with that of the conventional multipliers. It is clarified that the proposed Yavadunam is better than Urdhava and Dadda in terms of delay for the operand sizes beyond 4 bit.

Table 7. Area comparison of various multipliers.

Methods	90 nm				180 nm			
	4 Bit	8 Bit	16 Bit	32 Bit	4 Bit	8 Bit	16 Bit	32 Bit
Array	763	3479	14770	60789	2328	10616	45072	185502
Braun	763	3479	14770	60789	2328	10616	45072	185502
SAA	1732	9868	50311	238649	5285	30114	153526	728248
Wallace	511	3250	14701	61461	1560	9918	44862	187550
Dadda	648	3212	14160	60856	1979	9801	43209	165232
Urdhava	648	3456	15602	66015	1979	10546	47609	201449
Nikhilam	981	4502	27140	79671	2851	10854	54638	178462
Proposed Yavadunam	972	3175	13526	59684	2799	9784	42106	156871

Table 8. Delay comparison of various multipliers.

Methods	Delay in pS							
	90 nm				180 nm			
	4 Bit	8 Bit	16 Bit	32 Bit	4 Bit	8 Bit	16 Bit	32 Bit
Array	2142	4770	10418	21477	2922	6547	14236	29458
Braun	1513	3476	8432	15230	2075	4756	10742	20822
Shift-and-add	2122	5359	12358	27255	2921	7388	17110	37885
Wallace	1933	3058	6547	13241	2623	4200	8952	18250
Dadda	1306	2992	6345	11452	1810	4138	8702	16463
Urdhava	2160	5683	12349	24651	2958	4126	7463	15436
Nikhilam	2354	3851	7421	12354	2859	5234	10068	27032
Proposed Yavadunam	2592	2987	6120	10528	2951	4097	8654	15009

Table 9. Detailed Power comparison of various multipliers for 180 nm Technology.

Methods	16 Bit			32 Bit		
	LP (nW)	DP (nW)	TP (nW)	LP (nW)	DP (nW)	TP (nW)
Array	1498.83	3225490.493	3226989.306	6177.326	26939444.538	26945621.864
Braun	1498.813	3133028.708	3134527.521	6177.326	25924677.924	25930855.250
Shift-and-add	96572.023	12031383.266	4477597.992	449552.135	8487733.793	31260992.528
Wallace	1504.624	3560456.495	3561961.119	6286.087	18926034.961	18932331.048
Dadda	1437.605	3561339.265	3562776.869	6268.547	19562155.04	19568423.587
Urdhava	1631.327	2838220.367	2839851.693	6913.595	15218550.081	15225463.676
Nikhilam	1598.251	3958312.547	3959910.798	6285.213	27586942.654	2764979.867
Proposed Yavadunam	1392.16	3552961.521	3554353.68	6197.658	1936957.478	1943155.136

Table 9 depicts the leakage power, dynamic power and total power consumed by various 16 bit and 32 bit multipliers for 180 nm technology. 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. Meanwhile the proposed Yavadunam yields more power saving for the bit sizes beyond 16 bit. The figure 4 shows

Figure 4. Detailed Power comparison of various 32 bit multipliers for 180 nm technology.

the graphical representation of the detailed power analysis of various 32 bit multipliers for 180 nm technology.

The proposed multiplier is not advantageous for lower order bits as they consume more power, but they sound good for higher order bits. The leakage power is less in case of the proposed multipliers compared to the other multipliers. Hence the proposed multiplier is beneficial for higher order bits from 32 bit onwards. The performance analysis report of several 32 bit multipliers synthesized in 180 nm technology reveals that the delay of the proposed Yavadunam multiplier is 49% better than Array multiplier, 27.9% performs well than Braun multiplier, 60.35% superior than Shift and Add multiplier, 17.75% improved than Wallace multiplier, 8.81% better than Dadda multiplier, 2.78% superior than Urdhava multiplier and 44.46% better than Nikhilam multiplier. The area of the proposed Yavadunam multiplier is 15.43% better than Array multiplier, 15.43% performs well than Braun multiplier, 78.45% superior than Shift and Add multiplier, 16.38% improved than Wallace multiplier, 5.04% better than Dadda multiplier, 22.11% superior than Urdhava multiplier and 12.11% better than Nikhilam multiplier. The power saving of the proposed Yavadunam multiplier is 92.78% better than Array multiplier, 92.50% performs well than Braun multiplier, 93.784% superior than Shift and Add multiplier, 89.73% improved than Wallace multiplier, 90.07% better than Dadda multiplier, 87.23% superior than Urdhava multiplier and 29.59% better than Nikhilam multiplier.

The area delay product of the proposed Yavadunam multiplier is 56.88% better than Array multiplier, 39.01% performs well than Braun multiplier, 91.46% superior than Shift and Add multiplier, 31.17% improved than Wallace multiplier, 13.46% better than Dadda multiplier, 65.33% superior than Urdhava multiplier and 23.57% better than Nikhilam multiplier.

The power delay product of the proposed Yavadunam multiplier is 96.32% better than Array multiplier, 94.59% performs well than Braun multiplier, 97.53% superior than Shift and Add multiplier, 91.56% improved than Wallace multiplier, 88.37% better than Dadda multiplier, 57.23% superior than Urdhava multiplier and 39.56% better than Nikhilam multiplier.

8. Conclusion

The problem of designing a distinct squaring structure primarily based on Yavadunam sutra is resolved and further completely unique multiplier architecture for positive, negative or mixed deficiencies based totally on Yavadunam sutra is designed. The proposed architecture is precise and gives accurate end result for any type of inputs. As compared with the prevailing techniques like array multiplier, shift and add multiplier, Wallace, Dadda, Urdhava and Nikhilam multipliers the speed of the proposed system is

improved, design complexity is reduced in higher order bits when compared with the lower order bits, proposed technique utilized less number of parts and interconnections and furthermore expends less area. The proposed architecture can still be modified to improve the speed further.

References

- [1] Akanksha K and Shobha S 2015 Applications of Vedic multiplier designs-a review. In: *Proceedings of the 4th international conference on reliability, infocom technologies and optimization (trends and future directions)*. pp. 1–6
- [2] Deepa A and Marimuthu C N 2017 A high speed VLSI architecture of a pipelined reed solomon encoder for data storage in communication systems. *Asian J. Res. Soc. Sci. Hum.* 7(2): 228–238
- [3] Neeraj M and Asmita H 2013 An advancement in the $N \times N$ multiplier architecture realization via the ancient Indian Vedic mathematics. *Int. J. Electron. Commun. Comput. Eng.* 4(2): 544–548
- [4] Jinesh S, Ramesh P and Thomas J 2015 Implementation of 64 bit high speed multipliers for DSP application-based on vedic mathematics. In: *Proceedings TENCON 2015 IEEE region 10 conference*. pp. 1–5
- [5] Arushi S, Dheeraj J, Sanjay J, Kumkum V and Swati K 2012 Compare Vedic multipliers with conventional hierarchical array of array multiplier. *Int. J. Comput. Technol. Electron. Eng.* 2(6): 52–55
- [6] Nisha Angeline M and Valarmathy S 2016 Implementation of N-bit binary multiplication using $N-1$ bit multiplication based on Nikhilam Sutra principles and bit reduction. *Transylv. Rev.* 24: 982–992
- [7] Richa S, Manjit K and Gurmohan S 2015 Design and FPGA implementation of optimized 32-bit Vedic multiplier and square architectures In: *Proceedings of the international conference on industrial instrumentation and control*. pp. 960–964
- [8] Sriraman L, Saravana Kumar K and Prabakar T N 2013 Design and FPGA implementation of binary squarer using Vedic mathematics In: *Proceedings of the 4th international conference on computing, communication and networking technologies*. pp. 1–5
- [9] Vaijyanath K, Linganaouda K and Subhash K 2013 Low power square and cube architectures using Vedic Sutras. In: *Proceedings of the 5th international conference on signal and image processing*. pp. 354–358
- [10] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaj 2009 *Vedic mathematics*. Delhi: Agarwala V S Delhi Motilal Banarasidass Publishers Pvt. Ltd
- [11] Deepa A and Marimuthu C N 2018 High speed VLSI architecture for squaring binary numbers using Yavadunam Sutra and bit reduction technique. *Int. J. Appl. Eng. Res.* 13(6): 4471–4474
- [12] Deepa A and Marimuthu C N 2017 A modified multiplier architecture using Yavadunam: a squaring algorithm of Vedic mathematics. *Perspectivas em ciencia da Informacao* 22(3): 322–332
- [13] Wang J S, Kuo C N and Yang T H 2004 Low power fixed width array multipliers. In: *Proceedings of international*

- symposium on low power electronics and design.* pp. 307–312
- [14] Wallace C S 1964 A suggestion for a fast multiplier. *IEEE Trans. Electron. Comput.* 13(1): 14–17
 - [15] Hussain R K and Sah 2015 Performance comparison of Wallace multiplier architectures. *Int. J. Innov. Res. Sci. Eng. Technol.* 4(1): 2347–6710
 - [16] Dadda L 1965 Some schemes for parallel multipliers. *Alta Freq.* 34: 349–356
 - [17] Ramesh A P 2011 Implementation of dadda and array multiplier architecture using tanner tool. *IJCSET* 2(2): 28–41
 - [18] Anitha R, Nelapati A, Lincy Jesima W and Bagyaveer-eswaran V 2012 Comparative study of high performance Braun's multiplier using FPGAs. *IOSR J. Electron. Commun. Eng.* 1(4): 33–37
 - [19] Arish S and Sharma R K 2015 An efficient binary multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm. In: *Proceedings of the 2015 global conference on communication technologies*. pp. 192–196
 - [20] Harish Kumar Ch 2013 Implementation and analysis of power, area and delay of array, Urdhva, Nikhilam Vedic multipliers. *Int. J. Sci. Res. Publ.* 3(1): 1–5
 - [21] 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. Very Large Scale Integr. (VLSI) Syst.* 17(2): 267–386