RESEARCH ARTICLE OPEN ACCESS

VeMSC: Application of Vedic Mathematics for designing Multipliers, Squarers and Cubers

Patil Siddhant Vitthal[1], Dr. Pravin Mane[2]
[1,2]Department of Electrical and Electronics Engineering,
BITS Pilani - Goa Campus, India - 403726
Email:[1]f20150475@goa.bits-pilani.ac.in, [2]pravinmane@goa.bits-pilani.ac.in

Abstract:

Multipliers are the indispensable part of an electronic system. Due to multipliers being at the heart of an electronic system, it is necessary to optimize them in terms of path delay, power dissipation and area consumption. The optimization of these parameters helps in improving the overall performance of the electronic system. The paper targets on the improvement in the performance of the existing multiplier, squarer and cuber designs by using the different principles mentioned in Vedic Mathematics. Simulation and synthesis is performed for designs using 2-bit, 4-bit, 8-bit and 16-bit operands using Cadence - Simulation Analysis Environment SimVision (64)14.10-s007, Cadence - NCLaunch (64)14.10-s007 and Cadence Encounter (R) RTL Compiler RC14.10. The most captivating observation from the results is that performing a cube of a 2-bit number using principles of Vedic Mathematics requires 96.52×less time, 99.141×less power and 98.92×less area as compared to performing cube of a 2-bit number using multiple stages of an array multiplier.

Keywords — **Accurate Computation, Vedic Mathematics, Urdhva Triyak Technique, Multipliers, Dvanda, Square, Cube**

I. INTRODUCTION

A multiplier is the basic component of an electronic system. It is at the heart of Microprocessors, Digital Signal Processing Systems [1] and Mobile Telecommunication Systems [2]. Multiplication is the basic computation required in Convolution, Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT). Multipliers also have wide applications in the field of Image Processing. Images are made up of pixels and the RGB values of these pixels are structured in the form of matrices. Image processing involves calculations with the pixel values that require matrix multiplication [3]. Therefore, the demand for high speed, low power and area-efficient multipliers is increasing day by day. The normal pen-paper multiplication algorithm needs to be modified in a way to reduce the number of calculation steps and introduce parallelism into the process. The reduced number of calculations help to reduce the required resources for getting the final accurate result. Reduction in the number of resources helps in reducing the power dissipation and the length of the critical path, which in turn reduces the delay time. Another possible solution for getting better performance would be using approximate multipliers as described in [4]–[6]. This solution would be application-specific. It would be effective only in the case of error-resilient applications [7], [8] because certain applications cannot tolerate the erroneous results produced by approximate calculations.

The paper considers the application of the proposed methodology for both types, error-resilient and non-error-resilient applications. Therefore, it considers Vedic Mathematics as a viable solution and focuses on its implementation. The paper also elaborates on designing Squarer and Cuber circuits due to their wide range of applications in polynomial approximation [9], [10] and robotics [11].

Vedic Mathematics is the ancient Indian system of mathematics that was rediscovered in the early twentieth century from the Vedas by Sri Bharati Krishna Tirthaji Maharaj (1884-1960) [12]. It is used by people to tackle mathematical problems mentally. Vedic Mathematics is classified

into 16 sutras, out of which, only two i.e. Nikhilam Sutra and Urdhva Triyak Technique are applicable

TABLE I
SIMULATION AND SYNTHESIS RESULTS OF ARRAY AND BOOTH'S MULTIPLIER

Design	AM2x2	AM4x4	AM8x8	AM16x16	BM2x2	BM4x4	BM8x8	BM16x16
Area (Number of LUTs)	11	82	397	1726	151	227	367	679
Delay (ps)	329	2322	7640	18935	1171	1158	1809	3301
Leakage Power (nW)	0.22	1.58	7.57	33.71	3.21	4.59	7.13	13.47
Dynamic Power (nW)	138.98	1379.28	8614.22	51901.25	6018.73	10344.42	15949.09	34723.81
Total Power (nW)	139.20	1380.86	8621.80	51934.96	6021.94	10349.02	15956.22	34737.28

for multiplication. Nikhilam Sutra is a specific technique that simplifies multiplication for integers closer to the powers of 10 and the Urdhva Triyak Technique is a general technique i.e. it has no restrictions on the numeric value of the multiplier and the multiplicand. Therefore, this paper explores the Urdhva Triyak Technique for performing multiplication. The implementation and analysis of different Vedic Multipliers are carried out using Cadence - Simulation Analysis Environment SimVision (64)14.10-s007, Cadence - NCLaunch (64) 14.10-s007 and Cadence Encounter (R) RTL Compiler RC14.10.

***Contributions*:** The major contributions of the present work are:

- Simulation and synthesis results of 2×2 bits, 4×4 bits, 8×8 bits, 16×16 bits Vedic multipliers using Urdhva Triyak Technique and integrated architecture approach
- Design and implementation results of 2-bit, 4-bit, 8-bit and 16-bit squarers using principles of Vedic Mathematics and getting improved performance as compared to performing the same operation using array multipliers
- A novel design for obtaining the cube of 2-bit, 4-bit, 8-bit and 16-bit number using principles of Vedic Mathematics and getting improved performance as compared to performing the same operation using array multipliers

The paper is organized into 5 sections. Section II provides the Simulation and Synthesis results of the Array and Booth's multiplier, which are the basis for showing improvement in the designs proposed in the later sections of this paper. Section II also enlightens on the related work in the field of Vedic Mathematics. Section III presents the methodology of Vedic multiplication and gives a clear picture to the reader about using Vedic Mathematics for performing Square and Cube of a number. Simulation and synthesis results of 2×2 bits, 4×4 bits, 8×8 bits and 16×16 bits Vedic Multipliers, 2-bit, 4-bit, 8-bit and 16-bit squarer and 2-bit, 4-bit, 8-bit and 16-bit cuber are described in Section IV. Section V concludes the paper by pointing out valuable observations and giving future directions.

II. BACKGROUND AND RELATED WORK

Background: The simulation and synthesis results of 2×2 bits Array Multiplier (AM2×2), 4×4 bits Array Multiplier (AM4×4), 8×8 bits Array Multiplier (AM8×8), 16×16 bits Array Multiplier (AM16×16), 2×2 bits Booth's Multiplier (BM2×2), 4×4 bits Booth's Multiplier (BM4×4), 8×8 bits Booth's Multiplier (BM8×8) and 16×16 bits Booth's Multiplier (BM16×16) are summarized in Table I. These values form the basis for finding the extent of improvement in proposed designs.

Related Work: Ranjan Kumar Barik et. al. (2016) in 'Time efficient signed vedic multiplier using redundant binary representation' [13] propose a new method of signed multiplication. The design is based on the Urdhva Triyak Sutra of Vedic Mathematics. The performance of the proposed multiplier is far better as compared to conventional multipliers.

Ms. Gadakh et. al. (2016) in 'FPGA implementation of high speed vedic multiplier' [14] elaborate on the 16 Sutras of Vedic Mathematics. The paper shows the implementation of 2×2 bits Vedic Multiplier using the Urdhva Triyak technique. It uses the 2×2 bits Vedic Multiplier as the basic building block and constructs a 4×4 bits Vedic Multiplier using these blocks. It also discusses the FPGA implementation of the 8×8 bits Vedic Multiplier using the newly constructed 4×4 bits Vedic Multiplier blocks.

Nagaraju N. et. al. (2015) in 'FPGA implementation of an efficient vedic multiplier' [15] aim to implement an 8×8 bits Vedic Multiplier using 4×4 bits Vedic Multiplier blocks. 3:2 compressors are used for reducing the number of gates required for performing the addition of the partial product terms. This method gives better performance as compared to the designs proposed

in [8] due to the reduction in the critical path length as less number of gates are utilized.

Anannya Maiti et. al. (2016) in 'Design and implementation of 4 bit vedic multiplier' [16] implements a 4×4 bits Vedic Multiplier without using a 2×2 bits Vedic Multiplier block. It uses propagation delay as the parameter for merit. It concludes that 4×4 bits Vedic Multipliers are better in terms of performance than Array Multipliers.

Premananda B.S. et. al. (2013) in 'Design and implementation of 8-bit vedic multiplier' [17] implement 8×8 bits Vedic Multiplier using 4×4 bits Vedic Multiplier blocks. It uses a modified ripple-carry addition for shifting the carry generated by the partial products.

Vengadapathiraj. M et. al. (2015) in 'Design of high speed 128×128 bit vedic multiplier using high speed adder' [18] uses 2×2 bits Vedic Multiplier as the building block for implementing 128×128 bits Vedic Multiplication. It considers the Combinational Path delay as the factor for deciding a better multiplier. It shows the effect of implementing addition using different adders like Carry Save Adder (CSA), Carry Select Adder (CLA) and Ripple Carry Adder (RCA) on the time delay.

L. Srirman et. al. (2013) in 'Design and FPGA implementation of binary squarer using vedic mathematics' [19] proposes squarer based on Ekadhikena Purvena Sutra. This squarer is useful only for decimal numbers ending with '5'. It is a specific squarer. The paper concludes that the proposed squarer is better as compared to an array squarer for squaring numbers ending with '5'.

The literature review shows that multiplication performed using Vedic Mathematics outperforms other techniques of multiplication. The implementations of Vedic Multipliers use modular architecture. No work can be found in the literature that uses an integrated architecture approach for designing Vedic multipliers having operand size more than 4-bits.

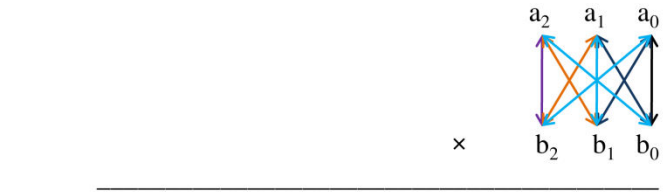

Fig. 1 Multiplication of two 3-bit binary numbers using Urdhva Triyak Technique

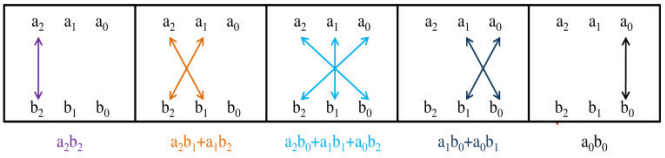

Fig. 2 Split version for calculation of partial product terms

Also, no designs have been proposed that use Vedic Mathematics for designing Cubers having operand size beyond 2-bits. No comparison exists between the simulation and synthesis results obtained on performing square and cube of a number using multiple stages of array multipliers and techniques prescribed in Vedic Mathematics. Therefore, the paper focuses on designing Vedic Multipliers, Squarers and Cubers as mentioned in the Contributions.

III. METHODOLOGY

A. Urdhva Triyak Technique

'Urdhva Triyak' is a Sanskrit word. It means Vertically and Crosswise. It is the most powerful technique when it comes to Vedic Multiplication as it is a General Technique. It is also referred to as Urdhva Triyagbhyam sutra. The multiplication methodology is explained with the help of Fig. 1. It demonstrates the multiplication of two 3-bit numbers. The first step involves computation of the sum of partial products i.e. a_0b_0, $a_1b_0 + a_0b_1$, $a_2b_0 + a_1b_1 + a_0b_2$, $a_2b_1 + a_1b_2$ and a_2b_2. The second step is the transfer of carry from right to left. It is to be noted that the terms pertaining to the partial products are generated at the same time as they are independent of each other. The simultaneous generation of partial products helps in the reduction of the computation time. Fig. 2 aims to aid the visualization of the user for understanding

the calculation of partial product terms. The methodology shown in Fig. 2 can be extended to 'n'× 'n' bits multiplication, where n ϵ Integer.

Fig. 3 shows the entire multiplication process with the help of a numerical example using a binary multiplicand and multiplier. It demonstrates bit-wise multiplication of two 4-bit numbers. As mentioned previously, in the first step, all the partial product terms are calculated. The horizontal arrows from right to left indicate the shift of carry bits. Fig. 4 represents the Data Flow Graph

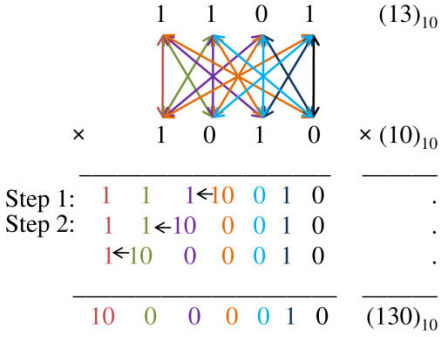

Fig. 3 Multiplication of two 4-bit numbers using Urdhva Triyak Technique

for the same. The proposed design is unlike the designs discussed in Section II. It uses the modular multiplier concept. It does not use the 2×2 bits Vedic Multiplier as a building block.

B. Square of a number

The square of a number is calculated by using the basic block called Dvanda. Calculating square of a number using Dvanda is known as Dvanda Yoga. The definition of a Dvanda is as follows:

$$D(a) = a^2 \quad (1)$$
$$D(bc) = 2 \times b \times c \quad (2)$$
$$D(def) = 2 \times d \times f + e^2 \quad (3)$$
$$D(ghij) = g \times j + h \times i \quad (4)$$

where 'a', 'bc', 'def', 'ghij' are 1-bit, 2-bit, 3-bit, and 4-bit numbers respectively. The Dvanda Yoga is further elaborated with the help of an example using decimal operands as shown in Fig. 5. The same method can be applied to binary numbers.

The first step is to calculate all the Dvandas. This step is followed by shifting the carries from right to left for arriving at the final result.

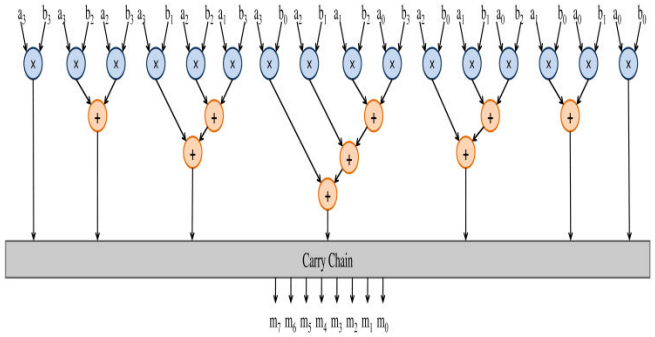

Fig. 4 Data Flow Graph for 4×4 bits Vedic Multiplication

52^2 = D(5) | D(52) | D(2)
= 5^2 | 2×5×2 | 2^2
= 25 ← 20 | 4
= 27 | 0 | 4
= 2704

Fig. 5 Numerical example for calculating the square of a 2 digit number using Vedic Mathematics

Fig. 6 shows the flowchart for finding the square of a 4-bit number. The calculations of this method seem to be similar to Urdhva Triyak Technique, but they are different in a way that they require less number of resources as compared to performing the same operation using Vedic multipliers using Urdhva Triyak Technique. Reduced number of resources also help to reduce power and delay of the circuit. This statement is supported by facts and numerical figures in the results section.

C. Cube of a number

Vedic Mathematics explores the identity mentioned in equation 5 for performing cube of a number.

$$(a + b)^3 = a^3 + 3 \times a^2 \times b + 3 \times a \times b^2 + b^3 \quad (5)$$

The use of this identity is explained by performing the calculations for finding the cube of 52 as shown in Fig. 7. In the example, 52 is split as (50+2) i.e. 50 and 2. Therefore, 'a' in the identity is replaced by 5 and 'b' is replaced by 2. Note that 'a' is 5 and not 50. This data computation concludes towards giving correct results because the carry

chain takes care of the place value of the number. Therefore, this method requires the computation of 5^3 and not 50^3. This methodology saves a lot of

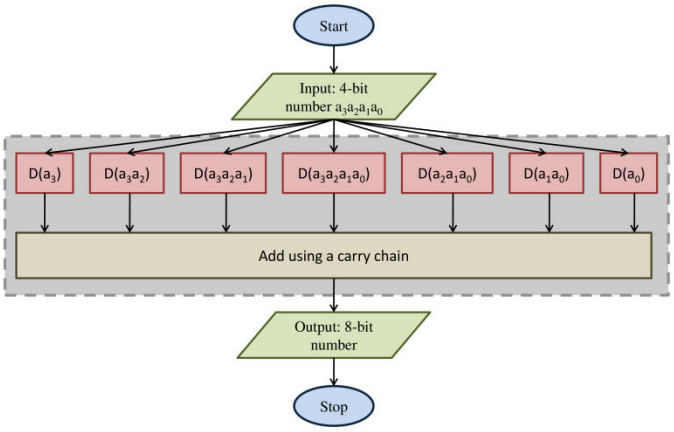

Fig. 6 Flowchart for finding the square of a 4-bit number

$$
\begin{aligned}
52^3 &= 5^3 \mid 3\times5^2\times2 \mid 3\times5\times2^2 \mid 2^3\\
&= 125 \mid 3\times25\times2 \mid 3\times5\times4 \mid 8\\
&= 125 \mid 150 \mid 60 \mid 8\\
&= 125 \mid 156 \mid 0 \mid 8\\
&= 140 \mid 6 \mid 0 \mid 8\\
&= 140608
\end{aligned}
$$

Fig. 7 Numerical example for calculating the cube of a 2 digit number using Vedic Mathematics

resources and reduces the path length and delay. The positivity of this design is supported by facts and figures in the results section. The partial product terms are calculated and the carry is shifted from right to left. In Fig. 7, the horizontal arrows from right to left indicate the shift of carry. The method shown works as long as the numbers are 2 digits wide. In the binary system, we can use the abovementioned technique only for 2-bit operands.

In order to extend the application of Vedic Mathematics for performing cube of 4-bit, 8-bit and 16-bit numbers the following method is proposed. The method uses an indirect form of modular approach and is explained with an example as shown in Fig. 8. Consider the case where we intend to calculate the cube of a 4 digit number. We split the number into two parts: the lower two digits and the upper two digits of significance. We use equation 5, where 'a' is replaced by the upper two digits and 'b' is replaced by the lower two digits of the given number. We use the previously designed Vedic Multiplier, Squarer and Cuber Circuits of suitable port widths for the intermediate computations. In each stage, 2 digits are retained and the carry is shifted from right to left.

In general, if we have an 'n' bit number, where 'n' is an even integer, we split the number into two parts of n/2 bits each. We apply equation 5 and retain n/2 bits in each stage.

$$
\begin{aligned}
5219^3 &= 52^3 \mid 3\times52^2\times19 \mid 3\times52\times19^2 \mid 19^3\\
&= 140608 \mid 3\times2704\times19 \mid 3\times52\times361 \mid 6859\\
&= 140608 \mid 154128 \mid 56316 \mid 6859\\
&= 140608 \mid 154128 \mid 56384 \mid 59\\
&= 140608 \mid 154691 \mid 84 \mid 59\\
&= 142154 \mid 91 \mid 84 \mid 59\\
&= 142154918459
\end{aligned}
$$

Fig. 8 Numerical example for calculating the cube of a 4 digit number using Vedic Mathematics

IV. Design and Implementation Results

The section is split into 3 parts, namely the results of Vedic Multiplier, the results of Vedic Squarer and the results of Vedic Cuber.

A. Results of Vedic Multiplier

Table II summarizes the result of Vedic Multipliers for operands of varying bit-width.

TABLE II
Simulation and Synthesis Results of Vedic Multiplier

Design	VM2x2	VM4x4	VM8x8	VM16x16
Area (Number of LUTs)	11	79	393	1722
Delay (ps)	329	2283	7613	18889
Leakage Power (nW)	0.22	1.51	7.51	33.53
Dynamic Power (nW)	138.98	1325.32	8798.70	52084.14
Total Power (nW)	139.20	1326.83	8806.21	52117.67

Comparing the results with Table I, an observation can be made that Vedic Multipliers always consume less amount of area as compared to the Array Multipliers. Although the power consumption of Vedic Multipliers is more than the Array Multipliers of respective bit-widths, their delay is considerably less. As two out of the three design parameters are improving, we can say that the Vedic Multipliers are better as compared to the traditional Array Multipliers. Fig. 9 shows the gate level representation of the 2×2 bits Vedic

Multiplier. It uses 1 NOT gate, 6 AND gates and 1 XOR gate.

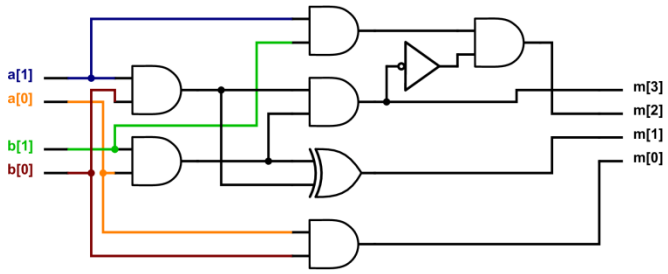

Fig. 9 Gate level representation for 2×2 bits Vedic Multiplier

B. Results of Vedic Squarer

The simulation and synthesis results of the Vedic Squarer are summarized in Table III. The Area, Power and Delay of Vedic Squarer is less in all the cases when compared to performing the same operation as compared to the Array Multiplier. Area and Power savings of up to 63.63% and 74.47% respectively are achieved by

TABLE III
SIMULATION AND SYNTHESIS RESULTS OF VEDIC SQUARER

Design	VS2	VS4	VS8	VS16
Area (Number of LUTs)	3	17	188	866
Delay (ps)	92	340	3698	10794
Leakage Power (nW)	0.06	0.30	3.57	16.29
Dynamic Power (nW)	35.50	215.69	3378.13	19616.44
Total Power (nW)	35.56	216.00	3381.70	19632.72

the Vedic Squarer when compared to the use of Array Multipliers for performing the square operation. Delay is reduced up to 72.036%.

Fig. 10 shows the gate level representation of the circuit for performing the square of a 2-bit number. It uses 1 NOT gate, 1 AND gate and 1 NOR gate. Comparing Fig. 9 and Fig. 10 it can be seen that Fig. 10 uses less number of resources as compared to Fig. 9. This is due to the simplification of equations 1 and 2 to the following equations, the reason being that we are dealing with binary numbers.

$$D(a_1) = a_1^2 = a_1 \quad (6)$$

$$D(a_1a_0) = 2 \times a_1 \times a_0 = [10]_2 \times a_1 \times a_0 \quad (7)$$

$$D(a_0) = a_0^2 = a_0 \quad (8)$$

Equation 7 involves multiplication with $[10]_2 = [2]_{10}$. Therefore, the bit s_1 is permanently set to 0. Thus, for performing square of a 2-bit number it would be efficient to implement the method described above, rather than using 2×2 bits Vedic Multiplier.

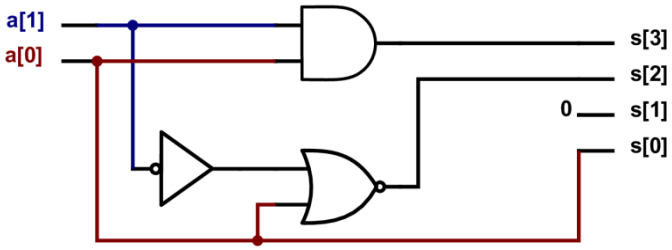

Fig. 10 Gate level representation for the square of a 2-bit number

C. Results of Vedic Cuber

Table IV summarizes the simulation and synthesis results of the Vedic Cuber. Area savings of up to 98.92%, a power reduction of up to 99.14% and delay reduction of up to 96.529% are achieved using the proposed designs when compared to using

TABLE IV
SIMULATION AND SYNTHESIS RESULTS OF VEDIC CUBER

Design	VC2	VC4	VC8	VC16
Area (Number of LUTs)	1	214	1238	5560
Delay (ps)	92	2748	7704	18700
Leakage Power (nW)	0.02	4.60	25.90	116.80
Dynamic Power (nW)	13.03	2337.18	17715.35	121443.75
Total Power (nW)	13.06	2341.78	17741.25	121560.55

an 'n'×'n' Array Multiplier followed by '2n'×'2n' Array Multiplier for performing cube of an 'n'-bit number.

Fig. 11 shows the gate level representation of the Vedic 2-bit Cuber. The circuit uses only a single AND gate. When compared to Fig. 9 and Fig. 10, this design is the most efficient in terms of performance parameters as it uses the least number of resources. The reduction in the number of resources can be explained by the simplification of equation 5 for binary numbers as shown in equations 9 and 10.

$$(a_1a_0)_2^3 = a_1^3 \mid 3 \times a_1^2 \times a_0 \mid 3 \times a_1 \times a_0^2 \mid a_0^3 \quad (9)$$

$$= a_1 \mid 3 \times a_1 \times a_0 \mid 3 \times a_1 \times a_0 \mid a_0 \quad (10)$$

The LSB bit of output is always set to a_0 and the method requires computation of one logical operation $a_1 \times a_0$ i.e. a_1 AND a_0.

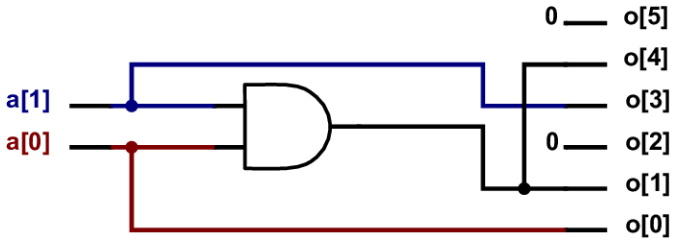

Fig. 11 Gate level representation for the cube of a 2-bit number

V. Conclusions

The literature survey shows that multipliers implemented using Vedic Mathematics are faster as compared to normal array multipliers. Also, the previously proposed designs use a modular design approach for designing higher bit multipliers. In this paper, we use an integrated architecture approach for implementation. The paper enlightens on the use of principles mentioned in Vedic Mathematics for performing square and cube of a number. A novel technique for performing a cube of a number exceeding 2-bits is proposed in this paper.

The most captivating observation from the results is that performing cube of a 2-bit number using principles of Vedic Mathematics requires 96.52×less time, 99.141×less power and 98.92×less area as compared to performing cube of a 2-bit number using multiple stages of an array multiplier. Similar values can be obtained for 4-bit, 8-bit and 16-bit cubers. Hence, the principles of Vedic Mathematics leave a large scope for improving the performance of current mathematical operations.

The designs discussed in the paper can be used in circuits which involve a large number of multiplications like the Digital Signal Processing and Image Processing Circuits. Also, approximate circuits can be designed using the circuits proposed in the paper, by manipulating with the carry chains or introducing truncation [6], [20], [21]. These approximate circuits can then be applied to error-resilient applications [22] for improving their performance. Thus, Vedic Mathematics leaves a large scope for exploration.

References

1. *A. Mittal, A. Nandi, and D. Yadav, "Comparative study of 16-order fir filter design using different multiplication techniques," IET Circuits, Devices Systems, vol. 11, no. 3, pp. 196–200, 2017.*
2. *G. R. Cooper and R. W. Nettleton, "Communications: Cellular mobile technology: The great multiplier: It can greatly expand mobile communications, but some technical and regulatory issues remain unsolved," IEEE Spectrum, vol. 20, no. 6, pp. 30–37, June 1983.*
3. *P. S. Vitthal, S. Balasubramanian, and P. S. Mane, "Color analysis and classification based on machine learning technique using rgb camera industrial practice and experience paper," in 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Dec 2017, pp. 1–5.*
4. *B. S. Prabakaran, S. Rehman, M. A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, and M. Shafique, "Demas: An efficient design methodology for building approximate adders for fpga-based systems," in 2018 Design, Automation Test in Europe Conference Exhibition (DATE), March 2018, pp. 917–920.*
5. *M. Masadeh, O. Hasan, and S. Tahar, "Comparative study of approximate multipliers," CoRR, vol. abs/1803.06587, 2018. [Online]. Available:http://arxiv.org/abs/1803.06587*
6. *M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited:Cross-layer approximate computing: From logic to architectures," in 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), June 2016, pp. 1–6.*
7. *A. El-Helw, M. Moniri, and C. Chibelushi, "Error-resilient pattern classification using a combination of spreading and coding gains," IET Image Processing, vol. 1, pp. 278–286(8), September 2007. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-ipr20070007*
8. *A. Kanduri, M. Haghbayan, A. M. Rahmani, P. Liljeberg, A. Jantsch, H. Tenhunen, and N. Dutt, "Accuracy-aware power management for many-core systems running error-resilient applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2749–2762, Oct 2017.*
9. *D. Lee, R. Cheung, W. Luk, and J. Villasenor, "Hardware implementation trade-offs of polynomial approximations and interpolations," IEEE Transactions on Computers, vol. 57, no. 5, pp. 686–701, May 2008.*
10. *S. Hsiao, Y. Chen, and H. Liang, "Architectural exploration of function computation based on cubic polynomial interpolation with application in deep neural networks," in 2018 21st Euromicro Conference on Digital System Design (DSD), Aug 2018, pp. 22–29.*
11. *W. P. Hunek, "An application of polynomial matrix -inverse in minimum-energy state-space perfect control of nonsquare lti mimo systems," in 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Aug 2015, pp. 252–255.*
12. *A. Gupta, The Power of Vedic Maths.*
13. *R. K. Barik, M. Pradhan, and R. Panda, "Time efficient signed vedic multiplier using redundant binary representation," The Journal of Engineering, vol. 2017, no. 3, pp. 60–68, 2017.*
14. *S. N. Gadakh and A. S. Khade, "Fpga implementation of high speed vedic multiplier," in International Conference Workshop on Electronics Telecommunication Engineering (ICWET 2016), Feb 2016, pp. 184–187.*
15. *A. N. D. S. M. R. Nargaraju N., Vidhya Shree. G, "Fpga implementation of an efficient vedic multiplier," International*

Journal of Emerging Technology and Advanced Engineering, vol. 5, no. 3, pp. 326–330, Mar. 2015.

16. *R. S. S. M. Anannya Maiti, Koustuv Chakraborty, "Design and implementation of 4-bit vedic multiplier," International Journal of Emerging Trends in Science and Technology, vol. 3, no. 5, pp. 3865–3868, May 2016.*
17. *P. B.S, S. S. Pai, and S. S. Bhat, "Design and implementation of 8-bit vedic multiplier," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, no. 12, pp. 5877–5882, Dec. 2013.*
18. *G. M. V. K. A. G. R. Vengadapathiraj. M, Rajendhiran. V, "Design of high speed 128x128 bit vedic multiplier using high speed adder," International Journal of Science, Engineering and Technology Research, vol. 4, no. 3, pp. 615–619, Mar. 2015.*
19. *L. Sriraman, K. S. Kumar, and T. N. Prabakar, "Design and fpga implementation of binary squarer using vedic mathematics," in 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), July 2013, pp. 1–5.*
20. *W. Liu, C. Tian, P. Yin, C. Wang, E. Jr, and F. Lombardi, "Design and analysis of approximate redundant binary multipliers," IEEE Transactions on Computers, vol. PP, pp. 1–1, 12 2018.*
21. *K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in Fifteenth International Symposium on Quality Electronic Design, March 2014, pp. 263–269.*
22. *V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Nov 2016, pp. 1–7.*