Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 7, No. 1, 2025 (pp. 66-74)
WWW.VLSIJOURNAL.COM

Efficient VLSI Architecture Integrating Vedic Mathematics for Square Computation

U. G. Patil[1], Prabodh Jagtap[2*], Kedar Adake[3], Sanika Dhanbhar[4], Ajay Paithane[5]

[1]Professor, Dept. of Electronics and Telecommunication, Rajarshi Shahu College of Engineering, Pune, India - 411033
[2-4]Research Scholar, Dept. of Electronics and Telecommunication, Rajarshi Shahu College of Engineering, Pune, India - 411033
[5]Professor, Dept. of Electronics and Telecommunication, Dr. D. Y. Patil Institute of Engineering Management and Research, Pune, India - 411044

KEYWORDS:
Vedic mathematics,
Urdhva-Tiryagbhyam Sutra,
FPGA,
Spartan-7,
Artix-7,
Virtex-4,
Multiplier circuit,
Booth multiplier.

ARTICLE HISTORY:
Received : 08.12.2024
Revised : 15.03.2025
Accepted : 02.04.2025

DOI:
https://doi.org/10.31838/jvcs/07.01.09

Abstract

Implementation of combinational logic circuits in combination with Vedic maths sutra has many advantages over traditional multiplier circuits, such as reduced time delay, less resource utilization, and less power consumption by the selection of the proper FPGA family. Multiplication is one of the important instructions used for performing complex operations in DSP processors. The proposed paper presents the use of Urdhva-Tiryagbhyam Sutra for the square operation. The results of the designed Vedic circuit show that, there is a 17.36% reduction in time delay and an 8.13% reduction in power consumption in Spartan-7 than Artix-7. There is a 33.6% reduced time delay than the Nikhilam sutra, 45.87% reduced delay than the Yavadunam sutra, 87.07% less delay than the Booth multiplier, and 67.83% reduced delay than Booth Wallace multiplier. Thus, implementing the Vedic Sutra for finding the square of a given number causes a reduction in time delay, power consumption and small chip size.

Author's e-mail ID: sapaithane_entc@jspmrscoe.edu.in, prabodhjagtap5703@gmail.com, kedaradake45@gmail.com, sanikadhanbhar1103@gmail.com, ajaypaithane@gmail.com

Author's IRCID ID: 0000-0001-9682-508X, 0009-0007-1205-462X, 0009-0006-4695-213X, 0009-0007-6503-0146, 0000-0001-6367-481X

How to cite this article: Patil UG, Jagtap P, Adake K, Dhanbhar S, Paithane A. Efficient VLSI Architecture Integrating Vedic Mathematics for Square Computation, Journal of VLSI Circuits and System, Vol. 7, No. 1, 2025 (pp. 66-74).

Introduction

In ancient times, mathematics in India was known as Vedic Mathematics, which was founded in 1884 and 1960 by Swami Bharati Krishna Tirthaji.[1] Vedic Mathematics basically consists of Vedic Sutras (Formulas) which are used to solve complex mathematical operations. Within Vedic mathematics, sixteen formulas (sutras) and thirteen Upa Sutras are essential. They are utilized in combination with each other to solve mathematical operations.[1] Vedic mathematics sutras are as follows: Ekādhikena Pūrvena uses one more than the previous number, while Nikhilam Navataścaramam Daśatah simplifies multiplication by subtracting from 9 and 10. Urdhva-Tiryagbhyam uses vertical and crosswise approaches for general multiplication. Parāvartya Yojayet helps solve equations by transposing terms. Śūnyam Sāmyasamuccaye eliminates equal sums, and (Ānurūpye) Śūnyamanyat zeros out ratios. Sankalana-Vyavakalanābhyām alternates between addition and subtraction. Pūrṇapūrṇābhyām completes expressions for simplification. Yāvadūnam and Chalana-Kalanābhyām address deficiencies and differences, respectively. Vyashtisamanashti breaks down problems, while Śeṣānyankena Charamena helps find remainders. Sopāntya Dvandva Yogah and Gunitasamuccayah aid in roots and sums, with Gunakasamuccayah and Ekanyūnena Pūrvena offering factorization shortcuts.[1] At present, in every processor one of the most time-consuming instructions is multiplication and division operation. More execution time is required for multiplication operation in comparison of other instructions like additions and subtract instructions.[2] Multiplication operations such as square and cube method are widely used in almost every Digital Signal Processor (DSP) application like convolution, Fast Fourier Transform (FFTs), Filters, Image processing and Arithmetic logic Unit.[3] Digital Signal Processor are very high-speed microprocessor used in various applications that are required for real-time processing of the digital signals.[2-4] In, traditional method square was performed by using algorithms like array multiplier circuits and booth multiplier this method requires large amount of hardware and they are one of the most time-consuming multiplication methods. Therefore, well-known techniques like Vedic mathematics and Quantum-Dot Cellular Automata (QCA) can be implemented for increasing the computational speed of doing complicated arithmetic functions easily and in less time.[5] Using this Vedic mathematics square operation can be implemented easily with less hardware requirement and less computational delay. This approach can be used in various FPGAs and DSP processor to increase their efficiency, speed and to decrease the power consumption rate. For finding the square of a binary number Vedic mathematics concept makes use of simple adder circuits.

Literature Survey

1. **Booth Multiplication:** Booth multiplication algorithm is very useful in binary multiplication, which can be used to reduce the number of addition operations that are being performed. It uses a method that speedily and efficiently performs a multiplication process with the help of a multiplier (M) and the multiplicand (Q). Booth's algorithm synthesizes the addition and subtraction operations by using sequences of same bits in the multiplicand which is supposed to decide when and for which purposes to perform addition or subtraction.[6] Booth multiplier architecture was introduced by Andrew D. Booth in 1951. The Booth developed this algorithm to improve the efficiency of binary multiplication, in specific case like positive or negative numbers. Booth's algorithm, helps to reduce no. of addition and subtraction operation by encoding the multiplier which is widely used in DSP and other fields requiring fast multiplication.[7]

As the growth of computer applications is considered in the fields like computer graphic and signal-processing, faster Arithmetic Logic Unit (ALU) and typically for multiplier circuits are required with increased size of bit multiplication. The advanced VLSI technology gave the free hand to integrate many complex component, which used to be the most difficult task in the past. Mainly there are two types of operations in order to implement modified Booth algorithm, for generation of partial product and accumulator partial products, Booth-Encoder and Booth-Selector are utilized. The adder or compressor circuit used in accumulating the entire partial product.[7] These two operations make the Booth multiplier fast and more efficient than traditional method. Booth Encoder (B.E.) - Booth Encoder will be designed in many different ways, with a trade off between how much space it uses and how fast it will work. B.E. is used to decrypt the multiplier signals and final outputs is passed to the Booth Selector (BS) for creating a partial-product term. The final product of this process is an adder tree, this will add up all partial-products terms to get the final multiplication result. In this setup, a 2's Complement Error Correction is usually required, it will be done within the adder to handle the negative numbers correctly. Result of this multiplier circuit will have n/2+1 partial-products for a n*n multipliers. It is derived from the traditional method used for multiplying signed numbers.[7]

2. **Shift and Add Algorithm:** Shift and add Method for Binary Multiplication is a technique where a binary number performed with repetitive addition and right shift according with the bits from multiplicand in given binary number. In the multiplication operation there exist two critical components called multiplier and multiplicand are used in Shift and Add Algorithm. For instance, if M is multiplier and Q is multiplicand, then to carry out the operation through add and shift method one more element would be required. In the second statement, the issue is not mentioned and instead it is said that the accumulator can be represented as A. The accumulator stores the additions of A and M. If result produced after the addition of a and M via a carry generator then it can be kept in the C. In addition to using the Least Significant Method of Multiplicand, the Q is to be checked whether it is 0 or 1 depending on the table. The corresponding operations then to be implemented performed. In the case of ALSB being 1, then add should be proceeded on A and M and should be stored again in A. Thus, Right Shift operation takes place after the addition and stored in A. If LSB is 0 the only instruction then is a shift right. It is the never-ending process of just checking of the LSB whether it has 0 or 1. For the computation of multiplication with the number in n bits n adding and n right shifts has to be executed. Thus, we tailored these A and Q sequences and finally the two sequences got concatenated. Such an algorithm visualizes how the Add and Shift scheme can be applied to place the number in the correct place by summing and shifting repeatedly to the right. Such modifications can take into account particular need for various binary numbers as well as possible optimizations that may require more.[8]

Methods Used

Urdhva-Tiryagbhyam Sutra: Urdhva-Tiryagbhyam is one of the 16 formulas from ancient Vedic mathematics. It is the most significant sutras in Vedic Formulas, which provide the unique and efficient technique for the squaring of numbers and also performing the multiplication operations. Urdhva-Tiryagbhyam sutra is also known as “Vertically and Crosswise” [10]. The main purpose of developed multiplication method is to break down the complicated numerical calculations into the simpler steps by just multiplying the given digits in both manners vertically and cross wise, and adding the product of cross multiplication [9-10]. This method can be illustrated by representation of 4-bit and 2-bit multiplication flow as given in fig.!1.a and Fig.!1.b.

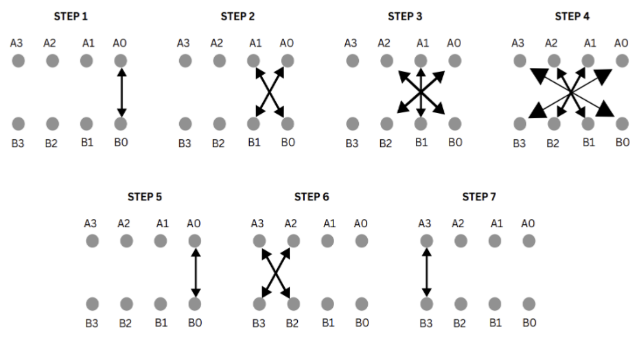


Fig. 1a: Urdhva-Tiryagbhyam Sutra Steps (4-bit)

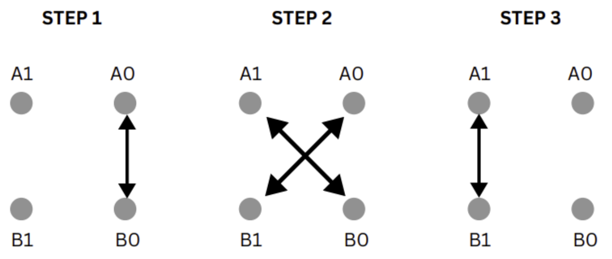


Fig. 1b: Urdhva-Tiryagbhyam Sutra Steps (2-bit)

Here is the detail example of steps in Urdhva-Tiryagbhyam Sutra with reference to fig. 5.a, the example shows decimal multiplication of 17 square.

Fig. 2 Represents the step followed for finding square of any number using Urdhva-Tiryagbhyam Sutra.

Step 1 – 7 x 7 = 49 (carry 4 to next step)
Step 2 – (1 x 7) + (1 x 7) = 14
= 14 + 4 = 18 (carry 1 to next step)
Step 3 – 1 x 1 = 1 (add carry if present)
= 1 + 1 = 2 [hundreds place 2]
Final Answer is 289 (Square of 17).

The primary advantage of this sutra is its efficiency, and reducing the number of steps compared to traditional techniques of multiplications. It also beneficial for the mental calculations and manual computations, where the speed is the main factor.

Performance Review

In this section, various critical parameters of designed square circuit are been mentioned and compared with

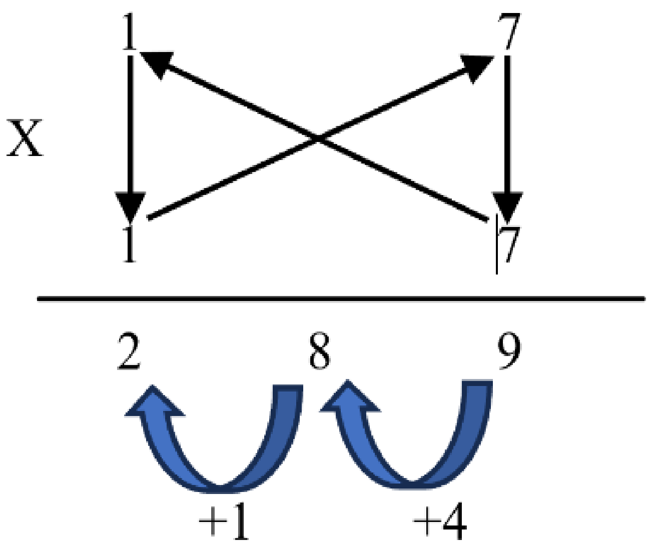


Fig. 2: Flow of Urdhva-Tiryagbhyam Method

other designing methods like Booth multiplier, Booth Wallace multiplier, Nikhilam Sutra and Yavadunam Sutra. Also, the developed squaring circuitry has been tested on other families of FPGA to monitor their performance. for discussion on efficiency of the method we have compare the parameters such as the total time delay, logic delay, net delay, total On-chip power, LUT, Input/ Output port utilization And Slice logic. The parameters of 2 bit, 4 bit, 8 bit and 16 bit have been generated and compared with other available data for square architecture.

Table 1, shows the results for square architecture using Spartan-7 xc7s100fgga676-2 part. Thus, we can compare these results with another Vedic method known as Nikhilam Sutra and Booth Multiplier for 8 bit and 16 bit circuit.

Fig. 3 below presents the performance review for total time delay of both the devices, to decide best FPGA family for designed architecture.

Fig. 3 shows, Total Time Delay Analysis for Spartan-7 and Artix-7. X axis represents the number of bit of the system and Y axis represents the Total Time Delay in nano seconds. For small designs like 2-bit system difference between Total Time Delay for both devices is less but as the number of bits in the architecture increases the Total Time Delay for the system also increases. Certainly, in case of Artix 7 family as the number of bits increases the delay due to routing is more as compared to logic delay. It can also be observed from the Table 2, which shows detailed timing analysis for both the device families. Therefore, by using Spartan-7 for implementing squaring circuit it reduces total time delay by 17.36% than using Aartix-7.

TABLE I: Parameter analysis for square circuits (Spartan-7)

Bit No.	Total Delay	Logic Delay	Net Delay	Total On-chip Power	Resource Used	
					LUT	IO Port
	(nsec)			(w)	(%)	
2	6.218	3.569	2.649	1.073	1	2
4	8.466	3.712	4.754	4.143	1	4
8	15.198	4.527	10.671	14.984	1	8
16	29.836	6.015	23.821	49.23	1	16

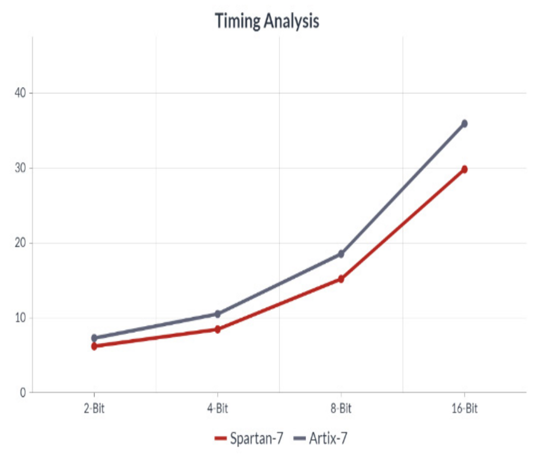


Fig. 3: Total Time Delay Analysis

TABLE II: Total Time Delay Comparison

Device family		Total Delay (nsec)	Logic Delay (nsec)	Net Delay (nsec)
Spartan-7	2-Bit	6.218	3.569	2.649
	4-Bit	8.466	3.712	4.754
	8-Bit	15.198	4.527	10.671
	16-Bit	29.836	6.015	23.821
Artix-7	2-Bit	7.306	3.918	3.388
	4-Bit	10.524	4.156	6.368
	8-Bit	18.537	5.357	13.180
	16-Bit	35.938	6.742	29.196

Fig. 4 below shows the Total On-chip Power Analysis for both the families of FPGA using same Vedic architecture.

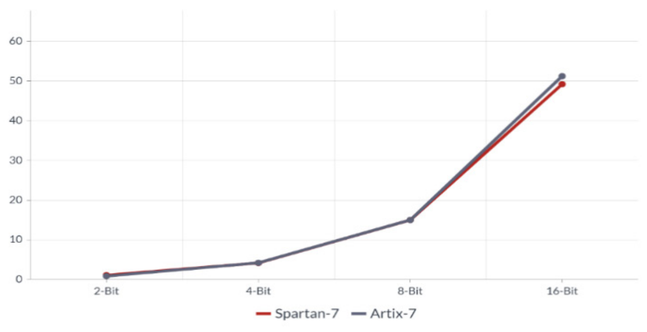


Fig. 4: Total On-chip Power Analysis

According to the observations from fig. 4, there is nearly equal power consumption for small architectures but, for Artix-7 family power consumption slightly starts to rise as larger circuitries are implemented. The amount of On-Chip Power consumption is 8.13% less in Spartan-7 family than Artix-7 family. Thus, from the above discussion we can conclude that Spartan-7 device is more efficient to use than Artix family. Table 3, shows detailed power analysis and resource utilization for both the device families.

TABLE III: Total On-Chip Power Comparison

Device family		Total On-chip Power (w)	Resource Utilization	
			LUT (%)	IO Port (%)
Spartan-7	2-Bit	1.073	1	2
	4-Bit	4.143	1	4
	8-Bit	14.984	1	8
	16-Bit	49.23	1	16
Artix-7	2-Bit	0.84	1	1
	4-Bit	4.18	1	3
	8-Bit	14.985	1	6
	16-Bit	51.248	1	13

Table 4, shows the total time delay comparison between four methods Urdhva-Tiryagbhyam Sutra, Booth Multiplier, Nikhilam Sutra and Yavadunam Sutra.

TABLE IV: Square Delay Comparison with other Methods (Spartan-7)

	(1) (nsec)	(2) (nsec) [11]	(3) (nsec) [11]	(4) (nsec) [13]
4-Bit	8.466	NR	NR	15.67
8-Bit	15.198	117	27	NR
16-Bit	29.836	232	39	NR

NR - Not reported in corresponding reference, (1)-Proposed Vedic Multiplier, (2) - Booth Multiplier, (3) - Nikhilam Sutra, (4) - Yavadunam Sutra

Table 4, shows the time delay comparison between Vedic multiplier circuit build using Urdhva-Tiryagbhyam method and other Vedic methods like Nikhilam sutra, Yavadunam sutraandotherusedmethodlikeBoothmultiplieralgorithm. By comparing the available data, squaring circuit designed using Urdhva-Tiryagbhyam gives 33.6% less delay than Nikhilam sutra[11] and 45.87% less time delay than Yavadunam sutra;[13] whereas, Urdhva-Tiryagbhyam square circuit gives 87.07% less delay than Booth multiplier method.[11] Therefore, from table 4 it can conclude that Urdhva-Tiryagbhyam technique is less time consuming then Nikhilam Sutra, Yavadunam Sutra and Booth Multiplier.

TABLE V: Vedic method Comparison with other Methods using Virtex

	Proposed Vedic Multiplier (nsec)	[14] (nsec)	Booth Wallace Multi-plier[14] (nsec)	[15] (nsec)
2-Bit	3.625	NR	NR	NR
4-Bit	4.907	11.477	13.77	4.993
8-Bit	8.412	21.55	25.756	10.32
16-Bit	16.732	28.08	59.38	NR

NR: Not reported in corresponding reference

Table 5, shows the timing analysis of 2 bit, 4 bit, 8 bit and 16 bit Vedic square multiplication circuit for developed algorithm using Virtex family of FPGA and other developed multipliers. According to the results mentioned in table 5, the developed Vedic multiplier shows 52.87% and 10.1% less time delay as compared to multiplication circuits of[14] and,[15] respectively. The proposed 4 bit, 8 bit and 16 bit Vedic circuit after comparison with the Booth Wallace Multiplier shows 67.83% less delay.[14]

Simulation

This section describes about the schematic diagram, synthesis result and simulation for proposed squaring circuit using Urdhva-Tiryagbhyam in Xilinx Vivado software by using Spartan-7 family from available devices.

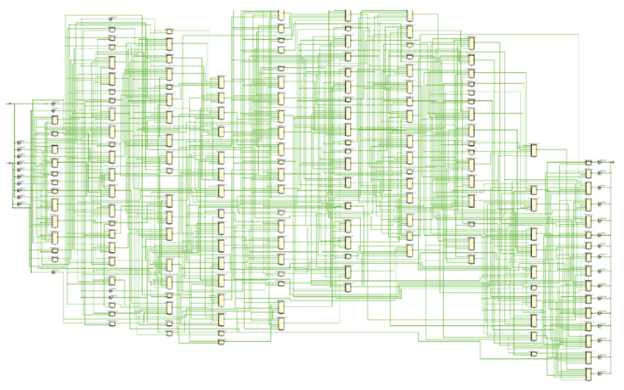


Fig. 5a: 8-bit Squaring Circuit Schematic

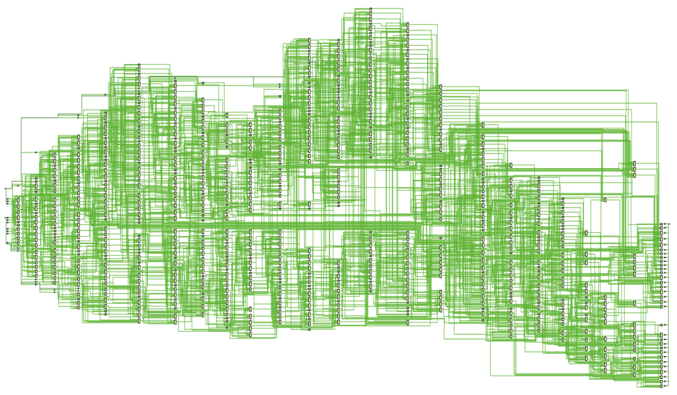


Fig. 5b: 16-bit Squaring Circuit Schematic

Fig. 5.a and Fig. 5.b shows, the schematic representation of 16-bit and 8-bit Squaring Circuit. The schematic diagram is a graphical representation of how components like AND gate, OR gate, half adder, flip-flops, are connected to each other. Schematic diagrams of system are high-level logical representation of the developed design. It is useful to identify whether the inputs, output, and interconnections are routed in correct manner. In Vivado software, schematic diagram can be generated once we are done with synthesizing of the Verilog code. For the developed square circuit LUT used are 595(0.93%) for 16-bit circuit and 127(0.20%) for 8-bit circuit from 64000 available LUTs. Total number of Input/Output port used for developing for 16-bit system are 64(16%) and 8-bit system are 32(8%) from 400 available I/O ports.

Fig. 6.a and Fig. 6.b is the device implementation of 16-bit and 8-bit Squaring Circuit. It is generated after we successfully fini sh synthesis and implementation steps in Xilinx Vivado. Device implementation is the process in which Vivado software maps all the actual physical resources of the FPGA such as LUTs (Look-Up Tables), flip-flops, etc.

Fig. 7.a and 7.b, shows the results for designed 8!bit and 16!bit square circuit using Urdhva-Tiryagbhyam method. Fig 12.a consists of 8_bits input each A and B respectively, and 16_bits output P. Fig 12.b consists of 16_bits input each A and B respectively and 32_bits output P. Simulation results of all the developed square circuits having verified by using testbench inputs. Thus, the results of the developed system are correct. Fig. 8.a ASIC design for 2-bit square circuit, below represents the ASIC layout for 2-bit square architecture using basic logic gates like AND, OR gate and Half Adder circuit.

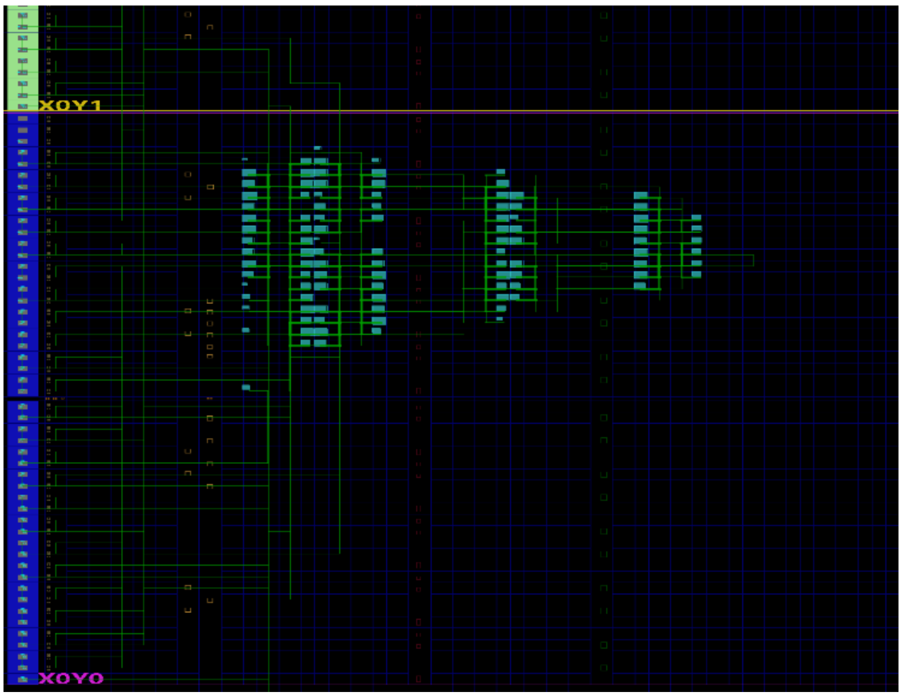


Fig. 6a: 16-bit Squaring Circuit Implementation

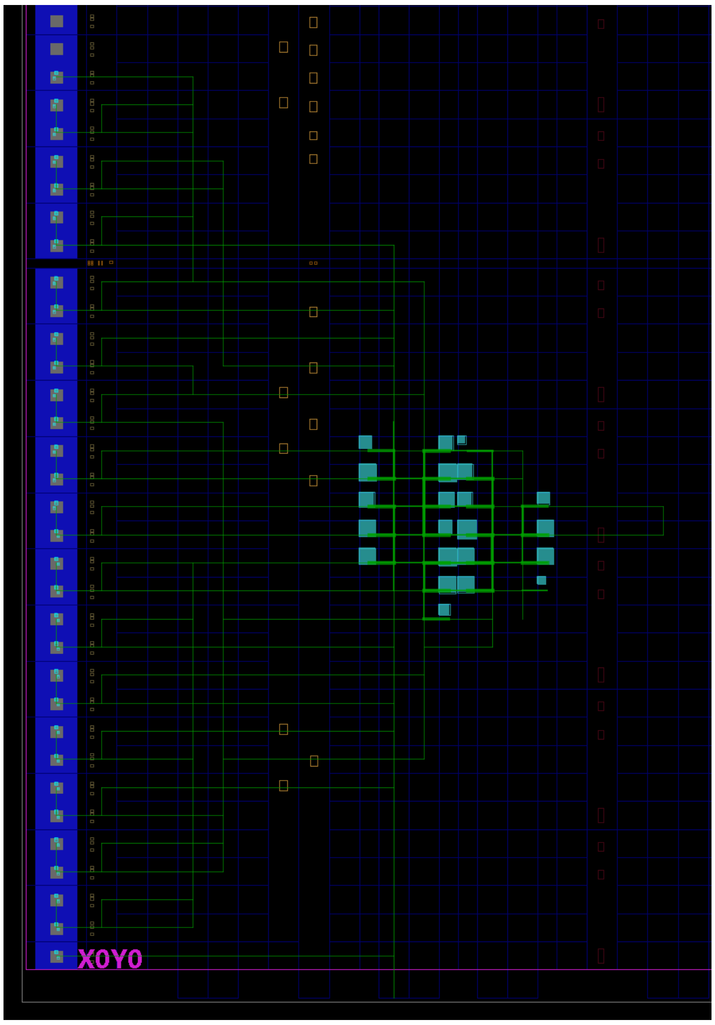
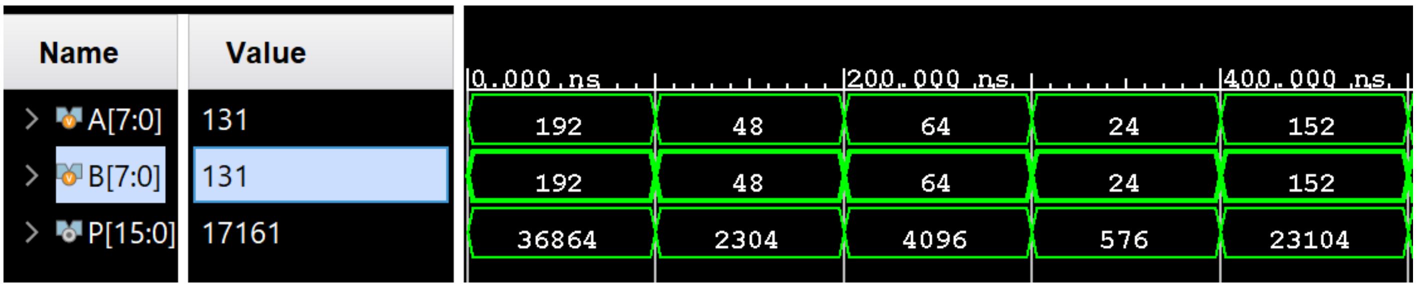
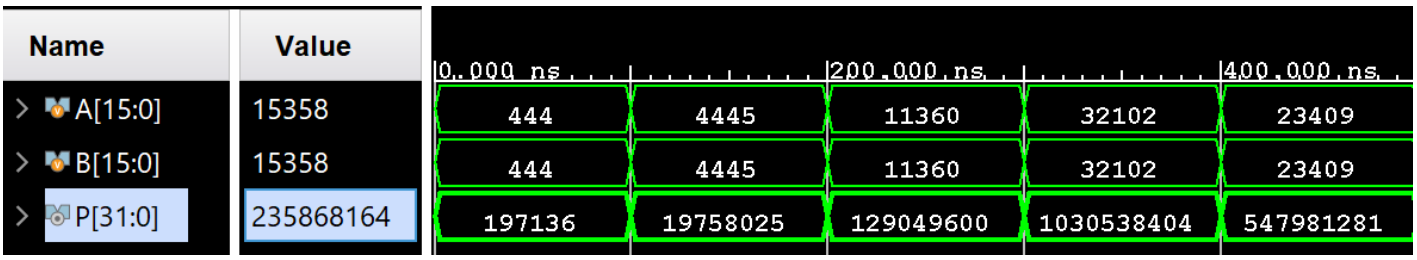


Fig. 6b: 8-bit Squaring Circuit Device Implementation



Name	Value	0.000 ns		200.000 ns		400.000 ns
A[7:0]	131	192	48	64	24	152
B[7:0]	131	192	48	64	24	152
P[15:0]	17161	36864	2304	4096	576	23104

Fig. 7.a 8-bit Squaring Circuit Simulation



Name	Value	0.000 ns		200.000 ns		400.000 ns
A[15:0]	15358	444	4445	11360	32102	23409
B[15:0]	15358	444	4445	11360	32102	23409
P[31:0]	235868164	197136	19758025	129049600	1030538404	547981281

Fig. 7b: 16-bit Squaring Circuit Simulation

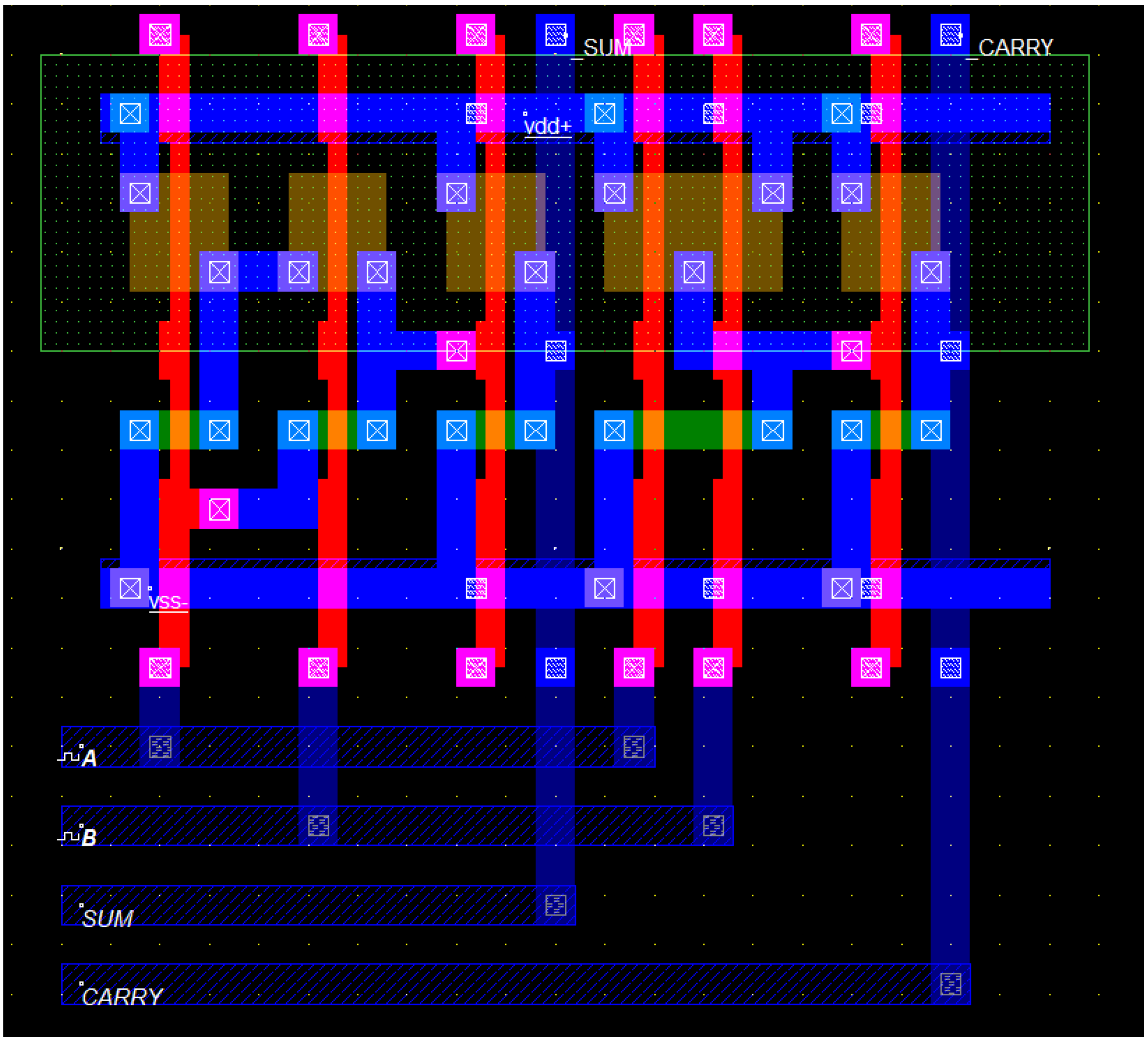


Fig. 8a: ASIC design for 2-bit square circuit

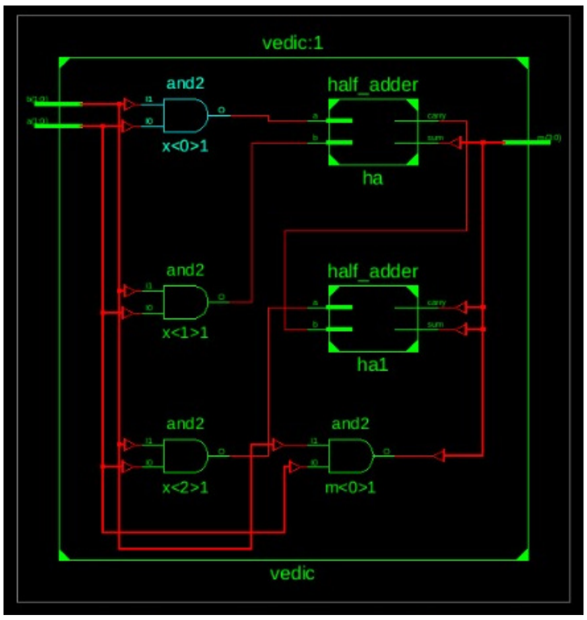


Fig. 8b: RTL schematic of the square circuit

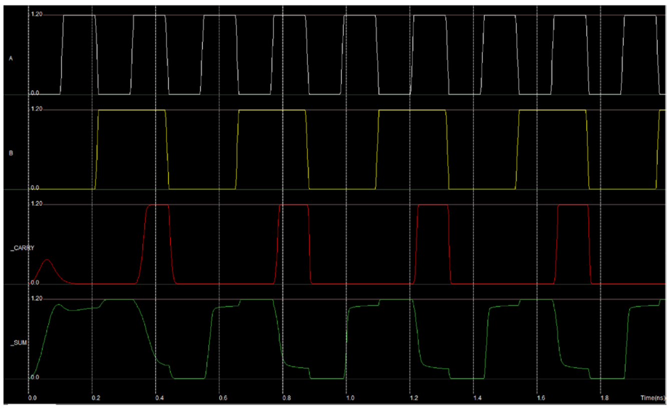


Fig. 8c: ASIC Design Output Waveform

Fig. 8.b, shows the RTL schematic of the square circuit which consist of components like AND gate and HALF ADDER.

Fig. 8.c, verifies the result of the designed 2-bit circuit for various input combinations.

Implementing 2-bit square architecture in ASIC design, will optimize speed as the Urdhva-Tiryagbhyam sutra uses parallel computation method, while, traditional methods like Booth multiplier and Shift n Add method perform more sequential operations. The architecture minimizes the number of sequential operations making the Vedic method more faster.

Conclusion

The paper presents a Vedic mathematic based method (Urdhva-Tiryagbhyam sutra) that is used for development of binary square multiplier having 2bit, 4bit, 8bit and 16bit binary inputs. The simulation result and performance review discussion illustrate that the developed squaring method is more efficient in terms of Total Time Delay and On-Chip Power Consumption. Thus, we can be concluded that using Spartan-7 family give better efficiency in terms of time delay and power optimization than Artix-7. One of the drawbacks of this method can be stated that, the net delay is more than the logical delay. As we implement more lager and complex squaring circuits, there is very large increase in the routing delay as compared to the logic delay. Therefore, to overcome this drawback we can develop a squaring multiplier by integration of Urdhva-Tiryagbhyam sutra and Carry Shift Adder (CSA) method. This method is useful to reduce the net delay required in circuit. The developed multiplier circuit can be utilized for enhancing the performance parameters and reduce the mathematical complexity.

Future Scope

The developed square architecture using Urdhva-Tiryagbhyam method shows less time delay then other Vedic methods like Nikhilam sutra, Yavadunam sutra and methods like Booth Wallace Multiplier. Further, as the number of bits increases like 16 or 32bit, the net delay increases. This increase in net delay can be reduced by designing similar circuit by integration of Carry Shift Adder (CSA) method.

References

[1] S. Shembalkar, S. Dhole, T. Yadav, P. Thakre, 'Vedic Mathematics Sutras -A Review,' International Conference on Recent Trends in Engineering Science and Technology (ICRTEST 2017) Volume: 5 Issue: 1, 2017.

[2] Verma, Pushpalata, and K. K. Mehta. "Implementation of an efficient multiplier based on vedic mathematics using eda tool." International Journal of Engineering and Advanced Technology (IJEAT) 1, no. 5 (2012): 75-79.

[3] Kunchigi, Vaijyanath, Linganagouda Kulkarni, and Subhash Kulkarni. "High speed and area efficient vedic multiplier." In 2012 International conference on devices, circuits and systems (ICDCS), pp. 360-364. IEEE, 2012.

[4] Panda, Siba Kumar, Ritisnigdha Das, and Tapasa Ranjan Sahoo. "VLSI implementation of vedic multiplier using Urdhva–Tiryakbhyam sutra in VHDL environment: A novelty." IOSR Journal of VLSI and Signal Processing 5, no. 1 (2015): 17-24.

[5] Mamdouh, Ahmed, Mbonea Mjema, Gurtac Yemiscioglu, Satoshi Kondo, and Ali Muhtaroglu. "Design of efficient ai accelerator building blocks in quantum-dot cellular automata (QCA)." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 12, no. 3 (2022): 703-712., doi: 10.1109/JETCAS.2022.3202043.

[6] Kuang, Shiann-Rong, and Jiun-Ping Wang. “Design of power-efficient configurable booth multiplier.” IEEE Transactions on Circuits and Systems I: Regular Papers 57, no. 3 (2009): 568-580., doi: 10.1109/TCSI.2009.2023763.

[7] Hussin, Razaidi, Ali Yeon Md Shakaff, Norina Idris, Zaliman Sauli, Rizalafande Che Ismail, and Afzan Kamarudin. “An efficient modified booth multiplier architecture.” In 2008 International Conference on Electronic Design, pp. 1-4. IEEE, 2008., doi: 10.1109/ICED.2008.4786767.

[8] Mottaghi-Dastjerdi, M., Ali Afzali-Kusha, and Massoud Pedram. “BZ-FAD: A low-power low-area multiplier based on shift-and-add architecture.” IEEE Transactions on very large scale integration (VLSI) systems 17, no. 2 (2009): 302-306.

[9] PASHA, SHEIK GOUSE, and AYESHA TARRANUM. “Design and Implementation of Vedic Sutras for Square and Cube Architecture on FPGA.” (2016), vol. 04, no. 11, pp. 1222-1225.

[10] Sriraman, L., K. Saravana 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), pp. 1-5. IEEE, 2013, doi: 10.1109/ICCCNT.2013.6726607.

[11] Vaidya, Sumit, and Deepak Dandekar. “Delay-power performance comparison of multipliers in VLSI circuit design.” International Journal of Computer Networks & Communications (IJCNC) 2, no. 4 (2010): 47-56.

[12] Kunchigi, Vaijyanath, Linganagouda Kulkarni, and Subhash Kulkarni. “Low power square and cube architectures using Vedic Sutras.” In 2014 Fifth International Conference on Signal and Image Processing, pp. 354-358. IEEE, 2014, doi: 10.1109/ICSIP.2014.62.

[13] Deepa, A., and C. N. Marimuthu. “VLSI design of a squaring architecture based on yavadunam sutra of Vedic mathematics.” In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 1162-1167. IEEE, 2020, doi: 10.1109/ICESC48915.2020.9155768.

[14] Kahar, Dravik KishorBhai, and Harsh Mehta. “High speed vedic multiplier used vedic mathematics.” In 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 356-359. IEEE, 2017., doi: 10.1109/ICCONS.2017.8250742.

[15] Akhter, Shamim, Saurabh Chaturvedi, and Shaheen Khan. “A distinctive approach for vedic-based squaring circuit.” In 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 27-30. IEEE, 2020, doi: 10.1109/SPIN48934.2020.9071158.