International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 3, June 2020, pp. 2951~2958
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i3.pp2951-2958

Pipelined Vedic multiplier with manifold adder complexity levels

Ansiya Eshack, S. Krishnakumar
Department of Electronics, School of Technology and Applied Sciences, Edapally, India

Article Info

Article history:

Received May 13, 2019
Revised Jan 3, 2020
Accepted Jan 8, 2020

Keywords:

FPGA
Low power
Pipelining
Vedic multipliers
Urdhava Tiryakbhyam

ABSTRACT

Recently, the increased use of portable devices, has driven the research world to design systems with low power-consumption and high throughput. Vedic multiplier provides least delay even in complex multiplications when compared to other conventional multipliers. In this paper, a 64-bit multiplier is created using the Urdhava Tiryakbhyam sutra in Vedic mathematics. The design of this 64-bit multiplier is implemented in five different ways with the pipelining concept applied at different stages of adder complexities. The different architectures show different delay and power consumption. It is noticed that as complexity of adders in the multipliers reduce, the systems show improved speed and least hardware utilization. The architecture designed using 2x2-bit pipelined Vedic multiplier is then compared with existing Vedic multipliers and conventional multipliers and shows least delay.

Corresponding Author:

Ansiya Eshack
Department of Electronics,
School of Technology and Applied Sciences,
Edapally, Ernakulam 682024, India
Email: ansiya@yahoo.com

1. INTRODUCTION

Speed and power consumption are the two main aspects considered while designing a system in the field of communication [1]. The multipliers (more specifically the adders) which form the major part of these systems affect its speed [2]. The more complex a multiplier or its related adder is, the more is its effect on the speed [3]. Although addition operations are simple, addition of large bits will surely increase the time required for generation of the output. As number of bits of addends increases, time to complete addition also increases, ultimately leading to decrease in throughput of the multiplier. DSPs and communication systems rely heavily on adders and multipliers for processing their data [4, 5]. Existing multiplier systems consume more power and time and are thus, not suitable for DSP and communication systems [6].

This paper explores ways to reduce the bit size of the addends and eventually, reduce the complexity of the adders in the multipliers to increase its processing speed and lower the power consumption of the system. The use of adders with different complexities shows that as adders become less complex, the speed of the system increases. The designed 64-bit Vedic multipliers (VM), using Urdhava Tiryakbhyam (UT) Sutra, employ pipelining at five different adder complexities and the design with the lowest complexity achieves highest throughput. The five designs include the use of 4 thirty-two-bit, 16 sixteen-bit, 64 eight-bit, 256 four-bit and 1024 two-bit pipelined VMs respectively. A reduction in the FPGA hardware used is also observed as VM are employed in the designed system. Faster output generation with low hardware utilization certainly brings down power consumption of this novel system. Thus, the designed multiplier systems are low power consuming and fast output generators with increase in speed and reduction in power consumption as the complexity of the adders decreases. The rest of paper is as follows: Research method is highlighted in Section 2 with explanation on VMs, Pipelined VMs, and Modified VMs. Section 3 gives the results and analysis. Finally, the conclusion is given in Section 4.

2. RESEARCH METHOD

2.1. Vedic multiplier

In the conventional multipliers, 'the carry' most often gets propagated all the way to the most significant bit, when binary multiplication is performed [7]. This is the major reason for higher delay in these multipliers. When Vedic multiplication technique like UT Sutra is used, there is partial product generation and these can be added up in a pipelined fashion [8, 9]. So, VM employing the UT technique for multiplication of two numbers is seen to produce outputs faster than other multipliers by reducing the delay to generate the product [10]. This technique uses 'Vertical and Crosswise' methodology which utilizes least delay, and allows for low hardware usage during processing [11]. Thus UT Sutra allows for parallel processing and provides better performance [12–14].

VMs, involving the UT Sutra, allow a higher order bit multiplication to be computed by breaking them to lower order bits. Generally, an NxN-bit VM is formed by four N/2xN/2-bit VMs [15]. Each of the N/2xN/2-bit VM can again be formed by four N/4xN/4-bit VMs [16]. Thus each higher bit VM can be formed by four lower bit VMs with the lower multipliers having half the bit size of the higher ones. This decomposition can continue until 2x2-bit VMs are reached, beyond which decomposition is not possible. This is as shown in Figure 1. A 2x2-bit UT Sutra based VM, having a 2-bit multiplier 11 and a 2-bit multiplicand 11, follows the steps shown in Figure 2 [17].

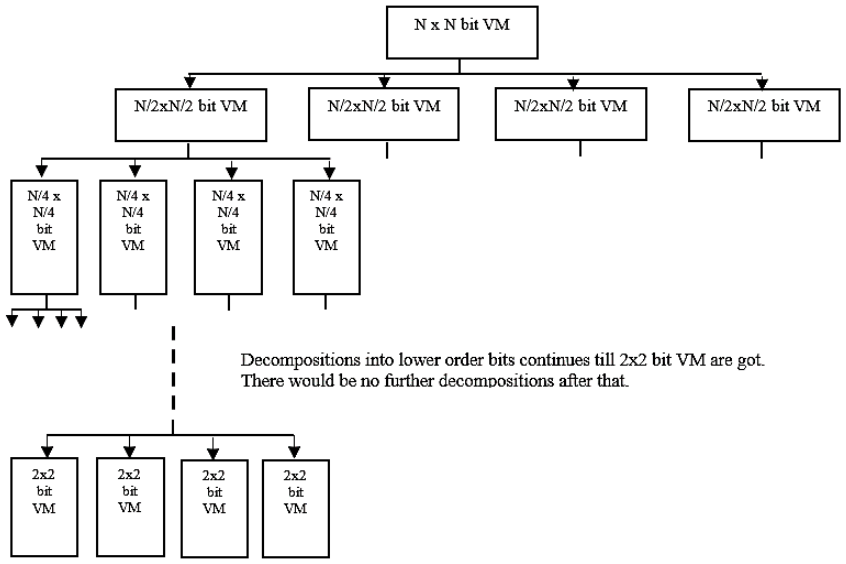

Figure 1. Decomposition of higher order bit VMs into lower order bit VMs

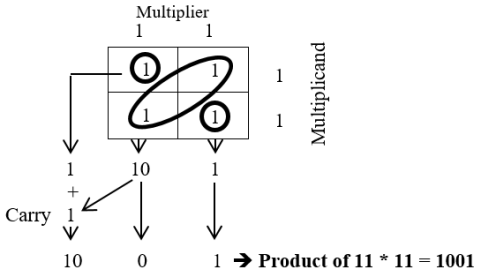

Figure 2. UT Sutra based 2x2-bit VM

2.2. Designed system

This section describes five architectures of the 64x64-bit VM. The diagrammatic representation of this is given in Figure 3(a) to (e). The five architectures with different bit sized pipelined VM are as follows:

a. Multiplier with four 32x32-bit pipelined VM
b. Multiplier with sixteen 16x16-bit pipelined VM

c. Multiplier with sixty-four 8x8-bit pipelined VM
d. Multiplier with two-fifty-six 4x4-bit pipelined VM
e. Multiplier with one-thousand-twenty-four 2x2-bit pipelined VM

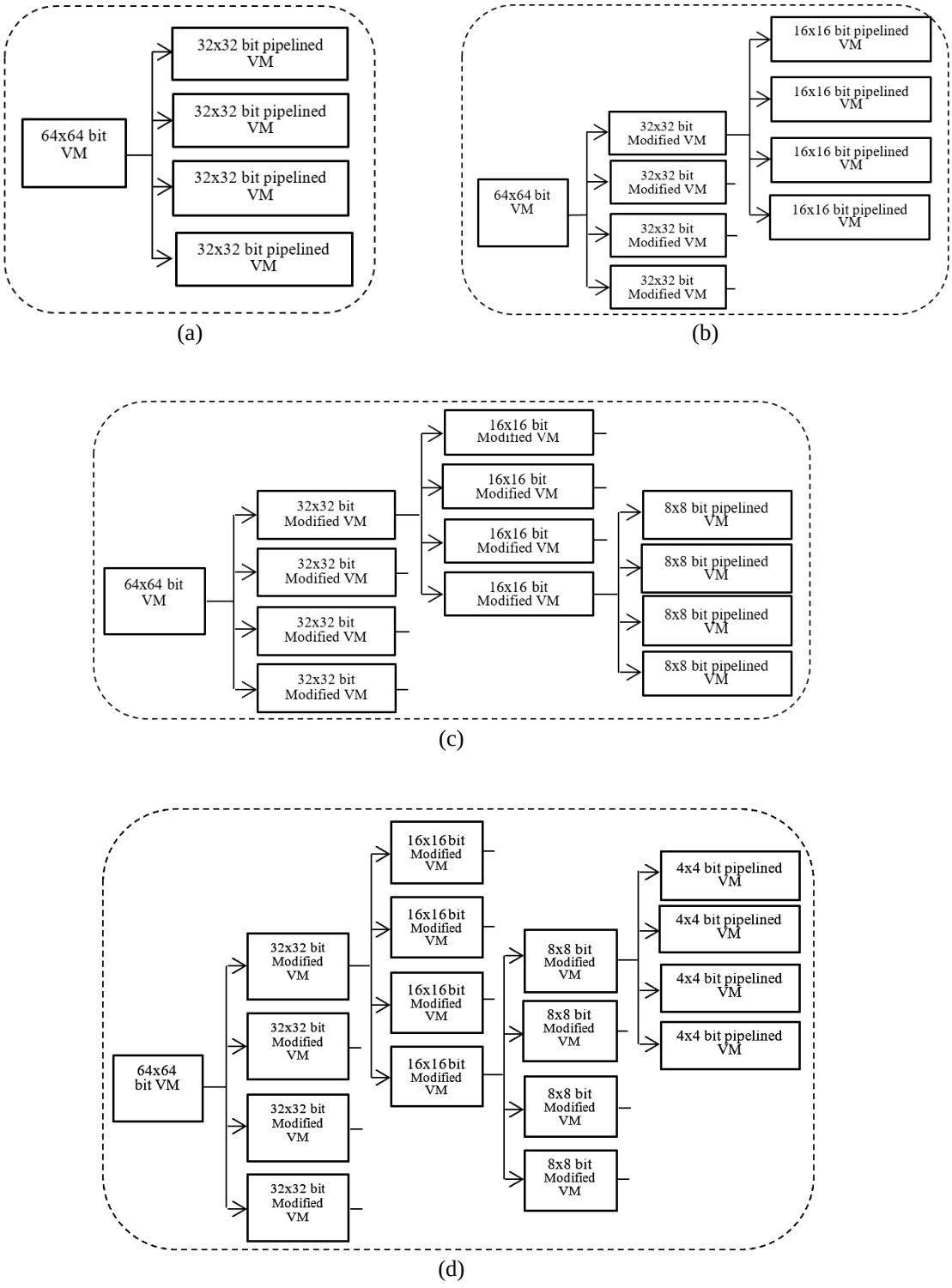

Figure 3. (a) Multiplier with four 32x32-bit pipelined VMs,
(b) Multiplier with sixteen 16x16-bit pipelined VMs,
(c) Multiplier with sixty-four 8x8-bit pipelined VMs,
(d) Multiplier with two-fifty-six 4x4-bit pipelined VMs

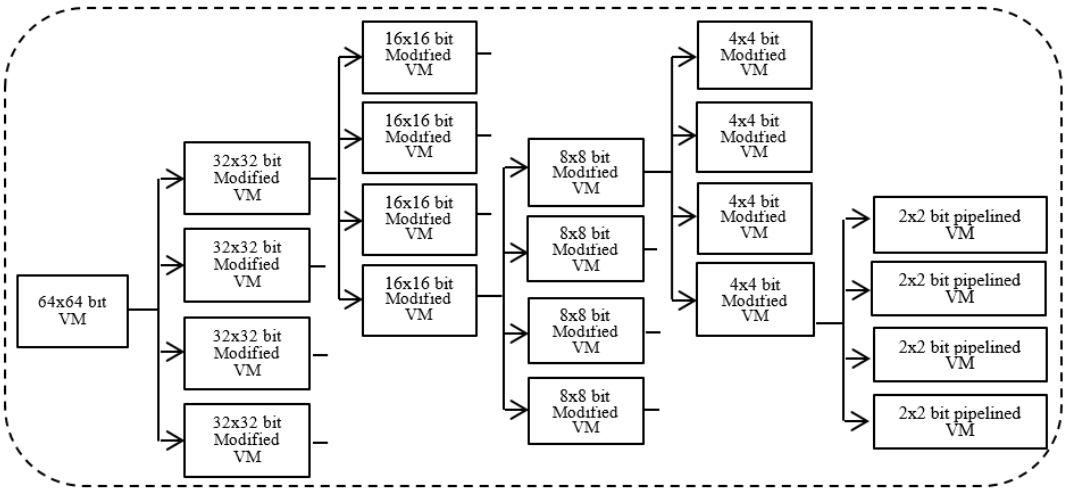

Figure 3. (e) Multiplier with one-thousand-twenty-four 2x2-bit pipelined VMs (*continue*)

These designed multipliers thus use the UT Sutra combined with the pipelining technique. The designs are created by using the pipeline methodology to produce PMs at different multiplier stages. It is observed that as the bit size of these PM becomes low, leading to lesser complexity of the adders involved in them, the time for throughput also decreases. The 64x64-bit VM designed using the 2x2-bit pipelined VM is seen to yield the fastest results when compared with those designed using 4x4-bit PMs, 8x8-bit PMs, 16x16-bit PMs or 32x32-bit PMs. As the bit size of the PM decreases, the adders become less and less complex eventually leading to faster throughput.

2.3.1. Pipelined Vedic multiplier

The 'Vertical and Crosswise' technique followed by UT Sutra is as shown in Figure 2. The bits of the input numbers are multiplied vertically and also in a crosswise manner in different steps and the concatenation of the partial products obtained in these individual steps lead to the output of the multipliers [18]. It can be observed that the UT Sutra supports pipelining and this method is followed for the faster output generation of the multipliers [19–21]. Thus, outputs in multipliers employing the UT Sutra are got faster than non-pipelined multipliers [22].

The general steps followed by the UT Sutra for a 2x2-bit PM is as follows [23]:

Input 1 = a_2a_1 ; Input 2 = b_2b_1
Step 1(vertical): $a_1 * b_1 = a_1b_1$ ➔ P1 ;
Step 2 (cross-wise): $a_1 * b_2 + a_2 * b_1 = a_1b_2 + a_2b_1$ ➔ P2;
Step 3(vertical): $a_2 * b_2 = a_2b_2$ ➔ P3
Output: P3 P2 P1

The general steps followed in this techniques for a 4 x 4-bit PM is as follows:

Input 1 = $a_4a_3a_2a_1$; Input 2 = $b_4b_3b_2b_1$
Step 1(vertical): $a_1 * b_1 = a_1b_1$ ➔ P1 ;
Step 2(cross-wise): $a_1 * b_2 + a_2 * b_1 = a_1b_2 + a_2b_1$ ➔ P2;
Step 3(cross-wise): $a_1 * b_3 + a_2 * b_2 + a_3 * b_1 = a_1b_3 + a_2b_2 + a_3b_1$ ➔ P3;
Step 4(cross-wise): $a_1 * b_4 + a_2 * b_3 + a_3 * b_2 + a_4 * b_1 = a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1$ ➔ P4;
Step 5(cross-wise): $a_2 * b_4 + a_3 * b_3 + a_4 * b_2 = a_2b_4 + a_3b_3 + a_4b_2$ ➔ P5;
Step 6(cross-wise): $a_3 * b_4 + a_4 * b_3 = a_3b_4 + a_4b_3$ ➔ P6;
Step 7(vertical): $a_4 * b_4 = a_4b_4$ ➔ P7
Output: P7 P6 P5 P4 P3 P2 P1

The general steps followed in this techniques for a 8 x 8-bit PM is as follows:

Input 1 = $a_8a_7a_6a_5a_4a_3a_2a_1$; Input 2 = $b_8b_7b_6b_5b_4b_3b_2b_1$
Step 1(vertical): $a_1 * b_1 = a_1b_1$ ➔ P1;
Step 2(cross-wise): $a_1 * b_2 + a_2 * b_1 = a_1b_2 + a_2b_1$ ➔ P2;

Step 3(cross-wise): $a_1 * b_3 + a_2 * b_2 + a_3 * b_1 = a_1b_3 + a_2b_2 + a_3b_1$ ➔ P3;
Step 4(cross-wise): $a_1 * b_4 + a_2 * b_3 + a_3 * b_2 + a_4 * b_1 = a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1$ ➔ P4;
Step 5(cross-wise): $a_1 * b_5 + a_2 * b_4 + a_3 * b_3 + a_4 * b_2 + a_5 * b_1 = a_1b_5 + a_2b_4 + a_3b_3 + a_4b_2 + a_5b_1$ ➔ P5;
Step 6(cross-wise): $a_1 * b_6 + a_2 * b_5 + a_3 * b_4 + a_4 * b_3 + a_5 * b_2 + a_6 * b_1 = a_1b_6 + a_2b_5 + a_3b_4 + a_4b_3 + a_5b_2 + a_6b_1$ ➔ P6;
Step 7(cross-wise): $a_1 * b_7 + a_2 * b_6 + a_3 * b_5 + a_4 * b_4 + a_5 * b_3 + a_6 * b_2 + a_7 * b_1 = a_1b_7 + a_2b_6 + a_3b_5 + a_4b_4 + a_5b_3 + a_6b_2 + a_7b_1$ ➔ P7;
Step 8(cross-wise): $a_1 * b_8 + a_2 * b_7 + a_3 * b_6 + a_4 * b_5 + a_5 * b_4 + a_6 * b_3 + a_7 * b_2 + a_8 * b_1 = a_1b_8 + a_2b_7 + a_3b_6 + a_4b_5 + a_5b_4 + a_6b_3 + a_7b_2 + a_8b_1$ ➔ P8;
Step 9(cross-wise): $a_2 * b_8 + a_3 * b_7 + a_4 * b_6 + a_5 * b_5 + a_6 * b_4 + a_7 * b_3 + a_8 * b_2 = a_2b_8 + a_3b_7 + a_4b_6 + a_5b_5 + a_6b_4 + a_7b_3 + a_8b_2$ ➔ P9;
Step 10(cross-wise): $a_3 * b_8 + a_4 * b_7 + a_5 * b_6 + a_6 * b_5 + a_7 * b_4 + a_8 * b_3 = a_3b_8 + a_4b_7 + a_5b_6 + a_6b_5 + a_7b_4 + a_8b_3$ ➔ P10;
Step 11(cross-wise): $a_4 * b_8 + a_5 * b_7 + a_6 * b_6 + a_7 * b_5 + a_8 * b_4 = a_4b_8 + a_5b_7 + a_6b_6 + a_7b_5 + a_8b_4$ ➔ P11;
Step 12(cross-wise): $a_5 * b_8 + a_6 * b_7 + a_7 * b_6 + a_8 * b_5 = a_5b_8 + a_6b_7 + a_7b_6 + a_8b_5$ ➔ P12;
Step 13(cross-wise): $a_6 * b_8 + a_7 * b_7 + a_8 * b_6 = a_6b_8 + a_7b_7 + a_8b_6$ ➔ P13;
Step 14(cross-wise): $a_7 * b_8 + a_8 * b_7 = a_7b_8 + a_8b_7$ ➔ P14;
Step 15(vertical): $a_8 * b_8 = a_8b_8$ ➔ P15
Output: P15 P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1

The same concept is applied to form 16x16-bit and 32x32-bit PMs. It should be noted that the pipelined VMs are not created from four lower order bit multipliers as is the case with a general VM, i.e., an NxN-bit pipelined VM just follows the vertical and cross-wise steps as listed above and is not built by four N/2-bit VMs.

2.3.2. Modified Vedic multiplier

There are two steps followed in multiplication: partial product generation and partial product accumulation [24, 25]. The changes in the processing method of these two steps result in difference in throughput of the multiplier. In the designed VMs, the method of partial product accumulation is exploited, as described below, to gain faster outputs. Consider a 4x4-bit VM formed from four 2x2-bit VMs. The two 4–bit numbers written as $a_4a_3a_2a_1$ and $b_4b_3b_2b_1$ can be partitioned into four 2-bit numbers a_4a_3, a_2a_1, b_4b_3 and b_2b_1. These 2-bit numbers serve as the inputs to the 2x2-bit VMs. The cross-wise and vertical technique of UT Sutra is used to generate the products of these 2x2-bit VMs. The partial products (PP) of the 4x4-bit VM is as shown in Figure 4. These partial products which are four-bit wide each are then added together to form the final product.

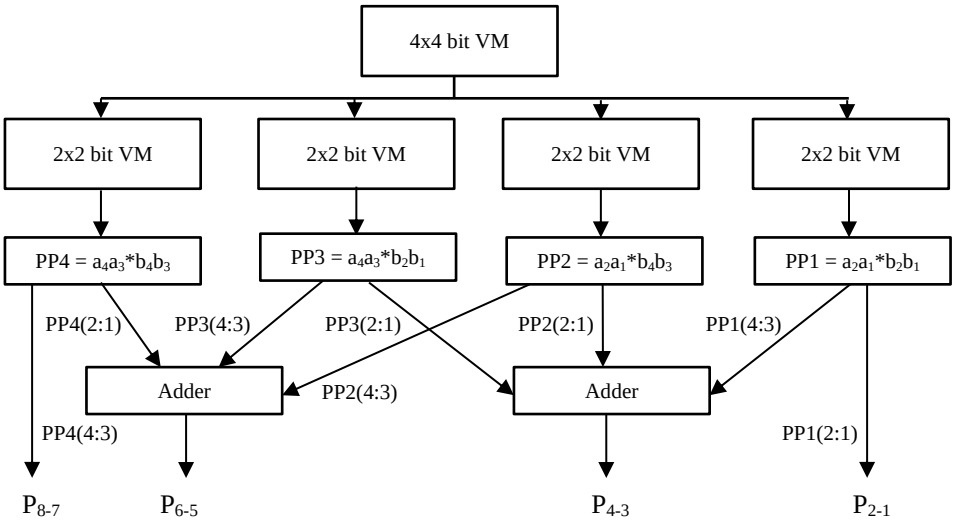

Figure 4. Modified 4x4-bit VM with the generated partial products and its accumulation

The modified 4x4-bit VM is thus constructed from four 2x2-bit VMs by using the principle of product accumulation as given below. The four partial products PP1-PP4 and the final product can be written as follows:

PP1	=	$a_2a_1 * b_2b_1$
PP2	=	$a_2a_1 * b_4b_3$
PP3	=	$a_4a_3 * b_2b_1$
PP4	=	$a_4a_3 * b_4b_3$
Final product	=	P_{8-7} P_{6-5} P_{4-3} P_{2-1}

where P_{8-7} = PP4(4:3), the first two bits of PP4, P_{6-5} = PP4(2:1) + PP3(4:3) + PP2(4:3), the sum of the first two bits of PP3 and PP2 & the last two bits of PP4, P_{4-3} = PP3(2:1) + PP2(2:1) + PP1(4:3), the sum of first two bits of PP1 and last two bits of PP3 & PP2 and P_{2-1} = PP1(2:1), the last two bits of PP1. The carry of P_{4-3} is added to the sum of P_{6-5} and that of P_{6-5} is added to P_{8-7}. Modified 8x8-bit, 16x16-bit and 32x32-bit VMs all follow the same principle of partial product accumulation for final product generation. This modification allows for faster generation of the products and thus reduces the delay of the system.

3. RESULTS AND ANALYSIS

Five different architectures of 64x64-bit VMs, with varying bit sizes of the pipelined VMs, have been designed and implemented. The designs consist of modified VMs and pipelined VMs at different adder complexities. The bit size of the pipelined VMs has a direct relation with the complexity of the adders and the throughput of the system. It is observed that as multiplier and the adder complexity reduces, the delay for generating the product also decreases. Figure 5 gives the delay for the five different designs of the 64x64–bit multiplier architectures.

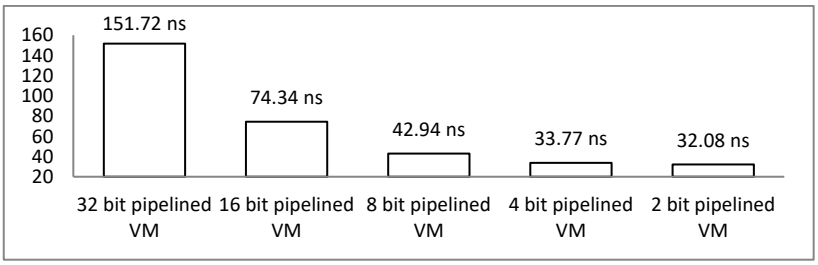

Figure 5. Delay for the five different architectures of 64x64-bit VM

The number of look-up tables (LUTs), of the FPGA, utilized by the five different pipelined architectures of 64x64-bit VM are as given in Table 1. It is seen that the architecture with 32x32-bit pipelined VM utilizes the highest number of LUTs. The utilization of LUTs decreases as the level of pipelining reduces and the multiplier with 2x2-bit pipelined VM has the least number of LUTs. The reason for increased throughput of the architectures with decrease in bit size of pipelined VM is due to the fact that as bit size reduces, the complexity of adders also decreases. The volume of additions and the size of the addends decreases and this leads to decrease in the time to produce the output. This is the reason for reduction in the number of LUTs used by the multiplier designs. The delay of the designed multiplier system employing 2x2-bit pipelined VM thus yields the best result in terms of hardware utilization of FPGA and system speed. This architecture design is compared with other existing 64x64-bit VMs as shown in Figure 6. The designed 64x64-bit multiplier system with least adder complexity, i. e. employing the 2x2-bit pipelined VM, shows the least delay when compared with the existing multipliers. Figure 7 shows the delay comparison between the conventional multipliers and the designed multiplier system.

Table 1. LUT utilization of the five different architectures of 64x64-bit VM

Types of VM	32 bit pipelined VM	16 bit pipelined VM	8 bit pipelined VM	4 bit pipelined VM	2 bit pipelined VM
No. of Look-up tables	13151	12805	12255	11905	11378

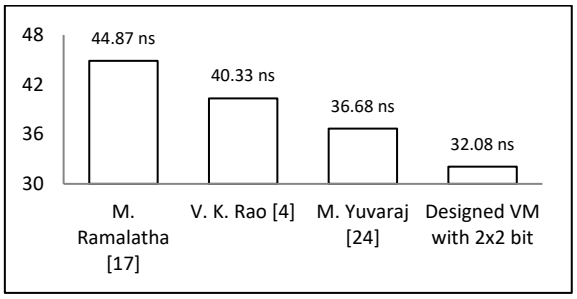

Figure 6. Delay comparison of designed system with existing 64x64-bit VMs

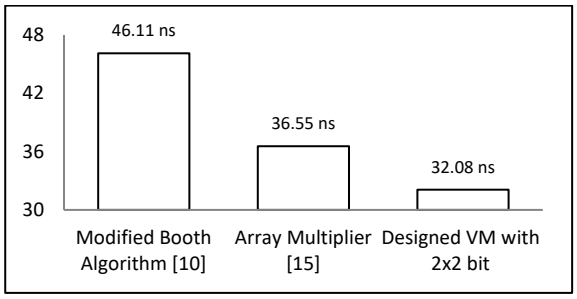

Figure 7. Delay comparison of designed system with conventional 64x64-bit multipliers

4. CONCLUSION

The paper proposes the design of five system architectures of the 64x64-bit VM. The designs show that the system designed using the 2x2-bit pipelined modified VM gives the highest throughput. Also this 2x2-bit pipelined modified VM design utilizes least amount of FPGA hardware out of all the five designs. This proves that as the bit size of the pipelined VM reduces, the complexity of the system reduces, and this in turn increases its computation speed and decreases its power consumption. The best architecture among the five designs of the 64x64-bit VM has been compared with other existing similar bit-sized VMs and standard multipliers and is observed to have the least delay.

REFERENCES

[1] M. Jhamb, Garima, and H. Lohani, "Design, implementation and performance comparison of multiplier topologies in power-delay space," *International Journal Engineering Science and Technology*, vol. 19, no. 1, pp. 355–363, Mar 2016.

[2] C. Chaitanya, C. Sundaresan, P. Venkateswaran, and K. Prasad, "Asic design of low power-delay product carry pre-computation based multiplier," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 2, pp. 845–852, 2019.

[3] T. Gupta and J. B. Sharma, "Han–Carlson adder based high-speed Vedic multiplier for complex multiplication," *Microsystem Technologies*, vol. 24, no. 9, pp. 3901–3906, Sep 2018.

[4] V. K. Rao and K. Lavanya, "An area efficient Q-format multiplier with high performance for digital processing applications," in *IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, pp. 137–141, 2015.

[5] M. Barakat, W. Saad, and M. Shokair, "Implementation of efficient multiplier for high speed applications using FPGA," in *13th International Conference on Computer Engineering and Systems (ICCES)*, pp. 211–214, 2018.

[6] K. D. Rao, C. Gangadhar, and P. K. Korrai, "FPGA implementation of complex multiplier using minimum delay Vedic real multiplier architecture," in *IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON)*, pp. 580–584, 2016.

[7] E. Prabhu, H. Mangalam, and P. R. Gokul, "A delay efficient Vedic multiplier," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 89, no. 2, pp. 257–268, Jun 2019.

[8] Y. S. Rao, M. Kamaraju, and D. V. S. Ramanjaneyulu, "An FPGA implementation of high speed and area efficient double-precision floating point multiplier using Urdhva Tiryagbhyam technique," in *Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG)*, pp. 271-276, 2015.

[9] A. Sai Ramya, B. S. S. V. Ramesh Babu, E. Srikala, M. Pavan, P. Unita, and A. V. S. Swathi, "Performance of optimized reversible Vedic multipliers," in *Comp. Comm., Netw. and Int. Sec.*, pp. 587–593, 2017.

[10] S. G. M. E. Atre and M. A. M. Alshewimy, "Design and implementation of new delay-efficient/configurable multiplier using FPGA," in *12th Int. Conference on Computer Engineering and Systems (ICCES)*, pp. 8–13, 2017.

[11] C. Chaitanya, C. Sundaresan, P. Venkateswaran, and K. Prasad, "Design of modified booth based multiplier with carry pre-computation," *Ind. Jour. of Elec.l Eng. and Comp. Sci.*, vol. 13, no. 3, pp. 1048–1055, 2019.

[12] S. L. G. Moses and M. Thilagar, "VHDL implementation of high performance RC6 algorithm using ancient Indian Vedic mathematics," in *3rd International Conference on Electronics Computer Technology*, vol. 4, pp. 140–143, 2011.

[13] P. Tuwanuti and N. Thongbai, "Implementation of Vedic multiplier technique on multicore processor," in *TENCON 2014 - 2014 IEEE Region 10 Conference*, pp. 1–6, 2014.

[14] K. Pichhode, M. D. Patil, D. Shah, and B. C. Rohit, "FPGA implementation of efficient Vedic multiplier," in *International Conference on Information Processing (ICIP)*, pp. 565–570, 2015.

[15] T. Gupta and J. B. Sharma, "A CSA-based architecture of Vedic multiplier for complex multiplication," in *Ambient Communications and Computer Systems*, pp. 41–52, 2018.

[16] Jinesh S, Ramesh P, and J. Thomas, "Implementation of 64Bit high speed multiplier for DSP application- based on Vedic mathematics," in *TENCON 2015 - 2015 IEEE Region 10 Conference*, pp. 1–5, 2015.

[17] M. Ramalatha, K. D. Dayalan, P. Dharani, and S. D. Priya, "High speed energy efficient ALU design using Vedic multiplication techniques," in *2009 International Conference on Advances in Computational Tools for Engineering Applications*, pp. 600–603, 2009.
[18] 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.
[19] V. Jayaprakasan, S. Vijayakumar, and V. S. Kanchana Bhaaskaran, "Evaluation of the conventional vs. ancient computation methodology for energy efficient arithmetic architecture," in *International Conference on Process Automation, Control and Computing*, pp. 1–4, 2011.
[20] K. Morghade and P. Dakhole, "Design of fast Vedic multiplier with fault diagnostic capabilities," in *International Conference on Communication and Signal Processing*, pp. 0416–0419, 2016.
[21] A. Eshack and S. Krishnakumar, "Low power 32 x 32 – bit reversible Vedic multiplier," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 10, Aug 2019.
[22] V. Gowreesrinivas and P. Samundiswary, "Resource efficient single precision floating point multiplier using Karatsuba algorithm," *Ind. Jour. of Elec. Eng. and Inf. (IJEEI)*, vol. 6, no. 3, pp. 333–342, 2018.
[23] A. Jais and P. Palsodkar, "Design and implementation of 64 bit multiplier using Vedic algorithm," in *International Conference on Communication and Signal Processing (ICCSP)*, pp. 0775–0779, 2016.
[24] M. Yuvaraj, B. J. Kailath, and N. Bhaskhar, "Design of optimized MAC unit using integrated Vedic multiplier," in *International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, pp. 1–6, 2017.
[25] A. Eshack and S. Krishnakumar, "Reversible logic in pipelined low power Vedic multiplier," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 3, pp. 1265–1272, Dec 2019.

BIOGRAPHIES OF AUTHORS

Ansiya Eshack received her B.Tech from MES College of Engineering, Kuttipuram and M.Tech from Model Engineering College, Kochi. She is presently an Associate Professor at KMEA Engineering College, Ernakulam. She is working on her PhD in Electronics at Mahatma Gandhi University, Kottayam. Her research interests include Low Power VLSI Design, Embedded Systems, Communication Systems and DSP Applications.

S. Krishnakumar completed his M.Sc. in Physics with Electronics specialization in 1987 from Mahatma Gandhi University, Kottayam. He was awarded Ph.D. in Thin Film Devices in 1995 from Mahatma Gandhi University, Kottayam. He received his M.Tech. in Computer Science from Allahabad Agricultural Institute (Deemed University) in 2006 and his MCA from IGNOU in 2010. He served as the Regional Director at the School of Technology and Applied Sciences (STAS), during the period 2014-17. Currently he is working in the STAS, Mahatma Gandhi University Research Centre, Edapally. His research interest fields include ANN, VLSI, Analog circuit design and Image processing. Dr. S. Krishnakumar is an Associate member of Institute of Engineers, India. He was a member of Board of studies of University of Calicut and a member of Academic council of Mahatma Gandhi University, Kottayam.