International Journal of Hybrid Information Technology
Vol.11, No.4 (2018), pp.27-34
http://dx.doi.org/10.21742/ijhit.2018.11.4.04

Comparative Review of Binary Multiplier Systems

Marcus Lloyde George[1] and Geetam Singh Tomar[2]

[1]Dept. of Electrical and Computer Engineering, University of West Indies, St. Augustine, Trinidad and Tobago, India
[2]THDC Institute of Hydropower Engineering and Technology, Bhagirathipuram, Tehri 249 124, India
[1]marcus.george@sta.uwi.edu, [2]gstomar@ieee.org

Abstract

This paper presents a comprehensive comparative review of existing 8-bit, 16-bit and 24-bit binary multiplier architectures and seeks to identify engineering techniques involved in their development. A comparison of the performance of these systems in terms of metrics such as path delay, hardware utilization and even power consumption in some cases are carried out. Weaknesses in the systems reviewed along with possible gaps in the area of research are identified. This paper also serves to identify several recommendations and considerations for the development of a multi-precision binary multiplier system capable of treating the weaknesses of multiplier systems identified.

Keywords: *Binary multiplier, Path delay, Hardware utilization, Multi precision multiplier*

1. Introduction

In any computing device Arithmetic Logic Units (ALUs) is the base unit to perform arithmetic and logic operations on operands according to instructions to perform various arithmetic operations such as multiplication, division, addition and subtraction [1]. Some processors are divided into two units, an Arithmetic Unit (AU) and a Logic Unit (LU). Some processor systems consist of multiple ALUs, e.g. one used for fixed-point operations and one used for floating-point operations. Multiplication is the most frequently used operation in ALUs. It allows one number to be scaled by another number.

The multiplication process consumes significant time compared to other arithmetic operations used in basic mathematical computations [2]. More is computation more is power consumption and thus power management has also become extremely important especially in the case of portable electronic systems [3]. Large power dissipation results in the chip having a higher temperature profile and as such, this affects the performance of the chip [3]. According to [3] the multiplier is a major power dissipation source and at the same time, high speed multiplication is a major requirement for high performance computing. As a result, it is beneficial in the area of mathematical computation to present faster and more efficient mechanisms for implementing mathematical operations which also can utilize less power. Before this can be achieved it may be useful to perform a comprehensive review of existing binary multiplier systems in such a way that can advise further evolution of the area of binary multiplication.

Article history:
Received (May 12, 2018), Review Result (August 20, 2018), Accepted (October 22, 2018)

2. Review of related work

In [4] authors reported a study of five high speed binary multipliers: Booth Multiplier, Modified Booth Multiplier, Vedic Multiplier, Wallace Multiplier and Dadda Multiplier. In Booth Multiplication number of partial products generated is equivalent to the number of bits of the multiplier. The generation of partial products and the corresponding computation of sums are done in parallel. The partial products are obtained as presented in [1]. The Wallace multiplier operates using two: first, the numbers are converted to binary after which the partial products are generated. The Dadda multiplier operates in two stages: in the first stage formation of the partial product matrix takes place and then in the second stage the matrix must be broken down into rows which are added using carry-propagating adders. According to [4] the Modified Booth multiplier reduces the number of partial products generated compared to other multipliers while the Dadda multiplier minimizes the number of adders used when compared to the Wallace multiplier. Therefore, [4] proposed a new multiplier architecture called the Booth Dadda Algorithm which combined the benefits of the Modified Booth Multiplier and Dadda Multiplier. As such [4] indicated that this proposed architecture will reduce the hardware utilization because of the reduction of the number of adders used, and also increased its speed because of the reduction in the number of partial products formed.

Authors in [5] presented an efficient method for partial product reduction for the binary multiplier. This system was designed for the 16nm TSMH CMOS technology and was done using the Tanner EDA 14.1 development tool. [5] presented a study of several partial product techniques such as Wallace and Dadda schemes. According to [5] the Dadda multiplier performed less reductions than the Wallace multiplier. [5] also claims that the Dadda multiplier consumed less power and area than the Wallace multiplier. And then [5] presented several compressors, eg. 4 to 2 compressor which introduced a horizontal path as a result of limited propagation of the carry of the multiplier unit. [5] produced a gate level redesign of this compressor for maximizing performance. Two operating modes were considered: active mode and sleep mode. [5] examined 3 to 2, 4 to 2, 5 to 2 and 7 to 2 compressors and their performance. According to [5] the compressors with sleep transistors consumed on average 47.35% less power than the same architecture of compressors without sleep transistors [Table 1].

Table 1. Power Consumption of the compression algorithm component of binary multiplier with and without sleep transistors

Compressor	Without sleep transistor (μwatt)	With sleep transistor (μwatt)	Percentage Difference (%)
3 to 2	86.97	47.26	45.65
4 to 2	173.58	90.47	47.88
5 to 2	259.62	136.67	47.36
7 to 2	428.86	220.66	48.54

Source: Data from [5]

In [5], authors have reported less path delay in compressors with sleep transistors while comparing with architecture containing compressors without sleep transistors. In [3], the authors proposed an 8x8 hybrid tree multiplier system having the combination of Dadda and Wallace strategies. The Dadda multiplier has partial products, which were divided into four parts and partial product addition reduction was reported to have performed on each part. The reported approach included the assignment of the name group1-4 to the four decomposition blocks and each group was assigned either a 4x4 Dadda or 4x4 Wallace algorithms to be used for compressing the partial products. In [6], the authors presented a high speed multiplier system that was based on Vedic mathematics. [18] does a comparison of the implemented multiplier with the conventional binary multiplier in 8-bit, 16-bit and 32-bit modes. The multipliers were designed and implemented using VHDL for the target device Spartan 3 XC3S50a-4tq144 using Xilinx 14.7 ISE. Table IV summarizes the path delays of the conventional binary multiplier and the Vedic multiplier in 8, 16 and 32-bit modes. The delay has been compared as shown in [Table 2].

Table 2. Path delay comparison for varying bit-sizes

	Path Delay / ns		
	8-bit	16-bit	32-bit
Conventional	11.0	11.0	23.5
Urdhava	5.5	6.0	7.5
Nikhilam Sutra	6.5	6.0	3.5

Source*: Data from [6]*

In [6] authors have reported significant improvement in delay using Urdhava and Nikhilam Sutra algorithms while comparing with the conventional multiplier at 8, 16 and 32-bit modes. Authors in [7] have compared 32-bit Vedic multiplication with the conventional binary multiplier by implementing systems on Xilinx Nexys 3 Spartan 6 FPGA using Xilinx ISE 13.4. The Vedic multiplier system implemented had a path delay of 168.43ns while the conventional multiplier implemented had a path delay of 19.29ns. Authors in [8] reported the implementation of an efficient reduction scheme of tree multipliers on FPGAs. The system implemented was not a binary multiplier system but rather a reduction scheme for partial product reduction using a library of *m:n* counters of varying sizes to maximize the partial product reduction operation. 32-bit multiplier scheme was implemented in Verilog on Xilinx ISE suite and targeted the Xilinx Spartan-6 platform. In [9] authors reported a low power, high speed 16-bit binary multiplier implementation using Vedic mathematics. In this design, the construction of a 2x2 multiplier block which is used in the construction of a 4x4 multiplier block was reported with an 8x8 multiplier block. The required 16x16 multiplier block was constructed using the 8x8 multiplier blocks. The system had a path delay of 27.15 ns and utilized 14,382 transistors in its implementation. In [10] authors reported a high speed, area efficient 16-bit Vedic Multiplier and 32-bit Booth Recoded Wallace Tree multiplier. The system was implemented in Verilog HDL and synthesized for Xilinx Virtex 6 FPGA where path delays of 13.45ns and 11.57ns respectively were reported. However, hardware utilization was not stated in the work. In [11] authors presented the design of a 24-bit binary multiplier for use in the implementation of a 32-bit floating-point multiplier. Vedic Mathematics was utilized in the implementation. [11][12] indicated that the path delay of this multiplier was 16.32ns. The hardware utilization was not stated. In [13] authors have reported a proposal of an efficient strategy for unsigned binary multiplication to improve the path delay and area. The reported system was reported to have utilized a combination of the Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm in implementing the required system [19][20]. The Urdhva-Tiryagbhyam algorithm on the other hand is was reported to have been best suited for the multiplication of large numbers and the strategy is a divide and conquer one in which the numbers are divided into their most significant and least significant half after which multiplication is performed. The delay of each segment was reported was with improvement. The reported 8-bit, 16-bit and 24-bit versions outperform their counterparts when it came to path delay while the 32-bit did not perform better than its 32-bit counterparts. [Table 3] summarizes the performance of the various multiplier systems on the Virtex 4 FPGA platform.

Table 3. Summary of performance for multiplier systems implemented

	8-bit Multiplier	16-bit Multiplier	24-bit Multiplier	32-bit Multiplier
Slices	113	410	972	1389
LUTs	120	451	1018	1545
IOBs	33	65	97	129
Delay	9.396ns	11.514ns	12.996ns	13.141ns
fmax (MHz)	274.469	248.964	226.508	209.606
Logic Levels	14	22	31	39

Source: Data from [13]

In [14], authors reported a design of an area-efficient multiplier using modified carry select adders (CSLAs), which was based on crosswise and vertical Vedic multiplier algorithms. The reported modified CSLA consisted of three stages - half sum generation, final sum generation and carry generation. This modified CSLA was then reported to have been used in an 8-bit Vedic Multiplier and it was claimed that the path delay of the Vedic Multiplier was 45.68ns while the hardware utilization was 1380 gates. In [2], it was reported that the design of a high speed 32-bit multiplier architecture based on Vedic mathematics by adjustment of the partial products using concatenation approach was an improved design [21]. The system was implemented on the Xilinx Spartan-3E device XC3S500e-fg320-5. This design was reported to have path delays of 13.43ns, 17.62ns and 22.47ns respectively. In [15] authors reported area efficient 8x8 and 16x16 multiplier systems using Vedic mathematics to improve performance. The system was implemented in Verilog HDL and synthesized on Xilinx ISE 12.2 for the target device Spartan 3E, XC3S500-5FG320. The systems were implemented with BEC adders [15]. [Table 4] summarizes the performance of the various multiplier systems implemented.

Table 4. Summary of Performance for Multiplier Systems Implemented

Logic Utilization	Vedic 8×8 BEC	Vedic 16×16 BEC	Array 8×8	Array 16×16
Power in mw	83.79	86.91	83.79	86.22
no.of Slice Registers	347 /9312	493 /9312	347 /9312	493 /9312
no.of 4-input LUTs	497 /9312	1243 /9312	411 /9312	844 /9312
no.of IOBs	34 /232	66 /232	34 /232	66 /232
Delay in ns	23.18	38.82	24.88	61.49
Memory in KB	138728	139624	136956	200524

Source: Data from [15]

All reviewed binary multiplier systems were compiled into four tables - 8-bit, 16-bit, 24-bit and 32-bit multiplier systems. [Tables 5-8] summarize the comparison of latencies amongst the multiplier systems reviewed in this section.

Table 5. Summary of path delay for various 8-bit binary multipliers

Source	Multiplier	Path Delay / ns
[6] - Spartan 3 (XC350A-4TQ144)	Conventional Multiplier	11.00
	Urdhava Vedic Multiplier	5.50
	Nikhilam Sutra Vedic Multiplier	6.25
[13] - Virtex 4 (XC4VFX12-10FF668)	Vedic Multiplier	9.40
[14] - Spartan 3 (XC3S1000-4FG256)	Vedic Multiplier	45.68
[2] - Spartan 3E (XC3S100E-5TQ144)	Vedic Multiplier	13.43
[15]	Vedic Multiplier	23.18

Table 6. Summary of path delay for various 16-bit binary multipliers

Source	Multiplier	Path Delay / ns
[9] - FPGA Platform Not Stated	Vedic Multiplier	27.15
[10] - Virtex 6 (XC6VLX75T-3FF484)	Vedic Multiplier	13.45
[6] - Spartan 3 (XC3S1000-4FG256)	Conventional Multiplier	11.00
	Urdhava Vedic Multiplier	6.00
	Nikhilam Sutra Vedic Multiplier	6.00
[13] - Virtex 4 (XC4VFX12-10FF668)	Vedic Multiplier	11.51
[2] - Spartan 3E (XC3S100E-5TQ144)	Vedic Multiplier	17.62
[15]	Vedic Multiplier	38.82

Table 7. Summary of path delay for various 24-bit binary multipliers

Source	Multiplier	Path Delay / ns
[11]	Vedic Multiplier	16.32
[13] - Virtex 4 (XC4VFX12-10FF668)	Vedic Multiplier	13.00

Table 8. Summary of path delay for various 32-bit binary multipliers

Source	Multiplier	Path Delay / ns
[10] - Virtex 6 (XC6VLX75T-3FF484)	Booth Recoded Wallace Tree Multiplier	11.57
[8] - Spartan 6 (XC6SLX9-2FTG256)	Efficient Multiplier Scheme	15.49
[7] - Spartan 6 (XC6SLX4-3TQ6144)	Conventional Multiplier	19.29
	Vedic Multiplier	168.43
[6] - Spartan 3 (XC3S1000-4FG256)	Conventional Multiplier	23.50
	Urdhava Vedic Multiplier	7.50
	Nikhilam Sutra Vedic Multiplier	3.50
[13] - Virtex 4 (XC4VFX12-10FF668)	Vedic Multiplier	13.14
[2] - Spartan 3E (XC3S100E-5TQ144)	Vedic Multiplier	22.47

3. Considerations for development of novel binary multiplier system

The development of a novel binary multiplier system capable of eliminating (or at least minimizing) the effects of the weaknesses of existing binary multiplier systems is a viable consideration for the benefit of digital electronic systems. This system can especially benefit the existing floating-point multiplier system.

Most of the multiplier systems reviewed in this paper carried out the processes of partial product generation, partial product storage and partial product reduction. For example, multipliers developed in [1][4][5] perform partial product reduction using Wallace or Dadda multipliers, thereafter the results are compressed using compressors. Others like [3] use a combination of multiplier and compressor techniques to perform the partial product reduction segment. Most of the existing systems reviewed utilized Vedic mathematics for partial product generation. Multiplier systems in [2][12][14][16][23] for instance developed Vedic multipliers by utilizing smaller multipliers as building blocks to developing bigger multipliers. For instance, the construction of a 2x2 multiplier block is used in the construction of a 4x4 multiplier block, after which an 8x8 multiplier block is constructed. Some multiplier systems such as that in [10] concurrently added the partial products during the multiplication operation, hence reducing the delay at the expense of hardware utilization. Others like [17][22][26] a technique for low power operation which utilized both Sleep and BIVOS techniques. When starting from the columns of least significance, some columns are switched to sleeping mode while the remaining is supplied with a biased voltage. This method resulted in a loss in accuracy. There is a need for the development of a multiplier system that is capable of accumulating partial products as they are generated, hence reducing the delay at the expense of hardware utilization. This is a viable consideration [24][25].

None of the existing binary multiplication systems analyzed past multiplication operations to further reduce the path delay of the multiplication operation. Focusing on previous multiplication operations could benefit future multiplications, hence preventing the system from having to undergo lengthy partial product generation operations especially when operands (multiplicand and multiplier) bit widths are very large - where the number of partial products can become very large. The inclusion of a novel system called Mantissa Similarity Investigation

(MSI) in the floating point multiplier system which can capable of further reduction in path delay is also a viable consideration.

4. Conclusion

This paper presented a comprehensive comparative review of existing 8-bit, 16-bit and 24-bit binary multiplier architectures, and also identified engineering techniques involved in their development. A comparison of the performance of these systems in terms of metrics such as path delay, hardware utilization and even power consumption in some case were carried out. Weaknesses in the systems reviewed along with possible gaps in the area of research were identified. This paper also identified several recommendations and considerations for the development of a multi-precision binary multiplier system capable of treating the weaknesses of multiplier systems identified. The development of systems involving such considerations is expected to result in shorter path delays than all existing implementations of binary multiplication systems reviewed in this paper. These contributions will likely be extremely useful to arithmetic operations in digital and computer systems presently and in the future.

References

[1] Kodali, Ravi Kishore, Lakshmi Boppana, and Sai Sourabh Yenamachintala, "FPGA implementation of vedic floating-point multiplier," IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 19-21 February, pp.1-4. New York: IEEE, **(2015)** DOI: 10.1109/SPICES.2015.7091534

[2] Sharma, Richa, Manjit Kaur, and Gurmohan Singh, "Design and FPGA implementation of optimized 32-bit vedic multiplier and square architectures," International Conference on Industrial Instrumentation and Control (ICIC), 28-30 May, pp.960-964, New York: IEEE, **(2015)** DOI: 10.1109/IIC.2015.7150883.

[3] Anitha P. and P. Ramanathan, "A new hybrid multiplieusing dadda and wallace method," International Conference on Electronics and Communication Systems (ICECS), 13-14, February, pp.1-4. New York: IEEE, **(2014)** DOI: 10.1109/ECS.2014.6892623

[4] Abraham, Sumod, Sukhmeet Kaur, and Shivani Singh, "Study of various highspeed multipliers," International Conference on Computer Communication and Informatics (ICCCI), 8-10 January, pp.1-5, New York: IEEE, **(2015)** DOI: 10.1109/ICCCI.2015.7218139

[5] Vyas, Keerti, Ginni Jain, Vijendra K. Maurya, and Anu Mehra, "Analysis of an efficient partial product reduction technique," International Conference on Green Computing and Internet of Things (ICGCIoT), 8-10 October, pp.1-6. New York: IEEE, **(2015)** DOI: 10.1109/ICGCIoT.2015.7380417

[6] Chopade S. S. and Rama Mehta, "Performance analysis of vedic multiplication technique using FPGA," IEEE Bombay Section Symposium (IBSS), 10-11 September, pp.1-6. New York, **(2015)** DOI: 10.1109/IBSS.2015.7456657

[7] Bisoyi, Abhyarthana, Mitu Baral, and Manoja Kumar Senapati, "Comparison of a 32-bit vedic multiplier with a conventional binary multiplier," International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 8-10 May, 1pp.757-1760, New York: IEEE, **(2014)** DOI: 10.1109/ICACCCT.2014.7019410

[8] Mhaidat, Khaldoon M., and Abdulmughni Y. Hamzah, "A new efficient reduction scheme to implement tree multipliers on FPGAs," 9th International Design and Test Symposium, 16-18 December, pp.180-184, New York: IEEE, **(2014)** DOI: 10.1109/IDT.2014.7038609

[9] Bathija R.K., R.S. Meena, S. Sarkar, and Rajesh Sahu, "Low power high speed 16x16 bit multiplier using vedic mathematics," International Journal of Computer Applications, vol.59, no.6, pp.41-44, **(2012)**

[10] Rao, Jagadeshwar M., and Sanjay Dubey, "A high speed and area efficient booth recoded wallace tree multiplier for fast arithmetic circuits," Asia Pacific Conference on Postgraduate Research in Microelectronics &

Electronics (PRIMEASIA), 5-7 December, pp.220-223, New York: IEEE, **(2012)** DOI: 10.1109/PrimeAsia.2012.6458658

[11] Jain Anna, Baisakhy Dash, Ajit Kumar Panda, and Muchharla Suresh, “FPGA design of a fast 32-bit floating-point multiplier unit,” International Conference on Devices, Circuits and Systems (ICDCS), 15-16 March, 545-pp.547, New York: IEEE, **(2012)**

[12] Arish S. and R. K. Sharma, “Run-time reconfigurable multi-precision floating-point multiplier design for high speed, low-power applications,” 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 19-20 February, pp.902-907. New York, **(2015)** DOI: 10.1109/SPIN.2015.7095315

[13] Arish S. and R. K. Sharma, “An efficient binary multiplier design for high speed applications using karatsuba algorithm and urdhva-tiryagbhyam algorithm,” Global Conference on Communication Technologies (GCCT), 23-24 April, pp.192-196. New York: IEEE, **(2015)** DOI: 10.1109/GCCT.2015.7342650

[14] Gokhale G. R. and P. D. Bahirgonde, “Design of vedic-multiplier using area-efficient carry select adder,” International Conference on Advances in Computing, Communications and Informatics (ICACCI), 10-13 August, pp.576-581, New York: IEEE, **(2015)** DOI: 10.1109/ICACCI.2015.7275671

[15] Ram, G. Challa, Y. Rama Lakshmanna, D. Sudha Rani, and K. Bala Sindhuri, “Area efficient modified vedic multiplier,” International Conference on Circuit and Computing Technologies (ICCPCT), 18-19, March, pp.276-279, New York: IEEE, **(2016)** DOI: 10.1109/ICCPCT.2016.7530294

[16] Thapliyal, Himanshu, and M. B. Srinavas, “A novel time-area-power efficient single precision floating multiplier,” Proceedings of MAPLD 16-18 June, pp.1-3, New York: IEEE, **(2005)**.

[17] Gupta, Aman, Satyam Mandavalli, Vincent J. Mooney, Keck-Voon Ling, Arindam Basu, Henry Johan, and Budianto Tandianus, “Low power probabilistic floating-point multiplier design,” 2011 IEEE Computer Society Annual Symposium on VLSI, 4-6 July, pp.182-187. New York: IEEE, **(2011)** DOI: 10.1109/ISVLSI.2011.54.

[18] IEEE (Institute of Electrical and Electronic Engineers). 2008. 754-2008 - IEEE Standard for Floating-Point Arithmetic. Revision of ANSI/IEEE Std 754-1985, New York: IEEE, **(2008)**

[19] George, Marcus, and Geetam Singh Tomar, “Hardware design procedure: principles and practices,” 5th International Conference on Communication Systems and Network Technologies, 4-6 April, pp.834 - 838, New York: IEEE, **(2015)** DOI: 10.1109/CSNT.2015.198.

[20] Hambley Allan, “Electrical engineering principles and applications,” 2nd. ed. New Jersey: Prentice Hall **(2001)**

[21] Anane N., H. Bessalah M. Issad, and M. Anane, “Hardware implementation of variable precision multiplication on FPGA,” 4th International Conference Design & Technology of Integrated Systems in Nanoscal Era, 6-9 April, pp.77-81, New York: IEEE, **(2009)** DOI: 10.1109/DTIS.2009.4938028.

[22] Benini L. and G. D. Micheli, “Automatic synthesis of low-power gated clock finite-state machines,” IEEE Transactions on CAD, vol.15, no.6, pp.630-643, **(1996)**

[23] Cheng, Fu-Chiung, Stephen H. Unger, Michael Theobald, and Wen-Chung Cho, “Delay-insensitive carry-look ahead adders,” Proceedings of 10th International Proceedings VLSI Design, Conference, 4-7, January, pp.37-63. New York: IEEE **(1997)**

[24] Hennessey, John, and David Patterson, “Computer architecture. a quantitative approach,” 3rd ed. San Francisco: Morgan Kaufmann Publishers, **(2003)**

[25] Kattamuri, R.S.N. Kumar, and S. K. Sahoo, “Computation sharing multiplier using redundant binary arithmetic,” IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 6-9, December, pp.108-111, New York: IEEE, **(2010)** DOI: 10.1109/APCCAS.2010.5774869

[26] GS Tomar and Marcus George, “Modified binary multiplier architecture to achieve reduced latency and hardware utilization,” Springers Wireless Personal Communication, vol.98, no.4, pp.3554-3561, **(2018)**