Wireless Pers Commun (2018) 98:3549–3561
https://doi.org/10.1007/s11277-017-5028-z

Modified Binary Multiplier Architecture to Achieve Reduced Latency and Hardware Utilization

Geetam Singh Tomar[1] · **Marcus Lloyde George**[2]

Published online: 31 October 2017

Abstract Arithmetic Logic Units (ALUs) are very important components of the processor, which performs various arithmetic and logical operations such as multiplication, division, addition, subtraction, cubing, squaring, etc. Of these all operations, multiplication is most elementary and most frequently used operation in the ALUs. The operation of multiplication also forms the basis of many other complex arithmetic operations such as cubing, squaring, convolution, etc. This paper presents the modified novel multi-precision binary multiplier architecture to achieve a reduced latency/delay and area/hardware utilization along with existing implementations of binary multiplication. This system will function as second stage of the of a novel multi-precision binary multiplier system. The system was implemented using Xilinx 14.2 ISE and simulated with ISIM which was available from Xilinx 14.2 ISE. The results of simulation indicate that the latency of the proposed novel binary multiplier systems (8-bit, 16-bit and 24-bit) with significantly shorter than existing implementations.

Keywords Arithmetic Logic Unit · Arithmetic Circuits · Binary Multiplier · Complex Arithmetic · FPGAs in Arithmetic

✉ Geetam Singh Tomar
gstomar@ieee.org

Marcus Lloyde George
marcus.george99@yahoo.com

[1] THDC Institute of Hydropower Engg and Technology, Bhagirathipuram, Tehri Garhwal, Uttarakhand 249124, India

[2] Department of Electrical and Computer Engineering, University of West Indies, St. Augustine, Indies, Trinidad and Tobago

1 Introduction

Arithmetic Logic Units (ALUs) are components of computer processors, which carry out arithmetic and logic operations on operands as per computer instruction words. The systems so designed have been implemented using VHDL on FPGA [1–3]. The Arithmetic Logic Units (ALUs) are not only important components of processors but mendatory to perform various arithmetic operations such as multiplication, division, addition, subtraction, cubing, squaring, etc. alongwith other logical operation [1]. The ALU of some of the processors are divided into two units, an arithmetic unit (AU) and a logic unit (LU). Some processor systems may consist multiple AUs, to perform fixed-point operations and floating-point operations separately. According to work done in [1], the multiplication is most elementary and most frequently used in ALUs. It allows one number to be scaled by another number. The binary multiplication is quite similar to the methodology utilized in the decimal number system, involves the multiplication of the multiplicands by the multipliers, one bit at a time and in each event a partial product is generated. After the multiplicand is multiplied by each bit of the multiplier the partial products would have been generated. The final step involves the summation of all generated partial products, which gives the complete product result. If the two binary numbers to be multiplied were in fractional representation then the placement of the binary point in the final product is determined in the same way in which the product of decimal numbers in fractional form were determined. An approach to multiplier fractional numbers is by first counting the number of decimal places after the binary point for both multiplicand and multiplier, then finding the sum of these two numbers to give a total that can be denoted by number P. The next step involves multiplication of both numbers as if they were integers to find the product, then placing the binary point such that P binary numbers are behind the binary point. In essence the binary point in the product is placed before total number of places P counted from right. The multiplication by zero will produce a partial product of zero. A multiplication by 1 resulted in the bits of multiplicand shifted towards the left by one bit position. The multiplication is done using hardware of adders and result is produced by using standard methods adopted for binary number system. This paper presents the implementation of a Novel Multi-Precision Binary Multiplier Architecture, which results in reduced latency/delay, reduced area and hardware utilization when compared to existing implementations of binary multiplication.

2 Literature Review

Authors in [2] presented a study of high speed binary multipliers: Booth Multiplier, Modified Booth Multiplier, Vedic Multiplier, Wallace Multiplier and Dadda Multiplier. The Booth Multiplier considers a set of two bits from the LSB of the multiplier and uses it to perform operations to develop the partial products. The number of partial products developed is equivalent to the number of bits of the multiplier; n. Due to this the number of bits of the multiplier increases the cost and reduces its speed as more partial products are involved. In the Modified Booth multiplier $n/2$ partial products are generated if n is even and $(n + 1)/2$ if it is odd. In case of the Vedic multiplier the generation of partial products and the corresponding sums are done in parallel. The partial products are obtained as presented in [1]. The Wallace multiplier operates using two states. First the numbers are converted to binary and partial products are generated. Thereafter the matrix produced is

compressed to 6, 4, 3 and 2 rows using (2,2) or (3,2) counters. The Dadda multiplier operates in two stages. The first stage consists of forming the partial product matrix. In second stage, the matrix must be broken down into rows, which are added using carry-propagating adder. This process needs minimum reduction at each level as reported by the authors. The Modified Booth multiplier reduces the number of partial products generated when compared to other multipliers while the Dadda multiplier minimizes the number of adders used when compared to the Wallace multiplier. Therefore, a new multiplier architecture called the Booth Dadda Algorithm which combines the benefits of the Modified Booth Multiplier and Dadda Multiplier. The architecture reduces the hardware utilization because of the reduction of the number of adders, and increases its speed because of the reduction in the number of partial products formed. Authors in [3] presented an efficient method for partial product reduction for binary multiplier by providing design for the 16 nm TSMH CMOS technology and was done using the Tanner EDA 14.1 development tool. To justify the desing several partial product techniques such as Wallace and Dadda schemes were investigated. Accordingly, the Dadda multiplier performs less reductions than the Wallace multiplier and claims that the Dadda multiplier consumes less power and area than the Wallace multiplier. It introduces several compressors, like 4 to 2 compressor, which introduced a horizontal path as a result of limited propagation of the carry of the multiplier unit and produced a gate level redesign of this compressor for maximizing performance. Two operating modes were considered: active mode and sleep mode. Authors also examined 3 to 2, 4 to 2, 5 to 2 and 7 to 2 compressors and their performance. The compressors with sleep transistors consumes on average 47.35% less power than the same architecture of compressor without sleep transistors and claims that the compressors with sleep transistors have less delay then the same architecture of compressor without sleep transistors. Authors in [4] proposed an 8×8 hybrid tree multiplier system using combination of the Dadda and Wallace strategies. The system was implemented on the DSCH2 tool and simulated on MICROWIND with 0.25um technology. The system indicates that the conventional 8x8 Dadda multiplier uses a bit more addition operations and therefore has greater overhead due to wiring. The decomposition logic type Dadda multiplier has partial products, which are divided into four parts and partial product addition (PPA) reduction is performed on each part, which results in the reduction of the carry propagation delay. The proposed approach includes the assignment of the name group1, group2, group3 and group4 to the four decomposition blocks and each group is assigned either a 4x4 Dadda or 4x4 Wallace algorithm, which is to be used for compressing the partial products. The preliminary results of this design reported a 40% reduction in power (via Power Delay Product PDP), which was achieved for the subject system over existing 8x8 Dadda, Wallace, Decomposed Dadda and Partitioned-type multiplier without compressors. In [5] authors presented a highspeed multiplier system, which was based on Vedic mathematics and also does a comparison of the implemented multiplier with 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. In this work both the Urdhava and Nikhilam Sutra algorithms showed significant improvements in delay over the conventional binary multiplier at 8, 16 and 32-bit modes. In [6] authors presented a comparison of 32-bit Vedic multiplication with the conventional binary multiplier. Both the systems were implemented on Xilinx Nexys 3 Spartan 6 FPGA using Xilinx ISE 13.4. It also reported that the Vedic multiplier system implemented had a latency of 168.43 ns while the conventional multiplier implemented had latency of 19.29 ns. Authors in [7] presented implementation of a new and efficient reduction scheme for implementation of tree multipliers on FPGAs. It has

proposed a library of *m:n* counters of varying sizes in order to maximize the partial product reduction ratio of the system, hence reducing the number of reduction steps hence minimizing latency and hardware utilization of the multiplier. The 32-bit multiplier scheme was implemented using Xilinx Spartan-6 on Xilinx ISE suite. It also reported that the latency of the implemented system was 15.49 ns and the hardware utilization was 1133 LUTs. Authors in [8] presented the implementation of a low power, high speed 16-bit binary multiplier using Vedic mathematics. This paper reported that the system had a latency of 27.15 ns. Authors in [9] presented the implementation of a high speed, area efficient 16-bit Vedic Multiplier and 32-bit Booth Recoded Wallace Tree multiplier for use in implementation of arithmetic circuits. It also reported that the multiplier systems implemented had latencies of 13.45 ns and 11.57 ns respectively. In [10] authros 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 and indicated that the latency of this multiplier was 16.32 ns. In [11] authors proposed an efficient strategy for unsigned binary multiplication which was expected to improve the implementation in terms of delay and area. This has utilized a combination of Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm in implementing the required system. The system was implemented using Verilog HDL using a target Spartan-3E and Virtex-4 FPGA. 8-bit, 16-bit, 24-bit and 32-bit versions of the multiplier. The delay of each was obtained and compared with existing systems of same bit sizes. The system indicates that the proposed 8-bit, 16-bit and 24-bit versions outperform their counterparts when it came to latency while the 32-bit did not perform better than its 32-bit counterparts. In [12] authors designed a area-efficient multiplier using a modified carry select adders (CSLAs) based on crosswise and vertical Vedic multiplier algorithms. The conventional BEC-based CSLAs utilized one ripple carry adder (RCA) and one binary to excess one converter (BEC) instead of dual ripple carry adders (RCAs) in its implementation. The modified CSLA consisted of three stages – half sum generation, final sum generation and carry generation [12]. Authros claimed that the 8-bit modified CSLA has shorter latency than the conventional 8-bit Vedic multiplier. This modified CSLA was then used in implementation of proposed 8-bit Vedic Multiplier and reported that the latency of the Proposed Vedic Multiplier was 45.68 ns while the hardware utilization was 1380 gates. In [13] authors presented the design of a high speed 32-bit multiplier architecture based on Vedic mathematics. This design implemented the system by adjustment of the partial products using concatenation approach. The partial products are also added using carry-save adders instead of two adders at each stage of partial product reduction. The system was implemented on the Xilinx Spartan-3E device XC3S500e-fg320-5. The desing reported that the 8-bit, 16-bit and 32-bit Vedic multiplier implementations had latencies of 13.43 ns, 17.62 ns and 22.47 ns respectively. System presented in [15] and [16] have been on the basis of existing desing and some implementation has been done using FPGA and results produced were not validated. However, the designs were on the basis of theoretical background of the sytem design [17, 18]. In [19] authors designed a structure, which can perform additional operations in parallel with reduced delay. It has used different way of parallel addition in partial product bits where tree of carry save adder has been used. The system, which is referred as Wallace tree multiplier reported to have increased speed. The system has been reported to have improved speed and some reduction in the latency by adopting two methods as discussed in the paper. The system has also used 4:2 compressor and 5:2 having conventional full and half adders.

3 Design of Modified Novel Binary Multiplier Architecture

In this paper, the previous version of novel architecture of a binary multiplier has been modified in order to reduce latency and hardware utilization drastically. The proposed architecture can be configured to operate in 1-bit to 24-bit mode at any moment. The proposed system performs most of the operations only when a non-zero bit of the multiplier is found. If more number of non-zero bits are found, the more number of cycles will be required to complete the multiplication process. However, unlike the previous multiplier system, the system has eliminated the partial product reduction phase completely. The multiplier system has been modified to reduce latency and hardware utilization by simply modifying the concept of the operation. The system stored all partial products in a 24x48-bit RAM block. Thereafter the partial product operations would have been performed using any reduction/compression approach as required. The multiplier system has eliminated the entire partial product reduction component by accumulating the partial products as they were computed. This operational decision was expected to reduce the latency and reduced the hardware utilization when compared to the system of previous version. The optimized multiplier system consisted of six (6) inputs and two (2) outputs, hence the system interface definition of the optimized system is unchanged. The input *clk* is the periodic clock waveform to the system. When *reset* is asserted HIGH the entire multiplier is initialized and the default state is enforced. *precision_mode* is used for selection of the floatingpoint precision mode (single, double, quadruple, octuple) for future upgrade of this system to facilitate other precision modes. When *start* is asserted HIGH the multiplication operations are commenced and the system performs binary multiplication of *A* (multiplicand) with *B* (multiplier). The result of the binary multiplication will be assigned to the output *product* after which the *done* signal will be asserted HIGH to indicate the end of the process as shown in Fig. 1.

The datapath design of the modified multiplier system consisted of nine (9) blocks has been shown in Fig. 2. In this figure, the Block 2A and 2B are hold-shift registers, which are used to hold the multiplicand and multiplier respectively. In this multiplier system, the multiplicand will be shifted left by Block 2A while Block 1 is an up-counter, which will keep track of the number of bit-shifts made. At the same time Block 2B, which holds the multiplier is shifted right and the least significant bit is analyzed with the aid of a comparator (block 3) to determine if it is non-zero. If it is non-zero, then the shifted multiplicand from block 2A is stored in the hold register of block 4B. At the same time the previously accumulated sum from the 48-bit binary adder (Block 5) is stored in the hold register of block 4A. The sum of the new partial product generated by the hold-shift

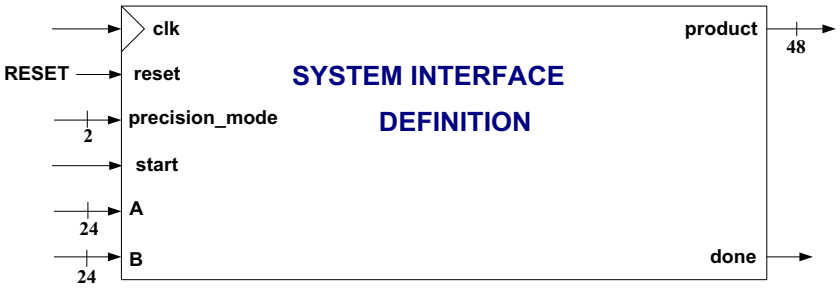


Fig. 1 System interface definition for modified novel binary multiplier

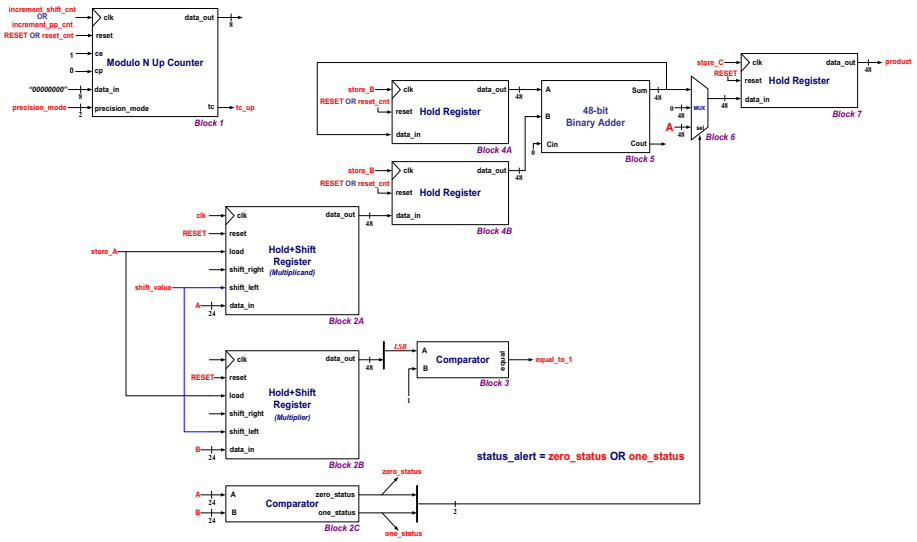


Fig. 2 Datapath design for modified novel binary multiplier

register Block 2A and stored in the hold register of Block 4B. The previously accumulated sum of Block 5which is stored in the hold register of Block 4A is computed by the 48-bit binary adder (Block 5). If on the other hand the least significant bit of the multiplier is zero then the up-counter (block 1) is still incremented and the shifting and checking operations are repeated. This process is done when all multiplier bits are examined. If either the multiplicand or multiplier were zero (0) the product must obviously be and as such it makes no sense to execute the entire operation outlined above. If the multiplier is one (1) it's obvious that the product will be equal to the value of the multiplier and as such executing the entire operation explained above will be a waste of processing time and power. As such the three possible cases for product are input to a 3-line to 1-line multiplexer (Block 6) and selection is depending on which situation arises. The selected result is then stored in a hold register (Block 7). This way the proposed design will produce product consuming less power.

4 Hardware Implementation of the of Modified Novel Binary Multiplier Architecture

The hardware implementation of the modified novel binary multiplier for single precision floating point multipolier has been done using VHDL in the Xilinx 14.2 ISE environment. The system consisted of several sub-modules and as such a structural aproach was used in the implementation of the system. Submodules were port-mapped together based on the previous version to implement the datapath. The controlpath was implemented as a finite state machine using same concept. Both datapath and controlpath were then portmapped together to produce the overall system. After implementation they were synthesized in the Xilinx 14.2 ISE environment in preparation for verification and validation stages.

5 Verification of Modified Novel Binary Multiplier Architecture

In this phase of desing the timing simulation was performed on all components of the modified multiplier system to determine if they were operating as expected. Simulation was done using ISim simulator on Xilinx ISE 14.2. The modified novel binary multiplier system was tested for a wide range of input multiplicand and multiplier values. Test cases were selected with input samples belonging to each of the following categories:

(i) Zero inputs
(ii) Non-zero inputs
(iii) Mixture of zero and non-zero values
(iv) Small values
(v) Medium-sized values
(vi) Large values
(vii) Mixed small, medium and large values

The actual outputs were compared to the expected outputs to verify that the 8-bit, 16-bit and 24-bit of the modified novel binary multiplier system correctly multiplied the inputs. Tables 1, 2, 3 indicated that the actual outputs of the 8-bit, 16-bit and 24-bit modified novel binary multiplier corresponded to their expected results. The actual outputs were compared to the expected outputs to verify that the 8-bit, 16-bit and 24-bit versions of the novel binary multiplier system correctly multiplied the inputs and have been displayed in the tables.

The ISim sinulated results are shown in Fig. 3 where it is clearly indicative of expected results in accordance to the theoretical expectations.

Table 1 Verification results for modified novel multiplier system in 8-bit mode

Test #	A		B		Product	
					Expected	Actual
1	Zero	0	Zero	0	0	0
2	Zero	0	Non-zero	9	0	0
3	Non-zero	9	Zero	0	0	0
4	Small	7	Small	7	49	49
5	Medium	56	Medium	56	3136	3136
6	Large	245	Large	245	60,025	60,025
7	Small	7	Medium	56	392	392
8	Medium	56	Small	7	392	392
9	Small	7	Large	245	1715	1715
10	Large	245	Small	7	1715	1715
11	Medium	56	Large	245	13,720	13,720
12	Large	245	Medium	56	13,720	13,720
13	Small	17	Small	7	119	119
14	Small	17	Small	1	17	17

Table 2 Verification results for modified novel multiplier system in 16-bit mode

Test #	A		B		Product	
					Expected	Actual
1	Zero	0	Zero	0	0	0
2	Zero	0	Non-zero	9	0	0
3	Non-zero	9	Zero	0	0	0
4	Small	7	Small	7	49	49
5	Medium	2077	Medium	2077	4,313,929	4,313,929
6	Large	65,500	Large	65,500	4,290,250,000	4,290,250,000
7	Small	7	Medium	2077	14,539	14,539
8	Medium	2077	Small	7	14,539	14,539
9	Small	7	Large	65,500	458,500	458,500
10	Large	65,500	Small	7	458,500	458,500
11	Medium	2077	Large	65,500	136,043,500	136,043,500
12	Large	65,500	Medium	2077	136,043,500	136,043,500
13	Small	17	Small	7	119	119
14	Small	17	Small	1	17	17

Table 3 Verification results for modified novel multiplier system in 24-bit mode

Test #	A		B		Product	
					Expected	Actual
1	Zero	0	Zero	0	0	0
2	Zero	0	Non-zero	9	0	0
3	Non-zero	9	Zero	0	0	0
4	Small	7	Small	7	49	49
5	Medium	1973	Medium	1973	3,892,729	3,892,729
6	Large	7,333,333	Large	7,333,333	53,777,772,888,889	53,777,772,888,889
7	Small	7	Medium	1973	13,811	13,811
8	Medium	1973	Small	7	13,811	13,811
9	Small	7	Large	7,333,333	51,333,331	51,333,331
10	Large	7,333,333	Small	7	51,333,331	51,333,331
11	Medium	1973	Large	8,388,607	16,550,721,611	16,550,721,611
12	Large	7,333,333	Medium	1973	14,468,666,009	14,468,666,009
13	small	17	Small	7	119	119
14	small	17	Small	1	17	17

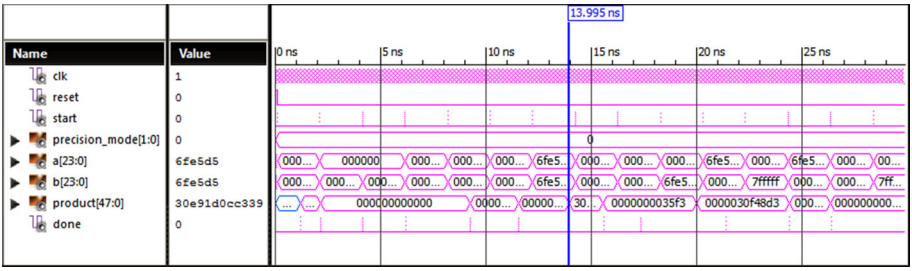


Fig. 3 ISim simulation for modified novel binary multiplier system

6 Validation of Modified Novel Binary Multiplier Architecture

The novel binary multiplier for use in a single precision floating pioint multiplier was simulated using ISim and latency was extracted from the simiulation screen. The hardware utilization and maximum frequency were determined upon synthesis of the system. 8-bit, 16-bit and 24-bit versions of the modified novel binary multiplier were constructed whereby, hardware utilization and maximum frequency were ontained. Table 4 indicates the performance parameters extracted via simulation and sytnthesis. The latency of the novel multiplier system increased as the number of non-zero bits of the multiplicand and multiplier inputs A and B incrtesed. When these inputs are zero (0) or one (1) the latency was 2 cycles.

The 8-bit, 16-bit and 24-bit versions of the modified novel binary multiplier after modification #1(Reduced Latency and Hardware Utilization) were validated by comparision of their latency with that of existing 8-bit, 16-bit and 24-bit binary multipliers. In all instances the latency of 8-bit, 16-bit and 24-bit versions of the novel binary multiplier was significantly shorter than existing implementations as indicated in Tables 5, 6, 7.

When all the bits of the multiplier inputs A and B were zero or if one of the inputs equals one (1) then the latency of the 8-bit version of the novel binary multioplier was approximately 138 times shorter than the best existing implementation of the 8-bit binary multiplier, Urdhava Vedic Multiplier from [14, 15]. When all the bits of the multiplier inputs A and B were non-zero then the latency of the 8-bit version of the novel binary multiplier was approximately five (5) times shorter than the best existing implementation of the 8-bit binary multiplier i.e. Urdhava Vedic Multiplier. When all the bits of the multiplier inputs A and B were zero or if one of the inputs equals one (1) then the latency of the 16-bit version of the novel binary multioplier was approximately 150 times shorter than the best existing implementation of the 16-bit binary multiplier. When all the bits of

Table 4 performance summary of modified novel binary multiplier architecture (removal of partial product reduction)

No. of Bits	Latency (cycles)	Hardware utilization (Sliced LUTs)	Max Frequency (MHz)
8	2–56	206	371.94
16	2–112	269	
24	2–168	320	

Table 5 Performance comparison between modified novel binary multiplier in 8-bit mode and various 8-bit binary multipliers reviewed

Source reference	Multiplier	Latency/cycles
Proposed	Novel Binary Multiplier Architecture	2–56
(Reference of version 1)	Novel Binary Multiplier Architecture	2–72
[7]	Regular Dadda Multiplier	440
	Decomposed Dadda Multiplier	410
	Partitioned Dadda Multiplier	550
	Wallace Tree Multiplier based on 3:2, 4:2 & 5:2 compressor	940
	Dadda Multiplier based on Higher Order Compressors	640
	Proposed Hybrid Multiplier Combination (Dadda/Wallace/Wallace/Dadda)	750
[8]	Conventional Multiplier	550
	Urdhava Vedic Multiplier	275
	Nikhilam Sutra Vedic Multiplier	325
[14]	Vedic Multiplier	1880
[15]	Vedic Multiplier	4568
[13]	Vedic Multiplier	672

Table 6 Performance comparison between modified novel binary multiplier architecture in 16-bit mode and various 16-bit binary multipliers reviewed

Source	Multiplier	Latency/cycles
Proposed	Novel Binary Multiplier Architecture	2–112
(Reference of version 1)	Novel Binary Multiplier Architecture	2–143
[11]	Vedic Multiplier	2715
[12]	Vedic Multiplier	1345
[8]	Conventional Multiplier	550
	Urdhava Vedic Multiplier	300
	Nikhilam Sutra Vedic Multiplier	300
[14]	Vedic Multiplier	2302
[13]	Vedic Multiplier	881

the multiplier inputs A and B were non-zero then the latency of the 16-bit version of the novel binary multiplier was approximately three (3) times shorter than the best existing implementation of the 16-bit binary multiplier. When all the bits of the multiplier inputs A and B were zero or if one of the inputs equals one (1) then the latency of the 24-bit version of the novel binary multioplier was approximately 816 times shorter than the best existing implementation of the 24-bit binary multiplier i.e. Vedic Multiplier [6–10]. When all the bits of the multiplier inputs A and B were non-zero then the latency of the 24-bit version of

Table 7 Performance comparison between modified novel binary multiplier architecture in 24-bit mode and various 24-bit binary multipliers reviewed

Source	Multiplier	Latency/cycles
Proposed	Novel Binary Multiplier Architecture	2–168
(Reference of version 1)	Novel Binary Multiplier Architecture	2–215
[13]	Vedic Multiplier	1632
[14]	Vedic Multiplier	2600

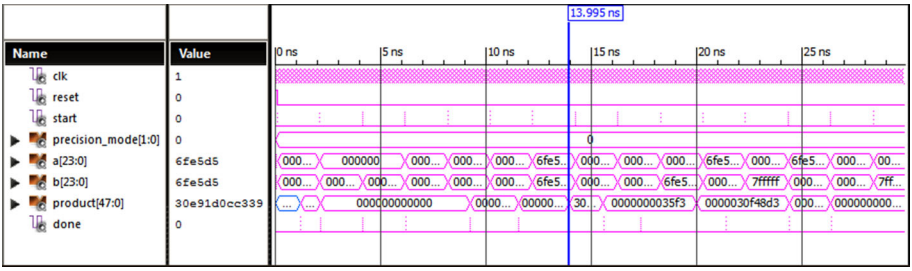


Fig. 4 ISim Simulation for modified novel binary multiplier system in 24-bit mode

the novel binary multiplier was approximately ten (10) times shorter than the best existing implementation of the 24-bit binary multiplier existed.

The results are shown in Fig. 4, where simulation outpur has been plotted using screen shot. The performance is in accordance to the predicted lines.

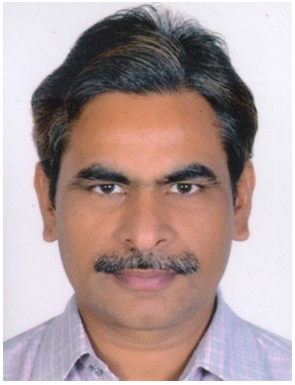
7 Conclusion

The proposed novel multi-precision binary multiplier architecture has been implemented using the knowledge of existing system, which has resulted to achieve reduction in latency and hardware utilization. In the proposed design, the RAM block has been removed and all partial products were added as they were generated instead of being stored and later accumulated. This resulted in reduction of the latency required for the multiplication process along with a reduction in hardware utilized while compared to the initial system before this modification. The modified systym had latency much shorter than all existing implemnetions compared in all modes i.e. 8-bit, 16-bit and 24-bit modes. The system desing has produced results as per the expections and will have high performance with drastic reduction in hardware.

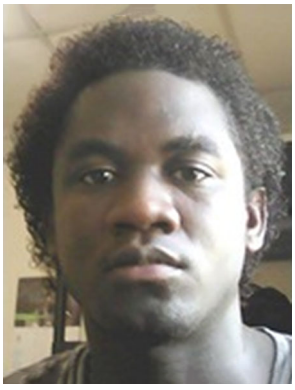
References

1. Perry, D. (1998). *VHDL* (3rd ed.). New York: McGraw-Hill.
2. Hennessey, J., & Patterson, D. (2003). *Computer architecture. A quantitative approach* (3rd ed.). San Francisco: Morgan Kaufmann Publishers.
3. Michael, D. C., Zhang, Y. Q., & Li, Q. (2005). *Advanced digital with the verilog HDL etc translating*. Beijing: Publishing House of Electronics Industry.

4. Kodali, R. K., Boppana, L., Yenamachintala, S. S. (2015). FPGA implementation of vedic floating point multiplier. In: *IEEE international conference on signal processing, informatics, communication and energy systems (SPICES)* (pp. 1–4). https://doi.org/10.1109/SPICES.2015.7091534.
5. Abraham, S., Kaur, S., & Singh, S. (2015). Study of various high speed multipliers. In *international conference on computer communication and informatics (ICCCI)* (pp. 1–5). https://doi.org/10.1109/ICCCI.2015.7218139.
6. Vyas, K., Jain, G., Maurya, V. K. & Mehra, A. (2015). Analysis of an efficient partial product reduction technique. In *International conference on green computing and internet of things (ICGCIoT)* (pp 1–6). https://doi.org/10.1109/ICGCIoT.2015.7380417.
7. Anitha, P., & Ramanathan, P. (2014). A new hybrid multiplie using Dadda and Wallace method. In *International conference on electronics and communication systems (ICECS)* (pp 1–4). https://doi.org/10.1109/ECS.2014.6892623.
8. Chopade, S. S., & Mehta, R. (2015). Performance analysis of vedic multiplication technique using FPGA. In *IEEE Bombay section symposium (IBSS)* (pp. 1–6). https://doi.org/10.1109/IBSS.2015.7456657.
9. Bisoyi, A., Baral, M., & Senapati, M. K. (2014). Comparison of a 32-bit vedic multiplier with a conventional binary multiplier. In *International conference on advanced communication control and computing technologies (ICACCCT)* (pp 1757–1760) https://doi.org/10.1109/ICACCCT.2014.7019410.
10. Mhaidat, K. M., & Hamzah, A. Y. (2014). A new efficient reduction scheme to implement tree multipliers on FPGAs. In *9th international design and test symposium* (pp 180–184) https://doi.org/10.1109/IDT.2014.7038609.
11. Bathija, R. K., Meena, R. S., Sarkar, S., & Sahu, R. (2012). Low power high speed 16 × 16 bit multiplier using vedic mathematics (0975–8887). *International Journal of Computer Applications., 59*(6), 41–44.
12. Rao, M. J., & Dubey, S. (2012). A high speed and area efficient booth recoded Wallace tree multiplier for fast arithmetic circuits. In *Asia Pacific conference on postgraduate research in microelectronics & electronics (PRIMEASIA)* (pp. 220–223).
13. Jain, A., Dash, B., Panda, A. K., & Suresh, M. (2012). FPGA design of a fast 32-bit floating point multiplier unit. In *International conference on devices, circuits and systems (ICDCS)* (pp. 545–547).
14. Arish, S., & Sharma, R. K. (2015). An efficient binary multiplier design for high speed applications using Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm. In *Global conference on communication technologies (GCCT)* (pp 192–196). https://doi.org/10.1109/GCCT.2015.7342650.
15. Gokhale, G. R., & Bahirgonde, P. D. (2015). Design of vedic-multiplier using area-efficient carry select adder. In *International conference on advances in computing, communications and informatics (ICACCI)* (pp. 576–581). https://doi.org/10.1109/ICACCI.2015.7275671.
16. Sharma, R., Kaur, M., & Singh, G. (2015). Design and FPGA implementation of optimized 32-Bit vedic multiplier and square architectures. In *International conference on industrial instrumentation and control (ICIC)* (pp. 960–964) College of Engineering Pune, India. May 28–30. https://doi.org/10.1109/IIC.2015.7150883.
17. Rao, Y. S., Kamaraju, M., & Ramanjaneyulu, D. V. S. (2015). 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). https://doi.org/10.1109/PCCCTSG.2015.7503923.
18. Marcus, G., & Tomar, G. S. (2015) Hardware design procedure: principles and practices. In *Fifth international conference on communication systems and network technologies* (pp. 834–838).
19. Sateesh, B. V., & Chacko, S. C. (2014). A high speed and area efficient wallace tree multiplier with booth recoded technique. *International Journal of Computer Science and Mobile Computing., 3*(3), 510–518.



Prof. Geetam Singh Tomar received his UG, PG, and Ph. D. degrees in electronics engineering from The Institute of Engineers, Calcutta, REC Allahabad and State Technology Univ Bhopal, respectively and Post Doctoral Fellowship from University of Kent, Canterbury U.K. He is Director of THDC Institute of Hydropower Engg & Technology, Tehri (Govt of Uttarakhand), and Prior to this worked as visiting Professor at School of Computing, University of Kent, UK, Faculty at University of west indies, Trinidad & Tobago. Apart from this, has served in IIITM Gwalior, MITS Gwalior and other Institutes of repute. Received International Pluto award for academic achievements in 2009 from IBC, Cambridge UK. He was listed in 100 top academicians of the world in 2009 and 2013. Listed in who is who in the world for 2008 and 2009 continuously. He is IEEE Senior Member, Fellow IETE and IE (I) member CSI, ACM and ISTE. He is actively involved in research and consultancy in the field of Air Interface and Advanced communication networks. He is chief editors of 5 International Journals and for 08 in past. Has published more than 190 research papers in international journals/conferences and has written 10 books from Springer, CRC, IGI and Indian publishers. Has published 04 patents. He is actively involved in IEEE activities and has organized more than 30 IEEE International conferences in India and other countries with which he is associated and delivered more than 10 Keynotes abroad and regularly visiting 02 Universities abroad as Research Consultant. Completed 05 sponsored projects for DRDO and DST including two major projects. He is member of two IEE/ISO Working group setup to finalise Standards for new protocols. *Member of Technical committee IEEE SMC and Industrial Electronics.* Associated with many boards and committees of Government and currently working for prototyping of the research outcomes of two of the patents to be commercialised in the area of Sensor networks and cognitive radio networks.



Marcus Lloyde George received the B.s.c degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine in 2007 and his MPhil degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine in 2011. Marcus is currently pursuing his PhD degree in Electrical and Computer Engineering from the University of the West Indies, St. Augustine. Since January 2008, he has been with the University of the West Indies, St. Augustine where he was also employed as the Chief Instructor of Electrical and Computer Engineering in the area of Digital Electronics Systems. His research engineering interest include the development of novel teaching methods and strategies for engineering courses, hardware specification, modelling and verification, field programmable architectures, digital integrated circuits, embedded systems design, intelligent electronic instrumentation, CADs for field programmable architectures, biomedical engineering, network on chip architectures, multimedia analysis and processing, reconfigurable computing, and information and communication technology (ICT).