Novel Binary Multiplier Architecture for Floating-Point Single Precision Multiplier

(IJCSNT) International Journal of Communication Systems and Network Technologies
ISSN Number: 2053-6283
Volume 13, Issue No. 2, 112–128

https://journals.mirlabs.in/index.php/ijcsnt
DOI-10.18486/ijcsnt/13.2.181
IJCSNT

Marcus Lloyde George[1], Abhineet Singh Tomar[2]

Abstract

In this paper a novel architecture of a binary multiplier is developed to be used in implementation of floating point multipliers. The proposed architecture can be configured to operate in at 1-bit to 23-bit mode at the moment. The system also performs most operations only when a non-zero bit of the multiplier is found and as such the more non-zero bits found, the greater the number of cycles required to complete the multiplication process. The results of the testing of this system indicates that it correctly multiplies its inputs and also the latency of the 8-bit version of this system is shorter than several other versions implemented by other researchers.

Keywords

Arithmetic Circuits, Binary Multiplier, Arithmetic Logic, FPGAs In Arithmetic

Received: 29 April 2024; **Revised:** 09 August 2024; **Accepted:** 23 August 2024;
Published: 31 August 2024;

[1]Department of Electrical and Computer Engineering, University of the West Indies, St. Augustine, Trinidad and Tobago.
[2]VOLVO, Gothenburg, Sweden.

Corresponding author:
Email: Marcus.George@sta.uwi.edu

Introduction

Fixed and floating point representations are generally used for digital signal processing applications. Floating point can represent very small and large numbers when compared to fixed point numbers, therefore the dynamic range of numbers that can be represented[1]. Many operations and processes in science utilize floating point arithmetic and therefore there is a need to develop units with shorter latency, smaller hardware utilization and less power consumption[2].

Arithmetic Logic Units (ALUs) are very important components of processors that perform various arithmetic operations such as multiplication, division, addition, subtraction, cubing, squaring, etc. Off all operations the operation of multiplication is most elementary and most frequently used in ALUs. It allows one number to be scaled by another number. The operation of multiplication also forms the basis of many other complex arithmetic operations such as cubing, squaring, convolution, etc. Floating point multiplication is the arithmetic operation most frequently utilized[3]. Floating point multiplication is basically a very important component of many engineering applications such as signal processing, video processing, image processing, etc[4]. In[5] authors have indicated that floating point multiplication is the most commonly utilized operation in many applications involving digital signal processing. Further indicates that floating point multiplication accounts for approximately 37% of the floating-point operations in benchmark applications. Multiplication consumes significant time compared to other arithmetic units used in basic mathematical computations[6]. Power management has also become extremely important especially in the case of portable electronic systems[7]. Large power dissipation results in the chip having a higher temperature profile and as such this affects the performance of the chip[8]. According to this 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.

Floating point numbers are basically numbers that cannot be represented by integers because they contain fractional components or simply because they fall outside the range of values possible within a systems bid width[9]. By representing the numbers in floating point format we preserve accuracy and resolution of the numbers when compared to fixed point format[10]. In the binary system floating point numbers are represented in both single precision and double precision formats. Three components make up these formats: sign, exponent and mantissa components. The single precision floating point format comprises a 1-bit sign (bit 31), 8-bit exponent (bits 23-30) and 23-bit mantissa (bits 0-22). The bias in this format is 127. The double precision floating point format comprises a 1-bit sign (bit 63), 11-bit exponent (bits 52-62) and 52-bit mantissa (bits 0-51). The bias in this format is 1023. The Quadruple precision floating point format comprises a 1-bit sign (bit 127), 15-bit exponent (bits 112-126) and 112-bit mantissa (bits 0-111). The bias in this format is 16383. The Octuple precision floating point format comprises a 1-bit sign (bit 254), 20-bit exponent (bits 235-254) and 235-bit mantissa (bits bits 0-234). The bias in this format is 524,287.

Floating point multiplication utilizing single, double, quadruple and octuple precisions formats first require the computation of the sign bit which is done by the XOR operation of the two sign bits. The product exponent is computed by summation of the two exponents. The value obtained is then added to the bias. The mantissa is computed by applying binary multiplication of the mantissa components of both floating point numbers[11]. In this paper we explore a new approach for binary multiplication for use in the 32-bit single precision floating point system.

Table 1. Comparison between Single, Double, Quadruple and Octuple Precision Floating Point Formats

ATTRIBUTES	IEEE-754 STANDARD FORMAT			
	SINGLE PRECISION	DOUBLE PRECISION	QUADRUPLE PRECISION	OCTUPLE PRECISION
Bits	32	64	128	256
Sign bit	1 (bit 31)	1 (bit 63)	1 (bit 127)	1 (bit 255)
Exponent	8 (bits 23-30)	11 (bits 52-62)	15 (bits 112-126)	20 (bits 235-254)
Mantissa	23 (bits 0-22)	52 (bits 0-51)	112 (bits 0-111)	235 (bits 0-234)
Bias	127	1023	16383	524,287
Range	$2^{-126} - 2^{+127}$	$2^{-1022} - 2^{+1023}$	$2^{-16382} - 2^{+16383}$	$2^{-524,286} - 2^{+524,287}$

Review of Existing Literature on Binary Multiplication

In[12] authors presented a study of four 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. As such as the number of bits of the multiplier increases the cost increases and its speed worsens 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's odd. The Vedic multiplier the generation of partial products and the corresponding sums are done in parallel. The partial products are obtained as presented in[13]. 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 with two stages. The first consist of forming the partial product matrix. Secondly the matrix must be broken down into rows which are added using carry-propagating adder. The advantage of the Dadda multiplier is that it does the minimum reduction required at each level.

According to[12] the Modified Booth multiplier reduces the number of partial predicts generated compared to others multipliers while the Dadda multiplier minimizes the number of adders used when compared to the Wallace multiplier. Therefore, proposed a new multiplier architecture called the Booth Dadda Algorithm which combines the benefits of the Modified Booth Multiplier and Dadda Multiplier. As indicated that this proposed architecture will reduce the hardware utilization because of the reduction of the number of adders used, and the architecture also increases its speed because of the reduction in the number of partial products formed. In[14] presents an efficient method for partial product reduction for binary multiplier. This system was designed for the 16nm TSMH CMOS technology and was done using the Tanner EDA 14.1 development tool. Also, in[11] authors present a study of several partial product techniques such as Wallace and Dadda schemes. Accordingly, the Dadda multiplier performs less reductions than the Wallace multiplier. also claims that the Dadda multiplier consumes less power and area than the Wallace multiplier and also introduces 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. produced a gate level redesign of this compressor for maximizing performance. Two operating modes were considered: active mode and sleep mode. examined 3 to 2, 4 to 2, 5 to 2 and 7 to 2 compressors and their performance. According to the compressors with sleep transistors consumes on average 47.35% less power than the same architecture of compressor without sleep transistors. In[11] also authors claim that

the compressors with sleep transistors have less delay then the same architecture of compressor without sleep transistors.

Table 2. Power Consumption of Compression Algorithm with and without Sleep Transistor

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

Table 3. Delay of Compression Algorithm with and without Sleep Transistor

Compressor	Without sleep transistor (ns)	With sleep transistor (ns)
3 to 2	20.44	20.43
4 to 2	24.27	22.42
5 to 2	31.059	29.838
7 to 2	25.672	22.170

In[7] authors proposed an 8x8 hybrid tree multiplier system by combination of the Dadda and Wallace strategies. The system was implemented on the DSCH2 tool and simulated on MICROWIND with 0.25um technology. indicates that the conventional 8x8 Dadda multiplier a bit more addition operations are utilized and therefore overhead due to wiring is greater. The decomposition logic type Dadda multiplier has partial products which are divided into four (4) parts and partial product addition (PPA) reduction is performed on each part and these 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 to be used for compressing the partial products. The preliminary results of the work indicate a 40% reduction in power (via Power Delay Product PDP) was achieved for the proposed system over existing 8x8 Dadda, Wallace, Decomposed Dadda and Partitioned-type multiplier without compressors.

Design Approach for 23-bit Binary Multiplier System

The multiplier system consists of six (6) inputs and two (2) outputs. 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 floating point precision mode (single, double, quadruple, octuple) for future upgrade of this system to facilitate other precision modes. In this paper the multiplier system is constructed such that it can be configured for use in any any floating point multiplication system with mantissa not exceeding 23-bits. When start is asserted

Table 4. Simulations Results for Various Types of Multipliers Reviewed[7]

Type of Multiplier	*Total Power Consumption (mw)*	*Delay (ns)*	*Power Delay Product $X10^{-12}$ (ws)*
Regular Dadda Multiplier	2.440	4.4	10.73
Decomposed Dadda multiplier	2.446	4.1	10.02
Partitioned Dadda Multiplier	2.498	5.5	13.73
Wallace tree multiplier based on 3:2,4:2&5:2 compressor	2.506	9.4	23.55
Dadda Multiplier Using Based On Higher Order Compressors	1.470	6.4	9.408
Proposed Hybrid Multiplier Combination(Dadda/Wallace/ Wallace/Dadda)	0.737	7.5	5.527

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.

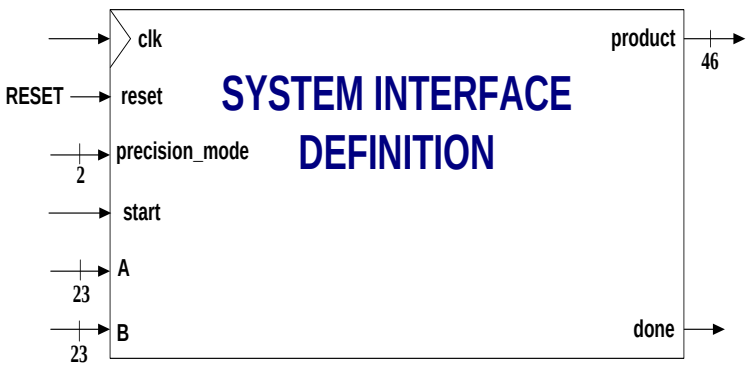

Figure 1. System Interface Definition for Novel Binary Multiplier

The datapath of the multiplication system consists of 13 blocks. Block 3A and 3B 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 3A while Block 1A is an up-counter which will keep track of the number of bit-shifts made. At the same time Block 3B which holds the multiplier is shifted right and the least significant bit is analyzed with the aid of a comparator (block 5) to determine if its non-zero. If it's non-zero then the shifted multiplicand from block 3A is stored in the RAM unit of block 6 and the both up-counters (block 1A and 1B) are incremented. If the least significant bit of the multiplier is zero then the up-counter (block 1A) is incremented, the multiplicand and multipliers are shifted again

and least significant multiplier bit checked. This process is done until all multiplier bits are examined. This operation only stores partial products resulting from non-zero multiplier bits. The up-counter (block 1A) keeps track of the number of shifts made nevertheless. When all the bits of the multiplier have been analyzed the RAM unit (block 6) would contain all partial products corresponding from the multiplication process, all of which will be non-zero. The next step will include partial product reduction.

The first step of the partial product reduction process involves storing the final count reached by the up-counter (block 1B) as the initial counting value of the down-counter. The down-counter provides the addresses for accessing partial products from the RAM unit (block 6). The RAM block (block 6) provides partial products from two address locations, the last value (location j) and the second-to-last value (location j-1) and makes it available for the 46-bit prediction adder (block 7) for summation. The sum is stored in location j-1. The process is repeated from the first step of partial product reduction until the output tc_down of the down-counter (block 2) is set HIGH, which at this time the final sum (the product of the binary multiplication) would be is stored location 0. This value will then be stored in the hold register (block 9).

It is to be noted that by manipulating the maximum count of the up-counter (block 1A) we can change the size of the multiplier to any size from 1 to 23. In this paper we will perform testing on 8-bit and 23-bit versions.

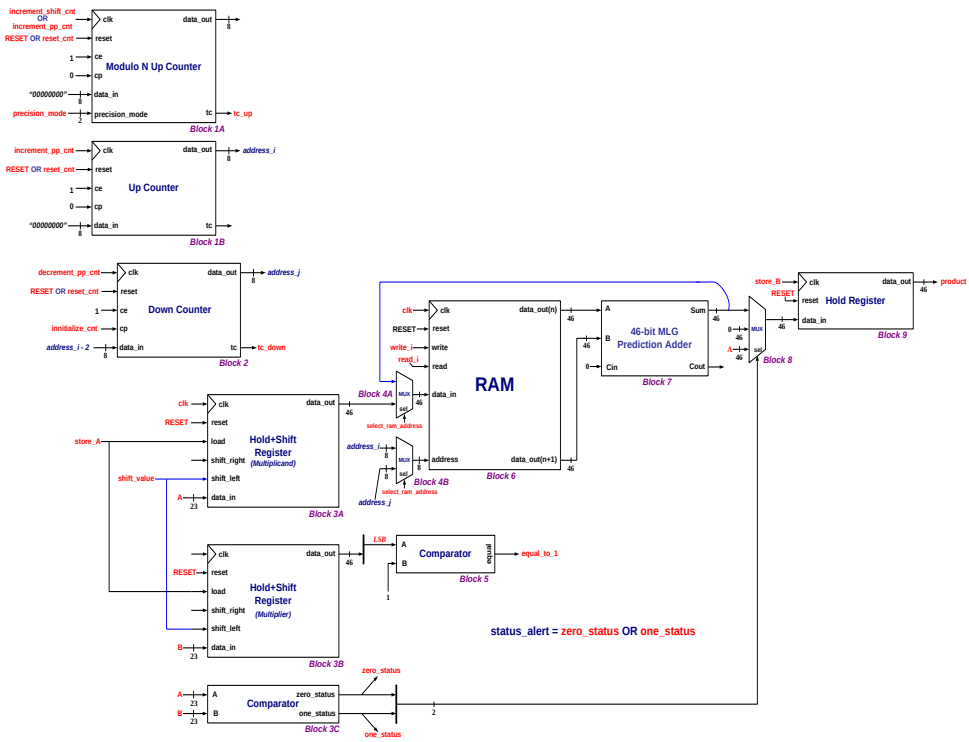

Figure 2. Datapath Design for Novel Binary Multiplier

The datapath interface definition for the system was constructed by simply extracting the inputs and outputs of the datapath design, clearly indicating which ports were connected to the control path. The control interface definition was constructed similarly. The FSM-D structure was developed by interfacing the datapath interface definition to the control interface definition.

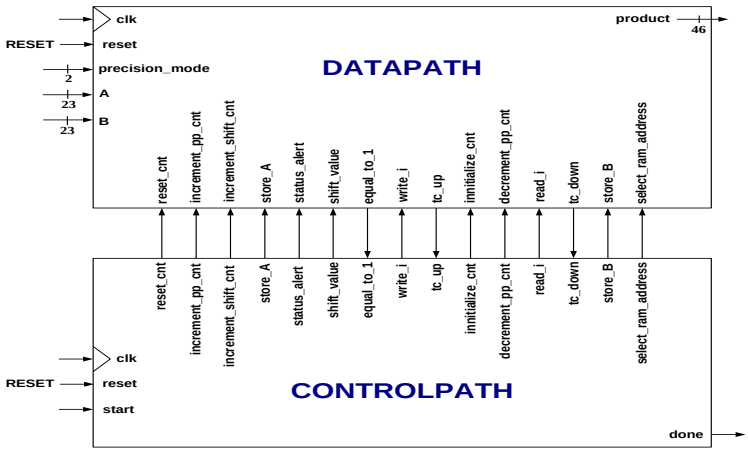

Figure 3. FSM-D Model for Novel Binary Multiplier

The operation of the datapath was then reviewed to ensure correctness and then the state transition table of the system was developed to meet the functional and timing requirements of the datapath.

The state transition table was then translated to the state diagram for the controlpath.

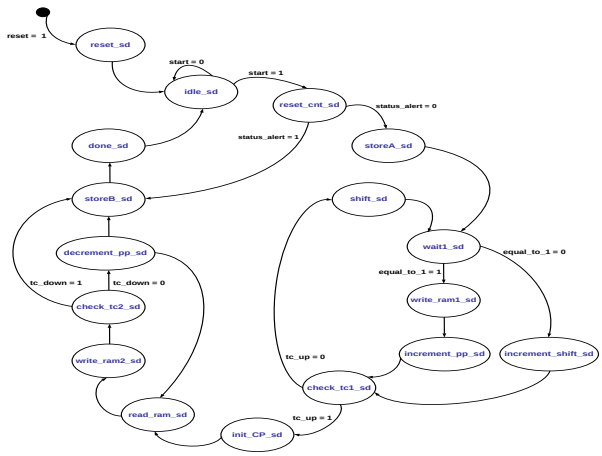

Figure 4. State Diagram for Novel Binary Multiplier's Controlpath

Table 5. State Transition Table for Controlpath

Present State	Clock (clk)	Stimuli	Next State
–	positive edge	reset = 1	reset_sd
reset_sd	positive edge		idle_sd
idle_sd	positive edge	start = 0	idle_sd
	positive edge	start = 1	reset_cnt_sd
reset_cnt_sd	positive edge	status_alert = 0	storeA_sd
reset_cnt_sd	positive edge	status_alert = 1	storeB_sd
storeA_sd	positive edge		wait1_sd
wait1_sd	positive edge	equal_to_1 = 0	increment_shift_sd
wait1_sd	positive edge	equal_to_1 = 1	write_ram1_sd
write_ram1_sd	positive edge		increment_pp_sd
increment_pp_sd	positive edge		check_tc1_sd
increment_shift_sd	positive edge		check_tc1_sd
check_tc1_sd	positive edge	tc_up = 0	shift_sd
check_tc1_sd	positive edge	tc_up = 1	init_CP_sd
shift_sd	positive edge		wait1_sd
init_CP_sd	positive edge		read_ram_sd
read_ram_sd	positive edge		write_ram2_sd
write_ram2_sd	positive edge		check_tc2_sd
check_tc2_sd	positive edge	tc_down = 0	decrement_pp_sd
check_tc2_sd	positive edge	tc_down = 1	storeB_sd
decrement_pp_sd	positive edge		read_ram_sd
storeB_sd	positive edge		done_sd
done_sd	positive edge		idle_sd

Implementation for Novel Binary Multiplier System

The novel binary multiplier system was implemented hardware implementation in VHDL using the Xilinx 13.2 ISE environment. The hardware units were port-mapped together using knowledge gained in literature review to implement the Novel Binary Multiplier System. A bottom-up approach was used in the implementation of the system. The datapath was first implemented after which the controlpath was implemented and tested. The two sub-units were then interfaced using port-mapping to obtain the complete system.

Results and Testing for Novel Binary Multiplier System

Functional and timing simulation was performed on all components of the multiplier system to determine if they were operating as expected. Simulation was done using ISim simulator on Xilinx ISE 13.2. The summary of the functional and timing simulation exercises are given in Table 6. All units operated as expected. ISim simulation of the various units were given in Figure 5 – 17.

Table 6. Summary of Simulation of Units of Novel Multiplier System

BLOCK	DESCRIPTION	SIMULATION PASSED?
1A	PMode Up-Counter	YES
1B	Up-Counter	YES
2	Down-Counter	YES
3A	Hold+Shift Register	YES
3B	Hold+Shift Register	YES
3C	Comparator Status-Check	YES
4A	46-bit 2-line to 1-line MUX	YES
4B	8-bit 2-line to 1-line MUX	YES
5	Comparator 1-bit	YES
6	RAM Block	YES
7	46-bit MLG Prediction Adder	YES
8	46-bit 3-line to 1-line MUX	YES
9	Hold Register	YES

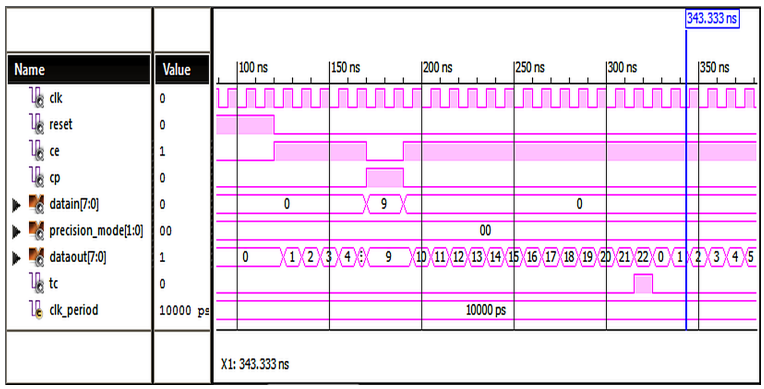

Figure 5. Timing Simulation for the Up-Counter (Block 1A)

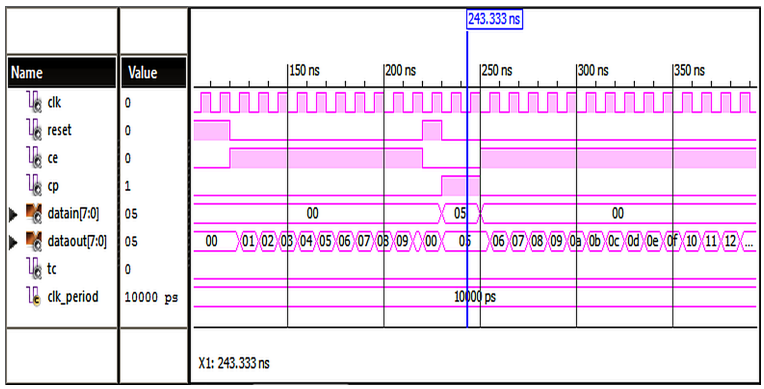

Figure 6. Timing Simulation for the Up-Counter (Block 1B)

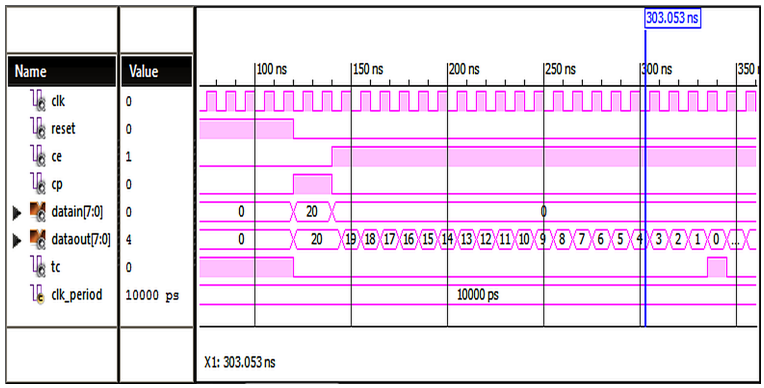

Figure 7. Timing Simulation for the Down-Counter (Block 2)

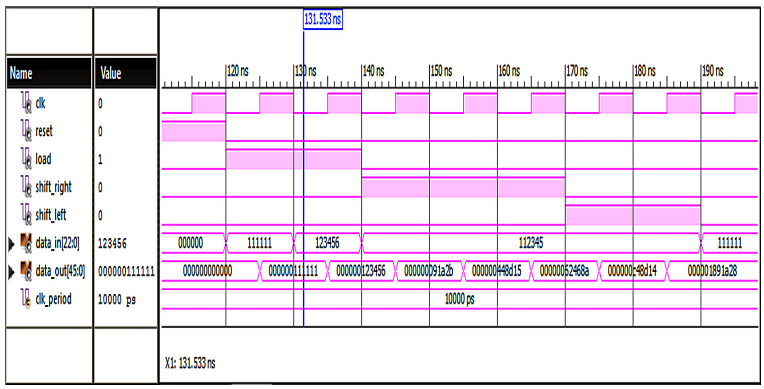

Figure 8. Timing Simulation for the Hold+Shift Register (Block 3A & 3B)

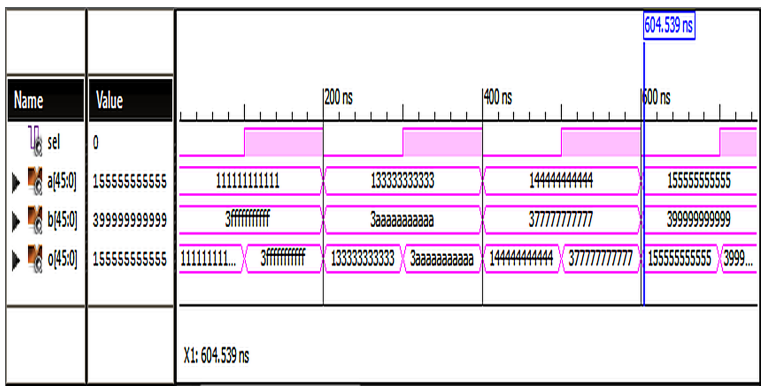

Figure 9. Timing Simulation for the 46-bit 2 line-to-1 line Multiplexer (Block 4A)

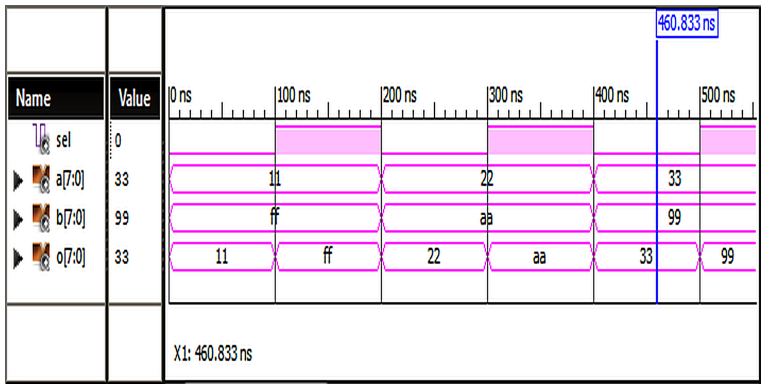

Figure 10. Timing Simulation for 8-bit 2 line-to-1 line Multiplexer (Block 4B)

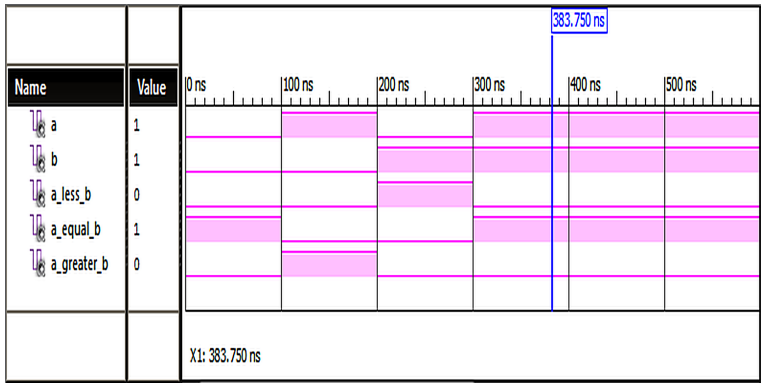

Figure 11. Timing Simulation for the 1-bit Comparator (Block 5)

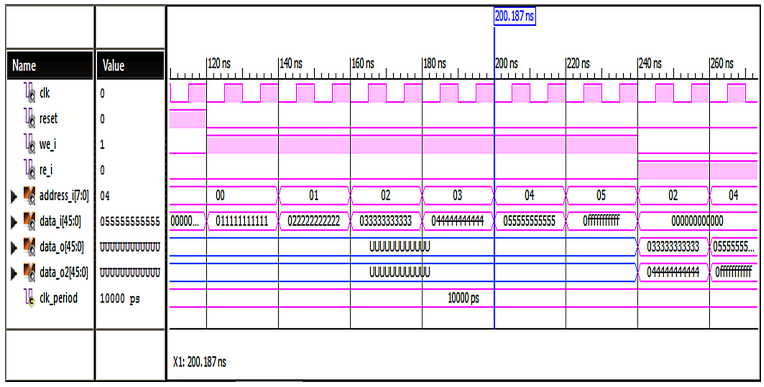

Figure 12. Timing Simulation for the 25x46-bit RAM Block (Block 6)

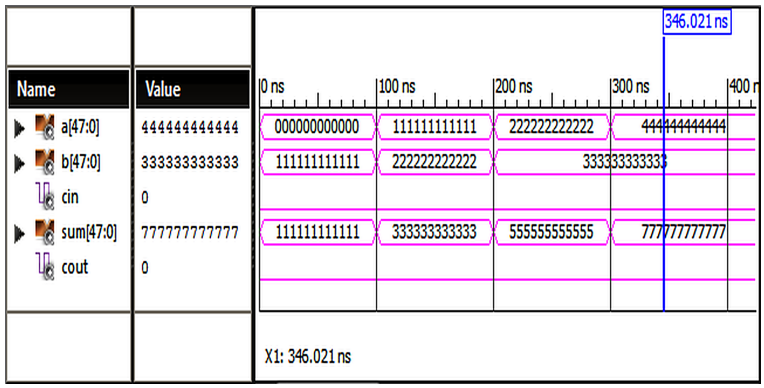

Figure 13. Timing Simulation for the 46-bit MLG Prediction Adder (Block 7)

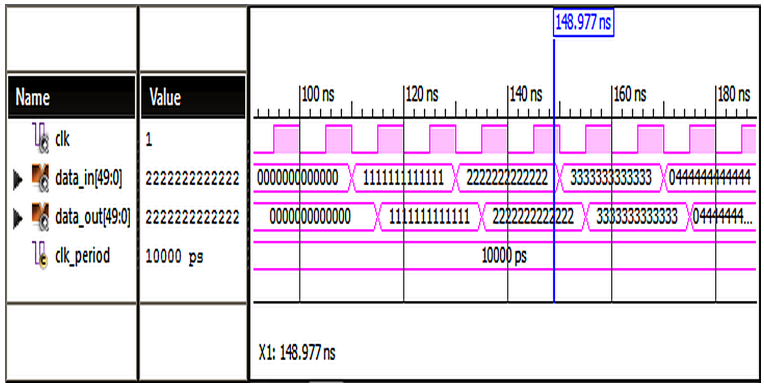

Figure 14. Timing Simulation for the Hold Register (Block 9)

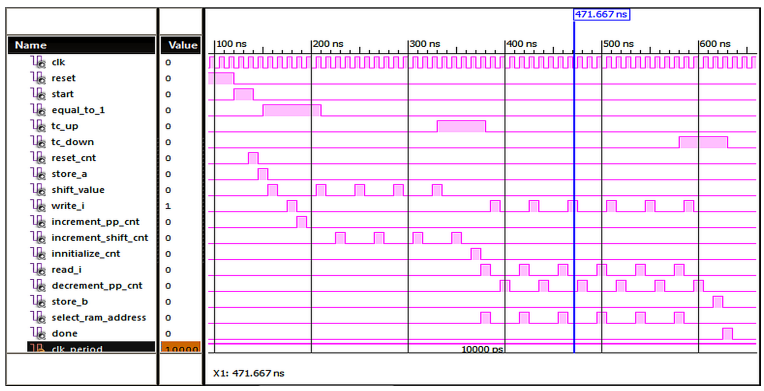

Figure 15. Timing Simulation for the CONTROLPATH of the System

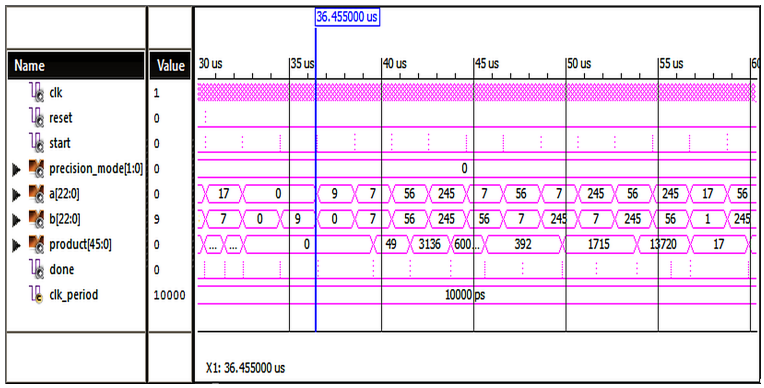

Figure 16. Timing Simulation (Compact Version) for the NOVEL MULTIPLIER SYSTEM in 8-bit MODE

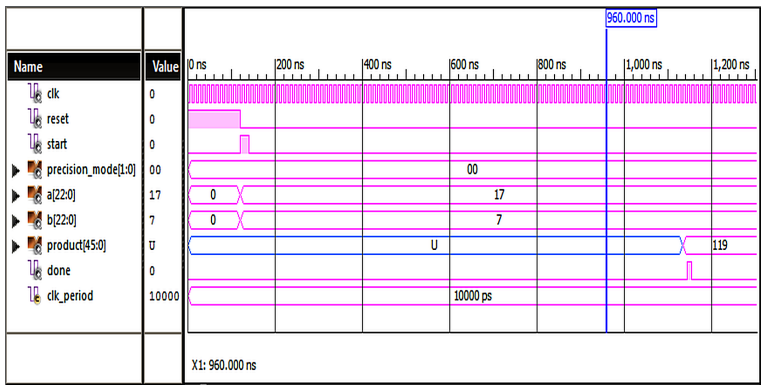

Figure 17. Timing Simulation (Expanded Version) for the NOVEL MULTIPLIER SYSTEM in 23-bit MODE

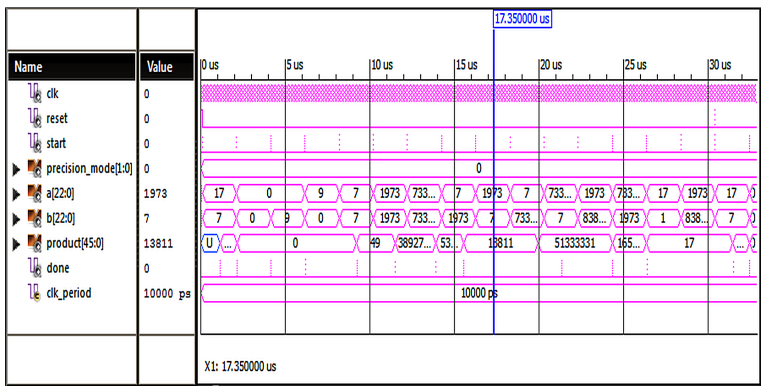

Figure 18. Timing Simulation (Compact Version) for the NOVEL MULTIPLIER SYSTEM in 23-bit MODE

The 23-bit 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:

- zero inputs
- non-zero inputs
- mixture of zero and non-zero values
- small values
- medium-sized values
- large values
- mixed small, medium and large values

The actual outputs were compared to the expected outputs to verify that the 8-bit and 23-bit versions of the novel binary multiplier system correctly multiplied the inputs. Table 7 indicated that the actual outputs of the 8-bit version of the novel binary multiplier corresponded to their expected results, while table 8 indicated that the actual outputs of the 23-bit version of the novel binary multiplier corresponded to their expected results.

Table 7. Verification Results for 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	60025	60025
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	13720	13720
12	large	245	medium	56	13720	13720
13	small	17	small	7	119	119
14	small	17	small	1	17	17

A performance analysis of the 8-bit and 23-bit versions of the novel binary multiplier were done to determine the latency, minimum clock period, maximum frequency and logic distribution of the implemented system. Table 8 gives performance metrics for the implemented system.

The latency of the system in 8-bit mode ranges 3 – 68 cycles while the system in 23-bit mode ranged 3 - 198 cycles. A range is involved simply because the system does most operations only when a non-zero bit of the multiplier is found and as such the more non-zero bits found, the greater the number of cycles required to complete the multiplication process.

The system developed in this research was compared to that of several multipliers reviewed. The latencies in[12] were given in nanoseconds and there was no mention of the clock resolution used in the simulation, as such in this research it was decided that latencies of our novel binary multiplier be converted from cycles to nanoseconds considering a clock resolution of 100ps. The results of the

Table 8. Verification Results for Novel Multiplier System in 23-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	3892729	3892729
6	large	7333333	large	7333333	53777772888889	53777772888889
7	small	7	medium	1973	13811	13811
8	medium	1973	small	7	13811	13811
9	small	7	large	7333333	51333331	51333331
10	large	7333333	small	7	51333331	51333331
11	medium	1973	large	8388607	16550721611	16550721611
12	large	7333333	medium	1973	14468666009	14468666009
13	small	17	small	7	119	119
14	small	17	small	1	17	17

Table 9. Performance Analysis of the 8-bit and 23-bit Multiplier System

PARAMETER	VALUE	
	8-bit Version	23-bit Version
Latency (cycles)	3 (min) – 68 (max)	3 (min) – 198 (max)
Minimum Clock Period (ns)	2.23	5.30
Maximum Frequency (MHz)	276.99	188.60
Logic Distribution:		
no. of occupied Slices:	1,558/7,680 (20%)	1,558/7,680 (20%)
no. of Slices containing only related logic:	1,558/1,558 (100%)	1,558/1,558 (100%)
no. of Slices containing unrelated logic:	0/1,558 (0%)	0/1,558 (0%)
no. of 4 input LUTs:	1,983/15,360 (12%)	1,983/15,360 (12%)

comparison of Table 10 indicate that the novel binary multiplier in 8-bit and 23-bit modes out-performs all other multipliers reviewed as the number of non-zero bits of the multiplier decreases. The novel binary multiplier in 8-bit mode with a multiplier X"FF" out-performs the Wallace Tree Multiplier, Dadda Multiplier Using Higher Order Compressors and Hybrid Multiplier.

Table 10. Performance Comparison of the Novel Binary Multiplier with Various Multipliers Reviewed in (Anitha and Ramanathan 2014)

Type of Multiplier	Bitwidth	Latency (ns)
Regular Dadda Multiplier	8x8	4.4
Decomposed Dadda Multiplier	8x8	4.1
Partitioned Dada Multiplier	8x8	5.5
Wallace Tree Multiplier	8x8	9.4
Dadda Multioplier Using Higher Order Compressors	8x8	6.4
Hybrid Multiplier Proposed in (Anitha and Ramanathan 2014)	8x8	7.5
Novel Binary Multiplier	8x8	0.3 - 6.8
Novel Binary Multiplier	23x23	0.3 - 19.8

Conclusion

The implemented system correctly computed the product for a variety of multiplicand and multiplier inputs. The novel binary multiplier in 8-bit and 23-bit modes out-performs all other multipliers reviewed as the number of non-zero bits of the multiplier decreases. This unit currently caters for the needs of a single-precision floating point multiplier, however future work should be done in upgrading this system to cater for the needs of double, quadruple and octuple precision floating point multiplication. Some work can also be done in rounding of the product for better use in the implementation of digital systems.

References

[1] Abraham, Sumod, Sukhmeet Kaur and Shivani Singh. 2015. Study of Various High Speed Multipliers. 2015 International Conference on Computer Communication and Informatics (ICCCI). pp 1 - 5, DOI: 10.1109/ICCCI.2015.7218139

[2] Bisoyi, Abhyarthana,Mitu Baral, Manoja Kumar Senapati. 2014. Comparison of a 32-bit Vedic Multiplier with a Conventional Binary Multiplier. 2014 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). pp 1757 - 1760, DOI: 10.1109/ICACCCT.2014.7019410

[3] Cui, Xiaoping, Weiqiang Liu, Xin Chen,Earl Swartzlander,Fabrizio Lombardi. 2015.

[4] X. Cui, W. Liu, X. Chen, E. E. Swartzlander and F. Lombardi, "A Modified Partial Product Generator for Redundant Binary Multipliers," in IEEE Transactions on Computers, vol. 65, no. 4, pp. 1165-1171, 1 April 2016, doi: 10.1109/TC.2015.2441711.

[5] George, Marcus and Geetam Singh Tomar. 2015. Hardware Design Procedure: Principles and Practices. 2015 Fifth International Conference on Communication Systems and Network Technologies.

[6] Kodali, R.K.; Boppana, L.; Yenamachintala, S.S. 2015. FPGA implementation of vedic floating point multiplier. 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES). Pages: 1 - 4, DOI: 10.1109/SPICES.2015.7091534

[7] Momeni, Amir, Jie Han, Paolo Montuschi, Fabrizio Lombardi. 2015. Design and Analysis of Approximate Compressors for Multiplication. IEEE Transactions on Computers. Volume 64 (4), pp 984 - 994, DOI: 10.1109/TC.2014.2308214

[8] P. Anitha; P. Ramanathan. 2014. A new hybrid Multiplieusing Dadda and Wallace Method. 2014 International Conference on Electronics and Communication Systems (ICECS). pp 1 - 4, DOI: 10.1109/ECS.2014.6892623

[9] Perry, D. 1998. VHDL. 3rd ed. New York: McGraw-Hill.

[10] S. Arish and R. K. Sharma. 2015. An Efficient Binary Multiplier Design for High Speed Applications using Karatsuba Algorithm and Urdhva-Tiryagbhyam Algorithm. 2015 Global Conference on Communication Technologies (GCCT), pp 192 - 196, DOI: 10.1109/GCCT.2015.7342650

[11] Vyas, Keerti,Ginni Jain, Vijendra K. Maurya and Anu Mehra. 2015. Analysis of an Efficient Partial Product Reduction Technique. 2015 International Conference on Green Computing and Internet of Things (ICGCIoT). pp 1 - 6, DOI: 10.1109/ICGCIoT.2015.7380417

[12] V. T. Pardhi and P. Palsodkar, "Partial product generator for signed binary multipliers," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2017, pp. 262-266, doi: 10.1109/ICECA.2017.8203683.

[13] GS Tomar, Marcus L George, "Modified Binary Multiplier Architecture to Achieve Reduced Latency and Hardware Utilization", Wireless Personal Communication, Vol.98, No.4, pp.3549-3561, 2018.

[14] GS Tomar, Marcus L George, AS Tomar, "Multi-Precision Binary Multiplier Architecture for use in Multi-Precision Floating Point Multiplication", IET Circuits Devices and systems, Vol.15, No.5, pp.455-464, 11 Mar 2021. DOI: 10.1049/cds2.12041