Design and Analysis of 64-Bit Approximate Multiplier for Accurate and High-Level Processing

Pradeep Baghel[1], Ms. Farha Khan[2]

[1]PG Scholar, [2]Assistant Professor

[1,2] Department of ECE, Mittal Institute of Technology, Bhopal, INDIA

Abstract—***This dissertation proposed a design of 64-bit approximate multiplier algorithm based on compressors and dadda multipliers for high accuracy & speed. Approximate multiplier are one of the fastest multiplier for the AI based FPGA-VLSI applications. The proposed research is presents the 64 X 64 bit approximate multiplier. The virtex 7 family FPGA IC is used to simulate the results. The proposed approximate multiplier designed for the 64 X 64 bit multiplication while previous it is designed for the 16 X 16 bit multiplication. The total number of component or utilized area is 10.66 mm2 while previously it is 18.37 mm2. The total delay value is 0.344 ns (16X16) and 1.376 ns (64X64) for proposed work while 0.624 ns are for previous work. The throughput achieved by this research is 46.51 Gbps while 25.64 Gbps for previous simulation results.***

I. INTRODUCTION

Different computer arithmetic systems can be utilized to execute an advanced multiplier. Out of these most procedures include computing a lot of halfway products, and afterward adding the incomplete products together. Until the 1970s, most minicomputers didn't have increase guidance be that as it may, Centralized server PCs had duplicate guidelines, yet they did the a few sorts of movements and includes as an 'increase schedule'. Early chip additionally had no duplicate guidance. At that point the Motorola 6809, presented in 1978, was probably the most punctual microchip with devoted equipment increase guidance. It did likewise sorts of movements and includes as a 'duplicate daily practice', yet the thing that matters is that execution was done in the microcode of the MUL guidance. As more transistors per chip got accessible because of bigger scale coordination, it got conceivable to put satisfactory adders on a single chip to total all the fractional products without a moment's delay, as opposed to reuse a single adder to deal with every halfway product each in turn. Presently, on the grounds that some normal advanced sign preparing algorithms invest the vast majority of their energy duplicating, computerized signal processor fashioners penance a great deal of chip area so as to make the increase as quick as would be prudent; a single cycle duplicate gather unit often spent the majority of the chip area of early DSPs. The development of most computerized frameworks is a huge assignment. Restrained creators in any field will subdivide the first undertaking into sensible subunits building squares and will utilize the standard subunits at every possible opportunity. In computerized equipment, the structure squares have such names as adders, registers, and multiplexers.

Different arithmetic activities, for example, duplication expansion, subtraction are significant bit of advanced circuit to accelerate the calculation speed of processor. Anyway the speed of processor enormously relies upon multiplier unit of the processor. This thusly expands the interest of rapid multiplier design in ALU and in different advanced sign processors. A few new multiplier engineering has been presented in the course of recent decades. Booth's multiplier [7] and modified booth's [12] multiplier are well known in current VLSI structure however they have their own arrangement of detriments. In this multiplier before showing up the last answer a few middle of the road steps are required that eases back the speed of processor. This middle of the road steps incorporates a few moving tasks, examination and subtraction which decrease the speed of processor exponentially as the quantity of bits present in multiplier and multiplicand increments. Since Speed is significant worry in creating processors now days, so new engineering must be presented which are quicker than previously mentioned multiplier. To address the previously mentioned hindrance of traditional multiplier booth's multiplier and modified booth's multiplier another engineering dependent on approximate multiplier is investigated.

1.2 Approximate Multiplier

Approximate computing is a calculation procedure which restores a potentially wrong outcome as opposed to an ensured precise outcome, and can be utilized for applications where an approximate outcome is adequate for its purpose.[1][2] One case of such circumstance is for a web crawler where no careful answer may exist for a specific inquiry question and subsequently, numerous answers might be satisfactory. Also, infrequent dropping of certain edges in a video application can go undetected because of perceptual confinements of people. Approximate computing depends on the perception that in numerous situations, despite the fact that performing precise calculation requires huge measure of assets,

permitting limited estimate can give lopsided gains in execution and vitality, while as yet accomplishing adequate outcome accuracy. For instance, in kmeans bunching algorithm, permitting just 5% misfortune in arrangement exactness can give multiple times vitality sparing contrasted with the completely precise classification [1].

1.3 Need of Approximate Multiplier

In two plans of approximate 4-2 blowers are displayed and utilized in halfway product decrease tree of four variations of 8 × 8 Dadda multiplier. The significant downside of the proposed blowers is that they give nonzero yield for zero esteemed data sources, which to a great extent influences the mean relative error (MRE) as examined later. The approximate plan proposed in these brief defeats the current disadvantage. This prompts better exactness. In static section multiplier (SSM) proposed, m-bit fragments are determined from-bit operands dependent on driving 1 bit of the operands. At that point, m× m augmentation is performed rather than n × n duplication, where m<n. Halfway product puncturing (PPP) multiplier excludes k progressive fractional products beginning from jth position, where j ∈ [0,n-1] and k ∈ [1, min(n-j, n-1)] of a n-bit multiplier. In 2 × 2 approximate multiplier dependent on altering a passage in the Karnaugh map is proposed and utilized as a structure square to build 4 × 4 and8 × 8 multipliers.

Duplication of paired numbers can be disintegrated into increments. Consider the increase of two 8-bit numbers An and B to create the 16 bit item P.

```
                                A7  A6  A5  A4  A3  A2  A1  A0
                              X B7  B6  B5  B4  B3  B2  B1  B0
                    ------------------------------------------------
                                A7.B0 A6.B0 A5.B0 A4.B0 A3.B0 A2.B0 A1.B0 A0.B0
                          +  A7.B1 A6.B1 A5.B1 A4.B1 A3.B1 A2.B1 A1.B1 A0.B1
                    +  A7.B2 A6.B2 A5.B2 A4.B2 A3.B2 A2.B2 A1.B2 A0.B2
              +  A7.B3 A6.B3 A5.B3 A4.B3 A3.B3 A2.B3 A1.B3 A0.B3
        +  A7.B4 A6.B4 A5.B4 A4.B4 A3.B4 A2.B4 A1.B4 A0.B4
   +  A7.B5 A6.B5 A5.B5 A4.B5 A3.B5 A2.B5 A1.B5 A0.B5
 +  A7.B6 A6.B6 A5.B6 A4.B6 A3.B6 A2.B6 A1.B6 A0.B6
+A7.B7 A6.B7 A5.B7 A4.B7 A3.B7 A2.B7 A1.B7 A0.B7
-----------------------------------------------------------------------------
P15  P14  P13  P12  P11  P10  P9  P8  P7  P6  P5  P4  P3  P2  P1  P0
```

Partial Products to be added

The equation for the addition is:

$$P(m+n) = A(m)B(n) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_i b_j 2^{i+j}$$

1.4 DSP Architecture

The structure of approximate multiplier is actualized into a custom DSP. The engineering displayed in figure 1.1, it is a specially crafted DSP structure with least control rationale and center significance is given to arithmetic unit planned with an accumulator, barrel shifter, approximate multiplier and an area effective convey select adder.

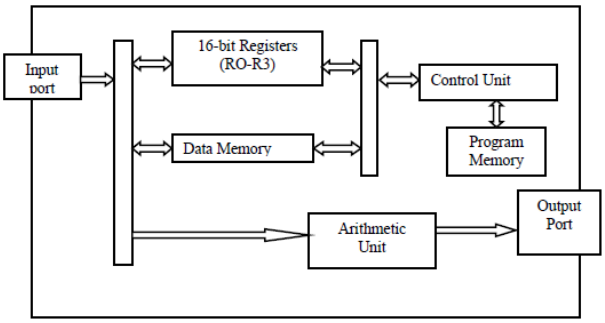

Figure 1.1: DSP Architecture

II. LITERATURE SURVEY

Digital multiplier play vital role in advance processing so there are lot of multiplier are continue enhancing and developing. This chapter include summary of various research

work based on multiplier around the world. Some of the multiplier research is following discussed.

P. J. Edavoor et al.,[1] High speed multimedia applications have paved way for a whole new area in high speed error-tolerant circuits with approximate computing. These applications deliver high performance at the cost of reduction in accuracy. Furthermore, such implementations reduce the complexity of the system architecture, delay and power consumption. This work explores and proposes the design and analysis of two approximate compressors with reduced area, delay and power with comparable accuracy when compared with the existing architectures. The proposed designs are implemented using 45 nm CMOS technology and efficiency of the proposed designs have been extensively verified and projected on scales of area, delay, power, Power Delay Product (PDP), Error Rate (ER), Error Distance (ED), and Accurate Output Count (AOC). The proposed approximate 4:2 compressor shows 56.80% reduction in area, 57.20% reduction in power, and 73.30% reduction in delay compared to an accurate 4:2 compressor. The proposed compressors are utilized to implement 8 × 8 and 16 × 16 Dadda multipliers. These multipliers have comparable accuracy when compared with state-of-the-art approximate multipliers. The analysis is further extended to project the application of the proposed design in error resilient applications like image smoothing and multiplication.

V. A et al.,[2] Approximate Arithmetic has a great potential to design digital systems that consume less power and area without compromising delay. This technique is mainly utilized to design systems used in error tolerant Digital Signal Processing (DSP) applications as it simplifies the conventional circuit using certain approximation design strategy sacrificing the accuracy of the output. In this work, a power and area efficient multipliers are proposed using approximate half adders,

full adders and OR gates for partial product accumulation. OR gates are used in the Lower Significant Bit positions (LSB) and the approximate adders are used in Most Significant Bit (MSB) positions. This reduces the carry propagation for LSB bit positions. Initially, approximate half and full adders are proposed and using that an approximate unsigned and signed 4×4 multipliers design is shown. Accurate Wallace Tree multipliers (WTM) are also designed in order to determine the performance of the proposed approximate multipliers.

R. Bhattacharjya et al.,[3] This work aims to present a reconfigurable rounding based multiplier with different accuracy levels. It is based on a divide and conquer approach for applications in image processing. Our proposed approach divides the multiplicand and multiplier into two halves. Each half is multiplied with the other and our architecture is made accuracy-configurable. Further, due to rounding based approach, our approximate multiplication technique generates results faster. Based on our experiments, we observe that our proposed multiplier is 26.3% more accurate on an average compared to other state-of-the-art approximate multipliers for 8-bit operations.

H. Waris et al.,[4] Radix-4 Booth encoding provides ease in the generation of partial products, thus is widely used to achieve power-efficient and low-area signed multipliers. Conversely, the radix-8 Booth encoding exhibits low-performance as it requires generation of odd multiples of the multiplicand. In this brief, this issue is addressed by approximating the odd multiples of radix-8 to their nearest power of two such that the errors complement each other. In the pursuit of an accuracy-energy trade-off, hybrid low radix (HLR) based two approximate Booth multipliers (HLR-BM1 and HLR-BM2) are designed. HLR-BM2, compared to the previous best error-optimized design (ABM1), achieved a reduced energy of 22% with a comparable MRED.

S. Venkatachalam et al.,[5] Approximate computing is an emerging technique in which power-efficient circuits are designed with reduced complexity in exchange for some loss

in accuracy. Such circuits are suitable for applications in which high accuracy is not a strict requirement. Radix-4 modified Booth encoding is a popular multiplication algorithm which reduces the size of the partial product array by half. In this work, three Approximate Booth Multiplier Models (ABM-M1, ABM-M2, and ABM-M3) are proposed in which approximate computing is applied to the radix-4 modified Booth algorithm. Each of the three designs features a unique approximation technique that involves both reducing the logic complexity of the Booth partial product generator and modifying the method of partial product accumulation.

S. T. Bala et al.,[6] Over the years, the complexity of VLSI design circuits has increased dramatically. With this improvement, there comes the need for low area and low power VLSI circuits. The common known fact is that multiplier circuit plays an important role in the digital processor design. Nowadays, low power and low area multiplier designs are in high demand. Compared to other multipliers, Wallace tree multipliers are considered to be fast rather than other multipliers.

V. V. Kavipranesh et al.,[7] Approximate results are required in many embedded data processors as they reduce time delay and power. As error tolerance adder (ETA) has decreased power drastically trading with accuracy. This work focuses on reducing delay on existing adders when replaced with a fast adder. When compared to the past works on ETA, the proposed work has high power utilization and more accuracy of speed. The proposed design is compared and synthesized for the power and delay. When observed the existing ETA designs, the proposed work achieves significant improvement in power dissipation about 17.13%, 4.6%, 15.4%, 5.35% decrement for 4, 8, 16, 32 bits respectively, and significant improvement in delay about 28.90%, 23.59%, 20.08%, 24.44% decrement for 4, 8, 16, 32 bits respectively.

P. Huang et al.,[8] As one of the most promising energy-efficient emerging paradigms for designing digital systems, approximate computing has attracted a significant attention in recent years. Applications utilizing approximate computing can tolerate some loss of quality in the computed results for attaining high performance. Approximate arithmetic

circuits have been extensively studied; however, their application at system level has not been extensively pursued. Furthermore, when approximate arithmetic circuits are applied

at system level, error-accumulation effects and a convergence problem may occur in computation. Semi-supervised learning can improve accuracy and performance by using unlabeled examples.

III. PROBLEM FORMULATION

Digital signal processing (DSP) is the use of digital processing, such as by computers or more specialized digital signal processors, to perform a wide variety of signal processing operations. The digital signals processed in this manner are a sequence of numbers that represent samples of a continuous variable in a domain such as time, space, or frequency. In digital electronics, a digital signal is represented as a pulse train,[1][2] which is typically generated by the switching of a transistor.[3]

Digital signal processing and analog signal processing are subfields of signal processing. DSP applications include audio and speech processing, sonar, radar and other sensor array processing, spectral density estimation, statistical signal processing, digital image processing, data compression, video coding, audio coding, image compression, signal processing for telecommunications, control systems, biomedical engineering, and seismology, among others.

DSP can involve linear or nonlinear operations. Nonlinear signal processing is closely related to nonlinear system identification [4] and can be implemented in the time, frequency, and spatio-temporal domains.

For advance digital processing, need high speed multiplier. Therefore need of high speed and more accurate multiplier.

IV. PROPOSED METHODOLOGY

Proposed 64-bit approximate multiplier is design according to following flow chart. The details of the entire block are discussed also in this chapter.

4.1 PROPOSED WORK

- To implement 64 X 64 bit approximate multiplier.
- To use verilog coding on Xilinx platform for implementation.
- To calculate various parameters values like area, power, delay and power delay product (PDP).
- To compare proposed implementation form existing approximate multiplier.

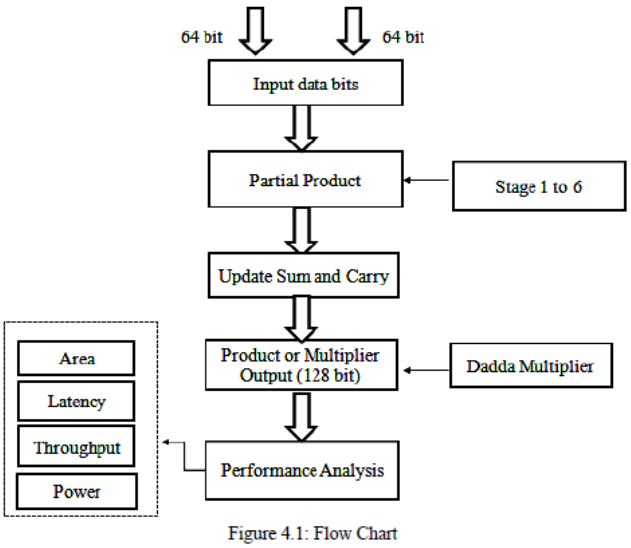

Figure 4.1: Flow Chart

Figure 4.1 is showing proposed flow chart. According to this working flow it can be clear Bthat proposed approximate multiplier is design and implemented according to following

sub modules-

- Carry Save Adder
- Dadda Multiplier
- Compressor
- Full Adder
- Half Adder

4.1.1 Carry Save Addition

A Carry-Save Adder is only a lot of one-bit full adders, with no carry-tying. Along these lines, a n-bit CSA gets three n-bit operands, to be specific A (n-1).A (0), B (n-1).B (0), and CIN (n-1).CIN (0), and produces two n-bit result esteems, Total (n-1). Entirety (0) and COUT (n-1).COUT (0). The most significant utilization of a carry-save adder is to compute the fractional products in number augmentation. This takes into account designs, where a tree of carry-save adders (a purported Wallace tree) is utilized to ascertain the fractional products extremely quick. One 'typical' adder is then used to include the last arrangement of carry bits to the last halfway products to give the last duplication result. Normally, an exceptionally quick carry-look forward or carry-select adder is utilized for this last stage, so as to acquire the ideal execution.

4.1.2 Dadda Multiplier

In a famous multiplication conspire the cluster, the summation continues in a more standard, yet more slow way, to getting the summation of the fractional items .Utilizing this plan just one column of bits in the lattice is disposed of at each phase of the summation. In a parallel multiplier the halfway items are created by utilizing exhibit of AND entryways. The fundamental issue is the summation of the fractional items, and it is the time taken to perform this summation which decides the greatest speed at which a multiplier may work. The Dadda plot basically limits the quantity of adder stages required to perform the summation of halfway items.

4.1.3 Compressor

It can see a full adder as a 3:2 lossy compressor: it aggregates three one-bit sources of info and returns the outcome as a single useless number; that is, it maps 8 information esteems to 4 yield esteems. In this way, for instance, a twofold contribution of 101 outcomes in a yield of 1 + 0 + 1 = 10 (decimal number 2). The carry-out speaks to bit one of the outcomes, while the total speaks to bit zero. In like manner, a half adder can be utilized as a 2:2 lossy compressor packing four potential contributions to three potential yields.

4.1.4 Full Adder

A full adder includes twofold numbers and records for values conveyed in just as out. A one-bit full adder includes three one-bit numbers, often composed as A, B, and Cin ; An and B are the operands, and Cin is a bit

conveyed in from the past less-noteworthy stage. The full adder is typically a segment in a course of adders, which include 8, 16, 32, and so on bit double numbers. The circuit delivers a no good yield. Yield carry and total commonly spoke to by the signs Cout and S, where total = 2 × C out + S in decimal framework. A full adder can be executed from various perspectives, for example, with a custom transistor-level circuit or made out of different gates. One model execution is with S = A ⊕ B ⊕ C and C out = (A · B) + (C in · (A ⊕ B)). In this usage, the last OR gate before the carry-out yield might be supplanted by a XOR gate without modifying the

subsequent rationale. Utilizing just two sorts of gates is helpful if the circuit is being actualized utilizing straightforward IC chips which contain just one gate type for each chip. A full adder can be built from two half adders by associating An and B to the contribution of one half adder, interfacing the whole from that to a contribution to the subsequent adder, interfacing Ci to the next information as well as the two carry yields. The basic way of a full adder goes through both XOR-gates and finishes at the sum bit s.

4.1.5 Half Adder

The half adder includes two single double digits An and B. It has two yields, whole (S) and carry (C). The carry signal speaks to a flood into the following digit of a multi-digit expansion. The estimation of the total in decimal framework is 2C + S. The most straightforward half-adder configuration, imagined on the right, consolidates a XOR gate for S and an AND gate for C. The Boolean rationale for the entirety (for this situation S) will be A'B+AB' while for carry (C) will be Abdominal muscle.

4.2 Methodology

Execution of multiplier includes three stages:

1) Increase (that is - AND) each bit of one of the contentions, by each bit of the other, yielding N2 results. Contingent upon position of the duplicated bits, the wires convey various loads.

2) Decrease the quantity of incomplete items to two layers of full and half adders.

3) Gathering the wires in two numbers, and includes them with a customary adder.

The amassing of produce signals is done section astute. As every component has a likelihood of 1/16 of being one, two components being 1in similar section even abatements. For instance, in a segment with4 produce signals, likelihood of all numbers being 0 is (1 − pr) 4, just a single component being one is 4pr (1 − pr) 3, the likelihood of two components being one in the section is 6pr 2(1 − pr) 2, three ones is4pr 3(1− pr)

and likelihood of all components being 1 is pr4, where pr is 1/16. The probability statistics for a number of generate elements m in each column. Using OR gate in the

accumulation of column wise generate elements in the altered partial product matrix provides exact result in most of the cases. As can be seen, the probability of m is prediction is very low. As the number of generate signals increases, the error probability increases linearly. However, the value of error also rises. To prevent this, the maximum number of generate signals to be grouped by OR gate is kept at 4. For a column having m generates signals, OR gates are used.

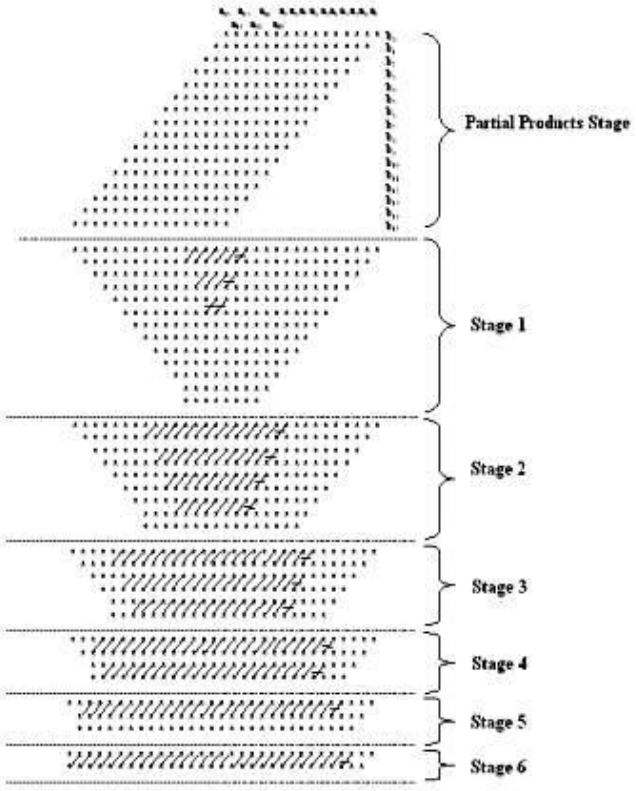

Figure 4.2: 64-bit dadda multiplier

V. SIMULATION AND RESULTS

5.1 Simulation Software

The implementation of the proposed 64-bit approximate multiplier is Xilinx 14.7 version using verilog language. Isim simulator is used for simulation and validation of result in test bench. Behavioral modeling style used to develop proposed algorithm. Artix Family is used to implementation.

5.1.1 ISE Design Suite: Logic Edition

The ISE Plan Suite: Rationale Version enables you to go from structure passage, through usage and check, to gadget programming from inside the bound together condition of the

ISE Venture Pilot or from the direction line. This release incorporates selective tools and innovations to help accomplish ideal plan results, including the accompanying:

5.1.2 ISE Design Suite: Embedded Edition

The ISE Structure Suite: Inserted Version incorporates every one of the tools and capacities of the Rationale Release with the additional abilities of the Installed Advancement Pack (EDK). This pre-arranged unit is an integrated software answer for planning implanted handling frameworks.

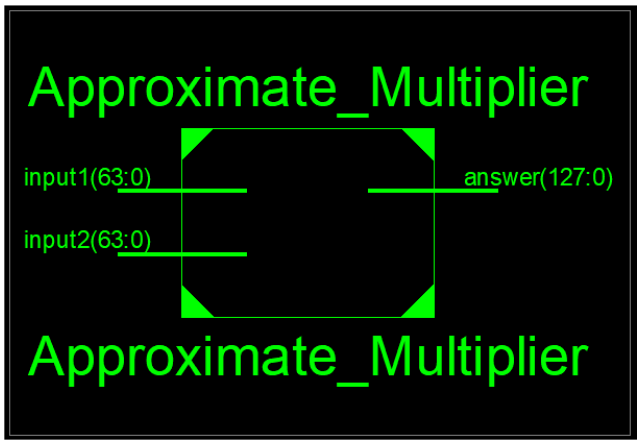

Figure 5.1: Top level View

Figure 5.1 is showing the top view of the proposed code, which includes the 64 bits input 1 and input 2and 128 bit output.

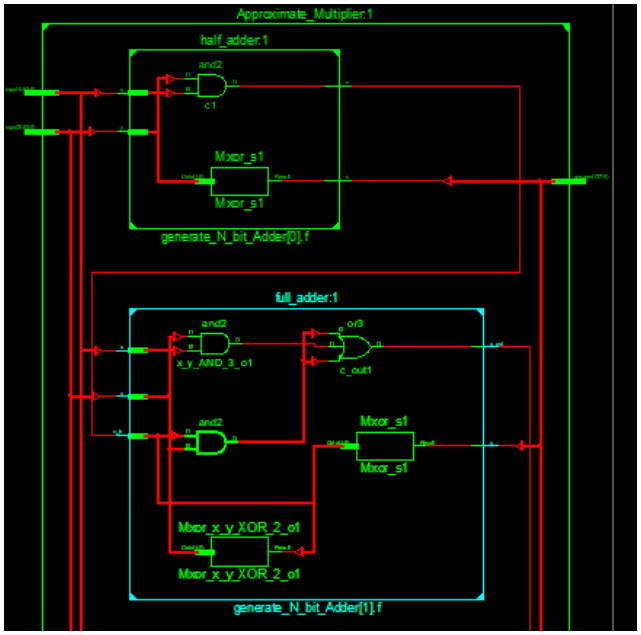

Figure 5.2: Combination of half and full adder

Half adder- The 64 bit input x and y goes to the half adder and generate sum (s) and carry (c). Full adder- The 64 bit input x and y goes to the full adder and carry (c) from the half adder. All the sums are added and generate final product.

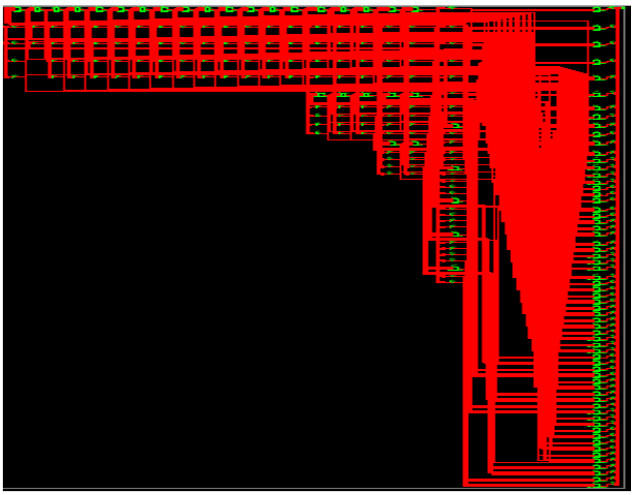

Figure 5.3: Completer register transfer level technology view

Figure 5.3 is showing technological register transfer level view. There are 6 stages of operation so many numbers of wires, full adder and other component are using.

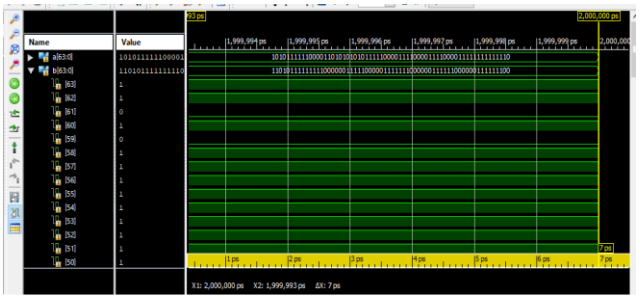

Figure 5.5: Values of input ‘b’

Figure 5.5 is showing input values of ‘b’, here binary number 1 shows the active signal and 0 shows the blank space means.

Figure 5.6: Values of internal signal q1

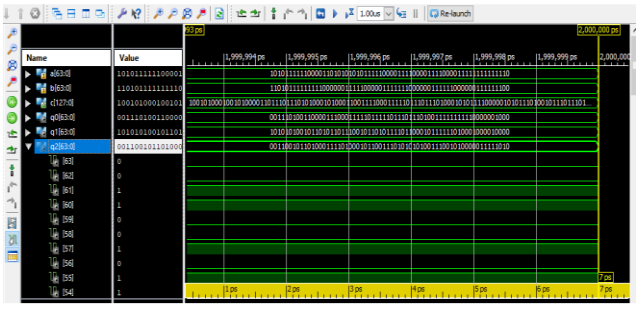

Figure 5.7: Values of internal signal q2

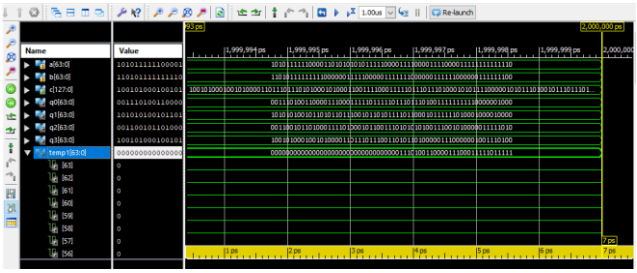

Figure 5.8: Values of internal signal temp1

Figure 5.8 shows internal signal values of temp1, temp2, temp3 and temp4.

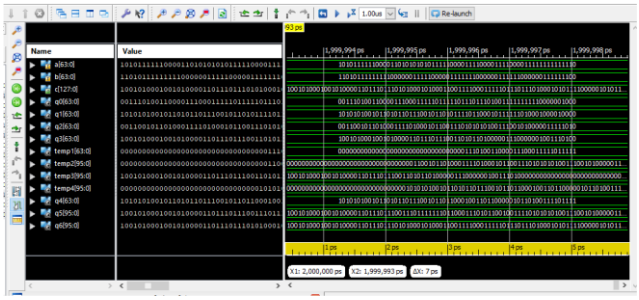

Figure 5.9: Output values during simulation

Figure 5.9 shows complete sub-module values of given input ‘a’ and input ‘b’.

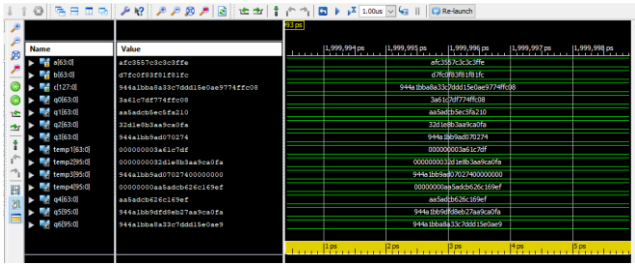

Figure 5.10: Output values during simulation in hexadecimal

5.1.1 Binary Number Input(64-bit)

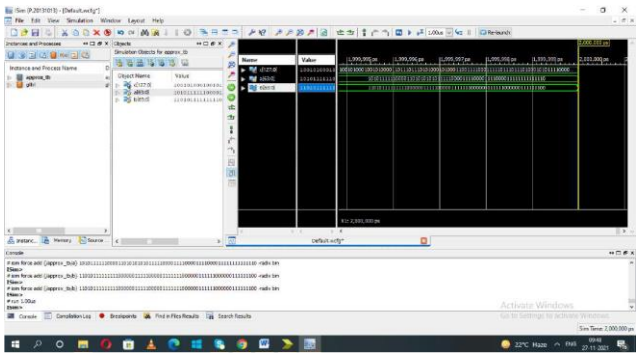

Figure 5.10: Result validation in Test Bench-1

Input (a) =

1010111111000011010101010111110000111100000111100001111111111110

Input (b) =

1101011111111110000001111100000111111100000011111100000111111100

Now simulation is done using Isim simulator. Figure 5.10 is showing test bench window.

Here ‘a’ and ‘b’ is 64-bit inputs and ‘c’ is 128-bit output. Value of’a’and ‘b’ is mentioned above. Now the output of ‘c’ is multiply of ‘a’ and ‘b’.

Output (c) =

100101000100101000011011101110100100010100011001110000111110111010010001

0101111000001010111010010111011110011111101110001100111001101

5.1.2 Hexadecimal Number Input (64-bit)

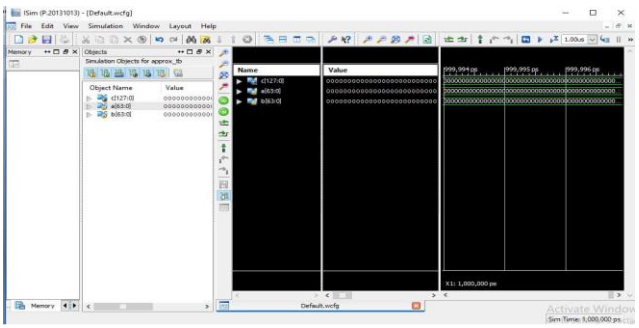

Figure 5.12: Result validation in Test Bench-3

Here ‘a’ and ‘b’ is 64 bit hexadecimal inputs and ‘c’ is 128 bit hexadecimal output. Value

of ‘a’ is aaaabbbbccccdddd and value of ‘b’ is eeeeffffaaaabbbb. The output of ‘c’ is

multiply of ‘a’ and ‘b’. So value of ‘c’ is 9f4a0feda3d720fd8d157e4b56787f6f.

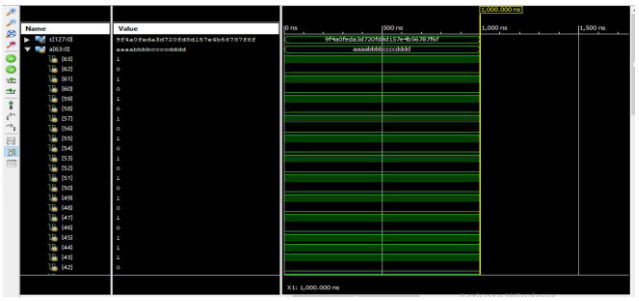

Figure 5.13: Input ‘a’signal state

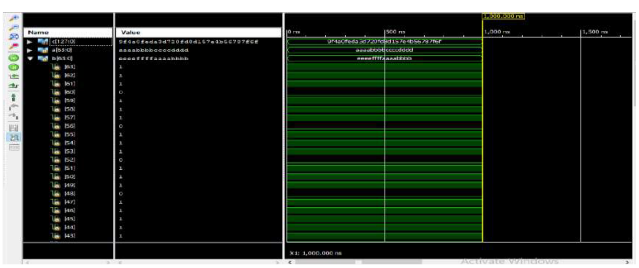

Figure 5.14: Input ‘b’signal state

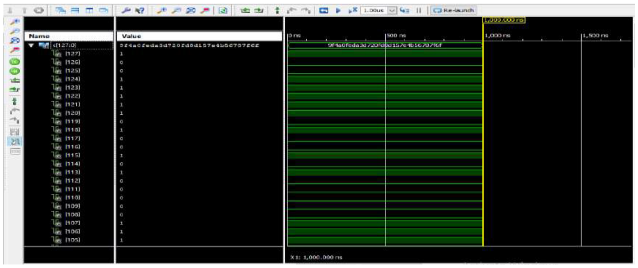

Figure 5.15: Output 'c' signal state

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	95	204000	0%
Number of fully used LUT-FF pairs	0	95	0%
Number of bonded IOBs	192	600	32%

Figure 5.22: Device utilization summary

Figure 5.22 is showing summary of components using duration proposed approximate multiplier implementation. Total number of slice look up table used 95 while availability is 204000. Look up table and flip flop pairs used 0 while availability is 95. Bonded input output block used 192 while availability is 600. Now total area is calculated from this utilization summary. Therefore, 10.66 % of area used for implementation of proposed 64-bit approximate multiplier.

Timing Details

All values displayed in nanoseconds (ns)

Timing constraint: Default path analysis

Total number of paths / destination ports: 4160 / 64

Delay: 13.745ns (Levels of Logic = 34)

Source: input1<2> (PAD)

Destination: answer<62> (PAD)

Data Path: input1<2> to answer<62>

Table 5.1: Primitive and Black Box Usage summary

Sr No.	Block Name	Quantity
1	BELS	95
2	LUT2	1
3	LUT3	31
4	LUT4	1
5	LUT5	60
6	LUT6	2
7	IO Buffers	192
8	IBUF	128
9	OBUF	64

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	2	0.001	0.698	input1_2_IBUF (input1_2_IBUF)
LUT6:I0->O	3	0.097	0.305	generate_N_bit_Adder[2].f/c_out1 (carry<2>)
LUT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[4].f/c_out1 (carry<4>)
LUT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[6].f/c_out1 (carry<6>)
LUT5:I4->O	3	0.097	0.305	generate_N_bit_Adder[8].f/c_out1 (carry<8>)
LUT3:I2->O	2	0.097	0.300	generate_N_bit_Adder[9].f/c_out1 (carry<9>)
LUT5:I4->O	2	0.097	0.515	generate_N_bit_Adder[61].f/c_out1 (carry<61>)
LUT3:I0->O	1	0.097	0.279	generate_N_bit_Adder[62].f/Mxor_s_xo<0>1 (answer_62_OBUF)
OBUF:I->O		0.000		answer_62_OBUF (answer<62>)
Total				11.610ns (1.376ns logic, 10.234ns route)

Table 5.2: Simulation Parameter

Sr No.	Parameters	Proposed Work
1	Type of Multiplier	64 X64 bit
2	Area	10.66%
3	Delay	1.376 ns
4	Power	128 microwatt
5	PDP (Power delay product)	44.032
6	Frequency	726 MHz
7	Throughput	46.51 Gbps

Table 5.3: Comparison of Simulation Results

Sr No.	Parameter	Previous Work [1]	Proposed Work
1	Order of multiplier	16 X 16	64 X 64
2	Area	18.37 mm^2	10.66 mm^2
3	Delay	0.624 ns	0.344 ns (16X16)
			1.376 ns (64X64)
4	Power	218.8 micro watt	128 microwatt
5	Power Delay Product	136.62	44.032
6	Throughput	25.64 Gbps	46.51 Gbps

Therefore proposed 64-bit approximate multiplier gives better result in term of calculated parameters. So it can be used in high speed, low area and latency.

VI. CONCLUSION AND FUTURE WORK

6.1 Conclusion

Multiplier is an important part in arithmetic processors, the current mobile applications and DSP applications need ICs with high-speed operations but low power consumption.

This dissertation presents various parameters analysis like power, area, latency, throughput, frequency and power delay product to identify the improved architectures for high-speed applications. It was also found that the power consumption of multipliers and the basic building block of more CMOS VLSI circuits depend on the switching activities of the adders. Therefore 64-bit efficient approximate multiplier is implemented and result validation using xilinx ISE 14.7 software successfully.

Approximate half-adder, fulladder, and 4-2 compressor are proposed to enhance speed of multiplier. The proposed multiplier plans can be utilized in applications with negligible misfortune in yield quality while sparing huge power and area.

Approximate multiplier are one of the fastest multiplier for the AI based FPGA-VLSI applications. The proposed research is presents the 64 X 64 bit approximate multiplier. The virtex 7 family FPGA IC is used to simulate the results. The proposed approximate multiplier id designed for the 64 X 64 bit multiplication while previous it is designed for the 16 X 16 bit multiplication. The total number of component or utilized area is 10.66 mm2 while previously it is 18.37 mm2. The total delay value is 0.344 ns (16X16) and 1.376 ns (64X64) for proposed work while 0.624 ns are for previous work. The throughput achieved by this research is 46.51 Gbps while 25.64 Gbps for previous simulation results.

6.2 Future Scope

Approximate computing has been utilized in an assortment of areas where the applications are error-tolerant, for example, mixed media preparing, AI, signal handling, logical computing, and so on Numerous applications for signal handling, PC vision and AI demonstrate an innate resistance to some computational error.

- This error flexibility can be misused to exchange off exactness for investment funds in power utilization and structure area. Since increase is a fundamental arithmetic activity for these applications, in this work we center specifically around this activity and propose a novel approximate multiplier with a powerful range choice plan.
- We structure the multiplier to have a fair-minded error appropriation, which prompts lower computational errors in genuine applications since errors counterbalance one another, as opposed to gather, as the multiplier is utilized over and again for a calculation.
- Implement approximate multiplier for 128 bit and 256 bit data multiplication.
- Modification in sub module like compressor, carry save adder etc.
- Practical implementation in any real time high speed digital multiplier.

REFERENCES

[1]. P. J. Edavoor, S. Raveendran and A. D. Rahulkar, ‘Approximate Multiplier Design Using Novel Dual-Stage 4:2 Compressors,’ in IEEE Access, vol. 8, pp. 48337-48351, 2020, doi: 10.1109/ACCESS.2020.2978773.

[2]. V. A and R. Dhavse, ‘Design of High Accuracy, Power Efficient and Area Efficient16x16 Approximate Multiplier,’ 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-6, doi: 10.1109/INDICON49873.2020.9342223.

[3]. R. Bhattacharjya, A. Kanani and N. Goel, ‘ReARM: A Reconfigurable Approximate Rounding-Based Multiplier for Image Processing,’ 2020 24th International Symposium on VLSI Design and Test (VDAT), 2020, pp. 1-4, doi: 10.1109/VDAT50263.2020.9190474.

[4]. H. Waris, C. Wang and W. Liu, ‘Hybrid Low Radix Encoding-Based Approximate Booth Multipliers,’ in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 12, pp. 3367-3371, Dec. 2020, doi: 10.1109/TCSII.2020.2975094.

[5]. S. Venkatachalam, E. Adams, H. J. Lee and S. Ko, ‘Design and Analysis of Area and Power Efficient Approximate Booth Multipliers,’ in IEEE Transactions on Computers, vol. 68, no. 11, pp. 1697-1703, 1 Nov. 2019, doi: 10.1109/TC.2019.2926275.

[6]. S. T. Bala, D. Shangavi and P. Sangeetha, ‘Area and Power Efficient Approximate Wallace Tree Multiplier using 4:2 Compressors,’ 2018 International Conference on Intelligent Computing and Communication for Smart World (I2C2SW), 2018, pp. 287- 290, doi: 10.1109/I2C2SW45816.2018.8997160.

[7]. V. V. Kavipranesh, J. Janarthanan, T. N. Amruth, T. M. Harisuriya and E. Prabhu, Power And Delay Efficient Exact Adder For Approximate Multiplier,’ 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 1896-1899, doi: 10.1109/ICACCI.2018.8554552.

[8]. P. Huang, C. Wang, R. Ma, W. Liu and F. Lombardi, ‘A Hardware/Software Codesign Method for Approximate Semi-Supervised K-Means Clustering,’ 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2018, pp. 575-580, doi: 10.1109/ISVLSI.2018.00110.

[9]. T. Su, C. Yu, A. Yasin and M. Ciesielski, ‘Formal Verification of Truncated Multipliers Using Algebraic Approach and Re-Synthesis,’ 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2017, pp. 415-420, doi: 10.1109/ISVLSI.2017.79.

[10]. A. Bonetti, A. Teman, P. Flatresse and A. Burg, ‘Multipliers-Driven Perturbation of Coefficients for Low-Power Operation in Reconfigurable FIR Filters,’ in IEEE Transactions on Circuits and Systems I: Regular Works, vol. 64, no. 9, pp. 2388-2400, Sept. 2017, doi: 10.1109/TCSI.2017.2698138.

[11]. R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha and M. Pedram, ‘RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing,’ in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 2, pp. 393-401, Feb. 2017, doi: 10.1109/TVLSI.2016.2587696.

[12]. P. Yin, C. Wang, W. Liu and F. Lombardi, ‘Design and Performance Evaluation of Approximate Floating-Point

Multipliers,' 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2016, pp. 296-301, doi: 10.1109/ISVLSI.2016.15.

[13]. A. Mehta, S. Maurya, N. Sharief, B. M. Pranay, S. Jandhyala and S. Purini, 'Accuracyconfigurable approximate multiplier with error detection and correction,' TENCON 2015 - 2015 IEEE Region 10 Conference, 2015, pp. 1-4, doi: 10.1109/TENCON.2015.7372902.

[14]. V. Mrazek and Z. Vasicek, 'Automatic Design of Low-Power VLSI Circuits: Accurate and Approximate Multipliers,' 2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing, 2015, pp. 106-113, doi: 10.1109/EUC.2015.20.

[15]. N. Maheshwari, Z. Yang, J. Han and F. Lombardi, 'A Design Approach for Compressor Based Approximate Multipliers,' 2015 28th International Conference on VLSI Design, 2015, pp. 209-214, doi: 10.1109/VLSID.2015.41.