ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2386-7876
Vol.01, Issue.03
March-2026
Pages: 319-329

DESIGN OF AREA OPTIMIZED ARITHMETIC AND LOGICAL UNIT FOR MICROCONTROLLER

[1] **M Madhavi**, [2] **Ramachandrapu Saron**, [3] **Mala Tharun**, [4] **Proddutooru Sandeep**, [5] **Saddala Rupesh**, [6] **Ukkala Ajay Kumar**

[1]Guide, Dept of ECE, ABR College of Engineering and Technology, Kanigiri, A.P.
[2,3,4,5,6]B. Tech, Dept of ECE, ABR College of Engineering and Technology, Kanigiri, A.P.

ABSTRACT: Digital design is an amazing and very broad field. The applications of digital design are present in our daily life, including Computers, calculators, video cameras etc. In fact, there will be always need for high speed and low power digital products which makes digital design a future growing business. ALU (Arithmetic logic unit) is a critical component of a microprocessor and is the core component of central processing unit. Furthermore, it is the heart of the instruction execution portion of every computer. ALU's comprise the combinational logic that implements logic operations, such as AND and OR, and arithmetic operations, such as ADD and multiplication.
Keywords: Arithmetic logic unit, Power optimization,

INTRODUCTION: As wireless communication and mobility of equipment become increasingly desirable, power dissipation of circuits has become a major concern in circuit synthesis. In performance driven synthesis of VLSI circuits, low-power design has joined the ranks of area and delay as major motivations in optimization. Digital circuit has rapidly evolved over the last twenty five years .The earliest digital circuits were designed with vacuum tubes and transistors. Integrated circuits were then invented where logic gates were placed on a single chip. The first IC chips were small scale integration (SSI) chips where the gate count is small. When technology became sophisticated, designers were able to place circuits with hundreds of gates on a chip. These chips were called MSI chips with advent of LSI designers could put thousands of gates on a single chip. At this point, design process is getting complicated and designers felt the need to automate these processes. For these reasons, we designed an ALU, which is the main part of any processor. CPU works as brain of any system and it consists of fast dynamic logic circuits and have carefully optimized structures. Of total power consumption in any processor, CPU accounts a significant portion of it. ALU also contribute to one of the highest power-density locations on the processor, as it is clocked at the highest speed and is busy mostly all the time which results in thermal hotspots and sharp temperature gradients within the execution core. Therefore this motivate us strongly for a energy-efficient ALU designs that satisfy the high-performance requirements, while reducing peak and average power dissipation. Basically ALU is a combinational circuit that performs arithmetic and logical operations on a pair of n bit operands for 8 bits. In the era of growing technology and scaling of devices up to nanometer regime, the arithmetic logic circuits are to be designed with compact size, less power and propagation delay. Arithmetic operations are indispensable and basic functions for any high speed low power application digital signal processing, microprocessors, image processing etc. Addition is most important part of the arithmetic unit rather approximately all other arithmetic operation includes addition. Thus, the primary issue in the design of any arithmetic logic unit is to have low power high performance adder cell. There are various topologies and Methodologies proposed to design full adder cell efficiently. This paper utilizes the concept of GDI technique in the design of ALU and its sub blocks as Multiplexers and Full adder.

The 3 main aspects considered in designing an ALU are:

- Power Dissipation
- Area
- Delay

POWER DISSIPATION IN CMOS CIRCUITS: The CMOS power dissipation has become a very hot topic during the last decade or so. The number of battery-powered hand-held applications, e.g. mobile phones and laptop computers is steadily increasing and more and more functions are integrated into the systems, e.g. multi-media applications in mobile phones. This is one of the driving forces for analysis of the mechanisms of power dissipation and power-reduction techniques. Another driving force is the incredible power dissipation of state-of-the-art microprocessors where heat removal and current delivery are very hard and expensive to accomplish.

LITERATURE SURVEY:

Optimization of power and delay in VLSI circuits using transistor sizing and input ordering

A fast and efficient low power design method using cell libraries is developed. This optimization routine utilizes accurate and efficient statistical power estimation methods, transistor sizing and input ordering. Algorithms that select the best cell versions to use in a circuit are developed. Circuits are first mapped with minimum-sized versions to ensure low power and then gate versions are replaced by larger versions as necessary to satisfy the delay constraint with minimal power increase. Statistical power estimation methods are used to accurately estimate power in a circuit and the switching probabilities of each node in the circuit obtained from such an analysis are used to make decisions regarding the use of sized cell versions at a local level. In addition to accurate power estimation, input ordering is also used in the algorithm to further optimize the circuit. Several unique circuit transversal algorithms are developed, each utilizing different aspects of circuit topology and node characteristics. Switching probabilities, output load, different raise and fall times and critical path analysis are all used in several different options to the main algorithm for future enhancement of the selection process. The heuristic methods developed have produced an optimization routine that takes little computation time, but produces circuits with desirable delay and power performance.

Minimum dynamic power CMOS design with variable input delay logic Glitches are power wastage in CMOS circuits and need to be eliminated for low power applications. The technique is based on new variable input delay logic, which is a design style where logic gates have different delays along different I/O paths through a single gate. Three new ways are been proposed to design gates at transistor level. Variable input delay gates can be designed by input capacitance manipulation, single NMOS transistor addition or a CMOS pass transistor addition. A first approximation is derived from lookup table and the fine tuning is done by a steepest descent method. The technique achieved an average power savings of 58% and peak power savings of 68%. Thus, the technique is scalable for large circuits as well.

Energy-efficient hard fault detection, diagnosis and isolation in the ALU Digital circuits are expected to increasingly suffer from more hard faults due to technology scaling. Especially, a single hard fault in the ALU might lead to a total failure in the embedded systems. In addition, energy efficiency is critical in these systems. To address these increasingly important problems in the ALU, we propose a novel energy-efficient fault-tolerant ALU design called Lizard. Lizard utilizes two 16-bit ALUs to perform 32-bit computations with fault detection and diagnosis. By exploiting predictable operations, fault detection is performed in a single cycle. The 16-bit ALUs can be partitioned into two 8-bit ALUs. When a fault occurs in one of the four 8-bit ALUs, Lizard diagnoses and isolates a faulty 8-bit ALU for itself. After the faulty 8-bit ALU is isolated, Lizard continues its operation using the remaining three 8-bit ALUs, which can detect and isolate another fault. In this way, Lizard can survive faults on at most two sub-ALUs increasing its lifetime and fault tolerance. We conducted comparative evaluations with an unprotected ALU, triple modular redundancy ALU, and quadruple time redundancy ALU in terms of area, energy consumption, performance, and reliability. It is demonstrated that Lizard out performs other ALU designs in most cases, especially in energy efficiency.

ADDER COMBINATIONAL:

Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs.

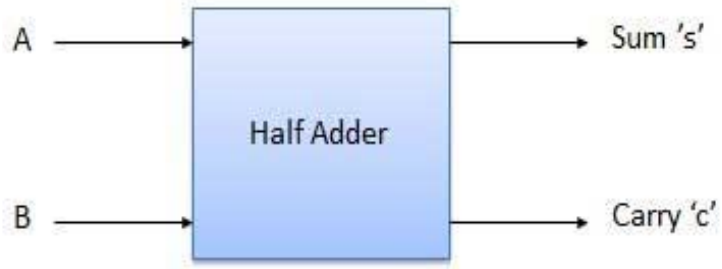
Block diagram

We're going to elaborate few important combinational circuits as follows.

Half Adder

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

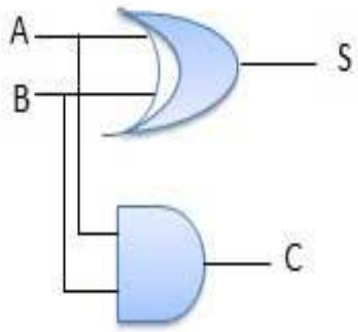
Block diagram



Truth Table

Inputs		Output
A	B	S C
0	0	0 0
0	1	1 0
1	0	1 0
1	1	0 1

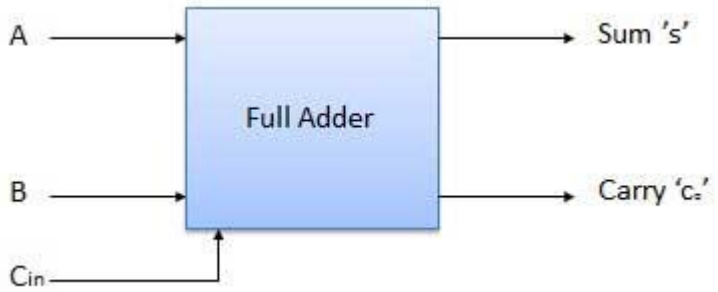
Circuit Diagram



Full Adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

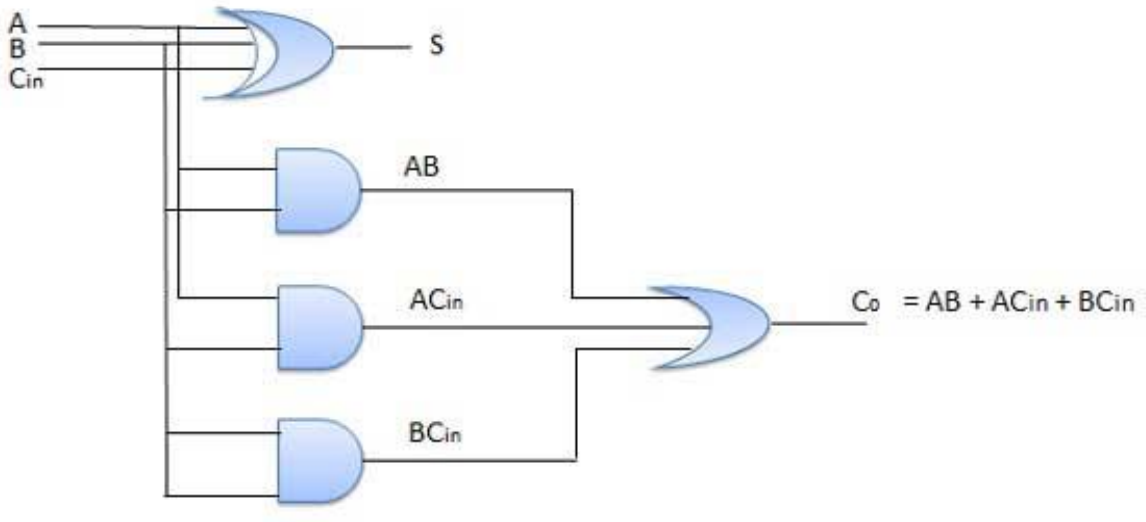
Block diagram



Truth Table

Inputs			Output	
A	B	Cin	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Diagram



N-Bit Parallel Adder

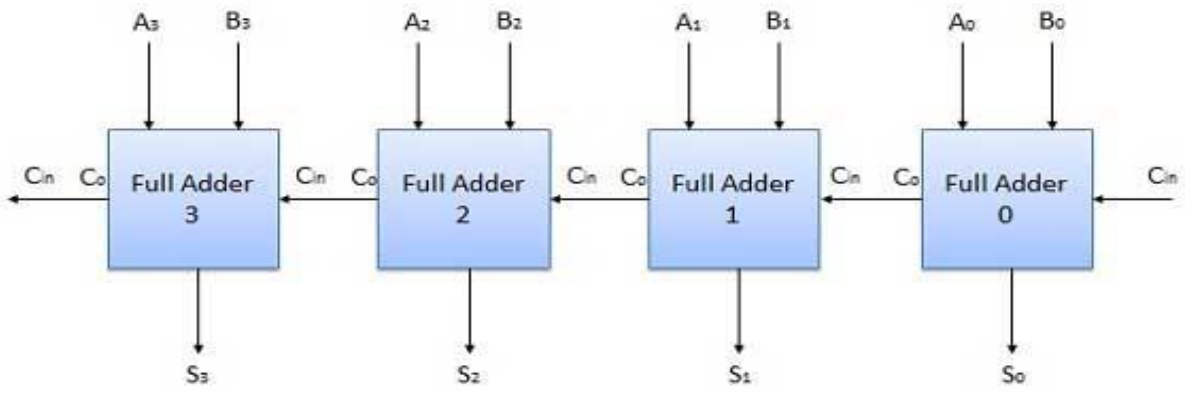
The Full Adder is capable of adding only two single digit binary number along with a carry input. But in practical we need to add binary numbers which are much longer than just one bit. To add two n-bit binary numbers we need

to use the n-bit parallel adder. It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder.

4 Bit Parallel Adder

In the block diagram, A_0 and B_0 represent the LSB of the four bit words A and B. Hence Full Adder-0 is the lowest stage. Hence its C_{in} has been permanently made 0. The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig. The four bit parallel adder is a very common logic circuit.

Block diagram



BINARY SUBTRACTION The rules for subtraction of binary numbers are again similar to decimal. When a large digit is to be subtracted from a smaller one, a 'borrow' is taken from the next column to the left. In decimal subtractions the digit 'borrowed in' is worth ten, but in binary subtractions the 'borrowed in' digit must be worth 2_{10} or binary 10_2. After borrowing from the next column to the left, a 'pay back' must occur. The subtraction rules for binary are quite simple even if the borrow and pay back system create some difficulty. Depending where and when you learned subtraction at school, you may have learned a different subtraction method, other than 'borrow and payback', this is caused by changing fashions in education. However any method of basic subtraction will work with binary subtraction but if you do not want to use 'borrow and payback' you will need to apply your own subtraction method to the problem.

RULES The rules for binary subtraction are quite straightforward except that when 1 is subtracted from 0, a borrow must be created from the next most significant column. This borrow is then worth 2_{10} or 10_2 because a 1 bit in the next column to the left is always worth twice the value of the column on its right. As their name implies, a Binary Subtractor is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers. Unlike the Binary Adder which produces a SUM and a CARRY bit when two binary numbers are added together, the binary subtractor produces a DIFFERENCE, D by using a BORROW bit, B from the previous column. Then obviously, the operation of subtraction is the opposite to that of addition. We learnt from our maths lessons at school that the minus sign, "–" is used for a subtraction calculation, and when one number is subtracted from another, a borrow is required if the subtrahend is greater than the minuend. Consider the simple subtraction of the two denary (base 10) numbers below.

123 X

– 78 Y (Subtrahend)

45 DIFFERENCE

We can not directly subtract 8 from 3 in the first column as 8 is greater than 3, so we have to borrow a 10, the base number, from the next column and add it to the minuend to produce 13 minus 8. This "borrowed" 10 is then return

back to the subtrahend of the next column once the difference is found. Simple school math's, borrow a 10 if needed, find the difference and return the borrow.The subtraction of one binary number from another is exactly the same idea as that for subtracting two decimal numbers but as the *binary number system* is a Base-2 numbering system which uses "0" and "1" as its two independent digits, large binary numbers which are to be subtracted from each other are therefore represented in terms of "0's" and "1's". **Binary Subtraction** can take many forms but the rules for subtraction are the same whichever process you use. As binary notation only has two digits, subtracting a "0" from a "0" or a "1" leaves the result unchanged as 0-0 = 0 and 1-0 = 1. Subtracting a "1" from a "1" results in a "0", but subtracting a "1" from a "0" requires a borrow. In other words 0 – 1 requires a borrow.

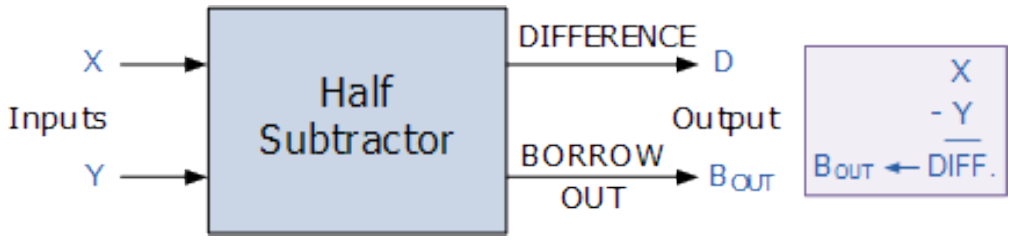
BINARY SUBTRACTION OF TWO BITS

0	1	1 (borrow)1→0	
-0	-0	-1	-1
0	1	0	1

For the simple 1-bit subtraction problem above, if the borrow bit is ignored the result of their binary subtraction resembles that of an Exclusive-OR Gate. To prevent any confusion in this tutorial between a binary subtractor input labelled, B and the resulting borrow bit output from the binary subtractor also being labelled, B, we will label the two input bits as X for the minuend and Y for the subtrahend. Then the resulting truth table is the difference between the two input bits of a single binary subtractor is given as:

HALF SUBTRACTOR CIRCUIT A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage.

Half Subtractor with Borrow-out



From the truth table of the half subtractor we can see that the DIFFERENCE (D) output is the result of the Exclusive-OR gate and the Borrow-out (Bout) is the result of the NOT-AND combination. Then the Boolean expression for a half subtractor is as follows.

For the **DIFFERENCE** bit:

D = X XOR Y = X ⊕ Y

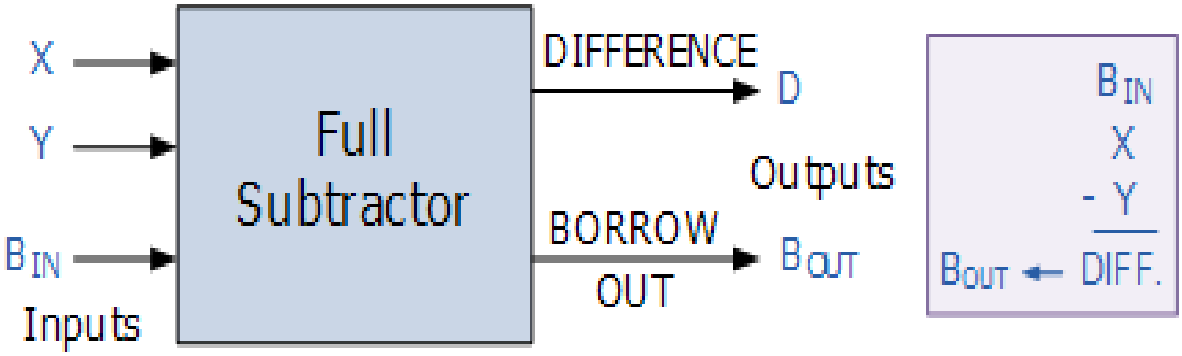
For the **BORROW** bit

B = not-X AND Y = X.Y

If we compare the Boolean expressions of the half subtractor with a half adder, we can see that the two expressions for the SUM (adder) and DIFFERENCE (subtractor) are exactly the same and so they should be because of the Exclusive-OR gate function. The two Boolean expressions for the binary subtractor BORROW is also very similar to that for the adders CARRY. Then all that is needed to convert a half adder to a half subtractor is the inversion of the minuend input X. One major disadvantage of the *Half Subtractor* circuit when used as a binary subtractor, is that there is no provision for a "Borrow-in" from the previous circuit when subtracting multiple data bits from each other. Then we need to produce what is called a "full binary subtractor" circuit to take into account this borrow-in input from a previous circuit.

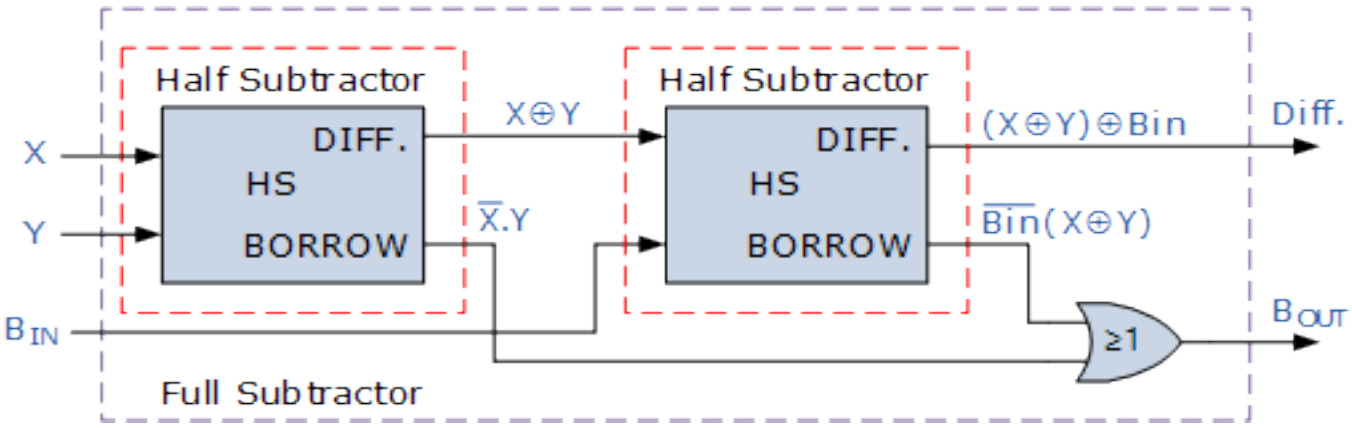
A FULL BINARY SUBTRACTOR CIRCUIT The main difference between the Full Subtractor and the previous Half Subtractor circuit is that a full subtractor has three inputs. The two single bit data inputs X (minuend) and Y (subtrahend) the same as before plus an additional *Borrow-in* (B-in) input to receive the borrow generated by the subtraction process from a previous stage as shown below.

Full Subtractor Block Diagram



Then the combinational circuit of a "full subtractor" performs the operation of subtraction on three binary bits producing outputs for the difference D and borrow B-out. Just like the binary adder circuit, the full subtractor can also be thought of as two half subtractors connected together, with the first half subtractor passing its borrow to the second half subtractor as follows.

FULL SUBTRACTOR LOGIC DIAGRAM



As the full subtractor circuit above represents two half subtractors cascaded together, the truth table for the full subtractor will have eight different input combinations as there are three input variables, the data bits and the *Borrow-in*, B_{IN} input. Also includes the difference output, D and the Borrow-out, B_{OUT} bit.

BINARY MULTIPLICATION

Binary multiplication is one of the four binary arithmetic. The other three fundamental operations are addition, subtraction and division. In the case of a binary operation, we deal with only two digits, i.e. 0 and 1. The operation performed while finding the binary product is similar to the conventional multiplication method. The four major steps in binary digit multiplication are:

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$

Note: The binary product of the two binary numbers 1 and 1 is equal to 1 only. And no additional number is borrowed or carried forward in this operation.

The binary multiplication operation is actually a process of addition and shifting operation. This process has to be continued until all the multiplier is done, and finally, the addition operation is made.

Similar to the decimal system, the multiplication of the binary numbers is done by multiplying the multiplicand with the multiplier. It is noted that the multiplication by zero makes all the bits zero, and this step may be ignored in the intermediate steps. The multiplication by 1 makes all the multiplicand value unchanged.

Binary Multiplication Table

The multiplication table for binary numbers is given below.

Binary Number	Multiplication Value
0 × 0	0
1 × 0	0
0 × 1	0
1 × 1	1

RESULT:

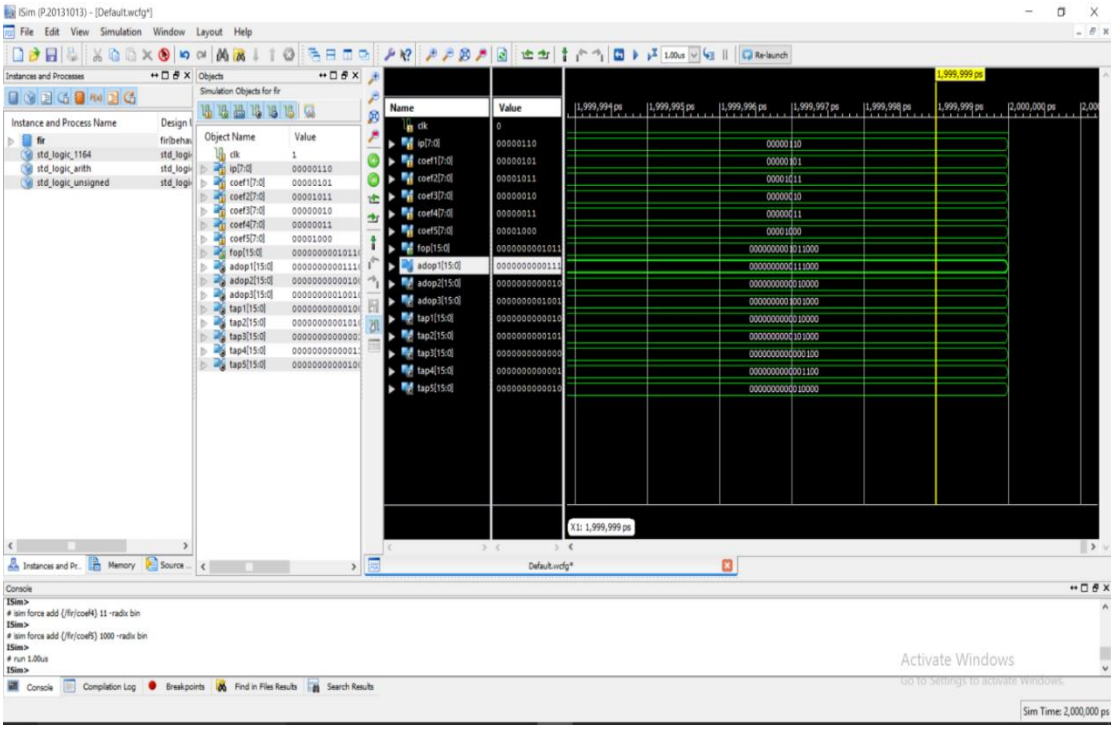


Fig: Simulation result

ADVANTAGES:
Low area occupancy
Low latency
Less power consumption with more throughput

APPLICATIONS:
FIR filters,
FFT processors,
Computers
Transmitters and Receivers modules

CONCLUSION: In this paper, we have proposed efficient VHDL behavioral coding verification method. We have also proposed several algorithms using different design levels. Our proposals have been implemented in VHDL and verified using VIVADO 2024 analyzer. We have reduced the number of bus lines and all the designs have been implemented and tested. This ALU design using VHDL is successfully designed, implemented, and tested.

FUTURE SCOPE: In computing, an Arithmetic and logic unit is a digital circuit that performs arithmetic and logic operations. It will find its requirement in the field of Nanotechnology. Commercially it would be very useful in the smart mobile phones and calculating devices. As the numbers of input bits are increased, occupied area also increases. Now the plan is to implement ALU with more number of bits by maintaining the same area. Designers are aware that fabrication process introduces error because of the usage of large capacitance. Now, the time is to turn towards smaller capacitance. Logic Designs that are realized by smaller capacitance, consumes less power and optimizes area efficiently.

REFERENCES:

1. D. Gajski and R. Khun, Introduction: New VLSI Tools,IEEE Computer, Vol. 16, No. 12, pp. 11-14, Dec. 1983.

2. http://www.forteds.com/behavioralsynthesis/index.asp

   Douglas L. Perry, VHDL, third edition, McGraw-Hill, pp.60-63, 238, July 1999.

3. S.Yalamanchali, Introductory VHDL: From simulation to synthesis, Prentice Hall, United States, 2002.

4. http://www.xilinx.com

5. B.Stephen Brown, V.Zvonko, Fundamentals of digital logic with VHDL Design 2nd Edition , Mc Graw Hill International Edition, 2005.

6. Charles H.Roth, Jr., Digital System Design using VHDL, PWS Publishing Company, 2006.

7. Mark Zwolinski, Digital System Design with VHDL, Prentice Hall, 2000.Pedroni, Digital Logic Design using VHDL.

8. S.Kaliamurthy, R.Muralidharan, VHDL Design of FPGA Arithmetic Processor International Conference on Engineering and ICT, 2007.

9. Xilinx Technologies, Xilinx Data Sheet for XC3S100E. http:// direct. xilinx.com/bvdocs/ publications/ ds312.pdf.

10. http://www.forteds.com/behavioralsynthesis/index.asp

11. Prof. S. Kaliamurthy & Ms. U. Sowmmiya,VHDL design of arithmetic processor ,Global Journals Inc.(USA) , November 2011.

12. Geetanjali and Nishant Tripathi VHDL Implementation of 32-Bit Arithmetic Logic Unit (Alu)

13. Shikha Khurana, Kanika Kaur Implementation of ALU using FPGA

14. Mr. Abhishek Gupta , Mr. Utsav Malviya , Prof. Vinod Kapse A Novel Approach to Design High Speed Arithmetic Logic Unit Based On Ancient Vedic Multiplication Technique International Journal of Modern Engineering Research (IJMER) www.ijmer.com. Vol.2, Issue.4, July-Aug 2012 pp-2695-2698. ISSN: 2249-6645 www.ijmer.com