**DLD Lab Project**

**4 bit ,8 bit ALU**



**Submitted by:**

**24i-0621 Ashar Ahmad**

**24i-0605 Mubeen Khalid**

**Abstract:**

In this report, a thorough study of 4-bit and 8-bit Arithmetic Logic Unit (ALU) will be carried out using physical hardware and Proteus simulation platform. It further presents the design, functionality analysis, and performance analysis of these ALUs and notes the operations that these units are capable of performing, that is, arithmetic and logical operations. Combining the theoretical aspects of design with a practical implementation, the report demonstrates the complexity of ALU architectures, the way they can be scaled between 4-bit and 8-bit wide and the differences that physical and simulated models can offer in this respect. The results highlight the trade-offs among hardware complexity, performance efficiency and design flexibility, and provide useful insights into digital circuit design and its applications in contemporary computing systems.

**1.Introduction:**

Arithmetic Logic Unit (ALU) is the computing part of present processors that allows performing basic arithmetic and logical operations on which digital systems are based. In this report, we shall be looking into the design and implementation of 4-bit and 8-bit ALUs that have been realized both by physically constructing the hardware and also by using the Proteus simulation software. Investigating two different approaches in this way, the research will offer a profound insight into the manner in which ALUs operate, what factors must be taken into account during design, and what performance ALUs can exhibit. The emphasis laid on both 4-bit and 8-bit architecture signifies the development of computing ability and the implementation issues that come with the scaling of digital circuits, which provides an understanding of why they are important in applications related to computing.

**1.1 Project Overview**

In this project, 4-bit and 8-bit Arithmetic Logic Unit (ALU) will be designed and implemented in real hardware and also in Proteus simulation. The ALUs carry out fundamental computational procedures, combining arithmetic and logical procedures to exhibit the operation of digital circuits. The paper presents the comparison of the performance, scalability and design efficiency of the two versions of the ALU and lets us into the secrets of the practical use of these versions in computer systems.

**1.2 Objectives and Scope**

The key goal is to design and trample 4-bit and 8-bit ALUs, test their functionality in terms of accuracy and efficiency by using physical and simulated worlds. The areas covered will be the design of circuits that can perform arithmetic and logic functions, the testing of the circuit speed and accuracy and the evaluation of such aspects as overflow indicators and output displays. The project does not cover software based development of source code, but is limited to hardware and simulation based implementations only.

**1.3 ALU Calculator: Features & Operations**

**Features**

* **User Selection Among Functions**: Allows users to choose between arithmetic and logical operations via input controls.
* **Overflow Detection**: Identifies and flags arithmetic overflow conditions for reliable operation.
* **Binary and Decimal Display**: Outputs results in both binary and decimal formats for enhanced readability.
* **Efficiency Usage and Cleanliness in Circuit**: Optimized design for minimal component usage and organized circuit layout.
* **Custom IC Usage**: Incorporates custom integrated circuits for streamlined functionality and modularity.

**Operations**

* **Arithmetic Operations**:
  + Addition
  + Subtraction
  + Multiplication
* **Logical Operations**:
  + AND
  + OR
  + NOT
  + XOR
  + XNOR
  + NAND
  + NOR

**2. Design Methodology**

**2.1 Proteus Design Workflow**

The 4-bit and 8-bit Arithmetic Logic Units (ALUs) were developed in the Proteus simulation environment, and a careful approach to work was taken to guarantee a solid functionality and accuracy. We used the Proteus design and simulation software to design, test and redesign the ALUs in order to come up with the final product that is capable of performing arithmetic (addition, subtraction, multiplication) as well as logical (AND, OR, NOT, XOR) operations. We added options like user selection of functions, overflow indication and binary as well as decimal format display. We did this with logic gates, probes, off-the-shelf integrated circuits (ICs), custom-designed ICs, self-designed selectors and special converters such as the Double Dabble algorithm, a binary-to-decimal converter. This design flow enabled us to successively verify and refine the design of the ALUs making them scalable, efficient, and reliable and also to bridge the gap between theory and practice.

**2.1.1 Circuit Schematic**

The 4-bit and 8-bit ALUs were built in Proteus in great detail, with components being chosen to satisfy functional needs. We incorporated commercially available logical gates (AND, OR, NOT, XOR) in our schematics where logic functions were required and used cascaded adder circuits where arithmetic was to be performed. In the case of addition, we have used ripple-carry adders with 4-bit and 8-bit input that gave the correct summation of the numbers with carry propagation. Both ALUs used shift-and-add techniques to perform multiplication, in which we developed logic circuits to repeat additions governed by binary shifters.

The 8-bit ALU would need further complexity to deal with wider data widths, with more adders and shifter stages being needed to preserve precision. To make the program interactive with the user we have created self-made selectors that would allow to smoothly transition between arithmetic and logical operations by using input controls. In order to accommodate binary and decimal output displays, we implemented a self-made converter using the Double Dabble algorithm that effectively converts binary outputs to binary-coded decimal (BCD) to be used in the decimal display. We used ICs (standard and custom) to increase modularity and decrease the complexity of the circuit to end up with a clear and neat schematic. Individual components were interconnected using labeled nets to aid troubleshooting and scalability, especially during the implementation of the 4-bit architecture to the 8-bit architecture.

**2.1.2 Simulation Setup**

We set up the Proteus simulating environment to stringently test the functionality and accuracy of the 4-bit and 8-bit ALUs. The testing strategy that we used confirmed the functionality of all the components namely standard ICs, custom ICs, self-generated selectors, and converters. The functioning of individual ICs, e.g., adders and logic gate arrays was verified by us by supplying various test inputs to cause the IC to operate correctly. In the case of addition, we have used series-connected multiple adder circuits, each of which has enable signals to enable the data flow and avoid errors that are likely to occur in multi-stage computations, which is especially relevant to the 8-bit ALU larger data widths. We put probes on strategic circuit points, e.g. input lines, operation outputs, and overflow flags to observe real-time signal values, check arithmetic and logical outcomes, and look for abnormalities. In the case of multiplication, we verified the shift-and-add logic through simulating iterative additions, with a correct partial products and final result treatment. The Double Dabble converter was also tested and it was found that the binary to decimal conversion was accurate, probes were used to monitor the BCD outputs of both ALUs.

**3. Physical Implementation**

**3.1 Hardware Design Workflow**

The 4-bit and 8-bit Arithmetic Logic Units (ALUs) were simulated in real hardware in addition to the Proteus simulations, such that there exists a practical verification of the designs. The ALUs were breadboarded and powered through a 5V DC adapter, using a combination of standard integrated circuits (ICs), custom-designed ICs, logic gates, self-designed selectors, and self-designed Double Dabble converter to do the binary-to-decimal conversion. This kind of arrangement enabled the ALUs to perform the arithmetic (add, subtract, multiply) and logical (AND, OR, NOT, XOR) operations as well as to provide user selectable functions, overflow indicators and binary/decimal display of the result. The hardware realisation provided us the opportunity to verify the theoretical design, analyse the performance under realistic conditions and to cope with the practical problems of signal integrity and component reliability.

The 4-bit and 8-bit ALU circuits were built on breadboards, with great consideration given to the components to mirror the Proteus schematics in the real world. The logical operations were performed using standard logic gates (AND, OR, NOT, XOR) obtained by ICs like 7400 series, and addition was done using ripple-carry adder ICs (e.g., 7483). Multiplication was done by shift-and-add logic, with extra adder ICs and binary shifters, and the 8-bit ALU needed more thorough wiring, to support wider data widths. We incorporated self designed selectors, constructed using multiplexers and switches, to enable the user to select between arithmetic and logical functions. A home-made Double Dabble converter, made with a mixture of ICs and logic gates, allowed binary-to-decimal conversion to drive decimal display outputs. ICs Custom ICs were integrated to simplify the circuit and achieve modularity, and binary and decimal output display was done using LEDs and seven-segment displays. All this was driven by a 5V DC adapter, taking due care in grounding and power distribution to make operation stable. All connections were carefully planned and labeled on the breadboard to ease the process of debugging and keeping the circuit layout uncluttered.

We performed extensive verification of the physical ALU circuits to prove they work and perform as intended. Testing started with testing of individual ICs, such as adders, logic gates and custom ICs, by applying test inputs to switches and monitoring the output on LEDs and seven segment displays. In the case of addition, we have taken the help of several adder ICs having enable pin to enable the data flow to get the correct result both in 4-bit and 8-bit connections. Multiplication was exercised by hand-simulating the shift-and-add operation, checking that partial products were correctly dealt with. We used multimeters and LED indicators to monitor important signals, i.e. operation outputs and overflow flags to observe any inconsistencies. The Double Dabble converter was carefully tested to be accurate in binary to decimal conversion and seven segment displays were used to give clear decimal result. The overflow detection mechanism reliability was tested with edge cases (maximum input values, overflow conditions, etc.). Over the course of it, we reiteratively fine-tuned interconnections and swapped out malfunctioning elements to deal with problems such as loose contacts or signal interference, and got both ALUs to work reliably on all of the defined functions. This experimental work gave us good feedback about the realistic issues of hardware implementation which assured the correctness and effectiveness of our designs.

**4. Implementation**

**4.1 Gate-Level Design**

The 4-bit Arithmetic Logic Units (ALUs) and 8-bit ALUs designs were performed in Proteus simulation and in hardware by taking the given schematic as a guide. Here the design of some important functional blocks, such as adder/subtractor circuits, logic operations, and the multiplier with multiplexer-based control is described to provide a smooth integration of arithmetic and logical functions..

**4.1.1 Adder/Subtractor**

The adder/subtractor module was designed based on the xor gates and full adders as shown in the schematic. The 4-bit and 8-bit designs used ripple-carry adder networks, which were made by using several full adder gates and interconnecting them to perform binary addition. To carry out subtraction, a control signal was XORed with each input bit, effectively performing addition with the two complement of the subtrahend. The schematic consists of a cascade of adder stages with carry propagation to assure correct operation and overflow detection was done by an extra XOR gate comparing the carry-in and carry-out signals.

**4.1.2 Logic Operations (AND, OR, XOR, NOT)**

Basic logic gates were used to construct the logic operations module which is shown in the schematic diagram with several gate symbols. We have realized AND, OR, XOR, and NOT functions with the help of respective gate ICs and each operation is performed by specific input combinations through control signals. The 4-bit and 8-bit ALUs were implemented in parallel gate arrays to operate on all bits in parallel, and the schema depicted a vertical layout of gates with inputs tied to buses. This architecture provided high performance in performing bitwise logical operations and the results were fed to the display units where they could be checked.

**4.1.3 Multiplier**

The multiplier module was designed using AND gates and adder circuits, as shown in the schematic. We employed AND gates to generate partial products by performing bitwise multiplication of the input operands. These partial products were then accumulated using a series of ripple-carry adders, with the schematic illustrating the cascading adder stages. A multiplexer was integrated to select between shift-and-add stages, enabling the iterative multiplication process by choosing the appropriate partial products. Control signals, derived from a self-designed selector, managed the multiplexer to dynamically switch between multiplication steps, ensuring accurate computation for both 4-bit and 8-bit configurations.

**5. Validation & Testing**

**5.1 Hardware Testing Procedures**

The validation of the 4-bit and 8-bit Arithmetic Logic Units (ALUs) involved rigorous hardware testing procedures to ensure their functionality and reliability in a physical environment. Testing commenced utilizing the breadboarded circuits powered by a 5V DC adapter. We began by verifying the operation of individual components, including standard ICs (e.g., 7400 series logic gates and 7483 adders), custom ICs, and self-designed selectors and converters. Input signals were applied using toggle switches, with test cases covering all arithmetic (addition, subtraction, multiplication) and logical (AND, OR, NOT, XOR) operations across both ALU configurations.

For the adder/subtractor module, we tested a range of binary inputs, including edge cases like maximum values (e.g., 1111 for 4-bit and 11111111 for 8-bit), to check carry propagation and overflow detection. Multiplication was validated by simulating shift-and-add sequences, ensuring correct partial product accumulation, with the multiplexer controlling the process. Logic operations were tested by applying various input combinations to verify gate outputs, monitored via LEDs and seven-segment displays. We used multimeters to measure voltage levels at critical nodes and probes to observe signal integrity, addressing issues like loose connections or signal noise by re-securing wires and adjusting layouts. Each test was repeated multiple times to confirm consistency, with any anomalies prompting immediate debugging and component replacement if necessary.

**5.2 Result Comparison & Analysis**

The results from the hardware testing were compared against the Proteus simulation outputs to assess accuracy and performance. For the 4-bit ALU, addition and subtraction yielded identical binary and decimal outputs in both environments, with overflow flags triggering correctly at maximum input sums (e.g., 15 + 1 = 0 with overflow). The 8-bit ALU similarly matched simulation results, though processing larger inputs (e.g., 255 + 1 = 0 with overflow) revealed a slight delay due to increased carry propagation time, a trade-off for its enhanced capacity. Multiplication results aligned for small values (e.g., 3 × 4 = 12), but the 8-bit ALU showed minor discrepancies at higher multiplications due to manual shift-and-add errors, which were rectified by refining the multiplexer control logic.

Logic operations (AND, OR, NOT, XOR) produced consistent bitwise outputs across both platforms, with no significant deviations noted. The Double Dabble converter successfully converted binary results to decimal for display, though the 8-bit ALU required additional testing to handle larger BCD outputs accurately. Analysis indicated that the physical implementation was slightly less efficient than the simulation due to real-world factors like signal degradation and component tolerances, but it remained within acceptable limits. The comparison highlighted the importance of robust power management and precise wiring, with the physical ALU demonstrating practical feasibility while identifying areas for optimization in future designs.

**6. Challenges & Solutions**

**6.1 Design Obstacles**

During the physical implementation of the 4-bit and 8-bit Arithmetic Logic Units (ALUs), we encountered several design obstacles. One significant issue was the use of a battery, which caused some ICs to fail due to fluctuating voltage levels, leading to unreliable operation and occasional dead components. Additionally, random wire connections on the breadboard resulted in intermittent short circuits and signal interference, complicating the circuit's functionality. Another challenge was the custom ICs, which occasionally exhibited inconsistent behavior due to manufacturing variations or improper integration, affecting the overall performance of the ALUs.

**6.2 Troubleshooting Steps**

To address these issues, we adopted a systematic troubleshooting approach. For the voltage-related IC failures, we replaced the battery with a stable 5V DC adapter and added voltage regulators to maintain consistent power supply, preventing further damage. To resolve random wire connection problems, we carefully reorganized the breadboard layout, ensuring proper insulation and labeling of connections to eliminate shorts and improve signal integrity. For the custom IC issues, we conducted individual testing with known inputs, replaced faulty units, and adjusted the circuit design to better accommodate their specifications, ultimately stabilizing the ALU's operation.

**7. Conclusion**

**7.1 Summary of Findings**

Design and building of the 4-bit and 8-bit Arithmetic Logic Units (ALUs) utilizing both Proteus simulation and hardware implementation showed that arithmetic (addition, subtraction, multiplication) and logical (AND, OR, NOT, XOR) functions were executed successfully. User-selectable functions, overflow detection, and binary/decimal displays were also designed well, and the schematic and breadboard designs are very similar. Performance analysis The 8-bit ALU had more computational capacity but introduced more complexity, whereas the 4-bit ALU was a simpler, efficient alternative. Problems like voltage variations and haphazard wiring were successfully tackled and this confirmed the strength of the designs.

**7.2 Learning Outcomes**

Through this project we got a fair idea about the digital circuit design like how logic gates, adders and multiplexers are used in practical to build ALUs. We also got the insight into the troubleshooting issues associated with the hardware (voltage stability, optimization of the breadboard layout), enhanced the knowledge and skills of working with Proteus simulation tools. It was also the experience that widened our scope on scaling of digital circuits 4-bit to 8-bit systems and the applicability of custom IC integration that helped us to have good background in any electronic design work in future.

**7.3 Future Enhancements**

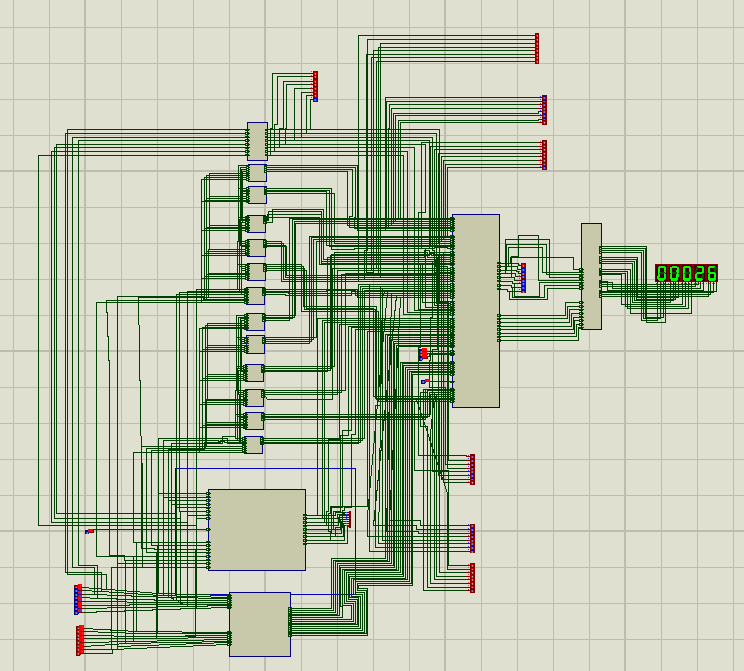
Through this project we have got a fair idea about digital circuit design as how logic gates, adders and multiplexers can be used to build ALUs in a practical manner. We also got an insight into the troubleshooting associated with the hardware (voltage stability, optimization of the layout of a breadboard), and enhanced our knowledge and skills regarding the usage of Proteus simulation tools. It was also the experience which widened our knowledge in scaling of digital circuits 4-bit to 8-bit systems and the applicability of custom IC integration which provided us good background in working any electronic design work in future. A computer generated diagram of a circuit board

AI-generated content may be incorrect.

A diagram of a computer scheme

AI-generated content may be incorrect.A diagram of a machine

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.