**Department of Computer Science**

**Air University**

**PROJECT REPORT**

**DIGITAL LOGIC DESIGN (DLD)**

**Project Title: Extended ALU with 8-bit RAM**

**Group Members:**

Muhammad Hammad (243545)

Muhammad Umair (243587)

Muhammad Faisal (248784)

Date of Submission: June 2025

**1. Objective**

To design a **4-bit Extended ALU** that supports **multiple logical and arithmetic operations**, with all outputs stored in an **8-bit RAM** constructed using **D flip-flops**, **multiplexers**, and **control logic** .

**2. Major Components Used**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **Component** | **Description** | | Logic Gates | For AND, OR, NOT, XOR operations. | | Multiplexers | Used for operation selection and output routing | | Full Adders | For binary addition, subtraction, increment, and decrement operations | | D Flip-Flops | Construct 8-bit RAM (1 flip-flop per bit per memory cell) | | Clock Pulse | Controls timing of data write to RAM | | Control Inputs | Operation selector (S0, S1, S2, S3), Clock Enable (CE), RAM Write/Read Control | | LEDs | Output visualization from ALU and RAM | | Switches | Inputs for operands A[3:0], B[3:0], and control lines | |

**3. ALU Design Overview**

**3.1 ALU Width**

* The ALU is **4-bit wide**, i.e., it performs operations on 4-bit operands A and B.

**3.2 Supported Operations (Controlled by 4-bit input S3 S2 S1 S0)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Code (S3–S0** | **Operation** | **Description** | | 0000 | Pass A | ALU = A | | 0001 | A + B | ALU = A + B | | 0010 | A + 1 | ALU = A + 1 | | 0011 | A′ | ALU = NOT A | | 0100 | A PLUS B | ALU = A + B | | 0101 | A - B | ALU = A - B (2’s comp logic) | | 0110 | A + 1 | ALU = A incremented | | 0111 | A - 1 | ALU = A decremented | | 1000 | A \* B | ALU = Multiplication | |

**3.3 Internal Modules**

* **Logic Extenders (LE0–LE3):** Handle bitwise operations like NOT, AND, OR, and pass-through for logic.
* **Arithmetic Extenders (AE0–AE3):** Handle arithmetic logic (add, subtract, increment).
* **MUX-based Operation Selector:** 8-to-1 multiplexers choose final ALU output based on S3–S0 control inputs.

**4. RAM Design (8-bit using Flip-Flops)**

**4.1 Construction**

* Composed of **8 individual D flip-flops**, each representing one bit of memory.
* RAM takes **4-bit ALU output**, padded or extended to 8 bits for storage.

**4.2 Control Signals**

* **Clock Enable (CE):** Triggers data write on rising edge.
* **MUX Select (M0–M7):** Chooses which ALU output bit goes into each RAM bit.
* **Write Control:** Flip-flops latch the data only when CE is active and clock transitions.

**4.3 Read/Write Mechanism**

1. On **write cycle**:

ALU result is routed via 8 MUXes to D inputs of flip-flops.

On **clock pulse + CE high**, the data is written.

1. On **read cycle**:

Flip-flop Q outputs are routed directly to LEDs for observation.

**5. Circuit Walkthrough**

**Step 1: Operand Setup**

1. Inputs A3 A2 A1 A0 and B3 B2 B1 B0 are selected via switches.

**Step 2: Operation Select**

1. Control lines S3 S2 S1 S0 determine the operation (e.g., add, subtract, NOT A).

**Step 3: ALU Output**

1. Depending on the operation:

**LE** units perform logical functions.

**AE** units use full adders and NOT gates to generate arithmetic results.

1. ALU output wires are routed into the MUX block for RAM storage.

**Step 4: Write to RAM**

1. When **Clock Enable = 1**, and a **clock pulse** is generated:

The ALU output is written into the 8-bit RAM made from flip-flops.

**Step 5: Display**

1. The RAM output is displayed through LEDs to confirm proper data storage.

**6. Testing & Results**

| **A** | **B** | **Control Code** | **Operation** | **ALU Output** | **Stored RAM Output** |
| --- | --- | --- | --- | --- | --- |
| 0101 | 0010 | 0001 (4) | A + B | 0111 | 00000111 |
| 0100 | 0010 | 0101 (5) | A - B | 0010 | 00000010 |
| 0011 | N/A | 0011 (3) | NOT A | 1100 | 00001100 |
| 0010 | 0010 | 1000 (8) | A \* B | 0100 | 00000100 |

All tested results were correctly stored in RAM and visible via LEDs.

**7. Conclusion**

This project successfully demonstrates:

1. A **functional ALU** with multiple logic and arithmetic operations.
2. **Efficient memory interaction** using RAM made purely from **flip-flops**, without any registers.
3. A working **write-read cycle** using clock enable and control logic.
4. **Clear visual output** through LED indicators.