Lab 1A: 8-bit Arithmetic Logic Unit (ALU)

Objective

The objective of this lab is to design and verify an **8-bit Arithmetic Logic Unit (ALU)** that performs arithmetic and logical operations. The ALU should handle addition, subtraction, logic functions, and shift operations while also generating **status outputs** such as Zero, Carry, and Overflow. This experiment emphasizes **datapath design, control logic, and verification** for FPGA-based systems.

Design Requirements

The ALU must meet the following requirements:

- **Inputs**:
    - Two 8-bit operands (`a`, `b`)
    - 3-bit operation select (`op_sel`)
- **Supported Operations**:
    - ADD (Addition)
    - SUB (Subtraction)
    - AND (Bitwise AND)
    - OR (Bitwise OR)
    - XOR (Bitwise XOR)
    - NOT (Bitwise NOT of operand A)
    - SLL (Shift Left Logical)
    - SRL (Shift Right Logical)
- **Outputs**:

    - 8-bit result (`out`)
    - **Zero Flag** (asserted when result = 0)
    - **Carry Flag** (for addition, subtraction, and shifts)
    - **Overflow Flag** (for signed arithmetic operations)
- **Implementation Goal**: Optimized for FPGA implementation with efficient logic utilization.

Design Methodology

1. **Truth Table & Operation Encoding**
    - Each of the 8 operations was assigned a **3-bit code** (000–111).
    - A truth table was developed mapping each `op_sel` value to its corresponding operation.
2. **Datapath Design**
    - Arithmetic operations (ADD, SUB) use a temporary 9-bit register to capture the carry-out.
    - Logical operations (AND, OR, XOR, NOT) operate bitwise on the inputs.
    - Shift operations (SLL, SRL) use operand A as the source and assign carry to the shifted-out bit.
3. **Status Flag Logic**
    - **Zero Flag**: Asserted when result = 0.
    - **Carry Flag**: Extracted from the 9th bit in addition/subtraction and MSB/LSB in shifts.
    - **Overflow Flag**: Detected based on 2's complement rules for signed addition/subtraction.
4. **FPGA Considerations**

- Efficient combinational logic was used instead of cascaded if-else structures.
- Overflow logic was minimized to reduce LUT usage.

Simulation and Verification

Testbench Setup

- A testbench was created with different input patterns to test each operation.
- Stimulus covered **boundary cases** (e.g., large positive/negative values for overflow).
- `$display` statements were used to print results during simulation.

Observed Results

1. **Addition (op_sel = 000)**
   - Correctly performed addition.
   - Carry flag asserted when sum exceeded 8-bit range.
   - Overflow correctly detected for signed addition.
2. **Subtraction (op_sel = 001)**
   - Correct subtraction with borrow handling.
   - Carry flag worked as expected in subtraction mode.
   - Overflow detected when signed range was exceeded.
3. **Logical Operations (AND, OR, XOR, NOT)**
   - Produced correct bitwise outputs.
   - Zero flag asserted when all bits of result were 0.
4. **Shift Operations (SLL, SRL)**

    - Left shift correctly moved bits left with MSB as carry.
    - Right shift correctly moved bits right with LSB as carry.

5. **Status Flags Verification**
    - **Zero flag**: Asserted only when output was 0.
    - **Carry flag**: Asserted correctly for addition, subtraction, and shifts.
    - **Overflow flag**: Worked correctly for signed arithmetic tests (e.g., 127+1, 128-255).

Results and Discussion

- The ALU **successfully implemented all 8 operations** with correct status flag behavior.
- Simulation confirmed correctness across a wide range of test cases.
- **Zero, Carry, and Overflow flags** were validated for correctness and can be used in higher-level CPU designs.
- The design is modular and scalable to larger bit-widths (e.g., 16-bit or 32-bit).
- Optimizations ensure efficient mapping to FPGA hardware resources.

Conclusion

In this lab, an **8-bit ALU** was designed, implemented, and simulated. The ALU supported arithmetic, logical, and shift operations with status flag outputs. Simulation verified correct functionality for all operations, including edge cases. This experiment provided practical experience with **combinational logic design, flag generation, and FPGA-oriented optimization**.