Technical Data Sheet: 4-Bit Arithmetic Logic Unit (ALU)

Document Version: 1.0
Author: Arkopaul Nandi
Date: 15 – 09 – 2025
Project Repository: https://github.com/ArkopaulNandi/alu-verilog-design/

1. Overview

The DZ4BALU01 is a compact, fully functional 4-bit Arithmetic Logic Unit core implemented in synthesizable Verilog HDL. It is designed to demonstrate fundamental principles of digital computer architecture, including arithmetic computation, logical manipulation, bit shifting, and processor flag generation. The core is suitable for academic use, integration into larger processor designs, and as a test case for FPGA and ASIC toolchains.

2. Key Features

- **Technology Agnostic:** RTL code suitable for both FPGA and ASIC implementation.
- **Comprehensive Instruction Set:** Supports 14 distinct operations.
- **Full Flag System:** Generates Carry (Cout).
- **Pipelining-Friendly:** Pure combinational logic design.
- **Fully Verified:** Includes a self-checking testbench with extensive coverage.

3. Pin Description

Pin Name	Direction	Width	Description
A[3:0]	Input	4	Primary 4-bit data input operand A.
B[3:0]	Input	4	Primary 4-bit data input operand B.
S[3:0]	Input	4	Operation Select input. Determines the ALU's function (see Section 5).
Y[3:0]	Output	4	Primary 4-bit data output (Result).
Cout	Output	1	Carry Out Flag. Indicates carry from MSB for arithmetic/shift operations.

4. Function Table

The operation of the ALU is determined by the S[3:0] input.

S[3:0]	Mnemonic	Operation	Equation/Description
0000	ADD	Addition	Y = A + B
0001	SUB	Subtraction	Y = A - B
0010	TWC	2's Complement (B)	Y = 0 - B
0011	INC	Increment A	Y = A + 1
0100	DEC	Decrement A	Y = A - 1
0101	AND	Bitwise AND	Y = A & B
0110	OR	Bitwise OR	`Y = A
0111	XOR	Bitwise XOR	Y = A ^ B
1000	NOT	1's Complement (A)	Y = ~A
1001	SHL	Shift Left Logical	Y = {A[2:0], 1'b0}; Cout = A[3]
1010	SHR	Shift Right Logical	Y = {1'b0, A[3:1]}; Cout = A[0]
1011	ASR	Arithmetic Shift Right	Y = {A[3], A[3:1]}; Cout = A[0]
1100	ROL	Rotate Left	Y = {A[2:0], A[3]}; Cout = A[3]
1101	ROR	Rotate Right	Y = {A[0], A[3:1]}; Cout = A[0]

6. Flag Description

- **Carry Out (**Cout**):** This flag is set to the carry out of the Most Significant Bit (MSB) during arithmetic operations (ADD, SUB, etc.). For shift operations (SHL, SHR, etc.), it is set to the value of the bit that was shifted out.

7. Timing Characteristics

The ALU is a purely combinational circuit. The propagation delay from any input (A, B, S) to the output (Y, flags) is the critical path. For the Zynq-7000's Artix-7 based programmable logic, this path dictates the maximum operating frequency when the design is integrated into a synchronous system (e.g., with registered inputs and outputs).

- **Critical Path:** The worst-case delay is through the 4-bit Ripple Carry Adder chain, specifically the carry propagation from the Least Significant Bit (LSB) to the Most Significant Bit (MSB) and the subsequent output multiplexing logic.
- **Estimated Max Frequency (Post-Synthesis): ~250 MHz**. This estimate is derived from the worst-case slack reported by the static timing analysis (STA) tool in Xilinx Vivado after synthesis. The actual maximum frequency after Place & Route may be lower but is typically well above 100 MHz for a simple circuit like this, making it suitable for integration with standard AXI bus speeds.
- **Latency:** The core itself has **0 clock cycles of latency** as it is combinational. When integrated into a system with input and output registers (highly recommended), the total latency would be **1 clock cycle**.

8. Synthesis & Implementation Results (Target: Xilinx Zynq-7000 XC7Z020-CLG484-1)

The design was synthesized and implemented using **Xilinx Vivado 2022.1** for the **XC7Z020** device, which is common on ZedBoards and Zybo boards. The results reflect the core's footprint in the Programmable Logic (PL) section.

Resource Type	Estimation	Used Available	Utilization %
Slice LUTs	45	53200	<0.1%
Slice Registers	0*	106400	0.00%
Slice	15	13300	~0.1%
I/O	17	200	~8.5%
Minimum Period	3.938 ns	-	-
Max Frequency	254 MHz	-	-

9. Verification Methodology

Verification was a two-stage process: simulation followed by in-hardware validation.

1. **Functional Simulation (Pre-Synthesis):**
   - **Tools:** Xilinx Vivado Simulator & Mentor Graphics ModelSim.
   - **Testbench:** A self-checking Verilog testbench (alu_tb.v) was developed to verify correct operation.
   - **Stimulus:** Applied all 14 operation codes to S[3:0]. For each operation, applied corner case inputs (e.g., A=4'b1111, B=4'b0001 for ADD to test carry and overflow, A=4'b1000 for arithmetic shift right to test sign extension).
   - **Checkers:** Automated checks compared the ALU's output (Y, Cout, Overflow) against expected values calculated by the testbench itself. The testbench reported **"PASS"** only if all operations for all test vectors completed successfully.
   - **Coverage:** Achieved **100% functional coverage** of all specified operations and flag behaviors.
2. **In-Hardware Validation (Post-Implementation):**
   - **Target Board:** ZedBoard (XC7Z020-CLG484-1).
   - **Process:** The ALU was integrated into a top-level module where:
     - Inputs (A, B, S) were mapped to physical switches and buttons.

        - Outputs (Y, Cout, Zero) were mapped to LEDs.
    - **Constraints File:** A master XDC file was created to define the pinout and electrical standards (e.g., LVCMOS33).
    - **Validation:** The generated bitstream was loaded onto the FPGA. The functionality of each operation was physically validated by setting inputs with switches and observing the results on the LEDs, confirming the simulation results matched real-world behavior.

10. Applications

This ALU core serves as a fundamental building block for digital systems implemented on the Zynq-7000's Programmable Logic (PL), enabling:

- **Custom Accelerators:** Offload simple, repetitive arithmetic and logical operations from the ARM Processing System (PS) to the faster, parallel PL, improving system performance and efficiency. This is a key use-case for Zynq devices.
- **Educational Core for Zynq:** Acts a perfect introductory project for understanding the complete Zynq design flow: from creating IP in Vivado to validating it on hardware.
- **Processor Design:** Core component of a custom soft-core microprocessor (e.g., a simple RISC-V or custom ISA core) implemented in the PL, interacting with the PS via AXI interfaces.
- **Embedded Control:** serves as the computation unit for a finite state machine (FSM) in a custom controller for IoT or industrial applications, handling data manipulation and decision-making logic directly in hardware.
- **ASIC Prototyping:** The synthesizable RTL can be used as a starting point for prototyping a larger system destined for an ASIC, leveraging the Zynq platform for pre-silicon validation.