

Nano-Stack: 8-Bit Stack Processor Design

1. Project Objective

To design, implement, and verify a fully functional Zero-Operand (Stack-Based) Microprocessor from scratch using Verilog HDL. This project serves as a demonstration of Computer Architecture fundamentals, FSM design, and RTL verification.

2. Engineering Choice: Why a Stack Machine?

Instead of a standard Register-based architecture, I implemented a Stack Machine.

- Reasoning: This architecture eliminates the need for complex operand decoding (e.g., ADD R1, R2). Operations implicitly act on the Top-of-Stack (TOS), allowing me to focus on perfecting the Control Logic vs. Datapath separation.

3. Architecture Breakdown

The design is modular, consisting of two primary blocks:

- The Datapath (Hardware Stack):
 - A 16-deep LIFO memory with a dynamic Stack Pointer (SP).
 - Custom Feature: Integrated ALU logic allows "Read-Modify-Write" operations (like ADD/SUB) to execute in a single cycle by manipulating the memory and pointer simultaneously.
- The Control Unit (FSM):
 - A 2-stage Finite State Machine (Fetch -> Execute).
 - Handles opcode decoding and generates precise control signals to prevent race conditions.

4. Verification & Results

The design was simulated in Xilinx Vivado.

- Test Case: A hardcoded ROM program calculating $5 + 3 - 2$.
- Expected Result: 6
- Simulation Outcome: The waveform shows the stack transitioning $5 \rightarrow 3 \rightarrow 8 \rightarrow 2 \rightarrow 6$. The done signal asserts successfully, confirming logical correctness.

| Name | Value | 144,996 ps | 144,997 ps | 144,998 ps | 144,999 ps | 145,000 ps |
|-------------------|-------|------------|------------|------------|------------|------------|
| clk | 0 | | | | | |
| rst | 0 | | | | | |
| > result_out[7:0] | 06 | | | | | |
| done | 1 | | | | | |