ARM968E-S Processor Implementation on FPGA

Overview

The ARM968E-S processor implemented in this project features a pipelined architecture with five stages: Instruction Fetch (IF), Instruction Decode (ID), Execution (EXE), Memory Access (MEM), and Write-Back (WB). This processor is developed using Verilog and deployed on an Altera Cyclone II EP2C35F672C6 FPGA. Each pipeline stage handles specific tasks essential to the processor's functionality, enabling efficient instruction execution.

The data flow starts with fetching an instruction from memory in the IF stage. The instruction is decoded in the ID stage to generate control signals and determine operand sources. The EXE stage performs the specified ALU operations, while the MEM stage handles any memory access required by load or store instructions. Finally, the WB stage completes the cycle by updating the Register File with the computed result.

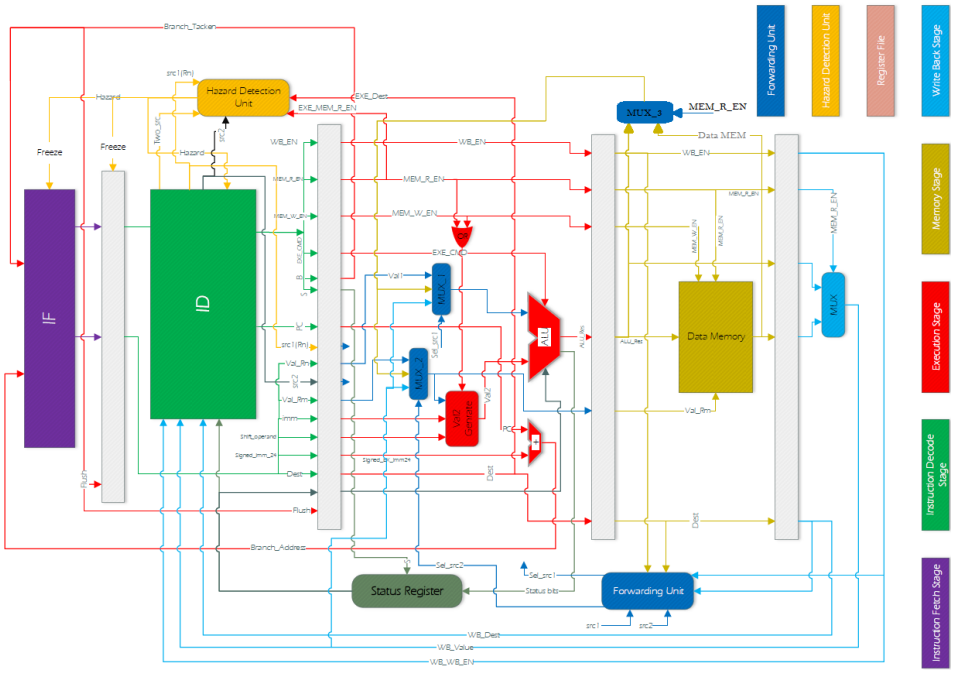

Figure 1: ARM Data path

Instruction Fetch

The IF (Instruction Fetch) stage is responsible for loading instructions from the instruction memory and feeding them into the pipeline. The Program Counter (PC) register holds the address of the instruction to be fetched. Each cycle, the PC increments by four to accommodate 32-bit instruction words. For branch instructions, a multiplexer (MUX) selects between PC + 4 and the target branch address based on the Branch Taken signal.

The IF stage also includes a Freeze signal connected to the PC and pipeline registers, ensuring stability during data hazards and control flow changes.

Instruction Decode

The ID (Instruction Decode) stage decodes the fetched instruction and generates necessary control signals. It contains the Register File, which includes 15 general-purpose registers. The Control Unit in the ID stage generates signals such as Write Back Enable, Memory Read Enable, Memory Write Enable, and Execution Command, based on the decoded instruction type.

A Condition Check unit evaluates conditional instructions using the Status Register, which stores flags like Zero (Z), Negative (N), Carry (C), and Overflow (V). The ID stage thus ensures the correct operation of control and data flow within the pipeline.

Execution

The EXE (Execution) stage performs arithmetic and logical operations as specified by the ALU (Arithmetic Logic Unit). In addition to the ALU, the EXE stage includes components for generating ALU inputs and updating the Status Register. The Val-2-Generator, a module within this stage, provides operand manipulation functions such as shifting and masking.

Results from the ALU are passed to the next stages, and the Status Register is updated with relevant flags based on ALU output, informing the Condition Check unit in ID about the status of previous operations.

Memory Access

The MEM (Memory Access) stage handles load and store operations by interacting with the Data Memory. In load instructions, data is fetched from memory and passed to the WB stage, while in store instructions, data from the Register File is written to memory at a specified address.

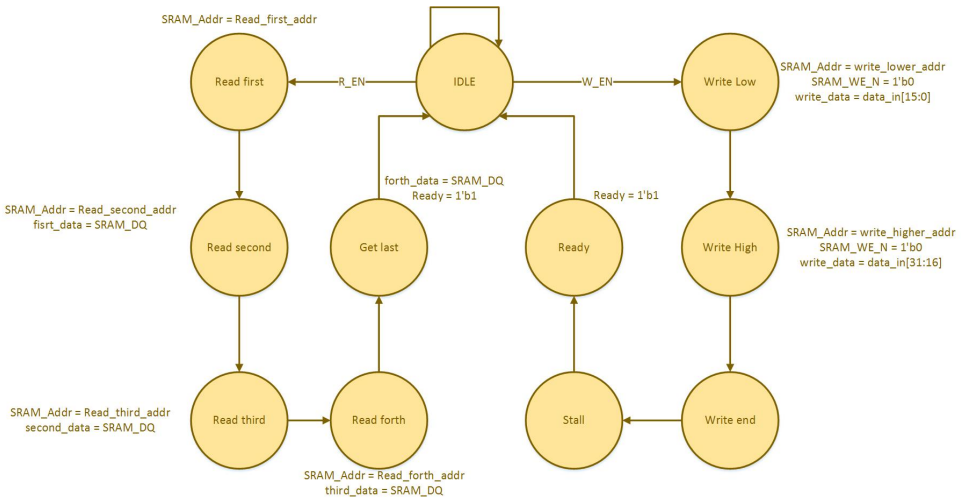

Figure 2: SRAM Controller

The MEM stage plays a crucial role in memory management, handling memory read and write enable signals based on the decoded instruction and ensuring smooth interaction with external memory components.

Write Back

The WB (Write Back) stage writes results back to the Register File, finalizing the execution of an instruction. If the instruction involves memory access, the loaded data from MEM is written to the Register File; otherwise, the ALU result is saved. This stage ensures the data produced by the processor's operations is available for future instructions.

Cache Implementation

The Cache Implementation introduces a two-way associative cache to reduce memory access latency. Each cache set includes two ways, with each way having storage for data, tag, and a valid bit. The controller checks if the required data is in the cache (cache hit) or if it needs to retrieve it from main memory (cache miss). An LRU (Least Recently Used) policy manages data replacement.

The cache enhances processor performance by minimizing memory access delays, synchronizing with the main memory only when necessary.

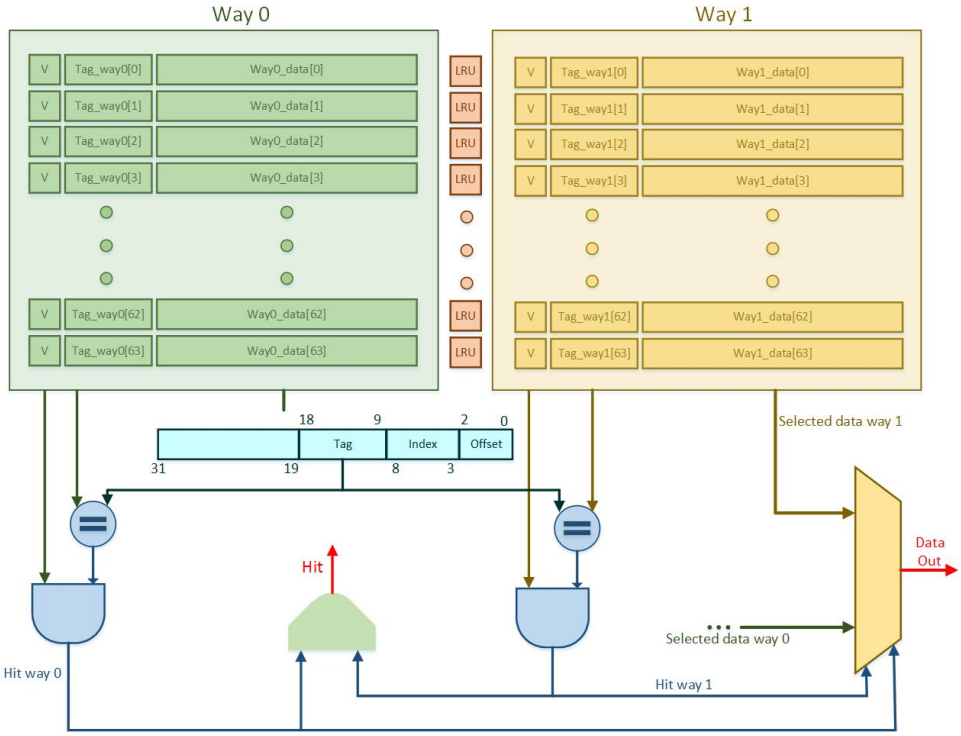

Figure 3: Cache

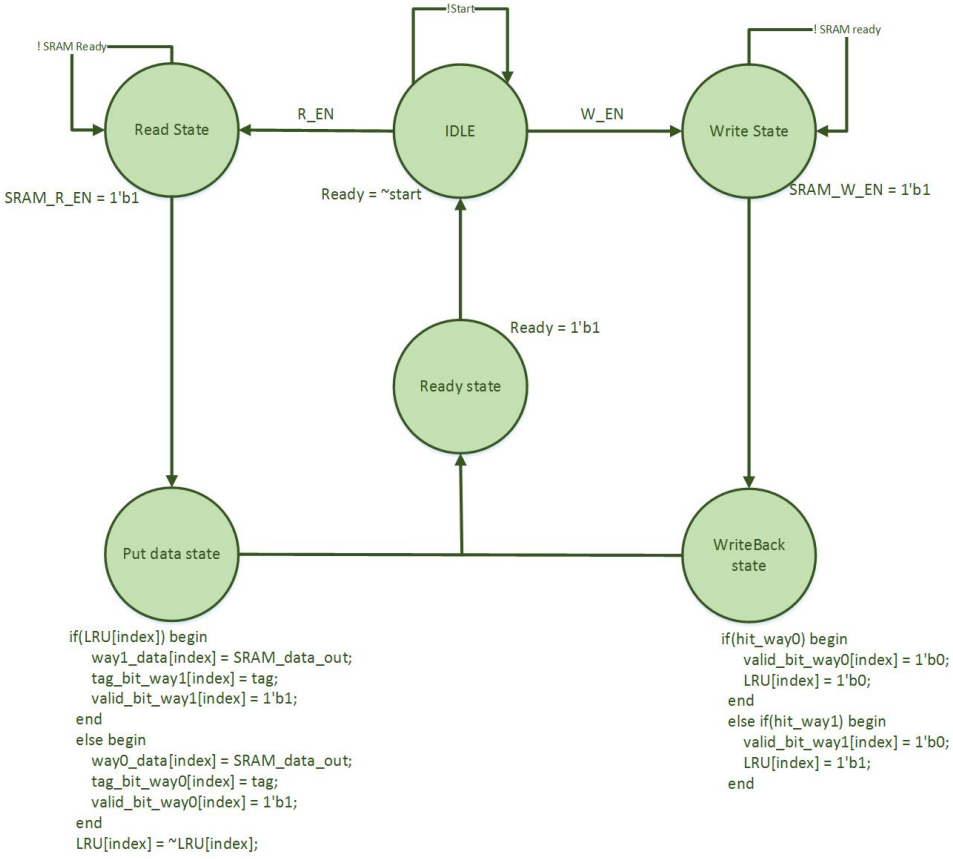

Figure 4: Cache Controller