Milestone 2 Report

Schematic of Proposed Datapath

Provided in a separate file

Implementation

We designed a RISC-V single-cycle processor where we focused on supporting all 42 user-level unprivileged instructions. To manage the branching instructions, we designed a branching module that supports the various branching instructions. Here are the main modules that are being used in our RISCV main module:

NBitRegister (PC): A 32-bit program counter register that stores the current instruction address. It updates every clock cycle unless reset or PC load conditions change.

nBitRCA (PCAdder & AddressAdder): Ripple carry adder used for incrementing the PC by 4 (PCAdder) and computing branch/jump targets (AddressAdder).

InstMem: Instruction memory that outputs the current instruction based on the PC address.

controlUnit (CU): Decodes the instruction and generates control signals for branching, memory access, register writing, ALU operations, and PC selection.

RegFile: Register file that reads two registers (rs1, rs2) and writes to one (rd) depending on the instruction and control signals.

MUX4By1NBit (RegisterDataMux & PCFinalMux):

- RegisterDataMux: Chooses the value to write back to the register file.
- PCFinalMux: Chooses the next value for the program counter based on the control signals (e.g., jump, branch, etc.).

rv32_ImmGen: Immediate generator that extracts and sign-extends the immediate value from the instruction.

NBitShiftLeft: Shifts the immediate value left (typically by 1 or 2 bits) for address calculations.

BranchingUnit: Determines if a conditional branch should be taken based on ALU flags and branch type.

nBitMUX2to1: 2-to-1 MUX used in multiple places to:

- Select between immediate and register operands for the ALU.

- Choose between ALU result or memory data for register write-back.
- Select between branch and sequential address in PC update.

DataMem: Memory module for load and store operations using the computed ALU address.

ALUControlUnit (ALUCU): Translates instruction and ALUop signals into specific ALU function codes (ALUSel).

prv32_ALU: Main Arithmetic Logic Unit performing operations like add, sub, shift, compare, etc., and setting condition flags (cf, zf, vf, sf).

R-Type Instruction

- The control unit decodes the opcode to determine ALU-related control signals, including ALUOp.
- ALUOp, along with funct3 and bit 30 of the instruction, is passed to the ALUControl Unit to select the appropriate ALU operation.
- It utilizes *alufn* to distinguish between different types of shifts (e.g., logical, arithmetic).
- The module processes both the shift amount and the value to be shifted accordingly

I-Type Instruction

- ALUOp is used to select the ALU function.
- The second ALU input is taken from the immediate generator output rather than a second register.
- Use the MemRead signal and funct3 to determine the type of load (e.g., byte, halfword, word).
- Result is written to the destination register after memory access.
- The *jalr* instruction updates the program counter (PC) using jump, pcl, and PCs control signals.
- Instructions like *ebreak* freeze the PC by deactivating pc1.

U-Type Instruction

- The 20-bit immediate is shifted 12 bits then is placed into the destination register.
- The shifted immediate in AUIPC is added to the current *pc,* and the result is written into the destination register using *auipc* control signal.
- Since register data can originate from different sources (memory or u-type instruction), a 2-1 multiplexer is used which has control line of *jump* and *auipc* concatenated.

S-Type Instruction

- Uses the *memWrite* signal along with *funct3* to specify the type of store.
- Ensures proper data and memory handling during store operations.

B-Type Instruction

- Branch instructions depend on flags from the ALU, such as the zf, cf, etc.
- Uses the *branch* signal and *funct3* to determine the specific type of branch for example beq, bne, etc.
- Compines the branch unit with a *jump* signal via an OR gate to decide whether a control transfer occurs.
- A mux chooses between the current *pc+4* or the shifted immediate target for branches and jumps.
- A multiplexer that determines the next *pc* value:
  - From previous mux (jal)
  - From the Alu (jalr)
  - From pc +4 (for normal execution)
  - Or the same pc (ebreak)

J-Type Instruction

- Activates the jump signal
- Stores the return address pc+4 into the destination register.
- Calculates the new pc by adding the instruction's immediate value to the current pc.
- The computed jump address is sent through a multiplexer to select the next program counter value.