EE309

Pipelined IITB RISC CPU

Course Instructor: Prof. Virendra Singh

Tejash Chauhan 22b4231
Saurabh Gupta 22b4227

Team ID: 11

Abstract

This project focuses on the design of a 6-stage pipelined processor, the IITB-RISC-23, tailore specifically for educational purposes within the context of computer architecture and digital systems. The processor is based on a 16-bit in struction set architecture (ISA), featuring 8 general-purpose registers (R0 to R7) where R0 is reserved as the Program Counter.The architecture is also optimized for performance including hazard mitigation techniques.

Contents

1 Introduction

IITB-RISC is a 16-bit very simple computer developed for the teaching that is based on the Little Computer Architecture. The IITB-RISC-23 is an 8-register, 16-bit computer system. It has 8 generalpurpose registers (R0 to R7). Register R0 is always stores Program Counter. All addresses are byte addresses and instructions. Always it fetches two bytes for instruction and data. This architecture uses condition code register which has two flags Carry flag (C) and Zero flag (Z). The IITB-RISC-23 is very simple, but it is general enough to solve complex problems. The architecture allows predicated instruction execution and multiple load and store execution. There are three machine-code instruction formats (R, I, and J type) and a total of 14 instructions.

2 Data-Path Components

2.1 Instruction Fetch

- PC: stored as register 0 in the register file, stores the PC value of the current instruction.
- IM: Instruction memory, takes PC input and gives the instruction as indicated by PC value.
- ALU2: takes 16-bit input, used to update PC to PC+2.

2.2 Instruction Decode

- Generates signals for control signals for multiplexers in the pipeline stages.
- Makes decisions like:
  - Sign-extended value of the immediate data in the instruction if present.
  - Control signal for SE which decides if to sign extend 6-bit or 9-bit immediate data.
  - The ALU control bits which tell the ALU of its operations.
  - Flag control bits which enable or disable the flag registers.
  - Clear bit for when JAL instruction arrives to clear the pipeline register (IF-ID).

2.3 Register Read

- Register File: Contains 8 registers of 16 bits each, which stores data or memory location, etc.
- Hazard_MUX1 (blue): this mux is used for correction of immediate dependencies for operand addresses defined by bits 8-6. Inputs are all data forwarding paths.
- Hazard_MUX2 (blue): this mux is used for correction of immediate dependencies for operand addresses defined by bits 5-3. Inputs are all data forwarding paths.

- LM_SM mux1: chooses between a new op-code and the opcode for LM-SM. This is done because the LM-SM state should be present for 8 cycles.
- LM_SM mux2: chooses the input bits 8-0, the immediate bits for pipeline register 3. It has two inputs, either usual Imm bits or shifted 8 bits. The control line for this mux comes from pipeline register 3 and is implemented using if-else conditions in the code.
- 9-bit Zero extender: selects immediate bits from pipeline register 3. This corrects the immediate dependencies in LLI instruction.

2.4 Execution

- ALU1: performs ADD, NAND, comparator, and complement operations. It also updates the flags such as carry, zero, and overflow flags.
- ALU1_mux_A: takes Ra and Rb values as immediate is fixed to ALU_mux_B input.
- ALU1_mux_B: takes constants, Rb, sign-extended Imm6 bits, and sign-extended Imm9 bits.
- ALU1_out_MUX: used for LM-SM instruction.
- 6-bit SE: extends the imm6 bits of I type instruction.
- 9-bit SE: extends the Imm9 bits of J type instruction.
- 1-bit shifter: used in LM-SM stage to get the bit which is set and the corresponding register in the register file is selected.
- 1_subtractor: used in LM-SM state to decrease additional three bits which we have stored in pipeline register 4. These three bits indicate the rf_a3 value. These three bits will be 111 until the instruction LM-SM is executed.
- Shifter mux: selects if direct 8 bits or shifted 8 bits are to be transferred.

2.5 Memory Write/Read

- Data memory: data memory from which the instruction reads data writes data.
- Mem_di_mux: takes input as data of registers in the register file or the output of ALU1 and sends it to mem_di port.
- Mem_write_mux: takes usual input 1/0 or takes the value of the shifted 8th bit from the shifter.
- ALU5: used for LM-SM instruction, to update the memory address to consecutive memory addresses. This ALU takes input as ALU5's output.
- ALU5_B_mux: control bit is the 8th shifted bit from the shifter.

2.6 Write Back

- ALU3: calculates PC+2*Imm for instructions like BEQ, BLU, BLT, and JAL.
- ALU3_B_mux: selects 6Imm bits or 9Imm bits correspondingly.
- ALU4: Updates PC, PC+2.
- 9-bit Zero extender: Zero extends 9Imm bits for J type instructions.
- 6-bit Sign Extender: Sign-extends 6Imm bits for I type instructions.
- 9-bit Sign Extender: Sign-extends 9Imm bits for J type instructions.
- Rf_a3_mux: has a separate input for LM-SM states.
- Rf_d3_mux: selects values to be entered in the register.
- Pc_in: 4 select lines 2 are PC+2 and two from the write back stage.
- Rf_write_mux: have a separate select line for LM-SM state.

3 HAZARD DETECTION AND MITIGATION

- **R0 HAZARD:**
  - As per our understanding of the problem statement, we are considering that the programmer is highly skilled and given the specifications of our design, he/she won't run instructions which are 'R0 hazard' specific. i.e., none of the instructions will have the destination address as "000", which is the address for PC.
  - The above mentioned hazard is very frequent for such a design in which the PC is in the register file itself.
- **Dependencies without R0:**
  - If the operand register for new instructions is the same as that of the destination register of the previous instruction, let it be in the execution stage or memory stage or writeback stage, there will be errors and we can use forwarding directly from those stages. One thing to be noted is that in case of conditional execution like ADC, ADZ, NDC, NDZ if the respective flags are not set then there is no need for forwarding even if the destination register of previous instructions are the same.
    - For all these forwarding we are using two muxes and the output is to be given to pipeline reg3 ports data47_32 and data63_48.
    - For ADD, ADC, ADZ, NDU, NDC, NDZ, ADI and other R-type instructions we are forwarding from ALU_1 and thus correcting IMMEDIATE dependencies.
    - For ADD, ADC, ADZ, NDU, NDC, NDZ, ADI and other R-type instructions we are also forwarding from 79-64 bits of pipeline register 4 and thus correcting LEVEL 2 dependencies.

    - For LW we are forwarding from the mem_d0 LEVEL 2 dependencies.
    - For JAL and JLR instructions, forwarding is the same as other R-type instructions.
  - We can't correct LW immediate dependencies and so we have to stall the pipeline for 1 cycle so that load reaches memory stage and add or nand is in register-read stage and then we do the forwarding from mem_do.

- **LM-SM hazards:**
  - In case of LM-SM instruction we have to stall pipeline register1, pipeline register 2 and stop updating the PC.
  - **LM:**
    - Subtractor is turned ON when LM is in Pipe4.
    - Wrp1, wrp2 (write enable for pipeline registers) takes input 0 when LM is in pipe3.
    - PC updation stops when LM is in pipe3.
    - Simultaneously transfer Shifter output into pipe4 when LM is in pipe3.
    - When 8 cycles complete:
      - We are checking the above condition by checking sig_pipe4[84-82] != "000".
    - After 8 cycles PC updation starts.
    - Subtractor is turned off, and bits 84-82 of pipeline register 3 are selected as "111".
    - wrp¡=1, wrp2¡=1.
    - All muxes related to LM-SM, start passing normal signals for other instructions.
  - **SM:**
    - Subtractor is turned ON when SM is in pipe2.
    - Wrp1, wrp2 (write enable for pipeline registers) are set 0 when SM is in pipe3.
    - PC updation stops when SM is in pipe3.
    - Simultaneously transfer Shifter output into pipe3 when SM is in pipe2.
    - When 8 cycle completes:
      - We are checking the above condition by checking, sig_pipe5[100-98] != "000".
    - After 8 cycles PC update starts.
    - Subtractor is turned OFF.
    - wrp¡=1, wrp2¡=1 (enable write enable for pipeline registers).
    - All muxes related to LM-SM start passing normal signals for other instructions.

4 RTL simulation

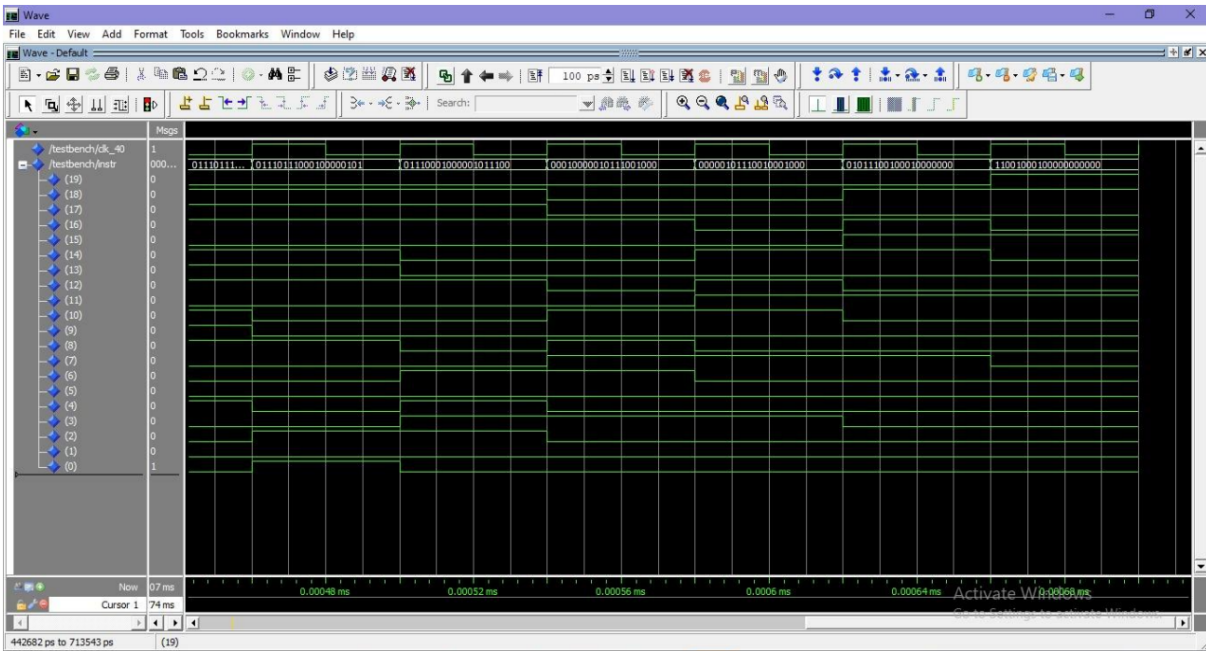

Figure 1: RTL Simulation

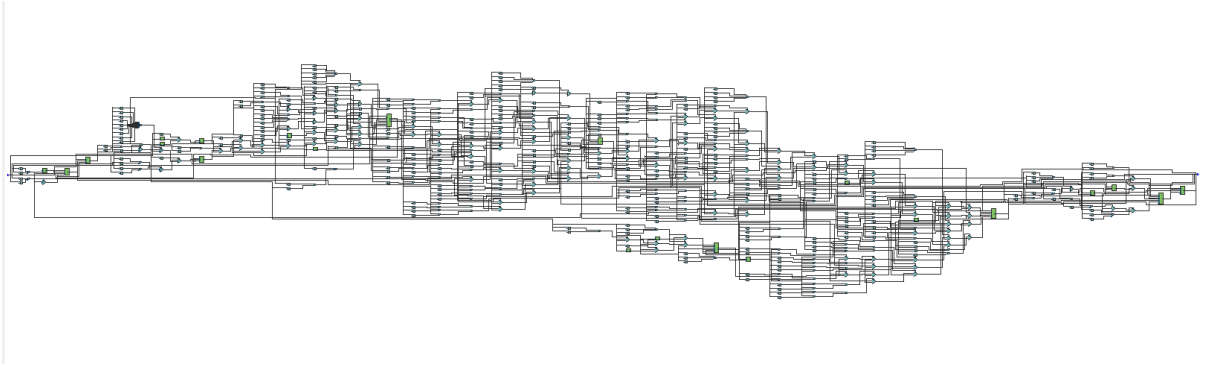

Figure 2: RTL Viewer

5 Work Distribution

- Tejash Chauhan - Hazard Detection, Datapath, Control Unit Design, Report
- Saurabh Gupta - ALU Designing, Hazard Detection, Register File, Report