BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering

2024/2025

Fall Semester

ENCS4370| Computer Architecture

Project No. 2

Design and Verification of a Simple Pipelined RISC Processor in Verilog

Deadline: January 3, 2025 at 23:59

---

1. Objectives:

- To design the datapath and control path of a simple pipelined 16-bit RISC processor in Verilog
- To implement and verify a working processor using simulation

2. Processor Specifications

1. The instruction size and the words size is 16 bits
2. Eight 16-bit general-purpose integer registers
3. 16-bit PC (Program Counter)
4. 16-bit RR (Return Register) to store the return address during function calls.
5. The processor has two separate physical memories, one dedicated to instructions and the other to data.
6. Three instruction types
   - R-type: Register Type
   - I-type: Immediate Type
   - J-type: Jump Type
7. Word-addressable memory

8. The ALU generates a **zero** signal to indicate if the result of the last operation is zero or not
9. **Performance Registers:** This processor includes several performance-monitoring registers to track various program execution metrics, such as:
    - Total number of executed instructions
    - Total number of load instructions
    - Total number of store instructions
    - Total number of ALU instructions
    - Total number of control instructions
    - Total number of clock cycles
    - Total number of stall cycles in case of a pipelined implementation

3. Instruction Types and Formats

As mentioned above, this ISA has three instruction types. These instruction types have the following formats:

R-Type (Register Type)

Opcode[4]	Rd[3]	Rs[3]	Rt[3]	Function[3]

- **4-bit opcode:** opcode.
- The opcode is zero for all R-Type instructions.
- **3-bit Rd:** destination register
- **3-bit Rs:** first source register
- **3-bit Rt:** second source register
- **3-bit:** Function. This field determines the specific operation of the instruction.

I-Type (Immediate Type)

Opcode[4]	Rs[3]	Rt[3]	Signed Imm[6]

- **4-bit opcode:** opcode.
- **3-bit Rs:** first source register
- **3-bit Rt:** destination register
- The immediate value is zero-extended for logical instructions and sign-extended for all other instructions.
- In the case of BEQ and BNE instructions, which are I-type instructions, the branch target is calculated as follows:
  Branch target = Current PC + sign extended immediate

J-Type (Jump Type)

Opcode[4]	9-bit offset	Function[3]

- **4-bit opcode:** opcode.
- The opcode is 1 for all J-Type instructions.
- **3-bit:** Function. This field determines the specific operation of the instruction.
- The jump target address is calculated by concatenating these two fields: **PC[15:9], 9-bit offset**

4. Instructions' Encoding

For simplicity, you are required to implement only a subset of this processor's ISA. The table below presents the various instructions to be implemented, along with their type, opcode value, function value, and corresponding meaning in Register Transfer Notation (RTN).

Instruction	Meaning	Opcode Value	Function Value
R-Type Instructions			
AND Rd, Rs, Rt	Reg(Rd) = Reg(Rs) & Reg(Rt)	0000	000
ADD Rd, Rs, Rt	Reg(Rd) = Reg(Rs) + Reg(Rt)	0000	001
SUB Rd, Rs, Rt	Reg(Rd) = Reg(Rs) - Reg(Rt)	0000	010
SLL Rd, Rs, Rt	Reg(Rd) = Reg(Rs) << Reg(Rt)	0000	011
SRL Rd, Rs, Rt	Reg(Rd) = Reg(Rs) >> Reg(Rt)	0000	100
I-Type Instructions			
ANDI Rt, Rs, Imm	Reg(Rt) = Reg(Rs) & Imm	0010	NA
ADDI Rt, Rs, Imm	Reg(Rt) = Reg(Rs) + Imm	0011	NA
LW Rt, Imm(Rs)	Reg(Rt) = Mem(Reg(Rs) + Imm)	0100	NA
SW Rt, Imm(Rs)	Mem(Reg(Rs) + Imm) = Reg(Rt)	0101	NA
BEQ Rs, Rt, Imm	if (Reg(Rs) == Reg(Rt)) Next PC = branch target else Next PC = PC + 1	0110	NA
BNE Rs, Rt, Imm	if (Reg(Rs) != Reg(Rt)) Next PC = branch target else Next PC = PC + 1	0111	NA
FOR Rs, Rt	• **Rs** stores the loop target address, i.e., the address of the first instruction in the loop block • **Rt** stores the initial number of the loop iterations, i.e., the initial value of the loop counter • The **Rt** register is decremented at the end of each iteration. The loop exits when the content of the **Rt** register becomes zero • The immediate field is ignored in this instruction	1000	NA
J-Type Instructions			
JMP Offset	Next PC = jump target	0001	000
CALL Offset	Next PC = jump target PC + 1 is stored on the RR	0001	001
RET	Next PC = value of the RR The 9-bit field is ignored in this instruction	0001	010

5. RTL Design

You are required to design a **5-stage pipelined** processor (fetch, decode, ALU, memory access, and write back). The design should include a datapath and control path that support all of the aforementioned instructions.

6. Verification

To verify the RTL design, write a testbench around it. Moreover, you need to write multiple code sequences (small binary programs) in the given ISA and show how the processor you designed executes these code sequences.

7. Project Report

The report document must contain sections highlighting the following:

1 – Design and Implementation

1. Detailed description of the datapath, its components, and the assembly of these components.
2. A complete description of the control signals, description, truth table, state diagrams, and Boolean equations, etc.
3. The implementation details, and the design choices you made with justification
4. Datapath and control path block diagrams.
5. A list of sources for any parts of your design that are not entirely yours (if any).
6. Carry out the design and implementation with the following aspects in mind:
   - Correctness of the individual components
   - Correctness of the overall design when connecting these components together
   - Completeness: all instructions were implemented properly.

2 – Simulation and Testing

1. Carry out the simulation of the processor developed
2. Describe the test programs that you used to test your design with enough comments describing the program, its inputs, and its expected output. List all the instructions that were tested and work correctly. List all the instructions that do not run properly.
3. Also, provide snapshots of the simulator's window with your test program loaded and showing the simulation output results.

8. Teamwork

1. Work in groups of up to three students. Groups of more than three students are not allowed.
2. Group members are required to coordinate the work equally among themselves so that everyone is involved in all the following activities:
   - Design and implementation
   - Simulation and testing
   - Report writing
   - Project demonstration and discussion
3. Clearly show the work done by each group member.
4. The team members can be from different sections.

9. Submission Guidelines

Please attach a single ZIP file containing all the design circuits, source code, binary instruction files used for testing your design, the associated test data, and the report document.

10. Grading Criteria

Note: each of the following items is a prerequisite to the next one, e.g., we cannot consider the RTL code if there is no design, and we cannot consider the verification part if there is no RTL code.

Item	**Weight**
Control signals generation (truth tables and Boolean equations)	10
Processor Microarchitecture Design (complete datapath and control path) that supports all instructions	20
Complete **modular** RTL design of the above microarchitecture that supports all instructions	20
Verification environment (testbench) around the RTL design such that you can write code sequences in the ISA, store them in the processor's instruction memory, execute them and show results using waveform diagrams.	20
Code organization and documentation	5
Detailed report that includes control signals truth tables, Boolean equations, detailed and clear microarchitecture design schematics, test cases, simulation screenshots, etc.	15
Discussion (answering questions, the way of answering questions, etc.)	10
Total	**100**