HW #2: Instruction Decoder

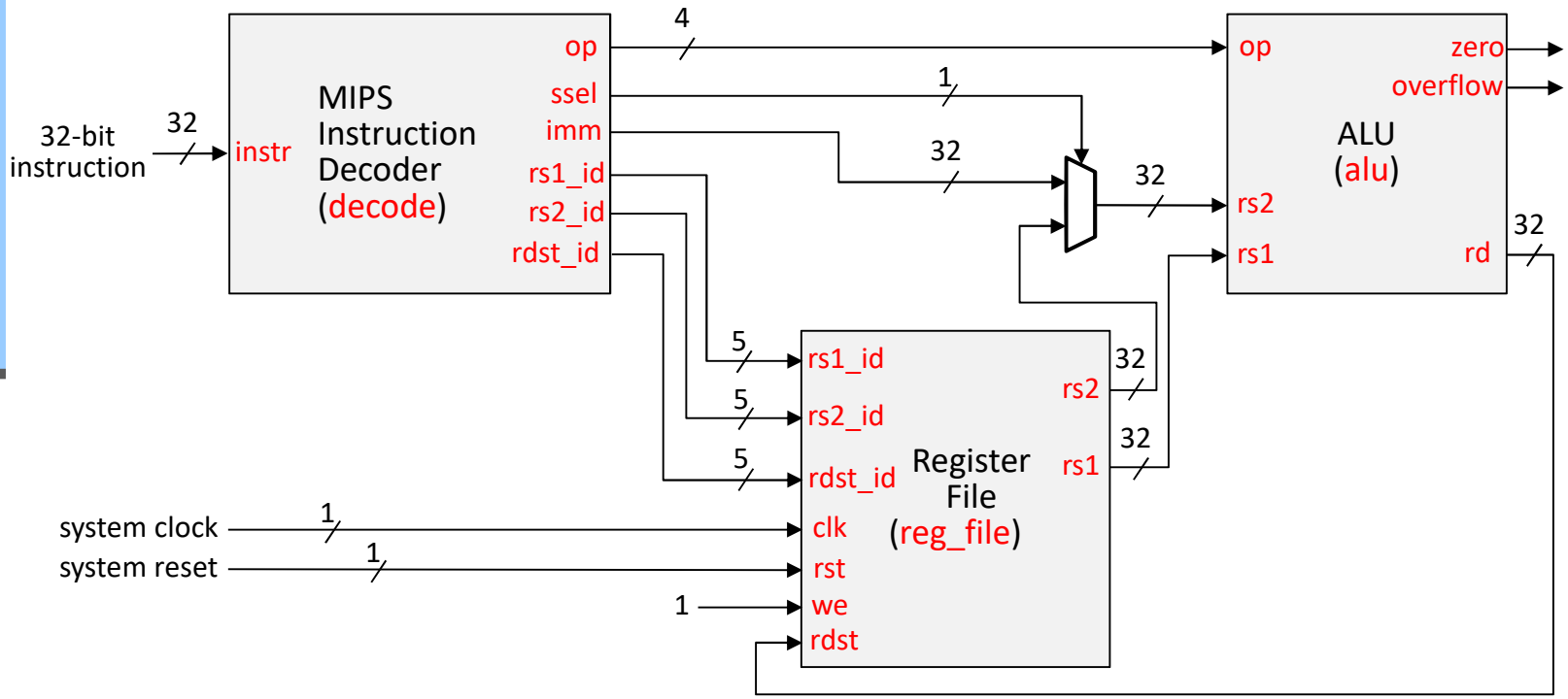
Chun-Jen Tsai
NYCU
03/17/2023

HW 2: Instruction Decoder

- Goal: design a MIPS instruction decoder
  - The input to the decoder is a 32-bit MIPS instruction
  - The output to the decoder should be connected to the register file and the ALU for computation
- The deadline of the lab is on 4/03, by 5:00pm.

The Top-Level Block Diagram

- The block diagram of the system is as follows:
  - Note that your declarations must use the module & port names in red precisely; the bit-width must be the same as well
  - The immediate value (imm) must be sign-extended to 32-bit



The Instruction Decoder

- Your instruction decoder must decode the following instructions
  - The complete MIPS instructions are shown in the file `MIPS_ISA_Sheet.pdf`

Assembly instruction	Function	Format	opcode	funct
`add rd, rs, rt`	addition: rd ← rs + rt	R	0x00	0x20
`addi rt, rs, imm`	add immediate: rt ← rs + imm	I	0x08	0x00
`sub rd, rs, rt`	subtraction: rd ← rs - rt	R	0x00	0x22
`and rd, rs, rt`	rd ← rs and rt	R	0x00	0x24
`or rd, rs, rt`	rd ← rs or rt	R	0x00	0x25
`nor rd, rs, rt`	rd ← ~(rs or rt)	R	0x00	0x27
`slt rd, rs, rt`	rd ← (rs < rt)? 32'h1 : 32'h0;	R	0x00	0x2a
`slti rt, rs, imm`	rt ← (rs < imm)? 32'h1 : 32'h0;	I	0x0a	0x00

A Design Flaw in the MIPS ISA

- Note that for MIPS, there is a design flaw in the instruction format:
  - The name for the destination register is not consistent: for different instructions, it can be either `rd` or `rt`!
  - This design flaw has been corrected in the RISC-V ISA, where `rd` is used universally as the target register
- Therefor, the output port `rdst_id` of the `decode` module may be the ID for `rd` or `rt`, depending on the instructions

Port Definitions of `decode` Module

- The `decode` module is a combinational circuit with the I/O ports defined as follows:

```
module decode #(parameter DWIDTH = 32)
(
  input  [DWIDTH-1:0] instr, // Input instruction.

  output [3 : 0]      op,   // Operation code for the ALU.
  output              ssel, // Select the signal for either
                            //   the immediate value or rs2.

  output [DWIDTH-1:0] imm,     // The immediate value (if used).
  output [4 : 0]      rs1_id,  // register ID for rs.
  output [4 : 0]      rs2_id,  // register ID for rt (if used).
  output [4 : 0]      rdst_id, // register ID for rd or rt (if used).
);
```

Declaration of `reg_file` Module

- The register file module contains 32 32-bit registers:

```
module reg_file #(parameter DWIDTH = 32)
(
    input                   clk, // system clock
    input                   rst, // system reset

    input  [4 : 0]          rs1_id, // register ID of data #1
    input  [4 : 0]          rs2_id, // register ID of data #2 (if any)

    input                   we,      // if (we) R[rdst_id] ← rdst
    input  [4 : 0]          rdst_id, // destination register ID
    input  [DWIDTH-1 : 0] rdst,    // input to destination register

    output [DWIDTH-1 : 0] rs1, // register operand #1
    output [DWIDTH-1 : 0] rs2  // register operand #2 (if any)
);

reg [DWIDTH-1:0] R[0:31];

        . . . (You must finish the rest of the code) . . .
```

The Register File Behavior

- The register file is a sequential module that updates the register array at positive edges of the clock signal
  - Register 0 is a read-only register of the value 32'h0
  - All the registers shall be initialized to all zeros upon reset
- It has two read ports and one write port (`rdst`)
  - Read ports (`rs1`, `rs2`) are combinational output of the data
  - Write port stores the data into one of the registers when the write enable (`we`) signal is 1 at every positive clock edge
    - For this HW, `we` is always connected to 1 since we assume one instruction will be executed at every clock cycle

HW 2 Guidelines

- You should upload all your code to E3 before the deadline
- The sample package, `HW2_sample_tb.tgz`, contains sample testbench files and the templates of `decode.v` and `reg_file.v`. Please read `readme.txt` in that package carefully!
- For grading, TAs will use a more thorough testbench