Homework #4

B05902120 / Yu-Ting, TSENG

Nov 14, 2018

Basic Execution

- Platform: linux (CSIE workstation)

1. Some of the files have been completed by the TAs, and the following is those what I have finished.
2. The first thing is to construct the basic connection (`input` and `output`) between the units in `CPU.v`. The concept is to connect the units by the wire.
3. In some case, we may need some signal to tell what to do. Hence, write the content of `Control.v`, we might find that there are only two types of instructions, which is R-type and I-type; and the most significant difference would be `ALUSrc_o`.

```
assign  temp = (Op_i==7'b0110011) ? {4'b1001}:
               (Op_i==7'b0010011) ? {4'b1111}:
               // in order is R-type, I-type
```

4. In order to execute the next instruction, write the content of `Adder.v` to calculate the address of the next program counter.

```
assign  data_o = data1_i + data2_i;
```

5. Extend the immediate value to 32 bits from 12 bits, since it will be easier for `ALU` to process in regular format, in other words, it is better that all the input of the `ALU` is 32 bits.

```
assign  data_o ={ {20{data_i[11]}}, data_i};
```

6. `MUX32` is ta selector controlled by the control signal, decide which data is the source of the `ALU`.

```
assign  data_o = (select_i==0) ? data1_i:
                 (select_i==1) ? data2_i:
                 // in order choose reg or imm data
```

7. Besides `MUX32`, what we also need is `ALU_Control`. We do addition, subtraction and lots of thing in the same unit `ALU`, so we need to tell the unit which kinds of operation is going to be done.

```
assign  temp = (funct_i==10'b0000000110) ? {3'b000}:
               (funct_i==10'b0000000111) ? {3'b001}:
               (funct_i==10'b0000000000) ? {3'b010}:
               (funct_i==10'b0100000000) ? {3'b011}:
               (funct_i==10'b0000001000) ? {3'b100}:
               // in order do or, and, add, sub, mul
```

8. The most important unit in this pipeline cycle is `ALU`. `ALU` can do plenty of things, such as calculating the address for the instruction load and store, or calculate the result for arithmetic instruction and choose the final consequence.

```
assign  cal_or = data1_i | data2_i;
assign  cal_and = data1_i & data2_i;
assign  cal_add = data1_i + data2_i;
assign  cal_sub = data1_i - data2_i;
assign  cal_mul = data1_i * data2_i;
                  // in order do operations
```