107-2 Digital System Design Homework 2

TA Information
劉力仁
b04901068@ntu.edu.tw

1. 8-bit arithmetic logic unit (ALU) (40%)

In problem 1, you are asked to model an 8-bit arithmetic logic unit (ALU). The input and output ports are defined in Figure 1. The functions of ALU are defined in Table 1.

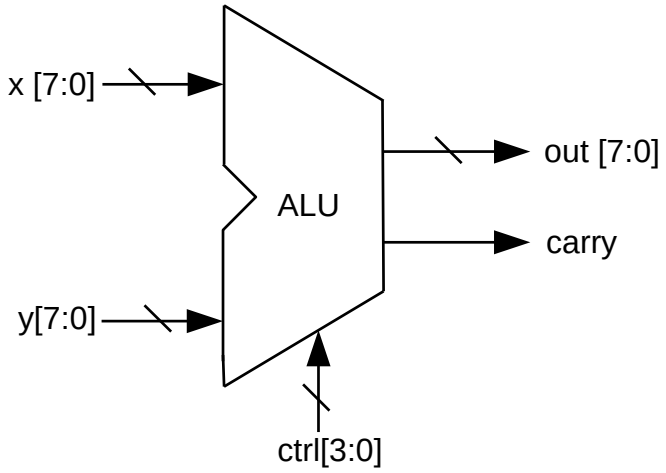

Figure 1

Table 1

Control Signal(ctrl)	Description	Function
0000	Add(signed)	out = x + y, Carry: 進位
0001	Sub(signed)	out = x - y , Carry: 進位
0010	And	out = and(x, y)
0011	Or	out = or(x , y)
0100	Not	out = not(x)
0101	Xor	out = xor(x , y)
0110	Nor	out = nor(x , y)
0111	Shift left logical variable	out = y << x[2:0]
1000	Shift right logical variable	out = y >> x[2:0]
1001	Shift right arithmetic	out ={x[7],x[7:1]}
1010	Rotate left	out = {x[6:0] , x[7]}
1011	Rotate right	out = {x[0] , x[7:1]}
1100	Equal	out = (x==y)?1:0

1101	NOP(No operation)	out = 0
1110	NOP	out = 0
1111	NOP	out = 0

P.S.只有加減法需要考慮Carry，其餘運算Carry可以為任何值

(1) (10%)

Use Verilog to implement the **RT**-level (use *continuous assignment*, ***assign***) model of the 8-bit ALU. Modify the "alu_assign.v" file, which contains the module name and input/output ports. Use the given test bench, "alu_assign_tb.v" to verify you design. To simulate, use the following command

```
ncverilog alu_assign_tb.v alu_assign.v +access+r
```

(2) (10%)

Use Verilog to implement the **RT**–level (use *event-driven construct*, ***always block***) model of the 8-bit ALU. The input and output ports are the same as previous one. Modify the "alu_always.v" file, and use the given test bench, "alu_always_tb.v" to verify you design. To simulate, use the following command

```
ncverilog alu_always_tb.v alu_always.v +access+r
```

(3) (20%)

The given two testbenches "alu_assign_tb.v" "alu_always_tb.v" don't check the all required functions of the ALU. You need to modify the two given testbenches and rename the testbenches to "alu_assign_tb2.v" and "alu_always_tb2.v" (They are same but for different modules!). Use your modified testbenches to verify if all functions of your design are correct. Show the waveform results. Describe how you verify the correctness in your report. If how you execute your testbench is different from 1-(1) and 1-(2), please provide a README so that TA can correctly test your design.

2. 8x8 Register File (40%)

A register file consists of a set of registers which can be read or written. There are 8 registers in this register file, and the width of the register is 8-bits. The input and output ports are described in Figure 2.

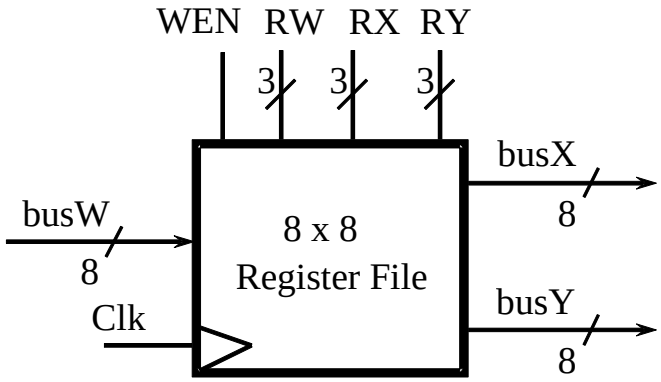

Figure 2

You must follow these specifications:

A. I/O Port Functionality

(1)busW: 8 bit input data bus

(2)busX、busY: 8 bit output data buses

(3)WEN: active high write enable (WEN==1)

(4)RW:select one among 8 registers to be written

(5)RX: select one among 8 registers to be read, output on busX

(6)RY: select one among 8 registers to be read, output on busY

B. Register File

(1) 8 registers.

(2) $r0~$r7

(3) $r0=zero (constant zero, don't care any write operation)

C. Write Operation

(1)BusX and busY could be an arbitrary 8-bit vector during write operation

(2)The data on busW will be written into a specified register synchronously on positive edge of **Clk**

(3)RW is the index of register to be written.

D. Read Operation

(1) The register file behaves as a combinational logic block when reading

(2) Read the data in the register file **synchronously**

(1) (20%)

Implement the 8 8-bit register file in Verilog. Modify “register_file.v”, which contains the module name and input/output ports.

(2) (20%)

Write the testbench (“register_file_tb.v”) to verify your design. Run a simulation by using the following command:

```
ncverilog register_file_tb.v register_file.v +access+r
```

You must consider the following requirement:

A. You need to write a testbench and generate random test patterns to verify your register file. Don’t read a test vector from data file

B. Test patterns need to toggle every bit in the register bank.
Hint:1111_1111 -> 0000_0000 ->1111_1111

C. Your testbench should verify the functionally of the register file automatically.

3. Simple Calculator (20%)

Combine the previous two designs into a simple calculator unit. You can use this unit to execute some simple programs. The input and output ports are defined in Figure 3. And there is a control signal "Sel" to select which data is input to ALU.

(1) When Sel = 0, DataIn is passed to port x of ALU.

(2) When Sel = 1, the data loaded to port x of ALU is from register file.

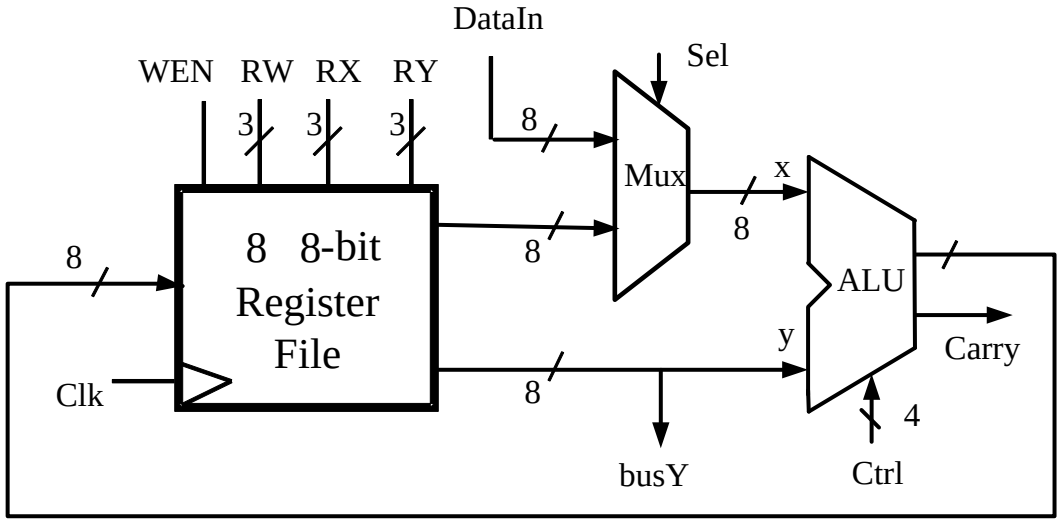

Figure 3

You must define the following ports in your design:

A. Input Port
   (1) Clk
   (2) WEN
   (3) RW
   (4) RX
   (5) RY
   (6) DataIn
   (7) Sel
   (8) Ctrl

B. Output Port
   (1)busY
   (2)Carry

(1) (20%)

Use the previous two modules to implement the simple calculator unit ("simple_calculator.v"). Use the always block to describe the multiplexer. After finishing the calculator, use testbench ("simple_calculator_tb.v") to test your

design correctness. Run the simulation by using the following command:

```
ncverilog simple_calculator_tb.v simple_calculator.v +access+r
```

4. Describe what you found (bonus 5%)

Write down what you found. Feel free to share your experience. (Ex: some mistakes you spend a lot of time, your environment, naming method) Anything you feel is special. The points depend on your answers!

Submission requirement:

1. All the files need to be compressed as a single **ZIP file** and **uploaded to Ceiba**.

   Example of filename

   DSD_HW2_b04901068.zip

   Your submitted file should include the following files:

   DSD_HW2_b04901068/
   1-ALU/ 1_assign/ alu_assign.v
   1-ALU/ 1_assign/ alu_assign_tb2.v
   1-ALU/ 2_always/ alu_always.v
   1-ALU/ 2_always / alu_always_tb2.v
   2-RegisterFile/ register_file.v
   2-RegisterFile/ register_file_tb.v
   3-SimpleCalculator/ simple_calculator.v
   Report_ HW2_b04901068.pdf

1. **Deadline: 2019/04/03 24:00**
2. The homework will be graded ONLY IF the filename of your submission are correct!