BLG222E Computer Organization

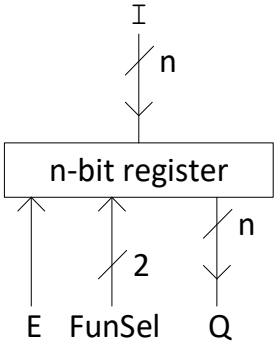
Project 1

Due Date: 13.03.2019 20:00

In this project, registers and register files will be designed and implemented. It has 3 parts. Design each part as a library unit, so that it can be reused in other parts and next projects. **You can use any available combinational/sequential logisim units in your projects.**

(Part-1) Design 2 different types of registers: **(1)** 8-bit register and **(2)** 16-bit register. These registers have 4 functionalities that are controlled by 2-bit control signals (**FunSel**) and an enable input (**E**).

The graphic symbol of the registers and the characteristic table is shown in Figure 1. Symbol ϕ means don't care. Build these registers as a library in logisim software to use them in **Part-2**.



E	FunSel	Q^+	
0	ϕ	Q	(Retain Value)
1	00	I	(Load)
1	01	Q-1	(Decrement)
1	10	Q+1	(Increment)
1	11	0	(Clear)

Figure 1: Graphic symbol of the registers (Left) and the characteristic table (Right)

(Part-2) Design a register file (a structure that contains many registers) that works as follows.

(Part-2a) Design the system shown in Figure 2 which consists of four 8-bit general purpose registers: **R0**, **R1**, **R2**, and **R3**. The details of inputs and outputs are as follows.

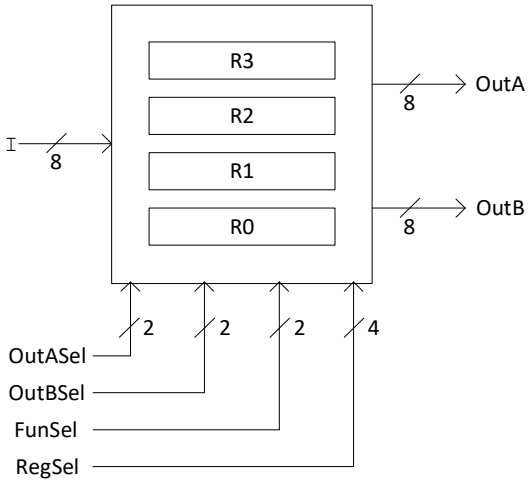


Figure 2: 8-bit general purpose registers, inputs, and outputs

OutASel and **OutBSel** are used to feed output lines **OutA** and **OutB**, respectively. 8 bits of the selected registers are output to **OutA** and **OutB**. Figure 3 shows selection of output registers based on the **OutASel** and **OutBSel** control inputs.

OutASel	Output A
00	R3
01	R2
10	R1
11	R0

OutBSel	Output B
00	R3
01	R2
10	R1
11	R0

Figure 3: OutASel and OutBSel controls

RegSel (*Figure 4*) is a 4-bit signal that selects the registers to apply the function that is determined by **FunSel** (*Figure 5*) signal.

RegSel	Enabled Registers
0000	N0 register is enabled, All registers retain their values
0001	Only R0 is enabled, Function selected by FunSel will be applied to R0
0010	Only R1 is enabled, Function selected by FunSel will be applied to R1
0011	R0 and R1 are enabled, Function selected by FunSel will be applied to R0 and R1
0100	Only R2 is enabled, Function selected by FunSel will be applied to R2
0101	R0 and R2 are enabled, Function selected by FunSel will be applied to R0 and R2
0110	R1 and R2 are enabled, Function selected by FunSel will be applied to R1 and R2
0111	R0, R1 and R2 are enabled, Function selected by FunSel will be applied to R0, R1 and R2
1000	Only R3 is enabled, Function selected by FunSel will be applied to R3
1001	R0 and R3 are enabled, Function selected by FunSel will be applied to R0 and R3
1010	R1 and R3 are enabled, Function selected by FunSel will be applied to R1 and R3
1011	R0, R1 and R3 are enabled, Function selected by FunSel will be applied to R0, R1 and R3
1100	R2 and R3 are enabled, Function selected by FunSel will be applied to R2 and R3
1101	R0, R2 and R3 are enabled, Function selected by FunSel will be applied to R0, R2 and R3
1110	R1, R2 and R3 are enabled, Function selected by FunSel will be applied to R1, R2 and R3
1111	R0, R1, R2 and R3 are enabled, Function selected by FunSel will be applied to R0, R1, R2 and R3

Figure 4:RegSel Control Input

FunSel	R_x^+	
00	I	(Load)
01	R_x-1	(Decrement)
10	R_x+1	(Increment)
11	0	(Clear)

Figure 5: FunSel Control Input

For example: If **RegSel** is 1101 and **FunSel** is 10, then the registers **R0**, **R2**, and **R3** will be incremented with next clock cycle. **R1** will not be affected since it is not enabled by **RegSel**.

(Part-2b) Design the system shown in Figure 6 which consists of three 8-bit address registers: **AR**, **SP**, and **PC**. FunSel and Regsel works as in **Part-2a**.

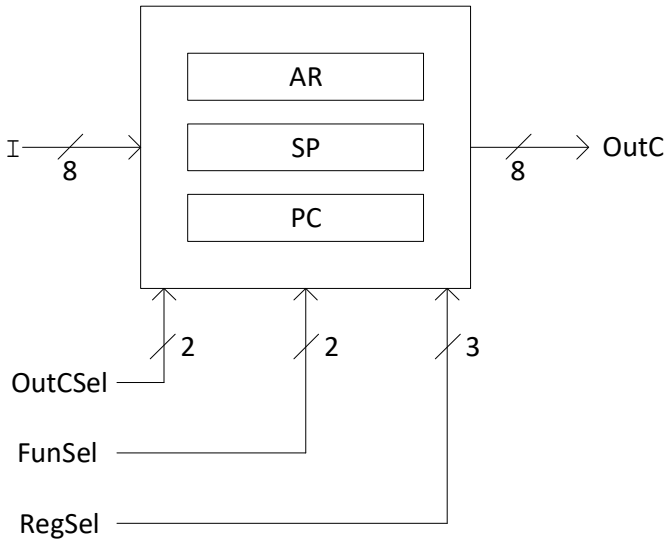


Figure 6: 8-bit address registers, inputs, and outputs

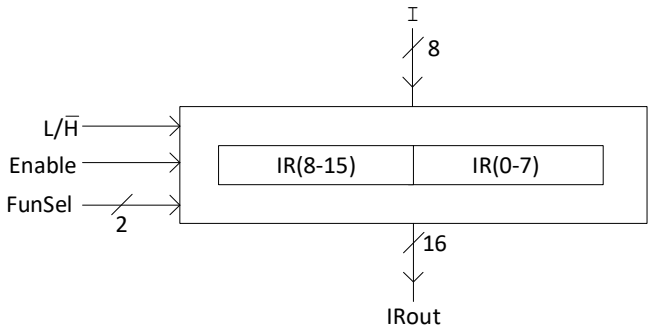
OutCSel selects the register whose contents will be given to the output **OutC** (shown as **Output C** in Figure 7.)

OutCSel	Output C
00	AR
01	SP
10	PC
11	PC

Figure 7: OutCSel control

(Part-2c) Design an 16-bit **IR** register whose graphic symbol and characteristic table are given in Figure 8.

This register can store 16 bit binary numbers. However, the input of this register file is only 8 bits. Hence, using the 8 bit input you can load either the lower (bits 7-0) or higher (bits 15-8) half. This is determined by $\boldsymbol{L/\overline{H}}$ signal.



L/$\overline{H}$	Enable	FunSel	IR$^+$
ϕ	0	ϕϕ	IR
1	1	00	IR(0-7) <- I
0	1	00	IR(8-15) <- I
ϕ	1	01	IR - 1
ϕ	1	10	IR + 1
ϕ	1	11	0

Figure 8: Graphic symbol of the IR register (Left) and its characteristic table (Right)

Submission:

Implement your design in logisim software, upload a single compressed (zip) file to Ninova before the deadline. Only one student from each group should submit the project file. This compressed file should contain your design files (.circ) and a report that contains:

- the number&names of the students in the group
- list of control inputs and corresponding functions of the ALU

Group work is expected for this project. Form groups of 5-6 students, and design together. Make sure to connect pins (under Wiring group of logisim) to the inputs and control inputs of your design, so that different inputs and functions can be tested. Similarly connect your inputs and outputs to a “Hex Digit Display" in logisim (under Input/output group of logisim) so that the test outputs can be observed and use proper labelling to improve the clarity of your circuits.

Late turnin policy:

If you cannot turn in your project through Ninova in time, please do not send any emails as they will be ignored. For late turnins -20%/day penalty will be applied. Please remember, you need to turn this project in and get at least 30% (3 out of 10) to avoid VF (see syllabus at Ninova).