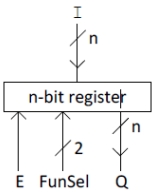
Project 1
BLG222E-Computer Organization

Name: Oben Özgür Student Number: 150190719
Name: Yasin Abdülkadir Yokuş Student Number: 150190739
Name: Ramazan Yetişmiş Student Number: 150190708
Name: Faruk Orak Student Number: 150180058

1 Part 1

In this part of the project, we have designed 8-bit and 16-bit registers that is controlled by 2-bit input function selector.



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

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

2 Part 2

In this part of the project, we designed circuits which consist of many registers and capable of performing different operations on selected registers. These circuits are also capable of displaying the outputs of chosen registers.

2.1 Part 2a

In this part, we have implemented a circuit who has I, OutASel, OutBSel, FunSel, RegSel inputs and OutA, OutB outputs by using four registers that we designed. In the following sections, we will explain these inputs, outputs and how we designed them.

2.1.1 I input (8-bits)

I input is neccessary to send integer which will be loaded to 8-bits registers. I input is connected to related pin of registers directly.

2.1.2 FunSel (2-bits)

FunSel input is used to select the operation which will be performed in the 8-bits registers. FunSel input is connected to register's FunSel pin directly. The coding of the operations is given below.

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

2.1.3 RegSel (4-bits)

Regsel input is used to decide which register or registers will be affected by operation which is selected by funsel input. We connected the RegSel's bits to register's Enable inputs like that:
0th bit → R0
1st bit → R1
2nd bit → R2
3nd bit → R3
The RegSel input's operation table is given below.

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

2.1.4 OutASel, OutBSel inputs (2-bits) and OutA, OutB (8-bits) outputs

OutASel and OutBsel inputs select a register and OutA and OutB outputs shows the content of that register. By using a 4:1 Mux, we have designed that operation easily. The table of OutASel, OutBSel inputs and OutA, OutB outputs is given below.

OutASel	Output A
00	R0
01	R1
10	R2
11	R3

OutBSel	Output B
00	R0
01	R1
10	R2
11	R3

2.2 Part 2b

This system has three 8-Bit Address Registers PC, AR and SP as main components. 8-Bit input is connected those three Registers as input.
We used REGSEL input to choose, on which registers the functions we select with FunSel input is conducted. We also used OutCSel and OutDSel and connected them to 4 to 1, 8-Bit multiplexer as selection inputs.
RegSel and FunSel control inputs:

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

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

Those two 2-Bit selection inputs decide which 8-Bit data output will be displayed on two different display outputs which are OUTPUT C and OUTPUT D.
OUTPUT C and OUTPUT D controls:

OutCSel	Output C
00	PC
01	AR
10	SP
11	PC

OutDSel	Output D
00	PC
01	AR
10	SP
11	PC

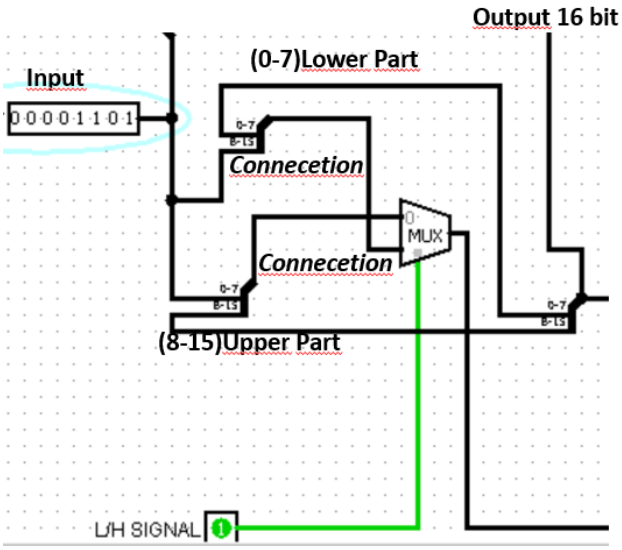
2.3 Part 2c

In this part, we designed a 16-bit Instruction Register that has a special load and arithmetic operations. This circuit has four inputs and one output.

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

2.3.1 L/H Input

The first input(L/H) helps us to determine which part of the output we are want to write the input such as if the input is '0' which means we are going to write to the lower 8 bit(0-7) of the register. If the input is '1' then it will be loaded into the upper 8 bit(8-15). To make this Lower and upper part selection we used 2:1 Multiplexer. We divided the 16-bit output into 2 part and we connected the input with them separately. The MUX's first input is the output's upper part(8-15)+our input this helps us to contain the register other bits unchanged. The MUX's second input is our input+output's Lower part(0-7).



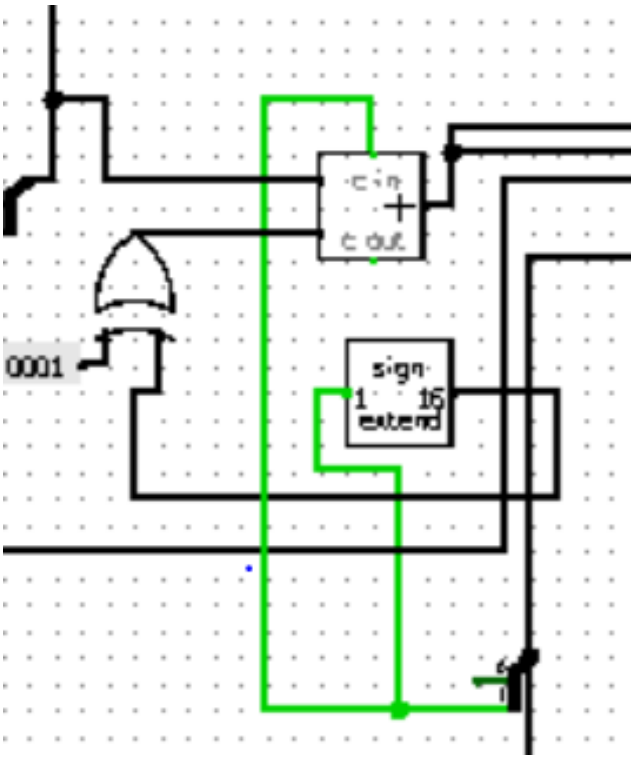
2.3.2 Enable Input

The second input is(Enable) if that input is '0 ' that means keep the register contend as it is, otherwise the operation determined with the combination of the other inputs. To implement this input we simply connected this one bit to the enable of the D Flip Flops enable.

2.3.3 Function Select Input

The third input (FS) helps us to choose which operation we are going to make with the content f the register such as if the input is '00' which means clear the content. To implement this operation we load '0x0' hex value to our register. If the input is '01' that means to increment the content of the register by one. To implement this we took the 16-bit output and made it the one of the 16-bit Full Adders input. And the other input is '-1' or '1' (16 bits). To make this part less costly we used an XOR gate to invert to input and

we gave 1 bit Cin input to select addition/substruction operation. We get this one bit from the 'XY' second bit(X bit) of the FS bit.



2.3.4 8 Bit Input

This input helps us to load the content of the register.

2.3.5 16 Bit Input

This input helps us to load the content of the register.

2.4 Discussion

This project gave us the opportunity to design and implement registers and register files. It helped us grasp abstract concepts we covered in the Computer Organization lessons. We also saw bus and flip flops on action.
Also, we gained experience on how to use Logisim software built-in libraries effectively, create our own circuits as library units and many features of it.