ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT REPORT

PROJECT NO	:	1
PROJECT DATE	:	12.03.2020
GROUP NO	:	36

GROUP MEMBERS:

150170087	:	Sırrı Batuhan ÇOKSAK(Group Representative)
150170069	:	Furkan Yusuf GÜRAY
150170039	:	Fatih MURAT
150160142	:	Ekin Zuhat YAŞAR

SPRING 2020

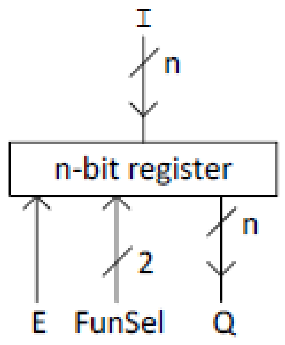
Contents

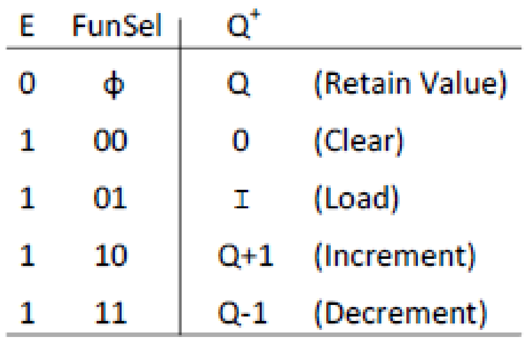
1 INTRODUCTION

In this project, we were expected to design and implement register and register files(systems with multiple registers).

2 PART 1

In the first part of the project, we were expected to design 2 different types of registers: (1) 8-bit register and (2) 16-bit register with the 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 with the functionalities is shown in Figure 1.





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)

2.1 Designing 8-bit Register

To start with, registers are n-bit binary number holding memory units. Normally, we use one flip flop among SR, JK, D and T flip flops as memory unit to hold 1 bit binary number. It means that in order to hold more than one binary number, we need more than one memory units and our register exactly do that operation. In our design, we used delay (D) flip flops to hold data. In order to do arithmetic operations which are increment and decrement we used full adders and to be able to control the rest of the system we used AND gates, OR gates, multiplexer(1:2) and decoder(2:4). How we implemented given functions will be explained one by one below.

First, in order to determine which function will be served by the circuit, we used decoder(2:4) according to table in the Figure 1 and we took inputs from 2-bit FunSel input pins.

Clear function means that set all 8 bits of the output to 0. In order to do that, we used D flip flop's 0 input pin. When 00 input comes from the FunSel input pins to decoder, clear function works. We connected our clear cable coming from the decoder with clock signal to AND gate. If we did not do that, our output would have been set to 0 instantly when the 00 input comes from the FunSel input pin whether clock signal is 1 or 0. And that would be totally disaster for our synchronized circuit design.

Load function means that set our 8-bit output to 8-bit input that we manually give to the 8-bit input pin. When 01 input comes from the FunSel input pins to decoder, load function works. We connected the related cable with the related input bit to AND gate to prevent the changing input of flip flops if the incoming input pin from FunSel is different from 01.

Increment function is adding 1 to the current output and decrement function is subtracting 1 from the current output. Since we can do the subtraction operation by using 2's complement system, we used full adder for both increment and decrement. Adding 1 to the current output is adding 0000 0001 in binary form. Subtracting 1 from the current output is adding 1111 1111 in binary form (0000 0001 -> 1111 1110 + 1 -> 1111 1111(2's complement of 0000 0001 in 8-bit binary form)). We connected decoder's 10 and 11 outputs to OR gate and output of the OR gate to the multiplexer's enable pin. This method works like if the 10 or 11 comes from the FunSel, one of the third or fourth output pin of the decoder has to be 1 and our OR gate gives 1 output. This output 1 coming from the OR gate enables the multiplexer and our increment or decrement function is delivered.

At the end, we put hex displayers and 8-bit output to observe our 8-bit register circuit design's behaviour with the given operations. Figure 2 shows our 8-bit register implementation.

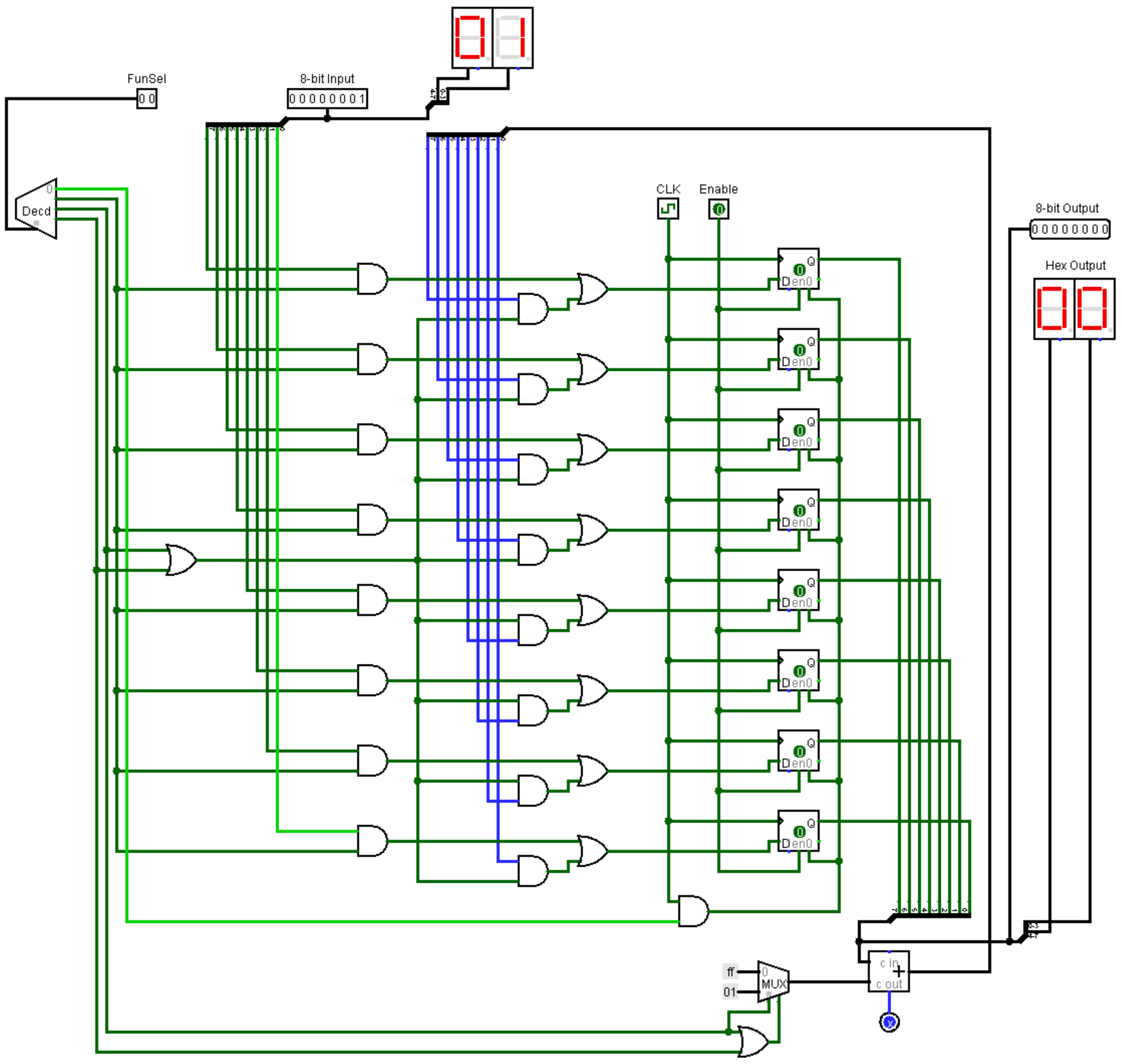


Figure 2: 8-bit Register Circuit Design

2.2 Designing 16-bit Register

In the second part of the first part, we were exptected to design 16-bit register with the same functionalities as we did in the 8-bit register design. In our 16-bit register design, there were no differences with the 8-bit register. Only difference is the number of bits that were used(16-bit for this design and 8-bit for for former 8-bit design). Figure 3 shows our 16-bit register design.

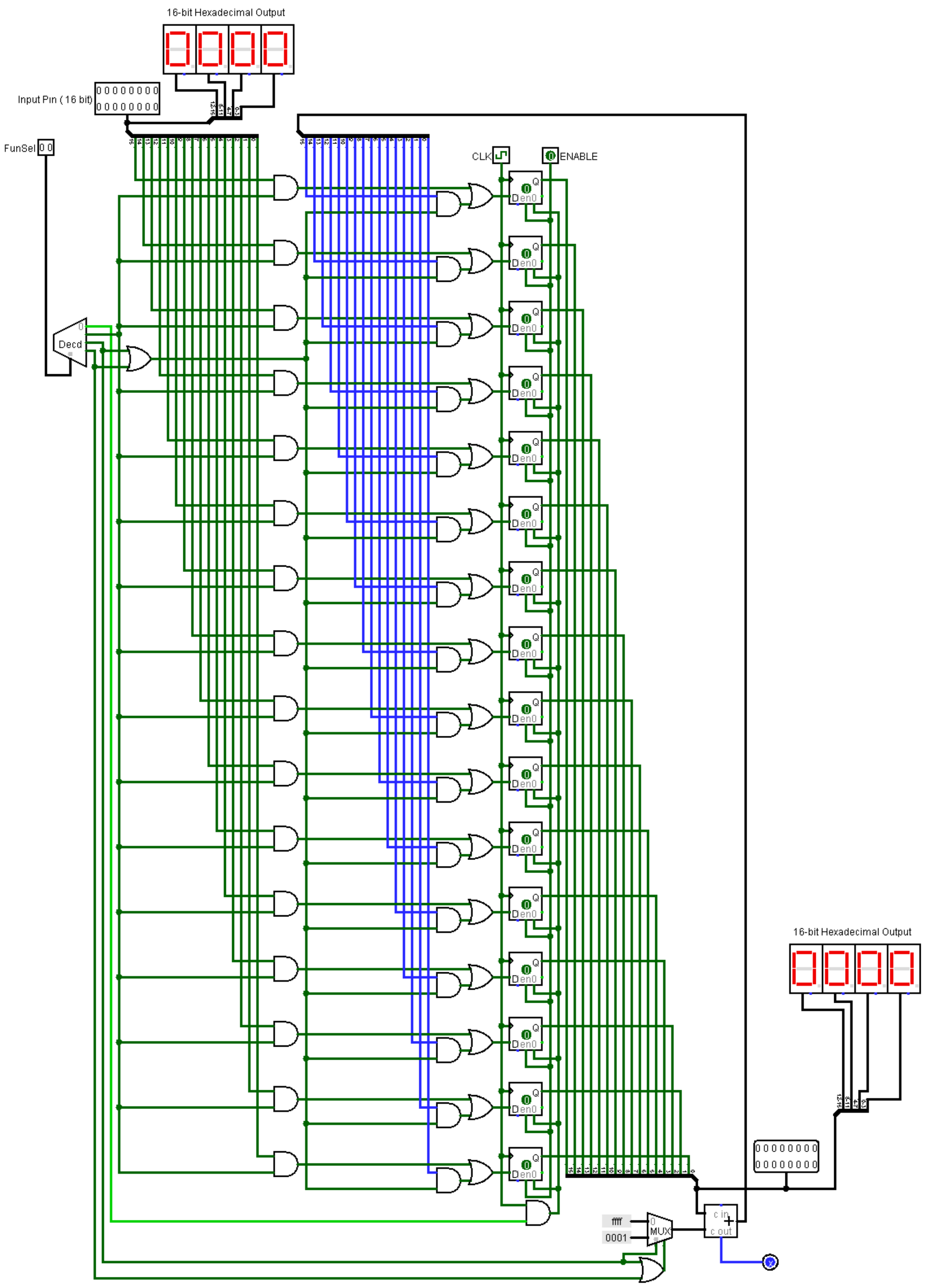


Figure 3: 16-bit Register Circuit Design

As a result of design of the registers, we can simply show them as in Figure 4 and Figure 5. Of course the visual of this forms of the registers that we used in part-2 may differs from this figures. But the usage is same, only the pins are differently located.

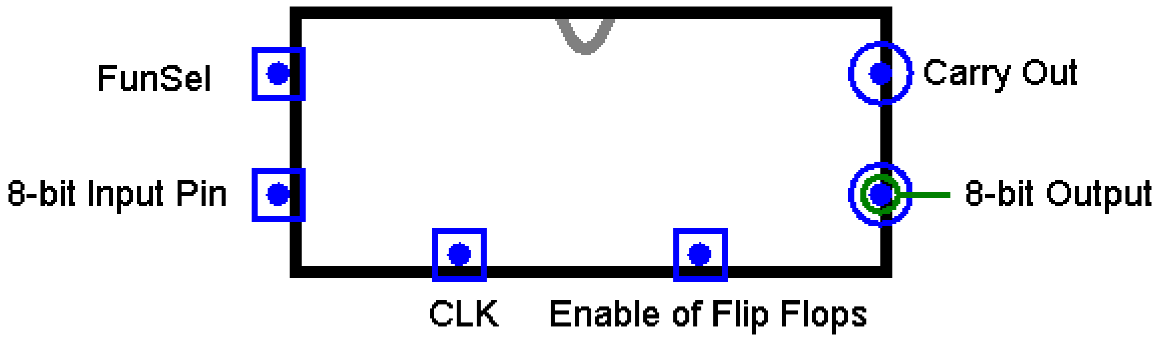


Figure 4: 8-bit Register as a box

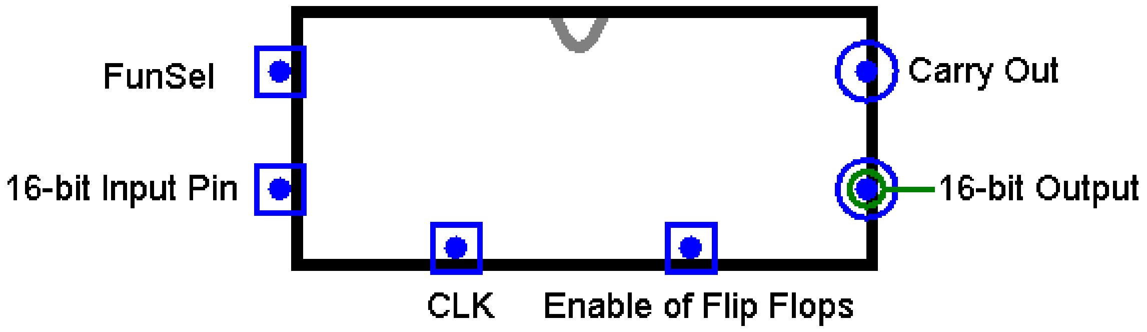
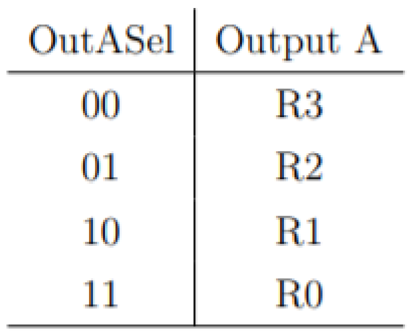


Figure 5: 16-bit register as a box

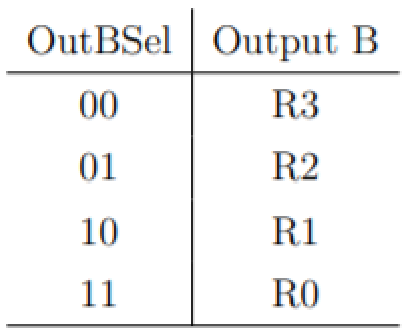
3 PART 2

3.a Designing 8-bit General Purpose Registers

In this part, we designed a system which includes 4 register(8-bit registers designed in part 1). This design can select registers and applying selected functions to these registers. Also, displaying only desired registers's output values. Registers are selected via 2-bit RegSel signal and functions are selected with the 2-bit FunSel signal and applied to these registers with the next clock cycle. Registers displayed on the hex display are controlled by 2-bit OutASel and OutBSel signals.



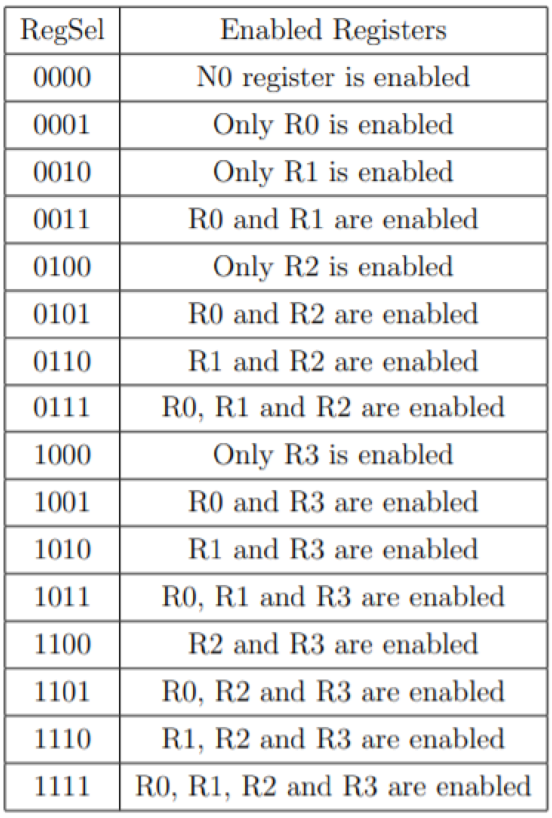
OutASel	Output A
00	R3
01	R2
10	R1
11	R0



OutBSel	Output B
00	R3
01	R2
10	R1
11	R0

Figure 6: OutASel and OutBSel tables

OutASel and OutBSel control which register will be displayed as shown table above. Two register's output can be seen on the two hex display. Two 4x1 multiplexers are used(One of them displaying register selected with OutASel, other one OutBSel) for selecting the display register.



RegSel	Enabled Registers
0000	N0 register is enabled
0001	Only R0 is enabled
0010	Only R1 is enabled
0011	R0 and R1 are enabled
0100	Only R2 is enabled
0101	R0 and R2 are enabled
0110	R1 and R2 are enabled
0111	R0, R1 and R2 are enabled
1000	Only R3 is enabled
1001	R0 and R3 are enabled
1010	R1 and R3 are enabled
1011	R0, R1 and R3 are enabled
1100	R2 and R3 are enabled
1101	R0, R2 and R3 are enabled
1110	R1, R2 and R3 are enabled
1111	R0, R1, R2 and R3 are enabled

Figure 7: RegSel control inputs.

Which registers will be selected are controlled by 4-bit RegSel signal as shown table above. Regsel connect to register's Enables. When 8-bit input signal entered to system, selected register will take input and applied selected function on the FunSel.

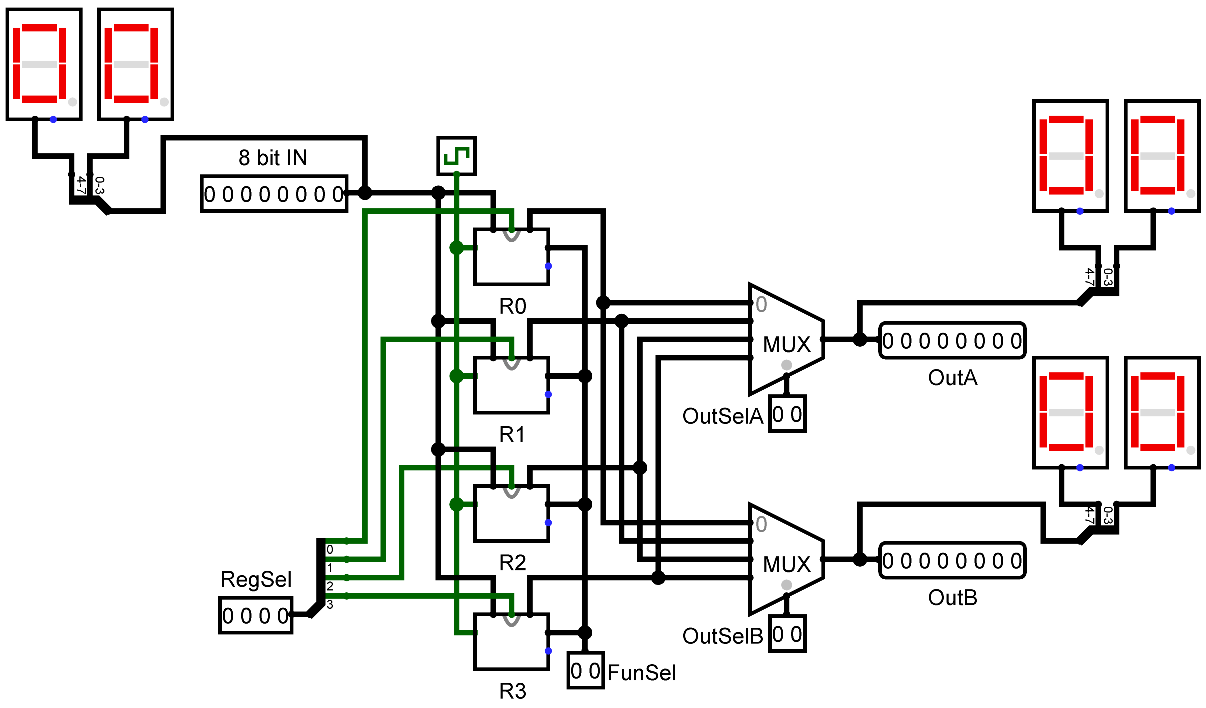


Figure 8: Our 4 8-bit register build.

3.b Designing 8-bit Address Registers

In Part 2.b, we were asked to design the system shown in Figure 9, which includes three 8-bit address registers: PC, AR and SP. FunSel and RegSel worked identical to parts before.

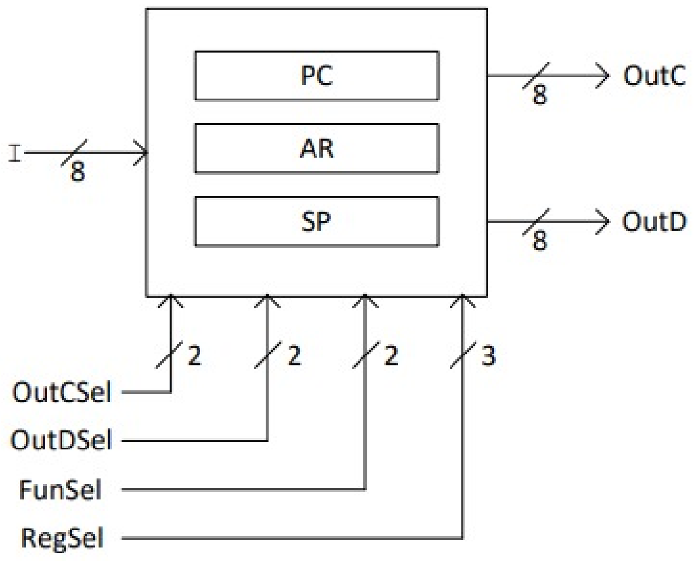
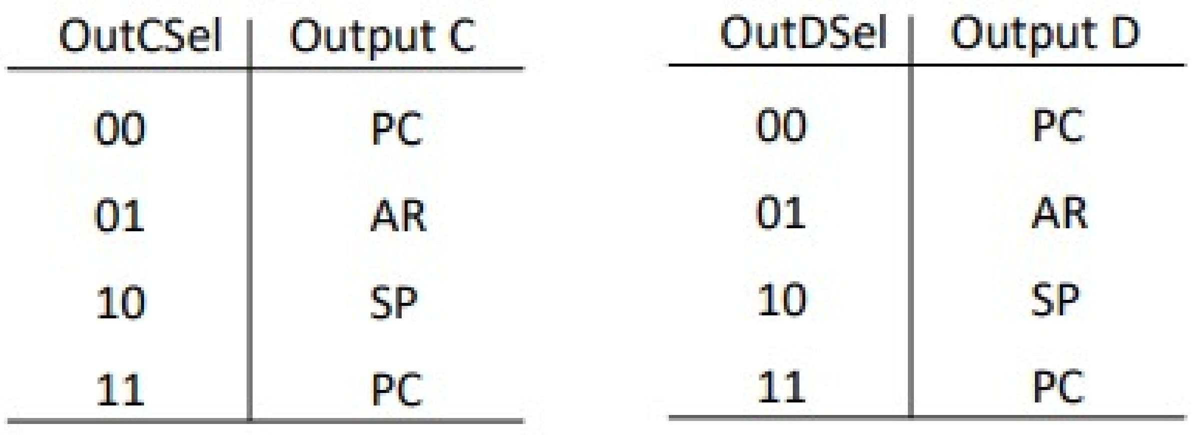


Figure 9: The system we had to build in Part 2.b



OutCSel	Output C
00	PC
01	AR
10	SP
11	PC

OutDSel	Output D
00	PC
01	AR
10	SP
11	PC

Figure 10: Reference table used to build part 2-b

Our design featured RegSel as individual ENABLE inputs for PC, AR, SP registers respectively. Thus, this modification made sure only intended registers were enabled and exposed to whatever FunSel function was activated at the time. We used 4:1 multiplexers for OutSelC and OutSelD to accurately keep track of which register was intended at Output C and Output D, respectively. At the end we were able to test all cases and made sure this design implemented the system and table given us. Figure 9 shows our implementation.

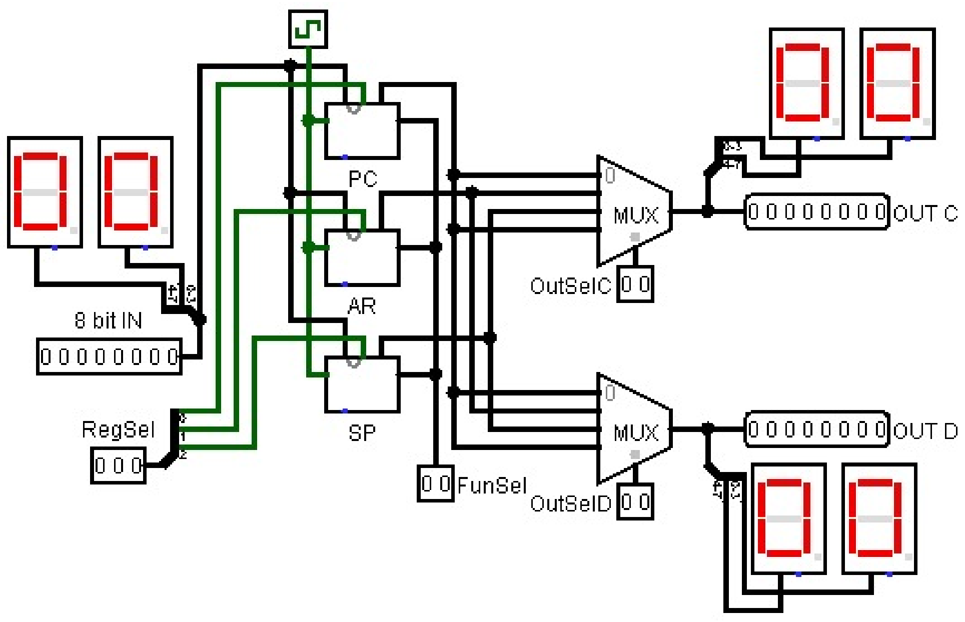
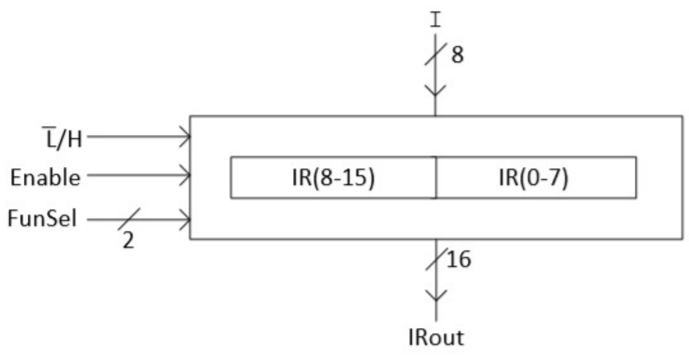
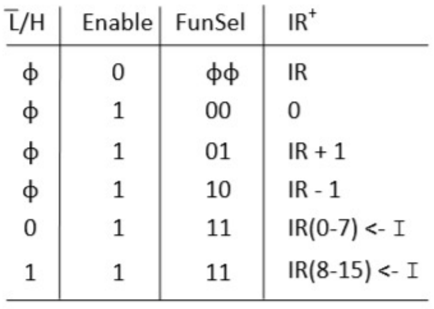


Figure 11: Our 3 8-bit register build

3.c Designing 16-bit IR Register

In the last part of the project we were expected to design an 16-bit instruction register with the graphic symbol and characteristic table are given in the Figure 12.





$\overline{L}$/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

Figure 12: Graphic symbol of the IR register (Left) and its characteristic table (Right)

After analyzing the characteristic table of the IR, we designed and implemented the IR as shown in Figure 13.

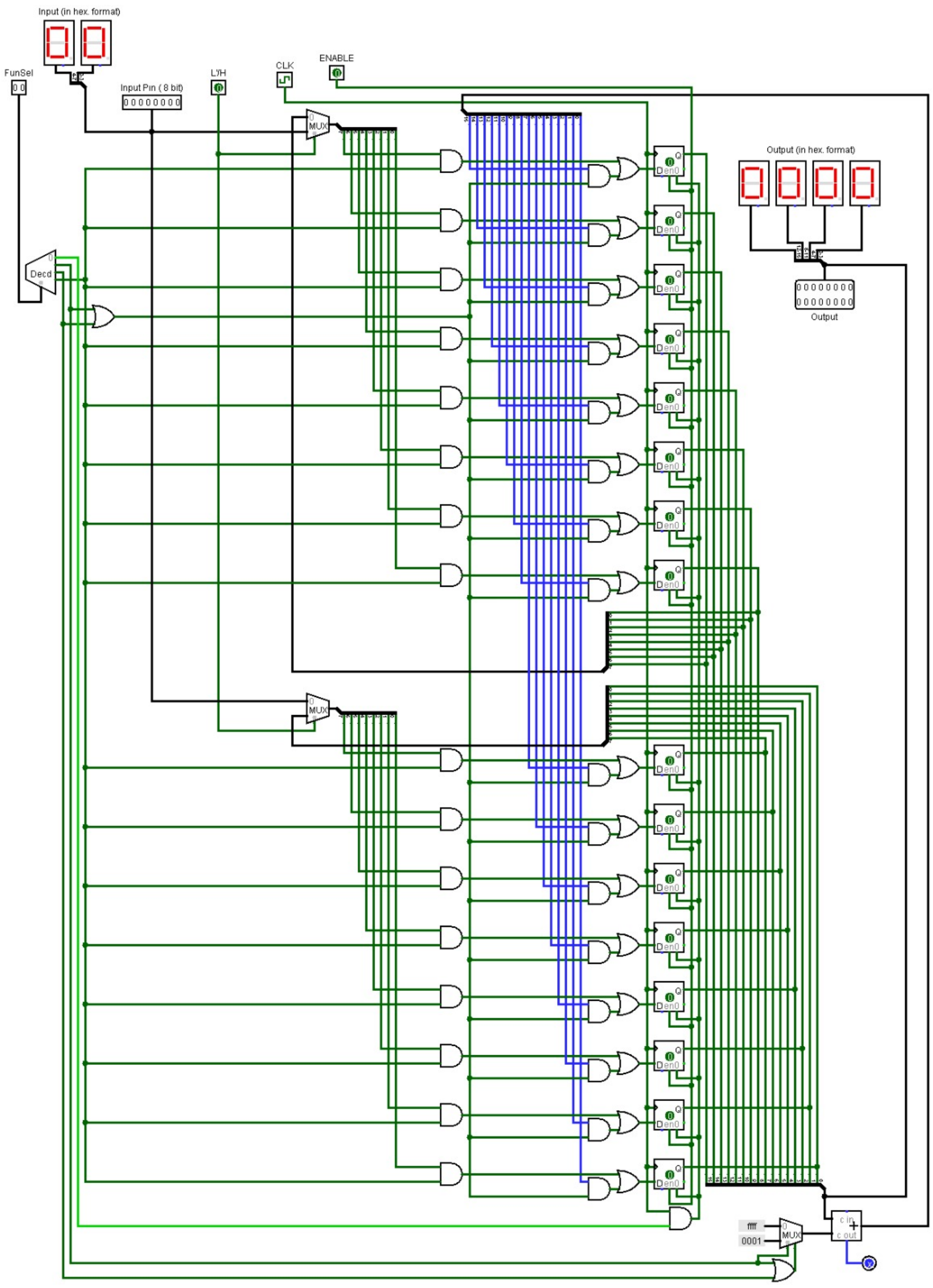


Figure 13: 16-bit IR register with the functions described in the previous Figure

While designing the IR we used the 16-bit register we designed in the Part 1. However, while designing we noticed there is a need of small adjustments such as different FunSel values for same operations hence, we had to rewire some cables. Additionally, Load operation was different from the operation for the 16-bit register in Part 1. Originally there were 16 bit input in the register however, in the characteristic table of the IR load operation had only 8 bit input furthermore, this 8 bit input could be loaded into the first or last 8 bit of the register. In order to overcome this challenge we feed the output of the first and the last 8 flip flops to two different 2:1 Mux's. Additionally, we connected the 8 bit input to both of these multiplexer(At this point it should be noted that the 8 bit input is connected to I_0 of one of the multiplexers and I_1 for the other multiplexer to use one less not gate.) and the selection line for these multiplexers are connected to the new input $\bar{L}/H$. In short, either the current value for the first and last 8 bit or the 8 bit input is selected from the multiplexers which has a selection input connected to $\bar{L}/H$ and loaded into the register in the next clock signal.

4 CONCLUSION

In conclusion, 2e designed and implemented several registers and register files with the given functionalities. During the design phase our group analyzed the tasks together found the design ideas collectively, additionally at part 2c we come up with more than one design. During the implementation phase we worked with 2 sometimes 3 computers at once and implemented the same designs and picked the most eye friendly design of them all and continued with it. Furthermore, in part 2c we implemented more than one design and picked the design which was easiest to understand. From start to finish including the report, every member of our group had added or changed several part of the project, every member were aware that it was a group project and had to be done collectively.