Report from Lab 4

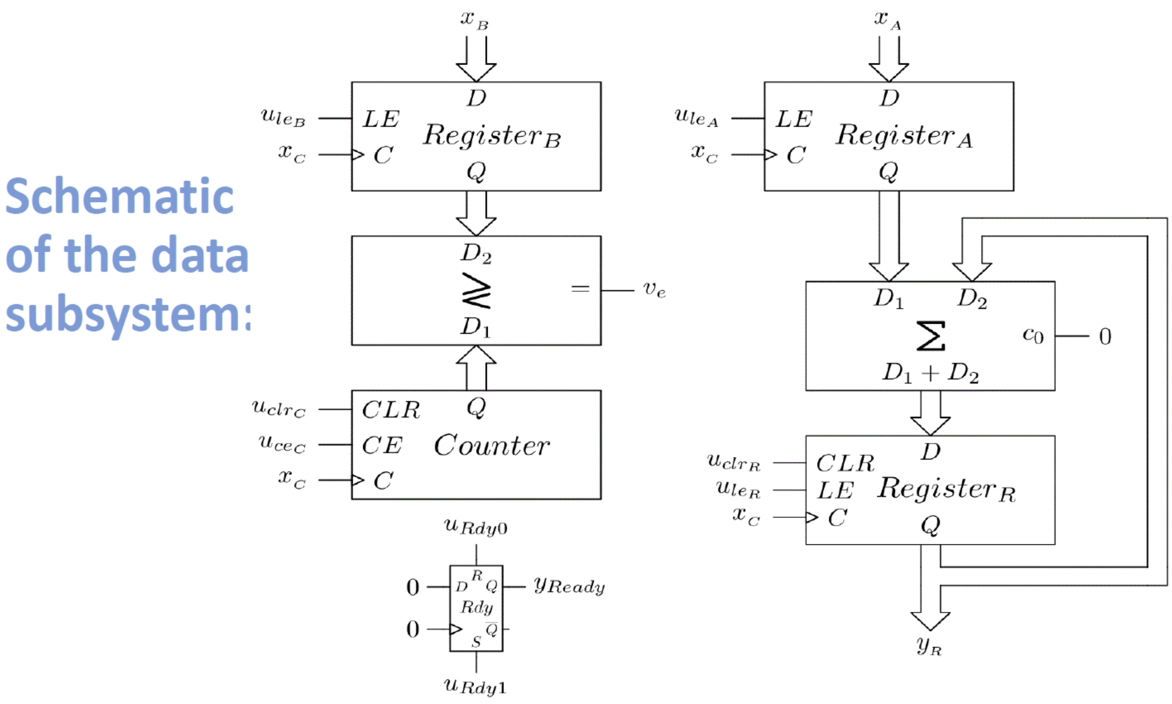
Functional blocks

I declare that this piece of work, which is the basis for recognition of achieving learning outcomes in the Digital Circuits course, was completed on my own.

First and last name: Michał Łezka
Student record book number (Student ID number): 303873
Date: 08.05.2021

Theoretical design

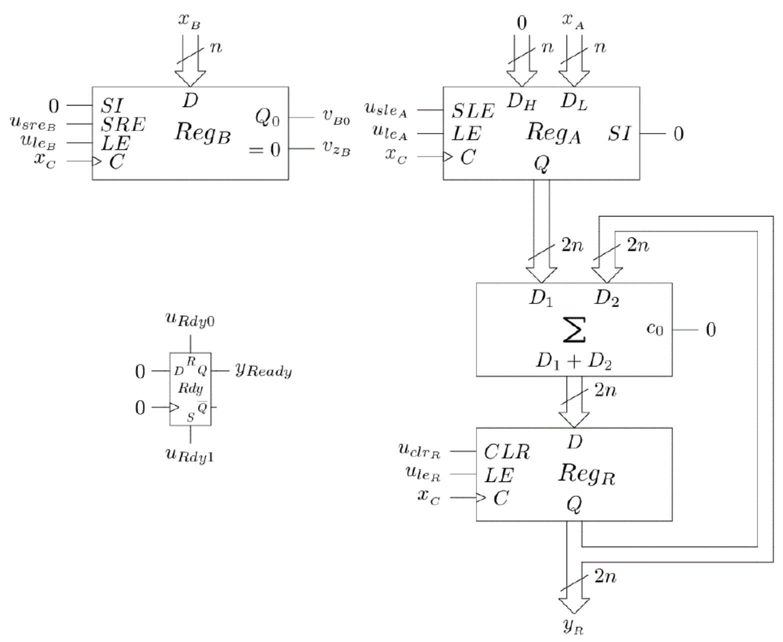
This time around theoretical design was provided during intro presentation and we just had to copy them to the Xilinix ISE. For part A and B we were supposed to use this schematic:



It represents a simple way of multiplying numbers.

And for part C, this schematic:

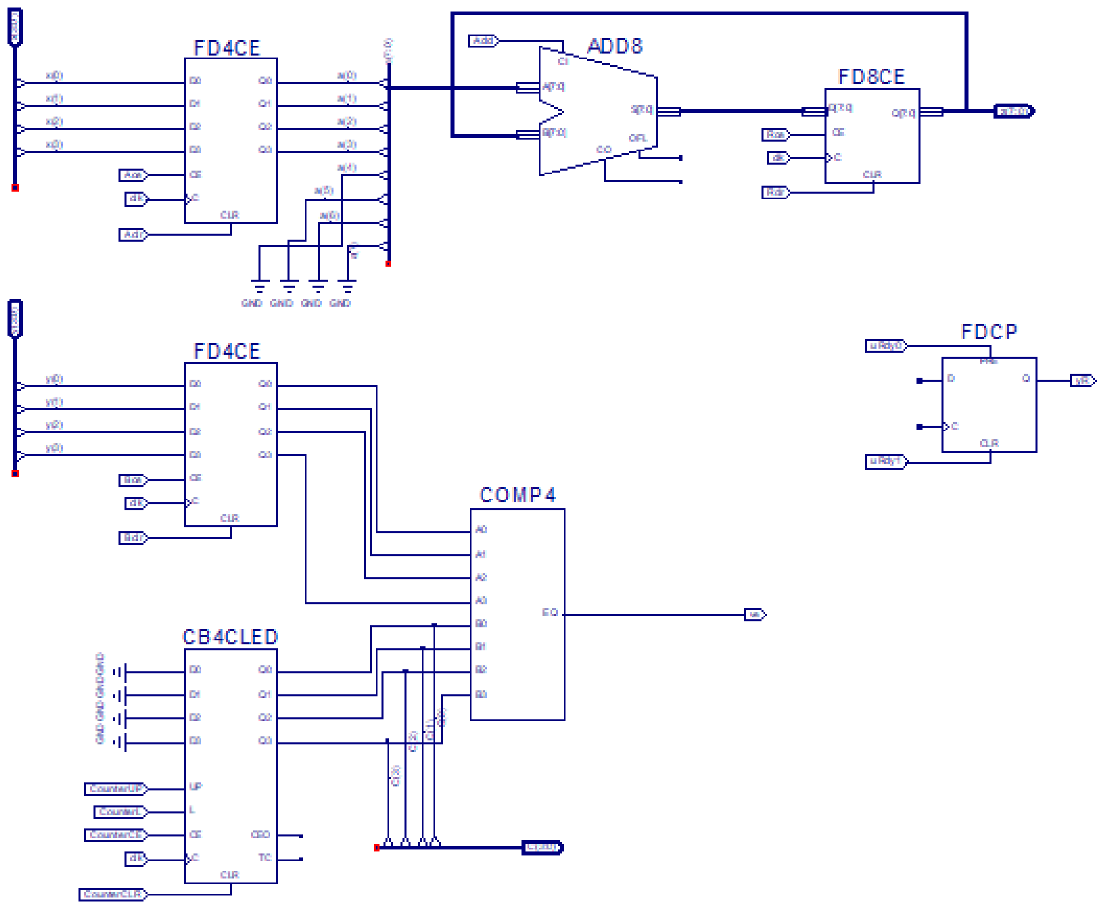
Schematic of the data subsystem:



This one represents a more complicated, but much more efficient way of multiplying numbers.

Xilinix schematics and test results

The first schematic when transferred to Xilinix looked like that:

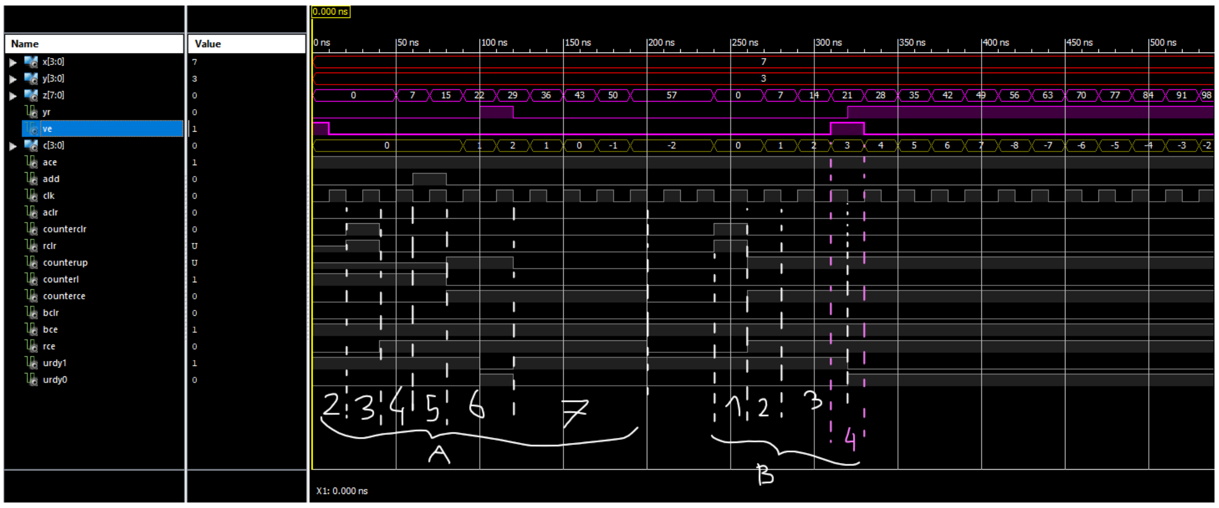


I used the most basic registers possible as I didn’t need to shift anything. As for the counter, I have used a more complex one to be able to both increment and decrement, as it was needed in tasks for part A.

This brings us to task for Part A, they were, in order:

1.Take two rightmost digits greater than 2 numbers from your student ID.
2.Load these numbers into registers A and B.
3.Clear register R and counter C.
4.Load register R.
5.Set c 0 =1 and load register
6.Increment the counter 2 times
7.Decrement the counter 4 times

As for task 1, the numbers I should take from my ID: 303873 would be 7 and 3. The rest of the tasks were performed using testbench, and the results can be seen in the simulation:



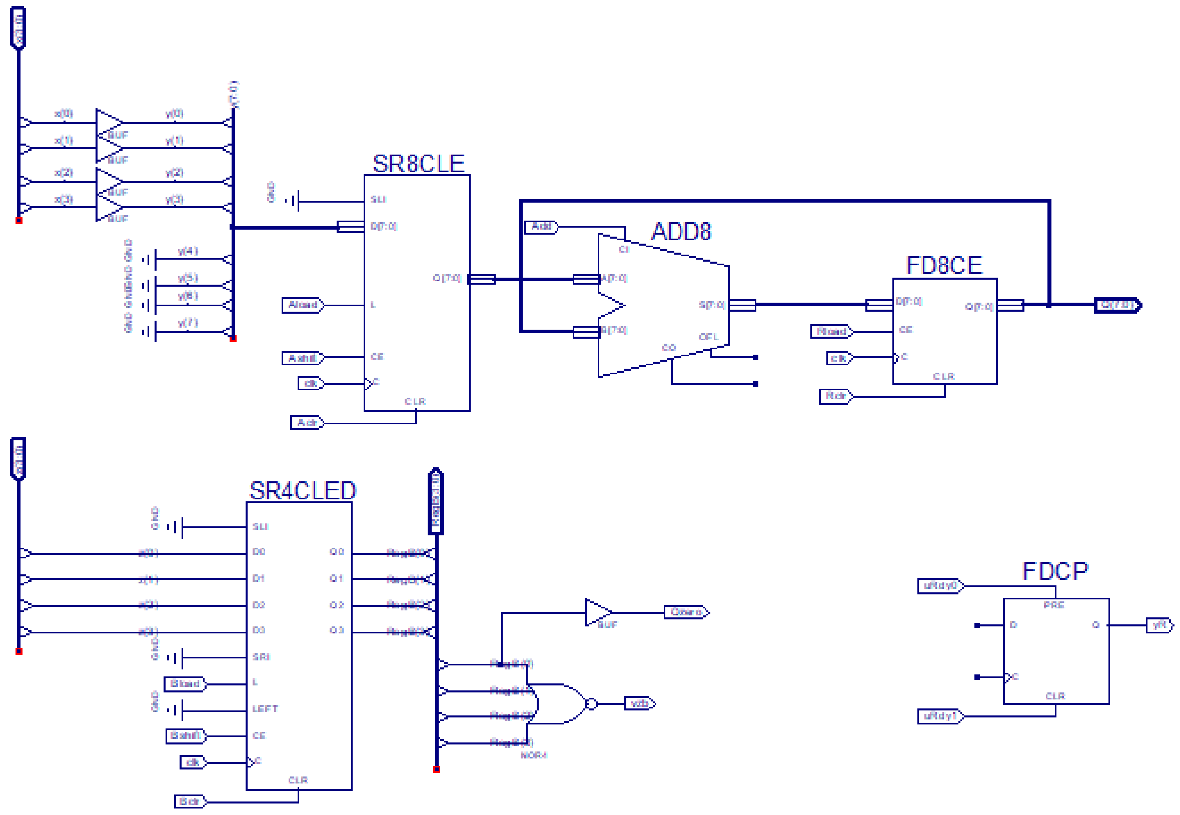
Task 2: we can see the numbers are loaded from the beginning, as clock enable on A and B were set to ‘1’ from the start.
Task 3: “counterclr” and “rclr” were turned on for on clock pulse
Task 4: “rce” was turned on
Task 5: c0 = 1 “add” was turned on for one clock pulse and “rce” is still turned on
Task 6: counter output is the bust c[3:0] and it works correctly
Task 7: once again, c[3:0] works correctly

Now, moving on to Part B:

1.Clear register R and counter C
2.Load register R and increment counter C;
3.Repeat step 2 until counter is equal to register B
4.Set and reset flip flop (“1” indicates the moment, when result is valid)

Task 1: “counterclr” and “rclr” turned on for one clock pulse
Task 2: “rce”, “counterup” and “counterce” turned on
Task 3: repeating step 2
Task 4: “urdy1” changed from 1 to 0 and “urdy0” changed from 0 to 1

As for part C I had to use a different schematic:

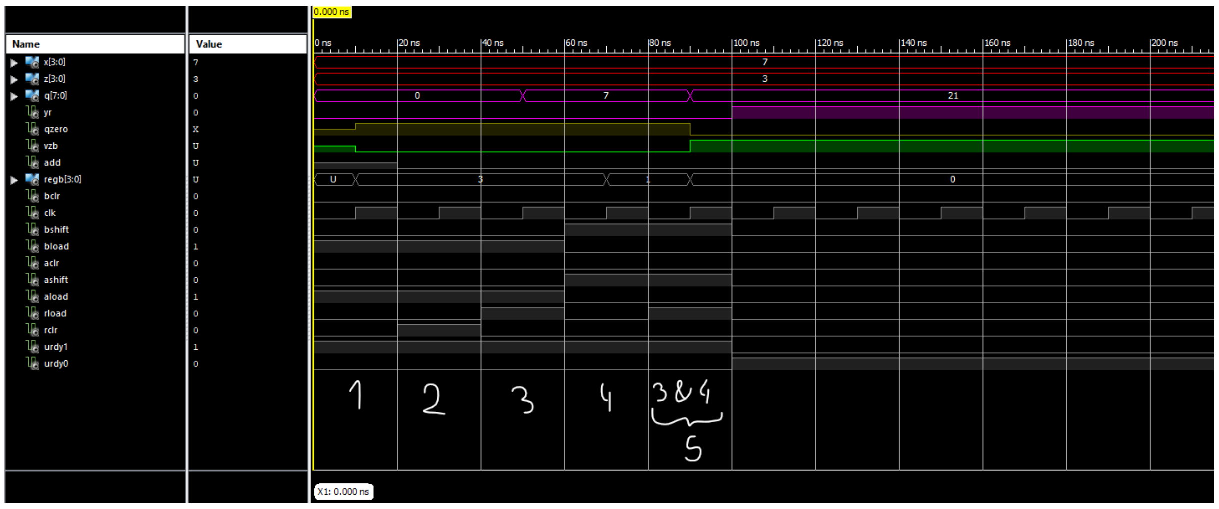


Here we needed more complex registers with the ability to shift left or right, except for the rightmost register, there we could use a simple one.

Part C tasks:

1.Load your numbers into registers A and B
2.Clear register R and set C0 = 0
3.If least significant bit of B is 1 load register R
4.Shift left register A and shift right register B
5.Repeat steps 3 4 until output of B becomes 0.

Part C simulation results:



Task 1: Numbers are visibly loaded from the start
Task 2: "rclr" changed to one for one clock pulse and C0 = 0 - "add" changed to 0
Task 3: "rload" switched on for one clock pulse
Task 4: "aload" and "bload" turned off to enable shifting, "ashift" and "bshift" turned on to start shifting
Task 5: repeated task 3 and 4 once, this time within a one clock pulse

Troubleshooting

At first I thought this device is supposed to be able to function somewhat automatically so in the first schematic I have connected comparator output with clock enable in R register, but then noticed it is supposed I am supposed to do that job in the testbench.

Conclusions

Overall, the experiment succeeded. It took more time than expected but I believe everything works correctly. Although, now looking at it from a perspective, "load register" might have not meant "turned on loading in a register" but "turn on loading in a register for one clock pulse", but I am uncertain as which one should I interpret it.