# ECE 375: Computer Architecture

# Final Project

Honor Code:

I have neither given nor received unauthorized assistance on this graded report.

**X Hussein El-Souri**

## Contents

## 1) Procedure

First I started off as instructed with designing the timing sequence using the given hint. By mimicking the figure provided in the assignment (Figure 1); I was able to have a schematic (Figure 2) representing the timer then converting that schematic into a symbol (Figure 3).

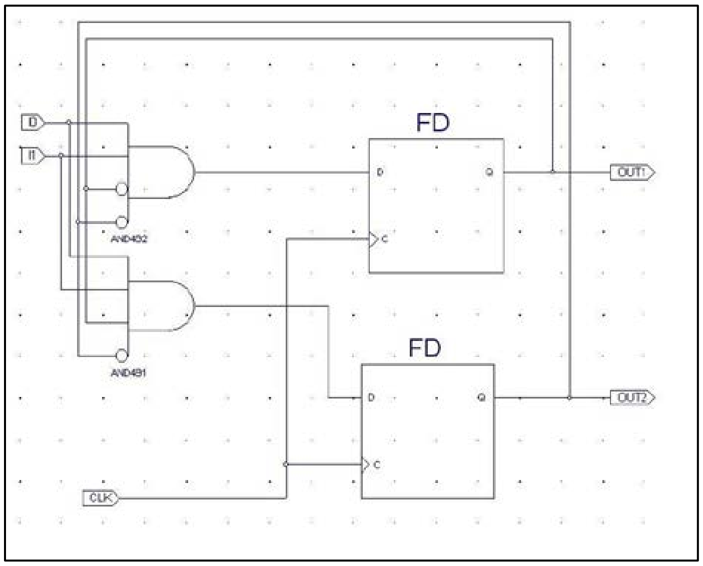

*Figure 1: Timing schematic provided in requirements*

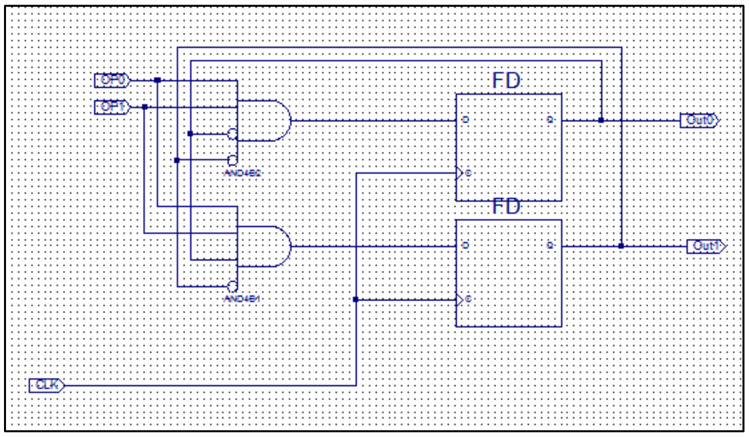

*Figure 2: My timing Schematic*

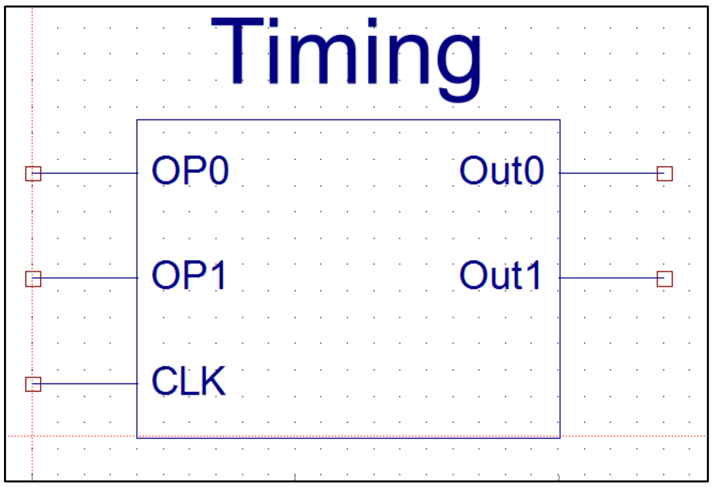

*Figure 3: My Timing Symbol*

I then proceeded to design the sequencer as instructed in the lab according to Figure 4

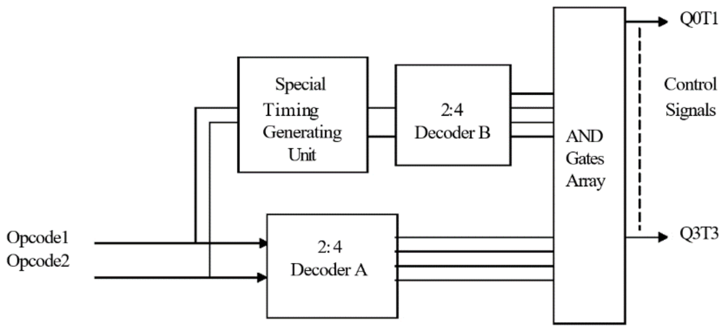

*Figure 4: Sequencer figure from Assignment*

The only extra step here is to figure out the AND gates array logic.

| OP0 | OP1 | D0 | D1 | D2 | D3 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

*Table 1:Output for Register A*

| OP0 | OP1 | Out0 | Out1 | D0 | D1 | D2 | D3 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | X | X | 0 | X | X | 0 |

*Table 2: Expected output from Register B*

For Table 2 the input for the decoder in the output from the timer and I notice the following equation for the needed instructions (I will refer to each output of the decoder with the name as subscript ex $D0_A$):

LoadReg1: $D0_A D0_B$  when OP0=0 and OP1=0

CompReg1: $D1_A D1_B'$ when OP0=0 and OP1=1

MOVREG1: $D2_A D2_B'$ when OP0=1 and OP1=0

SwapRegStep0: $D3_A D3_B'$ when OP0=1 and OP1=1 (Step0 $D3_A$=1 $D3_B$=0)

SwapRegStep1: $D3_A D1_B$ when OP0=1 and OP1=1(Step1 $D3_A$=1 $D1_B$=1)

SwapRegStep2: $D3_A D2_B$ when OP0=1 and OP1=1(Step1 $D3_A$=1 $D2_B$=1)

Note when OP0=1 and OP1=1 Out0 and Out1 toggle (when one is 0 the other is 1 and vice versa)

This results in the following schematic Figure 5

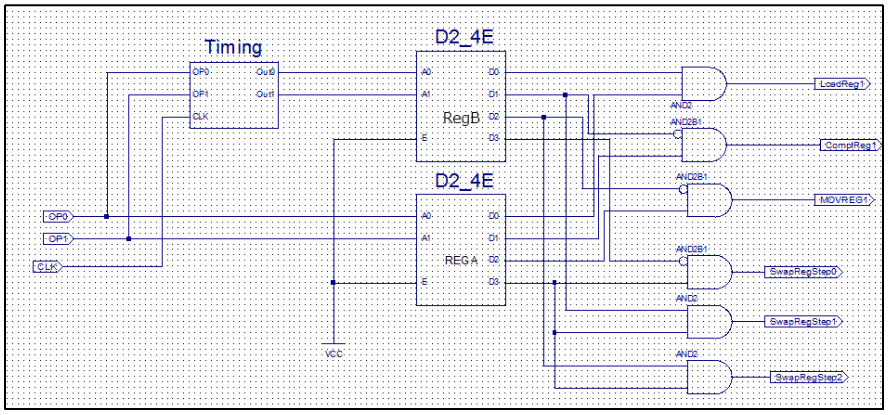

*Figure 5: Sequencer Schematic*

It would make sense for the swap to need three steps: Step0 move Reg2 to a temp (buffer) step1 move reg1 to 2 and step 3 move buffer to Reg 1. This AND Gate array design result in SwapRegStep0, SwapRegStep1 and SwapRegStep2 to have this signal 100, 110, 101 repeating.

(This is will be reference later as ref1).

I then make the schematic into a symbol

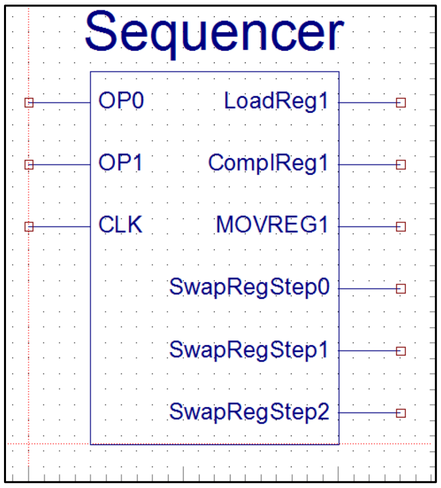

*Figure 6: Sequencer Symbol*

I then proceeded to design other parts needed for this project such as a 16X4S2 multiplexer (Figure 7 )

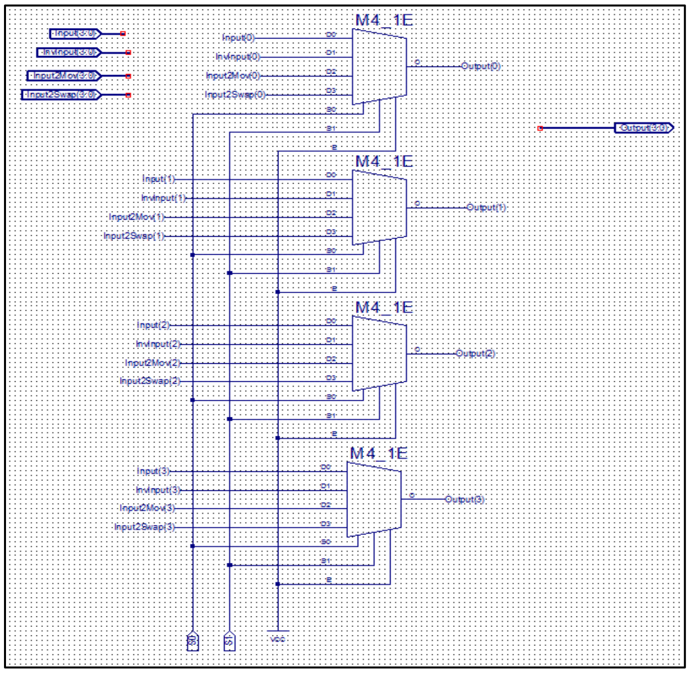

*Figure 7: 16X4S2 MULTIPLEXER*

I used bus lines to remove the hustle of connecting many wires.

I then converted that schematic into a symbol

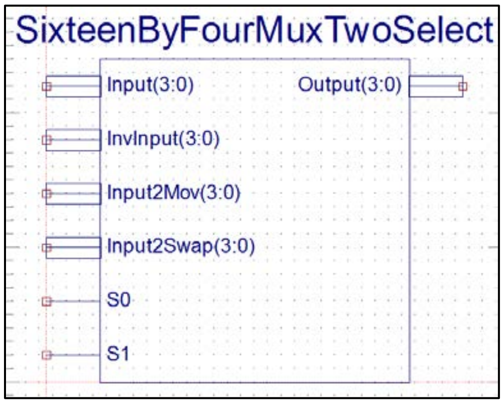

*Figure 8:16x4S2 multiplexer*

I also made a symbol for a 4 bit inverter (Figure 9 and Figure 10) and a register (Figure 11 and Figure 12) for ease of use

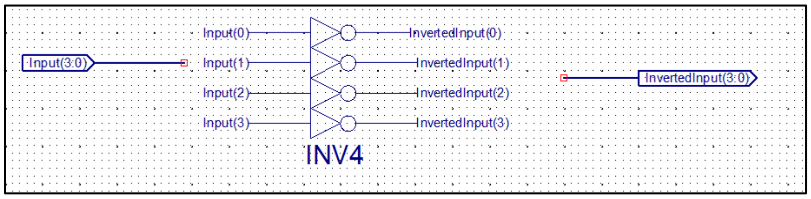

*Figure 9: FourBitInv Schematic*

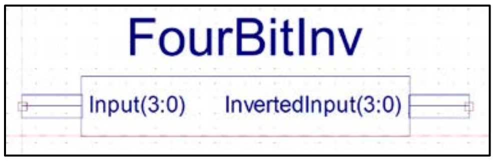

*Figure 10: FourBitInv Symbol*

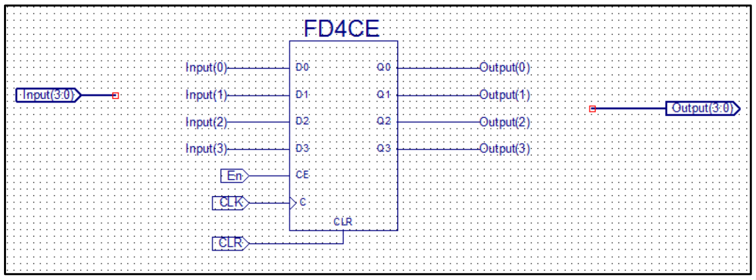

*Figure 11:MyRegister Schematic*

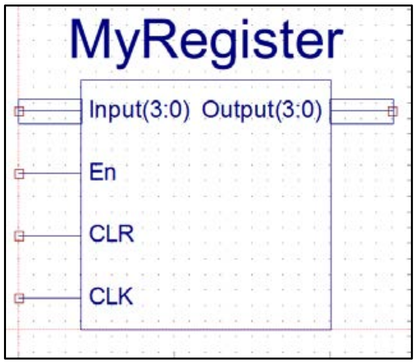

*Figure １２: MyRegister Symbol*

Finally we can combine all these parts into a main schematic using the help of Figure １３

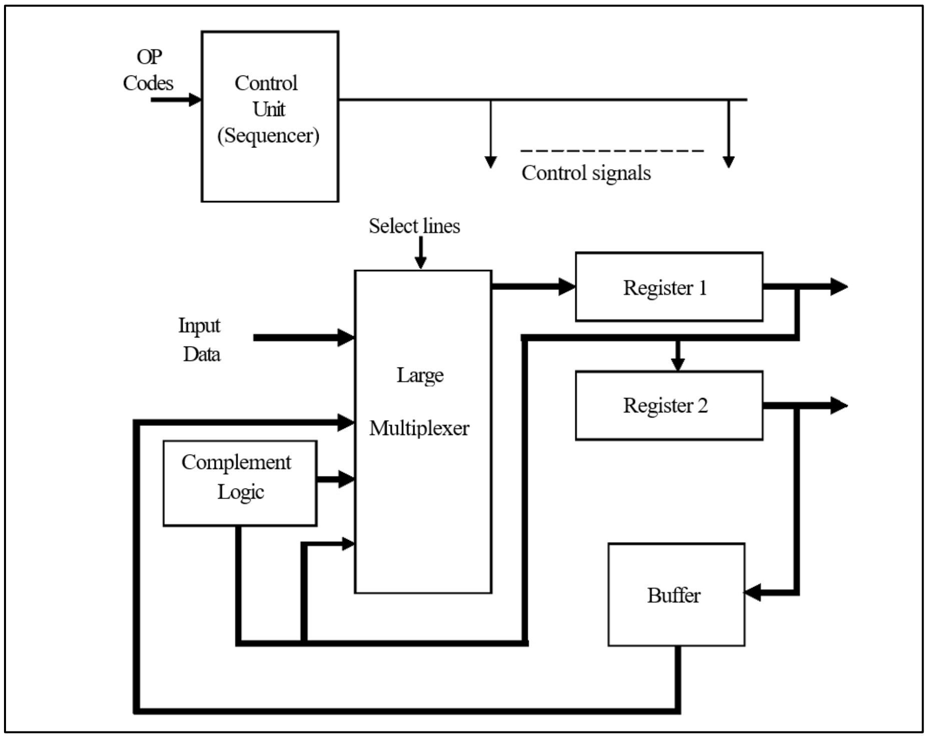

*Figure １３: Tiny Processor hint*

Making the proper connections and labeling we can finally connect the main schematic which will look like this Figure １４:

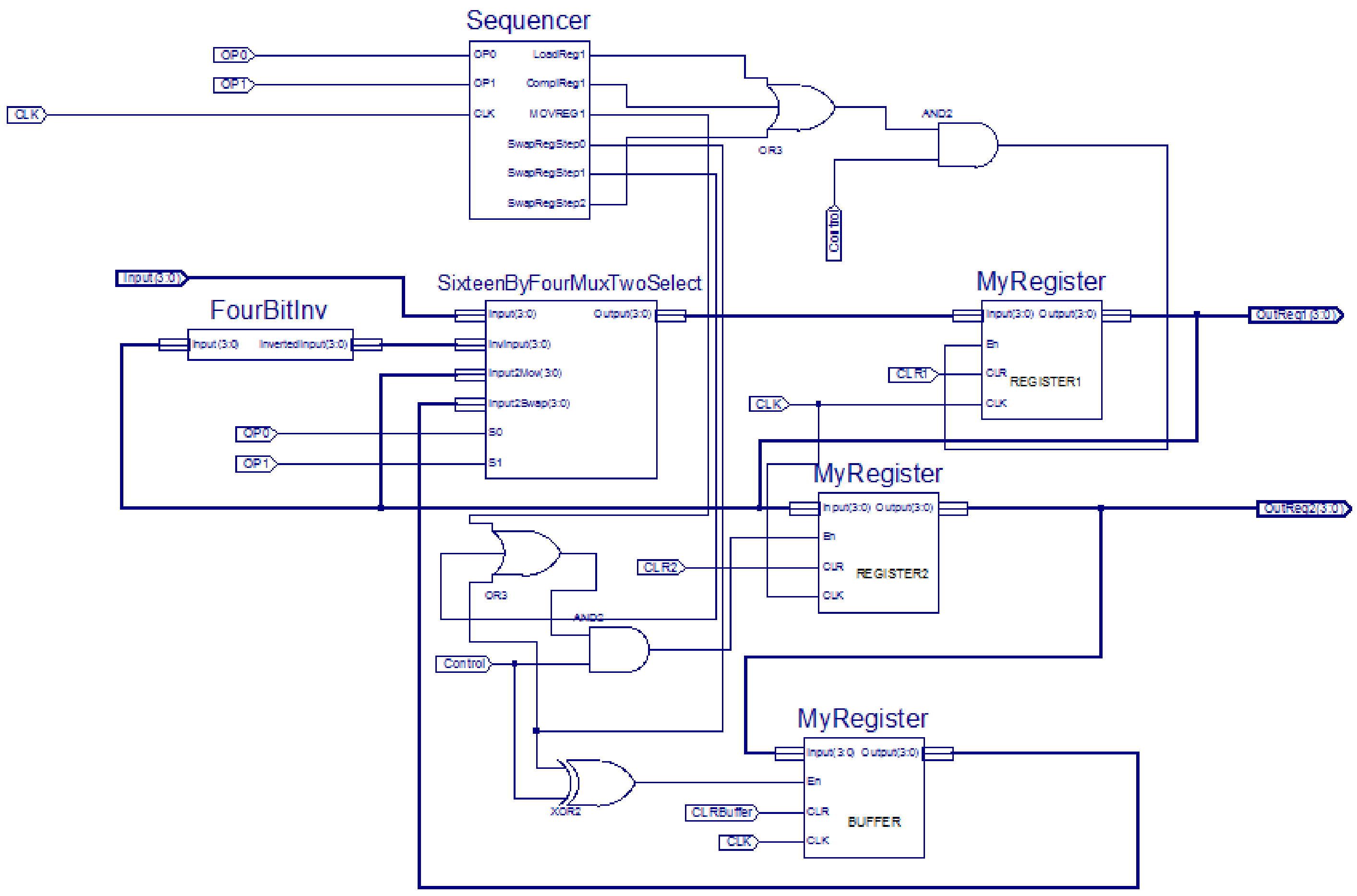

*Figure 1 4: Main Schematic*

A few things to explain: notice that register 1 should be enabled when LoadReg1 or CompReg1 or SwapStep2 are high. I then added a control signal since the signal is repeating to avoid over writing to the register too many times. This is similar to a valid bit in an N-way cache and this can be avoided if we have an operating system to control whether or not the data in this location is desired. The same is done for register 2 which is enabled when MOVEREG1 or SwapStep1 is high but here I had to add an extra signal to OR SwapStep2 because of #Ref1 along with the AND gate. This again ensures that the registers are only enabled when desired. Finally the buffer is only enabled when SwapStep0 and the control are opposite and this is also to avoid rewriting into the buffer and looping the swap (This will be referenced as Ref2).

I also added signals to clear the registers and those will be assigned to push buttons.

## 2) Waveforms and test bench

```
-- *** Test Bench - User Defined Section ***
CLOCK : PROCESS
      BEGIN
      CLK <= '0';
      WAIT FOR 50 ns;
      CLK <= '1';
      WAIT FOR 50 ns;
      END PROCESS;

   tb : PROCESS
   BEGIN
      Control <= '1'; CLR1 <= '0'; CLR2 <= '0'; CLRBuffer <= '0';
            -----------
            OP0 <= '0'; OP1 <= '0';
            Input(3) <= '1'; Input(2) <= '0'; Input(1) <= '0'; Input(0) <= '0';
            wait for 50 ns;
            ----------------------
            OP0 <= '1';
            wait for 50 ns;
            ---------------------
            OP1 <= '1'; OP0 <= '0';
            wait for 50 ns;
            -----------------
            OP0 <= '1';
            wait for 50 ns;
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

This test bench is used to simulate the waveforms which might seem wrong but that is only because the test bench is not all inclusive and only features a few cases.

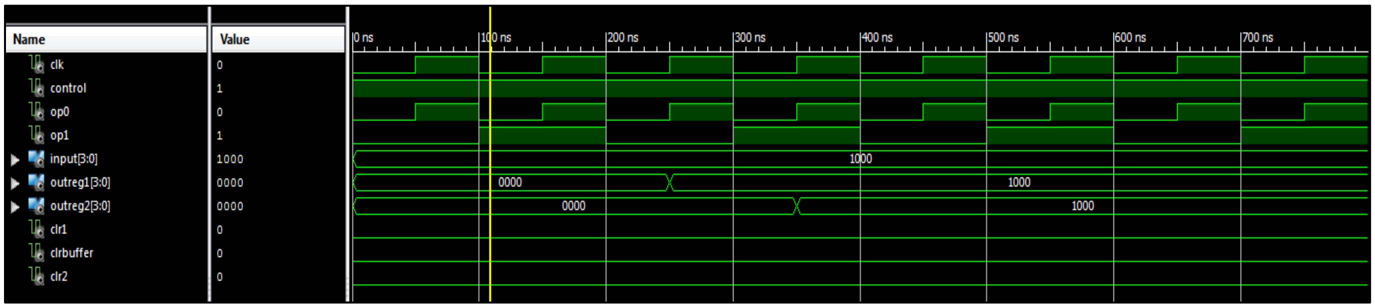

Figure 15: Waveforms

## 3) Pinning and user constraints

```
# PlanAhead Generated physical constraints

NET "Input[0]" LOC = P11;

NET "Input[1]" LOC = L3;

NET "Input[2]" LOC = K3;

NET "Input[3]" LOC = B4;

NET "OutReg1[0]" LOC = M5;

NET "OutReg1[1]" LOC = M11;

NET "OutReg1[2]" LOC = P7;

NET "OutReg1[3]" LOC = P6;

NET "OutReg2[0]" LOC = N5;

NET "OutReg2[1]" LOC = N4;

NET "OutReg2[2]" LOC = P4;

NET "OutReg2[3]" LOC = G1;

NET "CLK" LOC = C8;

NET "CLR1" LOC = A7;

NET "CLR2" LOC = M4;

NET "CLRBuffer" LOC = C11;

NET "Control" LOC = G12;

NET "OP0" LOC = E2;

NET "OP1" LOC = N3;
```

## 4) Conclusion and remarks

This lab revealed many important ideas when it comes to designing the lab assignment and figures provided along with it were extremely helpful and this project even helped me tackle minor issues I suffered from in previous labs.

The most important part in this project is #Ref1 and the explanation provided in #Ref2. Assigning the control signal to a push button is also important to have more control over when exactly are changes happening.

It is also fair to point out that the swap only works once.