ÓBUDAI EGYETEM NEUMANN JÁNOS
INFORMATIKAI KAR

Digital systems
e –learning - materials

Lab pratice

Lab-05
Multiplexer circuit

ÓE-NIK

Budapest, 2018.

1. Lab-05: Multiplexer circuit

1.1. Aim of laboratory:

Aim of the laboratory is the analysis of multiplexer circuits. You will design a multiplexer with eight inputs and you shall create the circuit schematic afterwards. The circuit can be implemented on the Basys board after simulation and testing of schematic.

1.2. Creation of schematic

The multiplexer is a circuit which let only a single of its inputs through itself, depending on the address received on the selector bits (dedicated inputs). This is one of the most commonly used combinatorial circuit. In the simplest case the address is a single bit selector signal, which select one from two input lines (I0 and I1) (see Fig. 1 and Fig 2.). The truth table for two inputs can be found below:

S	Y
0	I0
1	I1

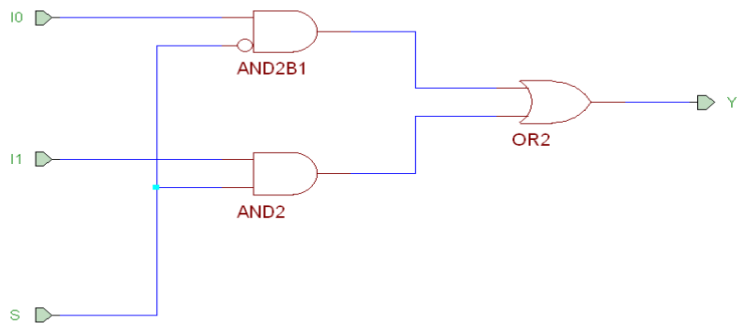

Figure 1. – Schematic of a two bit multiplexer

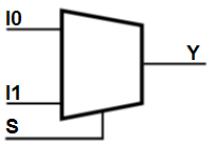

Figure 2.- Block scheme of two bit multiplexer

Multiplexers are used to create complex signal routing, such as in the ALU of processor. There are other applications as well, two of them are presented below.

Increasing the number of address bits by one doubles the number of possible inputs. Thus, 4 bit multiplexer is the result of two selector bits (Fig. 3 and Fig 4.). The truth table of 4 bit multiplexer can be found below:

S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

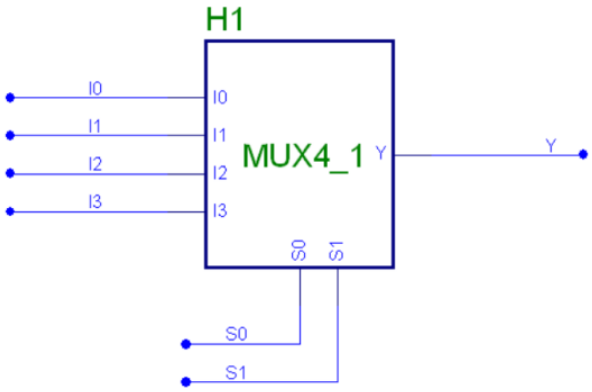

Figure 3. –Block scheme of 4 bit multiplexer

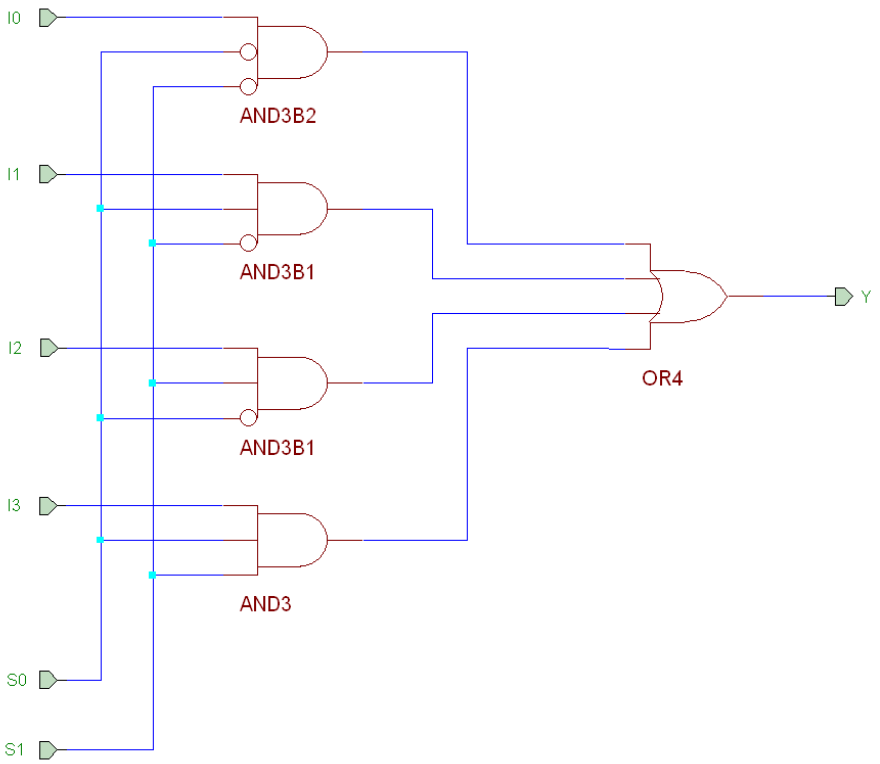

Figure 4. – Schematic of 4 bit multiplexer

Multiplexer with increased number of inputs can be designed in two ways. You can design it by using the necessary amount of gates or you can create it from multiplexers with less inputs.

Multiplexer helps in the implementation of look-up tables (see introductory lab description). The used FPGA (on the Basys panel) utilize 4 bit LUTs in its CLB to implement logic functions up to 4 variables. If this would be solved by containers, then each CLB would need 64k bit memory. Instead, it is enough to use e.g. a 16 bit register with a 16 bit multiplexer. The address bits of the multiplexer are connected to 4 variables of the logic function. The software calculates, based on the schematic, which logic function is needed for implementation and it connects the 16 input bits to the given input of the multiplexer in every combination of the mentioned 4 input variables. It can be seen that such realization requires less resources.

1.2.1. Demultiplexer

The operation of the demultiplexer is the opposite of the multiplexer's operation. It has a single input and the address buts define the output where to propagate (Fig. 5., Fig. 6.). The other outputs are in inactive state. Its most important application field is computer science.

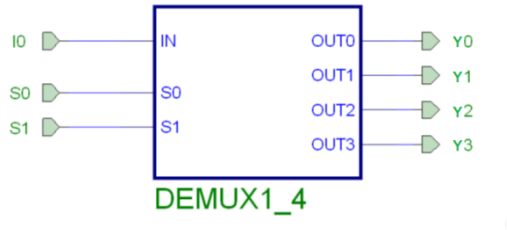

Figure 5. – 4 bit multiplexer

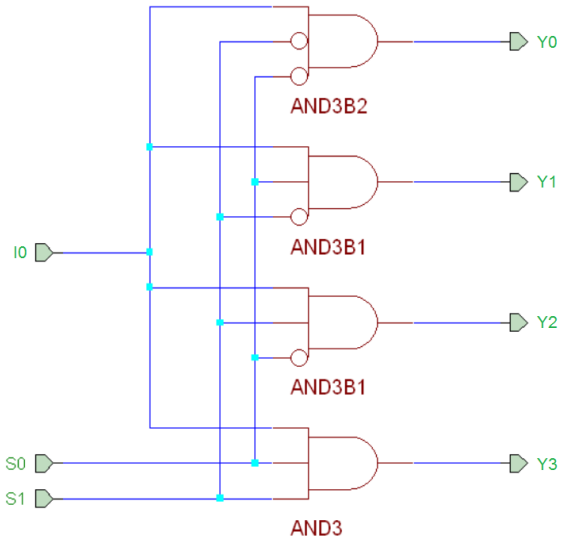

Figure 6. – Schematic of 4 bit multiplexer

1.3. Creation of the schematic during lab

Design the schematic for an 8 bit multiplexer (Fig. 7.). Give **mplxtop.sch** as the name of the top module in your project. Create the multiplexer in a separate macro (by using AND and OR gates), to do this create a new schematic (mplx.sch). Connect the data inputs of the multiplexer to the switches (SW0-SW7) on the Digilent Basys board, the address bits are controlled by a 3 bit counter. The counter can be found among the symbols by the name of CB4CE, which shall be connected to the clock signal (clk) via the BNT0 push button. The local values are corresponding to the local values of switches and counter values. Execute the simulation of the schematic.

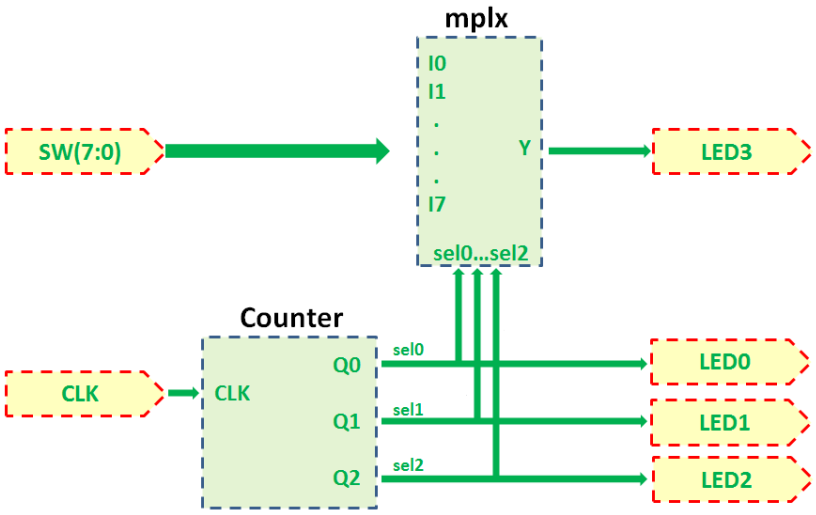

Figure 7. – Block scheme of the multiplexer

If the operation is satisfactory, then implement it on the Basys board. Test its operation with the help of switches and push buttons

Port names need for the exercise (UCF file)			
Port name	Bus	CP132 package	Description
CLK	-	NET "clk" LOC = "C11";	Clock signal for the counter, BTN0 push button
SW(7:0)	8 bit	NET "SW<7>" LOC = "N3; NET "SW<6>" LOC = "E2"; NET "SW<5>" LOC = "F3; NET "SW<4>" LOC = "G3"; NET "SW<3>" LOC = "B4; NET "SW<2>" LOC = "K3; NET "SW<1>" LOC = "L3"; NET "SW<0>" LOC = "P11";	Inputs of multiplexer
LED0, LED1, LED2, LED3	-	NET "LED3" LOC = "P6"; NET "LED2" LOC = "P7"; NET "LED1" LOC = "M11"; NET "LED0" LOC = "M5";	Status feedback LEDs for the counter: LED0, LED1,LED2 Multiplexer output: LED3

First, create the schematic of the multiplexer. Give the name mplx.sch for this schematic. Create the 8 bit multiplexer on the figure below and generate the macro file from it.

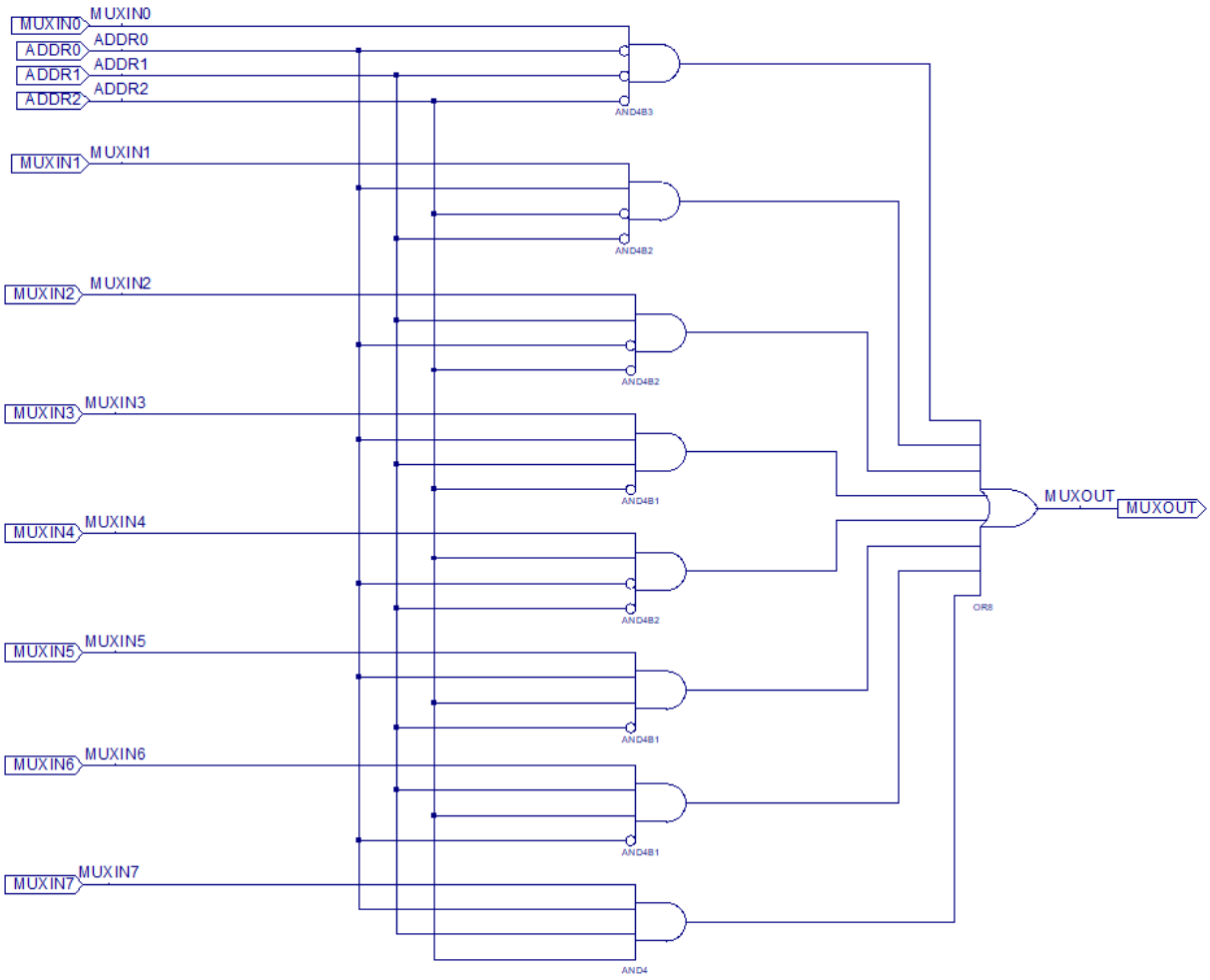

Figure 8. – 8 bit multiplexer (MACRO)

Secondly, create the counter controlling the multiplexer. Give the name Cou8.sch for this schematic. Create the counter on the figure below and generate the macro file from it. (The built-in CB4CE counter can also be used.)

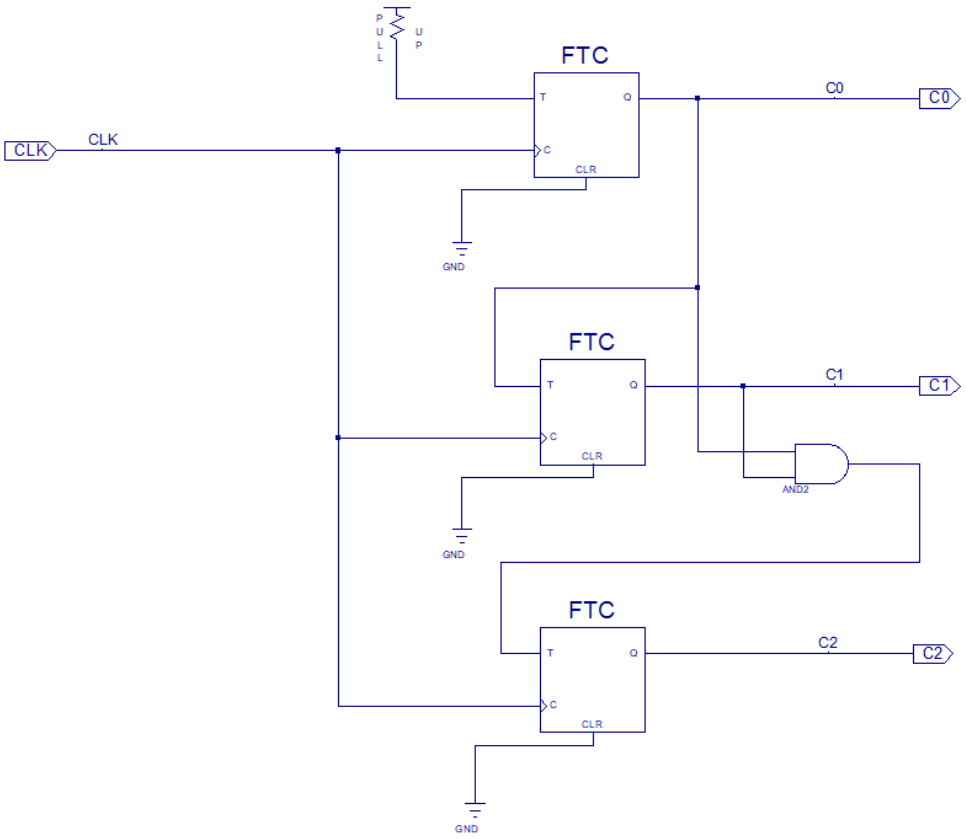

Figure 9. – 3 bit counter (MACRO)

Finally, create the schematic according the figure below:

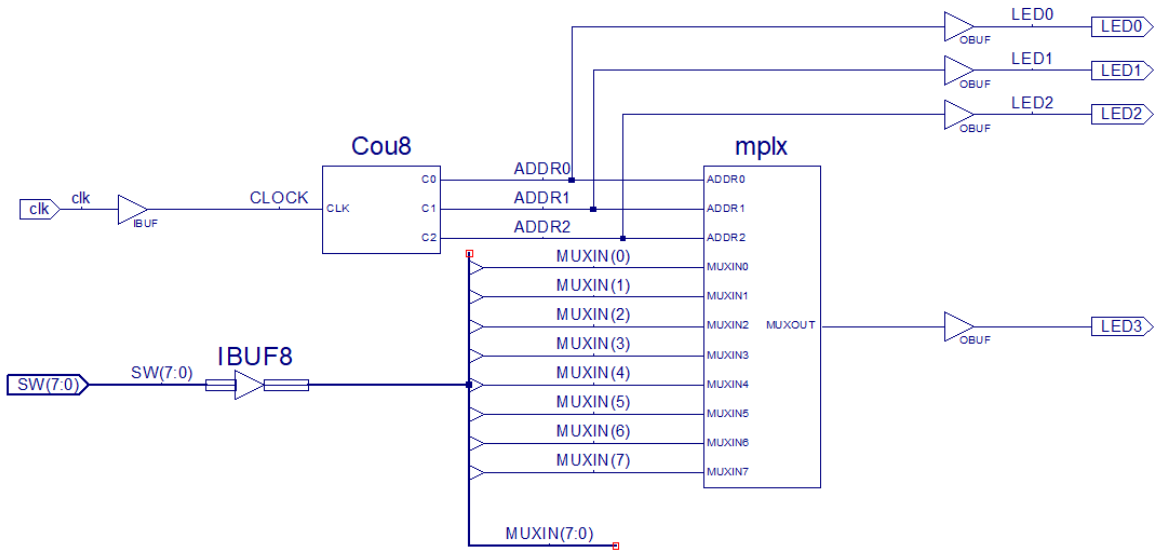

Figure 10. – Multiplexer circuit by using the previous macros

1.1. Testing

Simulate the behavior of the system as on the previous labs. Create an empty testbench and copy the content preented below (it can also be found in the moodle system as a stub file).

```
-- Clock period definitions
   CONSTANT clk_period : time := 1 us;
--*******************************************************

-- Clock process definitions
   clk_process : process
   begin
--CLK <= NOT CLK AFTER clk_period/2;
       CLK <= '0';
        wait for clk_period/2;
        CLK <= '1';
        wait for clk_period/2;
    end process;

  tb : PROCESS(CLK)
   variable sh_one : integer := 0 ;
   BEGIN
      if CLK='1' then
         sh_one := (sh_one + 1) mod 8;
         CASE sh_one is
           WHEN  0 =>
               SW <= "00000001" ;
               SW <= "11111110" after clk_period/2;
           WHEN  1 =>
               SW <= "00000010" ;
               SW <= "11111101" after clk_period/2;
           WHEN  2 =>
               SW <= "00000100" ;
               SW <= "11111011" after clk_period/2;
           WHEN  3 =>
               SW <= "00001000" ;
               SW <= "11110111" after clk_period/2;
           WHEN  4 =>
               SW <= "00010000" ;
               SW <= "11101111" after clk_period/2;
           WHEN  5 =>
               SW <= "00100000" ;
               SW <= "11011111" after clk_period/2;
           WHEN  6 =>
               SW <= "01000000" ;
               SW <= "10111111" after clk_period/2;
           WHEN  others =>
               SW <= "10000000" ;
               SW <= "01111111" after clk_period/2;
         END CASE;
       END if ;
   END PROCESS tb;
```

Simulation results can be seen on the figure below.

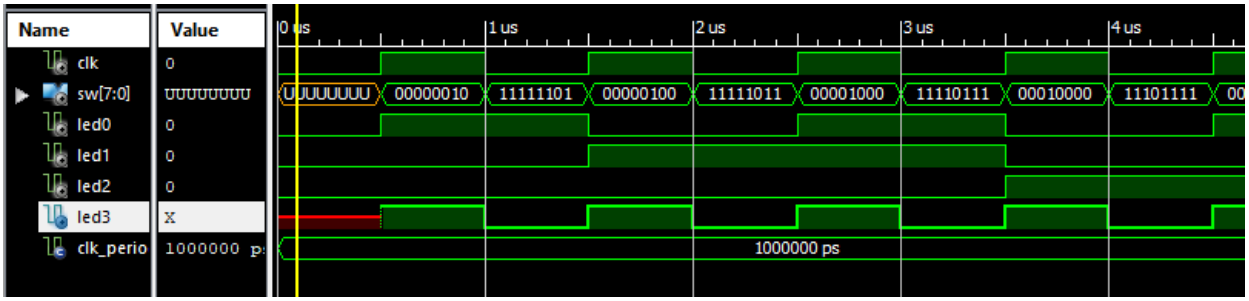

Figure 11. – Simulation of multiplexer circuit

Finally implement the circuit and analyze its operation.