Laboratory 6

Decoders. Multiplexers. Adders.

6.1 Objectives

The objectives of this laboratory are:

- Strengthening students' knowledge about combinatorial logic circuits like decoders, multiplexers and adders;
- Exercising the usage of XILINX ISE Webpack for digital system implementation, starting from schematic description;
- Testing the combinatorial logic circuits implemented on the Spartan-3E board, using the oscilloscope and the logic analyser.

6.2 Decoders

A decoder is a combinatorial circuit with **n** inputs and $\mathbf{2}^n$ outputs, which identifies an input combination by activating the corresponding output.

Figure 6.1 shows the symbol of a 2-bit decoder. Table 6.1 shows the truth table which describes the function of the 2-bit decoder.

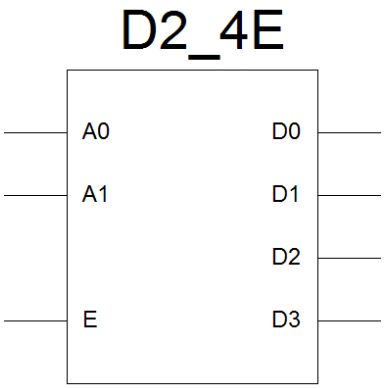

Figure 6.1 Symbol of the 2-bit decoder.

Table 6.1

The truth table of the 2-bit decoder.

E	A1	A0	D3	D2	D1	D0
1	x	x	0	0	0	0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0

Create a project in XILINX ISE having the starting point set as schematic description to study the behavior of a 2-bit decoder. Use the **D2_4E** symbol from the standard Xilinx library. For input and output interconnections use wire buses, as shown in figure 6.2.

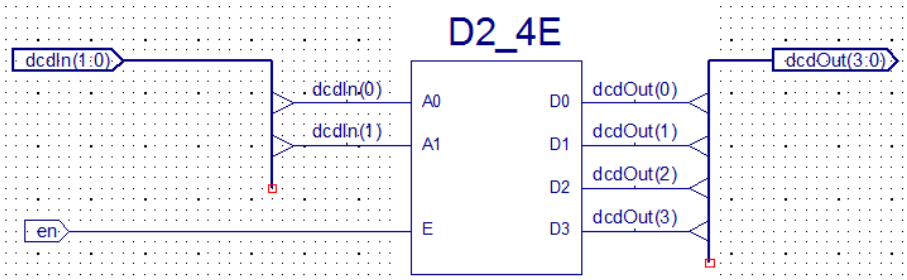

Figure 6.2 Schematic drawing showing the connections of the 2-bit decoder.

As the stimulus generator use the **genSignal8** module. The schematic file (**genSignal8.sch**) and symbol file (**genSignal.sym**) of it can be found on the website dedicated to this lab. Download the files and add them to the current ISE project.

Draw the schematic of the system which includes the symbol of the decoder (**dcd2.sym**) as well as the symbol of the stimulus generator (**genSignal8.sym**). Make sure the connections are made as shown in figure 6.3. Note the correspondence by name of the **dcdIn** wire buses.

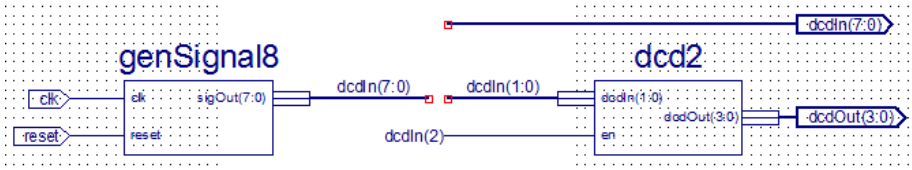

Figure 6.3 Schematic drawing of the test environment for the 2-bit decoder.

Download from the lab's website the constraint file and add it to the project (**dcd2test.ucf**). The project file for the logic analyser (**dcd2test.lpf**) can be downloaded from this location as well.

Study the behavior of the 2-bit decoder by viewing the input and output signals with the oscilloscope and the logic analyser. When using the oscilloscope the **en** signal should be chosen as the triggering source. Move the probes of channels 1, 2 and 3 between the inputs and outputs of the system. Thus, you can observe the timing correlation of the signals, taking the **en** as the reference.

Table 6.2 shows the mapping between the FPGA pins and the logic analyser channels used to test the 2-bit decoder.

Table 6.2

Mapping between the FPGA pins and the logic analyser channels for the 2-bit decoder test setup.

Port	FPGA Pin	Board connector	Channel, Connector logic analyser
en	D7	J4-I09	0, black/white
dcdin(1)	C7	J4-I010	1, brown/white
dcdin(0)	F8	J4-I011	2, red/white
dcdOut(3)	B4	J1-IO1	4, yellow/white
dcdOut(2)	A4	J1-IO2	5, green/white
dcdOut(1)	D5	J1-IO3	6, blue/white
dcdOut(0)	C5	J1-IO4	7, mauve/white

Draw the waveforms shown by the oscilloscope by taking into account the timing correlation of the signals.

Verify the operation of the reset button (**BTN South (RESET)(K17)**) using the logic analyser. How can you do that?

Download from the lab's website the Verilog files which describes the behavior of the same modules **dcd2.v** and **genSignal8.v**, as well as the source files of the simulation environment, **dcd2test.v** and **dcd2tb.v**. Run the simulation for the 2-bit decoder.

Re-implement the system, this time based on the Verilog source code.

Compare the results of the simulation with the results obtained by investigating the system with the logic analyser.

6.3 Multiplexer

The multiplexer is a combinatorial circuit which bypasses one of the $\mathbf{2}^n$ data input lines to the output, based on the value of the **n**-bit selection code.

A multiplexer may have an input signal for output validation.

Figure 6.4 shows the symbol of an 4:1 multiplexer from the Xilinx symbol library. The 6.3 table shows the truth table of the same circuit.

Table 6.3

The truth table of the 4:1 MUX with output validation.

E	S1	S0	O
1	-	-	0
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3

Create a new project in XILINX ISE to study the behavior of the 4:1 multiplexer. Use the **M4_1E** symbol from the Xilinx library. For the input/output interconnections use wire buses, as shown in figure 6.5.

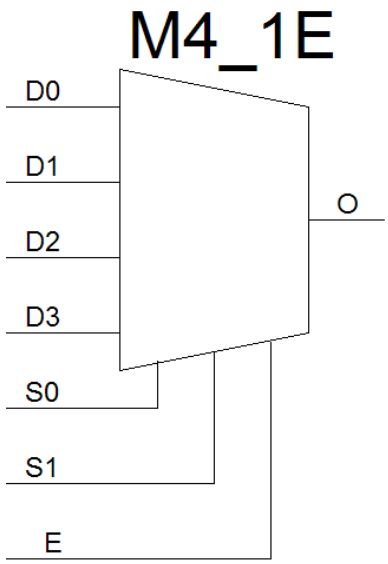

Figure 6.4 The symbol of a MUX4:1 with output validation input (XILINX ISE).

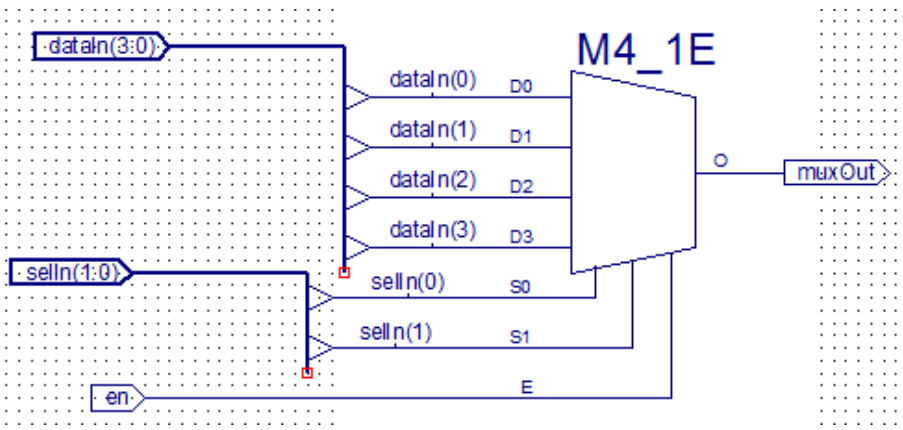

Figure 6.5 Schematic showing the interconnections of the 4:1 multiplexer.

As stimulus generator, use the **genSignal8** module. The schematic file (**genSignal8.sch**) and symbol file (**genSignal.sym**) of it can be found on the website dedicated to this lab. Download the files and add them to the current ISE project.

Draw a schematic which includes the 4:1 multiplexer (**mux4x1.sym**) and the stimulus generator (**genSignal8.sym**) as well. Make sure the connections are done as shown in figure 6.6.

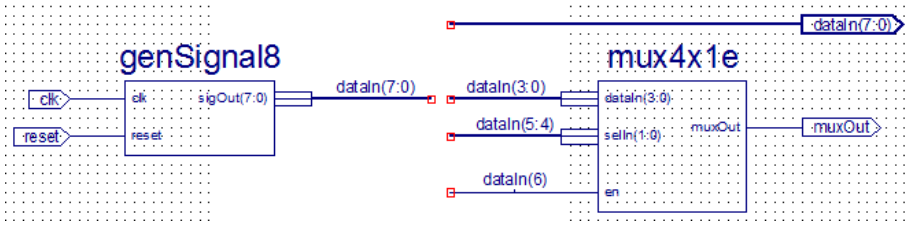

Figure 6.6 Schematic of the test environment for the 4:1 MUX.

Download the constraint file (**mux4x1eTest.ucf**) from the lab's website and add it to the project. The project file for the logic analyser (**mux4x1eTest.lpf**) can be downloaded from the same location.

Study the behavior of the multiplexer by viewing the input and output signals with the oscilloscope and the logic analyser. When using the oscilloscope the **en** signal should be chosen as the triggering source. Move the probes of channels 1, 2 and 3 between the inputs and outputs of the system. Thus, you can observe the timing correlation of the signals, taking the **en** as the reference.

Table 6.4 shows the mapping between the FPGA pins and the channels of the logic analyser for the 4:1 MUX test setup.

Table 6.4

Mapping between the FPGA pins and the channels of the logic analyser for the 4:1 multiplexer test setup.

Port	FPGA Pin	Board connector	Channel, Connector logic analyser
en	D7	J4-I09	0, black/white
selIn(1)	C7	J4-I010	1, brown/white
selIn(0)	F8	J4-I011	2, red/white
dataIn(3)	E8	J4-I012	3, orange/white
dataIn(2)	B4	J1-IO1	4, yellow/white
dataIn(1)	A4	J1-IO2	5, green/white
dataIn(0)	D5	J1-IO3	6, blue/white
muxOut	C5	J1-IO4	7, mauve/white

Draw the waveforms shown by the oscilloscope by taking into account the timing correlation of the signals.

6.4 2-bit adder/subtractor

The block diagram of an adder/subtractor circuit is shown in figure 6.7. The mapping of the ports to the FPGA pins (board) can be found in table 6.5.

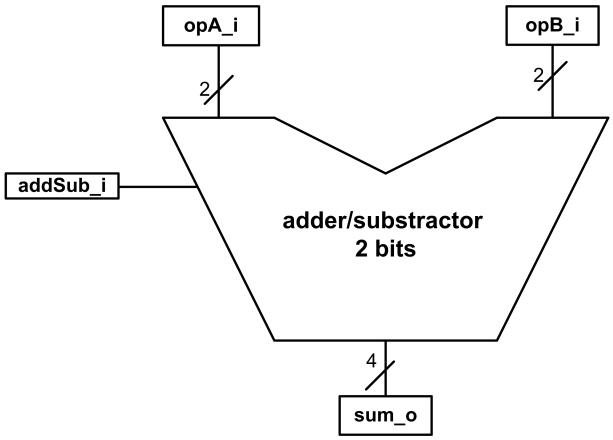

Figure 6.7 Block diagram of a 2-bit adder/subtractor.

Table 6.5

Mapping of the adder/subtractor's ports to the FPGA pins (board).

Port	Direction	Number of bits	Board location	FPGA Pin	Description
opA_i	in	2	SW1 SW0	L14, L13	First operand
opB_i	in	2	SW3 SW2	N17, H18	Second operand
sum_o	out	4	LED3-0	F11, E11, E12, F12	Result
addSub_i	in	1	BTN South	K17	Command: 0 = addition, 1 = subtraction

The adder/subtractor module can be described in the Verilog HDL as follows (you can also download it from the lab's website: **addSub.v**):

```
module addSub(
  opA_i,
  opB_i,
  sum_o,
  addSub_i
);

input[1:0]  opA_i;
input[1:0]  opB_i;
output[3:0] sum_o;
input       addSub_i;

reg[3:0]    sum_o;

always @(opA_i or opB_i or addSub_i)
if (addSub_i) sum_o <= opA_i - opB_i; else
              sum_o <= opA_i + opB_i;

endmodule
```

The mapping of the ports of the system to the the FPGA pins can be edited in the constraint file of the project (**addSub.ucf**). You can download it from the lab's website.

Download the **addSub.v** and **addSub.ucf** source files from the lab's website and create a new project in Xilinx ISE specifying "HDL" as project start-point. As an alternative, you can configure the device using the **addSub.bit** bitstream file, which can be downloaded from the lab's website. Programming the device can be done by launching the **impact** application. Launching this application with the proper parameters can be done by using the **prog.bat** file, which is also available on the lab's website.

Execute the **.bat** file with the following command run at the command prompt:

```
prog addSub
```

Configure the device with the **addSub.bit** file (you can run the implementation flow to obtain the file or you can download it from the lab's website)

Using the FPGA board with the **addSub** design loaded into it, fill in the table 6.6. Check the functionality of the system by converting the numbers from binary to decimal.

Table 6.6

Verification of the adder/subtractor circuitry.

Operation decimal	**a** binary	**b** binary	**s** binary	**s** decimal
1 + 1 = 2				
1 + 2 = 3				
2 + 1 = 3				
3 + 3 = 6				
1 - 1 = 0				
1 - 2 = -1				
2 - 1 = 1				
1 - 3 = -2				

What can you observe regarding the operations whose result should be a negative number?