Custom Computing & Embedded Systems

FPGA simulation and debug – combinational logic

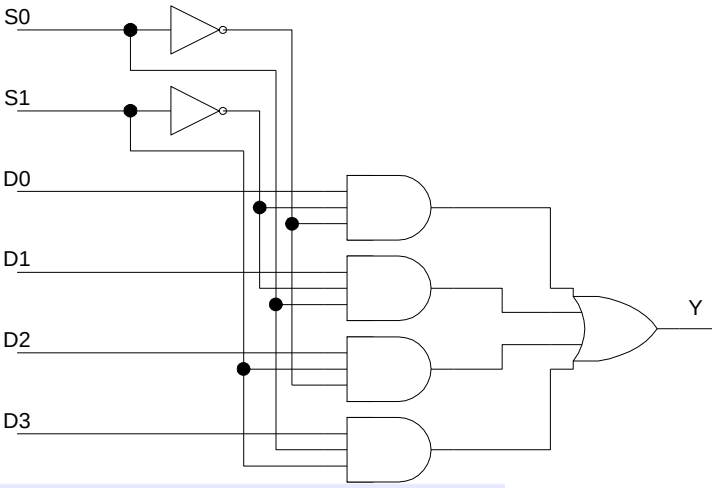
1. **Thematic scope of the exercise:**
   - description of combinational circuits in VHDL,
   - construction of structural architecture, specification of components,
   - simulation using the VHDL testbench.

2. **Modeling of combinational circuits**

The description of combinational circuits requires the use of concurrent instructions or processes with a complete sensitivity list. A characteristic feature of such a description is the lack of a clock signal (we do not use conditions controlling the signal edge in the processes).

2.1. Description of the multiplexer

A standard example of a combinational circuit is a multiplexer with a 1-bit data path. Depending on the value of the selection signal S, the appropriate input signal D is switched to the Y output.




```
5  entity mux4x1 is
6      Port ( d0,d1,d2,d3 : in std_logic;
7             s0,s1 : std_logic;
8             y : out std_logic);
9  end entity mux4x1;
```

The system from the above diagram can be described using Boolean equations (it is then necessary to know the structure of the system).

```
11 architecture equation of mux4x1 is
12 begin
13 y <= (d0 and (not s1 and not s0) ) or
14      (d1 and (not s1 and     s0) ) or
15      (d2 and (    s1 and not s0) ) or
16      (d3 and (    s1 and     s0) );
17 end architecture;
```

A simpler way to describe it is to use a concurrent conditional or selection statement (we don't need to know the structure - just knowledge of the operation).

```
19  architecture behav of mux4x1 is
20  signal sel: std_logic_vector(1 downto 0);
21  begin
22  y <= d0 when (sel="00") else
23       d1 when (sel="01") else
24       d2 when (sel="10") else
25       d3 ;
26  sel <= s1 & s0;
27  end architecture;
```

```
33  architecture selection of mux4x1 is
34      signal sel : std_logic_vector(1 downto 0);
35  begin
36      with sel select y <=
37          d0 when "00",
38          d1 when "01",
39          d2 when "10",
40          d3 when others;
41
42      sel <= s1 & s0;
43  end architecture selection;
```

[!] Create a new Questa simulator project and add .vhd from the course server

[!] Add a new mux4x1.vhd file to the project
Describe the operation of the circuit from the diagram, using any architecture.
Compile from the command line with the command:

```
vcom -work work -2008 mux4x1.vhd
```

After removing syntax errors, simulate with the command:

```
vcom -2008 -autoorder *.vhd
vsim -voptargs=+acc work.mux4x1_tb
```

Verify the correct operation of the multiplexer by observing the time waveforms

2.2. Code converters

Code converters are also implemented as combinational circuits. The simplest way to describe it is to use a concurrent select operation. The following example describes the operation of a 7-segment code decoder based on the truth table. In the presented case, the code output (sseg) is active low

```
42  entity dec7seg is
43      Port ( bcd : in std_logic_vector(3 downto 0);
44             seg : out std_logic_vector(6 downto 0)); --low
45  end entity dec7seg;
46
47  architecture ttable of dec7seg is
48  begin
49   with bcd select                    --        0
50    seg<= "1111001" when x"1",        --       -----
51          "0100100" when x"2",        --    5|     |1
52          "0110000" when x"3",        --     |  6  |
53          "0011001" when x"4",        --       -----
54          "0010010" when x"5",        --    4|     |2
55          "0000010" when x"6",        --     |     |
56          "1111000" when x"7",        --       -----
57          "0000000" when x"8",        --         3
58          "0010000" when x"9",        --
59          "1000000" when x"0",        --
60          "0111111" when others;      --
61  end architecture ttable;
```

[!] Add a new file dec7seg.vhd to the project
Describe in it the operation of the 7-seg code decoder with the output active low and displaying characters in the range '0'..'F'
Compile from the command line with the command:

```
vcom -work work -2008 dec7seg.vhd
```

After removing syntax errors, simulate with the command:

```
vcom -2008 -autoorder *.vhd
vsim -voptargs=+acc work.dec7seg_tb
```

Verify the correct operation of the decoder by observing the time waveforms

3. Component instantiation

A component substitution (instantiation) instruction is a concurrent instruction specifying the interconnection of signals and ports of components. There are two ways to assign signals:

a) positional association,
b) name association.

example :

```
port map(d, clk, rst, q); --  port positional association
 map(clk => clk_s, rst => rst_s, d => d_s, q => q_s); -- name association
```

According to the 1987 standard, a component should first be declared in an architecture block or package. Then you can concretize it in the architecture body:

syntax :

```
-- scope of architecture declaration
component <entity_name>
      port (...);
end component ;
-- architecture body
<label> : <entity_name> port map (...);
```

example :

```
-- scope of architecture declaration

component dff
      port (...);
end component ;
-- architecture body
Flip_flop : dff port map (...);
```

Direct instantiation does not require prior declaration of the component in the declarative scope of the architecture.

syntax :

```
-- architecture body
<label> : entity <libraryname>.<entityname> ( <architecture> )
            port map (...);
```

example :

```
-- architecture body
Flip_flop : entity work.dff(flop) port map (...);
```

4. Construction of a structural description

To implement the structural architecture, knowledge of the internal structure of the modeled device is required. The structured description consists of instructions for component assignments and definitions of connections between embedded modules. Connections must be made using internal **signals** whose type corresponds to the type of ports in the connected modules.

Task A

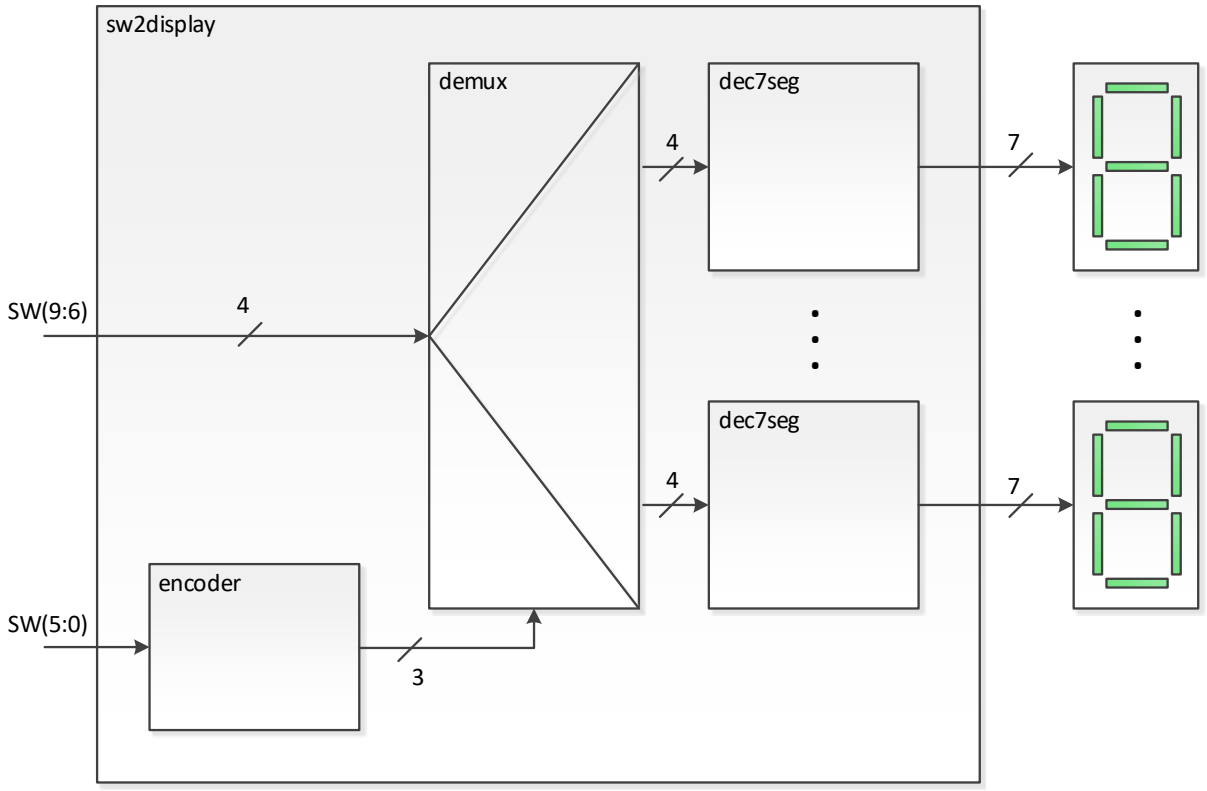
[!] Add a new file called **sw2display**.vhd to the project
Define a design unit with the interface given below

```
10  entity sw2display is
11    port(sw: in std_logic_vector(9 downto 0);
12         hex5,hex4,hex3,hex2,hex1,hex0: out std_logic_vector(6 downto 0) );
13  end entity;
```

[!] Design a structural architecture enabling the display of hexadecimal digits
on any 7-segment display of the DE10Lite board.
Use blocks designed during classes or design your own if necessary (e.g. demultiplexer).

Divide the input SW vector into two parts:
SW(9:6) – value of the displayed digit
SW(5:0) – display address in 1zN code

[!] Add a new file called **sw2display_tb**.vhd to the project
Define a simple testbench to verify the operation of the designed device
Prepare a compilation/simulation macro and demonstrate how the system works



Task B

[!] Implement the device from Task A on the **DE10-Lite** prototype board.
Use the tutorial " Quartus Prime - quick start".
[!] Present the results of the operation along with the generated RTL diagram of your system (example below) to the teacher. Verify that no latches occurred as a result of the synthesis!