Lecture 4: Seven Segment Display

G. Kemnitz[*], TU Clausthal, Institute of Computer Science

May 25, 2011

The test board has a four digit seven segment display with combined cathode signals and a common anode signal per digit, so that in every time interval only one digit can be displayed. To visualize multiple digits the single digits has to be displayed multiplex, which means one after the other. In this lecture such a display, where the digits are set by buttons will be developed step by step.

1 Test of the seven segment display

The seven segment display on the test board has four digits. The cathode signals of all segments *a* to *g* and *dp* are grouped together to save wires. The four anode signals »AN0« to »AN3« of each digit are separately available and inverted by transistors between the FPGA and the display (figure 1). To turn on a display element the corresponding cathode signals, which are *a*, *b*, ... and *dp*, have to be set to »0« and the corresponding anode signal »AN0«, »AN1«, ... or »AN3« has to be set also to »0.

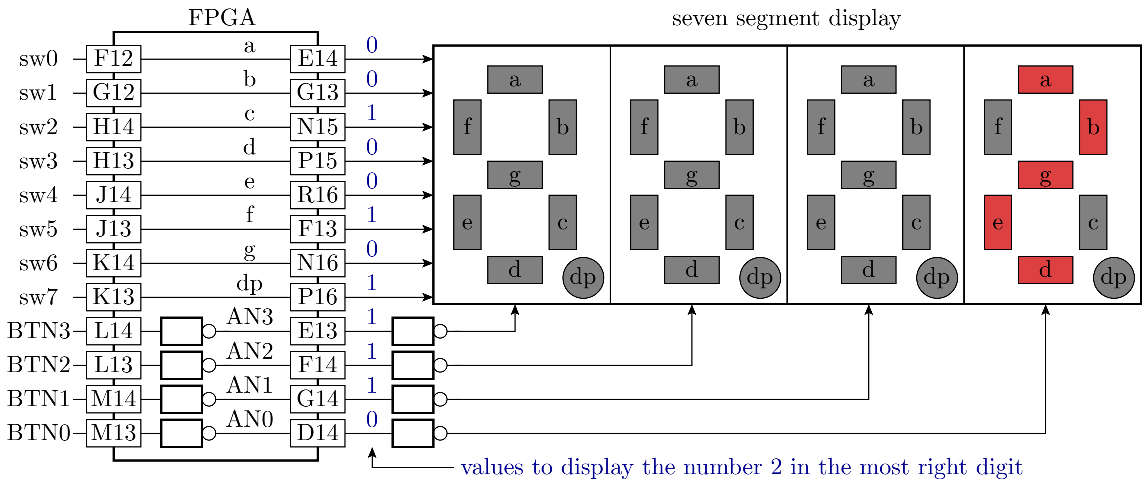


Figure 1: Test of the seven segment display

In the first experiment as shown in figure 1, the cathode signals are provides by switches and the anode signals by buttons and inverters. Design an appropriate architecture description to the following entity description

```
entity Test1_Seg7 is
 port(sw:  in STD_LOGIC_VECTOR(7 downto 0);
      btn: in STD_LOGIC_VECTOR(3 downto 0);
      a, b, c, d, e, f, g, dp, AN0, AN1, AN2, AN3: out STD_LOGIC);
end entity;
```

[*]Tel. 05323/727116

and the constraint file. Program the circuit in the FPGA on the test board and test it. Fill in the truth table at the handout sheet for the seven segment decoder in figure 2 and test it with the programmed circuit.

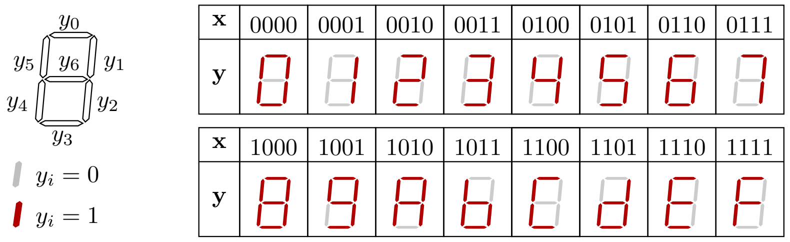


Figure 2: Function of the seven segment decoder

2 Design of the seven segment decoder

Design a seven segment decoder as a package function:

```
library ieee;
use ieee.std_logic_1164.all;
package Seg7_pack is
 function DecSeg7(x: STD_LOGIC_VECTOR(3 downto 0)) return STD_LOGIC_VECTOR;
end package;
package body Seg7_pack is
 function DecSeg7(x: STD_LOGIC_VECTOR(3 downto 0)) return STD_LOGIC_VECTOR is
  variable y: STD_LOGIC_VECTOR(6 downto 0);
 begin
  <complete the function description>
  return y;
 end function;
end package body;
```

Test the package function by simulation with the following testbench, which calls the function after another with all combinations of the input vector and prints the results on the screen:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.Seg7_pack.DecSeg7;
library Tuc;
use Tuc.Ausgabe.all;
use Tuc.Numeric_sim.all;
entity TestDecSeg7 is  end entity;
architecture a of TestDecSeg7 is
begin
 process
  variable x: STD_LOGIC_VECTOR(3 downto 0):="0000";
  variable y: STD_LOGIC_VECTOR(6 downto 0);
 begin
  while not x="1111" loop
   y := DecSeg7(x);
   write("x=" & str(x) & " y=" & str(y));
   x := x+1;
  end loop;
  wait;
  end process;
end architecture;
```

Test the same function »DecSeg7« by embedding it in the circuit in figure 3, synthesize the circuit and downloading it into the FPGA.

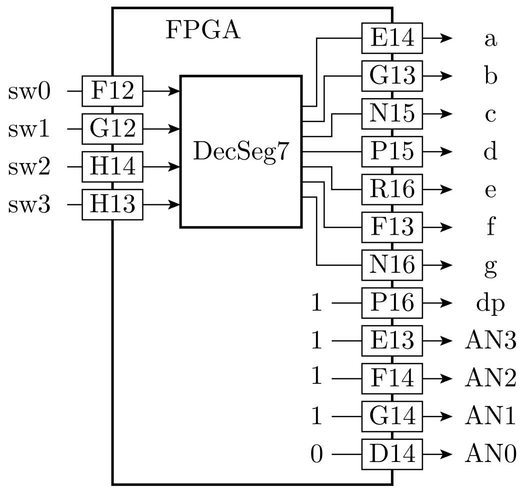


Figure 3: Test circuit for the seven segment decoder

3 Multiplex display

At the same time only one digit can be displayed. Multiple digits must be displayed cyclic after another. In the circuit in figure 4 the 50MHz input clock is scaled down by a 10 bit binary counter, i.e. by $2^{10} = 1024$. A clock divider has already been used in the previous lectures. It is described by a process with the 50MHz clock in the sensitivity list, in which with each rising edge of the input clock a counter register is increased by one and on overflow the output clock is inverted. The down scaled clock is the sample clock of the process, producing the 4-bit sliding zero vector for the anode signals. The rest of the circuit should be described in a combinatorial process. If the two left display elements are selected, the constants »1110« and »1010«, which are converted in the characters »E« and »A«, respectively, should be displayed. The values for the right digits should be selectable via the switches and transformed by the seven segment decoder into the anode signals a to g. Combinatorial process means that all input signals, here the signals from the switches and the anode signals, must be in the sensitivity list. Sampling of the asynchronous switching signals is in this special case unnecessary for the bouncing does inf fact disturb the output signals, but not in a perceptible way.

4 Via buttons selectable display values

In the next design, the four 4-bit hexadecimal values to be displayed should be provided by counters. There should be one counter per digit controlled by own push button each as shown in figure 5. The button signal is sampled as in the lecture before by a 2-bit shift register. If the button is pressed and was released in the sample step before, the counter value will be increased by one, circularly. Circular means that the next step after »1111« is »0000«. The debouncing clock frequency should be about 50 Hz. The circuit in figure 5 should be described as a separate design unit and simulated with the input waveform on the handout sheet. Complete the sketch on the handout sheet withe the simulation results.

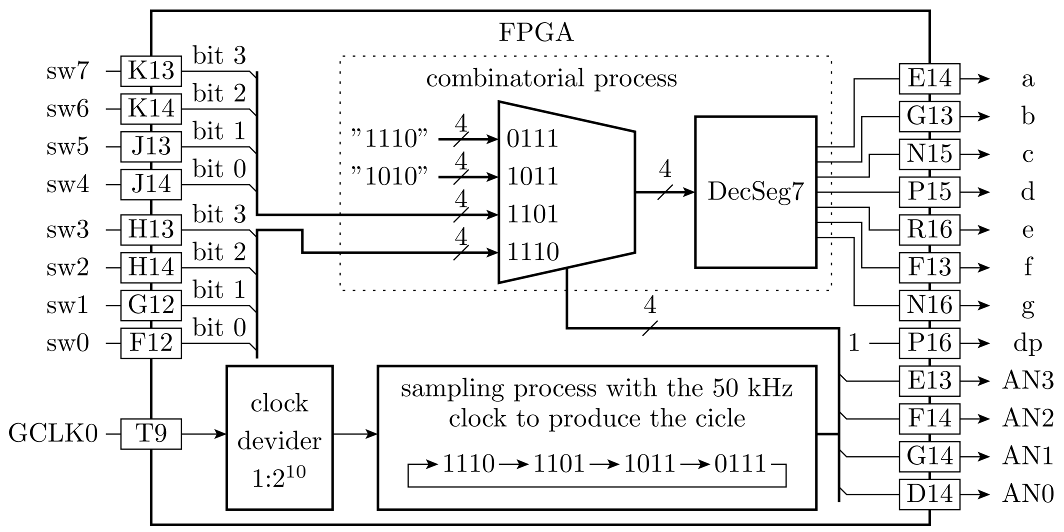
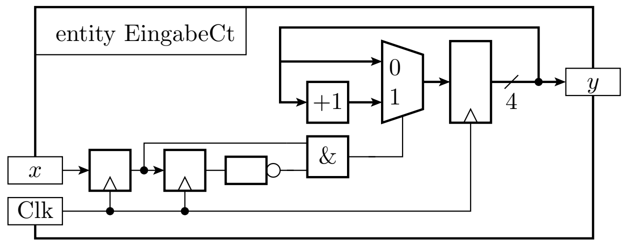


Figure 4: Circuit of the multiplex display



x	button signal
y	4-bit counter value
Clk	clock (ca. 50 Hz)

Figure 5: Counter units for setting a 4-bit value by a button

Next, four instances of the counter unit in figure 5 should be inserted in the previous description in figure 4 to set the displayed values by the buttons. Figure 6 shows the complete circuit, that should be designed, programmed, tested and presented to the supervisor[1]

[1]Keep also the programming files of the circuits in the figures 1, 3 and 4, to be also able to present it on request to the supervisor.

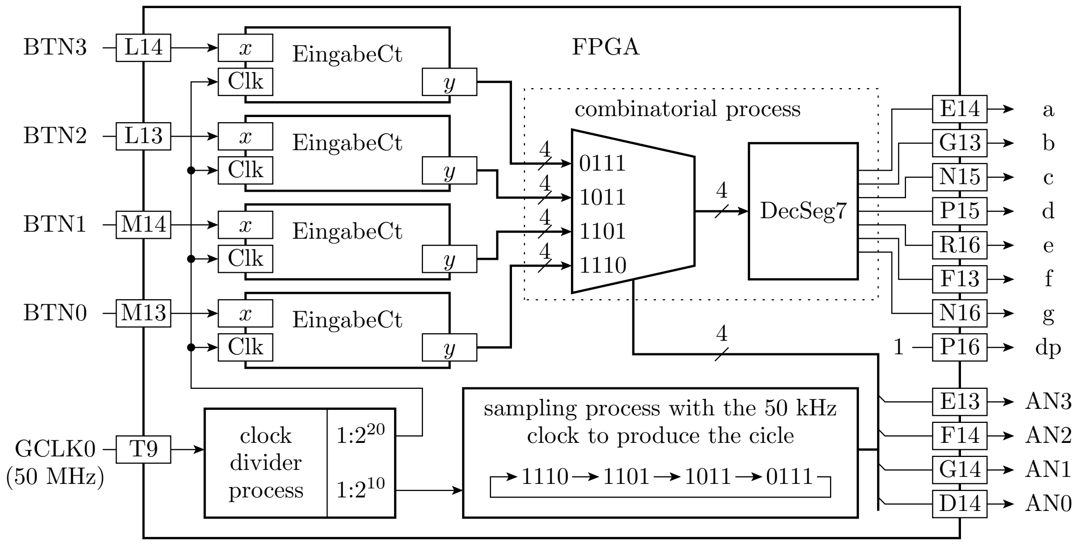


Figure 6: Multiplex display with counter units to set the display values

5 Check list for the compliance test

Exercise 1:

- presentation of the circuit description to figure 1
- filled in truth-table of the seven segment decoder on the hand-out sheet

Exercise 2:

- presentation of the simulation and the correct operation of the final downloaded circuit in figure 3

Exercise 3:

- presentation of the operation of the final downloaded circuit to figure 4

Exercise 4:

- simulation results on the handout sheet to figure 5
- presentation of the final downloaded circuit in figure 6