Laborversuch 2

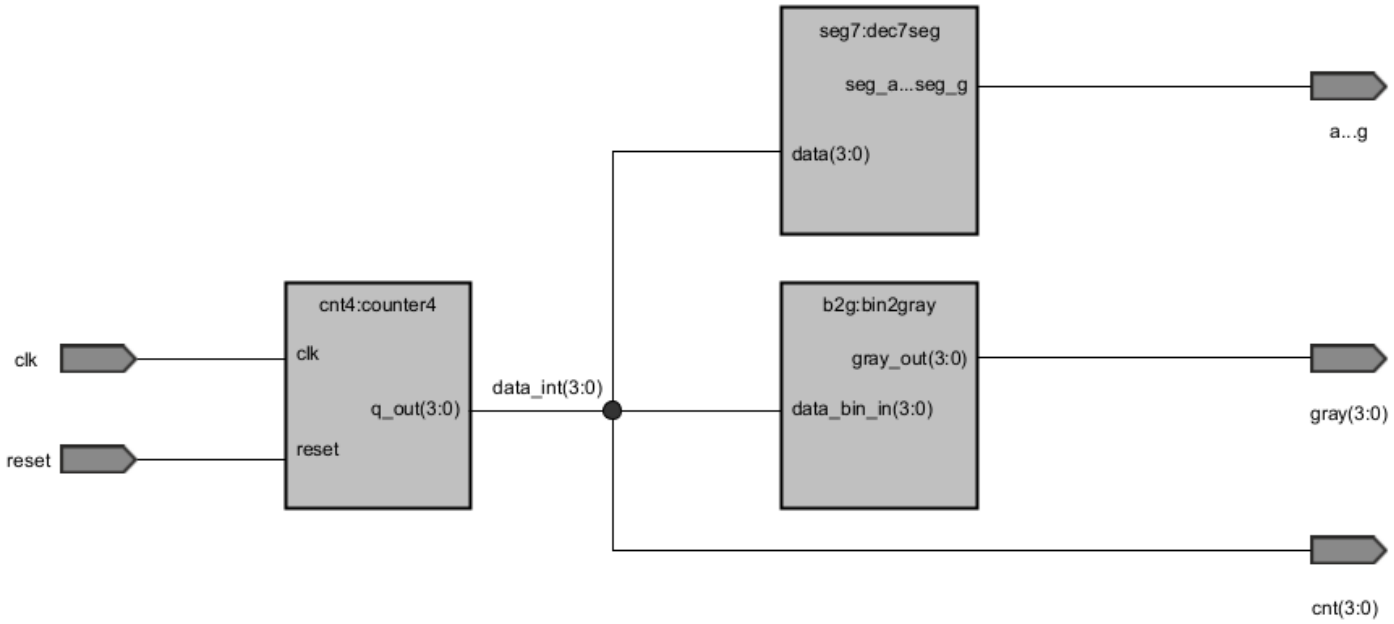
VHDL-Strukturbeschreibungen / Logiksynthese

Vorbereitungsaufgaben :

- *VHDL-Quellcode der Datei dec7seg_c.vhd*

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity dec7seg_c is
5       Port ( data  : in std_logic_vector (3 downto 0);
6              seg_a : out std_logic;
7              seg_b : out std_logic;
8              seg_c : out std_logic;
9              seg_d : out std_logic;
10             seg_e : out std_logic;
11             seg_f : out std_logic;
12             seg_g : out std_logic);
13  end dec7seg_c;
14
15
16  architecture Behavioral of dec7seg_c is
17
18
19  begin
20
21      process(data)
22          variable seg7 : std_logic_vector(6 downto 0);
23      begin
24          case data is
25              when "0000" => seg7 := "0000001";
26              when "0001" => seg7 := "1001111";
27              when "0010" => seg7 := "0010010";
28              when "0011" => seg7 := "0000110";
29              when "0100" => seg7 := "1001100";
30              when "0101" => seg7 := "0100100";
31              when "0110" => seg7 := "0100000";
32              when "0111" => seg7 := "0001111";
33              when "1000" => seg7 := "0000000";
34              when "1001" => seg7 := "0000100";
35              when "1010" => seg7 := "0001000";
36              when "1011" => seg7 := "1100000";
37              when "1100" => seg7 := "1110010";  --old value : 0110001
38              when "1101" => seg7 := "1000010";
39              when "1110" => seg7 := "0110000";
40              when "1111" => seg7 := "0111000";
41              when others => seg7 := "0000001";
42          end case;
43
44          seg_a <= seg7(6);
45          seg_b <= seg7(5);
46          seg_c <= seg7(4);
47          seg_d <= seg7(3);
48          seg_e <= seg7(2);
49          seg_f <= seg7(1);
50          seg_g <= seg7(0);
51
52
53      end process;
54
55  end Behavioral;
56
57
58
```

- *<u>Struktur-Blockschaltbild der Gesamtschaltung aus 4-Bit-Zähler, Siebensegment-Dekoder und Gray-Kodierer:</u>*



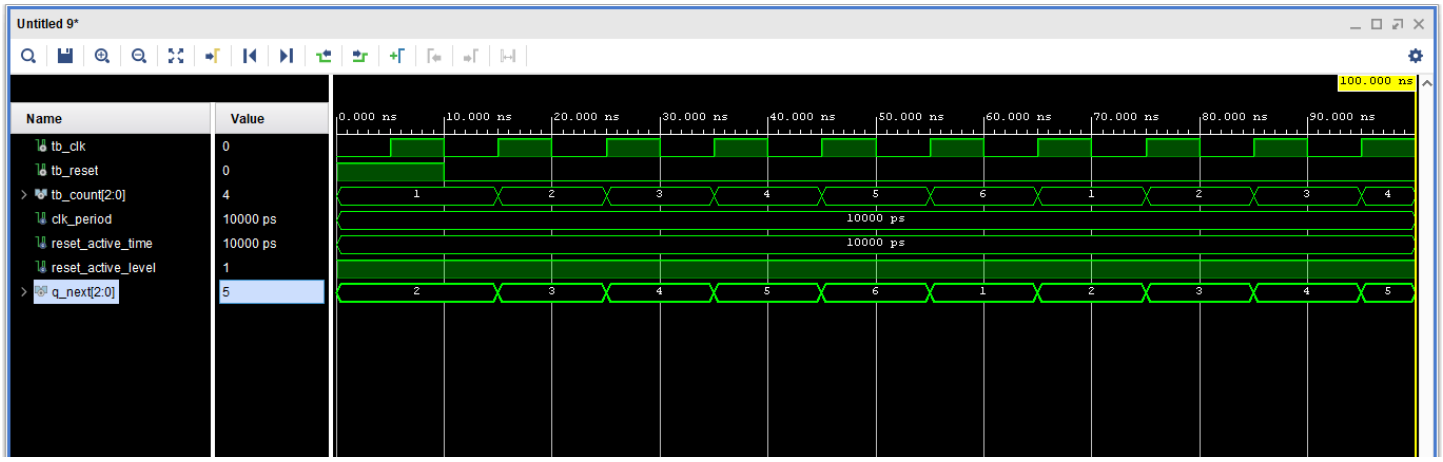
- *<u>1-bis-6-Zähler</u>*
    - *Spezifikation*

Reset	cnt_t *(2:0)* *Binärcode, 1-aktiv*	cnt_{t+1} *(2:0)* *Zustandswechsel mit steigender Taktflanke*
1	001	Nicht definiert
0	cnt_t	**If** $cnt_t = 110_2$ ***then*** 001_2 **else** $cnt_t + 1$ (von 1 bis 6 zählen)

- *Testplan*

Testumfang	*count1to6* Zähler	
Eingangssignale	Reset clk	- Der Resetsignal ist 1-aktiv, asynchron und setzt den Zähler auf 1 zurück - Systemtakt , 100 MHz - Zählerstand ändert sich mit der steigenden Taktflanke
Ausgangssignale	cnt(2:0)	- Zählerstand , binär - Kleinste Wert : 001_2 - Größter Wert : 110_2
Testsequenz / Testdauer	1. Reset 2. Durchlaufen aller Zustandsübergänge	1 Taktzykus 6 Taktzyklen

- *Screenshot der Signalverläufe in der Simulation*



- *Constrains-Datei constr.xdc*

```
1  set_property PACKAGE_PIN U16 [get_ports {count[0]}]
2  set_property PACKAGE_PIN E19 [get_ports {count[1]}]
3  set_property PACKAGE_PIN U19 [get_ports {count[2]}]
4  set_property PACKAGE_PIN A16 [get_ports clk]
5  set_property PACKAGE_PIN U18 [get_ports reset]
6  set_property IOSTANDARD LVCMOS33 [get_ports {count[2]}]
7  set_property IOSTANDARD LVCMOS33 [get_ports {count[1]}]
8  set_property IOSTANDARD LVCMOS33 [get_ports {count[0]}]
9  set_property IOSTANDARD LVCMOS33 [get_ports clk]
10 set_property IOSTANDARD LVCMOS33 [get_ports reset]
11 create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]
12
```

- Screenshot des Fensters Project Summary