Laborversuch 1

Einführung Laborsystem und Entwurfssoftware / RTL

Vorbereitungsaufgaben :

V3)

a) Wie reagiert der Zähler counter4 auf das Taktsignal? Reagiert der Zähler auf den Pegel oder auf eine Flanke des Taktsignals?

⇨ Im Normalfall würde der *counter4* auf eine steigende Flanke und *q* um 2 inkrementiert.
Jedoch sind syntaktische /semantische Fehler enthalten, wodurch die Funktion nicht gegeben ist.

```
21      reg_proc:process (clk)   -- process for register function
22      begin
23          if reset='1' then
24              q <= (others => '0');
25          elsif ck'event and clk = '1' then    -- rising clock edge
26              q <= q;
27          end if;
28      end process;
29
30      nsd_proc:process (q)     -- process for next state decoder
31      begin
32          q_ns <= q + 2;
33      end process;
```

Zeile 25 : Stellt sicher dass der *counter4* auf eine steigende Flanke reagiert.
Zeile 32 : das Signal *q_ns* wird um 2 inkrementiert.

b) Wie reagiert der Zähler counter4 auf das Resetsignal? Reagiert der Zähler auf den Pegel oder auf eine Flanke des Resetsignals?

⇨ Sobald Reset auf 1 gesetzt wird (Zeile 23) , wird *q* (4 Bits) auf 0 gesetzt (Ausgänge). Wenn Reset auf 0 ist wird eine andere Bedingung geprüft.

```
21      reg_proc:process (clk)   -- process for register function
22      begin
23          if reset='1' then
24              q <= (others => '0');
25          elsif ck'event and clk = '1' then    -- rising clock edge
26              q <= q;
27          end if;
28      end process;
29
```

c) Wie heißen die Signale in der Testbench (nicht die Ports der Komponente counter4!), die an den Takt- und Reseteingang des Zählers angeschlossen sind?

⇨ *tb_clk* und *tb_reset* heißen die Signale. (Zeile 33 und 34)

```
30
31  begin
32      uut: counter4 port map (
33          clk               => tb_clk,
34          reset             => tb_reset,
35          q_out(3 downto 0) => tb_q_out(3 downto 0)
36      );
37
38
39      clk_process: process
```

d) In welchem der Testbench-Prozesse wird das Takt-, in welchem das Resetsignal generiert? Beschreiben Sie die Arbeitsweise der beiden Prozesse!

⇨ Der Takt wird im *clk_process* generiert: Dort wird zu Beginn der Takt auf 0 gesetzt nach 12.5ns (halbe Periode) wird er auf 1 gesetzt und schließlich wieder 12.5ns gewartet. Danach beginnt der Prozess erneut.

```
39      clk_process: process
40      begin
41          tb_clk <= '0';
42          wait for clk_period / 2;
43          tb_clk <= '1';
44          wait for clk_period / 2;
45      end process;
46
```

⇨ Reset wird im *stimulus_process* generiert: Dort wird zu Beginn der Reset auf 0 gesetzt nach 50ns (Periode * 2) wird er auf den invertierten Wert vom Anfang (1) gesetzt. Danach wartet der Prozess, solange bis die Simulation neu startet.

```
46
47      stimulus_process: process
48      begin
49          tb_reset <= reset_active_level;
50          wait for reset_active_time;
51          tb_reset <= not reset_active_level;
52          -- ...
53          wait;
54      end process;
55
```

e) Das FPGA auf dem Laborsystem kann extern mit einer Frequenz von 100 MHz getaktet werden (siehe Dokumentation zum BASYS3-Board). Welche Periodendauer des Taktes ergibt sich daraus? Wo könnten Sie diese Periodendauer im Quellcode der Testbench einstellen?

⇨ $\boldsymbol{T = 1 / f = 1 / 100\ MHz = 10\ ns}$
Einstellungen: *constant clk_period* : time= xxx ns.
Quellcode-Abschnitt: Zeile 25

```
24
25      constant clk_period : time := 25 ns; --40 Mhz
26      constant reset_active_time : time := clk_period * 2;
27      constant reset_active_level : std_logic := '1';
28
29
```

f) Welchen Signalpegel hat das von der Testbench generierte Reset-Signal in seiner aktiven Phase? Passt der Pegel des Reset-Impulses zu den Anforderungen der VHDL Komponente, die Sie testen möchten? Wo können Sie ggf. den Signalpegel des generierten Reset-Signals ändern?

⇨ In seiner aktiven Phase hat das Resetsignal eine 0 (vor Änderung). Nein es passt nicht, da der Reset **High-aktiv** ist, somit soll er in seiner aktiven Phase den Pegel 1 haben und in seiner nicht aktiven Phase 0.

```
1       reg_proc:process (clk, reset)    -- process for register functi
2       begin
3           if reset='1' then
4               q <= (others => '0');
5           elsif clk'event and clk = '1' then  -- rising clock edge |
6               q <= q_ns;
7           end if;
8       end process;
```

Zeile 3+4 (in counter4.vhd) => beweist, dass der Reset High-Aktiv sein muss.

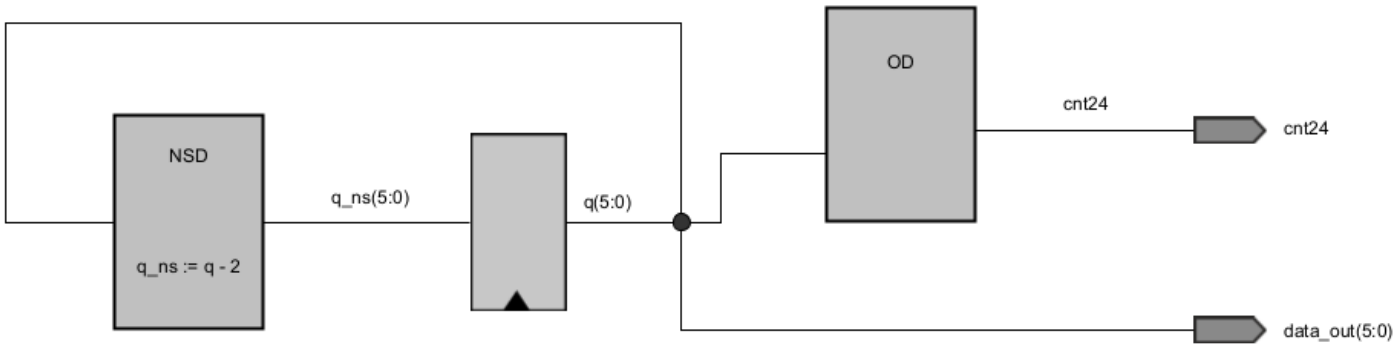
⇨ Man kann in Zeile 27 (tb_counter4.vhd) der Konstante:
constant reset_active_level einen gewünschten Wert zuweisen.

```
24
25      constant clk_period : time := 25 ns; --40 Mhz
26      constant reset_active_time : time := clk_period * 2;
27      constant reset_active_level : std_logic := '1';
28
29
```

g) Wie können Sie den Taktgeber des Peripherieboards mit einem FPGAAnschlusspin verbinden?

⇨ Um den Taktgeber des Peripherieboards mit einem FPGAAnschlusspin zu verbinden, kann man die sogenannte *Constraints (Randbedingungen)* festlegen.

V4)



Reset auf 0 , asynchron , 0-Aktiv

OD:

```
if q = 24 then
  cnt24 := 1
  else
  cnt24 := 0
  end if
```