Hochschule Karlsruhe
University of Applied Sciences
Fakultät für Elektro- und Informationstechnik

Studiengang Elektro- und Informationstechnik (Bachelor)

Versuchsvorbereitung Digitaltechnik

Versuch 7: Lasst die Würfel rollen!

von Jan Hoegen*

erstellt am 30. Mai 2022
Betreuer: Prof. Dr.-Ing. Jan Bauer

Inhaltsverzeichnis

*Matrikelnumer: 82358. E-Mail: jan.hoegen@web.de

1 Pinzuweisung

Die Abbildung 1 zeigt die Pinbelegung des FPGA für den siebten Laborversuch der Vorlesung Digitaltechnik.

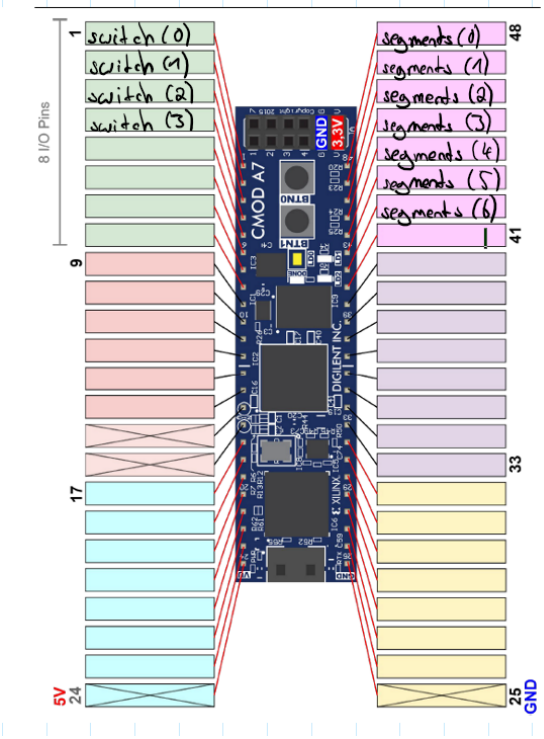

Abbildung 1: Pinbelegung des FPGA

2 Übergangsschaltnetz des Zählers

Der folgende Quellcode zeigt das Übergangsschaltnetz des Zählers. Dabei gibt *dir* die Zählrichtung an mit 1 für vorwärts und 0 für rückwärts. *Counter* ist der Zählzustand zu jedem Zyklus, *next_counter* der des folgenden Zyklus. Der Maxmimalwert wird mit *COUNTER_CEILING* angegeben. Hier wurde er auf den Wert 7 festgelegt.

vorbereitung.vhdl

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD_UNSIGNED.ALL;
4
5  ENTITY counter IS
6      PORT (
```

```
7           dir : IN STD_LOGIC; -- set counter direction. 1: forward, 0:
                backward
8           counter : OUT STD_LOGIC_VECTOR; -- increment counter for each
                cycle
9       );
10  END ENTITY;
11
12  ARCHITECTURE Behavioral OF counter IS
13      CONSTANT COUNTETR_CEILING : STD_LOGIC_VECTOR := X"7"; -- set maximum
            value for counter
14      signal next_counter : STD_LOGIC_VECTOR ;
15  BEGIN
16
17      -- Missing synchroner Teil: Folgewert übernehmen bei positiver
            Taktflanke.
18
19      -- Kombinatorik des Übergangsschaltnetzes
20      IF dir = '1' THEN
21          -- increase counter, move forward until ceiling value is reached.
                Then reset to 1.
22          IF counter >= COUNTER_CEILING THEN -- when counter is the same or
                larger than ceiling value
23              next_counter <= "001"; -- set counter to 1
24          ELSE
25              next_counter <= counter + 1; -- until then increment counter.
26          END IF;
27      ELSE
28          -- decrement counter, move backward until 1 is reached. The reset
                to ceiling value.
29          IF counter = "001" THEN -- when counter has reached 1
30              next_counter <= COUNTER_CEILING; -- set counter to ceiling
                    value
31          ELSE
32              next_counter <= counter - 1; -- until then decrement counter.
33          END IF;
34      END IF;
35
36  END ARCHITECTURE
```

3 Vervollständigen der Testbench

Der nachfolgende Quellcode zeigt die Vervollständigte Testbench für den Zähler. Es werden die Taktzyklen 9 und 10 eingesetzt.

counter_3bit_tb.vhd

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   USE std.env.finish;
4
5   ENTITY counter_3bit_tb IS
```

```
6       -- empty
7   END counter_3bit_tb;
8
9   ARCHITECTURE Behavioral OF counter_3bit_tb IS
10      COMPONENT counter_3bit
11          PORT (
12              clk : IN STD_LOGIC;
13              rst : IN STD_LOGIC;
14              enb : IN STD_LOGIC;
15              dir : IN STD_LOGIC;
16              ceil : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
17              count : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
18          );
19      END COMPONENT;
20
21      CONSTANT CLK_PERIOD : TIME := 10ns; -- Periodendauer fuer clk
22      SIGNAL clk_count : INTEGER := 0; -- Zaehler fuer durchlaufene
            Taktzyklen
23      SIGNAL clk, rst, dir, enb : STD_LOGIC;
24      SIGNAL ceil, count : STD_LOGIC_VECTOR(2 DOWNTO 0);
25  BEGIN
26      -- DUT instanziieren
27      DUT : counter_3bit PORT MAP(
28          clk => clk,
29          rst => rst,
30          enb => enb,
31          ceil => ceil,
32          dir => dir,
33          count => count
34      );
35
36      -- Taktgenerator mit Zaehler fuer Taktzyklen
37      clk_generator : PROCESS BEGIN
38          clk <= '0';
39          WAIT FOR CLK_PERIOD/2;
40          clk <= '1';
41          WAIT FOR CLK_PERIOD/2;
42          clk_count <= clk_count + 1; -- Taktzyklen zaehlen
43      END PROCESS;
44
45      -- Den Zaehler testen
46      test_process : PROCESS (clk_count) IS BEGIN
47          CASE clk_count IS
48                  ----------------------
49                  -- enb testen
50                  ----------------------
51              WHEN 0 => -- Vor der ersten Taktlanke
52                  rst <= '1', '0' AFTER 2ns; -- DUT zuruecksetzen, count ist
                        0
53                  enb <= '1'; -- enable 1 setzen
54                  dir <= '1'; -- dir 1 setzen (vorwaerts)
55                  ceil <= "111"; -- ceil auf 7 setzen
56              WHEN 2 => -- nach zweiter Taktflanke
57                  -- enb war 1, count muss 2 sein
```

```
58                  ASSERT count = "010" REPORT "enb test failed" SEVERITY
                        failure;
59                  enb <= '0'; --enb 0 setzen
60              WHEN 3 => -- nach dritter Taktflanke
61                  -- enb war 0, count muss immernoch 2 sein
62                  ASSERT count = "010" REPORT "enb test failed" SEVERITY
                        failure;
63                  ----------------------
64                  -- ceil und dir testen
65                  ----------------------
66                  rst <= '1', '0' AFTER 2ns; -- DUT zuruecksetzen, count ist
                         0
67                  enb <= '1'; -- enb auf 1 setzen
68                  ceil <= "101"; -- ceil auf 5 setzten
69                  dir <= '1'; -- dir 1 setzen (vorwaerts)
70              WHEN 9 => -- nach neunter Taktflanke
71                  -- count muss nun 0->1->2->3->4->5->1 sein
72                  ASSERT count = "001" REPORT "ceil test failed" SEVERITY
                        failure;
73                  dir <= '0'; --dir low schalten (rueckwaerts)
74              WHEN 10 => -- nach zehnter Taktflanke
75                  -- count muss nun 1->5 sein (rueckwaerts)
76                  ASSERT count = "101" REPORT "ceil test failed" SEVERITY
                        failure;
77              WHEN 11 => -- nach elfter Taktflanke
78                  finish; --sim beenden
79              WHEN OTHERS => -- nichts tun
80          END CASE;
81      END PROCESS;
82  END Behavioral;
```