RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

LEHRSTUHL INFORMATIK 11 SOFTWARE FÜR EINGEBETTETE SYSTEME

Professor Dr.-Ing. Stefan Kowalewski
Dipl.-Inform. Dominik Franke
Dipl.-Inform. Sebastian Biallas

Aachen, 14. Januar 2011
SWS: V4/Ü2, ECTS: 7

Einführung in die Technische Informatik

WS 2010/2011

Blatt 11: Musterlösung

ACHTUNG: Die Musterlösung ist ein zusätzliches Serviceangebot. Sie erhebt weder Anspruch auf Vollständigkeit noch auf Korrektheit.

Aufgabe 1: (⋆) Fehlersuche

Aufgabe 2: Implementierung

Vervollständigen Sie den VHDL-Code in Abbildung 2. Dem Timer wird ein `vorteiler` Wert übergeben, der die Clock des FPGA teilt: Zum Beispiel erzeugt bei einem Wert von 4 der Vorteilerprozess nach jeweils vier Clock-Zyklen eine steigende Flanke an einem internen Signal. Nur wenn das Signal `an` den logischen Wert 1 liefert, soll eine steigende Flanke an diesem internen Signal den Zähler um eins erhöhen, ansonsten bleibt der Zählerwert unverändert. Man benötigt also sowohl für den Vorteiler als auch für den eigentlichen Zähler eine interne Zählervariable. Das Signal `reset` setzt sowohl den Vorteiler als auch den eigentlichen Zähler zurück und sollte analog zu Abbildung 1 abgearbeitet werden. Erreicht der Zähler den Wert `max`, wird das Signal `abgelaufen` auf 1 gesetzt, bis es durch eine 1 im Signal `quittiert` wieder gelöscht wird. Bei der Bestimmung des Wertes für `abgelaufen` ist ein Zurücksetzen dominanter als ein Setzen. Gehen Sie davon aus, dass die Werte für `vorteiler` und `max` konstant sind.

Lösungsvorschlag

```
library IEEE;
use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_ARITH.ALL;
use IEEE.std_logic_UNSIGNED.ALL;

entity Timer is
  port (
    clock: in std_logic;
    reset: in std_logic;
    vorteiler: in std_logic_vector(7 downto 0);
    an: in std_logic;
    max: in std_logic_vector(7 downto 0);
    abgelaufen: out std_logic;
    quittiert: in std_logic);
```

```
end Timer;

architecture Verhalten of Timer is
signal overflow: std_logic;

begin
  process (clock, reset) -- vorteiler
  variable x: std_logic_vector(7 downto 0);
  begin
    if (reset = '1') then
      x := "00000000";
      overflow <= '0';
    elsif rising_edge(clock) then
      -- so ok, vorTEILER darf nicht 0 sein
      x := x + 1;
      if (x = vorteiler) then
        x := "00000000";
        overflow <= '1';
      else
        overflow <= '0';
      end if;
    end if;
  end process;

  process (clock, reset) -- zaehler
  variable x: std_logic_vector(7 downto 0);
  begin
    if (reset = '1') then
      x := "00000000";
      abgelaufen <= '0';
    elsif rising_edge(clock) then
      if (an = '1' and overflow = '1') then
        -- achtung: max kann auch 0 sein
        if (max = "00000000") then
          abgelaufen <= '1';
        else
          x := x + 1;
          if (x = max) then
            x := "00000000";
            abgelaufen <= '1';
          end if;
        end if;
      end if;
      if (quittiert = '1') then
        abgelaufen <= '0';
      end if;
    end if;
```

```
  end process;
end Verhalten;
```