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, 24. Dezember 2010
SWS: V4/Ü2, ECTS: 7

Einführung in die Technische Informatik

WS 2010/2011

Blatt 11: VHDL und Mikrocontroller

Ihre Lösung zu den mit (⋆) gekennzeichneten Übungen sollen Sie am **14.1.2011** in der Übung abgeben. Die Bearbeitung der Aufgaben in Lerngruppen ist sinnvoll. Bitte geben Sie nur eine Lösung pro Lerngruppe ab.

Aufgabe 1: (⋆) Fehlersuche

Gegeben ist der VHDL-Code aus Abbildung 1. Dieser Code soll eine Datenleitung entprellen, die an einen Taster angeschlossen ist. Bitte beachten Sie, dass auf Variablen und Signale vom Typ `std_logic_vector` sowohl Elementweise (`name(index)`) als auch als Ganzes zugegriffen werden kann. Eine Konstante vom Typ `std_logic_vector` definiert man in VHDL durch Einschließen in doppelte Anführungszeichen (`"`), wobei auf die Wortbreite zu achten ist. In VHDL ist es nicht unüblich, einer Entity einen Reset-Eingang hinzuzufügen. Ist dieser Eingang gesetzt, führt er in der Regel zu einem asynchronen (d. h. ohne Beachtung der Clock) Reset aller internen Variablen und Signale. Dies wird für gewöhnlich nur nach dem Einschalten einer Komponente gemacht, um einen wohldefinierten Zustand zu garantieren.

a) Das Programm enthält vier einfache Fehler (erstrecken sich jeweils über eine Zeile). Finden und korrigieren Sie diese.

b) Nach Korrektur der einfachen Fehler kompiliert das Programm zwar, erfüllt seine Aufgabe jedoch nicht richtig. Finden und korrigieren Sie die Fehler.

c) Vervollständigen Sie den Signalverläuf in Abbildung 2 für die korrigierte Version des Codes. Gehen Sie davon aus, dass die Komponente in der Vergangenheit ordnungsgemäß resettet wurde und seit einiger Zeit an `eingang` der Wert Eins anliegt, so dass `puffer` voller Einsen ist. Vernachlässigen Sie die Ausführungszeit des Prozesses: zeichnen Sie Signalausgangsänderungen auf den Hilfslinien ein.

Aufgabe 2: Implementierung

Vervollständigen Sie den VHDL-Code in Abbildung 3. 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`

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

entity Signalprozessor is
  port (
    clock: in std_logic;
    reset: in std_logic;
    eingang: in std_logic;
    ausgang: in std_logic);
end Signalprozessor;

architecture Verhalten of Signalprozessor is
signal puffer: std_logic_vector(4 downto 0);
signal aenderung: std_logic;
begin

  process (clock, reset, eingang)
  begin
    if (reset = '1') then
      puffer <= "0000";
      ausgang := '0';
    elsif rising_edge(clock) then
      for I in 4 downto 1 loop
        puffer(I) <= puffer(I-1);
      end loop;
      puffer(0) := eingang;

      aenderung <= '0';
      for I in 4 downto 1 loop
        aenderung <= aenderung or (puffer(I) xor puffer(I-1));
      end loop;
      if (aenderung = '0') then
        ausgang <= eingang;
      end if;
    end if;
  end process;

end Verhalten;
```

Abbildung 1: VHDL Debouncer

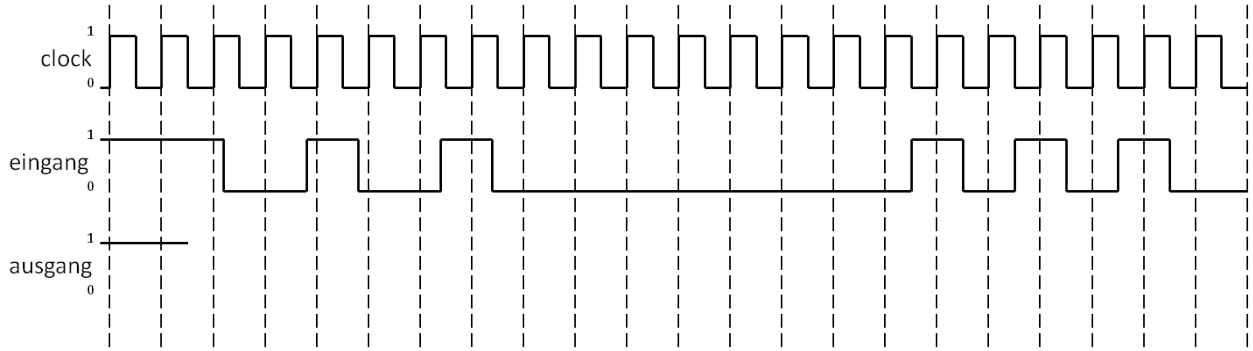

Abbildung 2: Signalverläufe

```
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
-- eigene Signale

begin
  process (clock, reset) -- vorteiler
  -- eigene Variablen
  begin
    -- eigener Code
  end process;

  process (clock, reset) -- zaehler
  -- eigene Variablen
  begin
    -- eigener Code
  end process;
end Verhalten;
```

Abbildung 3: VHDL Timer

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.

Aufgabe 3: ($\star$) **Mikrocontroller**

a) Beschreiben Sie den Unterschied zwischen einem Mikrocontroller und einem Mikroprozessor.

b) Nennen Sie vier Mikrocontrollerkomponenten, die mit der Außenwelt interagieren, und erklären Sie deren Aufgaben.

c) Wozu benötigt man Pull-up-Widerstände in digitalen I/O-Ports?

d) Wie sind LEDs und Taster üblicherweise an digitalen I/O-Ports verschaltet? Was bedeutet das für den Programmierer?

e) Das Betätigen eines Tasters soll eine interne Variable inkrementieren. Nennen Sie zwei mögliche Verfahren dies zu realisieren und erklären Sie die Unterschiede.

f) Was versteht man unter Prellen (engl.: bouncing)? Wie kann man es verhindern?

g) Erklären Sie kurz die Funktionsweise des Watch-Dog-Timers.

Aufgabe 4: ($\star$) **Interrupts**

a) Wann wird eine Interrupt Service Routine (ISR) ausgeführt?

b) Welche Schritte laufen zwischen zwei Befehlen des Hauptprogramms ab, wenn eine ISR aufgerufen wird?

c) Welche Information muss beim Aufruf einer ISR zwingend von der Hardware gesichert werden?