Aufgabenblatt 2: VHDL

Motivation: Mit VHDL (Very High Speed Integrated Circuit Hardware Description Language) beschreibt man das Verhalten einer Schaltung auf der Gatter-, RTL- bis hin zur algorithmischen Ebene. VHDL ermöglicht das schnelle Entwickeln großer und komplexer Schaltungen, wie z.B. eines Mikroprozessors. VHDL basierte Tools erlauben es ein System zu simulieren, zu verifizieren und schließlich eine Netzliste zu synthetisieren. Aus der Netzliste können Masken für die Herstellung von MPGAs (mask programmable gate array) oder ähnlichen LSI (Large scale integration)-Chips produziert werden oder sie kann (nach Konvertierung in einen geeigneten Bitstream) direkt in ein FPGA (Field Programmable Gate Array) oder CPLD (Complex Programmable Logic Device) geladen werden.

Problemstellung: Ziel dieser Übung ist es, sich mit der in der Vorlesung vorgestellten Syntax, und natürlich der Semantik, von VHDL-Programmen vertraut zu machen. An praktischen Beispielen sollen Sie lernen VHDL Programme zu lesen und ihre Funktion zu begreifen.

Aufgabe 1: Entity „A1“

a) Nachfolgend sehen Sie ein VHDL-Programm das eine *e*ntity „A1“ mit ihrer *a*rchitecture „A1_Behavior“ beschreibt. Durch genaueres Hinsehen lässt sich erkennen, dass die *e*ntity einen Automaten modelliert.
Handelt es sich dabei um einen **Mealy-** oder einen **Moore-Automaten**? Begründen Sie Ihre Antwort und geben Sie den Zustandsübergangsgraphen für den Automaten an.

b) Welche Funktion berechnet der Automat?

```
entity A1 is
    port ( A, CLK : in std_logic; OutP: out integer );
end A1;

architecture A1_Behavior of A1 is
type StateType is ( NULL, EINS, ZWEI, DREI, VIER);
signal CurrentState: StateType := NULL;
begin
  analyse: process (CLK)
  begin
    if( CLK'event and CLK = '1' ) then
      if( A = '1') then
        if( CurrentState = NULL ) then
          CurrentState <= EINS;
          OutP <= 1;
        elsif( CurrentState = EINS ) then
          CurrentState <= DREI;
          OutP <= 3;
        elsif( CurrentState = ZWEI ) then
          CurrentState <= NULL;
          OutP <= 0;
        elsif( CurrentState = DREI ) then
          CurrentState <= ZWEI;
          OutP <= 2;
        end if;
      else
        if( CurrentState = EINS ) then
          CurrentState <= ZWEI;
          OutP <= 2;
```

```
          elsif( CurrentState = ZWEI ) then
            CurrentState <= VIER;
            OutP <= 4;
          elsif( CurrentState = DREI ) then
            CurrentState <= EINS;
            OutP <= 1;
          elsif( CurrentState = VIER ) then
            CurrentState <= DREI;
            OutP <= 3;
          end if;
        end if;
      end if;
    end process analyse;
end A1_Behavior;
```

Aufgabe 2: Package „TSL“ und „System“

Ein Package enthält Konstrukte wie Typ- oder Objektdeklarationen und die Beschreibung von Prozeduren und Funktionen, die in mehreren VHDL-Beschreibungen genutzt werden können. Zum Beispiel kann in einem Package der zu verwendende Logiktyp (zwei- oder mehrwertige Logik) mit allen korrespondierenden Operatoren definiert werden.
In dem VHDL-Package auf der nachfolgenden Seite wird unter anderem die Funktion DO_WORD beschrieben. Als Parameter wird an die Funktion ein Wert vom Typ $WORD$ übergeben.

a) Was liefert die Funktion zurück, bzw. was berechnet sie?

b) Erläutern Sie die Funktionsweise. Welche Eigenschaft der Binärzahlen nutzt das Verfahren aus?

Aufgabe 3: Weitere Funktion

Im bereits vorgestellten Package aus der Aufgabe 3 wird auch die Funktion $DO8$ beschrieben.

a) Was berechnet diese Funktion?

b) Erläutern Sie die Funktionsweise?

VHDL Package zu Aufgabe 2 + 3

```
package TSL is
  type MVL is ('Z', '0', '1');
  type TSV is array (integer range <>) of MVL;
  subtype WORD is TSV(7 downto 0);
end TSL;

package body SYSTEM is

  function DO_WORD(COUNT:WORD)
    return WORD is
    variable A: WORD;
  begin
    A := COUNT;
    for i in COUNT'low to COUNT'high loop
      if A(i) = '0' then
        A(i) := '1';
        exit;
      else
        A(i) := '0';
      end if;
    end loop;
    return A;
  end DO_WORD;


  function DO8(A: WORD; B: WORD)
    return WORD is
    variable C: MVL := '0';
    variable S: MVL(1 to 3);
    variable NUM: INTEGER range 0 to 3 := 0;
    variable RESULT: WORD;
  begin
    for i in 0 to 7 loop
      S := A(i) & B(i) & C;
      for k in 1 to 3 loop
        if S(k) = '1' then
          NUM := NUM + 1;
        end if;
      end loop;
      case NUM is
        when 0 => RESULT(i) := '0'; C := '0';
        when 1 => RESULT(i) := '1'; C := '0';
        when 2 => RESULT(i) := '0'; C := '1';
        when 3 => RESULT(i) := '1'; C := '1';
      end case;
      NUM := 0;
    end loop;
    return RESULT;
  end DO8;

end SYSTEM;
```

Aufgabe 4: Entity „Puzzel“

In dem folgenden VHDL-Programm werden drei Prozesse A, B und C in der Architecture beschrieben. Nach genauerer Betrachtung lassen sich Abhängigkeiten unter den einzelnen Prozessen feststellen. Das in der Vorlesung vorgestellte „Zeitmodell von VHDL“ findet hier Anwendung.

a) Worin bestehen die Abhängigkeiten zwischen den Prozessen? Existiert ein von den anderen unabhängiger Prozess? Wenn ja, welcher?

b) Zeichnen Sie alle Signale (interne und externe) über eine Zeitperiode von 140ns als Prozessgraph auf.

```
entity PUZZEL is
  port ( CLK : in std_logic;
         OutP1, OutP2: out std_logic );
end PUZZEL;

architecture PUZZEL_arc of PUZZEL is
signal PipeB, PipeC: std_logic;
begin

  A : process
  begin
    PipeB <= '1';
    wait for 10 ns;
    PipeB <= '0' after 20 ns;
    wait for 10 ns;
    PipeB <= '1' after 5 ns,
             '0' after 15 ns,
             '1' after 30 ns;
    wait for 50 ns;
  end process A;

  B : process ( PipeB )
  begin
    OutP1 <= PipeB;
    OutP2 <= PipeC;
  end process B;

  C : process
  begin
    PipeC <= '0';
    wait on PipeB for 30 ns;
    PipeC <= '1';
    wait for 10 ns;
  end process C;

end PUZZEL_arc;
```