Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Lehrstuhl für Technische Informatik
Prof. Dr. Manfred Schimmler
Dipl.-Inf. Lars Wienbrandt

FPGA-Entwurf mit VHDL

Sommersemester 2011

Serie 3

Abgabe: bis Fr. 13.05. um 10 Uhr im Schrein oder per Mail an lwi@informatik.uni-kiel.de.

Aufgabe 1

Folgender VHDL-Code stellt eine synthesefähige Matrixmultiplikation dar. Warum sollte man es im Allgemeinen dennoch vermeiden eine Matrixmultiplikation auf diese Weise zu implementieren? Machen Sie Vorschläge (in Worten) zur Verbesserung.

Hinweis: Die verwendeten Datentypen `matrix_8bit` und `matrix_16bit` wurden in einem eigenem Package `types` definiert. Sie beschreiben eine Matrix aus 8-bit- bzw. 16-bit-Vektoren (`std_logic_vector(7 downto 0)` bzw. `std_logic_vector(15 downto 0)`).

```
use ieee.std_logic_unsigned.all;
use work.types.all; -- Einbinden des Packages fuer Datentypen

entity Matrix_mult is
generic (
  A_rows : positive := 8;
  A_cols : positive := 8;
  B_cols : positive := 8
);
port (
  CLK : in std_logic;
  A : in matrix_8bit(A_rows-1 downto 0, A_cols-1 downto 0);
  B : in matrix_8bit(A_cols-1 downto 0, B_cols-1 downto 0);
  C : out matrix_16bit(A_rows-1 downto 0, B_cols-1 downto 0)
);
end Matrix_mult;

architecture Behavioral of Matrix_mult is
begin

process
  variable temp : std_logic_vector(15 downto 0) := (others => '0');
begin
  wait until CLK='1' and CLK'event;

  for i in 0 to A_rows-1 loop
    for k in 0 to B_cols-1 loop
      temp := (others => '0');
```

```
      for j in 0 to A_cols-1 loop
        temp := temp + A(i,j) * B(j,k);
      end loop;
      C(i,k) <= temp;
    end loop;
  end loop;
end process;

end Behavioral;
```

10 Punkte

Aufgabe 2

Wie ist die Belegung des Signals `Q` in folgenden VHDL-Codebeispielen wenn der Eingang `I` von `'0'` nach `'1'` und im nächsten Takt wieder zurück von `'1'` nach `'0'` wechselt?

Betrachten Sie die Stellen direkt nach dem ersten Wechsel im gleichen Takt, eine Taktperiode später, zwei Taktperioden später und drei Taktperioden später.

Sollte die Belegung von `Q` nicht eindeutig sein, schreiben Sie "undefiniert" oder "`U`".

Hinweis: Der Wechsel von `I` erfolgt stets taktsynchron bei einer positiven Taktflanke.

1.

```
process
begin
  wait until CLK = '1' and CLK'event;
  if I = '1' then
    Q <= '1';
  end if;
end process;
```

1 Punkt

2.

```
process
begin
  wait until CLK = '1' and CLK'event;
  Q <= '0';
  if I = '1' then
    Q <= '1';
  end if;
end process;
```

1 Punkt

3.

```
process
begin
  wait until CLK = '1' and CLK'event;
  if I = '1' then
    Q <= '1';
  end if;
  Q <= '0';
end process;
```

1 Punkt

4.

```
Q <= I when CLK = '1' else Q;
```

1 Punkt

5.

```
signal A : std_logic;

process
begin
  wait until CLK = '1' and CLK'event;
  A <= I;
  Q <= A;
end process;
```

1 Punkt

6.

```
signal A : std_logic;

process
begin
  wait until CLK = '1' and CLK'event;
  Q <= A;
  A <= I;
end process;
```

1 Punkt

7.

```
process
  variable A : std_logic;
begin
  wait until CLK = '1' and CLK'event;
  A := I;
  Q <= A;
end process;
```

1 Punkt

8.

```
signal A : std_logic;

process
begin
  wait until CLK = '1' and CLK'event;
  Q <= I;
  A <= I;
  if A = '1' then
    Q <= '1';
  end if;
end process;
```

1 Punkt

9.

```
signal A,B : std_logic;

process
begin
  wait until CLK = '1' and CLK'event;
  A <= I;
  B <= I;
  if A = '1' then
    B <= '1';
  end if;
  Q <= B and not A;
end process;
```

1 Punkt

10.

```
signal A : std_logic_vector(2 downto 0);

process
begin
  wait until CLK = '1' and CLK'event;
  A <= A(1 downto 0) & I;
end process;
Q <= A(2);
```

1 Punkt

Aufgabe 3 – Programmieraufgabe

Entwerfen Sie einen Parallel-Seriell-Wandler und einen Seriell-Parallel-Wandler mit generischer Wortbreite anhand unten vorgegebener Entity-Deklarationen. Die Wandlung soll mit dem MSB beginnen.

Die Schnittstelle sei im folgenden beschrieben:

CLK: Taktsignal

D_in: Dateneingang. Das hier anliegende Signal soll gewandelt werden.

D_new_in: Eine "1" signalisiert, dass ein neues valides Datenwort/-bit am Eingang anliegt. Für den `Serializer` bedeutet dies, dass mit der Wandlung begonnen wird.

D_out: Datenausgang.

D_new_out: Eine "1" signalisiert, dass ein neues valides Datenwort/-bit am Ausgang anliegt.

Hinweis: Verwenden Sie intern einen Zähler.

a) Parallel-Seriell-Wandler

```
entity Serializer is
generic (
  WIDTH : positive := 8
);
port (
  CLK : in std_logic;
  D_in : in std_logic_vector(WIDTH-1 downto 0);
  D_new_in : in std_logic;
  D_out : out std_logic;
```

```
  D_new_out : out std_logic
);
end Serializer;
```

10 Punkte

b) Seriell-Parallel-Wandler

```
entity Deserializer is
generic (
  WIDTH : positive := 8
);
port (
  CLK : in std_logic;
  D_in : in std_logic;
  D_new_in : in std_logic;
  D_out : out std_logic_vector(WIDTH-1 downto 0);
  D_new_out : out std_logic
);
end Deserializer;
```

10 Punkte

c) Testbench

Testen Sie beide Einheiten mit einer selbst erstellten Testbench im ISE-Simulator. Verwenden Sie auch verschiedene Wortbreiten.

10 Punkte

Zusatzaufgabe 4 – Programmieraufgabe

Erstellen Sie eine VHDL-Einheit `DelayedSerialComm`, die beide in Aufgabe 3 erstellten Einheiten verwendet um über eine virtuelle Busleitung zu kommunizieren. Die Busleitung soll dabei eine Verzögerung einer generisch einstellbaren Anzahl von Takten besitzen.

Testen Sie Ihre Einheit.

Wieviele Takte Verzögerung entsteht in Ihrer Einheit zwischen Ein- und Ausgabe insgesamt?

```
entity DelayedSerialComm is
generic (
  BUS_DELAY : positive := 3
);
port (
  CLK : in std_logic;
  D_in : in std_logic_vector(WIDTH-1 downto 0);
  D_new_in : in std_logic;
  D_out : out std_logic_vector(WIDTH-1 downto 0);
  D_new_out : out std_logic
);
end DelayedSerialComm;
```

10 Zusatzpunkte

Die Abgabe der Programmieraufgaben soll bitte folgendermaßen erfolgen:

1. Zuerst führen Sie in ISE "`Project` → `Cleanup Project Files`" aus.
2. Danach verpacken Sie den Projektordner in ein gepacktes Archiv (`.zip`, `.tar.gz` o.ä.).
3. Senden Sie das Archiv per Mail an `lwi@informatik.uni-kiel.de`