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 4

Abgabe: bis Fr. 20.05. um 10 Uhr im Schrein oder per Mail an lwi@informatik.uni-kiel.de.

Aufgabe 1

Ist folgendes VHDL-Beispiel synthetisierbar? Begründen Sie. Welche Funktion ist hier beabsichtigt?

```
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity example1 is
generic (
  WIDTH : positive := 4
);
port (
  clk : in std_logic;
  A : in std_logic_vector(WIDTH-1 downto 0);
  B : in std_logic_vector(WIDTH-1 downto 0);
  X : out std_logic_vector(2*WIDTH-1 downto 0)
);
end example1;

architecture Behavioral of example1 is
begin

process
  variable tmp : std_logic_vector(2*WIDTH-1 downto 0);
begin
  wait until clk = '1' and clk'event;

  tmp := (others => '0');
  for I in conv_integer(B)-1 downto 0 loop
    tmp := tmp + A;
  end loop;
  X <= tmp;
end process;

end Behavioral;
```

5 Punkte

Aufgabe 2

Folgendes Beispiel benötigt schon bei geringer Wortbreite sehr viel Hardware-Ressourcen. Schreiben Sie dieses Beispiel unter starker Einsparung von Ressourcen neu. Das Verfahren soll dabei nicht verändert werden, allerdings darf Ihr Design mehrere Taktzyklen benötigen und Sie dürfen auch Signale am Port hinzufügen.

Was wird hier berechnet? Wieviele Taktzyklen benötigt Ihr Design für die Berechnung?

```
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity example2 is
generic (
  WIDTH : positive := 4
);
port (
  clk : in std_logic;
  A : in std_logic_vector(WIDTH-1 downto 0);
  B : in std_logic_vector(WIDTH-1 downto 0);
  X : out std_logic_vector(2*WIDTH-1 downto 0)
);
end example2;

architecture Behavioral of example2 is
begin

process
  variable tmp : std_logic_vector(2*WIDTH-1 downto 0);
begin
  wait until clk = '1' and clk'event;

  tmp := (others => '0');
  for I in WIDTH-1 downto 0 loop
    tmp := tmp(2*WIDTH-2 downto 0) & '0';
    if B(I) = '1' then
      tmp := tmp + A;
    end if;
  end loop;
  X <= tmp;
end process;

end Behavioral;
```

10 Punkte

Aufgabe 3 – Programmieraufgabe

Das Ziel ist es mit einer sehr minimal gehaltenen Benutzerschnittstelle eine Einheit zu konstruieren, die zwei Zahlen addieren oder subtrahieren kann. Die Steuerung dieser Einheit soll lediglich über zwei "Push Buttons" erfolgen.

a) Addieren / Subtrahieren

Erstellen Sie eine Einheit `add_sub`, die in Abhängigkeit des Signals `op_in` die Eingänge `A_in` und `B_in` entweder addiert ($A + B$ bei `op_in` $= 0$) oder subtrahiert ($A - B$ bei `op_in` $= 1$). Die Ausgabe soll mit positiver Flanke getaktet erfolgen. Die Datenbreite soll über ein `generic` variabel sein.

Benutzen Sie die Bibliothek `IEEE.NUMERIC_STD.ALL` und achten Sie auf die korrekte Belegung des Carry-Out `C_out`.

```
entity add_sub is
generic (
  DATA_WIDTH : positive := 4
);
port (
  clk : in std_logic;
  A_in : in std_logic_vector(DATA_WIDTH-1 downto 0);
  B_in : in std_logic_vector(DATA_WIDTH-1 downto 0);
  op_in : in std_logic;
  S_out : out std_logic_vector(DATA_WIDTH-1 downto 0);
  C_out : out std_logic
);
end add_sub;
```

10 Punkte

b) Zähler

Erstellen Sie einen einfachen Binärzähler, dessen Zählerstand über einen Eingang `sync_reset` taktsynchron zurückgesetzt werden kann. Der Zählerstand soll mit einem Signal `count` inkrementiert werden. Die Datenbreite soll ebenfalls generisch gesetzt werden können.

```
entity up_counter is
generic (
  COUNT_WIDTH : positive := 4
);
port (
  clk : in std_logic;
  sync_reset : in std_logic;
  count : in std_logic;
  counter_value : out std_logic_vector(COUNT_WIDTH-1 downto 0)
);
end up_counter;
```

5 Punkte

c) Gesamteinheit

Erstellen und testen Sie die in der Einleitung beschriebene Addierer-Subtrahierer-Einheit.

Die "Push-Buttons" sollen folgendes Verhalten hervorrufen. Button A wechselt bei "1" den internen Zustand. Button B inkrementiert je nach internem Zustand den Wert des ersten oder des zweiten Operanden bzw. wechselt den Operator (+ oder −).

In der Schaltung sollen vier interne Zustände möglich sein, die bei Betätigung von Button A nacheinander aktiv werden sollen:

1. `SHOW_RESULT`: In diesem Zustand hat eine Betätigung des Buttons B keine Auswirkung. Dieses soll auch der Default-Zustand nach einem Reset sein.
2. `CHANGE_A`: Button B ändert den Wert des ersten Operanden.
3. `CHANGE_OP`: Button B wechselt den Operator.
4. `CHANGE_B`: Button B ändert den Wert des zweiten Operanden.

mit folgender Schnittstelle:

clk: Taktsignal

sync_reset: Taktsynchroner Rücksetzeingang. Bei "1" sollen die Werte der Operanden zurückgesetzt, der Operator auf "Addition" gestellt und in den Default-Zustand `SHOW_RESULT` gewechselt werden.

enable_in: Sollte dieser Eingang auf "0" sein, so hat das Betätigen der Buttons keine Auswirkung.

push_A_in, push_B_in: Button-Eingänge.

val_A_out, val_B_out: Werte der Operanden.

op_out: Eingestellte Operation.

result_out, carry_out: Das Ergebnis der Operation.

state_out: Der interne Zustand in einer One-Hot-Kodierung.

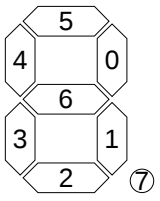
```
entity simple_calculator is
generic (
  DATA_WIDTH : positive := 4
);
port (
  clk : in std_logic;
  sync_reset : in std_logic;
  enable_in : in std_logic;
  push_A_in : in std_logic;
  push_B_in : in std_logic;

  val_A_out : out std_logic_vector(DATA_WIDTH-1 downto 0);
  val_B_out : out std_logic_vector(DATA_WIDTH-1 downto 0);
  op_out : out std_logic; -- 0 add, 1 sub
  result_out : out std_logic_vector(DATA_WIDTH-1 downto 0);
  carry_out : out std_logic;
  state_out : out std_logic_vector(3 downto 0)
);
end simple_calculator;
```

20 Punkte

Zusatzaufgabe 4 – Programmieraufgabe

Entwickeln Sie eine Einheit, die in Abhängigkeit des gewählten Zustands in der Einheit aus Aufgabe 3c entweder den Operanden A, B den Operator oder das Ergebnis auf einer 7-Segment-Anzeige anzeigt. Der Ausgang dieser Einheit soll ein 8-bit-Vektor sein, dessen Indizes für je ein Segment bzw. den Punkt der 7-Segment-Anzeige stehen. Ein gesetztes Bit entspricht einem leuchtendem Segment. Testen Sie Ihre Einheit.
Die Kodierung entnehmen Sie bitte der nebenstehenden Abbildung.



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`