DHBW
Duale Hochschule
Baden-Württemberg

Programmentwurf

für

Entwurf digitaler Systeme

TEL18
14.12.2020
Torben Mehner

Matrikelnummer: _____

Erlaubte Hilfsmittel:

- Nur ausgeteilte Formelsammlung

Wichtige Hinweise zur Durchführung der Klausur:

- Tragen Sie Ihre Matrikelnummer in das Deckblatt ein.
- Wenn Sie die Heftung lösen, müssen Sie jedes Blatt mit Ihrer Matrikelnummer kennzeichnen.
- Verwenden Sie keinen Rotstift.
- Ergebnisse werden nur gewertet, wenn der Lösungsweg ersichtlich ist!
- Bei Täuschungsversuch wird die gesamte Klausur mit der Note 5,0 bewertet.

Aufgabe Nr.:	1	2	3	4	5	6	**Summe**
Punktzahl:	8	5	5	18	8	6	50
Davon erreicht:							

Note: _____

Unterschrift
Korrektor: _____

Viel Erfolg!

Verständnis und Wissen

1. Die folgenden Fragen befassen sich mit dem in der Vorlesung übermittelten Wissen und dessen Verständnis. Eine Antwort in Stichpunkten genügt.

   (a) Beschreibe den Aufbau eines Configurable Logic Blocks (CLB). Erkläre dabei wie eine Logikfunktion abgebildet wird und wie es realisiert wird, den CLB entweder taktflankengesteuert oder asynchron zu verwenden. (3)

   Solution: Ein Configurable Logic Block besteht aus LUT, Flip-Flop und Multiplexer.
   Die Look-Up-Table bildet eine Logikfunktion ab, indem sie alle Ergebnisse in einem Speicher vorhält, die dann je nach Eingang abgerufen werden.
   Mithilfe des Multiplexers kann der CLB auswählen, ob der Ausgang der LUT direkt an den Ausgang geführt wird, oder in einen FF und somit erst bei einer Taktflanke am Ausgang ankommt.

   (b) Nenne einen Vorteil von FPGAs gegenüber Full Custom ICs. (1)

   Solution: FPGAs können rekofiguriert werden und eignen sich daher fürs Prototyping.

   (c) Nenne zwei Möglichkeiten den Quellcode in VHDL zu strukturieren. (2)

   Solution: Prozesse, Funktionen, Komponenten

   (d) Erkläre wann in VHDL Variabeln verwendet werden und wie sie sich von Signalen unterscheiden. (2)

   Solution: Variabeln werden innerhalb von Prozessen verwendet, um Zwischenergebnisse zu speichern. Im Gegensatz zu Signalen werden sie sofort geschrieben und nicht erst am Ende des Prozesses.

2. Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. Für ein richtiges Kreuz gibt es +1 Punkte. Ein falsches Kreuz bringt -1 Punkte. Wird bei einer Aussage nichts angekreuzt, gibt es keinen Abzug. (5)

   (a) Bei einem aktuellen Mehrkern-Prozessor handelt es sich um einen MISD-Prozessor.
   ◯ Wahr √ **Falsch**

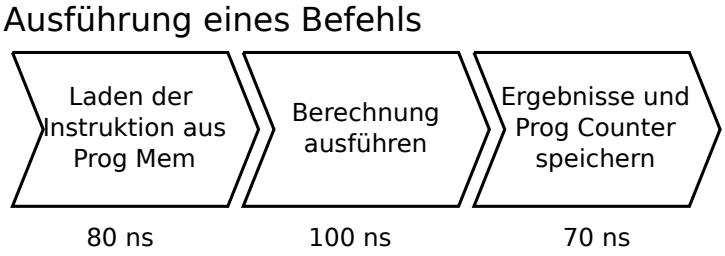
   (b) Beim Standardzellenentwurf gibt es für die meisten Standardzellen mehrere Varianten.
   √ **Wahr** ◯ Falsch

   (c) Die Cortex-M-Serie von ARM Mikroprozessoren hat die höchste Leistung.
   ◯ Wahr √ **Falsch**

(d) *Sieben_Segment_Anzeige* ist ein zulässiger VHDL-Identifier.
√ **Wahr** ◯ Falsch

(e) Bei einem STD_LOGIC_VECTOR(7 downto 0) ist das 7. Bit (Index = 7) das Least-Significant-Bit
◯ Wahr √ **Falsch**

3. Der folgende Code zeigt eine Darstellung der Abläufe in einer einfachen CPU. Nach Durchlauf der drei Abschnitte ist ein Befehl und somit ein Takt ausgeführt.



(a) Was ist die maximale Taktfrequenz dieser CPU? (1)

Solution:

$$f = \frac{1}{T} \quad (1)$$
$$= \frac{1}{250\text{ns}} \quad (2)$$
$$= 4\text{MHz} \quad (3)$$

(b) Nenne eine Möglichkeit die Taktfrequenz in diesem Fall zu erhöhen und beschreibe, wie das umgesetzt wird. Erkläre anhand eines Beispielbefehls, wo es zu Problemen kommen kann. (3)

Solution: Man kann die Taktfrequenz durch Pipelining erhöhen (1). Dafür muss die Ausführung eines Befehls auf drei Takte aufgeteilt werden, die dann parallel ausgeführt werden (1).
Bei einem Jump werden 2 falsche Befehle in die Pipeline geladen, da der Program Counter erst im letzten Schritt geschrieben wird.(1)

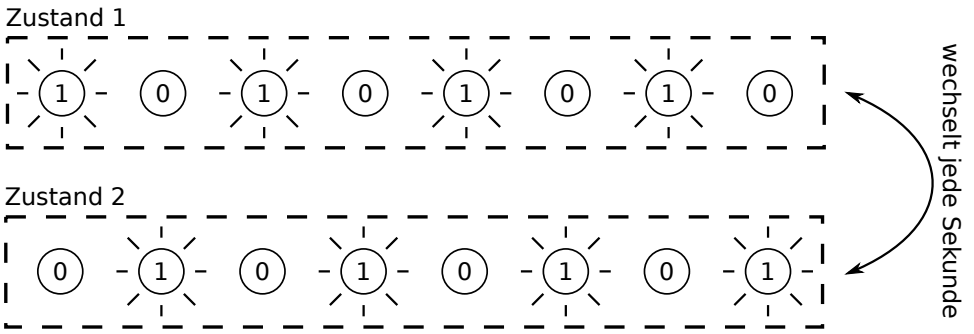
(c) Angenommen das Beheben des Problems ändert nichts am Timing der Ausführung, wie schnell kann der Takt nach der Änderung maximal sein? (1)

Solution:

$$f = \frac{1}{T_{max}} \quad (4)$$
$$= \frac{1}{100\text{ns}} \quad (5)$$
$$= 10\text{MHz} \quad (6)$$

VHDL Programmieren

4. Der weltweit größte Lichterkettenhersteller möchte eine blinkende Lichterkette mit einem Absatz von ca. 10 Millionen jährlich herstellen. Dabei sollen die Lampen durch einen zentralen IC gesteuert werden. Die Länge soll beliebig sein. Der Hersteller möchte jedoch maximal 8000 Lampen verbauen.



(a) Welche Realisierungsvariante sollte der Hersteller wählen? Begründe die Wahl (1)

> **Solution:** Full-Custom-Design aufgrund der hohen Stückzahlen

(b) Schreibe eine Entity für das VHDL-Modul `LICHTERKETTE`. (5)
An diesem Modul liegt eine Clock (`CLK`) und ein Reset (`RST`) vom Typ `STD_LOGIC` als Eingang an. Der Hersteller möchte die Lichterkette in verschiedenen Größen herstellen, weshalb der Ausgang `LICHTER` vom Typ`STD_LOGIC_VECTOR` eine variable Breite benötigt, die in diesem Beispiel durch `N` = 16 angegeben wird.

> **Solution:** Entity Syntax richtig (1) Generic richtig (1) Eingänge richtig (2) Ausgang richtig (1)

```
entity LICHTERKETTE is
    generic( N : integer := 16 );
    port( CLK : IN STD_LOGIC;
          RST : IN STD_LOGIC;
          LICHTER : OUT STD_LOGIC_VECTOR(N-1 downto 0) );
end LICHTERKETTE;
```

(c) Als CLK-Eingang wird ein Oszillator mit 32768 Hz benutzt. Die Lichterkette soll aber nur jede Sekunde blinken. Verwende die bereits angelegt Komponente `clk_div` um die Frequenz von 1 Hz zu erzeugen. Ergänze den Inhalt der Architektur, um die Blink-Funktion für eine beliebige Größe von `N` zu realisieren. Es ist egal, ob zuerst die geraden oder ungeraden Lampen leuchten. (10)

```
architecture BEHAVIORAL of LICHTERKETTE is

    component counter
        Generic (
            FREQ_IN : INTEGER;
            FREQ_OUT : INTEGER );
        Port (
            CLK_IN : in  STD_LOGIC;
            RST : in  STD_LOGIC;
            CLK_OUT : out  STD_LOGIC );
    end component;

    -- Beginne mit der Signal-Deklaration
```

Solution: Signal richtig (1) begin gesetzt (0,5) Prozess angelegt (1) Reset-Block richtig (2) CLK-Abfrage richtig (1) Algorithmus richtig (4) end Behavioral gesetzt (0,5)

```
    signal TEMP : STD_LOGIC := '0';
    signal CLK_1_HZ : STD_LOGIC;

begin

    counter_0 : counter
    generic map (
        FREQ_IN => 32768,
        FREQ_OUT => 1 )
    port map (
        CLK_IN => CLK,
        RST => RST,
        CLK_OUT => CLK_1_HZ );

    clk_proc : process( RST, CLK_1_HZ )
        variable temp_I : unsigned(31 downto 0);
    begin
        if( RST = '1' ) then
            TEMP <= '0';
        elsif( rising_edge(CLK) ) then
            TEMP <= not TEMP;

            for I in 0 to N-1 loop
                temp_I := to_unsigned(I, 32);
```

```
                LICHTER(I) <= TEMP xor temp_I(0);
            end loop;
        end if;
    end process clk_proc;

end Behavioral;
```

(d) Begründe, warum dieser Ansatz in der Praxis nicht sinnvoll ist. Wie ließe sich das sinnvoller umsetzen? (2)

Solution:

Fehlersuche

5. Beim Programmieren des Hardware-Multiplizierers meldet die Entwicklungsumgebung Syntax-Fehler in Zeile 5, 8, 11, 24 und 32.

   (a) Korrigiere die Fehler. Trage dazu die Zeile, in welcher der Fehler ist und die gesamte, korrigierte Zeile in die Tabelle ein. (Librarys wurden eingebunden und verursachen keine Fehler) (5)

   Besteht der Fehler darin, dass eine gesamte Zeile fehlt, kann diese durch das Nennen der vorherigen Zeile mit einem "+" eingefügt werden (Beispiel: 1+: das hier fehlt zwischen Zeile 1 und 2).

```
1  entity hw_mul is
2      Generic(
3          N : integer := 4;
4          M : integer := 4 )
5      Port (  f1 : in  STD_LOGIC_VECTOR (N-1 downto 0);
6              f2 : in  STD_LOGIC_VECTOR (M-1 downto 0);
7              clk : in STD_LOGIC;
8              q : out  STD_LOGIC (M+N-1 downto 0));
9  end hw_mul;
10
11 architecture Behavioral is
12     signal sum : unsigned( M+N-1 downto 0);
13 begin
14
15     process( clk )
16         variable temp : unsigned( M+N-1 downto 0);
17     begin
18         if( risind_edge(clk) then
19             sum <= to_unsigned(0, M+N);
20
21             for I in 0 to N-1 loop
22
23                 temp := (others => '0');
24                 if( f1(I) = 1) then
25                     temp(I+M-1 downto I) := unsigned(f2);
26                 end if;
27
28                 sum <= sum + temp;
29
30             end loop;
31
32             q <= sum;
33         end if;
34
35     end process;
36
37 end Behavioral;
```

Tabelle 1: Verbesserungen eintragen

Fehler	Zeile	Korrektur
1	4	M : integer := 4);
2	8	q : out STD_LOGIC_VECTOR(N+M-1 downto 0));
3	11	architecture Behavioral of hw_mul is
4	24	if(f1(I) = '1') then
5	32	q <= std_logic_vector(sum);

(b) Nachdem die Syntax-Fehler korrigiert sind, lässt sich der Code auf der Hardware ausführen. Doch anstatt dem richtigen Ergebnis kommt bei der ersten Berechnung nur 0 oder das 8-Fache des Eingangsvektors `f2` heraus. Danach ist kein Muster mehr zu erkennen. (3)

Worin liegt der Fehler? Erkläre, warum dieser vorliegt und wie man diesen ausbessern kann.

Solution: Sum sollte eine Variable sein.

Der Fehler entsteht durch die Eigenschaft eines Signals nur am Ende eines Prozesses geschrieben zu werden. Dadurch wird immer nur der letzte Schleifendurchlauf in Sum geschrieben.

Sum wie temp als Variable anlegen.

Testbench

6. Die Komponente `SOME_ENTITY` soll mittels der Testbench `testbench` getestet werden.

Listing 1: Die Testbench

```
entity testbench is
end testbench;

architecture arch_0 of testbench is

component SOME_ENTITY is
    port(   A, B, CLK: in STD_LOGIC;
            P, Q: out STD_LOGIC );
end component;

signal CLK, A, B, P, Q : STD_LOGIC := '0';

begin
    uut : SOME_ENTITY
        port map( A=>A, B=>B, CLK=>CLK, P=>P, Q=>Q );

    clocking : process
    begin
        wait for 5ns;
        CLK <= not(CLK);
    end process clocking;

    stimulus : process
    begin
        wait for 10 ns;
        A <= not(A);
        B <= not(B);
        wait for 20 ns;
        A <= '0';
        wait for 20 ns;
        B <= '1';
        wait for 5 ns;
        B <= '0';
        wait for 5 ns;
        B <= '1';
    end process stimulus;
end arch_0;
```

Listing 2: Architecture der Unit-under-test

```
architecture arch_uut of SOME_ENTITY is
begin
    process( clk )
    begin
        if rising_edge(clk) then
            P <= A NOR B;
        end if;
    end process;

    Q <= A XOR B;
end arch_uut;
```

(a) Zeichnen Sie den Verlauf von `A` und `B` in das unten stehende Diagramm ein. (2)

(b) Zeichnen Sie nun den Verlauf der Ausgänge `P` und `Q` in das unten stehende Diagramm ein. (4)