

Programmmentwurf für Entwurf digitaler Systeme

TEL17
30.11.2019
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:	13	5	9	12	8	8	55
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) Führen Sie eine schriftliche, binäre Multiplikation der Binärzahlen **0b101010** und **0b001100** durch. (2)

[illegible]

- (b) ARM-Prozessoren verfügen über drei unterschiedliche Kern-Familien. Nennen Sie die ideale Familie um einen Einplatinenrechner (vollständiger Computer auf einer Platine) zu betreiben. (1)

- (c) Erklären Sie, was der Taktschlupf ist und nennen Sie eine Technik diesen zu minimieren. (2)

- (d) Erklären Sie, warum sich die Frequenz Integrierter Schaltkreise nicht weiter steigern lässt. (1)

- (e) Wie lässt sich die Leistung von Prozessoren steigern, wenn die Frequenz und die Anzahl der Transistoren pro Fläche auf dem Chip begrenzt ist? (2)

- (f) Nennen Sie die fünf Zustände, die man einem physikalischen Ausgang in STD_LOGIC zuweisen kann und erklären Sie jeden Zustand kurz. (5)

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) Für eine digitale Audio-Signalverarbeitung ist der FPGA die beste und wirtschaftlichste Lösung
☐ Wahr ☐ Falsch
- (b) Bei der Synthese wird ein VHDL-Programm von der Verhaltens- in die Strukturebene übersetzt.
☐ Wahr ☐ Falsch
- (c) Bei Standardzellen-Entwürfen kommt es im Gegensatz zu Sea-Of-Gate-Array-Entwürfen nicht zu Problemen bei der Verdrahtung großer Designs.
☐ Wahr ☐ Falsch
- (d) Die Ausgabe eines Moore-Automats ist sowohl vom Zustand als auch von den Eingängen abhängig. ☐ Wahr ☐ Falsch
- (e) Hazards lassen sich vermeiden, indem bei einer bedingten Zuweisung alle möglichen Zustände abgedeckt werden.
☐ Wahr ☐ Falsch

VHDL Programmieren

3. Der folgende Code zeigt die Architektur der Entity bits_adder.

```
architecture Structural of bits_adder is

    component fulladder
    Port (  A    : in  STD_LOGIC;
           B    : in  STD_LOGIC;
           CIN  : in  STD_LOGIC;
           Q    : out STD_LOGIC;
           COUT : out STD_LOGIC);
    end component;

    signal sig_c0 : STD_LOGIC := '0';
    signal sig_c1 : STD_LOGIC := '0';

begin

fulladder_0 : fulladder
port map(
    A => A(0),
    B => B(0),
    CIN => '0',
    Q => Q(0),
    COUT => sig_c0 );

fulladder_1 : fulladder
port map(
    A => A(1),
    B => B(1),
    CIN => sig_c0,
    Q => Q(1),
    COUT => sig_c1 );

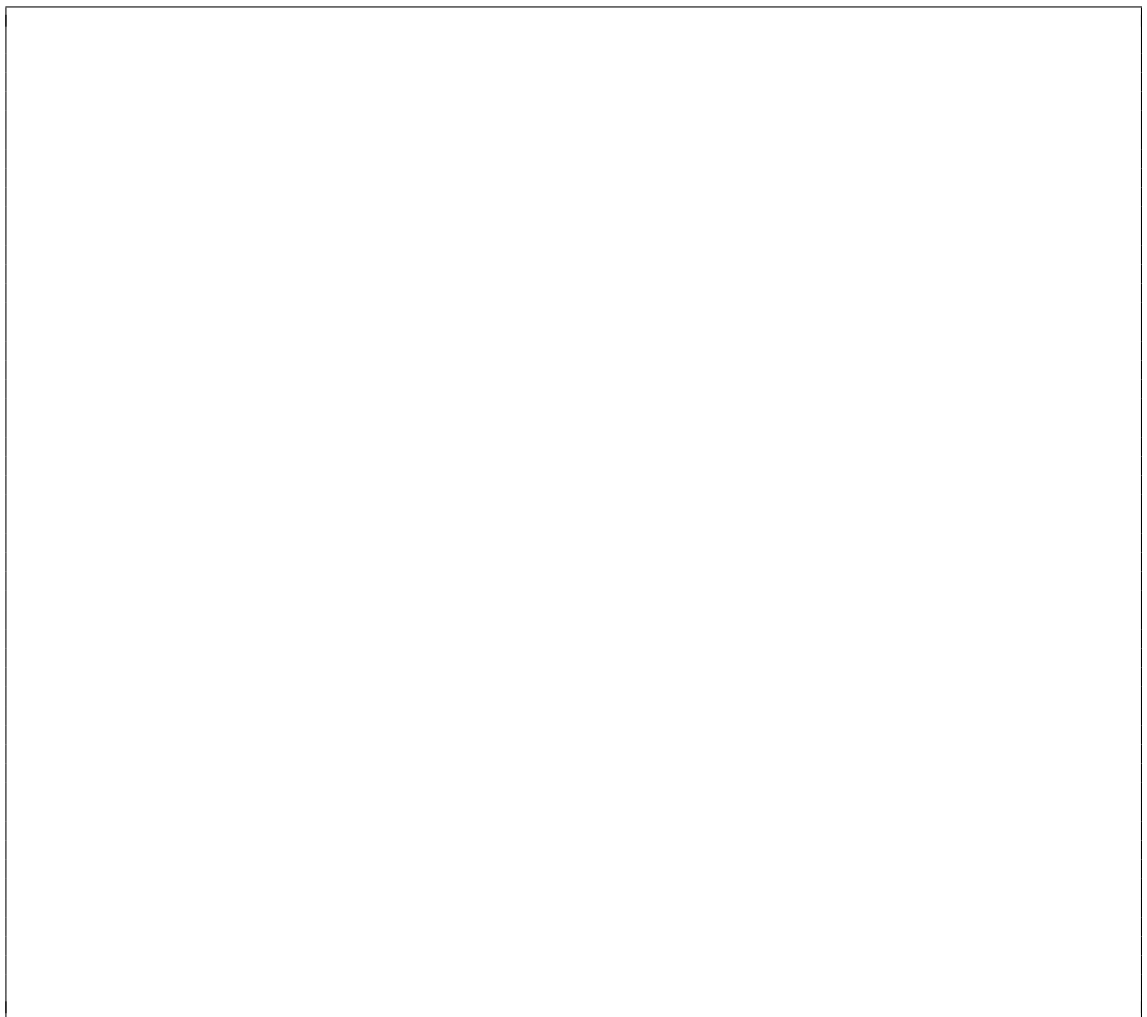
fulladder_2 : fulladder
port map(
    A => A(2),
    B => B(2),
    CIN => sig_c1,
    Q => Q(2),
    COUT => C );

end Structural;
```

- (a) Erarbeiten Sie die zur Architektur gehörende Entity und formulieren Sie diese inklusive der Ports aus. Includes müssen nicht aufgeführt werden. (3)

- (b) Nennen Sie die genaue Aufgabe dieser Logikschaltung. (1)

- (c) Zeichnen Sie den strukturellen Aufbau dieser Architektur. Nehmen sie dazu den Volladdierer als Komponente an. Beschriften Sie interne Signale mit den im Code verwendeten Namen. Sollte Ihnen der Platz nicht ausreichen, verwenden Sie die Rückseite des Arbeitsblatts. (5)



4. Ihre Aufgabe ist es, ein Code-Schloss basierend auf einem FPGA zu programmieren. Dieses hat vier Schalter ($S(3..0)$). Der Code dieses Schlosses lautet 0,3,2. Folglich entsperrt sich das Schloss, wenn zuerst $S(0)$, dann $S(3)$ und zuletzt $S(2)$ gedrückt werden. Danach geht das Unlock-Signal auf logisch '1'. Ein erneuter Tastendruck oder eine '1' am Reset-Signal setzen das Unlock-Signal wieder auf logisch '0'. Wird eine falsche Taste gedrückt oder während dem Eingeben ein Reset ausgelöst, muss der Code erneut von Beginn eingegeben werden.

```
entity num_lock is
    port(
        rst : IN STD_LOGIC;
        S    : IN UNSIGNED(3 downto 0);
        unlock : OUT STD_LOGIC;
    );
end num_lock;
```

- (a) Nennen Sie die notwendige State-Machine für diesen Anwendungsfall (1)

- (b) Realisieren Sie die Architektur für `num_lock` als State-Machine. (11)

[illegible]

VHDL Korrigieren

5. Im folgenden sind zwei Beispiele für VHDL-Codes zu sehen, wovon in jedes ein Fehler eingebaut ist (Kein Syntaxfehler).

```

1 architecture Behavioral of counter is
2     constant MAX : UNSIGNED(31 downto 0) :=
3         to_unsigned((FREQ_IN/FREQ_OUT/2)-1, 32);
4     signal sig_CNT : UNSIGNED(31 downto 0) := (others=>'0');
5 begin
6
7     cnt_proc: process( CLK_IN, RST )
8     begin
9         if( RST = '1' ) then
10             sig_CNT <= (others => '0');
11             CLK_OUT <= '0';
12         elsif( rising_edge( CLK_IN ) ) then
13             if( sig_CNT = MAX ) then
14                 sig_CNT <= (others => '0');
15                 CLK_OUT <= not CLK_OUT;
16             else
17                 sig_CNT <= sig_CNT + 1;
18             end if;
19         end if;
20     end process cnt_proc;
21
22 end Behavioral;

```

(a) Finden und nennen Sie den Fehler in der Architektur von `counter`. Erklären Sie, wie man diesen Fehler beheben kann.

[illegible]


```
1 architecture moore of ampel is
2
3     type state_type is (s_rot, s_rotgelb, s_gruen, s_gelb);
4     signal state_reg, state_next : state_type := s_rot;
5
6 begin
7
8     clock_process : process( clk, rst )
9     begin
10         if( rising_edge(rst) ) then
11             state_reg <= s_rot;
12         elsif( rising_edge(clk) ) then
13             state_reg <= state_next;
14         end if;
15     end process clock_process;
16
17
18     state_transition : process( state_reg )
19     begin
20         case( state_reg ) is
21             when s_rot =>
22                 state_next <= s_rotgelb;
23             when s_rotgelb =>
24                 state_next <= s_gruen;
25             when s_gruen =>
26                 state_next <= s_gelb;
27             when s_gelb =>
28                 state_next <= s_rot;
29             when others =>
30                 state_next <= s_rot;
31             end case;
32         end process state_transition;
33
34         -- State output fehlt, dies ist kein Fehler!
35
36 end moore;
```

- (b) Finden und nennen Sie den Fehler in der Architektur von moore. Erklären Sie, wie man diesen Fehler beheben kann.

(4)

Testbench

6. Im folgenden ist der volle Funktionsumfang eines Volladdierers zu testen. Dafür wird die folgende Testbench erstellt.

Listing 1: Die Testbench

```
entity testbench is
end testbench;

architecture arch_0 of testbench is

    component fulladder is
        port(
            A, B, CIN: in STD_LOGIC;
            Q, COUT : out STD_LOGIC );
    end component;

    signal A, B, CIN, Q, COUT : STD_LOGIC := '0';

begin

    uut : fulladder
        port map( A=>A, B=>B, CIN=>CIN, Q=>Q, COUT=>COUT );

    proc_stimuli : process
    begin
        A <= '0';
        B <= '0';
        CIN <= '0';
        wait for 100ns;

        A <= '1';
        -- weitere Stimuli
    end process proc_stimuli;

    proc_assert_q : process( Q )
    begin

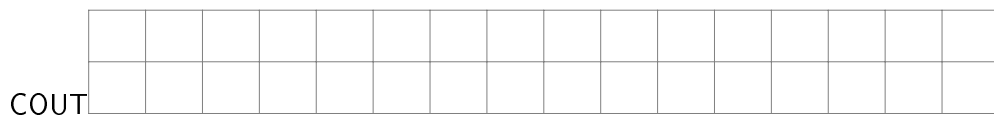
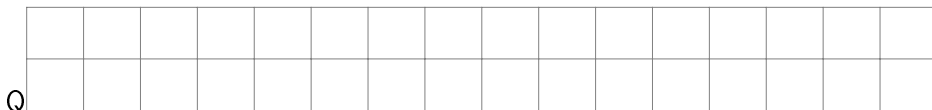
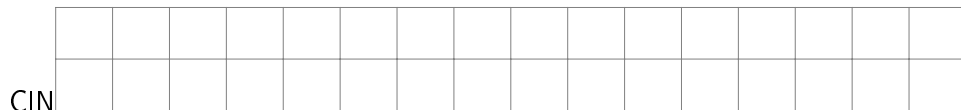
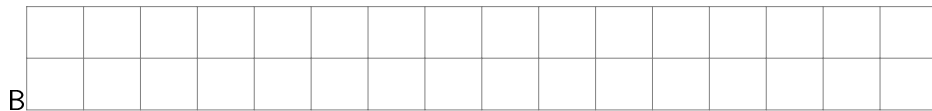
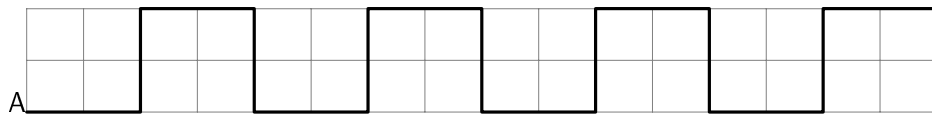
        -- hier Aufgabenteil c)

    end process proc_assert_q;

end arch_0;
```

(a) Ergänzen Sie in den unten stehenden Signalverlaufsgraphen die übrigen Eingangssignale B und CIN, die nötig sind um alle Eingangskombinationen zu testen. (2)

(b) Berechnen Sie die Ausgänge Q und COUT und tragen sie diese ebenfalls in die Graphen ein. (2)



(c) Programmieren Sie den Assertion-Prozess für Q der jedes Ergebnis überprüft. Geben Sie eine Warning mit der Meldung "Es ist ein Fehler aufgetreten." aus, wenn der empfangene Wert nicht dem erwarteten entspricht. Der Einfügeort Ihres Codes ist im Listing vermerkt. (4)

Um Schreibarbeit zu sparen, können Sie die Assert-Statements ab dem Zweiten folgendermaßen abkürzen: "assert *Bedingung* ...;"

Syntax Assert:

```
assert condition
    report string
    severity severity_level;
```

[illegible]