

# Programmmentwurf für Entwurf digitaler Systeme

TEL21 - Nachholtermin  
07.03.2023  
Torben Mehner

Matrikelnummer: \_\_\_\_\_

## Erlaubte Hilfsmittel:

- Ausgeteilte Formelsammlung
- Nicht-programmierbarer Taschenrechner

## 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	<b>Summe</b>
Punktzahl:	11	5	8	12	9	7	52
Davon erreicht:							

Note: \_\_\_\_\_

Unterschrift

Korrektor: \_\_\_\_\_

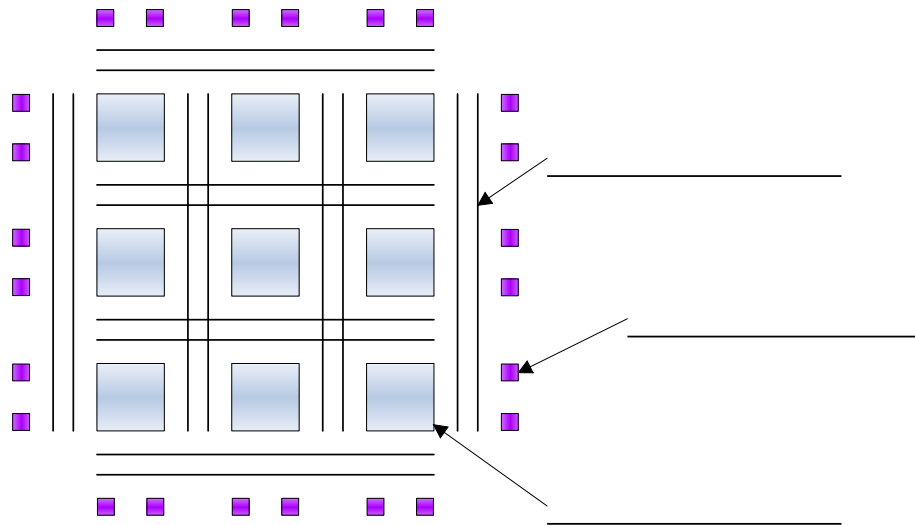
Viel Erfolg!

# 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) Benennen Sie die gekennzeichneten Stellen im Aufbau eines FPGAs und erklären Sie knapp deren Aufgaben.

(3)



- (b) Führen Sie eine schriftliche, binäre Multiplikation der Binärzahlen **0b111001** und **0b101010** durch.

(2)

A full-page sheet of white graph paper with a light gray grid. The grid consists of small squares, approximately 10 units wide by 10 units high. There are no margins or additional markings on the page.

- (c) Erklären Sie kurz den Vorteil von einem Multiplizierer mit Pipelining im Vergleich zu einem Multiplizierer ohne Pipelining. Beziehen Sie sich dabei auf den Taktschlupf. (1)

---

---

---

---

---

- (d) Nennen Sie für 5 Zustände der STD\_LOGIC das in VHDL verwendete Kurzzeichen und den Namen. (Beispiel: '-': Don't care, die Nennung des Beispiels gibt keinen Punkt) (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 -0.5 Punkte. Wird bei einer Aussage nichts angekreuzt, gibt es keinen Abzug. (5)

- (a) Beim Standardzellenentwurf gibt es verschiedene Implementierungen einer Zelle, die jeweils anders optimiert sind.  
☐ Wahr ☐ Falsch
- (b) Bei Standardzellen-Entwürfen kommt es im Gegensatz zu Sea-Of-Gate-Array-Entwürfen nicht zu Problemen bei der Verdrahtung großer Designs.  
☐ Wahr ☐ Falsch
- (c) Die Ausgabe eines Moore-Automats ist sowohl vom Zustand als auch von den Eingängen abhängig. ☐ Wahr ☐ Falsch
- (d) *Sieben\_Segment\_Anzeige* ist ein zulässiger VHDL-Identifizier.  
☐ Wahr ☐ Falsch
- (e) Die Entity einer VHDL-Testbench hat die selben Ein- und Ausgänge wie das zu testende Modul  
☐ 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)

---

---

---

---

---

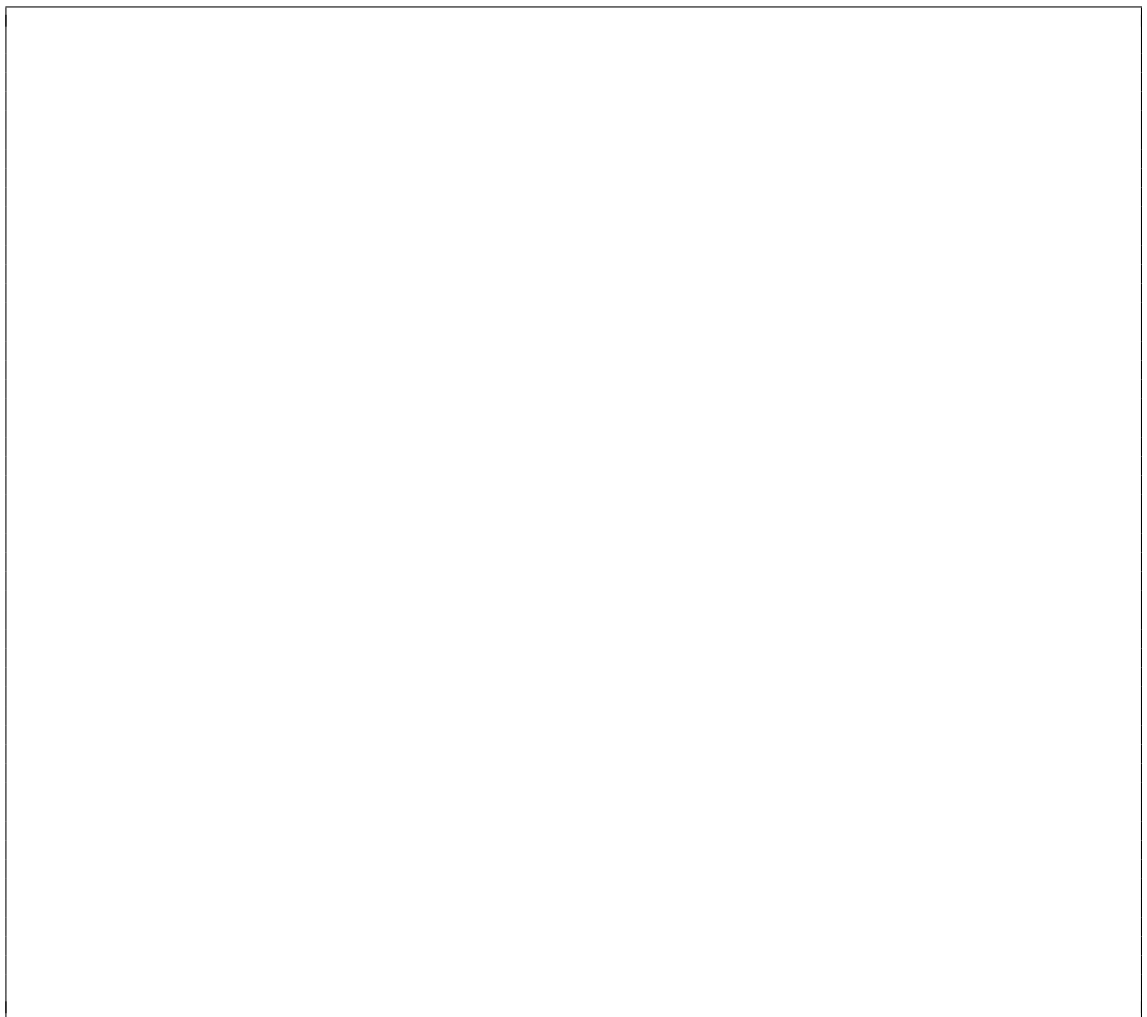
---

---

- (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. (4)



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 an eingegeben werden.

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

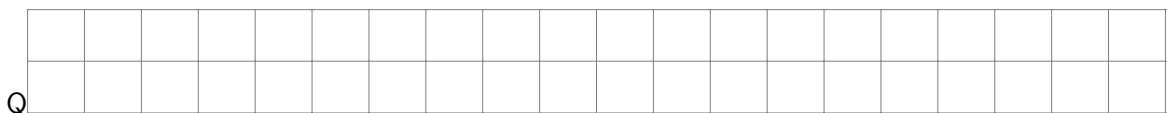
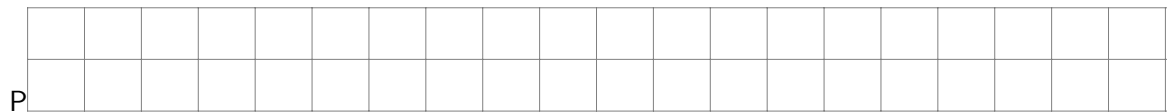
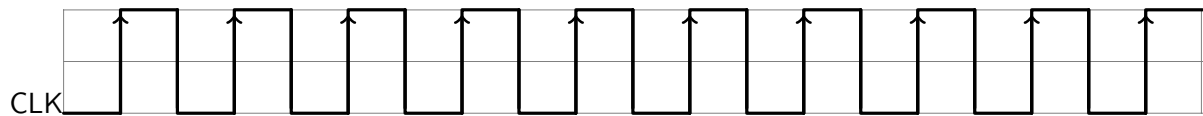
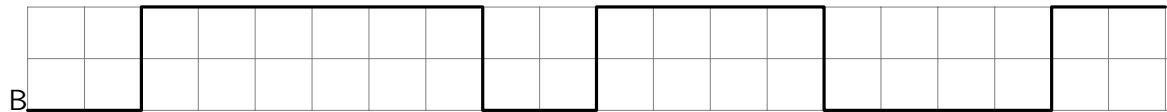
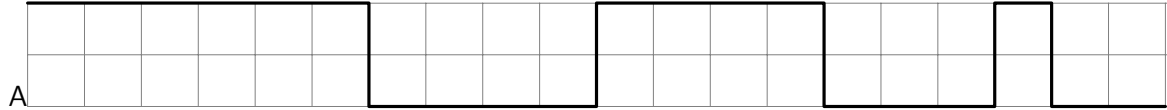
- (a) Nennen Sie die notwendige State-Machine für diesen Anwendungsfall (1)

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

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

## Simulation

5. Die folgenden Schaubilder zeigen die Stimuli einer Testbench für die UUT. Ein Clock-Zyklus (High und Low) dauert 100ns. Listing 1 zeigt die Testbench, Listing 2 die Architektur der Unit-Under-Test. Die Signallaufzeiten sind idealisiert nicht vorhanden.





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 50ns;
        CLK <= not(CLK);
    end process clocking;

    stimulus_a : process
    begin
        -- hier Aufgabenteil a)
    end process stimulus_a;

    stimulus_b : process
    begin
        -- hier Aufgabenteil b)
    end process stimulus_b;
end arch_0;
```

Listing 2: Architecture der Unit-under-test

```
architecture arch_uut of SOME_ENTITY is
    P <= A AND CLK;

    process( CLK )
    begin
        if rising_edge(CLK) then
            Q <= P OR B;
        end if;
    end process;
end arch_uut;
```

- (a) Programmieren Sie den Stimuli-Prozess für A. Sie müssen den umgebenden Prozess nicht  
nocheinmal abschreiben. Ersetzen Sie einfach das Kommentar " - - hier Aufgabenteil a)" (5)

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

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

## Fehlersuche

6. 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		
2		
3		
4		
5		

- (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  $\mathbf{x}_2$  heraus. Danach ist kein Muster mehr zu erkennen. Worin liegt der Fehler? Erkläre, warum dieser vorliegt.

(2)

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.