Lehrstuhl für Informatik 12
(Hardware-Software-Co-Design)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Prof. Dr.-Ing. Jürgen Teich

Klausur
Grundlagen der Technischen Informatik

5. April 2017

Name	
Matrikelnummer	
Studienrichtung	

Aufgabe	1	2	3	4	5	Σ
Max. Punkte	15	15	20	15	15	80
Erreichte Punkte						
Note						

Organisatorische Hinweise

Bitte sorgfältig lesen und die Kenntnisnahme durch Unterschrift bestätigen

---

a) Bitte legen Sie Ihren Studentenausweis bereit.

b) Als Hilfsmittel sind nur Schreibmaterialien und ein beidseitig handbeschriebenes DIN A4-Blatt zugelassen.

c) Verwenden Sie weder Rot- noch Bleistifte.

d) Schmierpapier wird nicht abgegeben und auch nicht korrigiert.

e) Sie können bei der Aufsicht zusätzliche Bearbeitungsblätter anfordern.

f) Unleserliches wird nicht bewertet.

g) Bei mehreren präsentierten Lösungen wird die Aufgabe nicht gewertet. Streichen Sie daher bei Angabe mehrerer Lösungsansätze die nicht zu bewertenden Lösungen durch.

h) Die Bearbeitungszeit beträgt 120 Minuten.

Erklärung

a) Im Falle einer während der Prüfung auftretenden Prüfungsunfähigkeit zeige ich dies sofort der Aufsicht an und befolge deren Anweisungen. Ich weiß, dass ich die volle Beweislast trage. Ich lasse mir das Formular des Prüfungsamts, das für diese Fälle vorgesehen ist, aushändigen und verfahre nach den dort niedergelegten Richtlinien.

b) Ich weiß, dass im Falle des Täuschungsversuchs oder der Benutzung unerlaubter Hilfsmittel („Unterschleif“) der Prüfungsausschuss die Entscheidung treffen kann, die betroffene Prüfungsleistung als mit „nicht ausreichend“ bewertet gelten zu lassen.

c) Ich habe die obigen Hinweise zur Kenntnis genommen.

Erlangen, den .....

Unterschrift

Aufgabe 1 (Zahlensysteme) (15 Punkte)

a) Wie lautet der Wertebereich einer n-stelligen vorzeichenlosen Ternärzahl, d. h. mit den Ziffern 0, 1 und 2? (1 Punkt)

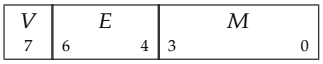
b) Wie lautet der Wertebereich einer n-stelligen Binärzahl im Zweierkomplement? (1 Punkt)

c) Welche Zahl im Ternärsystem entspricht der vorzeichenlosen Zahl 473_9 im Neuner-Zahlensystem? (3 Punkte)

d) Beantworten Sie folgende Auswahlfragen. Jede richtige Antwort gibt einen Punkt, jede falsche Antwort führt zum Abzug eines Punktes, nicht beantwortete Fragen werden nicht gewertet, weniger als null Punkte sind nicht möglich. (3 Punkte)

1. Mit Und- und Oder-Gattern sowie den Konstanten **wahr** und **falsch** lässt sich jede beliebige Schaltfunktion darstellen. ☐ wahr ☐ falsch
2. Mit Und- und Antivalenz-Gattern (XOR) sowie den Konstanten **wahr** und **falsch** lässt sich jede beliebige Schaltfunktion darstellen. ☐ wahr ☐ falsch
3. Die Addition zweier Zahlen im Zweierkomplement mit 32 Bit ist aufwändiger als die Addition zweier vorzeichenloser Binärzahlen mit 32 Bit. ☐ wahr ☐ falsch

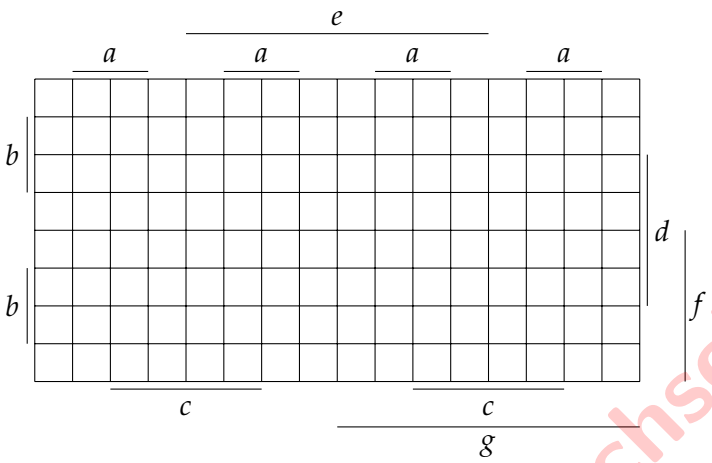
e) In dieser Aufgabe soll mit 8-Bit Gleitkommazahlen gearbeitet werden. Diese werden analog zum IEEE-Format gebildet. Das Format der Gleitkommazahl sieht dabei wie folgend aus:
Vorzeichen (1 Bit), *Exponent (3 Bit)*, *Mantisse (4 Bit)*



Führen Sie nun die Division der beiden in diesem Format dargestellten Gleitkommazahlen 1 011 0101 und 0 010 1100 aus, und geben Sie das Ergebnis im gleichen Format an. (7 Punkte)

Aufgabe 2 (Minimierung von Schaltfunktionen) (15 Punkte)

a) Gegeben sei die Schaltfunktion $f_1(g,f,e,d,c,b,a) = fe\bar{d}\bar{c}b + \bar{f}dca + dca + c\bar{b}a + fd$. Bilden Sie mit Hilfe des gegebenen Symmetriediagramms eine DMF von f_1. Leere Felder im Symmetriediagramm beschreiben eine 0. (5 Punkte)



b) Es sei die Schaltfunktion $f_2(d,c,b,a) = c(ab + ad)(c + \bar{d})$ gegeben. Implementieren Sie diese mit so wenig Transistoren wie möglich unter Verwendung der *CMOS*-Technologie. Vereinfachen Sie dazu gegebenenfalls f_2. Stellen Sie zunächst die Booleschen Ausdrücke für das *PUN* und *PDN* auf. Übertragen Sie diese anschließend in das vorgegebene Diagramm. Es stehen Ihnen ausschließlich die Eingänge a, b, c und d nicht negiert zur Verfügung. (4 Punkte)

VDD

GND

c) Im Folgenden sollen Sie das *Quine/McCluskey*-Verfahren zur Minimierung von Schaltfunktionen anwenden.

1. Seien die Primimplikate der in folgender Funktionstabelle spezifizierten Funktion $f_3(c,b,a)$ gesucht. Geben Sie für diesen Fall die Klassen $Q_{3,3}$ bis $Q_{3,0}$ an. (1 Punkt)

c	b	a	f_3
0	0	0	1
0	0	1	0
0	1	0	-
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	-

$Q_{3,3} =$

$Q_{3,2} =$

$Q_{3,1} =$

$Q_{3,0} =$

2. Bestimmen Sie die Primimplikanten der Schaltfunktion $f_4(e,d,c,b,a) = \bar{e}\bar{d}c\bar{b}\bar{a} + e\bar{d}c\bar{b}\bar{a} + \bar{e}\bar{d}\bar{c}ba + \bar{e}dc\bar{b}\bar{a} + ed\bar{c}\bar{b}\bar{a} + \bar{e}d\bar{c}ba + edc\bar{b}\bar{a} + \bar{e}\bar{d}cba + \bar{e}dcba + e\bar{d}cba$. Die Klassen $Q_{5,0}$ bis $Q_{5,5}$ seien bereits gegeben. Markieren Sie die im Verlauf des Verfahrens genutzten Terme durch Unterstreichen. (5 Punkte)

$Q_{5,5} = \{\}$

$Q_{5,4} = \{\bar{e}\bar{d}c\bar{b}\bar{a}\}$

$Q_{5,3} = \{e\bar{d}c\bar{b}\bar{a},\ \bar{e}\bar{d}\bar{c}ba,\ \bar{e}dc\bar{b}\bar{a},\ ed\bar{c}\bar{b}\bar{a}\}$

$Q_{5,2} = \{\bar{e}d\bar{c}ba,\ edc\bar{b}\bar{a},\ \bar{e}\bar{d}cba\}$

$Q_{5,1} = \{\bar{e}dcba,\ e\bar{d}cba\}$

$Q_{5,0} = \{\}$

Aufgabe 3 (Automaten) (20 Punkte)

a) Im Folgenden soll ein Milchautomat entworfen werden, der an Milchtankstellen eingesetzt wird, um die täglich frische Landmilch rund um die Uhr und ohne Personaleinsatz zu verkaufen. Hierfür verfügt der Automat über vier verschiedene Eingaben, die jedoch nicht gleichzeitig anliegen können: Durch Einwurf von Münzen wird das Guthaben erhöht (Eingabe *ICredit*); der Abfüllvorgang wird gestartet (Eingabe *IStart*); der Abfüllvorgang wird beendet bzw. Guthaben wird ausbezahlt (Eingabe *IStop*); kein Guthaben ist mehr vorhanden (Eingabe *INoFunds*).

Das Verhalten ist wie folgt spezifiziert: Im Startzustand sollen alle Eingaben außer dem Münzeinwurf zum Erhöhen des Guthabens ignoriert werden (Ausgabe *NoOut*). Nach dem ersten Münzeinwurf ist der Automat einsatzbereit (Ausgabe *Ready*), und es kann entweder der Abfüllvorgang begonnen werden (Ausgabe *Milk*), das Guthaben direkt wieder ausbezahlt werden (Ausgabe *Credit*) oder das Guthaben durch einen weiteren Münzeinwurf erhöht werden (Ausgabe *Ready*). Wird zu diesem Zeitpunkt fälschlicherweise signalisiert, dass kein Guthaben vorhanden ist, soll in einen Fehlerzustand übergegangen werden (Ausgabe *Error*), aus dem nicht wieder zurückgekehrt werden kann, da ein ordnungsgemäßer Betrieb nicht garantiert ist. Liegt während des Abfüllvorgangs die Eingabe *IStart* an bzw. wird das Guthaben erhöht, soll weiterhin Milch ausgegeben werden (Ausgabe *Milk*). Der gestartete Abfüllvorgang wird entweder durch ein aufgebrauchtes Guthaben (Ausgabe *NoOut*) oder manuell beendet. Das bei einem manuellen Abbruch noch vorhandene Guthaben soll anschließend ausbezahlt werden (Ausgabe *Credit*). Sobald das Guthaben komplett ausbezahlt wurde, kehrt der Automat in den Startzustand zurück (Ausgabe *NoOut*). Liegt während der Geldausgabe fälschlicherweise die Eingabe *ICredit*, *IStart* oder *IStop* an, soll auch hier in den Fehlerzustand übergegangen werden (Ausgabe *Error*).

Die Ein- und Ausgaben sind dabei wie folgt durch binäre Variablen codiert:

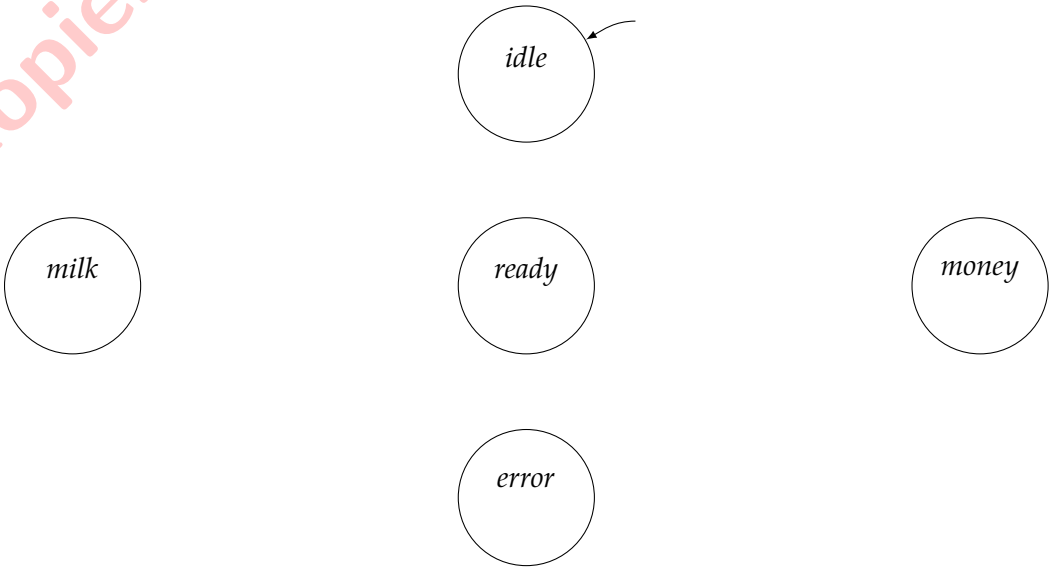
Eingabe	i_1	i_0
ICredit	0	0
IStart	0	1
IStop	1	0
INoFunds	1	1

Eingaben des Automaten

Ausgabe	o_2	o_1	o_0
NoOut	0	0	0
Milk	0	0	1
Credit	0	1	0
Ready	0	1	1
Error	1	0	0

Ausgaben des Automaten

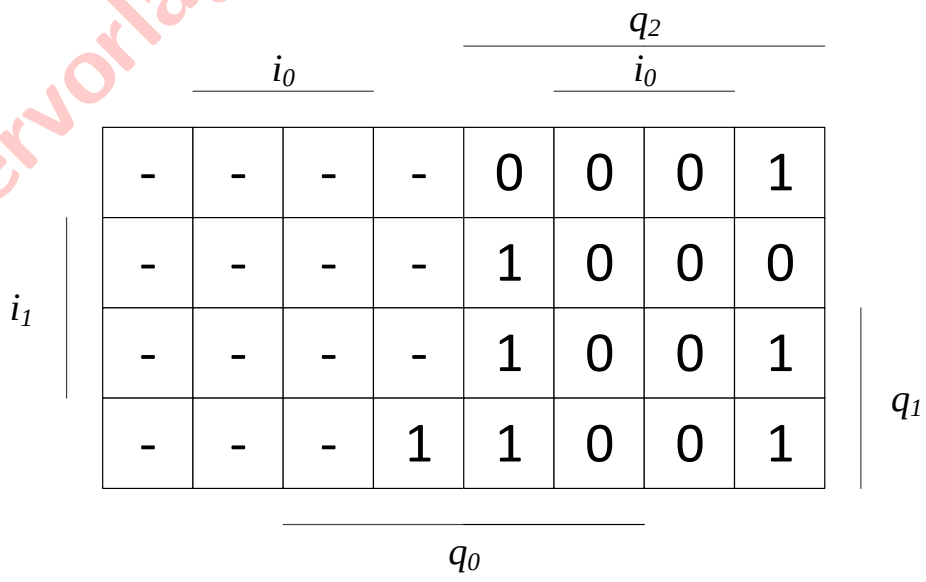
1. Spezifizieren Sie den beschriebenen Automaten als Moore-Automat unter Verwendung der fünf Zustände *idle*, *ready* (Automat ist einsatzbereit), *milk* (Ausgabe von Milch), *money* (Ausgabe des Guthabens) und *error*. Geben Sie den resultierenden Automatengraphen an. (5 Punkte)



2. Vervollständigen Sie die nachfolgend gegebene Automatentafel unter Verwendung von taktflankengesteuerten D-, T- bzw. JK-Flipflops. (6 Punkte)

Zustandsname	Aktueller Zustand			Eingabe		Nachfolgezustand			Ansteuerfunktion				Ausgabe		
	q_2	q_1	q_0	i_1	i_0	q_2'	q_1'	q_0'	D_2	T_1	J_0	K_0	o_2	o_1	o_0
idle	0	0	0	0	0										
idle	0	0	0	0	1										
idle	0	0	0	1	0										
idle	0	0	0	1	1										
ready	0	0	1	0	0										
ready	0	0	1	0	1										
ready	0	0	1	1	0										
ready	0	0	1	1	1										
milk	0	1	0	0	0										
milk	0	1	0	0	1										
milk	0	1	0	1	0										
milk	0	1	0	1	1										
money	0	1	1	0	0										
money	0	1	1	0	1										
money	0	1	1	1	0										
money	0	1	1	1	1										
error	1	0	0	0	0										
error	1	0	0	0	1										
error	1	0	0	1	0										
error	1	0	0	1	1										

3. Nehmen Sie im Folgenden an, dass die Ansteuerfunktion J_0 des JK-Flipflops im unten gegebenen Symmetriediagramm spezifiziert sei. Entwickeln Sie eine DMF der Ansteuerfunktion J_0 und geben Sie den resultierenden schaltalgebraischen Ausdruck an.
Hinweis: Die hier angegebene Ansteuerfunktion J_0 entspricht nicht der Lösung von Teilaufgabe 2.
(3 Punkte)



4. Zeichnen Sie das Schaltwerk des in Teilaufgabe 2 erstellten Moore-Automaten. Verwenden Sie dazu die in Teilaufgabe 3 bestimmte DMF der Ansteuerfunktion J_0, und gehen Sie davon aus, dass Sie die benötigten Ansteuerfunktionen D_2, T_1 und K_0 als Eingangssignale zur Verfügung haben. Auf die Realisierung der Ausgabe des Automaten kann verzichtet werden. (3 Punkte)

b) Beantworten Sie folgende Auswahlfragen. Jede richtige Antwort gibt einen Punkt, jede falsche Antwort führt zum Abzug eines Punktes, nicht beantwortete Fragen werden nicht gewertet, weniger als null Punkte sind nicht möglich. (3 Punkte)

1. Die technische Realisierung eines endlichen, diskreten und deterministischen Automaten wird Schaltwerk genannt. ☐ wahr ☐ falsch
2. Bei $|S|$ verschiedenen Zuständen werden $r = \lceil ld|S| \rceil - 1$ Zustandsvariablen benötigt. ☐ wahr ☐ falsch
3. Mit einer Schaltung bestehend aus einem Inverter und einem Oder-Gatter lassen sich positive Flanken erkennen. ☐ wahr ☐ falsch

Aufgabe 4 (Codierung und Rechnerarithmetik) (15 Punkte)

a) Gegeben ist ein Zeichenvorrat bestehend aus folgenden Zeichen mit ihren zugehörenden Häufigkeiten:

Zeichen	*A*	*B*	*C*	*D*	*E*
Häufigkeit [in %]	5	10	16	29	40

1. Konstruieren Sie den Huffman-Baum. Hierbei sollen die mit der kleineren Auftrittshäufigkeit verbundenen Kanten mit einer 0 und die mit der höheren Auftrittshäufigkeit verbundenen Kanten mit einer 1 codiert werden. (3 Punkte)

2. Codieren Sie nun die Nachricht *ADE* entsprechend. (1 Punkt)

3. Welchen Vorteil hat der Huffman-Code gegenüber dem Shannon-Fano-Code? (1 Punkt)

b) Gegeben sei eine Quelle, deren Alphabet aus 4-Bit Binärwörtern besteht.

1. Erweitern Sie das Binärwort 0010 dieser Quelle nach dem Paritätsprüfverfahren (*odd parity*). (1 Punkt)

2. Geben Sie nun die Konstruktionsvorschrift für Prüfvektoren entsprechend des Hamming-Code-Verfahrens an. Erstellen Sie dann das Codewort für 0010 entsprechend dieser Vorschrift. (2 Punkte)

c) Entwerfen Sie einen Multiplizierer, der eine 4-Bit Zahl $x_4x_3x_2x_1$ in Vorzeichen-Betragsdarstellung mit einer 2-Bit Zahl y_2y_1 in Vorzeichen-Betragsdarstellung multipliziert. Bits x_4 und y_2 codieren jeweils das Vorzeichen. (2 Punkte)

d) Erstellen Sie unter ausschließlicher Verwendung von NOR-Gattern...

1. ein XOR-Gatter. (3 Punkte)

2. einen Komparator, der zwei 2-Bit Zahlen x_2x_1 und y_2y_1 auf Gleichheit prüft gemäß folgender Vorschrift:

$$\text{equals}(x_2x_1,\ y_2y_1) = \begin{cases} 1, & \text{wenn } x_2x_1 = y_2y_1 \\ 0, & \text{sonst} \end{cases}.$$

(2 Punkte)

Aufgabe 5 (VHDL) (15 Punkte)

In dieser Aufgabe wird folgender Moore-Automat modelliert:

$$S = \{0,1,2,3,4,5,6,7\} \quad \text{(Zustandsmenge)}$$
$$E = \{0,1,2,3,4,5,6,7\} \quad \text{(Eingabealphabet)}$$
$$A = \{\texttt{G}, \texttt{S}, \texttt{I}, \texttt{U}, \texttt{␣}, \texttt{T}, \texttt{E}, \texttt{?}\} \quad \text{(Ausgabealphabet)}$$
$$s_0 = 0 \quad \text{(Startzustand)}$$

Seine Zustandsüberführungsfunktion δ und Ausgabefunktion λ sind durch folgende Automatentafel spezifiziert:

Eingabe:	0	1	2	3	4	5	6	7	
Zustand	Folgezustand								Ausgabe
0	0	1	2	3	4	5	6	7	G
1	1	2	3	4	5	6	7	0	S
2	2	3	4	5	6	7	0	1	I
3	3	4	5	6	7	0	1	2	U
4	4	5	6	7	0	1	2	3	␣
5	5	6	7	0	1	2	3	4	T
6	6	7	0	1	2	3	4	5	E
7	7	0	1	2	3	4	5	6	?

Die Eingabe drückt aus, wie weit der Folgezustand in (positiven) Schritten entfernt ist. (Nach Zustand 7 folgt wieder Zustand 0.) Eine Eingabe von 0 bedeutet zum Beispiel, dass der aktuelle Zustand beibehalten wird, eine Eingabe von 2, dass in den zwei Schritte entfernten Zustand übergegangen wird (etwa von 1 nach 3, von 4 nach 6, von 7 nach 1).

Entwerfen Sie eine *synchrone* Schaltung, die diesem Automaten entspricht. Die Schaltung soll *asynchron* zurückgesetzt werden können.

a) Vervollständigen Sie die folgende `entity`-Deklaration. Sowohl Eingabe `input` als auch Ausgabe `output` sollen als `std_logic_vector` übermittelt werden. Die Ausgabe sei dabei mit ASCII codiert, es gibt also insgesamt 128 Zeichen. (2 Punkte)

```
entity automaton is




end automaton;
```

b) Geben Sie die Eingabefolge an, die die Zeichenkette „G␣GG␣GGG" erzeugt. (1 Punkt)

c) Wie müsste der Automat geändert werden, damit die Zeichenkette „IST␣GTI␣GUT?" ausgegeben werden kann? (1 Punkt)

d) Vervollständigen Sie folgende Funktion, die δ implementiert, also für eine gegebene Kombination von Zustand und Eingabe den nächsten Zustand zurückliefert. (2 Punkte)

```
function get_next_state(state: integer, input: integer)
    return integer is


begin




end function;
```

e) Vervollständigen Sie folgende Funktion, die λ implementiert, also für einen gegebenen Zustand die entsprechende Ausgabe zurückliefert. Im Falle eines ungültigen Zustands soll `X` zurückgegeben werden. *Hinweis: Die Ausgabe `X` ist nicht Teil der Automatenspezifikation, weil sie genau dann auftritt, wenn die Hardwarerealisierung aufgrund von Laufzeitfehlern die Spezifikation nicht mehr erfüllt.* (3 Punkte)

```
function get_output(state: integer) return character is
begin -- Beispielsyntax: return 'P';
```



```
end function;
```

f) Vervollständigen Sie den Rumpf der folgenden `architecture` unter Verwendung der beiden obigen Funktionen. (4 Punkte)

```
architecture behavioral of automaton is
  signal state : integer := 0;
  signal input_int : integer := 0;
  signal output_char : character := ' ';
begin
  -- Ein-/Ausgabe in korrekte Typen umwandeln
  output <= character_to_std_logic_vector(output_char);
  input_int <= std_logic_vector_to_integer(input);









end architecture;
```

g) Geben Sie die Werte der Signale `d1`, `d2` und `d3` nach dem nächsten Simulationsschritt des folgenden `process` an, falls vorher gilt: `d1='0'`, `d2='0'` und `d3='1'`. Alle Signale sind vom Typ `std_logic`. (2 Punkte)

```
-- vorher: d1='0', d2='0', d3='1'
process(d1, d2, d3)
  variable temp : std_logic := '1';
begin
  temp := d2 and d3;
  d1 <= temp nand temp;
  d2 <= d1;
  temp := not temp;
  d3 <= d3 xor temp;
end process;
```