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

Klausur
Grundlagen der Technischen Informatik

21. März 2018

Name	
Matrikelnummer	
Studienrichtung	

Aufgabe	1	2	3	4	5	Σ
Max. Punkte	16	16	16	16	16	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) (16 Punkte)

a) Wie heißen die Zahlensysteme zur Basis 10 und 16? (1 Punkt)

b) Wie lautet der Wertebereich einer n-stelligen vorzeichenlosen Oktalzahl? (2 Punkte)

c) Welche Zahl im Ternärsystem entspricht der vorzeichenlosen Zahl 256_9 im Neunersystem? (3 Punkte)

d) Was sind die minimal und maximal möglichen Ergebnisse bei der Division $\frac{Z_n}{Z_m}$ einer n-stelligen Binärzahl Z_n im Zweierkomplement mit einer m-stelligen Binärzahl $Z_m \neq 0$ im Zweierkomplement? (2 Punkte)

e) 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. (4 Punkte)

1. Eine Gray-Codierung von Zahlen hat die Eigenschaft, dass sich benachbarte Codewörter, d. h. von einer Zahl n auf ihren Nachfolger $n + 1$, nur in einer einzigen binären Ziffer unterscheiden. ☐ wahr ☐ falsch
2. Mit Invertern und Antivalenz-Gattern sowie den Konstanten **wahr** und **falsch** lässt sich jede beliebige Schaltfunktion darstellen. ☐ wahr ☐ falsch
3. Bei der Umwandlung von Ganzzahlen im Ternärsystem in Zahlen im Oktalsystem kann es zu Rundungsfehlern kommen. ☐ wahr ☐ falsch
4. Eine Multiplikationseinheit für zwei vorzeichenlose 32 Bit breite Binärzahlen a und b mit einem 32 Bit breiten Ergebnis $s = a \cdot b$ berechnet auch für die Multiplikation zweier 32-Bit-Zahlen im Zweierkomplement ein richtiges Ergebnis, wenn dieses Ergebnis im Wertebereich der 32-Bit-Zweierkomplement-Darstellung liegt. ☐ wahr ☐ falsch

f) Gegeben sei eine Recheneinheit (4-Bit-ALU) zur Berechnung von $r = x \,\mathrm{op}\, y$ mit den Operationen $\mathrm{op} \in \{+, -, =, \neq, >_\mathrm{s}, \geq_\mathrm{s}, <_\mathrm{s}, \leq_\mathrm{s}, >_\mathrm{ns}, \geq_\mathrm{ns}, <_\mathrm{ns}, \leq_\mathrm{ns}\}$ sowie den Booleschen Verknüpfungen $\mathrm{op} \in \{\wedge, \vee, \neg\}$. Die Operanden x und y sowie das Ergebnis r sind dabei 4 Bit breite Binärzahlen. Von den Operationen $\{>_\mathrm{s}, \geq_\mathrm{s}, <_\mathrm{s}, \leq_\mathrm{s}\}$ werden diese Binärzahlen als Zweierkomplement-Zahlen interpretiert. Im Gegensatz dazu interpretieren die Operationen $\{>_\mathrm{ns}, \geq_\mathrm{ns}, <_\mathrm{ns}, \leq_\mathrm{ns}\}$ ihre Operanden als vorzeichenlose Binärzahlen. Die restlichen Operationen $\{+, -, =, \neq, \wedge, \vee, \neg\}$ sind unabhängig von der Entscheidung, ob ihre Operanden als vorzeichenlose Zahl oder vorzeichenbehaftete Zahl in Zweierkomplement-Darstellung angesehen werden.

Die 4-Bit-ALU führt nun die folgende Subtraktion aus:

$$d_\mathrm{s} := x_\mathrm{ns} - y_\mathrm{ns}$$

Die Operanden x_ns und y_ns sind dabei als vorzeichenlose Binärzahlen zu interpretieren, wohingegen das Ergebnis d_s als vorzeichenbehaftete Zahl in Zweierkomplement-Darstellung zu interpretieren ist. Bei dieser Subtraktion kann ein Überlauf bzw. Unterlauf auftreten, d. h. das Ergebnis der Subtraktion liegt außerhalb des Wertebereichs einer 4 Bit breiten Binärzahl in Zweierkomplement-Darstellung und lässt sich deshalb nicht korrekt in d_s darstellen.

Geben Sie einen Test an, welcher zu **wahr** auswertet, wenn bei der Subtraktion $d_\mathrm{s} := x_\mathrm{ns} - y_\mathrm{ns}$ mit der 4-Bit-ALU ein Überlauf oder Unterlauf aufgetreten ist. Dem Test stehen ausschließlich die 15 Operationen der 4-Bit-ALU, Konstanten darstellbar als 4 Bit breite Binärzahlen, die drei Werte x_ns, y_ns und d_s, sowie beliebige Zwischenergebnisse zur Verfügung. (4 Punkte)

Ein solcher Test könnte beispielsweise wie folgt aussehen:

$$(x_\mathrm{ns} >_\mathrm{ns} d_\mathrm{s} \wedge y_\mathrm{ns} >_\mathrm{s} -1) \vee y_\mathrm{ns} - x_\mathrm{ns} >_\mathrm{s} 1 \vee d_\mathrm{s} = 7$$

Aufgabe 2 (Minimierung von Schaltfunktionen) (16 Punkte)

a) Minimieren Sie f_1 unter Verwendung des Quine/McCluskey-Verfahrens und unterstreichen Sie alle Primimplikanten. (4 Punkte)

j	c	b	a	$f_1(c,b,a)$
0	0	0	0	1
1	0	0	1	1
2	0	1	0	-
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	-

b) Sei der schaltalgebraische Ausdruck des Pull-Up-Netzwerks (PUN) einer CMOS-Schaltung der Schaltfunktion $f_2(d,c,b,a)$ gegeben:

$PUN(f_2) = \overline{a \cdot b + \bar{d}} \cdot (c + a)$.

1. Zeichnen Sie $PUN(f_2)$ in das vorgegebene Diagramm ein. (2 Punkte)

VDD

GND

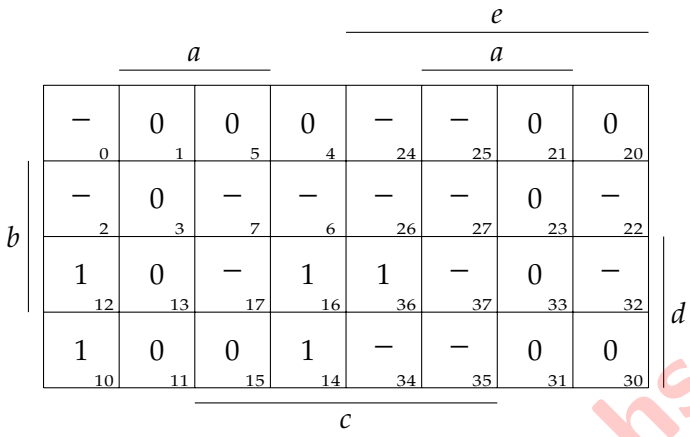
2. Bestimmen Sie den schaltalgebraischen Ausdruck des Pull-Down-Netzwerks von f_2: (2 Punkte)

   $PDN(f_2) =$

3. Ergänzen Sie die CMOS-Schaltung im Diagramm mit $PDN(f_2)$. (2 Punkte)

c) Sei die Schaltfunktion $f_3(e,d,c,b,a)$ mit Primimplikanten PI und zugehörigem Symmetriediagramm gegeben:

$$PI = \{\bar{e}\bar{c}\bar{a}, \bar{e}d\bar{a}, dc\bar{a}, b\bar{a}, cb, ec\}$$



1. Füllen Sie folgende Überdeckungstabelle für f_3 aus. (3 Punkte)

		j		
k	PI		p_i	c_i
0	$\bar{e}\bar{c}\bar{a}$		A	3
1	$\bar{e}d\bar{a}$		B	3
2	$dc\bar{a}$		C	3
3	$b\bar{a}$		D	2
4	cb		E	2
5	ec		F	2

2. Bestimmen Sie den Petrick-Ausdruck. (1 Punkt)

$PA(f_3) =$

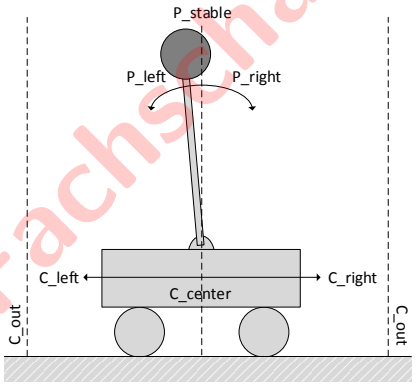
3. Bestimmen Sie alle kostenminimalen DMFs von f_3 mittels des Petrick-Verfahrens. (2 Punkte)

Aufgabe 3 (Automaten) (16 Punkte)

a) Im Folgenden soll ein Automat zur Steuerung eines inversen Pendels entworfen werden. Bei einem inversen Pendel wird ein Pendelstab senkrecht nach oben zeigend balanciert, wobei sich das Pendel in seinem höchsten Punkt in einer instabilen Ruhelage befindet, d.h. ohne Stabilisierung würde das Pendel herunterfallen. Das Stabilisieren des Pendels wird mittels eines Wagens realisiert, der durch Ausgleichsbewegungen der Schwankung des Pendels entgegenwirkt. Hierfür verfügt der Automat über vier verschiedene Eingaben, die jedoch nicht gleichzeitig anliegen können, und fünf verschiedene Ausgaben.

Eingaben:

- Pendel bewegt sich nach links (Eingabe *P_left*)
- Pendel bewegt sich nach rechts (Eingabe *P_right*)
- Pendel befindet sich in Ruhelage (Eingabe *P_stable*)
- Wagen befindet sich außerhalb des erlaubten Bereichs (Eingabe *C_out*).



Ausgaben:

- Wagen soll sich nach links bewegen (Ausgabe *C_left*)
- Wagen soll sich nach rechts bewegen (Ausgabe *C_right*)
- Wagen soll stoppen (Ausgabe *C_stop*)
- Wagen soll nach Verlassen des erlaubten Bereichs wieder zentral ausgerichtet werden (Ausgabe *C_center*).
- Aufschwingen des Pendels (Ausgabe *C_swingUp*).

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

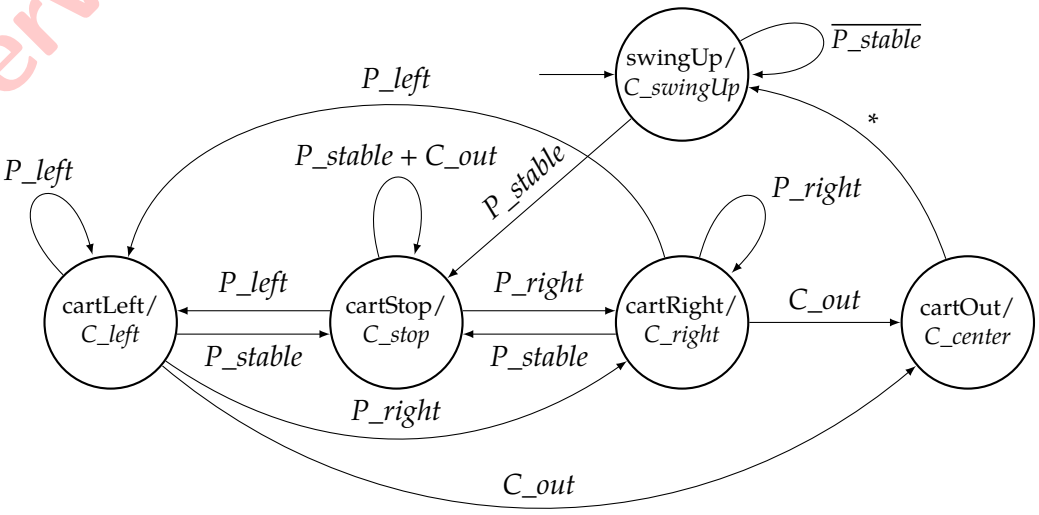
Eingabe	i_1	i_0
P_left	0	0
P_right	0	1
P_stable	1	0
C_out	1	1

Eingaben des Automaten

Ausgabe	o_2	o_1	o_0
C_left	0	0	0
C_right	0	0	1
C_stop	0	1	0
C_center	0	1	1
C_swingUp	1	0	0

Ausgaben des Automaten

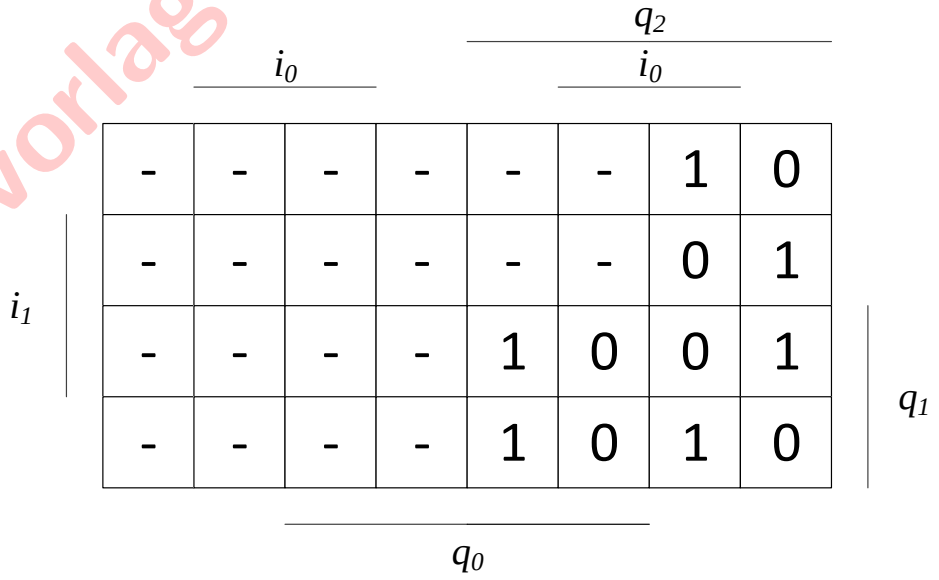
Das Verhalten des Automaten ist in folgendem Automatengraphen spezifiziert:



1. 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
swingUp	0	0	0	0	0										
swingUp	0	0	0	0	1										
swingUp	0	0	0	1	0										
swingUp	0	0	0	1	1										
cartStop	0	0	1	0	0										
cartStop	0	0	1	0	1										
cartStop	0	0	1	1	0										
cartStop	0	0	1	1	1										
cartLeft	0	1	0	0	0										
cartLeft	0	1	0	0	1										
cartLeft	0	1	0	1	0										
cartLeft	0	1	0	1	1										
cartRight	0	1	1	0	0										
cartRight	0	1	1	0	1										
cartRight	0	1	1	1	0										
cartRight	0	1	1	1	1										
cartOut	1	0	0	0	0										
cartOut	1	0	0	0	1										
cartOut	1	0	0	1	0										
cartOut	1	0	0	1	1										

2. 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 1.
(3 Punkte)



3. Zeichnen Sie ein Schaltwerk für den in Teilaufgabe 1 realisierten Automaten. Verwenden Sie dazu die in Teilaufgabe 2 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)

4. Geben Sie den Typen des in Teilaufgabe a) spezifizierten Automaten an. (1 Punkt)

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. Bei einem Mealy-Automat ist die Ausgabe eine beliebige Funktion, abhängig alleine vom Zustand. ☐ wahr ☐ falsch
2. Ein Active-High-RS-Latch lässt sich mit einer Schaltung bestehend aus zwei NAND-Gattern realisieren. ☐ wahr ☐ falsch
3. Mit einer Schaltung bestehend aus einem Inverter und einem Und-Gatter lassen sich negative Flanken erkennen. ☐ wahr ☐ falsch

Aufgabe 4 (Codierung und Rechnerarithmetik) (16 Punkte)

a) Gegeben ist eine Zeichenquelle bestehend aus folgenden Zeichen mit ihren zugehörenden Häufigkeiten:

Zeichen	A	B	C	D	E
Häufigkeit [in %]	25	31	7	8	29

1. Was ist der Informationsgehalt von Zeichen A? (1 Punkt)

2. Wie ist die Vorschrift zur Berechnung der Entropie dieser Quelle? (1 Punkt)

3. 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)

4. Wann spricht man in diesem Zusammenhang von einem optimalen Code? (1 Punkt)

b) Gegeben sei eine Quelle, deren Alphabet aus 3-Bit-Binärwörtern $x_3x_2x_1$ besteht.

1. Erweitern Sie das Binärwort 001 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 001 entsprechend dieser Vorschrift. (2 Punkte)

3. Nennen Sie einen Vorteil und einen Nachteil des Hamming-Code-Verfahrens gegenüber dem Paritätsprüfverfahren. (1 Punkt)

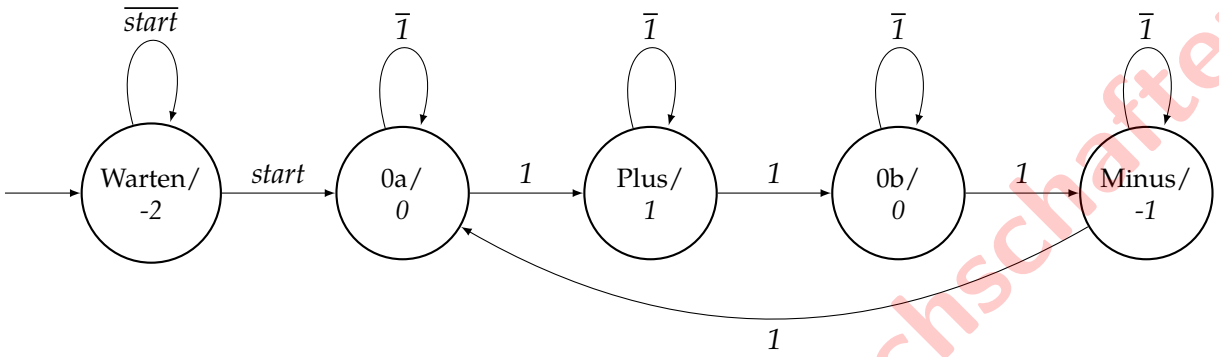
c) Zeichnen Sie unter ausschließlicher Verwendung von NOR-Gattern die Schaltnetze folgender Komponenten:

1. Halbaddierer (3 Punkte)

2. Volladdierer (3 Punkte)

Aufgabe 5 (VHDL) (16 Punkte)

MLT-3 ist ein in der Nachrichtentechnik verwendeter Leitungscode mit drei Spannungspegeln, die als +, 0 und - bezeichnet werden. Diese Spannungspegel werden in der festen Folge [0, +, 0, -] durchlaufen. Bei jeder zu codierenden 1 wird der nächste Pegel in der Folge ausgegeben; bei einer 0 ändert sich das Ausgabesignal nicht. Folgender endlicher Automat realisiert dieses Verhalten, wobei + mit 1 und - mit -1 codiert wird:

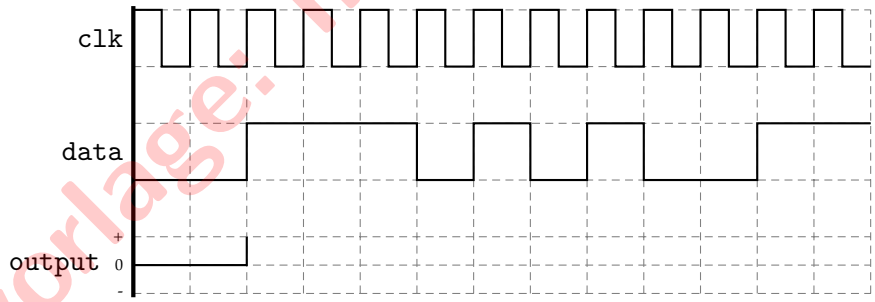


Zuerst wird gewartet, bis das Startsignal `start` anliegt. In den anderen Zuständen wird entsprechend dem Eingabebit `data` die Ausgabe `output` erzeugt.

Implementieren Sie im Folgenden den spezifizierten Automaten als *synchrone* Schaltung, die sich ebenfalls *synchron* zurücksetzen lässt. Folgender Datentyp stehe zur Verfügung:

```
type mlt3_state is (mlt3_warten, mlt3_0a, mlt3_plus, mlt3_0b, mlt3_minus);
```

a) Vervollständigen Sie folgendes Wellenformdiagramm, indem Sie `data` mit dem MLT-3-Verfahren codieren und bei `output` einzeichnen. (2 Punkte)



b) Vervollständigen Sie folgende Funktion, die die Zustandüberführungsfunktion δ implementiert, also für eine gegebene Kombination von Zustand und Eingabe den nächsten Zustand zurückliefert. (3 Punkte)

```
function get_next_state(state: mlt3_state, data: std_logic,
    start: std_logic) return mlt3_state is
begin
```

```
end function;
```

c) Vervollständigen Sie folgende Funktion, die die Ausgabefunktion λ implementiert, also für einen gegebenen Zustand die entsprechende Ausgabe zurückliefert. Im Falle eines ungültigen Zustands soll -2 zurückgegeben werden. (2 Punkte)

```
function get_output(state: mlt3_state) return integer is
begin
```

```
end function;
```

d) Vervollständigen Sie die folgende `entity`-Deklaration unter Beachtung der vorherigen Teilaufgaben. Die Ausgabe `output` soll als `std_logic_vector` übermittelt werden. (3 Punkte)

```
entity mlt3 is
```

```
end mlt3;
```

e) Vervollständigen Sie den Rumpf der folgenden `architecture` unter Verwendung der beiden obigen Funktionen. (3 Punkte)

```
architecture behavioral of mlt3 is
  signal state : mlt3_state := mlt3_warten;
  signal output_int : integer := 0;
begin
  -- vorgegeben: Ein-/Ausgabe in korrekte Typen umwandeln
  output <= integer_to_std_logic_vector(output_int);
```

```
end architecture;
```

f) Geben Sie die Werte der Signale `d1`, `d2` und `d3` nach dem nächsten Simulationsschritt des folgenden `process` an, wenn zu dessen Beginn gilt: `d1="110"`, `d2="111"` und `d3="001"`. Alle Signale sind vom Typ `std_logic_vector(2 downto 0)`. (3 Punkte)

```
-- anfangs: d1="110", d2="111", d3="001"
process(d1, d2, d3)
  variable temp : std_logic_vector(2 downto 0) := "000";
begin
  temp := d2 and d3;
  d1 <= temp xor temp;
  temp := not temp;
  d3 <= d3 and temp;
  d2 <= d1;
end process;
```