Klausur

Technische Informatik

Prof. Dr. Bernd Becker
Dipl-.Inf. Karsten Scheibler
M. Sc. Dominik Erb

Freiburg, 06. März 2015

- **Bearbeitungszeit:** 120 Minuten
- **Erlaubte Hilfsmittel:** Keine
- Das Erreichen von **55** Punkten ist hinreichend zum Bestehen der Klausur

Aufgabe 1 (7 + 2 Punkte)

Gegeben sei der *Schaltkreis* $SK_1 := (X_3, G.typ, IN, Y_2)$ mit

$$X_2 = (x_1,\ x_2,\ x_3)$$
$$Y_2 = (v_5,\ v_8)$$
$$G = (V,\ E)$$
$$V = \{x_1,\ x_2,\ x_3\} \cup v_1,\ v_2,\ v_3,\ v_4,\ v_5,\ v_6,\ v_7,\ v_8\} \cup \{0,\ 1\}$$
$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}\}$$

Die Abbildungen Q und Z sind durch Tabelle 1.1, die Abbildungen typ und IN durch Tabelle 1.2 gegeben:

Tabelle 1.1: Q und Z

$e_i \in E$	$Q(e_i)$	$Z(e_i)$
e_1	x_1	v_1
e_2	x_1	v_2
e_3	x_1	v_3
e_4	x_2	v_1
e_5	x_2	v_2
e_6	x_2	v_4
e_7	x_3	v_3
e_8	x_3	v_4
e_9	x_3	v_5
e_{10}	x_1	v_6
e_{11}	v_2	v_5
e_{12}	v_3	v_6
e_{13}	v_3	v_7
e_{14}	v_4	v_7
e_{15}	v_6	v_8
e_{16}	v_7	v_8

1. Zeichnen Sie SK_1.
2. Bestimmen Sie die *Kosten* sowie die *Tiefe* von SK_1

Tabelle 1.2: IN und typ

$v_i \in V$	$IN(v_1)$	$typ(v_i)$
v_1	(e_1, e_4)	OR_2
v_2	(e_2, e_5)	XOR_2
v_3	(e_3, e_7)	OR_2
v_4	(e_6, e_8)	OR_2
v_5	(e_9, e_{11})	XOR_2
v_6	(e_{10}, e_{12})	AND_2
v_7	(e_{13}, e_{14})	AND_2
v_8	(e_{15}, e_{16})	AND_2

Aufgabe 2 (3 + 6 Punkte)

1. Ist der Code $A \mapsto 00, B \mapsto 01, C \mapsto 101, D \mapsto 110, E \mapsto 010$ ein Huffmancode? Begründen Sie Ihre Aussage

2. Bestimmen Sie die Häufigkeiten der einzelnen Zeichen des Wortes `SOMMERSEMESTER`. Konstruieren Sie für diese Zeichen einen Huffman-Baum und geben Sie einen entsprechenden Huffmancode dafür an.

Aufgabe 3 (2 + 5 + 5 Punkte)

1. Bei einer Datenübertragung soll der Hammingcode verwendet werden. Die Wortbreite beträgt 60 Bit. Wieviele zusätzliche Prüfbits werden für die Übertragung benötigt? Begründen Sie!
2. a soll mit Hilfe des Hammingcodes übertragen werde, die Prüfbits wurden aber noch nicht hinzugefügt. Die niederwertigste Stelle ist dabei ganz rechts, und Prüfbits ergänzen die relevanten Stellen auf *gerade* Parität. Geben Sie das tatsächlich zu übertragende Codewort an.
3. Bei einer weiteren Übertragung entstand an genau einer Bitstelle ein Fehler. Finden Sie heraus, welches Bit falsch übertragen wurde und geben Sie das korrigierte Wort an.

Aufgabe 4 (10 Punkte)

Sei die Funtion $f : \mathbb{B}^4 \mapsto \mathbb{B}$ durch ihre **ON**-Menge

$$ON(f) = \{0001, 0010, 0011, 0110, 0111, 1000, 1001, 1010\}$$

gegeben.

Bestimmen Sie alle Primimplikanten nach dem *Verfahren von Quine-McCluskey*. Geben Sie all Zwischenschritte, d.h. alle Mengen $L_i^{M(f)}$ und $Prim(f)$, an.

Hinweis: Sie dürfen die abkürzende Schreibweise für Monome verwenden, d.h. statt z.B. $\overline{x_1}x_2x_4$ können Sie `01-1` schreiben.

Aufgabe 5 (2 + 9 Punkte)

1. Geben Sie die Interpretationsfunktion $[\cdot]_1$ für *Einerkomplementzahlen* mit $n+1$ Vor- und k Nachkommastellen (also $d_n d_{n-1} \dots d_1 d_0 d{-}1 \dots d{-}k$) an.

2. Zeigen Sie, dass sich die Negation einer ganzen Einerkomplementszahl, d.h. einer Einerkomplementzahl ohen Nachkommastelle, durch das Komplementieren aller Bits der Zahl umsetzen lässt:

$$-[a]_1 = [\bar{a}]_1$$

Hinweise:

- $[\bar{a}]_1$ ist die Festkommazahl im Einerkomplement, die aus $[a]_1$ durch Komplementieren aller Bits hervorgeht, d.h. $\bar{0} = 1$ und $\bar{1} = 0$

- Für den Beweis darf die folgenden Gleichung unbewiesen verwendet werden:

$$\sum_{i=-k}^{n} -12^i = 2^n - 2^{-k}$$

Aufgabe 6 (7 Punkte)

Sei $(M, \cdot, +, \bar{\ })$ eine Boolsche Algebra. Ferner sei der $\oplus$-Operator wie folgt definiert:

$$x \oplus y = (x + y) \cdot (\bar{x} + \bar{y}) \ \forall x, y \in M$$

Zeigen Sie formal:

$$\bar{x} \oplus y = \overline{x \oplus y} \ \forall x, y \in M$$

Verwenden Sie hierfür ausschließlich die Axiome der Booleschen Algebra, die de-Morgan-Regel, das doppelte Komplement und die Definition des $\oplus$-Operators.

Hinweise:

- Geben Sie in jedem Schritt an, welche Regel Sie benutzt haben.
- Ein “Beweis” durch Funktionstabellen ist **nicht** zulässig, da diese Aussage für jede Boolesche Algebra gelten soll.

Axiome der Booleschen Algebra, de-Morgan-Regel und doppeltes Komplement:

(i)	Kommutativität	$x + y = y + x$	$x \cdot y = y \cdot x$
(ii)	Assoziativität	$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) + z$
(iii)	Absorption	$x + (x \cdot y) = x$	$x \cdot (x + y) = x$
(iv)	Distributivität	$x + (y \cdot z) = (x + y) \cdot (x + z)$	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
(v)	Komplementregel	$x + (x \cdot \bar{y}) = x$	$x \cdot (x + \bar{y}) = x$
(vi)	de-Morgan	$\overline{(x + y)} = \bar{x} \cdot \bar{y}$	$\overline{(x \cdot y)} = \bar{x} + \bar{y}$
(vii)	doppeltes Komplement	$\bar{\bar{x}} = x$	

Aufgabe 7 (3 + 4 + 4 + 5 Punkte)

Gegeben sei nun die Boolesche Funktion

$$f : \mathbb{B}^3 \mapsto \mathbb{B}, f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \oplus$$

1. Wann ist ein BDD geordnet? Wann ist ein BDD reduziert? Wann ist ein BDD eine kanonische Darstellung einer Booleschen Funktion?
2. Geben Sie für f eine kanonische disjunktive Normalform sowie ein Minimalpolynom an.
3. Erstellen Sie für f einen vollständig reduzierten und geordneten BDD (ROBDD) in der Variablenordnung Ihrer Wahl. Vergessen Sie nicht den ROBDD eindeutig zu beschriften!
4. Skizzieren Sie den Schaltkreis für f. Verwenden Sie ausschließlich Schaltelemente aus der Bibliothek $LSTD := \{NOT, OR, AND\}$ und kennzeichnen Sie Ein- und Ausgänge in geeigneter Weise.

Aufgabe 8 (2 + 7 + 7 Punkte)

Betrachten Sie die hierarchischen Realisierungen eines 3-Bit-Addierers in Abbildung 1. Die Verzögerungszeiten für die Gatter der Bausteinfamilie NANGATE v2 sind in Tabelle 3 angegeben. Die Anstiegs- und Abfallzeiten an den primären Eingängen sind kleiner als $\delta = 2.5ns$. Weiterhin sind die Anstiegs- und Abfallzeiten an den Ausgängen eines Gatters kleiner als δ, falls die Anstiegs- und Abfallzeiten an den Eingängen des Gatters kleiner als δ sind.
Alle primären Eingänge schalten zum Zeitpunkt t_0, d.h. sie durchlaufen M zum Zeitpunkt t_0.

1. Bestimmen Sie die Kosten $C(ADD_3)$ und die Tiefe $depth(ADD_3)$ des 3-Bit-Addierers.
2. Bis zu welchem Zeitpunkt liegt an s_0 von ADD_3 mindestens der alte logisch Wert an und ab welchem Zeitpunkt liegt an s_0 sicher der neue logische Wert an?
3. Geben Sie den längsten Pfad von ADD_3 an und besteimmen Sie für dessen Ausgang den Zeitpunkt zu dem der neue logische Wert sicher anliegt.

Aufgabe 9 (2 + 4 + 6 Punkte)

Gegeben sie der Mealy-Automat $M = (I, O, S, S_0, \delta, \lambda)$ in Tabelle 9.1 mit:
$I = \{0, 1\}, O = \{0, 1\}, S = \{s_0, s_1, s_2, s_3\}$ und $S_0 = \{s_0\}$. δ und λ sind über das Zustandsdiagramm in Abbildung 9.1 gegeben.

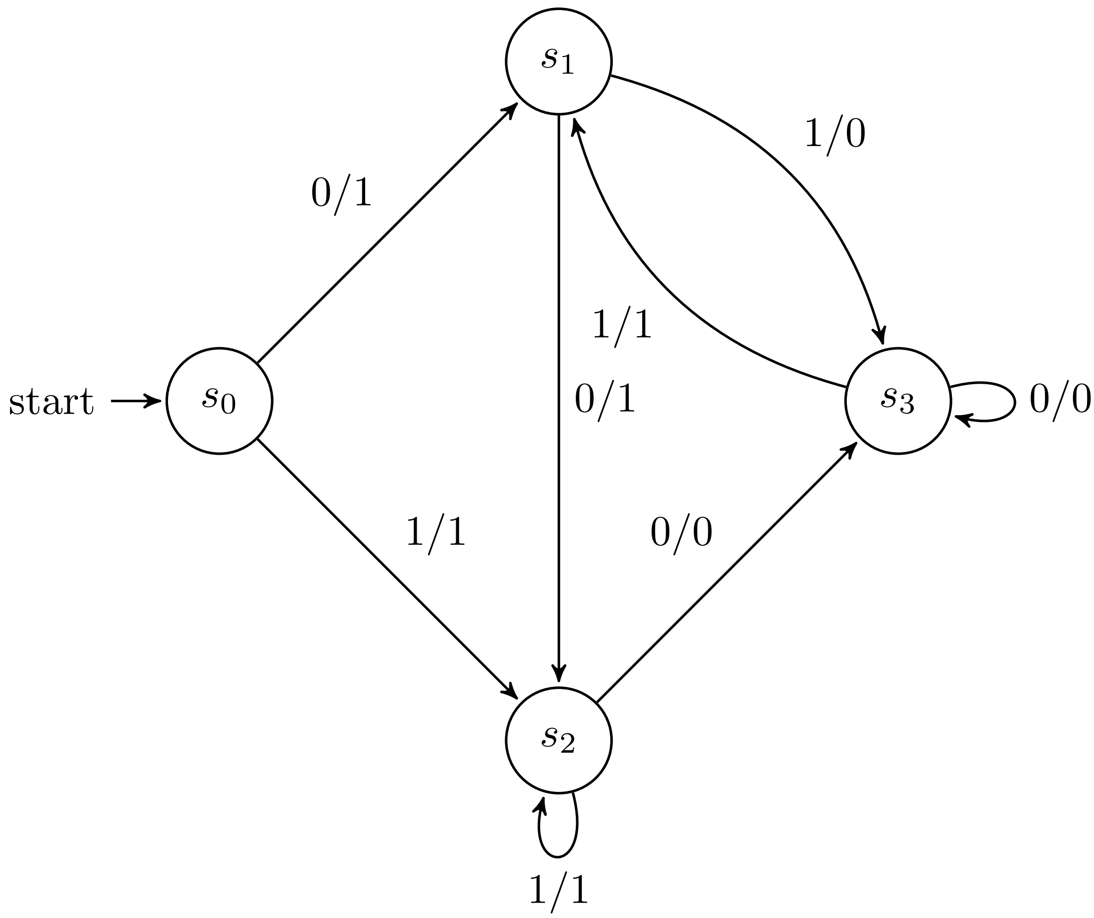

Abbildung 9.1: Mealy-Automat M

1. Welche Ausgabe erhält man und in welchem Zusant befindet sich der Automat M nach der Eingabe von $X = 101010$ (das linke Bit ist das erste Eingabe-Bit)?

Tabelle 9.1: Zustandsübergangstafel für M

x	s	s	λ

2. Füllen Sie die Zustandsübergangstafel für M (Tabelle 9.1) vollständig aus.
   s bezeichne hierbei den aktuellen Zustand, s' den Folgezustand und x den Eingabewert.

3. Ein Zustand $s \in S$ sei mit Hilfe von Booleschen Zustandsvariablen z_0 und z_1 codiert: $s = (z_1, z_0)$. Für die einzelnen Zustände von M sei die folgende Codierung gewählt:
   $s_o = (0,0), s_1 = (0,1), s_2 = (1,0), s_3 = (1,1)$.
   Geben Sie eine Boolesche Funktion $\lambda : \mathbb{B}^3 \mapsto \mathbb{B}, \lambda(z_0, z_1, z_2)$ für die Ausgabe des Automaten M an (x sei hierbei wieder der Eingabewert). Erläutern Sie Ihr Vorgehen.

Aufgabe 10 (3 + 9 Punkte)

```
0 LOADI IN2 1^24
1 LOAD ACC 0
2 ADDI ACC 1
3 OR ACC 0
4 STORE 0
5 JUMP≥ -4
```

1. Kommentieren Sie ausführlich die Auswirkungen jedes einzelnen Befehles des Programms. Eine Befehlsübersicht des ReTI-Rechners finden Sie am Ende der Klausur.

2. Das Programm ist in den Speicherzellen $M[0]$ bis $M[5]$ abgelegt. Wie ändert sich der Inhalt der Speicherzelle $M[0]$ während des Programmablaufs?
   Welchen Inhalt hat die Speicherzelle $M[0]$ nach Ablauf des Programms?
   Wie oft wird die Schleife durchlaufen? Begründen Sie Ihre Antwort.

Hinweise:

- Befehlscodierung für `LOADI IN2`: $I[31:24] = 01110010$.
- Beachten Sie die Notation aus der Vorlesung:
  `b^j` $=$ j-mal b mit `$b \in {0, 1}`

Aufgabe 11 (6 Punkte)

Markieren Sie in dem Datenpfadschaubild der ReTI all diejenigen Pfade, die in der Execute-Phase bei der Ausführung des Befehls `SUB IN2 100` benötigt werden.Eine Übersicht über die Befehle der ReTI befindet sich am Ende der Klausur.

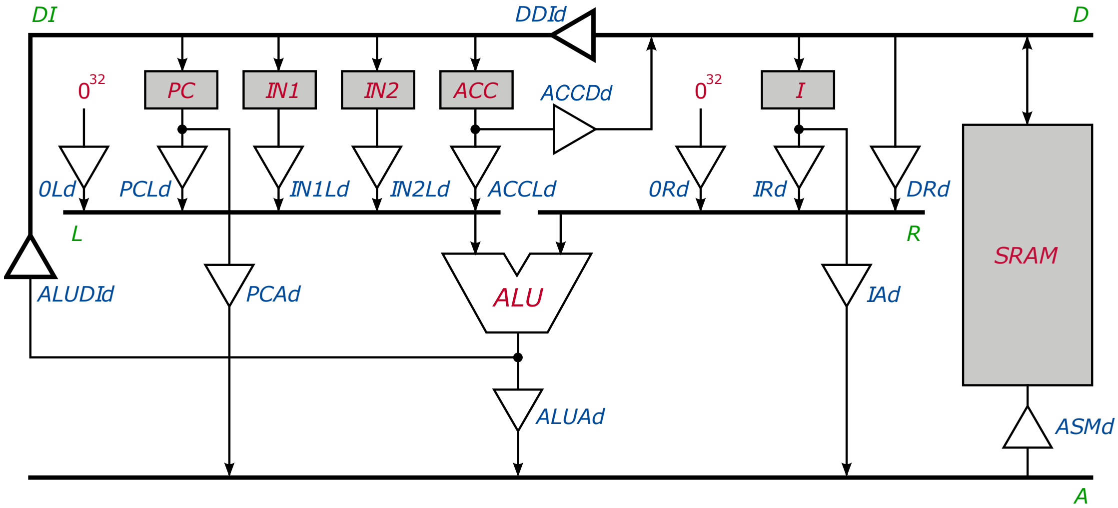

Abbildung 11.1: Datenpfade der ReTI

Load-Befehle		$I[25:24] = D$	
$I[31:28]$	Befehl	Wirkung	
0100	LOAD D i	$D := M(\langle i\rangle)$	$\langle PC\rangle := \langle PC\rangle + 1$ falls $D \neq PC$
0101	LOADIN1 D i	$D := M(\langle IN1\rangle + [i])$	
0110	LOADIN2 D i	$D := M(\langle IN2\rangle + [i])$	
0111	LOADI D i	$D := 0^8 i$	

Store-Befehle		MOVE: $I[27:24] = SD$	
$I[31:28]$	Befehl	Wirkung	
1000	STORE i	$M(\langle i\rangle) := ACC$	
1001	STOREIN1 i	$M(\langle IN1\rangle + [i]) := ACC$	$\langle PC\rangle := \langle PC\rangle + 1$
1010	STOREIN2 i	$M(\langle IN2\rangle + [i]) := ACC$	
1011	MOVE S D	$D := S$	$\langle PC\rangle := \langle PC\rangle + 1$ falls $D \neq PC$

Compute-Befehle		$I[25:24] = D$	
$I[31:26]$	Befehl	Wirkung	
000010	SUBI D i	$[D] := [D] - [i]$	
000011	ADDI D i	$[D] := [D] + [i]$	
000100	OPLUSI D i	$D := D \oplus 0^8 i$	$\langle PC\rangle := \langle PC\rangle + 1$ falls $D \neq PC$
000101	ORI D i	$D := D \vee 0^8 i$	
000110	ANDI D i	$D := D \wedge 0^8 i$	
001010	SUB D i	$[D] := [D] - [M(\langle i\rangle)]$	
001011	ADD D i	$[D] := [D] + [M(\langle i\rangle)]$	
001100	OPLUS D i	$D := D \oplus M(\langle i\rangle)$	$\langle PC\rangle := \langle PC\rangle + 1$ falls $D \neq PC$
001101	OR D i	$D := D \vee M(\langle i\rangle)$	
001110	AND D i	$D := D \wedge M(\langle i\rangle)$	

Jump-Befehle		
$I[31:27]$	Befehl	Wirkung
11000	NOP	$\langle PC\rangle := \langle PC\rangle + 1$
11001	$JUMP_{>} i$	$\langle PC\rangle := \begin{cases} \langle PC\rangle + [i] & \text{falls } [ACC]\ c\ 0 \\ \langle PC\rangle + 1 & \text{sonst} \end{cases}$
11010	$JUMP_{=} i$	
11010	$JUMP_{\geq} i$	
11011	$JUMP_{<} i$	
11100	$JUMP_{\neq} i$	
11110	$JUMP_{\leq} i$	
11111	JUMP i	$\langle PC\rangle := \langle PC\rangle + [i]$

Abbildung 11.2: Befehlstabelle der ReTI