Rechnerstrukturen SS 2000
bei Prof. Lakemeyer

Lösung zur Scheinklausur

erstellt von Sandip Sar-Dessai

Aufgaben

Aufgabe 1 (2+2+2 Punkte)

1. Wie lautet die Ziffernschreibweise von $(114)_{10}$ im Dualsystem?
2. Wie lautet die Ziffernschreibweise von $(10110010)_2$ im Dezimalsystem?
3. Wie lautet die Ziffernschreibweise von $(DC9)_{16}$ im Dezimalsystem?

Aufgabe 2 (11 Punkte) Skizzieren Sie eine Schaltung aus Und-Gattern, Oder-Gattern und Invertern mit vier Eingängen x_1, x_2, x_3, x_4 und einem Ausgang y, so daß $y = 1$ genau dann gilt, wenn $x_1x_2x_3x_4$ der BCD-Code (*Binary Coded Decimal*-Code) einer Dezimalziffer ist.

Aufgabe 3 (11 Punkte) Verwenden Sie das untenstehende Karnaugh-Diagramm, um das Minimalpolynom der durch die Wertetabelle gegebenen Booleschen Funktion f zu bestimmen.

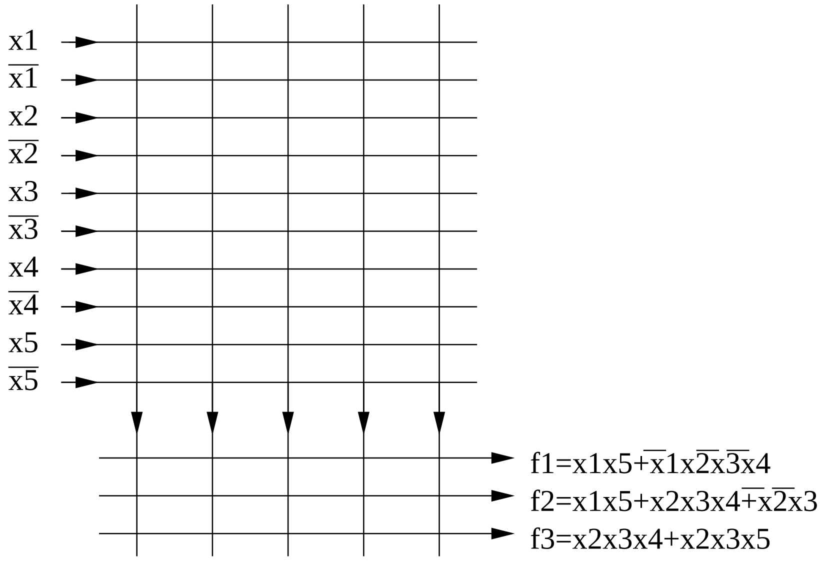
x3x4 \ x1x2	00	01	11	10
00				
01				
11				
10				

x1 x2 x3 x4	f(x)	x1 x2 x3 x4	f(x)
0 0 0 0	1	1 0 0 0	0
0 0 0 1	0	1 0 0 1	0
0 0 1 0	1	1 0 1 0	1
0 0 1 1	1	1 0 1 1	1
0 1 0 0	0	1 1 0 0	0
0 1 0 1	0	1 1 0 1	0
0 1 1 0	0	1 1 1 0	1
0 1 1 1	1	1 1 1 1	1

Aufgabe 4 (7 Punkte) Skizzieren Sie, wie sich die Boolesche Funktion f aus Aufgabe 3 mittels eines Multiplexers mit drei Steuerleitungen (3-MUX) realisieren läßt, wenn x_1, x_2, x_3 an die Steuerleitungen angelegt werden.

Aufgabe 5 (6 Punkte) Skizzieren Sie, wie sich die Boolesche Funktion f aus Aufgabe 3 mittels eines 4x16-Decoders und eines Oder-Gatters mit beliebigem Fan-In realisieren läßt.

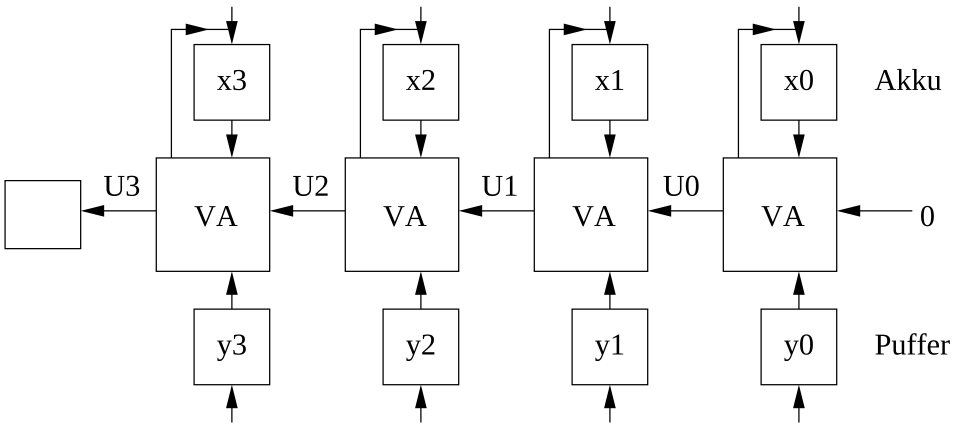
Aufgabe 6 (6 Punkte) Skizzieren Sie, wie sich die unten angegebenen Booleschen Funktionen f_1, f_2, f_3 mittels eines PLA realisieren lassen, indem Sie die entsprechenden Punkte in das folgende punktorientierte PLA eintragen.



Aufgabe 7 (10 Punkte) Die untenstehende Abbildung zeigt den Aufbau eines 4-Bit-Parallel-Addierwerks (aus Volladdierern); es berechnet $(x_1x_2x_3x_4)_2 + (y_1y_2y_3y_4)_2$.
Skizzieren Sie ein ganz ähnlich aufgebautes 4-Bit-Parallel-Subtrahierwerk für das Zweier-Komplement, das

$$(x_1x_2x_3x_4)_2 + K_2((y_1y_2y_3y_4)_2) = (x_1x_2x_3x_4)_2 - (y_1y_2y_3y_4)_2$$

berechnet.



Aufgabe 8 (2+3 Punkte)

1. Wie wird -70 als achtstellige Dualzahl im Zweier-Komplement dargestellt?
2. Führen Sie die Subtraktion 65-28 mit achtstelligen Dualzahlen im Zweier-Komplement durch.

Aufgabe 9 (3+2 Punkte)

1. Tragen Sie die Darstellung von 5,375 als Gleitkomma-Zahl in normalisierter Form (mit Exponenten-Basis 2) ein:

Vorzeichen: 1 Bit	Mantisse: 23 Bits	Exponent: 8 Bits

2. Wie ist die Excess-128-Darstellung von 5,375?

Vorzeichen: 1 Bit	Mantisse: 23 Bits	Exponent: 8 Bits

Aufgabe 10 (7+7+7 Punkte)

1. Was ist der "von-Neumannsche Flaschenhals?"
   Wie versucht man, damit umzugehen?

2. Wofür stehen die Abkürzungen RAM, SRAM, DRAM?
   Was ist damit jeweils gemeint?

3. Was ist unter dem virtuellen Speicher im PowerPC zu verstehen?

Aufgabe 11 (12 Punkte) Schreiben Sie ein Assembler-(Unter-)Programm für den PowerPC als `asm`-Funktion[1] `contains101`, die ein Word (=32 Bit) als Parameter erhält und 1 bzw. 0 zurückgibt, wenn der Parameter die Bitfolge 101 enthält bzw. nicht enthält. Verwenden Sie nur die auf der letzten Seite aufgelisteten Befehle, und kommentieren Sie Ihr Programm möglichst gut.

```
asm int contains101(int)
{
    ...
}
```

Anlage: PowerPC-Befehle für 32-Bit-Mikroprozessoren

[1] Allgemeine Hinweise zu asm-Funktionen:

- Die Parameter einer `asm`-Funktion stehen bei Beginn der Funktionsausführung der Reihe nach in den Registern `r3`, `r4`, `r5` usw.
- Falls eine `asm`-Funktion einen Rückgabewert berechnet, so soll dieser nach Ausführung der Funktion im Register r3 stehen.
- Beim Aufruf einer `asm`-Funktion (aus dem "Hauptprogramm" heraus) wird die zugehörige Rücksprungadresse ins Link-Register LR geschrieben.

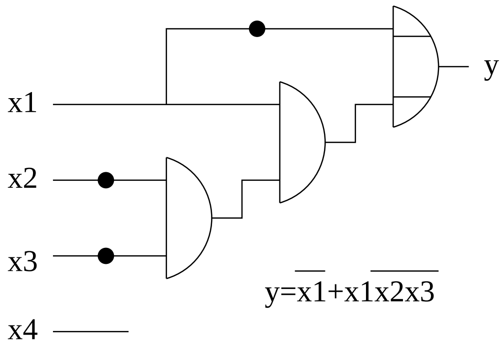
Lösungen

Aufgabe 1 (2+2+2 Punkte)

1. $(114)_{10} = (2^6 + 2^5 + 2^4 + 2^1)_{10} = (1110010)_2$
2. $(10110010)_2 = (2^1 + 2^4 + 2^5 + 2^7)_{10} = (178)_{10}$
3. $(DC9)_{16} = (9 \cdot 16^0 + 12 \cdot 16^1 + 13 \cdot 16^2)_{10} = (3529)_{10}$

Aufgabe 2 (11 Punkte)

x1 x2 x3 x4	f(x)	x1 x2 x3 x4	f(x)
0 0 0 0	1	1 0 0 0	1
0 0 0 1	1	1 0 0 1	1
0 0 1 0	1	1 0 1 0	0
0 0 1 1	1	1 0 1 1	0
0 1 0 0	1	1 1 0 0	0
0 1 0 1	1	1 1 0 1	0
0 1 1 0	1	1 1 1 0	0
0 1 1 1	1	1 1 1 1	0



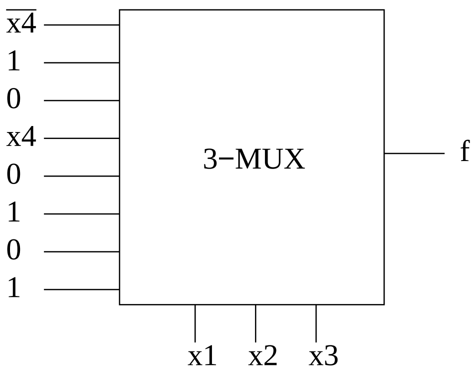
Aufgabe 3 (11 Punkte)

x3x4 \ x1x2	00	01	11	10
00	1			
01				
11	1	1	1	1
10	1		1	1

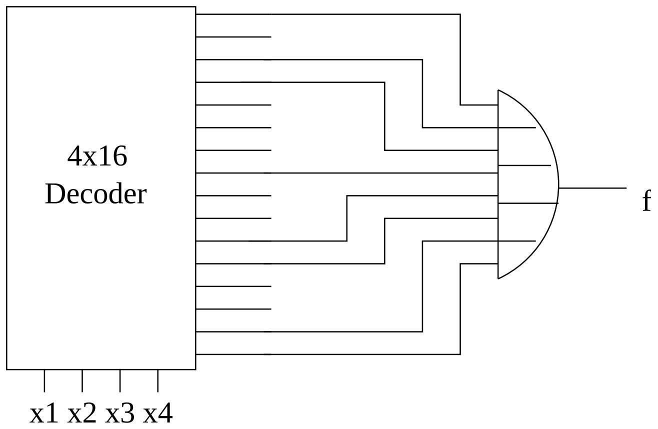
Minimalpolynom zu f: $\overline{x_1 x_2 x_4} + x_3 x_4 + x_1 x_3$

Aufgabe 4 (7 Punkte)

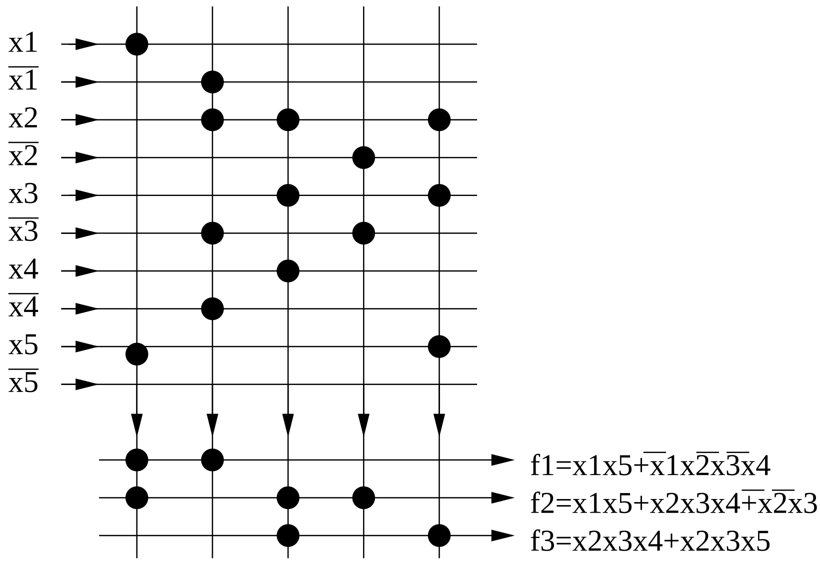
x1	x2	x3	f
0	0	0	$\overline{x4}$
0	0	1	1
0	1	0	0
0	1	1	x4
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Aufgabe 5 (6 Punkte)

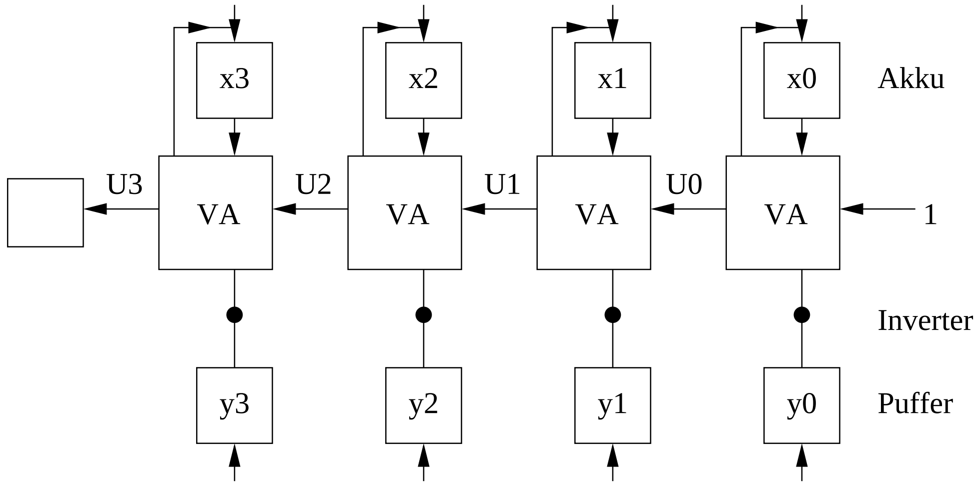


Aufgabe 6 (6 Punkte)



Aufgabe 7 (10 Punkte)

Funktionsweise: Bei der K_2-Subtraktion wird der Subtrahend zunächst invertiert ($K_2(y)$), dann werden die beiden Zahlen addiert und schließlich wird eine Eins aufaddiert. Die Addition der Eins erfolgt durch Einspeisen einer Eins als ersten Übertrag.



Aufgabe 8 (2+3 Punkte)

1. Es ist $(70)_{10} = (2^6 + 2^2 + 2^1)_{10} = (1000110)_2$. Dann gilt:

$$K_2^8((70)_{10}) = K_2^8((01000110)_2) = (10111001)_2 + (00000001)_2 = (10111010)_2$$

2. Es ist $(65)_{10} = (01000001)_2$ und $(28)_{10} = (00011100)_2$ sowie $K_2((28)_{10}) = (11100100)_2$. Die Subtraktion $65 - 28$ entspricht einer Addition von $65 + K_2(28)$, mit gewohntem Rechenverfahren ergibt sich $(00100101)_2 = (37)_{10}$.

Aufgabe 9 (3+2 Punkte)

1. Zunächst einmal ist

$$(5,375)_{10} = (101,011)_2 = (0,101011)_2 \cdot 2^3$$

Also Mantisse 101011 (bzw. wegen Hidden Bit 01011) und Exponent 11:

0   010 1100 0000 0000 0000 0000   0000 0011

2. Änderung des Exponenten $Excess(3) = (3 + 127)_2 = (10000011)_2$:

0   010 1100 0000 0000 0000 0000   1000 0011

Aufgabe 10 (7+7+7 Punkte)

1. Durch immer schnellere Befehlsausführungszeiten der CPU wird immer stärker der Datenkanal zwischen CPU und Speicher zum Engpass: denn er kann die benötigten Daten nicht in einer beliebigen Geschwindigkeit liefern. Diesen architekturbedingten Umstand nennt man von-Neumannschen Flaschenhals. Man versucht das Problem zu umgehen, indem in der CPU selbst ein Zwischenspeicher (Cache) verbaut wird, der die zuletzt genutzten Daten zwischenspeichert. Da der Cache die Daten wesentlich schneller liefert, und ca 90% der benötigten Daten "lokal" sind (also im Cache liegen), wird der Flaschenhals "ausgetrickst".

2. RAM ("Random Access Memory"): Hauptspeicher; über Adreßleitungen spezifiziert man eine Adresse, die über die Datenkanäle geladen bzw. geschrieben werden soll.
   SRAM ("Static RAM"): Speicher, dessen Inhalt auch dann erhalten bleibt, wenn keine Versorgungsspannung mehr anliegt.
   DRAM ("Dynamic RAM"): Im Gegensatz zum SRAM benötigt dieser Speichertyp regelmäßige Refreshes, damit die Daten erhalten bleiben. Dafür ist die Realisation i.A. kostengünstiger.

3. Da der Hauptspeicher relativ teuer ist (verglichen mit dem Hintergrundspeicher) nutzt man einen Teil des Hintergrundspeichers als "Erweiterung" des Hauptspeichers, sofern dieser voll ist. Diesen nicht vollständig physischen Hauptspeicher nennt man Hintergrundspeicher.

Aufgabe 11 (12 Punkte) Mögliche Lösung nach folgender Idee:

Es werden insgesamt 30 Durchläufe gemacht (denn die 101 kann in 32 Bit nur an 30 Stellen stehen); dabei wird die Zahl in jedem Schritt die unterste Stelle gelöscht (Rechtsshift), und das untere Ende wird auf 101 verglichen. Dazu wird eine Modulo-8-Operation (letzte drei Ziffer) ausgeführt, und das Ergebnis mit 5 (101) verglichen. Ist der Vergleich positiv, wird mit Rückgabe Eins beendet. Sind dagegen alle Durchläufe abgeschlossen, kam die 101 nicht vor, und es wird mit Null beendet.

Das ganze sieht etwa so aus:

```
asm int contains101 (int) {
    li r6, 30         # Schleifenzähler: max. 30 Durchläufe,
    mtctr r6          # danach kann keine 101 mehr drin stehen

  loop:
    srwi r4, r3, 3    # r4 := (r3 div 8)*8
    slwi r4, r4, 3    #    (letzte drei Stellen streichen)
    sub r5, r3, r4    # r5 := r3 mod 8 (== r5 := r3 - r4)
    cmpdi r5, 5       # Rest 5 (101)?
    beq contains      # => dann ist 101 drin
    bdnz loop         # ggf. nächster Durchlauf

    li r3, 0          # Falls bis jetzt keine 101,
    b return          # => Ergebnis 0

  contains:
    li r3, 1          # Ergebnis 1

  return:
    blr               # Ende
}
```

Keine Gewährleistung für die Richtigkeit der Lösungen!