



UNIVERSITÄT KARLSRUHE  
Fakultät für Informatik  
Institut für Rechnerentwurf und Fehlertoleranz (IRF)  
Prof. Dr. W. Karl

**Musterlösungen**  
zur Klausur „Technische Informatik I/II“  
am 02. September 2004, 9.00 - 11.00 Uhr

|                      |                          |                               |
|----------------------|--------------------------|-------------------------------|
| Name:<br><b>Bond</b> | Vorname:<br><b>James</b> | Matrikelnummer:<br><b>007</b> |
|----------------------|--------------------------|-------------------------------|

| <b>Technische Informatik I</b> |                   |
|--------------------------------|-------------------|
| Aufgabe 1                      | 11 von 11 Punkten |
| Aufgabe 2                      | 9 von 9 Punkten   |
| Aufgabe 3                      | 7 von 7 Punkten   |
| Aufgabe 4                      | 10 von 10 Punkten |
| Aufgabe 5                      | 7 von 7 Punkten   |

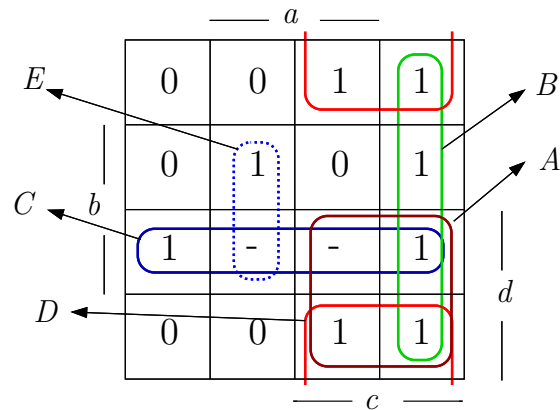
| <b>Technische Informatik II</b> |                   |
|---------------------------------|-------------------|
| Aufgabe 6                       | 8 von 8 Punkten   |
| Aufgabe 7                       | 9 von 9 Punkten   |
| Aufgabe 8                       | 10 von 10 Punkten |
| Aufgabe 9                       | 11 von 11 Punkten |
| Aufgabe 10                      | 7 von 7 Punkten   |

|                         |                   |
|-------------------------|-------------------|
| <b>Gesamtpunktzahl:</b> | 90 von 90 Punkten |
|-------------------------|-------------------|

|  |                         |
|--|-------------------------|
|  | <b>Note:</b> <b>1,0</b> |
|--|-------------------------|

# Aufgabe 1

1.  $f(d, c, b, a)$ :



Primimplikanten:

$$A : (dc) \qquad B : (\underline{c\bar{a}}) \qquad C : (\underline{db})$$

$$D : (\underline{c\bar{b}}) \qquad E : (\underline{\bar{c}ba})$$

2. Disjunktive Minimalform von  $f(d, c, b, a)$ :

$$\begin{aligned} f(d, c, b, a) &= B \vee C \vee D \vee E \\ &= \underline{c\bar{a}} \vee \underline{db} \vee \underline{c\bar{b}} \vee \underline{\bar{c}ba} \end{aligned}$$

3. Die Schaltfunktion ist unvollständig definiert, da

- Primimplikanten bei vollständig definierten Funktionen aus  $2^n$  Mintermen bestehen. Der Primimplikanten  $B$  überdeckt 3 Minterme, d. h. er muss noch eine Freistelle enthalten. ODER
- Primimplikant  $A$  ist im Primimplikanten  $B$  enthalten. Somit wäre  $A$  kein Primimplikant  $\rightarrow$  Widerspruch.

4. Kernprimimplikanten:  $C$  und  $D$

5. Überdeckungsfunktion:

$$\begin{aligned} \ddot{u}_g &= (A \vee B)(A \vee B)CD(B \vee D) \\ &= (A \vee B)CD(B \vee D) \\ &= (ACD \vee BCD)(B \vee D) \\ &= ACDB \vee ACD \vee BCD \vee BCD \\ &= ACD \vee BCD \end{aligned}$$

## Aufgabe 2

1.  $y =$

$$\begin{aligned} y &= \bar{d}\bar{a}(1) \vee \bar{d}a(\bar{c}) \vee d\bar{a}(0) \vee da(\bar{c} \vee b) \\ &= \bar{d}\bar{a} \vee \bar{d}a\bar{c} \vee da\bar{c} \vee dab \end{aligned}$$

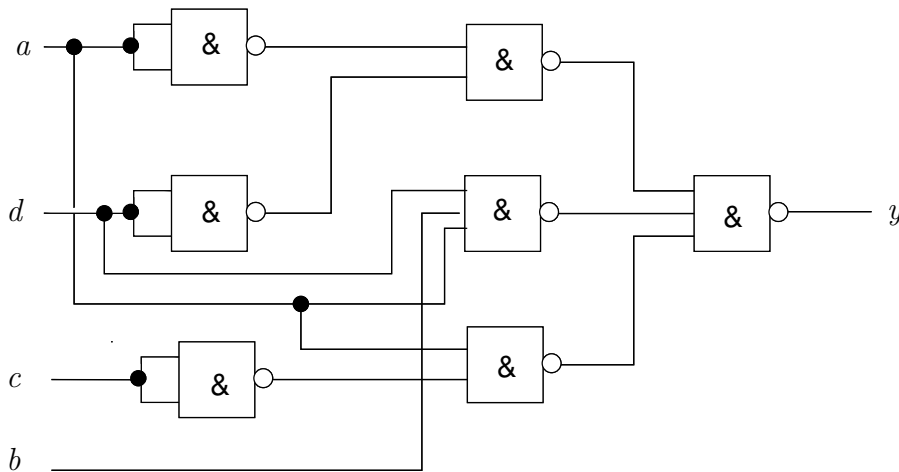
2. Minimalform von  $y$ :

$$\begin{aligned} y &= \bar{d}\bar{a} \vee \bar{d}a\bar{c} \vee da\bar{c} \vee dab \\ &= \bar{d}\bar{a} \vee dba \vee \bar{c}a(d \vee \bar{d}) \\ &= \bar{d}\bar{a} \vee dba \vee \bar{c}a \end{aligned}$$

3. Minimalform von  $y$  in NAND-Form:

$$\begin{aligned} y &= \overline{\overline{\bar{d}\bar{a} \vee dba \vee \bar{c}a}} \\ &= \overline{\overline{\bar{d}\bar{a}} \wedge \overline{dba} \wedge \overline{\bar{c}a}} \\ &= \text{NAND}_3 \left( \text{NAND}_2(\bar{d}, \bar{a}), \text{NAND}_3(d, b, a), \text{NAND}_2(\bar{c}, a) \right) \\ (\bar{x} &= x \wedge \bar{x}) \end{aligned}$$

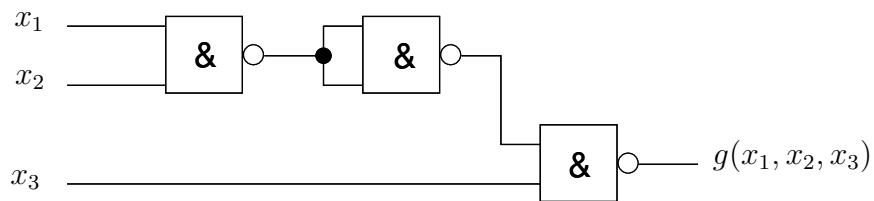
Schaltnetz:



4. Realisierung von  $g(x_1, x_2, x_3)$  mit NAND-Gattern:

$$g(x_1, x_2, x_3) = \overline{x_1 \wedge x_2 \wedge x_3} = \overline{(x_1 \wedge x_2)} \wedge \overline{x_3}$$

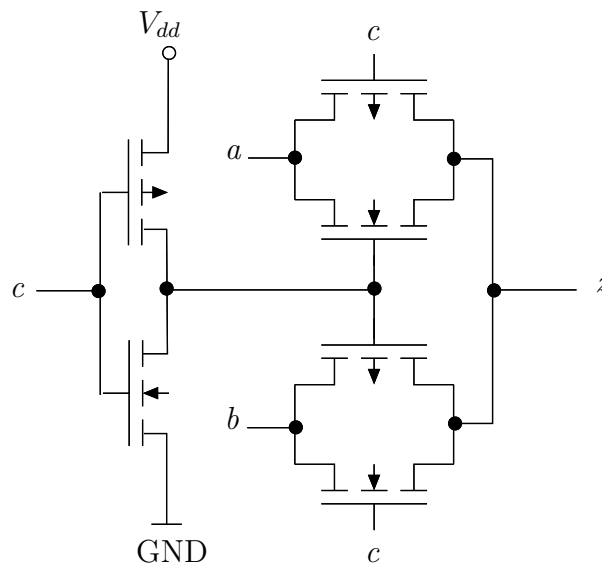
Schaltbild:



5. CMOS-Realisierung eines 2:1-Multiplexers:

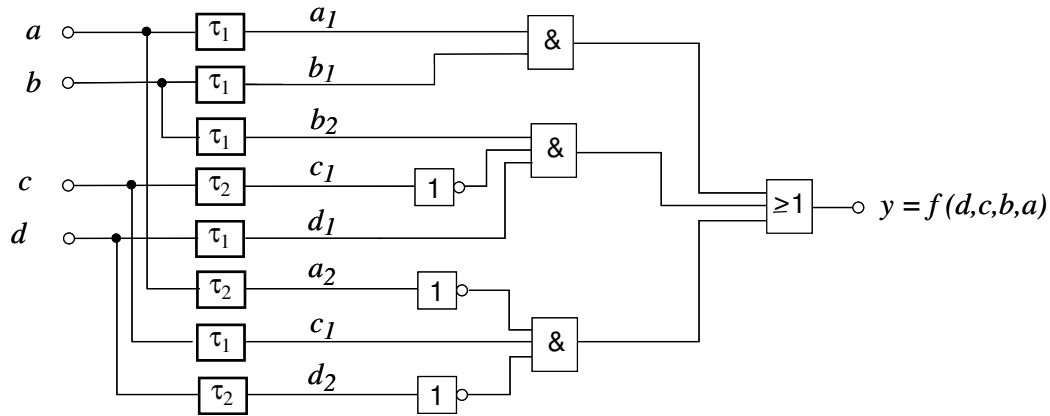
$$\text{Schaltfunktion: } z(c, b, a) = \bar{c} a \vee c b$$

CMOS-Schaltbild:



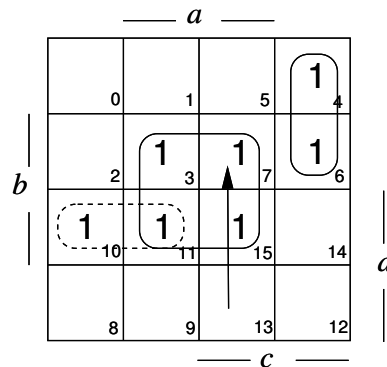
### Aufgabe 3

1. Totzeitmodell:



Dabei ist  $\tau_1 = \tau_{AND} + \tau_{OR}$  und  $\tau_2 = \tau_{AND} + \tau_{OR} + \tau_{NOT}$

2. KV-Diagramm:



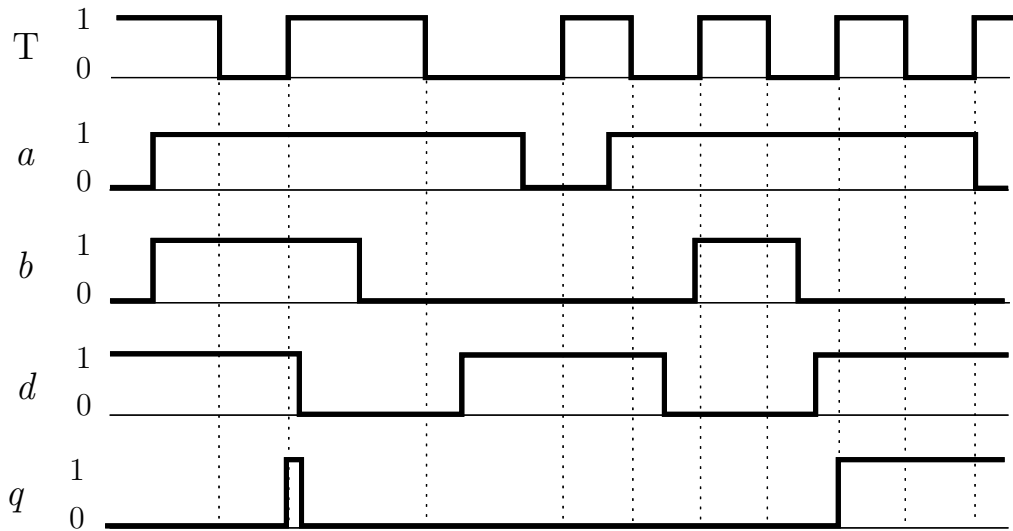
3. Ist das Schaltnetz frei von allen statischen Strukturhasards? Nein

Begründung: Die Realisierung enthält nicht alle Primimplikanten der Schaltfunktion (Satz von Eichelberger)

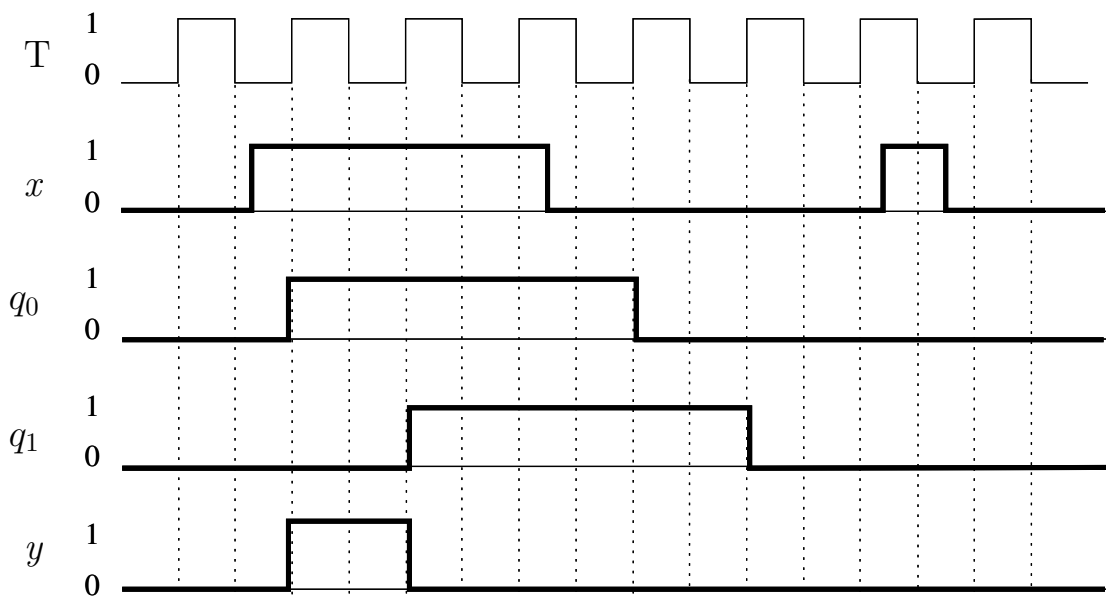
4. Maßnahme zur Behebung des Hasardfehlers beim Übergang  $(1, 1, 0, 1) \rightarrow (0, 1, 1, 1)$ : Übergangsbereich ist  $ca$ . Block  $ba$  ragt in den Übergangsbereich hinein und überdeckt die Einsbelegung des Übergangs. Der Hasardfehler kann dann nur durch günstige Auswahl der Verzögerungszeiten behoben werden.

### Aufgabe 4

1.



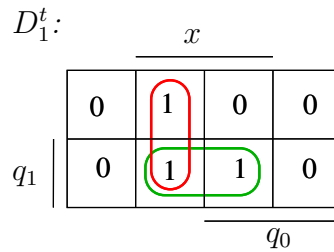
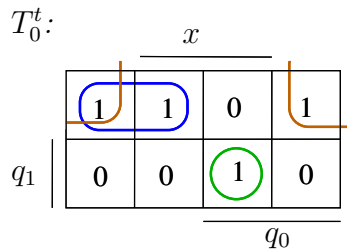
2.



3. (a) Disjunktive Minimalformen der Ansteuerfunktionen:

$$D_1^t = q_1^{t+1} = q_1^t x^t \vee \bar{q}_0^t x^t$$

| $q_0^t$ | $q_1^t$ | $x^t$ | $q_0^{t+1}$ | $q_1^{t+1}$ | $T_0^t$ | $D_1^t$ |
|---------|---------|-------|-------------|-------------|---------|---------|
| 0       | 0       | 0     | 1           | 0           | 1       | 0       |
| 0       | 0       | 1     | 1           | 1           | 1       | 1       |
| 0       | 1       | 0     | 0           | 0           | 0       | 0       |
| 0       | 1       | 1     | 0           | 1           | 0       | 1       |
| 1       | 0       | 0     | 0           | 0           | 1       | 0       |
| 1       | 0       | 1     | 1           | 0           | 0       | 0       |
| 1       | 1       | 0     | 1           | 0           | 0       | 0       |
| 1       | 1       | 1     | 0           | 1           | 1       | 1       |

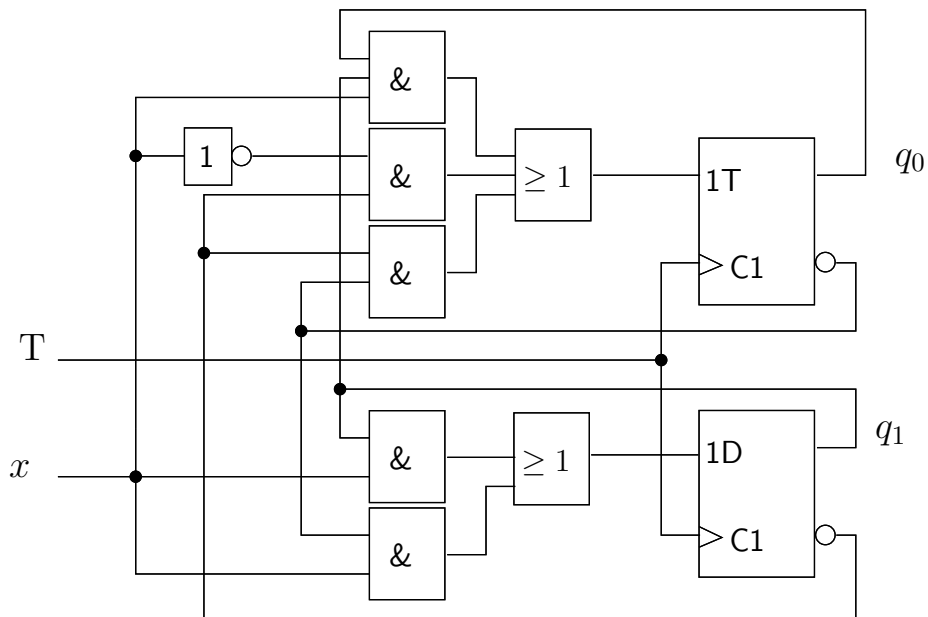


Man erhält:

$$T_0^t = \bar{q}_0^t \bar{q}_1^t \vee \bar{q}_1^t x^t \vee q_0^t q_1^t x^t$$

$$D_1^t = q_1^t x^t \vee \bar{q}_0^t x^t$$

(b) Schaltwerk:



## Aufgabe 5

1.  $512_{10}$  als 32-Bit-Zweierkomplementzahl:

$$512_{10} = 0000\ 0000\ 0000\ 0000\ 0000\ 0010\ 0000\ 0000$$

2.  $-1023_{10}$  als 32-Bit-Zweierkomplementzahl:

$$1023_{10} = 0000\ 0000\ 0000\ 0000\ 0000\ 0011\ 1111\ 1111$$

$$-1023_{10} = 1111\ 1111\ 1111\ 1111\ 1111\ 1100\ 0000\ 0001$$

3. Dezimalzahl:

$$VZ = 1$$

$$Char = 1000\ 0001_2 = 129_{10} \Rightarrow Exp = (129 - 127)_{10} = 2_{10}$$

$$Mantisse = 010000000000000000000000$$

$$\begin{aligned} \text{Dezimalzahl} &= (-1)^1 \cdot (1, \text{Mantisse})_2 \cdot 2^{Exp} \\ &= -1,01 \cdot 2^2 = -(1 + 0,25) \cdot 2^2 = -5_{10} \end{aligned}$$

4.  $1001\ 0101 \times 1100\ 1001 =$

M(0:7)

|           |
|-----------|
| 1100 1001 |
|-----------|

A(0:7)

|           |
|-----------|
| 0000 0000 |
|-----------|

Q(0:7)

|           |
|-----------|
| 1001 0101 |
|-----------|

|           |           |                          |
|-----------|-----------|--------------------------|
| 0100 1001 | 1001 0101 | Addiere M(1:7) zu A(1:7) |
|-----------|-----------|--------------------------|

|           |           |                |
|-----------|-----------|----------------|
| 0010 0100 | 1100 1010 | Rechtsschieben |
|-----------|-----------|----------------|

---

|           |           |                |
|-----------|-----------|----------------|
| 0001 0010 | 0110 0101 | Rechtsschieben |
|-----------|-----------|----------------|

|           |           |                          |
|-----------|-----------|--------------------------|
| 0101 1011 | 0110 0101 | Addiere M(1:7) zu A(1:7) |
|-----------|-----------|--------------------------|

|           |           |                |
|-----------|-----------|----------------|
| 0010 1101 | 1011 0010 | Rechtsschieben |
|-----------|-----------|----------------|

---

|           |           |                |
|-----------|-----------|----------------|
| 0001 0110 | 1101 1001 | Rechtsschieben |
|-----------|-----------|----------------|

|           |           |                          |
|-----------|-----------|--------------------------|
| 0101 1111 | 1101 1001 | Addiere M(1:7) zu A(1:7) |
|-----------|-----------|--------------------------|

|           |           |                |
|-----------|-----------|----------------|
| 0010 1111 | 1110 1100 | Rechtsschieben |
|-----------|-----------|----------------|

---

|           |           |                |
|-----------|-----------|----------------|
| 0001 0111 | 1111 0110 | Rechtsschieben |
|-----------|-----------|----------------|

|           |           |                |
|-----------|-----------|----------------|
| 0000 1011 | 1111 1011 | Rechtsschieben |
|-----------|-----------|----------------|

---

|           |           |                            |
|-----------|-----------|----------------------------|
| 0000 1011 | 1111 1010 | Korrektur: Q(7) = 0 setzen |
|-----------|-----------|----------------------------|

## Aufgabe 6

### 1. von-Neumann-Flaschenhals:

Daten und Maschinenbefehle werden über die gleiche Verbindungseinrichtung zwischen dem Prozessor und dem Hauptspeicher transportiert. Befehle und Operanden können nur nacheinander aus dem Speicher geladen werden.

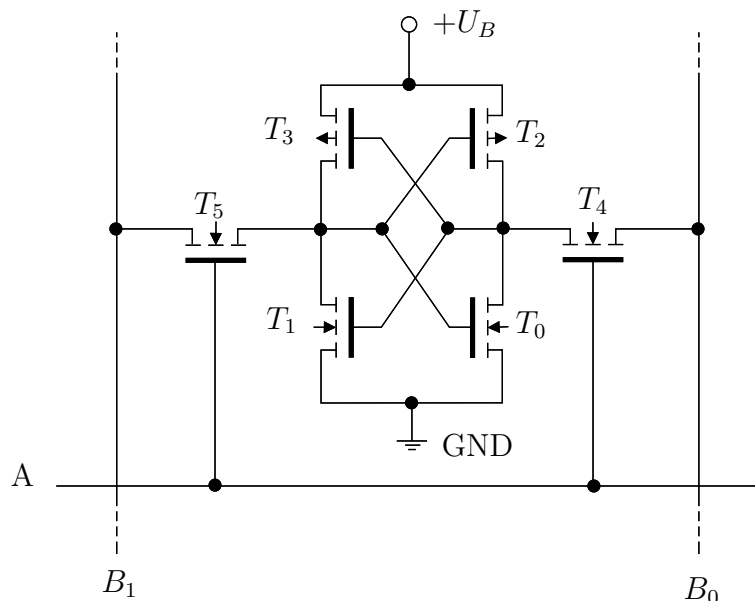
### 2. Bits im Statusregister:

- Übertragsbit (*Carry Flag CF*): Übertrag bei Addition oder Subtraktion (Borrow).
- Hilfsübertragsbit (*Auxiliary Carry AF*): Übertrag von Bit 3 in Bit 4 des Ergebnisses bei BCD-Arithmetik.
- Nullbit (*Zero Flag ZF*): Zeigt an, ob das Ergebnis der letzten Operation gleich 0 war (bedingte Programmverzweigungen, Zählschleifen).
- Vorzeichenbit (*Sign Flag SF*): Zeigt an, dass das Ergebnis negativ ist.
- Überlaufbit (*Overflow Flag OF*): Bereichsüberschreitung im Zweierkomplement.
- Even Flag (*EF*): zeigt an, ob das Ergebnis eine gerade Zahl ist.
- Paritätsbit (*Parity Flag PF*): Signalisiert ungerade Parität des Ergebnisses

### 3. Aufgaben von Schnittstellenbausteinen:

- Pufferung von Ein-/Ausgabe-Daten zur Anpassung unterschiedlicher Arbeitsgeschwindigkeiten im System
- Umsetzung von Daten: parallel/seriell, digital/analog
- Erzeugung von Steuersignalen für Peripheriegeräte, z. B. zur Synchronisation
- Annahme und Erzeugung von Unterbrechungsanforderungen für Peripheriegeräte

### 4. Statische CMOS-Speicherzelle:



## Aufgabe 7

### 1. C-Kontrollstrukturen in MIPS-Assembler:

(a)

```

addi $t0, $0, $zero, 0 # a = 0
addi $t1, $0, $zero, 0 # b = 0
addi $t2, $0, $zero, 0 # c = 0

bge $s0, 5, marke      # if (i >=5) goto marke
addi $t0, $zero, 1     # a = 1
addi $t1, $zero, 2     # b = 2
addi $t2, $zero, 3     # c = 3
marke: addi $t3, $zero, 5 # d = 5

```

(b)

```

addi $t0, $0, $zero, 0 # a = 0
addi $t1, $0, $zero, 0 # b = 0
addi $t2, $0, $zero, 0 # c = 0

bge $s0, 5, marke1    # if (i >=5) goto marke1
addi $t0, $zero, 1     # a = 1
addi $t1, $zero, 2     # b = 2
addi $t2, $zero, 3     # c = 3
j      marke2          #ogoto marke2

marke1: addi $t0, $zero, 4 # a = 4
        addi $t1, $zero, 5 # b = 5
        addi $t2, $zero, 6 # c = 6

marke2: addi $t3, $zero, 5 # d = 5

```

(c)

```

la    $t0, a           # $t0 = &a[0]
addi $a0, $zero, 0     # sum=0
addi $a1, $zero, 0     # i=0
addi $a3, $zero, 100   # $a3 = 100
loop: mult $a2, $a1, 4  # $a2=i*4
      addu $t1,$t0,$a2  # $t1=&a[i]
      lw  $t2,0($t1)    # $t2=a[i]
      add $a0,$a0,$t2   # sum=sum+a[i]
      addi $a1,$a1,1    # i++
      blt $a1,$a3,loop  # if i<100 goto loop

```

### 2. Funktion der MIPS-Befehle:

- (a) `addu $t3, $t2, $t1`: Addition:  $\$t3 = \$t2 + \$t1$  Bereichsüberschreitung wird nicht berücksichtigt.
- (b) `andi $t3, $t2, 0x2000`: Logisches UND:  $\$t3 = \$t2 \text{ UND } 0x2000$

(c) `slt $t3, $t2, $t1`

`if ($t2 < $t1)`

`$t3 = 1`

`else`

`$t3 = 0`

(d) `lui $t3, 0x2000`: Lade das niedrigstwertige Halbwort vom Imm (hier 0x2000) in das höchstwertige Halbwort des Registers `$t3`

3. Register für die Rücksprungadresse: `$ra`

## Aufgabe 8

1. Aufgaben der einzelnen Pipeline-Stufen der DLX-Pipeline für bedingte Sprünge mit PC-relativer Adressierung:

In der Instruction Fetch-Stufe wird der Befehl aus dem Speicher geladen. Die Instruction-Decode-Stufe erzeugt die prozessorinternen Steuersignale und leitet das Displacement und den PC sowie den Vergleichsregisterinhalt weiter. In der Execute-Stufe wird einerseits entschieden, ob der Sprung genommen wird und andererseits aus dem PC und dem Displacement der neue PC berechnet, der dann, falls der Sprung genommen wird, in der MEM-Stufe den PC ersetzt. Im nächsten Takt kann in der IF-Stufe der Befehl von der Sprungzieladresse geladen werden. Die Write-Back-Stufe ist bei bedingten Sprüngen nicht weiter von Bedeutung.

2. (a) Datenabhängigkeiten:

- Echte Abhängigkeiten (*True Dependence*)

$S_1 \rightarrow S_3$  (R1)     $S_1 \rightarrow S_9$  (R1)

$S_2 \rightarrow S_3$  (R2)     $S_2 \rightarrow S_4$  (R2)     $S_2 \rightarrow S_6$  (R2)

$S_3 \rightarrow S_6$  (R3)

$S_4 \rightarrow S_9$  (R1)

$S_5 \rightarrow S_7$  (R4)

$S_6 \rightarrow S_8$  (R5)

- Gegenabhängigkeiten (*Anti-Dependence*):  $S_3 \rightarrow S_4$  (R1)

- Ausgabe-Abhängigkeiten (*Output Dependence*):  $S_1 \rightarrow S_4$  (R1)

(b) Behebung der Konflikte:

S1:    `lw`    R1, 1000(R0)

S2:    `lw`    R2, 1004(R0)

`NOP`

`NOP`

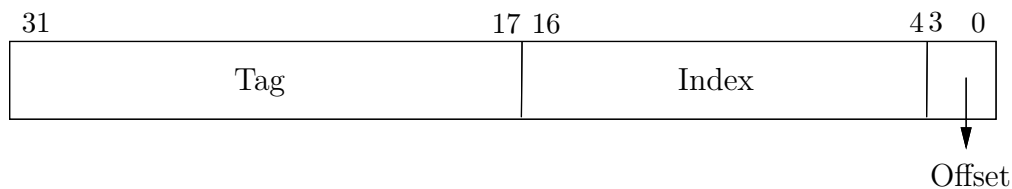
S3:    `add`    R3, R2, R1

```

S4:   addi  R1, R2, 8
S5:   subi  R4, R0, 2
S6:   and   R5, R3, R2
      NOP
S7:   sw    R4, 1000(R0)
S8:   sw    R5, 1004(R0)
S9:   sw    R1, 1008(R0)
    
```

## Aufgabe 9

1. (a) Unterteilung der Hauptspeicheradresse:



(b) Anzahl der Zeilen:

$$\frac{128\text{KByte}}{16\text{Byte}} = 8\text{K Zeilen} = 8192 \text{ Zeilen}$$

(c) Hauptspeicheradresse 2004:

$$2004_{10} = \underbrace{111\ 1101}_{\text{Zeilenindex}}\ 0100_2$$

Hauptspeicheradresse wird in die Zeile mit der Nummer  $111\ 1101_2 = 125_{10}$  geladen.

2. (a) Tag-Breite: 16 Bit

Der Cachespeicher hat  $8\text{K}$  Zeilen  $\rightarrow 4\text{K} = 2^{12}$  Sätze  $\rightarrow$  Satzindex ist 12 Bit breit.  
 Der Byte-Offset ist 4 Bit breit  $\rightarrow$  Tag ist  $32 - 12 - 4 = 16$  Bit breit.

(b) Erforderlicher Speicherbedarf:

Für jede Zeile sind (Tag + 2 Statusbit + Daten pro Zeile) Bits erforderlich, d. h.  $(16 + 2)$  Bits + 16 Byte

Speicherbedarf für den gesamten Cache:

$$(16 \text{ Bits} + 2 \text{ Bits} + 16 \text{ Byte}) \cdot 8\text{K} = 16 \text{ KByte} + 2 \text{ KByte} + 128 \text{ KByte} = 146 \text{ KByte}$$

3. Mittlere Zugriffszeit auf den Speicher für das Programm `test`:

- Befehls-cache:

$$0,8 \cdot 10 \text{ ns} + 0,2 \cdot 80 \text{ ns} = 24 \text{ ns}$$

- Datencache:

$$0,5 \cdot 20 \text{ ns} + 0,5 \cdot 80 \text{ ns} = 50 \text{ ns}$$

- Mittlere Zugriffszeit für das Programm:

$$0,8 \cdot 24 \text{ ns} + 0,2 \cdot 50 \text{ ns} = 29,2 \text{ ns}$$

## Aufgabe 10

1. Unterteilung der virtuellen Adresse:



2. Physikalische Adressen:

| Virtuelle Adresse | Physikalische Adresse |
|-------------------|-----------------------|
| 2100              | 1076 (0x434)          |
| 4095              | 4095 (0xFFF)          |
| 5620              | 500 (0x1F4)           |
| 6200              | 2104 (0x838)          |
| 1023              | <i>page fault</i>     |

3. Eine Beschleunigung der Adressumsetzung durch den *TLB* wird beim zweiten Zugriff auf eine Seite und solange die entsprechenden Einträge aus dem Seitentabellen-Verzeichnis und der Seitentabelle aus dem *TLB* nicht verdrängt wurden.