Tutorat 4

Nachkommastellen beim den Darstellungen von Festkommazahlen, RETI, CMOS, p-Kanal und n-Kanal Transistoren

Gruppe 9

Präsentator:
Jürgen Mattheis
(juergmatth@gmail.com)

Vorlesung von:
Prof. Dr. Scholl

Übungsgruppenbetreuung:
Tobias Seufert

11. Juni 2023

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

Gliederung

Organisatorisches

Organisatorisches

Abgaben

- schreibt bitte euren Namen und Matrikelnummer auf die Abgaben
- Vorrechnen nicht vergessen:
  1. entweder generell immer mal wieder Melden, dann zählt das irgendwann als Vorrechnen
  2. oder vor dem Tutorat ansprechen, dann werdet ihr während des Tutorats dazu gefragt, ob ihr zu einer Aufgabe vielleicht irgendetwas gehaltvolles sagen könnt
- um die Leute, die in die Tutorate kommen zu belohnen und dem Ereignis entgegenzuwirken, dass das Tutorat irgendwann leer ist, werden die Folien während des Tutorats auf einem USB-Stick verteilt
  - die Folien aller bisherigen Tutorate werden immer auch alle auf dem USB-Stick sein

Aufgabe 1

Aufgabe 1 I

Nachkommastellen bei den Darstellungen von Festkommazahlen

Voraussetzungen 1.1

```python
#!/usr/bin/env python

LENGTH = 4
NUM_DEICMAL_BITS = 2
NUM_BITS = 4


def value_1s_complement(bits):
    acc = 0
    i = -1
    for i, ai in enumerate(map(lambda bit: int(bit), reversed(bits[1:]))):
        acc += ai * 2 ** (i - NUM_DEICMAL_BITS)
    return acc - int(bits[0]) * (2 ** (i + 1 - NUM_DEICMAL_BITS) - 2**-2)


def value_2s_complement(bits):
    acc = 0
    i = -1
    for i, ai in enumerate(map(lambda bit: int(bit), reversed(bits[1:]))):
        acc += ai * 2 ** (i - NUM_DEICMAL_BITS)
    return acc - int(bits[0]) * 2 ** (i + 1 - NUM_DEICMAL_BITS)


if __name__ == "__main__":
    print("{", end="")
    i = 0
    while i < 2**NUM_BITS:
        bits = str(bin(i))[2:]
        bits = "0" * (LENGTH - len(bits)) + bits
        print(bits, end="|->(")
        print(value_1s_complement(bits), end=", ")
        print(value_2s_complement(bits), end=")")
        i += 1
        if i != 2**NUM_BITS:
            print(", ", end="")
    print("}", end="")
```

Aufgabe 1 II

Nachkommastellen bei den Darstellungen von Festkommazahlen

Voraussetzungen 1.1

```
> ./value_of_representation.py
{0000->(0.0, 0.0), 0001->(0.25, 0.25), 0010->(0.5, 0.5), 0011->(0.75, 0.75), 0100->(1.0, 1.0), 0101->(1.25, 1.25),
0110->(1.5, 1.5), 0111->(1.75, 1.75),  1000->(-1.75, -2.0), 1001->(-1.5, -1.75), 1010->(-1.25, -1.5),
1011->(-1.0, -1.25), 1100->(-0.75, -1.0), 1101->(-0.5, -0.75), 1110->(-0.25, -0.5), 1111->(0.0, -0.25)}
```

Aufgabe 1 III

Nachkommastellen bei den Darstellungen von Festkommazahlen

Lösung 1.1

- $[a]_1 = \sum_{i=-k}^{n-1} a_i \cdot 2^i - a_n \cdot (2^n - 2^{-k})$
- *Beim Einerkomplement wird, da es zwei 0en gibt wegen der Symmetrie die größtmögliche positive Zahl abgezogen. Diese ist:*

$$\begin{aligned}\sum_{i=-k}^{n-1} 1 \cdot 2^i &= \sum_{i=0}^{n-1} 1 \cdot 2^i + \sum_{i=-k}^{-1} 1 \cdot 2^i \\ &= \sum_{i=0}^{n-1} 2^i + \sum_{i=0}^{k-1} 2^{i-k} \\ &= \sum_{i=0}^{n-1} 2^i + \sum_{i=0}^{k-1} \frac{2^i}{2^k}\end{aligned}$$

Aufgabe 1 IV

Nachkommastellen bei den Darstellungen von Festkommazahlen

Lösung 1.1

$$
\begin{aligned}
&= \sum_{i=0}^{n-1} 2^i + \frac{\sum_{i=0}^{k-1} 2^i}{2^k} \\
&= 2^n - 1 + \frac{2^k - 1}{2^k} \\
&= 2^n - 1 + 1 - \frac{1}{2^k} \\
&= 2^n - 2^{-k}
\end{aligned}
$$

Aufgabe 1 I

Nachkommastellen bei den Darstellungen von Festkommazahlen

Lösung 1.1

- $[a]_2 = \sum_{i=-k}^{n-1} a_i \cdot 2^i - a_n \cdot 2^n$
- *Beim Zweierkomplement wird, da es nur eine* 0 *gibt wegen dem dadurch resultierenden asymmetrischen Zahlenbereich* (-2^{n-1} *ist die größte negative Zahl und* $2^{n-1} - 1$ *ist die größte positive Zahl) die größtmögliche Zahl größer* 0 $+$ *die kleinstmögliche Zahl größer* 0 *(da die zusätzliche* 0 *nicht mehr eine Kodierung braucht und übersprungen wird, sodass man direkt zur kleinstmöglichen Zahl größer* 0 *geht, also* 2^{-k}*) abgezogen:*

$$\sum_{i=-k}^{n-1} 1 \cdot 2^i + 2^{-k} = \sum_{i=0}^{n-1} 1 \cdot 2^i + \sum_{i=-k}^{-1} 1 \cdot 2^i + 2^{-k} = 2^n - 2^{-k} + 2^{-k} = 2^n$$

Aufgabe 1b

Lösung 1.2

1. $-1,0_{10} = -32 + 16 + 8 + 4 + 2 + 1 = -0.25 - 0.5 - 0.25 = [111111.00]_2$
2. $2,25_{10} = 2 + 0.25 = [000010.01]_2$
3. $-2.4_{16} = -(2 \cdot 16^0 + 4 \cdot 16^{-1})$

   $$= -([0010]_2 \cdot (2^4)^0 + [0100]_2 \cdot (2^4)^{-1}) = -([00100100]_2 \cdot 2^{-4}) = -([0010.0100]_2 \cdot 2^2)$$

   $$= -[000010.01]_2 = [111101.10]_2 + 2^{-2} = [111101.10]_2 + [000000.01]_2 = [111101.11]_2$$

   - $-2.4_{16} = -(2 \cdot 16^0 + \frac{4}{16}) = -2.25_{10} = -32+16+8+4+1+0.5+0.25_{10} = -0.25-2 = [111101.11]_2$
4. *größte darst. Zahl:* $31.75 = 16 + 8 + 4 + 2 + 1 + 0,5 + 0,25 = [011111.11]_2$
5. *größte darst. Zahl* < 1*:* $0.75 = 0.5 + 0.25 = [000000.11]_2$
6. *größte darst. Zahl* < 0*:* $-0.25 = -32 + 16 + 8 + 4 + 2 + 1 + 0.5 + 0.25 = -0.25 = [111111.11]_2$

Anmerkungen

- 8 Bits $d_5 d_4 d_3 d_2 d_1 d_0 d_{-1} d_{-2}$, wobei d_5 sign bit ist, daher $2^5 - 2^{-2} + 2^{-2} = 2^5 = 32$ (größte positive Zahl mit 5 Vorkommastellenbits + kleinster Zahlenabstand mit 2 Nachkommastellenbits)

Aufgabe 2

Aufgabe 2 I

RETI

Voraussetzungen 2.1

- *Seien* $a, b \in \mathbb{N}$. *Wenn wir wiederholt den Euklidischer Algorithmus durchführen:*

$$
\begin{aligned}
a &= b \cdot q_1 + r_1 \\
b &= r_1 \cdot q_2 + r_2 \\
r_1 &= r_2 \cdot q_3 + r_3 \\
&\dots \\
r_{n-3} &= r_{n-2} \cdot q_{n-1} + r_{n-1} \\
r_{n-2} &= r_{n-1} \cdot q_n + 0
\end{aligned}
$$

  dann gilt: $r_{n-1} = ggt(a, b)$.
- $ggt(a, b) = ggt(max(a, b) - min(a, b), min(a, b))$ und $ggt(a, 0) = a$

Aufgabe 2 II

RETI

Voraussetzungen 2.1

- *Beweis:* ***https://www.youtube.com/watch?v=8cikffEcyPI***
  - *Anmerkung zum Beweis im Video: Wann immer wir eine geordnete Folge von Zahlen haben, deren Wert immer kleiner wird und nichtnegativ sind, können wir sicher sein, dass diese Folge den Wert 0 erreicht.*
  - *Man braucht folgende Definition für GGT: Eine Zahl g wird „größter gemeinsamer Teiler“ von a und b genannt, wenn sie die folgenden beiden Bedingungen erfüllt:*
    1. *g is a „gemeinsamer Teiler“ von a und b. In anderen Worten: g teilt a und g teilt b*
    2. *Wenn d ein gemeinsamer Teiler von a und b ist dann: d teilt g*
- *Wichtig für RSA, da Primfaktorzerlegung mit der man den GGT zweier Zahlen berechnen kann sehr langsam ist, aber der Euklidische Algorithmus den GGT zweier Zahlen sehr schnell berechnen kann ohne die beiden Zahlen vollständig faktorisieren zu müssen.*

Aufgabe 2 III

RETI

Voraussetzungen 2.1

- *Was eigentlich gerechnet wird: Division mit Rest*

$$a = b \cdot q_1 + r_1 \quad \xRightarrow{\text{eigentlich}} \quad \frac{a}{b} = q \text{ Rest } r$$

- *Algorithmus vereinfacht ausgedrückt:*
  1. *Divisor nach vorne*
  2. *Rest zum Divisor*
  3. *den Qutoienten ignorieren*
- *Euklidischer Algorithmus am Beispiel* $a = 12$ *und* $b = 16$*:*

$$12 = 16 \cdot 0 + 12$$

$$16 = 12 \cdot 1 + \boxed{4}$$

$$12 = \boxed{4} \cdot 3 + 0$$

dann gilt: $4 = ggt(12, 16)$.

Aufgabe 2 IV

RETI

Lösung 2.1

```python
def gcd(front: int, divisor: int) -> int:
    while divisor:
        remainder = front % divisor  # calculate remainder
        front = divisor              # divisor to front
        divisor = remainder          # remainder to divisor
    return front                     # as soon as the divisor becomes 0, the divisor goes to the front


if __name__ == "__main__":
    print(gcd(12, 16))
```

- *letzter Durchlauf:* $\boxed{4} = 0 \cdot \textit{whatever} + 4$

Aufgabe 2 V

RETI

Lösung 2.1

```
int ggt(int a, int b) {
  while(a != b) {
    if(a < b)
      b = b-a;
    else // (a > b)
      a = a-b;
  }
  return a;
}

void main() {
  print(ggt(16, 12));
}
```

Aufgabe 2 VI

RETI

Lösung 2.1

- *Modulo ist q Mal die eine Zahl von der anderen abziehen:* $a \bmod b = a - q \cdot b$. *Für den GGT muss das Modulo nehmen wie aus der Funktion gcd bekannt fortgeführt werden. Bei genauer Betrachtung lässt sich das gesamte Vorgehen auf folgende Fälle beschränken:*

$$\begin{cases} b = b - a & \text{if } a < b\,, \\ a = a - b & \text{if } a > b\,, \\ a \text{ oder } b & \text{if } a = b\,. \end{cases}$$

  bis: $a = b$
- *damit am Ende der Rest* 0 *sein kann, muss es am Ende der Fall sein, dass* $a = b$, *da nur so* 0 *als Rest rauskommen kann, wenn man dann* $a - b$ *bzw.* $b - a$ *rechnet*

Aufgabe 2 VII

RETI

Lösung 2.1

- *Beispiel:*

$$
\begin{aligned}
20 &= 32 - 12 \\
8 &= 20 - 12 \\
4 &= 12 - 8 \\
\boxed{4} &= 8 - \boxed{4}
\end{aligned}
$$

Aufgabe 2 VIII

RETI

Lösung 2.1

```
# INITIALISATION
LOADI ACC 24
STORE ACC 40
LOADI ACC 16
STORE ACC 41
# PROGRAM
LOAD ACC 40
SUB ACC 41
JUMP== 10
JUMP> 5
LOAD ACC 41
SUB ACC 40
STORE ACC 41
JUMP 4
LOAD ACC 40
SUB ACC 41
STORE ACC 40
JUMP -11
LOAD ACC 40
STORE ACC 42
```

Aufgabe 2 IX

RETI

Lösung 2.1

```
index:        22
instruction:  SUB ACC 41;
ACC:          8
ACC_SIMPLE:   8
IN1:          0
IN1_SIMPLE:   0
IN2:          0
IN2_SIMPLE:   0
PC:           2147483667
PC_SIMPLE:    19
SP:           2147483693
SP_SIMPLE:    45
BAF:          2147483650
BAF_SIMPLE:   2
CS:           2147483651
CS_SIMPLE:    3
DS:           2147483671
DS_SIMPLE:    23
SRAM:
  00000 JUMP 0;
  00001 2147483648
  00002 0 <- BAF
  00003 LOADI ACC 24; <- CS
  00004 STORE ACC 40;
  00005 LOADI ACC 16;
  00006 STORE ACC 41;
  00007 LOAD ACC 40;
  00008 SUB ACC 41;
  00009 JUMP== 12;
  00010 LOAD ACC 40;
  00011 SUB ACC 41;
  00012 JUMP> 5;
  00013 LOAD ACC 41;
  00014 SUB ACC 40;
  00015 STORE ACC 41;
  00016 JUMP 4;
  00017 LOAD ACC 40;
  00018 SUB ACC 41;
  00019 STORE ACC 40; <- PC
  00020 JUMP -13;
:noh
```

```
|  00021 LOAD ACC 40;
|  00022 STORE ACC 42;
|  00023 0 <- DS
|  00024 0
|  00025 0
|  00026 0
|  00027 0
|  00028 0
|  00029 0
|  00030 0
|  00031 0
|  00032 0
|  00033 0
|  00034 0
|  00035 0
|  00036 0
|  00037 0
|  00038 0
|  00039 0
|  00040 24
|  00041 16
|  00042 0
|  00043 0
|  00044 0
|  00045 0 <- SP
|UART:
|  00000 0
|  00001 0
|  00002 0
|EPROM:
|  00000 LOADI DS -2097152; <- IN1 <- IN2
|  00001 MULTI DS 1024;
|  00002 MOVE DS SP;
|  00003 MOVE DS BAF;
|  00004 MOVE DS CS;
|  00005 ADDI SP 45;
|  00006 ADDI BAF 2;
|  00007 ADDI CS 3;
|  00008 ADDI DS 23; <- ACC
|  00009 MOVE CS PC;
```

Aufgabe 3

Appendix I

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

a	*b*	*c*	*out*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabelle 1: Wertetabelle for boolesche Funktion $\overline{c} \wedge (\overline{a} \vee \overline{b})$

Appendix II

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

- *DNF:* $\overline{a}\overline{b}\overline{c} + \overline{a}b\overline{c} + a\overline{b}\overline{c}$

Karnaugh Map:

a \ bc	$\overline{b}\overline{c}$	$\overline{b}c$	bc	$b\overline{c}$
$\overline{a}$	1	0	0	1
a	1	0	0	0

- *Oder Webseite zur Hilfe nehmen:* http://www.32x8.com/index.html

Appendix III

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

- *Vereinfachter Boolescher Ausdruck (1en):* $\overline{b}\overline{c} + \overline{a}\overline{c} = \overline{c} \cdot (\overline{a} + \overline{b})$

a	b	c	$\overline{b}\overline{c} + \overline{a}\overline{c}$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Appendix IV

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

- *DNF:* $(\overline{a} \cdot \overline{b} \cdot c) + (\overline{a} \cdot b \cdot c) + (a \cdot \overline{b} \cdot c) + (a \cdot b \cdot \overline{c}) + (a \cdot b \cdot c)$

Karnaugh Map:

a \ bc	$\overline{b}\overline{c}$	$\overline{b}c$	bc	$b\overline{c}$
$\overline{a}$	1	0	0	1
a	1	0	0	0

Appendix V

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1

- *Vereinfachter Boolescher Ausdruck (0en):* $c + ab$

a	b	c	$c + ab$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Appendix VI

CMOS, P-Kanal und N-Kanal Transistoren

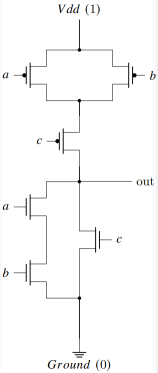
Lösung 3.1

- *auf der einen Seite mit den P-Kanal-Transistoren wird der vereinfachte Ausdruck umgesetzt, der für genau die Modelle wahr ist, für welche die logische Funktion* $\overline{c} \cdot (\overline{a} + \overline{b})$ *wahr ist.*
- *Auf der anderen Seite mit den N-Kanal Transistoren wird der vereinfachte Ausdruck umgesetzt, der für genau die Modelle wahr ist, für welche die logische Funktion* $\overline{c} \cdot (\overline{a} + \overline{b})$ *falsch ist.*
- *damit wird sichergestellt, dass es zu keinem Kurzschluss kommen kann, weil immer nur entweder die eine oder anderen Seite ein Signal* 0 *(Ground) oder* 1 *(Vdd) zu out durchlässt, aber niemals beide, da der eine mit P-Kanal-Transistoren umgesetzte Ausdruck für genau die Modelle wahr ist, für die der andere mit N-Kanal-Transistoren umgesetzte Ausdruck falsch ist.*

Appendix VII

CMOS, P-Kanal und N-Kanal Transistoren

Lösung 3.1



Appendix

Appendix I

Hinweis zu Hypercubes und Karnaugh Map

- Wenn sich zwei Knoten nur durch genau 1 Bit unterscheiden, dann werden sie durch eine Kante verbunden.

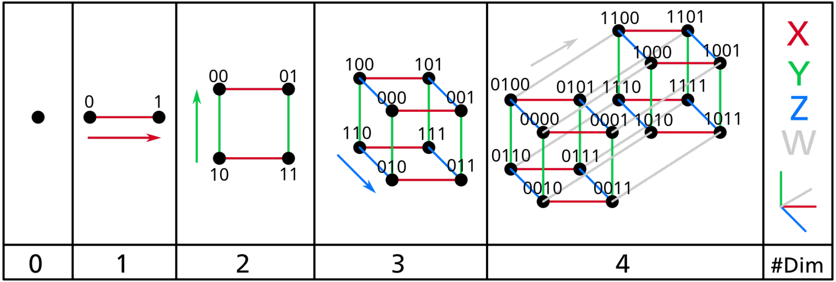
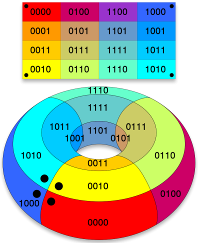


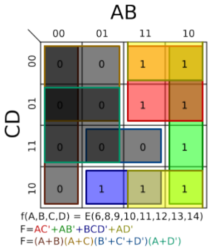
Abbildung 1: *Hypercubes von $n = 0$ bis $n = 4$*

Appendix II

Hinweis zu Hypercubes und Karnaugh Map



(a) Torus oder umgangssprachlich Donut



(b) Beispiel für Karnaugh Map

- Anordnung ist so, weil man einen Hypercube bis $n = 4$ (Tesseract) in einer flachen Tabelle darstellen will und im Hypercube gibt es immer nur eine Kante, wenn es nur genau 1 Bit Unterschied bei 2 Knoten gibt.

- In der Karnaugh Map sind entsprechend immer nur Bitvektoren benachbart, die sich nur in einem Bit unterscheiden.

Appendix III

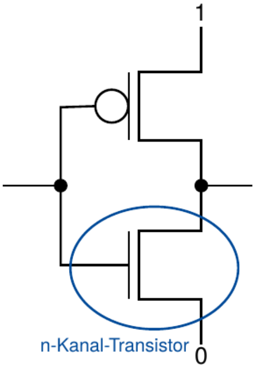
Hinweis zu Hypercubes und Karnaugh Map

- Wie bei einem Donut sind Reihen und Spalten miteinander von unten nach oben und von links nach rechts über die Tabelle hinausgehend benachbart.
- Von $\bar{a}b$ zu ab gibt es nur 1 Änderung. Von $\bar{a}b$ zu $a\bar{b}$ gibt es dagegen 2 Änderungen. Daher sind $\bar{a}b$ und ab benachbart.

Appendix I

Kurzschluss

- $I = \frac{U}{R}$
- $P = U \cdot I$



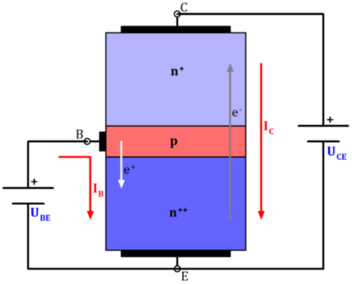
Appendix

Transistoren

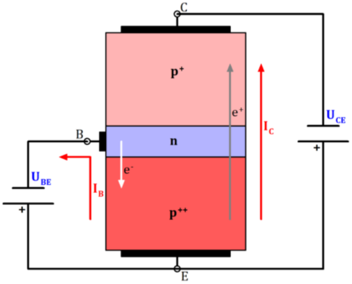
- Ströme steuern. Man kann den Stromfluss innerhalb einer elektrischen Schaltung abbremsen, sodass überhaupt kein Strom fließt (der Transistor funktioniert als Schalter). Man kann aber den Stromfluss auch stark beschleunigen, wodurch ein viel stärkerer Strom fließt (der Transistor funktioniert als Verstärker).
- im Kern ist ein Transistor entweder ein strom- oder spannungsgesteuerter Widerstand. Feldeffekttransistoren sind spannungsgesteuerte, Bipolartransistoren sind stromgesteuerte Widerstände.
- Unterschiede bei Art der Lagungsträger, die zum Stromfluss beiträgt. Bei einem Bipolartransistor sind das Elektronen und Löcher. Daher kommt auch der Teil „Bi“ im Namen. Bei einem Feldeffekttransistor sind das hingegen entweder Elektronen oder Löcher, also nur eine Art an Ladungsträger.
- Feldeffekttransistoren werden überwiegend überall dort verwendet, wo hohe Ströme , Bipolartransistoren hingegen dort, wo geringe Ströme fließen.

Appendix I

Bipolartransistoren



(a) PNP-Transistor



(b) npn-Transistor

- drei Anschlüsse: Collector (C), Base (B) und Emitter (E)

Appendix I

Feldeffekttransistoren

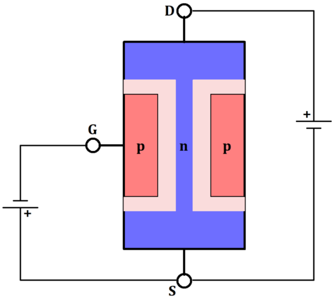


Abbildung 4: n-Kanal-Feldeffekttransistor

- drei Anschlüsse: Drain (D, „Senke, Abfluss“), Gate (G, „Tor“) und Source (S, „Quelle, Zufluss“)

Appendix II

Feldeffekttransistoren

- Sperrschicht-Feldeffekttransistor (SFET, engl. JFET)
- Je nachdem wie der Kanal, durch denen die Ladungsträger fließen, dotiert ist, findest man die Bezeichnungen n-Kanal-FFET und p-Kanal-FFET