Digitaltechnik
Übung 5

Matteo Dietz
mdietz@student.ethz.ch

Polybox

- https://polybox.ethz.ch/index.php/s/VehRU12QqdJv98i

Theorie – Zahlensysteme

Berechnung einer positiven Zahl D mit der Basis („Radix“) R und den Koeffizienten („Ziffern“) b_i:

$$D = \sum_{i=-\infty}^{i=\infty} b_i \cdot R^i$$

Zahlendarstellung in Basis R: $\cdots b_{i=+2} b_{i=+1} b_{i=0} b_{i=-1} \cdots_{(R)}$

- Es gilt immer: $0 <= b_i < R$

Theorie – wichtige Zahlensysteme

- Dezimalsystem $15_{(10)}$
- Octalsystem $17_{(8)}$
- Dualsystem $1111_{(2)}$
- Hexadezimalsystem $F_{(16)}$ oder 0xF

Theorie – Hexadezimalzahlen

- Basis 16
- Lange Dualzahlen können übersichtlicher mit Hexadezimalzahlen dargestellt werden
- $2A_{(16)}$ oder 0x2A
- Päckchenweise Umrechnung mit Binärzahlen (Päckchen von 4 Ziffern in Binärzahlen)

Dezimalzahl	Hexadezimalzahl
0…9	0…9
10	A
11	B
12	C
13	D
14	E
15	F

Theorie – Zahlenumwandlung

- Welche Potenz der Basis passt gerade noch rein?
- Wieviel Mal passt sie rein?

- Umwandlung vor und nach Komma separat
- Nicht jede Zahl kann endlich in einem anderen Zahlensystem dargestellt werden

Theorie – Zahlenumwandlung D > 1

Verfahren für eine ganze Zahl D:

- Wiederholte ganzzahlige Division mit der Zahlenbasis R; der jeweilige Rest r entspricht der Ziffer im neuen Zahlensystem, beginnend mit der letzten Ziffer

$$D/R = Q_0 + r_0 \text{ und dann } Q_i/R = Q_{i+1} + r_{i+1} \text{ mit } i \geq 0$$

- Das Verfahren endet, wenn als Divisionsergebnis nach m Divisionen Null erreicht wird, d.h. der Quotient $Q_{m-1} = 0$
- Diese Konversion ergibt im Zahlensystem R die Zahl

$$Z = r_{m-1} \cdot R^{m-1} + r_{m-2} \cdot R^{m-2} + \cdots + r_0$$

die sich als $Z = r_{m-1} r_{m-2} \cdots r_{0\,(R)}$ darstellen lässt

Theorie – Zahlenumwandlung D > 1

Binär

D	R			b
36	**:2**	**=**	18	0
18	:2	=	9	0
9	:2	=	4	1
4	:2	=	2	0
2	:2	=	1	0
1	:2	=	0	1

LSB

MSB

$36_{10} = 100100_2$

Oktal

D	R			b
36	**:8**	**=**	4	4
4	:8	=	0	4

$36_{10} = 44_8$

Hexadezimal

D	R			b
36	:16	=	2	4
2	:16	=	0	2

$36_{10} = 24_{16}$

Theorie – Zahlenumwandlung D < 1

Verfahren für eine Zahl $1 > D \geq 0$:

- Multiplikation von $a_0 = D$ mit der Zahlenbasis R

$$a_0 \cdot R = P_0 \rightarrow K_{-1} = \text{floor}(P_0),\ a_{-1} = P_0 - K_{-1}$$

(Die Abrundungsfunktion floor(x) ordnet x die nächstliegende nicht grössere ganze Zahl zu, z.B. $\text{floor}(2.8) = 2$ und $\text{floor}(-2.8) = -3$)

- Die Prozedur geht weiter mit $i = -1, -2, -3, \ldots$

$$a_i \cdot R = P_i \rightarrow K_{i-1} = \text{floor}(P_i),\ a_{i-1} = P_i - K_{i-1}$$

- Das Verfahren endet, wenn $a_{-m} = 0$ nach m Schritten
- Diese Konversion ergibt im Zahlensystem R die Zahl

$$Z = K_{-1} \cdot R^{-1} + K_{-2} \cdot R^{-2} + \cdots + K_{-m} \cdot R^{-m}$$

die sich als $Z = 0.K_{-1}K_{-2}\cdots K_{-m_{(R)}}$ darstellen lässt

Theorie – Zahlenumwandlung D < 1

$\boldsymbol{D = 0.171875_{(10)}}$ als Dualzahl, also Basis $R = 2$

Operation	Produkt	Koeffizient	Bemerkung
$D \cdot R = 0.171875 \cdot 2$	$P_0 = 0.34375$	$K_{-1} = \text{floor}(P_0) = 0$ $(a_{-1} = P_0 - K_{-1})$	Most Significant Bit (MSB)
$a_{-1} \cdot R = 0.34375 \cdot 2$	$P_{-1} = 0.6875$	$K_{-2} = \text{floor}(P_{-1}) = 0$ $(a_{-2} = P_{-1} - K_{-2})$	
$a_{-2} \cdot R = 0.6875 \cdot 2$	$P_{-2} = 1.375$	$K_{-3} = \text{floor}(P_{-2}) = 1$ $(a_{-3} = P_{-2} - K_{-3})$	
$a_{-3} \cdot R = 0.375 \cdot 2$	$P_{-3} = 0.75$	$K_{-4} = \text{floor}(P_{-3}) = 0$ $(a_{-4} = P_{-3} - K_{-4})$	
$a_{-4} \cdot R = 0.75 \cdot 2$	$P_{-4} = 1.5$	$K_{-5} = \text{floor}(P_{-4}) = 1$ $(a_{-5} = P_{-4} - K_{-5})$	
$a_{-5} \cdot R = 0.5 \cdot 2$	$P_{-5} = 1.0$	$K_{-6} = \text{floor}(P_{-5}) = 1$ $(a_{-6} = P_{-5} - K_{-6} = \mathbf{0})$	Least Significant Bit (LSB)

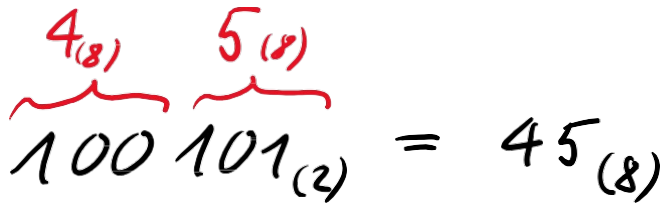
Also $0.171875_{(10)} = 0.001011_{(2)} = 0.K_{-1}K_{-2}K_{-3}K_{-4}K_{-5}K_{-6}$

⚠ Nicht jede rationale Dezimalzahl ist exakt mit **endlicher** Stellenanzahl in einem anderen Zahlensystem darstellbar

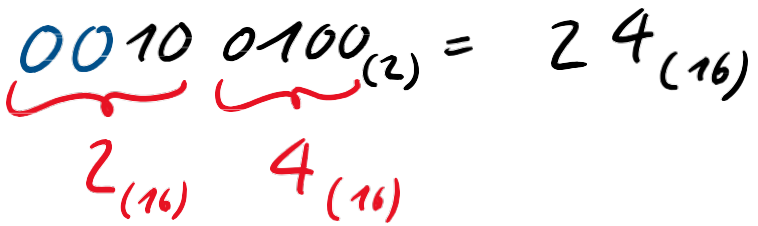
5

Theorie – Zahlenumwandlung Tricks

- Tipps:
  - Binär -> Oktal: Päckchen von 3 Ziffern:



  - Binär -> Hex: Päckchen von 4 Ziffern:



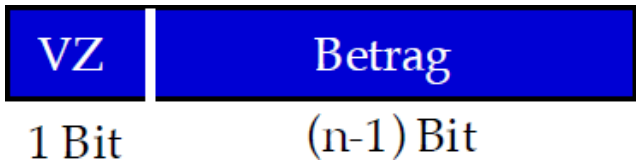
  - Verwendet Zahlentabellen!

Aufgabe – Zahlenumwandlung

- 13.75(10) als Binärzahl
- 0x1A2 als Dezimalzahl
- 1110111101(2) als Hexadezimalzahl

Theorie – Zweierkomplement

- Um auch mit negativen Zahlen im Binärsystem rechnen zu können, wird das Zweierkomplement benutzt.



- Falls VZ = 1, negieren wir die höchsten Potenz und addieren die hinteren Bits
- Für rationale Zahlen: mQn-Darstellung
  m Ziffern vor dem Komma und n Ziffern nach dem Komma, Achtung ohne Vorzeichenbit!

$$D_{(10)} = -b_m \cdot 2^m + \sum_{i=0}^{m-1} b_i \cdot 2^i + \sum_{i=1}^{n} b_i \cdot 2^{-i}$$

Theorie – Zweierkomplement

- Umrechnen einer positiven Zahlen in die korrespondierende negative Zahl:

	VZ	Betrag
positiv	0	$\boldsymbol{MSB, MSB_{-1}, MSB_{-2}, \cdots, LSB} = \left\lvert \boldsymbol{X_{(2)}} \right\rvert$
negativ	1	$\rightarrow \left\lvert X_{(2)} \right\rvert$ Bitweise invertieren → Eine $1_{(2)}$ muss an der LSB Stelle addiert werden $\overline{\boldsymbol{MSB}}, \overline{\boldsymbol{MSB_{-1}}}, \overline{\boldsymbol{MSB_{-2}}}, \cdots, (\overline{\boldsymbol{LSB}} + \boldsymbol{1_{(2)}})$

-> Vorzeichenbit hinzufügen

Aufgabe – Zweierkomplement

- −16.25(10) im Zweierkomplement als Binärzahl

- 101.01(2) (im Zweierkomplement) als Dezimalzahl

- 1011.0100(2) mQn Zahl mit m=3, n=4 als Dezimalzahl

Theorie – Rechnen

- Addition: normale schriftliche Addition
  - Überlauf auf nächste Stelle, falls zu gross
- Subtraktion: Zweierkomplement addieren (Resultat als Zweierkomplement betrachten)
- Bits mit 0 ergänzen für gleiche Anzahl Stellen

  <u>Ausnahme:</u> Bei Zweierkomplement vor dem VZ-Bit mit 1 ergänzen.

Addition:

$$11001.011_{(2)} + 111011.1_{(2)}$$

	1	1	1		1	1				
		0	1	1	0	0	1.	0	1	1
+		1	1	1	0	1	1.	1	0	0
=	1	0	1	0	1	0	0.	1	1	1

Aufgabe – Rechnen

- Zwei positive Binärzahlen A=$111011.01_{(2)}$ und B=$110001.1011_{(2)}$ sind gegeben. Gesucht ist C=A-B. Geben Sie A, -B und C als Zweierkomplementzahlen (7 Bits vor dem Komma, 4 nach dem Komma) sowie C als Dezimalzahl an.

Theorie und Beispiel – Multiplikation

1. Bitweise Multiplikation des Multiplikanden a mit b_i des Multiplikator.
2. Sukzessive Multiplikationen werden um ein Bit (0) nach links verschoben.
3. Anzahl Nachkommabits ergibt sich aus der Summe der Anzahl Nachk.bits der Operatoren.

$$
\begin{array}{r}
b_0 \cdot a \\
+b_1 \cdot a\ 0 \\
+b_2 \cdot a\ 0\ 0 \\
+b_3 \cdot a\ 0\ 0\ 0 \\
\hline
= \text{Sum}
\end{array}
$$

110*1011.1 =

Theorie – Codes

- Binäre Codes werden gebraucht, um Informationen (Zahlen) auf das Binärsystem nach einem gegebenen Konstruktionsprinzip abzubilden.

- Jeder Code hat eine bestimmte Eigenschaft
  - BCD: Jede Dezimalstelle wird als 4-Bit Binärzahl dargestellt
  - Gray: Einschrittiger Code

Binär	BCD	Excess-3	Aiken	4-2-2-1	Gray
0000	0		0	0	0
0001	1		1	1	1
0010	2		2	2	3
0011	3	0	3	3	2
0100	4	1	4		7
0101	5	2			6
0110	6	3		4	4
0111	7	4		5	5
1000	8	5			
1001	9	6			
1010		7			
1011		8	5		
1100		9	6	6	8
1101			7	7	9
1110			8	8	
1111			9	9	

Theorie – Parity Bits

- Zum Finden und Korrigieren von Fehlern
- Even Parity Bit: bei gerader Anzahl Bits 0, bei ungerader Anzahl Bits 1
- Odd Parity Bit: bei gerader Anzahl Bits 1, bei ungerader Anzahl Bits 0

BCD-CODE **Sender**

Dezimal Ziffer	2^3 8	2^2 4	2^1 2	2^0 1	P_E	P_O
0	0	0	0	0	0	1
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1		
4	0	1	0	0		
5	0	1	0	1		

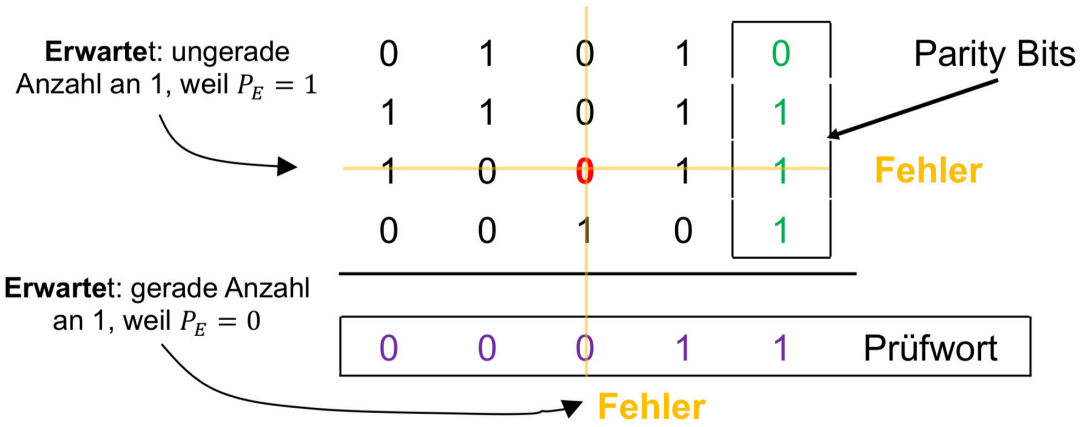
Theorie – Fehlerkorrektur

- Mithilfe von Parity bits
- Voraussetzung: nur ein einziger Fehler pro Viererpäckchen
- An jedes Wort (Zeile) wird ein Parity Bit hinzugefügt und am Ende ein Prüfwort (für Spalten) mitgeschickt

$$|01010|11011|10111|00101|00011|_{(2)}$$

$$|01010|11011|10011|00101|00011|_{(2)}$$

Fehlererkennung und –korrektur:



Aufgabe – Parity Bits

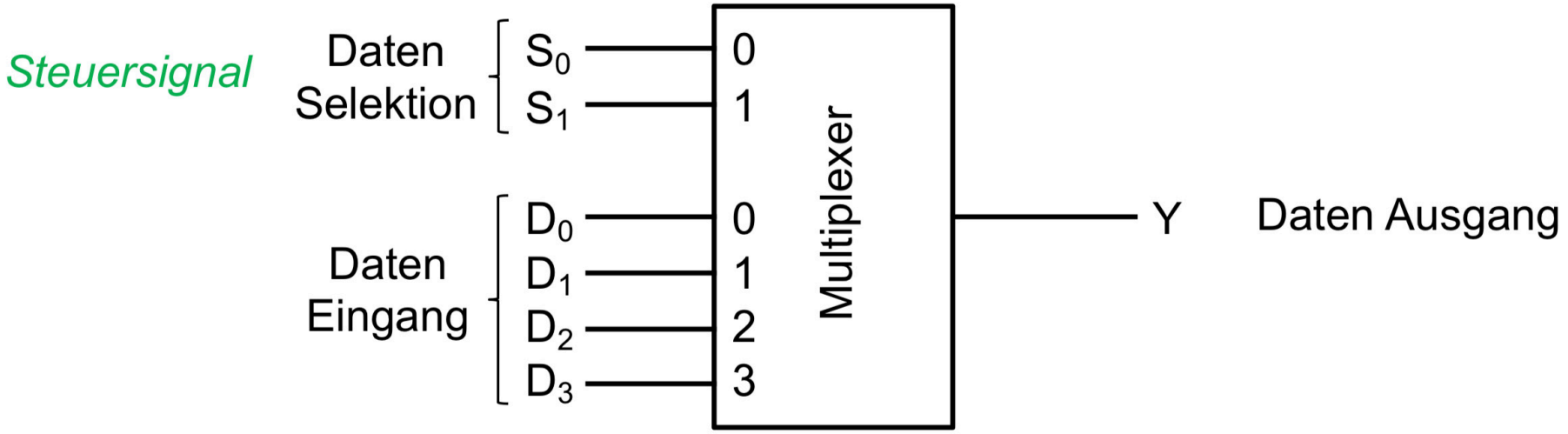
- Wir empfangen 010000011111001010101011000
- Besteht aus 4-Bit Wörtern mit je einem Odd-Parity-Bit
- Am Ende der Übertragung wird ein Prüfwort bestehend aus Even-Parity-Bits geliefert


Kahoot!

Theorie – Rechenschaltungen und Datenpfadkomponenten

Theorie – Multiplexer

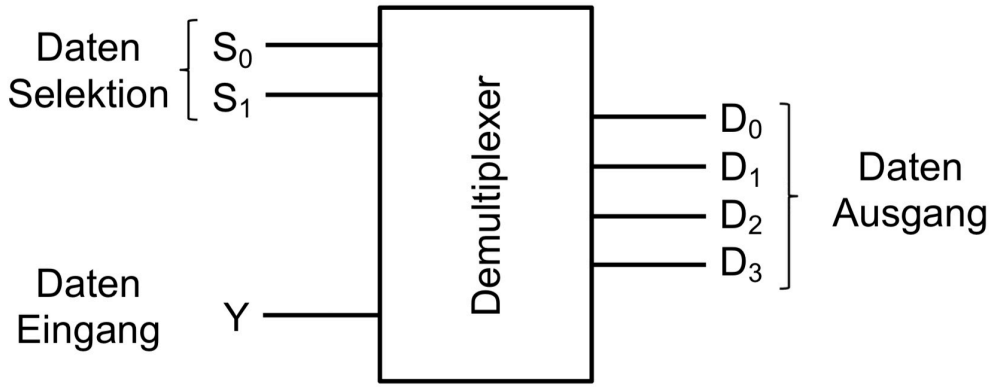
- Funktion: Durch Auswertung eines n-Bit-Auswahlsignals einen von 2^n Eingängen auf den Ausgang schalten.
- In anderen Worten ein Datenselektor



Theorie – Demultiplexer

- Gegenstück zum Multiplexer
- Nimmt Daten aus einem einzigen Kanal und verteilt dieses über mehrere Kanäle

  Beispiel: 1-zu-4 Demultiplexer



Die Ausgänge können zum Beispiel so programmiert werden:

$$D_0 = \overline{S_0} \wedge \overline{S_1} \wedge Y, \qquad D_1 = \overline{S_0} \wedge S_1 \wedge Y,$$

$$D_2 = S_0 \wedge \overline{S_1} \wedge Y, \qquad D_3 = S_0 \wedge S_1 \wedge Y$$

Theorie – Halbaddierer

- Ziel: Entwurf einer Schaltung, die zwei 1Bit Zahlen A und B addieren kann:

A B

$0 + 0 = 00$

$0 + 1 = 01$

$1 + 0 = 01$

$1 + 1 = 10$

CO Übertrag, Carry Out

SUM Summe

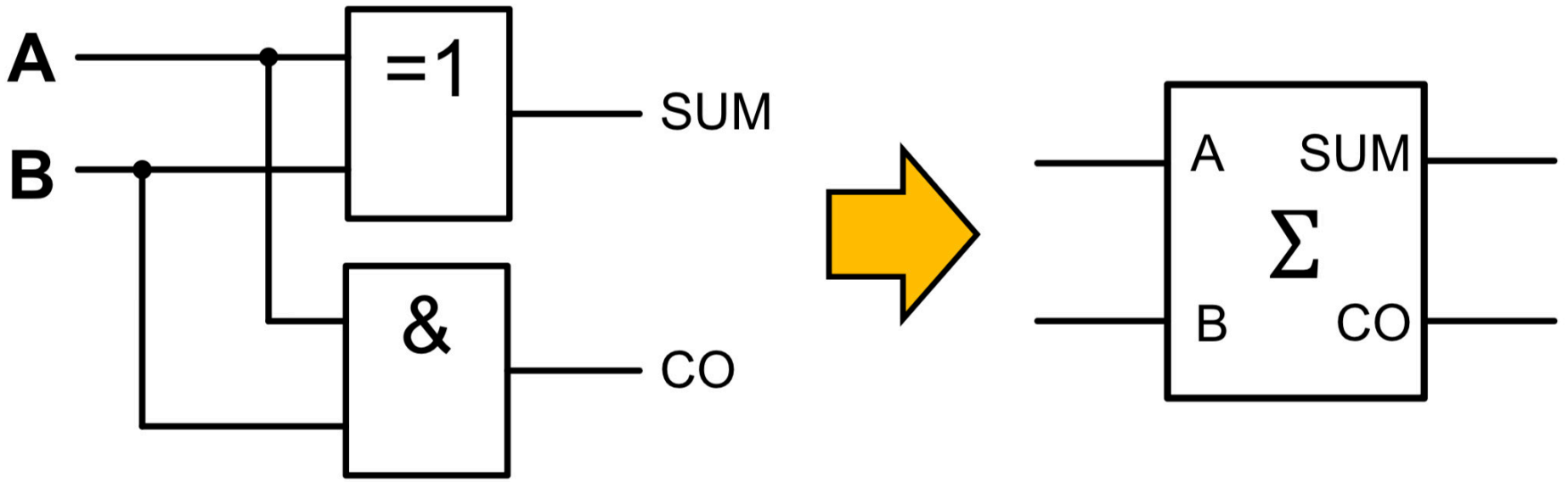
Wahrheitstabelle

A	B	SUM	CO
0	0		
0	1		
1	0		
1	1		

$SUM =$

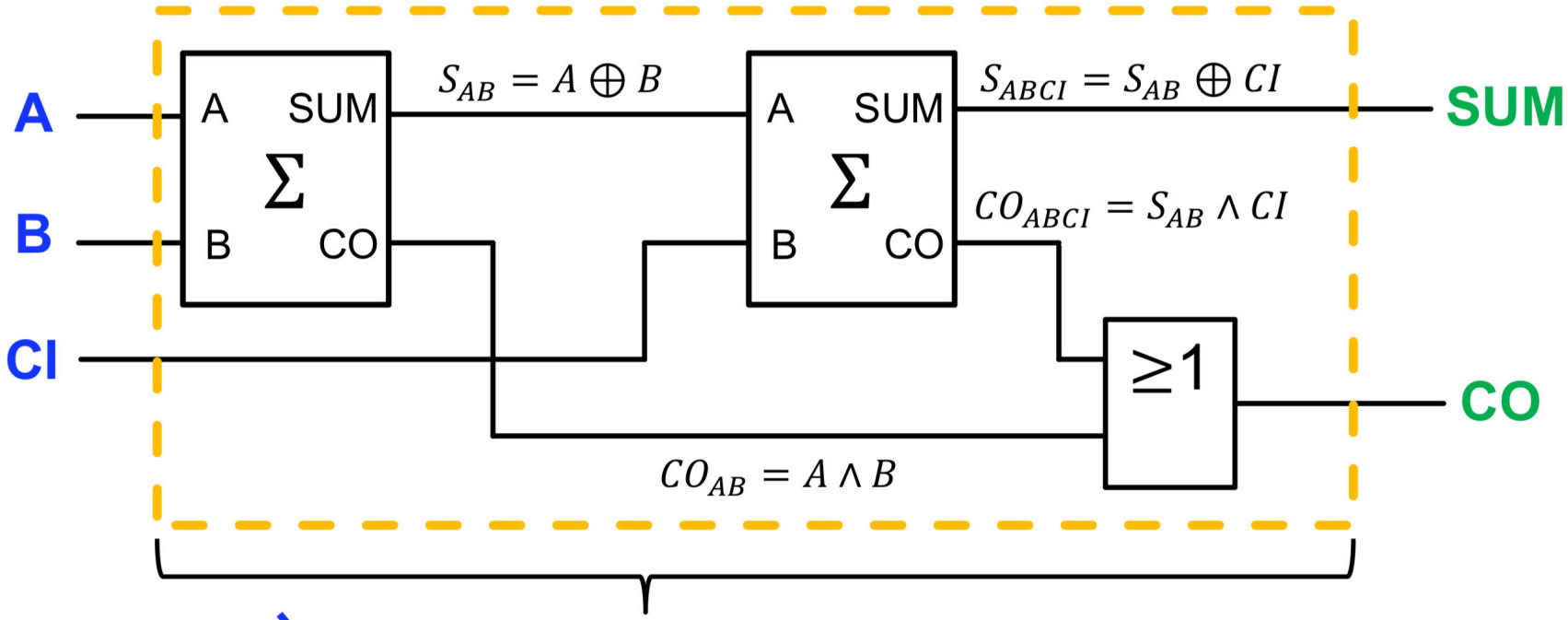
$CO =$

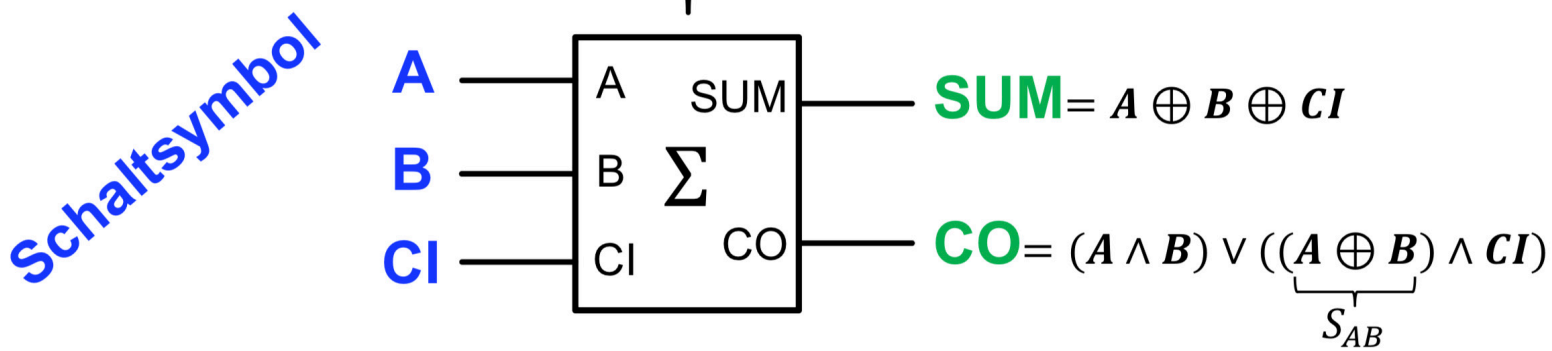
Theorie – Halbaddierer



Theorie – Volladdierer

- Ziel: Entwurf einer Schaltung, die drei 1Bit Zahlen A, B und CI addieren kann:





Theorie – Mehrbit-Addierer

- Ziel: Addition von mehrstelligen (Dual-)Zahlen
- Serienaddierer:
  - Pro Taktschritt wird eine Stelle addiert
- Paralleladdierer:
  - Pro Taktschritt werden alle Stellen addiert
  - 1. Paralleladdierer in der Normalform (Schaltnetz)
  - 2. Ripple-Carry Addierer (Kaskadierung von Volladdierern)
  - 3. Carry-Look-Ahead Addierer (Mischung aus 1.&2.)

Theorie – Paralleladdierer in der Normalform

- Ziel: Addition von zwei n-stelligen Dualzahlen mit möglichst geringer Laufzeit

- Für jedes Bit wird eine eigene Schaltgleichung generiert und in eine kombinatorische Schaltung umgesetzt. (mittels Min- oder Maxtermen)

$$S_0 = A_0 \oplus B_0 = (A_0 \wedge \overline{B_0}) \vee (\overline{A_0} \wedge B_0)$$

$$C_0 = A_0 \wedge B_0$$

$$S_1 = A_1 \oplus B_1 \oplus C_0 = ((A_1 \wedge \overline{B_1}) \vee (\overline{A_1} \wedge B_1)) \oplus (A_0 \wedge B_0)$$
$$= (A_0 \wedge B_0 \wedge A_1 \wedge B_1) \vee (A_0 \wedge B_0 \wedge \overline{A_1} \wedge \overline{B_1})$$

$$C_1 = (A_1 \wedge B_1) \vee (A_1 \wedge A_0 \wedge B_0) \vee (B_1 \wedge A_0 \wedge B_0)$$

Theorie – Paralleladdierer in der Normalform Eigenschaften

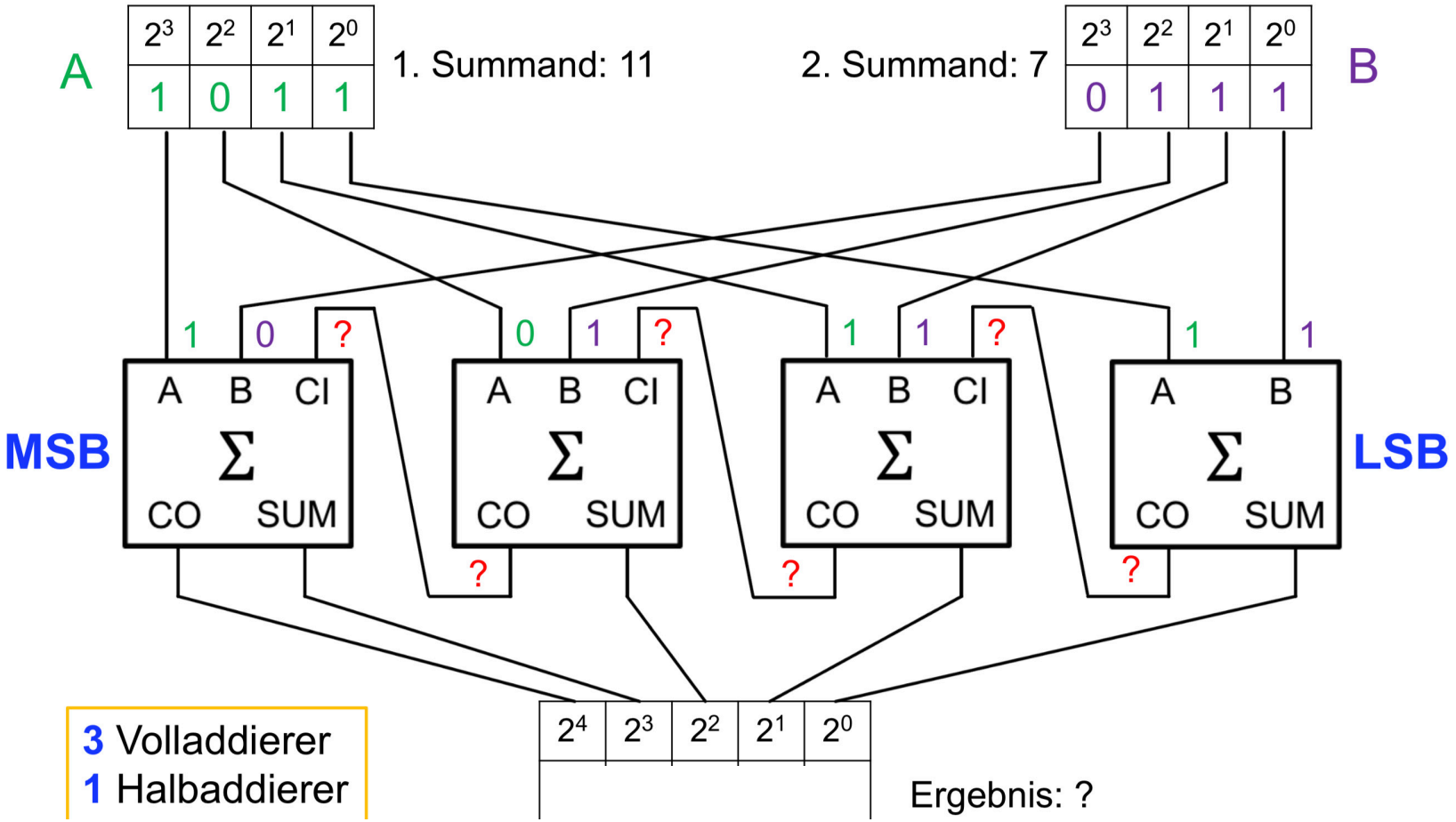
- Vorteile:
  - Zwischen Eingangsbit und Ausgangssignal gibt ex max. 3 Grundgatter
  - Laufzeit beträgt unabhängig von der Stellenanzahl der Summanden maximal 3 Gatterlaufzeiten => sehr schnell

- Nachteile:
  - Für die (n+1) Ausgänge einer Addition zweier n-stelliger Summanden müssen ~n*2^(2n-1) Min- bzw. Maxterme verknüpft warden.

- => Schneller aber schaltungsaufwendiger Addierer

Theorie – Ripple-Carry Addierer

4-Bit-Parallel-Addierschaltung (Kaskadierung von Addierern)



Theorie – Ripple-Carry Addierer Eigenschaften

- Vorteile:
  - Einfache Skalierung/Erweiterung möglich durch Kaskadierung
  - z.B. aus zwei 4-Bit-Addierern kann direct ein 8-Bit-Addierer aufgebaut warden
  - Schaltungsaufwand wächst linear mit Stellenzahl

- Nachteile:
  - Summe und Übertrag der i-ten Stelle können erst berechnet werden, wenn alle vorherigen Summen und Überträge berechnet wurden.
  - => Addierzeit wächst linear mit Stellenzahl

- => Langsamer aber einfacher Addierer

Theorie – Carry-Look-Ahead Addierer Eigenschaften

- Ziel: Kombination der Vorteile des Normalform und Ripple-Carry Addierers

- Implementation:
  - Kaskadierung der Addierer wie im Ripple-Carry Addierer
  - Berechung der Überträge erfolgt parallel zur Summenbildung mittels einer extra kombinatorischen Schaltung in einem einzigen Schritt.

- Eigenschaften:
  - Gleiche Geschwindigkeit wie Carry-Look-Ahead Addierer, braucht aber weniger Grundgatter
  - Aufwand ist grösser als beim Ripple-Carry

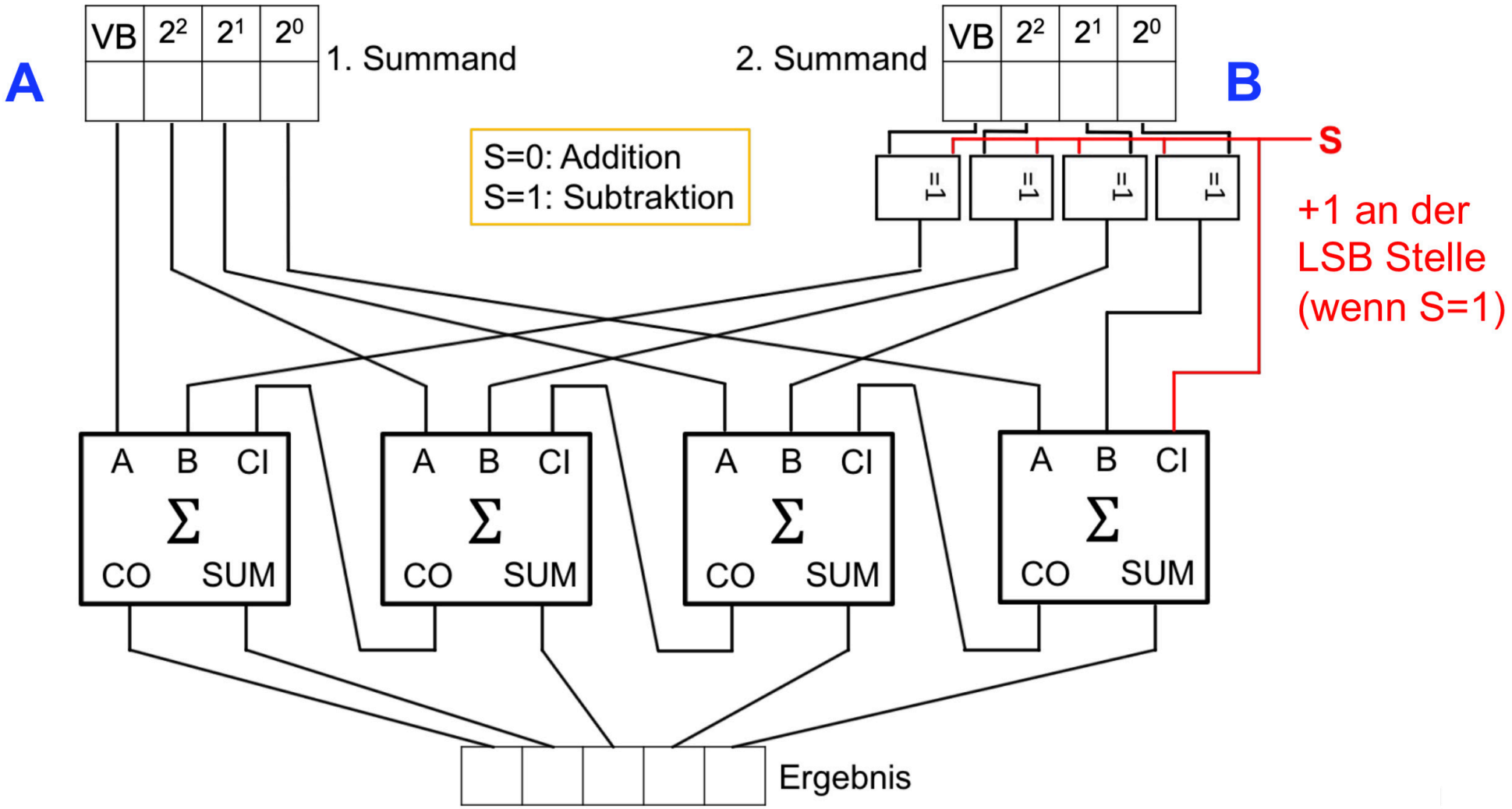
Theorie – Subtrahierer

- Ziel: Entwurf einer Schaltung, die Mehrbit-Addition und Subtraktion durchführt.
- Subtraktion: Zweierkomplement des Subtrahenden bilden und anschliessend normale Addition durchführen.
- Zweierkomplement bilden:
  - Bitinversion mit XOR-Gattern:

$B_{alt,i}$	S	$B_{neu,i} = B_{alt,i} \oplus S$	
0	0	0	Passiert nichts
1	0	1	
0	1	1	Bit Inversion
1	1	0	

  - +1 im LSB mit CI (siehe nächste Folie)

Theorie – Subtrahierer




Kahoot!