Kapitel 3 – Kombinatorische Logik

1. Kombinatorische Schaltkreise
2. Boolesche Algebren
3. Boolesche Ausdrücke, Normalformen, zweistufige Synthese
4. Berechnung eines Minimalpolynoms
5. **Arithmetische Schaltungen**
6. Anwendung: ALU von ReTI

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Prof. Dr. Christoph Scholl

Institut für Informatik

WS 2015/16

Arithmetische Schaltungen

- Addieren nach der Schulmethode: Carry-Ripple-Addierer.
- Effizienteres Addieren: Conditional-Sum-Addierer.
- Addition von Zweierkomplement-Zahlen.
- Subtrahierer.
- Multiplizierer.

Kosten von Schaltkreisen

- Um unterschiedliche Schaltkreise, die eine Funktion (z.B. Addierer) implementieren, miteinander zu vergleichen, benötigt man ein Kostenmaß.

Definition

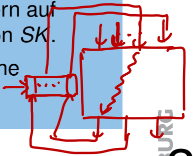
Die Kosten $C(SK)$ eines Schaltkreises SK sind durch die Anzahl seiner Gatter gegeben.

- Deutet auf die Fläche und den Energieverbrauch des resultierenden Hardware-Blocks hin.

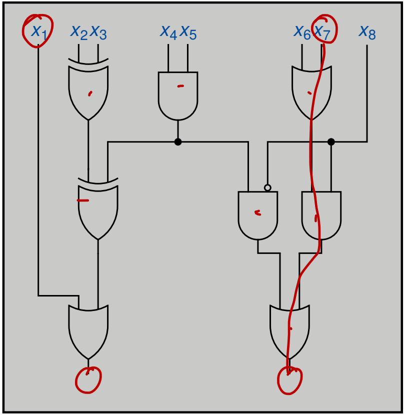
Definition

Die Tiefe $depth(SK)$ eines Schaltkreises ist die maximale Anzahl von Gattern auf einem Pfad von einem beliebigen Eingang zu einem beliebigen Ausgang von SK.

- Deutet auf die Signallaufzeit durch SK und somit die maximal mögliche Taktfrequenz (Geschwindigkeit) des Schaltkreises hin.



Beispiel: Kosten und Tiefe



$C(SK) = 8$

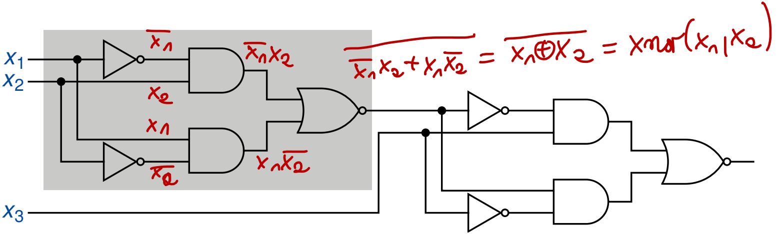
$Depth(SK) = 3$

(Macht nur dann Sinn, wenn die Gatter aus der Gatterbibliothek BIB in der Realität ähnliche Kosten (Fläche) und ähnliche Verzögerung haben.)

z.B. $BIB = \mathbb{B}_1 \cup \mathbb{B}_2$

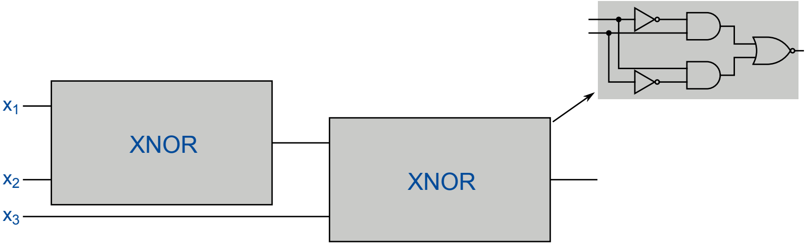
Teilschaltkreise, hierarchischer Entwurf (informell)

- Illustration eines Teilschaltkreises.



Hierarchische Schaltkreise

- In hierarchischen Schaltkreisen sind Teilschaltkreise durch Symbole ersetzt.
- Den zugehörigen („flachen”) Schaltkreis erhält man, indem man die Symbole durch Einsetzen der Teilschaltkreise wieder entfernt.



Wiederholung Zahlendarstellung

Sei $a = a_{n-1} \dots a_0$ eine Folge von Ziffern, $a_i \in \{0, 1\}$.

- Binärdarstellung: $\langle a \rangle = \sum_{i=0}^{n-1} a_i 2^i$
- Zweierkomplement: $[a_n a_{n-1} \dots a_0] = \sum_{i=0}^{n-1} a_i 2^i - a_n 2^n$
- Rechenregel: $-[a] = [\overline{a}] + 1$

mit $\overline{a} = \overline{a}_n \overline{a}_{n-1} \dots \overline{a}_0$.

7

Addierer für nichtnegative Zahlen

- **Gegeben:**
  2 positive Binärzahlen $\langle a\rangle = \langle a_{n-1}\dots a_0\rangle, \langle b\rangle = \langle b_{n-1}\dots b_0\rangle$ mit Eingangsübertrag $c \in \{0,1\}$.

- **Gesucht:**
  Schaltkreis, der Binärdarstellung s von $\langle a\rangle + \langle b\rangle + c$ berechnet.

- Wegen $\langle a\rangle + \langle b\rangle + c \leq 2\cdot(2^n - 1) + 1 = 2^{n+1} - 1$ genügen $n+1$ Stellen für die Darstellung von s, d.h. der Schaltkreis hat $n+1$ Ausgänge.

Formale Definition n-Bit-Addierer

- Ein *n-Bit-Addierer* ist ein Schaltkreis, der die folgende boolesche Funktion berechnet:

$$+_n : \mathbb{B}^{2n+1} \to \mathbb{B}^{n+1},$$
$$+_n : (a_{n-1}, \dots, a_0, b_{n-1}, \dots, b_0, c) = (s_n, \dots, s_0)$$
$$\text{mit } \langle s \rangle = \langle s_n \dots s_0 \rangle = \langle a_{n-1} \dots a_0 \rangle + \langle b_{n-1} \dots b_0 \rangle + c$$

$$\sum_{i=0}^{n} s_i \cdot 2^i = \sum_{i=0}^{n-1} a_i \cdot 2^i + \sum_{i=0}^{n-1} b_i \cdot 2^i + c$$

9

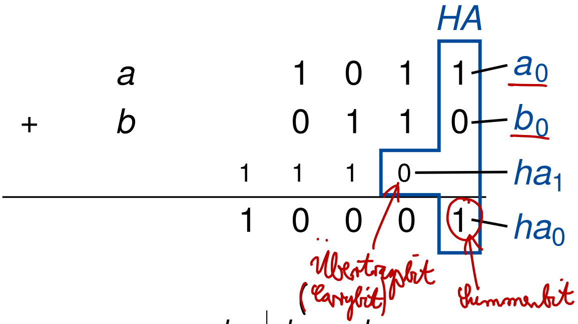
Addieren nach der Schulmethode (1/4)

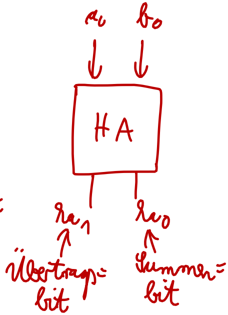
- Wir werden im Folgenden den einfachsten Addierertypen einführen, der die „Schulmethode" umsetzt.
- Hierzu werden einige Grundschaltungen (Halb- und Volladierer) notwendig sein.
- Beispiel für die Schulmethode:

$$\begin{array}{r} 4378 \\ +\ 56{}_{1}13 \\ \hline 9991 \end{array}$$

$$\begin{array}{clr} & a & 1011 \to 11 \\ + & b & 0110 \to 6 \\ & & {\scriptstyle 1\,1\,1\,0} \\ \hline & & 10001 \to 17 \end{array}$$

Addieren nach der Schulmethode (2/4)





a_0	b_0	ha_1	ha_0	
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	2

Halbaddierer (Half Adder, *HA*)

- Ein Halbaddierer dient zur Addition zweier 1-Bit-Zahlen ohne Eingangsübertrag.

- Er berechnet die Funktion $ha : \underline{\mathbb{B}^2} \to \underline{\mathbb{B}^2}$ mit $ha(a_0, b_0) = (ha_1, ha_0)$, wobei $2ha_1 + ha_0 = a_0 + b_0$.

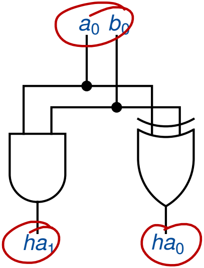
  $\langle ha_1, ha_0 \rangle = a_0 + b_0$

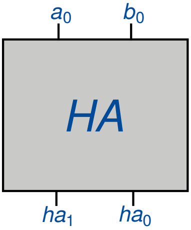
a_0	b_0	ha_1	ha_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$ha_0(a_0, b_0) = a_0 \oplus b_0$$
$$ha_1(a_0, b_0) = a_0 \wedge b_0$$

12

Schaltkreis eines Halbaddierers



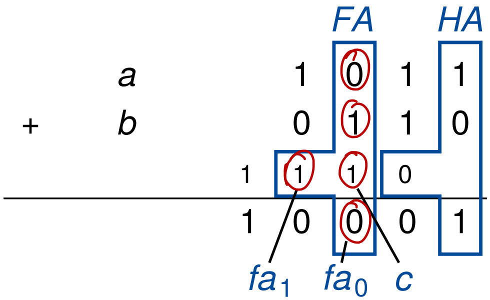


dabei gilt:

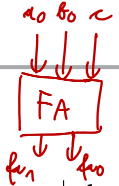
$C(HA) = 2, \quad depth(HA) = 1$

13

Addieren nach der Schulmethode (3/4)



Volladierer (Full Adder, *FA*)



- Ein Volladdierer dient zur Addition zweier 1-Bit-Zahlen mit Eingangsübertrag.

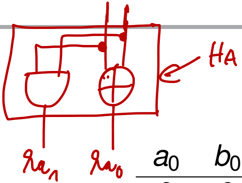
- Er berechnet die Funktion
  $fa : \mathbb{B}^3 \to \mathbb{B}^2$ mit
  $fa(a_0, b_0, c) = (fa_1, fa_0)$
  wobei $2fa_1 + fa_0 = a_0 + b_0 + c$
  $\langle fa_1, fa_0 \rangle$

a_0	b_0	c	fa_1	fa_0	
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

$$fa_0(a_0, b_0, c) = xor_3(a_0, b_0, c) = a_0 \oplus b_0 \oplus c = \boxed{(a_0 \oplus b_0)} \oplus c$$

$$fa_1(a_0, b_0, c) = a_0 b_0 + a_0 c + b_0 c = a_0 b_0 + c \cdot (a_0 + b_0) = a_0 b_0 + c \cdot \boxed{(a_0 \oplus b_0)}$$

Volladdierer als Funktion von *HA*s



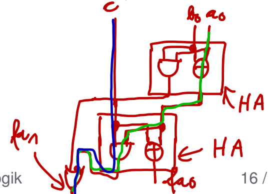
Aus der Tabelle folgt:

$fa_0 = a_0 \oplus b_0 \oplus c = (a_0 \oplus b_0) \oplus c$
$= ha_0(c, ha_0(a_0, b_0))$

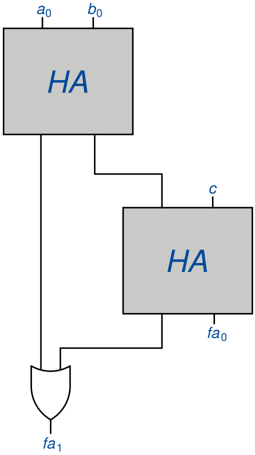
$fa_1 = (a_0 \wedge b_0) \vee (c \wedge (a_0 \oplus b_0))$
$= ha_1(a_0, b_0) + ha_1(c, ha_0(a_0, b_0))$

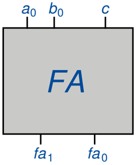
a_0	b_0	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Kosten und Tiefe eines *FA*:
$C(FA) = 5, \quad depth(FA) = 3$

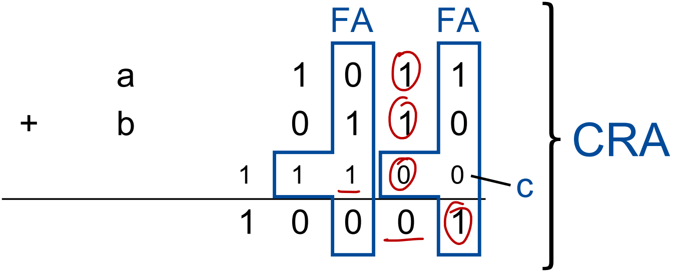


Schaltkreis eines Volladdierers



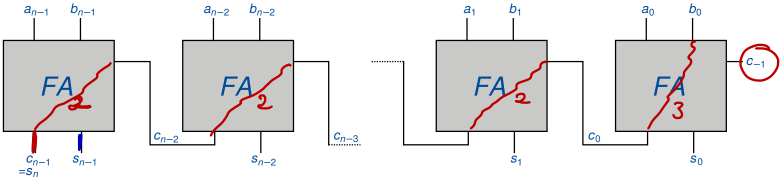


Addieren nach der Schulmethode (4/4)



Aufbau eines Carry-Ripple-Addierers

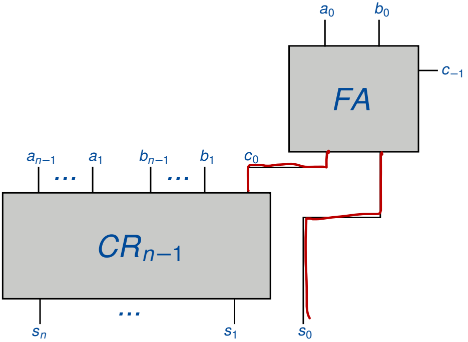
Übertrag



$$
\begin{array}{rccccccc}
 & a_{n-1} & a_{n-2} & \dots & & a_1 & a_0 & \\
+ & b_{n-1} & b_{n-2} & \dots & & b_1 & b_0 & \\
+ c_{n-1} & c_{n-2} & c_{n-3} & & c_1 & c_0 & c_{-1} & \\
\hline
c_{n-1} & s_{n-1} & s_{n-2} & \dots & & s_1 & s_0 & \\
\| & & & & & & & \\
s_n & & & & & & &
\end{array}
$$

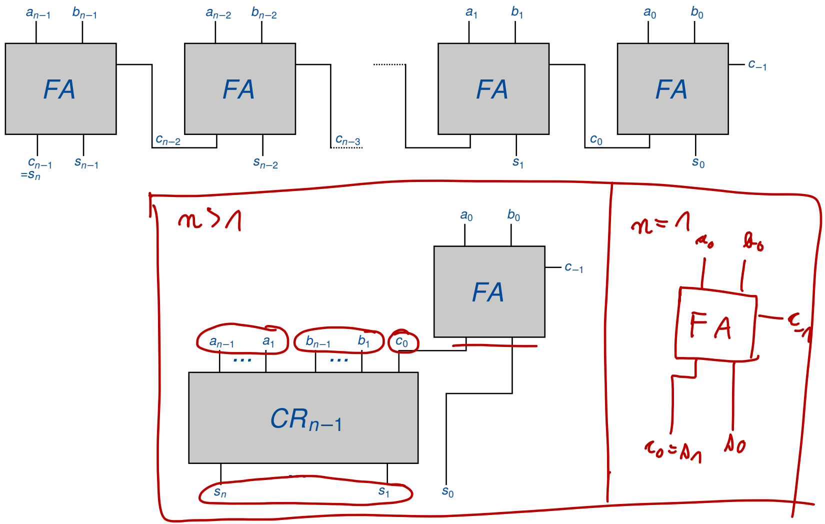
Induktive Definition des Carry-Ripple-Addierers CR

- Für $n = 1$: $CR_1 = FA$
- Für $n > 1$: Folgender Schaltkreis:



c_{-1}:	Eingangsübertrag
c_i:	Übertrag von
	Stelle i nach $i+1$

Zwei (identische) Darstellungen von CR



Carry-Ripple-Addierer

Satz

CR_n ist ein n-Bit-Addierer.

$(\langle a_{n-1} \ldots a_0 \rangle + \langle b_{n-1} \ldots b_0 \rangle + c_{-1} = \langle s_n \ldots s_0 \rangle)$

Beweis (durch Induktion):

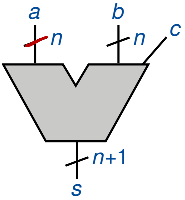
- $n = 1$ $(CR_1 = FA)$ ✓
- $n-1 \to n$: Eingabe an CR_n : $(a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0, c_{-1})$
  Zeige für Ausgabe $(s_n, \ldots, s_0)$ von CR_n:
  $\langle s \rangle = \langle s_n \ldots s_0 \rangle = \langle a_{n-1} \ldots a_0 \rangle + \langle b_{n-1} \ldots b_0 \rangle + c_{-1}$.
- Nach Induktionsvoraussetzung gilt für CR_{n-1}:
  $\langle s_n \ldots s_1 \rangle = \langle a_{n-1} \ldots a_1 \rangle + \langle b_{n-1} \ldots b_1 \rangle + c_0$. (a)
  Wegen FA-Eigenschaft gilt $\langle c_0, s_0 \rangle = a_0 + b_0 + c_{-1}$. (b)
  $2c_0 + s_0$
- Insgesamt: $\langle s_n \ldots s_0 \rangle = 2 \cdot \langle s_n \ldots s_1 \rangle + s_0$
  $\overset{(a)}{=} 2 \cdot (\langle a_{n-1} \ldots a_1 \rangle + \langle b_{n-1} \ldots b_1 \rangle + c_0) + s_0$
  $= 2 \cdot (\langle a_{n-1} \ldots a_1 \rangle + \langle b_{n-1} \ldots b_1 \rangle) + 2 \cdot c_0 + s_0$ $\to a_0 + b_0 + c_{-1}$
  $\overset{(b)}{=} 2 \cdot \langle a_{n-1} \ldots a_1 \rangle + a_0 + 2 \cdot \langle b_{n-1} \ldots b_1 \rangle + b_0 + c_{-1}$
  $= \langle a \rangle + \langle b \rangle + c_{-1}$

$$\langle s_n \ldots s_0 \rangle = \sum_{i=0}^{n} s_i \cdot 2^i = \sum_{i=1}^{n} s_i \cdot 2^i + s_0$$
$$= \sum_{i=0}^{n-1} s_{i+1} 2^{i+1} + s_0$$
$$= 2 \cdot \sum_{i=0}^{n-1} s_{i+1} \cdot 2^i + s_0$$
$$= 2 \cdot \langle s_n \ldots s_1 \rangle + s_0$$

22

Schaltbild und Komplexität von *CR*

- $C(CR_n) = n \cdot C(FA) = 5n.$
- $depth(CR_n) = 3 + 2(n-1).$
- Sowohl die Kosten als auch die Tiefe von *CR* sind somit linear in *n*.



- Es gibt (asymptotisch) bessere Addierer. Wir werden hier den Conditional-Sum-Addierer kennen lernen, für den wir wieder eine Hilfsschaltung (Multiplexer) benötigen.
- Eine weitere wichtige Schaltung ist der Inkrementer.

n-Bit-Inkrementer

Definition

Ein n-Bit-Inkrementer INC_n berechnet die Funktion
$inc_n : \mathbb{B}^{n+1} \to \mathbb{B}^{n+1}$
$inc_n(a_{n-1}, \ldots, a_0, c) = (s_n, \ldots, s_0)$ mit $\langle s_n \ldots s_0 \rangle = \langle a \rangle + c$

- Ein Inkrementer ist ein Addierer mit $b_i = 0$ für alle i.
  $\Rightarrow$ Ersetze in CR_n die FA durch HA.

- Kosten und Tiefe:
  - $C(INC_n) = n \cdot C(HA) = 2n$
  - $depth(INC_n) = n \cdot depth(HA) = n$

n-Bit-Multiplexer

Definition

Ein n-Bit-Multiplexer MUX_n berechnet die Funktion

Select-Bit

$$sel_n : \mathbb{B}^{2n+1} \to \mathbb{B}^n$$

$$sel_n(a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0, s) = \begin{cases} (a_{n-1} \ldots a_0), & \text{falls } s = 1 \\ (b_{n-1} \ldots b_0), & \text{falls } s = 0 \end{cases}$$

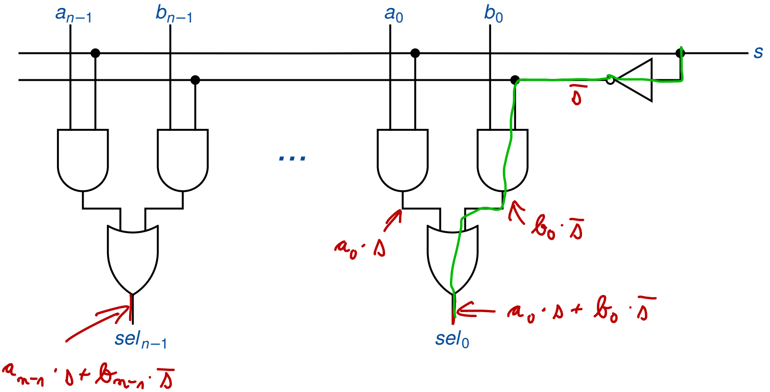
- Es gilt: $(sel_n)_i = s \cdot a_i + \overline{s} \cdot b_i$

$s=1: (sel_n)_i(a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0, 1) = 1 \cdot a_i + \overline{1} \cdot b_i = a_i$

$s=0: (sel_n)_i(a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0, 0) = 0 \cdot a_i + \overline{0} \cdot b_i = 0 + 1 \cdot b_i = b_i$

25

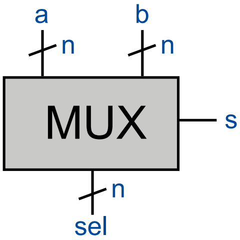
Aufbau von MUX_n



Schaltbild und Kosten MUX_n

Kosten und Tiefe:

$C(MUX_n) = 3n + 1.$
$depth(MUX_n) = 3.$



27

Rückkehr zum Addierer

Gibt es billigere Addierer als CR_n?

Notation: Sei $f \in \mathbb{B}_n$.
Dann sind $C(f)$ und $depth(f)$ wie folgt definiert:

$$C(f) := \min\{C(SK) \mid f_{SK} = f\}$$

$$depth(f) := \min\{depth(SK) \mid f_{SK} = f\}$$

(Wir legen hier die Bibliothek $BIB = \mathbb{B}_1 \cup \mathbb{B}_2$ zugrunde.)

Untere Schranken:
$C(+_n) \geq 2 \cdot n, \quad depth(+_n) \geq \log(n) + 1$

Binäre Bäume mit $2n+1$ Blättern haben $2n$ innere Knoten.
Binäre Bäume mit n Blättern haben mindestens Tiefe $\lceil \log(n) \rceil$.

Im Folgenden sei $n = 2^k$.

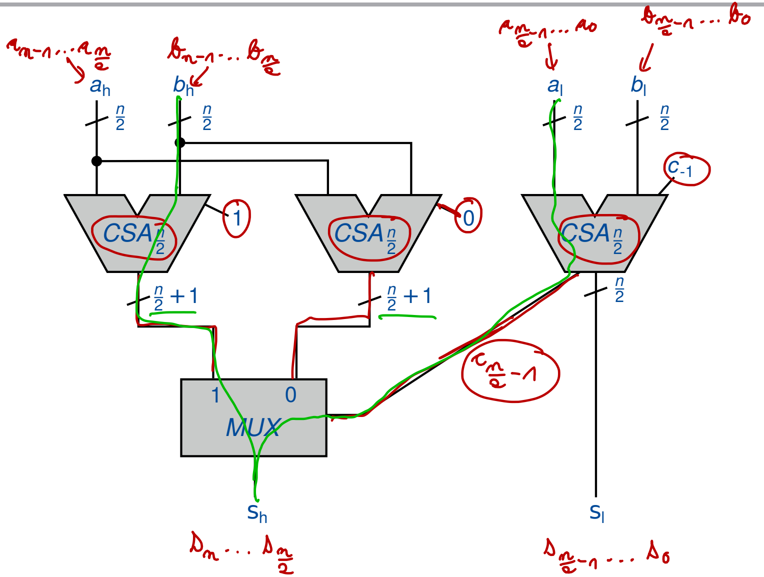
Conditional-Sum-Addierer (*CSA*)

- Idee: Nutze Parallelverarbeitung, um Tiefe zu reduzieren!

$$
\begin{array}{rr|r}
 & a_{n-1} \dots a_{\frac{n}{2}} & a_{\frac{n}{2}-1} \dots a_0 \\
+ & b_{n-1} \dots b_{\frac{n}{2}} & b_{\frac{n}{2}-1} \dots b_0 \\
+ & c_{\frac{n}{2}-1} & c_{-1} \\
\hline
 & s_n s_{n-1} \dots s_{\frac{n}{2}} & s_{\frac{n}{2}-1} \dots s_0
\end{array}
$$

- $CSA_1 = \underline{FA}$.
- CSA_n: Siehe nächste Folie.
- Im Folgenden sei $n = 2^k$.

Aufbau von CSA_n



Komplexität von CSA_n: Tiefe

$n = 2^k$

Satz

CSA_n hat Tiefe $\leq 3\log(n) + 3$.

Beweis:

- $n = 1$: $depth(CSA_1) = depth(FA) = 3$.

$depth(CSA_n) \leq depth(CSA_{\frac{n}{2}}) +$
3

- $n > 1$: $depth(CSA_n) \leq depth(CSA_{\frac{n}{2}}) + depth(MUX_{\frac{n}{2}+1})$

3

$\leq depth(CSA_{\frac{n}{2}}) + 3$
$\leq depth(CSA_{\frac{n}{4}}) + 3 + 3$
$\leq depth(CSA_{\frac{n}{8}}) + 3 + 3 + 3$
$\dots$
$\leq depth(CSA_{\frac{n}{2^k}}) + k \cdot 3$
$= depth(CSA_1) + k \cdot 3 = 3 + k \cdot 3$
$\leq 3 \cdot (k+1) = 3\log(n) + 3.$

$n = 2^k \iff k = \log n$

Komplexität von CSA_n: Kosten

Satz

$C(CSA_n) = \underline{10n^{\log(3)} - 3n - 2.}$ $\approx 10n^{1.58} - 3n - 2$

$$\underline{C(CSA_n) = 3 \cdot C(CSA_{\frac{n}{2}}) + \frac{3}{2} \cdot n + 4}$$

$n = 2^k$

Beweis: ...

$$C(CSA_1) = C(FA) = 5$$

$$C(CSA_n) = 3 \cdot C(CSA_{n/2}) + C(MUX)_{\frac{n}{2}+1}$$

$$= 3 \cdot \underbrace{C(CSA_{\frac{n}{2}})} + \frac{3}{2} n + 4$$

$$= 3 \cdot \left[3 \cdot C(CSA_{\frac{n}{4}}) + \frac{3}{2} \cdot \frac{n}{2} + 4 \right] + \frac{3}{2} n + 4$$

$$= 3^2 \cdot \underbrace{C(CSA_{\frac{n}{4}})} + \underbrace{\frac{3}{2} n + 4} + \underbrace{3 \cdot \left(\frac{3}{2} \cdot \frac{n}{2} + 4\right)}$$

$$= 3^2 \cdot \left[3 \cdot C(CSA_{\frac{n}{8}}) + \underbrace{\frac{3}{2} \cdot \frac{n}{4} + 4} \right] + \underbrace{\frac{3}{2} n + 4} + 3 \cdot \left(\frac{3}{2} \cdot \frac{n}{2} + 4\right)$$

$$= 3^3 \cdot \underbrace{C(CSA_{\frac{n}{8}})} + \frac{3}{2} n + 4 + 3 \cdot \left(\frac{3}{2} \cdot \frac{n}{2} + 4\right) + 3^2 \cdot \left(\frac{3}{2} \cdot \frac{n}{4} + 4\right)$$

$$= 3^3 \cdot \left[3 \cdot C(CSA_{\frac{n}{16}}) + \frac{3}{2} \cdot \frac{n}{8} + 4 \right] + \frac{3}{2} n + 4 + 3 \cdot \left(\frac{3}{2} \cdot \frac{n}{2} + 4\right) + 3^2 \cdot \left(\frac{3}{2} \cdot \frac{n}{4} + 4\right)$$

-32-

$$= 3^4\cdot C(CSA_{\frac{n}{16}}) + \underline{\tfrac{3}{2}n+4} + \underline{3\cdot\left(\tfrac{3}{2}\tfrac{n}{2}+4\right)} + \underline{3^2\cdot\left(\tfrac{3}{2}\tfrac{n}{4}+4\right)} + \underline{3^3\cdot\left(\tfrac{3}{2}\cdot\tfrac{n}{8}+4\right)}$$

$$\vdots$$

$$= 3^k\, \underbrace{C(CSA_{\frac{n}{2^k}})}_{= C(CSA_1) = 5} + \sum_{i=0}^{k-1} 3^i\left(\tfrac{3}{2}\tfrac{n}{2^i}+4\right)$$

(Beweis durch Induktion über k)

$n = 2^k$

$$= \boxed{3^{\log n}\cdot 5 + \sum_{i=0}^{\log n-1} \underline{3^i\left(\tfrac{3}{2}\cdot\tfrac{n}{2^i}+4\right)}}$$

$$= 3^{\log n}\cdot 5 + 4\cdot\underline{\sum_{i=0}^{\log n-1} 3^i} + \tfrac{3}{2}n\underline{\sum_{i=0}^{\log n-1}\left(\tfrac{3}{2}\right)^i}$$

Geometrische Summenformel:

$$\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1} \quad \text{für } q \neq 1$$

$$= 5\cdot 3^{\log n} + 4\cdot\left(\frac{3^{\log n}-1}{3-1}\right) + \tfrac{3}{2}n\cdot\frac{\left(\frac{3}{2}\right)^{\log n}-1}{\frac{3}{2}-1} =$$

$$= 5\cdot 3^{\log n} + 2\cdot\left(3^{\log n}-1\right) + 3n\cdot\left[\frac{3^{\log n}}{n}-1\right] =$$

$$= 7\cdot 3^{\log n} - 2 + 3\cdot 3^{\log n} - 3n = 10\cdot 3^{\log n} - 3n - 2$$

$$= 10\cdot n^{\log 3} - 3n - 2$$

$$3^{\log n} = \left(2^{\log 3}\right)^{\log n} = \left(2^{\log n}\right)^{\log 3} = n^{\log 3}$$

Kosten und Tiefe von Addierern

- Man kann den hier vorgestellten *CSA* in einfacher Weise modifizieren, so dass
  - Tiefe = $O(\log(n))$,
  - Kosten = $O(n \cdot \log(n))$.
- Es gibt auch Addierer mit linearen Kosten und logarithmischer Tiefe.
  - Carry-Lookahead-Addierer (*CLA*).
  - $C(CLA_n) \leq \underline{11n}$,
  - $depth(CLA_n) \leq \underline{4 \cdot \log(n) + 2}$.

34

Addition von Zweierkomplementzahlen

$[a_n \dots a_0]_2 = \sum_{i=0}^{n-1} a_i \cdot 2^i - a_n \cdot 2^n$

$[b_n \dots b_0]_2 = \sum_{i=0}^{n-1} b_i \cdot 2^i - b_n \cdot 2^n$

- Auszurechnen ist:

  $[a_n a_{n-1} \dots a_0] + [b_n b_{n-1} \dots b_0] = (-a_n 2^n) + (-b_n 2^n) + \sum_{i=0}^{n-1} a_i 2^i + \sum_{i=0}^{n-1} b_i 2^i$

  $[s_n \dots s_0]$

- Im Fall von $(n+1)$-Bit-Zweierkomplementzahlen können Ergebnisse im Bereich $R_n = \{-2^n, \dots, 2^n - 1\}$ dargestellt werden; andernfalls kommt es zu einem Überlauf.

- Der Satz auf der nächsten Folie sagt aus:
  - Kommt es bei der Addition nicht zu einem Überlauf, so kann man den „gewöhnlichen" Binäraddierer zur Addition von Zweierkomplementzahlen benutzen.
  - Ob es zu einem Überlauf kommt, lässt sich anhand von Werten a_n, b_n und s_n im Binäraddierer entscheiden.

Zweierkomplement-Addition formal

Satz

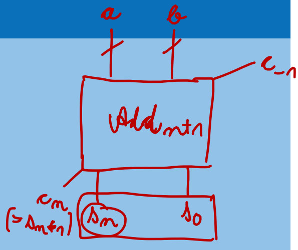
Seien $a, b \in \mathbb{B}^{n+1}$, $c_{-1} \in \{0,1\}$ und $s \in \{0,1\}^{n+1}$, so dass $\langle c_n, s \rangle = \langle a \rangle + \langle b \rangle + c_{-1}$.

Dann gilt:

$\{-2^n, \ldots, 2^n - 1\}$

(i) $[a] + [b] + c_{-1} \notin R_n \Leftrightarrow \boxed{(a_n = b_n) \wedge (b_n \neq s_n)}$

(ii) $[a] + [b] + c_{-1} \in R_n \Rightarrow [a] + [b] + c_{-1} = [s]$,

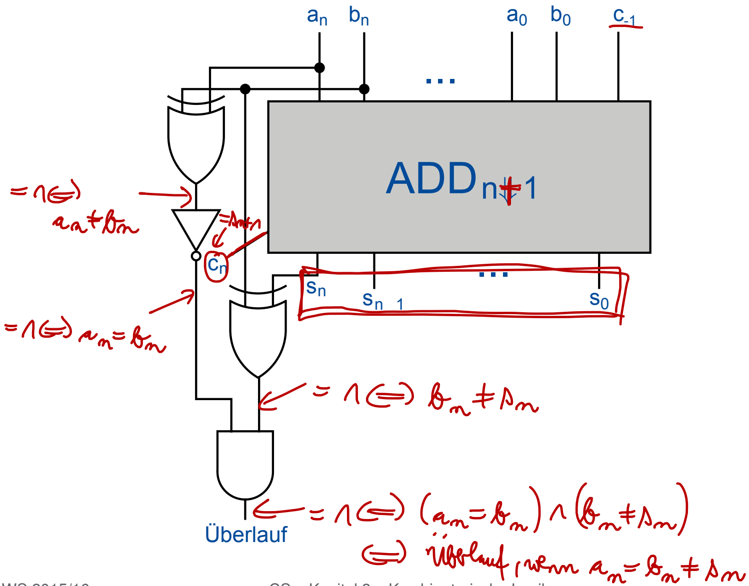


- Beweis durch Fallunterscheidung ($[a], [b]$ beide positiv, beide negativ, $[a]$ positiv / $[b]$ negativ) und Nachrechnen (~~Induktion~~).

- Man kann einen alternativen Überlauftest zeigen:

$$[a] + [b] + c_{-1} \notin R_n \Leftrightarrow c_n \neq c_{n-1}$$

Addierer für $(n+1)$-Bit-Zweierkomplement-Zahlen



$$[s_n \dots s_0] = \underbrace{\sum_{i=0}^{n-1} s_i \cdot 2^i - s_n \cdot 2^n}_{[s]} \stackrel{?}{=} \underbrace{\sum_{i=0}^{n-1} a_i \cdot 2^i - a_n \cdot 2^n}_{[a]} + \underbrace{\sum_{i=0}^{n-1} b_i \cdot 2^i - b_n \cdot 2^n}_{[b]}$$

$R_n = \{-2^n, \dots, 2^n - 1\}$

Bsp.: $n = 3$, $R_3 = \{-8, \dots, 7\}$

$$\begin{array}{rl} \boxed{0}001 & \hat{=}\ 1 \\ \boxed{0}011 & \hat{=}\ 3 \\ \hline (0)\ \boxed{0}100 & \hat{=}\ 4 \end{array}$$

kein Überlauf

$$\begin{array}{rl} \boxed{0}100 & \hat{=}\ 4 \\ \boxed{0}101 & \hat{=}\ 5 \\ \hline (0)\ \boxed{1}001 & \quad 9 \end{array}$$

Überlauf

$$\begin{array}{rl} \boxed{1}010 & \hat{=}\ -6 \\ +_1\ \boxed{0}110 & \hat{=}\ 6 \\ \hline (1)\ \boxed{0}000 & \hat{=}\ 0 \end{array}$$

kein Überlauf

$$\begin{array}{rl} \boxed{0}110 & \hat{=}\ 6 \\ +\ \boxed{1}101 & \hat{=}\ -3 \\ \hline (1)\ \boxed{0}011 & \hat{=}\ 3 \end{array}$$

kein Überlauf

$$\begin{array}{rl} \boxed{1}100 & \hat{=}\ -4 \\ +_1\ \boxed{1}011 & \hat{=}\ -5 \\ \hline (1)\ \boxed{0}111 & \end{array}$$

Überlauf

$$\begin{array}{rl} \boxed{1}111 & \hat{=}\ -1 \\ \boxed{1}101 & \hat{=}\ -3 \\ \hline (1)\ \boxed{1}100 & \hat{=}\ -4 \end{array}$$

kein Überlauf

Beweis:

$$
\begin{array}{cc|cccc}
 & a_n & a_{n-1} & \dots & & a_0 \\
 & b_n & b_{n-1} & \dots & & b_0 \\
c_n & c_{n-1} & c_{n-2} & & c_0 & c_{-1} \\
\hline
c_n & s_n & s_{n-1} & & s_1 & s_0
\end{array}
$$

Sei $a' = a_{n-1} \dots a_0$

$b' = b_{n-1} \dots b_0$

$s' = s_{n-1} \dots s_0$

Dann gilt: $\underline{\langle a' \rangle + \langle b' \rangle + c_{-1}} = \underline{\langle c_{n-1}\, s' \rangle}$

$c_{n-1} = 1 \iff \langle a' \rangle + \langle b' \rangle + c_{-1} \geq 2^n \Leftarrow$

$[a] = \sum_{i=0}^{n-1} a_i \cdot 2^i - a_n \cdot 2^n = \underline{\langle a' \rangle - a_n \cdot 2^n}$

$[b] = \langle b' \rangle - b_n \cdot 2^n$

Fall 1: $a_n = b_n = 0$ $\quad\quad$ $R_n = \{-2^n, \dots, 2^n - 1\}$

$$\begin{array}{cccccc}
 & 0 & a_{n-1} & \dots & a_0 & \\
 & 0 & b_{n-1} & \dots & b_0 & \\
 & c_{n-1} & & & c_{-1} & \\
\hline
(0) & s_n & s_{n-1} & & s_0 &
\end{array}
\implies \underline{s_n = c_{n-1}}$$

Fall 1.1: $c_{n-1} = 1 \iff \langle a' \rangle + \langle b' \rangle + c_{-1} \geq 2^n$

$\implies s_n = 1$

Zu i) Es gilt a) $a_n = b_n = 0$, aber $s_n = 1 \implies a_n = b_n \neq s_n$

b) $\underbrace{[a]}_{\langle a' \rangle - \underbrace{a_n}_{=0} 2^n} + \underbrace{[b]}_{\langle b' \rangle - \underbrace{b_n}_{=0} \cdot 2^n} + c_{-1} = \langle a' \rangle + \langle b' \rangle + c_{-1} \geq 2^n$

$\implies [a] + [b] + c_{-1} \notin R_n$

$\implies$ Für diesen Fall ist i) gezeigt.

Zu ii) Nichts zu zeigen, da wir gemäß Fallannahme 1.1. nicht im darstellbaren Bereich sind.

Fall 1.2 : $\underline{c_{n-1} = 0} \Longleftrightarrow \langle a' \rangle + \langle b' \rangle + c_{-1} < 2^n$

zu i) ~~a~~ a) $a_n = b_n = 0$, $\underline{s_n = 0}$

b) $[a] + [b] + c_{-1} = \langle a' \rangle + \langle b' \rangle + c_{-1} \underset{\text{nach Fallannahme}}{<} 2^n$

Außerdem $\langle a' \rangle + \langle b' \rangle + c_{-1} \geq 0$

$\Longrightarrow \quad 0 \leq \underbrace{[a] + [b] + c_{-1}}_{\in R_n} < 2^n$

$\Longrightarrow$ i) zu dem Fall gezeigt

zu ii) Aus Fallannahme folgt $[a] + [b] + c_{-1} \in R_n$, d.h.

z.z. $[a] + [b] + c_{-1} = [s]$

$[a] + [b] + c_{-1} = \langle a' \rangle + \langle b' \rangle + c_{-1}$

$= \langle \underset{\overset{\shortparallel}{0}}{c_{n-1}}\, s' \rangle = \langle \underset{\overset{\shortparallel}{s_n}}{0}\, s' \rangle = [\underset{\overset{\shortparallel}{s_n}}{0}\, s'] = [s_n s_{n-1} \ldots s_0]$

$= [s]$

2. Fall: $a_n = 1$, $b_n = 0$ . . .

3. Fall: $a_n = 0$, $b_n = 1$

(genau wie Fall 2)

4. Fall: $a_n = 1$, $b_n = 1$

Subtraktion

- Wegen $-[b] = [\overline{b}] + 1$ kann $[a] - [b]$ zurückgeführt werden auf $[a] + [\overline{b}] + 1$.

- **Beispiel:**

$$[a] = [0110] = 6_{10}, \quad [b] = [0111] = 7_{10}, \quad [\overline{b}] = [1000]$$

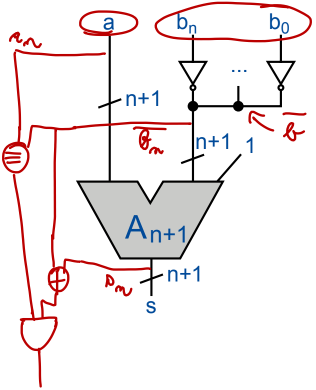
$[a] - [b]$

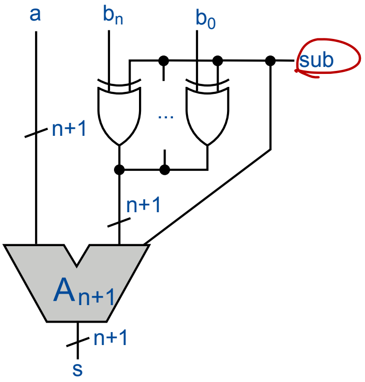
$$\begin{array}{r l} & 0\,1\,1\,0 \\ + & 1\,0\,0\,0 \\ + & 1 \\ \hline & 1\,1\,1\,1 \end{array} \qquad 1111 = (-1)_{10}$$

- Den Schaltkreis für Subtraktion gewinnt man aus einem Addierer.

- Kombinierter Addierer/Subtrahierer.

Subtrahierer





Überlauftest ergänzen!

$b_i \oplus 0 = b_i \qquad sub = 0 : [a] + [b] + 0$

$b_i \oplus 1 = \overline{b_i} \qquad sub = 1 : [a] + [\overline{b}] + 1 = [a] - [b]$

$b_i \oplus 1 = \overline{b_i}$