

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

Prof. Dr. Christoph Scholl

Institut für Informatik

WS 2015/16

- 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

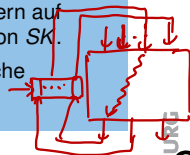
Die Kosten $C(SK)$ eines Schaltkreises SK sind durch die Anzahl seiner Gatter gegeben.

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

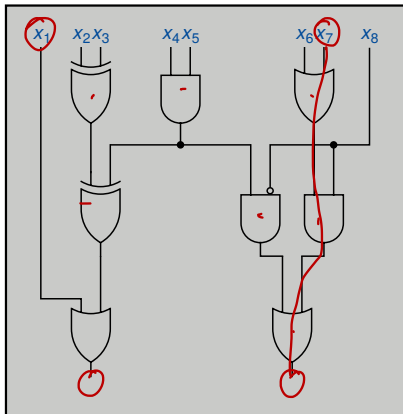
Definition

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



$$\underline{C(SK) = 8}$$

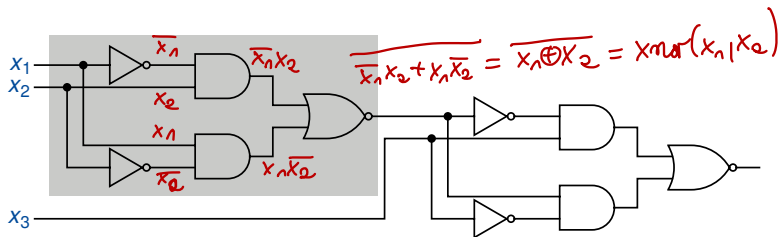
$$\underline{\underline{Depth(SK) = 3}}$$

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

z.B. $BIB = B_1 \vee B_2$

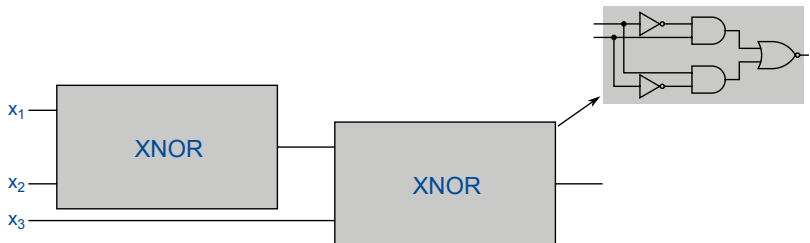
Teilschaltkreise, hierarchischer Entwurf (informell)

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

■ Binärdarstellung: $\langle a \rangle = \sum_{i=0}^{n-1} a_i 2^i$

■ Zweierkomplement: $\underline{[a_n a_{n-1} \dots a_0]} = \sum_{i=0}^{n-1} a_i 2^i - \underline{a_n 2^n}$

■ Rechenregel: $-[a] = [\bar{a}] + 1$

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

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:

$$+_n : \mathbb{B}^{2n+1} \rightarrow \mathbb{B}^{n+1},$$

$$+_n : (a_{n-1}, \dots, a_0, b_{n-1}, \dots, b_0, c) = (s_n, \dots, s_0)$$

$$\text{mit } \langle s \rangle = \langle s_n \dots s_0 \rangle = \langle a_{n-1} \dots a_0 \rangle + \langle b_{n-1} \dots b_0 \rangle + c$$

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

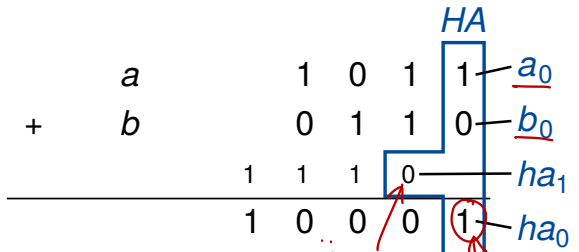
Addieren nach der Schulmethode (1/4)

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

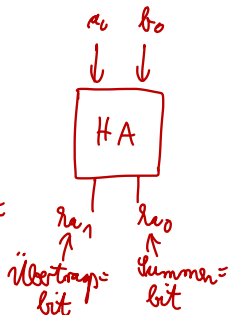
$$\begin{array}{r} 4378 \\ + 5613 \\ \hline 9991 \end{array}$$

$$\begin{array}{r} a \quad 1011 \rightarrow 11 \\ + b \quad 0110 \rightarrow 6 \\ \hline 1110 \\ \hline 10001 \rightarrow 17 \end{array}$$

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

$$ha : \mathbb{B}^2 \rightarrow \mathbb{B}^2 \text{ mit}$$

$$ha(a_0, b_0) = (ha_1, ha_0),$$

$$\text{wobei } 2ha_1 + ha_0 = a_0 + b_0.$$

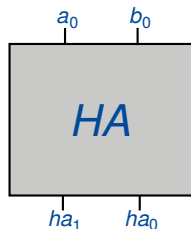
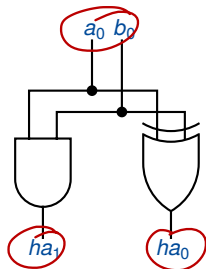
$$\langle ha_1, ha_0 \rangle = a_0 + b_0$$

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

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

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

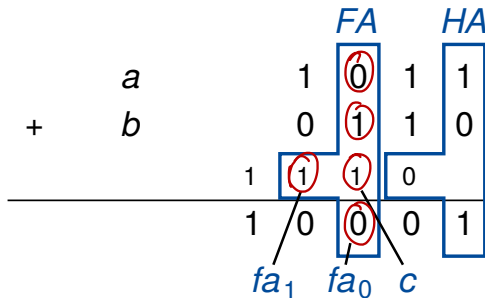
Schaltkreis eines Halbaddierers



dabei gilt:

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

Addieren nach der Schulmethode (3/4)



Volladdierer (Full Adder, FA)

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

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

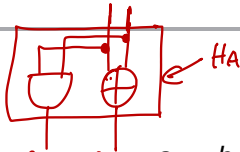
$\langle fa_1, fa_0 \rangle$

$$fa_0(a_0, b_0, c) = \text{xor}_3(a_0, b_0, c) = a_0 \oplus b_0 \oplus c = (a_0 \oplus b_0) \oplus c$$

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

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

Volladdierer als Funktion von HAs



Aus der Tabelle folgt:

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

$$= ha_0(c, ha_0(a_0, b_0))$$

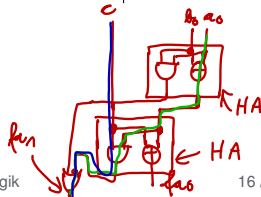
$$fa_1 = (a_0 \wedge b_0) \vee (c \wedge (a_0 \oplus b_0))$$

$$= ha_1(a_0, b_0) + ha_1(c, ha_0(a_0, b_0))$$

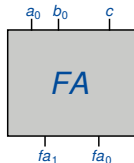
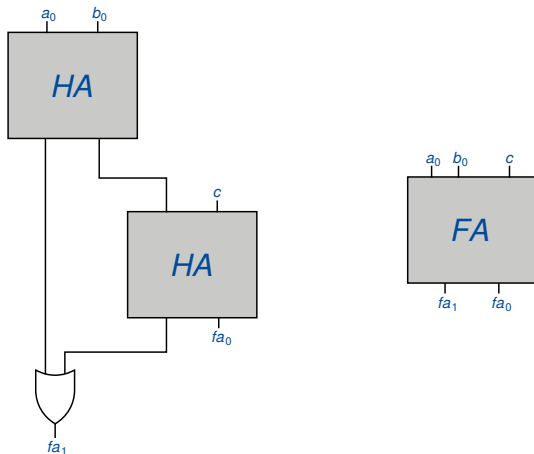
Kosten und Tiefe eines FA:

$$C(FA) = 5, \quad \text{depth}(FA) = \underline{3}$$

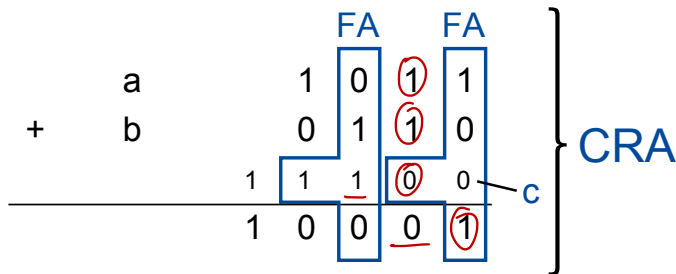
a_0	b_0	c	fa_1	fa_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Schaltkreis eines Volladdierers

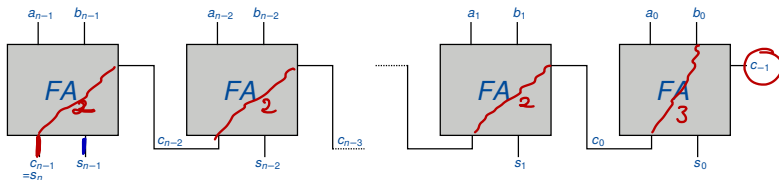


Addieren nach der Schulmethode (4/4)



Aufbau eines Carry-Ripple-Addierers

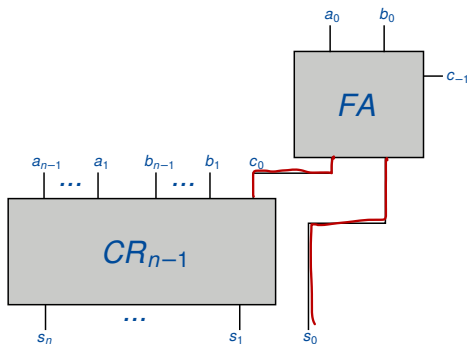
↑
Übertrag



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

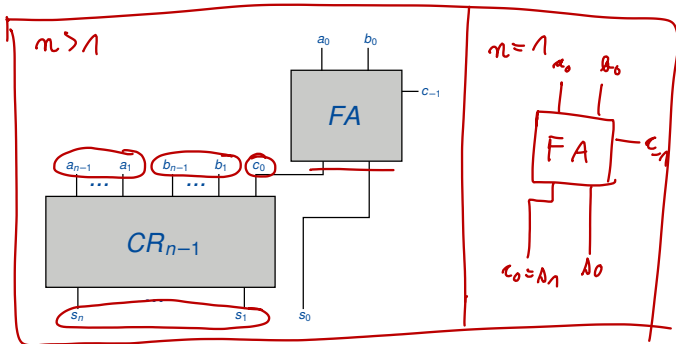
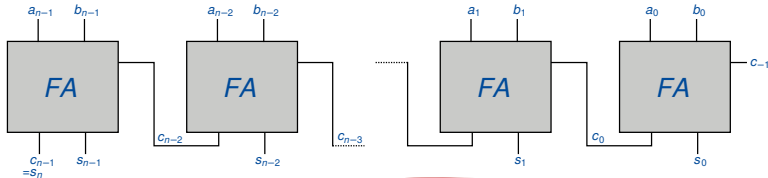
Induktive Definition des Carry-Ripple-Addierers CR

- 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.

$$\langle a_{n-1} \dots a_0 \rangle + \langle b_{n-1} \dots b_0 \rangle + c_{-1} = \langle d_n \dots d_0 \rangle$$

Beweis (durch Induktion):

- $n = 1$ ($CR_1 = FA$) ✓
- $n - 1 \rightarrow n$: Eingabe an CR_n : $\langle a_{n-1} \dots a_0, b_{n-1} \dots b_0, c_{-1} \rangle$
Zeige für Ausgabe $\langle s_n \dots s_0 \rangle$ von CR_n :
 $\langle s \rangle = \langle s_n \dots s_0 \rangle = \langle a_{n-1} \dots a_0 \rangle + \langle b_{n-1} \dots b_0 \rangle + c_{-1}$.

- Nach Induktionsvoraussetzung gilt für CR_{n-1} :

$$\langle s_n \dots s_1 \rangle = \langle a_{n-1} \dots a_1 \rangle + \langle b_{n-1} \dots b_1 \rangle + c_0 \quad (a) \leftarrow$$

Wegen FA-Eigenschaft gilt $\langle c_0, s_0 \rangle = a_0 + b_0 + c_{-1} \quad (b)$

- Insgesamt: $\langle s_n \dots s_0 \rangle = 2 \cdot \langle s_n \dots s_1 \rangle + s_0$

$$\stackrel{(a)}{=} 2 \cdot (\langle a_{n-1} \dots a_1 \rangle + \langle b_{n-1} \dots b_1 \rangle + c_0) + s_0$$

$$= 2 \cdot (\langle a_{n-1} \dots a_1 \rangle + \langle b_{n-1} \dots b_1 \rangle) + 2 \cdot c_0 + s_0$$

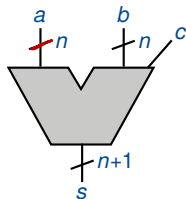
$$\stackrel{(b)}{=} 2 \cdot \langle a_{n-1} \dots a_1 \rangle + a_0 + 2 \cdot \langle b_{n-1} \dots b_1 \rangle + b_0 + c_{-1}$$

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

$$\begin{aligned} \langle d_n \dots d_0 \rangle &= \sum_{i=0}^n d_i \cdot 2^i = \sum_{i=n}^n d_i \cdot 2^i + \sum_{i=0}^{n-1} d_i \cdot 2^i \\ &= \sum_{i=0}^{n-1} d_{i+1} \cdot 2^{i+1} + d_0 \\ &= 2 \cdot \sum_{i=0}^{n-1} d_{i+1} \cdot 2^i + d_0 \\ &= 2 \cdot \langle d_n \dots d_1 \rangle + d_0 \end{aligned}$$

Schaltbild und Komplexität von CR

- $C(CR_n) = n \cdot C(FA) = \underline{5n}$.
- $depth(CR_n) = \underline{3 + 2(n - 1)}$.
- 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.



Definition

Ein n -Bit-Inkrementer INC_n berechnet die Funktion

$$inc_n: \mathbb{B}^{n+1} \rightarrow \mathbb{B}^{n+1}$$

$$inc_n(\underline{a_{n-1}, \dots, a_0}, \underline{c}) = (\underline{s_n, \dots, s_0}) \text{ mit } \langle \underline{s_n \dots s_0} \rangle = \langle \underline{a} \rangle + \underline{c}$$

- Ein Inkrementer ist ein Addierer mit $b_i = 0$ für alle i .
 \Rightarrow Ersetze in CR_n die FA durch HA .
- Kosten und Tiefe:
 - $C(INC_n) = \underline{n \cdot C(HA)} = \underline{2n}$
 - $depth(INC_n) = \underline{n \cdot depth(HA)} = \underline{n}$

Definition

Ein n -Bit-Multiplexer MUX_n berechnet die Funktion

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

select-bit
↓

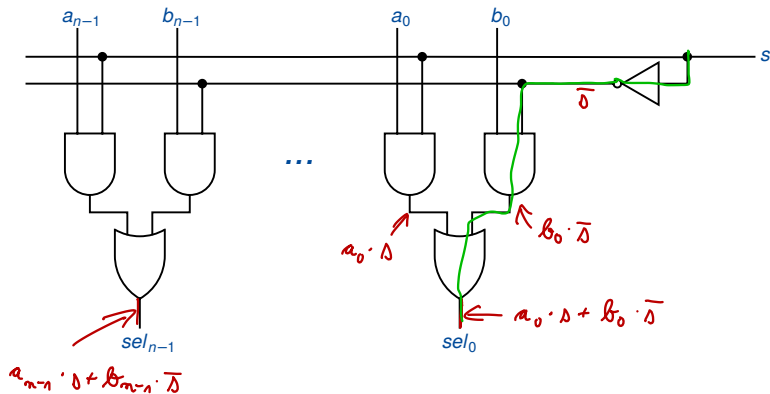
$$sel_n(\underline{a_{n-1}, \dots, a_0}, \underline{b_{n-1}, \dots, b_0}, \underline{s}) = \begin{cases} \underline{(a_{n-1} \dots a_0)}, & \text{falls } \underline{s = 1} \\ \underline{(b_{n-1} \dots b_0)}, & \text{falls } \underline{s = 0} \end{cases}$$

■ Es gilt: $(sel_n)_i = \underline{s \cdot a_i + \bar{s} \cdot b_i}$

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

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

Aufbau von MUX_n

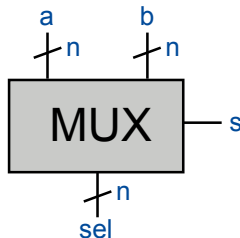


Schaltbild und Kosten MUX_n

Kosten und Tiefe:

$$C(MUX_n) = \underline{3n+1}.$$

$$depth(MUX_n) = 3.$$



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:

$$C(+_n) \geq 2 \cdot n, \quad \underline{depth(+_n) \geq \log(n) + 1}$$

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)

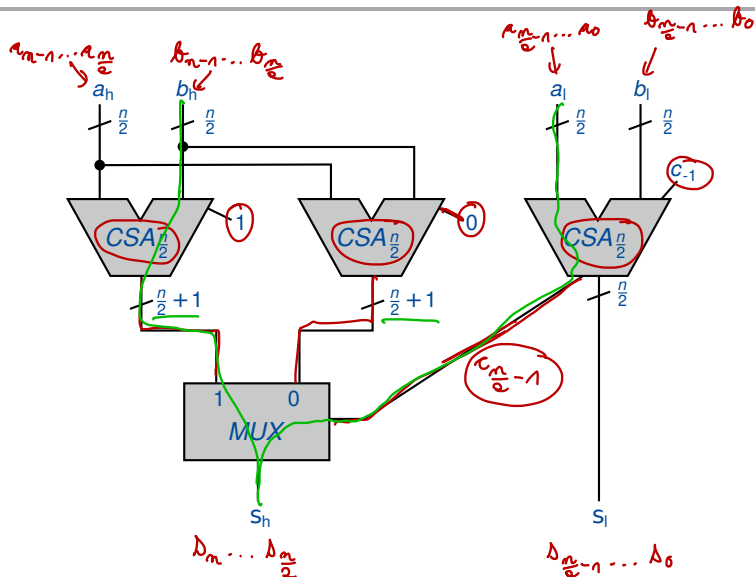
- Idee: Nutze **Parallelverarbeitung**, um Tiefe zu reduzieren!

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

The term $c_{\frac{n}{2}-1}$ in the third row is circled in red.

- $CSA_1 = \underline{FA}$.
- 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:

■ $n = 1$: $\text{depth}(CSA_1) = \text{depth}(FA) = 3$.

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

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

$$\leq \text{depth}(CSA_{\frac{n}{4}}) + 3 + 3$$

$$\leq \text{depth}(CSA_{\frac{n}{8}}) + 3 + 3 + 3$$

...

$$\leq \text{depth}(CSA_{\frac{n}{2^k}}) + k \cdot 3$$

$$= \text{depth}(CSA_1) + k \cdot 3 = 3 + k \cdot 3$$

$$\leq 3 \cdot (k + 1) = 3\log(n) + 3.$$

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

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

Komplexität von CSA_n : Kosten

Satz

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

Beweis: ...

$$\underline{C(CSA_m) = 3 \cdot C(CSA_{m/2}) + \frac{3}{2} \cdot m + 4}$$

$$C(CSA_n) = C(FA) = 5$$

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

$$m = 2^k$$

$$= 3 \cdot C(CSA_{m/2}) + \frac{3}{2}m + 4$$

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

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

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

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

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

$$= 3^4 \cdot C(\text{CSA}_{\frac{n}{16}}) + \underline{\frac{3}{2}n + 4} + \underline{3 \cdot (\frac{3}{2} \cdot \frac{n}{2} + 4)} + \underline{3^2 \cdot (\frac{3}{2} \cdot \frac{n}{4} + 4)} + \underline{3^3 \cdot (\frac{3}{2} \cdot \frac{n}{8} + 4)}$$

⋮

$$= 3^k C(\text{CSA}_{\frac{n}{2^k}}) + \sum_{i=0}^{k-1} 3^i (\frac{3}{2} \cdot \frac{n}{2^i} + 4) \quad (\text{Beweis durch Induktion über } k)$$

$$\underbrace{= C(\text{CSA}_n)}_{n=2^2} = 5$$

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

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

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

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

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

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

geometrische Summenformel:

$$\sum_{i=0}^k 2^i = \frac{2^{k+1} - 1}{2 - 1} \quad \text{für } 2 \neq 1$$

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

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

- 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).
 - $C(CLA_n) \leq \underline{11n}$,
 - $\underline{depth(CLA_n) \leq 4 \cdot \log(n) + 2}$.

Addition von Zweierkomplementzahlen

$$\begin{aligned}\underline{[a_m \dots a_0]_2} &= \sum_{i=0}^{n-1} a_i \cdot 2^i - a_n \cdot 2^n \\ \underline{[b_m \dots b_0]_2} &= \sum_{i=0}^{n-1} b_i \cdot 2^i - b_n \cdot 2^n\end{aligned}$$

- Auszurechnen ist:

$$\underline{[a_n a_{n-1} \dots a_0]_2} + \underline{[b_n b_{n-1} \dots b_0]_2} = (-a_n 2^n) + (-b_n 2^n) + \sum_{i=0}^{n-1} a_i 2^i + \sum_{i=0}^{n-1} b_i 2^i$$

- 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

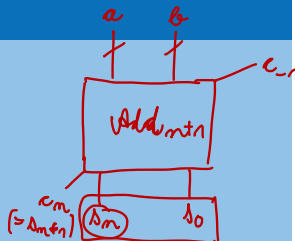
Satz

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:

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

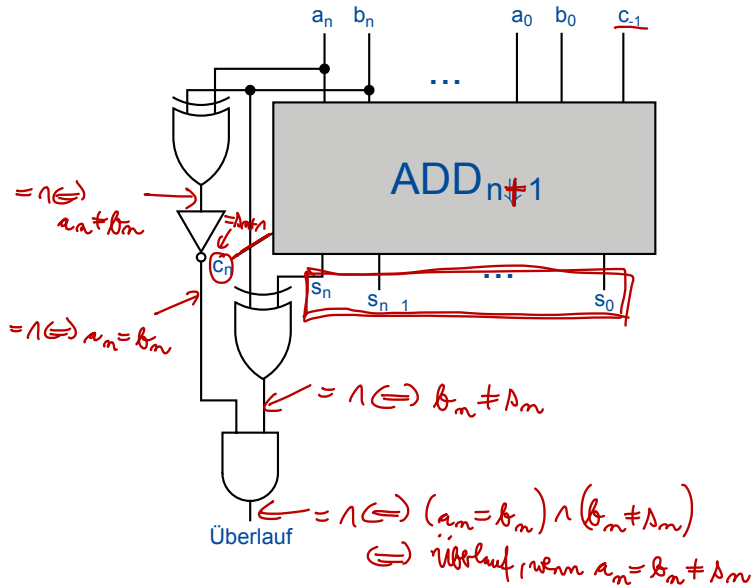
(ii) $[a] + [b] + c_{-1} \in R_n \Rightarrow [a] + [b] + c_{-1} = [s]$



- Beweis durch Fallunterscheidung ($[a], [b]$ beide positiv, beide negativ, $[a]$ positiv / $[b]$ negativ) und Nachrechnen (~~Induktion~~).
- Man kann einen alternativen Überlauftest zeigen:

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

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



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

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

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

$$\begin{array}{r} 0001 \triangleq 1 \\ 0011 \triangleq 3 \\ \hline (0)0100 \triangleq 4 \end{array}$$

↑
kein Überlauf

$$\begin{array}{r} 1010 \triangleq -6 \\ + 0110 \triangleq 6 \\ \hline (1)0000 \triangleq 0 \end{array}$$

↑
kein Überlauf

$$\begin{array}{r} 1100 \triangleq -4 \\ + 1011 \triangleq -5 \\ \hline (1)0111 \end{array}$$

↑
Überlauf

$$\begin{array}{r} 0100 \triangleq 4 \\ 0101 \triangleq 5 \\ \hline (0)1001 \triangleq 9 \end{array}$$

↑
Überlauf

$$\begin{array}{r} 0110 \triangleq 6 \\ + 1101 \triangleq -3 \\ \hline (1)0011 \triangleq 3 \end{array}$$

↑
kein Überlauf

$$\begin{array}{r} 1111 \triangleq -1 \\ 1111 \triangleq -3 \\ \hline (1)1100 \triangleq -4 \end{array}$$

↑
kein Überlauf



Beweis:

$$\begin{array}{ccccccc}
 & & \overbrace{a_m} & a_{m-1} & \dots & a_0 & \\
 & & b_m & b_{m-1} & \dots & b_0 & \\
 c_m & \boxed{c_{m-1}} & c_{m-2} & \dots & c_0 & c_{-1} & \\
 \hline
 c_m & \boxed{a_m} & a_{m-1} & \dots & a_1 & a_0 & \\
 \hline
 \end{array}$$

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

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

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

$$\text{Dann gilt: } \langle a' \rangle + \langle b' \rangle + c_{-1} = \langle c_{m-1} a' \rangle$$

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

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

$$[b] = \langle b' \rangle - b_m \cdot 2^m$$

Fall 1: $a_m = b_m = 0$

$$R_m = \{-2^m, \dots, 2^m - 1\}$$

$$0 a_{m-1} \dots a_0$$

$$0 b_{m-1} \dots b_0$$

$$\begin{array}{r} c_{m-1} \\ \hline (0) \quad a_m \quad a_{m-1} \quad \dots \quad a_0 \end{array} \quad \Rightarrow \quad \underline{a_m = c_{m-1}}$$

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

$$\Rightarrow a_m = 1$$

zu i) Es gilt a) $a_m = b_m = 0$, aber $a_m = 1 \Rightarrow a_m = b_m \neq a_m$

$$b) \underbrace{[a]} + \underbrace{[b]} + c_{-1} = \langle a' \rangle + \langle b' \rangle + c_{-1} \geq 2^m$$

$$\begin{array}{ccc} \langle a' \rangle - a_m 2^m & \leftarrow & \langle b' \rangle - b_m 2^m \\ \parallel & & \parallel \\ 0 & & 0 \end{array}$$

$$\Rightarrow [a] + [b] + c_{-1} \notin R_m$$

\Rightarrow Für diesen Fall ist i) gezeigt.

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

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

zu ii) ~~da~~ a) $c_m = b_m = 0, \underline{d_m = 0}$

$$b) [a] + [b] + c_{-n} = \langle a' \rangle + \langle b' \rangle + c_{-n} < 2^m$$

↑
nach Fallannahme

$$\begin{aligned} &\text{Außerdem } \langle a' \rangle + \langle b' \rangle + c_{-n} \geq 0 \\ \Rightarrow &0 \leq \underbrace{[a] + [b] + c_{-n}}_{\in R_m} < 2^m \end{aligned}$$

\Rightarrow i) zu den Fall gezeigt

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

$$\text{z.z. } [a] + [b] + c_{-n} = [d]$$

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

$$= \underbrace{\langle c_{m-n} d' \rangle}_{= 0} = \underbrace{\langle 0 d' \rangle}_{d_m} = \underbrace{[0 d']}_{d_m} = [d_m d_{m-1} \dots d_0] = [d]$$

2. Fall: $a_m = 1, b_m = 0$...

3. Fall: $a_m = 0, b_m = 1$

Genaue wie Fall 2)

4. Fall: $a_m = 1, b_m = 1$

-

