Technische Informatik 1

Prof. Dr. Rolf Drechsler

Christina Plump

Überblick

Teil 1: Der Rechneraufbau (Kapitel 2-5)

- Rechner im Überblick
- Pipelining
- Speicher
- Parallelverarbeitung

Teil 2: Der Funktionalitätsaufbau (Kapitel 6-12)

- Kodierung
- Grundbegriffe, Boolesche Funktionen
- Darstellungsmöglichkeiten
- **Schaltkreise, Synthese, spezielle Schaltkreise**
  - Addierer, Inkrementer
  - **Subtrahierer, Multiplizierer, ALU**

Kapitel 11: Arithmetische Schaltkreise

Addierer, Multiplexer und Inkrementer

Subtrahierer, Multiplizierer und ALU

Subtrahierer im Allgemeinen

Gegeben:

Zwei Binärzahlen $a = a_{n-1} \dots a_0$, $b = b_{n-1} \dots b_0$

Gesucht:

Schaltkreis, der Binärdarstellung s mit $< s > = < a > - < b >$ berechnet.

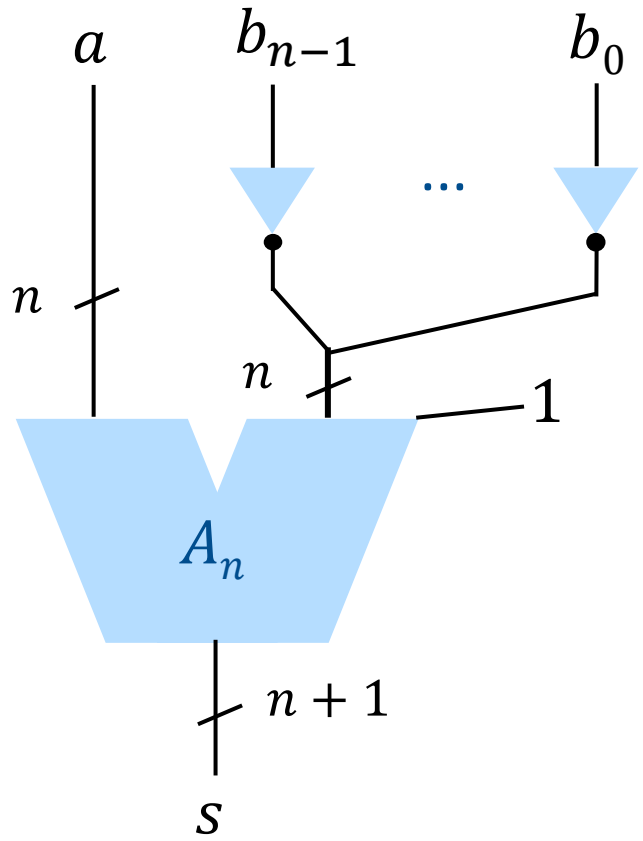
Bemerkung:

- Wegen $-[b]_2 = \overline{[b]} + 1$, gilt $[a]_2 - [b]_2 = [a]_2 + \overline{[b]}_2 + 1$.
- Rückführung eines Subtrahierers auf einen Addierer
- Bau kombinierter Addierer/Subtrahierer Schaltkreise

Bespiel und Schaltbild eines Subtrahierers

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

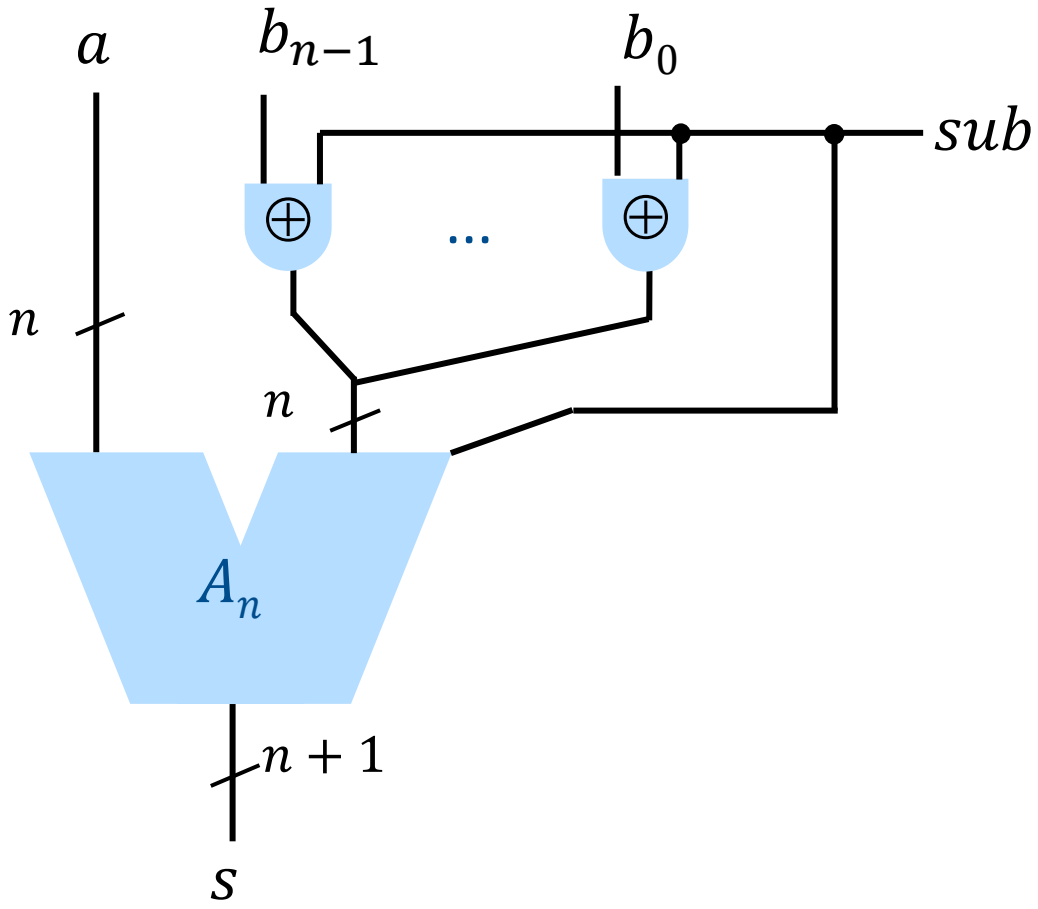
$$\begin{array}{rl} 0100 & 6_{10} \\ 1000 & \\ 1 & -7_{10} \\ \hline 1101 & -1_{10} \end{array}$$



Schaltbild für einen Addierer/Subtrahierer

- Erinnerung:
  - $b_i \oplus 0 = b_i$
  - $b_i \oplus 1 = \overline{b_i}$

- Verhalten für Belegungen von sub:
  - $sub = 0$: $[a]_2 + [b]_2 + 0 = [a]_2 + [b]_2$
  - $sub = 1$: $[a]_2 + [\overline{b}]_2 + 1 = [a]_2 - [b]_2$



Multiplizierer im Allgemeinen

Gegeben:

- Zwei Binärzahlen $a = a_{n-1} \dots a_0$, $b = b_{n-1} \dots b_0$

Gesucht:

Schaltkreis, der Binärdarstellung s mit $< s > = < a > \cdot < b >$ berechnet.

Bemerkung:

- Wegen $< a > \cdot < b > \leq (2^{n-1}-1) \cdot (2^{n-1} - 1) = 2^{2n} - 2^{n+1} + 1 \leq 2^{2n} - 1$, genügen $2n$ Bits für die Darstellung von s, d.h. als Ausgang des Schaltkreises.
- Da sich die Multiplikation vorzeichenbehafteter Binärzahlen auf die Multiplikation positiver Binärzahlen zurückführen lässt (Vorzeichen wird getrennt berechnet), reicht die Betrachtung positiver Binärzahlen aus.

Multiplizierer im Allgemeinen (2)

Definition (n-Bit Multiplizierer):
Ein n-Bit Multiplizierer ist ein Schaltkreis, der die folgende Boolesche Funktion $\cdot_n$ berechnet:

$$\cdot_n : \mathbb{B}^{2n} \to \mathbb{B}^{2n}$$
$$(a_{n-1}, \dots, a_0, b_{n-1}, \dots, b_0) \mapsto (p_{2n-1}, \dots, p_0),$$

sodass $< p >=< a > \cdot < b >$ gilt.

Beispiel:

$$6_{10} \cdot 5_{10} = (110)_2 \cdot (101)_2 = (11110)_2 = 30_{10}$$

1	1	0	·	1	0	1
		1	1	0		
			0	0	0	
				1	1	0
		1	1	1	1	0

Berechnung der Partialprodukte (Multiplikationsmatrix)

- Multiplikationsmatrix beinhaltet Partialprodukte
- Realisierung der Multiplikationsmatrix mit n^2 AND-Gattern und n^2 Konstanten 0.

$$\begin{pmatrix} pp_0 \\ pp_1 \\ \vdots \\ pp_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & a_{n-1}b_0 & a_{n-2}b_0 & \dots & a_1b_0 & a_0b_0 \\ 0 & 0 & \dots & 0 & a_{n-1}b_1 & a_{n-2}b_1 & a_{n-3}b_1 & \dots & a_0b_1 & 0 \\ \vdots & & & & \vdots & \vdots & \vdots & & & \vdots \\ 0 & a_{n-1}b_{n-1} & \dots & a_2b_{n-1} & a_1b_{n-1} & a_0b_{n-1} & 0 & & 0 & 0 \end{pmatrix}$$

Notwendig zur Berechnung der Multiplikation

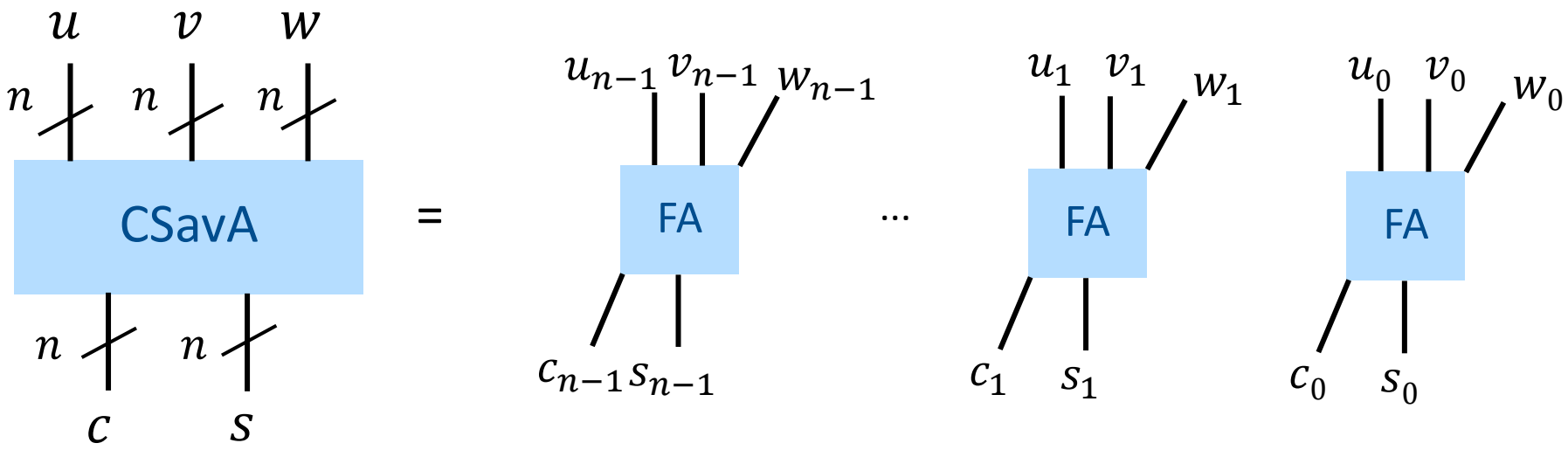
- Schnelle Addition von n Partialprodukten der Länge $2n$
- Mit CLAs lösbar mit
  - Kosten: $\mathcal{O}(n^2)$
  - Tiefe: Unterscheide die Art der Aufsummierung:
    - Lineares Aufsummieren: $\mathcal{O}(n \log n)$
    - Baumartiges Aufsummieren: $\mathcal{O}(\log^2 n)$

Schnelleres Summieren: Carry-Save Addierer

- Verbesserbar durch Carry-Save Addierer
  - Reduziert drei Eingabewerte zu zwei Ausgabewerten
  - Lösbar durch Hintereinandersetzen von Volladdierern, d.h. keine Carry Chain

$$< u > + < v > + < w > = < c > + < s >$$

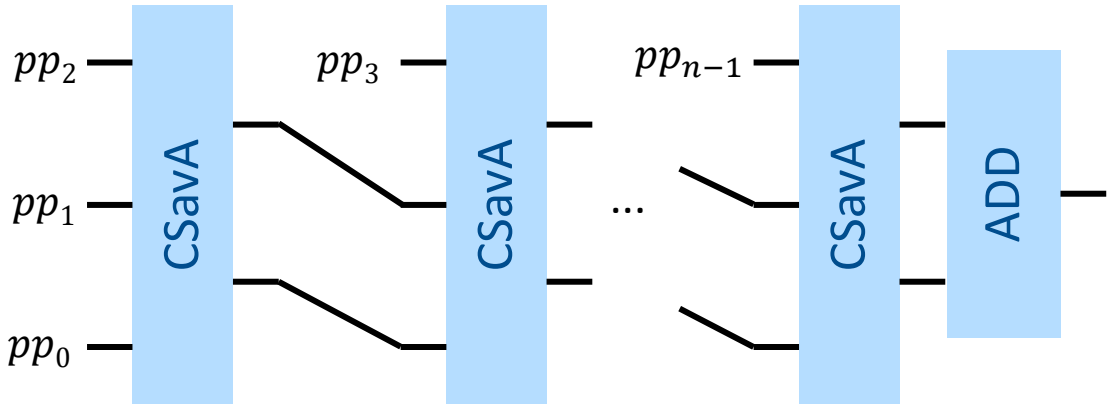
$$\begin{array}{cccccccc} & u_{n-1} & u_{n-2} & \dots & u_2 & u_1 & u_0 \\ & v_{n-1} & v_{n-2} & \dots & v_2 & v_1 & v_0 \\ & w_{n-1} & w_{n-2} & \dots & w_2 & w_1 & w_0 \\ \hline c_{n-1} & c_{n-2} & c_{n-3} & \dots & c_1 & c_0 & 0 \\ 0 & s_{n-1} & s_{n-2} & \dots & s_2 & s_1 & s_0 \end{array}$$



Nutzung für Partialprodukte!

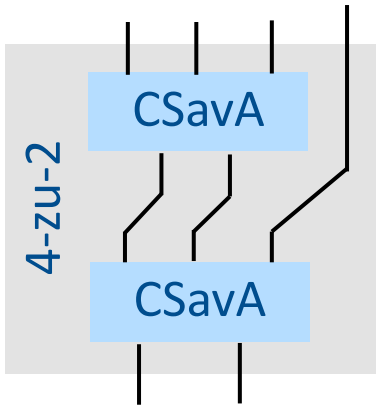
1. Serielle Lösung:

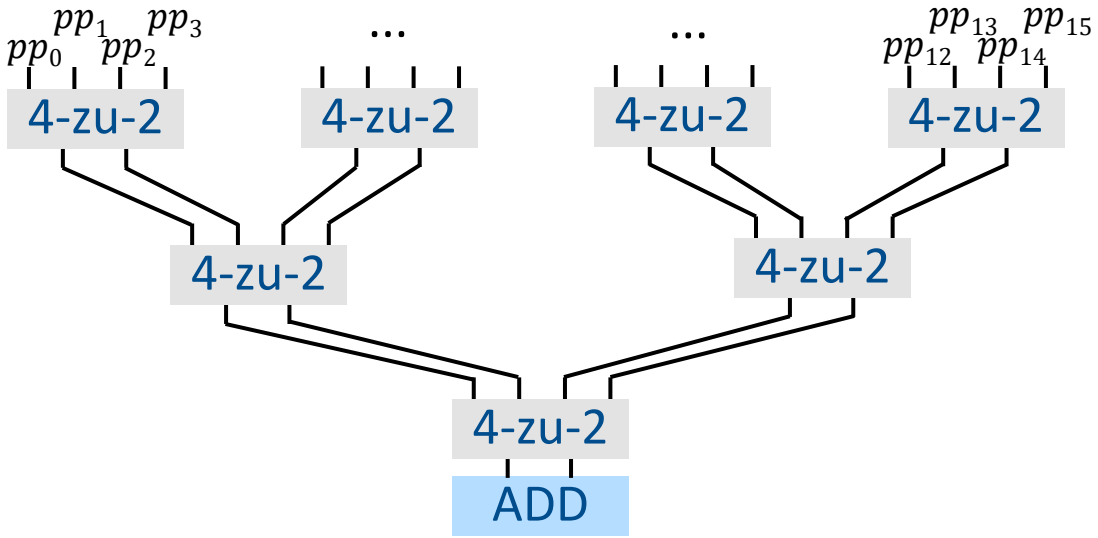
- Hintereinanderschalten von $(n-2)$ CSavA-Addierern der Länge $2n$
  - Fasse n Partialprodukte zu zwei $2n$-Bit-Worten zusammen
  - Addiere die $2n$-Bit-Worte mit CLA
- **Kosten:** $\mathcal{O}(n^2)$
- **Tiefe:** $\mathcal{O}(n)$



2. Baumartige Lösung:

- Neue Grundzelle (4-zu-2) zur Reduktion von vier $2n$-Bit Eingabeworten zu zwei Ausgabeworten, bestehend aus zwei CSavA
- Baumartiges Zusammenfassen der Partialprodukte mit 4-zu-2-Bausteinen zu zwei $2n$-Bit-Worten
- Addiere die $2n$-Bit-Worte mit CLA
- **Kosten:** $\mathcal{O}(n^2)$
- **Tiefe:** $\mathcal{O}(log\ n)$





Aufbau einer ALU

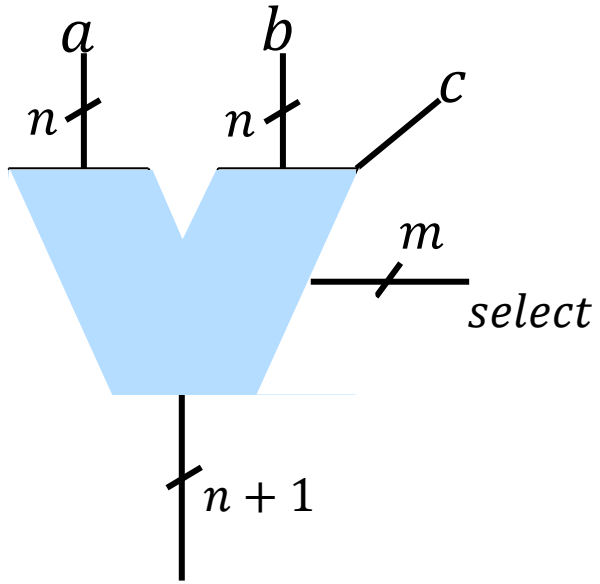
Definition (n-Bit ALU):
Eine n-bit Arithmetic Logic Unit (ALU) ist ein Schaltkreis zur Berechnung von arithmetischen und logischen Funktionen mit

- zwei n-Bit Operanden a, b,
- einem Eingangsübertrag c
- einem m-Bit select Eingang, der die auszuführende Funktion auswählt
- einem $n + 1$ -Bit Ausgang s

Aufbau einer ALU (2)

Beispiel für ALU mit acht Funktionen: 3-bit *select*-Eingang

Funktionsnr.			ALU-Funktion
s_2	s_1	s_0	
0	0	0	$(0, \dots, 0)$
0	0	1	$[b]_2 - [a]_2$
0	1	0	$[a]_2 - [b]_2$
0	1	1	$[a] + [b] + c$
1	0	0	$a \oplus b = (a_{n-1} \oplus b_{n-1}, \dots, a_0 \oplus b_0)$
1	0	1	$a + b = (a_{n-1} + b_{n-1}, \dots, a_0 + b_0)$
1	1	0	$a \cdot b = (a_{n-1} \cdot b_{n-1}, \dots, a_0 \cdot b_0)$
1	1	1	$(1, \dots, 1)$



Mögliche Realisierungen einer ALU

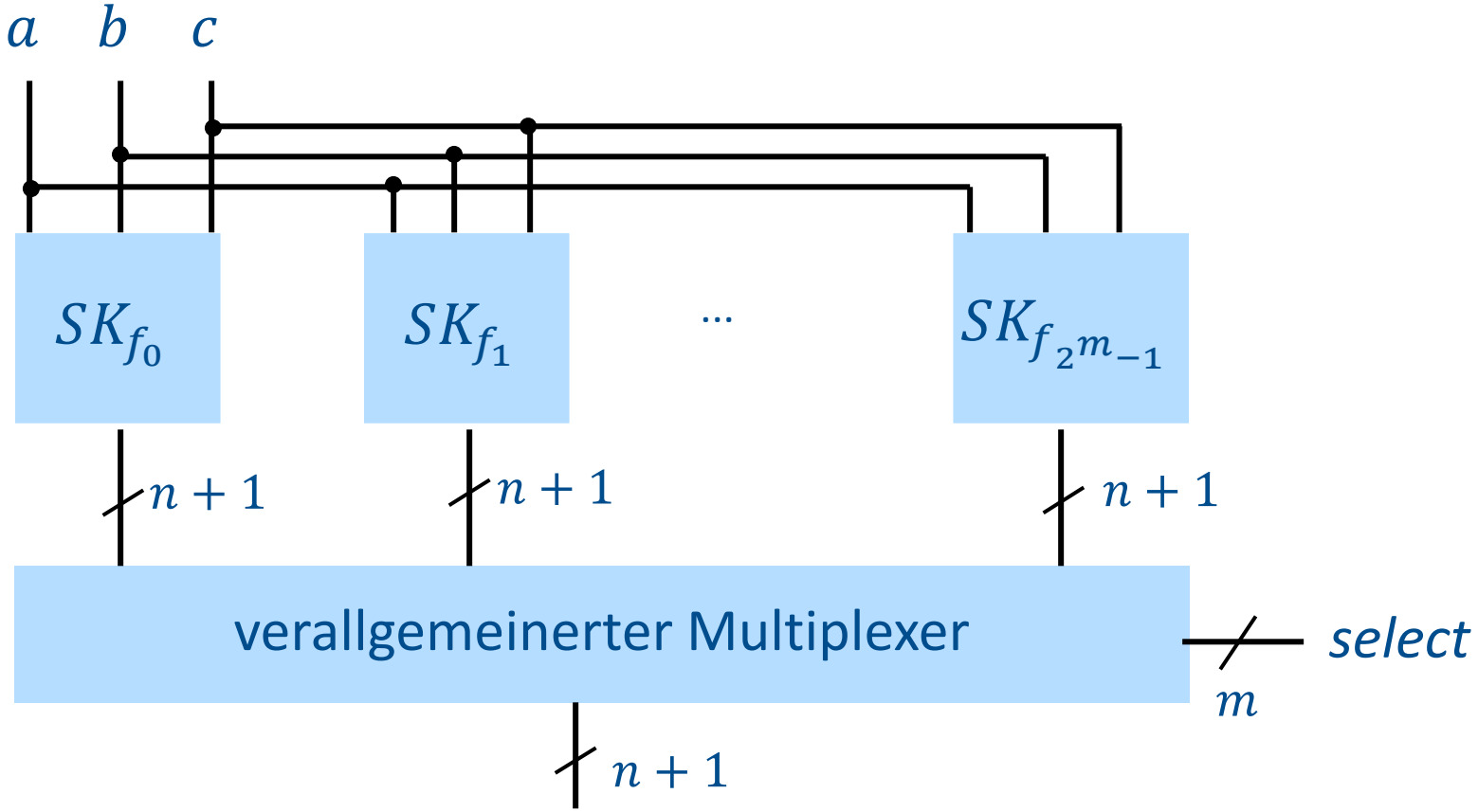
Getrennte Schaltkreise

- Realisiere $2^m - 1$ unterschiedliche Funktionen $f_0, \ldots, f_{2^m-1}$ durch getrennte Schaltkreise SK_i
- Wähle Funktion durch verallgemeinerten Multiplexer (siehe nächste Folie)

Gemeinsamer Schaltkreis

- Entwirf Schaltkreis mit gemeinsamer Behandlung ähnlicher Funktionen
- Beispiel auf übernächster Folie

Realisierung der *n*-Bit-ALU (Einzelfunktionen getrennt)



Schaltungsrealisierung der *n*-Bit-ALU mit Überlappung von Einzelfunktionen