Exkurs – Multiplizierer

Albert-Ludwigs-Universität Freiburg

Dr. Tobias Schubert, Dr. Ralf Wimmer

Professur für Rechnerarchitektur

WS 2016/17

Multiplizierer

- **Gesucht**: Schaltkreis zur Multiplikation zweier Binärzahlen $< a_{n-1}, \ldots, a_0 >, < b_{n-1}, \ldots, b_0 >$.

- **Beispiel**:

$$\underbrace{(110)}_{6_{10}} \cdot \underbrace{(101)}_{5_{10}}$$

$$\begin{array}{cccccc}
 & & & 1 & 1 & 0 \\
+ & & 0 & 0 & 0 & \\
+ & 1 & 1 & 0 & & \\
\hline
 & 1 & 1 & 1 & 1 & 0
\end{array}$$

$$= 30_{10}$$

Allgemeines zum Multiplizierer

- Wieviele Stellen werden für das Ergebnis benötigt?

$$
\begin{aligned}
< a > \cdot < b > \quad &\leq \quad (2^n - 1) \cdot (2^n - 1) \\
= \quad 2^{2n} - 2^{n+1} + 1 \quad &\leq \quad 2^{2n} - 1
\end{aligned}
$$

- **Also**:
  2n Stellen zur Multiplikation von Binärzahlen.

Vorgehen bei der Multiplikation

- Multipliziere die Beträge der Zahlen.
- Bestimme das Vorzeichen des Produkts.
- Setze das Endergebnis zusammen.

n-Bit-Multiplizierer

Definition

Ein n-Bit-Multiplizierer ist ein Schaltkreis, der die folgende Funtkion berechnet:

$mul_n : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$
$mul_n(a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0) = (p_{2n-1}, \ldots, p_0)$ mit
$< p_{2n-1}, \ldots, p_0 > = < a > \cdot < b >$

$$< a > \cdot < b > = < a > \cdot \sum_{i=0}^{n-1} b_i \cdot 2^i = \sum_{i=0}^{n-1} (< a > \cdot b_i \cdot 2^i)$$

UNI FREIBURG

Die Multiplikationsmatrix

$$\begin{pmatrix} pp_0 \\ pp_1 \\ \vdots \\ pp_{n-1} \end{pmatrix} =$$

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

- Realisierung der Multiplikationsmatrix mit n^2 AND-Gattern (und n^2 Konstanten 0).

Daraus entstehende Aufgabe:

- Schnelle Addition von n Partialprodukten der Länge $2n$.
- Mit Carry-Lookahead-Addierern (CLAs) lösbar mit Kosten $O(n^2)$, Tiefe $O(n\, log(n))$ bei *linearem Aufsummieren* der Partialprodukte $((((pp_0 + pp_1) + pp_2) + \ldots) + pp_{n-1})$,

  Tiefe $O(log^2(n))$ bei *baumartigem Zusammenfassen* der Partialprodukte.

UNI FREIBURG

Verbesserung

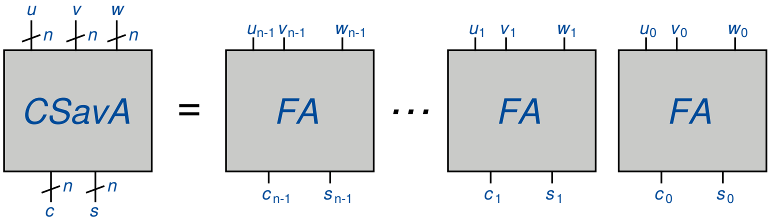
- Verwende Carry-Save-Addierer.

- Reduktion von 3 Eingabeworten u, v, w zu zwei Ausgabeworten s, c mit $< u > + < v > + < w > = < s > + < c >$.

	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
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

- Gelöst durch Nebeneinandersetzen von Volladdierern (keine Carry-Chain!)

Carry-Save-Addierer (CSavA)



Bemerkung zum Aufbau des CSavA

- Speziell bei Partialprodukten:
  Reduziere 3 2n-Bit-Zahlen zu 2 2n-Bit-Zahlen.

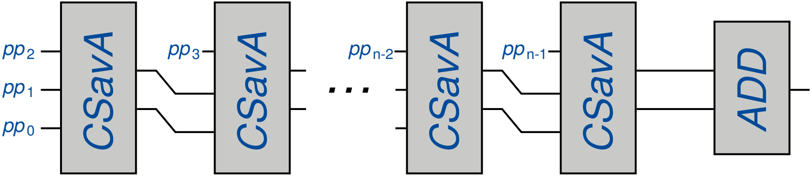
$$
\begin{array}{cccc}
pp_{0,2n-1} & \cdots & pp_{0,1} & pp_{0,0} \\
pp_{1,2n-1} & \cdots & pp_{1,1} & pp_{1,0} \\
pp_{2,2n-1} & \cdots & pp_{2,1} & pp_{2,0} \\
\hline
c_{2n-2} & \cdots & c_0 & 0 \\
s_{2n-1} & \cdots & s_1 & s_0
\end{array}
$$

- ($c_{2n-1} = 0$: Carry-Ausgang des letzten FA nicht verwendet.)

1. Serielle Lösung

- Hintereinanderschalten von n-2 CSavA-Addierern der Länge 2n.
  - Fasse *n* Partialprodukte zu 2 2n-Bit-Worten zusammen.
- Addiere die 2n-Bit-Worte mit CLA.
- *siehe Abb. einer Addierstufe*
- Kosten $O(n^2)$, Tiefe $O(n)$

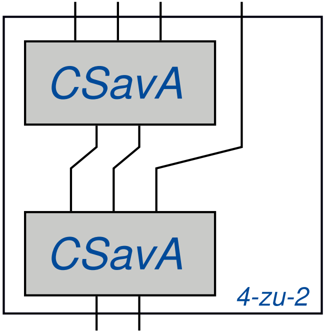
Addierstufe im Multiplizierer



2. Baumartige Lösung

- Neue Grundzelle zur Reduktion von 4 2n-Bit-Eingabeworten zu zwei Ausgabeworten, bestehend aus 2 CSavA-Addierern (*siehe Abb. zur Reduktionszelle*).
- Baumartiges Zusammenfassen der Partialprodukte mit 4-zu-2-Bausteinen zu 2 2n-Bit Worten.
- Addiere die 2n-Bit-Worte mit CLA.
- *siehe Abb. der Addierstufe mit log. Zeit*
- Kosten $O(n^2)$, Tiefe $O(logn)$

4-zu-2 Reduktions-Grundzelle



Addierstufe des log-Zeit-Multiplizierers für 16 Bit