Prof. Dr. Christoph Scholl
Tobias Seufert

Freiburg, 21. Juni 2023

Technische Informatik
Musterlösung zu Übungsblatt 9

Hinweis: Auf diesem Blatt befindet sich eine „Bonusaufgabe“. Diese Aufgabe zählt nicht in die Gesamtheit der Aufgaben, bei sinnvoller Bearbeitung wird sie jedoch zur Menge der sinnvoll bearbeiteten Aufgaben gerechnet.

Aufgabe 1 (3 + 1 + 2 Punkte)

a) Zeichnen Sie den 4-Bit-Carry-Ripple-Addierer CR_4 über der Bibliothek $BIB = \mathbb{B}_1 \cup \mathbb{B}_2$. Verwenden Sie dabei keine hierarchischen Teilschaltkreise.

b) Kennzeichnen Sie den längsten Pfad in ihrem Schaltkreis und geben Sie dessen Tiefe an.

c) Bestimmen Sie für jedes Gatter des ermittelten Schaltkreises den Wert des Gatterausganges für die Belegung
$b_3 = 1$, $b_2 = 1$, $b_1 = 0$, $b_0 = 1$, $a_3 = 1$, $a_2 = 0$, $a_1 = 0$, $a_0 = 1$, $c_{-1} = 0$.

Lösung:

Siehe Abbildung 1.

Allgemein gilt $depth(CR_n) = 3 + 2(n - 1)$. Hier: $depth(CR_4) = 3 + 2(4 - 1) = 9$. Siehe auch Pfad.

Aufgabe 2 (3 + 3 Punkte)

Ein n-Bit Inkrementer INC_n berechnet die Funktion $inc_n : \mathbb{B}_{n+1} \mapsto \mathbb{B}_{n+1}$, $inc_n(a_{n-1}, \dots, a_0, c) = (s_n \dots, s_0)$ mit $\langle s_n \dots s_0 \rangle = \langle a \rangle + c$. In der Vorlesung wurde vorgestellt, wie man einen n-Bit Inkrementer nach dem beim n-Bit Carry-Ripple-Addierer verwendeten Schema konstruieren kann.

a) Konstruieren Sie nun einen schnelleren n-Bit Inkrementer $CSA\text{-}INC_n$ auf Basis des in der Vorlesung vorgestellten n-Bit Conditional-Sum-Addierers. Geben Sie hierzu die Basiszelle $CSA\text{-}INC_1$ und den rekursiven Aufbau von $CSA\text{-}INC_n$ für $n > 1$ an. n sei hierbei eine Zweierpotenz, d. h. $n = 2^k$ für $k \in \mathbb{N}$.

b) Geben Sie die Tiefe Ihres n-Bit Inkrementers $CSA\text{-}INC_n$ an, und beweisen Sie Ihre Aussage. Geben Sie die Kosten Ihres n-Bit Inkrementers $CSA\text{-}INC_n$ *nur asymptotisch* an, und begründen Sie lediglich kurz.

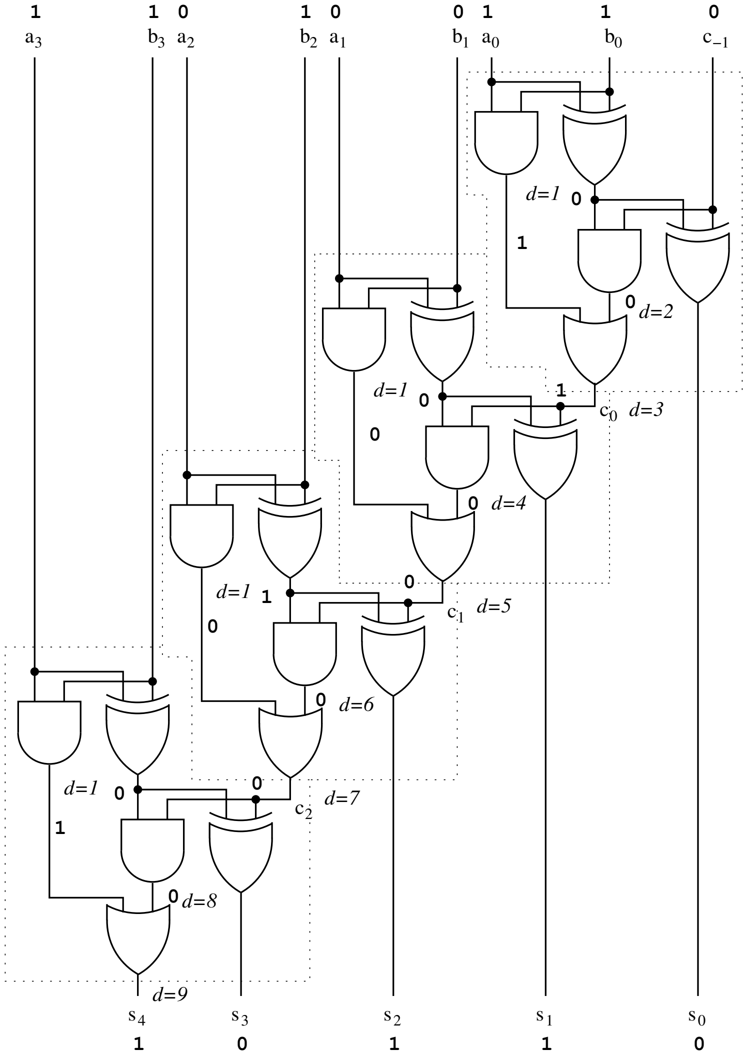
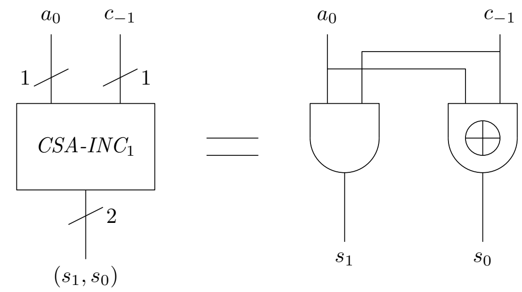
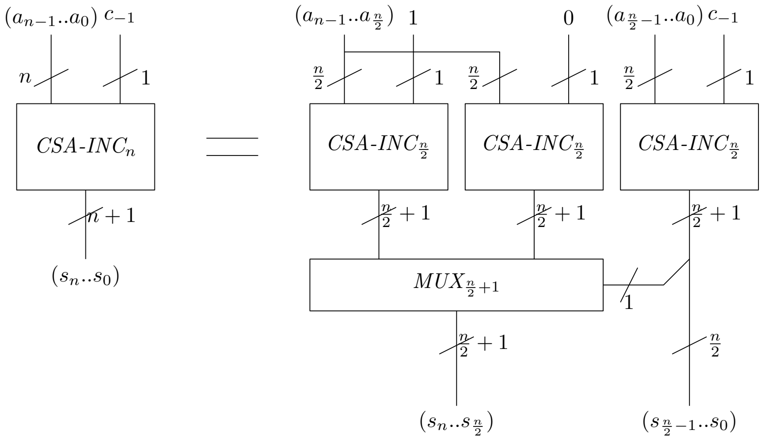


Abbildung 1: 4-Bit-Carry-Ripple-Addierer

Lösung:





`Punkte:` $CSA - INC_1$ [1]; $CSA - INC_n$ [2]

Tiefe [1.5 P]

$depth(CSA - INC_1) = depth(HA) = 1$

$depth(CSA - INC_n) = depth(MUX_{n/2+1}) + depth(CSA - INC_{n/2}) = 3 + depth(CSA - INC_{n/2}) = 3 + 3 + depth(CSA - INC_{n/2-1}) = ... = 3 + ... + 3 + depth(CSA - INC_{n/2^k})$ (mit k mal 3) $= k \cdot 3 + depth(CSA - INC_1) = 1 + 3 \cdot k = 1 + 3 \cdot \log n$

Kosten [1.5 P] $C(CSA - INC_n) \in O(n^{log(3)})$, wie $C(CSA_n)$: gleiche Konstruktion, der Unterschied ist der Grunbaustein (n=1) der aus einem Halbaddierer statt einem Volladdierer besteht, sprich 2 statt 5 Gatter.

$C(1) = C(HA) = 2$

$C(CSA - INC_n) = 3 \cdot C(CSA - INC_{n/2}) + C(MUX_{n/2+1}) = 3 \cdot C(CSA - INC_{n/2}) + 3 \cdot n/2 + 4$

Aufgabe 3 (3 Punkte)

In der Vorlesung wurde $\text{sext}(y) := y_{23}^8 y$ mit $y \in \mathbb{B}^{24}$ als die *Sign Extension* von y definiert. Betrachten Sie hier den allgemeinen Fall für $y \in \mathbb{B}^n$ mit $y = y_{n-1} \dots y_0$. Dann ist die *Sign Extension* von y um k Bits definiert als: $\text{sext}_\text{k}(y) := y_{n-1}^k y$

Beweisen Sie, dass $[y]_2 = [\text{sext}_\text{k}(y)]_2$ gilt.

Hinweis: Beachten Sie, dass es sich hier um die Zweier-Komplement-Darstellung handelt!

Lösung:

Es sollen hier zwei Lösungsmöglichkeiten betrachtet werden:

(1) Zurückführung auf Sign Extention um 1 Bit:

Lemma: Sei $a \in \mathbb{B}^n$, $a = a_{n-1} \dots a_0$. Dann gilt $[a] = [a_{n-1}a]_2$.
Beweis:

$$[a_{n-1}a]_2 = -a_{n-1} \cdot 2^n + \sum_{i=0}^{n-1} a_i \cdot 2^i = -a_{n-1} \cdot 2^n + \left(a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i \right) = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i = [a]_2$$

Damit: $[y]_2 = [y_{n-1}^1 y]_2 = [y_{n-1}^2 y]_2 = \dots = [y_{n-1}^k y]_2 = [\mathrm{sext_k}(y)]_2$.

(2) Direkter Beweis:
Sei $y \in \mathbb{B}^n$, $y = y_{n-1} \dots y_0$

$$\begin{aligned}
[\mathrm{sext_k}(y)]_2 = [\underbrace{y_{n-1} \dots y_{n-1}}_{k-mal} y]_2 &= \sum_{i=0}^{n-2} y_i \cdot 2^i + \sum_{i=n-1}^{n+k-2} y_{n-1} \cdot 2^i - y_{n-1} \cdot 2^{n+k-1} \\
&= \sum_{i=0}^{n-2} y_i \cdot 2^i + y_{n-1} \cdot \left(\sum_{i=0}^{n+k-2} 2^i - \sum_{i=0}^{n-2} 2^i \right) - y_{n-1} \cdot 2^{n+k-1} \\
&= \sum_{i=0}^{n-2} y_i \cdot 2^i + y_{n-1} \cdot \left(\frac{2^{n+k-1} - 1}{2-1} - \frac{2^{n-1} - 1}{2-1} \right) - y_{n-1} \cdot 2^{n+k-1} \\
&= \sum_{i=0}^{n-2} y_i \cdot 2^i + y_{n-1} \cdot 2^{n+k-1} - y_{n-1} \cdot 2^{n-1} - y_{n-1} \cdot 2^{n+k-1} \\
&= \sum_{i=0}^{n-2} y_i \cdot 2^i - y_{n-1} \cdot 2^{n-1} \\
&= [y]_2
\end{aligned}$$

□

`Punkte`: Rechnung mit der Interpretationsfunktion [1.5]; Wiederanwendung bis $[y] = [\mathrm{sext}(y)]$ [0.5], für direkten Beweis [2], falls nicht allgemein gezeigt [-1]

Aufgabe 4 (*Bonusaufgabe:* 3 Punkte)

Addieren Sie die folgenden Paare von 6-Bit Zweierkomplementzahlen mit der in der Vorlesung gezeigten Methode. Ist das Ergebnis nicht als 6-Bit Zweierkomplementzahl darstellbar, so geben Sie dies explizit an und begründen Sie.

a) $[100000]_2$ und $[011111]_2$

b) $[100000]_2$ und $[100000]_2$

c) $[010001]_2$ und $[011011]_2$

Lösung:

a) $[100000]_2 + [011111]_2$: $a_n \neq b_n$

	100000	= -32
	+011111	= 31
(Übertrag)	000000	
	=(0)111111	= -1

b) $[100000]_2$ und $[100000]_2$: $a_n = b_n \wedge b_n \neq s_n$

$$\begin{array}{rl} 100000 & = \text{-32} \\ +100000 & = \text{-32} \\ \text{(Übertrag) } 100000 & \\ \hline = (1)000000 & = 0 \end{array}$$

$\Rightarrow$ Überlauf

7 Bits nötig.

c) $[010001]_2$ und $[011011]_2$: $a_n = b_n \wedge b_n \neq s_n$

$$\begin{array}{rl} 010001 & = 17 \\ +011011 & = 27 \\ \text{(Übertrag) } 010011 & \\ \hline =(0)101100 & = \text{-20} \end{array}$$

$\Rightarrow$ Überlauf

7 Bits nötig.

Abgabe: 28. Juni 2023, $13^{\underline{00}}$ über das Übungsportal