Formale Grundlagen: Übung 4

Alexander Waldenmaier, Tutorin: Constanze Merkt

5. Dezember 2020

Aufgabe 4.1

Es gibt insgesamt $2^n + 1$ numerisch monotone Boolesche Funktionen $f : \mathcal{B}^n \to \mathcal{B}$.

Begründung: Im einfachsten Fall, $n = 1$ gibt es exakt 3 monotone Funktionen: $f_1(x) = 0, f_2(x) = x, f_3(x) = 1$. Die Funktion $f_4(x) = \overline{x}$ erfüllt die Bedingung nicht, da $f_4(x) = 1 \not\leq 0 = f_4(y)$ mit $x = 0 < 1 = y$. Aus einer Wertetabelle kann man leicht das Verhalten für höhere n ablesen:

$(x_1, x_2, \ldots, x_n)_{10}$	x_1	x_2	$\cdots$	x_n	f_1	f_2	$\cdots$	f_{2n+1}
0	0	0	$\cdots$	0	0	0	$\cdots$	1
1	0	0	$\cdots$	1	0	0	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2n-1$	1	1	$\cdots$	1	0	1	$\cdots$	1

Betrachtet man die möglichen Funktionen f_i in der Tabelle rechts, so erkennt man, dass für größer werdende i die „Grenze“, bei der die Funktion nicht mehr 0 sondern 1 herausgibt, sich je um eins nach oben verschiebt, bis irgendwann für jeden beliebigen Input x der Funktionswert stets 1 ergibt.

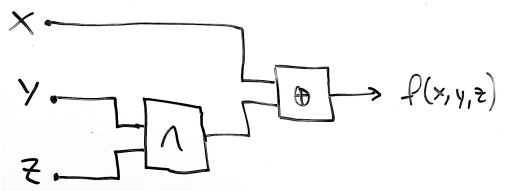
Aufgabe 4.2

Zunächst muss f in Ringsummennormalform (RSNF) umgewandelt werden, da darin nur $\oplus$ und $\wedge$ enthalten sind. Wir nutzen ein KV-Diagramm um eine orthogonale Form von f herzustellen:

$$
\begin{aligned}
f(x,y,z) &= (x\overline{z}) \vee (x\overline{y}) \vee (\overline{x}yz) \\
&\overset{\text{KV}}{=} x\overline{y}\overline{z} \vee x\overline{y}z \vee xy\overline{z} \vee \overline{x}yz \\
&= x\overline{y}\overline{z} \oplus x\overline{y}z \oplus xy\overline{z} \oplus \overline{x}yz \\
&= x(y \oplus 1)(z \oplus 1) \oplus x(y \oplus 1)z \oplus xy(z \oplus 1) \oplus (x \oplus 1)yz \\
&= \cancel{xyz} \oplus \cancel{xy} \oplus \cancel{xz} \oplus x \oplus \cancel{xyz} \oplus \cancel{xz} \oplus \cancel{xyz} \oplus \cancel{xy} \oplus \cancel{xyz} \oplus yz \\
&= x \oplus yz
\end{aligned}
$$

y \ z,x	00	01	11	10
0	0	1	1	0
1	0	1	0	1

Damit lässt sich die folgende Schaltung implementieren:



Aufgabe 4.3

Wir betrachten zunächst die folgende einfache Multiplikation zweier Binärzahlen:

	$1011 \cdot 1010$	$=$
	1011000	
+	0000000	
+	0010110	
+	0000000	
c	0100000	
s	1101110	

Die rechte Zahl hat eine 1 an der 1. und der 3. Stelle (0-basiert). Demnach wird die linke Zahl einmal um 3 Bits nach links verschoben (rechts mit Nullen aufgefüllt) bzw. einmal um 1 Bit nach links verschoben. Beide resultierenden Zahlen (grün markiert) werden dann addiert, um das Ergebnis der Multiplikation zu erhalten. Für jede 0 in der rechten Zahl muss nichts addiert werden (rot markiert).

Generell lässt sich also sagen. Das Ergebis der Multiplikation, m, wird zunächst mit 0 initialisiert. Dann werden alle Stellen b_i von b vom LSB bis hin zum MSB durchgegangen. Mit jeder Erhöhung von i wird die Zahl a um ein Bit nach links verschoben und rechts mit einer Null ergänzt. An jeder Stelle i, an der $b_i = 1$ ist, wird die derzeitige (verschobene) Version von a zu m hinzuaddiert (ansonsten wird 0 addiert). Sobald alle Stellen von b durchgegangen sind, ist die Multiplikation beendet und das Ergebnis m kann ausgelesen werden.

Um zwei n-stellige Binärzahlen a und b zu addieren, benötigt es n miteinander verschaltetete Volladdierer (siehe Abbildung 1). Dabei bekommt der Volladdierer (VA), der die beiden Least Significant Bits (LSB) addiert, einen „Carry-In“ von 0 und gibt seinen „Carry-Out“ an den darauffolgenden VA weiter. Diese Kette führt sich fort, bis am Ende alle Stellen s_i der Summe s bekannt sind, sowie ein optionaler Carry-Out aus dem letzten VA. Dies würde gleichzeitig einen Overflow signalisieren. Die Größe eines Volladdierers (wie im Skript) beträgt $C(\text{add}) = 5$, die Tiefe $D(\text{add}) = 3$. Für den n-stelligen Volladdierer gilt: $C(\text{nadd}) = n \cdot C(\text{add}) = 5n$, $D(\text{nadd}) = n \cdot D(\text{add}) = 3n$ (da jeder VA zunächst auf den Carry-In vom vorigen VA warten muss).

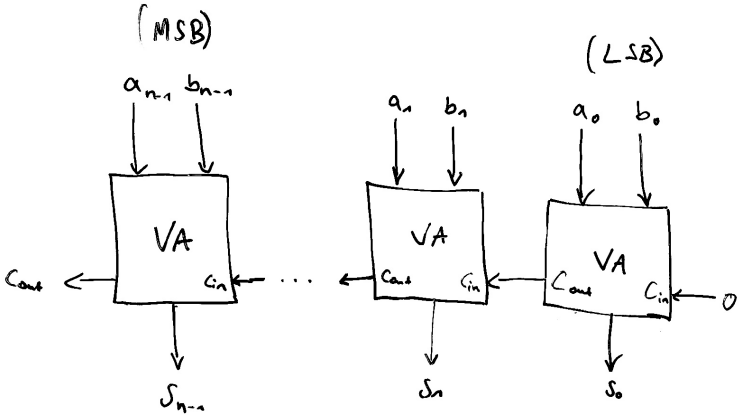
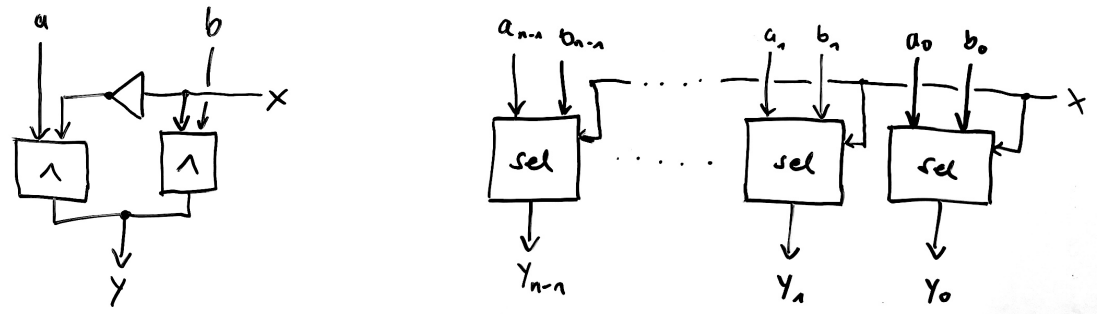


Abbildung 1: n-stelliger Volladdierer (NVA)

Um den Multiplikator in einer Schaltung umzusetzen, wird zusätzlich noch ein Multiplexer „selector“ benötigt. Abhängig von einer Variable x (in diesem Fall b_i) wird entweder a oder b ausgewählt (in diesem Fall a_j oder 0). Ein solcher Selektor ist in Abbildung 2a dargestellt. Seine Größe beträgt $C(\text{sel}) = 3$ und seine Tiefe $D(\text{sel}) = 2$. Die n-stellige Variante verbindet n solcher Selektoren und ist in Abbildung 2b dargestellt. Die Größe beträgt $C(\text{nsel}) = n \cdot C(\text{sel}) = 3n$ und die Tiefe $D(\text{nsel}) = 2$.

Mit den n-stelligen Volladdierern und Multiplexern lässt sich nun der Multiplikator zusammenbauen, der in Abbildung 3 dargestellt ist. Das Ergebnis der Multiplikation zum i-ten Schritt steckt jeweils in der Variable $m^{(i)}$, die der Reihe nach an die n n-stelligen Volladdierer (NVA) übergeben wird. Diese erhalten als zweiten Summanden die um i Bits verschobene Zahl a, oder 0 (wenn b_i 0 war). Es ist eine Menge an Verdrahtung notwendig, um die Zahl n Bits der Zahl a „aufzudröseln“ und an die n Selektoren zu übergeben.



(a) 1-stelliger Selektor (sel) (b) n-stelliger Selektor (n-sel)

Abbildung 2: Multiplexer

Die Größe des Multiplikators beträgt $C(\text{nmult}) = n \cdot (C(\text{nsel}) + C(\text{nadd})) = n \cdot (5n + 3n) = 8n^2$ und die Tiefe $D(\text{nmult}) = n \cdot D(\text{nadd}) + D(\text{nsel}) = 3n^2 + 2$ (da der letzte Volladdierer zunächst auf das Ergebnis aller vorigen NVA warten muss).

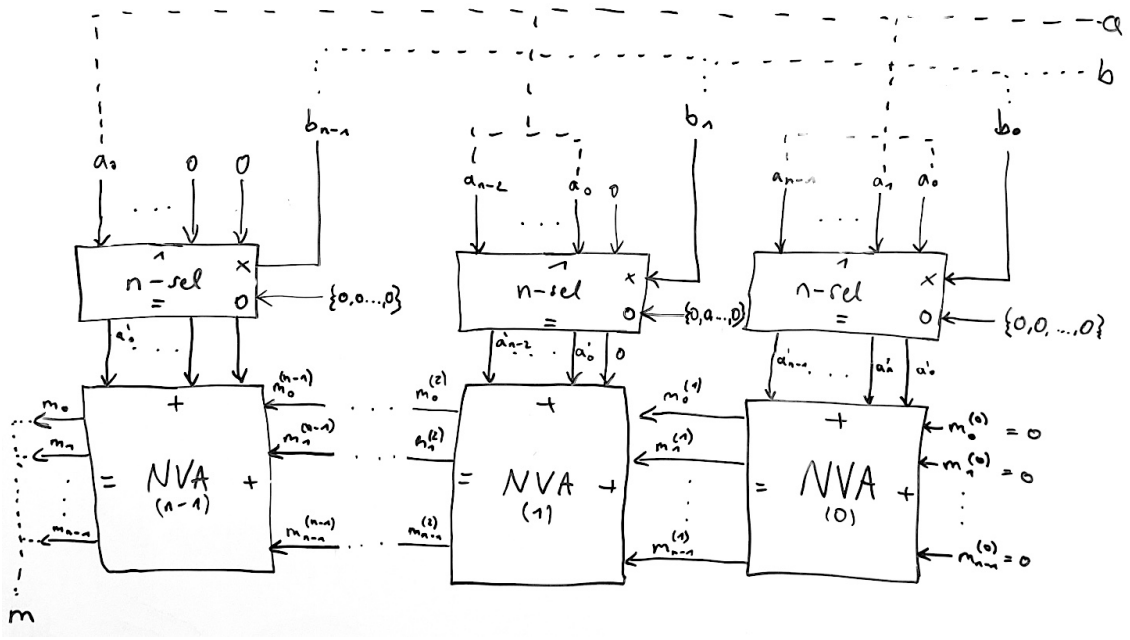


Abbildung 3: n-stelliger Multiplikator

Aufgabe 4.4

a)

$$
\begin{aligned}
f(x,y,z) &= () + () \\
&= () \cdot () + () \cdot () \\
&= \overline{()} \cdot () + () \cdot \overline{()} \\
&= \overline{z} \cdot (() + ()) + z \cdot \overline{(() + ())} \\
&= \overline{z} \cdot (() \cdot () + () \cdot ()) + z \cdot \overline{() \cdot () + () \cdot ()} \\
&= \overline{z} \cdot (\overline{()} \cdot x + y \cdot \overline{()}) + z \cdot \overline{\overline{()} \cdot x + y \cdot \overline{()}} \\
&= \overline{z} \cdot (\overline{y}x + y\overline{x}) + z \cdot \overline{\overline{y}x + y\overline{x}}
\end{aligned}
$$

Die leeren Klammern stellen jeweils einen Platzhalter für weiterfolgende Teile der Schaltung dar.

b)
- Größe: 4x NOT, 4x AND, 2x OR $\Rightarrow C(S) = 10$
- Tiefe (Längster Pfad): x $\rightarrow$ NOT $\rightarrow$ AND $\rightarrow$ OR $\rightarrow$ NOT $\rightarrow$ AND $\rightarrow$ OR $\Rightarrow D(S) = 6$