Entwurf und Implementierung digitaler Schaltungen mit VHDL

Lösungen zur 1. Übung

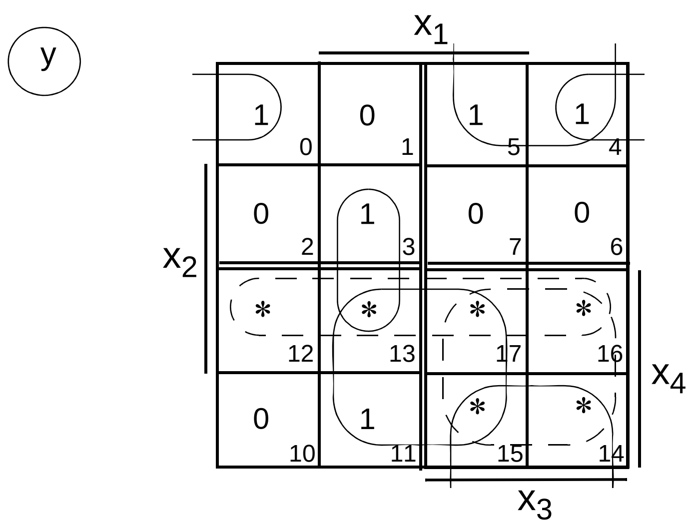
1. Aufgabe:

1.1

j (dez.) 0-9: Linie	x_4,x_3,x_2,x_1	y
0	0000	1
1	0001	0
2	0010	0
3	0011	1
4	0100	1
5	0101	1
6	0110	0
7	0111	0
8	1000	0
9	1001	1
10	1010	*
11	1011	*
12	1100	*
13	1101	*
14	1110	*
15	1111	*

Rest: *

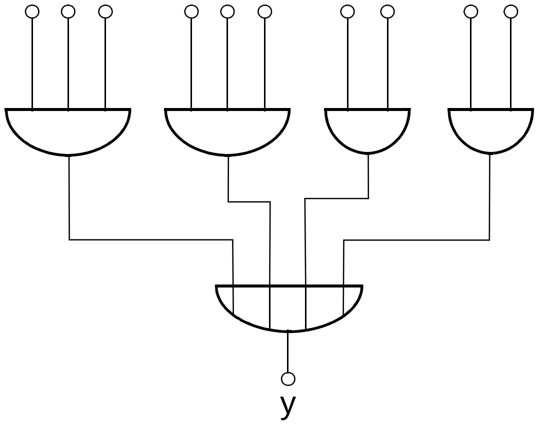
1.2, 1.3



$\tau_{max}(\underline{E}\cup\underline{D})$: alle eingezeichneten Schleifen (gestrichelte und durchgezoge Linien)

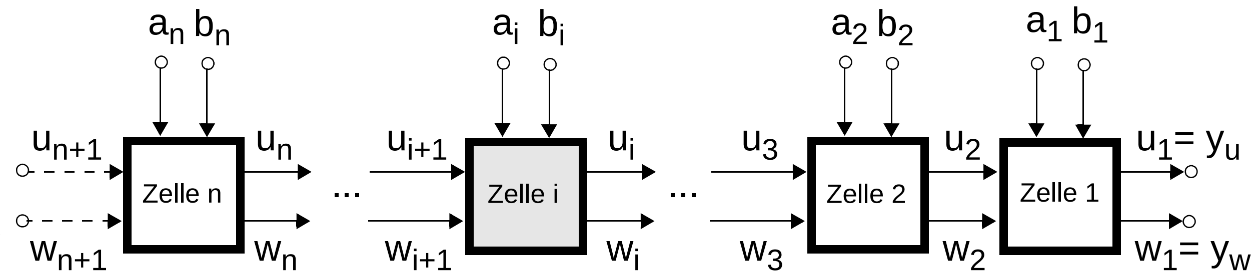
$\tau_{min}(\underline{E}\cup\underline{D}')$ = { 0–00, –011, 1– –1, –10– } (nur mit durchgezogener Linie gezeichnete Schleifen; ausschließlich Kernkuben; eindeutige Lösung)

DMF: $y = \overline{x}_4\,\overline{x}_2\,\overline{x}_1 \lor \overline{x}_3\,x_2\,x_1 \lor x_4\,x_1 \lor x_3\,\overline{x}_2$



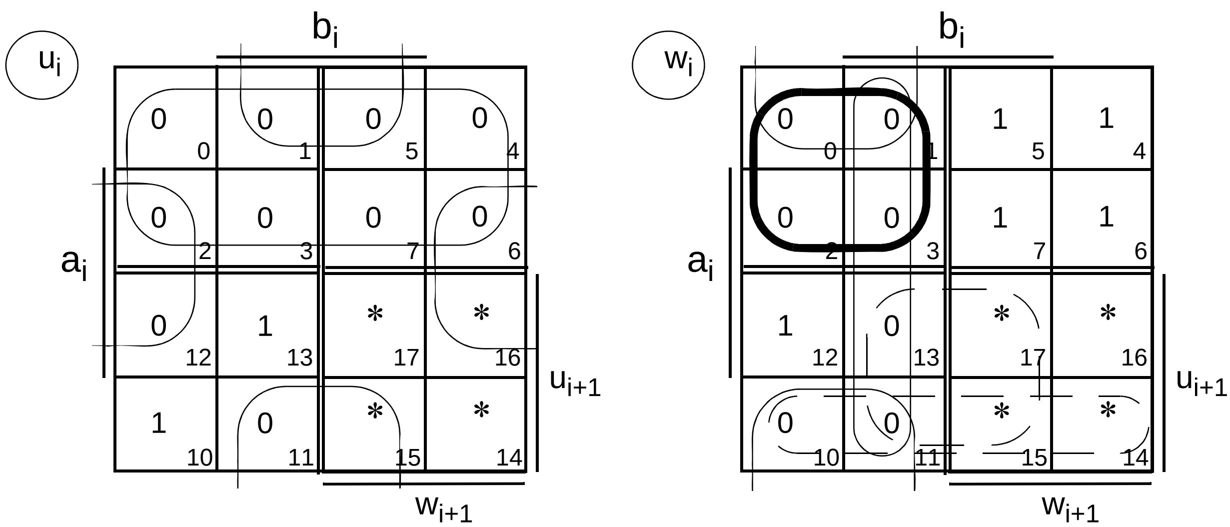
2. Aufgabe:

2.1) Wie y_u und y_w am Ausgang von Zelle 1, können u_i und w_i am Ausgang der i-ten Zelle Auskunft darüber geben, ob $a_n...a_i$ grösser als, kleiner als oder gleich $b_n... b_i$ ist.



2.2

u_{i+1}	w_{i+1}	a_i	b_i	u_i	w_i
0	0	–	–	0	0
0	1	–	–	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	0
	Rest			*	*



(fett: einziger Kernkubus)

Konjunktive Minimalformen => maximale *Null*kuben suchen!

u_i: $\tau_{min}(\underline{N}\cup\underline{D}') = \{$ 0– – –, – –01, – –10 $\}$ (jeder ein Kernkubus)

=> einzige KMF: $u_i = u_{i+1} \bullet (a_i \vee \bar{b}_i) \bullet (\bar{a}_i \vee b_i)$

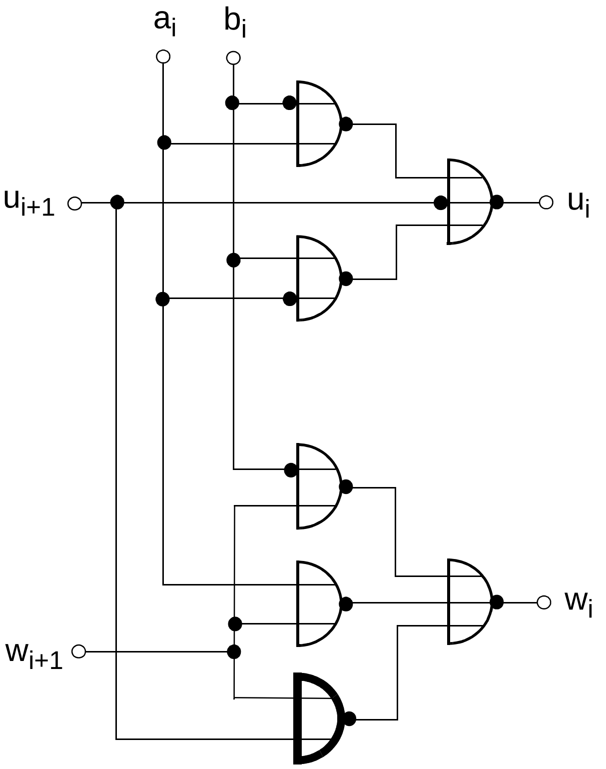
w_i: $\tau_{min1}(\underline{N}\cup\underline{D}'') = \{$ **00**– –, –0–1, –00– $\}$, $\tau_{min2}(\underline{N}\cup\underline{D}''') = \{$ **00**– –, 1– –1, 1–0– $\}$

$\tau_{min3}(\underline{N}\cup\underline{D}'''') = \{$ **00**– –, –0–1, 1–0– $\}$, $\tau_{min4}(\underline{N}\cup\underline{D}''''') = \{$ **00**– –, 1– –1, –00– $\}$

=> KMF$_1$: $w_i = (\mathbf{u_{i+1} \vee w_{i+1}}) \bullet (w_{i+1} \vee \bar{b}_i) \bullet (w_{i+1} \vee a_i)$

KMF$_2$: $w_i = (\mathbf{u_{i+1} \vee w_{i+1}}) \bullet (\bar{u}_{i+1} \vee \bar{b}_i) \bullet (\bar{u}_{i+1} \vee a_i)$, KMF$_3$= ... , KMF$_4$= ...

Ein ODER-UND-Netz lässt sich kostenneutral in ein strukturgleiches, äquivalentes NOR-NOR-Netz wandeln (Regel von de Morgan oder „Punkteschieben“). Bei Wahl von KMF_1 für w_i ergibt sich folgendes Netz:

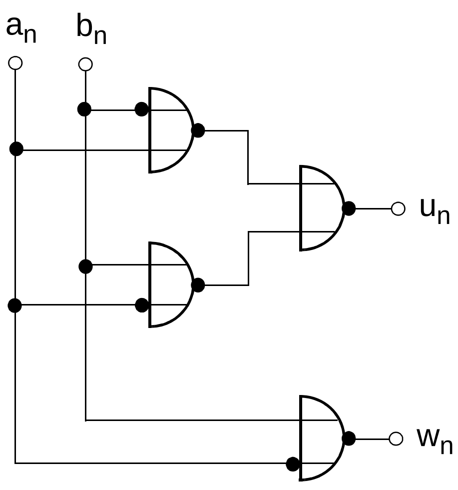


2.3

- Zelle 1 lässt sich nicht vereinfachen; sie muss ja ggf. allein die Entscheidung treffen.
- Zelle n dagegen wäre mit $u_{n+1} = 1$ und $w_{n+1} = 0$ anzusteuern. Daraus ergibt sich für diese Zelle folgende Vereinfachung:

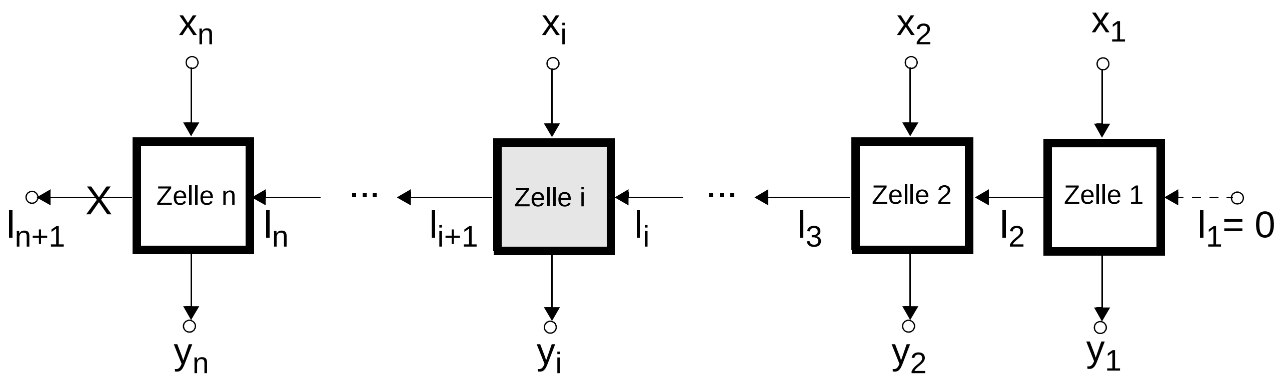
$u_n = 1 \bullet (a_n \vee \overline{b_n}) \bullet (\overline{a_n} \vee b_n)$

$w_n = (1 \vee 0) \bullet (0 \vee \overline{b_n}) \bullet (0 \vee a_n) = \overline{b_n} \bullet a_n = \overline{(b_n \vee \overline{a_n})}$ (aus KMF_1)



3. Aufgabe:

3.1



x_i: i-te Komponente des Eingabevektors X

l_i: Angabe, ob mindestens eine Komponente x_j rechts von x_i gleich 1 ist ($l_i = 1$) oder nicht ($l_i = 0$)

l_{i+1}: Wie l_i, aber an die linke Zelle ($l_{i+1} = 1$: mindestens eine der Komponenten $x_i \ldots x_2 x_1$ ist gleich 1; $l_{i+1} = 0$: keine der Komponenten $x_i \ldots x_2 x_1$ ist gleich 1)

y_i: i-te Komponente von Y (Zweierkomplement von X)

3.2

l_i	x_i	l_{i+1}	y_i
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

Aus der Funktionstabelle ist unmittelbar abzulesen:

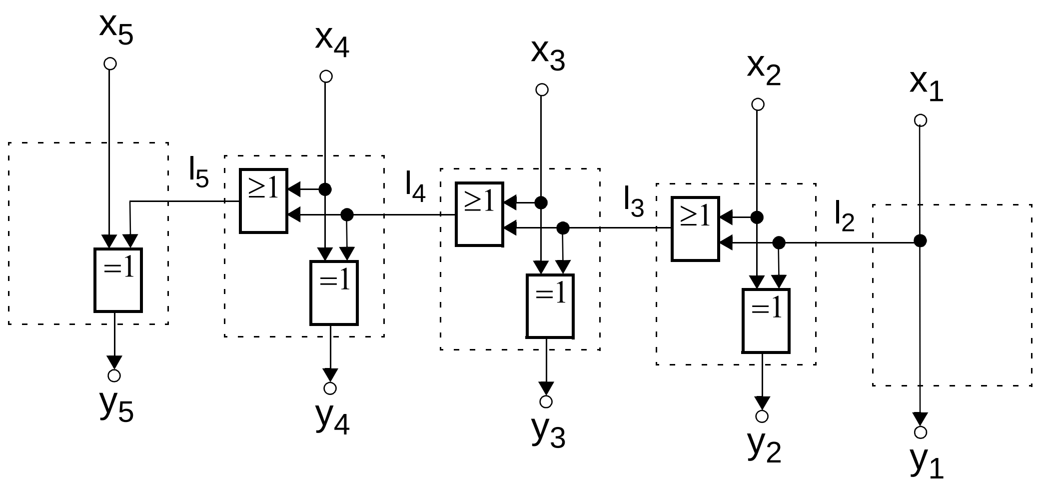
$y_i = l_i \not\equiv x_i$

$l_{i+1} = l_i \vee x_i$

Für n-te Zelle: l_{n+1} entfällt (nicht erforderlich)

Für 1. Zelle: $l_1 = 0 \Rightarrow y_1 = 0 \not\equiv x_1 = x_1$; $l_2 = 0 \vee x_1 = x_1$

3.3



4. Aufgabe:

x_2,x_1,x_0	y_2,y_1,y_0
0 0 0	0 * *
0 0 1	1 0 0
0 1 0	1 0 1
0 1 1	1 0 1
1 0 0	1 1 0
1 0 1	1 1 0
1 1 0	1 1 0
1 1 1	1 1 0

Kompakte Tabelle:

x_2,x_1,x_0	y_2,y_1,y_0
0 0 0	0 * *
0 0 1	1 0 0
0 1 -	1 0 1
1 - -	1 1 0