Prof. Dr.-Ing. Dr. h. c. J. Becker

Digitaltechnik

8. Tutorium - Lösung

Institut für Technik der Informationsverarbeitung (ITIV), Karlsruher Institut für Technologie (KIT)

Aufgabe 1: Schieberegister

Im Folgenden soll ein zwei-Bit Schieberegister als Zustandsautomat dargestellt werden. Die Eingangsgröße E_0^v des Automaten entspreche dabei dem einzuschiebenden Bit, während die zwei Bits q_0 und q_1 des Schieberegisters sowohl den Zustand S_k^v als auch die Ausgabe A_v^h des Automaten bilden sollen.

1.1 Vervollständigen Sie folgenden Automatengraphen des Schieberegisters, sodass das Schieberegister nach *links* verschiebt.

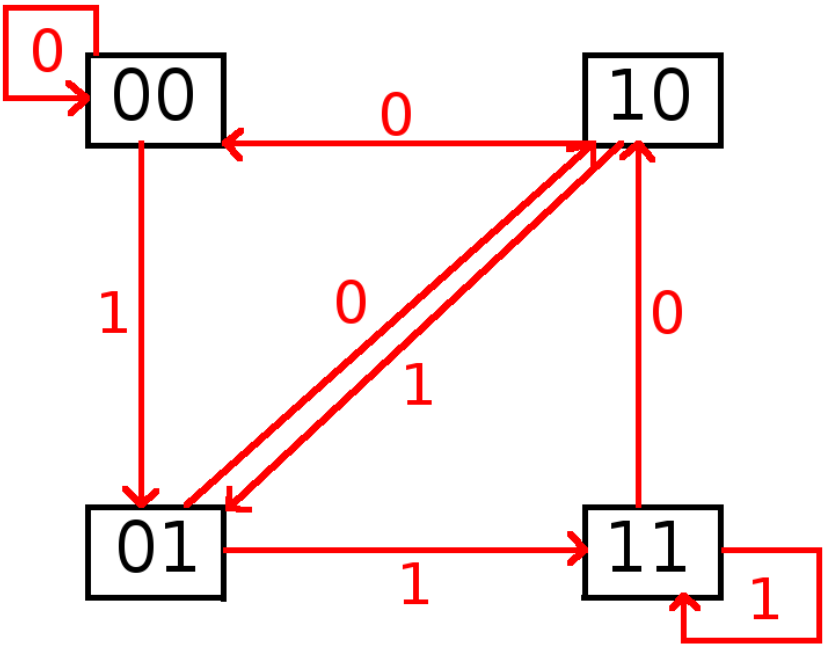


Abbildung 1: Übergrangsgraph des Schieberegisters

1.2 Um welchen Automatentyp handelt es sich?

Medwedew, da die Ausgabe gleich dem Zustand ist.

© ITIV 2015

Aufgabe 2: CMOS-Schaltnetze

2.1 Gegeben sei die pull-up-Funktion eines CMOS-Schaltnetzes:

$$F = (\overline{a} + \overline{c}\&b)\&d$$

Geben Sie die für eine wohldefinierte CMOS-Schaltung nötige Abhängigkeit zwischen G und F an. Geben Sie die pull-down-Funktion G so an, dass diese als Transistorschaltnetz realisiert werden kann.

Bedingung für wohldefinierte Schaltung: $G = \overline{F}$

$G = \overline{(\overline{a} + \overline{c}b)d} = (a(\overline{c}\overline{b})) + \overline{d} = (a(\overline{b} + c)) + \overline{d}$

2.2 Zeichnen Sie das pull-down Schaltnetz der Funktion G aus Aufgabenteil 2.1 in Abbildung 2.

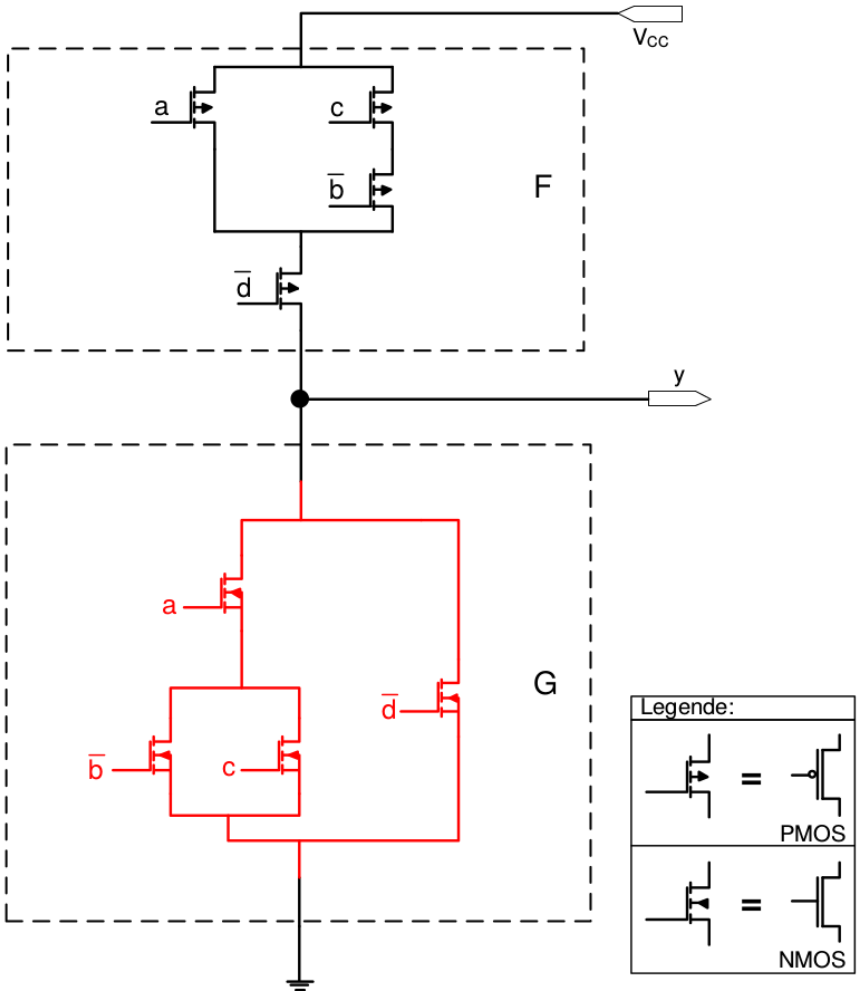


Abbildung 2: CMOS-Schaltnetz

Hinweis: Die Substratanschlüsse sind in der Abbildung der Übersichtlichkeit wegen nicht kontaktiert. Die Substratanschlüsse der PMOS-Transistor werden V_{cc} , die der NMOS-Transistoren mit Ground kontaktiert.

2.3 Gegeben sind folgende pull-up- bzw. pull-down-Funktionen F‘ bzw. G‘. Stellen Sie mithilfe einer Wahrheitstabelle fest, ob die durch die beiden Funktionen definierte CMOS-Schaltung wohldefiniert ist. Geben Sie, falls erforderlich, die Eingangsbelegungen an, welche zu Kurzschlüssen oder einem undefinierten Ausgangssignal führen.

$$G‘ = \overline{a}d + \overline{c}\overline{b}d$$
$$F‘ = c\overline{b} + b\overline{d}a + \overline{c}\overline{d}a$$

Lösung über Wahrheitstabelle:
→ die Eingangskombination a=0, b=0, c=1, d=1 führt zu einem Kurzschluss

d	c	b	a	F‘	G‘	y
0	0	0	0	0	0	Undef.
0	0	0	1	1	0	1
0	0	1	0	0	0	Undef.
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	0	0	Undef.
0	1	1	1	1	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	0	Undef.
1	1	0	0	1	1	KS
1	1	0	1	1	0	1
1	1	1	0	0	1	0
1	1	1	1	0	0	Undef.

Tabelle 1: Wahrheitstabelle der Schaltung

→ folgende Eingangskombinationen sind nicht vollständig abgedeckt:
a=1, b=1, c=1, d=1; a=0, b=0, c=0, d=0;
a=1, b=1, c=0, d=1; a=0, b=1, c=1, d=0;
a=0, b=1, c=0, d=0 → Undef= abd, $\overline{a}c\overline{d}$, $\overline{a}bc\overline{d}$

Alternativ ist auch eine Lösung der Aufgabe mit den aus der Vorlesung bekannten Formeln möglich:
Prüfen auf Kurzschlüsse:
Kurzschlussfreiheit bei: $F‘ \cdot G‘ = 0$

$$\begin{aligned} F‘ \cdot G‘ &= [c\overline{b} + b\overline{d}a + \overline{c}\overline{d}a][\overline{a}d + \overline{c}\overline{b}d] \\ &= c\overline{b}\overline{a}d + c\overline{b}\overline{c}\overline{b}d + b\overline{d}a\overline{a}d + b\overline{d}ac\overline{b}d + \overline{c}\overline{d}a\overline{a}d + \overline{c}\overline{d}ac\overline{b}d \\ &= c\overline{b}\overline{a}d \end{aligned}$$

→ Die Eingangskombination a=0, b=0, c=1, d=1 führt zu einem Kurzschluss.
Prüfen auf vollständige Definition:
$F‘ + G‘ = [c\overline{b} + b\overline{d}a + \overline{c}\overline{d}a] + [\overline{a}d + \overline{c}\overline{b}d]$

Bestimme die fehlerhafte Eingangskombination durch Negation der Gleichung:

$$\overline{F' + G'} = 0$$

$$\begin{aligned}
\overline{F' + G'} &= \overline{c\bar{b} + b\bar{d}a + \bar{c}\bar{d}a + \bar{a}d + \bar{c}\bar{b}d} \\
&= (\bar{c} + b)(\bar{b} + d + \bar{a})(c + d + \bar{a})(a + \bar{d})(c + b + \bar{d}) \\
&= (\bar{c}\bar{b} + \bar{c}d + \bar{c}\bar{a} + bd + b\bar{a})(ca + c\bar{d} + da + \bar{a}\bar{d})(c + b + \bar{d}) \\
&= (\bar{c}\bar{b}da + \bar{c}\bar{b}\bar{a}\bar{d} + \bar{c}da + \bar{c}\bar{a}\bar{d} + bdca + bda + b\bar{a}c\bar{d} + b\bar{a}\bar{d})(c + b + \bar{d}) \\
&= bdca + bdac + b\bar{a}c\bar{d} + b\bar{a}\bar{d}c + \bar{c}dab + \bar{c}\bar{a}\bar{d}b + bdca + bda + b\bar{a}c\bar{d} + b\bar{a}\bar{d} \\
&\quad + \bar{c}\bar{b}\bar{a}\bar{d} + \bar{c}\bar{a}\bar{b} + b\bar{a}n\bar{d} + b\bar{a}\bar{d} \\
&= bda + b\bar{a}\bar{d} + \bar{c}\bar{a}\bar{d}
\end{aligned}$$

$$\overline{F' + G'} \neq 0 \Rightarrow F' + G' \neq 1$$

→ Es existieren fehlerhafte Eingangskombination:
a=1, b=1, d=1
a=0, b=1, d=0
a=0, c=0, d=0

Weitere Alternative:
Die Funktionswerte von F‘ und G‘ werden in ein einziges Symmetriediagramm geschrieben (hier F‘ links und G‘ rechts). In den Zellen, in denen zwei Einsen oder zwei Nullen auftauchen, liegt ein Fehler vor:

- „1|0“bedeutet $F' = 1$ und $G' = 0$, also $y = 1$
- „0|1“bedeutet $F' = 0$ und $G' = 1$, also $y = 0$
- „0|0“bedeutet $F' = 0$ und $G' = 0$, ⇒ „undefiniert“
- „1|1“bedeutet $F' = 1$ und $G' = 1$, ⇒ Kurzschluss

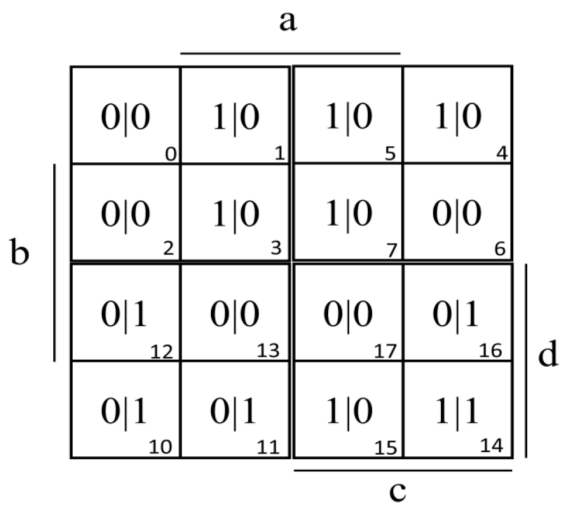


Abbildung 3: Lösung über Eintragung ins Symmetriediagramm

Benachbarte Einträge können noch zu Blöcken zusammengefasst werden, was das Finden der fehlerhaften Eingangsbelegungen beschleunigt.

Aufgabe 3: Steuerung einer Lichtanlage

Die folgende Zusatzaufgabe soll nur teilweise im Tutorium besprochen werden und den Studenten primär zur Klausurvorbereitung dienen. In der Aufgabe wird ein Querschnitt der in der Vorlesung besprochenen Themengebiete behandelt und miteinander verknüpft.

Die Eingabe durch ein Wort soll im Folgenden codiert werden um leichter mit ihr arbeiten zu können. Daraufhin sollen die entstandenen Codewörter auf verschiedene Weisen geprüft werden. Im Anschluss soll eine Konvertierung der Wörter zu einer boolschen Funktion stattfinden, mit welcher am Schluss die Ansteuerung einer Lichtschaltung vorgenommen werden soll. Diese Ansteuerfunktion soll von einer Gatterschaltung in eine CMOS Schaltung überführt werden.

Die möglichen Eingaben der Anlage sind folgende:

- „licht“: Aktivierung des Lichtbusses; die Lichtanlage wartet auf einen Befehl
- „an“: Lichtleistung auf volle Leistung
- „aus“: Licht aus machen
- „staerker“: Lichtleistung verstärken
- „schwaecher“: Lichtleistung abschwächen
- „effekt“: Lichteffekt wird ein/aus geschaltet

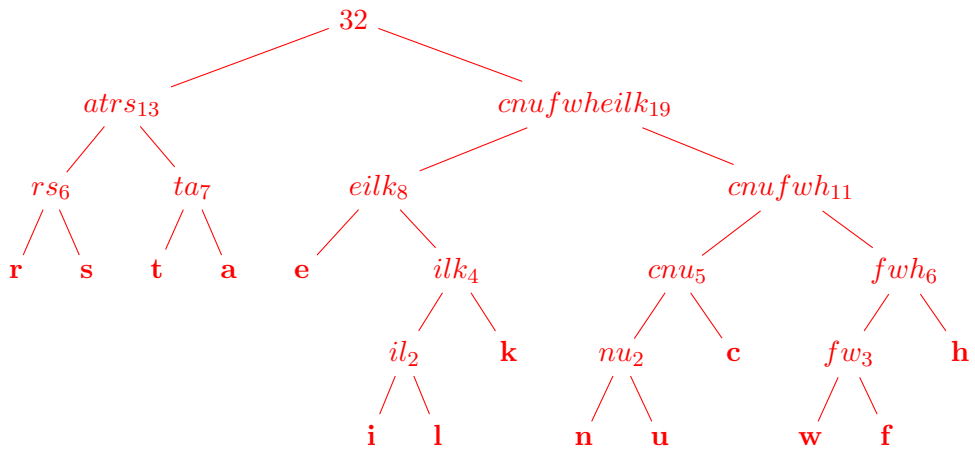
3.1 Um den Bus möglichst schnell wieder frei geben zu können, sollen die gesendeten Wörter (Eingaben der Anlage) möglichst kurz codiert werden. Aus Gründen sollen die Wörter jedoch komplett gesendet werden, das heißt, jeder Buchstabe soll nach Auftrittshäufigkeit optimiert codiert werden. Die Häufigkeit in der die Wörter auftreten soll nicht beachtet werden, lediglich die Häufigkeit der Buchstaben. Erstellen Sie einen Huffman Codierbaum und tragen Sie das jeweilige Codewort für die Buchstaben in die vorgegebene Tabelle ein.

Buchstabe	Anzahl des Buchstaben	Code
a	4	011
c	3	1101
e	4	100
f	2	11101
h	3	1111
i	1	10100
k	2	1011
l	1	10101
n	1	11000
r	3	000
s	3	001
t	3	010
u	1	11001
w	1	11100

Tabelle 2: Auftrittsanzahl der Buchstaben

Partitionierungskonvention:

- *Sortieren Sie die Elemente zu Beginn entsprechend den Auftrittshäufigkeiten* ***aufsteigend von links nach rechts****. Falls unterschiedliche Knoten dieselbe Auftrittshäufigkeiten haben, sortieren Sie diese bitte alphabetisch von links nach rechts. Gehen Sie bei jedem nötigen Sortierschritt nach diesem Schema vor.*
- *Weisen Sie den linken Ästen des entstehenden Baumes die „0“ zu, den rechten Ästen die „1“*



3.2 Wie groß ist die durchschnittliche Codewortlänge? Wie viele Bits kann man durch die Huffman-Codierung einsparen, wenn man sonst eine ungewichtete Codierung der Buchstaben vornehmen würde?

Durchschnittliche Codewortlänge=$(17 \cdot 3 + 8 \cdot 4 + 7 \cdot 5) : 32 = \frac{118}{32} = 3,6875$
Ungewichteter Code: 14 nötige Codewörter $\rightarrow$ Anzahl $CW = \lceil ld14 \rceil = 4$
Somit spart man durchschnittlich 0,3125 Bit pro Wort ein, wenn man die gefundene Codierung verwendet. Vorausgesetzt hierfür ist, dass alle Wörter gleich oft gesendet werden.

3.3 *Hinweis: Gleichen Sie Ihre Ergebnisse mit denen der Musterlösung ab und nutzen Sie ggf. die der Musterlösung um weiter zu rechnen.*
Im Folgenden soll folgende Abfolge gesendet werden: „licht“„an“„aus“. Die Wörter sollen zusätzlich mit einer Blocksicherung gesichert werden. Es soll gerade Parität verwendet werden. Tragen Sie die Bits in folgende Tabelle ein. Jedes Datenwort soll aus 10 Nutzdatenbits bestehen. Um die Übertragung gegen Burstfehler zu schützen soll zusätzlich Scrambling verwendet werden. Geben Sie die ersten zehn gesendeten Bits an. Welche Länge an Burstfehlern kann maximal erkannt werden?

1	0	1	0	1	1	0	1	0	0	1
1	1	0	1	1	1	1	1	0	1	0
0	0	1	1	1	1	0	0	0	0	0
1	1	1	1	0	0	1	0	0	1	0
1	0	1	1	1	1	0	0	0	0	1

Tabelle 3: Paritätssicherung der Übertragung

Es können maximal Burstfehler der Länge fünf erkannt werden.
Erste zehn gesendete Bits: 1101 1010 10

3.4 Wie groß ist der Overhead der durch die Sicherung erzeugt wird?
Nutzdatenbits n= 40 Bits für Sicherung s = 4+11 = 15. Overhead = $\frac{s}{n} = \frac{15}{40} = 37,5\%$

3.5 Die Lampe der Anlage soll mittels eines Motors nach oben oder unten bewegt werden, um den Strahl bewegen zu können. Diese Bewegung wird durch einen mechanischen Taster gesteuert, der die beiden Steuersignale Licht hoch (SH=1) und Licht runter (SR=1) ausgibt, wenn dieser in die entsprechende Richtung bewegt wird. Weiterhin stehen ein oberer und unterer Endschalter (ESH, ESR) zur Verfügung, die jeweils zu eins gesetzt werden, sobald das Licht die maximale (minimale) Höhe erreicht.
Als Ausgänge werden zwei Steuersignale für den Motor verwendet mit denen die Lichtanlage herauf bewegt (Motor_H = 1) oder hinunter bewegt (Motor_R = 1) werden kann. Bei Tastendruck bewegt sich das Licht in die entsprechende Richtung bis der obere oder untere Maximalwert erreicht ist. Bei losgelassenem Taster bewegt sich das Licht weiter. Die Bewegung kann durch einen entgegen gesetzten Tastendruck jederzeit gestoppt werden.
Hinweise:

- Die Tastersignale sind jeweils nur für einen Takt eins, also nur bei einem Zustandsübergang aktiv
- Es ist mechanisch ausgeschlossen, dass beide Tastersignale gleichzeitig eins sind
- Die Endschalter können mechanisch ebenfalls nicht beide eins sein
- Um Beschädigungen des Motors zu vermeiden ist darauf zu achten, dass immer nur eines der Steuersignale eins ist.

Realisieren Sie diese Motorsteuerung mit Hilfe eines Moore-Automaten.

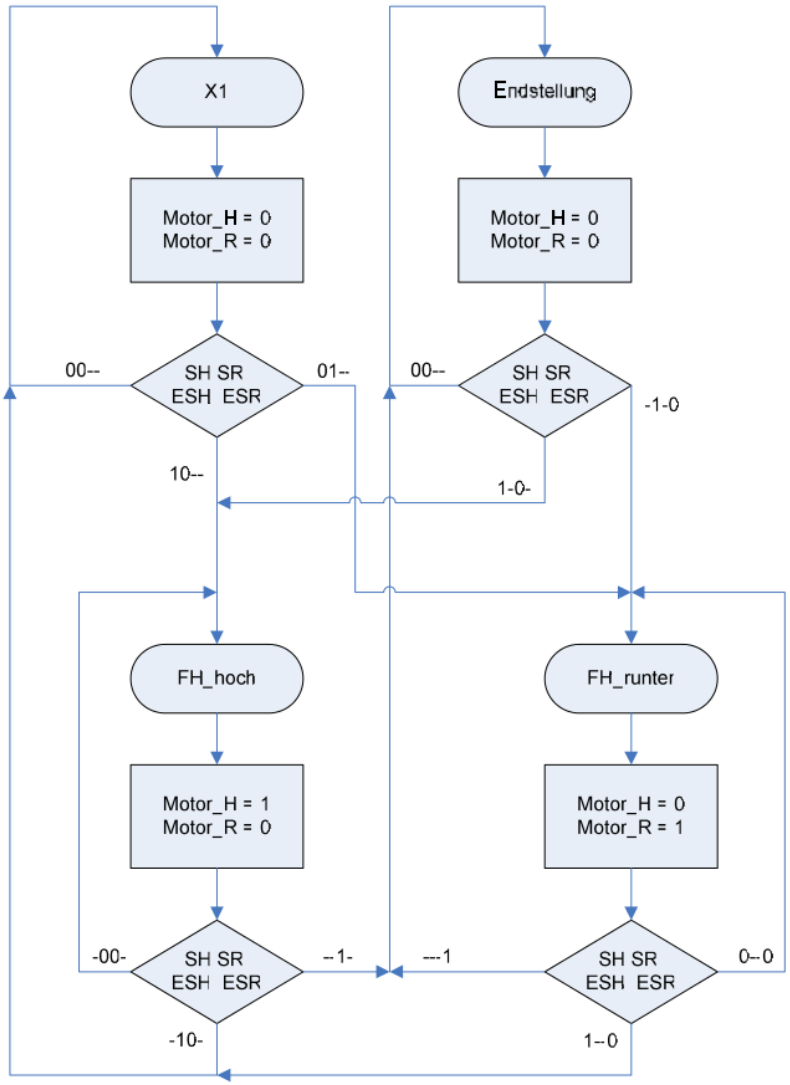


Abbildung 4: Übergangsgraph des Automaten zur Lichtanlagensteuerung

3.6 Die möglichen Eingaben „staerker“, „schwaecher“ und „licht“ sollen im Folgenden vernachlässigt werden, nur noch „an“ oder „aus“ ist relevant.
Nun soll die einfache Ansteuerung der Stromversorgung für die Lichtanlage realisiert werden. Für bestimmte Codewörter soll hier die Ausgabe der Funktion „1“ sein, ansonsten „0“.
Die Codewörter für rot, gelb, blau, weiß sind: -101, 00-0, 100- und 0-11, somit ergibt sich für die Stromversorgung die Ansteuerfunktion:

$$y(d,c,b,a) = a\bar{b}c \vee \overline{a}c\overline{d} \vee \bar{b}\bar{c}d \vee ab\overline{d}$$

Minimieren Sie die Funktion in folgender Reihenfolge: d,c,b,a. Zeichnen Sie eine minimale Multiplexerschaltung um die Minimierung zu realisieren.

Entwicklung nach d:

$$y(0,c,b,a) = a\bar{b}c \vee \overline{ac} \vee ab$$
$$y(1,c,b,a) = a\bar{b}c \vee \bar{b}\bar{c}$$

Entwicklunug nach c:

$$y(0,0,b,a) = \overline{a} \vee ab \qquad y(1,0,b,a) = \bar{b}$$
$$y(0,1,b,a) = a\bar{b} \vee ab = a \qquad y(1,1,b,a) = a\bar{b}$$

Entwicklung nach b:

$$y(0,0,0,a) = \overline{a} \qquad y(1,0,0,a) = 1$$
$$y(0,0,1,a) = 1 \qquad y(1,0,1,a) = 0$$
$$y(0,1,-,a) = a$$
$$y(1,1,0,a) = a$$
$$y(1,1,1,a) = 0$$

Entwicklung nach a:

$$y(0,0,0,0) = 1$$
$$y(0,0,0,1) = 0$$
$$y(0,1,-,0) = 0 \qquad y(1,1,0,0) = 0$$
$$y(0,1,-,1) = 1 \qquad y(1,1,0,1) = 1$$

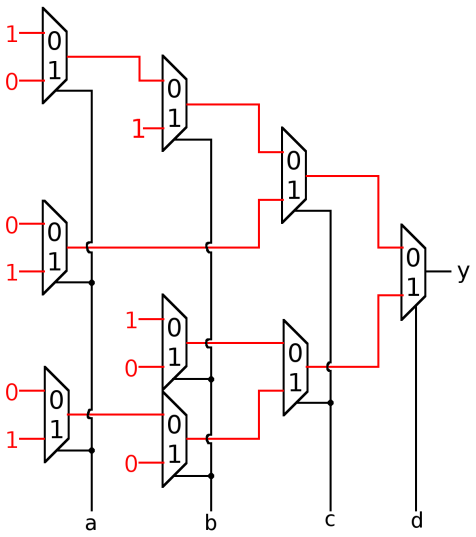


Abbildung 5: Multiplexerschaltung zur Stromversorgung der Lichtanlage

3.7 Der Zustand des Lichtes soll vier mal pro Sekunde aktualisiert werden. Hierfür soll ein Clock-Signal benutzt werden. Der zu diesem Zeitpunkt anliegende Zustand soll für die nächsten 0,25 Sekunden in einem D-Flipflop gespeichert werden. Zusätzlich sollen folgende Funktionen erfüllt sein:
Es werden vier Lampen angesteuert.

Lampe 1 (L_1) soll aktiv sein, wenn der Pegel des Flipflops *high* ist.

L_2 soll komplementär zu L_1 leuchten.

L_3 soll leuchten, wenn der Flipflop *high* ist und ein zusätzlicher Tastereingang auf *low*.

L_4 soll aktiv sein, wenn entweder der Taster auf *low* steht, der Flipflop auf *high*, oder beide Bedingungen erfüllt sind.

Erstellen Sie eine Gatterschaltung die die voranstehenden Funktionen erfüllt. Benutzen Sie hierfür nur ein D-Flipflop, ein Negierungsglied, ein NOR-Gatter und ein NAND-Gatter

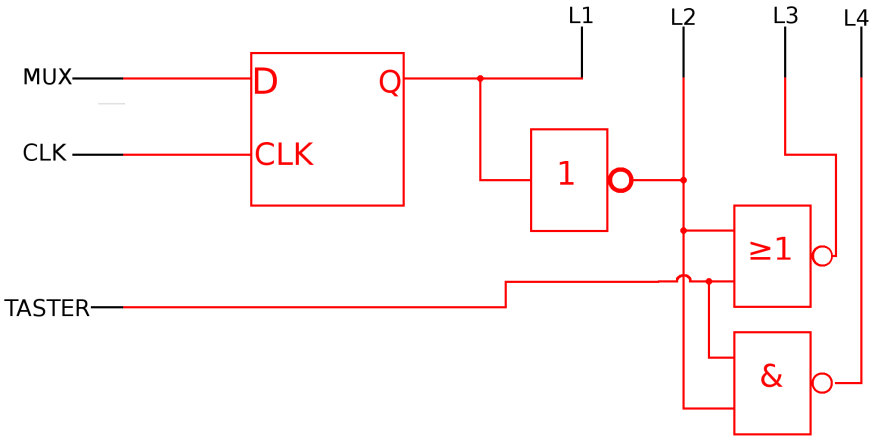


Abbildung 6: Kontakte der Gatterschaltung

3.8 Um eine schnelle und leistungsarme Schaltung zu gewährleisten soll die Gatterschaltung (ohne das FlipFlop) aus Aufgabe 3.6 in CMOS realisiert werden. Überführen Sie die Schaltung in ein CMOS-Schaltnetz.

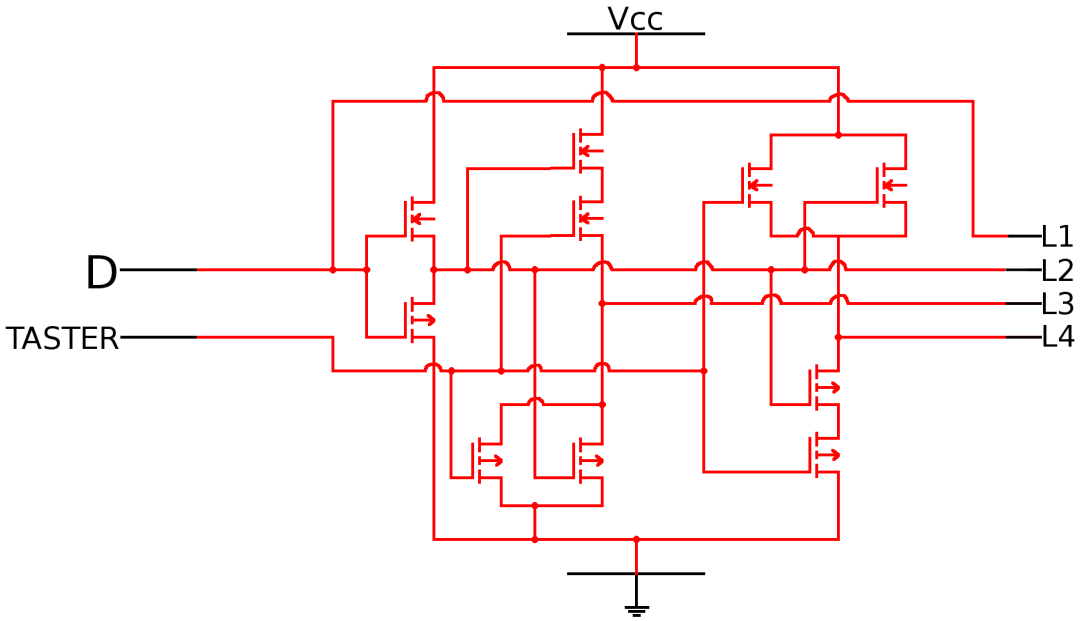


Abbildung 7: In CMOS überführte Gatterschaltung