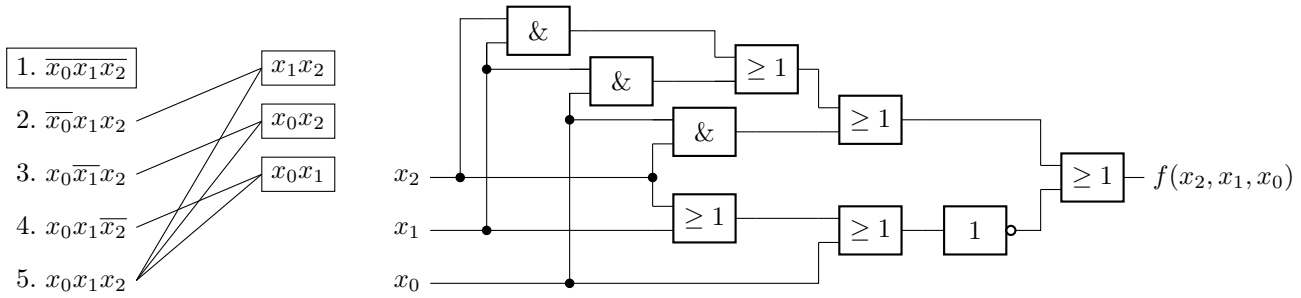
# Grundlagen der Rechnerarchitektur: Übungsblatt 6

Alexander Waldenmaier, Maryia Masla

18. Dezember 2020

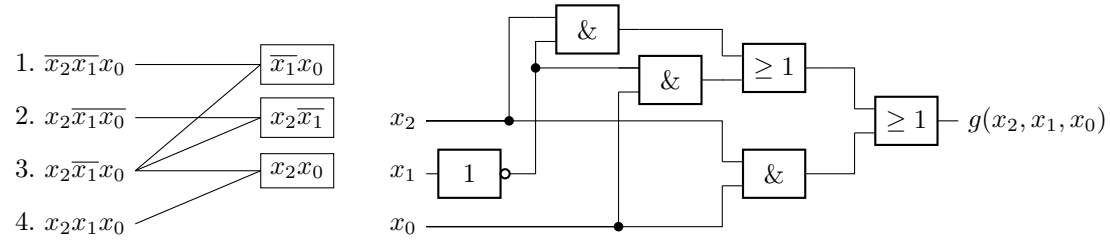
## Aufgabe 1: Quine McCluskey

a) Minimierung von $f$:




$\Rightarrow f(x_0, x_1, x_2) = \overline{x_0x_1x_2} + x_1x_2 + x_0x_2 + x_0x_1$

b) Minimierung von $g$:




$\Rightarrow g(x_0, x_1, x_2) = \overline{x_1}x_0 + x_2\overline{x_1} + x_2x_0$

## Aufgabe 2: Aiken-Code

a) Siehe Tabelle 1.

b)

$$y_1(x_1, x_2, x_3, x_4) = \overline{x_1}x_2\overline{x_3}x_4 + \overline{x_1}x_2x_3\overline{x_4} + \overline{x_1}x_2x_3x_4 + x_1\overline{x_2x_3x_4} + x_1\overline{x_2x_3}x_4$$
$$y_2(x_1, x_2, x_3, x_4) = \overline{x_1}x_2\overline{x_3x_4} + \overline{x_1}x_2x_3\overline{x_4} + \overline{x_1}x_2x_3x_4 + x_1\overline{x_2x_3x_4} + x_1\overline{x_2x_3}x_4$$
$$y_3(x_1, x_2, x_3, x_4) = \overline{x_1x_2}x_3\overline{x_4} + \overline{x_1x_2}x_3x_4 + \overline{x_1}x_2\overline{x_3}x_4 + x_1\overline{x_2x_3x_4} + x_1\overline{x_2x_3}x_4$$
$$y_4(x_1, x_2, x_3, x_4) = \overline{x_1x_2x_3}x_4 + \overline{x_1x_2}x_3x_4 + \overline{x_1}x_2\overline{x_3}x_4 + \overline{x_1}x_2x_3x_4 + x_1\overline{x_2x_3}x_4$$

c) Siehe Abbildung 1.

d) Da die Funtionen mittels KV minimiert wurden, liegen sie in orthogonaler Form vor. Daher können die OR-Verknüpfungen einfach dur XOR ersetzt werden. Anschließend müssen die Komplemente noch in

$y_1(x_1, x_2, x_3, x_4)$:

| $x_1x_3$ \ $x_2x_4$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | * | * | * | * |
| 10 | 1 | 1 | * | * |

$\Rightarrow y_1(x_1, x_2, x_3, x_4) = x_1 + x_2x_4 + x_2x_3$

$y_2(x_1, x_2, x_3, x_4)$:

| $x_1x_3$ \ $x_2x_4$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | * | * | * | * |
| 10 | 1 | 1 | * | * |

$\Rightarrow y_2(x_1, x_2, x_3, x_4) = x_1 + x_2\overline{x_4} + x_2x_3$

$y_3(x_1, x_2, x_3, x_4)$:

| $x_1x_3$ \ $x_2x_4$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | * | * | * | * |
| 10 | 1 | 1 | * | * |

$\Rightarrow y_3(x_1, x_2, x_3, x_4) = x_1 + x_2\overline{x_3}x_4 + \overline{x_2}x_3$

$y_4(x_1, x_2, x_3, x_4)$:

| $x_1x_3$ \ $x_2x_4$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | * | * | * | * |
| 10 | 0 | 1 | * | * |

$\Rightarrow y_4(x_1, x_2, x_3, x_4) = x_4$

Abbildung 1: Minimierte Schaltfunktionen

der Form $\overline{a} = a \oplus 1$ dargestellt werden.

$$
\begin{aligned}
y_1(x_1, x_2, x_3, x_4) &= x_1 \oplus x_2x_4 \oplus x_2x_3 \\
y_2(x_1, x_2, x_3, x_4) &= x_1 \oplus x_2\overline{x_4} \oplus x_2x_3 = x_1 \oplus x_2(x_4 \oplus 1) \oplus x_2x_3 \\
y_3(x_1, x_2, x_3, x_4) &= x_1 \oplus x_2\overline{x_3}x_4 \oplus \overline{x_2}x_3 = x_1 \oplus x_2(x_3 \oplus 1)x_4 \oplus (x_2 \oplus 1)x_3 \\
y_4(x_1, x_2, x_3, x_4) &= x_4
\end{aligned}
$$

Die Schaltfunktionen sind in Abbildung 2 dargestellt.

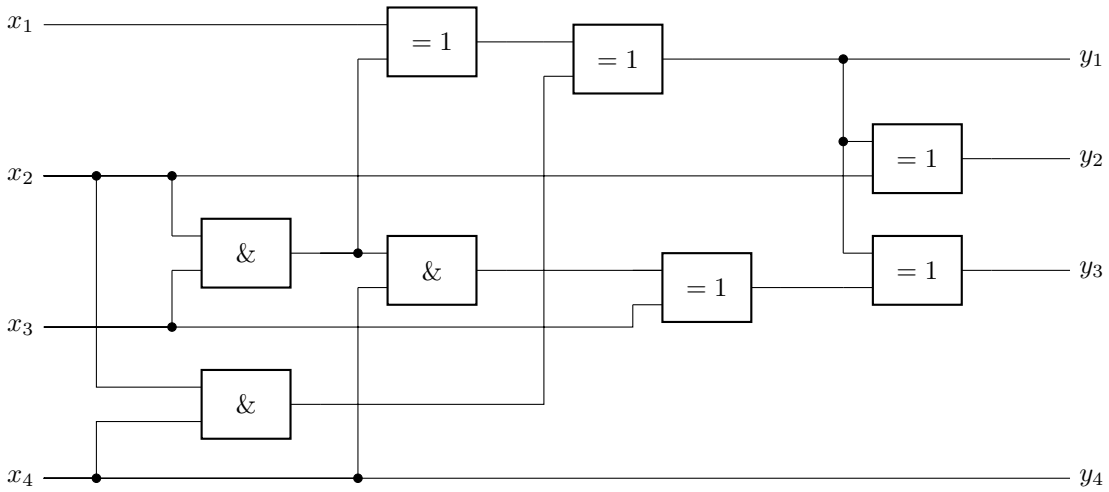



Abbildung 2: Aiken-Code Schaltung

| Dezimal | Binär | | | | Aiken-Code | | | |
|---|---|---|---|---|---|---|---|---|
| $d$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | n.d. | | | |
| 11 | 1 | 0 | 1 | 1 | n.d. | | | |
| 12 | 1 | 1 | 0 | 0 | n.d. | | | |
| 13 | 1 | 1 | 0 | 1 | n.d. | | | |
| 14 | 1 | 1 | 1 | 0 | n.d. | | | |
| 15 | 1 | 1 | 1 | 1 | n.d. | | | |

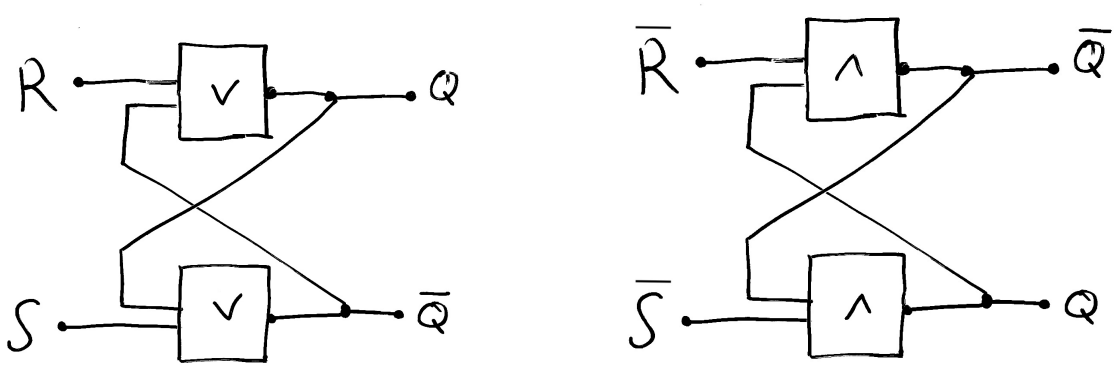
Tabelle 1: Vervollständigte Tabelle aus Aufgabe 2a)

## Aufgabe 3: RS-Flipflop

a) Der $R$-Eingang ist der „Reset“-Eingang. Wird dort eine logische 1 angelegt, stellt sich der Ausgang $Q$ stets auf 0 um - egal ob dort vorher eine 1 stand, oder nicht. Umgekehrt entsteht durch eine logische 1 an Eingang $S$ stets eine 0 am Ausgang $Q$, unabhängig davon, was dort vorher gespeichert war. Wird keiner der beiden Eingänge betätigt, bleibt der Ausgang in $Q$ gespeichert. Somit kann in einem RS-Flipflop 1 Bit Information dauerhaft gesichert werden.

Betägtigt man gleichzeitig zu $S = R = 1$, gelangt man in den „verbotenen“ Zustand. Im Falle eines NOR-RS-Flipflop entsteht dann an beiden Ausgängen eine logische 0, im Falle eines NAND-RS-Flipflop entsteht an beiden Ausgängen eine logische 1. Lässt man beide Eingänge im selben Moment los, gelangt man theoretisch in den „metastabilen“ Zustand. In der Praxis entsteht aufgrund minimaler Zeitdifferenzen ein beliebiger Ausgang - das Resultat ist undefiniert.

b)/c) Siehe Abbildung 3.




(a) NOR-RS-Flipflop (b) NAND-RS-Flipflop

Abbildung 3: RS-Flipflop

d) Siehe Abbildung 4.

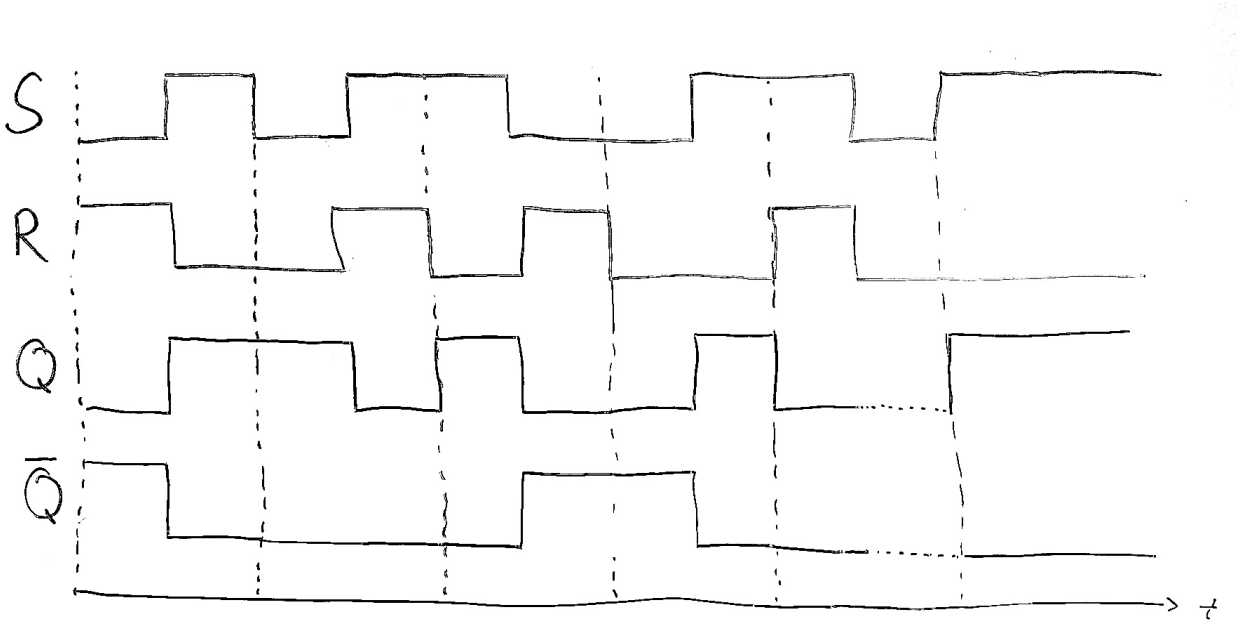
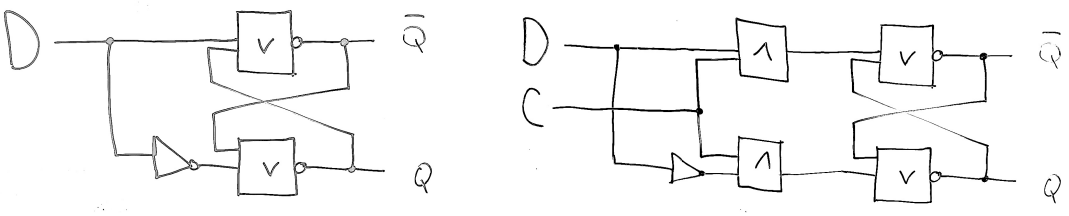



Abbildung 4: Signalverlauf RS-FF (Gezeigt ist der Verlauf eines NOR-RS-Flipflop)

## Aufgabe 4: D-FF und D-Latch

a) Siehe Abbildung 5




(a) D-Latch (b) Positiv taktzustandsgesteuerter D-Flipflop

Abbildung 5: D-Latch und D-FF

b) Beim D-Latch bzw. D-FF steht das „D“ für *Data*. Will man also eine 1 speichern, liegt an D eine 1 an - will man eine 0 speichern, liegt eine 0 an. Dadurch, dass nur dieser eine Input (und sein Komplement) vorliegt, kann auch nie der verbotene Zustand erreicht werden.

Der D-FF ist eine Erweiterung des D-Latch: Er erlaubt einen Wechsel des gespeicherten Zustands nur, wenn auch das Control-Signal „C“ aktiv ist. Ansonsten verharrt der D-FF im Hold-Zustand. Der Vorteil dabei ist, dass der D-Eingang nicht ständig auf der gewünschten 1 oder 0 gehalten werden muss, sondern beliebige Werte annehmen kann, solange nicht auch der Control-Eingang aktiviert wird. Im D-FF kann somit dauerhaft ein Zustand gespeichert werden und ggf. durch Aktivierung des Control-Eingangs verändert werden.

c) Siehe Abbildung 6.

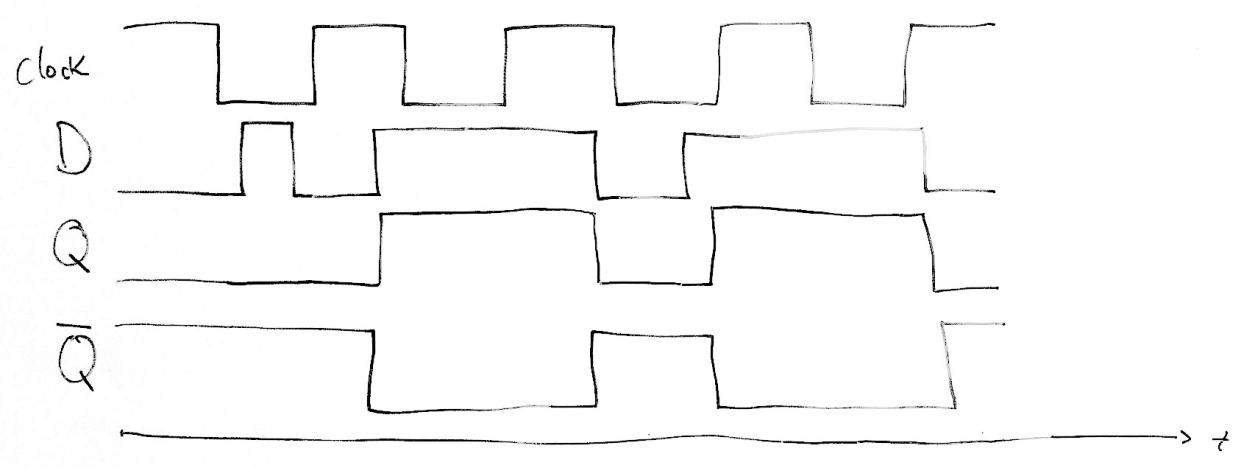



Abbildung 6: Signalverlauf D-FF