Digitaltechnik
Wintersemester 2017/2018
7. Übung

TECHNISCHE UNIVERSITÄT DARMSTADT

Andreas Engel, Raad Bahmani **LÖSUNGSVORSCHLAG** **KW49**

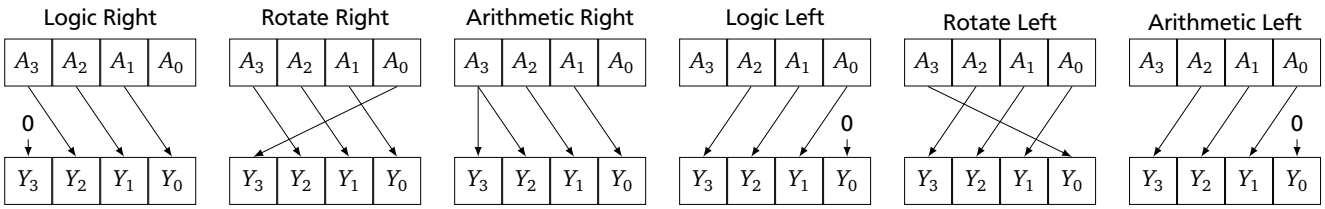
Die Präsenzübungen werden in Kleingruppen während der wöchentlichen Übungsstunde bearbeitet. Bei Fragen hilft Ihnen Ihr Tutor gerne weiter. Mit der angegebenen Bearbeitungszeit für die einzelnen Aufgaben können Sie Ihren Leistungsstand besser einschätzen.
Die mit **„Zusatzaufgabe“** gekennzeichneten Aufgaben sind zur zusätzlichen Vertiefung für interessierte Studierende gedacht und daher nicht im Zeitumfang von 90 Minuten einkalkuliert.

Übung 7.1 Barrel Shifter - **Zusatzaufgabe**

Shifter sind wichtige Bauteile vieler digitaler Systeme. Beim Shiften wird ein mehrere Bit breiter Wert um eine bestimmte Anzahl von Stellen verschoben. Neben dieser Schiebedistanz und der Schieberichtung (links oder rechts) unterscheidet man Shifter noch nach dem Konzept zum Auffüllen der „freigeschobenen“ Stellen:

- *logischer* Rechts- oder Linksshift: Auffüllen mit Nullen
- *umlaufender* Rechts- oder Linksshift: Auffüllen mit den aus der anderen Seite herausfallenden Bits (Rotation)
- *arithmetischer* Rechtsshift: Auffüllen mit Vorzeichen des als Zweierkomplement interpretierten Dateneingangs (entspricht Division durch $2^{\text{Schiebedistanz}}$)
- *arithmetischer* Linksshift: Auffüllen mit Nullen (entspricht Multiplikation mit $2^{\text{Schiebedistanz}}$)

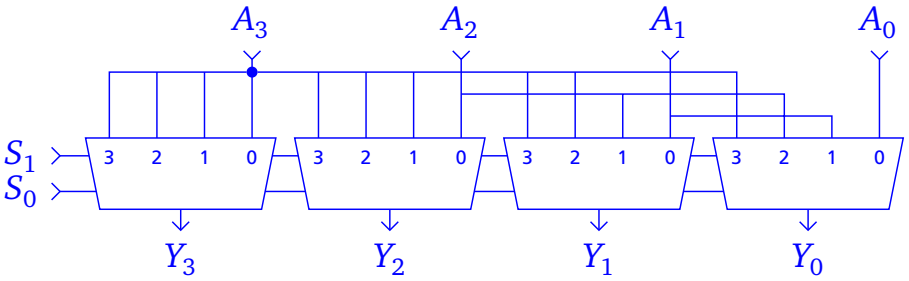
Im Folgenden sind diese Shifter für eine Schiebedistanz von 1 dargestellt:



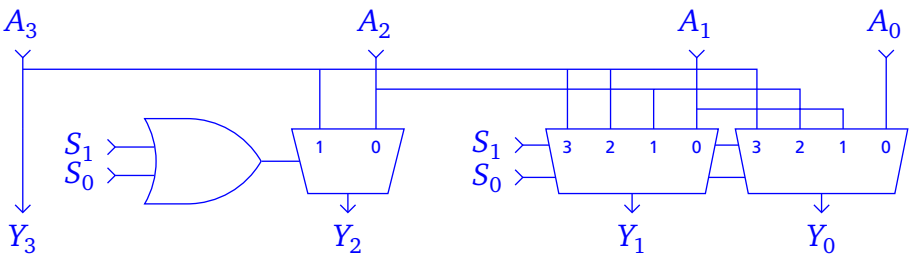
Unabhängig von Schieberichtung und Auffüllkonzept bezeichnet man einen Shifter auch als „Barrel-Shifter“, wenn seine Verzögerungszeit prinzipiell unabhängig von der Schiebedistanz ist.

Übung 7.1.1 Realisierung mit Basisgattern

Realisieren Sie den arithmetischen Rechtsshift AR4: $\mathbb{B}^6 \rightarrow \mathbb{B}^4$ mit jeweils 4 bit breiten Datenein- $(A_3,...,A_0)$ und Ausgängen $(Y_3,...,Y_0)$ sowie einem 2 bit breiten Selektor (S_0,S_1) für die Schiebedistanz zwischen 0 und 3. Verwenden Sie dafür ausschließlich die Basisgatter AND, OR, NOT, MUX und DECODE.
Eine direkte Realisierung mit Multiplexern:



Diese kann noch vereinfacht werden, da für Y_3 kein Multiplexer nötig ist, und für Y_2 nur aus zwei Werten gewählt werden muss:



Ein MUX4 kann mit drei MUX2 realisiert werden, und ein MUX2 besteht aus zwei AND2, zwei NOT und einem OR2. Daher benötigt die vereinfachte AR4-Schaltung 14 AND2, 14 NOT und acht OR2.

Übung 7.1.2 Zweistufige Optimierung

Optimieren Sie AR4 mit Espresso und zeichnen Sie die minimierte Schaltung. Vergleichen Sie die Anzahl der notwendigen AND2, OR2 und NOT Gatter dieser minimierten Schaltung mit Ihrer Lösung aus Übung 7.1.1.

Die Espresso-Repräsentation von AR4 mit sechs Eingängen und 4 Ausgängen lässt sich am einfachsten erstellen, in dem man zunächst die 16 Zeilen mit $S_1 = S_0 = 0$ generiert. Für diesen Block sind die Ausgabebits $(Y_3, ..., Y_0)$ identisch mit den vier Eingangsbits $(A_3, ..., A_0)$. Die drei weiteren Blöcke mit je 16 Zeilen erhält man jeweils durch Kopieren des vorherigen Blocks, Einfügen einer 1-Spalte vor dem Ausgangsblock, Löschen der letzten Ausgangsspalte, und Anpassen der Spalten für die Schiebeweite.

Die Minimierung dieser Funktion mit Espresso erzeugt dann folgende Datei:

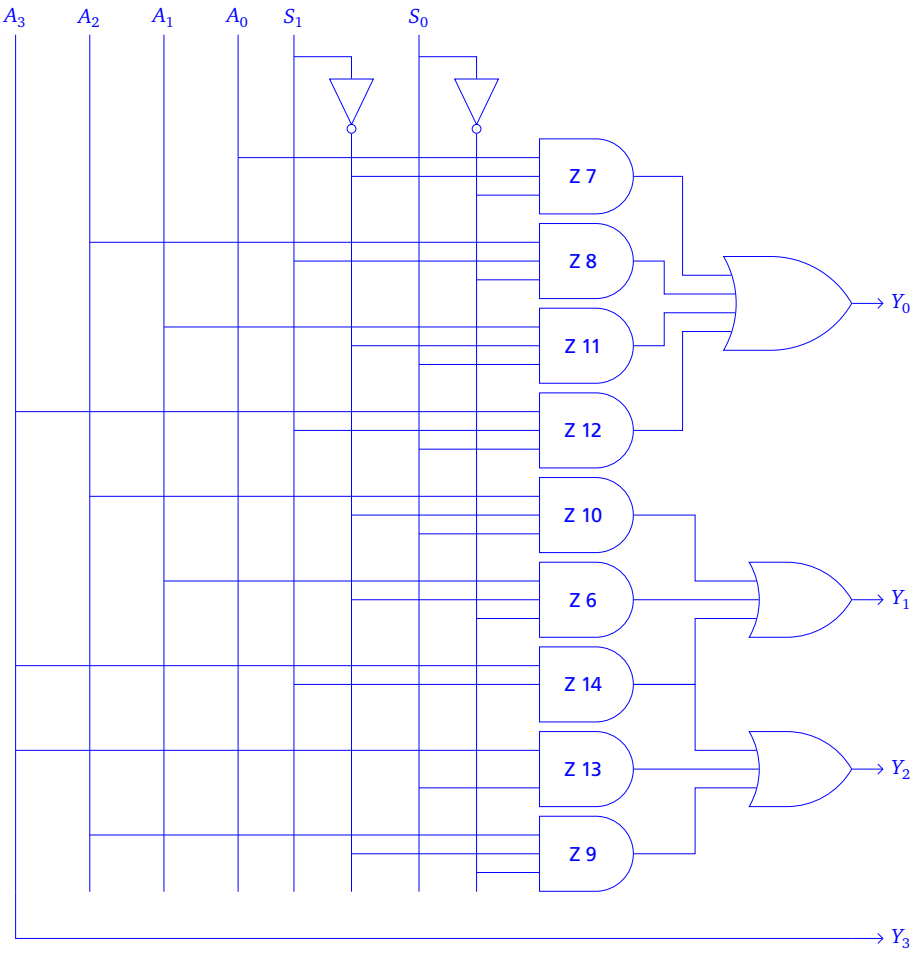
espresso AR4.esp

```
.i 6
.o 4
.ilb A3 A2 A1 A0 S1 S0
.ob Y3 Y2 Y1 Y0
.p 10
--1-00 0010
---100 0001
-1--10 0001
-1--00 0100
-1--01 0010
--1-01 0001
1---11 0001
1----1 0100
1---1- 0110
1----- 1000
.e
```

Aus dieser lassen sich die disjunktiven Ausdrücke der vier Ausgänge ablesen:

$$
\begin{aligned}
Y_3 &= A_3 \\
Y_2 &= A_3\, S_1 + A_3\, S_0 + A_2\, \overline{S_1}\, \overline{S_0} \\
Y_1 &= A_3\, S_1 + A_2\, \overline{S_1}\, S_0 + A_1\, \overline{S_1}\, \overline{S_0} \\
Y_0 &= A_3\, S_1\, S_0 + A_2\, S_1\, \overline{S_0} + A_1\, \overline{S_1}\, S_0 + A_0\, \overline{S_1}\, \overline{S_0}
\end{aligned}
$$

Y_2 und Y_1 teilen sich also einen gemeinsamen Implikanten, und von den Dateneingängen werden keine negierten Literale verwendet. In nachfolgender Schaltung sind die AND Gatter mit den zugehörigen Zeilennummern der Espresso- Ausgabe markiert:

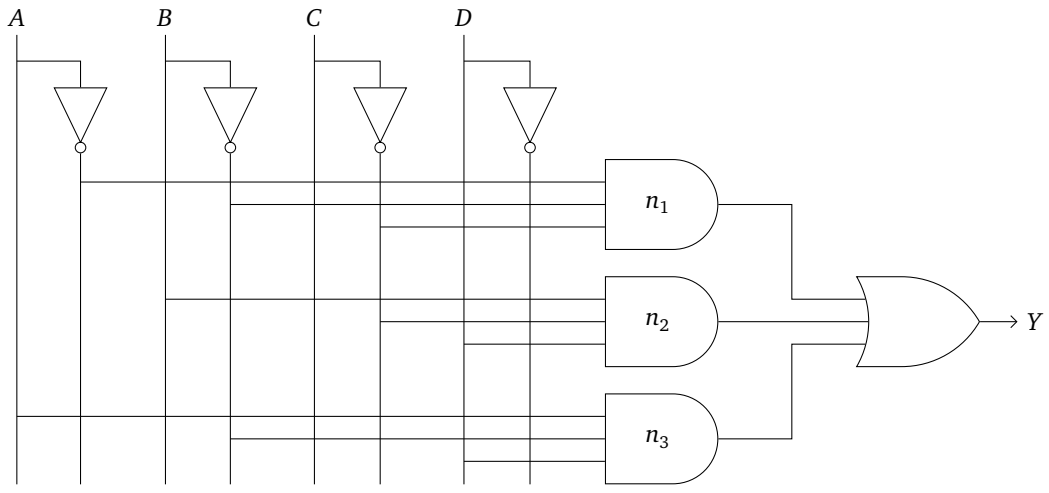


Da OR4 mit drei OR2, OR3 mit zwei OR2 und AND3 mit zwei AND2 Gattern realisiert werden kann, benötigt die zweistufige AR4 Realisierung 16 AND2, zwei NOT und sieben OR2 Gatter.
Beim Vergleich mit der Lösung aus Übung 7.1.1 ist zu beachten, dass Espresso (ohne weitere Konfigurationsangaben) nur zweistufig minimiert. Die Multiplexer-Realisierung ist aber mehrstufig, was potentiell effizientere Lösungen erlaubt.

Übung 7.2 Störimpulse - Wiederholung [20 min]

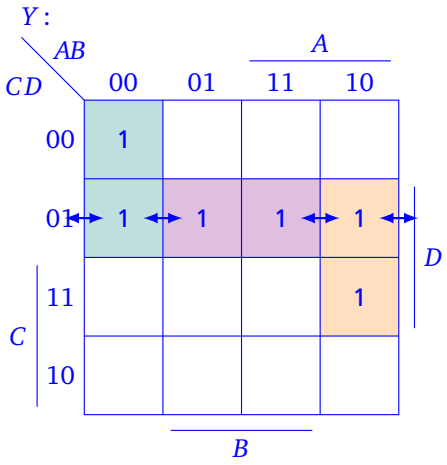
Gegeben sei die folgende Schaltung mit den Verzögerungszeiten

$$t_{\text{pd,NOT}} = t_{\text{cd,NOT}} = 1\,\text{ns}, \quad t_{\text{pd,AND3}} = t_{\text{cd,AND3}} = 2\,\text{ns} \quad \text{und} \quad t_{\text{pd,OR3}} = t_{\text{cd,OR3}} = 1\,\text{ns}$$

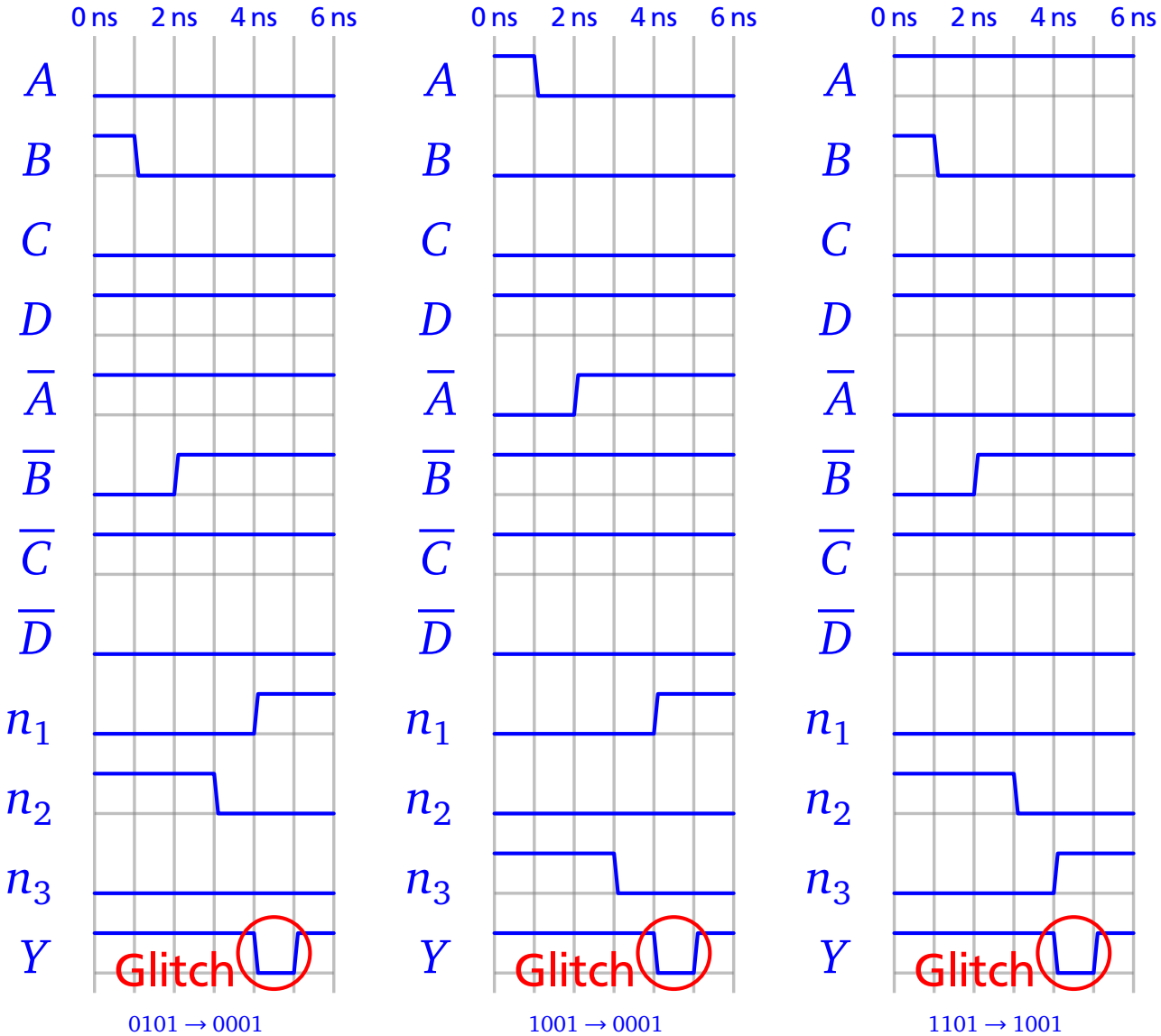


Übung 7.2.1 Störimpulse erkennen

Identifizieren Sie die kritischen Eingangstransitionen (mit nur einer geänderten Variable) im Karnaugh Diagramm, bei denen Störimpulse auftreten können. Verifizieren Sie anhand von Timing-Diagrammen, ob an diesen Stellen tatsächlich Störimpulse auftreten. Analysieren Sie dafür jeweils die *fallenden Flanken* der kritischen Übergänge.

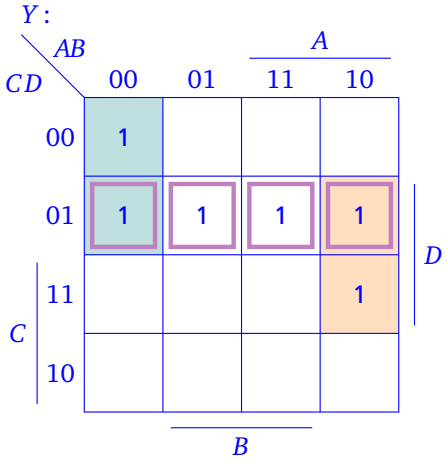


$$Y = (\overline{A}\,\overline{B}\,\overline{C}) + (B\,\overline{C}\,D) + (A\,\overline{B}\,D)$$

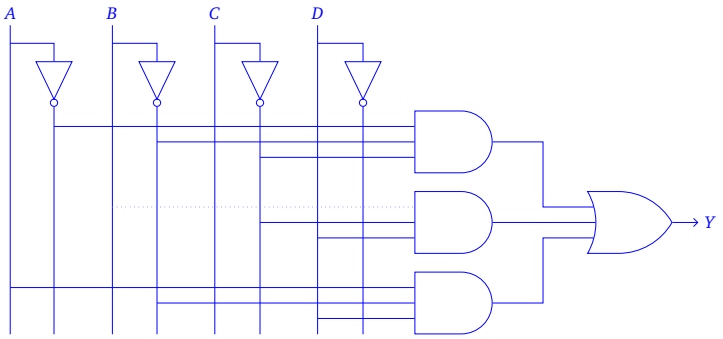


Übung 7.2.2 Störimpulse beheben

Geben Sie eine funktional äquivalente Schaltung an, die keine Störimpulse enthält. Versuchen Sie, dies mit möglichst wenigen Modifikationen der bestehenden Schaltung zu erreichen.

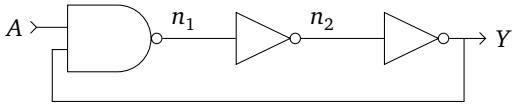


$$Y = (\overline{A}\,\overline{B}\,\overline{C}) + (\overline{C}\,D) + (A\,\overline{B}\,D)$$

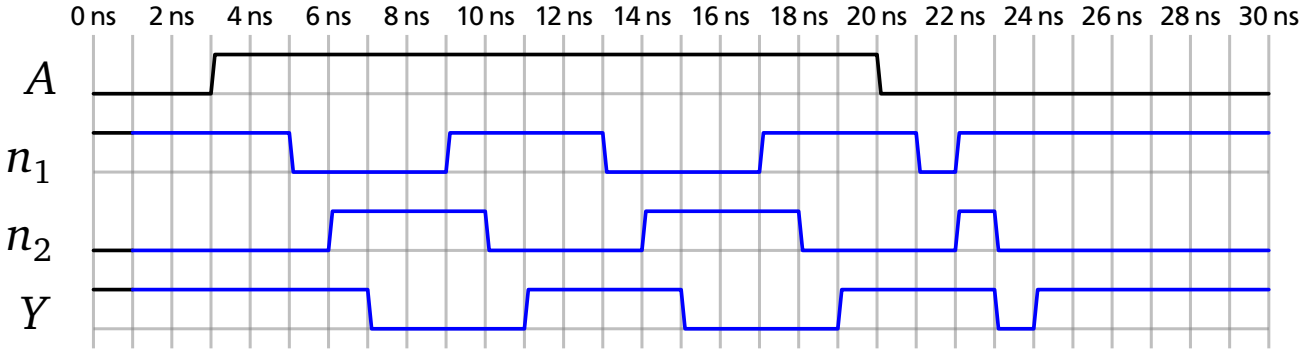


Übung 7.3 Instabile Schaltung [10 min]

Erweitert man die bistabile Grundschaltung um ein NAND Gatter, so erhält man eine Schaltung, die zwischen stabilem und instabilem Zustand wechseln kann:



Vervollständigen Sie das folgende Timing-Diagramm für $t_{\text{pd,NOT}} = t_{\text{cd,NOT}} = 1\,\text{ns}$, $t_{\text{pd,NAND}} = t_{\text{cd,NAND}} = 2\,\text{ns}$:

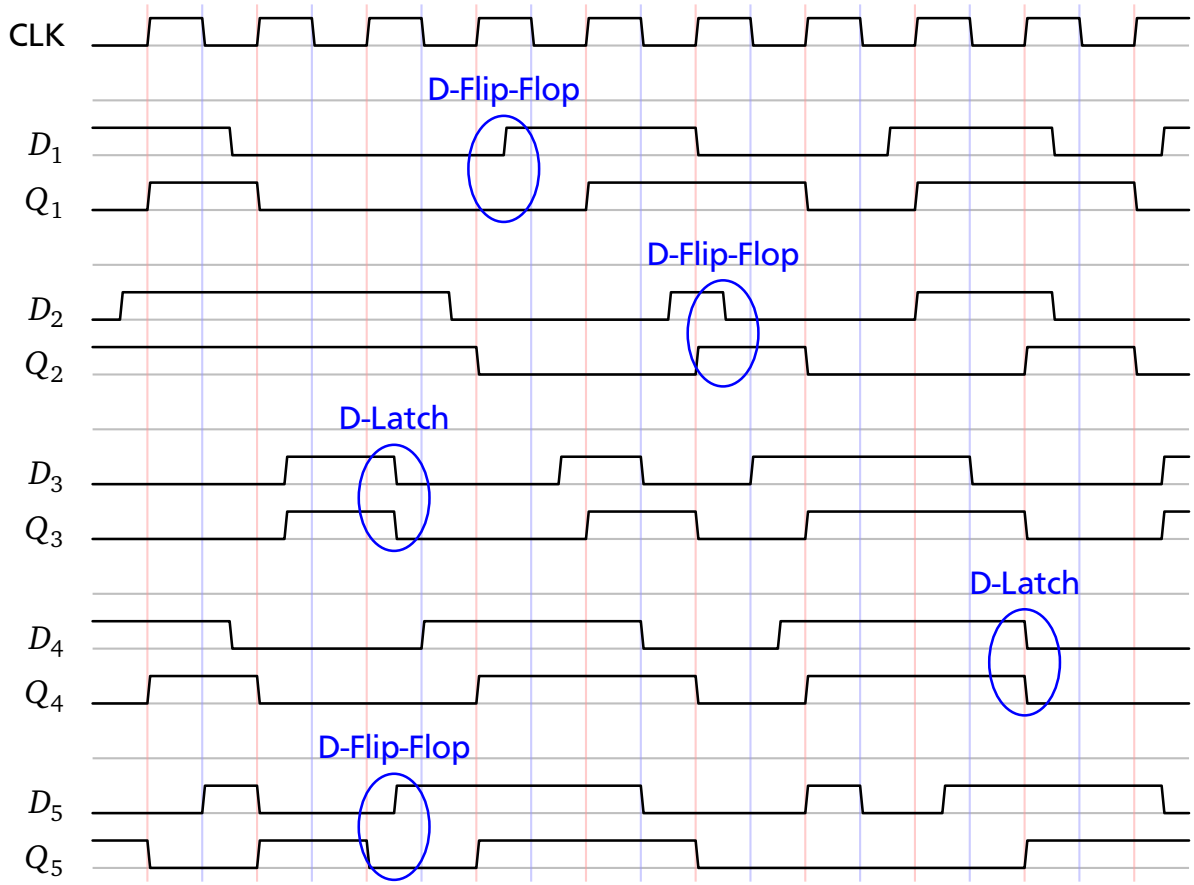


Wieviele steigende Flanken (Wechsel von 0 auf 1) erzeugt diese Schaltung im instabilen Zustand innerhalb von einer Sekunde?
Der Abstand zwischen zwei steigenden Flanken beträgt 8 ns. Das entspricht einem Takt von $\frac{1}{8\,\text{ns}} = 125\,\text{MHz}$.
Wieviele Inverter müssen noch in die Schaltung eingefügt werden, um einen 20 MHz Takt im instabilen Zustand zu generieren?
Ein 20 MHz Takt entspricht einem Abstand von $\frac{1}{20\,\text{MHz}} = 50\,\text{ns}$ zwischen zwei steigenden Flanken, also 25 ns zwischen zwei Flanken. Dem NAND Gatter müssen also 23 Inverter folgen. Demnach sind noch *21* Inverter einzufügen.

Übung 7.4 Flip-Flops und Latches [10 min]

Übung 7.4.1 Am Schaltverhalten erkennen

Folgendes Timing-Diagramm beschreibt das Schaltverhalten von fünf Speicherelementen mit Takteingang CLK, Dateneingang D_i und Datenausgang Q_i. Welche dieser Speicherelemente sind D-Flip-Flops, welche sind D-Latches? Begründen Sie ihre Antwort.

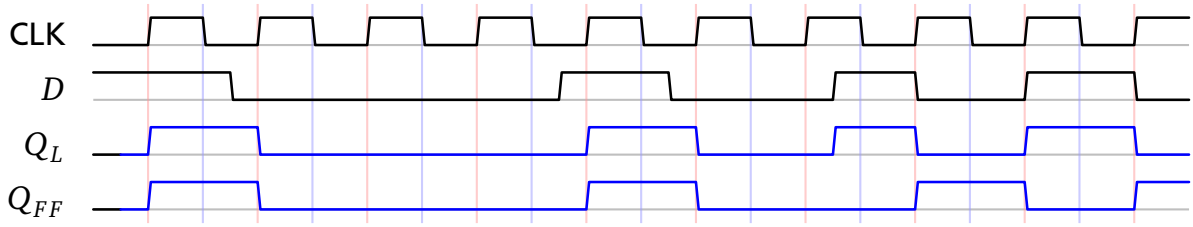


Ein D-Latch ist Takt-phasengesteuert, während ein Flip-Flop Takt-flankengesteuert ist. Das bedeutet, dass D-Flip-Flops nur zu steigender Taktflanke den Wert des Eingangs auf den Ausgang übernehmen. Ein D-Latch hingegen kann den Ausgang immer ändern, solange der anliegende Takt 1 ist. Im Timing-Diagramm ist für jedes Speicherelement beispielhaft je eine Stelle markiert, an der die Zuordnung festgemacht werden kann:

- $i \in \{1, 2, 5\}$: D_i wechselt mitten in 1-Phase der CLK, ohne dass dies von Q_i übernommen wird ⇒ D-Flip-Flop
- D_3 wechselt mitten in 1-Phase der CLK, was von Q_3 übernommen wird ⇒ D-Latch
- D_4 ist unmittelbar vor steigender Taktflanke 1, was von Q_4 aber nicht übernommen wird ⇒ D-Latch

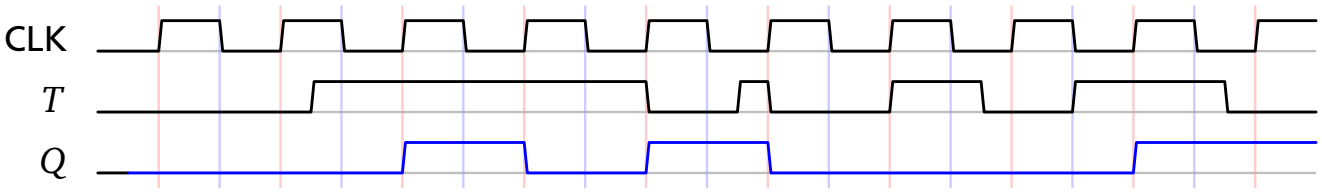
Übung 7.4.2 Schaltverhalten reproduzieren

Ergänzen Sie in folgendem Timing-Diagramm das Schaltverhalten eines D-Latches mit Ausgang Q_L und eines D-Flip-Flops mit Ausgang Q_{FF}. Beide Speicherelemente werden vom gleichen Takt- (CLK) und Datensignal (D) getrieben.



Übung 7.5 Toggle-Flip-Flops [30 min]

Ein Toggle-Flip-Flop hat neben dem Taktsignal nur einen weiteren Eingang T. Für $T = 1$ wechselt sein Zustand bei jeder steigenden Taktflanke. Für $T = 0$ bleibt der Zustand hingegen unverändert. Ergänzen Sie folgendes Timing-Diagramm für ein solches (flankengesteuertes) Toggle-Flip-Flop:



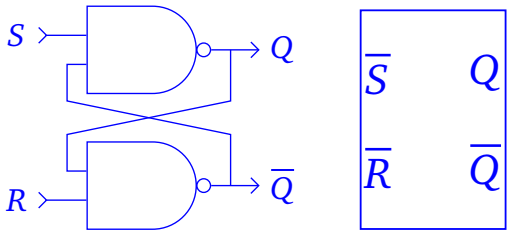
Realisieren Sie ein (flankengesteuertes) Toggle-Flip-Flop *ausschließlich mit NAND3* Gattern. Überlegen Sie dazu

- wie ein SR-Latch mit NAND Gattern aufgebaut werden kann
- wie dieses ausschließlich in der 1-Phase eines Taktsignals aktiviert werden kann
- wie Taktphasen-gesteuerte SR-Latches nach dem Master/Slave-Prinzip zu einem positiv Taktflanken-gesteuertem JK-Flip-Flop verschalten können
- wie ein JK-Flip-Flop zu einem T-Flip-Flop verschaltet werden kann
- wie Sie alle Basisgatter durch NAND3 Gatter ersetzen können

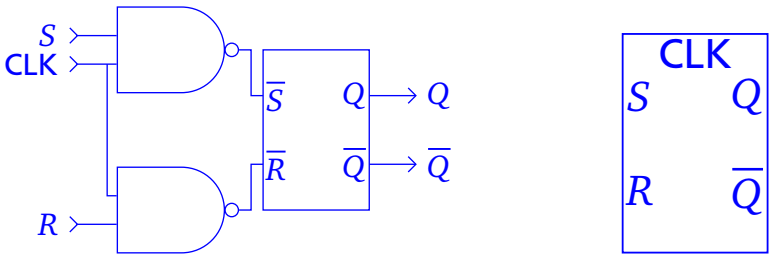
Folgende Schaltung realisiert ein SR-Latch mit NAND Gattern. Im Gegensatz zur NOR-Variante muss der S-Eingang an das obere (Q generierende) Gatter angelegt werden. Außerdem sind die S und R Eingänge jetzt low-active:

- $S = R = 1 \Rightarrow$ keine Änderung („latch“)
- $S = 0, R = 1 \Rightarrow$ setzen
- $S = 1, R = 0 \Rightarrow$ zurücksetzen
- $S = R = 0 \Rightarrow$ ungültig

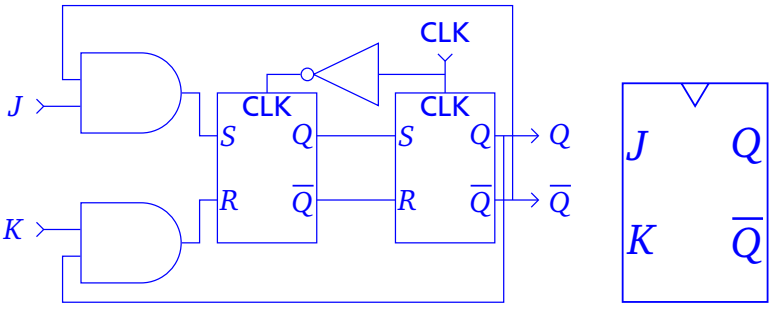
Dies wird durch ein entsprechend angepasstes Symbol für das SR-Latch verdeutlicht.



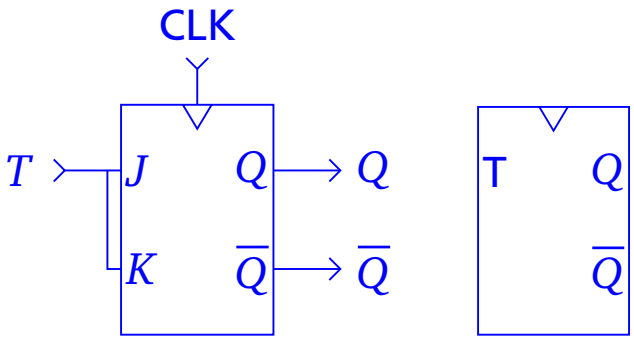
Analog zum D-Latch macht eine NAND-Verschaltung mit dem Taktsignal das SR-Latch Taktphasen-gesteuert. Gleichzeitig invertieren die NAND-Gatter die Eingänge, weshalb das Schaltsymbol für das getaktete SR-Latch wieder nichtnegierten Eingänge zeigt.



Kombiniert man das Master/Slave-Prinzip (zwei Latches mit invertiertem Taktsignal) mit der JK-Rückkopplung (zur Vermeidung ungültiger Zustände am SR-Latch), so erhält man ein JK-Flip-Flop.



Für $J = K = 1$ invertiert das JK-Flip-Flop seinen Zustand. Daher müssen beide Eingänge mit T beschaltet werden, um ein T-Flip-Flop zu erhalten.



Um alle Gatter durch NAND3 zu ersetzen, verwendet man folgende Äquivalenzen:

$$\overline{\text{CLK}} = \overline{\text{CLK CLK CLK}}$$
$$\overline{(T\ Q)\ \text{CLK}} = \overline{T\ Q\ \text{CLK}}$$
$$\overline{Q\ \text{CLK}} = \overline{Q\ Q\ \text{CLK}}$$

Das resultierende T-Flip-Flop: