

Versuch 2

Digitale Elektronik

Seit der zweiten Hälfte des 20. Jahrhunderts basiert jede Computertechnik auf einer digitalen Elektronik. In diesem Versuch sollen Sie deren Prinzipien anhand konkreter Schaltungen kennenlernen. Im ersten Teil bauen Sie dazu Logikgatter aus analogen Bausteinen auf, im zweiten Teil realisieren Sie logische Funktionen mittels integrierter Schaltkreise.

2.1 Boolesche Funktionen und logische Gatter

In der Vorlesung haben Sie die so genannte Boolesche Logik kennengelernt. In der Digitaltechnik werden Boolesche Ausdrücke bzw. Verknüpfungen verwendet, um komplexe logische Funktionen zu entwerfen. Darauf aufbauend werden einfache Gatter, die logische Verknüpfungen implementieren, zu Schaltungen zusammengesetzt, um eben diese Boolesche Funktionen in Hardware realisieren. Tabelle 2.1 zeigt Funktionstabellen und DIN-Symbole einiger einfacher Gatter.

Mit bestimmten Gruppen von Gattern lassen sich alle möglichen Booleschen Verknüpfungen realisieren. Solche Gruppen nennt man vollständig. NAND_2 , NOR_2 , AND_2 , OR_2 und NOT sind Beispiele für vollständige Gruppen von Gattern (Indizes entsprechen der Anzahl der Eingänge des Gatters).

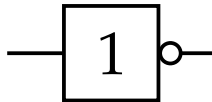
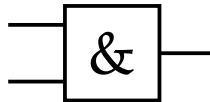
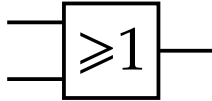
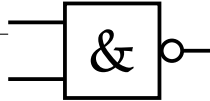
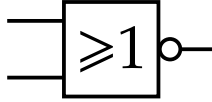
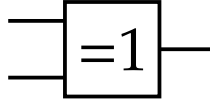
<table><tr><th>A</th><th>NOT (¬)</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	NOT (¬)	0	1	1	0		<table><tr><th>A</th><th>B</th><th>AND (∧)</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	AND (∧)	0	0	0	0	1	0	1	0	0	1	1	1										
A	NOT (¬)																																
0	1																																
1	0																																
A	B	AND (∧)																															
0	0	0																															
0	1	0																															
1	0	0																															
1	1	1																															
<table><tr><th>A</th><th>B</th><th>OR (∨)</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	OR (∨)	0	0	0	0	1	1	1	0	1	1	1	1		<table><tr><th>A</th><th>B</th><th>NAND (¬∧)</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	NAND (¬∧)	0	0	1	0	1	1	1	0	1	1	1	0	
A	B	OR (∨)																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	1																															
A	B	NAND (¬∧)																															
0	0	1																															
0	1	1																															
1	0	1																															
1	1	0																															
<table><tr><th>A</th><th>B</th><th>NOR (¬∨)</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	NOR (¬∨)	0	0	1	0	1	0	1	0	0	1	1	0		<table><tr><th>A</th><th>B</th><th>XOR (⊕)</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	XOR (⊕)	0	0	0	0	1	1	1	0	1	1	1	0	
A	B	NOR (¬∨)																															
0	0	1																															
0	1	0																															
1	0	0																															
1	1	0																															
A	B	XOR (⊕)																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	0																															

Tabelle 2.1: Beispiele für logische Gatter.

2.2 Diskrete Transistorlogik

Aus der Vorlesung wissen Sie, dass voll ausgesteuerte Transistoren als Schalter verwendet werden können. Durch Kombination solcher Schalter lassen sich dann logische Gatter realisieren. Die ersten Gatter wurden mit Widerstands-Transistor-Logik oder Diode-Transistor-Logik verwirklicht, seit den 1960er Jahren kommt aber überwiegend nur noch die so genannte Transistor-Transistor-Logik (TTL) zum Einsatz. Bei TTL werden als aktives Bauelement planare npn-Bipolartransistoren verwendet. Bausteine in TTL-Technik sind relativ unempfindlich gegen elektrostatische Entladungen, haben aber einen hohen Stromverbrauch.

Gatter und logische Schaltungen lassen sich statt mit Bipolartransistoren auch mit Feldeffekttransistoren, genauer gesagt mit MOS-FETs, realisieren. Die ersten Schaltungstechniken mit MOS-FET verwendeten ausschließlich p-Kanal-MOS-FETs (PMOS) oder n-Kanal-MOS-FETs (NMOS). Diese Schaltungstechniken wurden zwar für viele Digitalschaltungen eingesetzt, es entstanden jedoch keine eigentlichen Logikfamilien. Stattdessen wird heute üblicherweise die so genannte CMOS-Technik eingesetzt. Diese basiert auf der Paarung komplementärer (n- und p-Kanal) MOS-FETs und zeichnet sich durch einen sehr geringen statischen Stromverbrauch aus. Abbildung 2.1 zeigt einen Inverter in CMOS-Technik.

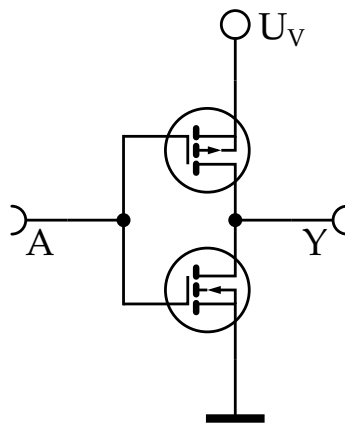
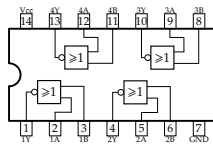
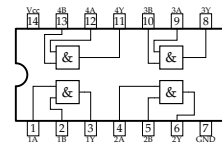
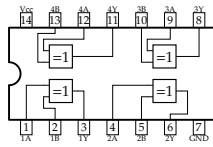


Abbildung 2.1: Inverter diskret aufgebaut mit MOS-FETs.

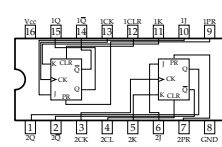
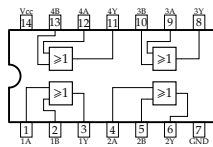
2.3 Logikfamilien

Elektronische Bausteine, die logische Gatter oder komplexere logische Schaltungen mit der gleichen Schaltungstechnik realisieren und ähnliche elektrische und physikalische Eigenschaften haben, werden in der Digitaltechnik als Logikfamilie bezeichnet. Am häufigsten werden heute Logikbausteine der sogenannten 74er-Reihe eingesetzt. Diese wurde in den 1970er-Jahren von Texas Instruments entwickelt, aber schon bald von anderen Herstellern nachgebaut. Der erste und bekannteste Vertreter der Reihe ist ein 4-fach-NAND-Gatter mit jeweils zwei Eingängen (siehe auch Tab. 2.2). Die Reihe wird immer noch erweitert, aber auch die frühen Typen werden noch hergestellt.

Auch die Nummerierung von Texas Instruments hat sich als Industriestandard durchgesetzt. Dessen 4-fach-NAND-Gatter etwa wurde SN7400 genannt, bei den nachfolgenden wurde die Nummer hochgezählt. Andere Hersteller ersetzen lediglich das Präfix SN durch eigene Buchstabenfolgen, beispielsweise DM7400 bei National Semiconductor. 7400 oder kurz '00 wurde so zur herstellerunabhängigen Bezeichnung für vier NAND-Gatter. Bausteine unterschiedlicher Hersteller können sich allerdings in der maximalen Taktfrequenz oder anderen Daten unterscheiden, sie sind daher nicht immer ohne weiteres austauschbar.

4 NOR₂ Gatter**02**Logikfunktion:
 $Y = A \vee B$ 4 AND₂ Gatter**08**Logikfunktion:
 $Y = A \wedge B$ 4 XOR₂ Gatter**86**Logikfunktion:
 $Y = A \vee B$ 

2 JK-FlipFlops

274 OR₂ Gatter**32**Logikfunktion:
 $Y = A \vee B$ 

6 NOT Gatter

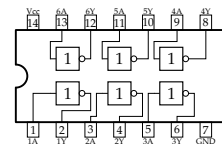
69Logikfunktion:
 $Y = \bar{A}$ 

Tabelle 2.2: Liste typischer Logikbausteine (herstellernabhängige Typenbezeichnung).

Die Original-Baureihe hat mittlerweile stark an Bedeutung verloren und wird kaum noch eingesetzt. Stattdessen wurden verschiedene 74xx-Logikfamilien entwickelt, die vor allem hinsichtlich Geschwindigkeit und Stromverbrauch verbessert wurden, die Logikfunktionen dagegen sind unverändert. Diese Reihen werden durch bis zu fünf zusätzliche Buchstaben zwischen „74“ und „xx“ gekennzeichnet, beispielsweise 74LS00 für einen in Low-Power-Schottky-Technik gefertigten 4-fach-NAND-Baustein.

2.4 Signallaufzeiten

Die Zeit, die ein einzelnes Gatter benötigt, um den Ausgang entsprechend der Eingangssignale zu schalten, nennt man Signallaufzeit. Bei Schaltungen aus mehreren Gattern addieren sich diese Signallaufzeiten naturgemäß.

Ist die Signallaufzeit durch die Schaltung vor einem Eingang eines Gatters länger als die Laufzeit durch die Schaltung vor dem anderen Eingang, so kann der Ausgang dieses Gatters kurzzeitig falsche Werte annehmen. Dieses Phänomen wird als *Hazard* bezeichnet. Man unterscheidet dabei zwischen dynamischen und statischen Hazards. Ein statischer Hazard entsteht, wenn die Änderung an den Eingängen unter idealen Bedingungen keine Änderung am Ausgang bewirken würde. Ein dynamischer Hazard ist entsprechend ein Störimpuls am Ausgang, der einem gewollten Pegelwechsel vorausgeht.

Um Hazards zu vermeiden, muss man dafür sorgen, dass bei jedem Gatter die Signallaufzeiten aller Eingänge gleich sind. Das kann man beispielsweise erreichen, indem man die gewünschte Funktion in die disjunktive oder konjunktive Normalform bringt und ohne Optimierungen aufbaut. Alternativ kann man durch eine „Taktung“ dafür sorgen, dass die Eingangssignale erst zu einem bestimmten Zeitpunkt ausgelesen werden. Dabei werden die Eingänge des Gatters abgekoppelt und erst durch das Taktsignal durchgeschaltet. Zum Freischalten dient entweder der Pegel oder eine Flanke des Taktsignals.

2.5 Flipflops

Flipflops sind Schaltungen, die einen logischen Zustand speichern können. Sie erlauben insbesondere einen definierten Zustandswechsel und ermöglichen dadurch den Entwurf von sequentiellen Schaltungen bzw. Schaltwerken. Flipflops können als 1-Bit-Speicher verwenden.

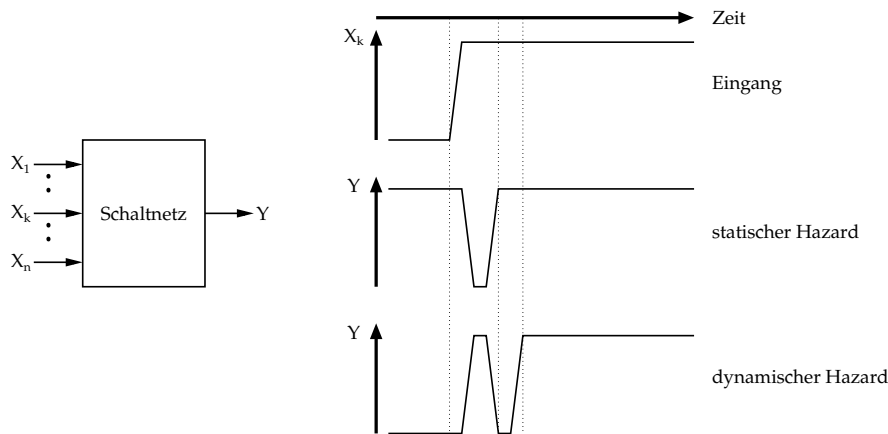


Abbildung 2.2: Dynamische und statische Hazards

det werden, aber auch zu Zählern oder Schieberegistern kombiniert werden. Das einfachste Flipflop auf Basis digitaler Logikbausteine besteht aus zwei rückgekoppelten NAND-Gattern und wird als RS-Flipflop bzw. RS-Latch bezeichnet (Abbildung 2.3a).

In der Literatur bzw. in Datenblättern wird teils zwischen Latch und Flipflop unterschieden. Dann ist ein Latch zustandsgesteuert und ein Flipflop flankengesteuert. Zustandsgesteuert bedeutet, dass der gespeicherte Wert entweder über den gesamten High-Pegel oder Low-Pegel seinen Zustand ändern kann. Flankengesteuert bedeutet, dass der gespeicherte Wert nur bei Wechsel des Taktes von 0 auf 1 oder von 1 auf 0 geändert werden kann. Ein flankengesteuertes Flipflop kann durch Hintereinanderschalten zweier RS-Latches realisiert werden (Abbildung 2.3b). Dieser Aufbau wird Master-Slave-Flipflop genannt.

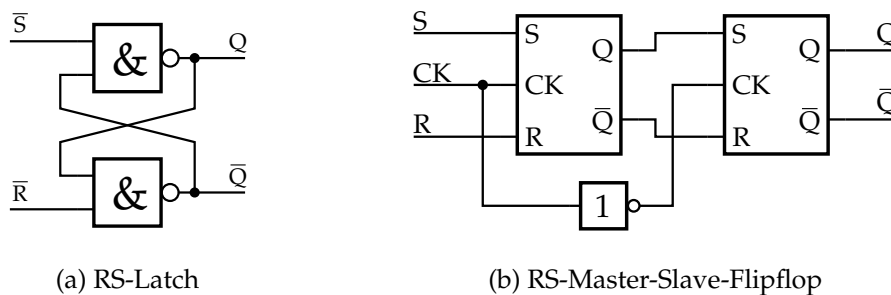


Abbildung 2.3: Beispiele für RS-Flipflops.

2.6 Weiterführende Literatur

Bei diesem Kapitel handelt es sich um eine Zusammenfassung, die Sie in den Themenkomplex einführen soll. Zusätzlich werden folgende Materialien zum Selbststudium empfohlen:

- Wolfram Schiffmann, Robert Schmitz. Technische Informatik 1 - Grundlagen der digitalen Elektronik. Springer, Berlin, 5. Auflage, 2004. (Kapitel 3, 4 und 5)
- Ulrich Tietze, Christoph Schenk, Eberhard Gamm. Halbleiter-Schaltungstechnik. Springer Vieweg, Berlin, 15. Auflage, 2016. (Kapitel 6)

2.7 Versuche

Hinweis: Die folgenden Aufgaben sollen mit der Anwendung QUCS umgesetzt werden.

1. Inverter aus diskreten Bauteilen

Setzen Sie die Schaltung aus Abbildung 2.1 in QUCS um. Überprüfen Sie die Funktionalität des Inverters, in dem Sie Eingangs- und Ausgangsspannung messen und die Wahrheitswertetabelle ableiten.

Hinweis: Verwenden Sie, wie in der Einführung gezeigt, einen Parameterdurchlauf, um alle Eingangskombinationen zu erzeugen.

2. AND-Gatter aus diskreten Bauteilen

Bereiten Sie eine CMOS-Schaltung für die AND-Funktion vor. Setzen Sie diese in QUCS um, messen Sie Eingangs- und Ausgangsspannungen aller Kombinationsmöglichkeiten und leiten Sie die Wahrheitswertetabelle ab.

Hinweis: Verwenden Sie, wie in der Einführung gezeigt, einen Parameterdurchlauf, um alle Eingangskombinationen zu erzeugen.

3. Logikfunktion

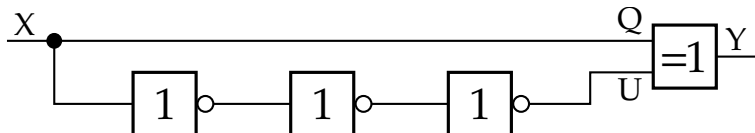
Gegeben Sei folgende Logikfunktion:

$$f(C, B, A) = \overline{C}B \vee C\overline{B} \vee \overline{A}$$

Stellen Sie für f die Logiktable auf und entwerfen Sie eine dazugehörige Schaltung. Setzen Sie diese Schaltung in QUCS um und überprüfen Sie die Funktionsweise, in dem Sie die Ergebnisse der Simulation Ihren Erwartungen gegenüberstellen.

4. Hazards

Gegeben Sei eine Logikschaltung aus drei Invertern und einem XOR-Gatter.



(a) Analysieren Sie die Schaltung mittels Impulsdiagramm auf Hazardfehler. Dabei sei X ein Rechtecksignal mit einer Frequenz von $0,5\text{ GHz}$. Nehmen Sie außerdem eine zeitliche Verzögerung von 20 ps für Inverter und 10 ps für die XOR-Gatter an. Kabelverbindungen sind ohne Laufzeiten.

(b) Setzen Sie die Schaltung in QUCS um und überprüfen Sie, ob Ihre Erwartungen aus Aufgabe 4.(a) stimmen. Nutzen Sie hierfür ein U-t-Diagramm.

Hinweis: Die Verzögerungszeiten müssen in den Eigenschaften der Logikgatter eingegeben werden.