Versuch 3

Entwurf digitaler Schaltungen

Aus den vorherigen Versuchen kennen Sie Grundgatter, die in separate Siliziumchips integriert sind. Durch Kombination solcher Bauelemente können komplexere logische Funktionen wie Addition oder Multiplikation realisiert werden. Diese Art digitale Schaltungen umzusetzen, ist jedoch vergleichsweise aufwendig. In industriellen Anwendungen wird daher heute häufig ein spezialisierter hochintegierter Schaltkreis (engl. application-specific integrated circuit, ASIC) entwickelt und in großer Stückzahl produziert.

Ein anderer weit verbreiteter Ansatz zum Entwurf digitaler Schaltungen ist die Verwendung eines frei konfigurierbaren Chips, der seine interne Verschaltung nach Vorgabe über eine Beschreibung jederzeit verändern kann. Diesen Ansatz werden Sie in diesem Versuch kennenlernen. Für einen solchen Chip sollen Sie in diesem Versuch eine komplexere Schaltung entwerfen. Diese kann dann auf einen rekonfigurierbaren Chip geladen und ausgeführt werden.

3.1 7-Segment-Anzeige

Die 7-Segment-Anzeige dient der Darstellung von Hexadezimal-Ziffern. Sie besteht aus sieben LEDs, die entweder einen gemeinsamen Anoden- oder einen gemeinsamen Kathodenanschluss besitzen. Handelt es sich dabei um die gemeinsame Kathode, so nennt man die LEDs auch high-aktiv, da man über die Anode (positiver Anschluss) das Leuchten ansteuert. Bei der gemeinsamen Anode werden sie entsprechend low-aktiv genannt. Die Abbildung 3.1 zeigt eine 7-Segment-Anzeige. Auf der linken Seite ist dabei die Anschlusskodierung zu sehen. Die rechte Seite stellt die übliche Umsetzung der Hexadezimalzahlen dar.

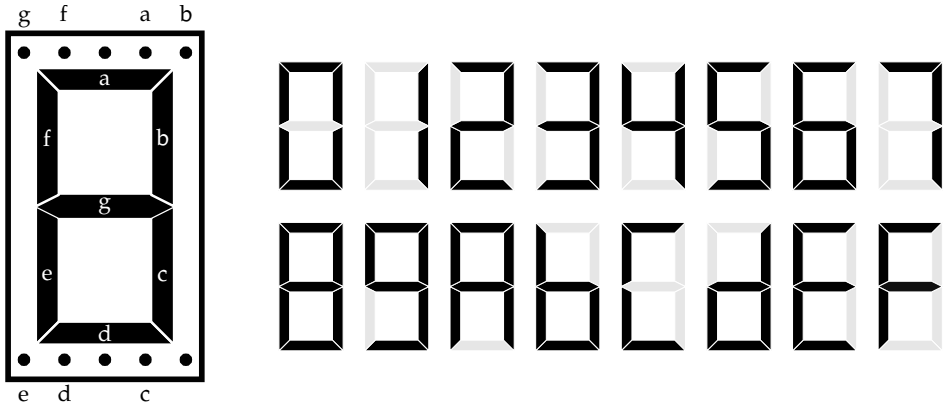


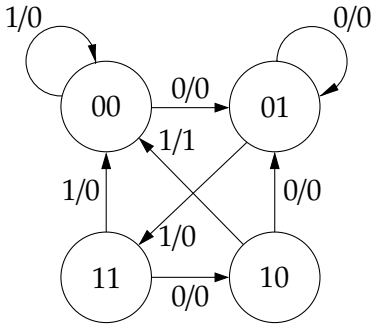
Abbildung 3.1: 7-Segment-Anzeige mit der Pinbelegung und allen darstellbaren Hexadezimal-Ziffern. Nicht bezeichnete Pins sind mit Ground zu verbinden.

3.2 Automaten

In der Informatik versteht man unter einem Automaten eine digitale, zeitdiskrete Modellmaschine, die auf eine Eingabe reagiert und eine Ausgabe produziert. In der theoretischen Informatik bzw. in der Automatentheorie ist es dabei unerheblich, ob eine solche Maschine tatsächlich gebaut werden kann. Vielmehr soll das Verhalten solcher Modelle durch Vereinfachung ihrer Fähigkeiten leichter verständlich und vergleichbar gemacht werden. Endliche Automaten sind Automaten, die lediglich eine endliche Menge an Zuständen besitzen. In der technischen Informatik sind endliche Automaten ein häufig genutztes Werkzeug für den Entwurf digitaler Schaltungen. Dabei wird zwischen den zwei Typen Mealy- und Moore-Automat unterschieden.

z_1^t	z_0^t	E	z_1^{t+1}	z_0^{t+1}	A
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	0	0	0

(a) Zustandsfolgetabelle



(b) Zustandsübergangsgraph

Abbildung 3.2: Repräsentationen eines Beispiel-Automaten.

3.3 Schaltwerke

Schaltwerke sind die praktische Umsetzung theoretischer Automaten. Dabei werden die Zustände durch Flipflops erzeugt. Die Zustandsübergangsfunktion und die Ausgabefunktion sind einfache Logikschaltungen (Schaltnetze). Die Abbildung 3.3 zeigt einen schematische Darstellung eines Mealy-Automaten, erkennbar an der Verbindung von Eingabe an die Ausgabefunktion. Hängt die Ausgabe nicht von der Eingabe ab, so liegt ein Moore-Automat vor. Liegt keine Ausgabefunktion vor (Ausgabe entspricht den Flipflop-Ausgängen), so wird er Medvedev-Automat genannt, welcher eine Sonderform der Moore-Automaten darstellt.

Für die technische Umsetzung lässt sich die Anzahl der benötigten Flipflops wie folgt bestimmen: Sei n_Z die Anzahl der Zustände des Automaten, dann ist die Anzahl der Flipflops

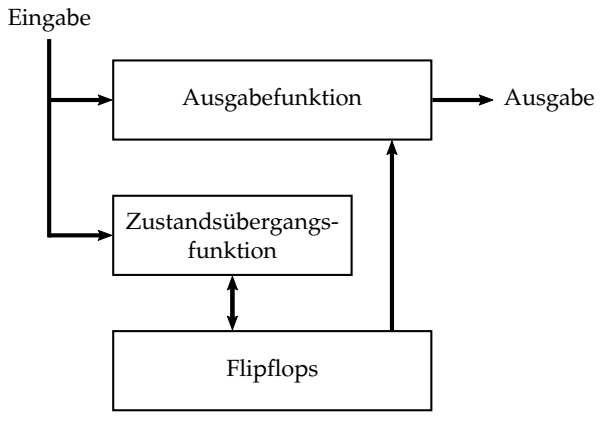


Abbildung 3.3: Schematische Darstellung des Mealy-Automaten.

$n_F = \lceil \log_2(n_Z) \rceil$ (die oberen Gaußklammern $\lceil \ \rceil$ runden das Ergebnis auf die nächste Ganzzahl auf). Ferner lässt sich dadurch sagen, dass ein Automat mit n_F Flipflops maximal 2^{n_F} Zustände besitzen kann.

Bei der Erstellung der Logikschaltung für die Zustandsübergangsfunktion muss die Art der Flipflops beachtet werden. So klingen Beispielsweise D-Flipflop und T-Flipflop sehr ähnlich, jedoch unterscheiden Sie sich in ihrer Funktionsweise signifikant. Bei größeren Automaten sollte außerdem darauf geachtet werden, dass die Zustandsübergangsfunktion frei von Hazards ist, da ansonsten der Automat in falsche, bzw. fehlerhafte Zustände wechseln kann. Hierfür können die in der Vorlesung eingeführten Normalformen DNF oder KNF verwendet werden.

Bei der Ausgabefunktion hängt es von der Anwendung ab, ob diese Hazards erlaubt. Hängen zum Beispiel an der Ausgabe nur LEDs, so fallen Schaltfehler nicht auf, da sich diese maximal durch ein sehr kurzes Flackern bemerkbar machen. Ist die Anzahl der Eingänge der Ausgabefunktion kleiner als die Anzahl der Ausgänge (z.B.: die Eingänge sind die Ausgänge von vier Flipflops (vier Bits) und die Ausgabe ist für eine 7-Segment-Anzeige (sieben Bits) ausgelegt) so spricht man von einem Decoder. Die wenigen Bits enthalten also mehr Information, als sie eigentlich darstellen können. Deshalb muss diese Information erst entschlüsselt (decodiert) werden. Ist die Anzahl der Eingänge größer als die Anzahl der Ausgänge, so handelt es sich um einen Encoder. Man verschlüsselt (encodiert) die Informationen in wenigen Bits.

3.4 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 6)
- Ulrich Tietze, Christoph Schenk, Eberhard Gamm. Halbleiter-Schaltungstechnik. Springer Vieweg, Berlin, 15. Auflage, 2016. (Kapitel 7, 8 und 9)

3.5 Versuche

Hinweis: Die folgenden Aufgaben sollen mit der Anwendung QUCS umgesetzt werden.
Hinweis: Das Resultat der folgenden Aufgaben ist eine komplexere Schaltung. Nutzen Sie für Ihre Messpunkte passende Bezeichnungen, um die Übersichtlichkeit zu wahren.

1. **Binärzähler**
   Es soll ein Schaltwerk entwickelt werden, welches die Zahlen von 0 bis 5 hoch zählt und danach wieder auf 0 zurücksetzt.

   (a) Entwickeln Sie einen Moore-Automaten $\mathcal{M} = (X,Y,S,\delta,\lambda,s_0)$, welcher diesen Zähler umsetzt. Als Zustandsmenge soll $S = \{s_0,s_1,...,s_5\}$ verwendet werden. Im jeweiligen Zustand s_n soll n als Binärzahl ausgegeben werden. Zeichnen Sie den Automatengraphen als Repräsentation von $\mathcal{M}$.

   (b) Entwerfen Sie auf Basis von $\mathcal{M}$ ein synchrones Schaltwerk. Verwenden Sie für die Zustandsspeicherung D-Flipflops. Setzen Sie diese Schaltung in QUCS um und überprüfen Sie die Funktionalität. Beachten Sie dabei folgende Punkte:

   - Jeder D-Eingang bekommt eine Digitalquelle, damit die Erzeugung der Wahrheitswertetabelle erleichtert wird.
   - Nutzen Sie die Reset-Eingänge für die Rücksetzlogik.
   - Legen Sie an die Set-Eingänge die konstante 1 (Logic 1) an.

   Hinweis: Jeder Flipflop hat normalerweise auch einen S(et)- und einen R(eset)-Eingang. Diese sind asynchron, also vom Takt unabhängig. Liegt an S eine 1 an, so wird der Ausgang Q unmittelbar auf 1 gesetzt. Liegt an R eine 1 an, so wird der Ausgang Q unmittelbar auf 0 gesetzt. $S = R = 1$ muss vermieden werden, da ansonsten ein undefiniertes Verhalten Auftritt. Die S- und R-Eingänge am D-Flipflop in QUCS werden negiert.

   (c) Entwerfen Sie einen Decoder für eine 7-Segment-Anzeige. In jedem Zustand s_n soll n als Dezimalzahl auf dieser Anzeige ausgegeben werden. Verwenden Sie für die einzelnen LEDs zu Abbildung 3.1 analoge Bezeichnungen. Minimieren Sie die Ausgabefunktion mittels KV-Diagramme und geben Sie für jedes Segment das Schaltnetz an.

   (d) Erweitern Sie in QUCS Ihre Schaltung um diesen Decoder und überprüfen Sie die Funktionalität.
   Hinweis: In QUCS ist keine 7-Segment-Anzeige verfügbar. Es reicht aus, wenn der Ausgabevektor $(ab...g)$ für jeden Zustand richtig ist.