Einführung in die technische Informatik
Skript

Nils Friess

Wintersemester 17/18

Inhaltsverzeichnis

1 Grundlagen

Moorsches Gesetz (1965, 1975)

- Verdopplung der Transistorzahl der ICs alle zwei Jahre
- Verdopplung der Verarbeitungsleistung der Prozessoren alle 1.5 Jahre
- Vervierfachung der Speichergröße alle 3 Jahre
- Verdopplung der Speicherleistung alle 10 Jahre

1.1 Boolsche Algebra (Schaltfunktionen, Normalformen)

Im Folgenden wird die *Konjugation* (Und-Verknüpfung) mit einem $\cdot$ abgekürzt, die *Disjunktion* (Oder-Verknüpfung) mit einem $+$. Negierte Variable werden mit einem Querstrich dargestellt (Bsp. $\overline{a}$).

Produktterme sind Konjugationen einfacher Variablen (können negiert auftreten), also insbesondere auch einzelne Variablen (evtl. negiert).

Summenterme sind Disjunktionen einfacher Variablen bzw. einzelne Variablen (jeweils evtl. negiert).

Minterme sind Produktterme, in denen jede Variable einer Schaltfunktion genau einmal vorkommt (einfach oder negiert). Also ist $\overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot x_4$ ein Minterm für die Funktion $f(x_1, x_2, x_3, x_4)$.

Maxterme sind Summenterme, in denen jede Variable einer Schaltfunktion genau einmal vorkommt (einfach oder negiert). Also ist $\overline{x_1} + x_2 + \overline{x_3} + x_4$ ein Maxterm für die Funktion $f(x_1, x_2, x_3, x_4)$.

Ein Term heißt *disjunktive Normalform* (DNF), falls er nur aus Disjunktionen von Produkttermen besteht (Bsp.: $x + x \cdot \overline{y} + y \cdot \overline{z}$).

Ein Term heißt *konjunktive Normalform* (KNF), falls er nur aus Konjunktionen von Summentermen besteht (Bsp.: $x \cdot (x + \overline{y}) \cdot (y + \overline{z})$).

Ein Term heißt *kanonische disjunktive Normalform* (KDNF), falls er nur aus Disjunktionen einer Menge von Mintermen mit gleichen Variablen besteht. Also wäre eine KDNF zur Funktion $f(x_1, x_2, x_3, x_4)$:
$x_1 \cdot \overline{x_2} \cdot x_3 \cdot x_4 + \overline{x_1} \cdot x_2 \cdot x_3 \cdot \overline{x_4}$.

Ein Term heißt *kanonische konjunktive Normalform* (KKNF), falls er nur aus Konjunktionen einer Menge von Maxtermen mit gleichen Variablen besteht.

1.2 Hauptsatz der Schaltalgebra

Jede Schaltfunktion lässt sich als genau eine KDNF darstellen (bis auf Vertauschungen). Sie wird folgendermaßen gebildet: für jedes $f(\overline{x}) = 1$ aus der Wahrheitstafel bildet man einen Minterm für die KDNF. Dabei wird eine Variable x_i invertiert, wenn die Variable für diesen Eintrag in der Tabelle 0 ist, ansonsten einfach verwendet. Sei bspw. folgender Ausschnitt einer Wahrheitstabelle gegeben:

x_1	x_2	x_3	x_4	Ergebnis
..	..	..	..	..
0	1	0	1	1
..	..	..	..	..

Der Minterm für die KDNF der angegebenen Zeile ist also: $\overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot x_4$.

Analog gilt, dass sich jede Schaltfunktion eindeutig als KKNF darstellen lässt. Hierbei bildet man für jedes $f(\overline{x} = 0$ aus der Wahrheitstafel einen Maxterm. Eine Variable

x_i wird invertiert, wenn die Variable für diesen Eintrag in der Tabelle 1 ist , ansonsten einfach verwendet. Der Maxterm der Zeile aus obiger Tabelle wäre also: $\overline{x_1}+x_2+\overline{x_3}+x_4$.

Es gilt (wegen Dualität):

$$KDNF(f(\overline{x})) = \overline{KKNF(\overline{f(\overline{x})})} \tag{1}$$

$$KKNF(f(\overline{x})) = \overline{KDNF(\overline{f(\overline{x})})} \tag{2}$$

Schaltfunktionen in KDNF (oder KKNF) lassen sich durch die Gesetze der boolschen Algbra *minimieren*. Hierbei ist jedoch ein Größenbegriff notwendig, der das Ziel der Minimierung definiert: Menge der notwendigen Gatter, Anzahl der Variablen, Anzahl der notwendig ICs, Anzahl der notwendigen Kontakte etc. Im Folgenden wird immer so minimiert, dass die Anzahl der verwendeten Symbole minimal wird. Es gibt algorithmische oder graphische Verfahren, die hierzu angewandt werden. Insbesondere die graphischen Verfahren sind jedoch nur für Funktionen weniger Variablen geeignet.

2 Schaltnetze

Ein Schaltnetz ist eine Verknüpfung mehrerer Schaltfunktionen, die jeweils von den gleichen Eingangsvariablen abhängen. Die Ausgänge werden hierbei nicht auf die Eingänge zurückgeführt[1], d. h. bei gleichen Werten an den Eingängen liefert ein Schaltnetz immer die gleichen Werte an den Ausgängen. Typische Schaltnetze sind bspw. *1-aus-k-Multiplexer*, die durch Steuerleitungen viele Eingabeleitungen einem Ausgang zuweisen oder *1-zu-k-Demultiplexer*, die eine Eingabeleitung durch Steuerleitungen vielen Ausgängen zuweist.

Weitere Beispiele sind *k-zu-n-Kodierer*, die einer k-elementigen Eingangskombination, bei der immer nur genau eine Eingangsleitung auf 1 geschaltet ist, eine n-elementige Ausgangskombination zuordnet oder *n-zu-k-Dekodierer*, bei dem n Eingänge genau einen von k Ausgängen selektieren.

3 Schaltwerke (Sequentielle Logik)

Schaltwerke sind Schaltnetze, bei denen Ausgänge auf Eingänge rückgekoppelt werden. Sie können also als gerichtete zyklische Graphen aufgefasst werden. Man unterscheidet zwischen *asynchronen* und *synchronen* Schaltwerken. In ansychronen Schaltwerke sorgen veränderte Eingänge sofort für veränderte Ergebnisse. Damit sind zwar sehr schnelle Schaltungen möglich, zuverlässiges Design gestaltet sich jedoch schwierig. Synchrone Schaltwerke benötigen einen Taktgeber, der für die Weiterleitung stabiler Ergebnisse in die nächste Stufe sorgt.

3.1 Schaltwerkentwurf

Als Systemmodell für Schaltwerke können endliche Automaten herangezogen werden. Diese haben einen endliche Menge von Zuständen. Übergänge zwischen den Zuständen hängen ab von den Eingabenwerten und den vorherigen Zuständen.

3.2 Moore-Automat

TODO

[1] Schaltnetze können auch als gerichtete azyklische Graphen aufgefasst werden, wobei die Kanten jeweils den Verbindungsleitungen entsprechen (gerichtet von Eingang zu Ausgang

3.3 Mealy-Automat

TODO

4 Programmierbare Logikbausteine

4.1 Tri-State-Ausgabelogik

Ausgabeleitungen elektronischer Digitalschaltungen haben entweder den Wert 0 (geringe Spannung bzw. Stromfluss) oder 1 (hohe Spannung bzw. Stromfluss). Gelegentlich sollen sich Ausgänge jedoch elektrisch neutral verhalten (bspw. in einem Bussystem, das mehrere Schaltwerke innerhalb eines Rechners verbindet[2]). Als Lösung dienen Tri-State-Puffer, die zwischen die Schaltwerke und den Bus geschaltet werden.

4.2 Programmierbare Logik

Simple Programmable Logic Devices (SPLD) sind einfache programmierbare Logikbausteine.

(Programmable) Read Only Memory sind Festwertspeicher, deren Werte bei der Herstellung festgelegt wird (ROM) bzw. einmalig durch den Anwender programmiert werden kann (PROM).

(Electrically) Erasable Programmable ROM (EPROM, E^{2}PROM) sind Festwertspeicher, die elektrisch programmierbar sind und durch UV-Bestrahlung bzw. elektrisch löschbar sind.

Flashspeicher sind Festwertspeicher, die in der Regel eine geringere Größe als E^{2}PROM haben und nur blockweise löschbar sind.

Bei *Programmable Logic Arrays* (PLA) handelt es sich um eine frei programmierbare Und-Matrix, welche mit einer festen Oder-Matrix verknüpft ist[3]. Damit kann jede minimierte Schaltfunktion realisiert werden (solange die Anzahl der Produktterme pro Schaltfunktion klein genug ist).

FPGA TODO

5 Zahlendarstellungen und Kodierung

Ein *Zeichen* c ist ein Element einer vereinbarten endlichen, nicht-leeren Menge (s. g. *Zeichenvorrat*) $c \in \{c_1, c_2, \ldots, c_m\}$. Ein *Alphabet* Σ ist ein Zeichenvorrat, auf dem eine lineare Ordnung definiert ist. Dann heißt eine endliche Folge $w = a_1 \ldots a_n$ von Zeichen eines Alphabets Σ Wort über Σ.
$|w| = n$ bezeichnet die Länge der Zeichenkette, wobei für das leere Wort ϵ gilt: $|\epsilon| = 0$.

Für $\Sigma = (0, 1)$ heißen die Elemente der Menge Σ^n Binärwörter der Länge n[4].

[2] Wenn diese Schaltwerke als Eingänge und Ausgänge benutzt werden sollen, gibt es Zustände, die zu Kurzschlüssen bzw. Spannungen im verbotenen Bereichen führen

[3] Es existieren auch PAL mit einer frei programmierbaren Oder-Matrix.

[4] Σ^n bezeichnet die Menge aller Zeichenketten der Länge n über Σ

5.1 Maschinenwörter

Die Hardware eines Rechners verwaltet i. A. nur Binärwörter fester Länge (s. g. Maschinenwörter). Typischerweise wird das 0-te Bit als *Least Significant Bit* (LSB) und das n-1-te Bit als *Most Significant Bit* (MSB) bezeichnet. Technisch existieren zwei unterschiedliche Adressierungsweisen:

- *Big-Endian*: höherwertiges Byte an niedrigerer Adresse
- *Little-Endian*: niederwertiges Byte an niedrigerer Adresse

5.2 Zahlensysteme

Da Menschen gewöhnlich im Dezimalsystem rechnen, Computer aber i. d. R. nur Dualzahlensystem arbeiten können, bedarf es einer Konvertierung. Grundlage hierfür sind Stellenwertsysteme (B-adische Systeme). Wichtige B-adische Zahlensysteme sind:

Basis	Zahlensystem	Ziffern
10	Dezimalsystem	0 ... 9
2	Dual-/Binärsystem	0,1
8	Oktalsystem	0 ... 7
16	Hexadezimalsystem	0 ... 9,A ... F

5.2.1 Horner-Schema

Das Hornerschema kann benutzt werden, um Zahlen von einem in ein anderes Zahlensystem zu konvertieren. Ist b die Ausgangszahl und B die Basis, in die konvertiert werden soll, so konvertiert man diese Zahl folgendermaßen:

$$n = \sum_{i=0}^{k} b_i B^i = ((\dots(b_k \cdot B + b_{k-1}) \cdot B + \cdots + b_2) \cdot B + b_1) \cdot B + b_0 \qquad (3)$$

TODO

5.2.2 Euklidischer Algorithmus

TODO

5.3 Kodierung

Oft ist es nicht praktikabel oder nicht möglich zu übertragende (bzw. zu speichernde) Informationen im Original darzustellen. Deshalb werden diese Informationen kodiert. Man kann hierbei grundsätzlich drei Kodierungsarten unterscheiden:

- Quellkodierung: z. B. ASCII-Code (Text), TIFF (Bilder), MPEG (Videos)
- Kanalkodierung: Darstellung der zu übertragenden Daten in Codewörtern, die den Eigenschaften des Übertragungskanals angepasst sind (u. a. Redundanz)
- Leitungskodierung: Sicherung gegen Übertragungsfehler durch fehlererkennende bzw. -korrigierende Codes

Codes oder Kodierungen sind dabei nichts anderes als Abbildungen $c : A \to B$. Die Ausgangszeichen $a \in A$ heißen hierbei Klarzeichen, die Zielelemente $b \in B$ heißen Codezeichen/Codewörter. Ein Code $c : A \to B^n$, dessen Codewörter alle die gleiche Länge n haben, heißt (n-stelliger) Block-Code. Ein n-stelliger Block-Code heißt *dicht*, wenn alle $b \in B^n$ Codewörter unter c darstellen. Ist $|A| = m$, d. h. A besteht aus m Zeichen, so benötigt man für einen binären Block-Code $c : A \to \{0, 1\}^n$ mindestens $\lceil \log_2 m \rceil$ Stellen.

5.3.1 BCD-Kodierung

BCD steht für **B**inary **C**oded **D**ecimal. Dabei wird jede dezimale Ziffer einer Zahl durch jeweils vier Bit dargestellt (0000 bis 1001). Die Zahl 8127_{10} wird also folgendermaßen dargestellt:

Als BCD-Zahl:	1000	0001	0010	0111_{BCD}
Als Dualzahl:	0001	1111	1011	1111_2

Vorteil der BCD-Kodierung ist, dass die Zahlen leicht zu lesen sind, allerdings ist die Speichernutzung nicht optimal, da die oberen 6 der 16 möglichen Kodierungen keine gültigen Dezimalzahlen darstellen. Diese s. g. Pseudotetraden werden nicht benutzt, BCD-Code ist damit *nicht* dicht.

5.3.2 Gray-Kodierung

Bei der Gray-Kodierung werden aufeinanderfolgende Zahlen so durch Bits kodiert, dass sich stets nur ein Bit ändert (s. g. einschrittige Kodierung). Die einzelnen Stellen besitzen hierbei keine feste Stellenwertigkeit. Untenstehende Tabelle zeigt die Zahlen 0 bis 7 in Gray-Kodierung:

Dezimal	Gray-Kodierung	Dezimal	Gray-Kodierung
0	**0** 0 0 0	4	0 **1** 1 0
1	0 0 0 **1**	5	0 1 1 **1**
2	0 0 **1** 1	6	0 1 **0** 1
3	0 0 1 **0**	7	0 1 0 **0**

5.3.3 Fano-Bedingung

Die Fano-Bedingung besagt, dass kein Codewort als Präfix eines anderen Codes auftreten darf. Man nennt den Code dann präfixfrei. Zeichenfolgen aus Codeworten können so von vorne beginnend eindeutig Wort für Wort dekodiert werden, ohne nachfolgende Zeichen zu beachten.

Der Code $C = \{0, 10, 110, 111\}$ erfüllt also die Fano-Bedingung, da keine der beiden kurzen Sequenzen 0 und 10 als einleitender Bestandteil der längeren Codes auftritt.

5.4 Komprimierende Codes

Komprimierende Codes haben zum Ziel, die Länge kodierter Informationen durch Kompression zu reduzieren um damit bspw. Kosten einzusparen. Im Folgenden wird nur verlustfreie Kodierung behandelt, bei der eine vollständige Dekodierung der Originalinformation möglich ist. Varianten sind Lauflängenkodierung, Wörterbuchkompression, Huffmann-Codierung oder Arithmetische Codierung.

5.4.1 Lauflängenkodierung

Viele Daten enthalten Läufe, d. h. Folgen identischer Zeichen. Die Idee der Lauflängenkodierung ist es, ebendiese Folgen zu kodieren. Die Folge wird bspw. ersetzt durch das Zeichen aus der sie besteht und die Anzahl der Folgenglieder. Zusätzlich wird ein Marker eingefügt, der angibt, das an dieser Stelle eine Kodierung erfolgt ist. Offensichtlich ist diese Ersetzung erst sinnvoll bei mehr als drei Zeichen pro Folge. Im folgenden Beispiel wird das Zeichen # als Marker verwendet[5]:

Daten:	AAABBBBBBBCDDEEEEEEEEEEF#34777777
Kodiert:	AAAB#7CDDE#11F##347#6

5.4.2 **Wörterbuch-Kompression** (Lempel-Ziv)

Hierbei wird schrittweise ein Wörterbuch mit Tupeln (Phrase, Codewort) erzeugt, wobei die Phrase eine Folge von Eingabezeichen ist und das erzeugt Codewort Verweise in das Wörterbuch enthält. Ein solches Wörterbuch ist adaptiv (selbstanpassend) und damit optimal, wenn die Tabelle beliebig groß wird. Anstelle einer Tabelle, kann als Datenstruktur auch ein Baum oder eine Hash-Funktion verwendet werden.

5.4.3 **Informationsgehalt** (Entropie)

Als Entropie bezeichnet man den mittleren Informationsgehalt eines Zeichens. Sie wird definiert durch

$$-\sum_{i=1}^{N} p_i \log_a p_i = \sum_{i=1}^{N} p_i \log_a \frac{1}{p_i} \tag{4}$$

mit N Anzahl der verschiedenen Zeichen
p_i Häufigkeit des Zeichens $i \quad (i = 1, \ldots, N)$
a Basis (für Binärdarstellung $a = 2$)

Ist die Basis $a = 2$, so gibt die Entropie an, wie viele Bits mindestens zur Codierung benötigt werden.

5.4.4 Huffman-Code

Huffman-Code können Daten kodiert werden, bei denen die Häufigkeit aller Zeichen bekannt ist. Der erzeugt Code ist präfixfrei. Durch die Huffman-Codierung werden häufigerauftretende Zeichen kürzer codiert als seltenere. Das Codewort eines Zeichens ergibt sich aus einem Baum, der wiefolgt entsteht[6]:

1. Liste alle Zeichen und deren Häufigkeit auf
2. Wähle die zwei Knoten mit den geringsten Häufigkeiten
3. Mache sie zu Blättern eines binären Teilbaumes, wobei die Häufigkeiten für beide Knoten addiert werden
4. Füge den Teilbaum statt der Knoten wieder in die Liste ein

[5] Der Marker selbst wird durch (Marker, Marker) ersetzt

[6] Für ein vollständiges Beispiel, siehe `http://www.ziegenbalg.ph-karlsruhe.de/materialien-homepage-jzbg/cc-interaktiv/huffman/codierung.htm`

5. Wiederhole Schritte 2 bis 4, bis nur ein Baum vorhanden ist
6. Markiere alle Kanten folgendermaßen
   - Kante führt zu linkem Kind-Element → Kante = 0
   - Kante führt zu rechtem Kind-Element → Kante = 1
7. Das Codewort ergibt sich aus dem Pfad von der Wurzel zum jeweiligen Blatt

5.5 Zeichenkodierung

Die Darstellung von Buchstaben, Sonderzeichen etc. zur Textverarbeitung, -speicherung und -übertragung erfordert i. d. R. eine andere Kodierung als bei Zahlen. Die bekannteste Kodierung hierfür ist der *American Standard Code for Information Interchange* (ASCII). Er verwendet zur Darstellung sieben Bit und kann damit 128 verschiedene Zeichen kodieren (2 mal 26 Buchstaben, 10 Ziffern, 32 Steuer- und Interpunktionszeichen).

Dabei fällt auf, das Umlaute oder andere besondere Zeichen (¿, £ etc.) mit ASCII nicht codiert werden können. Dieses Problem versuchte man durch nationale ASCII-Varianten oder die Erweiterung auf 8-Bit Kodierung zu lösen.

Ender der 80er-Jahre begann ein Konsortium aus Hard- und Softwarefirmen (Apple, IBM, Microsoft ...) mit der Entwicklung einer neuen Kodierung, dem so genannten *Unicode*. Mit Unicode kodierte Zeichen haben eine Länge von 16 Bit.

5.6 Fehlererkennung

5.6.1 Parität

Fehlererkennung durch Parität wird vor allem bei Übertragung und Speicherung von Information eingesetzt. Zum Beispiel kann eine 7-Bit-Kodierung auf eine redundante 8-Bit-Kodierung erweitert werden. Dabei wird das achte Bit durch ein XOR der anderen sieben Bit erzeugt und wird dann Paritätsbit genannt. Dieses Bit ist also genau dann 1, wenn die ersten sieben Bit eine ungerade Zahl von enthalten[7]. Folglich hat die erzeugt Kodierung also immer eine gerade Anzahl an Einsen. Wurde also bspw. bei der Übertragung eines der Bits verfälscht, so kann dieser Fehler erkannt werden. Werden mehrere Bit verfälscht und ist die Anzahl dieser gerade, so ist kein Fehler erkennbar.

5.6.2 Cyclic Redundancy Checksum (CRC)

5.6.3 Hamming-Code

[7] Alternativ kann auch eine Verknüpfung gewählt werden, die dann 1 ist, wenn die Zahl der Einesn gerade ist.