

Boolesche Gleichungen

Aufpassen - wir müssen unterscheiden zwischen:

- a) *deklarativem Gebrauch*. Gleichung im eigentlichen Sinne. Beide Seiten müssen gleich sein, dann ist Gleichung erfüllt. Eine Wertebelegung von a, b, c... erfüllt $f(a, b, c...) = 1$ genau dann, wenn nach dem Einsetzen der konkreten Werte in den Funktionsausdruck sich beim Ausrechnen der Wahrheitswert 1 ergibt. In der richtigen Schaltalgebra handelt es sich wirklich um Gleichungen (mit zwei gleichen Seiten). Solche Gleichungen dürfen nach den üblichen Regeln umgeformt werden - Gleichung bleibt erhalten, wenn auf beide Seite gleiche Umformungen angewendet werden.
- b) *imperativem Gebrauch*. Wertzuweisung Funktionsausdruck => Ergebnisvariable. Ist typisch für das Formulieren von Entwurfsabsichten mit Booleschen Formelausdrücken.

$x = f(a, b, c...)$ meint typischerweise $x = 1$ (wahr, true, asserted), wenn beim Einsetzen der konkreten Werte von a, b, c... in den Funktionsausdruck eine 1 herauskommt.

Negationszeichen auf Ergebnisseite:

- a) nur symbolisch als Teil des Signalbezeichners

$x\# = f(a, b, c...)$ ist nach wie vor dann erfüllt, wenn beim Einsetzen der konkreten Werte von a, b, c... in den Funktionsausdruck eine 1 herauskommt. ($x\#$ wird zu Eins; #-Symbol ist nur schmückendes Beiwerk.)

- b) impliziert Negation (z. B. in ABEL).

$!x = f(a, b, c...)$ heißt: wenn beim Einsetzen der konkreten Werte von a, b, c... in den Funktionsausdruck eine 1 herauskommt, wird $!x$ zu Null (implizite Negation).

Im Fall des Falles auf die jeweiligen Konventionen achten - sonst wird es genau verkehrt herum. Ggf. anhand einfacher Probeentwürfe überprüfen.

Rechentechnische Darstellung Boolescher Ausdrücke

Zwei Einsatzgebiete:

1. Berechnen von Funktionswerten (durch Einsetzen konkreter Variablenbelegungen). Auf vielen Gebieten einsetzbar.
2. Rechnen mit Booleschen Ausdrücken und Gleichungen (Umformen, Optimieren, Erkennen bestimmter Eigenschaften usw.). Typischerweise nur in entwurfsautomatisierungssoftware (EDA).

Berechnen von Funktionswerten

Anwendungen: Nachbilden von kombinatorischen und sequentiellen Schaltungen mittels Software (Simulation, Mikrocontroller als Logikersatz, E-A-Programmierung, Ereignissteuerung, Programmablaufsteuerung (z. B. mittels State Machines).

Nachempfinden der funktionellen Wirkungen durch prozeduralen Ablauf. Die Booleschen Ausdrücke selbst werden rechentechnisch gar nicht ausgewertet; vielmehr wird ein Programm geschrieben, das die äquivalenten Wirkungen erbringt (funktionelle Simulation/Emulation).

Naives Ausprogrammieren mit logischen Verknüpfungsbefehlen (AND, OR, NOT usw.)

Ausprogrammieren durch Ketten von Verzweigungen (= prozedurale Implementierung des binären Entscheidungsdiagramms)

IF-THEN-ELSE-Darstellung (ITE). IF - THEN - ELSE = einfachste Form des binären Entscheidungsdiagramms.

$$\text{ITE}(f, g, h) = f \cdot g \vee \bar{f} \cdot h$$

$$\text{IF } f \text{ THEN } g \text{ ELSE } h$$

Zurückführung der elementaren Aussagefunktionen auf die ITE-Darstellung:

$$\text{AND}(f, g) = \text{ITE}(f, g, 0): \text{IF } f \text{ THEN } g \text{ ELSE } 0$$

$$\text{OR}(f, g) = \text{ITE}(f, 1, g): \text{IF } f \text{ THEN } 1 \text{ ELSE } g$$

$$\text{NOT}(f) = \text{ITE}(f, 0, 1): \text{IF } f \text{ THEN } 0 \text{ ELSE } 1$$

Formel­ausdrücke als Zeichenketten interpretieren (Akzeptorautomat mit Booleschen Operationen). Rechnen mit (Umformen von) Booleschen Ausdrücken durch regelbasierte Symbolverarbeitung (wie Mathcad o. Matlab).

Bitprozessor. Im allgemeinen Sinne typischerweise als Stackmaschine ausgeführt. Beispiele: diverse industrielle Steuerungssysteme. Beispiel einer einfachen Befehlsliste:

```
PUSH Bitadresse
PUSH Festwert (0 oder 1)
POP Bitadresse
AND, OR, NEG, XOR, XNOR, EXIT (= mit dem Berechnen aufhören)
```

Wahrheitstabellen. Bitmuster im Speicher. Gegebene Variablenbelegung dient zum Adressieren des Eintrags.

Binäre Belegungslisten. Mit gegebener Variablenbelegung durchsuchen.

Ternäre Belegungslisten. Listeneintrag ist doppelt so lang. Lohnt sich dann, wenn je Eintrag wenigstens 2 Strichelemente vorkommen. Mit gegebener Variablenbelegung durchsuchen. Dabei Strichlemente ausblenden.

Belegungsliste (binär oder ternär) = Lösungsmenge der Booleschen Gleichung. Rechnen mit Booleschen Funktionen durch Ausführen der entsprechenden isomorphen Operationen über Lösungsmengen:

UND \triangleq Durchschnitt ($\wedge \triangleq \cap$),
ODER \triangleq Vereinigung ($\vee \triangleq \cup$),
Antivalenz \triangleq symmetrische Differenz ($\oplus \triangleq \Delta$),
Negation \triangleq Komplement.

Negation einer gegebenen Funktion kann u. U. auf extrem lange Listen führen. Implementierung ist rechentechnisch an sich einfach (primitive innere Schleifen), aber nicht ohne Spitzfindigkeiten (Mehrfachbelegungen müssen vermieden werden; jedes Listenelement muß eine disjunkte Teilmenge repräsentieren (Orthogonalisierung)). Nützliche Befehle: MMX, SSE usw.

Binäre Entscheidungsdiagramme. Werden rechnerintern als Zeigerlisten dargestellt.

Jeder Knoten als:

- Variable,
- Zeiger auf 0-Nachfolger,
- Zeiger auf 1-Nachfolger.

Diagramm wird durch Negation nicht verlängert. Beim Negieren werden nur die Endwerte ausgewechselt

Verknüpfen binärer Entscheidungsdiagramme gemäß ITE-Schema.

Der Rechenzeitbedarf hängt maßgeblich von der gewählten Variablenreihenfolge ab. (Bei gewissen Schaltfunktionen führt eine ungünstige Festlegung der Variablenreihenfolge unweigerlich zu exponentiellem Rechenzeitbedarf. Das betrifft u. a. auch Schaltfunktionen, die zur Beschreibung von Additionsnetzwerken verwendet werden.)