Inhaltsverzeichnis

1 Grundlagen

1.1 Boolesche Axiome

Kommutativität:
$x + y = y + x \quad \forall x, y \in \{0, 1\}$
$x * y = y * x \quad \forall x, y \in \{0, 1\}$
Assoziativität:
$x + (y + z) = (x + y) + z \quad \forall x, y, z \in \{0, 1\}$
$x * (y * z) = (x * y) * z \quad \forall x, y, z \in \{0, 1\}$
Apsorption:
$x + (x * y) = x \quad \forall x, y \in \{0, 1\}$
$x * (x + y) = x \quad \forall x, y \in \{0, 1\}$
Distributivität:
$x + (y * z) = (x + y) * (x + z) \quad \forall x, y, z \in \{0, 1\}$
$x * (y + z) = (x * y) + (x * z) \quad \forall x, y, z \in \{0, 1\}$
Komplement:
$x + (y * \neg y) = x \quad \forall x, y \in \{0, 1\}$
$x * (y + \neg y) = x \quad \forall x, y \in \{0, 1\}$

1.2 Boolesche Regeln

Doppeltes Komplement:
$\neg(\neg x) = x \quad \forall x \in \{0, 1\}$
Idempotenz:
$x + x = x * x = x \quad \forall x \in \{0, 1\}$
De-Morgan-Regel:
$\neg(x + y) = \neg x * \neg y$
$\neg(x * y) = \neg x + \neg y$
Consensus-Regel:
$(x * y) + ((\neg x) * z) = (x * y) + ((\neg x) * z) + (y * z)$
$(x + y) * ((\neg x) + z) = (x + y) + ((\neq x) + z) * (y + z)$

1.3 Graphen

1.3.1 Allgemeines

$G = (V, E)$
$V \ldots$ endliche Menge an Knoten
$E \ldots$ endliche Menge an Kanten
$\mathrm{Q}(e) \ldots$ Quelle der Kante e $\mathrm{Q} : E \to V$
$\mathrm{Z}(e) \ldots$ Ziel der Kante e $\mathrm{Z} : E \to V$
$\mathrm{indeg}(v)$ $\mathrm{indeg} : V \Rightarrow \mathbb{N}$, Eingangsgrad $\mathrm{indeg}(v) = \{v | \mathrm{Z}(e) = v\}$
$\mathrm{outdeg}(v)$ $\mathrm{outdeg} : V \Rightarrow \mathbb{N}$, Ausgangsgrad $\mathrm{outdeg}(v) = \{v | \mathrm{Q}(e) = v\}$

1.3.2 Bestimmung der Knoten

Ein Knoten heißt:

- Wurzel $\Leftrightarrow \text{indeg}(v) = 0$
- Blatt $\Leftrightarrow \text{outdeg}(v) = 0$
- innerer Knoten $\Leftrightarrow \text{outdeg}(v) > 0$

Die Graphtiefe ist der längste Pfad in einem Graphen, ferner ist ein Pfad eine Folge von k Kanten. ($k \in \mathbb{N}$)

1.3.3 Bäume

Als Baum wird ein gerichteter azyklischer Graph mit einer Wurzel bezeichnet.
Binärer Baum: $\text{outdeg}(v) \leq 2$

1.4 Landausche $\mathcal{O}$-Notation

Die landausche $\mathcal{O}$-Notation ist eine asymptotische Abschätzung für die Größe parametrisierbarer Objekte, Laufzeit von Algorithmen, et cetera.

$$\exists c, x_0 \in \mathbb{R}_0^+ : \text{f}(x) \leq c * \text{g}(x) \Leftrightarrow f(x) \in \mathcal{O}(\text{g}(x))$$

Bsp.:

$5 \cdot x + 2 \in \mathcal{O}(x^2)$

$c = 6,\ x_0 = 2 \quad \forall x > x_0$

$5 \cdot x + 2 <_{x>2} 5 \cdot x + x = 6 \cdot x \leq_{x \geq 1} 6 \cdot x^2 \quad \forall x > 2$

2 Kodierung

2.1 Kodierung von Zeichen

2.1.1 Alphabet, Wörter und Zeichen

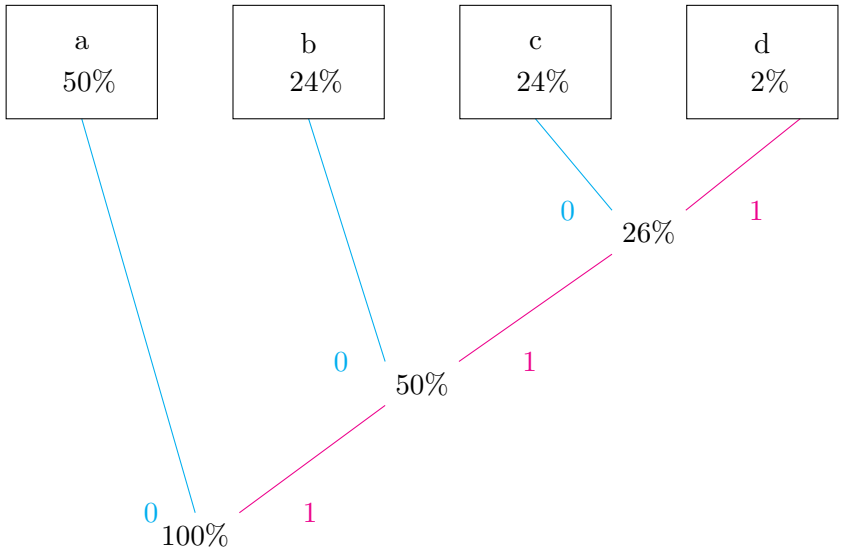
A= $\{a_1, \ldots, a_n\}$ A$\neq \emptyset$
A ist endliches Alphabet der Größe m.
$a_1, \ldots, a_n$ sind Zeichen des Alphabets.
A*= $\{w | w = b_1, \ldots, b_n \quad n \in \mathbb{N} \quad \forall i \quad 1 \leq i \leq n : b_i \in \text{A}\}$
A* ist die Menge aller Wörter über dem Alphabet A.
$|b_1 \ldots b_n| := n$ heißt Länge des Wortes $b_1 \ldots b_n$
$\mathcal{E}$ ist das leere Wort ($n = 0$)

2.1.2 Huffmann-Code

Vorgehensweise:

1. Relative Häufigkeitsverteilung der Zeichen ermitteln.
2. Nun werden solange die beiden geringsten Häufigkeitsverteilungen baumartig miteinander verbunden und addiert, bis keine Häufigkeiten mehr übrig sind.
3. Die entstandene Baumstruktur dient zur Kodierung. Linke Pfade werden mit der Ziffer 0 oder 1 belegt und rechte Pfade mit der entsprechend anderen Ziffer.

Bsp.:



Zeichen:	a	b	c	d
Code:	0	10	110	111

Mittlere Codelänge des Huffmann-Code:
$C : A \to \{0,1\}* : \sum_{i=1}^{m} \mathrm{p}(a_i) \cdot |\mathrm{c}(a_i)|$

2.2 Kodierung von Zahlen

Festkommazahlen: n+1 Vorkommastellen und k ≥ 0 Nachkommastellen b ... Basis i ... Stelle $\delta(d_i)$... Zuordnung der Ziffer zur natürlichen Zahl
$< d >= \sum_{i=-k}^{n} b^i \cdot \delta(d_i)$

2.2.1 Betrag und Vorzeichen

$$(-1)^{d_n} \cdot \sum_{i=-k}^{n-1} d_i \cdot 2^i \quad \text{symmetrischer Zahlenbereich: } R_n = [-2^n + 2^{-k}, 2^n - 2^{-k}]$$

2.2.2 Einerkomplement

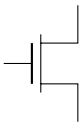
$$\sum_{i=-k}^{n-1} d_i \cdot 2^i - d_n \cdot (2^n - 2^{-k}) \quad \text{symmetrischer Zahlenbereich: } R_n = [-2^n + 2^{-k}, 2^n - 2^{-k}]$$

2.2.3 Zweierkomplement

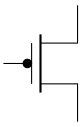
$$\sum_{i=-k}^{n-1} d_i \cdot 2^i - d_n \cdot 2^n \quad \text{asymmetrischer Zahlenbereich: } R_n = [-2^n, 2^n - 2^{-k}]$$

3 Kombinatorische Logik

3.1 Kombinatorische Schaltkreise



n-Kanal Transistor
leitet bei logisch 1
sperrt bei logisch 0



p-Kanal Transistor
leitet bei logisch 0
sperrt bei logisch 1

3.2 Formale Beschreibung von Schaltkreisen

S_k ist ein 5-Tupel bestehend aus:

$$S_k = \{\overrightarrow{X_n}, G, \text{typ}, \text{IN}, \overrightarrow{Y_m}\}$$

$\overrightarrow{X_n}$ ist eine endliche Folge von Eingängen.

G: Gerichteter Graph mit Menge der Knoten, V, und Menge der Kanten, E. Hierbei sind Knoten des Graphen Gatter, Konstanten und Eingangsvariablen. Kanten des Graphen sind die Verbindungen.

$I = V \backslash (\{0, 1\} \cup \{x_1, \ldots, x_n\})$: Die Menge der Gatter in G.

typ : $I \rightarrow BIB$ (Gatterbibliothek), typ ordnet also jedem Gatter des Schaltkreises seinen typ zu. (OR, EXOR, AND, NAND, ...)

IN: $I \rightarrow E$ ordnet jedem Gatter die eingehenden Kanten zu.

$\overrightarrow{Y_m} = (y_1, \ldots, y_m)$ zeichnet die Knoten $y_1, \ldots, y_m$ als Ausgänge aus.

3.3 Belegung α, Simulation und Interpretationsfunktion Ψ

$$\alpha = (\alpha_1, \ldots, \alpha_n)$$

$$\Phi_{\mathrm{SK},\alpha}(X_i) = \alpha_i \quad \forall 1 \leq i \leq n$$

Berechnung von $\Phi_{\mathrm{SK},\alpha}$ bei Eingangsbelegung α heißt Simulation. Die boolesche Funktion an einem Knoten heißt Interpretationsfunktion Ψ:

$$\Psi(v) : \mathcal{B}^n \to \mathcal{B}$$

$$\Psi(v)(\alpha) := \Phi_{\mathrm{SK},\alpha}$$

$$\Psi(0) = 0 \qquad \Psi(1) = 1$$

$$\Psi(X_i)(\alpha_1, \ldots, \alpha_n) = a_i \quad \forall \alpha \in \mathcal{B}^n$$

$$\Psi((g + h)) = \Psi(g) + \Psi(h)$$

$$\Psi(g \cdot h) = \Psi(g) \cdot \Psi(h)$$

$$\Psi(\neg g) = \neg\Psi(g)$$

3.4 Literale, Monome, Minterme

$x_i, x_i' \in BE(x_n)$
x_i heißt positives Literal (auch x_i^1)
x_i' heißt negatives Literal (auch x_i^0 oder $\bar{x_i}$)

$\bigwedge_1^n x_i^{\alpha_i}$ $x_i = \{x_1, \ldots, x_n\}$ heißt Monom, wenn jedes x_i maximal einmal vorkommt. Zu keinem x_i, x_i' darf der Gegensatz auftreten.

Bsp.: $n = 4 \alpha = (0, 1, 1, 0) \in \mathcal{B}^4$
$m(\alpha) = x_1^0 \cdot x_2^1 \cdot x_3^1 \cdot x_4^0 = \bar{x_1} \cdot x_2 \cdot x_3 \cdot \bar{x_4}$
$\Psi(m(\alpha))(\alpha) = \bar{0} \cdot 1 \cdot 1 \cdot \bar{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$
$\Psi(m(\alpha))(1, 1, 1, 0) = \bar{1} \cdot 1 \cdot 1 \cdot \bar{0} = 0 \cdot 1 \cdot 1 \cdot 1 = 0$

3.5 Polynome, KDNF

$\bigvee_{i=1}^m \bigwedge_{i=0}^{n_i} x_i^{\alpha_i}$ heißt Polynom.

Polynome sind also eine Disjunktion von Konjunktionen paarweise unterschiedlicher Literale (Monome). Die Monome sind hierbei ebenfalls paarweise verschieden.
Ein Polynom von f heißt auch disjunktive Normalform von f. Ein vollständiges Polynom von f heißt auch kanonische Disjunktive Normalform (KDNF) von f.
Vollständig ist ein Polynom dann, wenn alle Monome des Polynoms Minterme sind.

Als ON-Menge oder Erfüllbarkeitsmenge von f werden alle Belegungen $\alpha \in \mathcal{B}^n$ bezeichnet, die unter Anwendung der booleschen Funktion f 1 ergeben.
$ON(f) : \mathcal{B}^n \to \mathcal{B}$

3.6 Programmable logical Array (PLA)

$M(p_1, \ldots, p_m) \ldots$ Menge der in diesen Polynomen verwendeten Monome
$q \ldots$ Monome in den Polynomen
$\text{cost}_1(p1, \ldots, p_m)$: Anzahl der benötigten Zeilen im PLA.
$\text{cost}_2(p1, \ldots, p_m)$: Anzahl der benötigten Transistoren.
$\text{cost}_1(p1, \ldots, p_m) = |M(p1, \ldots, p_m)|$
$\text{cost}_2(p1, \ldots, p_m) = \sum_{q \in M(p_1, \ldots, p_m)} |q| + \sum_{i=1}^{m} |M(p_i)|$

3.7 Implikanten, Primimplikanten

$f, g \in \mathcal{B}_n$, boolesche Funktionen.
Es gilt: $f \leq g$, wenn:
$\forall \alpha \in \mathcal{B}_n$:

$$\begin{aligned} & f(\alpha) \leq g(\alpha) \\ \Rightarrow & |ON(f)| \leq |ON(g)| \\ \Rightarrow & ON(f) \subset ON(g) \end{aligned}$$

Ein Implikant von f ist ein Monom q mit $q \leq f$. Ein Primimplikant von f ist ein Monom q von f, bei dem es keinen Implikanten s von f gibt, sodass: $q \leq s$.

3.8 Minimalpolynom

Ein Minimalpolynom p einer booleschen Funktion f ist mit der Eigenschaft
$\text{cost}(p) \leq \text{cost}(p') \ \forall p'$ gegeben

3.9 Quine-McCluskey

Verfahren: Vergleiche alle Elemente der ON-Menge (L_0), die sich in einer Stelle unterscheiden ($m \cdot x, m \cdot x'$). Schreibe die nicht verwendeten Elemente in Prim(f) (Menge) und schreibe die verglichenen Elemente in der Form m - , wobei - die beliebige Variable ist, in die neue Menge L_{n+1}^m. Verfahre weiter, bis keine Elemente mehr verglichen werden können.

3.10 Primimplikantentafel

3.10.1 Motivation

Beim Verfahren von Quine und McCluskey könnenals Ergebnis sich überdeckende Primimplikanten herauskommen.Zur Überprüfung wird dann eine Primimplikantentafel gebidet:

	$\min(\alpha_1)$	...	$\min(\alpha_n)$
m_1			
...			
m_n			

$$\alpha_1, \ldots, \alpha_n \in \mathrm{ON}(f), \quad m_1, \ldots, m_n \in \mathrm{Prim}(f)$$

3.10.2 Reduktionsregeln für die Primimplikantentafel

1. Entferne aus der PIT alle wesentlichen Primimplikanten und alle MInterrme, die von diesen überdeckt werden.
   Wesentlich ist ein Primimplikant, wenn er als einziger einen Minterm (Element aus der ON-Menge) überdeckt.

2. Entferne aus der Primimplikantentafel alle Minterme, die einen anderen Minterm dominieren, also an allen Stellen des dominierten Minterms, bei dem dieser den Wert 1 hat ebenfalls den Wert 1 haben oder mehr vorkommen hiervon haben.

3. Entferne aus der PIT(f) alle Primimplikanten, die durch einen anderen, nicht teureren Primimplikanten dominiert werden.
   Dominiert wird ein Primimplikant, wenn er die gleichen Minterme überdeckt und der dominierende Minterm zusätzlich an anderen Mintermen eine 1 hat. Letzteres gilt nicht zwangsläufig.

Ein Primimplikant heißt wesentlich, wenn es einen Minterm $\min(\alpha)$von f gibt, der nur von diesem Primimplikanten überdeckt wird.

$$\mathrm{PIT}(f)[m, min(\alpha)] = 1$$
$$\mathrm{PIT}(f)[m', min(\alpha)] = 0 \quad \forall m' \in Prim(f)$$

Eine PIT heißreduziert, wenn keine der Reduktionsregeln mehr anwendbar sind.
Ist eine reduzierte PIT nicht leer, so spricht man vom zyklischen Überdeckungsproblem. Dieses ist nur heuristisch lösbar. (z.B. Petrick's Methode)

3.10.3 Petrick's Methode

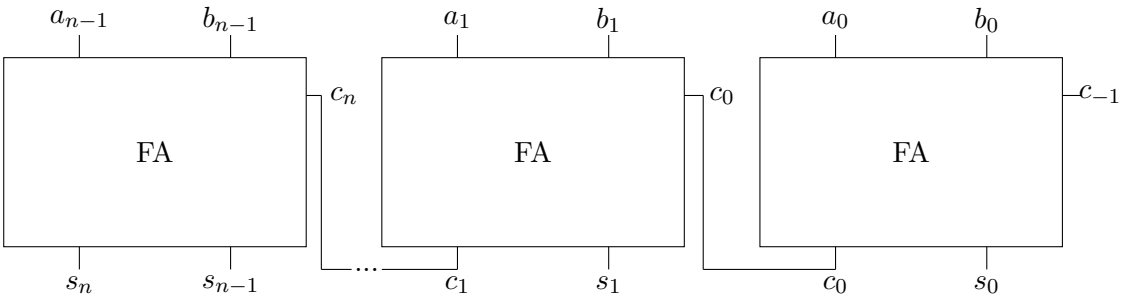
Überdeckte Minterme werden disjunktiv verknüpft und alle überdeckenden Minterme werden konjunktiv verknüpft. Es wird jeder Ausdruck miteinander ausmultipliziert. Die Terme mit der geringsten Anzahl an Literalen ergeben das Minimalpolynom.

3.11 Arithmetische Schaltungen

3.11.1 Kostenmaß

Die Kosten eines Schaltkreises sind durch die Anzahl seiner Gatter gegeben (große Kosten bedeuten hohes Energiepensum und eine größere Fläche)
Die Tiefe eines Schaltkreises ist definiert durch die maximale Anzahl an Gattern auf einem Pfad eines beliebigen Eingangs zum Ausgang. (Signallaufzeit)

3.11.2 Carry-Ripple-Addierer (Schulmethode)



3.12 Full Adder (FA)