Overhead
Sicherungsbit
Nutzdatenbits

Effizienz
Nutzdaten
alle Daten

Zeit: x-Achse
Wert: direkt nach Änderung

Diskriminierung
hart: mit ohne Grenze
weich: mit undef. Bereich

Addition STIBITZ
→ addieren der STIBITZ-Zahlen
→ bei Übertrag 0011 in übertragende Tetrade addieren
→ sonst 1101 addieren
WICHTIG: 0011 & 1101 - Überträge werden weggelassen

Absolut: # unbenutzte Worte = R

relativ: $r = \frac{R}{2^n}$

(in Bit)

Informationsgehalt
$H_e = ld(1/p)$

Entropie ($\emptyset$ Informationsgehalt)
$H = \sum_{i=1}^{N} p(i) \cdot ld(\frac{1}{p_i})$

Addition BCD
→ Bei Pseudotetrade Addition 0110
→ Bei Übertrag Addition von 0110 zu übertragenden Tetrade
→ wenn Addition durch Pseudotetrade → Übertrag, keine neue Addition!

mittlere CW-Länge
$\sum p(x_i) \cdot l(x_i) = \bar{m}$

$\binom{m}{k} = \frac{m!}{k!(m-k)!}$
m → Bits
k → Einsen
→ $HD_{min} = 2$

ASCII
MSB → 1. drei
LSB → letzten 4

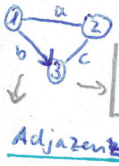
Hexadezimal A=10, B=11, C=12, D=13, E=14, F=15

Fehler
erkennbare Fehler = $HD_{min} - 1$
Korrigierbare Fehler ≤ $HD_{min} - 1$

Quotient, Basis ⇓ Quotient, Rest 1
N_D / Basis ⇓ Quotient, Rest 0
N_B = Rest, Rest 0
Zahlensystem

Graphen
V = Menge Knoten, E = Menge Kanten
$\phi(e)$ = Abbildung zweier Knoten auf eine Kante e
Bsp.: ① —a— ② $\phi(a) = \{1,2\}$

→ $G(V, E, \phi)$ = abstrakter Graph



Inzidenz
→ Kante a ist inzident zu Knoten 1 und 2

Inzidenzmatrix:

Kanten \ Knoten	a	b	c
1	1	1	0
2	1	0	1
3	0	1	1

Adjazenz
→ Knoten 1 und 2 sind adjazent zu Kante a

Adjazenzmatrix:
- Richtung beachten!
- symmetrisch falls ungerichtet

Knoten \ Knoten	1	2	3
1	0	1	1
2	1	0	1
3	1	1	0

Isomorphie
Beibehalten der Inzidenzbeziehung (Umordnung durch Umbenennung)

Grad einer Knoten
$d(g)$ = # Kanten am Knoten
$d^+(g)$ = # Abgehende Kanten vom Knoten
$d^-(y)$ = # Ankommende Kanten am Knoten

Arten:
① zusammenhängend
→ von jedem Knoten kann man zu jedem anderen gelangen
② Endlicher Graph:
V und E endlich
③ Entarteter Graph
→ keine Kanten
④ Einfacher Graph
→ zwei Knoten höchstens eine Kante
⑤ Planarer Graph
→ Graph zudem es einen Kreuzungsfreien, isomorphen Graphen gibt
⑥ zyklischer Graph
→ enthält Zyklen, z.B. Schleife
⑦ Baum
- zusammenhängender, zyklenfreier Graph
⑧ streng zusammenhängend
- unter Beachtung der Kantenrichtung zusammenhängend

Packed (Unpacked-Code (nur im BCD)
- Unpackt-Code hat gleich viele Nullen wie Buchstaben vorn
- Packed ist der normale Code

Bsp.: 1001 → Packed Code
0000 1001 → unpacked Code

Relationen
① Reflexivität $x \alpha x$ (Schleife) $\alpha \Leftrightarrow =$
② Symmetrie: aus $x \alpha y$ folgt $y \alpha x$
0 ⟷ 0 (Spiegelung an Matrixdiagonalen) $[\alpha \Leftrightarrow =]$
③ Antisymmetrie aus $x \alpha y$ und $y \alpha x$ folgt $x = y$
$\Leftrightarrow \alpha \triangleq \leq / \geq \Rightarrow 0 \rightarrow 0$
Jedes Matrixelement nicht spiegelsymmetrisch an Diagonale
④ Transitivität $x \alpha y$, $y \alpha z \Rightarrow x \alpha z \rightarrow \alpha \Leftrightarrow =$
0 → 0 → 0

Spezielle Relationen
① Ordnungsrelation: Reflexiv, transitiv, antisymmetrisch $=, \leq$
② strenge Ordnungsrelation: Antisymmetrisch, transitiv, Antireflexiv $<$
③ Äquivalenzrelation: Reflexiv, transitiv, symmetrisch $=$
④ Verträglichkeitsrelation: Reflexiv, symmetrisch, nicht transitiv $\sim$

Null-/Einsblocküberdeckung
- Nullblocküberdeckung aufstellen
→ beschreibt alle Nullblöcke ohne Freistellen (nicht negieren!)
- Einsblocküberdeckung aufstellen
→ beschreiben aller Einsblöcke ohne Freistellen.
$T_1 = \{B_{11}, B_{12}, \ldots, B_{1r}\}$

Beispiel:

x1 \ x2 x3	00	01	11	10
0	1	-	1	
1	0	0	1	

$T_0 = ((0,0,0), (-,1,1))$
$T_1 = ((1,-,0), (-,1,0), (0,0,1))$

Nelson

Null-/Einsvervollständigung
Nullblocküberdeckung (NBÜ) → Einsvervollständigung f^E
→ einzelne Literale der NBÜ negieren und verodern
→ dann verunden der Blöcke (nun hat man f^E) => Kurzform der KNF
→ ausdistribuieren von f^E und man hat alle Primimplikanten [KMF]
(hier kann man noch Freistellenblöcke streichen)

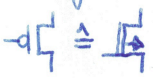
Einsblocküberdeckung (EBÜ) → Nullvervollständigung f^N
→ einzelne Literale der EBÜ verunden
→ verodern der Blöcke (nun hat man f^N) => Kurzform der DNF
→ ausdistribuieren und man hat Primimplikanten [DMF]
(hier kann man noch Freistellenblöcke streichen)

CMOS
① oben bei VDD: Pull-Up-Netz (oft P)
→ in PMOS ≙ negative Logik. Wird mit negierung gezeichnet
deshalb werden alle Literale aus der Funktion beim zeichnen negiert.

Beispiel: b, $\bar{a}$ ≙ $(\bar{b} \vee a)$ in PUN

VDD, y

Wichtig:

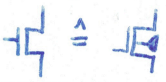


② unten bei GROUND: Pull-Down-Netz (oft G)
→ in NMOS ≙ positive Logik, Wird eingezeichnet wie in der Funktion, deshalb auch ohne negierungskreis:

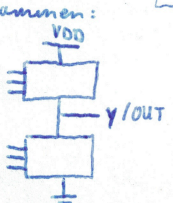
Beispiel: b, $\bar{a}$, y, GND

≙ $b \vee \bar{a}$ in PDN

Wichtig:

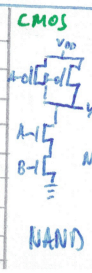


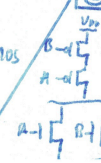
Zusammen:

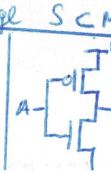


Bit	x	2^x	x	2^x
0	0	1	9	512
1	1	2	10	1024
2	2	4	11	2048
3	3	8	12	4096
4	4	16	13	8192
5	5	32	14	16384
6	6	64	15	32768
7	7	128	16	65536
8	8	256		

CMOS

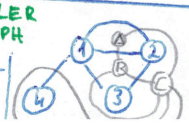






Mengen
① Menge: $M = \{a, b, \ldots\}$ (ungeordnet)
② Tupel: $T = (a, b)$ (geordnet/Reihenfolge wichtig!)
③ Kardinalität: $|M|$ Anzahl aller Elemente einer Menge ($M = \{1,2,3\}, |M| = 3$)
④ Potenzmenge P(M) Menge aller Untermengen
$M_2 = \{0,1\} \rightarrow P(M_2) = \{\{\}, \{0\}, \{1\}, \{0,1\}\}$
⑤ Vereinigung $S \cup T$
⑥ Schnitt $S \cap T$
⑦ Komplement $C_M(S)$: Alle Elemente die zu M, aber nicht zu S gehören.
⑧ Kartesisches Produkt: $S \times T$ = Menge aller Tupel von S, T:
$S = \{0,1\}$, $T = \{a,b\} \rightarrow S \times T = \{(0,a), (0,b), (1,a), (1,b)\}$ | $S \times T \neq T \times S$
$|S \times T| = |S| \cdot |T|$
⑨ Untermenge $S \subset M$: jedes Element von S gehört zu M

DUALER GRAPH
Vorschrift



Petrick

a) Graphisch

1. Erstellen einer Überdeckungstabelle
2. Spalten mit Kosten versehen zur Bewertung
3. Überdeckungstabelle in folgenden Zyklus bearbeiten:

Kerne → Spaltendominanz → Zeilendominanz (Kosten beachten) ↑

4. Bei gleichwertigen Lösungen: Petrickausdruck
5. sonst: disjunktion der Ermittelten Kerne (DMF)

b) Algebraisch

1. Überdeckungstabelle erstellen
2. Jede Zeile Präsenzvariabel geben
3. Petrickausdruck erstellen
   → in Spalten mit x verschenen Präsenzvariablen verodern
   → und „Spaltenblöcke" verunden
4. Ausschistribuieren ergibt verODERte Lösung
5. Kostenminimalste Lösung suchen (durch Kosten-Funktion)

Nelson - Petrick

- Nelson: Bestimmung aller Primimplikanten / Primimplikate
- Petrick: Bestimmung der kostenminimalsten Auswahl an Primimplikanten / Primimplikaten

⇒ Nelson-Petrick: liefert insgesamt einen kostenminimalen algebraischen Ausdruck der Schaltfunktion

Primimplikanten / Primimplikate

- Blöcke die Eins- und Freistellen überdecken (für DMF) heißen Primimplikanten
- Blöcke die Null- und Freistellen überdecken (für KMF) heißen Primimplikate

Kostenfunktion L(y)

→ Bewertung der Minimierung, entweder durch Anzahl der Literale, oder durch Anzahl der Literale und der Terme

Bei Petrick

→ Bei Zeilen werden die kleineren von den größeren gestrichen
→ Bei Spalten werden die größeren von der kleineren gestrichen
Bei beiden gilt aber: viele Kreuze = stark | wenige Kreuze = schwach

- Spaltendominanz: Streichen der „stärkeren", dominierenden Zeile
- Zeilendominanz: Streichen der „schwächeren", dominierten Zeile

Minimale Anzahl von Transistoren in CMOS

→ PDN / PUN ausdistribuieren, so weit wie möglich
→ Nun Anzahl der Transistoren ≙ Anzahl der Literale

Minterm / Maxterm

Minterm: beschreibt die Belegung der Funktionswerte auf 1 | Maxterm: beschreibt Belegung der Funktionswerte auf 0 (da Maximale Anzahl 1en) → Literale verodert

DNF / KNF

→ Jede beliebige Funktion mit darstellbar

→ DNF: „Veroderung der Minterme" $y = \bigvee_{j=0}^{2^n-1} (f_j \& m_j)$

→ KNF: „Verundung der Maxterme" $y = \&_{j=0}^{2^n-1} (f_j \vee M_j)$

Belegungsmenge

≙ Einsstellenmenge / Nullstellenmenge / Freistellenmenge.

⇒ E = {1,2} / N = {0} / F = {3} [meist Oktal]

DMF / KMF

2 Verfahren

1. geometrisch: → Blockbildung im Symmetriediagramm
   DMF: Einsblöcke verodern (Literale verunden)
   KMF: Nullblöcke verunden (Literale verodern)
2. algebraisch → ausdistribuieren von KNF/DMF

KNF → DMF: (A∨B)&(A∨C) → A&A ∨ A&C ∨ A&B ∨ B&C → A∨BC

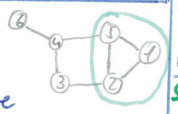
DNF → KMF: (AB) ∨ (AC) → (A∨A)(A∨C)(B∨A)(B∨C) → A ∆ (B∨C)

Primblöcke

alle nehmen die man findet!

Clique

Teilmenge von Knoten in einem ungerichteten Graphen, bei der jeden Knotenpaar durch eine Kante verbunden ist



Vorteil Parität zu zyklischer Redundanzprüfung

- einfach zu implementieren

Nachteil: bei Mehrbitfehlern

Shannon-Fano code

sortierung bleibt beibehalten

Subtraktion Stibitz

1. BCD Zahlen in STIBITZ bringen
2. Subtraktion mit 2-er Komplement
3. Entsteht bei der ersten Tetrade ein Übertrag dann ist Ergebnispos. sonst nicht.
4. Korrektur wie beim STIBITZ (vorderster übertrag Nicht!)
5. Subtraktion zu Ende führen

Kantenprogression

ungerichteter Graph		gerichteter Graph	
$g^1 \neq g^{u+1}$	$g^1 = g^{u+1}$	$g^1 \neq g^{u+1}$	$g^1 = g^{u+1}$
offene KPG	geschlossene KPG	offene	geschlossene KPG

alle Kanten einer PG verschieden

KettenPg	geschlossene Kantenzug PG	Weg PG	Zyklus PG

Kanten einer PG ohne Ordnung

Kette	geschlossener Kantenzug	Weg	Zyklus

KPG → Kantenprogression | KPG Kantenprogression

Wert-Zeit-Digitalisierung

wertdiskret-zeitkontinuierlich → wert
→ zeitdiskret → wertkontinuierlich → zeit

Mächtigkeit der Potenzmenge

$|P_i| = 1 + \sum_{i=1}^{n} \binom{n}{i} / 2^n = |P|$

Gray code

kann nicht zu zweideutigen digitalsignalen kommen, da nur 1 Bit sich ändert → Vorteil

Relais Gatter

Relais:

Gatter:

Artikulation

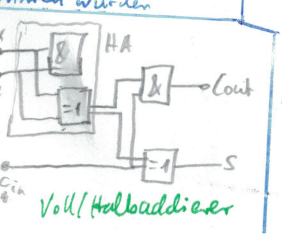
Knoten nach dessen entnahme das Graph in isolierte Teilgraphen zerfällt

Prüfstrich

Um verwechselbare CW → kein CW ist teil eines anderen

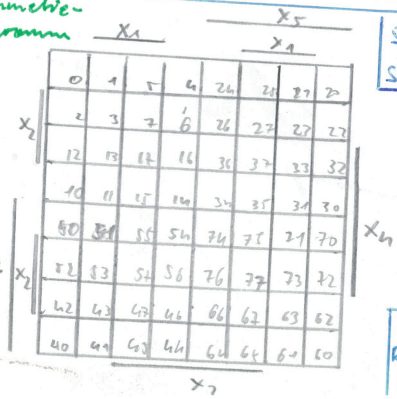
Digitalisierung

Hiermit können Signalschwankungen abgefangen werden, die sonst zu ungewollten Wertänderungen führen würden



Voll / Halbaddierer

Symmetriediagramm



SNT

SUT

Benötigte Flip-Flops

#Flip-Flop = ld(Anzahl der Zustände)

isomorpher Graph

Man kann beweisen, dass zu jedem Graphen ein isomorpher, dreidimensionaler Graph existiert

Zyklische Restklasse

→ nicht vereinfachbare Überdeckungstabelle →

Entropie

theoretisches Maximum einer Komprimierung

Bipartiter Graph

Knoten teilen sich in 2 disjunkte Teilmengen A und B auf, sodass zwischen den Knoten beiden Teilmengen Keine Kanten verlaufen

Burstfehler

→ Blocksicherung
→ Hamming Codes

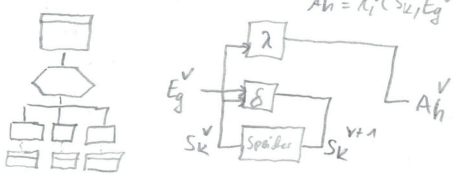
Grey Code	Dez	Binär	Okt	Hex
0000	0	0000	00	0
0001	1	0001	01	1
0011	2	0010	02	2
0010	3	0011	03	3
0110	4	0100	04	4
0111	5	0101	05	5
0101	6	0110	06	6
0100	7	0111	07	7
1100	8	1000	10	8
1101	9	1001	11	9
1111	10	1010	12	A
1110	11	1011	13	B
1010	12	1100	14	C
1011	13	1101	15	D
1001	14	1110	16	E
1000	15	1111	17	F
	16	10000	20	10

Automatentypen

- Mealy: - Ausgabe von Zustand und Eingabe abhängig

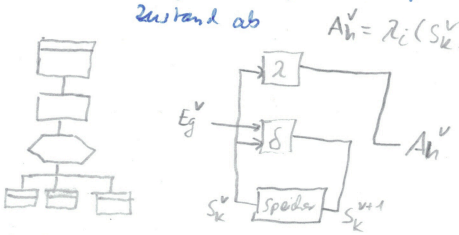
„In jedem Zustand hängt die Ausgabe von der aktuellen Eingabe ab."

$A_h^v = \lambda_i(S_k^v, E_g^v)$



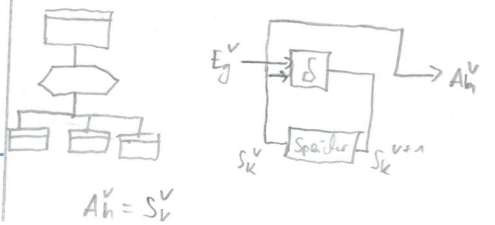
- Moore: - Ausgabe hängt ausschließlich von Zustand ab

$A_h^v = \lambda_i(S_k^v)$



- Medvedew „Zustandsautomat"

→ Es gibt keine Ausgabe, da die Ausgabe direkt der Zustand ist.



$A_h^v = S_k^v$

K	Übergang in Zustand K
V	Ausgabe des Vektors y
XR	Abfrage der relevanten Eingangsvariablen XR_1^v XR_2...
A_h^v	→ Ausgangsvektor
S_k^v	→ Zustandsvektor
E_g^v	→ Eingangsvektor