Zapisy funkcji boolowskiej przydatne przy projektowaniu

- tabela prawdy → pomocna przy przechodzeniu od zapisu słownego do formalnego
- postać dziesiętna → łatwy zapis
- Mapa Karnaugha → pomocna przy minimalizacji
- postać algebraiczna → zapis boolowski
- schemat logiczny na bramkach

Zestaw funkcjonalnie pełny

Każdą funkcję logiczną można wyrazić stosując wyłącznie NAND albo wyłącznie NOR

Postać sumacyjna APN

dla funkcji 3 zmiennych $y = \Sigma(3,4,5,6,7)$

$$y = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 \bar{x}_0 + x_2 \bar{x}_1 x_0 + x_2 x_1 \bar{x}_0 + x_2 x_1 x_0$$

Implikant (np $\bar{x}_2 x_1 x_0$) – taki iloczyn zmiennych, że dla wszystkich wektorów binarnych, dla których przyjmuje wartość 1, funkcja też przyjmuje wartość 1

Postać iloczynowa KPN

dla funkcji 3 zmiennych $y = \Pi(0,1,2)$

$$y = (x_2 + x_1 + x_0)(x_2 + x_1 + \bar{x}_0)(x_2 + \bar{x}_1 + x_0)$$

Implicent (np. $x_2 + \bar{x}_1 + x_0$) – taka suma zmiennych, że dla wszystkich wektorów binarnych, dla których przyjmuje wartość 0, funkcja też przyjmuje wartość 0

Minimalizacja funkcji

przekształcenia algebraiczne

$y = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 \bar{x}_0 + x_2 \bar{x}_1 x_0 + x_2 x_1 \bar{x} + x_2 x_1 x_0$

$y = \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 + x_2 x_1$

$y = \bar{x}_2 x_1 x_0 + x_2$ prawo pochłaniania

$y = x_1 x_0 + x_2$

minimalizacja na mapie Karnaugha

$y = \sum(3, 4, 5, 6, 7)$

$x_2 \backslash x_1 x_0$	00	01	11	10
0	0_{0}	0_{1}	1_{3}	0_{2}
1	1_{4}	1_{5}	1_{7}	1_{6}

Kod Graya

2 sąsiednie słowa kodowe różnią się tylko 1 bitem

pierwsze i ostatnie słowo kodowe różni się tylko 1 bitem

→ Sklejanie jedynek → zapisuję te bity, które nie ulegają zmianie

pola mapy sąsiadują przez ściankę (np. 4 z 6)

$\left.\begin{array}{l} x_1 x_0 \\ x_2 \end{array}\right\}$ afirmacje

$x_2 + x_1 x_0$ → minimalna postać funkcji

po użyciu wszystkich jedynek zostaje postać minimalna

Mapa Karnaugha 4 zmiennych

$y = \sum(1, 3, 5, 6, 9, 11, 12, 13, 14)$

$x_3 x_2 \backslash x_1 x_0$	00	01	11	10
00	0	1	1	0
01	0	1	0	1
11	1	1	0	1
10	0	1	1	0

$\bar{x}_1 x_0$

$x_2 x_1 \bar{x}_0$

$x_3 x_2 \bar{x}_1$

$\bar{x}_2 x_0$ → sąsiedztwo jest też przez krawędź (torus)

grupować można prostokąty $2^n \times 2^m$

$y = \bar{x}_1 x_0 + x_2 x_1 \bar{x}_0 + x_3 x_2 \bar{x}_1 + \bar{x}_2 x_0$

Dla KPN sklejają się zera

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	1	1	0
01	0	1	0	1
11	1	1	0	1
10	0	1	1	0

$(\bar{x}_2 + \bar{x}_1 + \bar{x}_0)$

$(x_3 + x_1 + x_0)$

$(x_2 + x_0)$

KPN: $y = (\bar{x}_2 + \bar{x}_1 + x_0)(x_3 + x_1 + x_0)(x_2 + x_0)$

Dla 5 zmiennych

$x_4x_3 \backslash x_2x_1x_0$	000	001	011	010	110	111	101	100
00								
01								
11								
10								

sąsiedztwo przez oś symetrii na środku

Dla więcej niż 5 zmiennych mapa robi się zbyt skomplikowana

Metoda Quine'a - McCluskey'a

→ powszechnie implementowany algorytm

1) Grupowanie implikantów prostych wg. liczby jedynek

2) Sklejanie implikantów

3) Tabela Quine'a

4) Wybór minimalnego zbioru implikantów pokrywającego wszystkie jedynki funkcji

Dla $y = \sum(3, 4, 5, 6, 7)$

1) Zapisuję binarnie jedynki funkcji

3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

grupuję wg. liczby jedynek w zapisie →

4	1	0	0
3	0	1	1
5	1	0	1
6	1	1	0
7	1	1	1

2) Sklejanie implikantów

✓	4	1	0	0
✓	3	0	1	1
✓	5	1	0	1
✓	6	1	1	0
✓	7	1	1	1

✓	4, 5	1	0	-
✓	4, 6	1	-	0
	3, 7	-	1	1
✓	5, 7	1	-	1
✓	6, 7	1	1	-

4, 5, 6, 7	1	-	-
~~4, 5, 6, 7~~	~~1~~	~~-~~	~~-~~

→ duplikat

Każdy wiersz z grupy n porównuję z każdym z grupy n+1
Jeśli różnią się 1 bitem, to przepisuję dalej ze znakiem – a pierwotne wiersze odznaczam
Tak samo z kolejną tabelą ale – muszą być na jednakowych miejscach
Powtarzam tak długo aż zostanie tylko 1 albo nie można dalej grupować
Nieodznaczone wiersze trafiają do tabeli Quine'a

3)

implikanty				3 011	4 100	5 101	6 110	7 111	
3, 7	-	1	1	X				X	✓
4, 5, 6, 7	1	-	-		X	X	X	X	✓

Implikant zasadniczy - jedyny, który pokrywa jakąś jedynkę, musi trafić do końcowego wyniku

Zaznaczam jedynki pokrywane przez implikanty
Znajduję minimalne pokrycie

$$y = x_1 x_0 + x_2$$

Funkcja nie w pełni określona

$$y = \sum(2, 4, 12, 14, 15, (1, 3, 6, 8))$$

→ wartości dla których nie jest określona

$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	0	–	–	1
10	1	0	0	–
11	1	0	1	1
01	–	0	0	0

– to wartość nieokreślona, można ją sklejać ale nie trzeba

$\bar{x}_3 \bar{x}_2 x_1 + x_1$

metoda Quine'a - McCluskey'a dla funkcji nie w pełni określonej

$$y = \sum(2, 4, 12, 14, 15, (1, 3, 6, 8))$$

1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
6	0	1	1	0
8	1	0	0	0
12	1	1	0	0
14	1	1	1	0
15	1	1	1	1

✓	1	0	0	0	1
✓	2	0	0	1	0
✓	4	0	1	0	0
✓	8	1	0	0	0
✓	3	0	0	1	1
✓	6	0	1	1	0
✓	12	1	1	0	0
✓	14	1	1	1	0
✓	15	1	1	1	1

	1, 3	0	0	–	1
	2, 3	0	0	1	–
	2, 6	0	–	1	0
✓	4, 6	0	1	–	0
✓	4, 12	–	1	0	0
	8, 12	1	–	0	0
✓	6, 14	–	1	1	0
✓	12, 14	1	1	–	0
	14, 15	1	1	1	–

4, 6, 12, 14	–	1	–	0
~~4, 6, 12, 14~~	~~–~~	~~1~~	~~–~~	~~0~~

		2 (0010)	4 (0100)	12 (1100)	14 (1110)	15 (1111)	
1, 3	0 0 – 1						
2, 3	0 0 1 –	X					✓ } dowolnie który
2, 6	0 – 1 0	X					
~~8, 12~~	~~1 – 0 0~~			~~X~~			→ wiersz zdominowany przez -1-0
14, 15	1 1 1 –				X	X	✓
4, 6, 12, 14	– 1 – 0		X	X	X		✓

$$y = \bar{x}_3 \bar{x}_2 x_1 + x_3 x_2 x_1 + x_2 \bar{x}_0$$

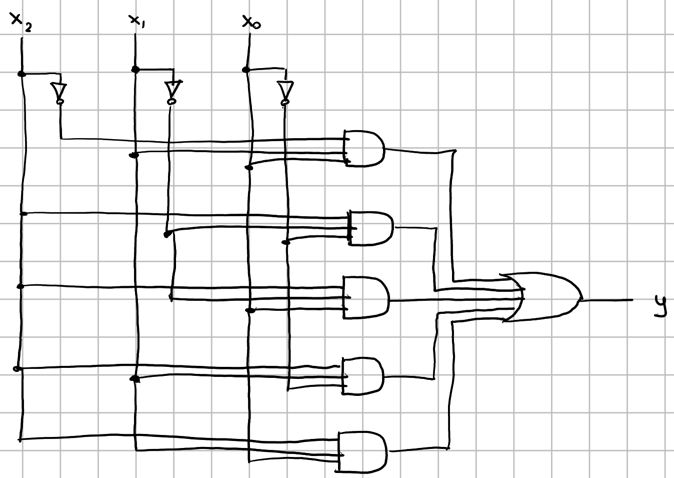
Realizacja postaci minimalnej

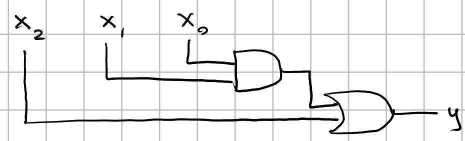
- najprostsza na bramkach logicznych
- na bramkach AND, OR, NOT – sposób podstawowy
- realizacja na zestawie funkcjonalnie pełnym
  → postać APN – na bramkach NAND
  → postać KPN – na bramkach NOR
- realizacje postaci funkcji z faktoryzacją rzadko są efektywne

$y = \Sigma(3, 4, 5, 6, 7)$

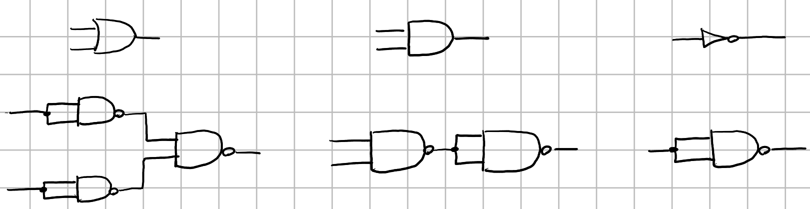
$y = \overline{x}_2 x_1 x_0 + x_2 \overline{x}_1 \overline{x}_0 + x_2 \overline{x}_1 x_0 + x_2 x_1 \overline{x}_0 + x_2 x_1 x_0$

$y = x_1 x_0 + x_2$ – postać minimalna





Zestaw funkcjonalnie pełny NAND



Zestaw funkcjonalnie pełny NOR