EE1410: Digitale Systemen

BSc. EE, 1e jaar, 2012-2013, 3e college

Arjan van Genderen, Stephan Wong, Computer Engineering

18-2-2013

Hoorcollege 3

Canonieke vorm two-level netwerken

- Som van Producten vs. Product van Sommen

Two-level vereenvoudiging

- Karnaugh-maps

Multi-level netwerken

- factorisatie
- afbeelding op NAND-NAND- en NOR-NOR netwerken
- afbeelding op AND-OR-inv- en OR-AND-inv poorten

Timing in combinatorische netwerken

- poort vertragingstijden, spikes

Corresponderende stof in boek "Digital Logic":

2.6 – 2.7, 3.8.5 tot aan "Equations 3.1 and 3.2 specify ….",
4 – 4.7, 4.11

Canonieke vormen

Canonieke vorm

- **Expressie ⇒ unieke waarheidstabel**
- **Waarheidstabel ⇒ vele alternatieve expressions**
- **Canonieke vorm: unieke standaardvorm voor expressies**
  - **Som-van-Producten, of**
  - **Product-van-Sommen**

Som-van-Producten vorm:

A	B	C	F	F'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

0 1 1 1 0 0 1 0 1 1 1 0 1 1 1

F = A' B C + A B' C' + A B' C + A B C' + A B C

F' = A' B' C' + A' B' C + A' B C'

0 0 0 0 0 1 0 1 0

Som van Producten

A	B	C	*Mintermen*	F
0	0	0	A'B'C' = m_0	0
0	0	1	A'B'C = m_1	0
0	1	0	A'BC' = m_2	0
0	1	1	A'BC = m_3	1
1	0	0	AB'C' = m_4	1
1	0	1	AB'C = m_5	1
1	1	0	ABC' = m_6	1
1	1	1	ABC = m_7	1

Canonieke vorm:

$$F(A,B,C) = A' B C + A B' C' + A B' C + A B C' + A B C$$

$$= m_3 + m_4 + m_5 + m_6 + m_7$$

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

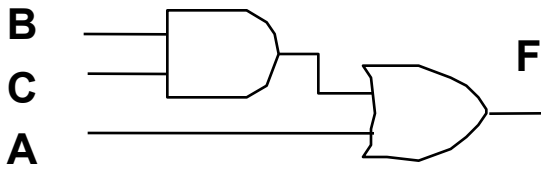
$$F'(A,B,C) = A' B' C' + A' B' C + A' B C'$$

$$= m_0 + m_1 + m_2$$

$$= \Sigma\, m(0,1,2)$$

Canonieke vorm naar minimale vorm:

$$F = A B' (C + C') + A' B C + A B (C' + C)$$
$$= A B' + A' B C + A B$$
$$= A (B' + B) + A' B C$$
$$= A + A' B C$$
$$= A + B C$$



Product van Sommen

A	B	C	*Maxtermen*		F
0	0	0	A+B+C	$= M_0$	0
0	0	1	A+B+C’	$= M_1$	0
0	1	0	A+B’+C	$= M_2$	0
0	1	1	A+B’+C’	$= M_3$	1
1	0	0	A’+B+C	$= M_4$	1
1	0	1	A’+B+C’	$= M_5$	1
1	1	0	A’+B’+C	$= M_6$	1
1	1	1	A’+B’+C’	$= M_7$	1

$$F(A,B,C) = \Pi\, M(0,1,2) = M_0\, M_1\, M_2$$
$$= (A + B + C)\,(A + B + C')\,(A + B' + C)$$

$$F'(A,B,C) = \Pi\, M(3,4,5,6,7) = M_3\, M_4\, M_5\, M_6\, M_7$$
$$= (A + B' + C')\,(A' + B + C)\,(A' + B + C')\,(A' + B' + C)\,(A' + B' + C')$$

Two-level canonieke vormen

- **(Som van Producten)' ⇒ Product van Sommen:**

$$F = m_3 + m_4 + m_5 + m_6 + m_7$$
$$= A'\,B\,C + A\,B'\,C' + A\,B'\,C + A\,B\,C' + A\,B\,C$$

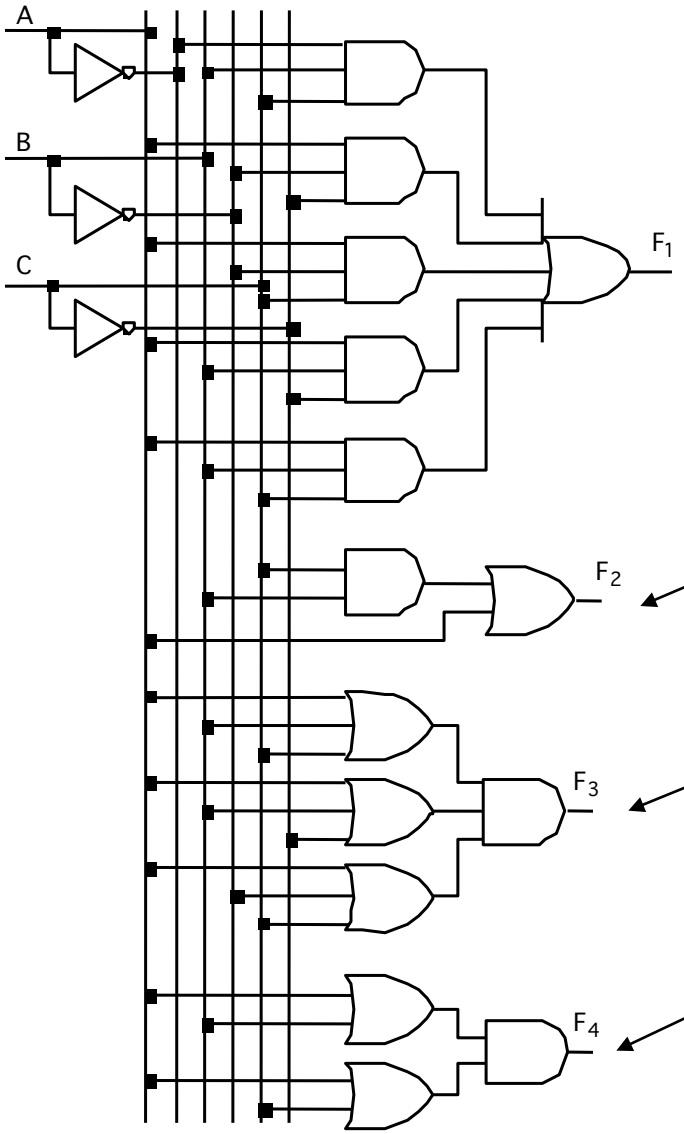
$$F' = (A'\,B\,C + A\,B'\,C' + A\,B'\,C + A\,B\,C' + A\,B\,C)'$$
$$= (A'\,B\,C)'\,(A\,B'\,C')'\,(A\,B'\,C)'\,(A\,B\,C')\,(A\,B\,C)'$$
$$= (A + B' + C')\,(A' + B + C)\,(A' + B + C')\,(A' + B' + C)\,(A' + B' + C')$$
$$= M_3\,M_4\,M_5\,M_6\,M_7$$

- **(Product van Sommen)' ⇒ Som van Producten:**

$$F = M_0\,M_1\,M_2$$
$$= (A + B + C)\,(A + B + C')\,(A + B' + C)$$

$$F' = \{(A + B + C)\,(A + B + C')\,(A + B' + C)\}'$$
$$= (A + B + C)' + (A + B + C')' + (A + B' + C)'$$
$$= A'\,B'\,C' + A'\,B'\,C + A'\,B\,C'$$
$$= m_0 + m_1 + m_2$$

Alternatieve implementaties van F



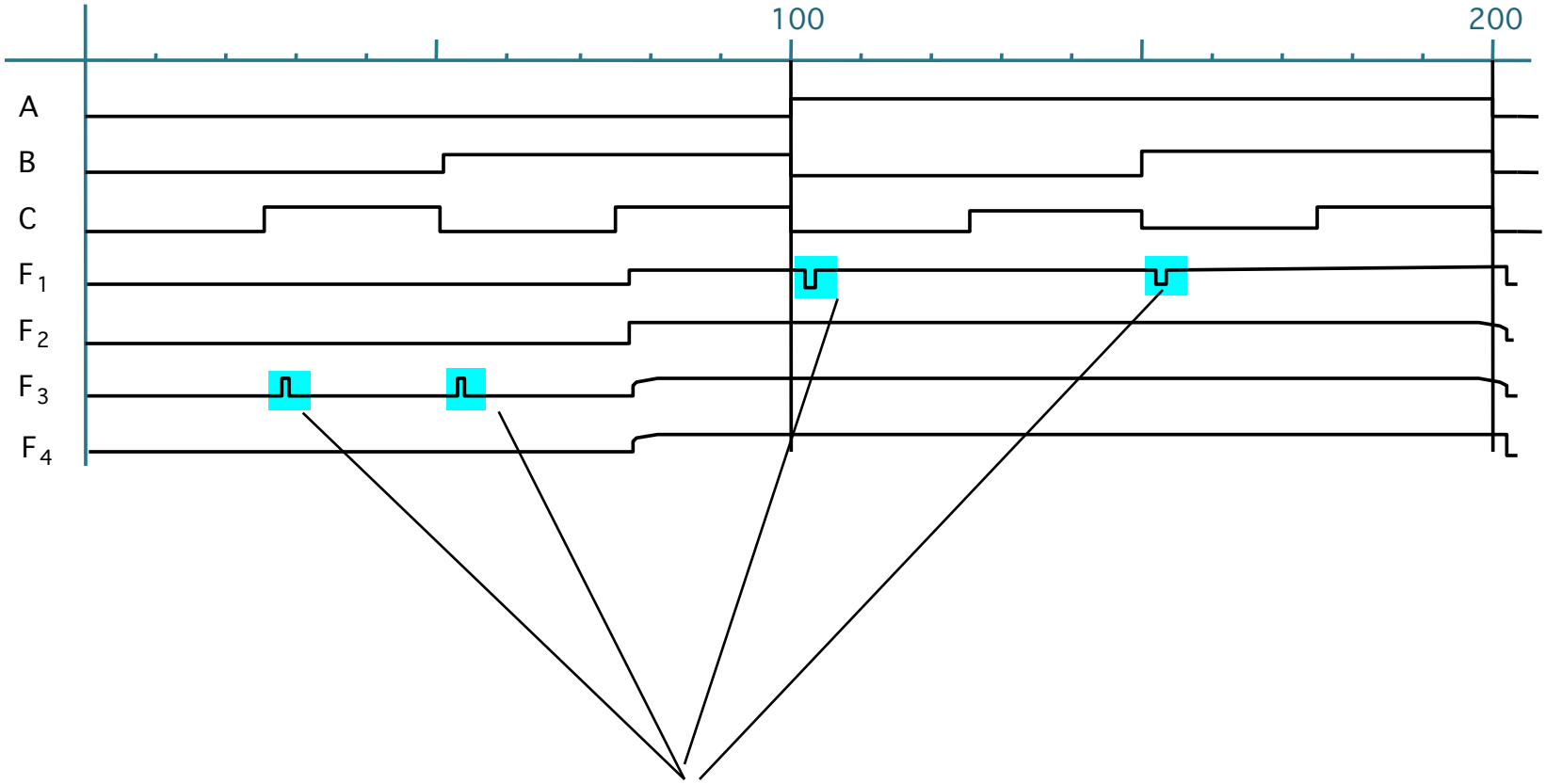
Canonieke Som van Producten

Minimale Som van Producten

Canoniek Product van Sommen

Minimaal Product van Sommen

Vergelijking tijdsgedrag



Afgezien van *timing glitches* zijn de verschillende implementaties identiek

Onvolledig gespecificeerde functies

n-input functie ⇒ 2^n mogelijke ingangscombinaties
niet alle mogelijkheden zijn altijd relevant

Voorbeeld: Binary-Coded-Decimal-Digit-Increment-by-1

	inputs					outputs			
Digit	A	B	C	D	Digit	W	X	Y	Z
0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	2	0	0	1	0
2	0	0	1	0	3	0	0	1	1
3	0	0	1	1	4	0	1	0	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	6	0	1	1	0
6	0	1	1	0	7	0	1	1	1
7	0	1	1	1	8	1	0	0	0
8	1	0	0	0	9	1	0	0	1
9	1	0	0	1	0	0	0	0	0
	1	0	1	0		X	X	X	X
	1	0	1	1		X	X	X	X
	1	1	0	0		X	X	X	X
	1	1	0	1		X	X	X	X
	1	1	1	0		X	X	X	X
	1	1	1	1		X	X	X	X

ingangscombinaties waarbij Z = 0:
Off-set **van Z**

ingangscombinaties waarbij Z = 1:
On-set **van Z**

ingangscombinaties waarbij Z = X:
Don't care (DC) set **van Z**

X = don't care (waarde is niet relevant)

$Z = m_0 + m_2 + \ldots + m_8 + d_{10} + d_{11} + \ldots + d_{15} = M_1 M_3 \ldots M_9 D_{10} D_{11} \ldots D_{15}$

Two-level vereenvoudiging

Two-level vereenvoudiging

Algebraïsche vereenvoudiging:

- geen vaste procedure
- hoe weet je dat minimale vorm is gevonden?

Vereenvoudiging van two-level netwerken m.b.v. Karnaugh-maps (zie hierna) :

- systematische manier
- altijd minimale vorm

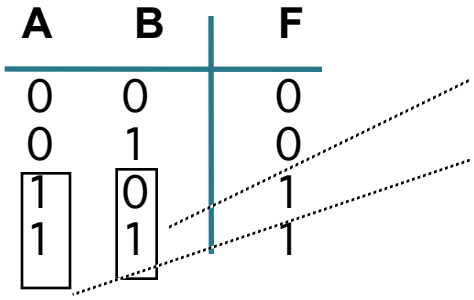
Computer-Aided Design (CAD) Tools (gereedschappen):

- optimale vereenvoudigingen kosten veel rekentijd
  met name voor functies met veel ingangsvariabelen (>10)
- gewoonlijk gebruikt men daarom benaderende methoden
  - minder rekentijd benodigd
  - oplossingen zijn niet optimaal maar meestal wel acceptabel

Handmatige vereenvoudiging blijft belangrijk:

- bij kleine circuits: handmatige vereenvoudiging geeft meer inzicht
- inzicht in CAD programma's (espresso, Xilinx ISE)
- mogelijkheid tot controle van CAD resultaten (voor kleine circuits)
- geen CAD programma's beschikbaar tijdens het tentamen....

Vereenvoudiging

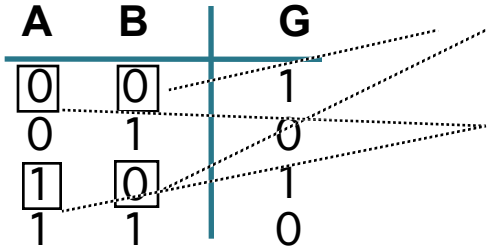


B-waarden veranderen binnen de on-set rijen

A-waarden veranderen niet binnen de on-set rijen

B wordt geëlimineerd, A blijft

immers: F = A B' + A B = A (B' + B) = A



B-waarden veranderen niet binnen de on-set rijen

A-waarden veranderen binnen de on-set rijen

A wordt geëlimineerd, B blijft

immers: G = A' B' + A B' = (A' + A) B' = B'

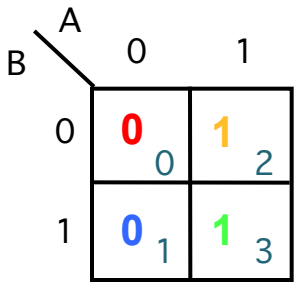
De essentie van vereenvoudiging:

Vindt twee subsets van de ON-set waarin slechts één variable van waarde verandert. Deze variabele kan worden geëlimineerd.

Karnaugh diagrammen (K-maps)

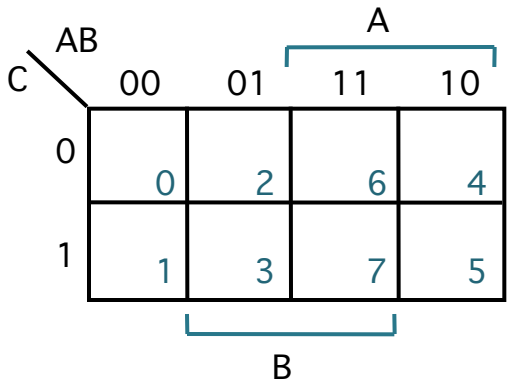
Alternatieve methode om waarheidstabellen te representeren, waarbij tussen 2 "buren" altijd precies één variabele van waarde verandert.

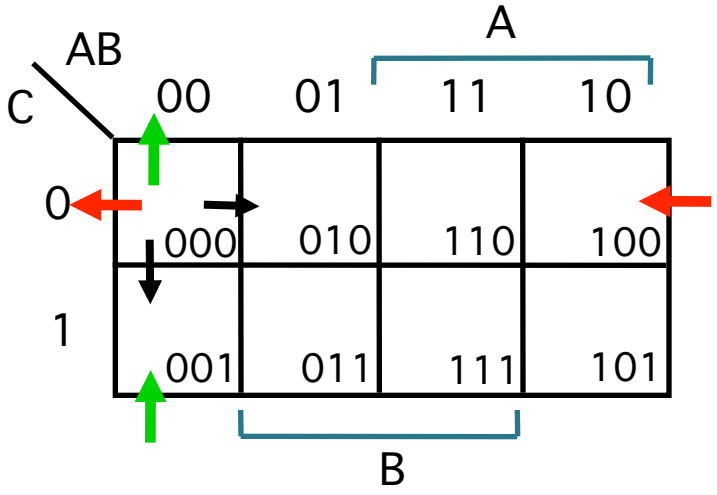
2-variable K-map



A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

3-variable K-map

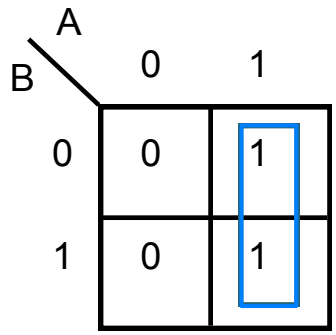




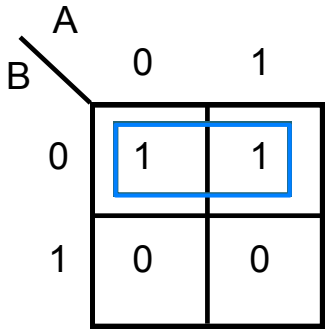
Naburigheden in K-maps:

- **slechts één var. verandert tussen buren**
- **eerste-laatste kolom zijn ook buur**
- **bovenste en onderste rij zijn ook buur**

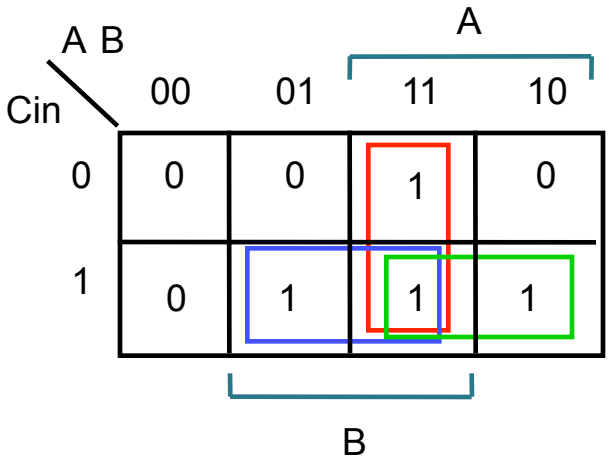
Toepassingsvoorbeelden



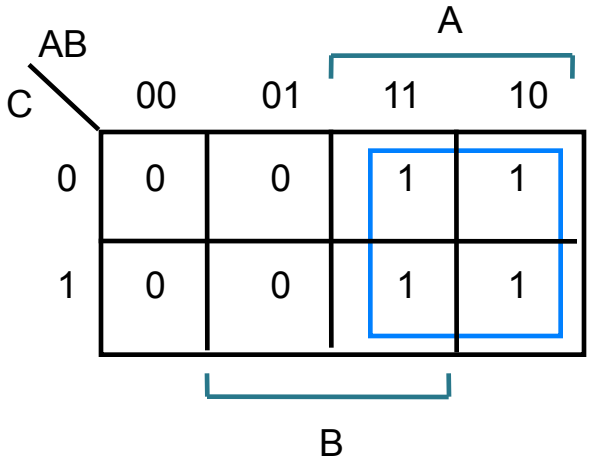
F = A



G = B'



(Fadder): Cout = A B + B Cin + A Cin

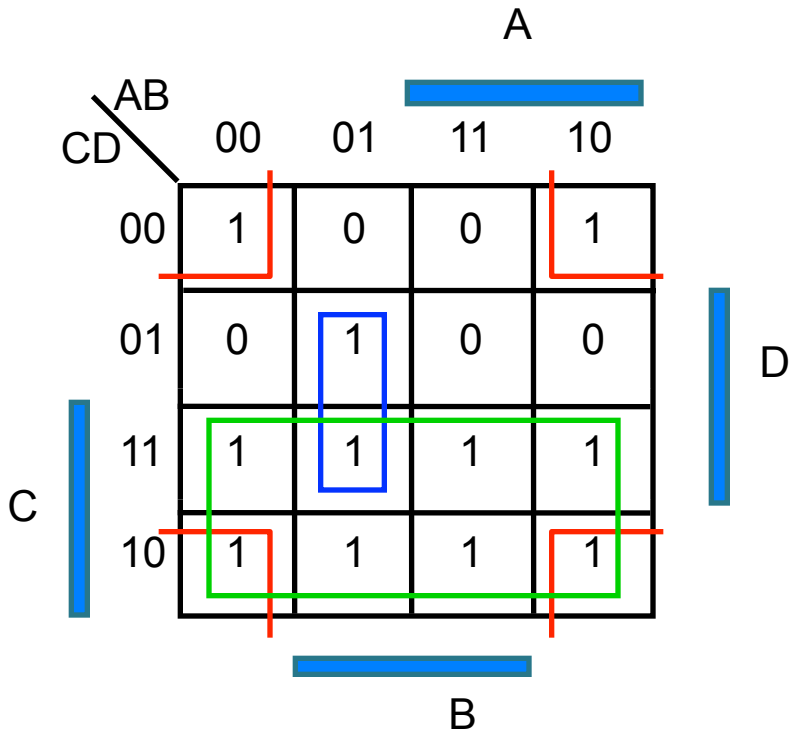


F(A,B,C) = A

Rechthoek met 1-en ⇔ Productterm
Rechthoek met 1-en groter ⇔ Bijbehorende productterm kleiner !

Voorbeeld afleiding minimale som met 4 variabelen

F(A,B,C,D) = Σ m(0,2,3,5,6,7,8,10,11,14,15)



F = C + A' B D + B' D'

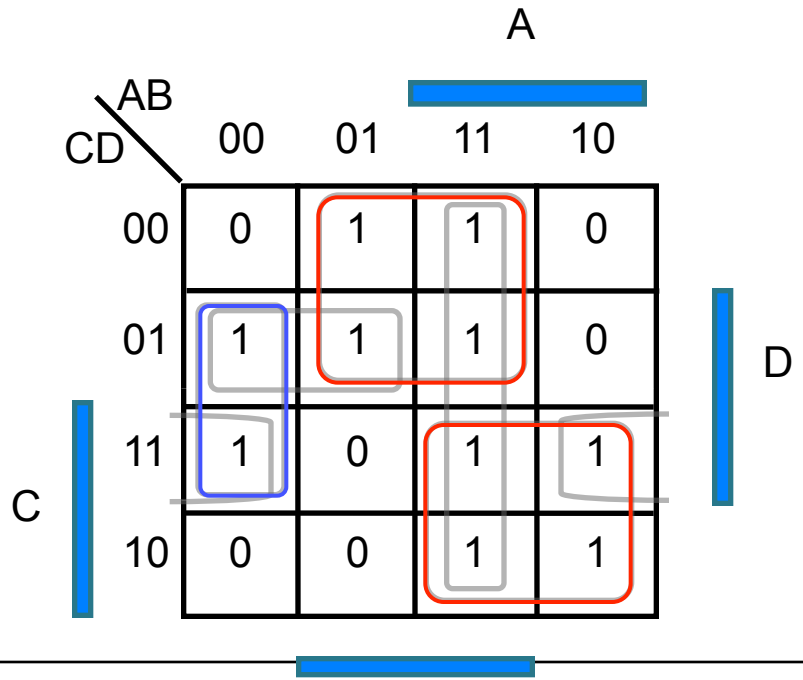
N.B. alleen rechthoeken met 1, 2, 4, 8 … (2^n) énen doen mee

Minimale som van producten:
Vindt het kleinste aantal grootste rechthoeken (subcubes) dat de ON-set beslaat

Want: aantal rechthoeken = aantal producttermen
grotere rechthoek = minder variabelen in productterm

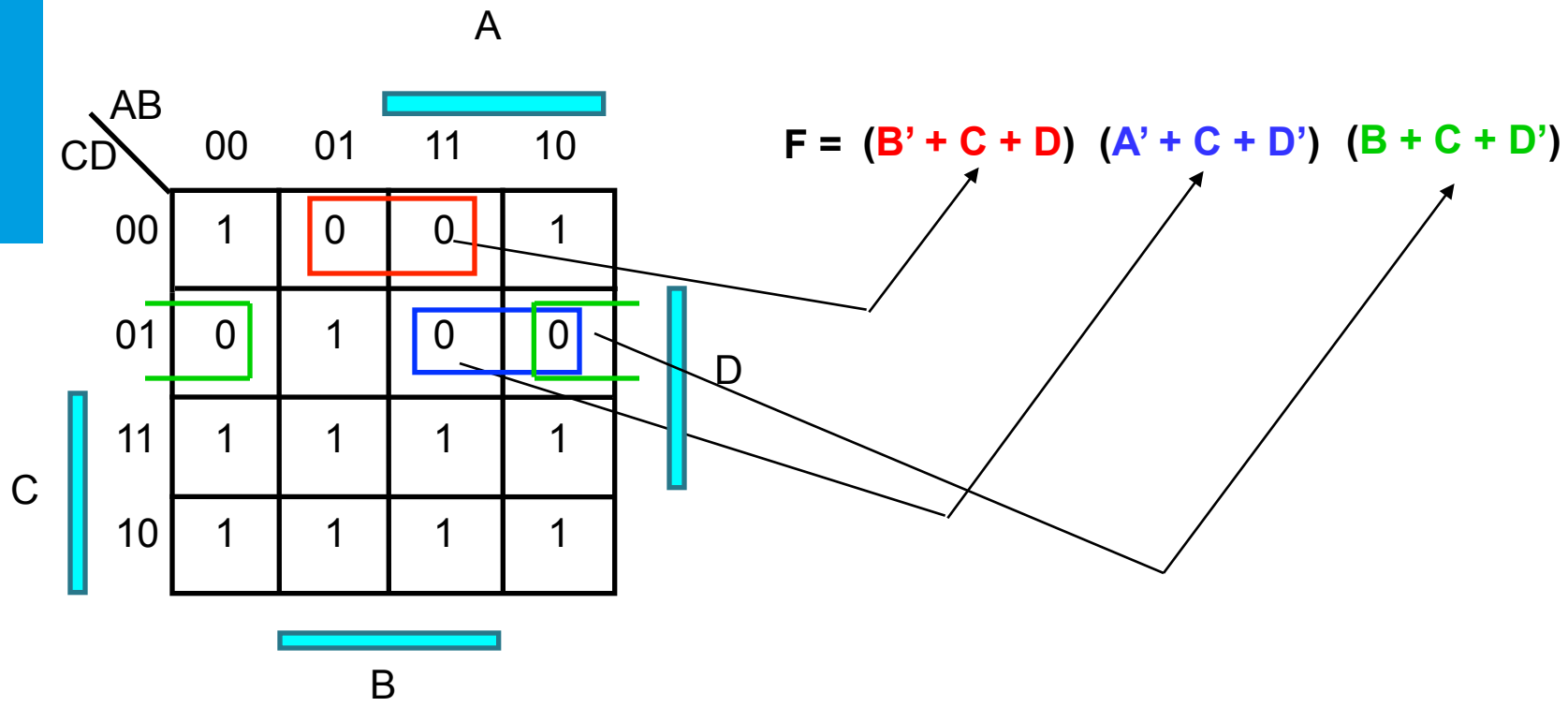
Recept minimale som van producten

- **Vindt alle (maximaal grote) rechthoeken van 1-en: priemimplicanten.**
- **Vindt alle priemimplicanten die als enige een bepaalde 1 afdekken: essentiële priemimplicanten.**
- **De essentiële priemimplicanten komen in ieder geval voor als productterm in de minimale expressie.**
- **Dek alle overgebleven 1-en af met zo weinig mogelijk niet-essentiële priemimplicanten.**



F = BC' + AC + A'B'D

Duale methode: minimaal product van somtermen



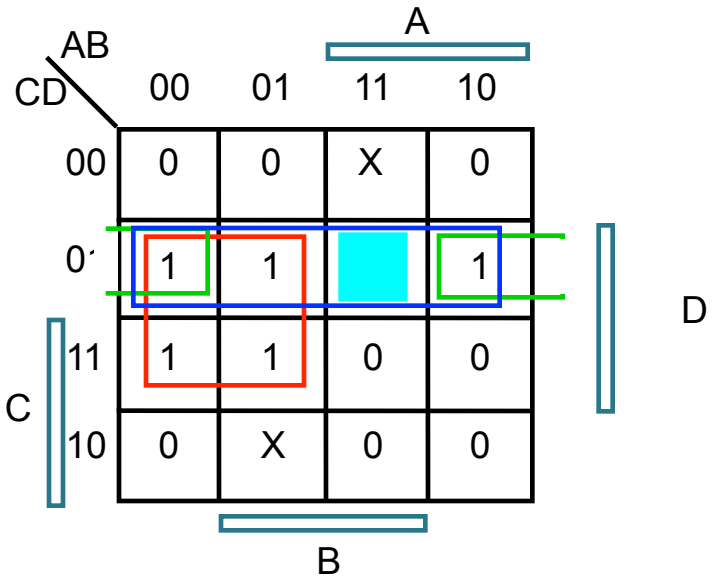
Deze methode is ook als volgt in te zien: Kijk in K-map wanneer F = 0 → F' = 1

F' = B C' D' + A C' D + B' C' D

(F')' = (B C' D' + A C' D + B' C' D)'

F = (B' + C + D) (A' + C + D') (B + C + D')

Don’t Cares

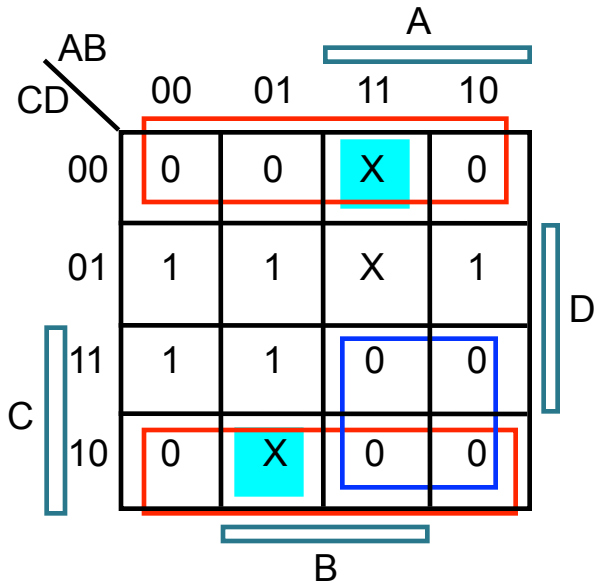


F(A,B,C,D) = Σ m(1,3,5,7,9) + Σ d(6,12,13)

via K-map:

F = A’ D + B' C' D zonder don't cares

F = A' D + C' D met gebruik don't cares



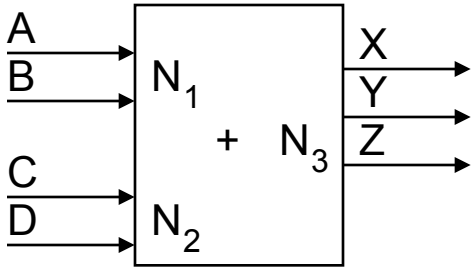
Via duale methode:

F = D (A' + C')

Dus hier (nog) minder termen

Don't Cares kunnen als 1-en of 0-en worden gebruikt indien beter resultaat

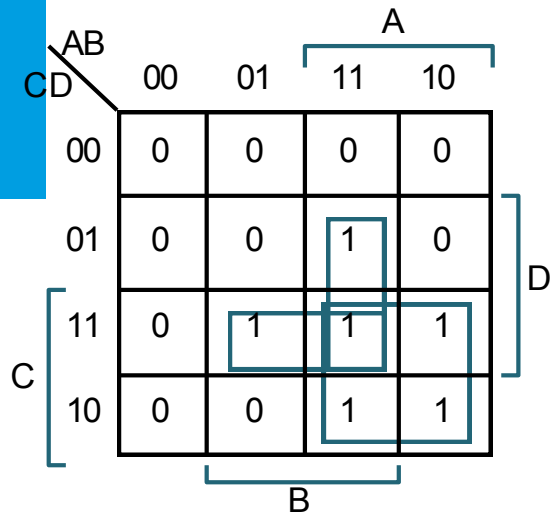
Case Study: Two-Bit Adder



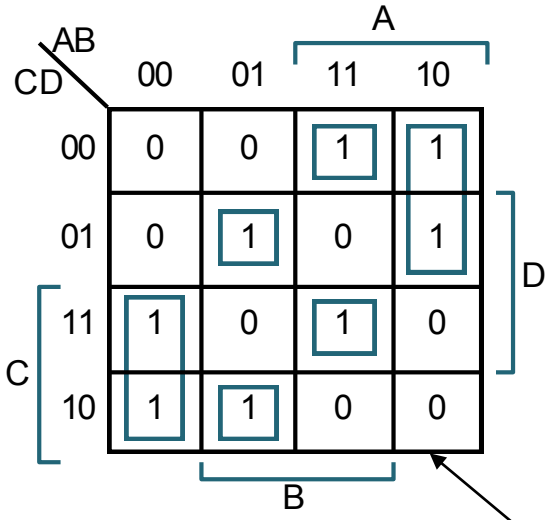
A	B	C	D	X	Y	Z
0	0	0	0	0	0	0
		0	1	0	0	1
		1	0	0	1	0
		1	1	0	1	1
0	1	0	0	0	0	1
		0	1	0	1	0
		1	0	0	1	1
		1	1	1	0	0
1	0	0	0	0	1	0
		0	1	0	1	1
		1	0	1	0	0
		1	1	1	0	1
1	1	0	0	0	1	1
		0	1	1	0	0
		1	0	1	0	1
		1	1	1	1	0

==> 4-variable K-map voor X, Y en Z

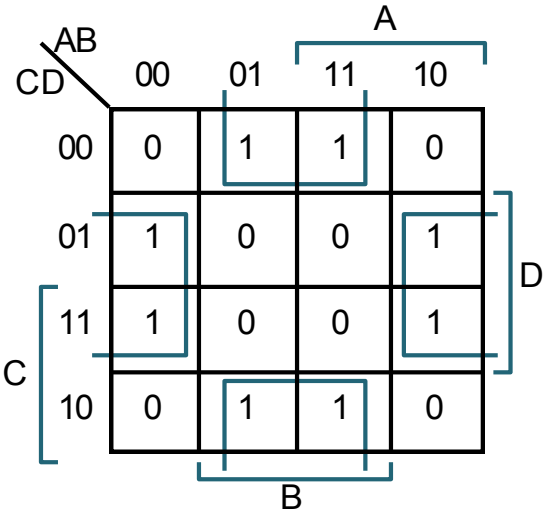
Case Study: Two-Bit Adder



K-map for X



K-map for Y



K-map for Z

X = A C + B C D + A B D

1'en op diagonalen: indicatie voor XORs

Z = B D' + B' D = B xor D

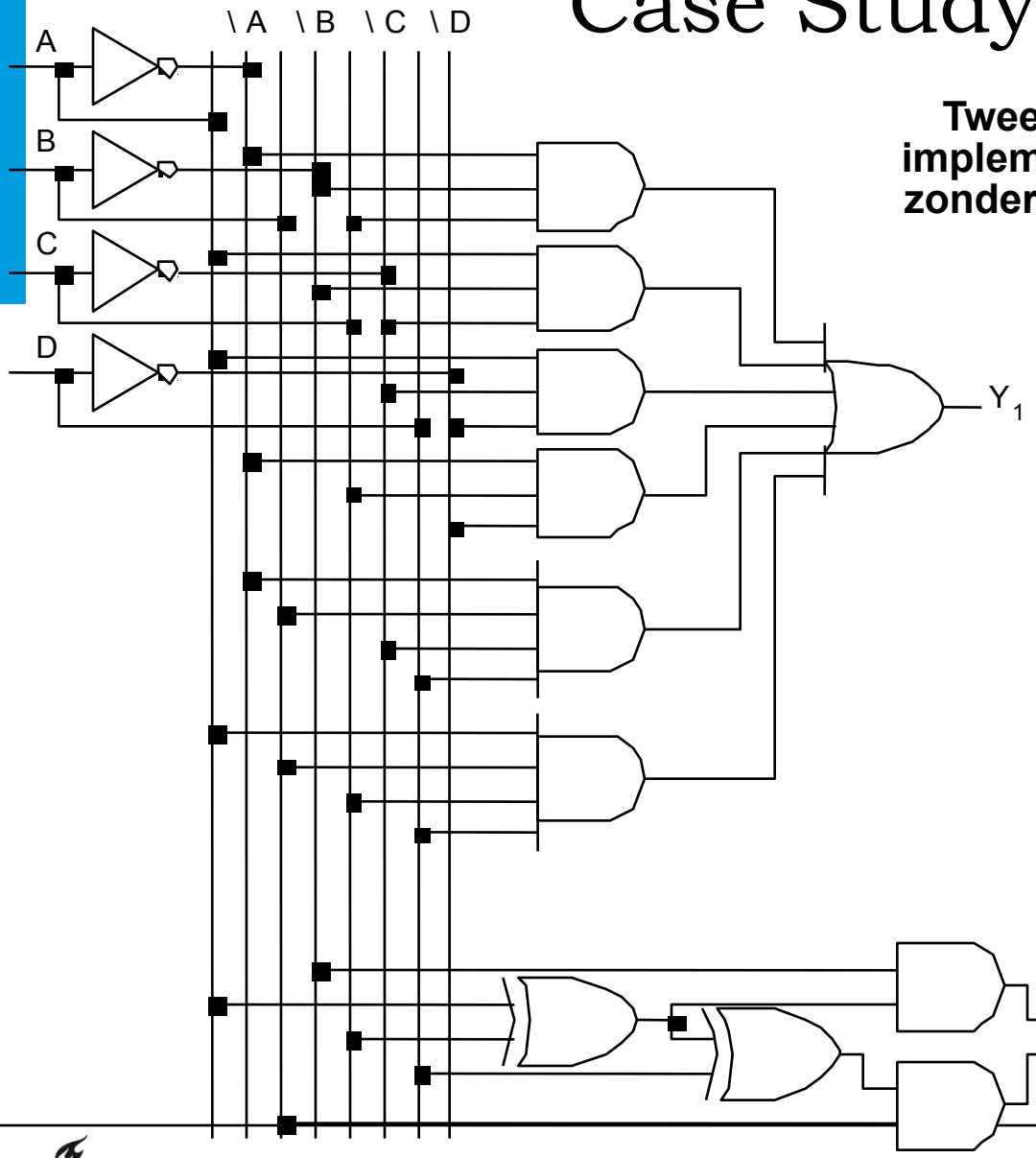
Y = A' B' C + A B' C' + A' B C' D + A' B C D' + A B C' D' + A B C D

= B' (A xor C) + A' B (C xor D) + A B (C xnor D)

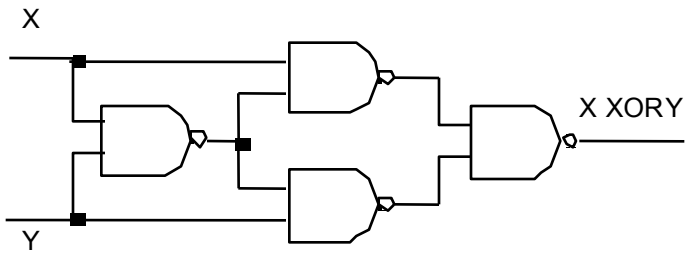
= B' (A xor C) + B (A xor B xor C)

Aantal poorten kan verder worden gereduceerd als XORs worden toegepast!

Case Study: Two-Bit Adder



Twee alternatieve implementaties van Y zonder en met XORs:



NB: XOR implementatie kost wel 4 NAND gates

Samenvatting two-level netwerken

Primitieve logische bouwblokken:

INVERTER, AND, OR

Canonieke vormen:

Som van Producten (mintermen), Product van Sommen (maxtermen) , incl. don't cares

Logische vereenvoudiging:

2-level realisaties met minimum hoeveelheid poorten en/of poortingangen

K-map methode (incl. duale methode)

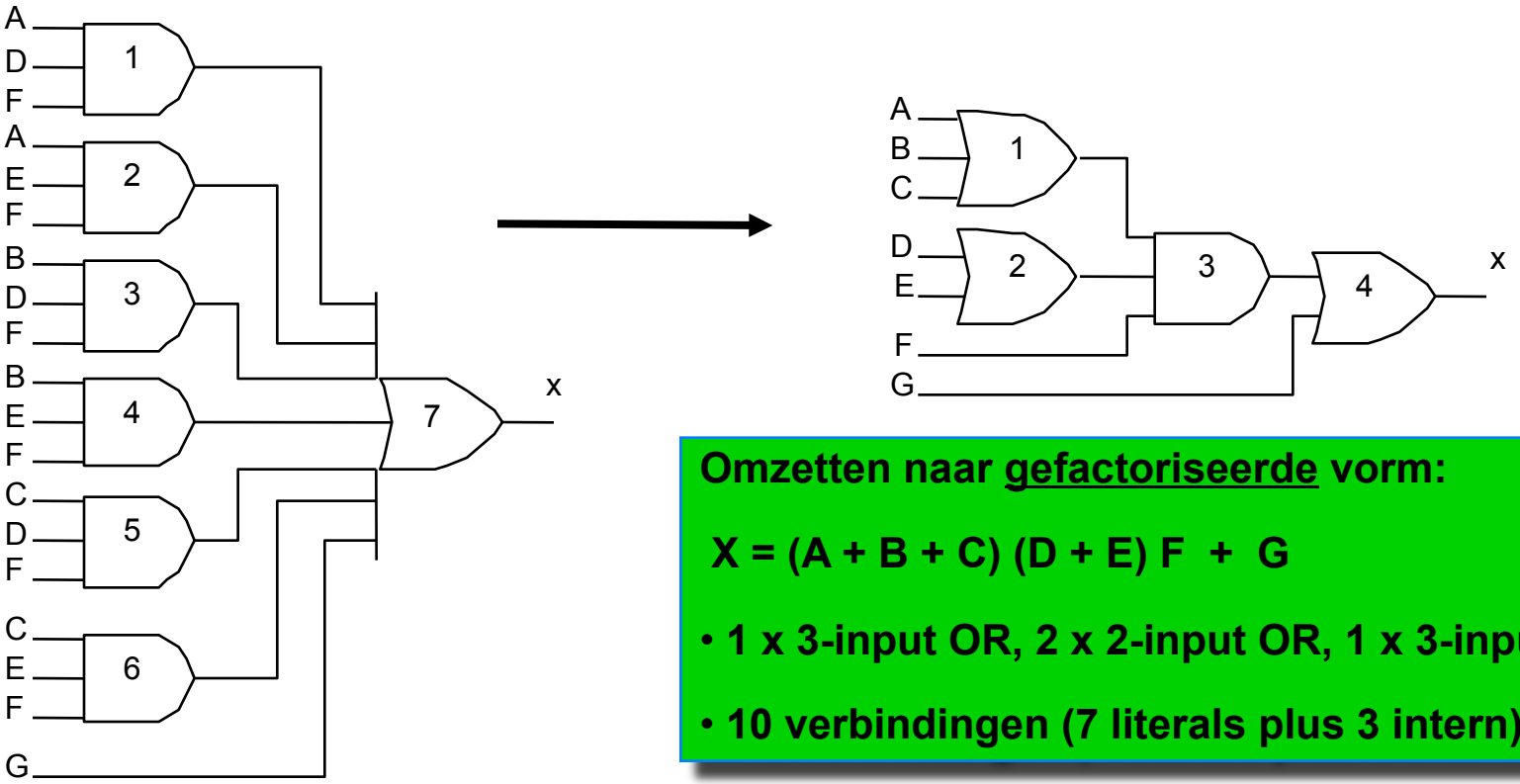
Multi-level netwerken

Voordelen van multilevel netwerken

Som-van-producten vorm (reeds 2-level-gereduceerd!):

X = A D F + A E F + B D F + B E F + C D F + C E F + G

- **6 x 3-input AND poorten + 1 x 7-input OR poort (vaak niet-bestaand)**
- **25 verbindingen (19 literals plus 6 interne verbindingen)**



Omzetten naar gefactoriseerde vorm:

X = (A + B + C) (D + E) F + G

- **1 x 3-input OR, 2 x 2-input OR, 1 x 3-input AND**
- **10 verbindingen (7 literals plus 3 intern)**

Conversie naar NAND/NAND en NOR/NOR netwerken

- **Initieel netwerk vaak uitgedrukt in ANDs en ORs (canonieke vorm)**
- **Echter technologisch direct realiseerbaar: NANDs en NORs**
- **Dus herschrijven we expressies naar netwerken van NANDs en NORs**

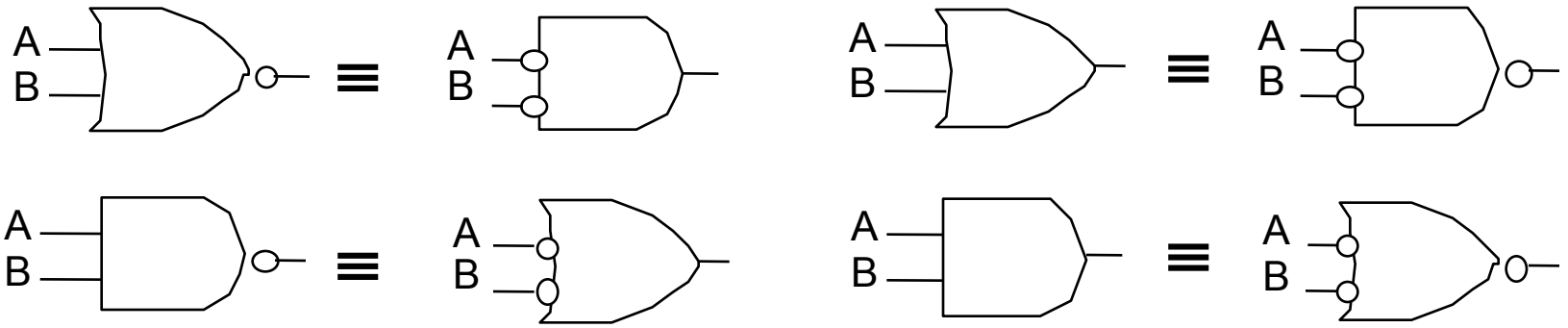
Conversie naar NAND/NAND en NOR/NOR netwerken

De Morgan's wet: $(A + B)' = A' \cdot B' \Rightarrow A + B = (A' \cdot B')'$

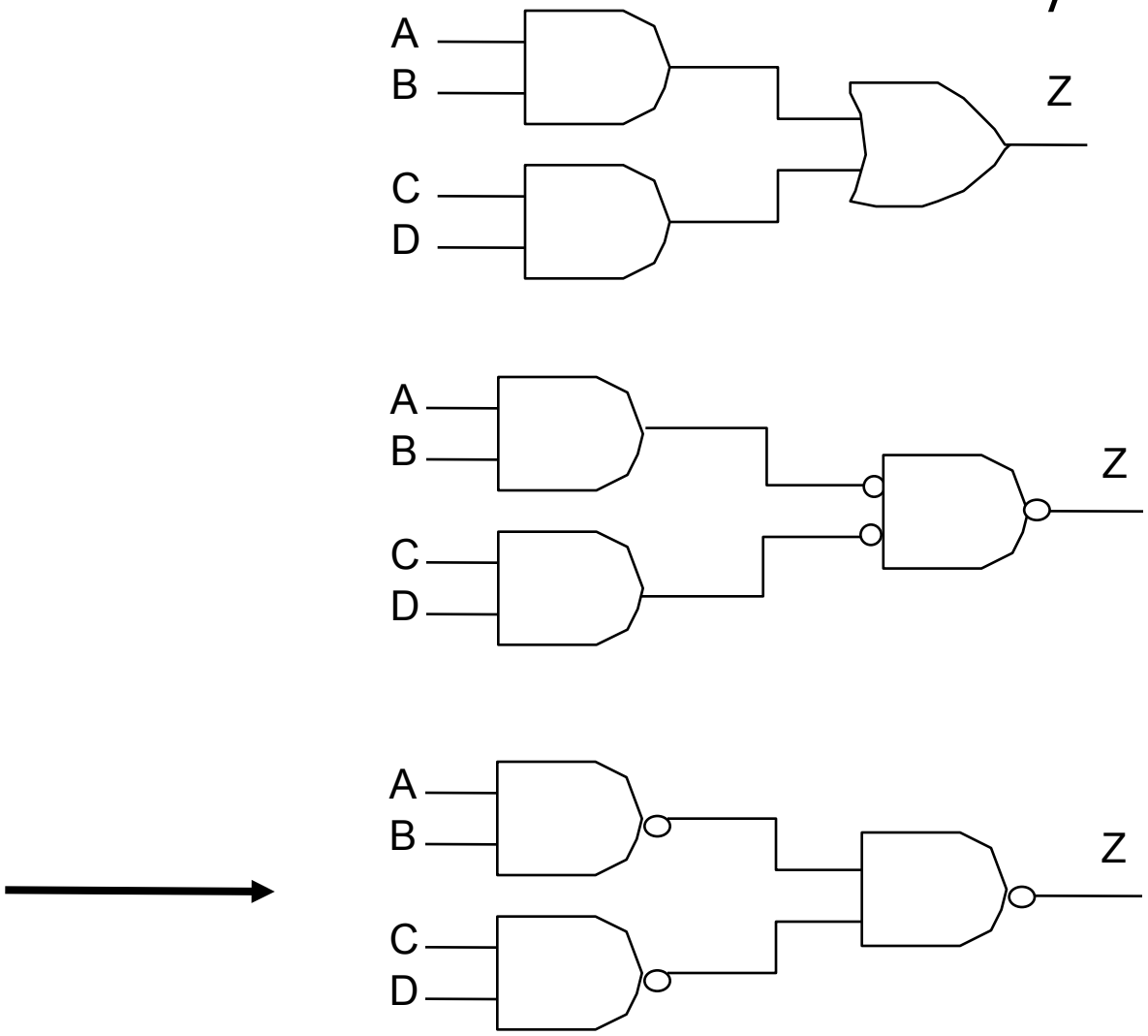
$(A \cdot B)' = A' + B' \Rightarrow A \cdot B = (A' + B')'$

Dus:

- **NOR is hetzelfde als een AND met complementaire ingangen**
- **OR is hetzelfde als NAND met complementaire ingangen**
- **NAND is hetzelfde als een OR met complementaire ingangen**
- **AND is hetzelfde als NOR met complementaire ingangen**



Toepassing: Conversie AND/OR netwerk naar NAND/NAND netwerk



Evenzo is een OR/ AND netwerk om te zetten in een NOR/NOR netwerk

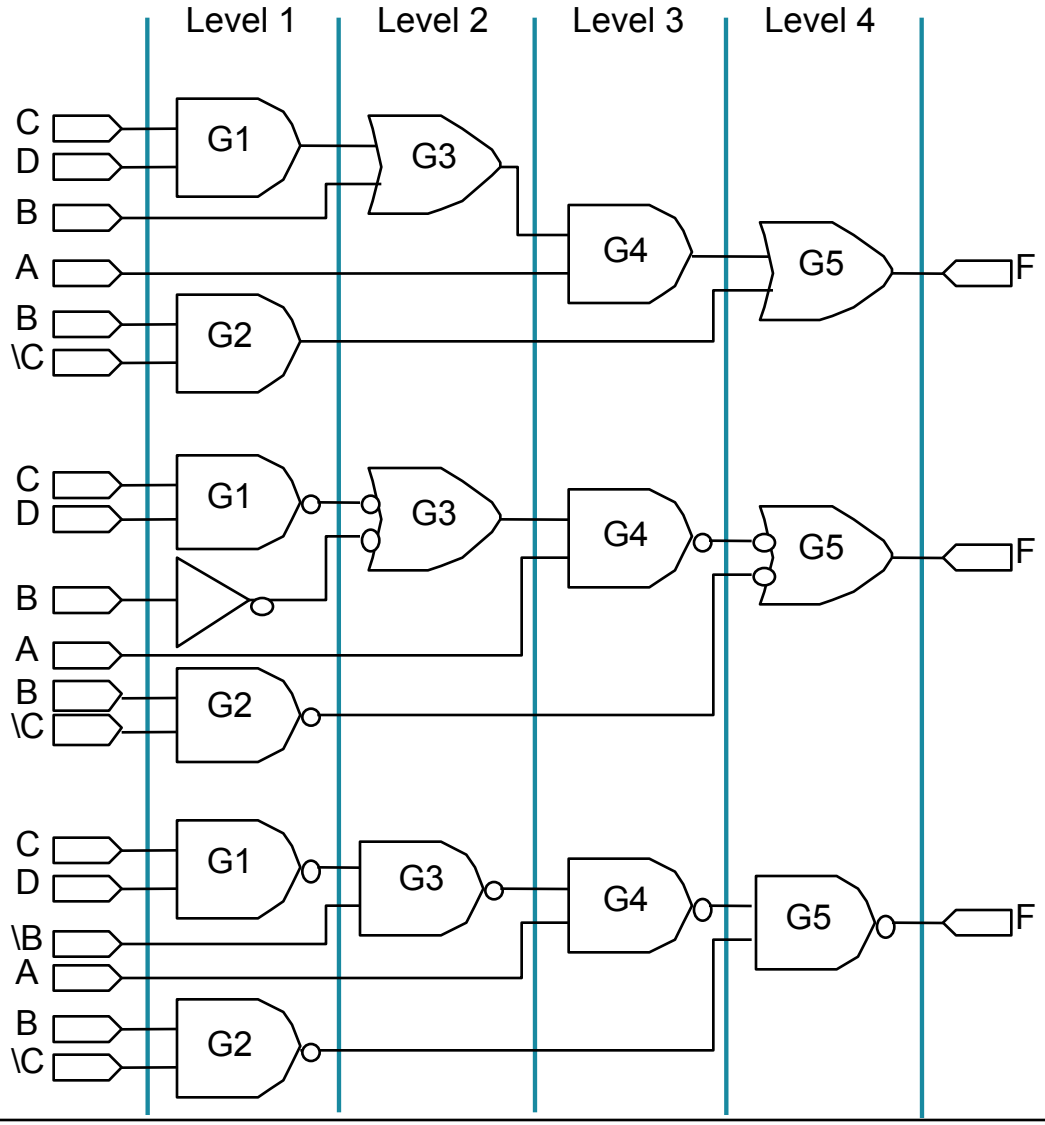
Voorbeeld: multilevel netwerk in NANDs

F = A (B + C D) + B C'

Oorspronkelijke AND-OR netwerk:

Introductie van bubbles:

Omwerken naar NAND poorten:



Recept afbeelden naar NAND/NAND (NOR/NOR) schakeling

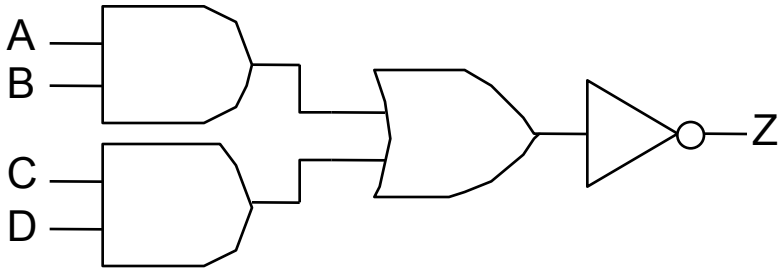
- **Stel waarheidstabel/K-map op**
- **Bepaal minimale som van producten (product van sommen)**
- **Pas eventueel factorisatie toe**
- **Beeld af op NAND/NAND (NOR/NOR)**

(zie voorafgaande slides)

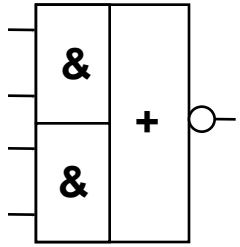
AND-OR-Invert & OR-AND-Invert bouwstenen

Behalve NANDs en NORs blijken ook de AOI en OAI bouwstenen goedkoper/sneller te fabriceren in vergelijking tot een realisatie m.b.v. ANDs, ORs en inverters.

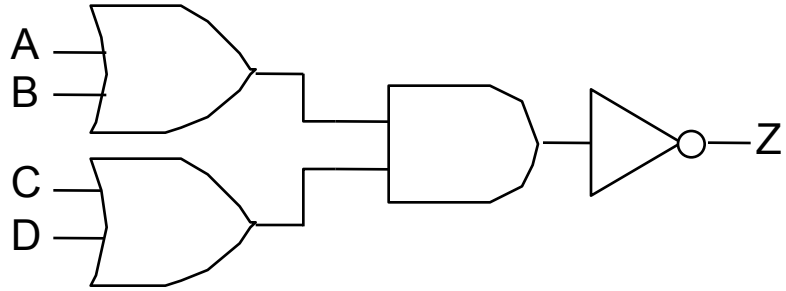
AOI



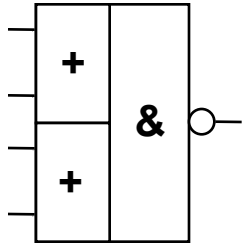
2x2 AOI Schematic Symbol



OAI



2x2 OAI Schematic Symbol

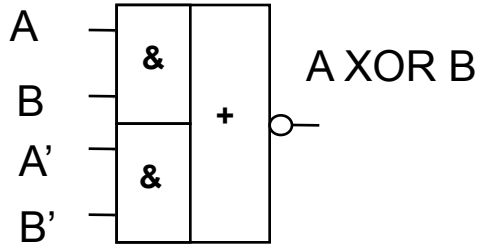


Implementatievoorbeelden

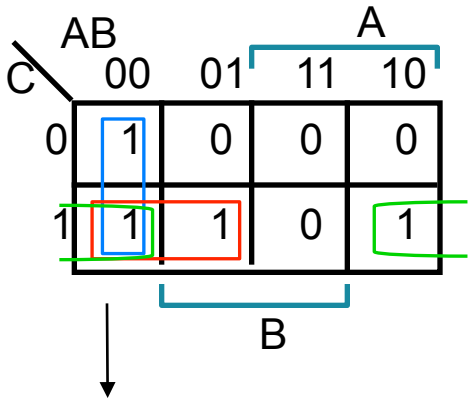
F = A XOR B

F' = (A XOR B)' = (A' B + A B')'

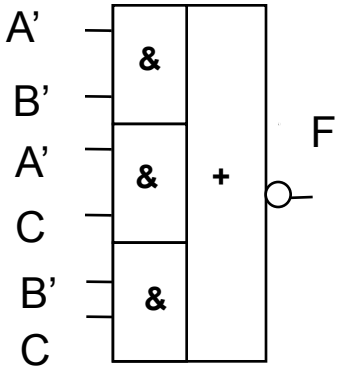
= (A + B') (A' + B) = A B + A' B'

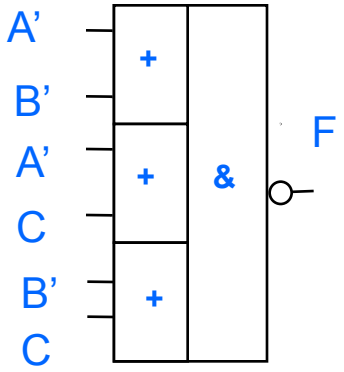


F = ∑ m(2,4,6,7) ⇒ **F' = ∑ m(0,1,3,5)**



F' = A' B' + A' C + B' C

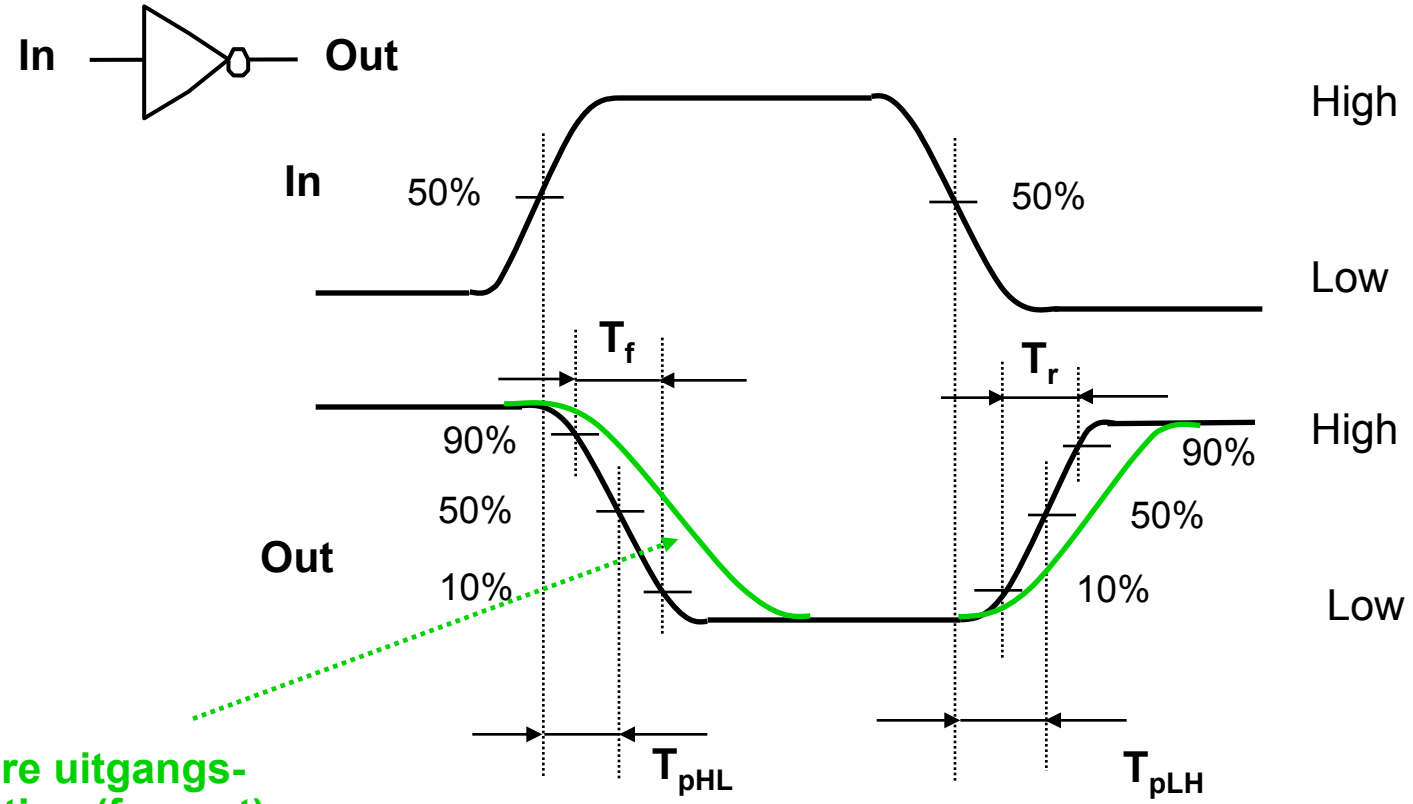




F' = (A' + B') (A' + C) (B' + C)

Timing in combinatoriek

Tijdsresponsie van poorten



Tf : Fall Time H-L overgang
T_{pHL} : propagatietijd H-L overgang

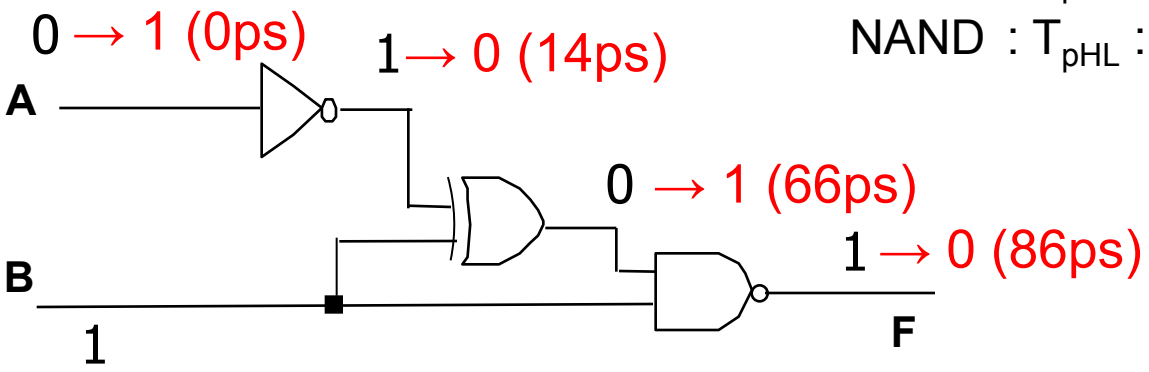
T_r : Rise Time L-H overgang
T_{pLH} : propagatietijd L-H overgang

Vaak nominale, minimum and maximum waarden beschikbaar per type poort

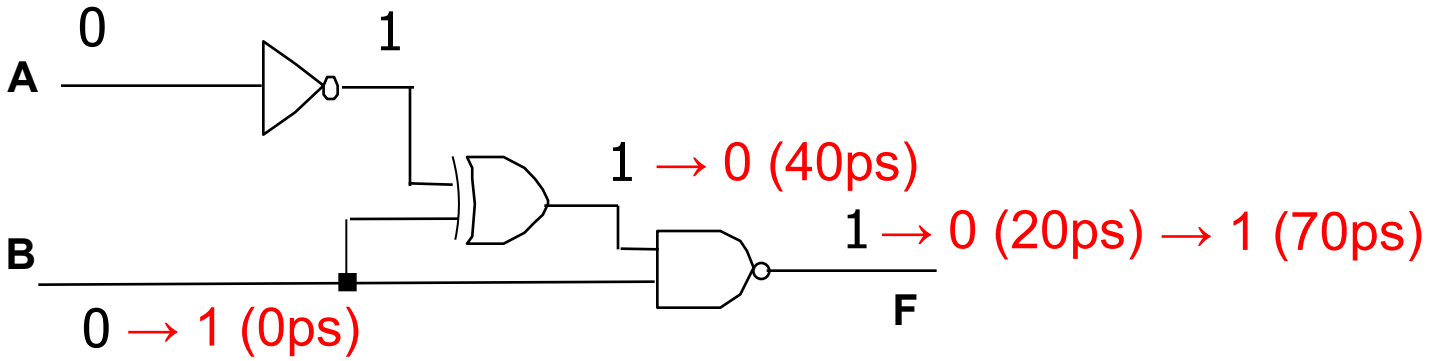
Overgangs of propagatietijd bijv. T_{pHL} = 2.0 + 1.2 x L ps, met L belasting

Tijdsresponsie van combinatorische schakelingen

INV : T_{pHL} : 14ps T_{pLH} : 18ps
EXOR : T_{pHL} : 40ps T_{pLH} : 52ps
NAND : T_{pHL} : 20ps T_{pLH} : 30ps

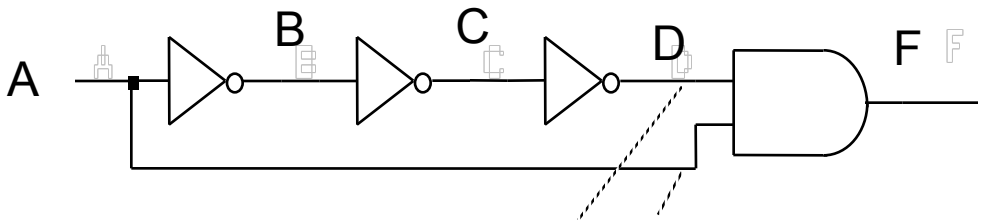


Tijdsvertragingen tellen bij elkaar op

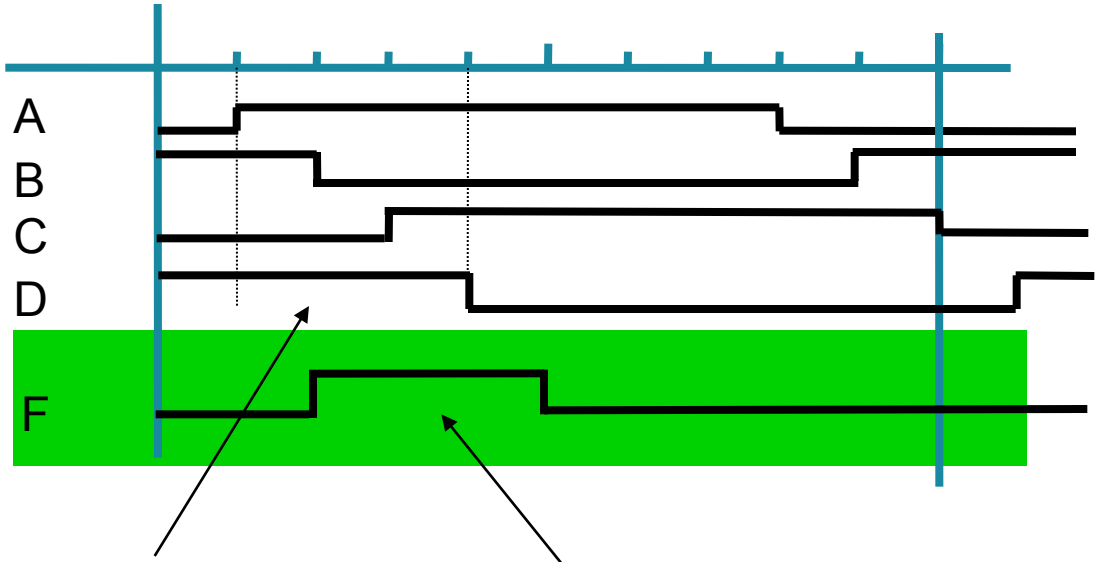


spike/glitch (1-0-1) op de uitgang !

Toepassing van spikes: Pulsvormer



statische theorie: F = A' • A = 0



D blijft 1 voor 3 gate delays nadat A van 0 naar 1 verandert

F is dus niet altijd 0 ==> puls

Samenvatting

Belangrijkste elementen:

- wat zijn two-level canonieke vormen?
- hoe pleegt men two-level vereenvoudiging?
- wat zijn multi-level netwerken en hoe te realiseren?
- wat is het tijdsgedrag van combinatoriek?
- hoe onstaan spikes/glitches?

Volgende keer:
Sequentiële netwerken en Finite State Machines

Woensdag: toets over hoorcollege 1 t/m 3

Huiswerkopgaven

Doel huiswerkopgaven:

- terugkoppeling individuele vaardigheden
- voorbereiding op toetsen / tentamens

Opmerking:

Er is minimaal $n = 1$ antwoord de juiste. Er kunnen echter *meerdere* antwoorden juist zijn.

Canonieke vorm

Vraag 1: Wat is de juiste canonieke vorm?

a. $F = \sum m(1,2,4,5,6)$

b. $F = \sum m(0,5,7)$

c. $F = \sum m(1,2,3,4,6)$

d. $F = \sum m(2,3,4,6)$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Vraag 2: Wat is de juiste canonieke vorm?

a. $F = \sum m(0,2,5,7)$

b. $F = \prod M(0,2,5,7)$

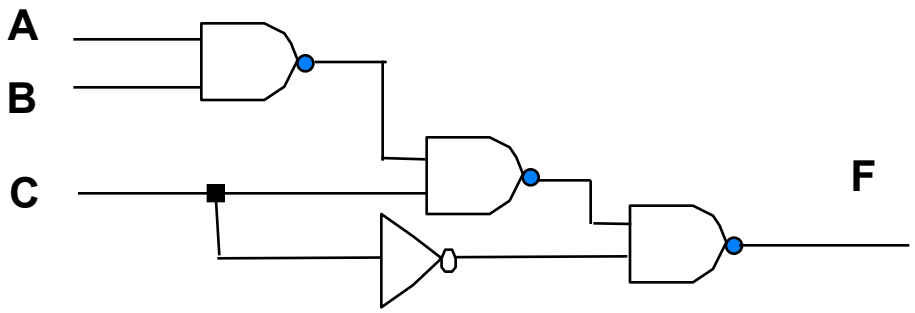
c. $F = \prod M(1,3,4,6)$

d. $F = \sum m(1,3,4,6)$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Canonieke vorm

Vraag 3: Wat is de juiste canonieke vorm?



A	B	C	F
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

a. $F = \sum m(0,2,3,5,6)$

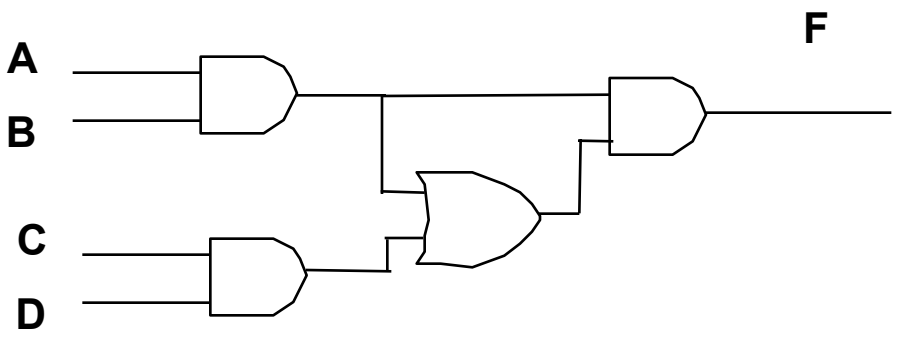
b. $F = \sum m(1,3,5,7)$

c. $F = \sum m(0,2,4)$

d. $F = \sum m(0,2,4,6)$

Canonieke vorm

Vraag 4: Wat is de juiste canonieke vorm?



A	B	C	D	F
0	0	0	0	?
0	0	0	1	?
.....				
1	1	1	1	?

a. F = $\sum$ m(0,2,4,6,8)

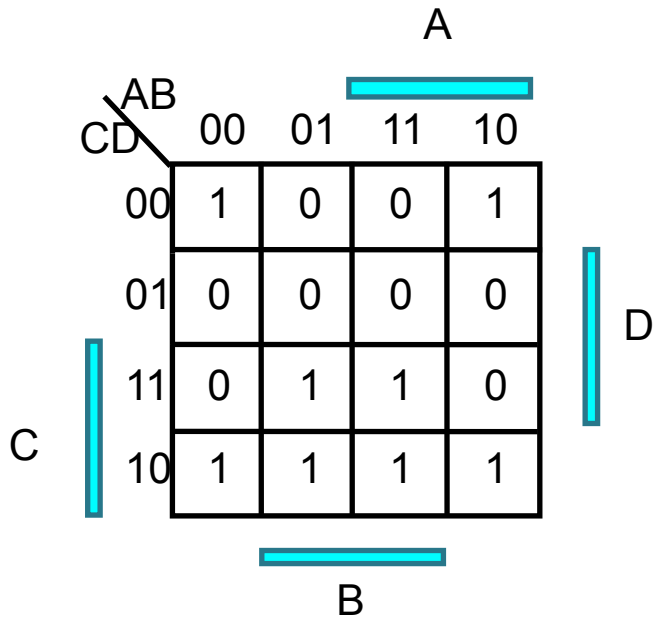
b. F = $\sum$ m(1,3,5,7,9)

c. F = $\sum$ m(12,13,14,15)

d. F = $\sum$ m(8,9,10,12)

Two-level netwerken

Vraag 5: Wat is de juiste vereenvoudiging?



a. F = BC + CD + B'C'D'

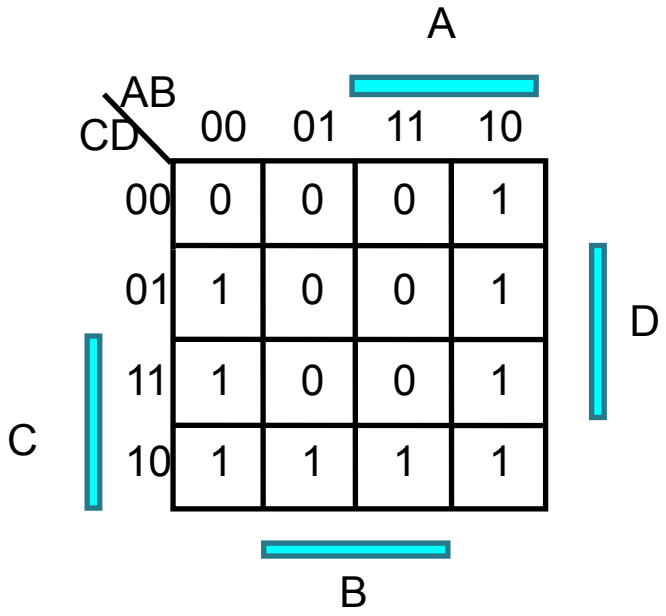
b. F = B'D' + CD'

c. F = B'D' + BC

d. F = BD + C'D + B'C'

Two-level netwerken

Vraag 6: Wat is de juiste vereenvoudiging?



a. F = A'B' + AB' + CD'

b. F = AB' + CD' + B'D

c. F = B'D' + BC + AB'

d. F = AB + CD' + BD

Multi-level netwerken

Vraag 7: Welke realisatie van Y is correct?

Y = ACE + ACF + ADE + ADF + BCE + BCF + BDE + BDF

a. Y = AB(C+D)(D+E)

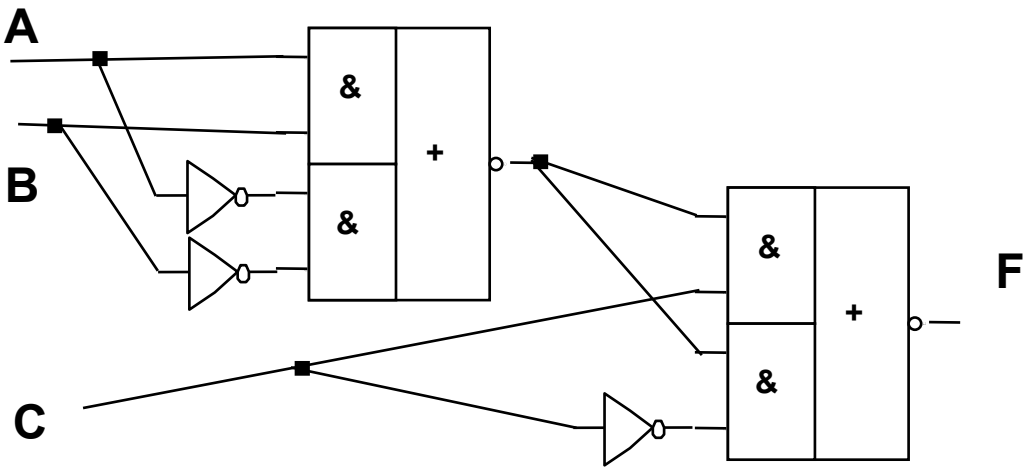
b. Y = (A+B)(C+D)(E+F)

c. Y = AB + CD + DE

d. Y = (A+B)(C+D+E)

Multilevel netwerken

Vraag 8: Welke expressie berekent dit netwerk?



a. F = A ⊕ B ⊕ C

b. F = C

c. F = A ⊕ B

d. F = (A ⊕ B)'

Multilevel netwerken

Vraag 9: Ontwerp een 2-bit opteller m.b.v. Half Adders volgens:

```
      a1  a0
      b1  b0
      -------- +
c2  c1  c0
```

Hoeveel Half Adders zijn benodigd?

a. 2

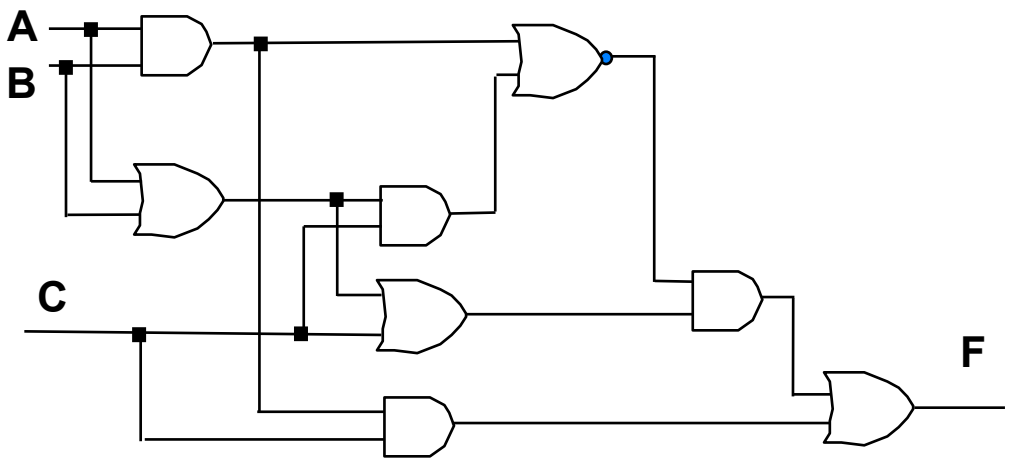
b. 3

c. 4

d. Geen van bovenstaande antwoorden.

Multilevel netwerken

Vraag 10: Onderstaand circuit moet worden gerealiseerd met uitsluitend NANDs met maximale fanin van 4. Hoeveel NANDs zijn er nodig?



a. 8

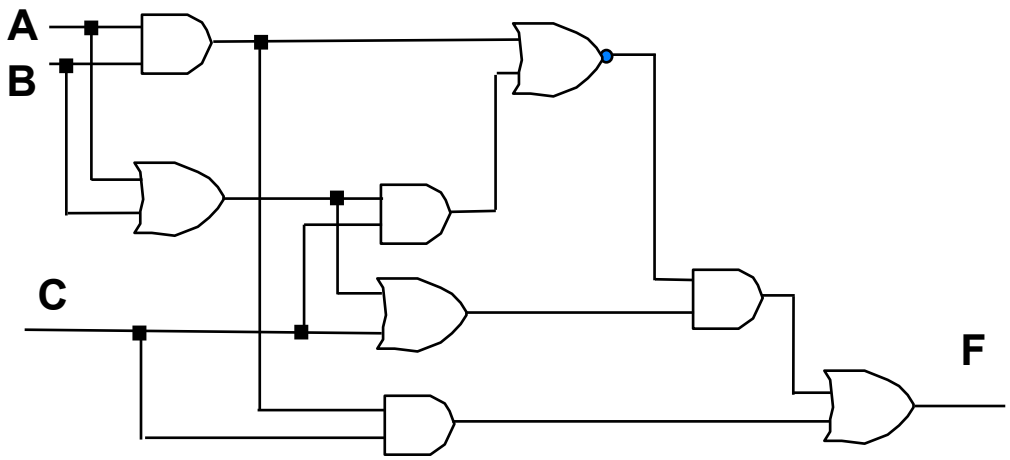
b. 5

c. 6

d. 4

Timing

Vraag 11: Gegeven onderstaand circuit. De propagatietijd van een AND en een OR poort zijn respectievelijk 5 ns en 7 ns. Alle ingangen zijn initieel 1. Op een zeker moment wordt A gelijk aan 0. Wat gebeurt er met de uitgang F?



a. Er gebeurt niets

b. F verandert van waarde na 24 ns

c. F verandert van waarde na 17 ns

d. F verandert van waarde na 19 ns