FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA

**Wouter Geraedts**

# Processen & Processoren

Radboud Universiteit Nijmegen

# Overzicht

Welkom op het $2^e$ werkcollege van Processen & Processoren!

- Uitwerkingen vorige opgavenserie
- Behandelen oefenopgaven

# Opgave 1

Breng de onderstaande formules in disjunctieve normaalvorm.

$$\neg(\neg a \wedge \neg b)$$

$$a \wedge \neg b \vee c$$

$$(b \vee \neg c) \wedge (a \vee b)$$

# Opgave 1 2

**Vraag:** $\neg(\neg a \wedge \neg b)$ in DNF

# Opgave 1 2

**Vraag:** $\neg(\neg a \wedge \neg b)$ in DNF

| a | b | $\neg(\neg a \wedge \neg b)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Opgave 1 2

**Vraag:** $\neg(\neg a \wedge \neg b)$ in DNF

| a | b | $\neg(\neg a \wedge \neg b)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Antwoord**: $(\neg a \wedge b) \vee (a \wedge \neg b) \vee (a \wedge b)$

# Opgave 1 3

**Vraag:** $a \wedge \neg b \vee c$ in DNF

# Opgave 1 3

**Vraag:** $a \wedge \neg b \vee c$ in DNF

| a | b | c | $(a \wedge \neg b \vee c)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

# Opgave 1 3

**Vraag:** $a \wedge \neg b \vee c$ in DNF

| a | b | c | $(a \wedge \neg b \vee c)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Antwoord**: $(\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge b \wedge c)$

# Opgave 1 4

**Vraag:** $(b \vee \neg c) \wedge (a \vee b)$ in DNF

# Opgave 1 4

**Vraag:** $(b \vee \neg c) \wedge (a \vee b)$ in DNF

| a | b | c | $(b \vee \neg c) \wedge (a \vee b)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

# Opgave 1 4

**Vraag:** $(b \vee \neg c) \wedge (a \vee b)$ in DNF

| a | b | c | $(b \vee \neg c) \wedge (a \vee b)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

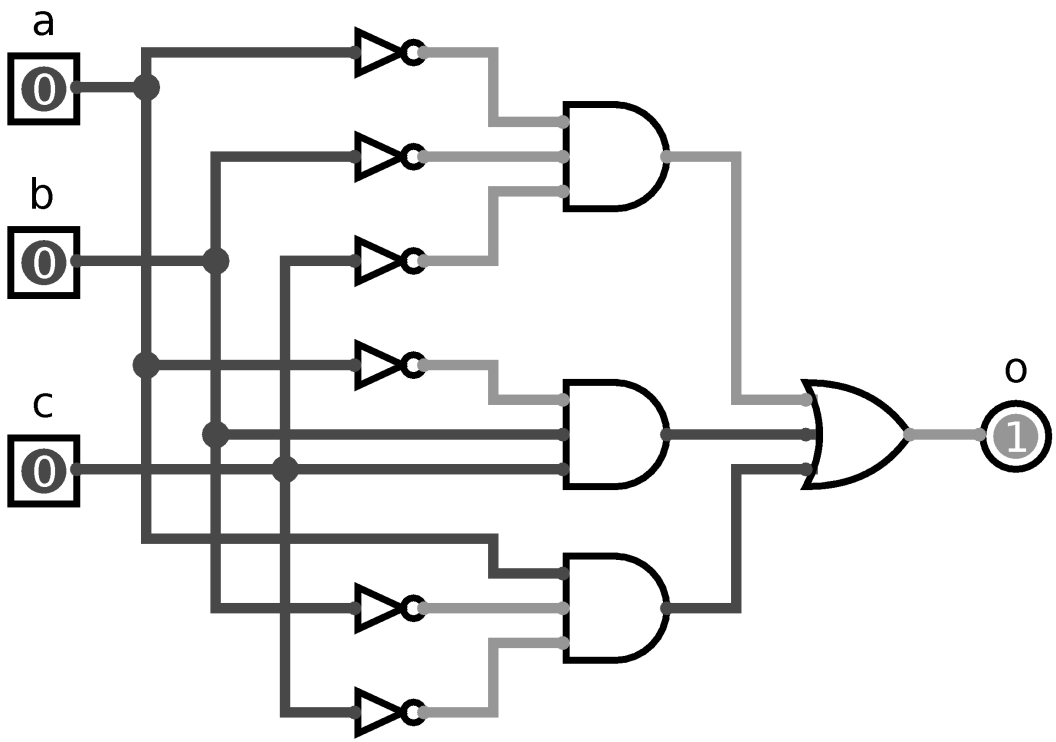
**Antwoord**: $(\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c)$

# Opgave 2a

Ontwerp een circuit dat de volgende formule berekent.

$$(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$$

# Opgave 2a 2




$$(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$$

# Opgave 2b

Probeer het circuit te vereenvoudigen, zodat je minder transistoren gebruikt.

(NOT = 1 transistor, NAND en NOR = 2 transistoren, AND en OR = 3 transistoren. Je mag ook gates met meer dan twee ingangen gebruiken, maar die hebben per extra ingang één extra transistor nodig, b.v. AND met 3 ingangen = 4 transistoren.)

# Opgave 2b [2]

- 6 NOT-gates: $6 \times 1 = 6$ transistoren
- 3 3-AND-gates: $4 \times 3 = 12$ transistoren
- 1 3-OR-gates: $4 \times 1 = 4$ transistoren

Wat in totaal neerkomt op $6 + 12 + 4 = 22$ transistoren.

# Opgave 2b [3]

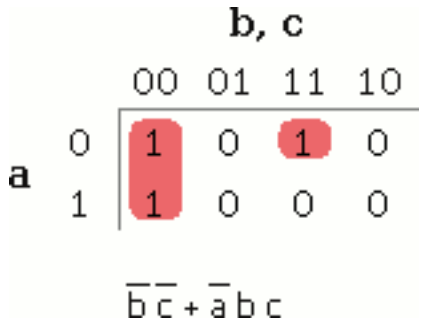
Als eerste moeten we de formule versimpelen:

$$(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$$

# Opgave 2b 3

Als eerste moeten we de formule versimpelen:

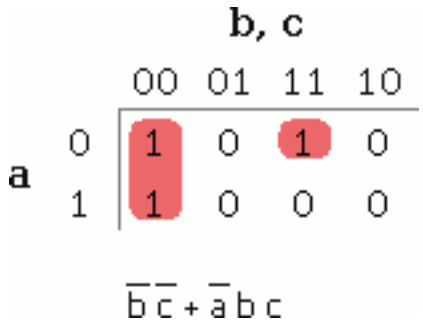
$$(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$$



# Opgave 2b 3

Als eerste moeten we de formule versimpelen:

$$(\neg a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$$




$$(\neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c)$$

# Opgave 2b 4

Deze functie dient dan nog goedkoper gemaakt te worden:

$$(\neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c)$$

# Opgave 2b [4]

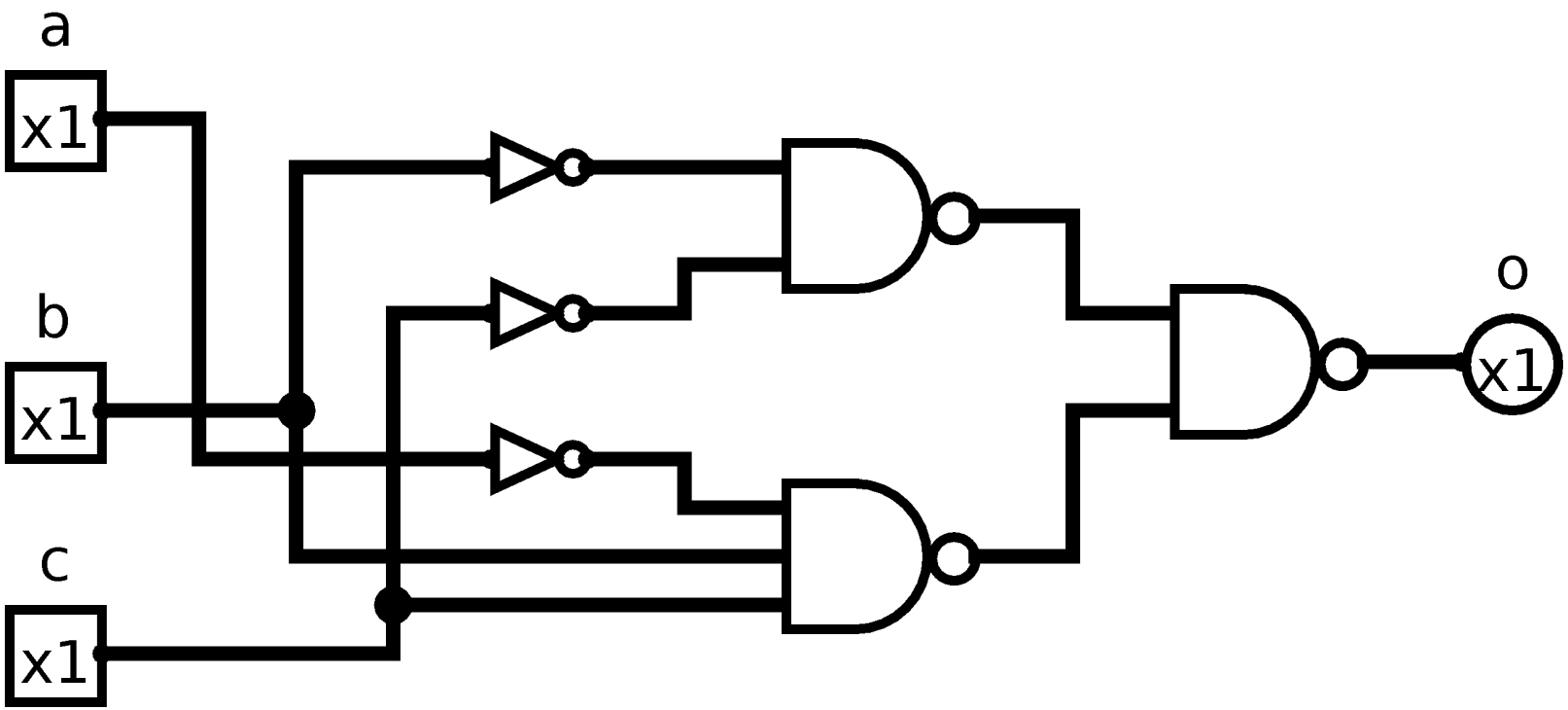
Deze functie dient dan nog goedkoper gemaakt te worden:

$$(\neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c)$$

Hier passen we DeMorgan voor toe:

$$\neg(\neg(\neg b \wedge \neg c) \wedge \neg(\neg a \wedge b \wedge c))$$

# Opgave 2b 5




$$\neg(\neg(\neg a \wedge \neg b \wedge c) \wedge \neg(a \wedge \neg c))$$

# Opgave 2b 6

- 3 NOT-gates: $3 \times 1 = 3$ transistoren
- 2 2-NAND-gates: $2 \times 2 = 4$ transistoren
- 1 3-NAND-gate: $1 \times (2 + 1) = 3$ transistoren

In totaal zijn dit $3 + 4 + 3 = 10$ transistoren.

# Opgave 3

Een machientje heeft drie knoppen en een LCD-paneeltje.

Dit LCD-paneeltje laat zien hoeveel van de drie knoppen indrukt zijn. Het paneeltje kan dus de getallen 0 tot en met 3 weergeven, welke in twee bits passen bij een unsigned getallenrepresentatie.

**Vraag:** ontwerp een circuit voor dit machientje.

# Opgave 3 2

Implementeer de “tel”-functie.

# Opgave 3 2

Implementeer de “tel”-functie.

| a | b | c | $g1$ | $g0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

# Opgave 3 3

Formules?

# Opgave 3 3

Formules?

Uitvoerlijn $g1$ komt neer op de "meerderheids"-functie:

$$(b \wedge c) \vee (a \wedge c) \vee (a \wedge b)$$

# Opgave 3 4

De formule voor uitvoerlijn $g0$ is niet echt makkelijk te noteren:

# Opgave 3 4

De formule voor uitvoerlijn $g0$ is niet echt makkelijk te noteren:

Voor één enkele knop:

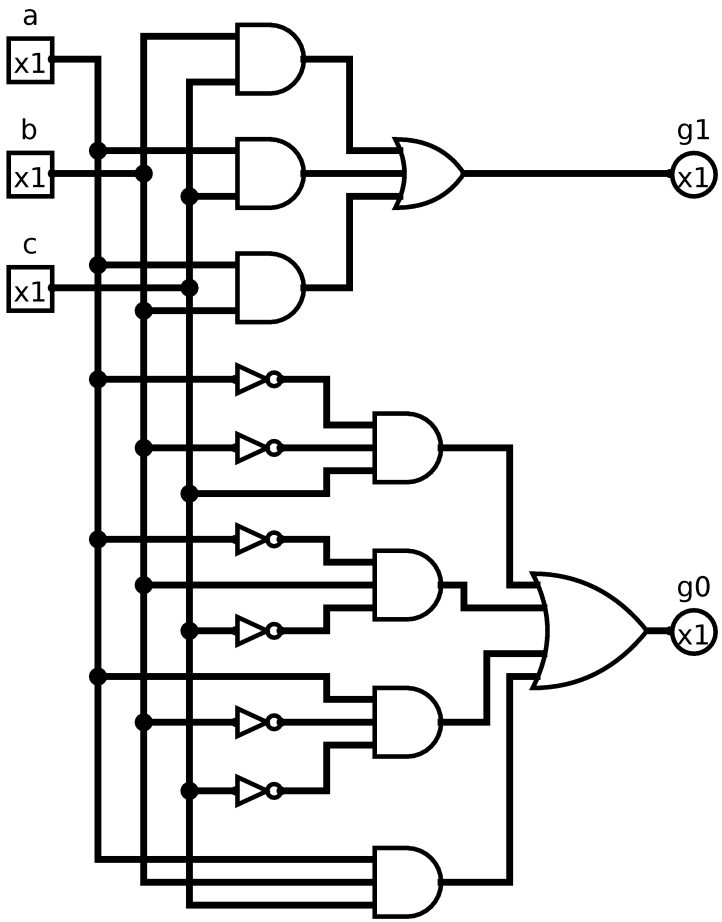
$$\text{xor}_3(a, b, c) =$$

$$(a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c)$$

Voor alle knoppen:

$$a \wedge b \wedge c$$

# Opgave 3 5



# Opgave 4

Ontwerp een circuit dat twee getallen van twee bits bij elkaar optelt, $(a_1 a_0) + (b_1 b_0)$. Je mag geen gebruik maken van addeerwerken.

# Opgave 4 2

De brute-force methode: waarheidstabel.

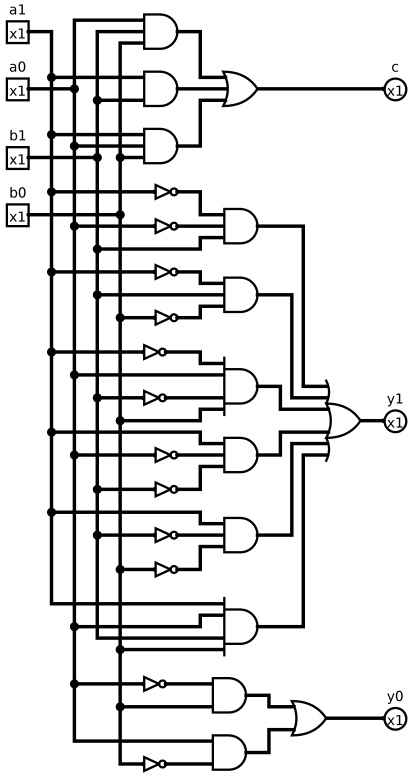
# Opgave 4 2

De brute-force methode: waarheidstabel.

| $a_1$ | $a_0$ | $b_1$ | $b_0$ | $C$ | $y_1$ | $y_0$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

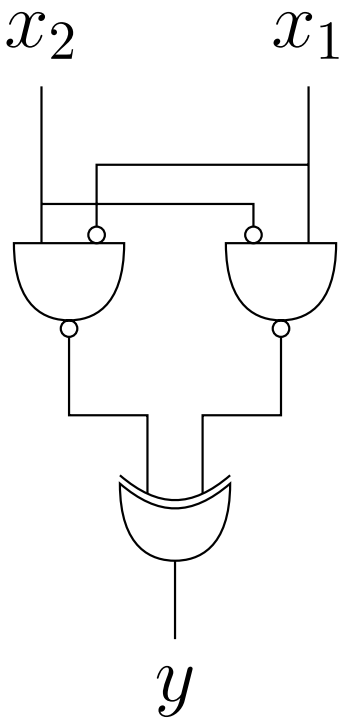
# Opgave 4 3

Implementeer dit vervolgens in een PLA; of maak met de hand zoiets:



# Opgave 5

Welke functie berekent het volgende circuit?



# Opgave 5 2

| a | b | $f$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Opgave 5 [2]

| a | b | $f$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Antwoord**: ofwel de XOR

# Oefenopgaven

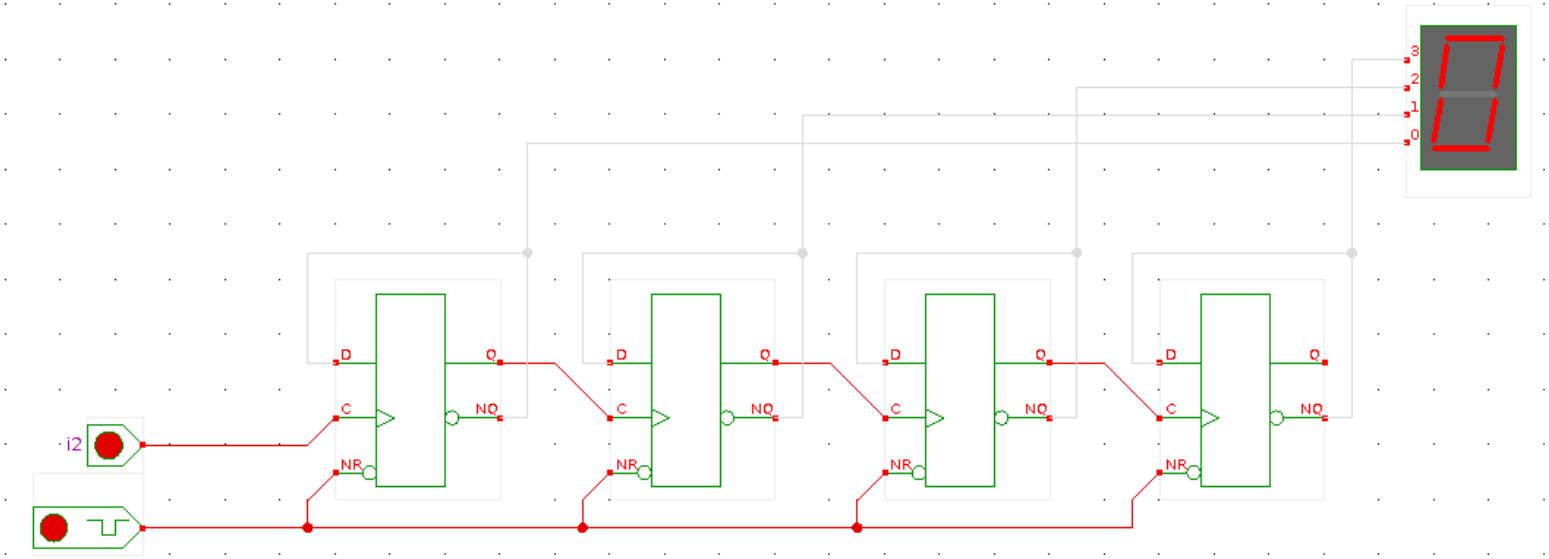
Op naar de oefenopgaven!

# Opgave 1

Teken een schakeldiagram voor een timer.

De timer heeft vier uitvoerlijnen die een binair getal van 0 tot 15 kunnen produceren. Hij heeft een invoerlijn *reset*, die de uitvoer op 0 zet. De timer telt het aantal klokpulsen (die via de tweede invoerlijn *clock* binnenkomen) sinds hij is gereset.

# Opgave 1 2



# Opgave 2

Implementeer de „meerderheid”-functie in een PLA.

# Opgave 3

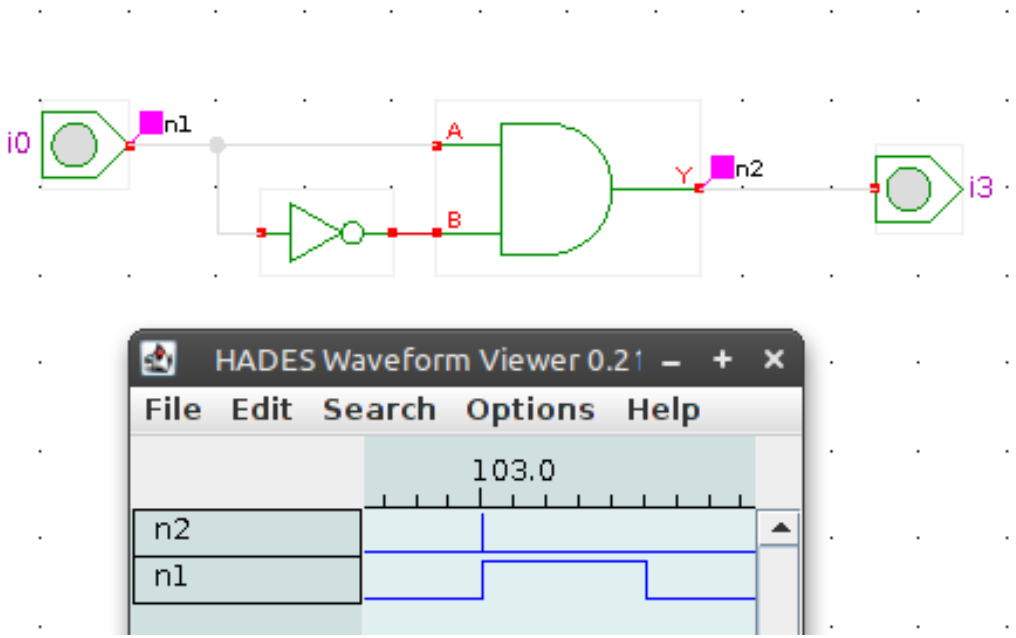
Teken twee circuits voor de herkenning van de rising edge en de falling edge.

# Opgave 3 2

Teken een circuits voor de herkenning van de rising edge.

# Opgave 3 2

Teken een circuits voor de herkenning van de rising edge.

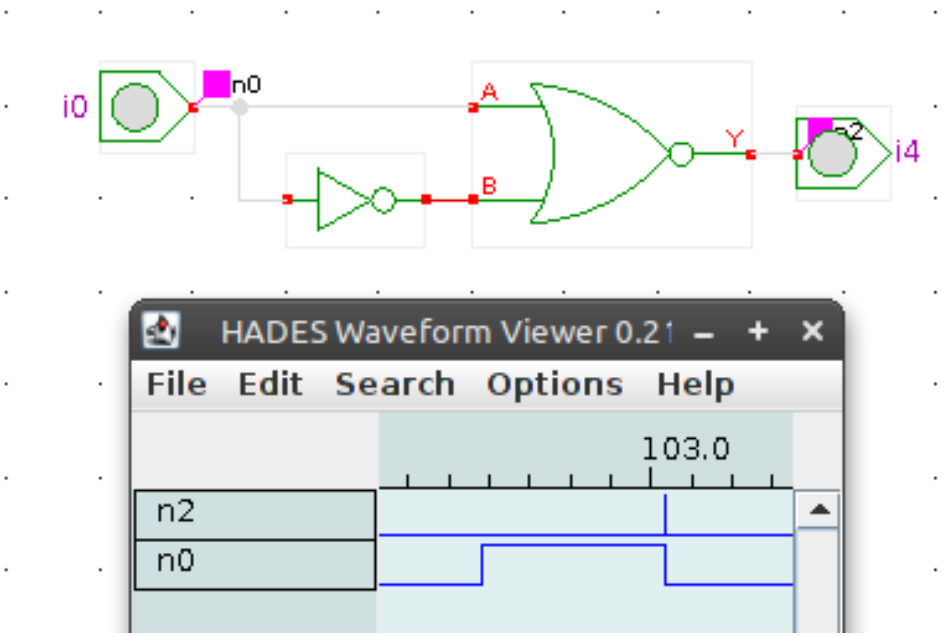


# Opgave 3 3

Teken een circuits voor de herkenning van de falling edge.

# Opgave 3 3

Teken een circuits voor de herkenning van de falling edge.



# Opgave 4

We hebben in het college gezien dat een moderne chip meerdere instructies tegelijk behandelt (in een pipeline).

Een van de fasen in de pipeline is het lezen van de instructie uit het geheugen (fetch).

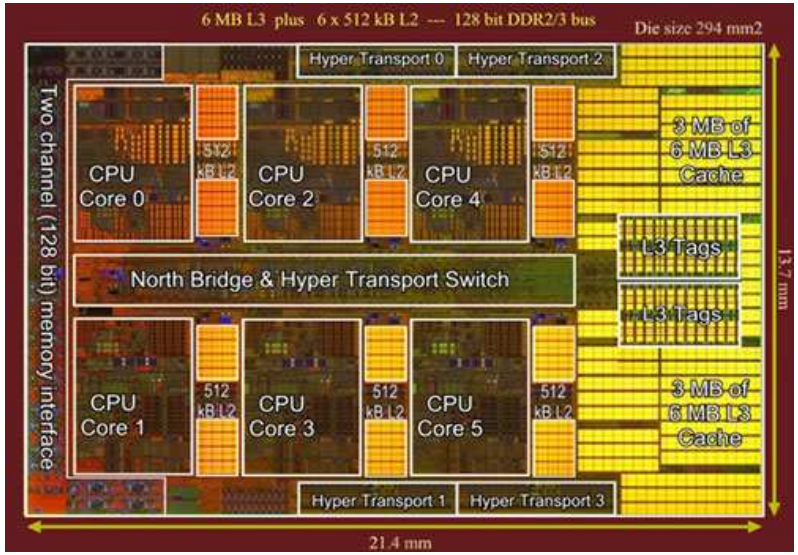
Wat gebeurt er als een instructie zelf ook gegevens uit het geheugen wil lezen?

# Opgave 4 2

Electrisch signaal plant zich langzaam voort!
Maar, CPU is klein, en dus snel.

# Opgave 4 2

Electrisch signaal plant zich langzaam voort!
Maar, CPU is klein, en dus snel.

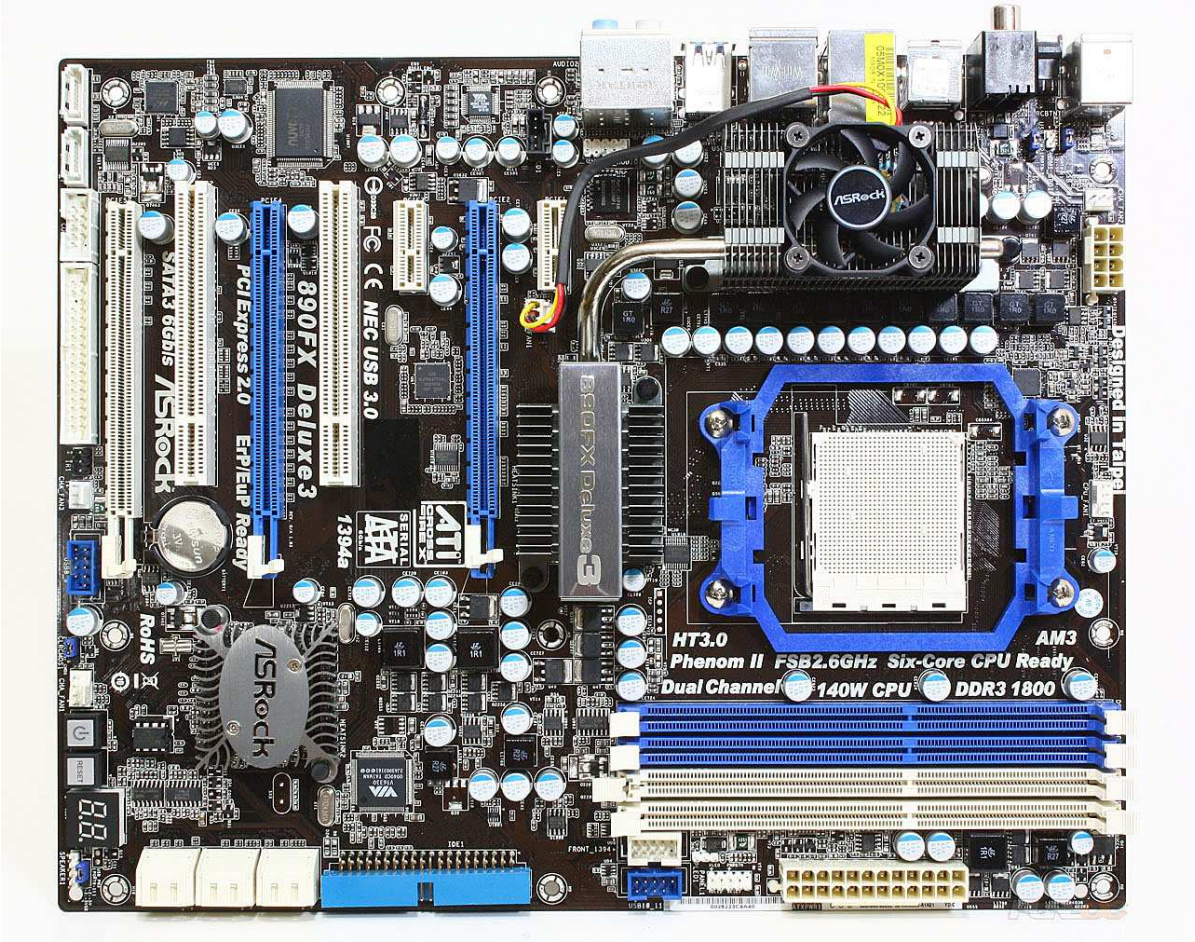



# Opgave 4 3

Het geheugen zit echter een heel eind weg.

# Opgave 4 3

Het geheugen zit echter een heel eind weg.



# Opgave 4 4

De *fetch*-instructie kost dus veel tijd. (= veel CPU-cycles)

# Opgave 4 4

De *fetch*-instructie kost dus veel tijd. (= veel CPU-cycles)

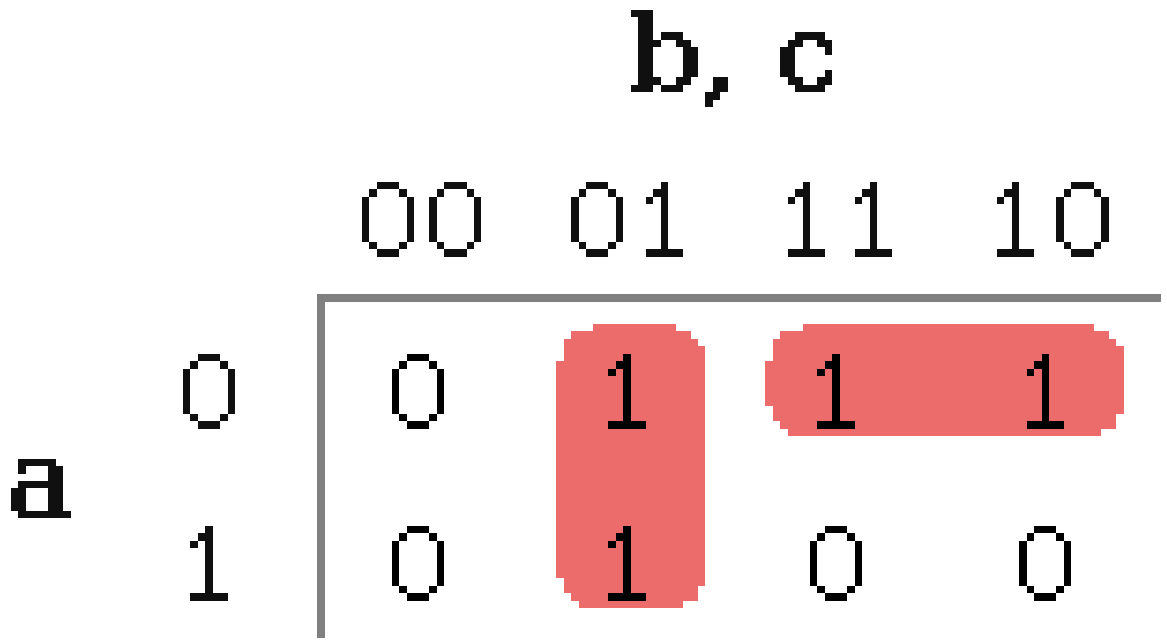
De gehele pipeline moet dus wachten, totdat de gegevens opgehaald zijn uit het geheugen en in de CPU-registers zijn gezet.

# Opgave 5

Teken een karnaugh-diagram voor de volgende waarheids-tabel, en geef a.d.v. de diagram de minimale expressie voor $f$:

| a | b | c | $f(a,b,c)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

# Opgave 5 2



# Opgave 5 2

| a \ b, c | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |

$$(\neg b \wedge c) \vee (\neg a \wedge b)$$

# Einde

Fin