Digitale voorstelling van informatie

Samenvatting

Gilles Callebaut

Over Bits en Bytes

Binaire woorden - bytes

binair woord is voorgesteld als $b_{n-1}\ b_{n-2} \cdots b_1\ b_0$

voorstellen M symbolen → n bits nodig

$$2^{n-1} < M \leq 2^n$$

$2^{10} = 1024$ of 1kbit

$2^{20} = 1\,041\,576$ of 1Mbit

Voorstelling van getallen

Zuivere binaire voorstelling

Binaire codering

binair getal $b_n \dots b_0 , b_{-1} \dots b_{-m}$

decimale waarde $b_n \times 2^n + \dots + b_0 \times 2^0 + b_{-1} \times 2^{-1} + \dots + b_{-m} \times 2^{-m}$

Most Significant Bit

Least Significant Bit

Opmerking

verplaatsing binaire komma $\xleftarrow{L}$ delen door 2

$\xrightarrow{R}$ vermenigvuldigen met 2

---

schuifbewerking $\xleftarrow{L}$ vermenigvuldiging met 2

0 rechts toevoegen

probleem als MSB = 1

↳ overstappen op meer bits anders overflow

$\xrightarrow{R}$ delen door 2

rechtse bit verloren

↳ rest niet behouden

0 links toev.

Binair → Decimaal

1	1	0	1	0	1	1

1·2	3·2	6·2	26	52	106
+	+	+		+	+
1	0	1		1	1
3	6	13	26	53	**107**

0,	0	0	1

$\frac{1}{2}$ $\frac{1}{2}$ $1 \cdot \frac{1}{2}$

$+ 0$ $+ 0$

$\frac{1}{8}$ $\frac{1}{4}$ $\frac{1}{2}$

Decimaal → Binair

als > 1 → − 1 → 1 binair

0 1 3 7 · **14** (/2)

1	1	1	0

rest

0,2 $\xrightarrow{\times 2}$ 0,4 → 0,8 $\xrightarrow{\times 2}$ 0,6 $\xrightarrow{\times 2}$ 0,2 →

0,	0	0	1	1	0	0	1	1

Octale en hexadecimale voorstelling

octale code: 3bits samen nemen → van 0 → 7

hexadecimale code: 4 bits samen nemen → van 0 → 9, A → F

vb.

octaal	4			7			2			3		
binair	1	0	0	1	1	1	0	1	0	0	1	1
hexa-dec.	9				D				3			

Voorstelling v. negatieve getallen

Sign-plus-value voorstelling

MSB: tekenbit → 0 +
→ 1 -

nadeel: getal 0 → +0 en → -0 kan zijn.

Two's Complement voorstelling

voordeel: eenvoud vh gebruik bij rekenkundige bewerkingen

byte: pos. getallen: 0 → 127 met $b_7 = 0$
neg. getallen: -128 → -1 met $b_7 = 1$

vb. $+17_{10}$ 0001 0001
-17_{10} 1110 1111

one's comple.

$$\begin{array}{r} 1110\ 1110 \\ +\ 0000\ 0001 \\ \hline 1110\ 1111 \end{array}$$

two's comple.

Floating-point voorstelling

principe

$x = \underbrace{M}_{\text{mantisse}} \times 2^{E}$ → exponent

· genormaliseerde voorstellingswijze : E gekozen zodat de M v/d vorm $0,\underset{\uparrow}{1}\ldots$ is.

IEEE Standaard

→ E : 1 → 254 : 8 bits
M : 23 bits
S : 1 bit

} 32 bits = 4 bytes

voorstelling: $(-1)^S \cdot 2^{(E-127)} \cdot (\underset{\uparrow}{1},M)$

hidden bit

vb. $(-144,5)_d$ ⟶ 144 → 0 1 2 4 9 18 36 72 144
1 0 0 1 0 0 0 0

0,5 → $0,5 \overset{2}{\frown} 1$
0, 1

$(144,5)_d \to (1001\ 0000,1)_b = (1,0010\ 0001 \cdot 2^7)_b$

$7 = E - 127$

→ $E = (134)_d$ → $E = (1000\ 0110)_b$

1 100 0011 0001 0000 1000 0000 0000 0000

Sign bit E M opvullen → 32 bits (totaal)

Enkele Precisie (32 bits)

Exponent E	Mantisse M	waarde	
255	$\neq 0$	gn betekenis	
255	0	$(-1)^S . \infty$	
$0 < E < 255$	M	$(-1)^S . 2^{E-127} . (1,M)$	} Genormaliseerd
0	$\neq 0$	$(-1)^S 2^{E-126} (0,M)$	} Gedenormaliseerd
0	0	$(-1)^S 0$	

Genormaliseerd

Grootste getal:

$M = 1,1 \ldots 1$ & $E = 254$

$$\begin{array}{rl} & 1,1\ldots1 \\ + & 2^{-23} = 0,0\ldots01 \\ \hline & 2 = 10,0\ldots00 \end{array}$$

$\rightarrow (\pm)(2-2^{-23}) . 2^{127}$

Kleinste getal:

$M = 1,0 \ldots 0$ & $E = 1$

$\rightarrow (\pm) 2^{-126}$

Gedenormaliseerd

Grootste getal:

$M = 0,1 \ldots 1$ (& $E = 0$ altijd)

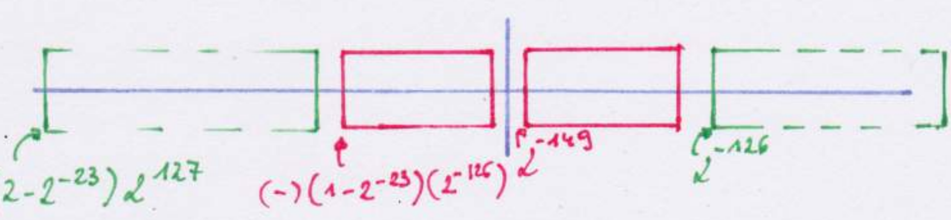
$$\begin{array}{rl} & 0,1\ldots1 \\ + & 2^{-23} = 0,0\ldots01 \\ \hline & 1 = 1,0\ldots00 \end{array}$$

$\rightarrow (\pm)(1-2^{-23})(2^{-126})$

Kleinste getal:

$M = 0,0 \ldots 01$ (& $E = 0$)

$\rightarrow (\pm)(2^{-23})(2^{-126})$



Andere codering

Karaktercodes

ASCII - code → 7 bits + 1 bit

7 bits ↓ karakter

1 bit ↓ pariteitsbit ter controle

- even pariteit: even # 1'jes
- oneven pariteit: oneven # 1'jes

Instructiecodes

processors → 8 bit werken
=
256 ≠ instructies

Opmerking

verschillende interpretaties aan binaire code.

Opslag Data

- sequentieel toegankelijk
  ↓
  massageheugens
  datatype
  ≠ individueel geadresseerd gelezen geschreven
- willekeurig toegankelijk
  op ieder ogenblik toeg.
  ↳ plaats cel = adres
  - ROM: Read-only-mem.
  - RAM: Random-acces-mem.

Aritmetische en Logische functies

Samenvatting

Gilles Callebaut

Rekenkundige bewerkingen

Optelling van binaire getallen

carry bit → 1 1

vb.

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ 0\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1 \end{array}$$

De aftrekking

1 1 1 ← borrow bit

vb.

$$\begin{array}{r} 1\ \cancel{0}\ \cancel{0}\ 1 \\ \hline 1\ 0\ 1\ 0 \\ 0\ 1\ 1\ 1 \\ \hline 0\ 0\ 1\ 1 \end{array}$$

Bewerkingen in two's complement voorstelling

1. pos. getallen → binair equivalent met MSB = 0
2. neg. getallen → two's complement met MSB = 1

⇒ aftrekken = optelling A + (−B)

uitkomst: pos → binaire equi. ; neg → 2's compl.

probleem: Overflow → buiten −128 tot +127

vb.

$$\begin{array}{rr} 1010\ 0001 & -95 \\ 1101\ 1000 & -40 \\ \hline \cancel{1}\ 0111\ 1001 & +121\,! \end{array}$$

De vermenigvuldiging

```
   0100
x  0011
-------
   0100
  0100
-------
  1100
```

De deling

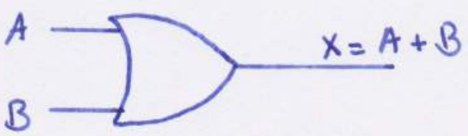
```
11001001 | 1001
-1001    |------
-----    | 10110
  111
- 0000
-----
  1110
- 1001
-----
   10101
 - 1001
 ------
      11
```

Eenvoudige logische functies

De fundamentele logische functies

NOT

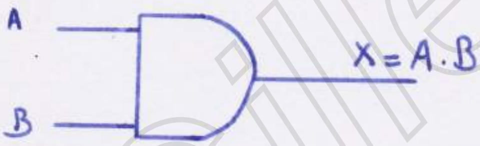
OR



← inclusieve OR

want $A = B = 1 \Rightarrow X = 1$

AND



Eigenschappen

$A \times B = B \times A$

$A \times (B \times C) = (A \times B) \times C$

$A \times (B + C) = (A \times B) + (A \times C)$

$A + B = B + A$

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

$A \times 0 = 0 \qquad A + 0 = A$

$A \times 1 = A \qquad A + 1 = 1$

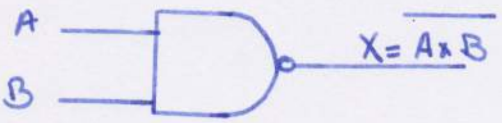
$A \times A = A \qquad A + A = A$

$A \times \overline{A} = 0 \qquad A + \overline{A} = 1$

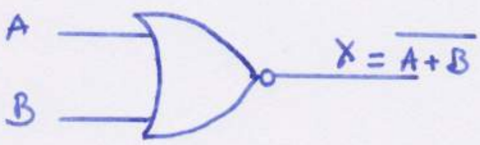
$$\left.\begin{array}{l} \overline{A \times B} = \overline{A} + \overline{B} \\ \overline{A + B} = \overline{A} \times \overline{B} \end{array}\right\} \text{De Morgan}$$

Afgeleide logische functies

NAND



NOR



XOR

X = 1 als # enen ingangen oneven is.

Eigenschappen van XOR:

$A \oplus B = A.\overline{B} + \overline{A}.B = (A+B)(\overline{A}+\overline{B})$

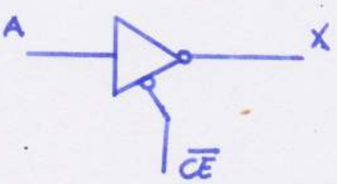
$A \oplus 0 = A$

$A \oplus 1 = \overline{A}$

als $A \oplus B = 1$ dan $\left.\begin{array}{l} A \oplus 1 = B \\ B \oplus 1 = A \end{array}\right\}$ A & B zijn elkaars complement

Bouwstenen voor combinatorische circuits

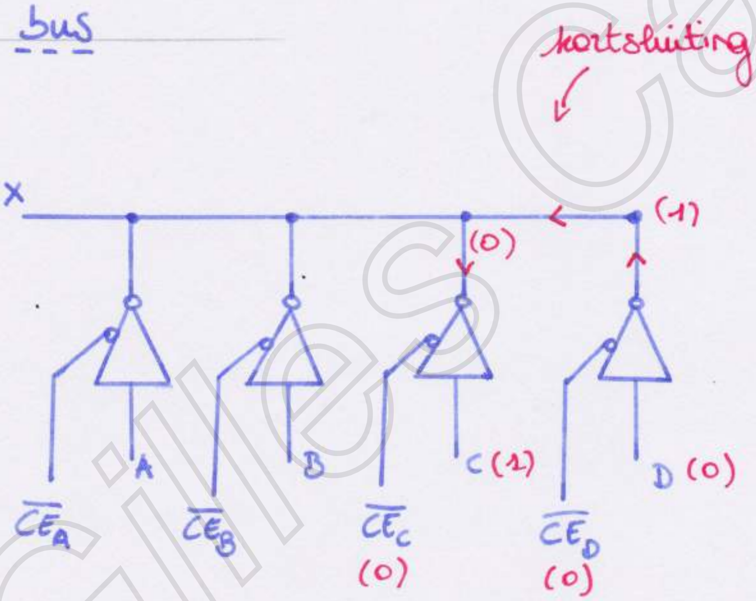
Tri - State Poorten



$\equiv$

A	$\overline{CE}$	X
0	0	1 ($\overline{A}$)
1	0	0 ($\overline{A}$)
x	1	hoge imped. zweven

bus



maar 1 actief !

↓

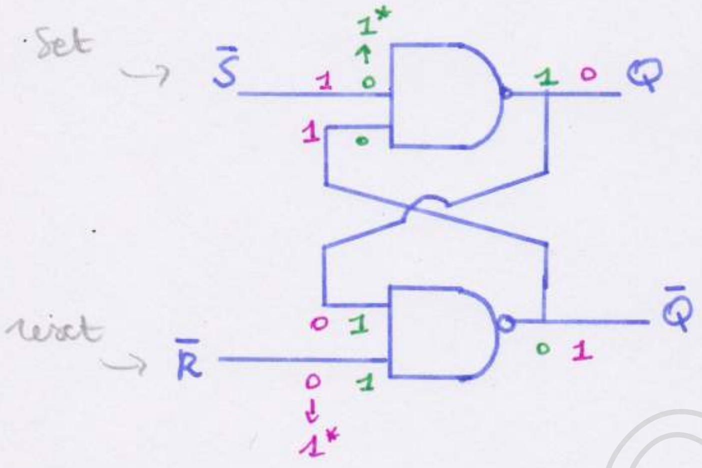
anders lijn onzeker

kortsluiting mogelijk

Bouwstenen voor sequentiële schakelingen

(uitvoerig besproken volgend hoofdstuk)

De SR-latch



$\bar{S} = 0$ & $\bar{R} = 1$ → $Q = 1$ & $\bar{Q} = 0$

als $\bar{S} = 1^*$ → toestand onthouden

$\bar{S} = 1$ & $\bar{R} = 0$ → $Q = 0$ & $\bar{Q} = 1$

als $\bar{R} = 1^*$ → toestand onthouden

$\bar{S} = 0$ & $\bar{R} = 0$ → $Q = 1$ & $\bar{Q} = 1$

als $\bar{S} = 1$ & $\bar{R} = 1$ → race-condition

De flip-flop

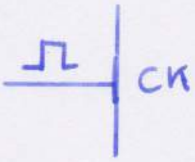
transitietabel / statentabel $\quad Q_n \quad Q_{n+1}$

master-slave flip-flops

ingangsklemmen → nieuwe toestand

oude toestand nog even vasthouden

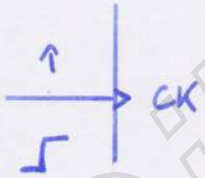
Triggering van Flip-Flops



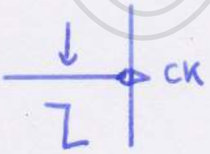
triggering vanaf stijgende flank (0 → 1)
terwijl kloksignaal '1' is
stopt dalende flank (1 → 0)

omgekeerde van hier boven

flank - getriggerde flip - flops



triggering bij stijgende flank (0 → 1)
zeer snelle toestandsverandering



omgekeerde van hier boven.

Implementaties van logische functies

Standaard vormen

standaardproduct (minterm)

and-functie

waarheidstabel → 1

enkel één

$F = \Pi(\ldots)$

standaardsom (maxterm)

or-functie

waarheidstabel → 0

enkel één

$F = \Sigma(\ldots)$

voorbeeld:

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

...

Sum-Of-Products

SOP

P_1	P_2	...
0	0	
1	0	
0	0	
0	1	

$\bar A \bar B C$ $\quad$ $\bar A B C$ ...

$P_1 + P_2 + \ldots$

$F = \bar A \bar B C + \bar A B C + \ldots$

$F = \sum(1, 3, \ldots)$

via De Morgan

Products-of-sums

POS

S_1	S_2	...
0	1	
1	1	
1	0	
1	1	

$A + B + C$ $\quad$ $A + \bar B + C$ ...

$S_1 \times S_2 \times \ldots$

$F = (A+B+C) \times (A+\bar B+C) \times \ldots$

$F = \Pi(0, 2, \ldots)$

Combinatorische logische functies

Gilles Callebaut

Betekenis van "dont care" - functiewaarden

Ontstaan:

- functiewaarde gn enkel belang bij de betreffende combinatie vd onafh veranderlijken.

- betreffelijke combinatie v onafh. verand. kan nt voorkomen.

  ↳ BCD-getallen $\left.\begin{matrix}1010\\1111\end{matrix}\right\}$ gn betekenis

  want naar voorstellen 1 t.e.m. 9

Gebruik van de Karnaughkaart

Vaarstelling van functies

c \ ab	00	01	11	10
0				
1				

nodig voor het opl. van de Karnaughkaarten

slechts één veranderlijke veranderen → en ↓

Vereenvoudiging v. functies m.b.v. Karnaughkaarten

- identificeer alle vakjes → 1 die nt kunnen gecombineerd w̄
- identificeer alle vakjes → 1 die op één enkele wijze kunnen gecombineerd w̄. . .
  - met 2 buren
  - met 4 buren
  - met 8 buren
- gebruik "dont care"-waarden in uw voordeel

⟹ SOP- vorm opschrijven (kijken naar 1'en)

POS- vorm opschrijven (kijken naar 0'en)

(let op AB 00 ↳ (A+B) niet $\bar{A}+\bar{B}$ zoals bij SOP

Methode van Quine - McClusky

- Opsporen priemimplicanten.
- Bepalen essentiële priemimplicanten

Voorbeeld: $f = \sum m(0,1,2,5,6,7,8,9,10,14)$

zie dropbox

Implementatie van logische functies

XOR - poorten

bijna dambord patroon → som XOR + andere logische functie

te veel enen:

CD \ AB	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	**1**	1	0	1
10	1	**1**	1	0

$\xrightarrow{\text{XOR} +}$

CD \ AB	00	01	11	10
00	0	X	0	X
01	X	0	X	0
11	1	X	0	X
10	X	1	X	0

$$F = A \oplus B \oplus C \oplus D + \bar{A} . C$$

te veel nullen:

CD \ AB	00	01	11	10
00	0	0	1	**0**
01	0	0	**0**	1
11	1	**0**	0	0
10	1	1	0	0

$\xrightarrow{\text{XOR} \times}$

CD \ AB				
	X	X	1	0
	X	X	0	1
	1	0	X	X
	1	1	X	X

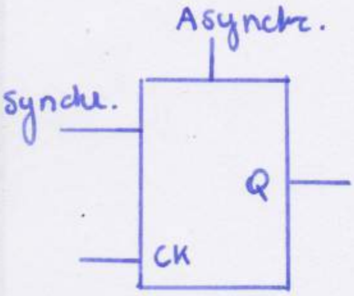
$$F = (A \oplus C) \times (C\bar{D} + B \oplus D)$$

Bouwstenen voor

Sequentiële netwerken

Gilles Callebaut

Flip - Flops

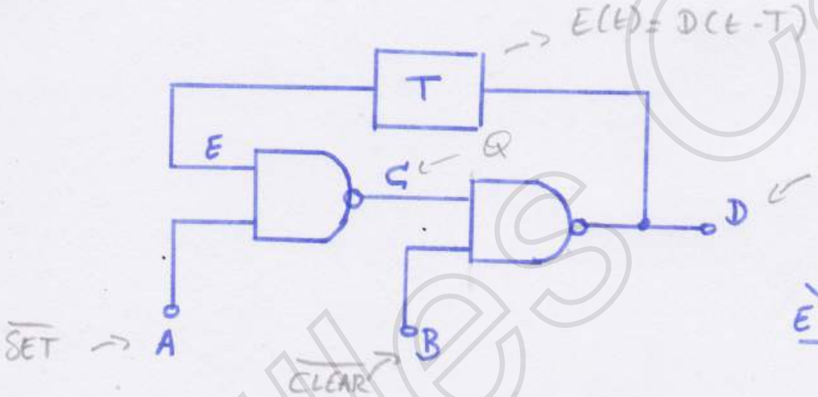


synchrone ingangen → hoe flip-flop reageren op CK

klokingang → flip-flop bevel aan te passen aan de synchrone ingangen

asynchrone ingangen → flip-flop in toestand dwingen onafh. v.d. CK

De basis - latch schakeling



$D = \overline{B \cdot C} = \overline{B} + \overline{C} = \overline{B} + EA$

E \ AB	00	01	11	10
0	1*	0	0	1*
1	1	0*	1	1

* instabiele toestand vermits $E \neq D$

Stel A & B = 1 en D = 0 → D = 1 maken

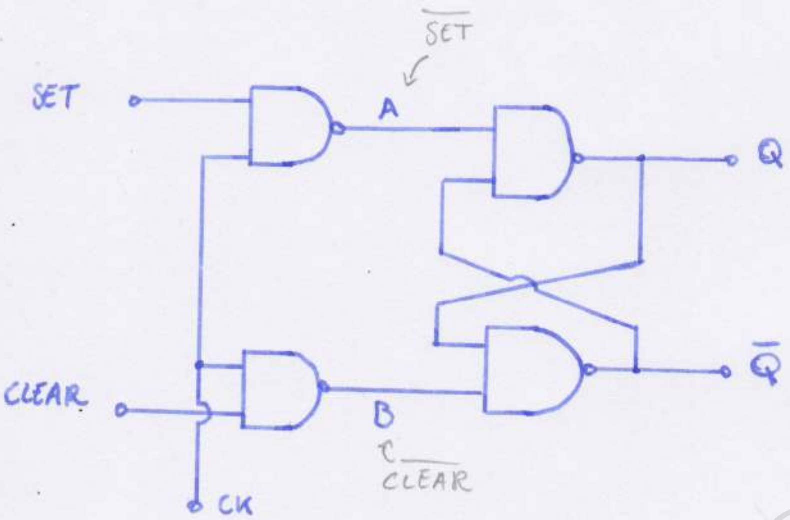
B = 0 zodat E = 1 → E = D

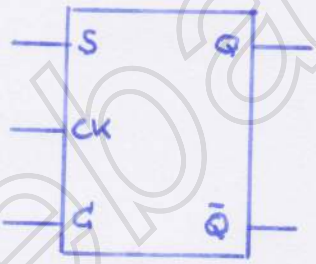
B = 1 zodat D = 1 bij AB = 11

Stel A & B = 0 ⟶ A & B = 1 hangt het eindresultaat af van wie eerst '1' is

$\overline{SET}$	$\overline{CLEAR}$	Q	$\overline{Q}$	
0	0	1	1	niet toegelaten ingangscombinatie
0	1	1	0	latch is ge-set
1	0	0	1	latch is ge-cleared
1	1	NC	NC	geheugentoestand (No Change)

De flip-flop met klokingang



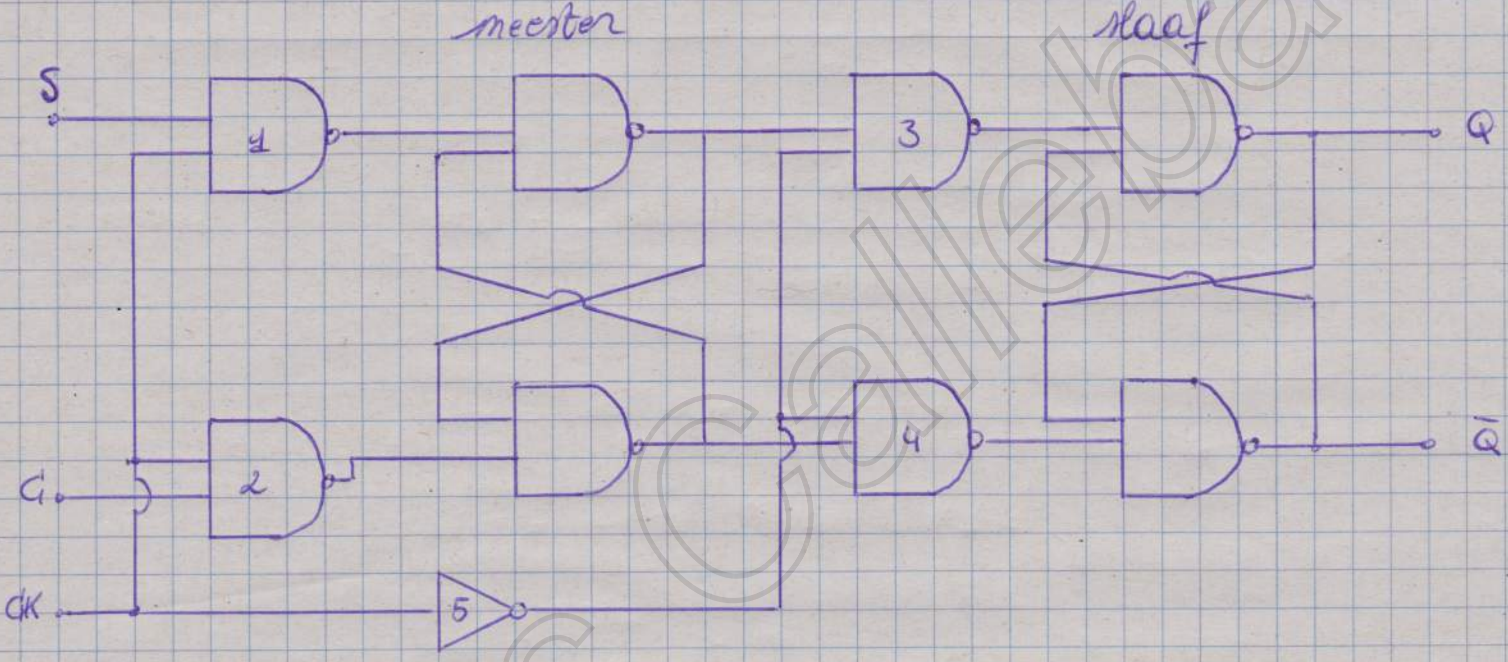


bij CK = '1'

SET	CLEAR	Q_{n+1}	
0	0	Q_n	NG
0	1	0	ge-cleared
1	0	1	ge-set
1	1	?	niet toegelaten

De Meester-slaaf flip-flop (MS-flipflop)

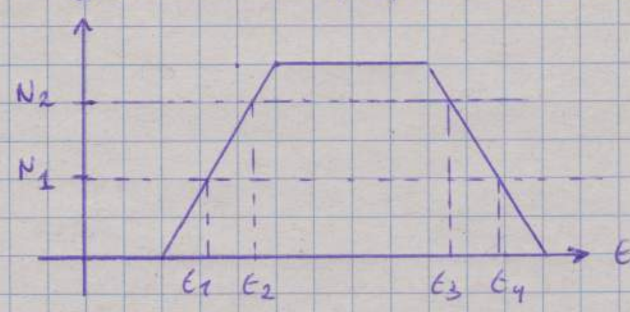
nieuwe info — oude toestand even onthouden



- CK laag : 1, 2, 5 → hoog — meester los van ingang, slaaf verbonden met meester.

- t_1 : 5 : '0' voor 3,4
  → slaaf losgekoppeld v/d meester
  oude toestanden behouden.

N_1 drempelsp. 5
N_2 drempelsp. 1, 3, 4, 2



- t_2 : meester verbonden aan S & C
  slaaf nog altijd losgekoppeld.

  meester nieuwe staat Q_{n+1}

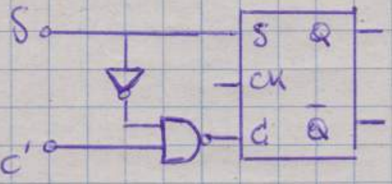
- t_3 : '0' voor 1 & 2 → meester losgekoppeld v/d ingangen.

- t_4 : slaaf verbonden met meester
  → nieuwe toestand overnemen
  →

na CK puls

S	C	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	??

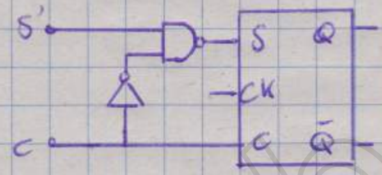
Prioritaire-set Flip-Flop



zodat

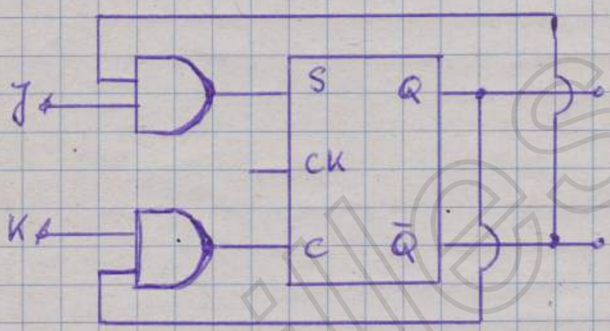
S	C'	Q_{n+1}
1	1	1

Prioritaire-Clear Flip-Flop



S'	C'	Q_{n+1}
1	1	0

JK-MS Flip-Flop



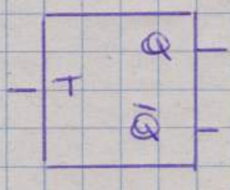
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$ *

* Stel

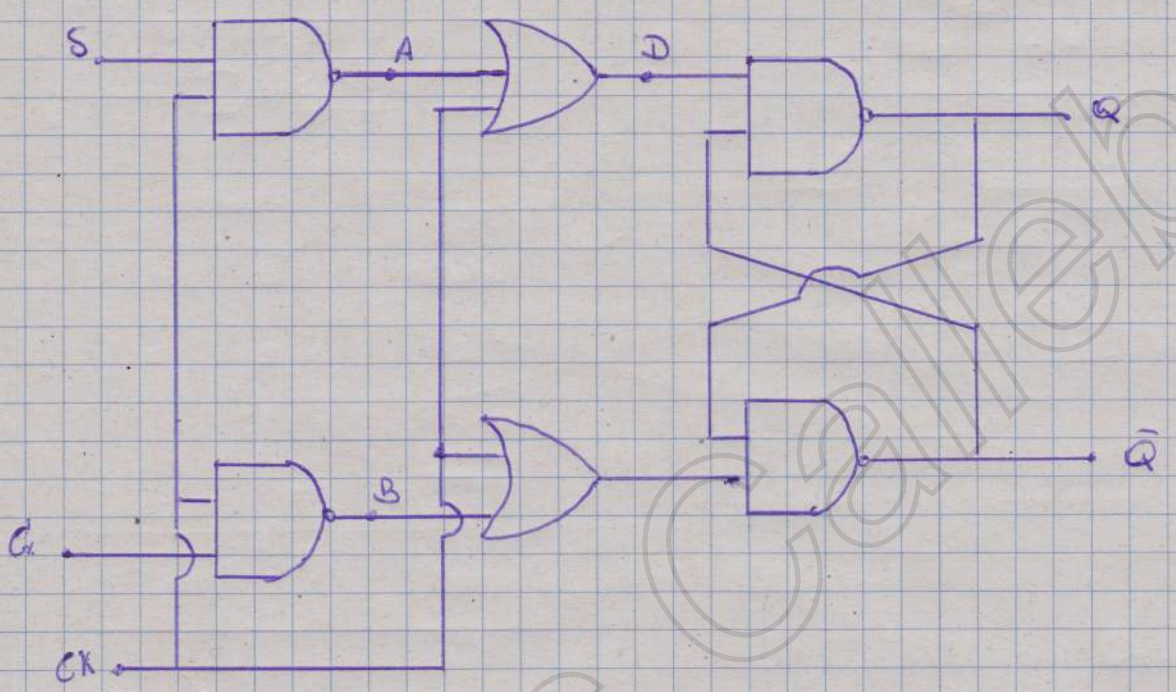
$$\left.\begin{array}{l} \left.\begin{array}{l} Q_n = 1 \rightarrow C = 1 \\ \bar{Q}_n = 0 \rightarrow S = 0 \end{array}\right\} \Rightarrow Q_{n+1} = 0 \\ \\ \left.\begin{array}{l} Q_n = 0 \rightarrow C = 0 \\ \bar{Q}_n = 1 \rightarrow S = 1 \end{array}\right\} \Rightarrow Q_{n+1} = 1 \end{array}\right\} \Rightarrow \bar{Q}_n$$

Als $J = K = 1$ → bij elke CK puls → uitgang omkippen.

Toggle Flip-Flop



Flankgetriggerde Flip-Flop

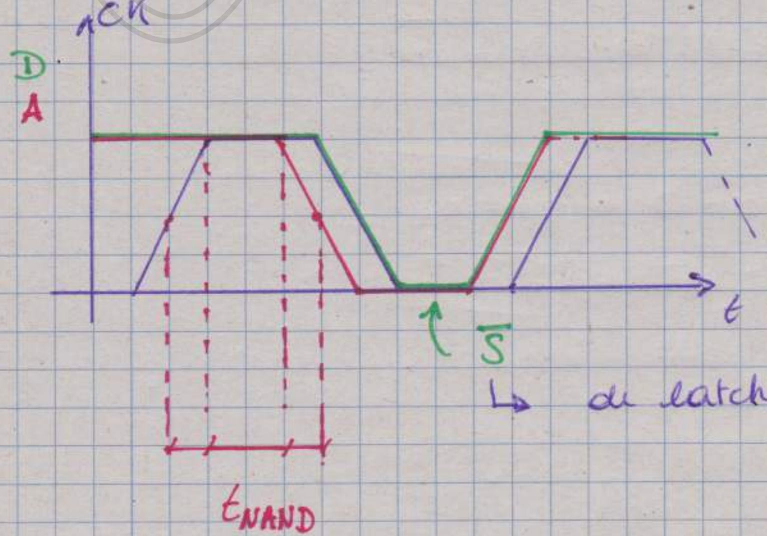


CK transitie $\rightarrow$ ingangen OR poorten L-niveau
H → L

→ door eindige reactietijd poorten ⇒ NAND poorten waarde toen CK nog H was.

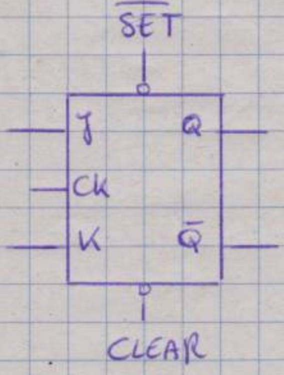
we stellen G = 0 & S = 1

we stellen $t_{OR} = 0$

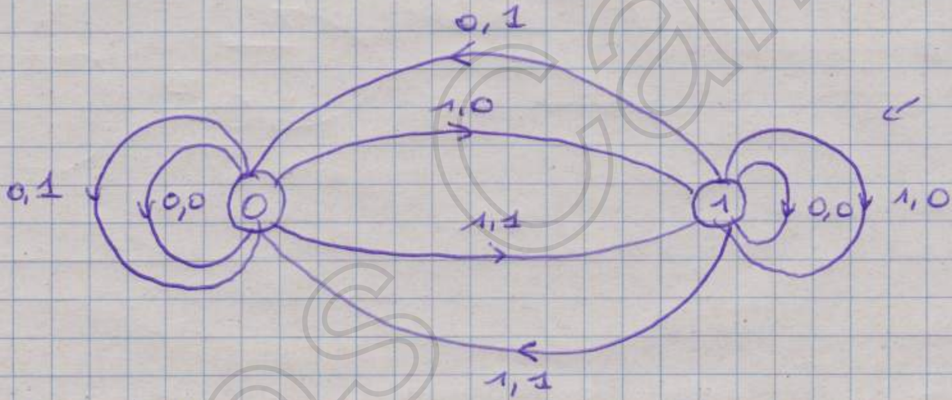


↳ de latch $\bar{w}$ geset (door D)

JK - Flip-Flop



Q(t)	J	K	Q(t+T)
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	0



transitiediagr.

transitietabel: (= excitatietabel)

Q(t)	Q(t+T)	J(t)	K(t)
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

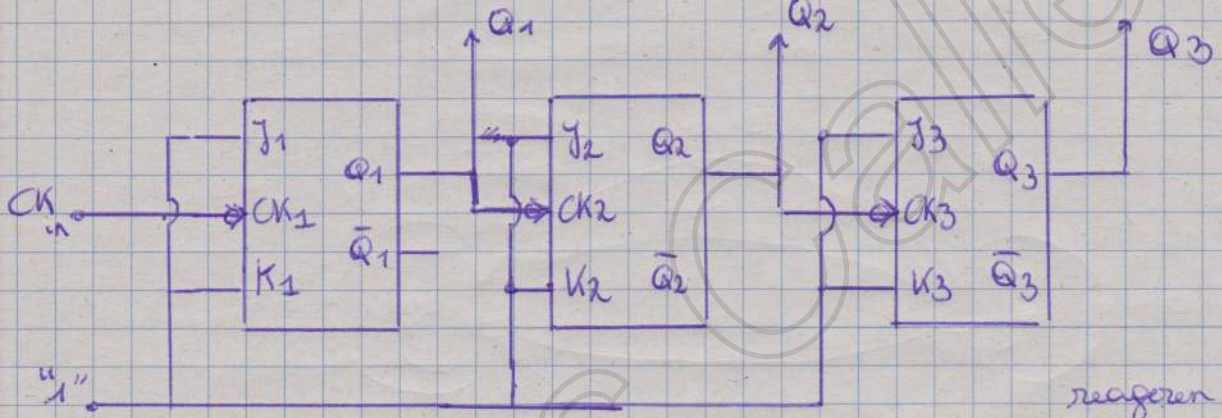
⇗

vanbuiten kennen.

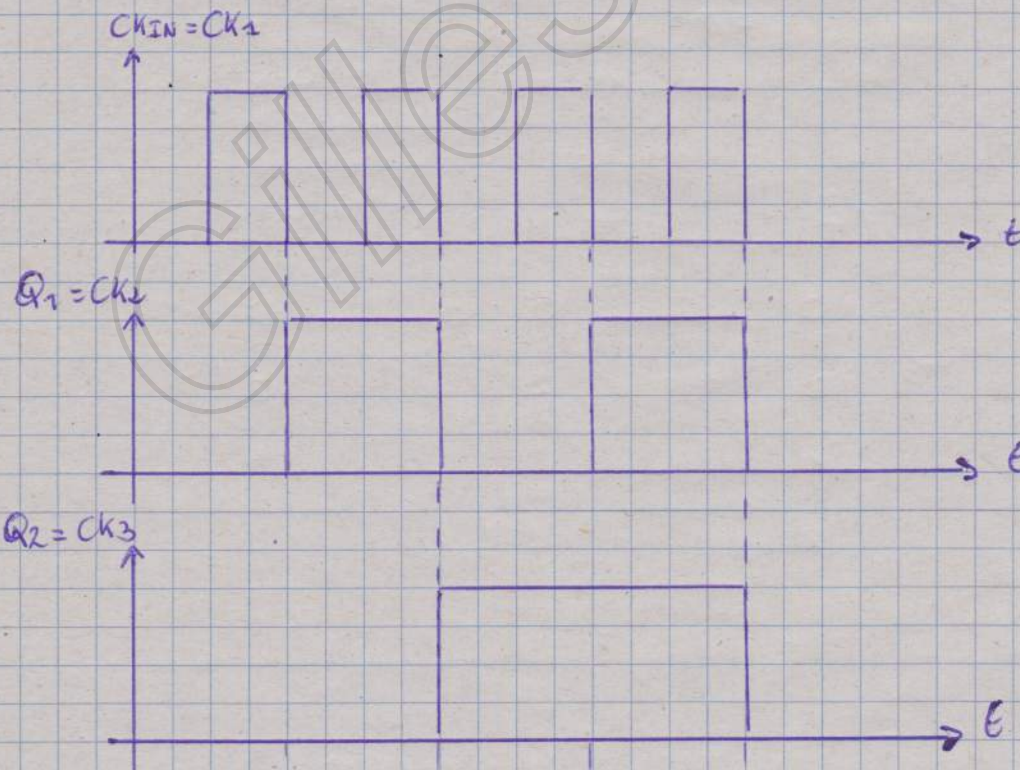
Synchrone sequentiële netwerken

Asynchrone tellers

Asynchr. volledige binaire tellers



reageren op dalende flank



...

Q3 Q2 Q1

000 | 001 | 010 | 011 → modulo-2^n teller.

Asynchrone, afgeknotte, binaire tellers (fig5 p7)

BCD $b_3\ b_2\ b_1\ b_0$

Als b_3 & $b_1 = 1$ → clear via NAND.

probleem: Als één flip-flop sneller → '0' (gecleared is)
dan verdwijnt de clear-puls
→ onzeker of andere flip-flops gecleared zijn.

Parallel laden v.e teller

lees p7 & 8 (nt zo belangrijk denk ik)

Synchrone tellers

Eenvoudige binaire teller ← eens lezen (niet zo belangrijk denk ik)

Gray-code teller met JK-Flip-Flops

Excitatietabel voor JK-Flip-Flops

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Cki	huidige staat			volgende staat			huidige synchr. ingangen					
	Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	1	0	X	1	X	X	0
2	0	1	1	0	1	0	0	X	X	0	X	1
3	0	1	0	1	1	0	1	X	X	0	0	X
4	1	1	0	1	1	1	X	0	X	0	1	X
5	1	1	1	1	0	1	X	0	X	1	X	0
6	1	0	1	1	0	0	X	0	0	X	X	1
7	1	0	0	0	0	0	X	1	0	X	0	X

Karnaugh kaarten opstellen
voor de huidige synchr. ingangen.

vb. K_2 :

Q_3 \ Q_2Q_1	00	01	11	10
0	$X^{=1}$	0	1	$X^{=0}$
1	$X_{=0}$	1	0	$X^{=1}$

↲ huidige staat

$\Rightarrow \quad K_2 = Q_2 \oplus Q_3$

Möbiusteller

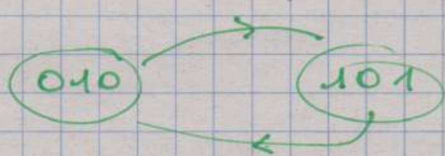
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	? =1	? =0	? =1	X =1	X =0	X =0	X =1	X =1	X =0
0	1	1	1	1	1	1	X	X	0	X	0
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	? =0	? =1	? =0	X	X	X	X	X	X
1	1	0	1	0	0	X	0	X	1	0	X
1	1	1	1	1	0	X	0	X	0	X	1

$\Rightarrow$ $J_3 = Q_2$ $\quad K_3 = \overline{Q_2}$

$J_2 = Q_1$ $\quad K_2 = \overline{Q_1}$

$J_1 = \overline{Q_3}$ $\quad K_1 = Q_3$

hang-up staat



vullen ??? zodat