Inhoudsopgave

11 Digitale techniek (korte versie)

11.1 Wat leer je in dit hoofdstuk

Digitale techniek is gebaseerd op het tweetallige (binaire) getalstelsel dat als enige cijfers de 0 en de 1 kent. We leggen verbanden met het voor ons "normale" tientallige (decimale) stelsel en met het zestientallige (hexadecimale) stelsel.

Daarna volgen zogenoemde logische functies in de vorm van poortschakelingen en hoe je daaraan kunt rekenen met Booleaanse algebra die ook wel *schakelalgebra* wordt genoemd. Als sluitstuk van dit deel dat ook wel *combinatielogica* heet, behandelen we een optelschakeling voor binaire getallen met de zogenoemde *half adders* en *full adders*.

Vervolgens komt een stuk *sequentiële logica* aan bod. Daarin wordt behalve van poorten gebruik gemaakt van opslag van gegevens (data) in zogenoemde registers. Die registers hebben een geheugenfunctie, maar bevatten vrijwel altijd ook poorten.

Al deze schakelingen worden tegenwoordig niet meer gemaakt van losse transistoren of FETs, maar van geïntegreerde schakelingen waarin heel veel van die elementen zijn samengebracht op één Si-chip.

11.2 Logische functies en schakelingen

11.2.1 Inleiding

Logische functies zijn functies rondom de begrippen "waar" of "niet waar". Die worden aangeduid met de getallen 0 of 1. 0 en 1 worden voorgesteld door spanningen. De spanning voor de 0 ligt meestal onder de 0,8 V en voor de 1 tussen de halve en de hele (positieve) voedingsspanning. Wiskundig gezien kan het ook andersom. Dat heet *negatieve logica,* maar die is geen onderdeel van het zendexamen.

Het tientallige stelsel is ongeschikt voor logische schakelingen. Dan zou een schakeling 10 goed te onderscheiden toestanden (lees: spanningen) moeten kunnen weergeven en die ook nog eens feilloos kunnen onderscheiden. Als dat miljoenen keren per minuut moet gebeuren, is dat voor onze elektronica te veel gevraagd. Vandaar dat zonder uitzondering het eenvoudigst denkbare getalstelsel wordt gebruikt: het tweetallige of binaire.

11.2.2 Digitaal en analoog

Digitale schakelingen is een andere term voor *logische schakelingen*. *Digitaal* is afgeleid van het Latijnse woord voor vinger, *digitus*. Rekenen op je vingers is vooral iets voor de lagere klassen van de basisschool. Dat we voor dagelijks gebruik het tientallige getalstelsel hebben, is niet vreemd. "Dat kun je op je vingers natellen" is een bekende uitdrukking.

Een binair cijfer, afgekort tot *bit* van het Engelse ***bi**nary digit* kan 0 of 1 zijn. Meer mogelijkheden zijn er niet. Met 2 bits zijn 2^2=4 combinaties mogelijk (0, 1, 10 en 11), met 3 bits maak je 2^3=8 combinaties en zo verder. "Digitaal" betekent altijd een eindig aantal bitcombinaties.

Tot nu toe hebben we ons in deze cursus niet bekommerd om dit soort zaken. Dat kwam doordat we steeds bezig zijn geweest met *analoge* systemen. In theorie kennen die een oneindig aantal toestanden. Een spanning, stroom of vermogen kan oneindig veel waarden hebben. In theorie kun je die meten, maar in de praktijk is de nauwkeurigheid van de meting beperkend. Is die bijvoorbeeld 0,1% van de maximale waarde, dan zijn verschillen kleiner dan die 0,1% niet vast te stellen.

11.2.3 Binair, decimaal en hexadecimaal

Binaire getallen hebben het getal 2 als basis, decimale getallen het getal 10 en hexadecimale het getal 16. Dat stond al in hoofdstuk 2, maar we doen een korte herhaling.

Het tweetallige stelstel kent twee cijfers, 0 en 1

Het tientallige stelsel kent er tien: 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9

Het zestientallige stelsel kent er zestien: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E en F

De betekenis van de cijfervolgorde in een getal is bij alle drie dezelfde. Van (let op!) rechts naar links: het aantal maal het grondtal *n* tot de macht 0, tot de macht 1, 2 en zo verder. In Tabel 11.2-1 zien we de weergave van het getal 237 in de drie getallenstelsels.

Tabel 11.2-1. Weergave van het getal 237 in binair, decimaal en hexadecimaal

Basis	n^7	n^6	n^5	n^4	n^3	n^2	n^1	n^0
2	1	1	1	0	1	1	0	1
10						2	3	7
16							E	D

Het getal is als volgt terug te rekenen. Voor tweetallig naar decimaal: $1.2^0 + 0.2^1 + +1.2^2 + 1.2^3 + 0.2^4 + 1.2^5 + 1.2^6 + 1.2^7 = 1 + 0 + 4 + 8 + 0 + 32 + 64 + 128 = 237$

Voor decimaal naar decimaal: $7.10^0 + 3.10^1 + 2.10^2 = 7 + 30 + 200 = 237$

Voor hexadecimaal naar decimaal: $13.16^0 + 14.16^1 = 13 + 14.16 = 13 + 224 = 237$

Hoe groter het grondtal, des te korter is de schrijfwijze van een getal. Dat is de reden voor het gebruik van het zestientallige stelsel in de computertechniek. Binaire getallen zijn prachtig voor een machine, maar lastig voor ons menselijk brein. Het getal 16 is 2^4 en de cijferreeks 0-15 omvat dus precies 4 bits. Zestientallig omvat dus opeenvolgende blokken van 4 bits. De rechter 1101 van het binaire getal is gelijk aan 13 decimaal en D hexadecimaal. De linker 1101 is 14 decimaal en E hexadecimaal (of kortweg “hex”).

Hoe reken je nu decimaal om naar binair? Het gaat in drie stappen:

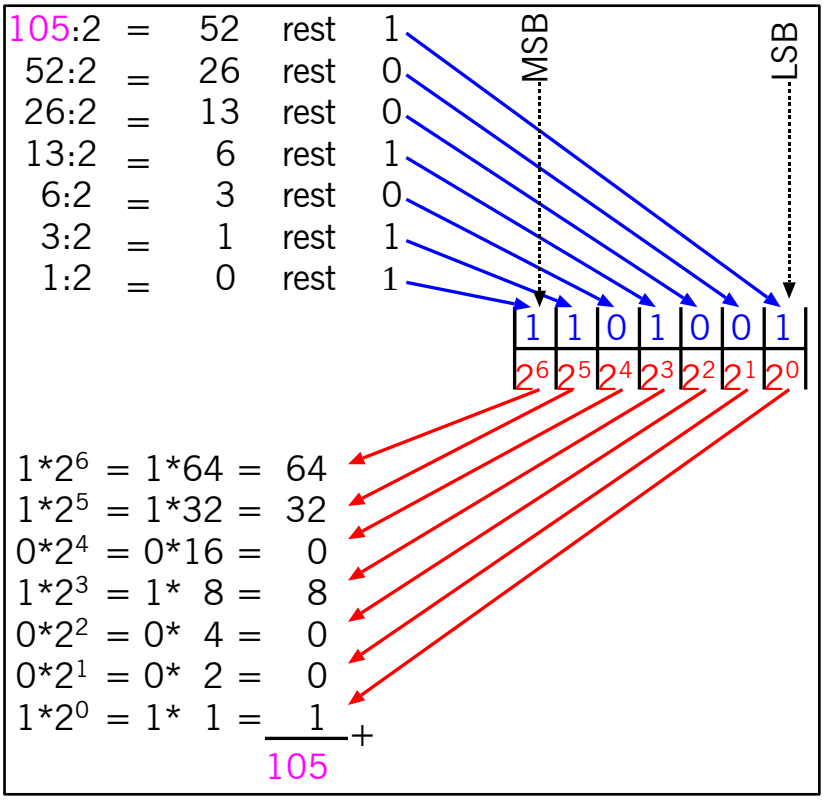
Stap 1. Is het meest rechtse decimale cijfer oneven of even? Is het oneven, dan is het meest rechtse binaire cijfer een 1, zoals in Tabel 11.2-1 te zien is. Is het even, dan is het meest rechtse binaire cijfer een 0. Dat cijfer wordt wel aangeduid met de afkorting “LSB”. Dat staat voor “Least Significant Bit”, het minst betekenende bit.

Stap 2. Trek het gevonden binaire cijfer (dus 0 of 1) af van het oorspronkelijke decimale. Deel de uitkomst door 2. Is het meest rechtse cijfer van het gevonden decimale getal oneven of even? In andere woorden: levert delen door 2 een rest van 1 of van 0? Is de rest 1, dan komt links van het laatst gevonden binaire cijfer een 1; is de rest 0, dan komt daar een 0.

Stap 3. Herhaal stap 2 tot de hele uitkomst 0 is, dus zonder rest. Het laatst ingevulde binaire cijfer wordt wel aangeduid met "MSB", "Most Significant Bit".

LSB en MSB worden ook wel geschreven als LSD en MSD, waarbij de D staat voor "Digit".

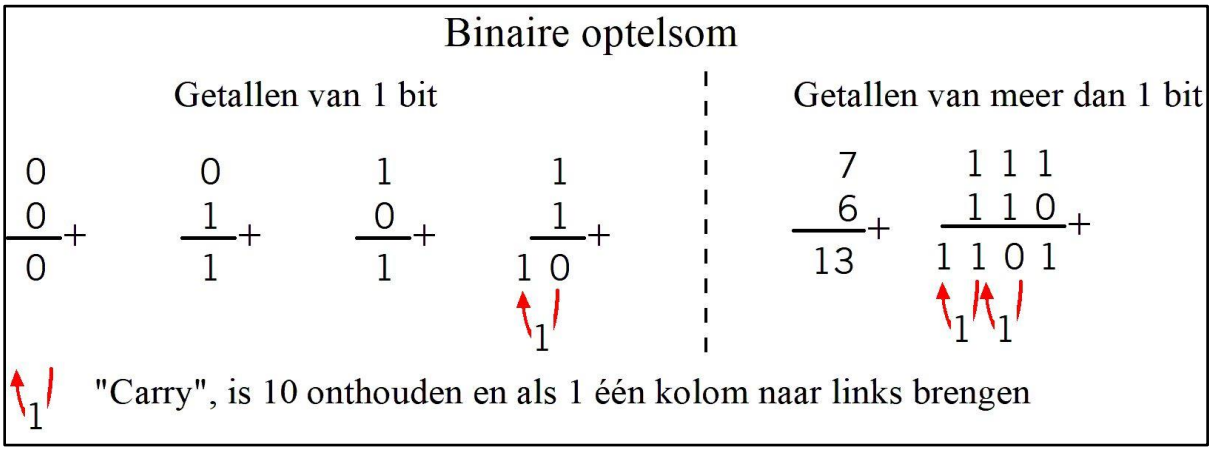
Een voorbeeld met het getal 105 decimaal is uitgewerkt in Figuur 11.2-1. De terugrekening naar decimaal die we voor Tabel 11.2-1 al hebben toegepast, staat erin.



Figuur 11.2-1. Omzetting van het decimale getal 105 naar het binaire getal 1101001 met dezelfde waarde en daarna weer terug. MSB staat voor "Most Significant Bit"; LSB voor "Least Significant Bit". Blauwe pijlen: decimaal naar binair; rode pijlen: binair naar decimaal.

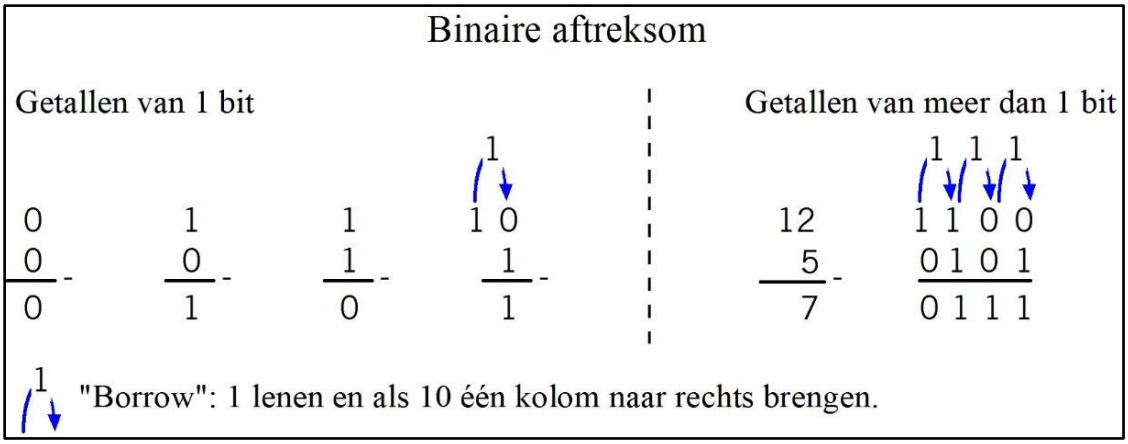
11.2.4 Binair rekenen

Doordat de opbouw van de getallen dezelfde is, zijn de rekenregels voor binaire getallen dezelfde als die voor decimale. Bij binair rekenen is 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1 en 1 + 1 = 10. 10 binair is 2 decimaal. Bij een optelling met een optelstreep eronder komt dan een 0 te staan en is het "1 bewaren" voor de volgende kolom cijfers (Figuur 11.2-2).



Figuur 11.2-2. Binair optellen. Links de basisbewerkingen met enkelvoudige bits; rechts met meerdere bits.

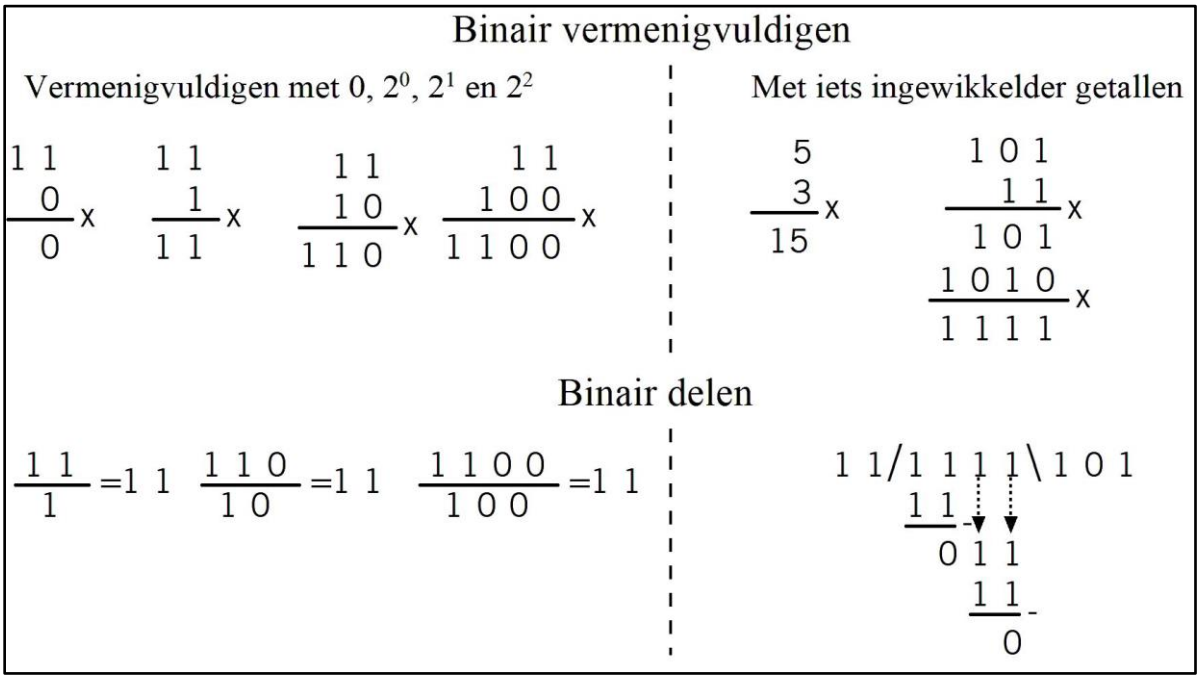
Met binair aftrekken is $1 - 0 = 1, 1 - 1 = 0$ en $10 - 1 = 1$ (Figuur 11.2-3)



Figuur 11.2-3. Binair aftrekken. Links de basisbewerkingen met enkelvoudige bits; rechts met meerdere bits.

Binair vermenigvuldigen is schuiven naar links. Een getal met twee vermenigvuldigen is er rechts een 0 tegenaan zetten. Net als bij vermenigvuldigen met 10 in het decimale stelsel.

Binair delen moet dan wel schuiven naar rechts zijn en dat is het ook. Bij delen door 10 (2 decimaal) gaat het getal een plekje naar rechts. Als rechts een 1 staat, wordt dat een “rest”. Je kunt verder werken met negatieve machten (cijfers achter de komma), maar dat doen we hier niet.

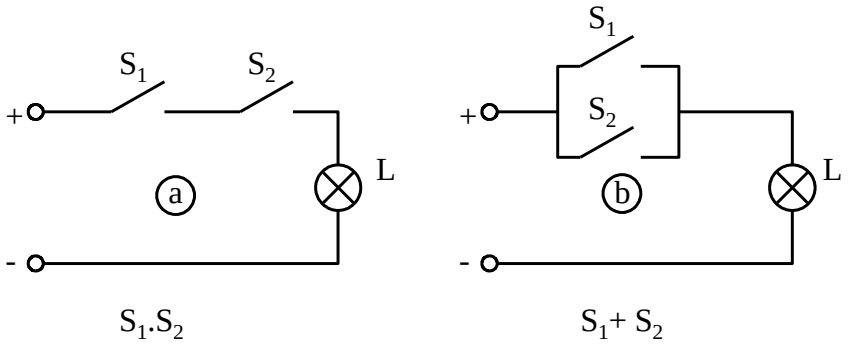


Figuur 11.2-4. Binair vermenigvuldigen en delen. Vermenigvuldigen is schuiven naar links, delen naar rechts. Zelfs een klassieke staartdeling werkt.

11.3 Logische functies

11.3.1 Van 1 en 0 tot tegenstelling

1 en 0 hoeven niet noodzakelijkerwijs een getalbetekenis te hebben. “Waar” en “niet waar”, “ja” en “nee” of “aan” en “uit” kunnen evengoed. Van dit laatste geven we een voorbeeld in Figuur 11.3-1.



Figuur 11.3-1. a. EN (AND)-schakeling met schakelaars, b. OF (OR)-schakeling met schakelaars.

In afbeelding a zijn de schakelaars S_1 en S_2 beide open (uit). Het lampje L brandt pas als S_1 èn S_2 gesloten (aan) zijn. In afbeelding b moet S_1 óf S_2 gesloten (aan) zijn om L te laten branden. De schakeling links heet daarom een EN-schakeling, die rechts een OF-schakeling. In symbolentaal:

$L = S_1 \wedge S_2 \equiv L = S_1.S_2$ voor afbeelding a en $L = S_1 \vee S_2 \equiv L = S_1 + S_2$ voor afbeelding b. Het symbool "≡" betekent: "is precies hetzelfde als". Spreek afbeelding a uit als: **L is S_1 èn S_2**. Voor afbeelding b: **L is S_1 òf S_2**. Je kunt het gedrag van beide schakelingen in de vorm van 0 en 1 samenvatten in een *waarheidstabel*, zoals in Tabel 11.3-1.

Tabel 11.3-1 Waarheidstabellen voor de schakelingen a en b in Figuur 11.3-1.

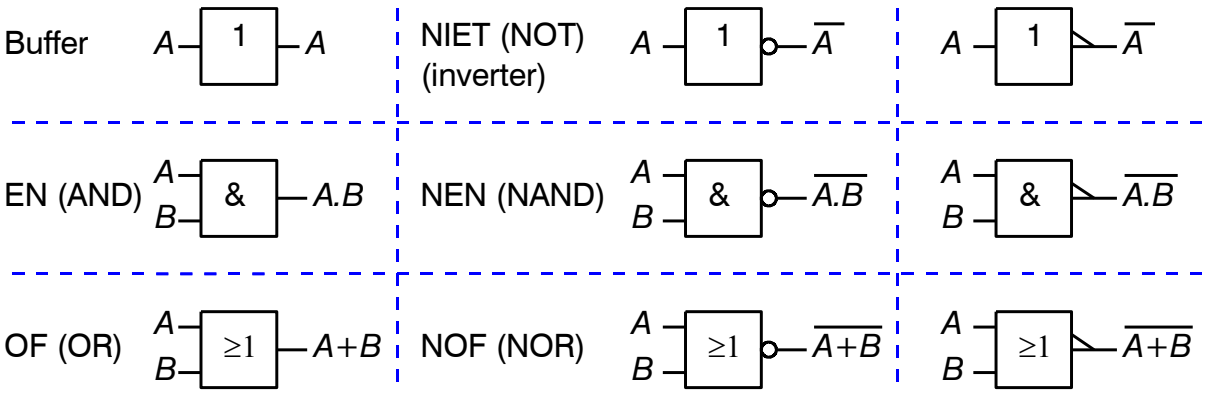
Schakeling a			**Schakeling b**		
$L = S_1 \wedge S_2 = S_1.S_2$			$L = S_1 \vee S_2 = S_1 + S_2$		
S_1	S_2	L	S_1	S_2	L
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

Nog iets over de gebruikte symbolen: "∧" en "." staan voor EN. Ze mogen beide worden gebruikt. De "." kom je het meest tegen. "∨" en "+" staan voor OF. Je komt de "+" bijna alleen maar tegen. Verwar "." en "+" niet met de precies gelijke vermenigvuldig- of opteltekens. Uit het verband van de tekst blijkt vanzelf waarover het gaat.

11.3.2 Poorten

De schakelingen in Figuur 11.3-1 kun je zien als poorten. De stand van de schakelaars gaat erin. Eruit komt wel of geen licht van het lampje. Bij poortschakelingen in de digitale elektronica is de invoer een spanning en de uitgang ook. Zoals al eerder gezegd: een spanning tussen de halve en de hele voedingsspanning staat doorgaans voor een 1 en minder dan 0,8 V voor een 0. Omdat meestal de voedingsspanning 5 V of 3,5 V is, is meetnauwkeurigheid niet vereist en worden de waarden 0 en 1 in logische schakelingen feilloos onderscheiden.

De belangrijkste poorten zien we in Figuur 11.3-2.



In plaats van "." mag ook "∧" worden geschreven en in plaats van "+" ook "∨"

Figuur 11.3-2. Poorten en wat eruit komt in IEC-tekenwijze. Links de poorten zonder NIET, rechts die met NIET. De streep boven uitdrukkingen is een negatiestreep of NIET-streep. Het rondje of driehoekje geeft een genegeerde ofwel NIET-uitgang aan.

De tekenwijze is volgens de norm van IEC (**I**nternational **E**lectrotechnical **C**ommission), in Europa de standaard. Er zijn ook Amerikaanse symbolen. Omdat ze veel worden gebruikt, staan ze wel in de volledige tekst, maar ze zijn geen exameneis en dus staan ze er in deze verkorte tekst niet bij.

Poorten die een NIET maken, zijn aan de uitgang voorzien van een rondje of een driehoekje. Het rondje wordt verreweg het meest gebruikt. De uitkomsten hebben een streep boven de symbolen. Die heet ook wel *negatiestreep*. $\bar{A}$ betekent dus NIET A en de uitdrukking $\overline{A+B}$ betekent NIET A of B.

De eenvoudigste zijn de buffer en de NIET-poort. Wat er bij de buffer ingaat, komt er ook uit. Toepassing: als een poortuitgang meer ingangen moet aansturen dan waarvoor hij gemaakt is. De NIET-poort maakt van een 0 een 1 en van een 1 een 0. Daarom heet hij ook wel *inverter*, wat *omkeerder* betekent.

De bijbehorende waarheidstabellen zijn de eenvoudigste van allemaal (Tabel 11.3-2)

Tabel 11.3-2. Waarheidstabellen voor de buffer (links) en voor de NIET-poort of inverter (rechts)

$Q = A$	
Ingang (A)	Uitgang (Q)
1	1
0	0

$Q = \bar{A}$	
Ingang (A)	Uitgang (Q)
1	0
0	1

De wereld in zijn eenvoudigste vorm, zogezegd.

Voor de EN-poort geldt dat de uitgang 0 is, tenzij A èn B 1 zijn. Voor de NEN geldt het omgekeerde: de uitgang is 1, tenzij A èn B 1 zijn. De waarheidstabellen in Tabel 11.3-3 laten het zien.

Tabel 11.3-3. Waarheidstabellen voor de EN-poort (links) en voor de NEN-poort (rechts)

$Q = A.B \equiv A \wedge B$		
A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

$Q = \overline{A.B} \equiv \overline{A \wedge B}$		
A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

De tabellen laten zien dat één ingang met een 0 de invloed van de andere ingang op de toestand Q van de uitgang blokkeert. Datzelfde geldt voor EN- en NEN-poorten met meer dan 2 ingangen. Het verschil tussen EN en NEN is dat de uitgang (Q in de tabel) bij de EN dan 0 blijft en bij de NEN 1. Dat komt door de ingebouwde inverter in de NEN-poort. In een EN-poort ontbreekt die.

Tabel 11.3-4 laat de waarheidstabellen voor OF- en NOF-poort zien.

Tabel 11.3-4. Waarheidstabellen voor de OF-poort (links) en voor de NOF-poort (rechts)

$Q = A + B \equiv A \vee B$		
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	1

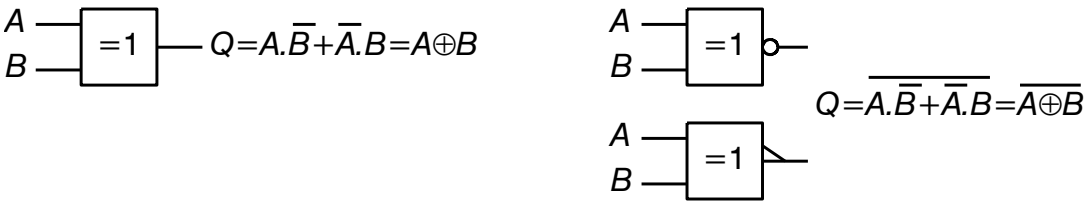
$Q = \overline{A + B} \equiv \overline{A \vee B}$		
A	B	Q
0	0	1
1	0	0
0	1	0
1	1	0

Voor de OF-poort geldt dat de toestand Q van de uitgang 1 is als maar één van de ingangen 1 is. Dat geldt ook voor OF-poorten met meer dan 2 ingangen. Hetzelfde geldt voor de NOF-poort, met dit verschil dat Q dan 0 is. Ook hier: kwestie van aan- of afwezigheid van een inverter vóór de poortuitgang.

Om te onthouden.

1. Een EN of NEN met een 0 op één enkele ingang is ongevoelig voor wat er op de andere ingang(en) gebeurt. De EN heeft dan een 0 op de uitgang, de NEN een 1.
2. Een OF of NOF met een 1 op één enkele ingang is ongevoelig voor wat er op de andere ingang(en) gebeurt. De OF heeft dan een 1 op de uitgang, de NOF een 0.

Dan zijn er nog twee bijzondere leden van de familie Poort. Dat is de Exclusieve OF en de Exclusieve NOF. Eigenlijk bestaan ze uit meer dan één poort en het aantal ingangen is altijd 2. Het bijzondere eraan is dat de uitgang aangeeft of de toestand op beide ingangen dezelfde of is of verschilt. We beginnen met de IEC-schemasymbolen (Figuur 11.3-3).



Figuur 11.3-3. Schemasymbolen met vergelijking voor EXOF (links) en EXNOF (rechts).

Het symbool voor Exclusieve OF is ⊕, een OF-teken met een rondje eromheen. De waarheidstabellen staan in Tabel 11.3-5.

Tabel 11.3-5. Waarheidstabellen voor de EXOF-poort (links) en voor de EXNOF-poort (rechts).

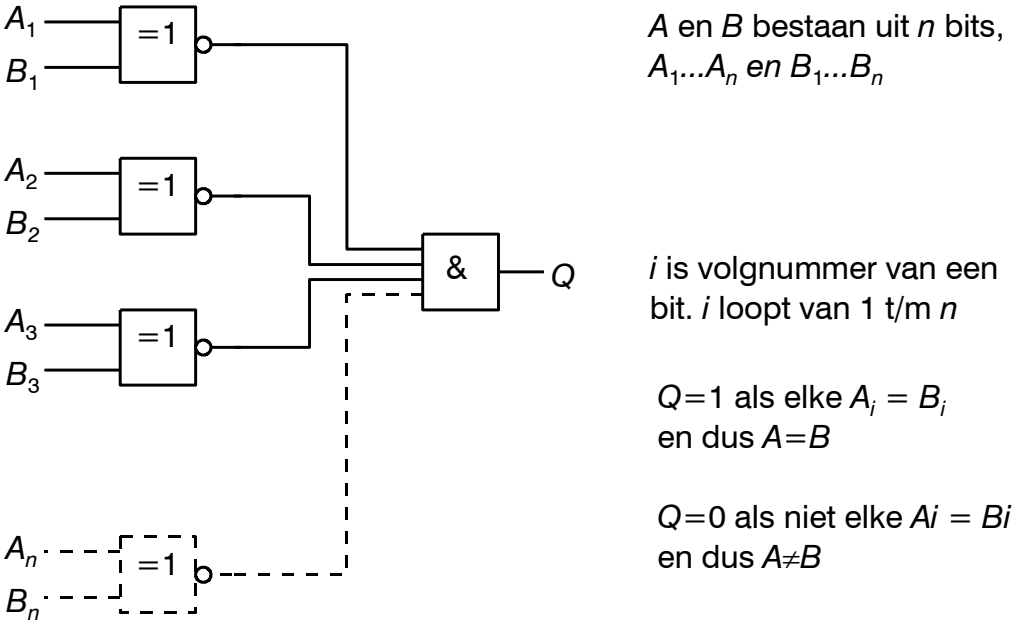
$Q = A.\bar{B} + \bar{A}.B = A \oplus \mathrm{B}$		
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	0

$Q = \overline{A.\bar{B} + \bar{A}.B} = \overline{A \oplus \mathrm{B}}$		
A	B	Q
0	0	1
1	0	0
0	1	0
1	1	1

Voor de EXOF-poort is het resultaat $Q = 1$ als A en B ongelijk zijn. De uitkomst is 0 als ze gelijk zijn. Voor de EXNOF geldt vanzelfsprekend het omgekeerde: is A ongelijk aan B, dan geldt $Q = 0$; is $A = B$, dan geldt $Q = 1$.

De EXOF kun je gebruiken om vast te stellen of twee binaire getallen gelijk of ongelijk aan elkaar zijn. Je hebt dan een rij EXOF's (of EXNOF's) nodig. Evenveel poorten als de getallen bits hebben. 15 bits per getal betekent 15 stuks EXOF of EXNOF. In Figuur 11.3-4 staat een voorbeeld met EXNOF's en een EN-poort met net zoveel ingangen als er bits zijn.

Het kan ook met EXOF's. Dan moet de EN-poort in het schema een OF-poort worden.



Figuur 11.3-4. Vergelijken van twee binaire getallen die elk bestaan uit n bits, $A_1 \ldots A_n$ en $B_1 \ldots B_n$. De poorten zijn EXNOF's. De EN-poort moet n ingangen hebben. Die kan eventueel uit meer dan één EN-poort worden samengesteld.

Hoe je een EN-poort met een heleboel ingangen kunt maken uit kleinere poorten, zie je nadat we de Booleaanse algebra hebben behandeld.

11.4 Booleaanse algebra (schakelalgebra)

11.4.1 Inleiding

Bij de verschillende poorten stonden algebra-achtige uitdrukkingen. Daarmee zijn inderdaad algebra-achtige bewerkingen mogelijk. Een ontworpen schakeling kan ermee worden vereenvoudigd en er kan worden gecontroleerd of hij doet wat de bedoeling is. Dit rekenwerk met zijn eigen regels is ontworpen en in 1847 gepubliceerd door de Britse wiskundige George Boole. Het meeste spreekt redelijk vanzelf, maar soms is het opletten. De stellingen van De Morgan (aan het eind) zijn de belangrijkste.

11.4.2 Regels met 1 of 0

Er zijn maar twee binaire getallen en ook maar twee regels.

$$\overline{1} = 0$$

$$\overline{0} = 1$$

Zoals in de volledige tekst van dit hoofdstuk ook al staat: meer smaken zijn er niet.

11.4.3 Regels met één variabele

De variabele heet hier A. A stelt één binair cijfer voor en dat kan 0 of 1 zijn.

$$\overline{\overline{A}} = A \qquad (11.4\text{-}1)$$

Twee strepen erboven, NIET NIET heft elkaar op, net als twee mintekens achter elkaar.
$\overline{\overline{1}} = 1$, $\overline{\overline{0}} = 0$, $\overline{\overline{Pietje}} = Pietje$

$$A.A = A \qquad (11.4\text{-}2)$$

Als $A = 0$, dan is $A.A = 0$; is $A = 1$, dan is $A.A = 1$. De punt in de gelijkheid $A.A = A$ kun je ook schrijven als $\wedge$, dus $A \wedge A = A$.

Ook geldt:

$$A.1 = A \qquad (11.4\text{-}3)$$

(11.4-3) zegt dat als één ingang van een EN-poort met 2 ingangen 1 is, dan volgt de uitgang de toestand op de andere ingang. Dat kwam al aan de orde bij de behandeling van de EN-poort.

En dan krijgen we meteen het broertje van (11.4-3) met voor de verandering "$\wedge$" i.p.v. ".":

$$A \wedge 0 = 0 \qquad (11.4\text{-}4)$$

Is één ingang van een EN-poort 0, dan is de uitgang altijd 0.

Voor OF-poorten is er iets dergelijks:

$$A + A = A \qquad (11.4\text{-}5)$$

Nu volgt het OF-zusje van (11.4-4)

$$A + 0 = A \qquad (11.4\text{-}6)$$

En het OF-zusje van (11.4-3):

$$A + 1 = 1 \qquad (11.4\text{-}7)$$

Een tenslotte de OF-versie van (11.4-2):

$$\bar{A} + A = 1 \qquad (11.4\text{-}8)$$

In (11.4-8) is altijd één van de twee variabelen een 1, zodat de uitkomst altijd 1 is.

11.4.4 Regels met twee of meer variabelen

Verwisseling van volgorde

$$A.B = B.A \qquad (11.4\text{-}9)$$

Bij een EN-poort met twee ingangen maakt het niets uit wat aan welke ingang hangt. Bij drie, vier en meer ingangen geldt hetzelfde. Dat geldt voor alle soorten poorten. Dus ook:

$$A + B = B + A \qquad (11.4\text{-}10)$$

Of

$$\overline{A \oplus B} = \overline{B \oplus A} \qquad (11.4\text{-}11)$$

Een EXOF heeft altijd twee ingangen, nooit meer!

Associatieve en distributieve wetten

Vertaald: *verbindingswetten* en *verdelingswetten.* Die gaan over het gebruik van haakjes. Verbindingswetten hebben betrekking op haakjes in vergelijkingen met één soort bewerking, verdelingswetten op alle andere. De betekenis van haakjes is vergelijkbaar met die in de gewone rekenkunde: wat tussen haakjes staat, doe je eerst. Er is één verschil: er is geen speciale voorrang voor bepaalde bewerkingen, zoals we dat wel kennen van de rekenkunde. Daarin heeft bijvoorbeeld vermenigvuldigen voorrang boven optellen.

$$A.(B.C) = (A.B).C = A.B.C \qquad (11.4\text{-}12)$$

Van twee EN-poorten met twee ingangen, maak je zo effectief één EN-poort met drie ingangen. $B.C$ is één poort en de uitkomst gaat met A de volgende poort in. Of $A.B$ is er één en de uitkomst gaat met C de volgende poort in. Dat komt in de volgende paragraaf over combinatielogica aan de orde.

$$A + (B + C) = (A + B) + C = A + B + C \qquad (11.4\text{-}13)$$

Hetzelfde verhaal als bij (11.4-12): van twee OF-poorten met twee ingangen maak je er effectief één met drie ingangen.

Nu twee verschillende soorten bewerkingen, EN en OF in één schakeling:

$$A.(B + C) = A.B + A.C \qquad (11.4\text{-}14)$$

(11.4-14) gaat net als bij de rekenkunde. Maar de volgende wijkt af doordat OF en EN gelijkwaardig zijn. **Eén om te onthouden**!

$$A + (B.C) = (A + B).(A + C) \qquad (11.4\text{-}15)$$

Absorptiewetten

Dit zijn wetten waarbij iets verdwijnt, waardoor vereenvoudiging optreedt. Er zijn vier van die verdwijntrucs. Maak eventueel voor elk een waarheidstabel.

$$A + A.B = A \qquad (11.4\text{-}16)$$

Want als $A = 0$, komt er 0 uit en als $A = 1$, is de uitkomst 1.

$$A.(A + B) = A \qquad (11.4\text{-}17)$$

Eigenlijk hetzelfde verhaal als bij (11.4-16), want $A.A = A$ en dan blijft $A + A.B$ over.

De volgende is ook zoiets:

$$A.(\bar{A} + B) = A.B \qquad (11.4\text{-}18)$$

$A.\bar{A} = 0$; daarmee houden we $0 + A.B = A.B$ over. Zie vergelijking (11.4-6).

Dan verdwijntruc nummer 4:

$$A + \bar{A}.B = A + B \qquad (11.4\text{-}19)$$

Deze is wat lastiger te snappen. Maar vul een keer $A = 0$ en een keer $A = 1$ in. Kijk wat er gebeurt. Begin met $A = 0$. Dan krijgen we $0 + 1.B$. Dat is $A + B$, want een EN-poort met een 1 op de ene ingang en B op de andere levert op de uitgang B. Nu A=1: $1 + 0.B = A$. Bij een OF bepaalt de ene ingang die 1 is, dat de uitgang ook 1 is. Die B mag blijven staan, want bij $A = 1$ heeft hij geen invloed op de uitkomst.

Verwar (11.4-19) niet met (11.4-16)!

De stellingen van De Morgan

Zoals in het begin van deze paragraaf gemeld, zijn dit de twee belangrijkste regels van de Booleaanse algebra. We laten ze zien met schema's en vatten ze daarna samen in 1 recept.

De stellingen komen neer op het omzetten van een NEN- naar een OF-poort met genegeerde ingangen (ingangen met ingebouwde NIET-poort) en van een NOF- naar een EN-poort. Hier komen ze.

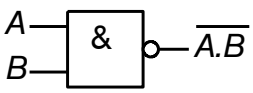
$$\overline{A.B} = \bar{A} + \bar{B} \qquad (11.4\text{-}20)$$

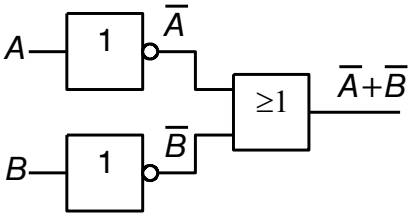
Een NEN-poort voor A en B is dus een OF-poort voor $\bar{A}$ en $\bar{B}$. Datzelfde trucje geldt voor een NOF-poort, maar nu komt er een EN-poort voor $\bar{A}$ en $\bar{B}$ uit, dus ook met genegeerde ingangen.

$$\overline{A + B} = \bar{A}.\bar{B} \qquad (11.4\text{-}21)$$

In Figuur 11.4-1 komen beide vergelijkingen terug in de vorm van schema's en waarheidstabellen.

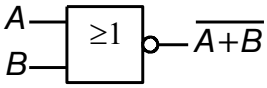
Vergelijking (11.4-20)



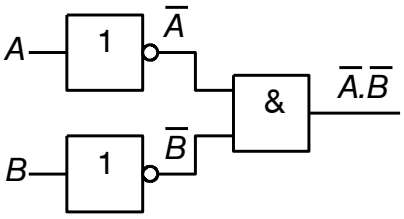


A	B	$\overline{A.B}$	$\overline{A}+\overline{B}$
0	0	1	1
1	0	1	1
0	1	1	1
1	1	0	0

Vergelijking (11.4-21)



A.B is hetzelfde als A∧B



A	B	$\overline{A+B}$	$\overline{A}.\overline{B}$
0	0	1	1
1	0	0	0
0	1	0	0
1	1	0	0

Figuur 11.4-1. Vergelijkingen van De Morgan (11.4-20) en (11.4-21) in beeld met waarheidstabellen.

Het recept:

- Vervang elke EN door OF;
- Vervang elke OF door EN;
- Waar een NIET stond, verdwijnt deze;
- Waar geen NIET stond, komt er één.

In de hoofdtekst heetten deze vier regels samen “het rode recept”.

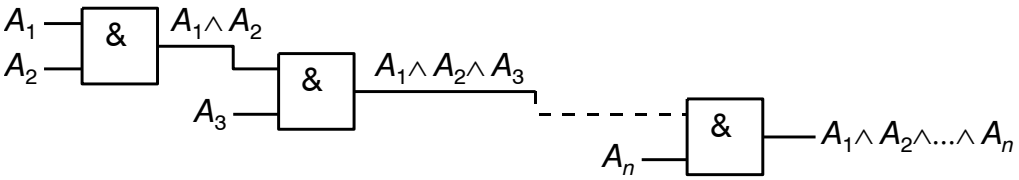
11.5 Combinatielogica (“combinational logic”)

11.5.1 Inleiding

Combinatielogica is het zodanig aan elkaar knopen van poorten dat het geheel een zeker doel dient. Booleaanse algebra helpt hierbij, vooral de stellingen van De Morgan. De Morgan-toepassingen duiken met enige regelmaat op in zendexamens.

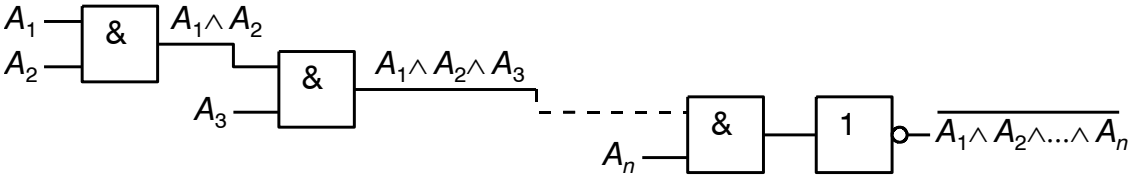
11.5.2 Een EN- of OF -poort met *n* ingangen

Dit is toepassing van de vergelijkingen (11.4-12) en (11.4-13). Een voorbeeld voor EN vinden we in Figuur 11.5-1, voor OF in Figuur 11.5-2. EN- en OF-poorten zijn in IC-vorm met vele aantallen ingangen te krijgen. De meer gangbare hebben er niet meer dan 4, maar aantallen van 5 en zelfs 12 kun je tegenkomen. Zijn veel ingangen nodig, dan kan de benodigde poort ook van meerdere poorten worden gemaakt. De schakeling bevat minder poorten, naarmate poorten met meer ingangen worden gebruikt, want één ingang is altijd nodig om de uitkomst van het voorgaande deel van de rij verder te verwerken.



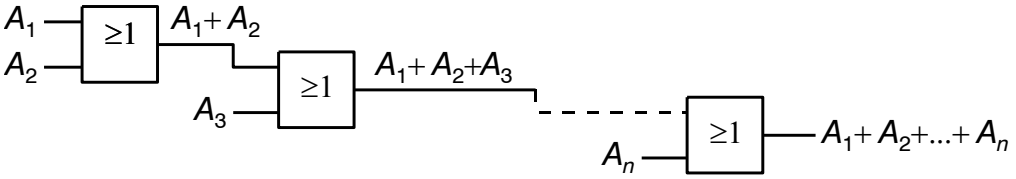
Figuur 11.5-1. Een EN-poort met n ingangen maak je op deze manier.

Voor een grote NEN-poort gebruik je ook EN-poorten op de laatste na, die een NEN moet zijn of een inverter (Figuur 11.5-2).



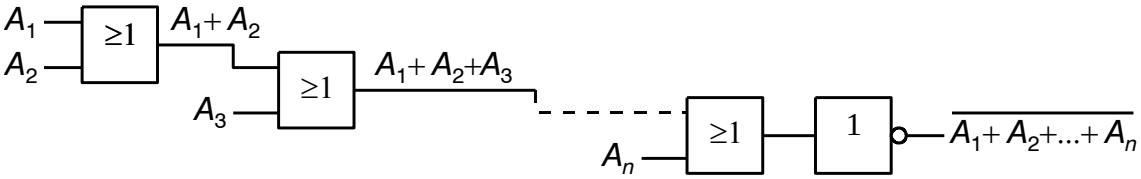
Figuur 11.5-2. De NEN-versie van de schakeling van Figuur 11.5-1.

Voor het maken van heel grote OF-poorten geldt precies hetzelfde als voor EN:



Figuur 11.5-3. Een OF-poort met n ingangen maak je op dezelfde manier als de EN in Figuur 11.5-1.

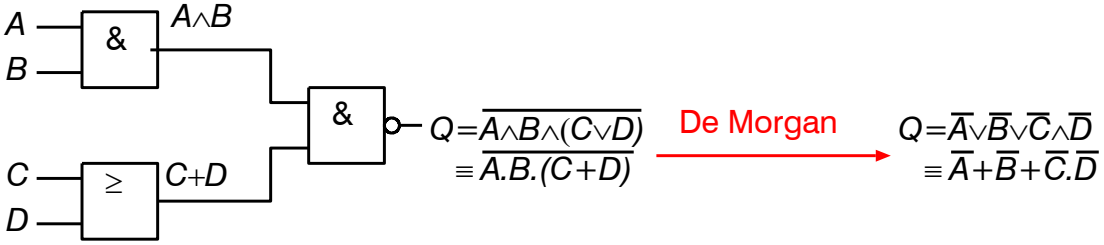
En de grote NOF komt op dezelfde manier tot stand als de grote NEN in Figuur 11.5-2:



Figuur 11.5-4. De NOF-versie van Figuur 11.5-3.

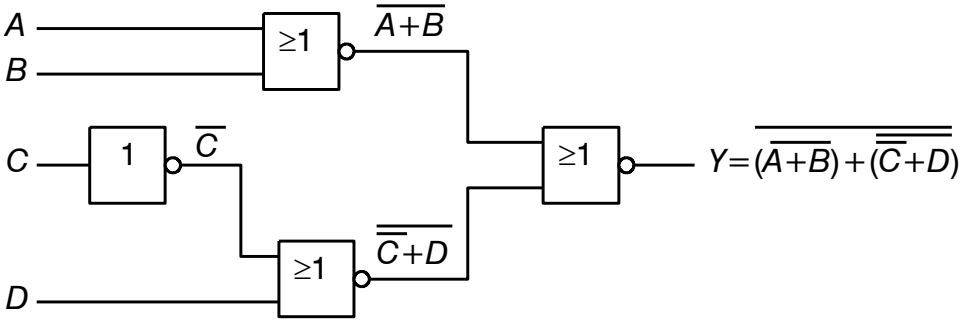
11.5.3 Enkele toepassingen van de stellingen van De Morgan

Figuur 11.5-5 geeft een eerste voorbeeld. Gebruik het rode recept en je vindt de uitkomst.



Figuur 11.5-5. EN- en OF-poort, samen gevolgd door EN-poort met toepassing van het rode recept.

Een tweede voorbeeld is Figuur 11.5-6.



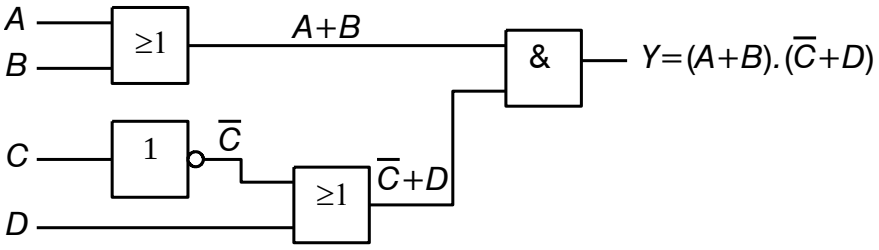
Figuur 11.5-6. Schema met een inverter en drie NOF-poorten. Uitwerking in de tekst.

De schakeling levert $Y = \overline{\overline{(A+B)} + (\overline{\overline{C}+D})}$

Wat binnen de haakjes staat, blijft ongemoeid. Dan verdwijnt de lange ontkenningsstreep (negatiestreep) en de twee negatiestrepen boven de uitdrukkingen tussen de haakjes. De OF ertussenin wordt een EN. Dat levert de vergelijking

$$Y = (A+B).(\overline{C}+D)$$

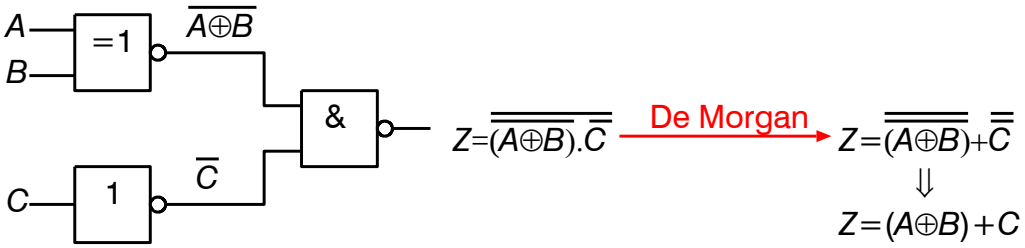
Het bijbehorende schema staat in Figuur 11.5-7.



Figuur 11.5-7. Het omgezette schema van Figuur 11.5-6

Ee staan evenveel poorten in, maar net als de vergelijking is het schema beter te volgen dan het oorspronkelijke.

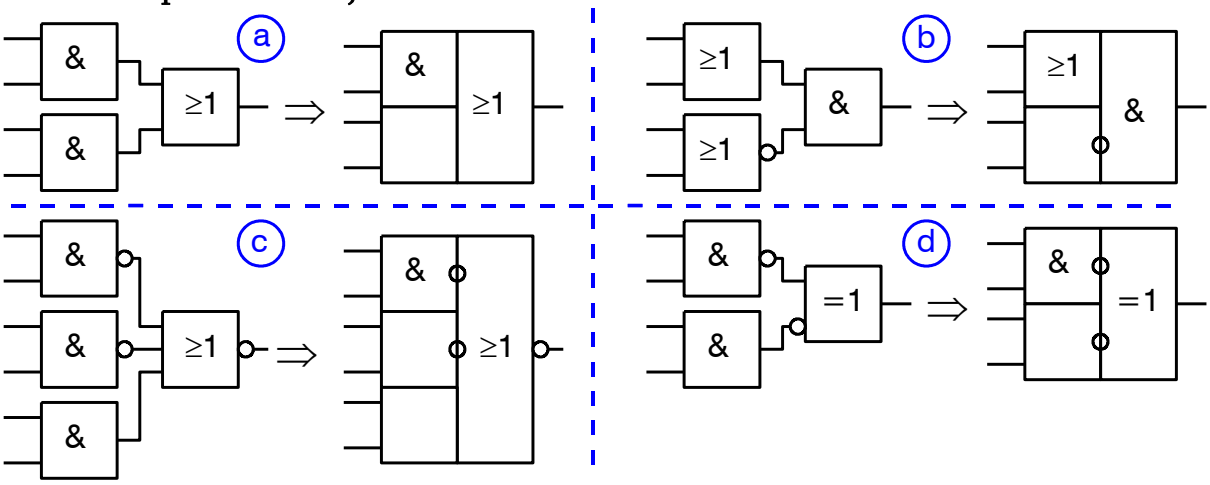
Een voorbeeld met een EXNOF-poort, vinden we in Figuur 11.5-8. Een EXOF of EXNOF behandel je vaak het gemakkelijkst als was het een enkele variabele.



Figuur 11.5-8. Zo behandel je een EXNOF via de stellingen van De Morgan. De EXOF-uitdrukking wordt in zijn geheel behandeld als was het een enkele variabele.

De EXOF is uit te schrijven als $A\oplus B = A\bar{B} + \bar{A}B$, maar dat heeft zelden nut.

11.5.4 Compacte tekenwijze



Figuur 11.5-9. Omzetting van "normale" naar compact getekende schema's. Links van elke pijl het "normale" schema, rechts ervan de compacte variant.

Schema's met IEC-blokken kunnen worden vereenvoudigd door ze tegen elkaar aan te tekenen. Dan vervallen lijnen die verbindingen tussen poorten voorstellen. Dit leidt vaak tot overzichtelijker schema's. Figuur 11.5-9 laat wat voorbeelden zien.

11.5.5 De optelschakeling

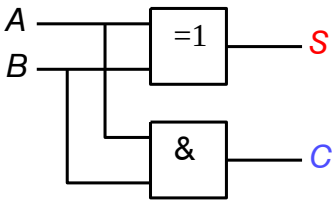
Met poorten kun je getallen optellen. Het hart van een optelschakeling is de EXOF-poort. De waarheidstabel is Tabel 11.5-1.

Tabel 11.5-1. Waarheidstabel voor de EXOF-poort (kopie van Tabel 11.3-5)

$Q = A.\bar{B} + \bar{A}.B = A \oplus B$		
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	0

Het optellen gaat goed tot en met de derde regel. In regel 4 is 1+1 binair 10. De 0 staat er netjes in, maar de 1 van "1 bewaren" is weg. In poortentaal heet die "carry", dat is "1 bewaren". Die 1 moet naar een volgende kolom die de EXOf-schakeling niet biedt.

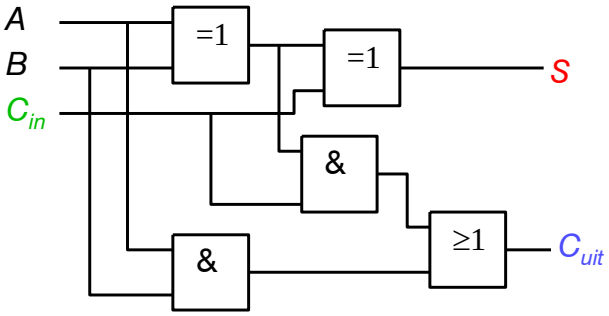
Die moet "buitenom" worden gerealiseerd met een poortschakeling die alleen een 1 oplevert als $A = 1$ èn $B = 1$. Dat doet een EN-poort. Een EXOF en een EN met twee ingangen zijn daarom genoeg. Figuur 11.5-10 laat het schema met waarheidstabel zien



A	B	C	S
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

Figuur 11.5-10. Half-adder met waarheidstabel voor de som S en de carry C.

Dit schema heet een *half adder*, vertaald: een halve opteller. Wat er half aan is, is dat de mogelijkheid ontbreekt, een eventuele carry van een vorige opteltrap toe te voegen. De meeste getallen bevatten nu eenmaal (veel) meer bits dan de schakeling van Figuur 11.5-10 aan kan. Met een zogenoemde *full adder* gaat het wel. Die omvat twee stuks EXOF, twee stuks EN en een OF om de uitkomsten van de twee EN-poorten samen te voegen. Figuur 11.5-11 toont de schakeling met waarheidstabel.



A	B	C_{in}	C_{uit}	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Figuur 11.5-11. Full adder met waarheidstabel. A_1 en A_2 zijn signaalinvoeren; C_{in} (groen) is de invoer voor de carry (transportbit), C_{uit} (blauw) de uitvoer van de carry (transportbit) en S (rood) is de uitvoer van de som.

In blokschema wordt het overzichtelijker (Figuur 11.5-12).

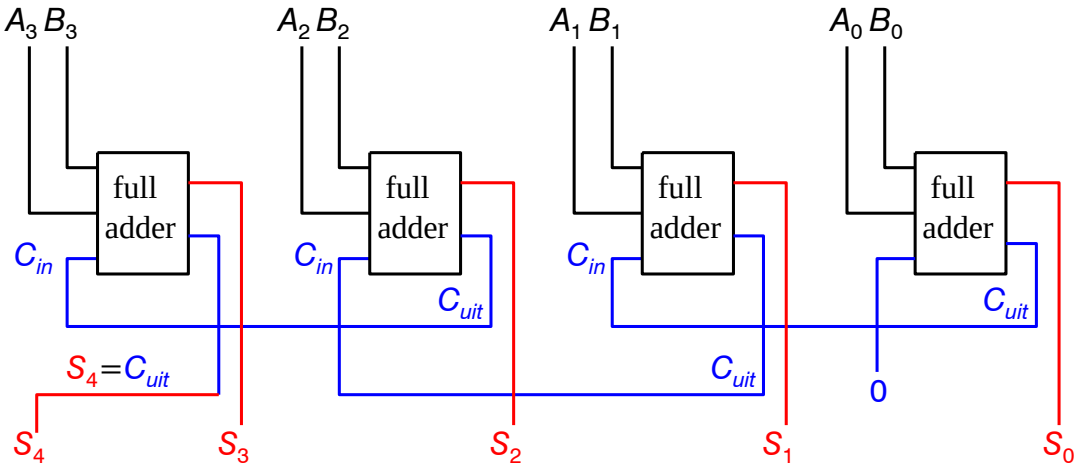
Figuur 11.5-12. Half adder en full adder in blokschema.

Hiermee hebben we een adder, optelschakeling, voor 2 getallen van 1 bit plus een carry van een voorgaand bitpaar.

Voor het optellen van twee getallen met een heleboel bits hebben we evenveel optellers nodig als er bits zijn. Daarin zit een volgordeprobleem. Normaal bekijken we alles, tekst en schema's, van links naar rechts. Bij het optellen van getallen werken we van rechts naar links. Bij decimale getallen doen we dat voor ons gevoel vanzelf al, omdat we dat ooit op school zo hebben geleerd. Bij binaire getallen gaat dat net zo, want een binair getal is

net zo ingedeeld als een decimaal getal. De kleinste waarde staat rechts, de grootste links. Het meest rechtse bit heet Least Significant Bit, afgekort LSB. Het meest linkse bit heet Most Significant Bit, MSB. Vaak zitten optellers in IC's met 4 stuks op één chip. De LSB's van beide getallen $A_0 \ldots A_3$ en $B_0 \ldots B_3$ komen binnen op de meest rechtse opteller.

Dat zou dus een half adder mogen zijn, maar in de praktijk wordt dit soort schakelingen ook voor grotere getallen gebruikt en daarom achter elkaar gezet. Dan is een half adder rechts onhandig, omdat in een grotere schakeling die adder kan worden voorafgegaan door nog een blok van vier (of meer). Figuur 11.5-13 laat een schakeling van vier full adders zien.

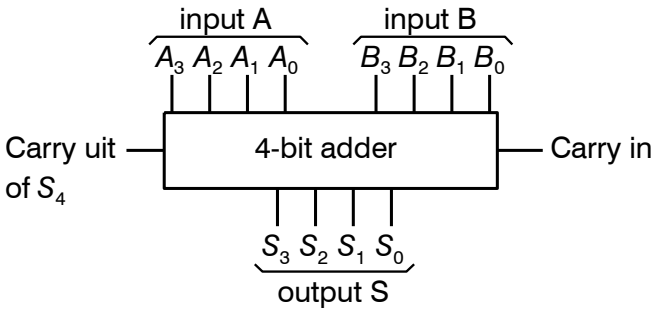


Figuur 11.5-13. Optelschakeling voor twee 4-bits getallen A en B. Sommen S in rood, carry-bits C_{in} en C_{uit} in blauw.

Als er aan het eind, links dus, een bit (carry) overblijft, kan dat ofwel naar een volgende optelschakeling, ofwel het is het vijfde bit van de uitkomst, S_4 dus. In vergelijkingvorm:

$$A_0 \ldots A_3 + B_0 \ldots B_3 = S_0 \ldots S_4$$

In blokvorm ziet het eruit als in Figuur 11.5-14.



Figuur 11.5-14. Schematische weergave van een 4-bits adder

11.6 Sequentiële logica: flipflops

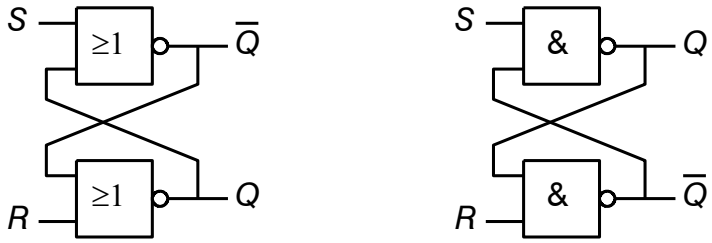
11.6.1 Inleiding

In schakelingen met poorten, combinatielogica, wordt de gang van zaken bepaald door de toestand op de ingangen van de betrokken poorten. Bij sequentiële logica worden de gebeurtenissen mede bepaald door vorige toestanden op de ingangen. Dan moeten vorige toestanden worden onthouden. Dat kan, mits de schakeling geheugen bevat.

Dat geheugen zit in zogenoemde flipflops. Dat zijn schakelingen die twee stabiele toestanden kennen. Ze hebben twee gelijke helften, die elkaars toestand bepalen. De ene helft geleidt, de andere helft staat gesperd. De sperring van de tweede helft zorgt dat de andere helft geleidt en andersom. Als we de ene toestand 1 noemen, is de andere 0 en omgekeerd. Omklappen van 1 naar 0 of omgekeerd gebeurt door middel van een blok- of pulsvormig signaal. Deze paragraaf gaat over soorten flipflops en hun eigenschappen.

11.6.2 De RS-flipflop

Flipflops kun je maken met NOF- en NEN-poorten (Figuur 11.6-1).



Figuur 11.6-1. RS-flipflops. Links: gebaseerd op NOF-poorten. Rechts: gebaseerd op NEN-poorten. Waarheidstabellen in Tabel 11.6-1.

De figuur toont twee RS-flipflops. Links één, gemaakt van twee NOF-poorten en rechts één van twee NEN-poorten. R en S staan voor resp. “reset” en “set”. De betekenis van die termen volgt iets verderop. Eerst de twee schema’s.

We beginnen met het linker schema (met de NOF-poorten). In rust zijn R en S beide 0. De uitgang van de bovenste NOF is verbonden met de overgebleven ingang van de onderste NOF-poort, de uitgang van de onderste NOF met de overgebleven ingang van de bovenste. Welke van de twee uitgangen is nu 1? Antwoord: als $Q = 1$, dan is $\bar{Q} = 0$ en omgekeerd. Beide poorten hebben elkaar in een soort houdgreep. Maken we $R = 1$, dan krijgen we $Q = 0$ en $\bar{Q} = 1$. Maken we $S = 1$, dan ontstaat $Q = 1$ en $\bar{Q} = 0$. Dat verklaart de namen R voor “Reset” en S voor “Set”.

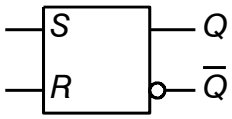
Bij een OF en een NOF reageert de uitgang niet op de toestand van een ingang als de andere ingang 1 is. Bij EN en NEN is het andersom: als één van de ingangen 0 is, reageert de uitgang niet op de toestand van de andere ingang. Dat vertaalt zich in de waarheidstabellen voor de schakelingen van Figuur 11.6-1 (Tabel 11.6-1)

Tabel 11.6-1. Waarheidstabellen van de flipflops van Figuur 11.6-1. Links: waarheidstabel voor de RS-flipflop van NOF-poorten. Rechts: idem voor die van NEN-poorten.

R	S	Q	Niet Q	Toestand
0	0	Q	$\bar{Q}$	onthoud
1	0	0	1	reset
0	1	1	0	set
1	1	0	0	"verboden"

R	S	Q	Niet Q	Toestand
1	1	Q	$\bar{Q}$	onthoud
0	1	0	1	reset
1	0	1	0	set
0	0	1	1	"verboden"

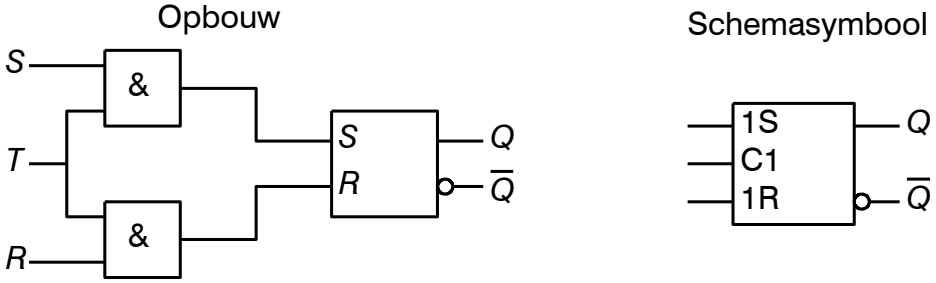
$R = 1$ en $S = 1$ zijn samen een "verboden" toestand voor de NOF-flipflop, want ze sturen de flipflop in een tegengestelde toestand: Q en $\bar{Q}$ moeten tegelijk 0 zijn. Dat kan niet en daardoor is de toestand onvoorspelbaar geworden. Dat is het laatste wat je in een digitale schakeling wilt. Voor de uit NEN's opgebouwde flipflop is de "verboden toestand" 0 op R en S tegelijk, wat zou moeten leiden tot de onmogelijke situatie van een 1 op Q èn op $\bar{Q}$. Het korte en meest gebruikte schemasymbool voor de RS-flipflop staat in Figuur 11.6-2.



Figuur 11.6-2. Schemasymbool voor een RS-flipflop.

11.6.3 De impulsgestuurde (geklokte) RS-flipflop

Een gewone RS-flipflop is gevoelig voor stoorpulsjes. Daar is wat aan te doen met de impulsgestuurde RS. Opbouw en schemasymbool staan in Figuur 11.6-3.

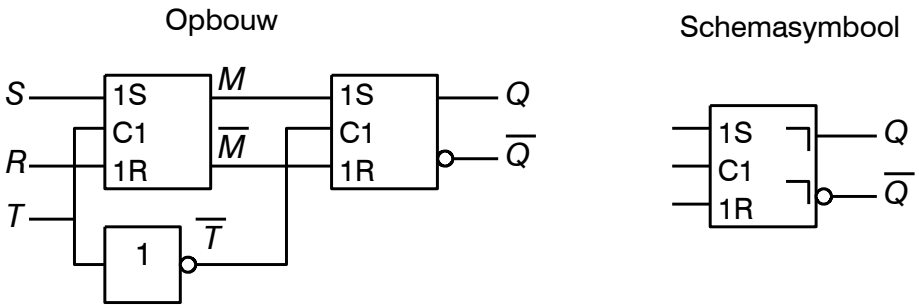


Figuur 11.6-3. Links: impulsgestuurde RS-flipflop die alleen kan schakelen als T=1. Rechts: schemasymbool.

De EN-poorten maken dat de flipflop alleen gevoelig is voor S en R als $T = 1$. Hoe korter de duur van de T-puls, des te minder gevoelig is de schakeling voor storingen. De waarheidstabel verschilt ten opzichte van Tabel 11.6-1 maar op één punt: een extra regel voor $T = 0$. Wat R en S ook zijn, bij $T = 0$ verandert de inhoud van de flipflop niet.

11.6.4 Het master-slave systeem

De master-slave flipflop bestaat uit twee flipflops, de meester en de slaaf. De meester bestuurt de slaaf. Het signaal komt binnen bij de meester, de slaaf neemt het over. Figuur 11.6-4 toont opbouw en schemasymbool.



Figuur 11.6-4. Master-slave RS-flipflop. Links: opbouw, rechts: schemasymbool.

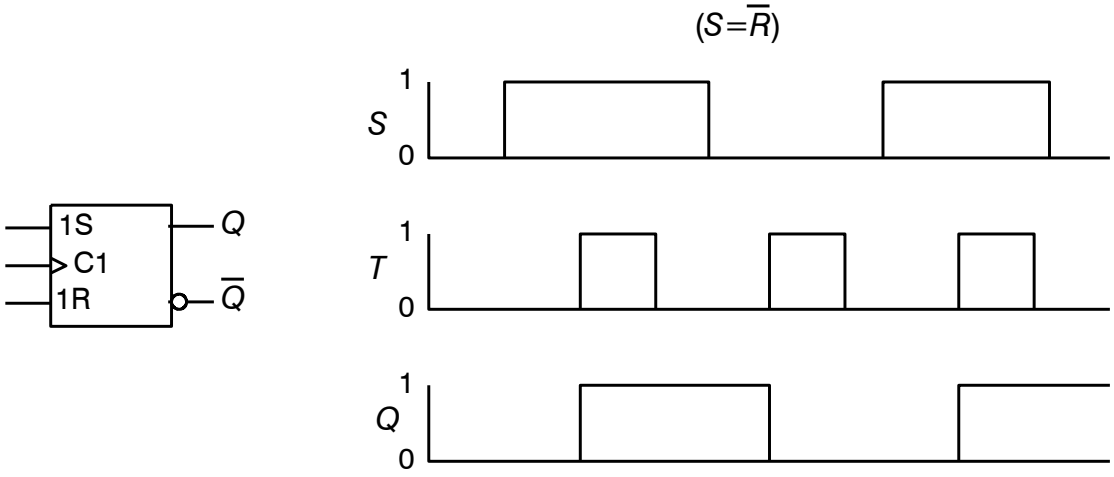
De eerste flipflop (master) neemt de toestand van R en S over als $T = 1$. De tweede flipflop (slave) ontvangt dan via de NIET-poort een $\bar{T}$.

De slave neemt de informatie op de uitgangen M en $\bar{M}$ over als $T = 0$, want dan is $\bar{T} = 1$.

In het schemasymbool is het master-slave principe te herkennen aan de hoekjes bij de uitgangen. Het systeem geeft een betere storingsvrijheid dan de geklokte enkele flipflop. Storingen die optreden gedurende de tijd dat $T = 1$ hebben geen invloed op de toestand van de uitgangen Q en $\bar{Q}$. De uitgangen nemen de inhoud van R en S over van het ogenblik direct vóór de overgang van $T = 1$ naar $T = 0$, mits $R = \bar{S}$.

11.6.5 De flankgestuurde RS-flipflop

Ook bij de master-slave kan er nog ongewenste inhoud in de flipflop terecht komen. De betrouwbaarste van allemaal is de flankgestuurde flipflop. Die reageert op de op- of neergaande flank van een klokpuls. Korter dan een flank kun je het niet hebben.

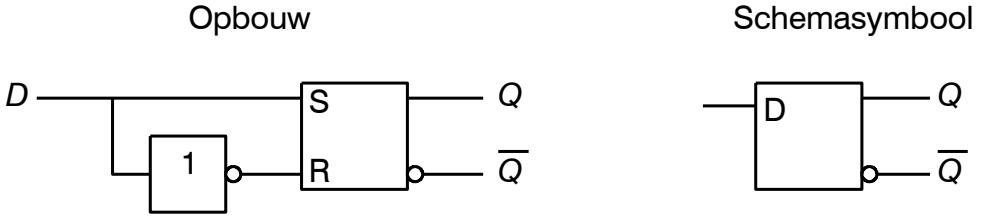


Figuur 11.6-5. Flankgestuurde RS-flipflop. Links: schemasymbool. Rechts: een voorbeeld van de gang van zaken. Een hoge positie is 1, een lage 0. Verondersteld is dat $S = \bar{R}$.

Schemasymbool en de gang van zaken bij het overnemen van R en S staan in Figuur 11.6-5. Q verandert op de voorflank van de klokpuls T. Wat er tussentijds op de R- en S-ingangen staat, doet er niet toe. Het driehoekje in het schemasymbool bij C1 geeft de flanksturing aan.

11.6.6 De D-flipflop

De D-flipflop heeft geen "verboden" toestand. Er is maar één D-ingang. Door de inverter geldt gegarandeerd dat $S = \bar{R}$ (Figuur 11.6-6).



Figuur 11.6-6. D-flipflop.

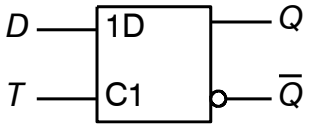
De waarheidstabel (Tabel 11.6-2) is op het eerste gezicht aangenaam eenvoudig.

Tabel 11.6-2. Waarheidstabel van de D-flipflop in Figuur 11.6-6.

D	Q	$\bar{Q}$	Toestand
0	0	1	Reset
1	1	0	Set

Helaas is de werking zo versimpeld dat de D-flipflop ongeschikt is om de toestand van D te onthouden. Hij is een doorgeefluik voor D met $\bar{D}$ in de vorm van $\bar{Q}$ als toegift.

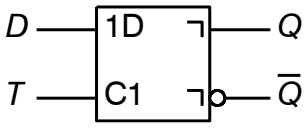
Maar in geklokte vorm wordt hij interessant. Q kan zich dan alleen aan D aanpassen tijdens een klokpuls. Dan onthoudt het ding tussen twee klokpulsen in wèl iets.



D	T	Q	$\bar{Q}$	Toestand
x	0	Q	$\bar{Q}$	Onthoud
0	1	0	1	Reset
1	1	1	0	Set

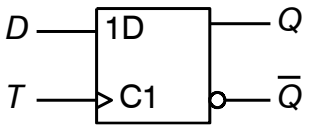
Figuur 11.6-7. Geklokte D-flipflop. Links: schemasymbool. Rechts: waarheidstabel.

De waarheidstabel in Figuur 11.6-7 laat het zien. Links zien we het schemasymbool. Met een master-slave-opbouw kan het ook. Figuur 11.6-8 laat het schemasymbool zien.



Figuur 11.6-8. Master-slave D-flipflop.

Het schemasymbool ziet er net zo uit als dat in Figuur 11.6-7, maar nu met de bekende hoekjes bij de uitgangen. Met een flankgestuurde flipflop kan het ook (Figuur 11.6-9)



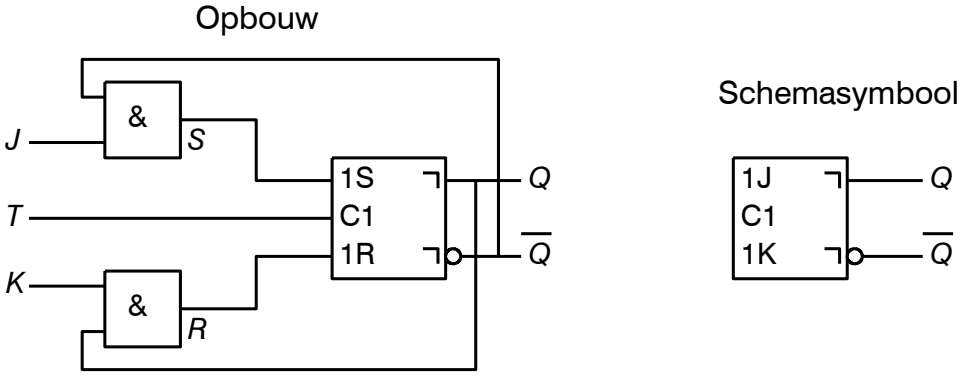
D	T	Q	$\bar{Q}$	Toestand
x	0	Q	$\bar{Q}$	Onthoud
x	1	Q	$\bar{Q}$	Onthoud
0	0→1	0	1	Reset
1	0→1	1	0	Set

Figuur 11.6-9. Flankgestuurde D-flipflop. Links: schemasymbool. Rechts: waarheidstabel.

Deze geklokte flipflop heet ook wel *bistable latch*.

11.6.7 De JK-flipflop

Ook de master-slave JK-flipflop is een flipflopschakeling zonder verboden toestand. De opbouw en het schemasymbool zijn weergegeven in Figuur 11.6-10.



Figuur 11.6-10. De JK-flipflop. Links de opbouw, rechts het schemasymbool.

De schakeling bestaat uit een geklokte master-slave RS-flipflop. De R- en de S-ingang worden voorafgegaan door een EN-poort. Die voor de S-ingang is met één ingang verbonden met de $\bar{Q}$-uitgang, de andere is de J-ingang. Bij de R-ingang zien we hetzelfde, maar is de EN met één ingang verbonden met de Q-uitgang en is de andere de K-ingang. De klokingang is T. Tabel 11.6-3 is de waarheidstabel. De vierde regel met $J = 1$ en $K = 1$ maakt een frequentiedeler: de klokfrequentie wordt door 2 gedeeld (Foto 11.6-1).

Tabel 11.6-3. Waarheidstabel van een J-K flipflop.

J	K	T	Q	Niet Q	Toestand
0	0	010	Q	$\bar{Q}$	Onthoud
0	1	010	0	1	Reset
1	0	010	1	0	Set
1	1	010	$Q \rightarrow \bar{Q}$	$\bar{Q} \rightarrow Q$	Omslag

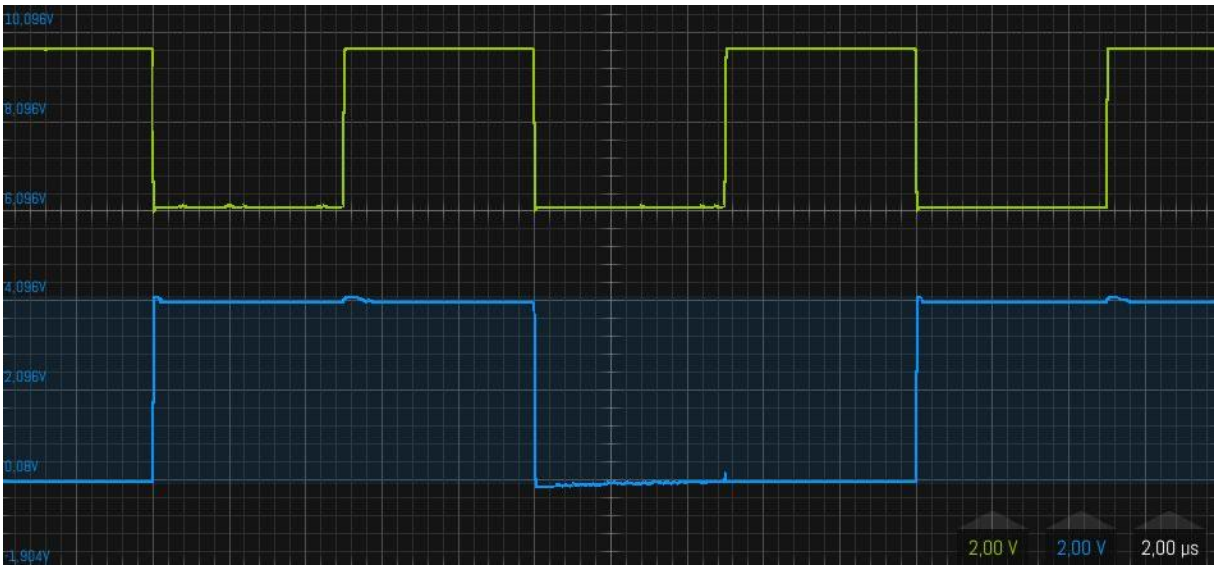
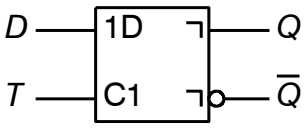


Foto 11.6-1. Frequentiedeling met een JK-flipflop bij J=1 en K=1. Geel: klok. Blauw: Spanning op uitgang Q.

11.6.8 De master-slave D-flipflop als frequentiedeler

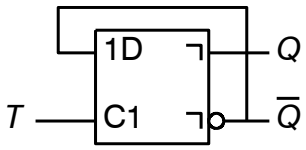
Als de $\bar{Q}$-uitgang wordt verbonden met de D-ingang, ontstaat ook een frequentiedeler.



D	T	Q	$\bar{Q}$	Toestand
x	0	Q	$\bar{Q}$	Onthoud
0	1	0	1	Reset
1	1	1	0	Set

Figuur 11.6-11. De master-slave D-flipflop van Figuur 11.6-8 (links) met rechts de bijbehorende waarheidstabel.

En nu de als frequentiedeler geschakelde M-S D-flipflop (Figuur 11.6-12):



Figuur 11.6-12. Als frequentiedeler geschakelde Master-Slave D-flipflop.

Daarin is $D = \bar{Q}$. Bij elke klokpuls is er een set of een reset (zie waarheidstabel in Figuur 11.6-11). Ook dat leidt tot een frequentiedeler, vergelijkbaar met het beeld op Foto 11.6-1.

11.6.9 De flipflops samengevat

Het rijtje flipflops dat we hebben gezien is:

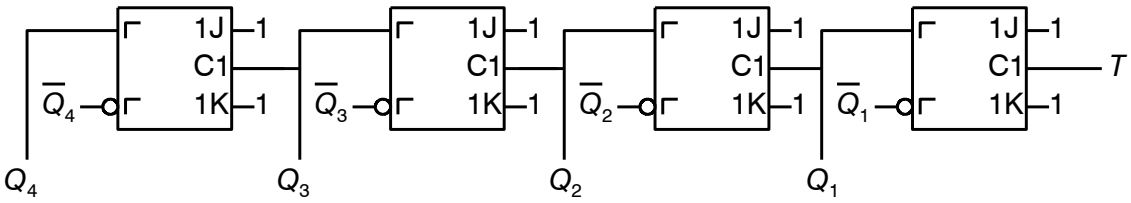
1. De RS-flipflop in zijn simpelste vorm: geen klok, alleen een R- en een S-ingang. En één "verboden" toestand.
2. De geklokte RS-flipflop. Minder storingsgevoelig, wel een "verboden" toestand.

3. De master-slave (M-S) RS-flipflop. Nog minder storingsgevoelig, nog steeds een “verboden” toestand.
4. De flankgestuurde RS-flipflop, bijna immuun voor storingen, wel een “verboden” toestand
5. De D-flipflop; in zijn eenvoudigste vorm een vrij nutteloos ding, maar in geklokte vorm geschikt als geheugenelement en in M-S-vorm bruikbaar als frequentiedeler. Geen “verboden” toestand.
6. De M-S JK-flipflop: 2 ingangen zonder verboden toestand, geschikt als geheugenelement èn als frequentiedeler. De veelzijdigste en ingewikkeldste.

11.7 Frequentiedelers, tellers en registers

11.7.1 Frequentiedelers en -tellers

Frequenties delen door elke gewenste hele macht van 2 kan met een rij M-S JK-flipflops. Het aantal is even groot als de macht van 2 waardoor wordt gedeeld. Voor een 16-deler bijvoorbeeld, dus delen door 2^4, zijn 4 delers nodig (Figuur 11.7-1).



Figuur 11.7-1. Asynchrone 4-bits frequentiedeler, bestaand uit 4 JK-flipflops.

Q_1 levert de klokfrequentie T gedeeld door 2^1=2, Q_2 is T gedeeld door 2^2 =4 en zo verder.

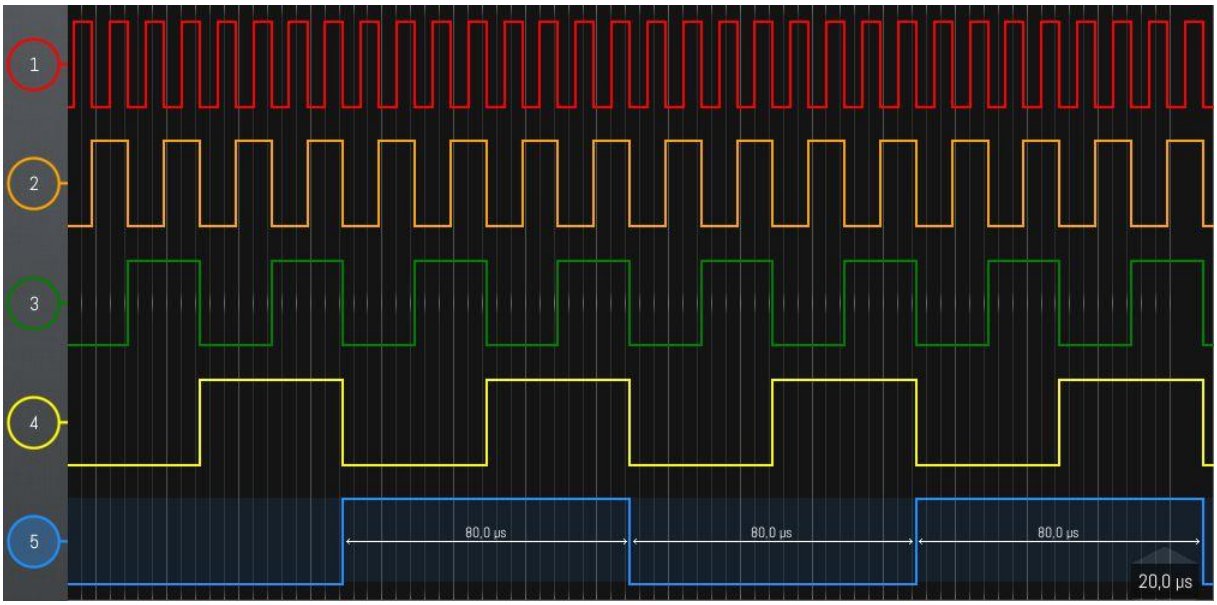


Foto 11.7-1.Golfvormen van de schakeling van Figuur 11.7-1. Spoor 1 (rood) is de klokpuls op ingang T. Spoor 2 (oranje) is Q_1; spoor 3 (groen) is Q_2; spoor 4 (geel) is Q_3 en spoor 5 (blauw) is Q_4.

Foto 11.7-1 laat de klokpuls *T* zien (spoor 1), de klokfrequentie gedeeld door 2 (spoor 2), door 4 (spoor 3), door 8 (spoor 4) en door 16 (spoor 5).

De deler is asynchroon, dat wil zeggen dat de afzonderlijke delers pas reageren als ze de signaalsprong van de voorgaande schakeling binnen krijgen. Omdat elke deler een kleine vertraging veroorzaakt, reageert de laatste deler iets later dan de eerste. Om een synchrone deler te maken waarin de afzonderlijke deeltrappen wel gelijktijdig reageren, is aanvullende elektronica nodig die (ver) buiten de exameneisen valt.

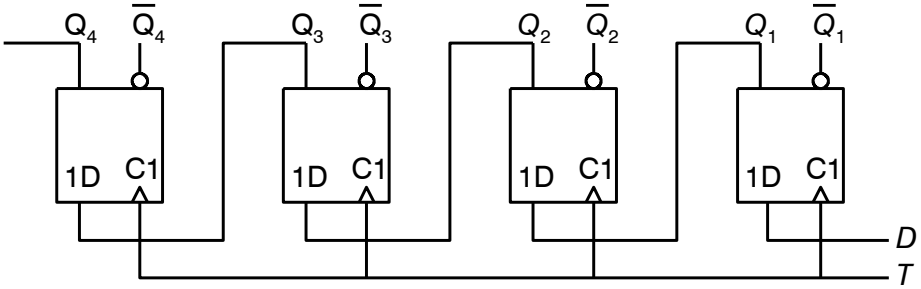
De schakeling van Figuur 11.7-1 is ook een teller. Q_1 t/m Q_4 vormen een binair getal waarvan de laagste waarde 0 is en de hoogste waarde 15.

11.7.2 Delers en tellers voor andere getallen dan machten van 2

Voor ons zijn binaire getallen lastige dingen om mee te werken. Vaak wordt dan de BCD-code gebruikt. BCD is een afkorting van *binary coded decimal*, tweetallig gecodeerd tientallig. Dat kan door de teller zichzelf na 9 terug te laten zetten op 0 en een volgende teller een pulsje bij te laten tellen. Dan krijg je het decimale getal 10 (of 20 of 30, enz.).

11.7.3 Schuifregisters

Schuifregisters zijn naaste familie van tellers. Hun uitgangstoestand is niet alleen afhankelijk van aantallen klokpulsen, maar ook van data die ze op een data-ingang krijgen aangeleverd. De databits worden bij elke klokpuls een plekje doorgeschoven. Datadoorgifte vraagt om een flankgestuurde flipflop. De eenvoudige reden is dat bij een pulsgestuurd register data op de data-ingang voldoende tijd heeft om door te rollen naar de laatste flipflop in de rij. Bij flanksturing is die tijd te kort en komt een databit niet verder dan 1 flipflop. Figuur 11.7-2 toont de opzet van een 4-bits schuifregister.



Figuur 11.7-2. De opbouw van een 4-bits schuifregister met D-flipflops.

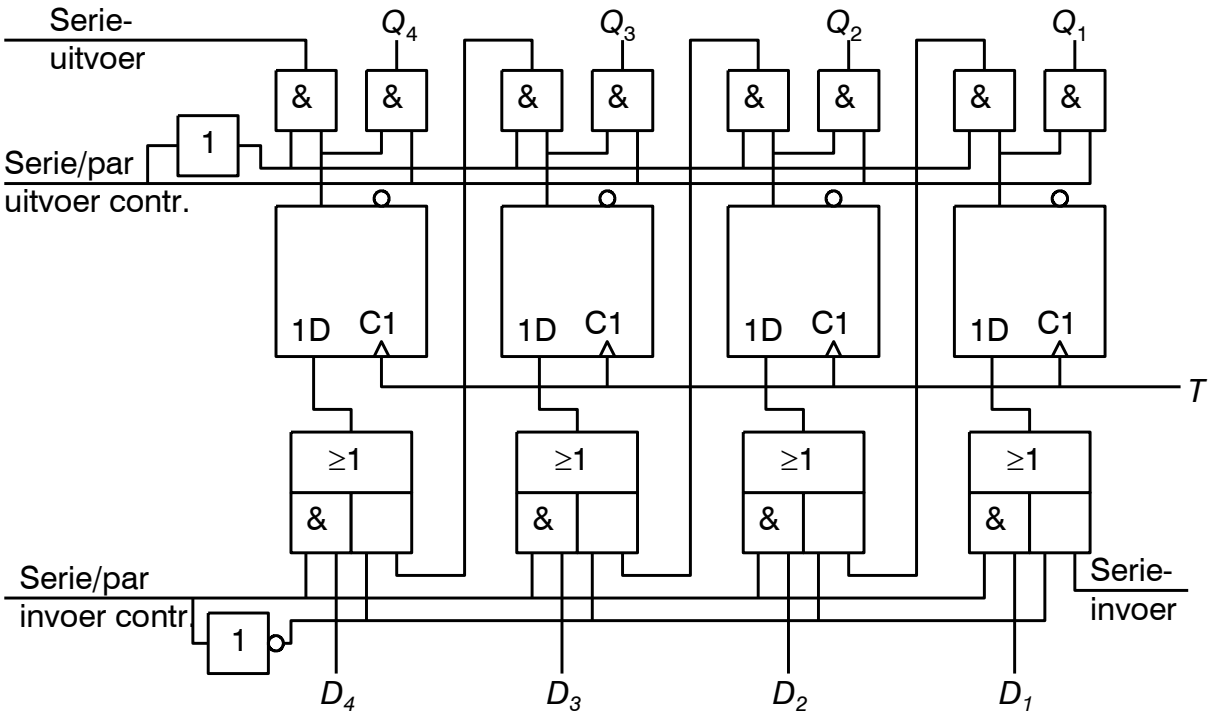
Het kan ook met master-slave flipflops. Dan wordt bij elke flipflop de toestand op de ingang, zoals die was bij het optreden van de voorflank van de klokpuls, pas bij het optreden van de achterflank doorgegeven naar de uitgang van de flipflop. Zo is bij het opnemen van een nieuw bit door de slaaf de doorgifte naar de volgende flipflop alweer geblokkeerd.

Deze manier van bit-voor-bit doorgeven heet *serieel*. Je kunt de toestand van een register ook *parallel* uitlezen, dat is alle bits, $Q_1 \ldots Q_n$ of $Q_0 \ldots Q_{n-1}$, net hoe je n bits wilt

nummeren, in één keer uitlezen in een schakeling die daarvoor geschikt is. Veel registers kunnen ook parallel worden voorzien van inhoud, waarna eventueel de inhoud wordt doorgeklokt als een serie bits. Het eerste heet serie-parallel conversie (omzetting), het tweede parallel-serie conversie.

De parallelle uitvoer kan bijvoorbeeld worden doorgegeven naar een optelschakeling. De inhoud één plaats vooruit klokken is vermenigvuldigen met 2, één plaats terug delen door 2.

Een voorbeeld van zo'n register zien we ter afsluiting in Figuur 11.7-3, Niet uit het hoofd leren, wel (ongeveer) snappen!



Figuur 11.7-3. 4-bits serie-parallel- en parallel-serie-omzetter, deels getekend met de poorten in compacte vorm.