DIGTEC/2021-2022

Jesse op den Brouw

DIGTEC

Schakelalgebra

Schakelalgebra

- Schakelalgebra is de wiskunde waarmee digitale schakelingen ontworpen kunnen worden.
- Het is gebaseerd op de Boole-algebra (Boole, 1854) en is dus zo’n 170 jaar oud.
- Boole onderzocht de wiskunde van verzamelingen.
- In 1937 toonde Shannon aan dat de *tweewaardige* Boole-algebra gebruikt kon worden bij het analyseren en ontwerpen van contactschakelingen (relais).

Schakelalgebra

- De schakelalgebra bestaat uit:
- Variabelen: a, Z, X, open1, ...
- Constanten: 0, 1
- Operatoren
  - = (is gelijk aan)
  - . (AND)
  - + (OR),
  - ‾ (NOT, *overbar*)
- Haakjes: ( ) of { } of [ ]

Schakelalgebra

De schakelalgebra is gebaseerd op de volgende rekenregels voor constanten:

$0 \cdot 0 = 0$	$1 + 1 = 1$
$0 \cdot 1 = 0$	$1 + 0 = 1$
$1 \cdot 0 = 0$	$0 + 1 = 1$
$1 \cdot 1 = 1$	$0 + 0 = 0$

Daarnaast geldt:

$x = 0$ als $x \neq 1$ $\qquad$ $x = 1$ als $x \neq 0$

Schakelalgebra

- Hoe wordt de volgende functie uitgewerkt?

$$s = a \cdot (b + \overline{c})$$

- Prioriteitsvolgorde:

Haakjes
NOT
AND
OR

Schakelalgebra

- Dus:

$$s = a \cdot (b + \overline{c})$$

- Eerst $\overline{c}$ uitwerken
- dan *b* met de uitkomst van boven uitwerken $(b + \overline{c})$
- daarna *a* met de uitkomst van boven uitwerken $(a \cdot (b + \overline{c}))$

- Gegeven dat a=1, b=0 en c=0 dan is de uitkomst

$$\overline{c} = 1$$

$$b + \overline{c} = 0 + 1 = 1$$

$$a \cdot (b + \overline{c}) = 1 \cdot (1) = 1 \cdot 1 = 1$$

Schakelalgebra

- Om de schakelalgebra enige meerwaarde te geven zijn er nog al wat *theorema's*:

Commutatieve wetten

$$x + y = y + x$$

$$x \cdot y = y \cdot x$$

Associatieve wetten

$$(x + y) + z = x + (y + z)$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

Distributieve wetten

$$(x + y) \cdot (x + z) = x + y \cdot z$$

$$x \cdot y + x \cdot z = x \cdot (y + z)$$

Schakelalgebra

Gelijkheidswetten

$$x + x = x$$
$$x \cdot x = x$$

Negatiewetten

$$x + \overline{x} = 1$$
$$x \cdot \overline{x} = 0$$
$$\overline{\overline{x}} = x$$

Moduluswetten

$$x + 0 = x \qquad x + 1 = 1$$
$$x \cdot 1 = x \qquad x \cdot 0 = 0$$

Schakelalgebra

Absorptiewetten

$$x + x \cdot y = x$$

$$x \cdot (x + y) = x$$

$$x + \overline{x} \cdot y = x + y$$

$$x \cdot (\overline{x} + y) = x \cdot y$$

$$x \cdot y + \overline{x} \cdot z + y \cdot z = x \cdot y + \overline{x} \cdot z$$

$$(x + y) \cdot (\overline{x} + z) \cdot (y + z) = (x + y) \cdot (\overline{x} + z)$$

Schakelalgebra

De Morgan

$$\overline{x+y} = \overline{x} \cdot \overline{y}$$

$$\overline{x \cdot y} = \overline{x} + \overline{y}$$

Substitutie

$$a \cdot b + c \cdot d \rightarrow x + y = \overline{\overline{x} \cdot \overline{y}} \rightarrow \overline{\overline{a \cdot b} \cdot \overline{c \cdot d}}$$

Waarheidstabel

- De meest basale weergave van een logische functie is de waarheidstabel.
- Gegeven is de algemene opbouw van een waarheidstabel, in dit geval voor drie variabelen.
- De rijen zijn volgens de normale binaire telcode oplopend genummerd.
- De variabelen f_0 t/m f_7 zijn de bijbehorende functiewaarden.

x	y	z	s
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

Waarheidstabel

- Elke regel levert een bijdrage aan de functie.
- De eerste regel leest als volgt:

$$s = f_0 \text{ als } x = 0, y = 0, z = 0$$

- De bijdrage van deze regel aan de functie is:

$$\overline{x} \cdot \overline{y} \cdot \overline{z} \cdot f_0$$

- De overige regels op vergelijkbare wijze.

x	y	z	s
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

Mintermen

- De functie kan geschreven worden als:

$$s = \overline{x} \cdot \overline{y} \cdot \overline{z} \cdot f_0 + \overline{x} \cdot \overline{y} \cdot z \cdot f_1 + \overline{x} \cdot y \cdot \overline{z} \cdot f_2 + \overline{x} \cdot y \cdot z \cdot f_3 +$$
$$x \cdot \overline{y} \cdot \overline{z} \cdot f_4 + x \cdot \overline{y} \cdot z \cdot f_5 + x \cdot y \cdot \overline{z} \cdot f_6 + x \cdot y \cdot z \cdot f_7$$

- De termen $\overline{x} \cdot \overline{y} \cdot \overline{z}$ t/m $x \cdot y \cdot z$ worden *mintermen* genoemd.

- Mintermen zijn producttermen waarin alle variabelen of een inverse (maar niet beide tegelijkertijd) voorkomen.

Som van mintermen

- Mintermen worden doorgaans afgekort door een *m* met een index.
- Zo is de minterm die hoort bij $\overline{x} \cdot \overline{y} \cdot \overline{z}$ gelijk aan m_0 en $x \cdot y \cdot z$ gelijk aan m_7.
- De functie kan dus ook geschreven worden als:

$$s = m_0 \cdot f_0 + m_1 \cdot f_1 + m_2 \cdot f_2 + m_3 \cdot f_3 + m_4 \cdot f_4 + m_5 \cdot f_5 + m_6 \cdot f_6 + m_7 \cdot f_7$$

- De vorm van deze functie wordt de *som van mintermen* genoemd. Dit is één van de twee *standaardvormen.*

Waarheidstabel

- Gegeven is de waarheidstabel rechts.
- De functie kan geschreven worden als:

$$s = m_0 \cdot 0 + m_1 \cdot 1 + \ldots + m_6 \cdot 1 + m_7 \cdot 1$$

- Of gewoon:

$$s = m_1 + m_3 + m_5 + m_6 + m_7$$

- Of nog sneller:

$$s = \sum m(1,3,5,6,7)$$

x	y	z	s
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Som van producten

- In veel gevallen kan een som-van-mintermen-functie vereenvoudigd worden.

- Door bewerking van de functie worden variabelen en producttermen geëliminieerd.

- Een voorbeeld: $s = \overline{a} \cdot \overline{b} + a \cdot c + b \cdot \overline{c}$

- Dit wordt een *som van producten* (SOP) genoemd.

- Dit levert een AND-OR-schakeling op.

Don’t cares

- Het kan zijn dat bepaalde functiewaarden niet gespecificeerd zijn. Zo’n functiewaarde heet *don’t care*.
- Don’t cares komen voort uit het feit dat bepaalde uitgangswaarden niet interessant zijn.
- Don’t cares komen voort uit het feit dat bepaalde ingangscombinaties nooit voorkomen (can’t happen).
- Don’t cares worden weergegeven door een – (minus). Andere representaties: x, d

Don’t cares

- Gegeven nevenstaande waarheidstabel.
- De functie wordt als volgt genoteerd:

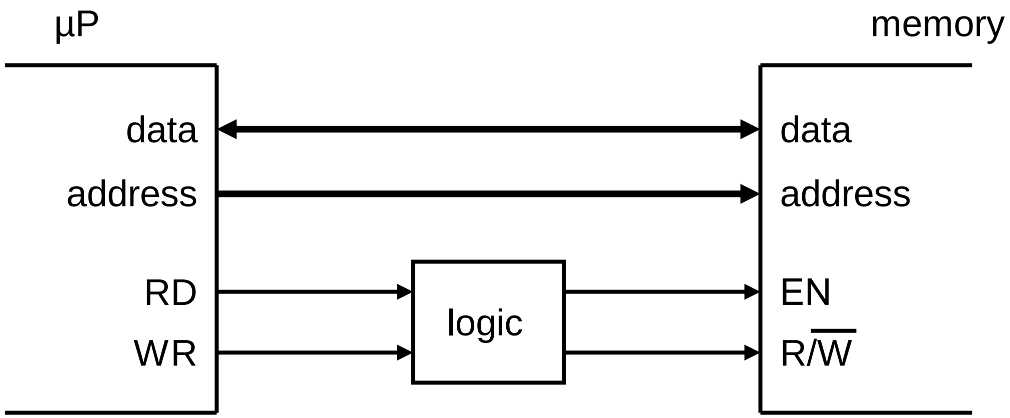
$$s = \sum m(2,3) + d(1)$$

- Don’t cares bestaan alleen bij de specificatie van een functie, niet bij de realisatie van een functie.
- Bij realisatie worden ze omgezet in een 0 of 1.

x	y	s
0	0	0
0	1	-
1	0	1
1	1	1

Don't cares – voorbeeld

- Een microprocessorschakeling met geheugen.



- Microprocessor: lezen -> RD=1, schrijven -> WR=1
- Geheugen: actief bij EN=1, R/$\overline{W}$=1 -> lezen, R/$\overline{W}$=0 -> schrijven
- Ontwerp de conversielogica

Don’t cares – voorbeeld

- Waarheidstabel microprocessor <-> geheugen:

RD	WR	EN	R/$\overline{W}$
0	0	0	-
0	1	1	0
1	0	1	1
1	1	-	-

waarde niet van belang

- Functie EN = RD + RW
- Functie R/$\overline{W}$ = RD

Vereenvoudigen

- Het is mogelijk om met behulp van de schakelalgebra functies te bewerken.
- Meestal wordt gezocht naar de eenvoudigste vorm, de vorm die het minste aantal operaties en variabelen bevat.
- Maar het kan ook zijn de vorm wordt gezocht die het eenvoudigst *afbeeldbaar* is op poorten of transistoren op IC’s.
- Voor alsnog wordt de eerste definitie gehanteerd.

Vereenvoudigen

- Als voorbeeld wordt de volgende functie vereenvoudigd:

$s = x \cdot \overline{y} + x \cdot y + \overline{x} \cdot y$	De functie
$s = \overline{x} \cdot y + x \cdot \overline{y} + x \cdot y$	Commutatieve wet
$s = \overline{x} \cdot y + x \cdot (\overline{y} + y)$	Distributieve wet
$s = \overline{x} \cdot y + x \cdot 1$	Negatiewet
$s = \overline{x} \cdot y + x$	Moduluswet
$s = x + \overline{x} \cdot y$	Commutatieve wet
$s = x + y$	Absorbtiewet

Vereenvoudigen

- Vereenvoudigen van schakelfuncties met algebra is lastig.
- Er is gedegen kennis en ervaring nodig om alle regels en wetten toe te passen.
- Andere methoden:
  - Karnaughdiagrammen
  - Tabellenmethode

let’s change