Academie voor Technology, Innovation & Society Delft
Academie voor ICT & Media

Inleiding Digitale Techniek

Week 5 – Two's complement representatie, BCD-optellen

Jesse op den Brouw

INLDIG/2018-2019

DE HAAGSE HOGESCHOOL

Introductie negatieve getallen

- Tot nu toe zijn alleen positieve getallen (en nul) behandeld.
- Getallen kunnen echter ook negatief zijn, bijvoorbeeld om een tekort aan te geven (banksaldo) of een richting (negatieve stroom).
- In het alledaags gebruik wordt de teken-en-grootte-representatie (*signed magnitude*) gebruikt.
- Er is een extra teken (het ‘–’-teken) nodig in het bekende decimale talstelsel.

Signed magnitude

- In de signed magnitude representatie bestaat een getal uit een *grootte* en een *teken* dat aangeeft of het een positief of negatief getal is.
- Dus +13 en –13 zijn even groot (magnitude) maar tegengesteld.
- Er zijn twee weergaven van nul, +0 en –0, beide met dezelfde waarde.
- Een getal zonder ‘+’ wordt als positief beschouwd.

Signed magnitude

- Het *optellen* van twee signed magnitude getallen kost enige moeite:
- Als de twee getallen hetzelfde teken hebben, moeten de grootten worden opgeteld en krijgt het antwoord hetzelfde teken.
- Als de twee getallen verschillende tekens hebben, moet de kleinste grootte worden afgetrokken van de grootste grootte, en het antwoord krijgt het teken van de grootste grootte.
- Al met al een hoop ‘als’, ‘optellen’, ‘aftrekken’, ‘vergelijken’.

Signed magnitude

- Hieronder vier voorbeelden:

+321	–456	+321	+231
+244	–123	–132	–342
+565	–579	+189	↓
			+342
			–231
			+111
			↓
			–111

- Er zijn nog meer voorbeelden te bedenken.

DE HAAGSE HOGESCHOOL

Complement representatie

- Al dit ‘als’, ‘optellen’, ‘aftrekken’, ‘vergelijken’ levert een grote hoeveelheid digitale schakelingen op.

- Eén pluspunt: als bekend is hoe een signed magnitude opteller gebouwd moet worden, dan is een signed magnitude aftrekker erg simpel.

- Slimmer (beter?) is om gebruik te maken van een *complement* (of *radix-complement)* representatie.

- Complement representaties zijn gebaseerd op *modulair rekenen.* Dat wordt eerst besproken, daarna wordt de 2’s complement representatie geintroduceerd.

Modulair rekenen

- In digitale systemen worden getallen opgeslagen in een eindig aantal bits. Er zijn dus maar een eindig aantal getallen weer te geven.
- Zo zijn met 4 bits de getallen 0 t/m $2^4 - 1$ (0 t/m 15) weer te geven. Er zijn 2^4 (16) combinaties (verschillende getallen) mogelijk.
- Een optelling van twee getallen van 4 bits levert een resultaat van 5 bits, maar het 5^e bit wordt genegeerd.
- Alle berekeningen zijn modulo 2^4. Er zijn 2^4 combinaties mogelijk.

Modulair rekenen

- Voorbeelden met 4 bits:

$(0 + 0) \mod 2^4 = 0 \mod 2^4 = 0 \mod 2^4$
$(10 + 5) \mod 2^4 = 15 \mod 2^4 = 15 \mod 2^4$

$(13 + 9) \mod 2^4 = 22 \mod 2^4 = 6 \mod 2^4$
$(15 + 15) \mod 2^4 = 30 \mod 2^4 = 14 \mod 2^4$

$(1 + 15) \mod 2^4 = 16 \mod 2^4 = 0 \mod 2^4$
$(2 + 15) \mod 2^4 = 17 \mod 2^4 = 1 \mod 2^4$
$(3 + 15) \mod 2^4 = 18 \mod 2^4 = 2 \mod 2^4$

- Merk op dat alle getallen positief zijn.

Modulair rekenen

- We kunnen de optelling met n bits als volgt beschrijven:

  $A + B \mod 2^n$

- We mogen een willekeurig aantal keer 2^n bij optellen of aftrekken:

  $$A + B \mod 2^n = A + B + 2^n \mod 2^n = A + B + 2^n + 2^n \mod 2^n$$
  $$= A + B - 2^n \mod 2^n = A + B - 2^n - 2^n \mod 2^n$$

- Of:

  $A + B \mod 2^n = A + B + k \cdot 2^n \mod 2^n$ (met $k \in Z$)

DE HAAGSE HOGESCHOOL

Modulair rekenen

- We kunnen gebruik maken van de mooie eigenschap:

  $A + B \mod 2^n = A + (2^n + B) \mod 2^n$

- We kunnen B dus voorstellen als $(2^n + B)$.
  - dit alles mod 2^n natuurlijk
  - een mooi woord voor voorstellen is *representeren*.
  - B kan niet willekeurig groot zijn

- Dit geldt ook als B negatief is!

- We kunnen een negatief getal voorstellen door de positieve variant af te trekken van 2^n.

Modulair rekenen

- Als voorbeeld een paar negatieve getallen met 4 bits:

$$
\begin{aligned}
-1 \bmod 2^4 &= (2^4 - 1) \bmod 2^4 = (16 - 1) \bmod 2^4 = 15 \bmod 2^4 \\
-2 \bmod 2^4 &= (2^4 - 2) \bmod 2^4 = (16 - 2) \bmod 2^4 = 14 \bmod 2^4 \\
-3 \bmod 2^4 &= (2^4 - 3) \bmod 2^4 = (16 - 3) \bmod 2^4 = 13 \bmod 2^4 \\
-4 \bmod 2^4 &= (2^4 - 4) \bmod 2^4 = (16 - 4) \bmod 2^4 = 12 \bmod 2^4 \\
-5 \bmod 2^4 &= (2^4 - 5) \bmod 2^4 = (16 - 5) \bmod 2^4 = 11 \bmod 2^4 \\
-6 \bmod 2^4 &= (2^4 - 6) \bmod 2^4 = (16 - 6) \bmod 2^4 = 10 \bmod 2^4 \\
-7 \bmod 2^4 &= (2^4 - 7) \bmod 2^4 = (16 - 7) \bmod 2^4 = 9 \bmod 2^4 \\
-8 \bmod 2^4 &= (2^4 - 8) \bmod 2^4 = (16 - 8) \bmod 2^4 = 8 \bmod 2^4
\end{aligned}
$$

(hmmm, –8 is gelijk aan 8...., en hoe zit het dan met –9?)

Modulair rekenen

- Als voorbeeld een paar positieve getallen met 4 bits:

$1 \mod 2^4 = 1 \mod 2^4$
$2 \mod 2^4 = 2 \mod 2^4$
$3 \mod 2^4 = 3 \mod 2^4$
$4 \mod 2^4 = 4 \mod 2^4$
$5 \mod 2^4 = 5 \mod 2^4$
$6 \mod 2^4 = 6 \mod 2^4$
$7 \mod 2^4 = 7 \mod 2^4$

(hmmm, hoe zit het dan met 8?)

Modulair rekenen

- Merk op dat in de vorige slides alle representaties positieve getallen zijn, er zijn geen negatieve getallen:

$$-1 \rightarrow 15 \qquad -2 \rightarrow 14 \qquad -3 \rightarrow 13 \qquad \ldots$$
$$1 \rightarrow 1 \qquad 2 \rightarrow 2 \qquad 3 \rightarrow 3 \qquad \ldots$$

- Aftrekken kan nu makkelijk worden omgezet in optellen:

$$A - B = A + (2^4 - B) \qquad \text{(dit alles mod } 2^4\text{)}$$

- Dus: alleen positieve getallen en optellen!

- Merk op dat het bereik van de getallen begrensd is.

DE HAAGSE HOGESCHOOL

2’s complement

- In digitale systemen wordt de 2’s complement representatie gebruikt.
- Hierbij wordt een getal negatief voorgesteld door het af te trekken van een macht van 2, bijvoorbeeld 2^4 (16, 4-bits getal).
- Als voorbeeld +5 → – 5

$$\begin{array}{r} 16 \\ 5 \\ \hline 11 \end{array} - \qquad \begin{array}{r} 10000_2 \\ 0101_2 \\ \hline 1011_2 \end{array} - \qquad \begin{array}{r} 2^4 \\ +5 \\ \hline 2^4 - 5 \end{array} -$$

- Het binaire getal 1011_2 is de *representatie* van -5_{10}. Merk op dat de getallen *modulo* 2^4 zijn!

DE HAAGSE HOGESCHOOL

2’s complement

- Nog een tweetal voorbeelden met 4 bits en 8 bits representatie:

16	10000_2	2^4		256	100000000_2	2^8	
1	0001_2	+1		97	01100001_2	+97	
15	1111_2	$2^4 - 1$		159	10011111_2	$2^8 - 97$	

- Het binaire getal 1111_2 is de *representatie* van -1_{10}. Merk op dat de getallen *modulo* 2^4 zijn!

- Het binaire getal 10011111_2 is de *representatie* van -97_{10}. Merk op dat de getallen *modulo* 2^8 zijn!

DE HAAGSE HOGESCHOOL

2’s complement

- We kunnen nu de getallen gebruiken.
- Als voorbeeld: +5 + (–5) = 0

0101		+5
1011	+	$2^4 - 5$
1) 0000		$2^4 + 0$
↓		
0000		0

- De uitgaande carry moet genegeerd worden, die is het resultaat van de wijze waarop de 2’s complement representatie werkt.

2’s complement

- Bij het omkeren van het teken is het handiger om uit te gaan van $(2^4 – 1) + 1$, want $(2^4 – 1)$ levert allemaal 1-en op!

- Dus +5 → –5:

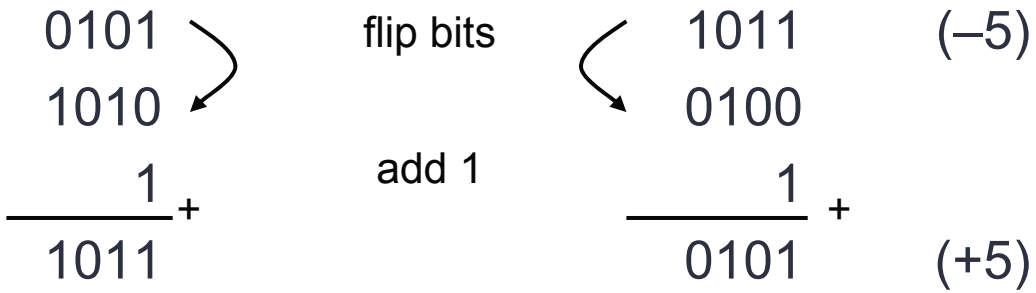
1111		
0101	-	+5
1010		
1	+	
1011		–5

- Merk op dat de schrijfwijze $2^4 – 5$ is vervangen door gewoon –5.

DE HAAGSE HOGESCHOOL

2’s complement

- Een binair getal aftrekken van 1...1 is heel gemakkelijk.
- Het antwoord is namelijk de inverse van de afzonderlijke bits van dat getal (*flip bits*). Daarna moet er 1 bij opgeteld worden (*add 1*). Dit werkt ook van negatief naar positief.
- Als voorbeeld +5 → –5 en –5 → +5



2’s complement

- We willen graag een evenwichtige verdeling van de positieve en negatieve representaties. De helft van de combinaties representeren positieve getallen (inclusief 0) en de andere helft representeert de negatieve getallen.

- Het bereik van een getal *M* gerepresenteerd met 4 bits ligt als volgt:

  $-8 \leq M \leq +7$ (0 wordt positief als beschouwd)

- We zullen later zien dat niet alle resultaten van rekenkundige bewerkingen in het bereik van de representatie van 4 bits (of algemeen: *n* bits) liggen. Er is dan sprake van *overflow*.

Lijst van 2’s complement getallen

- Hieronder een lijst van de getallen –8 t/m +7.

dec	bits	2’s com	
+7	0111	7	
+6	0110	6	
+5	0101	5	
+4	0100	4	
+3	0011	3	
+2	0010	2	
+1	0001	1	
0	0000	0	
–1	1111	15	$= 2^4-1$
–2	1110	14	$= 2^4-2$
–3	1101	13	$= 2^4-3$
–4	1100	12	$= 2^4-4$
–5	1011	11	$= 2^4-5$
–6	1010	10	$= 2^4-6$
–7	1001	9	$= 2^4-7$
–8	1000	8	$= 2^4-8$

2’s complement

- Er is geen tegengesteld getal voor –8. Er is slechts één representatie van 0.

Tekenbit

- Een 2’s complement getal is positief als het meest significante bit 0 is.
- Een 2’s complement getal is negatief als het meest significante bit 1 is.
- Het getal 0 wordt dus als positief gezien.
- Het meest significante bit wordt het *tekenbit* genoemd.
- Merk op dat het tekenbit ook een *gewicht* heeft, zie verderop.

Bereik

- Het bereik van 2's complement getallen is als volgt:

4 bits	–8	≤ N ≤	+7
8 bits	–128	≤ N ≤	+127
16 bits	–32768	≤ N ≤	+32767
32 bits	–2147483648	≤ N ≤	+2147483647
n bits	-2^{n-1}	≤ N ≤	$+2^{n-1} - 1$

- Er is geen tegengesteld getal voor -2^{n-1}. Het bereik van 2's complement getallen is asymmetrisch.

Tekenuitbreiding

- Een 2's complement getal kan met meer bits geschreven worden dan oorspronkelijk door middel van *tekenuitbreiding* (sign extension).
- Het tekenbit moet gekopieerd worden naar de nieuw toe te voegen bits.
- Positieve getallen worden aangevuld met 0-en.
- Negatieve getallen worden aangevuld met 1-en.

Tekenuitbreiding

- Voorbeeld van een positief en negatief getal:

+6	0110	00000110
–6	1010	11111010

- Omrekenen:

+6	00000110	
	11111001	flip bits
	+1	add 1
–6	11111010	

Rekenvoorbeelden

- Enige rekenvoorbeelden

+5	0101	–1	1111
+2 +	0010 +	+3 +	0011 +
+7	0111	+2	~~1~~ 0010
–2	1110	+4	0100
–3 +	1101 +	–8 +	1000 +
–5	~~1~~ 1011	–4	1100

- De uitgaande 1 (carry!) moet genegeerd worden.

Decimaal naar 2’s complement

- Om een decimaal getal om te zetten naar 2’s complement kunnen de volgende regels gebruikt worden:

- Is het decimale getal positief, dan volgt de omzetting zoals al eerder is besproken.

- Is het decimale getal negatief, dan volgt eerst de omzetting van het getal met dezelfde grootte maar positief. Daarna volgt omzetting van het positieve binaire getal naar het negatieve binaire getal.

- Let erop dat de binaire getallen met voldoende bits moeten worden berekend.

Decimaal naar 2's complement

- Hieronder twee voorbeelden. De binaire getallen worden met acht bits beschreven.

omzetting

$76_{10} \longrightarrow 01001100_2$

teken omkeren — omzetting — flip bits, add 1

$-87_{10} \longrightarrow 87_{10} \longrightarrow 01010111_2 \longrightarrow 10101001_2$

DE HAAGSE HOGESCHOOL

2’s complement naar decimaal

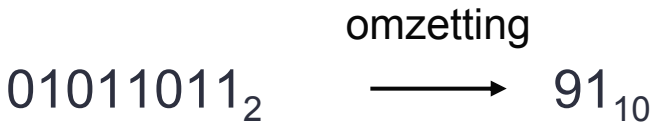
- Om een 2’s complement getal om te zetten naar decimaal kunnen de volgende regels gebruikt worden:

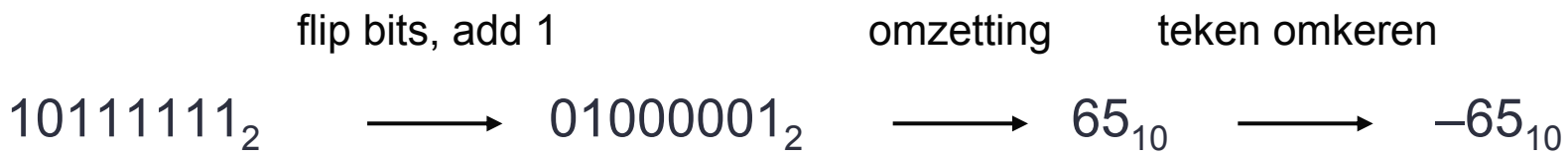
- Is het 2’s complement getal positief, dan volgt de omzetting zoals al eerder is besproken.

- Is het 2’s complement getal negatief, dan volgt eerst de omzetting van het getal met dezelfde grootte maar positief. Daarna volgt omzetting van het positieve decimale getal naar het negatieve decimale getal.

2’s complement naar decimaal

- Hieronder twee voorbeelden. De binaire getallen zijn met acht bits beschreven.





2’s complement naar decimaal (alternatief)

- Het omzetten van een 2’s complement naar decimaal kan eenvoudig door het tekenbit als *negatief* gewicht te gebruiken. De overige bits hebben hetzelfde gewicht als bij unsigned.

- Gegeven een *n* bits 2’s complement getal A = $a_{n-1}a_{n-2}$ .. a_1a_0, dan is het decimale equivalent:

$$A = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$$

(let op het minteken bij $-a_{n-1}$)

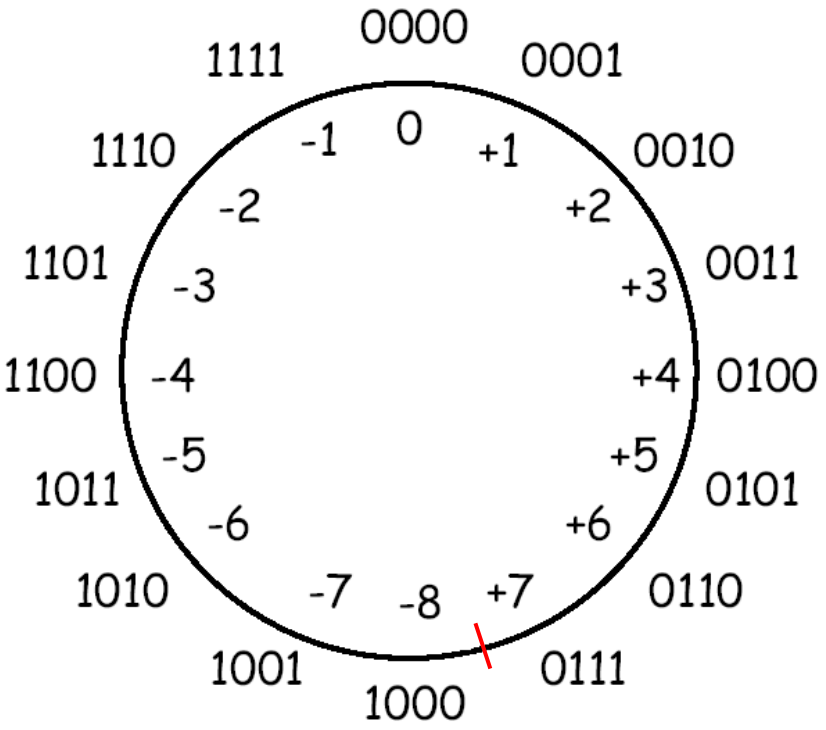
- Voorbeeld: $1011 \rightarrow -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -8 + 3 = -5$

  $0110 \rightarrow 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 0 + 6 = 6$

DE HAAGSE HOGESCHOOL

2’s complement cirkel

- 2’s complement kan inzichtelijk gemaakt worden door een zogenaamde 2’s complement cirkel.
- Getal 0 staat bovenaan, met de klok mee volgen de positieve getallen t/m +7.
- Tegen de klok in volgen de negatieve getallen t/m –8.
- Overflow bij +7 ↔ –8.



Unsigned vs. 2's complement

- Hoe de getallen gebruikt/gelezen moeten worden, hangt af van de *interpretatie* van de gebruiker. Voor de hardware is er geen verschil*.

unsigned	bits	2's compl.	dec
+15	1111	$2^4 - 1$	–1
+15	1111	$2^4 - 1$	–1
+16 +14	1 1110	$2^4 + 2^4 - 2$	–2
↓	↓	↓	↓
+14	1110	$2^4 - 2$	–2

* En dat was ook de bedoeling!

Aftrekken

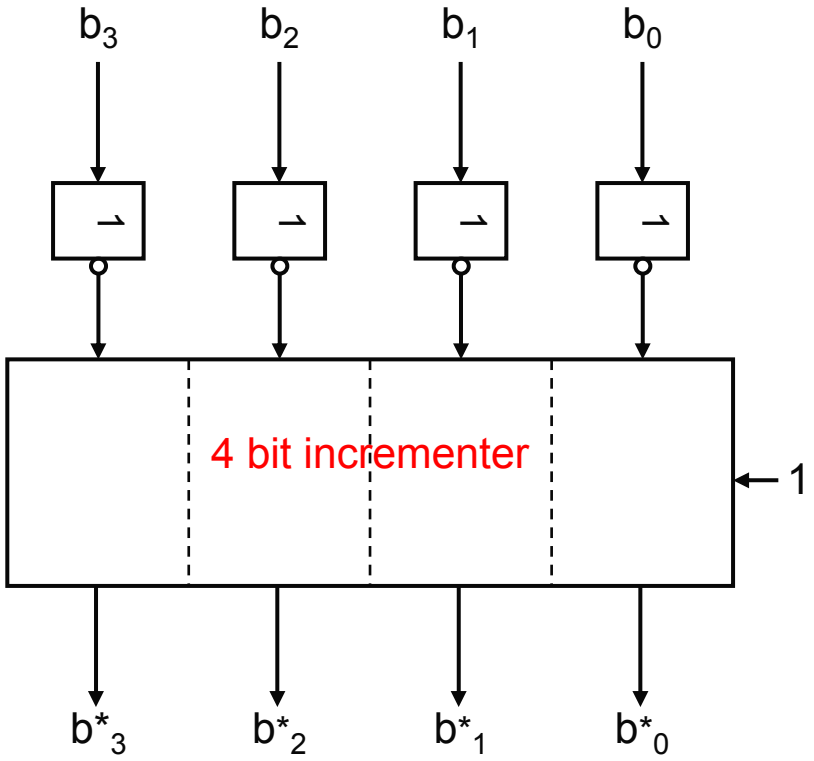
- Het ontwerpen van een aftrekschakeling kan op vergelijkbare wijze als een optelschakeling.
- Het is echter veel handiger om gebruik te maken van de gelijkheid:

$$A - B = A + (-B)$$

- Nu kan een normale full adder gebruikt worden en is alleen een *negator* (schakeling die het getal negatief maakt) nodig.
- Het negatief maken van een 2's complement getal gebeurt door de afzonderlijke bits te inverteren en daarna het getal 1 er bij op te tellen.

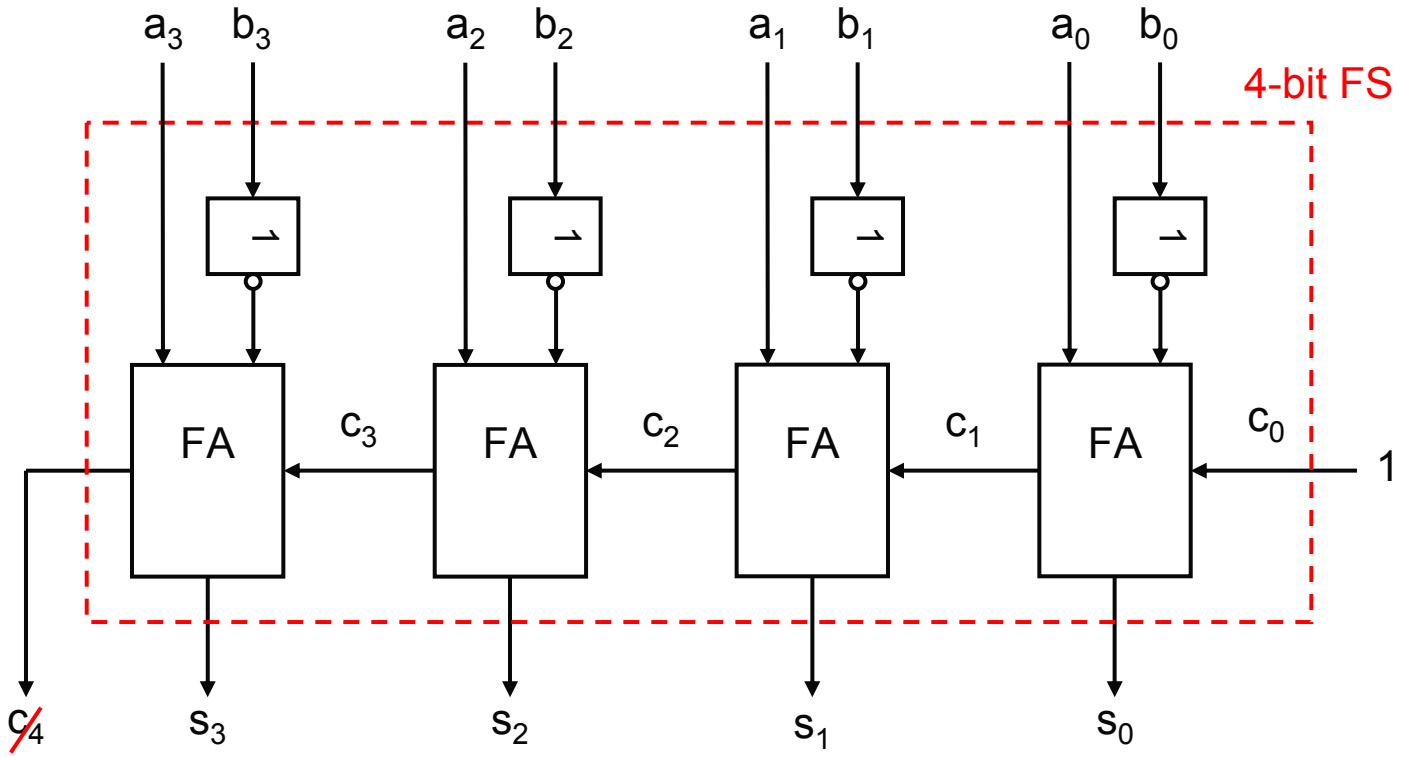
Negator

- Inverteren gebeurt door middel van NOT-poorten.
- Het optellen van 1 kan gedaan worden door de *incrementer*.



Aftrekschakeling

- Het werking van de negator kan worden opgenomen in een gewone full adder. Dit levert een aftrekschakeling (full subtractor).



Optel/aftrekschakeling

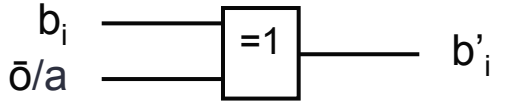
- Een optelschakeling kan gecombineerd worden met een aftrekschakeling.
- Een (nieuw) besturingssignaal ō/a geeft aan op welke operatie uitgevoerd moet worden.
- Als ō/a = 0 moet er worden opgeteld.
- Als ō/a = 1 moet er worden afgetrokken.

Optel/aftrekschakeling

- De operatie A – B moet worden omgezet in A + (–B) want dan kan een gewone full adder gebruikt worden.
- Voor aftrekken is dus –B nodig. Het negatief maken van B gebeurt door alle bits te inverteren en er dan 1 bij optellen (2’s complement).
- Voor optellen is B nodig. Er hoeft niets te worden aangepast.
- Het (al dan niet) inverteren van de afzonderlijke bits van B kan gedaan worden door de omschakelbare buffer/NOT-poort.

Optel/aftrekschakeling

- De buffer/NOT-poort is de bekende EXOR!



b_i is het i^e bit van getal B

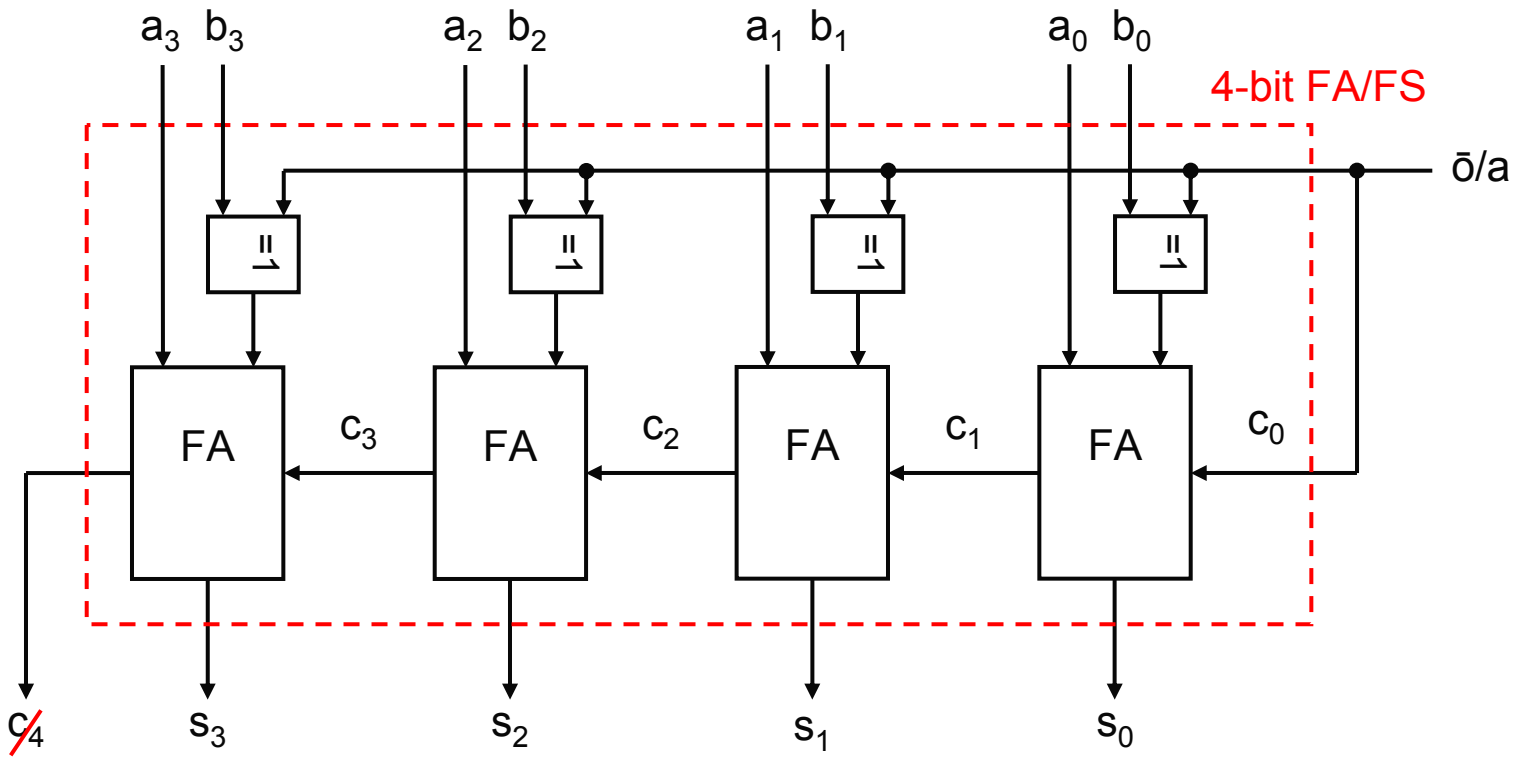
$$\left.\begin{array}{l} b'_i = b_i \quad \text{als} \quad \bar{o}/a = 0 \\ b'_i = \overline{b_i} \quad \text{als} \quad \bar{o}/a = 1 \end{array}\right\} \quad b'_i = \overline{\bar{o}/a} \cdot b_i + \bar{o}/a \cdot \overline{b_i}$$

- Bij optellen moet de c_0 aan een logische 0 verbonden zijn, dus als ō/a = 0.

- Bij aftrekken moet de c_0 aan een logische 1 verbonden zijn, dus als ō/a = 1.

Optel/aftrekschakeling

- Hieronder de complete optel/aftrekschakeling.



Overflow

- Bij het het optellen van twee 2's complement getallen kan het zijn dat het antwoord buiten het bereik van de representatie komt te liggen.
- Er wordt dan gesproken van *overflow* (overstromen, overlopen).
- Overflow kan voorkomen als twee getallen van gelijk teken worden opgeteld.
- Getallen van verschillend teken kunnen nooit overflow opleveren.
- NB: overflow bij *unsigned getallen* is eenvoudig te zien aan de *uitgaande* carry.

Overflow

- Enige rekenvoorbeelden

+5	0101			–8	1000		
+5	+ 0101	+		–8	+ 1000	+	
+10	1010	(= -6)		–16	1 0000	(= 0)	
–7	1001			+4	0100		
–4	+ 1100	+		–8	+ 1000	+	
–11	1 0101	(= +5)		–4	1100		

- De uitgaande 1 (carry!) moet genegeerd worden.

Overflow

- Overflow is gemakkelijk te detecteren. Overflow kan alleen gebeuren als twee getallen met gelijk teken worden opgeteld.
- Als de tekens van de op te tellen getallen anders zijn dan dat van het antwoord is er overflow.
- Er zijn twee 4-bit getallen A = $a_3a_2a_1a_0$ en B = $b_3b_2b_1b_0$.
- Het antwoord is de 4-bit som S = $s_3s_2s_1s_0$.
- Er is overflow als $(a_3 = b_3) \neq s_3$.

Overflow

- De functie voor overflow van een 4-bit full adder is

$$ov_{op} = a_3 \cdot b_3 \cdot \overline{s_3} + \overline{a_3} \cdot \overline{b_3} \cdot s_3$$

- De functie voor overflow van een 4-bit full subtractor is (zonder aan te tonen)

$$ov_{af} = a_3 \cdot \overline{b_3} \cdot \overline{s_3} + \overline{a_3} \cdot b_3 \cdot s_3$$

- De functie voor overflow is te combineren

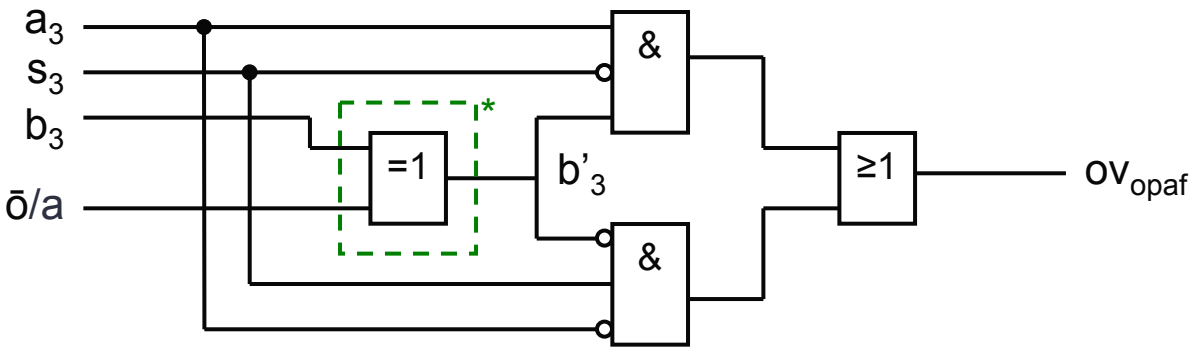
$$ov_{opaf} = \overline{\overline{o}/a} \cdot \left(a_3 \cdot b_3 \cdot \overline{s_3} + \overline{a_3} \cdot \overline{b_3} \cdot s_3\right) + \overline{o}/a \cdot \left(a_3 \cdot \overline{b_3} \cdot \overline{s_3} + \overline{a_3} \cdot b_3 \cdot s_3\right)$$

Overflow

- Deze functie is om te werken:

$$ov_{opaf} = \overline{\overline{o}/a} \cdot \left(a_3 \cdot b_3 \cdot \overline{s_3} + \overline{a_3} \cdot \overline{b_3} \cdot s_3\right) + \overline{o}/a \cdot \left(a_3 \cdot \overline{b_3} \cdot \overline{s_3} + \overline{a_3} \cdot b_3 \cdot s_3\right)$$
$$= a_3 \cdot \left(\overline{o}/a \oplus b_3\right) \cdot \overline{s_3} + \overline{a_3} \cdot \left(\overline{\overline{o}/a \oplus b_3}\right) \cdot s_3$$

- De hardware voor deze functie:

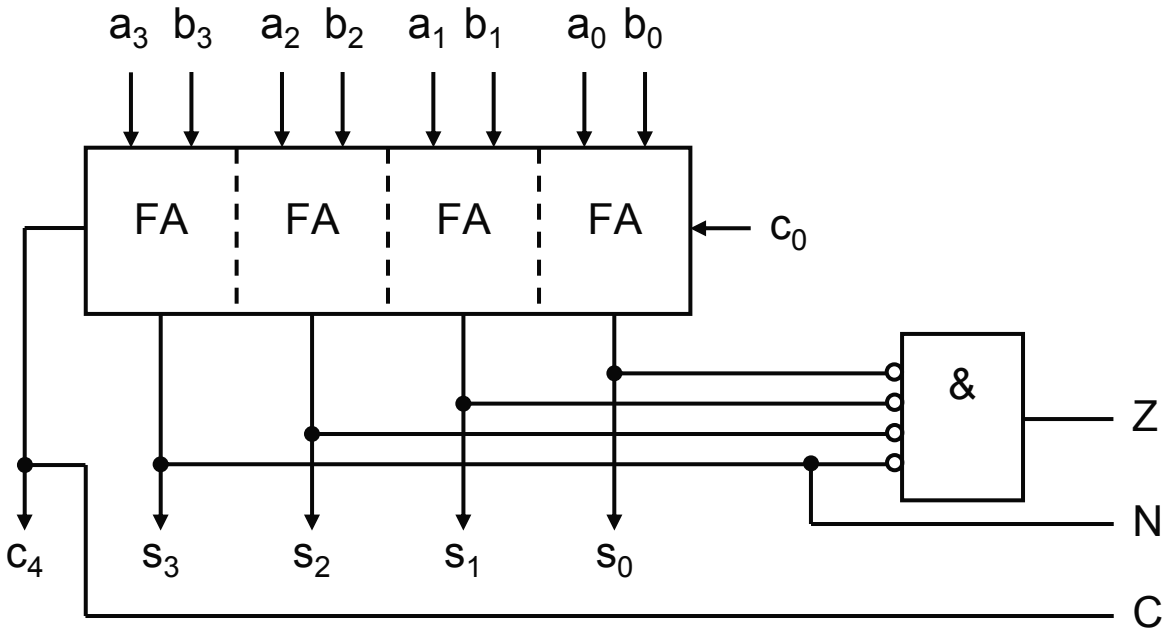


Zero en negative flags

- In elke microprocessor zit een rekeneenheid. Deze eenheid wordt doorgaans de ALU (*Arithmetic and Logic Unit*) genoemd, en kan naast optellen en aftrekken ook alle logische bewerkingen.
- Het detecteren van overflow is noodzakelijk om berekeningen betrouwbaar te laten verlopen.
- Daarnaast is het handig om te weten of een berekening 0 heeft opgeleverd of negatief is.
- Dit wordt weergegeven in de *flags*: o<u>V</u>erflow, <u>Z</u>ero, <u>N</u>egative, <u>C</u>arry

Zero, carry en negative flags

- Het detecteren van de zero-conditie is vrij simpel. Alle sombits moeten 0 zijn, dan is Z =1. De N is hetzelfde als het tekenbit. C levert 1 als er een carry uit de (totale) FA komt.



Opgaven

- Reken om van decimaal naar 2's complement:

  +5, –5, –100, –64,+25, –25

- Reken uit in 2's complement en let daarbij op overflow en tekenuitbreiding:

  011001 + 100100
  011001 – 001110

  110011 + 100011
  101010 – 010101

  1101 + 111001
  011100 – 011001

  101011 – 1100111
  10001 + 1100111

1’s complement

- Bij 1’s complement wordt het negatief maken van een binair getal gedaan door alleen de afzonderlijke bits te inverteren.
- Het voordeel is dat er geen +1 bij opgeteld hoeft te worden.
- Het nadeel is dat er twee getalrepresentaties zijn voor 0, en dat optellen nu niet meer zo makkelijk gaat: “door de 0 heentellen” levert nu problemen op (*end around carry*).
- 1’s complement wordt niet meer gebruikt, maar komt in de boeken nog wel voor om aan te geven dat alle bits geinverteerd worden.

2’s complement en hexadecimaal

- Hexadecimale getallen worden gebruikt om binaire getallen gecomprimeerd op te schrijven en kunnen dus ook negatief zijn.

- Hexadecimale getallen die beginnen met de cijfers 8 t/m F zijn negatief, want ze beginnen met een binaire 1.

  8 = 1000, 9 = 1001, A = 1010, B = 1011

  C = 1100, D = 1101, E = 1110, F = 1111

- Voorbeeld: $D0_{16} = 1101.0000_2 = -2^7 + 2^6 + 2^4 = -48_{10}$

  $FFFFFFFF_{16} = 11...11_2 = -2^{31} + 2^{30} + ... + 2^1 + 2^0 = -1_{10}$

DE HAAGSE HOGESCHOOL

2’s complement en hexadecimaal

- We kunnen gebruik maken van 2’s complement representatie. Een hexadecimaal cijfer tussen 8 en F stelt een negatief getal voor. Dit geldt uiteraard alleen voor het meest significante cijfer.

- Dus: $8_{16} = -8_{10}$ $9_{16} = -7_{10}$ $A_{16} = -6_{10}$ $B_{16} = -5_{10}$
  $C_{16} = -4_{10}$ $D_{16} = -3_{10}$ $E_{16} = -2_{10}$ $F_{16} = -1_{10}$

- Voorbeeld:

  $C5_{16} = -4 \cdot 16 + 5 = -48 + 5 = -43_{10}$

  $FFFF_{16} = -1 \cdot 16^3 + 15 \cdot 16^2 + 15 \cdot 16^1 + 15 \cdot 16^0 = -1_{10}$

DE HAAGSE HOGESCHOOL

Opgaven

- Geef het decimale equivalent van de volgende hexadecimale 2's complement getallen:

  $FFFF_{16}$ $53BA_{16}$ CB_{16} $7EA3_{16}$

- Wat zijn de kleinste en grootste decimale waarden van een hexadecimaal 2's complement getal van 8 cijfers?

- Schrijf als hexadecimale 2's complement getallen met vier cijfers:

  F_{16} 6_{16} $7A_{16}$ CB_{16} $35B_{16}$ $D73_{16}$

BCD-optellen

- De eerder besproken binaire opteller kan alleen zuiver binaire getallen op tellen (er wordt ook wel gesproken van normale binaire code = NBC).

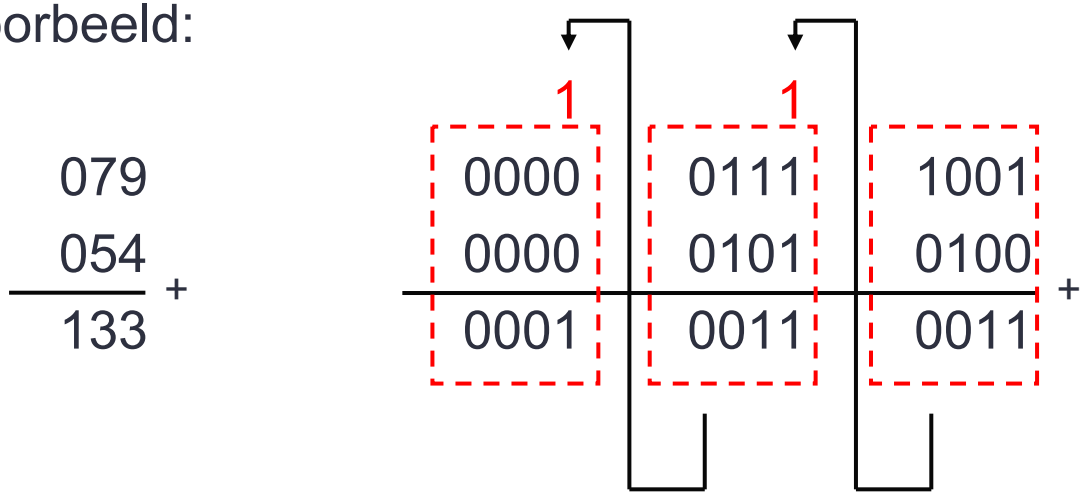
- Optellen van BCD-getallen gaat niet direct:

45	0100 0101	079	0000 0111 1001
43 +	0100 0011 +	054 +	0000 0101 0100 +
88	1000 1000	133	0000 1100 1101
decimaal	BCD	decimaal	? BCD ?

BCD-optellen

- Het probleem zit in het feit dat het optellen van twee BCD-cijfers een resultaat groter dan 9 kan opleveren, waardoor er een (BCD-)carry naar de volgende sectie zal moeten worden doorgegeven.

- Voorbeeld:



BCD-optellen

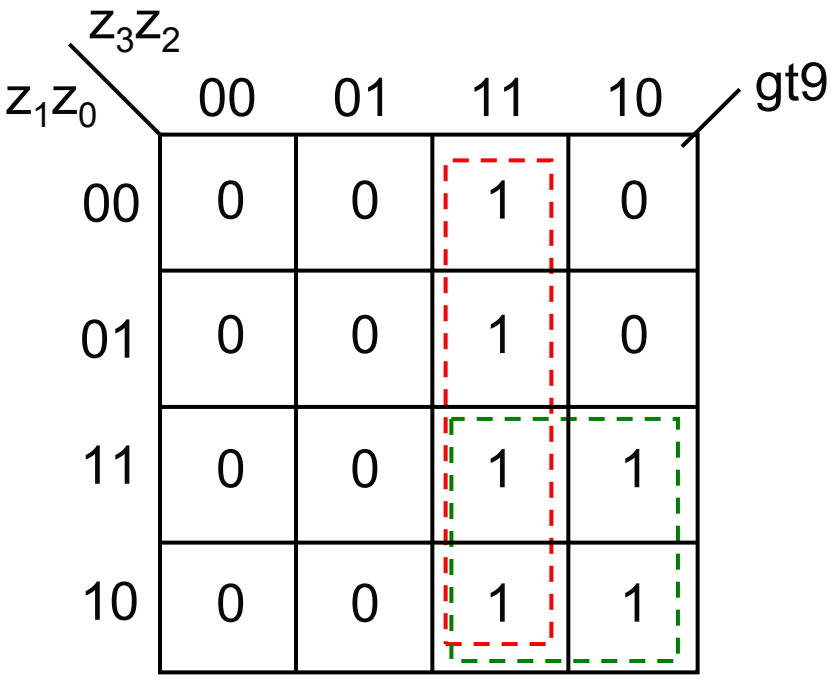
- Het (binaire) resultaat moet gecorrigeerd worden door 6 erbij op te tellen.
- De BCD-carry komt er dan automatisch uit.
- Voorbeelden:

0	0	1	1
6	0110	9	1001
7 +	0111 +	9 +	1001 +
13	1101	19	1 0011 +
	0110 +		0110 +
	1 0011		1 1001

BCD-optellen

- Het resultaat moet worden gecorrigeerd als het groter is dan 9, dus bij 1010_2, 1011_2, 1100_2, 1101_2, 1110_2, en 1111_2 (en $c_4 = 0$).

- Invullen in een Karnaughdiagram levert:

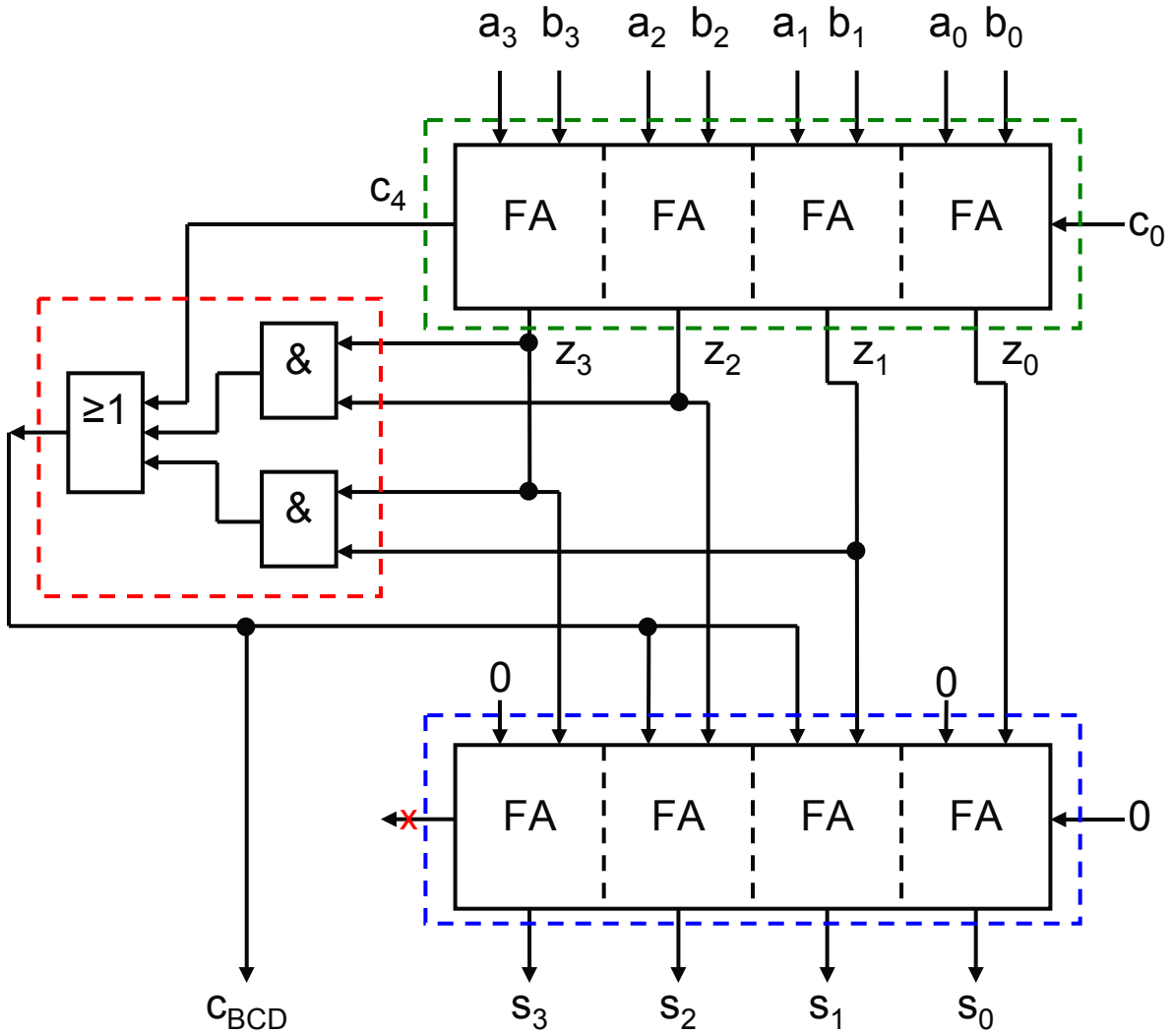


$$gt9 = z_3 \cdot z_2 + z_3 \cdot z_1$$

BCD-optellen

- Maar als het resultaat groter is dan 15 (8+8, 9+9+1 etc) moet er ook een correctie plaatsvinden, dus als $c_4 = 1$.
- De functie is dan: $c_{BCD} = c_4 + z_3 \cdot z_2 + z_3 \cdot z_1$
- De naam c_{BCD} is niet zomaar gekozen, de functie geeft ook aan wanneer er een carry is voor de volgende BCD-sectie.
- Een schakeling voor één BCD-opteller bestaat uit een binaire opteller, een detectieschakeling en een correctie-opteller.

BCD-opteller



- De BCD-opteller:
- Groen is de eerste 4-bit FA, die zuiver binair optelt.
- Rood is de detectie-schakeling.
- Blauw is de correctie-FA, die er 6 bij optelt.

DE HAAGSE HOGESCHOOL

Literatuur

- Boeken:

  Computers (organisatie/architectuur/communticatie), F.J. Dijkstra, 1e druk.
  Digitale techniek, van probleemstelling tot realisatie deel 1; A.P. Thijssen; 5e druk.
  Digital Design, Principles and Practices; J.F. Wakery; 3rd Ed.
  Digital Design, M. Morris Mano, 4th Ed, blz 130-142.
  Fundamentals of Digital Logic with VHDL Design, 3rd Ed, S. Brown.
  Digital Systems, 11th Ed, R. J. Tocci
  Dictaat hoofdstuk 5

Literatuur

- Internet:

  Binary Coded Decimal op Wiki: http://en.wikipedia.org/wiki/Binary-coded_decimal

  2's complement op Wiki: http://en.wikipedia.org/wiki/Two's_complement (lees vooral het onderdeel “Why it works”)

  Signed Number Representations (inclusief signed maginitude, 2's complement en 1's complement) op Wiki: http://en.wikipedia.org/wiki/Signed_number_representations

  Emytologie negator: http://en.wiktionary.org/wiki/negator

  Status Register (flags) op Wiki: http://en.wikipedia.org/wiki/Status_register

  ALU op Wiki: http://en.wikipedia.org/wiki/Arithmetic_logic_unit

Efficient coderen?

- Een BCD-cijfer gebruikt vier bits, dus een n-cijferig BCD-getal gebruikt $m_{BCD} = 4 \cdot n$ bits.

- Het aantal bits voor een n-cijferig decimaal getal (dus niet BCD) is:

$$m_{dec} = \log_2 10^n = n \cdot \log_2 10 = 3{,}322 \cdot n$$

- De verhouding is: $r_n = \frac{m_{BCD}}{m_{dec}} = \frac{4}{3{,}322} = 1{,}204$

- Hieruit kan geconcludeerd worden dat voor BCD-codering 20% meer bits nodig zijn.

De carry bij aftrekken

- De carry-out heeft een andere rol bij aftrekken. De carry-out geeft nu aan of er geleend moet worden, d.w.z. als A kleiner is dan B.

Vergelijk	Subtract	Add	C_{out}	Leen
A < B	A – B < 0	A + (2^n – B) < 2^n	C = 0	L = 1
A = B	A – B = 0	A + (2^n – B) = 2^n	C = 1	L = 0
A > B	A – B > 0	A + (2^n – B) > 2^n	C = 1	L = 0

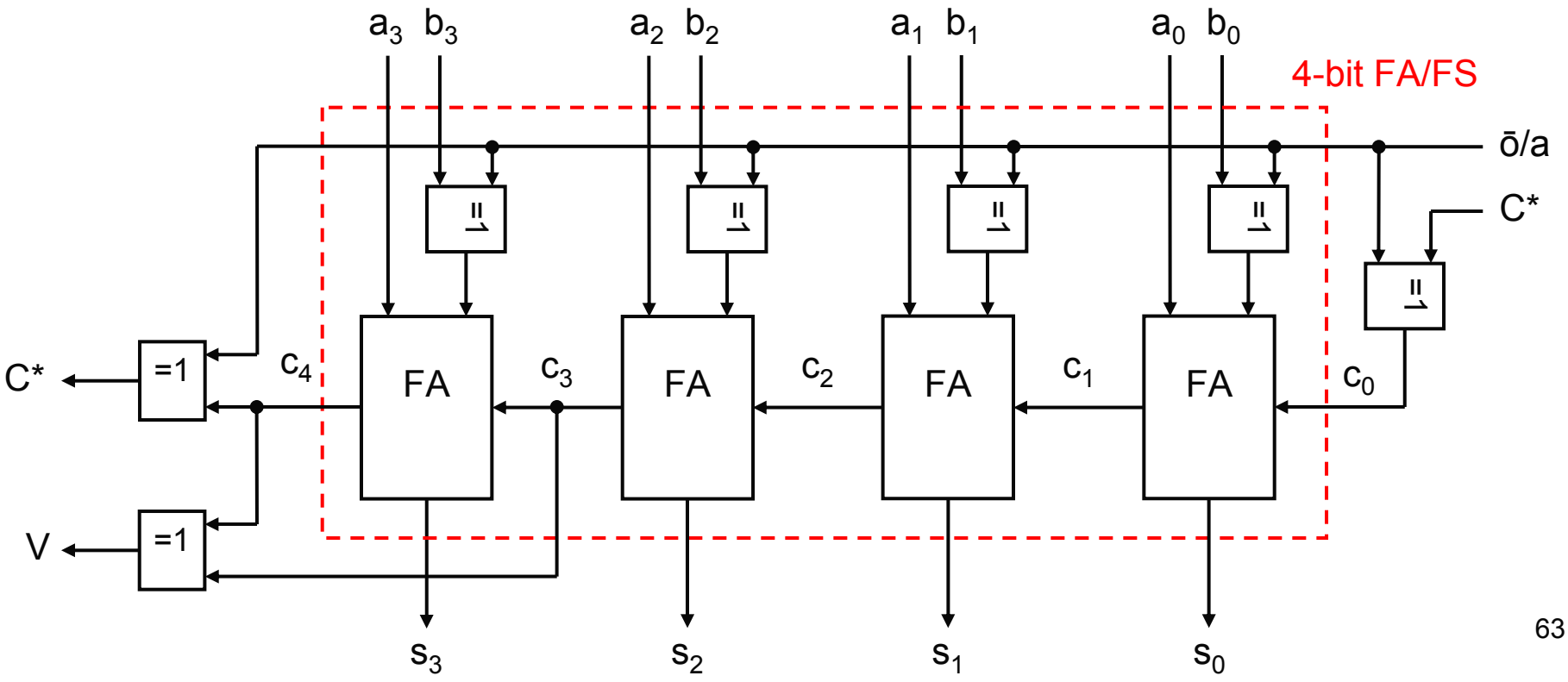
- Uit de tabel blijkt dat de Leen de inverse is van de C_{out}. Het engelse woord voor Leen is *Borrow*.

- In microprocessors worden de Carry en de Borrow gecombineerd in de C-flag. Bij aftrekken wordt de inverse van de C_{out} opgeslagen in de C-flag*.

* zie slide Alternatieve hardware overflow en carry

Alternatieve hardware overflow en carry

- Na bestudering van het optel/aftrekproces blijkt dat overflow ook te detecteren is als de ingaande carry bij de tekenbit ongelijk aan de uitgaande carry bij het tekenbit, dus: $V = c_4 \oplus c_3$



* zie slides De carry bij aftrekken

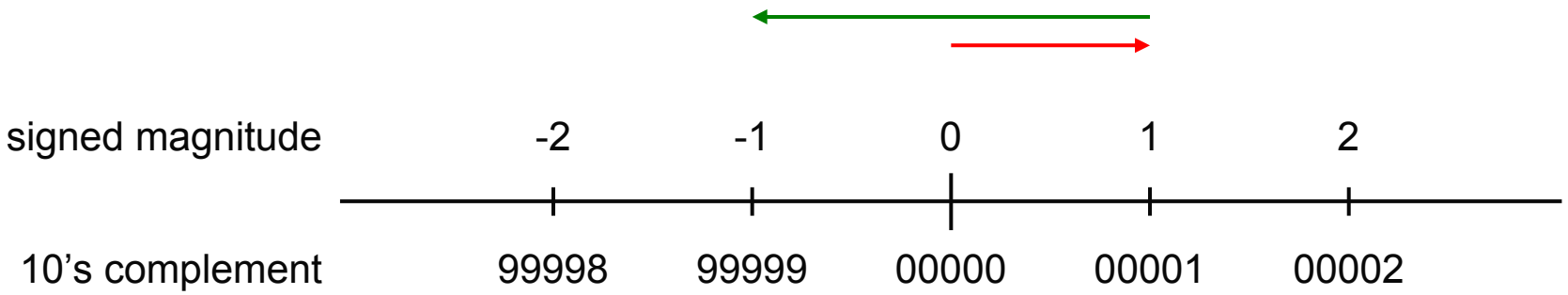
10’s complement

- De complement-representatie is ook in het decimale talstel toe te passen. Dit wordt de 10’s complement representatie genoemd
- Het 10’s complement representatie kan inzichtelijk gemaakt worden door het vergelijken met een kilometerteller.
- In het begin staat de kilometerteller op 00000.
- Als er één kilometer *vooruit* wordt gereden, staat de kilometerteller op 00001.
- Dit kan worden opgevat als 1 kilometer *vooruit*.

10’s complement

- Als er nu twee kilometer *achteruit* wordt gereden, staat de kilometerteller op 99999.

- Dit kan worden opgevat als –1 kilometer *vooruit*:



10’s complement

- Nu kan een tabel worden opgesteld voor enige getallen:

5 → 00005
4 → 00004
3 → 00003
2 → 00002
1 → 00001
0 → 00000
–1 → 99999
–2 → 99998
–3 → 99997
–4 → 99996
–5 → 99995

1849 → 01849
–1849 → 98151
–19999 → 80001
–49999 → 50001

10’s complement

- Optellen (en later ook aftrekken) gaat nu heel eenvoudig:

+2	00002	+1849	01849
–1	99999	–679	99321
+1 +	~~1~~ 00001 +	+1170 +	~~1~~ 01170 +
+457	00457	–3456	96544
–2345	97655	–1234	98766
–1888 +	98112 +	–4690 +	~~1~~ 95310 +

- De uitgaande 1 (carry!) moet genegeerd worden.

10’s complement

- Het 10’s complement van een negatief getal met 5 cijfers wordt gemaakt door het getal af te trekken van 10^5.
- Dus –1234 → 10^5 – 1234 = 98766
- Slimmer is om te doen: (10^5 – 1) – 1234 +1
- 10^5 – 1 levert namelijk allemaal negens op. Per kolom hoeft er dan niet geleend te worden.

100000		99999		(=10^5 – 1)
1234	-	1234	-	
98766		98765		
		1	+	
		98766		

10’s complement

- Het signed magnitude systeem kan een onbeperkt aantal getallen weergeven, van -∞ tot +∞. Het bereik is *onbegrensd*.
- Bij het 10’s complement is het bereik wel begrensd. Dat houdt in dat 5 decimalen alle berekeningen *modulo* 10^5 worden uitgevoerd.
- In het voorbeeld loopt het gebied bereik –50000 (50000) tot +49999 (49999).
- Er is dus geen tegengesteld getal voor –50000!
- Het bereik is asymmetrisch!
- 10’s complement wordt gebruikt bij BCD-rekenschakelingen.

Opgaven

- Negatieve getallen in BCD-formaat worden weergegeven in 10's complement. Dit is rekenkundig te doen door het BCD-getal af te trekken van 9…9 en daarna er 1 bij op te tellen (uiteraard met een BCD-opteller). Dit aftrekken van 9…9 is het zogenaamde 9's complement. Het voordeel van hiervan is dat per kolom niet geleend hoeft te worden.

  Ontwerp een digitale schakeling waarin een aangeboden BCD-cijfer wordt afgetrokken van 9. Stel een waarheidstabel op, minimaliseer de functies m.b.v. Karnaughdiagrammen en teken de schakeling van de functies met poorten naar eigen keuze.

Academie voor Technology, Innovation & Society Delft

Academie voor ICT & Media

De Haagse Hogeschool, Delft
+31-15-2606311
J.E.J.opdenBrouw@hhs.nl
www.dehaagsehogeschool.nl

DE HAAGSE HOGESCHOOL