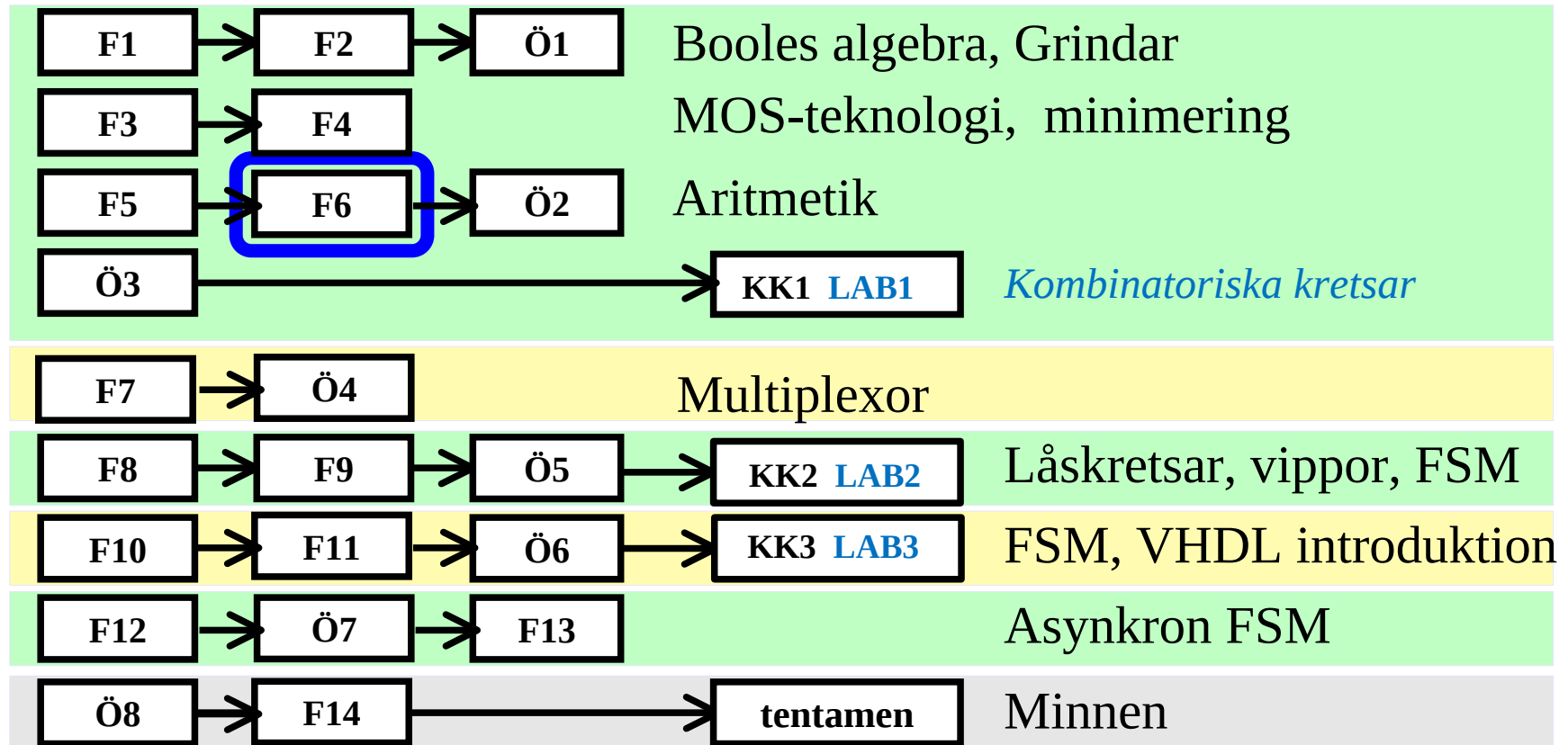


Digital Design IE1204

F6 Digital aritmetik II

`william@kth.se`

IE1204 Digital Design



*Föreläsningar och övningar bygger på varandra! Ta alltid igen det Du missat!
Läs på i förväg – delta i undervisningen – arbeta igenom materialet efteråt!*

Detta har hänt i kursen ...

Decimala, hexadecimala, oktala och binära talsystemen

AND OR NOT EXOR EXNOR Sanningstabell, mintermer Maxtermer PS-form

Booles algebra SP-form deMorgans lag Bubbelgrindar Fullständig logik

NAND NOR CMOS grindar, standardkretsar Minimering med Karnaugh-diagram 2, 3, 4, 5, 6 variabler

Registeraritmetik tvåkomplementrepresentation av binära tal

Additionskretsar

Talrepresentation

Ett tal kan representeras binärt på många sätt.
De vanligaste taltyperna som skall representeras är:

- Heltal, positiva heltal (eng. integers)
ett-komplementet, två-komplementet, sign-magnitude
- Decimala tal med fix tal-område
Fix-tal (eng. fixed-point)
- Decimala tal i olika talområden
Flyt-tal (eng. floating-point)

Heltal

Positiva Heltal:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	0	1	1	0	1

$$= 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 = 109$$

Negativa Heltal:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	1	0	1

$$= -1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^0 = -19$$

Multiplikation av två positiva heltal

$$\begin{array}{r}
 \boxed{0010} \\
 * 1011 \\
 \hline
 0010 \\
 0010 \\
 0000 \\
 + 0010 \\
 \hline
 0010110
 \end{array}$$

$$\begin{array}{r}
 2 \\
 *11 \\
 \hline
 \end{array}$$

Skifta, addera
multiplikanden eller 0

$$=22$$

Multiplikation med teckenbit

Teckenbiten har negativ vikt!
=> Två-komplementera!

$$\begin{array}{r} 0010 2 \\ * 1011 -5 \\ \hline 0010 \\ 0010 \\ 0000 \\ + 1110 \\ \hline 1110110 -10 \end{array}$$

Teckenbiten på det andra stället

Teckenförläng! (Sign Extension, här till 7 bitar)

Diagram illustrating the addition of two 8-bit numbers in two's complement:

Number 1: 1 0 1 1 (Decimal: -5)

Number 2: 0 0 1 0 (Decimal: 2)

Sum: 1 1 0 1 (Decimal: -3)

The diagram shows the addition process step by step, with the final result 1101 highlighted in a green box.

Multiplikation av två negativa tal

Teckenförläng! →

Teckenbiten har negativ vikt!
=> Två-komplementera!

				1	0	1	1	-5
	*			1	0	1	1	-5
		1	1	1	1	0	1	1
		1	1	1	0	1	1	
			0	0	0	0		
+		0	1	0	1			
		0	0	1	1	0	0	1
								+25

Svar 7-bitar med tecken

Det här verkade komplicerat ...

Eller så gör vi det enkelt för oss

Använd enbart positiva tal i multiplikationen

Konvertera till positiva tal

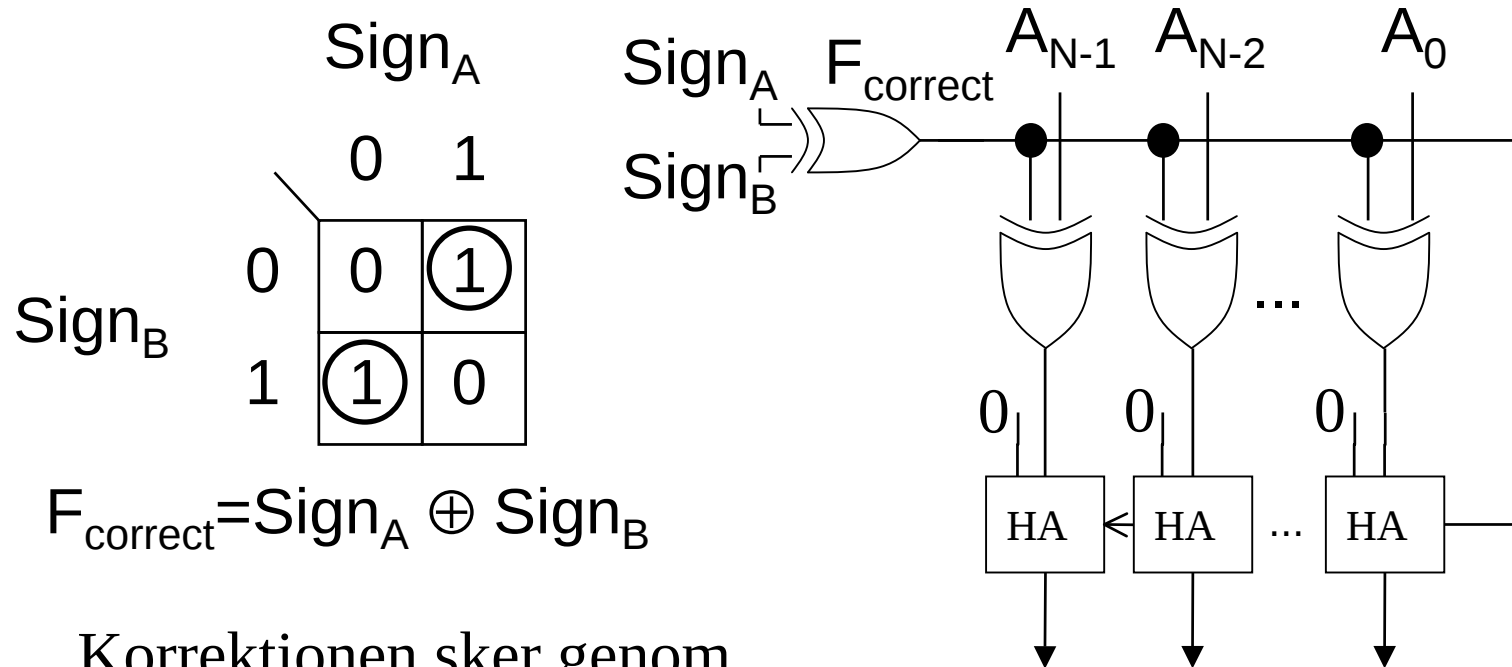
Håll reda på resultatets tecken

$$(+) \cdot (+) \Rightarrow (+) \quad (+) \cdot (-) \Rightarrow (-)$$

$$(-) \cdot (+) \Rightarrow (-) \quad (-) \cdot (-) \Rightarrow (+)$$

Två-komplementera till negativt tal om
nödvändigt

Enkel lösning, forts ...

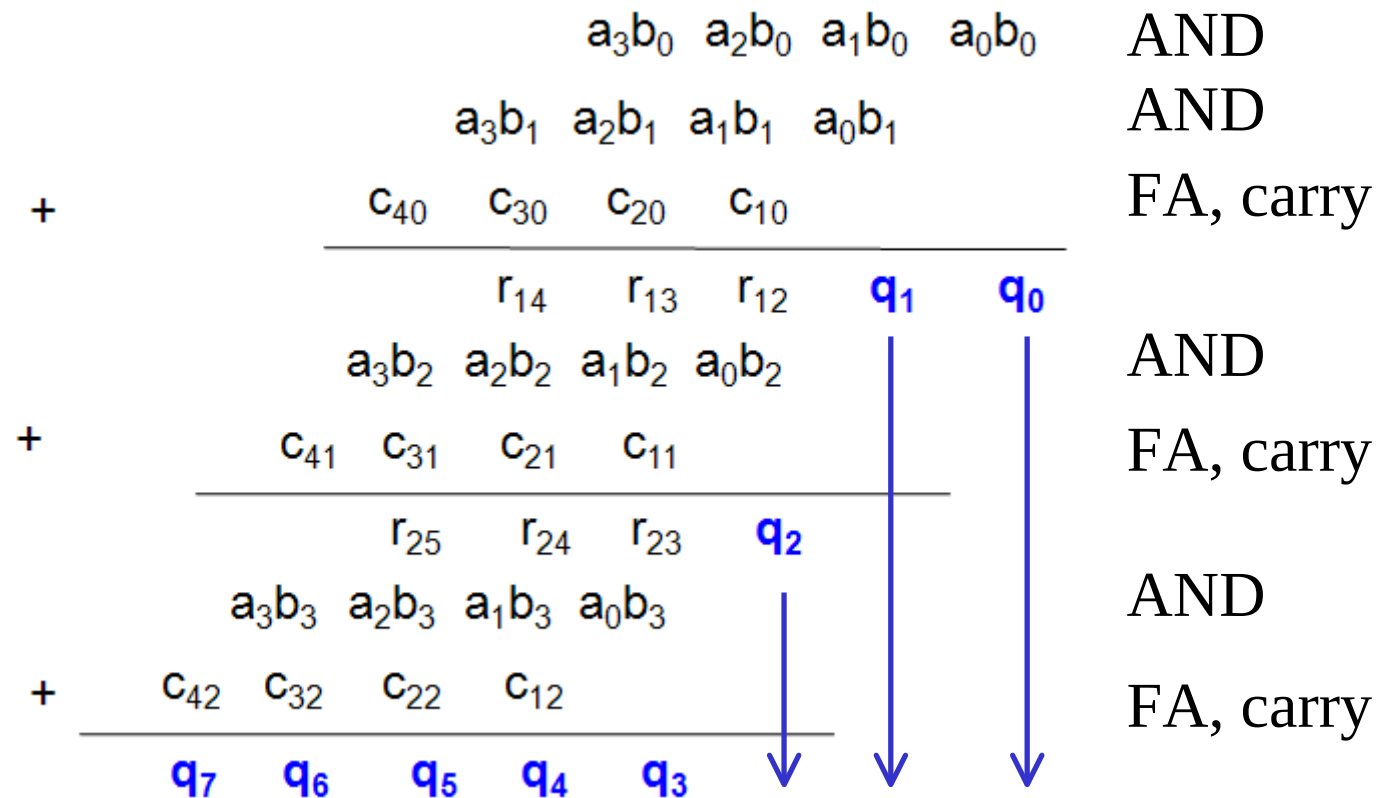


Korrektionen sker genom
att man invertera bitarna
och lägg till 1

**2's complement of Product,
when correction is needed.**

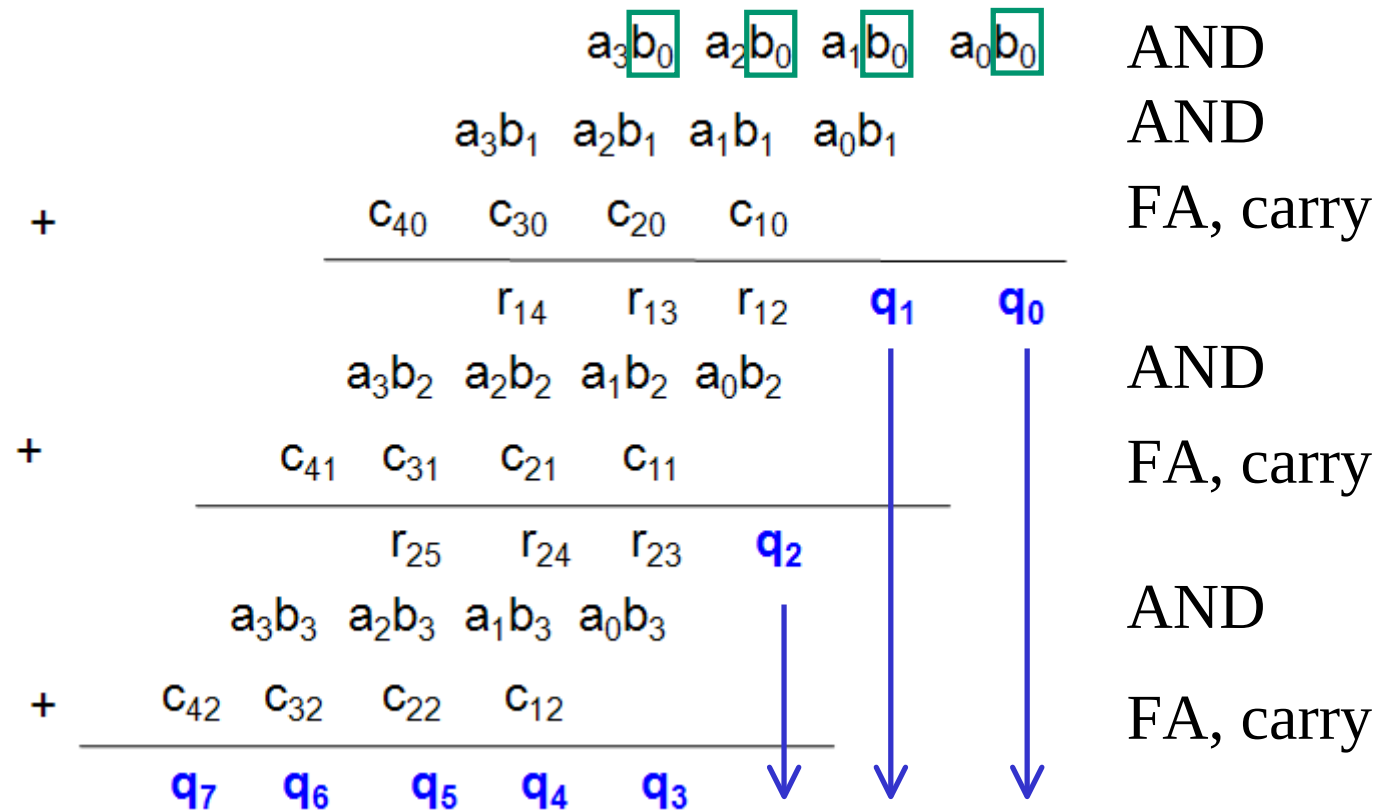
Multiplikation (två positiva tal)

$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$



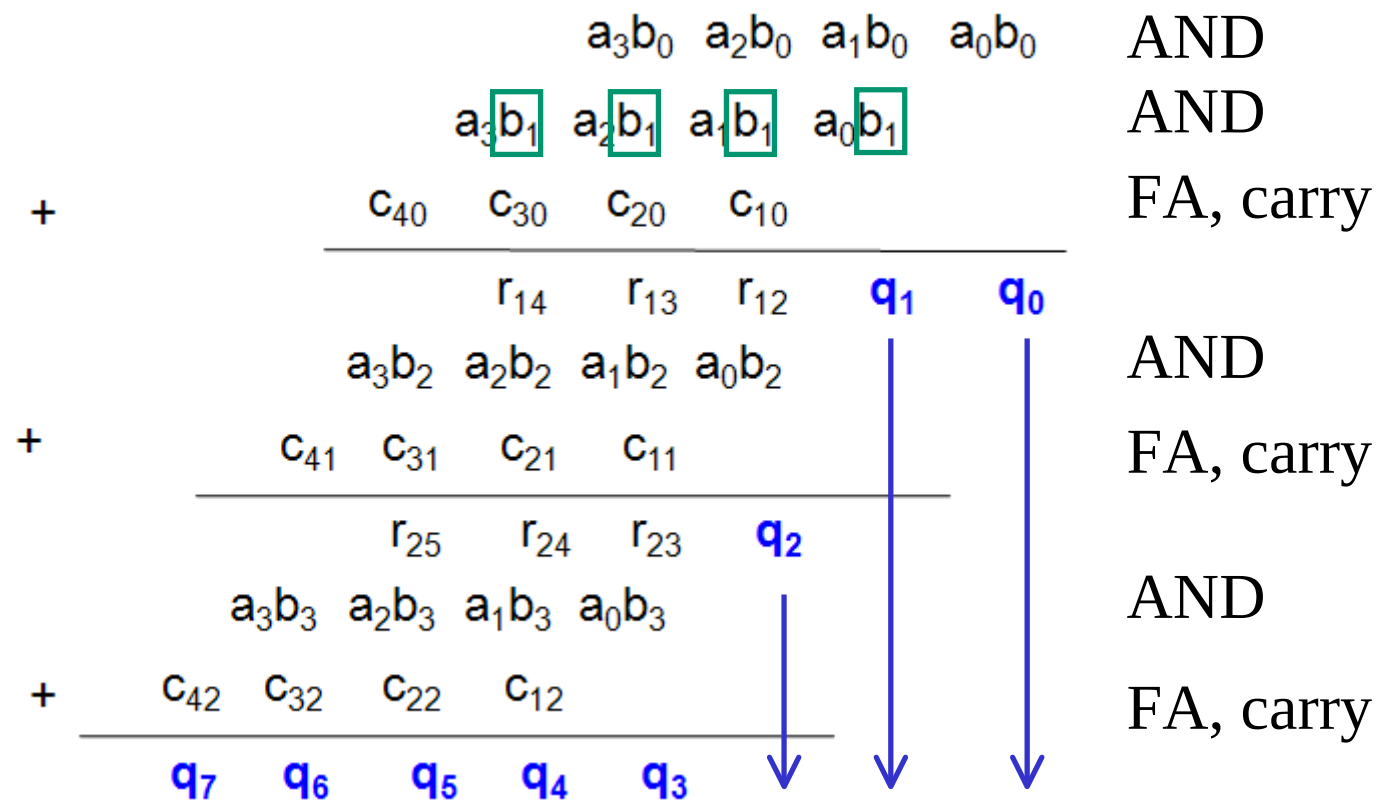
Multiplikation (två positiva tal)

$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$



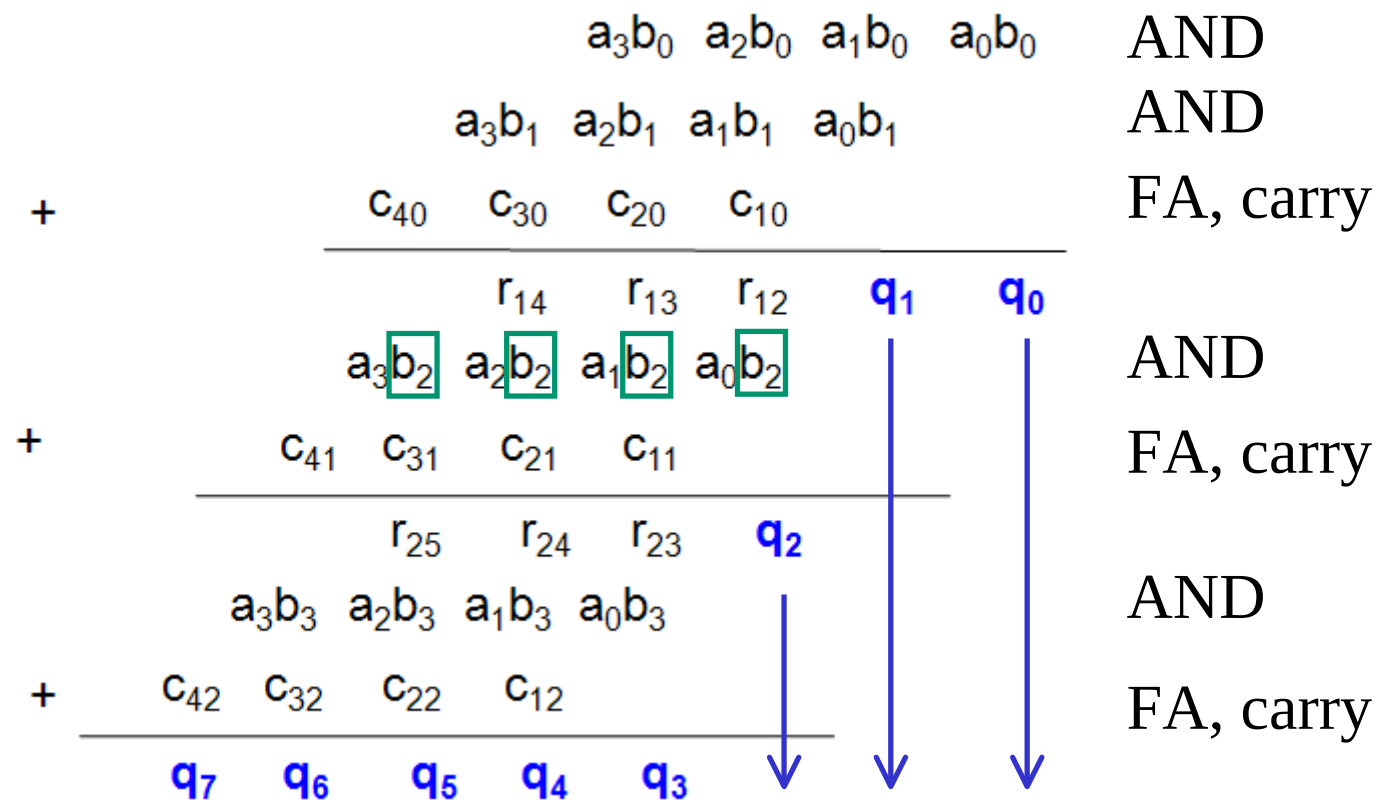
Multiplikation (två positiva tal)

$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$



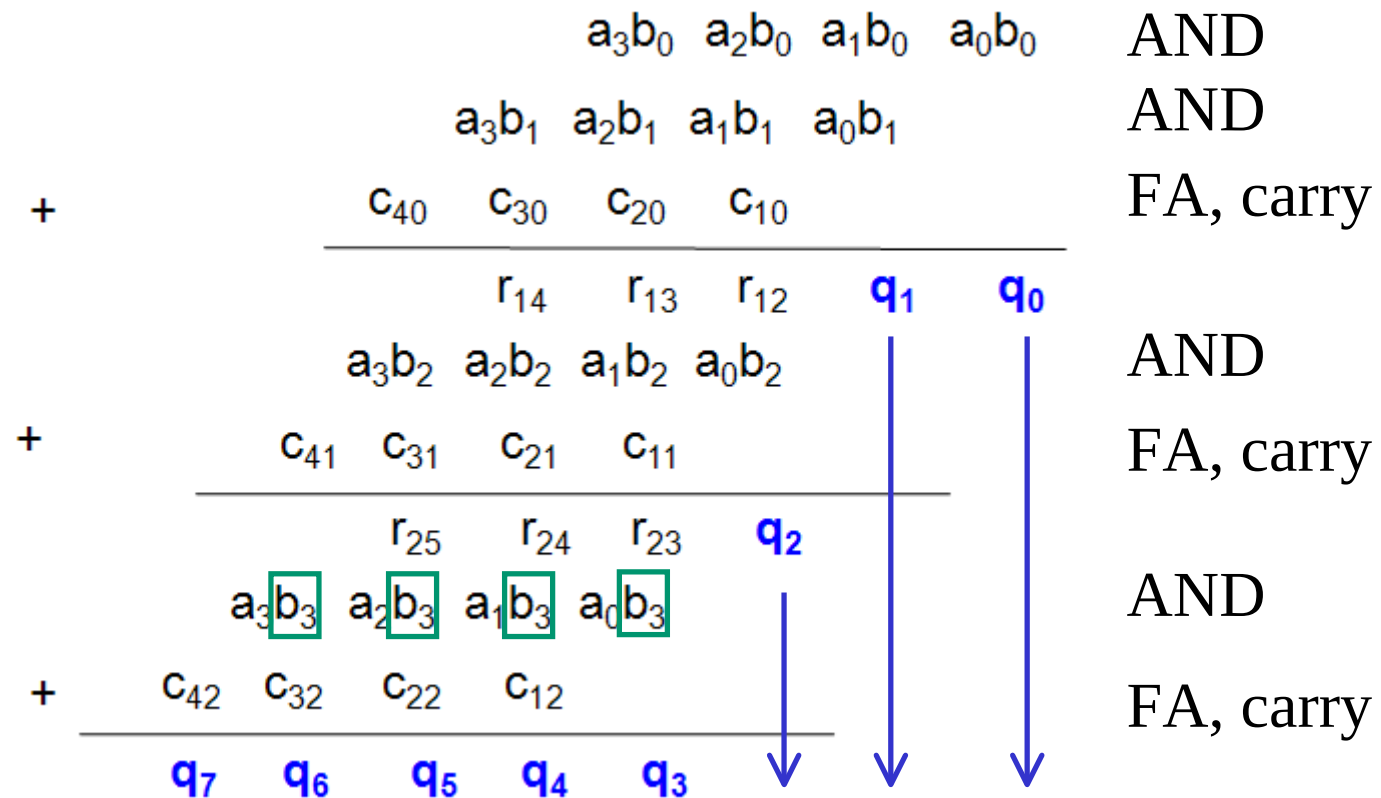
Multiplikation (två positiva tal)

$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$



Multiplikation (två positiva tal)

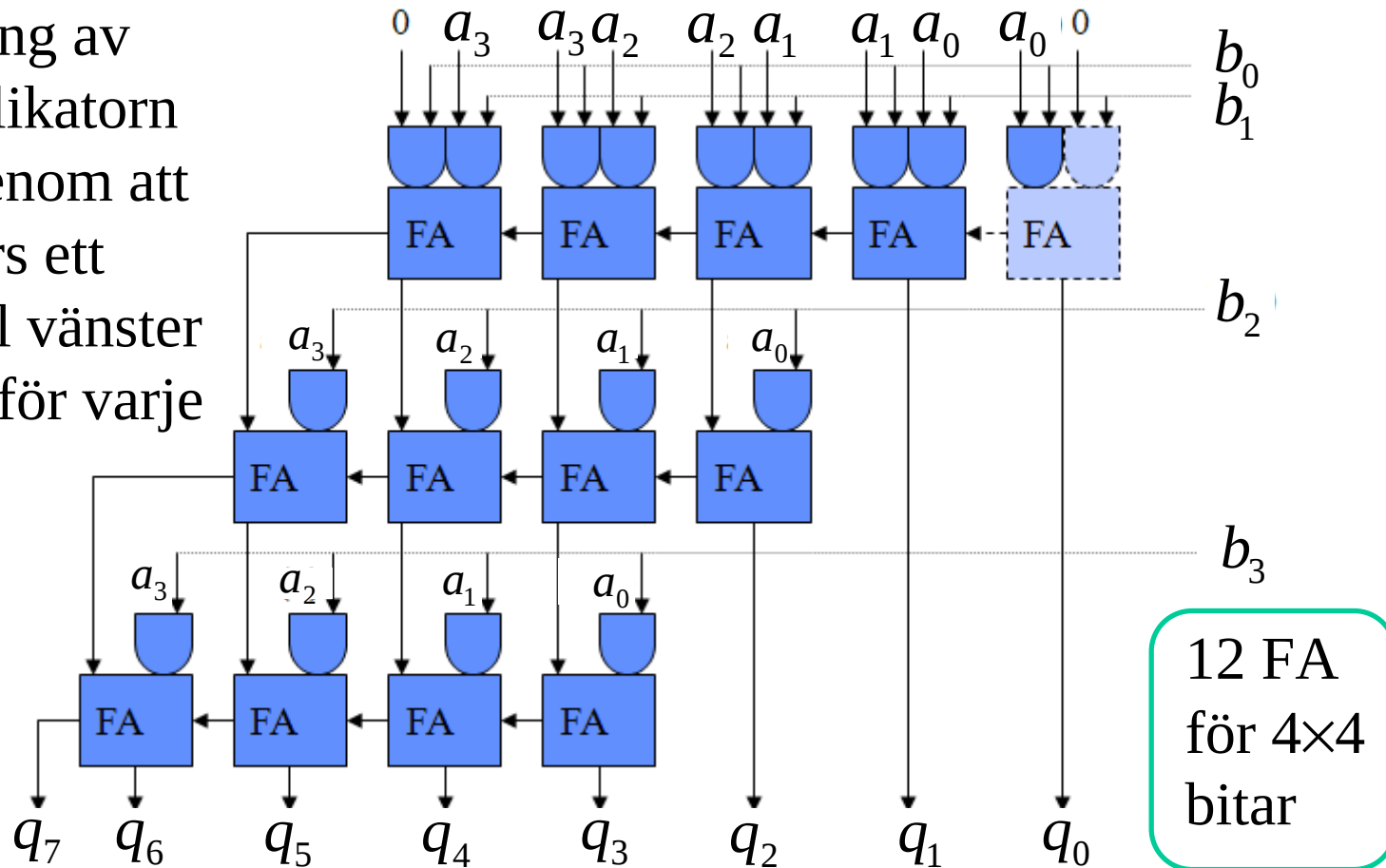
$$(a_3a_2a_1a_0) \cdot (\boxed{b_3}b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$



$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$

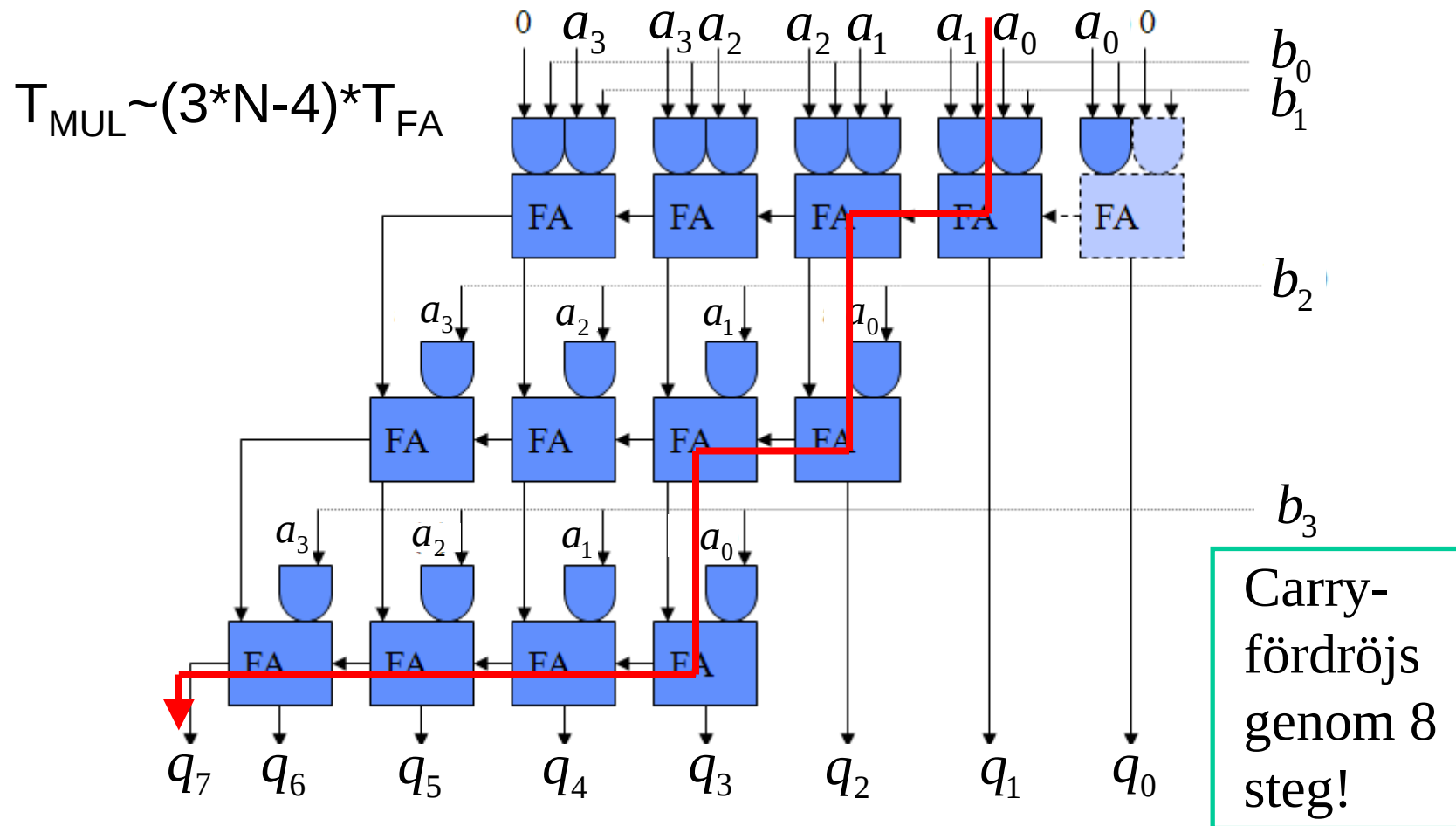
Multiplikation (två positiva tal)

Skiftning av multiplikatorn sker genom att den förs ett steg till vänster i nätet för varje nivå

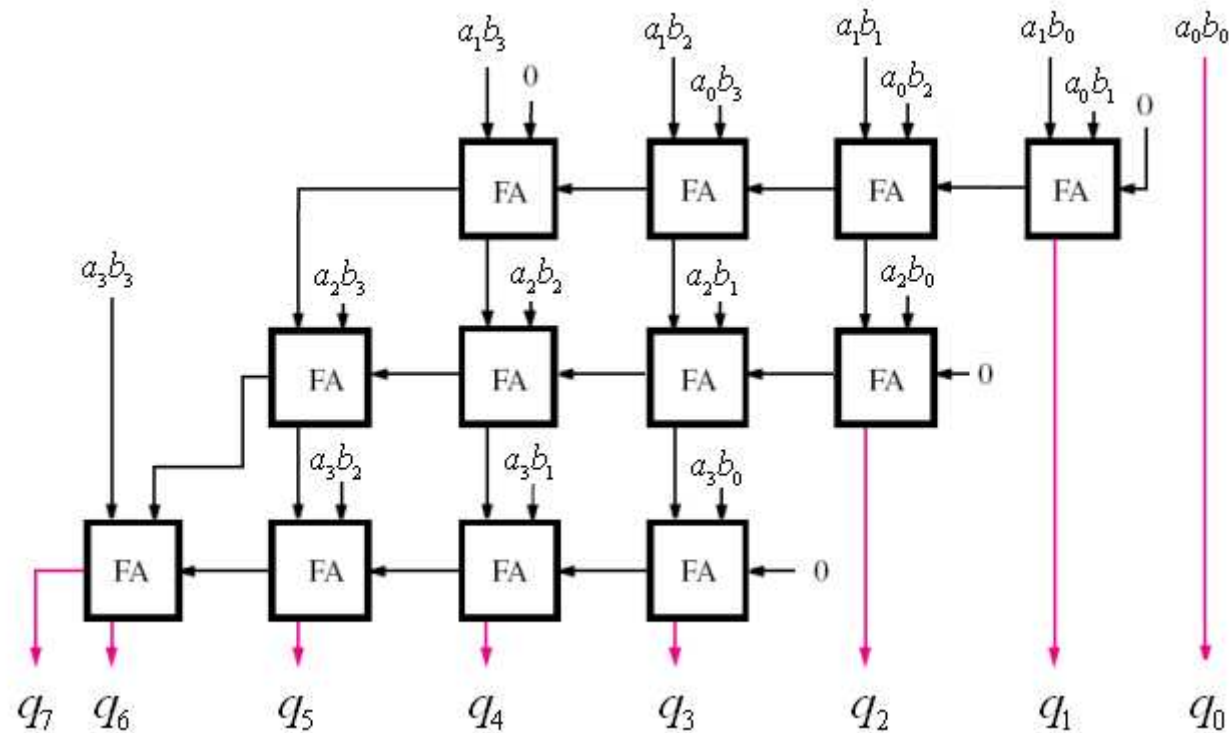


$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$

Multiplikation (två positiva tal)

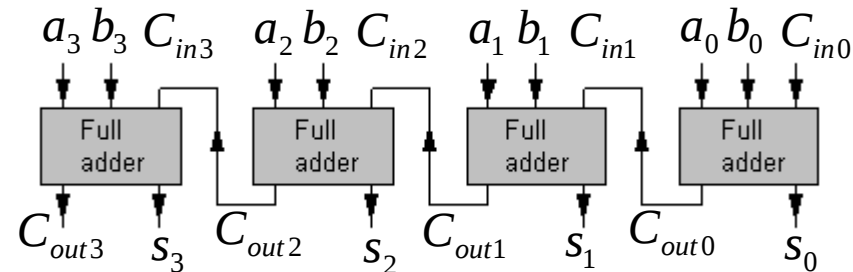


Kan bitarna adderas i
någon annan ordning?



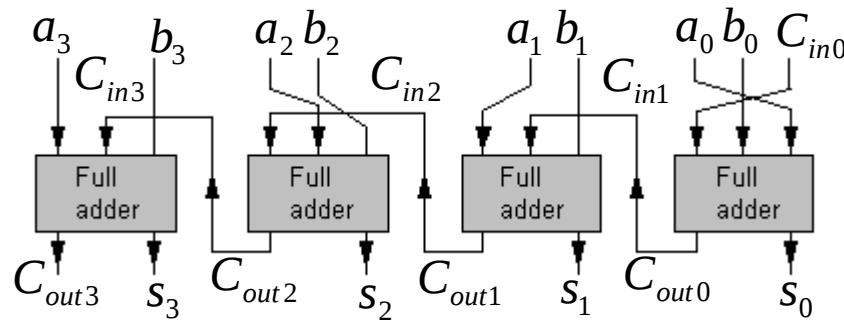
Snabbfråga

Ripple Carry-adderaren:



- Vad händer om man *förväxlar* ledningarna a b C_{in} ?

Här har någon
rört till det
ordentligt med
ledningarna!



a) Katastrof!

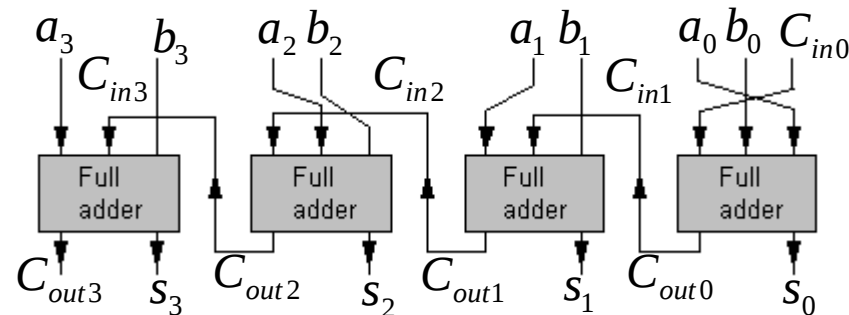
b) Don't care!

***Diskutera fram svaret tillsammans med din
bänkgranne!***

William Sandqvist william@kth.se



Snabbfråga



b) Don't care!

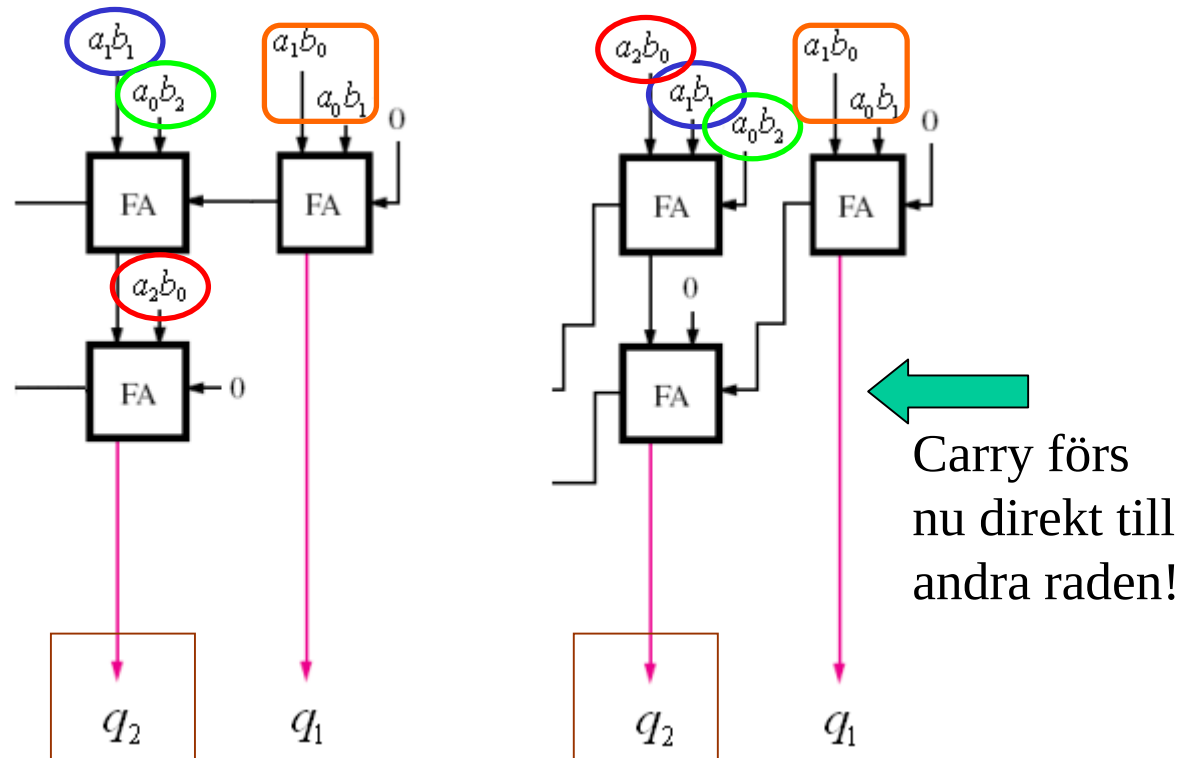
- Summabiten är lika med **"udda paritet"** av inbitarna.
- Carry out är lika med **"majoritetsvärdet"** av inbitarna.

I ingendera fallen har bitarnas inbördes ordning någon som helst betydelse.



Kan bitarna adderas i någon annan ordning?

Så här får q_2 samma resultat men med en annan bitordning!



$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$

En snabbare lösning - Carry-Save Multiplier (BV: sida 311)

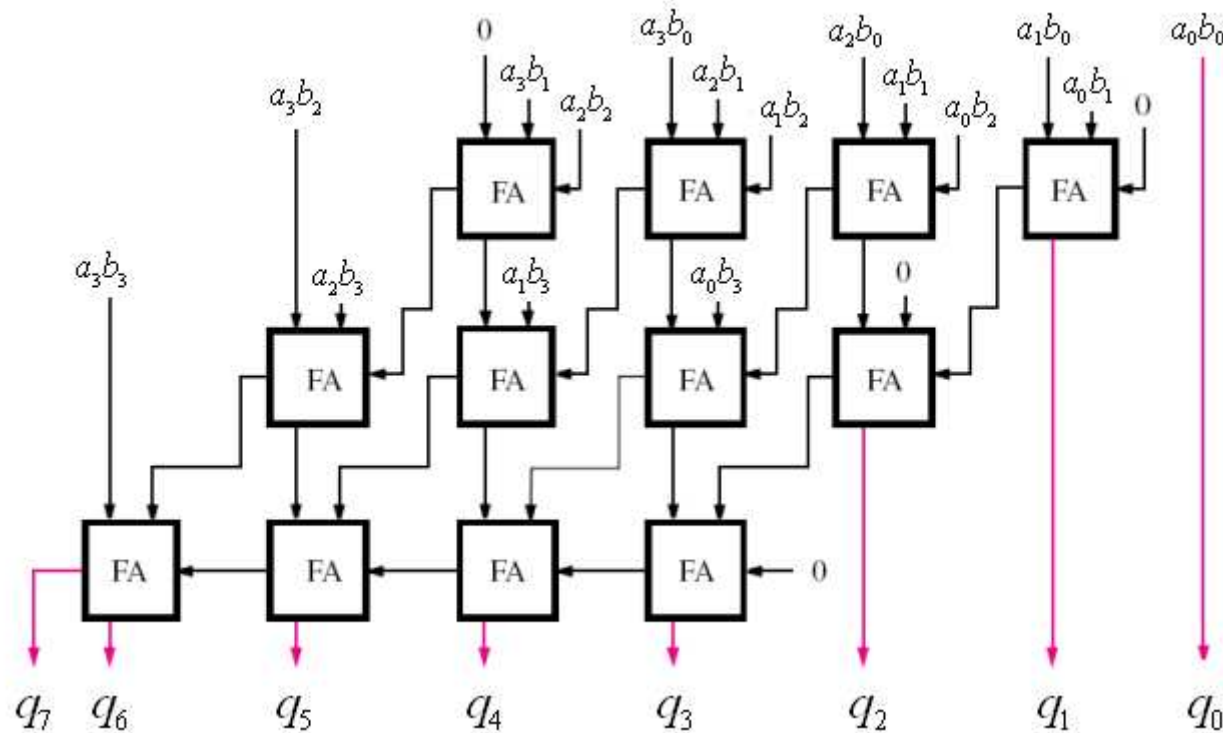


Figure 5.45 Multiplier carry-save array.

$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$

En snabbare lösning - Carry-Save Multiplier (BV: sida 311)

$$T_{\text{MUL}} \sim (2 \cdot N - 2) \cdot T_{\text{FA}}$$

Minskar fördröjningen eftersom carry utgången kopplas direkt till nästa steg!

Carry-fördröjs nu genom 6 steg!

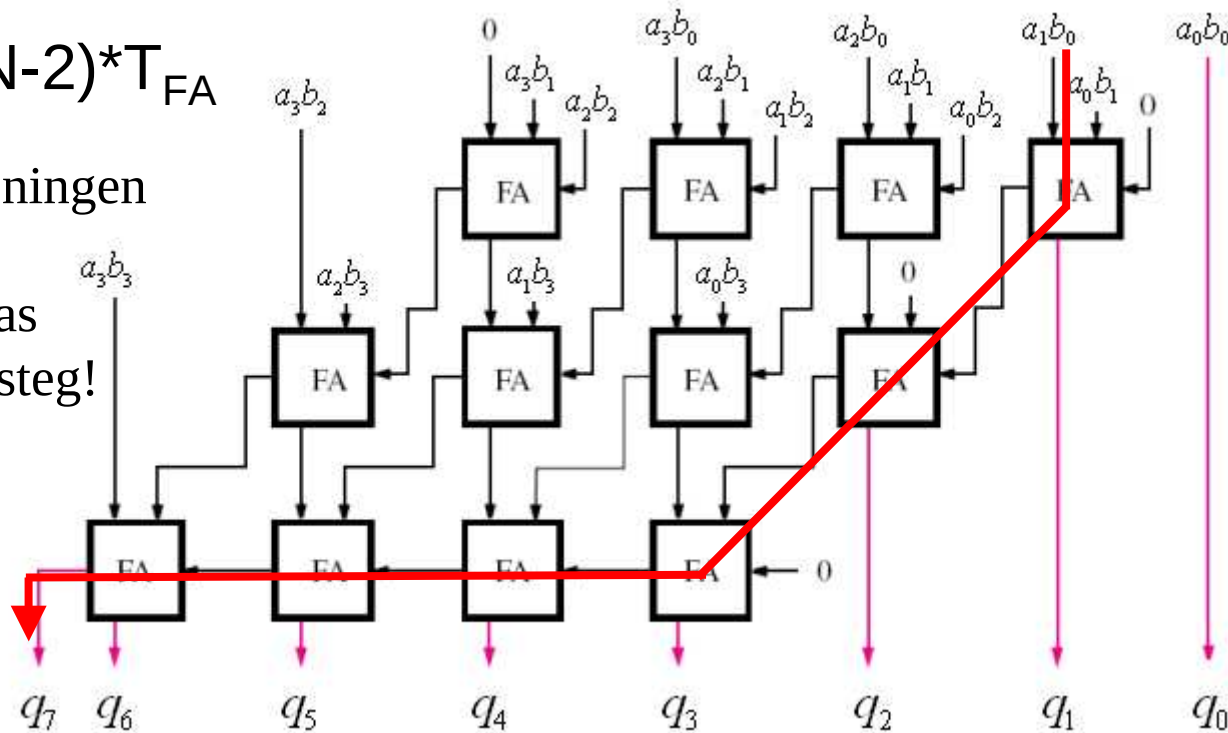


Figure 5.45 Multiplier carry-save array.

$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (q_7q_6q_5q_4q_3q_2q_1q_0)$$

En snabbare lösning - Carry-Save Multiplier (BV: sida 311)

Extra tips:
snabb Carry-look ahead adderare här!

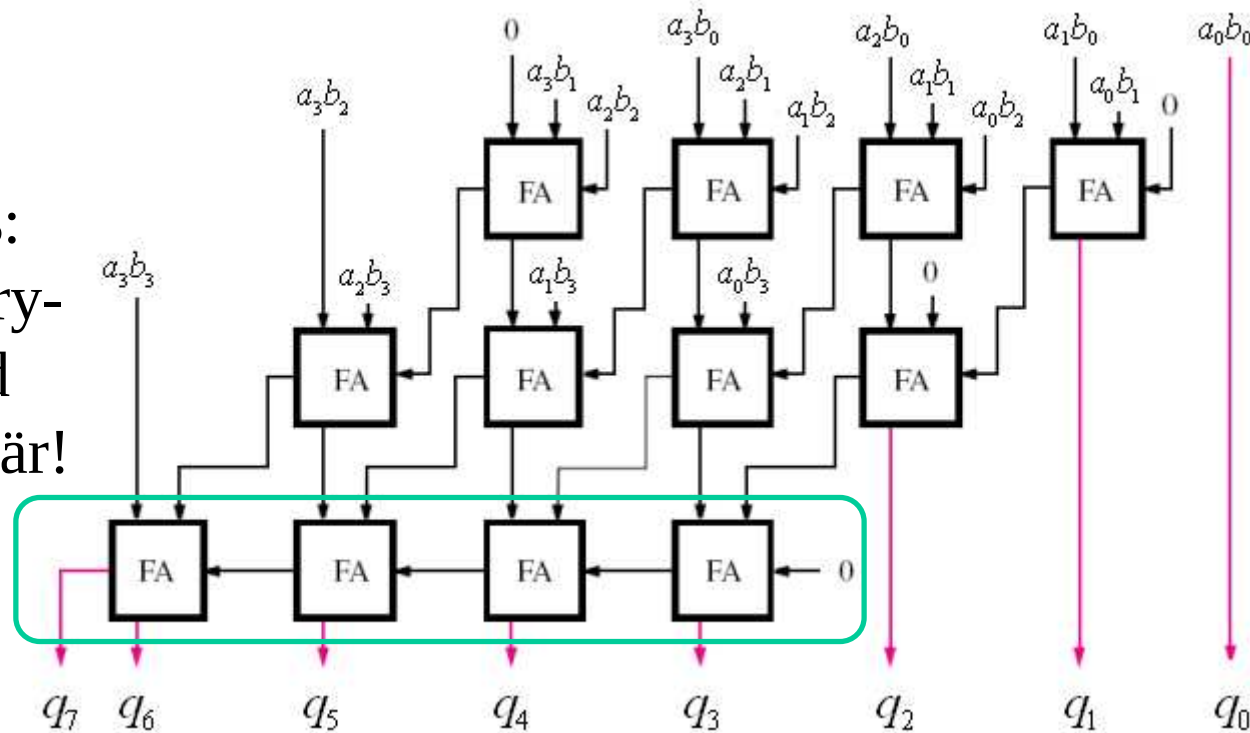


Figure 5.45 Multiplier carry-save array.

William Sandqvist william@kth.se

Multiplikation med två

$0101 * 2 = 1010$	$(5 * 2 = 10)$
$1010 * 2 = 10100$	$(-6 * 2 = -12)$
$01010101 * 2 = 010101010$	$(85 * 2 = 170)$
$10010101 * 2 = 100101010$	$(-107 * 2 = -214)$

jmfr multiplikation med 10 i basen 10:
 $63 * 10 = 630$, $-63 * 10 = -630$ etc.

Multiplikation med 2^n

$$0101 * 2 = 1010 \quad (5 * 2 = 10)$$

$$0101 * 2^2 = 10100 \quad (5 * 4 = 20)$$

$$0101 * 2^3 = 101000 \quad (5 * 8 = 40)$$

$$0101 * 2^4 = 1010000 \quad (5 * 16 = 80)$$

jmfr multiplikation med 10 i basen 10:

$$6 * 10 = 60, 6 * 100 = 600, 6 * 1000 = 6000 \text{ etc.}$$

Multiplikation med 2^n

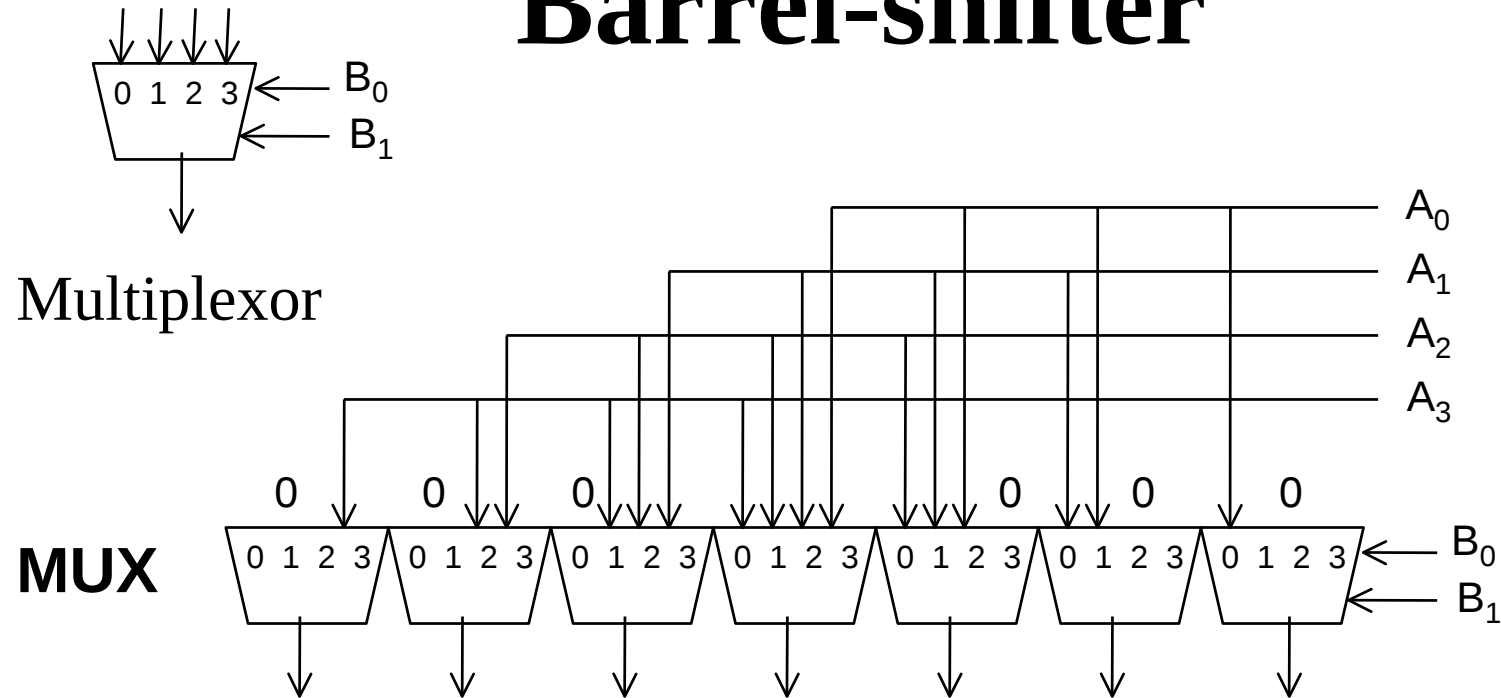
En multiplikation med 2^n kan göras med genom att skifta alla bitar n steg till vänster och att fylla på med nollor

13×8 kan beräknas genom att skifta (01011) tre bitar till höger

Resultat: 01011000 motsvarar $(104)_{10}$

Observera att man behöver flera bitar för att representera resultatet!

Barrel-shifter



Multiplikation med $(2^0, 2^1, 2^2, 2^3)$ – dvs 1, 2, 4, 8

$$(S_6 S_5 S_4 S_3 S_2 S_1 S_0) = (A_3 A_2 A_1 A_0) \cdot 2^{(B_1 B_0)}$$

Division mellan två positiva heltal (BV sid 693 fig 10.21 b)

$$\begin{array}{r}
 0101 \leftarrow 11/2 = 5 \\
 10 \overline{)1011} \\
 \underline{-10} \\
 001 \\
 \underline{-00} \\
 011 \\
 \underline{-10} \\
 1
 \end{array}$$

$\frac{a}{b} = q + \frac{r}{b}$
 Rest = 1

Lite mer detaljerat ... $\frac{a}{b} = q + \frac{r}{b}$

$$\begin{array}{r}
 0101 \\
 10 \overline{) 01011} \\
 \underline{- 00} \\
 10 \\
 \underline{- 10} \\
 001 \\
 \underline{- 00} \\
 011 \\
 \underline{- 10} \\
 1
 \end{array}$$

$11/2 = 5$
 Rest = 1

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

$a:$		0	1	0	1	1	$r:$
$b:$	1	0					$q:$

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

$a:$	0	1	0	1	1	$r:$
$b:$	-1	0				 $q: 0$

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

$a:$	0	1	0	1	1	$r:$	
$b:$	-1	0					$q: 0 \ 0$

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

$a:$	0	1	0	1	1	$r:$
$b:$	-1	0				✓ $q: 0 \ 0 \ 1$

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

$a:$	0	0	0	1	1	$r:$
$b:$			-1	0		$q: 0 \ 0 \ 1 \ 0$

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

$a:$	0	0	0	1	1	$r:$
$b:$				-1	0	$q: 0 \ 0 \ 1 \ 0 \ 1$

Lite mer detaljerat ...

$$\frac{a}{b} = q + \frac{r}{b} \quad \frac{11}{2} = 5 + \frac{1}{2}$$

0 0 0 0 1 → $r: 1$

$q: 0 0 1 0 1$

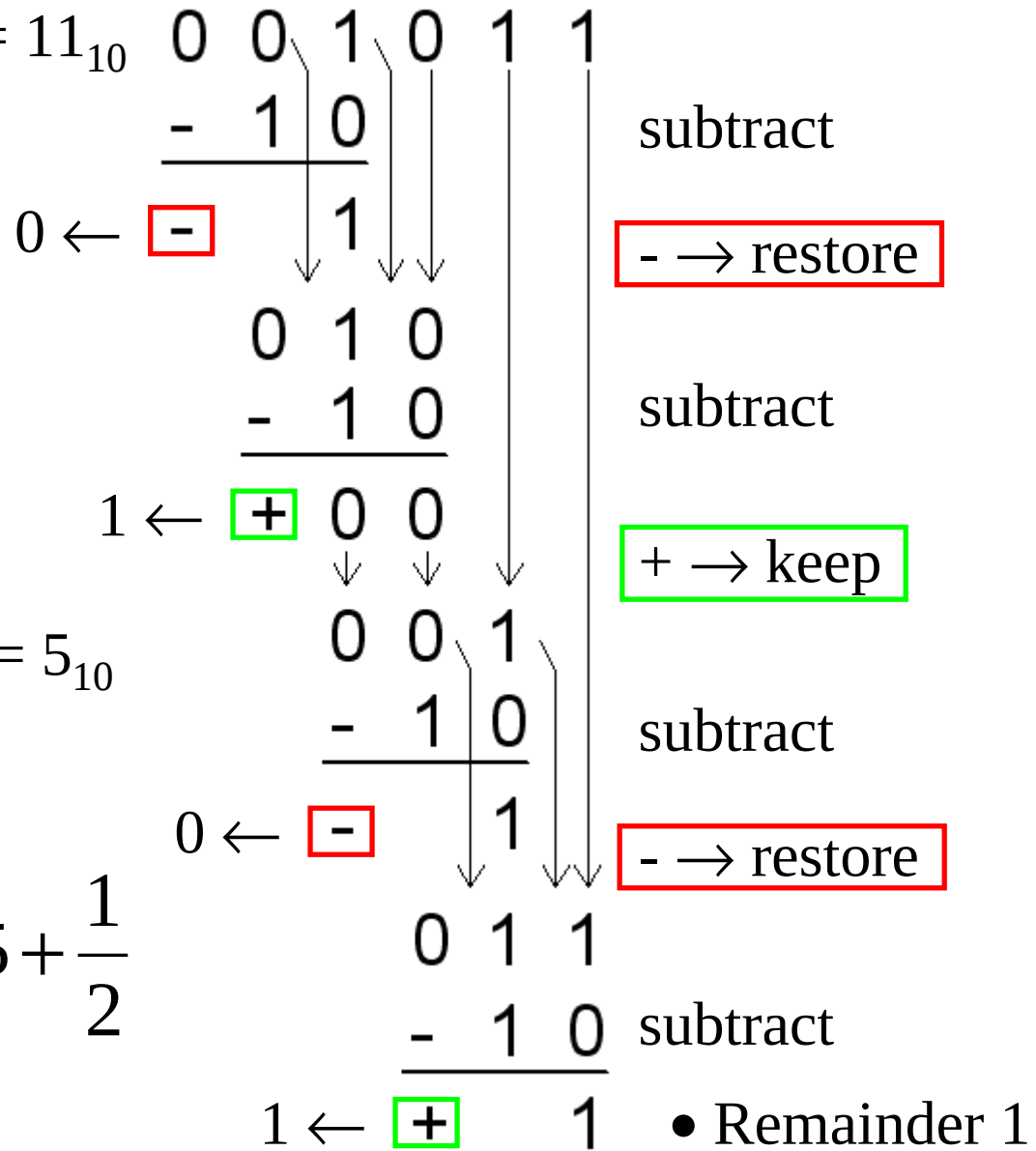
• Dividend $1011_2 = 11_{10}$

• Divisor $10_2 = 2_{10}$

• Quotient $0101_2 = 5_{10}$

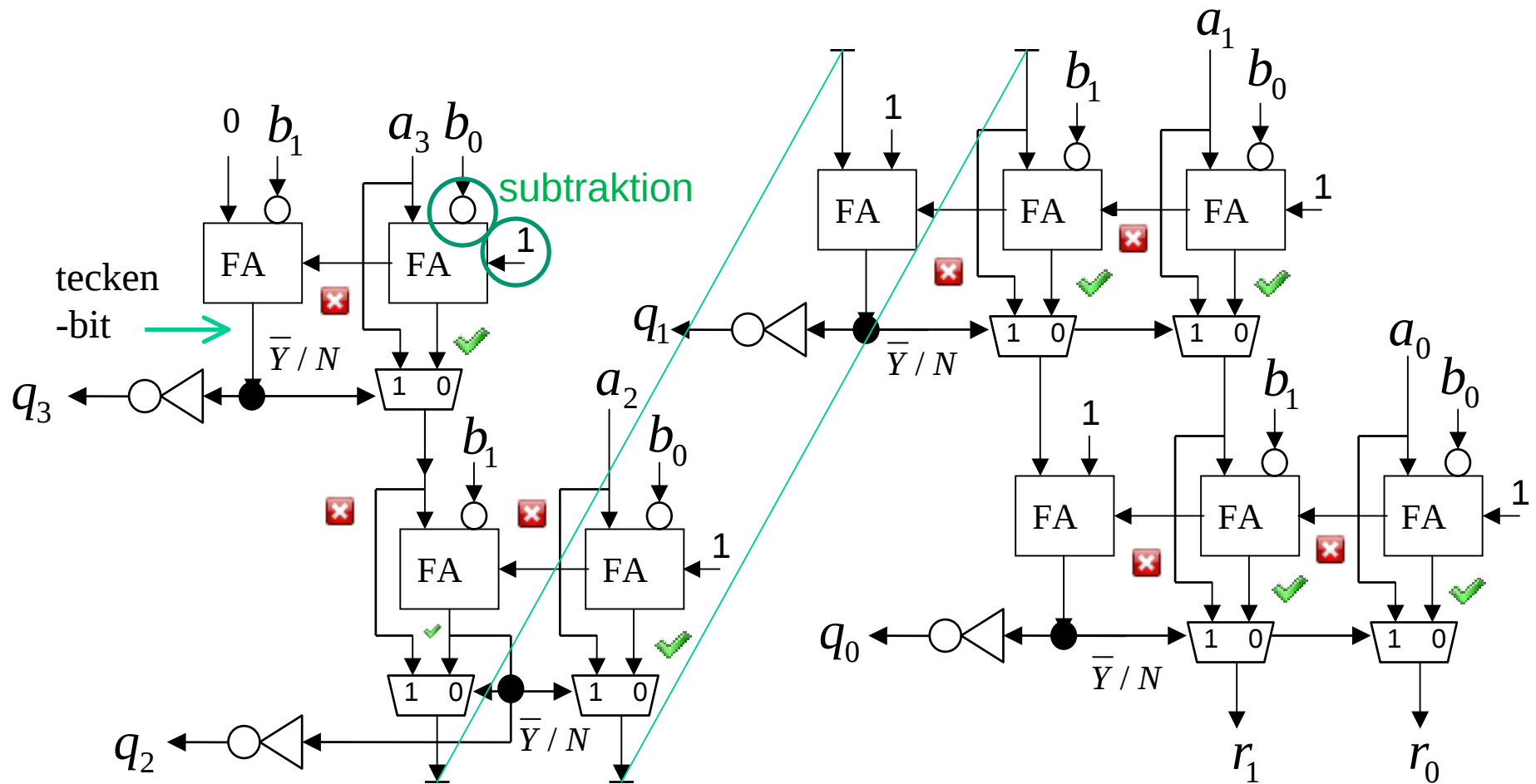
$$\frac{a}{b} = q + \frac{r}{b}$$

$$\frac{11}{2} = 5 + \frac{1}{2}$$

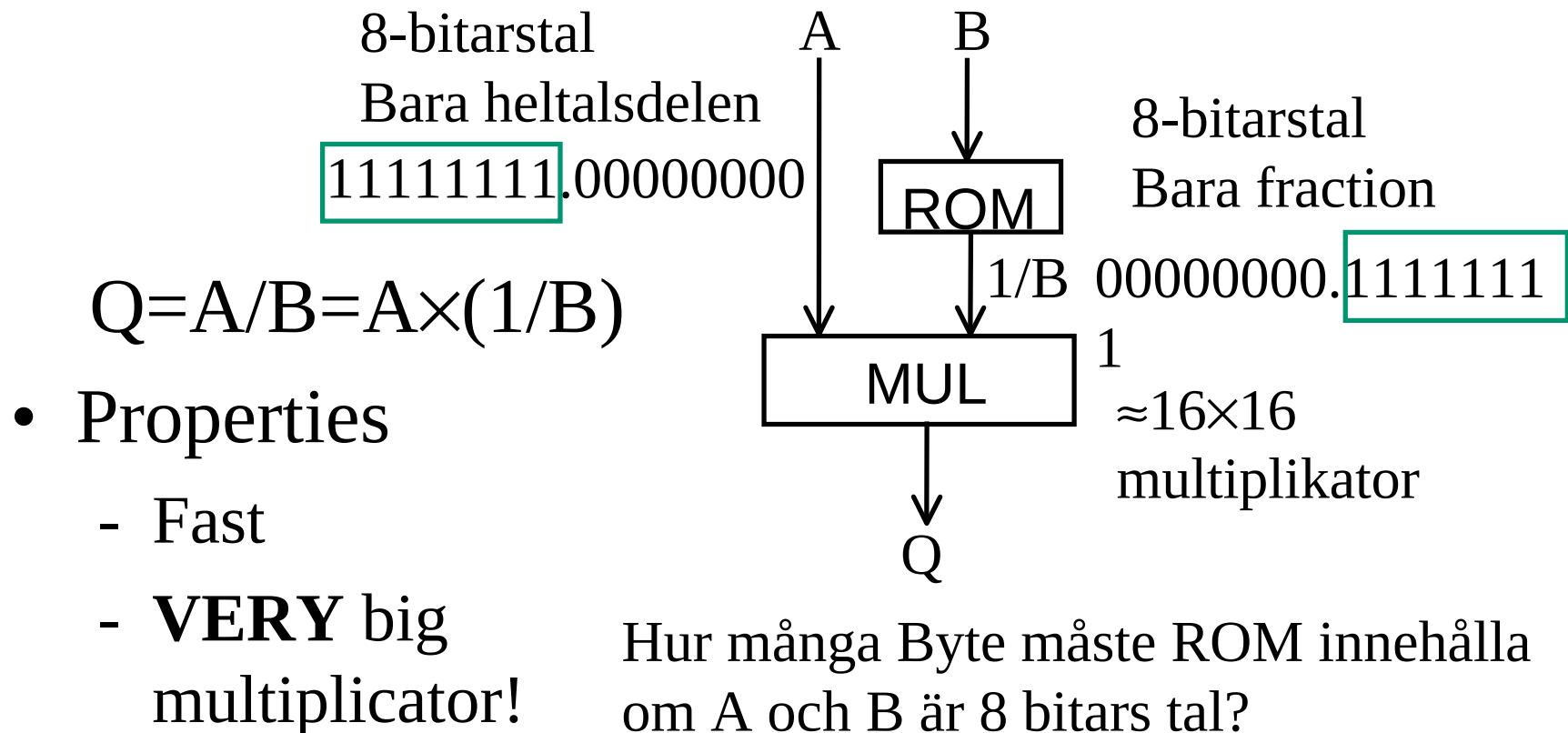


Divideraren

$$\frac{a_3 a_2 a_1 a_0}{b_1 b_0} = q_3 q_1 q_1 q_0 + \frac{r_1 r_0}{b_1 b_0}$$

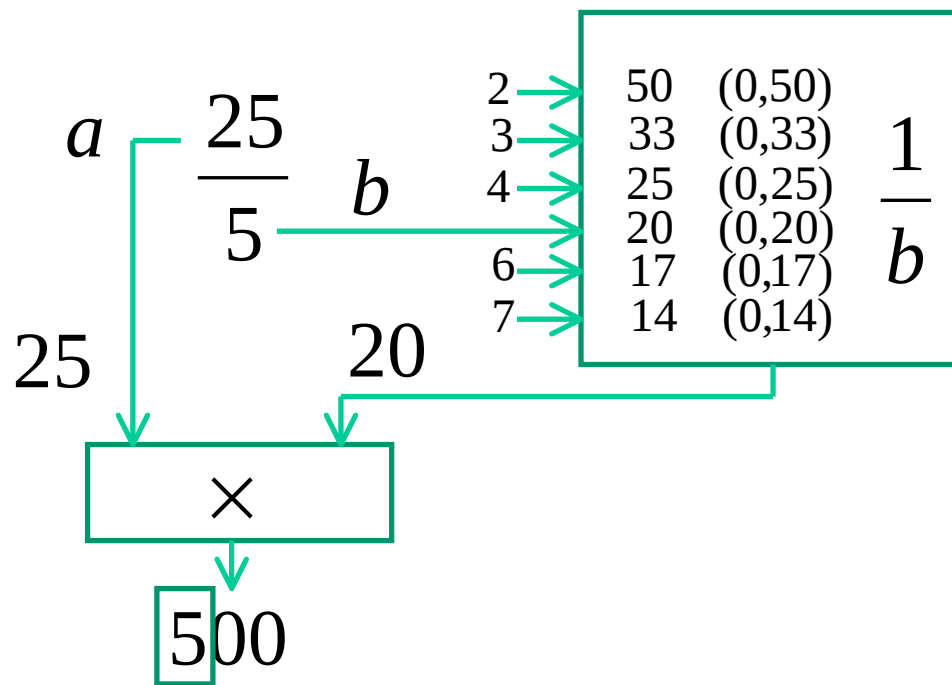


Using a multiplier and a ROM for division



Using a multiplier and a ROM for division

$$\frac{a}{b} = \frac{25}{5} = 25 \times \frac{1}{5} = 25,0 \cdot 0,2 = 5,00$$



Division med negativa heltal

Division med negativa tal är ganska knepigt.

Ett sätt att utföra divisionen ändå

Konvertera till positiva tal

Håll reda på resultatets tecken

$$(+) \cdot (+) \Rightarrow (+) \quad (+) \cdot (-) \Rightarrow (-)$$

$$(-) \cdot (+) \Rightarrow (-) \quad (-) \cdot (-) \Rightarrow (+)$$

Två-komplementera till negativt tal om nödvändigt

Division med 2

$$\begin{array}{ll} 0 \rightarrow 01010/2 = 00101 & (10/2 = 5) \\ 1 \rightarrow 10100/2 = 11010 & (-12/2 = -6) \end{array}$$

jmfr division med 10 i basen 10:
 $630/10 = 63$, $-630/10 = -63$ etc.

Logiskt och Aritmetisk shift höger

Man skiljer mellan logisk och aritmetiskt shift

***Logisk shift* höger shiftar bara till höger. Bitarna skall ej tolkas som ett tal. Man fyller bara på med 0:or.**

***Aritmetisk shift* höger behandlar bitarna som ett tal. Teckenbiten behålls vid shift. Man fyller på till vänster med teckenbiten.**

Division med 2^n

$1010/2 = 101$	$(10/2=5)$
$10100/2^2 = 101$	$(20/4=5)$
$101000/2^3 = 101$	$(40/8=5)$
$1010000/2^4 = 101$	$(80/16=5)$

jmfr division med 10 i basen 10:

$$60/10 = 6, 600/100 = 6, 6000/1000 = 6 \text{ etc.}$$

Division med 2^n

En division med 2^n kan göras med genom att skifta alla bitar n steg till höger och att fylla på med nollor

Observera att resultatet behöver *inte* bli korrekt, eftersom man egentligen behöver bitar ”efter kommatecknet”

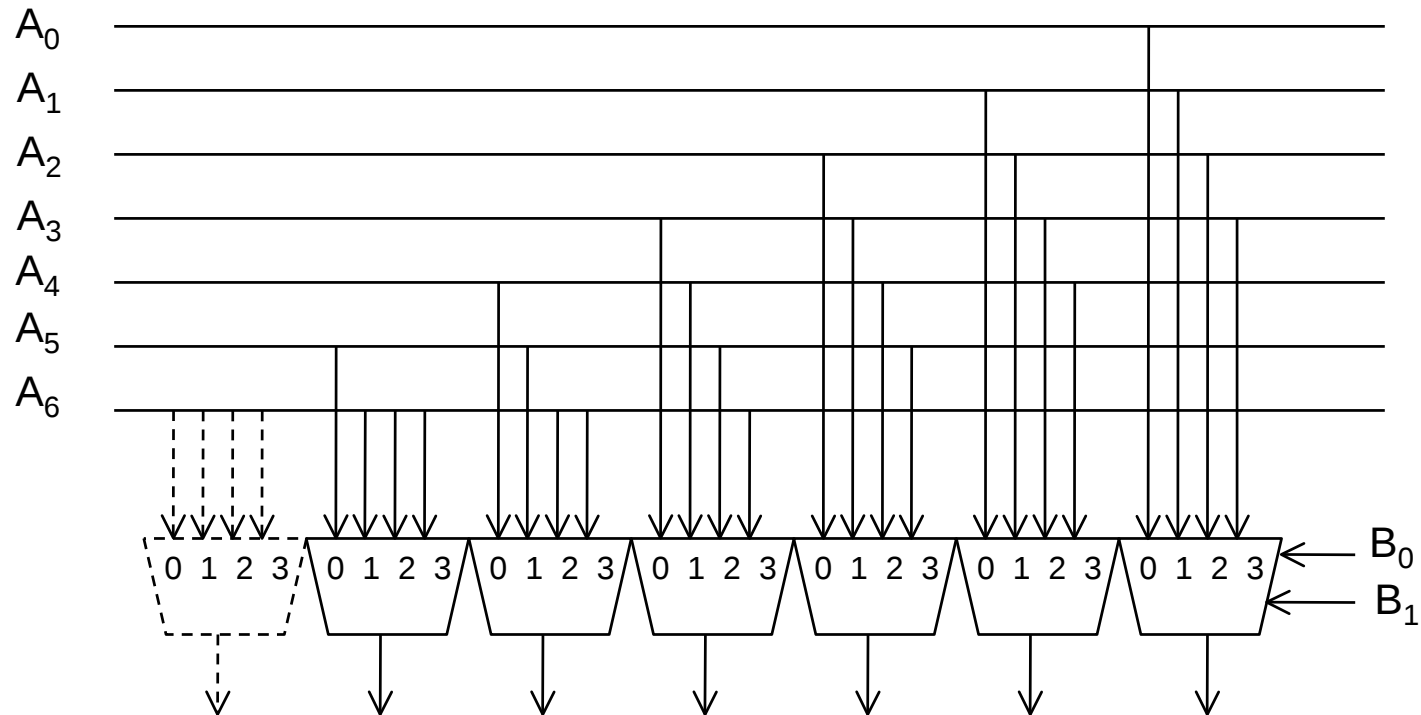
17/4 motsvarar att skifta 010001 2-bitar till höger

Resultat: $000100 = (4)_{10}$

Eftersom $(0.25)_{10}$ inte kan representeras är resultatet inte korrekt!

William Sandqvist william@kth.se

Barrel-shifter



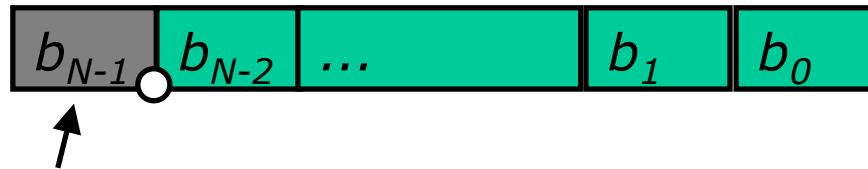
Division med $(2^0, 2^1, 2^2, 2^3)$ – dvs 1, 2, 4, 8

$$(S_6 S_5 S_4 S_3 S_2 S_1 S_0) = (A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0) / 2^{(B_1 B_0)}$$

Fix-tal (Fixed-point numbers)

Två-komplementsrepresentation

$$B = b_{N-1} . b_{N-2} \dots b_1 b_0 \quad \text{where } b_i \in \{0, 1\}$$



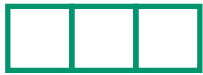
Sign Bit

Decimalt värde

$$FiP(B) = -b_{N-1} 2^0 + b_{N-2} 2^{-1} + \dots + b_1 2^{-(N-2)} + b_0 2^{-(N-1)}$$

Detta format kallas också för Q_{N-1} -format eller *fractional representation*

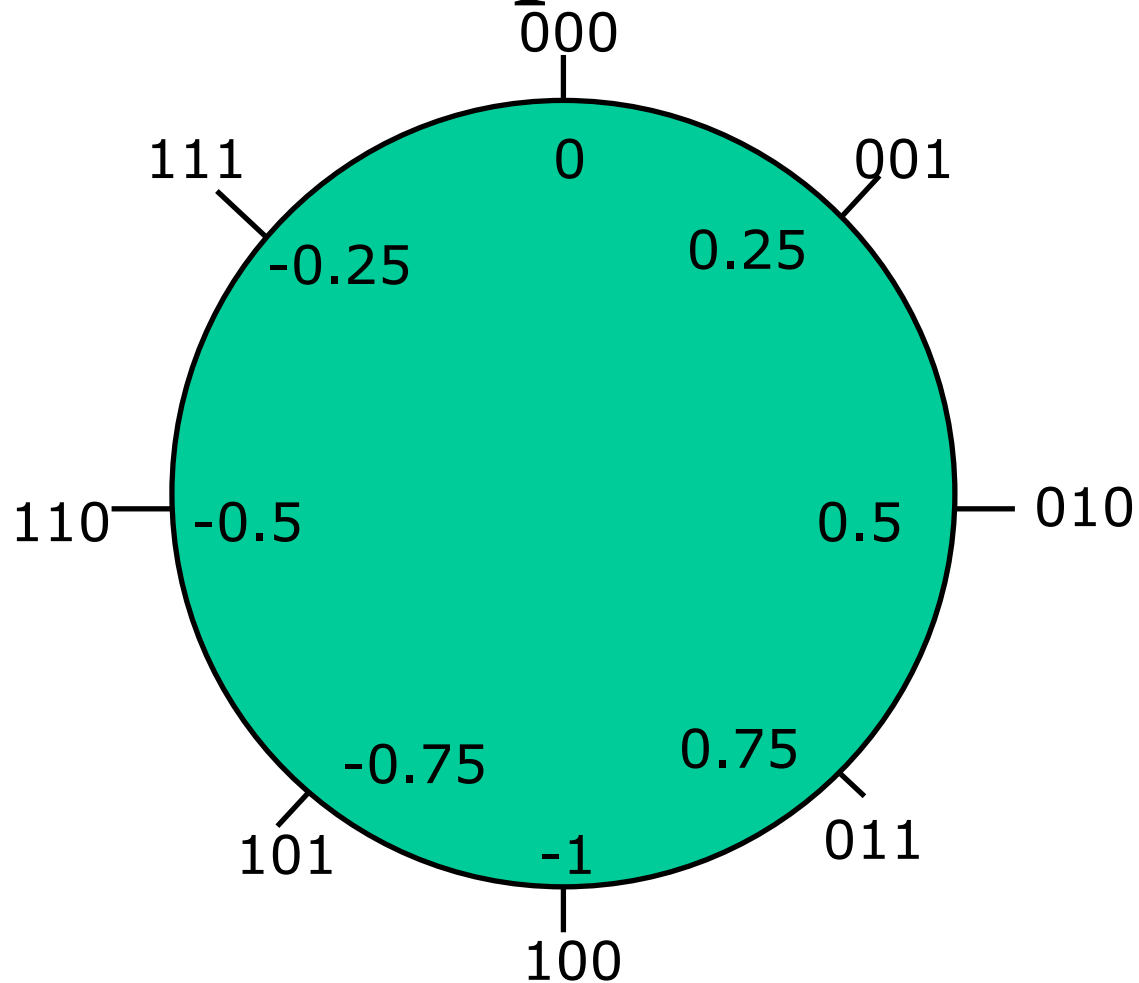
Fixed-Point Q2-Representation



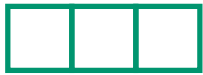
Se detta som ett
pedagogiskt
exempel:

**Tre-bitars
fixpunktstal**

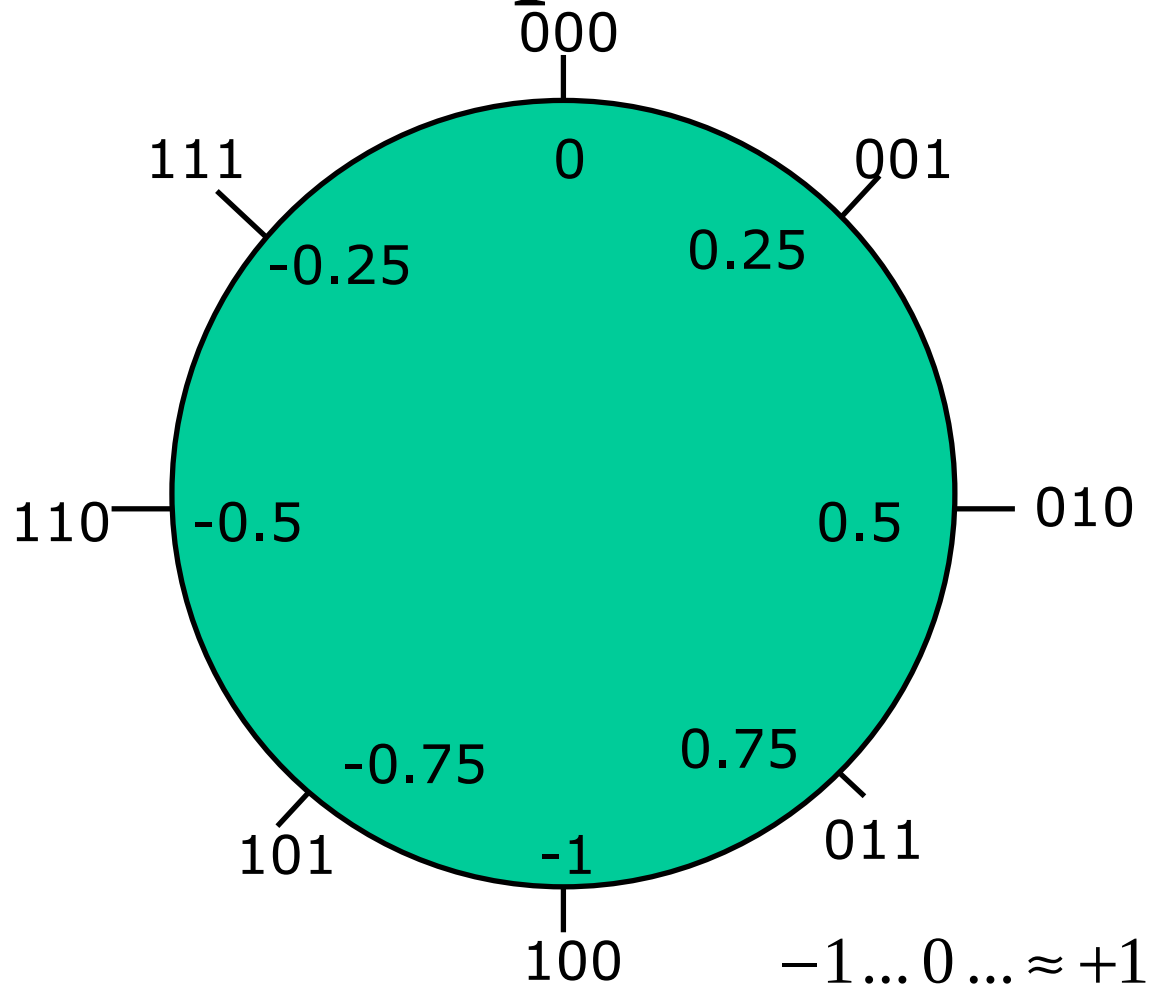
Ett talsystem
med bara 8 tal
är ju i praktiken
oanvändbart!



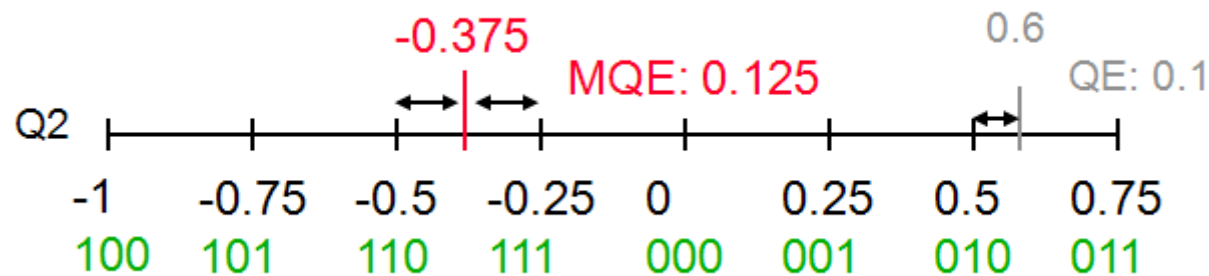
Fixed-Point Q2-Representation



0.11	0,75
0.10	0,5
0.01	0,25
0.00	0
1.11	-0,25
1.10	-0,5
1.01	-0,75
1.00	-1.0



Maximalt kvantiseringsfel



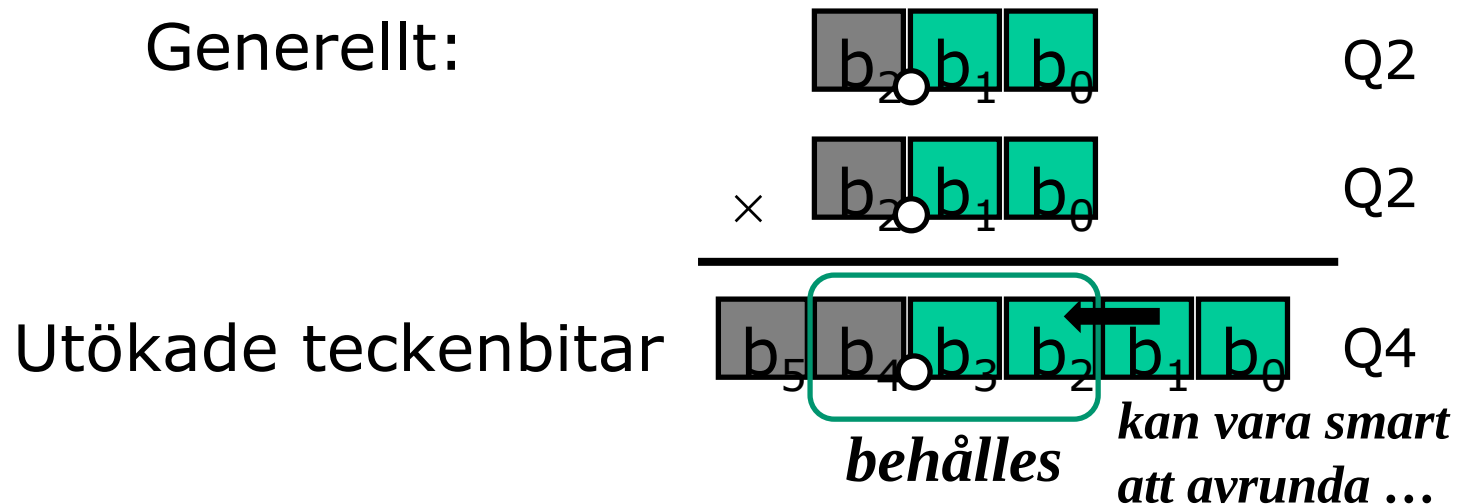
- Maximum Quantization Error:
 - Since not all numbers can be represented, quantization errors occur.

Multiplikation i Q-formatet

Multiplikation orsakar inte overflow i Q-formatet, men kan resultera i *förlust av precision*

$$\begin{aligned} & (1.10)_2 \times (0.11)_2 \\ &= (1.1010)_2 \quad (\text{Q4-format}) \\ &= (1.10)_2 \quad (\text{Q2-format}) \end{aligned}$$

Generellt:



Q-formatet

**Fixpunktsmultiplikation och
fixpunktsdivision utnyttjar samma
hårdvara som
heltalsmultiplikation och
heltalsdivision.**

Användning: Digitala filter – tex. ljudkortet

William Sandqvist william@kth.se

Flyt-tal (eng. Floating-Point Numbers)

- Ett flyt-tal representeras med en *tecken-bit*, *exponent-bitar* och en mantissa (*fraktions-bitar*)

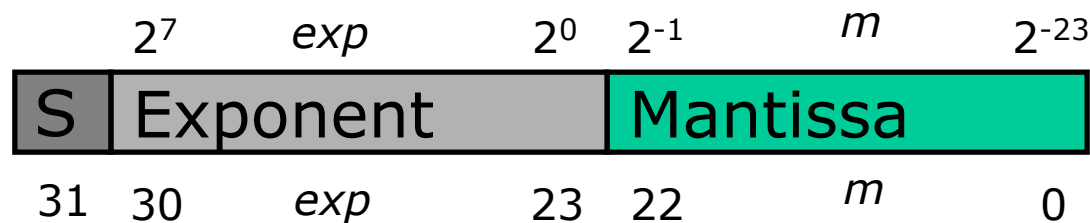


- Värdet beräknas som
$$FIP(B) = (-1)^s * (1.m) * 2^{\pm exp}$$
- Ofta är *exp* **biaserad** (har en **offset**), vilket då ger

$$FIP(B) = (-1)^s * (1.m) * 2^{exp-(bias)}$$

IEEE-754

- Flyttals-standarden IEEE-754 definierar ett 32-bit flyttal som



- Värdet beräknas för en 8-bitars exponent enligt nedan
$$FIP(B) = (-1)^s * (1.m) * 2^{exp-(127)}$$
- Speciala bit pattern har reserverats för att representera negativ och positiv nolla

Floating-Point Numbers (IEEE)

- Exponent Values 1 to 254: normalized non-zero floating-point numbers; biased exponent (-126...+127)
- Exponent of zero and fraction of zero: positive or negative zero
- Exponent of ones and fraction of zero: positive or negative infinity
- Exponent of zero and fraction of non-zero: Denormalized number (true exponent is -126), represent numbers from 0 to 2^{-126}

$$FIP(B) = (-1)^s * (\mathbf{0}.m) * 2^{-126}$$

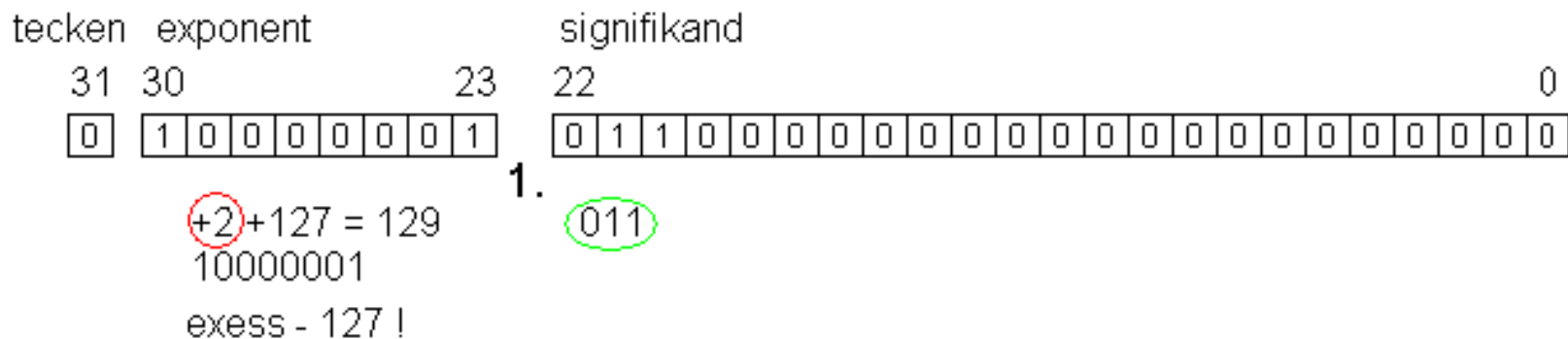
- Exponent of ones with a non-zero fraction: **NotANumber** (Exception Condition)
- There is also a standard for a 64-bit number

IEEE – 32 bit float

$$+5,5_{10} = +\overbrace{10}^5 \overbrace{1.1}^{1/2}_2$$

Normaliserat: $+1.011 \cdot 2^{+2}$

Tecken 1 bit, exponent 8 bitar, signifikand (mantissa) 23 bitar.



Genom att exponenteten skrivs excess–127 kan flyttal storlekssorteras med vanlig heltalsaritmetik!

Behandlas igen i Datortekniken ...

Decimalt additionsexempel

$$\begin{array}{ll} & \text{normalized} \quad \text{aligned} \\ a = 123456.7 & = 1.234567 \cdot 10^5 \\ b = 101.7654 & = 1.017654 \cdot 10^2 = 0.001017654 \cdot 10^5 \end{array}$$

Det **tal som är minst** (här **b**) skiftas (align) så båda talen får **samma** exponent.

$$\begin{array}{r} c = a + b \\ 1.234567 \cdot 10^5 \\ + 0.001017654 \cdot 10^5 \\ \hline 1.235584654 \cdot 10^5 \end{array}$$

Svaret kan behöva normaliseras (skiftas).

*Flyttalsoperationer är mycket krävande, om man inte har specialiserad hårdvara - **addition** kan vara mer krävande än multiplikation!*

Addition med flyttal

- Givet två flyttal:

$$a = a_{frac} \cdot 2^{a_{exp}}$$

$$b = b_{frac} \cdot 2^{b_{exp}}$$

- Summan av dessa tal är:

$$c = a + b$$

*Det tal som är
minst skiftas*

$$= \begin{cases} (a_{frac} + (b_{frac} \cdot 2^{-(a_{exp} - b_{exp})})) * 2^{a_{exp}} & , \text{if } a_{exp} \geq b_{exp} \\ (b_{frac} + (a_{frac} \cdot 2^{-(b_{exp} - a_{exp})})) * 2^{b_{exp}} & , \text{if } b_{exp} \geq a_{exp} \end{cases}$$

Subtraktion med flyttal

- Givet två flyttal:

$$a = a_{frac} \cdot 2^{a_{exp}}$$

$$b = b_{frac} \cdot 2^{b_{exp}}$$

- Differensen mellan dessa tal är:

$$c = a - b$$

*Det tal som är
minst skiftas*

$$= \begin{cases} (a_{frac} - (b_{frac} \cdot 2^{-(a_{exp} - b_{exp})})) * 2^{a_{exp}} & , \text{if } a_{exp} \geq b_{exp} \\ (b_{frac} - (a_{frac} \cdot 2^{-(b_{exp} - a_{exp})})) * 2^{b_{exp}} & , \text{if } b_{exp} \geq a_{exp} \end{cases}$$

Decimalt multiplikationsexempel

$$c = a \cdot b$$

$$a = 4,734612 \cdot 10^3 \quad b = 5,417242 \cdot 10^5$$

*Resultatet har fler
siffror än vad som
ryms – avrunda.*

$$c = 4,734612 \cdot 5,417242 \cdot 10^{3+5} = 25,648538980104 \cdot 10^8$$

$$c = 2,564854 \cdot 10^9 \quad \text{normalisera}$$

Multiplikation innebär att man gör en addition av exponenterna, och en multiplikation med fraktionsdelarna. Svaret måste sedan normaliseras (skiftas).

Multiplikation med flyttal

- Givet två flyttal:

$$a = a_{frac} \cdot 2^{a_{exp}}$$

$$b = b_{frac} \cdot 2^{b_{exp}}$$

- Produkten av dessa tal är:

Enklare!

$$c = a * b$$

$$= \left(a_{frac} * b_{frac} \cdot 2^{a_{exp} + b_{exp}} \right)$$

Division med flyttal

- Givet två flyttal:

$$a = a_{frac} \cdot 2^{a_{exp}}$$

$$b = b_{frac} \cdot 2^{b_{exp}}$$

- Kvoten mellan dessa tal är:

$$c = a / b$$

$$= \left(a_{frac} / b_{frac} \cdot 2^{a_{exp} - b_{exp}} \right)$$

Uppstädning efter flyttals-operationer...

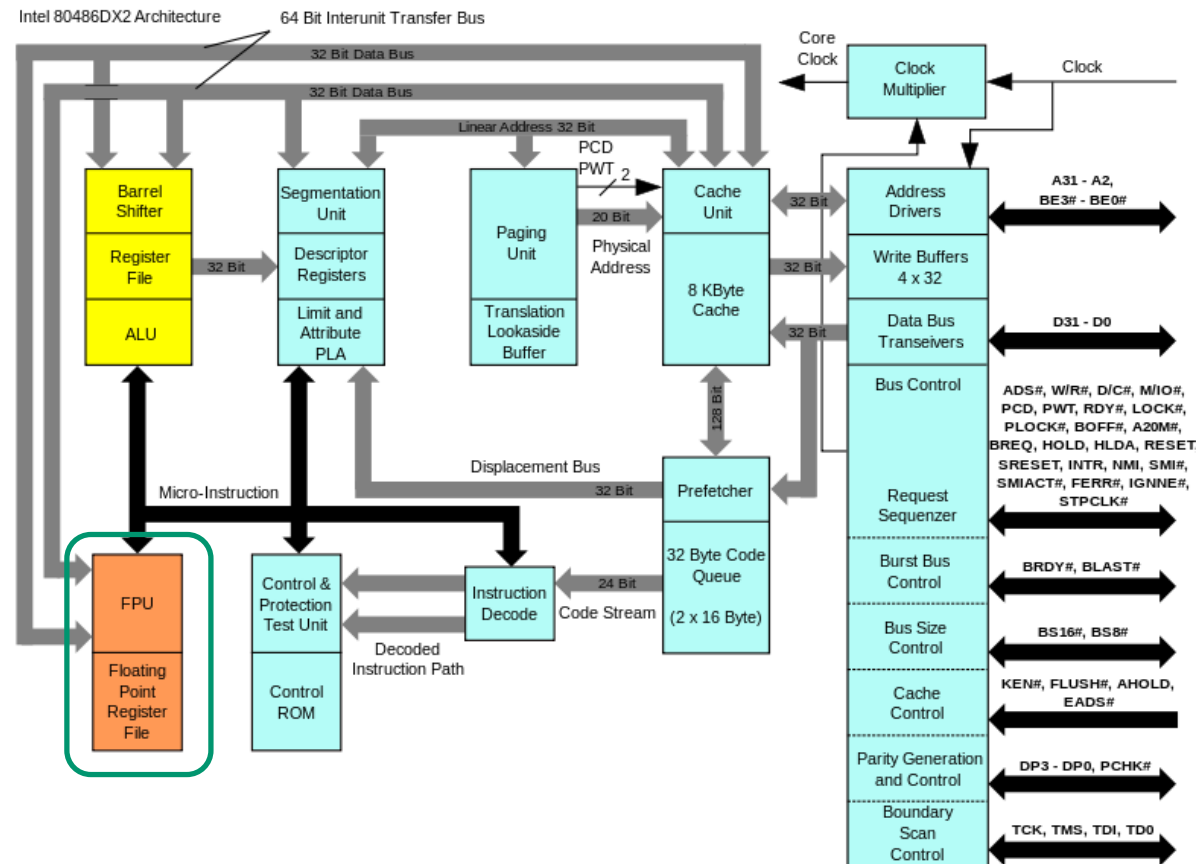
- När en flyttals-operation är klar måste den normaliseras
 - Mantissans skiftas tills dess första bit är 1
 - För varje skift-steg så räknas exponenten upp eller ned med ett.
 - Mantissans bitar till höger om den första ettan sparas
- Om exponenten är noll är mantissans första bit 0

$$FLP(B) = (-1)^s * (1.m) * 2^{exp-(127)}$$

$$FLP(B) = (-1)^s * (0.m) * 2^{-(126)}$$

Flyttalsenhet

Det krävs mycket kod och beräkningstid för att utföra flyttalsoperationer med en dator som saknar hårdvarustöd för detta. PC-datorerna har haft inbyggda flyttalsenheter från och med 486 (1989).



William Sandqvist william@kth.se

Dyraste mjukvarubuggen?

[ESA rocket crashes at launch - 1996](#)

double (64-bitars flyttal)

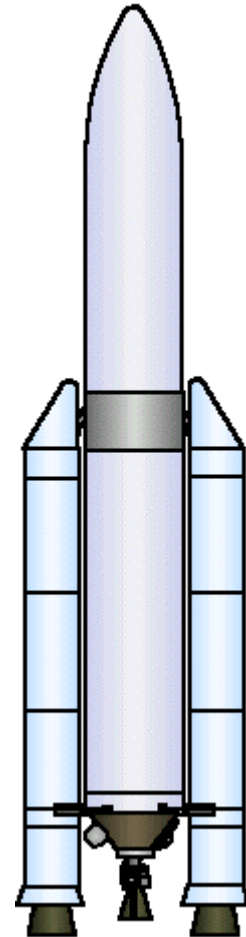


Översättningen blev fel –
rymdes inte!

signed short int (16-bitars 2-komplementtal)

Användes i systemet för horisontel bias

Någon hade sovit under föreläsningen ”aritmetik2”!



Fixed-Point vs. Floating-Point

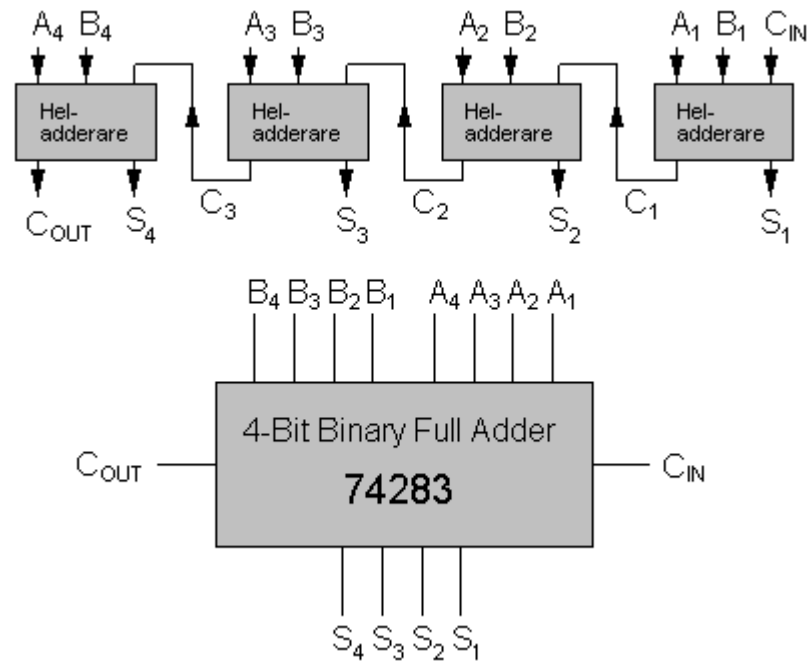
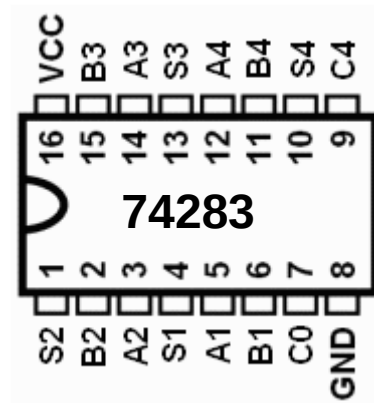
- Fixed-Point operationer fungerar pss som heltals-operations och är snabbare
- Fixed-point värden behöver skalas, vilket ofta leder till förlust av precision
- Kostnaden för att bygga hårdvara är signifikant större för flyttals-processorer/räknare

Sammanfattning

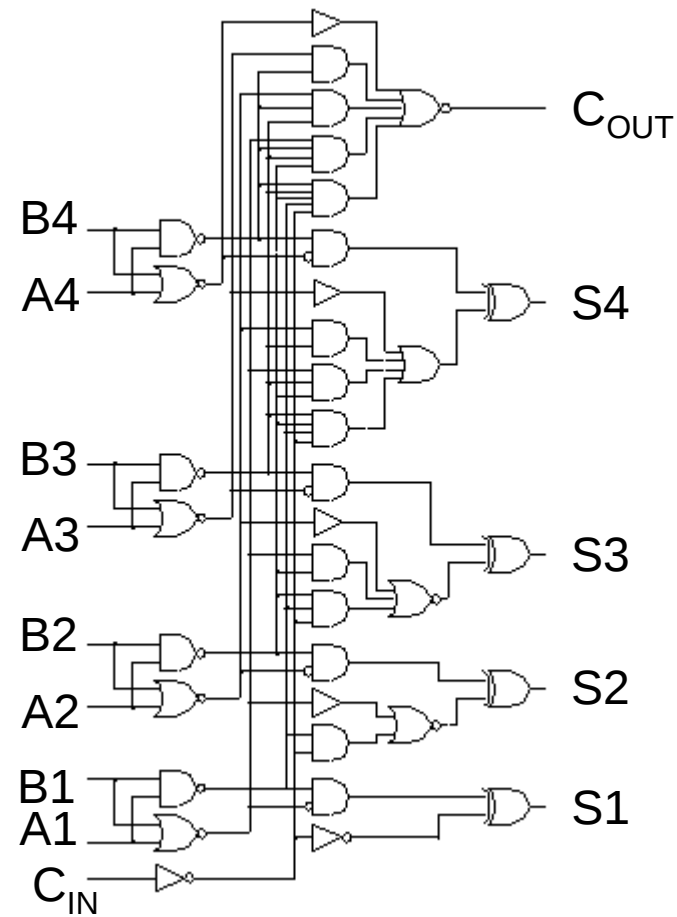
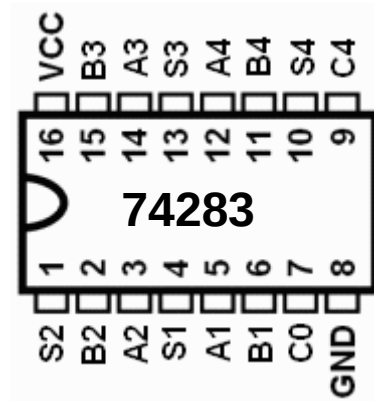
- **Multiplikation och division av heltal**
 - Konvertera negativa tal till sitt positiva dito.
 - Utför multiplikationen eller divisionen
 - Håll reda på vilket tecken resultatet skall ha
 - Konvertera positivt resultat till sitt negativa dito om resultatet skall vara negativt
- **Multiplikation med potenser av 2 (mul med 2^k)**
 - Implementeras som ett skift till vänster med k steg
- **Division med potenser av 2 (div med 2^k)**
 - Implementeras som ett (aritmetiskt) skift till höger med k steg. Teckenbiten kopieras till vänster.

William Sandqvist william@kth.se

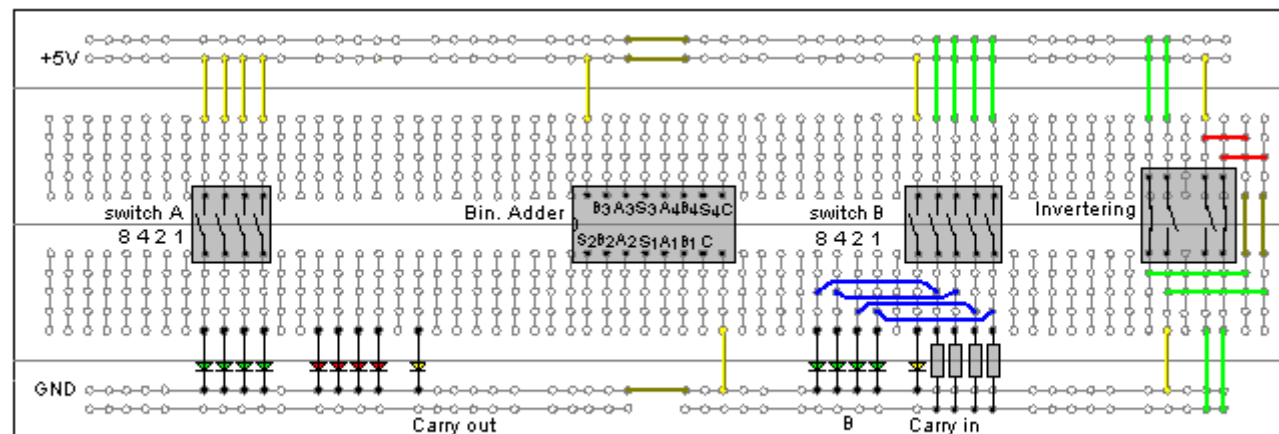
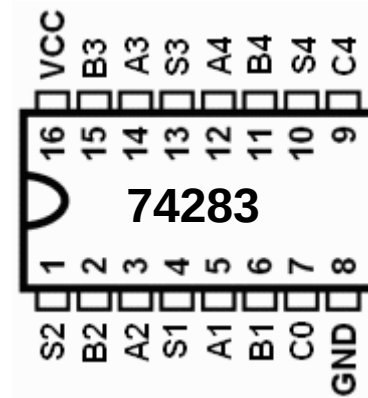
Addition vid Lab1



Addition vid Lab1

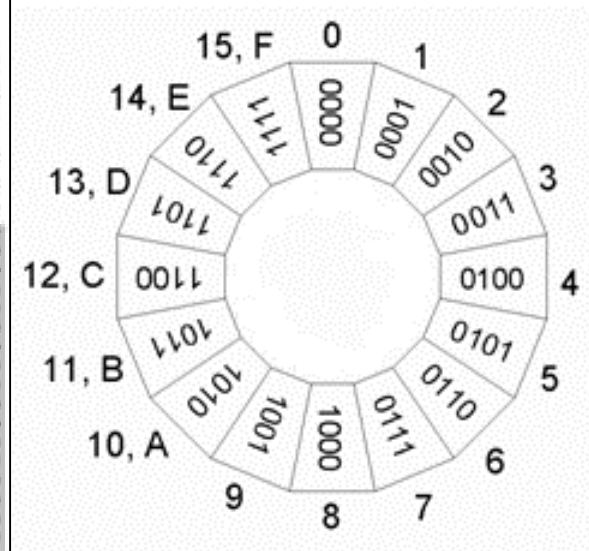
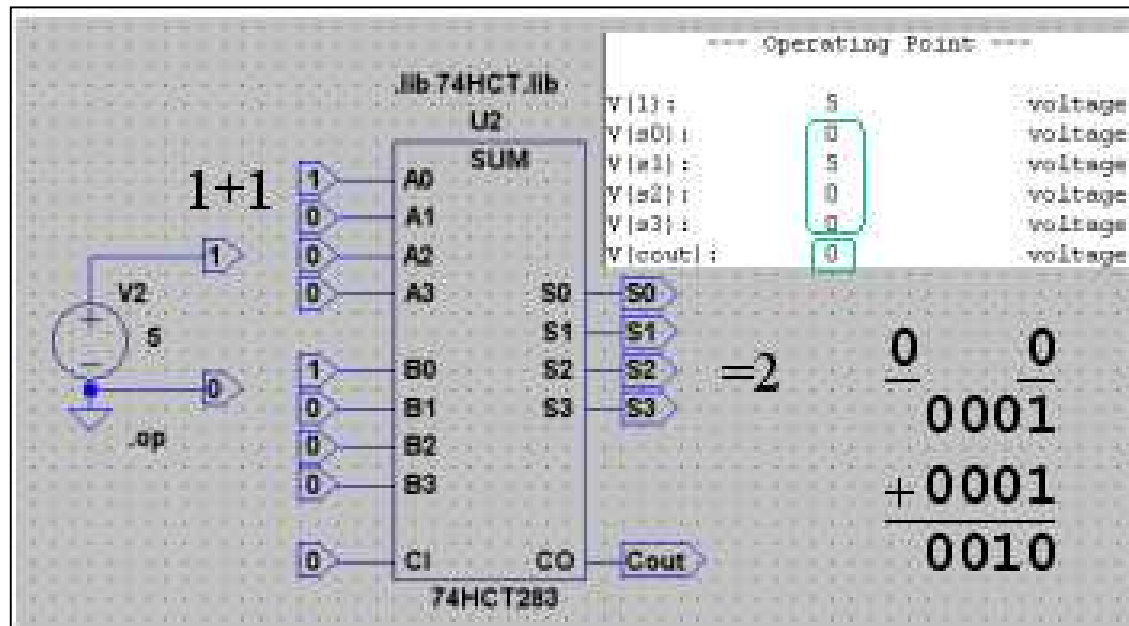


Addition vid Lab1

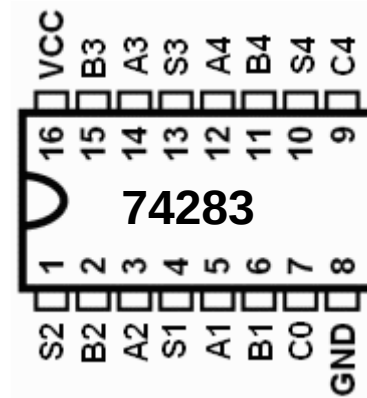
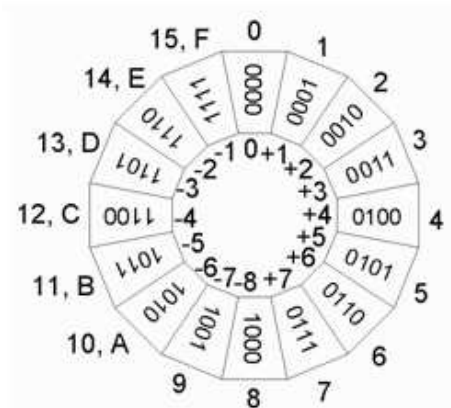


Kopplingsdäck med 4-bitsadderararkrets 74283.

Simulera addition

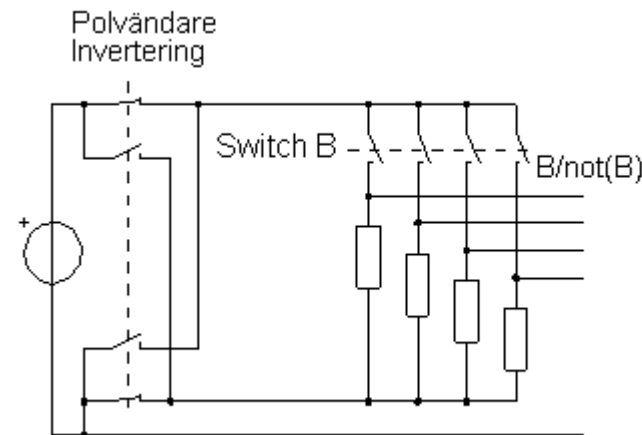
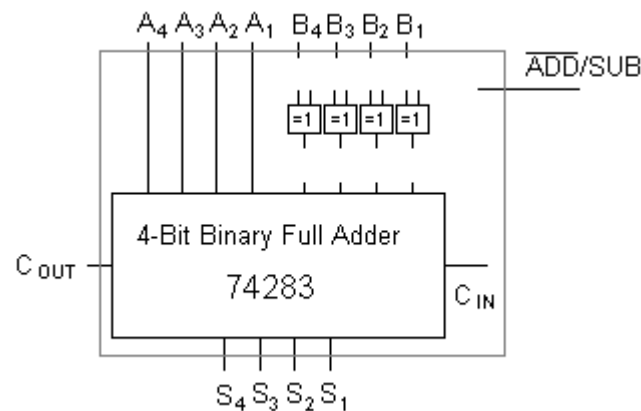


Subtraktion vid Lab1

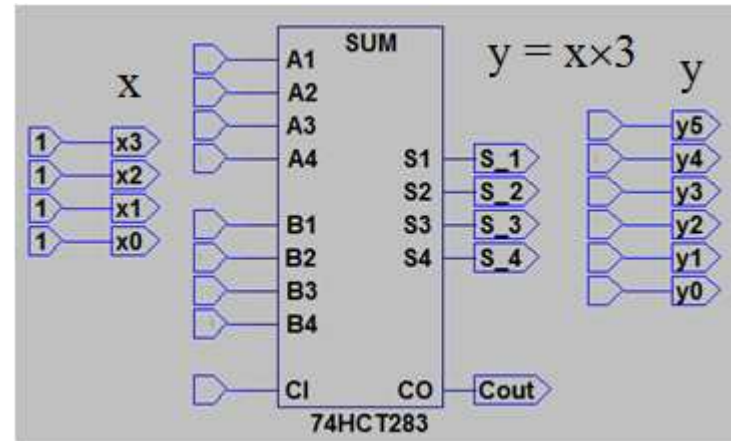
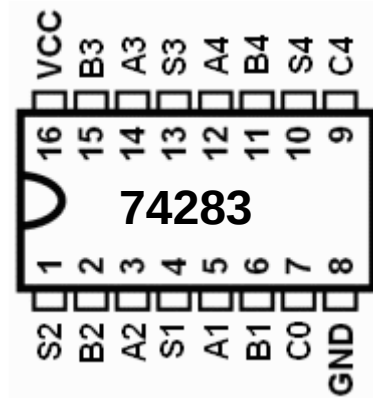


Inverteringen sker med en kontakt i stället för med XOR-grindar!

Rita klart ...



Multiplikation med konstant



Antag att vi behöver multiplicera ett tal x med 3. Det kan man göra som $2 \cdot x + 1 \cdot x = 3 \cdot x$.

Multiplikation med den jämna 2-potensen 2, sker genom att man "skiftar" anslutningarna för talets inbiter *ett* steg åt vänster.

- Eller som $2 \cdot (x + 0,5 \cdot x) = 3 \cdot x$

William Sandqvist william@kth.se