IE1205 Digital Design:

F6 : Digital aritmetik 2

Talrepresentationer

- 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

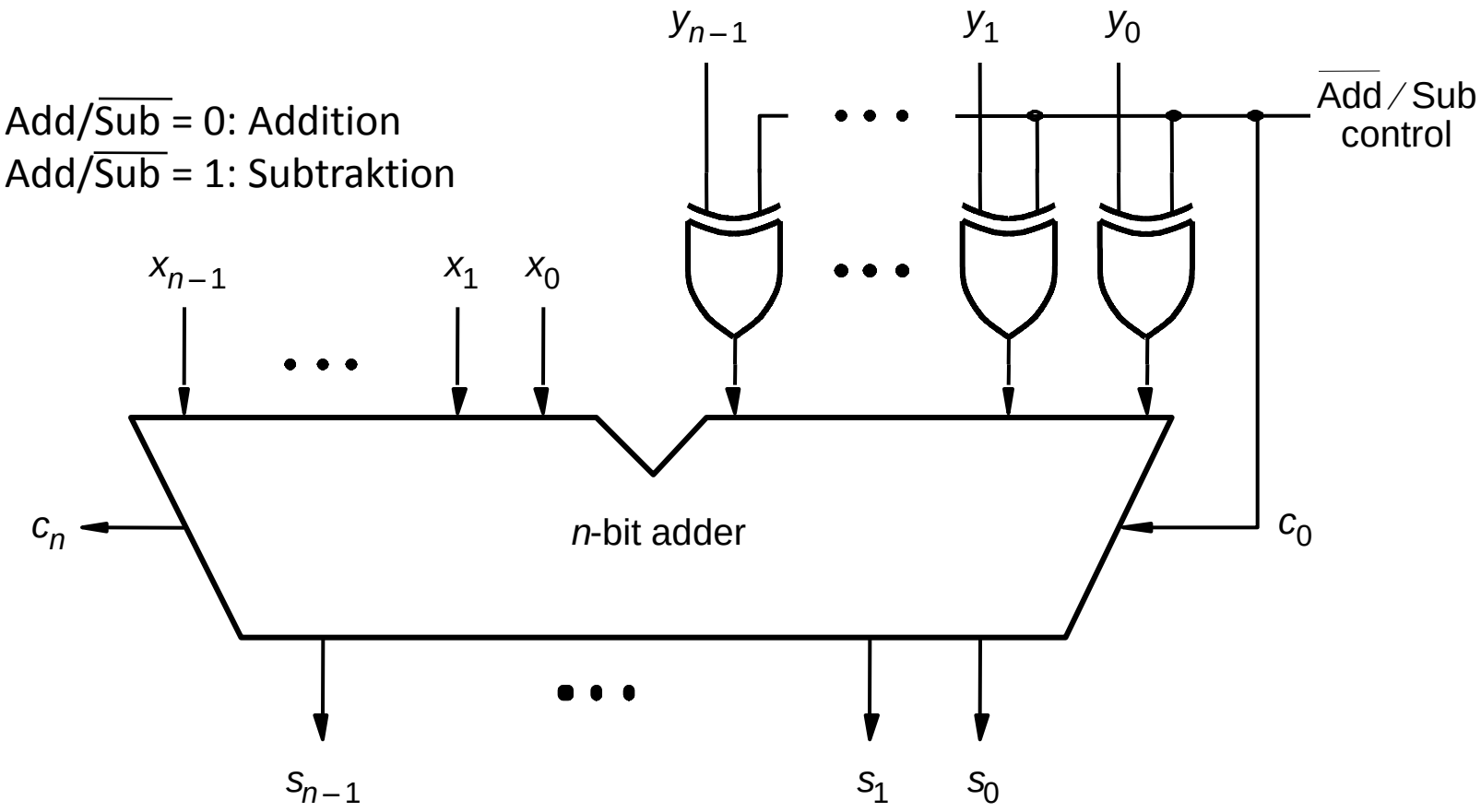
$=1*2^6 + 1*2^5 + 1*2^3 + 1*2^2 + 1*2^0 = 109$

Negativa Heltal:

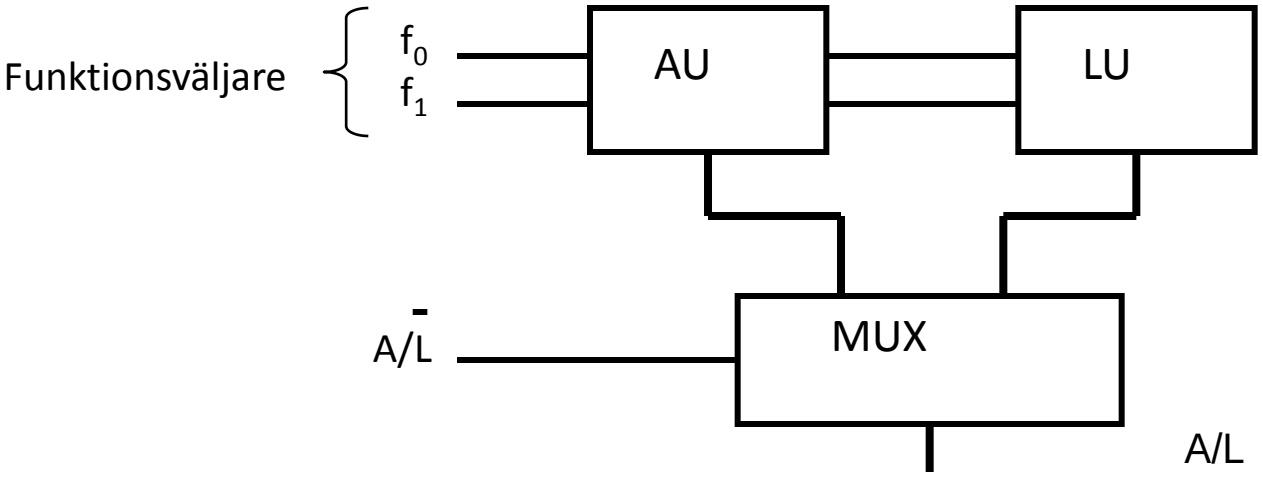
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	1	0	1

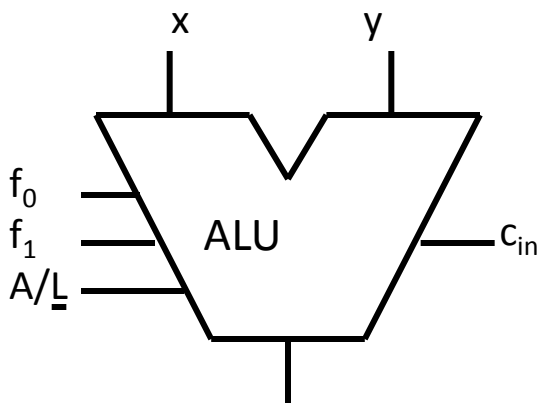
$=-1*2^7+1*2^6 + 1*2^5 + 1*2^3 + 1*2^2 + 1*2^0 = -19$

Add/sub-enheten



Arithmetic Logic Unit (ALU)

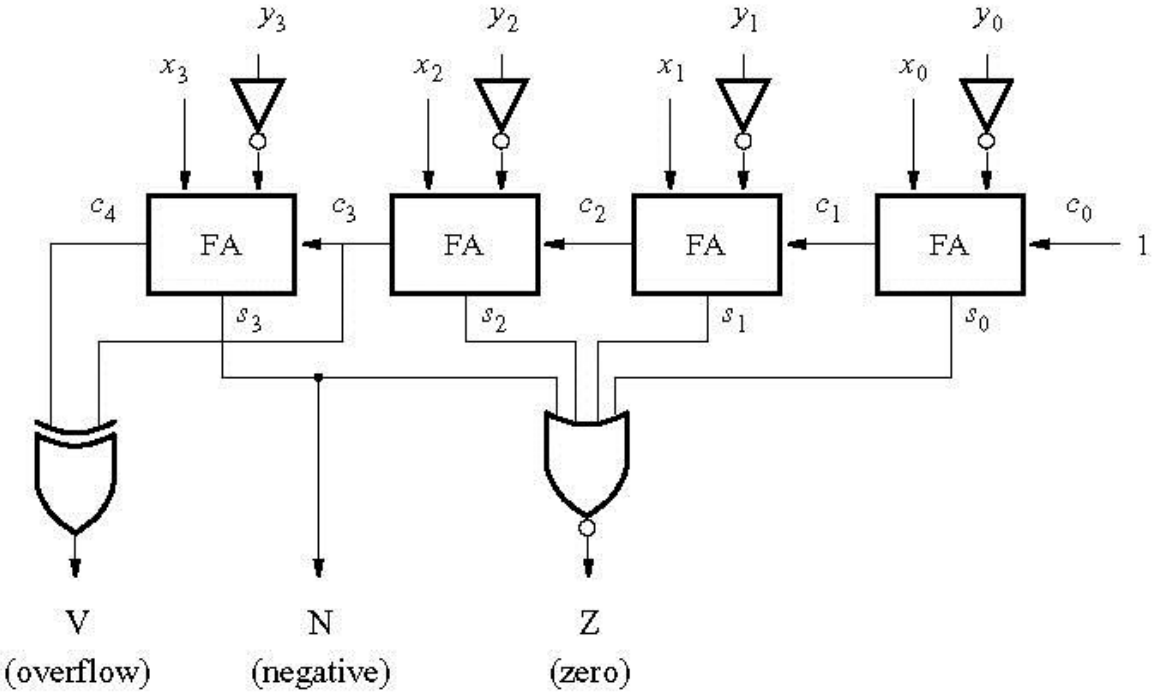




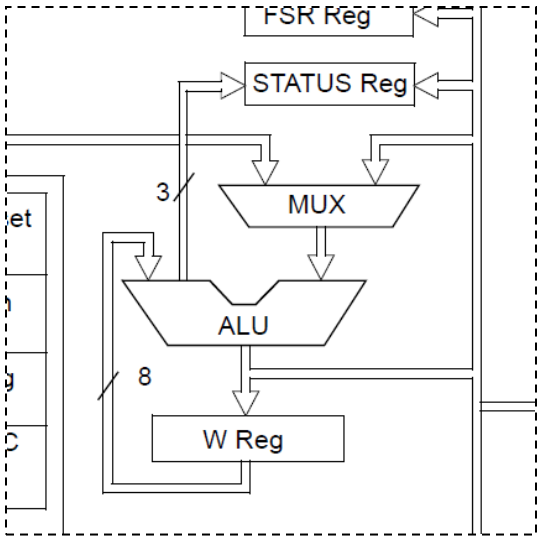
A/L	f1	f0	Funktion
0	0	0	x+y
0	0	1	$x+y+c_{in}$
0	1	0	x-y
0	1	1	$x-y-\overline{c_{in}}$
1	0	0	x or y
1	0	1	x and y
1	1	0	x xor y
1	1	1	inv x

Komparator

- Komparatorn implementeras som subtraktionskrets



Exempel: ALU i mikrokontroller



Microkontroller finns tex i smartcard

9.1 Instruction Description

ADDWF	**Add W and f**
Syntax:	[*label*] ADDWF f,d
Operands:	$0 \leq f \leq 31$ $d \in [0,1]$
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register and register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

Exempel på instruktion i mikrokontroller

Multiplikation av två (positiva) heltal

				0	0	1	0		2
		*		1	0	1	1		11
				0	0	1	0		
			0	0	1	0			
		0	0	0	0				
+	0	0	1	0					
	0	0	1	0	1	1	0		22

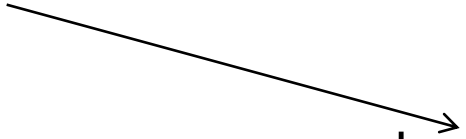
Multiplikation med en teckenbit

Teckenförläng!
(Sign Extension)

				1	0	1	1	**-5**
		*		0	0	1	0	**2**
				0	0	0	0	
	1	1	1	0	1	1		
		0	0	0	0			
+	0	0	0	0				
	1	1	1	0	1	1	0	**-10**

Teckenbiten på det andra stället

Teckenbiten har negativ vikt!
=> Två-komplementera!



				0	0	1	0	**2**
	*			1	0	1	1	**-5**
				0	0	1	0	
			0	0	1	0		
		0	0	0	0			
	+	1	1	1	0			
	1	1	1	0	1	1	0	**-10**

Eller ta två-komplement av båda termer och använd tidigare algoritm

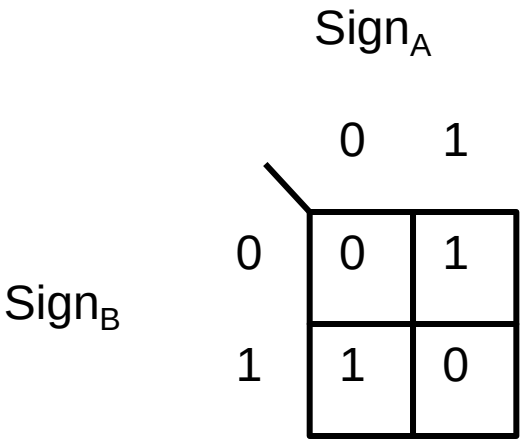
Eller så gör vi det enkelt för oss...

- Använd enbart positiva tal i multiplikationen
  - Konvertera till positiva tal
  - Håll reda på resultatets tecken

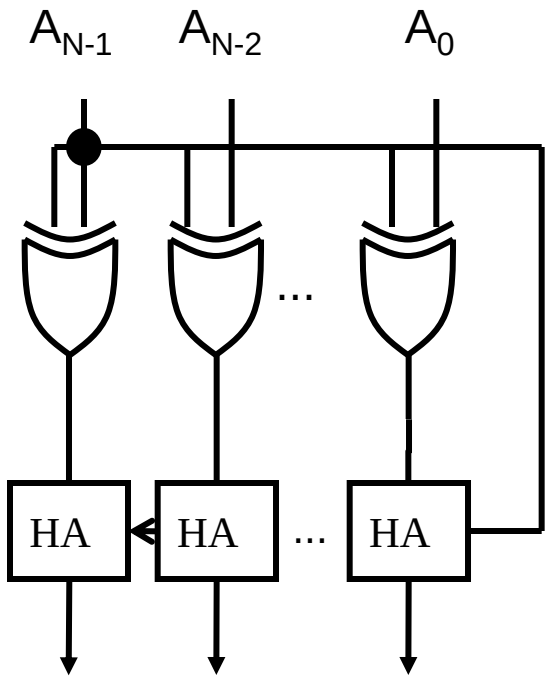
    (+,+)=>+; (+,-)=>-; (-,+)=>-; (-,-)=>+

  - Två-komplementera till negativt tal om nödvändigt

En enkel lösning (forts.)

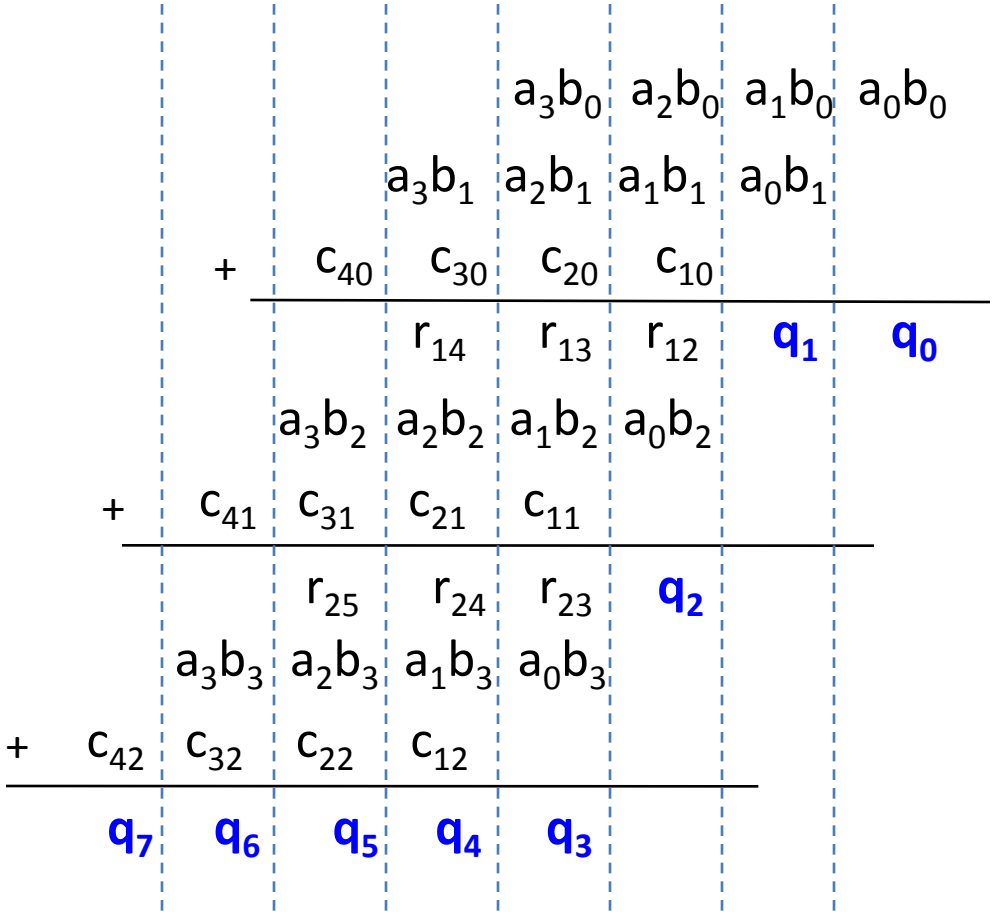


$F_{invert}=S_A$ xor S_B

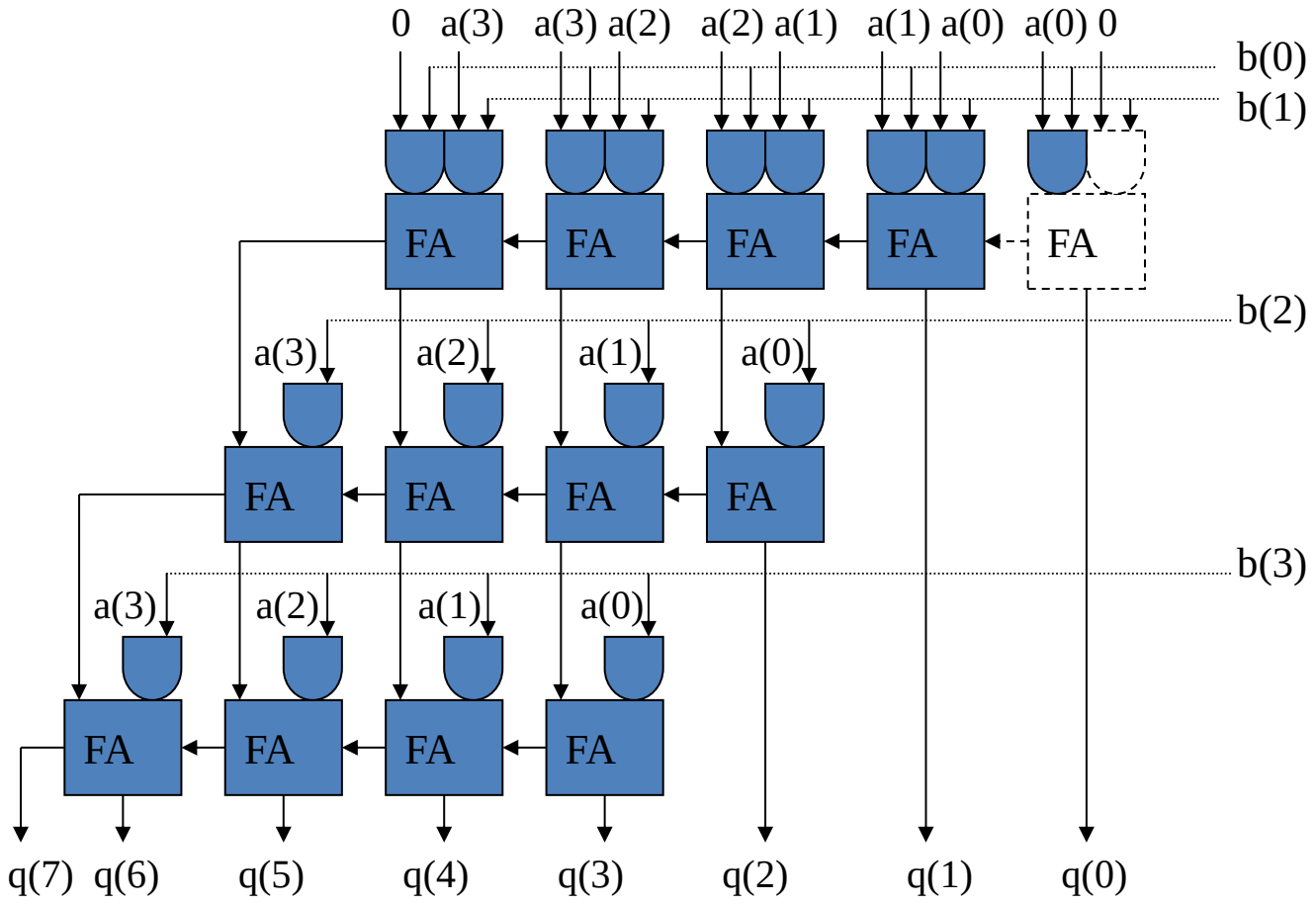


2's complement of A

Multiplikation



Multiplikatorn (två positiva tal)



Snabbfråga repetition

Varför är en carry look ahead adder snabbare än en ripple adder?

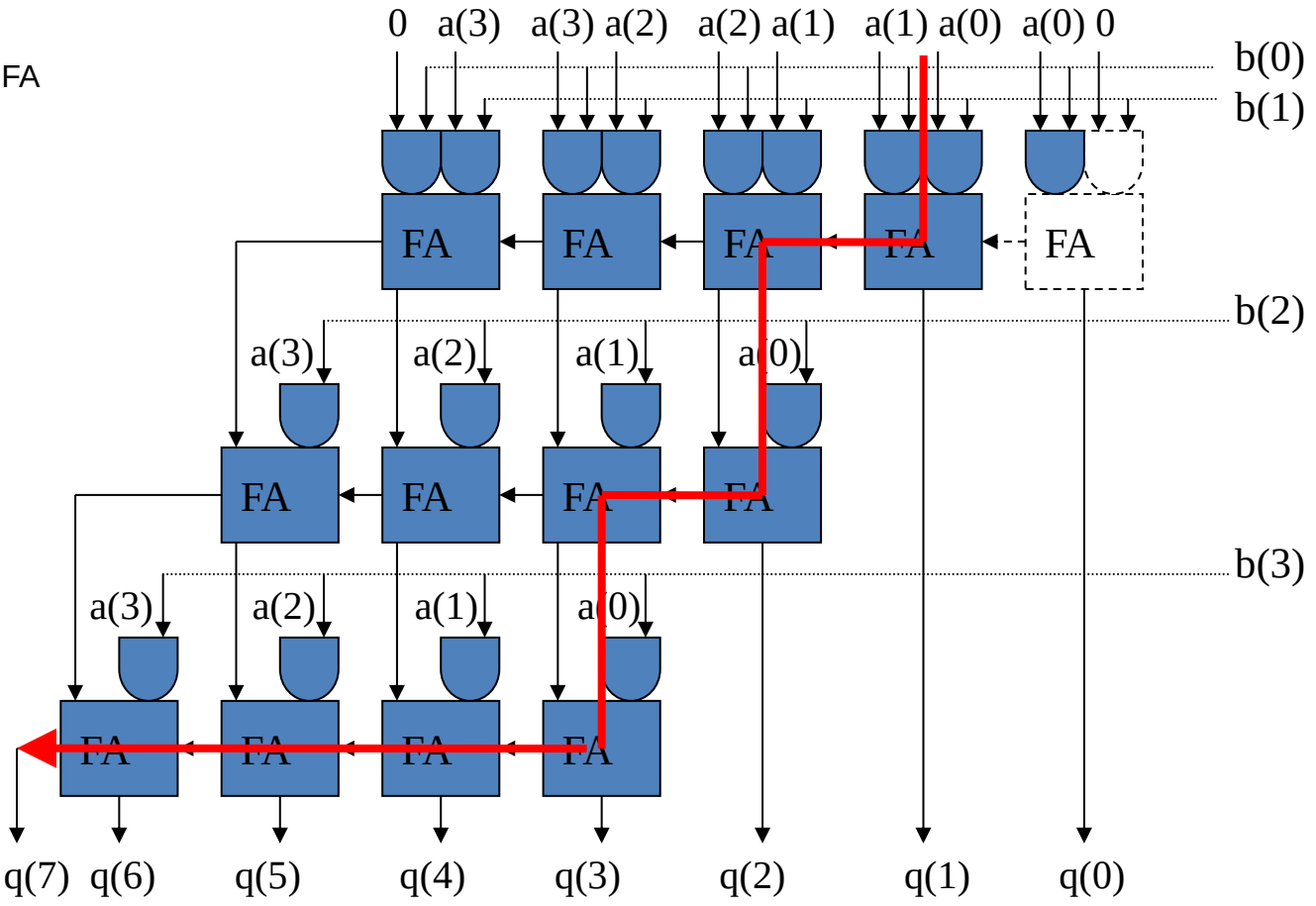
A: Den har fler antal grindar

B: Den har färre antal grindar i signalvägen

C: Den kan implementeras med en bättre teknologi

Multiplikatorn (två positiva tal)

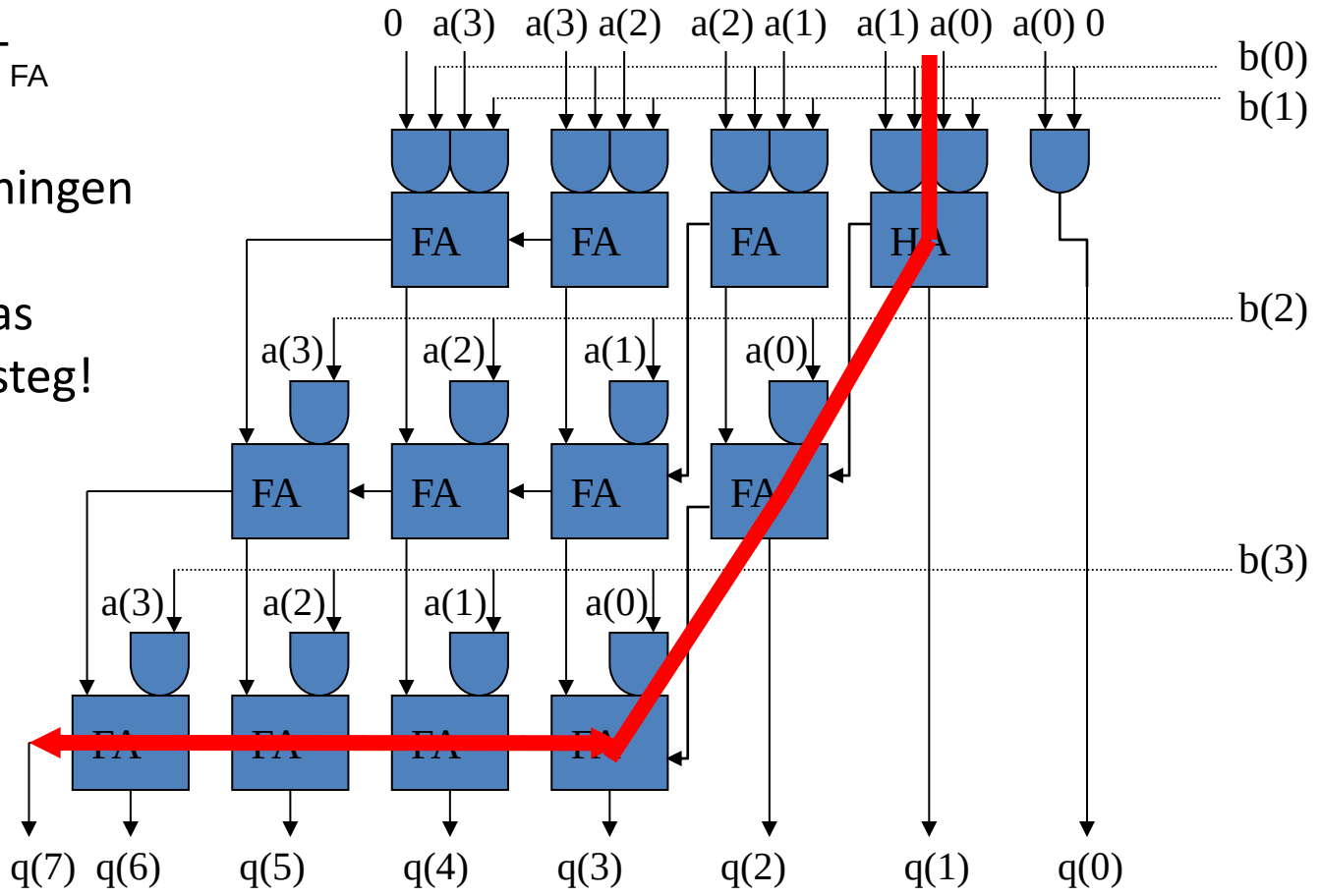
$T_{MUL}=\sim(3*N-4)*T_{FA}$



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

$T_{MUL}=\sim(2*N-2)*T_{FA}$

Minskar fördröjningen eftersom carry utgången kopplas direkt till nästa steg!



Multiplikation med 2

0101*2=1010	(5*2=10)
1010*2=10100	(-6*2=-12)
01010101*2=010101010	(85*2=190)
10010101*2 = 100101010	(-107*2=-214)

jfr multiplikation med 10 i basen 10:

63*10 = 630, -63*10=-630 etc.

Multiplikation med 2^n

0101*2=1010 (5*2=10)
0101*2^2=10100 (5*4=20)
0101*2^3=101000 (5*8=40)
0101*2^4=1010000 (5*16=80)

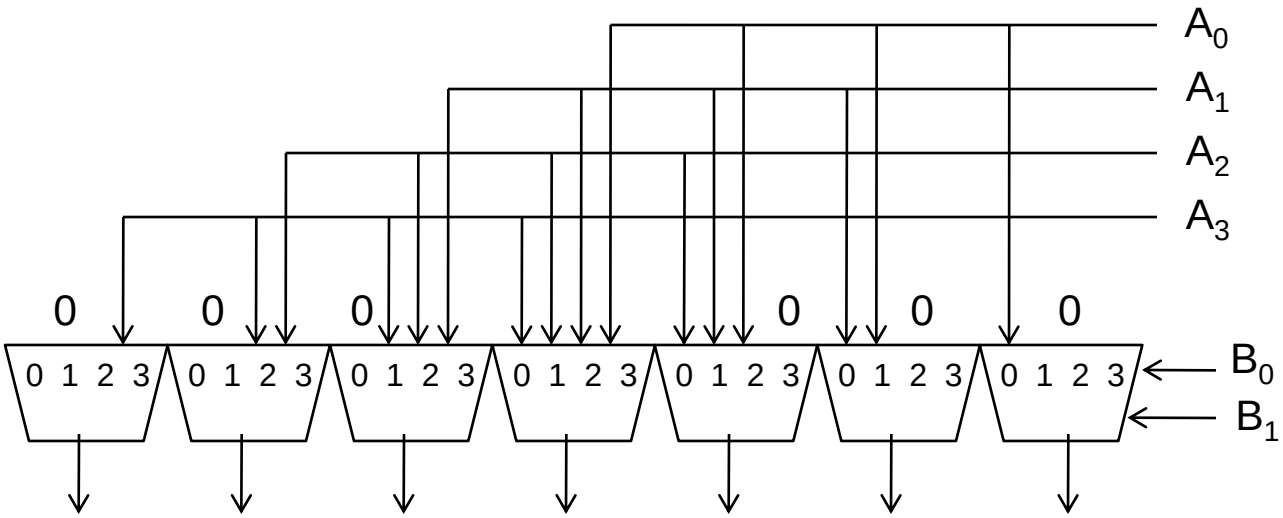
jmfr multiplikation med 10 i basen 10:
6*10 = 60, 6*100=600, 6*1000=6000 etc.

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-shiftern



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

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

```
        0 1 0 1
    ---------------
1 0 | 1 0 1 1
      - 1 0
    -------
        0 0 1
          - 0 0
    -----------
            0 1 1
              - 1 0
            -------
                  1
```

11/2 = 5

Rest = 1

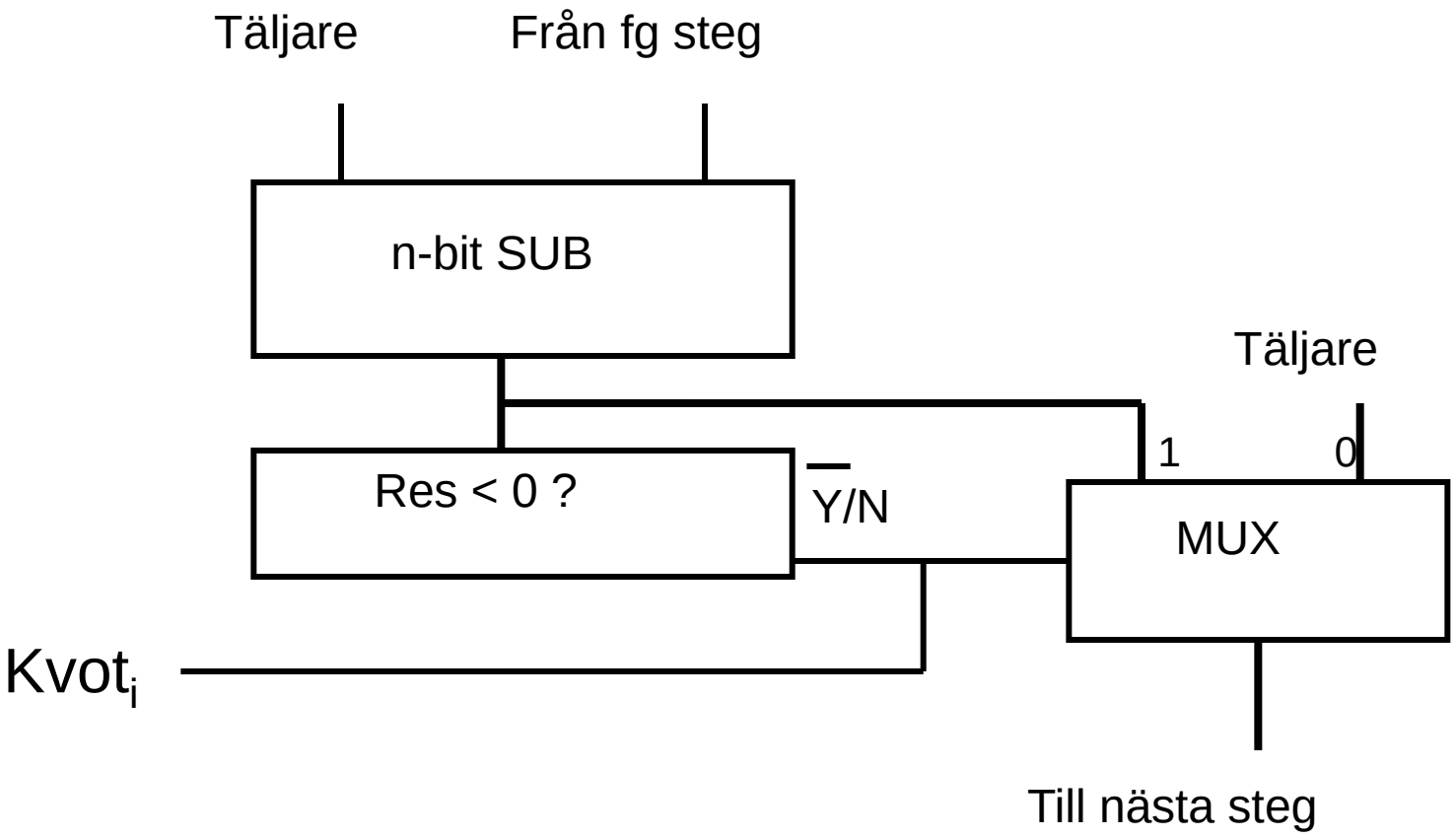
Lite mer detaljerat...

```
          0 1 0 1
      ┌──────────
1 0   │ 0 1 0 1 1
    - | 0 0
      ──────────
          1 0
        - 1 0
          ─────
          0 0 1
            - 0 0
              ─────
              0 1 1
                - 1 0
                  ─────
                    1
```

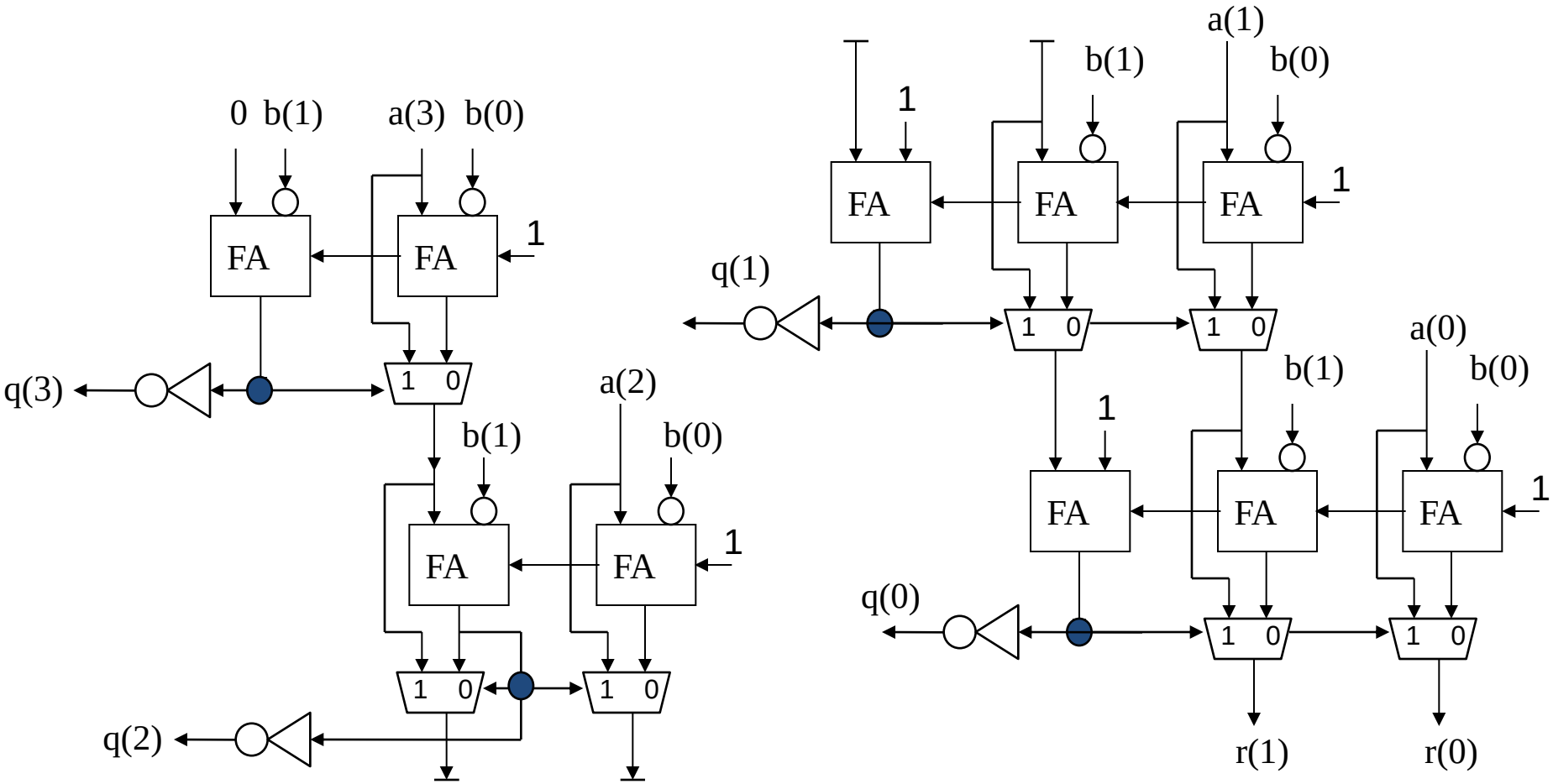
11/2 = 5

Rest = 1

Divideraren

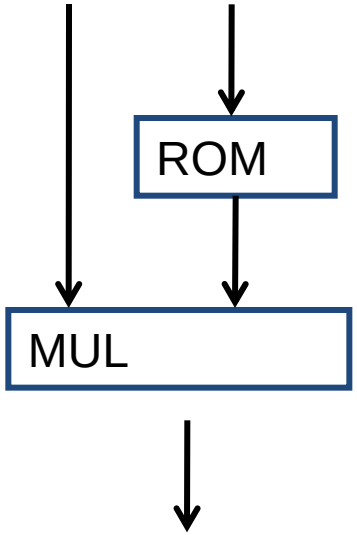


Divideraren (forts.)



Using a multiplier and a ROM for division

$$Q=A/B=A*(1/B)$$



- Properties
  - Fast
  - **VERY** big!!

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

    (+,+)-=>+, (+,-)=>-, (-,+)=>-, (-,-)=>+

  - Två-komplementera till negativt tal om nödvändigt

Division med 2

	01010/2=00101	(10/2=5)
⟶	10100/2=11010	(-12/2=-6)

jmfr division med 10 i basen 10:

630/10 = 63, -630/10=-63 etc.

Logiskt vs 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)

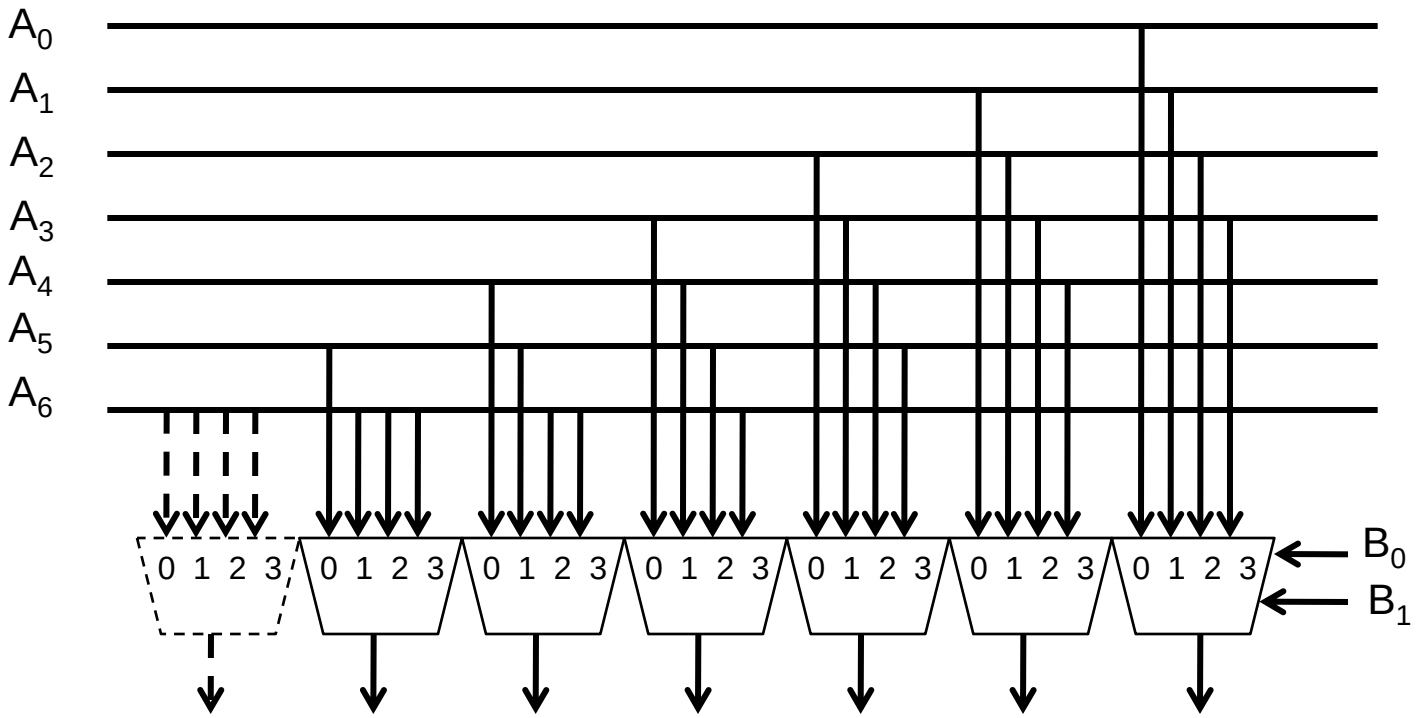
jmfr division med 10 i basen 10:

60/10 = 6, 600/100=6, 6000/1000=6 etc.

Division med 2^n

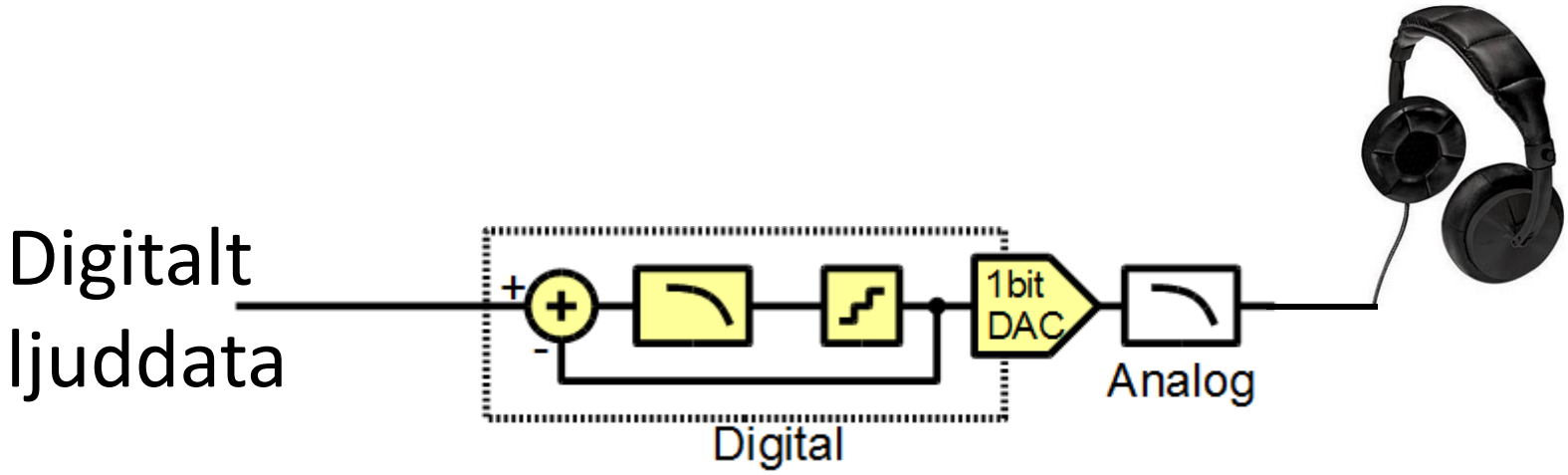
- En division med 2^n kan göras med genom att skifta alla bitar n steg till höger och att fylla på med nollor
- Observera att resultatet behöver inte vara korrekt, eftersom man egentligen behöver bitar ”efter kommatecknet”
- 17/4 motsvarar att skifta 010001 2-bitar till höger
  - Resultat: 000100 = $(4)_{10}$
  - Eftersom $(0.25)_{10}$ inte kan representeras är resultatet inte korrekt!

Barrel-shiftern (2)



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

Exempel signalbehandling



Digital till Analog
omvandlare (DAC)

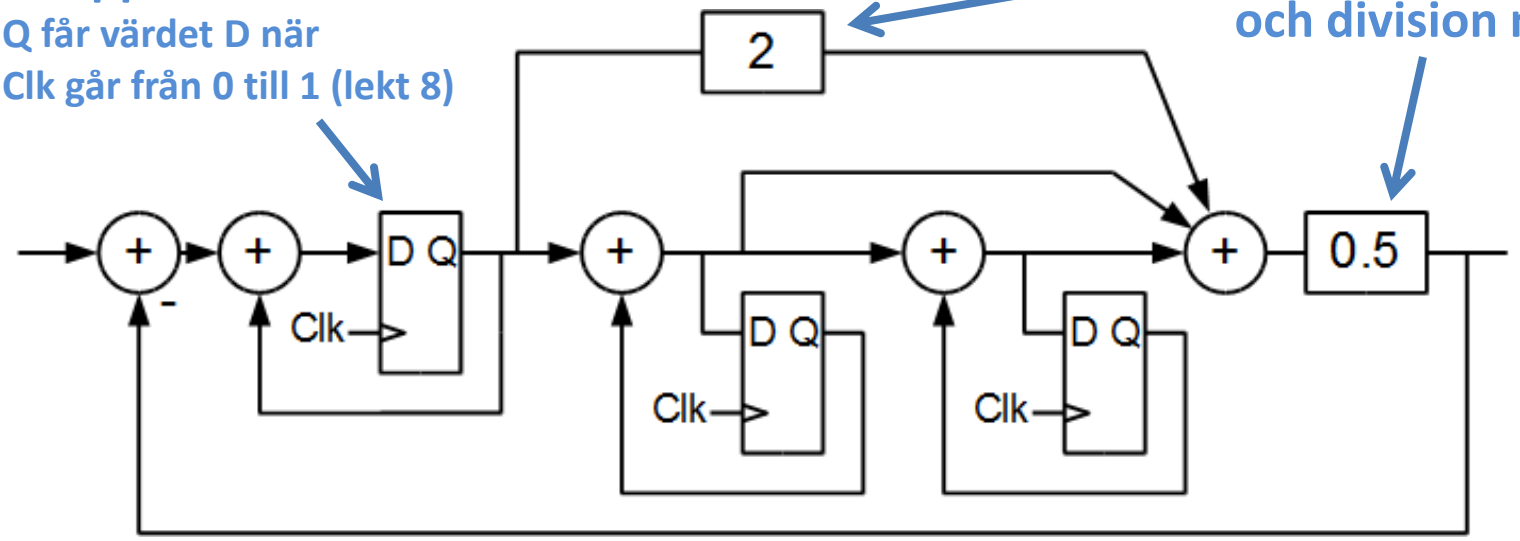
Exempel signalbehandling

Matematisk funktion:

$$H_{qn}(z) = \frac{(1 - z^{-1})^3}{1 - z^{-1} + 0.5z^{-2}}.$$

D-vippa ”låser” värdet
Q får värdet D när
Clk går från 0 till 1 (lekt 8)

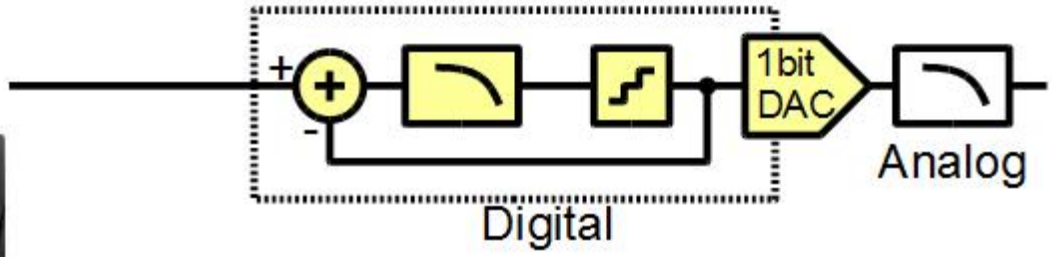
Endast multiplikation och division med 2



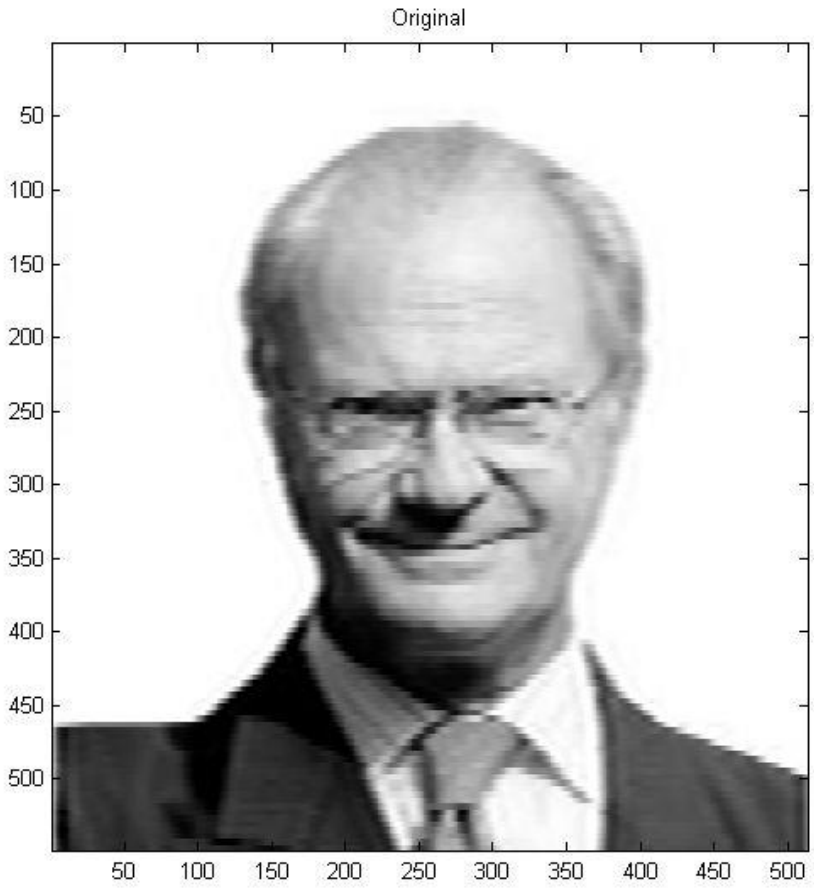
Implementering av funktion i digital hårdvara:

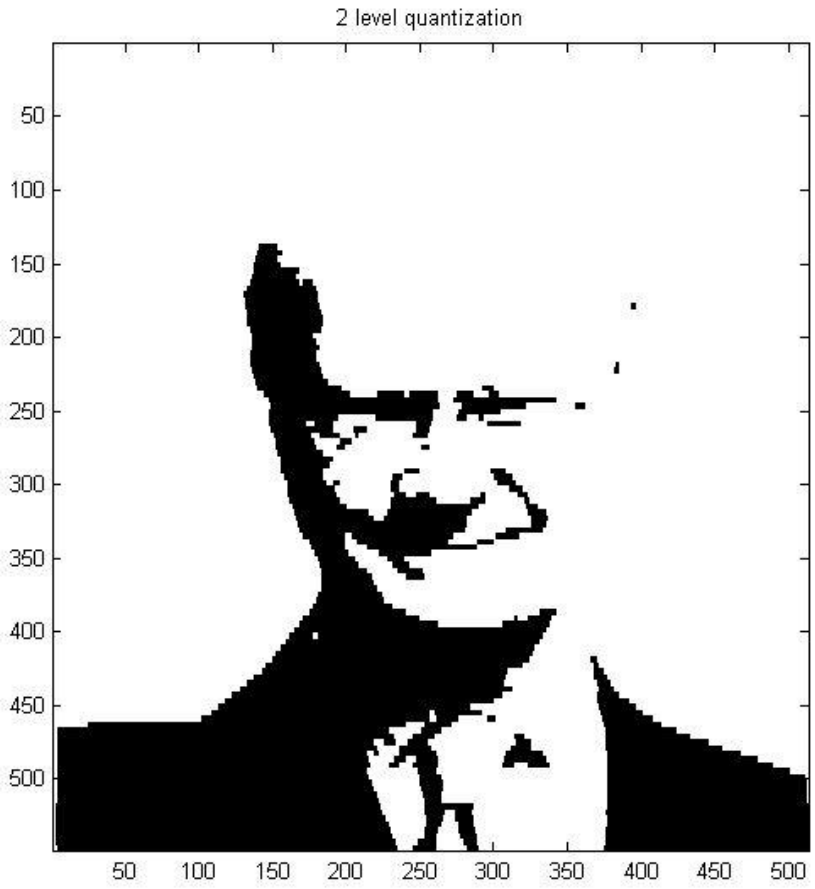
Exempel signalbehandling

Carl XVI Gustaf



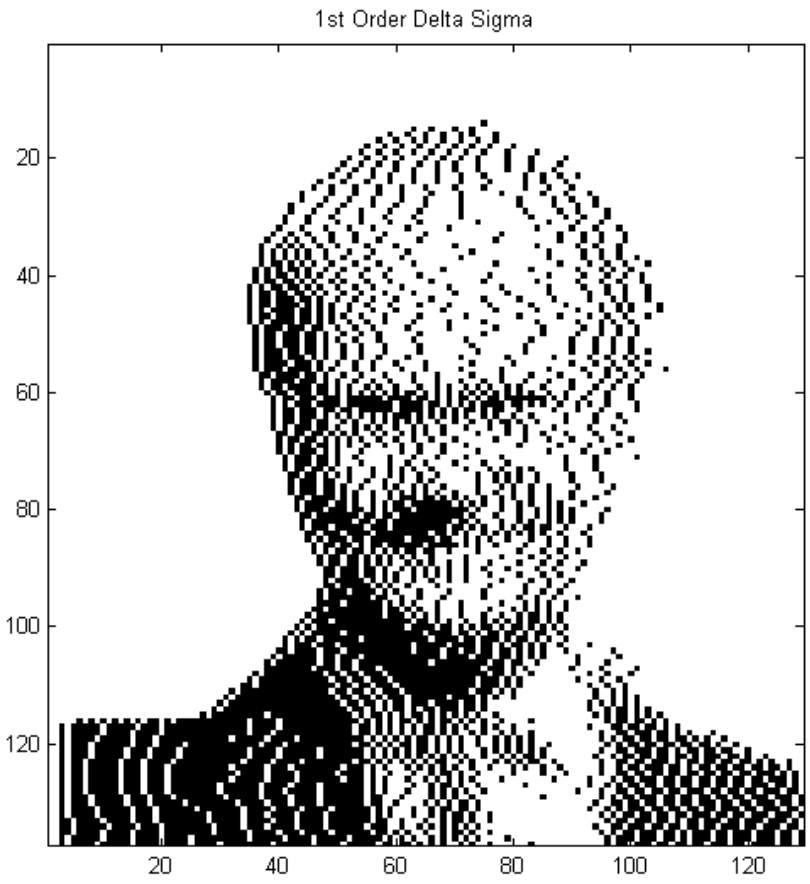
Exempel signalbehandling

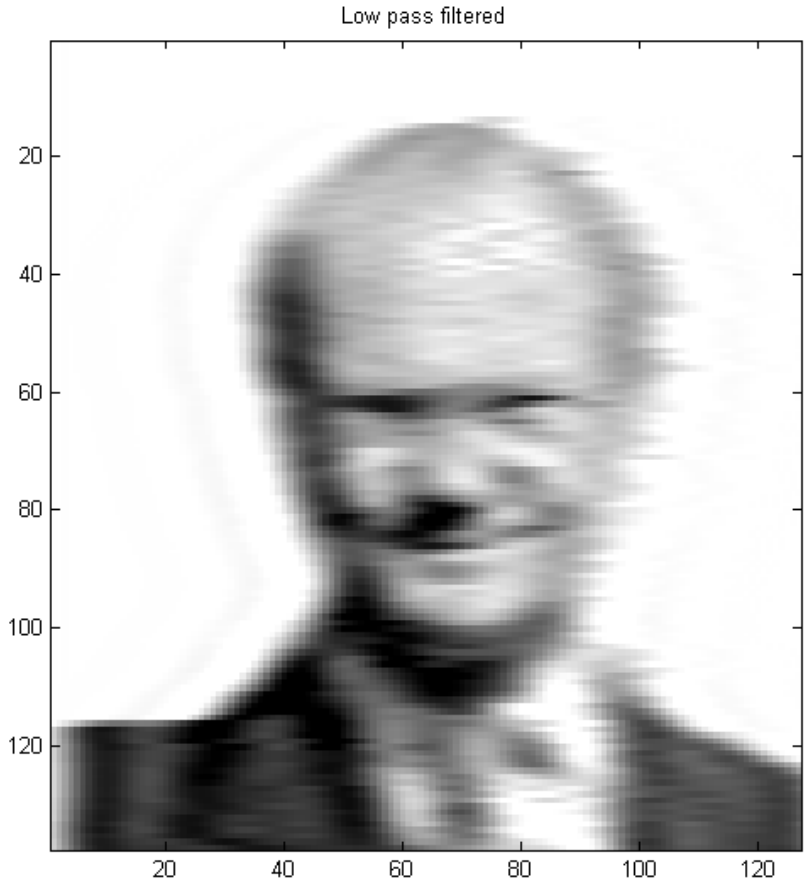




En digital utgång har bara två diskreta värden

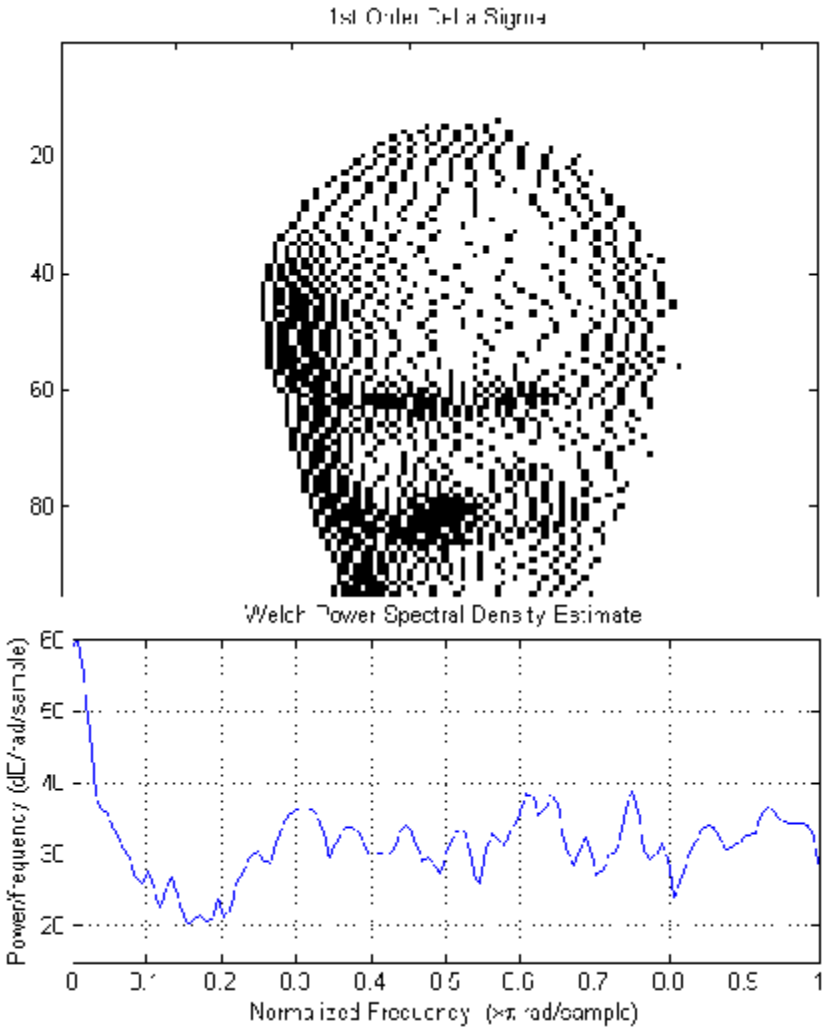
Exempel signalbehandling

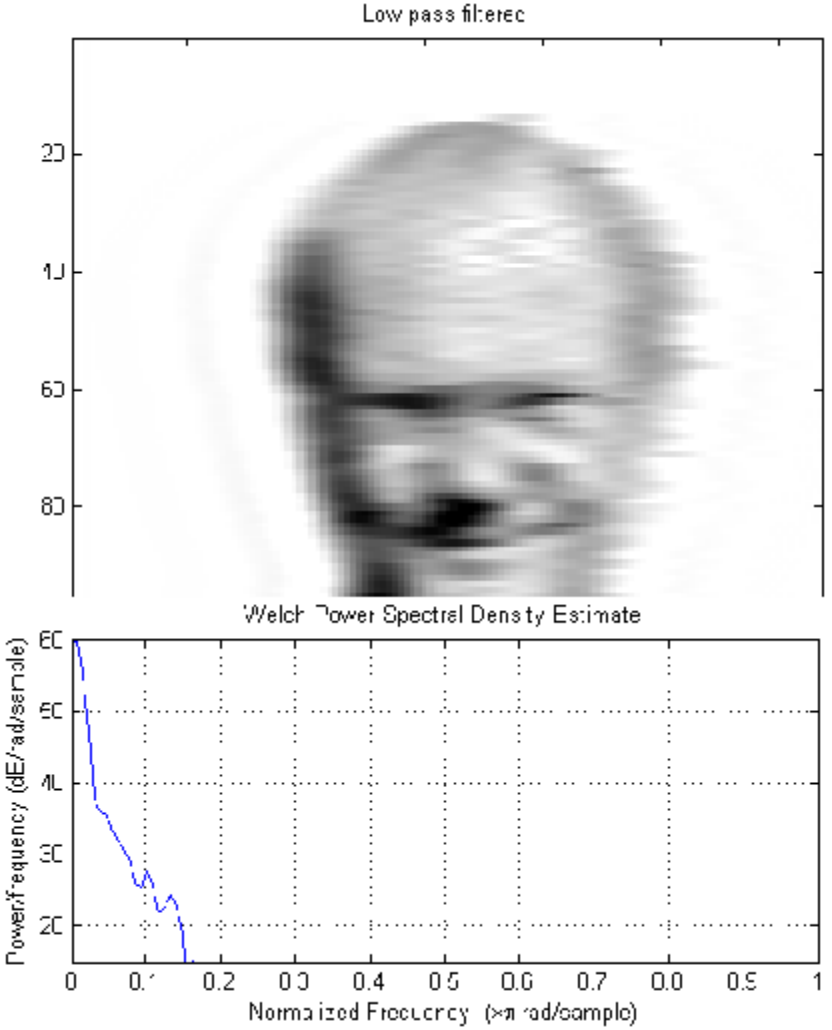




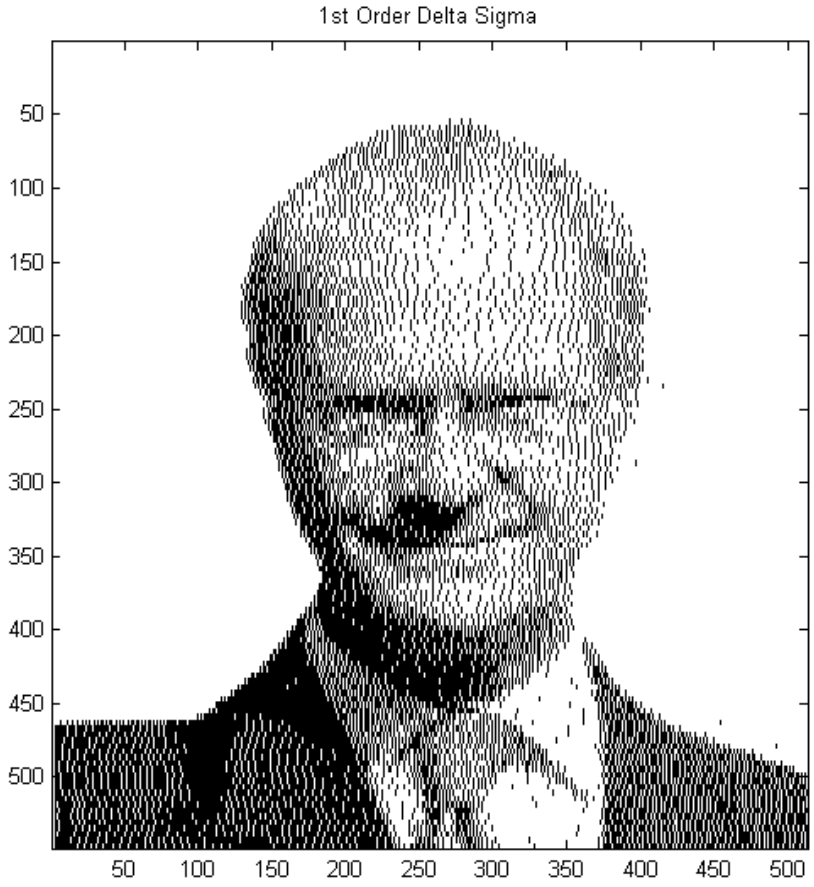
Genom digital signalbehandling kan signalen brusformas
Efter filtrering återskapas den analoga signalen

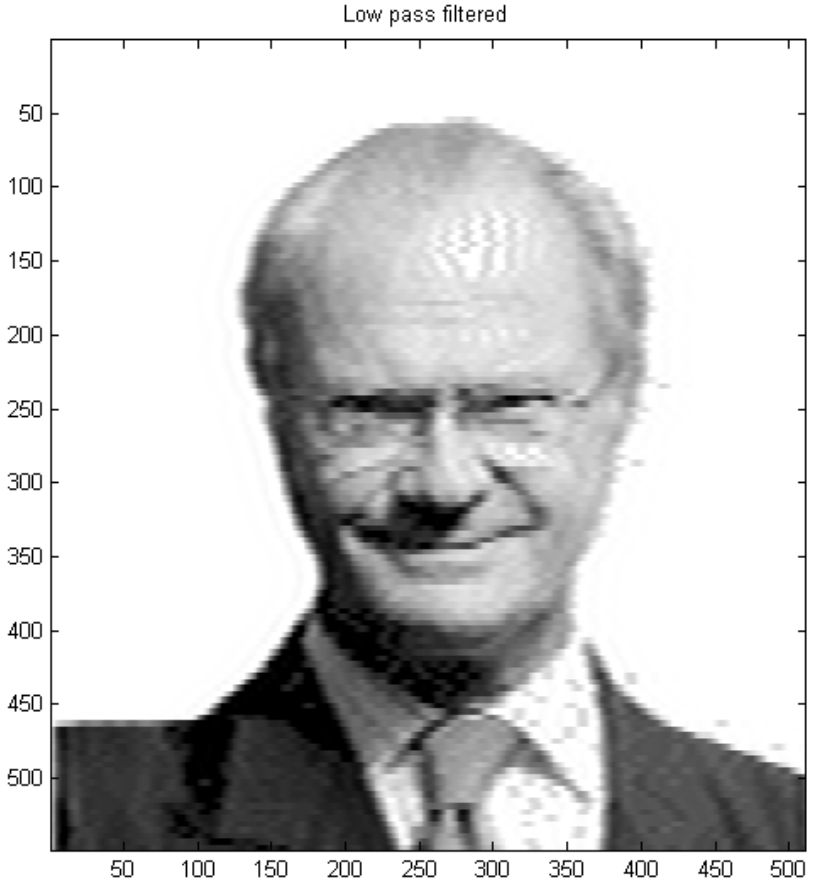
Exempel signalbehandling





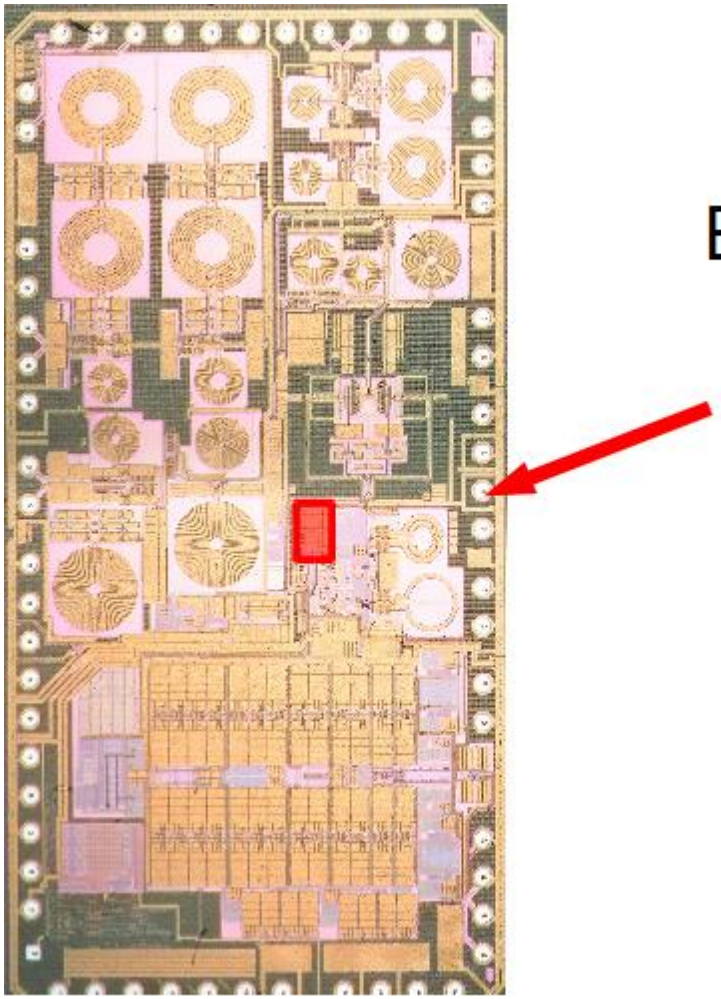
Exempel signalbehandling





Högre klockhastighet möjliggör högre bandbredd hos signalen

Exempel signalbehandling

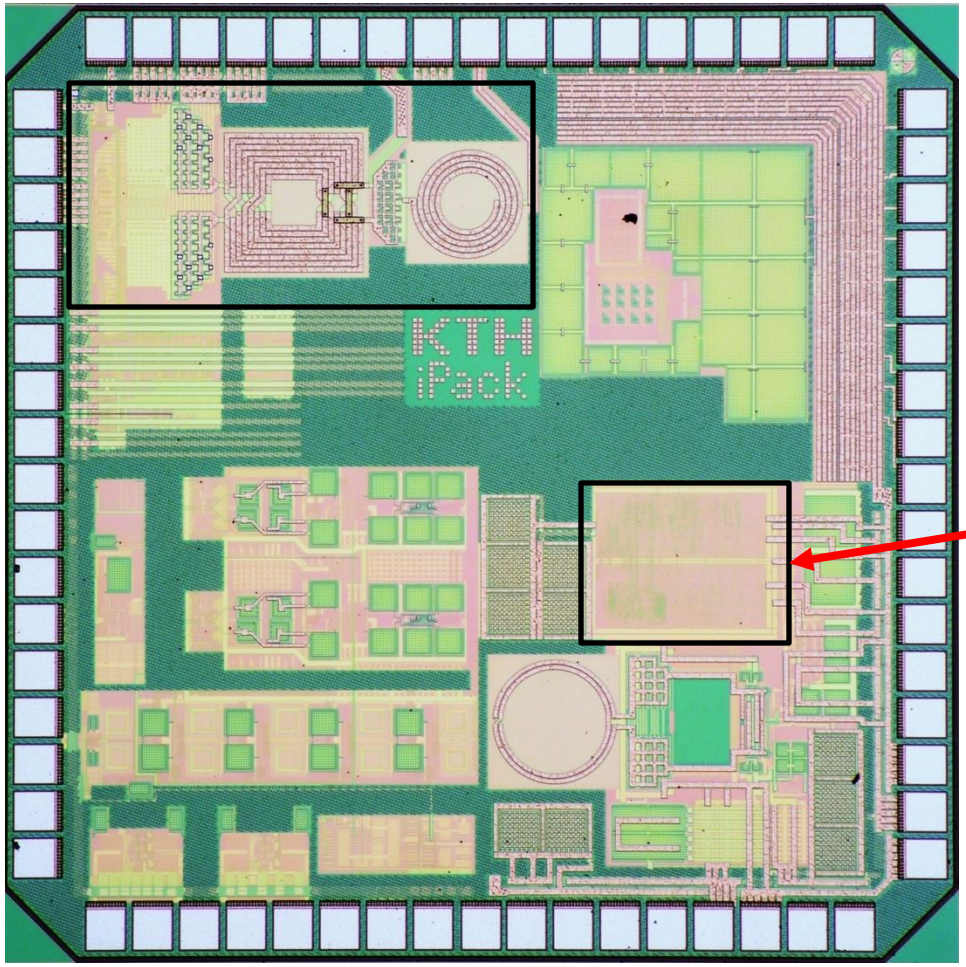


Exempel:

Brusformning i TV tuner
$\Delta\Sigma$ modulated Synthesizer
75 MHz – 1.8 GHz

Tack vare att vi endast multiplicerar/ dividerar med 2 blir ytan och strömförbrukningen mycket liten

Exempel signalbehandling



Digital

I denna krets är digitaldelen inte lika optimerad (innehåller en division). Den tar större yta och har högre strömförbrukning.

Fix-tal
(Fixed-point numbers)

Två-komplementsrepresentation

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

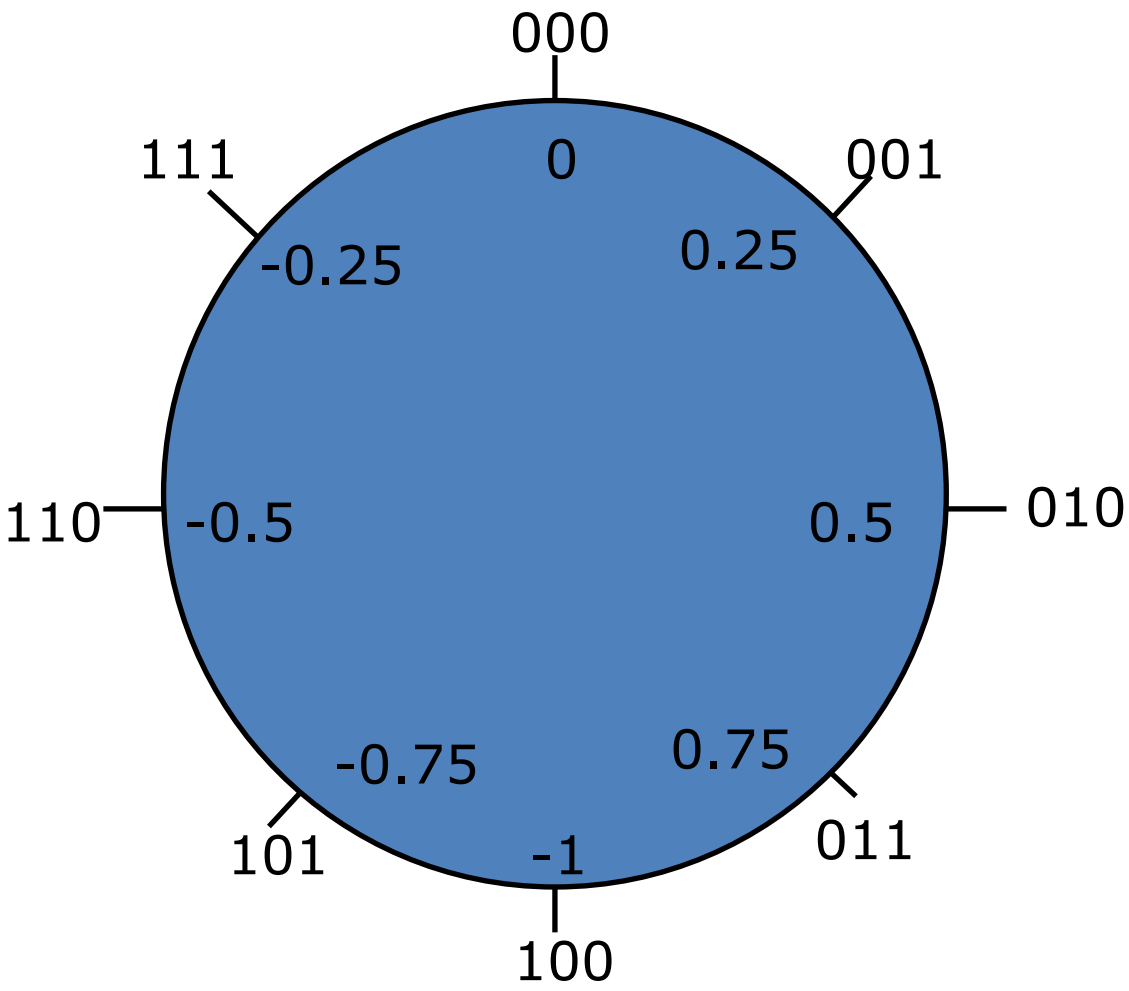
Sign Bit

Decimalt värde

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

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

Fixed-Point Q2-Representation



Multiplikation i Q-formatet

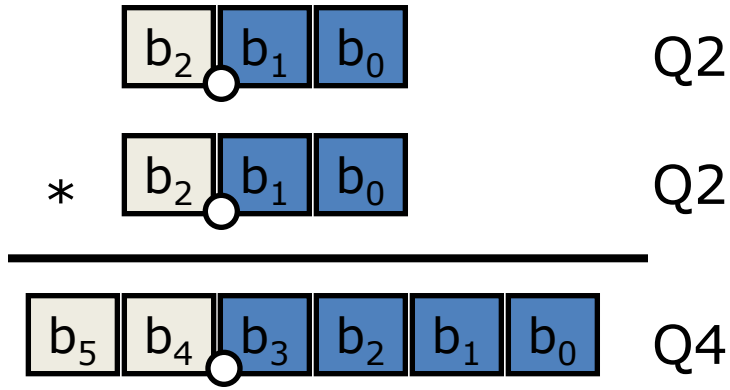
- Multiplikation orsakar inte overflow i Q-formatet, men kan resultera i förlust av precision

  $(1.10)_2 * (0.11)_2$

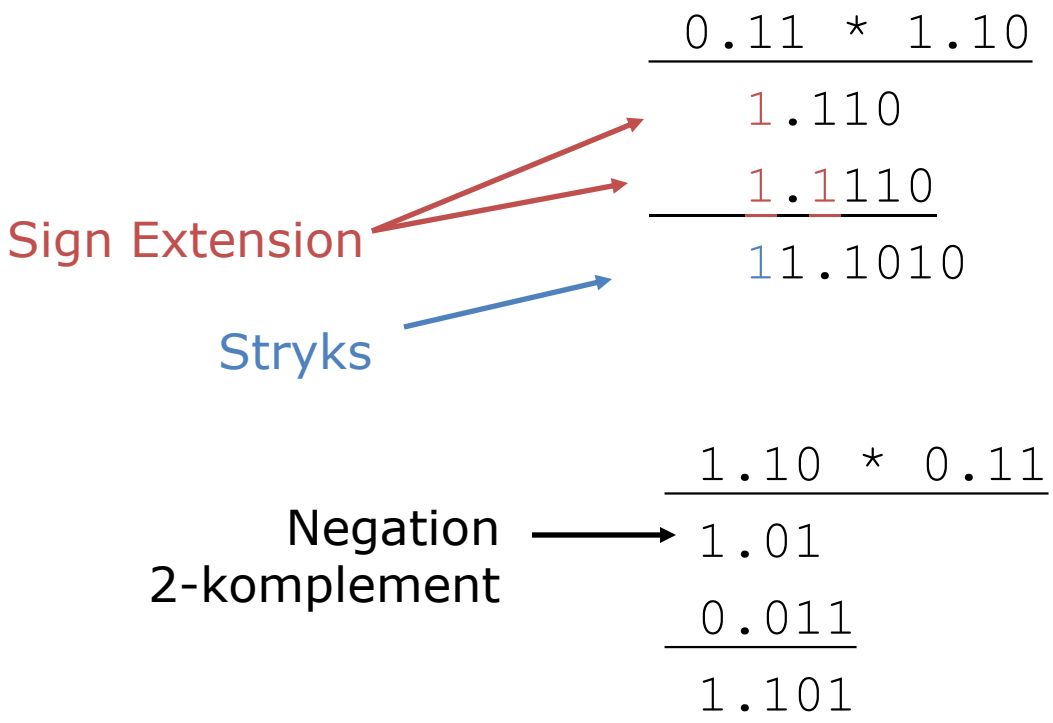
  $= (1.1010)_2$ (Q4-format)

  $= (1.10)_2$ (Q2-format)

Generellt:

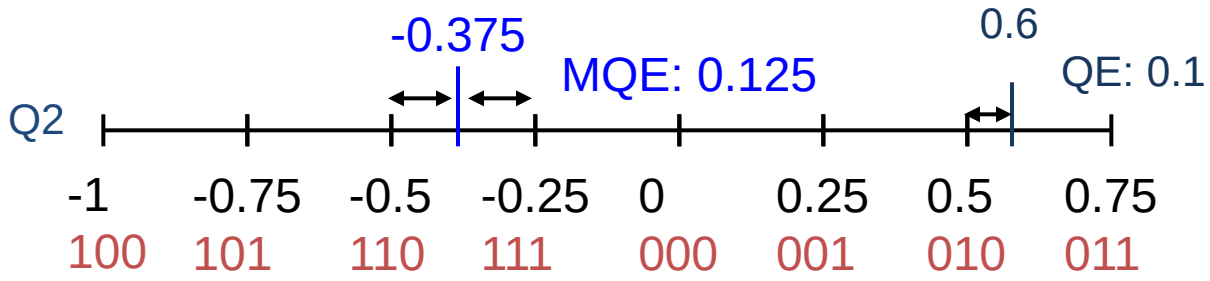


Multiplikation i Q-formatet



Glöm inte att expandera teckenbiten!

Kvantiseringsfel



- Maximalt kvantiseringsfel:
  - Eftersom inte alla tal kan representeras uppstår kvantiseringsfel
  - Kvantiseringsfelet blir mindre om vi använder fler bitar

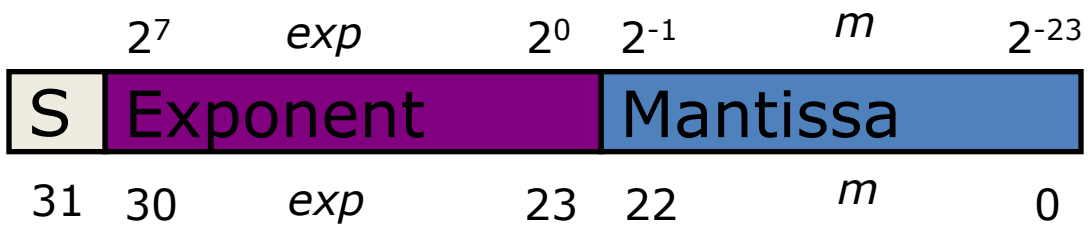
Flyt-tal (eng. Floating-Point Numbers)

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

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

IEEE-754

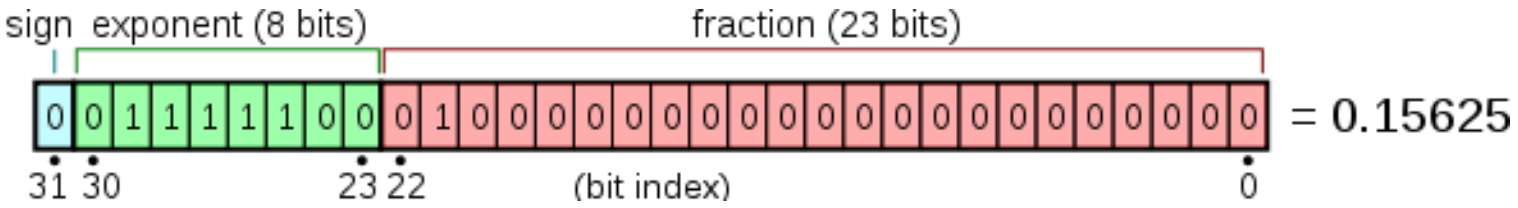
- Flyttals-standarden IEEE-754 definierar ett 32-bit flyttal som



  - Värdet beräknas för en 8-bitars exponent enligt nedan

    $FlP(B) = (-1)^s * (1.m) * 2^{exp-(127)}$
  - Specialla bit pattern har reserverats för att representera negativ och positiv nolla

Exempel



Sign = 0: Positivt

Exp: (64 + 32 + 16 + 8 + 4) – 127 = 124 – 127 = - 3

Mantissa:

Binary 1.01 = 1 + 1 / 4 = 1.25

Första biten är alltid ett eftersom mantissan är normaliserad

$1.25 * 2^{-3} = 1.25 * 0.125 = 0.15625$

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 (255) 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}

  $$FlP(B) = (-1)^s * (\mathbf{0}.m) * 2^{-126}$$
- Exponent of ones (255) with a non-zero fraction: *NotANumber* (Exception Condition)
- There is also a standard for a 64-bit number

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:

$$c = a + b$$

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

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:

$$c = a - b$$

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

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:

$$c = a * b$$

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

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

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

  - Om exponenten är noll är mantissans första bit 0

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

Fixed-Point vs. Floating-Point

- Fixed-Point operationer fungerar på samma sätt 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

Snabbfråga

Hur representeras 0.75 i Q4 fixed point format?

A: 01001011

B: 00001100

C: 1100

Information om laborationer

- Kontrollera labbtid och kunskapsgrupp i Daisy
- Gör förberedelseuppgifter och fyll i web-formulär

Sammanfattning

- **Multiplikation och division av heltal**
  - Konvertera negativa tal till sitt positiva ditto.
  - Utför multiplikationen eller divisionen
  - Håll reda på vilket tecken resultatet skall ha
  - Konvertera positivt resultat till sitt negativa ditto 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.