# Moment 2 - Digital elektronik

## Föreläsning 1
## Binära tal och logiska grindar

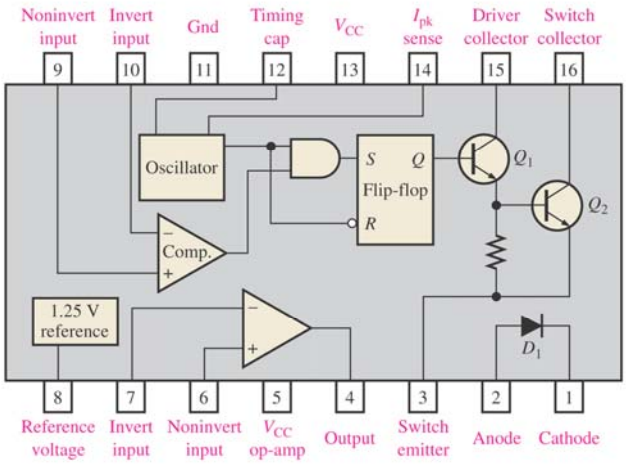
Jan Thim



## F1: Binära tal och logiska grindar

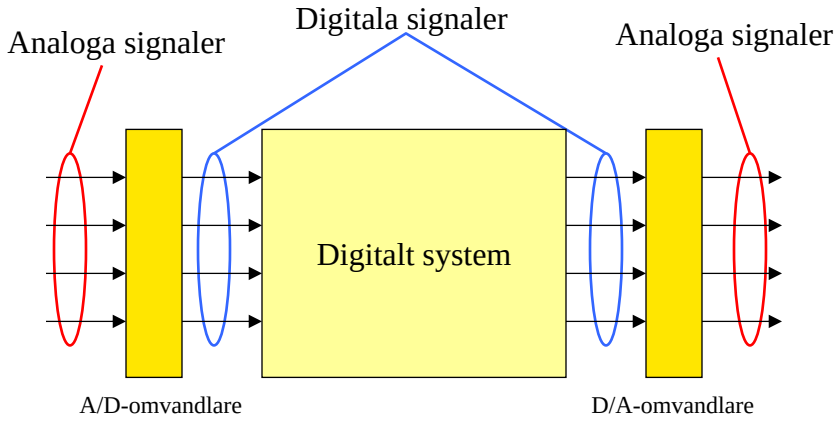
- Innehåll:
  - Introduktion
  - Talsystem och koder
  - Räkna binärt
  - Logiska grindar
  - Boolesk algebra

## Varför lär vi oss detta?



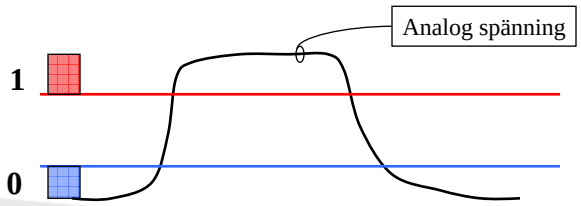



## Analoga-digitala system



## Digitala signaler

- Värden på digitala siffror är
  - 0
  - 1
- En digital signals värde representeras av en spänning






## Analogt jämfört med Digitalt

- Digitalt
  - Diskontinuerligt
    - Värde (kvantisering)
      Variabeln kan endast anta ett begränsat antal värden
    - Tid (tidsdiskret)
      Variabeln är enbart definierad vid vissa tidpunkter
    - Exempel på diskreta värden
      En omkopplares position (på/av), digital logik (0/1), värdet på en sedel (20/50/100/500 …)

## Varför digitalt?

- Okänslig för störningar
- Tillförlitlig
- Inte dyrt
- Programmerbart
- Reproducerbart
- Många digitala funktioner kan integreras i en integrerad krets



## Digital teknik finns överallt

- Elektronik baserad på digitalteknik används i många olika tillämpningar
  - Bilmotorer
  - Mikrovågsugnar
  - Digitala ur
  - Mobiltelefoner
  - Videospel
  - Etc. etc.

## Binära talsystemet

- Binärt
  - Positionssystem
  - Två symboler används, B = { 0, 1 }
  - Binära tal gör det lätt att bygga elektronik baserade på elektroniska omkopplare
  - En algebra utvecklad av Boole gör det lätt att hantera logiska uttryck baserade på binära tal



## Representera positiva heltal

- För positiva heltal:

$$N = \sum_{i=0}^{q-1} s_i b^i$$

- *ex.*, Decimalt tal

$$(234)_{10} = s_2 b^2 + s_1 b^1 + s_0 b^0$$

låt $b = 10, \quad S = \{0,1,\ldots,8,9\}$

$$(234)_{10} = 2 \bullet 10^2 + 3 \bullet 10^1 + 4 \bullet 10^0$$

$$(234)_{10} = 2 \bullet 100 + 3 \bullet 10 + 4 \bullet 1$$

$$(234)_{10} = 200 + 30 + 4$$

## Representera positiva heltal

- *exempel*, Binärt tal (basen 2)

$$(1011)_2 = s_3 b^3 + s_2 b^2 + s_1 b^1 + s_0 b^0$$

låt $b = 2$, $S = \{0,1\}$

$$(1011)_2 = 1 \bullet 2^3 + 0 \bullet 2^2 + 1 \bullet 2^1 + 1 \bullet 2^0$$

$$(1011)_2 = 1 \bullet (1000)_2 + 0 \bullet (100)_2 + 1 \bullet (10)_2 + 1 \bullet (1)_2$$

$$(1011)_2 = (1)_{10} \bullet (8)_{10} + (0)_{10} \bullet (4)_{10} + (1)_{10} \bullet (2)_{10} + (1)_{10} \bullet (1)_{10}$$

$$(1011)_2 = (8+2+1)_{10} = (11)_{10}$$



## Exempel på decimal tal

I det här exemplet så ser vi på ett decimalt tal som har tre siffror till höger om decimalpunkten

$$b = 10$$

$$F = s_{-1} b^{-1} + s_{-2} b^{-2} + s_{-3} b^{-3}$$

$$(0.625)_{10} = 6 \bullet 10^{-1} + 2 \bullet 10^{-2} + 5 \bullet 10^{-3}$$

$$(0.625)_{10} = 6 \bullet 0.1 + 2 \bullet 0.01 + 5 \bullet 0.001$$

$$(0.625)_{10} = 0.600 + 0.0200 + 0.005$$

## Exempel på Binära ”decimaltal”?

$$b = 2$$

$$(0.101)_2 = 1 \bullet 2^{-1} + 0 \bullet 2^{-2} + 1 \bullet 2^{-3}$$

$$(0.101)_2 = 1 \bullet \frac{1}{2} + 0 \bullet \frac{1}{4} + 1 \bullet \frac{1}{8}$$

$$(0.101)_2 = (0.100)_2 + (0.000)_2 + (0.001)_2$$



## Procedur för bas-konvertering

Exempel: Omvandla $57_{10}$ till binärt tal

| | *kvot* | *rest* |
|---|---|---|
| 57 / 2 = | 28 | 1 |
| 28 / 2 = | 14 | 0 |
| 14 / 2 = | 7 | 0 |
| 7 / 2 = | 3 | 1 |
| 3 / 2 = | 1 | 1 |
| 1 / 2 = | 0 | 1 |

Minst signifikanta biten *(eng. Least Significant Bit)*

Mest signifikanta biten *(eng. Most Significant Bit)*

Svar: $57_{10} = 111001_2$

## Att tänka på vid omvandling

- Syftet med binär representation är att erhålla tal i ett format som passar digital logik
- Ju större noggrannhet ett binärt tal har desto fler bitar krävs → mer digitala kretsar
- Alla tal i en bas kan INTE representeras exakt i en annan bas (avrundningsfel)



## Binära, Oktala och Hexadecimala tal

- Det är lätt att konvertera binära tal till andra, mer lättarbetade format genom att gruppera bitar tillsammans och sedan konvertera till lämplig bas
  - Oktala tal S={ 0, 1, ... , 7 }, basen = 8
    - 3-bitars grupper
  - Hexadecimal S={ 0, ... , 9, A, B, C, D, E, F }, basen = 16
    - 4-bitars grupper

## Exempel: Binär till Oktal omvandling

$N_2 = 1010110100.1$

gruppera i 3-bitars grupper från "decimal"-punkten

$N_2 = 1\,\underline{010}\,\underline{110}\,\underline{100}.1$

fyll ut med nollor för att få ett oktalt tal (en full grupp)

$N_2 = \underline{001}\,\underline{010}\,\underline{110}\,\underline{100}.\underline{100}$

Konvertera varje 3-bitars grupp

$N_8 = 1\ 2\ 6\ 4.\ 4$

$N_8 = (1264.4)_8$



## Exempel: Binär till hexadecimal

$N_2 = 1010110100.1$

gruppera i 4-bitars grupper från "decimal"-punkten

$N_2 = 10\,\underline{1011}\,\underline{0100}.1$

fyll ut med nollor för att få en komplett grupp

$N_2 = \underline{0010}\,\underline{1011}\,\underline{0100}.\underline{1000}$

Konvertera varje 4-bitars grupp

$N_{16} = 2\ \ B\ \ 4\ .\ 8$

$N_{16} = (2B4.8)_{16} = (2B4.8)_H$

## Viktade koder

- Godtycklig vikt kan tilldelas varje position
  - Binary Coded Decimal (BCD)
    - 8, 4, 2, 1
    - exempel, 1001 = 8 + _ + _ + 1 = 9

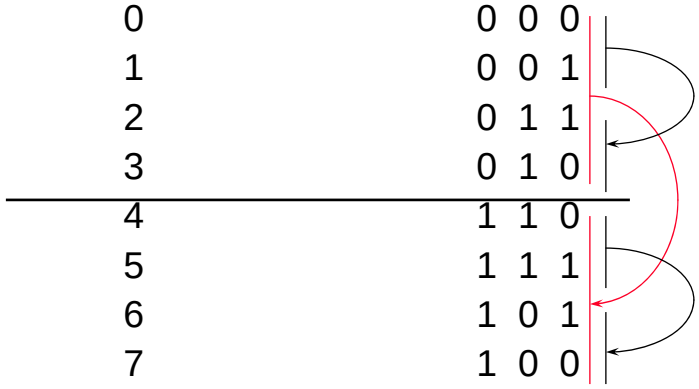


## Icke-viktade koder

- Positionen i talet spelar ingen roll
  - För viktade koder har varje position en vikt som multipliceras med siffran på den positionen
  - Den vanligaste icke-viktade koden kallas för Gray kod

## 3-bitars Gray kod

| *Ordning* | *Kodord* |
|---|---|
| 0 | 0 0 0 |
| 1 | 0 0 1 |
| 2 | 0 1 1 |
| 3 | 0 1 0 |
| 4 | 1 1 0 |
| 5 | 1 1 1 |
| 6 | 1 0 1 |
| 7 | 1 0 0 |





## Alfanumeriska koder

- Alfanumeriska koder representerar både
  - Decimala siffersymboler
    - 0 - 9
  - Alfabetets tecken
    - A – Z, a – z
  - Övriga skrivbara tecken
    - T.ex: %, &, ?, *, @
  - Styrsymboler
    - Blanksteg, ny rad, etc.
  - Standardkoder: **ASCII**, EBCDIC, etc.

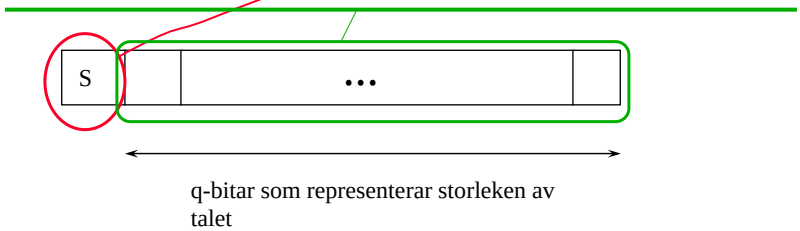
## ASCII

- ASCII kodning
  - *American Standard Code for Information Interchange*
  - ASCII-tabell
    - Ger hexadecimal kod
    - Rad: minst signifikanta positionen
    - Kolumn: mest signifikanta positionen
    - Exempel: ASCII('C') = $43_{16}$



## Negativa tal

- teckenbit
  - Biten längst till vänster (S) representerar talets tecken
    - 1 negativt
    - 0 positivt
  - Bitarna till höger om teckenbiten är storleken på talet



## Två-komplement

- Två-komplement representation
  - För ett n-bitars tal
    - är vikten för MSB –2n-1 (istället för +2n-1 )
    - övriga bitars vikter är samma som för binär kodning
  - Procedur för att utföra två-komplement
    - invertera samtliga bitar i talet
    - addera 1 till talet
  - Exempel:
  - Bilda två-komplement till 8-bitars talet $00010001_2$ (=$17_{10}$)
  - $00010001_2$
  - 11101110
  - + 1
  - $11101111_{2\text{-k}} = -17_{10}$



## Addition och subtraktion

- Exempel:

3 + 2 = ?

0011
0010
0101 = $5_{10}$

7 - 1 = ?

0111
1111
1 0110 = $6_{10}$

-1 - 3 = ?

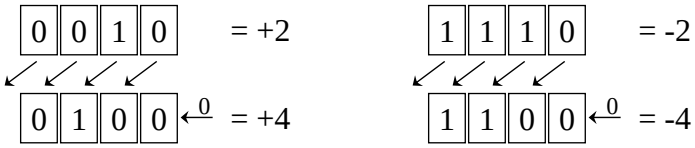
1111
1101
1 1100 = $-4_{10}$

2 - 3 = ?

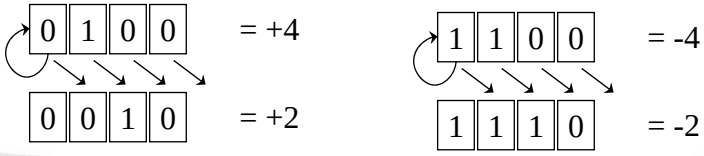
0010
1101
1111 = $-1_{10}$

## Multiplikation/Division med 2

- Skifta det binära talet ett steg vänster




- Skifta det binära talet ett steg höger






## Boolesk Algebra – Definitioner

**Konstanter**

| | |
|---|---|
| 0 | (Falsk) |
| 1 | (Sann) |

**Operationer**

| | |
|---|---|
| + | (ELLER) |
| · | (OCH) |
| ’ | (ICKE) |

**Axiom**

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

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

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

$0' = 1$
$1' = 0$

## Räknelagar för en variabel

| | |
|---|---|
| $x + x = x$ | $x + 1 = 1$ |
| $x \cdot x = x$ | $x \cdot 0 = 0$ |
| $x + x' = 1$ | $x + 0 = x$ |
| $x \cdot x' = 0$ | $x \cdot 1 = x$ |
| | $(x')' = x$ |

Dessa räknelagar kan enkelt visas utifrån axiomen.

Visa att $x + x = x$

Låt $x = 1$: $1 + 1 = 1$

Låt $x = 0$: $0 + 0 = 0$

V.L = H.L



## Räknelagar för flera variabler

- Associativa lagar
  $x + (y + z) = (x + y) + z$
  $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
- Kommutativa lagar
  $x + y = y + x$
  $x \cdot y = y \cdot x$
- Distributiva lagar
  $x \cdot (y + z) = x \cdot y + x \cdot z$
  $x + y \cdot z = (x + y) \cdot (x + z)$

## Räknelagar för flera variabler

- absorptionslagar
  x + x·y = x
  x·(x + y) = x

$x + x \cdot y = x \cdot (1 + y) = x \cdot 1 = x$

$x \cdot (x + y) = x \cdot x + x \cdot y = x + x \cdot y$
$= x \cdot (1 + y) = x \cdot 1 = x$

- Concensuslagen
  $x \cdot y + x' \cdot z = x \cdot y + x' \cdot z + y \cdot z$

Augustus de Morgan

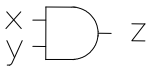
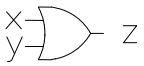
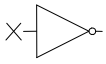
- De Morgans lag
  $(x + y)' = x' \cdot y'$
  $(x \cdot y)' = x' + y'$

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

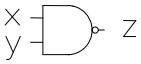
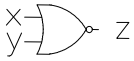
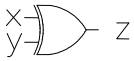
$\overline{x \cdot y} = \overline{x} + \overline{y}$



## Grundläggande logiska grindar

| Namn/operator | Symbol | Funktion | Logisk operation |
|---|---|---|---|
| OCH, *eng. AND*<br>· |  | X Y Z<br>0 0 0<br>0 1 0<br>1 0 0<br>1 1 1 | Z = X • Y |
| ELLER, *eng. OR*<br>+ |  | X Y Z<br>0 0 0<br>0 1 1<br>1 0 1<br>1 1 1 | Z = X + Y |
| ICKE, *eng. NOT*<br>' |  | X Z<br>0 1<br>1 0 | Z = X' |

## Grundläggande logiska grindar

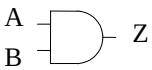
| Namn/operator | Symbol | Funktion | Logisk operation |
|---|---|---|---|
| *NAND* |  | X Y Z<br>0 0 1<br>0 1 1<br>1 0 1<br>1 1 0 | $Z = \overline{X \cdot Y}$<br>$Z = (X \cdot Y)'$ |
| *NOR* |  | X Y Z<br>0 0 1<br>0 1 0<br>1 0 0<br>1 1 0 | $Z = \overline{X + Y}$<br>$Z = (X + Y)'$ |
| *XOR* $\oplus$ |  | X Y Z<br>0 0 0<br>0 1 1<br>1 0 1<br>1 1 0 | $Z = X \oplus Y$ |



## Olika sätt att representera logiska funktioner

- Sanningstabell
- Grindnät
- Boolesk algebra
- Normalform

## Sanningstabell

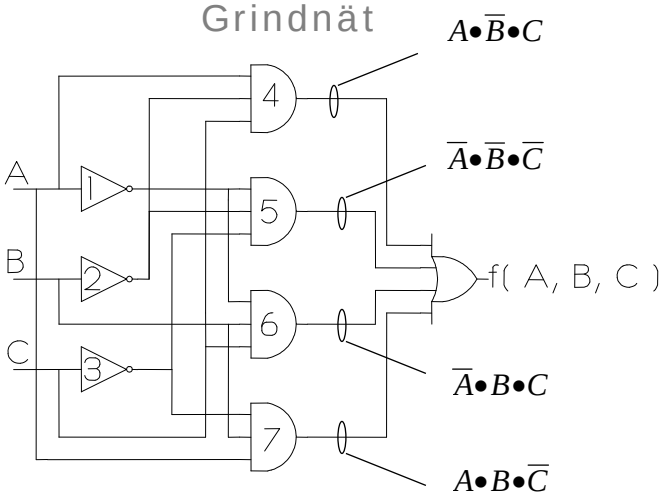



Z = A • B

| Ingångar | | Utgång |
|---|---|---|
| A | B | Z |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



## Grindnät




$$f(A,B,C) = A\overline{B}C + \overline{A}\overline{B}\overline{C} + \overline{A}BC + AB\overline{C}$$

## Boolesk algebra

- Algebraisk manipulering
  - Visa att: $x + yz = (x + y)(x + z)$

$$
\begin{aligned}
(x + y)(x + z) &= xx + yx + xz + yz \\
&= x + yx + xz + yz \\
&= x + xy + xz + yz \\
&= x + x(y + z) + yz \\
&= x(1 + y + z) + yz \\
&= x + yz
\end{aligned}
$$



## SLUT på Föreläsning 1