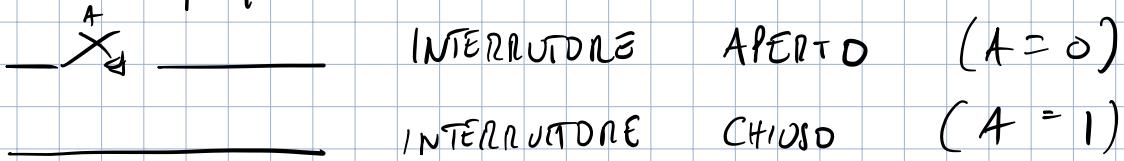


Interruttori e lampadina
sono in RELAZIONE.

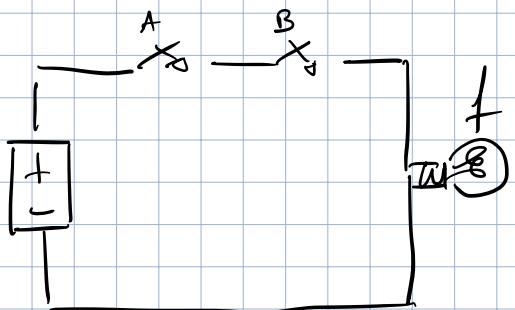
Per semplificazione:



Se f indica la lampadina con "f" allora

- $f = 0$ SPENTA
- $f = 1$ ACCESA

FUNZIONE "AND" (CONGIUNTIONE)

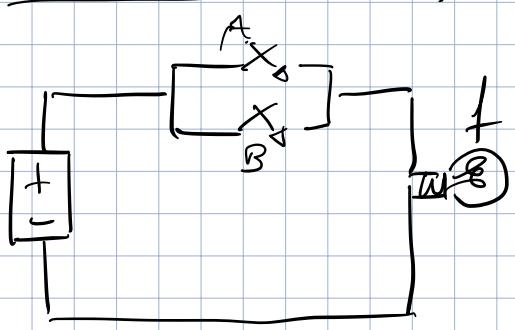


A	B	f
0	0	0
1	0	0
0	1	0
1	1	1

$f = 1$ solo se A AND B sono chiusi.

$$f(a, b) = a \text{ AND } b = ab$$

FUNZIONE "OR" (DISGIUNZIONE)

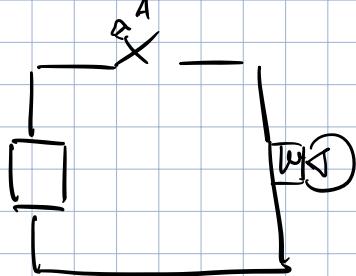


A	B	f
0	0	0
1	0	1
0	1	1
1	1	1

$f = 1$ job se $A \text{ OR } B$ job chiusi.

$$f(a, b) = a \text{ OR } b = a + b$$

FUNZIONE "NOT" (complementare)



$$\begin{aligned} f &= 1 \text{ job se } A = 0 \\ f(a) &= \text{NOT } a = \bar{a} = a' \end{aligned}$$

A	f
1	0
0	1

Queste funzioni possono essere composte

$$- [\overline{B}] - \quad f(a, b, c) = (a+b)c$$

Ogni implementazione ha soggiorni
per doppio le possibili combinazioni

Analisi: informazioni può essere rappresentata come una sequenza di: 0, 1. Infatti AND, OR, NOT sono definiti nell'insieme Booleano $\{0, 1\}$.

TEORIA DEGLI INTERRUATORI

Sia AND, OR, NOT può essere implementata con una rete di interruttori. Tutte le espressioni Booleane possono essere ricritte con una combinazione di queste 3.

Digitale = ha due definizioni

↳ MODERNA: caratterizzata da tecnologia elettronica

ESATA: Rappresentato con una sequenza finita di segni messi da un alfabeto finito.

COMUNICAZIONE:

La comunicazione risale già a 200.000 a.C., mentre la prima forma di Algebra o di Algoritmo è di Muhammad ibn Musa al-Khwarizmi.

Cenni Storici:

1455 Gutenberg, stampa a caratteri mobili

1840 Morse, telefono e codice Morse

1847 Boole, scrive un Trattato su logica, algebra

1847 De Morgan, scrive un Trattato su logica, algebra

1936 Turing, crea la sua macchina

Qualunque problema se posto in modo esatto può essere risolto dalla Macchina d. Turing

1938 Shannon, conia il "Bit", e scrive trattati sulla logica

1965 Moore's Law: la capacità di calcolo raddoppia ogni 1,5 anni

INFORMAZIONE

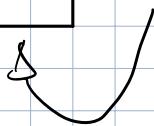
: Riduzione di incertezza

La salta elementare fra due option' è la minima quantita' di informazione, secondo questa acczione del fammme

PAROLA: Una sequenza di caratteri o cifre.

Che siamo numeri o parole, utilizzando un numero finito di caratteri si parla di **CODIFICA**.

--	--	--	--



$\left\{ \begin{array}{l} \{0, 1, 2, 3\} \\ \{0, 1\} \\ \{A, B, \dots, Z\} \end{array} \right\}$ Ci sono S^n configurazioni diverse di "n" caratteri da un alfabeto di S simboli.

Ex,

Nelle Tonghi si usano numeri (0-9) e lettere (A-Z)

$$\begin{array}{|c|c|c|c|c|c|c|} \hline A & B & | & 2 & 3 & C & D \\ \hline 26 & 26 & 10 & 10 & 10 & 26 & 26 \\ \hline \end{array} = 26^6 \cdot 10^3$$

Codice: Un' element. sintattico che puo' essere descritto come un insieme di parole o regole.

PROPRIETA' delle regole di CODIFICA:

- Non ridondanza: Se ognuno degli elementi ha a disposizione una sola codifica
- Lunghezza: Se tutte le codifiche hanno stessa lunghezza.
- Esattezza: Se ci permette di rappresentare in modo non ambiguo tutte le informazioni che vogliamo rappresentare.

Esempio:

Nelle Tonghi ogn' macchina ha la sua codifica.

Le Tonghi sono:

ESATTE: non vi sono due auto con stessa Tonga

LUNGHEZZA COSTANTE: tutte le Tonghi hanno stessa lunghezza

NON RIDONDANTI: ad ogni auto è associata una sola Tonga.

CODIFICA DI COMBINAZIONI FINITE

Sarà il numero minimo di caratteri ol' lunghezza costante per esprimere esatta combinazione ol' M elementi.

$$N = \lceil \log_2 M \rceil - \text{APPROXIMAZIONE PER ECESSO}$$

$$S^N \geq M$$

Bit necessari per

codificare 8.000.000

informazioni 8.1000.1000

$$2^3 \cdot 2^{10} \cdot 2^{10} = 2^{23}$$

CONDIFICA DI COMBINAZIONI INFINITE

Un insieme infinito non è rappresentato digitalmente, quindi devono essere ristretti:

INFINITI INFINTAMENTE DENSI: $\{m, m \in M\} \cup \{x \in M \mid m < x < m\}$

Discutibilità: Partitionare l'insieme in un numero finito di sottoinsiemi che non si sovrappongono e che coprono tutto l'insieme di partenza.

Operazioni Binarie:

SOMMA

+	0	1
0	0	1
1	1	1

PRODOTTO

*	0	1
0	0	0
1	0	1

Questa "normale" che abbiamo utilizzato è la rappresentazione modulare degena, dove per rappresentare il numero negativo decidiamo una cella di segno, dove 1 è negativo.

$$3 = 0 \ 0011 \quad -3 = 1 \ 0011$$

Nella codifica complemento od uno innanzo gli 0 e 1

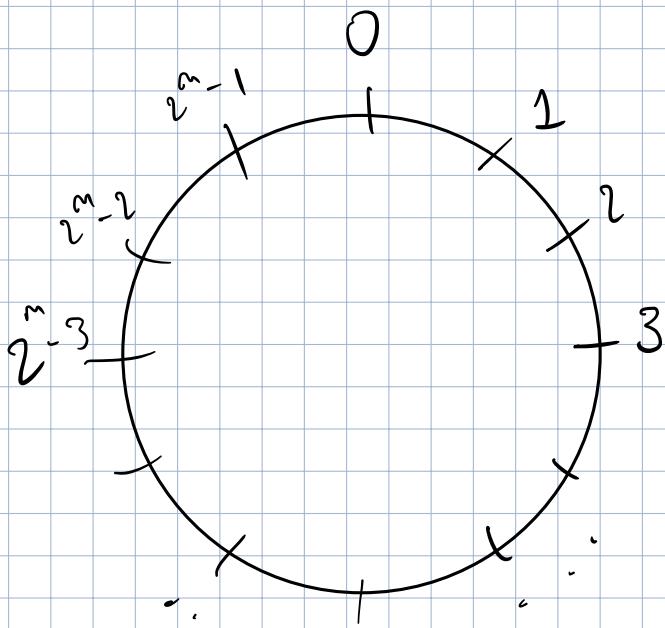
$$3 = 0011 \quad -3 = 1100$$

Per passare da complemento a 1 a 2, sommo 1

$$(-3)_{\text{COMP } 1} = 1100$$

$$(-3)_{\text{COMP } 2} = 1101$$

Se abbiamo un numero pulito di numeri si avrà ad un massimo dopo cui si torna alla prima cifra possibile



Nella numerazione posizionale è importante leggere da destra,

162

0	0	1	6	2
---	---	---	---	---

$\begin{array}{l} 10^0 \\ 10^1 \\ 10^2 \\ 10^3 \end{array}$

Le celle devono essere piene perché il resto equivalgono ad un tuo valore.

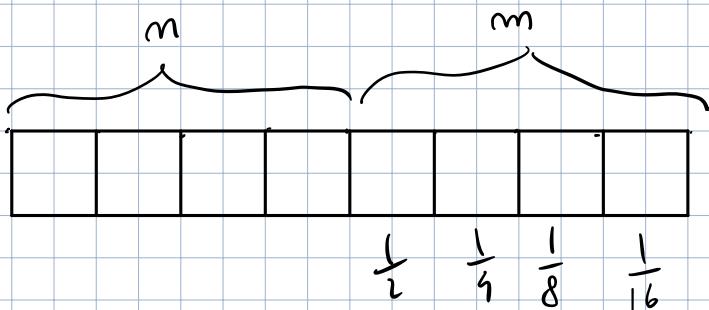
$$101_2 = 5_{10} = 5_8 = 10_5$$

Do base b a decimali:

$$C_{m-1} \cdot 2^{m-1} + C_{m-2} \cdot 2^{m-2} + \dots + C_3 \cdot 2^3 + C_2 \cdot 2^2 \cdot C_1 \cdot 2 + C_0$$
$$= (C_{m-1} \cdot 2^{m-1} + C_{m-2} \cdot 2^{m-2} + \dots + C_3 \cdot 2^3 + C_2 \cdot 2^2 \cdot C_1 \cdot 2) \cdot 2 \cdot C_0$$

Il resto delle divisioni è la prima cifra della rappresentazione.

ESPRESSIONE NUMERI FRAZIONARI



$$\frac{2^{m+m} - 1}{2^m} = 2^m - \frac{1}{2^m}$$

Rappresentazione in singola pizza: indica al calcolatore un numero di bit dedicato alla parte intura e quella frazionaria.

Ex.

$$\begin{array}{r} 0,625 \cdot 2 \\ 0,25 \cdot 2 \\ 0,5 \cdot 2 \end{array} \quad \begin{array}{l} 1,25 \\ 0,5 \\ \downarrow 1,0 \end{array}$$

101

0,1₁₀ è periodico in base 2.

Se utilizziamo la rappresentazione complemento a 2 possiamo scrivere numeri negativi con positivi, sommando più numeri di segno zero potremmo ottenere un **OVERFLOW**.

FLOATING POINT NUMBER

- S = segno (1 bit)
- M = mantissa (m bit)
- b = base (b bit)
- SE = segno esponente (1 bit)
- E = esponente (e bit)

$$S \cdot O \cdot M \cdot b^{SE \cdot E}$$

Ese.

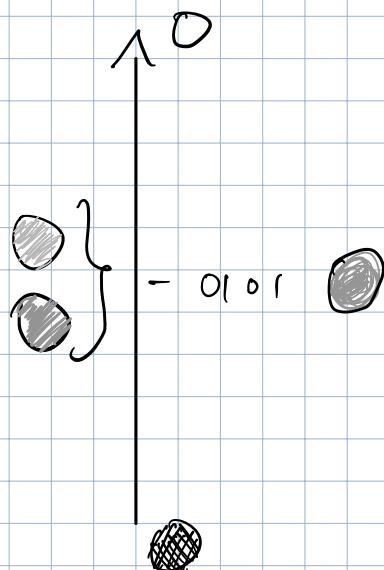
1,25 -> 01,01 > 0,1101

[0|0|1|0|1|0]1[0|]

Per scegliere da dove far partire la mantissa
parto dall' 1 più grande.

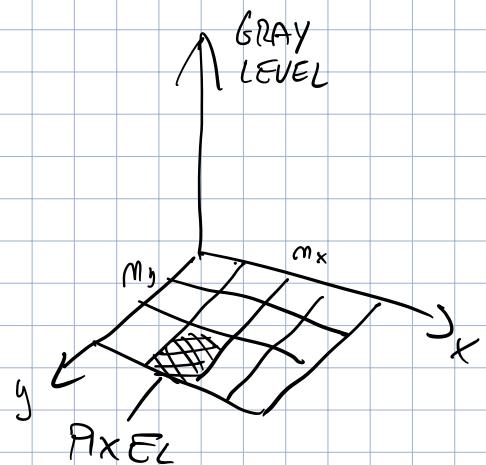
CODIFICA DEI COLORI

Se prendiamo una scia di colori rappresentati da m-bit, i colori non rappresentabili a causa della scarsa capacità di memoria gioveranno "approssimati".



DIMENSIONE IMMAGINE 2D

$$\text{SIZE: } m_x \cdot m_y \cdot \log_2 m_{\text{col}}$$



$$\text{Ex. } 1910 \times 1080 \cdot \log_2 16 \cdot 10^3$$

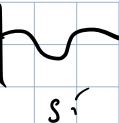
11

$$1990 \times 1080 \cdot 8 \text{ byte.}$$

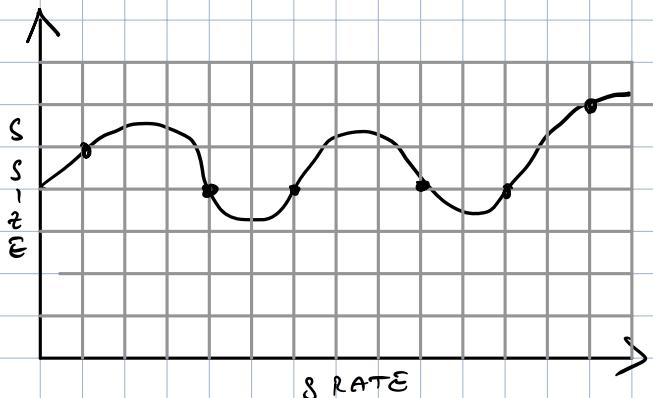
Nella sintesi additiva si usano 3 colori della luce per creare il resto, noi utilizziamo RGB, c'è dicono un bit per colore.

SEGNALE: una grandezza fisica che varia nel tempo.

Con questa definizione molte cose possono essere definite.

Analogici: valori con continuità  s

Dobbiamo però discutere i segnali:



Il segnale è rappresentato
soltanto quando lo attraversi
ma non nel mezzo.

Nel corso del corso parliamo di "compionimento"
Nel corso dei segnali parliamo di "quantificazione"

VIDEO

$$S_{ITE} : S_{RATE} \cdot T \cdot S_{site} = S_{RATE} \cdot T \cdot \log_2 m_a \cdot m_x \cdot m_y$$

Ex:

$$S_{RATE} = 16 \text{ kHz}$$

$$T = 2'$$

$$m_w = 156$$

$$S_{ITE} = (16k) \cdot (2 \cdot 60) \cdot (\log_2 256)$$

$$F_{ITE} : 25 \text{ pps}$$

$$T = 1'$$

$$m_x \cdot m_y = 1080 \cdot 1920$$

$$m_w = 16 \text{ M}$$

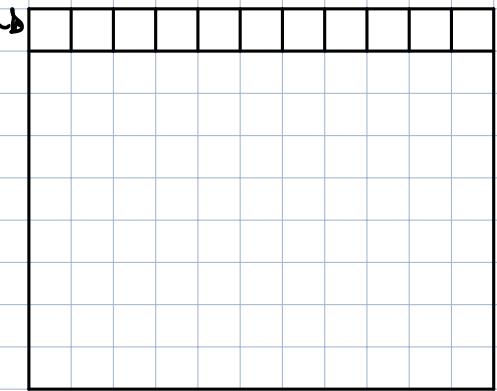
$$S_{ITE} = (25) \cdot (60) \cdot (1080 \cdot 1920) \cdot \log_2 (16 \text{ M})$$

COMPRESSESIONE

Arendo una linea binaria

In un'immagine posso semplicemente dire al posto di "biombo" 1000 volte potrei dire (biombo, 1000)

RLE = Run length encoding



Per i video esiste la compressione MPEG che divide in quadrettini il video per vedere se puoi ristabilire vecchi frame.

ERROR MODEL

La probabilità che una parola di m bit non abbia nessun errore è:

$$P_w = 1 - (1-p)^m \approx mp$$

La probabilità che almeno un bit abbia errore è:

$$\frac{\binom{m}{1} \cdot p \cdot (1-p)^{m-1}}{m!} \approx mp \left(1 - (m-1)p\right)$$

Se p è molto piccolo: $mp - m(m-1)p \approx mp = O(p)$

Se l'errore è più di un altro la probabilità è $O(p^m)$.

PROPORTIONE A p

Essendo gli errori multipli esponenzialmente più improbabili considerare la probabilità di errori singoli.

Distanza di Hamming: Tra 2 parole di m bit.

0 1 1 0
0 1 0 1
xx

la minima distanza tra due parole da 8 bit è 1.

TRA codice: $d_H(\text{code}) = \min \left\{ d_H(w_i, w_j) \right\}$

 $w_1, w_2 \in \{0, 1\}$

Alcuni codici hanno un bit di controllo a cui dedicano gli errori.

Ese.

(1) 00
(2) 01
(3) 10
(4) 11

$d_H = 1$

(1) 0000
(2) 0101
(3) 1010
(4) 1111

$d_H = 2$

Se un bit ha un errore =

$$\begin{array}{r} 0101 \\ + \\ 0111 \\ \hline \end{array}$$

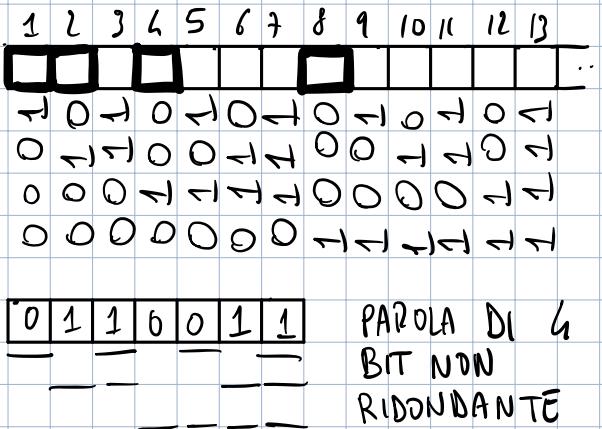
Non esiste nella matrice codifica.

m : bit di info

$$2^m \geq m + r + 1$$

m : bit di controllo

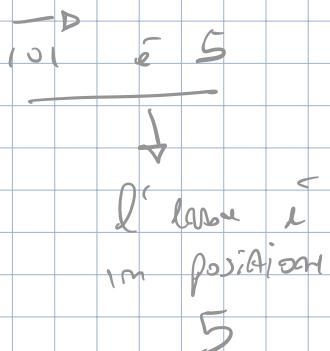
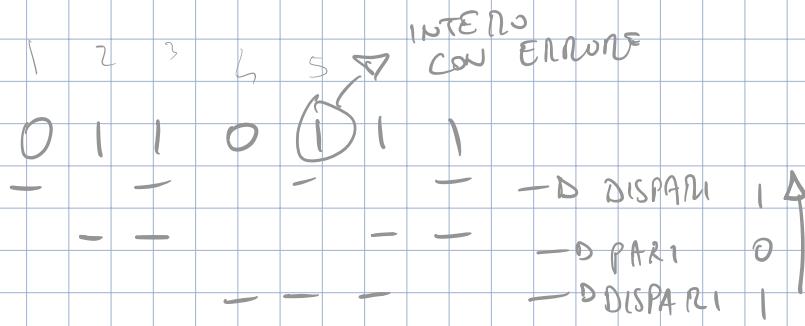
CODIFICA DI HAMMING



Se riserviamo i bit di controllo alle potenze intive di 2, noto che ammontano i bit di info. alle caselle libere la regola citata prima vale:

- Il primo bit di controllo, controlla la parità dei bit di informazione con 1
- Il secondo bit di controllo, controlla la parità dei bit di hamming un' 1 nella seconda posizione.
- Il terzo bit di controllo, controlla la parità dei bit di hamming un' 1 nella terza posizione.
- ⋮

Ex,

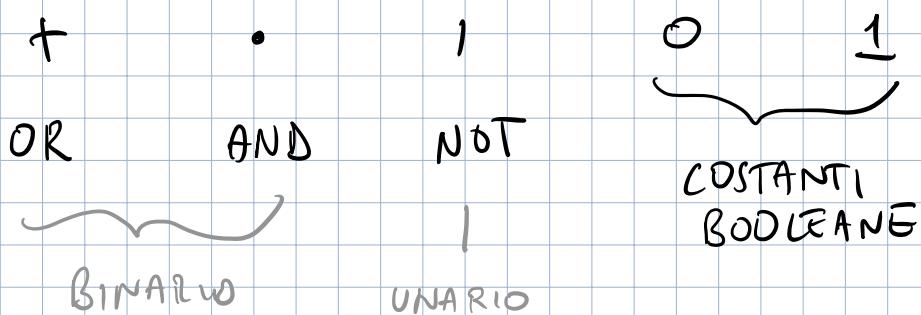


ALGEBRA BOOLEANA

Sistema matematico basato su:

- elementi
- operatori
- assiomi

$$A = (A, +, \cdot, ', 0, 1)$$



Axiomi:

- | | SOMMA | |
|-----------------|---|---|
| • Commutativa: | $x+y = y+x$ | PRODOTTO |
| • Identità: | $x+0 = x$ | $x \cdot 1 = x$ |
| • Distributiva: | $x+(y \cdot t) = (x+y) \cdot (x+t)$ | $x \cdot (y+t) = (x \cdot y) + (x \cdot t)$ |
| • Complemento: | $\forall x \in A \exists x' \in A : x + x' = 1$ | $x \cdot x' = 0$ |

$$A = \{a, a', b, b', \dots\}$$

FUNZIONE
BOOLEANA

$$B^m \rightarrow B$$

ALGEBRA DI BOOLE DI COMMUTAZIONE (SWITCHING)

$$\mathcal{B} = (\mathcal{B}, +, \cdot, 1, 0, \bar{1})$$

- Insieme di definizione: $\mathcal{B} = \{0, 1\}$
- Variabili Booleane: $x \in \{0, 1\}$
- Funzione Booleana: $t = f(x_1, x_2, \dots, x_m)$ $f: \mathcal{B}^m \rightarrow \mathcal{B}$

and: $\mathcal{B} \times \mathcal{B} \rightarrow \mathcal{B}$
 (x, y)

$$t = x \cdot y = xy$$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

or: $\mathcal{B} \times \mathcal{B} \rightarrow \mathcal{B}$
 (x, y)

$$t = x + y$$

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

not: $\mathcal{B} \rightarrow \mathcal{B}$
 x

$$t = x' = \bar{x}$$

x	y
0	1
1	0

Nella TABELLA DELLA VERITÀ il numero di righe che associa un valore booleano ad ogni configurazione di m variabili è 2^m .

Le funzioni si possono anche rappresentare in altri modi:

PROPRIETÀ:

Commutative laws

$$xy = yx$$

$$x+y = y+x$$

Identity elements

$$x \cdot 1 = x$$

$$x+0 = x$$

Distributive laws

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

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

Complement laws

$$x \cdot x' = 0$$

$$x+x' = 1$$

Associative laws

$$xyz = x(yz) = (xy)z$$

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

Null laws (forcing elements)

$$x \cdot 0 = 0$$

$$x+1 = 1$$

Idempotent laws

$$x \cdot x = x$$

$$x+x = x$$

Absorption laws

$$x+xy = x$$

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

De Morgan's laws (duality principle)

$$(xy)' = x'+y'$$

$$(x+y)' = x'y'$$

SOMMA DI MINTERMINE

Forme di prodotto in cui appaiono forme diverse delle variabili.

$$f(a,b,c) \quad abc \quad ab'c \quad a'b'c' \quad \dots$$

6 LETTERALI

$a b c$	$\overbrace{ab'c + a'b'c}^f = f$
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	0

Il numero di variabili che appaiono in forma
nella o negata.

Ogni funzione Booleana è rappresentabile con tanti minitermini quanti ne servono perché le variabili valgono 1

$$\begin{aligned}
 \text{Ex. } f &= a'b'c' + a'b'c + ab'c' + abc' + abc = \\
 &= c'(a'b' + a'b + ab' + ab) + abc = \\
 &= c'(b'(a' + a) + b(a' + a)) + abc = \\
 &= c'(b' + b) + abc = \\
 &= c' + abc = \text{ POTREBBE ANDARE BENE MA NON È LA FORMA MINIMA}
 \end{aligned}$$

$$= c'(a' + a')$$

$$= c'a + c'a' + abc =$$

$$= a(c' + bc) + c'a =$$

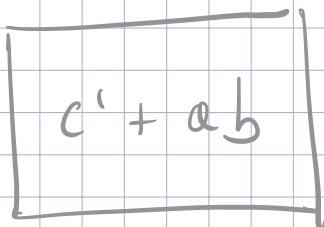
$$\begin{aligned} x + yx &= x \\ x + x'y &= x + y \end{aligned}$$

$$c' + bc = c + c'b + cb = c' + b(c + c') = c' + b$$

$$a(c' + b) + ac' =$$

$$ac' + ab + a'c' =$$

$$c'(a' + a) + ab =$$



Potrà capitare che appaiano condizioni di indifferenza in cui non è possibile, o non è osservabile l'output di quella funzione.

RETI LOGICHE

Porta Logica: Componente che prende in ingresso uno o più segnali logici e fornisce in uscita un segnale logico il cui valore è una configurazione del segnale all'ingresso.

Le porte logiche sono implementate come reti di interconnessioni. L'output di una porta logica sarà sempre 0 o 1.

Esempi di porte logiche:

$$\neg \rightarrow \text{NOT} \quad f = a' = a$$

$$\wedge \rightarrow \text{AND} \quad f = ab$$

$$\vee \rightarrow \text{OR} \quad f = a + b$$

$$\overline{\wedge} \rightarrow \text{NAND} \quad f = (ab)'$$

$$\overline{\vee} \rightarrow \text{NOR} \quad f = (a+b)'$$

$$\overline{\wedge} \rightarrow \text{EXOR} \quad f = (ab' + a'b)$$

$$\overline{\vee} \rightarrow \text{EXNOR} \quad f = (ab' + a'b)'$$

		EXOR
a	b	
0	0	0
0	1	1
1	0	1
1	1	0

		EXNOR
a	b	
0	0	1
0	1	0
1	0	0
1	1	1

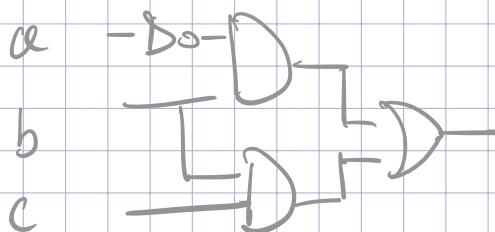
↓

$$(ab' + a'b) = (ab')' \cdot (a'b)' = (a'+b)(a+b') = \\ = a'a + a'b' + ab + bb' = a'b' + ab$$

RETI BOOLEANE

Sono reti con porte logiche collegate in base alle espressioni booleane.

$$f = a'b + bc$$



FAMIGLIA LOGICA

Libreria: Insieme di tipi di porte logiche

{ AND, OR, NOT }

{ NAND }

Se non contiene uno di questi due, non è completa.

Come riguarda le funzioni Booleane con il NAND.

NOT $a - \overline{D}o - a'$

AND $\begin{matrix} a \\ b \end{matrix} - \overline{D}o - \overline{D}o - ab$

OR $\begin{matrix} a \\ b \end{matrix} - \overline{D}o - \overline{D}o - a+b$

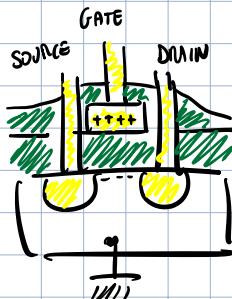
Un AND con ingressi negati è un NOR.

Un OR con ingressi negati è un NAND.

Un OR con ingressi negati è un NOR.

TRANSISTOR

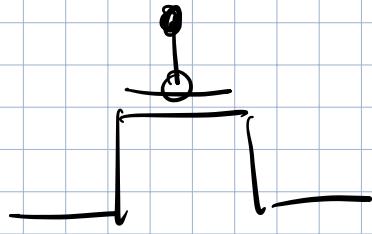
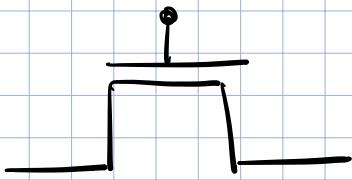
Metallo, Ossido, Silicio Field Effect Transistor



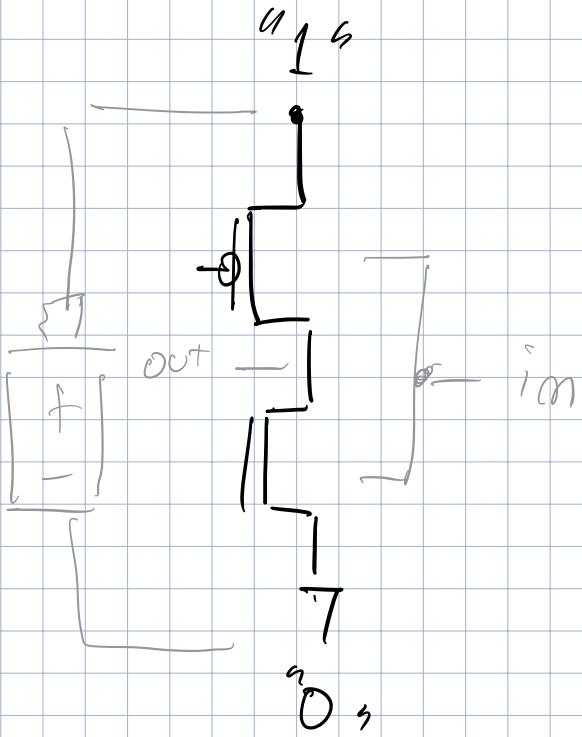
n MOS FET



p MOS FET



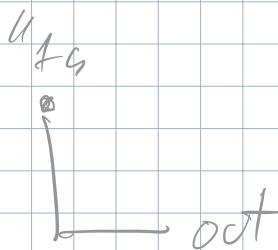
INVERTER



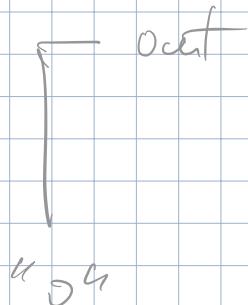
$$im - D-o- out = im'$$

im	out
0	1
1	0

$$im = \phi$$



$$im = 1$$



Questo tipo di rette logiche non ha componenti
di potenza statica.

Questa famiglia logica è detta **CMOS FET**.

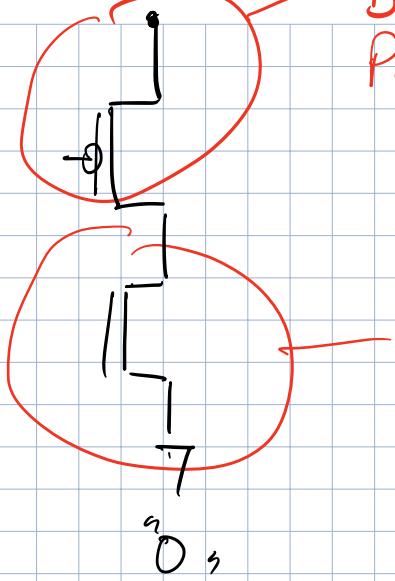
"1s

CIRCUITO

D1

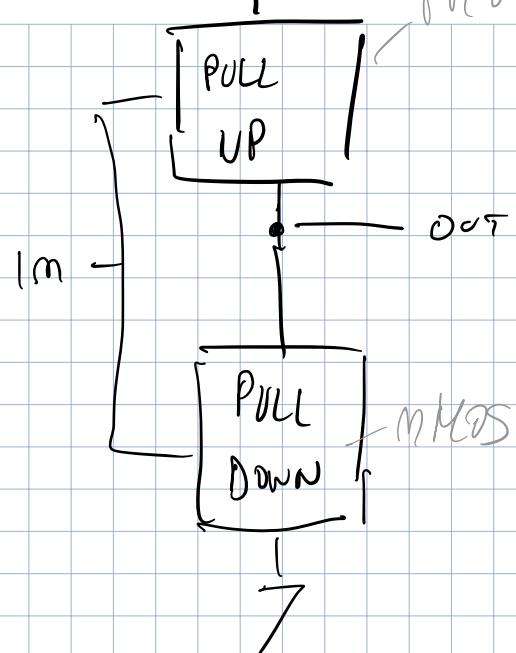
i

PIMPS



PULL-UP

CIRCUITO
DI
PULL-DOWN

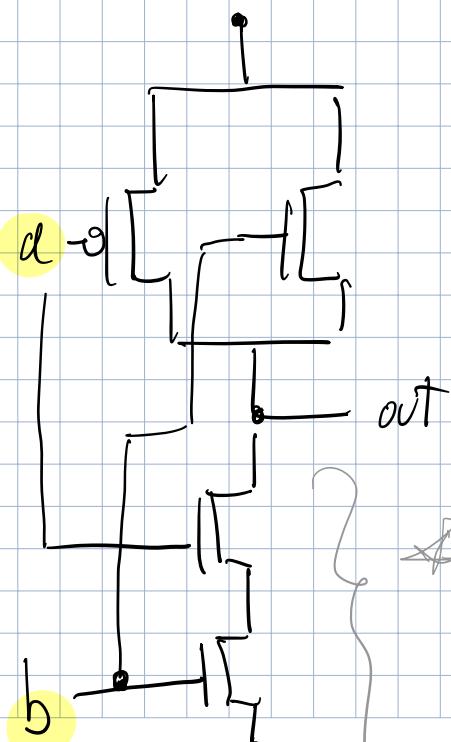


Ex

NAND

$$\frac{a}{b} \cdot \overline{D} \cdot \overline{D} \cdot \text{out} = (ab)' = a' + b' \quad \begin{cases} \text{PULL UP} \\ \text{De Morgan} \end{cases}$$

Ude



If PULL DOWN dev
where output opposite of
PULL UP inverter
 $((ab)')' = ab$

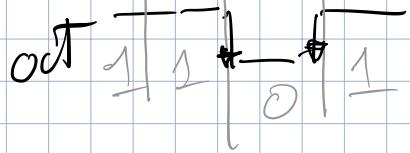
a	0	1	1	0
b	0	0	1	1

$a = \overline{D}$ $\text{out} = D$

7

b

b - 1

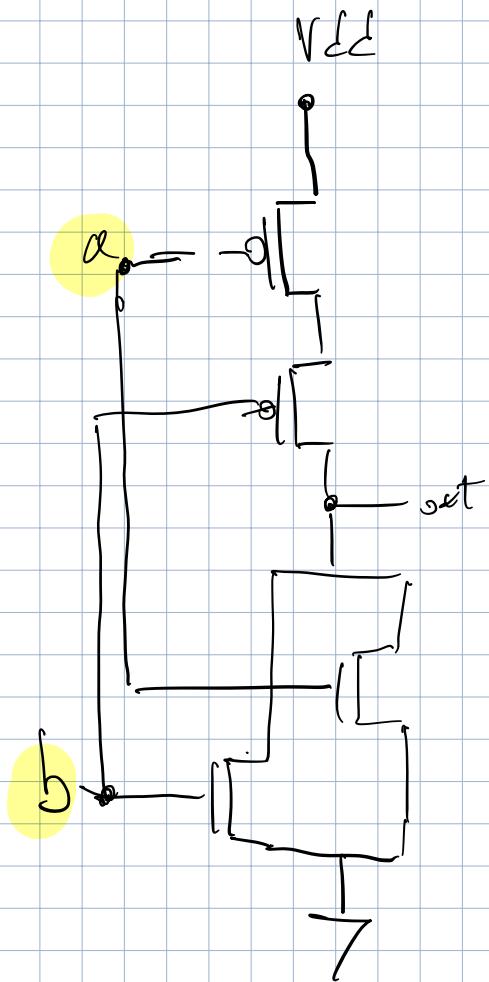
NOR

$$\begin{array}{l} a \\ b \end{array} \rightarrow \text{O} - \text{o} \text{ut}$$

Come rappresentiamo
se una rette i
rappresentabile
in
FCMOS?

11
✓

Dove sono tutti i letterali
in forma negata

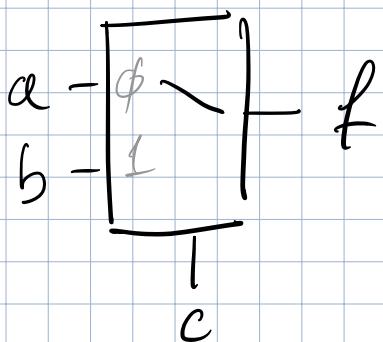


CIRCUITI COMBINATORI

Circuiti che implementano funzioni Booleane.

Il valore logico assegnato ad un segnale di output è una funzione Booleana con la corrente configurazione di input.

MUX : MULTIPLEX



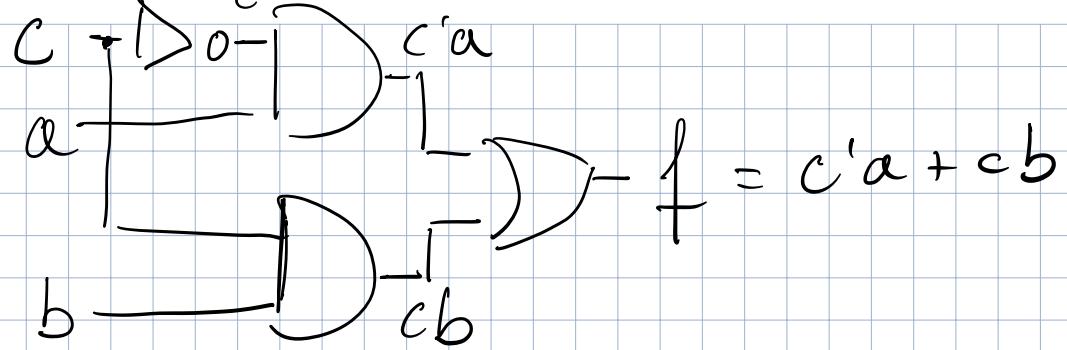
$$\begin{array}{ll} \text{Se } c = 0, & f = a \\ \text{se } c = 1, & f = b \end{array}$$

Ex.

c	a	b	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

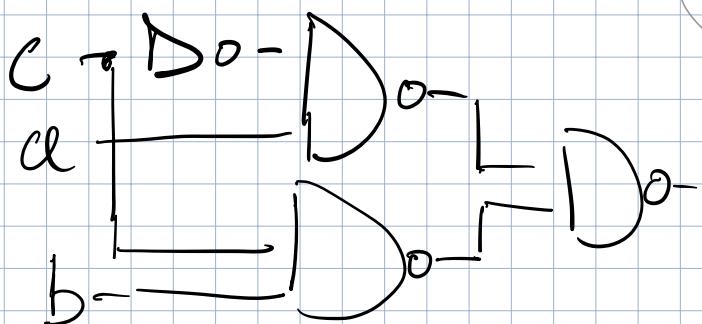
$$\begin{aligned} f &: c'ab' + c'ab + ca'b + \\ &\quad cab = \\ &= c'(ab' + ab) + c(a'b + ab) \\ &= c'(a(b' + b)) + c(b(a' + a)) \\ &= c'(a) + c(b) \\ &= c'a + cb \end{aligned}$$

If TAA NISSON



n

16 Transistor



CAD

COMPUTER ASSISTED DESIGN

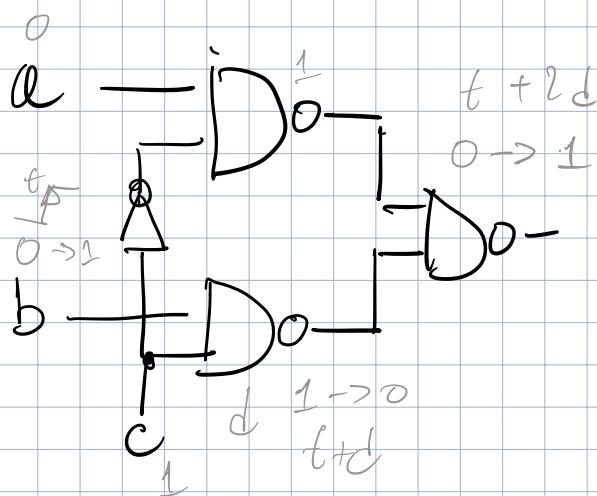
Software di design sostituendo ai Tecnigrafi.

Tk gate N.B. Non fare copia e incolla

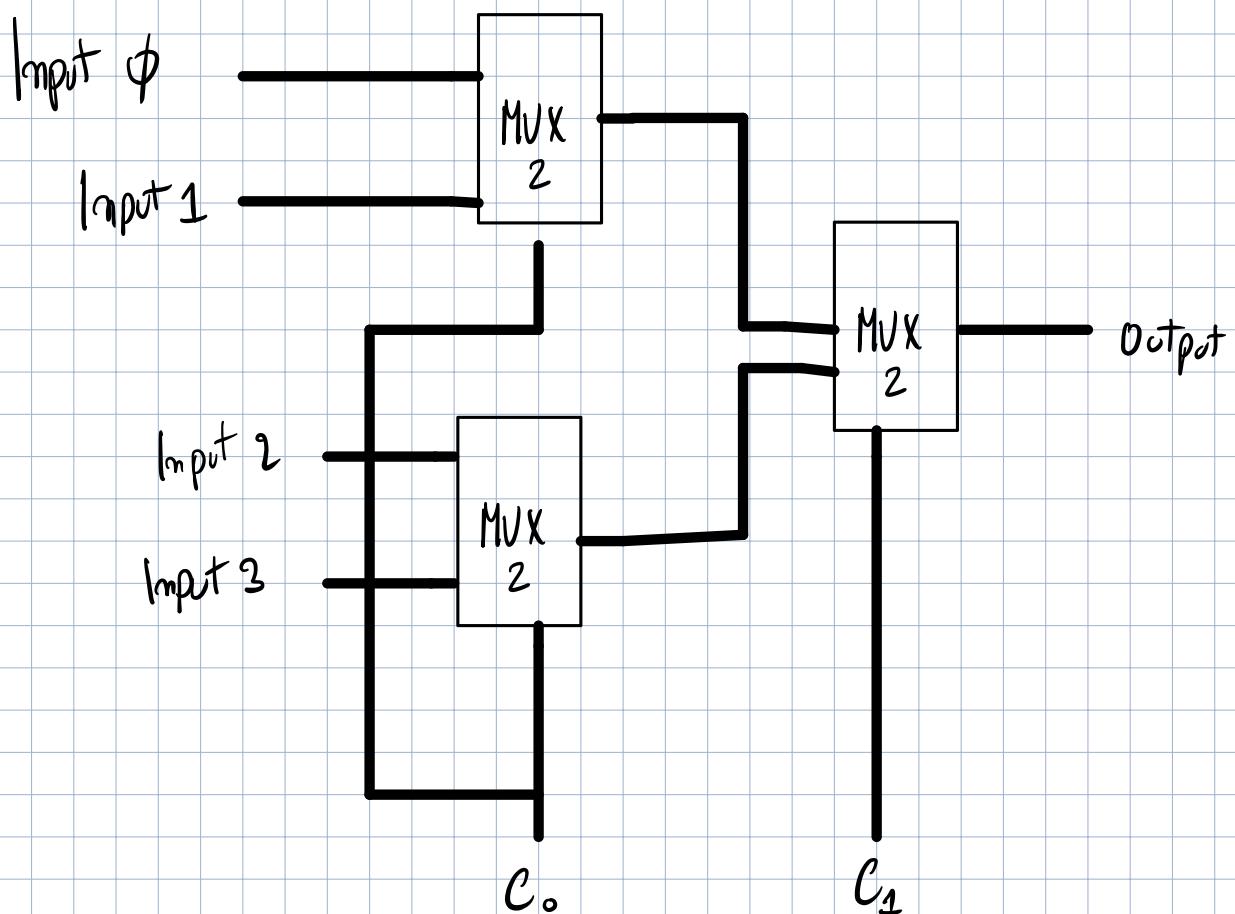
Per trattare il tempo nel simulatore ci sono
due metodi:

- dividere il tempo in intervalli ed analizzarli di conseguenza;
- Cycle Accurate:

MUX 2

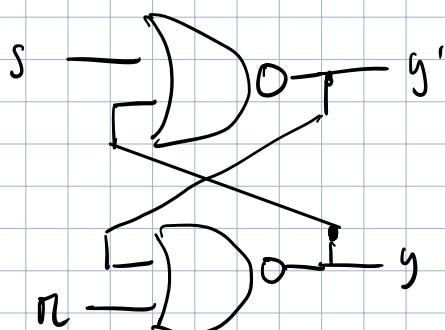


MUX 4



CIRCUITI SEQUENZIALI

Quelli che abbiamo già visto prima erano CIRCUITI COMBINATORI, in cui i segnali di uscita erano funzione di quell. di ingresso, mentre i circuiti SEQUENZIALI hanno memoria quindi non dipendono unicamente dall'input, ma alterando il proprio stato mantengono lo stato del circuito (memoria finita per stati finiti).



s	r	y	y'
0	0	?	?
0	1	0	1
1	0	1	0
1	1	0	0

Per risolvere questo problema immaginiamo che $y = y'$ abbiano valore.

Ese

s	r	y	y'
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

Se gli applichi il criterio delle un circuito con uscite oscillante.

Non è più corretto parlare di disponibilità indipendente ma

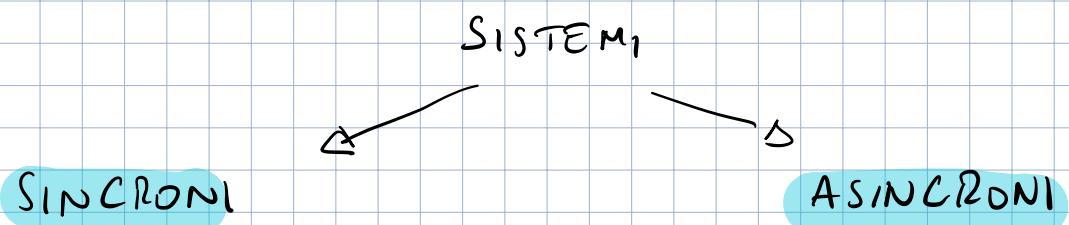
VARIABILI DI STATO.

FSM_s : Finite State Machines (S, I, O, f, g, s_0)

- funzione di output $o = f(s_i)$
- funzione di stato futuro $s_{\text{EXT}} = g(s_i, i)$
- rappresentabile come DIAGRAMMI A STATI

Classificazione:

- Combinatori: $f: I \rightarrow O$
- Sequentiali (Mealy's) $f: S_{\text{ext}} \rightarrow O$ $g: S_{\text{ext}} \rightarrow S$
- Sequentiali (Moore's) $f: S \rightarrow O$ $g: S_{\text{ext}} \rightarrow S$



La presenza o meno di un segnale di sincronismo (nel PC è il clock) che definisce lo stato "Pulsing" e quello presente.

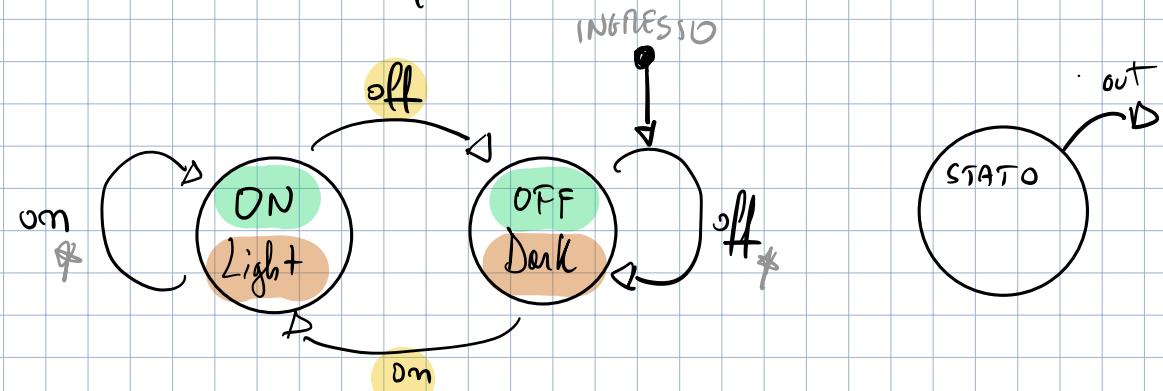
Con un circuito sincrono è possibile avere un circuito che oscilla all'infinito.

DIAGRAMMA DEGLI STATI

STATO DI UN LED

Moore's:

Esempio con un trasformatore con interruttore e un LED.



I CERCHI SONO STATI

* Finché sono off o on, rimanono in quell' stato.

$$\begin{aligned} S &= \text{ON} \\ O &= \text{Light, Dark} \\ I &= \text{on, off} \end{aligned}$$

$$S_0 = \text{OFF}$$

$$f : S \rightarrow O$$

$$g : S_0 \rightarrow S$$

Per poterli considerare in modo Booleano dobbiamo mapparla in tabella

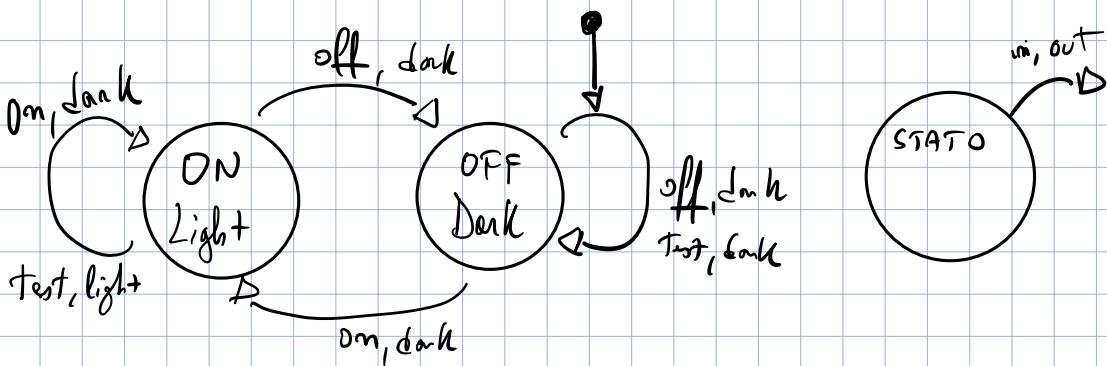
State	Input	Output	S_{NEXT}
OFF	off	Dark	OFF
OFF	on	Dark	ON
ON	off	Light	OFF
ON	on	Light	ON

$$S = \{ \text{ON}, \text{OFF} \}$$

$$O = \{ \text{Light}, \text{Dark} \}$$

$$I = \{ \text{on}, \text{off} \}$$

Mcally:



$$S = \{ \text{ON}, \text{OFF} \}$$

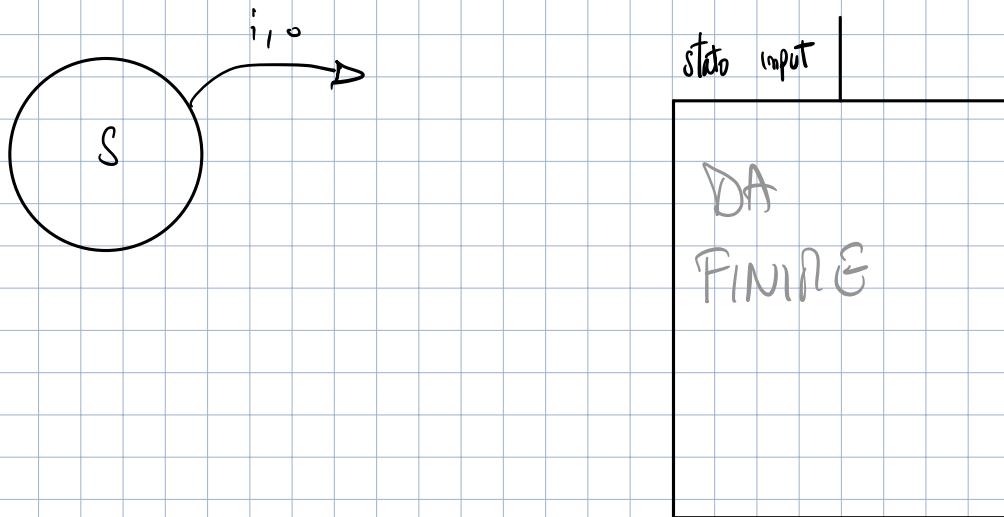
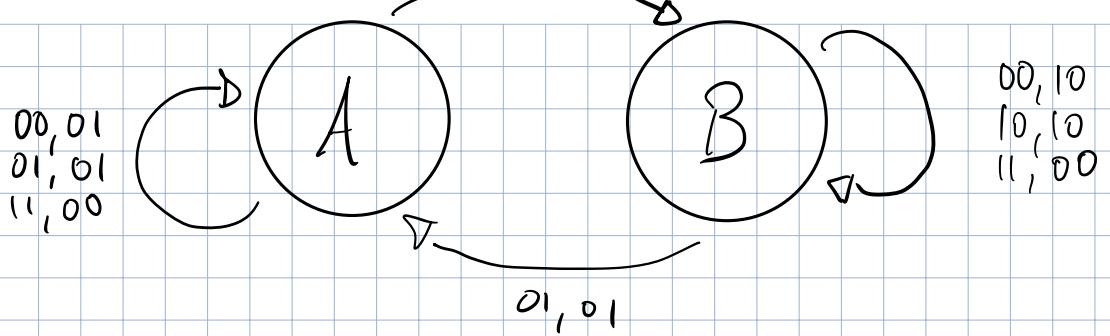
$$S_0 = \text{OFF}$$

$$O = \{ \text{light}, \text{dark} \}$$

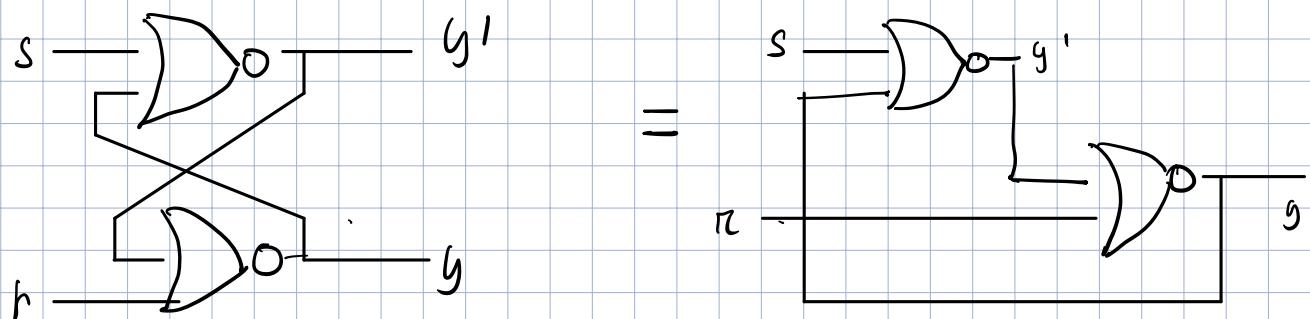
$$I = \{ \text{on}, \text{off}, \text{test} \}$$

In questo circuito abbiamo un comando "test" che deve richiedere un output al led, in caso contrario non riceve output.

State	Input	Output	S_{NEXT}
OFF	off	dark	OFF
OFF	on	dark	ON
OFF	test	dark	OFF
ON	off	dark	OFF
ON	on	dark	ON
ON	test	light	ON



Latch SR (set reset)



y'	S	\bar{n}	f	y_{NEXT}
0	0	0	1	0
0	0	1	0	1
0	1	0	1	1

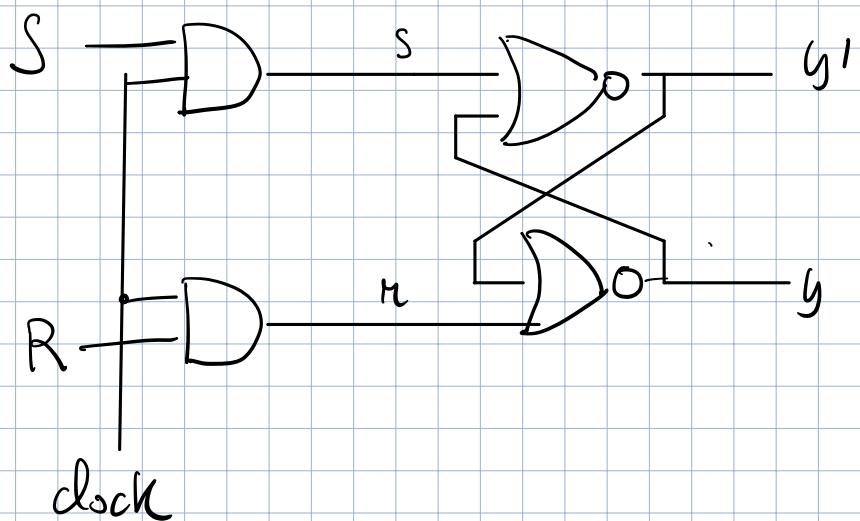
● HOLD
● RESET

	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

SET

la configuration 11 non ami interne.

FLIP FLOP

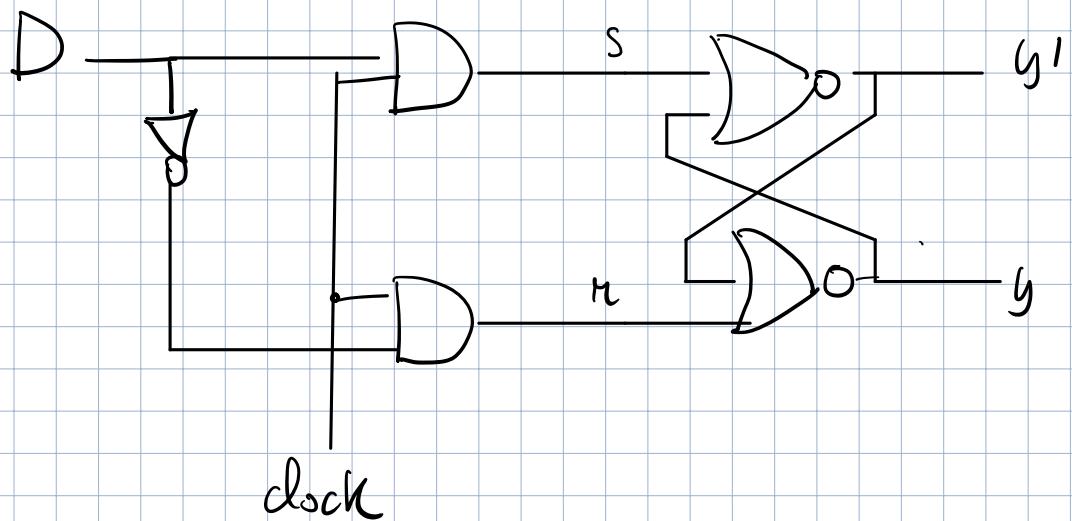


clock	S	R	S	R
0	0	0	0	0
0	0	0	0	0
0	1	1	0	1
1	0	0	0	0
1	1	1	1	0

Annotations on the right side of the table:

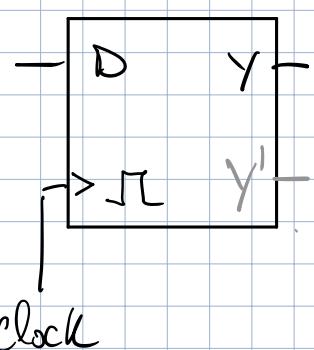
- A brace groups the first four rows under the heading "HOLD".
- A brace groups the last three rows under the headings "HOLD", "RESET", and "SET".

FLIP FLOP D



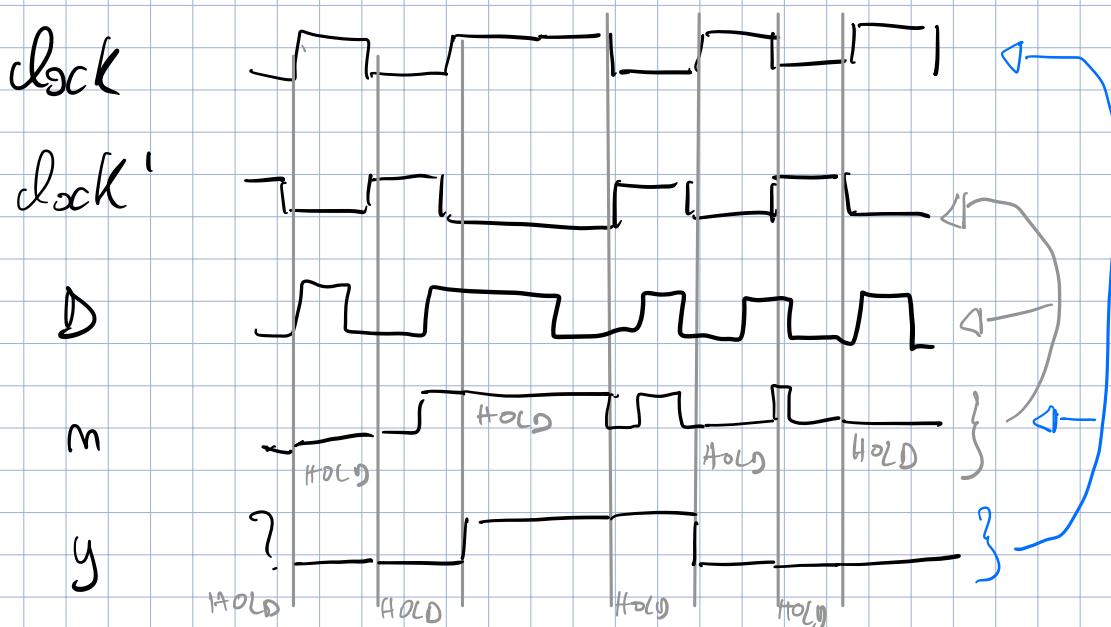
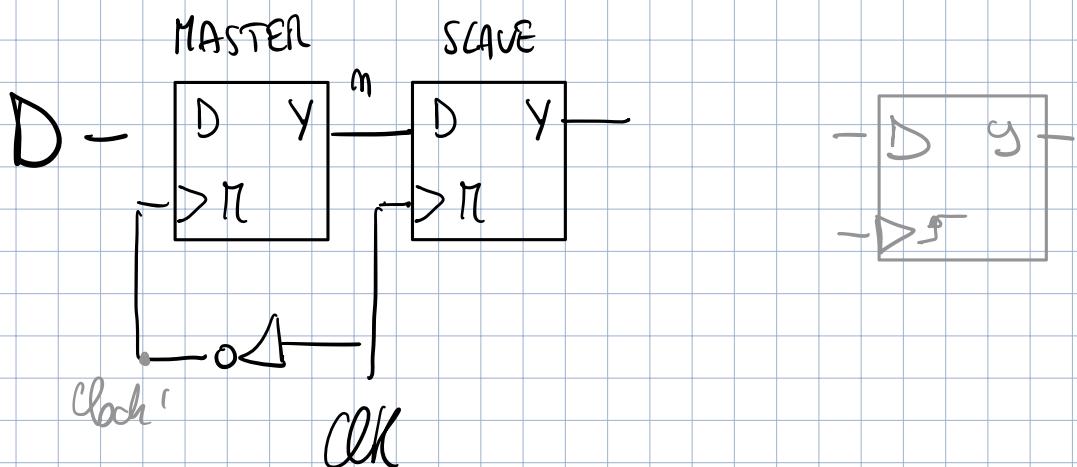
clock	D	S	R	S	R
0	0	0	1	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	1	1	0	1	0

Lo rappresentiamo così:



il simbolo IL indica
che è il simbolo che
adottiamo per l'ingresso.

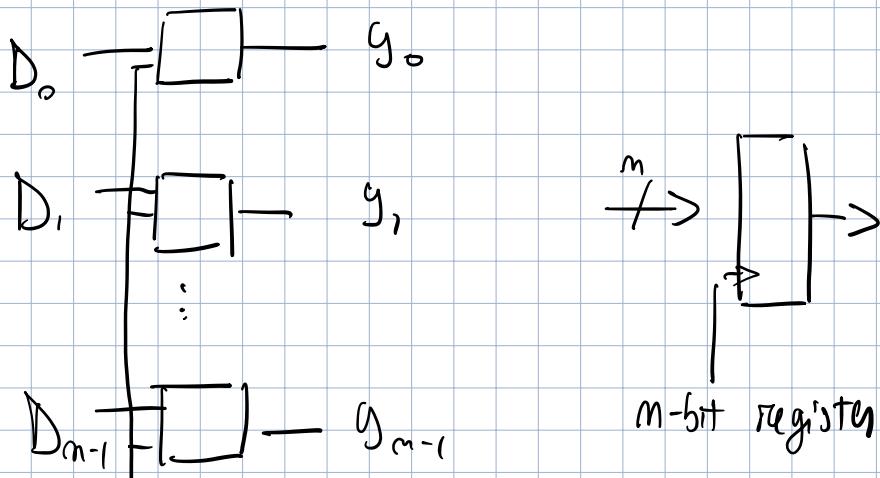
Flip Flop D Edge Triggered



→ Es macht nützlich für Synchronisation; circuit.

REGISTER

Se utilizziamo molti FFDET in parallelo:



Registrano parallelamente serie di bit.

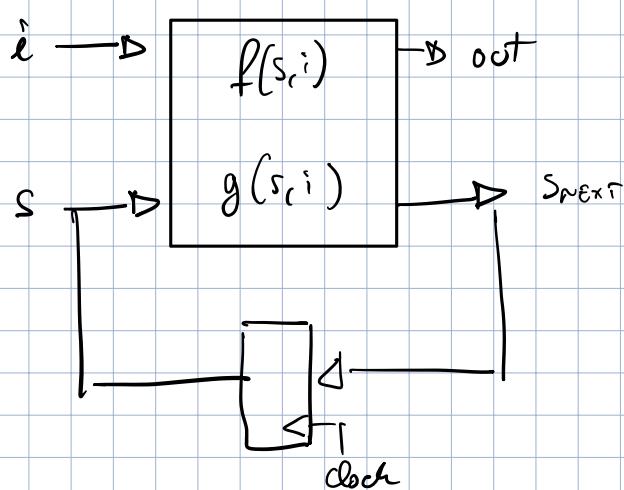
Circuiti regolatori sincroni:

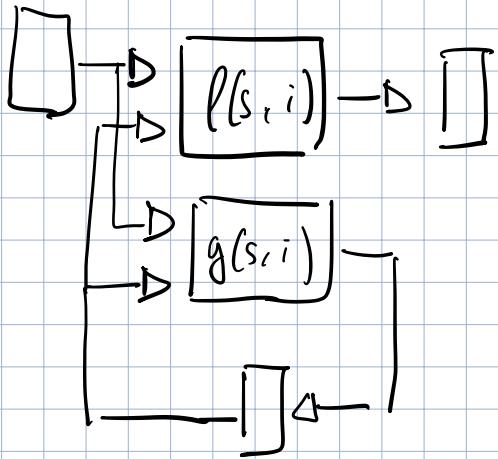
Se utilizziamo circuiti sincroni otteniamo per orologio il clock
esendo uguale per tutto il circuito, d' cui (essendo
meccanicamente periodico) interessa unicamente il periodo.

$$f : \delta \times I \rightarrow O$$

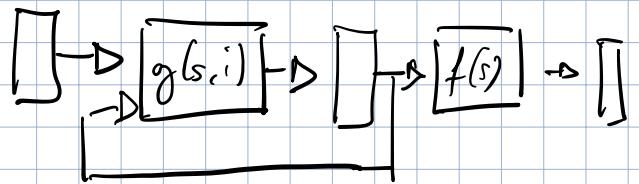
$$g : \delta \times I \rightarrow S$$

$$(s, i) \rightarrow s_{\text{NEXT}}$$





Melny's



Melny's

METRICHE DI DESIGN

- AREA (A)

- ↳ Numero di porte logiche
- ↳ Numero di NAND a 2 input
- ↳ Numero di input alle porte logiche

- PERFORMANCE

↳ Tempo di propagazione: (T_p)

dopo quanto l'uscita si stabilizza

↳ Tempo di contaminazione: (T_c)

dopo quanto c'è effett. c'è un'uscita

↳ Throughput (rate)

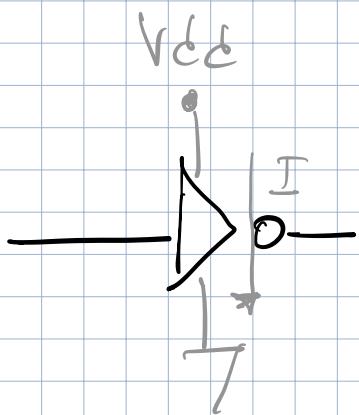
$$\leq \frac{1}{T_p}$$

- POTENZA

L_D Statico (w)

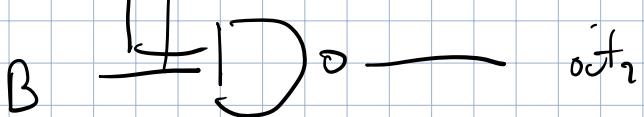
L_D Dinamico (w)

Da cosa è dato il consumo di potenza in digitale?



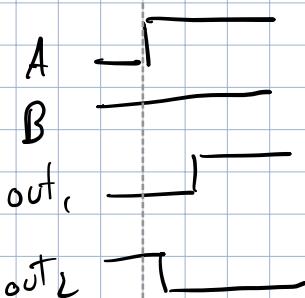
T_P e T_C

Ese.



$$T_P = 2$$

$$T_C = 1$$

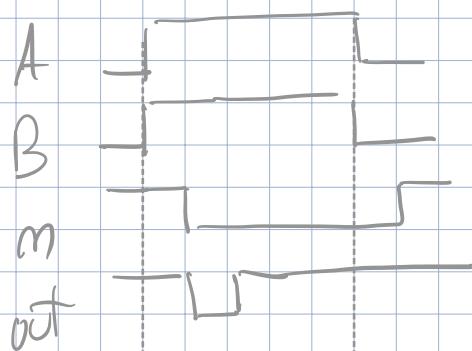
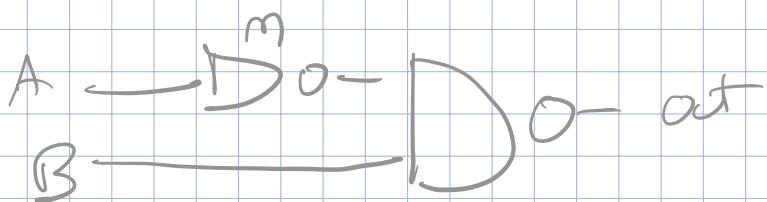


	out ₁	out ₂
A	2	1
B	2	1

In base a quale porta logica ottieni
Tp pin to pin

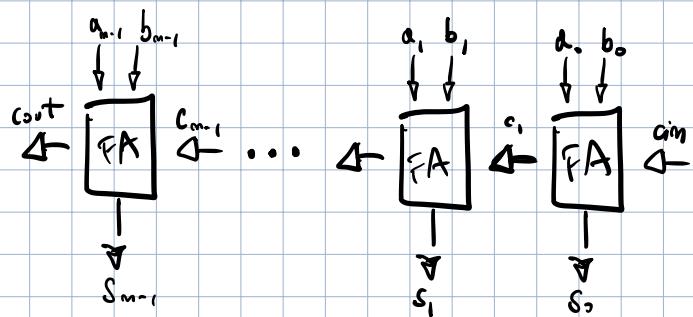
Il T_p del circuito è il tempo più lungo (2)
Il T_c del circuito è il tempo più corto (1)

Ex 2



Esempio 1-D : Ripetuto

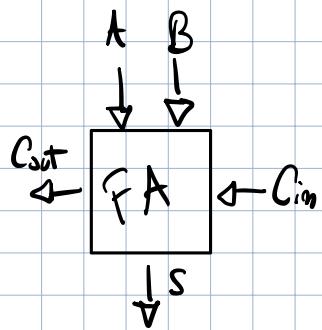
$$S = A + B$$



Se il primo modulo prende solo 2 ingressi (senza C_{in}) detto HALF ADDER.

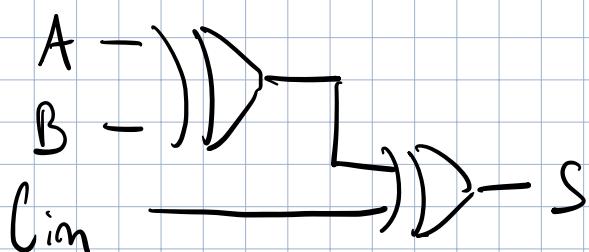
In caso contrario è detto FULL ADDER.

Rappresentazione Full Adder:



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

$$\begin{aligned}
 S &= C_{in}' A' B + C_{in}' A B' + C_{in} A' B' + C_{in} A B = \\
 &= C_{in}' (A' B + A B') + C_{in} (A' B' + A B) = \\
 &= C_{in}' (\underbrace{A \oplus B}_{\text{EXOR}}) + C_{in} (\underbrace{A \oplus B}_{\text{EXNOR}})' = x \cdot y + \bar{x} \bar{y} \\
 &= C_{in} \oplus (A \oplus B)
 \end{aligned}$$

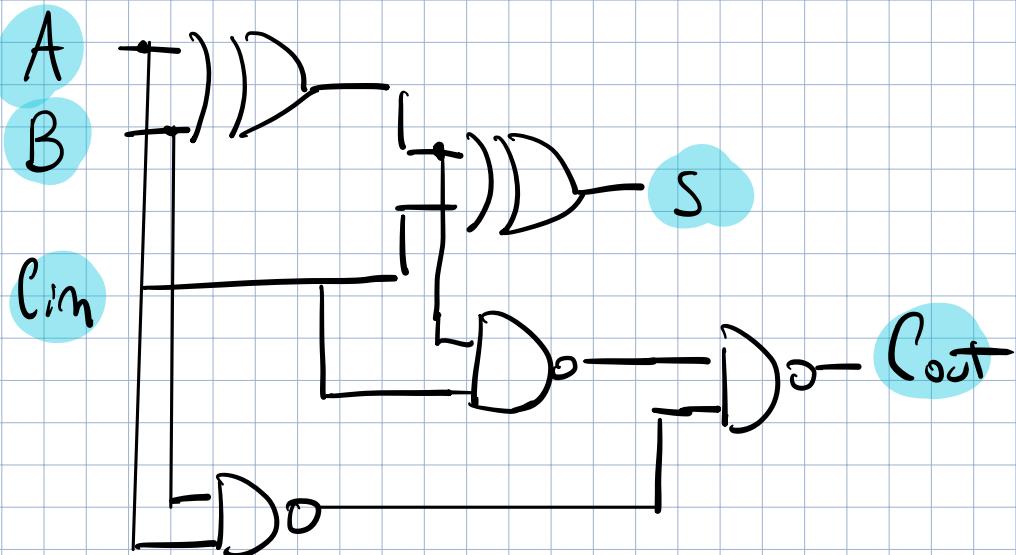


$$\begin{aligned}
 C_{out} &= C_{in} A B + C_{in} A' B + C_{in} A B' + C_{in} A B = \\
 &= C_{in}' A B + C_{in} (A' B + A B + A B) =
 \end{aligned}$$

$$= C_{in}'(AB) + C_{in}(A \oplus B)$$

Altro risultato possibile essere $AB + C_{in}(A \oplus B)$

FULL ADDER con Cout



Calcolo tempi di propagazione :

	S	Cout
A	6	5
B	6	5
Cin	3	2

$$A = 2(\text{EXOR} = \text{INV} + 2 \text{NAND})$$

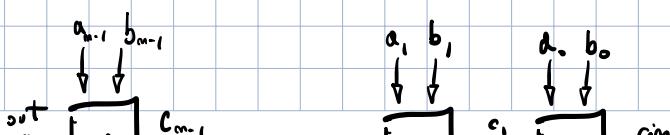
$$\begin{aligned} T_p &= 6 \\ T_c &= 2 \end{aligned}$$

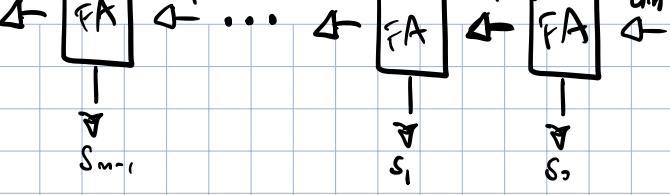
Tra i due 2 e 6 non ha senso guardare la JCT, te.

Ripple Carry Adder (RCA_m)

m = numero bit operando

$O(m)$ = complessità asintotica





$$A_{\text{nc}}(\text{RCA}_m) = m A(\text{FA}) = O(m)$$

$$T_p(\text{RCA}_m) = m T_p(\text{FA}) = O(m)$$

$$T_c(\text{RCA}_m) = T_c(\text{FA}) = O(1)$$

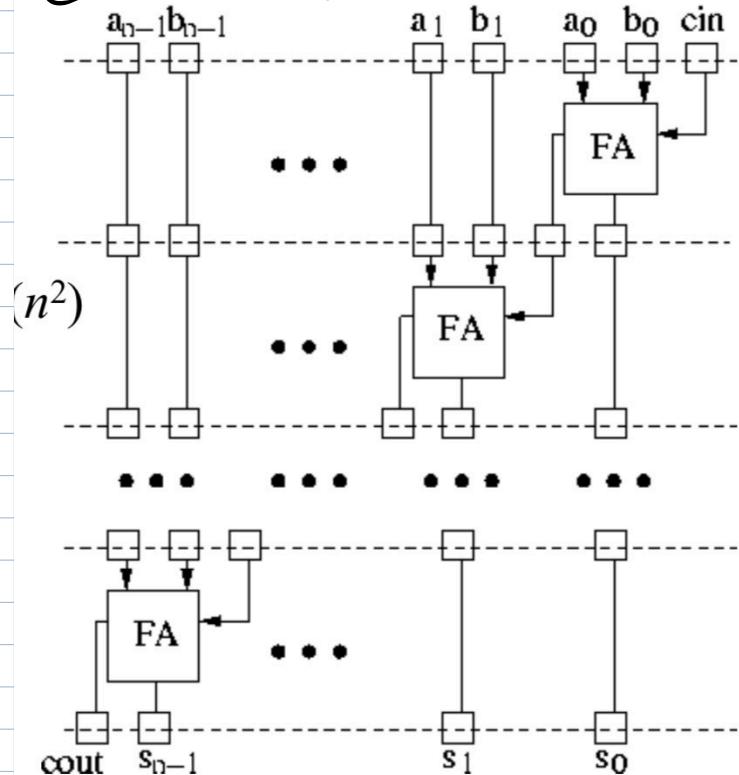
$$\text{Rate}(\text{RCA}_m) \leq 1/T_p(\text{RCA}_m) = O(1/m)$$

$T_c(2)$ abbiamo detto 2 ma effettivamente

non potrei amo' essere pochi il commuto più
carico i da cin a S_0 . (sottostimato)

$T_p(6)$ abbiamo detto 6 ma effettivamente
il percorso più lungo è quello Cin al Cout
quindi 2.

Synchronous RCA_m (SRCAm)



La somma dei
Full Adder viene
realizzata "dai"
Flip Flop.

"CATENA DI
MONTAGGIO"

$$A(\text{SRCAm}) = O(m^2)$$

$$RATE = O(1/n)$$

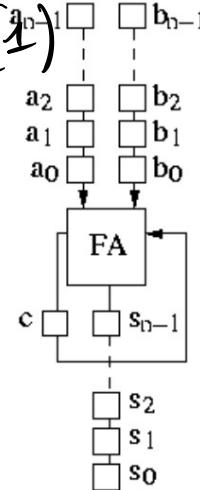
Se gli diamo ad ogni ciclo di clock
addizioni da compilare altra i dati
Pipeline'd. Rate ($PRCA_n$) = $O(1)$

Immaginiamo che 1

Full adder per

ogni row di tutti

Funzionamento così:



$$A(BST_n) = O(1)$$

$$Rate(BST_n) = O(1/n)$$

$$\text{Abbiamo dato prima che } C_{\text{out}} = C_{\text{in}}'AB + C_{\text{in}}(A+B)$$

$$= C_{\text{in}}'AB + \underline{C_{\text{in}}(A+B+A'B)}$$

PER ASSORBIM.

$$= C_{\text{in}}'AB + C_{\text{in}}(A+B) + C_{\text{in}}AB =$$

$$= AB(C_{\text{in}}+C_{\text{in}}') + C_{\text{in}}(A+B) =$$

$$= \underbrace{AB}_{T} + \underbrace{C_{\text{in}}(A+B)}_{P(\text{PROPAGATE})}$$

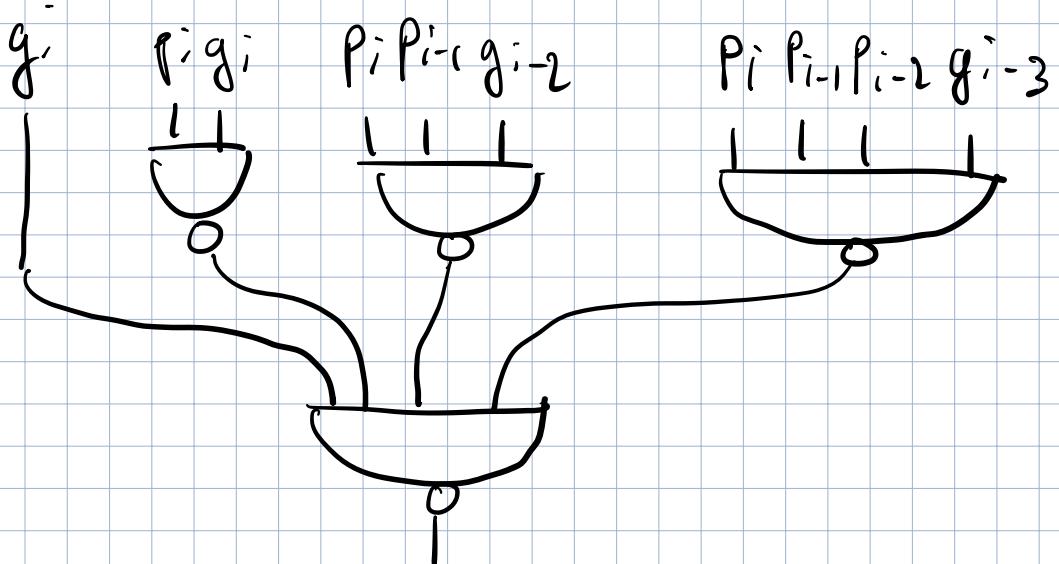
G(GENERATE)

P(PROPAGATE)

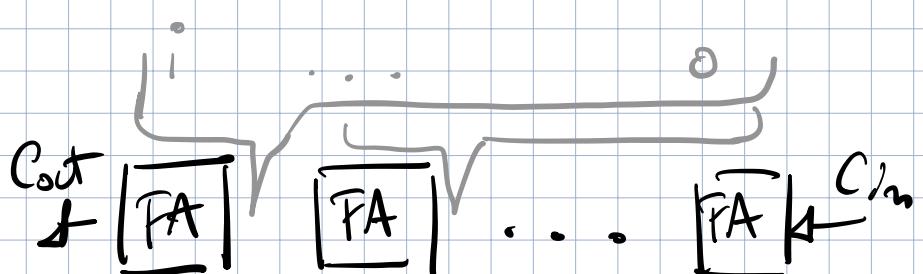
$$C_{\text{out}} = g + C_{\text{in}}(p)$$

$$c_i = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} g_{i-3} + \dots + p_i p_{i-1} p_{i-2} \dots p_0 C_{in}$$

Rete logica che implementa l'espressione:



Per calcolare in automatico il proprio C_{in}

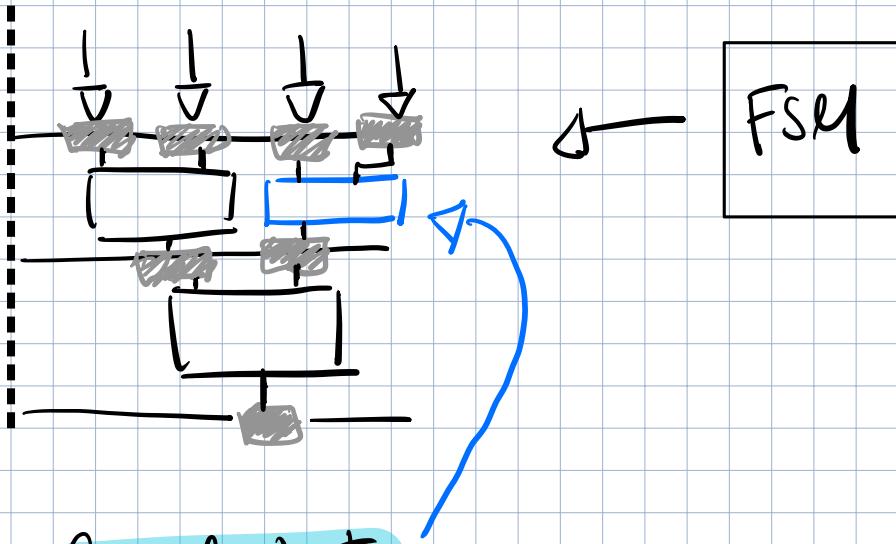


In realtà non comincia sempre con un RCA perché tutti i transistor in serie nei NAND rallenterebbe troppo il circuito.

Quindi nelle reti si utilizzano transistor più grandi ma l'onda potrebbe da $O(n^3)$ a un numero esageratamente grande.

REGISTER TRANSFER LEVEL (RTL)

- Date path:

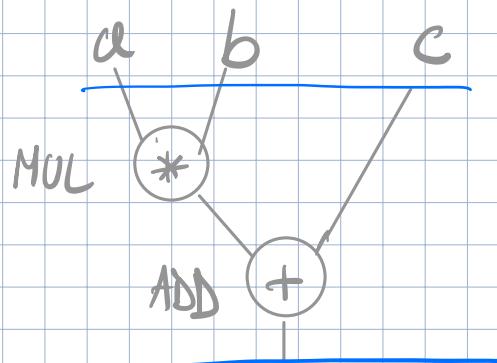


- Control Unit:

Utilizzata per dare il segnale di controllo ai Multiplexer (MUX).

Ese.

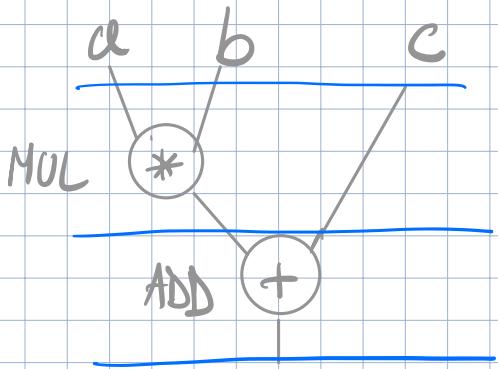
$$F = a * b + c$$



$$T_{clk} > T_p = T_p(\text{MUL}) + T_p(\text{ADD})$$

$$T_c = T_c(\text{ADD})$$

Se il tempo di clock èusto:

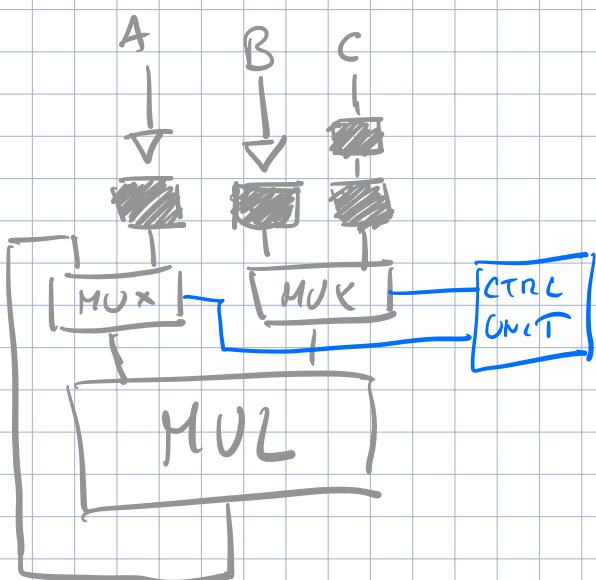
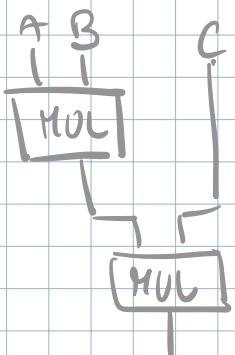


$$T_{clk} > \text{Max}(T_p(\text{MUL}), T_p(\text{ADD}))$$

STILL DI PROGETTO :

- Progetto a singolo stadio
 - Computation in un solo ciclo di clock
- Progetto ad n-stadii
- Progetto con pipelining
- Progetto in memoria sharing :

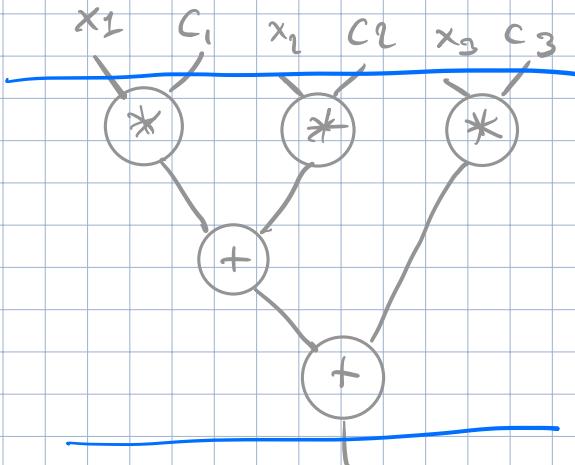
Ex.



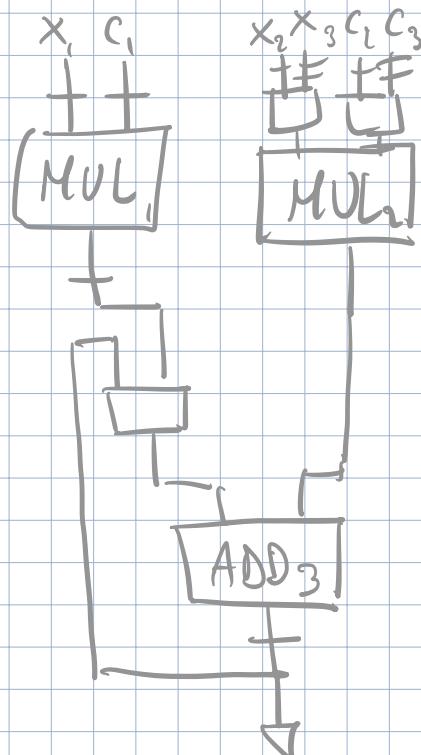
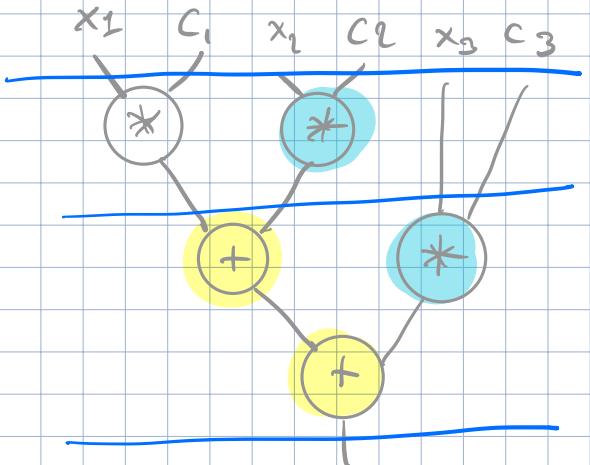
Ex.

$$F = x_1 * c_1 + x_2 * c_2 + x_3 * c_3$$

Dsteflow graph :



Dsteflow graph :

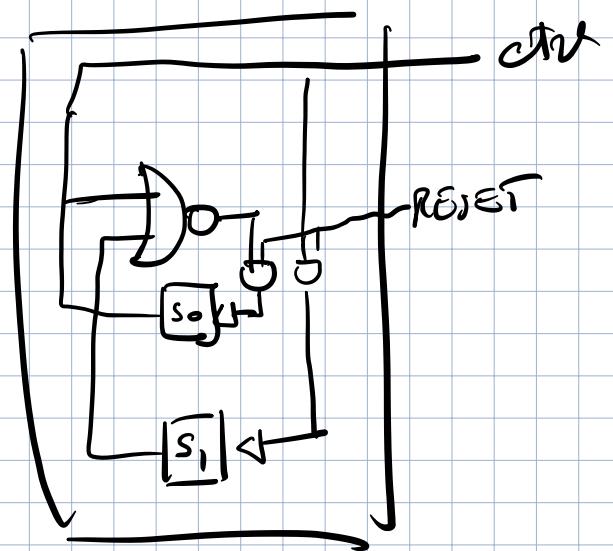
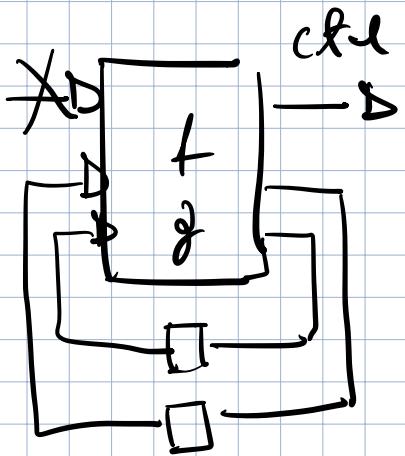


$$CU = \begin{array}{c|cc|ccc} \text{state}_0 & g & & a & b & c \\ \hline 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 \end{array}$$

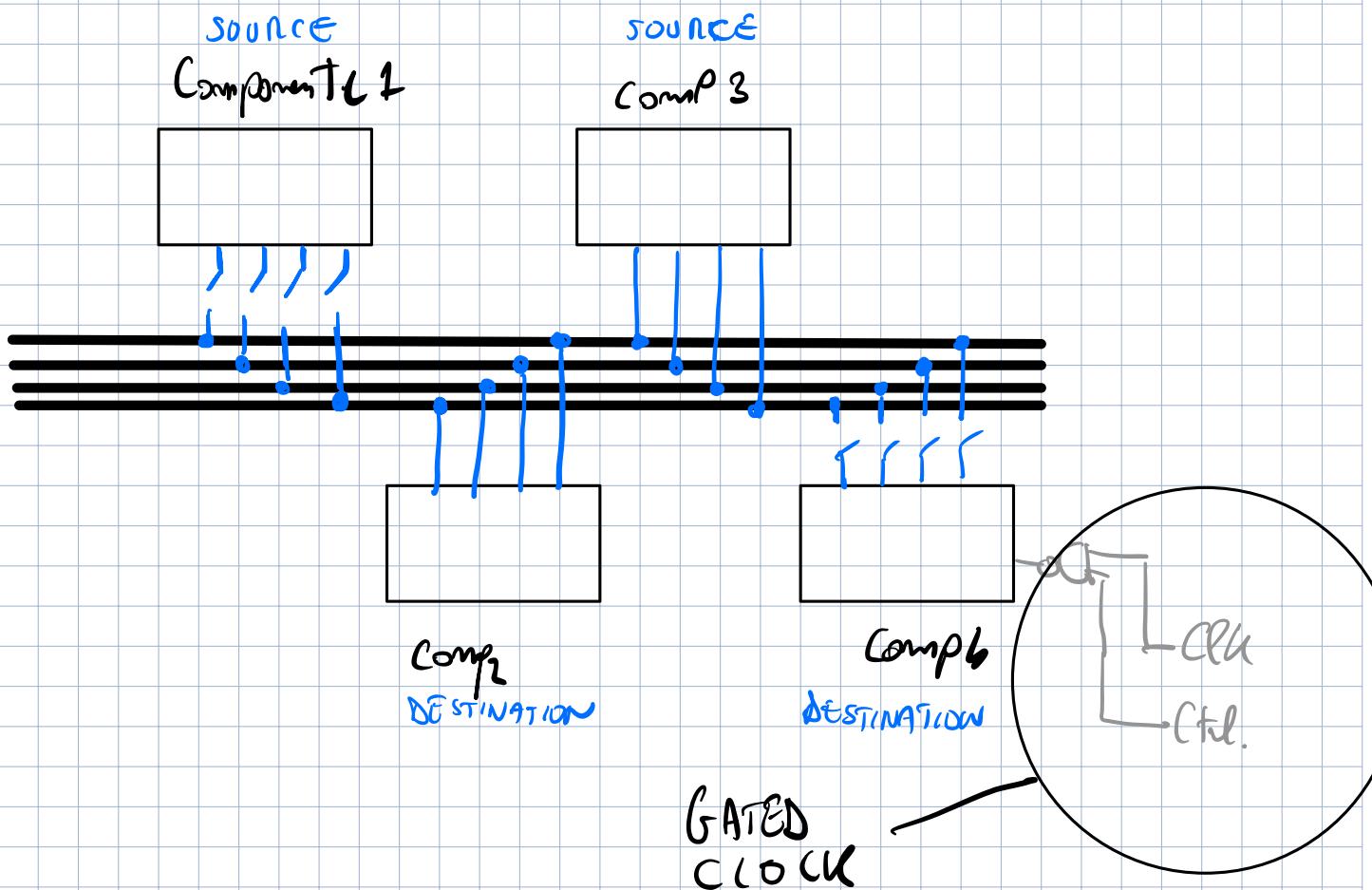
$$\begin{array}{l} \text{next} \quad \text{ctrl} \\ \hline s_1 \quad s_0 \quad s_1 \quad s_0 \quad \text{ctrl} \\ 00 \quad 01 \quad 0 \\ 01 \quad 10 \quad 1 \\ 10 \quad 00 \quad 0 \\ \hline 11 \quad 10 \quad 1 \end{array}$$

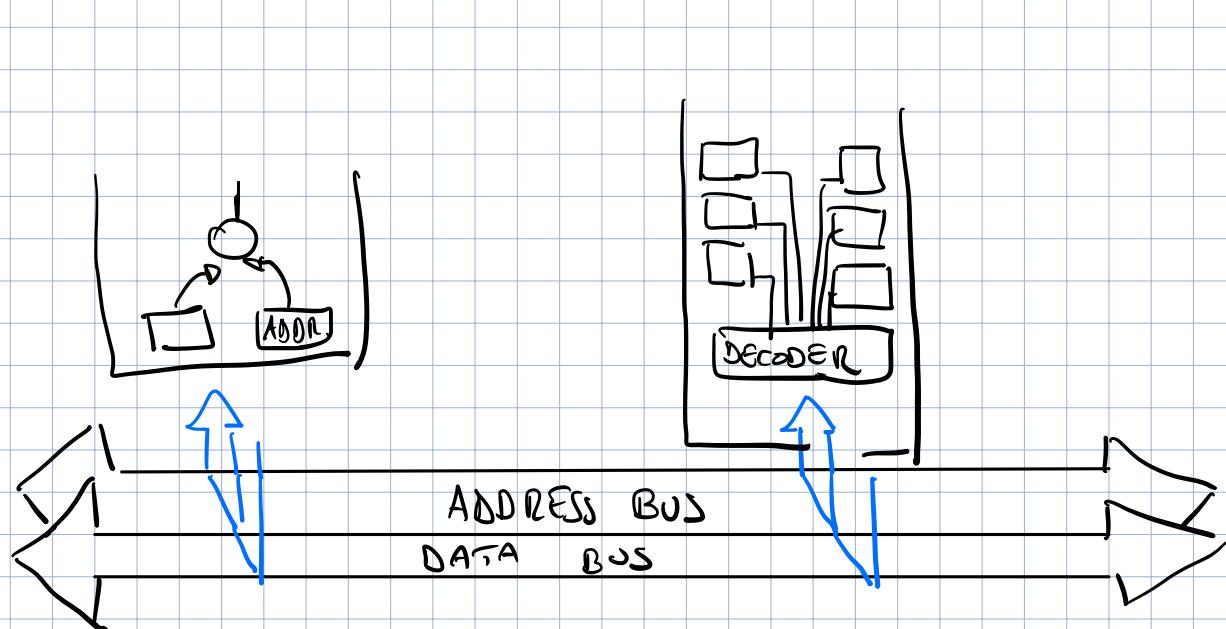
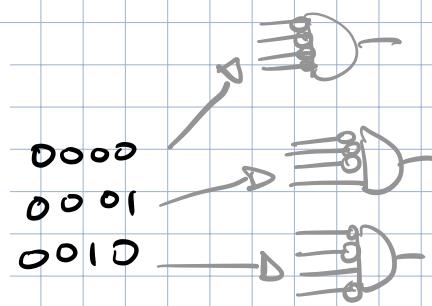
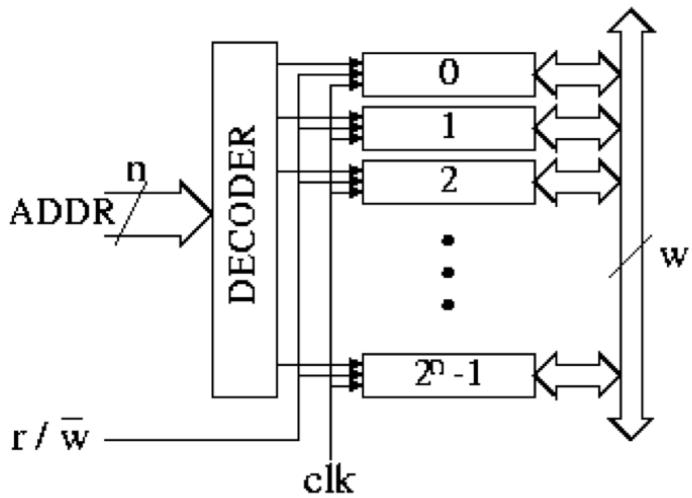
$$\begin{aligned} S_1^{\text{next}} &= S_0 \\ S_0^{\text{next}} &= (S_0 - S_1)^T \end{aligned}$$

$$\text{ctrl} = S_0$$



DATA TRANSFER TOA REGISTER (BUS)





T