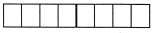
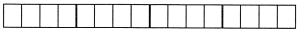
Riassunto tecnica digitale

Riassunto tecnica digitale

Introduzione

Formati Binari

Bit		(Binary digit): 0 o 1
Nibble		(4 bit): numero da 0 a15
Byte		(8 bit): numero da 0 a 255
Word		(16 bit): numero da 0 a 65535

1kbytes (kB)	=	2^{10} byte	=	**1'024** bytes
1Mbytes (MB)	=	2^{20} byte	=	1'048'576 bytes
1Gbytes (GB)	=	2^{30} byte	=	1'073'741'824 bytes
1Tbytes (TB)	=	2^{40} byte	=	2 e^{112} bytes

Riassunto tecnica digitale

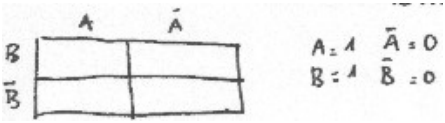
La tecnica digitale usa numeri 1 o 0 e quindi il segnale diventa perfetto.

Livello logico:
L = livello basso
H = livello alto

Stato logico:
Logica positiva = 0 = L
1 = H
Logica negativa = 0 = H
1 = L

La tabella della verità
In questa tabella sono presenti tutte le combinazioni del segnale in entrata con il corrispondente in uscita.

Diagramma di Karnaught
Le entrate sono considerate come insiemi.
Es:



La formula

Il segno $\wedge$ può essere tralasciato: $A \wedge B = AB$

operazione	simbolo	esempio	equivalenza insiemi
NOT	$\bar{}$	$\bar{A}$	complemento
OR	$\vee$	$A \vee B$	unione
AND	$\wedge$	$A \wedge B$	intersezione
XOR	$\forall$	$A \forall B$	somma disgiuntiva

Riassunto tecnica digitale

Operazioni OR
Somma

OPERAZIONE OR (somma)

Tabella dell'operazione OR:

$$0 \vee 0 = 0$$
$$0 \vee 1 = 1$$
$$1 \vee 0 = 1$$
$$1 \vee 1 = 1$$

Indicando con A una variabile che può assumere i valori 1 e 0:

$$A \vee 0 = A$$
$$A \vee 1 = 1$$
$$A \vee A = A$$
$$A \vee \overline{A} = 1$$

PROPRIETÀ

L'operazione OR ha le seguenti caratteristiche:

è interna e sempre definita	
è associativa	$(A \vee B) \vee C = A \vee (B \vee C)$
è commutativa	$A \vee B = B \vee A$
0 è l'elemento neutro	$A \vee 0 = A$
1 è assorbente	$A \vee 1 = 1$

L'operatore OR non ha la struttura di un gruppo commutativo e quindi non si possono risolvere le equazioni del tipo: $A \vee X = B$

AND
Moltiplicazione

Tabella dell'operazione AND:

$$0 \wedge 0 = 0$$
$$0 \wedge 1 = 0$$
$$1 \wedge 0 = 0$$
$$1 \wedge 1 = 1$$

Indicando con A una variabile che può assumere i valori 1 e 0:

$$A \wedge 0 = 0$$
$$A \wedge 1 = A$$
$$A \wedge A = A$$
$$A \wedge \overline{A} = 0$$

PROPRIETÀ

L'operazione AND ha le seguenti caratteristiche:

è interna e sempre definita	
è associativa	$(A \wedge B) \wedge C = A \wedge (B \wedge C)$
è commutativa	$A \wedge B = B \wedge A$
1 è l'elemento neutro	$A \wedge 1 = A$
0 è assorbente	$A \wedge 0 = 0$

L'operatore AND non ha la struttura di un gruppo commutativo e quindi non si possono risolvere le equazioni del tipo: $A \wedge X = B$

Riassunto tecnica digitale

XOR
Somma disgiuntiva

OPERAZIONE XOR (Somma disgiuntiva)

Tabella dell'operazione XOR:

$$0 \veebar 0 = 0$$
$$0 \veebar 1 = 1$$
$$1 \veebar 0 = 1$$
$$1 \veebar 1 = 0$$

Indicando con A una variabile che può assumere i valori 1 e 0:

$$A \veebar 0 = A$$
$$A \veebar 1 = \bar{A}$$
$$A \veebar A = 0$$
$$A \veebar \bar{A} = 1$$

PROPRIETÀ

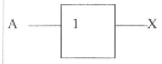
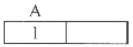
L'operazione XOR ha le seguenti caratteristiche:

è interna e sempre definita	
è associativa	$(A \veebar B) \veebar C = A \veebar (B \veebar C)$
è commutativa	$A \veebar B = B \veebar A$
0 è l'elemento neutro	$A \veebar 0 = A$
ogni elemento è il suo simmetrico	$A \veebar A = 0$

L'operatore XOR ha la struttura di un gruppo commutativo e quindi si possono risolvere le equazioni del tipo: $A \veebar X = B \rightarrow X = A \veebar B$

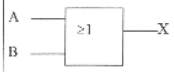
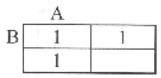
Operatori logici

Gli operatori logici sono componenti che svolgono una funzione logica. Il simbolo internazionale è un rettangolo con le entrate collegate su un lato (normalmente a sinistra) e le uscite sul lato opposto. All'interno vi è il simbolo distintivo dell'operatore. L'operatore qui sotto viene detto buffer perché in uscita ritorna il valore d'entrata ma più pulito.

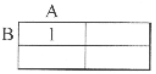
Simbolo	Tabella	Diagramma
 X = A	A | X 0 | 0 1 | 1	

Riassunto tecnica digitale

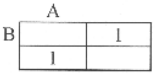
Porta OR

Simbolo	Tabella	Diagramma
 $X = A \vee B$	A B X 0 0 0 0 1 1 1 0 1 1 1 1	

Porta AND

Simbolo	Tabella	Diagramma
 $X = A \wedge B$	A B X 0 0 0 0 1 0 1 0 0 1 1 1	

Porta XOR

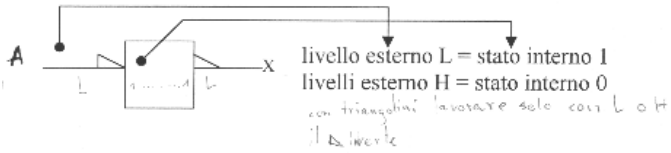
Simbolo	Tabella	Diagramma
 $X = A \forall B$	A B X 0 0 0 0 1 1 1 0 1 1 1 0	

Negazione

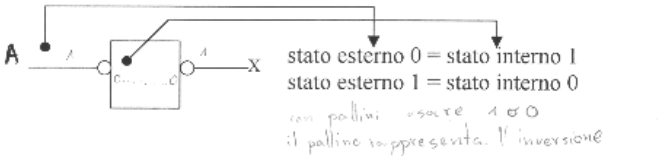
Il simbolo di negazione

I segnali “interni” all’operatore vanno sempre definiti con lo stato logico (0 o 1). I segnali “esterni” all’operatore possono essere definiti con lo stato logico (0 o 1) o con il livello logico (L, H). Non è possibile utilizzare i due metodi nell’ambito dello stesso schema. Il simbolo di negazione è un *triangolo* quando si usa il *livello logico* e un *cerchio* quando si usa lo *stato logico*.

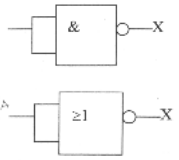
Riassunto tecnica digitale



ESEMPIO CON STATO LOGICO



Operatori fondamentali
Con un collegamento una NAND diventa una NOT.
Con una NOR.



A	X
0	1
1	0

A	X
0	1
1	0

Ottimizzare mediante il teorema di De Morgan

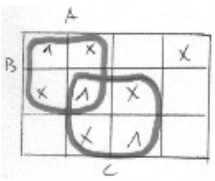
Sostituire l'operatore OR con una AND.

$\overline{A \vee B} = \overline{A} \wedge \overline{B}$ $\qquad$ $\overline{A \vee B \vee C} = \overline{A} \wedge \overline{B} \wedge \overline{C}$

Sostituire l'operatore AND con una OR.

$\overline{A \wedge B} = \overline{A} \vee \overline{B}$ $\qquad$ $\overline{A \wedge B \wedge C} = \overline{A} \vee \overline{B} \vee \overline{C}$

Condizioni di indifferenza
Simbolo = x o Ø.
Queste due condizioni
sfruttate per semplificare i circuiti.



Riassunto tecnica digitale

VHDL

```
library ieee;
use ieee.std_logic_1164.a11;
use ieee.numeric std.all;    → (+,-
,*,/,>,<) operazioni aritmetiche

entity ese is
port(
        clk, rst, entrata : in std_logic;
        uscita : out std_logic;
        X: unsigned       → numeri positivi;
        XX: singed        → anche numeri negativi);
end ese;

architecture arc of ese is
signal dati: std_logic_vector(7 downto 0);
        → signal - <=
        → variable - :=
begin
        process (clk, rst, dati, entrata)
        begin
                if rst = '1' then
                  dati <= "00000000";
                elsif clk'event and clk='0' then
                  dati(6 downto 0) <= dati(7 downto 1);
                  dati (7) <= entrata;
                  uscita <= (others => '0'); → mette tutto a 0
        end if;
        end process;
        uscita <= dati(0);
end arc;
```

Riassunto tecnica digitale

When_else:
fuori dal process!!

```
q <= a when sel = “00” else
     b when sel = “01” else
     c when sel = “10” else
     z; --altre condizioni
```

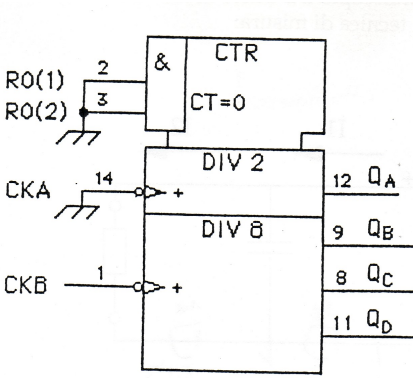
With_select:

```
with sel select
q <= a when “1010”
     b when “1111”
     '0' when others; --altre condizioni
```

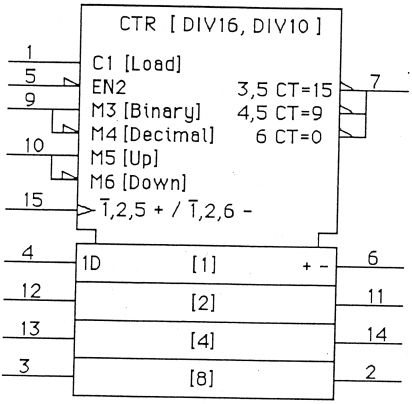
Riassunto tecnica digitale

Contatori

Questo contatore può contare fino a 10. Una particolarità sta nel sistema di reset. Questo comprende due modi di resettare il contatore, con la prima le uscite vanno tutte a 0 mentre con la seconda le uscite assumono 1001.



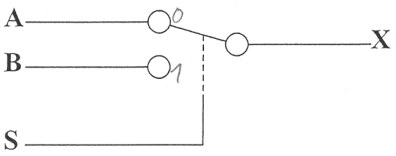
Un'altro tipo di contatore con varie possibilità di scelta delle opzioni.



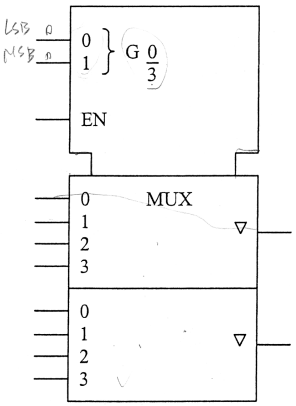
Riassunto tecnica digitale

Multiplexer

Il multiplexer é come un grande switch, dove tramite delle entrate di comando si può selezionare che collegamento effettuare.
Rappresentazione grafica:



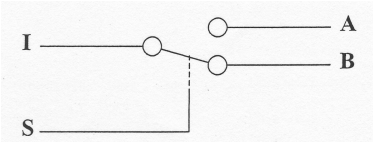
esempio: multiplexer a 4 entrate di 2 bit con uscita tri-state



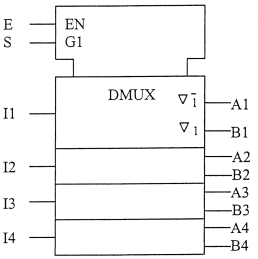
Riassunto tecnica digitale

Demultiplexer

É il contrario del multiplexer. Praticamente con dei segnali di comando si seleziona l'uscita giusta del segnale.



Simbolo



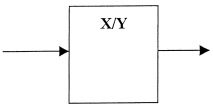
S	E	Uscita selezionata
0	0	A
1	0	B
X	1	uscita isolata (Z)

Riassunto tecnica digitale

Convertitori

Il convertitore è una logica, generalmente combinatoria, che trasforma una quantità data, da una radice a un'altra (da una forma ad un'altra).

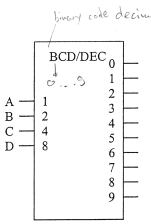
Simbolo generale di un convertitore



dove X/Y = da X a Y, esempio: binario–decimale

Convertitore binario–decimale

Sulle entrate si mette il numero binario; sulle uscite appare un solo segnale corrispondente al valore del numero binario d'entrata. Vediamo un esempio:



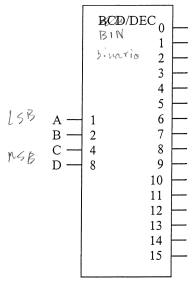
D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Esempio di integrati: 74…42, 74…45

Riassunto tecnica digitale

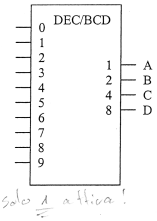
Convertitore binario–1 su 16

È come il precedente, ma senza limiti al valore d'entrata, quindi abbiamo 16 uscite.



Esempio di integrati: 74…154, 74…159

Convertitore decimale–binario

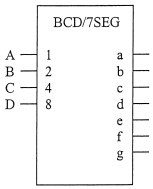


0	1	2	3	4	5	6	7	8	9	D	C	B	A
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

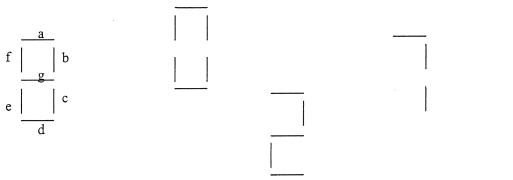
Esempi di integrati: 74…147, 74…148

Convertitore binario–7 segmenti

Serve a convertire un numero espresso in codice binario in un codice a 7 bit capace di comandare un display.



D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	dipende dal tipo di convertitore						
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							



Esempi di integrati: 74…47, F4511

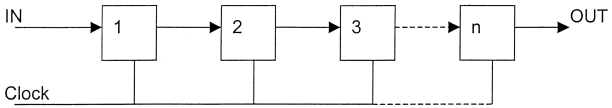
Riassunto tecnica digitale

Shift-register

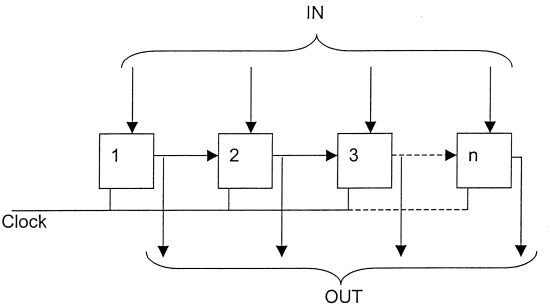
I registri a scorrimento sono formati da una serie di celle di memoria elementari (flip-flop).
Ad ogni clock l'informazione contenuta nei flip-flop viene spostata avandi di una posizione.

Esempi delle 4 varianti possibili:

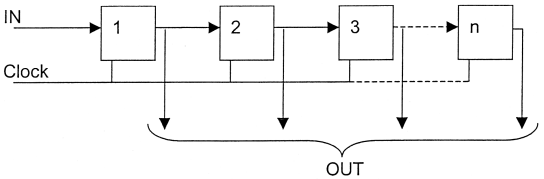
a) Entrata e uscita serie



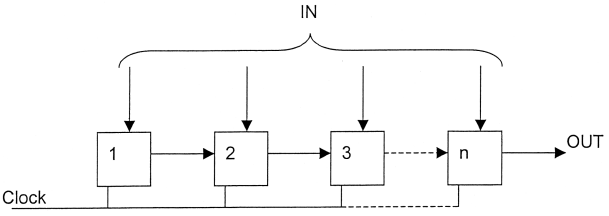
b) Entrata e uscita parallela



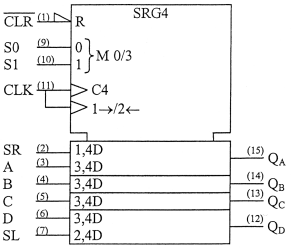
c) Entrata serie e uscita parallela



d) Entrata parallela e uscita serie



Descrizione simbolo:



S0-S1	Selezione del modo di funzionamento: 0 = nessuna operazione. 1 = scorrimento a destra (entrata in SR). 2 = scorrimento a sinistra (entrata in SL). 3 = caricamento in palallelo (entrata A, B, C, D).
A, B, C, D	Entrate parallele.
Q_A, Q_B, Q_C, Q_D	Uscite parallele.
SR	Entrata serie per scorrimento a destra (verso il basso).
SL	Entrata serie per scorrimento a sinistra (verso l'alto).
CLK	Entrata di clock, sensibile sul fianco di salita.
CLR	Reset generale asincrono (non dipende da CLK).

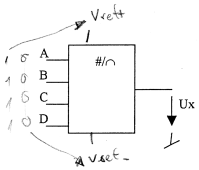
Riassunto tecnica digitale

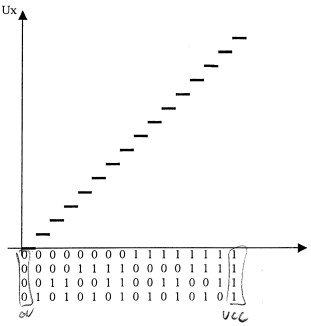
Convertitore digitale analogico DA

Questo elemento é in grado di convertire un segnale digitale in una tensione o corrente.

Siboli: # digitale **∩ analogico**

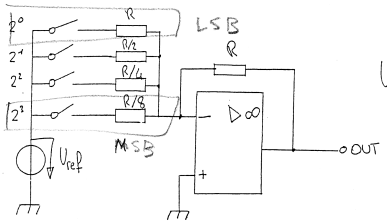
Schema di principio:



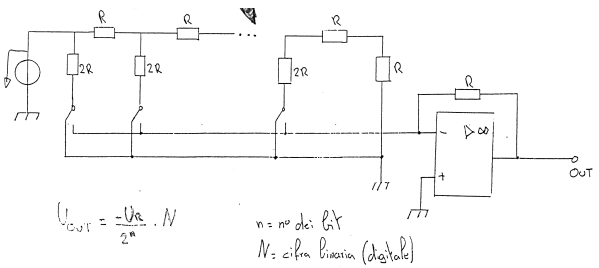


Altri due esempi di convertitori:

Con operazionale:



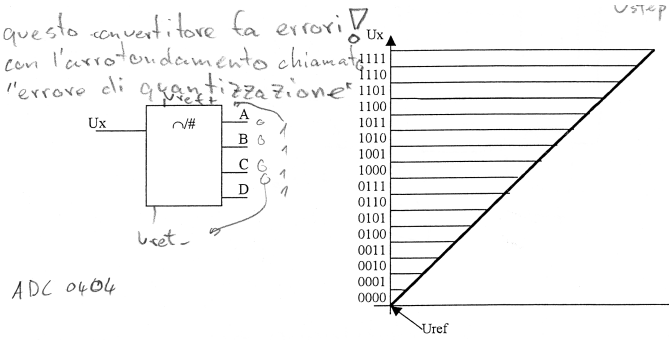
Rete R2R:



Riassunto tecnica digitale

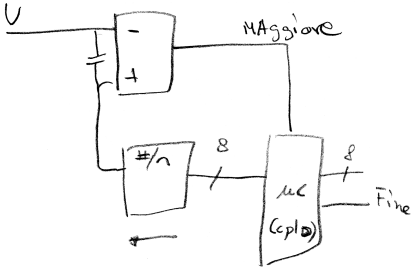
Convertitore analogico digitale AD

Questo componente converte un segnale analogico in un segnale digitale. Per fare cio deve quantizzare il segnale. In un AD c'é un segnale di riferimento per determinare la tensioie Uref- dove corrisponde lo 0 o poi Uref+ per il massimo valore digitale.



Il più efficente del mondo é il convertiore AD con comparatori e una logica, con questo si ottiene il valore in 3ns.

Il più utilizzato é il convertitore ad **approssimaizone sucessiva**:



Viene utilizzato il 90% dei casi.
Funzionamento: dimezza sempre il valore per avvicinarsi al risultato finale.

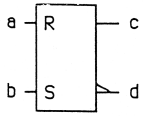
Vantaggi: costo, tempo costruzione
Svantaggi: lento nella conversione

Riassunto tecnica digitale

flip-flop

Multivibratori bistabili:

Flip-flop RS

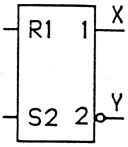


a	b	R	S	Q_{n+1}	c	d
L	L	0	0	Q_n	non cambia	
L	H	0	1	1	H	L
H	L	1	0	0	L	H

stato interno

L'entrata R = 1 mette a 0 l'uscita
L'entrata S = 1 mette a 1 l'uscita

Stato delle uscite con R = S = 1



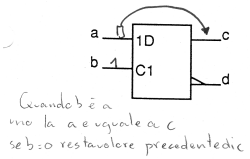
R	S	X_{n+1}	Y_{n+1}
0	0	X_n	Y_n
0	1	1	0
1	0	0	1
1	1	0	0

inversare correttamente R o S per ottenere R = S = 0
per sapere cosa dipende l'uscita basta guardere R = S = 1 e vedere che cosa mette a 0 o a 1 l'uscita.

Riassunto tecnica digitale

Flip-flop D comando statico

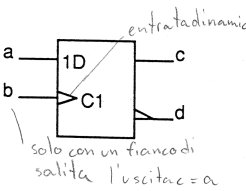
Quando C ha lo stato interno é 1 lo stato interno dell'uscita D e copiato in uscita.



a	b	C	D	Q	c	d
x	L	0	x	Qn	non cambia	
L	H	1	**0**	**0**	**L**	H
H	H	1	**1**	**1**	**H**	L
		stato interno				

Flip-flop D comando dinamico

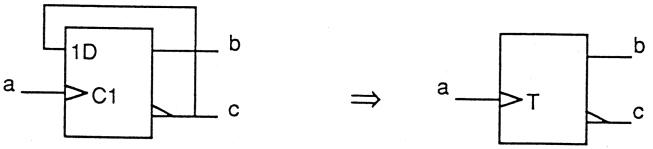
All'istante della transazione del segnale di comando D viene copiato in uscita.



a	b	C	D	Q	c	d
x	L	0	x	Qn	non cambia	
x	H	0	x	Qn	non cambia	
L	↑	1	**0**	**0**	**L**	H
H	↑	1	**1**	**1**	**H**	L
		stato interno				

Flip-flop T

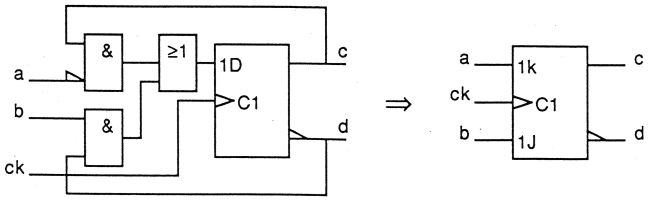
Ad ogni fianco di salita viene inversato il valore in uscita.



Riassunto tecnica digitale

Flip-flop JK

Circuito equivalente:

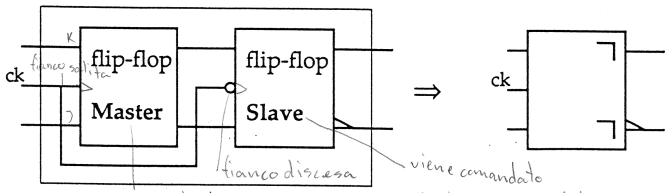


ck	a	b	C	J	K	Q	c	d
L	x	x	0	0	0	Qn	non cambia	
H	x	x	0	0	0	Qn	non cambia	
↑	L	L	1	0	0	Qn	non cambia	
↑	L	H	1	0	1	0	L	H
↑	H	L	1	1	0	1	H	L
↑	H	H	1	1	1	$\overline{Qn}$	commutano	

stato interno

Flip-flop Master Slave

Le uscite con effetto differito sono rappresentate con il sibolino┐
Funziona come un JK ma al fianco di salita "memorizza" i dati e solo al fianco di discesa li copia in uscita.



Riassunto tecnica digitale

Macchina a stati

Nella pratica esistono due tipi di macchine a stati.

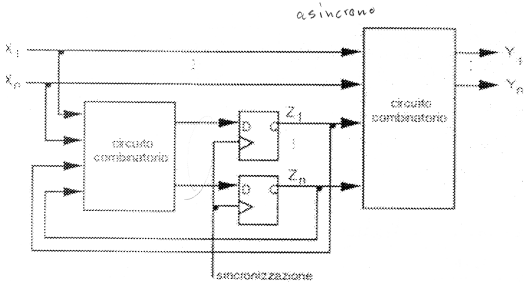
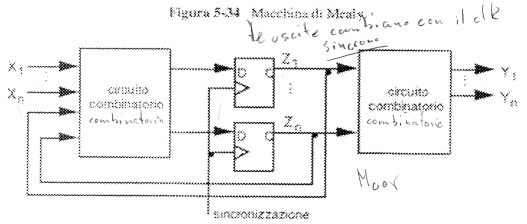
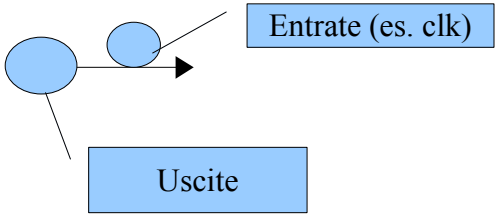


Figura 5-34 Macchina di Mealy

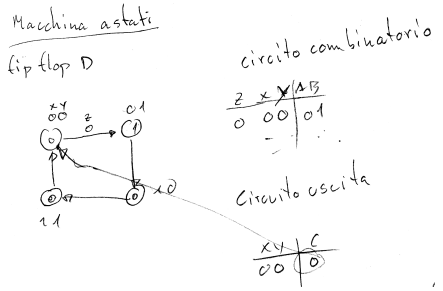


La prima la macchina di Mearly dove le uscite dipendono direttamente dalle entrate cosi che le uscite non sono sincrone con il clock.
Nella seconda é raffigurata la macchina di Moore dove le uscte sono sincrone con il clock.

Per calcolare quanti flip-flop: $2^x = numero\ bolle$



Riassumendo:

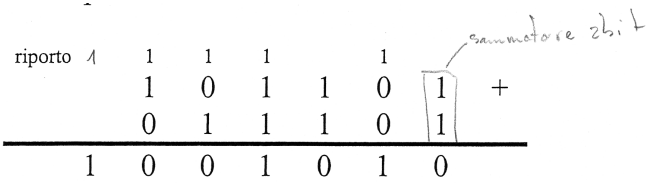



```
library ieee;
USE ieee.std_logic_1164.all;

ENTITY fsm IS
  PORT(
    clk,rst :in std_logic;
    a       :in std_logic;
    x       :out std_logic);
END fsm;

ARCHITECTURE arc_fsm OF fsm IS
TYPE state IS (S0, S1, S2, S3, S4);
SIGNAL stato_corrente, stato_prossimo: state;
BEGIN
  --Primo processo, per l'assegnazione del prossimo stato flip-flop
  PROCESS (clk, rst)
  BEGIN
      IF rst = '1' THEN
      stato_corrente <= S0;
      ELSIF clk'event and clk = '1' THEN
      stato_corrente <= stato_prossimo;
      END IF;
  END PROCESS;

  --Secondo processo, per la definizione del prossimo stato (combinatorio)
  PROCESS (stato_corrente,a)
  BEGIN
    CASE stato_corrente IS
      WHEN S0 =>  IF a = '0' THEN
                    stato_prossimo <= S0;
                    ELSE
                  stato_prossimo <= S1;
                    END IF;
      WHEN S1 =>  stato_prossimo <= S2;
      WHEN S2 =>  stato_prossimo <= S3;
      WHEN S3 =>  stato_prossimo <= S4;
      WHEN S4 =>  stato_prossimo <= S0;
      WHEN OTHERS => stato_prossimo <= S0;
    END CASE;
  END PROCESS;

 --Terzo processo, per la definzione delle uscite (combinatorio)
  p_uscite: PROCESS (stato_corrente)
  BEGIN
    CASE stato_corrente IS
      WHEN S0 =>  x <= '1';
      WHEN S1 =>  x <= '0';
      WHEN S2 =>  x <= '0';
      WHEN S3 =>  x <= '0';
      WHEN S4 =>  x <= '0';
      WHEN OTHERS => x <= '-';
    END CASE;
  END PROCESS p_uscite;

END arc_fsm;
```

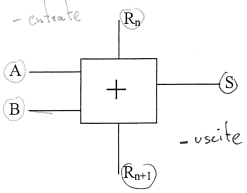
Riassunto tecnica digitale

Sommatore

Addizione Binaria:



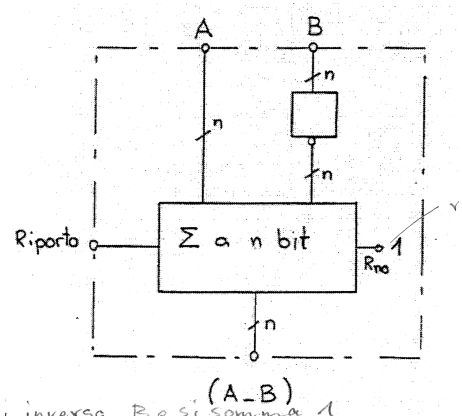
Per eseguire addizioni con più numeri a più bite c'é bisogno un addizionatore che tenga in conto il riporto.



A, B = entrate da sommare
R_n = riporto dell'addizione precedente
S = somma di A + B
R_{n+1} = riporto verso l'addizione seguente

Sotrattore

Per sottrarre un numero da un'altro bisogna inversare uno dei due dati e aggiungere 1. Questo per fare in complemento a 2



Riassunto tecnica digitale

Complemento a 2

Se un bus di 4 bit il MSB é a 1 vuol dire che é un numero negativo.

Le operazioni che seguono valgono solo se il numero in questione é negativo!

Da binario a decimale:
come calcolare il valore di 1001

	(1001)
1. **Invertiamo i singoli bit**:	0110
2. **Sommiamo 1:**	0110+
	0001
	0111 (7) ==> Risultato: -7

Da decimale a binario
trasformare il numero -5

1. **Conversione del valore (senza segno)**:	0101
2. **Inversione dei singoli bit**:	1010
3. Sommiamo 1:	1010+
	0001
	1011 <== Risultato

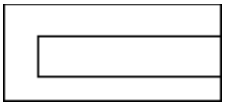
Riassunto tecnica digitale

C micro

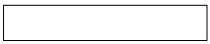
Riassunto tecnica digitale

Struttogrammi

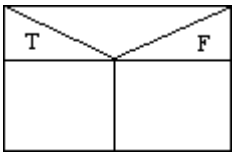
Main:



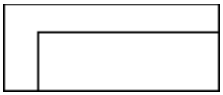
singola istruzione:



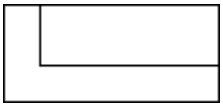
If:



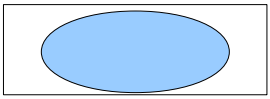
While ..... For



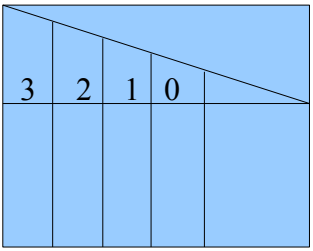
Do while



chiamata funzione



Switch



Riassunto tecnica digitale

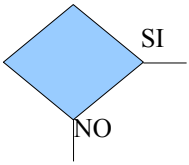
Diagrammi di flusso

inizio fine:

variabili:

input:

if:



chiamata funzione: