

# LABORATORIO DI ARCHITETTURA DEI CALCOLATORI lezione n° 12

Prof. Rosario Cerbone

rosario.cerbone@libero.it

<http://digilander.libero.it/rosario.cerbone>

a.a. 2005-2006

## L'approccio controllore-data path.

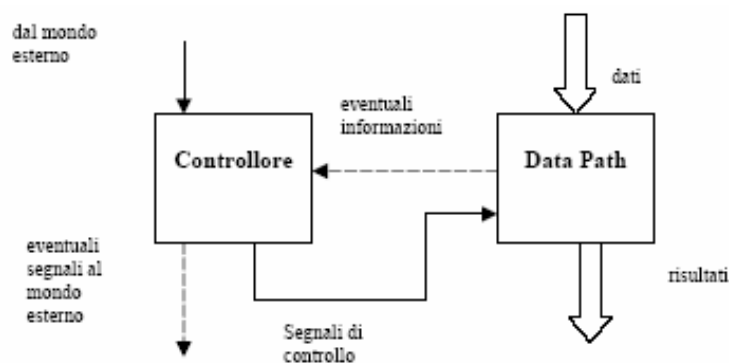
- In molti casi la specifica di un sistema digitale complesso è tale che il progetto non risulta ragionevolmente affrontabili in termini "monolitici" – tipicamente, come una singola macchina sequenziale sincrona: piuttosto, il progettista può individuare in una fase preliminare di analisi delle specifiche un insieme di funzionalità (spesso funzionalità note, quali unità aritmetiche, decodificatori, etc.) fra le quali i dati devono fluire secondo un ordine e una sincronizzazione ben precisi, eventualmente influenzati anche dal risultato di qualche operazione.

## L'approccio controllore-data path.

- Il progetto può dunque essere visto in termini di:
- 1) Definizione dell'insieme di operazioni necessarie per espletare le attività indicate dalle specifiche; corrispondente identificazione delle classi di unità funzionali richieste;
- 2) Identificazione delle specifiche operazioni svolte dalle unità funzionali, della necessità di memorizzazione di dati e risultati intermedi, definizione della sequenza di operazioni e memorizzazioni;
- 3) Progetto della sezione "operativa" che esegue le operazioni e le memorizzazioni sui dati, costituita da unità funzionali e registri (indicata normalmente come *Data-path*);
- 4) Identificazione dei segnali di controllo che devono garantire la corretta attivazione delle unità funzionali, nonché il flusso dei dati all'interno dell'architettura e la scrittura di risultati intermedi e finali negli elementi di memoria;
- 5) Progetto del *controllore* (macchina a stati finiti) che genera i segnali di controllo per il Data-Path.

## L'approccio controllore-data path.

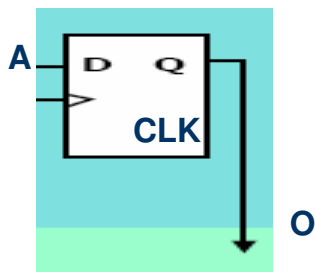
- Uno schema di larga massima dell'architettura risultante è riportato in figura:



## Elementi di un *data path*

- Registri e contatori
- Register file e FIFO
- Sommatore e comparatori
- Moltiplicatori
- Controllo hardwired
- Controllo microprogrammato

## Registro a 1 bit



### Registro a 1 bit

```
.model REGISTRO
.inputs A
.outputs O
.latch A O re NIL 0
.end
```

```
.latch <input> <output> [<type> <control>] [<init-val>]
```

*input* è l'ingresso del latch.

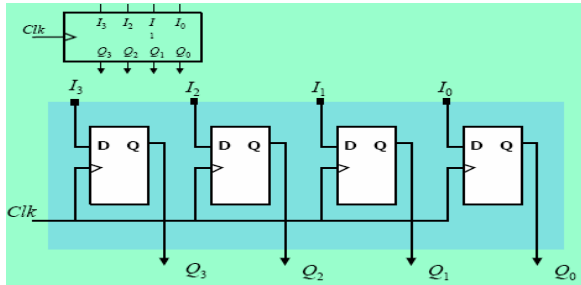
*output* l'uscita del latch.

*type* può essere {fe, re, ah, al, as}, che corrispondono a "falling edge," "rising edge," "active high," "active low," or "asynchronous."

*control* è il segnale di clock per il latch. Può essere un *.clock* del modello, l'uscita di una qualsiasi funzione del modello, o la parola "NIL" per nessun clock interno. Ciò significa che il registro utilizza il clock generale del circuito in cui è inserito.

*init-val* è lo stato iniziale del latch, che può essere {0, 1, 2, 3}. "2" indica "don't care" e "3" è "unknown." o non specificato.

## Registro a 4 bit



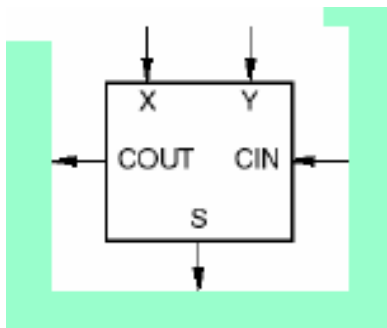
### Registro a 1 bit (registro.blif)

```
.model REGISTRO
.inputs A
.outputs O
.latch A O re NIL 0
.end
```

### Registro a 4 bit

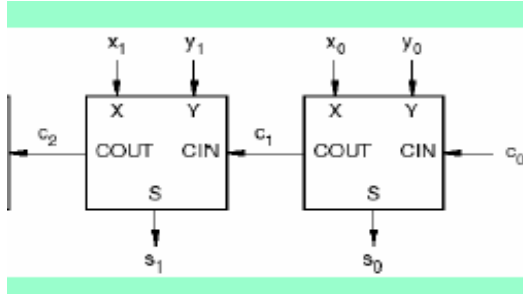
```
.model REGISTRO4
.inputs I3 I2 I1 I0
.outputs Q3 Q2 Q1 Q0
.subckt REGISTRO A=I3 O=Q3
.subckt REGISTRO A=I2 O=Q2
.subckt REGISTRO A=I1 O=Q1
.subckt REGISTRO A=I0 O=Q0
.end
.search registro.blif
```

## Sommatore *full adder*



```
.model SOMMATORE
.inputs X Y CIN
.outputs S COUT
.names X Y K
10 1
01 1
.names K CIN S
10 1
01 1
.names X Y CIN COUT
11- 1
1-1 1
-11 1
.end
```

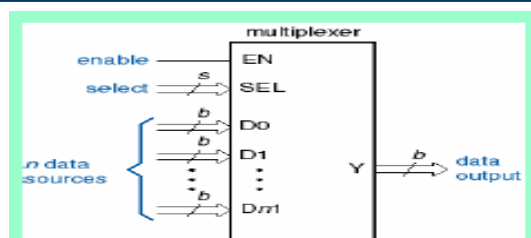
## Sommatore a 2 bit



```
Sommatore.blif
.model SOMMATORE
.inputs X Y CIN
.outputs S COUT
...
end
```

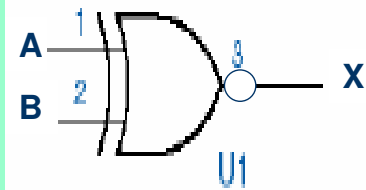
```
.model SOMMATORE2
.inputs X1 X0 Y1 Y0 C0
.outputs S1 S0 C2
.subckt SOMMATORE X=X0 Y=Y0
CIN=C0 S=S0 COUT=C1
.subckt SOMMATORE X=X1 Y=Y1
CIN=C1 S=S1 COUT=C2
.end
.search sommatore.blif
```

## Multiplexer a 2 ingressi da 3 bit ciascuno



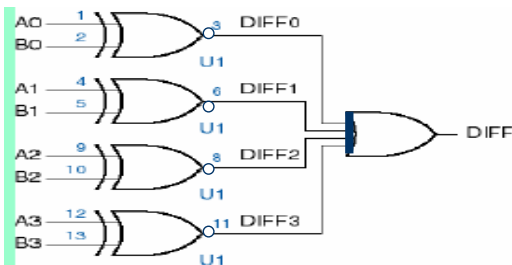
```
.model MUX3
.inputs D02 D01 D00 D12 D11 D10 S
.outputs Y2 Y1 Y0
.names S D02 D12 Y2
11- 1
0-1 1
.names S D01 D11 Y1
11- 1
0-1 1
.names S D00 D10 Y0
11- 1
0-1 1
.end
```

## Comparatore di eguaglianza a 1 bit



```
xnor.blif
.model comparatore1bit
.inputs A B
.outputs X
.names A B X
00 1
11 1
.end
```

## Comparatore di eguaglianza a 4 bit



```
.model UGUALE4
.inputs A3 A2 A1 A0 B3 B2 B1 B0
.outputs DIFF
.subckt xnor A=A3 B=B3 X=DIFF3
.subckt xnor A=A2 B=B2 X=DIFF2
.subckt xnor A=A1 B=B1 X=DIFF1
.subckt xnor A=A0 B=B0 X=DIFF0
.names DIFF3 DIFF2 DIFF1 DIFF0 DIFF
1111 1
.search xnor.blif
.end
```

## Maggiore a 8 bit

```
• .model 8_gt
• .inputs A7 A6 A5 A4 A3 A2 A1 A0 B7 B6 B5 B4 B3 B2 B1 B0
• .outputs AgtB
• .subckt xor A=A7 B=B7 X=X7
• .subckt xor A=A6 B=B6 X=X6
• .subckt xor A=A5 B=B5 X=X5
• .subckt xor A=A4 B=B4 X=X4
• .subckt xor A=A3 B=B3 X=X3
• .subckt xor A=A2 B=B2 X=X2
• .subckt xor A=A1 B=B1 X=X1
• .subckt xor A=A0 B=B0 X=X0
• .names A7 A6 A5 A4 A3 A2 A1 A0 X7 X6 X5 X4 X3 X2 X1 X0 AgtB
• 1-----1----- 1
• -1-----01----- 1
• --1-----001----- 1
• ---1----0001---- 1
• ----1---00001--- 1
• -----1--000001-- 1
• -----1-0000001- 1
• -----100000001 1
• .search xor.blif
• .end
```

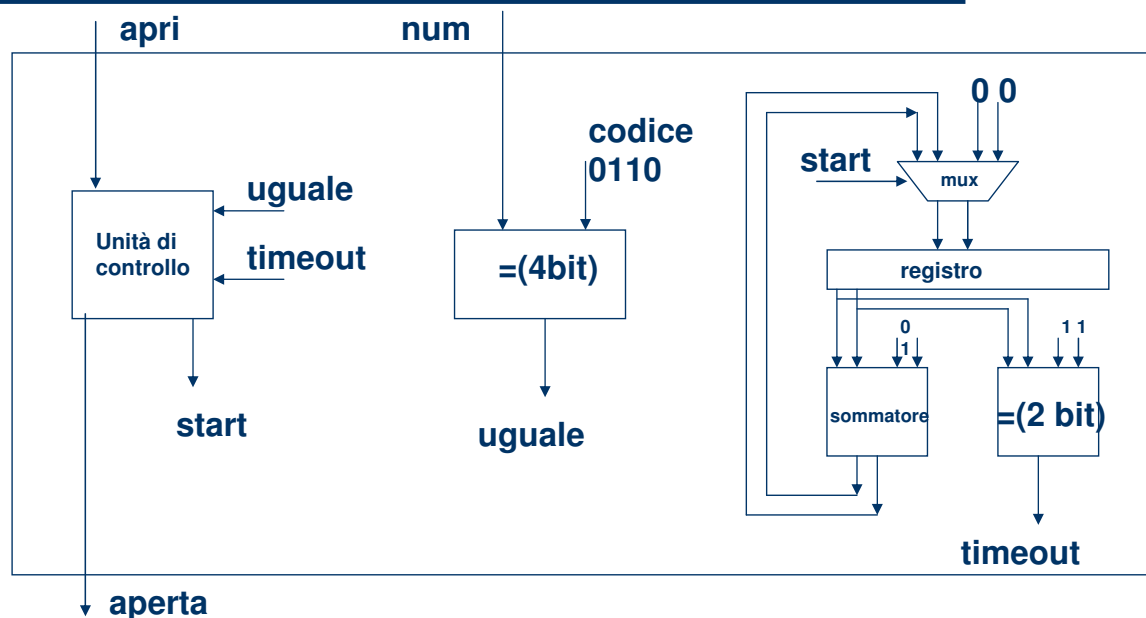
## Librerie

- Aggiungere alle librerie precedenti i seguenti modelli in blif:
- Multiplexer 4 in - 1 out
- Decodificatore 2 in - 4 out
- Decodificatore BCD - decimale
- Codificatore binario 4 in – 2 out
- Codificatore decimale - BCD

## Esercizio 12.1

- Realizzare in formato blif la FSMD del circuito che controlla l'apertura di una cassaforte.
- Gli ingressi del circuito sono:
  - - APRI[1]: vale 1 quando viene premuto il bottone *invio* per aprire la cassaforte. In tal caso il circuito deve controllare che il valore di NUM sia uguale a "0110" per aprire la cassaforte.
  - - NUM[4]: è il codice a 4 bit della combinazione.
- Le uscite del circuito sono:
  - - APERTA[1]: viene impostato a 1 quando viene premuto il bottone *invio* e NUM=0110. Viene impostato a 0 quando viene premuto il bottone *chiudi* (**APRI=0**) per richiudere la cassaforte oppure dopo che sono trascorsi 4 cicli di clock da quando la cassaforte è stata aperta.

## Esercizio 12.1





## Esercizio 12.2

- Realizzare in formato blif il seguente datapath:
- A e B sono due numeri positivi da 2 bit. Se  $SEL=0$  allora OUT fornisce il valore della somma tra A e B; se  $SEL=1$  allora OUT fornisce il valore della differenza tra A e B.

