Capitolo 12: Realizzazione dell'Unita' di Controllo

Reti Logiche
Contemporary Logic Design

Randy H. Katz
University of California, Berkeley

Trasparenze tradotte da:
Luciano Lavagno

Revisione: Tiziano Villa

Sommario del capitolo

Varie tecniche di realizzazione di FSM basate su:

- **macchine di Mealy e Moore "classiche"**
- **Contatori programmabili (jump counter)**
- **metodi basati su microprogrammazione (ROM)**

  controllori di sequenza

  microcodice orizzontale

  microcodice verticale

Realizzazione dell'Unita' di Controllo

Varie tecniche per realizzare la FSM dell'Unita' di Controllo

- **“Logica sparsa” basata su macchine di Moore e Mealy**
  Progetto *classico* di Macchine a Stati Finiti
- **Uso della gerarchia per ridurre la complessita’**
  Decomporre la FSM in molte FSM comunicanti
- **Uso di componenti MSI: Contatori programmabili**
  Contatori, multiplexer, decodificatori
- **Microprogrammazione: metodi basati su ROM**
  Codifica diretta di stato futuro ed uscite

Realizzazione dell'Unita' di Controllo

Logica sparsa

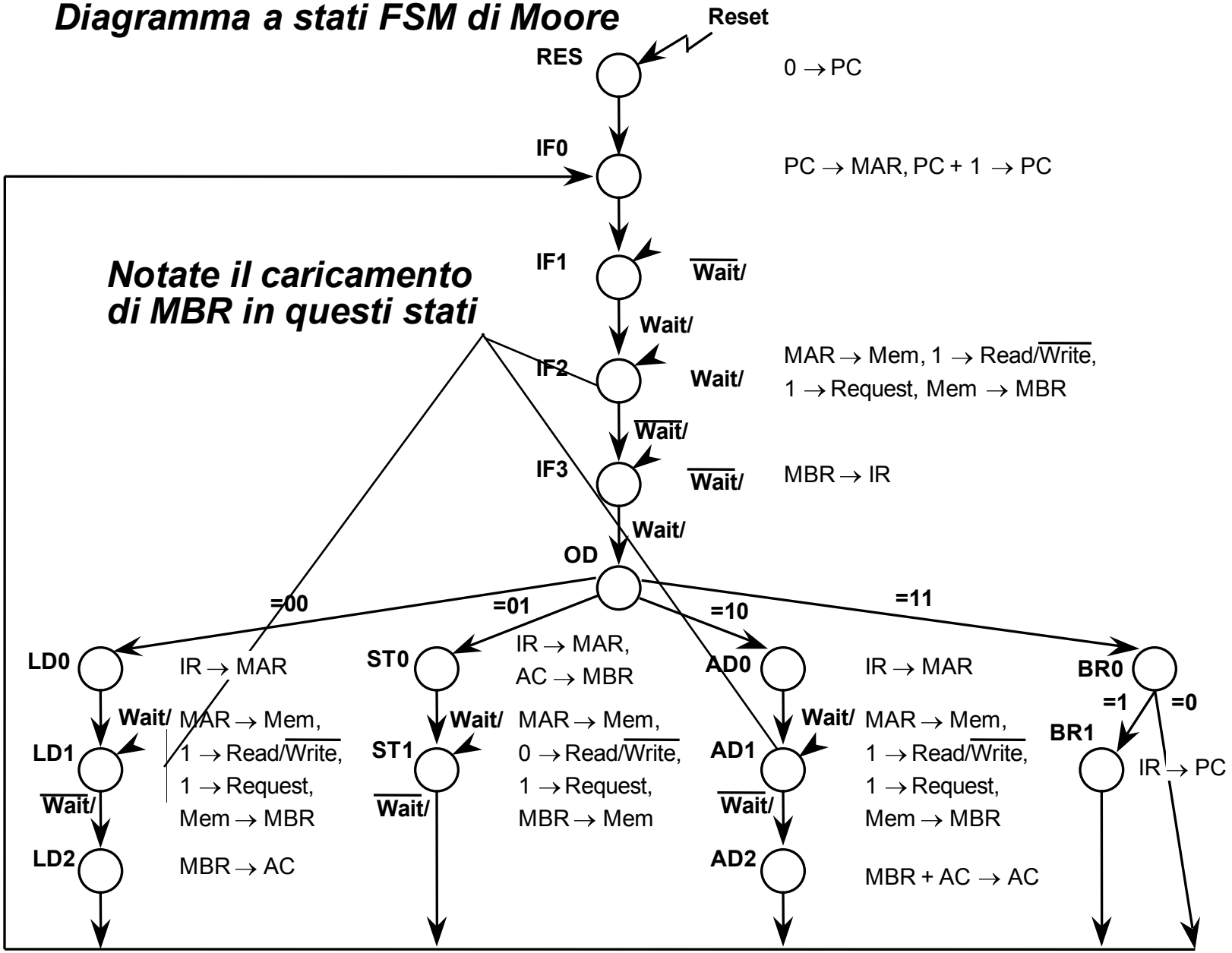
Forse un nome poco felice per la realizzazione "classica" di FSM

Usato in opposizione a logica piu' "strutturata":
PAL/PLA, FPGA, ROM

In effetti oggi macchine di Moore e Mealy sono realizzate molto sovente con tecniche strutturate usando questi componenti (ma il termine "logica sparsa" e' rimasto in uso anche per questi)

Realizzazione dell'Unita' di Controllo

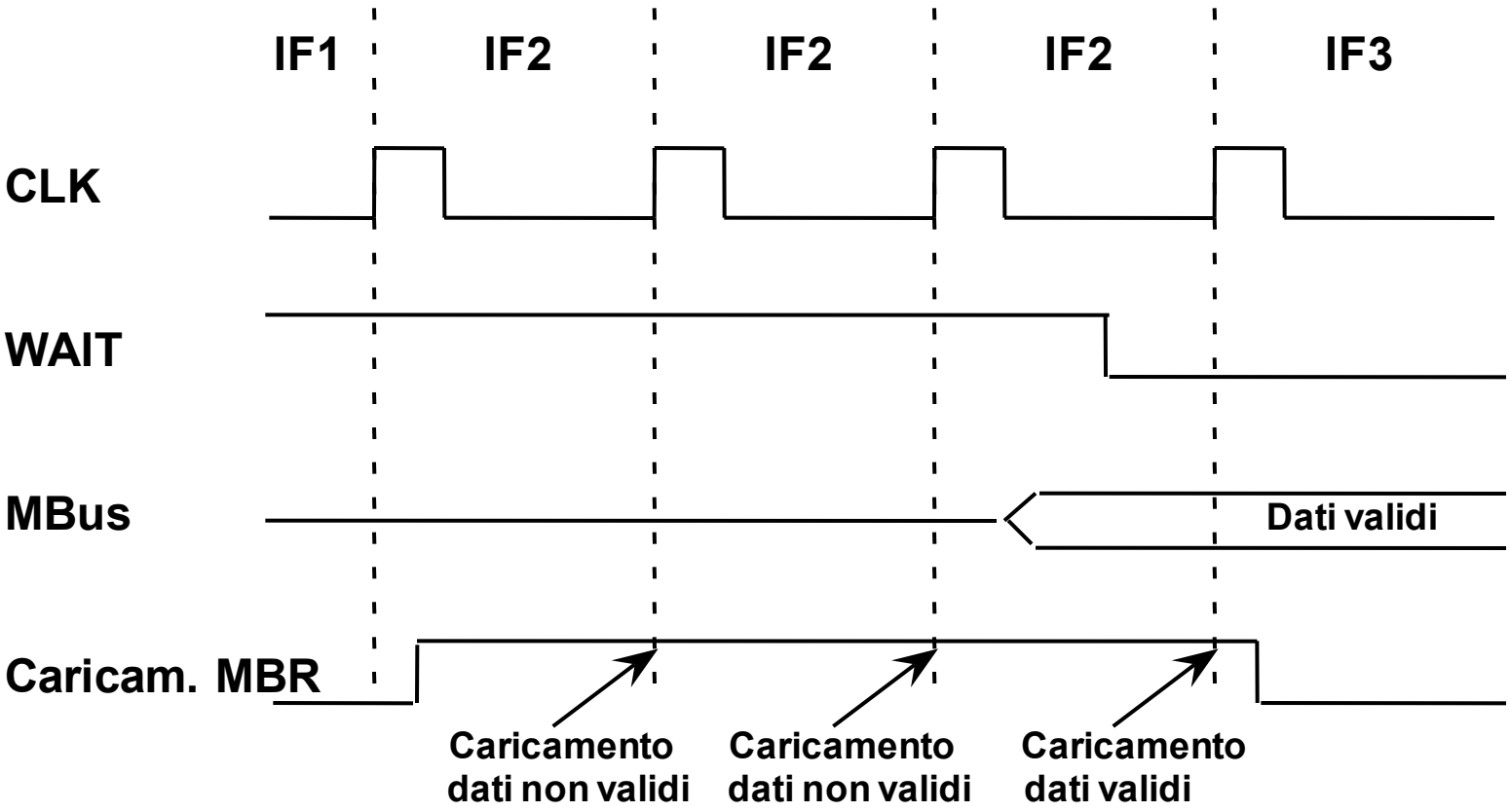
Diagramma a stati FSM di Moore



Notate il caricamento di MBR in questi stati

Realizzazione dell'Unita' di Controllo

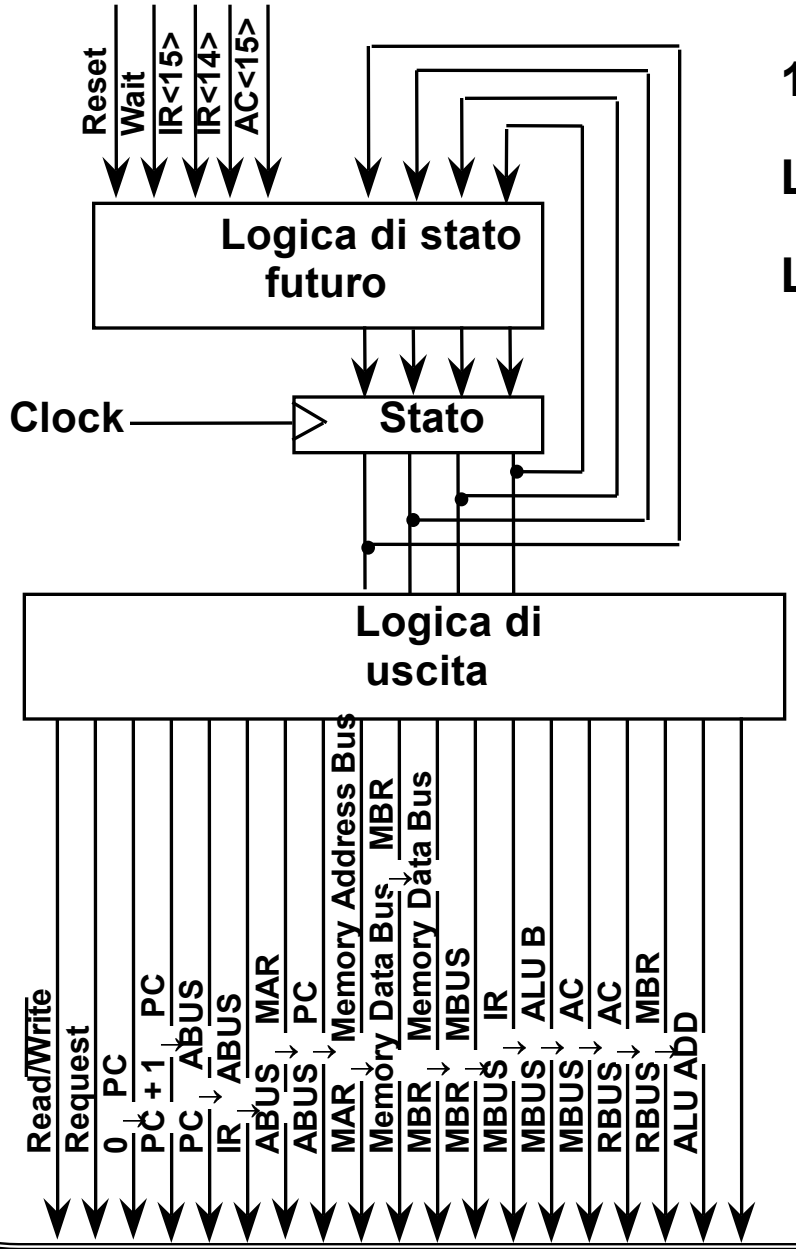
Temporizzazione interfaccia memoria-registri



Dati validi caricati sulla transizione da IF2 ad IF3 perche' i dati devono essere validi prima del fronte di discesa di Wait

Realizzazione dell'Unita' di Controllo

Schema a blocchi FSM di Moore



16 stati, registro di stato a 4 bit

Logica stato futuro: 9 ingressi, 4 uscite

Logica uscita: 4 ingressi, 18 uscite

Possono essere realizzate con ROM o con PAL/PLA

Stato futuro: ROM 512 x 4 bit
Uscite: ROM 16 x 18 bit

Realizzazione dell'Unita' di Controllo

Tabella di stato futuro in ROM

Reset	Wait	IR<15>	IR<14>	AC<15>	Stato presente	Stato futuro	Trasferimenti tra registri
1	X	X	X	X	X	RES (0000)	
0	X	X	X	X	RES (0000)	IF0 (0001)	0 → PC
0	X	X	X	X	IF0 (0001)	IF1 (0001)	PC → MAR, PC + 1 → PC
0	0	X	X	X	IF1 (0010)	IF1 (0010)	
0	1	X	X	X	IF1 (0010)	IF2 (0011)	
0	1	X	X	X	IF2 (0011)	IF2 (0011)	MAR → Mem, Read,
0	0	X	X	X	IF2 (0011)	IF3 (0100)	Request, Mem → MBR
0	0	X	X	X	IF3 (0100)	IF3 (0100)	MBR → IR
0	1	X	X	X	IF3 (0100)	OD (0101)	
0	X	0	0	X	OD (0101)	LD0 (0110)	
0	X	0	1	X	OD (0101)	ST0 (1001)	
0	X	1	0	X	OD (0101)	AD0 (1011)	
0	X	1	1	X	OD (0101)	BR0 (1110)	

Realizzazione dell'Unita' di Controllo

Tabella di uscita in ROM

Reset	Wait	IR<15>	IR<14>	AC<15>	Stato presente	Stato futuro	Trasferimenti tra registri
0	X	X	X	X	LD0 (0110)	LD1 (0111)	IR → MAR
0	1	X	X	X	LD1 (0111)	LD1 (0111)	MAR → Mem, Read,
0	0	X	X	X	LD1 (0111)	LD2 (1000)	Request, Mem → MBR
0	X	X	X	X	LD2 (1000)	IF0 (0001)	MBR → AC
0	X	X	X	X	ST0 (1001)	ST1 (1010)	IR → MAR, AC → MBR
0	1	X	X	X	ST1 (1010)	ST1 (1010)	MAR → Mem, Write,
0	0	X	X	X	ST1 (1010)	IF0 (0001)	Request, MBR → Mem
0	X	X	X	X	AD0 (1011)	AD1 (1100)	IR → MAR
0	1	X	X	X	AD1 (1100)	AD1 (1100)	MAR → Mem, Read,
0	0	X	X	X	AD1 (1100)	AD2 (1101)	Request, Mem → MBR
0	X	X	X	X	AD2 (1101)	IF0 (0001)	MBR + AC → AC
0	X	X	X	0	BR0 (1110)	IF0 (0001)	
0	X	X	X	1	BR0 (1110)	BR1 (1111)	
0	X	X	X	X	BR1 (1111)	IF0 (0001)	IR → PC

Realizzazione dell'Unita' di Controllo

Tabella delle transizioni della macchina di controllo di Moore

Osserviamo che:

- **Ci sono molti don't care**
- **Ciascun ingresso e' esaminato solo in pochi stati**
  p.es., AC<15> e' esaminato solo in BR0
  IR<15:14> sono esaminati solo in OD
- **Alcune uscite sono sempre attivate insieme**
- **Realizzazione a ROM non puo' usare i don't care**
- **Pero' una realizzazione a ROM puo' evitare la codifica degli stati**

Realizzazione dell'Unita' di Controllo

Realizzazione della macchina di Moore

Supponiamo realizzazione a PAL/PLA

Proviamo ad eseguire Espresso con una codifica semplice degli stati

21 termini prodotto

Da confrontare con le 512 parole ("termini prodotto") della realizzazione a ROM!

```
.i 9
.o 4
.ilb reset wait ir15 ir14 ac15 q3 q2 q1 q0
.ob p3 p2 p1 p0
.p 26
1----   ----    0000
0----   0001  0001
00---  0010  0010
01---  0010  0011
01---  0011  0011
00---  0011  0100
00---  0100  0100
01---  0100  0101
0-00-  0101 0110
0-01-  0101 1001
0-10-  0101 1011
0-11-  0101 1110
0----   0110  0111
01---  0111 0111
00---  0111 1000
0----  1000  0001
0----   1001 1010
01---  1010 1010
00---  1010  0001
0----  1011  1100
01---  1100 1100
00---  1100 1101
0----   1101  0001
0---0  1110  0001
0---1  1110 1111
0----   1111  0001
.e
```

```
.i 9
.o 4
.ilb reset wait ir15 ir14 ac15 q3 q2 q1 q0
.ob p3 p2 p1 p0
.p 21
0-00-0101  0110
0-01-0101  1001
0-11-0101  1110
0-10-0101  1011
01---1010   1010
00---0111   1000
00----011    0100
0----1000    0001
0---11110   1110
01---011-    0100
0----0001    0001
01---01-0    0001
0----1001    1010
0----1011    1100
00---1--0    0001
0----1100    1100
0----0-10    0010
0-----110    0001
0----11-1    0001
0----01-0    0100
01---0-1-    0011
.e
```

Realizzazione dell'Unita' di Controllo

Realizzazione della macchina di Moore

La codifica con NOVA da' risultati migliori

Risultato della codifica con NOVA

onehot_products = 22
best_products = 18
best_size = 414

18 termini prodotto (rispetto ai 21 della codifica precedente)

states[0]:IF0 Best code: 0000
states[1]:IF1 Best code: 1011
states[2]:IF2 Best code: 1111
states[3]:IF3 Best code: 1101
states[4]:OD Best code: 0001
states[5]:LD0 Best code: 0010
states[6]:LD1 Best code: 0011
states[7]:LD2 Best code: 0100
states[8]:ST0 Best code: 0101
states[9]:ST1 Best code: 0110
states[10]:AD0 Best code: 0111
states[11]:AD1 Best code: 1000
states[12]:AD2 Best code: 1001
states[13]:BR0 Best code: 1010
states[14]:BR1 Best code: 1100
states[15]:RES Best code: 1110

Realizzazione dell'Unita' di Controllo

Macchine di Mealy sincrone

Una macchina di Mealy normale ha uscite *asincrone*

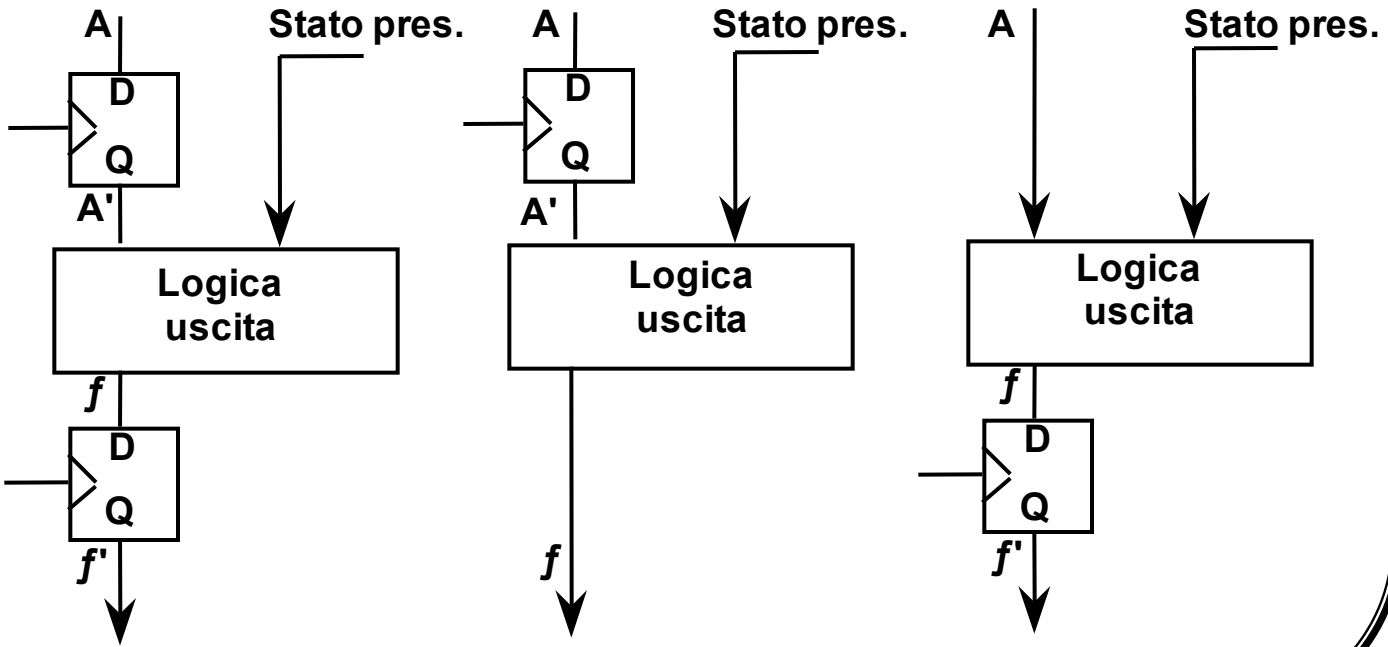
Le uscite cambiano in risposta a cambiamenti in ingresso, indipendentemente dal clock

Cambiare la realizzazione in modo che le uscite siano sincronizzate

Abbiamo gia' visto una soluzione: clock non sovrapposti

Tecniche realizzative adatte a tecnologia TTL:

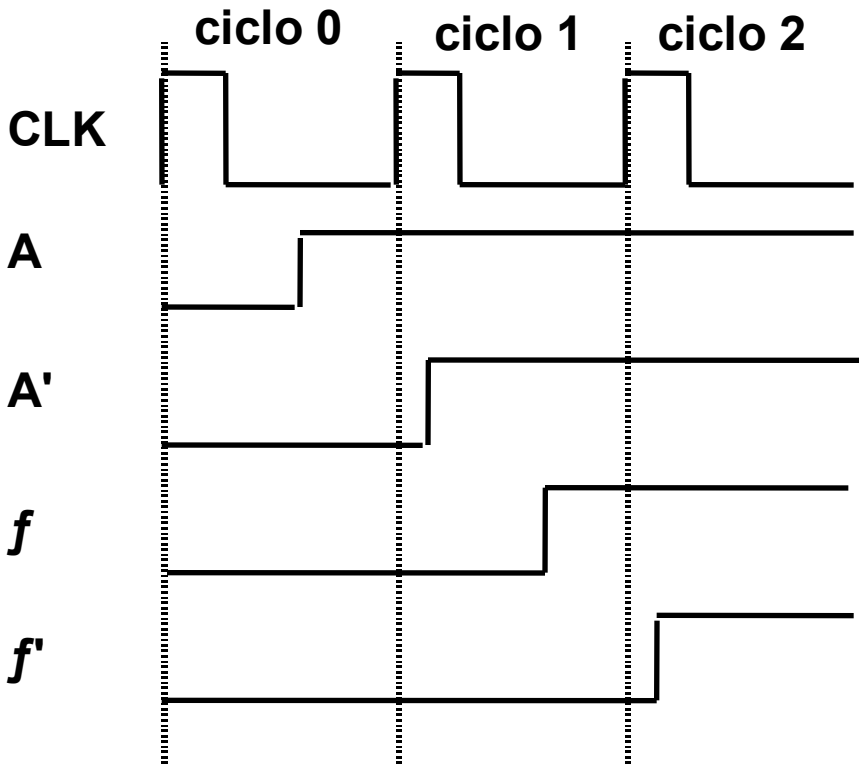
Circuiti di sincronizzazione su ingressi ed uscite

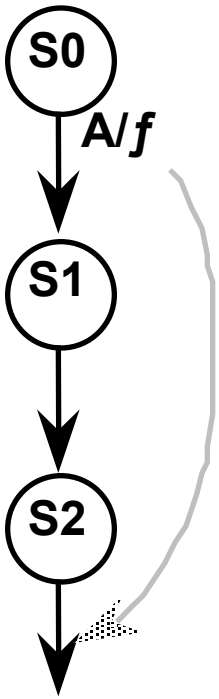


Realizzazione dell'Unita' di Controllo

Macchine di Mealy sincrone

Caso 1: sincronizzatori in ingresso ed uscita





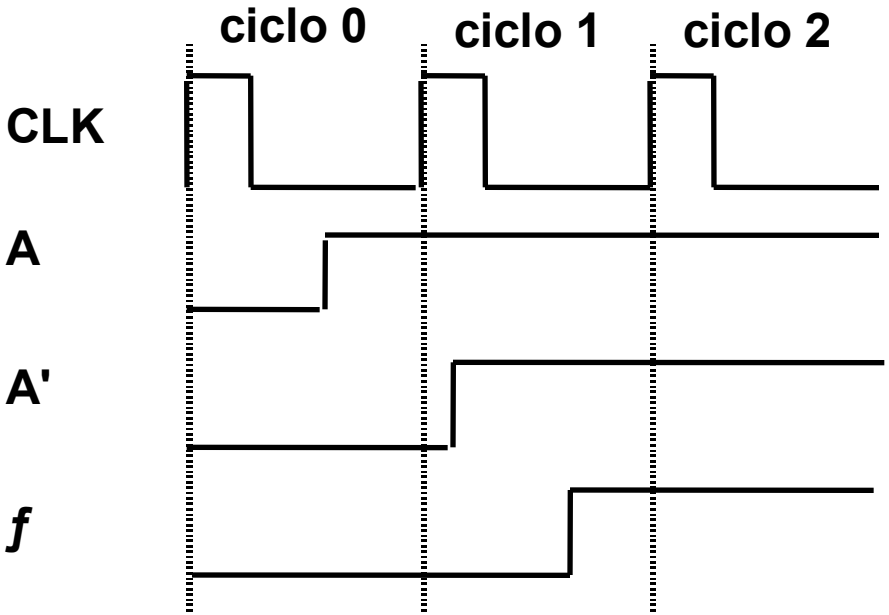
A e' attivato nel ciclo 0, *f* e' attivato dopo 2 cicli di ritardo!

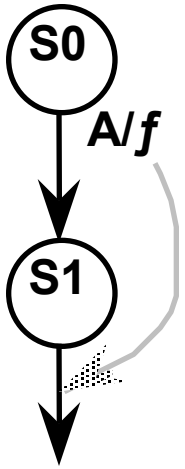
E' chiaramente troppo costoso!

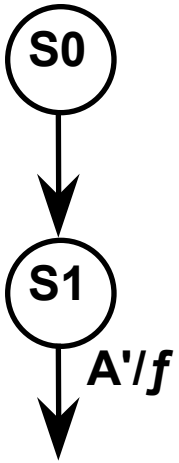
Realizzazione dell'Unita' di Controllo

Macchine di Mealy sincrone

Caso 2: sincronizzatori in ingresso







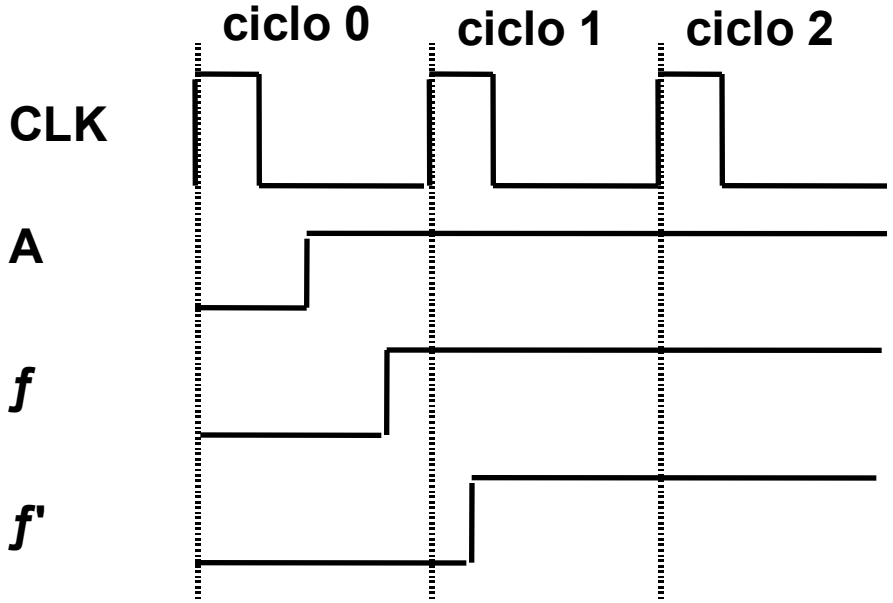
A e' attivato nel ciclo 0, *f* e' attivato nel ciclo 1

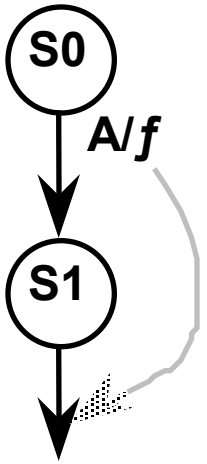
E' identico ad usare l'ingresso ritardato (A') nel ciclo 1!

Realizzazione dell'Unita' di Controllo

Macchine di Mealy sincrone

Caso 3: sincronizzatori in uscita





A e' attivato nel ciclo 0, *f* e' attivato nel ciclo 1 (come nel caso 2)

L'effetto di *f* e' ritardato di un ciclo

Realizzazione dell'Unita' di Controllo

Macchine di Mealy sincrone

Vediamo le implicazioni per la FSM di controllo gia' esaminata

Consideriamo gli ingressi Reset, Wait, IR<15:14>, AC<15>

Gli ultimi due arrivano direttamente da registri e quindi sono gia' sincronizzati rispetto al clock

Possiamo caricare IR con la nuova istruzione in uno stato ed eseguire la decisione di decodifica nel successivo

Miglior soluzione per Reset e Wait: ingressi sincronizzati

Mettere flipflop D tra questi segnali *esterni* e gli ingressi di controllo per la FSM

La versione sincronizzata di Reset e Wait e' ritardata di un ciclo di clock

Realizzazione dell'Unita' di Controllo

Decomposizione di FSM

Sommario

Metodo classico: realizzazione in un solo blocco

Metodo alternativo basato su “divide et impera”:

Decomporre la FSM in varie FSM comunicanti:

- **FSM di temporizzazione (p.es., IFetch, Decode, Execute)**
- **FSM di istruzione (p.es., LD, ST, ADD, BRN)**
- **FSM delle condizioni (p.es., AC < 0, AC $\geq$ 0)**

Realizzazione dell'Unita' di Controllo

Decomposizione di FSM

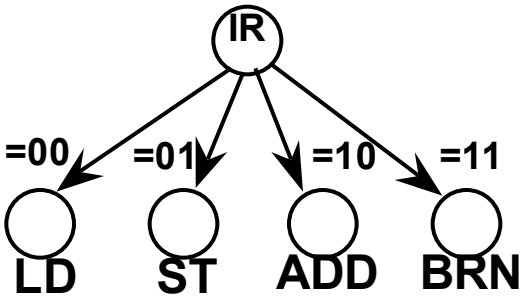
FSM di temporizzazione

La maggior parte delle istruzioni segue la stessa sequenza fondamentale

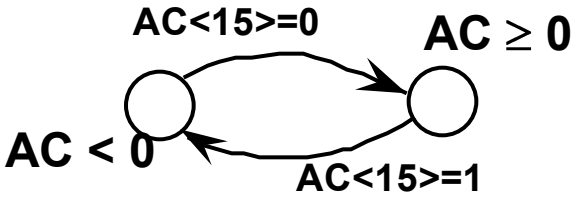
La differenza e' solo nei dettagli

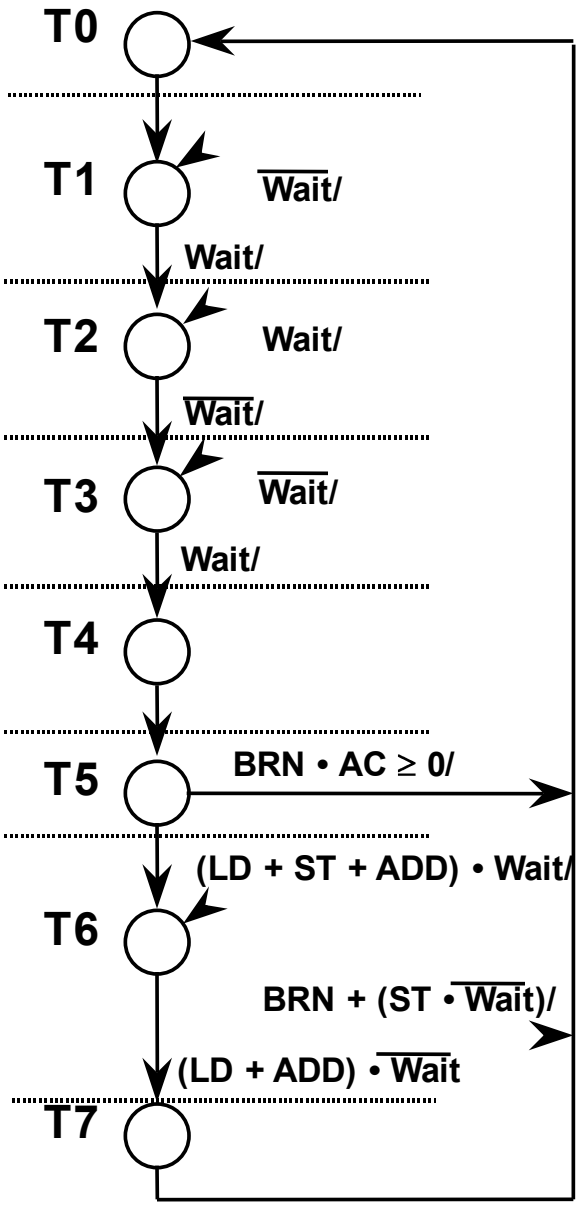
La FSM di temporizzazione puo' essere parametrizzata rispetto agli stati della FSM di istruzione e della FSM delle condizioni

Stato FSM di istruzione: memorizzato in IR<15:14>



Stato FSM delle condizioni: memorizzato in AC<15>





Realizzazione dell'Unita' di Controllo

Decomposizione di FSM

Generazione delle microoperazioni

0 → PC: Reset
PC + 1 → PC: T0
PC → MAR: T0
MAR → Memory Address Bus: T2 + T6 • (LD + ST + ADD)
Memory Data Bus → MBR: T2 + T6 • (LD + ADD)
MBR → Memory Data Bus: T6 • ST
MBR → IR: T4
MBR → AC: T7 • LD
AC → MBR: T5 • ST
AC + MBR → AC: T7 • ADD
IR<13:0> → MAR: T5 • (LD + ST + ADD)
IR<13:0> → PC: T6 • BRN
1 → Read/$\overline{\text{Write}}$: T2 + T6 • (LD + ADD)
0 → Read/$\overline{\text{Write}}$: T6 • ST
1 → Request: T2 + T6 • (LD + ST + ADD)

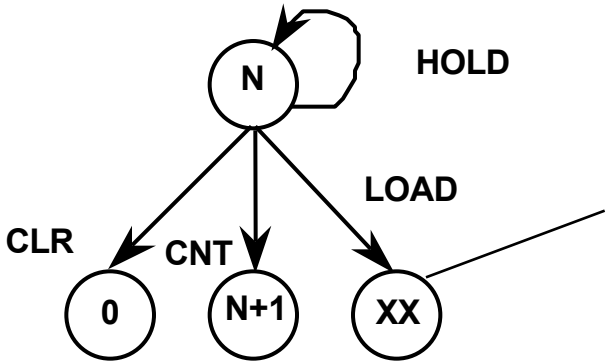
Realizzazione dell'Unita' di Controllo

Contatore programmabile

Idea fondamentale:

Realizzare una FSM usando componenti MSI: contatori, multiplexer, decodificatori

***Contatore puro*: solo 4 tipi di stato futuro**



Singolo "stato di salto" (usando caricamento) in funzione dello stato presente

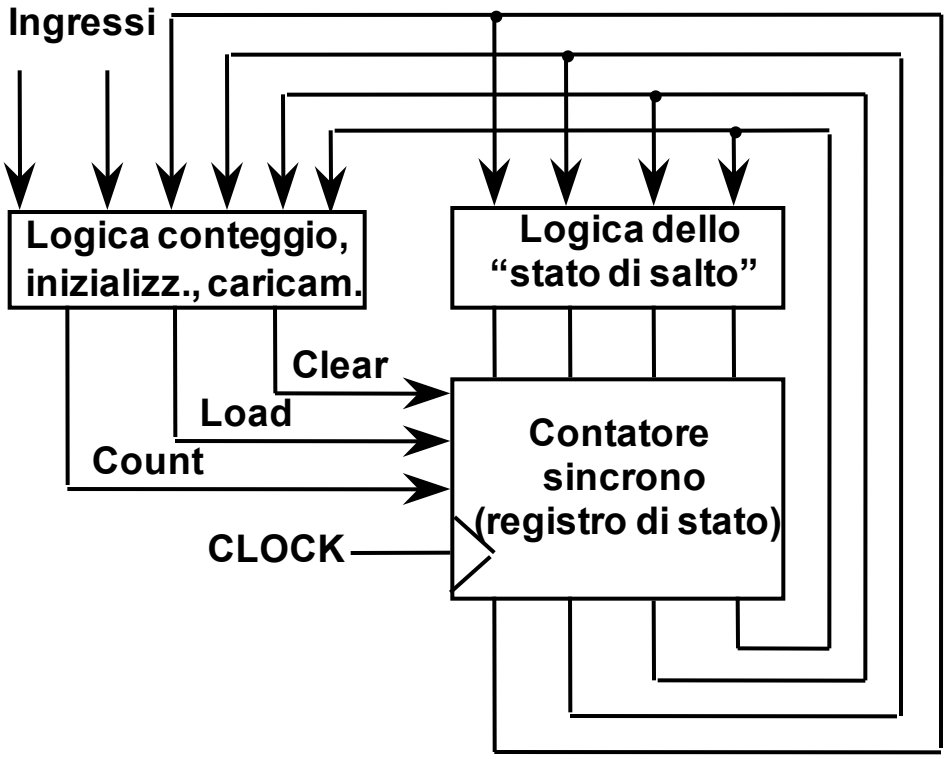
Contatore ibrido:

Molti "stati di salto" in funzione dello stato presente e degli ingressi

Realizzazione dell'Unita' di Controllo

Contatori programmabili

Contatore puro



Notare: gli ingressi NON vanno alla logica di salto

Blocchi logici realizzati a componenti discreti, PAL/PLA, ROM

Realizzazione dell'Unita' di Controllo

Contatori programmabili

Problema dei contatori puri

Difficile realizzare salti con piu' destinazioni

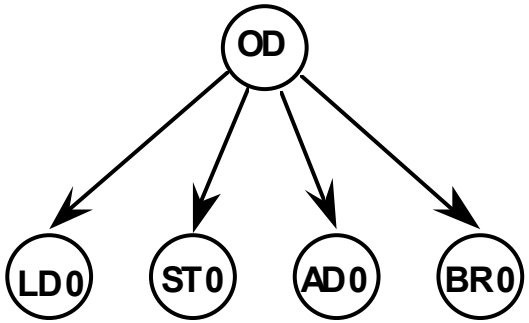


Diagramma degli stati "logico"

Stati aggiuntivi:

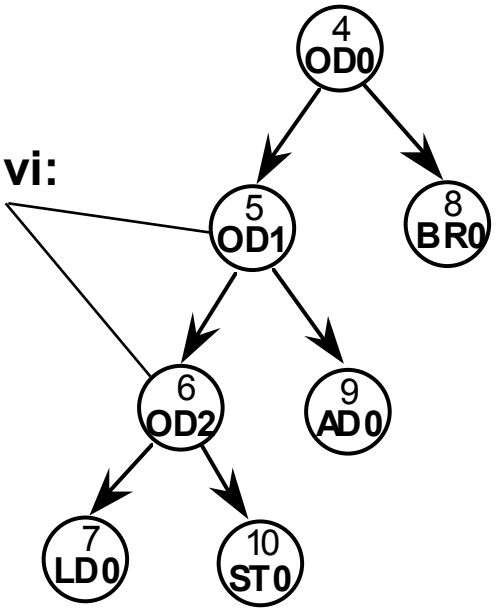
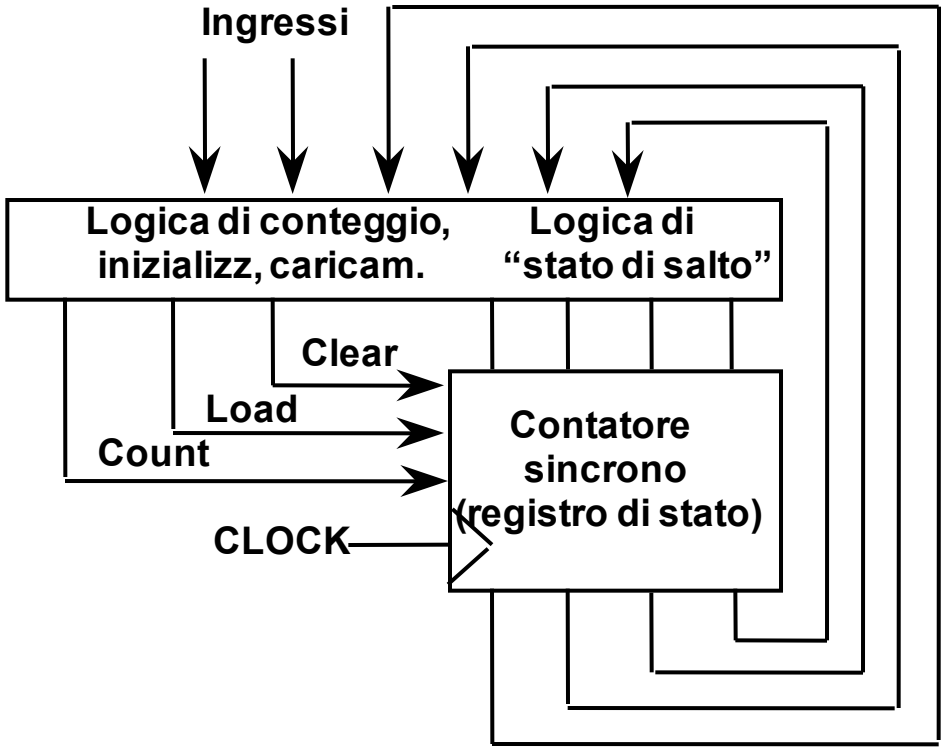


Diagramma a stati "reale" del contatore puro

Realizzazione dell'Unita' di Controllo

Contatori programmabili

Contatori ibridi



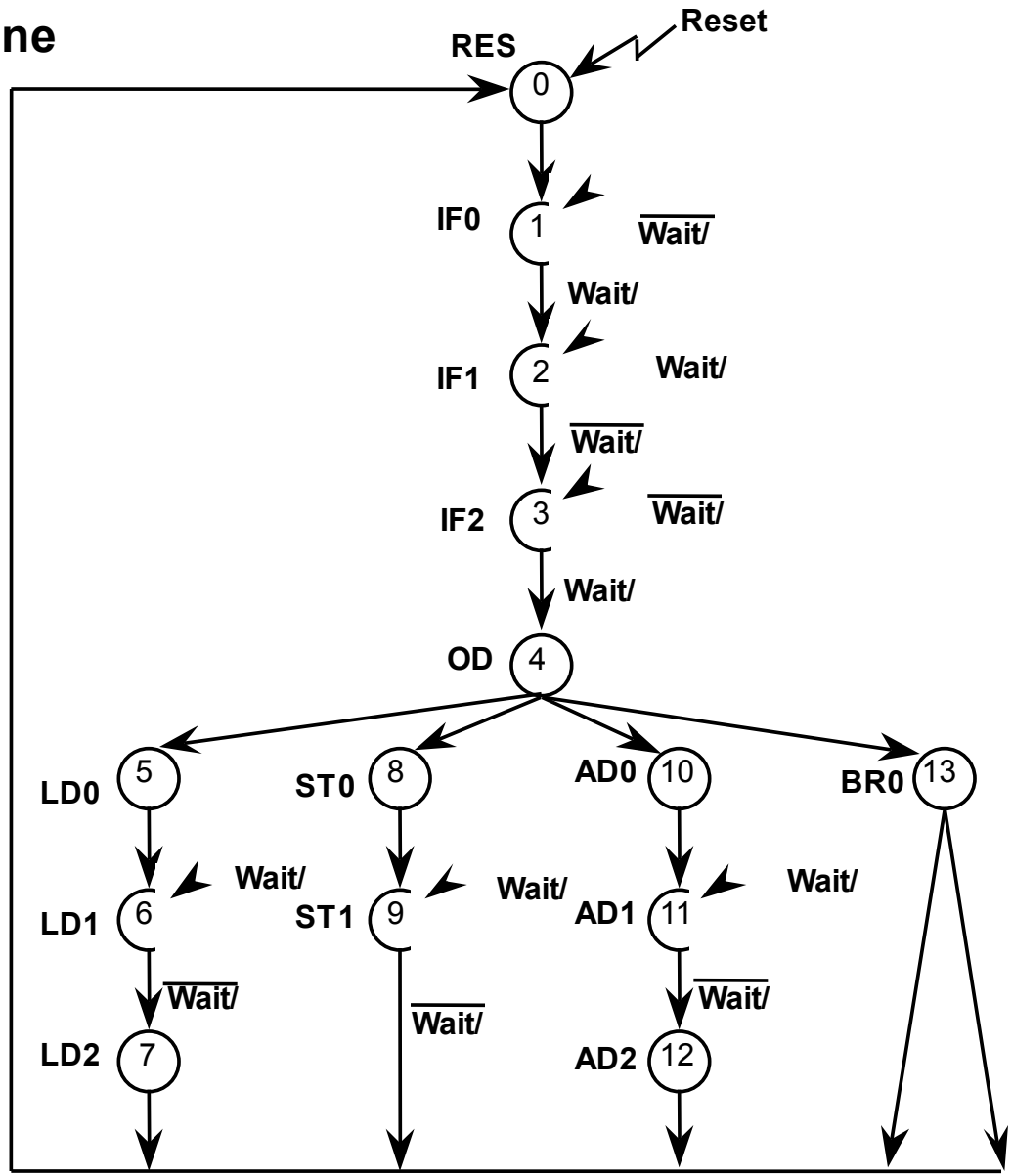
Ingressi da caricare ("stati di salto") sono funzione anche degli ingressi

Realizzazione dell'Unita' di Controllo

Contatori programmabili

Esempio di realizzazione

Codifica degli stati deve utilizzare il piu' possibile il conteggio



Realizzazione dell'Unita' di Controllo

Contatori programmabili

Esempio di realizzazione

$$CNT = (s0 + s5 + s8 + s10) + Wait \cdot (s1 + s3) + \overline{Wait} \cdot (s2 + s6 + s9 + s11)$$

$$\overline{CNT} = \overline{Wait} \cdot (s1 + s3) + Wait \cdot (s2 + s6 + s9 + s11)$$

$$CLR = Reset + s7 + s12 + s13 + (s9 \cdot \overline{Wait})$$

$$\overline{CLR} = \overline{Reset} \cdot \overline{s7} \cdot \overline{s12} \cdot \overline{s13} \cdot (\overline{s9} + Wait)$$

$$LD = s4$$

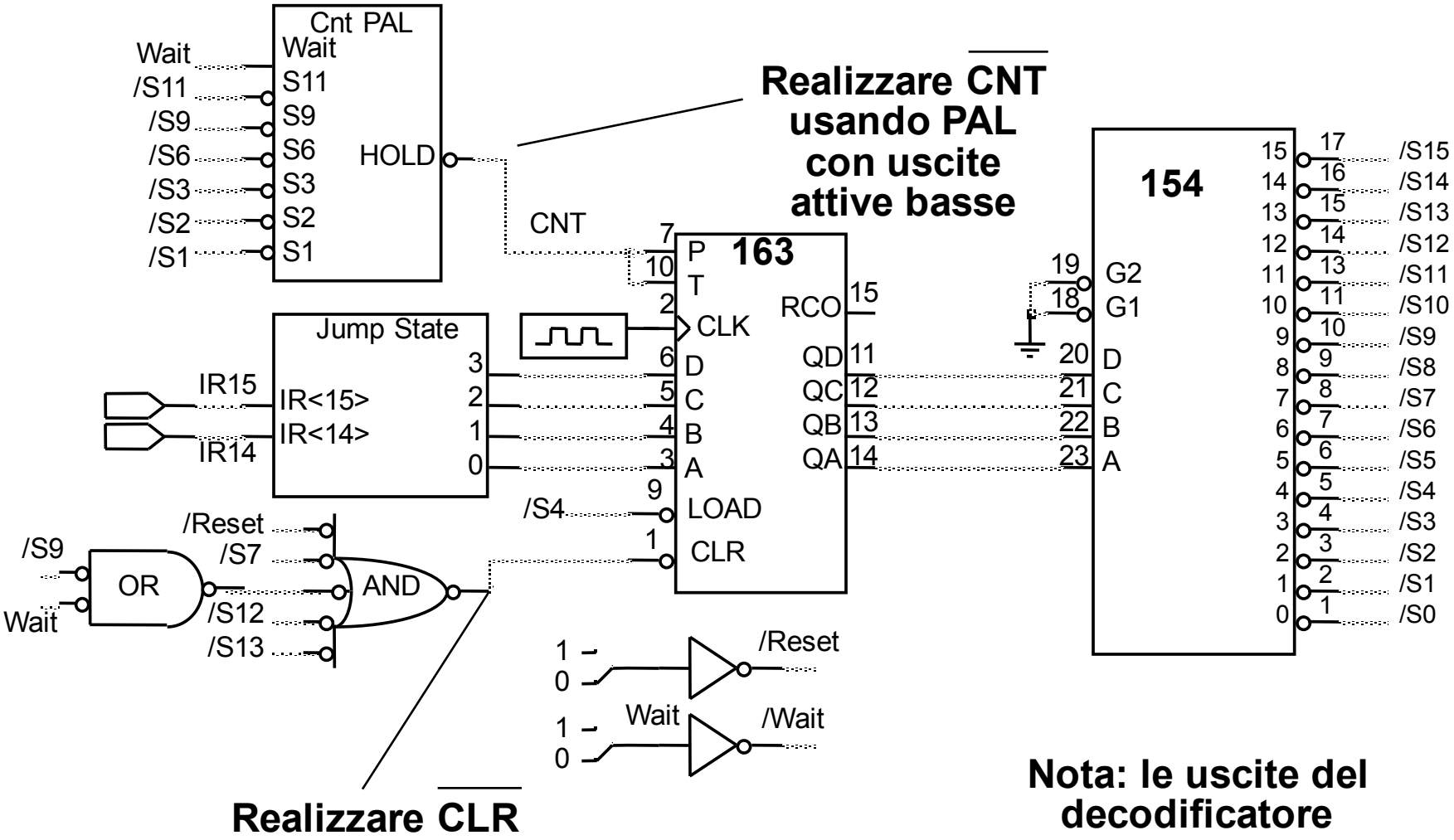
Contenuto della ROM di salto

Indirizzo	*Contenuto (stato simbolico)*
00	0101 (LD0)
01	1000 (ST0)
10	1010 (AD0)
11	1101 (BR0)

Realizzazione dell'Unita' di Controllo

Contatori programmabili

Esempio di realizzazione



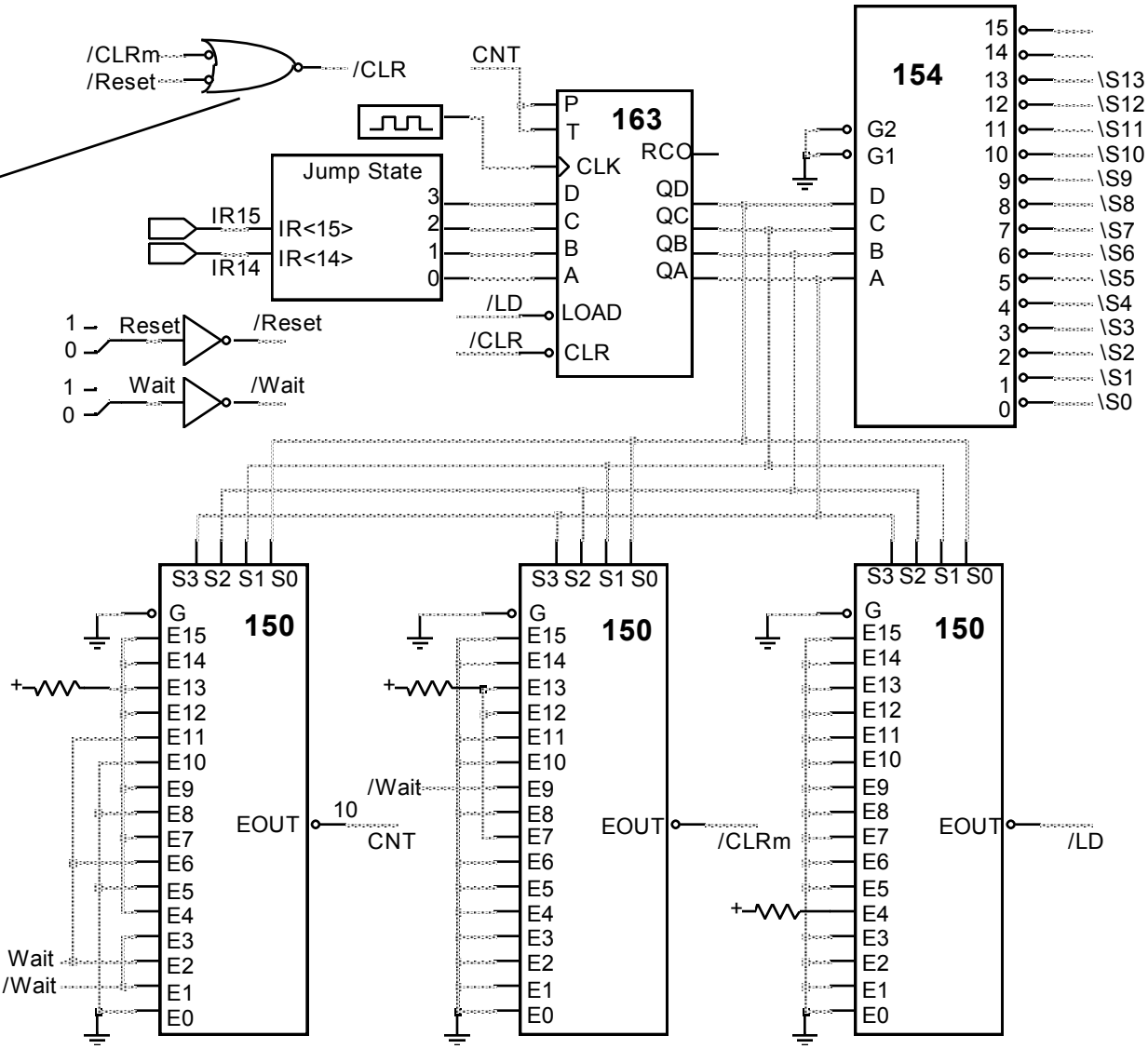
Realizzazione dell'Unita' di Controllo

Contatori programmabili

CLR, CNT, LD realizzati a multiplexer

$$CLR = CLRm + Reset$$

$$\overline{CLR} = \overline{CLRm + Reset}$$



Uscite attive basse:
ingresso attivo alto invertito all'uscita

Nota: CNT e' attivo alto nel contatore, quindi bisogna invertire gli ingressi dei MUX!

Realizzazione dell'Unita' di Controllo

Contatori programmabili

Realizzazione delle microoperazioni

$0 \rightarrow PC = Reset$
$PC + 1 \rightarrow PC = S0$
$PC \rightarrow MAR = S0$
$MAR \rightarrow$ Memory Address Bus =
$Wait \bullet (S1 + S2 + S5 + S6 + S8 + S9 + S11 + S12)$
Memory Data Bus $\rightarrow MBR = Wait \bullet (S2 + S6 + S11)$
$MBR \rightarrow$ Memory Data Bus $= Wait \bullet (S8 + S9)$
$MBR \rightarrow IR = Wait \bullet S3$
$MBR \rightarrow AC = Wait \bullet S7$
$AC \rightarrow MBR = IR15 \bullet IR14 \bullet S4$
$AC + MBR \rightarrow AC = Wait \bullet S12$
$IR<13:0> \rightarrow MAR = (IR15 \bullet IR14 + IR15 \bullet IR14 + IR15 \bullet IR14) \bullet S4$
$IR<13:0> \rightarrow PC = AC15 \bullet S13$
$1 \rightarrow$ Read/Write $= Wait \bullet (S1 + S2 + S5 + S6 + S11 + S12)$
$0 \rightarrow$ Read/Write $= Wait \bullet (S8 + S9)$
$1 \rightarrow$ Request $= Wait \bullet (S1 + S2 + S5 + S6 + S8 + S9 + S11 + S12)$

Controllo del contatore: CNT, CLR, LD dipendono da stato presente e Wait
Perche' non memorizzarli come uscite della ROM di salto?
Si possono usare Wait e stato presente come indirizzi della ROM
Il numero di parole di 7 bit cresce di 32 volte

Realizzazione dell'Unita' di Controllo

Controllori di sequenza

Idea fondamentale:

Realizzare la logica di stato futuro usando una ROM

Indirizzare la ROM con stato presente ed ingressi

Problema: la dimensione della ROM raddoppia per ogni ingresso aggiuntivo

Nota: il contatore riduce la ROM a spese di logica aggiuntiva
- **Solo gli stati di salto (fuori sequenza) sono in ROM**
- **Anche nella soluzione ibrida solo *alcuni* ingressi (oltre allo stato presente) indirizzano la ROM**

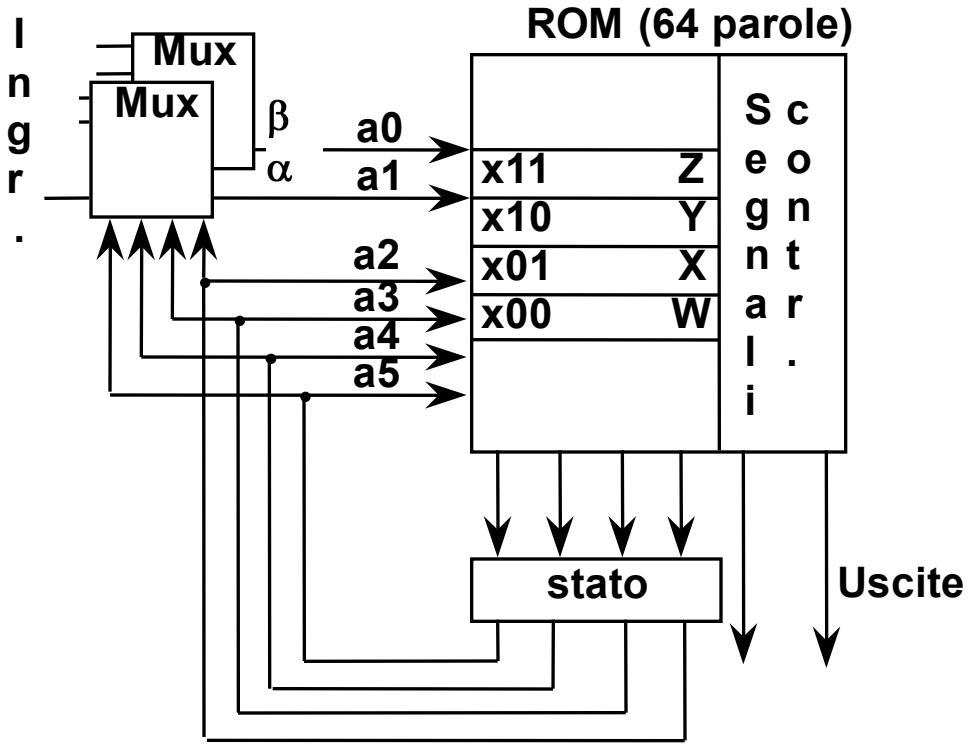
***Controllore di sequenza*: a meta' strada fra questi estremi**
- **Stato futuro memorizzato nella ROM**
- **Ogni stato ha solo pochi stati futuri**
- **Stati futuri sono comunque una potenza di 2 (o inutilizzati)**

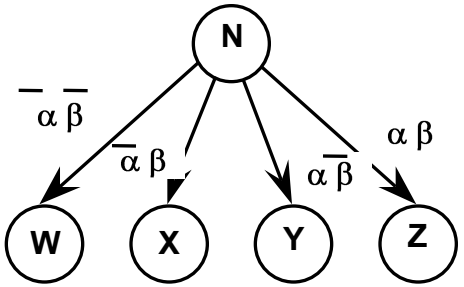
Nota: solo alcuni ingressi sono esaminati in ogni stato

Realizzazione dell'Unita' di Controllo

Controllori di sequenza

Controllore di sequenza a 4 stati futuri





Stato presente seleziona due ingressi di indirizzo della ROM

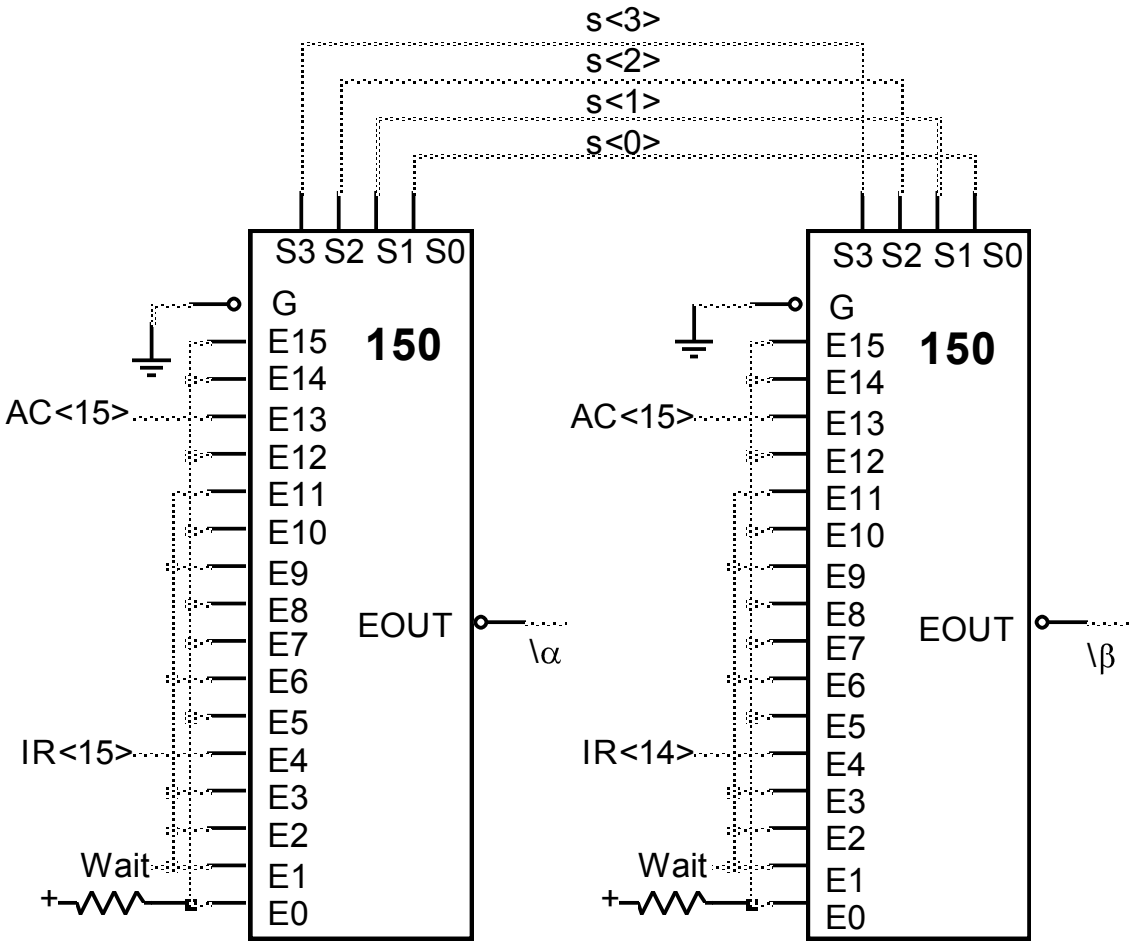
Questi selezionano uno tra quattro possibili stati futuri ed uscite

Ogni stato ha al massimo 4 stati futuri possibili

Realizzazione dell'Unita' di Controllo

Controllori di sequenza

Esempio di realizzazione del controllo del processore



Generazione degli ingressi α e β usando multiplexer

Controllori di sequenza

Esempio di realizzazione del controllo del processore

Indirizzo ROM (Reset, stato presente, a, b)					Contenuto ROM: Stato futuro	Trasferimenti tra registri (microoperazioni)
RES	0	0000	X	X	0001 (IF0)	PC → MAR, PC + 1 → PC
IF0	0	0001	0	0	0001 (IF0)	
	0	0001	1	1	0010 (IF1)	MAR → Mem, Read, Request
IF1	0	0010	0	0	0011 (IF2)	MAR → Mem, Read, Request
	0	0010	1	1	0010 (IF1)	Mem → MBR
IF2	0	0011	0	0	0011 (IF2)	
	0	0011	1	1	0100 (OD)	MBR → IR
OD	0	0100	0	0	0101 (LD0)	IR → MAR
	0	0100	0	1	1000 (ST0)	IR → MAR, AC → MBR
	0	0100	1	0	1001 (AD0)	IR → MAR
	0	0100	1	1	1101 (BR0)	IR → MAR

Controllori di sequenza

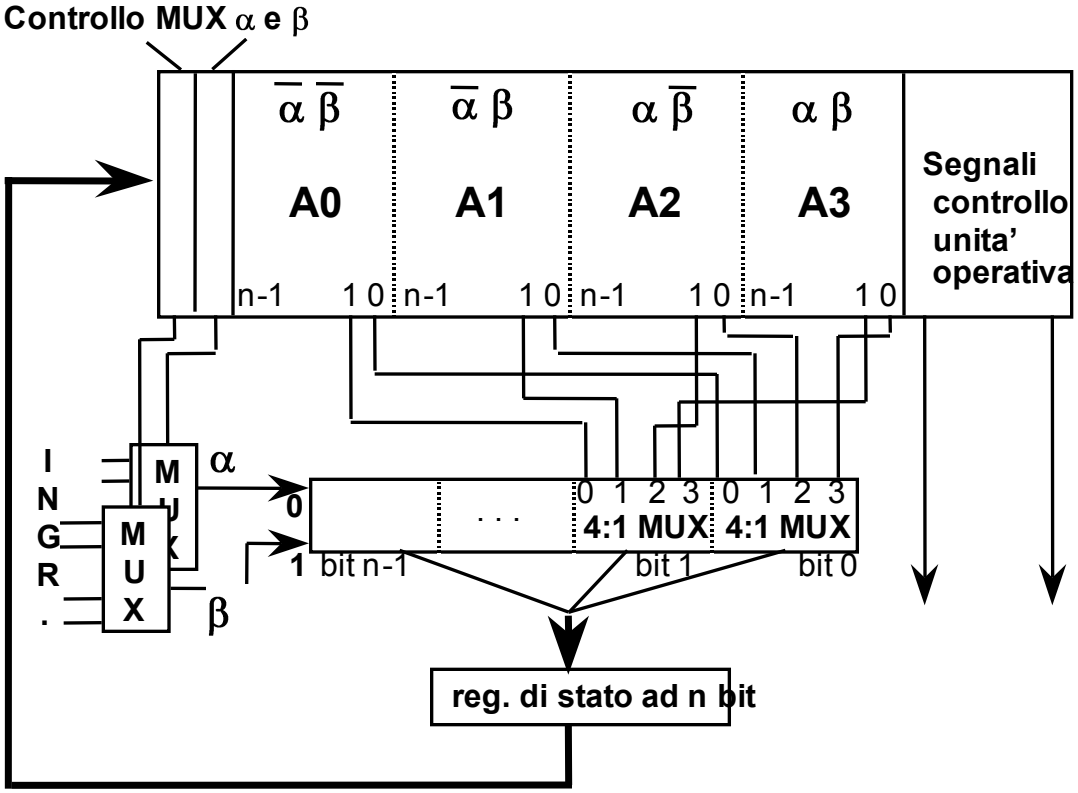
Esempio di realizzazione del controllo del processore

Indirizzo ROM (Reset, stato presente, a, b)					Contenuto ROM: Stato futuro	Trasferimenti tra registri (microoperazioni)
LD0	0	0101	X	X	0110 (LD1)	MAR → Mem, Read, Request
LD1	0	0110	0	0	0111 (LD2)	Mem → MBR
	0	0110	1	1	0110 (LD1)	MAR → Mem, Read, Request
LD2	0	0111	X	X	0000 (RES)	MBR → AC
ST0	0	1000	X	X	1001 (ST1)	MAR → Mem, Write, Request, MBR → Mem
ST1	0	1001	0	0	0000 (RES)	
	0	1001	1	1	1001 (ST1)	MAR → Mem, Write, Request, MBR → Mem
AD0	0	1010	X	X	1011 (AD1)	MAR → Mem, Read, Request
AD1	0	1011	0	0	1100 (AD2)	
	0	1011	1	1	1011 (AD1)	MAR → Mem, Read, Request
AD2	0	1100	X	X	0000 (RES)	MBR + AC → AC
BR0	0	1101	0	0	0000 (RES)	
	0	1101	1	1	0000 (RES)	IR → PC

Realizzazione dell'Unita' di Controllo

Controllori di sequenza

Realizzazione alternativa "orizzontale"



Multiplexer di ingresso controllato da segnali codificati, non dallo stato
Molti meno ingressi rispetto al numero degli stati!
Nella FSM di controllo processore, il MUX puo' essere 2:1!
Aggiungere bit alla parola della ROM risparmia (in termini di bit totali) rispetto al normale raddoppio delle parole:

Formato "verticale": (14 + 4) x 64 = 1152 bit di ROM
Formato "orizzontale": (14 + 4 x 4 + 2) x 16 = 512 bit di ROM

Realizzazione dell'Unita' di Controllo

Microprogrammazione

Metodo per organizzare i segnali di controllo

Realizzazione dei segnali di controllo memorizzando 0 ed 1 in ROM

Microprogrammazione orizzontale e verticale

Orizzontale: 1 uscita della ROM per ogni segnale di controllo
Verticale: segnali di controllo *codificati* nella ROM, decodificati esternamente
Si possono combinare segnali mutuamente esclusivi
Serve a ridurre la lunghezza di parola della ROM

Realizzazione dell'Unita' di Controllo

Microprogrammazione

Microoperazioni (trasferimenti tra registri)

14 trasferimenti tra registri diventano 22 microoperazioni:

PC → ABUS
IR → ABUS
MBR → ABUS
RBUS → AC
AC → ALU A
MBUS → ALU B
ALU ADD
ALU PASS B
MAR → Address Bus
MBR → Data Bus
ABUS → IR

ABUS → MAR
Data Bus → MBR
RBUS → MBR
MBR → MBUS
0 → PC
PC + 1 → PC
ABUS → PC
Read/Write
Request
AC → RBUS
ALU Result → RBUS

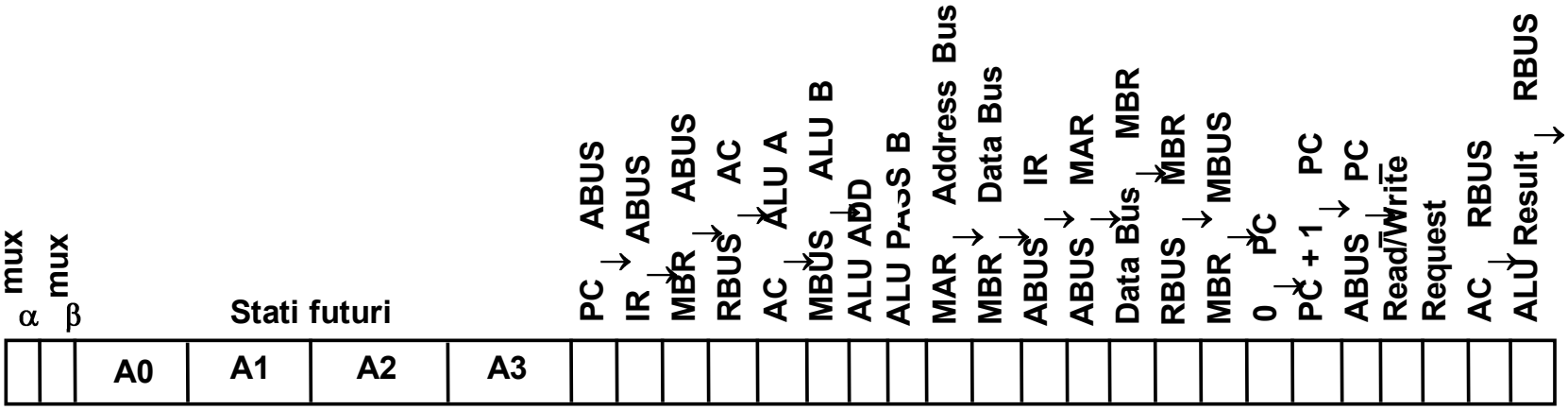
Realizzazione dell'Unita' di Controllo

Microprogrammazione orizzontale

Controllore di sequenza orizzontale

- Bit di controllo dei MUX α, β
- 4 x 4 bit di stato futuro
- 22 bit di controllo delle operazioni

40 bit in totale



Realizzazione dell'Unita' di Controllo

Microprogrammazione orizzontale

ROM per la FSM di Moore di controllo del processore

Stato pres. (indirizzo)	mux α	mux β	Stati futuri A0	A1	A2	A3	PC → ABUS	IR → ABUS	MBR → ABUS	RBUS → AC	AC → ALU A	MBUS → ALU B	ALU A + ALU B → Address Bus	ALU A PASS B → Data Bus	MAR → IR	MBR → MAR	ABU → MBR	ABUS → Data Bus	Data Bus → MBR	RBUS → MBR	MBR → MBUS	0 → PC	PC + 1 → PC	ABUS → PC	Read/Write	Request	AC → RBUS	ALU Result → RBUS
RES (0000)	0	0	0001	0001	0001	0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
IF0 (0001)	0	0	0010	0010	0010	0010	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
IF1 (0010)	0	0	0010	0010	0011	0011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IF2 (0011)	0	0	0100	0100	0011	0011	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0
IF3 (0100)	0	0	0100	0100	0101	0101	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
OD (0101)	1	1	0110	1001	1011	1110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LD0 (0110)	0	0	0111	0111	0111	0111	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
LD1 (0111)	0	0	1000	1000	0111	0111	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0
LD2 (1000)	0	0	0001	0001	0001	0001	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
ST0 (1001)	0	0	1010	1010	1010	1010	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0
ST1 (1010)	0	0	0001	0001	1010	1010	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
AD0 (1011)	0	0	1100	1100	1100	1100	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
AD1 (1100)	0	0	1101	1101	1100	1100	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0
AD2 (1101)	0	0	0001	0001	0001	0001	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
BR0 (1110)	0	1	0001	1111	0001	1111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BR1 (1111)	0	0	0001	0001	0001	0001	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Ingressi mux α: 0 = Wait, 1 = IR<15>
Ingressi mux β: 0 = AC<15>, 1 = IR<14>

Realizzazione dell'Unita' di Controllo

Microprogrammazione orizzontale

Vantaggi:

massima flessibilita': accesso completo in parallelo ai controlli dell'unita' operativa

Svantaggi:

parole della ROM molto lunghe: oltre 100 bit per processori veri!!

Nota: non tutte le combinazioni di microoperazioni hanno senso!

Codifica delle uscite:

raggruppare segnali mutuamente esclusivi
Usare logica esterna per decodificare

Esempio:

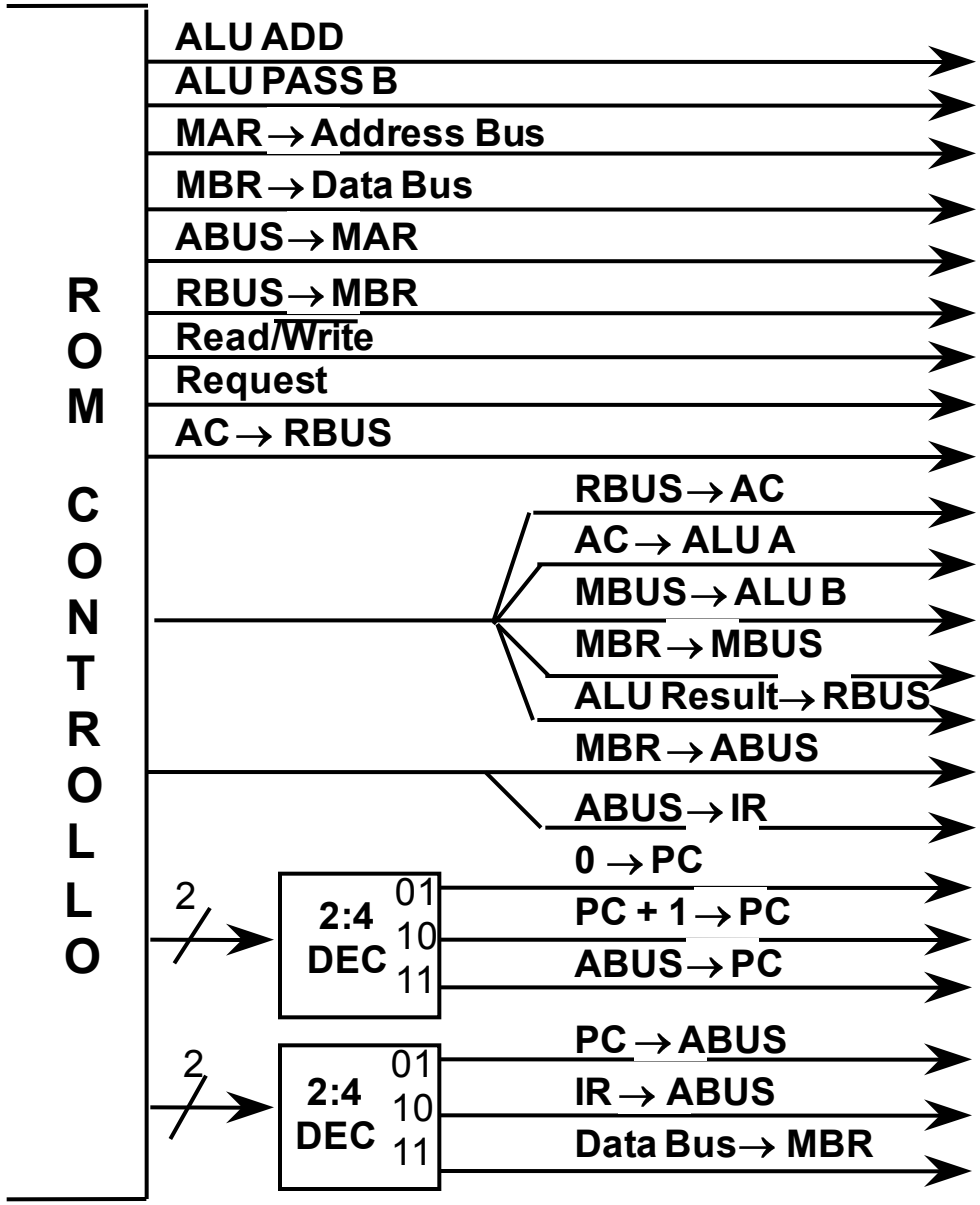
0 → PC, PC + 1 → PC, ABUS → PC sono mutuamente esclusive

Si possono risparmiare bit di ROM usando un decoder 2:4

Realizzazione dell'Unita' di Controllo

Microprogrammazione orizzontale

Uscite di controllo parzialmente codificate



Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

Codifica piu' spinta per ridurre ulteriormente la lunghezza di parola della ROM

Spesso si usano diversi formati di micro-parola:

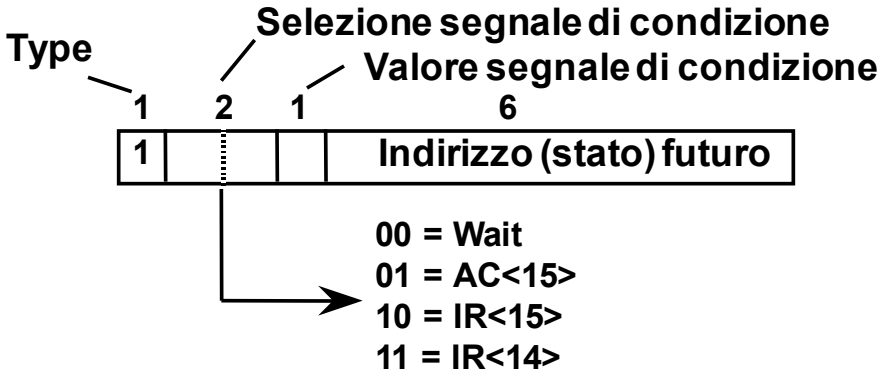
- **codifica orizzontale: stato futuro e uscite erano nella stessa parola**
- **codifica verticale: usa diversi formati per uscite e stato futuro**
- **puo' richiedere l'uso di molte micro-parole per realizzare la funzione di una sola micro-parola orizzontale**
- **se portato all'estremo, somiglia alla programmazione in linguaggio assembler**

Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

Salto condizionato
Saltare se il segnale selezionato ha il valore indicato

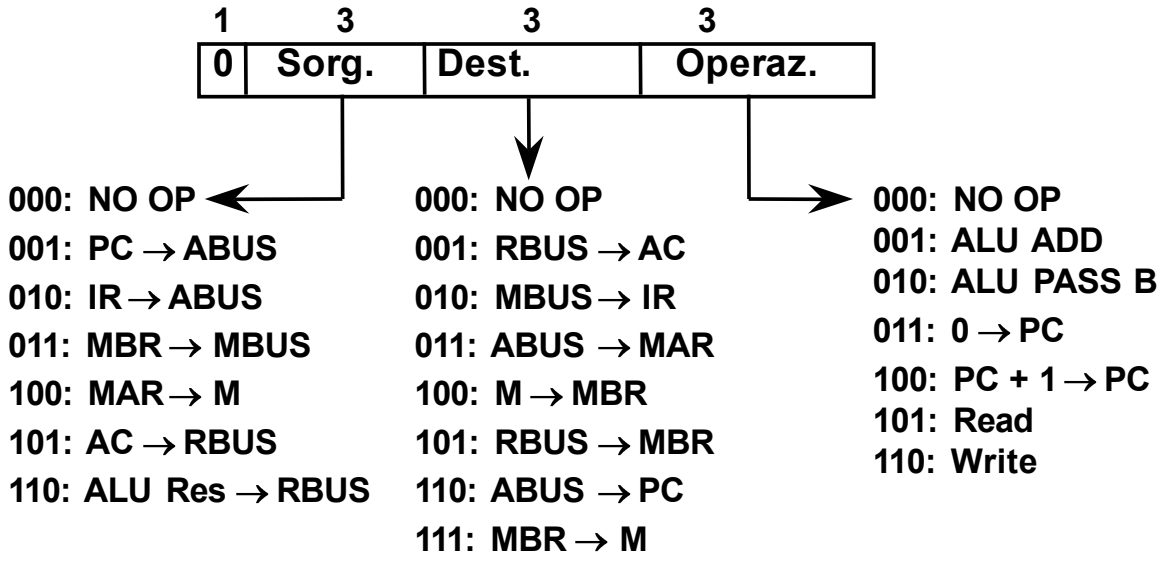
Formato micro-istruzione di salto



Trasferimento tra registri
sorgente, destinazione, operazione

10 bit di ROM

Formato micro-instruzione di trasferimento tra registri



Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

Contenuto della ROM

Indirizzo	Contenuto simbolico			Contenuto binario
000000	RES	RT	PC → MAR, PC +1 → PC	0 001 011 100
000001	IF0	RT	MAR → M, Read	0 100 000 101
000010		BJ	Wait=0, IF0	1 000 000 001
000011	IF1	RT	MAR → M, M → MBR, Read	0 100 100 101
000100		BJ	Wait=1, IF1	1 001 000 011
000101	IF2	RT	MBR → IR	0 011 010 000
000110		BJ	Wait=0, IF2	1 000 000 101
000111		RT	IR → MAR	0 010 011 000
001000	OD	BJ	IR<15>=1, OD1	1 101 010 101
001001		BJ	IR<14>=1, ST0	1 111 010 000
001010	LD0	RT	MAR → M, Read	0 100 000 101
001011	LD1	RT	MAR → M, M → MBR, Read	0 100 100 101
001100		BJ	Wait=1, LD1	1 001 001 011
001101	LD2	RT	MBR → AC	0 110 001 010
001110		BJ	Wait=0, RES	1 000 000 000
001111		BJ	Wait=1, RES	1 001 000 000

Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

Contenuto della ROM

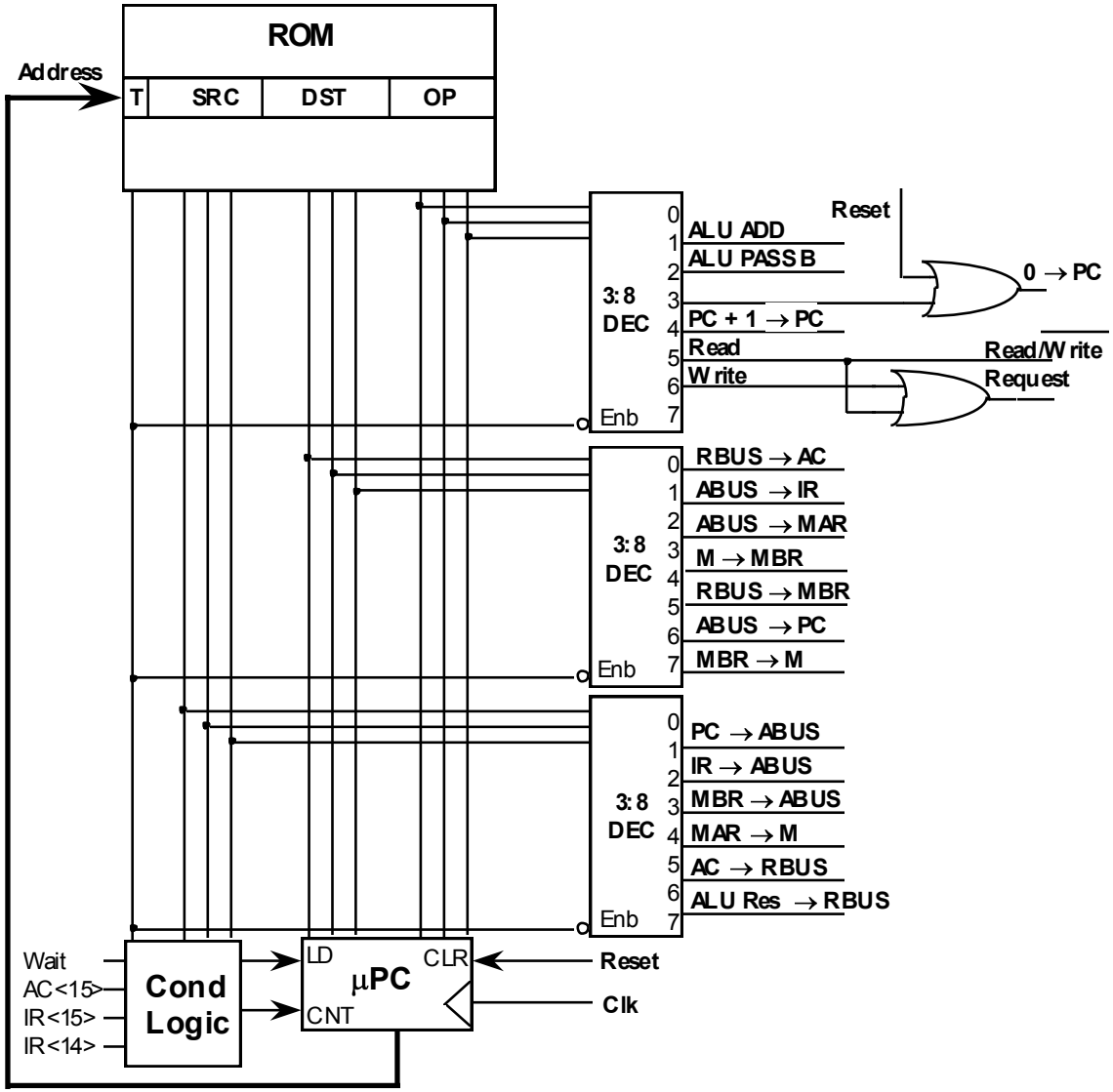
Indirizzo	Contenuto simbolico			Contenuto binario
010000	ST0	RT	AC → MBR	0 101 101 000
010001		RT	MAR → M, MBR → M, Write	0 100 111 110
010010	ST1	RT	MAR → M, MBR → M, Write	0 100 111 110
010011		BJ	Wait=0, RES	1 000 000 000
010100		BJ	Wait=1, ST1	1 001 010 010
010101	OD1	BJ	IR<14>=1, BR0	1 111 011 101
010110	AD0	RT	MAR → M, Read	0 100 000 101
010111	AD1	RT	MAR → M, M → MBR, Read	0 100 100 101
011000		BJ	Wait=1, AD1	1 001 010 111
011001	AD2	RT	AC + MBR → AC	0 110 001 001
011010		BJ	Wait=0, RES	1 000 000 000
011011		BJ	Wait=1, RES	1 000 000 000
011100	BR0	BJ	AC<15>=0, RES	1 010 000 000
011101		RT	IR → PC	0 010 110 000
011110		BJ	AC<15>=1, RES	1 011 000 000

31 parole da 10 bit di ROM = 310 in totale, rispetto a 16 x 38 = 608 bit nel caso orizzontale!

Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

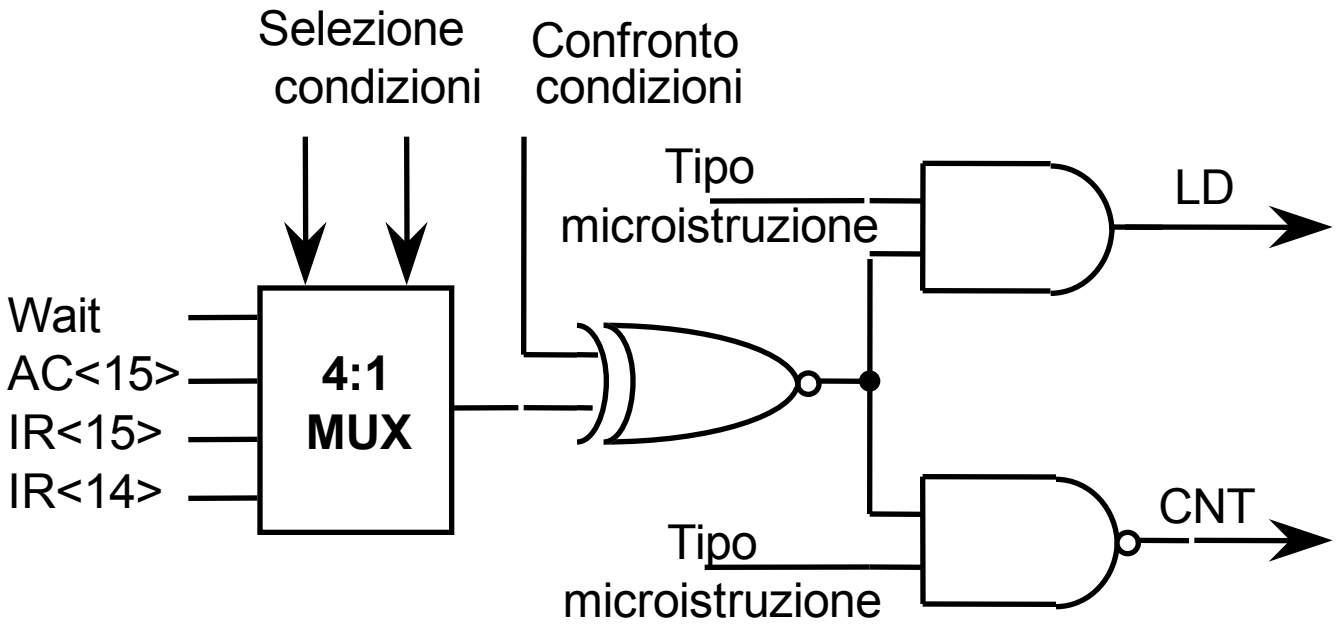
Schema a blocchi dell'unita' di controllo



Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

Logica di generazione delle condizioni



Realizzazione dell'Unita' di Controllo

Microprogrammazione verticale

Memoria di microcontrollo programmabile (Writeable Control Store)

Una parte della memoria di microcontrollo e' realizzata in RAM

Permette al programmatore in assembler di *aggiungere* nuove istruzioni alla macchina

Estende l'insieme di istruzioni "base" con istruzioni specifiche per un'applicazione

Scrittura di microcodice richiede moltissimo lavoro

Non piu' diffusa con processori RISC moderni

Rendere le istruzioni "base" facili e veloci

Scrivere funzioni a piu' alto livello come sequenze di microistruzioni "microprogrammazione in assembler"

Realizzazione dell'Unita' di Controllo

Riassunto del capitolo

- ***Organizzazione dell'Unita' di Controllo***

  Operazioni di trasferimento tra registri

  Macchine a stati di Moore e Mealy "classiche"

  Decomposizione di FSM

  Contatori programmabili

  Controllori di sequenza

  Microprogrammazione orizzontale e verticale