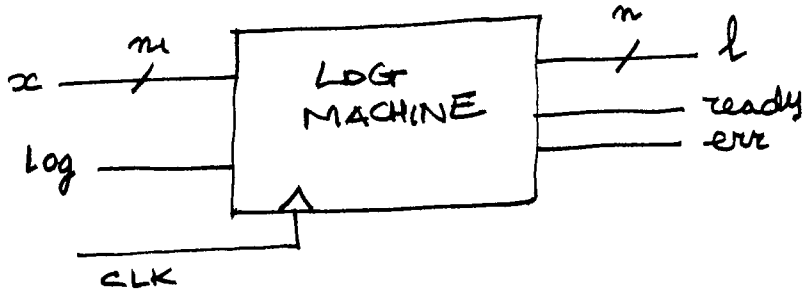
per lezione 21/10/2011
& 24/10/2011

ESERCIZIO

« Progettare col metodo Parte Operativa / Parte di Controllo una macchina sequenziale sincrona per il calcolo del logaritmo intero di un numero x in rappresentazione naturale. »

SOLUZIONE

Si suppone che il numero x, fornito in ingresso alla macchina, sia su n bit.



La macchina deve calcolare il numero l tale che

$$2^l \leq x < 2^{l+1}$$

ossia $$l = \lfloor \log_2 x \rfloor$$

Il numero di bit necessario per rappresentare l è $$m = \lceil \log_2 (n-1) \rceil$$

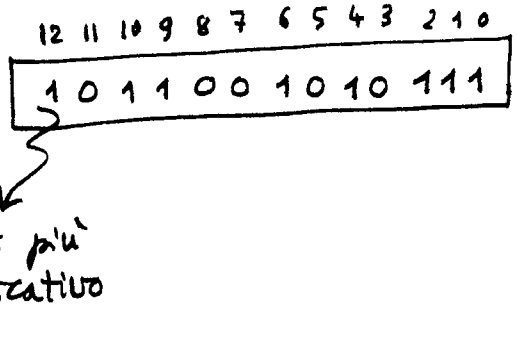
La macchina accetta in ingresso anche un segnale di inizio elaborazione (log), e fornisce in uscita un segnale di errore (err), prodotto se $x = 0$ ($\log 0 = -\infty$), ed un segnale di "ready" (= fine dell'elaborazione precedente, pronto per un nuovo numero)

Esempio. Supponiamo che x sia rappresentato su $m = 13$ bit, e che si abbia

$$2^{12} = (4096)_{10} \leq x = (1\,0\,1\,1\,0\,0\,1\,0\,1\,0\,1\,1\,1)_2 = (5719)_{10} < 2^{13}$$

In questo caso, $l = 12 = m - 1$, ed è sufficiente impiegare $\lceil \log_2 12 \rceil = 4$ bit per rappresentarlo.

In generale, la soluzione cercata corrisponde alla posizione, nella parola binaria che rappresenta x, del bit a 1 più significativo. Nel caso precedente tale posizione è la dodicesima:



Nota: Il valore di l non cambia se vengono modificati i bit meno significativi rispetto all'1 più significativo. Ad es., le parole

1 0 1 1 0 0 1 0 1 0 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1
1 x x x x x x x x x x x x

e in generale con x = "don't care"

Condividono lo stesso valore di l: 12.

Un algoritmo per il calcolo di l è il seguente:

"effettua ripetuti scorrimenti a destra della parola x finché non registri il valore $x=1$: il valore di l cercato corrisponde al numero di scorrimenti effettuati".

Ad es., nel caso precedente si avrebbe

#scorrimenti	12	11	10	9	8	7	6	5	4	3	2	1	0	valore di x
														$2^{12}+y_0$ $(y_0<2^{12}$
0	1	x	x	x	x	x	x	x	x	x	x	x	x	$2^{11}+y_1$ $(y_1<2^{1}$
1	0	1	x	x	x	x	x	x	x	x	x	x	x	$2^{10}+y_2$
2	0	0	1	x	x	x	x	x	x	x	x	x	x	$2^{9}+y_3$
3	0	0	0	1	x	x	x	x	x	x	x	x	x	$2^{8}+y_4$
4	0	0	0	0	1	x	x	x	x	x	x	x	x	$2^{7}+y_5$
5	0	0	0	0	0	1	x	x	x	x	x	x	x	$2^{6}+y_6$
6	0	0	0	0	0	0	1	x	x	x	x	x	x	$2^{5}+y_7$
7	0	0	0	0	0	0	0	1	x	x	x	x	x	$2^{4}+y_8$
8	0	0	0	0	0	0	0	0	1	x	x	x	x	$2^{3}+y_9$
9	0	0	0	0	0	0	0	0	0	1	x	x	x	$2^{2}+y_{10}$
10	0	0	0	0	0	0	0	0	0	0	1	x	x	$2^{1}+y_{11}$
11	0	0	0	0	0	0	0	0	0	0	0	1	x	$2^{0}+0=1$
12	0	0	0	0	0	0	0	0	0	0	0	0	1	

Il valore finale è il risultato di 12 successive divisioni intere (ignorando cioè il resto) per 2:

$$2^{13} > x = 2^{12} + y_0 = 2^{12}\left(2^0 + \underbrace{y_0/2^{12}}_{<1}\right) \geq 2^{12}(2^0+0)$$

<1: la divisione intera di y_0 per 2^{12} vale 0

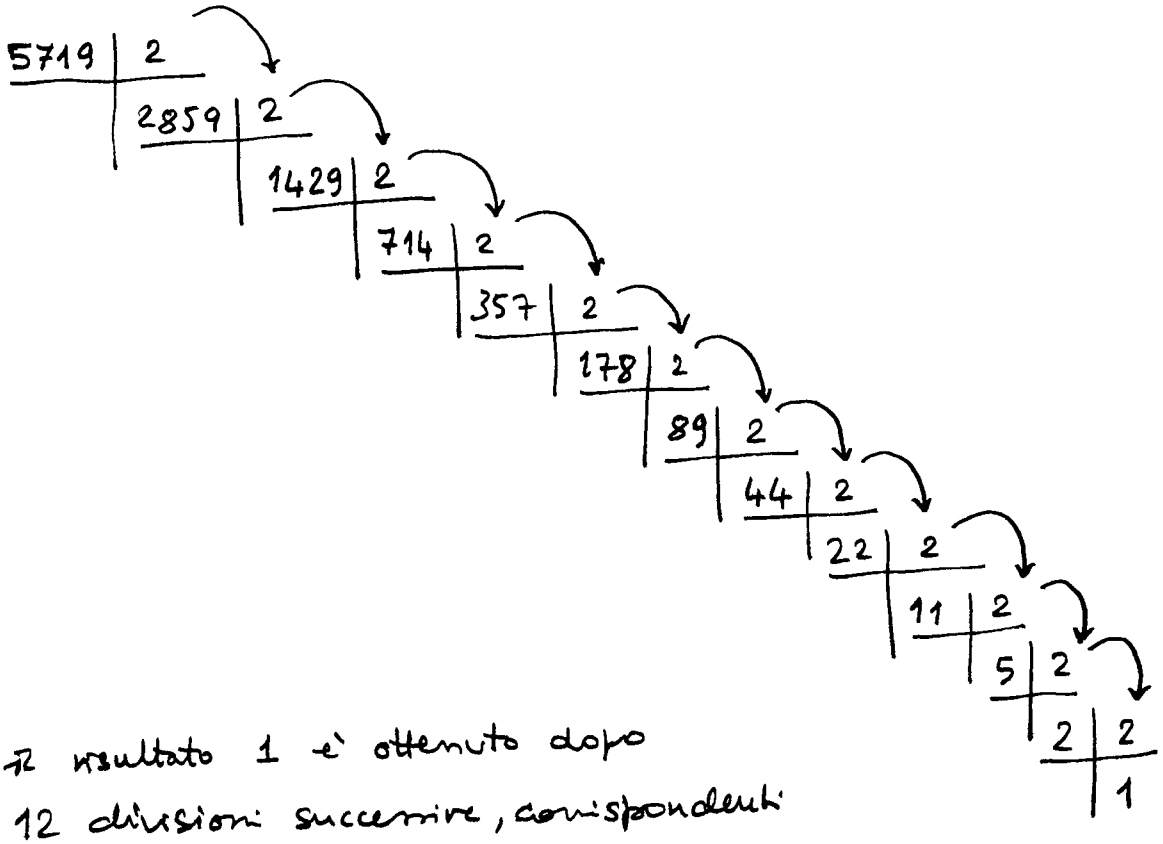
Ritorniamo all'esempio numerico usato prima:

$$x = (1\,0\,1\,1\,0\,0\,1\,0\,1\,0\,1\,1\,1)_2 =$$

$$= 2^{12} + 2^{10} + 2^9 + 2^6 + 2^4 + 2^2 + 2^1 + 2^0 =$$

$$= 4096 + 1024 + 512 + 64 + 16 + 4 + 2 + 1 =$$

$$= 5719$$

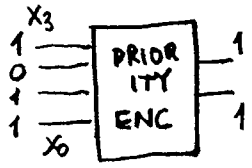


$\Rightarrow$ Il risultato 1 è ottenuto dopo 12 divisioni successive, corrispondenti a 12 scorrimenti a destra.

OSSERVAZIONE.

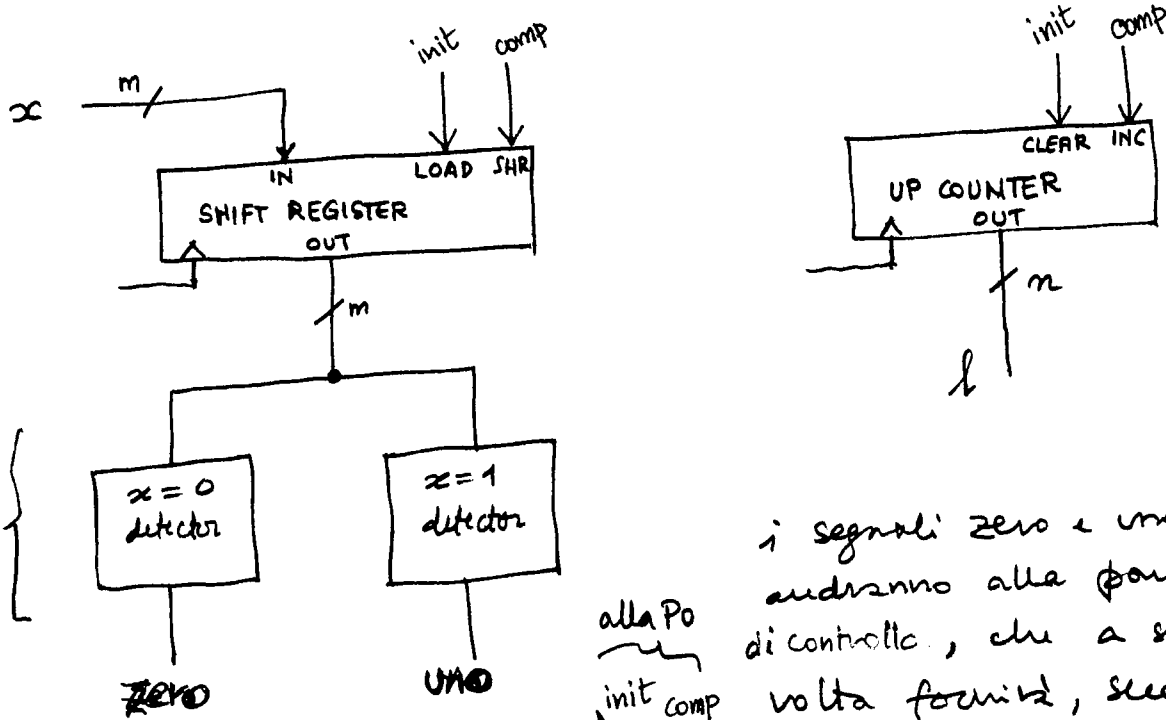
Il calcolo del $\lfloor \log_2 x \rfloor$ si può fare molto più semplicemente con una rete combinatoria! A tale scopo basta impiegare un PRIORITY ENCODER, la cui tabella nel caso $n = 4$ è la seguente

x_3	x_2	x_1	x_0	z_1	z_0	
1	X	X	X	1	1	(=3)
0	1	X	X	1	0	(=2)
0	0	1	X	0	1	(=1)
0	0	0	1	0	0	(=0)

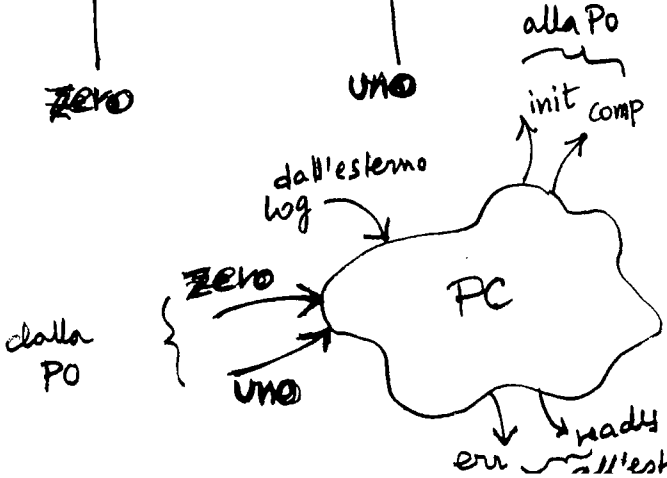


Dato l'algoritmo, una sua realizzazione può essere basata sul seguente hardware di <u>Parte Operativa</u>:

- un registro a scorrimento, che va inizialmente caricato col valore di x
- un contatore, inizializzato a 0, che tiene traccia del numero di scorrimenti a destra effettuati
- un circuito combinatorio per il rilevamento del raggiungimento della condizione di terminazione dell'algoritmo ($x=1$)
- un altro circuito combinatorio per il rilevamento della condizione di errore ($x=0$)



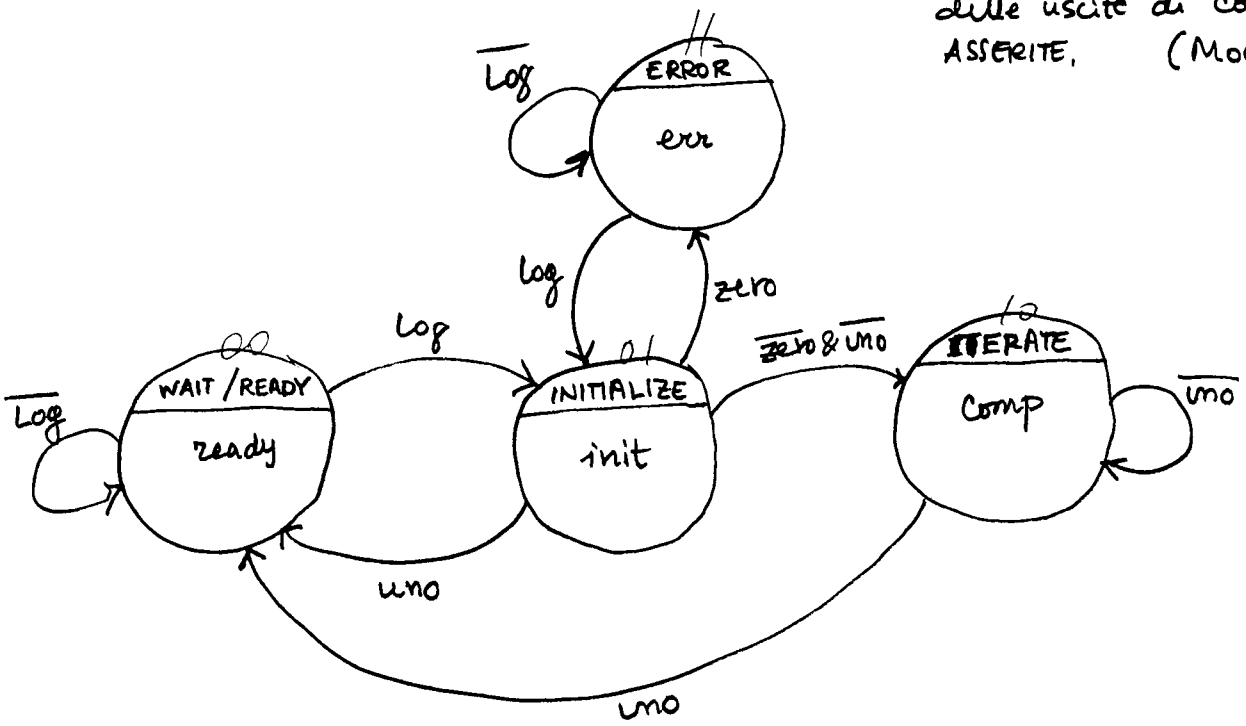
detectors: cfr. nota a pag. 15



I segnali zero e uno andranno alla parte di controllo, che a sua volta fornirà, secondo la giusta temporizzazione, i segnali load, shr, clear, inc.
Dato che load e clear sono forniti nello stato di inizializzazione, PC li asserirà simultaneamente col segnale init. Simile per shr, inc: segnale comp.

Automa di controllo

NOTA: L'automa riporta, in ogni stato, solo i nomi delle uscite di controllo ASSERITE. (Moore).



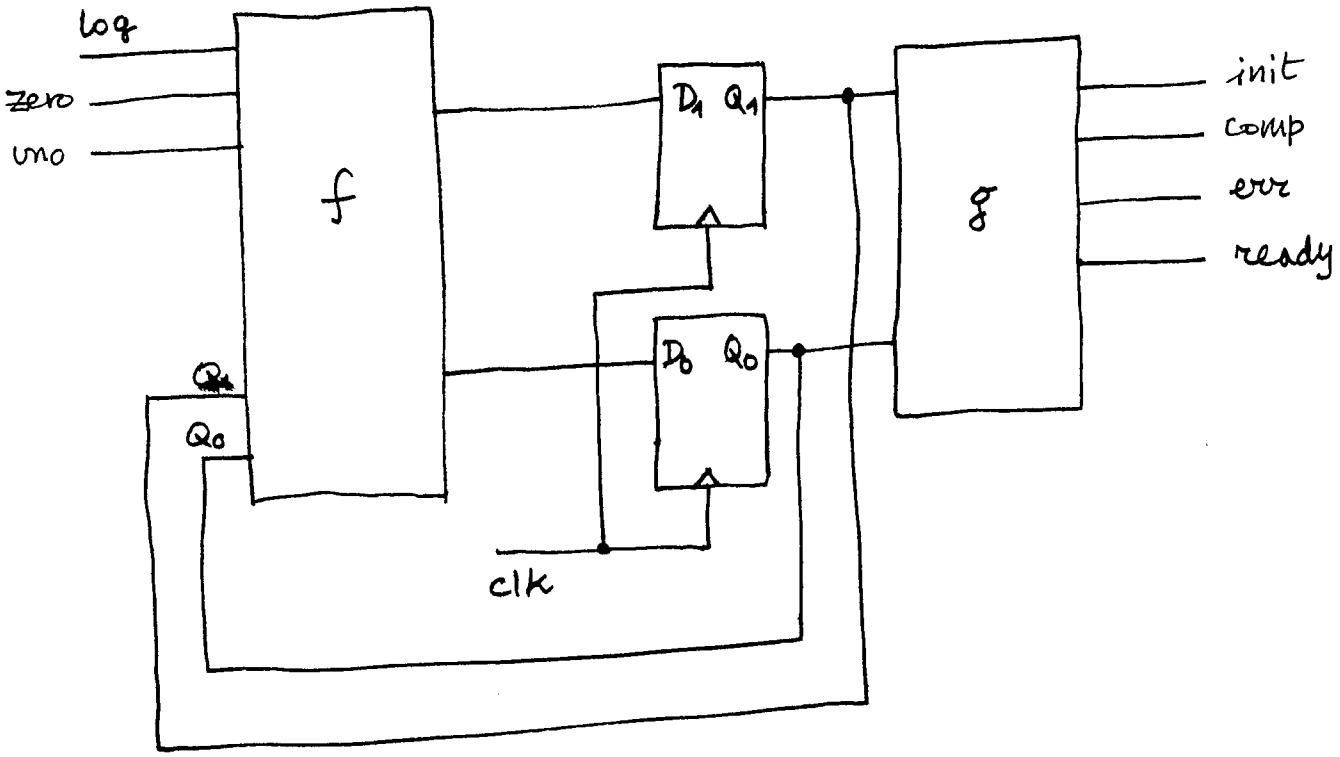
La realizzazione dell'automa di controllo seguirà il procedimento di sintesi "monoblocco".

Anzitutto, poiché gli stati di controllo sono 4, serviranno $\lceil \log_2 4 \rceil = 2$ Flip-flop di tipo D, che costituiranno il registro di stato. Per procedere con la sintesi, bisogna codificare gli stati di controllo in binario. Ad es.:

STATO DI CONTROLLO	CODIFICA BINARIA (Q_1, Q_0)	INGRESSI CAMPIONATI NELLO STATO
WAIT / READY	0 0	log
INITIALIZE	0 1	zero, uno
ITERATE	1 0	uno
ERROR	1 1	log

N.B.: (mutuamente esclusivi)

Lo schema a blocchi del controllo (real. Moore) è il seguente:

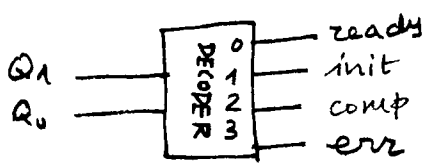


Funzione g

Potremmo progettarla scrivendo la tabella

Q_1 Q_0	init	comp	err	ready
0 0	0	0	0	1
0 1	1	0	0	0
1 0	0	1	0	0
1 1	0	0	1	0

ma ci accorgiamo che, poiché ogni uscita viene erogata esclusivamente in uno stato, allora la funzione g richiesta è la semplice decodifica dello stato:



Funzione f

Il metodo "automatico" per costruirla è quello di stendere la tabella di transizione degli stati:

Log	Zero	Uno	Q_1	Q_0	Q'_1	Q'_0
0	0	0	0	0	0	0
0	0	0	0	1	1	0
0	0	0	1	0	1	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	0	1	1	1
0	1	0	1	0	1	0
0	1	0	1	1	1	1
0	1	1	0	0	X	X
0	1	1	0	1	X	X
0	1	1	1	0	X	X
0	1	1	1	1	X	X
1	0	0	0	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	0	1
1	1	0	0	1	1	1
1	1	0	1	0	1	0
1	1	0	1	1	0	1
1	1	1	0	0	X	X
1	1	1	0	1	X	X
1	1	1	1	0	X	X
1	1	1	1	1	X	X

NOTA: Poiché Zero e Uno sono mutuamente esclusivi, le configurazioni di ingresso dove essi valgono simultaneamente 1 sono impossibili, e dunque le righe della tabella in cui ciò è previsto danno luogo a valori "don't care" delle funzioni da realizzare. Tali valori potranno essere scelti arbitrariamente a 1 o a 0, per semplificare le espressioni delle funzioni Q'_1 e Q'_0 da realizzare.

Per la realizzazione di $Q_1' = Q_0'$ (che secondo la tabella dipendono da 5 ingressi), useremo Karnaugh, dividendo il problema in questo modo

$$Q_i' = \overline{log}\, Q_i'\big|_{log=0} + log\, Q_i'\big|_{log=1}$$

ciascuna di queste due funzioni dipende solo dai 4 ingressi zero, uno, Q_1, Q_0, ed è realizzabile con una mappa di Karnaugh quadrata.

zero uno \ Q_1Q_0	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	X	X	X	X
10	0	1	1	1

$$Q_1'\big|_{log=0} = Q_1Q_0 + \overline{uno}(Q_1 \oplus Q_0)$$

zero uno \ Q_1Q_0	00	01	11	10
00	0	1	0	1
01	0	0	0	0
11	X	X	X	X
10	0	1	0	1

$$Q_1'\big|_{log=1} = \overline{uno}\,\overline{Q}_1Q_0 + \overline{uno}\,Q_1\overline{Q}_0 = \overline{uno}(Q_1 \oplus Q_0)$$

zero uno \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	X	X	X	X
10	0	1	1	0

$$Q_0'\big|_{log=0} = Q_1Q_0 + zero\,Q_0 = (Q_1 + zero)\,Q_0$$

zero uno \ Q_1Q_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	1	1	0

$$Q_0'\big|_{log=1} = \overline{Q}_1\overline{Q}_0 + Q_1Q_0 + zero\,Q_0 = (\overline{Q_1 \oplus Q_0}) + zero\,Q_0$$

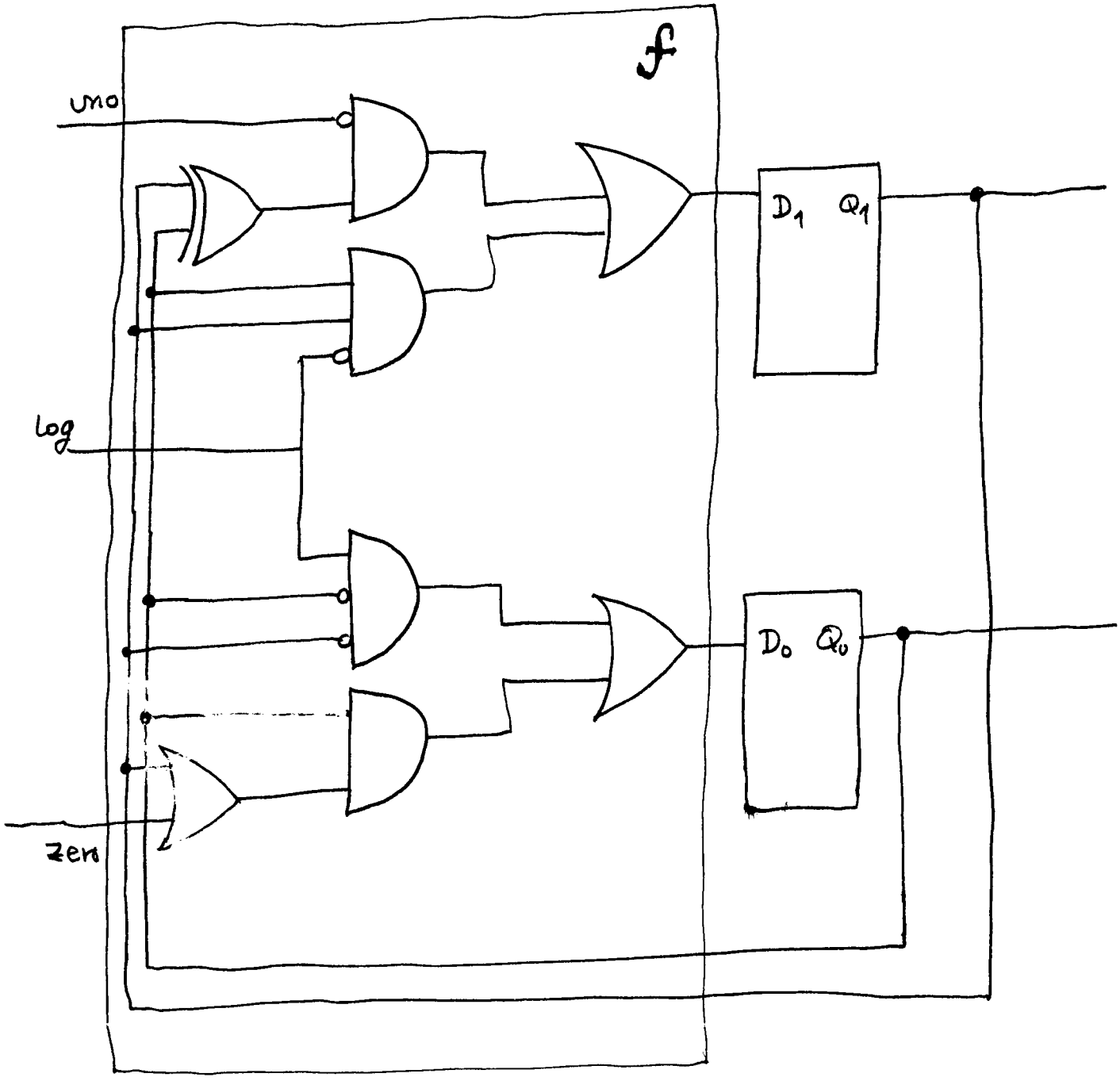
Segue

$$Q_1' = \overline{log}\left[Q_1Q_0 + \overline{uno}\,(Q_1 \oplus Q_0)\right] + log\left[\overline{uno}\,(Q_1 \oplus Q_0)\right] =$$

$$= \overline{log}\,Q_1Q_0 + \overline{uno}\,(Q_1 \oplus Q_0)$$

$$Q_0' = \overline{log}\left[Q_1Q_0 + zero\,Q_0\right] + log\left[\bar{Q}_1\bar{Q}_0 + Q_1Q_0 + zero\,Q_0\right] =$$

$$= log\,\bar{Q}_1\bar{Q}_0 + (Q_1 + zero)Q_0$$



Un metodo alternativo (e qui più efficace) per sintetizzare $f(Log, zero, uno; Q_1, Q_0)$ deriva dall'osservazione che gli ingressi esterni (Log, zero, uno) non sono campionati tutti insieme in nessuno stato. Quindi, si può pensare di risolvere il problema stato per stato, studiando quali sono gli ingressi effettivamente campionati in ciascuno stato (si veda la tab. a pag. 5).

Costruiamo la tabella di transizione di stato nel modo seguente:

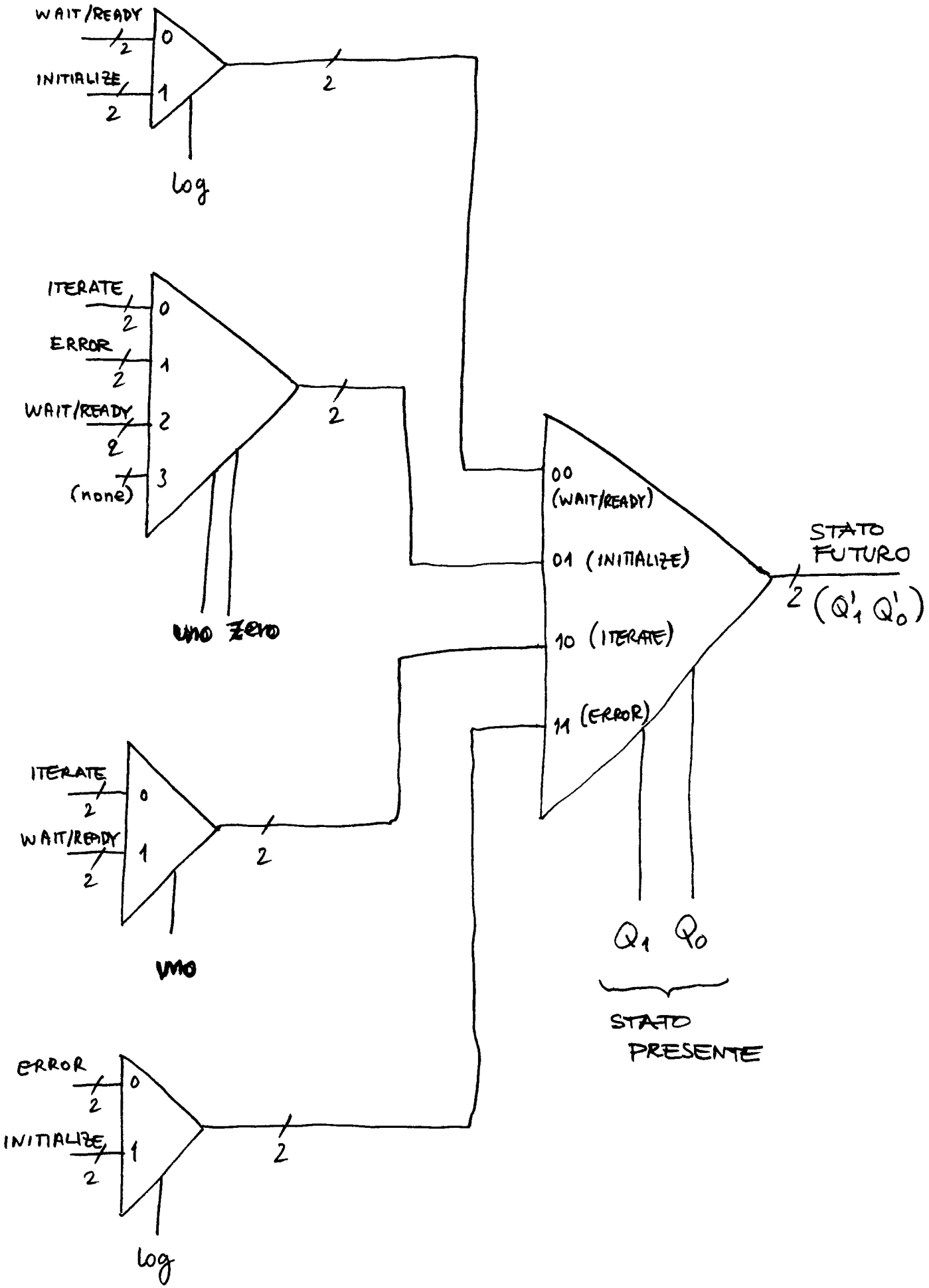
STATO PRESENTE	STATO FUTURO
WAIT/READY	$\overline{Log}$ (WAIT/READY) + Log (INITIALIZE)
INITIALIZE	uno (WAIT/READY) + zero (ERROR) + $\overline{uno}.\overline{zero}$ (ITERATE)
ITERATE	$\overline{uno}$ (ITERATE) + uno (WAIT/READY)
ERROR	$\overline{Log}$ (ERROR) + Log (INITIALIZE)

Notiamo altresì che, poiché uno e zero sono mutuamente esclusivi, si ha che $uno = 1 \Rightarrow uno.\overline{zero} = 1$, e anche $zero = 1 \Rightarrow \overline{uno}\,zero = 1$. Quindi possiamo riscrivere la seconda riga dello stato futuro come

$\overline{uno}.\overline{zero}$ (ITERATE) + $\overline{uno}.zero$ (ERROR) + $uno.\overline{zero}$ (WAIT/READY).

Dunque, lo stato futuro può essere generato dallo stato presente mediante multiplexer del modo seguente:

(schema simbolico)



In sostanza, la funzione di 5 ingressi da realizzare è stata decomposta nella combinazione "multiplexata" di 4 funzioni più semplici, ciascuna corrispondente a una data coppia di variabili di stato presenti:

$$f(log, uno, zero; Q_1, Q_0) = \bar{Q}_1\bar{Q}_0 f_{00}(log, uno, zero) + \bar{Q}_1 Q_0 f_{01}(log, uno, zero) + Q_1\bar{Q}_0 f_{10}(log, uno, zero) + Q_1 Q_0 f_{11}(log, uno, zero)$$

Ora: dall'analisi delle caratteristiche delle funzioni f da realizzare, è emerso che, in realtà,

- f_{00} dipende solo da log, e non da uno e zero;
- f_{01} " " " uno e zero, " " " log;
- f_{10} " " " uno, " " " log e zero;
- f_{11} " " " log, " " " uno e zero

Ci siamo dunque ricondotti a dover realizzare delle funzioni combinatorie che sono funzioni solo di **1** o 2 ingressi, e quindi sono molto più agevoli da sintetizzare.

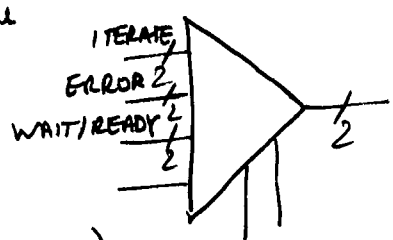
A pag. 11 è mostrata la realizzazione delle funzioni $f_{00}(log)$, $f_{01}(uno, zero)$, $f_{10}(uno)$, $f_{11}(log)$ mediante multiplexer. Naturalmente, queste semplici funzioni si possono realizzare anche col metodo classico della tabella di verità. Vediamo per esempio come realizzare la funzione più "complicata" delle quattro, cioè f_{01}, che dipende da 2 ingressi.

uno	zero	Q'_1	Q'_0	$(f_{01}(uno, zero))$
0	0	1	0	(ITERATE)
0	1	1	1	(ERROR)
1	0	0	0	(WAIT/READY)
1	1	X ↓ 0	X ↓ 1	(mutua esclusione ingressi uno e zero)

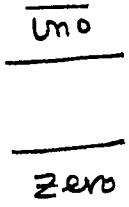
Scegliendo (come indicato sopra) i don't care rispettivamente a 0 e 1, abbiamo subito che

$$Q'_1 = \overline{uno}, \quad \text{mentre} \quad Q'_0 = zero$$

Dunque



$\Longleftrightarrow$



: un bel risparmio!

(cfr. pag. 11)

Vediamo anche f_{00}, f_{10}, f_{11} :

log	Q'_1 Q'_0 $(f_{00}(log))$
0	0 0
1	0 1

$\Rightarrow$ $Q'_1 = 0$, $Q'_0 = log$

uno	Q'_1 Q'_0 $(f_{10}(uno))$
0	1 0
1	0 0

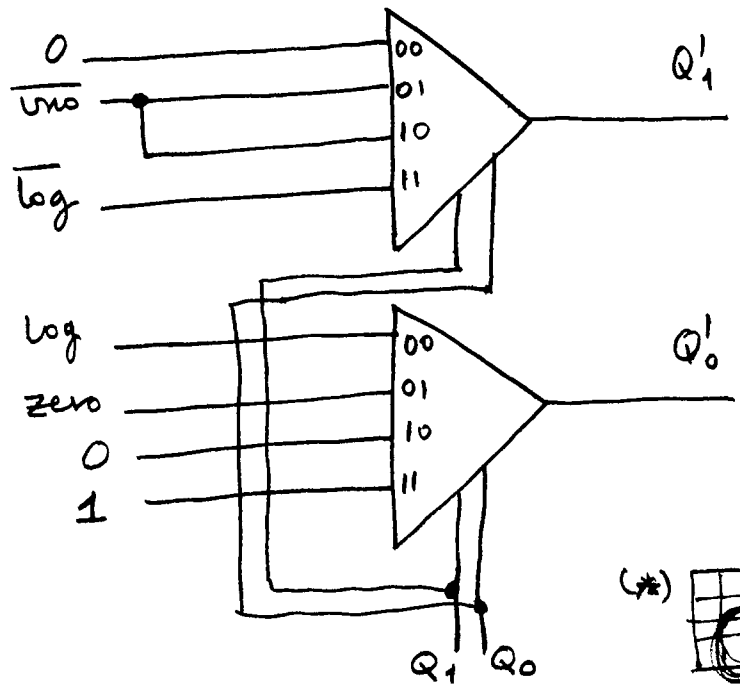
$\Rightarrow$ $Q'_1 = \overline{uno}$, $Q'_0 = 0$

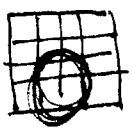
log	Q'_1 Q'_0 $(f_{11}(log))$
0	1 1
1	0 1

$\Rightarrow$ $Q'_1 = \overline{log}$, $Q'_0 = 1$

Per riassumere :

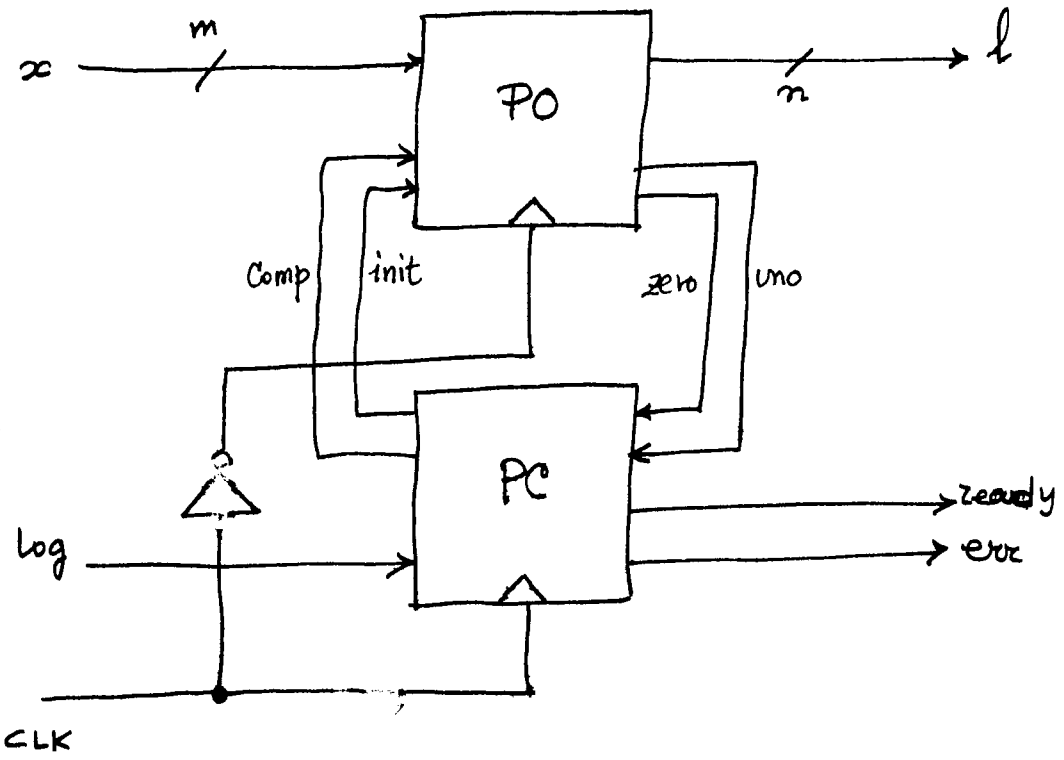
$$\begin{cases} Q'_1 = (\overline{Q}_1\overline{Q}_0 \cdot 0 +)\ \overline{Q}_1 Q_0 \cdot \overline{uno} + Q_1\overline{Q}_0 \cdot \overline{uno} + Q_1 Q_0 \cdot \overline{log} \\ Q'_0 = \overline{Q}_1\overline{Q}_0 \cdot log + \overline{Q}_1 Q_0 \cdot zero (+\ Q_1\overline{Q}_0 \cdot 0) + Q_1 Q_0 \cdot 1 \end{cases}$$



(*) 

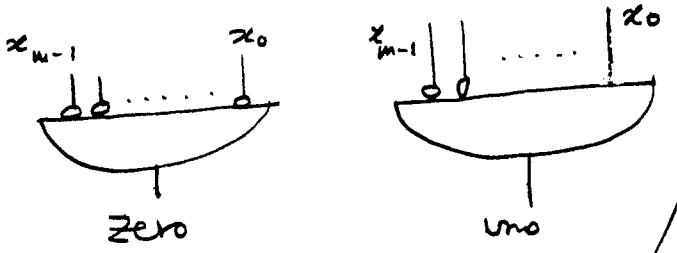
NOTA: le funzioni ottenute sono identiche a quelle trovate con la tabella a 32 righe, salvo che lì si è avuto, nell'espressione di Q'_0, il prodotto $zero\,Q_0$ anziché $zero\,\overline{Q}_1 Q_0$. Ciò è dovuto alla (*) particolare scelta fatta dei valori dei don't care. La scelta "  " avrebbe dato $zero\,Q_1 Q_0$.

Disegno della macchina completa:

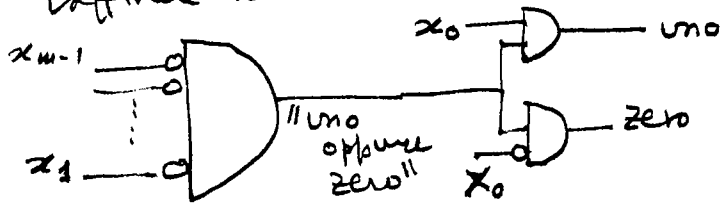


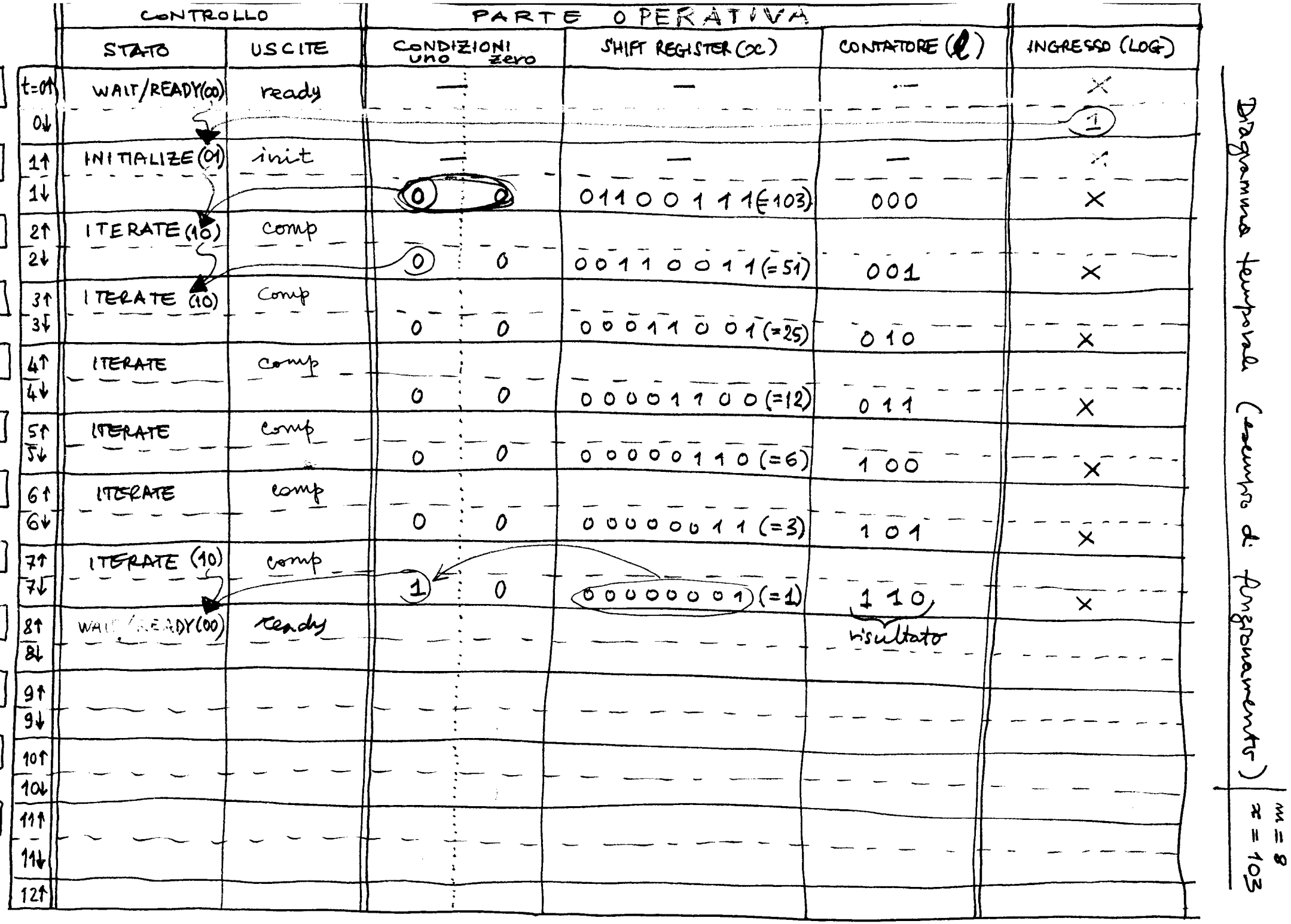
Nota sui detector di $x=0$ e $x=1$:

Si possono fare con porte AND, che operano come riconoscitori di configurazioni binarie (si ricordino gli AND all'interno di un decoder, cascano specializzato a riconoscere un particolare mintermine):



Un metodo alternativo per riconoscere queste due parole CONTIGUE nella rappresentazione (cfr. progetto di interfacce I/O: riconoscitori di indirizzi di porta) consiste nell'avere una logica che riconosca l'occorrenza di una qualsiasi delle due parole, e poi raffinare il riconoscimento usando il bit LSB che non si è usato prima:




CONTROLLO
PARTE OPERATIVA
STATO
USCITE
CONDIZIONI
uno
zero
SHIFT REGISTER (sc)
CONTATORE (ℓ)
INGRESSO (LOG)
t=0↑
WAIT/READY(00)
ready
0↓
1
1↑
INITIALIZE (01)
init
1↓
0
0
01100111 (=103)
000
2↑
ITERATE (10)
comp
2↓
0
0
00110011 (=51)
001
3↑
ITERATE (10)
comp
3↓
0
0
00011001 (=25)
010
4↑
ITERATE
comp
4↓
0
0
00001100 (=12)
011
5↑
ITERATE
comp
5↓
0
0
00000110 (=6)
100
6↑
ITERATE
comp
6↓
0
0
00000011 (=3)
101
7↑
ITERATE (10)
comp
7↓
1
0
00000001 (=1)
110
risultato
8↑
WAIT/READY(00)
ready
8↓
9↑
9↓
10↑
10↓
11↑
11↓
12↑
Diagramma temporale (esempio di funzionamento)
m = 8
n = 103
16