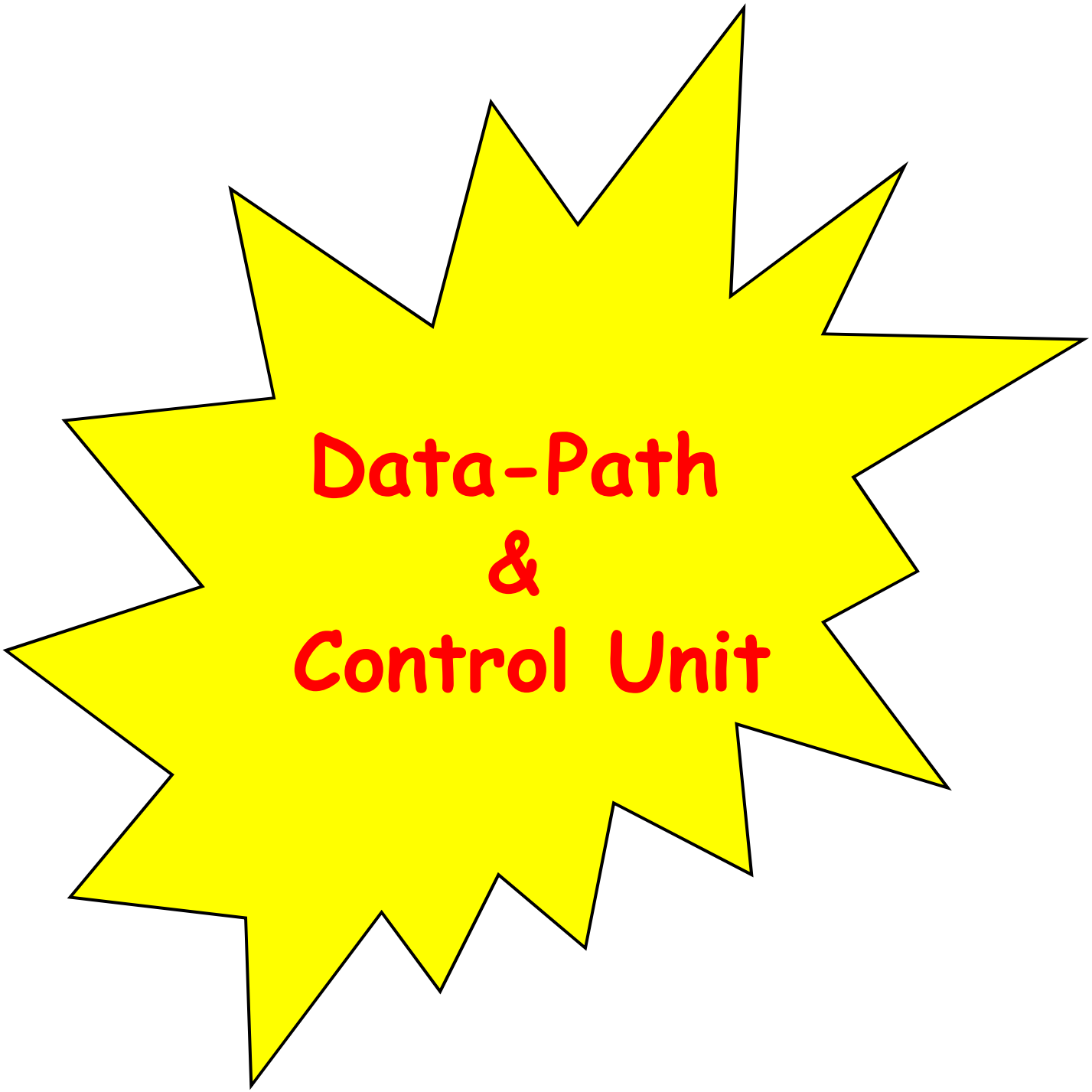
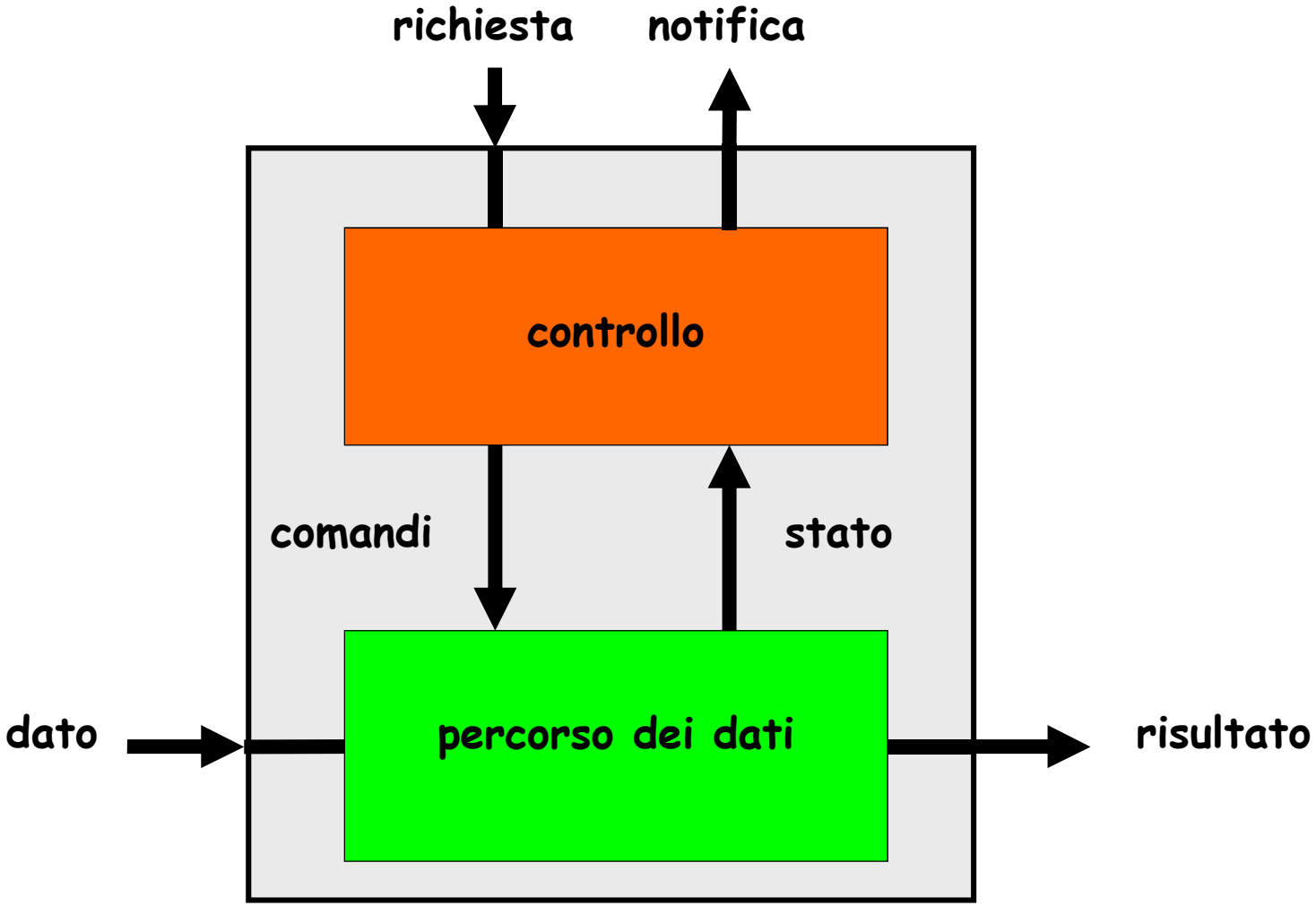

Data-Path
&
Control Unit

Modello di riferimento

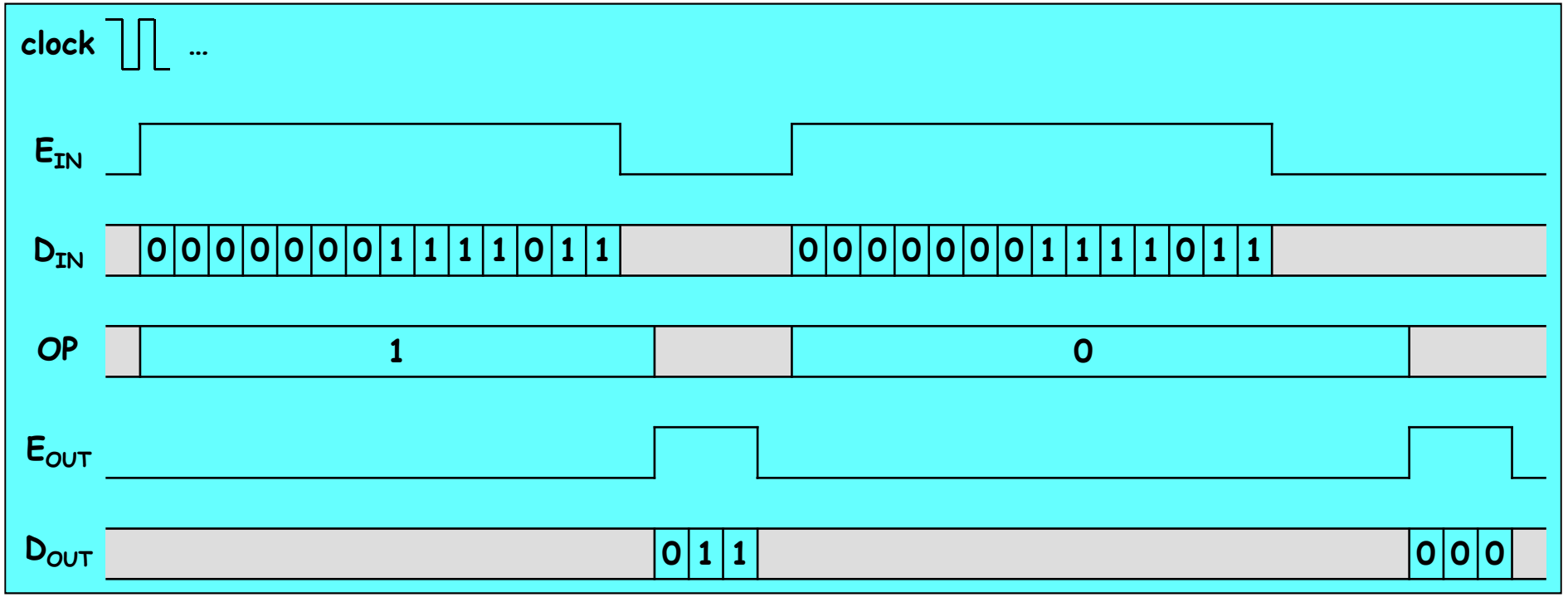


Problema 1

Un sistema sequenziale sincrono è caratterizzato da tre segnali di ingresso (E_{IN}, D_{IN}, OP) e da due segnali di uscita (E_{OUT}, D_{OUT}), tutti sincroni.

Attraverso l'ingresso D_{IN} il sistema riceve serialmente e a partire dal bit più significativo (MSB) numeri decimali di k = 4 cifre rappresentati secondo il sistema di numerazione binario mediante $h = \lceil \log_2(10^k) \rceil = 14$ bit. Il segnale E_{IN}, attivo (livello logico 1) per h periodi di clock, identifica l'intervallo di trasferimento in ingresso al sistema di ciascun numero N.

Compito del sistema è calcolare e generare in uscita il valore $N_{mod\ 5}$ o $N_{mod\ 3}$, in dipendenza del valore 1 o 0, rispettivamente, assunto dal segnale di ingresso OP. Ciascun risultato, rappresentato secondo il sistema di numerazione binario mediante m = 3 bit, deve essere trasferito in serie e a partire dal MSB attraverso l'uscita D_{OUT}. L'intervallo di generazione di ogni risultato deve essere evidenziato con l'attivazione del segnale E_{OUT} per m periodi di clock.



Il sistema deve essere strutturato in accordo al modello data-path & control unit. Il data-path comprende:

- ✧ una unità (U_1) preposta alla conversione serie/parallelo e transcodifica binario/BCD dei dati di ingresso;
- ✧ una unità (U_2) preposta alla elaborazione e generazione dei risultati di uscita.

L'unità U_1 consiste di k = 4 moduli identici M_3, M_2, M_1, M_0, ciascuno dei quali, al termine di ogni intervallo di attivazione del segnale E_{IN}, fornisce in uscita in parallelo 4 bit Q_3, Q_2, Q_1, Q_0 che codificano in BCD ($Q_3 \equiv$ MSB, $Q_0 \equiv$ LSB) una cifra decimale, rispettivamente migliaia (n_3), centinaia (n_2), decine (n_1) e unità (n_0), del numero $N \equiv (n_3\ n_2\ n_1\ n_0)$ corrispondentemente ricevuto in ingresso. Ogni modulo è caratterizzato da un ulteriore segnale di uscita C_{OUT} ($Carry_{OUT}$) e da 3 segnali di ingresso C_{IN} ($Carry_{IN}$), R (Reset), S/H' (Shift/Hold'). L'ingresso C_{IN} di M_0 è collegato a D_{IN}; l'ingresso C_{IN} di M_i (i = 1, 2, 3) è collegato all'uscita C_{OUT} di M_{i-1}. I segnali R e S/H' condizionano identicamente il funzionamento di tutti i moduli come segue:

$(Q_3\ Q_2\ Q_1\ Q_0)^{n+1} = 0000$ se $R^n = 1$

$(Q_3\ Q_2\ Q_1\ Q_0)^{n+1} = (Q_3\ Q_2\ Q_1\ Q_0)^n$ se $R^n = 0$ e $(S/H')^n = 0$

$(Q_3\ Q_2\ Q_1\ Q_0)^{n+1} = ((2 * (Q_3\ Q_2\ Q_1\ Q_0) + C_{IN})_{mod10})^n$ se $R^n = 0$ e $(S/H')^n = 1$

$C_{OUT}{}^n = ((2 * (Q_3\ Q_2\ Q_1\ Q_0) + C_{IN}) / 10)^n$

L'unità U_2 gestisce le cifre BCD di ciascun dato di ingresso rese disponibili in parallelo da U_1, calcolando il risultato R richiesto, in dipendenza del valore del segnale OP, come segue:

$OP = 1 \Rightarrow R = (n_0)_{mod5}$; $OP = 0 \Rightarrow$

```
R = 0;
for i = 0 to k – 1;
    R = (R + n_i)_mod3;
end_for;
```

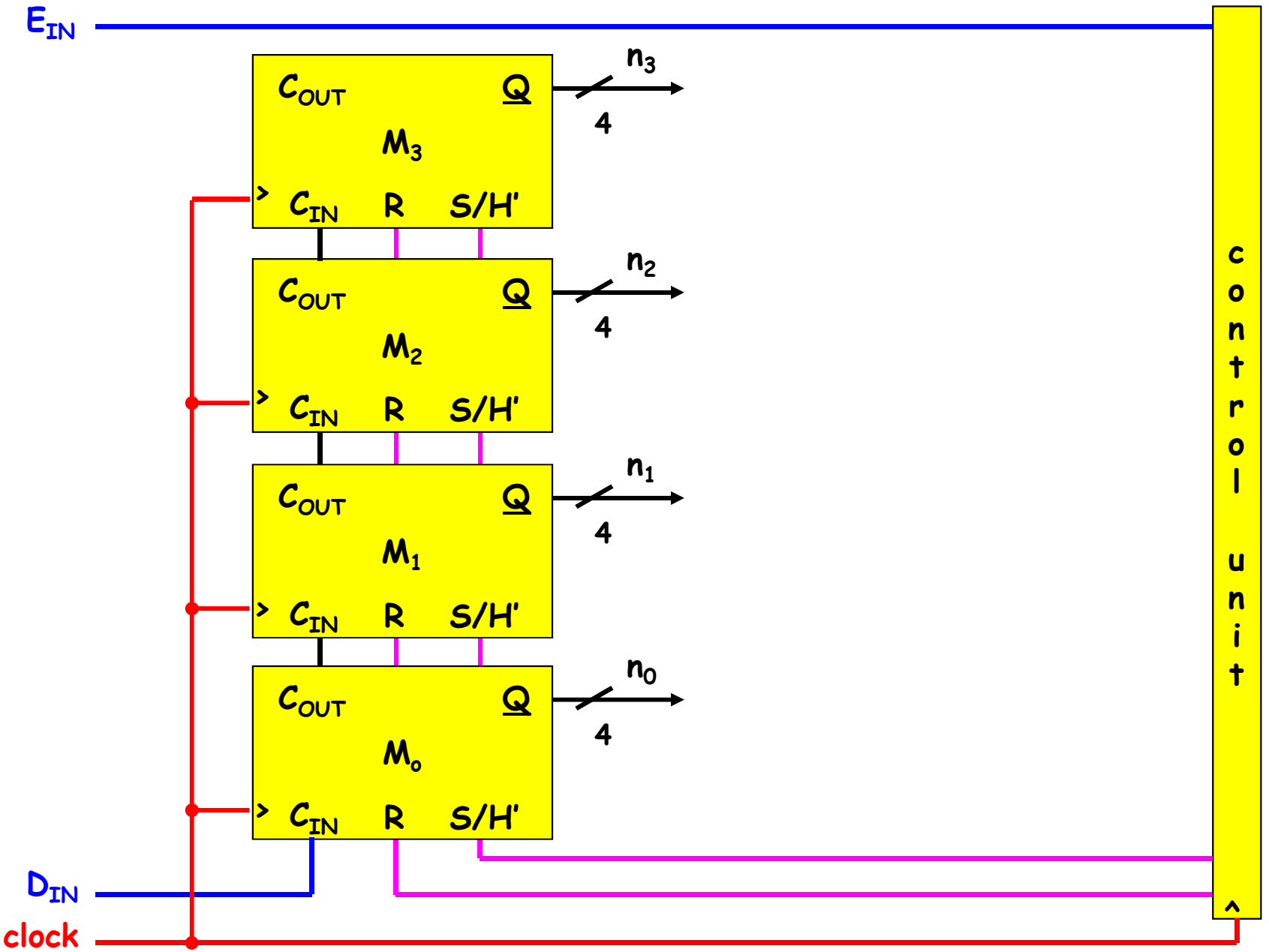
Tale unità comprende allo scopo un contatore con base di conteggio b = max (k, m) = 4 e dotato di comando di Reset (sincrono), un registro di m bit dotato di comando S/L' (Shift/Load'), un addizionatore binario a 4 bit e una rete combinatoria preposta al calcolo del modulo (5 o 3) dei risultati parziali e/o finale.

Si esegua il progetto del generico modulo dell'unità U_1.

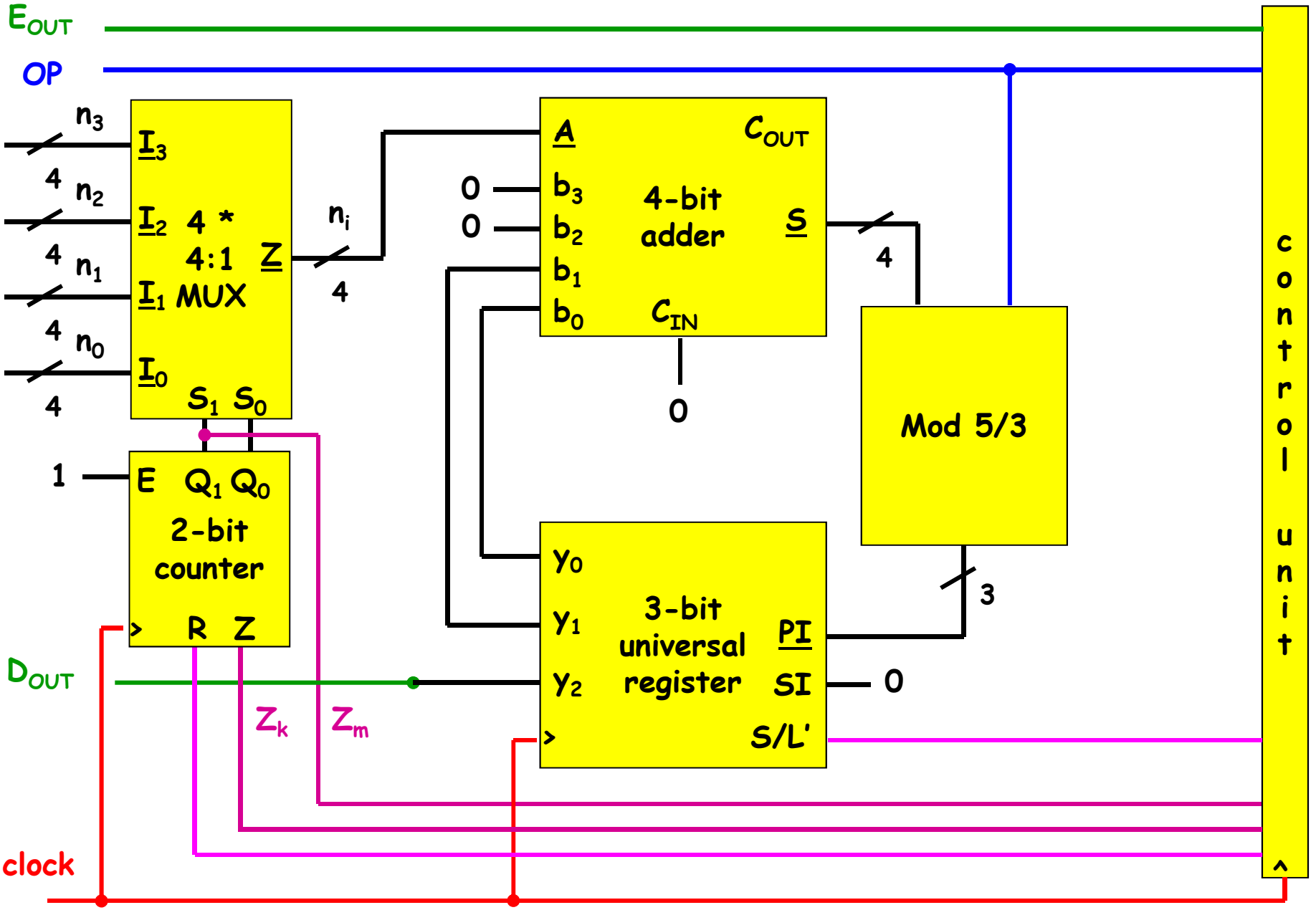
Si completi il progetto dell'unità U_2.

Si definisca il grafo degli stati dell'unità di controllo ed una sua possibile realizzazione basata su un contatore binario dotato di ingressi di Enable, Load e Reset (sincroni).

L'unità di conversione e transcodifica dei dati di ingresso



L'unità di elaborazione e generazione dei risultati di uscita



Progetto del generico modulo di conversione e transcodifica dei dati di ingresso …

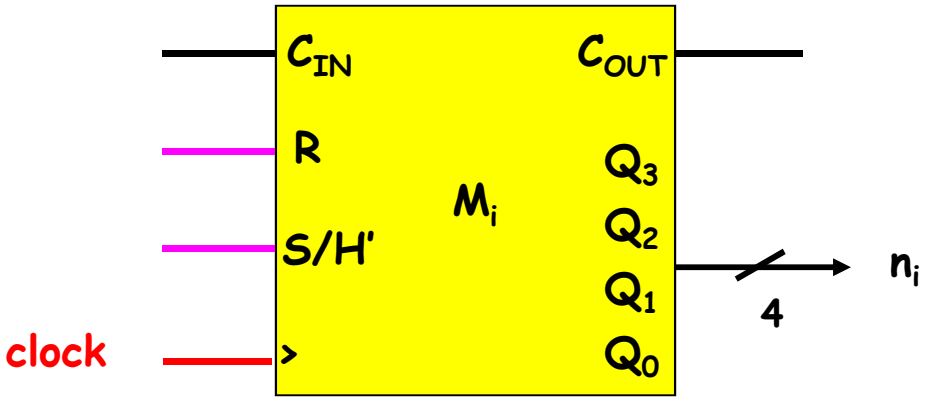
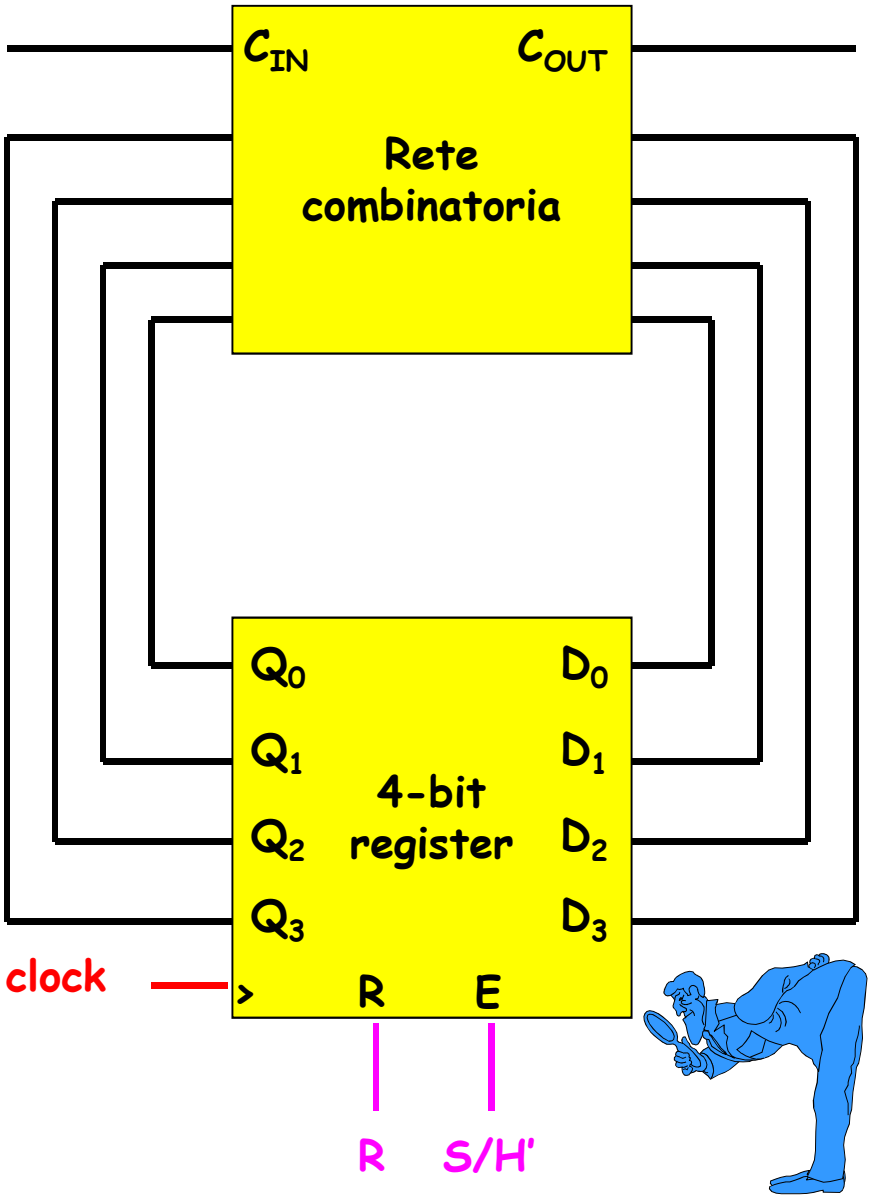


Tabella delle transizioni
relativa a
R = 0, S/H' = 1

$(C_{IN}Q_3Q_2)^n$ \ $(Q_1Q_0)^n$	00	01	11	10
000	0000,0	0010,0	0110,0	0100,0
001	1000,0	0000,1	0100,1	0010,1
011	----,-	----,-	----,-	----,-
010	0110,1	1000,1	----,-	----,-
100	0001,0	0011,0	0111,0	0101,0
101	1001,0	0001,1	0101,1	0011,1
111	----,-	----,-	----,-	----,-
110	0111,1	1001,1	----,-	----,-

$(Q_3Q_2Q_1Q_0)^{n+1}, C_{OUT}{}^n$

… Progetto del generico modulo di conversione e transcodifica dei dati di ingresso



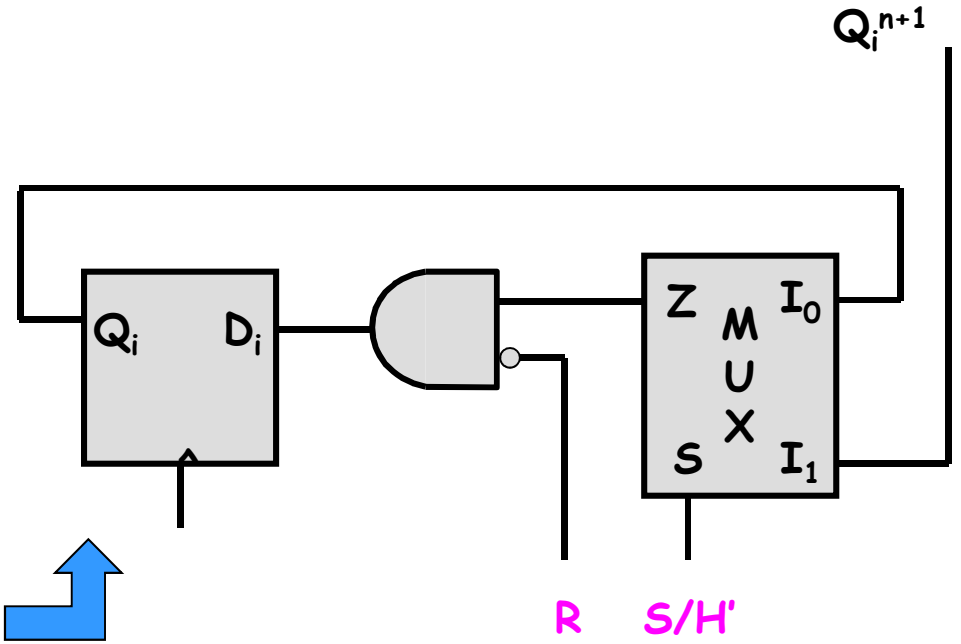
$$C_{OUT}^{n}=(Q_3+Q_2(Q_1+Q_0))^n$$

$$Q_3^{n+1}=(Q_2Q_1'Q_0'+Q_3Q_0)^n$$

$$Q_2^{n+1}=(Q_3Q_0'+Q_2'Q_1+Q_1Q_0)^n$$

$$Q_1^{n+1}=(Q_3Q_0'+Q_3'Q_2'Q_0+Q_2Q_1Q_0')^n$$

$$Q_0^{n+1}=C_{IN}^{n}$$



Progetto dell'unità di elaborazione e generazione dei risultati di uscita

OP S_3 S_2 \ S_1 S_0	00	01	11	10
000	000	001	000	010
001	001	010	001	000
011	---	---	---	---
010	010	000	010	001
100	000	001	011	010
101	100	000	010	001
111	---	---	---	---
110	011	100	---	---

PI_2 PI_1 PI_0

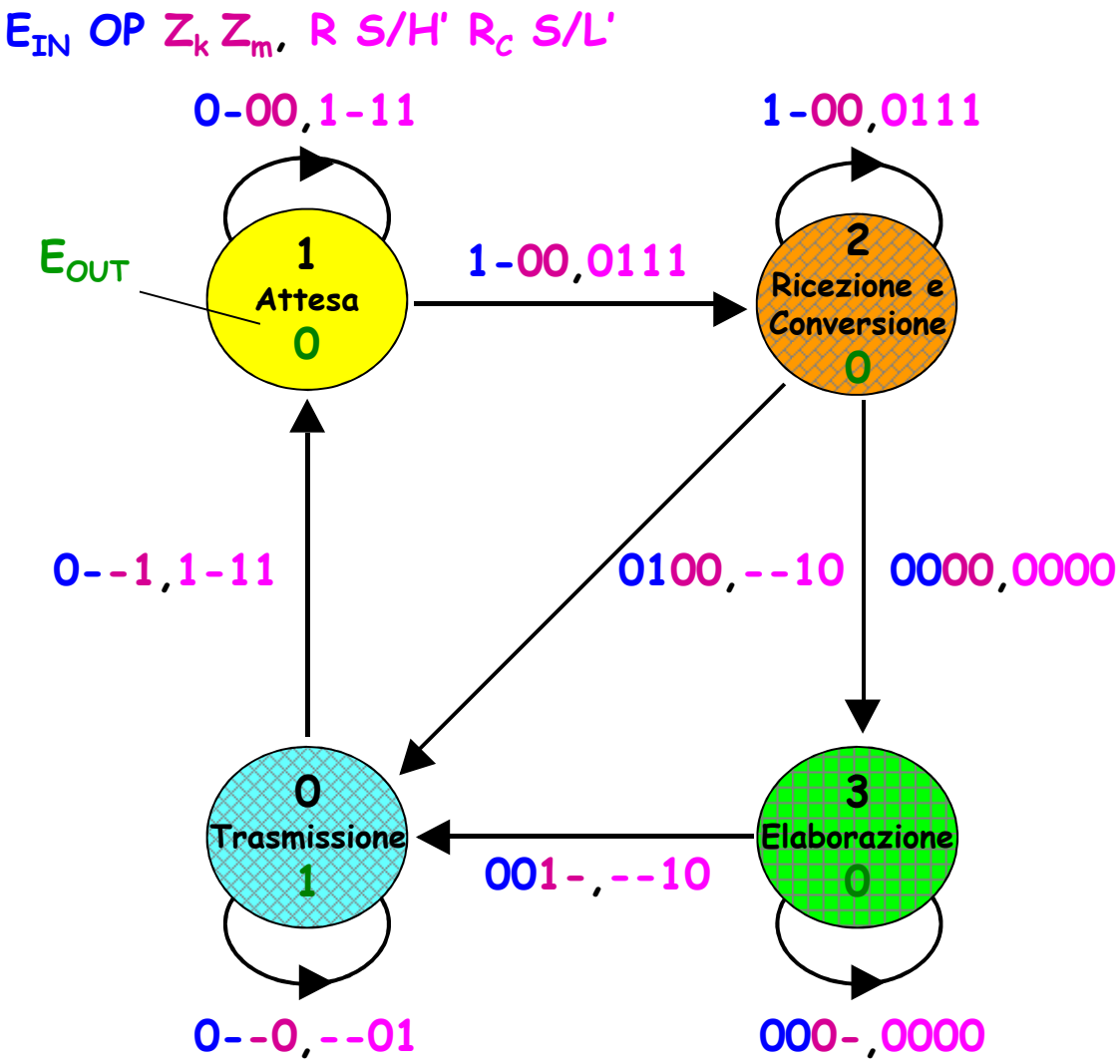
$$PI_2 = OP\ (S_2S_1'S_0' + S_3S_0)$$

$$PI_1 = OP\ S_1S_0 + OP'\ S_2S_1'S_0 + S_3S_1S_0 + S_3S_1'S_0' + S_3'S_2'S_1S_0'$$

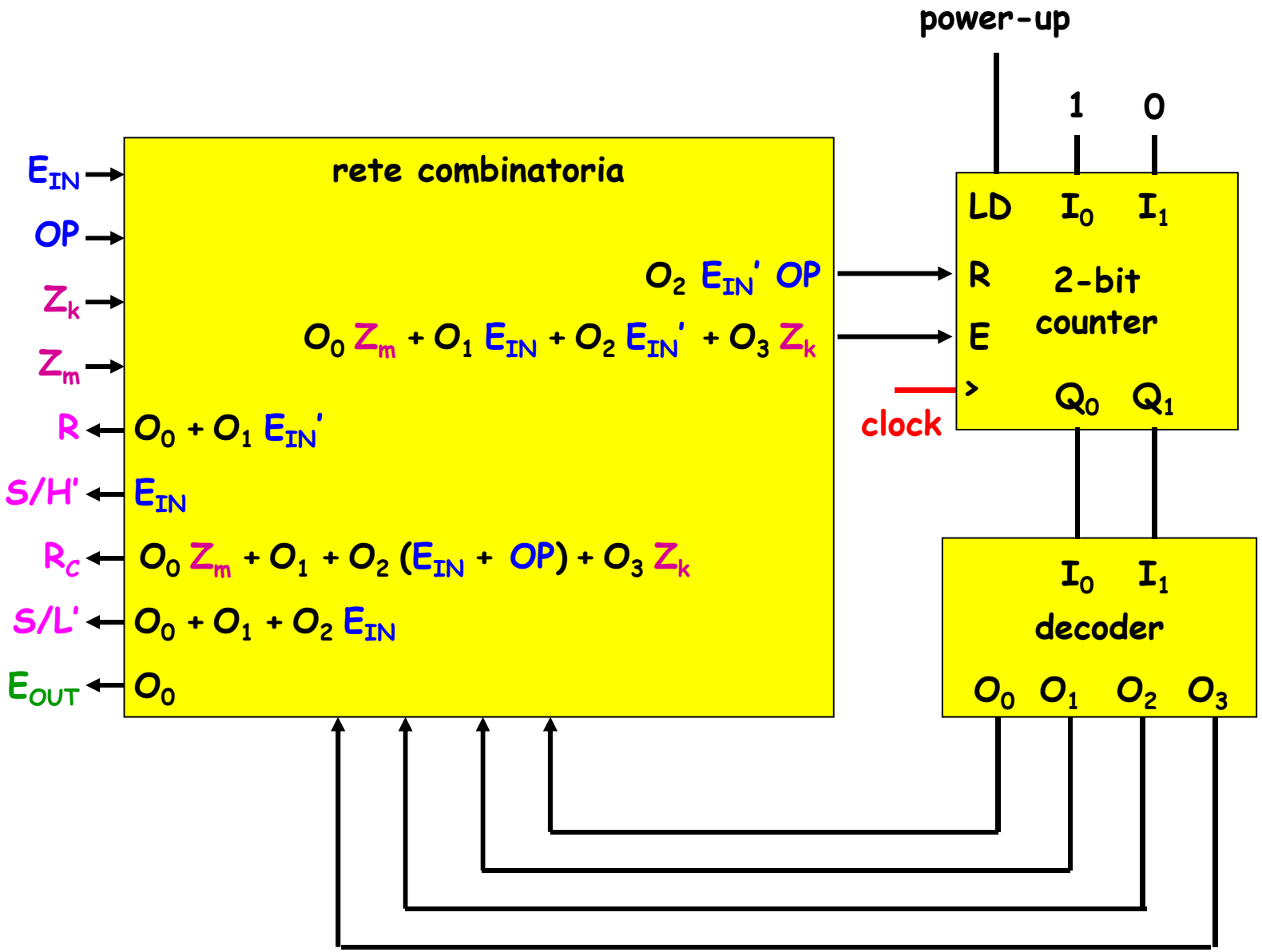
$$PI_0 = OP\ (S_3S_0' + S_2'S_1S_0 + S_2S_1S_0') + OP'\ (S_2S_1'S_0' + S_2S_1S_0) + S_3S_1S_0' + S_3'S_2'S_1'S_0$$

Progetto dell'unità di controllo …

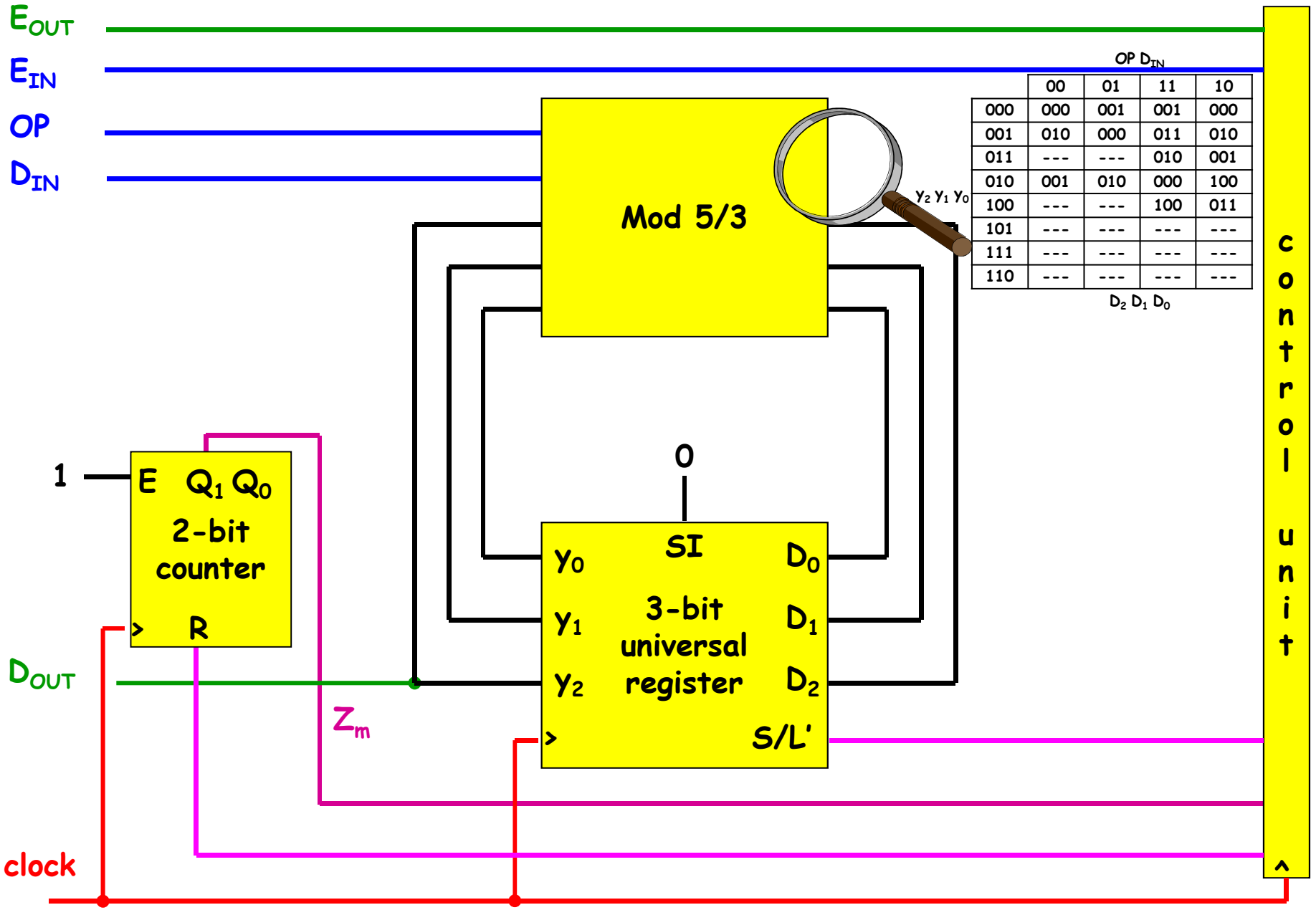
Grafo degli stati



… Progetto dell'unità di controllo



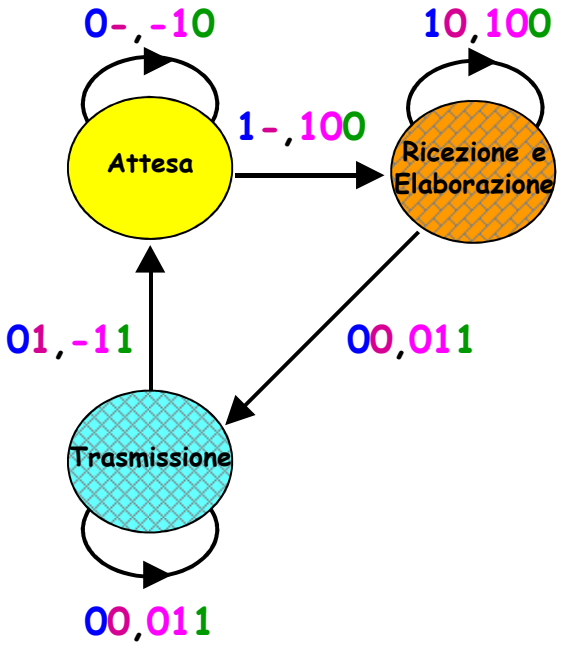
Una possibile soluzione alternativa: calcolo "on-line" del risultato

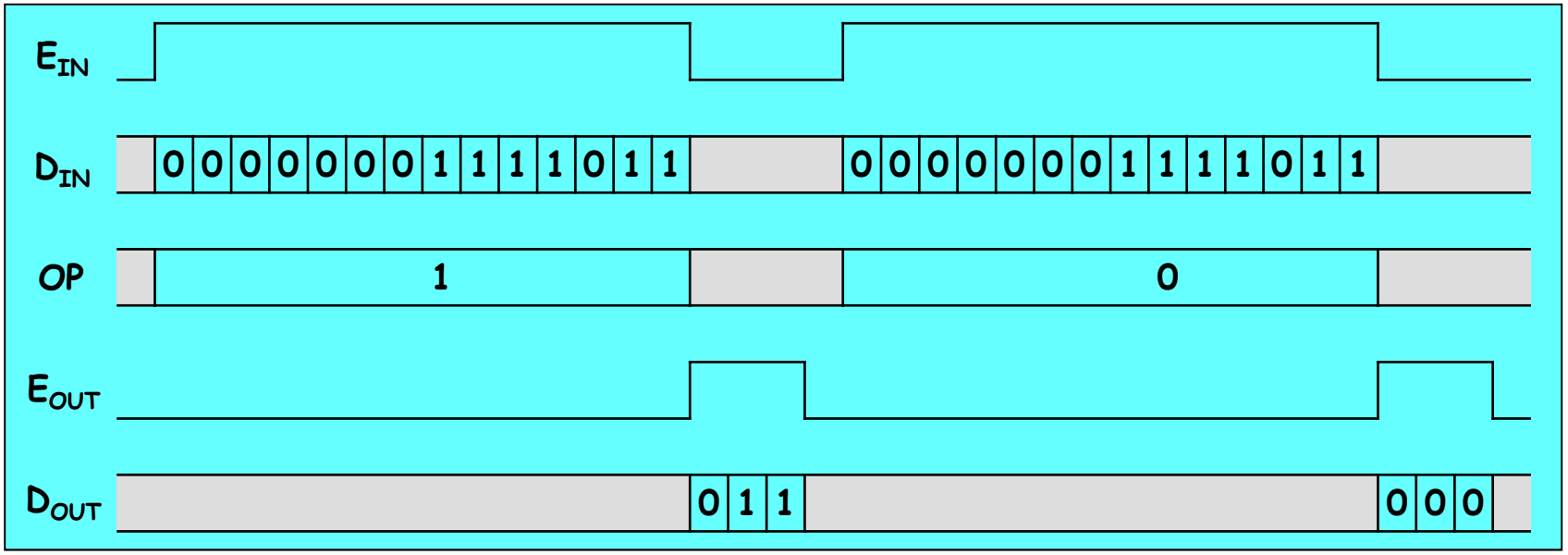


… Una possibile soluzione alternativa

Grafo degli stati dell'unità di controllo

E_{IN} Z_m, R S/L' E_{OUT}





Problema 2

Un sistema sequenziale sincrono é caratterizzato da quattro segnali di ingresso (START, I_2, I_1, I_0) e da due segnali di uscita (D_{out}, END), tutti sincroni. Il sistema, ad ogni attivazione del segnale START, ha il compito di generare serialmente in uscita uno fra i seguenti otto messaggi, in dipendenza del valore corrispondentemente assunto dagli altri ingressi I_2, I_1, I_0:

$$
\begin{aligned}
I_2\, I_1\, I_0 = 000 \quad &\Rightarrow \quad M_0 \equiv P_0\, P_1\, P_2\, P_3\, P_4\, P_5\, P_6\, P_7,\\
I_2\, I_1\, I_0 = 001 \quad &\Rightarrow \quad M_1 \equiv P_1\, P_2\, P_3\, P_4\, P_5\, P_6\, P_7,\\
I_2\, I_1\, I_0 = 010 \quad &\Rightarrow \quad M_2 \equiv P_2\, P_3\, P_4\, P_5\, P_6\, P_7,\\
I_2\, I_1\, I_0 = 011 \quad &\Rightarrow \quad M_3 \equiv P_3\, P_4\, P_5\, P_6\, P_7,\\
I_2\, I_1\, I_0 = 100 \quad &\Rightarrow \quad M_4 \equiv P_4\, P_5\, P_6\, P_7,\\
I_2\, I_1\, I_0 = 101 \quad &\Rightarrow \quad M_5 \equiv P_5\, P_6\, P_7,\\
I_2\, I_1\, I_0 = 110 \quad &\Rightarrow \quad M_6 \equiv P_6\, P_7,\\
I_2\, I_1\, I_0 = 111 \quad &\Rightarrow \quad M_7 \equiv P_7,
\end{aligned}
$$

essendo P_0, P_1, P_2, P_3, P_4, P_5, P_6 e P_7 parole di 8 bit codificate come segue:

$$
\begin{aligned}
P_0 &\equiv (P_{07}\ P_{06}\ P_{05}\ P_{04}\ P_{03}\ P_{02}\ P_{01}\ P_{00}) = 01010101,\\
P_1 &\equiv (P_{17}\ P_{16}\ P_{15}\ P_{14}\ P_{13}\ P_{12}\ P_{11}\ P_{10}) = 10101010,\\
P_2 &\equiv (P_{27}\ P_{26}\ P_{25}\ P_{24}\ P_{23}\ P_{22}\ P_{21}\ P_{20}) = 00110011,\\
P_3 &\equiv (P_{37}\ P_{36}\ P_{35}\ P_{34}\ P_{33}\ P_{32}\ P_{31}\ P_{30}) = 11001100,\\
P_4 &\equiv (P_{47}\ P_{46}\ P_{45}\ P_{44}\ P_{43}\ P_{42}\ P_{41}\ P_{40}) = 00111100,\\
P_5 &\equiv (P_{57}\ P_{56}\ P_{55}\ P_{54}\ P_{53}\ P_{52}\ P_{51}\ P_{50}) = 11000011,\\
P_6 &\equiv (P_{67}\ P_{66}\ P_{65}\ P_{64}\ P_{63}\ P_{62}\ P_{61}\ P_{60}) = 00001111,\\
P_7 &\equiv (P_{77}\ P_{76}\ P_{75}\ P_{74}\ P_{73}\ P_{72}\ P_{71}\ P_{70}) = 11110000.
\end{aligned}
$$

Ciascun messaggio M_i (i = 0, 1, …, 7) deve essere preceduto da un prologo di sei bit e seguito da un epilogo di un bit.

I primi tre bit del prologo I_{i2}, I_{i1}, I_{i0} coincidono ordinatamente con il valore dei tre segnali di ingresso I_2, I_1, I_0 da cui ha origine la selezione del messaggio stesso.

Gli altri tre bit del prologo N_{i2}, N_{i1}, N_{i0} codificano il "sequence number" del messaggio, ovvero un valore numerico via via incrementato modulo 8 ad ogni generazione di un messaggio del tipo M_i.

L'epilogo consiste nel bit di disparità D calcolato in base al valore di tutti i precedenti bit del messaggio.

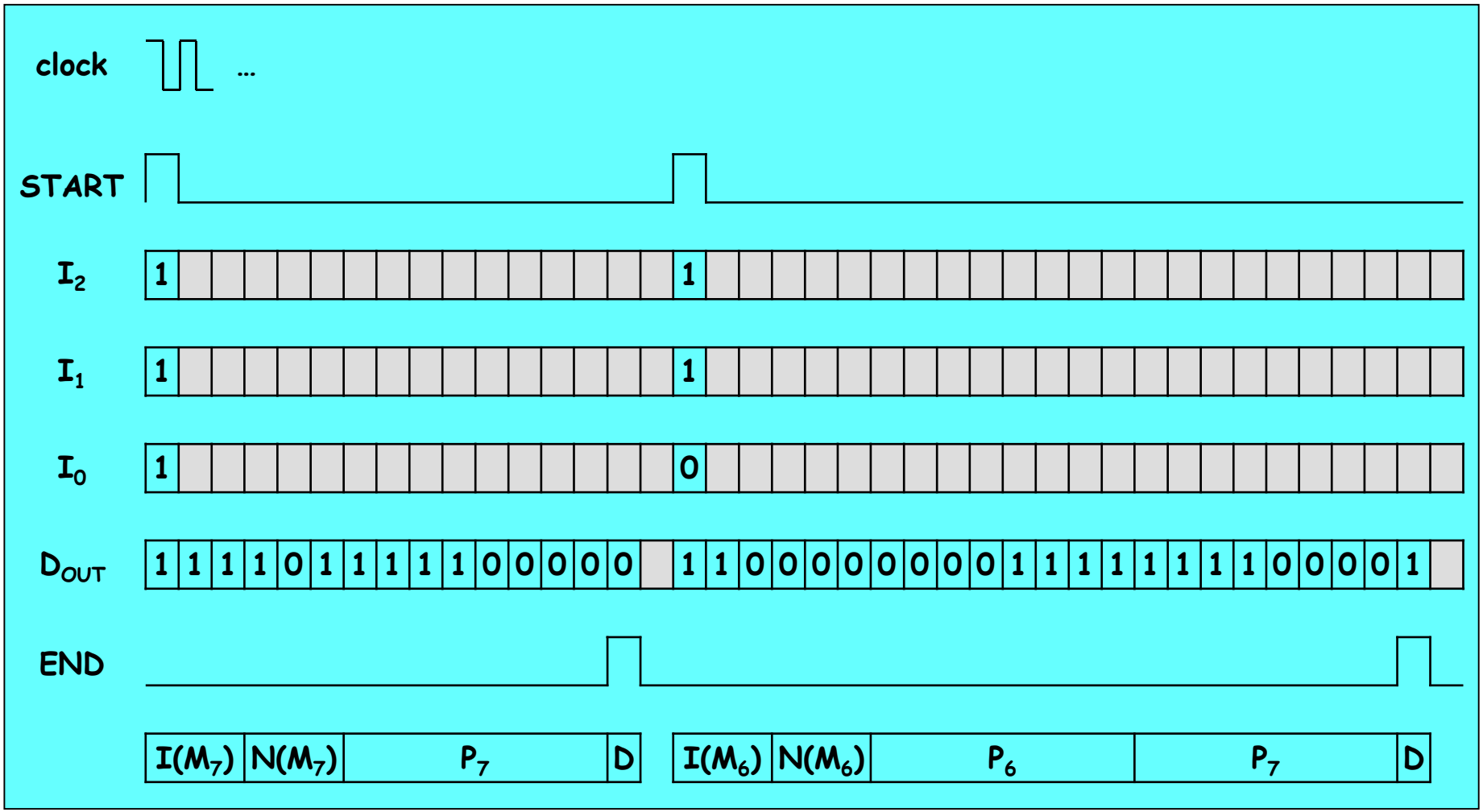
Il segnale START ha per ipotesi durata di attivazione (livello logico 1) unitaria. I segnali I_2, I_1, I_0 sono da intendersi significativi soltanto in corrispondenza dell'intervallo di attivazione del segnale START. Il segnale END deve essere attivato (livello logico 1) in corrispondenza dell'intervallo di generazione dell'ultimo bit di ogni messaggio.

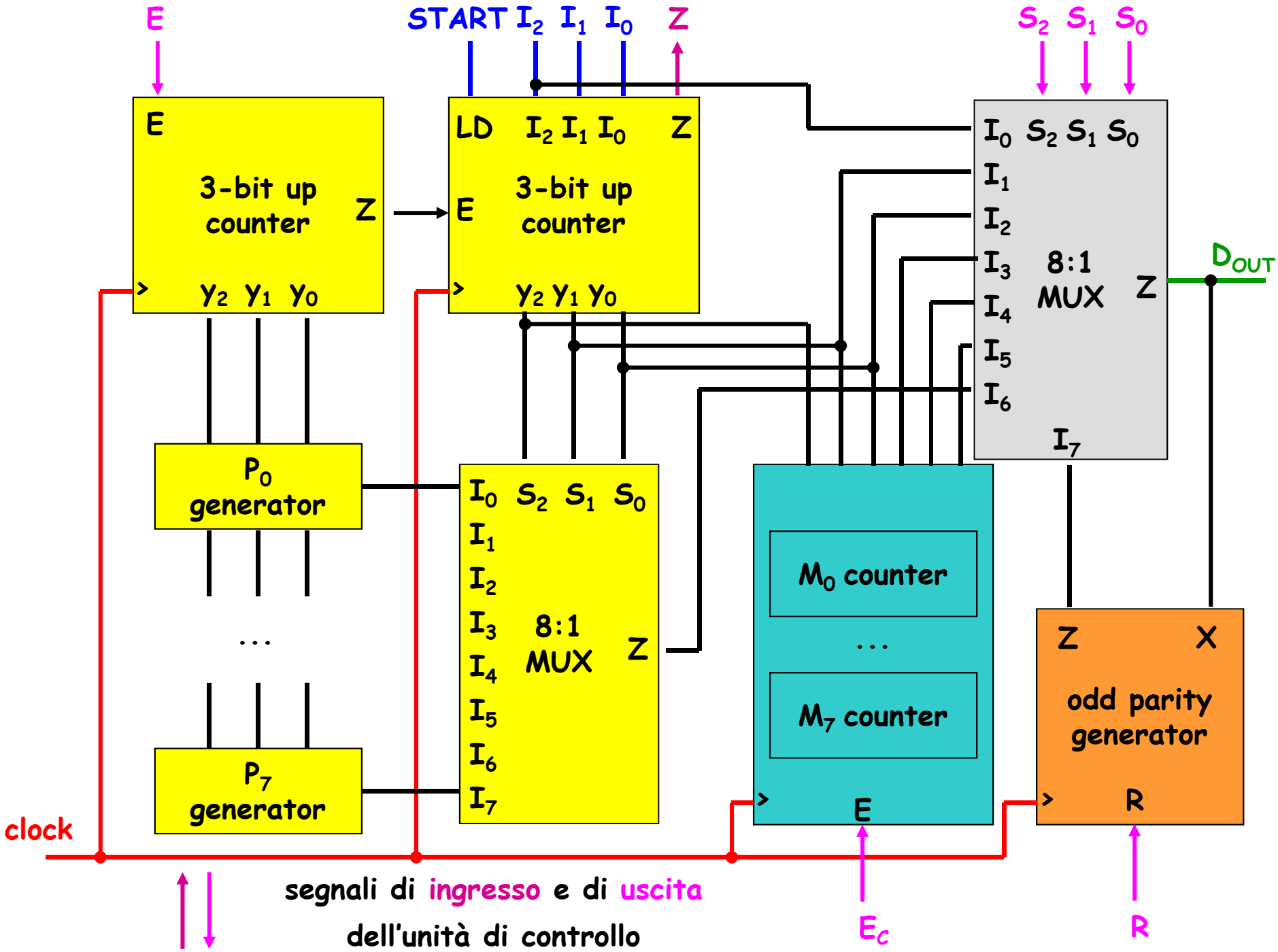
Il sistema deve essere strutturato in un'unità di controllo e in un data-path articolato, come indicato in figura, in:

- una unità di selezione di ciascun messaggio e di serializzazione dei singoli bit delle parole corrispondenti;
- una unità di elaborazione del numero di sequenza associato a ciascun tipo di messaggio;
- una unità di calcolo del bit di disparità associato a ciascun messaggio;
- una unità di multiplazione in uscita di tutti i bit (prologo, corpo ed epilogo) relativi a ciascun messaggio.

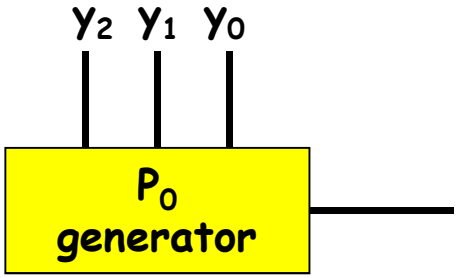
Si esegua il progetto delle suddette unità di elaborazione.

Si definisca il grafo degli stati dell'unità di controllo ed una sua possibile realizzazione basata su un contatore binario.


clock
...
START
I2
1
1
I1
1
1
I0
1
0
DOUT
1 1 1 1 0 1 1 1 1 1 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1
END
I(M7)
N(M7)
P7
D
I(M6)
N(M6)
P6
P7
D


E
START
I2 I1 I0
Z
S2 S1 S0
E
3-bit up counter
Z
> Y2 Y1 Y0
LD I2 I1 I0 Z
E
3-bit up counter
> Y2 Y1 Y0
I0 S2 S1 S0
I1
I2
I3
I4
I5
I6
I7
8:1 MUX
Z
DOUT
P0 generator
...
P7 generator
I0 S2 S1 S0
I1
I2
I3
I4
I5
I6
I7
8:1 MUX
Z
M0 counter
...
M7 counter
> E
Z X
odd parity generator
> R
clock
segnali di ingresso e di uscita
dell'unità di controllo
EC
R

Progetto dei generatori delle parole



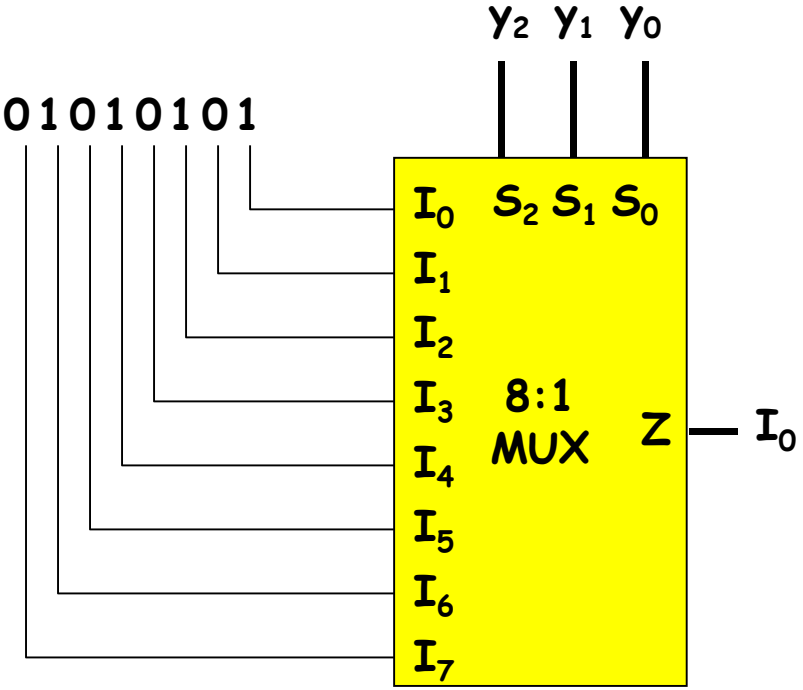
(analogamente per le altre parole)

$P_0 \equiv (P_{07}\, P_{06}\, P_{05}\, P_{04}\, P_{03}\, P_{02}\, P_{01}\, P_{00}) = 01010101$

Realizzazione mediante

logica programmabile

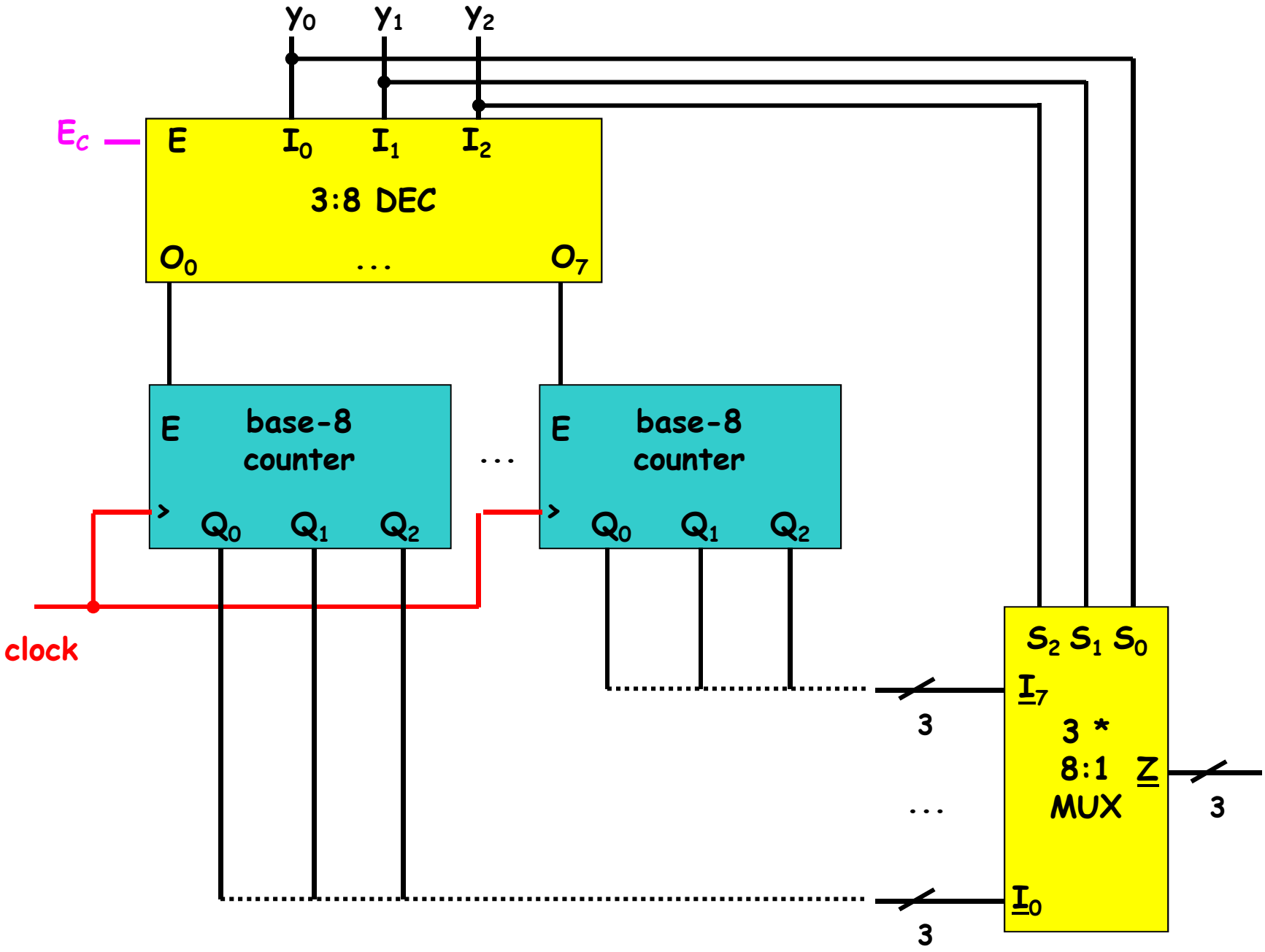
o logica cablata



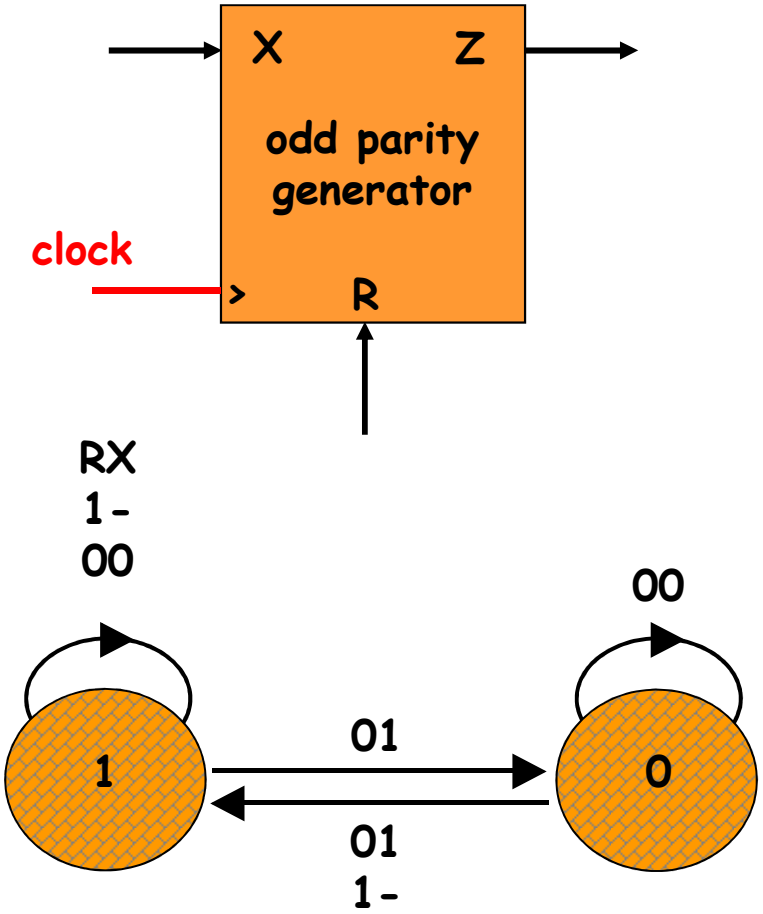
y_2 \ y_1y_0	00	01	11	10
0	1	0	0	1
1	1	0	0	1

$I_0 = y_0'$

Progetto dei generatori dei "sequence number"



Progetto del generatore del bit di disparità



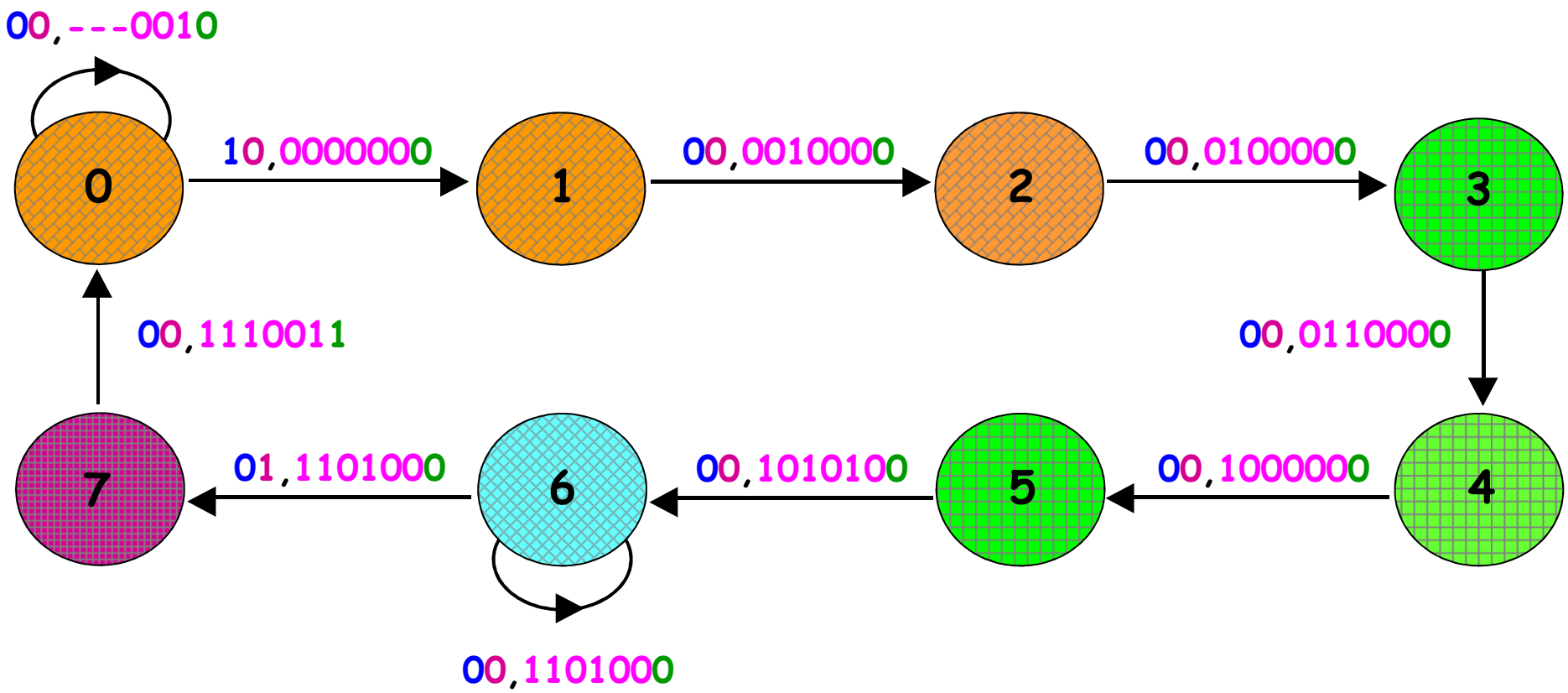
$$Y^{n+1} = (R + (X \oplus y))^n$$

$$Z^n = y^n$$

Progetto dell'unità di controllo

Il grafo degli stati

Legenda: START Z, $S_2S_1S_0$ E E_c R END



Problema 3

Il gioco del "SUDOKU" consiste nel riempire una matrice di 9x9 celle, o meglio nel completarne il riempimento essendo già predefinito il contenuto di un certo numero di esse in base all'annesso livello di difficoltà, in modo tale che nelle 9 celle di ciascuna riga R_i (i = 1, 2, …, 9), di ciascuna colonna C_j (j = 1, 2, …, 9) e di ciascuna sottomatrice S_k (k = 1, 2, …, 9) siano presenti tutte, e quindi ognuna una sola volta, le nove cifre decimali 1, 2, …, 9.

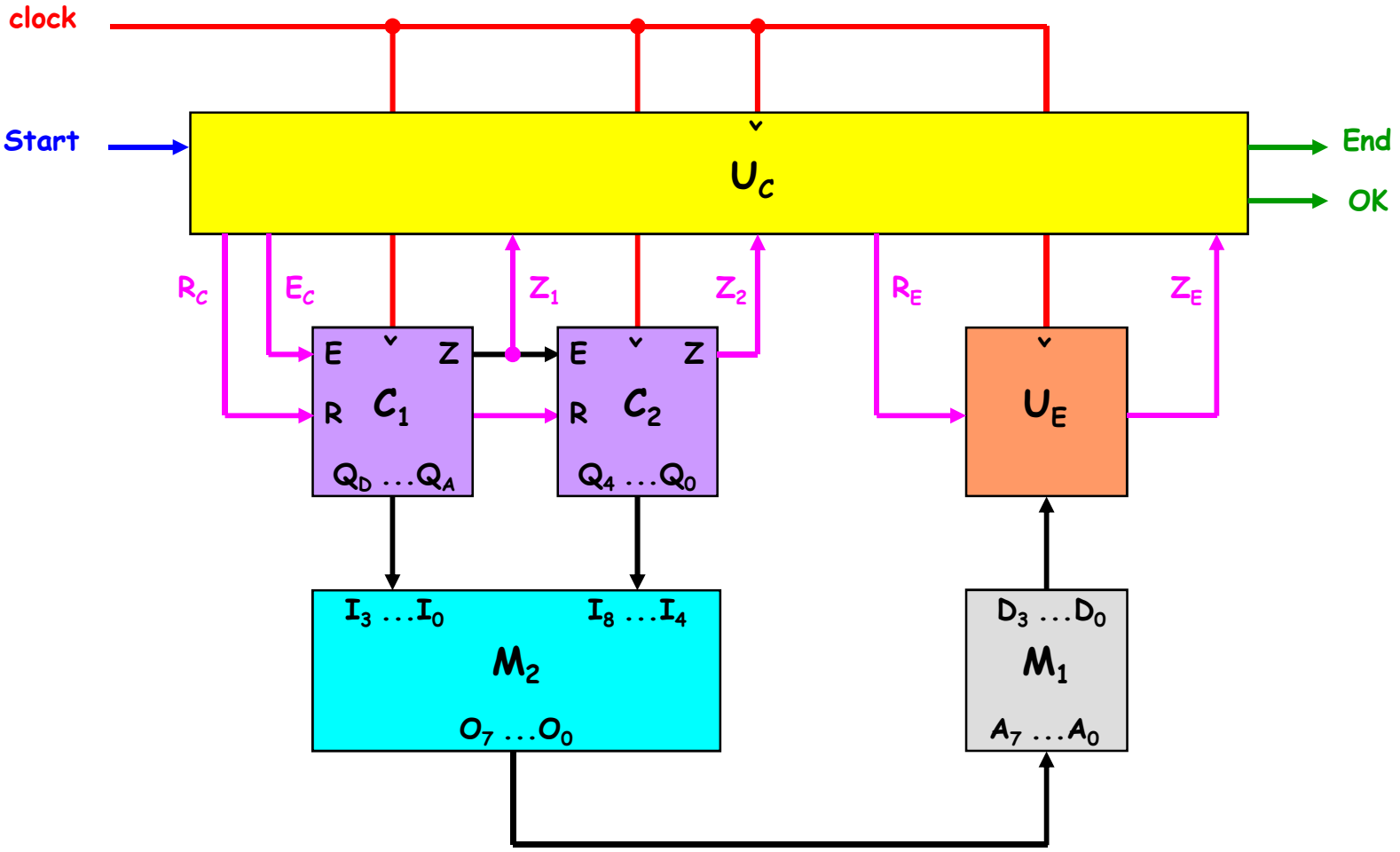
Nelle versioni portatili disponibili in commercio, la console del "SUDOKU GAME" comprende, oltre al display della matrice di celle, alcuni tasti funzionali per mezzo dei quali il giocatore può avviare un nuovo gioco selezionandone il livello di difficoltà, definire il valore numerico da associare ad ogni cella della matrice il cui contenuto non sia già predefinito, segnalare la conclusione del gioco sottoponendo così a verifica la soluzione individuata.

S_1	C_1	C_2						C_9	S_3
R_1						8		5	6
R_2		1		3		5			4
			8		1	7	9		
	7		1	2	9	4			8
	9			5	8	6	7		1
			7	4	5		6		
	1			8		3		4	
R_9	5	4		1					
S_7									S_9

2	7	3	9	4	8	1	5	6
6	1	9	3	2	5	8	7	4
4	5	8	6	1	7	9	2	3
7	3	1	2	9	4	5	6	8
8	6	5	7	3	1	4	9	2
9	2	4	5	8	6	7	3	1
3	8	7	4	5	2	6	1	9
1	9	6	8	7	3	2	4	5
5	4	2	1	6	9	3	8	7

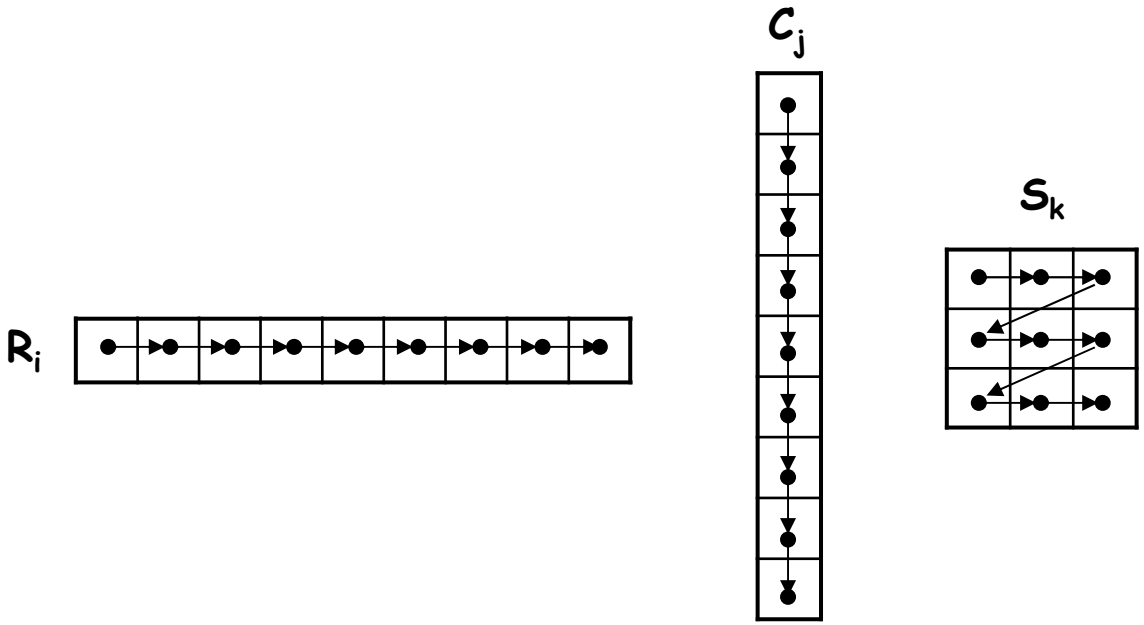
Limitando il progetto agli aspetti realizzativi connessi con quest'ultima funzionalità, il (sotto)sistema di controllo può essere strutturato, in accordo al modello "Data-Path & Control Unit", come indicato in figura.

L'unità di controllo U_C, avvalendosi delle risorse previste a livello di Data-Path e coordinandone opportunamente il funzionamento, ha il compito di gestire il processo di verifica della correttezza di una soluzione del gioco ogni qual volta viene dall'esterno attivato (livello logico 1, durata unitaria) il segnale di ingresso Start. Completata la verifica, U_C deve prontamente notificarne in uscita l'esito tramite il segnale OK (OK = 1 in caso di esito positivo, OK = 0 in caso contrario), contestualmente attivando (livello logico 1, durata unitaria) il segnale di uscita End.



Il processo di verifica consiste nell'esaminare in sequenza i 27 sottoinsiemi di 9 celle della matrice (le 9 righe, le 9 colonne, le 9 sottomatrici), al fine di accertare se il relativo contenuto viola o meno le regole del gioco (nel primo caso è chiaramente del tutto inutile estendere l'indagine ai successivi sottoinsiemi).

L'ordine secondo cui procedere nella scansione dei sottoinsiemi è R_1, R_2, …, R_9, C_1, C_2, …, C_9, S_1, S_2, …, S_9, mentre quello delle celle nell'ambito di un sottoinsieme è evidenziato in figura.



Il compito di evidenziare, una volta completata la scansione delle 9 celle di un sottoinsieme, se esse contengono valori numerici distinti è affidato all'unità di elaborazione U_E (segnale di uscita Z_E = 1).

Il ruolo di discriminare quale sia, in ogni intervallo di clock, il particolare sottoinsieme oggetto di indagine, e nell'ambito di esso la specifica cella, è affidato ai due contatori binari C_1 (base di conteggio 9) e C_2 (base di conteggio 27), dotati di segnale di reset sincrono ed opportunamente disposti in cascata. Più precisamente, le variabili di stato di C_2 (Q_4 (MSB), Q_3, Q_2, Q_1, Q_0 (LSB)) identificano il sottoinsieme di celle (il 1°, ovvero R_1, allorché $Q_4Q_3Q_2Q_1Q_0$ = 00000, …, il 27°, ovvero S_9, allorché $Q_4Q_3Q_2Q_1Q_0$ = 11010), mentre quelle di C_1 (Q_D (MSB), Q_C, Q_B, Q_A (LSB)) la specifica cella (la 1ª allorché $Q_DQ_CQ_BQ_A$ = 0000, …, la 9ª allorché $Q_DQ_CQ_BQ_A$ = 1000).

Per ipotesi, i valori numerici associati alle 81 celle della matrice, ciascuno rappresentato secondo il codice BCD, sono disponibili nell'unità di memoria M_1 (del tipo RAM).

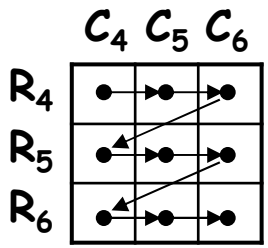
Tale memoria, contraddistinta da 8 bit di ingresso A_7 (MSB), A_6, A_5, A_4, A_3, A_2, A_1, A_0 (LSB) e da 4 bit di (ingresso-)uscita D_3 (MSB), D_2, D_1, D_0 (LSB), fornisce in uscita il valore numerico associato alla cella corrispondente alla riga R_i e alla colonna C_j (i, j = 1, 2, …, 9) a fronte della presentazione in ingresso dell'indirizzo $A \equiv A_{MSD}A_{LSD} = (ij)_{BCD}$, essendo $A_{MSD} \equiv A_7A_6A_5A_4$, $A_{LSD} \equiv A_3A_2A_1A_0$ (l'indirizzo $A_7A_6A_5A_4A_3A_2A_1A_0$ = 00010001 seleziona il contenuto della prima cella (in alto a sinistra) della matrice, …, l'indirizzo $A_7A_6A_5A_4A_3A_2A_1A_0$ = 10011001 quello dell'ultima cella (in basso a destra)).

L'unità di memoria M_2 (del tipo ROM), contraddistinta da 9 bit di ingresso I_8 (MSB), I_7, I_6, I_5, I_4, I_3, I_2, I_1, I_0 (LSB), nell'ordine connessi a $Q_4Q_3Q_2Q_1Q_0Q_DQ_CQ_BQ_A$, e da 8 bit di uscita O_7 (MSB), O_6, O_5, O_4, O_3, O_2, O_1, O_0 (LSB), nell'ordine connessi a $A_7A_6A_5A_4A_3A_2A_1A_0$, è quindi prevista allo scopo di generare l'indirizzo della cella oggetto di indagine in ogni intervallo di clock a partire dalla corrispondente modalità di identificazione adottata nell'ambito dell'unità di conteggio.

Motivando esplicitamente tutte le decisioni progettuali operate ed assumendo che il tempo complessivo di accesso alle due memorie M_1 e M_2 disposte in serie sia sensibilmente inferiore rispetto al periodo del segnale di clock (T_{CLK} = 1 μs) in base al quale opera il sistema:

- si definisca l'indirizzo ed il contenuto delle locazioni di memoria di M_2 corrispondenti alle celle della sottomatrice S_5;
- si esegua il progetto dell'unità di elaborazione U_E (ingressi: R_E (reset); uscite: Z_E), avvalendosi dei componenti ritenuti più idonei allo scopo;
- si formalizzi il comportamento dell'unità di controllo U_C in termini di automa a stati finiti (ingressi: Start, Z_1, Z_2, Z_E; uscite: R_C, E_C, R_E (modello di Moore), End, OK (modello di Mealy));
- si individui il tempo massimo impiegato dal sistema per attuare il processo di verifica.

M_2

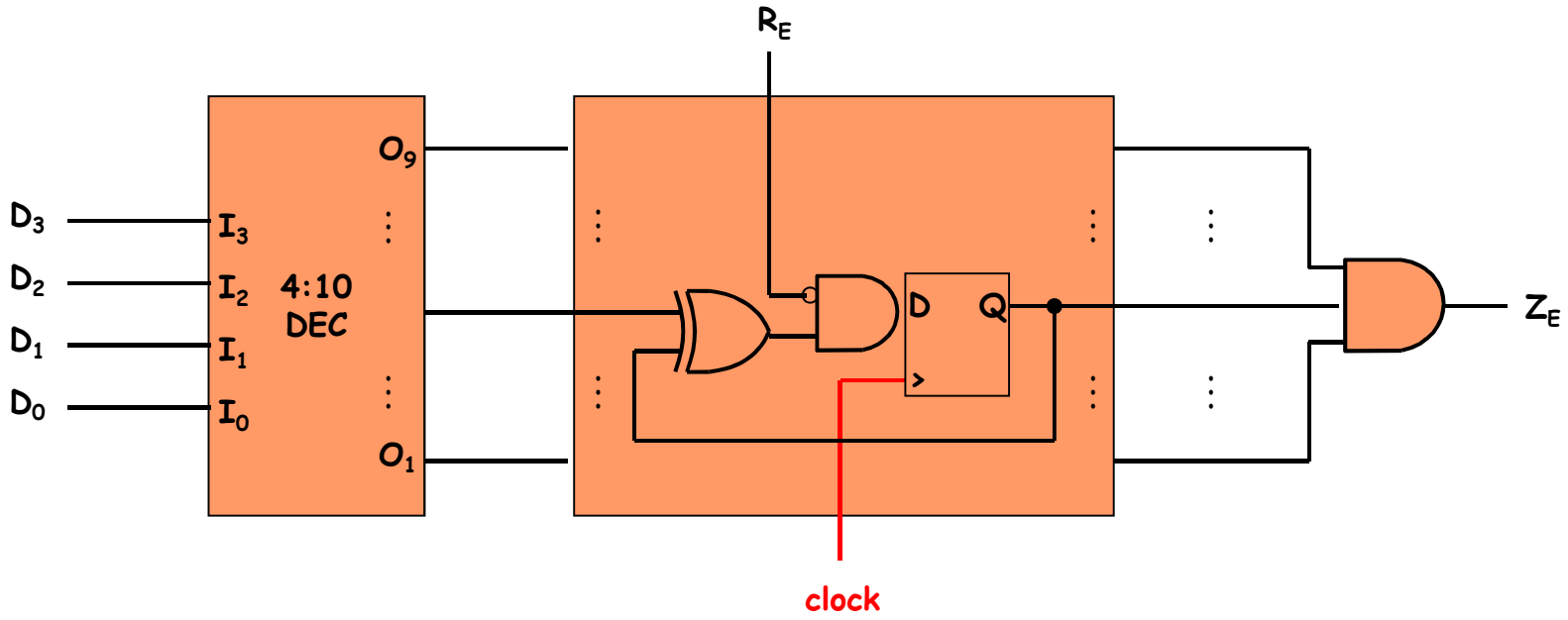


S_5

23° sottoinsieme: 23-1 = $(10110)_2$ = $(16)_{HEX}$

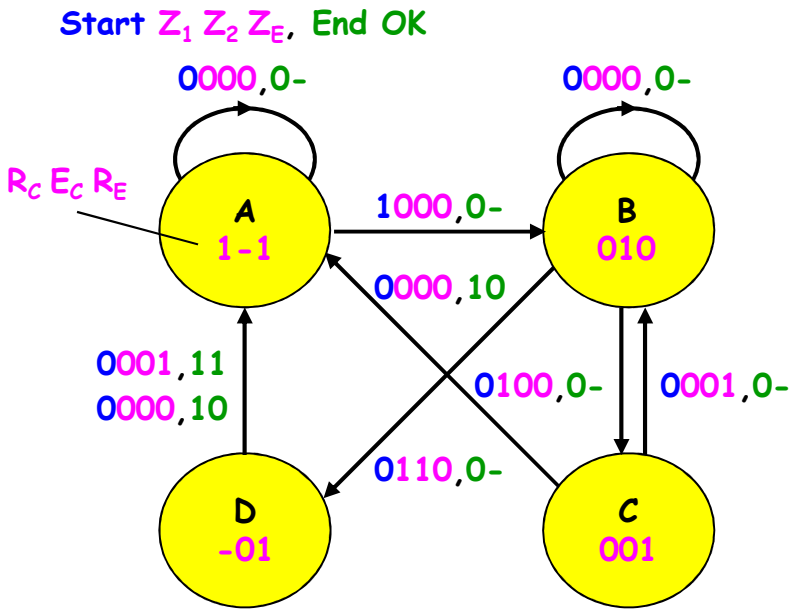
$(I_8 \ldots I_0)_{HEX}$	$(O_7 \ldots O_0)_{BCD}$
160	44
161	45
162	46
163	54
164	55
165	56
166	64
167	65
168	66

U_E



U_C

$Q_4 \ldots Q_0$	22									23				
$Q_D \ldots Q_A$	0	1	2	3	4	5	6	7	8	0	0	1	2	…
$D_3 \ldots D_0$	2	9	4	7	3	1	5	8	6	…	…	…	…	…
R_C														
E_C														
Z_1														
Z_2														
Z_E														
R_E														
stato U_C	B									C				B



$t_{MAX} = 27 * (9 + 1)\ T_{CLK} = 270\ \mu s$

Problema 4

Un sistema digitale, operante in base ad un segnale di clock di frequenza f_q, ha il compito di generare in uscita un segnale analogico con forma d'onda triangolare o a dente di sega.

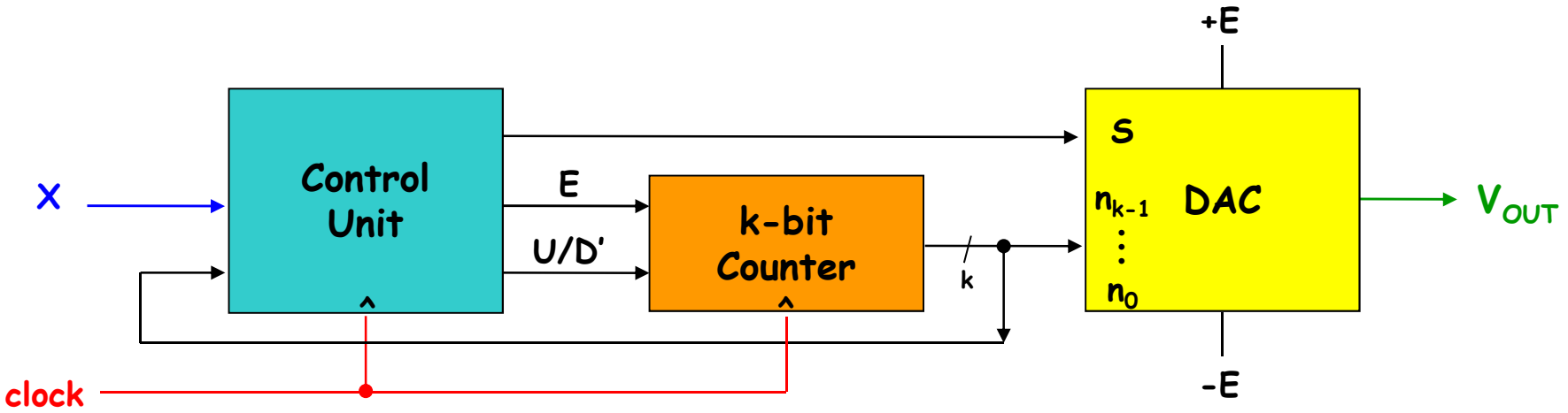
A tal fine il sistema si avvale di un convertitore digitale/analogico (DAC), dotato di duplice sorgente di alimentazione in continua (± E Volt). La tensione V_{OUT} che il DAC fornisce in uscita, di valore compreso nell'intervallo [-E, +E], dipende dalla configurazione contestualmente assunta dai relativi k+1 segnali digitali di ingresso n_{k-1}, …, n_1, n_0, S. Più precisamente, la configurazione binaria $N \equiv \{n_{k-1}$ (MSB) … n_1 n_0 (LSB)$\}$ dei primi k segnali definisce il valore assoluto di V_{OUT} ($|V_{OUT}| = N\,E / (2^k-1)$), mentre il segnale S ne stabilisce la polarità (positiva se S = 0, negativa se S = 1 (il valore di S è irrilevante se N = 0)).

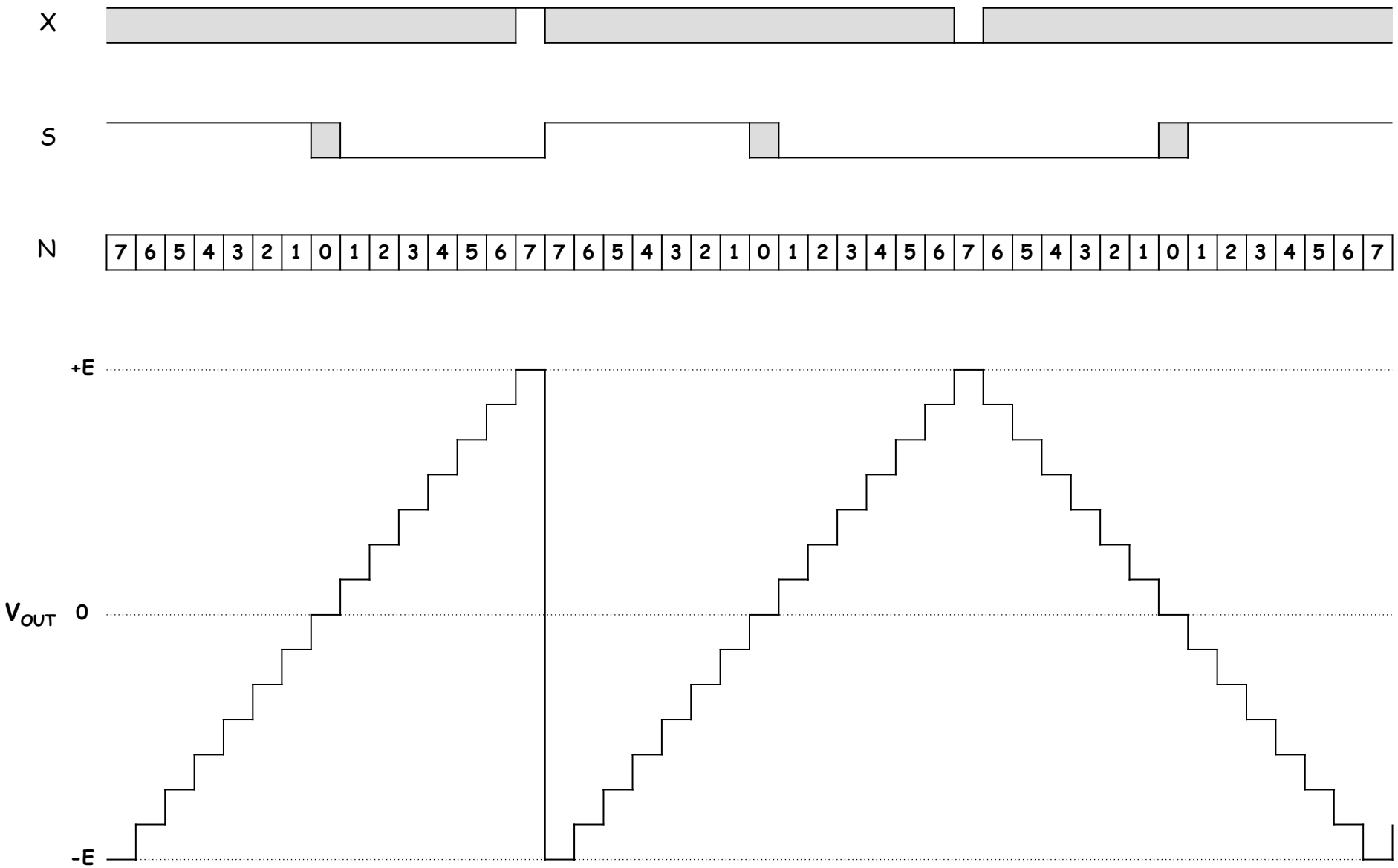
Come evidenziato nel seguente diagramma temporale che, a titolo esemplificativo, si riferisce ad un elementare, peraltro inesistente, DAC a 4 bit (k = 3), il sistema deve attuare il processo di generazione del segnale analogico di uscita alternando nel tempo, senza soluzione di continuità, forme d'onda dell'uno o dell'altro tipo in dipendenza del valore assunto da un segnale di ingresso X (sincrono): forma d'onda triangolare se X = 0, forma d'onda a dente di sega se X = 1. Il valore del segnale X deve essere tuttavia preso in considerazione dal sistema soltanto al termine del processo di generazione di un'intera forma d'onda dell'uno o dell'altro tipo, ovvero, più precisamente, in corrispondenza dell'intervallo di clock in cui V_{OUT} assume il valore massimo positivo (+E).

Il sistema deve essere strutturato in accordo al modello "data-path & control unit", secondo lo schema indicato in figura. Gli ingressi del DAC che definiscono il valore assoluto della tensione di uscita sono ordinatamente collegati alle uscite di un contatore binario bidirezionale a k bit, dotato di segnali di controllo E e U/D' (0-: HOLD, 11: UP, 10: DOWN). L'ingresso del DAC che definisce la polarità della tensione di uscita, così come i segnali di controllo E e U/D' del contatore, sono direttamente gestiti dall'unità di controllo.

1. Si definisca il grafo degli stati in forma minima dell'unità di controllo.
2. Si calcoli, in funzione di f_q e k, la frequenza nominale f_{NOM} del segnale di uscita del sistema per ciascun tipo di forma d'onda.
3. Si identifichino, avvalendosi dei componenti ritenuti più idonei allo scopo, le estensioni da apportare allo schema al fine di conseguire la programmabilità della frequenza del segnale di uscita del sistema in accordo ad un parametro di ingresso R, di valore intero compreso nel range [1, 100] e codificato in binario mediante 7 bit:

$$f_{OUT} = f_{NOM} / R.$$

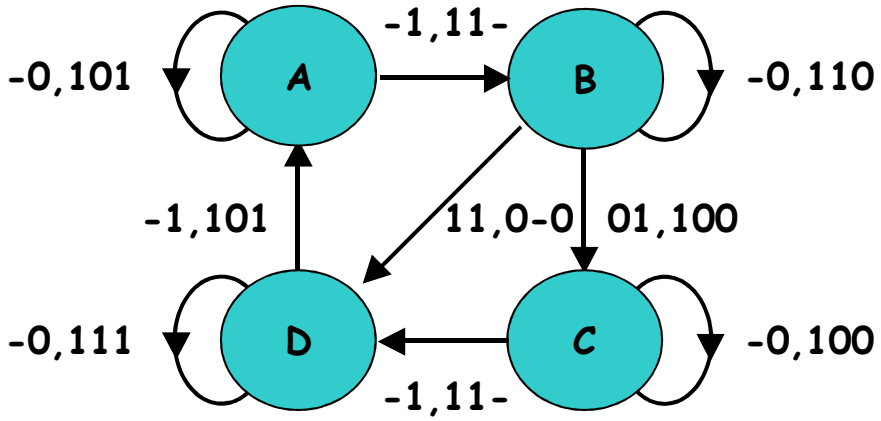


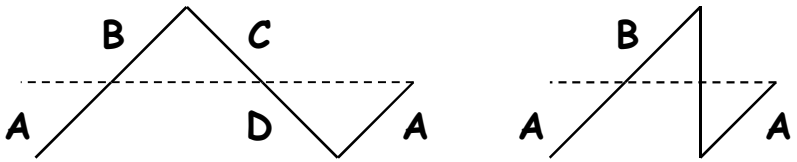

X
S
N
7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
+E
V_{OUT}
0
-E

Unità di controllo

$$Z = n_{k-1} \dots n_1 \; n_0 + n_{k-1}' \dots n_1' \; n_0'$$

X Z, E U/D' S

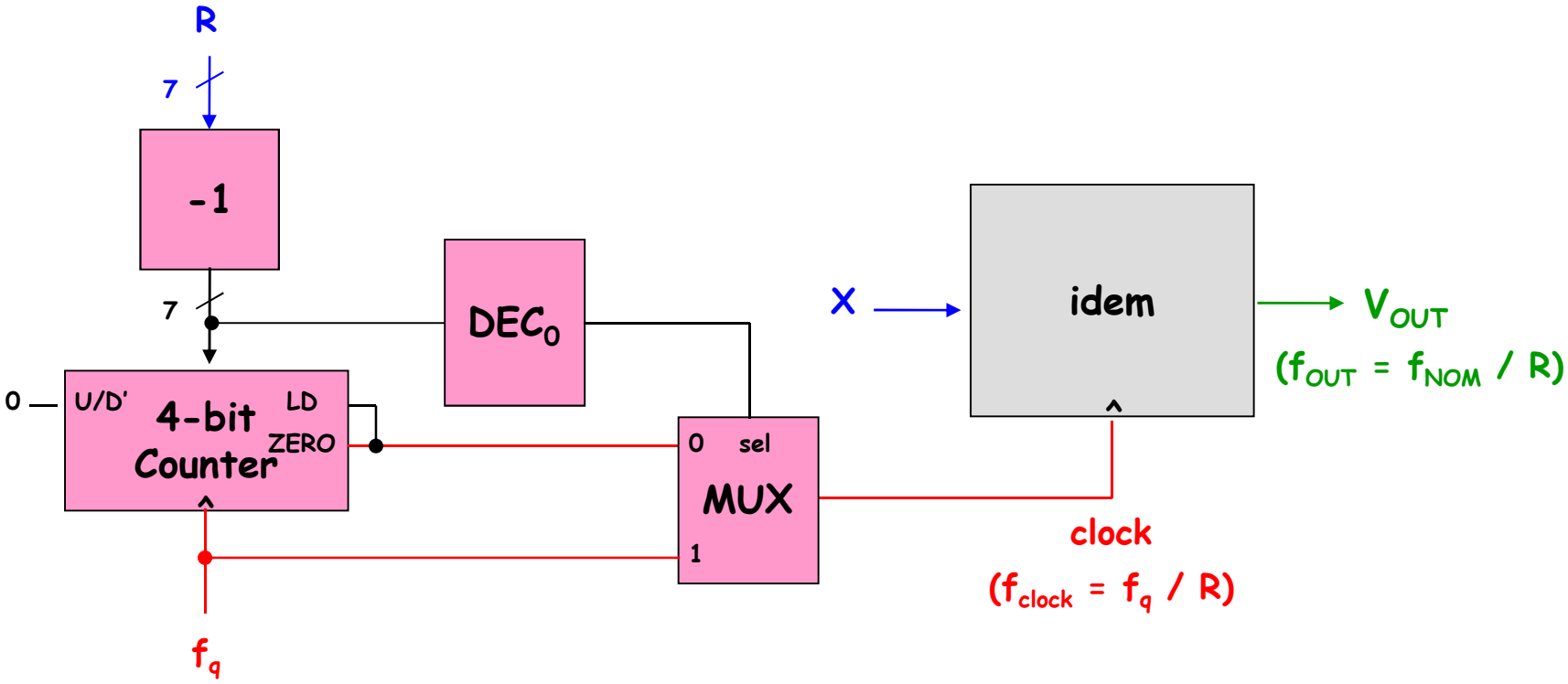




Estensioni orientate alla programmabilità della frequenza del segnale di uscita

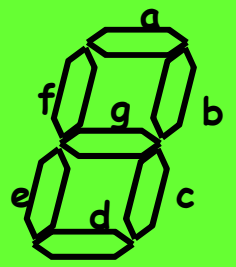
f_{NOM} (forma d'onda triangolare) $= f_q / (4 * (2^k-1))$

f_{NOM} (forma d'onda a dente di sega) $= f_q / (2 * (2^k-1) + 1)$



Problema 5

In un sistema di comunicazione digitale vengono trasferiti dati numerici di M cifre decimali, ciascuna rappresentata tramite il codice separabile 7 segmenti – Hamming mediante 11 bit (a, b, c, d, e, f, g, H_1, H_2, H_4, H_8). I primi 7 bit (bit di informazione) identificano secondo il codice 7 segmenti il valore di ciascuna cifra; gli altri 4 bit (bit di controllo), calcolati dal trasmettitore come segue



$$H_1 = a \oplus b \oplus d \oplus e \oplus g,$$
$$H_2 = a \oplus c \oplus d \oplus f \oplus g,$$
$$H_4 = b \oplus c \oplus d,$$
$$H_8 = e \oplus f \oplus g,$$

sono previsti onde consentire al ricevitore, a fronte di qualunque errore *singolo* verificatosi durante la trasmissione di una cifra, di correggere l'errore, ovvero di identificare la configurazione dei bit di informazione originariamente inviata dal trasmettitore. Allo scopo il ricevitore deve semplicemente calcolare le 4 sindromi di errore

$$E_1 = H_1 \oplus a \oplus b \oplus d \oplus e \oplus g,$$
$$E_2 = H_2 \oplus a \oplus c \oplus d \oplus f \oplus g,$$
$$E_4 = H_4 \oplus b \oplus c \oplus d,$$
$$E_8 = H_8 \oplus e \oplus f \oplus g,$$

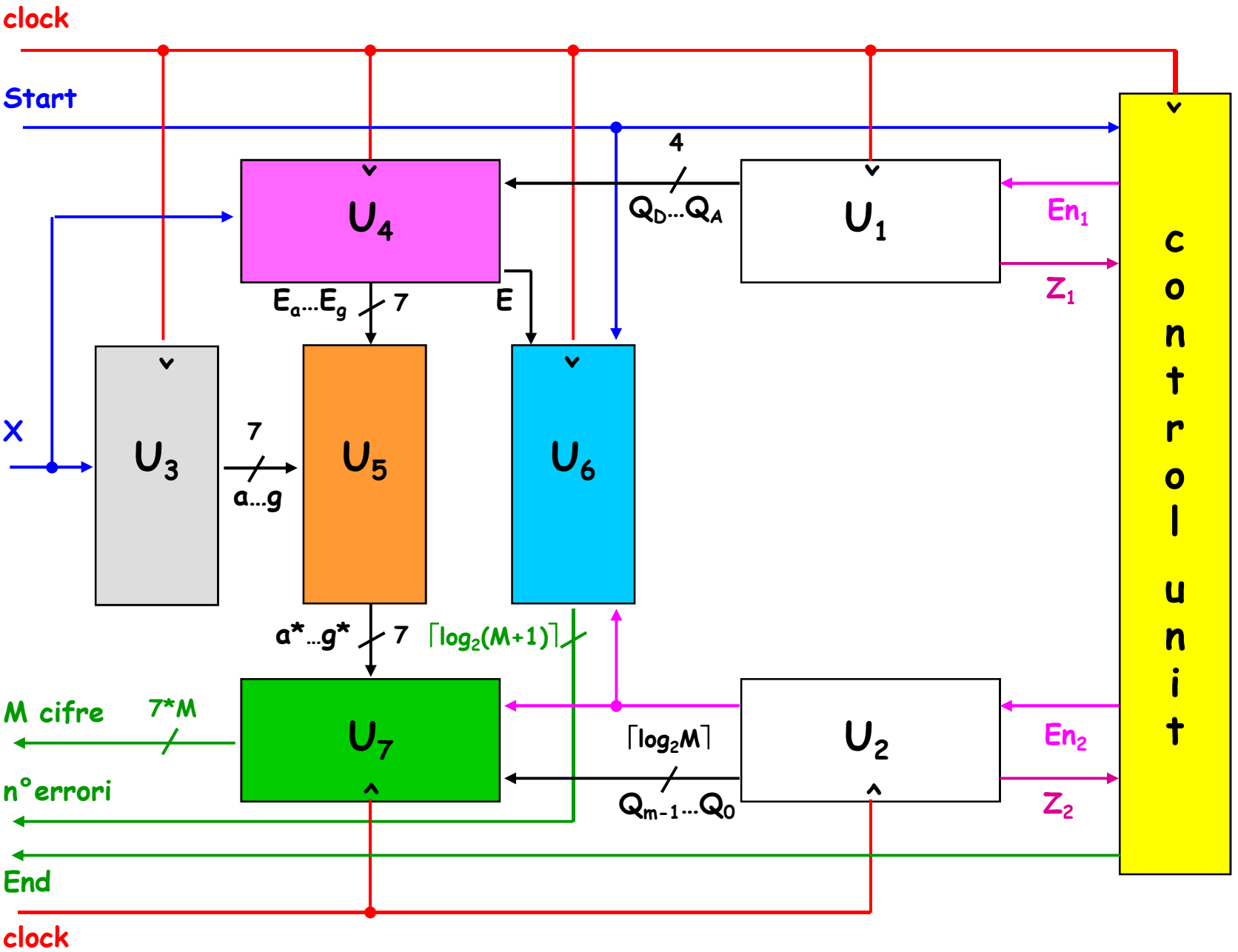
che congiuntamente identificano, con il valore zero, la condizione "nessun errore", ed in caso contrario il bit, di informazione o di controllo, affetto da errore. Più precisamente:

E_8E_4 \ E_2E_1	00	01	11	10
00	-	H_1	a	H_2
01	H_4	b	d	c
11	-	-	-	-
10	H_8	e	g	f

Le specifiche di progetto del ricevitore sono le seguenti. I bit rappresentativi di ogni dato numerico sono ricevuti serialmente, in 11 * M consecutivi intervalli di clock, attraverso il segnale di ingresso X. Gli 11 bit rappresentativi di ciascuna cifra sono ricevuti nell'ordine H_1, H_2, H_4, H_8, a, b, c, d, e, f, g. Il segnale di ingresso Start, attivo a livello logico 1 e di durata unitaria, identifica l'intervallo di ricezione del primo bit di ogni dato. Completato il processo di elaborazione di un dato, il ricevitore deve attivare il segnale End e fornire in uscita, in parallelo, sia i bit di informazione a*, b*, c*, d*, e*, f*, g* (non necessariamente coincidenti, ovviamente, con a, b, c, d, e, f, g) di tutte le corrispondenti M cifre, sia il numero di errori in esse riscontrato. Il risultato di ciascuna elaborazione, ovvero il valore dei $7 * M + \lceil \log_2 (M+1) \rceil$ bit di uscita, deve essere reso disponibile fino alla successiva attivazione del segnale Start.

Il sistema deve essere strutturato secondo il modello data-path & control unit, come indicato in figura. Il data-path comprende sette unità funzionali:

- una unità di conteggio dei bit relativi a ciascuna cifra presentata in ingresso;
- una unità di conteggio delle cifre relative ad ogni dato presentato in ingresso;
- una unità di conversione serie/parallelo dei bit relativi a ciascuna cifra presentata in ingresso;
- una unità di rilevazione degli errori, cui è delegato il compito di calcolare le sindromi di errore E_1, E_2, E_4, E_8 per ciascuna cifra presentata in ingresso e di generare conseguentemente, oltre ad un segnale cumulativo di errore E, i 7 segnali E_a, E_b, E_c, E_d, E_e, E_f, E_g che identificano quale bit di informazione sia eventualmente da correggere;
- una unità di correzione degli errori, cui è delegato il compito di generare la corretta configurazione dei bit di informazione di ciascuna cifra presentata in ingresso;
- una unità di conteggio degli errori riscontrati nell'ambito di ogni dato presentato in ingresso;
- una unità di memorizzazione, cui è delegato il compito di rendere disponibili in uscita i bit di informazione corrispondenti alle M cifre rappresentative di ogni dato presentato in ingresso.


clock
Start
4
$Q_D \ldots Q_A$
U_4
U_1
En_1
Z_1
control unit
$E_a \ldots E_g$
7
E
X
U_3
7
a...g
U_5
U_6
a*...g*
7
$\lceil \log_2(M+1) \rceil$
M cifre
7*M
U_7
$\lceil \log_2 M \rceil$
U_2
En_2
Z_2
n°errori
$Q_{m-1} \ldots Q_0$
End
clock

L'unità funzionale U_1 è costituita da un contatore binario x11, il cui comando di abilitazione al conteggio En_1 è generato dall'unità di controllo; le 4 variabili di stato del contatore Q_D, Q_C, Q_B, Q_A, unitamente al segnale $Z_1 = En_1 \,\&\, (Q_D\, Q_C\, Q_B\, Q_A = 10)$, rappresentano le uscite dell'unità.

L'unità funzionale U_2 è costituita da un contatore binario xM, il cui comando di abilitazione al conteggio En_2 è generato dall'unità di controllo; le $m = \lceil \log_2 M \rceil$ variabili di stato del contatore Q_{m-1}, …, Q_1, Q_0, unitamente al segnale $Z_2 = En_2 \,\&\, (Q_{m-1} \ldots Q_1\, Q_0 = M\text{-}1)$, rappresentano le uscite dell'unità.

L'unità funzionale U_3 è costituita da un registro a scorrimento di k (?) bit, al cui ingresso seriale è collegato il segnale X; l'unità rende disponibili in uscita i bit di informazione a, b, c, d, e, f, g.

L'unità funzionale U_4 comprende 4 moduli sequenziali identici, ognuno adibito a calcolare *serialmente* una sindrome di errore; ogni modulo riceve in ingresso, oltre al segnale X, due segnali di controllo C_1 e C_0 (generati combinatoriamente all'interno dell'unità stessa in base al valore dei segnali di ingresso Q_D, Q_C, Q_B, Q_A) che ne condizionano il funzionamento, in accordo al modello di Moore, come segue: $C_1C_0 = 00 \rightarrow$ Reset (inizializzazione della sindrome al valore 0), $C_1C_0 = 01 \rightarrow$ Load (inizializzazione della sindrome in base al valore attuale del segnale X), $C_1C_0 = 10 \rightarrow$ Hold (mantenimento del valore attuale della sindrome), $C_1C_0 = 11 \rightarrow$ Update (aggiornamento della sindrome in base al valore attuale del segnale X); i segnali di uscita dell'unità sono E, E_a, E_b, E_c, E_d, E_e, E_f, E_g.

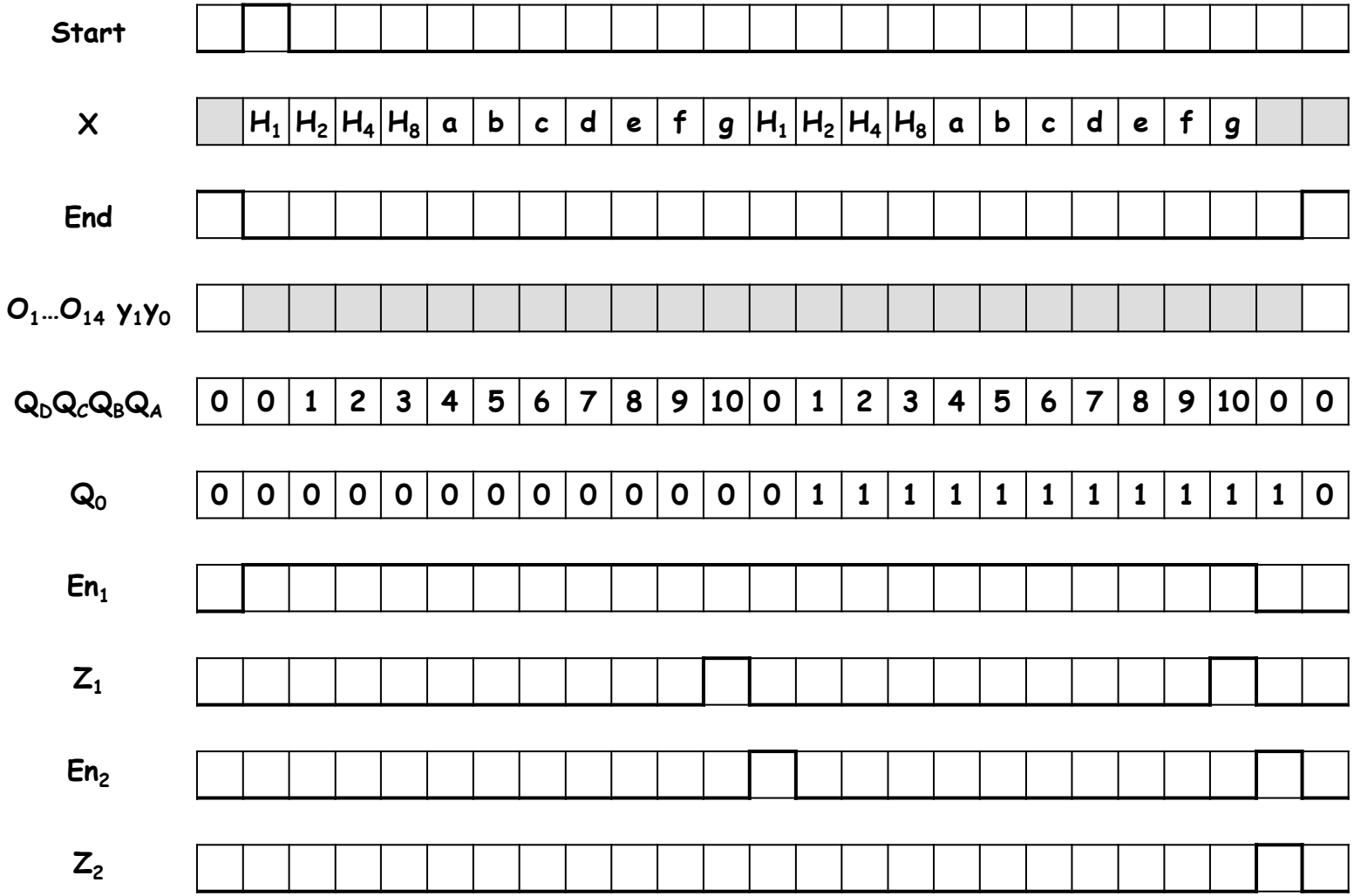
L'unità funzionale U_5 comprende 7 moduli combinatori identici, ognuno adibito a correggere un bit di informazione nel caso sia stata rilevata la sua alterazione in trasmissione da parte di U_4; i segnali di ingresso dell'unità sono a, b, c, d, e, f, g, E_a, E_b, E_c, E_d, E_e, E_f, E_g; i segnali di uscita dell'unità sono a*, b*, c*, d*, e*, f*, g*.

L'unità funzionale U_6 è costituita da un contatore binario x(M+1) dotato di comandi di abilitazione al conteggio e reset sincrono; i segnali di ingresso dell'unità sono Start, En_2 e E; i segnali di uscita dell'unità coincidono con le $n = \lceil \log_2 (M+1) \rceil$ variabili di stato del contatore y_{n-1}, …, y_1, y_0.

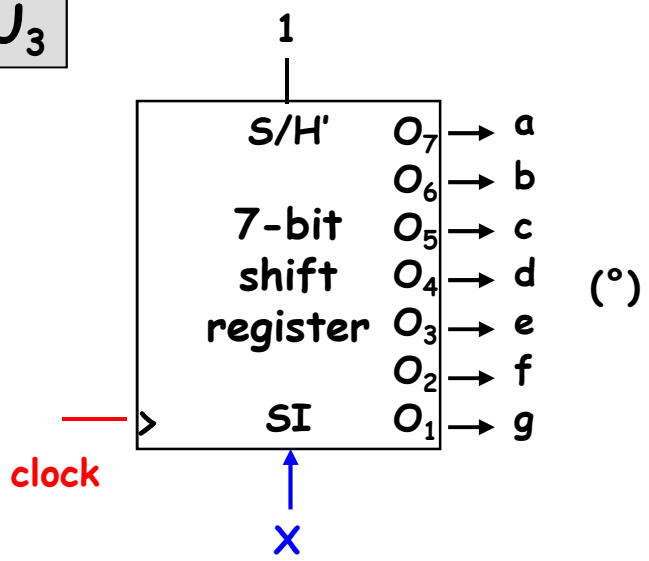
L'unità funzionale U_7 comprende M registri di 7 bit, ciascuno dotato di ingresso di abilitazione al campionamento; i segnali di ingresso dell'unità sono a*, b*, c*, d*, e*, f*, g*, En_2, Q_{m-1}, …, Q_1, Q_0; i segnali di uscita dell'unità O_1, O_2, …, O_{7M} coincidono con le uscite degli M registri.

- Si identifichi, con riferimento al caso M = 2, l'andamento temporale dei segnali di ingresso/uscita sia del sistema complessivo che dell'unità di controllo.

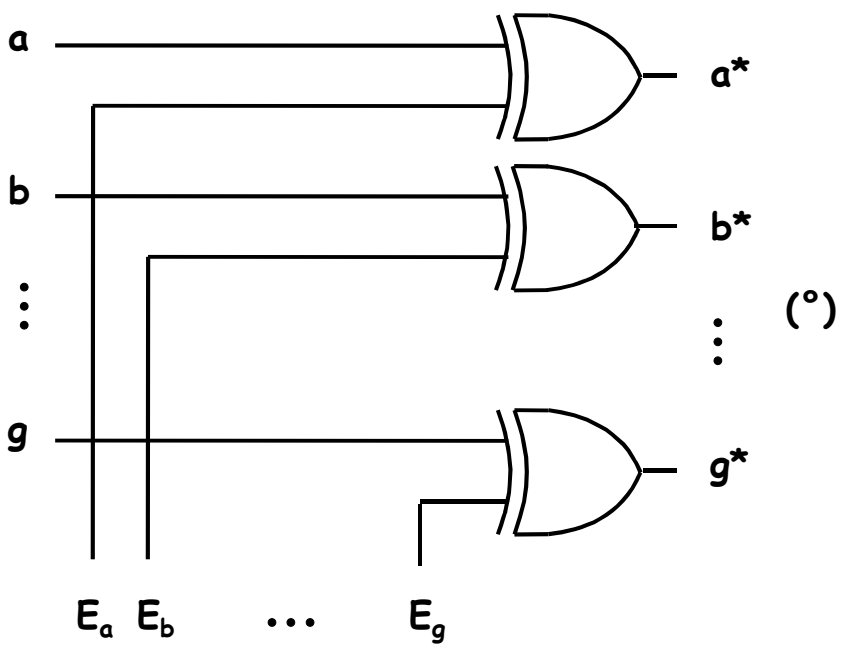
- Si completi il dimensionamento e il progetto del data-path relativamente alle unità funzionali U_3, U_4, U_5, U_6 e U_7, avvalendosi dei componenti ritenuti più idonei allo scopo e motivando esplicitamente tutte le scelte operate.
- Si definisca il grafo degli stati dell'unità di controllo (ingressi: Start, Z_1, Z_2; uscite: En_1, En_2, End).
- Si evidenzino le eventuali modifiche da apportare alle unità funzionali U_3, U_4, U_5, U_6 e U_7 nell'ipotesi che gli 11 bit rappresentativi di ciascuna cifra siano ricevuti nell'ordine a, b, c, d, e, f, g, H_1, H_2, H_4, H_8.



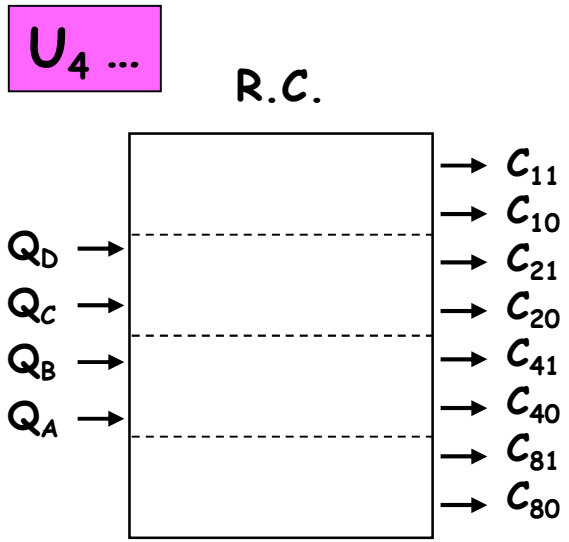
U3

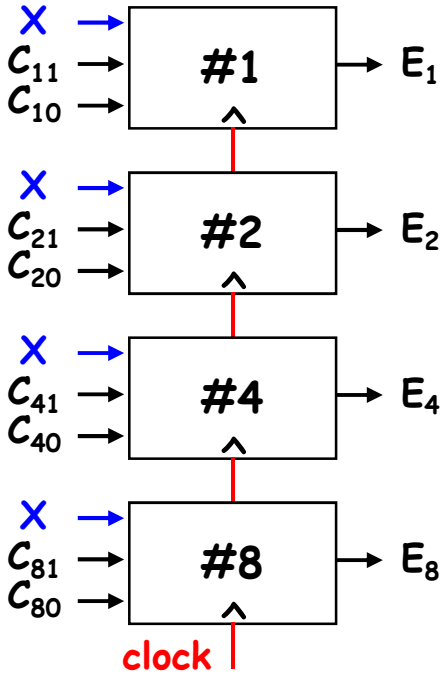


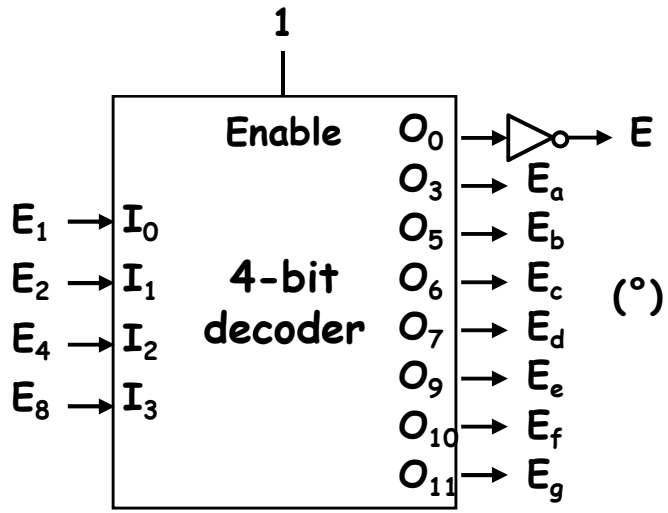
U5



U4 ...

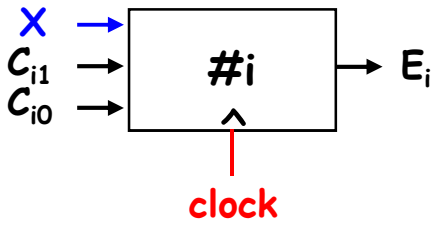






(°) allorché $En_2^n = Z_1^{n-1} = 1$

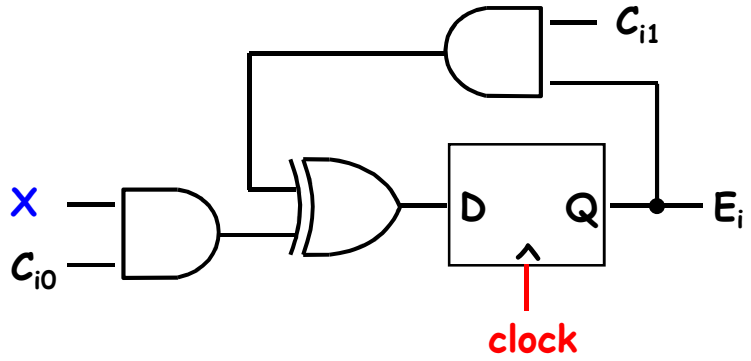
… U_4 …

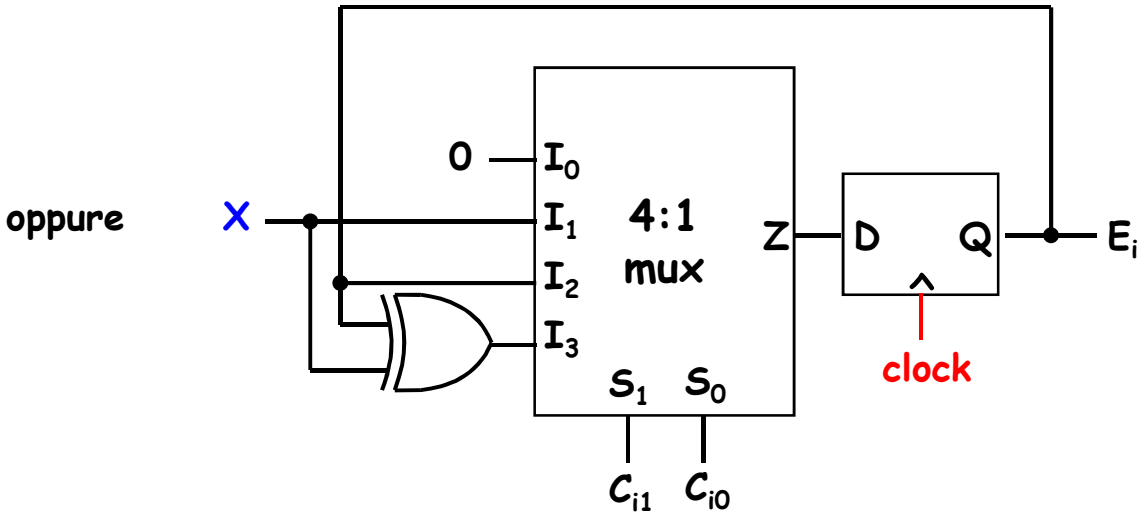


$(E_iX)^n$ \ $(C_{i1}C_{i0})^n$	Reset 00	Load 01	Update 11	Hold 10
00	0	0	0	0
01	0	1	1	0
11	0	1	0	1
10	0	0	1	1

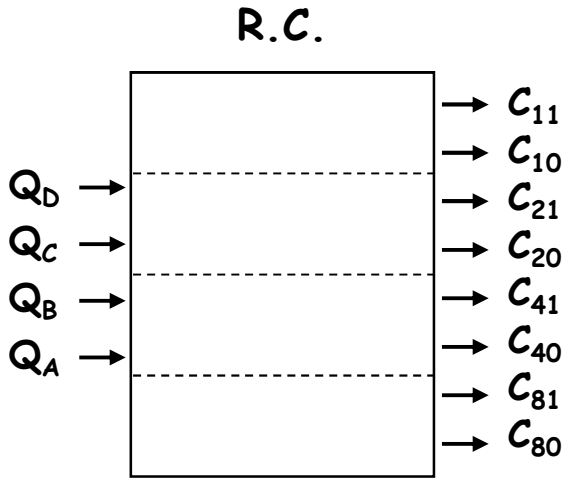
E_i^{n+1}

$$E_i^{n+1} = (X\,C_{i0}\,(E_i' + C_{i1}') + E\,C_{i1}\,(X' + C_{i0}'))^n$$
$$= (X\,C_{i0}\,(E_i\,C_{i1})' + E\,C_{i1}\,(X\,C_{i0})')^n = ((X\,C_{i0}) \oplus (E_i\,C_{i1}))^n$$





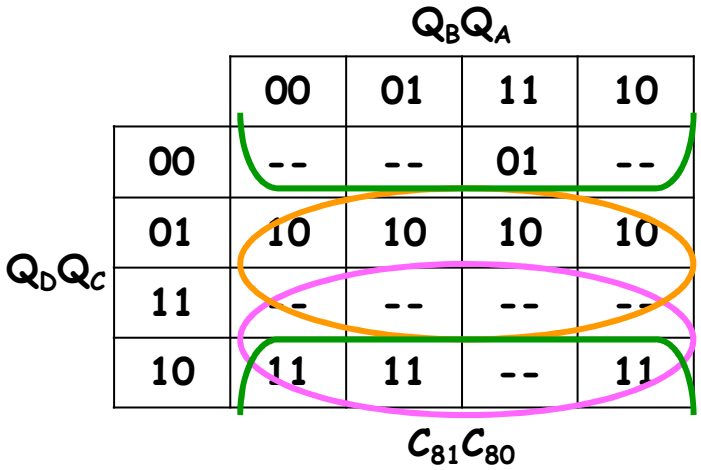
… U_4



$E_1 = H_1 \oplus a \oplus b \oplus d \oplus e \oplus g$
$E_2 = H_2 \oplus a \oplus c \oplus d \oplus f \oplus g$
$E_4 = H_4 \oplus b \oplus c \oplus d$
$E_8 = H_8 \oplus e \oplus f \oplus g$

$Q_DQ_CQ_BQ_A$	X	#1	#2	#4	#8
0000	H_1	L	-	-	-
0001	H_2	H	L	-	-
0010	H_4	H	H	L	-
0011	H_8	H	H	H	L
0100	a	U	U	H	H
0101	b	U	H	U	H
0110	c	H	U	U	H
0111	d	U	U	U	H
1000	e	U	H	H	U
1001	f	H	U	H	U
1010	g	U	U	H	U

Load / Hold / Update

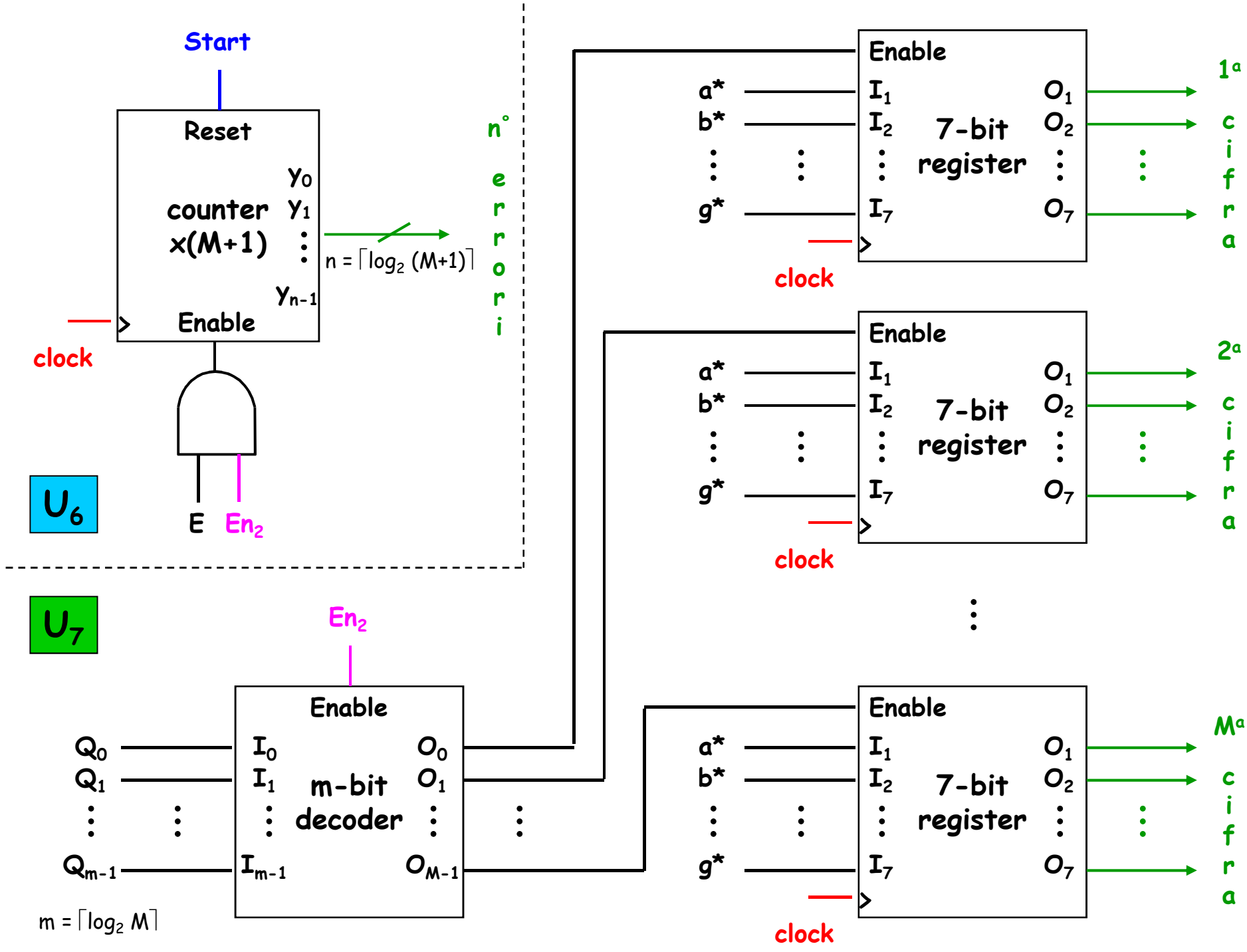


#8

analogamente #4, #2, #1

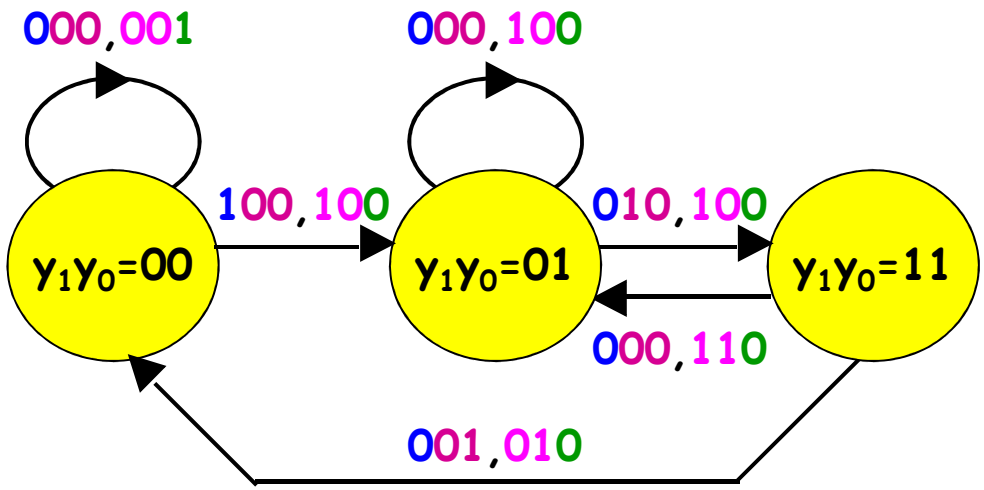
$C_{81} = Q_D + Q_C$

$C_{80} = Q_C'$


Start
Reset
counter
x(M+1)
y0
y1
yn-1
Enable
clock
n = ⌈log2 (M+1)⌉
n° errori
E
En2
U6
U7
Enable
m-bit
decoder
Q0
Q1
Qm-1
I0
I1
Im-1
O0
O1
OM-1
m = ⌈log2 M⌉
7-bit
register
a*
b*
g*
I1
I2
I7
O1
O2
O7
clock
1ª cifra
2ª cifra
Mª cifra

L'unità di controllo

Start Z_1 Z_2, En_1 En_2 End



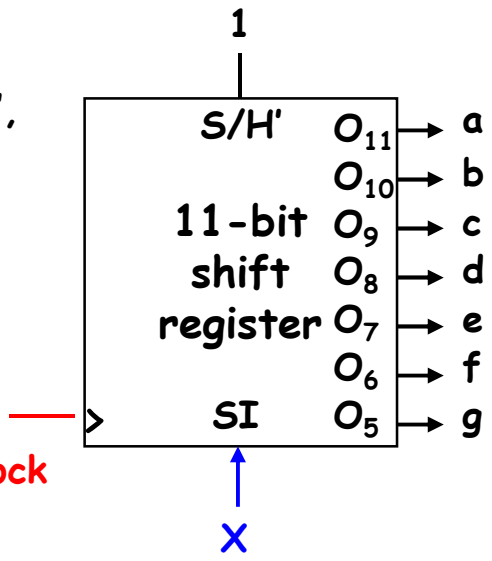
$En_1^n = y_0^{n+1} = (Start + y_0 Z_2')^n$ $\quad En_2^n = y_1^n = Z_1^{n-1}$

$End^n = (y_0' Start')^n$

idem con

$S/H' = (Q_D+Q_CQ_BQ_A)'$,

oppure:



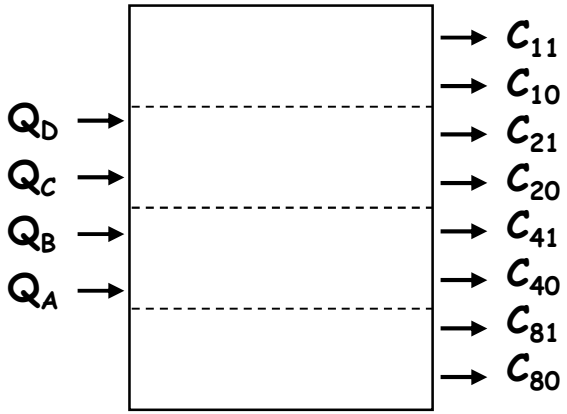
U_3

U_5 **U_6** **U_7** : idem

$Q_DQ_CQ_BQ_A$	X	#1	#2	#4	#8
0000	a	L	L	-	-
0001	b	U	H	L	-
0010	c	H	U	U	-
0011	d	U	U	U	-
0100	e	U	H	H	L
0101	f	H	U	H	U
0110	g	U	U	H	U
0111	H_1	U	H	H	H
1000	H_2	H	U	H	H
1001	H_4	H	H	U	H
1010	H_8	H	H	H	U

Load / Hold / Update

U_4 idem tranne:



$E_1 = H_1 \oplus a \oplus b \oplus d \oplus e \oplus g$

$E_2 = H_2 \oplus a \oplus c \oplus d \oplus f \oplus g$

$E_4 = H_4 \oplus b \oplus c \oplus d$

$E_8 = H_8 \oplus e \oplus f \oplus g$

#8

$C_{81} = Q_D + Q_B Q_A' + Q_A$

$C_{80} = Q_C Q_B' + Q_B Q_A'$

analogamente #4, #2, #1

Q_BQ_A / Q_DQ_C

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	--	--	--	--
01	01	11	10	11
11	--	--	--	--
10	10	10	--	11

$C_{81}C_{80}$

Problema 6

In un sistema di comunicazione digitale vengono trasferiti messaggi costituiti da al più N_{max} simboli, ciascuno individualmente rappresentato mediante k bit. Poiché in tali messaggi intervengono sovente stringhe formate da uno stesso simbolo ripetuto consecutivamente più volte, può risultare conveniente ricorrere, onde minimizzarne il tempo di trasferimento, alla seguente elementare tecnica euristica di compressione, che prevede di:

1. estendere la rappresentazione originaria di ciascun simbolo s ($s \equiv b_{k-1} \ldots b_1\ b_0$) con un ulteriore bit ($b_k$), cui è affidato il ruolo di evidenziare se il simbolo stesso è ($b_k = 1$) o meno ($b_k = 0$) seguito da altri simboli identici;
2. codificare individualmente, mediante (k + 1) bit, ogni simbolo s seguito da un simbolo diverso tramite la configurazione estesa $\sigma \equiv b_k\ b_{k-1} \ldots b_1\ b_0 = 0 \cup s$;
3. codificare cumulativamente, mediante (k + 1 + h) bit, i simboli di ogni stringa S (s, n) formata da uno stesso simbolo s ripetuto consecutivamente n volte ($2 \le n \le n_{max} = 2^h + 1$) tramite la configurazione estesa $\sigma \equiv b_k\ b_{k-1} \ldots b_1\ b_0 = 1 \cup s$ del simbolo stesso, indicata una volta sola, seguita dal valore numerico $\nu = n - 2$ ($0 \le \nu \le 2^h - 1$) rappresentato secondo il codice binario attraverso h bit;
4. partizionare ogni stringa S (s, n') formata da uno stesso simbolo s ripetuto consecutivamente $n' > n_{max}$ volte in $\lceil n' / n_{max} \rceil$ sottostringhe, di cui $\lfloor n' / n_{max} \rfloor$ costituite da n_{max} simboli e l'eventuale ultima da $n' \bmod n_{max}$ simboli, codificando poi individualmente ciascuna di esse come in precedenza (punti 2, 3) descritto.

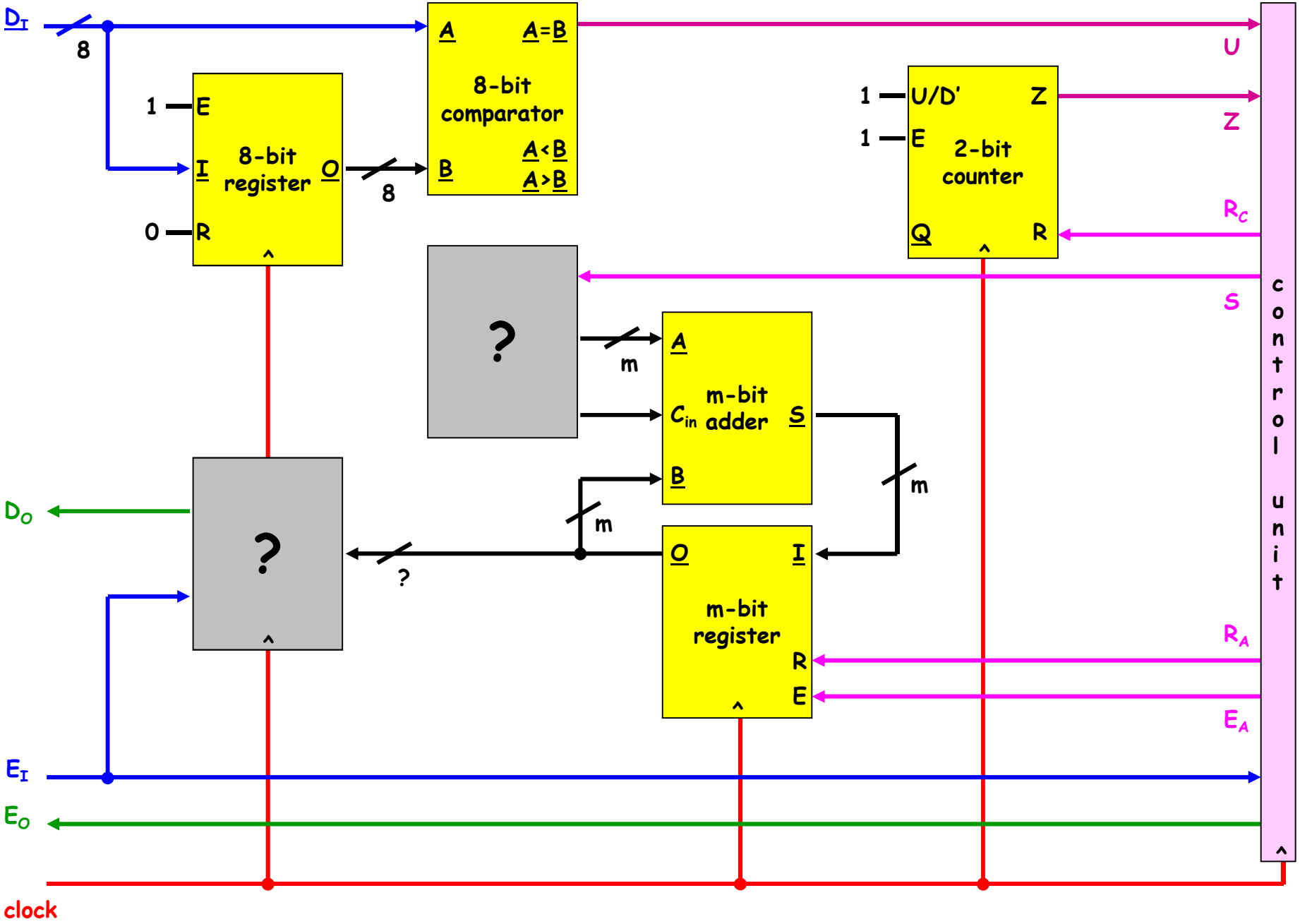
L'efficacia di tale tecnica è chiaramente correlata alla distribuzione di probabilità (frequenza relativa) secondo cui si presentano nei messaggi stringhe formate da uno stesso simbolo ripetuto più volte. Nota tale distribuzione, è possibile individuare il valore ottimale del parametro h e quindi calcolare il fattore di compressione FC di ogni messaggio, definito come rapporto fra la dimensione del messaggio codificato e quella del messaggio originario. Ad esempio, i seguenti 3 messaggi $M_1 \equiv$ abbccccccc, $M_2 \equiv$ ddddddddddde, $M_3 \equiv$ fghhhii, costituiti rispettivamente da 9, 11 e 7 simboli individualmente rappresentati mediante k = 8 bit, possono essere convenientemente rappresentati, una volta selezionato per h il valore 2, tramite 40, 31 e 40 bit soltanto ($FC(M_1) = 40/72 = 0.55$, $FC(M_2) = 31/88 = 0.35$, $FC(M_3) = 40/56 = 0.71$).

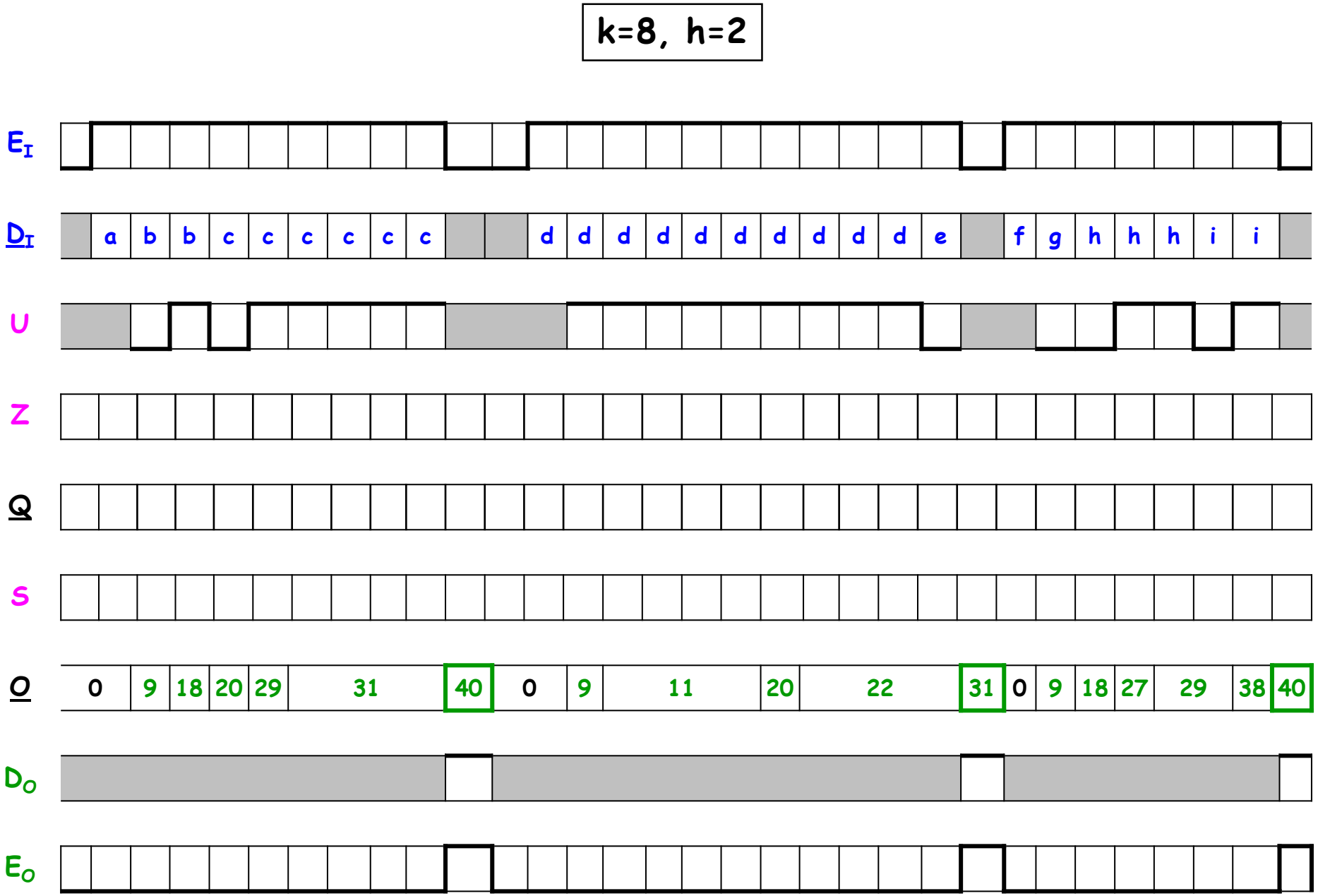
	1	2	3	4	
M_1	0 ∪ a	1 ∪ b 0	1 ∪ c 3	0 ∪ c	σ ν
M_2	1 ∪ d 3	1 ∪ d 3	0 ∪ e		σ ν
M_3	0 ∪ f	0 ∪ g	1 ∪ h 1	1 ∪ i 0	σ ν

Un sistema sequenziale sincrono, contraddistinto da 9 segnali di ingresso (E_I, $\underline{D_I} \equiv d_7, \ldots, d_1, d_0$) e da 2 segnali di uscita (E_O, D_O), è preposto ad elaborare messaggi di al più $N_{max} = 100$ simboli individualmente rappresentati tramite k = 8 bit. I simboli di ogni messaggio sono presentati sequenzialmente in ingresso al sistema, senza soluzione di continuità, attraverso i segnali $\underline{D_I}$. Il segnale E_I, attivo (livello logico 1) per N intervalli di clock, identifica la fase di presentazione in ingresso al sistema degli N simboli di un messaggio. Il sistema ha il compito di verificare, per ciascun messaggio ricevuto in ingresso, se il numero B_C di bit coinvolti nella sua rappresentazione codificata in accordo alla tecnica di compressione suddetta (h = 2) risulta essere minore del numero B_O di bit coinvolti nella sua rappresentazione originaria ($B_O = N \times k$). L'esito del processo di verifica deve essere evidenziato dal sistema tramite il segnale di uscita $D_O = (B_C < B_O)$ in corrispondenza dell'intervallo di clock immediatamente successivo a quello di ricezione dell'ultimo simbolo di un messaggio. Contestualmente il sistema deve attivare (livello logico 1, durata unitaria) il segnale di uscita E_O.

Il sistema può essere strutturato, in accordo al modello "data-path & control unit", secondo lo schema riportato in figura.

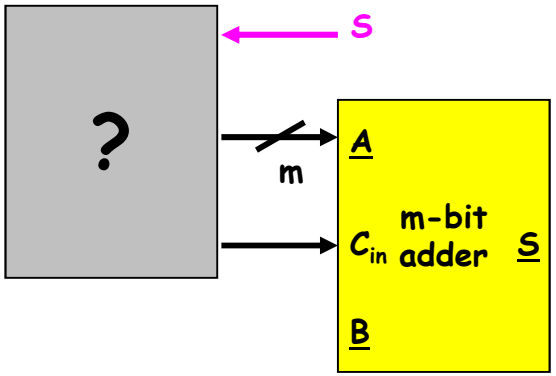
- Si completi il dimensionamento ed il progetto del data-path, utilizzando i componenti ritenuti più idonei allo scopo e motivando esplicitamente le scelte operate.
- Si esegua la sintesi dell'unità di controllo (ingressi: E_I, U, Z; uscite: R_A, E_A, S, R_C, E_O), avvalendosi, una volta completato, del diagramma temporale indicato in figura.


D_I
8
A
A=B
8-bit comparator
B
A<B
A>B
1
E
I
8-bit register
O
0
R
8
U
1
U/D'
Z
1
E
2-bit counter
Q
R
Z
R_C
S
control unit
?
m
A
m-bit adder
C_in
S
B
m
m
D_O
?
?
O
I
m-bit register
R
E
R_A
E_A
E_I
E_O
clock


k=8, h=2
E_I
D_I
a b b c c c c c c d d d d d d d d d d e f g h h h i i
U
Z
Q
S
O
0 9 18 20 29 31 40 0 9 11 20 22 31 0 9 18 27 29 38 40
D_O
E_O

$B_{C\,max} = N_{max} \times (k+1) = 900$

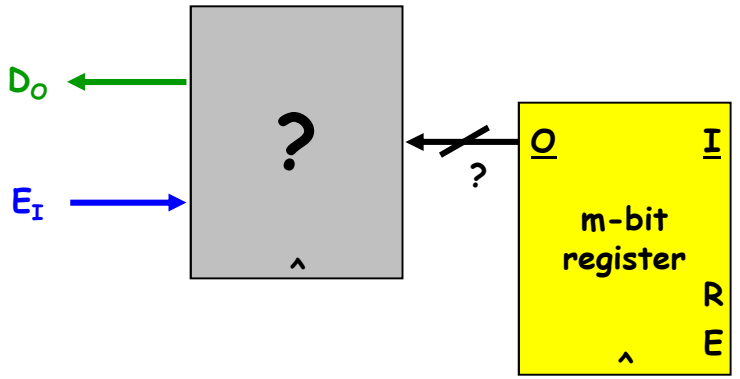
$m = \lceil \log_2 (B_{C\,max}+1) \rceil = 10$

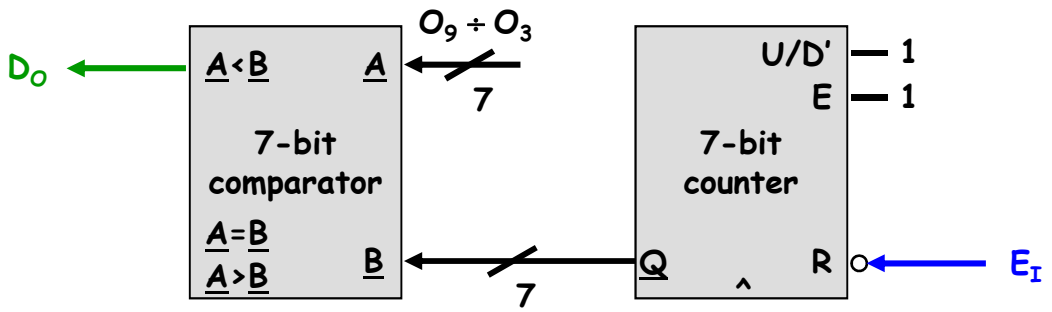


S=1: → S = B + 9 /* k+1 */ S=0: → S = B + 2 /* h */

$A_9 \div A_4 = 0,\ A_3 = S,\ A_2 = 0,\ A_1 = S',\ A_0 = 0,\ C_{in} = S$

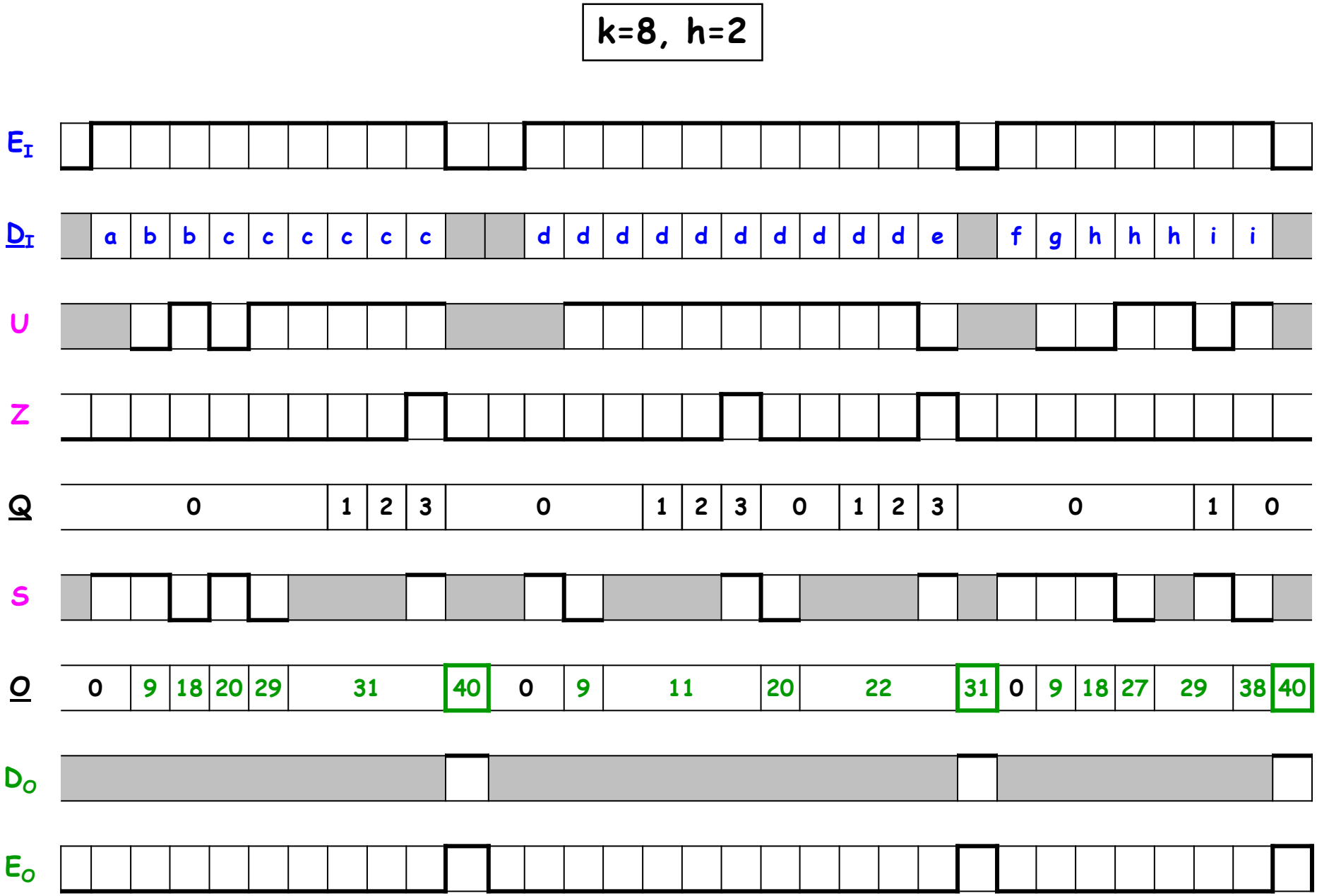
$\lceil \log_2 (N_{max}+1) \rceil = 7$





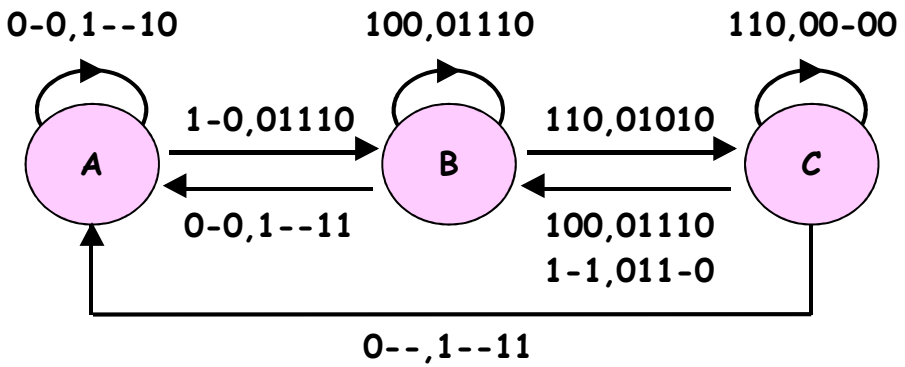
$B_C = (O_9\ O_8\ O_7\ O_6\ O_5\ O_4\ O_3\ O_2\ O_1\ O_0)_2$ $B_O = N \times 8$ /* k */ $= (Q_6\ Q_5\ Q_4\ Q_3\ Q_2\ Q_1\ Q_0\ 0\ 0\ 0)_2$

$D_O = (B_C < B_O) = ((O_9\ O_8\ O_7\ O_6\ O_5\ O_4\ O_3)_2 < (Q_6\ Q_5\ Q_4\ Q_3\ Q_2\ Q_1\ Q_0)_2)$


k=8, h=2
E_I
D_I
a b b c c c c c c d d d d d d d d d d e f g h h h i i
U
Z
Q
0 1 2 3 0 1 2 3 0 1 2 3 0 1 0
S
O
0 9 18 20 29 31 40 0 9 11 20 22 31 0 9 18 27 29 38 40
D_O
E_O

Unità di controllo

E_I U Z, R_A E_A S R_C E_O



Problema 7

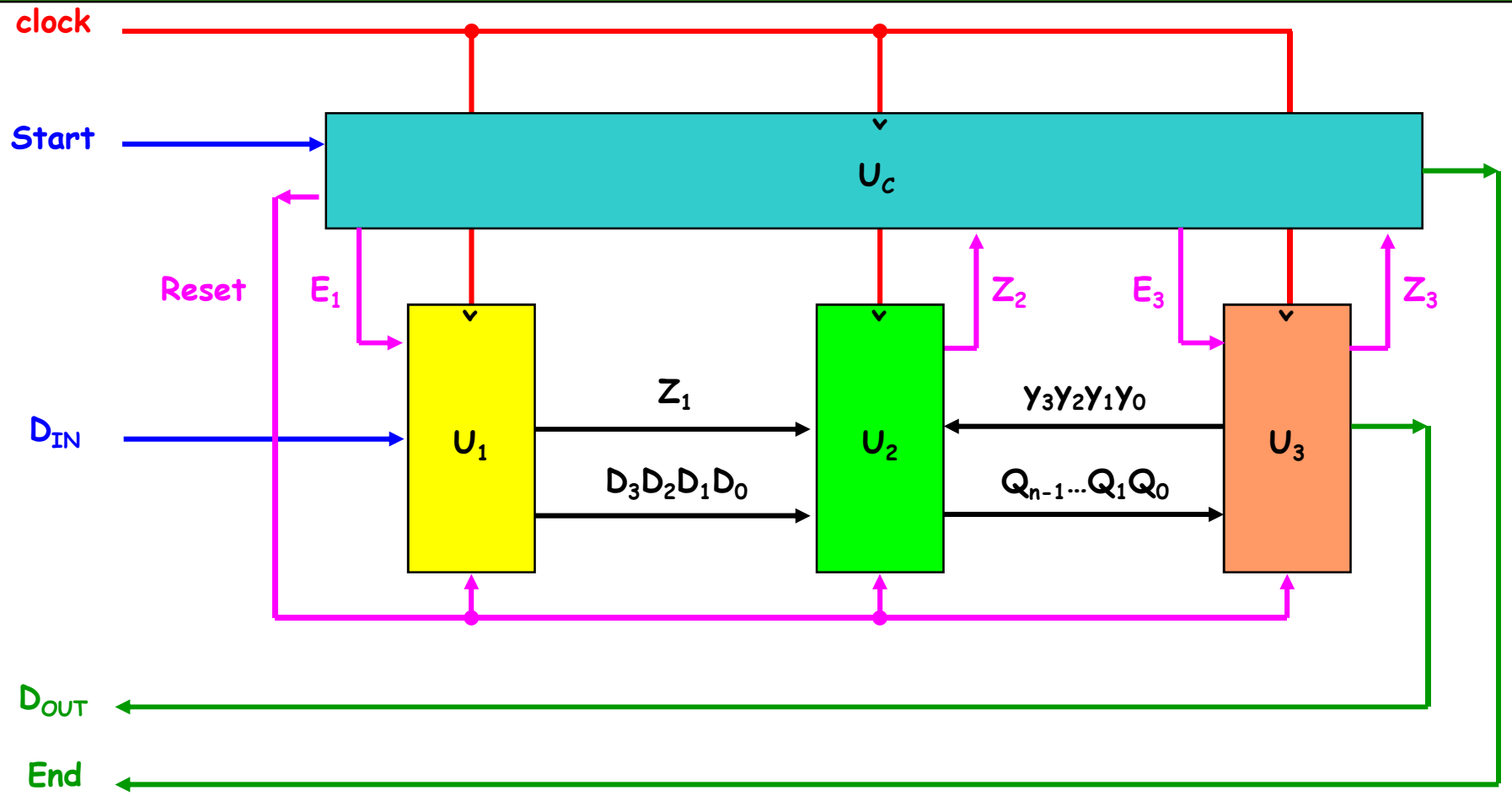
Un sistema sequenziale sincrono, caratterizzato da due segnali di ingresso (Start, D_{IN}) e da due segnali di uscita (End, D_{OUT}), tutti sincroni rispetto al clock, ha il compito di elaborare messaggi numerici costituiti da N cifre decimali incorrelate, individualmente codificate mediante un numero di bit opportunamente differenziato (codice di HUFFMAN), stante la non uniforme distribuzione della frequenza nominale con cui esse si presentano nell'ambito di ciascun messaggio:

cifra	frequenza nominale (%)	numero di bit	codifica
0	23	2	0 0
1	19	2	0 1
2	15	3	1 0 0
3	12	3	1 1 0
4	10	4	1 0 1 0
5	8	4	1 0 1 1
6	6	4	1 1 1 0
7	4	5	1 1 1 1 0
8	2	6	1 1 1 1 1 0
9	1	6	1 1 1 1 1 1

I bit rappresentativi delle N cifre costituenti ogni messaggio sono presentati in ingresso al sistema serialmente attraverso il segnale D_{IN}, senza soluzione di continuità ed a partire, per ogni cifra, dal bit più significativo (MSB). Il segnale Start, attivo a livello logico 1 e di durata unitaria, identifica l'intervallo di presentazione in ingresso del primo bit (ovvero del MSB della prima cifra) di ciascun messaggio.

Il sistema ha il compito di verificare, contando il numero di volte che ognuna delle 10 cifre si presenta nell'ambito di un messaggio, se l'effettiva frequenza di occorrenza di ciascuna di esse coincide con la corrispondente frequenza nominale. Al termine del processo di elaborazione di un messaggio, il sistema deve attivare il segnale di uscita End (livello logico 1, durata unitaria) e contestualmente indicare tramite il segnale D_{OUT} se la condizione suddetta è soddisfatta ($D_{OUT} = 1$) o meno ($D_{OUT} = 0$) per tutte le 10 cifre.

Il sistema deve essere strutturato in accordo al modello "data-path & control unit" secondo lo schema indicato in figura, rispettando le specifiche progettuali nel seguito delineate per quanto concerne l'unità di controllo U_C e le tre unità funzionali U_1, U_2, U_3 preposte all'elaborazione dei messaggi.

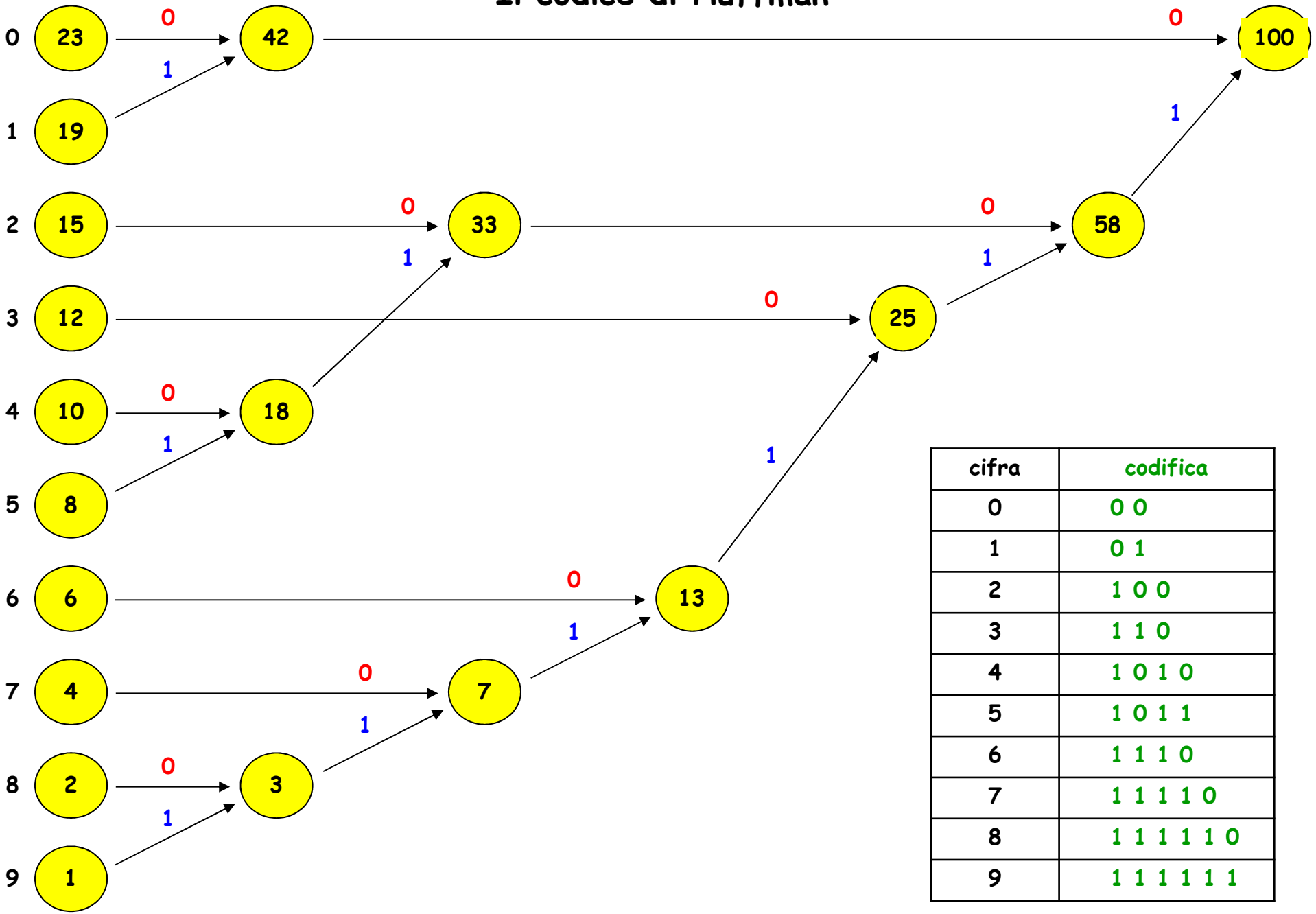


- L'unità U_1, allorché abilitata da U_C (segnale di ingresso $E_1 = 1$), opera il riconoscimento delle varie cifre presenti in un messaggio, procedendo ad attivare il segnale di uscita Z_1 in corrispondenza dell'intervallo di ricezione dell'ultimo bit di ciascuna di esse ed indicandone contestualmente la relativa rappresentazione secondo il codice BCD mediante i 4 segnali D_3, D_2, D_1, D_0.
- L'unità U_2 opera, in base al segnale Z_1 generato da U_1, il conteggio del numero di cifre riconosciute nell'ambito di un messaggio, provvedendo ad attivare il segnale di uscita Z_2 allorché tale numero coincide con N. Tale unità è adibita anche a gestire il numero di volte che ciascuna delle 10 cifre si presenta nell'ambito di un messaggio. Allo scopo si avvale di 10 contatori x(N+1), opportunamente abilitati al conteggio in base al valore dei segnali Z_1, D_3, D_2, D_1, D_0 generati da U_1. Attraverso una rete di multiplazione ed in base alla configurazione dei segnali di selezione y_3, y_2, y_1, y_0 generati da U_3, il valore di uno specifico contatore è reso disponibile in uscita tramite i segnali Q_{n-1}, …, Q_1, Q_0 ($n = \lceil \lg_2(N+1) \rceil$).
- L'unità U_3, allorché abilitata da U_C (segnale di ingresso $E_3 = 1$), procede a verificare se, nell'ambito del messaggio ricevuto e per ognuna delle 10 cifre, l'effettiva frequenza di occorrenza (f_e) coincide con la corrispondente frequenza nominale (f_n). Per ogni cifra, il valore di f_e è derivabile dall'unità U_2 tramite i segnali di selezione y_3, y_2, y_1, y_0, mentre il valore di f_n è localmente disponibile in una memoria del tipo ROM. Il completamento del processo di verifica è evidenziato con l'attivazione del segnale di uscita Z_3, mentre l'esito, positivo o negativo, con il valore logico 1 o 0, rispettivamente, del segnale D_{OUT}.

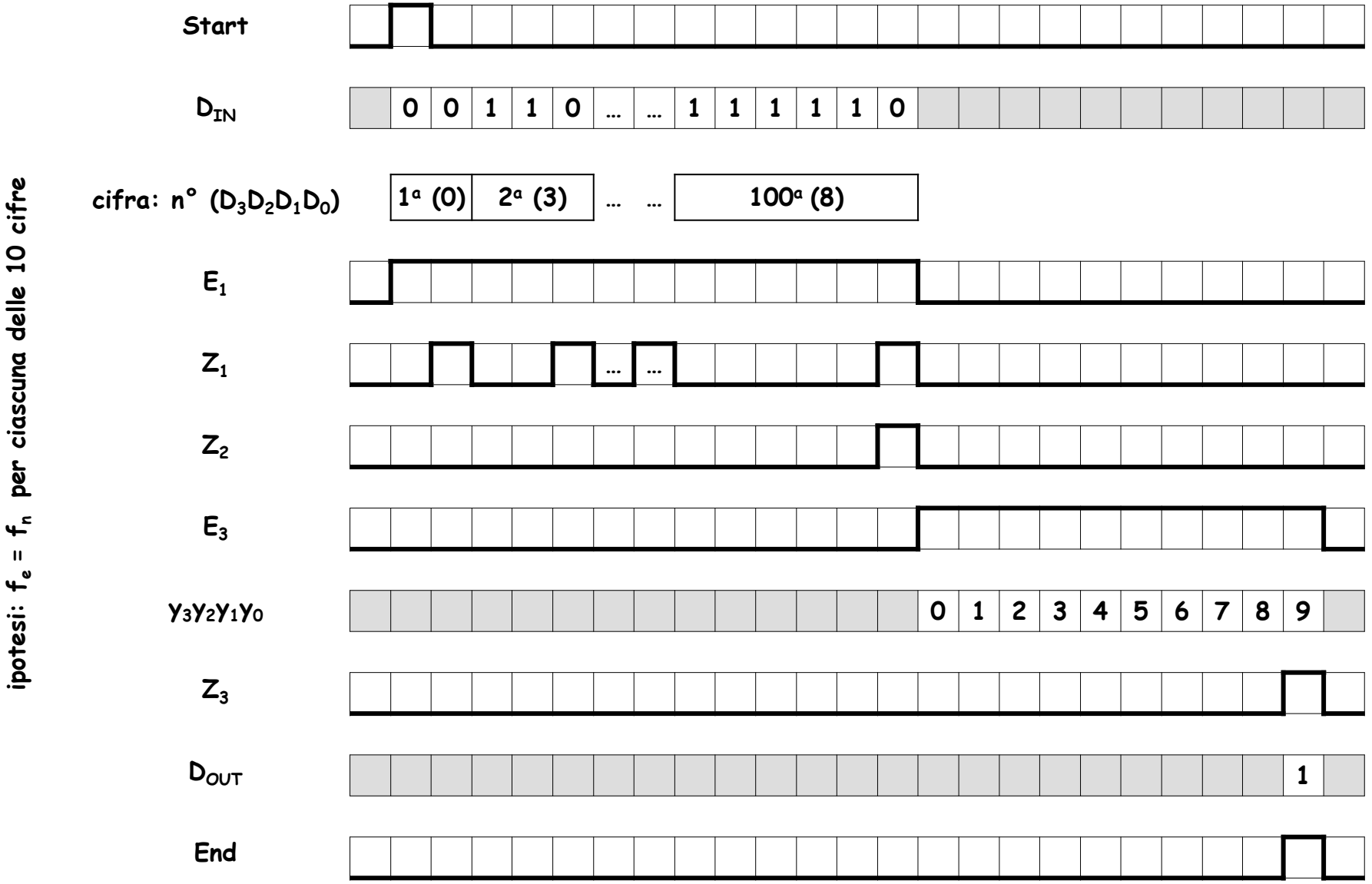
Nell'ipotesi che ciascun messaggio sia costituito da N = 100 cifre:

1) si determini il fattore di compressione dei messaggi derivante dall'applicazione del codice di HUFFMAN rispetto alla semplice rappresentazione di ciascuna delle N cifre mediante un codice irridondante a 4 bit;
2) si formalizzi il comportamento dell'unità U_1 in termini di automa a stati finiti;
3) si completi il progetto delle unità U_2 e U_3, avvalendosi dei componenti ritenuti più idonei allo scopo e motivando esplicitamente tutte le scelte operate;
4) si definisca il grafo degli stati dell'unità di controllo.

Il codice di Huffman



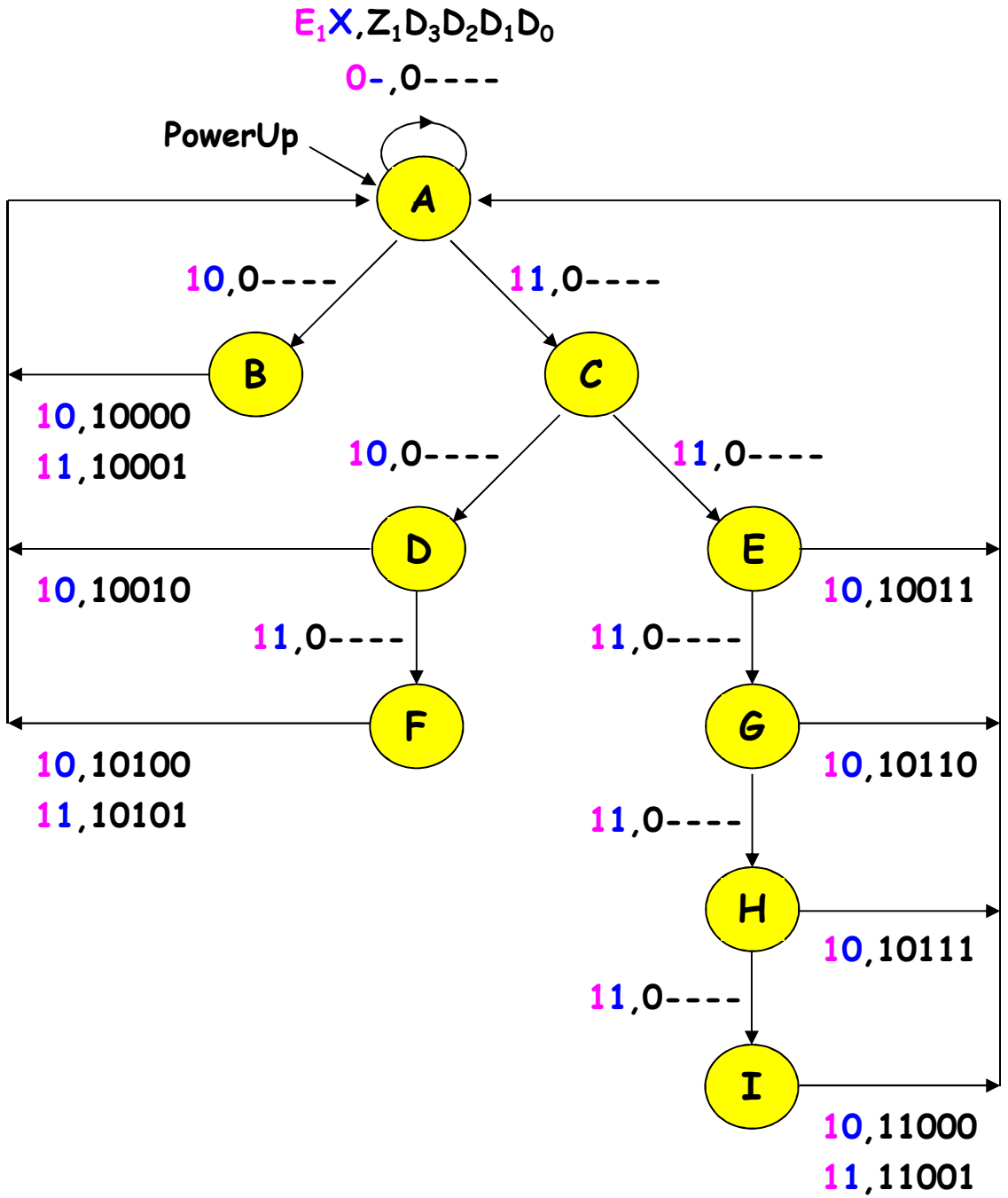
cifra	codifica
0	0 0
1	0 1
2	1 0 0
3	1 1 0
4	1 0 1 0
5	1 0 1 1
6	1 1 1 0
7	1 1 1 1 0
8	1 1 1 1 1 0
9	1 1 1 1 1 1

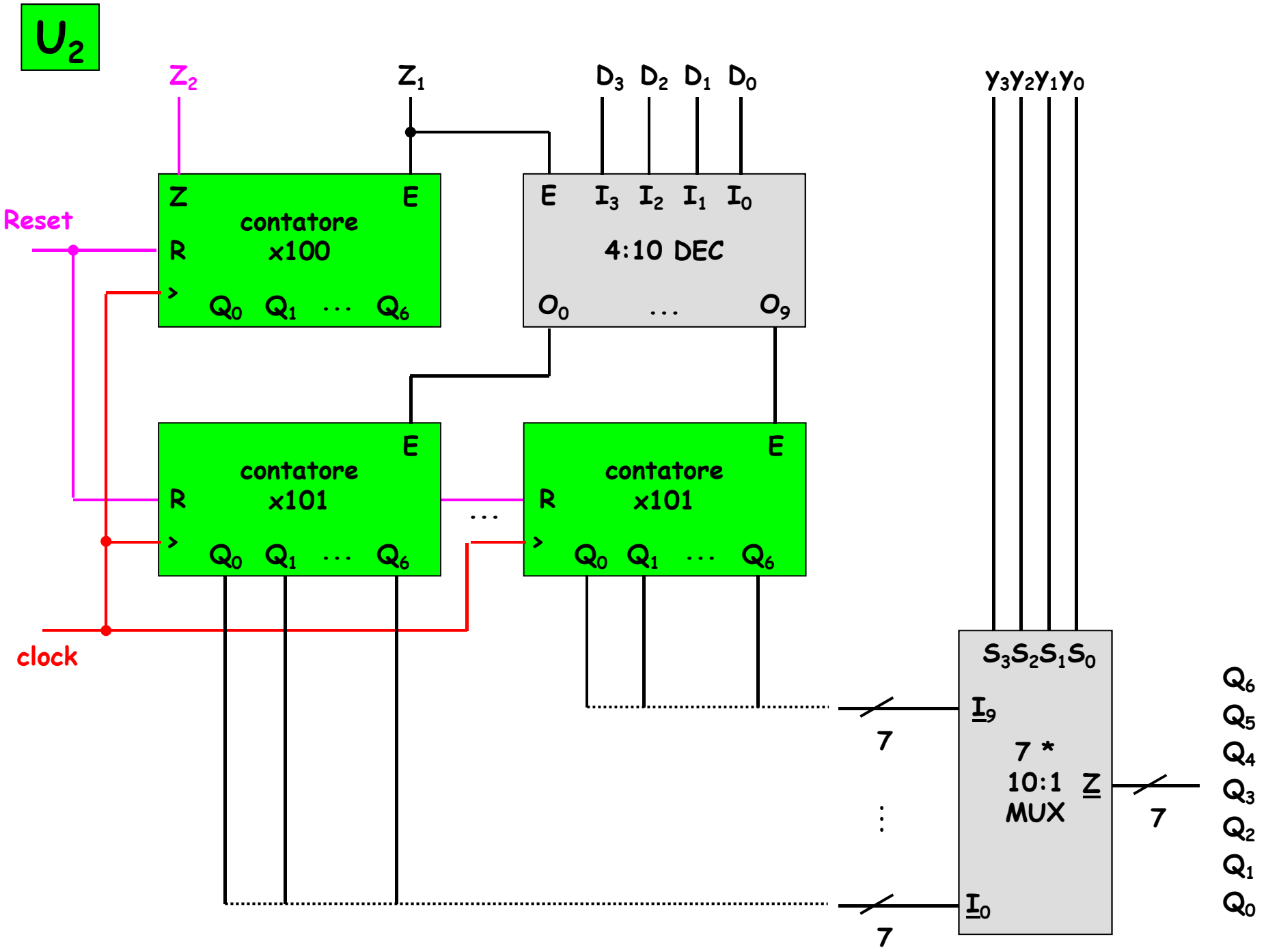


$$\text{Fattore di compressione} = \frac{(0.23 * 2 + 0.19 * 2 + 0.15 * 3 + \ldots + 0.01 * 6)\, N}{4\, N} = 0.75$$

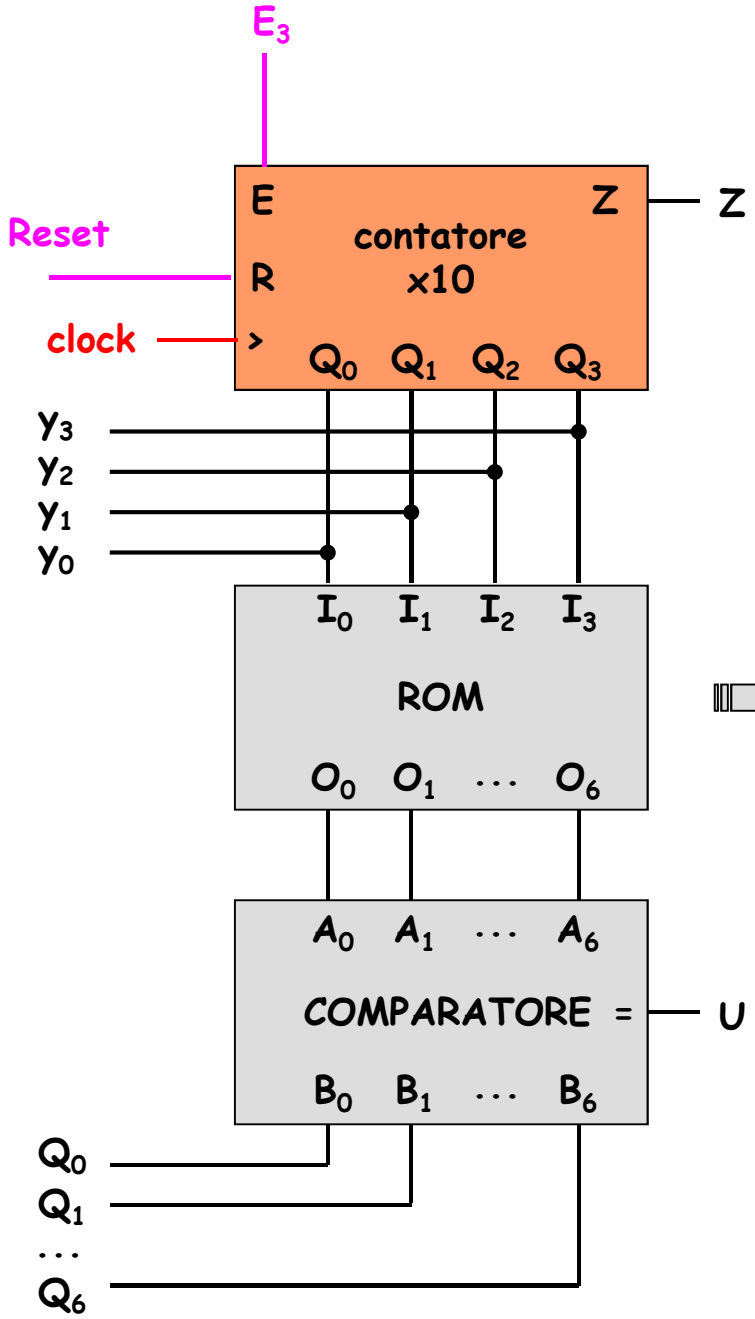
U_1

cifra	codifica
0	0 0
1	0 1
2	1 0 0
3	1 1 0
4	1 0 1 0
5	1 0 1 1
6	1 1 1 0
7	1 1 1 1 0
8	1 1 1 1 1 0
9	1 1 1 1 1 1

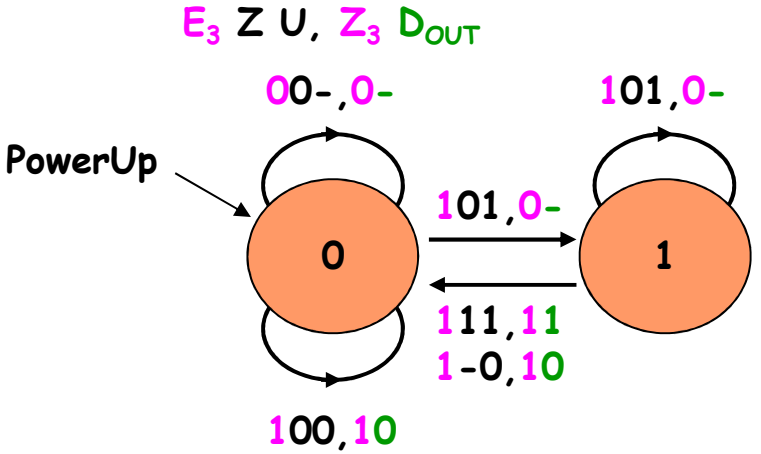



U2
Z2
Z1
D3 D2 D1 D0
Y3Y2Y1Y0
Reset
Z
E
contatore
x100
R
>
Q0 Q1 ... Q6
E I3 I2 I1 I0
4:10 DEC
O0 ... O9
E
contatore
x101
R
>
Q0 Q1 ... Q6
clock
S3S2S1S0
I9
7 *
10:1
MUX
Z
I0
7
7
Q6
Q5
Q4
Q3
Q2
Q1
Q0

U_3



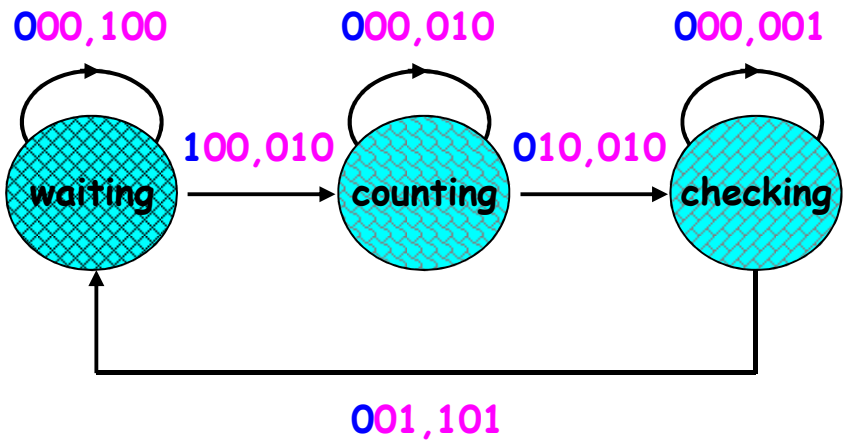
notifica dell'esito della verifica: ASAP



	I_3	I_2	I_1	I_0	O_6	O_5	O_4	O_3	O_2	O_1	O_0	f_n
0	0	0	0	0	0	0	1	0	1	1	1	23
1	0	0	0	1	0	0	1	0	0	1	1	19
2	0	0	1	0	0	0	0	1	1	1	1	15
3	0	0	1	1	0	0	0	1	1	0	0	12
4	0	1	0	0	0	0	0	1	0	1	0	10
5	0	1	0	1	0	0	0	1	0	0	0	8
6	0	1	1	0	0	0	0	0	1	1	0	6
7	0	1	1	1	0	0	0	0	1	0	0	4
8	1	0	0	0	0	0	0	0	0	1	0	2
9	1	0	0	1	0	0	0	0	0	0	1	1

U_C

Start Z_2 Z_3, R E_1 E_3



End = Z_3

Problema 8

Nell'ambito dei sistemi digitali per la compressione dei dati senza perdita di informazione sono spesso utilizzati i codici di Elias, il cui obiettivo è la rappresentazione efficiente dei numeri interi positivi, qualunque sia il relativo campo di variabilità, attraverso stringhe di bit non di lunghezza fissa, stabilita in base al valore massimo da rappresentare, bensì di lunghezza differenziata, correlata al numero di bit significativi che intervengono nella rappresentazione binaria di ciascun valore.

In particolare:

- il codice γ di Elias codifica un qualunque valore intero $N \geq 1$ premettendo alla sua rappresentazione binaria in forma minima N_{2min} un numero di "0" uguale al numero di bit che intervengono in N_{2min} dopo il bit più significativo;
- il codice δ di Elias codifica un qualunque valore intero $N \geq 1$ premettendo alla sua rappresentazione binaria in forma minima N_{2min}, senza il bit più significativo, la configurazione che identifica secondo il codice γ il numero di bit che intervengono in N_{2min}.

Un sistema digitale sincrono, caratterizzato da due segnali di ingresso START, X (entrambi sincroni rispetto al clock) e da undici segnali di uscita END, Z_9, Z_8, Z_7, Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0, ha il compito di elaborare numeri rappresentati secondo il codice δ compresi nel range [1-999], fornendo per ciascuno di essi in uscita la corrispondente configurazione binaria a dieci bit.

Ciascun dato è applicato in ingresso al sistema serialmente, a partire dal bit più significativo, attraverso il segnale X. L'intervallo di presentazione del primo bit di ciascun dato è evidenziato al sistema dall'esterno tramite il valore 1 del segnale START.

Ciascun risultato deve essere fornito in uscita dal sistema in parallelo tramite i segnali Z_9 (bit più significativo), …, Z_1, Z_0. L'intervallo di presentazione di ciascun risultato, coincidente con l'intervallo di ricezione del bit meno significativo del dato corrispondente, deve essere segnalato dal sistema tramite il valore 1 dell'uscita END.

N	N_{2min}	N_γ	N_δ
1	1	1	1
2	10	010	0100
3	11	011	0101
4	100	00100	01100
5	101	00101	01101
6	110	00110	01110
7	111	00111	01111
8	1000	0001000	00100000
9	1001	0001001	00100001
10	1010	0001010	00100010
11	1011	0001011	00100011
12	1100	0001100	00100100
13	1101	0001101	00100101
14	1110	0001110	00100110
15	1111	0001111	00100111
16	10000	000010000	001010000
32	100000	00000100000	0011000000
64	1000000	0000001000000	00111000000
128	10000000	00000010000000	00010000000000
256	100000000	000000010000000	00010010000000
512	1000000000	0000000001000000000	00010100000000000
1024	10000000000	00000000010000000000	00010110000000000

numero di bit: $1 + \lfloor \log_2 N \rfloor$ $1 + 2 \lfloor \log_2 N \rfloor$ $1 + \lfloor \log_2 N \rfloor + 2 \lfloor \log_2 (\lfloor \log_2 N \rfloor + 1) \rfloor$

Il sistema deve essere articolato in due unità funzionali disposte in cascata, come indicato in figura.

La prima unità, ricevendo in ingresso i due segnali START e X, ha il compito di operare la conversione serie-parallelo, nonché la trascodifica dal codice γ al codice binario, del preambolo che identifica, per ciascun dato applicato in ingresso, il numero dei bit corrispondenti alla rappresentazione del valore.

Tale informazione, codificata tramite i quattro segnali Z_3^* (bit più significativo), Z_2^*, Z_1^*, Z_0^*, è fornita in ingresso alla seconda unità, la quale ha semplicemente il compito di operare la conversione serie-parallelo dei bit rappresentativi del valore.

Questi ultimi bit, incluso il bit più significativo non previsto dal codice δ, vengono trasferiti dalla prima alla seconda unità in serie attraverso il segnale X*. L'intervallo di presentazione del primo bit (il più significativo) è notificato dalla prima alla seconda unità tramite l'attivazione (valore logico 1) del segnale START*.

Ciascuna unità deve essere strutturata in accordo al modello "data-path & control unit".

Il data-path della prima unità è costituito da:

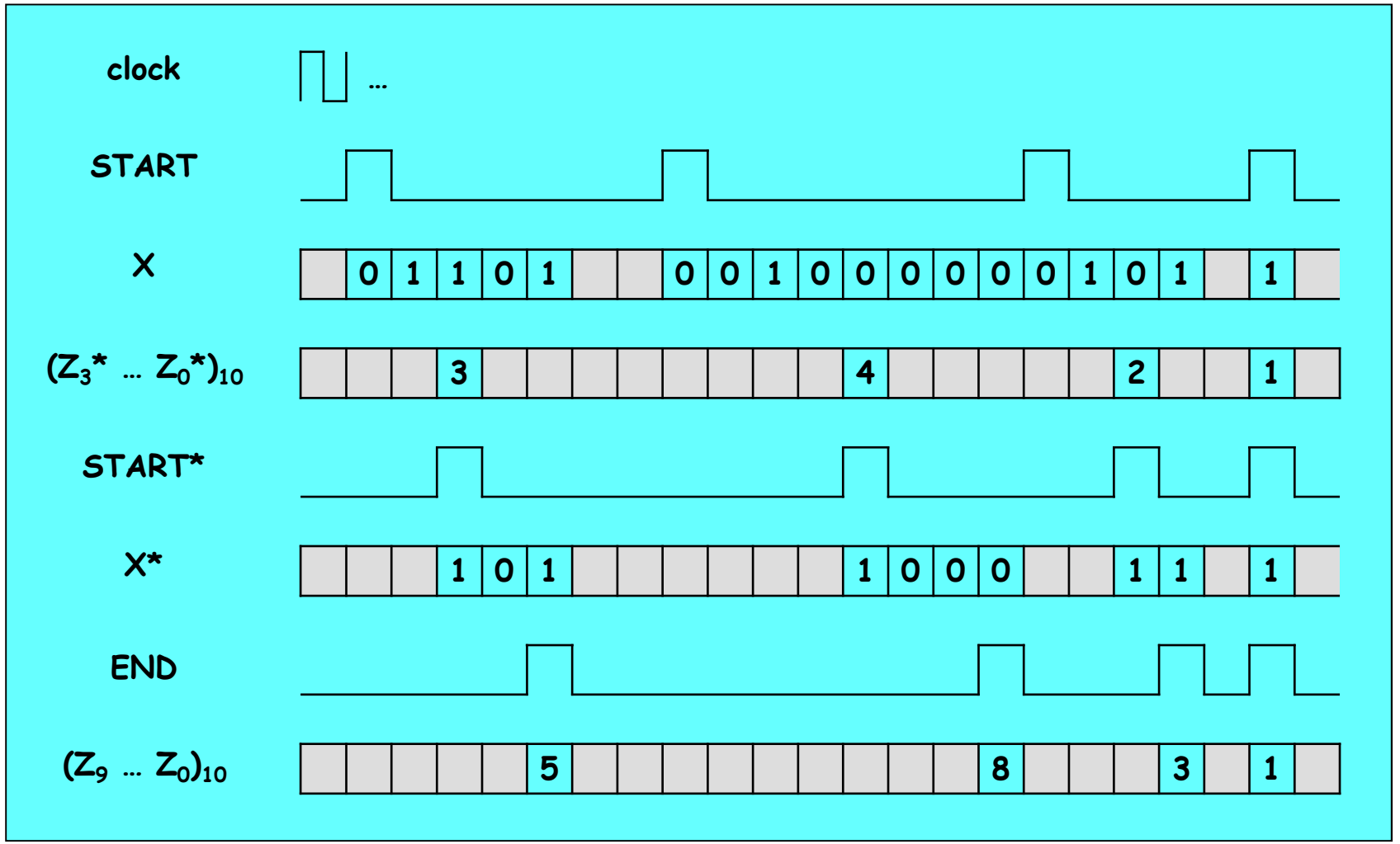
- un contatore binario bidirezionale $\times 4$, dotato dei comandi E (ENABLE), U/D' (UP/DOWN') ed uscita ZERO, attiva (livello logico 1) allorché lo stato interno del contatore è zero;
- un registro a scorrimento di 3 bit, dotato del comando S/R' (SHIFT/RESET').

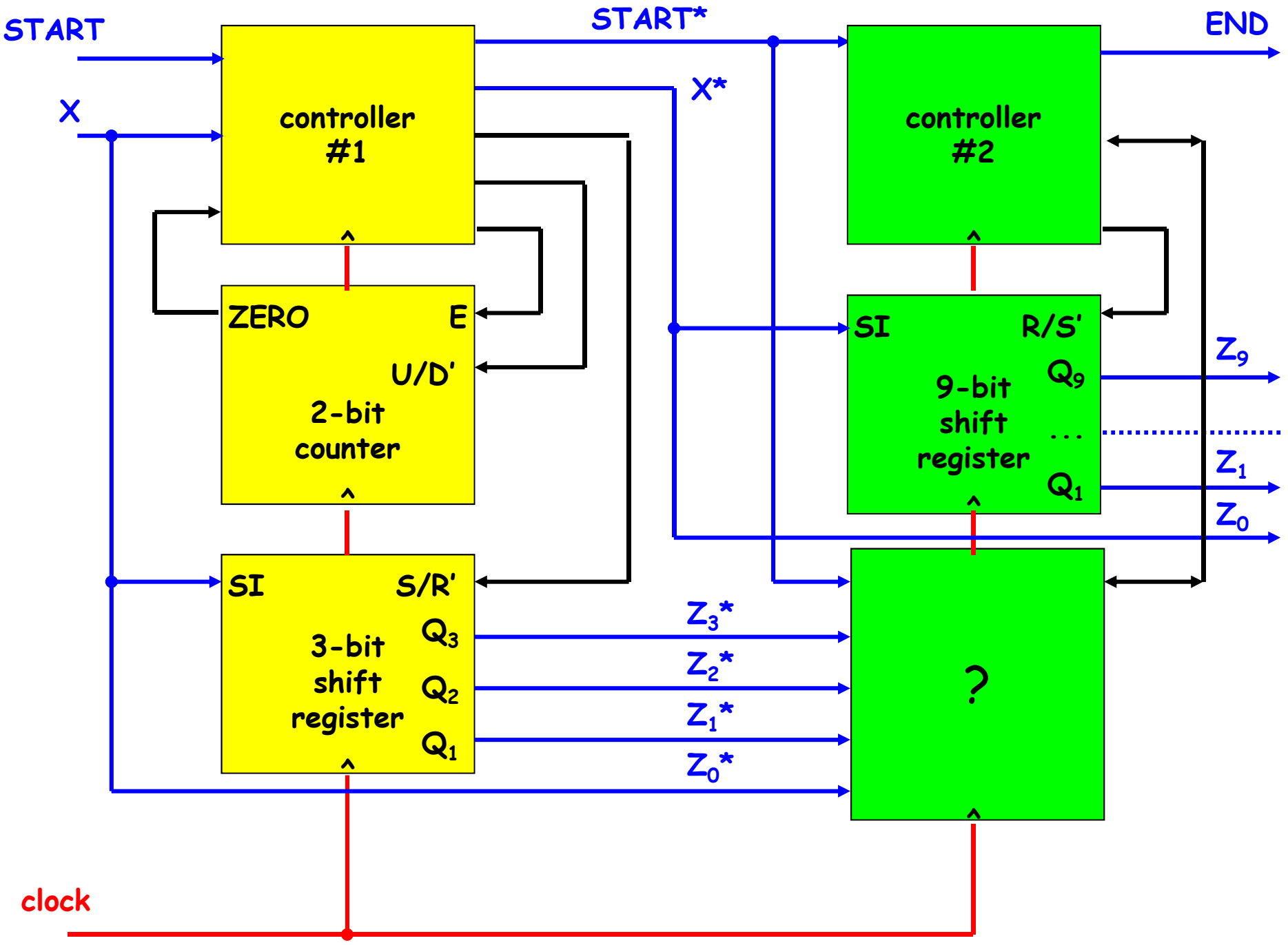
Il data-path della seconda unità comprende:

- un registro a scorrimento di 9 bit, dotato del comando R/S' (RESET/SHIFT');
- una rete sequenziale cui è affidato il compito di identificare l'intervallo di elaborazione dell'ultimo bit di ciascun dato applicato in ingresso.

Si definisca il grafo degli stati, caratterizzato dagli ingressi START, X, ZERO e dalle uscite E, U/D', S/R', START*, X*, che modella il comportamento del controller della prima unità.

Si completi la struttura del data-path della seconda unità e si esegua la sintesi del relativo controller in termini di grafo degli stati.

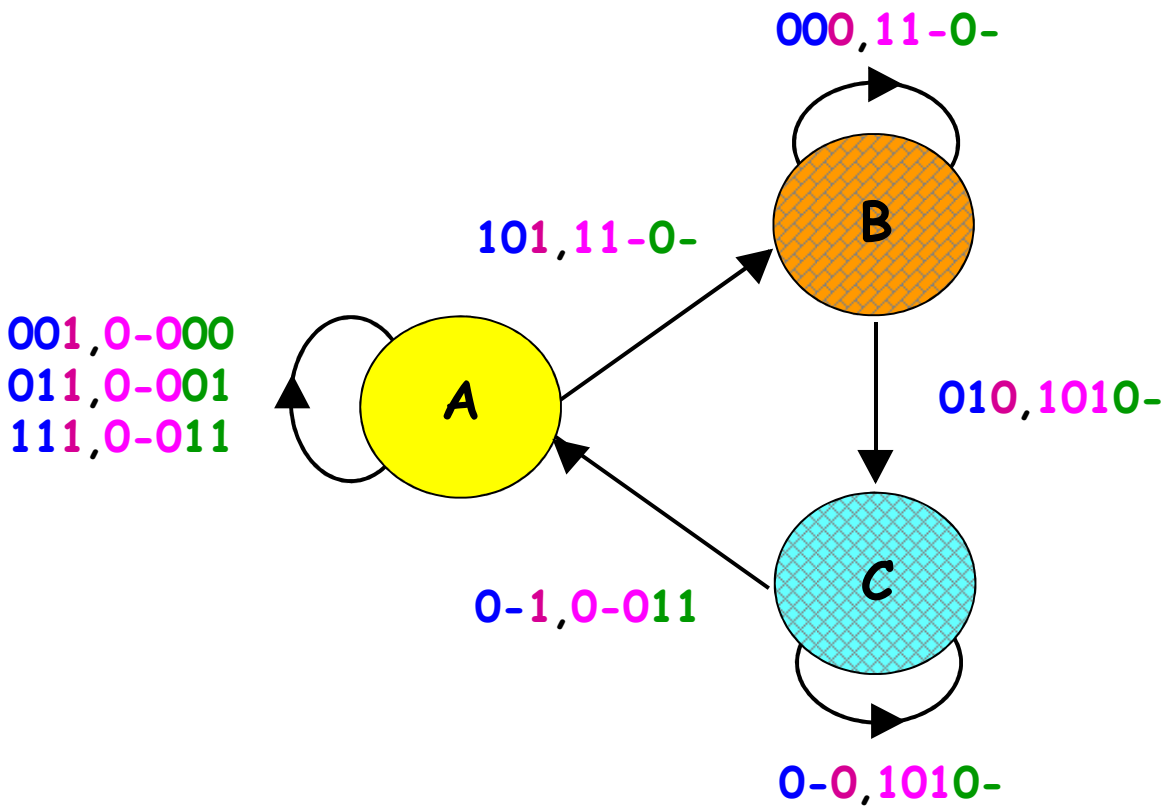

clock
...
START
X
0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1
$(Z_3^* \ldots Z_0^*)_{10}$
3 4 2 1
START*
X*
1 0 1 1 0 0 0 1 1 1
END
$(Z_9 \ldots Z_0)_{10}$
5 8 3 1


START
X
controller #1
START*
X*
END
controller #2
ZERO
E
U/D'
2-bit counter
SI
R/S'
9-bit shift register
Q9
...
Q1
Z9
Z1
Z0
SI
S/R'
3-bit shift register
Q3
Q2
Q1
Z3*
Z2*
Z1*
Z0*
?
clock

Progetto della prima unità di controllo

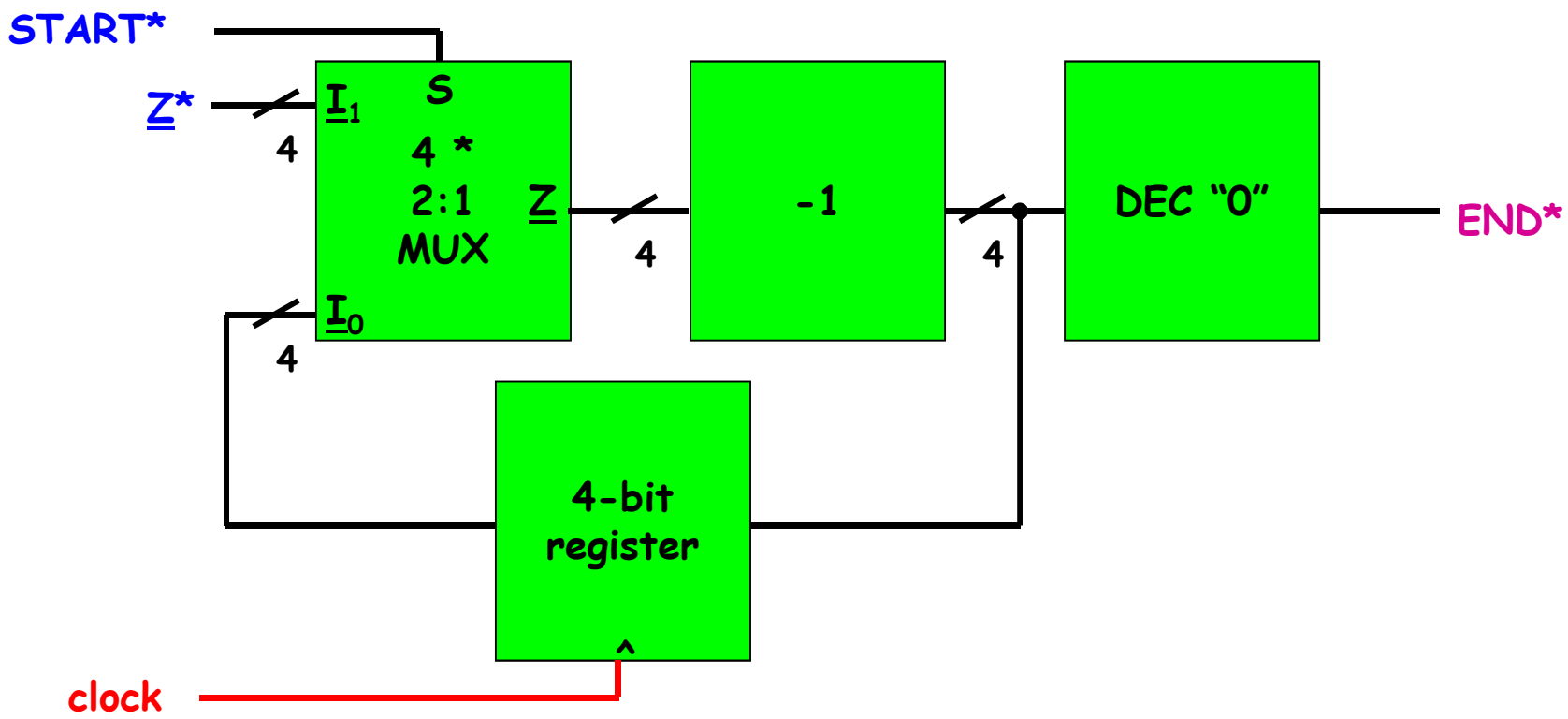
Il grafo degli stati

Legenda: START X ZERO, E U/D' S/R' START* X*



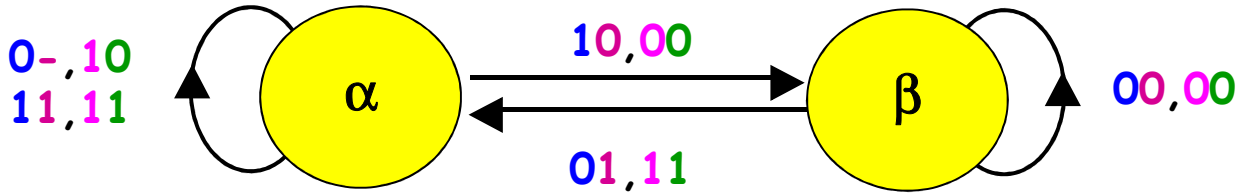
Progetto della seconda unità

Il completamento del data-path



Il grafo degli stati della control unit

Legenda: START* END*, R/S' END




Register
Transfer
Language

RTL: Register Transfer Language

R1 ← R2	il contenuto di R2 viene trasferito in R1
R1 ← *op* R2	il risultato della operazione *op* sul contenuto di R2 viene trasferito in R1
R1 ← R2 *op* R3	il risultato della operazione *op* sui contenuti di R2 e R3 viene trasferito in R1
label : R1 ← R2	ad uno statement può essere associata una "etichetta"
goto *label*	il linguaggio prevede istruzioni di salto incondizionato
if *condition* then ... else ...	il linguaggio prevede istruzioni di salto condizionato

N.B.: due o più statement sono eseguiti
contemporaneamente se separati da "," (ad es.: R1 ← R2, R3 ← R4),
sequenzialmente se separati da ";" (ad es.: R1 ← R2; R3 ← R4).

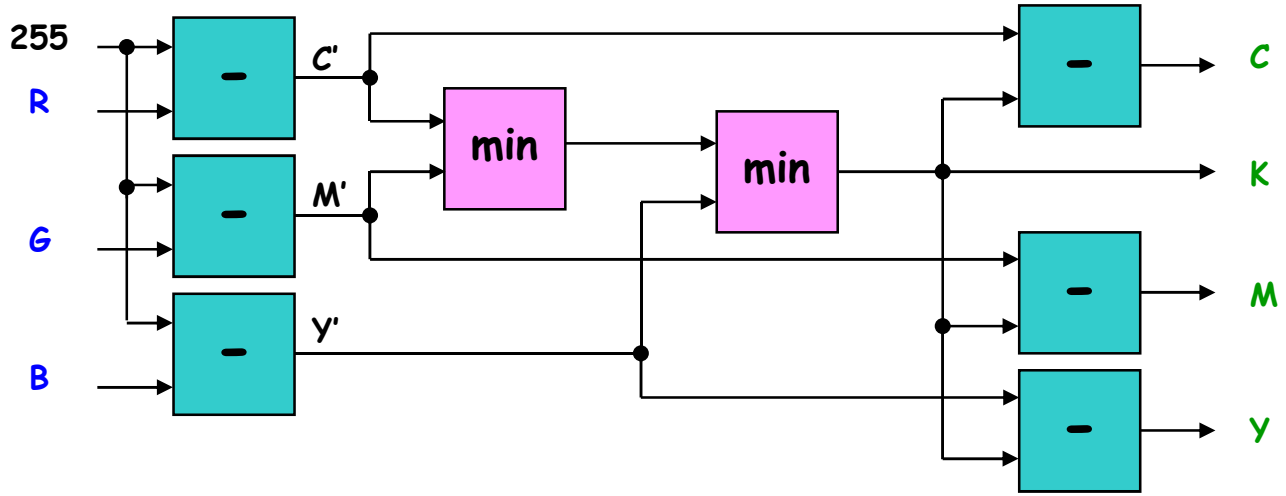
Problema 1

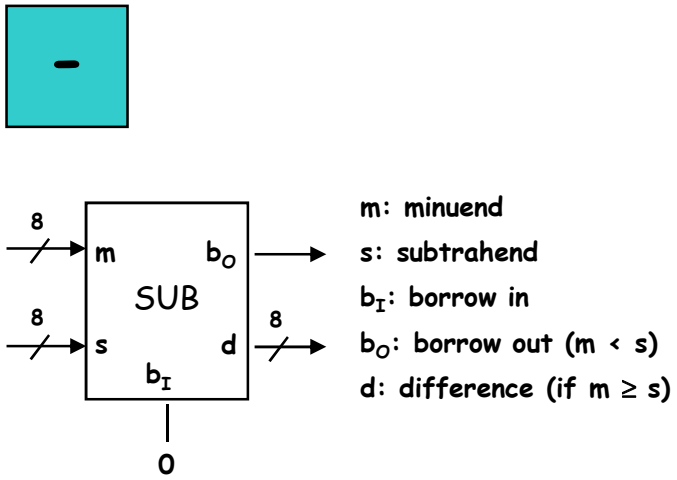
Nell'ambito di alcuni sistemi preposti all'elaborazione di immagini, come ad esempio le stampanti a colori a getto di inchiostro, si pone il problema di convertire la rappresentazione cromatica di ciascun pixel dallo spazio dei colori RGB (Red, Green, Blue) a quello KCMY (blacK, Cyan, Magenta, Yellow), essendo ciascuna componente di colore R, G, B, K, C, M, Y del tipo "8-bit unsigned integer".

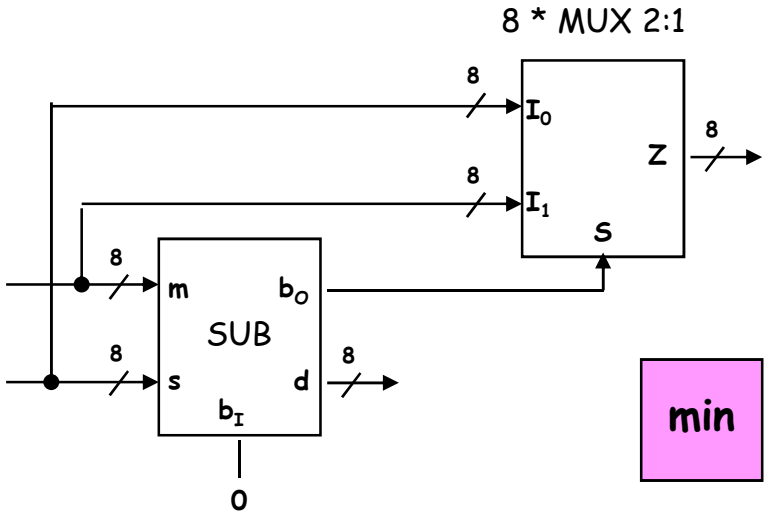
Il processo di conversione comporta le seguenti elaborazioni:

$$C' = 255 - R$$
$$M' = 255 - G$$
$$Y' = 255 - B$$
$$K = \min(C', M', Y')$$
$$C = C' - K$$
$$M = M' - K$$
$$Y = Y' - K$$

Ai fini della realizzazione digitale del convertitore, è possibile ricorrere o ad un sistema combinatorio comprendente *otto* sottrattori …

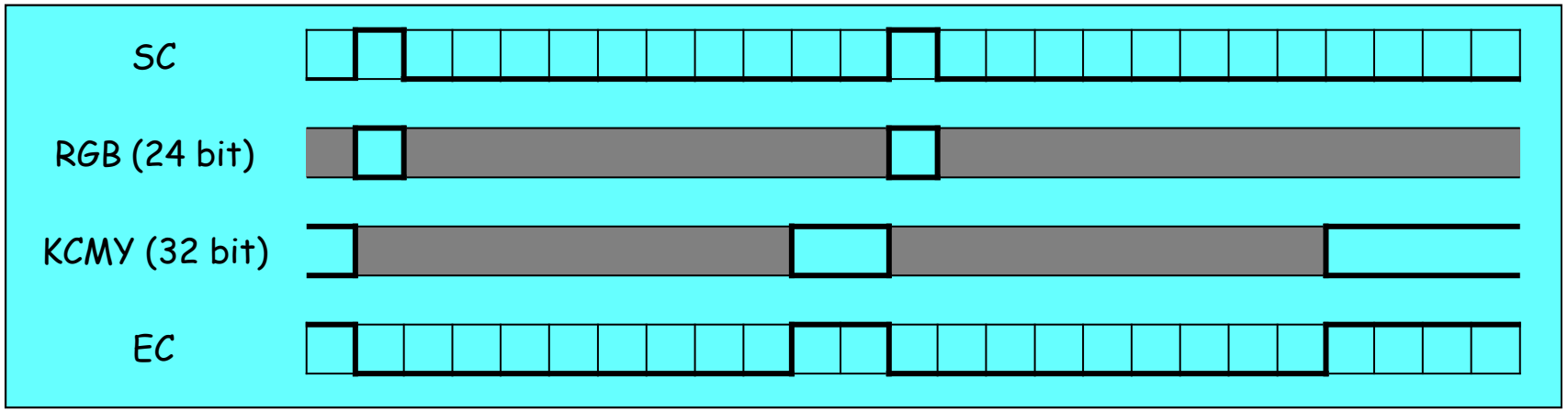




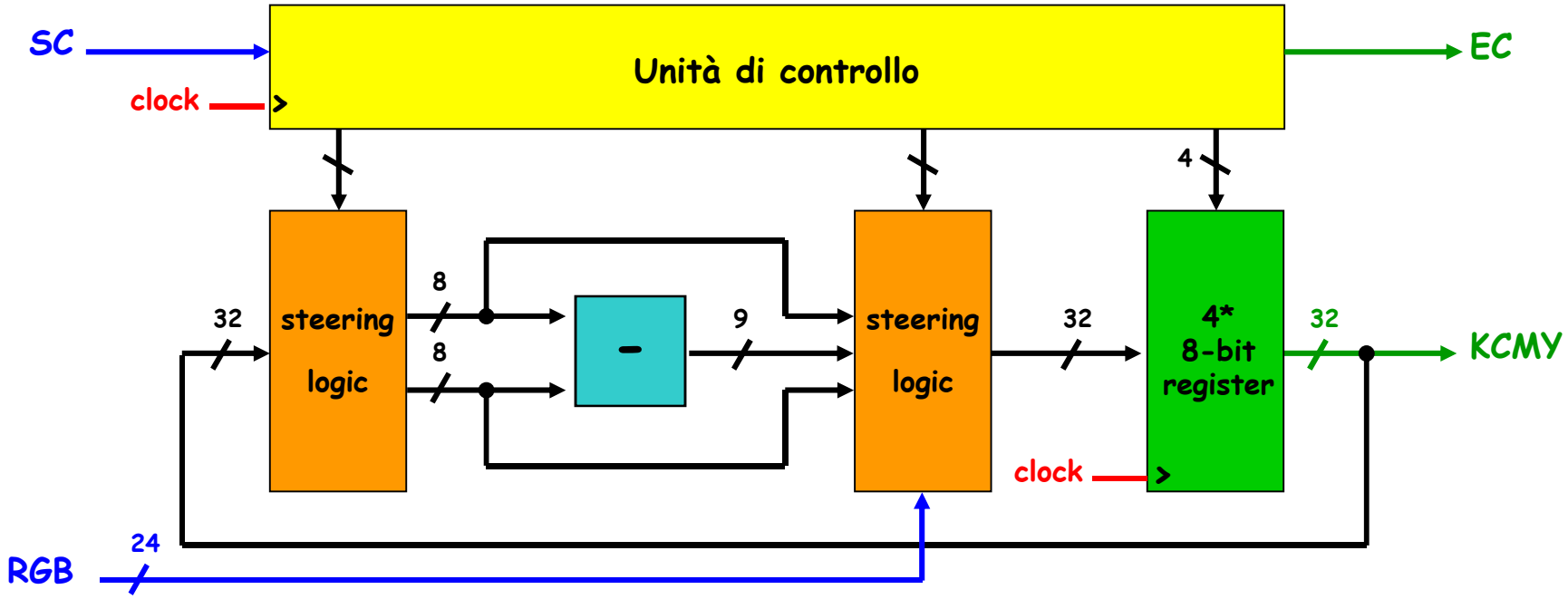


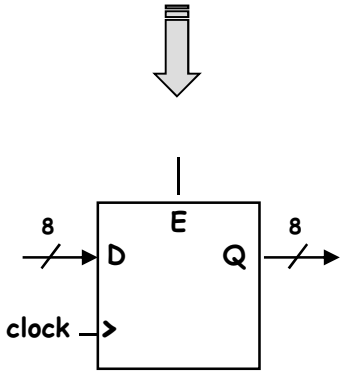
… oppure ad un sistema sequenziale basato sull'impiego di un *unico* sottrattore.

Con riferimento a quest'ultimo approccio, si esegua la sintesi di un sistema sequenziale che, ricevendo in ingresso il valore delle 3 componenti R, G, B di un pixel, sia in grado di generare in uscita il valore delle corrispondenti 4 componenti K, C, M, Y. Il sistema prevede 2 ulteriori segnali SC (StartConversion) e EC (EndConversion), adibiti a identificare con il valore logico 1, rispettivamente, l'intervallo di presentazione in ingresso di un nuovo dato e l'intervallo in cui il corrispondente risultato è disponibile in uscita. Il segnale SC ha durata unitaria, il segnale EC deve rimanere attivo fino alla successiva attivazione di SC.



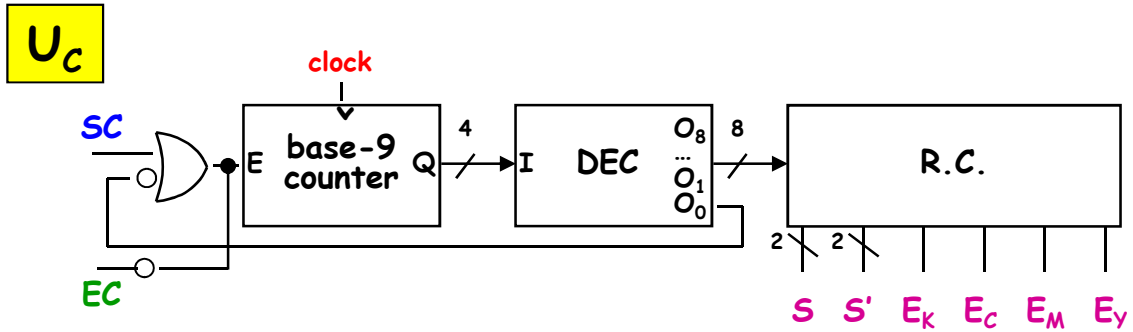
Il sistema deve essere strutturato in accordo al modello "data-path & control unit" secondo lo schema indicato in figura, avvalendosi dei componenti ritenuti più idonei allo scopo e motivando esplicitamente tutte le scelte progettuali operate.

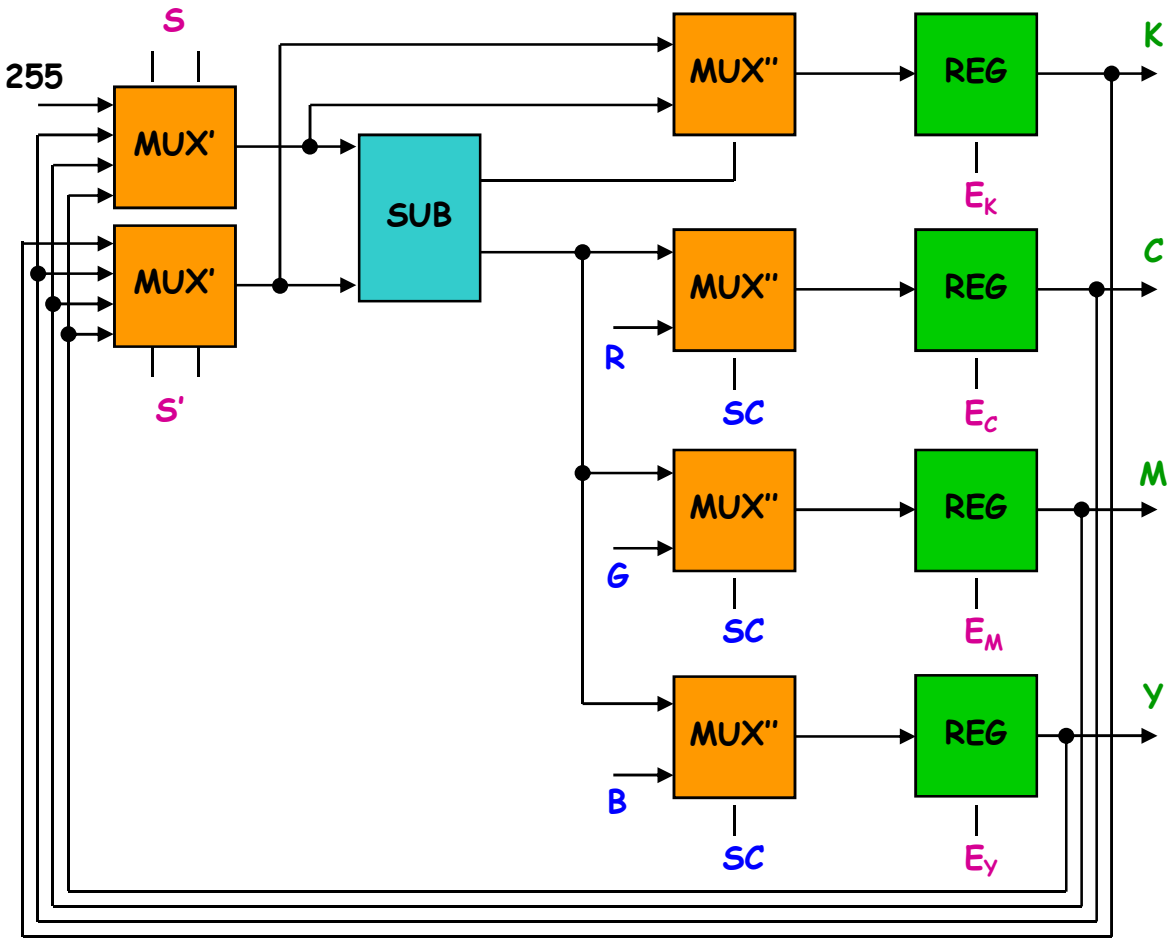




Descrizione RTL del processo di elaborazione

```
0:  if (not SC)
    then EC ← 1, K ← K, C ← C, M ← M, Y ← Y, goto 0;        /* hold outputs */
    else EC ← 0, C ← R, M ← G, Y ← B;                       /* sample inputs */

1:  EC ← 0, m ← 255, s ← C, C ← d, M ← M, Y ← Y;           /* C ← 255 - C */

2:  EC ← 0, C ← C, m ← 255, s ← M, M ← d, Y ← Y;           /* M ← 255 - M */

3:  EC ← 0, C ← C, M ← M, m ← 255, s ← Y, Y ← d;           /* Y ← 255 - Y */

4:  EC ← 0, C ← C, M ← M, Y ← Y, m ← C, s ← M,             /* K ← min (C, M) */
    if (b_O)
    then K ← m;
    else K ← s;

5:  EC ← 0, C ← C, M ← M, Y ← Y, m ← Y, s ← K,             /* K ← min (Y, K) */
    if (b_O)
    then K ← m;
    else K ← s;

6:  EC ← 0, K ← K, m ← C, s ← K, C ← d, M ← M, Y ← Y;      /* C ← C - K */

7:  EC ← 0, K ← K, C ← C, m ← M, s ← K, M ← d, Y ← Y;      /* M ← M - K */

8:  EC ← 0, K ← K, C ← C, M ← M, m ← Y, s ← K, Y ← d,      /* Y ← Y - K */
    goto 0;
```





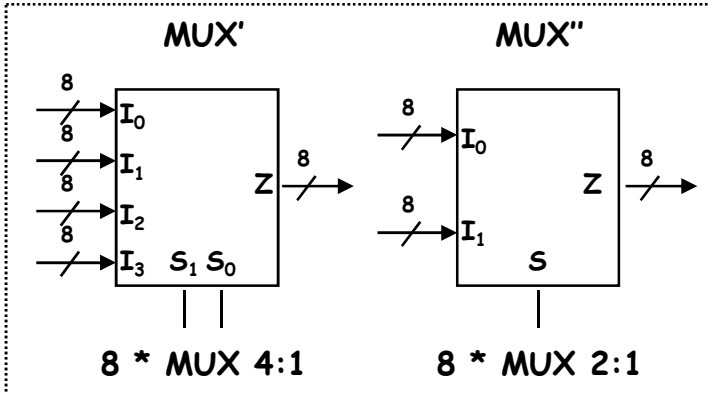
step	S	S'	E_k	E_C	E_M	E_y
0	-	-	0	SC	SC	SC
1	0	1	-	1	0	0
2	0	2	-	0	1	0
3	0	3	-	0	0	1
4	1	2	1	0	0	0
5	3	0	1	0	0	0
6	1	0	0	1	0	0
7	2	0	0	0	1	0
8	3	0	0	0	0	1

$S_1 = O_5 + O_7 + O_8$ $\quad E_K = O_4 + O_5$

$S_0 = O_4 + O_5 + O_6 + O_8$ $\quad E_C = SC + O_1 + O_6$

$S_1' = O_2 + O_3 + O_4$ $\quad E_M = SC + O_2 + O_7$

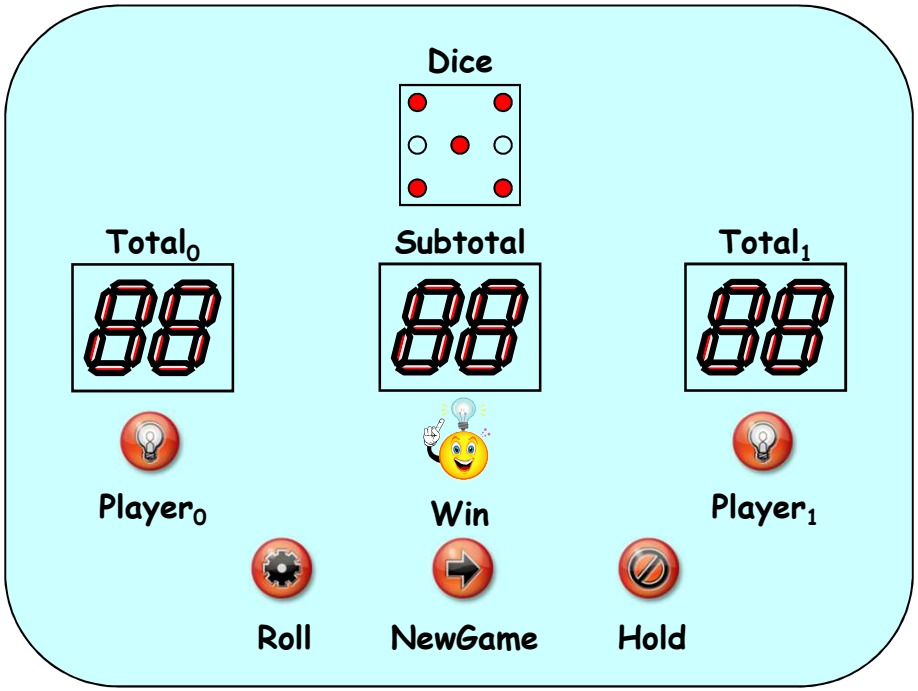
$S_0' = O_1 + O_3$ $\quad E_y = SC + O_3 + O_8$



Problema 2

Il gioco del "PIG", utilizzato nelle scuole elementari per consentire agli alunni di acquisire una intuitiva percezione del concetto di probabilità, prevede che ciascuno degli N ($N \geq 2$) giocatori partecipanti ad un incontro, a rotazione, effettui il lancio di un dado anche più volte consecutivamente, accumulando via via un punteggio parziale uguale alla somma dei valori corrispondentemente esibiti dal dado. Tale punteggio parziale è tuttavia prontamente azzerato ed il turno del giocatore ritenuto concluso senza alcun incremento del punteggio totale conseguito negli eventuali precedenti turni, se a seguito di un lancio il dado esibisce il valore 1. Al fine di prevenire questo indesiderabile evento, tanto più probabile quanto maggiore è il numero di lanci effettuati, un giocatore può decidere di capitalizzare il punteggio parziale fino al momento conseguito nell'ambito di un turno passando spontaneamente la mano al giocatore successivo. Vince l'incontro il giocatore che per primo raggiunge un punteggio complessivo ≥ 100.

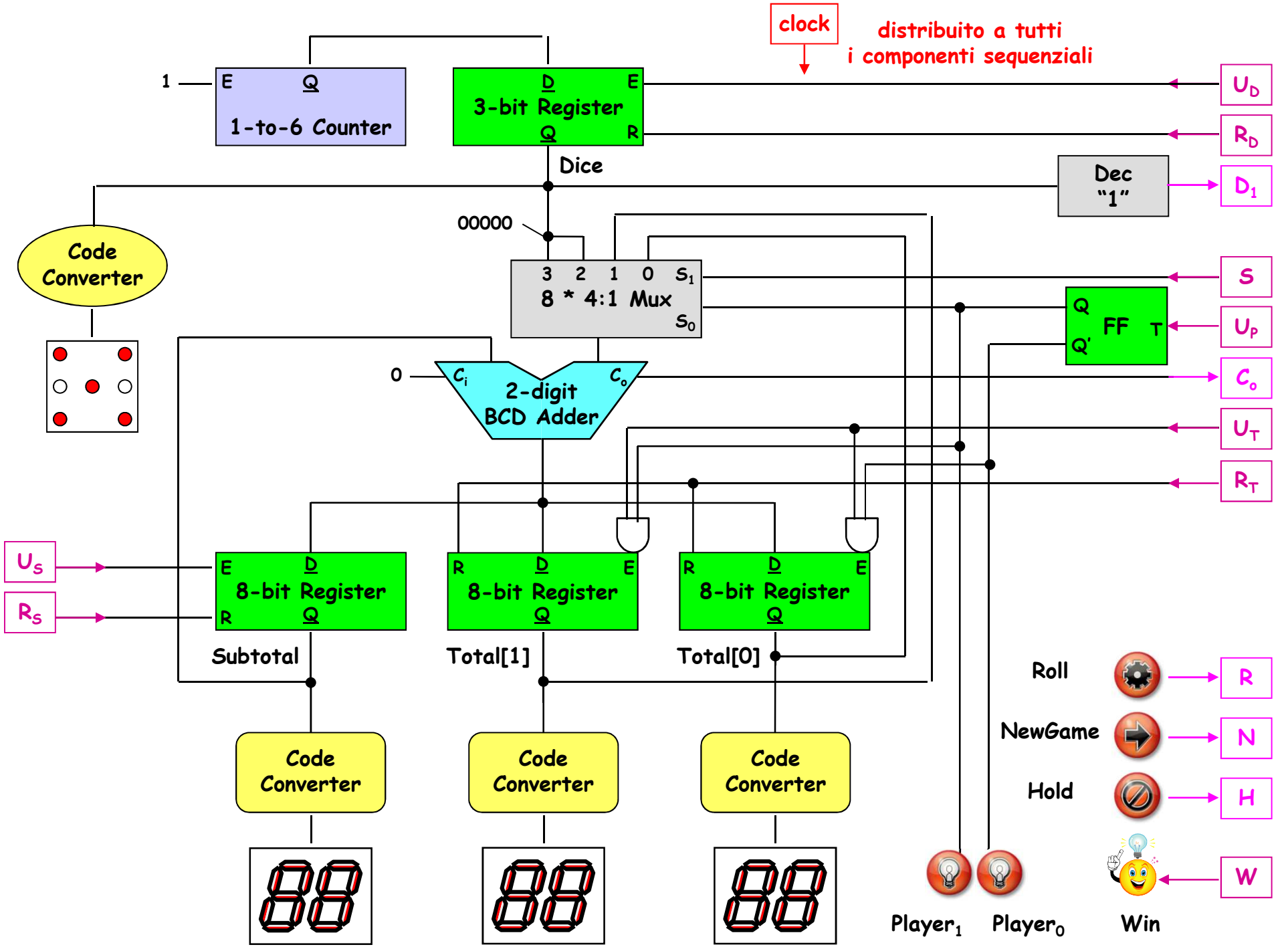
Con riferimento al caso di N = 2 giocatori, la console del "PIG", evidenziata in figura, comprende:



- due display numerici a 7-segmenti ("$Total_0$", "$Total_1$"), preposti alla visualizzazione del punteggio totale conseguito nei precedenti turni da ciascun giocatore;
- due led ("$Player_0$", "$Player_1$"), adibiti ad indicare in maniera mutuamente esclusiva quale sia il giocatore correntemente di turno ("Player");
- tre pulsanti ("Roll", "Hold", "NewGame"), tramite i quali il "Player" può, rispettivamente, richiedere il lancio del dado, notificare la propria decisione di passare la mano ritenendosi soddisfatto del punteggio parziale accumulato nel turno corrente, avviare un nuovo incontro in caso di vincita;
- un display a matrice di punti ("Dice"), preposto alla visualizzazione del punteggio conseguito dal "Player" a seguito del lancio del dado;
- un display numerico a 7-segmenti ("Subtotal"), preposto alla visualizzazione del punteggio parziale correntemente accumulato dal "Player" nell'ambito del turno;
- un led ("Win"), tramite il quale viene segnalata al "Player", una volta raggiunto un punteggio complessivo ≥ 100, la vincita dell'incontro.

Si esegua il progetto di un sistema sequenziale sincrono, strutturato in accordo al modello "data-path & control unit" riportato in figura, in grado di gestire il gioco del "PIG" secondo le modalità delineate, più precisamente definite dalla seguente descrizione in linguaggio RTL:

```
0:  Win ← 0, Dice ← 0, Subtotal ← 0, Total[0] ← 0, Total[1] ← 0,
    if (NewGame)
    then goto 0; /* else goto 1 */;
1:  if (Roll)
    then Dice ← Counter /* , goto 2 */;
    else  goto 1;
2:  if (not Roll)
    then if (Dice = 1)
            then Subtotal ← 0, Dice ← 0, Player ← Next (Player), goto 1;
            else  Subtotal ← Subtotal + Dice, Dice ← 0 /* , goto 3 */;
    else  goto 2;
```


clock
distribuito a tutti
i componenti sequenziali
1
E
Q
1-to-6 Counter
D
E
3-bit Register
Q
R
Dice
U_D
R_D
Dec
"1"
D_1
Code
Converter
00000
3 2 1 0 S_1
8 * 4:1 Mux
S_0
S
Q
Q'
FF
T
U_P
0
C_i
C_o
2-digit
BCD Adder
C_o
U_T
R_T
U_S
R_S
E
D
8-bit Register
R
Q
Subtotal
Total[1]
Total[0]
Code
Converter
Roll
NewGame
Hold
R
N
H
W
$Player_1$
$Player_0$
Win

```
3:  if ((Total[Player] + Subtotal) ≥ 100)
    then Win ← 1, goto 5;
    else  if (Roll)
          then Dice ← Counter, goto 2;
          else  if (Hold)
                then Total[Player] ← Total[Player] + Subtotal, Subtotal ← 0 /* , goto 4 */;
                else  goto 3;
4:  if (not Hold)
    then Player ← Next (Player), goto 1;
    else  goto 4;
5:  if (NewGame)
    then Player ← Next (Player), goto 0;
    else  goto 5;
```

Sia all'accensione (Step 0) che a seguito della pressione del pulsante "NewGame" al termine di un incontro (Step 5), il sistema, dopo aver selezionato quale primo "Player" il giocatore successivo a quello che ha vinto l'incontro precedente (all'accensione la scelta è operata in maniera casuale), procede ad azzerare tutti i totalizzatori dei punteggi ed a spegnere i dispositivi di visualizzazione "Win" e "Dice". Prontamente quindi, o al rilascio del pulsante "NewGame" in caso di avvio di un nuovo incontro, il sistema si pone in attesa (Step 1) della richiesta da parte del "Player" di operare il lancio del dado. A seguito della pressione del pulsante "Roll", il sistema genera un numero casuale compreso nel range [1-6], campionando lo stato di un contatore free-running e visualizzandone il valore in "Dice". Al rilascio del pulsante "Roll" (Step 2), se il valore del dado è 1, il gioco, previo azzeramento di "Subtotal", passa nelle mani del successivo giocatore; in caso contrario il sistema, accumulato il valore del dado in "Subtotal", si pone in attesa (Step 3) di una richiesta di "Hold" o di "Roll" da parte del "Player". A fronte della richiesta di "Hold", il sistema somma al punteggio totale conseguito dal "Player" nei precedenti turni il punteggio parziale accumulato in "Subtotal" nel turno corrente, quindi opera la commutazione del "Player" (Step 4); a fronte della richiesta di "Roll", il sistema procede alla generazione di un nuovo numero casuale, reiterando poi i passi precedenti. Il "Player" è dichiarato vincitore (Step 5) non appena

il suo punteggio complessivo raggiunge o eccede la soglia prestabilita.

(N.B.: la transizione diretta dallo Step 2 allo Step 5, corrispondente all'evento alquanto improbabile che il "Player" vinca l'incontro lanciando il dado ripetutamente e sempre con esito positivo durante il proprio (primo) turno (Subtotal + Dice ≥ 100, Subtotal + Total[Player] < 100), non è per semplicità contemplata).

Nell'ipotesi che i giocatori gestiscano i pulsanti in modo coerente (ovvero agiscano su uno solo di essi alla volta e soltanto se risulta contestualmente significativo ai fini dello sviluppo del gioco):

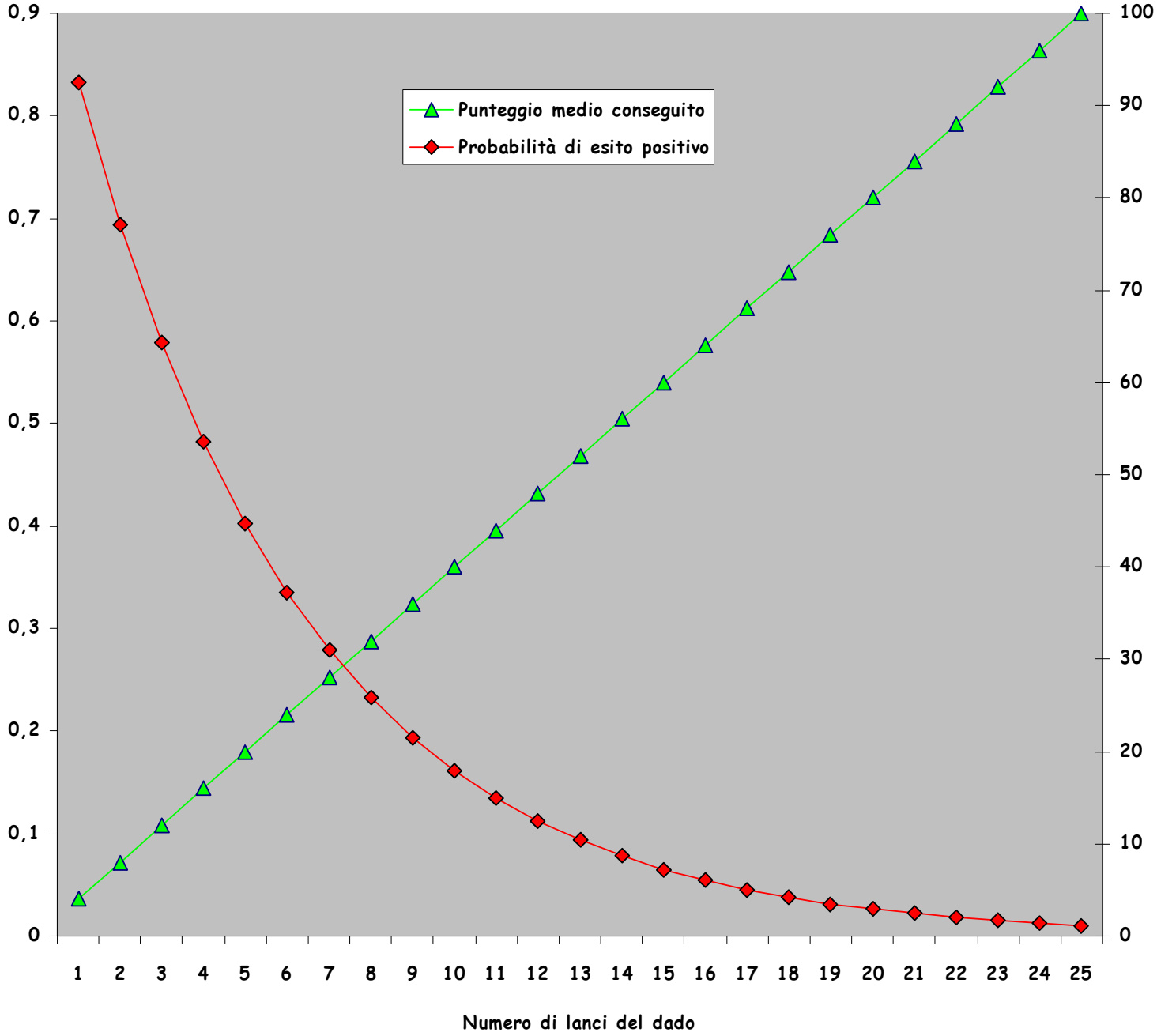
1. si determini il numero di volte che un giocatore "… fortunato" deve mediamente reiterare il lancio del dado onde poter vincere un incontro al primo turno, nonché la probabilità associata a tale evento;
2. si definisca formalmente, in termini di automa a stati finiti, il comportamento dell'unità di controllo, contraddistinta dai seguenti segnali di ingresso / uscita: N, R, H, D_1, C_o / R_D, U_D, R_S, U_S, R_T, U_T, U_P, S, W, di cui gli ultimi due sintetizzati secondo il modello di Moore (in luogo dell'esplicita indicazione dei valori delle tre coppie di segnali binari (R_D, U_D), (R_S, U_S), (R_T, U_T), si adotti la seguente più sintetica notazione basata sull'impiego di tre variabili C_D, C_S, C_T: $C_{\forall\ (\forall = D, S, T)}$ = h (hold) $\rightarrow R_\forall U_\forall = 00$, $C_\forall$ = u (update) $\rightarrow R_\forall U_\forall = 01$, $C_\forall$ = r (reset) $\rightarrow R_\forall U_\forall = 1\text{-}$);
3. si identifichino le modifiche e/o le estensioni che è necessario prevedere a livello di console, data-path e control unit nel caso in cui i giocatori partecipanti al gioco siano N = 8.

Ad ogni lancio:

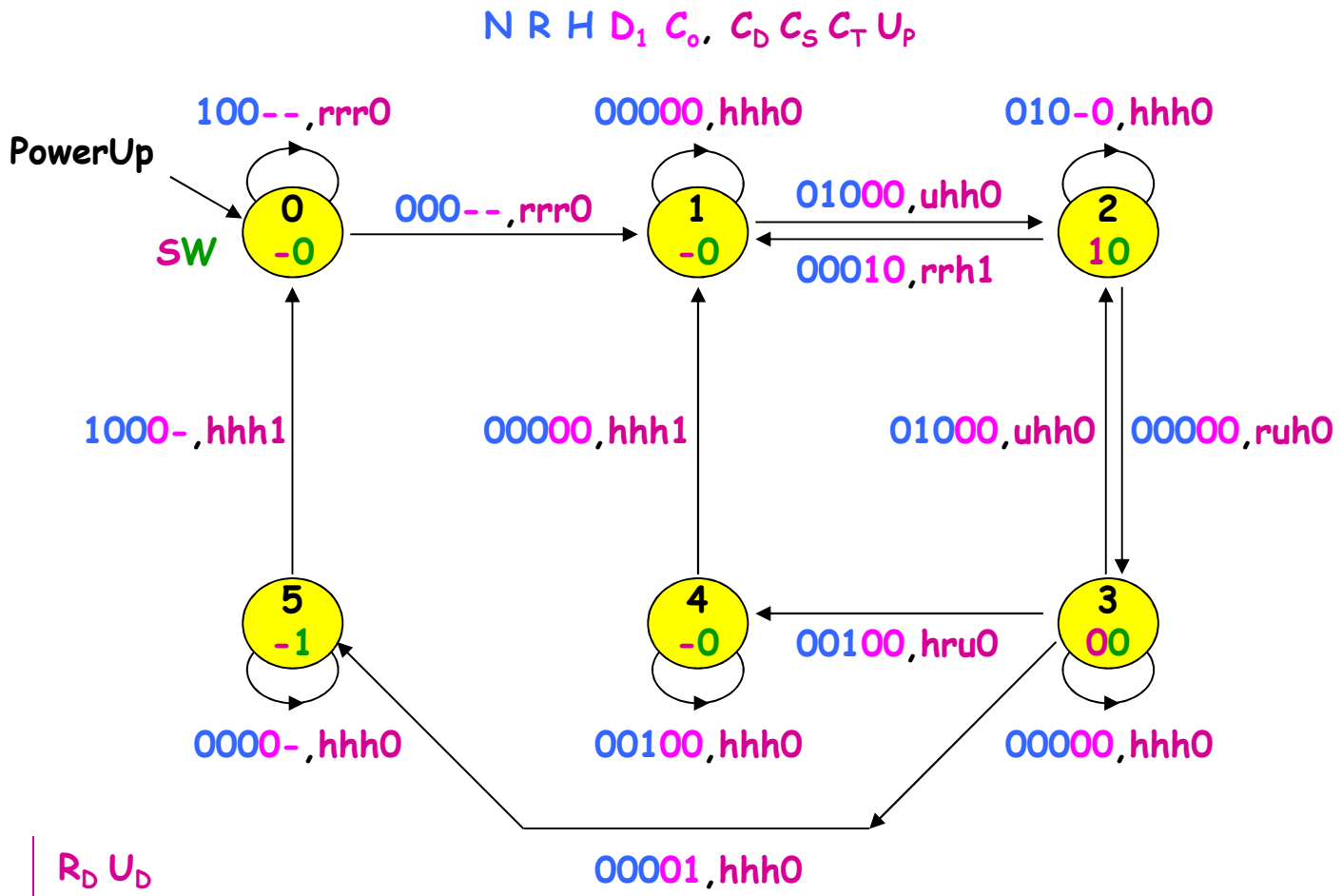
- **probabilità di esito positivo = 1 – probabilità di esito negativo = 1 – 1/6 = 5/6**
- **punteggio mediamente conseguito in caso di esito positivo = (2+3+4+5+6)/5 = 4**

Onde poter vincere un incontro al primo turno:

- **numero medio di lanci = 100/4 = 25**
- **probabilità di esito positivo in 25 lanci consecutivi = $(5/6)^{25} \cong 0.01$**


Punteggio medio conseguito
Probabilità di esito positivo
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
100
90
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Numero di lanci del dado

Unità di controllo: automa a stati finiti



C_D	R_D U_D
h (hold)	0 0
u (update)	0 1
r (reset)	1 -

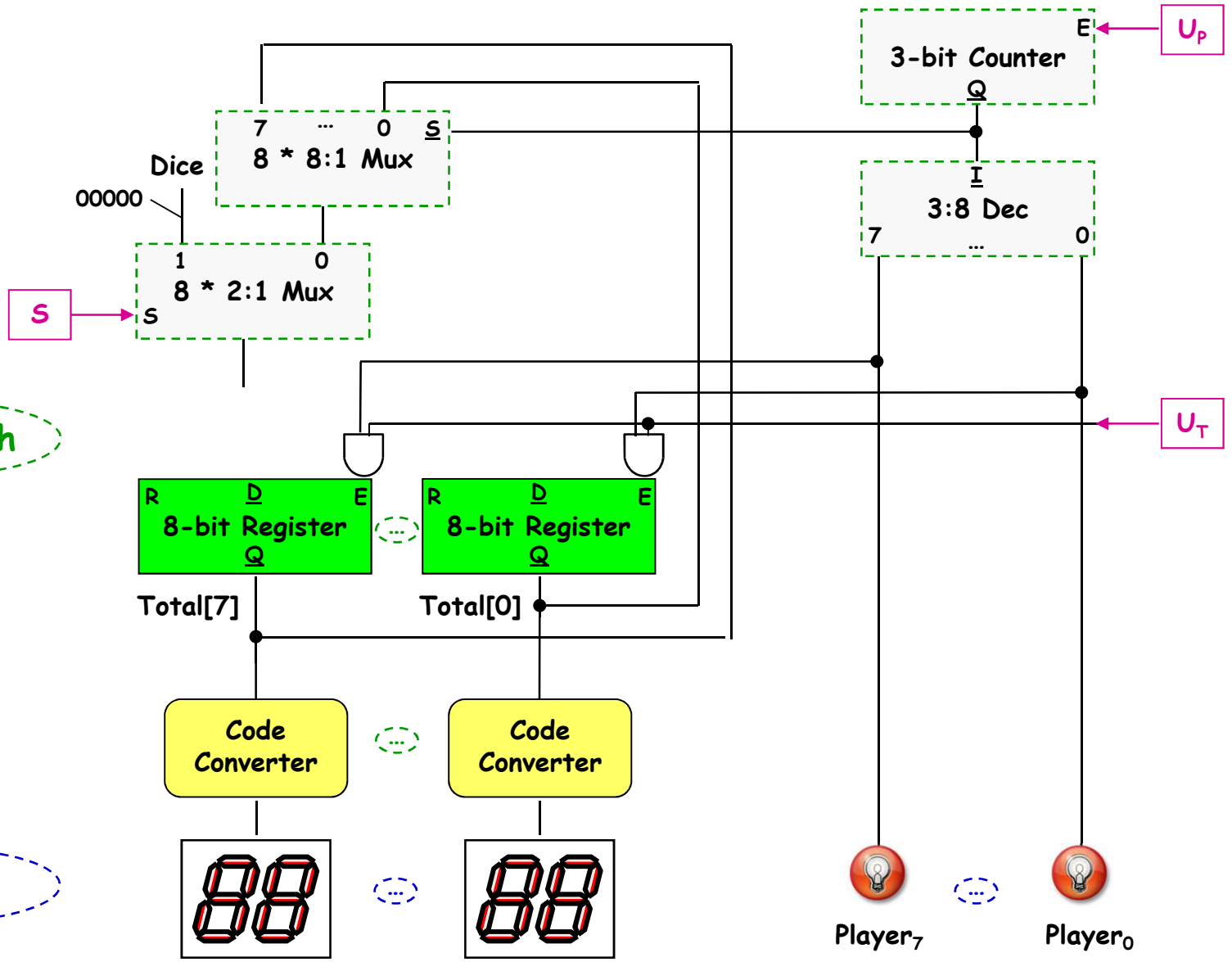
analogamente C_S e C_T

N = 8 → modifiche e/o estensioni a livello di:

control unit — **nessuna**

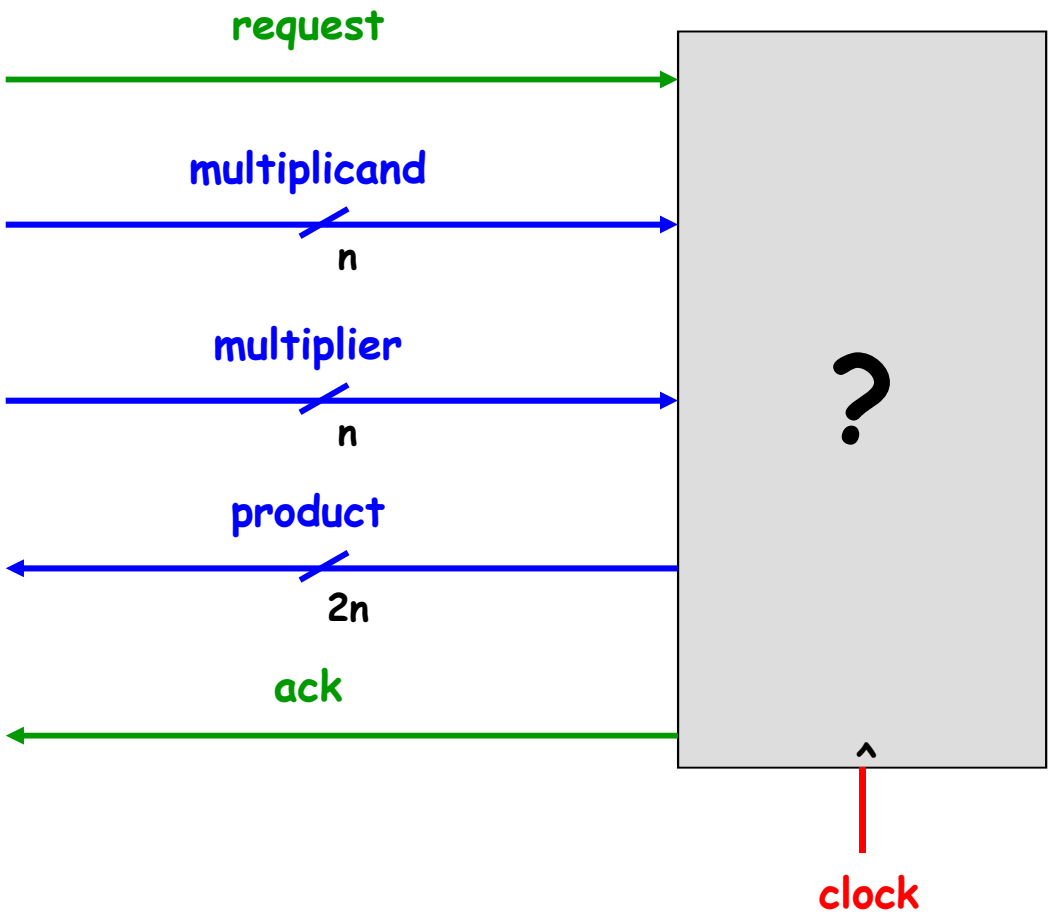
data-path

console



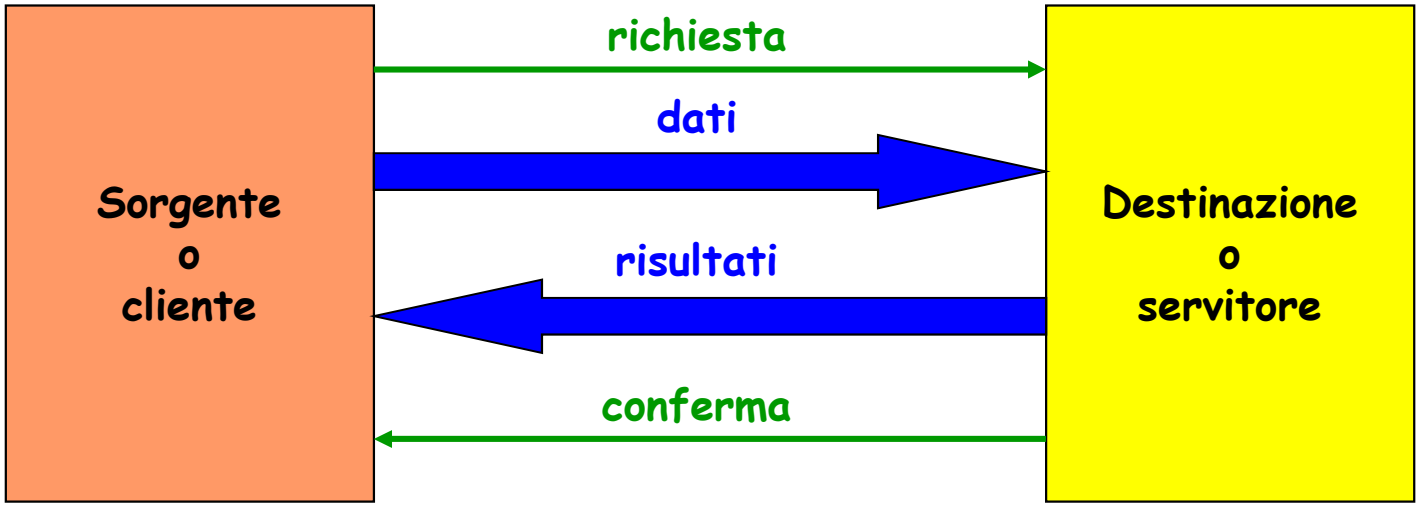
Problema 3

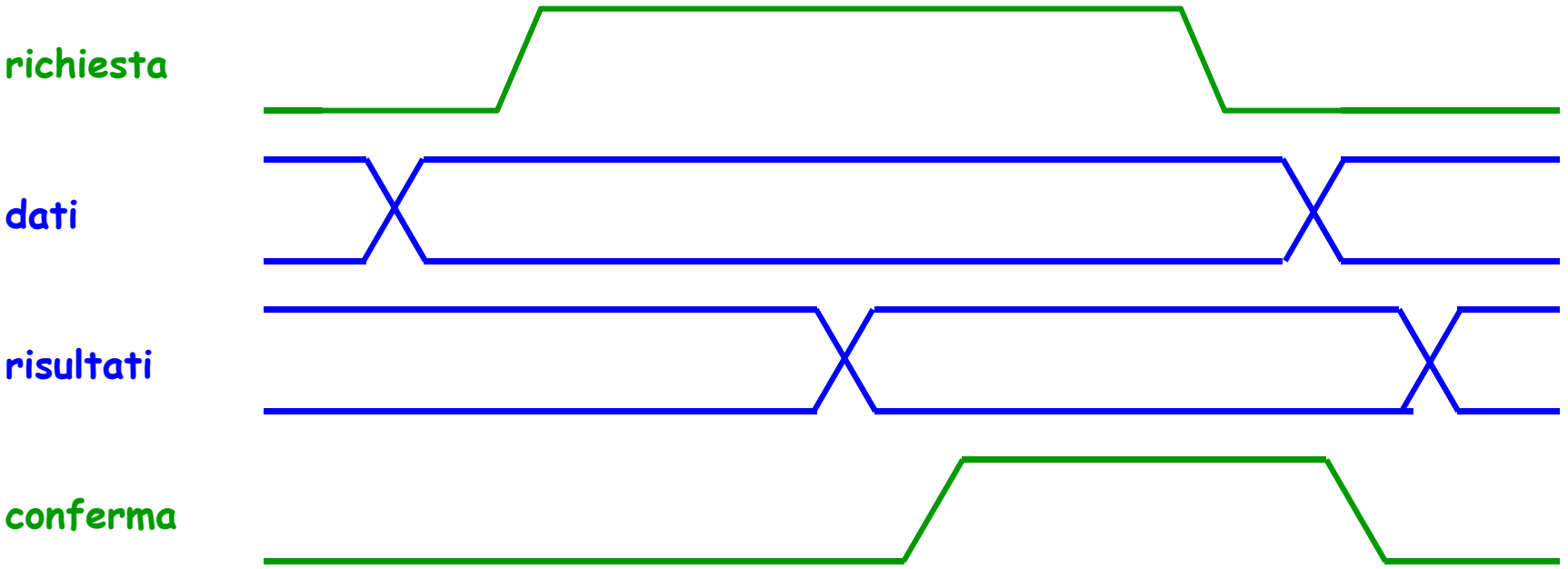
Sintesi di un moltiplicatore binario



multiplicand, multiplier, product: unsigned integer

Comunicazione asincrona: protocollo a stretta di mano





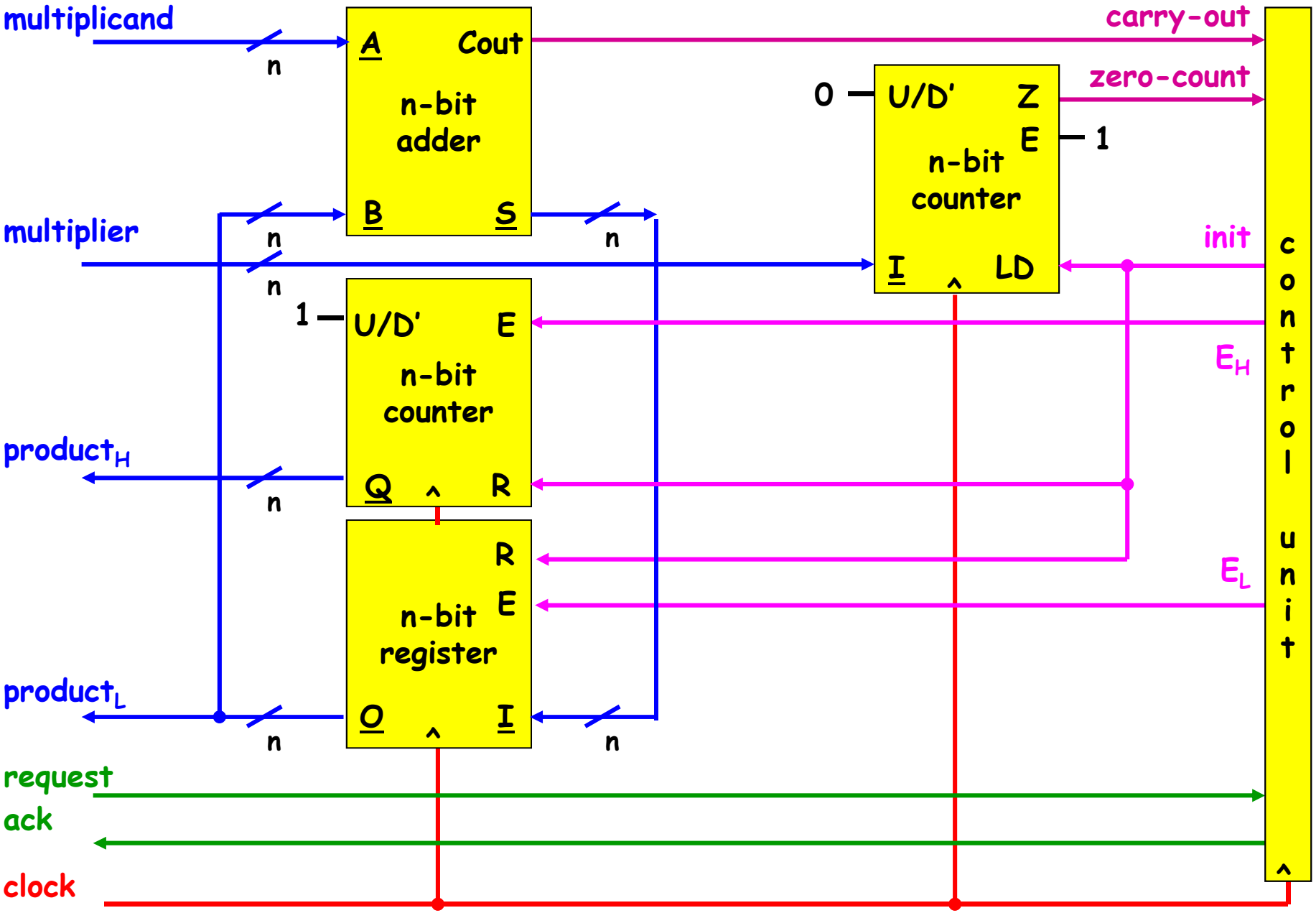
Descrizione algoritmica del processo di elaborazione (1ª soluzione)

```
while (true)
{
    ack = false;
    while (! request);
    counter = multiplier;
    product = 0;
    while (counter > 0)
    {
        product += multiplicand;
        counter --;
    }
    ack = true;
    while (request);
}
```

Descrizione RTL del processo di elaborazione (1ª soluzione)

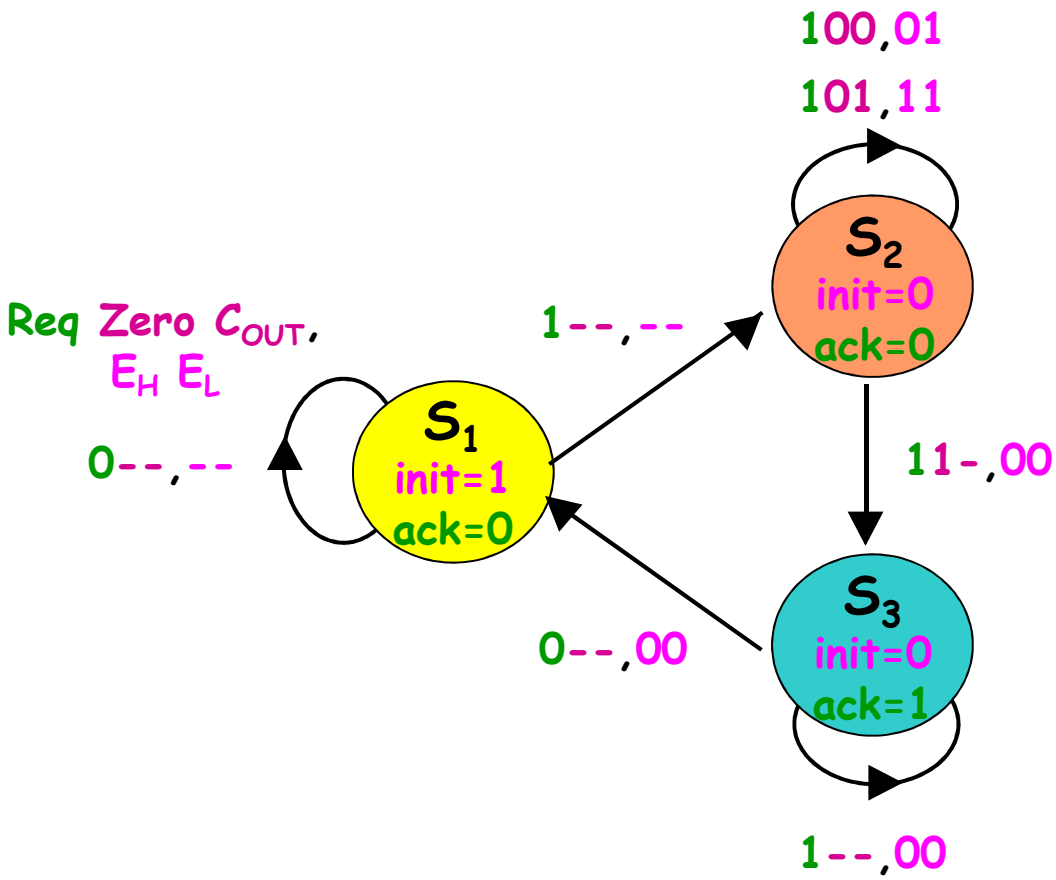
```
1:  ack ← 0,
    if (not request)
    then  goto 1;
    else  counter  ← multiplier,
          product  ← 0 /* , goto 2 */;

2:  ack ← 0,
    if (counter = 0)
    then  product  ← product /* , goto 3 */;
    else  product  ← product + multiplicand,
          counter  ← counter - 1,
          goto 2;

3:  ack ← 1,
    product ← product,
    if (request)
    then  goto 3;
    else  goto 1;
```

Progetto del data-path (1ª soluzione)



Progetto della control unit (1ª soluzione) …

Il grafo degli stati

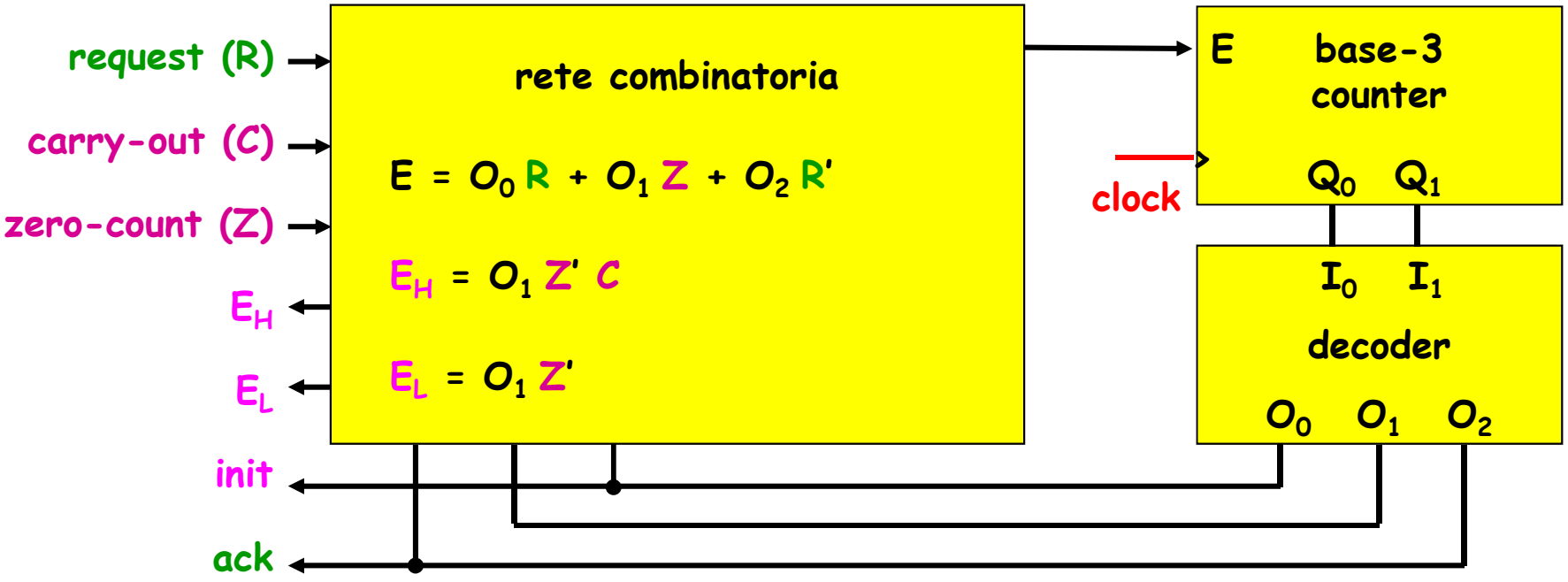


… Progetto della control unit (1ª soluzione)

request zero-count carry-out

s.p.	000	001	011	010	100	101	111	110	init	ack
S_1	S_1,--	S_1,--	S_1,--	S_1,--	S_2,--	S_2,--	S_2,--	S_2,--	1	0
S_2	-,--	-,--	-,--	-,--	S_2,01	S_2,11	S_3,00	S_3,00	0	0
S_3	S_1,00	S_1,00	S_1,00	S_1,00	S_3,00	S_3,00	S_3,00	S_3,00	0	1

s.f., E_H E_L



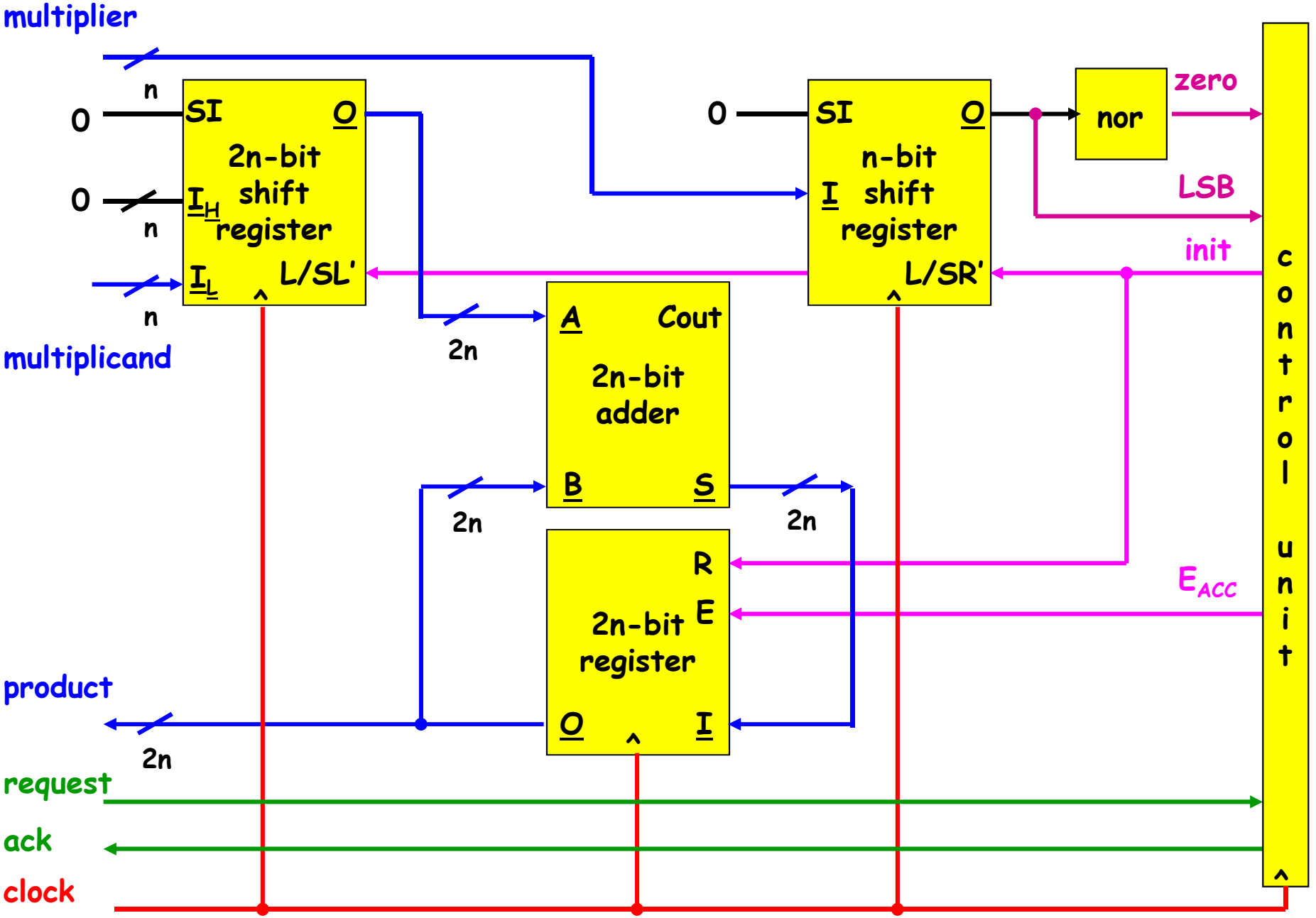
Descrizione algoritmica del processo di elaborazione (2ª soluzione)

```
while (true)
{
    ack = false;
    while (! request);
    _multiplicand_  = multiplicand;
    _multiplier_    = multiplier;
    product = 0;
    while (_multiplier_ > 0)
    {
        if ((_multiplier_ & 1) == 1)
            product += _multiplicand_;
        _multiplicand_  *= 2;
        _multiplier_    /= 2;
    }
    ack = true;
    while (request);
}
```

Descrizione RTL del processo di elaborazione (2ª soluzione)

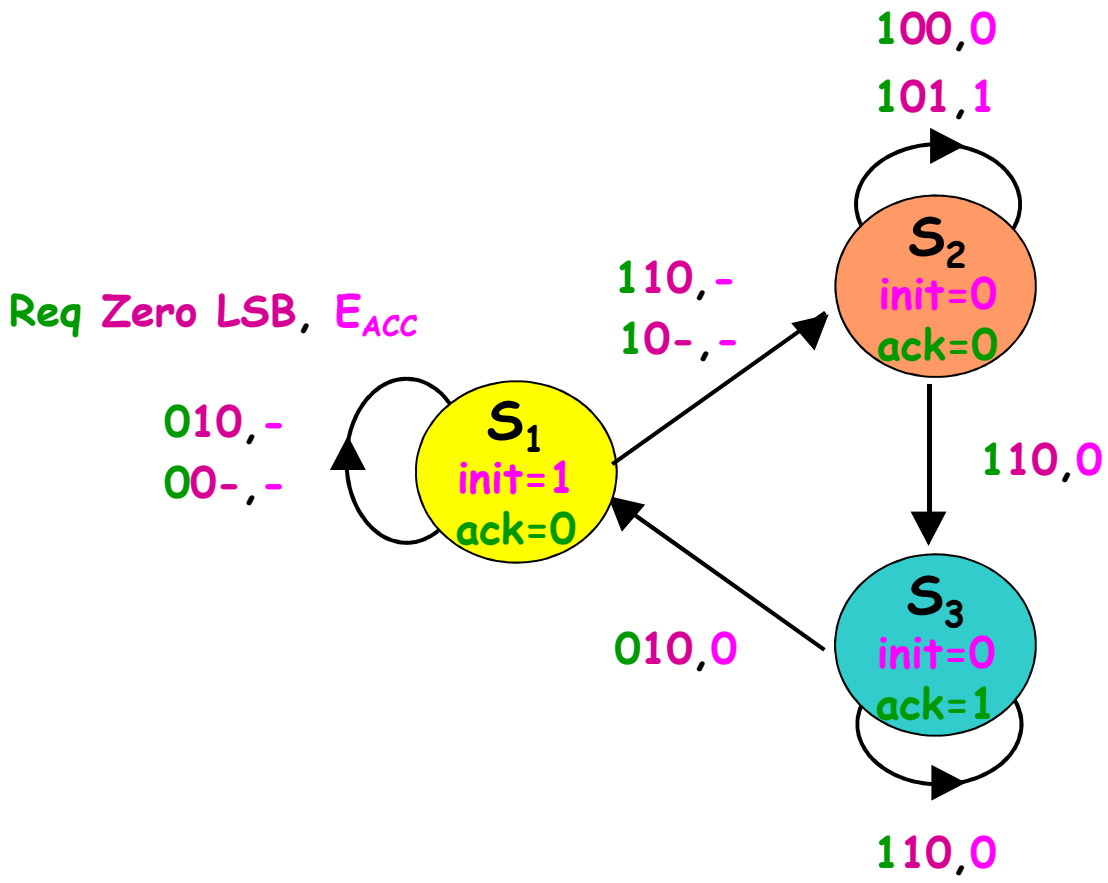
```
1:  ack ← 0,
    if (not request)
    then  goto 1;
    else  _multiplier_    ← multiplier,
          _multiplicand_  ← multiplicand,
          product         ← 0 /* , goto 2 */;

2:  ack ← 0,
    if (_multiplier_ = 0)
    then  product ← product /* , goto 3 */;
    else  if (least_significant_bit (_multiplier_) = 1)
          then product   ← product + _multiplicand_,
          _multiplicand_ ← shift_left  (_multiplicand_, 0),
          _multiplier_   ← shift_right (_multiplier_, 0),
          goto 2;

3:  ack ← 1,
    product ← product,
    if (request)
    then  goto 3;
    else  goto 1;
```

Progetto del data-path (2ª soluzione)



Progetto della control unit (2ª soluzione) …

Il grafo degli stati

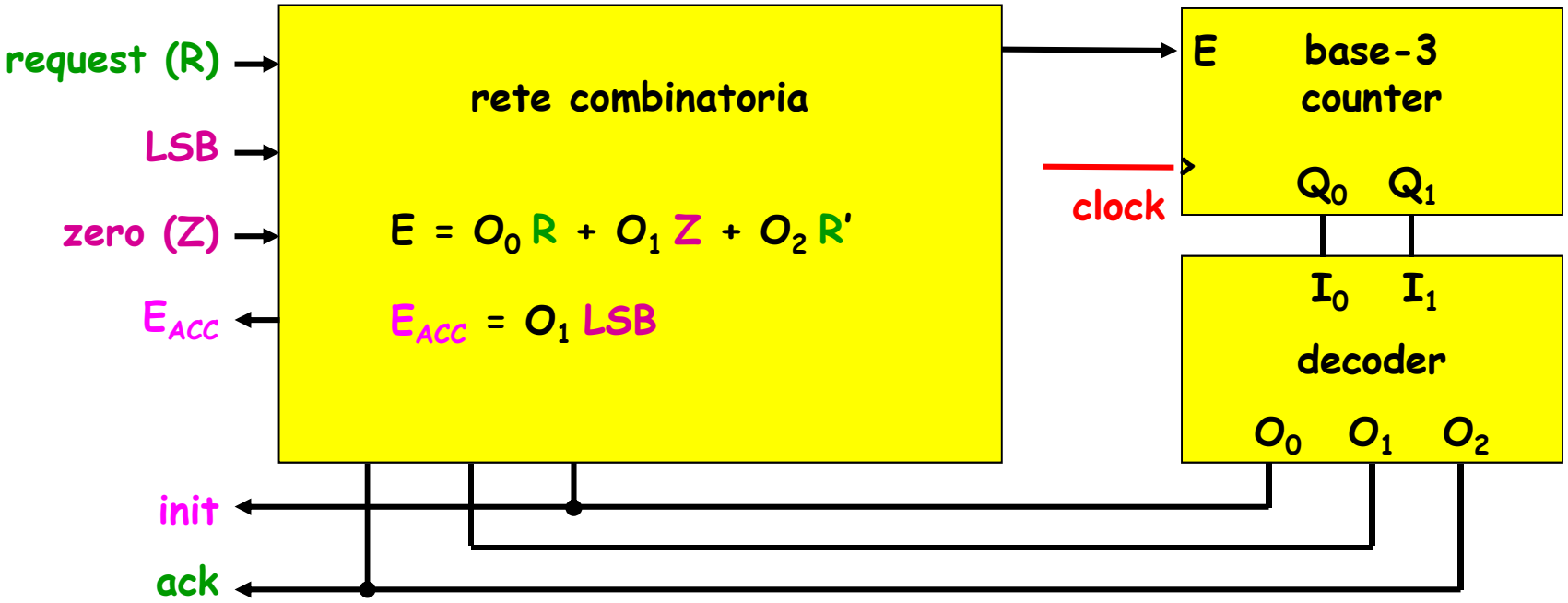


… Progetto della control unit (2ª soluzione)

request zero least-significant-bit

s.p.		000	001	011	010	100	101	111	110	init	ack
	S_1	S_1,-	S_1,-	-,-	S_1,-	S_2,-	S_2,-	-,-	S_2,-	1	0
	S_2	-,-	-,-	-,-	-,-	S_2,0	S_2,1	-,-	S_3,0	0	0
	S_3	-,-	-,-	-,-	S_1,0	-,-	-,-	-,-	S_3,0	0	1

s.f., E_{ACC}



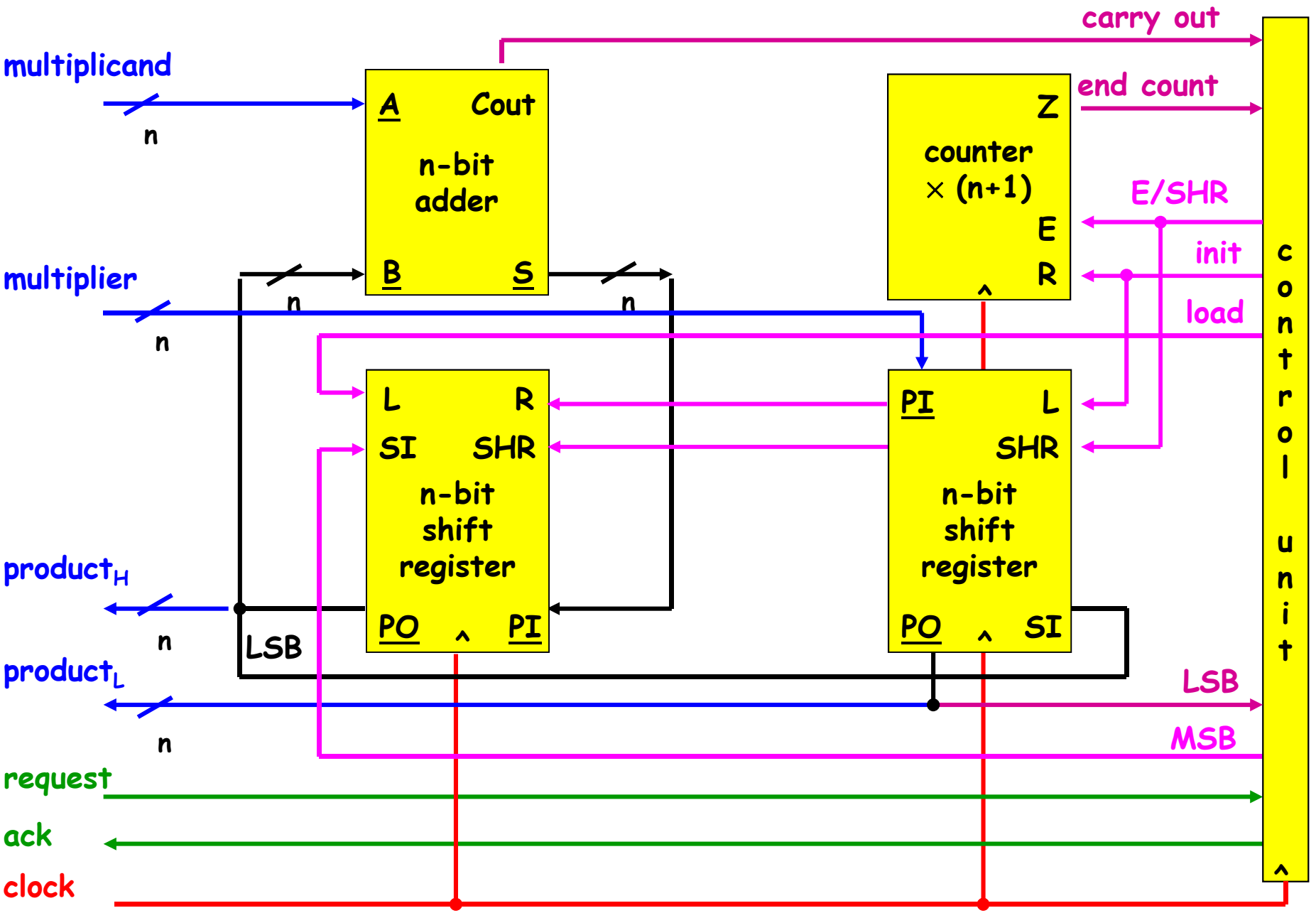
Descrizione algoritmica del processo di elaborazione (3ª soluzione)

```
while (true)
{
     ack = false;
     while (! request);
     product_L = multiplier;
     product_H = 0;
     for (counter = 0; counter < n; counter++)
     {
          if ((product_L & 1) == 1)
               product_H += multiplicand;
          product /= 2;
     }
     ack = true;
     while (request);
}
```

Descrizione RTL del processo di elaborazione (3ª soluzione)

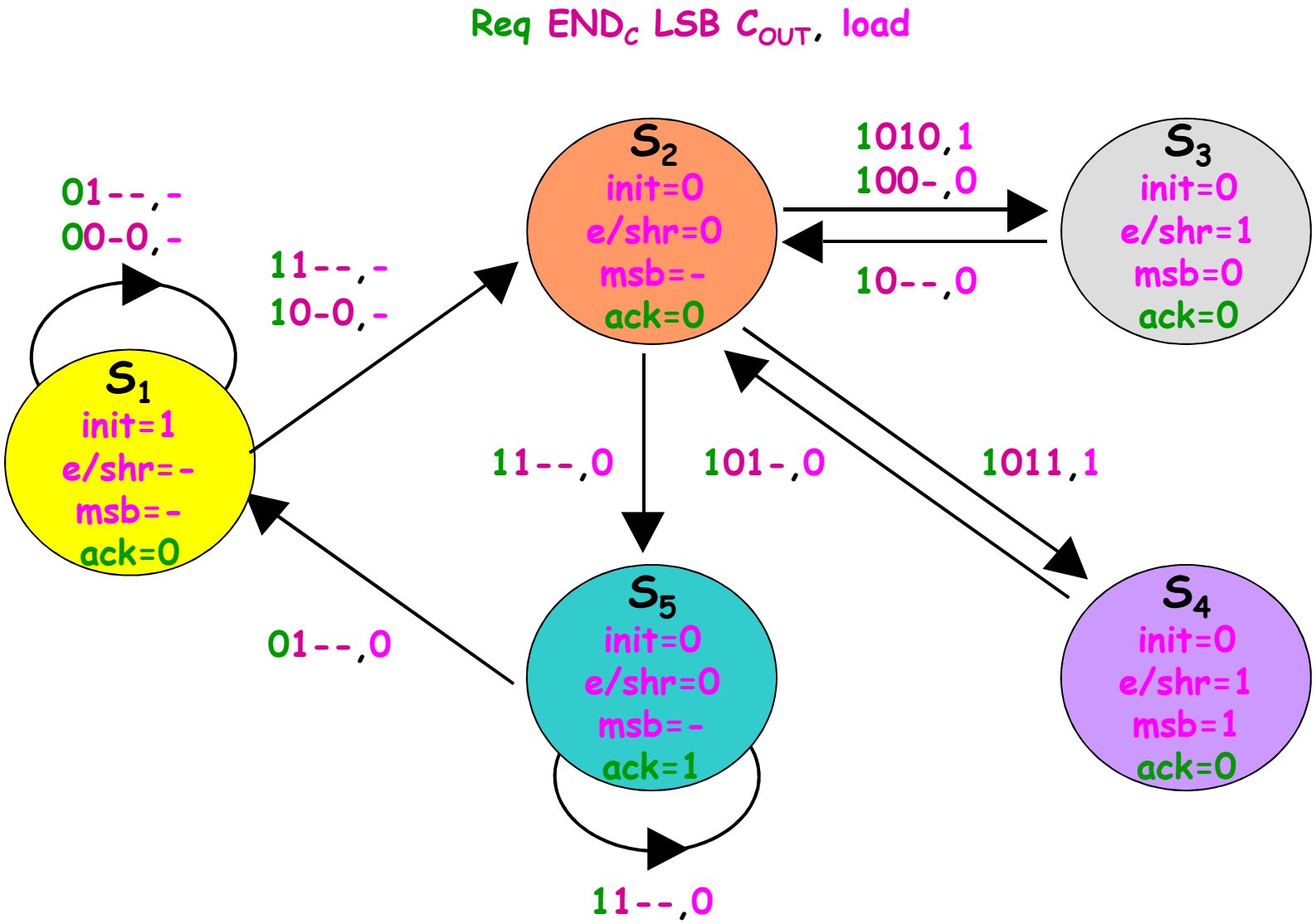
```
1:  ack ← 0,
    if (not request)
    then goto 1;
    else  product_L ← multiplier, product_H ← 0, counter ← 0;

2:  ack ← 0,
    if (counter = n)
    then product ← product, goto 5;
    else  counter ← counter, product_L ← product_L,
          if (least_significant_bit (product) = 0)
          then product_H ← product_H;
          else  product_H ← product_H + multiplicand,
                if ((most_significant_bit (product_H + multiplicand)) == 1)
                then goto 4;

3:  ack ← 0,
    product ← shift_right (product, 0), counter ← counter + 1, goto 2;

4:  ack ← 0,
    product ← shift_right (product, 1), counter ← counter + 1, goto 2;

5:  ack ← 1, product ← product,
    if (request) then goto 5; else goto 1;
```

Progetto del data-path (3ª soluzione)



Progetto della control unit (3ª soluzione) ...

Il grafo degli stati

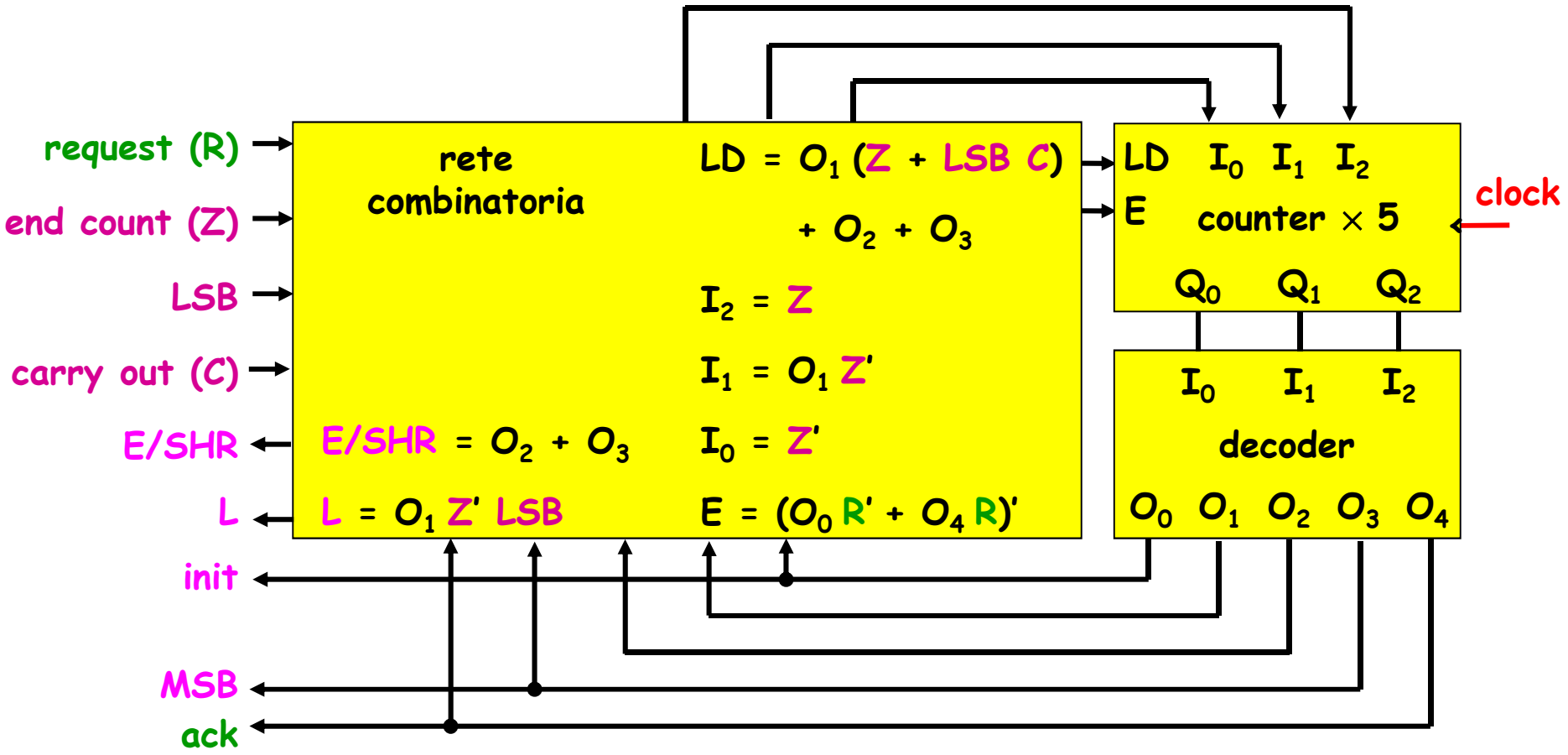


… Progetto della control unit (3ª soluzione) …

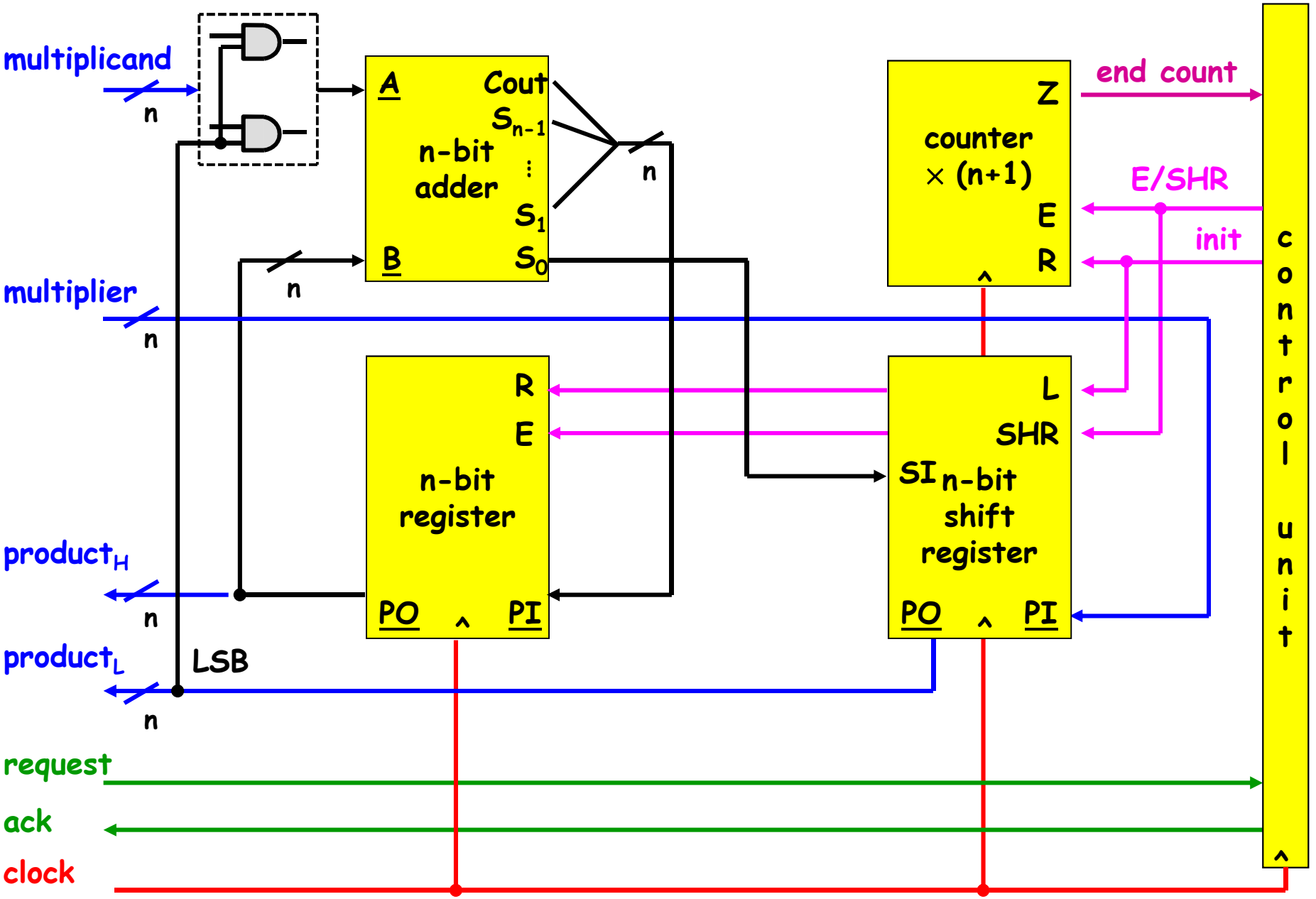
	request = 0								request = 1							
	end-count				least-significant-bit				carry-out							
s.p.	000	001	011	010	100	101	111	110	000	001	011	010	100	101	111	110
S_1	S_1,-	-,-	-,-	S_1,-	S_1,-	S_1,-	S_1,-	S_1,-	S_2,-	-,-	-,-	S_2,-	S_2,-	S_2,-	S_2,-	S_2,-
S_2	-,-	-,-	-,-	-,-	-,-	-,-	-,-	-,-	S_3,0	S_3,0	S_4,1	S_3,1	S_5,0	S_5,0	S_5,0	S_5,0
S_3	-,-	-,-	-,-	-,-	-,-	-,-	-,-	-,-	S_2,0	S_2,0	S_2,0	S_2,0	-,-	-,-	-,-	-,-
S_4	-,-	-,-	-,-	-,-	-,-	-,-	-,-	-,-	-,-	-,-	S_2,0	S_2,0	-,-	-,-	-,-	-,-
S_5	-,-	-,-	-,-	-,-	S_1,0	S_1,0	S_1,0	S_1,0	-,-	-,-	-,-	-,-	S_5,0	S_5,0	S_5,0	S_5,0
	s.f., load															

s.p.	init	e/shr	msb	ack
S_1	1	-	-	0
S_2	0	0	-	0
S_3	0	1	0	0
S_4	0	1	1	0
S_5	0	0	-	1

… Progetto della control unit (3ª soluzione)



Oppure, più efficientemente: data-path



Oppure, più efficientemente: control unit

Il grafo degli stati