1. Progettare una macchina a stati finiti (di Moore) che realizza una sistema di accesso a combinazione segreta: soltanto dopo aver premuto in sequenza i tasti: “1 4 4 2”, l’uscita che comanda l’apertura della porta va a livello logico 1 e consente l’apertura della porta. Dopodichè la pressione di un qualunque tasto riporta la macchina nello stato iniziale. (suggerimento: si consideri una sola linea d’ingresso per tutti i tasti diversi da “1”, “2” e “4”). Determinare STG, STT, STT codificata e la struttura circuitale completa della macchina. Definire la complessità della parte combinatoria ed il suo cammino critico.

Soluzione:

Supponiamo che la macchina legga in ogni istante un carattere e che questo carattere sia ‘1’, ‘2’, ‘4’ o ‘X’ dove ‘X’ indica qualunque altro carattere diverso dai precedenti. Inoltre supponiamo che all’inizio la macchina sia nello stato corrispondente a nessun carattere riconosciuto.

I possibili ingressi nel sistema sono 4:

Ingressi	Descrizione
1	Carattere ‘1’
2	Carattere ‘2’
4	Carattere ‘4‘
X	Carattere diverso da ‘1’, ‘2’, ‘4’

Le possibili uscite sono 2:

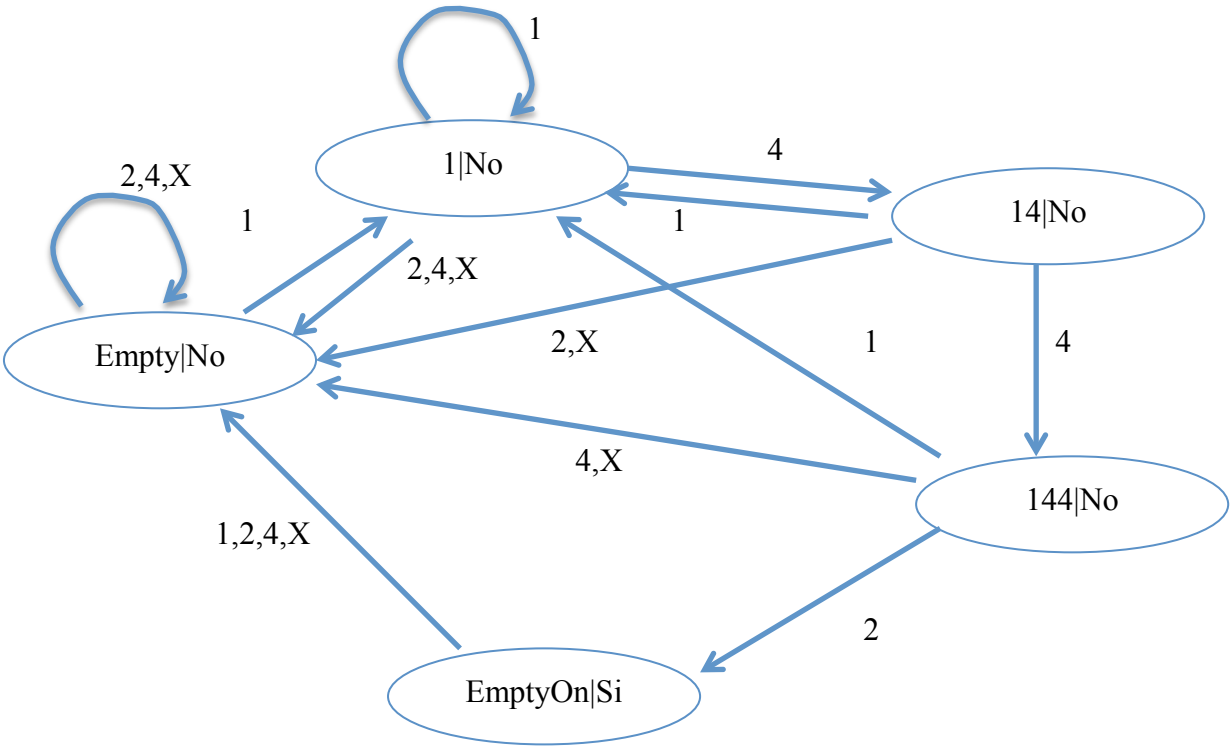
Uscite	Descrizione
No	Porta chiusa
SI	Porta aperta

Il sistema deve ricordare quale prefisso della stringa da cercare è già entrato nel sistema. Poiché useremo automi di Moore, può capitare che alcuni stati siano duplicati per generare output diversi. Ne segue, che i possibili stati del sistema sono:

Stati	Descrizione	Uscita
Empty	Nessun prefisso, nessuna parola riconosciuta	NO
1	Riconosciuto il prefisso “1”	NO
14	Riconosciuto il prefisso‘14’	NO
144	Riconosciuto il prefisso “144”	NO
EmptyOn	Riconosciuta la parola “1442”, apri porta, non ricordare niente	SI

Per come è dato il testo del problema, il sistema dopo aver riconosciuto la combinazione, ritorna nello stato iniziale di nessun prefisso riconosciuto. Nel caso di errore che comporti un suffisso riutilizzabile, es. ‘141’, il sistema può comportarsi in due modi: considerare valido l’ultimo ‘1’ inserito e ripartire dallo stato ‘1’ oppure considerare il riconoscimento fallito, scartare l’ultimo ‘1’ inserito e ripartire da ‘Empty’. In questo esempio poiché non diversamente specificato, consideriamo il secondo caso, cioè valido l’ultimo ‘1’ letto.

Lo STG è cosi strutturato:



La STT è la seguente:

	δ				λ
Stato	X	1	2	4	**O**
Empty	Empty	1	Empty	Empty	NO
1	Empty	1	Empty	14	NO
14	Empty	1	Empty	144	NO
144	Empty	1	EmptyOn	Empty	NO
EmptyOn	Empty	Empty	Empty	Empty	SI

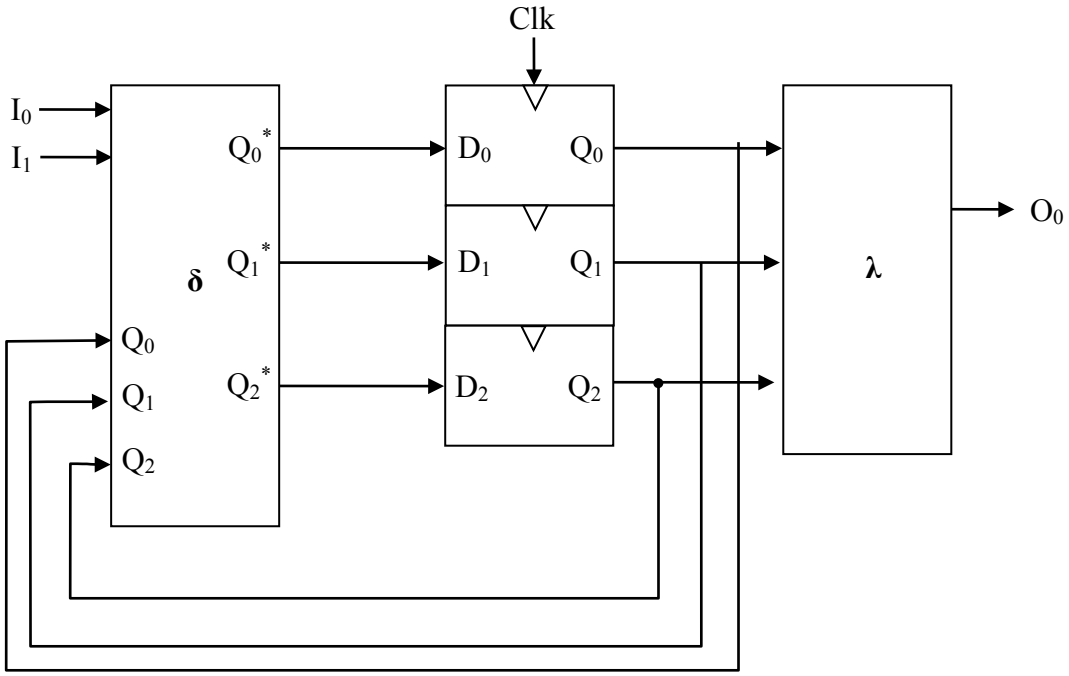
Per codificare i cinque stati servono ceil($\log_2 5$) = 3 bit. Analogamente servono 2 bit per l'ingresso e 1 bit per l'uscita. Possiamo dare la seguente mappatura sugli stati, gli ingressi e le uscite:

Stato	**$Q_2Q_1Q_0$**
Empty	000
1	001
14	010
144	011
EmptyOn	111

Uscite	**O_0**
NO	0
SI	1

Ingressi	I_1I_0
X	00
1	01
2	10
4	11

Il circuito che realizza l'automa di Moore delineato sarà quindi costituito da due F/F tipo D:



Possiamo riscrivere la STT usando le codifiche adottate:

	$\delta = Q^*_1Q^*_0$				λ
Stato = Q_1Q_0	**I_1I_0**=00 (X)	01 (1)	10 (2)	11 (4)	**O_0**
000 (Empty)	000	001	000	000	0
001 (1)	000	001	000	• 001	0
010 (14)	000	001	000	010	0
011 (144)	000	001	111 •	000	0
111 (EmptyOn)	000	000	000	000	1

La funzione di uscita è banalmente:

$$\mathbf{O_0 = Q_2}$$

Riscriviamo la funzione stato prossimo in forma tabellare:

$Q_2Q_1Q_0\,I_1I_0$	Q^*_2	Q^*_1	Q^*_0
000 00	0	0	0
000 01	0	0	1
000 10	0	0	0
000 11	0	0	0
001 00	0	0	0
001 01	0	0	1
001 10	0	0	0
001 11	0	1	0
010 00	0	0	0
010 01	0	0	1
010 10	0	0	0
010 11	0	1	1
011 00	0	0	0
011 01	0	0	1
011 10	1	1	1
011 11	0	0	0
100 00	X	X	X
100 01	X	X	X
100 10	X	X	X
100 11	X	X	X
101 00	X	X	X
101 01	X	X	X
101 10	X	X	X
101 11	X	X	X
110 00	X	X	X
110 01	X	X	X
110 10	X	X	X
110 11	X	X	X
111 00	0	0	0
111 01	0	0	0
111 10	0	0	0
111 11	0	0	0

Se prendiamo tutte le indeterminate X a 0 allora la parte bassa della tabella corrispondente a Q_2=1 è sintetizzabile banalmente con 0. E' possibile qundi dare una sitesi algebrica delle funzioni stato prossimo nel seguente modo:

$Q^*_2 = Q_2\ 0 + \sim Q_2\ f_2(Q_1, Q_0, I_1, I_0) = \sim Q_2\ f_2(Q_1, Q_0, I_1, I_0)$
$Q^*_1 = Q_2\ 0 + \sim Q_2\ f_1(Q_1, Q_0, I_1, I_0) = \sim Q_2\ f_1(Q_1, Q_0, I_1, I_0)$
$Q^*_0 = Q_2\ 0 + \sim Q_2\ f_0(Q_1, Q_0, I_1, I_0) = \sim Q_2\ f_0(Q_1, Q_0, I_1, I_0)$

Il problema quindi si riduce a sintentizzare la tabella a 4 ingressi $Q_1Q_0I_1I_0$ e tre uscite per Q_2=0. $f_2(Q_1, Q_0, I_1, I_0)$ poiché ha un solo 1 in tabella si riduce alla sua espressione SOP:

$$Q^*_2 = \sim Q_2\ f_2(Q_1, Q_0, I_1, I_0) = \sim Q_2\ Q_1Q_0I_1{\sim}I_0$$

La sua complessità è pari a 4, il cammino critico pari a 3:

$$\{\sim Q_2\ [(Q_1Q_0)\,(I_1{\sim}I_0)]\}$$

$f_1(Q_1, Q_0, I_1, I_0)$ condivide con f_2 il min-term, i restanti due 1 suggeriscono una espressione algebrica che utilizza una XOR tra Q_1 e Q_0 per I_1 e I_0 uguali a 1

$$Q^*_1 = \sim Q_2\ f_1(Q_1, Q_0, I_1, I_0) = \sim Q_2\ [Q_1Q_0I_1{\sim}I_0 + (Q_1 \text{ xor } Q_0)I_1I_0]$$

La sua complessità è pari a 5 (il min-termine $Q_1Q_0I_1{\sim}I_0$ è condiviso con f_2), il cammino critico pari a 4:

$$(\sim Q_2\ \{\ [(Q_1Q_0)(I_1{\sim}I_0)] + [(Q_1 \text{ xor } Q_0)(I_1I_0)]\ \})$$

$f_0(Q_1, Q_0, I_1, I_0)$ condivide con f_2 il min-term; i min-termini corrispondenti a I_1I_0 uguali a 01 possono essere raggruppati; infine va aggiunto il mintermine $Q_1{\sim}Q_0I_1I_0$:

$$Q^*_0 = \sim Q_2\ f_0(Q_1, Q_0, I_1, I_0) = \sim Q_2\ (Q_1Q_0I_1{\sim}I_0 + {\sim}I_1I_0 + Q_1{\sim}Q_0I_1I_0\ )$$

La sua complessità è pari a 6 (il min-termine $Q_1Q_0I_1{\sim}I_0\ I_0$ e la porta I_1I_0 sono condivise con f_2 e f_1), il cammino critico pari a 5:

$$[\sim Q_2\ (\ \{[(Q_1Q_0)(I_1{\sim}I_0)] + {\sim}I_1I_0\ \} + [(Q_1{\sim}Q_0)(I_1I_0)]\ )]$$

Identiche conclusioni possono essere dedotte considerando la tabella intera, derivando le SOP delle funzioni $Q^*_2\ Q^*_1\ Q^*_0$ e sempificando.

2. progettare una cache a 4 vie di 32kbytes di capacità in grado di memorizzare due word per blocco, si supponga un'architettura MIPS. Indicare cosa succede nella cache, inizialmente vuota, quando si effettuano gli accessi 0, 16, 32, 20. Quanto vale il tempo di accesso.

Soluzione:

Una cache a 4 vie è una cache in grado di memorizzare 4 blocchi per ogni indice, ognuno con un tag diverso. 32Kbyte corrispondono a 32K/4= 8Kword (4 byte = 1 word nel MIPS). Ogni blocco contiene 2 word quindi il numero di blocchi totali corrisponde a 8K/2word=4Kblocchi. Il numero di linee è quindi 4K/4linee = 1K indici diversi, cioè 10 bit di indice (2^{10} = 1024).

Poiché l'indirizzamento utile sul bus indirizzi è di 32 bit nel MIPS, nessun bit più significativo dell'indirizzo può essere ignorato. Un bit viene utilizzato per puntare quale delle due word del blocco estrarre. Il tag risulta quindi pari a 32-2-10-1 = 19 bit (- 2 bit per la lunghezza della word - 10 bit per l'indirizzamento degli indici – 1 bit della selezione del blocco). L'indirizzo a 32 bit viene smontato in questo modo:

	tag	Indice	Selezione blocco	Unused
Bit	**31-13**	**12-3**	**2**	**1-0**

Lo spazio fisicamente necessario per implementare la cache è pari a (non consideriamo i bit di *age* necessari per selezionare la cella da rimuovere in caso di conflitto):

1 bit di validate + 19 bit di tag + 2*32 bit di dato = 84 bit (+ age)

moltiplicati per il numero di blocchi:

84bit * 4 vie * 1024 linee= 344064 bit = 42Kbyte (+age)

L'indirizzo 0=0x00000000 si scompone in questo modo:

	tag	Indice	Selezione blocco	Unused
Bit	**31-13**	**12-3**	**2**	**1-0**
	0000 0000 0000 0000 000	**0 0000 0000 0**	**0**	**00**

che corrisponde all'indice 0, al tag = 0 ed all'offset = 0.

L'indirizzo 16 = 0x00000010 si scompone in questo modo:

	tag	Indice	Selezione blocco	Unused
Bit	**31-13**	**12-3**	**2**	**1-0**
	0000 0000 0000 0000 000	**0 0000 0001 0**	**0**	**00**

che corrisponde all'indice 2, al tag = 0 ed all'offset = 0.

L’indirizzo 32=0x00000020 si scompone in questo modo:

	tag	Indice	Selezione blocco	Unused
Bit	**31-13**	**12-3**	**2**	**1-0**
	0000 0000 0000 0000 000	**0 0000 0010 0**	**0**	**00**

che corrisponde all’indice 4, al tag = 0 ed all’offset = 0.

L’indirizzo 20=0x00000014 si scompone in questo modo:

	tag	Indice	Selezione blocco	Unused
Bit	**31-13**	**12-3**	**2**	**1-0**
	0000 0000 0000 0000 000	**0 0000 0001 0**	**1**	**00**

che corrisponde all’indice 1, al tag = 0 ed all’offset = 1.

Data la sequenza di accessi 0, 16, 32, 20 ed una cache inizialmente vuota avremo per

0 --> indice = 0, tag = 0, offset = 0. La cache è vuota (tutti i bit di validate a 0) quindi è una miss, l’algoritmo di age scelglie ad esempio la via 0 per memorizzare le due word di indirizzi 0 e 4 estratte dalla RAM. Il tempo di miss è due accessi alla RAM es. 60ns*2.

16 --> indice = 2, tag = 0, offset = 0. La cache nella linea 2 è vuota (tutti i bit di validate sulla linea sono a 0) quindi è una miss, l’algoritmo di age sceglie ad esempio la via 1 per memorizzare le due word di indirizzi 16 e 20 estratte dalla RAM. Il tempo di miss è due accessi alla RAM es. 60ns*2.

32 --> indice = 4, tag = 0, offset = 0. La cache nella linea 4 è vuota (tutti i bit di validate sulla linea sono a 0) quindi è una miss, l’algoritmo di age sceglie ad esempio la via 2 per memorizzare le due word di indirizzi 32 e 36 estratte dalla RAM. Il tempo di miss è due accessi alla RAM es. 60ns*2.

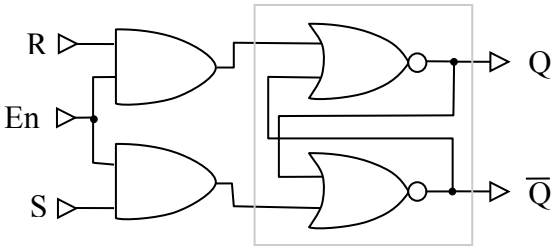
20 --> indice = 0, tag = 0 offset = 1. Sulla linea 0 della cache è presente una via con tag=0 e bit di validate 1 (caricata durante l’accesso a 16) quindi è una hit. Viene selezionata la seconda word del banco. Il tempo di hit è un accesso alla cache es. 2ns.

Ne segue che, partendo da cache vuota, gli accessi 0, 16, 32, 20 corrispondono ad un tempo totale di 3*t.miss +1 * t.hit = 3*60*2 + 1 *2.

3. Disegnare un latch di tipo SR e relativa tabella di eccitazione.

Latch sincrono SR (positive level-triggered)

*Un **latch** è un **bistabile sincrono sensibile ai livelli del segnale di controllo***



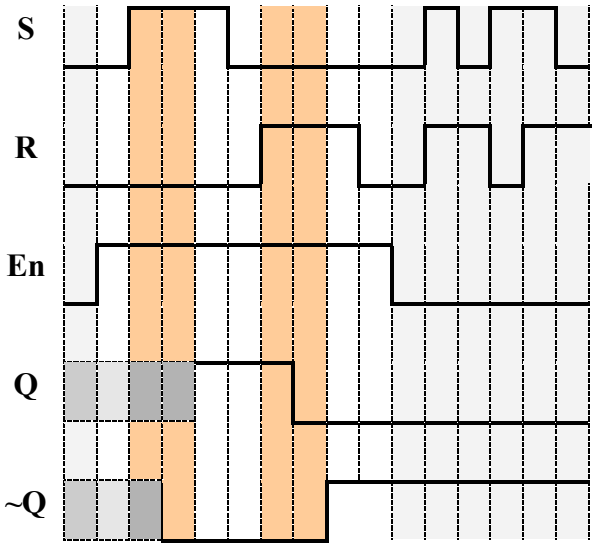


Tabella delle transizioni o *stato prossimo*:

S	R	En=E	Q	Q^*
X	**X**	0	0	0
X	**X**	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	1	0	0
0	1	1	1	0
1	0	1	0	1
1	0	1	1	1
1	1	1	0	**X**
1	1	1	1	**X**

*Configurazioni **ininfluenti**:il comportamento del circuito per queste configurazioni non dipende dalle variabili X.*

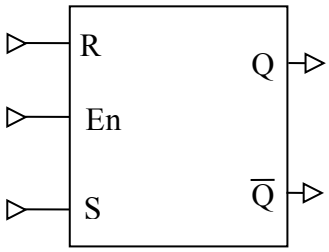


Tabella delle eccitazioni:

Q	Q*	En=E	S	R
0	**0**	0	0	0
0	**1**	0	1	1
1	0	1	0	1
1	1	1	1	1
0	1	1	0	0
0	1	1	1	0
1	0	1	0	1
1	0	1	1	1
1	1	1	0	**X**
1	1	1	1	**X**