SCHEDA **ASE1504**		Data: **09 Giugno 2015**
Cognome	Nome	Matricola

ESERCIZIO N°1

6 punti

a) Determinare la mappa di Karnaugh di una funzione logica $Y = f(X_4, X_3, X_2, X_1, X_0)$
dove X_3, X_2, X_1, X_0 rappresentano una cifra in codifica BCD,
mentre X_4 è un bit di parità (logica: Parità DISPARI).
Y vale 1 se la regola di parità è corretta e la cifra rispetta la codifica BCD (Y vale 0 altrimenti).
b) Realizzare a 2 livelli di logica (SP) con porte logiche elementari la funzione del punto a) scegliendo tra le diverse possibili soluzioni quella che minimizza il numero di porte logiche.
c) Se le porte logiche elementari (AND, OR, NOT) a K ingressi hanno T_{pd} = 0,15 ns + 0,2 K ns quale è il T_{pd} massimo del circuito di cui al punto b)? Se ingressi e uscite del circuito combinatorio di cui al punto b) sono registrati con registri aventi T_{co} = 0,5 ns e T_{su} = 0,5 ns quale è la massima frequenza di lavoro possibile?

ESERCIZIO N°2

4 punti

a) Realizzare la funzione $Y = f(X_4, X_3, X_2, X_1, X_0)$ di cui all'esercizio 1 tramite multiplexer.
b) Realizzare la funzione $Y = f(X_4, X_3, X_2, X_1, X_0)$ di cui all'esercizio 1 tramite decoder.

ESERCIZIO N°3

6 punti

Dati i numeri $A = -129{,}75$ $B = 0{,}125$ $C = -45{,}625$
a) Determinare la loro rappresentazione in virgola fissa e MS, C2, C1, Traslazione e il numero minimo di bit necessario per rappresentarli tutti correttamente.
b) Se si usa una ALU a 8 bit che opera in C2 si commettono errori di rappresentazione per A, B e C? Se sì, di che entità sono gli errori in valore assoluto e percentuale, se è stato usato troncamento?
c) Determinare la rappresentazione di A, B e C in virgola mobile formato standard IEEE754 singola precisione. Anche in questo caso determinare l'entità degli errori in valore assoluto e percentuale.

ESERCIZIO N°4

5 punti

Progettare un T-FF (positive edge-triggered) come macchina asincrona.

ESERCIZIO N°5

5 punti

Progettare una macchina di Mealy sincronizzata con 2 ingressi X e Y e una sola uscita U che viene posta e mantenuta a 1 nel caso in cui, a partire dalla situazione in cui entrambi gli ingressi sono 0, si ha prima la transizione a 1 di X seguita (con X sempre al valore 1) da quella di Y. L'uscita viene riportata a 0 soltanto da una sequenza per cui, a partire dalla situazione in cui entrambi gli ingressi sono 1, si ha la transizione a 0 di X seguita (con X sempre al valore 0) da quella di Y.
La macchina deve essere dotata di reset asincrono che la riporta nello stato iniziale con uscita 0.

ESERCIZIO N°6

7 punti

Realizzare una subroutine per il microcontrollore AVR XMEGA256A3BU, che valuti la somma modulo 10^4 di due numeri di 4 cifre BCD, contenuti in X e Y, e ponga il risultato nello stesso puntatore X (che, come è noto, è costituito dalla coppia di registri R27:R26).

(1) Funzione combinatoria.

x_1x_0 \ x_3x_2	00	01	11	10
00	0	1	0	1
01	1	0	0	0
11	0	1	0	0
10	1	0	0	0

$x_4 = 0$

x_1x_0 \ x_3x_2	00	01	11	10
00	1	0	0	0
01	0	1	0	1
11	1	0	0	0
10	0	1	0	0

$x_4 = 1$

Per realizzare la mappa alloco le 10 codifiche esatte
Le rimanenti 22 corrispondono agli errori

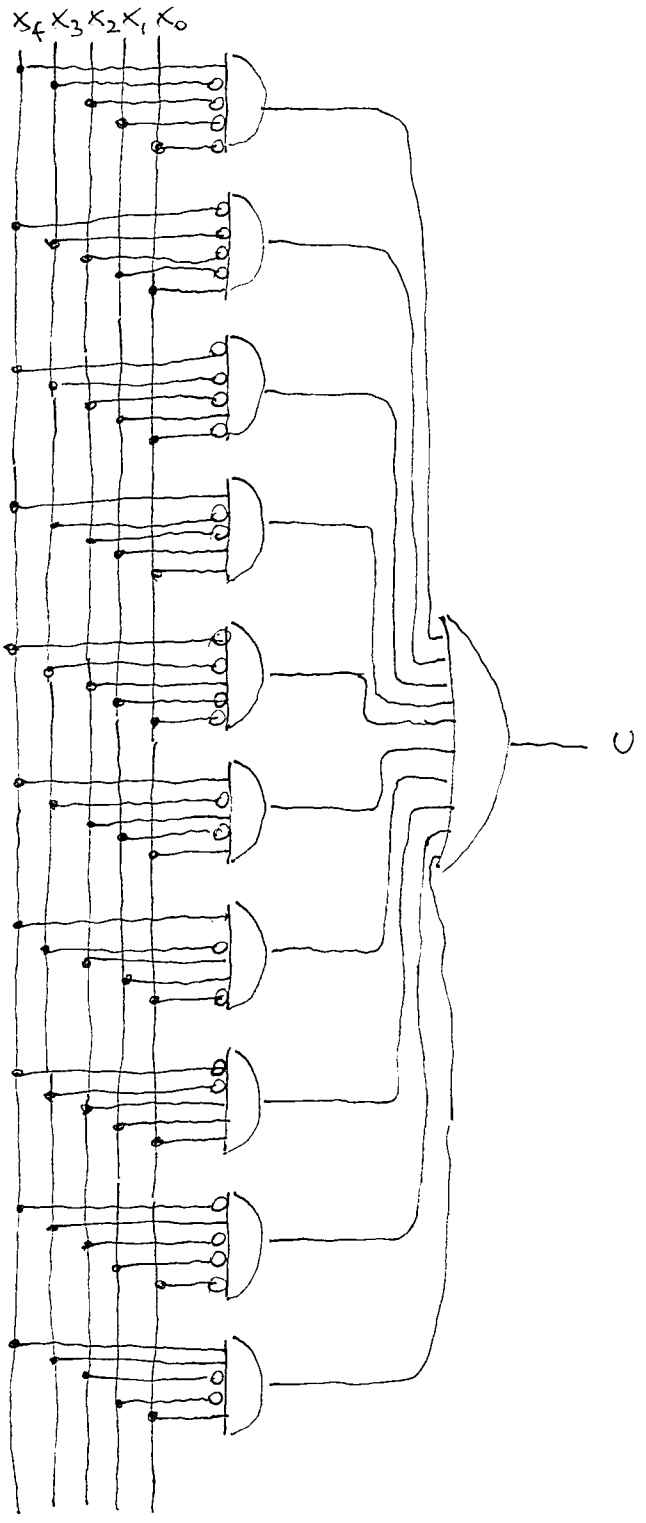
Gli 1 sono isolati - la sintesi SP a minimo numero di porte è costituita dalla somma dei 10 mintermini

Ritardo massimo: NOT + AND5 + OR10

$$t_{pd} = 0,15 + 0,2 + 0,15 + 1 + 0,15 + 2 = 3,65\ ns$$

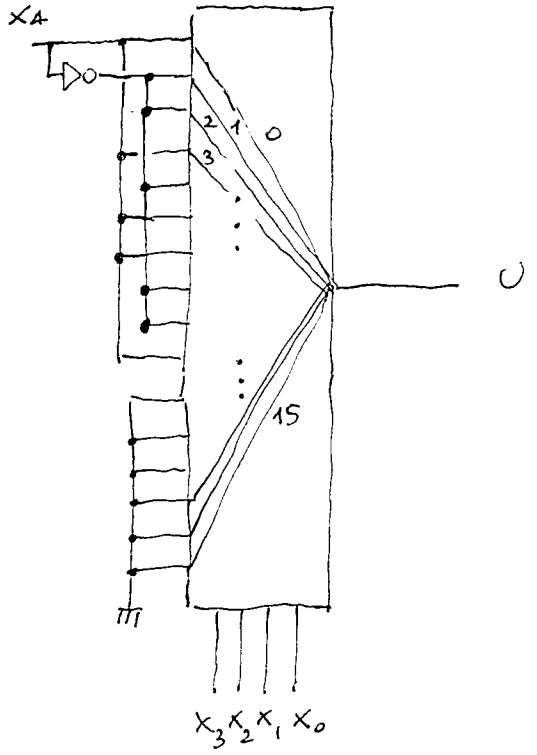
$$f_{max} = \frac{1}{T_{max}} = \frac{1}{t_{pd} + T_{co} + T_{su}} = 215\ MHz$$

Sintesi: schema a porte

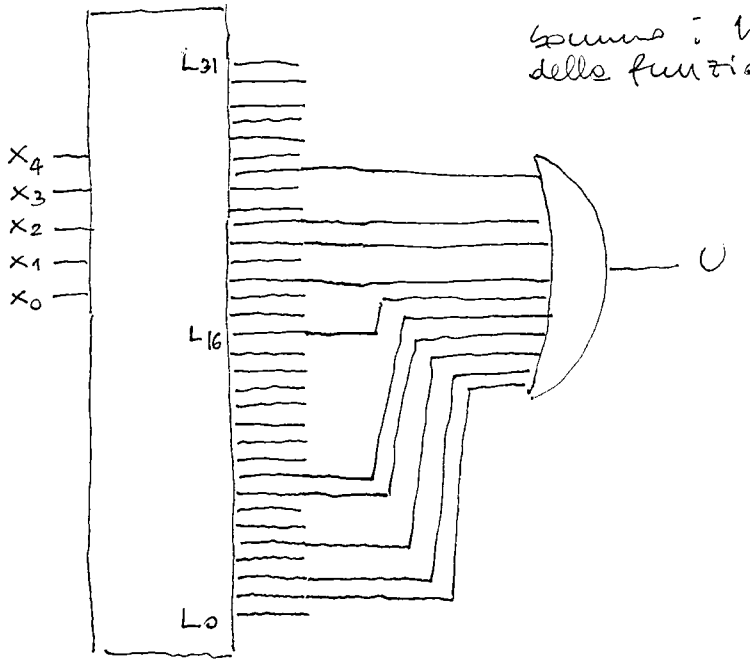


(2)

Soluzione tramite MUX
Uso un decoder 16:1



Soluzione tramite decoder 5:32



Somma: 10 mintermini della funzione

(3) Occorrono 8b per la parte intera (modulo)

3b per la parte frazionaria (esprimibile sempre in ottavi)

1b per il segno

Non essendoci casi limite, questi 12 b sono sufficienti per tutte le rappresentazioni

			8 bit	ε_A	ε_r
MS	1:1000000.110	-129.75	-128	1.75	1.35%
	0:0000000.001	0.125	0	0.125	100%
	1:0010110.101	-45.625	-44	1.625	3.56%
C2	10111111.010	-129.75	-130	0.25	0.193%
	00000000.001	0.125	0	0.125	100%
	11101001.011	-45.625	-46	0.375	0.822%
C1	10111111.001	-129.75	-128	1.75	1.35%
	00000000.001	0.125	0	0.125	100%
	11101001.010	-45.625	-44	1.625	3.56%
T	00111111.010	-129.75	-130	0.25	0.193%
	10000000.001	0.125	0	0.125	100%
	01101001.011	-45.625	-46	0.375	0.822%

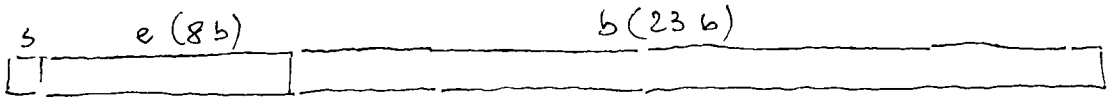
Usando solo 8b si commettono errori. Per determinarli occorre valutare i numeri rappresentati troncando i 4b meno significativi di ogni valore

$$\varepsilon_A = |x - \hat{x}| \qquad \varepsilon_r = \frac{\varepsilon_A}{|x|}$$

Nello standard IEEE 754 ; 3 valori si rappresentano SENZA errori, avendo a disposizione 23 b di mantissa.

Le rappresentazioni si ricavano (binary 32)

$$x = (-1)^s \cdot \left\{ 1 + \sum_{i=1}^{23} b_{23-i}\, 2^{-i} \right\} 2^{(e-127)} =$$



$$-129{,}75 = (-1) \cdot 2^7 \cdot \frac{129{,}75}{128}$$

$s = 1$; $e = 134$; $B = 11468\,8$

[1 1 0 0 0 0 1 1 0 | 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

$$0{,}125 = (-1)^0 \cdot 2^{-3} \cdot 1$$

$s = 0$; $e = 124$; $B = \emptyset$

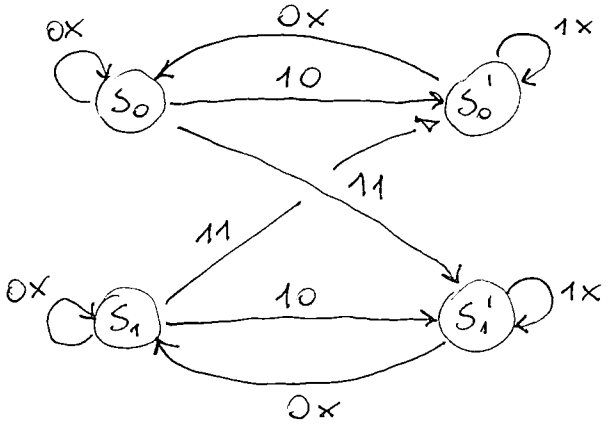
[0 | 0 1 1 1 1 1 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

$$-45{,}625 = (-1) \cdot 2^5 \cdot \frac{45{,}625}{32}$$

$s = 1$; $e = 132$; $B = 3571712$

[1 | 1 0 0 0 0 1 0 0 | 0 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]

④ Possiamo fare riferimento al seguente grafo di flusso



S_0 e S_0' uscita $\emptyset$
S_1 e S_1' uscita 1

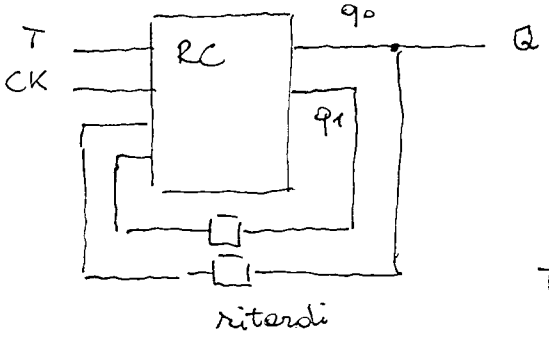
Richiediamo all'utente un pilotaggio in modo fondamentale e senza transizioni multiple

Codifichiamo con codici adiacenti, stati successivi

	$q_1\,q_0$
S_0	00
S_0'	10
S_1	11
S_1'	01

q_0 coincide con l'uscita

Architettura:



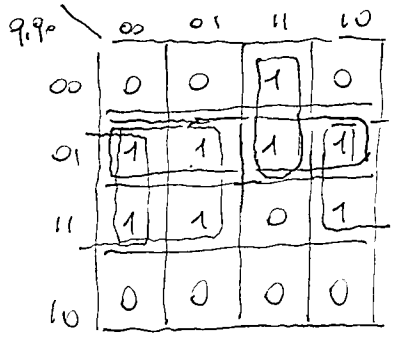
Sintetizziamo RC senza ALEE e poniamo ritardi per evitare ALEE essenziali

Tabella di flusso

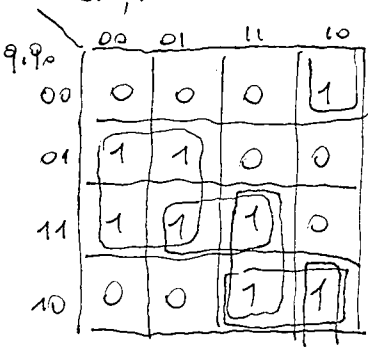
$q_1\,q_0$		CK,T = 00	01	11	10
S_0	00	(00)	(00)	01	10
S_1'	01	11	11	(01)	(01)
S_1	11	(11)	(11)	10	01
S_0'	10	00	00	(10)	(10)

Si ricavi senza alee

q_0



q_1



$$q_0 = q_0\overline{T} + q_0\overline{CK} + \overline{q_1}q_0 + \overline{q_1}CKT$$

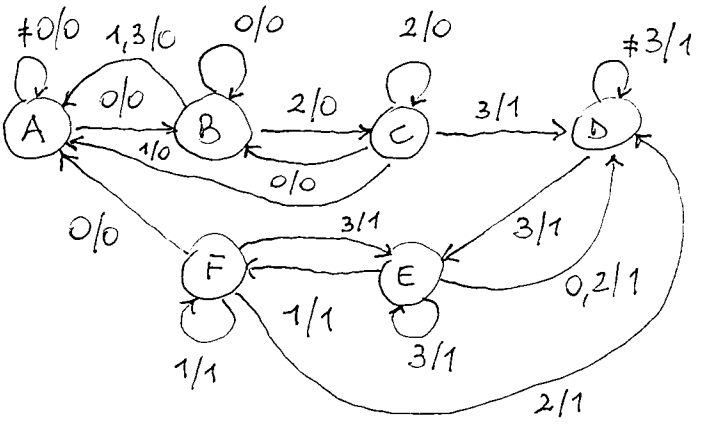
$$q_1 = q_0\overline{CK} + q_1q_0\overline{T} + q_1CKT + q_1\overline{q_0}CK + \overline{q_0}CK\overline{T}$$

Per avere la garanzia che alee essenziali non diano problemi, è sufficiente porre ritardi maggiori del massimo ritardo di RC

(5)

Grafo di flusso della macchina.

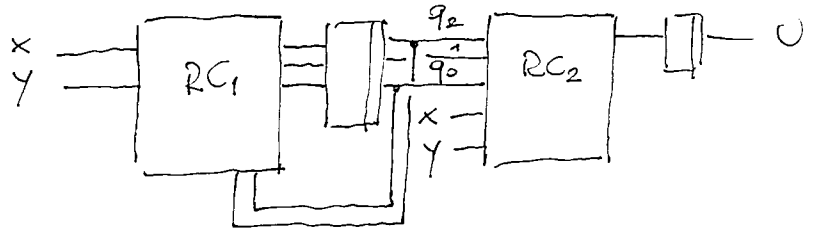
XY	
00	0
01	1
10	2
11	3



Codifica degli stati (senza alcuna tecnica di ottim.)

A	000	stato di reset
B	001	
C	010	
D	011	
E	100	
F	101	

Architettura



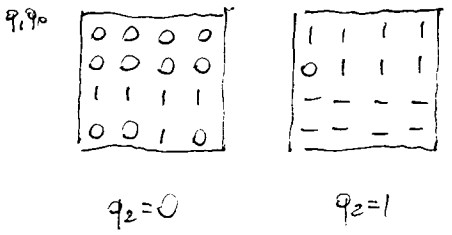
q_1q_0 \ XY	00	01	11	10	
00	001/0	000/0	000/0	000/0	A
01	001/0	000/0	000/0	010/0	B
11	011/1	011/1	100/1	011/1	D
10	001/0	000/0	011/1	010/0	C
	0	1	3	2	

$q_2 = 0$

q_1q_0 \ XY	00	01	11	10	
00	011/1	101/1	100/1	011/1	E
01	000/0	101/1	100/1	011/1	F
11	–	–	–	–	–
10	–	–	–	–	–
	0	1	3	2	

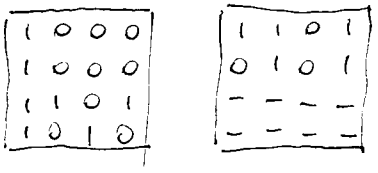
$q_2 = 1$

Sinteza RC_2



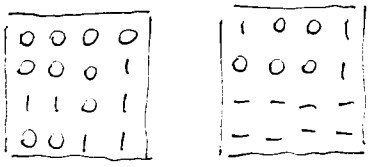
$$U = q_1q_0 + xyq_1 + q_2\bar{q}_0 + yq_2 + xq_2$$

Sinteza RC_1 q_0



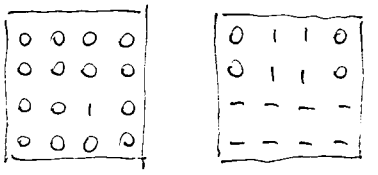
$$q_0 = \bar{x}\bar{y}\bar{q}_2 + \bar{x}q_1q_0 + \bar{y}q_1q_0 + xyq_1\bar{q}_0 + \bar{x}\bar{y}\bar{q}_0 + \bar{x}yq_2 + x\bar{y}q_2$$

q_1



$$q_1 = \bar{x}q_1q_0 + xq_1\bar{q}_0 + x\bar{y}q_0 + \bar{y}q_2\bar{q}_0$$

q_2



$$q_2 = xyq_1q_0 + yq_2$$

6

Realizzare una subroutine per il microcontrollore AVR XMEGA256A3BU, che valuti la somma modulo 10^4 di due numeri di 4 cifre BCD, contenuti in `X` e `Y`, e ponga il risultato nello stesso puntatore `X` (che, come è noto, è costituito dalla coppia di registri `R27:R26`).

```
/* La soluzione proposta spezza la difficoltà del problema in quello della somma
   delle due parti separate delle parole a 16 b. La soluzione può essere estesa
   facilmente a una dimensione arbitraria dei numeri BCD da sommare
*/


sum_4bcd:
   push R16
   push R17
   mov R16,XL
   mov R17,YL
   rcall sum_2bcd
   mov XL,R16
   mov R16,XH
   mov R17,YH
   rcall sum_2bcd_c
   mov XH,R16
   pop R17
   pop R16
   ret


/* Le seguenti subroutine sono usate solo internamente alla subroutine
   principale. Si possono quindi permettere di non salvare i registri usati,
   la cui integrità è garantita dalla subroutine chiamante.
*/


sum_2bcd: //somma due byte BCD e genera il carry corretto
   subi R16,-0x66
   add R16,R17
   brhc nn1
   brcc nn3
      ret //C=1,H=1
   nn1:
   brcc nn2
      subi R16,0x06 //C=1,H=0
      sec //per lasciare C=1
      ret
   nn2:
      subi R16,0x66 //C=0,H=0
      ret
   nn3:
      subi R16,0x60 //C=0,H=1
      ret
```

```
sum_2bcd_c: //somma due byte BCD col carry e genera il carry nuovo corretto
   brcc pp0
      inc R17
   pp0:
   subi R16,-0x66
   add R16,R17
   brhc pp1
   brcc pp3
      ret //C=1,H=1
   pp1:
   brcc pp2
      subi R16,0x06 //C=1,H=0
      sec //per lasciare C=1, nel caso multibyte
      ret
   pp2:
      subi R16,0x66 //C=0,H=0
      ret
   pp3:
      subi R16,0x60 //C=0,H=1
      ret
```