SCHEDA **ASE1801**	Data: **08 Gennaio 2018**
Cognome	Nome

ESERCIZIO N°1
8 punti

Realizzare una subroutine per il microcontrollore AVR XMEGA256A3BU, che lascia in R16 il valore di X modulo 13. Determinarne il tempo massimo di esecuzione in cicli di clock. Individuare se possibile una soluzione che, per qualsiasi valore di X, completa il compito con un numero di cicli di clock inferiore a 100.

ESERCIZIO N°2
6 punti

Sintetizzare una rete sequenziale sincronizzata secondo il modello di Moore, con in ingresso *IN* e una uscita *U*, in grado di riconoscere ponendo *U* a 1 per un ciclo di clock, le due sequenza, non interallacciate in alcun modo, 11011 e 10101. Minimizzare il numero di flip-flop usati.

ESERCIZIO N°3
4 punti

Avendo a disposizione chip di memoria SRAM da 2 M x 5 (costo 0,42 €) e da 1 M x 4 (costo 0,21 €), progettare un modulo di memoria da 4 M x 16 a costo minimo.

ESERCIZIO N°4
5 punti

Sintetizzare in forma ottima PS la rete combinatoria a 4 ingressi x_3, x_2, x_1, x_0 (che danno il valore binario di una cifra esadecimale) in grado di pilotare il segmento *f* di un display a 7 segmenti.

ESERCIZIO N°5
4 punti

Disegnare lo schema logico di un sequenziatore in grado di implementare il seguente microcodice. BGN è lo stato iniziale e va codificato con 0.

```
BGN: IF L THEN RST ELSE BGN; OP = 1000
GRP: IF K THEN ALT ELSE DWN; OP = 0000
CDN: IF M THEN GRP ELSE BGN; OP = 1110
DWN: IF M THEN FOR ELSE FOR; OP = 0011
ALT: IF L THEN RST ELSE ALT; OP = 1101
END: IF L THEN END ELSE CDN; OP = 1111
FOR: IF K THEN DWN ELSE RST; OP = 0101
RST: IF M THEN BGN ELSE END; OP = 0011
```

ESERCIZIO N°6
6 punti

Siano dati i 3 valori positivi $a = 1{,}88 \cdot 10^6$; $b = 1{,}13 \cdot 10^6$; $c = 5{,}418 \cdot 10^{13}$

a) Determinare la rappresentazione in virgola mobile IEEE754-2008 (binary32) dei 3 numeri (con arrotondamento al numero di macchina più vicino).

b) Valutare la rappresentazione della somma *s* tra i 3 numeri, eseguita come $s = a + (b + c)$ e come $s' = (a + b) + c$; i risultati parziali e quello finale sono ricondotti a numeri di macchina usando l'arrotondamento classico se necessario.

1

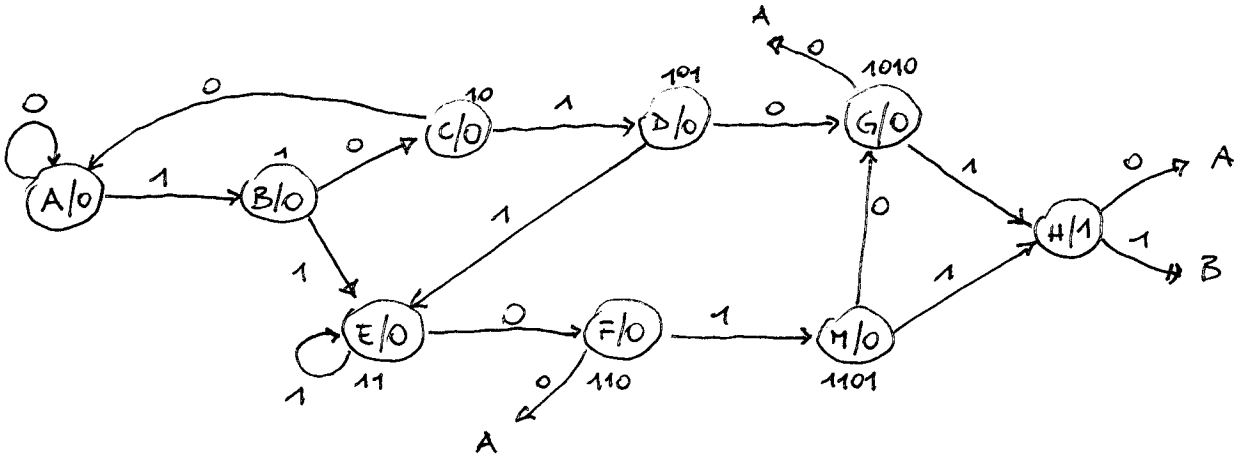
Realizzare una subroutine per il microcontrollore AVR XMEGA256A3BU, che lascia in R16 il valore di X modulo 13. Determinarne il tempo massimo di esecuzione in cicli di clock. Individuare se possibile una soluzione che, per qualsiasi valore di X, completa il compito con un numero di cicli di clock inferiore a 100.

```
/*
La soluzione seguente, gia` proposta e testata nel passato, funziona
correttamente ma sicuramente richiede tempi di esecuzione inaccettabili. Infatti
il massimo numero di iterazioni si ottiene con valori di X prossimi al massimo
(>=65533). Per questi valori, il tempo di esecuzione (compresa la rcall e la
ret) vale 20184 T. */

 mod13:
 push XH
 push XL
 m1:
  sbiw XH:XL,13
  brcc m1
 adiw XH:XL,13
 mov R16,XL
 pop XL
 pop XH
 ret
```

```
/*
Questa nuova soluzione, più articolata, sfrutta le proprietà del modulo per
ottenere un risparmio in tempo di esecuzione, ottenendo il risultato con 35
istruzioni. Evitando l'uso di loop dal numero di iterazioni dipendenti dal dato,
il tempo di esecuzione non cambia coi dati e vale 43 T (compresa rcall e ret),
soddisfacendo i requisiti assegnati. */

 mod13speed:
 push XL
 mov R16,XL
 swap R16
 andi R16,0x0F
 andi XL,0x0F
 add XL,R16
 lsl R16
 add XL,R16 //in XL ci va XLL+3XLH
 mov R16,XH
 swap R16
 andi R16,0x0F
 add XL,R16 //in XL ora c'è XLL+3XLH+XHH
 mov R16,XH
 andi R16,0X0F
 add XL,R16
 swap R16
 lsr R16
 add R16,XL //in R16 ora c'è XLL+3XLH+9XHL+XHH (al max 210)
 subi R16,208
 brcc lp01
 subi R16,-208
```

```
lp01: subi R16,104
brcc lp02
subi R16,-104
lp02: subi R16,52
brcc lp03
subi R16,-52
lp03: subi R16,26
brcc lp04
subi R16,-26
lp04: subi r16,13
brcc lp05
subi R16,-13
lp05: pop XL
ret //ora sicuramente siamo tra 0 e 12
```

(2) Partiamo dal grafo della macchina

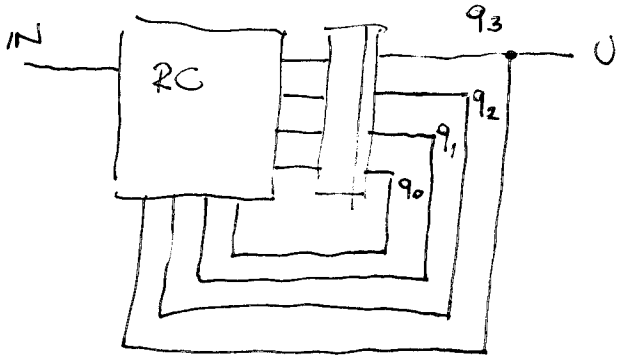


A un'ispezione visiva appare che il grafo non presenta stati equivalenti. Si hanno quindi 4 variabili di stato.

Codifica degli stati (A: stato di reset)

	q_3	q_2	q_1	q_0	U ($\equiv q_3$)
A*	0	0	0	0	0
B	0	0	0	1	0
C	0	0	1	1	0
E	0	1	0	1	0
D	0	0	1	0	0
F	0	1	1	1	0
G	0	1	1	0	0
M	0	1	0	0	0
H	1	0	0	0	1

Architettura



Mappe delle transizioni

q_1q_0 \ q_3q_2	00	01	11	10
00	A 0000	M 0110	–	H 0000
01	B 0011	E 0111	–	–
11	C 0000	F 0000	–	–
10	D 0110	G 0000	–	–

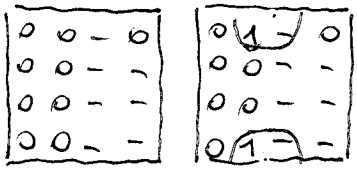
IN=0

00	01	11	10
A 0001	M 1000	–	H 0001
B 0101	E 0101	–	–
C 0010	F 0100	–	–
D 0101	G 1000	–	–

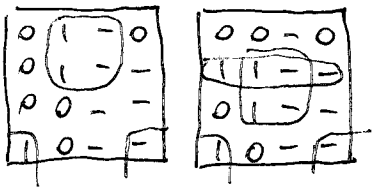
IN=1

Sintesi

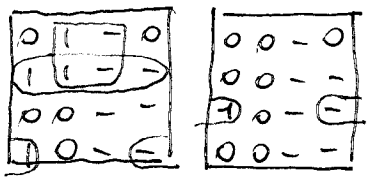
$D_3 = IN\ q_2 \bar{q}_0$



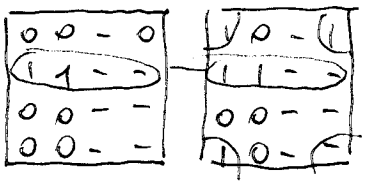
$D_2 = \bar{IN}\ q_2 \bar{q}_1 + \bar{q}_2 q_1 \bar{q}_0 + IN\ \bar{q}_1 q_0 + IN\ q_2 q_0$



$D_1 = \bar{IN}\ q_2 \bar{q}_1 + \bar{IN}\ \bar{q}_1 q_0 + \bar{IN}\ \bar{q}_2 q_1 \bar{q}_0 + IN\ \bar{q}_2 q_1 q_0$



$D_0 = \bar{q}_1 q_0 + IN\ \bar{q}_2 \bar{q}_0$



Analisi degli stati definiti dopo la sintesi

q_1q_0 \ q_3q_2	00	01	11	10
00			0110	
01			0111	0011
11			0000	0000
10			0000	0110

00	01	11	10
		1000	
		0101	0101
		0100	0010
		1000	0101

tutti gli stati futuri individuati sono già definiti nel grafo. Non si avranno, anche in caso di errore della rete, situazioni anomale NON transitorie.

(3) Esaminiamo il costo per 4M parole (ottenute con chip da 4 o da 5 bit per parola)

4M×4 → 4 chip da 1M×4 → 0,84 €

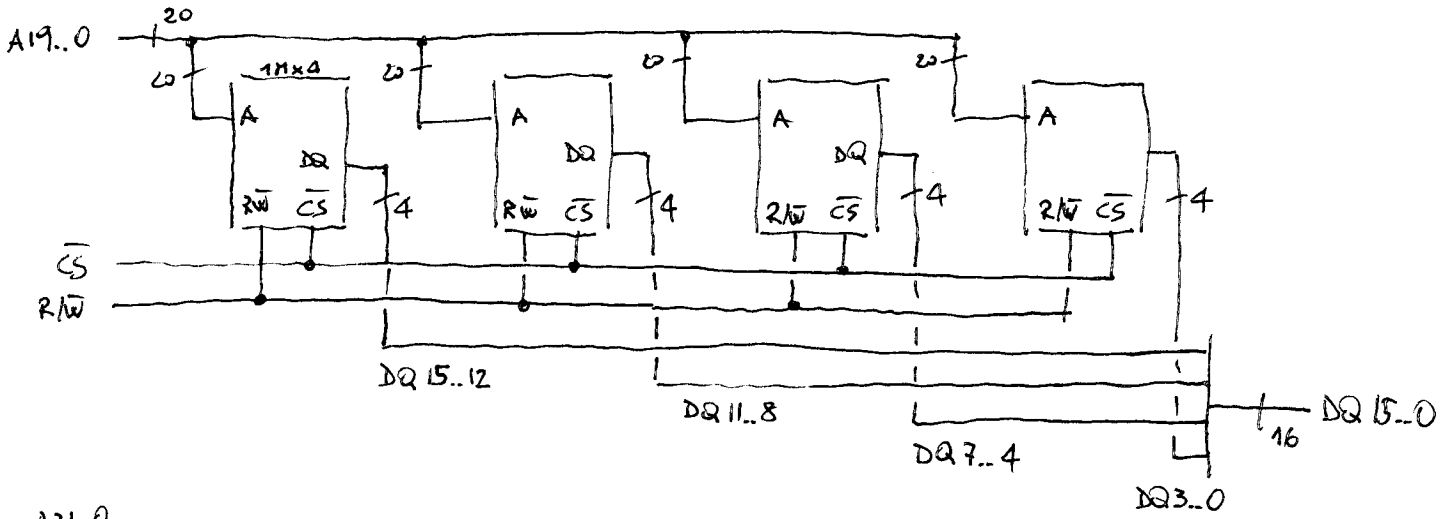
4M×5 → 2 chip da 2M×5 → 0,84 €

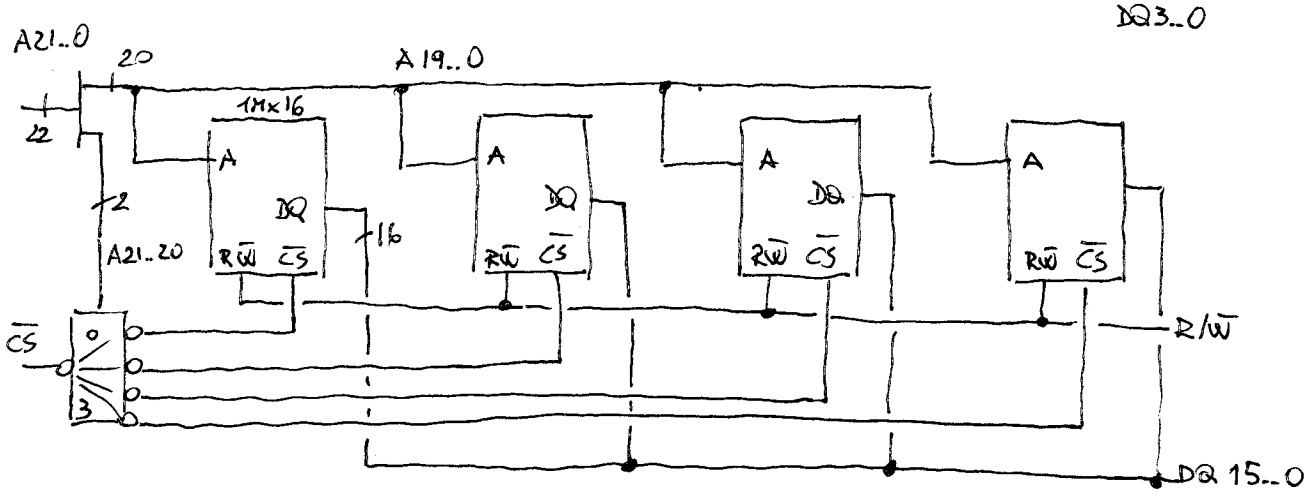
le chip da 5bit/parola è molto più economico, ma per realizzare parole da 16 b si sprecano comunque dei bit

Soluzioni possibili

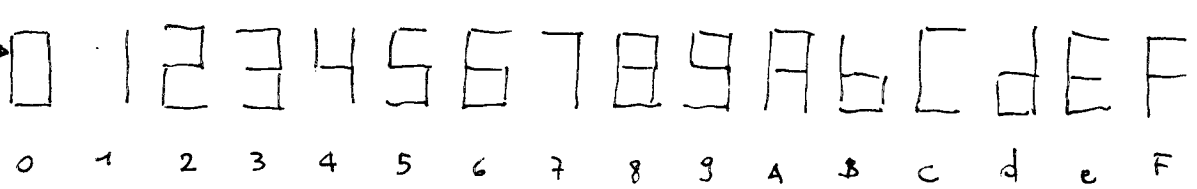
moduli	4M×4	4M×5	totale	costo
#	4	0	4M×16	3,36 € per tutti i casi
	3	1	4M×17	
	2	2	4M×18	
	1	3	4M×19	
	0	4	4M×20	

Realizzo il caso 1 (senza spreco di bit) aumentando prima la dimensione di parola e poi il n° di parole)





(4) Output a 7 segmenti (da hex)



Mappo.

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

Tutti essenziali

$$S_f = (x_3 + x_2 + \bar{x}_0)(x_3 + x_2 + \bar{x}_1)(x_3 + \bar{x}_1 + \bar{x}_0)(\bar{x}_3 + \bar{x}_2 + x_1 + \bar{x}_0)$$

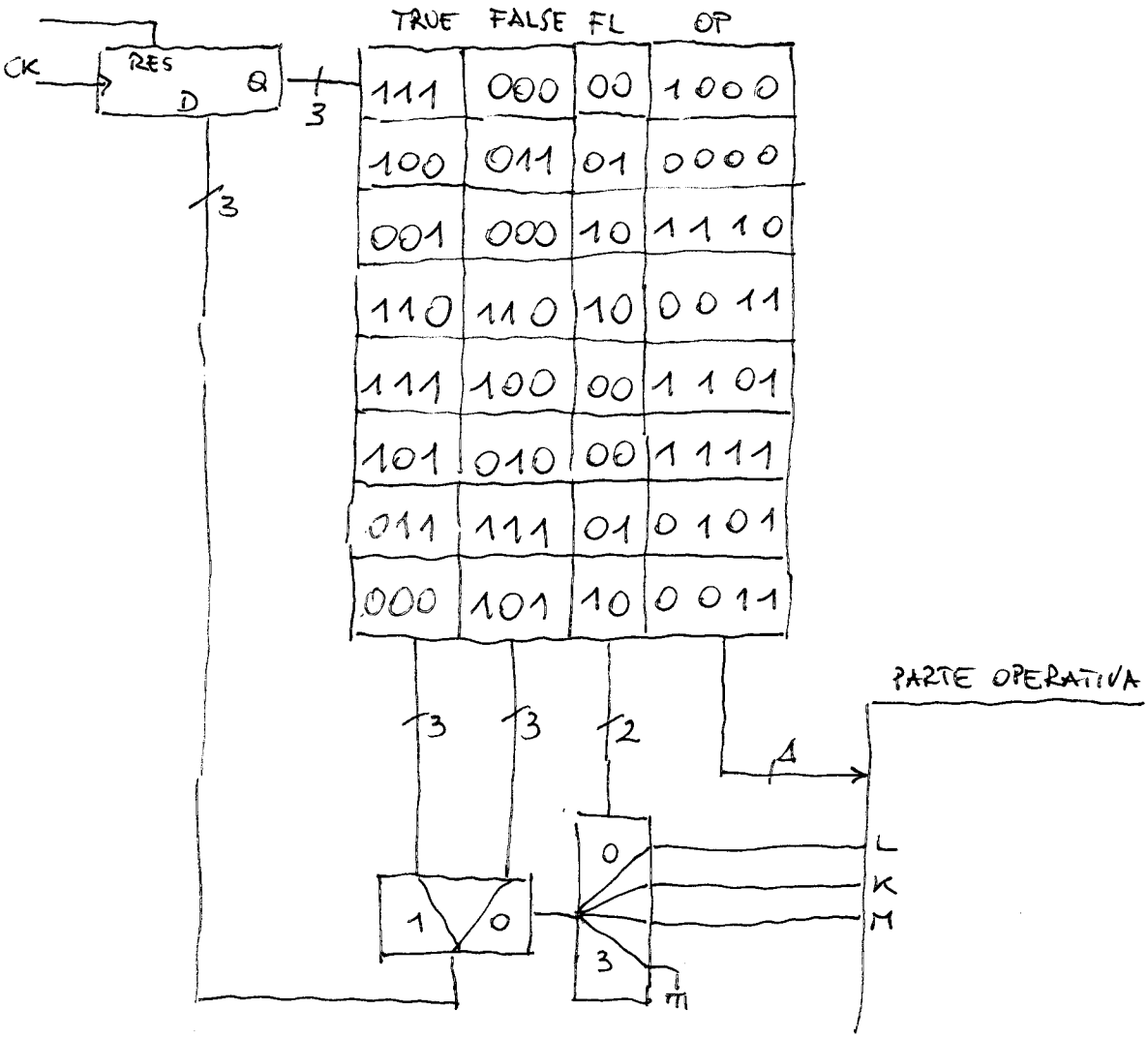
(5) Uso una struttura generica con registro

codifica stati

BGN : 000
GRP : 001
CDN : 010
DWN : 011
ALT : 100
END : 101
FOR : 110
RST : 111

flag

L : 00
K : 01
M : 10



(6) legge di rappresentazione (valori normalizzabili)

$$x = (-1)^S \, 2^{E-127} \left(1 + T\,2^{-23}\right)$$

$a = 1{,}88 \cdot 10^6 = (-1)^0 \cdot 2^{20} \cdot (1 + 6651392 \cdot 2^{-23})$

$0_x\,1001\,0011_x\,|\,|\,0.0101.0111.1101.0000.0000_x$

$b = 1{,}13 \cdot 10^6 = (-1)^0 \cdot 2^{20} \left(1 + 651392 \cdot 2^{-23}\right)$

$0_x\,1001\,0011_x\,000.1001.1111.0000.1000.0000_x$

$c = 5{,}418 \cdot 10^{13} = (-1)^0 \cdot 2^{45} \left(1 + 4528911 \cdot 2^{-23}\right)$ (a22.)

$0_x\,1010\,1100_x\,100.0101.0001.1011.0000.1111_x$

Vista la differenza tra ordini di grandezza, si ha

$a + (b+c) = a + c = c$ (la codifica coincide con quella di $-c$)

Nell'altro caso, invece

$(a+b) = (-1)^0 \, 2^{21} \left(1 + 3651392 \cdot 2^{-23}\right)$

Sommato a $-c$ fa scattare l'arrotondamento al valore successivo della mantissa di a
Quindi la codifica di S' è

$0_x\,1010\,1100_x\,100.0101.0001.1011.0001.0000_x$