SCHEDA **ASE2205**	Data: **06 Giugno 2022**

Il testo deve essere riconsegnato nella cartellina. Non è ammessa la consultazione degli appunti e dei compiti precedenti. Si possono consultare i data sheet. Non usare il colore rosso nello svolgimento.

ESERCIZIO N°1

5 punti

Una riga e una colonna della seguente tabella di numeri binari sono state poste in modo da avere sempre (H e V) parità pari. Determinare se la tabella contiene sicuramente errori e, in tal caso, quale cella andrebbe corretta per ottenere il più probabile valore originale della tabella senza errori.

1	0	0	1	0	1	1	0
0	0	0	0	0	1	1	0
1	1	0	0	1	1	0	0
1	1	0	0	1	1	0	1
1	0	0	1	0	0	0	0
1	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1
1	1	0	0	1	0	0	1

ESERCIZIO N°2

8 punti

Realizzare un sottoprogramma per il microcontrollore AVR XMEGA256A3BU che valuta il prodotto scalare di 2 vettori da 64 valori interi (su un byte, con segno) contenuti in memoria a partire dagli indirizzi in X e Y. Le operazioni vanno eseguite scegliendo per il risultato il numero minimo di byte che garantisce in ogni caso la rappresentabilità. Il risultato deve essere posto in memoria a partire dall'indirizzo contenuto in Z (partendo dal byte meno significativo).

ESERCIZIO N°3

4 punti

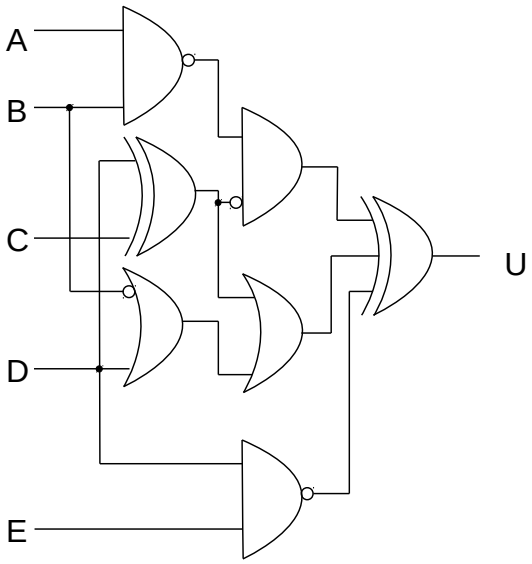
Codificare (facendo eventualmente ricorso al data sheet) in valore binario ed esadecimale l'insieme delle tre seguenti istruzioni assembly di un microcontrollore XMEGA. Individuare tutti i valori iniziali di R20 affinché il segmento di codice non resti bloccato permanentemente nel loop.

```
loop: ANDI R20,16
      SBRS R20,5
      BRNE loop
```

ESERCIZIO N°4

5 punti

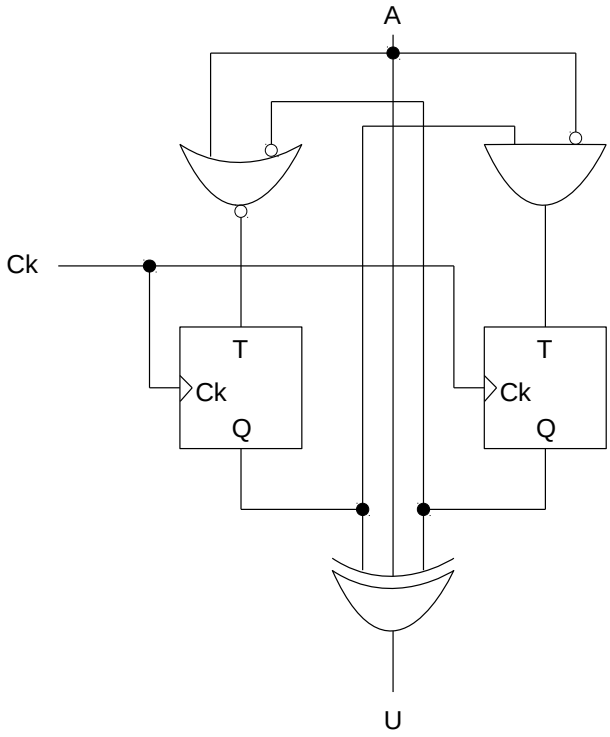
Realizzare in forma NAND-NAND ottima la seguente rete combinatoria.



ESERCIZIO N°5

5 punti

Determinare tipologia architetturale e disegnare coerentemente il grafo di flusso della seguente macchina sequenziale sincrona.



ESERCIZIO N°6

6 punti

Progettare una rete combinatoria, facente uso di blocchi noti (porte logiche elementari, multiplexer, full-adder, ecc.) in grado di eseguire la somma di 3 numeri relativi a 8 bit, rappresentati in C1 e il cui risultato (per il risultato scegliere il numero di bit minimo che garantisce in ogni caso la rappresentabilità) sia rappresentato in MS.

1

Nel seguente blocco di dati (assegnato) si verifica
la violazione della parità pari in una riga e una colonna,
indicate dalla X in rosso

1	0	0	1	0	1	1	0	✓
0	0	0	0	0	1	1	0	✓
1	1	0	0	1	1	0	0	✓
1	1	0	0	1	1	0	1	✗
1	0	0	1	0	0	0	0	✓
1	1	1	0	0	0	0	1	✓
0	0	1	0	1	1	0	1	✓
1	1	0	0	1	0	0	1	✓

Trasformando il bit 1 nel cerchietto rosso in 0, tutte le
righe e le colonne rispettano la parità pari.
Con elevata probabilità (a meno di non ipotizzare una
molteplicità di errori) questa è la tabella corretta.

2

```
/* Prodotto scalare tra vettori di 64 elementi
   con segno da 1 byte [-128..127].
   Il range del risultato sara` [-128*127*64..128*128*64]
   e per rappresentarlo in ogni caso servono 3 byte, che
   coprono un range da [-2^23..2^23-1]
*/
scalar:
  push R0
  push R1 //per il prodotto
  push R2 //estensione del segno
  push R16 //contatore
  push R18 //appoggio per gli elementi
  push R19
  push R20 //byte meno significativo del risultato
  push R21
  push R22 //byte piu` significativo del risultato
  ldi R16,64
  loop:
    ld R18,X+
    ld R19,Y+
    muls R18,R19
    sbc R2,R2 //estensione del segno sul terzo byte
    add R20,R0
    adc R21,R1
    adc R22,R2
    dec R16
    brne loop
  st Z,R20 //sistemazione del risultato
  std Z+1,R21
  std Z+2,R22
  subi XL,64
  sbci XH,0
  subi YL,64
  sbci YH,0 //ripristino dei puntatori
  pop R22 //ripristino dei registri
  pop R21
  pop R20
  pop R19
  pop R18
  pop R16
  pop R2
  pop R1
  pop R0
  ret
```

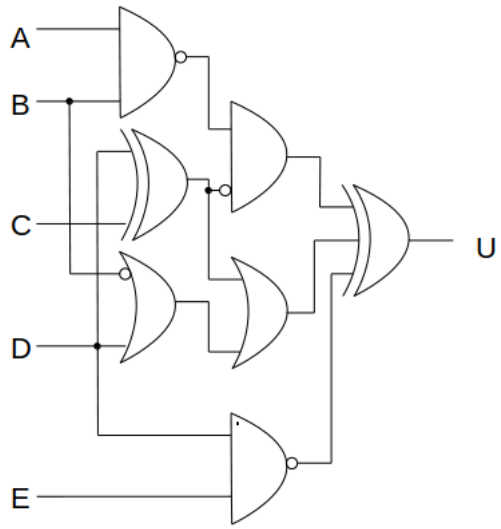
3

```
loop:
  ANDI R20,16  //codifica 0b0111 0001 0100 0000; 0x7140
  SBRS R20,5   //codifica 0b1111 1111 0100 0101; 0xFF45
  BRNE loop    //codifica 0b1111 0111 1110 1001; 0xF7E9
```

infatti R20->16+4 (nelle immediate il 16 è implicito)
e nella branch k=-3 (da esprimere in C2)

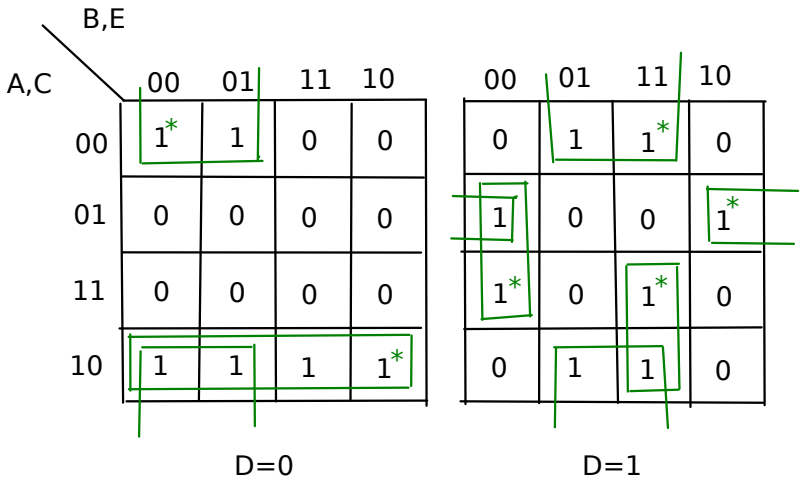
Riguardo all'esecuzione, se chiamiamo R7..R0 i bit di R20, dopo la ANDI abbiamo 0 0 0 R4 0 0 0 0; essendo il quinto bit nullo (R5 and 0) la skip non ha mai effetto.
Si esce dal loop se e solo se R4=0 fin dall'inizio, indipendentemente dal valore degli altri bit di R20.

4



Applicando la scomposizione di Shannon (# è la xor)

se D=0
U=((A!+B!)C!#(B!+C))!
se B=0
U=C!
se B=1
U=A!C!#C!=AC!
se D=1
U=E#(A!+B!)C
se E=0
U=(A!+B!)C
se E=1
U=AB+C!

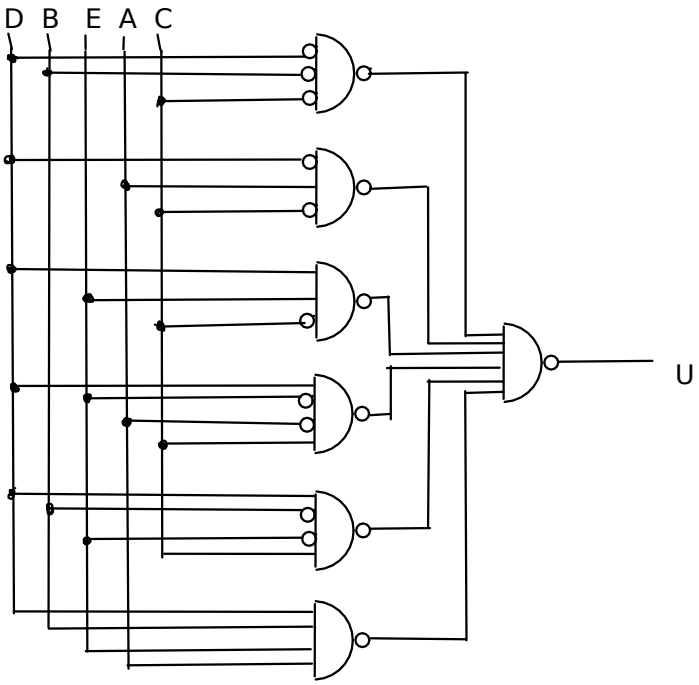


Sono tutti implicanti essenziali, che coprono tutti i mintermini della funzione

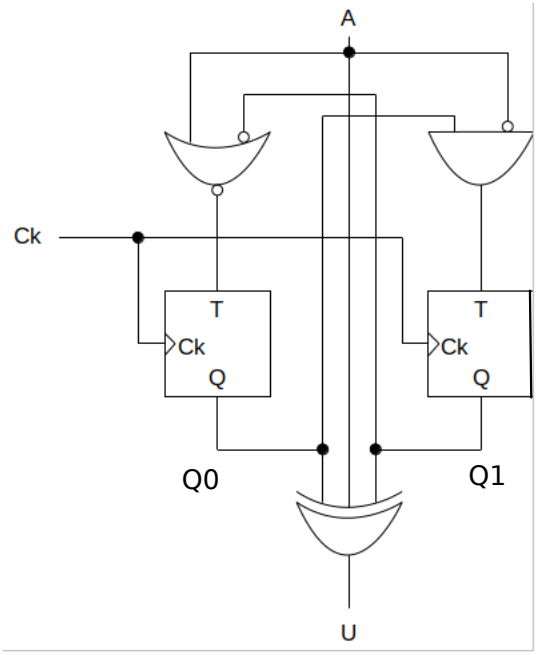
Forma SP

U=D!B!C!+D!AC!+DEC!+DE!A!C+DB!E!C+DBEA

Schema NAND-NAND (applico il teorema di De Morgan)



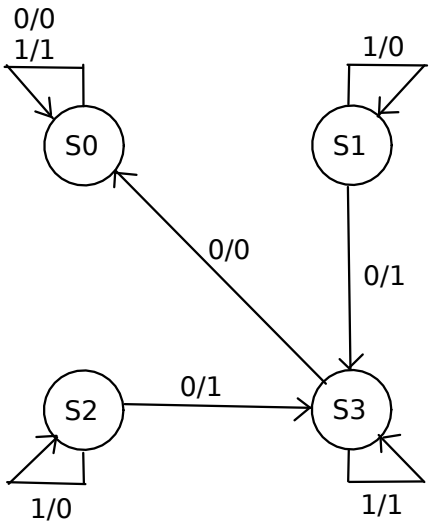
5



L'uscita dipende direttamente dall'ingresso A
Si tratta di una macchina di Mealy

Tabella di eccitazione/transizione
T1=A!Q0
T0=(A+Q1!)!=A!Q1
U=Q1#Q0#A (# è la xor)

Q1	Q0	A	T1	T0	Q1'	Q0'	U
0	0	0	0	0	0	0	0
(S0)		1	0	0	0	0	1
0	1	0	1	0	1	1	1
(S1)		1	0	0	0	1	0
1	0	0	0	1	1	1	1
(S2)		1	0	0	1	0	0
1	1	0	1	1	0	0	0
(S3)		1	0	0	1	1	1



6

Per arrivare al risultato, conviene impostare la seguente successione di azioni, da eseguire in appositi blocchi operatori

- Conversione dei dati C1 in C2, su 8 bit (X, Y e Z)
- Estensione dei 3 dati su 10 bit, per non doversi preoccupare di overflow
  range risultato: [-3*127..3*127]
- Somma con 2 normali sommatori a 10 bit (per esempio ripple-carry)
- Conversione da C2 a MS