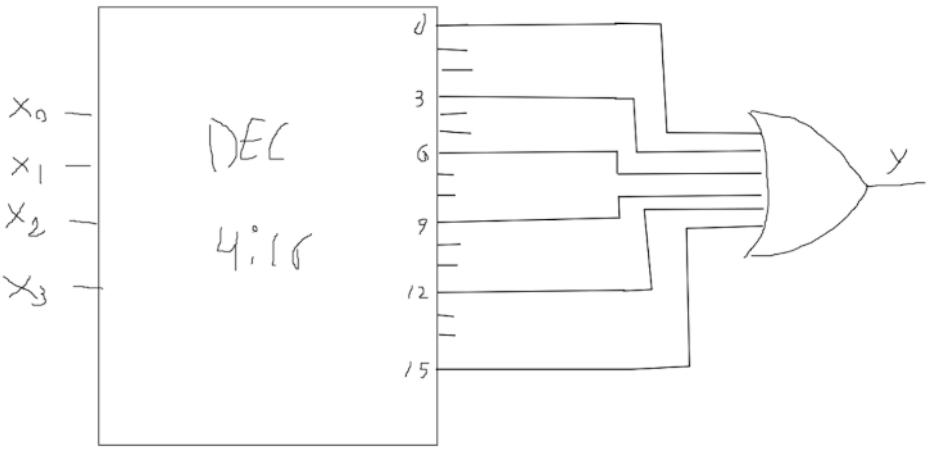
Matteo Di Gioacchino
Soluzioni Esercitazione 07/12/2023

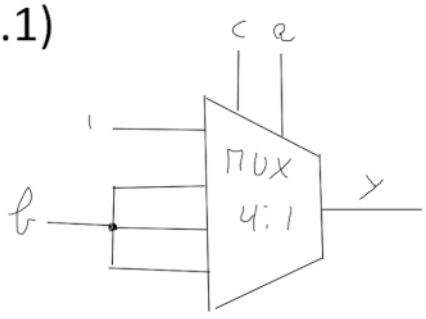
1)

X_3	X_2	X_1	X_0	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



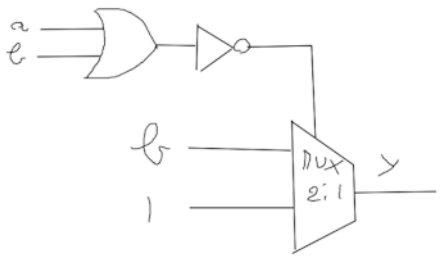
2) Y = (bc + !a!b!c + b!c) =
= b(c + !c) + !a!b!c =
= b + !a!b!c =
= b + !a!c

2.1)



2.2)

oss: $\bar{a}\bar{b} = \overline{a+b}$ (De Morgan)



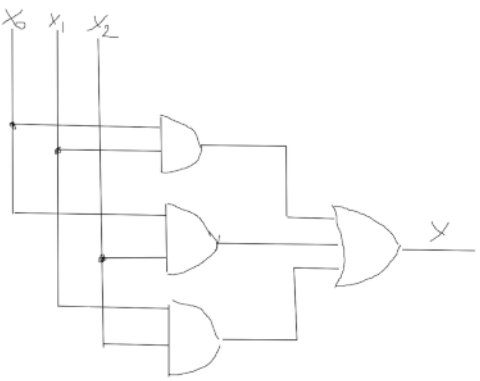
3)

x_2	x_1	x_0	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Karnaugh:

x_2 \ x_0x_1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$Y = x_0x_1 + x_0x_2 + x_1x_2$



Alternativa a Karnaugh:
Uso un DEC 3:8 similmente a come abbiamo fatto per l'ES 1

4)

$Y = ab + (ab \oplus cd) + \bar{a}d =$
$= ab + (\bar{a}bcd + ab\bar{c}\bar{d}) + \bar{a}d =$
$= ab(1 + \bar{c}\bar{d}) + \bar{a}bcd + \bar{a}d =$
$= ab + \bar{a}bcd + \bar{a}d = ab + cd + \bar{a}d$

a	b	c	d	y	
0	0	0	0	0	△
0	0	0	1	1	•
0	0	1	0	0	△
0	0	1	1	1	•
0	1	0	0	0	△
0	1	0	1	1	•
0	1	1	0	0	△
0	1	1	1	1	•
1	0	0	0	0	△
1	0	0	1	0	△
1	0	1	0	0	△
1	0	1	1	1	•
1	1	0	0	1	•
1	1	0	1	1	•
1	1	1	0	1	•
1	1	1	1	1	•

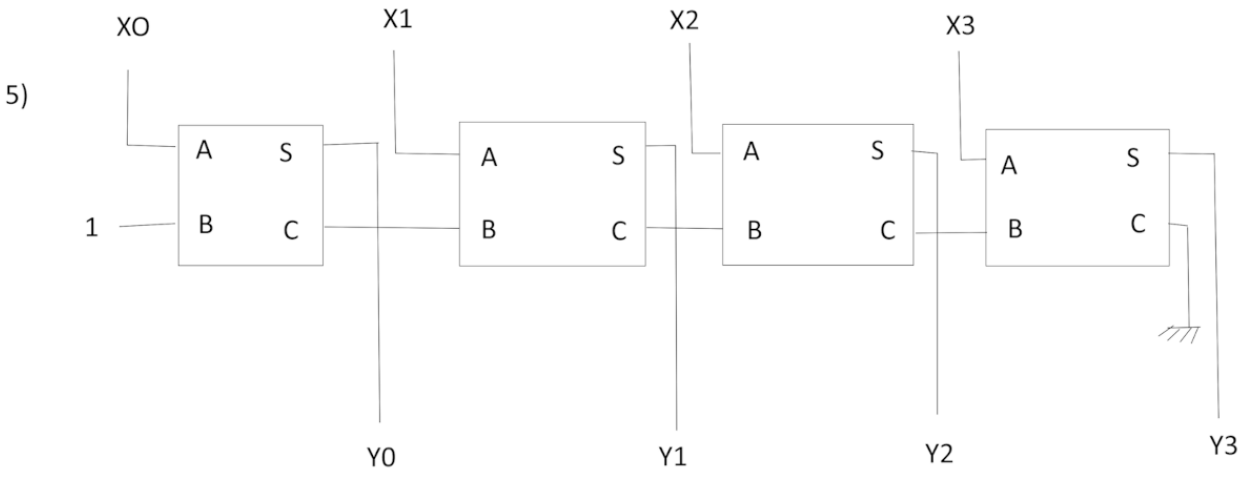
DNF: •

$y = \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d +$
$+ \bar{a}bcd + a\bar{b}cd + ab\bar{c}\bar{d} +$
$+ ab\bar{c}d + abc\bar{d} + abcd$

CNF: △

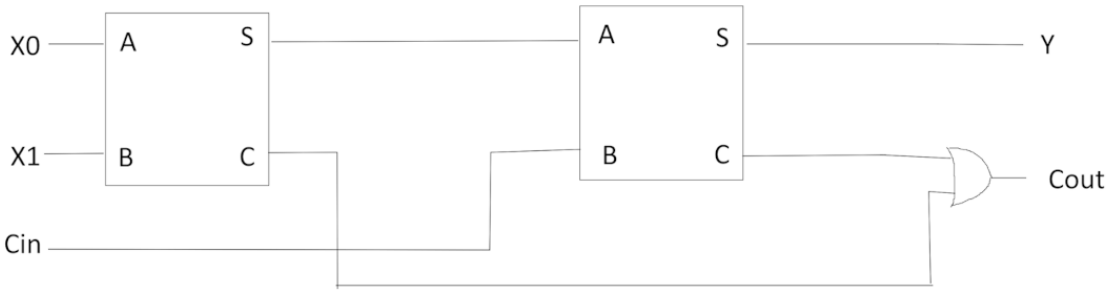
$Y = (a+b+c+d) \cdot (a+b+\bar{c}+d) \cdot$
$(a+\bar{b}+c+d) \cdot (a+\bar{b}+\bar{c}+d) \cdot$
$(\bar{a}+b+c+d) \cdot (\bar{a}+b+c+\bar{d}) \cdot (\bar{a}+b+\bar{c}+d)$

5)



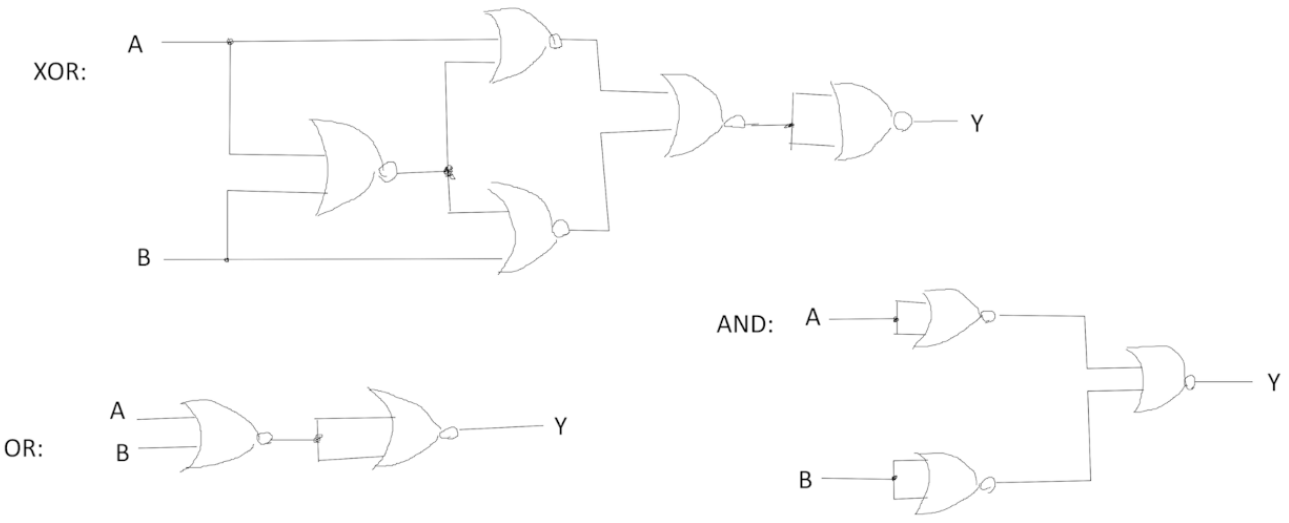
6)

STEP 1: sappiamo come passare da HALF ADDER a FULL ADDER:



STEP 2: sappiamo che per costruire un HALF ADDER ci serve 1 porta XOR ed 1 porta AND, quindi, in totale, abbiamo 2 porte XOR, 2 porte AND ed 1 porta OR, ora, se riuscissimo a costruire ognuna di queste porte usando solo la porta NOR avremmo finito!

STEP 3: Conversioni delle porte logiche

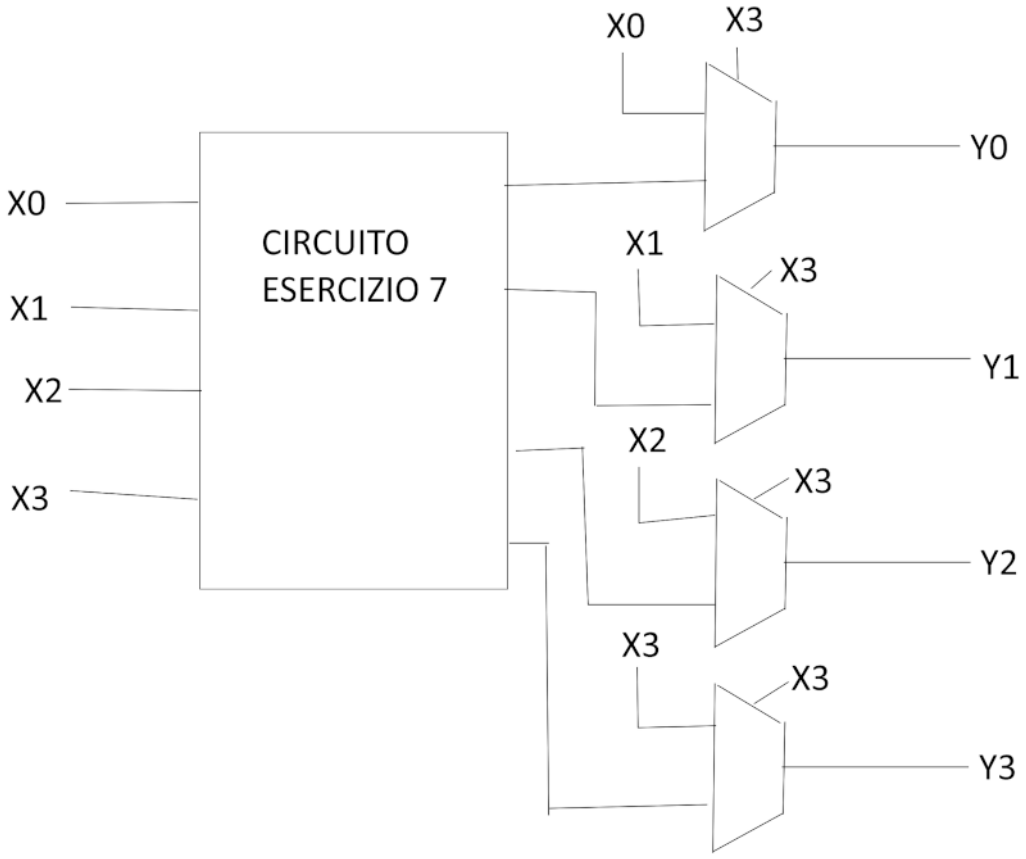


7) STEP 1: inverto tutti i bit (ci basta porre il NOT davanti ad ogni bit in input)
STEP 2: sommo 1 al numero ottenuto dallo STEP 1 (si riveda l'ESERCIZIO 5)

8) Il circuito deve fare le seguenti operazioni:

STEP 1: se il MSB è 0, allora il numero è positivo, lo ritorniamo così com'è
STEP 2: se il MSB è 1, allora il numero è negativo, per ottenerne
il valore assoluto dobbiamo invertire ogni bit e sommargli 1 (si riveda l'ESERCIZIO 7)

Osserviamo che gli step da seguire sono simili ad una IF, e ricordiamo che il circuito che imita il comportamento della IF è il MULTIPLEXER, quindi:



9) Osserviamo che (X1 >= X2) <--> (X1 - X2 >= 0), quindi possiamo prima sottrarre X2 ad X1, poi controllare se la somma è negativa o positiva, se è negativa, allora il MSB è posto ad 1, altrimenti è posto a 0 (come nel caso del complemento a 2).
Osserviamo inoltre che dovremmo utilizzare 5 bit per rappresentare un numero di 4 bit in complemento a 2 (potendo così rappresentare da -16 a +15), tuttavia, ai fini della somma, possiamo ignorare il 5o bit

ES:

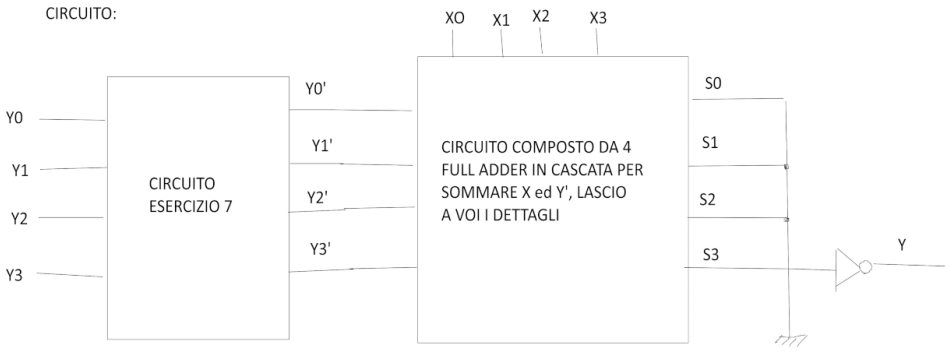
```
12 -
13 =          ==>     1100 -
------                1101 =        ==>   1101 in complemento a 2 = 10010+1 = 10011
 -1                   ---------           Vediamo che succede se ignoriamo il MSB
                      ?????
                                    ==>   1100 +
                                          0011 =
                                          ---------
                                          1111 = -1
```

```
13 -
12 =          ==>     1101 -
-----                 1100 =        ==>   Come prima, 1100 in CA2 = 10011+1 = 10100,
  1                   --------            nuovamente, ignoriamo il MSB
                      ????
                                          1101 +
                                          0100 =          ==> l'approccio funziona!
                                          --------
                                          0001
                                          (riporto di 1,
                                          ignoriamolo)
```

STEPS:

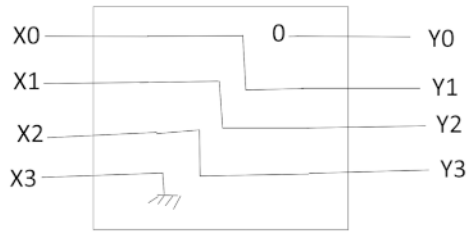
STEP 1: convertire il 2o numero in CA2 (guardare l'ESERCIZIO 7)
STEP 2: sommare i due numeri
STEP 3: controllare il MSB del risultato, se è 1 allora X2 > X1 ==> return 0
se è 0 allora X1 >= X2 ==> return 1

CIRCUITO:



1o numero = (X3-X2-X1-X0), 2o numero = (Y3-Y2-Y1-Y0), 2o numero CA2 = (Y3'-Y2'-Y1'-Y0'), somma = (S3-S2-S1-S0)

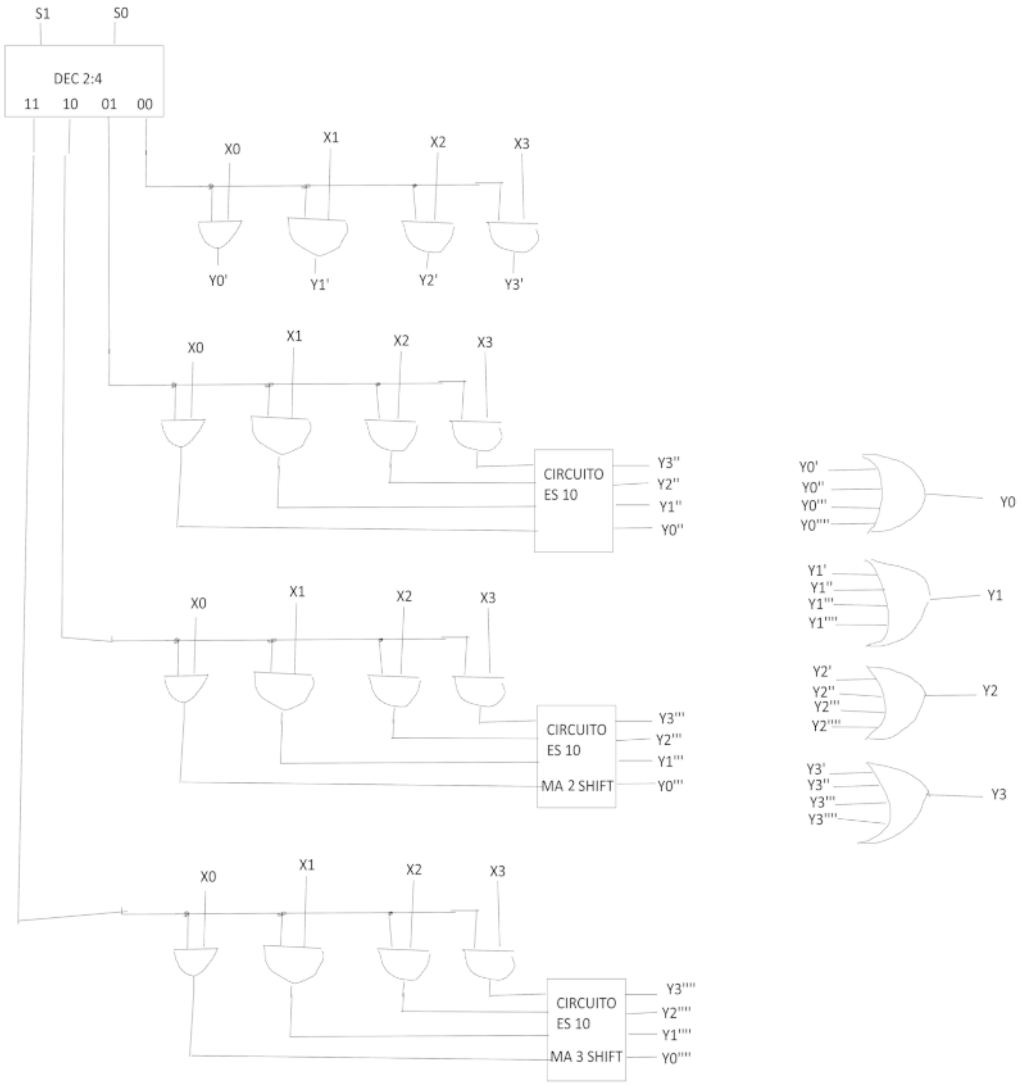
10) Shiftando il numero di 1 posizione a Sinistra stiamo effettuando una moltiplicazione x2



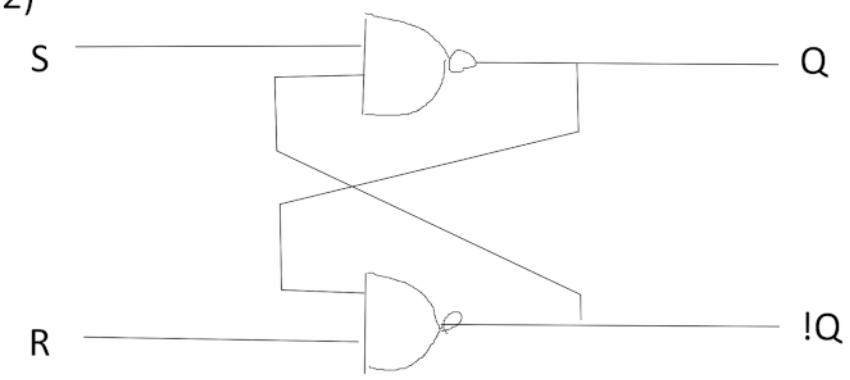
11) Come detto nell'esercizio precedente, ogni shift a Sinistra è una moltiplicazione x2, quindi se "s=0", allora ritorniamo il numero in input, se "s=1", allora moltiplichiamo il numero in input x2, se "s=2", allora moltiplichiamo il numero x4, se "s=3", allora moltiplichiamo il numero x8.

In generale (e questo vi servirà anche per il corso di Architettura dei Calcolatori), se "s = i", allora moltiplichiamo x2^i)

1 (delle tante) possibile soluzione:



12)

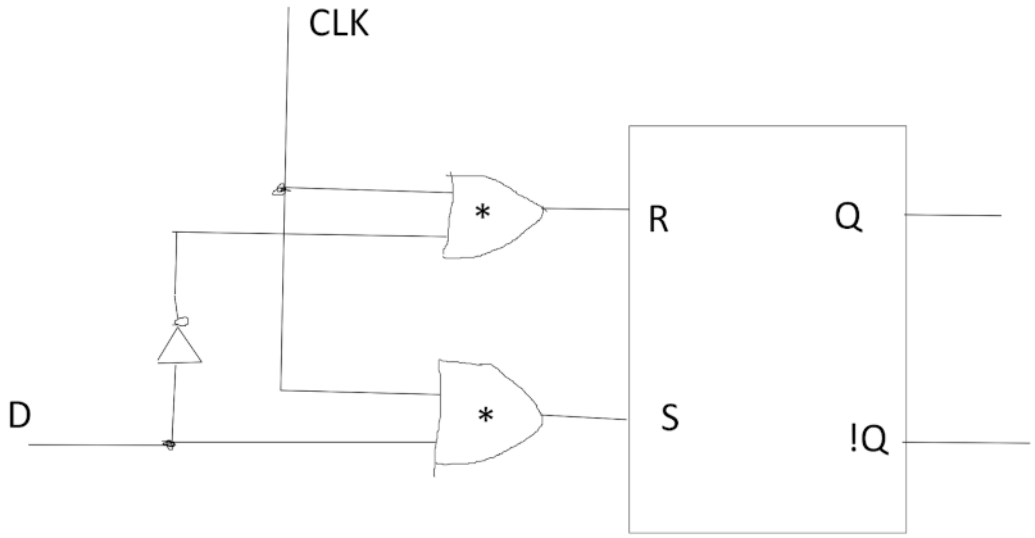


S	R	Q	!Q
O	O	??????????	
O	1	1	0
1	0	0	1
1	1	Qprec	!Qprec

Le differenze sono:

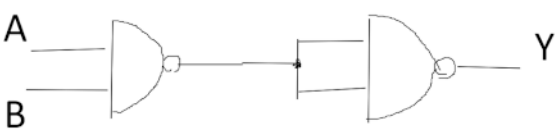
1) lo stato indesiderato ora è causato da (S=0,R=0) e non da (S=1,R=1)
2) lo stato di memoria ora è causato da (S=1,R=1) e non da (S=0,R=0)
3) Set e Reset sono "invertiti" rispetto all'SR-Latch con le porte NOR

Sappiamo costruire un SR-Latch con le porte NAND, ora, per costruire un D-FlipFlop con NAND e NOT dobbiamo fare:

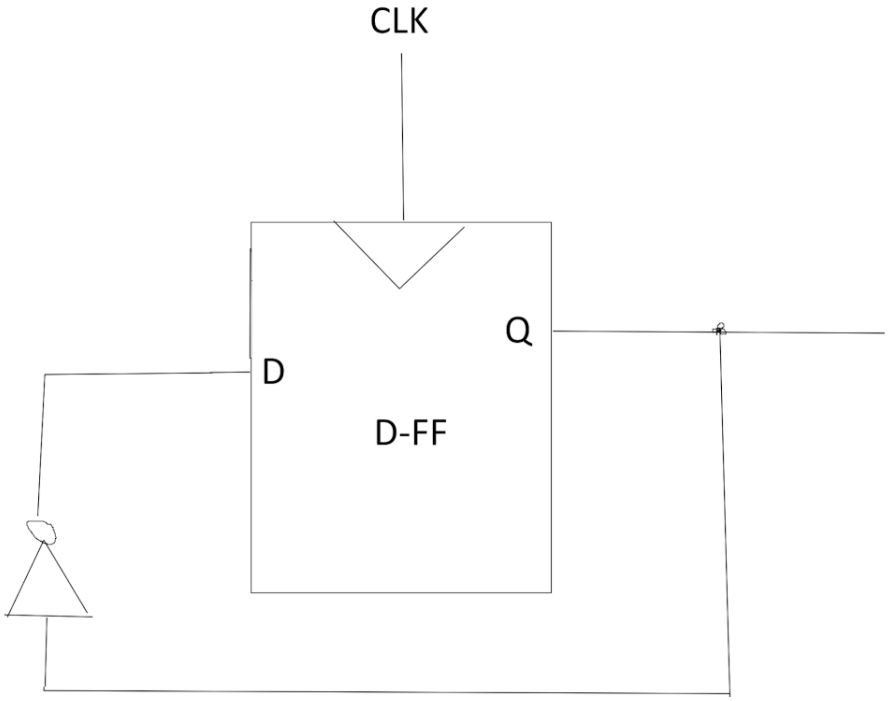


*

AND con solo NAND:

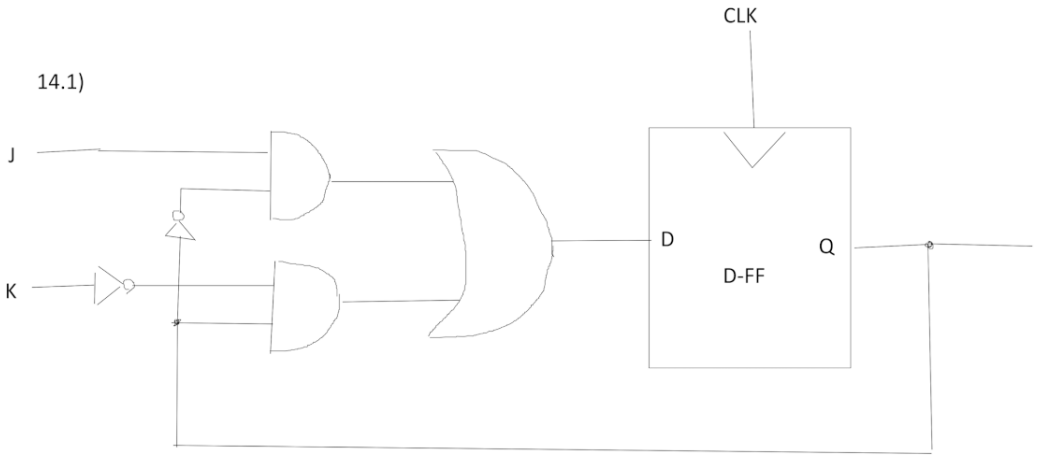


13)

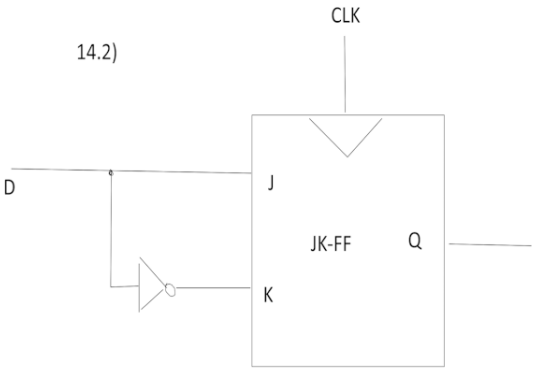


14) Per non rendere troppo pesante questa parte, ho omesso diversi dettagli, vi consiglio di dare un'occhiata al canale Youtube di Neso Academy per i dettagli passo x passo e per altre spiegazioni sulle reti logiche in generale

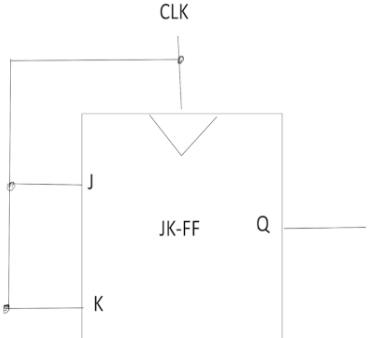
14.1)



14.2)



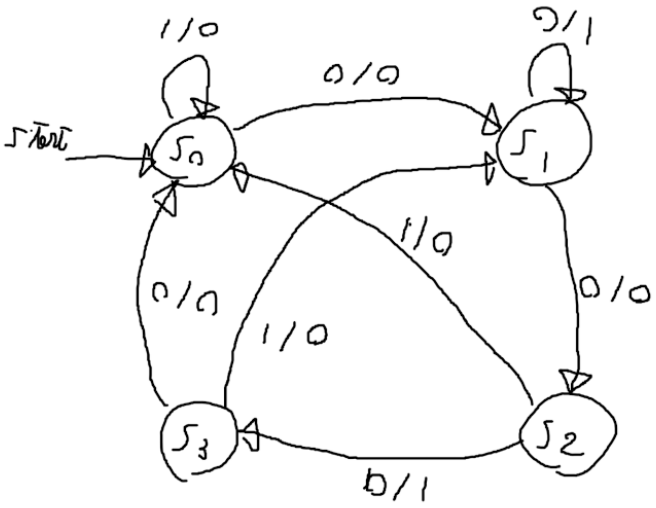
14.3)



15)

$$\begin{cases} D_1 = \overline{X}\,(Q_1 \oplus Q_2) \\ D_2 = \overline{X \oplus Q_2} \\ Y = \overline{X}\,Q_1\,\overline{Q_2} \end{cases}$$

S0 = (Q1=0,Q2=0)
S1 = (Q1=0,Q2=1)
S2 = (Q1=1,Q2=0)
S3 = (Q1=1,Q2=1)

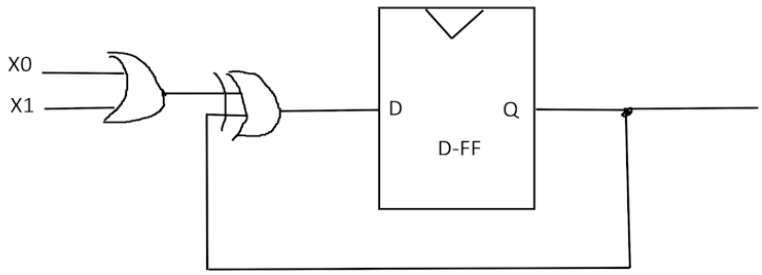


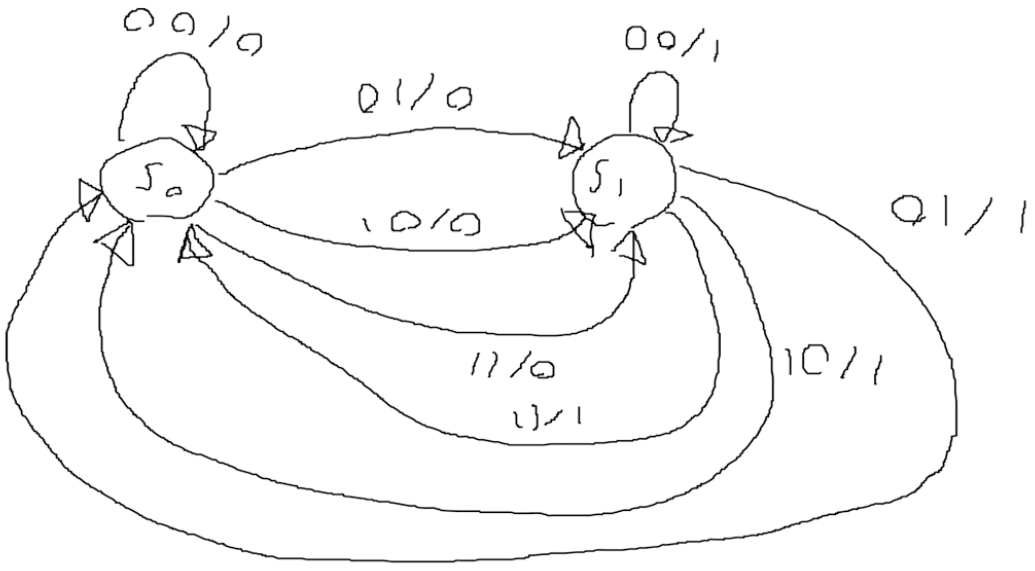
Q1	Q2	X	D1	D2	Y
0	0	0	0	1	0
0	0	1	0	0	0
1	0	0	1	1	1
1	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	0

16)

Y = Q
D = Q xor (X0 or X1)

Q	X0	X1	D	Y
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

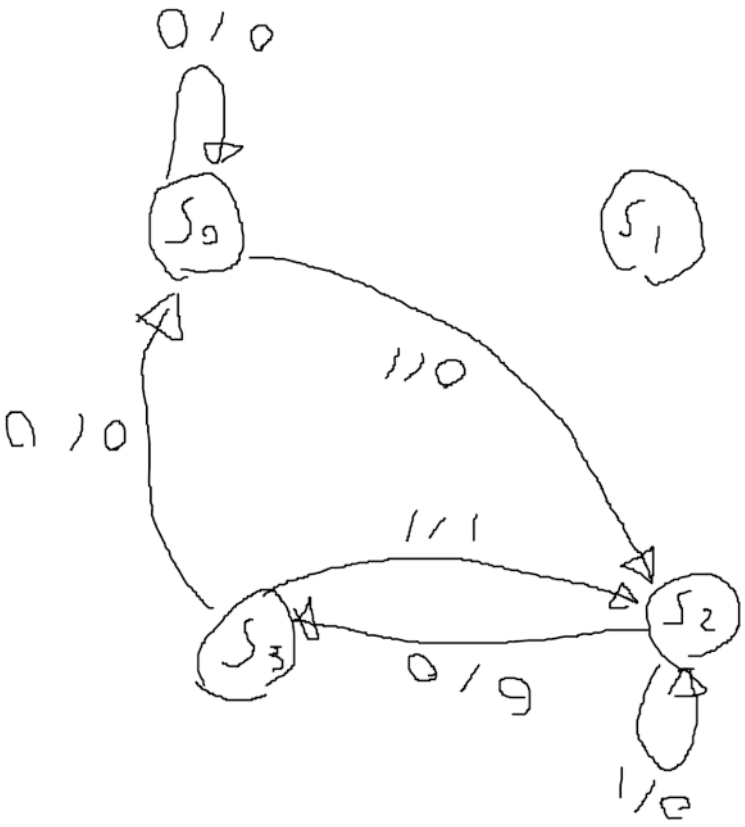




17)

$$\begin{cases} D_1 = Q_1\bar{Q}_2 + X \\ D_2 = (Q_1 \oplus Q_2)\bar{X} \\ Y = Q_1 Q_2 X \end{cases}$$

Q_1	Q_2	X	D_1	D_2	Y
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	X	X	X
0	1	1	X	X	X
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	1	0	1



IL DISEGNO DEL CIRCUITO

LO LASCIO A VOI

18)

18.1.1) y = (0,0,0,1,0,1)

18.1.2) y = (1,0,1,1,1,1)

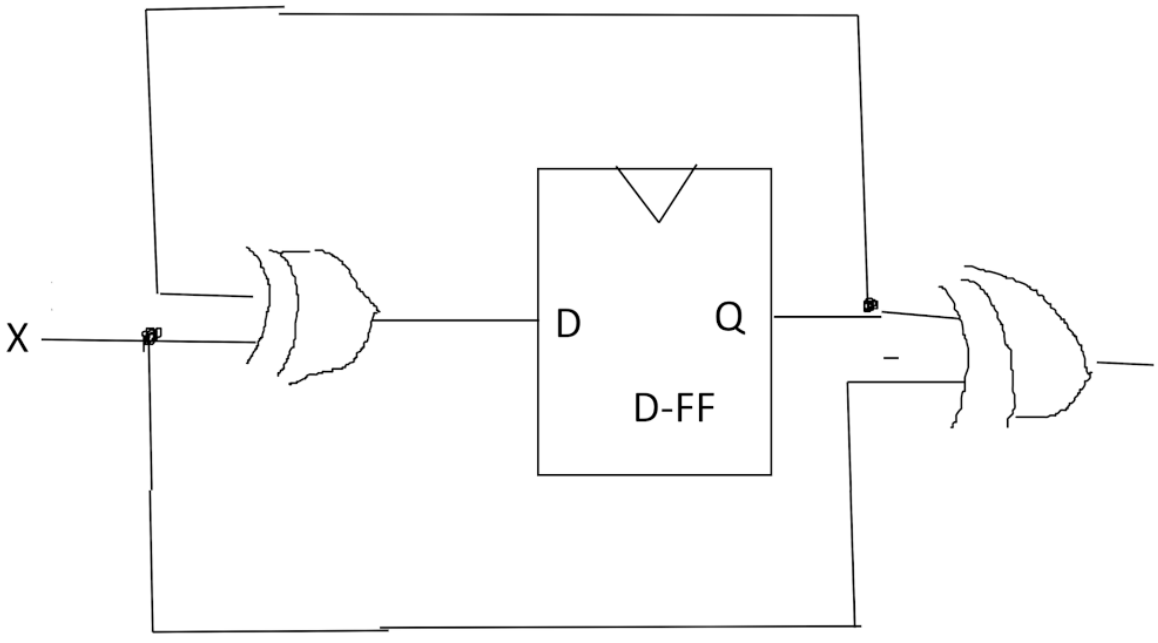
18.1.3) y = (0,1,1,0,0,1)

18.2) Fin quando è nello stato S0, l'OUTPUT è equivalente all'INPUT, se riceve 1 come INPUT, si sposta nello stato S1, nello stato S1, l'OUTPUT è il contrario dell'INPUT, se riceve 1 come INPUT, ritorna nello stato S0.

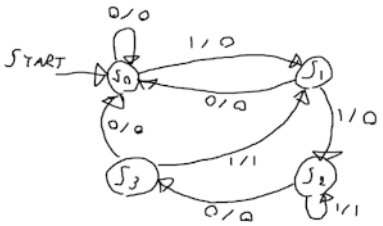
18.3)

Q	X	D	y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

$$\begin{cases} D = Q \oplus X \\ y = Q \oplus X \end{cases}$$



19)



S0 = (Q1=0, Q2=0)
S1 = (Q1=0, Q2=1)
S2 = (Q1=1, Q2=0)
S3 = (Q1=1, Q2=1)

Q_1	Q_2	X	D_1	D_2	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

LASCIO A VOI IL DISEGNO DEL CIRCUITO, MA AVETE TUTTI I PEZZI PER FARLO

D_1

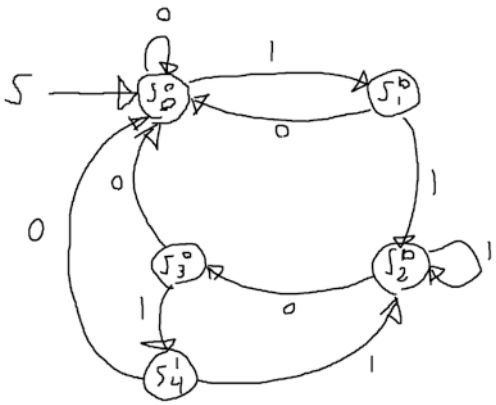
X \ Q_1Q_2	00	01	11	10
0	0	0	0	1
1	0	1	0	1

D_2

X \ Q_1Q_2	00	01	11	10
0	0	0	0	1
1	1	0	1	0

- $Y = Q_1 Q_2 X$
- $D_1 = \overline{Q_1} Q_2 X + Q_1 \overline{Q_2}$
- $D_2 = \overline{Q_1}\,\overline{Q_2} X + Q_1 Q_2 X + Q_1 \overline{Q_2}\,\overline{X}$

20) Passiamo da Automa alla "Mealy" ad un automa alla "Moore"

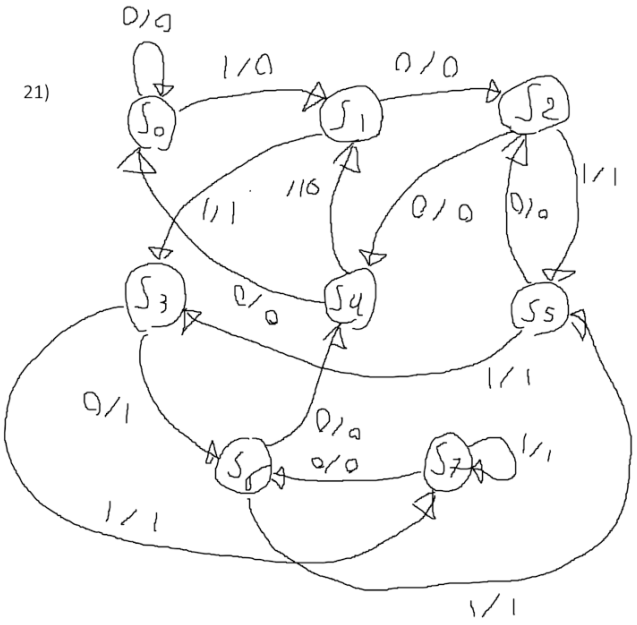


S0 = (Q1=0,Q2=0,Q3=0)
S1 = (Q1=0,Q2=0,Q3=1)
S2 = (Q1=0,Q2=1,Q3=0)
S3 = (Q1=0,Q2=1,Q3=1)
S4 = (Q1=1,Q2=0,Q3=0)

Q_1	Q_2	Q_3	X	D_1	D_2	D_3	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0
1	0	0	0	0	0	0	1
1	0	0	1	0	1	0	1
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Da qui potete ricavarvi facilmente
Le equazioni del circuito ed il corrispondente disegno, facendo attenzione che l'OUTPUT non deve dipendere dall'INPUT, ma dallo STATO del circuito.

21)

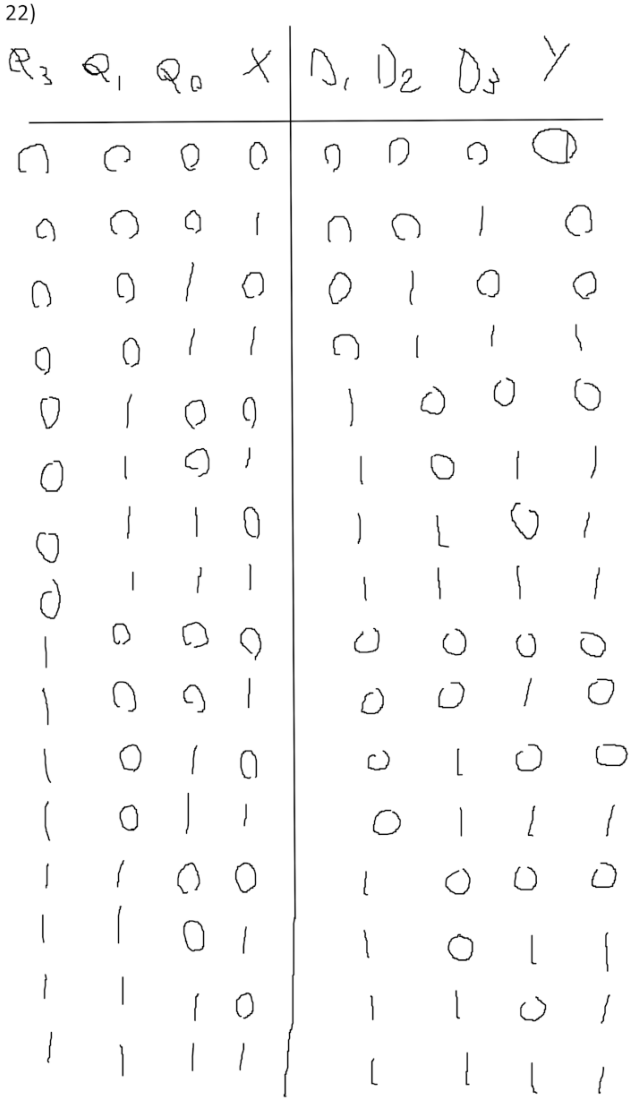


OK, SEGUITEMI:
Leggete gli stati da Dx verso Sx, ora, immaginate che se ci troviamo nello stato S2 = (0,1,0), vuol dire che gli ultimi 3 bit letti sono stati 0,1,0; se leggessi 1 come prossimo bit, allora gli ultimi 3 bit letti sarebbero 1,0,1, che corrisponde allo stato S5, se invece di 1 avessi letto 0, allora gli ultimi 3 bit letti sarebbero stati 1,0,0, che corrisponde allo stato S4, e così via per tutti gli stati.
Credo ci sia più di una soluzione possibile per questo esercizio, ma questa è quella che mi sembrava più intuitiva.

S0 = (0,0,0)
S1 = (0,0,1)
S2 = (0,1,0)
S3 = (0,1,1)
S4 = (1,0,0)
S5 = (1,0,1)
S6 = (1,1,0)
S7 = (1,1,1)

Forse potrebbe essere più intuitivo farlo con gli automi alla Moore piuttosto che con quelli alla Mealy, provateci!

22)



DA QUI RICAVARE LE ESPRESSIONI
DEL CIRCUITO ED IL RELATIVO DISEGNO
NON DOVREBBE ESSERE DIFFICILE...

SI TRATTA SOLO DI UN LAVORO MECCANICO

23) Seguiamo l'esempio nelle Note:

X0 = 1, X1 = 1, X2 = 0, X3 = 0
Numero in input = 0011
Complemento a 2 = 1101

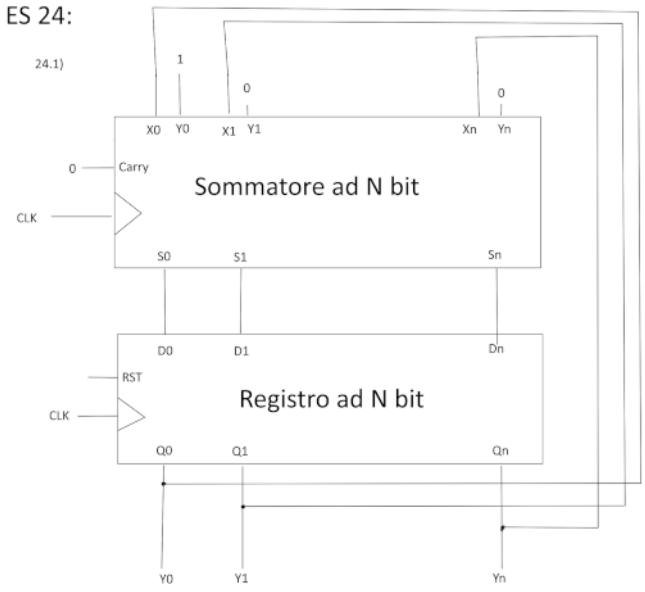
Se riceviamo come 5o bit 0...
Numero in input = 00011
Complemento a 2 = 11101

Se riceviamo come 5o bit 1...
Numero in input = 10011
Complemento a 2 = 01101

Sorge un grande problema, l'esercizio non specifica la lunghezza della sequenza di bit da convertire, quindi, fino a quando dobbiamo convertire numeri composti da "n" bit, dove "n" è un numero di bit ragionevole (per esempio, 4 o 5), l'esercizio si ridurrebbe ad una "semplice" tabella di Stato molto grande ma fattibile, tuttavia, più bit abbiamo in input e più stati ci serviranno per memorizzare i bit precedenti, e per "n" molto grandi (per esempio, 32112) capiamo subito che ciò è infattibile, perché il numero di Stati in cui il circuito si può trovare (e di conseguenza il numero di FlipFlop necessari per memorizzare TUTTI i bit precedenti) aumenterebbe in maniera dinamica man mano che riceviamo nuovi bit in input.

ES 24:

24.1)

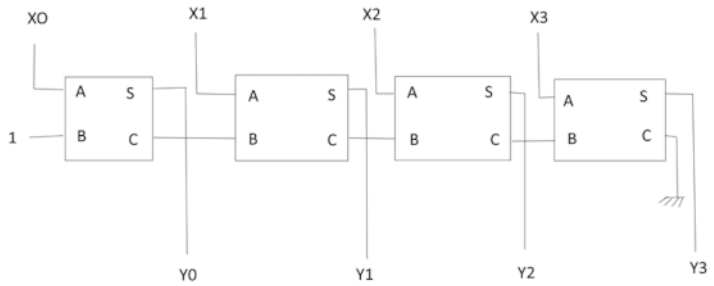


Il circuito è molto più semplice di quel che sembra:
Ad ogni ciclo di Clock il Registro passa il valore memorizzato al Sommatore, il quale lo incrementa di 1 e lo passa come nuovo input da memorizzare al Registro, quando RST viene posto ad 1, il Registro azzera i valori memorizzati per poi ripassarli al sommatore.

NOTA: e quando tutti gli N bit sono posti ad 1? Per esempio, stiamo lavorando in 4 bit ed ora il registro contiene 1111, che succede al prossimo ciclo di Clock?

24.2)

Per comodità, riprendiamo il circuito dell'esercizio 5, composto da 4 HALF ADDER che incrementava di 1 il numero in input, capiamo subito che è scalabile per qualsiasi N numero di bit in input (fissato).



Quindi, il "problema" di simulare il lavoro che faceva il sommatore nel punto 24.1 è risolto, non ci manca che creare un registro ad N bit con N Flip Flop, lascio a voi questa parte dell'esercizio ricordandovi che i Registri vengono trattati nella Lezione 14 a Pagina 6