Matteo Di Gioacchino

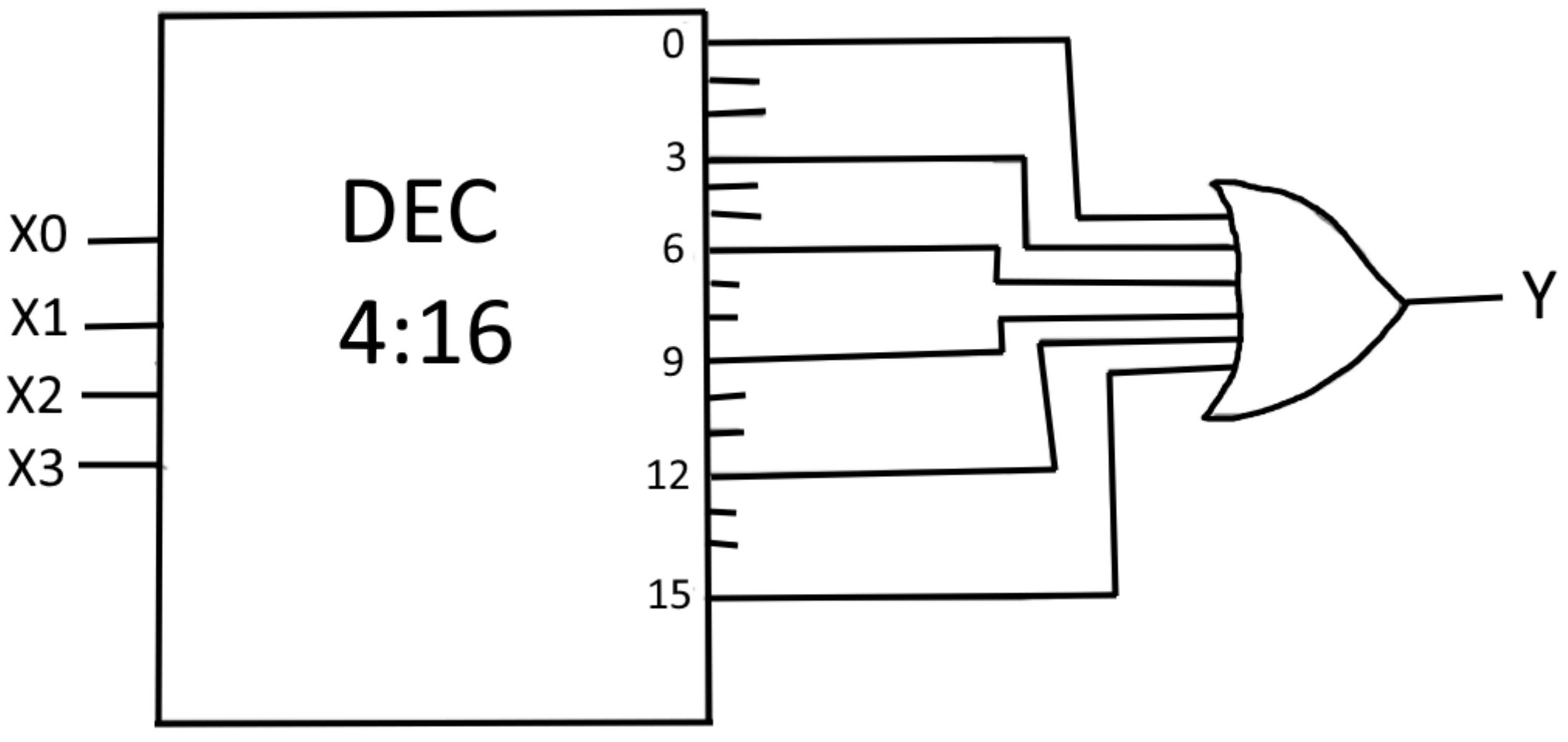
Soluzioni Esercitazione 16/12/2024

PER QUALSIASI DUBBIO, PROBLEMA O SE PENSATE DI AVER TROVATO UN ERRORE NELLE SOLUZIONI, SCRIVETEMI QUANDO VOLETE

NOTA: per migliorare la lettura, rispetto alle soluzioni dell'anno scorso, scriverò le tabelle di verità particolarmente complesse con dei Tool

1)

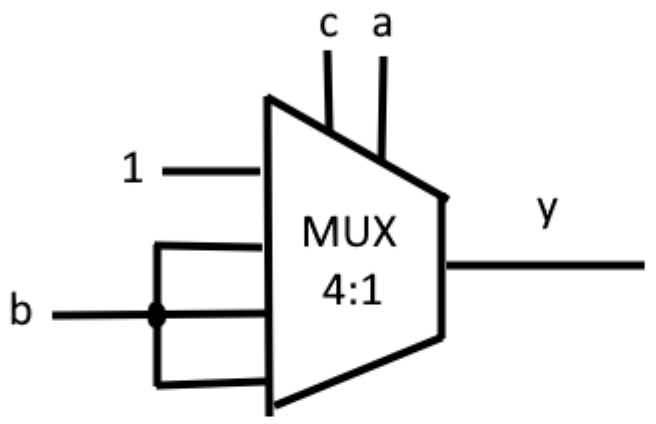
x_3	x_2	x_1	x_0	Decimal (N)	$N \mod 3$	Output (Divisible by 3)
0	0	0	0	0	0	1
0	0	0	1	1	1	0
0	0	1	0	2	2	0
0	0	1	1	3	0	1
0	1	0	0	4	1	0
0	1	0	1	5	2	0
0	1	1	0	6	0	1
0	1	1	1	7	1	0
1	0	0	0	8	2	0
1	0	0	1	9	0	1
1	0	1	0	10	1	0
1	0	1	1	11	2	0
1	1	0	0	12	0	1
1	1	0	1	13	1	0
1	1	1	0	14	2	0
1	1	1	1	15	0	1



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

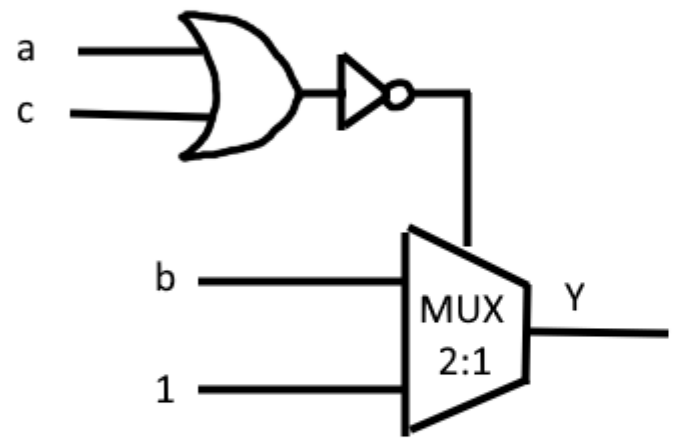
= b + !a!b!c = b + !a!c

2.1)



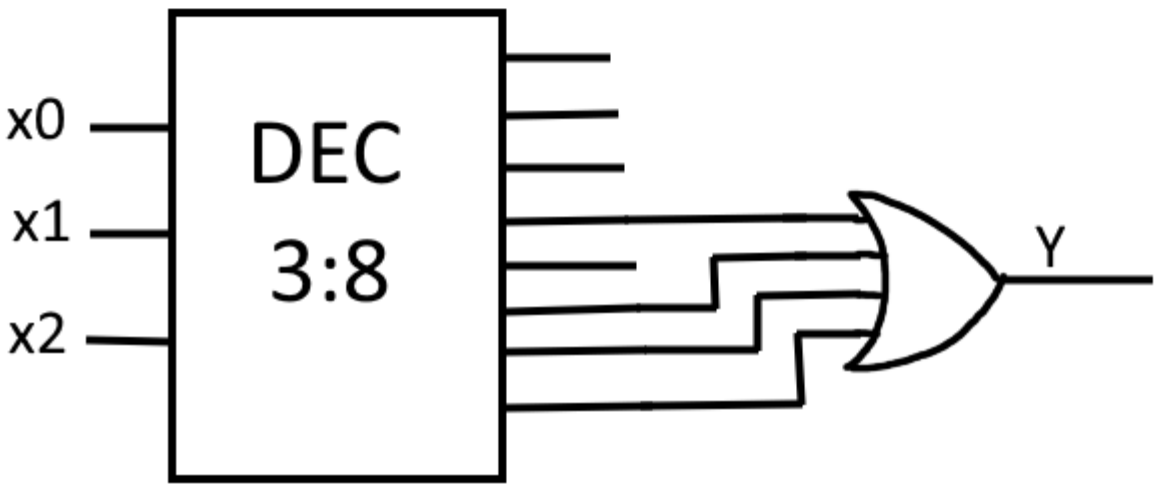
2.2) OSS: !a!c = !(a+c)

(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



NOTA: nelle soluzioni dell'anno scorso trovate lo svolgimento di questo esercizio utilizzando Karnaugh

4) Y = ab + (ab xor cd) + !ad = ab + (!(ab)cd + ab!(cd)) + !ad =

= ab(1 + !(cd)) + !(ab)cd + !ad =

= ab + !(ab)cd + !ad = ab + cd + !ad

a	b	c	d	ab	cd	$\neg a \wedge d$	$Y = ab + cd + \neg ad$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	0
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	0
0	1	1	1	0	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	1	0	0	1
1	1	0	1	1	0	0	1
1	1	1	0	1	0	0	1
1	1	1	1	1	1	0	1

CNF:

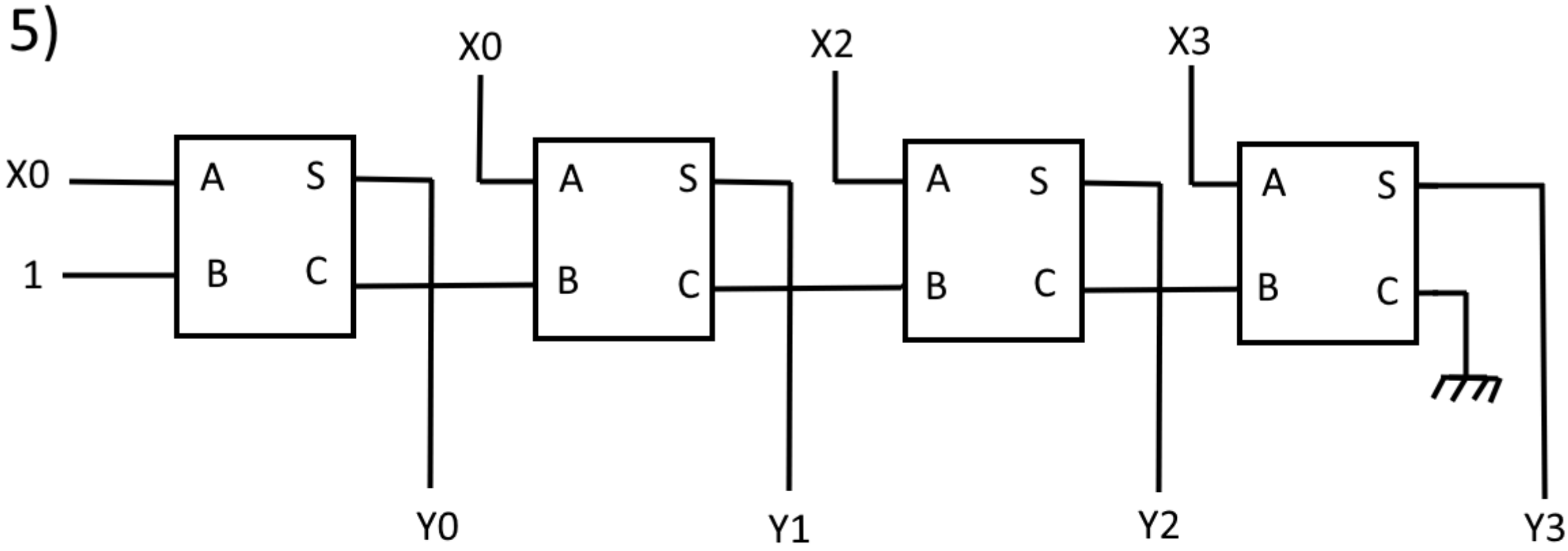
Y = (a + b + c + d) (a + b + !c + d) (a + !b + c + d)
(a + !b + !c + d) (!a + b + c +d) (!a + b + c + !d)
(!a + b + !c + d)

DNF:

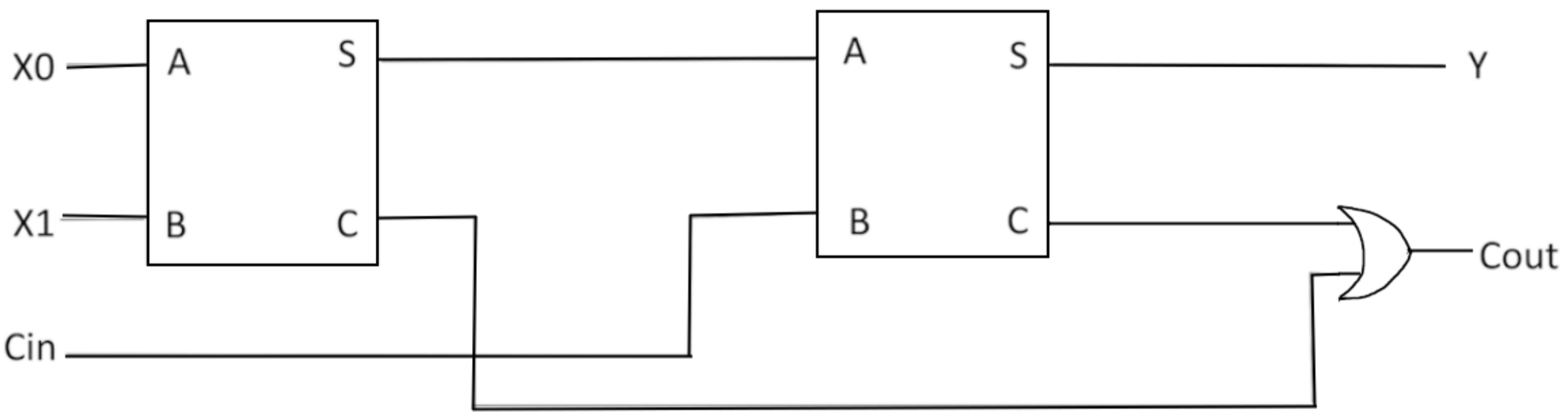
Potete o costruirla nella maniera classica o notare che la formula semplificata è già in DNF

(Nelle soluzioni dello scorso anno trovate la DNF completa)

5)



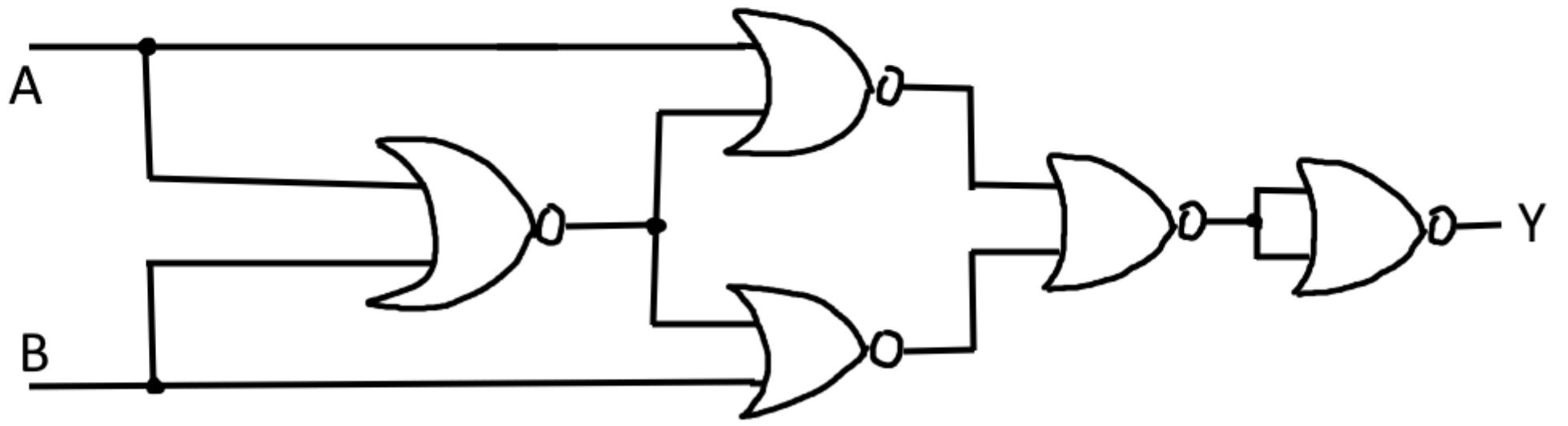
6) STEP 1: sappiamo come passare da HALF ADDER a FULL ADDER



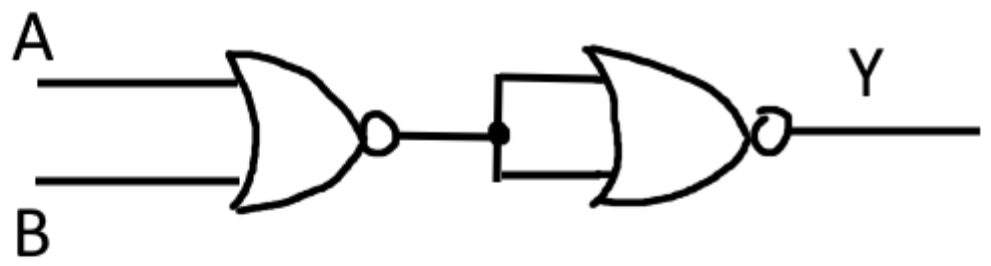
STEP 2: sappiamo che per costruire un HALF ADDER ci serve 1 porta XOR e 1 porta AND, quindi, in totale, abbiamo 2 XOR, 2 AND e 1 OR; ora, se riuscissimo a costruire ognuna di queste porte usando solo la porta NOR avremmo finito!

STEP 3: conversione delle porte logiche

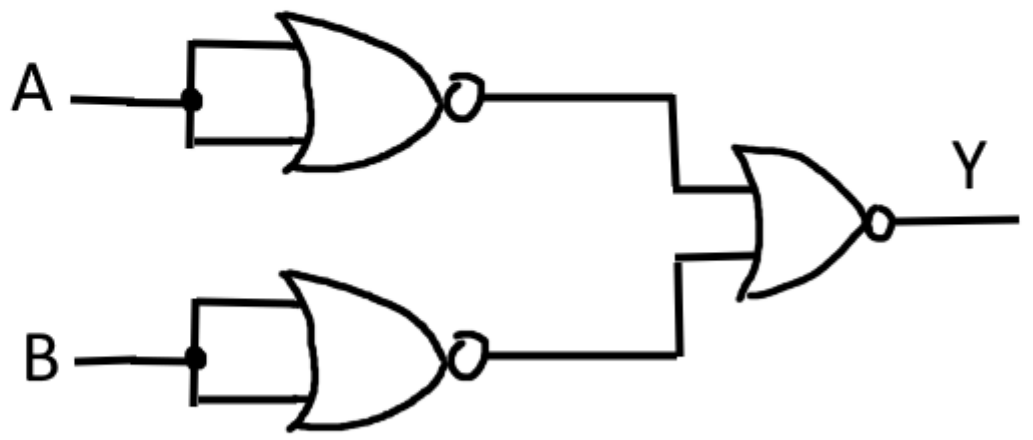
XOR:



OR:



AND:

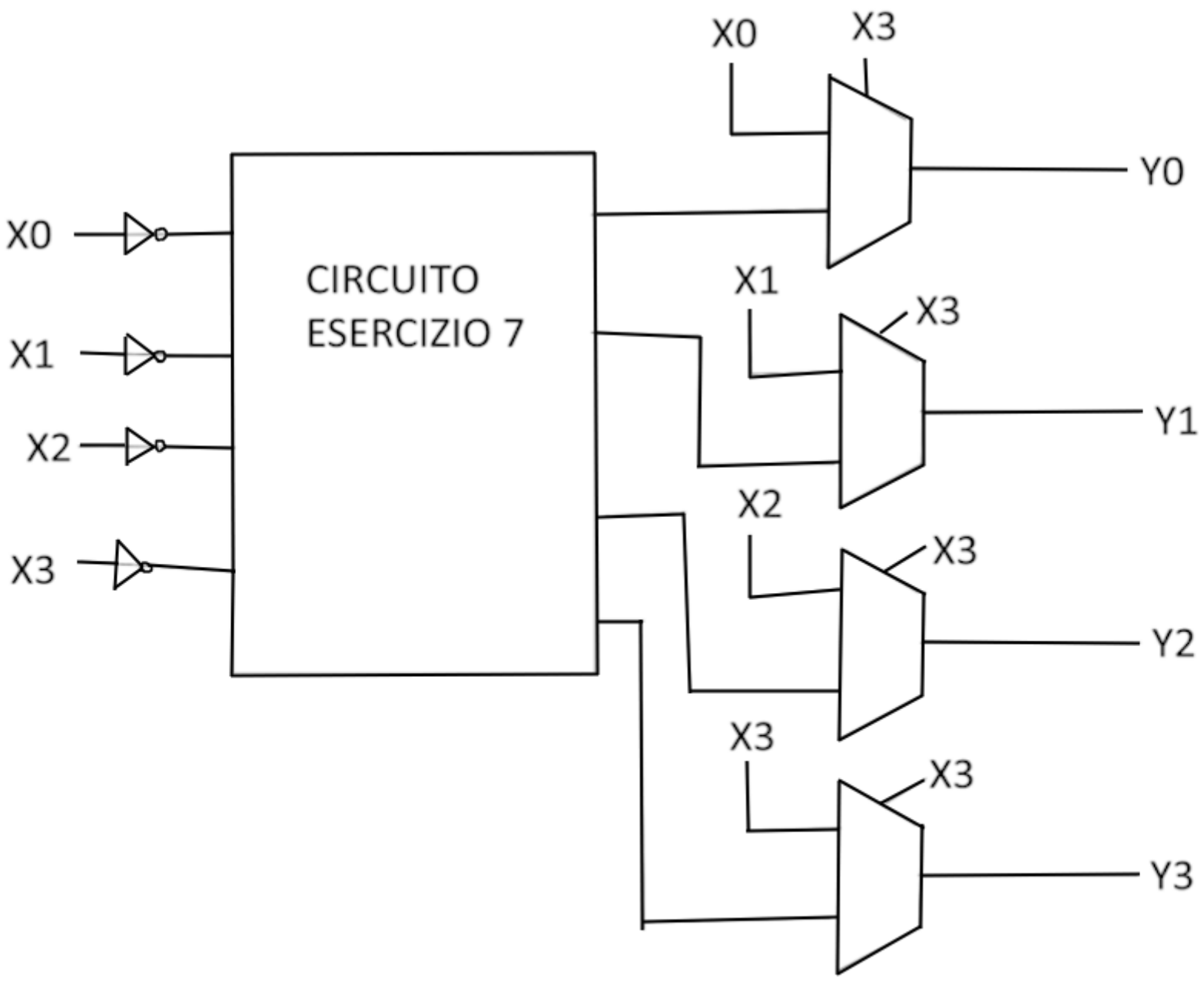


7)

STEP 1) inverto tutti i BIT (ci basta porre il NOT davanti ad ogni BIT di INPUT)
STEP 2) sommo 1 al numero ottenuto allo STEP 1 (si riveda l'ES. 5)

8) 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'Es. 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:

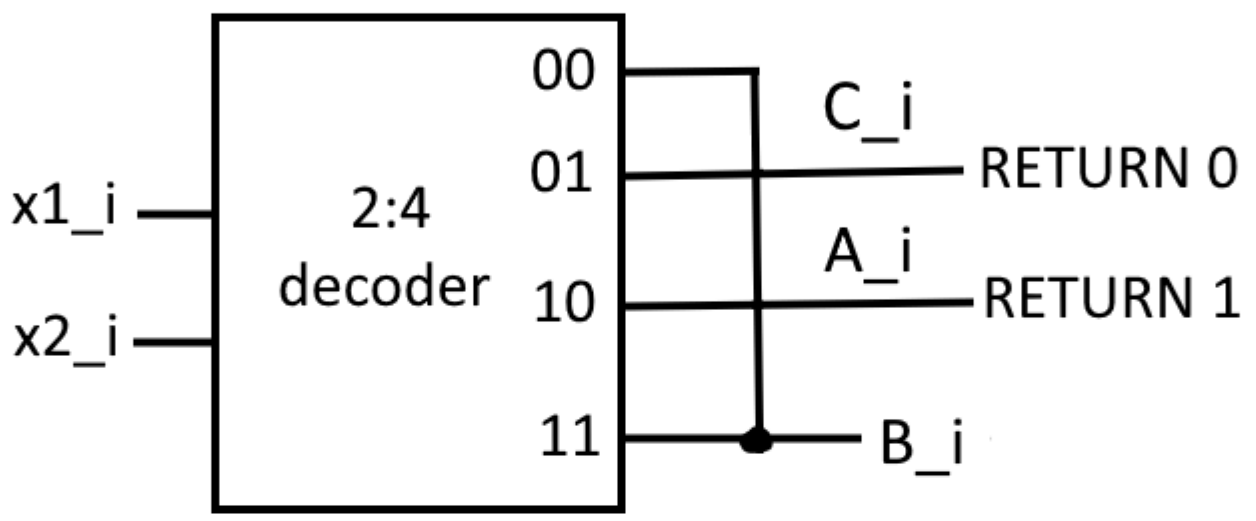


9)

Pensiamoci, il circuito restituisce 1 quando x1_i = x2_i per ogni i = 1,2,3,4; oppure quando incontriamo un x1_j > x2_j per un j = 1,2,3,4; altrimenti, il circuito restituisce 0 quando x1_k < x2_k per un k = 1,2,3,4. Un approccio potrebbe essere quindi il seguente:

STEP 1: controllo il MSB di x1_3 e x2_3, se x1_3 > x2_3, RETURN 1; se x1_3 < x2_3, RETURN 0, altrimenti, x1_3 = x2_3, passa allo STEP 2.
STEP 2: ripeti il procedimento di sopra per x1_i e x2_i, per i = 2,1
STEP 3: se x1_0 >= x2_0, RETURN 1, else RETURN 0

Dobbiamo quindi ora costruire un circuito che riesca a realizzare la funzione di comparazione tra due bit.



Pensiamo ad una visione d'insieme dell'output finale:

Se A3 è uguale ad 1 (x1_3 > x2_3), allora restituiamo 1, altrimenti, B3 DEVE essere uguale ad 1 (x1_3 = x2_3). Se B3 è uguale ad 1, allora compariamolo con lo stesso procedimento appena fatto con A2 e B2, poi con A1 e B1, ed infine con A0 e B0

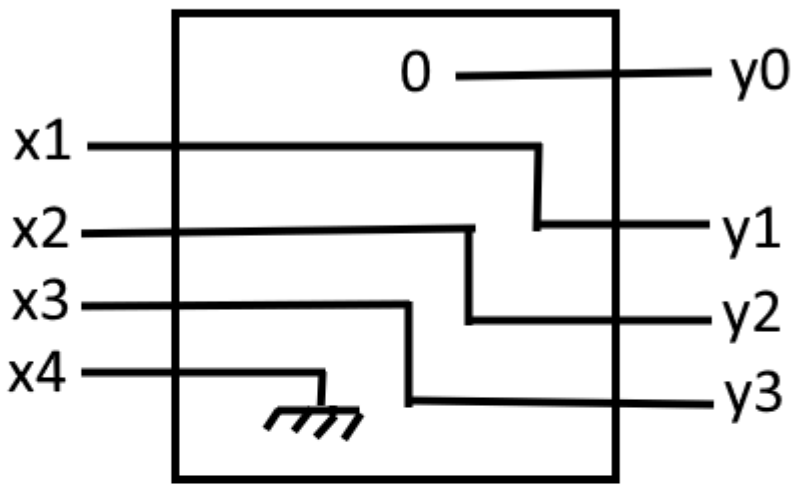
Y = A3 or (B3 and (A2 or (B2 and (A1 or (B1 and (A0 or B0))))))

x1_3 > x2_3
x1_3 = x2_3
x1_2 > x2_2
x1_2 = x2_2
x1_1 > x2_1
x1_1 = x2_1
x1_0 >= x2_0

In qualsiasi altro caso (C_i = 1), allora Y non sarà mai soddisfatta e l'output sarà 0.
Da qui è solo un esercizio di disegno nell'implementare il circuito appena spiegato.

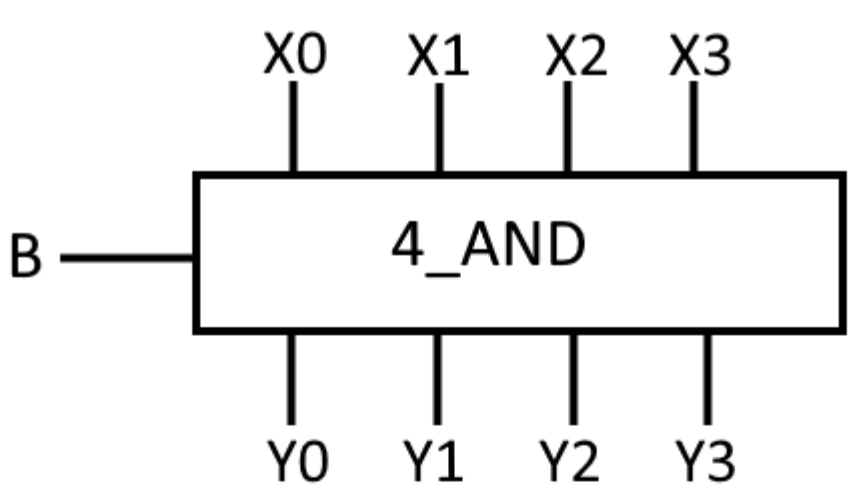
NOTA: nelle soluzioni dell'anno scorso c'era un altro svolgimento, ma rileggendolo non mi convinceva e l'ho rifatto

10) Shiftando il numero di 1 posizione a sinistra stiamo effettuando la moltiplicazione x2

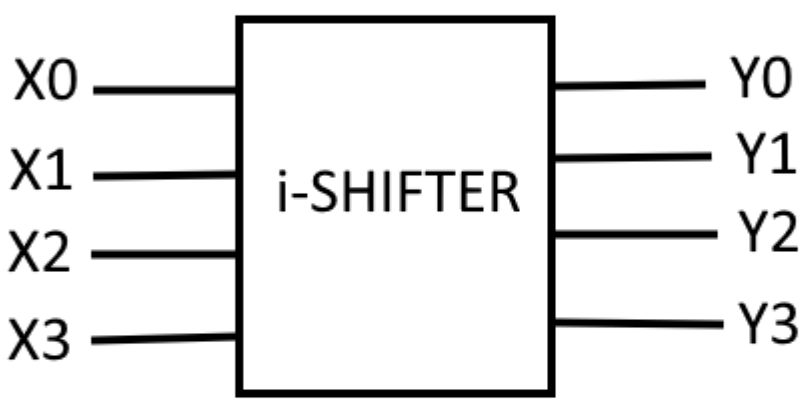


11) Se "s = 00", allora restituiamo il numero in input, se "s = 01", allora bisogna effettuare uno shift verso sx, se "s = 10", allora bisogna effettuare due shift verso sx, se "s = 11", allora bisogna effettuare 3 shift verso sx.
In generale (e questo vi servirà anche per il corso di Architettura dei Calcolatori) se "s = i", allora moltiplichiamo per 2^i, con i = 0,1,2,3.

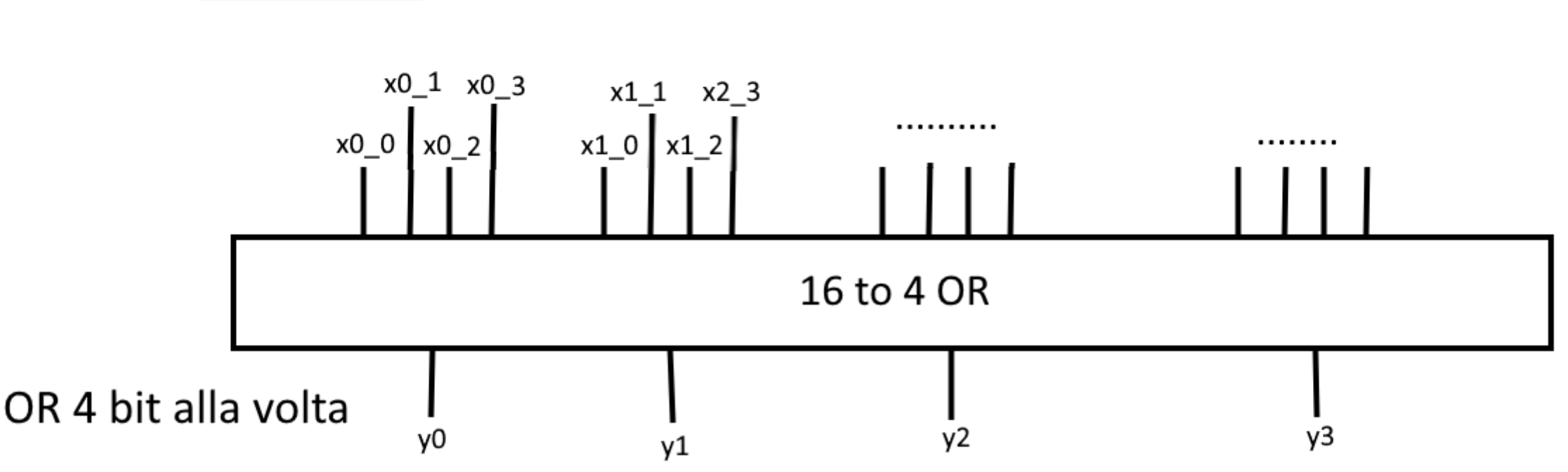
Proviamo a dare una struttura organizzata al circuito costruendo i seguenti blocchi logici:



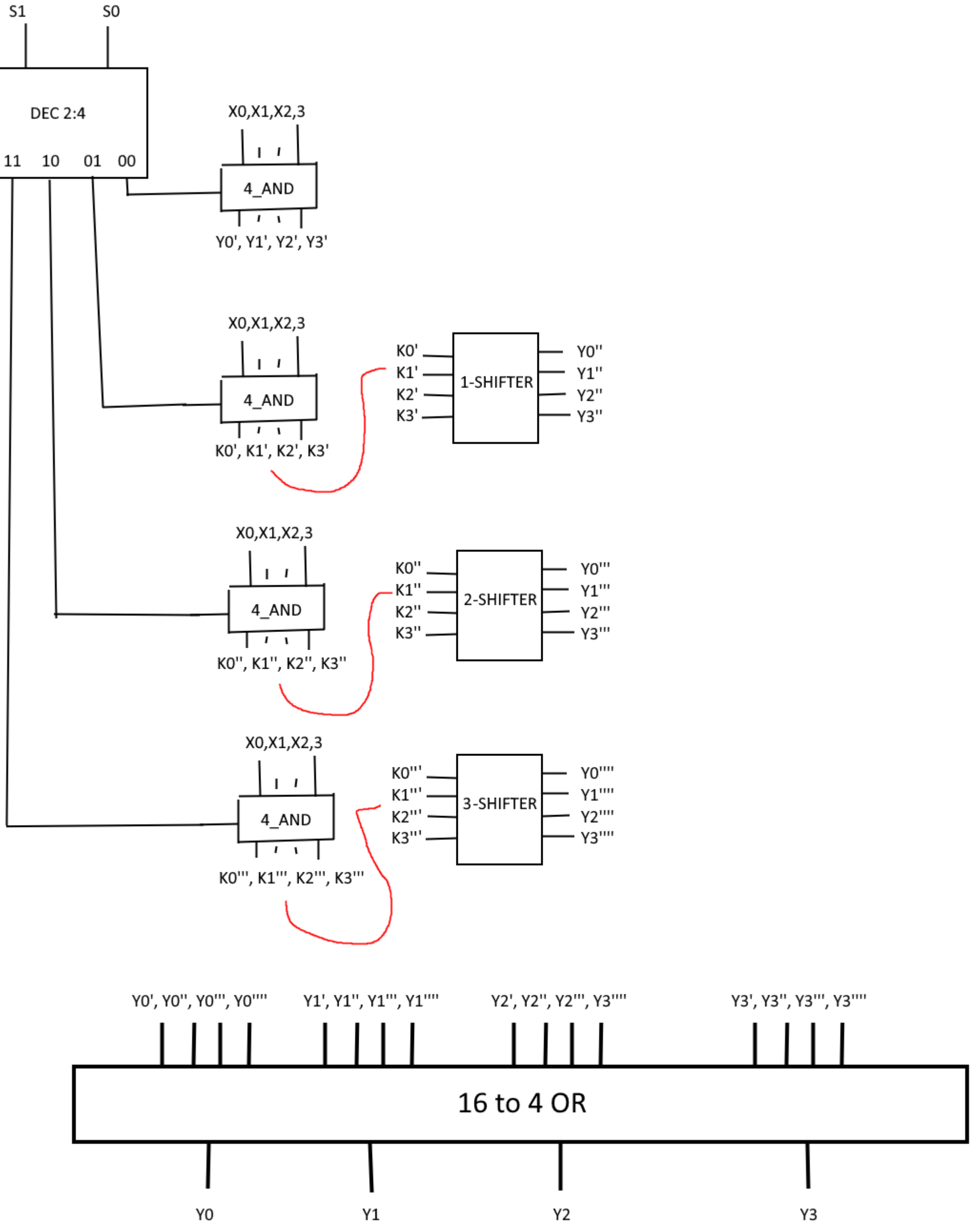
Prende ogni bit in input (X_3,X_2,X_1,X_0) e lo mette in AND con B, restituendo Y_i = (X_i and B)



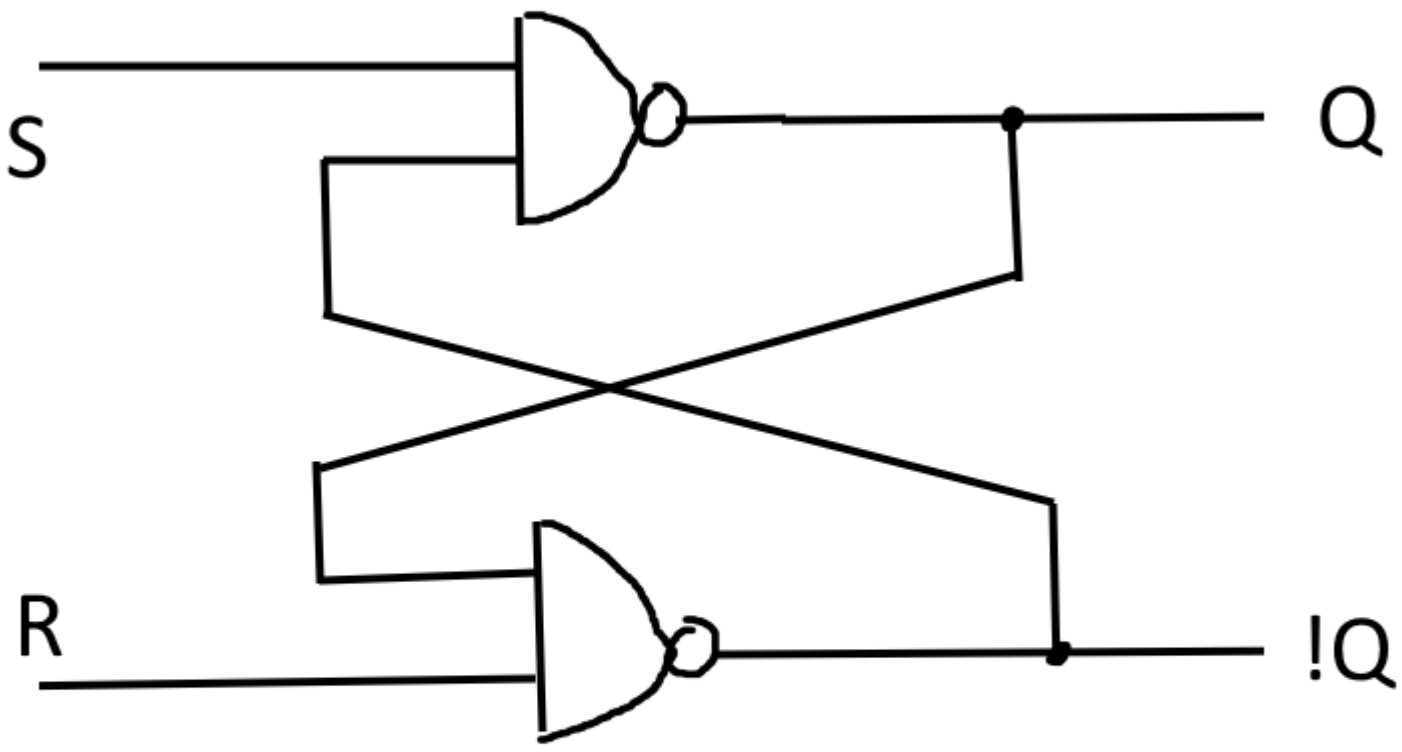
Effettua lo shift verso sx "i" volte



SCHEMA:


S1
S0
DEC 2:4
11
10
01
00
X0,X1,X2,3
4_AND
Y0', Y1', Y2', Y3'
X0,X1,X2,3
4_AND
K0', K1', K2', K3'
K0'
K1'
K2'
K3'
1-SHIFTER
Y0''
Y1''
Y2''
Y3''
X0,X1,X2,3
4_AND
K0'', K1'', K2'', K3''
K0''
K1''
K2''
K3''
2-SHIFTER
Y0'''
Y1'''
Y2'''
Y3'''
X0,X1,X2,3
4_AND
K0''', K1''', K2''', K3'''
K0'''
K1'''
K2'''
K3'''
3-SHIFTER
Y0''''
Y1''''
Y2''''
Y3''''
Y0', Y0'', Y0''', Y0''''
Y1', Y1'', Y1''', Y1''''
Y2', Y2'', Y2''', Y3''''
Y3', Y3'', Y3''', Y3''''
16 to 4 OR
Y0
Y1
Y2
Y3

12)

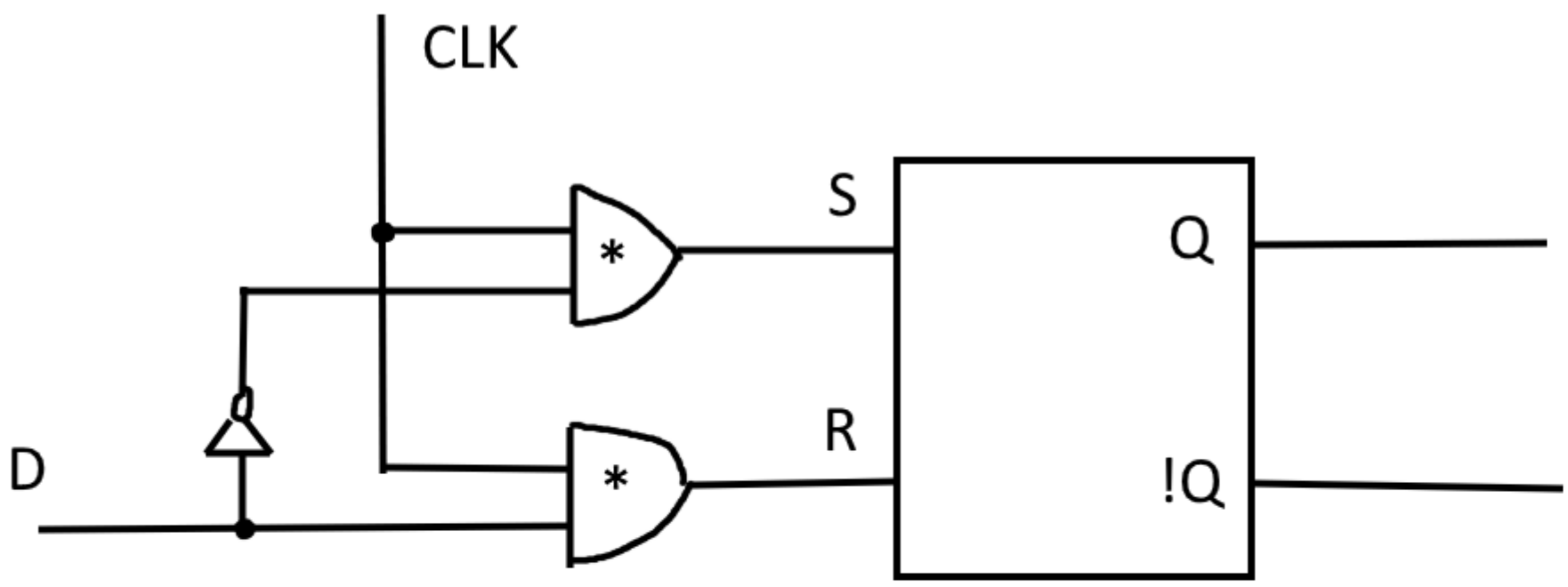


S	R	Q	!Q
0	0	????????????	
0	1	1	0
1	0	0	1
1	1	Qprec	!Qprec

Le differenze sono:

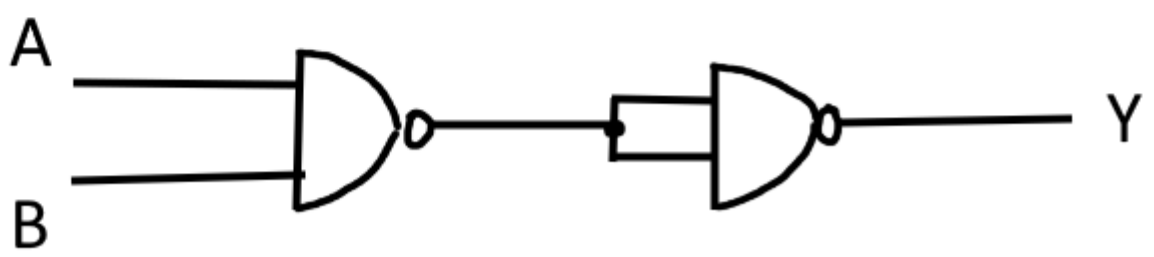
1) lo stato indesiderato ora è causato da (S=0,R=0) e non da (S=1,R=1)
2) lo stato di memoria ora è causato da (S=1,R=1) e non da (S=0,R=0)
3) Set e Reset sono "invertiti" rispetto all'SR-Latch con le porte NOR

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

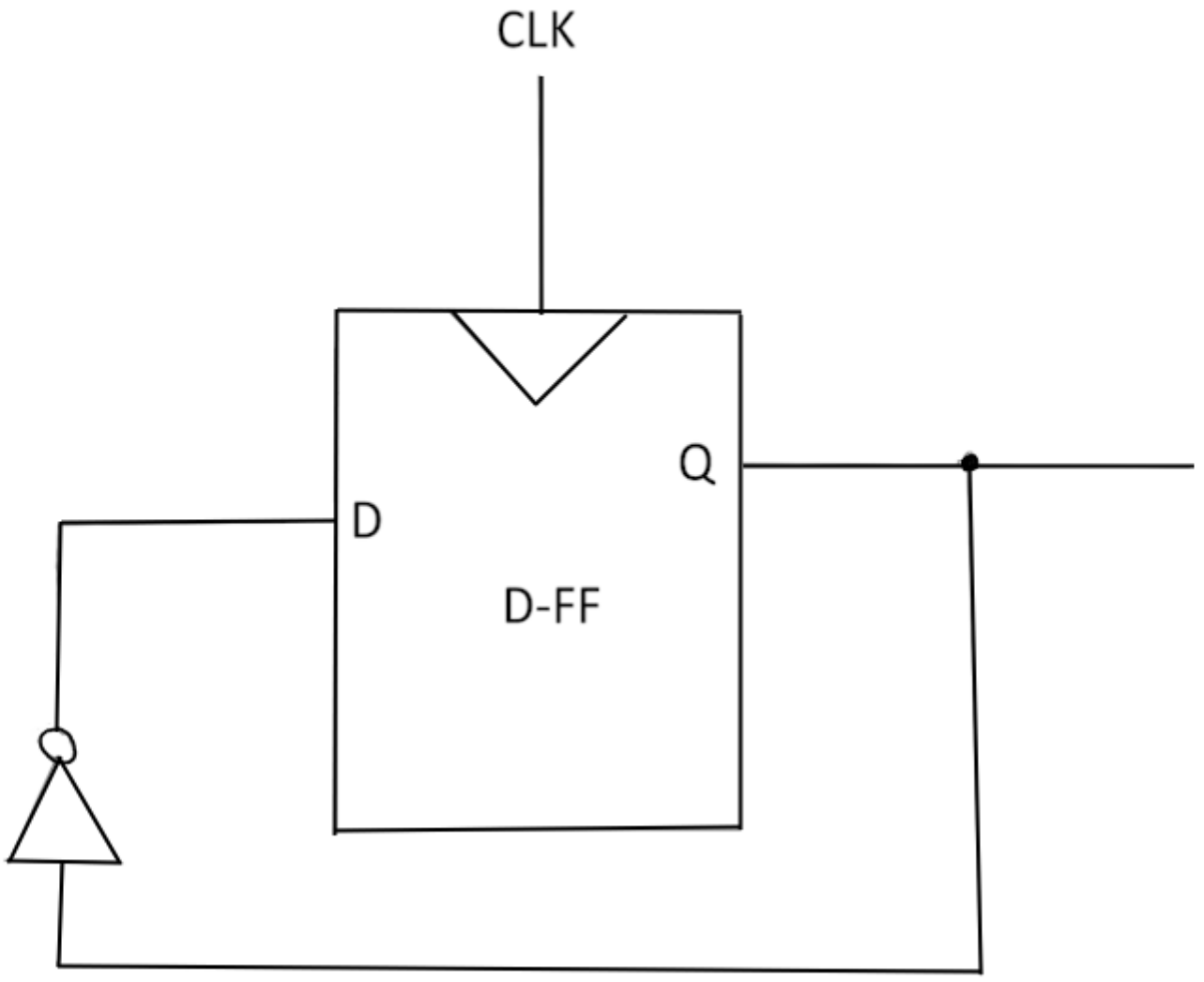


*

AND con solo NAND

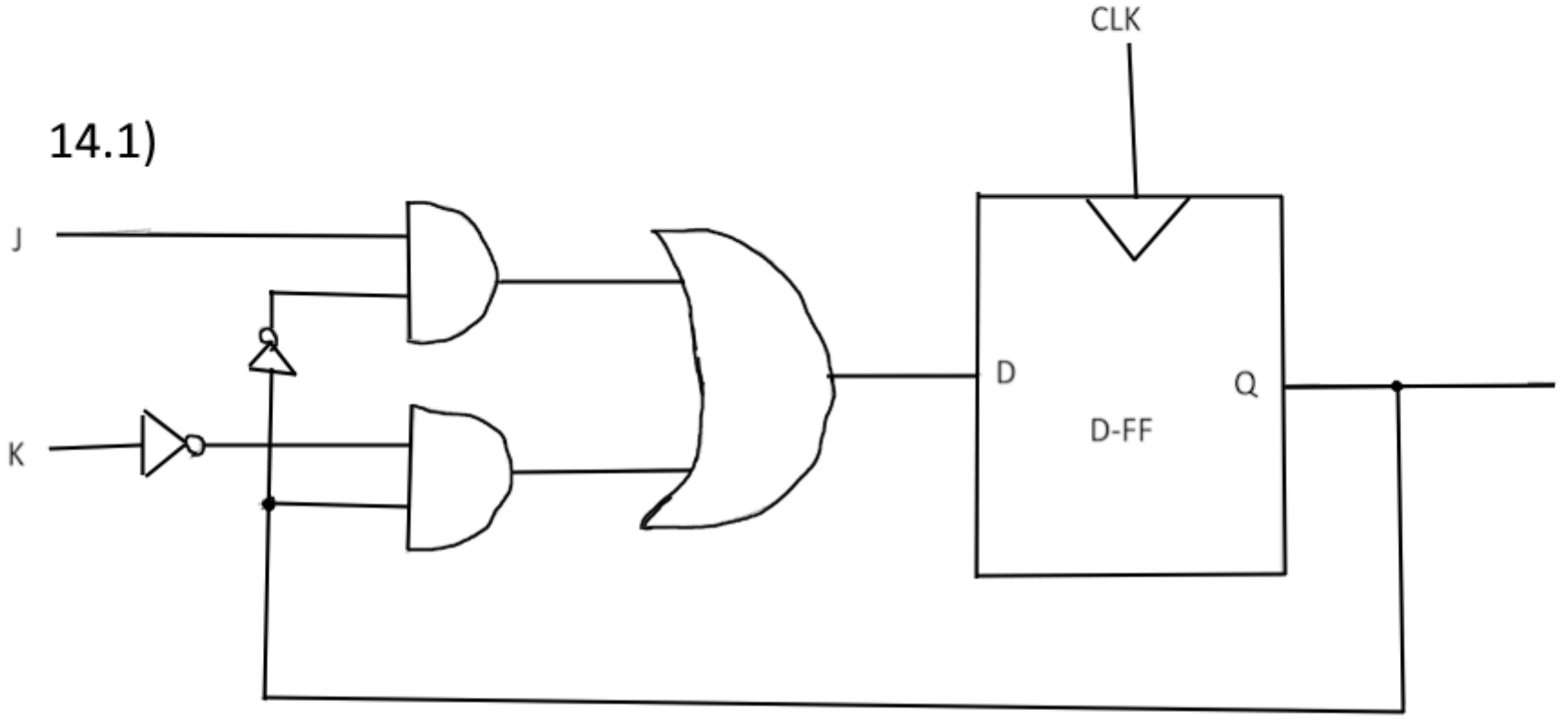


13)

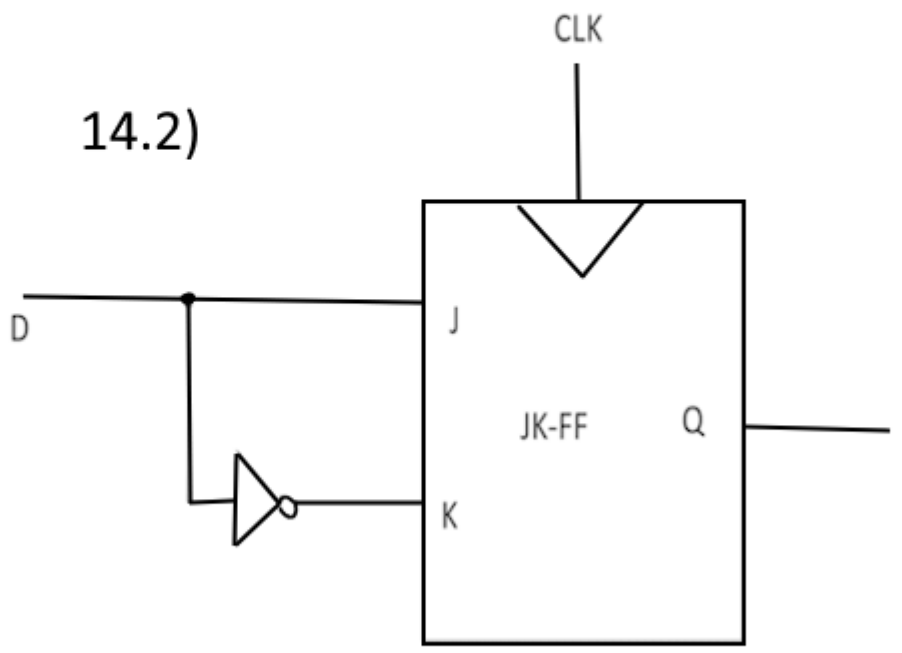


14) Per non rendere troppo pesante questa parte, ho omesso diversi dettagli,
vi consiglio di dare un'occhiata al canale Youtube di Neso Academy
per i dettagli passo x passo e per altre spiegazioni sulle reti logiche in generale

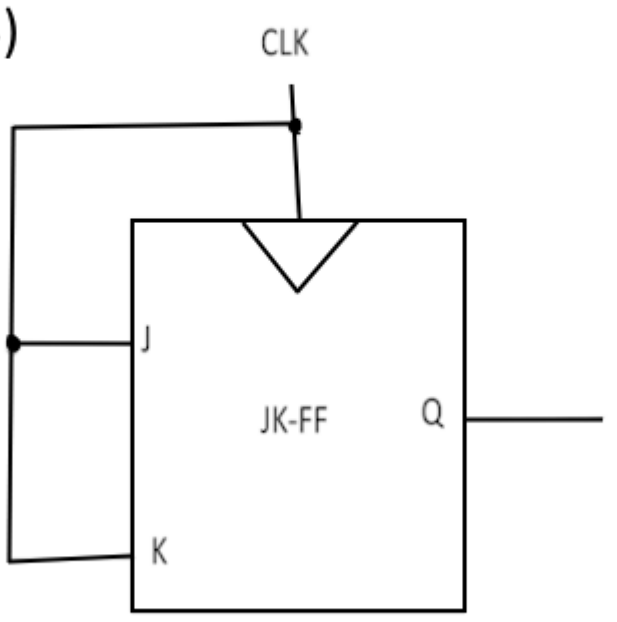
14.1)



14.2)



14.3)

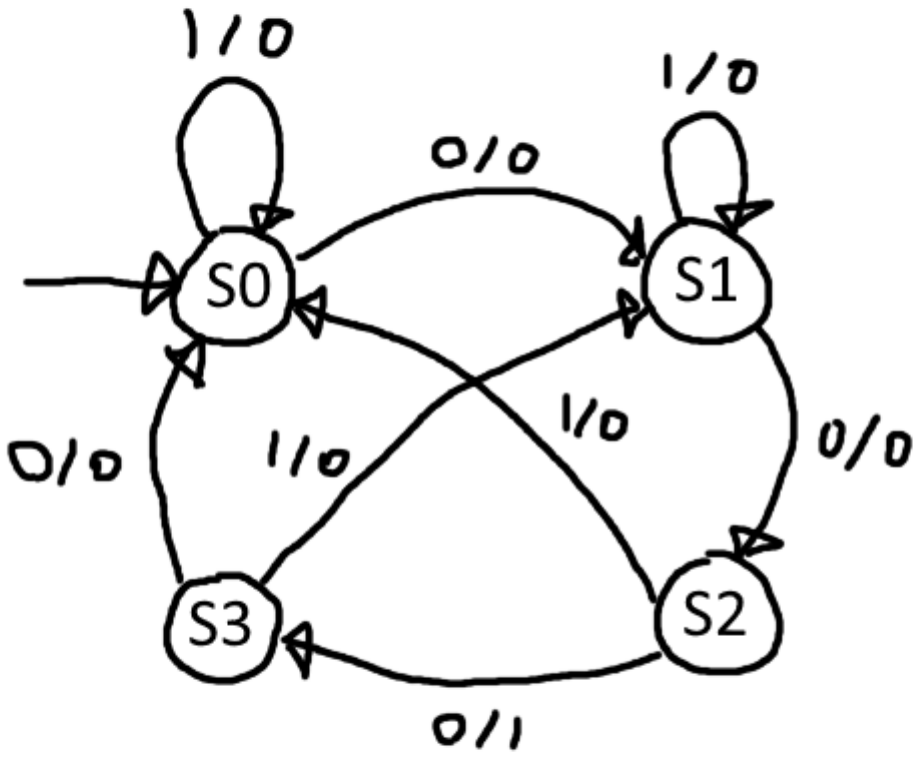


15)

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

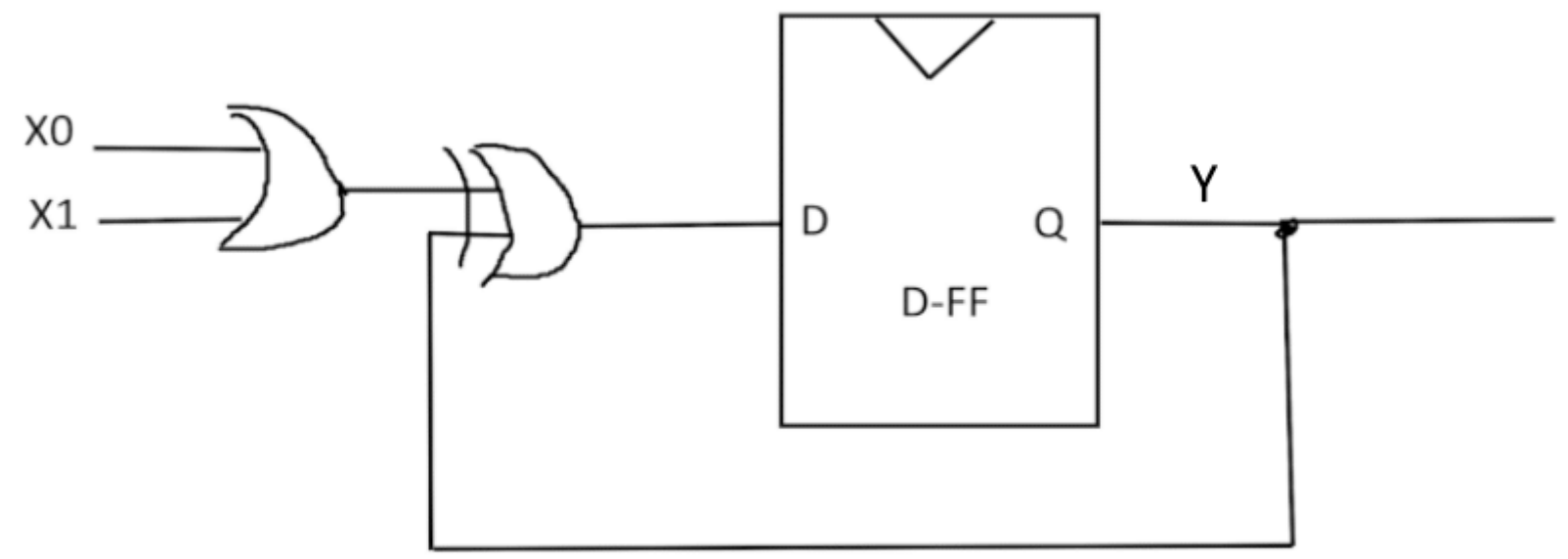
Q2	Q1	X	D2	D1	Y
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	1	0	0

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



16)

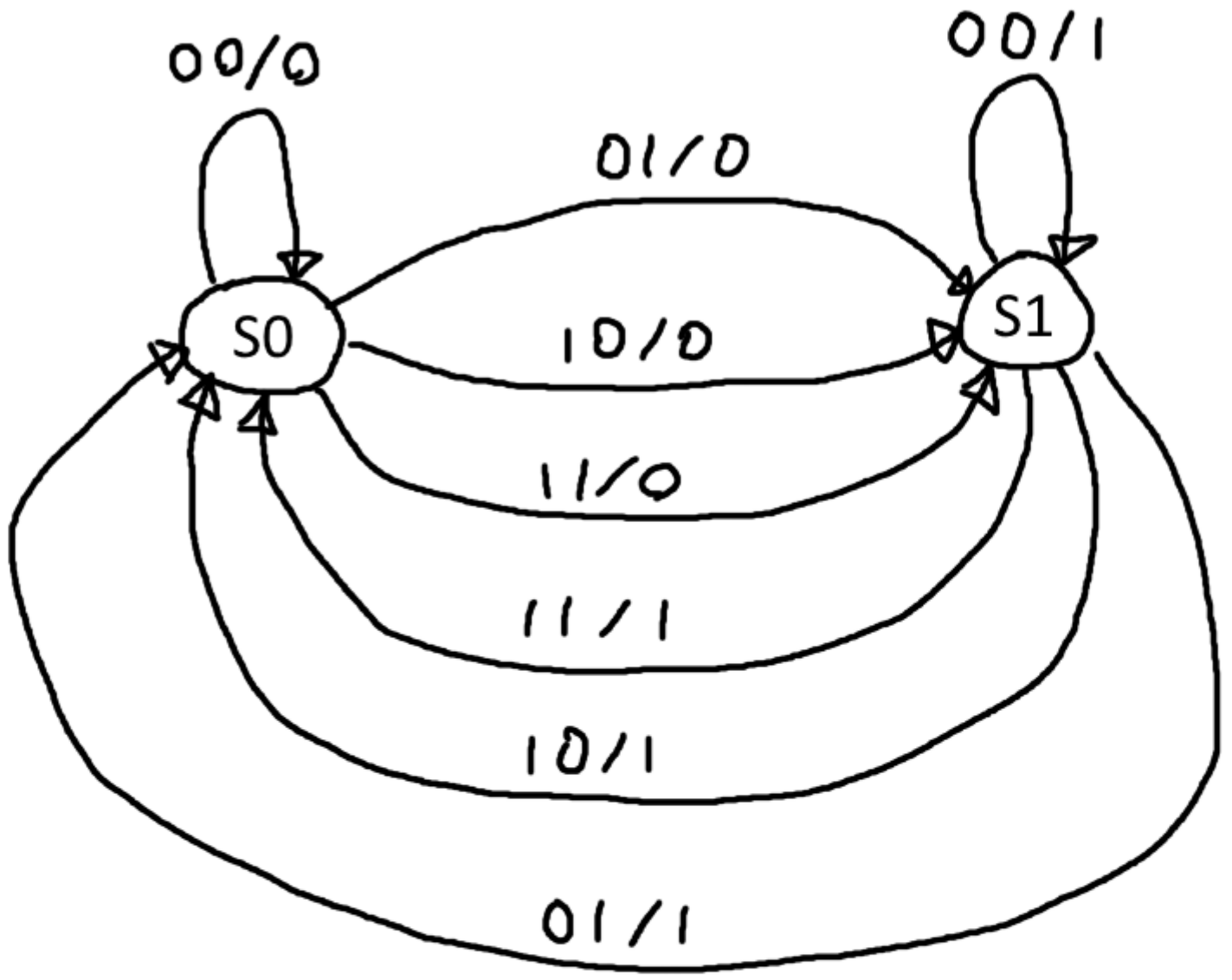
$$\begin{cases} Y = Q \\ D = Q \oplus (X_0 + X_1) \end{cases}$$



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

S0 = (Q = 0)

S1 = (Q = 1)

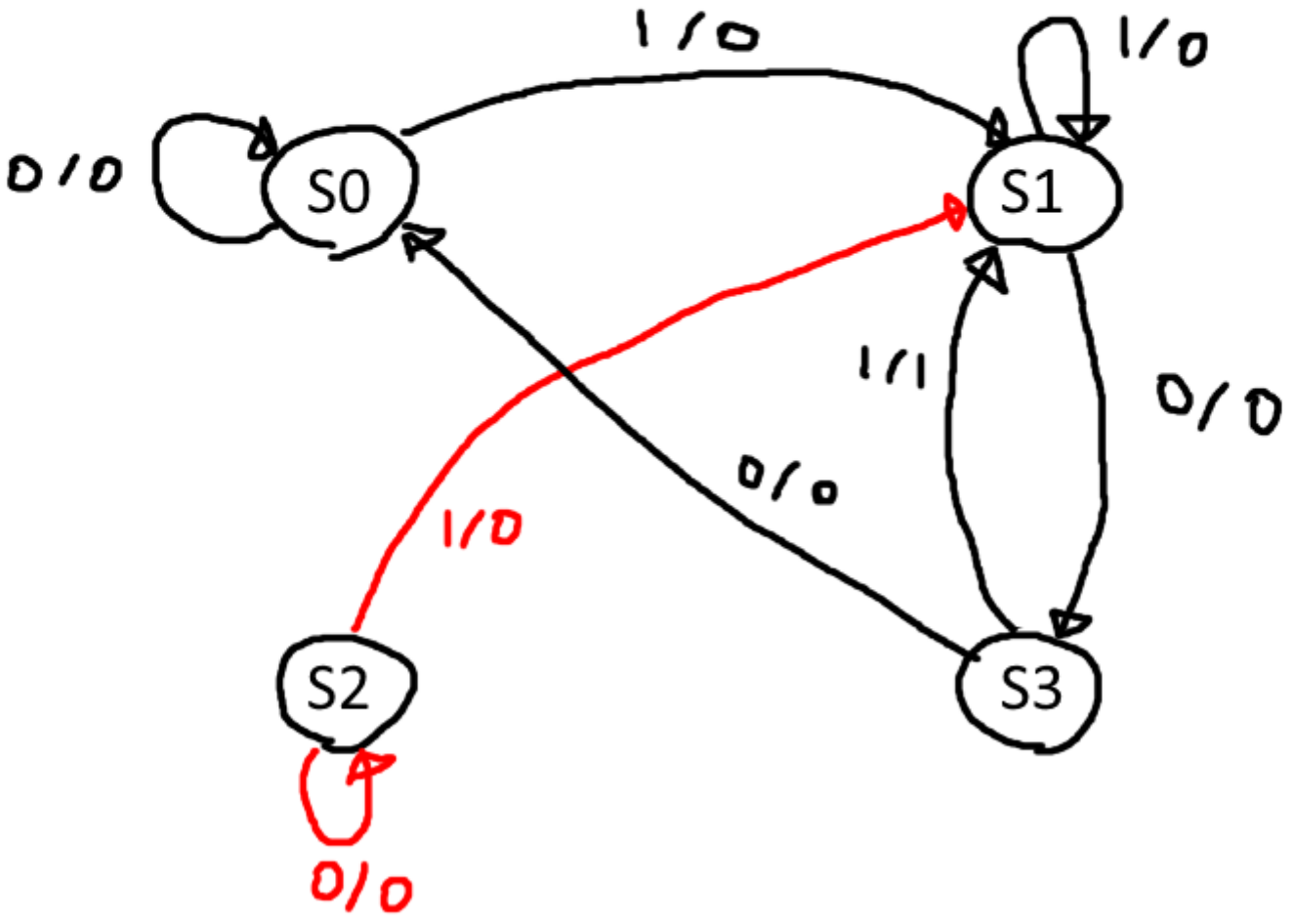


17)

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

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

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



IL DISEGNO DEL CIRCUITO LO LASCIO A VOI

18)

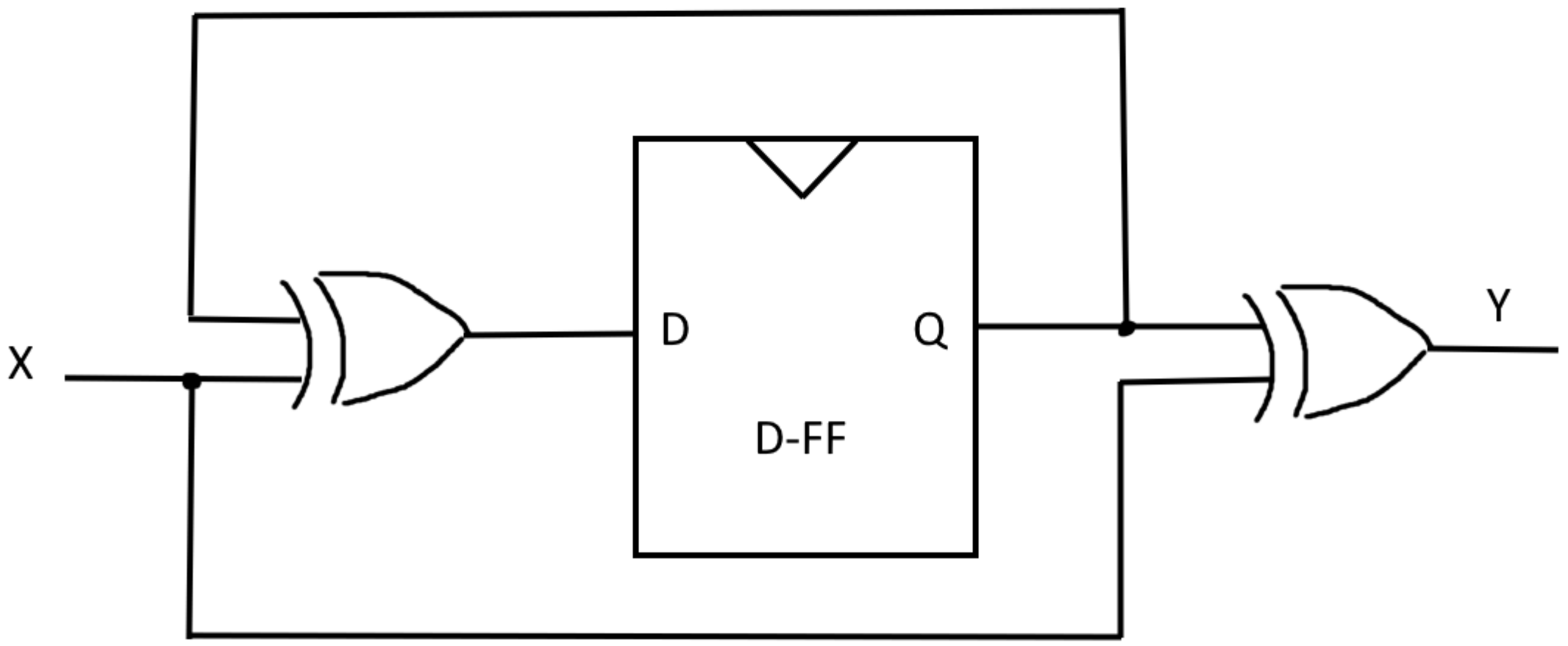
18.1.1) y = (0,0,0,1,0,1)
18.1.2) y = (1,0,1,1,1,1)
18.1.3) y = (0,1,1,0,0,1)

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)

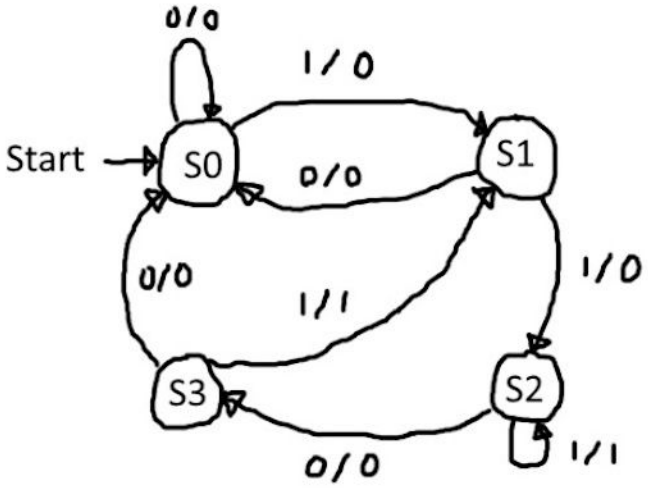
Q	X	D	Y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

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



19)

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



Q1	Q2	X	D1	D2	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

D1

X \ Q1 Q2	00	01	11	10
0	0	0	0	1
1	0	1	0	1

D2

X \ Q1 Q2	00	01	11	10
0	0	0	0	1
1	1	0	1	0

D1 = !Q1 Q2 X + Q1 !Q2
D2 = !Q1 !Q2 X + Q1 Q2 X + Q1 !Q2 X
Y = Q1 Q2 X

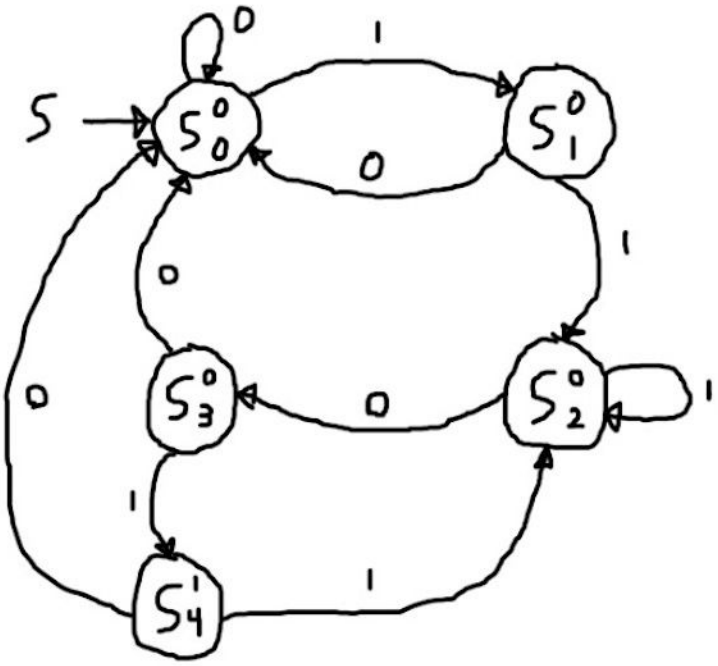
Lascio a voi il disegno del circuito, avete tutti i pezzi per poterlo fare

20)

Q1	Q2	Q3	X	D1	D2	D3	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	1	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

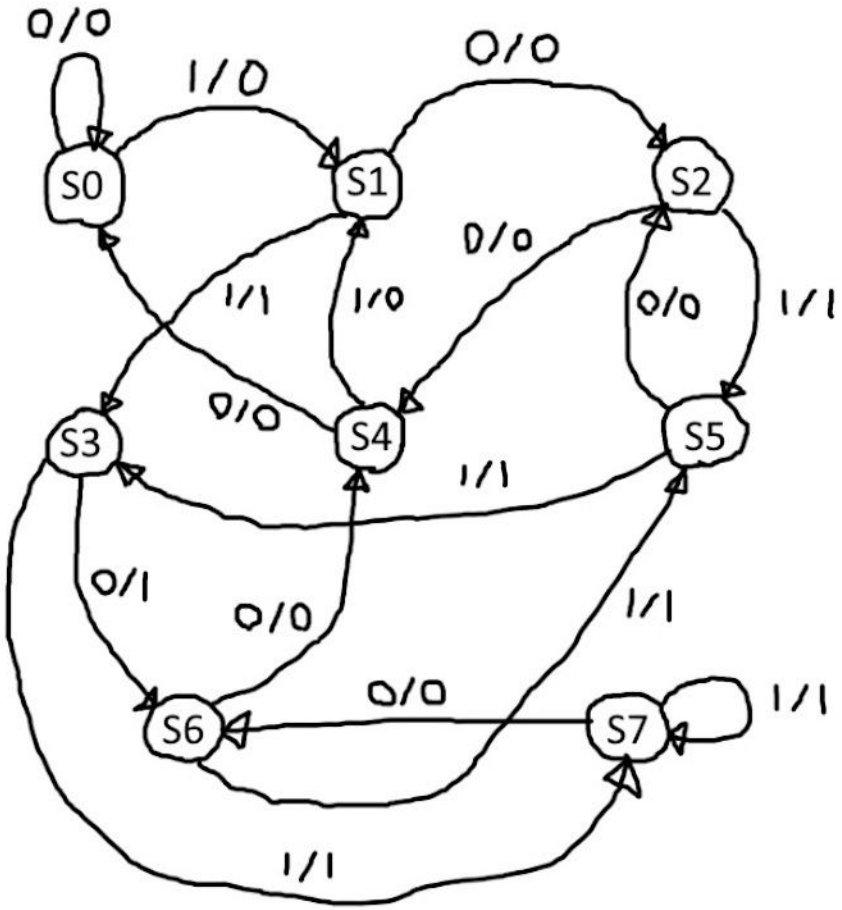
S0 = (Q1=0,Q2=0,Q3=0)
S1 = (Q1=0,Q2=0,Q3=1)
S2 = (Q1=0,Q2=1,Q3=0)
S3 = (Q1=0,Q2=1,Q3=1)
S4 = (Q1=1,Q2=0,Q3=0)

S5, S6, S7 mai raggiunte



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

21)



S0 = (Q3=0,Q2=0,Q1=0)
S1 = (Q3=0,Q2=0,Q1=1)
S2 = (Q3=0,Q2=1,Q1=0)
S3 = (Q3=0,Q2=1,Q1=1)
S4 = (Q3=1,Q2=0,Q1=0)
S5 = (Q3=1,Q2=0,Q1=1)
S6 = (Q3=1,Q2=1,Q1=0)
S7 = (Q3=1,Q2=1,Q1=1)

Ok, seguitemi, leggete gli stati da Dx verso Sx:
Immaginate che ci troviamo nello stato S2 = (Q3=0,Q2=1,Q1=0), vuol dire che gli ultimi 3 bit letti sono (0,1,0); se leggessi 1 come prossimo bit, allora gli ultimi 3 bit letti sarebbero (1,0,1), che corrisponde allo stato S5 = (Q3=1,Q2=1, Q1=0), se invece avessi letto 0, gli ultimi 3 bit sarebbero stati (1,0,0), che corrisponde allo stato S4 = (Q3=1,Q2=0,Q1=0), e così via per tutti gli stati...
Credo ci sia più di una soluzione per questo esercizio, ma questa è quella che mi sembrava più intuitiva.

NOTA: Forse potrebbe essere più intuitivo risolvere l'esercizio con gli automi alla Moore piuttosto che alla Mealy, provateci!

22)

Q3	Q2	Q1	X	D3	D2	D1	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	1
0	1	0	0	1	0	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	1	1
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1

Lascio a voi la costruzione del circuito, è solo un lungo processo meccanico, ma niente di troppo complesso a livello di ragionamento.

23)

Pensiamoci, riceviamo in input i bit dal MENO significativo al PIU' significativo, quindi:

- Se il bit meno significativo è 1, nella conversione a Complemento a 2 sarà posto a 0, e poi gli sarà aggiunto 1, tornando così a valere 1.
- Se il bit meno significativo è 0, nella conversione a Complemento a 2 sarà posto a 1, e poi gli sarà aggiunto 1, tornando così a valere 0.

- Non appena si incontra un bit posto a 1, a prescindere se sia LSB o meno, ciò vuol dire che il "riporto" dovuto all'aver sommato 1 al bit meno significativo ha smesso di influenzare il valore del prossimo bit (dopo la conversione).

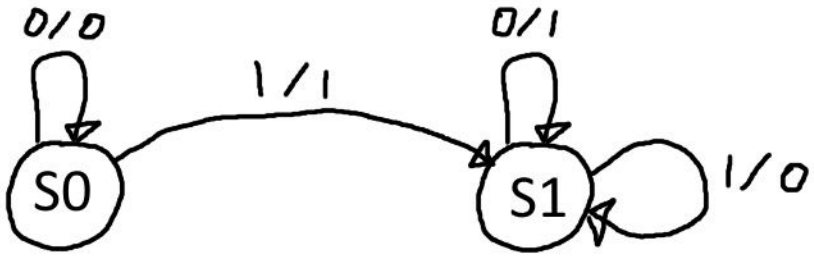
Ci basterà quindi "flippare" il valore di qualsiasi prossimo bit in input.

Ex 1: 0011 -> 1100 + 0001 = 1101
Ex 2: 1100 -> 0011 + 0001 = 0100

Q	X	D	Y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

S0: (Q = 0)
S1: (Q = 1)

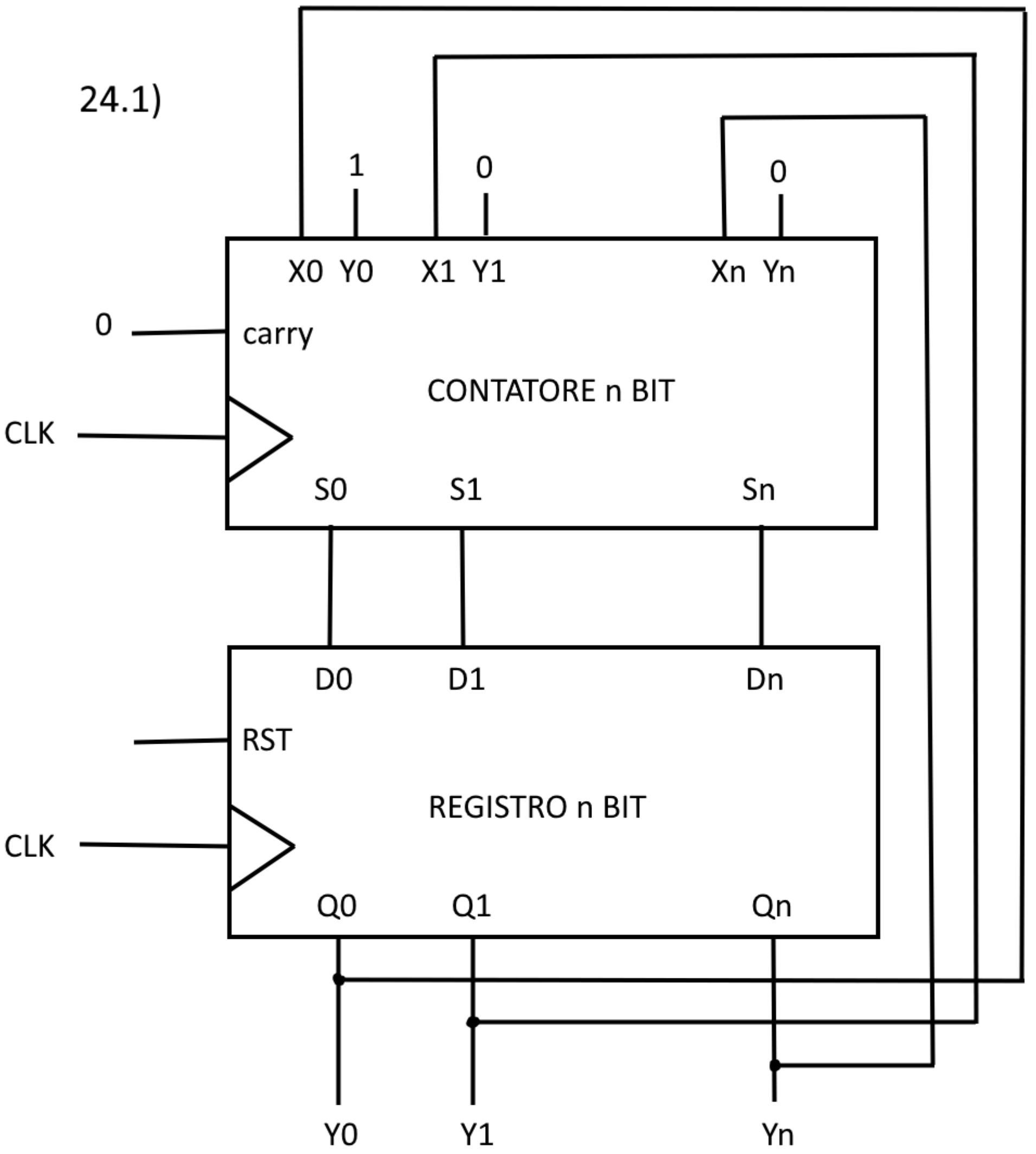
$$\begin{cases} D = Q + X \\ Y = Q \text{ xor } Y \end{cases}$$



L'ESERCIZIO NON LO RICHIEDE, MA SE VOLETE, PROVATE A FARE ANCHE IL CIRCUITO, DOVREBBE ESSERE MOLTO SEMPLICE

24)

24.1)

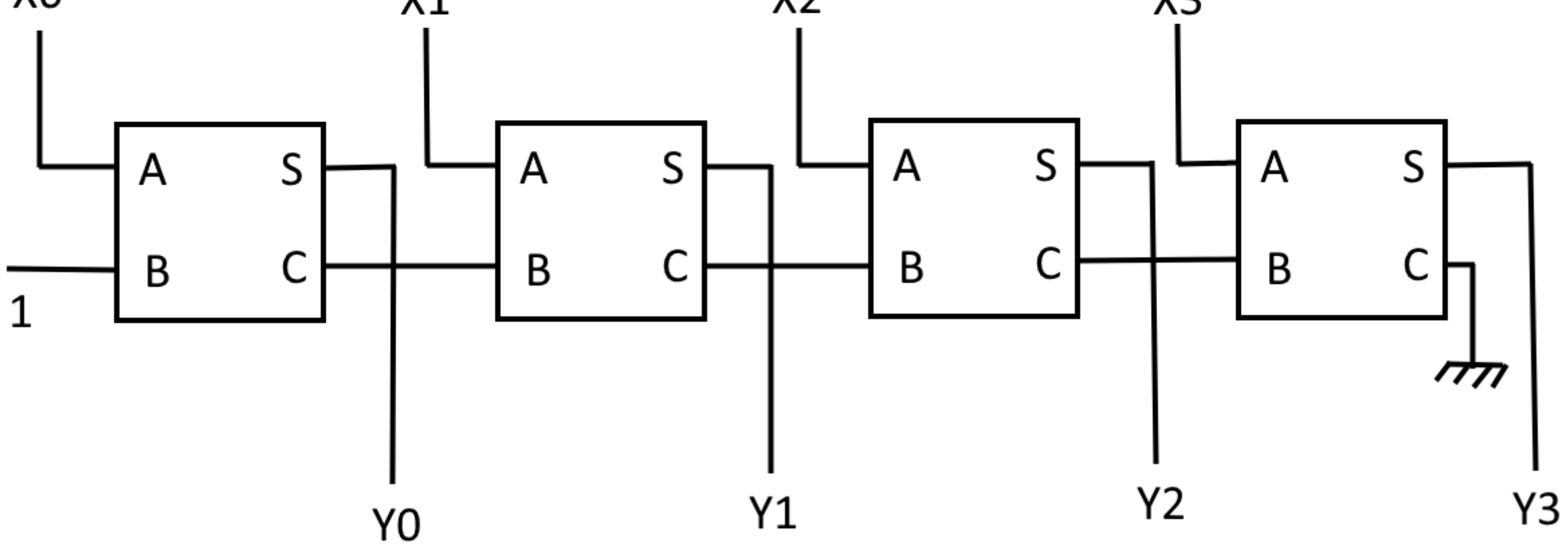


Il circuito è molto più semplice di quel che sembra:
Ad ogni ciclo di Clock il Registro passa il valore memorizzato al Sommatore, il quale lo incrementa di 1 e lo passa come nuovo input da memorizzare al Registro, quando RST viene posto ad 1, il registro azzera i valori memorizzati.

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'Es. 5, composto da 4 HA che incrementava di 1 il numero in input



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