

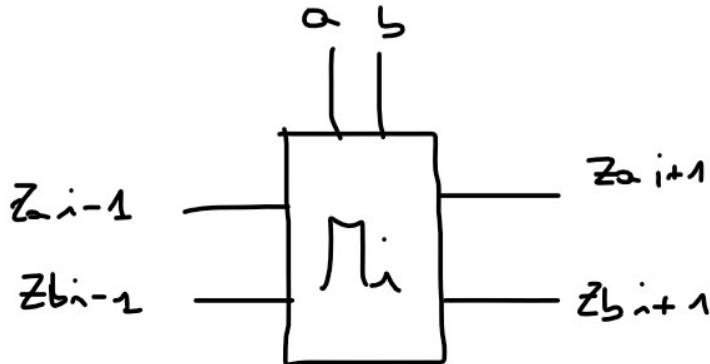
Lezione 13 Reti iterative 2 + Reti sequenziali 1

mercoledì 25 ottobre 2023 10:19

Vediamo ora un esempio di comparatore : prendiamo due numeri e confrontiamoli :

A=010011 Risultato di comparazione : $\begin{array}{r} 110000 \\ 001100 \end{array}$

B=000111



Se i bit sono uguali come uscita avrà la propagazione dello stato precedente (tratto rosso è propagazione) . Andiamo ora a costruire la tabella di verità :

Za (i-1)	Zb(i-1)	Ai	Bi	Za(i+1)	ZB(i+1)
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0

Le altre condizioni sono don't care . Minimizziamo con karnaugh :

a_{i-1}	b_{i-1}	c_0	c_1	c_2	c_3
0	0	0	1	1	1
0	1	0	-	-	1
1	0	0	-	-	0
1	1	0	1	0	0

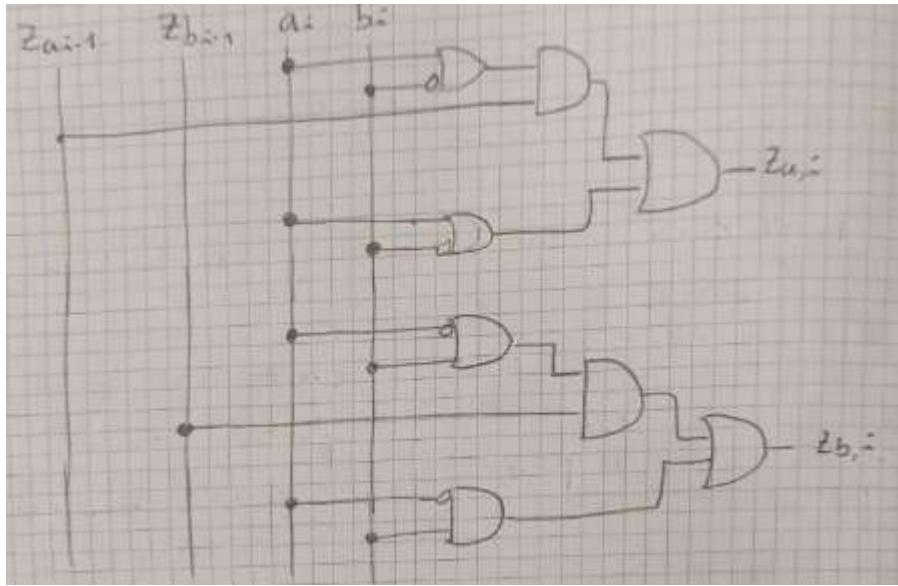
$$\begin{aligned}
 \underline{\underline{z_{ai}}} &= a_i z_{a,i-1} + \bar{b}_i z_{b,i-1} + a_i b_i = \\
 &= z_{a,i-1} (a_i + \bar{b}_i) + a_i b_i
 \end{aligned}$$

Per la propagazione lo stesso :

a_i	b_i	c_0	c_1	c_2	c_3
0	0	0	0	0	0
0	1	1	-	-	0
1	0	1	-	-	1
1	1	1	1	0	0

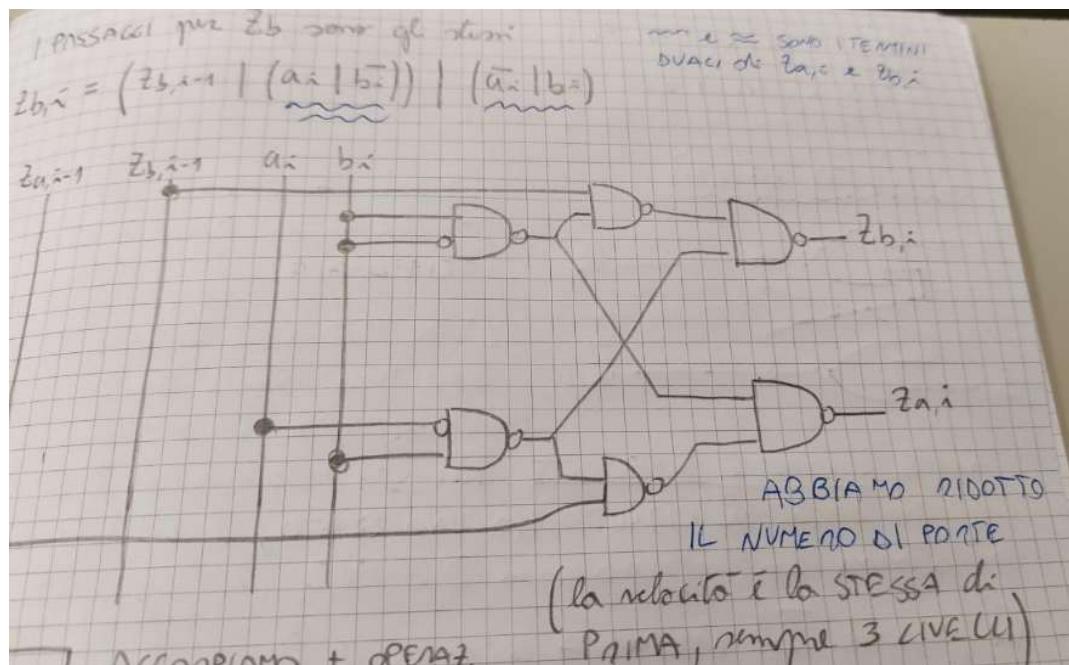
$$\begin{aligned}
 \underline{\underline{z_{bi}}} &= b_i z_{b,i-1} + \bar{a}_i z_{a,i-1} + b_i \bar{a}_i = \\
 &= z_{b,i-1} (b_i + \bar{a}_i) + \bar{a}_i b_i
 \end{aligned}$$

Il cui circuito è il seguente :

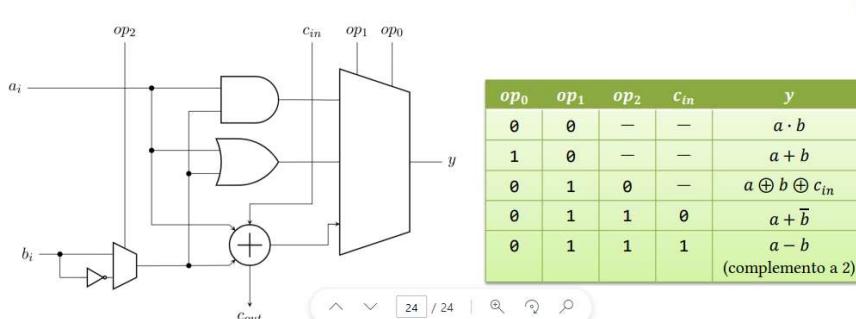
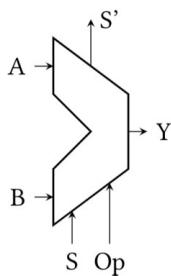


In ogni modulo ho 8 porte : possiamo semplificare questo circuito minimo?? Si in forma analitica :

$$\begin{aligned}
 z_{a,i} &= z_{a,i-1} (a_i + \bar{b}_i) + a_i b_i = z_{a,i-1} (a_i + \bar{b}_i) \cdot \bar{a}_i \bar{b}_i = \\
 &= z_{a,i-1} (a_i + \bar{b}_i) \mid \bar{a}_i \bar{b}_i = (z_{a,i-1} \mid (\overline{a_i + \bar{b}_i})) \mid (\bar{a}_i \mid \bar{b}_i) = \\
 &= (z_{a,i-1} \mid (\overline{\bar{a}_i \bar{b}_i})) \mid (\bar{a}_i \mid \bar{b}_i) = \\
 &= (z_{a,i-1} \mid (\underbrace{\bar{a}_i \mid b_i})) \mid (\underbrace{a_i \mid \bar{b}_i})
 \end{aligned}$$

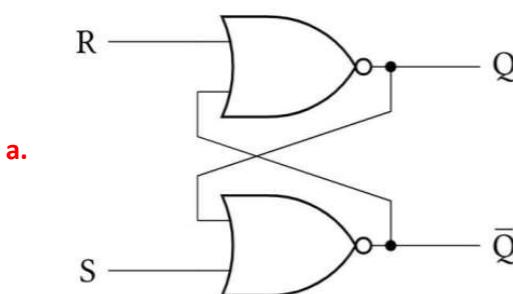


Proviamo a mettere insieme vari pezzi visti finora : andiamo a vedere un componente programmabile : la **ALU** (arithmetic and logic unit): dispositivo che opera su parole di dimensione fissa , permette di fare molte operazioni ed ha bisogno di due segnali di ingresso (op-code : codice che determina operazione da fare) : fa diminuire i fili e la circuiteria :



Le operazioni implementate sono 5 appunto delle 8 (3 bit di controllo) . Il dispositivo che contiene gli op-code è un mux a 2 ingressi : in base alla configurazione restituisce la funzione opportuna . La terza configurazione è un sommatore : full-adder , mentre nel penultima operazione è un inverter, invece nell'ultimo caso di usa come sottrattore (somma complemento a 2). **Il circuito calcola tutte le operazioni possibili contemporaneamente.** I circuiti visti finora sono chiamata combinatori in quanto combinano variabili booleane per avere una funzione in uscita : ricordiamo la stabilità del circuito : non possiamo costruire nessun elemento di memoria . Come possiamo ovviare a ciò? Usiamo l'uscita come ingresso : anello di retroazione . Vediamo un esempio :

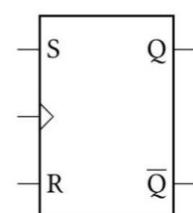
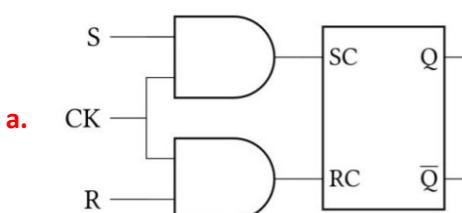
1. **Latch (questo circuito ha uno stato):** output precedente influenza input successivo . Consente di immagazzinare un'informazione se il risultato non cambia al variare del tempo.



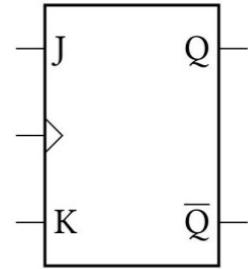
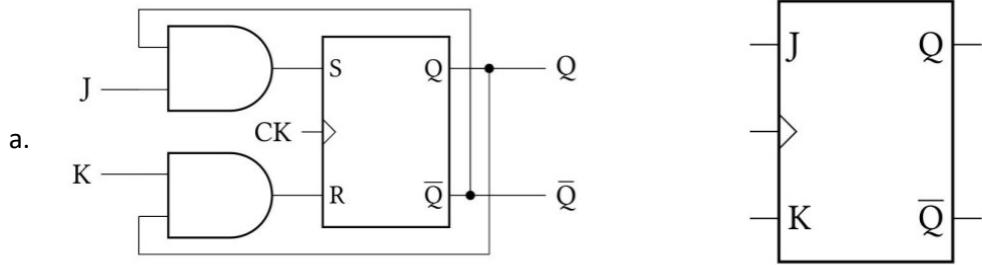
S	R	Q	Q'
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	indeterminato
1	1	1	indeterminato

- Da notare che il valore di r e s se sono soli ad uno , il latch 1-0 il latch è in modo set, analogamente per il reset (0-1).
- c. **Unico stato inconsistente si ha quando entrambi gli input sono settati ad 1**

2. **Flip flop (miglioria del latch: output stabili):**

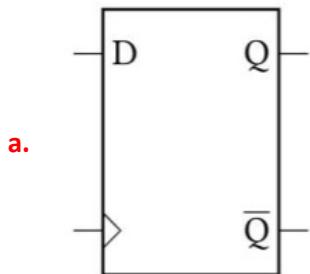


- b. Il clock serve a tenere il tempo e tiene traccia del tempo che le reti ci mettono a stabilizzarsi : quindi evita di leggere segnali instabili .
- c. Anche questo dispositivo non sopporta la configurazione 11
3. **Flip flop J-k** : segnali set / reset nominati e sfrutta seconda rete di retrazione per filtrare il segnale



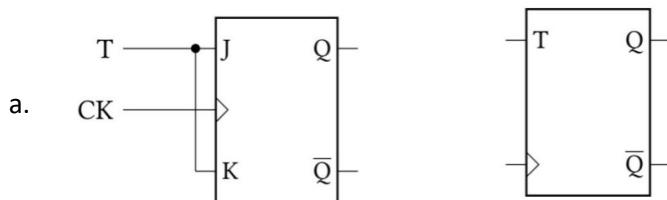
b. Con questo ff non è un problema la configurazione 11

4. **Flip-flop D (scrive un solo bit)**: non mantiene stabile l'input . Ha due segnali opposti

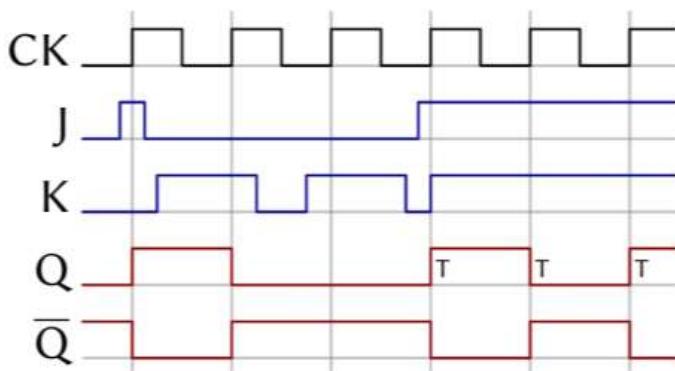


b. D sta per delay : se 1 su delay devo aspettare un colpo di clock per avere l'uscita : introduce un ritardo di un colpo di clock per avere uscita

5. Flip-flop T(switch) : inverte il valore dell'uscita :



Come funziona il clock nel tempo? Segnale che per un certo tempo vale 0 e per un certo periodo vale 1 : andiamo a vedere il fronte di commutazione (passaggio 0-1 e viceversa): momento in cui il mio ff commuta :



Vediamo ora un esempio : come si evolve il segnale nel latch s-r:

