

Lezione 13 Reti iterative 2 + Reti sequenziali 1

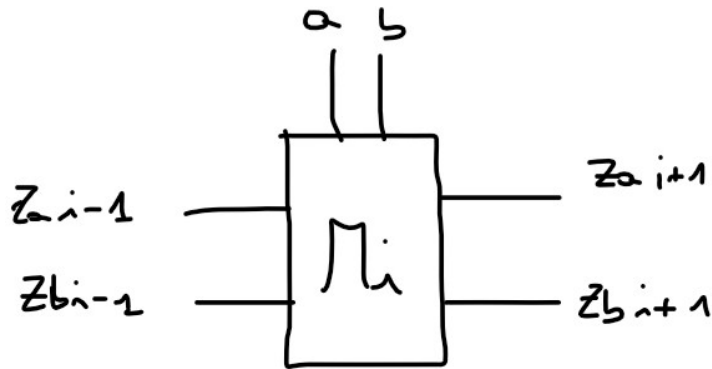
mercoledì 25 ottobre 2023 10:19

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

A=010011 Risultato di comparazione : 110000

001100

B=000111



Se i bit sono uguali come uscita avrò 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 :

Handwritten Karnaugh map for the function $z_{a,i}$. The map is a 4x4 grid with inputs a_{i-1} and b_{i-1} on the axes. The values are as follows:

| $a_{i-1} \backslash b_{i-1}$ | 00 | 01 | 11 | 10 |
|------------------------------|----|----|----|----|
| 00 | 0 | 1 | 1 | 1 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 0 | 1 | 0 |

Groupings: A blue circle highlights the 2x2 square of 1s in the first two rows. An orange circle highlights the 2x2 square of 1s in the last two columns. A pink circle highlights the 2x2 square of 1s in the last two rows and last two columns.

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

Per la propagazione lo stesso :

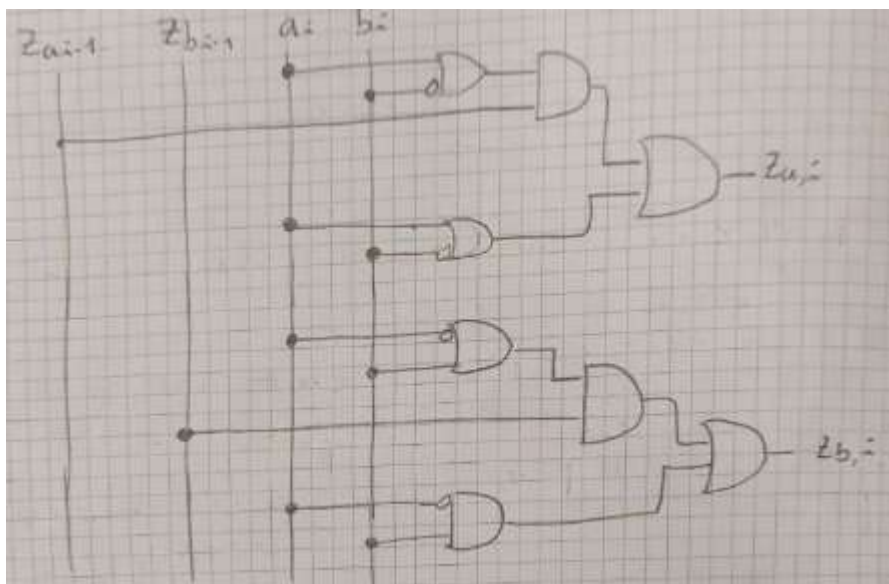
Handwritten Karnaugh map for the function $z_{b,i}$. The map is a 4x4 grid with inputs a_i and b_{i-1} on the axes. The values are as follows:

| $a_i \backslash b_{i-1}$ | 00 | 01 | 11 | 10 |
|--------------------------|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 0 | 0 |

Groupings: A yellow circle highlights the 2x2 square of 1s in the last two rows and last two columns. A pink circle highlights the 2x2 square of 1s in the last two rows and last two columns.

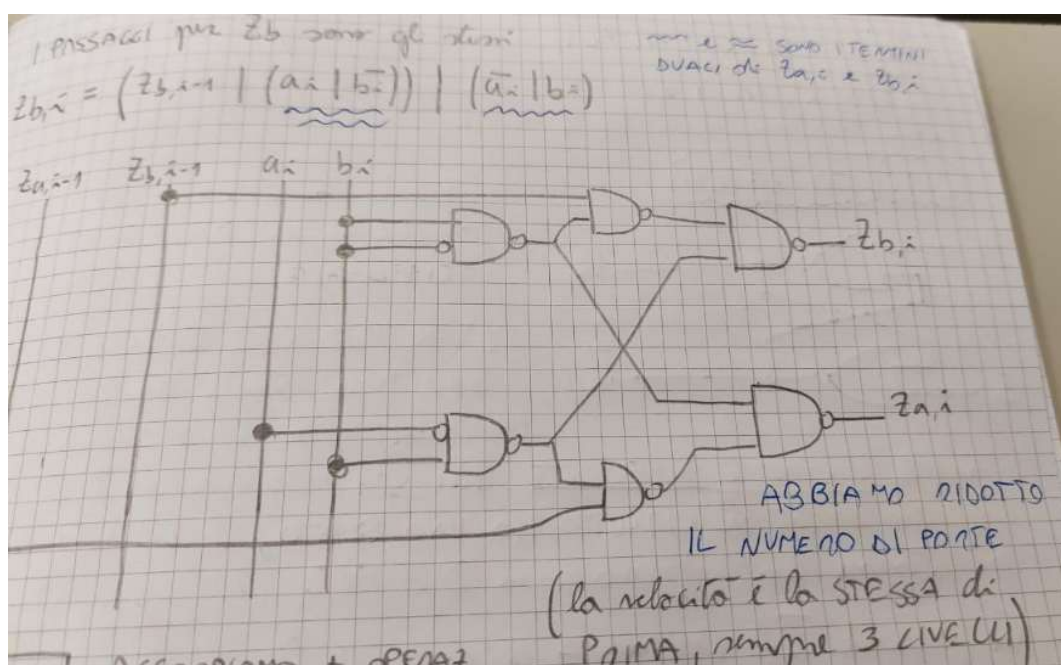
$$\begin{aligned} \underline{z_{b,i}} &= b_i z_{b,i-1} + z_{b,i-1} \bar{a}_i + b_i \bar{a}_i = \\ &= z_{b,i-1} (\bar{a}_i + b_i) + \bar{a}_i b_i \end{aligned}$$

Il cui circuito è il seguente :

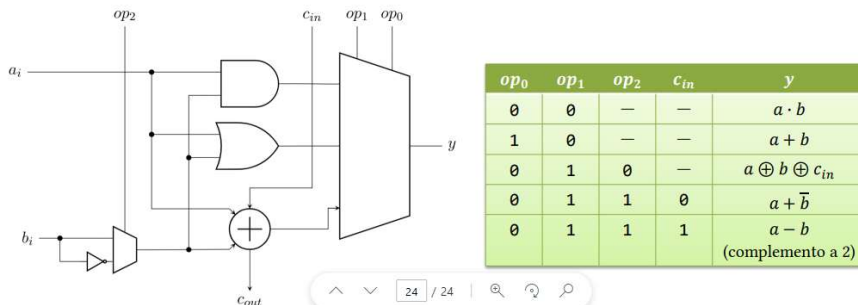
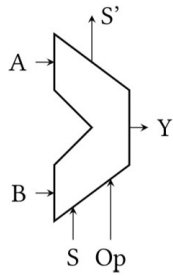


In ogni modulo ho 8 porte : possiamo semplificare questo circuito minimo?? Si in forma analitica :

$$\begin{aligned}
 Z_{a,i} &= Z_{a,i-1} (a_i + b_i) + a_i b_i = Z_{a,i-1} (a_i + b_i) \cdot \overline{a_i b_i} = \\
 &= Z_{a,i-1} (a_i + b_i) \mid \overline{a_i b_i} = (Z_{a,i-1} \mid (\overline{a_i + b_i})) \mid (a_i \mid b_i) = \\
 &= (Z_{a,i-1} \mid (\overline{a_i} \mid \overline{b_i})) \mid (a_i \mid b_i) = \\
 &= (Z_{a,i-1} \mid (\overline{a_i} \mid \overline{b_i})) \mid (a_i \mid b_i)
 \end{aligned}$$

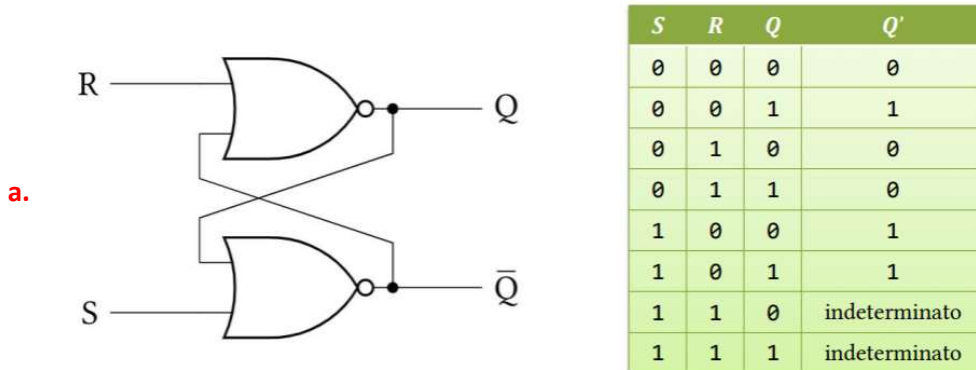


Proviamo a mettere insieme vari pezzi visti finora : andiamo a vedere un componente programmabile : la **ALU (arithmetic and logic unit)**: dispositivo che opera su parole di dimensione fissa , permette di fare molte operazioni ed ha bisogno di due segnali di ingresso (op-code : codice che determina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



Le operazioni implementate sono 5 apposto 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 chiamati 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 :

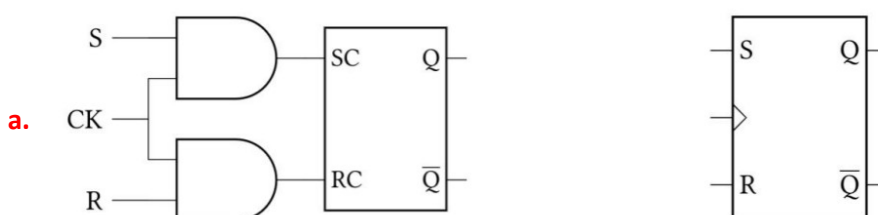
- Latch (questo circuito ha uno stato):** output precedente influenza input successivo . Consente di immagazzinare un'informazione se il risultato non cambia al variare del tempo.



- Da notare che il valore di r e s se sono soli ad uno , il latch 1-0 il latch e in modo set, analogamente per il reset (0-1).

- Unico stato inconsistente si ha quando entrambi gli input sono settati ad 1**

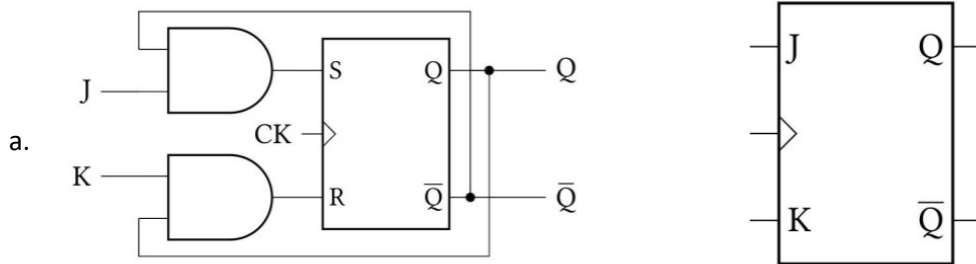
- Flip flop (miglioria del latch: output stabili):**



- 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 .

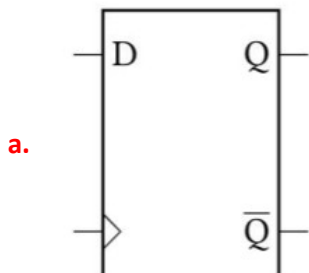
- Anche questo dispositivo non sopporta la configurazione 11

- Flip flop J-k** : segnali set / reset nominati e sfrutta seconda rete di retroazione per filtrare il segnale



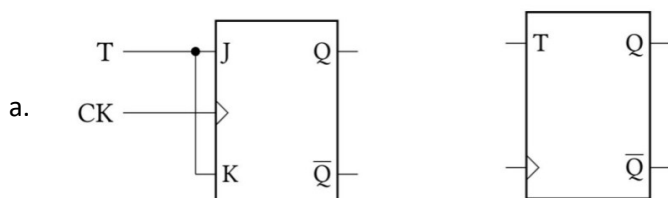
b. Con questo ff non è un problema la configurazione 11

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

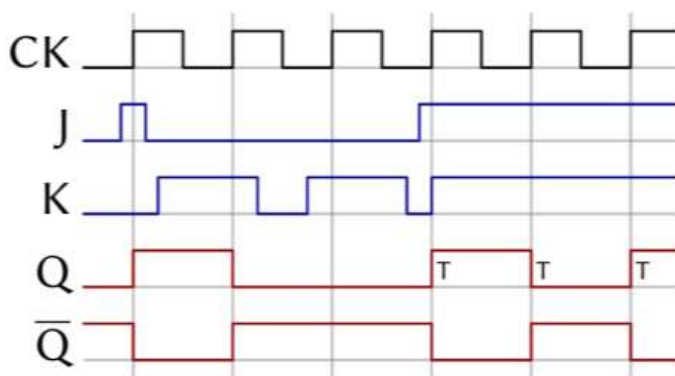


b. D sta per delay : se 1 su delay devo aspettare un colpo di clock per avere l'uscita :
introduce un ritardo di un colpo di clock per avere uscita

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:

