6. Registri di memorizzazione e contatori

Definizioni

Un registro è un circuito capace di gestire dati, ed è formato da Flip Flop per memorizzarli, ed eventualmente un circuito combinatorio che determina il dato che deve essere trasferito nei Flip Flop.

Un contatore è un registro, il cui stato evolve secondo una sequenza predeterminata di stati, tramite impulsi di clock. La parte di circuito combinatorio del contatore si occupa di produrre la sequenza di stati binari specificata.

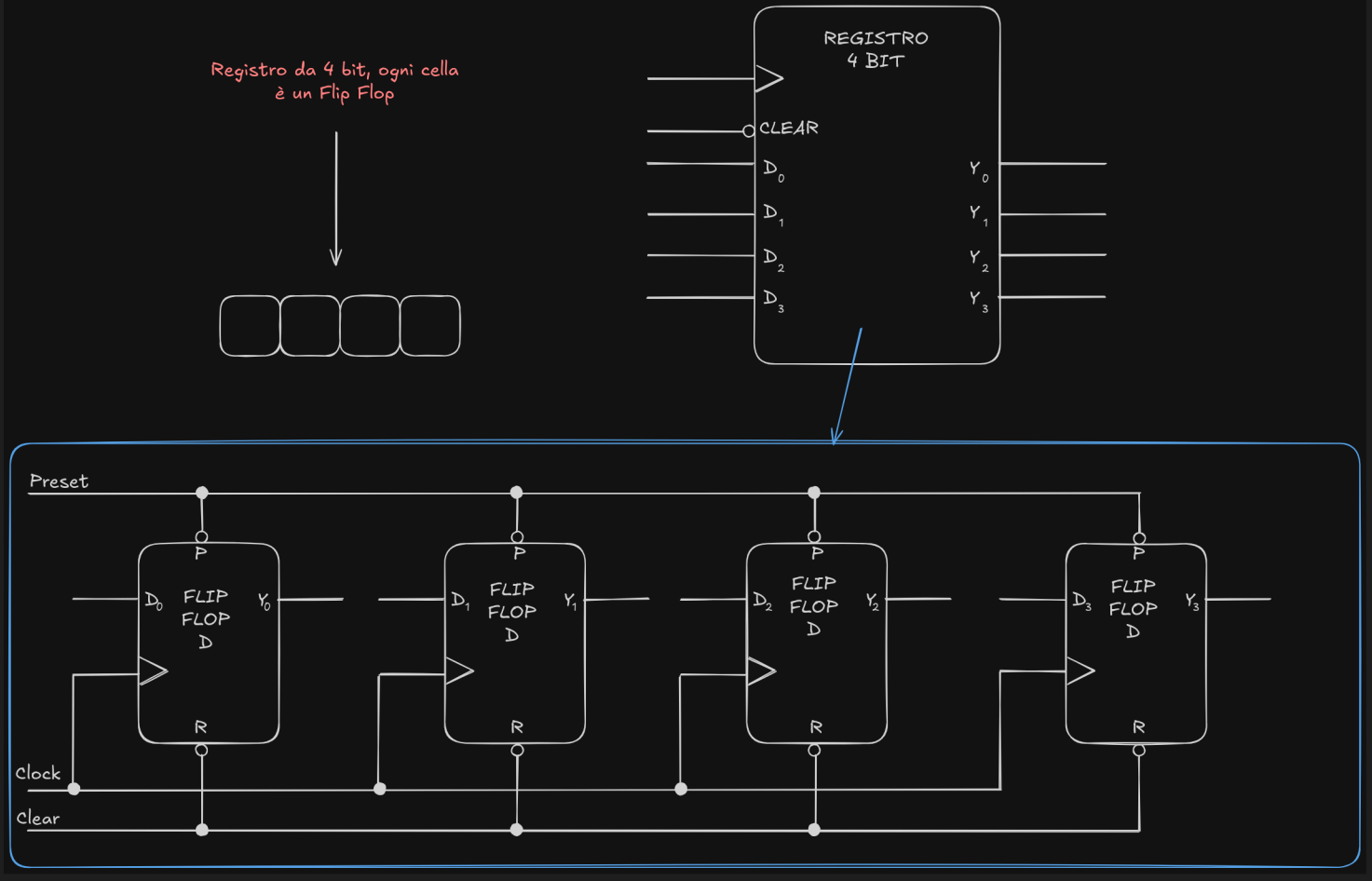
Nonostante i contatori siano registri particolari, in genere vengono differenziati perché hanno usi diversi: i registri sono utili per immagazzinamento e manipolazione dei dati, mentre i contatori vengono usati nei circuiti che controllano la sequenza delle operazioni eseguite.

Entrambi i componenti sono considerati blocchi sequenziali, facendo uso di elementi di memoria.

Registri

I registri hanno il compito di memorizzare informazioni, e per farlo usano i Flip Flop: poiché ogni FF memorizza un solo bit, un registro da n bit utilizza n Flip Flop.

Inoltre, è presente anche un segnale asincrono di clear, che imposta i bit a `0`. Si dice asincrono perché una volta impostato il segnale di clear a `0`, viene mandato ai Flip Flop, che reagiscono impostando il valore di output a `0` immediatamente, senza dover aspettare il prossimo fronte di clock. Al tempo stesso, sarebbe possibile inserire anche un segnale analogo di preset che imposta i bit ad `1` senza dover aspettare il prossimo fronte di clock. I segnali asincroni di clear e preset non devono essere attivi contemporaneamente, o si entra in uno stato non valido come se stessimo fornendo input S= `1` ed R= `1` contemporaneamente nel latch SR.



Metodi di caricamento dei registri

È necessario implementare un metodo non solo per caricare i registri, ma anche per indicare quando vogliamo caricare valori, e quando invece vogliamo che siano restituiti i valori memorizzati. Ci sono due metodologie di caricamento:

- Caricamento in parallelo
- Caricamento seriale

Caricamento in parallelo

Il caricamento in parallelo consiste nell'inviare contemporaneamente il caricamento a tutti i Flip Flop.

Per fare ciò ho bisogno

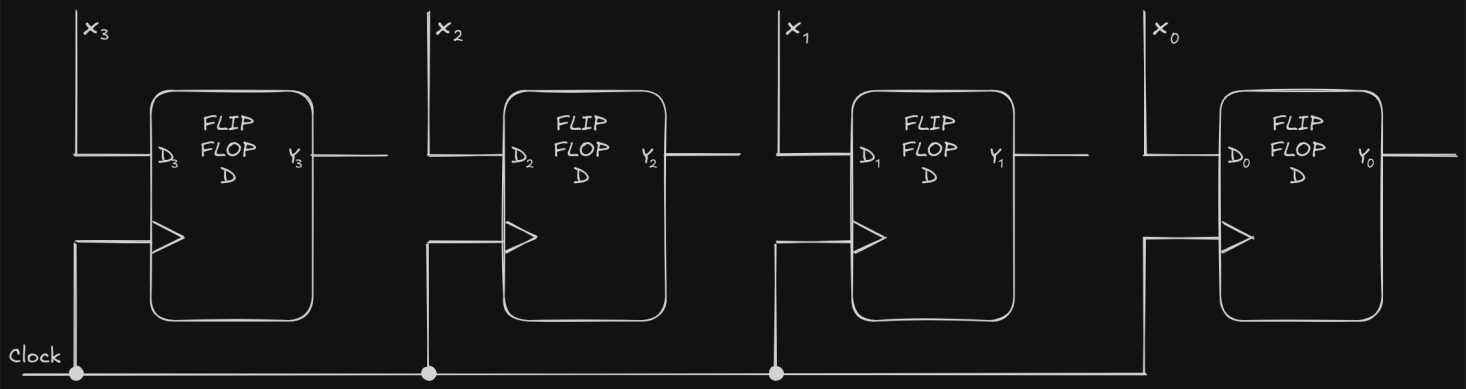
- Del clock, che sincronizza i Flip Flop per prendere gli input
- Di un segnale di load, per sapere se i Flip Flop devono caricare nuovi valori, o se devono restituire quelli memorizzati

Per capire meglio, diamo un'occhiata a questi due registri formati con Flip Flop D e Flip Flop SR (Master-Slave).

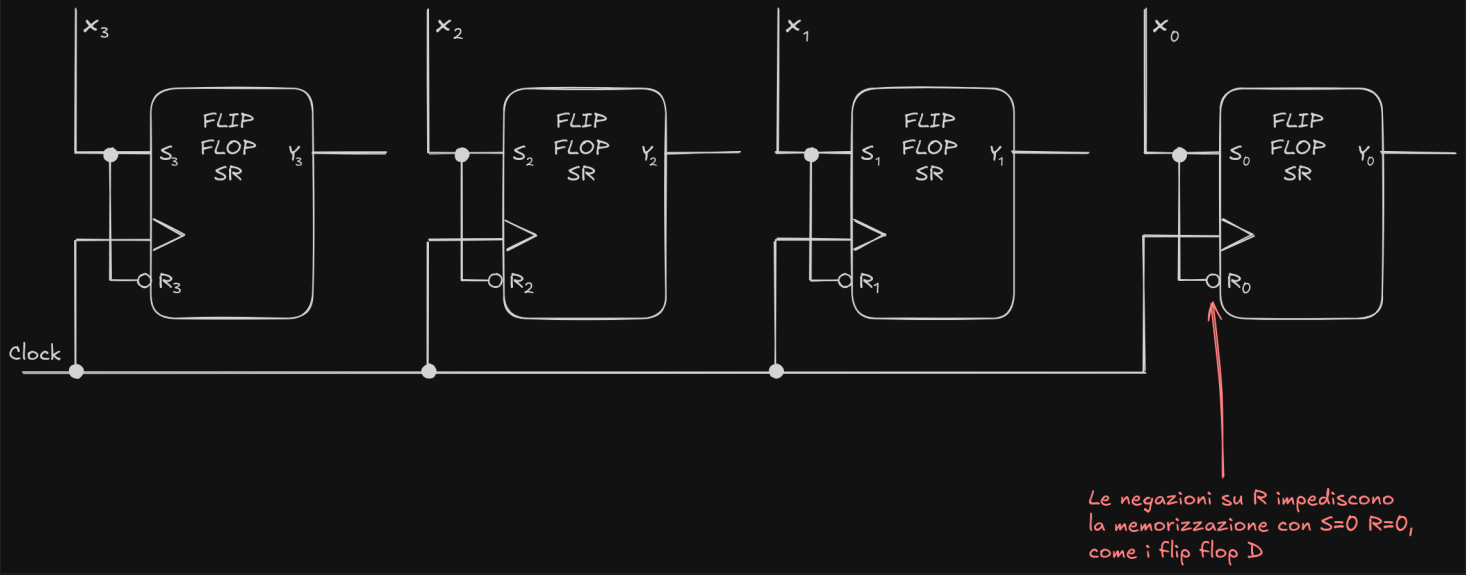
> 🖍️ Omissione segnale di clear e preset

Per praticità, il segnale di clear e preset verranno d'ora in poi omessi se non rilevanti.

CARICAMENTO PARALLELO CON FLIP FLOP D



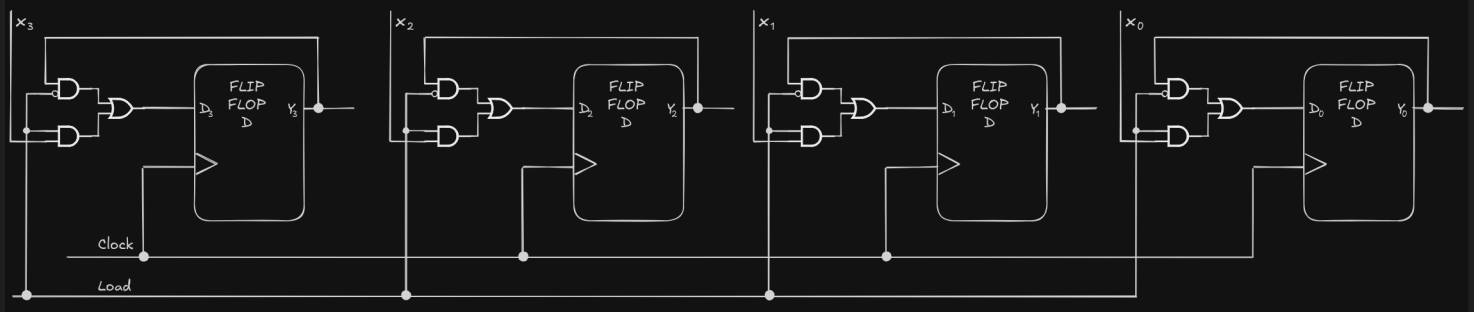
CARICAMENTO PARALLELO CON FLIP FLOP SR



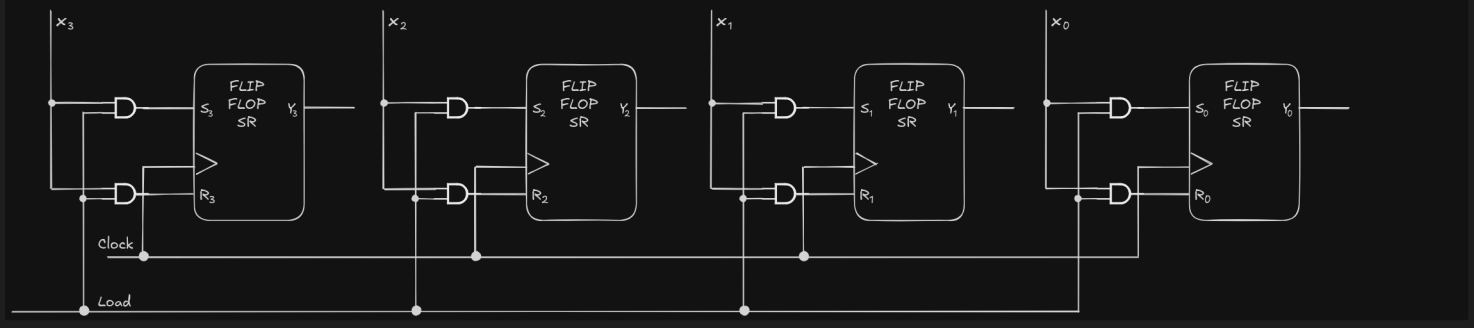
Ad ogni colpo di clock, i valori di x_3, x_2, x_1 ed x_0 vengono mandati contemporaneamente in input ai Flip Flop, che possono eseguire solo operazioni di set o reset. Per effettuare la memorizzazione, posso usare un segnale di load, che quando è `1` prende in ingresso i valori di input, e quando è `0` invece restituisce il valore memorizzato:

- Con i Flip Flop SR posso mettere in AND il segnale di load con gli input S ed R, così per load= `0` la combinazione in input è S= `0` ed R= `0`
- Con i Flip Flop D posso mettere in AND il segnale di load una volta con l'output Y del Flip Flop stesso, ed una volta con il valore di ingresso, per poi metterli in OR per indicare l'input D. Praticamente, è come se ci fosse un MUX che prende in input il valore memorizzato ed il nuovo ingresso D, e load fa da segnale di controllo che decide quale dei due segnali deve passare.

CARICAMENTO PARALLELO CON FLIP FLOP D



CARICAMENTO PARALLELO CON FLIP FLOP SR

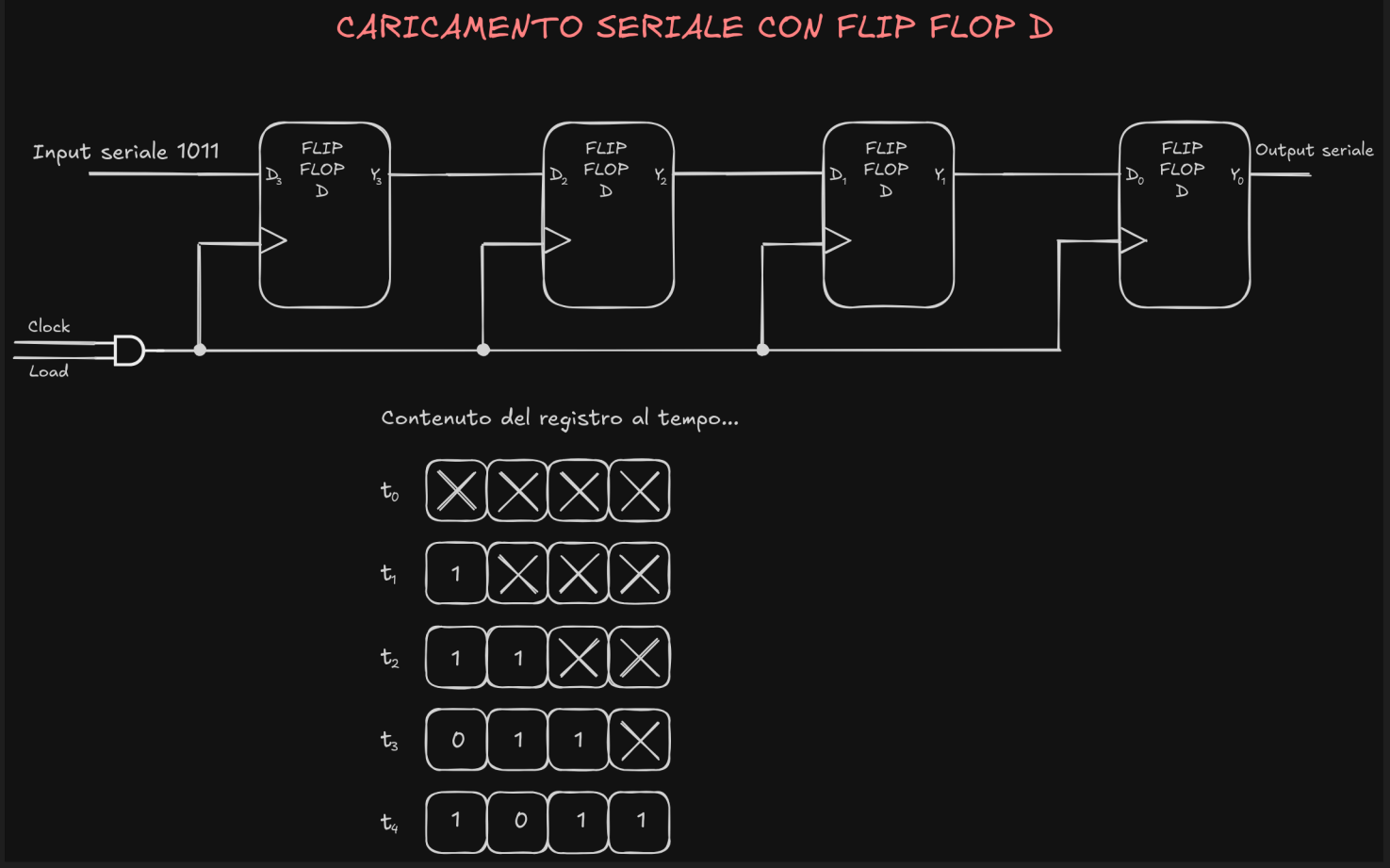


Tutti gli output Y vengono scaricati in output contemporaneamente in un unico ciclo di clock.

Caricamento seriale

Il caricamento seriale consiste nel caricare i gli n Flip Flop uno alla volta, con l'output del primo come input del seguente.

Quindi per implementarlo, metto i Flip Flop in cascata, con l'output di uno che fa da input del seguente. Il segnale di sincronizzazione collegato ai Flip Flop è il risultato di una AND tra il segnale di clock ed il segnale di load: finché load= `1` , i Flip Flop continueranno ad accettare gli input ad ogni fronte di clock, e quando load= `0` , smetteranno invece di essere sensibili al fronte di clock, e quindi di caricarsi. Inserendo un numero di bit superiore al numero dei Flip Flop, il bit più vecchio ad essere inserito (cioé il meno significativo) viene scaricato.



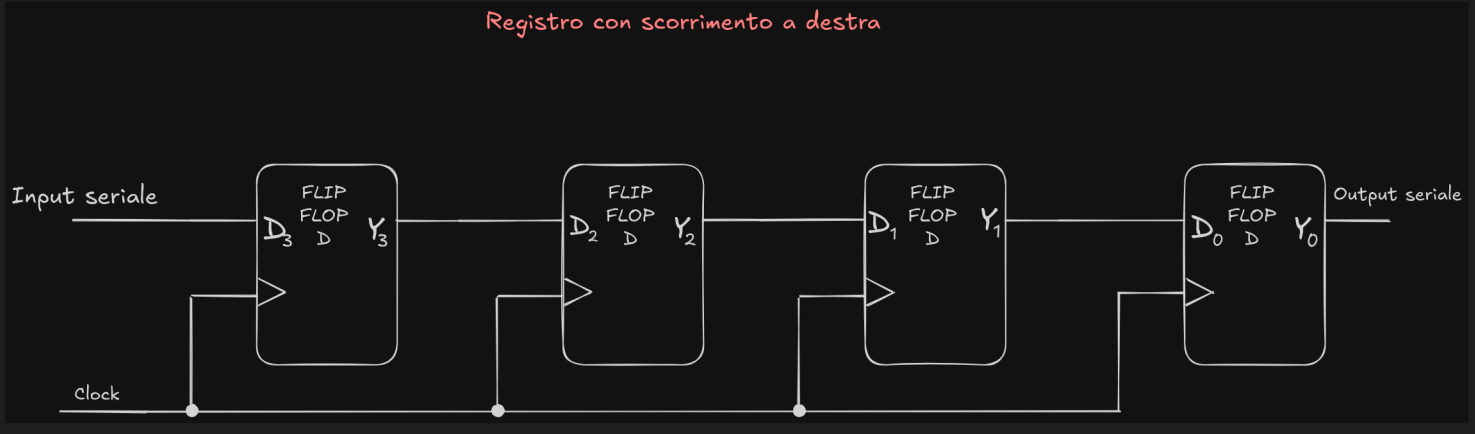
Gli output Y vengono scaricati uno alla volta dal Flip Flop più a destra e quindi otterrò l'output completo dopo n cicli di clock, dove n è il numero di Flip Flop.

Registri shifter

I registri a scorrimento sono registri in grado di far scorrere i bit in una direzione ad ogni colpo di clock. Per farlo, bisogna collegare i registri come nel caricamento seriale, per poi eventualmente modificare i collegamenti per ottenere il verso di scorrimento desiderato.

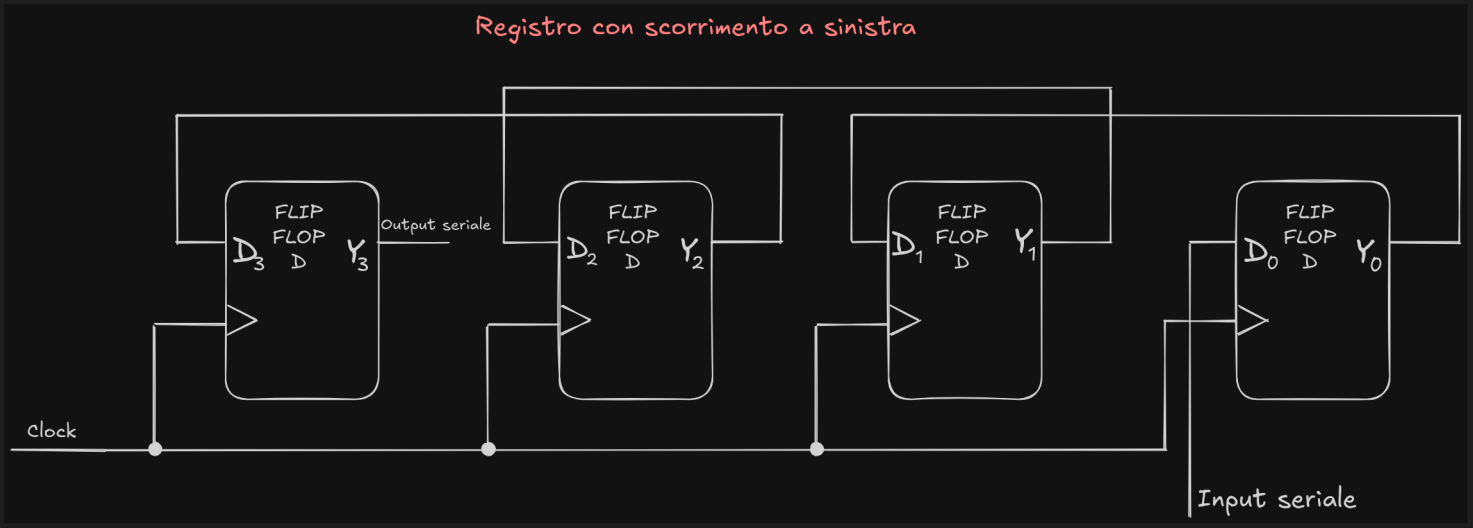
Scorrimento a destra

Per il registro a scorrimento a destra, basta collegare i Flip Flop come nel caricamento seriale: come abbiamo visto infatti, quando un bit viene inserito, gli altri si spostano a destra.



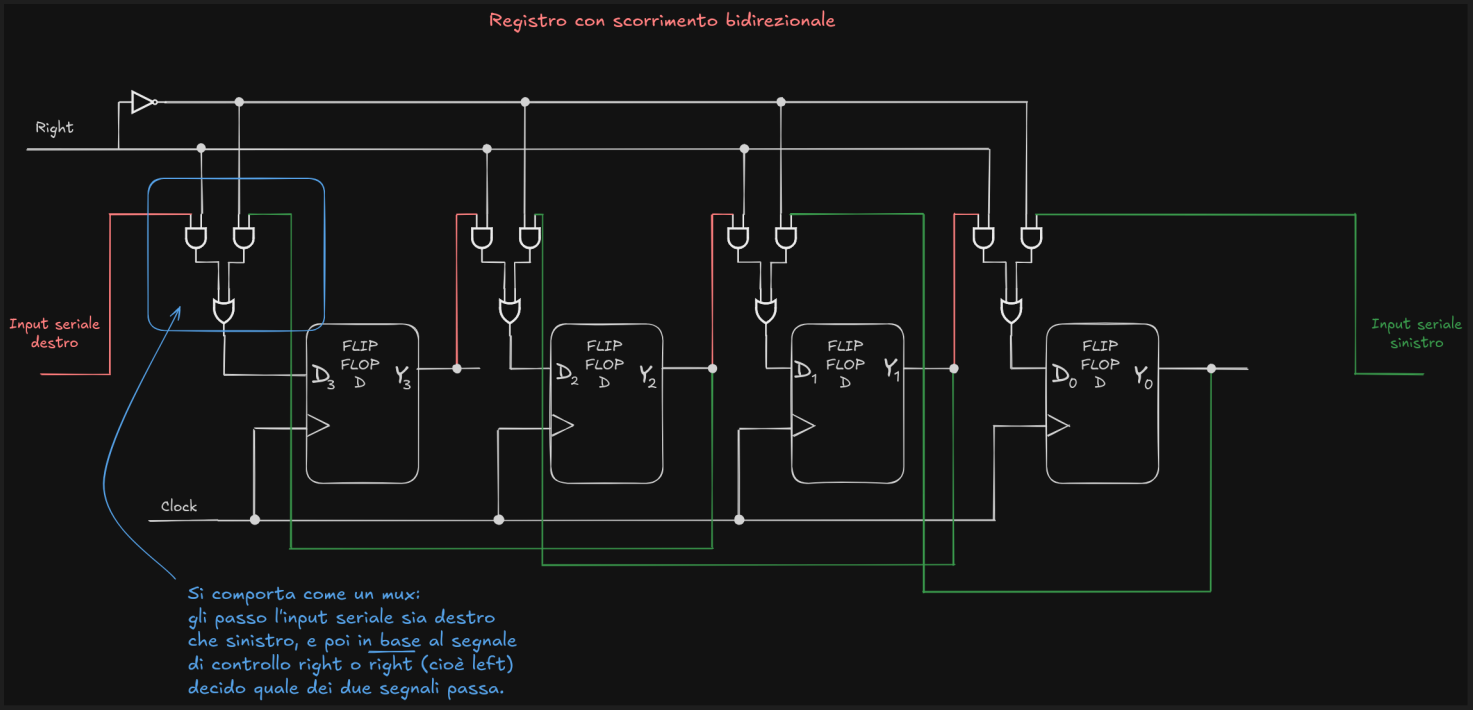
Scorrimento a sinistra

Per il registro a scorrimento a sinistra, il procedimento è il medesimo ma invertito: l'output dell'ultimo flip flop è l'input del precedente e così via, finendo con l'output seriale come output del primo Flip Flop.



Scorrimento bidirezionale

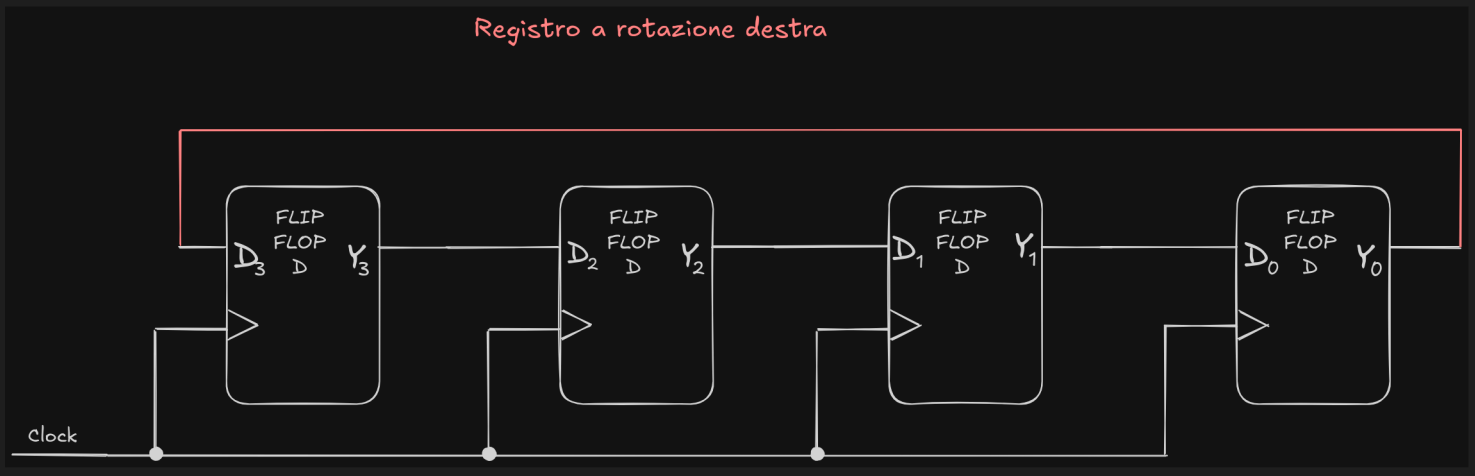
Il registro a scorrimento bidirezionale è un registro che implementa sia lo scorrimento a destra che a sinistra, ma presenta due input seriali ed usa dei mux per decidere dovere fare lo shift: per esempio per inserire un bit facendo scorrimento a destra, dovrò inserire il bit in questione nell'ingresso seriale destro, e dovrò impostare il segnale di controllo *right* su `1`. Al contrario, per inserire un bit facendo scorrimento a sinistra, dovrò inserire il bit in questione nell'ingresso seriale sinistro, e dovrò impostare il segnale di controllo *right* su `0`.



Rotazione destra

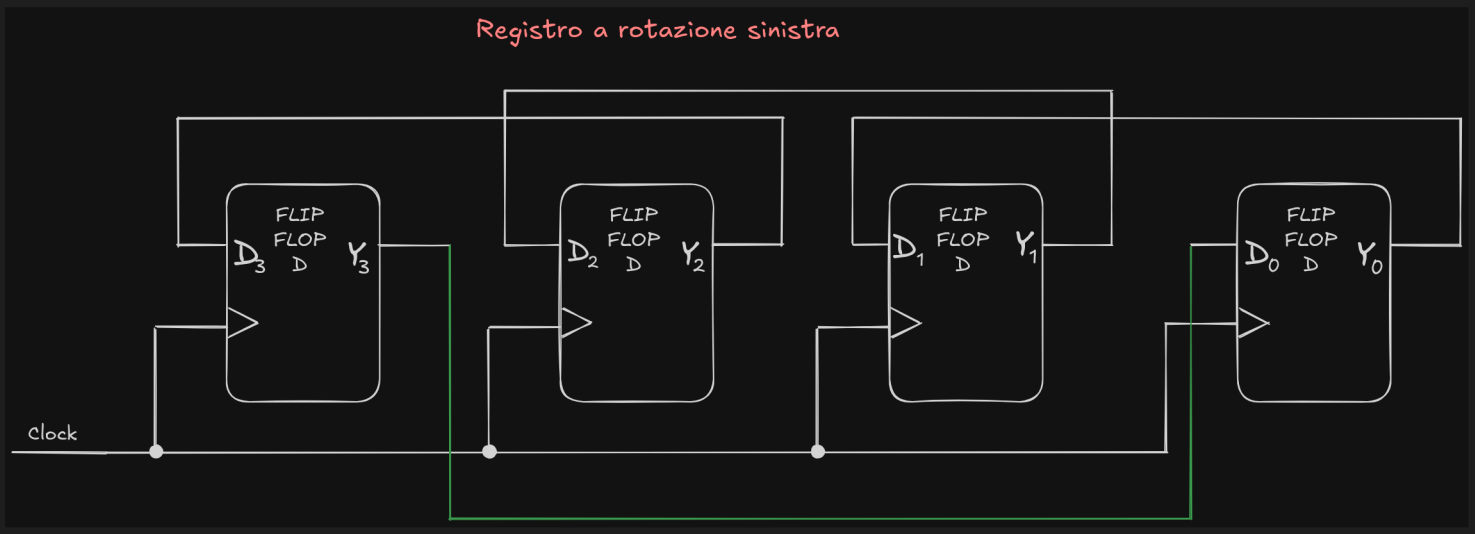
Il registro a rotazione destra è come un registro a scorrimento a destra, ma l'output seriale è collegato all'input del primo Flip Flop a sinistra, per non perdere il bit uscente: in questo

modo, il bit più vecchio ad essere inserito (cioè il meno significativo) invece di essere scaricato fa il giro e diventa il più significativo, memorizzato nel Flip Flop più a sinistra.



Rotazione sinistra

Il registro a rotazione sinistra è come un registro a scorrimento a sinistra, ma l'output seriale è collegato all'input dell'ultimo Flip Flop a destra, per non perdere il bit uscente: in questo modo, il bit più vecchio ad essere inserito (cioè il più significativo) invece di essere scaricato fa il giro e diventa il meno significativo, memorizzato nel Flip Flop più a destra.



Registro universale

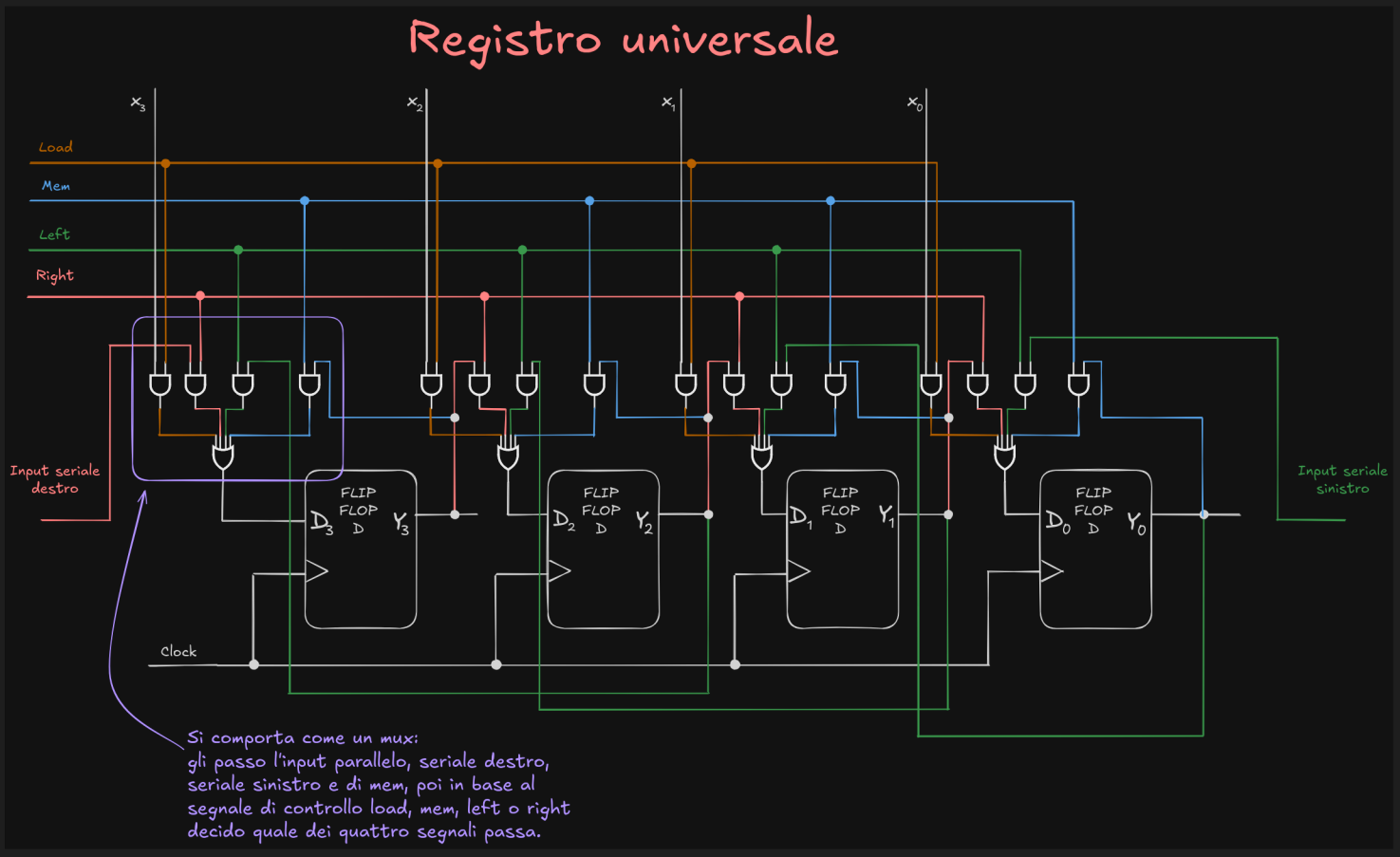
Il registro universale è un registro dotato di:

- Caricamento parallelo - attraverso il segnale *load*
- Scorrimento a destra - attraverso il segnale *right*
- Scorrimento a sinistra - attraverso il segnale *left*
- Memorizzazione - attraverso il segnale *mem* o quando gli altri tre segnali sono impostati a `0`

In un qualsiasi momento, solamente uno di questi segnali può avere valore `1`.

In modo simile allo scorrimento bidirezionale, uso su ogni Flip Flop un insieme di porte AND ed una porta OR per ricreare il comportamento di un multiplexer, ma questa volta oltre a

right e *left* ci sono anche *load* che fa caricare il valore di ingresso, e *mem* che manda in input al Flip Flop il valore di output appena generato.



Contatori

I contatori sono registri usati per contare il numero di occorrenze di un dato evento. Se il registro in questione è formato da n Flip Flop, allora si dice che è un contatore binario modulo 2^n, perché può contare numeri che vanno da 0 a $2^n - 1$. In genere i valori contati sono occorrenze di valori di input o colpi di clock.

Si distinguono in:

- Contatori sincroni, dove tutti i Flip Flop hanno lo stesso segnale di sincronizzazione
- Contatori asincroni, dove i Flip Flop hanno segnali di sincronizzazione diversi

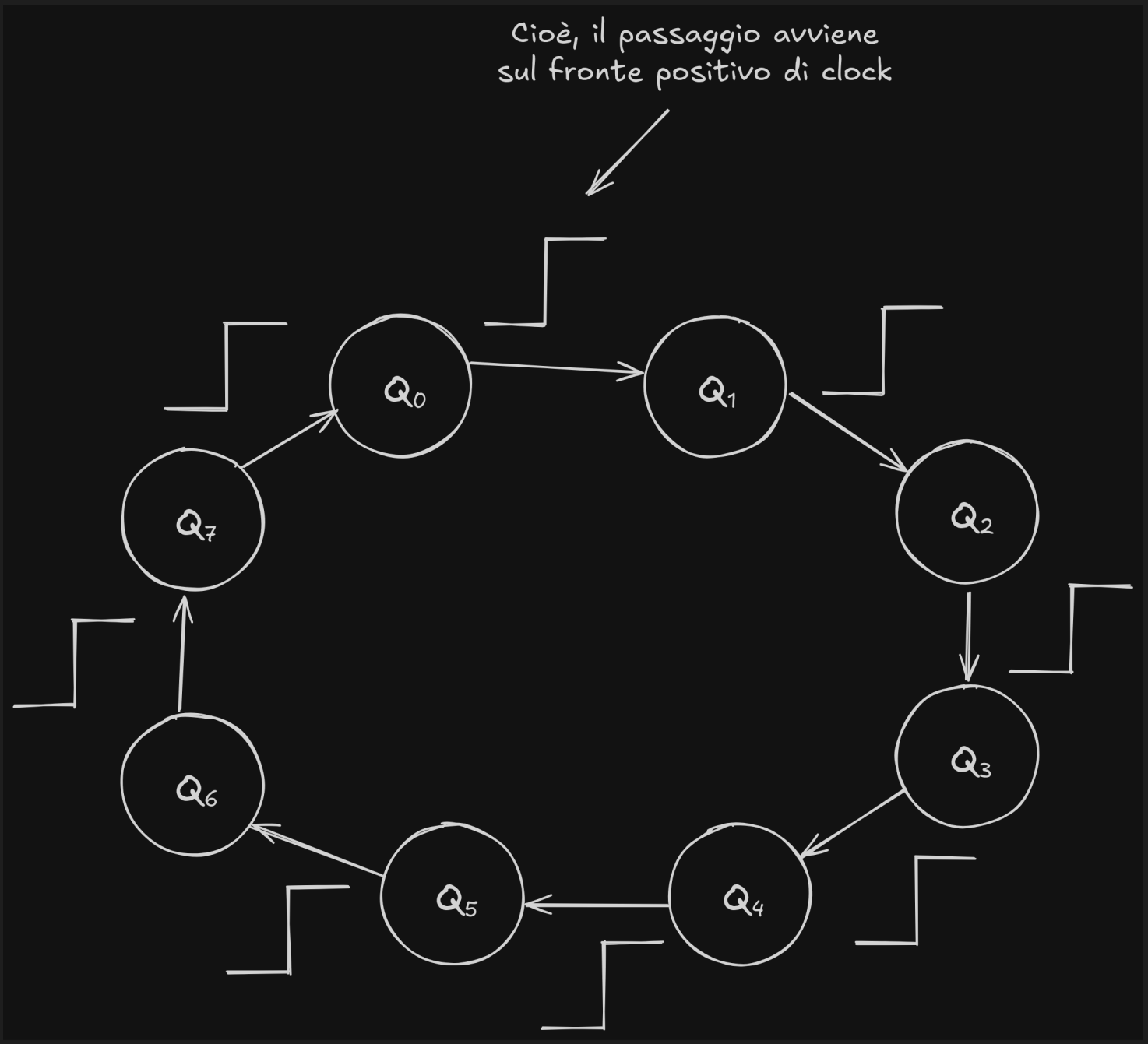
Sintesi di un contatore sincrono modulo 8 (up-counter)

Descrizione verbale del circuito

Vogliamo sintetizzare un contatore modulo 8, cioè un contatore che conta i valori da 0 a 7 in binario, quindi da `000` a `111` (visto che i numeri vengono incrementati, questo tipo di counter è detto *up-counter*). Per rappresentare questo range di valori mi servono 3 bit (ed infatti, il contatore modulo 8 è un contatore binario modulo 2^3). Inoltre, il valore deve essere incrementato dopo ogni fronte di clock positivo.

Automa a stati finiti

Possiamo rappresentare questo comportamento con un automa ad 8 stati:

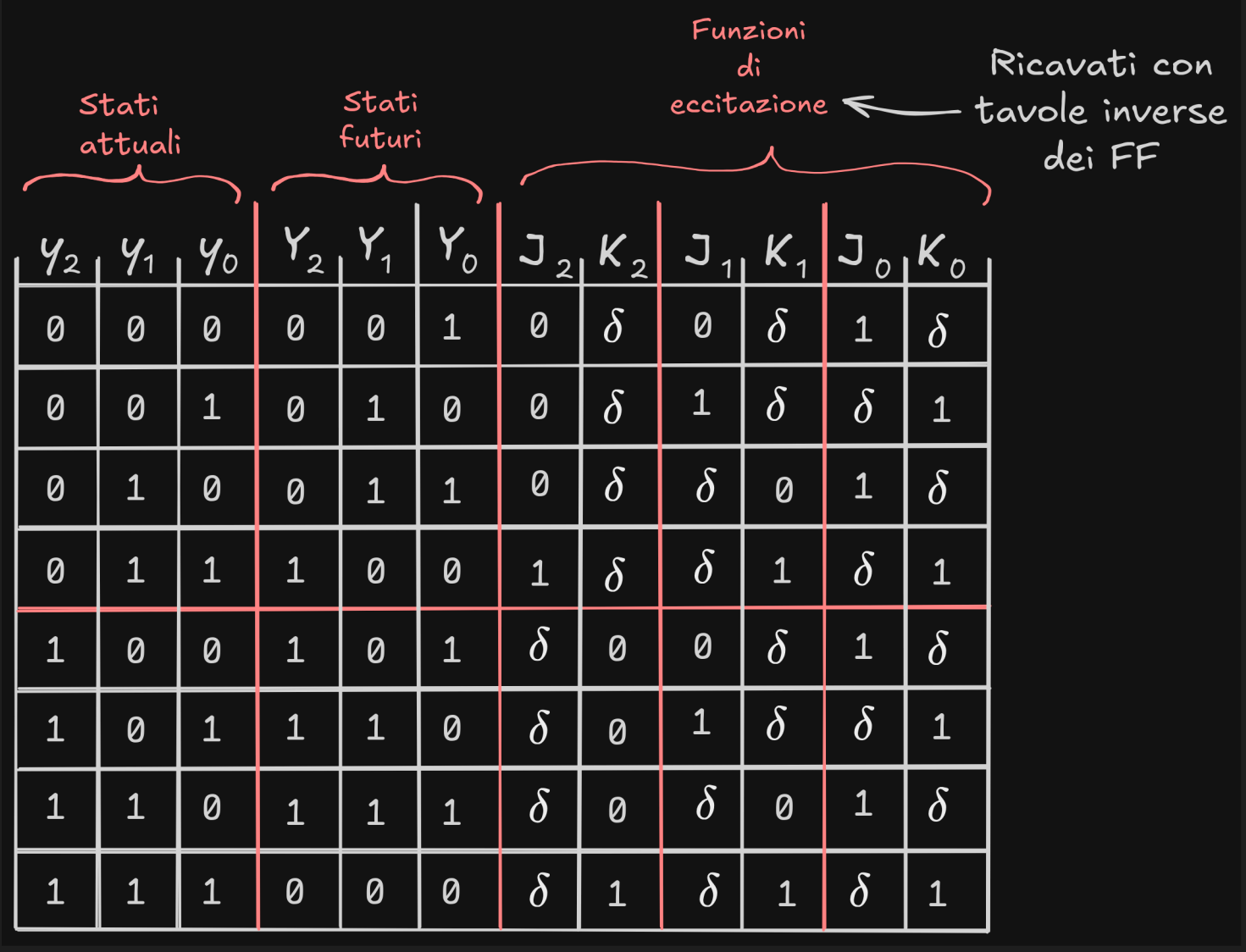


Una volta arrivati nello stato Q_7, avremmo memorizzato il valore 7, ed il clock successivo farà ricominciare il contatore da 0.

Tavola stati futuri

Per rappresentare 8 stati servono 3 bit, e cioè 3 Flip Flop: per implementare i contatori, la scelta migliore sono i Flip Flop JK pervia della loro capacità di complementazione, che vedremo sarà molto utile per passare da un numero all'altro in binario.

Poiché siamo nel processo di sintesi, torna utile la tavola inversa del FF JK.



Y_2	Y_1	Y_0	Y_2	Y_1	Y_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	δ	0	δ	1	δ
0	0	1	0	1	0	0	δ	1	δ	δ	1
0	1	0	0	1	1	0	δ	δ	0	1	δ
0	1	1	1	0	0	1	δ	δ	1	δ	1
1	0	0	1	0	1	δ	0	0	δ	1	δ
1	0	1	1	1	0	δ	0	1	δ	δ	1
1	1	0	1	1	1	δ	0	δ	0	1	δ
1	1	1	0	0	0	δ	1	δ	1	δ	1

Espressioni booleane

Ricaviamoci le espressioni booleane minimizzate con le mappe di Karnaugh:

J_2

Y_2 \ Y_1Y_0	00	01	11	10
0	0	0	1	0
1	δ	δ	δ	δ

→ Y_1Y_0

$J_2 = Y_1Y_0$

J_1

Y_2 \ Y_1Y_0	00	01	11	10
0	0	1	δ	δ
1	0	1	δ	δ

→ Y_0

$J_1 = Y_0$

J_0

Y_2 \ Y_1Y_0	00	01	11	10
0	1	δ	δ	1
1	1	δ	δ	1

→ 1

$J_0 = 1$

K_2

Y_2 \ Y_1Y_0	00	01	11	10
0	δ	δ	δ	δ
1	0	0	1	0

→ Y_1Y_0

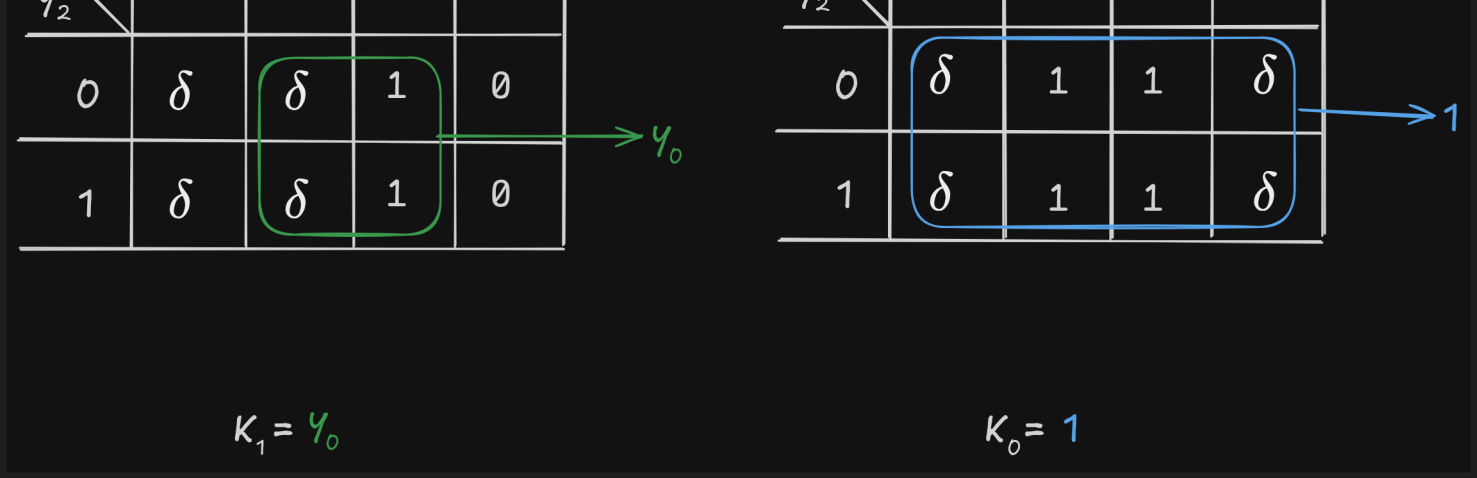
$K_2 = Y_1Y_0$

K_1

Y_2 \ Y_1Y_0	00	01	11	10

K_0

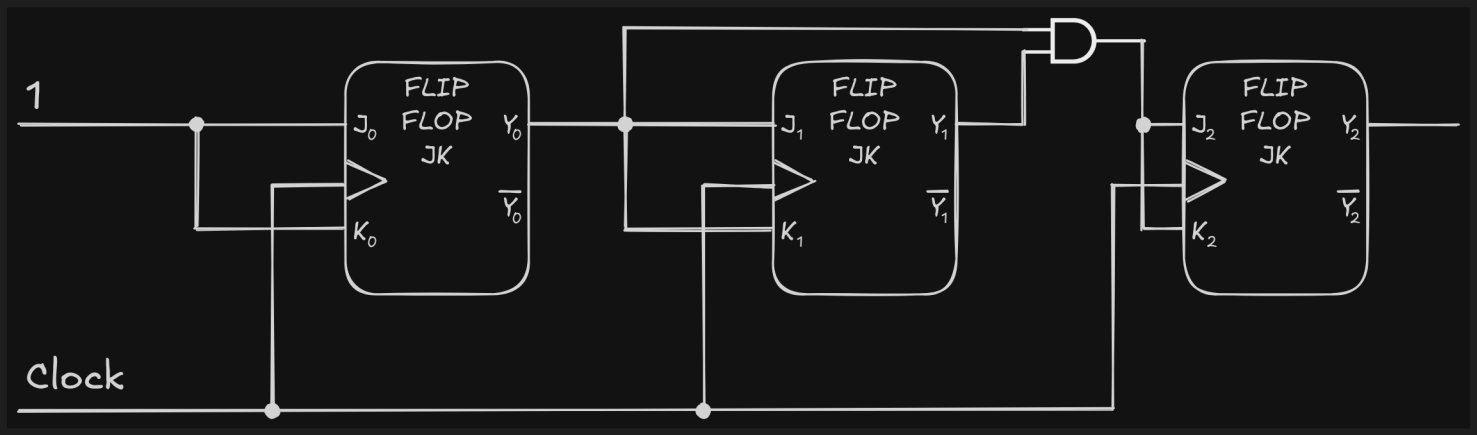
Y_2 \ Y_1Y_0	00	01	11	10



Risulta dunque che

$$
\begin{aligned}
J_0 &= K_0 = 1 \\
J_1 &= K_1 = y_0 \\
J_2 &= K_2 = y_1 y_0
\end{aligned}
$$

Diagramma Logico



In generale, un contatore modulo 2^n è composto da n Flip Flop, dove gli input $J_{i+1} = K_{i+1}$ sono l'AND tra Y_i e $J_i = K_i$.
Quindi il primo Flip Flop prende in input $J = 1$ e $K = 1$ e restituisce Y_0, il secondo Flip Flop ha come input l'AND $1Y_0$ e come output Y_1, il terzo Flip Flop ha come input l'AND $J_1 Y_1$ ed ha come output Y_2 ecc...

Funzionamento

Guardando il diagramma logico, ci si può rendere conto che il funzionamento del contatore gira intorno alla funzione di complementazione del Flip Flop JK: i Flip Flop sono collegati in cascata e sincronizzati dal clock, quindi ad ogni colpo di clock il bit meno significativo è complementato, passando da `0` ad `1` e viceversa. I bit successivi dipendono dall' AND tra i bit precedenti.
Quindi il bit x diventerà `1` nel colpo di clock dopo quello in cui tutti i bit meno significativi di x avranno assunto il valore `1`.

Per esempio, vediamo passo passo come contiamo da `000` a `101`:

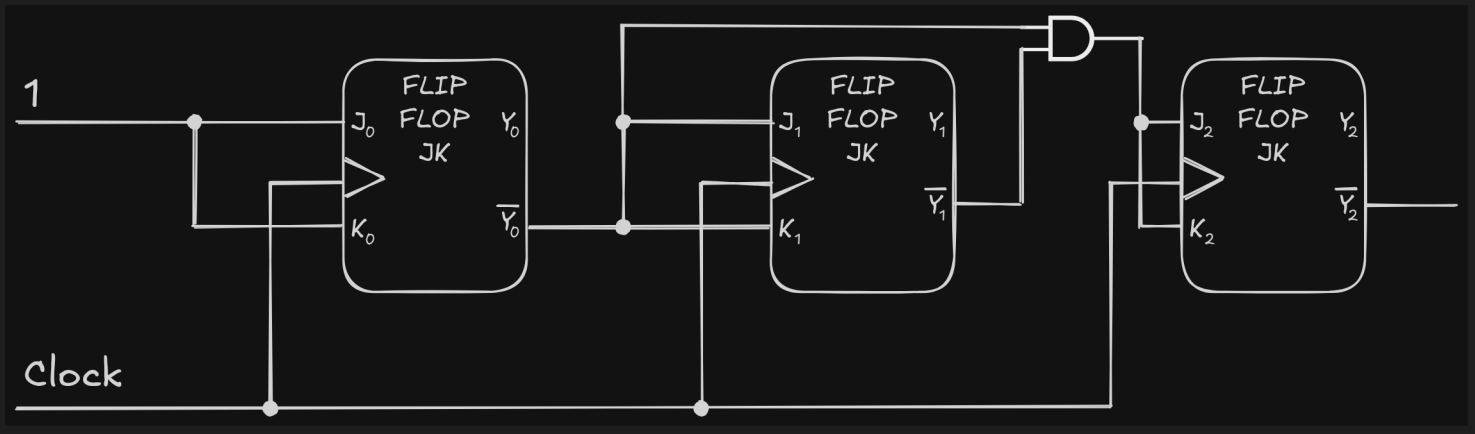
- All'inizio tutti i bit sono `0`, quindi Y_2= `0`, Y_1= `0`, Y_0= `0`
- Dopo il primo impulso di clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo in output `1`
  - Il secondo Flip Flop riceve l'Y_0 del precedente impulso di clock (cioè `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `0`
  - Il terzo Flip Flop riceve l'AND tra l'Y_1 e l'Y_0 del precedente impulso di clock (cioè rispettivamente `0` e `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `0`
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `0`, Y_0= `1`
- Dopo il secondo impulso di clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo in output `0`
  - Il secondo Flip Flop riceve l'Y_0 del precedente impulso di clock (cioè `1`), quindi J= `1` e K= `1`, ed esegue complementazione, restituendo in output `1`
  - Il terzo Flip Flop riceve l'AND tra l'Y_1 e l'Y_0 del precedente impulso di clock (cioè rispettivamente `0` e `1`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `0`
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `1`, Y_0= `0`
- Dopo il terzo impulso di clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo in output `1`
  - Il secondo Flip Flop riceve l'Y_0 del precedente impulso di clock (cioè `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `1`
  - Il terzo Flip Flop riceve l'AND tra l'Y_1 e l'Y_0 del precedente impulso di clock (cioè rispettivamente `1` e `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `0`
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `1`, Y_0= `1`
- Dopo il quarto impulso di clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo in output `0`
  - Il secondo Flip Flop riceve l'Y_0 del precedente impulso di clock (cioè `1`), quindi J= `1` e K= `1`, ed esegue complementazione, restituendo in output `0`
  - Il terzo Flip Flop riceve l'AND tra l'Y_1 e l'Y_0 del precedente impulso di clock (cioè rispettivamente `1` e `1`), quindi J= `1` e K= `1`, ed esegue complementazione, restituendo in output `1`
  - Quindi l'output del contatore è diventato Y_2= `1`, Y_1= `0`, Y_0= `0`
- Dopo il quinto impulso di clock:

- Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo in output `1`
- Il secondo Flip Flop riceve l'Y_0 del precedente impulso di clock (cioè `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `0`
- Il terzo Flip Flop riceve l'AND tra l'Y_1 e l'Y_0 del precedente impulso di clock (cioè rispettivamente `0` e `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo in output `1`
- Quindi l'output del contatore è diventato Y_2= `1`, Y_1= `0`, Y_0= `1`

Down counter

Il down counter esegue l'operazione inversa dell'up-counter: conta i numeri che vanno da $2^n - 1$ a 0, dove n è il numero di Flip Flop. Il nome deriva dal fatto che i numeri vengono decrementati.

Il funzionamento è analogo a quello dell'up-counter, con i numeri che vanno letti dalle uscite positive Y, ma invece di collegare l'output positivo Y del Flip Flop all'input del successivo, colleghiamo l'output negato $\overline{Y}$.



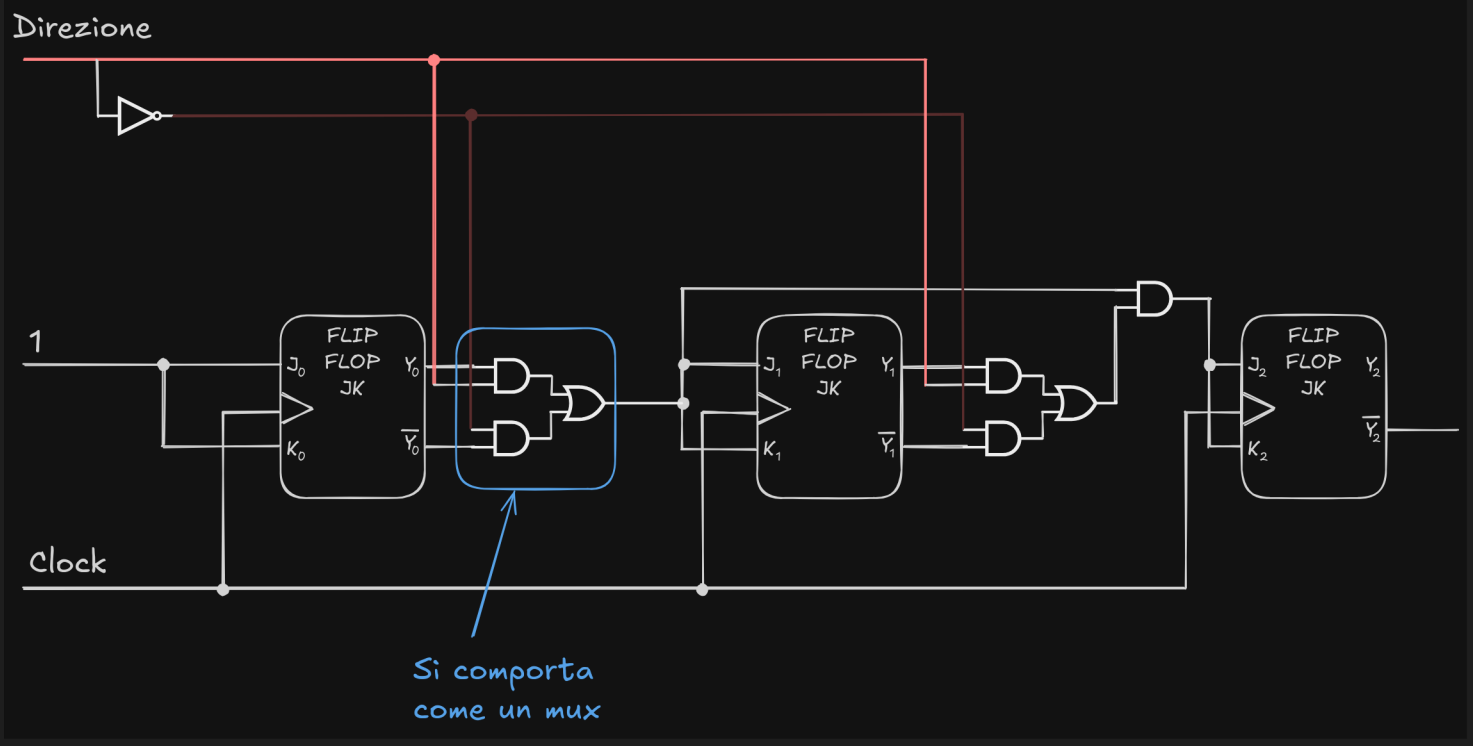
Per esempio, vediamo passo passo come contiamo da `011` a `000`:

- All'inizio i bit sono Y_2= `0`, Y_1= `1`, Y_0= `1`
- Dopo il primo impulso di clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo `0` sull'output Y_0 ed `1` sull'output $\overline{Y_0}$
  - Il secondo Flip Flop riceve l'$\overline{Y_0}$ del precedente impulso di clock (cioè `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo `1` sull'output Y_1 e `0` sull'output $\overline{Y_1}$
  - Il terzo Flip Flop riceve l'AND tra l'$\overline{Y_1}$ e l'$\overline{Y_0}$ del precedente impulso di clock (cioè rispettivamente `0` e `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo `0` sull'output Y_2 ed `1` sull'output $\overline{Y_2}$
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `1`, Y_0= `0`
- Dopo il secondo impulso di clock:

    - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo `1` sull'output Y_0 e `0` sull'output $\overline{Y_0}$
    - Il secondo Flip Flop riceve l'$\overline{Y_0}$ del precedente impulso di clock (cioè `1`), quindi J= `1` e K= `1`, ed esegue complementazione, restituendo `0` sull'output Y_1 ed `1` sull'output $\overline{Y_1}$
    - Il terzo Flip Flop riceve l'AND tra l'$\overline{Y_1}$ e l'$\overline{Y_0}$ del precedente impulso di clock (cioè rispettivamente `0` e `1`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo `0` sull'output Y_2 ed `1` sull'output $\overline{Y_2}$
    - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `0`, Y_0= `1`
- Dopo il terzo impulso di clock:
    - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, e viene complementato, restituendo `0` sull'output Y_0 e `1` sull'output $\overline{Y_0}$
    - Il secondo Flip Flop riceve l'$\overline{Y_0}$ del precedente impulso di clock (cioè `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo `0` sull'output Y_1 ed `1` sull'output $\overline{Y_1}$
    - Il terzo Flip Flop riceve l'AND tra l'$\overline{Y_1}$ e l'$\overline{Y_0}$ del precedente impulso di clock (cioè rispettivamente `1` e `0`), quindi J= `0` e K= `0`, ed esegue memorizzazione, restituendo `0` sull'output Y_2 ed `1` sull'output $\overline{Y_2}$
    - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `0`, Y_0= `0`

Up-down counter

L' up-down counter è un contatore che permette di essere utilizzato sia come up-counter che come down-counter. Vengono presi entrambi gli output del Flip Flop, Y e $\overline{Y}$, si fanno passare per un mux che in base ad un segnale di controllo decide quale dei due trasmettere agli ingressi del Flip Flop successivo, contando in avanti o indietro di conseguenza.



Se il segnale di Direzione è `1`, il circuito si comporta come un contatore up counter, se è `0` invece si comporta come un contatore down counter.

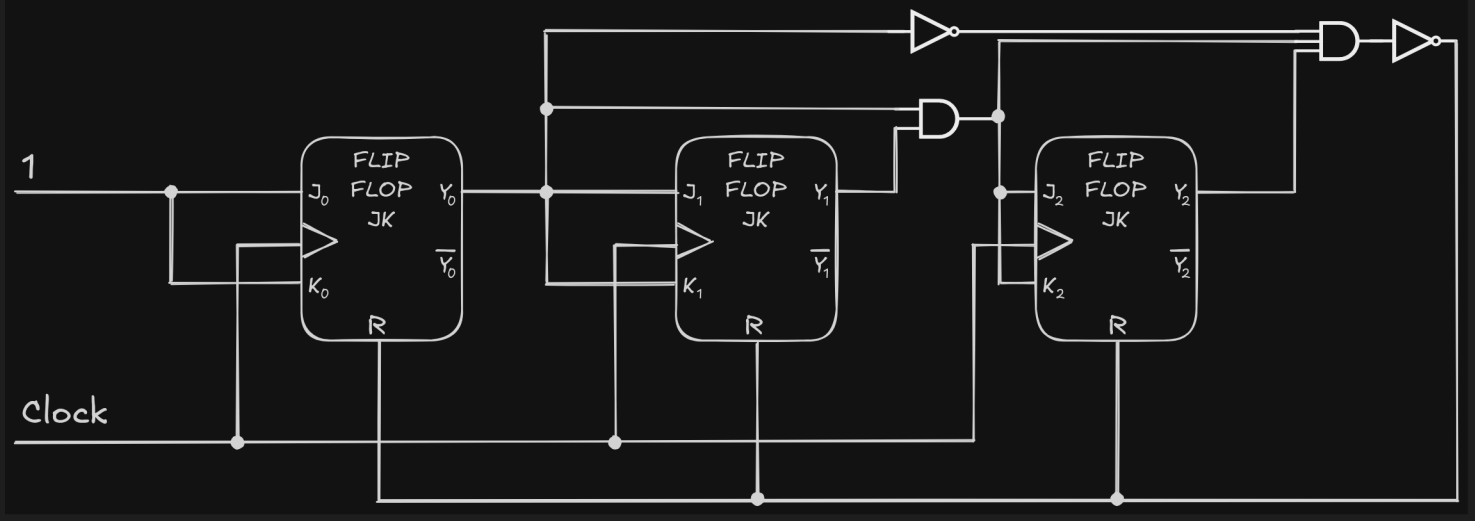
Contatore modulo non potenza di 2

Abbiamo detto che i contatori modulo 2^n dove n è il numero di Flip Flop sono standard e chiamati contatori binari. Tuttavia, potremmo avere necessità di un contatore non binario, per esempio un contatore modulo 6. Per implementarlo, abbiamo due possibilità:

1) Sintetizzare un progetto ad-hoc di un contatore che permette di contare fino al numero che ho scelto, quindi nel nostro caso da 0 a 5. È però sconsigliato perché se poi decidiamo di voler contare fino ad un numero diverso, dovremmo rifare il progetto da capo.
2) Usiamo i contatori pre-selezionabili, che è la scelta consigliata.

Per contatori pre-selezionabili, intendiamo dei normali contatori dove però vengono usati i segnali asincroni di preset e clear per intercettare la codifica del numero del modulo ed azzerare o portare ad un altro valore il contatore.

Per esempio, nel nostro caso vogliamo creare un contatore modulo 6 e che una volta raggiunto il valore massimo `101`, incrementare di nuovo il contatore riporti al valore `000`. Riprendiamo dunque un up-counter modulo 8 (perché per rappresentare i numeri da 0 a 5 servono 3 bit, e $2^3 = 8$), ma intercettiamo le uscite dei Flip Flop ed appena rilevata la combinazione `110` portiamo il segnale di clear di tutti i Flip Flop a `0` per un periodo brevissimo.



Volendo, invece di riportare il contatore a `000` una volta intercettato il valore `110`, potrei usare il segnale di preset per portarlo ad un'altra combinazione di bit.

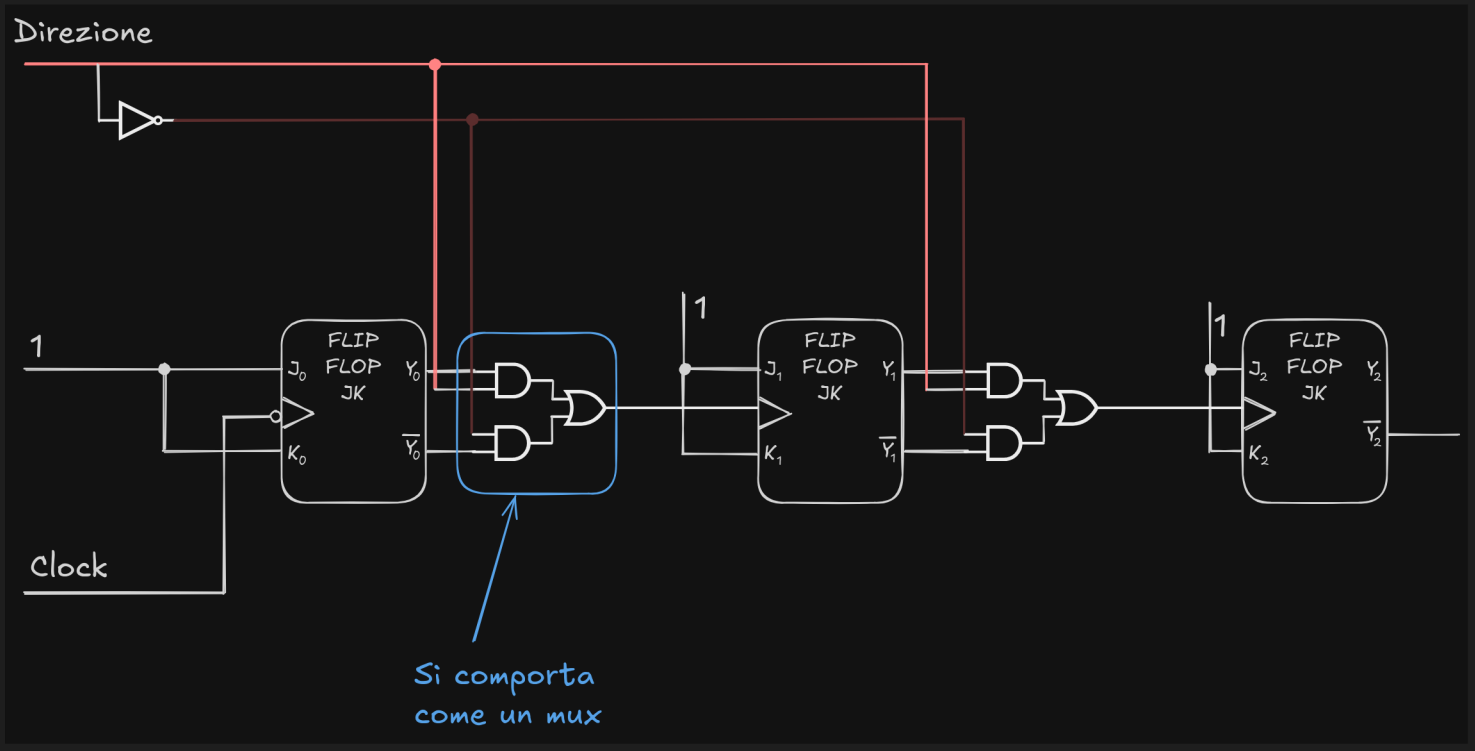
Contatore asincrono

Finora abbiamo visto solo contatori sincroni. I contatori asincroni sono dei contatori dove i Flip Flop svolgono le transizioni in base ai segnali in cascata che ricevono, invece di eseguirle contemporaneamente in sincronia al segnale di clock.

In questo tipo di contatore,

- Gli ingressi di tutti i Flip Flop è fisso ad `1`
- Il primo Flip Flop usa il clock come segnale di controllo
- I Flip Flop successivi usano l'output del Flip Flop precedente come segnale di controllo.

Di seguito, diamo un'occhiata ad un contatore asincrono modulo 8 up-down, con il primo Flip Flop sensibile al fronte d'onda negativo.



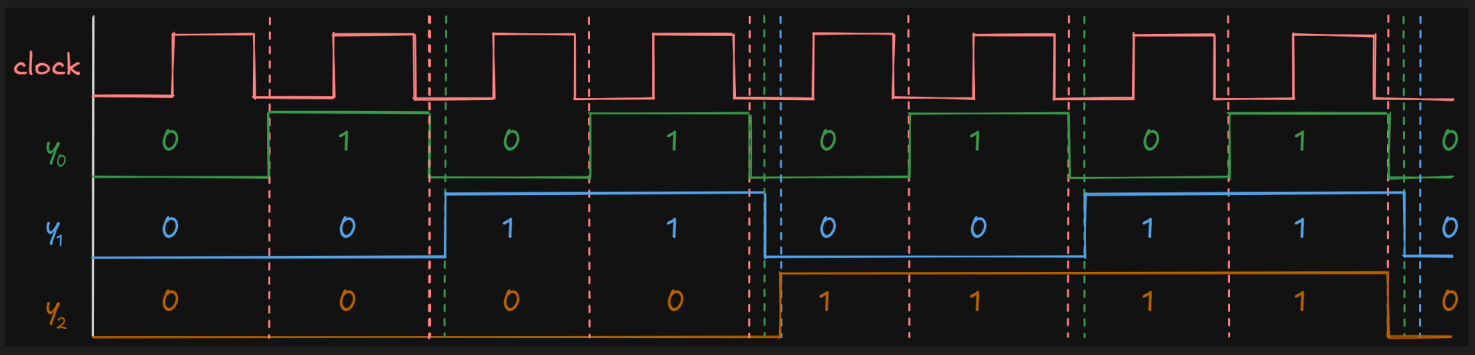
Il primo Flip Flop, che rappresenta il bit meno significativo, è collegato al clock ed è sensibile al fronte d'onda di discesa, cioè, esegue la transizione e si complementa ogni volta che il clock passa da `1` a `0`. I successivi invece, cercano di complementarsi perché hanno ambi gli ingressi ad `1`, ma sono abilitati alla modifica solo quando il segnale di controllo è `0` e cioè quando l'output del Flip Flop precedente sta passando da **`1`** a **`0`**.

Considerando il segnale direzione sempre abilitato ad `1`, vediamo come si va da `000` ad `100`.

- All'inizio i bit sono Y_2= `0`, Y_1= `0`, Y_0= `0`
- Al primo fronte di discesa del clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, ed avviene la complementazione, dunque Y_0= `1`.
  - Il secondo Flip Flop è disabilitato alla modifica, perché l'output Y_0 non è sul fronte di discesa, perché è andato da `0` ad `1`. Dunque l'output Y_1 rimane `0`.
  - Il terzo Flip Flop è disabilitato alla modifica, perché l'output Y_1 non è cambiato, quindi non c'è nessun fronte di discesa. Dunque l'output Y_2 rimane `0`
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `0`, Y_0= `1`
- Al secondo fronte di discesa del clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, ed avviene la complementazione, dunque Y_0= `0`.
  - Il secondo Flip Flop si abilita alla modifica, perché l'output Y_0 è sul fronte di discesa, cioè è andato da `1` a `0`. Dunque l'output Y_1 si complementa e diventa `1`.
  - Il terzo Flip Flop è disabilitato alla modifica, perché l'output Y_1 non è sul fronte di discesa, perché è andato da `0` ad `1`. Dunque l'output Y_2 rimane `0`.
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `1`, Y_0= `0`
- Al terzo fronte di discesa del clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, ed avviene la complementazione, dunque Y_0= `1`.
  - Il secondo Flip Flop è disabilitato alla modifica, perché l'output Y_0 non è sul fronte di discesa, perché è andato da `0` ad `1`. Dunque l'output Y_1 rimane `1`.
  - Il terzo Flip Flop è disabilitato alla modifica, perché l'output Y_1 non è cambiato, quindi non c'è nessun fronte di discesa. Dunque l'output Y_2 rimane `0`
  - Quindi l'output del contatore è diventato Y_2= `0`, Y_1= `1`, Y_0= `1`
- Al quarto fronte di discesa del clock:
  - Il primo Flip Flop riceve `1`, quindi J= `1` e K= `1`, ed avviene la complementazione, dunque Y_0= `0`.
  - Il secondo Flip Flop si abilita alla modifica, perché l'output Y_0 è sul fronte di discesa, cioè è andato da `1` a `0`. Dunque l'output Y_1 si complementa e diventa

`0`.

- Il terzo Flip Flop si abilita alla modifica, perché l'output Y_1 è sul fronte di discesa, cioè è andato da `1` a `0`. Dunque l'output Y_2 si complementa e diventa `1`.
- Quindi l'output del contatore è diventato Y_2= `1`, Y_1= `0`, Y_0= `0`



- La linea tratteggiata rossa rappresenta il punto in cui Y_0 reagisce al fronte di discesa del clock
- La linea tratteggiata verde rappresenta il punto in cui Y_1 reagisce al fronte di discesa di Y_0
- La linea tratteggiata blu rappresenta il punto in cui Y_2 reagisce al fronte di discesa di Y_1
- Le linee tratteggiate non corrispondono esattamente con i fronti di discesa perché stiamo considerando un piccolo ritardo dovuto al tempo di attraversamento delle porte e della stabilizzazione del circuito

Detto questo, i contatori asincroni essendo tali possono risultare poco affidabili, ed i ritardi di commutazione possono accumularsi (anche se solitamente l'effetto prodotto è trascurabile). Dunque, se possibile è meglio utilizzare contatori sincroni.