

5. Circuiti sequenziali

Circuiti sequenziali

I circuiti sequenziali sono dei circuiti combinatori che utilizzano elementi di memorizzazione collegati ad anello, cioè la cui uscita è usata come ingresso del circuito che ha prodotto l'uscita stessa (*Linea di feedback*).

Gli elementi di memoria sono circuiti in grado di immagazzinare informazioni binarie e definiscono, istante per istante, lo *stato* del circuito sequenziale, da cui dipende il comportamento del dispositivo: quindi vengono presi in considerazione

- I valori inviati negli ingressi del circuito dall'ambiente esterno
- Lo stato in cui si trova il circuito quando riceve gli input

E basandosi su queste informazioni, vengono prodotti:

- I valori di uscita del circuito
- Lo stato futuro, cioè lo stato in cui il circuito si troverà a transizione avvenuta.



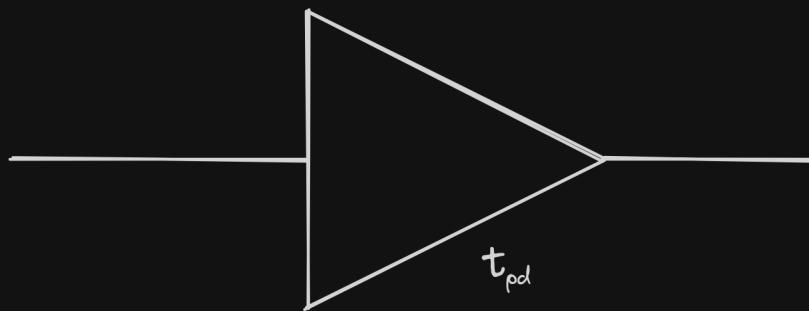
I circuiti sequenziali sono classificati in due categorie in base agli istanti in cui gli ingressi sono osservati e dalla evoluzione degli stati interni:

- 1) Il comportamento di un circuito sequenziale sincrono può essere definito conoscendo lo stato dei segnali in istanti discreti di tempo^[1].
- 2) Il comportamento di un circuito sequenziale asincrono dipende dal valore degli ingressi ad ogni istante continuo di tempo^[2] e dal modo in cui si evolvono nel tempo.

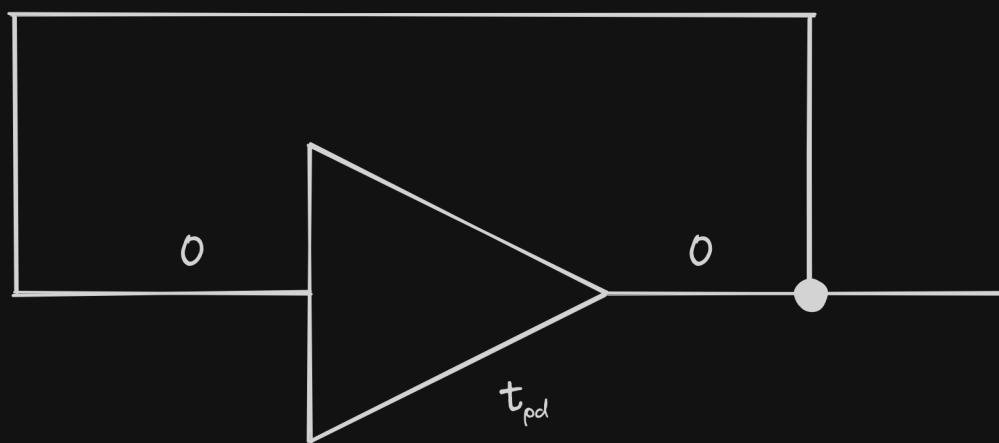
Elemento di memorizzazione base

Diamo un'occhiata a come potrebbe funzionare un elemento di memoria:

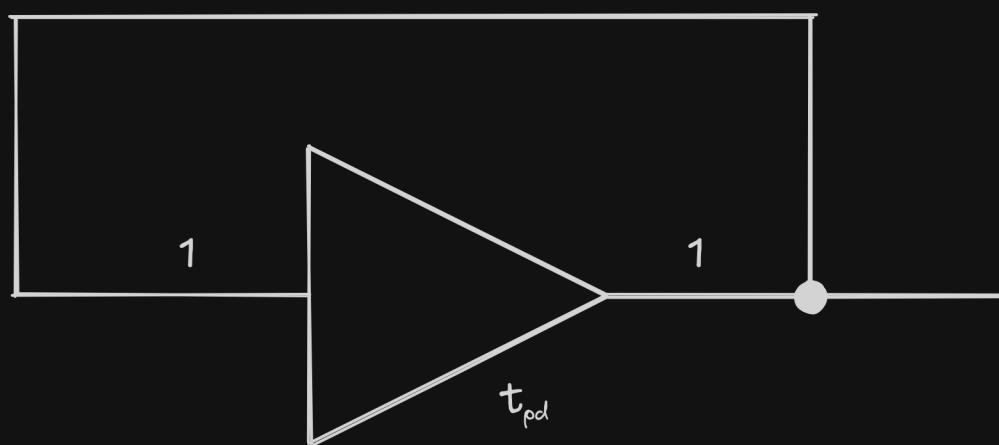
(a)



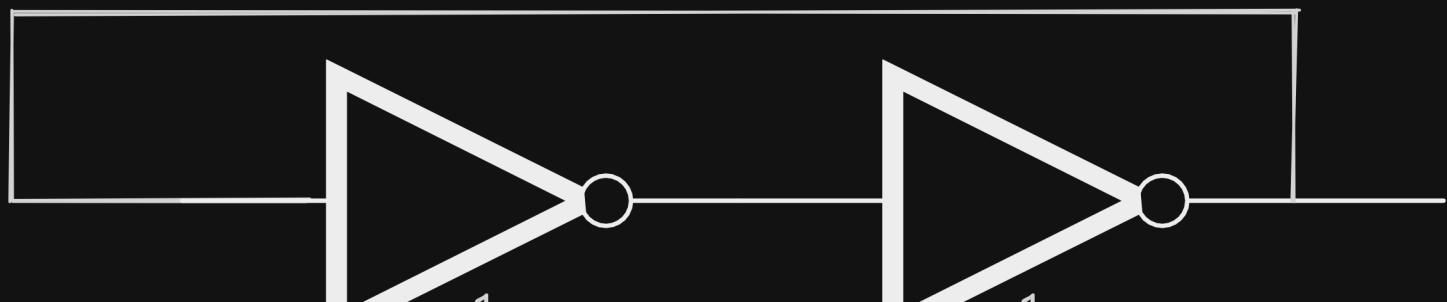
(b)



(c)



(d)



$$\frac{1}{2} t_{pd}$$

$$\frac{1}{2} t_{pd}$$

Nella figura *a*, c'è un [buffer](#) il cui segnale si propaga con un ritardo t_{pd} , quindi quando l'informazione entrerà nel buffer al tempo t , uscirà così com'è all'istante $t + t_{pd}$.

Se creiamo una linea di feedback per collegare l'uscita del buffer al proprio ingresso, come nelle figure *b* e *c*, assumendo che il valore dell'ingresso del buffer non sia stato modificato durante il ritardo di propagazione, allora all'istante $t + t_{pd}$ viene mandato in output [0](#) nel caso della figura *b* ed [1](#) nel caso della figura *c*. Questo valore viene poi reindirizzato all'ingresso del buffer e l'uscita assumerà di nuovo lo stesso valore al istante $t + 2t_{pd}$ e così via. In questo modo, siamo riusciti a memorizzare per un tempo indefinito il valore [0](#) o [1](#).

La figura *d* realizza il circuito delle figure precedenti attraverso due porte NOT: affinché un circuito con linea di feedback possa essere utilizzato per memorizzare informazioni, lungo l'anello di retroazione non devono presentarsi inversioni del segnale (come in questo caso, dove il segnale passando per due porte NOT viene prima invertito e poi riportato al valore originale).

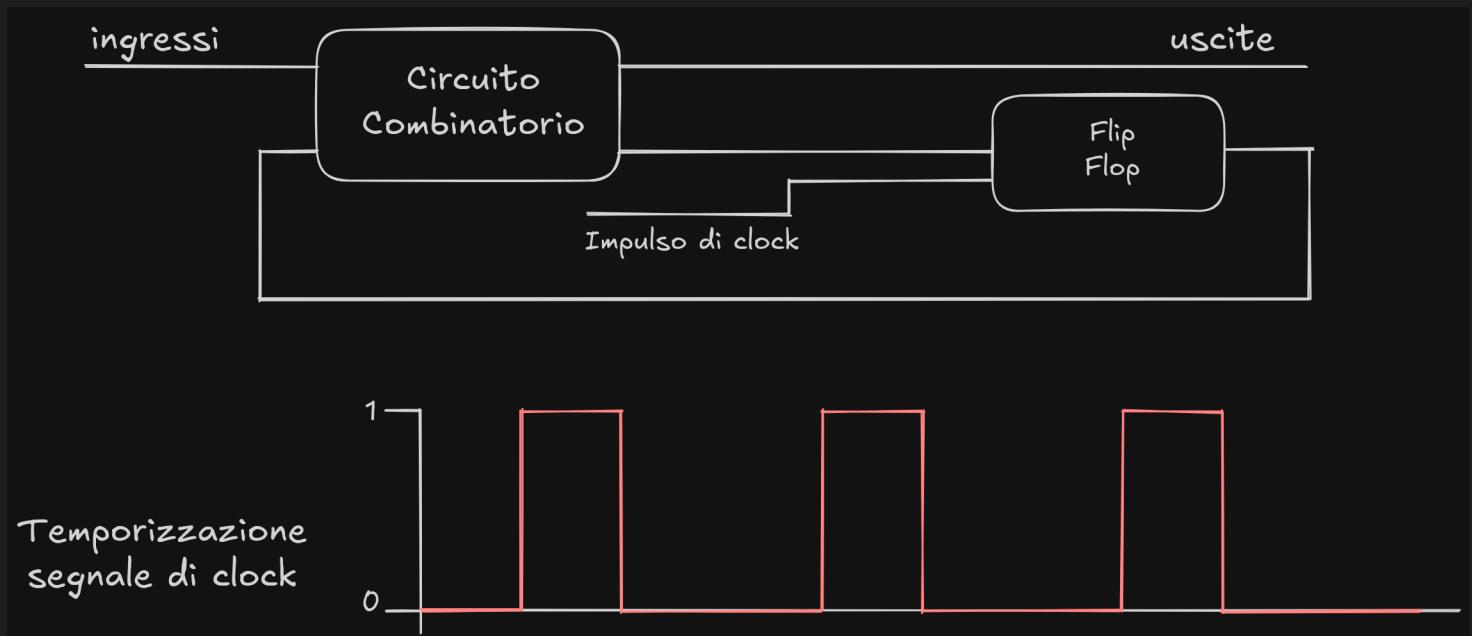
Il problema principale di questi elementi di memorizzazione è che non è possibile modificare l'informazione in essi contenuti senza modificarne la struttura. Vedremo poi che i circuiti di memoria asincroni (detti [latch](#), che approfondiremo tra poco) usano porte NOR e NAND invece di invertitori per risolvere questo problema.

In generale, progettare circuiti sequenziali asincroni risulta più complesso, visto che il loro comportamento dipende da fattori come il ritardo di propagazione e l'istante in cui gli ingressi cambiano, e dunque vengono solitamente preferiti i circuiti sequenziali sincroni (i quali però usano come blocchi di base i [flip flop](#), approfonditi dopo, che sono realizzati tramite l'uso dei latch, cioè circuiti sequenziali asincroni).

In un circuito sequenziale sincrono infatti, i segnali hanno effetto sugli elementi di memoria in istanti discreti di tempo, dettati da un segnale di sincronizzazione emesso ad intervalli di tempo costante (il segnale di clock) generato da un apposito circuito e distribuito a tutti i componenti del sistema in modo da effettuare le transizioni sulle uscite in sincronia con gli impulsi del segnale di clock. Quindi, gli elementi di memorizzazione prendono in ingresso anche il segnale di clock (oltre ai loro segnali che specificano la transizione di stato) per sapere *quando* cambiare stato, e quindi funzionano correttamente anche in presenza di ritardi tra i vari segnali del circuito.

Come abbiamo detto prima, questi elementi di memoria per i circuiti sincroni sono detti [flip flop](#) (*multivibratori bistabili*), in grado di memorizzare un bit ed hanno caratteristiche di temporizzazione (cioè, fanno uso del segnale di clock) che gli dicono quando cambiare stato. I flip flop prendono in ingresso le uscite del circuito combinatorio ed il segnale di clock, e li usano per generare le proprie uscite. In assenza del segnale di clock, il flip flop

rimane bloccato sullo stesso stato, che quindi diventa l'unico fattore a decidere qual'è l'uscita del flip flop. Infine, i flip flop in genere forniscono in uscita il valore in forma diretta, ed optionalmente, anche in forma complementare.



Adesso cerchiamo di capire il funzionamento dei latch, che stanno alla base della struttura dei flip flop

| Latch

Un elemento di memoria è in grado di mantenersi indefinitamente (purché alimentato) in uno stato binario finché un segnale di ingresso non ne causa il cambiamento di stato. I latch ed i flip flop si distinguono in base al numero di ingressi e dal modo in cui cambiano stato.

I latch sono elementi di memoria molto semplici, e vengono usati come componenti primitivi per la costruzione di dispositivi di memoria.

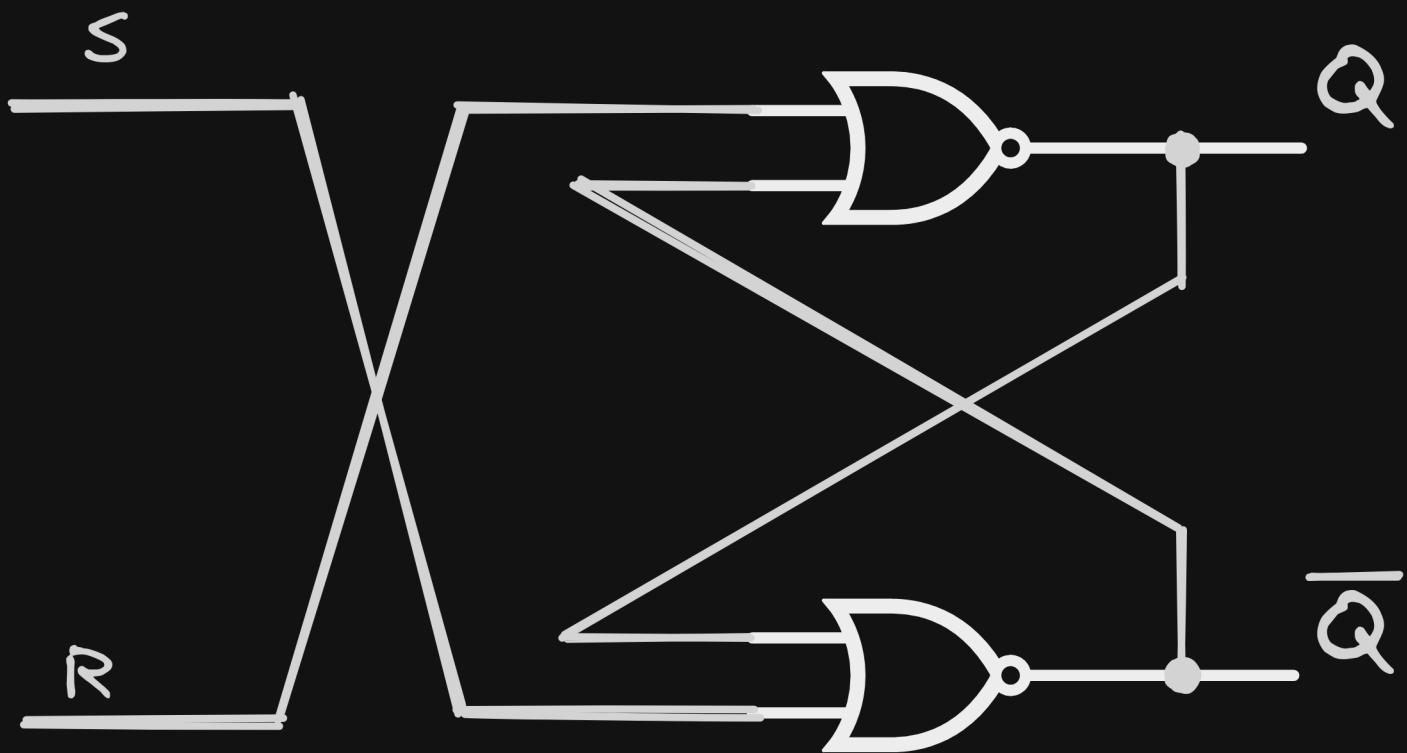
Si dividono in più tipi.

| Latch SR

Il Latch Set-Reset (SR) è un circuito composto da due porte NOR che prende due ingressi S ed R e da una uscita Q e la sua complementazione \bar{Q} .

Si ottiene connettendo due porte NOR tramite collegamenti intrecciati, che quindi hanno al primo ingresso un segnale esterno (S o R) ed al secondo ingresso l'uscita dell'altra porta NOR.

LATCH SR



Quando avviamo il circuito per la prima volta, in base alla velocità con il quale i segnali attraversano il circuito, uno tra Q e \bar{Q} sarà **1**, e l'altro sarà **0**. Per semplicità, ipotizziamo Q parta da **0** e quindi \bar{Q} parta da **1**, e proviamo ad applicare in ordine le combinazioni $S=1$ ed $R=0$, $S=0$ ed $R=1$, $S=0$ ed $R=0$ ed infine $S=1$ ed $R=1$

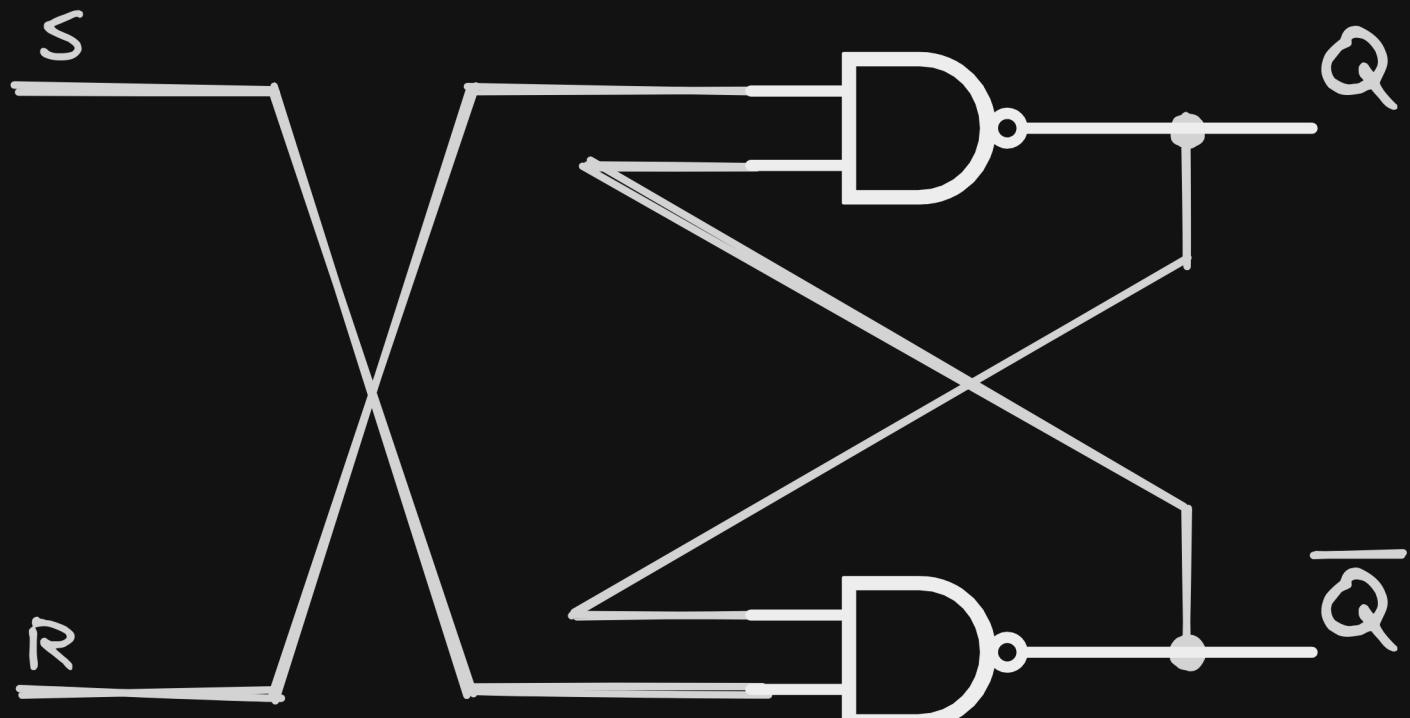
- Quando $S=1$ ed $R=0$, il latch è nello stato di *set*: il segnale S passa per la porta NOR in basso insieme al valore precedente di Q , cioè **0**, e dunque \bar{Q} diventa **1**. Adesso, il segnale R passa per la porta NOR in alto insieme al valore di \bar{Q} appena generato, propagando in uscita **1**, che diventa il valore di Q .
- Quando $S=0$ ed $R=1$, il latch è nello stato di *reset*: il segnale R passa per la porta NOR in alto insieme al valore precedente di \bar{Q} , cioè **0**, e dunque Q diventa **0**. Adesso, il segnale S passa per la porta NOR in basso insieme al valore di Q appena generato, propagando in uscita **1**, che diventa il valore di \bar{Q} .
- Quando $S=0$ ed $R=0$, il latch è nello stato di *memorizzazione*: il segnale S passa per la porta NOR in basso insieme al valore precedente di Q , cioè **0**, e dunque \bar{Q} rimane **1**. Adesso, il segnale R passa per la porta NOR in alto insieme al valore di \bar{Q} appena generato, propagando in uscita **0**, che diventa il valore di Q . Infatti, poiché $A + \bar{A} = A$, in questo stato i valori di uscita rimangono gli stessi impostati dallo stato precedente, cioè rimangono memorizzati finché non viene cambiato di nuovo stato.
- Quando $S=1$ ed $R=1$, il latch è in uno stato *indefinito e non consentito*: si rompe la condizione per cui \bar{Q} è sempre il complementare di Q , e quindi finché gli ingressi S ed R

R rimangono entrambi a 0, i valori di Q e \bar{Q} non sono prevedibili, e dunque *non bisognerebbe mai usare questa combinazione.*

S	R	Q	\bar{Q}	Stato
0	0	Precedente	Precedente	Memorizzazione
0	1	0	1	Reset
1	0	1	0	Set
1	1	δ	δ	Non consentito

Questo latch è implementabile anche con sole porte NAND, dove la tavola di verità è invertita. In questo caso è spesso rappresentato con la negazione sopra SR.

LATCH SR

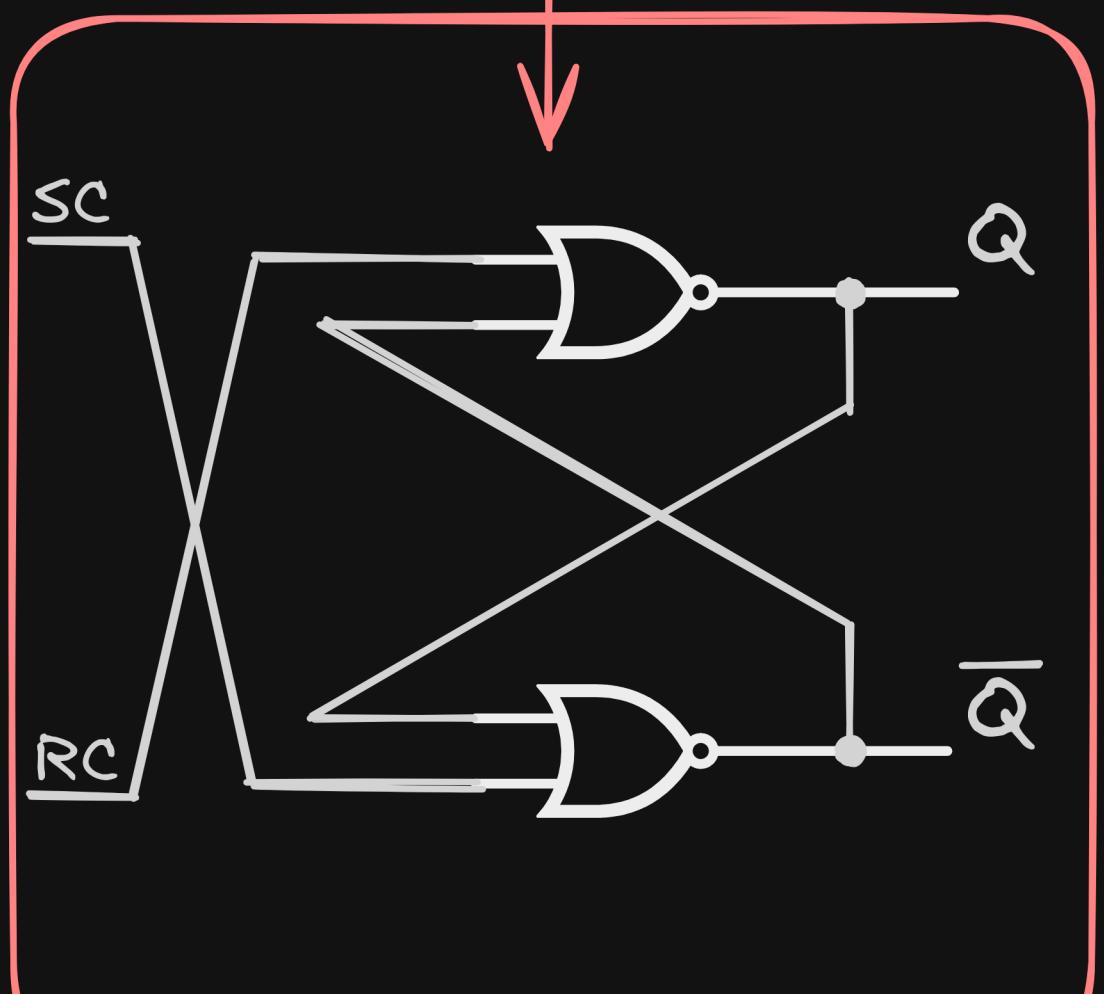
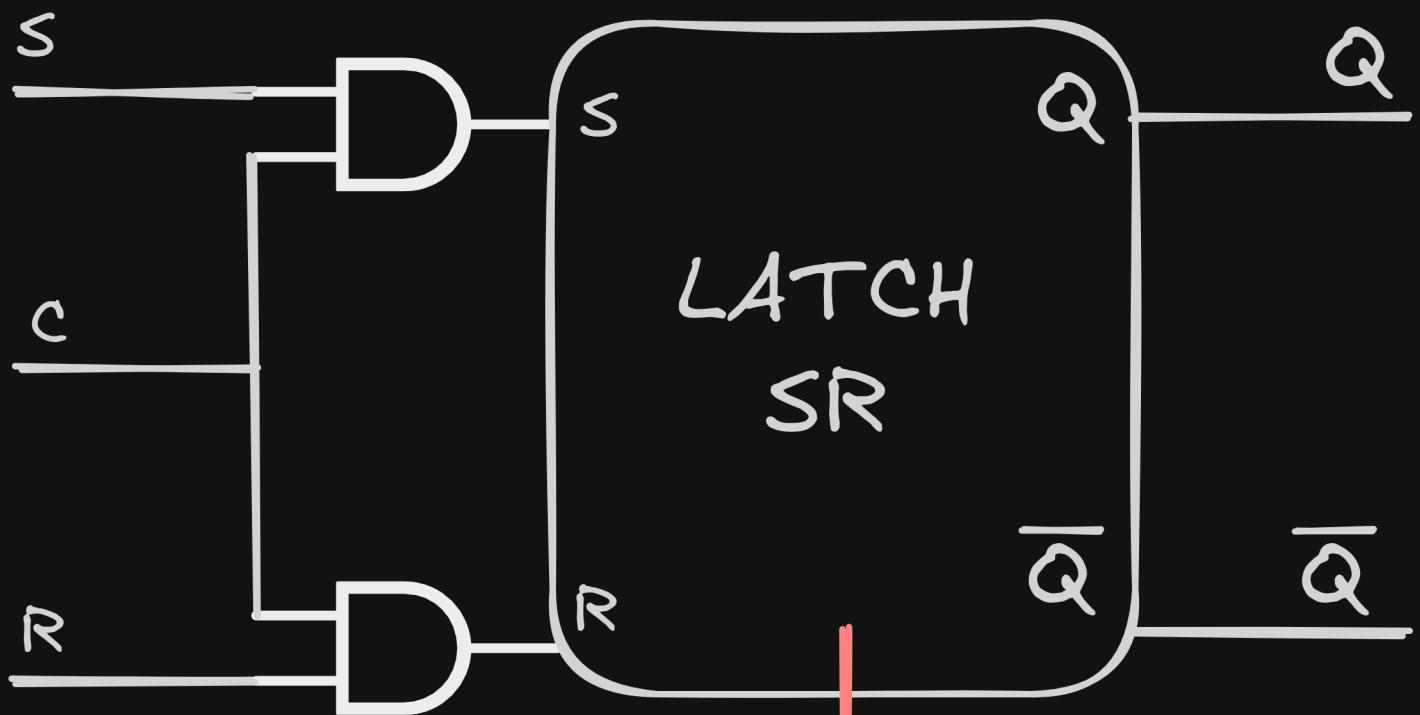


S	R	Q	\bar{Q}	Stato
0	0	δ	δ	Non consentito
0	1	1	0	Set
1	0	0	1	Reset
1	1	Precedente	Precedente	Memorizzazione

Il gated latch SR è un latch SR che prende in ingresso anche un segnale di controllo per abilitare o disabilitare le modifiche, utile per decidere se usare il valore memorizzato o no.

Semplicemente si pongono due porte AND davanti a S ed R , accoppiandole al segnale di controllo C . Quando C è 0 , l'uscita di queste porte NAND è obbligatoriamente 0 , indipendentemente dai valori di S ed R , e quindi viene selezionato lo stato di memorizzazione. Imponendo C a 1 , il latch viene normalmente pilotato da S ed R come un normale latch SR, e quindi rimane anche il fatto che non è consentito avere contemporaneamente S , R e C impostati ad 1 .

GATED LATCH SR

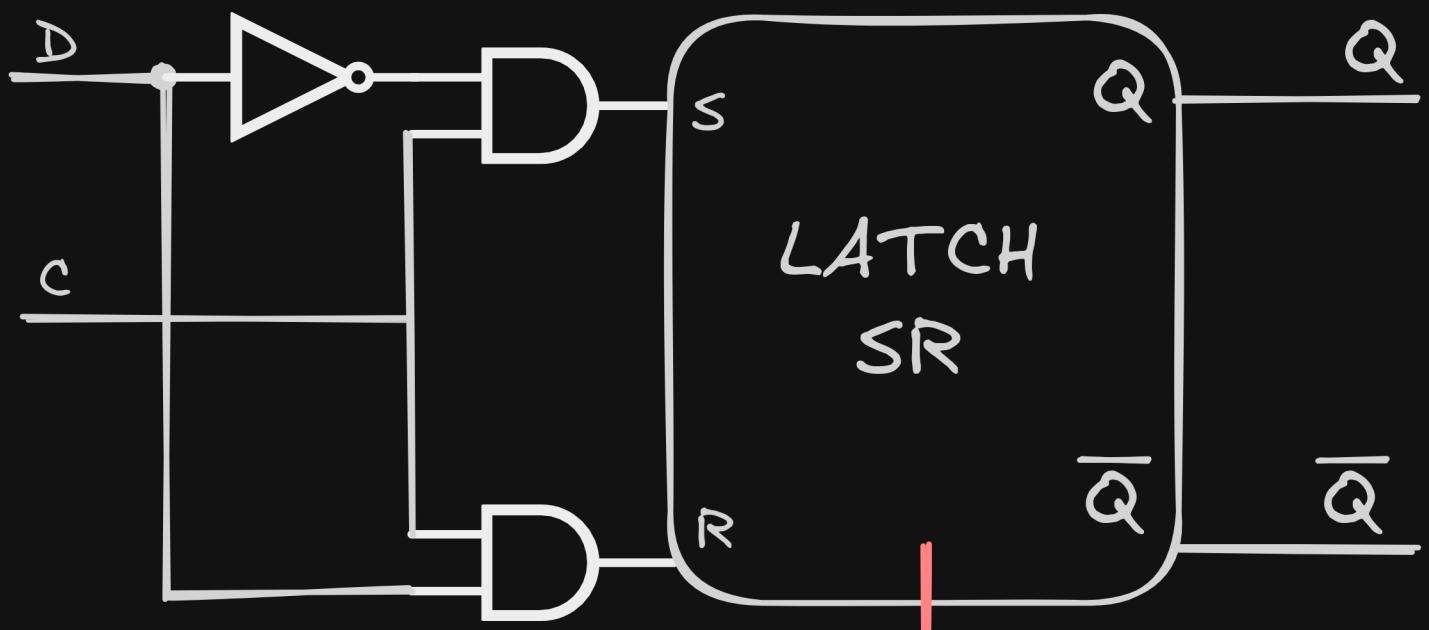


C	S	R	Q	\bar{Q}	Stato
0	δ	δ	Precedente	Precedente	Memorizzazione
1	0	0	Precedente	Precedente	Memorizzazione
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	δ	δ	Non consentito

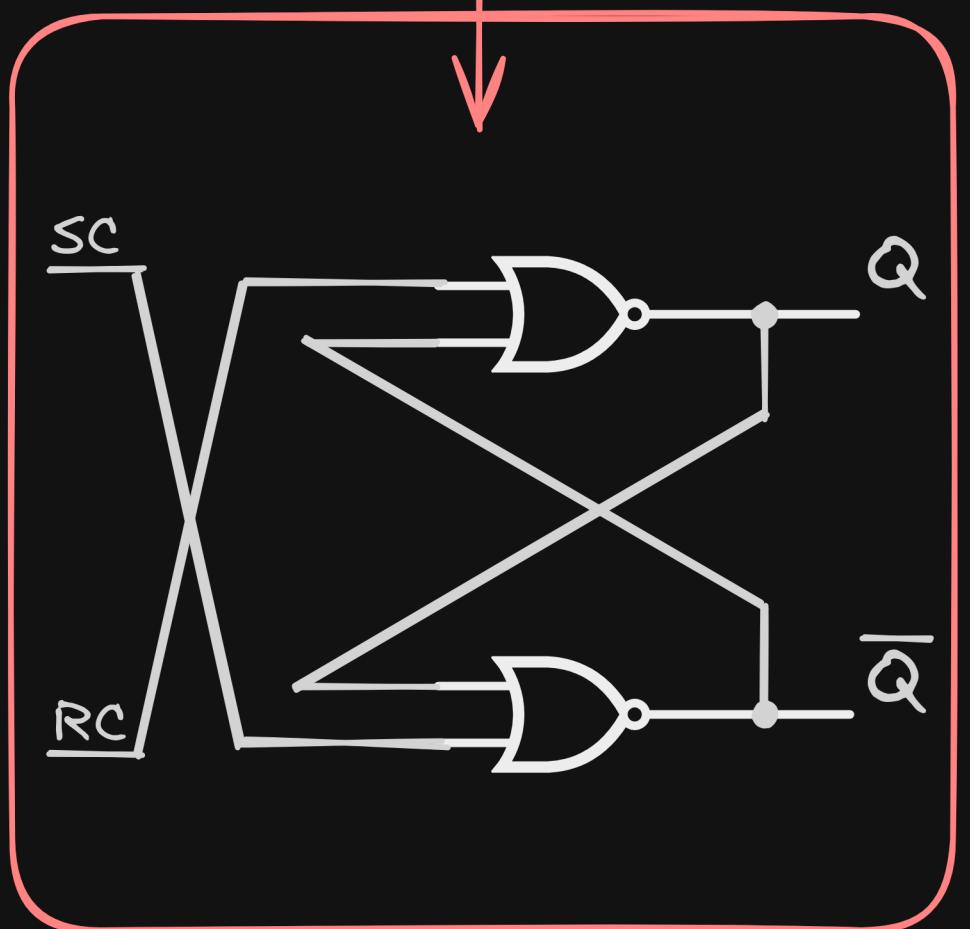
| Latch D

Un modo per eliminare lo stato indefinito nel latch SR, è evitando che gli ingressi S ed R siano **1** contemporaneamente. Ciò è possibile con il Latch Delay (D). Questo tipo di latch ha solo due ingressi, D (dati) e C (controllo), ed è essenzialmente un Gated Latch SR con D collegato all'AND per l'ingresso S , e \bar{D} collegato all'ingresso R : in questo modo, quando C è **0** le modifiche sono disabilitate, e il latch entra direttamente nello stato di memorizzazione, e per passare a set o reset basta portare D rispettivamente a **1** o **0** mentre C è 1. La combinazione non consentita diventa quindi irraggiungibile, non potendo avere D e \bar{D} uguali ad **1** contemporaneamente.

LATCH D



LATCH
SR



C	D	\bar{D}	Q	\bar{Q}	Stato
0	δ	δ	Precedente	Precedente	Memorizzazione
1	0	1	0	1	Reset
1	1	0	1	0	Set

Flip Flop

Lo stato di un latch interno ad un Flip Flop cambia quando avviene una variazione, anche momentanea, dell'ingresso di controllo: questa variazione fa scattare il flip flop ed è chiamato innesco (*trigger*). In un latch D con ingresso di controllo il segnale clock, per tutto il tempo nel quale il segnale rimane a livello alto, è possibile fornire valori di input per far cambiare lo stato: questo può risultare problematico, perché se i segnali di ingresso cambiano prima che il clock possa scendere al livello basso, avviene un'altro passaggio di stato non voluto nello stesso colpo di clock, e potenzialmente portando il latch ad uno stato indefinito.

Per risolvere questo problema, nei Flip Flop la connessione tra ingressi ed uscite dei latch è interrotta prima che l'uscita possa cambiare: in questo modo il nuovo stato del Flip Flop dipenderà dallo stato assunto in corrispondenza del precedente impulso di clock e si eviteranno le transizioni multiple durante lo stesso impulso. Ovviamente, i Flip Flop sono costruiti per funzionare con un singolo segnale di clock. Le combinazioni di input accettate da un Flip Flop e che gli fanno cambiare stato si chiamano funzioni di eccitazione, ed in genere si basano sui latch che compongono il Flip Flop.

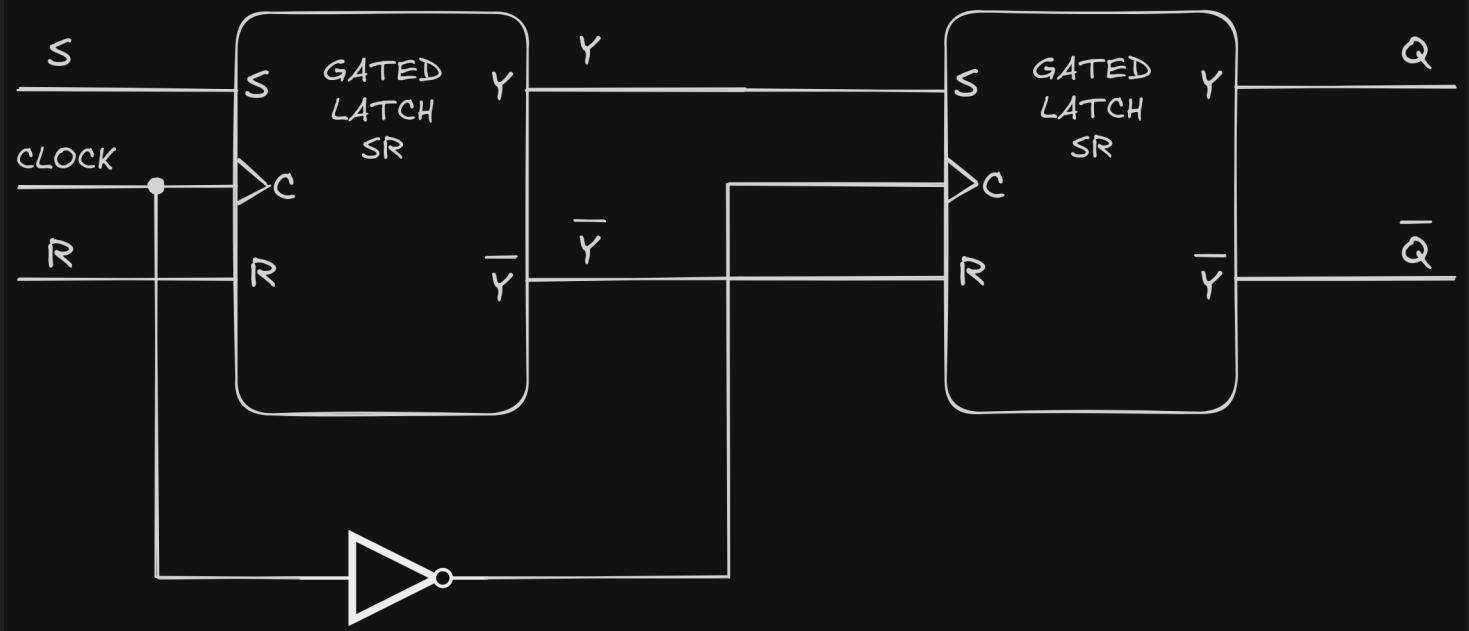
Esistono principalmente due modi per combinare i latch e formare un Flip Flop:

- 1) Gli ingressi del Flip Flop controllano lo stato solo quando il clock è alto, e lo stato del Flip Flop cambia solo quando il clock è basso.
- 2) Lo stato del Flip Flop cambia solo in presenza del fronte di salita, cioè nel momento in cui il clock va da **0** a **1** (*positive edge triggered Flip Flop*), oppure solo in presenza del fronte di discesa, cioè nel momento in cui il clock va da **1** a **0** (*negative edge triggered Flip Flop*).

Flip Flop Master-Slave (o SR)

Il Flip Flop Master-Slave segue il primo metodo per combinare i latch per creare un Flip Flop, perché il primo latch controlla il secondo (e da qui anche il nome).

Consiste in due gated latch SR, dove le uscite del primo sono le entrate del secondo, ed il segnale di controllo C è l'impulso di clock per il primo latch, e la sua negazione per il secondo.



Il latch a sinistra è chiamato master, e quello a destra slave. Il triangolo sul segnale di controllo C indica che stiamo usando il clock.

Quando l'ingresso di clock C è uguale a 1 , il master è attivato e dunque gli ingressi S ed R influenzano l'uscita Y , ma lo slave è disabilitato, e dunque manda sull'uscita Q il risultato memorizzato, cioè il valore Y del master ottenuto alla fine dell'impulso scorso.

Quando l'ingresso di clock C è uguale a 0 , il master è disattivato e quindi la sua uscita Y non cambia, mentre lo slave è abilitato e quindi la sua uscita Q corrisponde all'uscita Y del master.

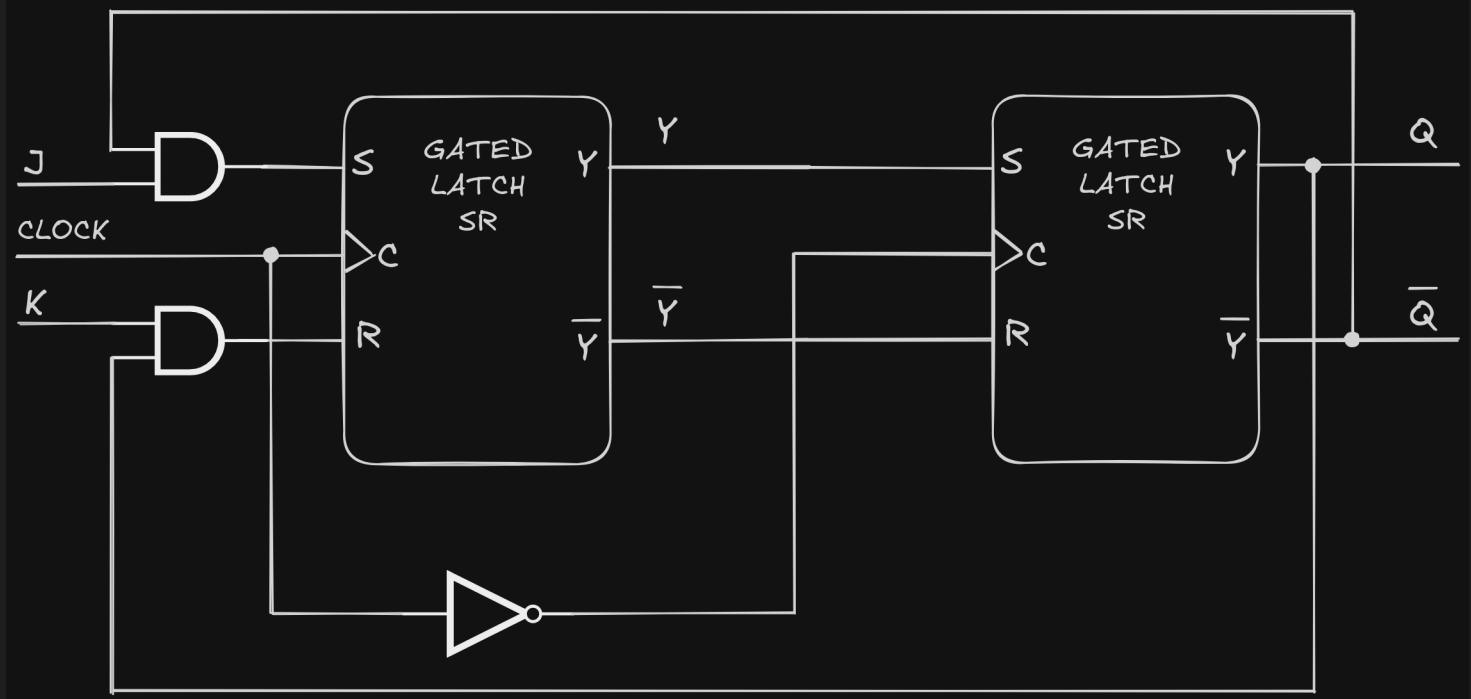
Quindi, stiamo facendo esattamente quanto descritto nel primo metodo per combinare i latch in Flip Flop: quando l'impulso di clock è alto, il master genera il nuovo output, ma lo slave continua a distribuire quello precedente, e quando poi l'impulso di clock diventa basso, lo slave distribuisce il nuovo output, ma il master non ne genera di nuovi.

Il problema del Flip Flop Master-Slave è che usando i gated latch SR, potrebbero comunque verificarsi casi in cui S ed R sono contemporaneamente 1 . Ciò è risolto dai Flip Flop JK.

| Flip Flop JK

Il Flip Flop JK è una versione modificata del Flip Flop Master-Slave che elimina la condizione di uscite indefinite e comportamenti indeterminati.

In questo caso, avere S ed R contemporaneamente ad 1 fa assumere all'uscita il valore complementare a quello che aveva nello stato precedente.



In questo Flip Flop, l'ingresso J si comporta come l'ingresso S , mentre l'ingresso K si comporta come R . Ricordando che quando $Q=0$ e $\bar{Q}=1$ vuol dire che si è nello stato di reset, e quando $Q=1$ e $\bar{Q}=0$ vuol dire che si è nello stato di set, capire il funzionamento è più semplice: nel caso in cui sia J che K siano uguali ad 1, devo controllare il valore di Q e \bar{Q} .

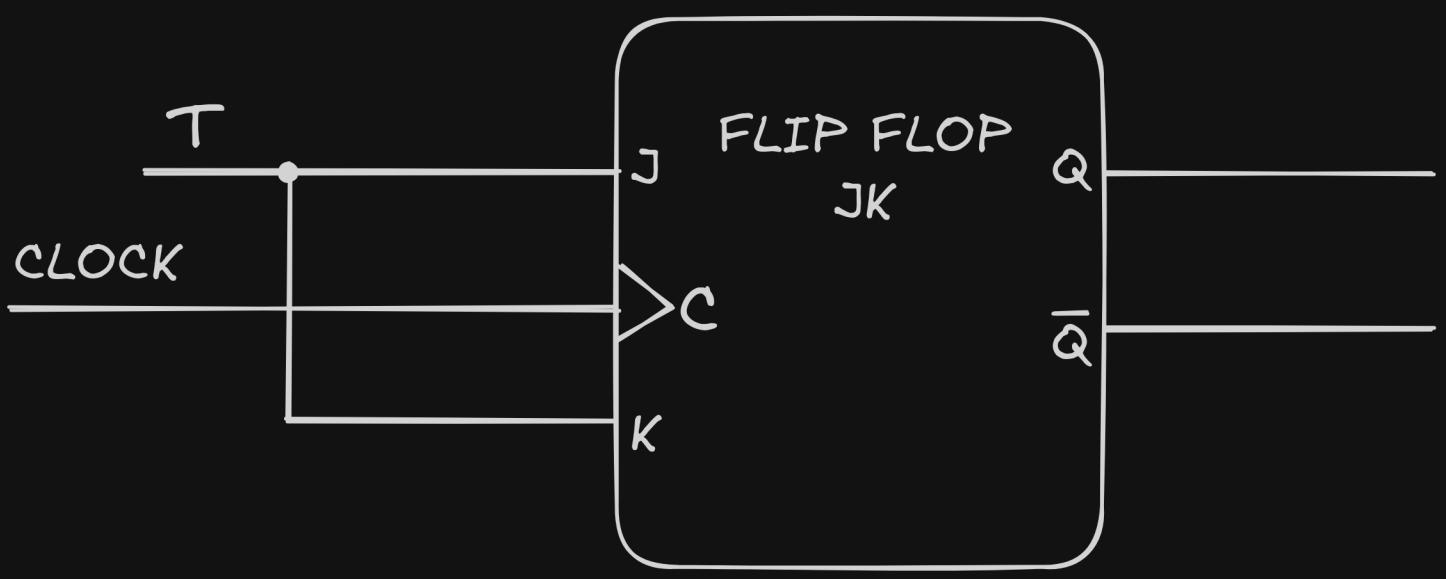
Se $Q=1$, vuol dire che eravamo nello stato di set, e per complementare dobbiamo effettuare un reset: avendo sia K che Q ad 1, si passa allo stato di reset, mentre S sarà 0, perché anche se $J=1$, \bar{Q} è 0 perché appunto ci trovavamo nello stato di set.

Allo stesso modo, se $Q=0$, vuol dire che eravamo nello stato di reset, e per complementare dobbiamo effettuare un set: avendo K ad 1 ma Q a 0, R nel latch risulta essere 0, mentre S nel latch sarà 1, perché $J=1$ e $\bar{Q}=1$ in quanto ci trovavamo nello stato di reset.

J	K	Q	\bar{Q}	Stato
0	0	Precedente	Precedente	Memorizzazione
0	1	0	1	Reset
1	0	1	0	Set
1	1	Negato del precedente	Negato del precedente	Complementazione

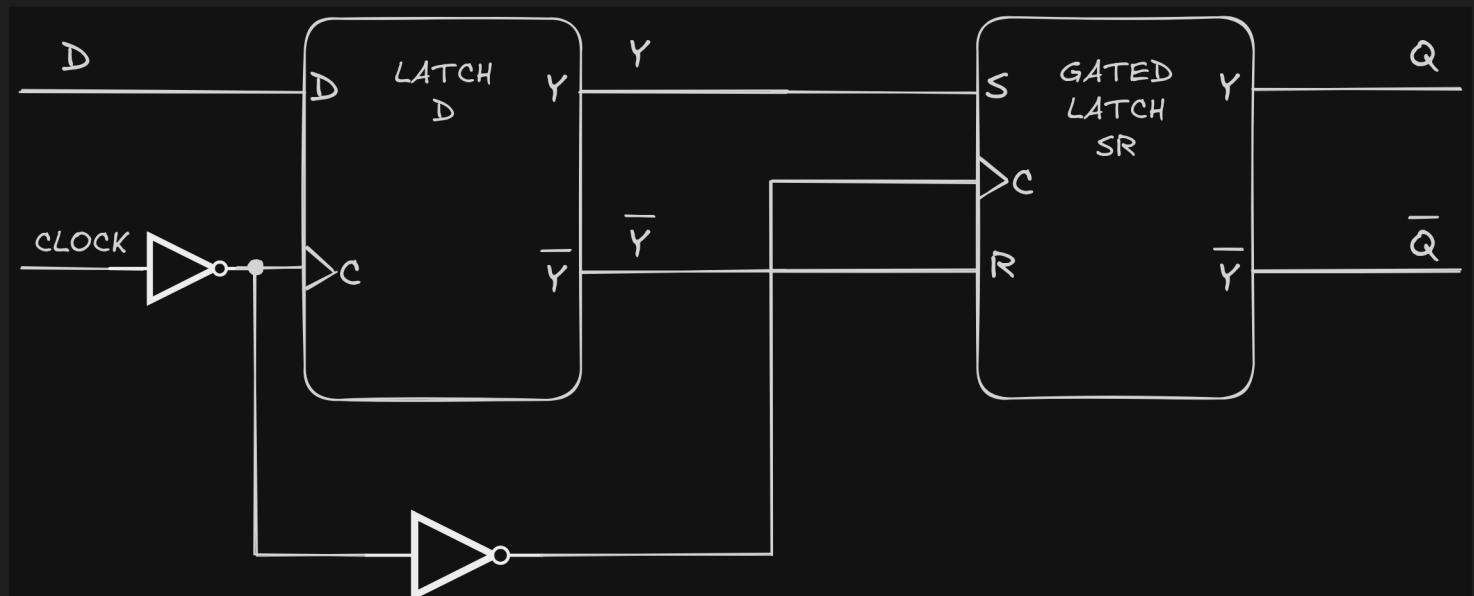
| Flip Flop T

Il Flip Flop T (*toggle*) è un Flip Flop JK dove oltre al clock si usa un unico segnale T che rappresenta l'ingresso sia per J che per K : in questo modo, quando $T=0$ l'uscita Q rimarrà la stessa (stato di memorizzazione), mentre quando $T=1$ invece verrà complementata (stato di complementazione).



| Flip Flop D

I Flip Flop D fanno parte della seconda categoria di Flip Flop, cioè quelli sensibili al fonte di salita o discesa, commutando solo in corrispondenza delle transizioni del segnale di clock.



È molto simile al Flip Flop Master-Slave, ma il master è un latch D ed il suo ingresso di clock è complementato: in questo modo, quando il clock è **0**, il master genera i propri output, e nel momento in cui il clock passa ad **1**, viene disabilitato il master e lo slave distribuisce i valori di output. Il comportamento complessivo del circuito appare quindi attivato dal fronte di salita.

Usare i Flip Flop sensibili ai fronti del clock richiede attenzione alla temporizzazione degli ingressi in relazione al tempo di risposta del Flip Flop: bisogna stabilire

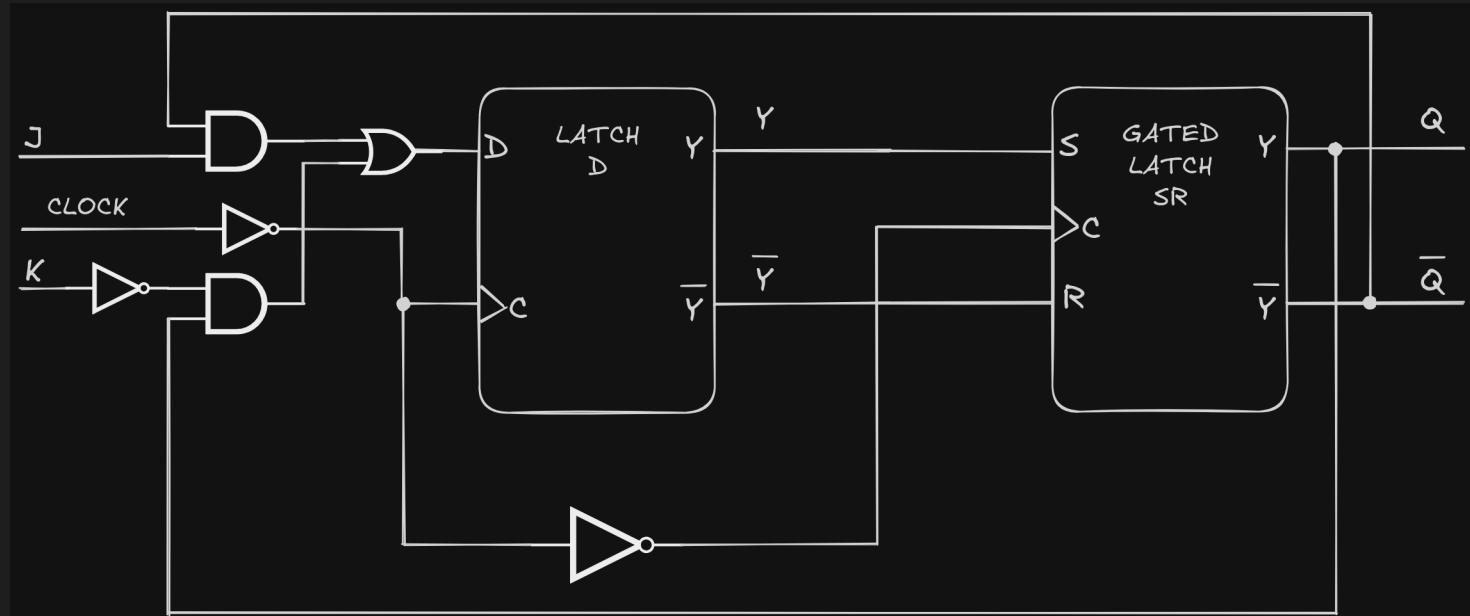
- Il tempo di impostazione (*setup time*), cioè il tempo minimo in cui l'ingresso deve essere stabile prima che arrivi il fronte del clock, in modo da poterlo copiare sul master
- Il tempo di mantenimento (*hold time*), cioè il tempo minimo in cui l'ingresso deve rimanere stabile dopo il passaggio del fronte di clock, affinché il nuovo stato sia stabile in uscita.

Il ritardo di propagazione del Flip Flop è l'intervallo di tempo tra il fronte del clock e l'istante in cui l'uscita è stabile nel nuovo stato. Poiché i cambiamenti delle uscite devono essere separati dal controllo degli ingressi, il ritardo di propagazione deve essere almeno maggiore del massimo hold time del circuito.

Tutti questi parametri sono solitamente forniti dai produttori nelle schede tecniche.

Flip Flop JK sensibile al fronte di clock

Questo tipo di Flip Flop è una variazione del Flip Flop JK, che usa un latch D come master per implementare la sensibilità al fronte di clock.



Analizzando direttamente il circuito, si ha $D=1$ se $J=1$ e $Q=0$ (Il circuito era in stato di reset ed ora sto passando allo stato di set) oppure se $K=0$ e $Q=1$ (Il circuito era in stato di set e ora sto passando allo stato di memorizzazione): quindi se il flop flop è nello stato 0, viene portato ad 1 se $J=1$ indipendentemente dal valore di K e continua a rimanere nello stato 1 quando $K=0$, indipendentemente dal valore di J .

Al contrario, si ha $D=0$ se $K=1$ e $Q=1$ (Il circuito era in stato di set ed ora sto passando allo stato di reset) oppure se $J=0$ e $Q=0$ (Il circuito era in stato di reset e ora sto passando allo stato di memorizzazione): quindi se il Flip Flop è nello stato 1, viene portato a 0 quando $K=0$ indipendentemente da J e rimane nello stato 0 per $J=1$ indipendentemente dal valore di K .

Errori nei Flip Flop dovuti al tempo

Considerando un circuito sequenziale contenente diversi Flip Flop master-slave, con le uscite di alcuni Flip Flop che diventano ingressi di altri, se il segnale di clock arriva a tutti i Flip Flop contemporaneamente ed in modo sincrono, il circuito funziona correttamente: All'inizio di ciascun impulso, alcuni master cambiano stato, mentre gli slave corrispondenti tengono lo stato originario, impedendo il cambio delle uscite dei Flip Flop finché il clock rimane alto. Quando poi il clock ritorna a 0, gli slave propagano i cambi di stato dove

necessario, ma senza compromettere i master, che rimarranno bloccati fino alla prossima salita del segnale.

Dunque, gli stati dei Flip Flop in un sistema sincrono non possono cambiare nello stesso impulso di clock anche se le loro uscite sono collegate ad i loro stessi ingressi o altri Flip Flop.

Tuttavia, ciò funziona solo se tutti i segnali che devono propagarsi dalle uscite dei Flip Flop passano per il circuito combinatorio e ritornano all'ingresso dei Flip Flop master nel lasso di tempo in cui il clock rimane a **0**, cioè prima che il clock passi ad **1**. Se per esempio il ritardo di un circuito combinatorio è tale che quando il clock torna ad **1** S ed R stanno ancora subendo transizioni, si andrebbe ad assumere lo stato sbagliato.

Quanto descritto ha due importanti conseguenze:

- 1) I Flip Flop Master-Slave sono anche Flip-Flop sensibili agli impulsi di clock, perché sono in grado di rispondere ai cambiamenti degli ingressi che si verificano in un qualsiasi momento in cui l'impulso di clock è alto
- 2) Per funzionare correttamente, bisogna assicurarsi che il ritardo generato dal circuito combinatorio sia sufficientemente breve da prevenire il cambiamento di S ed R durante l'impulso di clock.

| Tavole inverse dei Flip Flop

Per il procedimento di [sintesi di una rete](#), tornerà utile avere le tavole inverse dei Flip Flop, cioè sapere quali possono essere i valori di ingresso dei Flip Flop a partire dallo stato corrente y (quello in cui è il Flip Flop prima di elaborare gli ingressi) e dallo stato futuro Y (quello in cui passerà il Flip Flop dopo aver elaborato gli ingressi).

| Master-Slave

y	Y	S	R
0	0	0	δ
0	1	1	0
1	0	0	1
1	1	δ	0

- Se lo stato y è **0** e lo stato futuro Y sarà **0**, allora la combinazione SR è stata quella di memorizzazione ($S=0$ ed $R=0$) o quella di reset ($S=0$ ed $R=1$)
- Se lo stato y è **0** e lo stato futuro Y sarà **1**, allora la combinazione SR è stata per forza quella di set ($S=1$ ed $R=0$)
- Se lo stato y è **1** e lo stato futuro Y sarà **0**, allora la combinazione SR è stata per forza quella di reset ($S=0$ ed $R=1$)

- Se lo stato y è **1** e lo stato futuro Y sarà **1**, allora la combinazione SR è stata quella di memorizzazione ($S=0$ ed $R=0$) o quella di set ($S=1$ ed $R=0$)

| JK

y	Y	J	K
0	0	0	δ
0	1	1	δ
1	0	δ	1
1	1	δ	0

- Se lo stato y è **0** e lo stato futuro Y sarà **0**, allora la combinazione JK è stata quella di memorizzazione ($J=0$ e $K=0$) o quella di reset ($J=0$ e $K=1$)
- Se lo stato y è **0** e lo stato futuro Y sarà **1**, allora la combinazione JK è stata quella di set ($J=1$ e $K=0$) o quella di complementazione ($J=1$ e $K=1$)
- Se lo stato y è **1** e lo stato futuro Y sarà **0**, allora la combinazione JK è stata quella di reset ($J=0$ e $K=1$) o quella di complementazione ($J=1$ e $K=1$)
- Se lo stato y è **1** e lo stato futuro Y sarà **1**, allora la combinazione JK è stata quella di set ($J=1$ e $K=0$) o quella di memorizzazione ($J=0$ e $K=0$)

| T

y	Y	T
0	0	0
0	1	1
1	0	1
1	1	0

- Se lo stato y è **0** e lo stato futuro Y sarà **0**, allora $T=0$, cioè la combinazione JK è stata quella di memorizzazione ($J=0$ e $K=0$)
- Se lo stato y è **0** e lo stato futuro Y sarà **1**, allora $T=1$, cioè la combinazione JK è stata quella di complementazione ($J=1$ e $K=1$)
- Se lo stato y è **1** e lo stato futuro Y sarà **0**, allora $T=1$, cioè la combinazione JK è stata quella di complementazione ($J=1$ e $K=1$)
- Se lo stato y è **1** e lo stato futuro Y sarà **1**, allora $T=0$, cioè la combinazione JK è stata quella di memorizzazione ($J=0$ e $K=0$)

| D

y	Y	D
0	0	0
0	1	1
1	0	0
1	1	1

- Se lo stato y è **0** e lo stato futuro Y sarà **0**, allora $D=0$, che corrisponde allo stato di reset
- Se lo stato y è **0** e lo stato futuro Y sarà **1**, allora $D=1$, che corrisponde allo stato di set
- Se lo stato y è **1** e lo stato futuro Y sarà **0**, allora $D=0$, che corrisponde allo stato di reset
- Se lo stato y è **1** e lo stato futuro Y sarà **1**, allora $D=1$, che corrisponde allo stato di set

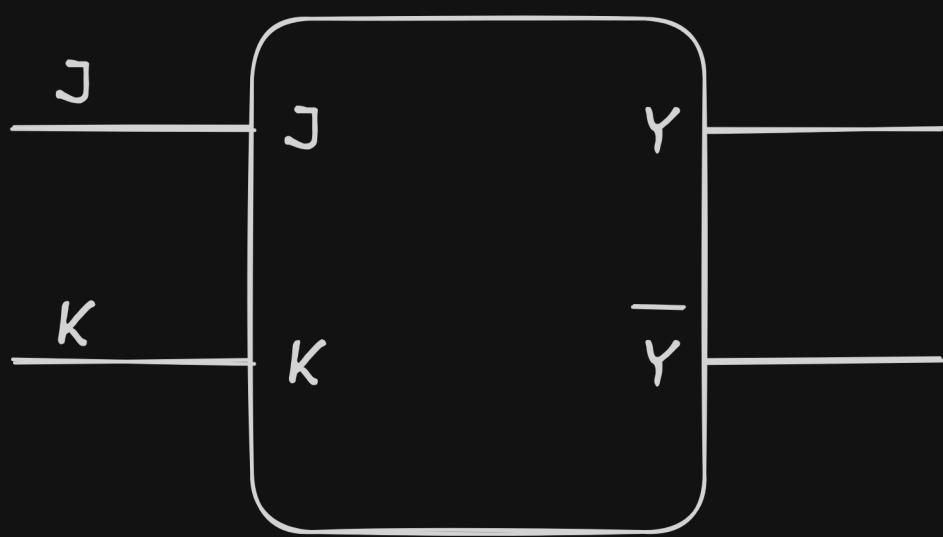
| Automi a stati finiti

Gli automi a stati finiti (*Finite State Machine* o *Finite State Automaton*) sono un metodo di rappresentazione degli stati di una rete sequenziale, dove si elencano tutti gli stati, si specifica da quali stati di partenza si può raggiungere quello di destinazione e con quali input, e l'output generato dal circuito durante il passaggio di stato.

Ci sono due modi equivalenti per rappresentare gli automi a stati finiti:

- 1) Diagramma di Mealy
- 2) Diagramma di Moore

Decidiamo dunque di disegnare l'automa a stati finiti del seguente Flip Flop JK



J	K	Y
0	0	Y
0	1	0
1	0	1
1	1	\bar{Y}

Ho riusato Y per indicare che l'output rimane uguale / viene complementato

Poiché dobbiamo rappresentare tutti gli stati, indicare come passare da uno all'altro e gli output che vengono prodotti, torna utile stendere la tavola degli stati futuri: elenchiamo le combinazioni di input (nel nostro caso, le varie combinazioni con J e K), il valore che viene mandato in output attualmente dal Flip Flop (cioè lo *stato attuale*, che indicheremo con y), il valore che verrà mandato in output dal Flip Flop dopo aver applicato gli ingressi J e K (cioè lo *stato futuro*, che indicheremo con Y) ed infine il valore di output dell'intero circuito sequenziale (che indicheremo con Z , ma in questo caso coincide ad Y perché l'intero circuito è formato dal Flip Flop JK e basta).

J	K	y	Y	Z
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1

J	K	y	Y	Z
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

Se può aiutare a capire meglio come è stata compilata questa tavola, si può leggere così:

- Nella prima riga, l'output del Flip Flop era $y=0$, quindi se ora applico gli input $J=0$ e $K=0$, allora in futuro il suo output sarà 0 , e l'output del circuito sarà 0 (Perché abbiamo detto che $J=0$ e $K=0$ rappresenta la memorizzazione, e quindi il nuovo output Y sarà uguale al vecchio output y , Z per il motivo detto prima è uguale ad Y)
- Nella quinta riga, l'output del Flip Flop era $y=0$, quindi se ora applico gli input $J=1$ e $K=0$, allora in futuro il suo output sarà 1 , e l'output del circuito sarà 1 (Perché abbiamo detto che $J=1$ e $K=0$ rappresenta il set, e quindi il nuovo output Y sarà 1 , Z per il motivo detto prima è uguale ad Y)
- Ecc.

Adesso che abbiamo tutte le informazioni, possiamo procedere a progettare l'automa.

| Diagramma di Mealy

Iniziamo cercando di capire da quanti stati è composto l'automa: la risposta è 2^n o meno stati, dove n è il numero di bit (e quindi di Flip Flop) usati per rappresentare lo stato dell'intero circuito sequenziale.

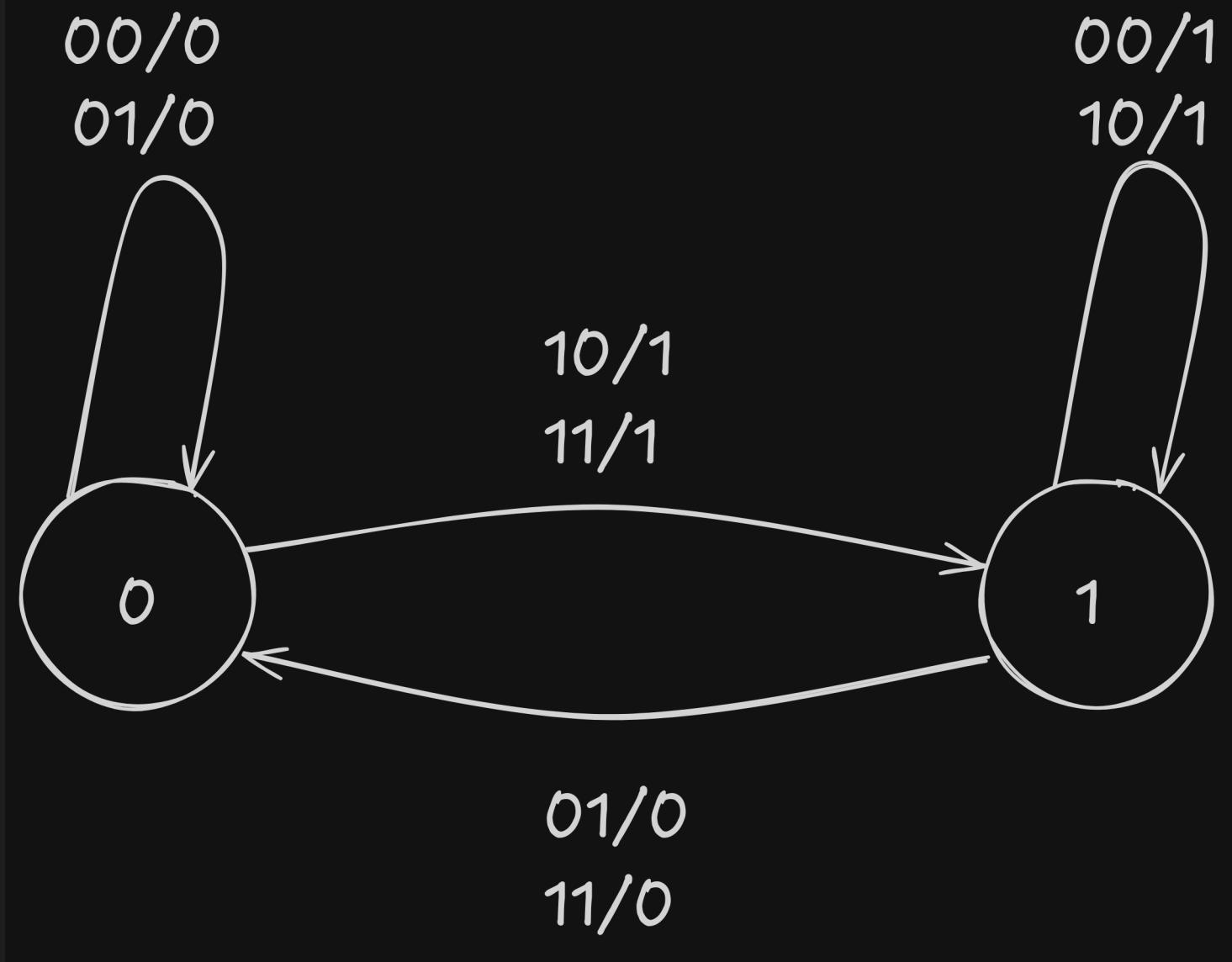
In questo caso, l'output del nostro circuito dipende da un solo Flip Flop, che produce un bit:

- O produce in output 0
- Oppure produce in output 1

Dunque possiamo rappresentare l'automa con $2^1 = 2$ stati:

- 1) L'output del Flip Flop è 0
- 2) L'output del Flip Flop è 1

Disegniamo l'automa:

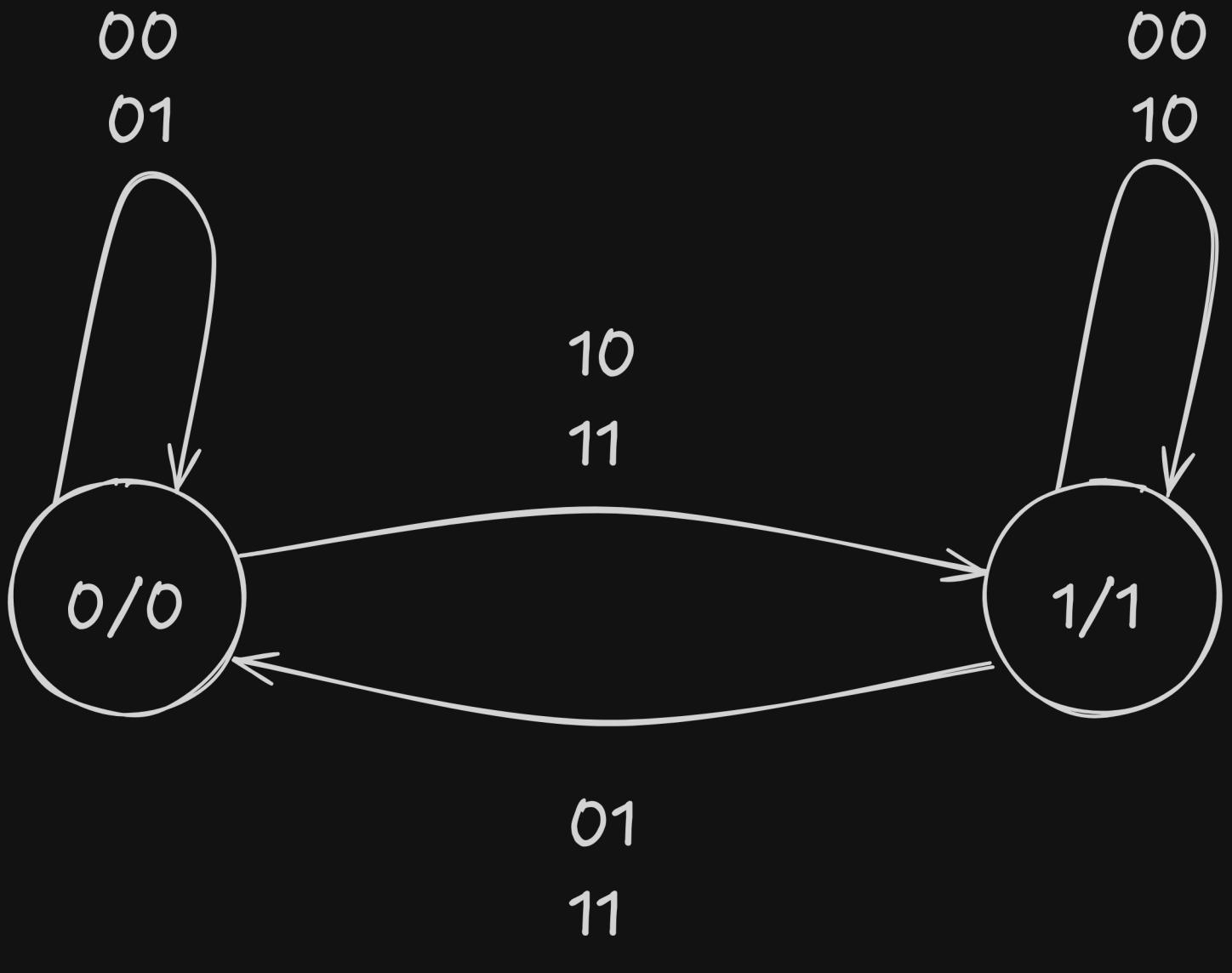


Nel diagramma di Mealy:

- I cerchi con i numeri all'interno sono gli stati. Ho deciso di chiamarli "0" ed "1" per indicare che in uno stato viene mandato in output `0` e nell'altro `1`, ma qualsiasi nome va bene, potevo chiamarli anche "spento" e "acceso"
- Le frecce con i numeri lungo il tracollo rappresentano il passaggio di stato: la freccia parte dallo stato di partenza y ed arriva allo stato di destinazione Y , mentre i numeri rappresentano i valori di input e di output dell'intero circuito, nel formato JK/Z . Posso anche mettere vicino le combinazioni che fanno la stessa cosa. Le frecce che partono da uno stato e puntano sullo stesso rappresentano una combinazione che non fa cambiare lo stato. Dunque, posso leggere per esempio la freccia centrale alta come "*Quando mi trovo nello stato 0, se fornisco come input $J=1$ e $K=0$, ottengo $Z=1$ e passo nello stato 1. Avviene la stessa cosa se quando mi trovo nello stato 0, fornisco come input $J=1$ e $K=1$, ottenendo come output 1 e passando nello stato 1.*".

| Diagramma di Moore

Il diagramma di Moore è molto simile a quello di Mealy, ma le uscite sono invece associate con gli stati invece che con gli input.



Invece di avere un cerchio per ogni stato, abbiamo un cerchio per ogni stato in cui si ottiene un certo output Z . Dunque, sulle frecce del cambio di stato poniamo solo gli input J e K , e le facciamo andare da uno stato alla rispettiva combinazione stato/output.

| Tabella del diagramma

La tabella del diagramma rappresenta in forma tabellare, lo stato futuro (Y) e l'output del circuito sequenziale (Z) a partire dallo stato corrente (y) e dagli ingressi (J e K).

Le righe rappresentano lo stato corrente, le colonne gli input, e nelle celle vanno inserite le combinazioni stato futuro/output.

Ingressi JK

A handwritten diagram of a state transition table for a JK flip-flop. The table has four columns labeled 00, 01, 10, and 11 at the top. The rows are labeled 0 and 1 on the left, representing current states. The entries in the table are binary pairs: (current state, next state). Red annotations include curly braces on the left and top, a bracket on the right side labeled "Stato futuro/output", and a red arrow pointing from the text to the entry 1/1 in the bottom-right cell.

		00	01	10	11	Stato futuro/output
Stati	0	0/0	0/0	1/1	1/1	
	1	1/1	0/0	1/1	0/0	

| Passaggio tra i modelli

Per ogni automa di Mealy, esiste l'equivalente automa di Moore e viceversa. Per eseguire il passaggio da un modello all'altro, è bene avere una definizione più completa dei loro componenti.

Un automa di Mealy è definito da:

- Q : l'insieme finito che ha come elementi gli stati che l'automa può assumere
- Σ : l'insieme che ha come elementi le combinazioni di input che si possono fornire all'automa
- δ : la funzione di transizione $f : Q \times \Sigma \rightarrow Q$, cioè la funzione che dato uno stato ed una combinazione di input restituisce uno stato.
- λ : la funzione di output $f : Q \times \Sigma \rightarrow \Gamma$ dove Γ è l'insieme che ha come elementi i valori di output che l'automa può fornire. Quindi λ è una funzione che dato uno stato ed una combinazione di input restituisce un output.
- q_0 : lo stato iniziale dell'automa

L'automa di Moore è definito nello stesso modo, ma con λ diverso:

λ è la funzione di output $f : Q \rightarrow \Gamma$ dove Γ è l'insieme che ha come elementi i valori di output che l'automa può fornire. Quindi la funzione di output λ nel caso dell'automa di Moore è una funzione che dato uno stato restituisce un output, senza necessitare e quindi dipendere dall'input.

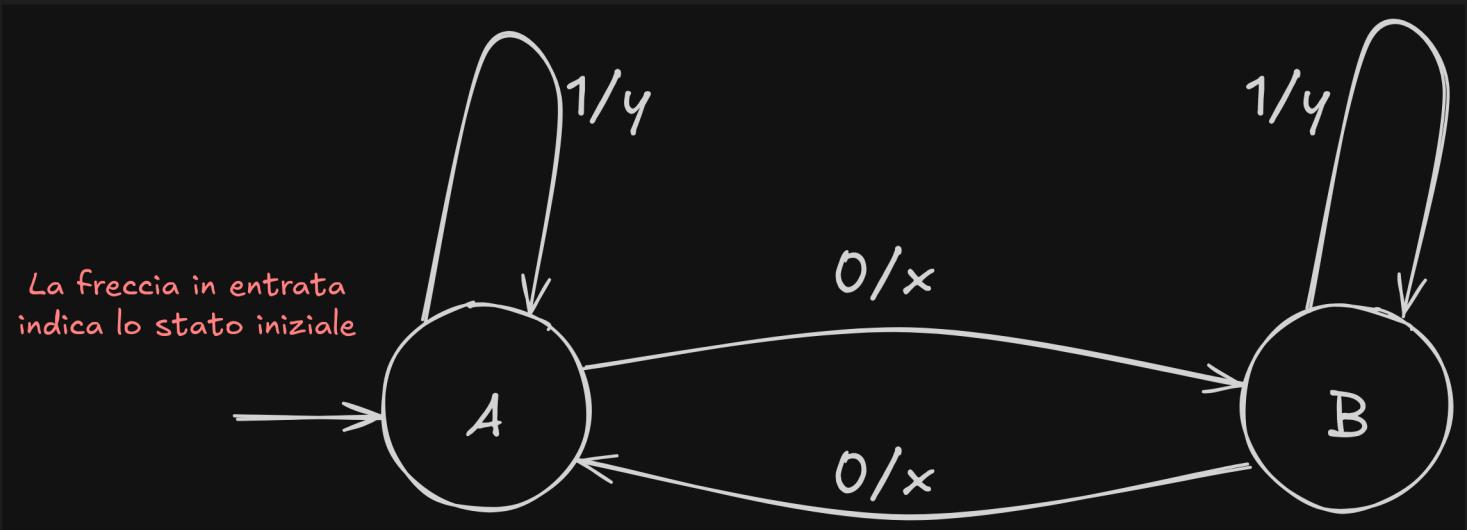
Invece, la funzione di transizione δ continua a necessitare e dipendere dall'input.

| Da Mealy a Moore

Dato il seguente diagramma di Mealy con:

- $Q = \{A, B\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{x, y\}$

- $\delta(A, 0) = B, \lambda(A, 0) = x$
- $\delta(A, 1) = A, \lambda(A, 1) = y$
- $\delta(B, 0) = A, \lambda(B, 0) = x$
- $\delta(B, 1) = B, \lambda(B, 1) = y$

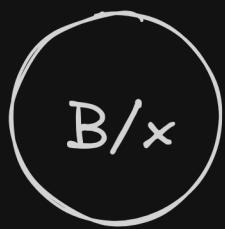
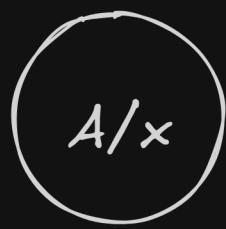


Vogliamo ricavare l'equivalente diagramma di Moore.

- 1) Per ogni stato q nel diagramma di Mealy, creiamo nel diagramma di Moore un nuovo stato q/out per ogni possibile output che lo stato q può produrre. Quindi in questo caso, andremo a creare gli stati $A/x, A/y, B/x$ e B/y e quindi, $Q = \{A/x, A/y, B/x, B/y\}$ e
 - 2) $\lambda(A/x) \rightarrow x$
 - 3) $\lambda(A/y) \rightarrow y$
 - 4) $\lambda(B/x) \rightarrow x$
 - 5) $\lambda(B/y) \rightarrow y$
- 6) Per ogni transizione $(q, \text{input}) \rightarrow q'$ nel diagramma di Mealy, creiamo nel diagramma di Moore una transizione $(q/out, \text{input}) \rightarrow q'/out$ per ogni out di q . Quindi in questo caso, visto che nel diagramma di Mealy c'è la transizione $(A, 0) \rightarrow B$, devo creare $(A/x, 0) \rightarrow B/x$ e $(A/y, 0) \rightarrow B/x$, e così via per gli altri stati. Infatti, non importa se precedentemente l'output era x o y , passando 0 come input ad A/x o A/y arrivo sempre a B/x . Dunque:
 - 7) $\delta(A/x, 0) = B/x$
 - 8) $\delta(A/y, 0) = B/x$
 - 9) $\delta(A/x, 1) = A/y$
 - 10) $\delta(A/y, 1) = A/y$
 - 11) $\delta(B/x, 0) = A/x$
 - 12) $\delta(B/y, 0) = A/x$
 - 13) $\delta(B/x, 1) = B/y$
 - 14) $\delta(B/y, 1) = B/y$

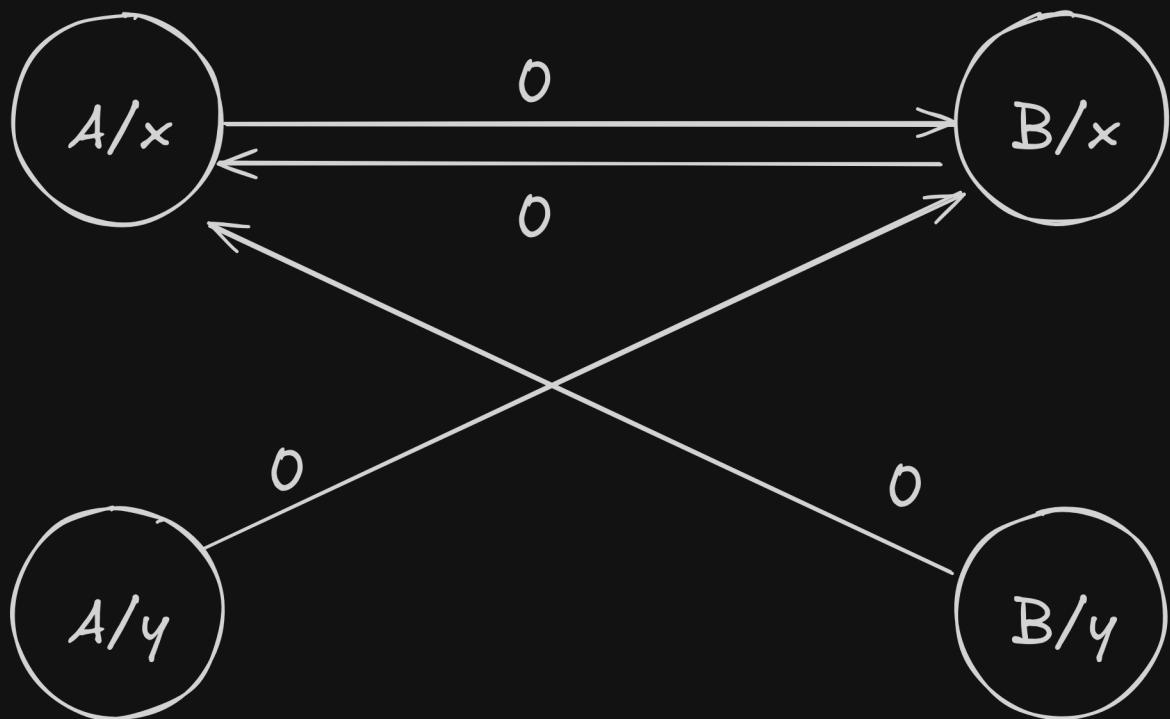
15) Lo stato iniziale q_0 rimane lo stesso del diagramma di Mealy.

1) Crea i nuovi stati

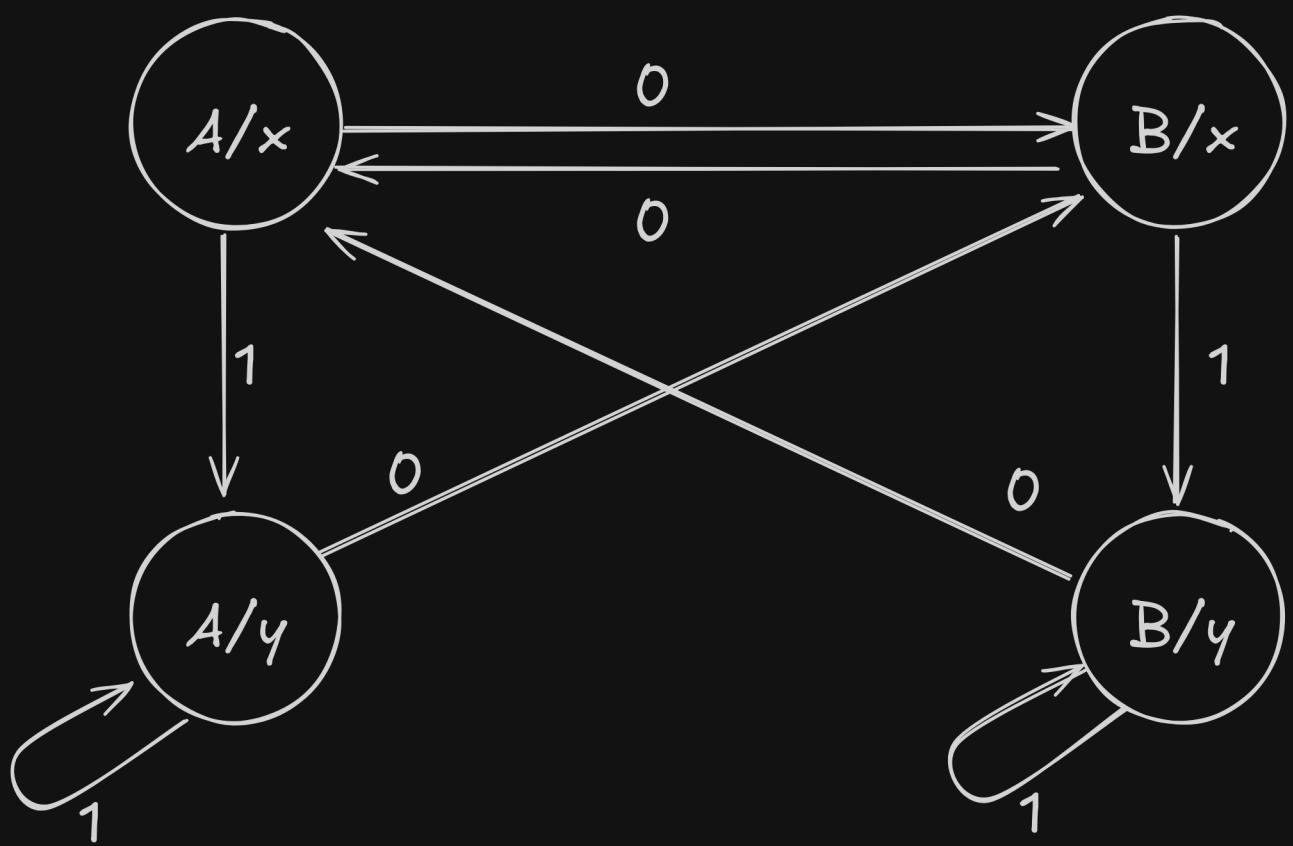


2) Crea le nuove transizioni

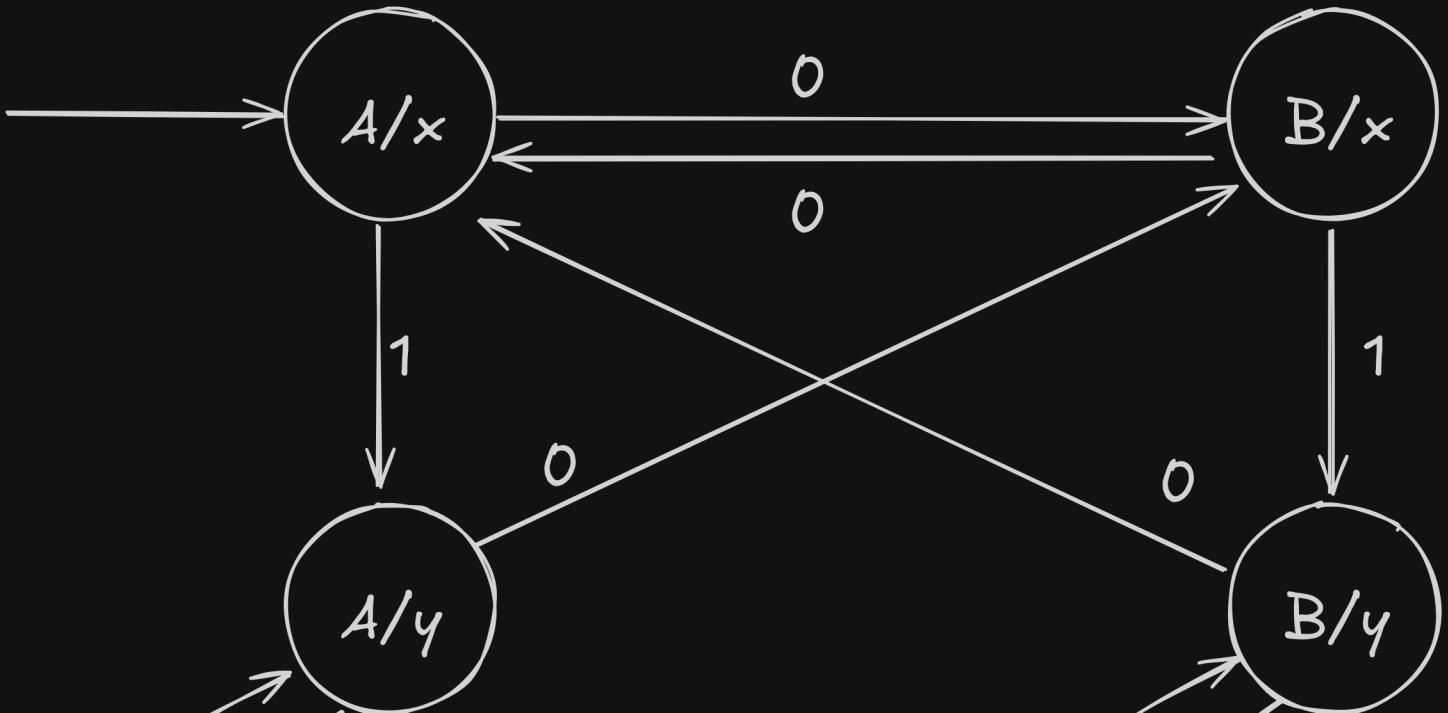
2.1) Per input 0



2.2) e per input 1

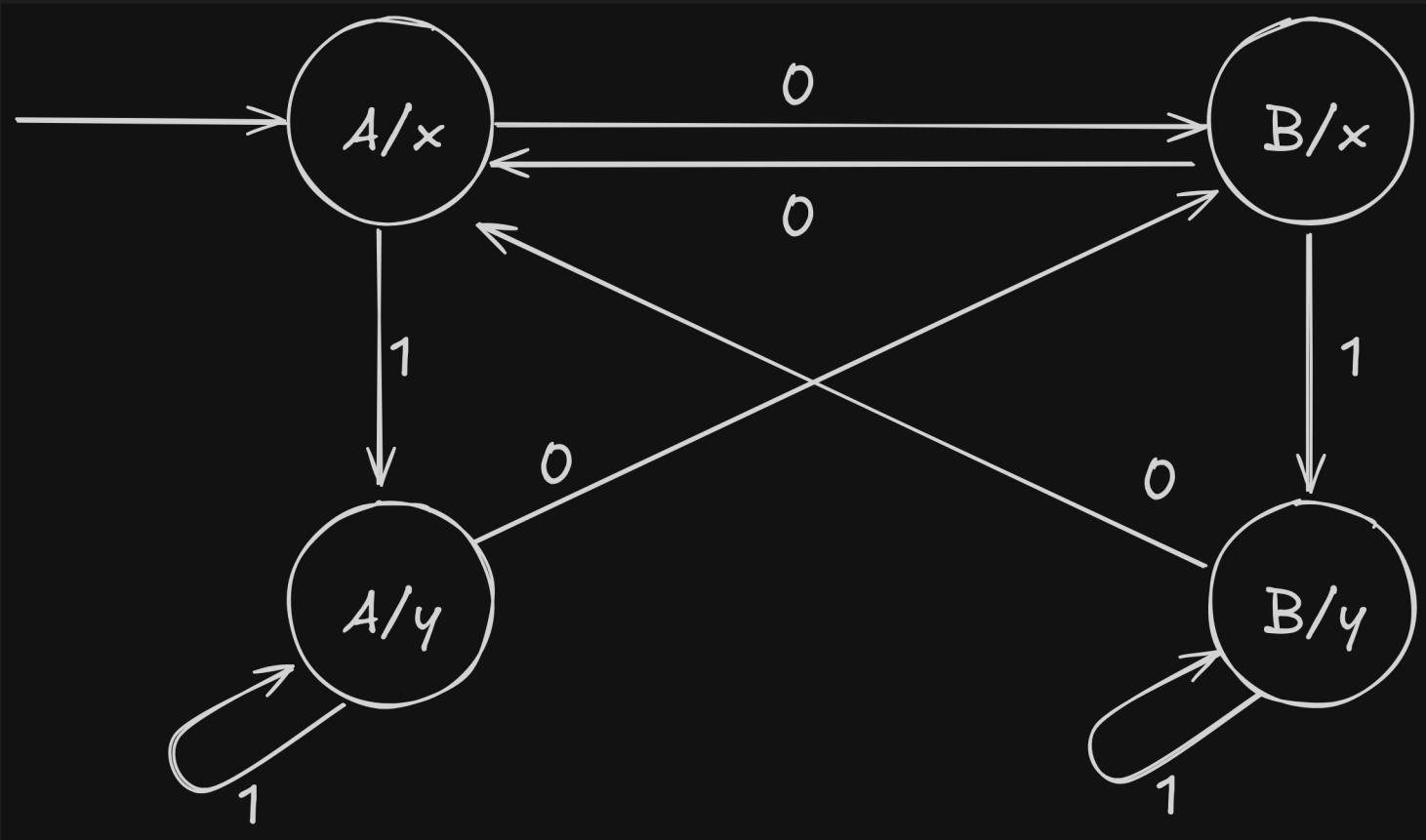


3) Reimposta lo stato iniziale



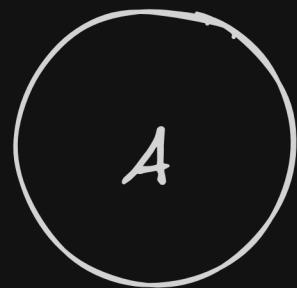
| Da Moore a Mealy

Cerchiamo di riportare il diagramma di Moore che ci siamo procurati in un diagramma di Mealy.



- 1) Per ogni stato q/out nel diagramma di Moore, crea nel diagramma di Mealy un singolo stato q , cioè, elenca ogni stato senza gli output e rimuovi i duplicati.
- 2) Per ogni transizione nel diagramma di Moore, prendi l'output associato con lo stato di destinazione del diagramma di Moore, ed assegna lo insieme agli input alla transizione nel diagramma di Mealy. Quindi per esempio, la transizione che va da A/x con input 0 a B/x nel diagramma di Moore, nel diagramma di Mealy diventa una transizione da A a B con input 0 ed output x .
- 3) Lo stato iniziale q_0 rimane lo stesso del diagramma di Moore.

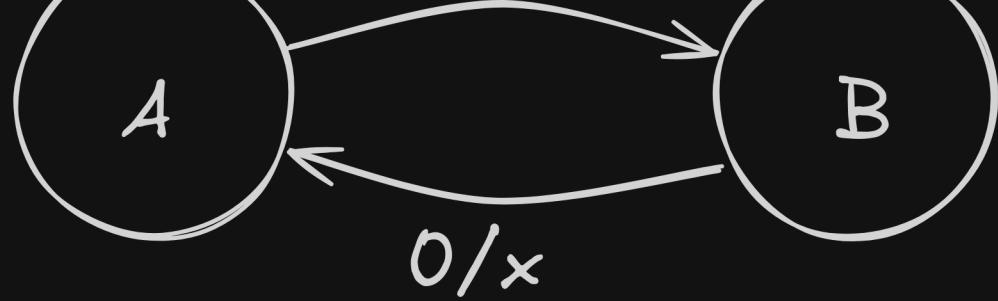
1) Crea i nuovi stati



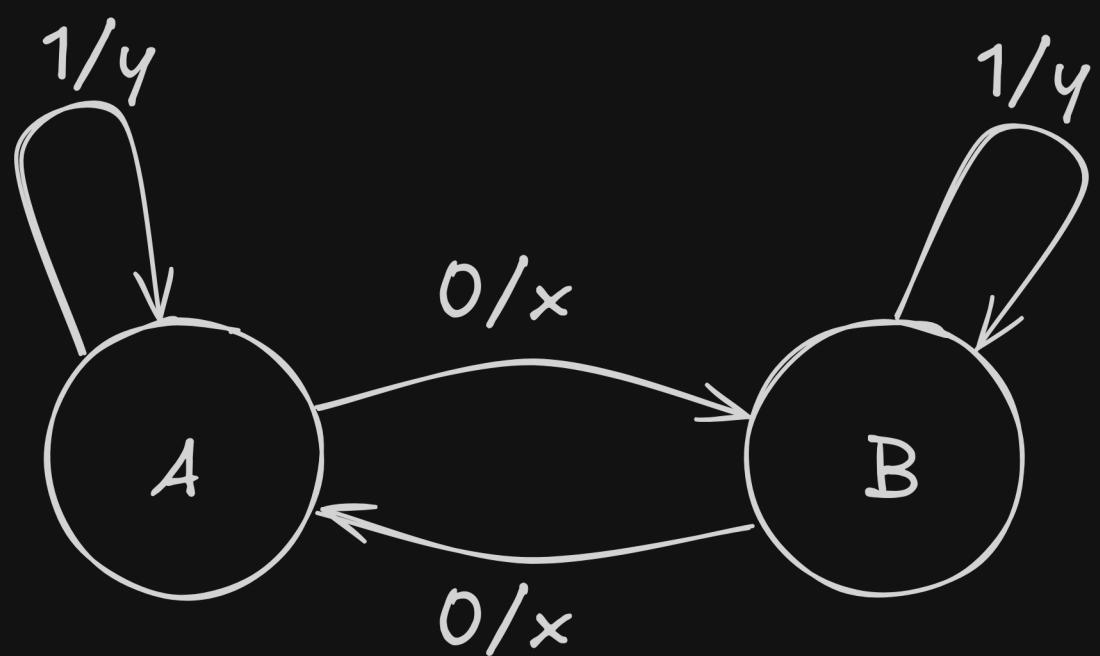
2) Crea le nuove transizioni

2.1) Per input 0

0/x



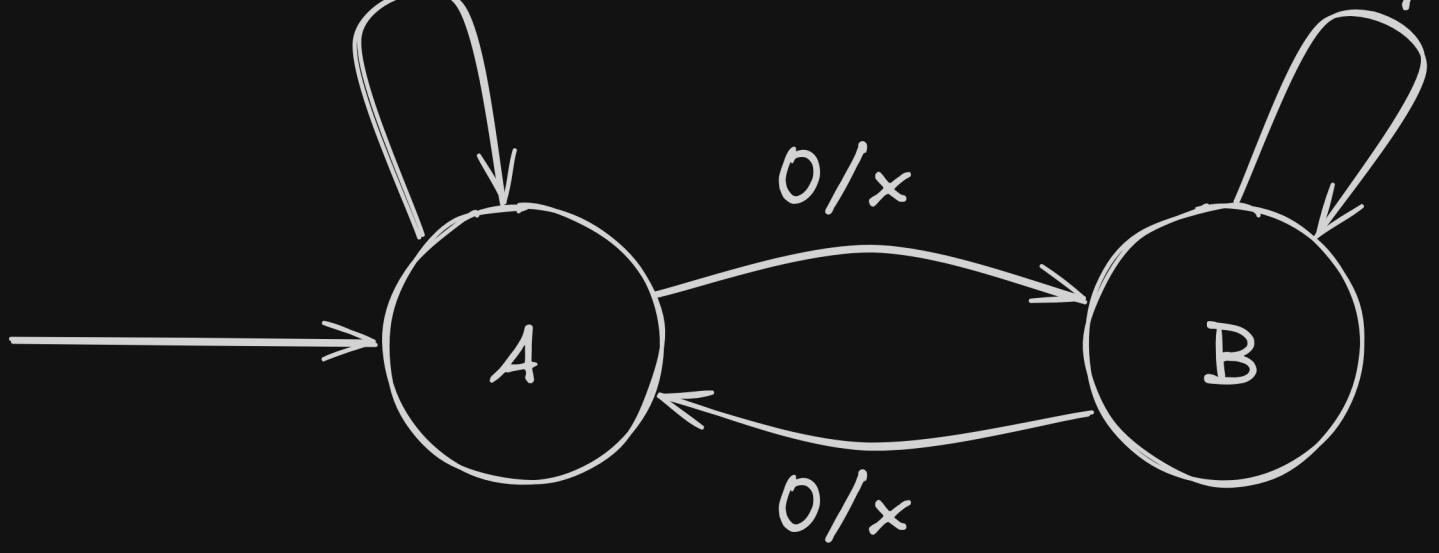
2.2) e per input 1



3) Reimposta lo stato iniziale

1/y

1/y



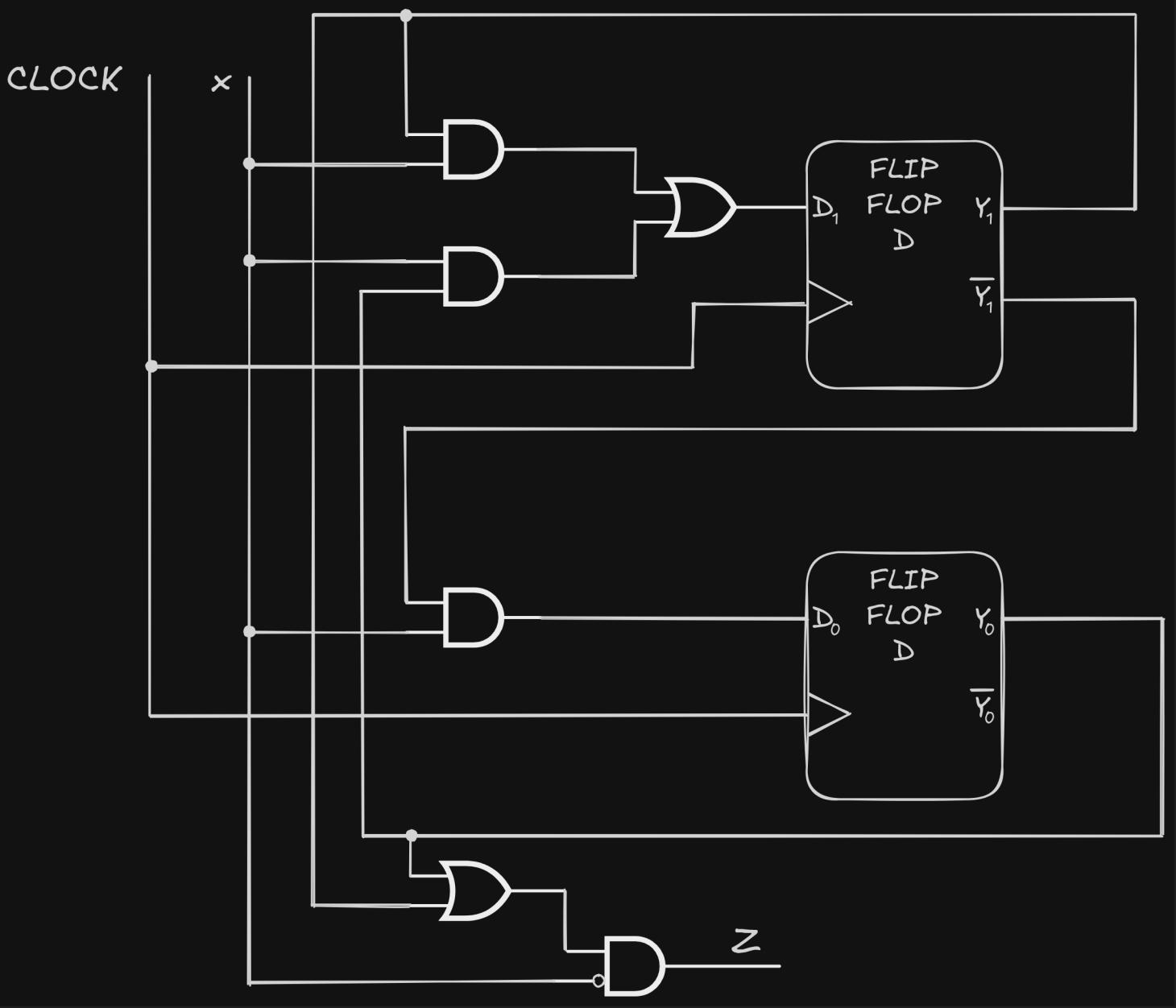
| Analisi di circuiti sequenziali

L'analisi di un circuito sequenziale consiste nello specificare il comportamento di un circuito a partire dal suo diagramma logico. Ciò significa che dobbiamo individuare:

- Espressioni booleane delle funzioni di eccitazione (cioè gli input dei Flip Flop) e dell'output del circuito
- Tavola degli stati futuri (che comprende ingressi del circuito, stato attuale dei Flip Flop, funzioni di eccitazione e stato futuro dei Flip Flop)
- Diagramma di stato della rete (Quindi automa a stati finiti e tabella del diagramma)

Procediamo dunque ad analizzare il seguente circuito:

| Diagramma Logico



Iniziamo facendo alcune osservazioni:

- Oltre al clock, il circuito accetta un solo ingresso, chiamato x
- Vengono usati due Flip Flop D sensibili al fronte di clock che accettano come ingresso un bit ciascuno, D_1 e D_0 , e rilasciano in output y_1 e \bar{y}_1 il primo e y_0 e \bar{y}_0 il secondo.
- Il circuito produce un bit come output, chiamato Z

| Espressioni booleane

Dobbiamo trovare le espressioni booleane che rappresentano le funzioni di eccitazione dei Flip Flop (cioè D_0 e D_1) e l'output del circuito (cioè Z).

Poiché stiamo facendo l'analisi di un circuito, ci basta ricavare le espressioni dal diagramma, osservando da quali segnali vengono attraversate le porte che producono i valori che stiamo cercando:

- $D_1 = xy_1 + xy_0$
- $D_0 = x\bar{y}_1$

- $Z = (y_0 + y_1)\bar{x}$

| Tavola stati futuri

Adesso che sappiamo quali variabili vengono usate per trovare ciò che ci interessa, elenchiamo per ogni combinazione i valori delle funzioni di eccitazione D_0 e D_1 , l'output del circuito Z e gli stati futuri dei Flip Flop Y_0 ed Y_1

Funzioni

Variabili	di eccitazione				Output	Stati futuri	
x	y_1	y_0	D_1	D_0	Z	y_1	y_0
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	1	0	0	1
1	0	1	1	1	0	1	1
1	1	0	1	0	0	1	0
1	1	1	1	0	0	1	0

Una volta essermi scritte le varie combinazioni delle variabili, posso usare le espressioni trovate al punto prima per calcolare i valori di D_0 , D_1 e Z , sostituendo le variabili con i rispettivi valori.

Per trovare Y_0 e Y_1 , devo ricordare che sto usando dei Flip Flop di tipo D, e gli input sono D_0 e D_1 . Dunque, ricavo i valori dalla tavola del [latch D](#), visto che le relazioni input/output corrispondono.

| Diagramma di stato della rete

| Automa a stati finiti

In questo caso, l'output del nostro circuito dipende da due Flip Flop, che producono 4 combinazioni da due bit:

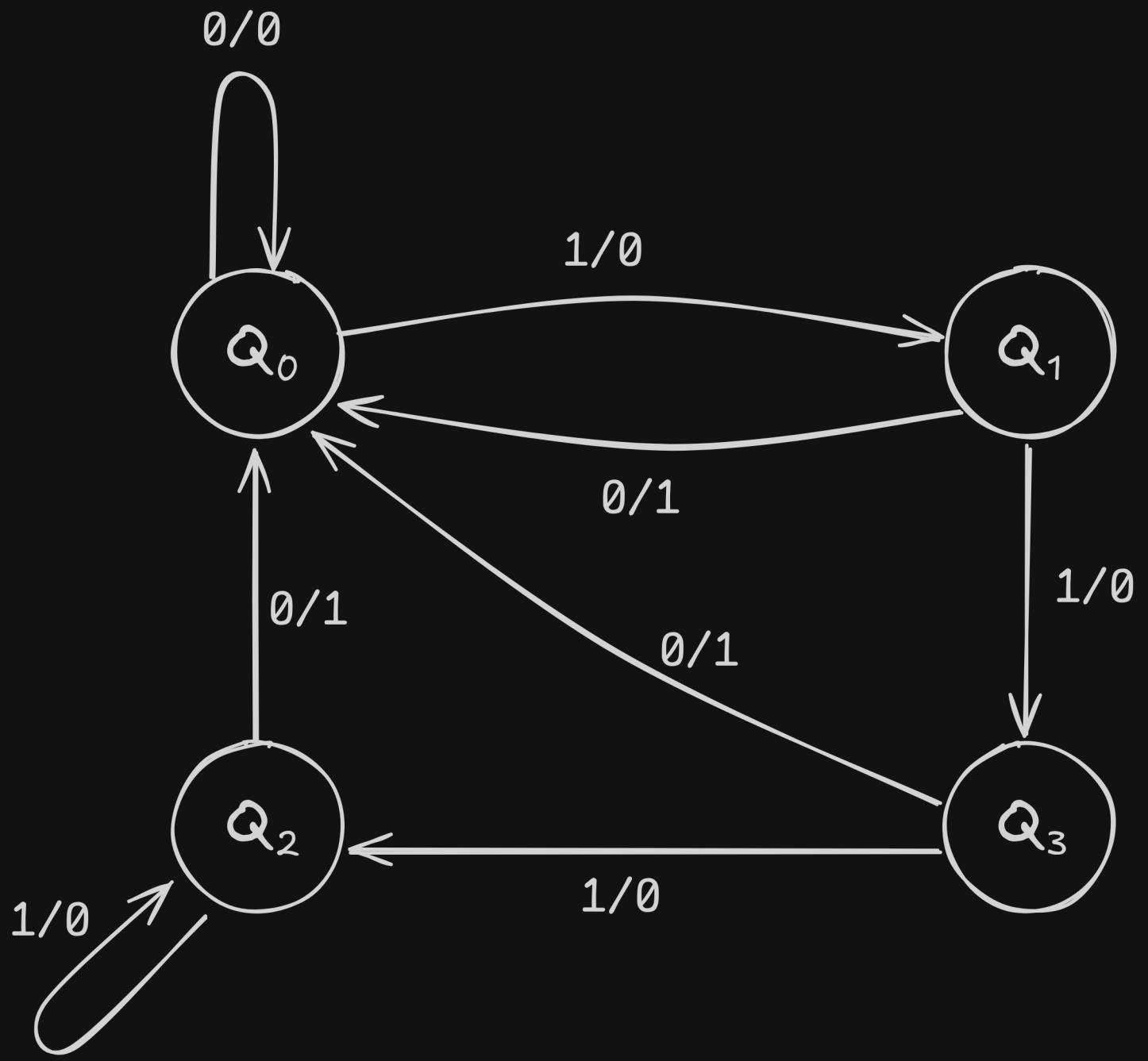
- Il primo Flip Flop produce $y_1 = 0$ ed il secondo $y_0 = 0$
- Il primo Flip Flop produce $y_1 = 0$ ed il secondo $y_0 = 1$
- Il primo Flip Flop produce $y_1 = 1$ ed il secondo $y_0 = 0$
- Il primo Flip Flop produce $y_1 = 1$ ed il secondo $y_0 = 1$

Dunque possiamo rappresentare l'automa con $2^2 = 4$ stati, che per praticità chiameremo:

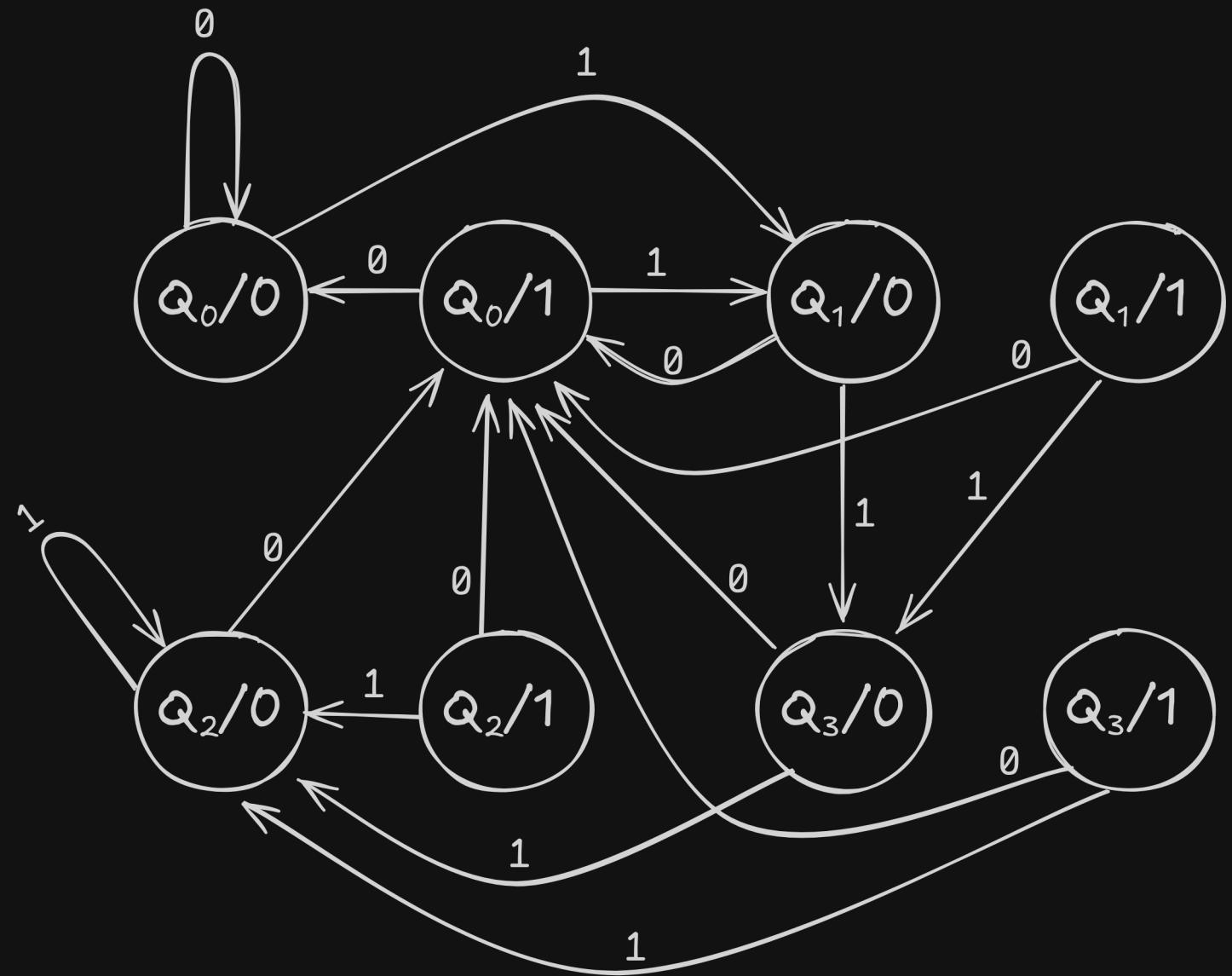
- $y_1 = 0, y_0 = 0 \rightarrow Q_0$
- $y_1 = 0, y_0 = 1 \rightarrow Q_1$
- $y_1 = 1, y_0 = 0 \rightarrow Q_2$
- $y_1 = 1, y_0 = 1 \rightarrow Q_3$

Procediamo a disegnare i diagrammi guardando la tavola degli stati futuri compilata prima, ricordando che le frecce partono dallo stato corrente y_1y_0 , arrivano allo stato futuro Y_1Y_0 e le combinazioni input/output sono x/Z nel diagramma di Mealy e solo x nel diagramma di Moore.

| Diagramma di Mealy



| Diagramma di Moore



| Tabella del diagramma

Ingresso x

	0	1
Q_0	$Q_0/0$	$Q_1/0$
Q_1	$Q_0/1$	$Q_3/0$
Q_2	$Q_0/1$	$Q_2/0$
Q_3	$Q_0/1$	$Q_2/0$

Sintesi di reti sequenziali

La sintesi di un circuito sequenziale consiste nello specificare il comportamento di un circuito e disegnarlo a partire da una descrizione verbale o dalle specifiche della rete da progettare. Ciò significa che dobbiamo individuare:

- Diagramma di stato della rete (Quindi automa a stati finiti e tabella del diagramma)
- Dopo aver scelto la tipologia di Flip Flop da usare, la tavola degli stati futuri (che comprende ingressi del circuito, stato attuale dei Flip Flop, funzioni di eccitazione e stato futuro dei Flip Flop)
- Espressioni booleane delle funzioni di eccitazione (cioè gli input dei Flip Flop) e dell'output del circuito, minimizzate
- Ed infine, disegnare il diagramma logico della rete

Praticamente, dobbiamo eseguire il procedimento inverso a quello di analisi. Cerchiamo quindi di sintetizzare il seguente circuito:

Descrizione verbale del circuito

"Abbiamo x , che è una linea di ingresso in input, e bisogna riconoscere se si riceve la sequenza 1101 , restituendo l'output $Z=1$, in tutti gli altri casi $Z=0$. Considerare anche i casi di sovrapposizione".

In questo caso, non basta un circuito combinatorio, perché ricevo bit per bit dalla stessa linea, e quindi per riconoscere la sequenza devo memorizzare i bit precedenti. Inoltre, devo anche considerare i casi di sovrapposizione (per esempio, se ricevo come primi tre bit 111 , so che questa combinazione è sbagliata perché il terzo bit è 1 invece di 0 , ma non posso "buttare" tutta la combinazione, perché quel terzo 1 potrebbe essere il secondo bit della sequenza 1101).

Per capire meglio la differenza tra il considerare o non la sovrapposizione, immaginiamo di ricevere la seguente sequenza di bit (da sinistra verso destra): 10111101 .

SENZA SOVRAPPOSIZIONE



Non trovato

CON SOVRAPPOSIZIONE



Trovato

| Diagramma di stato della rete

| Tabella del diagramma

Dobbiamo progettare degli stati che ci permettano di rilevare la sequenza specifica 1101 . Per farlo, possiamo usare 4 stati:

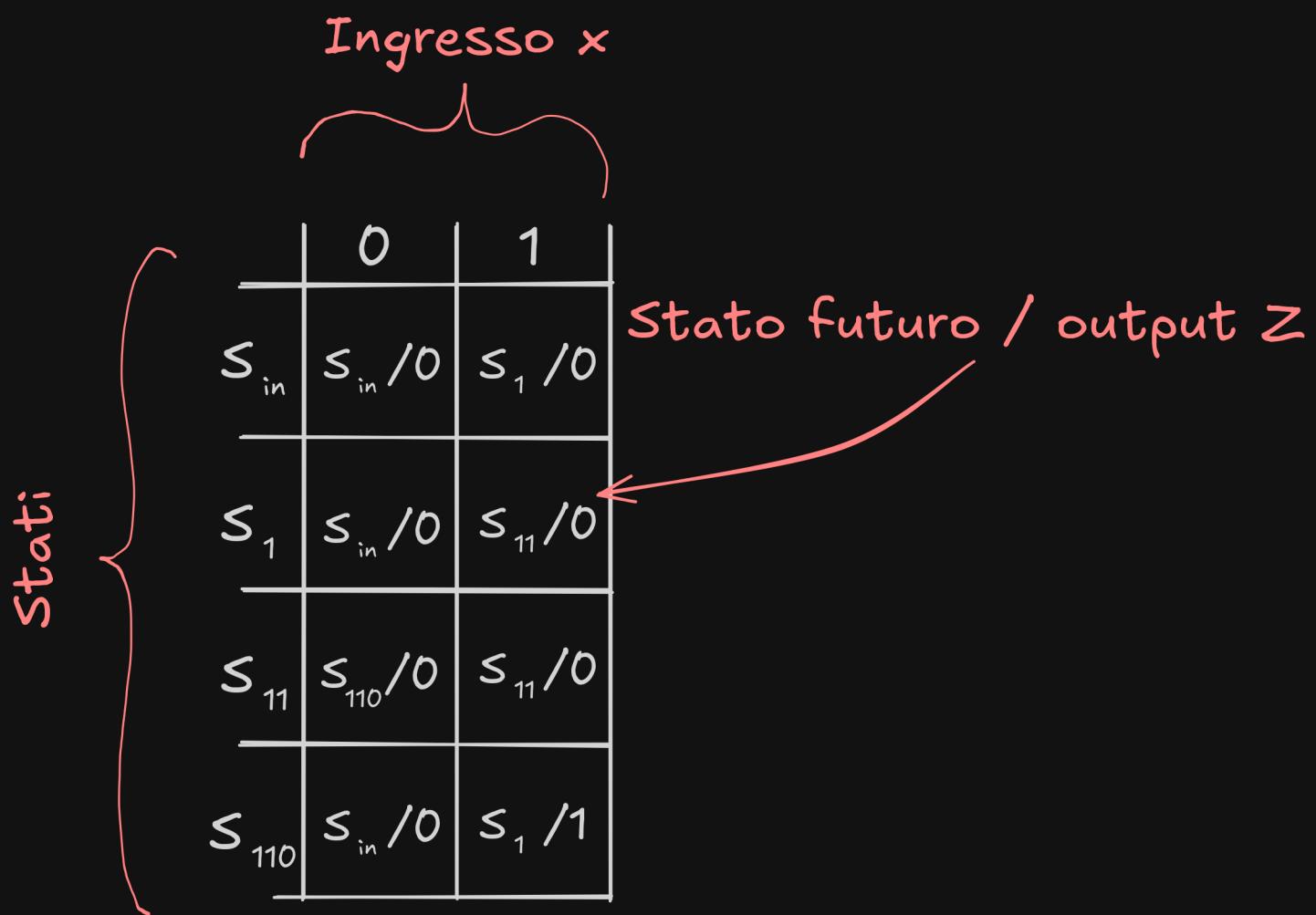
1) S_{in} : La sequenza esaminata è vuota e sono in attesa del primo bit 1

- 2) S_1 : La sequenza esaminata è 1 e sono in attesa del secondo bit 1
- 3) S_{11} : La sequenza esaminata è 11 e sono in attesa del terzo bit 0
- 4) S_{110} : La sequenza esaminata è 110 e sono in attesa dell'ultimo bit 1

In questo modo, quando arriviamo allo stato S_{110} se riceviamo come input 0, la sequenza inserita è sbagliata e dobbiamo buttare tutto, tornando allo stato S_{in} e mandando in output $Z=0$, mentre se riceviamo 1, la sequenza inserita è corretta e quindi possiamo mandare in output $Z=1$ e tornare allo stato S_1 , in modo da usare quest'ultimo 1 ricevuto come il primo bit della sequenza successiva, poiché l'esercizio ci dice di tenere conto della sovrapposizione.

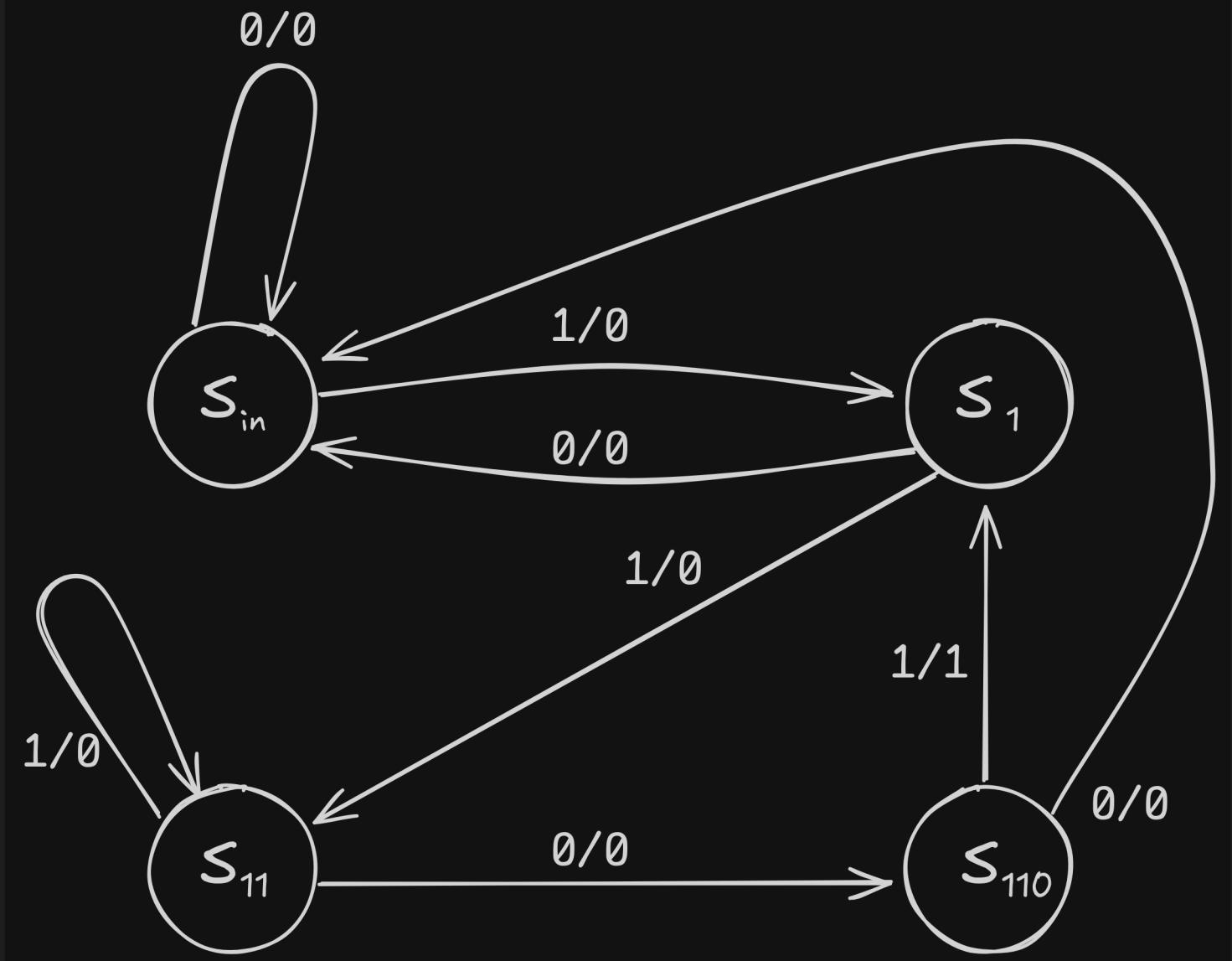
Negli stati S_{in} ed S_1 , se non riceviamo il bit 1 che ci aspettavamo allora buttiamo tutta la sequenza perché sbagliata e torniamo ad S_{in} .

Infine, se nello stato S_{11} ricevo il bit 1 invece di 0 allora rimango nello stato S_{11} per buttare solo il primo bit della sequenza precedente, mantenendo questo 1 ed il precedente perché potrebbero essere i primi due bit della sequenza successiva.

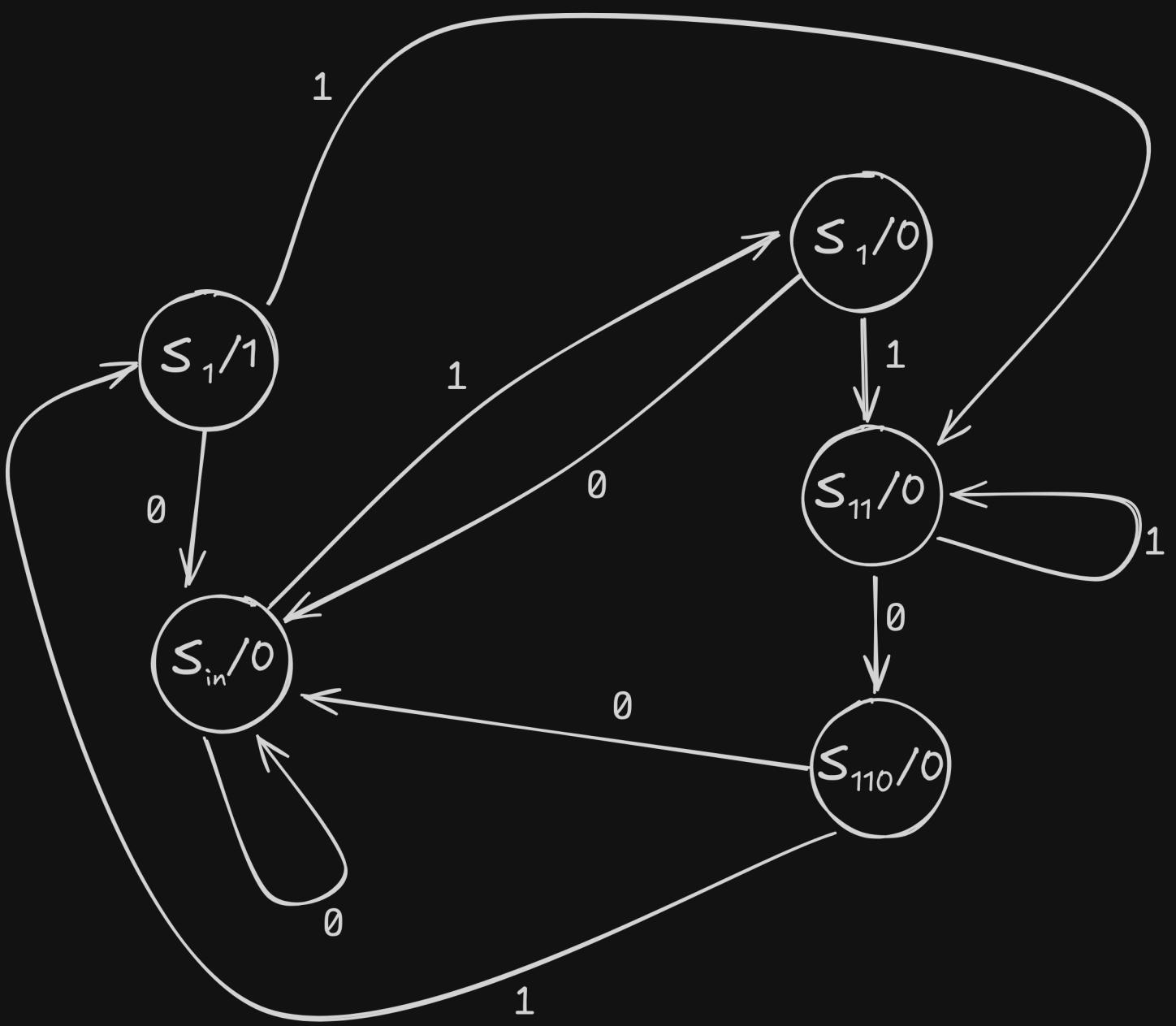


| Automa a stati finiti

| Diagramma di Mealy



| Diagramma di Moore



| Tavola stati futuri

Nella rappresentazione dell'automa, abbiamo usato 4 stati, quindi nel circuito ci serviranno $\log_2 4 = 2$ Flip Flop, in modo da avere 2 bit per rappresentare i 4 stati:

- $S_{in} \rightarrow y_1 = 0, y_0 = 0$
- $S_1 \rightarrow y_1 = 0, y_0 = 1$
- $S_{11} \rightarrow y_1 = 1, y_0 = 0$
- $S_{110} \rightarrow y_1 = 1, y_0 = 1$

Per scrivere la tavola degli stati futuri, dobbiamo prima scegliere quali Flip Flop usare. In genere, i più comodi sono i JK e D, perché il primo è utile quando bisogna gestire due stati di dati grazie alle sue capacità di complementazione, mentre il secondo è semplice da usare, permettendo di mandare 0 o 1 in input per poi ritrovarseli come uscita.

Normalmente basterebbe scegliere solo una delle due opzioni (in questo caso, quella con i FF-D sarebbe più semplice), ma puramente per esercizio, proviamo sia prima con due Flip Flop JK e poi con due Flip Flop D.

Poiché partiamo conoscendo lo stato corrente dei Flip Flop, e dobbiamo trovare le loro combinazioni di ingresso, è utile seguire le [tavole inverse dei Flip Flop](#).

Con i due FF-JK:

Variabili			Stati futuri		Output	Funzioni di eccitazione			
x	y_1	y_0	Y_1	Y_0	Z	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	δ	0	δ
0	0	1	0	0	0	0	δ	δ	1
0	1	0	1	1	0	δ	0	1	δ
0	1	1	0	0	0	δ	1	δ	1
1	0	0	0	1	0	0	δ	1	δ
1	0	1	1	0	0	1	δ	δ	1
1	1	0	1	0	0	δ	0	0	δ
1	1	1	0	1	1	δ	1	δ	0

Ricavati con tavole inverse dei FF

Con i due FF-D:

Variabili Stati futuri Output di eccitazione

x	y_1	y_0	Y_1	Y_0	Z	D_1	D_0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	1	1	0	1	1
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	1	0	0	1	0
1	1	0	1	0	0	1	0
1	1	1	0	1	1	0	1

Ricavati con tavole inverse dei FF

| Espressioni booleane

Dobbiamo trovarci le espressioni booleane minimali che rappresentano l'output del circuito Z , e le funzioni di eccitazione J_1, K_1, J_0 e K_0 nel caso dei due FF-JK, o D_1 e D_0 nel caso dei due FF-D

Per quanto riguarda Z , notiamo che l'unico caso in cui $Z=1$ è quando $x=1$, $y_0=1$, $y_1=1$, e quindi possiamo già affermare che l'espressione minimale di Z è

$$Z = xy_1y_0$$

Mentre le funzioni di eccitazione dobbiamo ricavarcelle dalle [mappe di Karnaugh](#).

Con i due FF-JK:

J_1

$x \backslash y_1 y_0$	00	01	11	10
0	0	0	δ	δ
1	0	1	δ	δ

$$J_1 = xy_0$$

 J_0

$x \backslash y_1 y_0$	00	01	11	10
0	0	0	δ	1
1	1	δ	δ	0

$$J_0 = \bar{x}y_1 + x\bar{y}_1 = x \oplus y_1$$

 K_1

$x \backslash y_1 y_0$	00	01	11	10
0	δ	δ	1	0
1	δ	δ	1	0

$$K_1 = y_0$$

 K_0

$x \backslash y_1 y_0$	00	01	11	10
0	δ	1	1	δ
1	δ	1	0	δ

$$K_0 = \bar{x} + \bar{y}_1$$

- $J_1 = xy_0$
- $J_0 = \bar{x}y_1 + x\bar{y}_1 = x \oplus y_1$
- $K_1 = y_0$
- $K_0 = \bar{x} + \bar{y}_1$

Con i due FF-D:

D_1

$y_1 y_0$	00	01	11	10
x	0	0	0	1
	1	0	1	0
$x\bar{y}_1 y_0$				
$\bar{y}_1 \bar{y}_0$				

$$D_1 = \bar{y}_1 \bar{y}_0 + x\bar{y}_1 y_0$$

$y_1 y_0$	00	01	11	10
x	0	0	0	1
	1	1	0	1
$\bar{x}\bar{y}_1 \bar{y}_0$				
$x\bar{y}_1 y_0$				

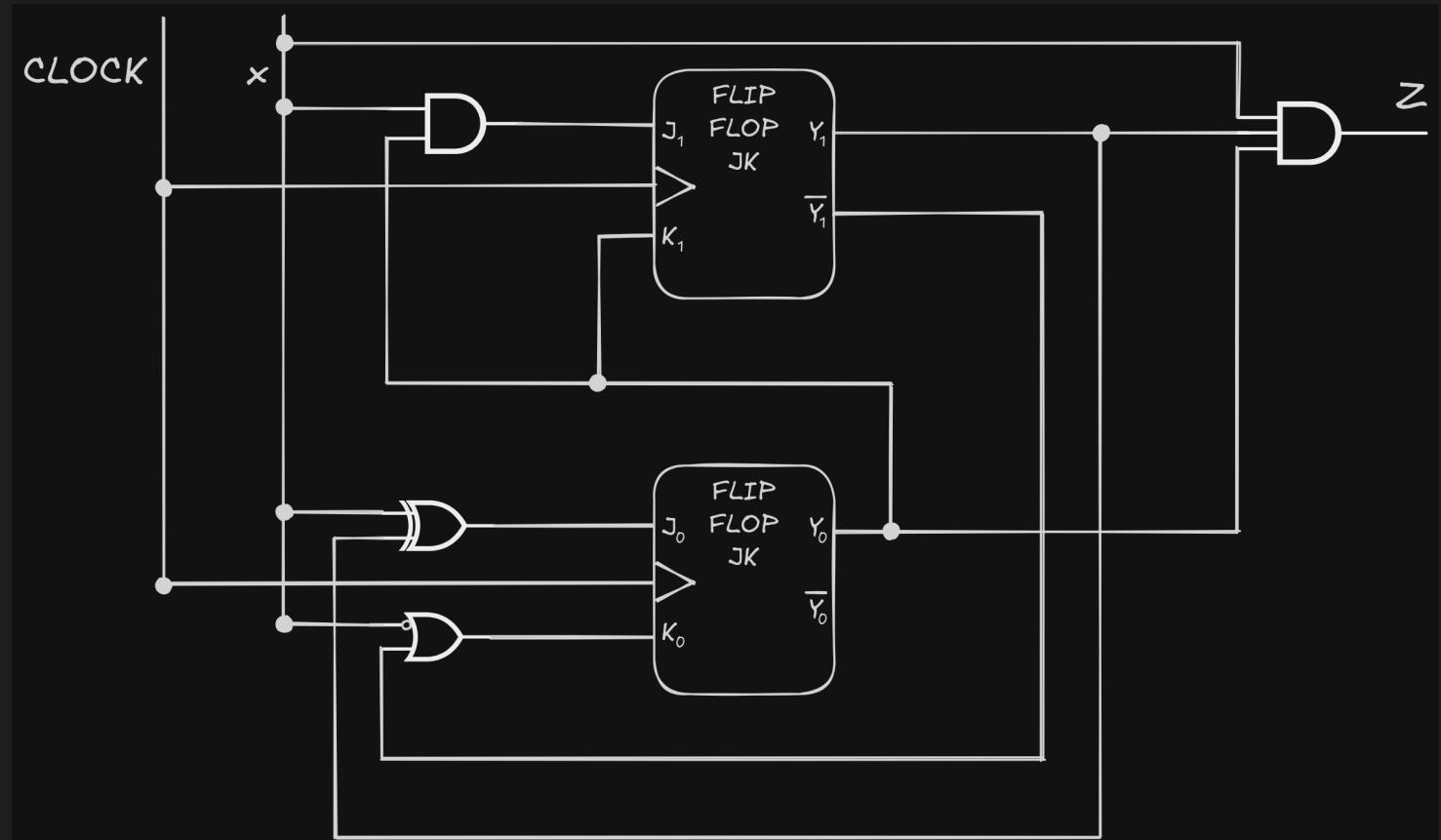
$$D_0 = \bar{x}\bar{y}_1 \bar{y}_0 + x\bar{y}_1 y_0 + x\bar{y}_1 y_0$$

- $D_1 = y_1 \bar{y}_0 + x\bar{y}_1 y_0$
- $D_0 = \bar{x}y_1 \bar{y}_0 + x\bar{y}_1 y_0 + xy_1 y_0$

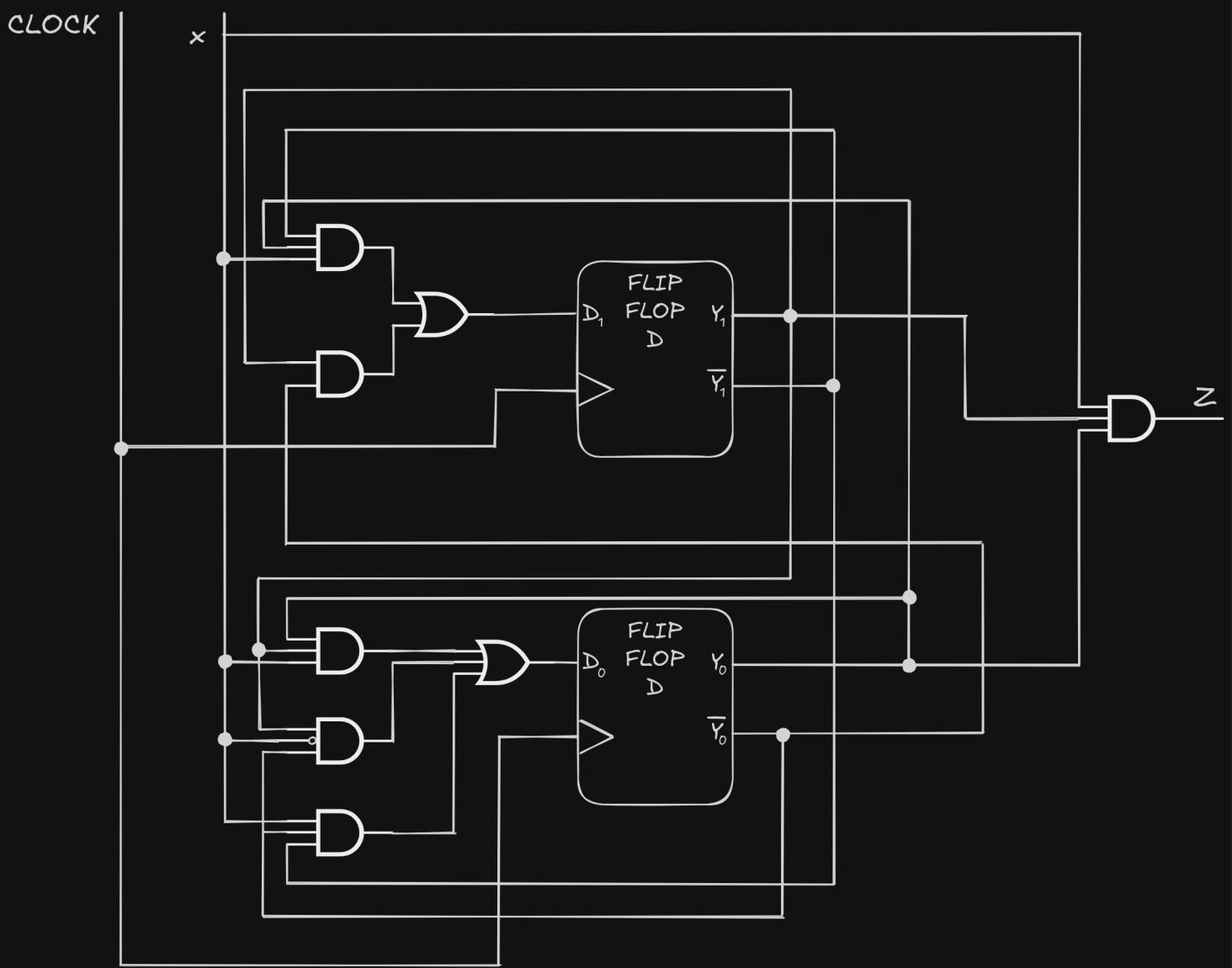
| Diagramma logico

Abbiamo tutto il necessario per disegnare il diagramma logico.

Con i due FF-JK:



Con i due FF-D:

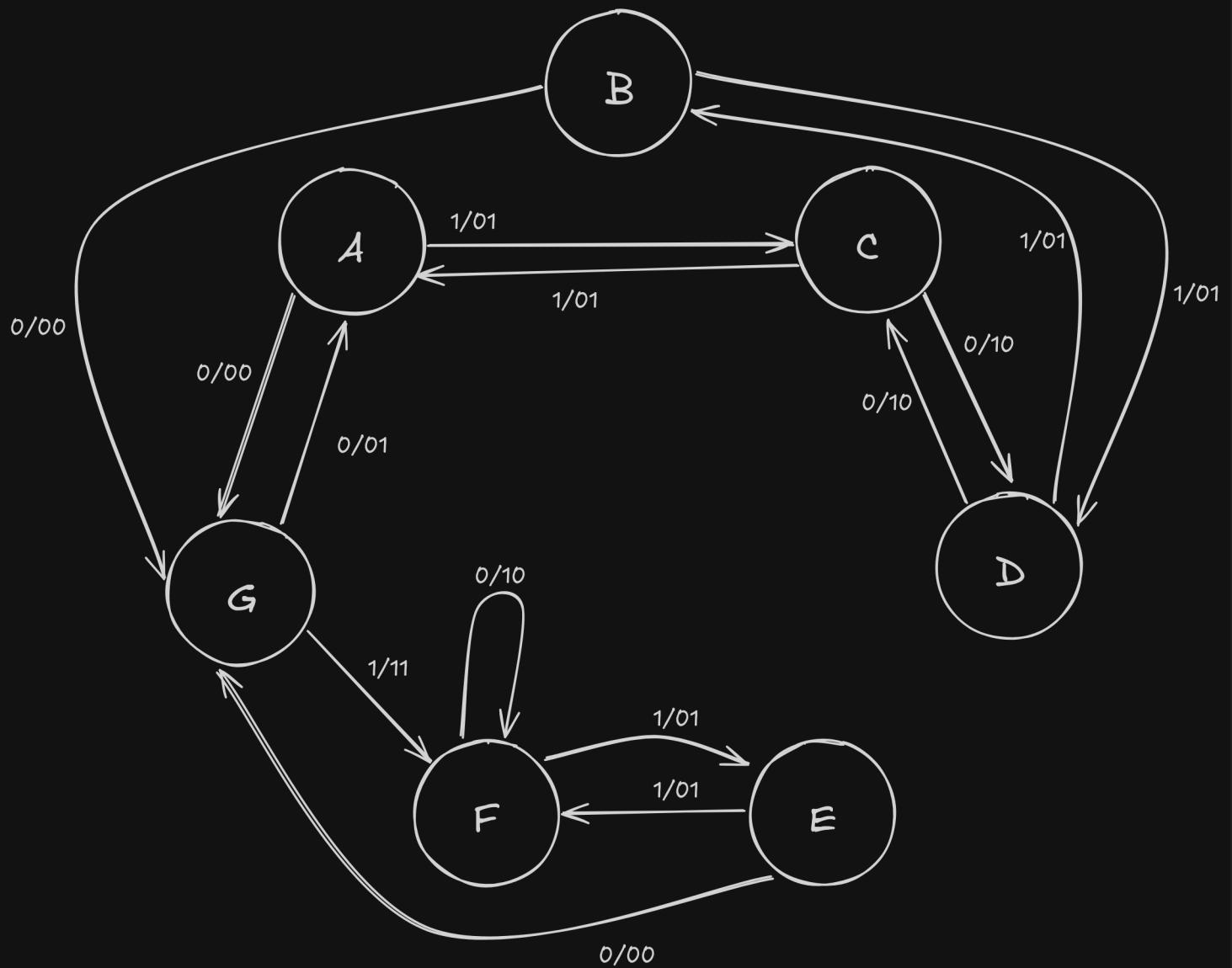


| Equivalenza tra stati e minimizzazione degli automi

La minimizzazione equivalente di un automa consiste nel semplificare un automa esistente affinché abbia il minor numero possibile di stati e transizioni, lasciando però il funzionamento equivalente (a fronte dello stesso input, deve dare lo stesso output).

Il vantaggio di minimizzare un automa è che se ci sono meno stati, allora servono anche meno componenti di memoria.

Dunque, procediamo a minimizzare questo automa di Mealy:



	0	1
A	$G/00$	$C/01$
B	$G/00$	$D/01$
C	$D/10$	$A/01$
D	$C/10$	$B/01$
E	$G/00$	$F/01$
F	$F/10$	$E/01$
G	$A/01$	$F/11$

Procedo a disegnare la *tabella triangolare*, dove confrontare gli stati tra di loro: è una tabella n stati \times n stati dove però manca

- La prima riga
- L'ultima colonna

- La diagonale principale

Questo perché non ci servono la proprietà riflessiva (evito di confrontare uno stato con se stesso) e simmetrica (per esempio confronto lo stato D e lo stato C in riga D e colonna C , ed evito di confrontarli di nuovo in riga C e colonna D). Avere la tabella del diagramma davanti rende più semplice riempire la tabella triangolare.

	B					
	C					
	D					
	E					
	F					
	G					
A		B	C	D	E	F

Per riempire la tabella, devo prima capire quando due stati sono indistinguibili tra di loro, e la risposta è che lo sono quando rispettano due condizioni:

- 1) Per tutte le combinazioni di input, ambi gli stati restituiscono lo stesso output (Per esempio se do 0 in input sia allo stato A che allo stato B , entrambi restituiscono lo stesso output, e la cosa deve avvenire anche se do in input 1, 00, 01, 10 o comunque qualsiasi combinazione accettata dagli stati)

2) Per tutte le combinazioni di ingresso, gli stati di destinazione devono a loro volta essere indistinguibili.

Dopo aver esaminato tutte le coppie di stati, gli stati S_1 ed S_2 possono essere:

- Distinguibili tra di loro
 - Perché esiste almeno una combinazione di ingresso per cui gli output sono diversi
 - Perché esiste una combinazione di ingresso che porta a stati di destinazione distinguibili
- Oppure indistinguibili tra di loro, perché per ogni ingresso dipendono da
 - Coppie di stati che abbiamo già verificato siano indistinguibili tra loro
 - Se stessi
 - Oppure dipendono da una coppia di stati di cui dobbiamo ancora provare la indistinguibilità

Adesso riempo la tabella, inserendo nelle celle

- Una croce se gli stati sono distinguibili, cioè, se hanno output diverso (se uso gli stati del diagramma di Moore) oppure se danno output diversi quando ricevono lo stesso input (se uso gli stati del diagramma di Mealy, come in questo caso)
- Le coppie di stati che devono essere a loro volta indistinguibili tra di loro per rendere la coppia in esame indistinguibile

B	CD					
C	X X	X X				
D	X X	X X	AB CD			
E	CF	DF	X	X X		
F	X X	X X	DF AE	CF BE	X X	
G	X X	X X	X X	X	X X	X X
	A	B	C	D	E	F

Dentro ogni cella, sono presenti le coordinate colonna-riga che devono essere indistinguibili per essere a loro volta considerati indistinguibili. Se seguendo le coordinate trovo delle altre coordinate, le seguo ricorsivamente.

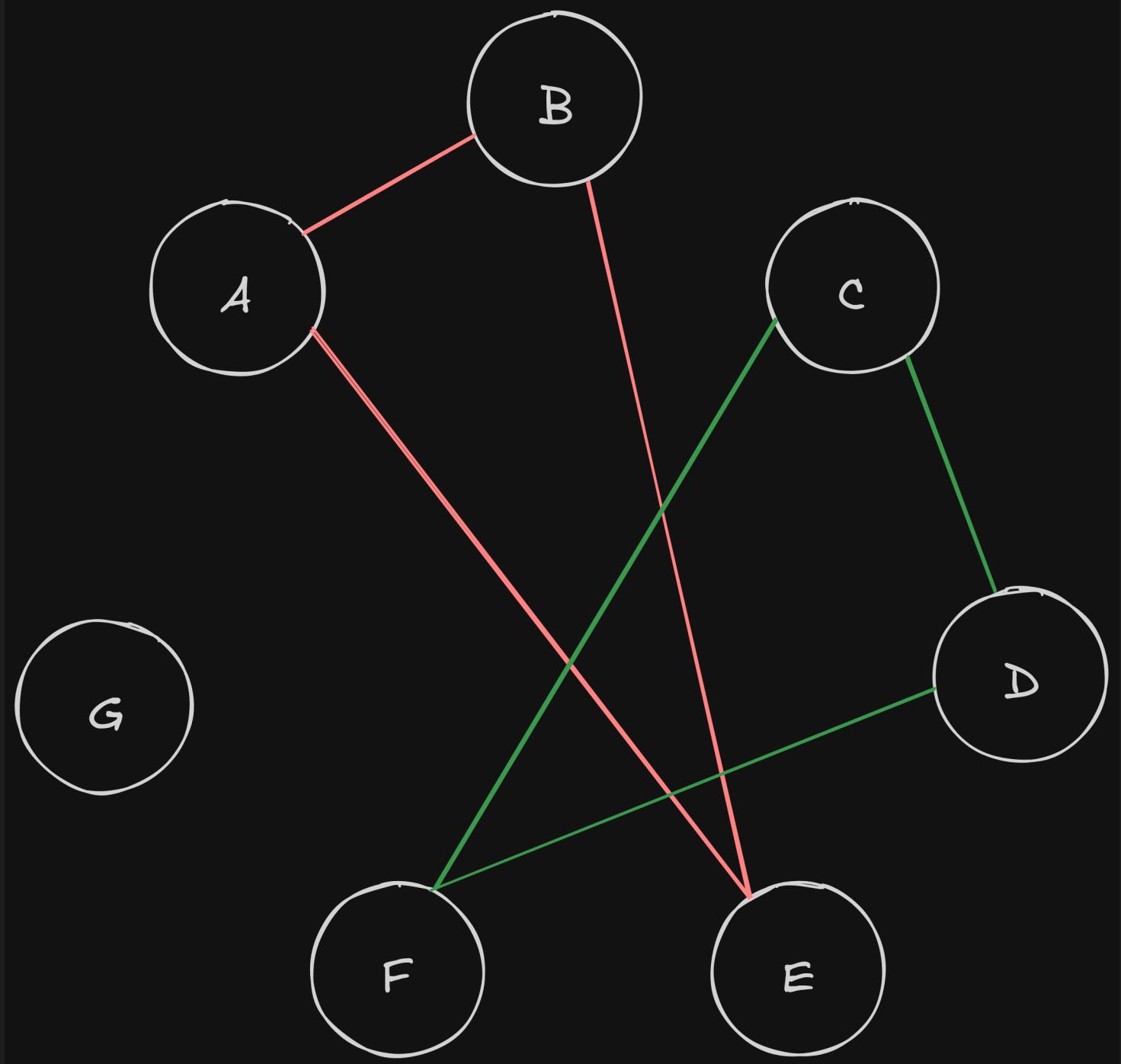
- Se infine giungo al punto di partenza, un punto già visitato o ad una cella che so già essere indistinguibile, allora la cella di partenza rappresenta coppie di stati indistinguibili.
- Se infine giungo ad una croce, allora la cella di partenza identifica due stati distinguibili e quindi posso metterci sopra una croce, dunque tutte le celle che a loro volta dipendono da quella cella sono distinguibili.

Applichiamo quindi questo ragionamento al nostro caso:

- La cella AB dipende da CD
 - CD dipende sia da se stessa, che da AB

- La dipendenza CD è soddisfatta perché dipende da se stessa
- Seguendo la dipendenza AB torniamo al punto di partenza, quindi tutte e due le dipendenze sono soddisfatte e possiamo dire che C e D sono indistinguibili.
- Abbiamo detto che CD è indistinguibile, quindi la dipendenza di AB è soddisfatta e possiamo dire che A e B sono indistinguibili.
- La cella AE dipende da CF
 - CF dipende sia da AE che da DF
 - Seguendo la dipendenza AE torniamo al punto di partenza, quindi possiamo considerare la prima dipendenza soddisfatta
 - DF dipende da CF e BE
 - Seguendo la dipendenza CF torno ad un punto già visitato, quindi possiamo considerare la prima dipendenza soddisfatta.
 - Seguendo la dipendenza BE torno ad un punto già visitato, quindi possiamo considerare la seconda dipendenza soddisfatta
 - Entrambe le dipendenze sono soddisfatte, quindi possiamo dire che D ed F sono indistinguibili e quindi la seconda dipendenza è soddisfatta.
 - Entrambe le dipendenze sono soddisfatte e quindi posso dire che C ed F sono indistinguibili.
- La dipendenza è soddisfatta, quindi possiamo dire che A ed E sono indistinguibili.
- La cella BE dipende da DF , che abbiamo già detto sono indistinguibili, quindi la dipendenza è soddisfatta e posso dire che B ed E sono indistinguibili.
- Abbiamo già detto che C e D sono indistinguibili.
- Abbiamo già detto che C e F sono indistinguibili.
- Abbiamo già detto che D e F sono indistinguibili.

Per chiarezza visiva, uniamo con lo stesso colore gli stati indistinguibili tra loro in un grafo equivalente:



Possiamo quindi procedere ad unire tutti gli stati indistinguibili tra di loro in unico stato:

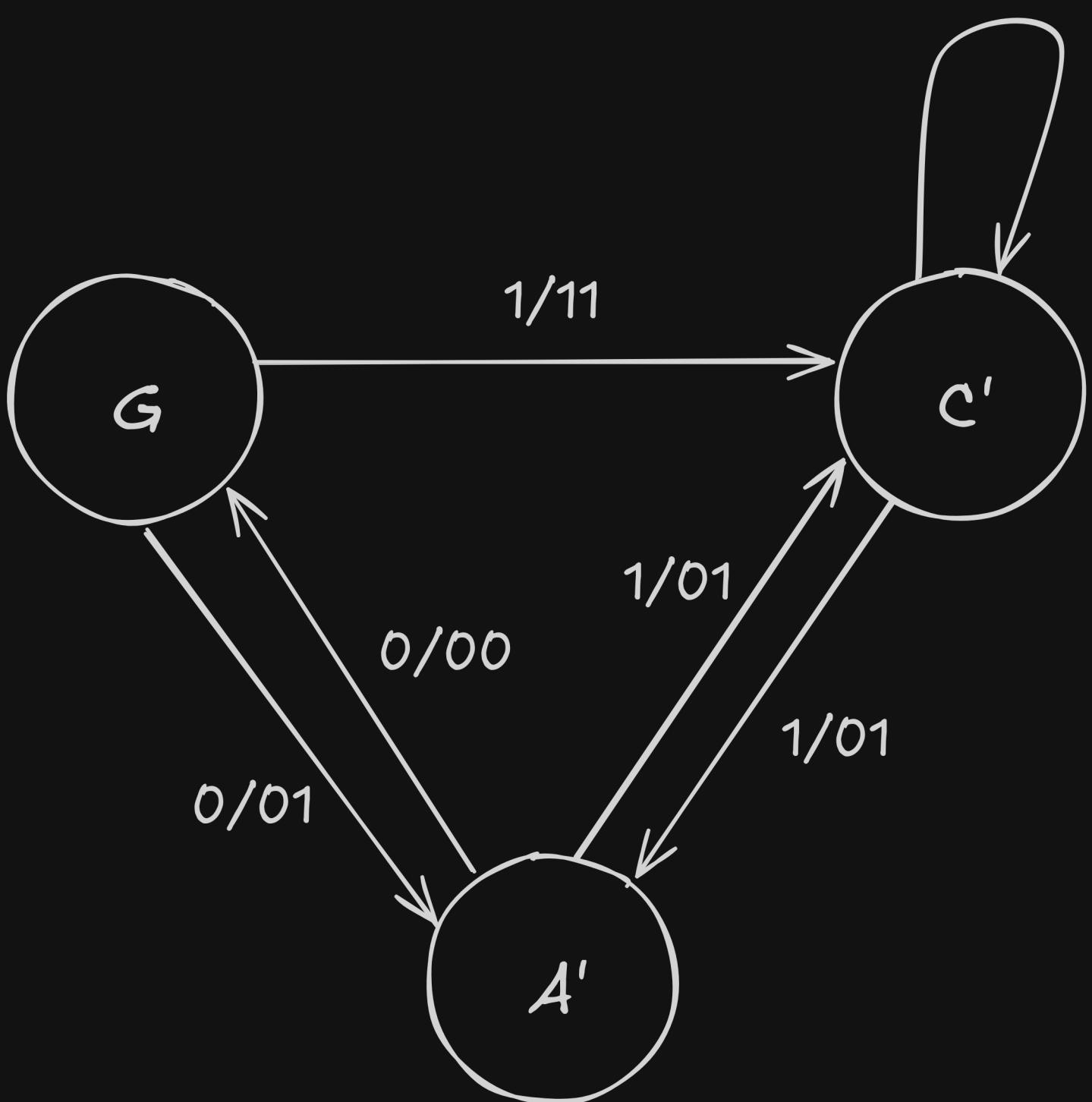
$$1^\circ : A' = \{A, B, E\}$$

$$2^\circ : C' = \{C, D, F\}$$

$$3^\circ : G$$

Abbiamo ridotto l'automa da 7 stati a 3.

Infine, disegniamo l'automa minimizzato e scriviamo la relativa tabella del diagramma. Per scrivere le transizioni per gli stati di partenza o destinazione che sono stati uniti, basta farle partire dallo stato minimizzato che contiene lo stato di partenza e farle arrivare allo stato minimizzato che contiene lo stato di destinazione, lasciando le stesse combinazioni input/output. Per esempio, nell'automa originale dando in ingresso **1** a *C*, si arrivava alla destinazione *A* con **01** in uscita, quindi nel diagramma minimizzato traccio una transizione che parte da *C'* (perché contiene *C*) ed arriva ad *A'* (perché contiene *A*) con input **1** ed output **01**.



	0	1
A'	G/00	C'/01
C'	C'/10	A'/01

-
- 1) Con *istanti discreti di tempo* si intende istanti di tempo campionati: è come i numeri naturali, dove ogni numero ha un successivo ben preciso, o un segnale digitale, dove il valore può essere 0 oppure 1 ↵
- 2) Con *istante continuo di tempo* si intende istanti di tempo precisi: è come i numeri razionali o reali, dove non è possibile definire un numero immediatamente successivo ad un altro, o un segnale analogico, dove tra un punto ed un altro del grafico ci possono essere infiniti valori. ↵