



Architettura degli Elaboratori I

Corso di Laurea Triennale in Informatica

Università degli Studi di Milano

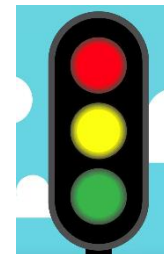
Dipartimento di Informatica "Giovanni Degli Antoni"

Edizione 2 (Cognomi H-Z), A.A. 2022-2023, Nicola.Basilico@unimi.it

Macchine a stati finiti (FSM)

Macchine a stati finiti (FSM)

- Macchine a Stati Finiti (Finite State Machines, Finite State Automata): è un modello matematico dell'elaborazione
- Si assume che l'elaborazione coincida con le operazioni fatte da una **macchina** che:
 - Riceve delle informazioni in **input** (i diversi possibili input sono in numero **finito**)
 - Ha uno **stato** interno (i diversi stati possibili sono in numero **finito**)
 - Emette delle informazioni in **uscita** (le diverse possibili uscite sono in numero **finito**)
- La macchina procede per passi in modo iterativo, ad ogni passo:
 - Calcola le uscite come funzione di input e stato corrente
 - Calcola un nuovo stato come funzione di input e stato corrente
 - Assume il nuovo stato appena calcolato come stato corrente
- Questo formalismo può essere usato per modellare sistemi in cui si svolgono elaborazioni che richiedono poca memoria, dove il concetto di stato raccoglie tutto quello che serve per portare avanti l'elaborazione



Cosa ci ricorda?

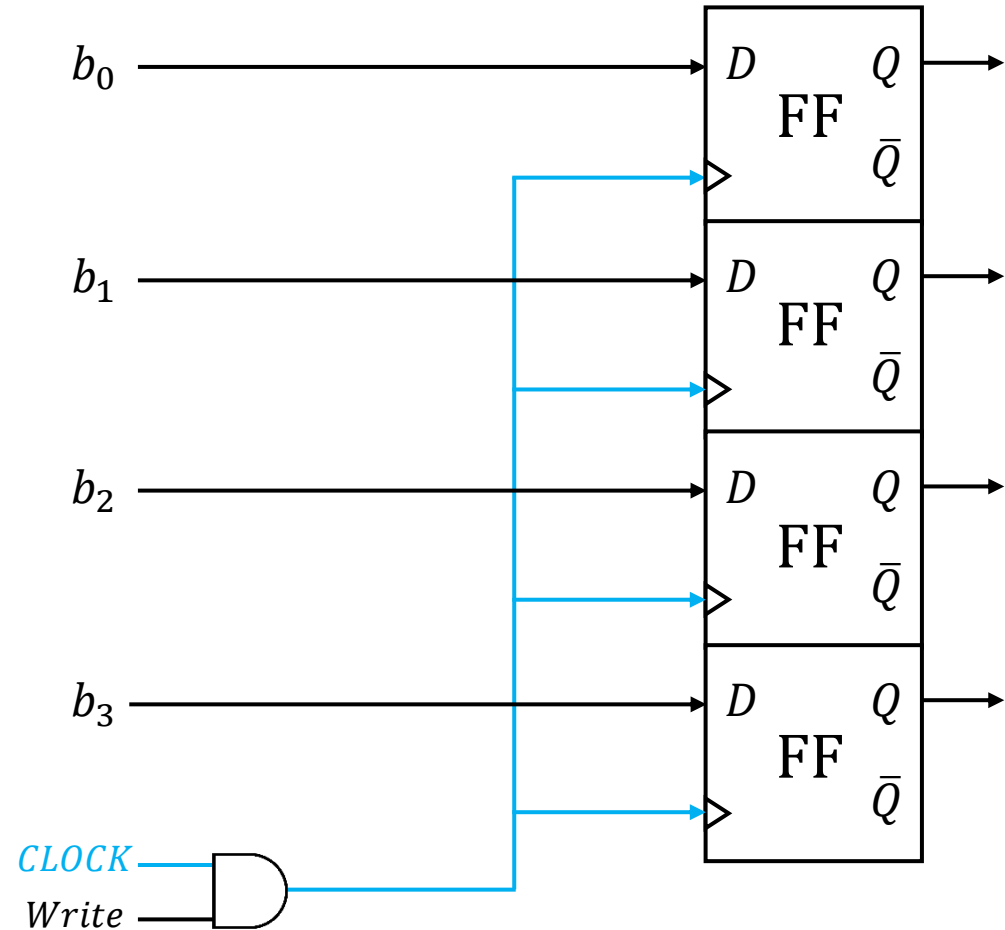
Macchine a stati finiti (FSM)

- Come definire circuiti sequenziali dove gli stati possibili possono essere più di 2?
- Abbandoniamo la logica di set/reset su cui si sono basati tutti i circuiti sequenziali elementari che abbiamo introdotto
- Possiamo mettere insieme più bistabili e rappresentare su n bit più di 2 stati, ma come gestire le transizioni?
- Per avere un circuito sequenziale con > 2 stati dobbiamo risolvere 2 problemi fondamentali:
 - **Ci serve più di un bit di memoria:** se il circuito può avere m stati, allora servirà una memoria con almeno $\lceil \log_2 m \rceil$ bit
 - **Ci serve un formalismo più astratto** per descrivere la dinamica del circuito (transizioni di stato e valori delle uscite)

Registri

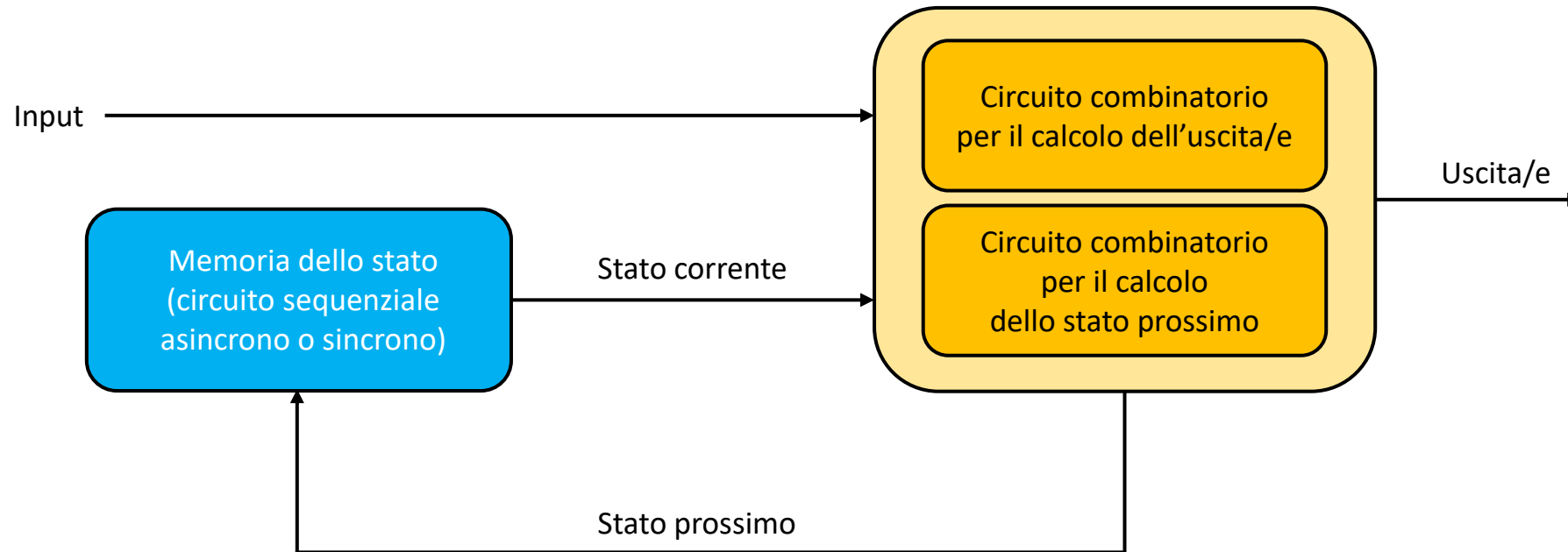
- Come si costruisce una memoria su più bit? Usando n Flip Flop, costruisco un registro da n bit

- Esempio:** registro su 4 bit con segnale di write (opzionale)
- Questo registro può contenere 2^4 diversi stati



Circuiti sequenziali

- Adottiamo questo schema generale per un circuito sequenziale



- **Esempio:** il bistabile SR può essere implementato come un caso particolare di questo schema in cui:
 - Input: s, r
 - Circuito per il calcolo delle uscite: lo stato Q (filo) e lo stato negato \bar{Q} (NOT)
 - Circuito per lo stato prossimo: $s + \bar{r}Q$
 - Memoria: 1 bit

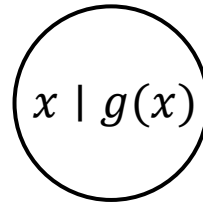
Formalizzazione della FSM

Dal punto di vista matematico, quali elementi definisco una FSM?

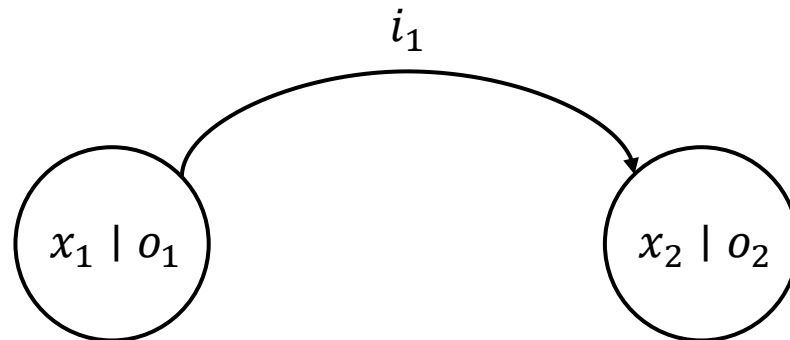
- $X = \{x_1, x_2, \dots, x_m\}$ insieme finito degli stati
- $x^i \in X$ stato iniziale della macchina
- $I = \{i_1, i_2, \dots, i_w\}$ insieme degli input che la macchina può ricevere
- $O = \{o_1, o_2, \dots, o_k\}$ insieme degli output che la macchina può generare
- $T: X \times I \rightarrow X$ funzione di transizione o di stato prossimo, dato lo stato corrente x e un input i , $T(x, i)$ rappresenta il prossimo stato della macchina
- $g()$ funzione di uscita:
 - $g: X \rightarrow O$ **macchina di Moore**: dato lo stato corrente x , $g(x)$ rappresenta l'uscita
 - $g: X \times I \rightarrow O$ macchina di Mealy: dato lo stato corrente x e l'input i , $g(x, i)$ rappresenta l'uscita

Grafo delle transizioni di stato

- Esiste un modo grafico più conveniente con cui rappresentare la FSM: **grafo orientato**
- Ogni nodo del grafo rappresenta uno stato x della FSM e l'uscita associata $g(x)$



- Ogni arco del grafo rappresenta una transizione tra due stati della macchina



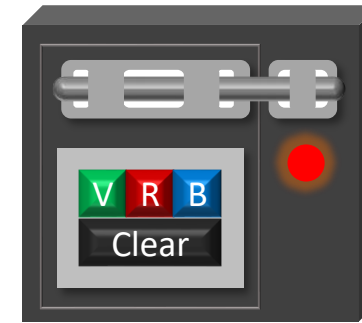
Dove:

- $g(x_1) = o_1, g(x_2) = o_2$
- $T(x_1, i_1) = x_2$

Esempio

Cassaforte con serratura digitale

- 4 pulsanti: V,R,B, Clear
- Led rosso: serratura chiusa
- Led verde: serratura aperta
- La sequenza V,R,B apre la cassaforte
- Il tasto clear
 - Se la cassaforte è chiusa resetta la pulsantiera
 - Se la cassaforte è aperta è l'unico modo per chiuderla (la pressione di altri tasti mentre la cassaforte è aperta non comporta nulla)



Problema: come formalizzare la FSM che implementa la logica di questa cassaforte?

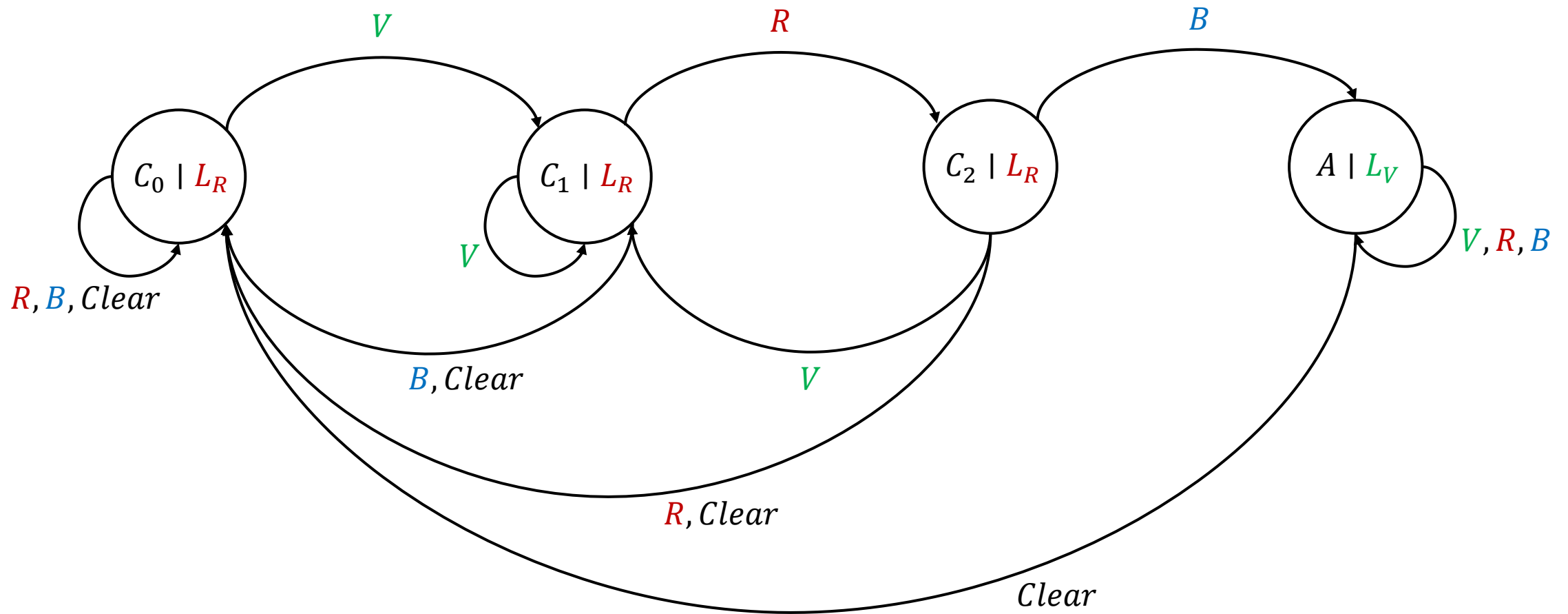
Esempio

Definiamo il modello matematico che serve per descrivere questa FSM

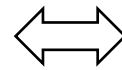
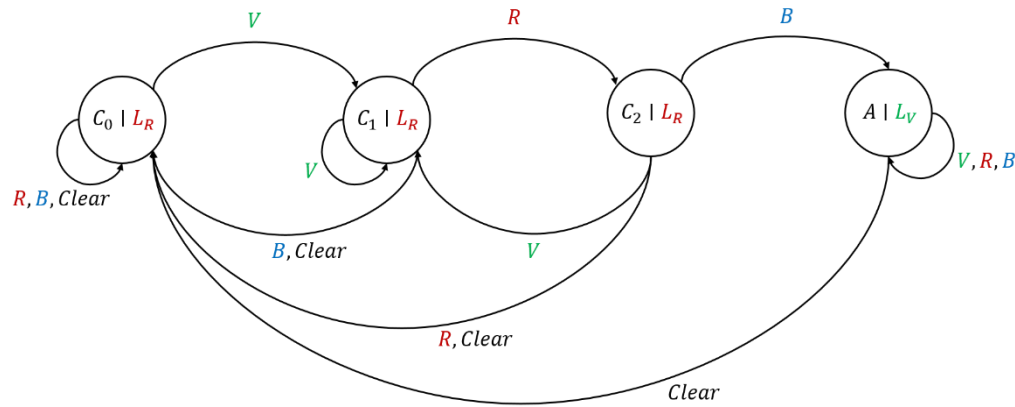
- Insieme degli stati $X = \{C_0, C_1, C_2, A\}$
 - Stato C_i serratura chiusa e ho riconosciuto i primi i simboli della combinazione
 - Stato A , serratura aperta
- Stato iniziale della macchina $C_0 \in X$
- Insieme degli input $I = \{V, R, B, Clear\}$
- Insieme degli output $O = \{L_R, L_V\}$
 - L_R led rosso
 - L_V verde
- Funzione di uscita $g(C_i) = L_R \ \forall i \in \{0,1,2\}, g(A) = L_V$

Esempio (grafo delle transizioni)

- Funzione di transizione



Esempio (tabella delle transizioni)



	V	R	B	$Clear$	Uscita
C_0	C_1	C_0	C_0	C_0	L_R
C_1	C_1	C_2	C_0	C_0	L_R
C_2	C_1	C_0	A	C_0	L_R
A	A	A	A	C_0	L_V

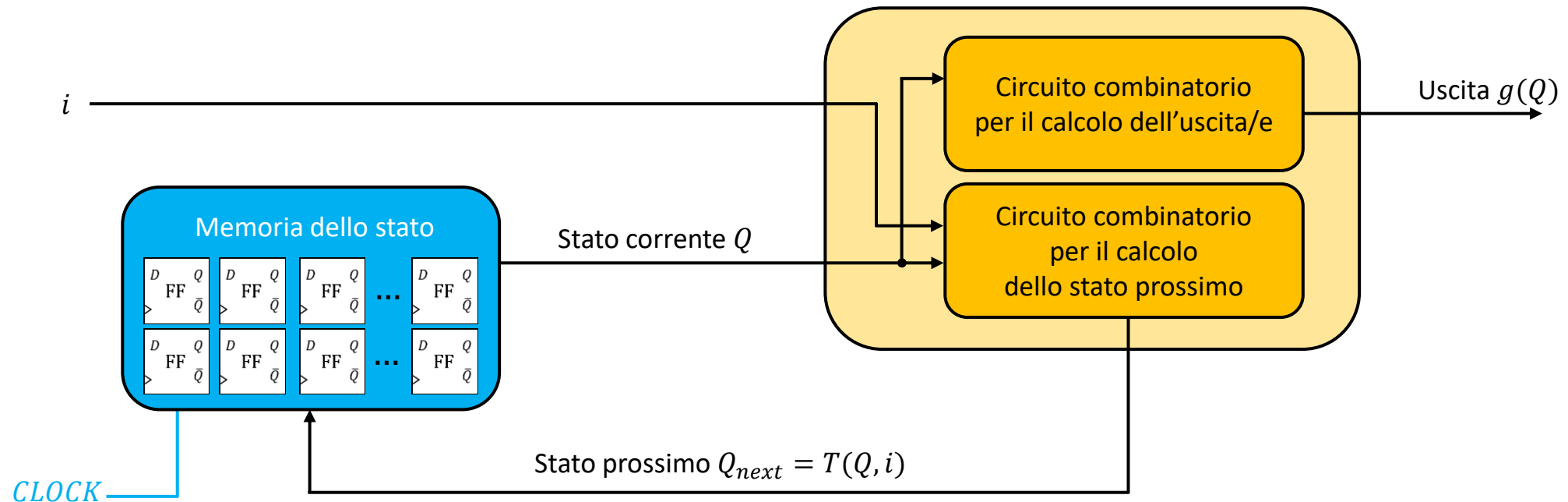
- La tabella delle transizioni riporta in forma tabellare la stessa specifica descritta dal grafo
- Per ogni stato corrente Q (sulla riga) e input i (su una colonna, esclusa l'ultima) viene riportato lo stato successivo $Q_{next} = T(Q, i)$
- L'ultima colonna riporta, per ogni stato Q , il valore dell'uscita $g(Q)$ (ricorda: nella macchina di Moore l'uscita dipende solo dallo stato corrente)

Sintesi di una FSM

- Problema: sintetizzare un circuito che implementa una macchina a stati finiti la cui specifica è descritta dal formalismo precedente
- In sostanza, una volta che abbiamo definito la specifica della macchina, **come è fatto** e **come si costruisce** il circuito che la implementa?
- **Cosa vuol dire che il circuito sequenziale implementa la specifica della FSM?**
 - Ha una memoria interna in cui mantiene un valore che rappresenta lo stato corrente. Se la FSM può avere m stati allora la memoria deve avere almeno $\lceil \log_2 m \rceil$ bit (useremo un Flip Flop per ogni bit)
 - Se in memoria abbiamo il valore che rappresenta lo stato Q e il circuito riceve un segnale che rappresenta l'input i , la memoria viene sovrascritta con il valore che rappresenta lo stato $T(Q, i)$ e l'uscita viene posta a $g(Q)$

Architettura del circuito di una FSM

- Cominciamo dalla prima domanda: **come è fatto il circuito che implementa la FSM?**
- Useremo lo schema introdotto in precedenza, che ora possiamo descrivere in modo più specifico:

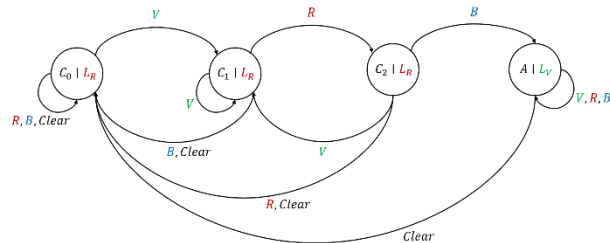


- Cosa ci serve per costruire un circuito di questo tipo?
 1. Sapere da quanti Flip Flop è costituita la memoria dello stato
 2. Sapere come è fatto il circuito per il calcolo dell'uscita
 3. Sapere come è fatto il circuito per il calcolo dello stato prossimo

Sintesi di una FSM

- Seconda domanda: **come si costruisce?**

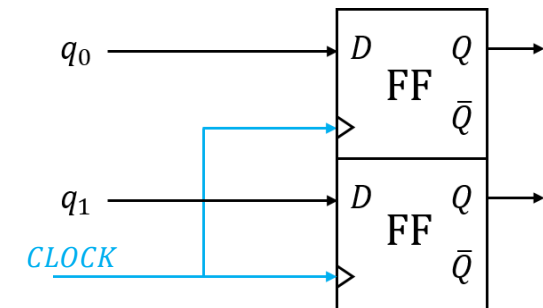
- Insieme degli stati $X = \{C_0, C_1, C_2, A\}$
 - Stato C_i serratura chiusa e ho riconosciuto i primi i simboli della combinazione
 - Stato A , serratura aperta
- stato iniziale della macchina $C_0 \in X$
- insieme degli input $I = \{V, R, B, Clear\}$
- insieme degli output $O = \{L_R, L_V\}$
 - L_R led rosso
 - L_V verde
- funzione di uscita $g(C_i) = L_R \forall i \in \{0,1,2\}, g(A) = L_V$



	V	R	B	Clear	Uscita
C_0	C_1	C_0	C_0	C_0	L_R
C_1	C_1	C_2	C_0	C_0	L_R
C_2	C_1	C_0	A	C_0	L_R
A	A	A	A	C_0	L_V

- Primo step:** ogni stato deve essere codificato in binario e memorizzato
- Dobbiamo **associare ad ogni stato una rappresentazione binaria**
- Ad esempio codifico lo stato C_0 con 00, lo stato C_1 con 01, etc. ... e determino quanti bit di memoria mi servono

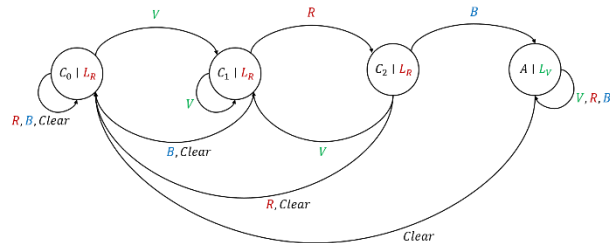
Stato Q	Rappresentazione binaria	
	q_1	q_0
C_0	0	0
C_1	0	1
C_2	1	0
A	1	1



- q_i è l' i -esimo bit di stato
- Ci servirà un banco di memoria (registro) da 2 Flip Flop

Sintesi di una FSM

- Insieme degli stati $X = \{C_0, C_1, C_2, A\}$
 - Stato C_i serratura chiusa e ho riconosciuto i primi i simboli della combinazione
 - Stato A , serratura aperta
- stato iniziale della macchina $C_0 \in X$
- insieme degli input $I = \{V, R, B, Clear\}$
- insieme degli output $O = \{L_R, L_V\}$
 - L_R led rosso
 - L_V verde
- funzione di uscita $g(C_i) = L_R \forall i \in \{0,1,2\}, g(A) = L_V$



	V	R	B	Clear	Uscita
C_0	C_1	C_0	C_0	C_0	L_R
C_1	C_1	C_2	C_0	C_0	L_R
C_2	C_1	C_0	A	C_0	L_R
A	A	A	A	C_0	L_V

- **Secondo step:** input ed uscite devono essere codificati in binario
- Dobbiamo associare ad ogni simbolo di input e di uscita una rappresentazione binaria
- Ad esempio l'input V con 00, l'input R con 01, etc. ... l'uscita L_R con 0 e l'uscita L_V con 1

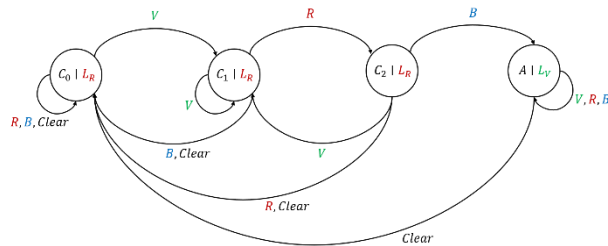
Input i	Rappresentazione binaria	
	i_1	i_0
V	0	0
R	0	1
B	1	0
$Clear$	1	1

Uscita	Rappresentazione binaria
	y_0
L_R	0
L_V	1

- i_k ora è il k -esimo bit di input, mentre y_k è il k -esimo bit di uscita
- Per questi segnali non serve memoria, non rappresentano lo stato

Sintesi di una FSM

- Insieme degli stati $X = \{C_0, C_1, C_2, A\}$
 - Stato C_i serratura chiusa e ho riconosciuto i primi i simboli della combinazione
 - Stato A , serratura aperta
- stato iniziale della macchina $C_0 \in X$
- insieme degli input $I = \{V, R, B, Clear\}$
- insieme degli output $O = \{L_R, L_V\}$
 - L_R led rosso
 - L_V verde
- funzione di uscita $g(C_i) = L_R \forall i \in \{0,1,2\}, g(A) = L_V$



	V	R	B	Clear	Uscita
C_0	C_1	C_0	C_0	C_0	L_R
C_1	C_1	C_2	C_0	C_0	L_R
C_2	C_1	C_0	A	C_0	L_R
A	A	A	A	C_0	L_V

Stato Q	Rappresentazione binaria	
	q_1	q_0
C_0	0	0
C_1	0	1
C_2	1	0
A	1	1

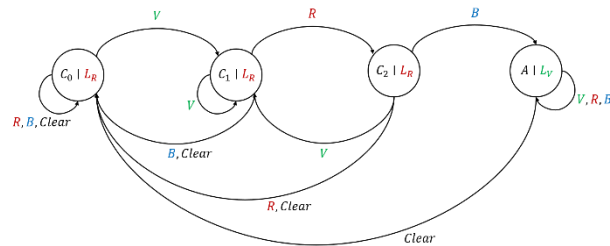
Input i	Rappresentazione binaria	
	i_1	i_0
V	0	0
R	0	1
B	1	0
Clear	1	1

Uscita	Rappresentazione binaria	
	y_0	
L_R	0	
L_V	1	

- **Terzo step:** sintetizzare l'espressione Booleana di ciascun bit di stato prossimo e ciascun bit di uscita come se fossero delle funzioni logiche indipendenti
- Implementiamo il circuito combinatorio che calcola le transizioni e le uscite
- Per fare questo dobbiamo riscrivere la tabella delle transizioni utilizzando le codifiche binarie che abbiamo scelto anziché i nomi astratti
- Cominciamo con la tabella di verità per queste funzioni logiche (bit di stato prossimo, bit di uscita)

Sintesi di una FSM

- Insieme degli stati $X = \{C_0, C_1, C_2, A\}$
 - Stato C_i serratura chiusa e ho riconosciuto i primi i simboli della combinazione
 - Stato A , serratura aperta
- stato iniziale della macchina $C_0 \in X$
- insieme degli input $I = \{V, R, B, Clear\}$
- insieme degli output $O = \{L_R, L_V\}$
 - L_R led rosso
 - L_V verde
- funzione di uscita $g(C_i) = L_R \forall i \in \{0,1,2\}, g(A) = L_V$



	V	R	B	Clear	Uscita
C ₀	C ₁	C ₀	C ₀	C ₀	L _R
C ₁	C ₁	C ₂	C ₀	C ₀	L _R
C ₂	C ₁	C ₀	A	C ₀	L _R
A	A	A	A	C ₀	L _V

Stato Q	Rappresentazione binaria	
	q ₁	q ₀
C ₀	0	0
C ₁	0	1
C ₂	1	0
A	1	1

Input i	Rappresentazione binaria	
	i ₁	i ₀
V	0	0
R	0	1
B	1	0
Clear	1	1

Uscita	Rappresentazione binaria
	y ₀
L _R	0
L _V	1

q ₁	q ₀	i ₁	i ₀	q ₁ ^{next}	q ₀ ^{next}	y ₀
0	0	0	0	0	1	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	1	0
1	0	0	1	0	0	0
1	0	1	0	1	1	0
1	0	1	1	0	0	0
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	0	0	1

- Costruiamo la SOP di ogni bit di stato e di uscita identificando i **mintermini**:

$$q_0^{next} = \bar{q}_1 \bar{q}_0 \bar{i}_1 \bar{i}_0 + \bar{q}_1 q_0 \bar{i}_1 \bar{i}_0 + q_1 \bar{q}_0 \bar{i}_1 \bar{i}_0 + q_1 \bar{q}_0 i_1 \bar{i}_0 + q_1 q_0 \bar{i}_1 \bar{i}_0 + q_1 q_0 \bar{i}_1 i_0 + q_1 q_0 i_1 \bar{i}_0$$

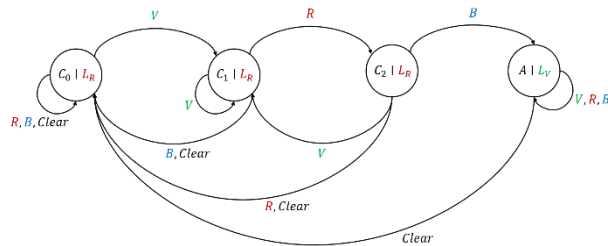
$$q_1^{next} = \bar{q}_1 q_0 \bar{i}_1 i_0 + q_1 \bar{q}_0 i_1 \bar{i}_0 + q_1 q_0 \bar{i}_1 \bar{i}_0 + q_1 q_0 \bar{i}_1 i_0 + q_1 q_0 i_1 \bar{i}_0$$

$$y_0 = q_1 q_0$$

- Abbiamo sintetizzato le espressioni Booleane e quindi i circuiti combinatori che implementano transizione di stato e calcolo dell'uscita!

Sintesi di una FSM

- Insieme degli stati $X = \{C_0, C_1, C_2, A\}$
 - Stato C_i serratura chiusa e ho riconosciuto i primi i simboli della combinazione
 - Stato A , serratura aperta
- stato iniziale della macchina $C_0 \in X$
- insieme degli input $I = \{V, R, B, Clear\}$
- insieme degli output $O = \{L_R, L_V\}$
 - L_R led rosso
 - L_V verde
- funzione di uscita $g(C_i) = L_R \forall i \in \{0,1,2\}$, $g(A) = L_V$



	V	R	B	Clear	Uscita
C_0	C_1	C_0	C_0	C_0	L_R
C_1	C_1	C_2	C_0	C_0	L_R
C_2	C_1	C_0	A	C_0	L_R
A	A	A	A	C_0	L_V

Stato Q	Rappresentazione binaria	
	q_1	q_0
C_0	0	0
C_1	0	1
C_2	1	0
A	1	1

Input i	Rappresentazione binaria	
	i_1	i_0
V	0	0
R	0	1
B	1	0
Clear	1	1

Uscita	Rappresentazione binaria
	y_0
L_R	0
L_V	1

- Prima di costruire il circuito finale, ripetiamo il procedimento di sintesi utilizzando la tabella delle transizioni di stato al posto della tabella di verità
- Questo procedimento è di norma più efficiente perché ci permette di applicare il metodo delle mappe di Karnaugh
- Riporto **una tabella per ogni funzione logica** (tralascio l'uscita che, in questo esempio, è triviale)

		$i_1 i_0$			
		00	01	11	10
$q_1 q_0$	00	0	0	0	0
	01	0	1	0	0
	11	1	1	0	1
	10	0	0	0	1
		Bit di stato q_1^{next}			

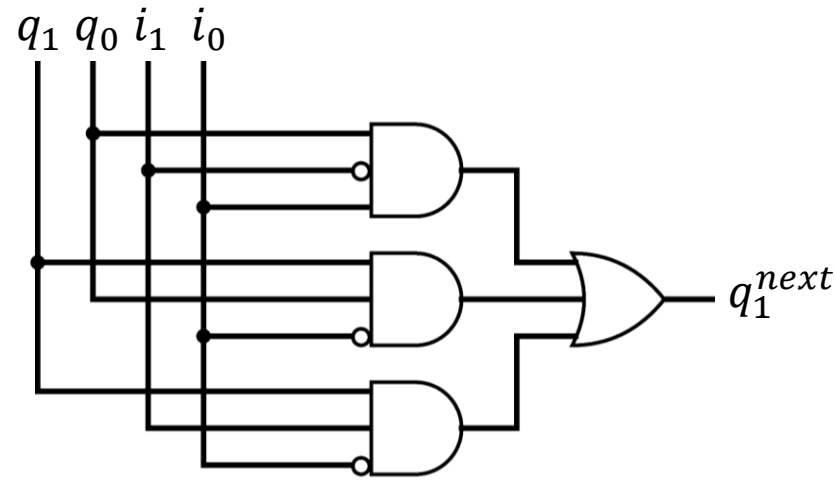
		$i_1 i_0$			
		00	01	11	10
$q_1 q_0$	00	1	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	0	0	1
		Bit di stato q_0^{next}			

- $q_1^{next} = q_0 \bar{i}_1 i_0 + q_1 q_0 \bar{i}_0 + q_1 i_1 \bar{i}_0$
- $q_0^{next} = \bar{i}_1 \bar{i}_0 + q_1 q_0 \bar{i}_1 + q_1 \bar{i}_0$
- $y_0 = q_1 q_0$ (la stessa di prima)

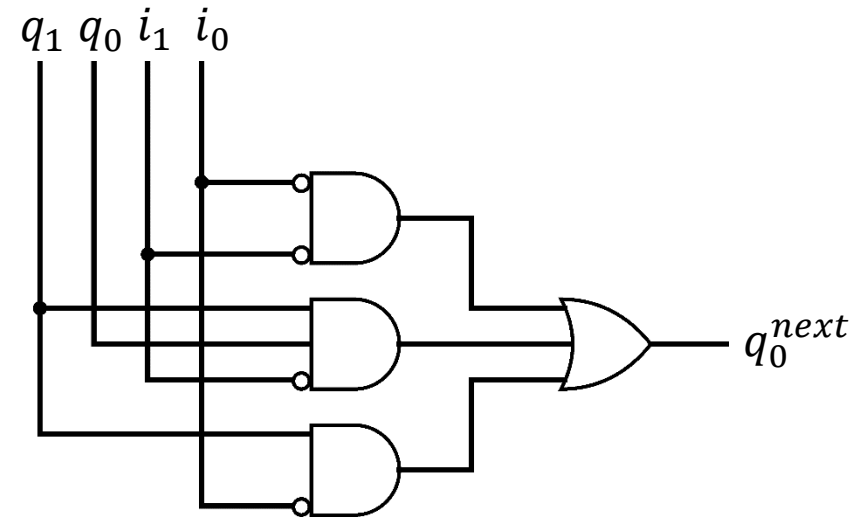
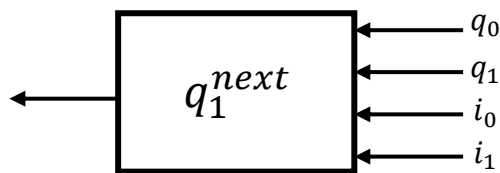
È tutto quello che ci serve da ora!

Sintesi di una FSM

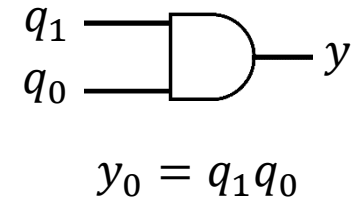
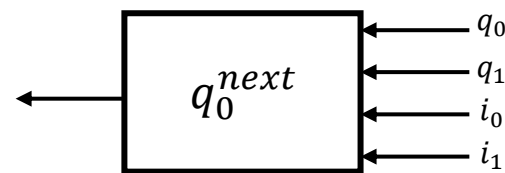
- **Quarto step:** sintetizzare i circuiti combinatori
- «no-brainer»: abbiamo già tutte le espressioni logiche



$$q_1^{next} = q_0 \bar{i}_1 i_0 + q_1 q_0 \bar{i}_0 + q_1 i_1 \bar{i}_0$$



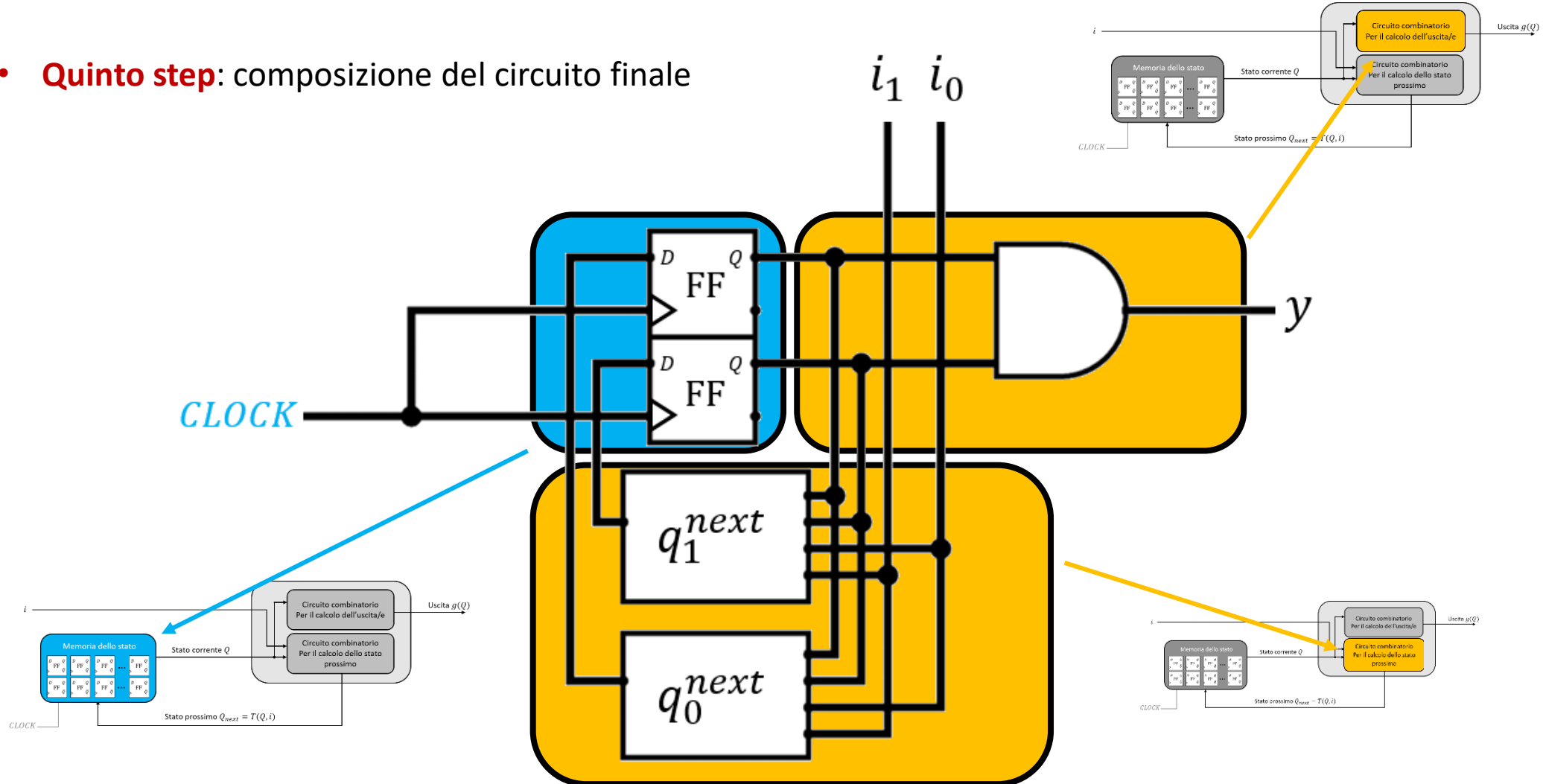
$$q_0^{next} = \bar{i}_1 \bar{i}_0 + q_1 q_0 \bar{i}_1 + q_1 \bar{i}_0$$



- Nella rappresentazione esterna dei circuiti di transizione di stato dispongo input e output in modo conveniente per come li andrò ad usare nel circuito finale

Sintesi di una FSM

- **Quinto step:** composizione del circuito finale



Recap

1. Definizione delle specifiche: grafo e tabella delle transizioni
2. Codifica binaria degli stati, dimensionamento della memoria (Flip Flop)
3. Codifica binaria dei simboli di input
4. Codifica binaria dei simboli di output (uscita)
5. Sintesi dell'espressione Booleana per ogni bit di stato prossimo e di uscita
6. Sintesi dei circuiti combinatori che calcolano ogni bit di stato prossimo e uscita
7. Collegamento dei circuiti combinatori e della memoria secondo l'architettura sincrona della FSM

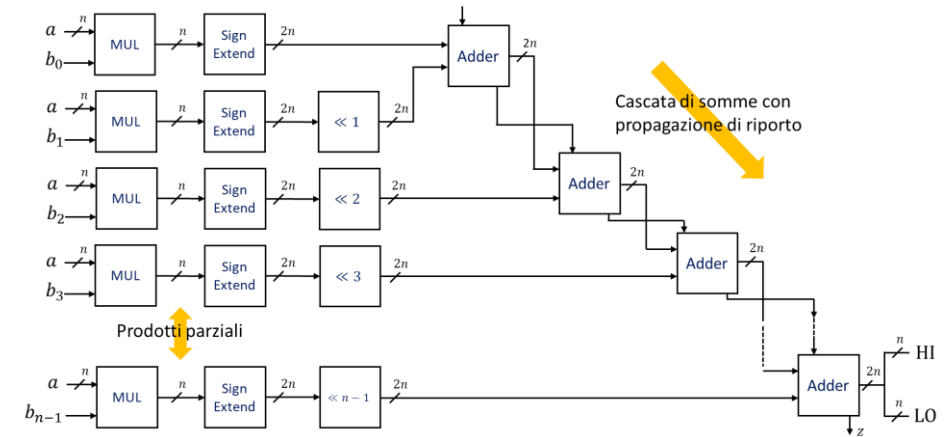
Esercizi

1. Si progetti una macchina a stati finiti che modelli il comportamento di una lampadina a tre modalità. Libretto di istruzioni: la lampadina ha due tasti "on" e "off". Per accendere la lampadina (luce fissa) si preme "on". Per spegnerla, si preme "off". Per ottenere l'effetto lampeggiante (acceso/spento intermittente) si premano "on" e "off" contemporaneamente.
2. Si progetti una FSM che riceva un solo bit in ingresso e che ponga la sua uscita ad 1 ogni qual volta riconosca la sequenza 101
3. Si progetti una FSM caratterizzata da un bit di input e due bit di uscita y_0 e y_1 . Ogni volta che viene riconosciuta la sequenza «011», y_0 cambia valore mentre l'uscita y_1 , normalmente a 0, va ad 1 per un ciclo di clock e poi torna a 0. Si suppongano inizialmente ingressi e uscite a 0.

Approccio firmware

Operazioni aritmetiche

- Moltiplicatore visto nelle lezioni precedenti: circuito combinatorio che, letti a e b su n bit, produce in uscita il valore del prodotto su $2n$
- Segue un **approccio combinatorio**: l'elaborazione procede stadio dopo stadio dove ogni stadio corrisponde ad una sotto-operazione (ad es. calcolo dei prodotti parziali) della moltiplicazione.
- **Approccio sequenziale**: costruire una FSM che
 - legge due sequenze a e b di n bit un bit per volta
 - una volta terminato di leggerle, si trova in uno stato finale la cui uscita è il prodotto ab ;
- Ora l'elaborazione procede per step, ognuno è una transizione della FSM



- Nel primo caso abbiamo costruito un singolo componente (moltiplicatore) che, in un singolo ciclo di clock, svolge un'operazione complessa (la moltiplicazione)
- Nel secondo caso la moltiplicazione è svolta come una sequenza di operazioni più semplici, su più cicli di clock e attraversando stati intermedi. (La sequenza corrisponde ai passi del procedimento della moltiplicazione, ovvero il suo algoritmo.)
- L'approccio combinatorio costa di più, ma performa meglio: è consigliato per operazioni che vengono svolte spesso (legge di Ahmdal)
- L'approccio sequenziale è meno performante, ma meno costoso e più flessibile (la FSM può essere pensata come un piccolo elaboratore che esegue un firmware)
- Come si progetta una FSM del genere? Vediamo due casi: **moltiplicazione** e **divisione**.

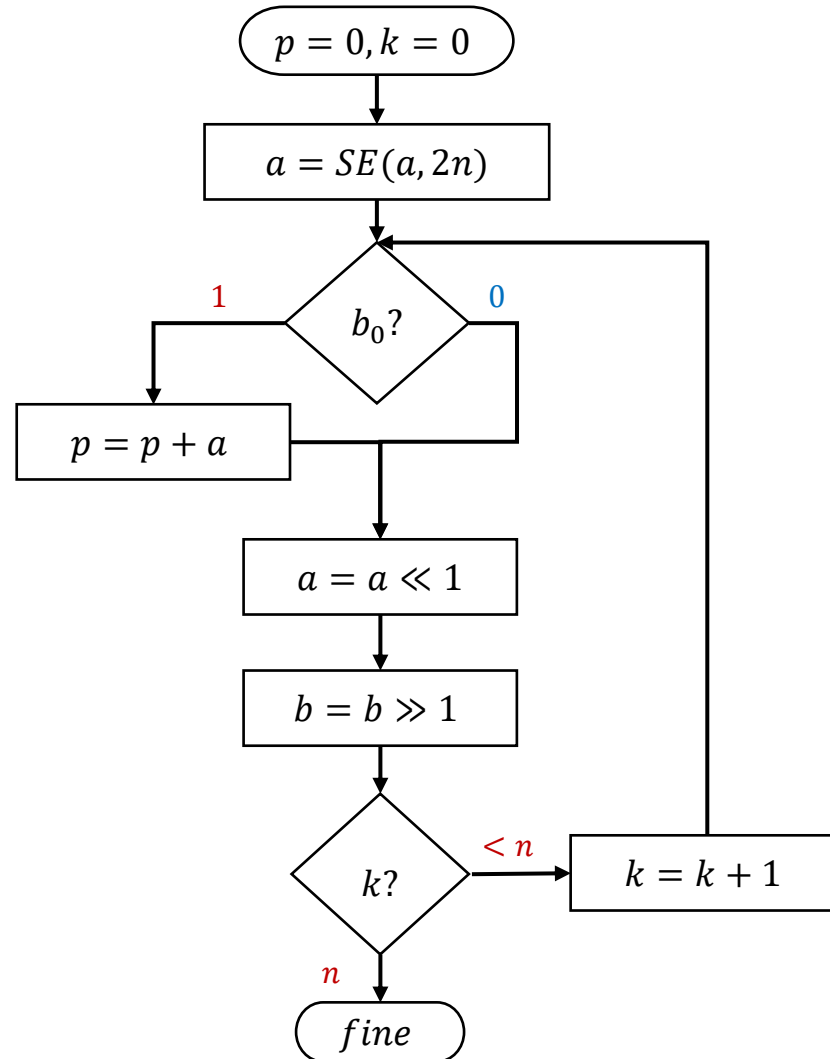
Approccio sequenziale alla moltiplicazione $a \times b$

Algoritmo della moltiplicazione

- a (su n bit) è il moltiplicando
- b (su n bit) è il moltiplicatore
- p (su $2n$ bit) contiene, ad ogni iterazione, la somma dei prodotti parziali. Alla fine conterrà il prodotto $a \times b$
- k è il contatore delle iterazioni

$n = 5$

a	10111	\times
b	00101	$=$
<hr/>		
	111	
$k = 0$	10111	
$k = 1$	00000	
$k = 2$	10111	
$k = 3$	00000	
$k = 4$	00000	
<hr/>		
$p = a \times b$	001110011	



1. Inizializzazione assegno zero a p (su $2n$ bit) e k e estendo il segno del moltiplicando, in modo da averlo su $2n$ bit così da poterlo sommare al risultato parziale in p

2. Controllo il bit più a destra del moltiplicatore, se vale 1 aggiungo il moltiplicando a al risultato parziale p

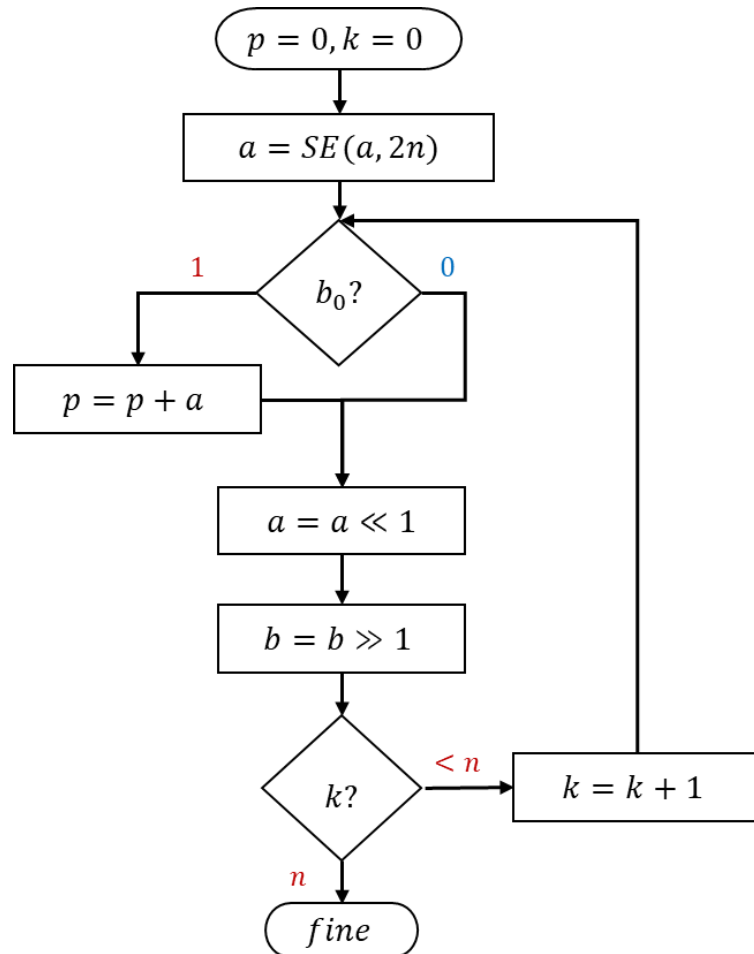
3. Shift a sinistra del moltiplicando per l'eventuale prossima somma parziale

4. Shift a destra del moltiplicatore: il prossimo bit da considerare nello step 2 diventa quello più a destra in b

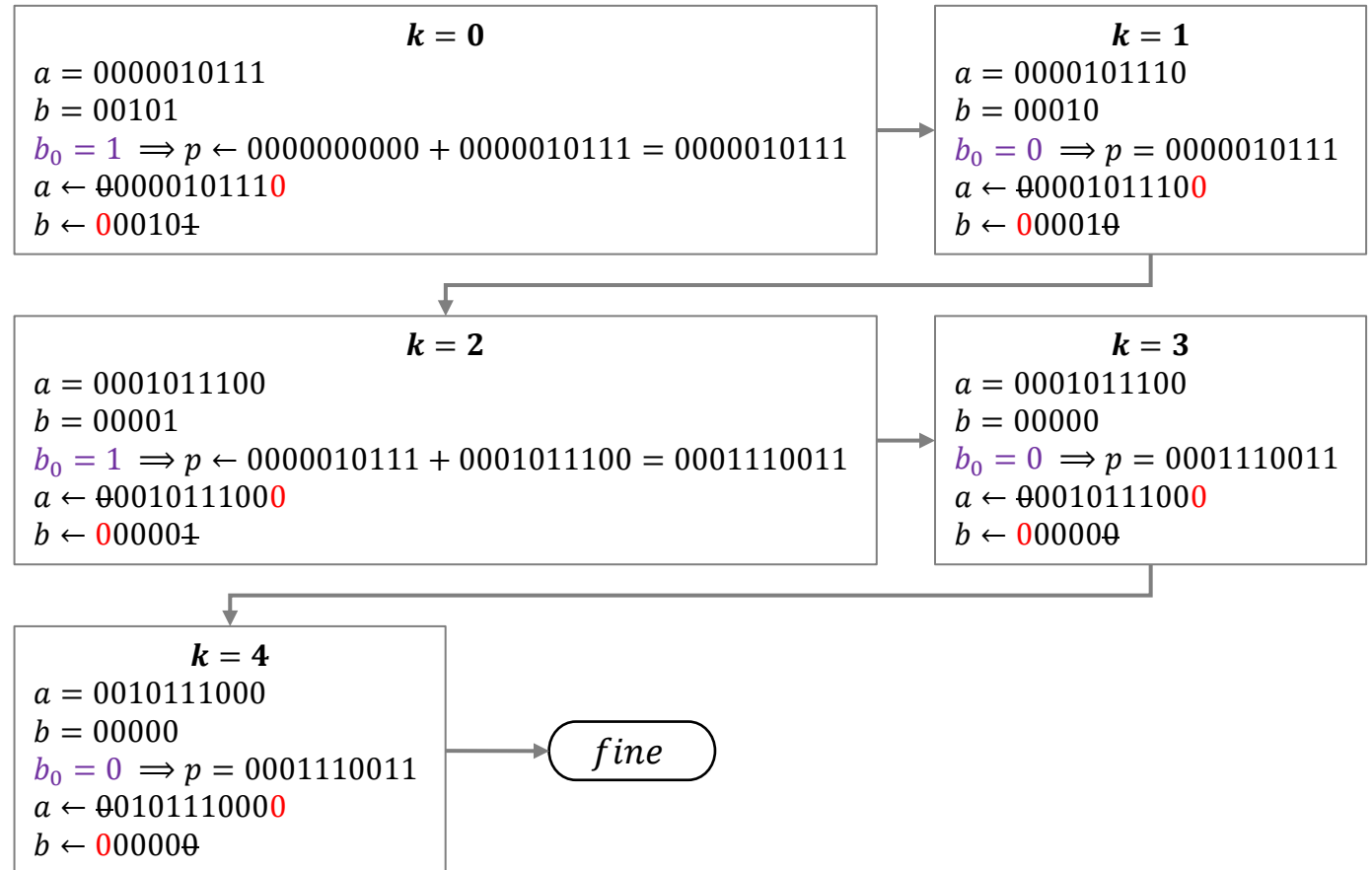
5. Termina una volta considerati tutti i bit del moltiplicatore

Approccio sequenziale alla moltiplicazione $a \times b$

Esempio: $a = 10111, b = 00101$ ($n = 5$)

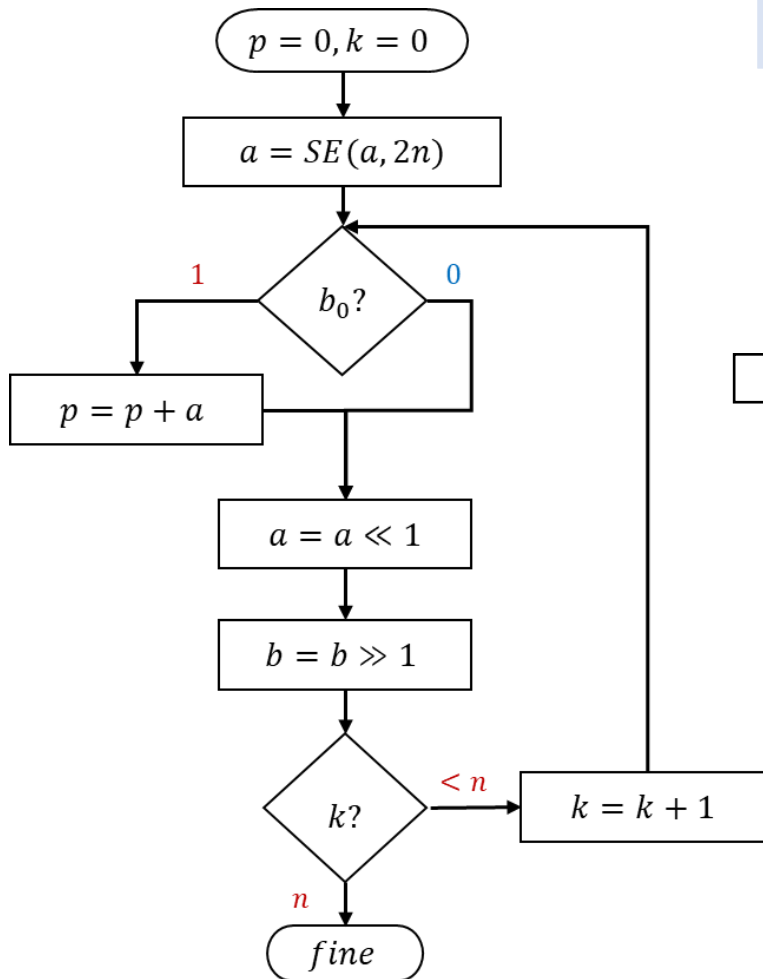


Inizializzazione: $p \leftarrow 0000000000, k \leftarrow 0, a \leftarrow 0000010111$



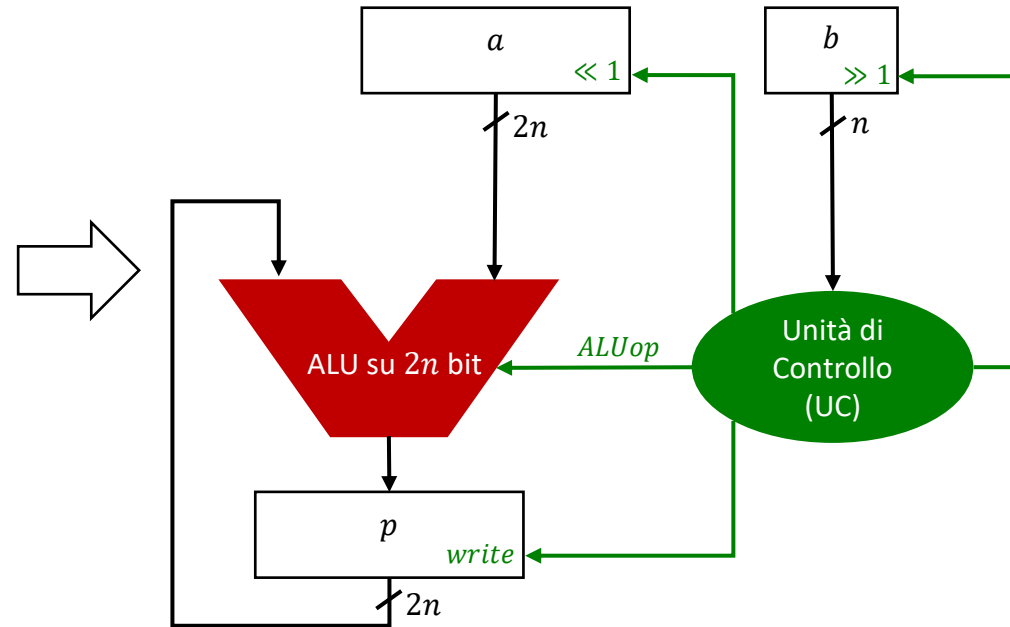
Approccio sequenziale alla moltiplicazione $a \times b$

- Implementazione



Facciamo uso di registri con una porta per ricevere un comando che agisce sul valore correntemente contenuto nel registro

registro
comando



L'unità di controllo è una FSM

- Input: b
- Output: 2 segnali di comando per shiftare e un segnale di selezione per l'operazione della ALU (somma)
- Sato: rappresenta il contesto dell'iterazione corrente nell'algoritmo della moltiplicazione

Ad ogni ciclo di clock la UC:

- controlla b_0 e, a seconda del suo valore, decide se aggiornare p con la somma fatta dalla ALU (chiede alla ALU una somma tramite $ALUop$ e mette $write$ a 1)
- Shift di a e b a sinistra e destra, rispettivamente (mettendo a 1 i rispettivi comandi)