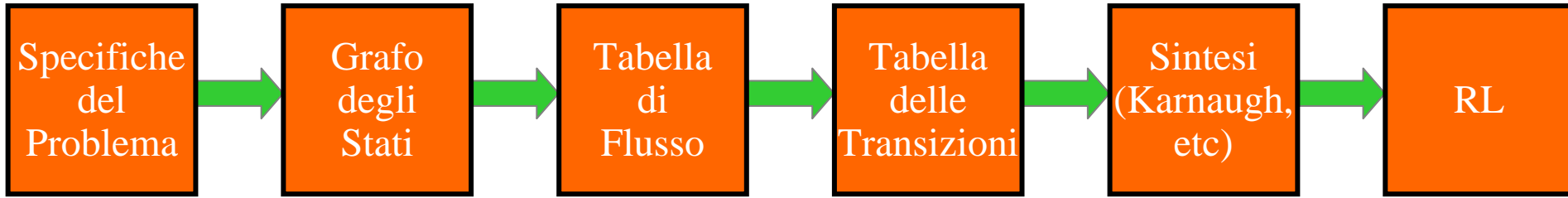


Calcolatori
Elettronici LA

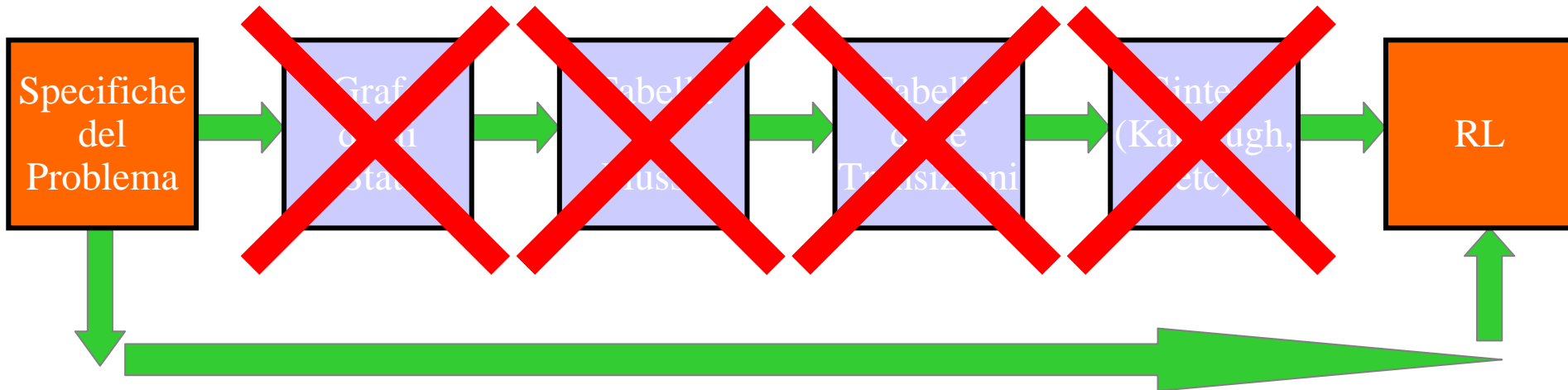
Complementi ed Esercizi
di
Reti Logiche

Introduzione

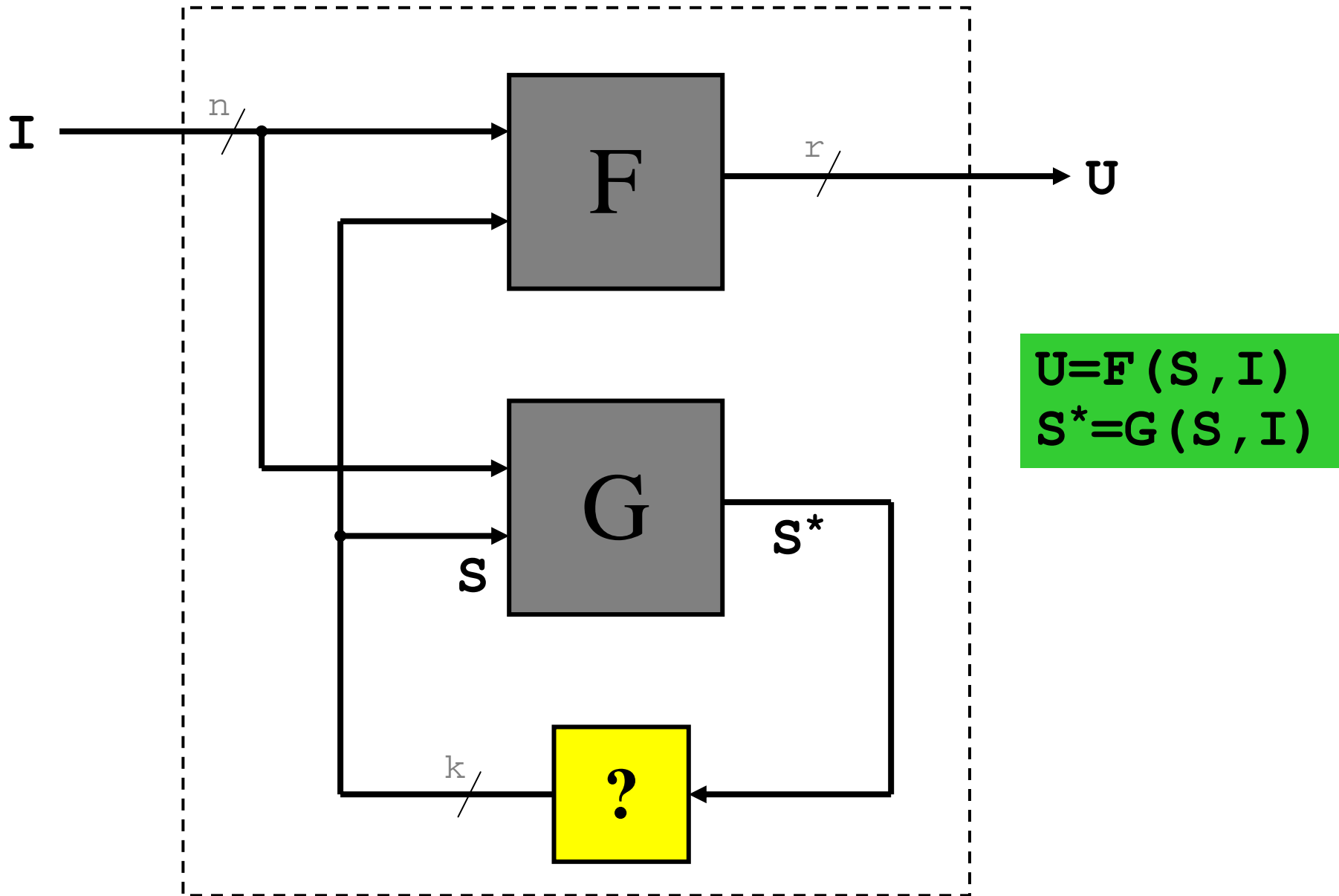
Reti Logiche: sintesi mediante approccio "formale"



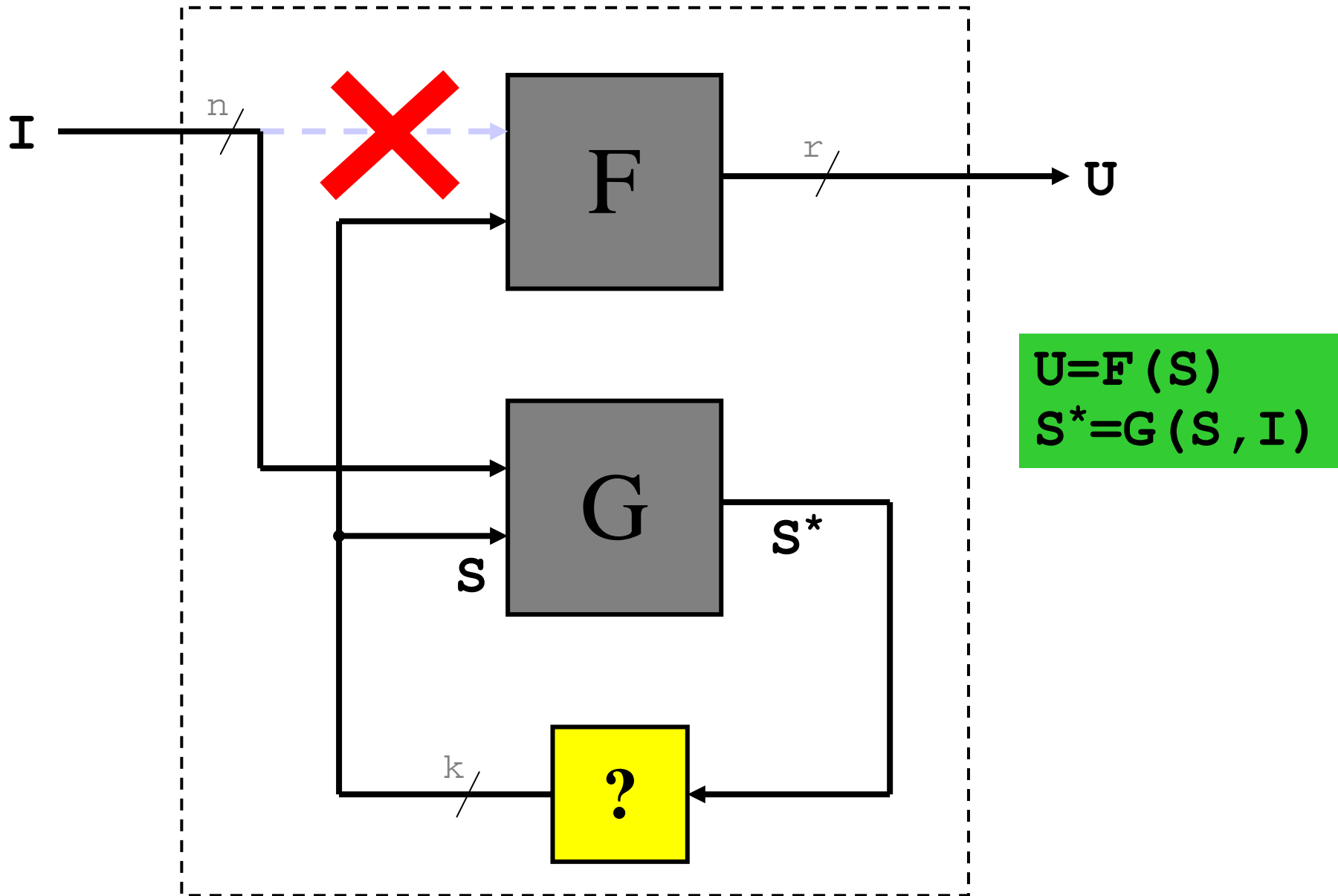
Calcolatori Elettronici: sintesi mediante approccio "diretto"



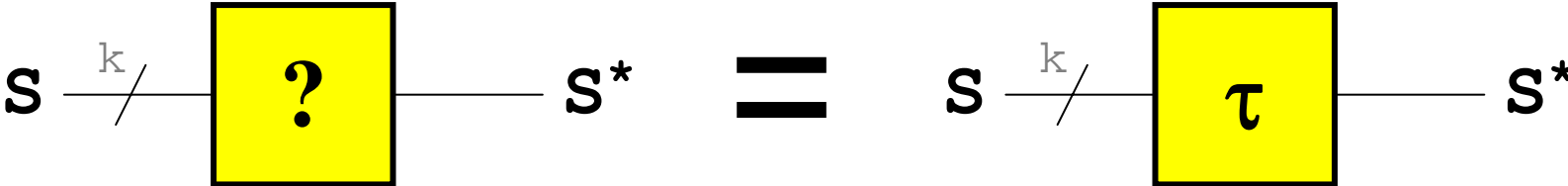
Modello della Macchina a Stati Finiti (FSM) - Mealy



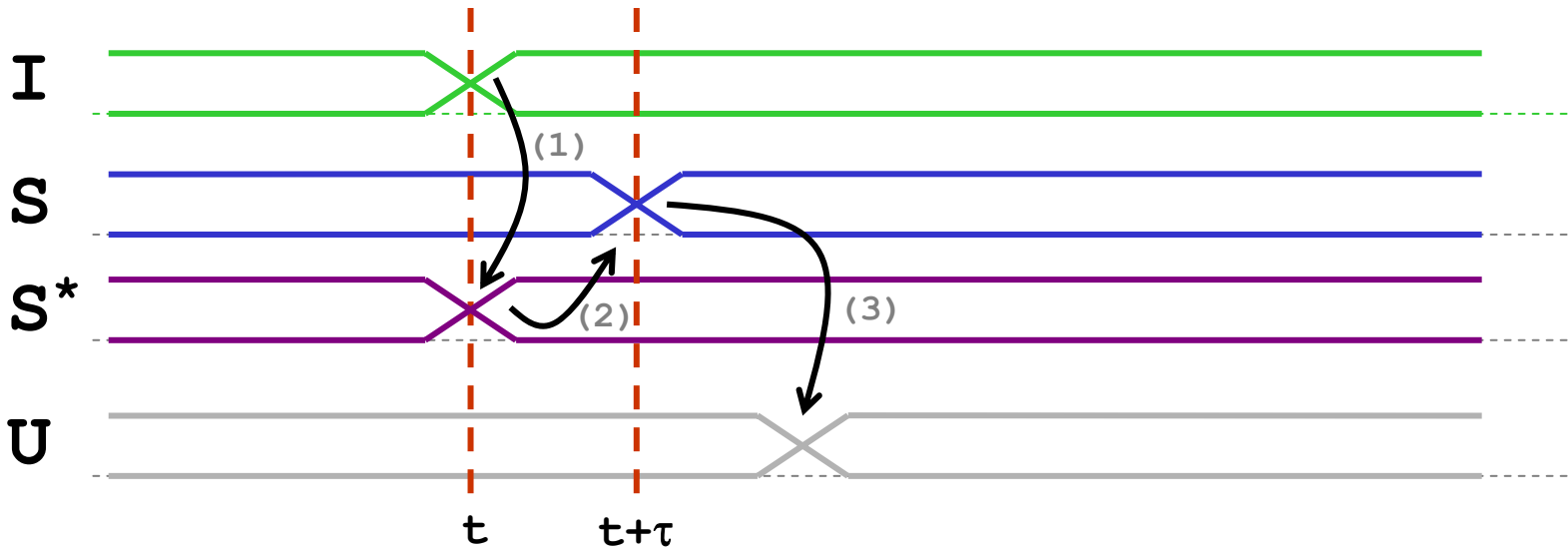
Modello della Macchina a Stati Finiti (FSM) - Moore



Reti Sequenziali Asincrone (RSA)



Retroazione diretta
 (τ : ritardo intrinseco della RC G)

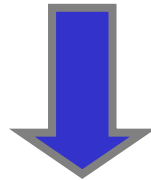


Aspetti positivi delle RSA (vs RSS)

- Le reti asincrone rispondono molto rapidamente (appena possibile) alle variazioni degli ingressi
- Non è necessario un segnale di sincronismo (clock)
- Ridotta dissipazione di potenza

Aspetti negativi delle RSA (vs RSS)

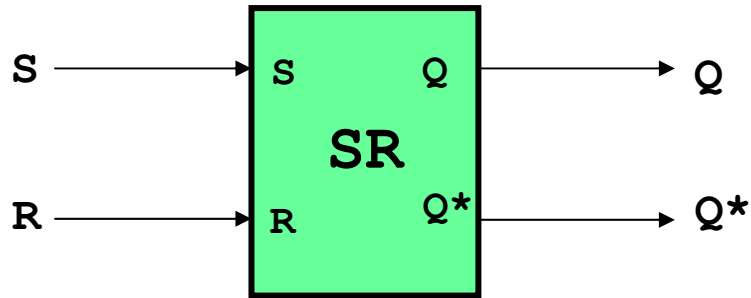
- Vincoli per il corretto impiego
 - l'ingresso può variare solo quando la rete ha raggiunto una condizione di stabilità
 - i segnali di ingresso possono variare uno alla volta
- Esposte a potenziali malfunzionamenti (corse critiche)
- Difficili da progettare



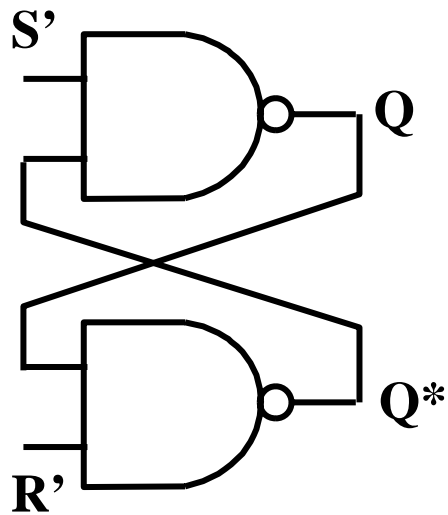
In pratica, sono utilizzate per realizzare latch e flip-flop.

A noi interessano (maggiormente) le reti sincrone (RSS) !

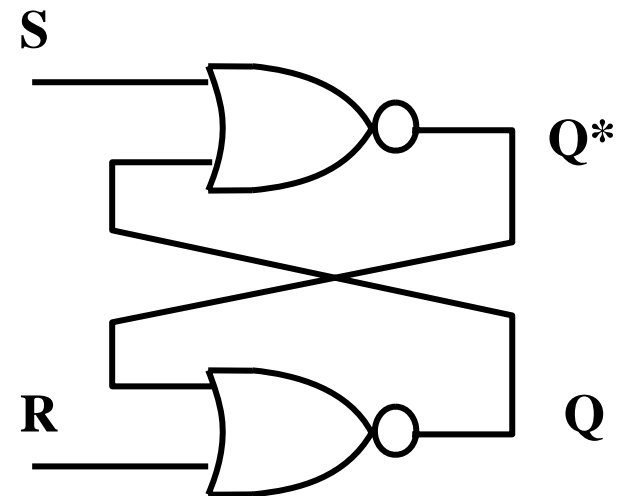
RSA notevoli: Latch SR



S	R	Q	Q*
0	0	Q	Q*
0	1	0	1
1	0	1	0
<u>1</u>	<u>1</u>	<u> </u>	



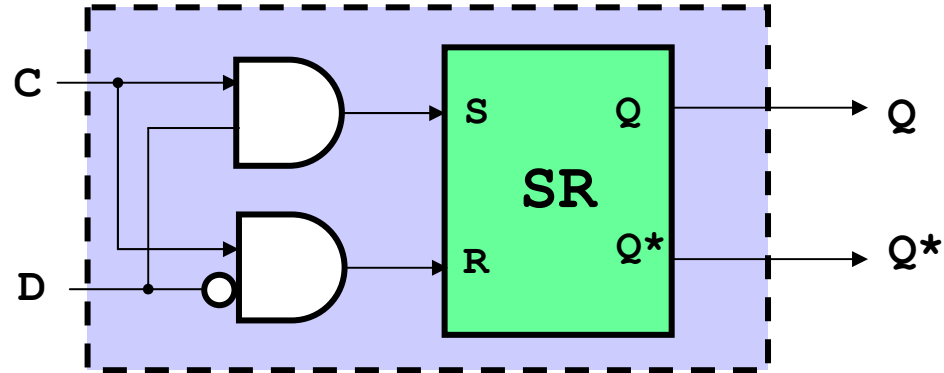
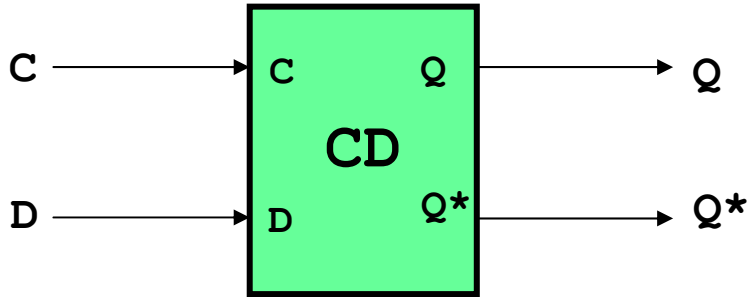
$$Q = S' \uparrow (q \uparrow R')$$



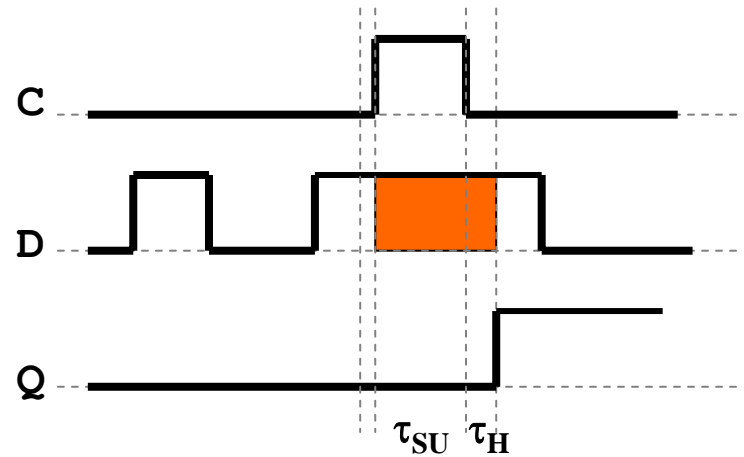
$$Q = R \downarrow (S \downarrow q)$$

I comandi di set e reset devono avere una durata minima (vedi datasheet) per consentire il raggiungimento della condizione di stabilità

RSA notevoli: Latch CD



C	D	Q	Q*
0	0	Q	Q*
0	1	Q	Q*
1	0	0	1
1	1	1	0

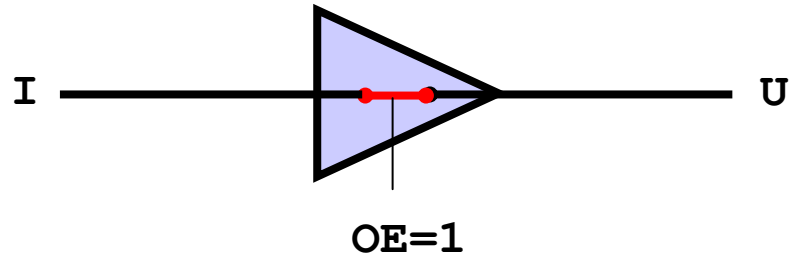
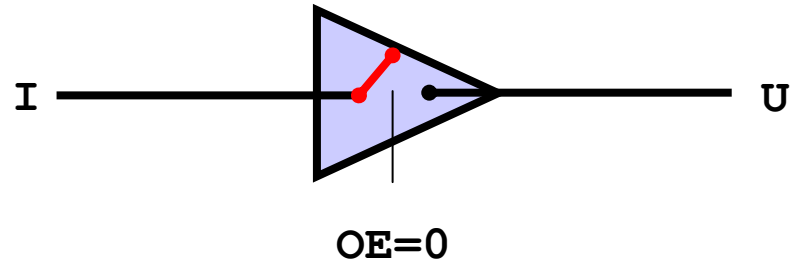
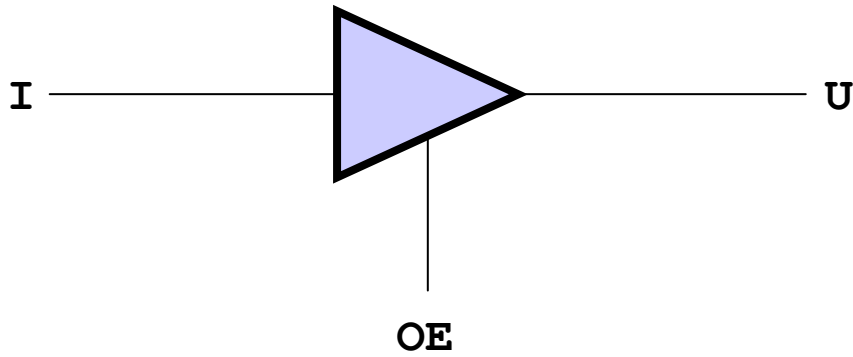


$$\text{Vincoli: } \begin{cases} \tau_{SU} \geq \tau_{SUmin} \\ \tau_H \geq \tau_{Hmin} \end{cases}$$

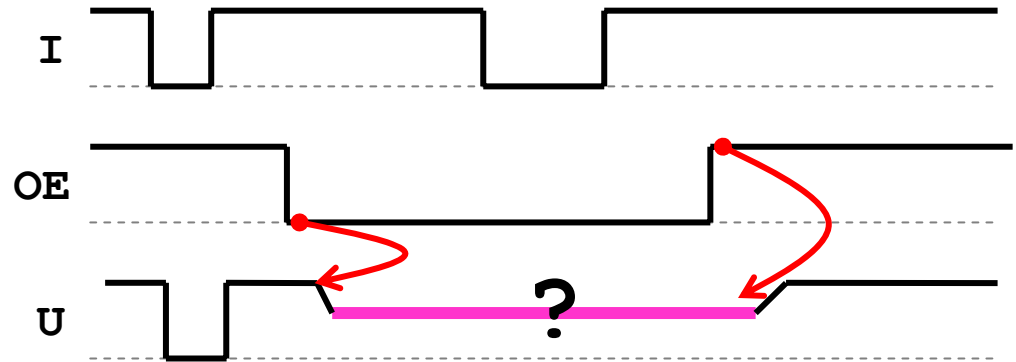
$$\text{Tempo di risposta: } \tau_R = \tau_{SU} + \tau_H$$

Latch CD: il problema/vantaggio delle "uscite trasparenti"

Driver 3-state

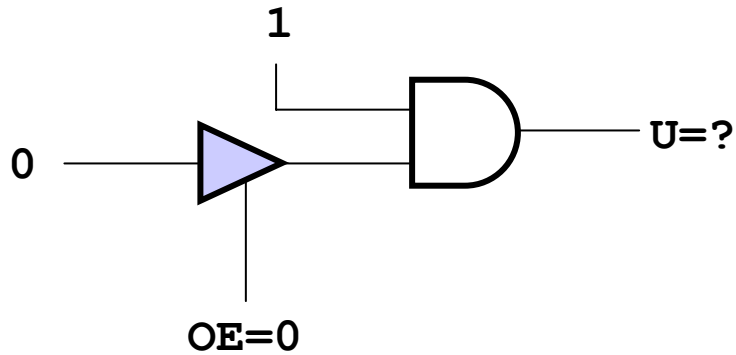


OE	I	U
1	0	0
1	1	1
0	0	Z
0	1	Z



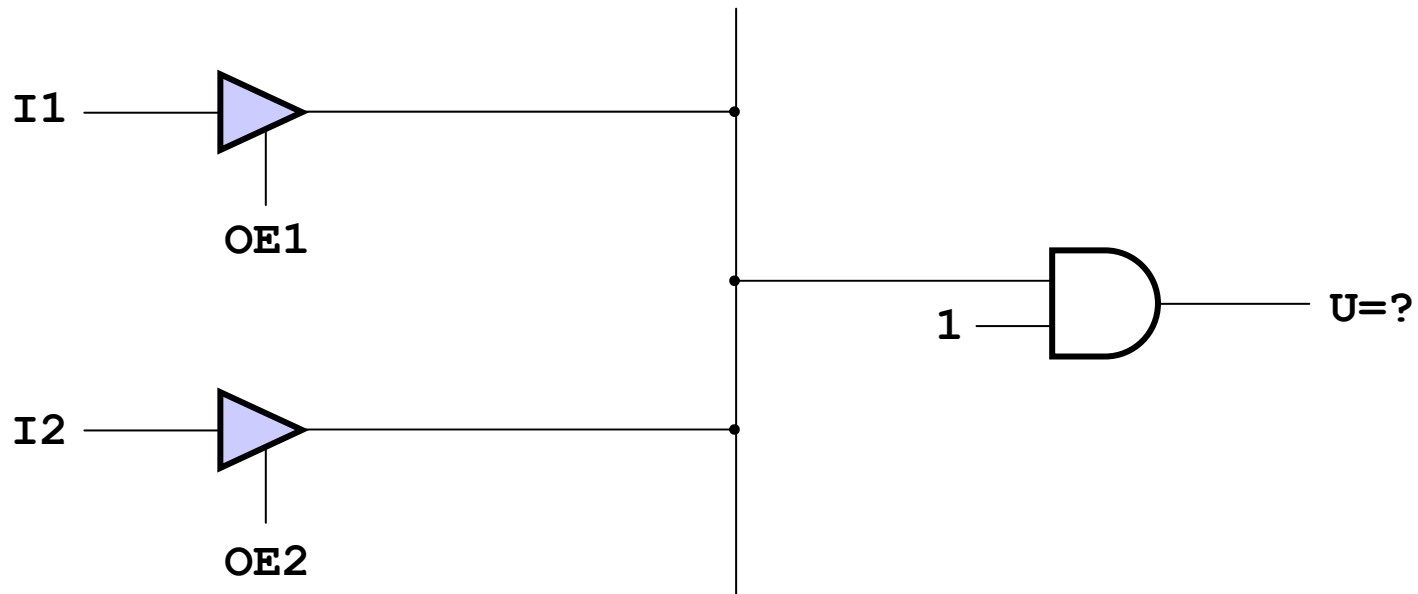
Quale è il valore della tensione ?

Quale valore logico assume U ?



Che cosa è necessario garantire nella rete seguente ?

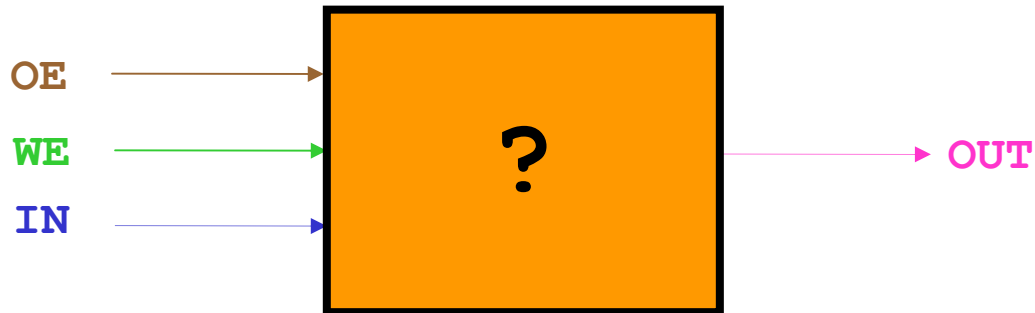
Quando il segnale U assume un valore logico significativo ?



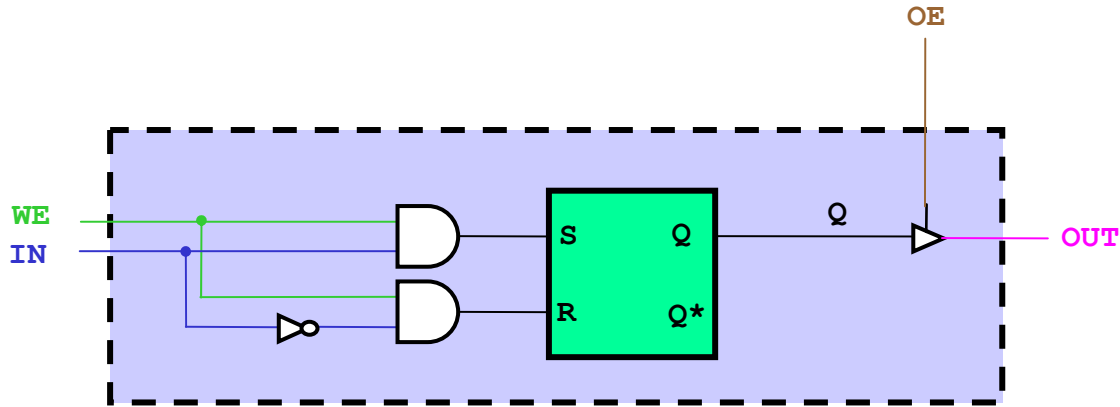
Esercizio 1

Registro a 1 bit con uscita tri-state

Utilizzando **latch SR** progettare una rete che trasferisce sull'uscita **OUT** il segnale di ingresso **IN** quando **WE=1**. L'ultimo valore trasferito in uscita deve essere mantenuto per tutto il tempo in cui il segnale **WE=0**. La rete deve essere inoltre dotata di un segnale **OE** che, se a livello logico 0, pone il segnale di uscita **OUT** nello stato di alta impedenza.



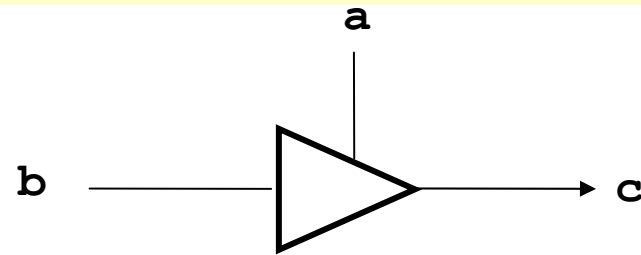
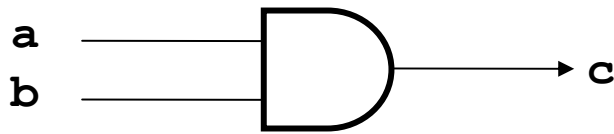
Soluzione



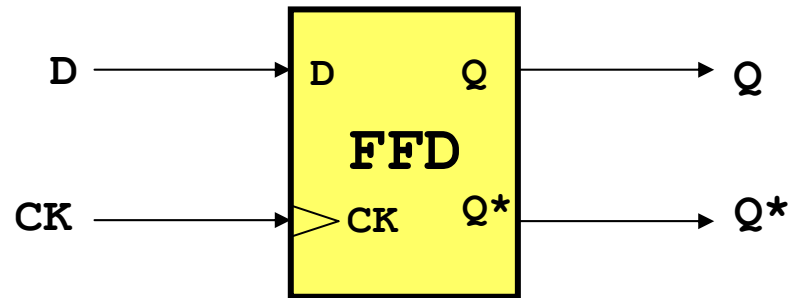
La rete tratteggiata (8X) è un latch CD dotato di uscita *tri-state* ed esiste in forma integrata ('373).

NOTA

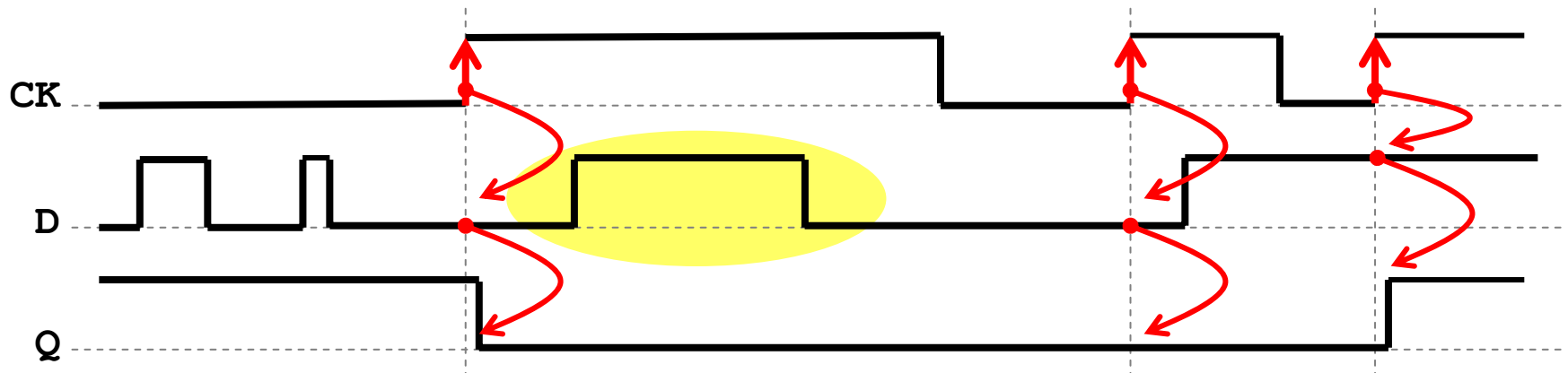
- Perché le due reti seguenti **NON** sono equivalenti ?



RSA notevoli: Flip-Flop D

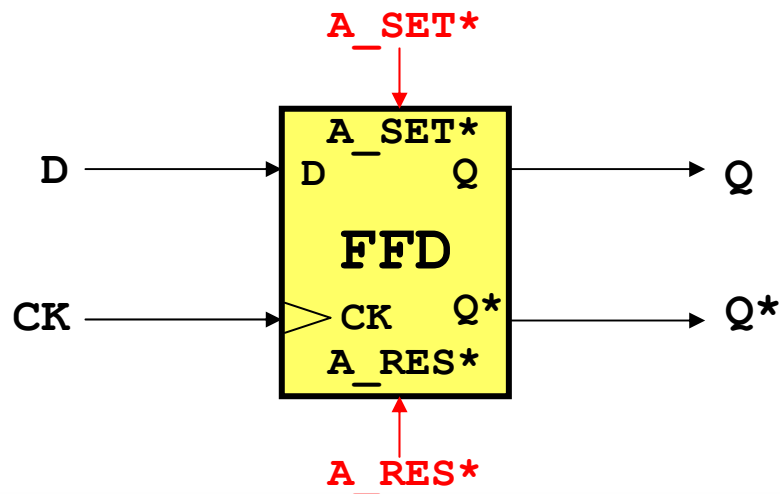


FFD: RSA che assume il valore logico presente sull'ingresso D durante i **fronti di salita** (positive edge triggered) dell'ingresso CK

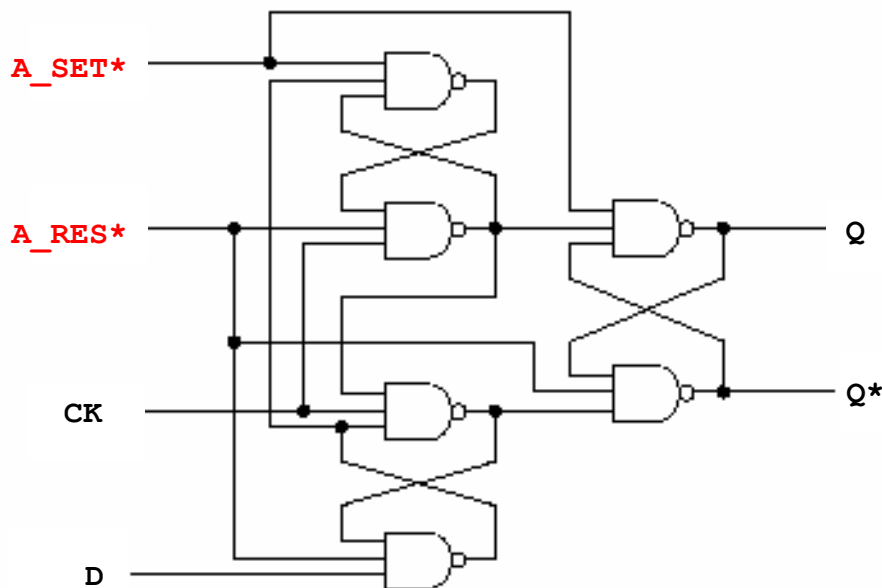


Il FFD è tipicamente utilizzato come cella elementare di memoria nelle reti sequenziali sincrone. In tal caso, il segnale CK, è un segnale di tipo periodico (*clock*).

I FFD sono dotati di due ulteriori ingressi "asincroni" che consentono di settare (**A_SET**) o resettare (**A_RES**) Q indipendentemente da CK e D.



logic diagram (positive logic)



Tipica realizzazione di un FFD della famiglia TTL ('374) mediante 3 latch SR.

$Q=0$ se **A_RES=1**

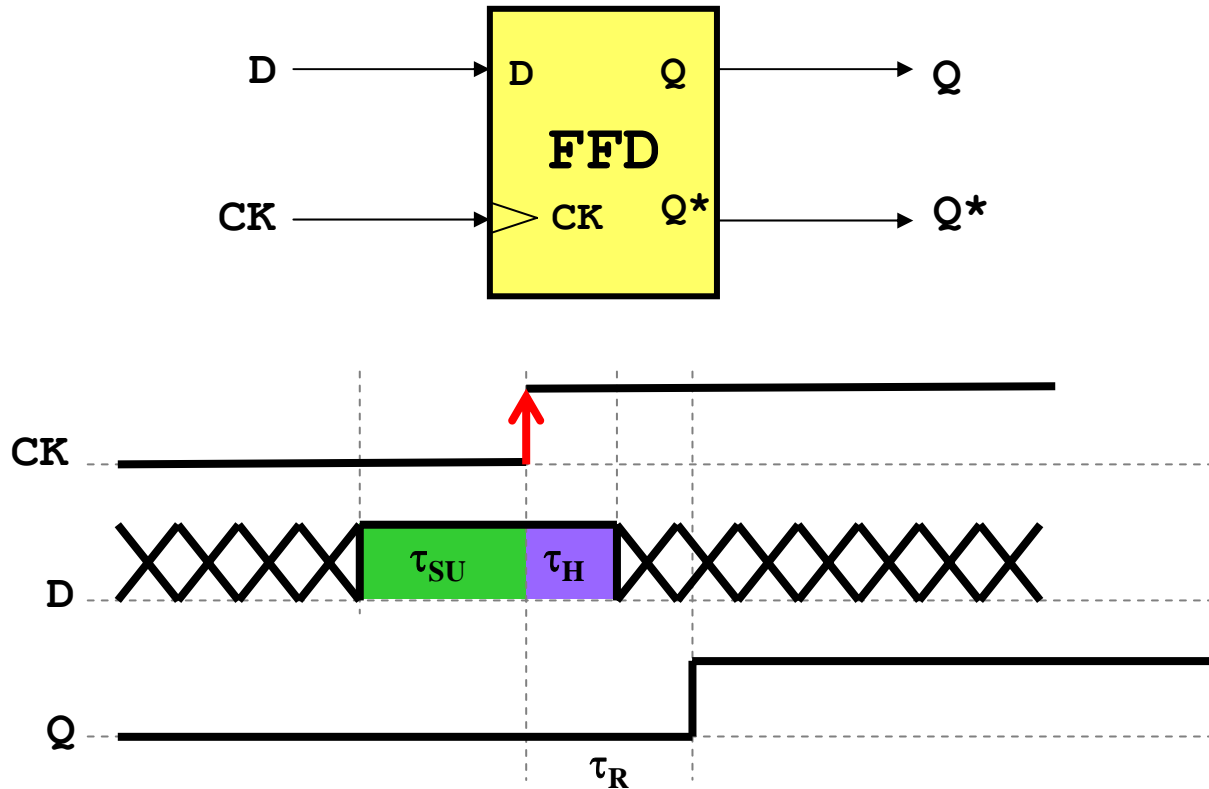
$Q=1$ se **A_SET=1**

A_SET e **A_RES** sono prioritari rispetto a CK e D

NOTA: i segnali asincroni di **set** e **reset** denominati nella slide (rispettivamente) **A_SET** e **A_RES** sono spesso denominati (rispettivamente) **PR** e **CL** oppure **S** e **R**. Inoltre, se non indicati nello schema logico si suppone che tali comandi siano inattivi (**A_SET=0** e **A_RES=0**).

Vincoli di corretto impiego per i FFD

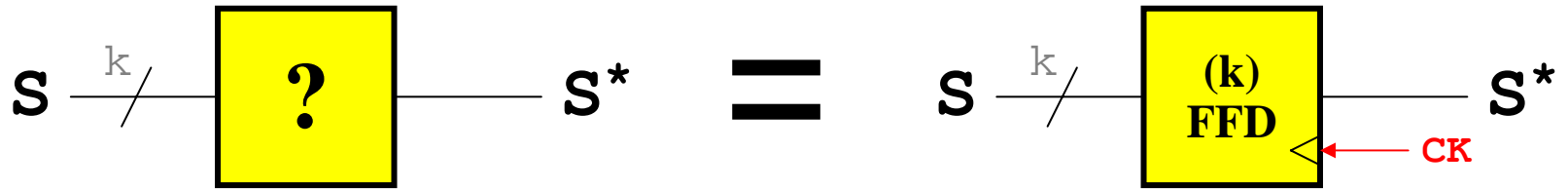
Tempi di Setup (τ_{SU}), Hold (τ_H) e Risposta (τ_R)



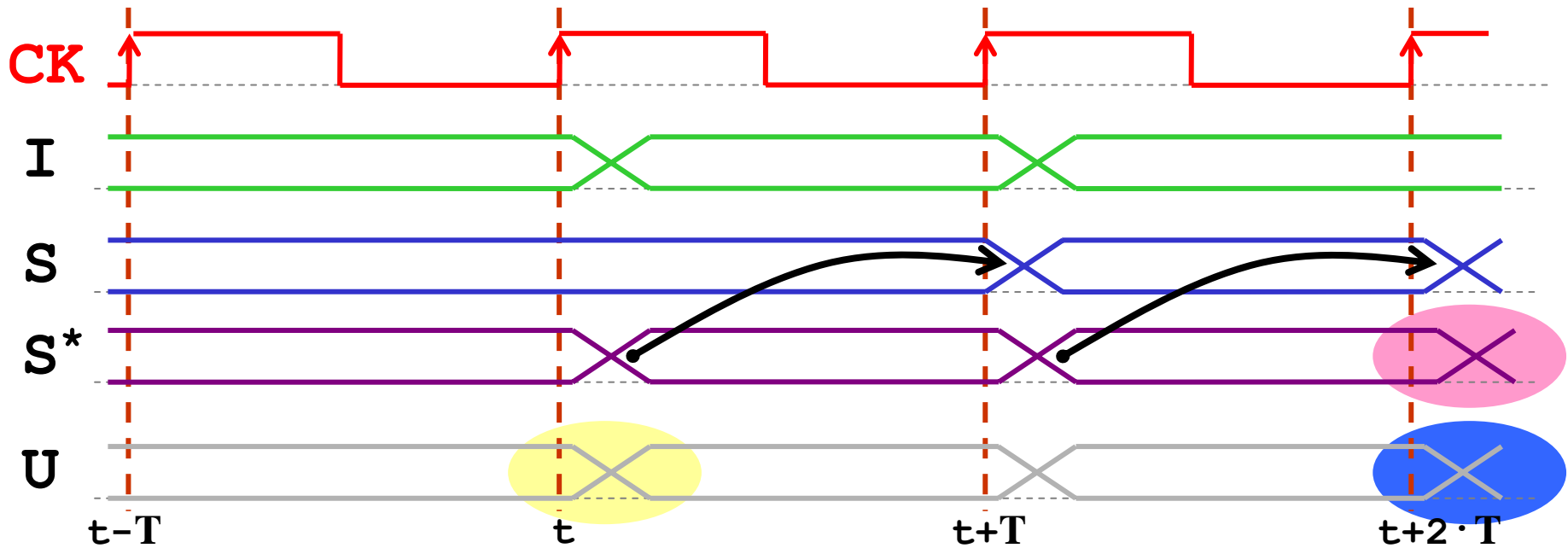
Il corretto funzionamento è garantito solo se $\tau_{SU} \geq \tau_{SUmin}$ e $\tau_H \geq \tau_{Hmin}$.
In caso contrario, metastabilità.

Cosa implicano i parametri τ_{SUmin} e τ_{Rmin} indicati nei datasheet ?

Reti Sequenziali Sincrone (RSS)



k FFD sull'anello di retroazione
Tutti con lo stesso clock di periodo T



Nel caso specifico: Moore o Mealy ?

Lo stato cambia anche se non cambia l'ingresso ?

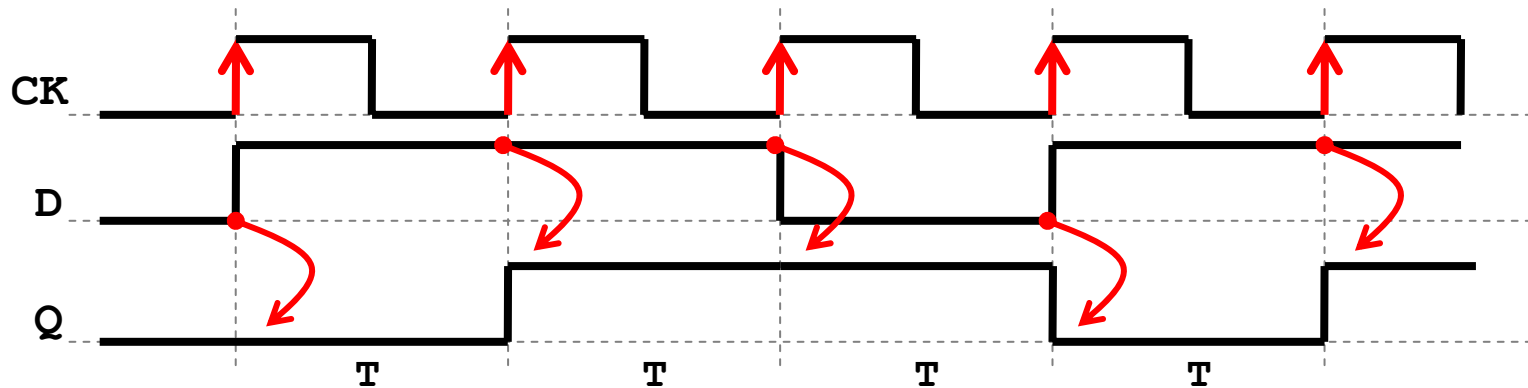
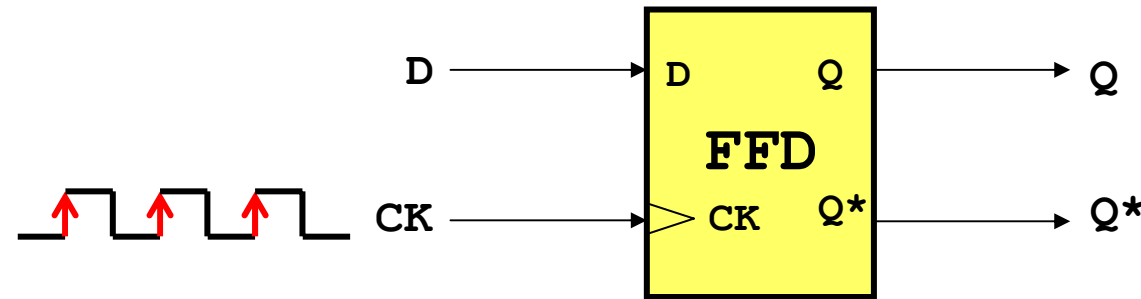
L'uscita cambia anche se non cambia l'ingresso ?

Il FFD come elemento fondamentale delle RSS

Se all'ingresso CK viene inviato un segnale periodico (clock):

il FFD ritarda ($D = \text{Delay}$) il segnale di uscita Q , rispetto al segnale di ingresso D , di un tempo pari al periodo di clock T

$$Q^{n+1} = D^n$$



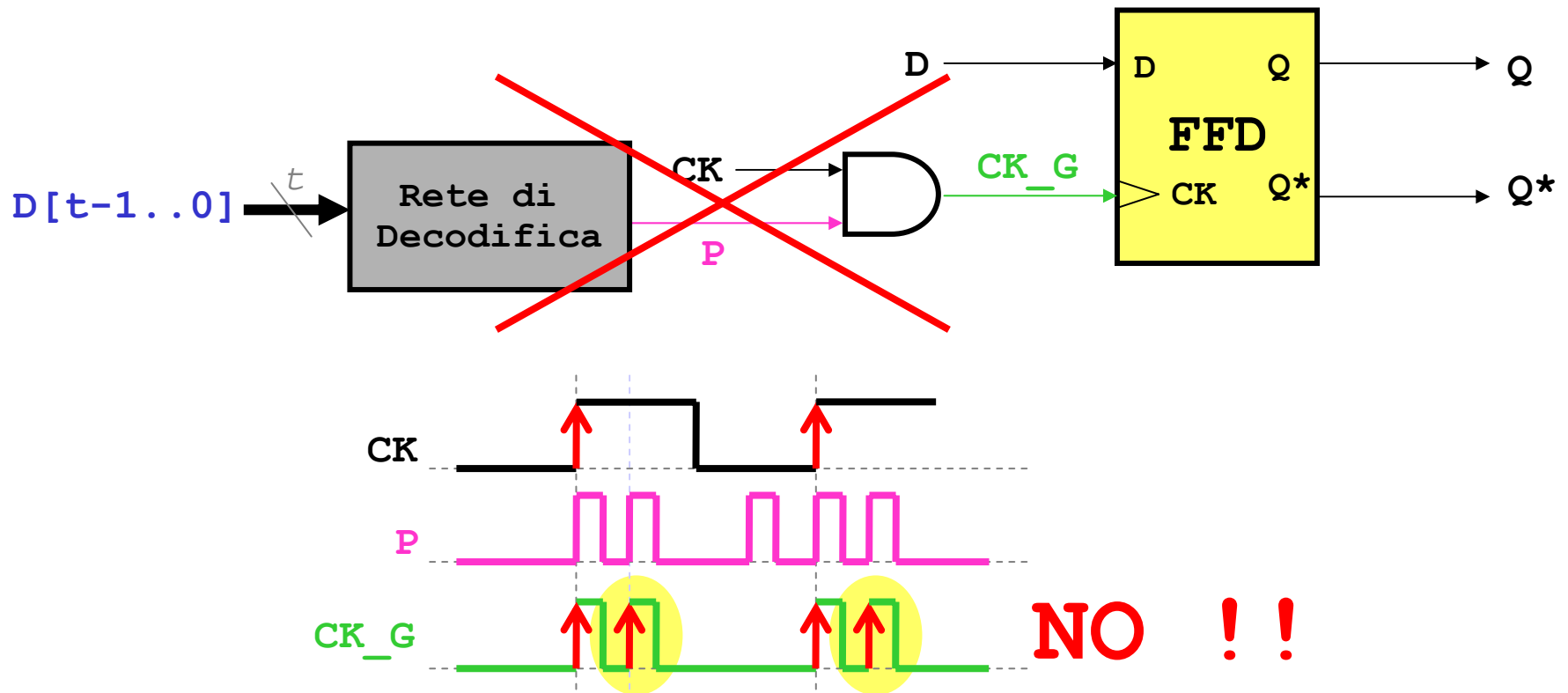
Alcune considerazioni sulle RSS

- Lo stato della rete cambia solo in corrispondenza dei fronti di salita del clock che si susseguono con periodo T
- La rete risponde ogni $T \Rightarrow$ se si desidera massimizzare la velocità di risposta della rete è necessario adottare il modello di Mealy
- La rete è svincolata dai ritardi della rete G ! Quindi, nessun problema di corse critiche (purché $T > \tau_{S\text{Umin}} + \tau_{R\text{min}}$!)
- All'interno di uno stesso progetto sono tipicamente presenti più RSS e non necessariamente per tutte le RSS il clock è lo stesso e/o coincide con il clock del processore
- Le RSS sono (più) facili da progettare delle RSA

Clock gating e glitch sul clock

Nelle reti sincrone è necessario evitare variazioni spurie (glitch) del segnale di clock che possono provocare commutazioni indesiderate dei FFD.

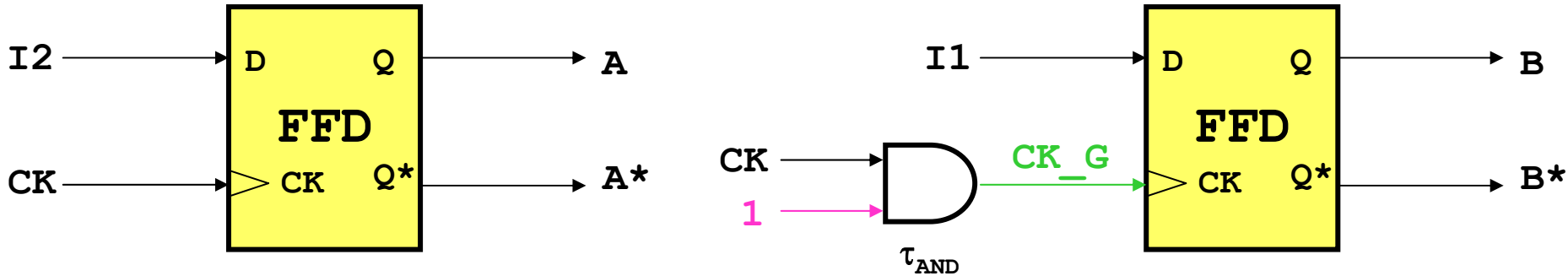
Ad esempio, per via dei ritardi tra i t segnali $D[t-1..0]$ e/o le alee introdotte dalla rete combinatoria di decodifica, a causa del "clock gating", può verificarsi quanto segue



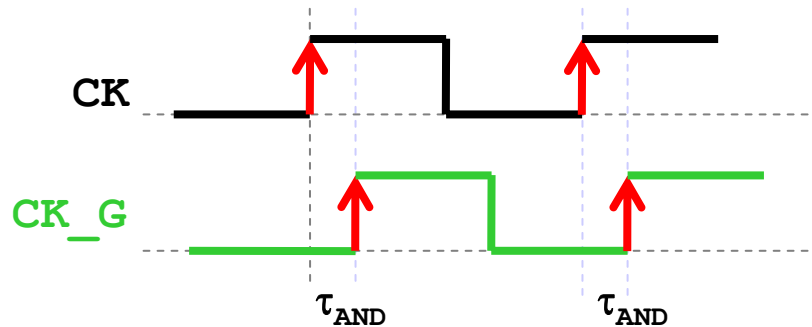
Glitch sul clock → commutazione spuria del FFD

Clock gating e clock-skew

Il clock gating, oltre a generare potenziali glitch introduce "clock-skew". Ad esempio, consideriamo le due RSS seguenti



I clock delle due reti sono sfasati di un tempo pari al ritardo introdotto dall'AND. Tale fenomeno ("clock-skew") è potenzialmente dannoso. Perché ?

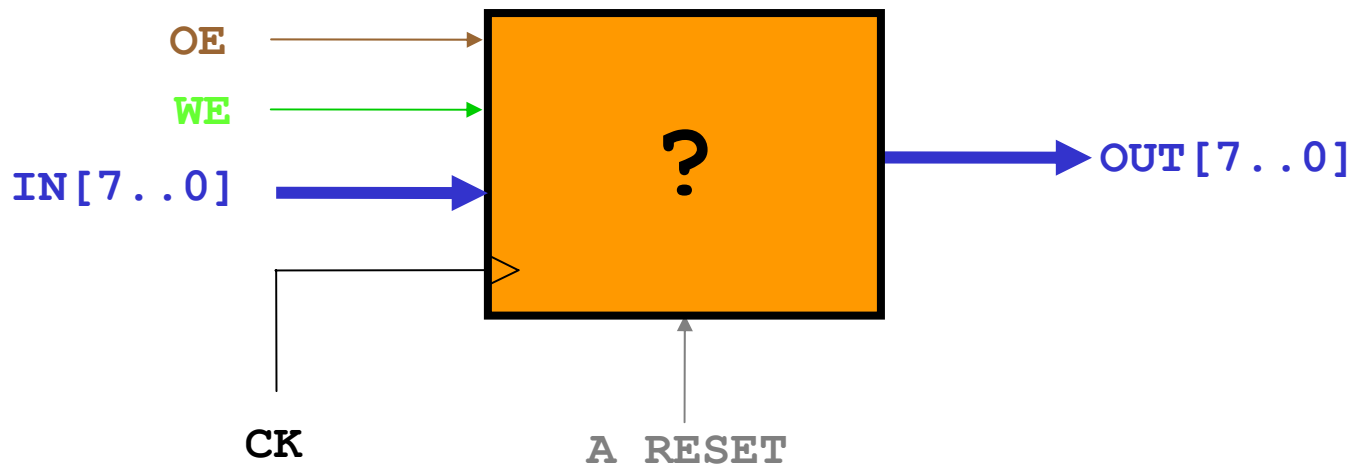


Il "clock-skew" non è causato solo dal clock gating ma anche (ad esempio) da percorsi elettrici di lunghezza diversa.

Esercizio 2

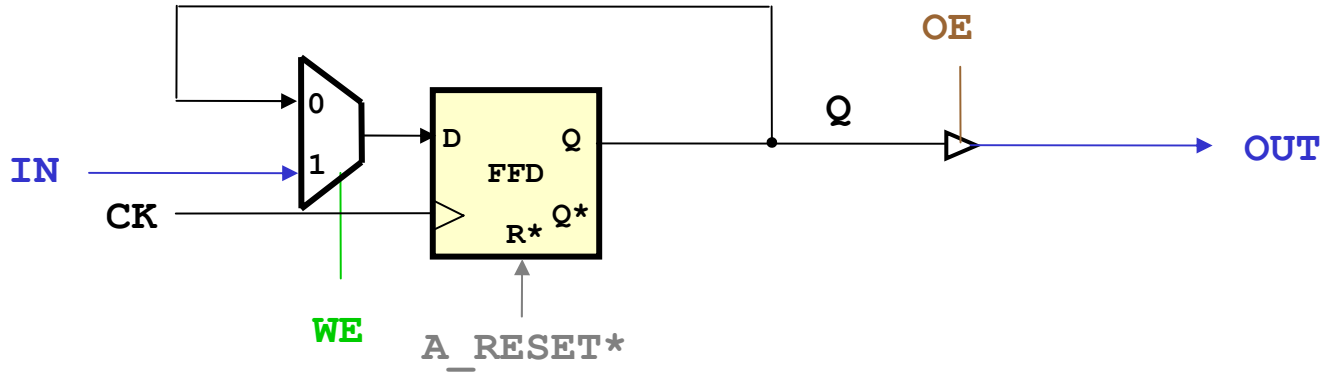
Progettare un *registro a 8 bit con uscita tri-state utilizzando FFD positive edge triggered*.

La rete, ad ogni fronte di salita del clock, memorizza il byte **IN[7..0]** in ingresso se **WE=1** mentre mantiene il valore precedentemente memorizzato in caso contrario (**WE=0**). L'uscita **OUT[7..0]** della rete deve essere posta nello stato di alta impedenza quando il segnale **OE=0**. Inoltre, la rete deve essere dotata di un ingresso asincrono di RESET (**A_RESET**) che, se 1, pone al livello logico 0 l'uscita **OUT[7..0]** indipendentemente dal valore dei segnali **WE**, **IN** e **CK**. Quali condizioni devono essere soddisfatte perché sia garantito il corretto funzionamento della rete ?



Soluzione

Caso singolo bit



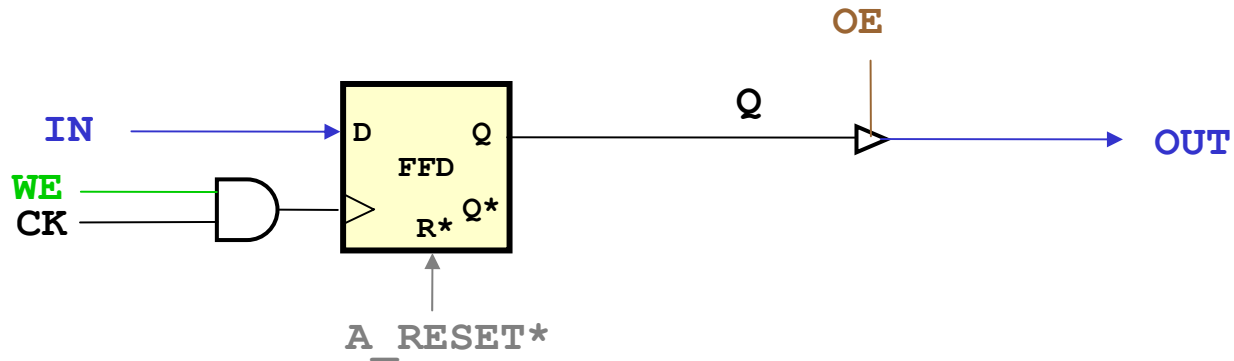
NOTA

- Per garantire il corretto funzionamento della rete è necessario rispettare tempi di **setup** e **hold**
- Il FFD esiste (8X) in forma integrata (74XX374) ed è dotato di comando di OE

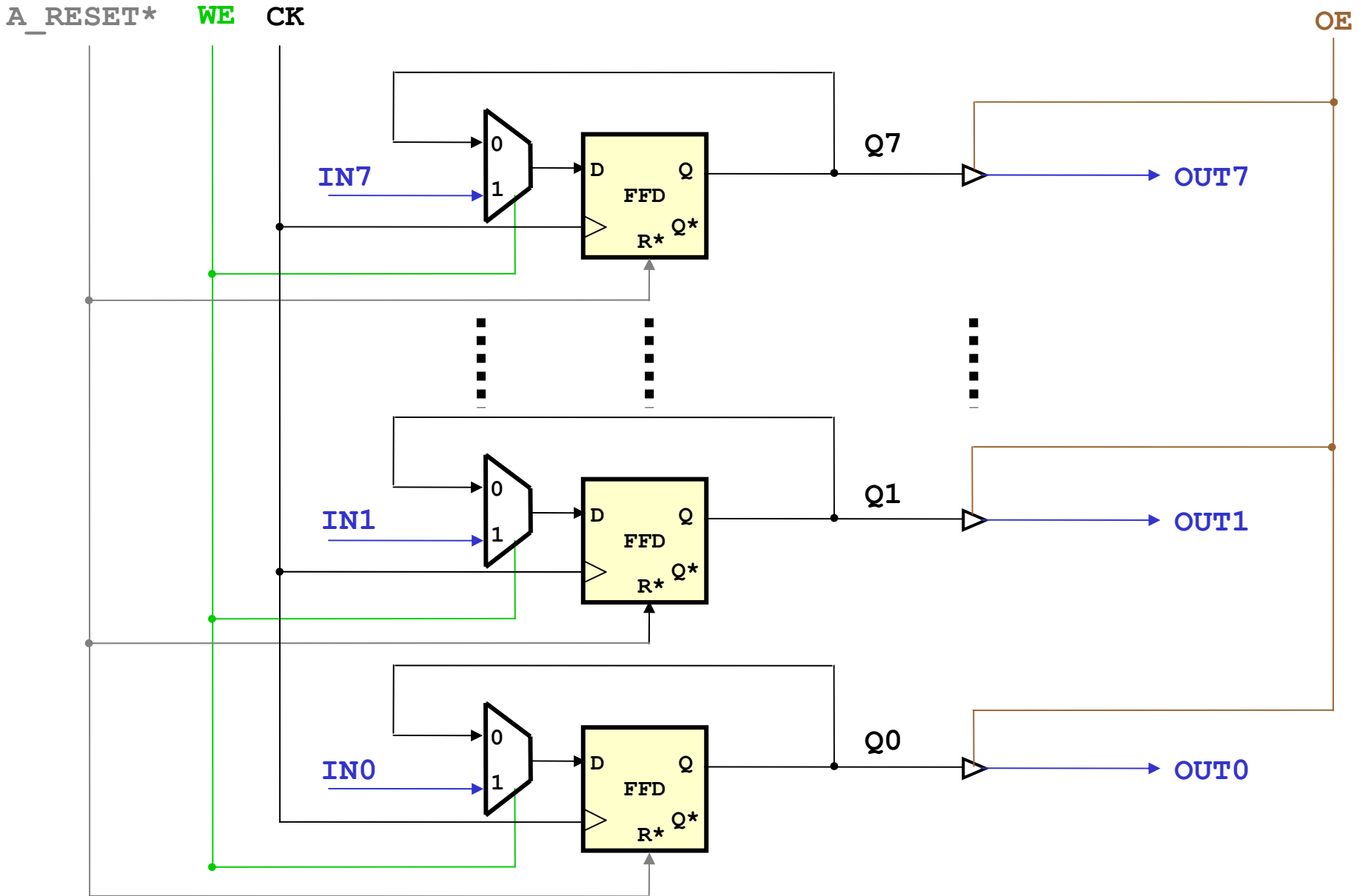
NOTA

- La soluzione seguente **NON** è corretta in quanto:

- variazioni spurie (glitch), dovute a instabilità del segnale **WE**, possono causare commutazioni indesiderate del flip-flop
- il gate ritarda il segnale di clock del FFD e potrebbe causare potenziali sfasamenti ("*clock-skew*") tra i clock dei vari componenti della rete sincrona complessiva

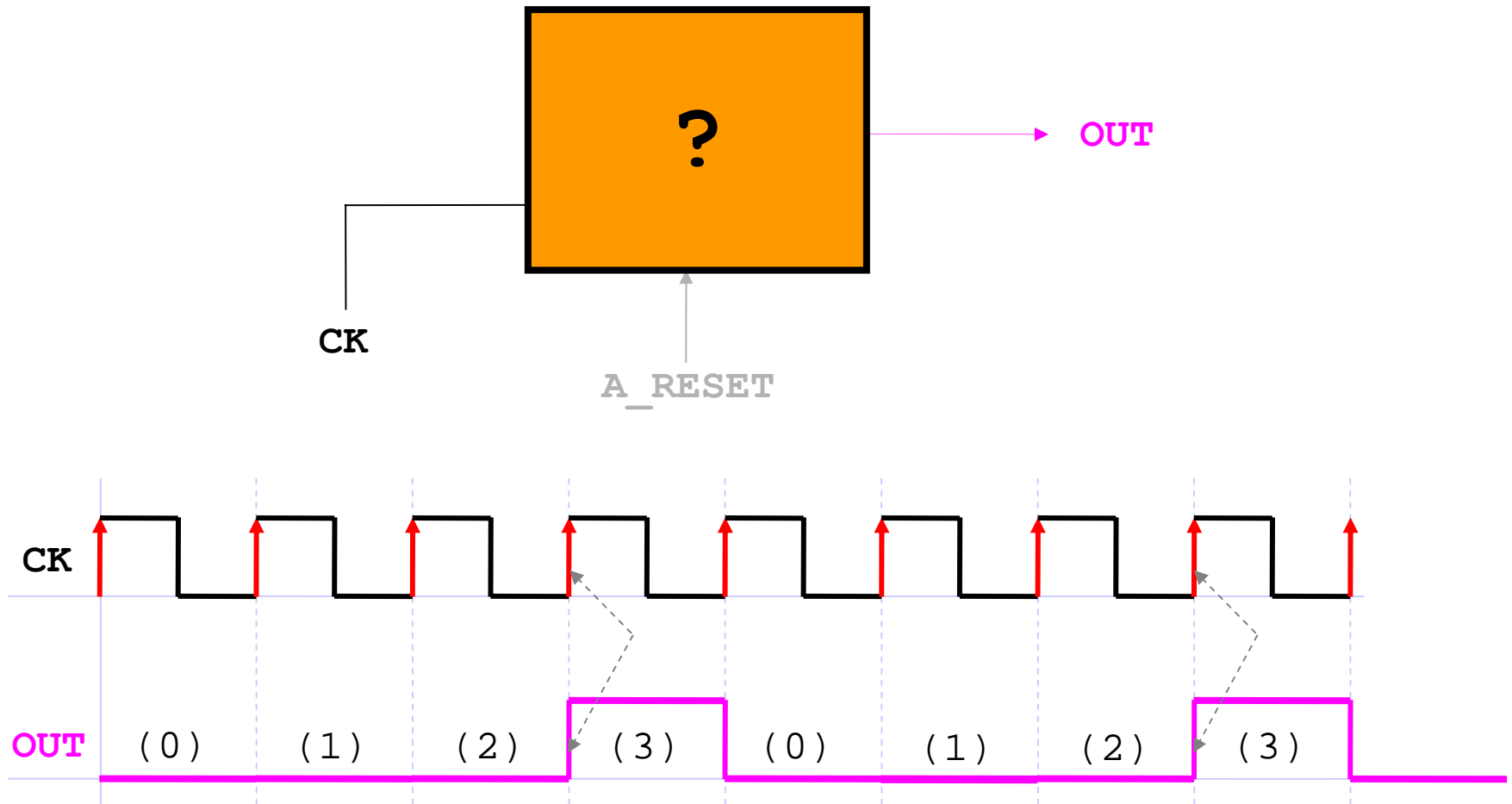


Estensione a 8 bit



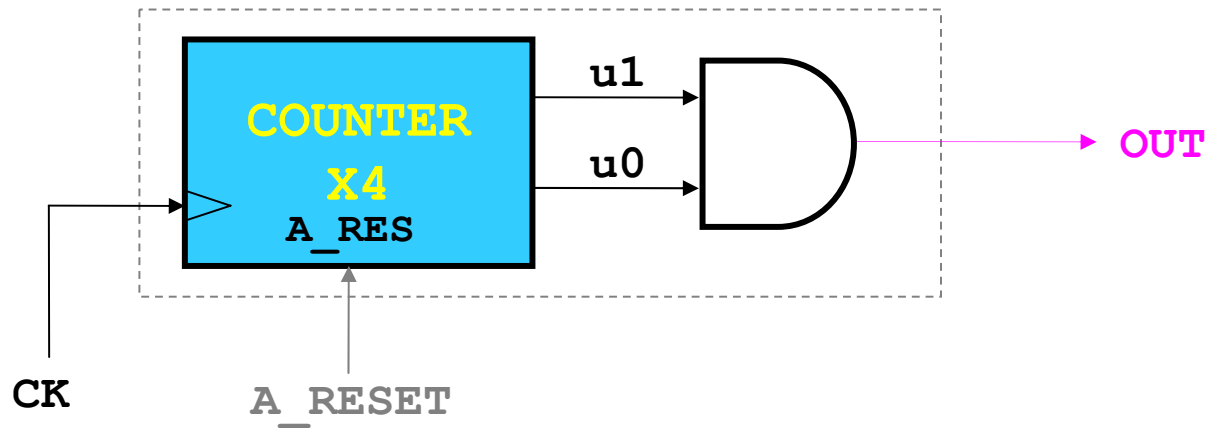
Esercizio 3

Progettare una rete che periodicamente dopo ogni tre periodi di clock setta al livello logico 1 la propria uscita per un periodo clock.



Soluzione 3.1

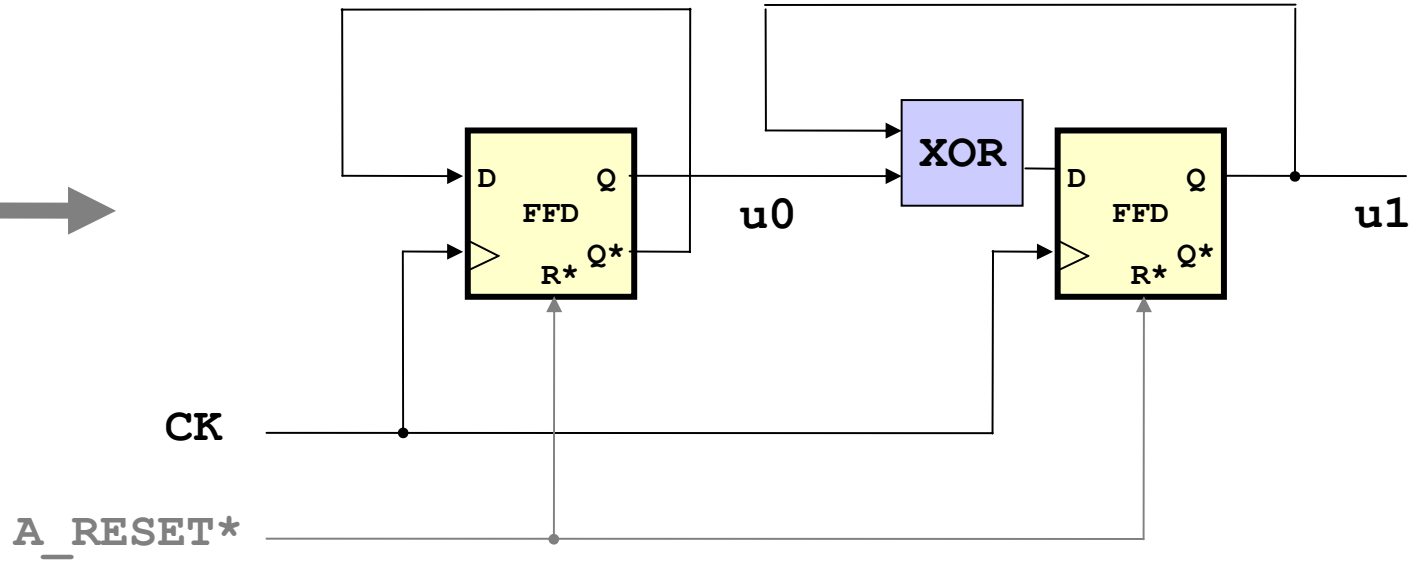
Una possibile soluzione si basa sull'utilizzo di un contatore modulo 4.



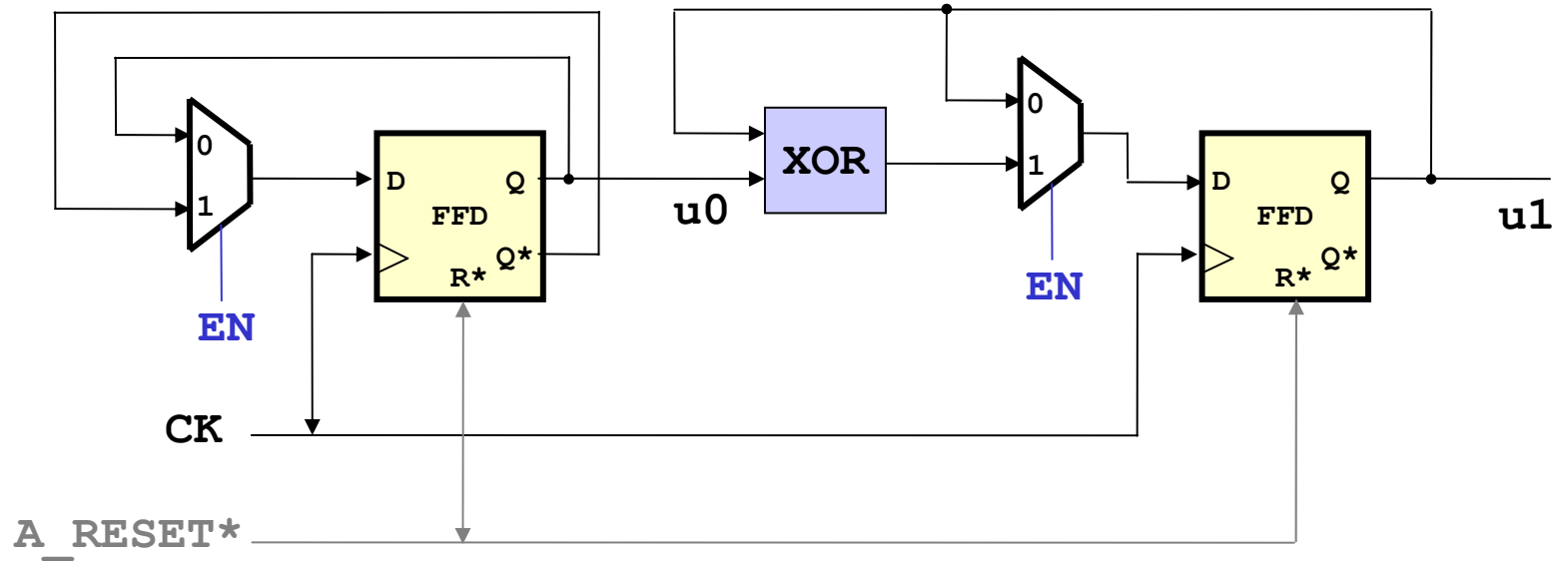
Progettare un contatore modulo 4...

Contatore modulo 4

u1	u0
0	0
0	1
1	0
1	1

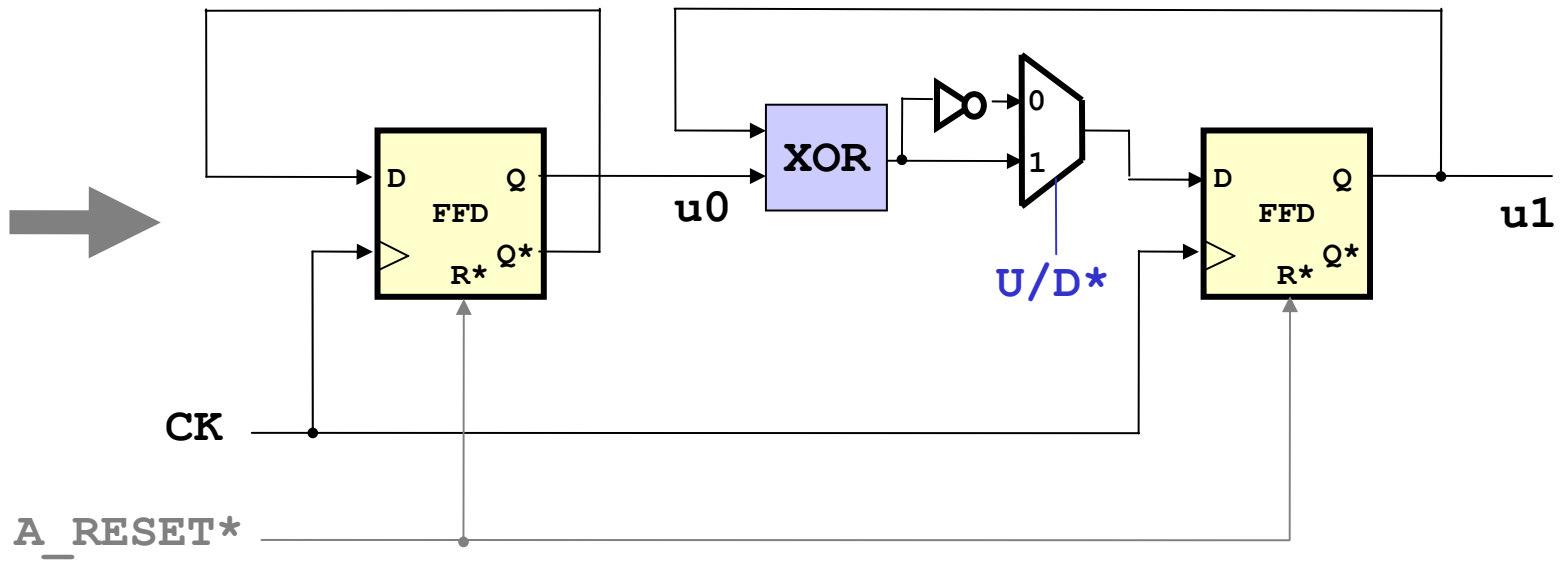
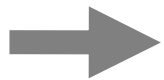


Contatore modulo 4 con segnale di ENABLE (EN)

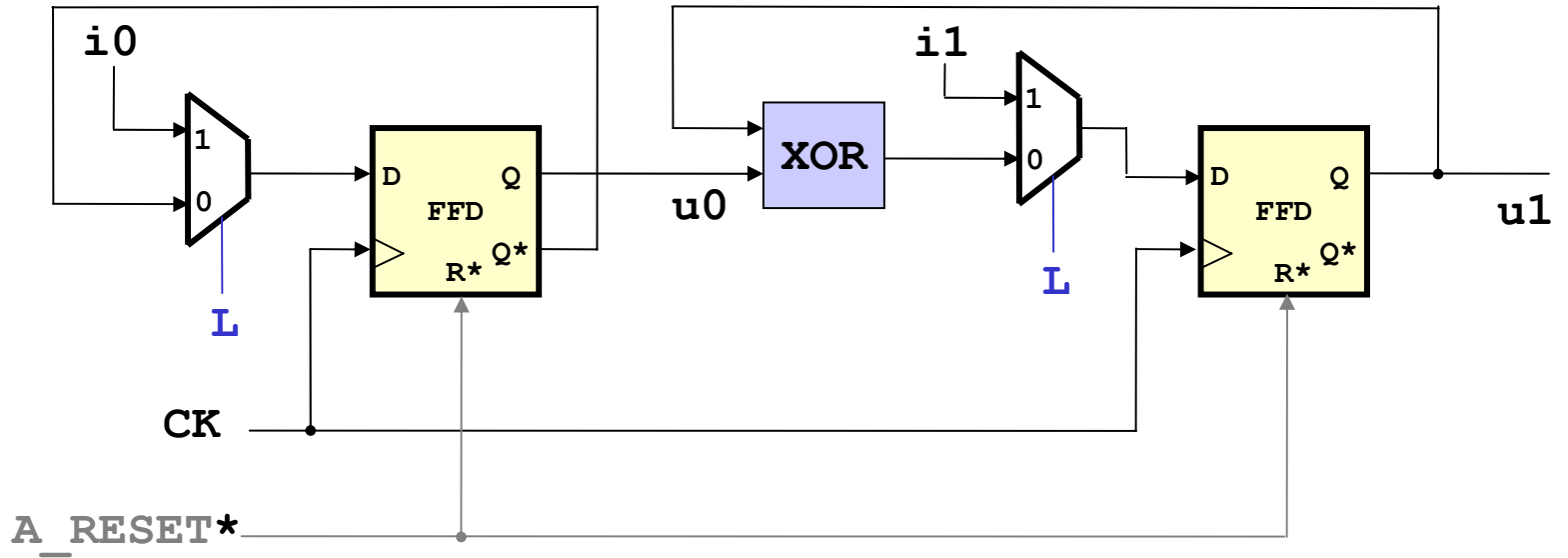


Contatore modulo 4 UP/DOWN (U/D*)

u1	u0
0	0
0	1
1	0
1	1



Contatore modulo 4 con LOAD (L)



Esercizi

E3-1) Progettare un contatore **modulo 4** dotato dei segnali **U/D***, **EN** e **L** nei seguenti 2 casi:

a) segnale **L** prioritario rispetto a **EN**

b) segnale **EN** prioritario rispetto a **L**

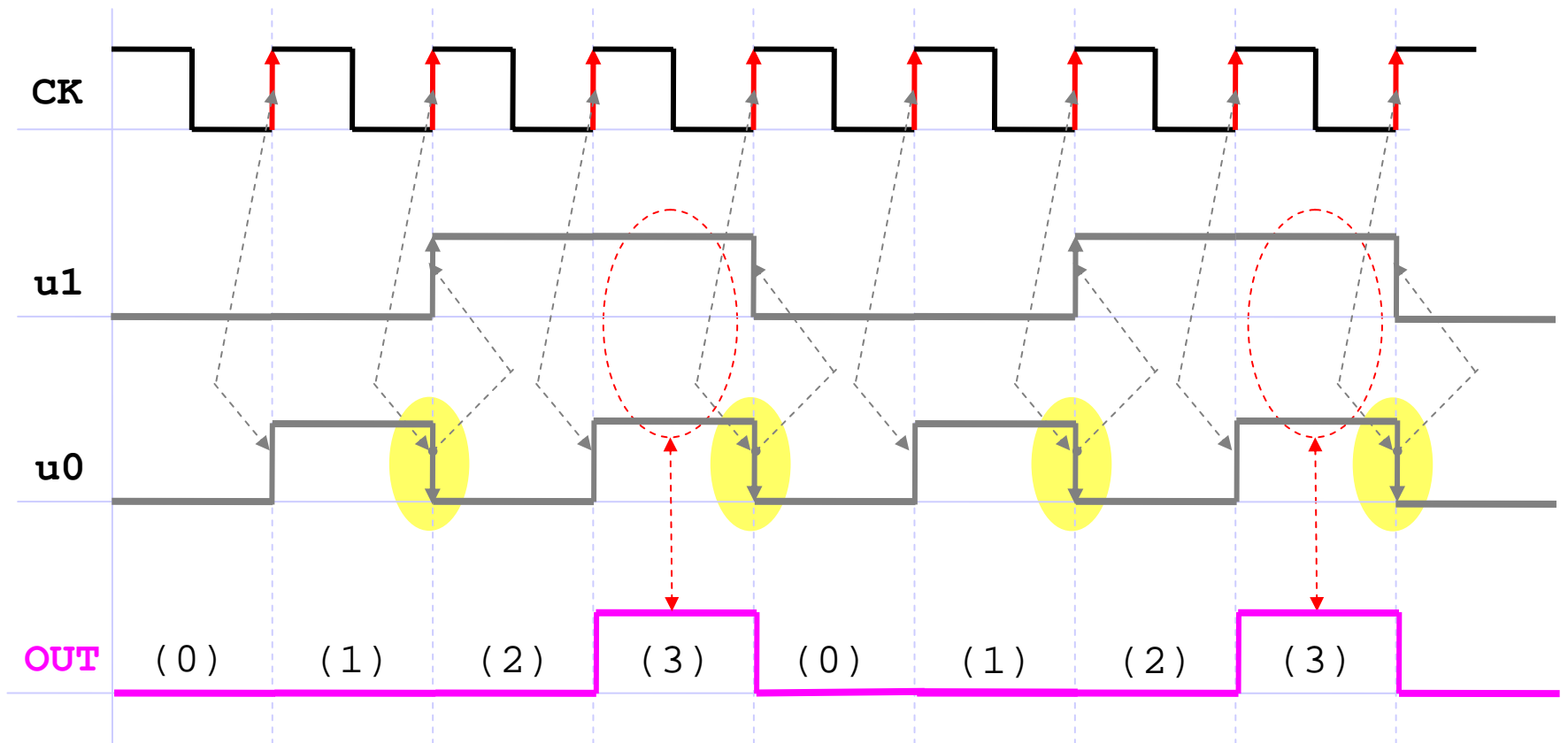
In entrambi i casi si supponga che **U/D*** sia il segnale meno prioritario tra i tre.

E3-2) Progettare un contatore **modulo 8**

E3-3) Progettare un contatore **modulo 5** utilizzando un contatore **modulo 8**

Soluzione 3.2

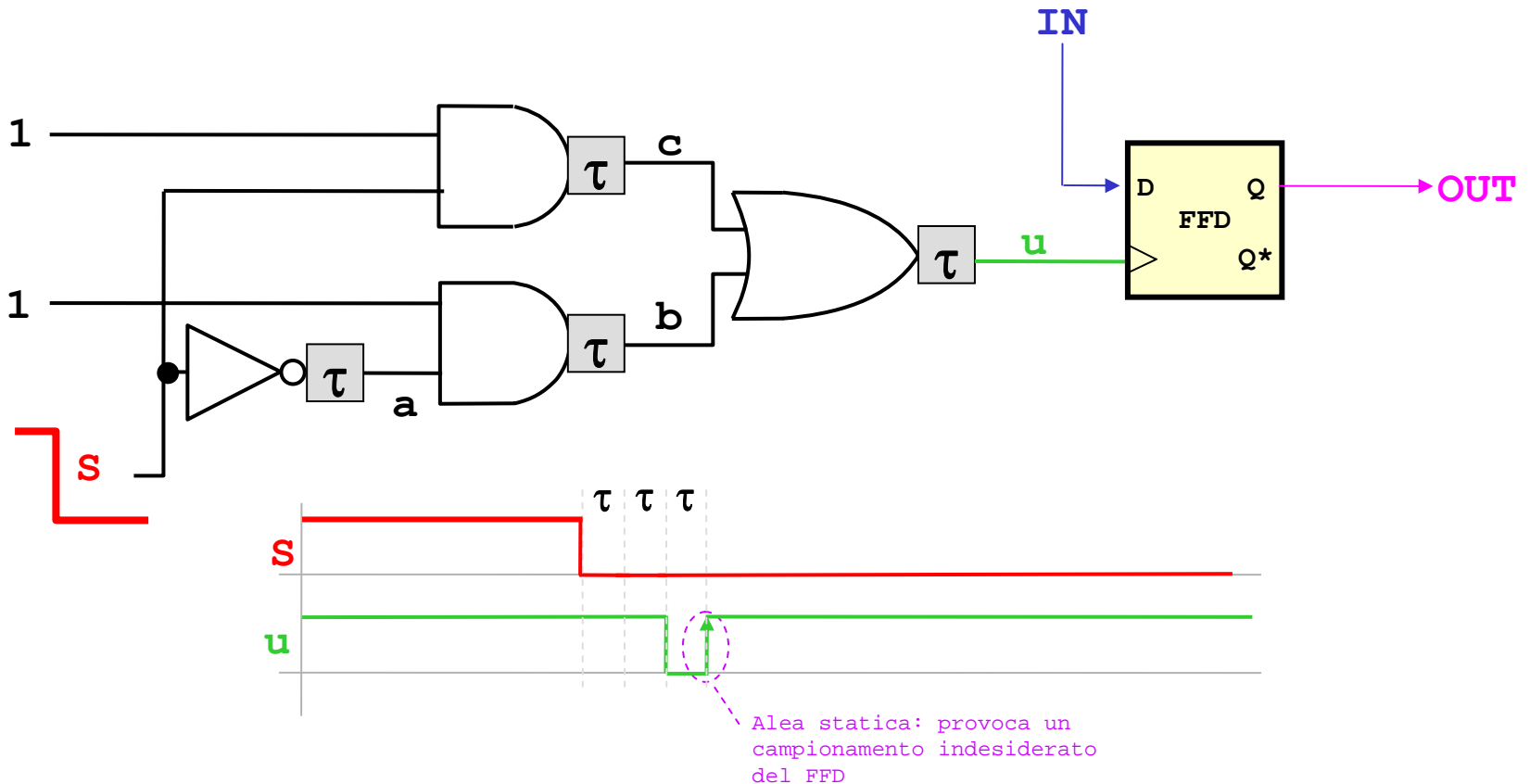
Osservando le forme d'onda mostrate sotto si può ottenere una soluzione alternativa alla precedente (3.1)



NOTA

- Non è il caso della rete della pagina precedente, ma la presenza di alee può creare problemi alle reti che seguono se queste utilizzano come ingresso di clock un segnale che presenta oscillazioni spurie (*glitches*).

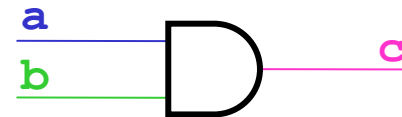
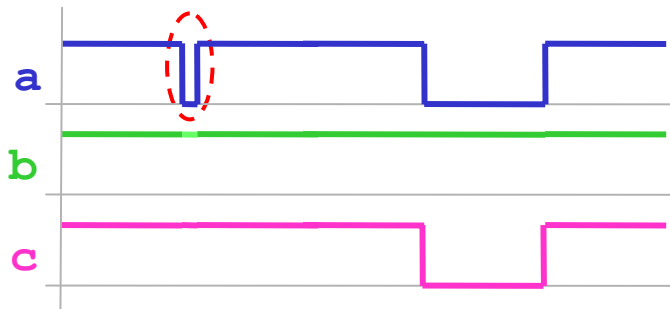
Si consideri ad esempio il caso seguente:



NOTA

- Le alee possono essere eliminate introducendo ulteriori gates (vedi reti logiche)
- In alcuni casi le alee possono essere filtrate dagli stessi *gates* (ad esempio nel caso di '*lentezza*' dei dispositivi rispetto ai tempi del glitch); questa possibilità deve essere verificata attentamente analizzando i *datasheets* dei componenti utilizzati

Un impulso troppo breve potrebbe essere filtrato dall'AND



Soluzione 3.3

Soluzione canonica ottenuta mediante *sintesi formale*.

Grafo degli stati

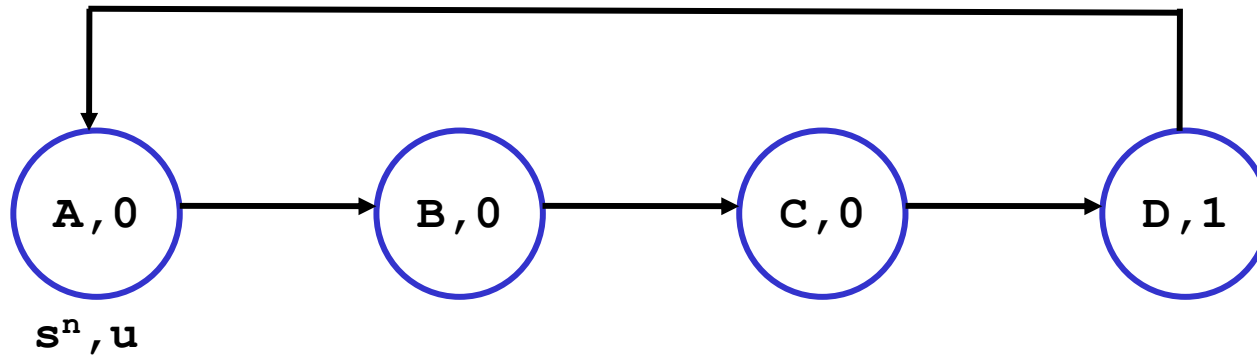


Tabella di
flusso

s^n	s^{n+1}	u
A	B	0
B	C	0
C	D	0
D	A	1

Tabella delle
transizioni

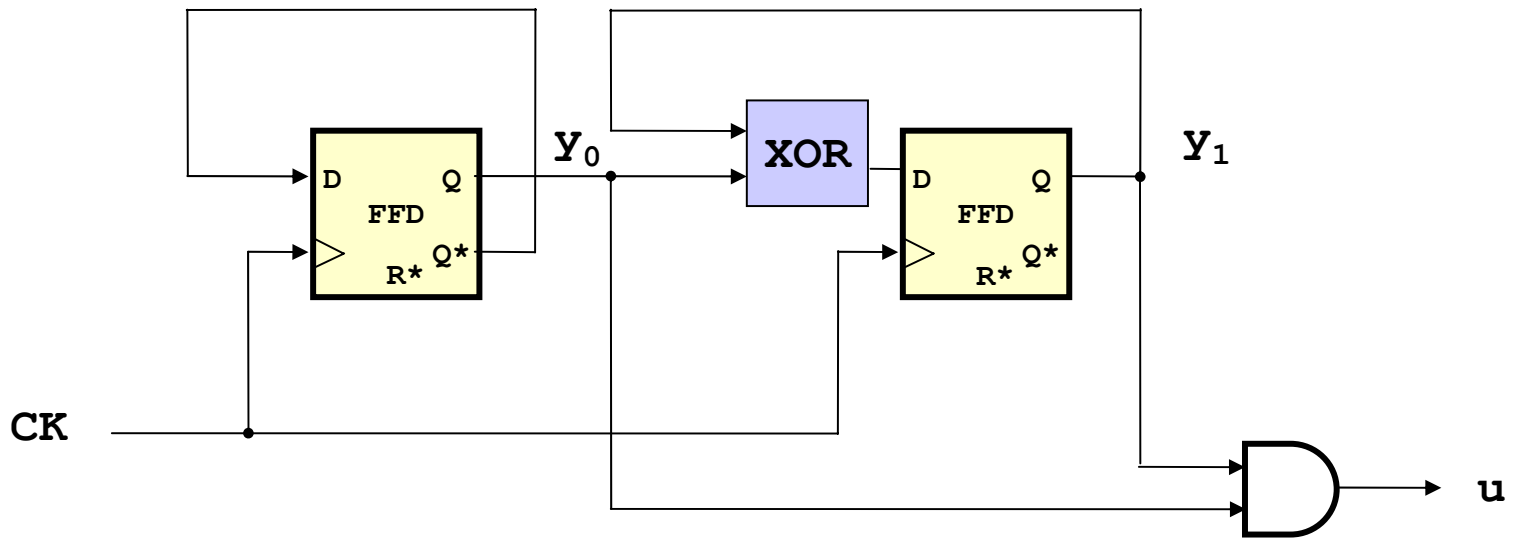
y_1^n	y_0^n	y_1^{n+1}	y_0^{n+1}	u
0	0	0	1	0
0	1	1	0	0
1	0	1	1	0
1	1	0	0	1

Sintesi minima
(mappe di Karnaugh,...)

$$u = y_1^n \cdot y_0^n$$

$$y_0^{n+1} = y_0^n$$

$$y_1^{n+1} = y_1^n \text{ XOR } y_0^n$$

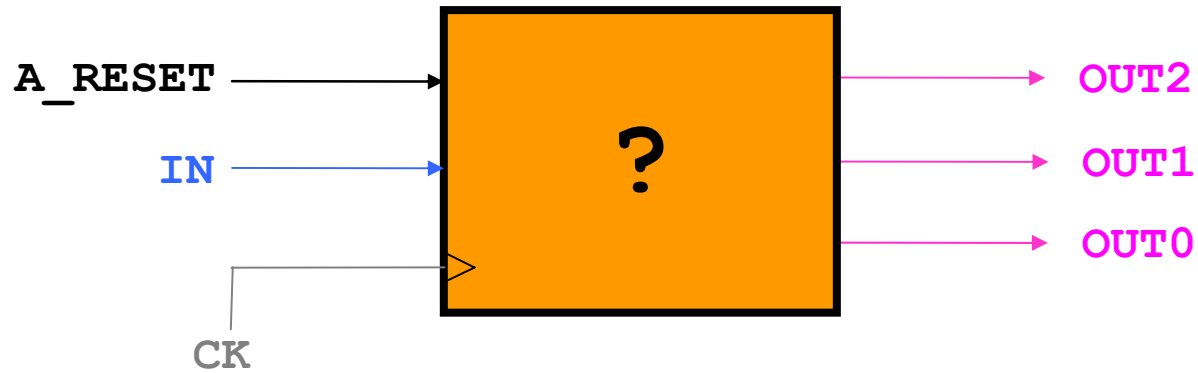


NOTA

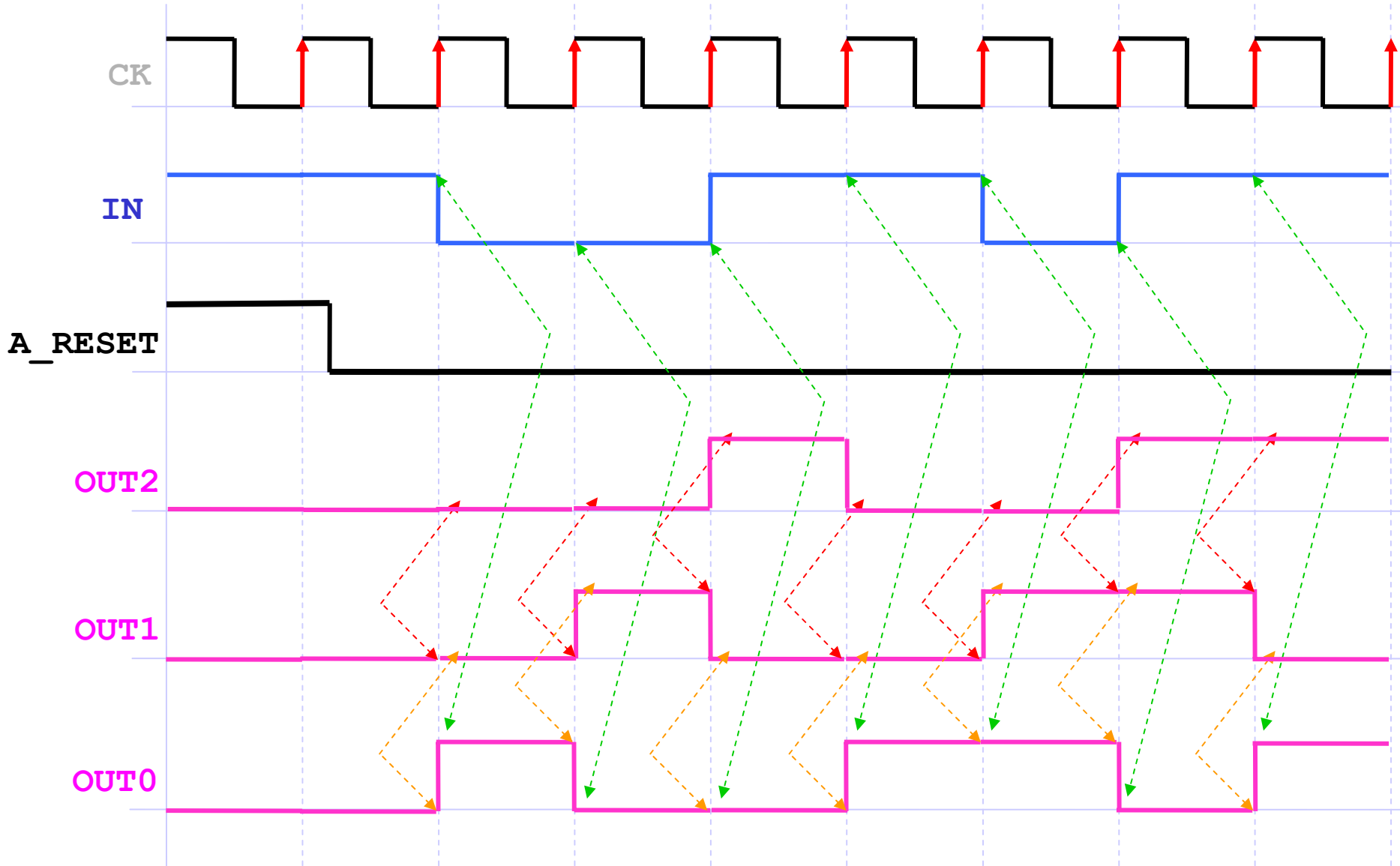
- Se si desidera aggiungere un segnale di **ENABLE** alla rete precedente mediante il metodo della sintesi formale ?
- E' necessario ripetere tutti i passi precedenti (grafo, diagramma stati, ...)

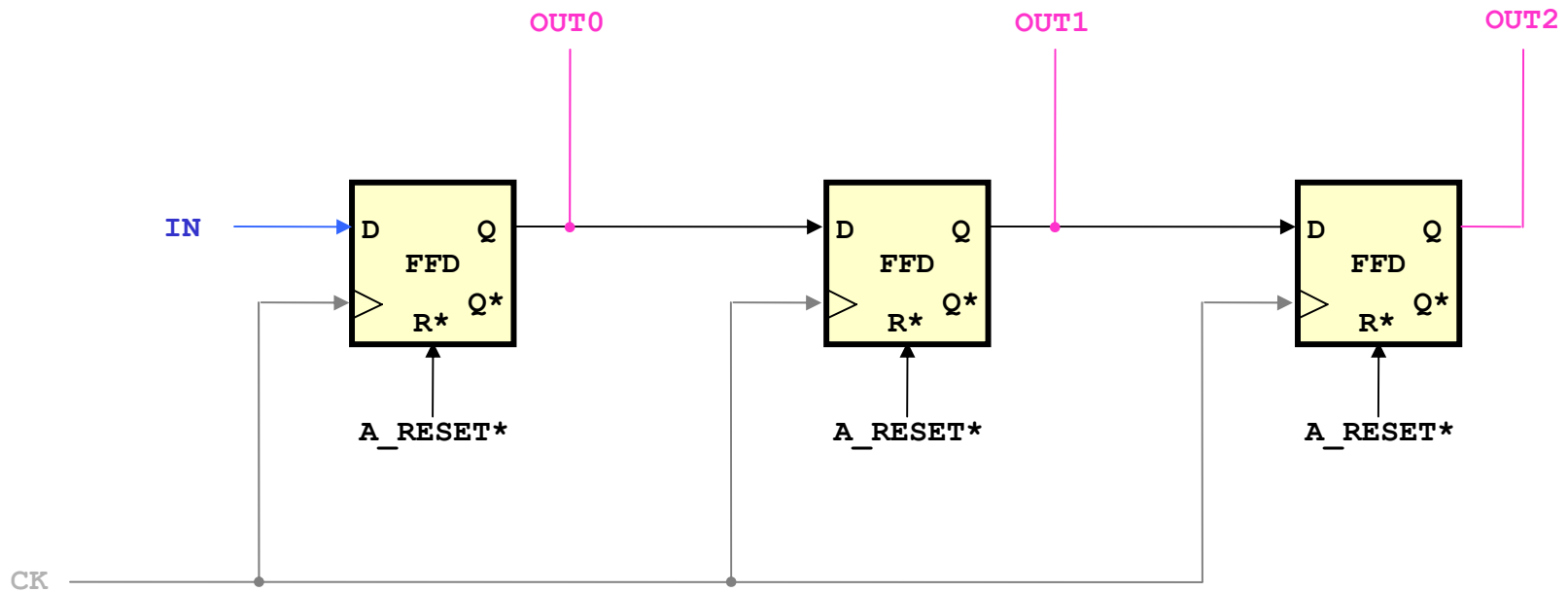
Esercizio 4

Progettare un registro a scorrimento (*shift-register*) a 3 bit.



Soluzione





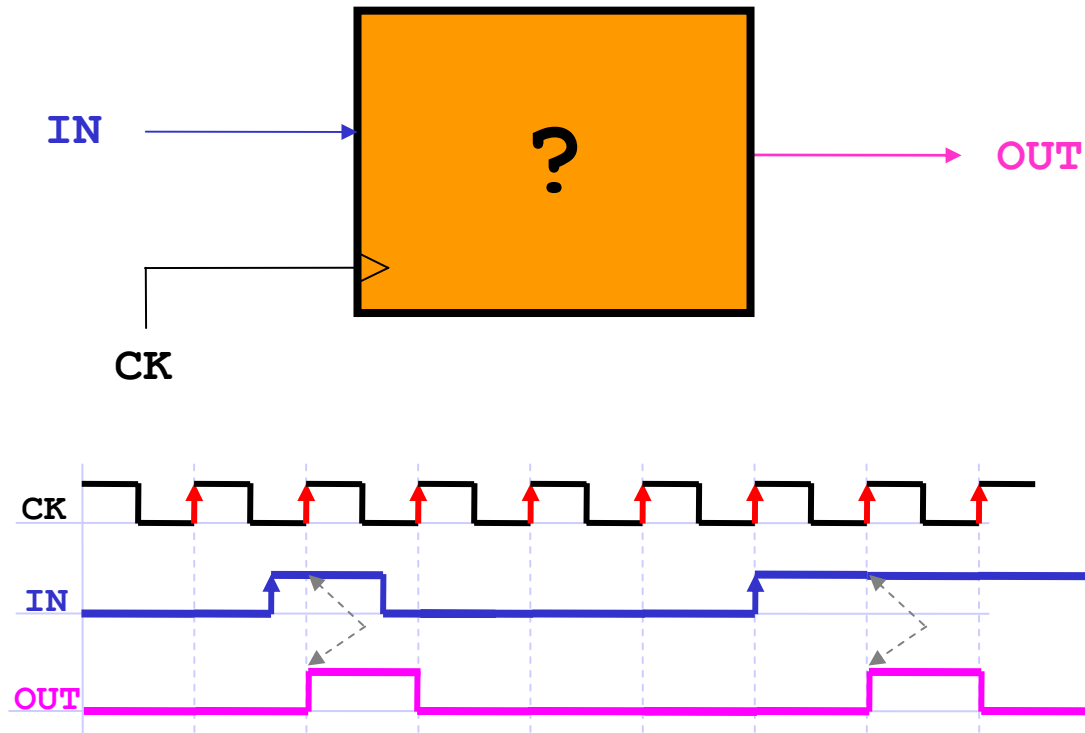
Esercizi

- E4-1)** Progettare uno shift-register dotato di comandi di enable *EN* e *LOAD* (parallelo e prioritario rispetto all'enable).
- E4-2)** Utilizzando due shift-register a 4 bit e un contatore modulo 8: progettare un convertitore serie parallelo a 8 bit dotato di un segnale (*ACK*) che comunica l'avvenuta ricezione degli 8 bit.

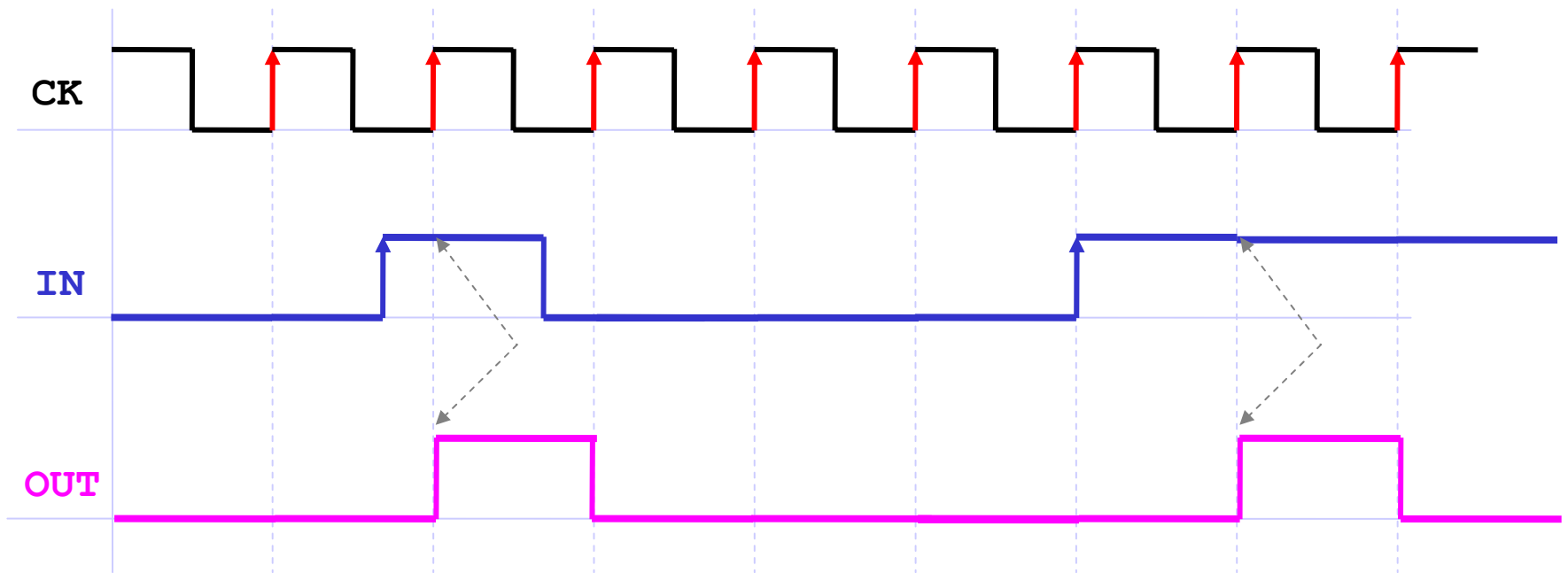
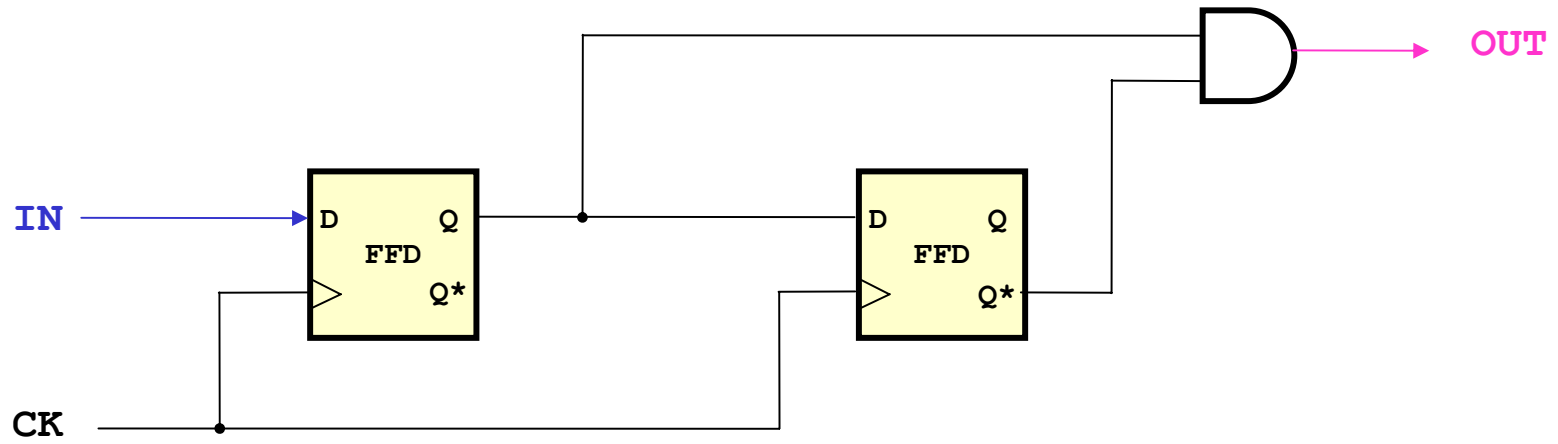
Esercizio 5

Progettare una rete sincrona dotata di un ingresso (**IN**) la cui uscita (**OUT**) deve asserirsi per un periodo di clock quando viene rilevata una transizione da 0 a 1 del segnale di ingresso (*monoimpulsore*).

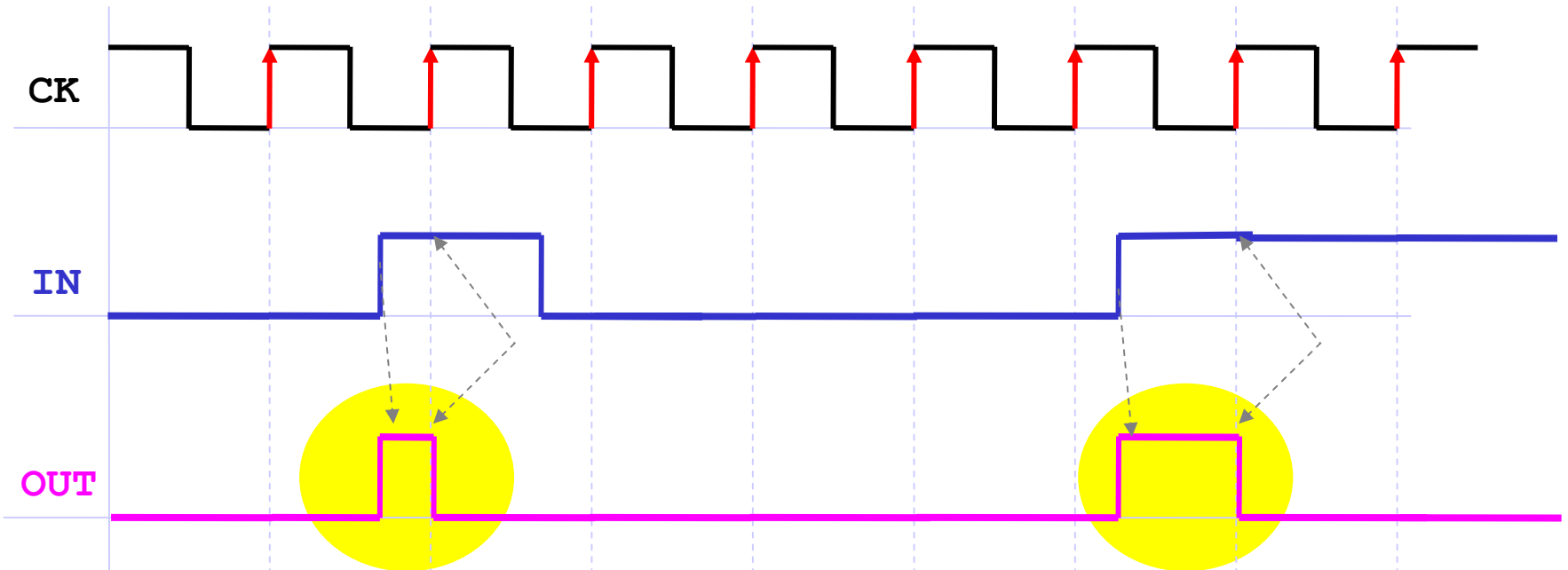
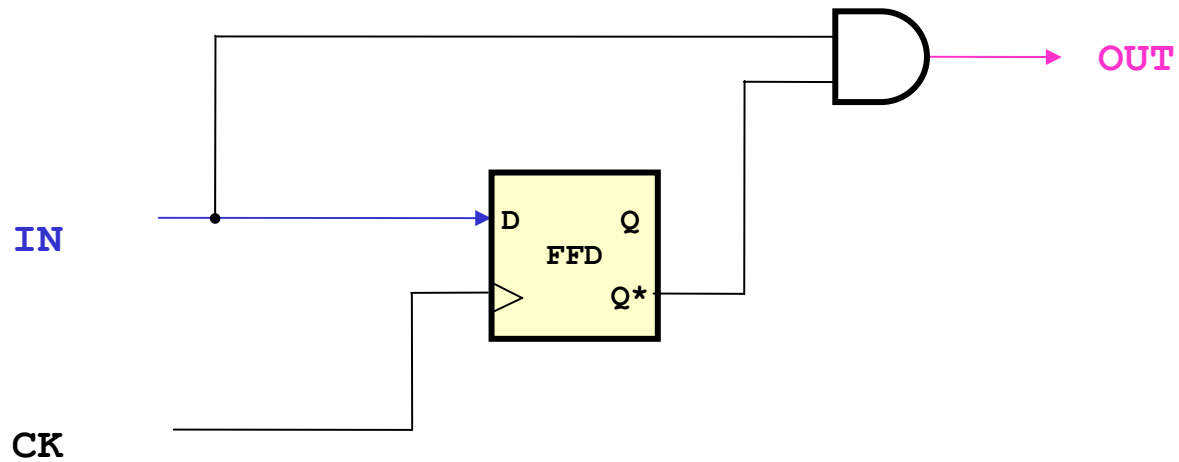
Si noti che il segnale di ingresso potrebbe anche essere non sincrono (purché rispetti tempi di *setup* e *hold*)



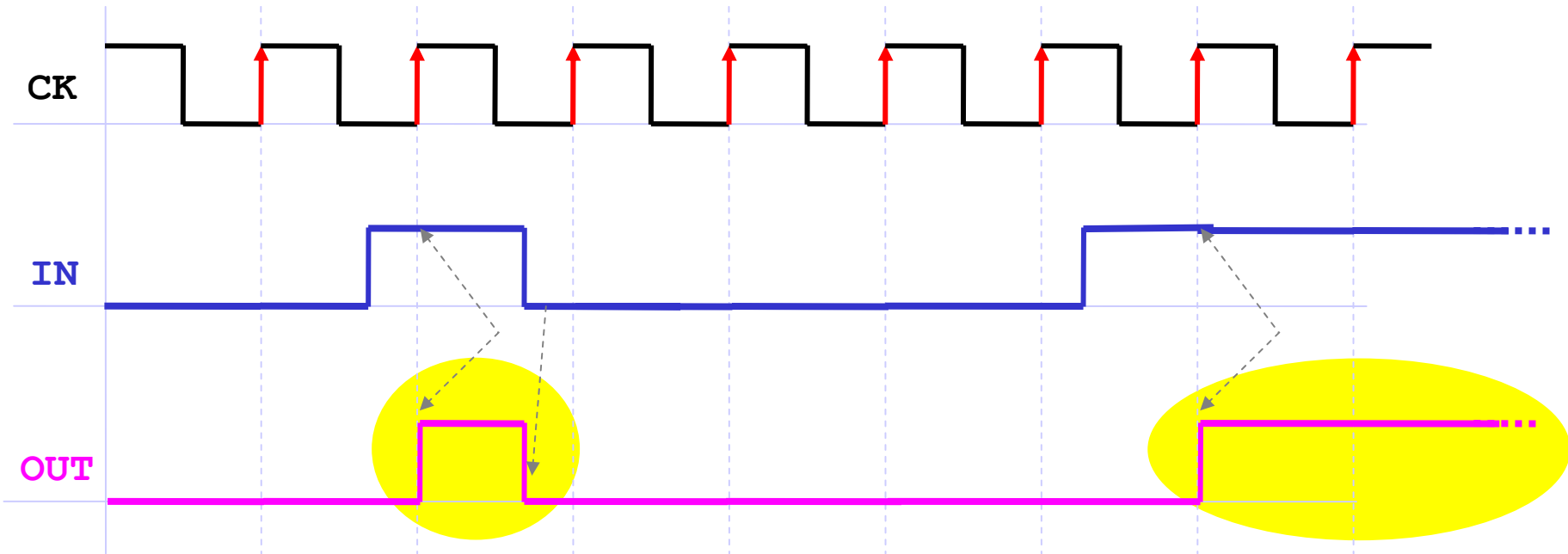
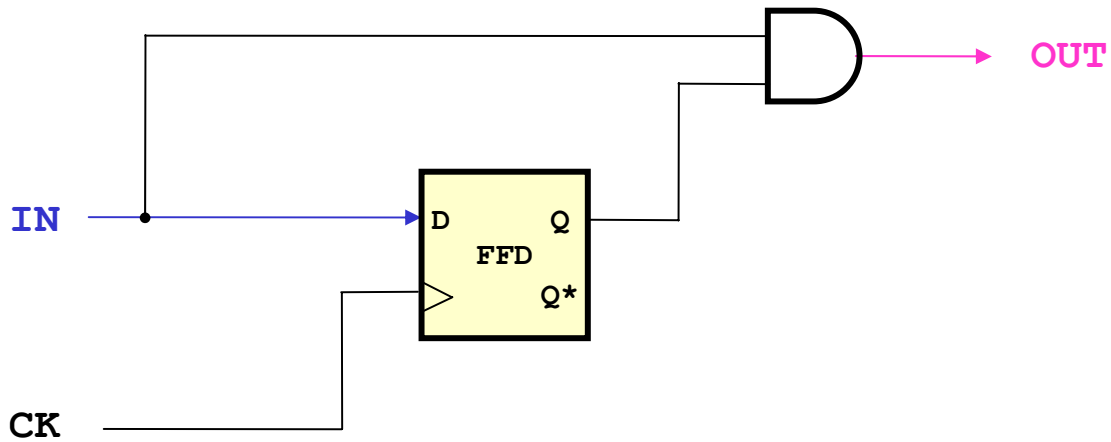
Soluzione



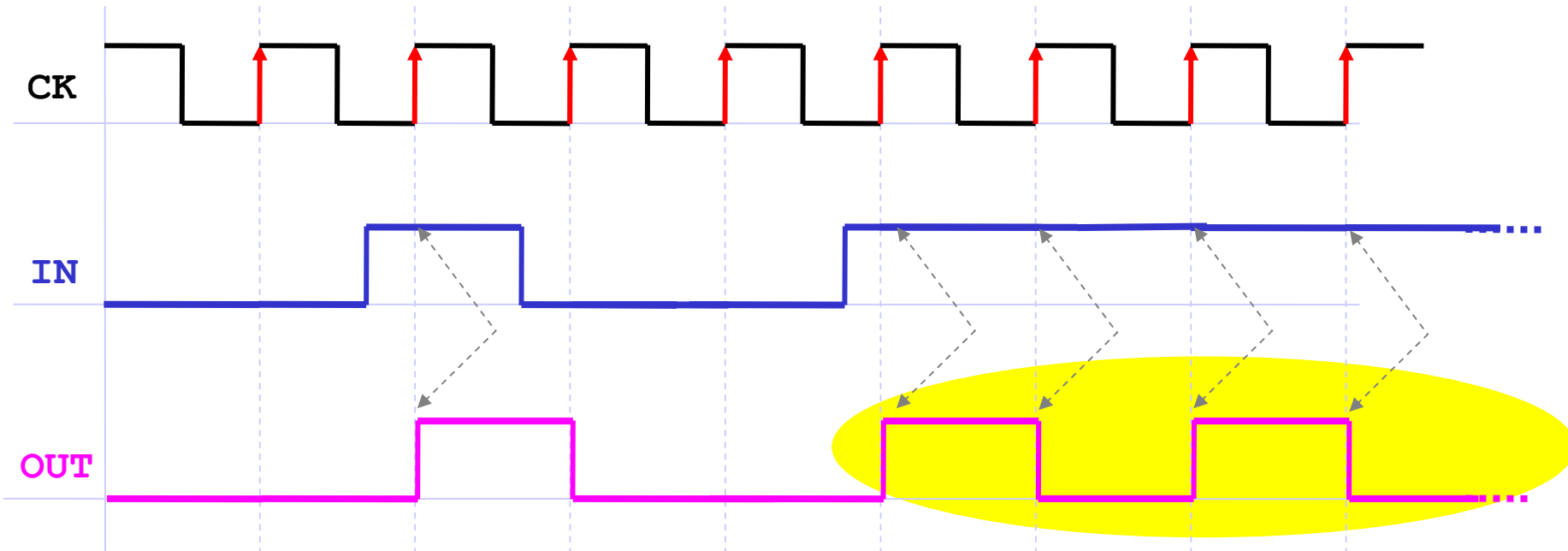
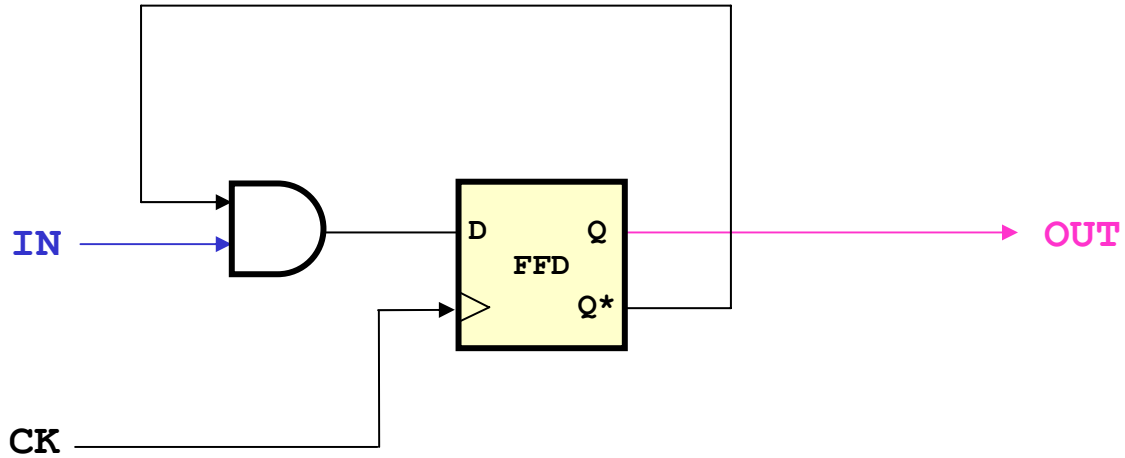
Perchè questa soluzione è sbagliata (1) ?



Perchè questa soluzione è sbagliata (2) ?

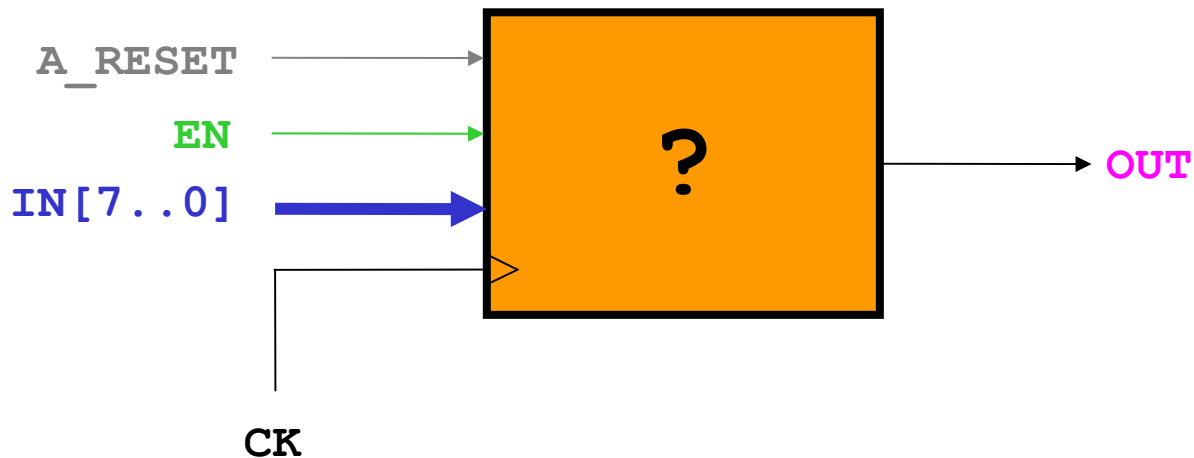


Perchè questa soluzione è sbagliata (3) ?

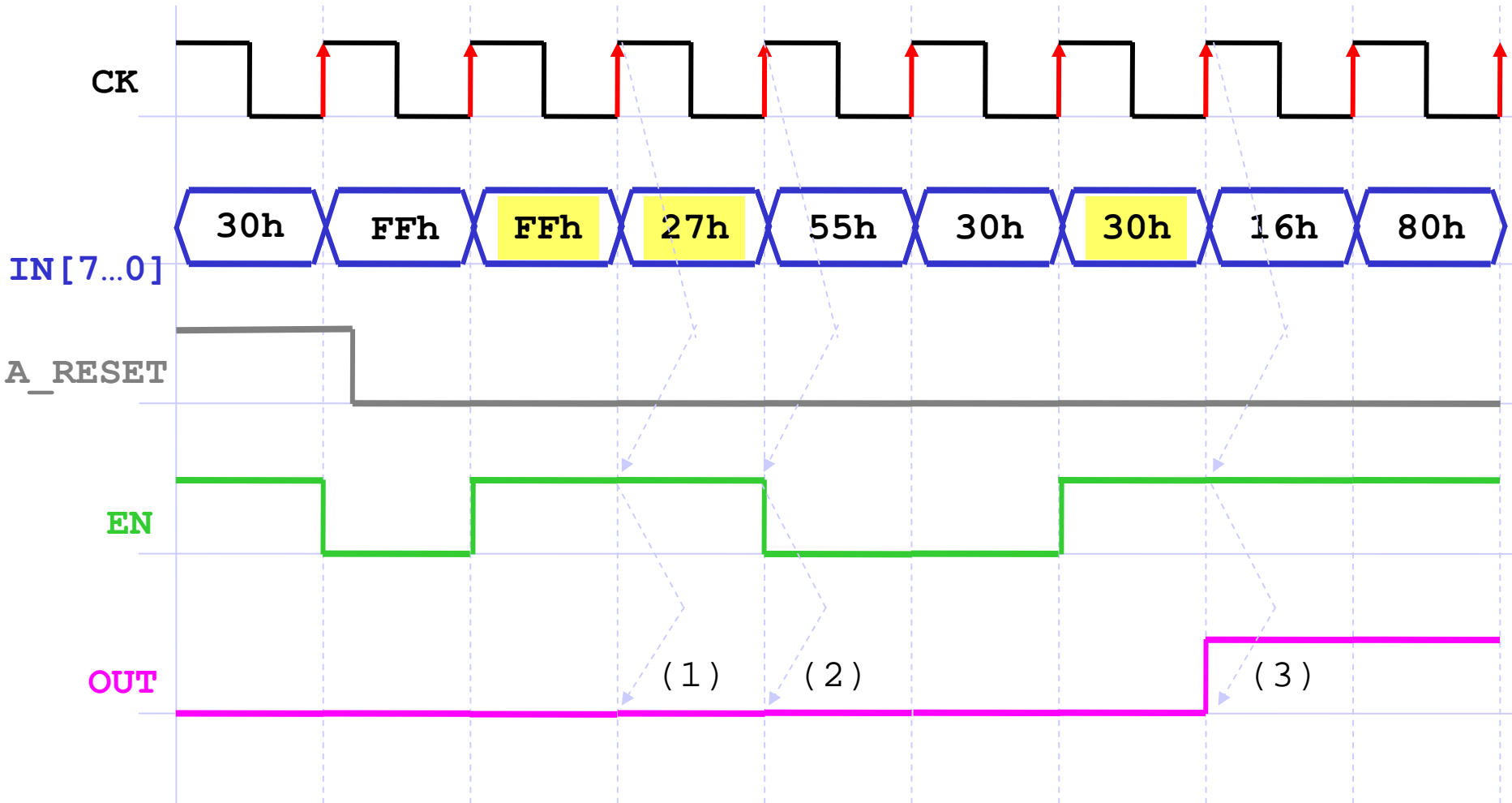


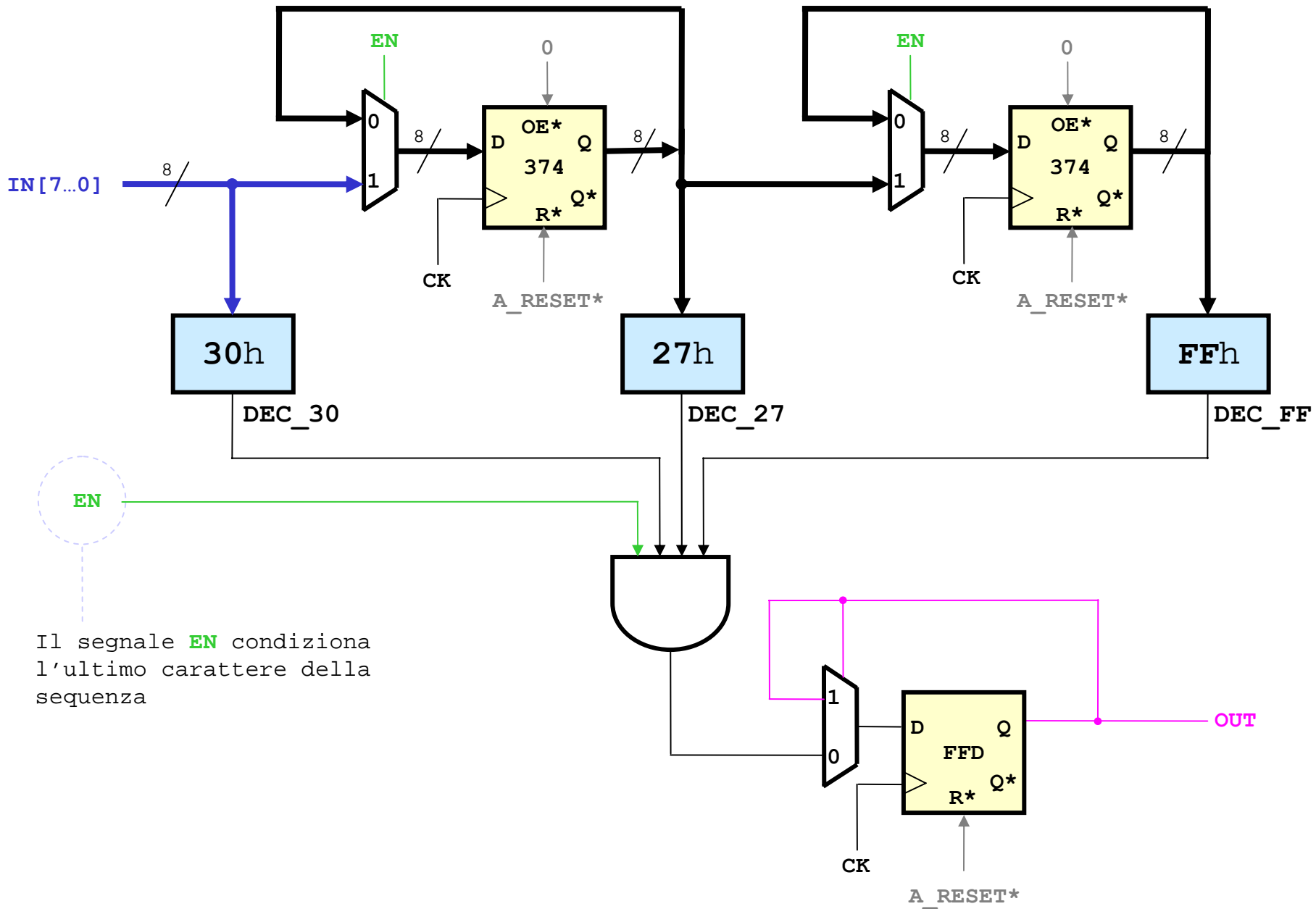
Esercizio 6

Progettare un rete che controlla se gli ultimi tre caratteri che si sono presentati sull'ingresso **IN[7..0]** mentre il segnale **EN** era a livello logico 1 sono stati **FFh** (primo carattere della sequenza), **27h** e **30h**. Nel caso sia rilevata la sequenza **FF-27-30**, nel periodo di clock successivo a quello dell'ultimo carattere ricevuto (**30h**), deve essere asserita l'uscita **OUT** e rimanere tale fino a che non viene asserito il segnale (asincrono) di reset **A_RESET**. In seguito ad un reset deve riprendere immediatamente il controllo della sequenza in ingresso.



Soluzione 6.1

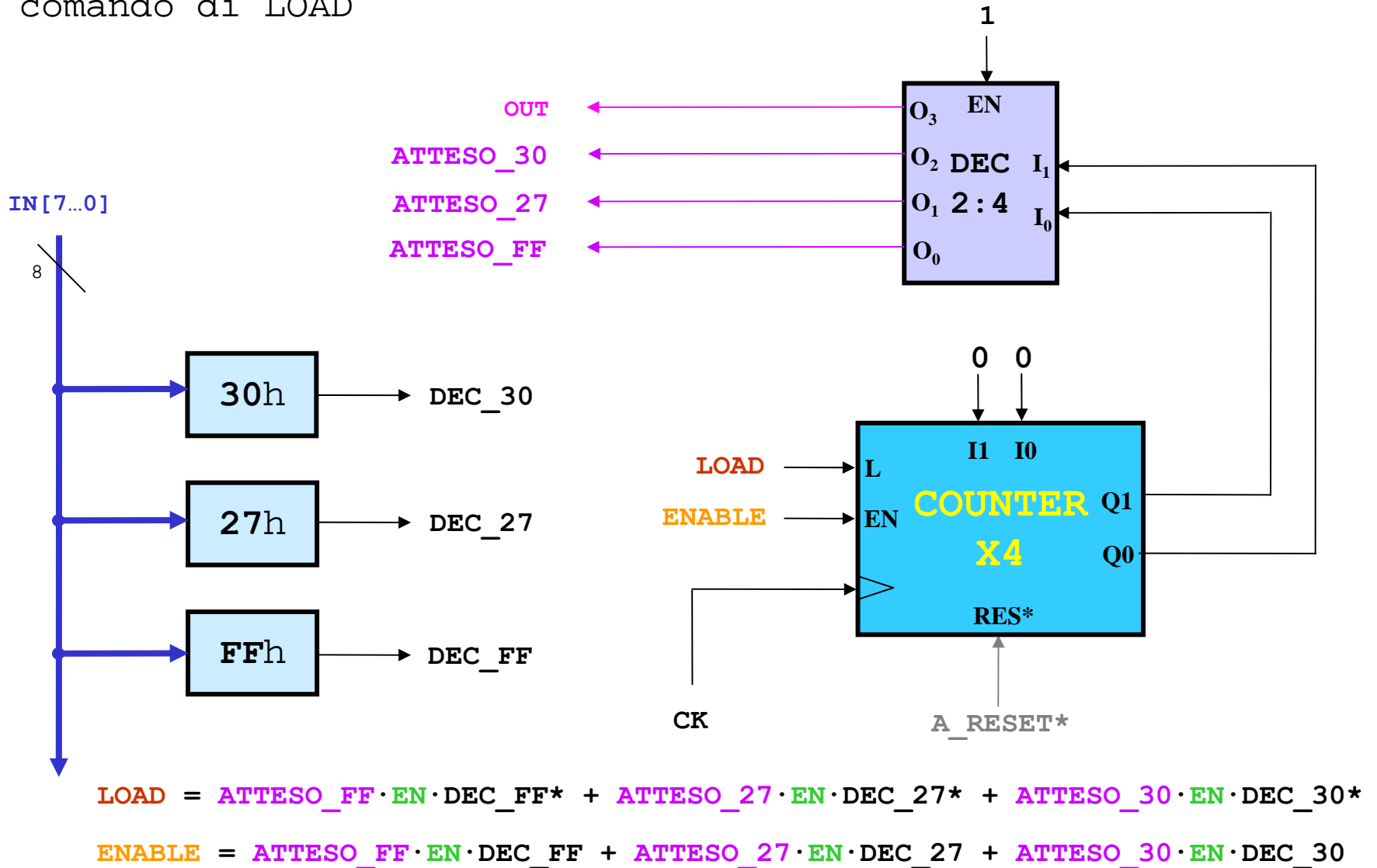




Il segnale EN condiziona l'ultimo carattere della sequenza

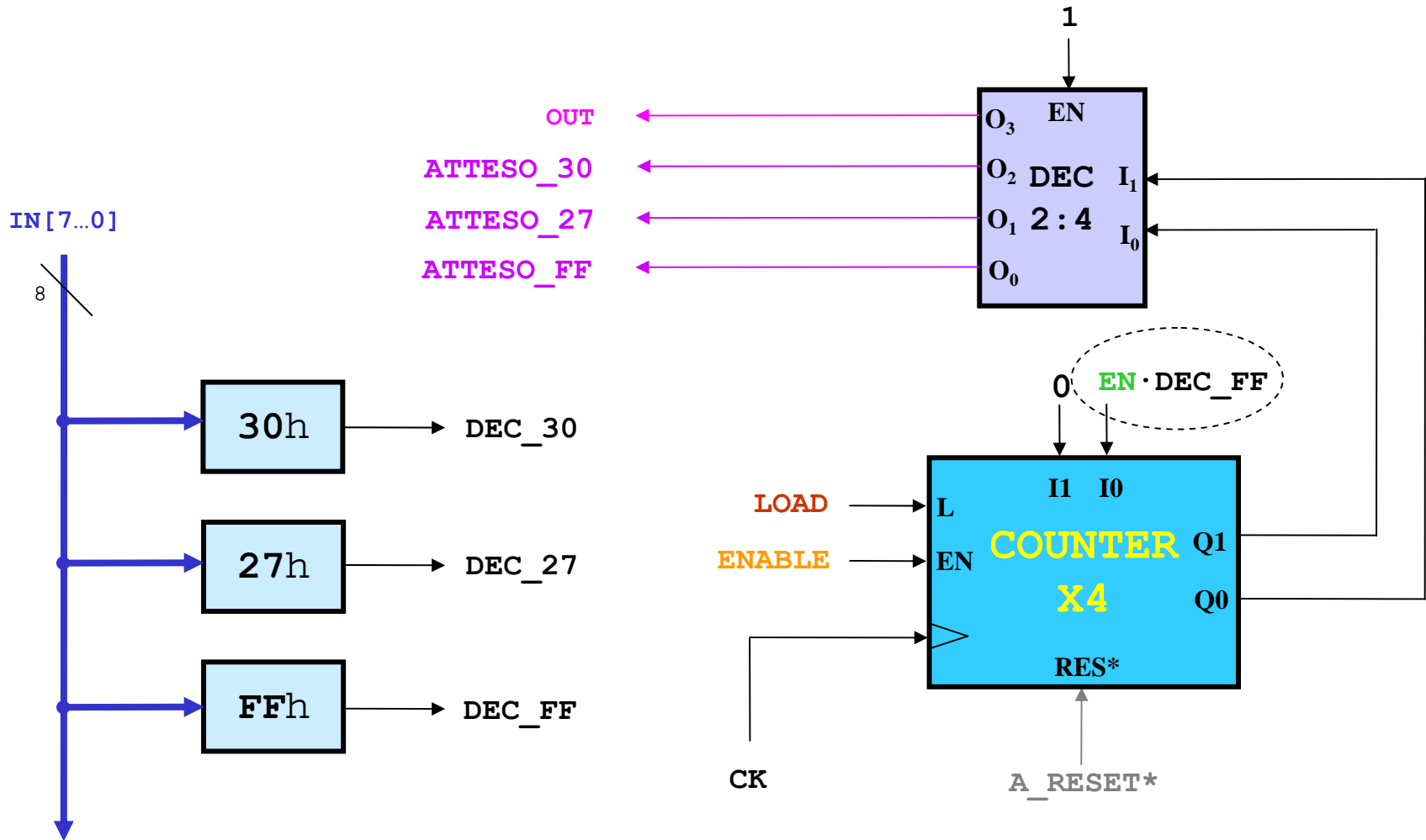
Soluzione 6.2

Una soluzione alternativa utilizzando un contatore dotato di comando di LOAD



C'è un problema...

.. nella soluzione della pagina precedente cosa accade se i caratteri ricevuti (con **EN=1**) sono **FF-FF-27-30** ?



$$\text{LOAD} = \text{ATTESO_FF} \cdot \text{EN} \cdot \text{DEC_FF}^* + \text{ATTESO_27} \cdot \text{EN} \cdot \text{DEC_27}^* + \text{ATTESO_30} \cdot \text{EN} \cdot \text{DEC_30}^*$$

$$\text{ENABLE} = \text{ATTESO_FF} \cdot \text{EN} \cdot \text{DEC_FF} + \text{ATTESO_27} \cdot \text{EN} \cdot \text{DEC_27} + \text{ATTESO_30} \cdot \text{EN} \cdot \text{DEC_30}$$

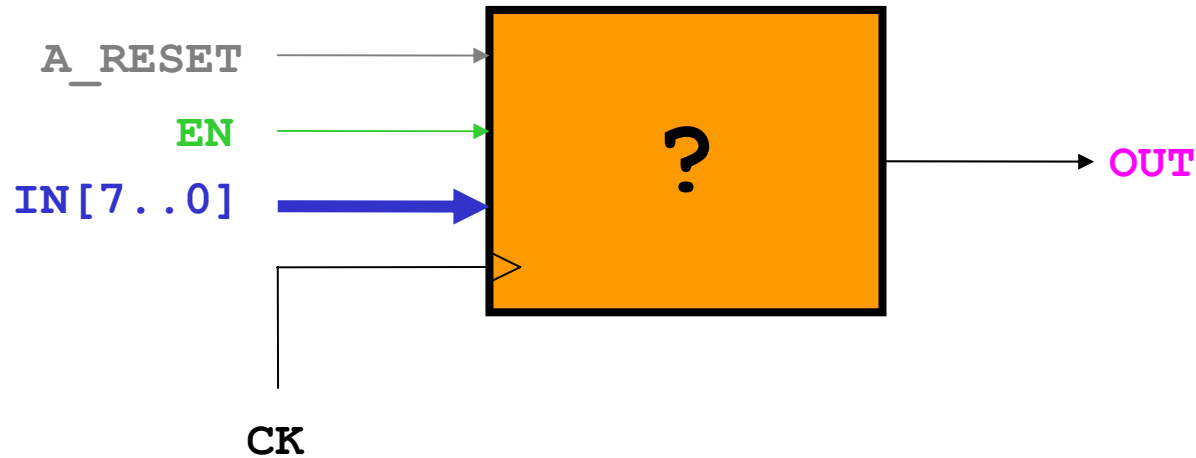
Esercizi

E5-1) Riprogettare la rete dell'esercizio 5 in modo che **OUT** assuma il valore logico 1 in seguito alla ricezione anche non consecutiva (con **EN=1**) dei caratteri **FFh**, **27h** e **30h**.

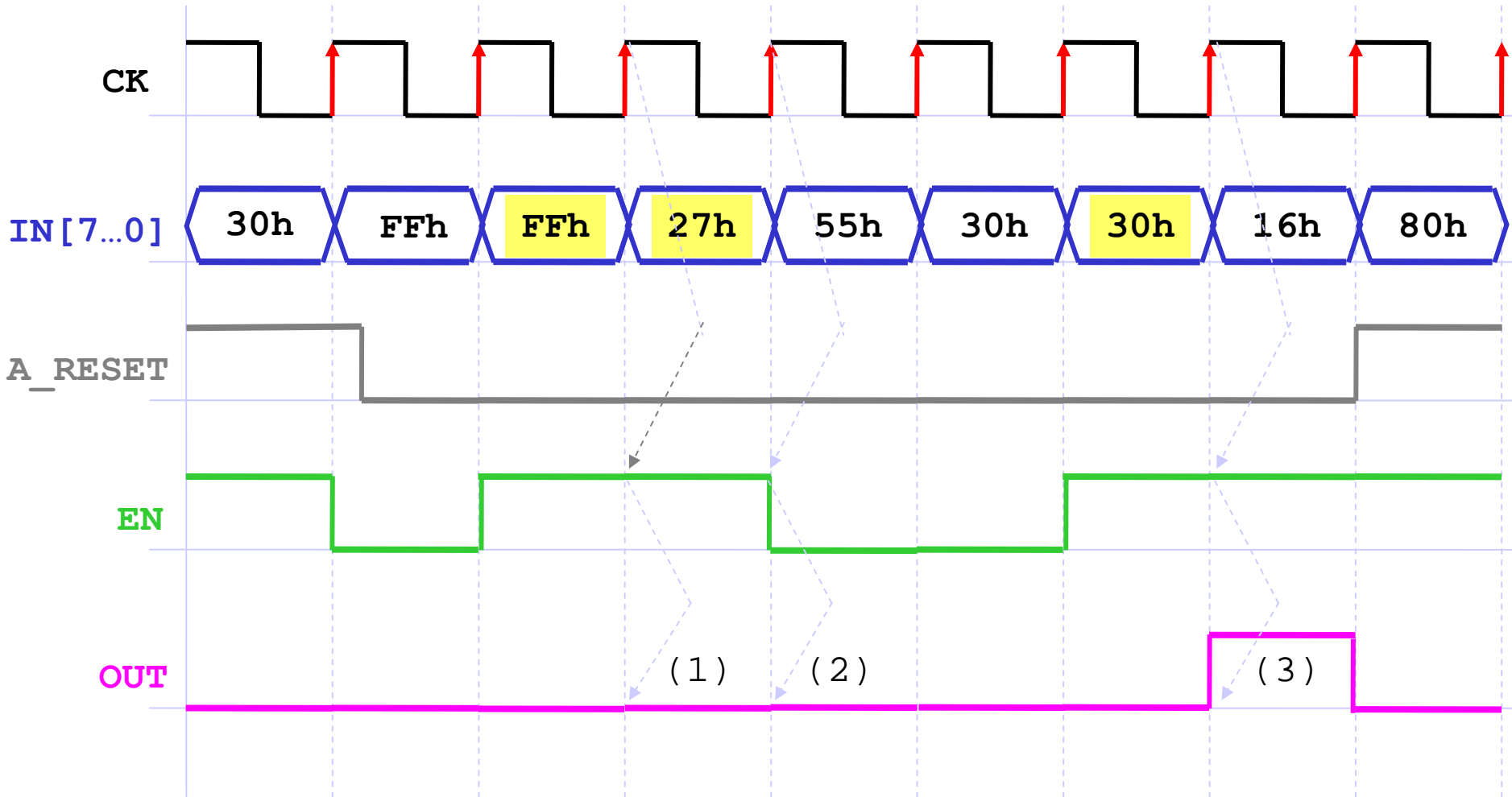
Ad esempio, **OUT=1** se i caratteri ricevuti (mentre **EN=1**) sono stati: **FF-7A-80-9F-27-B2-30-...**

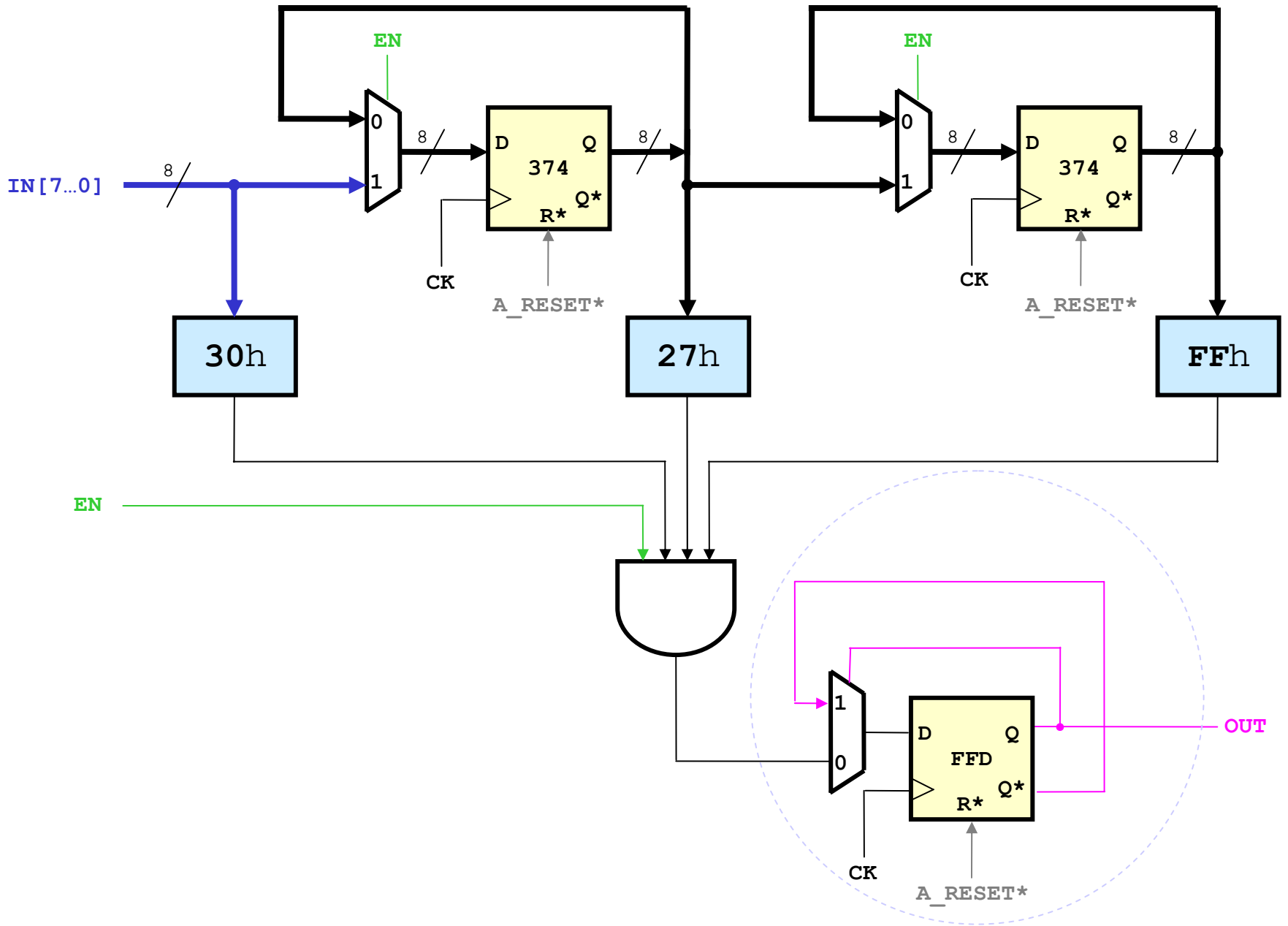
Esercizio 7

Modificare l'esercizio precedente in modo che, in seguito al rilevamento della sequenza, l'uscita **OUT** assuma il valore logico 1 per un solo periodo di clock. Appena ricevuta una sequenza completa il controllo dei caratteri in ingresso deve riprendere immediatamente.



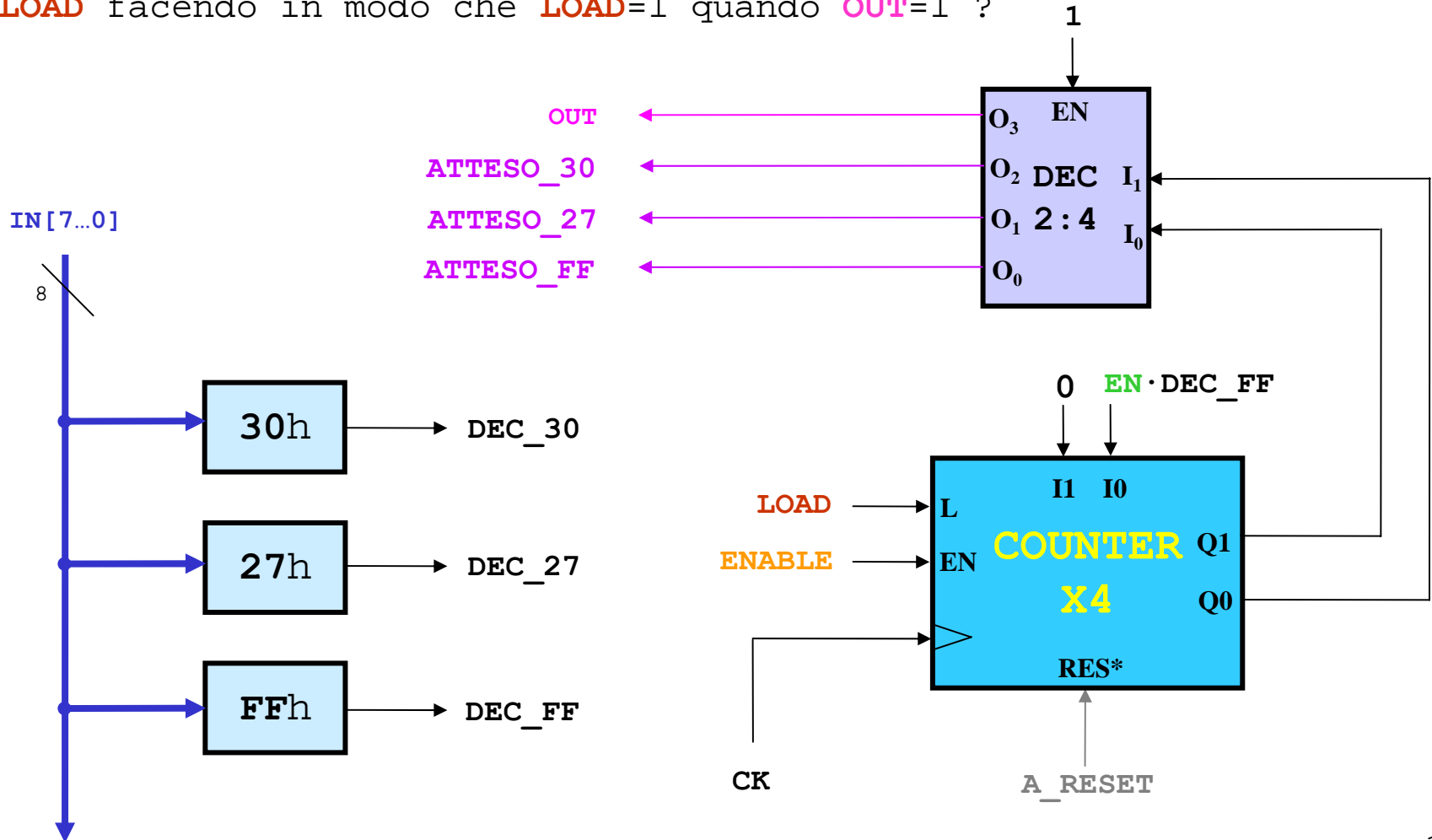
Soluzione 7.1





Soluzione 7.2

Rispetto all'esercizio 5.2 è sufficiente modificare il comando di **LOAD** facendo in modo che **LOAD=1** quando **OUT=1** ?



$$\text{LOAD} = \text{ATTESO_FF} \cdot \text{EN} \cdot \text{DEC_FF}^* + \text{ATTESO_27} \cdot \text{EN} \cdot \text{DEC_27}^* + \text{ATTESO_30} \cdot \text{EN} \cdot \text{DEC_30}^* + \text{OUT}$$

$$\text{ENABLE} = \text{ATTESO_FF} \cdot \text{EN} \cdot \text{DEC_FF} + \text{ATTESO_27} \cdot \text{EN} \cdot \text{DEC_27} + \text{ATTESO_30} \cdot \text{EN} \cdot \text{DEC_30}$$

Cosa accade se (con **EN=1**) la sequenza è **45-FF-27-30-FF-27-30-...** ?

Esercizi

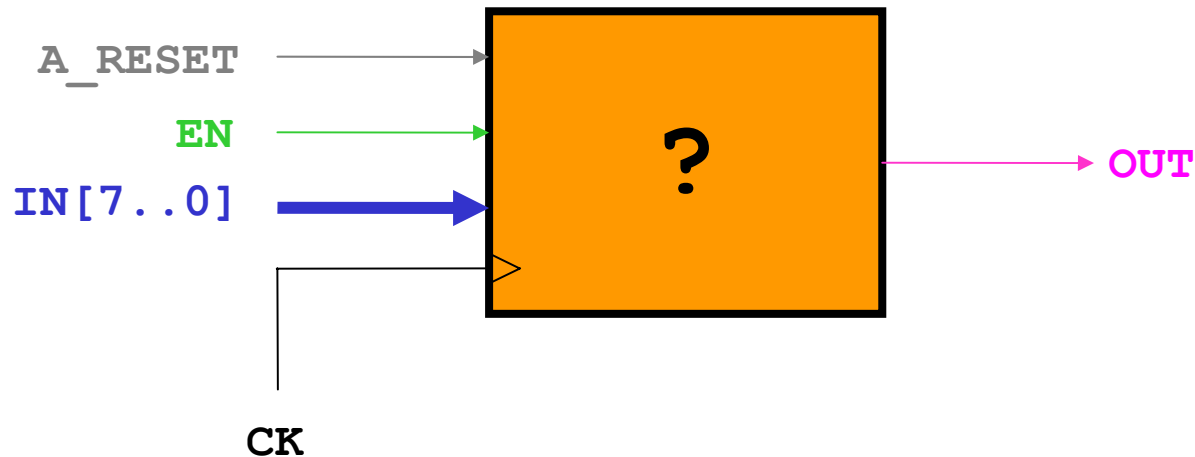
E6-1) Riprogettare la rete dell'esercizio 6 in modo che **OUT=1** in seguito alla ricezione anche non consecutiva (con **EN=1**) dei caratteri **FFh**, **27h** e **30h**.

Ad esempio, **OUT=1** se i caratteri ricevuti mentre **EN=1** sono stati: **FF-7A-80-9F-27-B2-30-...**

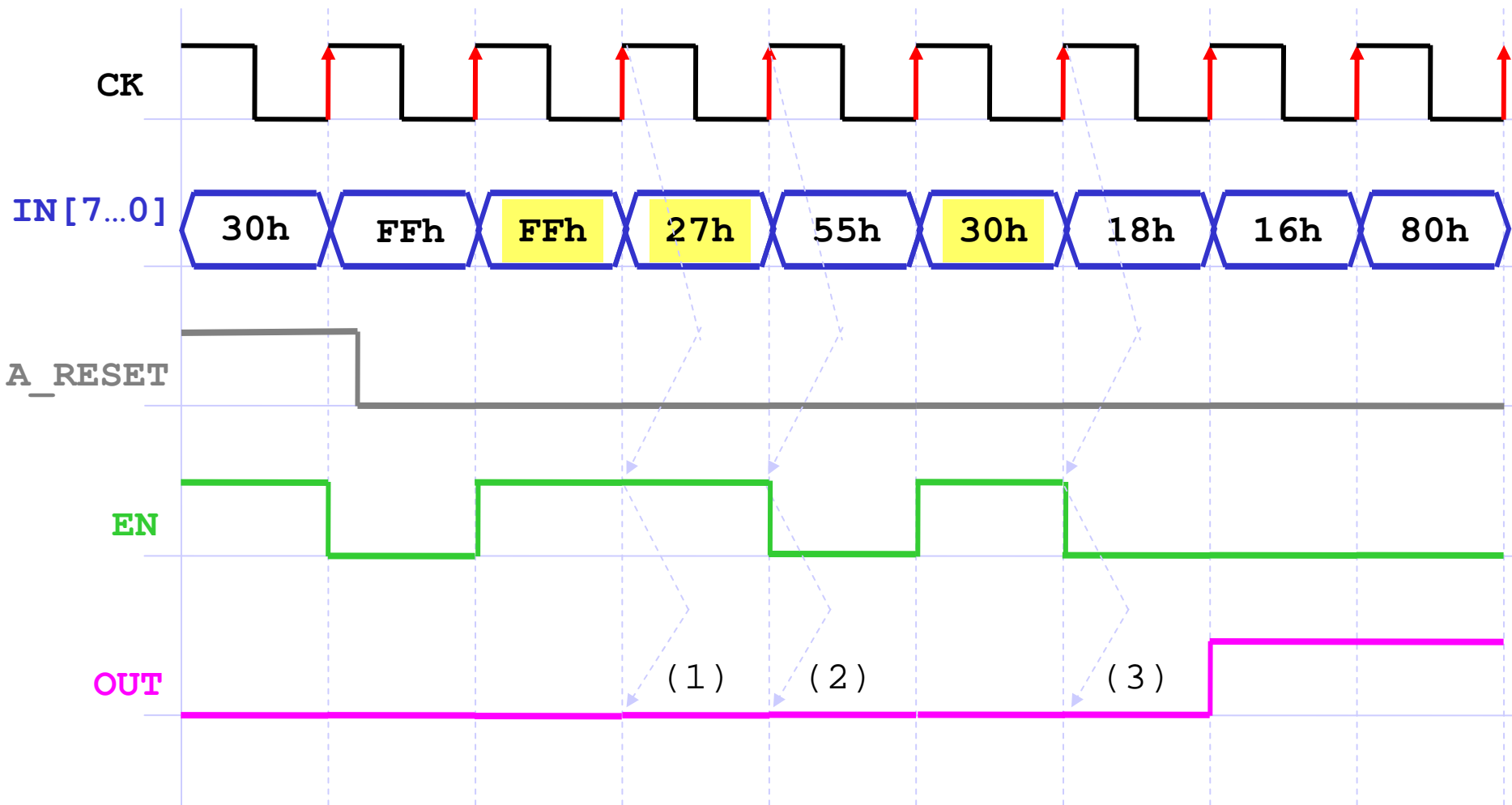
E6-2) Cosa accade alle soluzioni 6.1 e 6.2 se (mentre **EN=1**) la sequenza è: **45-FF-27-30-FF-27-30-... ?**

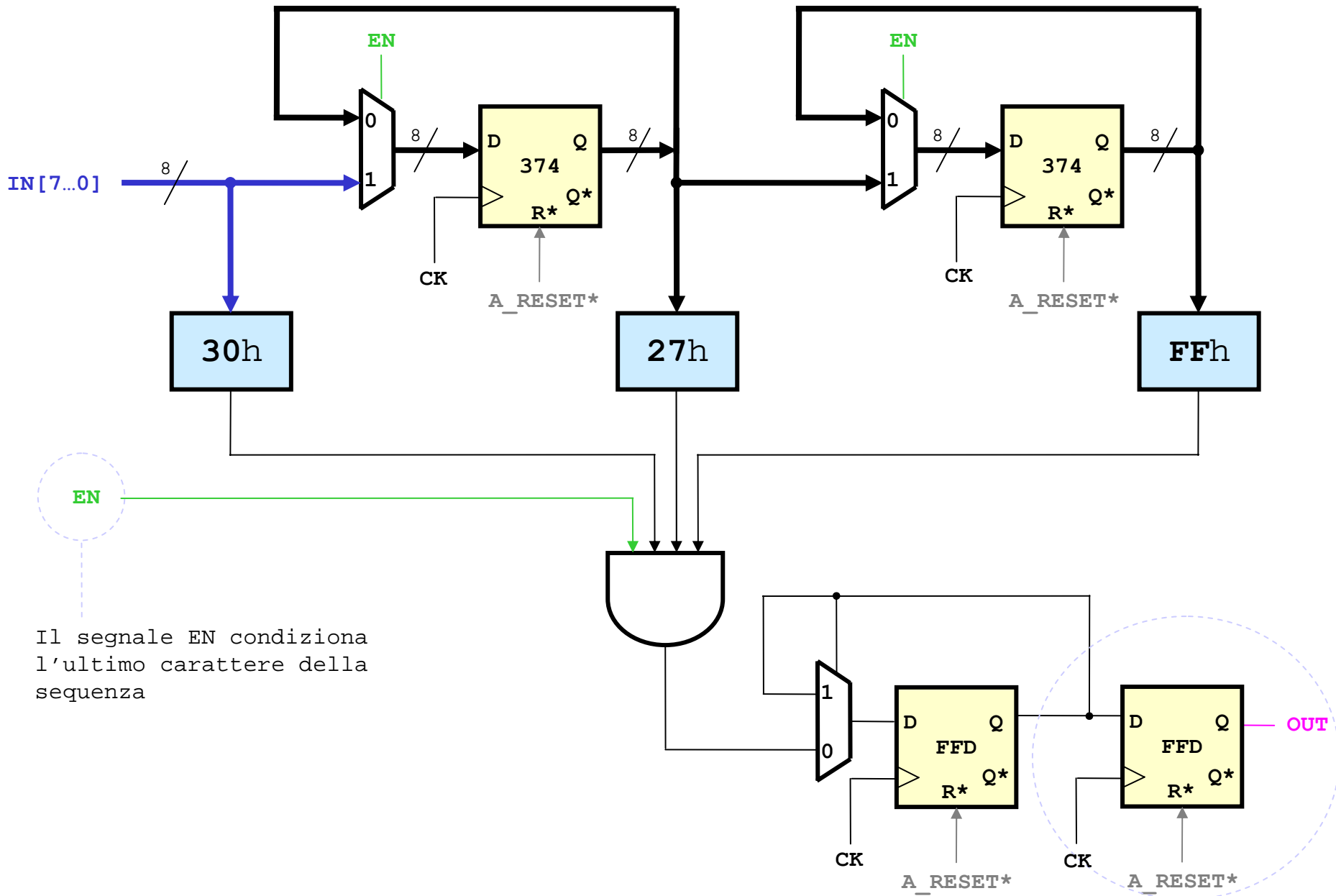
Esercizio 8

Progettare un rete che controlla se gli ultimi tre caratteri che si sono presentati in ingresso **IN[7..0]** mentre il segnale **EN=1** sono stati **FFh** (primo carattere della sequenza), **27h** e **30h**. Nel caso sia rilevata tale sequenza, **due periodi di clock successivi** a quello dell'ultimo carattere della sequenza ricevuto deve essere asserita l'uscita **OUT** e rimanere tale fino a che il segnale di reset (asincrono) **A_RESET** non assume il valore logico 1.



Soluzione

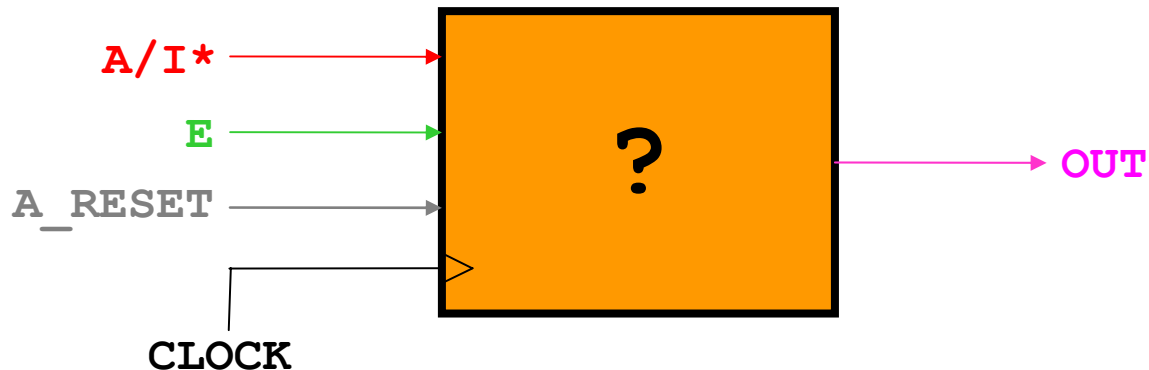




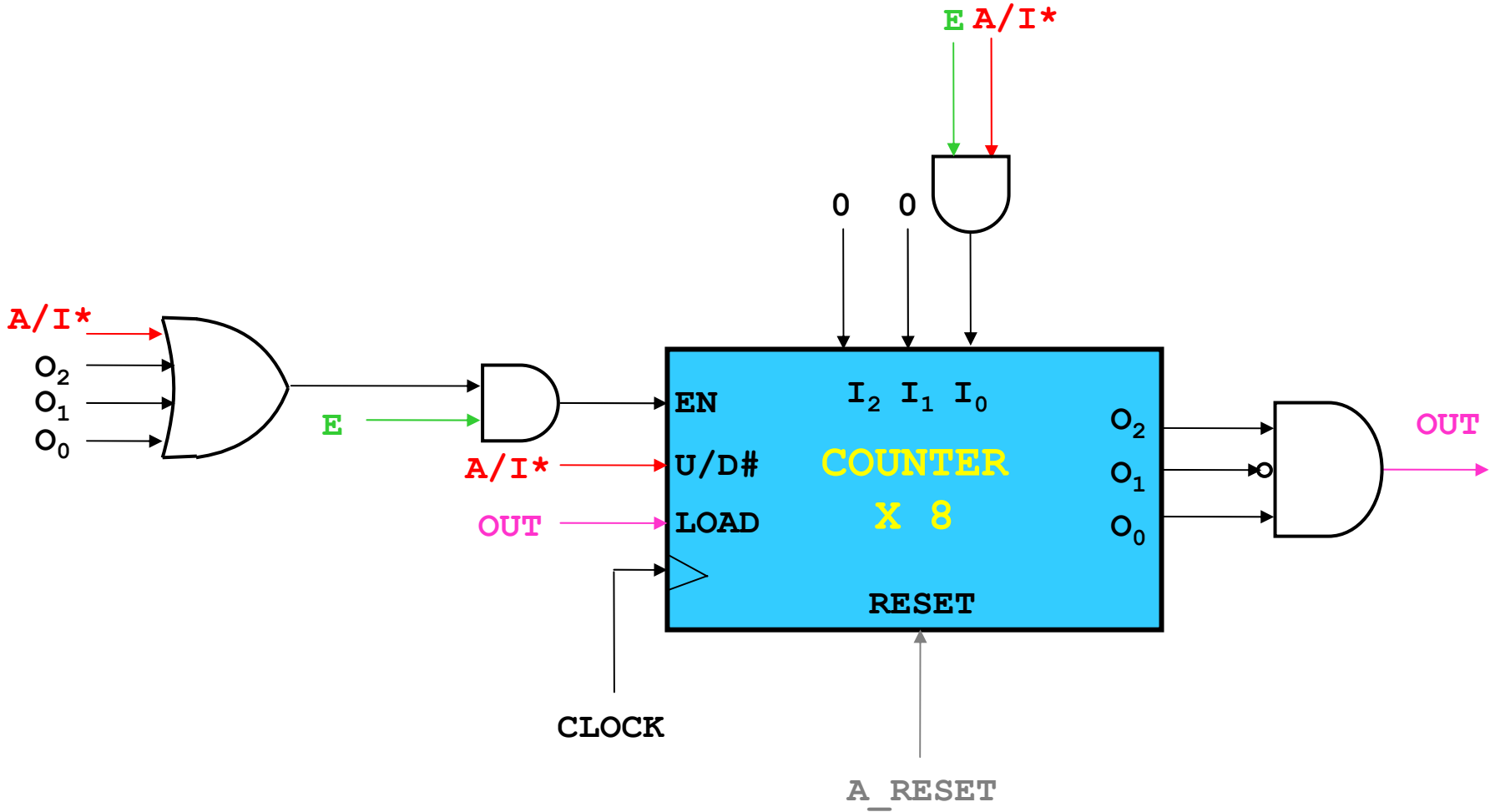
Il segnale EN condiziona l'ultimo carattere della sequenza

Esercizio 9

Progettare una rete dotata di tre ingressi **E**, **A/I***, **RESET** e un'uscita **OUT**. Il segnale di ingresso **A/I*** influisce sulla rete solo se contemporaneamente **E=1**. L'uscita della rete deve andare al livello logico 1 per un periodo di clock se viene rilevato per cinque volte, anche non consecutive, il valore 1 del segnale **A/I*** in presenza del segnale **E=1**. Ogni volta che il segnale **A/I*** vale 0 (con **E=1**) deve essere decrementato di uno il numero di eventi rilevati fino a quel momento. Successivamente ad un reset (segnale asincrono **A_RESET=1**) o nel caso nessun evento sia stato ancora rilevato la rete deve rimanere nello stato 000 anche se **A/I*=0** ed **E=1**. Dopo avere rilevato cinque eventi la rete deve riprendere il conteggio dallo stato 000.

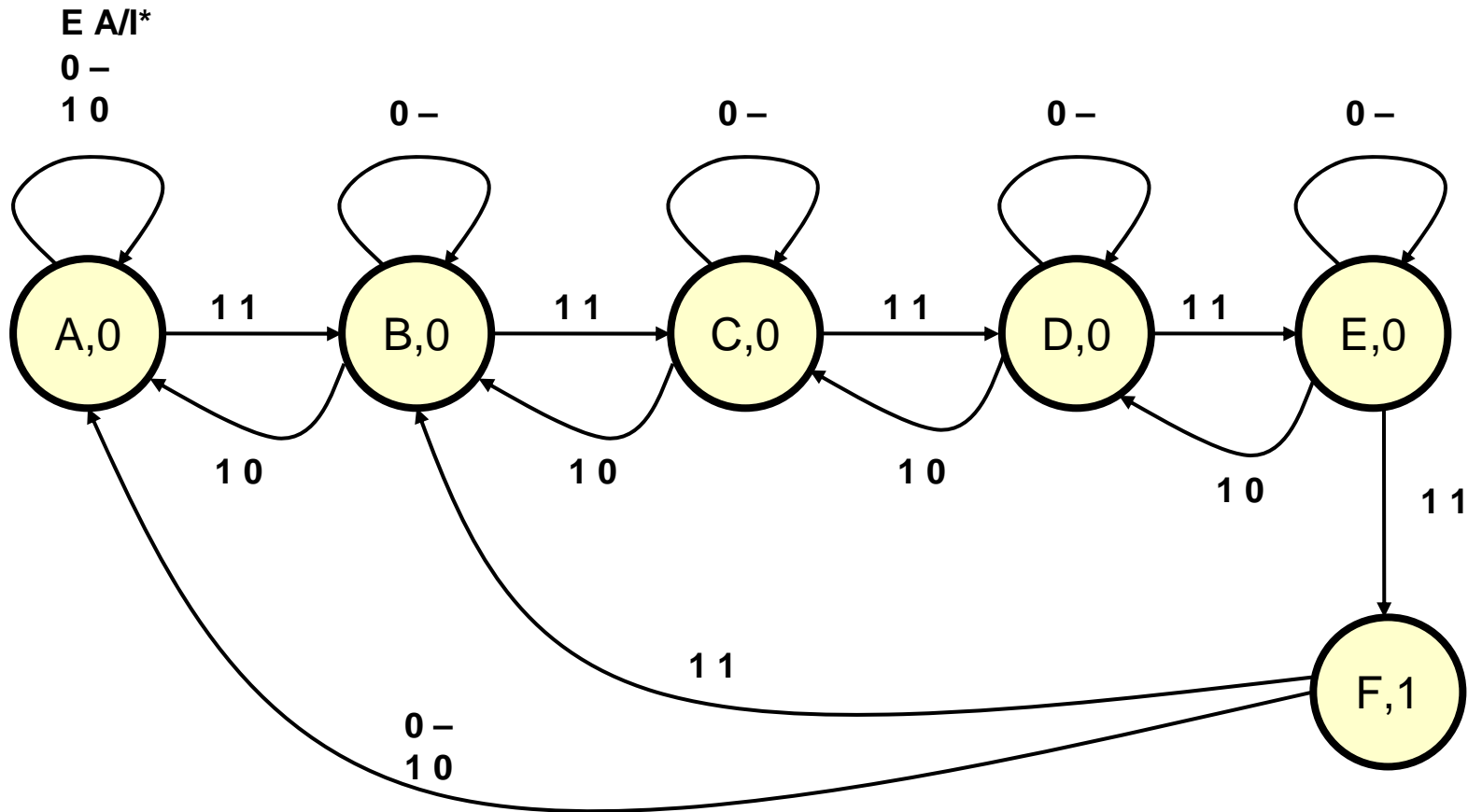


Soluzione 9.1



Soluzione 9.2

Soluzione mediante sintesi formale: grafo \rightarrow tabella di flusso \rightarrow tabella delle transizioni, ... NON SI USA !!!!



Esercizio 10

Utilizzando un microprocessore dotato di un bus indirizzi a 16 bit e di un bus dati a 8 bit: mappare nello parte bassa dello spazio di indirizzamento 12k di RAM e nella parte alta 16k di EPROM.

Soluzione

$A_{15} \dots A_{12}$ $A_{11} \dots A_8$ $A_7 \dots A_4$ $A_3 \dots A_0$

0000 0000 0000 0000 (0000h)

0001 1111 1111 1111 (1FFFh)

0010 0000 0000 0000 (2000h)

0010 0111 1111 1111 (27FFh)

0010 1000 0000 0000 (2800h)

0010 1111 1111 1111 (2FFFh)

Segnali di decodifica:

$CS_RAM_1 = A_{15}^* \cdot A_{13}^*$

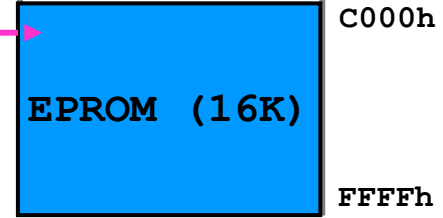
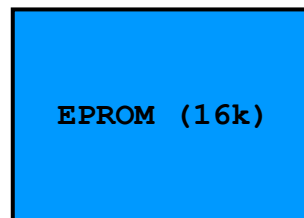
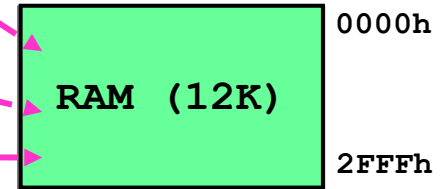
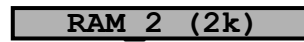
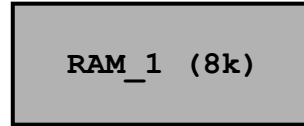
$CS_RAM_2 = A_{15}^* \cdot A_{13} \cdot A_{11}^*$

$CS_RAM_3 = A_{15}^* \cdot A_{13} \cdot A_{11}$

$CS_EPROM = A_{15}$

1100 0000 0000 0000 (C000h)

1111 1111 1111 1111 (FFFFh)

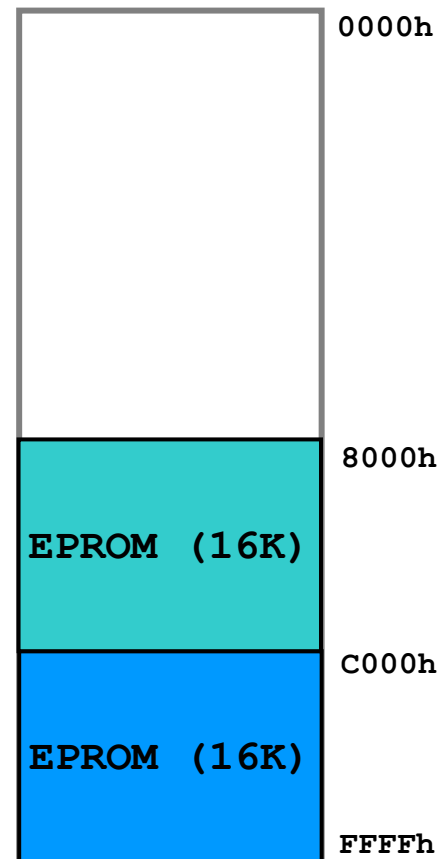
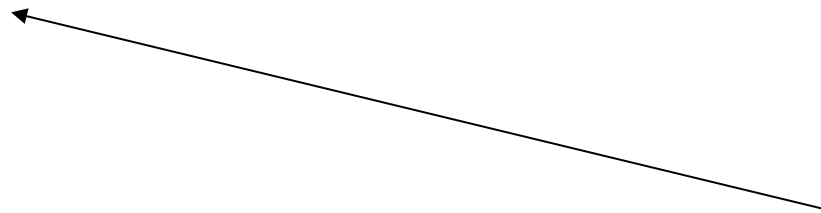


NOTA

- La codifica semplificata implica l'attivazione dei segnali di selezioni anche per indirizzi diversi da quelli in cui sono realmente presenti i dispositivi di memoria.
- Il segnale **CS_EPROM** si attiva per ogni indirizzo maggiore o uguale di **8000h** (seconda metà dello spazio di indirizzamento)

CS_EPROM=A15

Indirizzi di memoria con **A15=1**



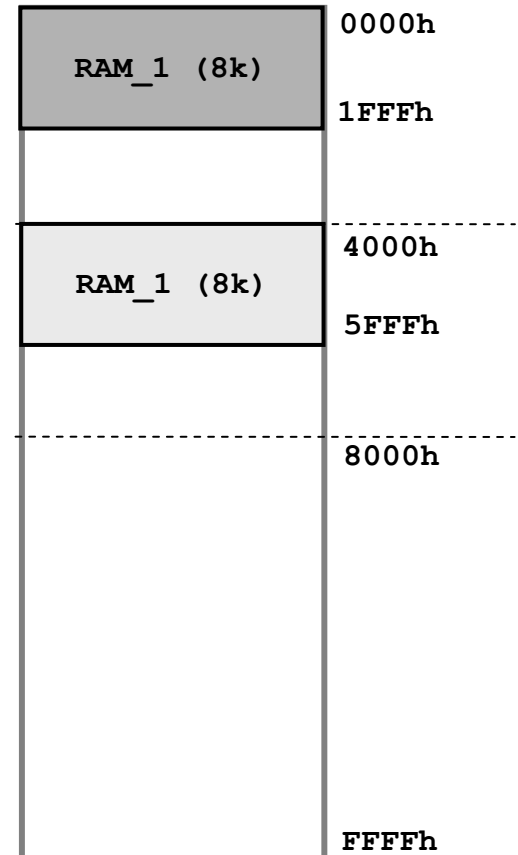
- Il segnale **CS_RAM_1** si attiva per ogni indirizzo compreso tra 0000h e 7FFFh (**A15=0**) per il quale **A13=0**:

$$\text{CS_RAM_1} = \text{A15} * \cdot \text{A13} *$$

Quindi, **CS_RAM_1=1** per entrambi i seguenti intervalli di memoria:

$A_{15} \dots A_{12}$	$A_{11} \dots A_8$	$A_7 \dots A_4$	$A_3 \dots A_0$	
0000	0000	0000	0000	(0000h)
0001	1111	1111	1111	(1FFFh)

$A_{15} \dots A_{12}$	$A_{11} \dots A_8$	$A_7 \dots A_4$	$A_3 \dots A_0$	
0100	0000	0000	0000	(4000h)
0101	1111	1111	1111	(5FFFh)

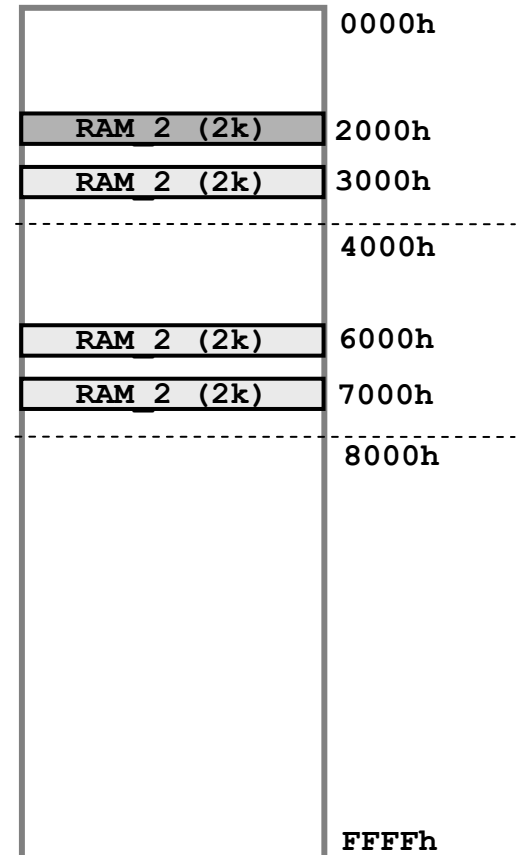


- Il segnale **CS_RAM_2** si attiva per ogni indirizzo compreso tra 0000h e 7FFFh (**A15=0**) per il quale **A13=1** e **A11=0** :

$$\text{CS_RAM_2} = \text{A15} * \text{A13} * \text{A11} *$$

Quindi, **CS_RAM_2=1** per i seguenti quattro intervalli di memoria:

$A_{15} \dots A_{12}$	$A_{11} \dots A_8$	$A_7 \dots A_4$	$A_3 \dots A_0$	
0010	0000	0000	0000	(2000h)
0010	0111	1111	1111	(27FFh)
$A_{15} \dots A_{12}$	$A_{11} \dots A_8$	$A_7 \dots A_4$	$A_3 \dots A_0$	
0011	0000	0000	0000	(3000h)
0011	0111	1111	1111	(37FFh)
$A_{15} \dots A_{12}$	$A_{11} \dots A_8$	$A_7 \dots A_4$	$A_3 \dots A_0$	
0110	0000	0000	0000	(6000h)
0110	0111	1111	1111	(67FFh)
$A_{15} \dots A_{12}$	$A_{11} \dots A_8$	$A_7 \dots A_4$	$A_3 \dots A_0$	
0111	0000	0000	0000	(7000h)
0111	0111	1111	1111	(77FFh)

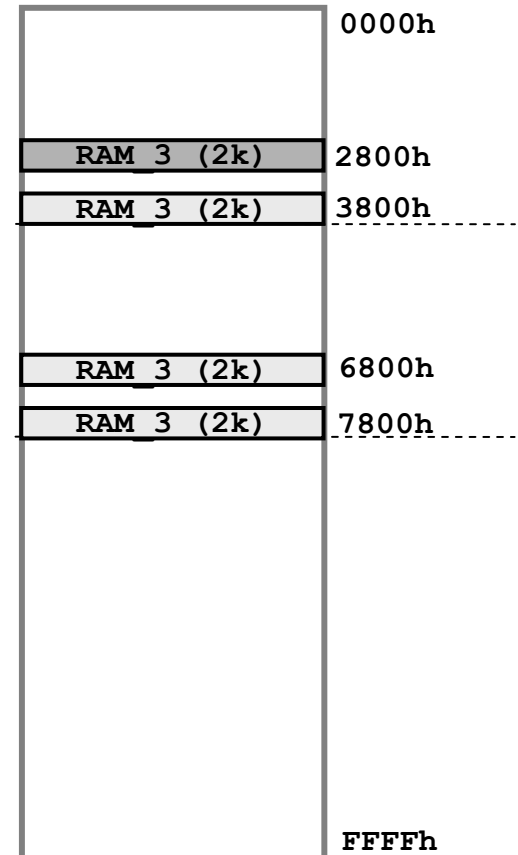


- Il segnale **CS_RAM_3** si attiva per ogni indirizzo compreso tra 0000h e 7FFFh (**A15=0**) per il quale **A13=1** e **A11=1** :

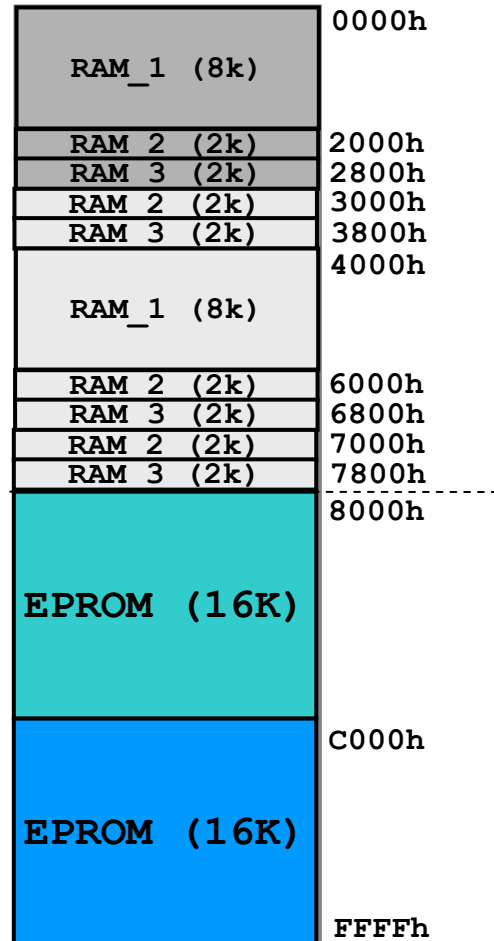
$$\text{CS_RAM_3} = \text{A15} * \text{A13} * \text{A11}$$

Quindi, **CS_RAM_3=1** per i seguenti quattro intervalli di memoria:

A₁₅..A₁₂	A₁₁..A₈	A₇..A₄	A₃...A₀	
0010	1000	0000	0000	(2800h)
0010	1111	1111	1111	(2FFFh)
A₁₅..A₁₂	A₁₁..A₈	A₇..A₄	A₃...A₀	
0011	1000	0000	0000	(3800h)
0011	1111	1111	1111	(3FFFh)
A₁₅..A₁₂	A₁₁..A₈	A₇..A₄	A₃...A₀	
0110	1000	0000	0000	(6800h)
0110	1111	1111	1111	(6FFFh)
A₁₅..A₁₂	A₁₁..A₈	A₇..A₄	A₃...A₀	
0111	1000	0000	0000	(7800h)
0111	1111	1111	1111	(7FFFh)



Effetto di replica nella mappatura in memoria dovuto alla decodifica semplificata. Nella figura seguente sono indicati solo gli indirizzi iniziali.



Esercizio 11

Utilizzando un microprocessore dotato di un bus indirizzi a 20 bit e di un bus dati a 8 bit:

- mappare nello parte bassa dello spazio di indirizzamento 32k di RAM e nella parte alta 32k di EPROM

Nel sistema sono presenti anche due dispositivi di I/O denominati **D1** (dotato di due registri interni) e **D2** (dotato di quattro registri interni):

- mappare **in memoria** anche i due dispositivi di I/O **D1** e **D2** agli indirizzi 2000h e 1000h

Soluzione

RAM: 1 chip da 32KB
RAM (00000h->07FFFh)
CS_RAM = **BA19***·**CS_D1***·**CS_D2***

EPROM: 1 chip da 32KB
EPROM (F8000h - FFFFFh)
CS_EPROM = **BA19**

D1: Mappato in memoria all'indirizzo **02000h**, occupa 2 locazioni (A0) nello spazio di indirizzamento.

CS_D1 = **BA19***·**BA14***·**BA13**·**BA12***·**BA11***·**BA10***·**BA9***·**BA8***·**BA7***·**BA6***·
BA5*·**BA4***·**BA3***·**BA2***·**BA1***

D2: Mappato in memoria all'indirizzo **01000h**, occupa 4 locazioni (A1A0) nello spazio di indirizzamento.

CS_D2 = **BA19***·**BA14***·**BA13***·**BA12**·**BA11***·**BA10***·**BA9***·**BA8***·**BA7***·**BA6***·
BA5*·**BA4***·**BA3***·**BA2***

Esercizio 12

Utilizzando un microprocessore dotato di un bus indirizzi a 20 bit e di un bus dati a 8 bit:

- mappare 32k di RAM nella parte bassa dello spazio di indirizzamento, 32k di RAM a partire dall'indirizzo **1C000h** e 64k EPROM nella parte alta dello spazio di indirizzamento

