

Laboratorio di Architetture degli Elaboratori I
Corso di Laurea in Informatica, A.A. 2021-2022
Università degli Studi di Milano



Esercizi vari

Esercizio 1

- Si realizzi un contatore ciclico in modulo $n=8$:
 - Il contatore va da 0 a $n-1$ incrementando il suo valore di 1 ad ogni ciclo di clock
 - Il valore successivo a $n-1$ nella sequenza è 0

Esercizio 1

- Sintetizziamo la rete combinatoria “add mod 8” che, dato lo stato del contatore al tempo t

$$s(t) = c$$

determini lo stato del contatore al tempo t+1:

$$s(t + 1) = c + 1 \pmod{8}$$

$s(t+1)$ verrà scritto nel contatore in **retroazione**

Esercizio 1

- I tre bit dello stato codificano il valore corrente del contatore
- Ad ogni ciclo di clock il valore (stato) passa al valore successivo, tranne il valore 7 che passerà a 0

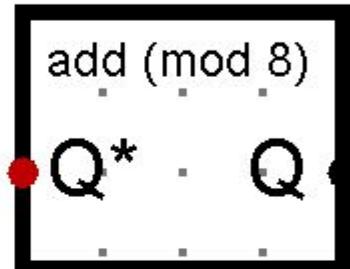
Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0



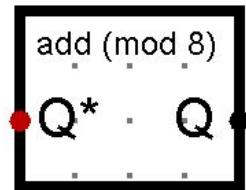
$$Q_0^* = \neg Q_0$$

$$Q_1^* = Q_0 \text{ XOR } Q_1$$

$$Q_2^* = (Q_0 Q_1) \text{ XOR } Q_2$$



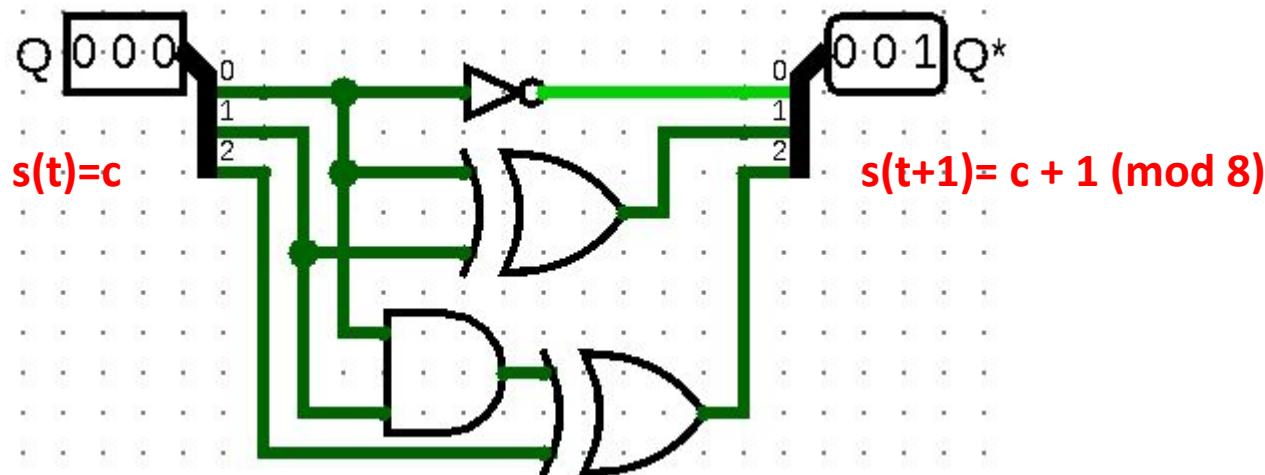
Esercizio 1



$$Q_0^* = \neg Q_0$$

$$Q_1^* = Q_0 \text{ XOR } Q_1$$

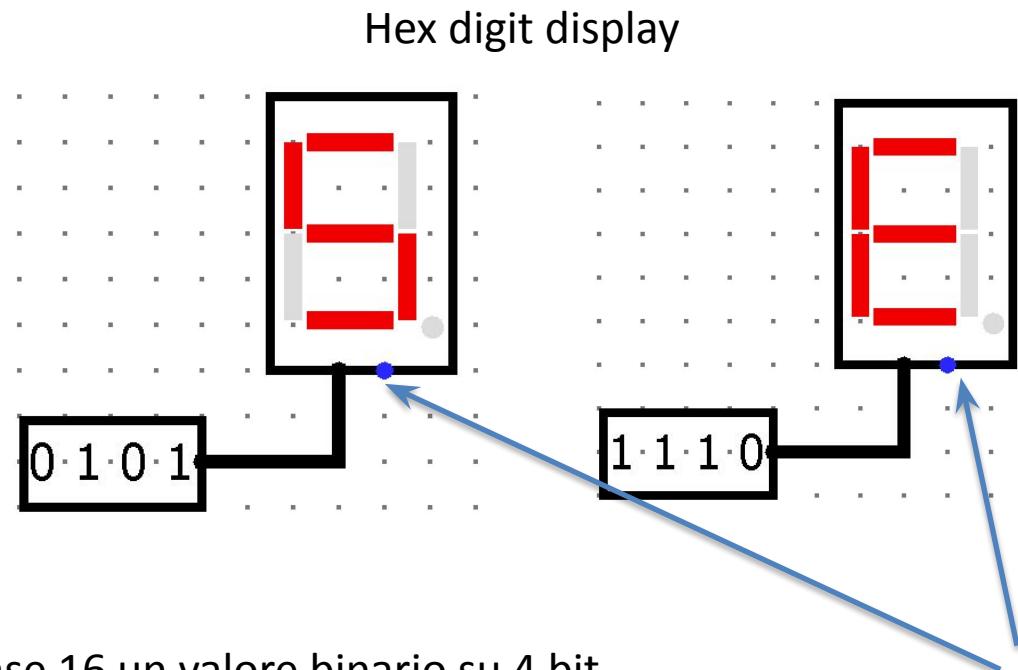
$$Q_2^* = (Q_0 Q_1) \text{ XOR } Q_2$$



NOTA: nella rappresentazione esterna del circuito la posizione di Q e Q^* sono scambiate di lato

Esercizio 1

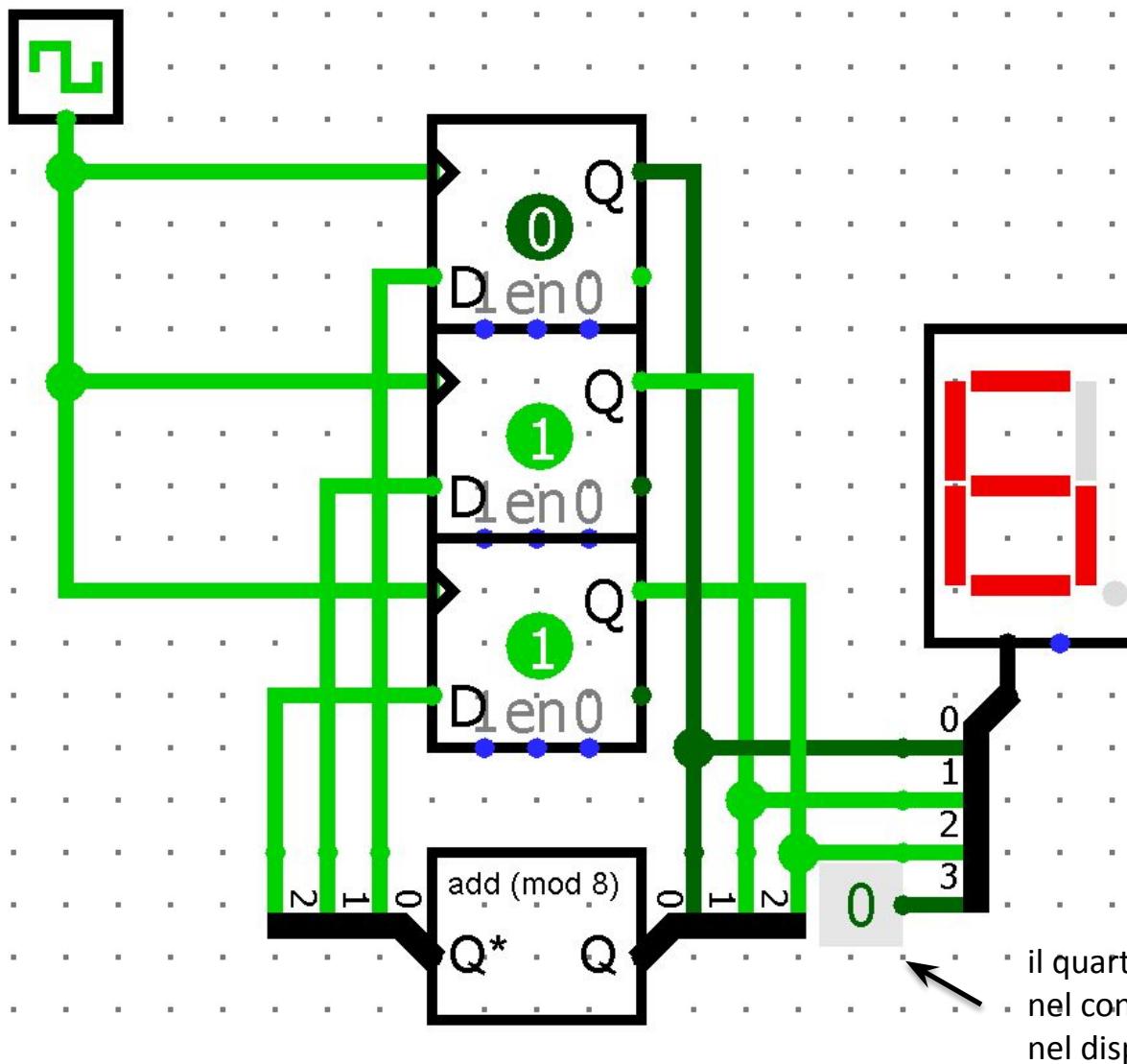
- Utilizziamo questo componente per visualizzare in modo “human-friendly” il valore corrente del registro



- Visualizza in base 16 un valore binario su 4 bit

Bit per accendere/spegnere
il punto decimale (se
undefined è spento)

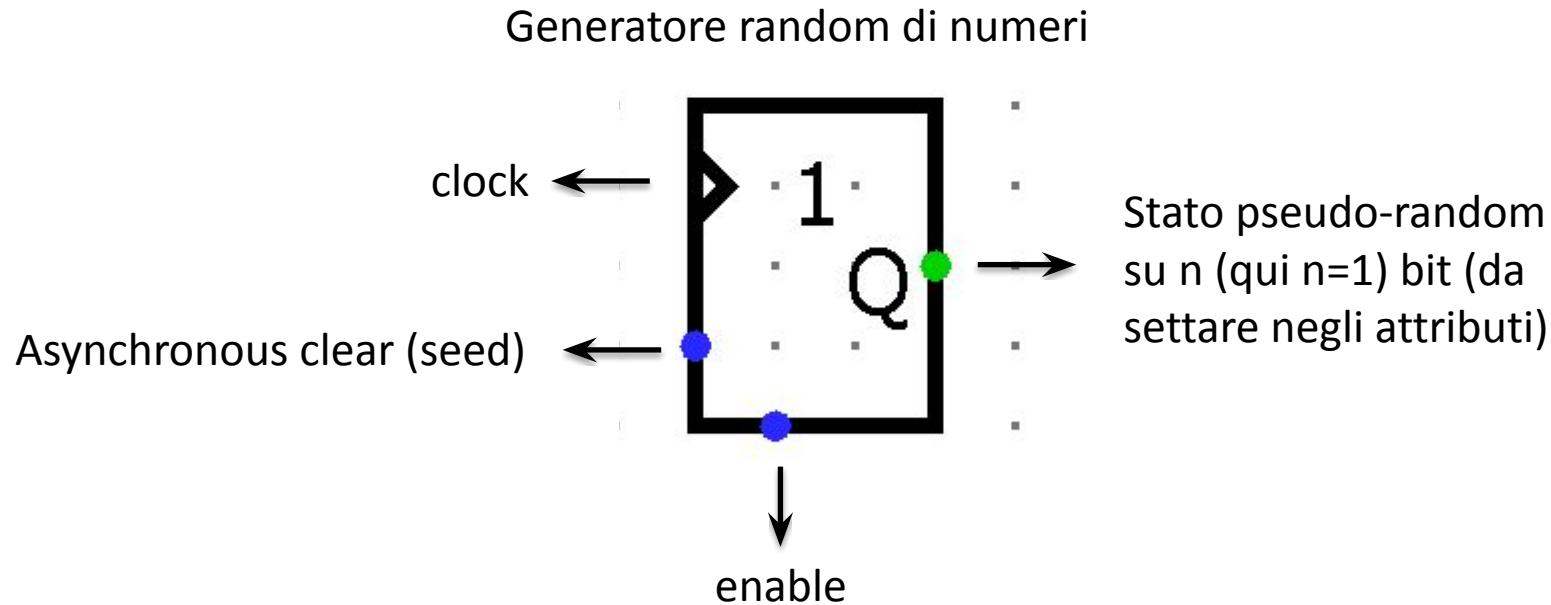
Esercizio 1



Esercizio 2

- Costruire un orologio digitale (mm:ss)
 - Visualizzare l'orario utilizzando un display esadecimale per ogni cifra

Generatore Random



- Stato è su **n** bit
- A ogni ciclo di clock effettua una transizione verso uno dei possibili 2^n stati
- La sequenza seguita è *pseudorandom*: significa che nella realtà è deterministica (ciclica e fissata da un valore iniziale che chiamiamo **seed**), ma che agli occhi di un osservatore attento è difficilmente prevedibile (quindi *sembra* random)

Generatore Random

- Come si realizza un generatore random di numeri?
- Uno degli approcci più usati e anche facili da implementare è l'uso di un tipo particolare di registri chiamati **Linear Feedback Shift Register** (LFSR)

LFSR

Linear Feedback Shift Register

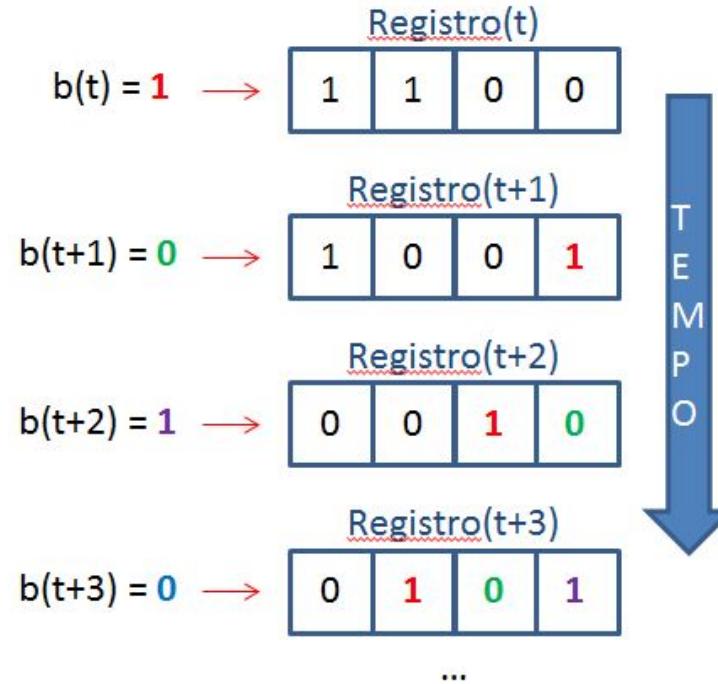


- E' un registro a scorrimento, un componente che già conosciamo

Ad ogni istante di tempo diamo in input un singolo bit $b(t)$ (input seriale)

Il registro shifta a sinistra tutto il suo contenuto per fare posto a $b(t)$ e lo memorizza nel bit più a destra

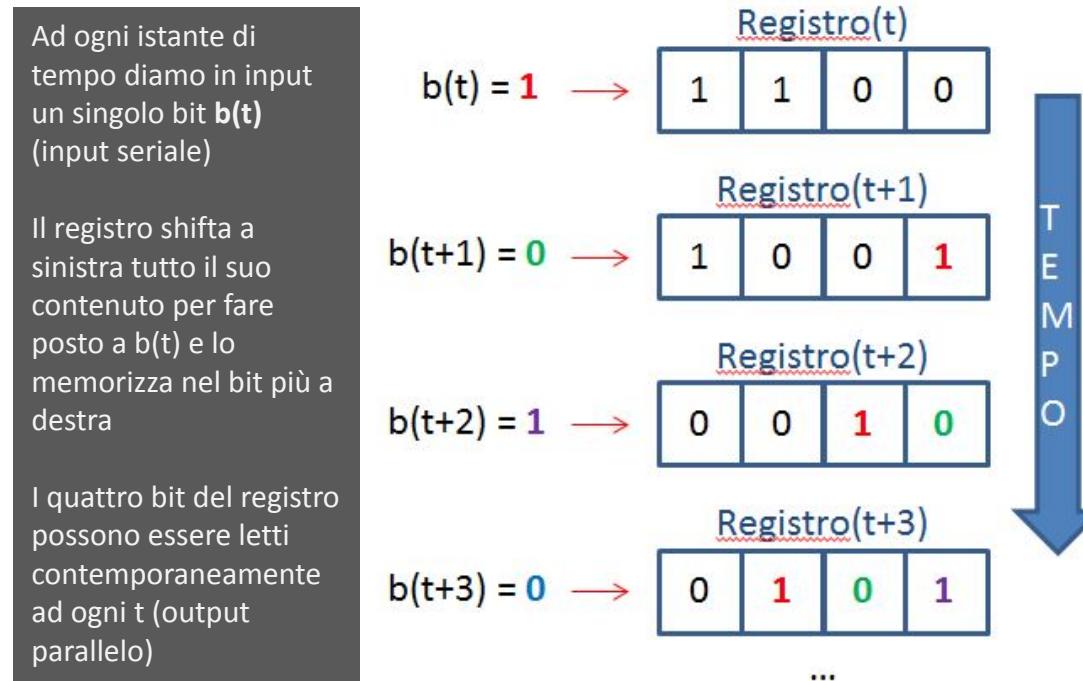
I quattro bit del registro possono essere letti contemporaneamente ad ogni t (output parallelo)



LFSR

Linear Feedback Shift Register

- E' un registro a scorrimento, un componente che già conosciamo



- Come si determina il bit $b(t)$ che entra in input ad ogni ciclo di clock?
La spiegazione sta nella dicitura “Linear Feedback”

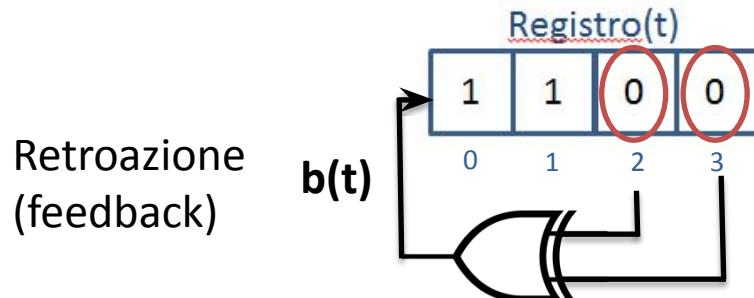
LFSR

Linear Feedback Shift Register

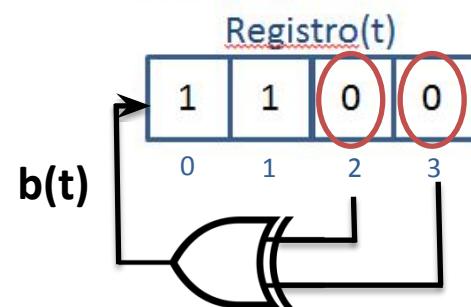
- Linear feedback: il bit $b(t)$ è una funzione lineare di un sottoinsieme di bit (detti taps) del registro al tempo t
- Esempio: registro da 4 bit, scelgo taps: 2,3 (nota, qui metto LSD a destra e MSD a sinistra)



- Calcolo una funzione lineare dei taps, nel nostro caso sarà sempre lo XOR (è anche il caso più comune)

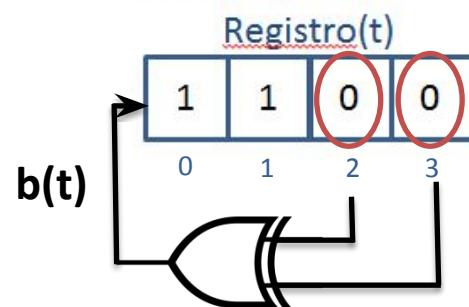


LFSR



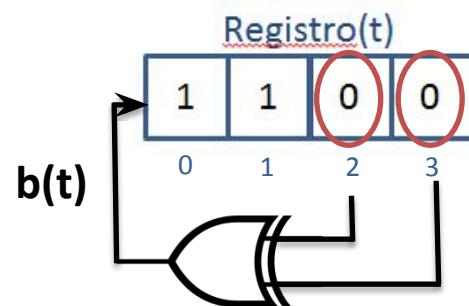
- Lo stato prossimo dipende solo dallo stato corrente
- La sequenza seguita dipende da:
 - I taps scelti
 - La funzione lineare di feedback (nel nostro caso sempre XOR)
 - Il valore iniziale del registro, detto anche **seed**
- La sequenza si ripete dopo un certo numero di stati che costituisce il periodo della sequenza
- **Domanda:** quale è il periodo massimo di una sequenza generata da un LFSR di n bit?

LFSR



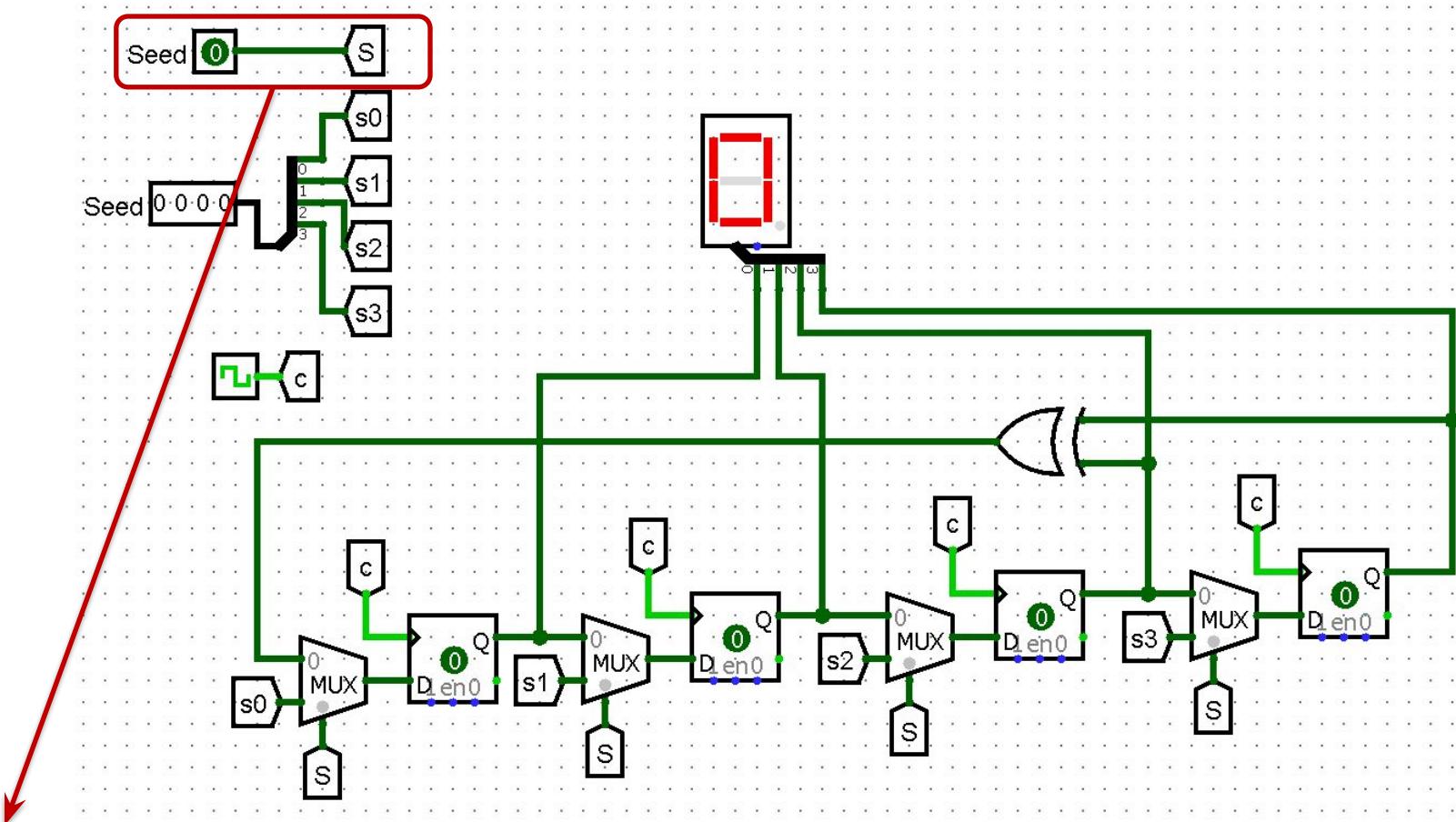
- Lo stato prossimo dipende solo dallo stato corrente
- La sequenza seguita dipende da:
 - I taps scelti
 - La funzione lineare di feedback (nel nostro caso sempre XOR)
 - Il valore iniziale del registro, detto anche **seed**
- La sequenza si ripete dopo un certo numero di stati che costituisce il periodo della sequenza
- **Domanda:** quale è il periodo massimo di una sequenza generata da un LFSR di n bit?
- **Risposta:** $2^n - 1$, tutti i possibili stati meno lo stato di soli zeri. Cosa succede in questo stato?

Esercizio 3



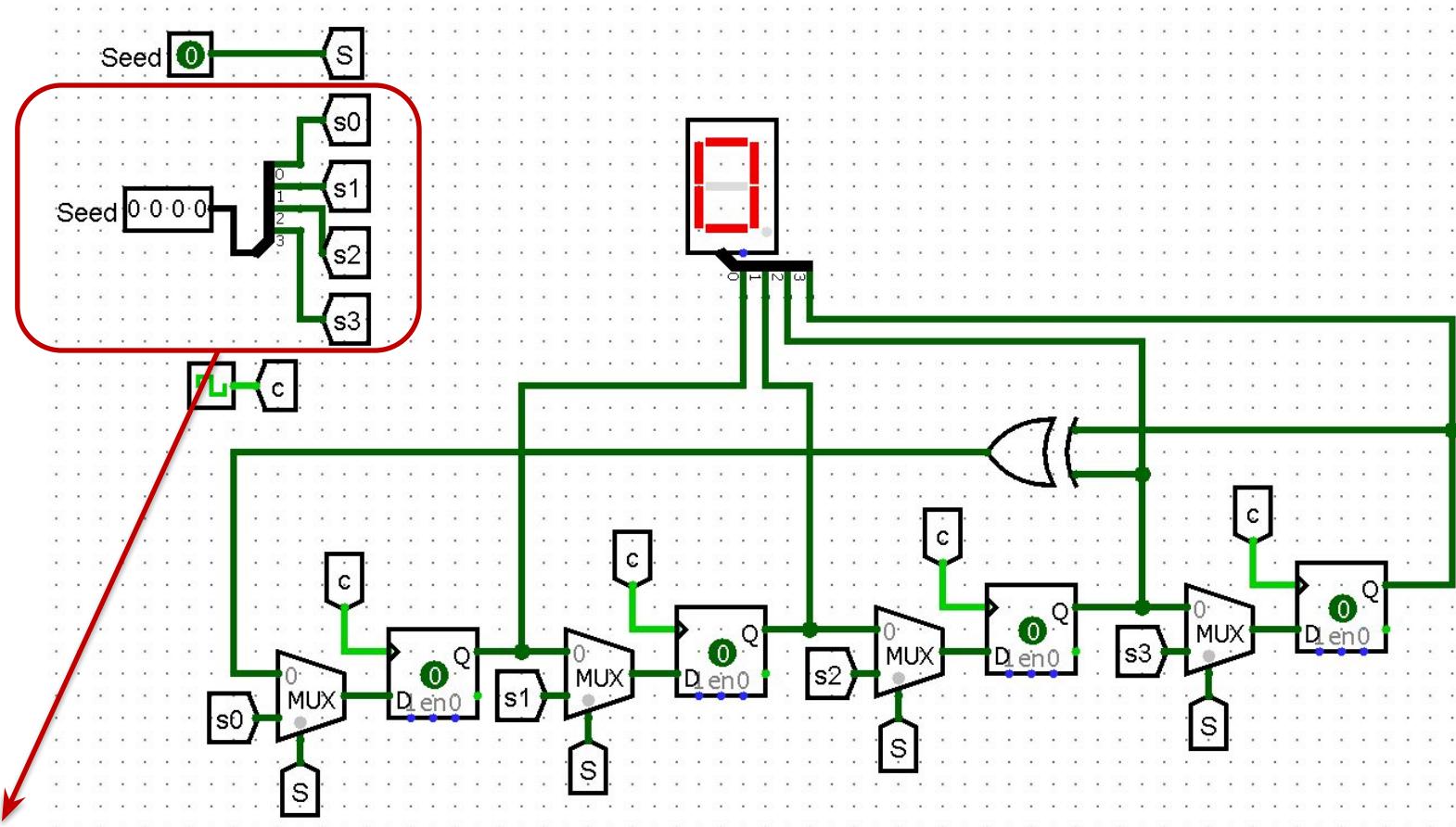
- Esercizio: implementiamo questo generatore e verifichiamo se produce una sequenza di periodo massimo
- Diamo anche la possibilità di operare in modalità scrittura (sincrona) del seed

Esercizio 3



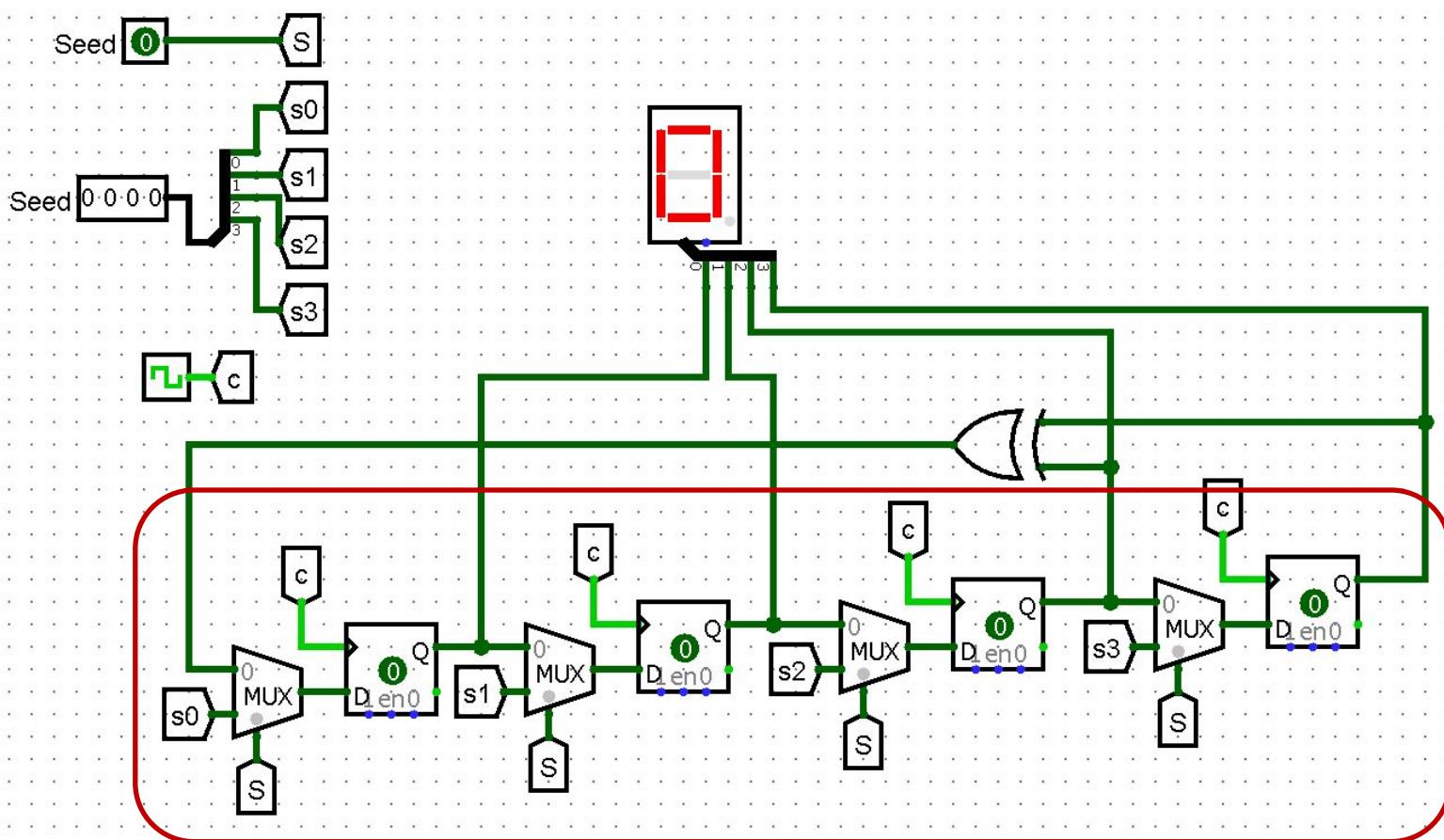
Bit per la modalità scrittura del seed: se posto a 1 scrivo in modo sincrono un seed nel registro

Esercizio 3



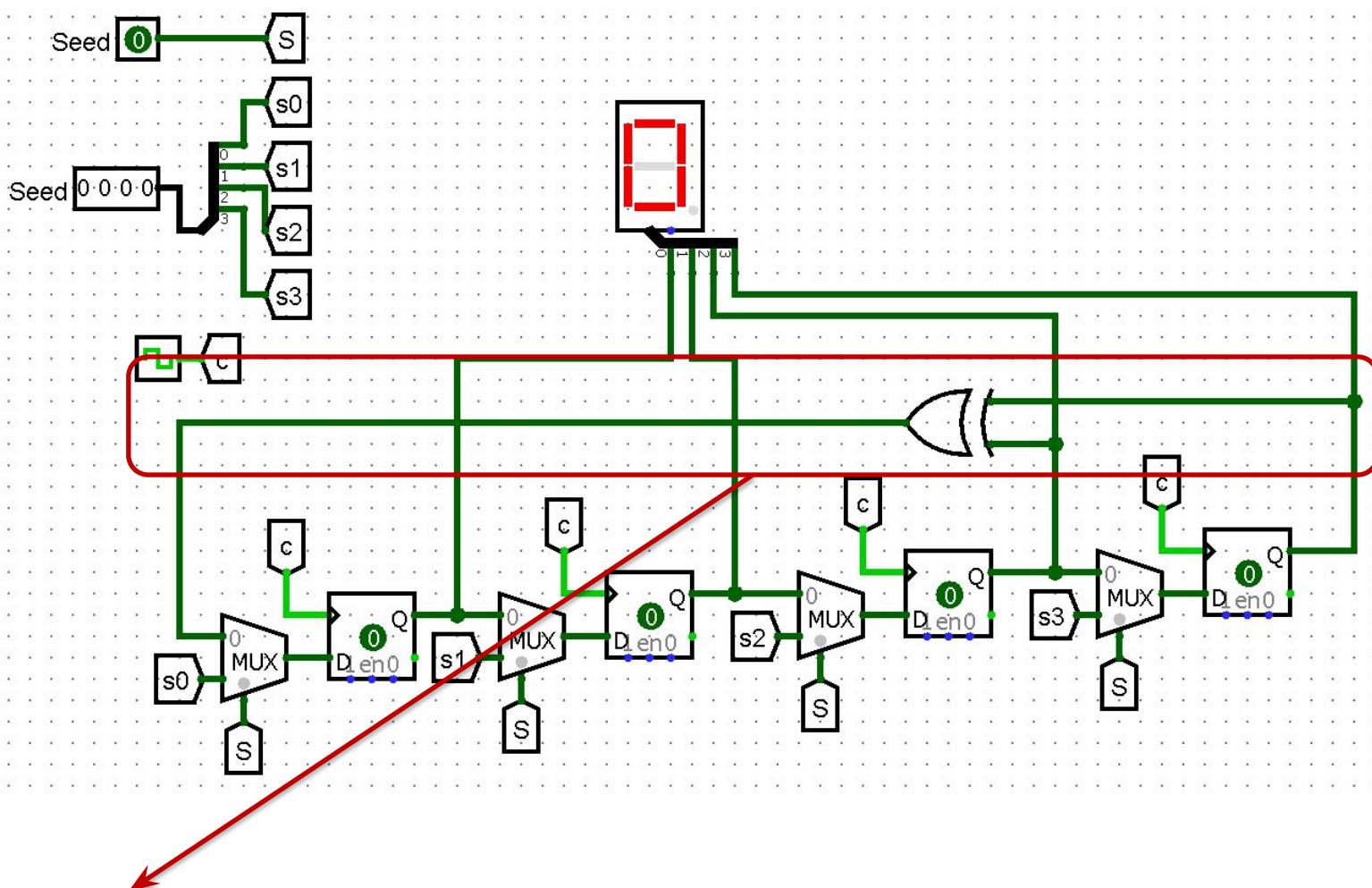
Seed da scrivere

Esercizio 3



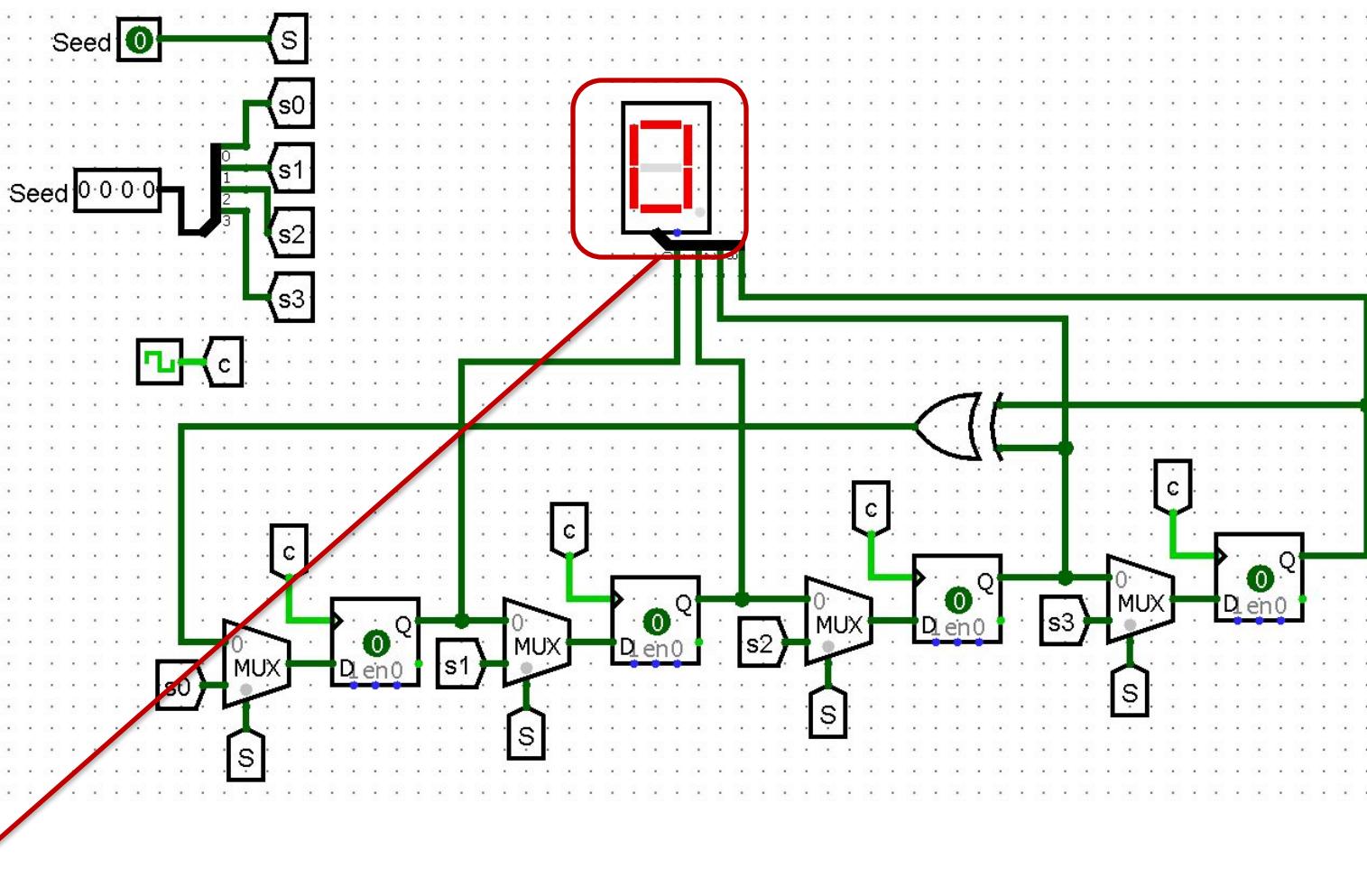
Registro a scorrimento classico; i multiplexer all'ingresso di ogni flip-flop permettono di selezionare tra lo scorrimento o la scrittura del seed (il bit di seed è il selezionatore)

Esercizio 3



Linear feedback: lo XOR dei taps 2 e 3 viene mandato in input al primo flip-flop

Esercizio 3



Sul display visualizzo il valore corrente della sequenza

Esercizio 4: esempio di progetto

- Realizzare un circuito che riproduca il gioco "acchiappa la talpa".
- Ci sono n buche ($n=2$) da cui, in modo imprevedibile, possono comparire talpe che restano visibili per un certo lasso di tempo.
- Scopo: colpire la talpa mentre è visibile. Per ogni talpa colpita si guadagna un punto.
- Qualche suggerimento su come impostare il circuito:
 - Buca = led, la presenza/assenza della talpa è modellata con lo stato del led (acceso/spento)
 - Colpire la talpa = pressione di un bottone
 - Stato del board di gioco = memoria da 1 bit per ogni buca. 1 se la talpa è visibile, 0 altrimenti
 - Se premo il bottone della buca quando la talpa è presente, il punteggio aumenta di 1 (usare un contatore per il punteggio)
 - Le talpe compaiono in maniera random (usare un random number generator)
 - Ogni talpa, quando esce dalla buca, rimane visibile per un tempo fisso, per es. otto cicli di clock

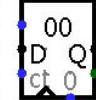
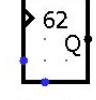
Esercizio 4



- Esempio di progetto base: realizziamo un circuito che riproduca il gioco «acchiappa la talpa»
- Come funziona? Ci sono n buche da cui, in modo imprevedibile, possono comparire talpe che restano visibili per un certo lasso di tempo. Lo scopo del gioco è colpire una talpa nel lasso di tempo in cui è visibile. Per ogni talpa colpita si ottiene un punto.
- Iniziamo interpretare questa specifica nei termini di un circuito logico e, soprattutto, considerando i componenti Logisim che conosciamo.

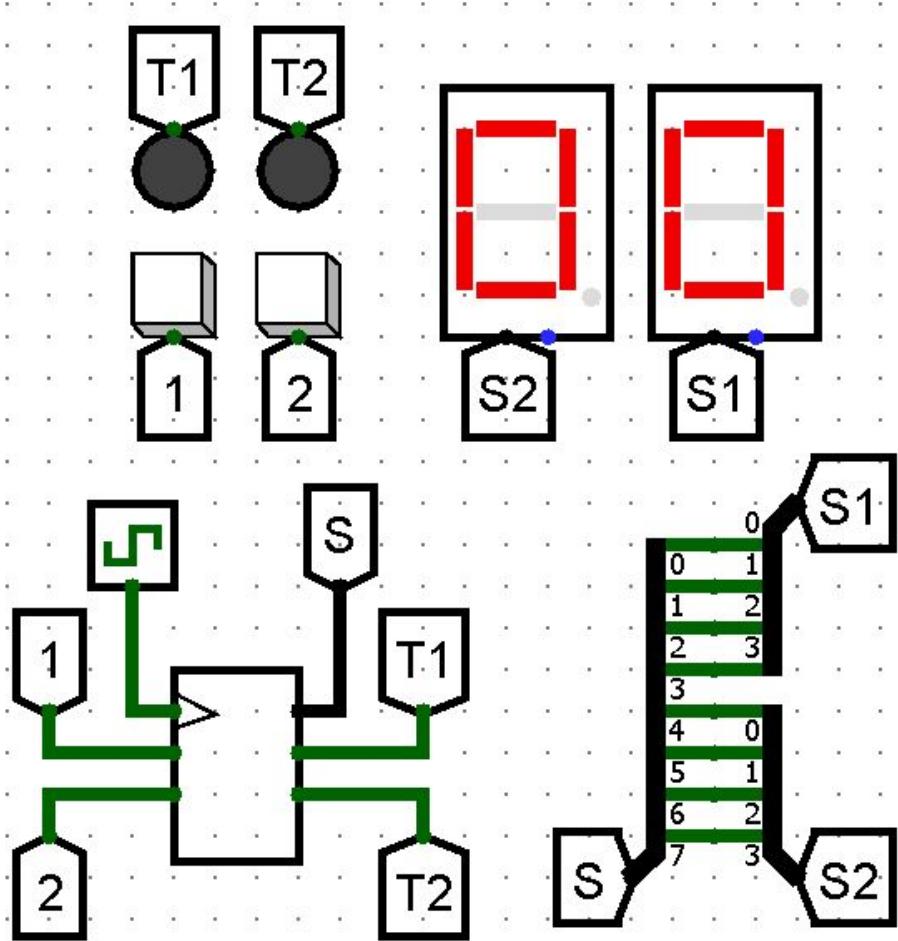
Esercizio 4



- Buca = led, la presenza/assenza della talpa è modellata con lo stato del led (acceso/spento) 
- Colpire la talpa = pressione di un bottone (input del circuito, uno per ogni buca) 
- Stato del board di gioco = memoria da 1 bit (flip flop) per ogni buca, memorizzo un 1 se la talpa è visibile 0 altrimenti 
- Se premo il bottone di buca quando la talpa è presente guadago un punto: stato del board e input determinano il punteggio (servirà contatore per punteggio) 
- Le talpe compaiono in modo random, servirà un *random number generator* 
- Per semplicità assumiamo che ogni talpa una volta comparsa, resti visibile per un tempo fissato (esempio: 8 cicli di clock)

Esercizio 4

- Il circuito di «acchiappa la talpa», visuale esterna



Nomenclatura:

T1: talpa 1

T2: talpa 2

1: colpire buca 1

2: colpire buca 2

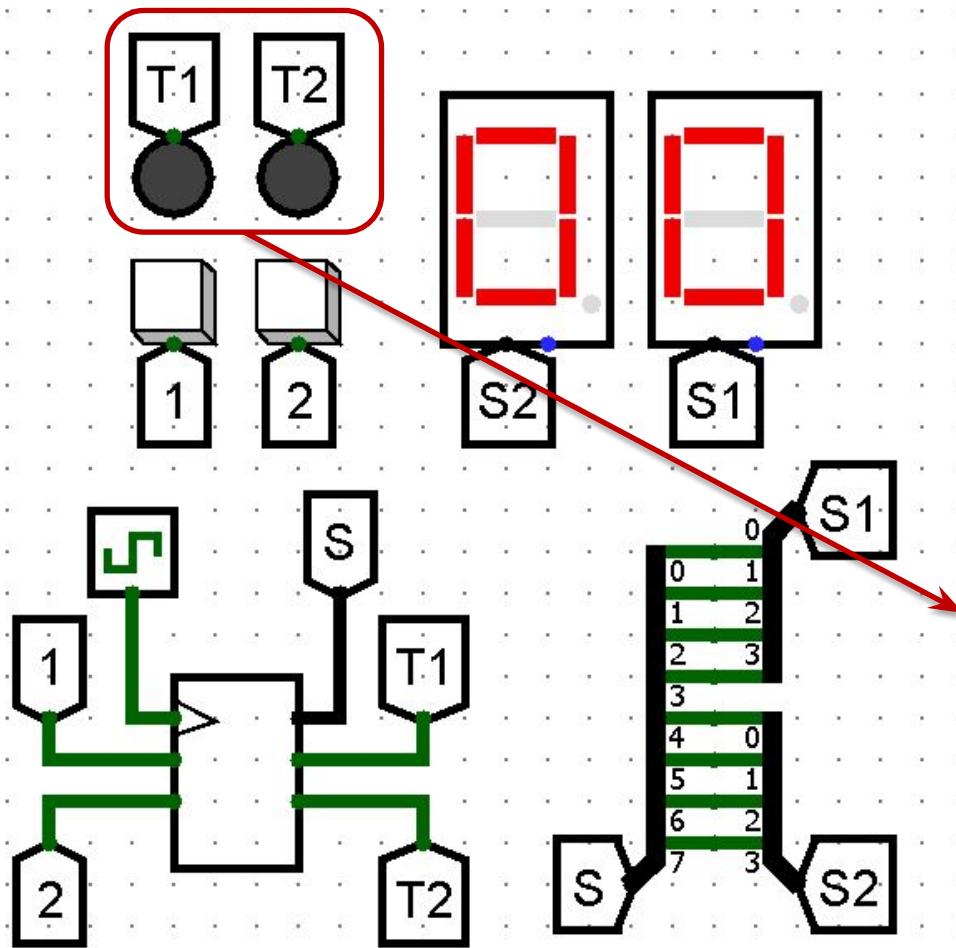
S: score su 8 bit

S1: 4 LSD dello score

S2: 4 MSD dello score

Esercizio 4

- Il circuito di «acchiappa la talpa», visuale esterna



Nomenclatura:

T1: talpa 1

T2: talpa 2

1: colpire buca 1

2: colpire buca 2

S: score su 8 bit

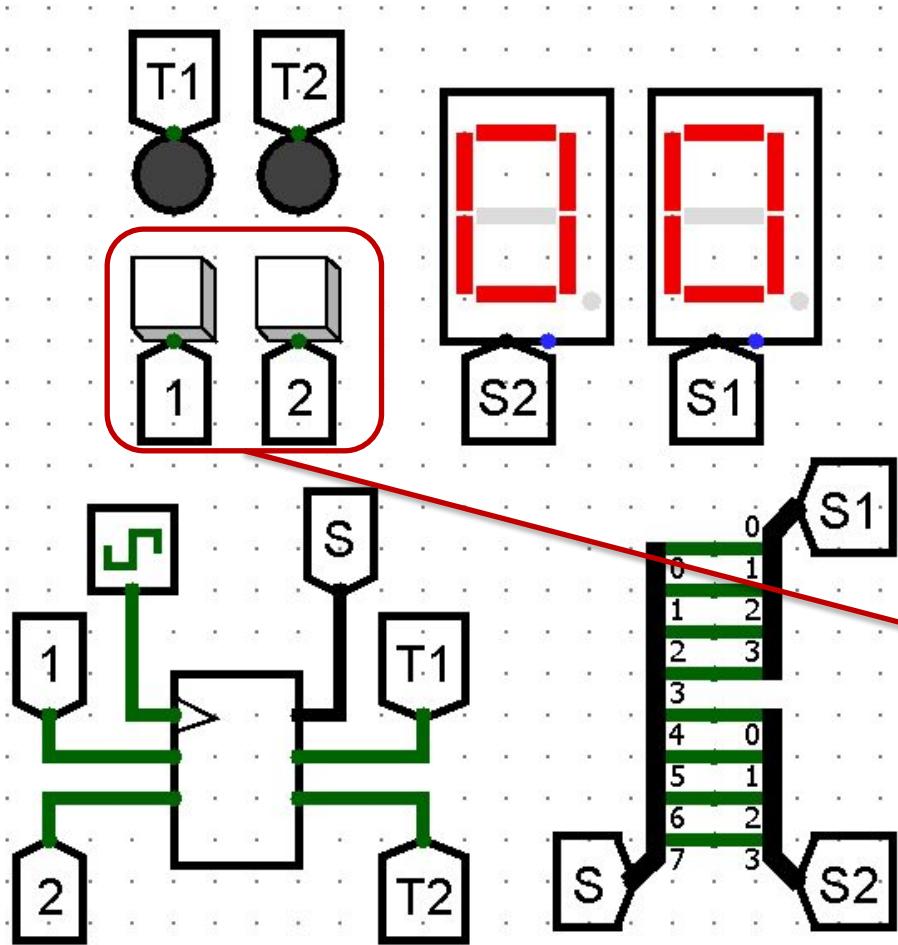
S1: 4 LSD dello score

S2: 4 MSD dello score

I due led rappresentano le due buche: led acceso significa talpa visibile, led spento significa talpa non visibile

Esercizio 4

- Il circuito di «acchiappa la talpa», visuale esterna



Nomenclatura:

T1: talpa 1

T2: talpa 2

1: colpire buca 1

2: colpire buca 2

S: score su 8 bit

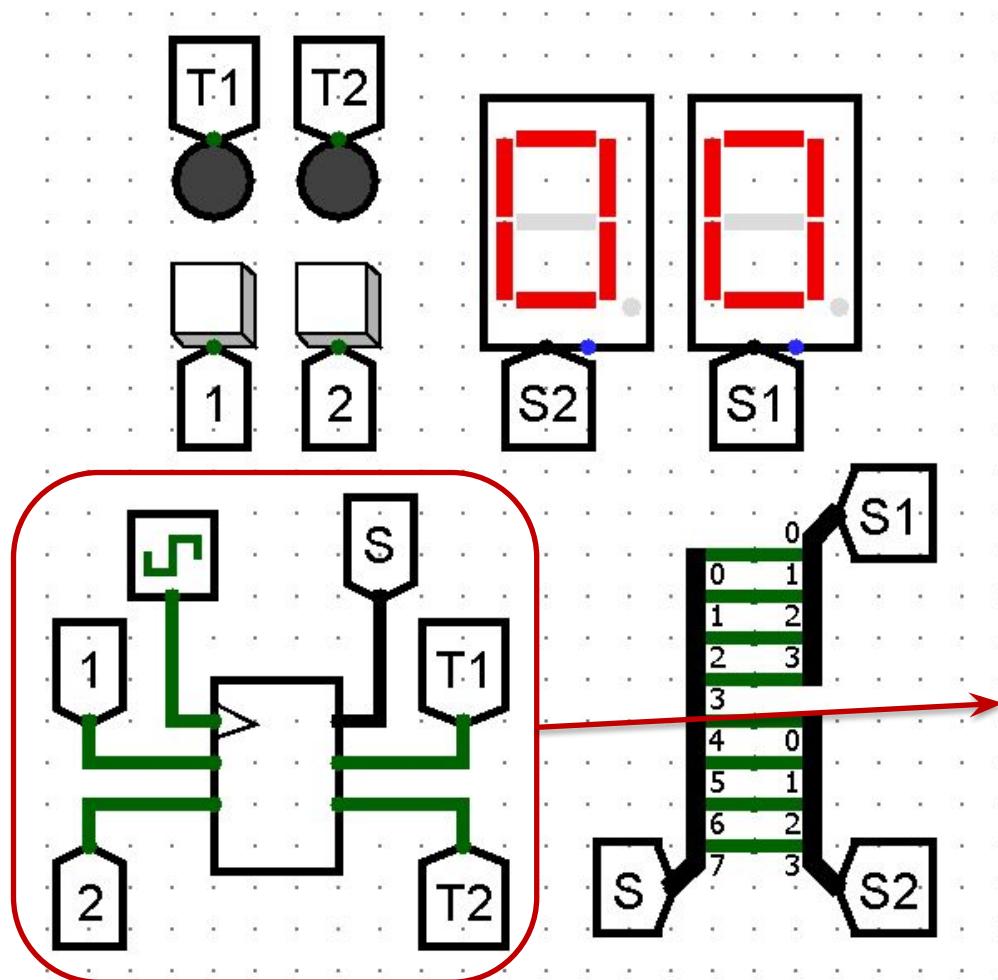
S1: 4 LSD dello score

S2: 4 MSD dello score

I due bottoni verranno utilizzati per colpire le talpe: premere il bottone 1 significa colpire nella buca 1, premere il bottone 2 significa colpire nella buca 2

Esercizio 4

- Il circuito di «acchiappa la talpa», visuale esterna



Nomenclatura:

T1: talpa 1

T2: talpa 2

1: colpire buca 1

2: colpire buca 2

S: score su 8 bit

S1: 4 LSD dello score

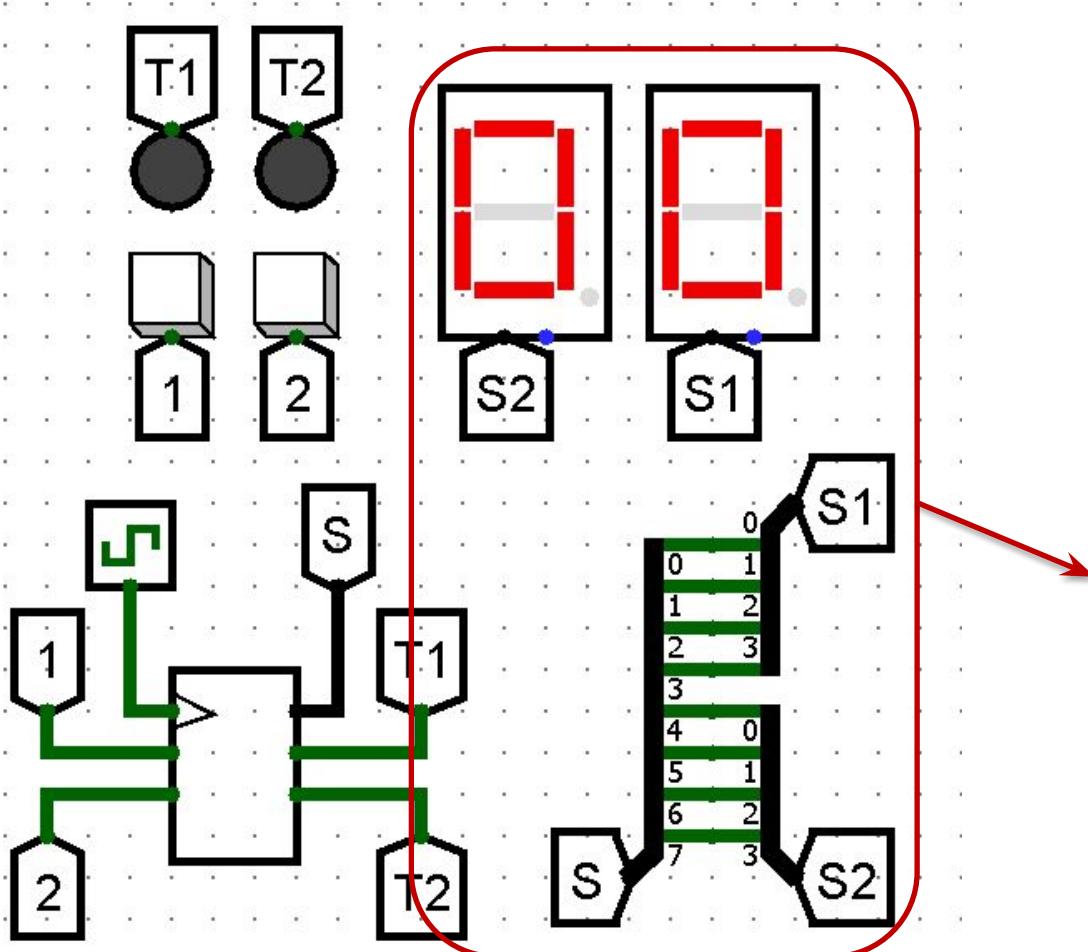
S2: 4 MSD dello score

Il gioco è regolato da questa macchina dove:

- input: bottone 1 e 2
- stato: visibilità/non visibilità della talpa in buca 1 e 2, score corrente
- uscita: stato

Esercizio 4

- Il circuito di «acchiappa la talpa», visuale esterna



Nomenclatura:

T1: talpa 1

T2: talpa 2

1: colpire buca 1

2: colpire buca 2

S: score su 8 bit

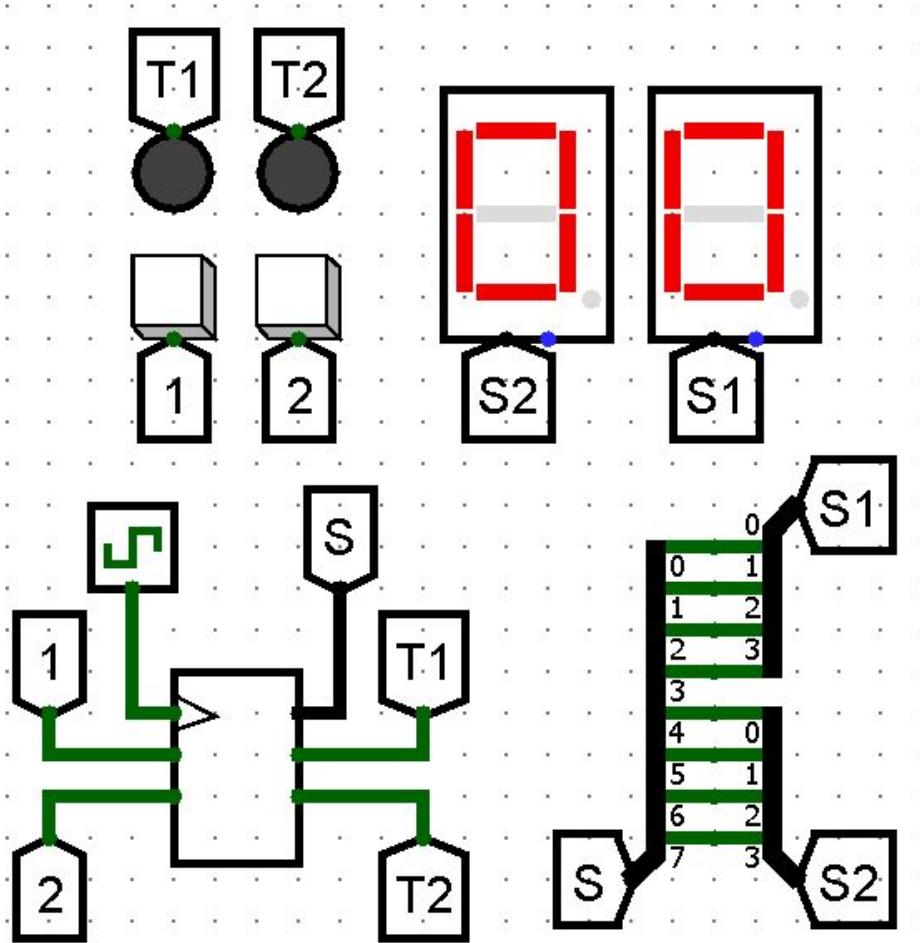
S1: 4 LSD dello score

S2: 4 MSD dello score

Lo score S è su 16 bit, li splitto in due gruppi da 4 (S1, S2) per poterli visualizzare sui due Hex Digit Display

Esercizio 4

- Il circuito di «acchiappa la talpa», visuale esterna



Nomenclatura:

T1: talpa 1

T2: talpa 2

1: colpire buca 1

2: colpire buca 2

S: score su 8 bit

S1: 4 LSD dello score

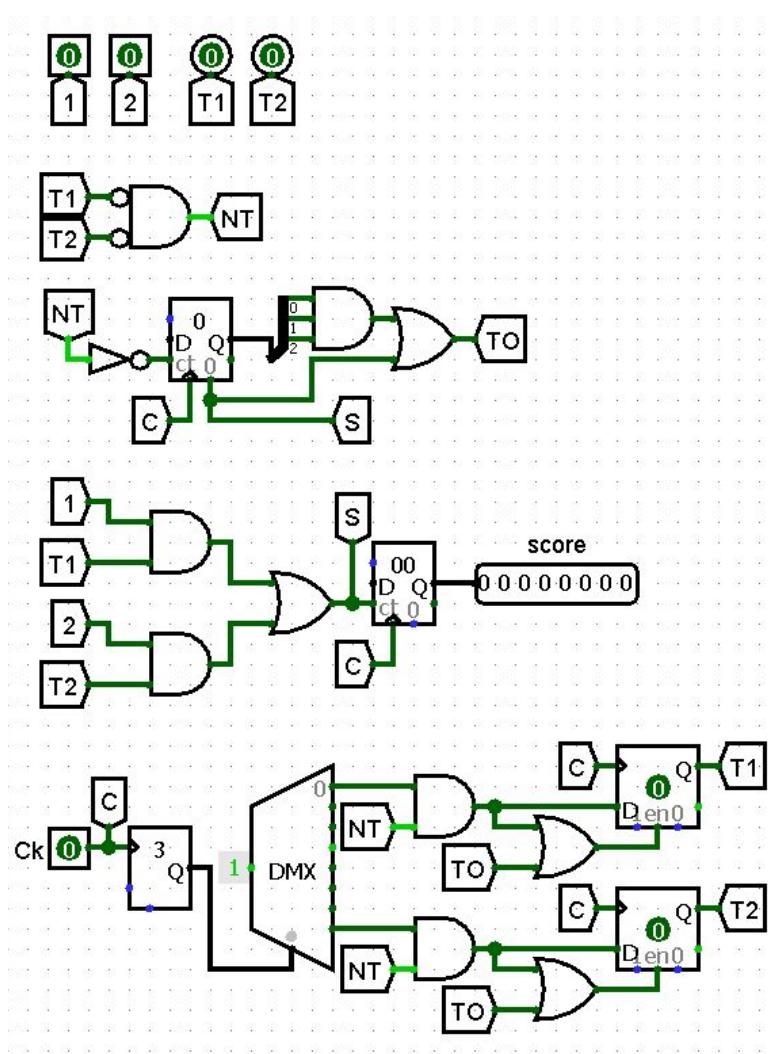
S2: 4 MSD dello score

Demo ...

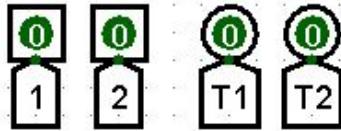
Esercizio 4

Soluzione

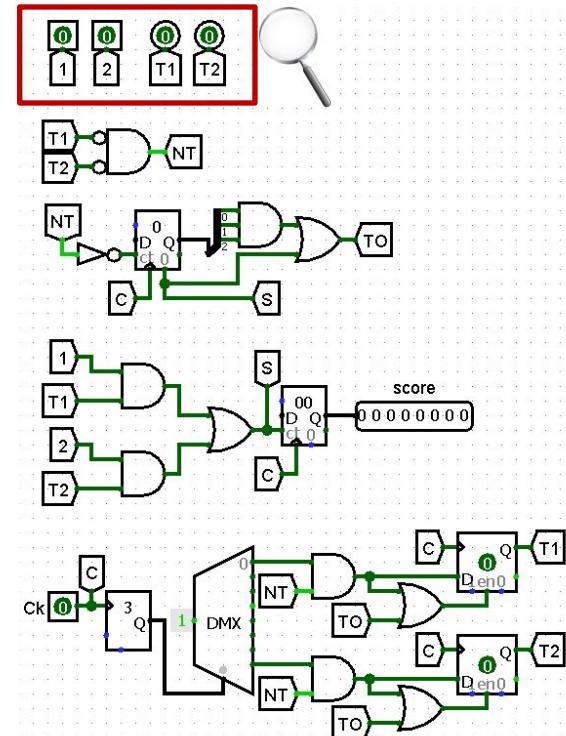
Analizziamola un
sottocircuito alla
volta



Esercizio 4

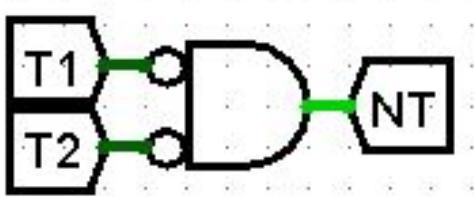


- **1 e 2** sono i pin di input per i due bottoni
- **T1 e T2** sono i pin di output a cui collegheremo i led

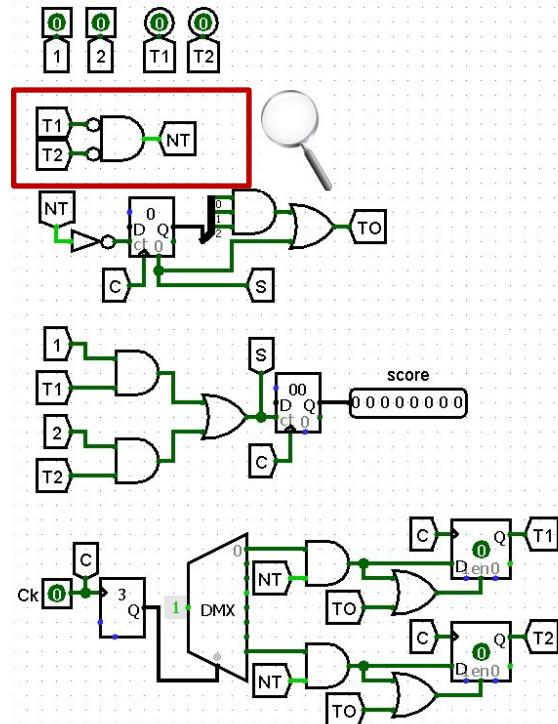


T1: talpa 1
T2: talpa 2
1: colpire buca 1
2: colpire buca 2
S: +1 score su 8 bit

Esercizio 4

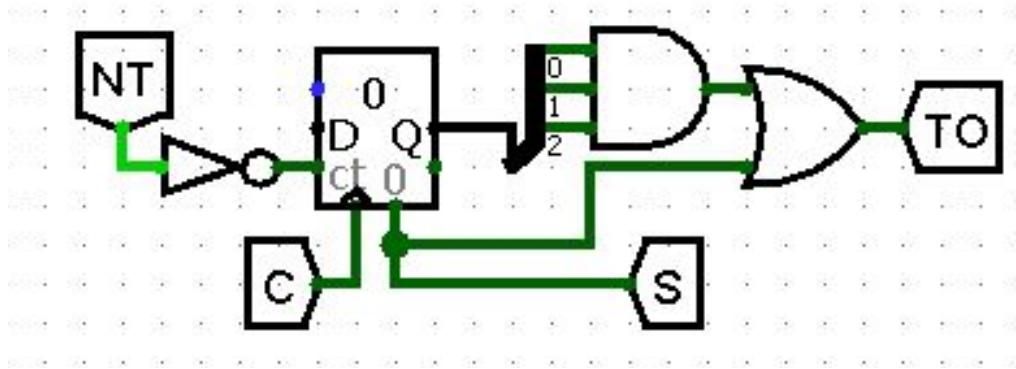


- Calcolo del Segnale **NT** = Nessuna Talpa visible



T1: talpa 1
T2: talpa 2
1: colpire buca 1
2: colpire buca 2
NT: no talpe

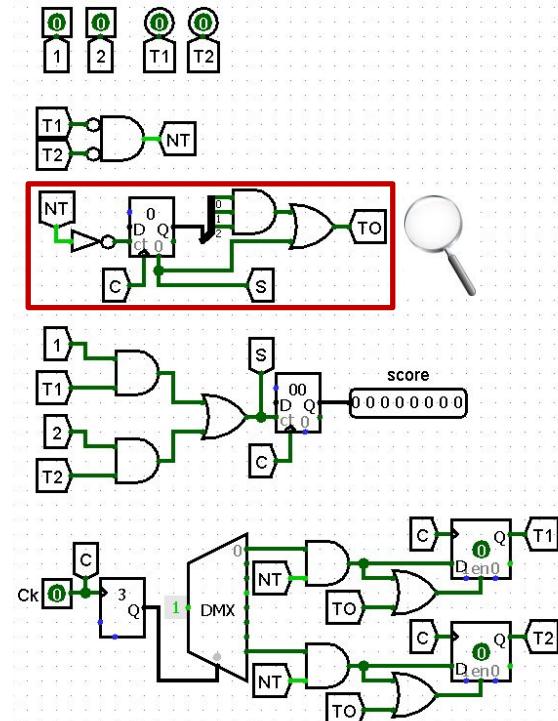
Esercizio 4



Gestione del tempo di visibilità della talpa

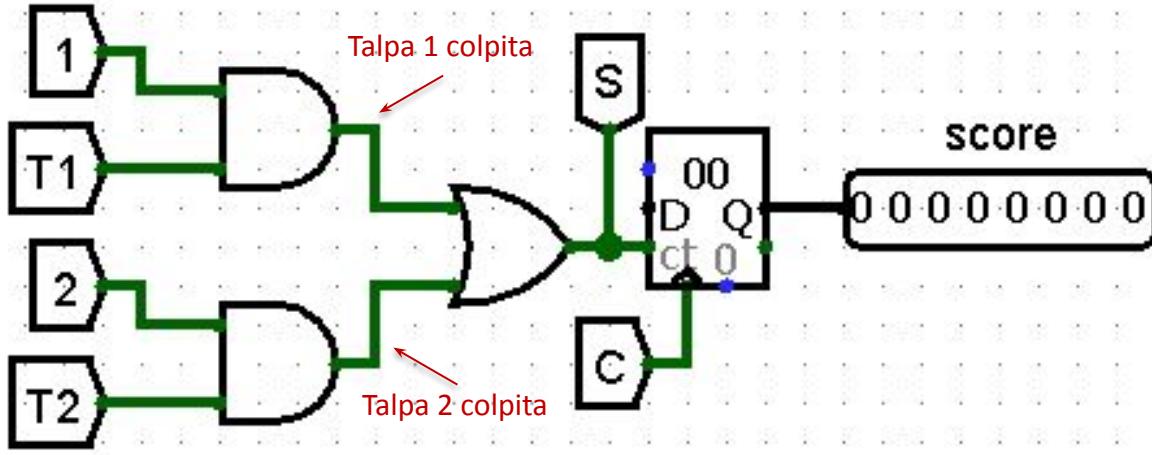
Counter:

- conta il tempo di visibilità su 3 bit
- per ogni ciclo di clock in cui c'è almeno una talpa (**NT = 0**) viene incrementato di 1 (**nel circuito non ci potrà essere più di una talpa visibile, almeno una talpa = esattamente una talpa**)
- se guadagno un punto (colpisco la talpa, **S = 1**), resetto il counter
- se il counter raggiunge il valore massimo (**Q = 111**) **OR** colpisco la talpa (**S=1**) genero segnale **TO** (time out)



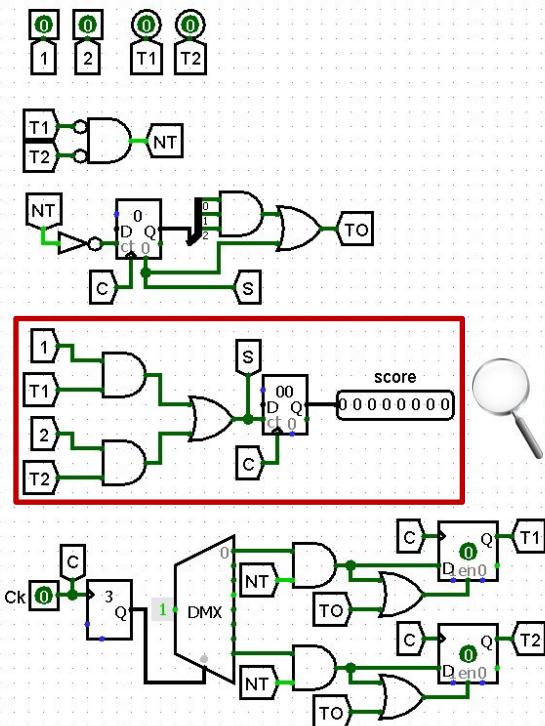
T1: talpa 1
T2: talpa 2
1: colpire buca 1
2: colpire buca 2
NT: no talpe
C: clock
TO: time out
S: +1 score su 8 bit

Esercizio 4



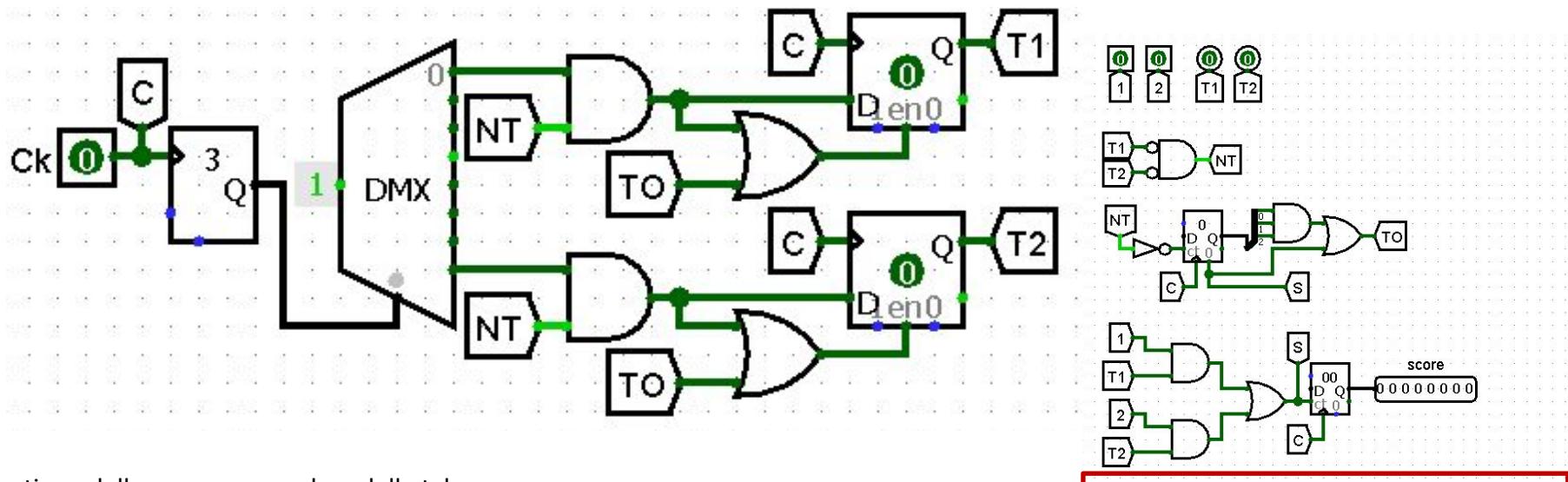
Gestione del punteggio:

- colpisco la talpa 1: **1 = 1 AND T1 = 1**
- colpisco la talpa 2: **2 = 1 AND T2 = 1**
- se (*colpisco la talpa*) 1 **OR** (*colpisco la talpa 2*) allora **S=1** e incremento di 1 il contatore dello score



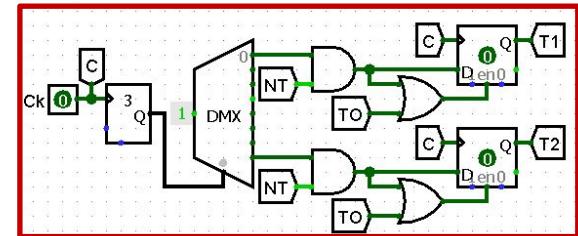
T1: talpa 1
T2: talpa 2
1: colpire buca 1
2: colpire buca 2
NT: no talpe
C: clock
S: +1 score su 8 bit

Esercizio 4



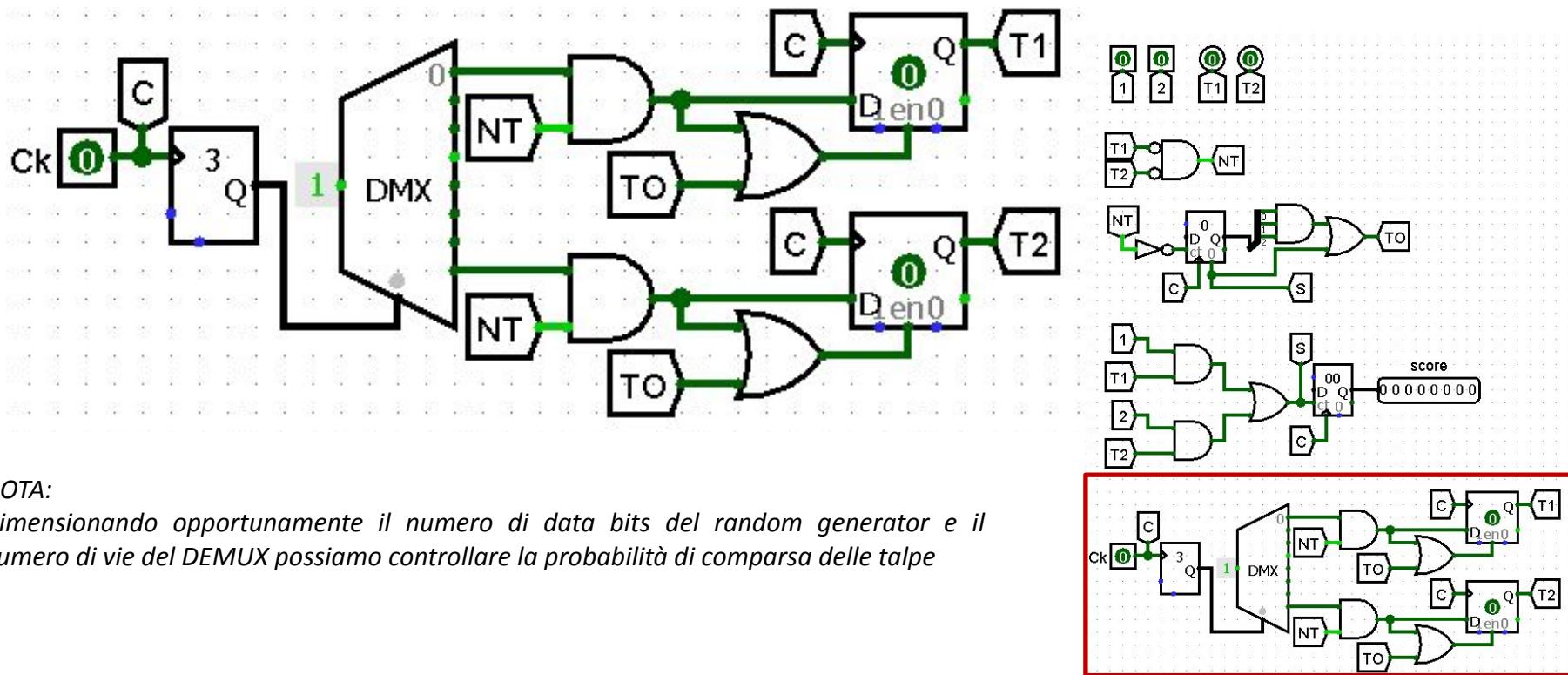
Gestione della comparsa random delle talpe:

- A ogni ciclo di clock genero un numero random tra 0 e 7, lo collego a un DEMUX ponendo a 1 una di otto vie in modo random; solo due vie possono essere accese = solo due talpe
- Se la linea del DEMUX è accesa **AND** non ci sono talpe, faccio *enable* del registro della buca corrispondente e scrivo 1
- Se c'è stato un time out allora per ciascun registro (buca):
 - se la linea del DEMUX è spenta, scrivo 0 nel registro (la talpa scompare)
 - se la linea del DEMUX è accesa, scrivo 1 nel registro (la talpa ricompare subito)



T1: talpa 1
T2: talpa 2
1: colpire buca 1
2: colpire buca 2
NT: no talpe
TO: time out
C: clock
S: +1 score su 8 bit

Esercizio 4



NOTA:

Dimensionando opportunamente il numero di data bits del random generator e il numero di vie del DEMUX possiamo controllare la probabilità di comparsa delle talpe

T1: talpa 1
T2: talpa 2
1: colpire buca 1
2: colpire buca 2
NT: no talpe
TO: time out
C: clock
S: +1 score su 8 bit

