17 giugno 2022

The Game of Life

1 Prefazione

The Game of Life è un automa cellulare sviluppato dal matematico inglese John Conway sul finire degli anni sessanta: consiste nella costruzione di universi artificiali, popolati da "organismi" che possono simulare la vita reale.
Alla base di tutto ci sono gli automi cellulari, costituiti da spazi omogenei divisi in celle elementari, ognuna delle quali, ad intervalli regolari, cambia stato, secondo regole di evoluzione che riguardano la cella stessa e quelle ad essa confinanti.

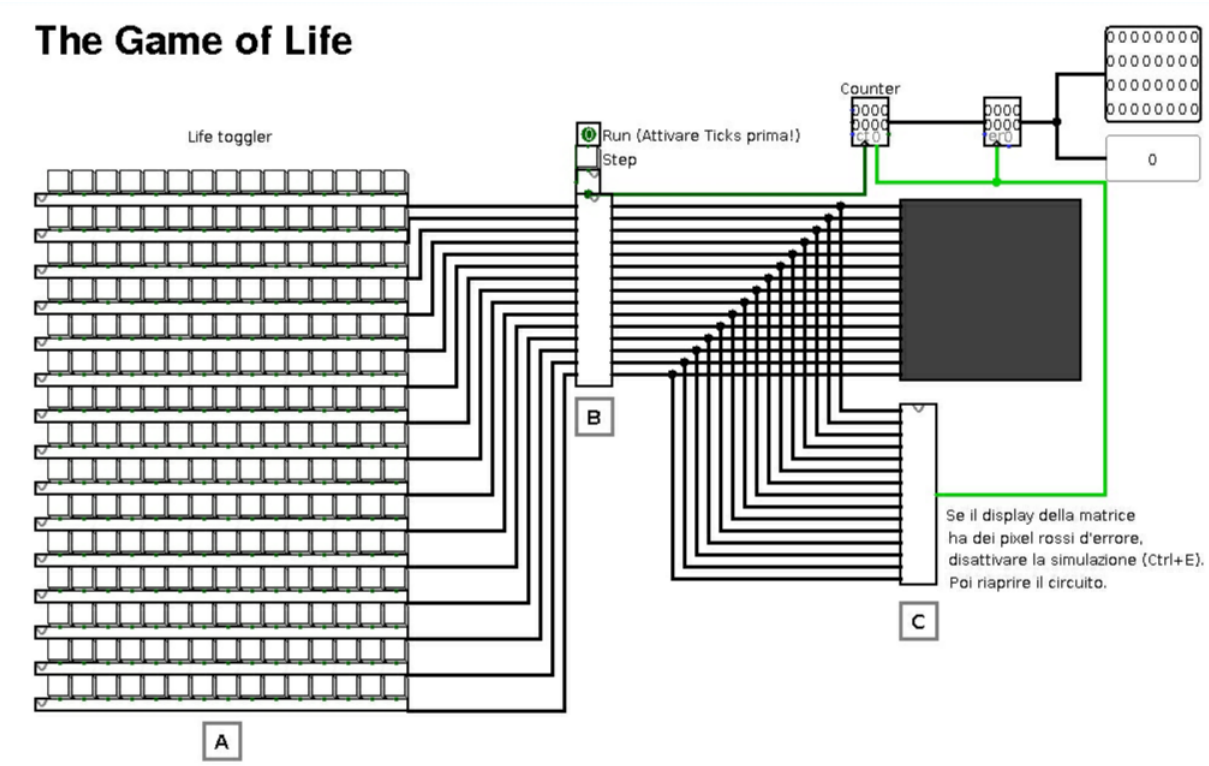
1.1 Obiettivo

Realizzazione di un circuito in Logisim che possa leggere da input la nascita di celle, in modo da comporre la configurazione iniziale della popolazione e creando uno spazio omogeneo diviso in celle elementari, e possa ricreare in output la vita di questi organismi, caratterizzati da regole di comportamento sempre più complesse.

2 Descrizione generale del progetto

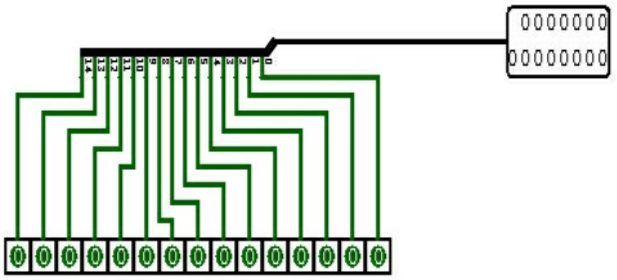
Il progetto propone un circuito costituito da 15 file, ciascuna costituita da 15 bottoni, che rappresentano i possibili input, da parte dell'utente, delle celle da raffigurare. Ogni fila di input è collegata a uno storage, che analizza lo stato della cella corrente(scelta dall'utente) e gli stati delle sue 8 celle confinanti.
Allo storage è collegato un sottocircuito di clock in modo che ogni cella venga aggiornata simultaneamente secondo le regole di evoluzione.
L'output dello storage è collegato a una matrice led quadrata 15x15 che riproduce inizialmente la nascita delle celle scelte dall'utente, e poi, dopo che il clock parte, l'evolversi della popolazione di celle.
Infine è presente anche un counter che conta il numero di evoluzioni degli organismi; esso si resetta(se non hanno una crescita infinita) quando non esisterà più una cella viva sulla matrice(cioè se la configurazione non ha forme oscillanti all'infinito), mandando in output, in binario e in decimale, il numero precedentemente ottenuto dal counter.

Di seguito è riportato il circuito **Main** del progetto:



3 Componenti

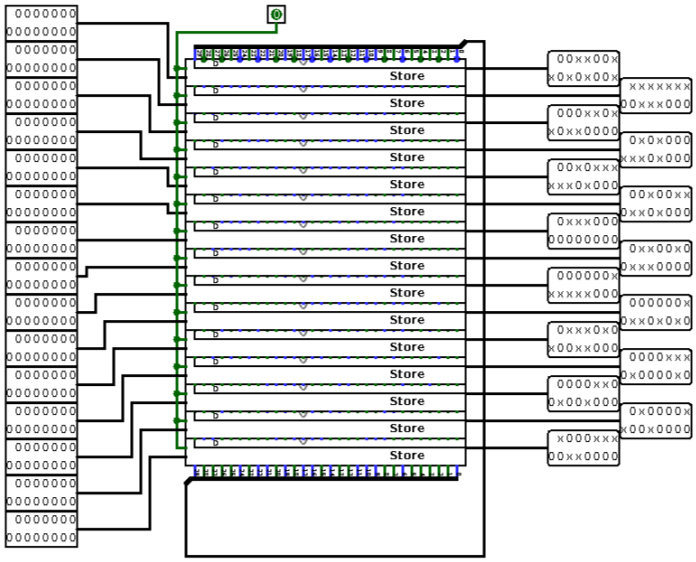
3.1 A: InputRow



Il circuito rappresenta una fila di input collegata ai 15 bottoni. E' composto da 15 pin di input, che nel main sono i 15 bottoni per attivare le celle, collegati a uno splitter per generare un pin di output da 15 data bits. Se l'utente preme, ad esempio, il terzo bottone, si attiverà il terzo pin di input, generando sulla matrice led la cella in posizione 3 in termini di x, sulla fila di **InputRow** scelta.

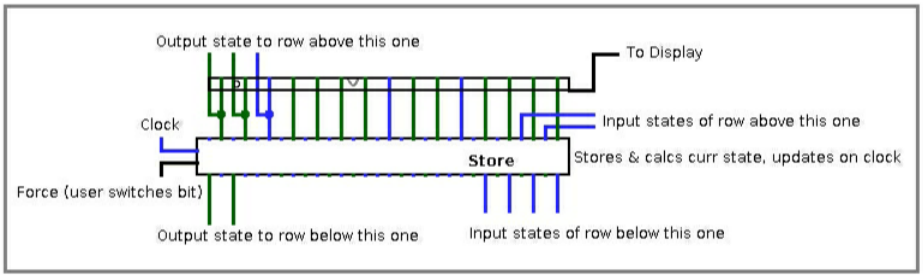
3.2 B: Storage

Lo storage è la parte principale di tutto l'automa. Si può vedere l'implementazione nella figura sottostante:



Come si vede dall'immagine lo **Storage** è formato dall'unione di due sottocircuiti principali: **StorageRow** e **Display**. Ci sono infatti 15 StorageRow, disposti a fila, ciascuno collegato a un display. Ogni StorageRow ha in ingresso un segnale di clock e un pin di input da 15 data bits, che rappresentano i bottoni di input; restituisce invece un pin di output da 15 data bits.

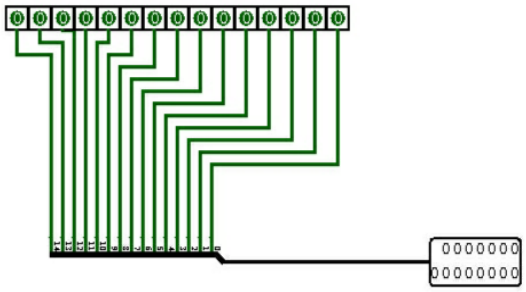
Uno schema di implementazione si può osservare qui:



Questa figura rappresenta una singola raffigurazione dell'unione tra StorageRow e Display. Abbiamo:

- un pin di clock per aggiornare l'evoluzione della popolazione.
- un "Force" che sarebbe l'InputRow per generare la cella corrente.
- in basso: 15 input dalla row sottostante per aggiornare la StorageRow stessa e 15 output alla row sottostante per aggiornarla.
- in alto: 15 input dalla row soprastante per aggiornare la StorageRow stessa e 15 output alla row soprastante per aggiornarla, che sono a loro volta collegati al Display per rappresentare sulla matrice l'evolversi dell'automa cellulare.

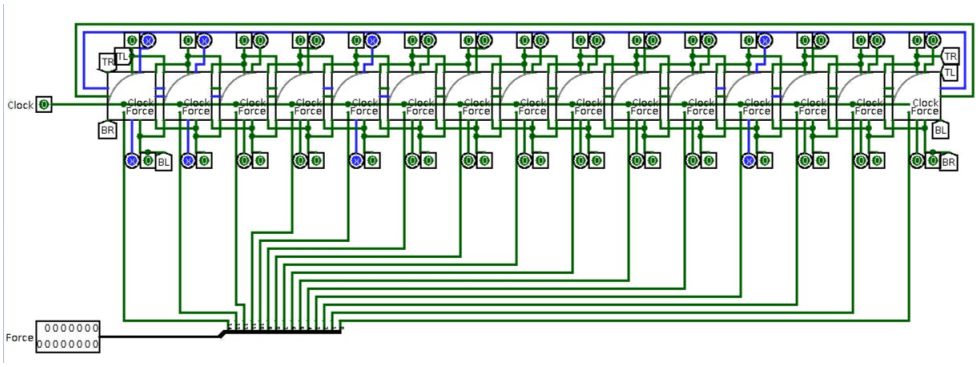
3.2.1 Display



Il Display ha un'implementazione uguale a quella dell'InputRow: 15 pin di input collegati a uno splitter, che dà origine a un output composto da 15 data bits. Serve a fare da tramite tra gli stati delle celle e la loro raffigurazione sulla matrice.

3.2.2 StorageRow

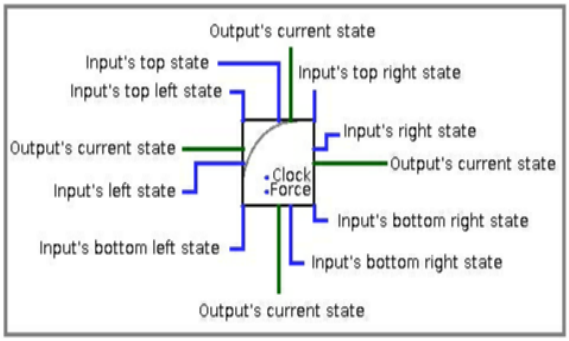
Questo circuito è il fulcro centrale di tutto l'evolversi del nostro "universo artificiale".



E' formato da 15 sottocircuiti **New**, ciascuno collegato da un clock, da un input di Force da 1 bit, da 8 input, che rappresentano gli stati delle celle confinanti, e 4 output, che rappresentano gli stati della cella corrente, opportunamente modificati dagli stati delle 8 celle adiacenti. Si può notare che nella prima e ultima New ci sono dei tunnel in quanto considero il mondo come un pianeta perfettamente sferico, quindi i confini sono tra loro connessi.
Uno schema di implementazione di una singola parte del circuito è visibile nella figura sotto:

3.2.3 New

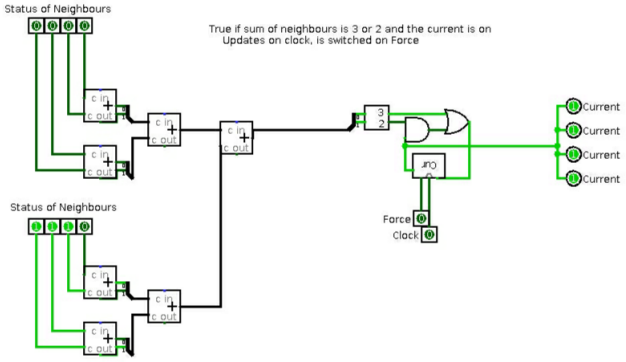
Il seguente circuito rappresenta il controllo delle regole di evoluzione delle celle dell'universo. Le regole di evoluzione del gioco di Conway son le seguenti:



Come si può vedere dalla foto, lo stato della New, che rappresenta la cella di "nascita", dipende dallo stato delle sue celle confinanti, dove per confinanti si intende che due celle sono connesse in una delle 8 direzioni possibili (anche in diagonale). Le celle possono essere in due possibili stati: vive o morte(1 o 0); in base a ciò si deciderà se la nuova cella nascerà o meno, a seconda delle regole del Game of Life.

- 1. Una cella viva rimane in vita se esistono 2 o 3 celle vive confinanti (sopravvivenza)
- 2. Una cella viva muore se confina con meno di due celle vive (isolamento) o con piu' di 3 celle vive(sovraffollamento)
- 3. Una cella morta con esattamente 3 celle vive confinanti nasce e diventa viva (riproduzione)

Il circuito si presenta cosi:



Abbiamo 8 input degli stati delle celle confinanti, collegati a degli adder per controllare, attraverso un sottocircuito **2or3**, se la somma dei "vicini" è 2 o 3, un sottocircuito **Current** che controlla se generare una nuova cella, e infine 4 output che rappresentano lo stato della cella corrente che potrà cambiare a seconda degli stati delle celle confinanti.
Analizziamo i tre casi delle regole di evoluzione:

1. Nel primo caso, se attivassimo due o tre stati delle celle confinanti di input, avremmo in uscita dal sottocircuito 2or3 il valore 2 o 3.
Nel caso avessimo 2 come valore, entrerebbe nell'and assieme ai bit del Current, generando un output positivo, in quanto la cella corrente è viva. Entrando nell'or assieme al valore 3(che in questo caso è zero), l'output sarà positivo e quindi la cella corrente rimarrà in vita, senza creare nuove celle, grazie al sottocircuito **Current**.
Nel caso invece avessimo 3 come valore, entrerebbe nell'or assieme all'output dell'and del valore di 2(che in questo caso è zero) e di Current(che ha valore positivo), e quindi l'uscita dell'or sarà

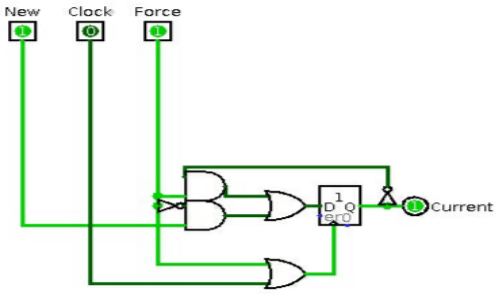
sempre positiva. Come prima, rimane in vita la cella corrente.

2. Nel secondo caso, se attivassimo meno di due stati delle celle confinanti di input o più di tre, e avessimo la cella corrente viva, avremmo zero in entrambi gli output del sottocircuito 2or3. Quindi l'and col valore due assieme ai valori di current avrà come output zero, cosi come l'or col valore 3. Di conseguenza non entrerà nessun input nel sottocircuito di Current e quindi la cella corrente muore.

3. Nel terzo e ultimo caso, con una cella morta e 3 celle confinanti vive avremmo in uscita dal sottocircuito 2or3 il valore 3 e i bit di output di Current sarebbero tutti a zero. In questo caso l'and col valore 2(che è a zero) e coi bit di Current darà in output zero, che messo in input nell'or col valore 3, darà un output positivo. Di conseguenza entrerà un input nel sottocircuito **Current** e avremo un output positivo che genererà una nuova cella.

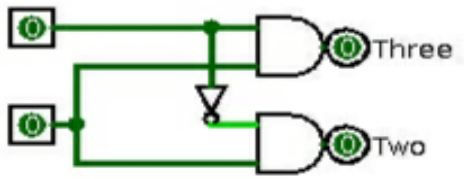
3.2.4 Current

Il circuito di Current serve a generare una nuova cella nel caso abbia 3 celle vive confinanti o nel caso venga premuto il bottone di input da parte dell'utente.



Il circuito **Current** è composto da tre input, un clock per aggiornare la crescita della popolazione, un input "New" che rappresenta l'output delle somme dei vicini nel circuito **New**(se la somma è 2 o 3 allora l'input "New" sarà a 1) e un input "Force" che raffigura il bottone di input da parte dell'utente.

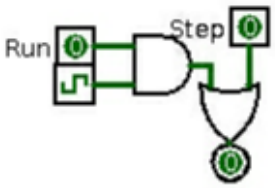
3.2.5 2or3



Questo circuito serve per controllare se dati due input da 1 bit, essi rappresentano in binario i valori 10 o 11, cioè rispettivamente i valori 2 e 3

.

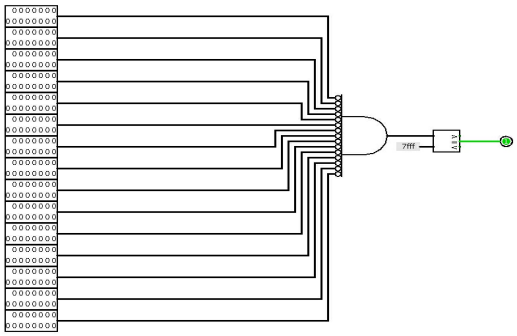
3.3 Clock



Il circuito **Clock** serve ad aggiornare l'evolversi dell'automa cellulare. Nel main è collegato allo Storage in modo da mostrare sulla matrice led i cambiamenti delle varie generazioni; è possibile avere un aggiornamento sia col pin di input "run", che permette una continua evoluzione a ogni giro di clock, oppure anche tramite un pin di "step" che permette un evoluzione step-by-step, a seconda di come vuole l'utente.

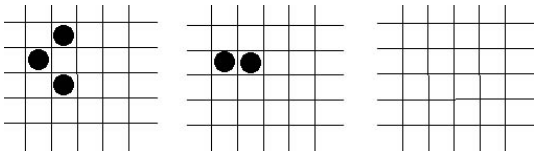
3.4 C: SetResetCounter

Il SetResetCounter serve a resettare il counter delle generazioni della popolazione, qualora il suo ciclo di vita non sia infinito; inoltre è utile per attivare l'output del register dove viene registrato il numero di generazioni raggiunto dalla configurazione iniziale, precedente al reset.
Il circuito si presenta in questo modo:



Abbiamo 15 pin di input, ciascuno da 15 data bits, che rappresentano le uscite dello storage(ovvero gli stati delle celle), collegati a un and con le porte negate.
Se tutti i pin di input sono a zero, allora l'and manderà in uscita 0xffff(cioè tutti 1 su 15 bits). Il valore viene comparato attraverso un comparator al numero 0xffff; se i valori sono uguali allora il **SetResetCounter** resetterà il counter, altrimenti non farà nulla.

4 Esempio di configurazione con evoluzione nel tempo



Si può vedere qui un trimino, o gruppo di tre organismi, che muore alla seconda generazione. Inizialmente ci sono 3 celle; alla generazione successiva se ne crea una in mezzo alle 3, per la terza regola di evoluzione di Conway, e ne muoiono due dalla generazione prima in quanto avevano meno di 2 celle confinanti; infine all'ultima generazione non rimane nessuna cella perchè entrambe le celle correnti son rimaste senza più di due celle vicine.
Questa è una configurazione che muore dopo breve tempo. La maggior parte delle configurazioni raggiungono in breve tempo forme stabili oppure oscillanti all'infinito. Esistono diverse tipologie di configurazioni:

- tipo statico (blocco, barca)
- oscillante (lampeggiatore, rospo)
- in movimento o navicelle spaziali (aliante, astronave leggera)
- fucili: stazionari che sparano alianti o navicelle spaziali
- fumatori: si muovono lasciando in coda frammenti di vita
- rastrelli: si muovono ed emettono navicelle
- reattore: lascia una coda di fucili (tasso di crescita quadratico) E' possibile trovare una classificazione di tutte le configurazioni qui → `http://www.conwaylife.com/wiki/Main_Page`.

5 Conclusioni e Considerazioni

Come output del **Counter** ho utilizzato un pin di output da 32 bits(il massimo possibile) in quanto certe configurazioni possono oscillare all'infinito.

Il gioco originale di Conway utilizza una matrice ben più grande di quella realizzata da me ma, per questioni di spazio e di bellezza del circuito principale, ho pensato fosse meglio utilizzare una matrice led 15x15.
Un'altra possibile aggiunta al progetto potrebbe essere un circuito che visualizzi il numero di generazioni della configurazione iniziale utilizzando gli hex digital display.