Nicola Melloni
mtr. 794324

Esame di Architettura degli Elaboratori I: Progetto in Logisim

Il rompicapo “2 + 2 = 3”

Regole e obiettivo del gioco

Il progetto da me realizzato è un gioco-rompicapo che ho pensato di chiamare "2+2=3" (ispirato al gioco "2048", uscito quest'anno come applicazione per dispositivi mobili quali iPad, iPhone e affini). Le regole del gioco sono le seguenti:

- L' interfaccia di gioco è composta da una griglia di caselle 4x4 contenenti una cifra esadecimale; 0 denota una cella vuota, qualsiasi altro numero indica che la cella è piena;
- Un tasto di reset permette l'inizio del gioco, per cui ogni casella o cella viene inizializzata a 0 e solo due a caso di queste vengono poi reinizializzate a 1;
- Il giocatore può effettuare per turno una della quattro mosse direzionali Up-Down-Right-Left, la quale farà scorrere all' interno della griglia nel senso della direzione indicata ogni cifra diversa da 0, che sarà poi soggetta alle seguenti regole:
    - se una cifra raggiunge il margine della griglia, vi si adagia e raggiunge una situazione stabile.
    - se lo scorrimento di una cifra la porta a ridosso di una cella già piena, il gioco si comporta così:
        - Se le cifre due cifre sono diverse, la cifra che si è mossa si stabilizza e si ferma a fianco all' altra come avendo trovato il margine della griglia.
        - Se le due cifre sono uguali, queste si "uniscono" in una sola cella, la quale conterrà poi una cifra di valore pari a quello delle due precedenti e incrementato di 1 (ad esempio, facendo scorrere due "2" nella stessa direzione e portandoli a una situazione di adiacenza, si ottiene 3, da cui il nome del gioco).
- Al termine di ogni mossa (quando il sistema raggiunge una posizione stabile per cui nessuna cella cambierà ulteriormente il proprio valore), una cella vuota scelta a caso verrà riempita da un 1, permettendo poi al giocatore di effettuare una nuova mossa.
- L' obiettivo del gioco è ottenere una cifra dal valore più alto possibile, prima che la griglia venga riempita completamente da cifre e il gioco abbia termine per l'impossibilità di eseguire altre "semplificazioni".

Struttura generale

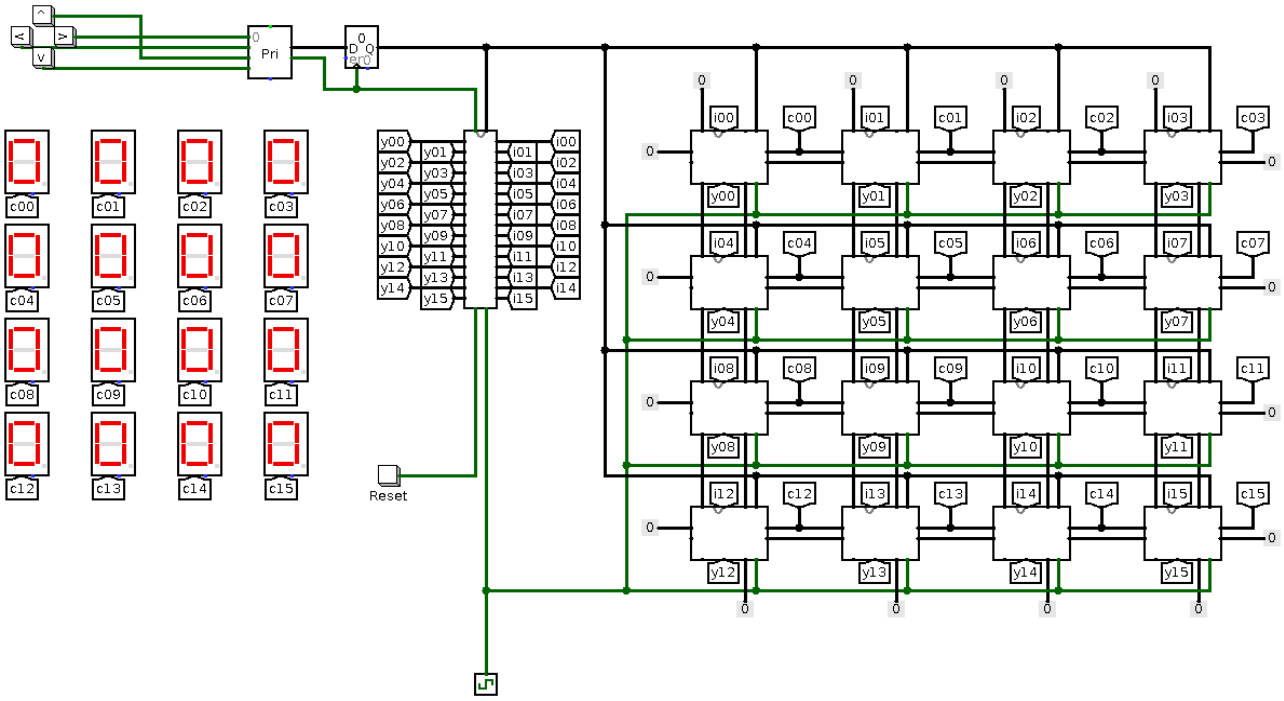
La struttura generale del circuito che realizza il gioco è composta da:

- 16 celle, ciascuna delle quali contenente:
    - un registro a 4 bit che ne memorizzi lo stato (espresso da una cifra esadecimale);
    - opportuni circuiti combinatori, che comunichino al sistema se si debba verificare lo shift o l' incremento del contenuto di una cella.

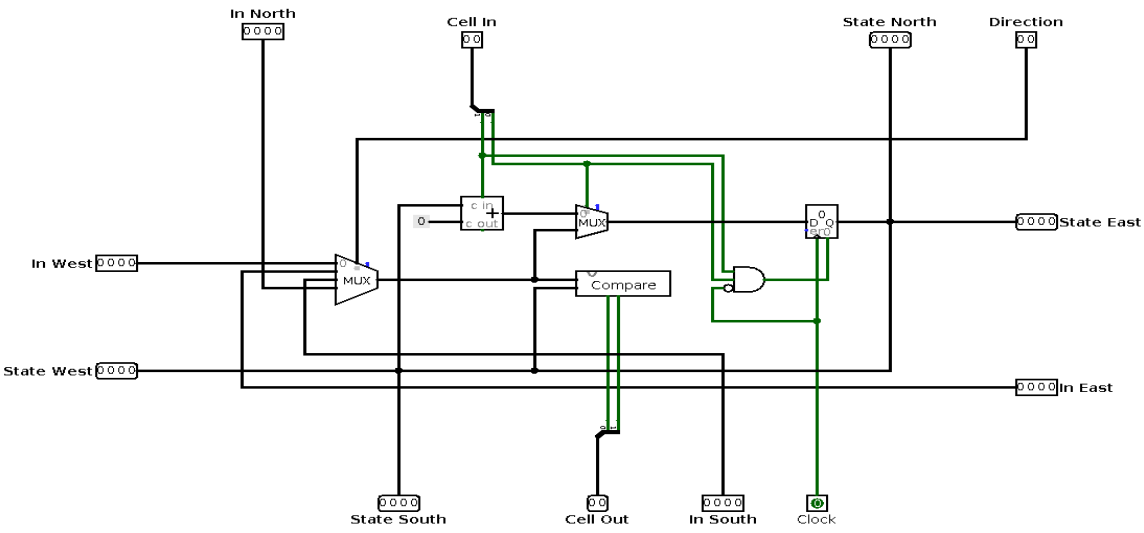
- Una UC, che riceva informazione dalle celle e ne ricavi un “quadro generale” del sistema, permetta il regolare svolgimento del gioco, e scandisca le diverse fasi di gioco (ad esempio, rendendo disponibile un tasto di reset, o riempiendo una cella vuota scelta casualmente al concludersi di ogni mossa del giocatore).

- Un piccolo circuito di ricezione, che codifichi la mossa effettuata dal giocatore e la comunichi ad ogni cella e all' UC.

Questo è composto da un ***Priority Encoder*** che assegna ad ogni mossa del giocatore (eseguita tramite l' azionamento di uno dei quattro tasti direzionali) due cifre binarie e segnali all' UC l' inizio di un turno di gioco (tramite l' uscita inferiore del dispositivo, che viene mandata a 1 nel caso un qualsiasi pulsante venga premuto).Un registro da 2 bit è messo a valle del *Priority Encoder*, affinché la mossa venga ricordata dal sistema per tutto il tempo necessario.



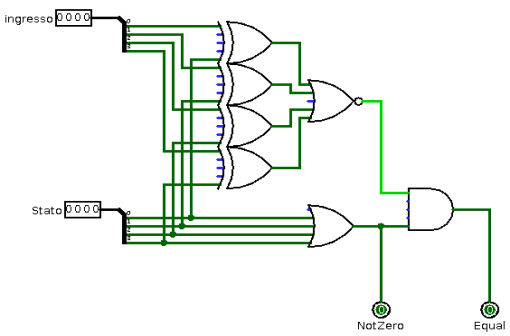
La struttura di una cella



Una cella è composta da:

- Un registro di memoria da 4 bit il cui stato viene mandato in uscita nelle quattro direzioni cardinali.
- Un primo MUX a 2 bit che seleziona come uscita uno dei quattro ingressi corrispondenti agli stati delle celle adiacenti; tale ingresso viene scelto mediante i 2 bit di selezione comunicati dal *Priority Encoder*: se lo scorrimento avviene in una tale direzione, viene selezionato l' ingresso corrispondente alla direzione inversa.

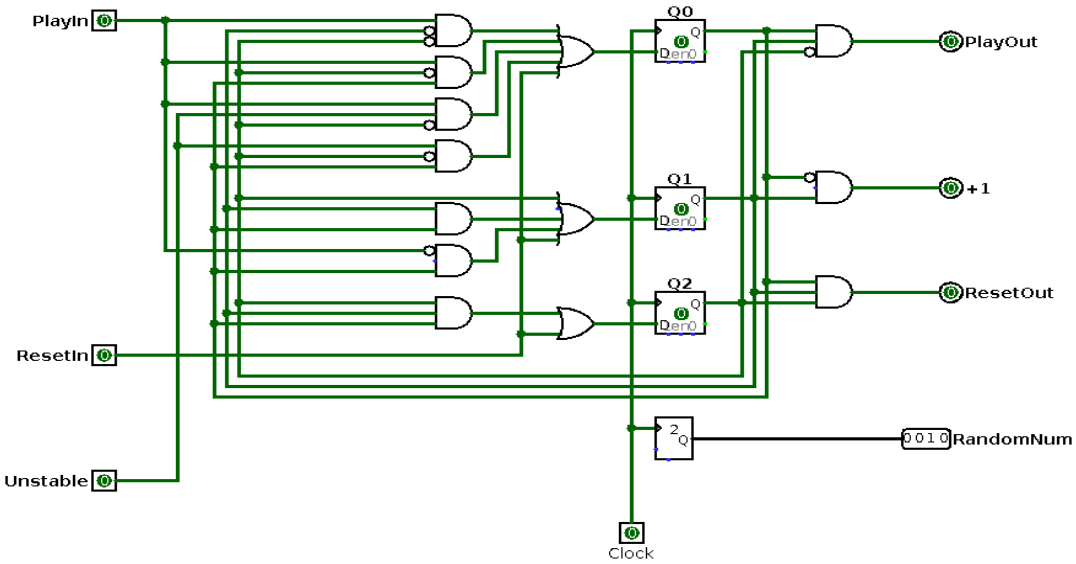
- Un **secondo MUX** a **1 bit** che indichi se lo stato della cella debba essere uguale a se stesso, al più incrementato di 1 mediante il riporto di un **addizionatore**, o se debba essere uguale all' ingresso prestabilito. ***CellIn*** è un ingresso a 2 bit proveniente dall' UC e decide lo stato prossimo della cella.



- Un **circuito** combinatorio ***Compare*.** Manda in **uscita 2 bit** che segnalano alla UC l'informazione utile al controllo dell' esecuzione: comunica se lo stato della cella è diverso da zero, o se questo è diverso da zero e uguale all' ingresso prestabilito.

- Poichè non si verifica che lo stato prossimo di una cella debba essere contemporaneamente uguale al proprio stato incrementato di 1 e uguale a un ingresso del primo Mux, la congiunzione dei due bit di *CellIn* vengono usati per una funzione di reset del registro: poiché la sua struttura consente il solo l'azzeramento asincrono, alla condizione di reset viene aggiunta la negazione del segnale di clock.

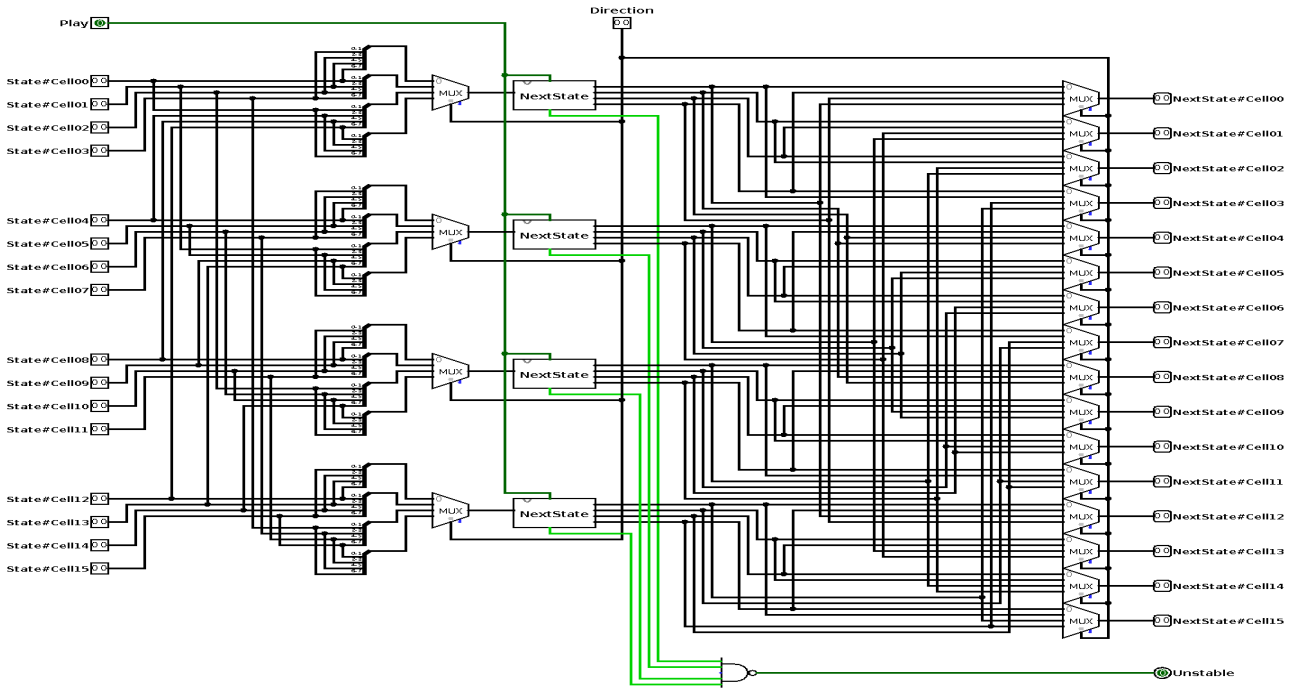
Automa a stati finiti



Un automa a stati finiti controlla la durata di un turno, genera i segnali che comandano il riempimento casuale di una cella e la funzione di reset. È composta da 3 ingressi e 4 uscite, di cui una data da un generatore di numeri casuali a 4 bit, e ha il seguente funzionamento: se l' ingresso *Unstable* è a 1, l' uscita *PlayOut* viene asserita dal fronte di discesa di *PlayIn* (data dalla pressione di un pulsante sul *Priority Encoder*) e sussiste finché l' ingresso *Unstable* non va a 0, in seguito viene posta a 1 l' uscita +1 per un giro di clock; se l' ingresso *Unstable* è a zero, l' attivazione di *PlayIn* ignora l' uscita *PlayOut* e viene asserita direttamente l' uscita +1.

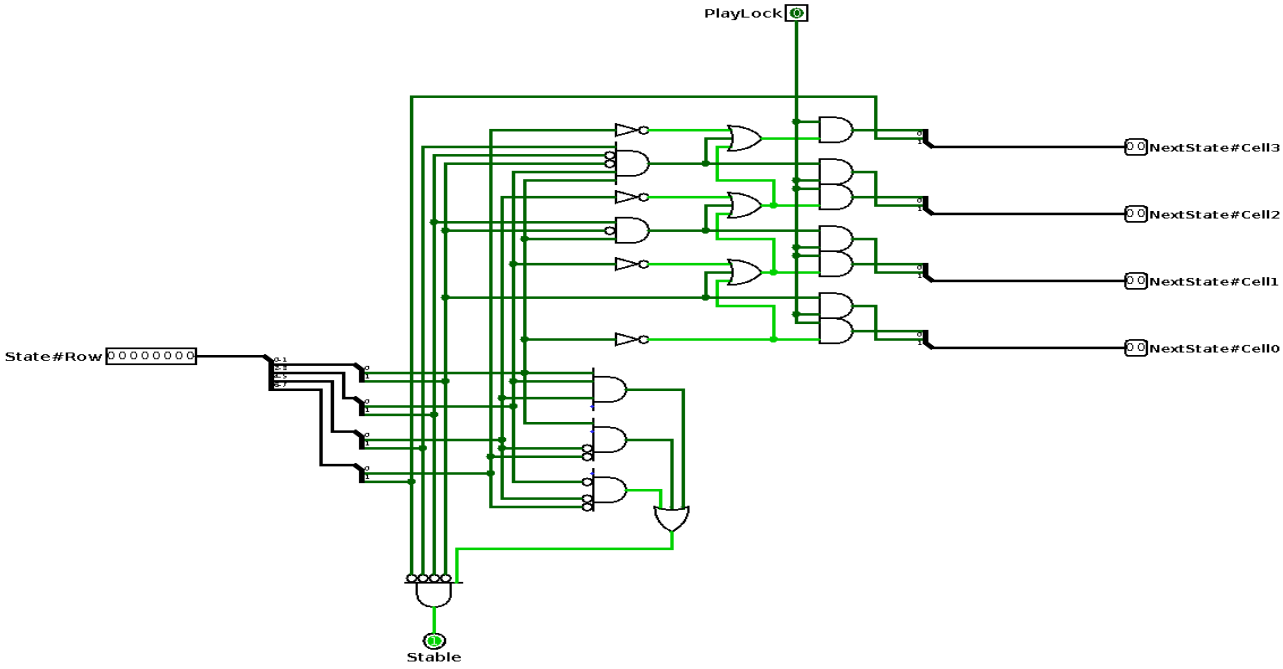
Il comando *ResetIn* ha precedenza su ogni altra operazione e permette il reset di tutte le celle mediante l' uscita *ResetOut* e l' asserzione per due giri di clock dell' uscita +1.

Circuito Play



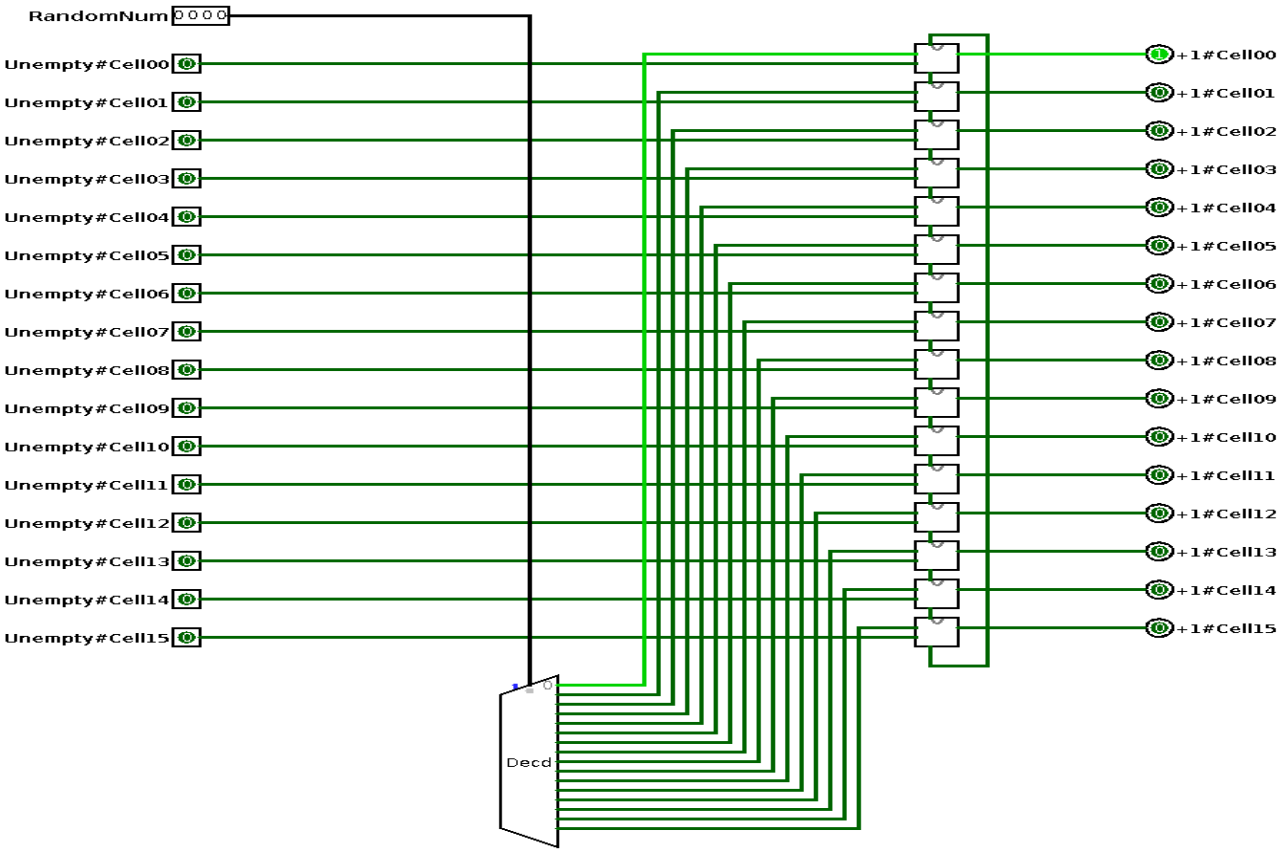
Poiché a seguito di una mossa le quattro righe o colonne della griglia risultano mutuamente disgiunte, ha senso considerarle come sistemi indipendenti il cui input e output viene deciso in funzione del senso di gioco. Una colonna di 4 mux riceve in ingresso i segnali di stato di tutte le celle e restituisce in uscita stringhe di 8 bit rappresentanti lo stato delle 4 righe o colonne. Da queste stringhe 4 circuiti di stato prossimo calcolano o stato prossimo del sistema e la stabilità di ogni riga. La seconda colonna di 16 mux si assicura infine che ogni cella riceva i giusti segnali per l'aggiornamento del suo stato.

Circuito di stato prossimo

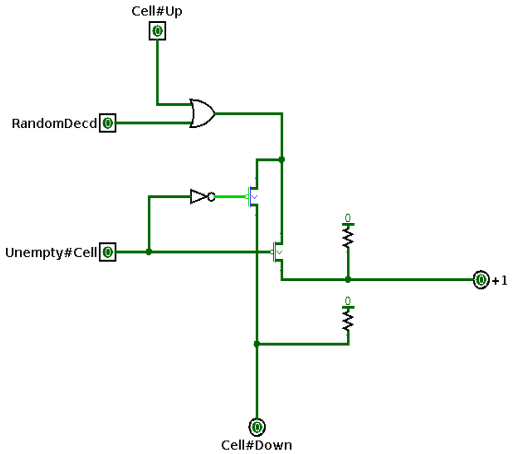


La parte superiore del circuito computa lo stato prossimo delle quattro celle di una riga o colonna designata; questi segnali vengono “bloccati” da una colonna di porte AND e “fatti passare” solo in presenza del segnale di *Play* dato dalla macchina a stati finiti. La parte inferiore del circuito asserisce l'uscita di 1 bit se e solo se la riga o colonna in questione si trova in una situazione di stabilità; questo segnale è poi inviato alla macchina a stati finiti per decidere se il turno di gioco debba esaurirsi o continuare.

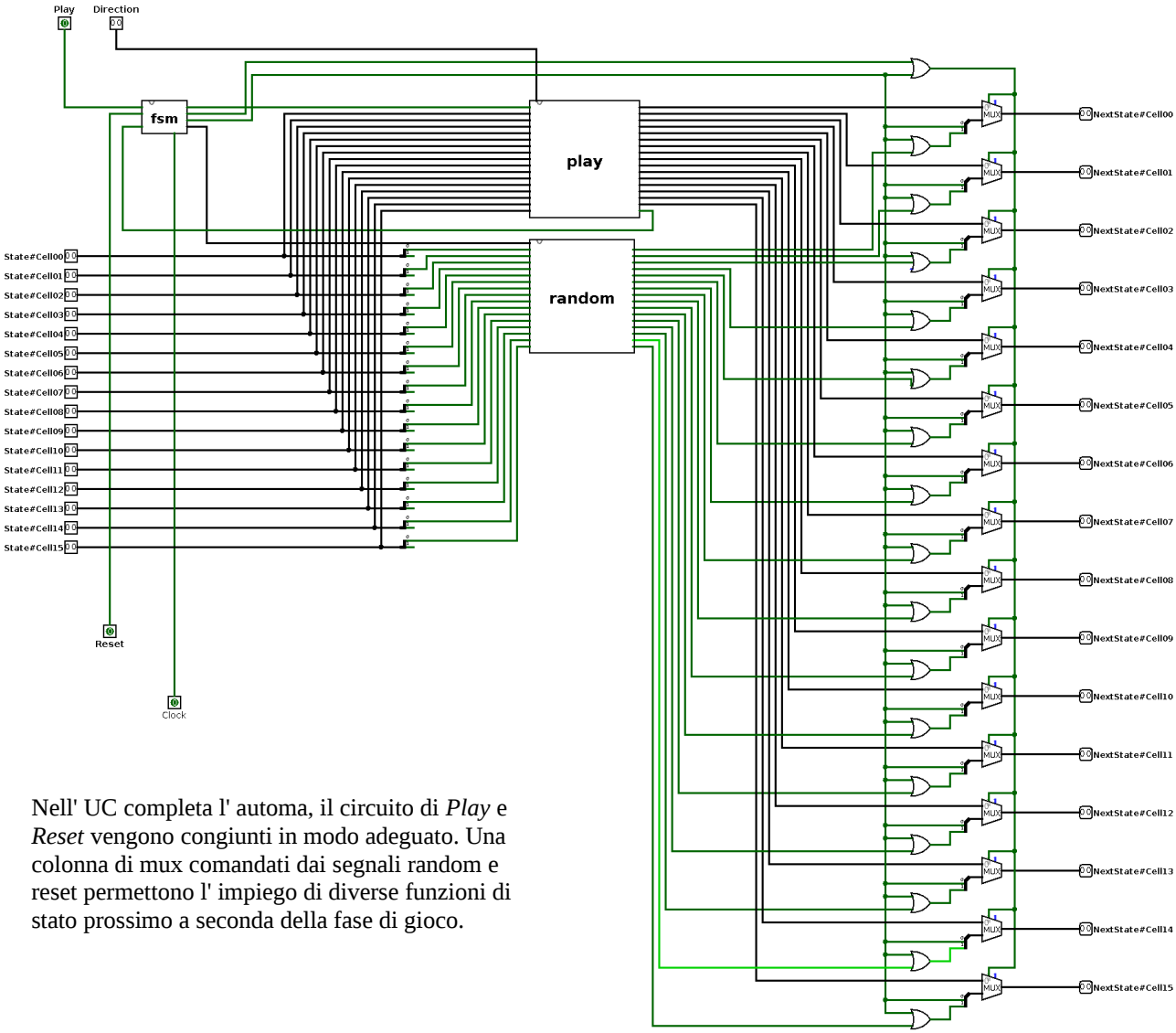
Circuito Random



Questo circuito permette il riempimento di una cella vuota in modo casuale. La selezione avviene mediante l'invio di numero casuale a un decoder a 4 bit. I 16 ingressi da un bit rappresentano lo stato di ogni cella e operano ciascuno sui gate di 2 transistor che lavorano in contrapposizione: se una cella già piena viene scelta casualmente, questa viene scollegata dall' ingresso del decoder e viene al suo posto collegata la cella successiva. Se ogni cella è stata riempita, ogni cella viene scollegata e non si verifica alcun incremento.



L' UC completa



Nell' UC completa l' automa, il circuito di *Play* e *Reset* vengono congiunti in modo adeguato. Una colonna di mux comandati dai segnali random e reset permettono l' impiego di diverse funzioni di stato prossimo a seconda della fase di gioco.

Conclusione

Senza dubbio la più grande vulnerabilità del sistema è dovuta alla totale sincronia di ogni sua parte, rendendolo esposto a glitch o interruzioni. Tuttavia vari accorgimenti sono stati usati al fine di evitare tali spiacevolezze: ad esempio, l' esecuzione di ogni mossa ha effetto sul fronte di discesa dell' ingresso *Play* affinché non vengano compiute più mosse in una sola pressione di pulsante, l' automa a stati finiti per il controllo di turni è stato progettato per comprendere una fase di ricezione per un più curato svolgimento del gioco, e ad ogni azione possibile dal giocatore è stata associata una regola di precedenza per evitare comportamenti inaspettati. Il cammino critico della circuito di stato prossimo è pari 9 (dato dalle 6 porte interne ai mux dei circuito di Play e dell'UC, più le 3 porte OR nel circuito *nextstate*), settando il clock di conseguenza la sincronizzazione del sistema dovrebbe risultare corretta anche nell' ipotesi di suo utilizzo nel mondo fisico.