Relazione del Progetto di Laboratorio Architettura degli Elaboratori I

Alessandro Di Gioacchino*
Matricola: 931271, Turno: A
alessandro.digioacchino@studenti.unimi.it

Indice

Introduzione

In questo progetto è proposta una possibile implementazione del gioco da me battezzato *Spegni le luci*.

Dopo aver iniziato una partita, alcuni dei nove LED presenti si illumineranno: scopo del giocatore è di spegnerli tutti mediante la pressione degli appositi tasti, prestando attenzione al fatto che questi non controllano unicamente la luce corrispondente ma anche quelle adiacenti ad essa.

Per la realizzazione del circuito sono stati usati diversi componenti di libreria, tra cui elementi di memoria, quali latch e flip-flop, generatori di numeri casuali su più bit, contatori, display esadecimali, ma anche i classici operatori sono risultati fondamentali per implementare alcune importanti condizioni logiche.

*Progetto approvato il 03/09/19, consegnato il 11/09/19

La relazione è strutturata come segue: in Sez. 1 presenterò una visione generale del circuito e tratterò in particolare due semplici componenti, ovvero il contatore, usato per il numero di mosse, e il cronometro. In Sez. 2 vedremo più da vicino le parti del circuito che fanno funzionare correttamente il gioco e i problemi affrontati per implementarle. In Sez. 3 trarrò infine le ultime conclusioni, e qualche idea per migliorare ulteriormente il progetto.

1 Il circuito principale

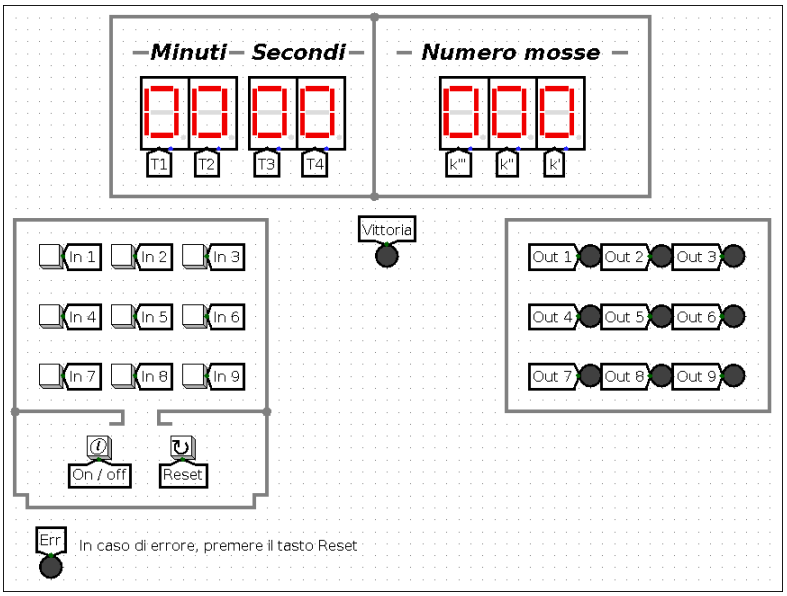

Figura 1: La schermata di gioco.

Il circuito principale è organizzato in modo da essere il più user-friendly possibile.

Il sistema di input, tutto ciò con cui l'utente può interagire, è costituito da nove bottoni disposti in un quadrato di lato tre; sotto a questi vi sono due ulteriori tasti dal nome autoesplicativo: *On / off* serve per attivare o disattivare il gioco, mentre *Reset* va premuto in caso di errore o semplicemente per iniziare una nuova partita.

A destra il sistema di output, che include nove LED disposti in maniera identica ai bottoni del sistema di input. Sono inoltre presenti due ulteriori LED: quello in basso, *Errore*, segnala all'utente un qualsiasi problema riscontrato durante la partita (che, come vedremo, potrà essere di un unico tipo), mentre quello posto tra le due scacchiere, *Vittoria*, si accende in caso di game over. In cima vi sono infine sette display esadecimali, in questa implementazione utilizzati come display decimali: il primo gruppo, composto da quattro elementi, misura il tempo impiegato per concludere la partita in minuti e secondi, mentre i restanti tre contano le mosse utilizzate.

1.1 Il contatore

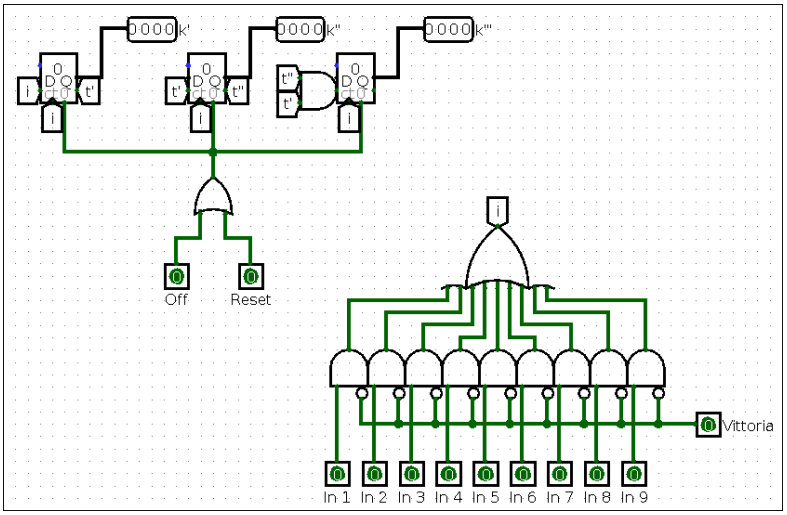

Figura 2: Implementazione di un contatore asincrono su 12 bit.

Il contatore riceve in input i segnali dei nove bottoni, in figura numerati da *In 1* a *In 9*, e il segnale di *Vittoria*. I nove AND tra questi due ingressi servono per disabilitare il contatore in caso di game over, in accordo con il seguente ragionamento:

if (!Vittoria AND (In 0 OR ... OR In 9)) then increase counter, implementato nel circuito sulla destra in figura grazie alla distributività dell'operatore AND.

Il segnale uscente dall'OR, contrassegnato dal tunnel *i*, pilota poi il clock di tutti i Counter, rendendoli in questo modo asincroni. Inoltre *i* entra anche in *Count* del Counter che gestisce le unità, innescando il cambio di stato ad ogni interazione da parte dell'utente. Il riporto di questo contatore è poi connesso a quello che si occupa delle decine, così da ottenere un incremento del numero memorizzato quando le unità raggiungono la cifra massima (nove in questa implementazione).

Il Counter che gestisce le centinaia ha bisogno di qualche attenzione in più. Se, come potrebbe essere intuitivo, vi collegassimo il riporto uscente dal contatore delle decine, non appena questo raggiunge la cifra massima le centinaia aumenterebbero in corrispondenza dell'input dell'utente: chiaramente il risultato desiderato è ben diverso. Perciò è necessario un AND all'ingresso *Count* tra il riporto del contatore delle unità e quello delle decine: così, il Counter incrementa il numero memorizzato solo quando gli altri due raggiungono la cifra di wrap around; in altri termini, le centinaia vengono incrementate quando gli altri due contatori sono a 99.

L'ingresso *Clear* è infine connesso ad un OR tra il segnale di *Off* e quello di *Reset*, in modo da cancellare lo stato di ogni contatore quando il gioco viene spento o quando si comincia una nuova partita.

1.2 Il cronometro

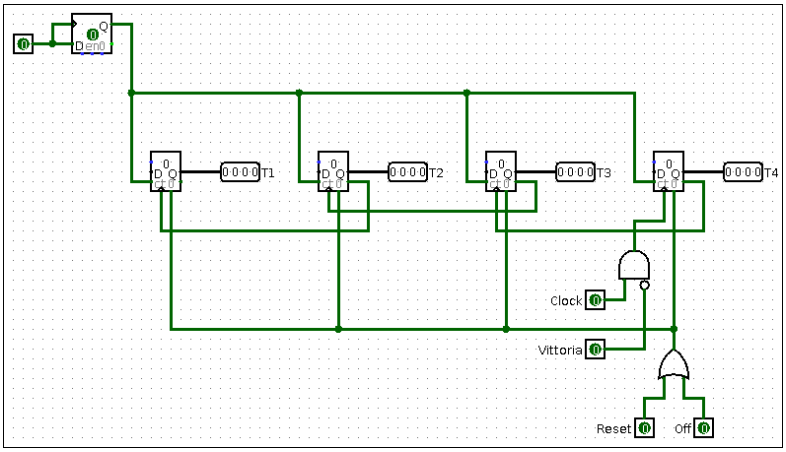

Figura 3: Implementazione di un cronometro per misurare minuti e secondi.

Il cronometro riceve in input il segnale di *On / off*: quando è asserito e la partita ha quindi avuto inizio, il T Flip-Flop in alto a sinistra memorizza il bit ricevuto e lo propaga all'ingresso dei quattro Counter; questi sono disposti in coppie: quello a destra si occupa della cifra meno significativa, per cui il massimo valore memorizzato è 9, mentre quello a sinistra arriva a contare fino a 5.

Il primo di essi, che memorizza le unità dei secondi, è pilotato da un AND tra il clock del circuito e il segnale negato di *Vittoria*:

if (Clk = 1 AND !Vittoria) then increase counter

In questo modo il cronometro si ferma non appena l'utente riesce a concludere la partita, e continua a contare altrimenti.

Il segnale di riporto di ciascun contatore è poi collegato in cascata all'ingresso *Clock* del successivo, ottenendo così un vero e proprio orologio digitale.

Come accade per quello che si occupa del numero di mosse, anche ogni Counter di questo circuito è connesso ad un OR tra il segnale di *Off* e quello di *Reset*, in modo da impostare il cronometro a zero in maniera asincrona nel caso uno dei due venga asserito.

2 I componenti fondamentali

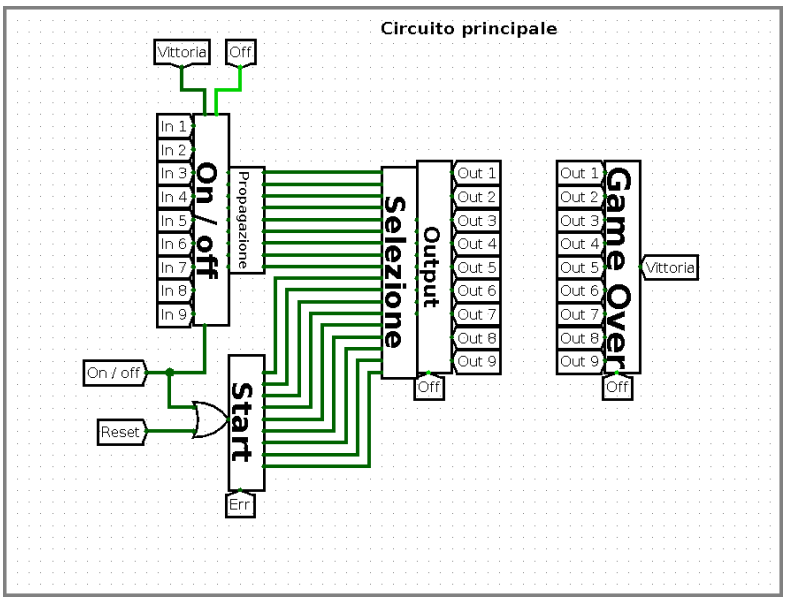

Figura 4: Tutto ciò che fa funzionare il circuito principale. Analizzeremo i moduli di questa immagine da sinistra verso destra nelle sottosezioni che seguono.

2.1 Controllo input

Scopo del circuito in figura 5 nella pagina seguente è quello di gestire l'input in maniera corretta, attivandolo solo in corrispondenza dell'inizio della partita.

Prende ovviamente in ingresso i segnali della scacchiera di gioco, come anche quello di *On / off*; in modo analogo all'implementazione del cronometro, un T Flip-Flop si occupa di memorizzare lo stato della partita. Il segnale uscente da questo elemento di memoria entra poi nella prima schiera di AND, di fatto attivando l'interazione dell'utente solo in seguito alla pressione del tasto di accensione.

Un ulteriore ingresso, quello di *Vittoria*, entra nella seconda colonna di AND assieme all'uscita dei precedenti operatori. La logica di questa parte del circuito, ancora una volta corrispondente a quella del cronometro, permette di disattivare automaticamente l'input nel caso in cui l'utente riesca a concludere il gioco.

Oltre a rielaborare l'input, *Controllo input* produce anche il segnale *Off*, asserito quando il gioco è spento, in uscita dal T Flip-Flop. In precedenza abbiamo constatato che questo segnale può essere utile per impostare a zero alcuni circuiti, come il contatore e il cronometro.

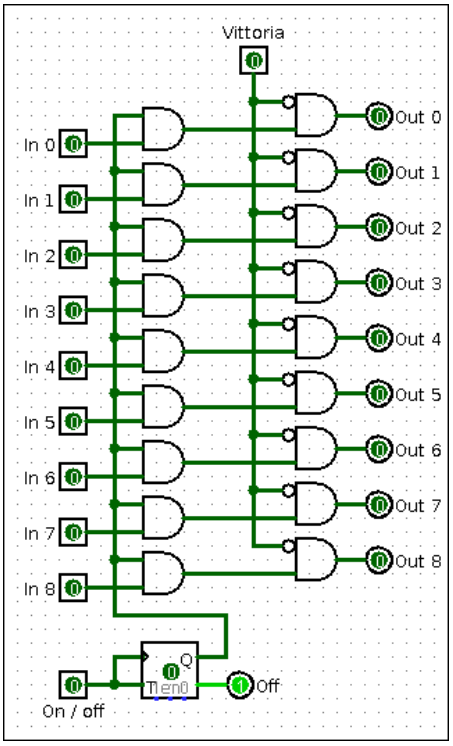

Figura 5: Il sistema di input è inattivo finché l'utente non decide di iniziare la partita con la pressione del bottone *On / off*.

2.2 Modifica input

Il circuito in figura 6 nella pagina successiva prende in ingresso i segnali prodotti da *Controllo input* e gestisce la propagazione degli stessi.

La domanda da porsi per comprendere la logica dietro *Modifica input* è: la pressione di quale bottone innesca il cambiamento di stato del LED in una certa posizione? Fermo restando che, data l'idea dietro il gioco, non vogliamo certamente che la luce in posizione *Out 9* venga accesa dal bottone *In 1*.

Consideriamo brevemente ciascuna luce e i tasti che interagiscono con essa in questa implementazione:

Etichetta del LED	Etichette dei bottoni
Out 1	In 1, In 2, In 4
Out 2	In 2, In 1, In 3, In 5
Out 3	In 3, In 2, In 6
Out 4	In 4, In 1, In 5, In 7
Out 5	In 5, In 2, In 4, In 6, In 8
Out 6	In 6, In 3, In 5, In 9
Out 7	In 7, In 4, In 8
Out 8	In 8, In 5, In 7, In 9
Out 9	In 9, In 6, In 8

La colonna di OR sortisce poi l'effetto di propagazione desiderato, secondo la seguente formula (per semplicità ne consideriamo una sola, le altre sono analoghe):

if ((In 1 = 1) OR (In 2 = 1) OR (In 4 = 1)) then turn Out 1 on / off

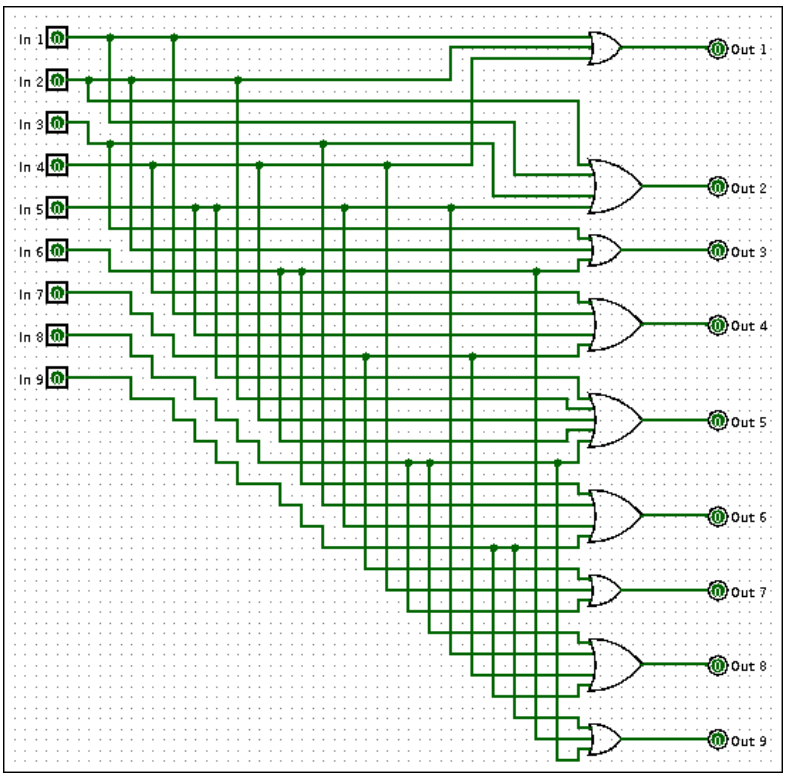

Figura 6: Scopo di questo circuito è di accendere / spegnere non solo la luce richiesta, ma anche quelle adiacenti ad essa.

2.3 Inizio casuale

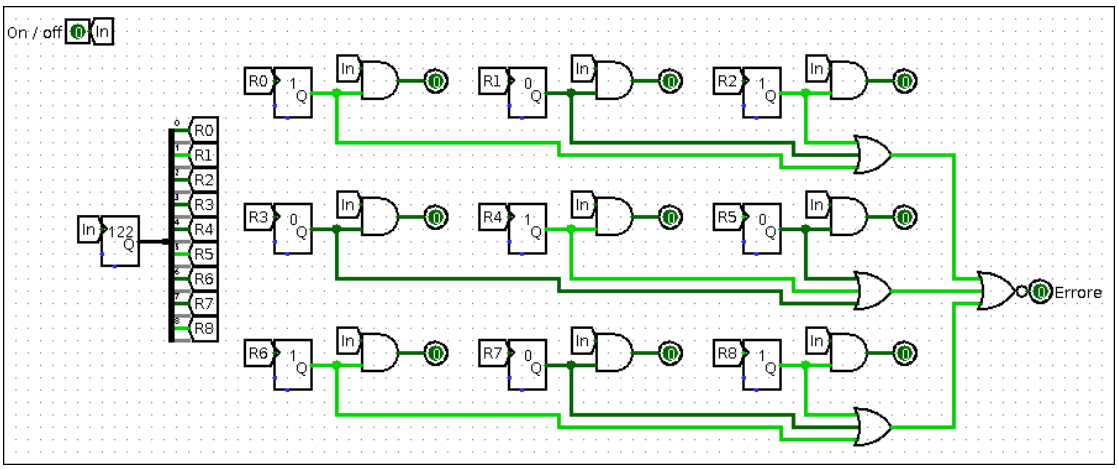

Figura 7: La pressione del bottone *On / off* causa l'accensione di alcuni LED in un pattern pseudo-casuale.

L'unico input del circuito è costituito dal segnale *On / off*: questo pilota l'ingresso *Clock* del Random Generator principale da 9 bit, innescando la generazione di un numero diverso ogni volta che viene asserito. Uno Splitter separa poi l'uscita di tale componente, in modo da ottenere nove segnali da un bit: ciascuno di questi entra infine in nove Random Generators, che generano un numero casuale tra 0 e 1.

Diversamente dagli altri circuiti che prendono in ingresso il segnale *On / off*, questo non lo memorizza in un T Flip-Flop: il motivo dietro tale scelta dipende dal fatto che *Inizio casuale* deve solo accendere qualche luce contestualmente all'inizio della partita, per poi passare immediatamente il controllo sul sistema di input all'utente. È possibile ottenere un risultato del genere in parte grazie al gestore del display, ma soprattutto mediante gli AND tra il segnale uscente dai Random Generators e *On / off*: così facendo, l'uscita del circuito viene praticamente disattivata in corrispondenza al keyup del tasto di accensione del gioco.

Un'ulteriore output di *Inizio casuale* è il segnale di *Errore*, menzionato in sezione 1 a pagina 2: si ottiene a partire dall'uscita dei Random Generators, collegate mediante un NOR.

if (!(R1 AND ... AND R9)) then turn Errore *on*, che per la legge di De Morgan diventa:

if (!R1 OR ... OR !R9) then turn Errore *on*

2.4 Selezione input

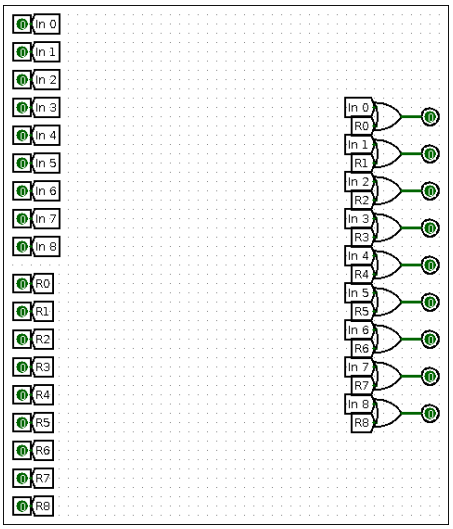

Figura 8: Come illustra il nome, *Selezione input* si occupa di unificare l'uscita dei due circuiti precedenti: *Modifica input* e *Inizio casuale*. Per fare ciò, prende in ingresso i diciotto segnali provenienti da quest'ultimi e ne calcola l'OR per determinare quali LED dovranno essere illuminati.

2.5 Display

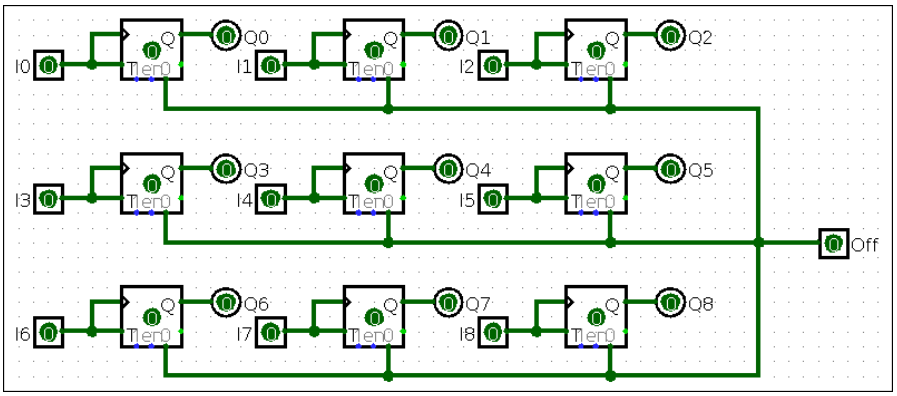

Figura 9: Il circuito atto a gestire i LED che fanno parte del sistema di output, memorizzando quale di questi debba essere illuminato o meno.

Display raccoglie infine i segnali in uscita da *Selezione input* in nove T Flip-Flop asincroni.

Come accennato in sottosezione 2.3 nella pagina precedente, questo componente aiuta anche nella coordinazione tra i Random Generators e il sistema di input che,

almeno ad inizio partita, si alternano sul controllo dell'output finale: nell'istante in cui l'utente preme il tasto *On / off*, i T Flip-Flop memorizzano i segnali uscenti dai generatori casuali; nell'istante immediatamente successivo, com'è normale che sia, tali elementi ricordano ancora il loro stato ma si apprestano a cambiarlo in base al modo in cui l'utente decide di interagire con il circuito. Ciò non sarebbe stato possibile se in *Inizio casuale* l'uscita dei Random Generators non fosse in AND con il segnale di *On / off*, poiché il giocatore non avrebbe avuto modo di sovrascrivere il contenuto dei bistabili.

Display ha anche bisogno del segnale *Off*, per poter resettare i flip-flop, e di conseguenza disattivare i LED, quando il gioco viene spento.

2.6 Vittoria

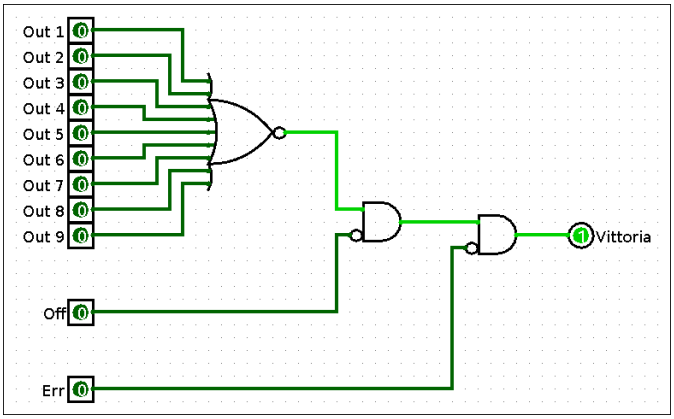

Figura 10: Implementazione del circuito che controlla il segnale *Vittoria*. Si tratta di un semplice NOR tra le etichette dei LED, il cui risultato è in AND con la negazione di *Off* e poi con quella di *Err*: *if (((!Out 1 OR . . . OR !Out 9) AND !Off) AND !ERR) then turn* Vittoria *on*. Per cui la vittoria viene segnalata soltanto quando tutte le luci sono state spente, il gioco è acceso e non si è verificato alcun errore.

3 Considerazioni finali

In questa relazione ho mostrato come implementare in un circuito logico il gioco *Spegni le luci*.

Per farlo, sono partito dalla memorizzazione asincrona dei segnali che gestiscono il display; poi mi sono concentrato sull'input dell'utente, in particolare su come organizzare il componente che si occupa di propagarlo, e sull'accensione casuale di LED che avviene ad inizio partita. Infine ho sistemato alcuni dettagli, come il segnale di *Vittoria* e il problema del doppio input, e realizzato un contatore e un cronometro per aumentare il livello di sfida.

Dal punto di vista del problema che questo circuito rappresenta, sarebbe utile capire se esistono delle combinazioni iniziali che l'utente non può risolvere: una volta individuate si potrebbero segnalare mediante *Errore*, oppure modificando l'implementazione

in modo da evitarle del tutto. Personalmente non sono stato in grado di capire se queste combinazioni esistessero o meno.

Un altro aspetto interessante potrebbe essere l'aggiunta di un contatore complementare a quello delle mosse, che va alla rovescia: raggiunto lo zero, la partita termina con una sconfitta.

Un possibile numero da cui far partire il count-down potrebbe essere nove per i giocatori più arditi: sappiamo infatti che premere lo stesso tasto due volte è inutile, perché ogni LED torna allo stato iniziale; questo vale anche nel caso in cui la doppia pressione dello stesso tasto è intervallata da un tasto diverso. Consideriamo infatti quattro tipologie di luci: quella che non viene modificata né dal tasto **A** né dal tasto **B**; quella che viene modificata dal tasto **A** ma non dal tasto **B**; quella che viene modificata dal tasto **B** ma non dal tasto **A**; quella che viene modificata sia dal tasto **A** che dal tasto **B**. Alla pressione di **A**, **B**, **A** accade che: la prima tipologia resta allo stato iniziale; la seconda tipologia torna allo stato iniziale dopo aver premuto **A** la seconda volta; la terza tipologia cambia di stato un'unica volta premendo **B**; la quarta tipologia cambia di stato tre volte e, poiché gli stati possibili sono due, ciò equivale a cambiare di stato una sola volta. In conclusione, la pressione di **A**, **B**, **A** è equivalente alla pressione del solo tasto **B**.

È per questo motivo che, tecnicamente, dovrebbe essere possibile concludere qualsiasi partita, ammesso che sia possibile farlo, in massimo nove mosse.