Esperienza 13: Semaforo

Gruppo BN
Lisa Bedini, Federico Belliardo, Marco Costa

May 7, 2017

1 Scopo dell'esperienza

Lo scopo dell'esperienza è di realizzare un semaforo come macchina a stati finiti tale che

- nello stato ENABLED esegua un ciclo in cui si abbiano accesi (per la durata di un colpo di clock e nel seguente ordine) Led Verde, Led Verde e Giallo, Led Rosso.
- nello stato NOT ENABLED faccia lampeggiare il Led Giallo (sincronamente col clock).

Il semaforo è stato realizzato sia tramite circuiti integrati, sia programmando Arduino.

2 Materiale a disposizione

- 2 SN7474 Dual D-FlipFlop
- 1 SN7400 Quad NAND gate
- 1 SN7408 Quad AND gate
- 1 7432 Quad OR gate
- DIP switch
- 3 Led: verde, giallo, rosso

3 Stato Enabled

Per realizzare il solo stato enabled abbiamo optato per una macchina di Moore, non essendoci alcun tipo di input. I tre stati della macchina sono 'Verde', 'Verde e Giallo', 'Rosso'. Abbiamo deciso di usare solo 2 FF in quanto due bit erano sufficienti per codificare i tre stati richiesti, poiché 'Verde' e 'Rosso' sono mutualmente esclusivi. Indicheremo sempre (anche nei punti successivi della relazione) Q_1 il bit più significativo, mentre Q_0 sarà sempre il bit meno significativo della codifica. In figura 1 abbiamo disegnato le transizioni e la codifica dei vari stati. Abbiamo deciso di codificare lo stato logico rimanente ($Q_1 = 0$, $Q_0 = 1$ in termini di bit) come lo stato fisico in cui il solo Led Giallo è acceso. Tale stato compare nel circuito solo nella eventualità in cui i FF si accendano in questa configurazione. Salvo in questo caso, 01 non compare più nei cicli successivi. Il vantaggio di questa codifica è che il Led Verde e il Led Giallo sono pilotati da due bit distinti, rispettivamente Q_1 e Q_0 nel nostro caso. In questo modo ci sarà più facile in seguito poter far lampeggiare il solo Led Giallo. Le tabelle di transizione implementate sono riportate in tabella **??**. I *don't care* sono già stati assegnati in questa tabella. Nelle sezioni successive diamo una breve spiegazione su come sono stati eliminati. Nella tabella D_0 e $D_!$ sono ovviamente gli ingressi dei flip-flop. Come si può vedere dalla tabella le uscite $Q_{1,n+1}$ e $Q_{0,n+1}$ diventano gli ingressi dei FF $D_{1,n}$ e $D_{0,n}$.

$Q_{1,n}$	$Q_{0,n}$	$D_{1,n}$	$D_{0,n}$	$Q_{1,n+1}$	$Q_{0,n+1}$	LV	LG	LR
0	0	1	0	1	0	0	0	1
0	1	0	0	0	0	0	1	0
1	0	1	1	1	1	1	0	0
1	1	0	0	0	0	1	1	0

Table 1: Tabella di verità delle transizioni fra stato n e il successivo $n+1$, e delle uscite (in funzione dello stato n).

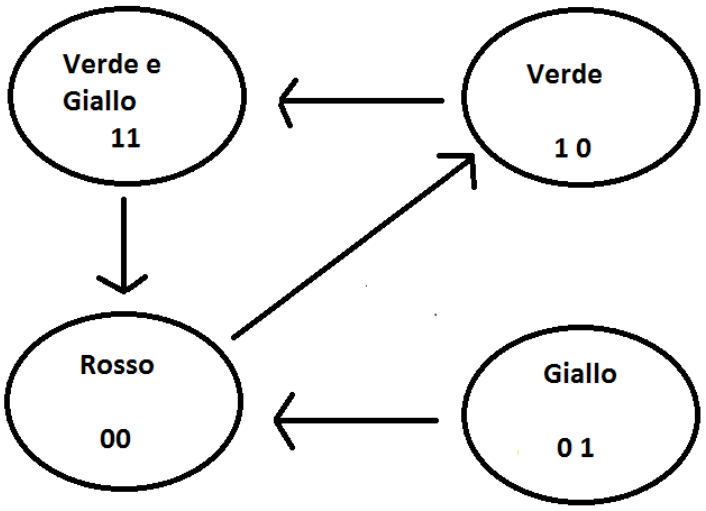

Figure 1: Diagramma dello stato Enabled. Le transizioni avvengono aad ogni colpo di clock.

3.1 Funzioni logiche delle transizioni

Nella funzione di transizione, abbiamo libertà di scegliere la transizione dello stato 'Giallo' 01. Con una semplificazione tramite mappe di Karnaugh si ottiene che ponendo i due *don't care* [1] pari a 0 si ottiene che la funzione logica richiesta è

$$Q_{1,n+1} = \bar{Q}_{0,n} \qquad Q_{0,n+1} = \bar{Q}_{0,n} \cdot Q_{1,n} \tag{1}$$

Così si ha che lo stato non richiesto 01 ("solo giallo acceso"), transisce nello stato 00 ("solo rosso acceso"). Dato che abbiamo dei FF di tipo D, il valore di $Q_{i,n+1}$ è uguale al valore dell'ingresso D_i nello stato n: minimizzare quindi il numero di operazioni logiche nelle funzioni dei $Q_{i,n+1}$ effettivamente minimizza il numero di porte logiche da implementare fisicamente.

3.2 Funzioni logiche delle uscite

Una volta codificati gli stati, abbiamo assegnato alle uscite L_V (Led Verde), L_G (Led Giallo), L_R (Led Rosso) i seguenti valori:

$$L_V = Q_1 \qquad L_G = Q_0 \qquad L_R = \bar{Q}_0 \cdot \bar{Q}_1 \tag{2}$$

Questo era in effetti il modo più semplice per realizzare i collegamenti fra i FF e le uscite: per come sono stati codificati gli stati, i Led Verde e Giallo sono pilotabili direttamente dal rispettivo bit, mentre per il Led Rosso abbiamo scelto l'unica funzione che valga 1 solo sullo stato 00.

3.3 Implementazione del circuito

Abbiamo implementato il circuito come in figura**??**. Sono stati collegati i clear e reset dei FF a $V_{CC} = 4.9 \pm 0.1$V tramite resistenze di pull-up di valore circa 1.5kΩ. Abbiamo collegato i Led a terra tramite resistenze da circa 330Ω che fungono da limitatori di corrente. Le forme d'onda del clock e del segnale a ciascuno dei tre Led sono state acquisite mediante l'oscilloscopio.

Si osservi che i segnali dei Led (e quindi delle uscite dei FF, a meno di ritardi trascurabili) diventano positivi quando il clock è sul fronte positivo (abbiamo utilizzato degli *edge-positive triggered* Flip-Flop). Inoltre, come atteso, si osserva che il Led Giallo e Rosso stanno accesi per un periodo di clock e spenti per i due successivi (Figure 3 e 4), mentre il Verde si comporta al contrario cioè sta acceso per due periodo e spento per il successivo (Figura 2).

4 Semaforo completo

Abbiamo scelto di usare una macchina di Mealy per realizzare il semaforo completo. Si è mantenuta la stessa codifica degli stati in termini di bit utilizzata precedentemente. In questo modo la funzione di transizione nello

[1] In realtà non sono prorpiamente dei *don't care*: è' lecito assegnare a questi tutte le combinazione tranne 01, caso in cui la macchina resterebbe perpetuamente in questo stato.

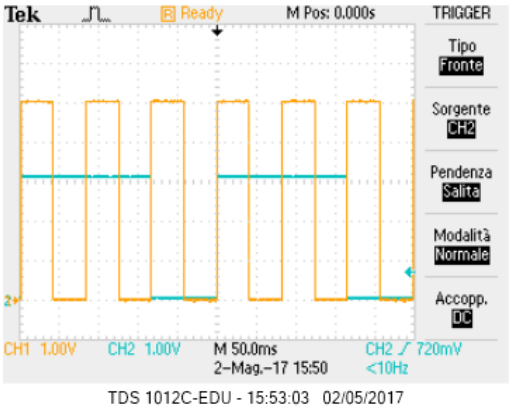

Figure 2: In CH1 il segnale del clock, in CH2 quello del Led Verde.

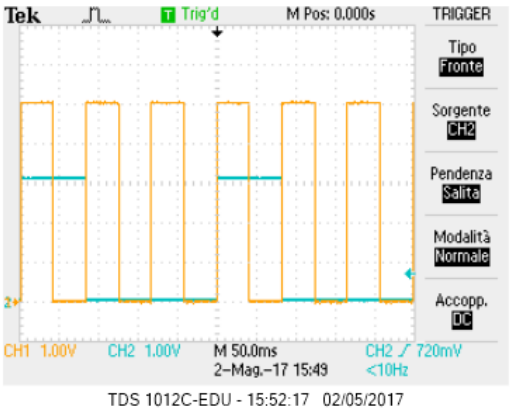

Figure 3: In CH1 il segnale del clock, in CH2 quello del Led Giallo.

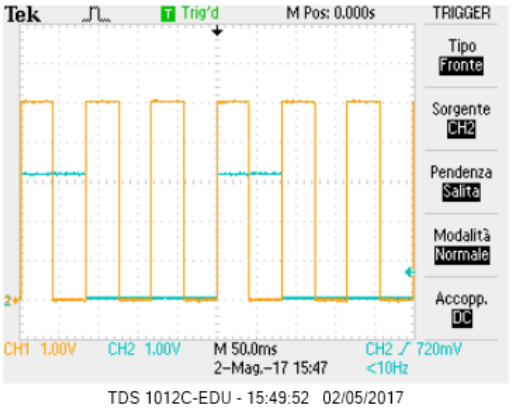

Figure 4: In CH1 il segnale del clock, in CH2 quello del Led Rosso.

state Enabled è identica a quella precedente. Abbiamo chiamato E il valore logico dell'enable. Si è deciso (per convenienza) di scegliere lo stato Enabled attivo alto (ossia quando $E = 1$). In tabella 2 abbiamo riportato le transizioni dei FF (i *don't care* sono già stati assegnati), mentre in figura 5 è stata disegnata la mappa schematica delle transizioni (nella figura gli output sono stati codificati nell'ordine LV-LG-LR).

4.1 Funzioni logiche delle transizioni

Nel caso Not Enabled, abbiamo deciso di mantenere lo stato logico 01 come stato fisico con solo il Led Giallo acceso. Si è deciso (arbitrariamente) di farlo transire verso 10: in modalità Enabled lo stato 00 aveva solo il Led Rosso acceso, pertanto con l'aggiunta di un AND con il bit E si è riusciti a mantenere il circuito già montato in precedenza. Le transizioni degli altri stati (10 e 11) sono state scelte in modo che le relative mappe di Karnaugh risultassero le più semplici possibili compatibilmente col fatto che gli stati 10 e 11 non devono ricomparire mai

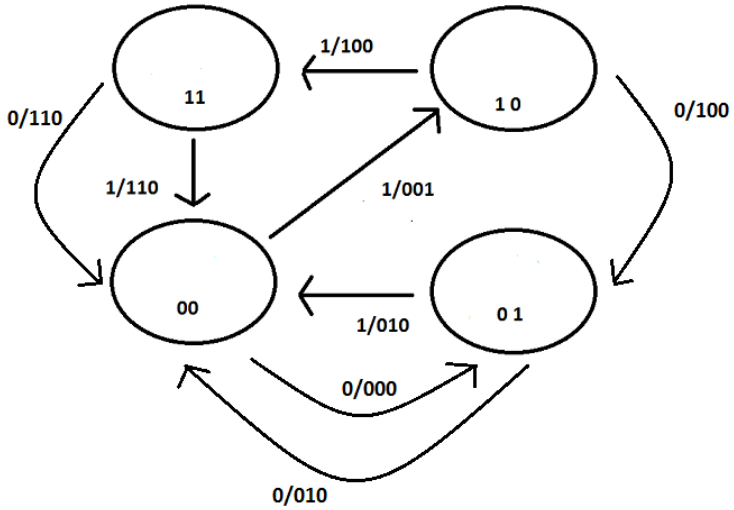

Figure 5: Diagramma delle transizioni per la macchina di Mealy realizzata. La terna degli output è LV-LG-LR.

E	$Q_{1,n}$	$Q_{0,n}$	$D_{1,n}$	$D_{0,n}$	$Q_{1,n+1}$	$Q_{0,n+1}$	LV	LG	LR
0	0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	0	0	1	0
0	1	0	0	1	0	1	1	0	0
0	1	1	0	0	0	0	1	1	0
1	0	0	1	0	1	0	0	0	1
1	0	1	0	0	0	0	0	1	0
1	1	0	1	1	1	1	1	0	0
1	1	1	0	0	0	0	1	1	0

Table 2: Matrice di transizione e valori di verità per le uscite nel caso di semaforo completo. (Uscite funzione dello stato n)

più nel ciclo. Abbiamo così ottenuto come funzioni logiche per le transizioni:

$$Q_{0,n+1} = \bar{Q}_{0,n} \cdot (\bar{E} + Q_{1,n}) \qquad Q_{1,n+1} = E \cdot \bar{Q}_{0,n} \tag{3}$$

4.2 Funzioni logiche delle uscite

Con lo stesso modo si sono scelte le funzioni di output dei Led relativamente ai soliti stati indesiderati 11 e 10. Stavolta i valori logici degli output sono dei veri *don't care* (possiamo effettivamente assegnare ad essi qualsiasi valore logico). Si sono scelte dunque le seguenti equazioni:

$$LV = Q_1 \qquad LG = Q_0 \qquad LR = E \cdot (\bar{Q}_1 \cdot \bar{Q}_0) \tag{4}$$

Si osservi che il Led Verde in teoria può essere acceso nello stato Not Enabled, ma questo può accadere solo se la macchina viene inizializzata in uno degli stati in cui è acceso. Dopo il primo colpo di clock il Led Verde si spegne e non si riaccende più. La macchina entra infatti nell'oscillazione desiderata.

4.3 Implementazione del circuito

Abbiamo implementato il circuito in figura **??**. Si osservi che in questa situazione è stato implementato un meccanismo di abilitazione asincrona: non appena si cambia lo stato di E, indipendentemente dal fronte del clock, si ha che i Led cambiano stato con effetto immediato. Dalla figura **??** si può in effetti vedere che il passaggio di E da 1 a 0 disabilita istantaneamente la porta AND che precede il Led Rosso. Si osservi tuttavia che lo stato dei FF non è cambiato: questi cambiano solo dopo un colpo di clock. Dunque in un certo senso l'effetto dell'abilitazione asincrona del semaforo rende indefinito lo stato del semaforo fino al prossimo colpo di clock.

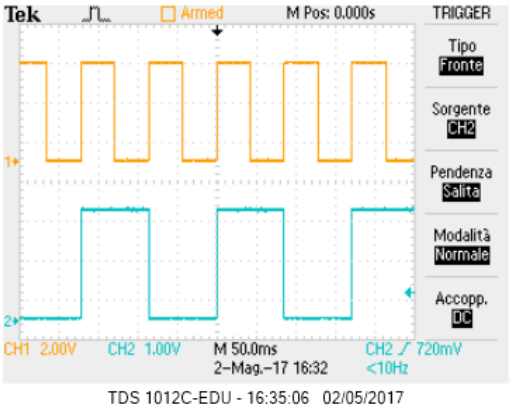

Figure 6: In alto il sengale del clock, in basso l'uscita al Led Giallo in modalità Not Enabled

In figura 6 possiamo osservare come nello stato Not Enabled il Led Giallo si accenda e si spenga il periodo di clock successivo. Il semaforo funziona come programmato.

4.4 Abilitazione sincrona

Per realizzare un meccanismo di abilitazione sincrona, ossia per fare in modo che il semaforo registri solo cambiamenti di E che avvengono sul fronte alto del clock, si è collegato E all'AND che lo porta al Led Rosso (che causava cambi di output istantanei) tramite un terzo D-Latch, come in figura**??**. E' sufficiente l'aggiunta di questo solo FF perché gli altri due collegamenti che coinvolgono l'Enable sono verso gli ingressi D dei FF, che registrano i cambiamenti degli input in modo sincrono.

5 Semaforo con Arduino

Adesso vogliamo implementare lo stesso semaforo programmando Arduino. Abbiamo optato per una macchina di tipo Moore, più semplice concettualmente da realizzare. Abbiamo collegato le uscite di Arduino 9, 10, 11 tramite il buffer rispettivamente ai Led Verde, Giallo, Rosso, inserendo resistenza da 330Ω su ciascuno per limitare la corrente. Queste uscite sono state dichiarate come OUTPUT nel programma utilizzato. L'enable è stato collegato all'uscita 8, e nel programma è dichiarato come INPUT-PULLUP (ossia alto se interruttore aperto e basso se chiuso). Su Arduino ad ogni Led è stato assegnato un bit, la codifica è quindi quella banale. Sono stati considerati solo i vari stati che ci servono: quelli indesiderati non si presentano mai dato che l'inizializzazione (nello stato in cui tutto è spento) viene fatta da noi nel programma (all'interno dell'*init*) e non casualmente come nel caso dei FF. Il funzionamento del programma mima il comportamento della macchina a stati finiti precedentemente costruita:

- Legge il valore dell'Enable (attivo alto)
- Legge lo stato in cui si trova attualmente
- Calcola lo stato successivo in cui andare a seconda dello stato attuale e del valore di Enable
- Accende i Led relativi allo stato in cui si trova la macchina (E' di tipo Moore, quindi lo stato determina le uscite) e attende per il tempo preimpostato.
- Dopo l'attesa, ripete il ciclo dall'inizio

Si osservi che in questa macchina si ha una abilitazione sincrona (avviene solo ad inizio ciclo la lettura dell'input)

6 Conclusioni

Abbiamo concludo con successo le realizzazioni delle macchine secondo le specifiche richieste. Si è realizzato in aggiunta un meccanismo di Enable sia asincrono che sincrono per il semaforo costruito con i FF.