

# Esperienza 13: Semaforo

Gruppo BN

Lisa Bedini, Federico Belliardo, Marco Costa

7 maggio 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. 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 rimanente (10 in termini di bit) come lo stato in cui il solo Led Giallo è acceso. Tale stato compare nel circuito solo nella eventualità in cui i FF si accendano in questo stato. 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 stati assegnati in questa tabella: nelle sezioni successive diamo una breve spiegazione su come sono stati eliminati).

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

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

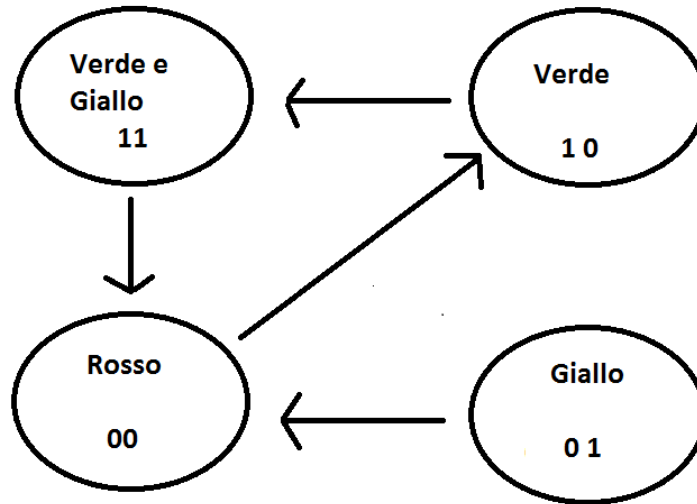


Figura 1: Diagramma dello stato Enabled. Le transizioni avvengono aad ogni colpo di clock.

### 3.1 Funzioni logiche delle transizioni

Nella funzione di transizione, abbiamo libertà di scegliere la transizione dello stato 'Giallo' 01. Con un rapido studio tramite mappe di Karnaugh ci si convince facilmente che ponendo i due don't care<sup>1</sup> pari a 0 si ottiene che la funzione richiesta è

$$Q_{1,n+1} = \bar{Q}_{0,n} \quad Q_{0,n+1} = \bar{Q}_{0,n} \cdot Q_{1,n} \quad (1)$$

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 le funzioni logiche 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:

$$L_V = Q_1 \quad L_G = Q_0 \quad L_R = \bar{Q}_0 \cdot \bar{Q}_1 \quad (2)$$

Questo era in effetti il modo più semplice per realizzare i collegamenti fra i FF e le uscite: per come erano 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???. Abbiamo collegato i clear e reset dei FF a  $V_{CC} = 4.9 \pm 0.1V$  tramite resistenze di pull-up di resistenza circa  $1.5k\Omega$ . Abbiamo collegato i Led a terra tramite resistenza da circa  $330\Omega$  per limitare la corrente. Abbiamo preso le forme d'onda di Clock e ciascuno dei tre Led tramite 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 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 il contrario (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 stato Enabled è identica a quella precedente. Abbiamo chiamato E il valore logico dell'enable. Abbiamo deciso

<sup>1</sup>In realtà non sono propriamente dei don't care: è lecito assegnare a questi tutte le combinazioni tranne 01, caso in cui la macchina resterebbe perpetuamente in questo stato.

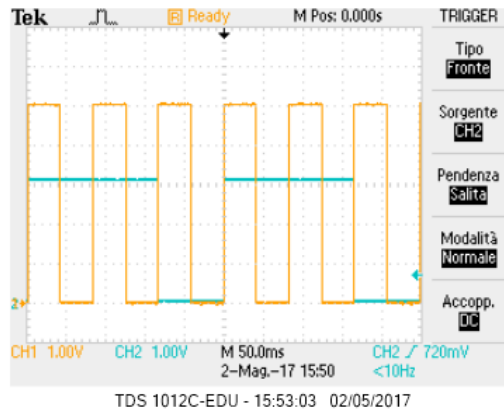


Figura 2: In CH1 il segnale del clock, in CH2 quello del Led Verde.

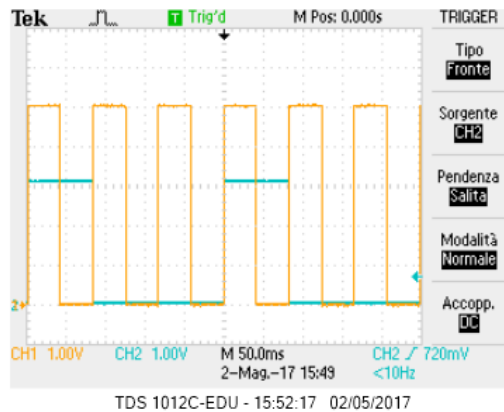


Figura 3: In CH1 il segnale del clock, in CH2 quello del Led Giallo.

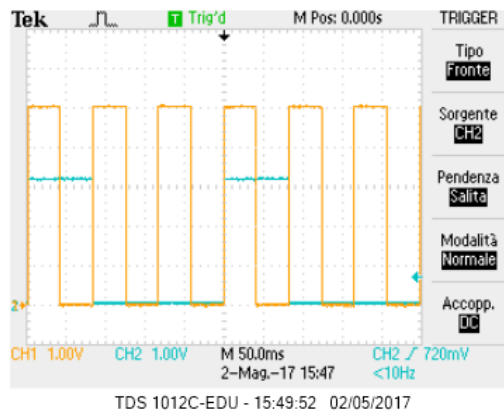


Figura 4: In CH1 il segnale del clock, in CH2 quello del Led Rosso.

di scegliere lo stato Enabled attivo alto (ossia quando  $E = 1$ ): In tabella 2 abbiamo le transizioni e i valori delle uscite implementata (i don't care sono stati assegnati), mentre in figura 5 abbiamo disegnato la mappa 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 01 ad essere con solo il Led Giallo acceso. Abbiamo deciso di farlo transire verso 10 : in modalità Enabled lo stato 00 aveva solo il Led Rosso acceso, pertanto con una 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 permanere mai nel ciclo. Abbiamo così

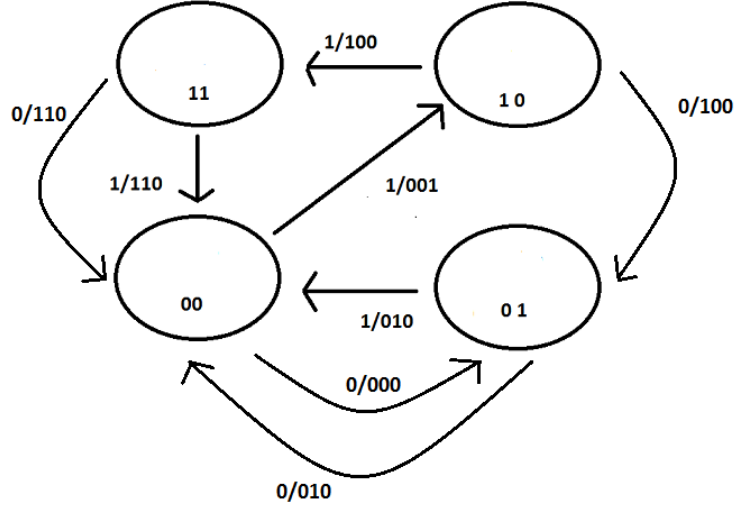


Figura 5: Diagramma delle transizioni per la macchina di Mealy realizzata. La terna degli output è LV-LG-LR.

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

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

ottenuto come funzioni logiche per le transizioni:

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

## 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). Abbiamo ottenuto le seguenti equazioni:

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

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 di quegli stati, poi dopo la prima transizione resta sempre spento SE  $E = 0$

## 4.3 Implementazione del circuito

Abbiamo implementato il circuito in figura ?? . Si osservi che in questo caso abbiamo implementato un meccanismo di abilitazione asincrona: non appena si cambia lo stato di  $E$ , indipendentemente dal fronte del clock, si ha che il semaforo cambia stato di abilitazione. 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. Pertanto se si è nello stato enabled  $E = 1$  a  $E = 0$  mentre è acceso il Led Rosso, si ha un immediato suo immediato spegnimento (e quindi si hanno tutte le uscite con valore logico 0). Si osservi tuttavia che lo stato dei FF non è cambiato: questi cambiano solo dopo un colpo di clock. In figura ?? possiamo osservare come nello stato Not Enabled il Led Giallo si accenda e si spenga il periodo di clock successivo.

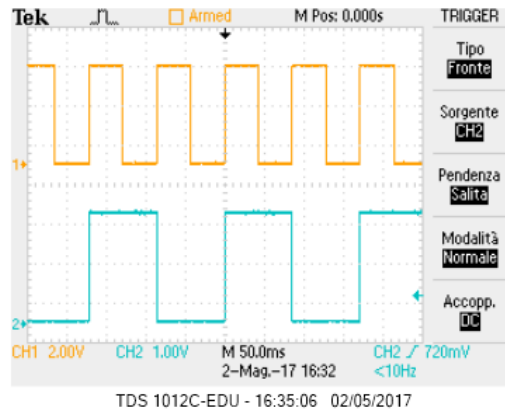


Figura 6: In alto il segnale del clock, in basso l'uscita al Led Giallo in modalità Not Enabled

#### 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 perchè gli altri due collegamenti che si hanno con 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 330 $\Omega$  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). Dato che con Arduino si hanno in genere un numero di bit a sufficienza, nello scrivere il programma abbiamo assegnato ad ogni Led un bit diverso. In questo modo si ha uno stato per ogni Led acceso, più lo stato spento e lo stato in cui Verde e Giallo sono accesi. Abbiamo considerato 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 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) per il tempo impostato
- 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 realizzato una macchina a stati finiti secondo le specifiche richieste. Si è realizzato un meccanismo di Enable sia asincrono che sincrono. Non sono state riscontrate deviazioni del comportamento dei circuiti costruiti rispetto a quanto atteso nel caso di macchina ideale.