Esercitazione 13 - Gruppo AC
Macchine a Stati Finiti: progettazione e implementazione del controllo di un semaforo

Marco Cilibrasi Tommaso Pajero

30 aprile 2015

L'esercitazione consiste nella progettazione e implementazione di un circuito predisposto alla gestione di un semplice semaforo (macchina a stati finiti).

1-2. Specifiche del circuito gestore del semaforo e prime linee di progetto

Si vuole realizzare un circuito che controlli un semaforo implementato tramite tre LED giallo, verde e rosso, i cui segnali d'ingresso - alti quando la luce è accesa -, si indicheranno con LV, LG e LR. Tale semaforo deve essere in grado di funzionare in due modalità differenti:

1. ABILITATO - cicli di un colpo di *clock* verde, un colpo di *clock* giallo e verde e un ultimo colpo di *clock* rosso;
2. DISABILITATO - luce gialla lampeggiante, con il LED acceso un colpo di *clock* sì e il seguente no.

La modalità di funzionamento del semaforo sarà decisa per mezzo di un di un interruttore che genera il segnale E, basso in modalità ABILITATO.

Poiché non si vuole mai che il LED verde e quello rosso siano accesi contemporaneamente, essi saranno controllati con l'uscita di un D-FF e con la sua negazione. Analogamente, poiché nella modalità DISABILITATO si vuole che il LED giallo lampeggi, si controllerà la sua entrata con un JK-FF, che in caso di entrata $J = K = 1$ implementa la funzione *toggle*.

In seguito, si indicheranno rispettivamente D e Q_D i segnali in ingresso e in uscita dal D-FF e analogamente JK e Q_{JK} quelli del JK-FF[1].

3. Progetto del circuito per il semaforo nella modalità ABILITATO

Si riporta in figura 1 il diagramma della FSM che gestisce il semaforo nella modalità ABILITATO.

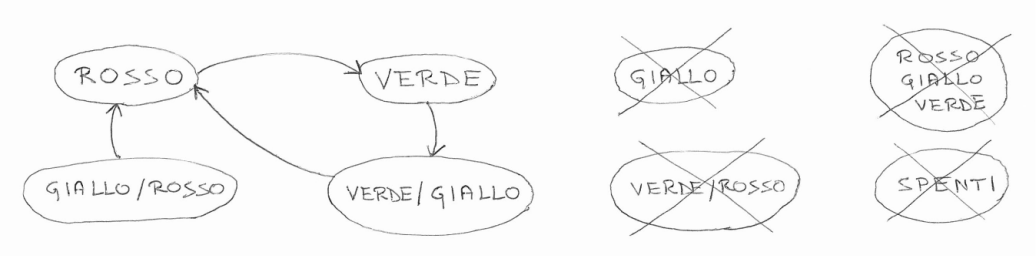


Figura 1: Diagramma della FSM che gestisce il semaforo nella modalità ABILITATO.

Le transizioni tra stati avvengono a ogni colpo di *clock*[2], seguendo lo schema presentato al punto 1 del paragrafo precedente. In particolare, può capitare che il semaforo si trovi nello stato GIALLO-ROSSO al

[1] Le due linee di ingresso J e K saranno sempre collegate insieme.

[2] Poiché il D-FF cambia il suo stato in corrispondenza del fronte in salita del *clock* e il JK-FF in corrispondenza del fronte in discesa, il segnale di *clock* sarà opportunamente negato prima di essere mandato al JK-FF, in modo da sincronizzare i due dispositivi.

momento dell'accensione[3]; in tal caso, esso passerà allo stato ROSSO al successivo colpo di *clock*. Non sono invece presenti altri stati indesiderati. Infatti, poiché si sceglie di usare $LV = Q_D$ e $LR = \overline{Q_D}$, ad ogni istante uno e solo uno fra il LED verde e quello rosso sarà acceso.

La tavola di verità che implementa il diagramma di figura 1 è riportata in tabella 1.

Tabella 1: Tavola di verità per il semaforo in modalità ABILITATO; COD ST sta per "codifica dello stato", INGR FF per "ingresso dei *flip-flop*".

	STATO n			COD ST n			STATO n+1			COD ST n+1		INGR FF	
	LV^n	LR^n	LG^n	Q_D^n	Q_{JK}^n		LV^{n+1}	LR^{n+1}	LG^{n+1}	Q_D^{n+1}	Q_{JK}^{n+1}	D	J/K
R	0	1	0	0	0	V	1	0	0	1	0	1	0
V	1	0	0	1	0	V/G	1	0	1	1	1	1	1
V/G	1	0	1	1	1	R	0	1	0	0	0	0	1
G/R	0	1	1	0	1	R	0	1	0	0	0	0	1

Le equazioni che definiscono i segnali per i LED e per gli ingressi dei FF sono:

- $LV = Q_D$
- $LR = \overline{Q_D}$
- $LG = Q_{JK}$
- $D = \overline{Q_{JK}}$
- $JK = Q_D + Q_{JK}$

Lo schema circuitale relativo è disegnato in figura 2; il montaggio e la verifica del corretto funzionamento sono stati fatti direttamente per il circuito, ampliato e più complesso, del punto 4.

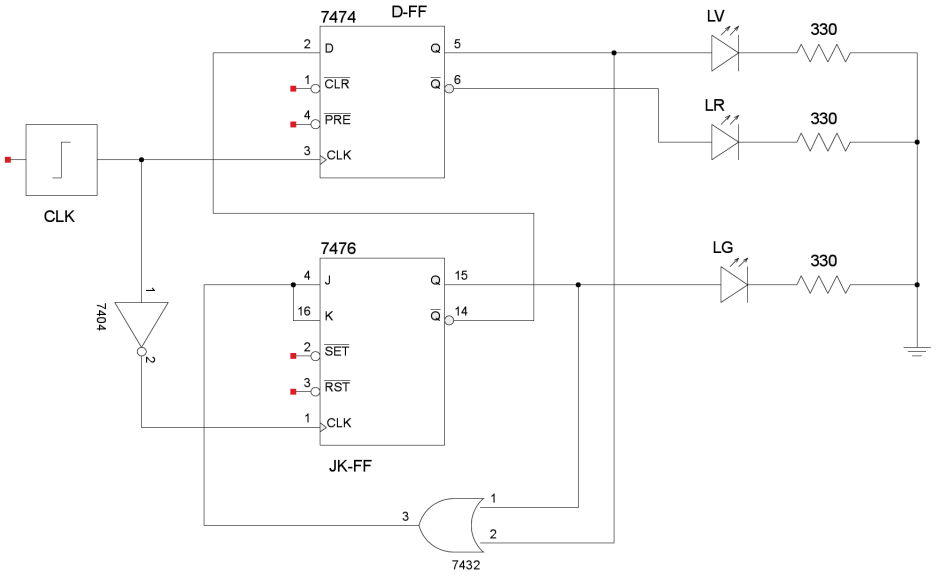


Figura 2: Schema circuitale della FSM che gestisce il semaforo in modalità ABILITATO.

4. Semaforo con lampeggiante

4.1-2 Progetto

Si vuole modificare il progetto del punto 3 in modo che il semaforo possa funzionare in entrambe le modalità, a seconda del valore dell'ingresso di abilitazione E. La tavola di verità, estensione della tabella 1, è riportata in tabella 2.

[3]È invece impossibile che la FMS si trovi in tale stato durante il normale funzionamento (non ci sono frecce che puntino verso l'ellisse relativa).

Tabella 2: Tavola di verità per il semaforo funzionante in entrambe le modalità; IN indica l'ingresso alla FSM.

IN	STATO n				COD ST n		STATO n+1				COD ST n+1		INGR FF	
E		LV^n	LR^n	LG^n	Q_D^n	Q_{JK}^n		LV^{n+1}	LR^{n+1}	LG^{n+1}	Q_D^{n+1}	Q_{JK}^{n+1}	D	J/K
0	R	0	1	0	0	0	V	1	0	0	1	0	1	0
0	V	1	0	0	1	0	V/G	1	0	1	1	1	1	1
0	V/G	1	0	1	1	1	R	0	1	0	0	0	0	1
0	G/R	0	1	1	0	1	R	0	1	0	0	0	0	1
1	G	0	0	1	x	1	NULL	0	0	0	0	0	0	1
1	NULL	0	0	0	x	0	G	0	0	1	0	1	0	1

Si noti che in modalità DISABILITATO si è scelto $Q_D^{n+1} = 0$ indipendentemente dallo stato della FSM. Ciò garantisce che, nel momento in cui il semaforo viene abilitato, esso riprenda il suo funzionamento abituale dallo stato ROSSO oppure GIALLO-ROSSO[4]. Le equazioni corrispondenti alla tavola di verità sono:

- $LV = Q_D \cdot \overline{E}$
- $LR = \overline{Q_D} \cdot \overline{E}$
- $LG = Q_{JK}$
- $D = \overline{Q_{JK}} \cdot \overline{E}$
- $JK = Q_D + Q_{JK} + E$

Lo schema del circuito che implementa la FSM gestrice del semaforo è riportato in figura 3. Esso è stato montato collegando gli ingressi asincroni dei FF (*Preset* e *Clear*), nonché l'ingresso '1' dell'interruttore, alla tensione di alimentazione da 5 V tramite delle resistenze di *pull-up* da 1 kΩ; le porte logiche NOT sono state implementate inviando il segnale da negare ai due ingressi di una stessa porta NAND.

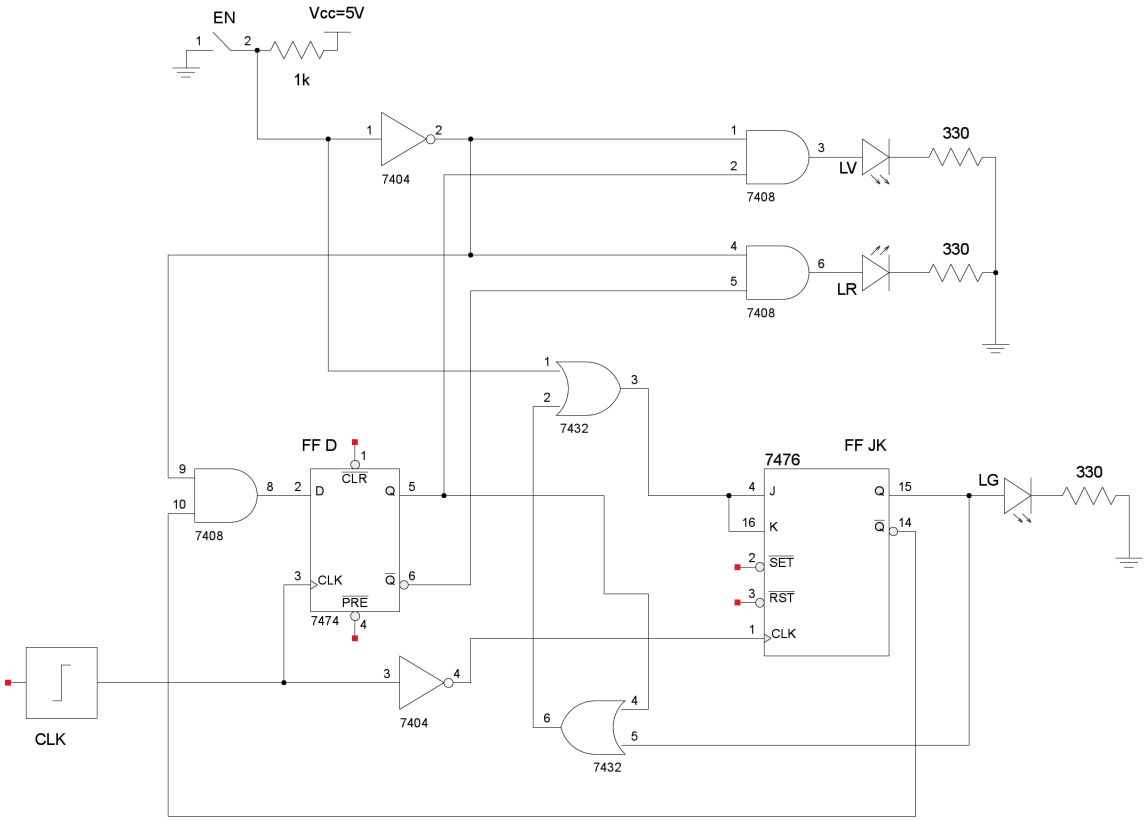


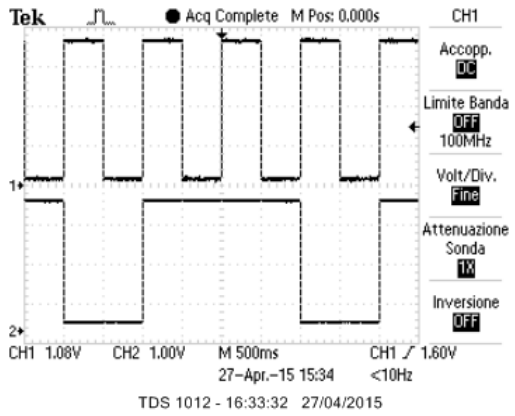
Figura 3: Schema circuitale della FSM che gestisce il semaforo (asincrono).

[4]Se non si richiede questa caratteristica, si può far assumere a Q_D^{n+1} (e quindi a D) un valore qualsiasi quando $E = 1$, il che rende possibile una diminuzione del numero di componenti e di collegamenti utilizzati nel circuito.

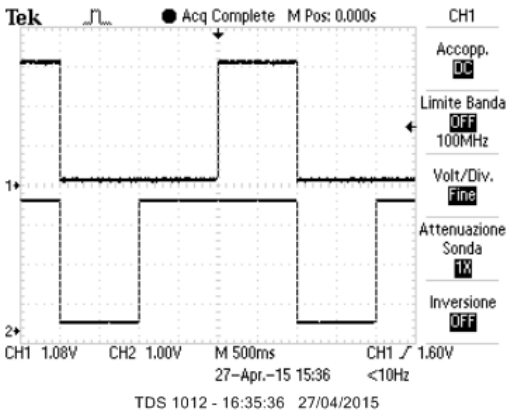
4.3 Verifica del corretto funzionamento del circuito

Si è verificato che il semaforo si comportasse secondo quanto previsto, inviando ai FF un segnale di *clock* di frequenza $\simeq$ 1 Hz, generato tramite l'impulsatore montato nell'esercitazione 10. Si sono esplorate entrambe le modalità di funzionamento ABILITATO ($E = 0$) e DISABILITATO ($E = 1$); le schermate che dimostrano il corretto funzionamento della FSM sono riportate in figura 4. Infine, si è controllato che dopo la transizione DISABILITATO-ABILITATO il semaforo ricominciasse i suoi cicli a partire dallo stato ROSSO o GIALLO/ROSSO, come desiderato.

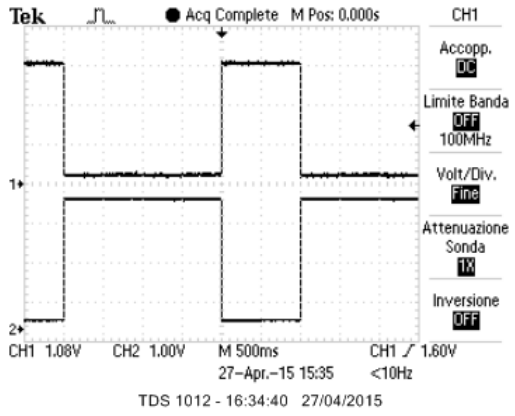
Figura 4: Segnali LV, LR, LG e di *clock* per il circuito nelle due modalità di funzionamento.



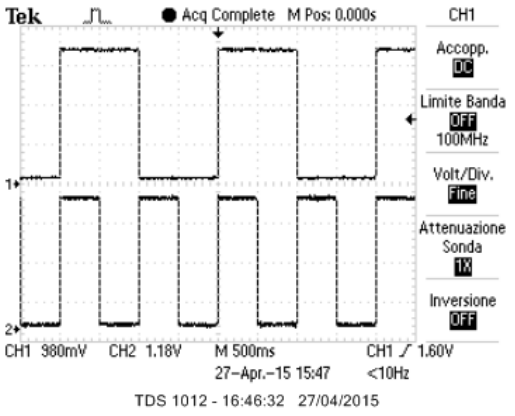
(a) Modalità ABILITATO ($E = 0$): *clock* sul Ch1 e LV sul Ch2.



(b) Modalità ABILITATO ($E = 0$): LG sul Ch1 e LV sul Ch2.



(c) Modalità ABILITATO ($E = 0$): LR sul Ch1 e LV sul Ch2.



(d) Modalità DISABILITATO ($E = 1$): LG sul Ch1 e *clock* sul Ch2.

4.4 Semaforo sincrono

Nella versione del semaforo montata precedentemente la transizione da ABILITATO a DISABILITATO avviene non appena commuta il valore di E, in modo asincrono, senza attendere il successivo colpo di *clock* né completando il ciclo VERDE - VERDE/GIALLO - ROSSO. Si può fare in modo che la transizione avvenga al colpo di *clock* successivo al cambiamento dell'ENABLE, e in particolare si può fare in modo che la transizione avvenga al colpo di *clock* successivo all'accensione del LED rosso. Vediamo come:

- Per fare sì che le transizioni avvengano al colpo di *clock* successivo al cambiamento dell'ENABLE si può semplicemente inserire un D-FF all'uscita dello *switch*. In questo modo, quando cambia E_{in} (così chiameremo l'uscita dell'interruttore, cioè l'entrata del D-FF), $E_{out} = E$ (uscita Q del D-FF) cambierà solo al successivo colpo di *clock*.
- Si può anche fare in modo che la transizione da ABILITATO a DISABILITATO avvenga, se $E = 1$, solo al colpo di *clock* successivo all'accensione del LED rosso. Questa versione è quella che abbiamo realizzato, sempre utilizzando un D-FF, e per trovare la configurazione adatta abbiamo costruito la tavola di verità relativa (tabella 3). Vogliamo costruire un circuito che, ogni volta che $E^n_{in} = 0$, fornisce $E^{n+1}_{out} = 0$

indipendentemente da quali siano i valori di LR e di E^n_{out}[5]. Quando invece $E^n_{in} = 1$ vogliamo che $E^{n+1}_{out} = 1$ solo se già $E^n_{out} = 1$ (cioè ci troviamo già nello stato DISABILITATO) o $LR = 1$. Considerando che in un D-FF $Q = E^{n+1}_{out} = D$, otteniamo $D = E_{in} \cdot (E_{out} + LR)$, implementato con il circuito in figura 5.

Tabella 3: Tavola di verità per il semaforo sincrono.

E^n_{in}	E^n_{out}	LR	E^{n+1}_{out}	D
1	1	1	1	1
1	1	0	1	1
1	0	1	1	1
1	0	0	0	0
0	x	x	0	0

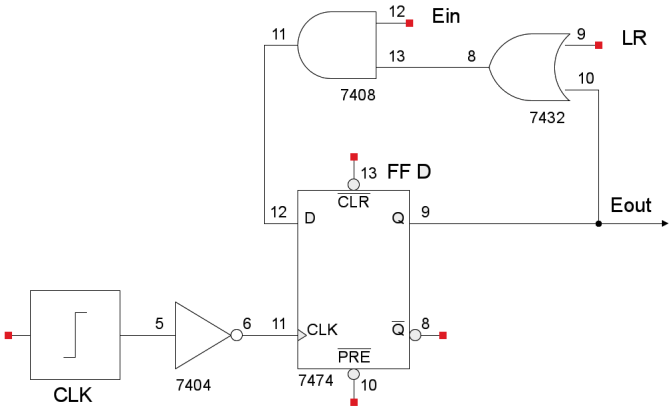


Figura 5: Implentazione della funzione $D = E_{in} \cdot (E_{out} + LR)$ tramite circuito (nel montaggio la porta NOT è implementata con una porta NAND e non con l'integrato 7404 in figura).

Abbiamo quindi provato a montare tale circuito. Possiamo osservare il suo corretto funzionamento in figura 6. La freccia indica il momento in cui E_{in} passa dal valore 0 a 1 attraverso lo SWITCH [6]. Lo SWITCH è stato azionato in un momento in cui era acceso il LED verde, ma il semaforo ha continuato a funzionare in modalità ABILITATO fino al colpo di *clock* successivo all'accensione del LED rosso. A quel punto tutte le luci si sono spente ed è cominciato il lampeggiamento della luce gialla (modalità DISABILITATO), a partire dallo stato col LED giallo spento.

È SBAGLIATO IL CLOCK NELL'IMMAGINE!!!

Possiamo osservare il funzionamento del circuito anche durante l'altra transizione (DISABILITATO → ABILITATO). In questo caso osserviamo che il lampeggiamento del LED giallo si interrompe e il ciclo del semaforo riparte dallo stato GIALLO-ROSSO o dallo stato ROSSO rispettivamente nelle figure 7 (a) e (b)[7].

4.5 Semaforo reale

Abbiamo provato poi a progettare, senza montarlo, un circuito che permetta di far durare i segnali verde e rosso per 10 T_{clk}, e il segnale giallo/verde 1 T_{clk}[8]. Useremo a questo proposito un contatore binario a 4 bit DM74LS93, a cui affiancheremo un comparatore SN74LS85 a cui manderemo agli ingressi B il numero 8[9] in sistema binario ($B_3 = 1$, $B_1 = B_2 = B_0 = 0$), attravero uno SWITCH a 4 ingressi[10]. Useremo il comparatore

[5]Stiamo cioè richiedendo che la transizione dallo stato DISABILITATO a quello ABILITATO avvenga al colpo di *clock* immediatamente successivo alla commutazione del valore di E_{in}.

[6]Per fare questa acquisizione abbiamo usato la funzione d'acquisizione di schermata singola con l'oscilloscopio, impostando il trigger sul fronte in salita dell'ENABLE E_{in}.

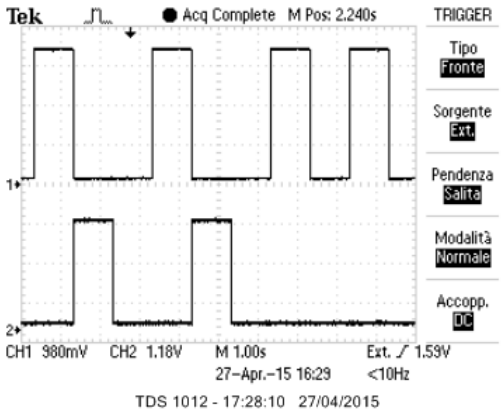
[7]Queste sono le uniche due possibilità che sono state osservate. Il circuito, come già spiegato in precedenza, non ricomincia mai il suo ciclo di funzionamento a partire dallo stato VERDE o VERDE/GIALLO (quando si trova in modalità DISABILITATO vale $D = 0$).

[8]Questo semaforo è progettato per funzionare nella sola modalità ABILITATO.

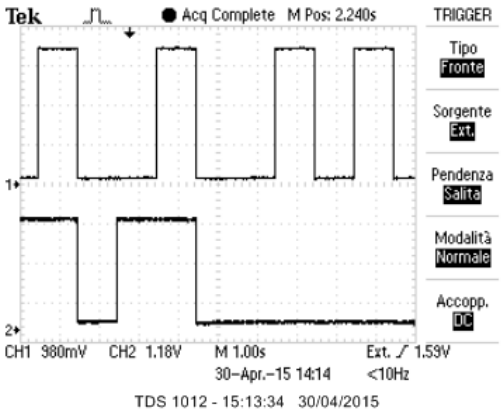
[9]Usiamo il numero 8 perchè il conteggio va da 0 a 8 più un altro colpo di clock dovuto al D-FF che useremo per sincronizzare, come vediamo più avanti.

[10]Per il funzionamento si veda il datasheet. In breve il compratore confronta due segnali binari di 4 bit A e B e restituisce in output il valore HIGH a uno solo dei pin $O_{A>B}$, $O_{A<B}$ e $O_{A=B}$ in base al risultato del confronto e al corretto settaggio dei "cascading inputs" ($I_{A=B} = 1$).

Figura 6: Segnali LV, LR e LG alla transizione ABILITATO → DISABILITATO; l'istante della transizione (commutazione del valore di E_{in}) è individuato dalla freccia in alto.

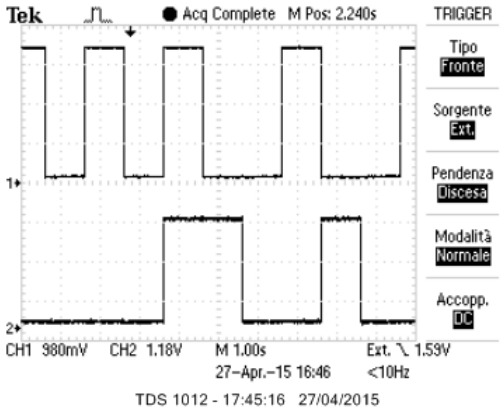


(a) LG su CH1 e LR su CH2.

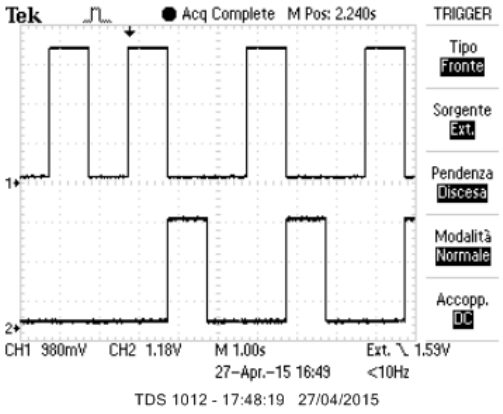


(b) LG su CH1 e LV su CH2.

Figura 7: Segnali LV, LR e LG alla transizione DISABILITATO → ABILITATO nelle due sole combinazioni osservate; l'istante della transizione (commutazione del valore di E_{in}) è individuato dalla freccia in alto.



(a) Esempio 1: LG su CH1 e LR su CH2.



(b) Esempio 2: LG su CH1 e LR su CH2.

sia per il *reset* del contatore (che si ottiene mandando alle entrate R_0 e R_1 del contatore un segnale HIGH) sia per implementare le corrette funzioni logiche da mandare alle entrate J,K e D dei *flip-flop* del circuito originario. Per comodità chiameremo $C(t)$ l'uscita del contatore e $COMP = O_{A=B} + O_{A>B}$ quella del comparatore[11].

Vediamo quindi la tabella di verità (tabella 4) del nostro circuito con tutte le possibili combinazioni di $C(t)$ (e quindi $COMP(t)$), $LV(t)$ e $LG(t)$, i valori desiderati di $C(t+\Delta t)$, $LV(t+\Delta t)$ e $LG(t+\Delta t)$ e i conseguenti valori di JK, D e $RESET$ da fornire per ottenere questi ultimi.

Tabella 4: Tavola di verità per il semaforo reale con stati della macchina e transizioni desiderate. Le scelte arbitrarie sono segnalate con l'asterisco; i valori di C sono riportati in sistema decimale per comodità.

$C(t)$	$COMP(t)$	$LV(t)$	$LG(t)$	$C(t+\Delta t)$	$LV(t+\Delta t)$	$LG(t+\Delta t)$	JK	D	$RESET$
$m < 8$	0	1	0	$m+1$	1	0	0	1	0
$m \geq 8$	1	1	0	0	1	1	1	1	1
x	x	1	1	0	0	0	1	0	1
$m < 8$	0	0	0	$m+1$	0	0	0	0	0
$m \geq 8$	1	0	0	0	1	0	0	1	1
x	x	0	1	*0	*0	*0	1	0	1

Dall'analisi della tabella di verità otteniamo in modo abbastanza semplice:

- $RESET = LG + COMP$
- $JK = LG + COMP \cdot LV$
- $D = (LV + COMP) \cdot \overline{LG}$

Il RESET può inoltre essere sincronizzato con lo stesso metodo visto in precedenza, cioè attraverso l'utilizzo di un D-FF collegato al *clock* del circuito. Per semplicità rappresentiamo l'implementazione del semaforo reale solo nello stato ABILITATO, tralasciando lo stato DISABILITATO e la presenza dell'ENABLE. Il risultato è il circuito progettato in figura 8.

Il circuito costruito in questo modo è versatile per quanto riguarda la durata dei segnali verde e rosso, modificabili cambiando lo stato dello SWITCH. Per simulare il circuito con FALSTAD abbiamo preferito però dedicarci solo al caso in cui il rosso e il verde durano 10 T_{clk}, prendendo come variabile di COMP l'uscita Q_3 del contatore (figura 9). Dalla simulazione abbiamo verificato il corretto funzionamento del circuito.

5. FSM in Software

Si sono scritti due programmi per far gestire ad Arduino Nano un semaforo reale e funzionante nelle due modalità descritte al punto 1. Il primo, dopo la transizione (sincrona) dalla modalità DISABILITATO a quella ABILITATO, fa incominciare il ciclo di funzionamento del semaforo con la luce verde; il secondo, leggermente più complicato[12], con la luce rossa. I codici relativi sono riportati in figura 10.

Si sono collegate le prime tre estremità dei LED verde, rosso e giallo rispettivamente ai *pin* D11, D12 e D13 di Arduino Nano e le seconde estremità a terra, tramite una resistenza da 330 Ω[13]. Inoltre, il *pin* D2 è stato collegato a terra tramite una resistenza da 10 kΩ, e a 5 V tramite lo *switch*. Infine, si è collegato Arduino Nano al computer e si sono caricati i programmi sul microcontrollore, verificando che funzionassero secondo le aspettative.

[11] Che ovviamente implementiamo mandando i due segnali ad una porta OR. Comprendiamo il caso A>B nel nostro circuito per evitare malfunzionamenti.

[12] Rispetto al primo programma si è introdotta una variabile aggiuntiva che descrive lo stato del LED rosso.

[13] In questa sezione dell'esercitazione si sono utilizzate la terra e la sorgente di tensione fornite dai *pin* 4 e 27 del microcontrollore collegato al computer

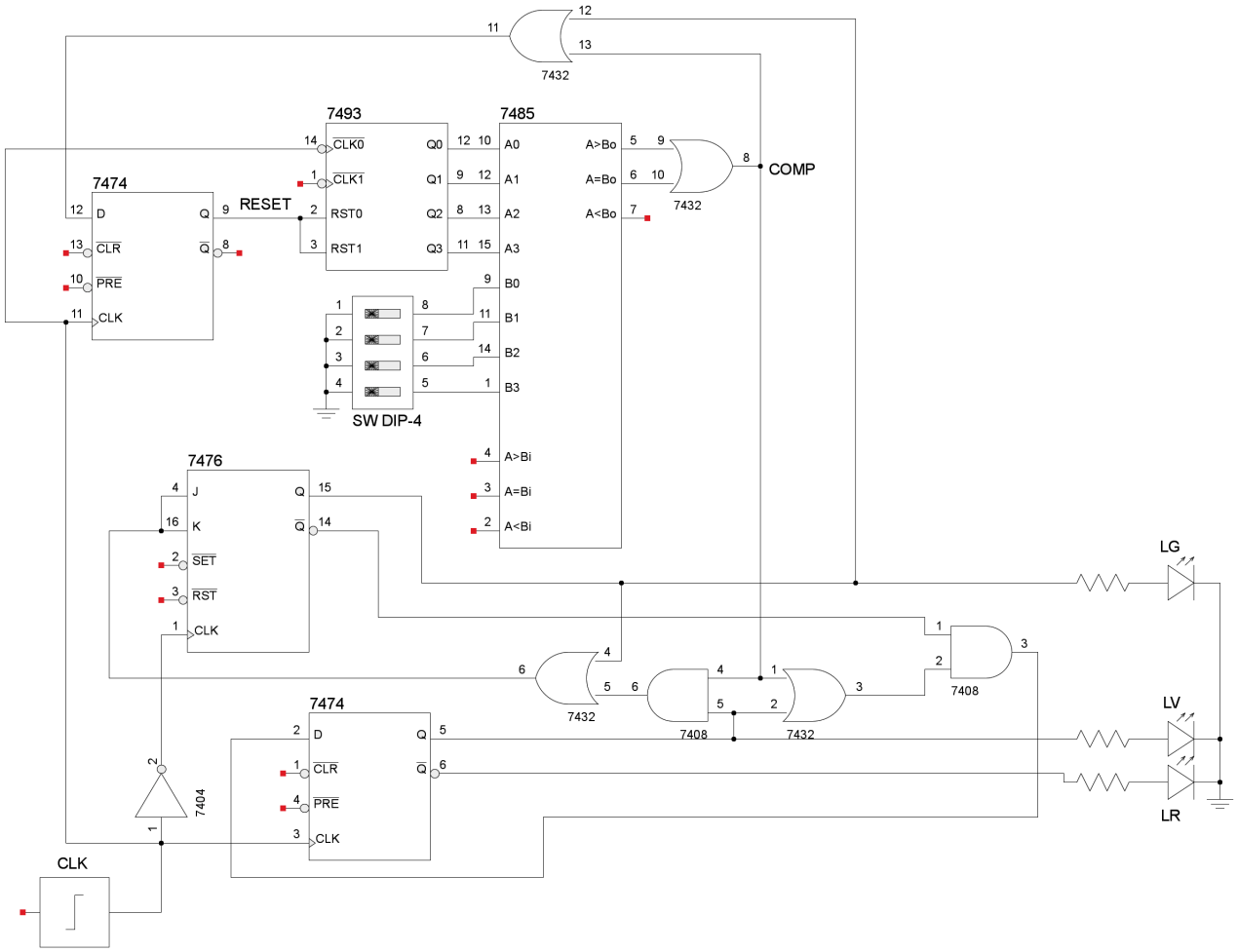


Figura 8: Implentazione del semaforo reale in modalità ABILITATO. Tutte le entrate scollegate sono da considerarsi alte.

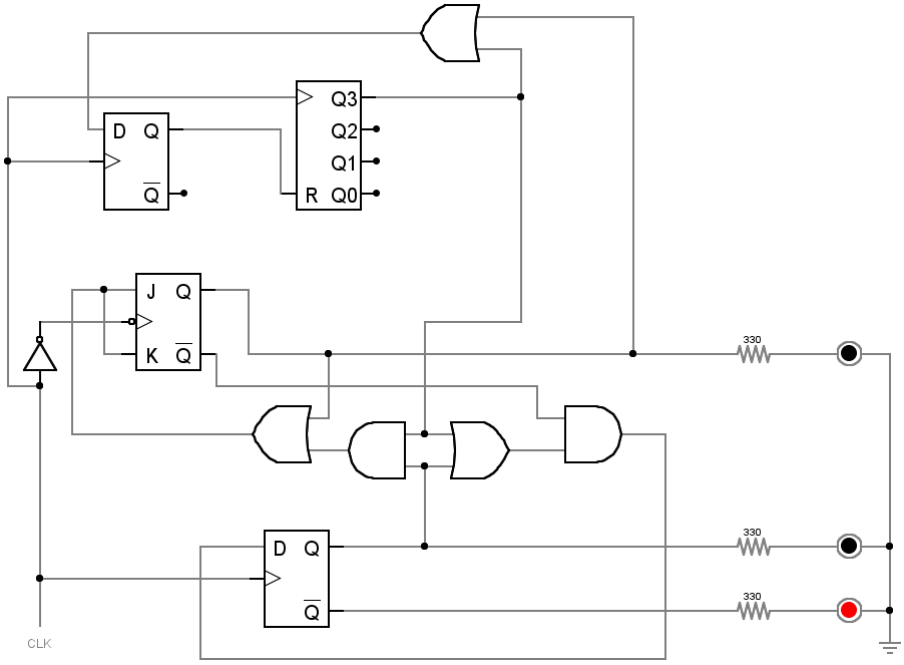


Figura 9: Implentazione del semaforo reale in stato ABILITATO con il simulatore FALSTAD.

Figura 10: Segnali LV, LR, LG e di *clock* per il circuito nelle due modalità di funzionamento.

```
// set pin numbers:
const int enablePin = 2;
const int ledvPin =  11;
const int ledrPin =  12;
const int ledgPin =  13;

// variable declaration
int enableState = 0;

// initialize the pins as inputs/outputs
void setup() {
  pinMode(ledvPin, OUTPUT);
  pinMode(ledrPin, OUTPUT);
  pinMode(ledgPin, OUTPUT);
  pinMode(enablePin, INPUT);
}

void loop(){
  // read the switch state value
  enableState = digitalRead(enablePin);

  // if the traffic light is disabled, yellow LED blinks
  if (enableState == HIGH) {
    digitalWrite(ledgPin, HIGH);
    delay(1000);
    digitalWrite(ledgPin, LOW);
    delay(1000);
  }
  // else, standard behaviour
  else {
    digitalWrite(ledvPin, HIGH);
    delay(10000);
    digitalWrite(ledgPin, HIGH);
    delay(1000);
    digitalWrite(ledrPin, HIGH);
    digitalWrite(ledvPin, LOW);
    digitalWrite(ledgPin, LOW);
    delay(10000);
    digitalWrite(ledrPin, LOW);
  }
}
```

(a) Programma che incomincia il ciclo di funzionamento della modalità ABILITATO con la luce verde.

```
const int enablePin = 2;
const int ledvPin =  11;
const int ledrPin =  12;
const int ledgPin =  13;
int enableState = 0;
int redState = 0;
void setup() {
  pinMode(ledvPin, OUTPUT);
  pinMode(ledrPin, OUTPUT);
  pinMode(ledgPin, OUTPUT);
  pinMode(enablePin, INPUT);
}

void loop(){
  enableState = digitalRead(enablePin);
  // if the traffic light is disabled, yellow LED blinks
  if (enableState == HIGH) {
    digitalWrite(ledrPin, LOW);
    redState = LOW;
    digitalWrite(ledgPin, HIGH);
    delay(1000);
    digitalWrite(ledgPin, LOW);
    delay(1000);}
  else {
    // if precedent state was enabled, normal behaviour
    if (redState == HIGH){
      digitalWrite(ledrPin, LOW);
      redState = LOW;
      digitalWrite(ledvPin, HIGH);
      delay(10000);
      digitalWrite(ledgPin, HIGH);
      delay(1000);
      digitalWrite(ledrPin, HIGH);
      redState = HIGH;
      digitalWrite(ledvPin, LOW);
      digitalWrite(ledgPin, LOW);
      delay(10000);}
    // if precedent state was disabled, make a red cycle
    else {
      redState = HIGH;
      digitalWrite(ledrPin, HIGH);
      delay(10000);}
}}
```

(b) Programma che incomincia il ciclo di funzionamento della modalità ABILITATO con la luce rossa.