PROGETTO 017
CONTROLLO SEMAFORICO

OBIETTIVO

Consegna:

realizzare un circuito digitale in grado di controllare un impianto semaforico posto al centro di un incrocio stradale. All'interno dell'incrocio sono presenti due sensori di traffico Ta e Tb, che rilevano la presenza o meno di auto indicando rispettivamente con gli stati “0” od “1”. Implementare dunque un controllo che preveda in ingresso i due sensori, il clock e un reset e come uscite i due semafori. Progettare la macchina a stati finiti (FSM) tramite metodologia di Mealy o Moore e specificare la codifica degli stati, mappe di Karnaugh ad infine realizzare e simulare il circuito tramite Logisim.

Approfondimento personale:

realizzare un circuito tramite Arduino in grado di emulare il sistema digitale. Gli ingressi collegati ad Arduino saranno due sensori di presenza (simili fotocellule), mentre le uscite saranno due semafori a tre colori (rosso, giallo, verde) realizzati tramite utilizzo di led. Il circuito deve seguire una configurazione di tipo industriale, a pilotare l'impianto sarà Arduino al quale sarà inoltre interfacciato un display Oled di supervisione per mostrare il sinottico del sistema e dare l'opportunità all'utente di regolare tramite pulsanti la tempistica di ogni stato.

DESCRIZIONE

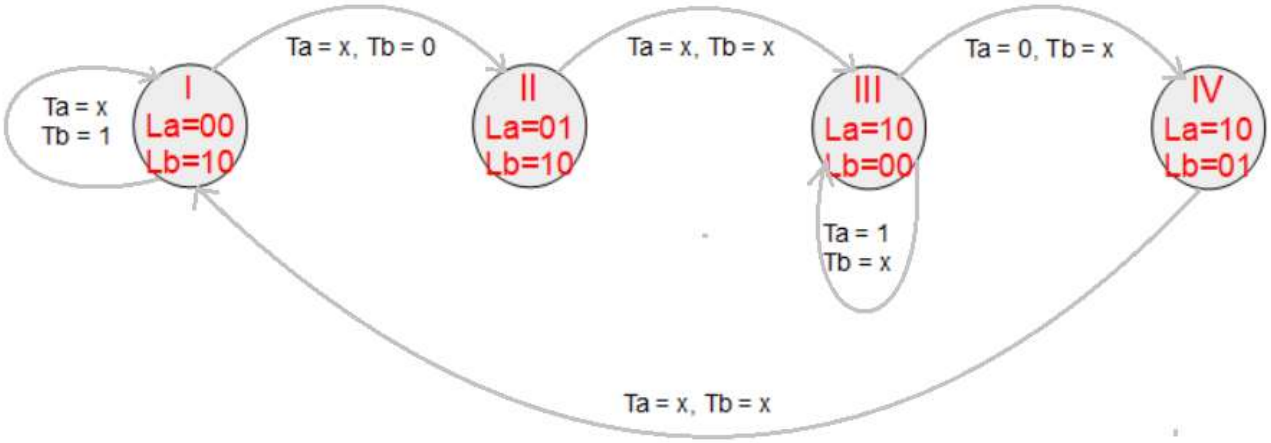
Il sistema si compone di un paio di ingressi denominati Ta e Tb e di due uscite che indicano i due semafori La ed Lb, ognuno avente 3 uscite in base alle colorazioni delle 3 luci.
Per comodità ed in termini di espansibilità, le uscite presenti nella FSM saranno 4 relative agli stati del sistema, attivando pertanto una linea su quattro in base allo stato.
Il numero totale degli stati come detto è 4, sono dunque necessari solo 2 bit per riuscire a rappresentare le diverse combinazioni.
La soluzione proposta nel corso della trattazione subirà una variazione per questioni di ottimizzazione, aggiungendo pertanto ingressi ed uscite non previste nella consegna che verranno approfondite in seguito.
L'automa a cui si fa riferimento è quello di Moore, ad ogni stato sono dunque associati i valori delle uscite in quel determinato momento e la transizione da uno stato all'altro avviene tramite le frecce che indicano tutte le combinazioni possibili degli ingressi del sistema.

Una volta ricavato il diagramma degli stati, si passerà alla tabella di transizione degli stati e tabella delle uscite con le relative mappe di Karnaugh.
Le espressioni ottenute serviranno per comandare i flip-flop utili a memorizzare lo stato dei singoli bit ed occuparsi pertanto di aggiornare, sul fronte di salita del clock, lo stato generale del sistema.
Tramite Logisim sarà poi realizzato il circuito di controllo che andrà a pilotare i due semafori con le rispettive luci per il controllo del traffico.

SVILUPPO

Diagramma degli Stati:

I	II	III	IV
00	**01**	**10**	**11**
La = 00 = G	La = 01 = O	La = 10 = R	La = 10 = R
Lb = 10 = R	Lb = 10 = R	Lb = 00 = G	La = 01 = O



Codifica:
il sistema per controllare entrambi i semafori necessita l'utilizzo di 4 stati, quindi 2 bit (2^2, stati massimi >= 4, stati necessari). Lo stato delle uscite è assunto come la colorazione dei semafori allo stato attuale del sistema, 00 indica verde (green), 01 arancione (orange), 10 rosso (red), 11 stato non considerato. Alla base del ragionamento di funzionamento, in breve, via è la concezione per cui un sensore comandi il semaforo opposto. In generale un semaforo resta verde se il sensore opposto non rileva presenza di traffico, al contrario verrà effettuata una chiamata per permettere all'altro semaforo di diventare verde. Il sistema funziona in maniera uguale per ogni semaforo e sensore opposto. Questo metodo di programmazione, sarà più chiaro dopo, porta ad una condizione di loop in caso di traffico.

Stato I, codice 00:
- Ta = x, Tb = 1, il sistema rimane invariato poiché il sensore Tb non rileva auto ferme nel senso opposto di marcia;
- Ta = x, Tb = 0, il sistema si porta nello stato successivo II, il sensore Tb risulta occupato, quindi delle auto sono ferme aspettando che il loro semaforo Lb diventi verde.
- La = G, Lb = R, il semaforo La è verde mentre, Lb rimane rosso.

Stato II, codice 01:
- Ta = x, Tb = x, il sistema qualunque sia la combinazione degli ingressi passa al prossimo stato III, durando solo un ciclo di clock.
- La = O, Lb = R, il semaforo La è arancione perché è stata fatta una richiesta da parte del sensore Tb lo stato precedente, Lb rimane ancora rosso.

Stato III, codice 10:
- Ta = 1, Tb = x, il sistema rimane invariato poiché il sensore Ta non rileva auto ferme nel senso opposto di marcia;
- Ta = 0, Tb = x, il sistema si porta nello stato successivo IV, il sensore Ta risulta occupato, quindi delle auto sono ferme aspettando che il loro semaforo La diventi verde.
- La = R, Lb = G, il semaforo La è rosso, mentre Lb rimane verde.

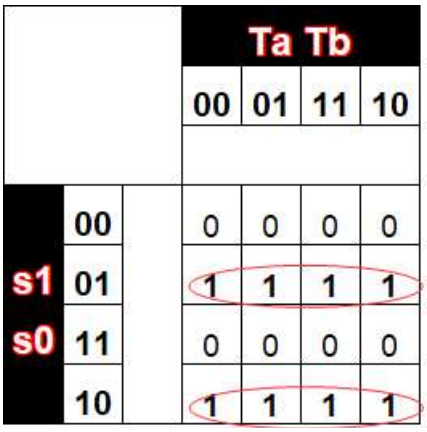
Stato IV, codice 11:
- Ta = x, Tb = x, il sistema qualunque sia la combinazione degli ingressi ritorna allo stato d'origine I, durando solo un ciclo di clock.
- La = R, Lb = O, il semaforo La rimane ancora rosso, mentre Lb è arancione perché è stata fatta una richiesta da parte del sensore Ta lo stato precedente.

Tabella transizione degli stati:

s	s1	s0	Ta	Tb	S	S0	S1
I	0	0	0	0	II	0	1
I	0	0	0	1	I	0	0
I	0	0	1	0	II	0	1
I	0	0	1	1	I	0	0
II	0	1	0	0	III	1	0
II	0	1	0	1	III	1	0
II	0	1	1	0	III	1	0
II	0	1	1	1	III	1	0
III	1	0	0	0	IV	1	1
III	1	0	0	1	IV	1	1
III	1	0	1	0	III	1	0
III	1	0	1	1	III	1	0
IV	1	1	0	0	I	0	0
IV	1	1	0	1	I	0	0
IV	1	1	1	0	I	0	0
IV	1	1	1	1	I	0	0

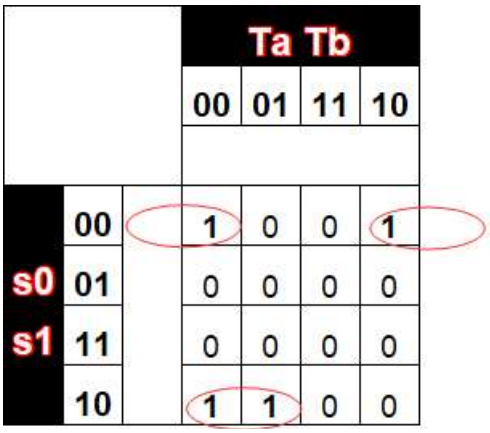
Input:
come ingressi sono considerati gli stati attuali s1, s0 dei 3 flip-flop D e il valore di Ta e Tb.
Output:
come uscite sono considerate gli stati futuri S1, S0 dei 3 flip-flop D.

Mappe di Karnaugh:
sintesi minima degli stati tramite forma SP degli stati futuri.



		Ta Tb			
		00	01	11	10
s1 s0	00	0	0	0	0
	01	1	1	1	1
	11	0	0	0	0
	10	1	1	1	1

S1: $\overline{s1}\ s0 + s1\ \overline{s0}$



		Ta Tb			
		00	01	11	10
s0 s1	00	1	0	0	1
	01	0	0	0	0
	11	0	0	0	0
	10	1	1	0	0

S1: $\overline{s1}\ \overline{s0}\ \overline{Tb} + s1\ \overline{s0}\ \overline{Ta}$

Tabella delle uscite:
sintesi minima delle uscite. Come detto in precedenza le uscite del sistema sono riportate come la codifica dello stato attuale, tuttavia le espressioni legate alla colorazione dei semafori rimangono le stesse, sia fatte all'interno della FSM, sia all'esterno.

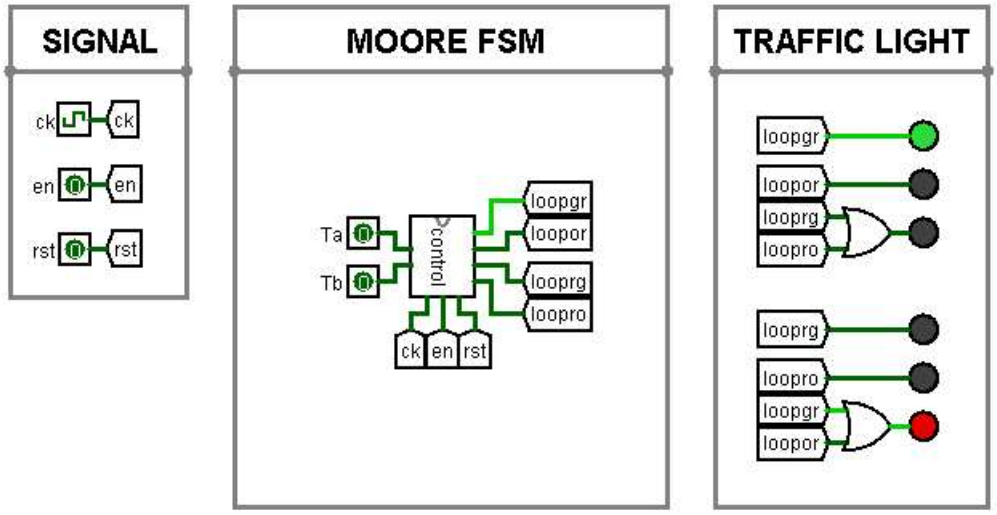
s	s1	s0	La	Lb
I	0	0	**00**	**10**
II	0	1	**01**	**10**
III	1	0	**10**	**00**
IV	1	1	**10**	**01**

La G: $\overline{s1}\ \overline{s0}$
La O: $\overline{s1}\ s0$
La R: $s1\ \overline{s0}$
La R: $s1\ s0$

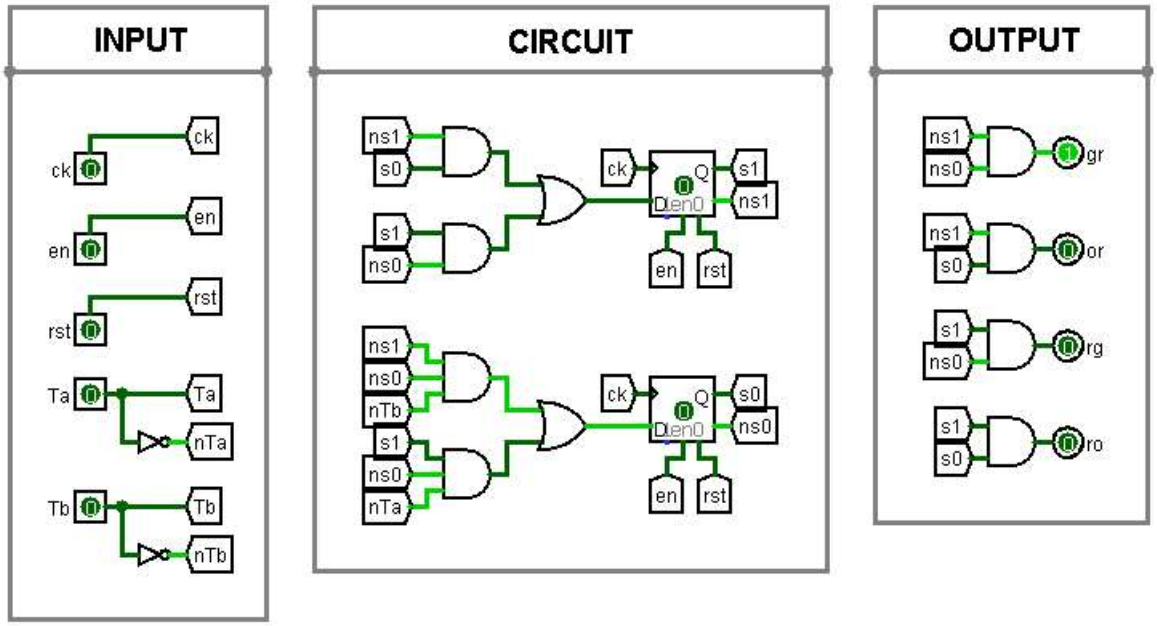
Lb R: $\overline{s1}\ \overline{s0}$
Lb R: $\overline{s1}\ s0$
Lb G: $s1\ \overline{s0}$
Lb O: $s1\ s0$

REALIZZAZIONE

Il circuito realizzato attraverso Logisim si compone di 2 circuiti principali: main, ctrl. Mentre il main rimane il circuito madre, l'altro è un sottocircuito sviluppato per rendere maggiormente ordinato e chiaro il funzionamento del sistema.



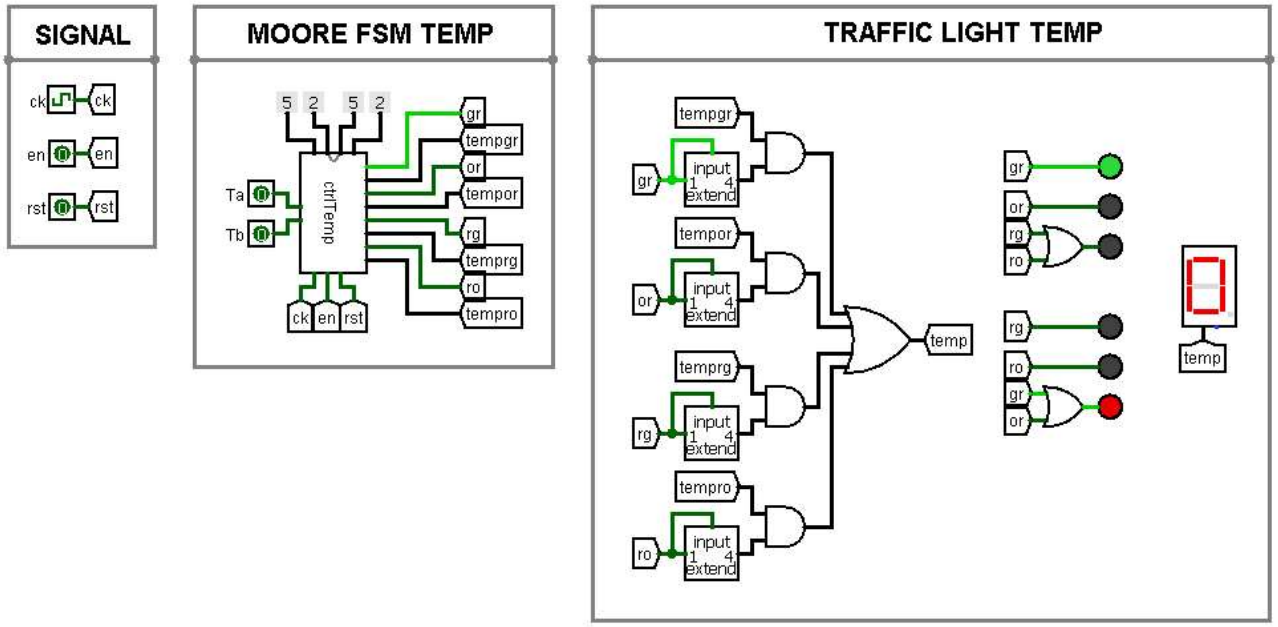
Main:
identificato come il circuito principale di ogni progetto Logisim, il main si occupa di gestire l'interfacciamento tra i segnali di input generali come il clock (per sincronizzare tutti gli elementi del sistema), enable (abilitazione all'aggiornamento dei vari componenti) e reset (azzeramento del sistema), con le uscite dal sottocircuito control (Moore FSM).
Dai segnali del circuito importato control è possibile accendere i led seguendo la tabella relativa alle uscite. Come in precedenza riportato, il set dei led è indifferente se fatto all'interno del control o del main. In questo modo, si vedrà dopo come sia più chiaro gestire il sistema in una seconda configurazione.



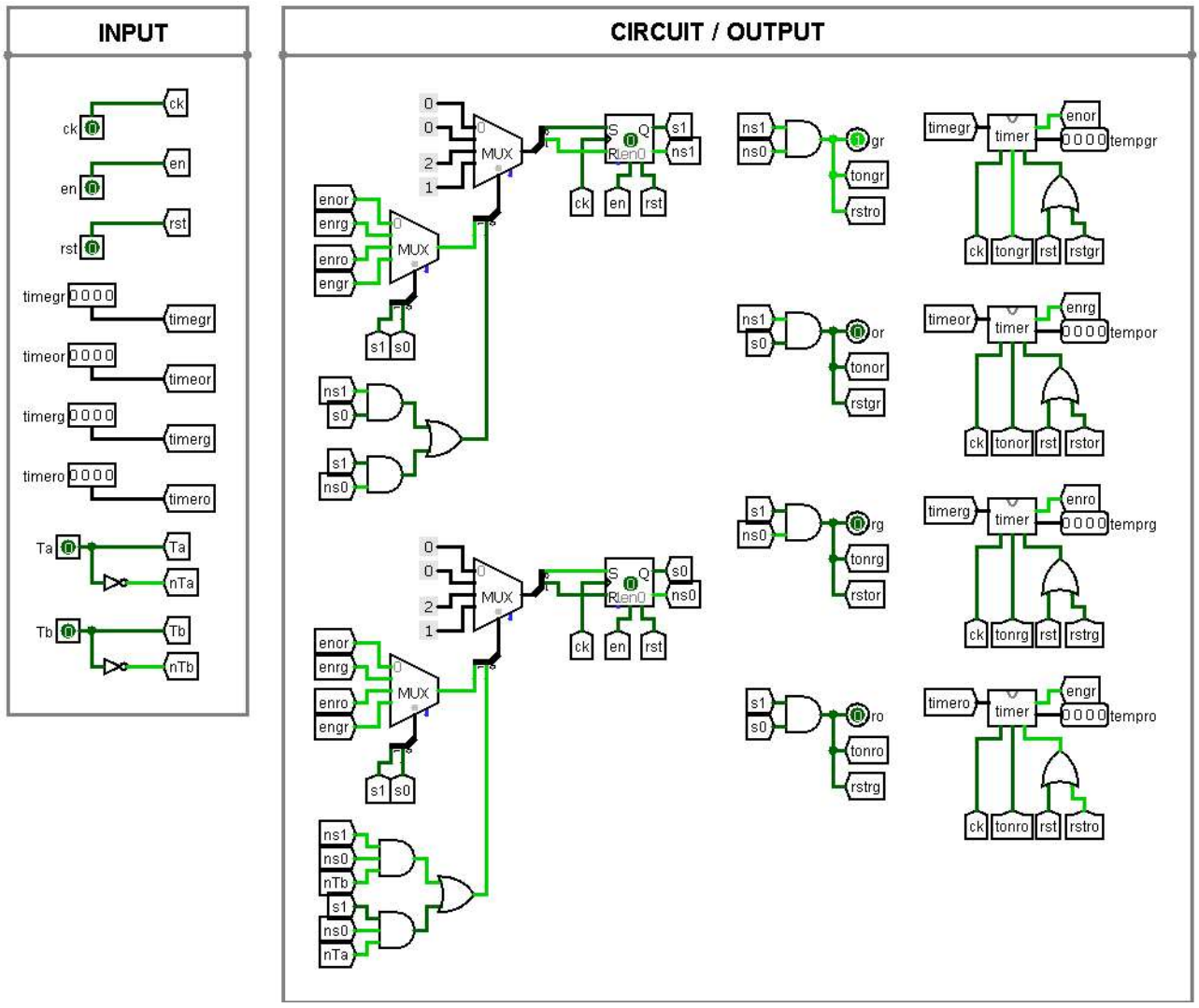
ctrl:
identificato come il circuito di controllo del sistema, questo sottocircuito implementa la macchina stati finiti secondo Moore sviluppata in precedenza.
I segnali di input sono il clock, enable e reset, ai quali solo si aggiungono i valori degli input Ta e Tb. Il circuito replica, tramite l'utilizzo di porte logiche (eventuali not, n and e 1 or generali, nel caso della forma SP) la semplificazione delle espressioni ottenute dalla mappe di Karnaugh. I segnali sono infatti negati con una not, mandati alle n and, in base a al numero di raggruppamenti presi, ed infine sommati tramite una sola or. I 2 risultati finali, uno per ogni mappa prodotta, fanno da ingresso ad un flip-flop D, che solo sul fronte positivo del segnale di clock, commuta il suo stato secondo la lettura dell'ingresso e del suo stato attuale, aggiornando pertanto il valore codificato per lo stato del sistema.
Le uscite dei flip-flop D sono riportate sugli ingressi in maniera di avere un sistema retroattivo, ovvero come appena detto, lo stato futuro non dipende solo dal valore degli ingressi ma anche dallo stato attuale in cui il sistema si trova. Le uscite dei flip-flop D pilotano poi, secondo le espressioni trovate in precedenza, le uscite dell'intero sistema, che individuano infatti lo stato attuale in cui il sistema si trova partendo dalla sua codifica binaria.

OTTIMIZZAZIONI

Come anticipato, il sistema presenta un problema notevole che non è stato finora consapevolmente trattato. La questione sta nel fatto che nella combinazione di ingressi Ta = 0 e Tb = 0, si nota come il sistema ad ogni fronte del clock passa ad uno stato successivo, portando il sistema in una fase di loop. In pratica questo vuol dire che in una condizione di ipotetico traffico lo stato delle uscite oscillerebbe in continuazione da uno stato all'altro, passando a tempo di clock tutti gli stati uno alla volta. Impensabile dunque riuscire a pilotare il traffico in questa maniera, sistema inefficace ed insicuro. Una soluzione potrebbe essere quella di imporre un sensore di master ed uno di slave, in questa maniera sarebbe il master a scandire le varie fasi del sistema, portando però l'altro sensore ad una posizione marginale nella gestione ed ottenendo un sistema in un certo senso “imparziale” nella decisione degli stati. La soluzione per cui si è optato è dunque una decisione poco più complessa ma che mantiene la dinamica decisiva utilizzata nel circuito di controllo della FSM realizzata antecedentemente. La risoluzione del progetto sta di fatto nell'appoggiarsi alla FSM prima realizzata ma trattandola sotto un punto di vista più pragmatico e professionale, inserendo pertanto dei temporizzatori a scandire i vari stati del sistema. In pratica ogni volta che il sistema cambia stato attiva un determinato temporizzatore che darà l'abilitazione a passare al prossimo stato solo se, oltre agli ingressi nella giusta combinazione, questo avrà superato il tempo di mantenimento dello stato attuale. In questa maniera il traffico verrà gestito secondo la FSM realizzata ma con l'apporto concreto di timer che evitano di cadere in condizioni di loop; o meglio, sebbene in realtà il ciclo infinito possa esistere, a scandirlo saranno i temporizzatori assegnati.

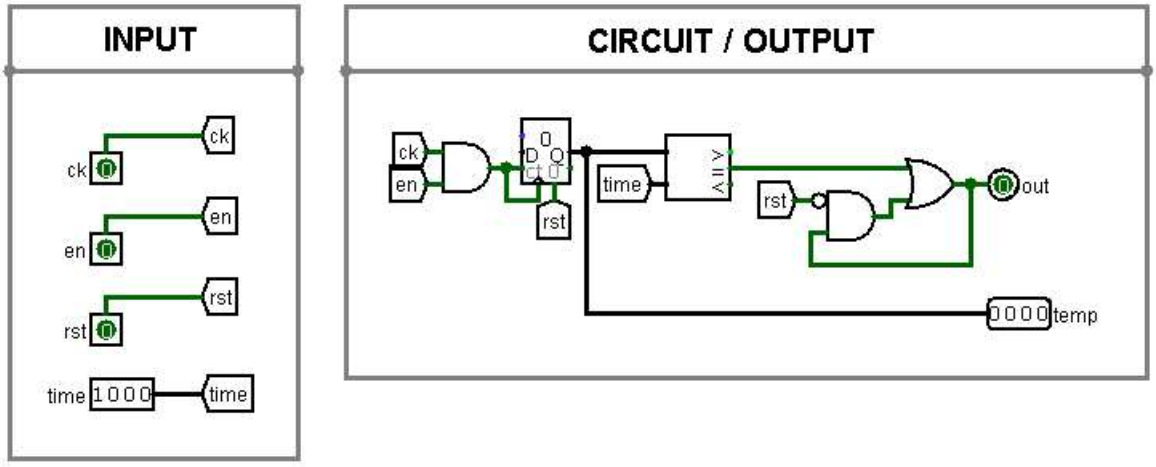


main:
il main si compone del nuovo sottocircuito di controllo temporizzato e come accennato prima molti più ingressi ed uscite. Tra gli ingressi oltre ai sensori si trovano delle costanti che saranno i cicli di clock da aspettare perché ogni stato posso dare l'abilitazione alla commutazione allo stato successivo. Le uscite sono raddoppiate, sebbene servano solamente le uscite gr, or, rg, ed ro, le uscite accessorie indicano lo stato del conteggio parziale relativo allo stato attuale, a dimostrazione della veridicità del sistema e come supporto all'utente. Tali uscite sono poi riportate ad un circuito combinatorio che, tramite estensori settati sull'enable di ingresso, porte AND ed una OR, mostra il conteggio parziale dello stato attuale del sistema. La gestione dell'accensione dei led rimane la stessa di prima.

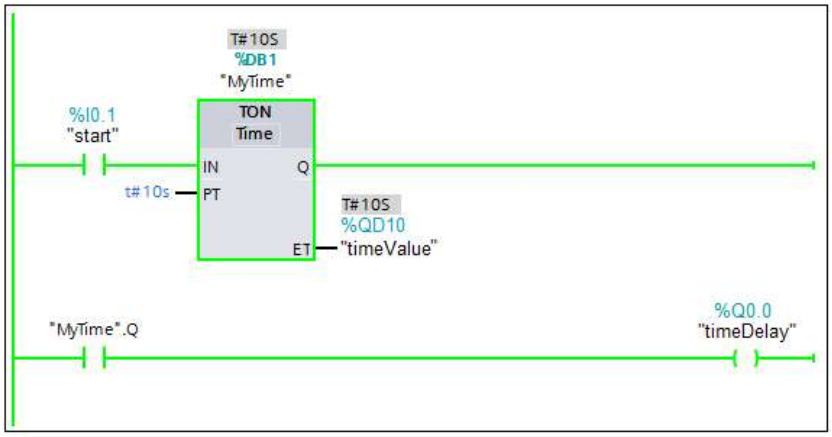


ctrlTemp:
il nuovo circuito di controllo implementato come FSM di Moore presenta dunque l'inserzione dei temporizzatori ed un'altra notevole differenza di funzionamento, ovvero l'utilizzo di fli-flop RS e non flip-flop D. La scelta si basa sulla considerazione per cui, fino a quando i temporizzatori non raggiungono il loro conteggio prestabilito attivando l'abilitazione del passaggio allo stato successivo, i flip-flop non devono aggiornare il loro stato, restando per cui, in altre parole, in una condizione di memoria. I flip-flop di tipo D non si adeguano a questo compito poiché, nel caso i temporizzatori andassero su una porta AND con il segnale derivante dall'OR corrispettiva al flip-flop, il segnale sarebbe ancora un valore o alto o basso per cui al fronte successivo del clock questo farebbe commutare il flip-flop D allo stato 0 poiché non ancora arrivata l'abilitazione del temporizzatore.
Una diversa opzione efficace ma meno elegante ed affidabile sarebbe quella di disabilitare il clock tramite il temporizzatore con una porta AND. In questo caso dunque lo stato commuterebbe solo al raggiungimento del tempo prestabilito ma tutto il sistema sarebbe fermo durante il conteggio poiché sarebbero come disabilitati i flip-flop D dato che non avrebbero possibilità di aggiornare il loro stato.
La soluzione per cui è quella di ricorrere ad un flip-flop di tipo SR. Questo garantisce infatti una condizione (S = 0, R = 0) di memoria del flip-flop, in linea dunque con la soluzione del problema. A questo punto bisogna solo occuparsi di mandare ai segnali di S ed R dei flip-flop i giusti valori. Questo vuol dire che fino a quando il temporizzatore relativo allo stato starà contando, il flip-flop dovrà essere in configurazione di memoria ovvero con tutti gli

ingressi a 0. Una volta finito il conteggio bisognerà portare l'uscita in fase di set o reset in base al segnale presente sulla OR. Per risolvere questo quesito ci si appoggia ud un multiplexer in cui i segnali di selezione saranno i bit del temporizzatore (MSB) e dell'OR corrispettiva al flip-flop SR (LSB). Mettendo all'ingresso del multiplexer le costanti 0, 0, 2, 1 il risultato sarà appunto che al codice 00 e 01 corrispondono degli 0 e quindi il flip-flop resterà in memoria, al codice 10 si avrà il comportamento analogo ad un reset e come ultimo il codice 01 setterà il flip-flop.
I bit dei temporizzatori infine si riducono ai diversi segnali di abilitazione e conteggio parziale, oltre ad ingressi di abilitazione ed uscite di fine conteggio. L'unica accortezza nella loro configurazione, sta nel fatto di abilitarli al giusto stato e “disabilitarli” oltre che resettarli al raggiungimento dello stato successivo. Senza questo accorgimento i temporizzatori conterebbero solo una volta e non venendo resettati avrebbero sempre la loro uscita settata alta. In altre parole, il sistema avrebbe un funzionamento corretto solo per il primo giro, dopodiché avrebbe lo stesso difetto di quello in precedenza trattato poiché avrebbe sempre l'abilitazione a commutare stato.
Un problema che viene inoltre evitato con l'inserzione di flip-flop SR è quella di una condizione di reset del sistema. In particolare è facile vedere come tutti i temporizzatori abbiano un segnale di abilitazione ed uno di conteggio raggiunto, e come questi segnali siano collegati insieme formando una catena di abilitazione tra i vari temporizzatori. Il sistema riesce comunque in un primo momento ad entrare in questa catena a prendere lo “spunto iniziale” per partire con l'abilitazione del primo e procedere con i restanti di conseguenza. Questo pregio è dovuto alla configurazione dei vari flip-flop e dal fatto che grazie alla condizione di memoria e alle uscite sempre presenti sui multiplexer i flip-flop abbiano sempre uno stato di reset iniziale con il codice impostato sullo stato 00.

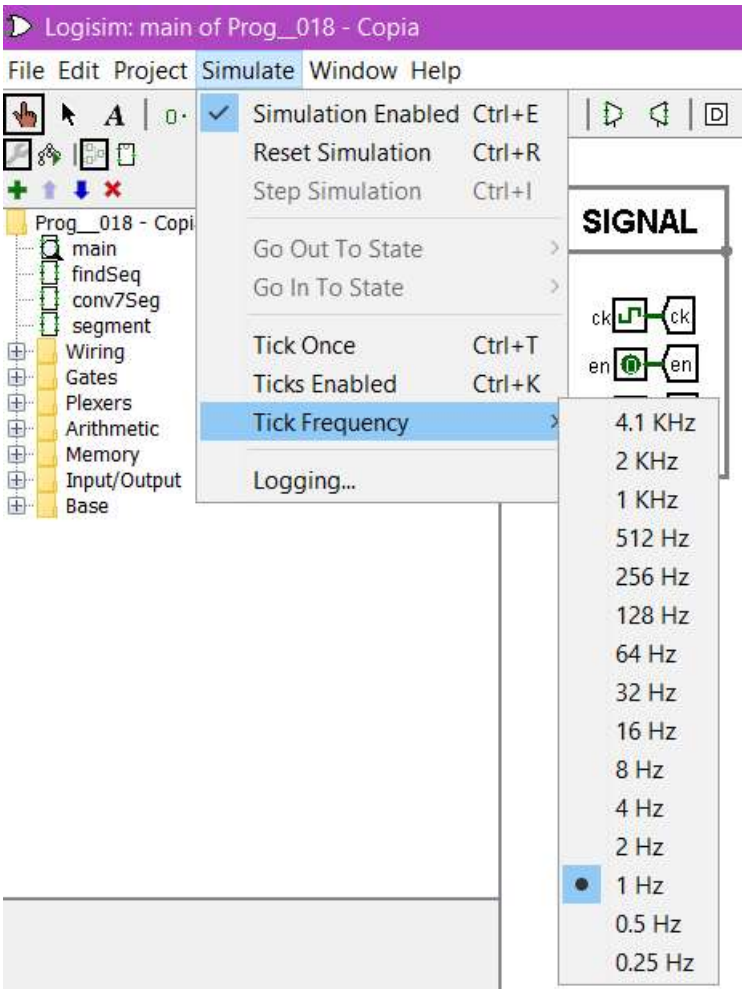


Timer:
il sottocircuito dedito alla temporizzazione del sistema, si basa su un contatore ed il confronto del valore con la soglia di attivazione. In diversi campi dell'automazione e dell'elettronica per svolgere il compito di temporizzazione si effettua un conteggio, in base al conteggio e la frequenza del clock è dunque possibile dire quanti millisecondi o secondi sono passati, dovendosi solo “ricordare l'istante di inizio”. Il principio usato nel timer Logisim è il medesimo, supponendo di partire da zero con il conteggio ed inserendo (in questo caso come metodo di prova) un clock con frequenza f = 1Hz quindi con periodo T = 1s, è possibile abilitare l'uscita dopo n secondi, quindi dopo n cicli di clock. Con questo semplice conteggio e confronto tramite comparatore è possibile pertanto ottenere un temporizzatore.

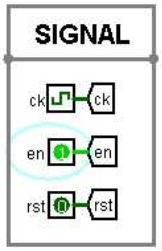


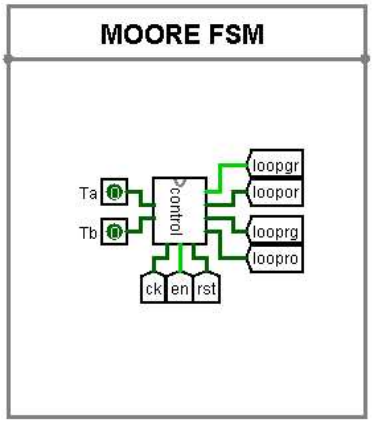
Il temporizzatore ottenuto nel progetto Logisim è ispirato al classico temporizzatore TON inserito nell'ambiente di programmazione TIA Portal per programmazione PLC Siemens. Come il TON Siemens, anche il timer Logisim una volta raggiunto il conteggio massimo mantiene l'uscita alta (grazie ad un sistema di retroazione) fino ad una data condizione di reset, riportando inoltre in uscita, oltre all'abilitazione anche il tempo parziale trascorso ed essendo completamente modificabile il preset time dall'esterno.

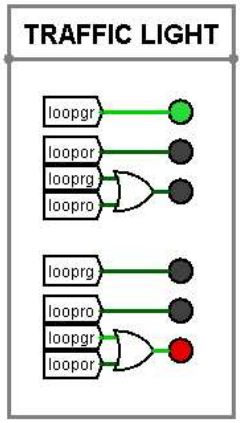
TEST

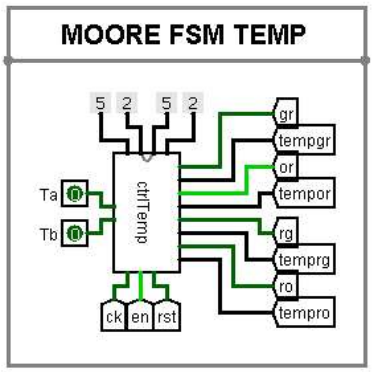


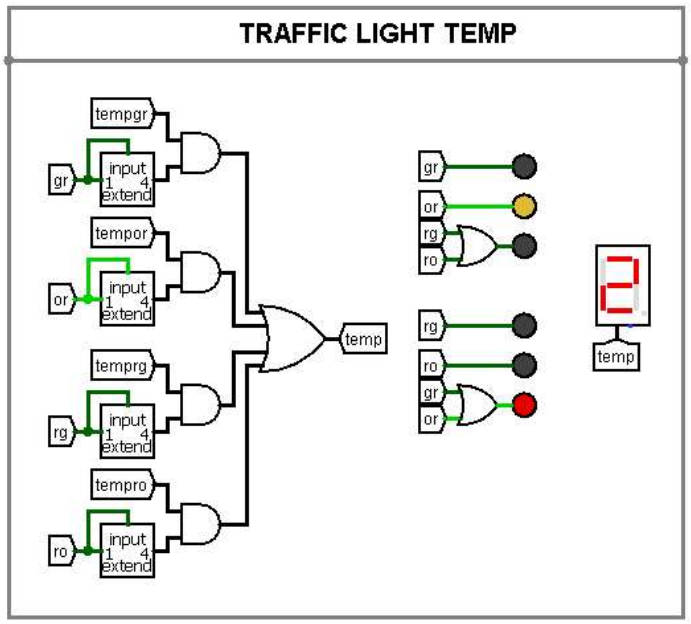
Impostazioni Logisim:
per realizzare un test pratico del circuito con Logisim bisogna portarsi in fase di simulazione tramite la voce Simulate → Simulation Enabled.
Potrebbe essere utile anche settare il clock sempre attivo in maniera di non doversi occupare anche del clock, Simulate → Ticks Enabled, e definire una frequenza, Simulate → Tick Frequency → xxx.x Hz. Nel caso specifico il clock per comodità e frequenza base viene settato ad f = 1Hz.
Da notare inoltre che l'unico clock inserito all'interno del progetto è quello del circuito main, tutti gli altri sottocircuiti infatti, assumendo il clock come ingresso, presuppongono il fatto che questo sia comandato dall'esterno. Sarebbe considerato come errore inserire diversi clock nel progetto, solo uno comune e generale deve gestire e dunque coordinare tutti gli elementi del circuito Logisim.











Simulazione:
In fase di esecuzione il circuito risulterà circa in queste condizioni, le linee evidenziate indicano quali bit sono ad 1, viceversa, a 0 o anche per canali di n bit, saranno di colore verde scuro. Le varie label facilitano notevolmente la lettura e chiarezza del circuito, nonostante anche queste non assumano alcuna colorazione in base allo stato del dato. Da notare come i due sistemi, seppure di basino sulla stessa valutazione per quanto riguarda la FSM, assumano stati diversi in proprio in relazione alla diverse configurazione dei flip-flop. Le costanti 5 2 5 2 in ingresso al circuito ctrlTemp indicano i secondi da aspettare per la commutazione dello stato, mentre il display segno il conteggio parziale.

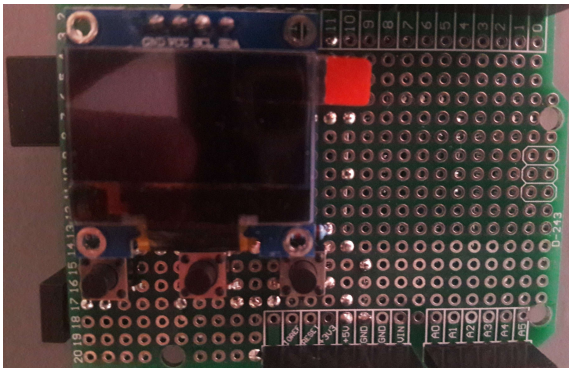
APPRONFONDIMENTO PERSONALE

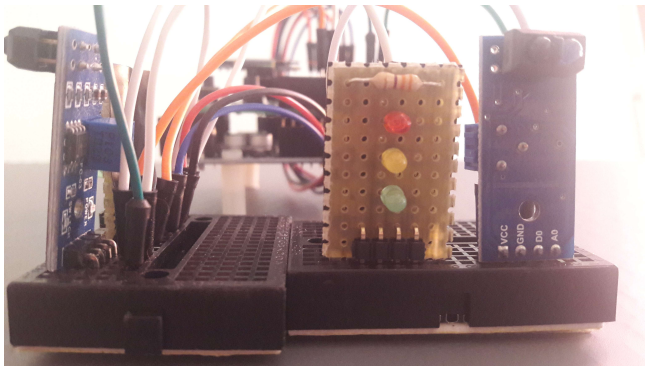
Descrizione:

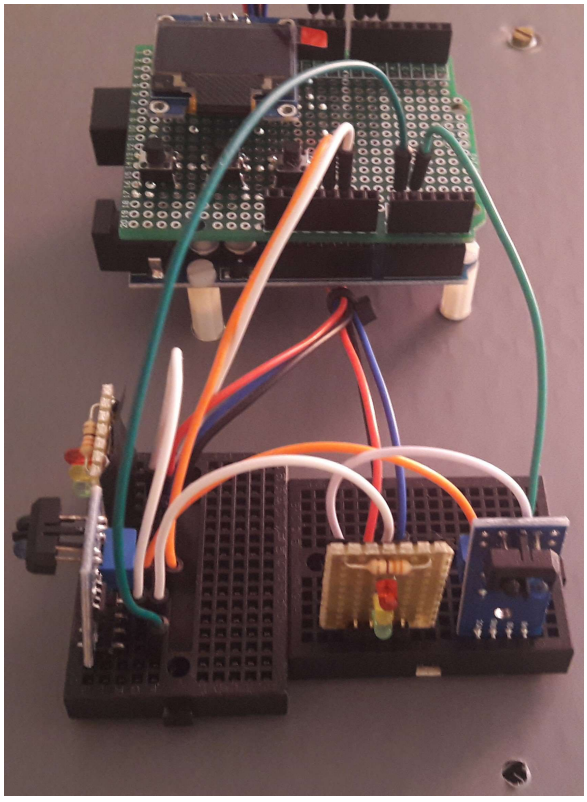
l'approfondimento personale in questione è realizzato tramite Arduino ed una shield personalizzata, connesso ad un display Oled.
Mentre Arduino si occupa della parte hardware del progetto, rilevazione ingressi e gestione ciclo, il display cura invece la parte di interfaccia e sinottico permettendo una facile modifica di alcuni parametri.

Shield Oled interface:

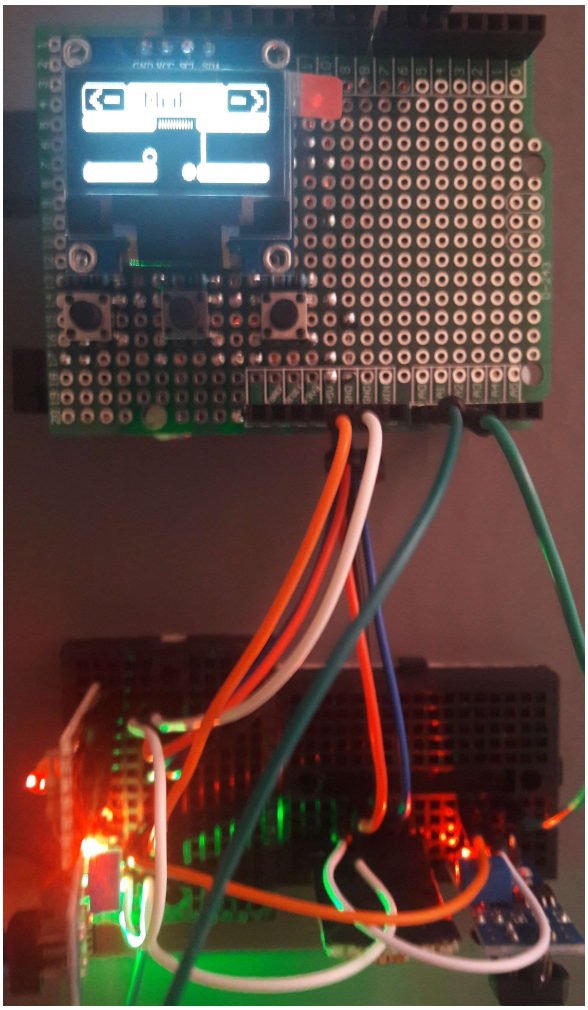
la shield e i semafori sono stati schematizzati e costruiti personalmente, avvalendosi di ua schedina conforme alla distanza tra le varie serie di pin di Arduino e di piastrine mille-fori adeguatamente tagliate. La shield è doppia faccia, tuttavia solo il piano inferiore è risultato utile per lo sbroglio delle piste. I pulsanti sono stati messi inseriti in maniera da formare un'interfaccia monitor insieme al display. Per interfacciare i sensori ed i semafori, per motivi di posto e modularità, sono state utilizzate 2 breadboard in maniera inoltre da simulare un incrocio stradale come conformazione.







Controllo semaforico:
il display presenta la pagina di sinottico con stato aggiornato dei sensori.



CONCLUSIONI

Il progetto approfondito è risultato davvero interessante poiché è stato spunto per molte implementazioni accessorie.
L'inserzione dei temporizzatori rende il sistema diverso per alcune caratteristiche, sebbene non alterando la concezione del controllo semaforico. Il progetto, con l'utilizzo di timer, assume un punto di vista diverso e modifica in alcuni parti il circuito, proponendo una versione molto diversa da quella affrontate durante il corso. Malgrado queste diversità, la condizione di loop originale non è mai stata presa come un problema, bensì una condizione programmata e corretta per l'implementazione del sistema. I temporizzatori non sono solo utili a risolvere la questione, ma necessari per realizzare un sistema collaudato e concreto, ovvero la metodologia di sviluppo non è dipendente dalla piattaforma o ambiente utilizzato, è una metodologia trasportabile ed applicabile dal campo elettronico/informatico a quello professionale/industriale.
A livello di programmazione ed approfondimento il progetto si è rivelato interessante per metodologia di programmazione. In campo industriale spesso è utilizzata la programmazione a passi, ma spesso presenta diversi difetti in alcune fasi produttive (fermi macchina imprevisti ad esempio). A questa metodologia si contrappone invece una

programmazione a condizioni, in cui non esistono “cicli” e tutto è fatto in base alla lettura degli ingressi; per contro il programma risulta molto più elaborato e complesso oltre che dispendioso in termini di tempo. Il progetto realizzato, si adegua molto bene al primo tipo di programmazione ed è infatti stato utile capire come procedere in questo ambito e poter applicare questo tipo di concezione.
Ultimo punto sta nel notare come da un progetto apparentemente ridotto si possa innescare un “legame” tra diversi ambiti, elettronica, informatica ed automazione. Il sistema realizzato con Arduino è realizzato sulla base del progetto e a questo aggiunge una trattazione a livello industriale proponendo uno sviluppo incentrato sulla tipologia PLC (Arduino) – HMI (display Oled), il tutto controllato con un linguaggio di programmazione che prende molto spunto dall'automazione e le basi teoriche studiate a lezione.