



PROGETTO 018 RICONOSCITORE SEQUENZE



OBIETTIVO

Consegna:

realizzare un circuito digitale in grado di riconoscere le sequenze binarie A = "1101" e B = "1110". Implementare successivamente, per ogni sequenza, due contatori collegati ad un display 7 segmenti, in maniera tale da mostrare il numero di quante sequenze sono state riconosciute. 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 e un applicativo Processing in grado di emulare il sistema digitale. Gli ingressi collegati ad Arduino saranno due pulsanti. Alla loro pressione invieranno tramite comunicazione seriale, rispettivamente uno "0" o "1". Tramite Processing creare un applicativo in grado di leggere i dati ricevuti da Arduino dal monitor seriale, confrontarli con 2 sequenze di stringhe A e B e contare quante volte sono state riconosciute tali sequenze. Mostrare inoltre, tramite un grafico, gli ultimi stati ricevuti.

DESCRIZIONE

Il sistema si compone di un solo ingresso denominato In e di due uscite A e B che indicano rispettivamente il completamento delle sequenze A = "1101" e B = "1110".

Il numero totale degli stati è 7, sono dunque necessari 3 bit per riuscire a rappresentare le diverse combinazioni. L'ultimo stato non verrà considerato, pertanto sarà solo utilizzato in caso di comodità nella semplificazione delle forme SP.

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.

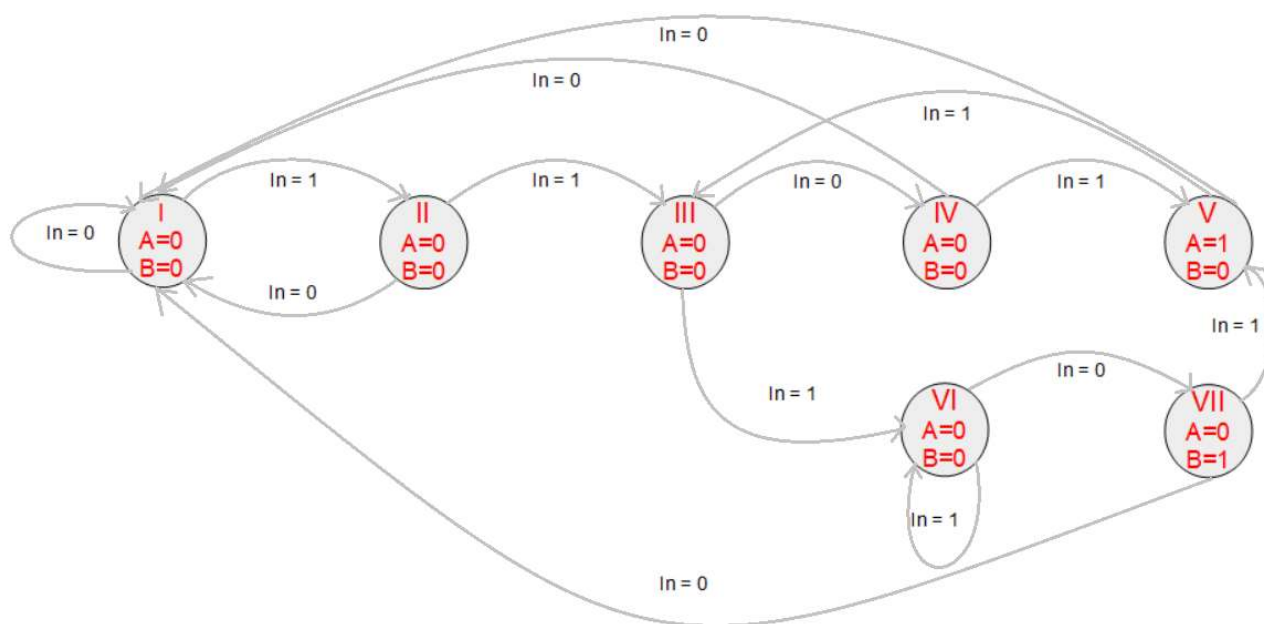
Le espressioni ottenute serviranno per comandare i flip-flop D utili a memorizzare lo stato dei singoli bit ed occuparsi pertanto di aggiornare, sul fronte di salita del clock, lo stato generale del sistema.

Tramite Logisim sarà poi realizzato il circuito di controllo che andrà a pilotare i due contatori con i rispettivi display per indicare il numero di conteggi dei ogni sequenza.

SVILUPPO

Diagramma degli Stati:

STATO CODIFICA	I	II	III	IV	V	VI	VII
	000	001	010	011	100	101	110
USCITE	SeqA = 0 SeqB = 0	SeqA = 0 SeqB = 0	SeqA = 0 SeqB = 0	SeqA = 0 SeqB = 0	SeqA = 1 SeqB = 0	SeqA = 0 SeqB = 0	SeqA = 0 SeqB = 1



Codifica:

il sistema per riconoscere entrambe le sequenze A e B necessita l'utilizzo di 7 stati, quindi 3 bit (2^3 , stati massimi > 7, stati necessari).

Stato I, codice 000:

- In = 0, il sistema rimane invariato poiché nessuna sequenza inizia con 0;
- In = 1, il sistema si porta nello stato successivo II, possibile inizio di sequenza A o B.

Stato II, codice 001:

- In = 0, il sistema ritorna allo stato precedente I poiché nessuna sequenza ha come secondo bit uno 0;
- In = 1, il sistema si porta nello stato successivo III, possibile continuo di sequenza A o B.

Stato III, codice 010:

- In = 0, il sistema si porta nello stato successivo IV, possibile continuo di sequenza A;
- In = 1, il sistema si porta nello stato successivo VI, possibile continuo di sequenza B.

Stato IV, codice 011:

- In = 0, il sistema si porta nello stato di origine I, annullando per cui tutti gli input memorizzati prima ricevuti;
- In = 1, il sistema si porta nello stato successivo V, riconoscendo la sequenza completa A = "1101".

Stato V, codice 100:

- In = 0, il sistema si porta nello stato di origine I, annullando per cui tutti gli input memorizzati prima ricevuti;
- In = 1, il sistema si porta nello stato precedente III, la sequenza A terminando con un 1, formerebbe, insieme all'input appena ricevuto, i primi due bit di una nuova sequenza.
- solo in questo stato l'uscita A viene attivata ad 1, poiché è stata completata la corretta sequenza "1101".

Stato VI, codice 101:

- In = 0, il sistema si porta nello stato successivo VII, riconoscendo la sequenza completa B = "1110".
- In = 1, il sistema rimane invariato poiché aggiungendo un nuovo 1 alla sequenza di 111 già ricevuta, è come fosse in attesa di uno 0 per completare la sequenza B.

Stato V, codice 100:

- In = 0, il sistema si porta nello stato di origine I, annullando per cui tutti gli input memorizzati prima ricevuti;
- In = 1, il sistema si porta nello stato precedente V, la sequenza B terminando con un 110, formerebbe, insieme all'input appena ricevuto, la sequenza A = "1101".
- solo in questo stato l'uscita B viene attivata ad 1, poiché è stata completata la corretta sequenza "1110".

Eventuale stato VIII, codice 111:

questo stato non è considerato poiché il sistema risulta completo anche senza la sua l'inclusione. La tabella della verità potrà pertanto essere riempita da x, cioè stati ininfluenti che possono essere considerati, a seconda della necessita dell'utente sia come 0 oppure 1, per eventuali semplificazioni circuitali.

Tabella transizione degli stati:

s	s2	s1	s0	In	S	S2	S1	S0
I	0	0	0	0	I	0	0	0
I	0	0	0	1	II	0	0	1
II	0	0	1	0	I	0	0	0
II	0	0	1	1	III	0	1	0
III	0	1	0	0	IV	0	1	1
III	0	1	0	1	VI	1	0	1
IV	0	1	1	0	I	0	0	0
IV	0	1	1	1	V	1	0	0
V	1	0	0	0	I	0	0	0
V	1	0	0	1	III	0	1	0
VI	1	0	1	0	VII	1	1	0
VI	1	0	1	1	VI	1	0	1
VII	1	1	0	0	I	0	0	0
VII	1	1	0	1	V	1	0	0
-	1	1	1	0	-	x	x	x
-	1	1	1	1	-	x	x	x

Input:

come ingressi sono considerati gli stati attuali s_2, s_1, s_0 dei 3 flip-flop D e il valore di ln .

Output:

come uscite sono considerate gli stati futuri S_2, S_1, S_0 dei 3 flip-flop D.

Mappe di Karnaugh:

sintesi minima degli stati tramite forma SP degli stati futuri.

			s0 ln			
			00	01	11	10
s2	00		0	0	0	0
	01		0	1	1	0
	11		0	1	x	x
	10		0	0	1	1

$$S_0: s_1 ln + s_2 s_0$$

			s0 ln			
			00	01	11	10
s2	00		0	0	1	0
	01		1	0	0	0
	11		0	0	x	x
	10		0	1	0	1

$$S_1: \overline{s_2} s_1 \overline{s_0} \overline{ln} + \overline{s_2} \overline{s_1} s_0 ln + s_2 \overline{s_1} \overline{s_0} ln + s_2 s_0 \overline{ln}$$

			s0 ln			
			00	01	11	10
s2	00		0	1	0	0
	01		1	1	0	0
	11		0	0	x	x
	10		0	0	1	0

$$S_0: \overline{s_2} \overline{s_0} ln + \overline{s_2} s_1 \overline{s_0} + s_2 s_0 ln$$

Tabella delle uscite:
 sintesi minima delle uscite.

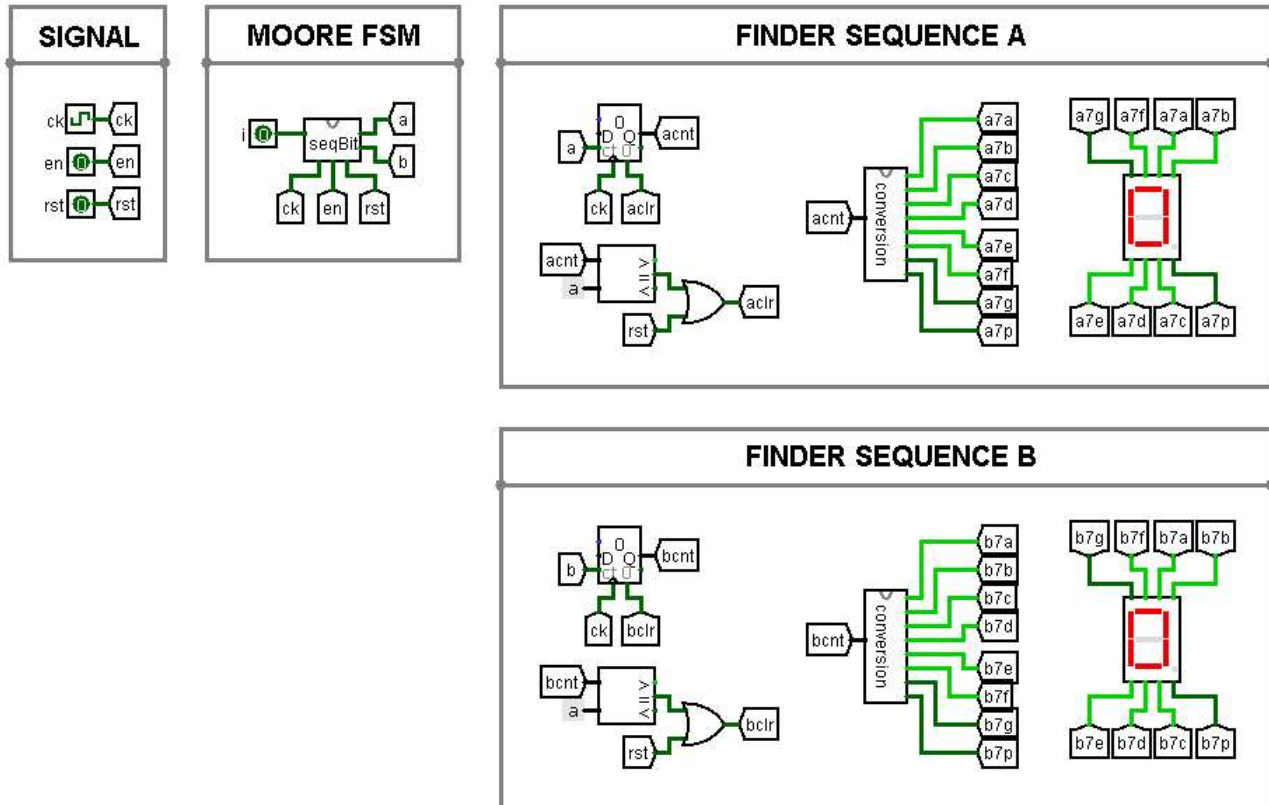
s		s2	s1	s0	A		B
I		0	0	0		0	0
II		0	0	1		0	0
III		0	1	0		0	0
IV		0	1	1		0	0
V		1	0	0		1	0
VI		1	0	1		0	0
VII		1	1	0		0	1

A: $s2 \overline{s1} \overline{s0}$

B: $s2 s1 \overline{s0}$

REALIZZAZIONE

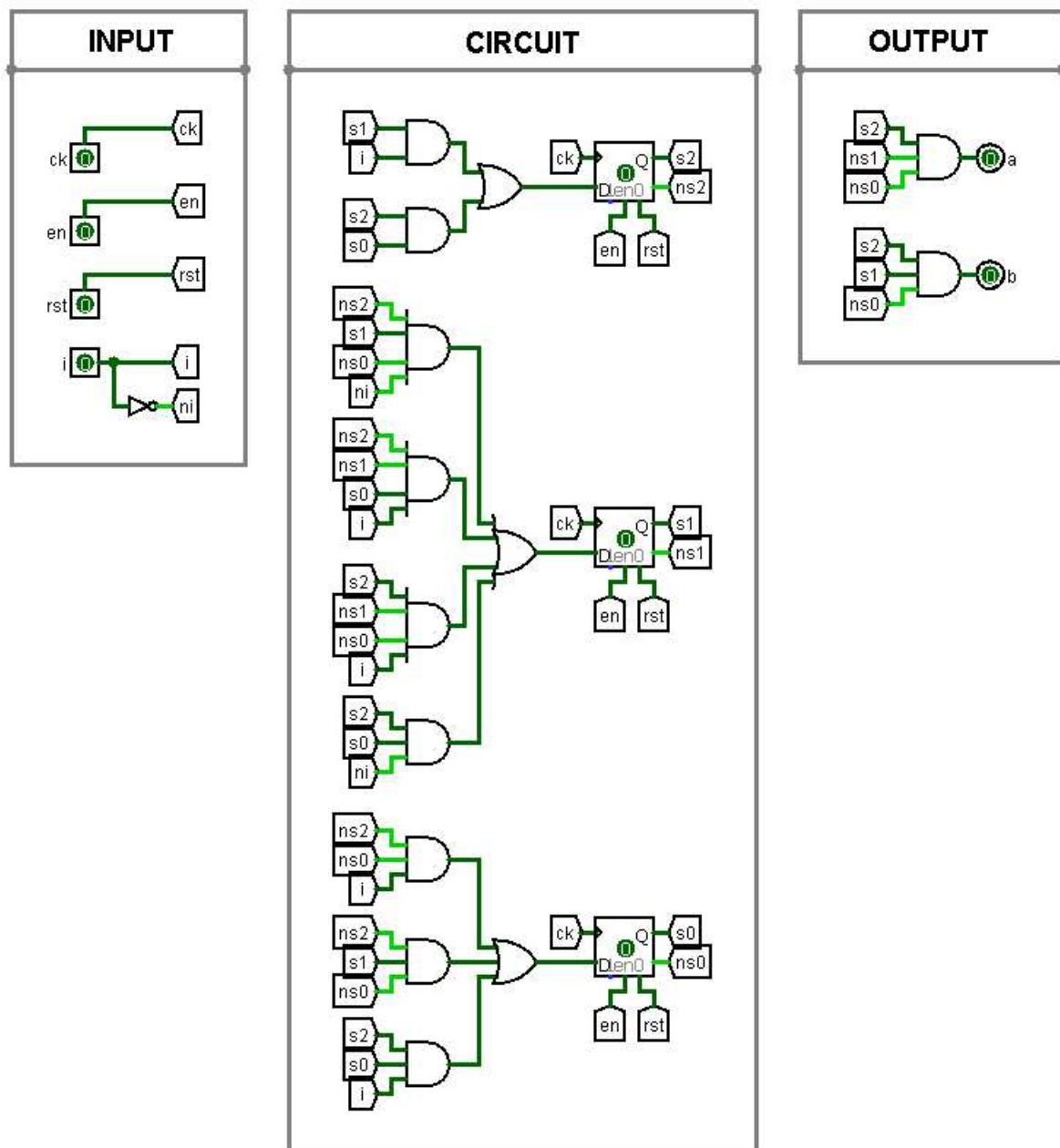
Il circuito realizzato attraverso Logisim si compone di 4 circuiti principali: main, findSeq, conv7Seg e segment. Mentre il main rimane il circuito madre, gli altri sono tutti sottocircuiti sviluppati per rendere maggiormente ordinato e chiaro il funzionamento del sistema.



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 *a* e *b* provenienti dal sottocircuito findSeq (Moore FSM). Da questi ultimi due segnali infatti si possono comandare i contatori che, sul fronte di salita del clock, incrementeranno il loro valore se leggeranno un 1 logico sugli ingressi *a* o *b*. I bit *aclr* e *bclr* si occupano dei clear asincroni dei contatori, disponendo infatti di un solo display 7 segmenti i valori che si possono rappresentare sono unicamente quelli delle unità (valori da 0 a 9) più, eventualmente, altre cifre in notazione esadecimale. Nel caso specifico il clear viene attivato, tramite un comparatore, dal raggiungimento del valore massimo, identificato come *a* (10 in notazione esadecimale), pertanto le cifre visualizzate saranno dalla 0 alla 9, al valore 10 il contatore verrà portato al valore 0 iniziando per cui di nuovo il proprio conteggio.

I bit *acnt* e *bcnt* rappresentano invece il valore dei contatori (0-9), queste uscite sono riportate in ingresso ad un convertitore che si occuperà della gestione e del set dei vari segmenti del display in uscita, per visualizzare il corretto numero letto sull'ingresso.

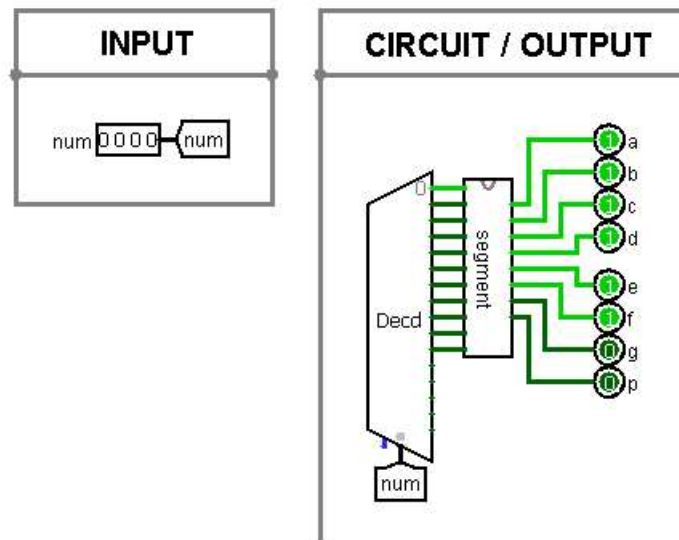


findSeq:

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 restano il clock, enable e reset, ai quali solo si aggiunge il valore dell'input *i* (identificato come *In* nella FSM). 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 3 risultati finali, uno per ogni mappe 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.

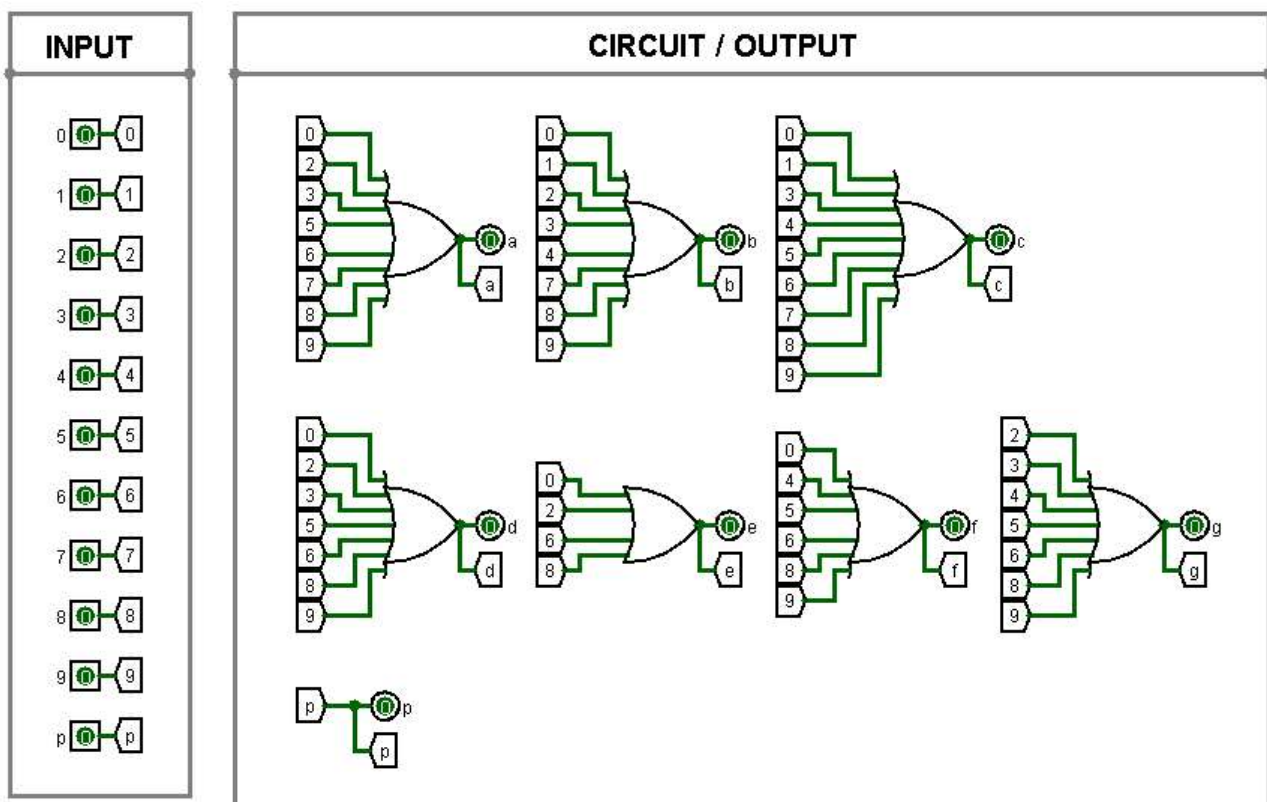
Le uscite dei flip-flop D sono riportate sugli ingressi in maniera di avere un sistema retroattivo, ovvero come appena detto, lo stato futuro non dipende solo dal valore degli ingressi ma anche dallo stato attuale in cui il sistema si trova. Le uscite dei flip-flop D pilotano poi, secondo le espressioni trovate in precedenza, le uscite *a* e *b* dell'intero sistema, che individuano infatti il completamento delle rispettive sequenze di bit.



conv7Seg:

identificato come il circuito in grado di generare le adeguate combinazioni dei segmenti del display, questo sottocircuito si occupa infatti di convertire il valore decimale dei contatori in segnali utili al display 7 segmenti per la corretta visualizzazione dei numeri.

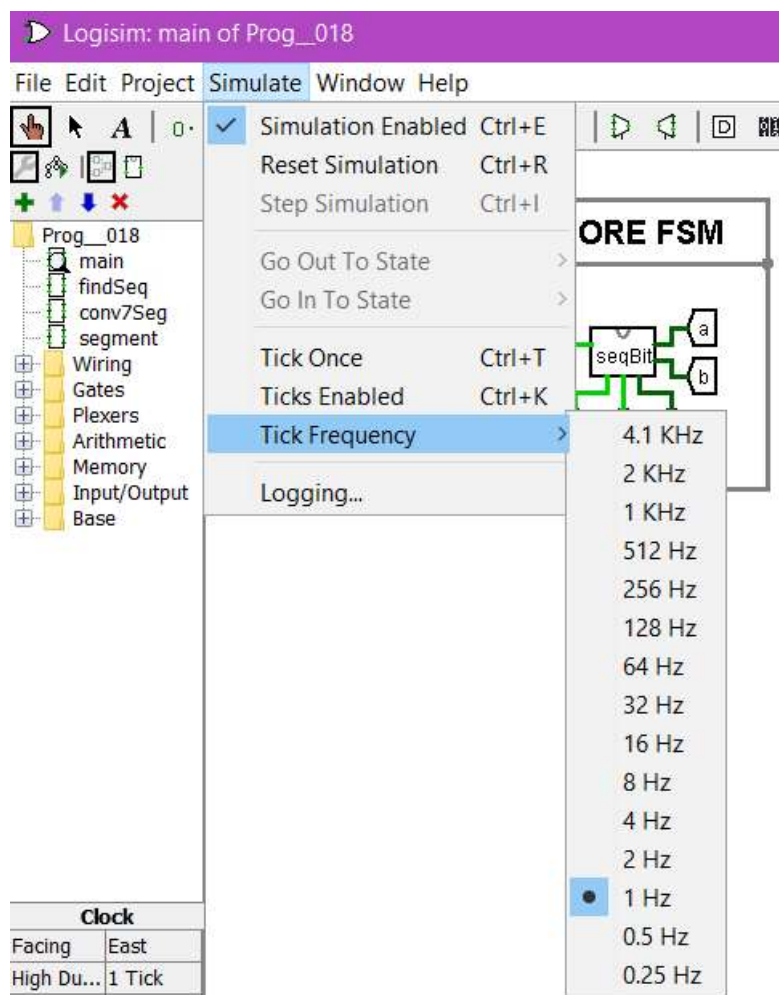
Il decoder infatti abilita una alla volta le vie di uscita, queste passando per un secondo circuito segment producono i segnali per i vari segmenti.



segment:

identificato come il cuore del circuito di conversione numero/segmenti, questo sottocircuito tramite l'attivazione di un determinato valore in ingresso attiva tramite una porta or i rispettivi segmenti del display. Ogni segmento è infatti pilotato da ingressi multipli che, uno alla volta, possono o meno abilitare il segmento in base alla codifica in ingresso.

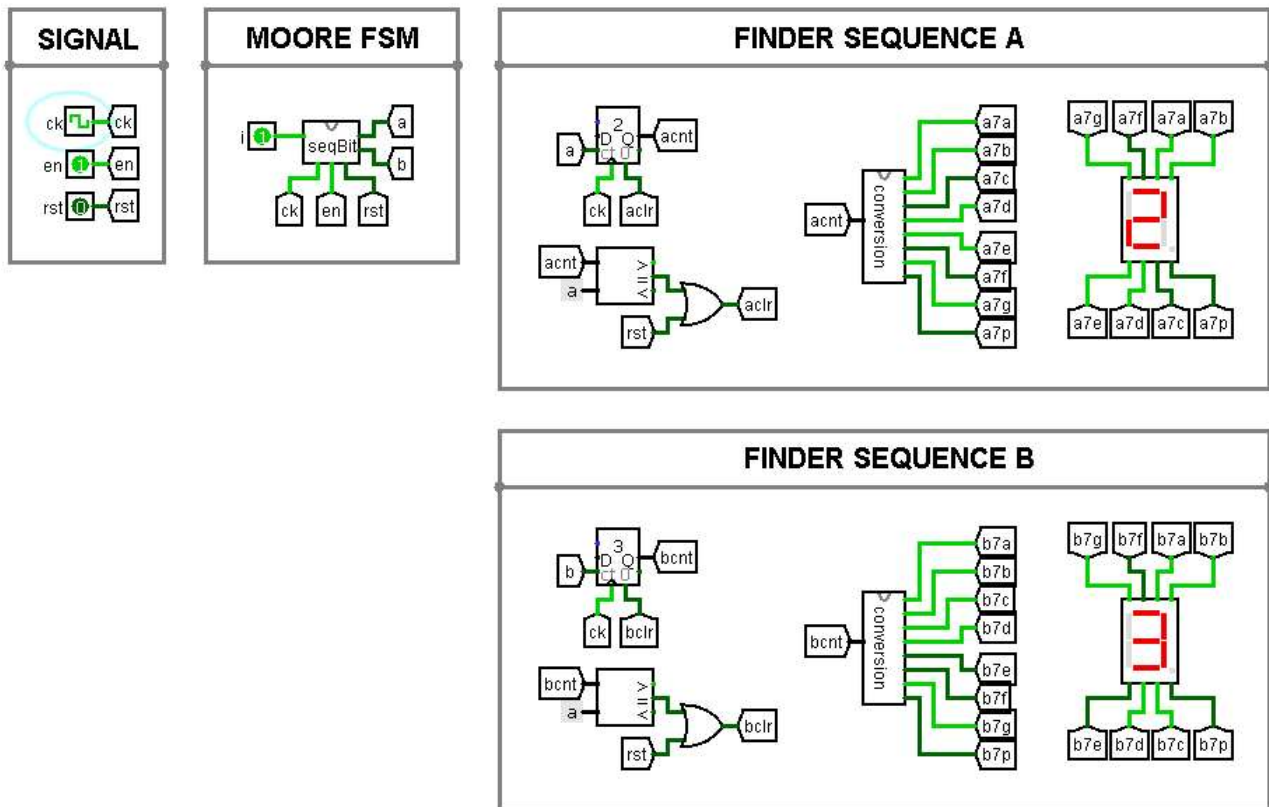
TEST



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 sarà comandato dall'utente in maniera da scandire meglio e con tranquillità gli stati ed ingressi del sistema. Da notare 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. I display, si può notare come visualizzino correttamente il valore presente sui rispettivi contatori, in questo caso il sistema ha già rilevato 3 sequenze B = "1110" ed altre 2 sequenze A = "1101".

APPRONFONDIMENTO PERSONALE

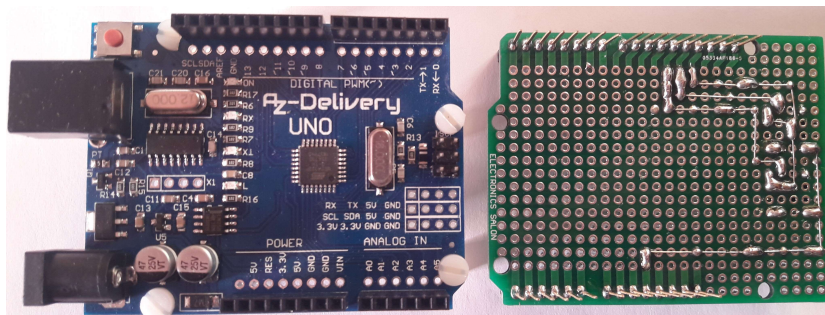
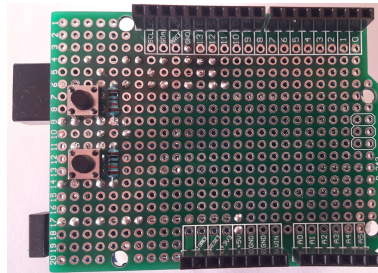
Descrizione:

l'approfondimento personale in questione è realizzato tramite Arduino ed una shield personalizzata, connesso ad un applicativo creato con l'IDE Processing.

Mentre Arduino si occupa della parte hardware del progetto, rilevazione ingressi ed invio dati tramite canale seriale, l'applicativo esegue invece il riconoscimento delle sequenze, leggendo dalla seriale la pressione dei diversi pulsanti e la visualizzazione dei contatori e grafico degli ultimi stati ricevuti.

Shield button:

la shield è stata schematizzata e costruita personalmente, avvalendosi di una schedina conforme alla distanza tra le varie serie di pin di Arduino. La shield è doppia faccia, tuttavia solo il piano inferiore è risultato utile per lo sbroglio delle piste. I pulsanti sono stati messi al bordo per lasciare spazio ad eventuali implementazioni e aggiunte future.



Applicativo:

la pagina presenta set delle sequenze, contatori con reset, grafico e scelta porta seriale.


TS

SEQUENCE RECOGNIZER

9/1/2020
15:25:48

CHANGE SEQUENCE		COUNTER SEQUENCE		COMMUNICATION
CHANGE SEQ.	SEQUENCE	1101		COM12
SEQ. A	1101	CNT. A 3	RESET A	
SEQ. B	1110	CNT. B 2	RESET B	CLOSE

CHART SEQUENCE



WARNING Serial correct opening

CONCLUSIONI

Il progetto approfondito non ha presentato particolari difficoltà nella sua realizzazione e sviluppo. In un primo momento, tuttavia, poteva essere confuso il suo funzionamento, avvalendosi di due FSM distinte per la lettura delle rispettive sequenze A e B. Prestando attenzione a questo dettaglio, e comprimendo la progettazione in una sola FSM, è stato interessante accorgersi come anche da un progetto "ristretto" possa nascere una trattazione maggiore con altri strumenti a disposizione, ad esempio Arduino, Processing, Shield, come in questo caso.