



Canzolino Gianluca

RELAZIONE
TECNOLOGIE DIGITALI

VHDL



Gruppo 16 - Traccia n° 9
Oggetto: Realizzazione di un parcheggio

Indice

1. Traccia d'esame	1
2. Introduzione.....	1
3. Ipotesi iniziali.....	2
4. Ingressi e uscite.....	3
5. Schema a blocchi del sistema	5
6. Automi.....	6
7. Codice VHDL	9
8. Parcheggio.vhd.....	9
9. Entity del parcheggio.....	12
10. RTL Schematic	13
11. Technology Schematic.....	13
12. Sintesi.....	13
13. Contatore_auto.vhd	16
14. Entity del contatore_auto	18
15. Sintesi.....	18
16. Test Bench	19
17. Post-Route.....	28
18. Progetto fisico	29
19. Modellino completo 3D	31
20. Modellino completo reale	31
21. Componenti utilizzati	32
22. Circuiti	33
23. Pinout	33
24. Codice Arduino	37

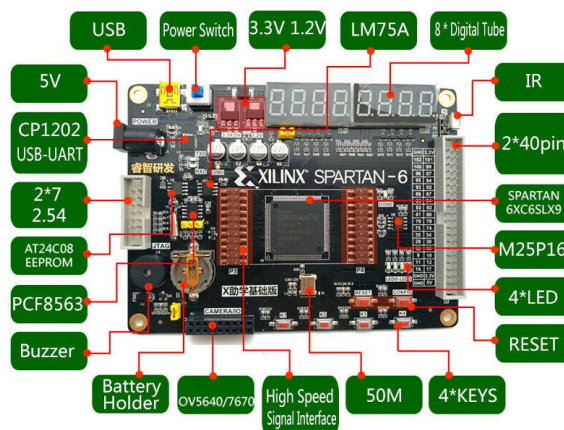
Traccia d'esame

Si realizzi un controllore per gestire un parcheggio dotato di due posti auto. Il parcheggio ha un varco di ingresso e un varco di uscita. Al varco di ingresso è presente una sbarra e un semaforo che servono per controllare il flusso in entrata al parcheggio: se ci sono posti ancora liberi, il controllore deve autorizzare l'accesso tramite una luce verde, se il parcheggio è tutto occupato deve bloccare l'accesso tramite una luce rossa. Per entrare nel parcheggio, l'utente deve premere un pulsante. Se sono disponibili posti, il controllore deve alzare la sbarra e tenerla aperta fino a quando l'auto non entra (il passaggio dell'auto è segnalato da una fotocellula) e per un massimo di 15 colpi di clock. Non appena l'auto è entrata, la sbarra deve essere abbassata. Se l'auto non entra prima dei 15 colpi di clock, la sbarra deve essere comunque abbassata. Al varco di uscita è presente una fotocellula che indica se un'auto intende uscire dal parcheggio. Quando il controllore rileva un'auto in uscita, alza la sbarra. La sbarra resta alzata fino a quando l'auto è uscita dal parcheggio.

Introduzione

Questa relazione ha per oggetto la realizzazione di una macchina per il controllo di un parcheggio il quale è predisposto per 2 posti auto. Un'auto o un veicolo generico può entrare solo e soltanto se è disponibile almeno un posto. Un civile che vuole entrare all'interno di esso, capisce dal semaforo se è disponibile almeno un posto auto (semaforo rosso: posti non disponibili, semaforo verde: almeno un posto libero). Per l'implementazione di tale progetto è stato adottato l'ambiente di sviluppo Xilinx ISE per lo sviluppo del codice, e dei software PlanAhead e Impact per la codifica dei pin sulla scheda FPGA e per il caricamento sulla flash.

La scheda utilizzata nel modellino fisico è la Xilinx Spartan6 XC6SLX9 TQG144.

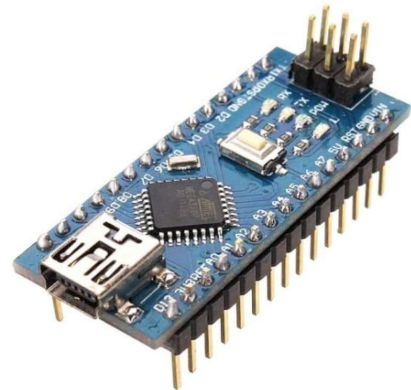


Ipotesi iniziali

Le considerazioni da fare prima dello sviluppo del progetto sono varie, le quali si suddividono in: **fisiche**, **funzionali** e **concettuali**.

Le ***considerazioni fisiche*** sono:

- La sbarra di ingresso e uscita è stata implementata fisicamente tramite due servomotori SG90;
- Le fotocellule usate sono dei sensori di prossimità ad infrarossi per poter simulare al meglio il reale funzionamento di una fotocellula;
- Il semaforo è costituito da due diodi led da 5mm, uno rosso e uno verde;
- Tra i sensori e attuatori e la scheda FPGA è stata implementata una scheda Arduino Nano. Questa scelta si è resa necessaria per lavorare su valori di tensione consoni alla FPGA, la quale può lavorare in ingresso e uscita con valore massimo di 3,3V, mentre i sensori e attuatori lavorano a 5V. Inoltre durante la fase di raccoglimento delle informazioni, i sensori di prossimità tendono ad avere dei valori molto instabili, i quali oscillano in alcuni casi in cui la luce viene riflessa male, oppure ci sono delle sorgenti luminose esterne che favoriscono una lettura errata. Dunque, il microcontrollore gestisce queste eventualità stabilizzando il segnale in uscita;
- Per la realizzazione del prototipo fisico sono stati introdotti anche dei modelli stampati in 3D per avere dei supporti ottimali e per avere il più possibile una somiglianza con il prototipo in scala reale;
- Il pulsante è stato reso impulsivo con il gradino alto di una larghezza adeguata tramite l'utilizzo del microcontrollore Arduino Nano.



Le ***considerazioni funzionali*** sono:

- La fotocellula di ingresso è stata predisposta come un blocco contenente due sensori, uno prima della sbarra (esterna) e uno dopo la sbarra (interna). La fotocellula di uscita è stata predisposta anch'essa come un blocco contenente due sensori, uno prima della sbarra (interna) e uno dopo la sbarra (esterna). Ognuna di essa ammette in uscita un valore binario pari a 0 se non è presente alcun veicolo davanti ad essa e 1, invece, se il veicolo è presente.

Le *considerazioni concettuali* sono:

- Il parcheggio è progettato tale che, in situazione di emergenza, permette di far uscire le auto in continuo. Quindi anche se, in caso ipotetico, il controllore non conta alcun'auto all'interno del parcheggio, esso continua comunque a far transitare in uscita le auto;
- Nel caso in cui un'auto sta per entrare e contemporaneamente una sta per uscire, il controllore dà precedenza all'auto in uscita così da semplificare le manovre all'interno del parcheggio.
- Nel caso in cui all'interno del parcheggio, accanto alla fotocellula di ingresso interna, ci fosse un ostacolo, il controllore non fa entrare nessun'auto. Analogamente anche per l'uscita (fotocellula esterna occupata).
- In ingresso e in uscita può transitare una sola auto per volta.

Ingressi e uscite

Il modello del parcheggio è stato ideato con due automi differenti, uno che gestisce l'intero sistema, mentre l'automa secondario serve a gestire il conteggio delle auto in ingresso e in uscita. Gli ingressi e le uscite sono suddivisi in interni ed esterni:

- **Interni:** i segnali tengono in stretto collegamento i due automi;
- **Esterni:** i segnali provengono dal mondo esterno (input), oppure sono destinati a pilotare attuatori (output).

Gli automi hanno il segnale di clock e il segnale di reset come ingressi in comune, i quali garantiscono un corretto ripristino allo stato iniziale in caso di anomalie o di alcuni casi particolari.

L'automa principale (parcheggio), ammette i seguenti ingressi:

- **FEE:** è il segnale della Fotocellula di Entrata Esterna la quale eroga un valore pari a '1' nel caso in cui abbia rilevato un'auto e '0' viceversa;
- **FEI:** è il segnale della Fotocellula di Entrata Interna la quale eroga un valore pari a '1' nel caso in cui abbia rilevato un'auto e '0' viceversa;
- **FUI:** è il segnale della Fotocellula di Uscita Interna la quale eroga un valore pari a '1' nel caso in cui abbia rilevato un'auto e '0' viceversa;
- **FUE:** è il segnale della Fotocellula di Uscita Esterna la quale eroga un valore pari a '1' nel caso in cui abbia rilevato un'auto e '0' viceversa;
- **Pulsante:** è un segnale di tipo impulsivo il quale assume un valore pari a '1' nel caso in cui un utente abbia premuto il pulsante e '0' viceversa;

E le seguenti uscite:

- **SBI:** è il segnale della **Sbarra** all'**Ingresso**, la quale è alzata nel caso in cui l'automa ammetta un'uscita pari a '1', mentre rimane abbassata nel caso in cui sia '0';
- **SBU:** è il segnale della **Sbarra** all'**Uscita**, la quale è alzata nel caso in cui l'automa ammetta un'uscita pari a '1', mentre rimane abbassata nel caso in cui sia '0';
- **Sem:** è il segnale del **Semaforo**, esso diventa verde se il valore in uscita è '1', mentre diventa rosso nel caso in cui l'uscita è '0'.

L'automa secondario (contatore_auto), ammette i seguenti ingressi:

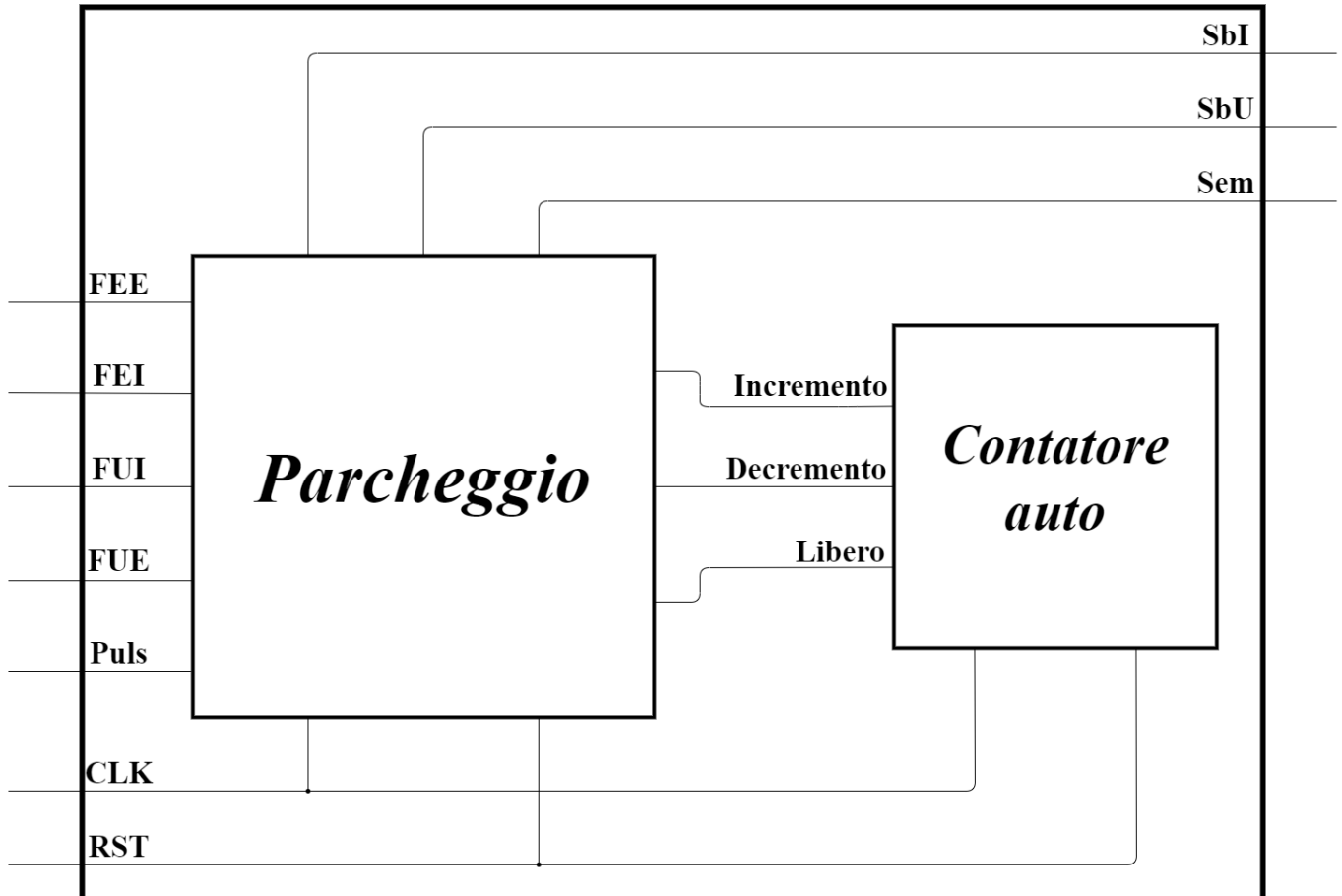
- **Incremento (interno):** è un segnale impulsivo proveniente dall'automa principale. Quando assume valore '1', l'automa cambia stato passando ad un nuovo stato il quale corrisponde al valore successivo (**incremento**). Quando assume valore '0', l'automa non effettua nessuna modifica;
- **Decremento (interno):** è un segnale impulsivo proveniente dall'automa principale. Quando assume valore '1', l'automa cambia stato passando ad un nuovo stato il quale corrisponde al valore precedente (**decremento**). Quando assume valore '0', l'automa non effettua nessuna modifica.

E la seguente uscita:

- **Libero (interno):** è un segnale interno, il quale può assumere valore pari a '1' nel caso in cui c'è almeno un posto **libero** all'interno del parcheggio e '0' nel caso in cui esso sia completamente saturo.

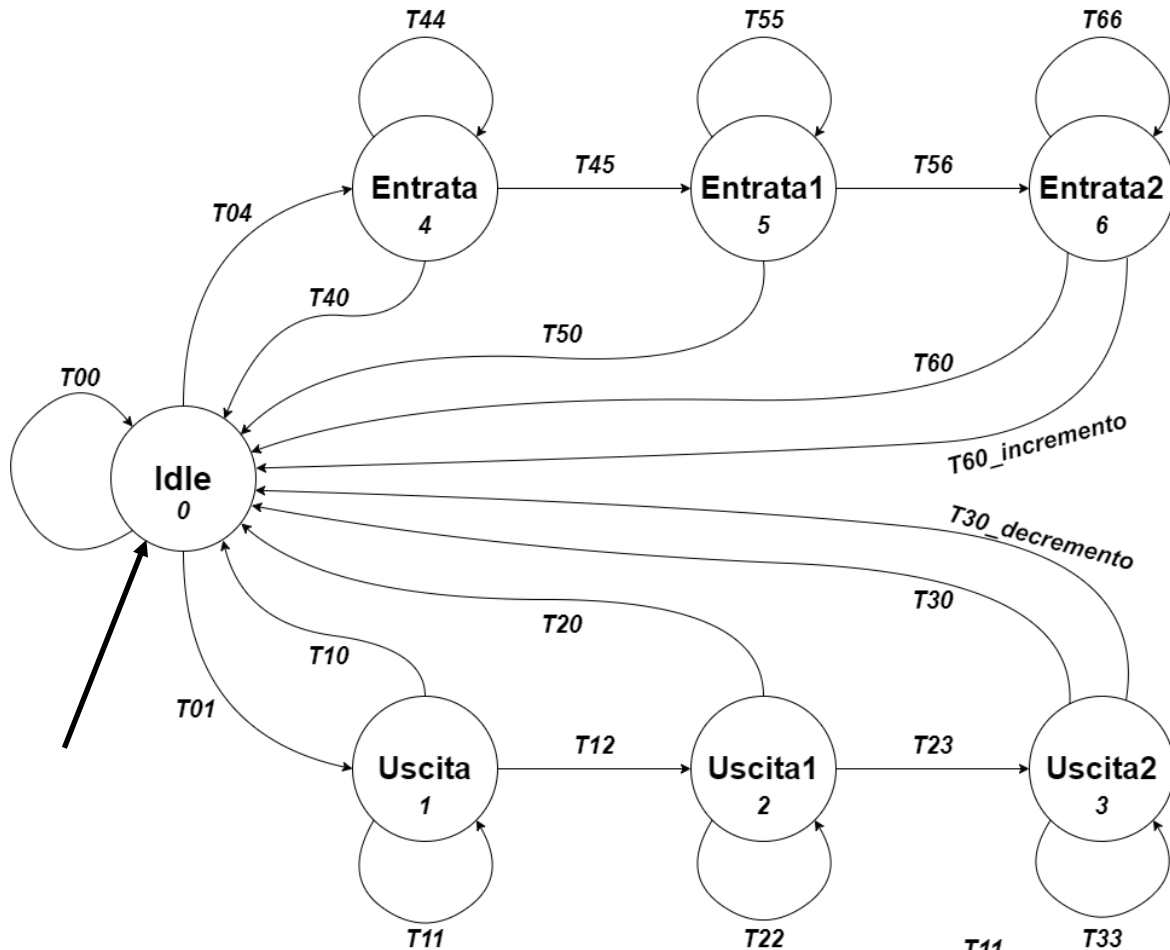
Schema a blocchi del sistema

Il seguente schema rappresenta in maniera sintetica la struttura dell'intero sistema vista dall'alto contenente solamente i segnali di ingresso, uscita e quelli interni.

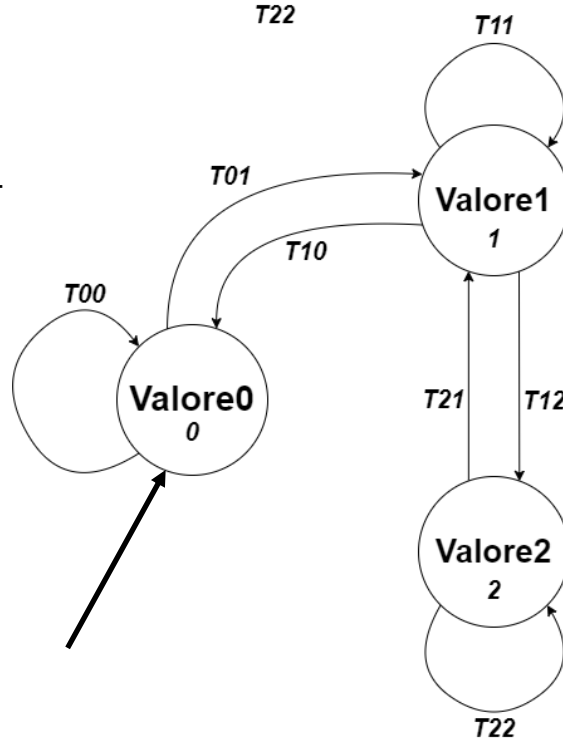


Automi

Come già accennato in precedenza, il sistema è caratterizzato da due FSM, la prima, cioè quella relativa al parcheggio ha il seguente automa:



Il secondo automa (il contatore_auto) è il seguente:



L'automa del parcheggio è un automa di **Moore**, infatti ogni uscita esterna *depende solo e soltanto dallo stato in cui si trova*. Gli stati **Entrata2** e **Uscita2** hanno due transizioni, poiché a seconda dell'ingresso relativo alla sequenza di ingresso o uscita della vettura, il sistema è in grado di rilevare se il veicolo è entrato oppure no, medesimamente se è uscito o meno.

Non si è deciso di estendere l'automa con un altro stato per l'entrata e uno per l'uscita per evitare la codifica su 4 bit (attualmente su 3 bit), perché i segnali di uscita (incremento e decremento) sono dei segnali interni e hanno lo scopo di pilotare l'automa contatore (sono appunto gli ingressi di quell'automa).

Di seguito è riportata la tabella delle transizioni di entrambi gli automi, per semplicità e per favorire una visione pulita dell'automa, sono stati inseriti dei codici per ogni transizioni, i quali possono essere codificati nel seguente modo:

$$T[\text{stato_partenza}][\text{stato_arrivo}]$$

Ad esempio il codice **T01** indica la transizione dallo *stato 0* allo *stato 1*.

Di seguito riportata la tabella delle transizioni dell'automa **parcheggio**.

Transizione	Stato Attuale	Ingressi					Stato Futuro	Uscite			Segnali	
		FEE	FEI	FUI	FUE	Puls		SBI	SBU	SEM	Inc	Dec
T01	Idle	—	—	1	0	—	Uscita	0	0	X	0	0
T04		1	0	0 0 1	0 1 1	1	Entrata					
T00		Tutte le altre combinazioni					Idle					
T10	Uscita	—	—	0 0	0 1	—	Idle	0	1	X	0	0
T12		—	—	1	1	—	Uscita1					
T11		—	—	1	0	—	Uscita					
T20	Uscita1	—	—	0 1	0 0	—	Idle	0	1	X	0	0
T23		—	—	0	1	—	Uscita2					
T22		—	—	1	1	—	Uscita1					
T30_dec	Uscita2	—	—	0	0	—	Idle	0	1	X	0	1
T30		—	—	1 1	0 1	—	Idle				0	0
T33		—	—	0	1	—	Uscita2					

Transi- zione	Stato Attuale	Ingressi					Stato Futuro	Uscite			Segnali	
		FEE	FEI	FUI	FUE	Puls		SBI	SBU	SEM	Inc	Dec
<i>T40</i>	Entrata	0 0	0 1	—	—	—	Idle	1	0	X	0	0
<i>T45</i>		1	1	—	—	—	Entrata1					
<i>T44</i>		1	0	—	—	—	Entrata					
<i>T50</i>	Entrata1	0 1	0 0	—	—	—	Idle	1	0	X	0	0
<i>T56</i>		0	1	—	—	—	Entrata2					
<i>T55</i>		1	1	—	—	—	Entrata1					
<i>T60_inc</i>	Entrata2	0	0	—	—	—	Idle	1	0	X	1	0
<i>T60</i>		1 1	0 1	—	—	—	Idle				0	0
<i>T66</i>		0	1	—	—	—	Entrata2					

La **X** del semaforo indica che è dipendente dall'uscita dell'automata contatore. Se l'uscita del contatore “**Libero**” è pari a **1**, *semaforo* assume valore **1**, se invece esso è pari **0** anche *semaforo* assume valore **0**.

Di seguito riportata la tabella delle transizioni dell'automata **contatore_auto** anch'esso usando **Moore**.

Transi- zione	Stato Attuale	Ingressi		Stato Futuro	Uscite
		Incremento	Decremento		Libero
<i>T01</i>	Valore0	0 0 1	0 1 1	Valore0	1
<i>T00</i>		1	0	Valore1	
<i>T10</i>	Valore1	0	1	Valore0	1
<i>T12</i>		1	0	Valore2	
<i>T11</i>		0 1	0 1	Valore1	
<i>T21</i>	Valore2	0	1	Valore1	0
<i>T22</i>		0 1 1	0 0 1	Valore2	

Codice VHDL

Una volta descritta l'architettura e dettagliata, si è passata alla fase di scrittura del codice VHDL. In seguito, è riportato l'intero codice commentato riga per riga per poter semplificare la vista e la comprensione di tale codice.

~ *Parcheggio.vhd*

```
1 -----
2 -- Company:      Ingegneria Informatica a.a. 2019/2020
3 -- Engineer:     Canzolino Gianluca 0612704462
4 --
5 -- Create Date:  10:21:15 01/20/2020
6 -- Design Name:
7 -- Module Name:  v_1_0_1_2020_01_20 - Behavioral
8 -- Project Name: Progetto parcheggio
9 -- Target Devices: Xilinx FPGA XC6SLX9-2TQG144
10 -- Tool versions:
11 -- Description:
12 --
13 -- Si realizzi un controllore per gestire un parcheggio dotato di due posti auto. Il parcheggio ha un varco di
14 -- ingresso e un varco di uscita. Al varco di ingresso è presente una sbarra e un semaforo che servono per
15 -- controllare il flusso in entrata al parcheggio: se ci sono posti ancora liberi, il controllore deve autorizzare
16 -- l'accesso tramite una luce verde, se il parcheggio è tutto occupato deve bloccare l'accesso tramite una luce
17 -- rossa. Per entrare nel parcheggio, l'utente deve premere un pulsante. Se sono disponibili posti, il controllore
18 -- deve alzare la sbarra e tenerla aperta fino a quando l'auto non entra (il passaggio dell'auto è segnalato da
19 -- una fotocellula) e per un massimo di 15 colpi di clock. Non appena l'auto è entrata, la sbarra deve essere
20 -- abbassata. Se l'auto non entra prima dei 15 colpi di clock, la sbarra deve essere comunque abbassata. Al varco
21 -- di uscita è presente una fotocellula che indica se un'auto intende uscire dal parcheggio. Quando il controllore
22 -- rileva un'auto in uscita, alza la sbarra. La sbarra resta alzata fino a quando l'auto è uscita dal parcheggio.
23 --
24 -- Dependencies:
25 --
26 -- Revision:      1_0_1
27 -- Revision 0.01 - File Created
28 -- Additional Comments:
29 --
30 -----
31 library IEEE;
32 use IEEE.STD_LOGIC_1164.ALL;
33 use IEEE.STD_LOGIC_unsigned.ALL;
34
35 entity Parcheggio is
36     Port (
37         clk      : in    STD_LOGIC; --Segnale di clock
38         rst      : in    STD_LOGIC; --Segnale di reset
39         FEE      : in    STD_LOGIC; --Fotocellula Entrata Esterna -> '1' rilevato, '0' non rilevato
40         FEI      : in    STD_LOGIC; --Fotocellula Entrata Interna -> '1' rilevato, '0' non rilevato
41         FUE      : in    STD_LOGIC; --Fotocellula Uscita Esterna -> '1' rilevato, '0' non rilevato
42         FUI      : in    STD_LOGIC; --Fotocellula Uscita Interna -> '1' rilevato, '0' non rilevato
43         Pulsante : in    STD_LOGIC; --Pulsante all'ingresso -> Impulsivo -> '1' per 20 millisecondi
44         SbI      : out   STD_LOGIC; --Sbarra Ingresso -> '1' apertura, '0' chiusura
45         SbU      : out   STD_LOGIC; --Sbarra Uscita -> '1' apertura, '0' chiusura
46         Sem      : out   STD_LOGIC; --Semaforo -> '1' semaforo verde, '0' semaforo rosso
47     end Parcheggio;
48
49     architecture Behavioral of Parcheggio is
50
51         --Gli stati, come già descritti nell'automa del parcheggio sono i seguenti:
52         type stato is (idle, entrata, uscita, entratal, entrata2, uscital, uscita2);
53
54         --Per poter individuare lo stato in cui si trova la FSM e lo stato successivo,
55         --sono stati implementati i seguenti segnali:
56         signal current_state, next_state: stato;
57
58         --Per il conteggio delle auto in ingresso e in uscita si ricorre all'uso di un
59         --componente esterno chiamato contatore_auto, il quale se assume incremento 1,
60         --il numero delle auto viene appunto incrementato, se decremento assume valore 1,
61         --il numero delle auto viene decrementato.
62         component contatore_auto
63             Port (
64                 clk      : in    STD_LOGIC;
65                 rst      : in    STD_LOGIC;
66                 incremento : in    STD_LOGIC;
67                 decremento : in    STD_LOGIC;
68                 libero    : out   STD_LOGIC;
69             end component;
70
71         --Dato che il valore Sem è un valore di sola uscita è stato implementato un segnale
72         --di supporto il quale contiene l'uscita "Libero" del componente contatore_auto.
73         signal valore_semaforo : std_logic;
74
75         --Sono stati implementati i segnali inc e dec, i quali sono impulsivi con valore 1.
76         --Essi sono passati al componente contatore_auto, il quale incrementa o decrementa il
77         --numero delle auto presenti
78         signal inc, dec : std_logic;
79     end architecture;
```

```

79
80 begin
81     --Viene istanziato il componente contatore_auto
82     cont_auto : contatore_auto port map (clk, rst, inc, dec, valore_semaforo);
83
84     --Si effettua un primo processo sul clock
85     processo_clock : process(clk)
86     --Si dichiara una variabile intera locale su 4 bit (da 0 a 15) per
87     --il conteggio dei colpi di clock.
88     variable contatore : integer range 0 to 15;
89     begin
90     --Se il clock effettua un fronte di salita, allora eseguo:
91     if(rising_edge(clk)) then
92         --Se il mio reset è 1, allora resetto la macchina
93         if(rst = '1') then
94             --Il mio stato attuale è riportato al valore iniziale, cioè idle
95             current_state <= idle;
96             --Mi resetto il contatore dei colpi di clock
97             contatore := 0;
98             --Altrimenti, se non ho nessun reset, eseguo:
99             else
100                 --Lo stato corrente, viene aggiornato con lo stato successivo
101                 current_state <= next_state;
102             end if;
103
104             --Se mi trovo in uno stato di entrata, allora eseguo:
105             if(current_state = entrata or current_state = entratal or current_state = entrata2) then
106                 --Incremento il contatore dei colpi di clock
107                 contatore := contatore + 1;
108                 --Se il contatore arriva al suo valore massimo, cioè quindi, allora:
109                 if (contatore = 15) then
110                     --Riporto il contatore al valore iniziale
111                     contatore := 0;
112                     --Sono trascorsi 15 colpi di clock quindi non devo più far entrare l'auto,
113                     --riportando quindi lo stato attuale, in quello iniziale
114                     current_state <= idle;
115                 end if;
116                 --Se non mi trovo in nessun stato di entrata, allora:
117                 else
118                     --Resetto il contatore, il quale serve solo in caso in cui l'auto sta entrando
119                     contatore := 0;
120                 end if;
121
122                 --Assegno il valore del Semaforo (1 verde, 0 rosso) all'uscita "Sem"
123                 Sem <= valore_semaforo;
124             end if;
125         end process;
126
127     --Si effettua il processo itinerante alle transizioni di stato
128     processo_FSM : process(current_state, FEE, FEI, FUE, FUI, Pulsante, valore_semaforo)
129
130     begin
131     --Vado a tenere di default i valori inc e dec a 0
132     inc <= '0';
133     dec <= '0';
134
135     --Inizializzo il case ... when per la macchina a stati sul current_state
136     case current_state is
137     --Se lo stato attuale è idle:
138     when idle =>
139         --Essendo un automa di Moore, le mie uscite sono indipendenti dagli ingressi
140         SBI <= '0'; SBU <= '0';
141
142         --Se l'auto sta impegnando l'uscita allora:
143         if(FUI = '1' and FUE = '0') then
144             --Il mio stato successivo è quello di uscita
145             next_state <= uscita;
146             --Altrimenti se sta impegnando l'entrata, e ci sono posti disponibili allora:
147             elsif(FEE = '1' and FEI = '0' and pulsante = '1' and valore_semaforo = '1') then
148                 --Il mio stato successivo è quello di entrata
149                 next_state <= entrata;
150             --Altrimenti (in tutti gli altri casi), allora:
151             else
152                 --Rimango in me stesso, ovvero il prossimo stato è di nuovo idle
153                 next_state <= idle;
154             end if;

```

```

155
156 --Se lo stato attuale è entrata
157 when entrata =>
158     --Essendo un automa di Moore, le mie uscite sono indipendenti dagli igressi
159     SBU <= '0'; SBI <= '1';
160
161     --Se l'auto rimane nella stessa posizione senza indietreggiare o avanzare, allora:
162     if(FEE = '1' and FEI = '0') then
163         --Rimango in me stesso, ovvero il prossimo stato è di nuovo entrata
164         next_state <= entrata;
165     --Altrimenti se l'auto avanza, allora:
166     elsif(FEE = '1' and FEI = '1') then
167         --Il mio prossimo stato è entratal
168         next_state <= entratal;
169     --Altrimenti (l'auto è indietreggiata oppure si verifica uno caso anomalo), allora:
170     else
171         --Ritorno allo stato iniziale
172         next_state <= idle;
173     end if;
174
175 --Se lo stato attuale è entratal
176 when entratal =>
177     --Essendo un automa di Moore, le mie uscite sono indipendenti dagli igressi
178     SBU <= '0'; SBI <= '1';
179
180     --Se l'auto rimane nella stessa posizione senza indietreggiare o avanzare, allora:
181     if(FEE = '1' and FEI = '1') then
182         --Rimango in me stesso, ovvero il prossimo stato è di nuovo entratal
183         next_state <= entratal;
184     --Altrimenti se l'auto avanza, allora:
185     elsif(FEE = '0' and FEI = '1') then
186         --Il mio prossimo stato è entrata2
187         next_state <= entrata2;
188     --Altrimenti (l'auto è indietreggiata oppure si verifica uno caso anomalo), allora:
189     else
190         --Ritorno allo stato iniziale
191         next_state <= idle;
192     end if;
193
194 --Se lo stato attuale è entrata2
195 when entrata2 =>
196     --Essendo un automa di Moore, le mie uscite sono indipendenti dagli igressi
197     SBU <= '0'; SBI <= '1';
198
199     --Se l'auto rimane nella stessa posizione senza indietreggiare o avanzare, allora:
200     if(FEE = '0' and FEI = '1') then
201         --Rimango in me stesso, ovvero il prossimo stato è di nuovo entrata2
202         next_state <= entrata2;
203     --Altrimenti se l'auto avanza, allora:
204     elsif(FEE = '0' and FEI = '0') then
205         --Incremento il numero delle auto
206         inc <= '1';
207         --E ritorno allo stato iniziale
208         next_state <= idle;
209     --Altrimenti (l'auto è indietreggiata oppure si verifica uno caso anomalo), allora:
210     else
211         --Ritorno allo stato iniziale
212         next_state <= idle;
213     end if;
214
215 --Se lo stato attuale è uscita
216 when uscita =>
217     --Essendo un automa di Moore, le mie uscite sono indipendenti dagli igressi
218     SBI <= '0'; SBU <= '1';
219
220     --Se l'auto rimane nella stessa posizione senza indietreggiare o avanzare, allora:
221     if(FUI = '1' and FUE = '0') then
222         --Rimango in me stesso, ovvero il prossimo stato è di nuovo uscita
223         next_state <= uscita;
224     --Altrimenti se l'auto avanza, allora:
225     elsif(FUI = '1' and FUE = '1') then
226         --Il mio prossimo stato è uscital
227         next_state <= uscital;
228     --Altrimenti (l'auto è indietreggiata oppure si verifica uno caso anomalo), allora:
229     else
230         --Ritorno allo stato iniziale
231         next_state <= idle;
232     end if;

```

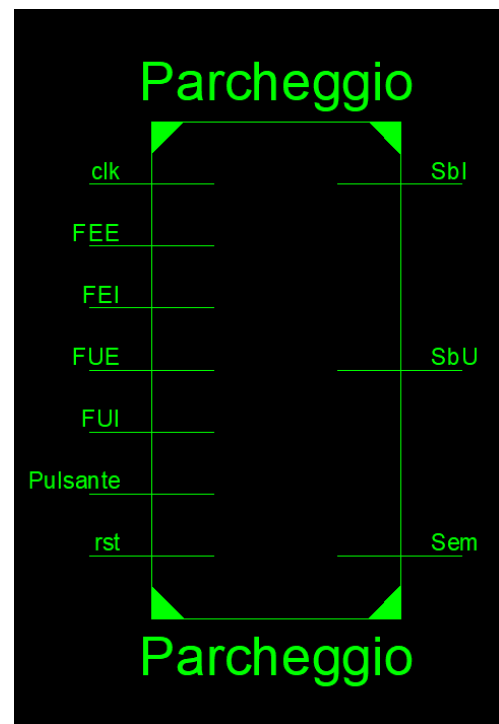
```

233
234 --Se lo stato attuale è uscital
235 when uscital =>
236     --Essendo un automa di Moore, le mie uscite sono indipendenti dagli igressi
237     SBI <= '0'; SBU <= '1';
238
239     --Se l'auto rimane nella stessa posizione senza indietreggiare o avanzare, allora:
240     if(FUI = '1' and FUE = '1') then
241         --Rimango in me stesso, ovvero il prossimo stato è di nuovo uscital
242         next_state <= uscital;
243     --Altrimenti se l'auto avanza, allora:
244     elsif(FUI = '0' and FUE = '1') then
245         --Il mio prossimo stato è uscital
246         next_state <= uscita2;
247     --Altrimenti (l'auto è indietreggiata oppure si verifica uno caso anomalo), allora:
248     else
249         --Ritorno allo stato iniziale
250         next_state <= idle;
251     end if;
252
253 --Se lo stato attuale è uscita2
254 when uscita2 =>
255     --Essendo un automa di Moore, le mie uscite sono indipendenti dagli igressi
256     SBI <= '0'; SBU <= '1';
257
258     --Se l'auto rimane nella stessa posizione senza indietreggiare o avanzare, allora:
259     if(FUI = '0' and FUE = '1') then
260         --Rimango in me stesso, ovvero il prossimo stato è di nuovo uscita2
261         next_state <= uscita2;
262     --Altrimenti se l'auto avanza, allora:
263     elsif(FUI = '0' and FUE = '0') then
264         --Decremento il numero delle auto
265         dec <= '1';
266         --E ritorno allo stato iniziale
267         next_state <= idle;
268     --Altrimenti (l'auto è indietreggiata oppure si verifica uno caso anomalo), allora:
269     else
270         --Ritorno allo stato iniziale
271         next_state <= idle;
272     end if;
273 end case;
274 end process;
275 end Behavioral;

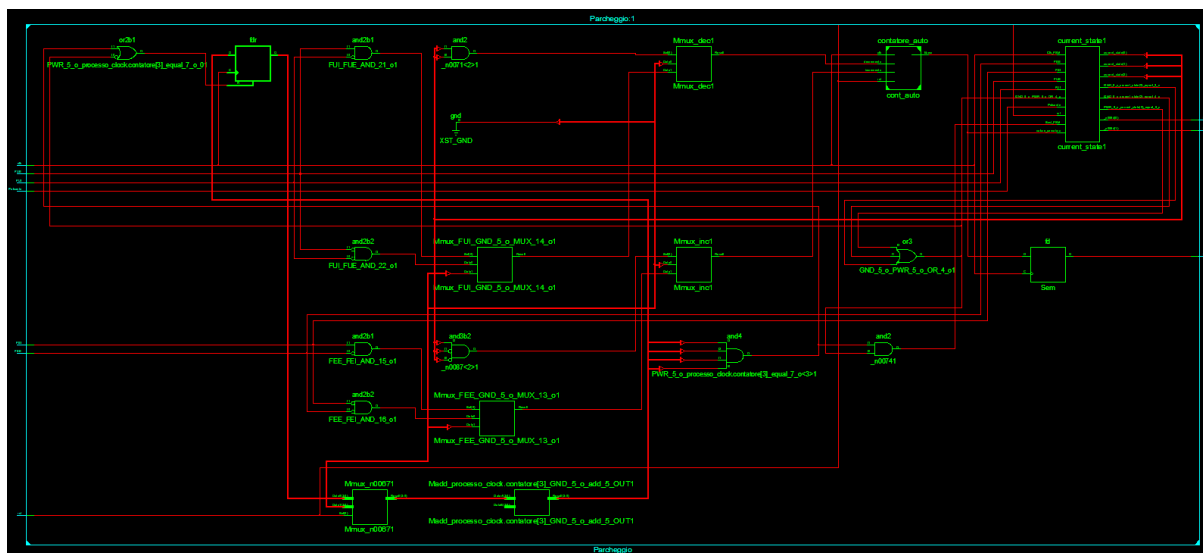
```

Entity del parcheggio

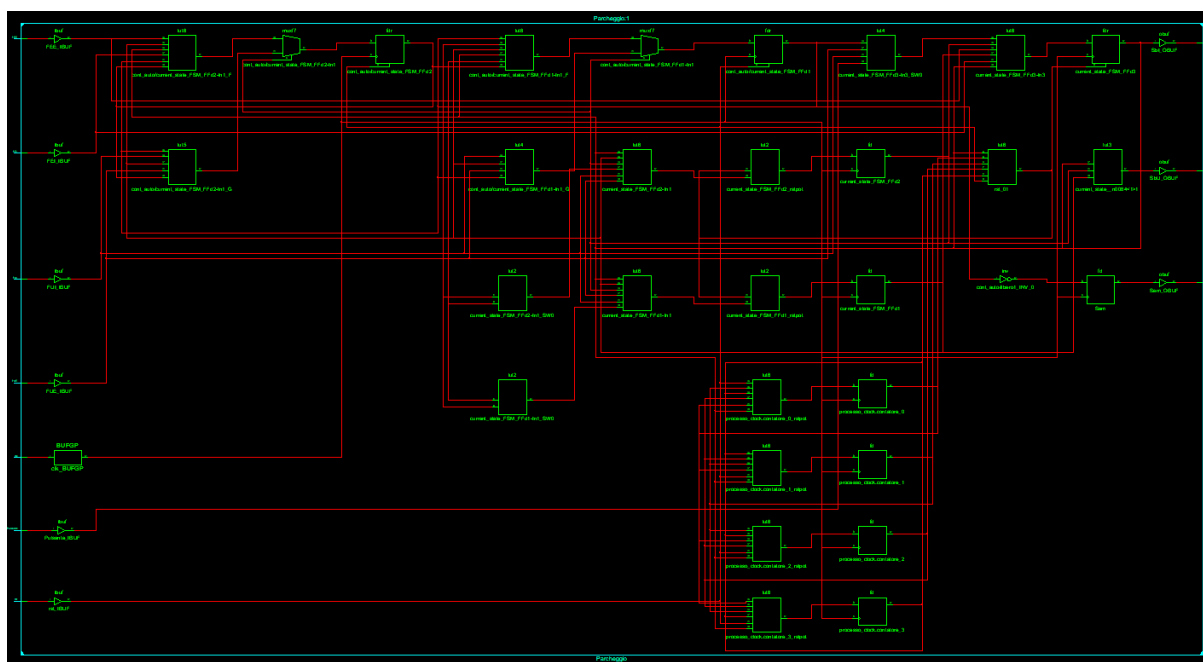
Una volta sintetizzato il codice, il risultato è stato il seguente componente (Entity), il quale è possibile osservarlo sia con la RTL Schematic che con la Technology Schematic.



RLT Schematic



Technology Schematic



Sintesi

Invece, per quanto riguarda la sintesi degli stati, degli input, degli output e di tutti gli elementi logici, possiamo apprezzare le informazioni che ci vengono fornite nella sezione “Synthesis Report”. I risultati sono i seguenti:

- Stati, transizioni, input, output e lo stato iniziale e di reset;

```
Synthesizing Unit <Parcheggio>.
  Related source file is "D:\ProgettoTD_Traccia9_Canzolino_Gianluca\Parcheggio.vhd".
  Found 4-bit register for signal <processo_clock.contatore>.
  Found 1-bit register for signal <Sem>.
  Found 3-bit register for signal <current_state>.
  Found finite state machine <FSM_0> for signal <current_state>.
-----
| States           | 7 |
| Transitions      | 36 |
| Inputs           | 8 |
| Outputs          | 8 |
| Clock            | clk (rising_edge) |
| Reset            | _n0074 (positive) |
| Reset type       | synchronous |
| Reset State      | idle |
| Power Up State   | idle |
| Encoding         | auto |
| Implementation   | LUT |
-----
Found 4-bit adder for signal <processo_clock.contatore[3]_GND_5_o_add_5_OUT> created at line 107.
Summary:
  inferred 1 Adder/Subtractor(s).
  inferred 5 D-type flip-flop(s).
  inferred 5 Multiplexer(s).
  inferred 1 Finite State Machine(s).
Unit <Parcheggio> synthesized.
```

- Componenti logici utilizzati (da notare che il codice è stato implementato per non presentare alcun latch);

```
=====
*                               Design Summary                               *
=====

Top Level Output File Name      : Parcheggio.ngc

Primitive and Black Box Usage:
-----
# BELS                          : 21
#   INV                         : 1
#   LUT2                        : 4
#   LUT3                        : 1
#   LUT4                        : 2
#   LUT5                        : 1
#   LUT6                        : 10
#   MUXF7                       : 2
# FlipFlops/Latches             : 10
#   FD                         : 7
#   FDR                         : 3
# Clock Buffers                 : 1
#   BUFGP                      : 1
# IO Buffers                    : 9
#   IBUF                       : 6
#   OBUF                       : 3
```

- Componenti logici utilizzati sul chip specifico (6slx9tqgl44);

```

Device utilization summary:
-----

Selected Device : 6slx9tqgl44-2

Slice Logic Utilization:
Number of Slice Registers:          10 out of 11440    0%
Number of Slice LUTs:              19 out of 5720     0%
    Number used as Logic:          19 out of 5720     0%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 19
    Number with an unused Flip Flop: 9 out of 19      47%
    Number with an unused LUT:       0 out of 19      0%
    Number of fully used LUT-FF pairs: 10 out of 19    52%
    Number of unique control sets:    3

IO Utilization:
Number of IOs:                      10
Number of bonded IOBs:              10 out of 102     9%

Specific Feature Utilization:
Number of BUFG/BUFGCTRLs:           1 out of 16       6%

-----

Partition Resource Summary:
-----

    No Partitions were found in this design.

-----

```

- La codifica degli stati.

```

Analyzing FSM <MFsm> for best encoding.
Optimizing FSM <FSM_0> on signal <current_state[1:3]> with sequential encoding.
-----
State   | Encoding
-----
idle    | 000
entrata | 001
uscita  | 010
entratal | 011
entrata2 | 101
uscital  | 100
uscita2  | 110
-----
Analyzing FSM <MFsm> for best encoding.

```

~ Contatore_auto.vhd

```
1  -----
2  -- Company:      Ingegneria Informatica a.a. 2019/2020
3  -- Engineer:     Canzolino Gianluca 0612704462
4  --
5  -- Create Date:  14:25:46 01/20/2020
6  -- Design Name:
7  -- Module Name:  v_1_0_1_2020_01_20 - Behavioral
8  -- Project Name: contatore_auto
9  -- Target Devices: Xilinx FPGA XC6SLX9-2TQG144
10 -- Tool versions:
11 -- Description:
12 -- Contatore delle auto in ingresso e in uscita all'interno di un parcheggio
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 -----
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23
24 entity contatore_auto is
25     Port (
26         clk          : in    STD_LOGIC;    --Segnale di clock
27         rst          : in    STD_LOGIC;    --Segnale di reset
28         incremento   : in    STD_LOGIC;    --Incremento
29                                         -- -> Impulsivo -> '1' per 1 colpo di clk
30         decremento   : in    STD_LOGIC;    --Decremento
31                                         -- -> Impulsivo -> '1' per 1 colpo di clk
32         libero       : out   STD_LOGIC);    --Posti liberi\
33                                         -- -> '1' almeno 1 posto libero, '0' saturo
34 end contatore_auto;
35
36 architecture Behavioral of contatore_auto is
37
38     --Gli stati, come già descritti nell'automa del contatore_auto sono i seguenti:
39     type stato is (valore0, valore1, valore2);
40
41     --Per poter individuare lo stato in cui si trova la FSM e lo stato successivo,
42     --sono stati implementati i seguenti segnali:
43     signal current_state, next_state : stato;
44
45 begin
46
47     --Si effettua un primo processo sul clock
48     processo_clock_cont : process(clk)
49     begin
50         --Se il clock effettua un fronte di salita, allora eseguo:
51         if(rising_edge(clk)) then
52             --Se il mio reset è 1, allora resetto la macchina
53             if(rst = '1') then
54                 --Il mio stato attuale è riportato al valore iniziale, cioè valore0
55                 current_state <= valore0;
56             --Altrimenti, se non ho nessun reset, eseguo:
57             else
58                 --Lo stato corrente, viene aggiornato con lo stato_successivo
59                 current_state <= next_state;
60             end if;
61         end if;
62     end process;
```

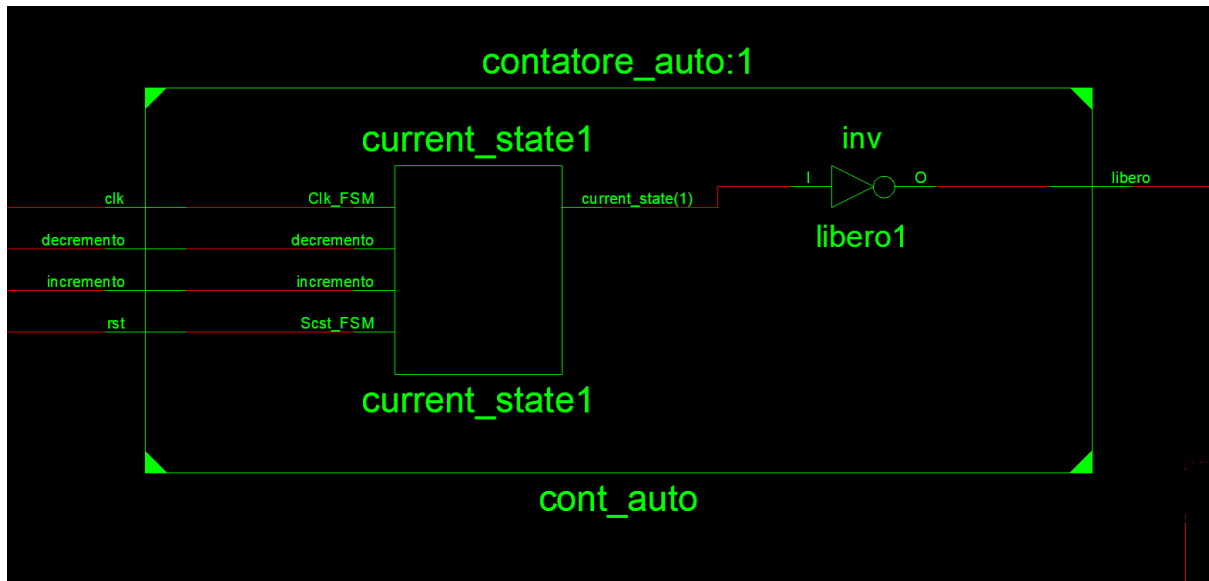
```

63
64 --Si effettua il processo itinerante alle transizioni di stato
65 processo_FSM_2 : process(incremento, decremento, current_state)
66 begin
67   --Inizializzo il case ... when per la macchina a stati sul current_state
68   case current_state is
69     --Se lo stato attuale è valore0:
70     when valore0 =>
71       --Essendo un automa di Moore, le uscite sono indipendenti dagli ingressi
72       libero <= '1';
73
74       --Se ho un incremento, allora:
75       if(incremento = '1' and decremento = '0') then
76         --Il mio stato successivo è valore1
77         next_state <= valore1;
78       --Altrimenti in tutti gli altri casi, eseguo:
79       else
80         --Ritorno in me stesso
81         next_state <= valore0;
82       end if;
83
84     --Se lo stato attuale è valore1:
85     when valore1 =>
86       --Essendo un automa di Moore, le mie uscite sono indipendenti dagli ingressi
87       libero <= '1';
88
89       --Se ho un incremento, allora:
90       if(incremento = '1' and decremento = '0') then
91         --Il mio stato successivo è valore2
92         next_state <= valore2;
93       --Altrimenti se ho un decremento, allora:
94       elsif(incremento = '0' and decremento = '1') then
95         --Il mio stato successivo è valore0
96         next_state <= valore0;
97       --Altrimenti in tutti gli altri casi, eseguo:
98       else
99         --Ritorno in me stesso
100        next_state <= valore1;
101      end if;
102
103     --Se lo stato attuale è valore2:
104     when valore2 =>
105       --Essendo un automa di Moore, le mie uscite sono indipendenti dagli ingressi
106       libero <= '0';
107
108       --Se ho un decremento, allora:
109       if(incremento = '0' and decremento = '1') then
110         --Il mio stato successivo è valore1
111         next_state <= valore1;
112       --Altrimenti in tutti gli altri casi, eseguo:
113       else
114         --Ritorno in me stesso
115         next_state <= valore2;
116       end if;
117   end case;
118 end process;
119 end Behavioral;

```

Entity del contatore_auto

Una volta sintetizzato il codice del contatore, il risultato è stato il seguente componente, il quale è possibile osservarlo con la seguente RTL Schematic.



Sintesi

Invece, per quanto riguarda la sintesi degli stati possiamo apprezzare le informazioni che ci vengono fornite nella sezione “Synthesis Report”. I risultati sono i seguenti:

```
Synthesizing Unit <contatore_auto>.
Related source file is "D:\ProgettoTD_Traccia9_Canzolino_Gianluca\contatore_auto.vhd".
Found 2-bit register for signal <current_state>.
Found finite state machine <FSM_1> for signal <current_state>.
-----
| States           | 3 |
| Transitions      | 10 |
| Inputs           | 2 |
| Outputs          | 1 |
| Clock            | clk (rising_edge) |
| Reset            | rst (positive) |
| Reset type       | synchronous |
| Reset State      | valore0 |
| Power Up State   | valore0 |
| Encoding         | auto |
| Implementation   | LUT |
-----
Summary:
    inferred    1 Finite State Machine(s).
Unit <contatore_auto> synthesized.
```

Test Bench

Una volta terminata la stesura del codice, si è passata alla fase di Test Bench durante la quale sono state provate le varie combinazioni di ingressi per testare il corretto funzionamento di essa.

Per testare al meglio sono state fatte 15 diverse tipologie di simulazioni. Il codice del Test Bench è il seguente:

- La prima operazione è stato il reset della macchina.

```
101      -- Stimulus process
102      stim_proc: process
103      begin
104          -- Reset della macchina
105          rst <= '1';
106          wait for 20 ns;
107          rst <= '0';
108
109          wait for clk_period*2;
```

Una volta resettata la macchina, è possibile procedere con le varie simulazioni:

- 1) Caso in cui una macchina deve entrare e ci sono 2 posti disponibili:

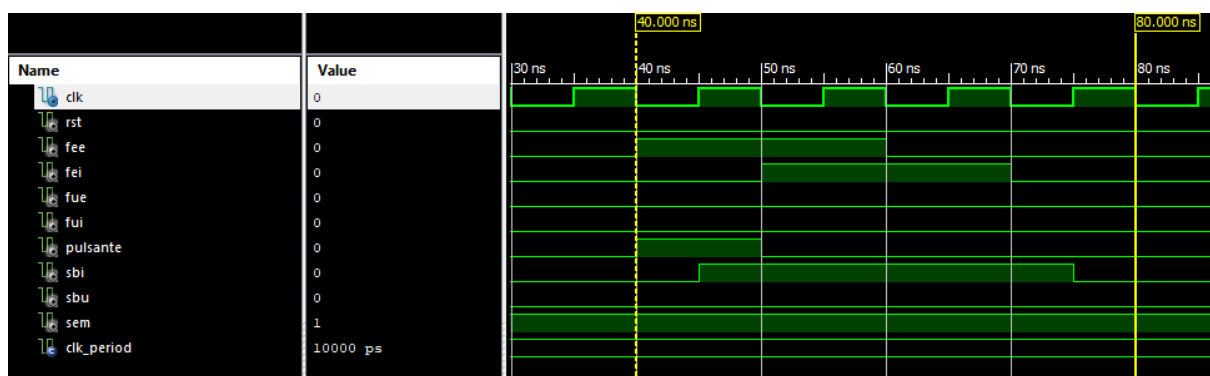
```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;
```

```
-- 2) l'auto sta entrando
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;
```

```
-- 3) l'auto è quasi entrata
fee <= '0'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;
```

```
-- 4) l'auto è entrata
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;
```

```
--Mi aspetto la sbarra che si richiude, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



2) Caso in cui una macchina deve entrare e c'è 1 posto disponibile:

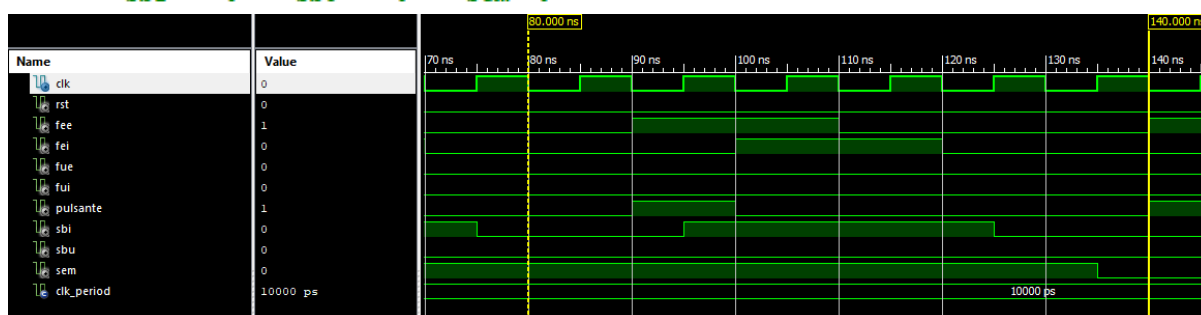
```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) l'auto sta entrando
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto è quasi entrata
fee <= '0'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 4) l'auto è entrata
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo diventa rosso, quindi:
--SbI = '0' - SbU = '0' - Sem '0'
```



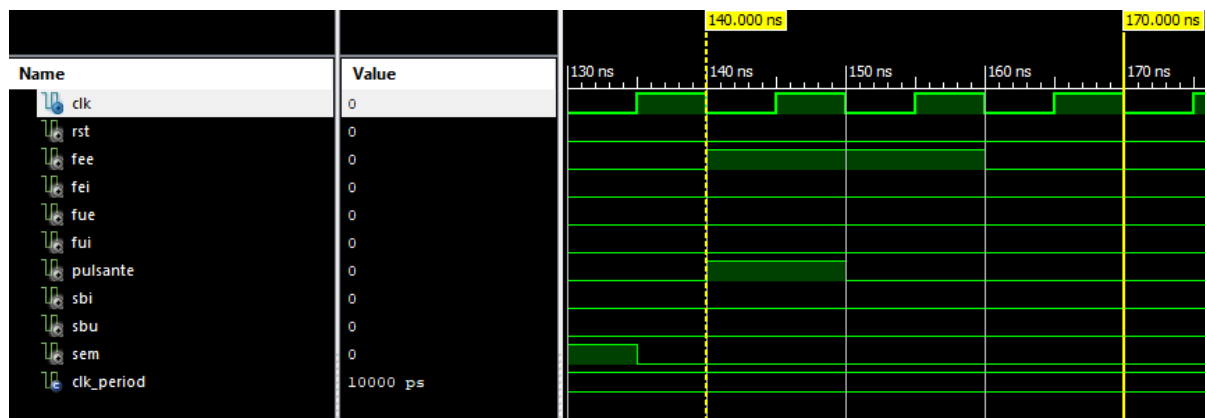
3) Caso in cui una macchina deve entrare ma non ci sono posti disponibili:

```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) l'auto è presente vicino alla fotocellula
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto rimane ad aspettare che si apra la sbarra
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra non si apre e infine che:
--SbI = '0' - SbU = '0' - Sem '0'
```



4) Caso in cui una macchina deve uscire e i posti sono tutti occupati:

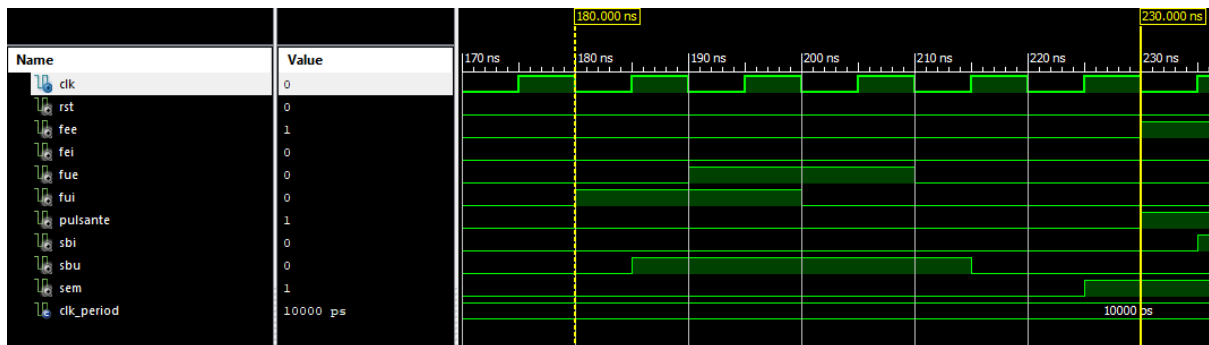
```
-- 1) l'auto è presente vicino alla fotocellula
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 2) l'auto sta uscendo
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto è quasi uscita
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 4) l'auto è uscita
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo diventa verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



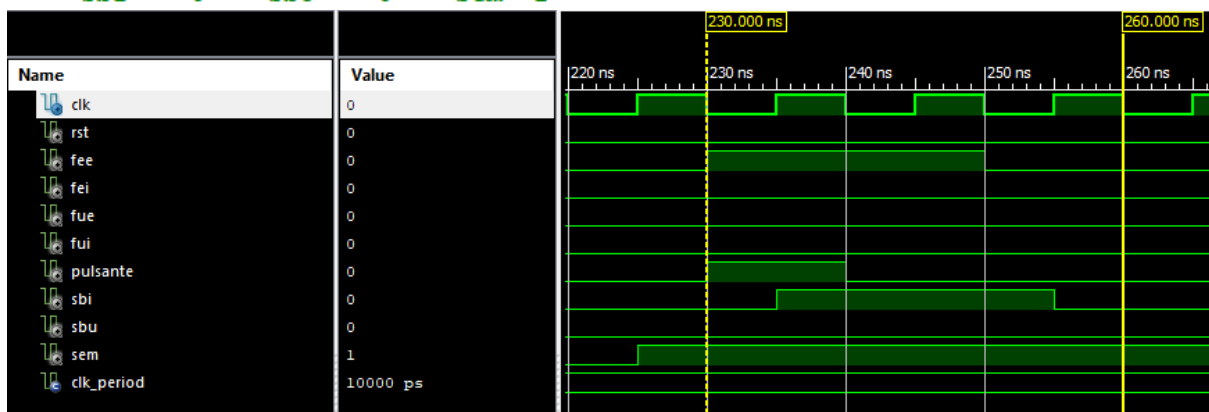
5) caso in cui una macchina deve entrare ma fa retromarcia:

```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) l'auto è presente vicino alla fotocellula
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto torna indietro
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



- 6) Caso in cui una macchina deve entrare, arriva a metà percorso ma fa retromarcia

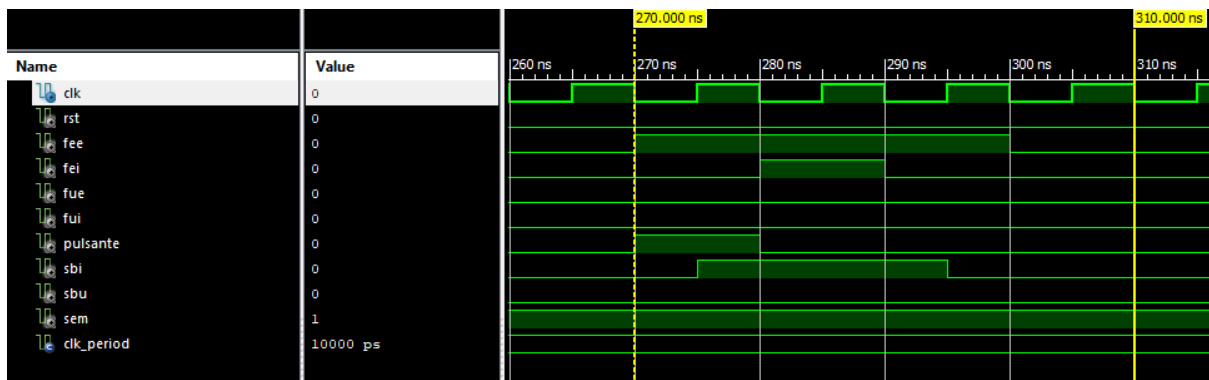
```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) l'auto sta entrando
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto torna indietro
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 4) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



- 7) Caso in cui una macchina deve entrare, è quasi entrata ma fa retromarcia:

```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

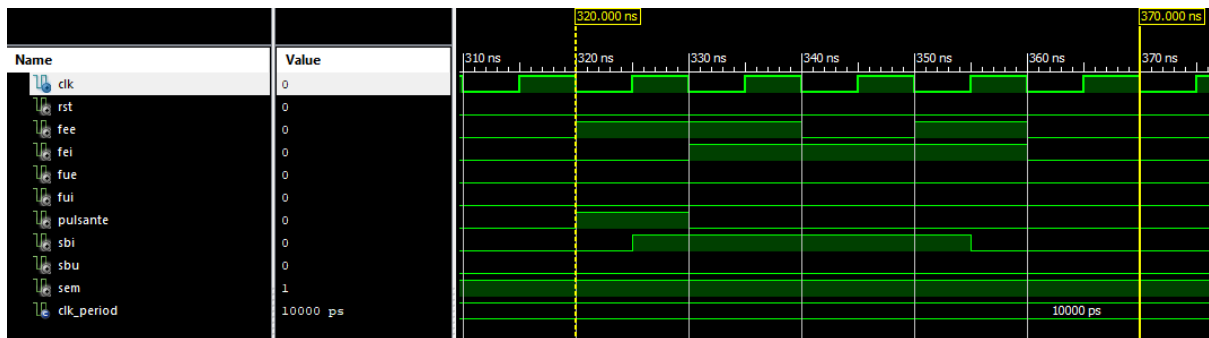
-- 2) l'auto sta entrando
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto è quasi entrata
fee <= '0'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 4) l'auto torna indietro
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 5) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



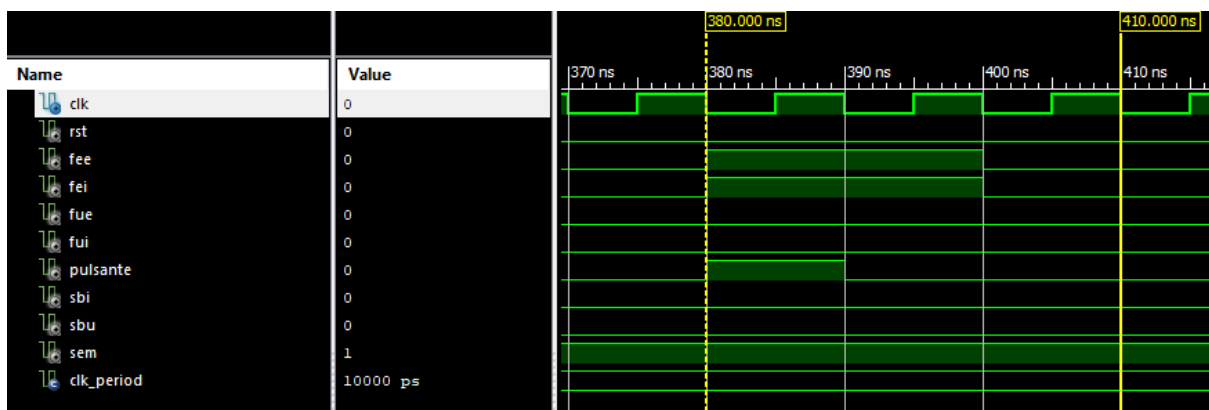
8) Caso in cui una macchina deve entrare, ma ci sta un ostacolo all'interno del parcheggio:

```
-- 1) l'auto è presente vicino alla fotocellula e preme il pulsante
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) l'auto aspetta che si apra la sbarra
fee <= '1'; fei <= '1'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra non si apra e che il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



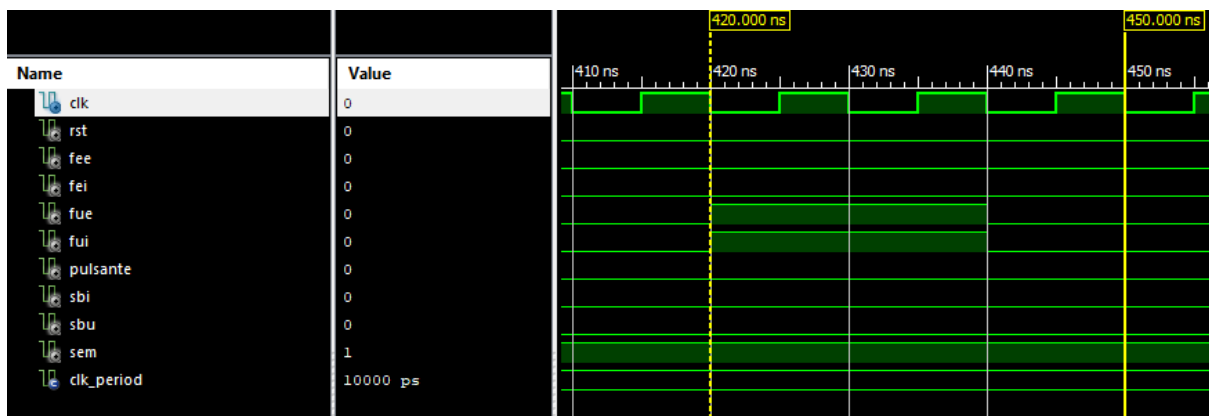
- 9) Caso in cui una macchina deve uscire, ma ci sta un ostacolo al di fuori del parcheggio:

```
-- 1) l'auto è presente vicino alla fotocellula
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 2) l'auto aspetta che si apre la sbarra
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 3) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra non si apra quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



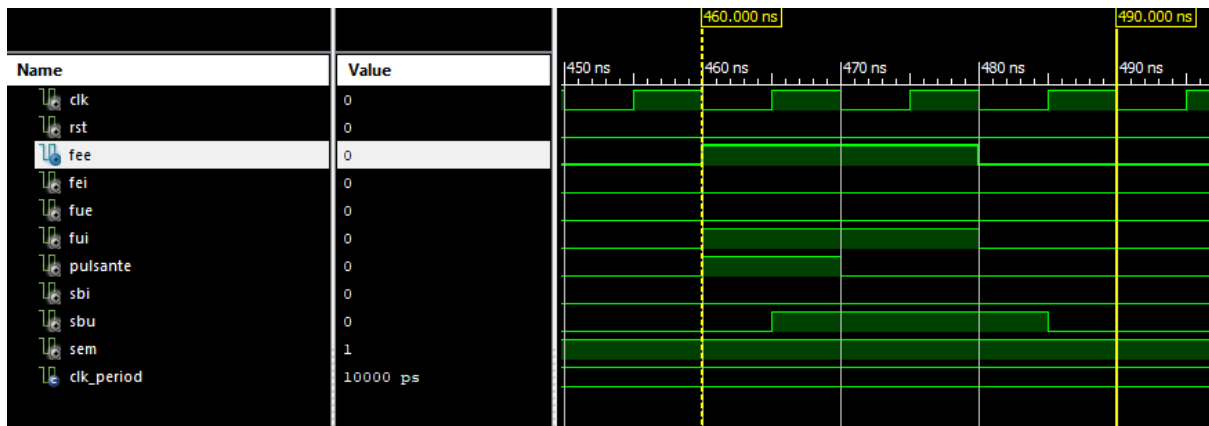
- 10) Caso in cui una macchina deve uscire e una deve entrare, contemporaneamente:

```
-- 1) le auto sono presenti vicino alle fotocellule
fee <= '1'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) le auto sono presenti vicino alle fotocellule
fee <= '1'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 3) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto che la sbarra di uscita si apra
--e quella di entrata non si apra quindi:
--SbI = '0' - SbU = '1' - Sem '1'
```



11) Caso in cui una persona preme il pulsante ma la fotocellula non rileva alcun'auto:

-- 1) la persona preme il pulsante

```
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;
```

-- 2) non ci sta nemmeno un'auto vicino alla fotocellula

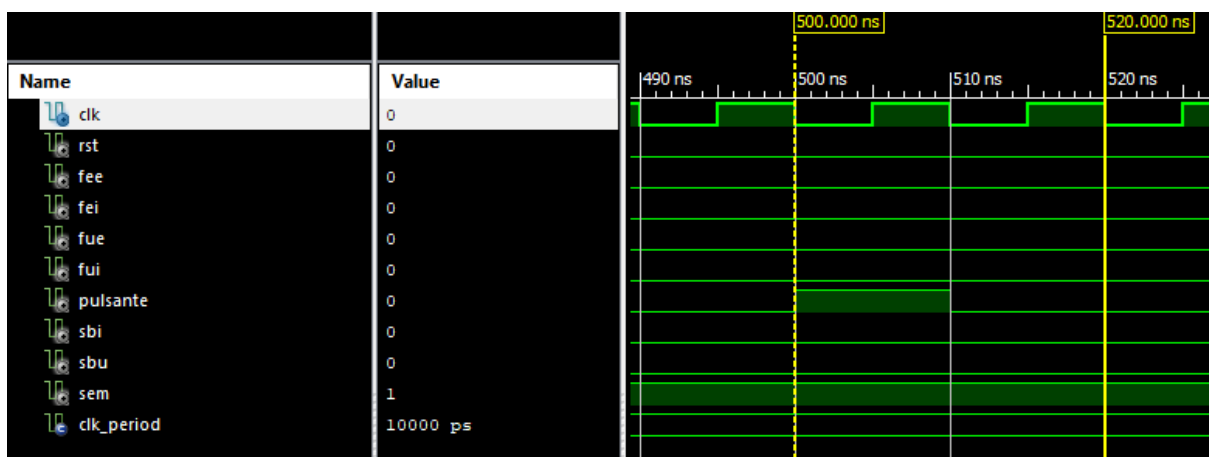
```
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;
```

-- 3) ingressi resettati

```
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;
```

--Mi aspetto che la sbarra di entrata non si apra quindi:

--SbI = '0' - SbU = '0' - Sem '1'



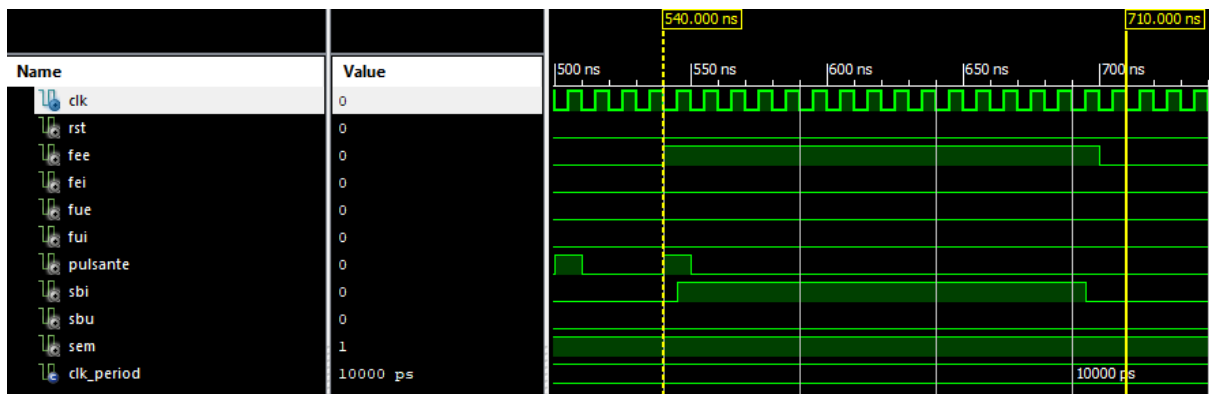
12) Caso in cui una macchina deve entrare, ma aspetta troppo tempo:

```
-- 1) 1'auto è presente vicino alla fotocellula
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '1';
wait for 10 ns;

-- 2) 1'auto aspetta
fee <= '1'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for clk_period*15;

-- 3) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto che la sbarra di entrata si apra e poi si richiuda quindi:
--SbI = '1' - SbU = '0' - Sem '1'
--SbI = '0' - SbU = '0' - Sem '1'
```

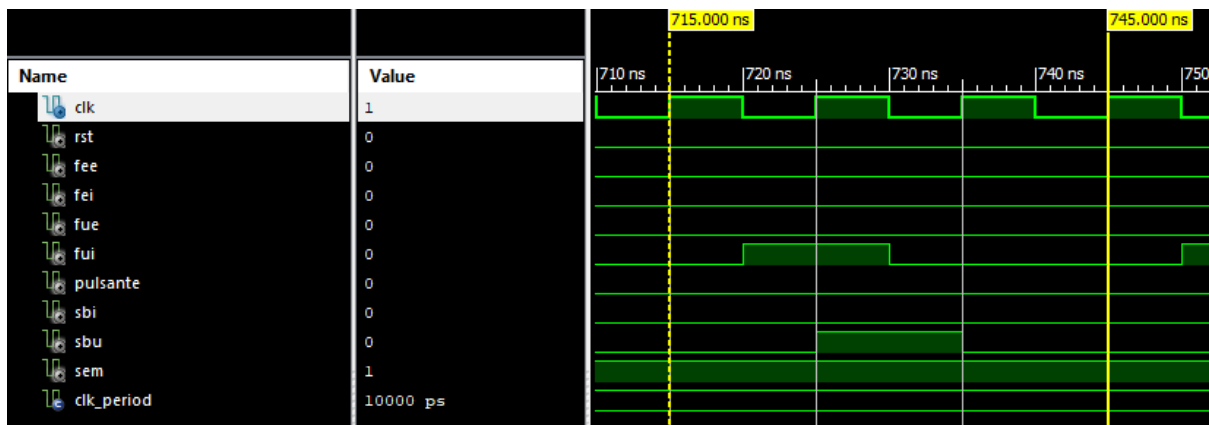


13) Caso in cui una macchina deve uscire ma fa retromarcia:

```
-- 1) 1'auto è presente vicino alla fotocellula
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 2) 1'auto torna indietro
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



14) Caso in cui una macchina deve uscire, arriva a metà percorso ma fa retromarcia:

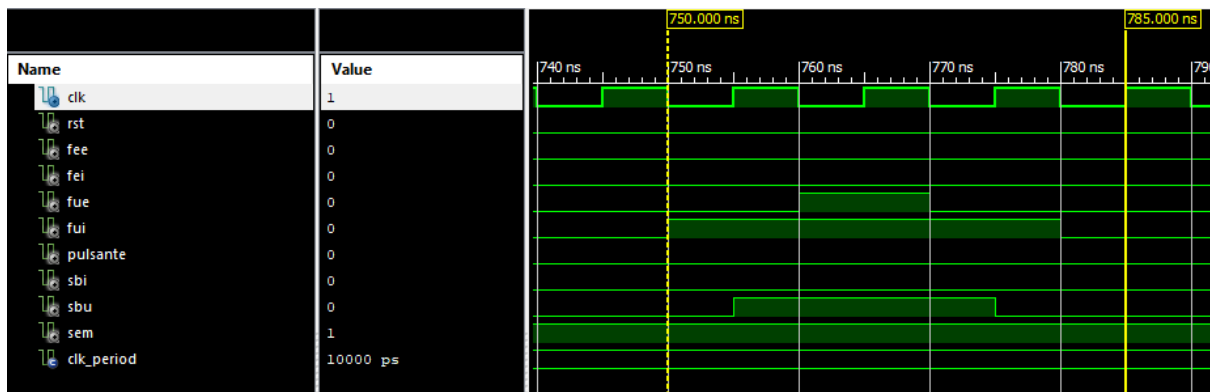
```
-- 1) l'auto è presente vicino alla fotocellula
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 2) l'auto sta uscendo
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto torna indietro
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

-- 4) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



15) Caso in cui una macchina deve uscire, è quasi uscita ma fa retromarcia:

```
-- 1) l'auto è presente vicino alla fotocellula
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '0'; Pulsante <= '0';
wait for 10 ns;

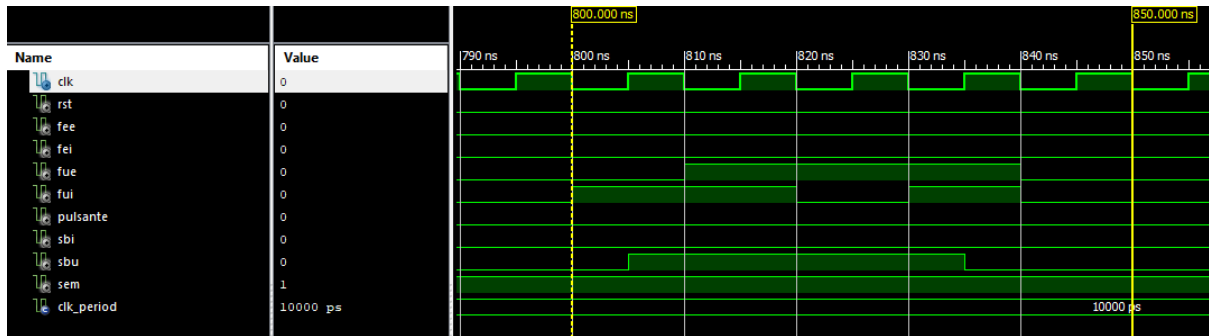
-- 2) l'auto sta uscendo
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 3) l'auto è quasi uscita
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 4) l'auto torna indietro
fee <= '0'; fei <= '0'; fui <= '1'; fue <= '1'; Pulsante <= '0';
wait for 10 ns;

-- 5) ingressi resettati
fee <= '0'; fei <= '0'; fui <= '0'; fue <= '0'; Pulsante <= '0';
wait for 20 ns;

--Mi aspetto la sbarra che si richiude il semaforo rimane verde, quindi:
--SbI = '0' - SbU = '0' - Sem '1'
```



Post-Route

La sintesi Post-Route è stata implementata anch'essa con risultati soddisfacenti. Dato che il codice è stato implementato senza usare latch, i ritardi reali nella propagazione del segnale sono minimi e dipendono soltanto dalla loro implementazione sulla scheda FPGA (il ritardo è possibile osservarlo come riportato dal sintetizzatore) evitando quindi stranezze durante la simulazione dello stesso.

Di seguito sono riportati tutti i tempi massimi e minimi di propagazione del segnale delle uscite, degli ingressi e del clock stesso.

Source	Max Setup to clk (edge)	Process Corner	Max Hold to clk (edge)	Process Corner	Internal Clock(s)	Clock Phase
FEE	2.663 (R)	SLOW	0.092 (R)	SLOW	clk_BUFPG	0.000
FEI	2.578 (R)	SLOW	-0.411 (R)	SLOW	clk_BUFPG	0.000
FUE	1.481 (R)	SLOW	0.091 (R)	SLOW	clk_BUFPG	0.000
FUI	1.644 (R)	SLOW	0.062 (R)	SLOW	clk_BUFPG	0.000
Pulsante	1.441 (R)	SLOW	-1.061 (R)	SLOW	clk_BUFPG	0.000
rst	1.516 (R)	SLOW	0.456 (R)	SLOW	clk_BUFPG	0.000

Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min (fastest) clk (edge) to PAD	Process Corner	Internal Clock(s)	Clock Phase
SbI	9.114 (R)	SLOW	6.712 (R)	FAST	clk_BUFPG	0.000
SbU	9.574 (R)	SLOW	6.679 (R)	FAST	clk_BUFPG	0.000
Sem	8.752 (R)	SLOW	6.350 (R)	FAST	clk_BUFPG	0.000

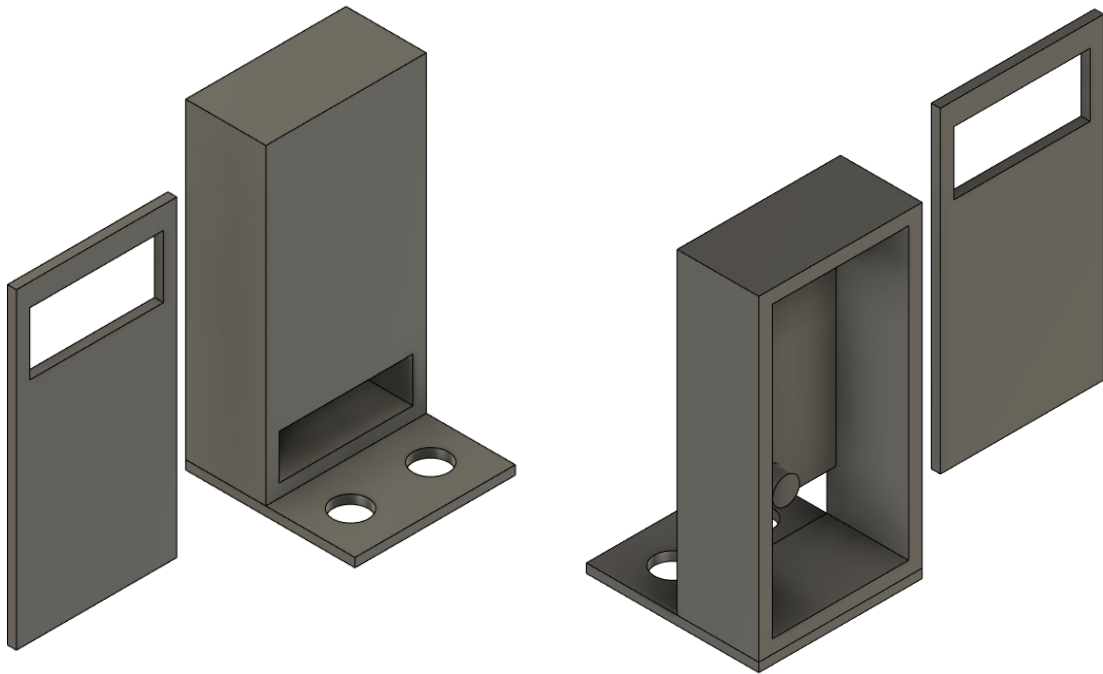
Source Clock	Src:Rise	Src:Fall	Src:Rise	Src:Fall	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall
clk	2.669							

Progetto fisico

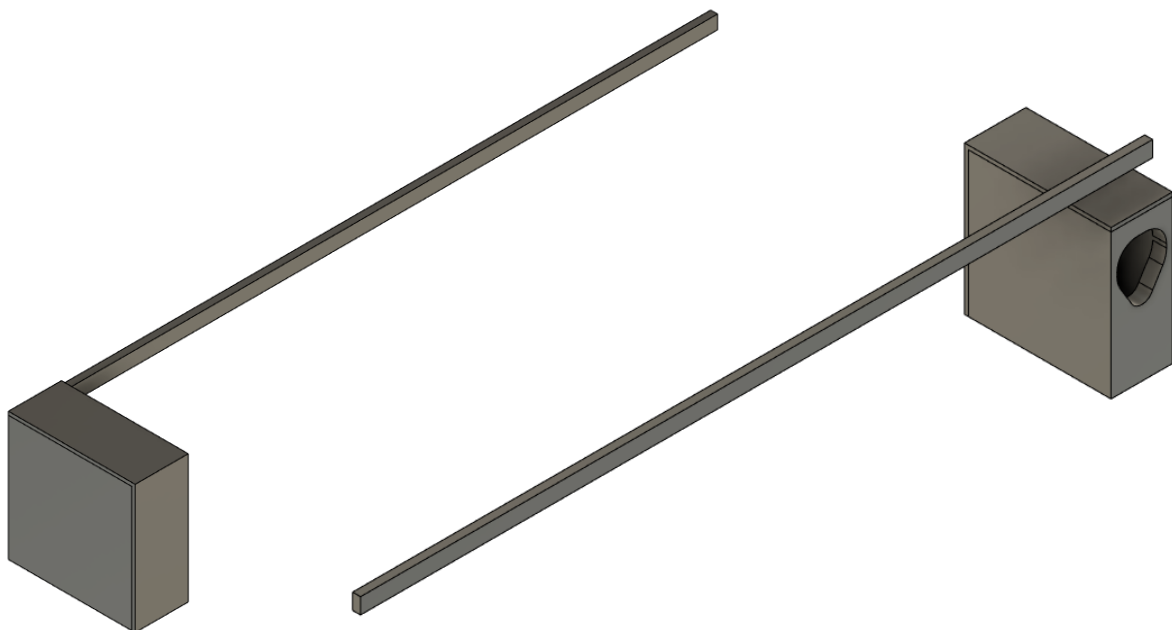
Oltre alle simulazioni fatte tramite il software di Xilinx, è stato implementato l'intero progetto in scala ridotta tramite un modellino fisico. Esso prima di realizzarlo fisicamente, si è passato ad una progettazione 3D di ogni singolo componente e infine dell'intero modello tramite l'ausilio del software ***Fusion360***.

I componenti sono i seguenti:

- Fotocellula (generica);



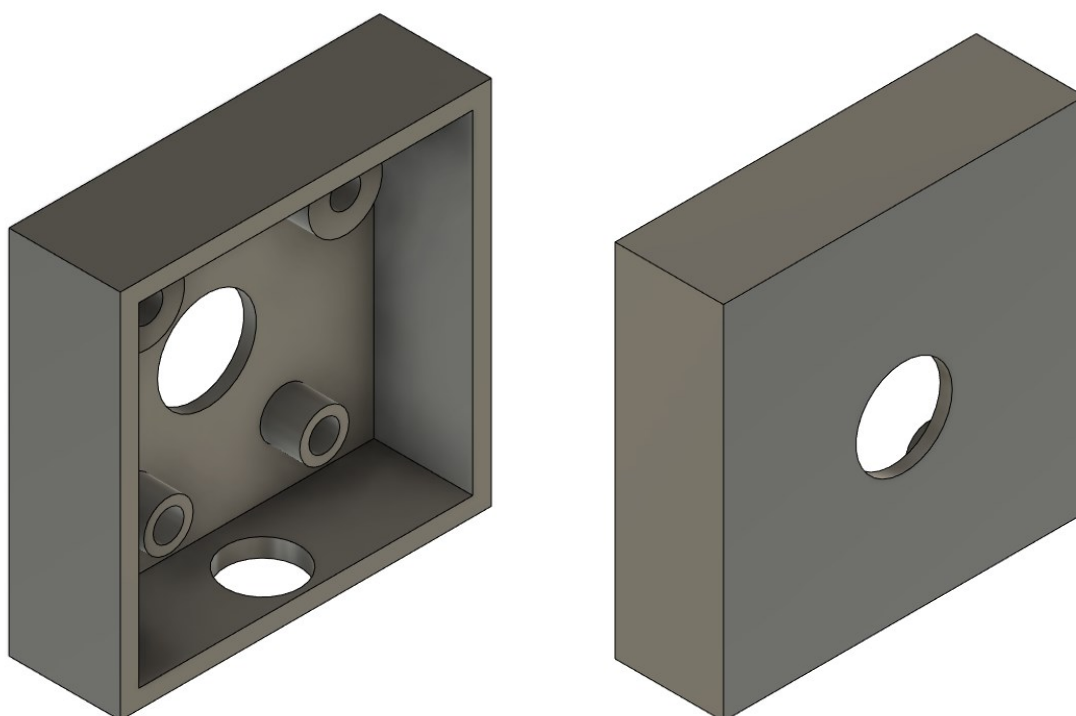
- Sbarra (generica);



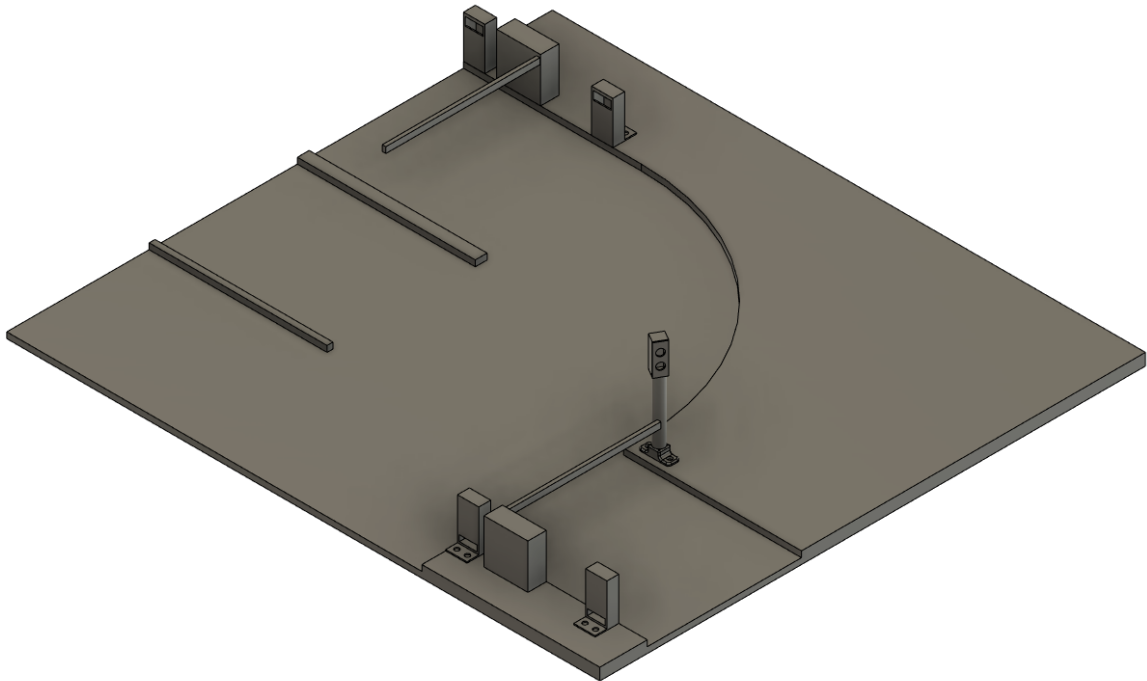
- Semaforo;



- Pulsante.

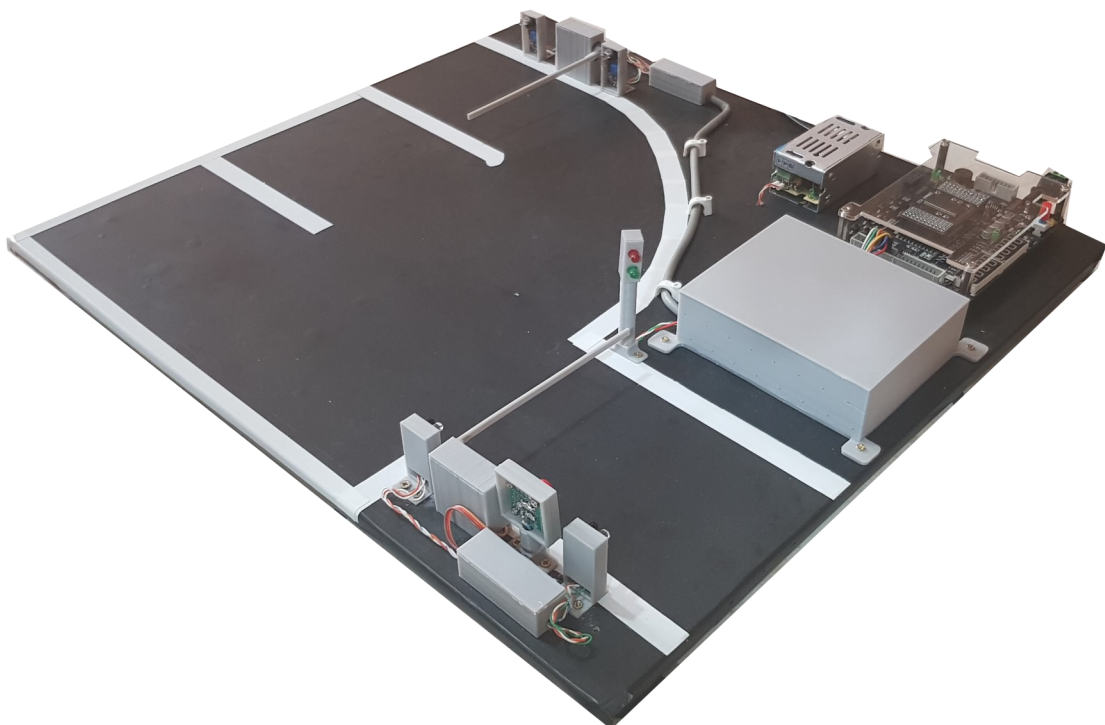


Modellino completo 3D



Modellino completo reale

L'idea iniziale per la realizzazione dello stesso è stata quella di utilizzare una base di legno rivestita di carta adesiva nera per imitare il colore dell'asfalto e del nastro adesivo bianco per imitare le linee di corsia.



Componenti utilizzati

Ogni componente è stato collegato alla scheda Arduino Nano tramite una scheda millefori completamente realizzata a mano, senza ricorrere allo sviluppo di una PCB industriale, dato che è soltanto un prototipo non si è optato per quella scelta.

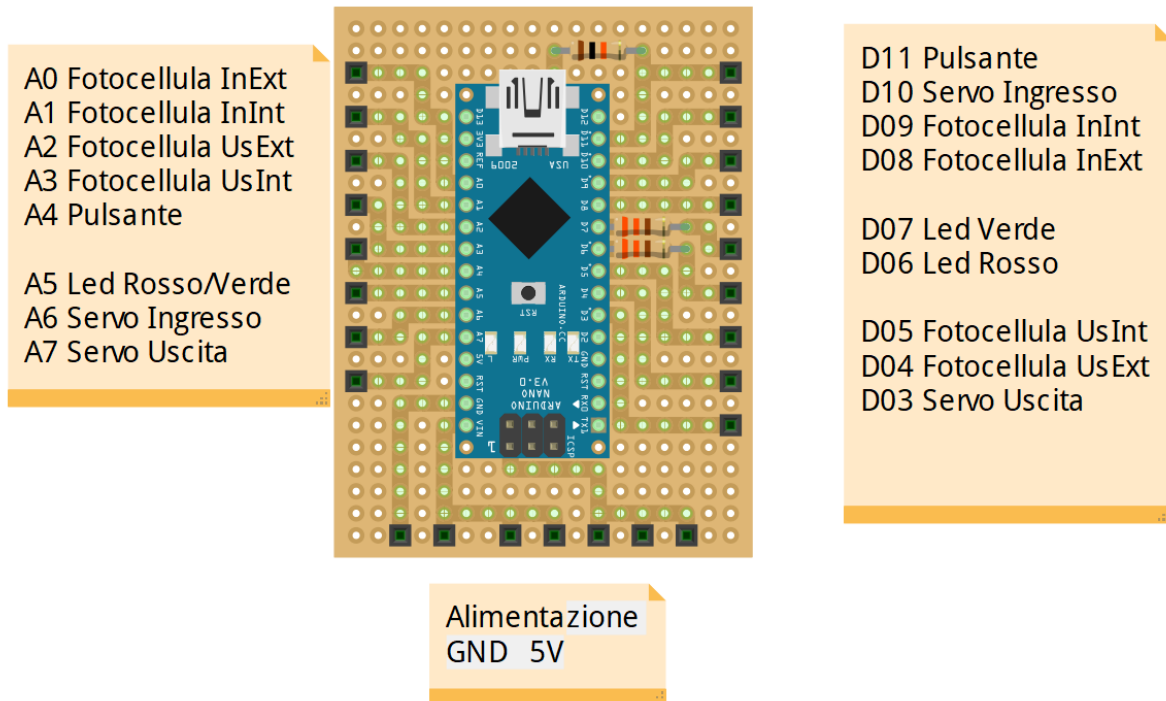
I componenti utilizzati sono:

Numero materiale	Descrizione	Quantità
1	Tavole di legno	n.d.
2	Alimentatore 5V	1
3	Arduino Nano	1
4	Xilinx Spartan6 XC6SLX9 TQG144	1
5	Schede di prototipazione (millefori)	3
6	Led rosso	1
7	Led verde	1
8	Resistori 10 K Ω	1
9	Resistori 330 Ω	2
10	Servomotori	2
11	Sensori di prossimità	4
12	Pulsante	1
13	Connettori e cavi	n.d.

Circuiti

Il circuito è estremamente semplice poiché sono tutti componenti acquistati da terzi. L'unico circuito implementato è quello della scheda Arduino Nano sulla quale ci sono le varie morsettiere per i pin di ingresso e di uscita.

Il circuito è il seguente:



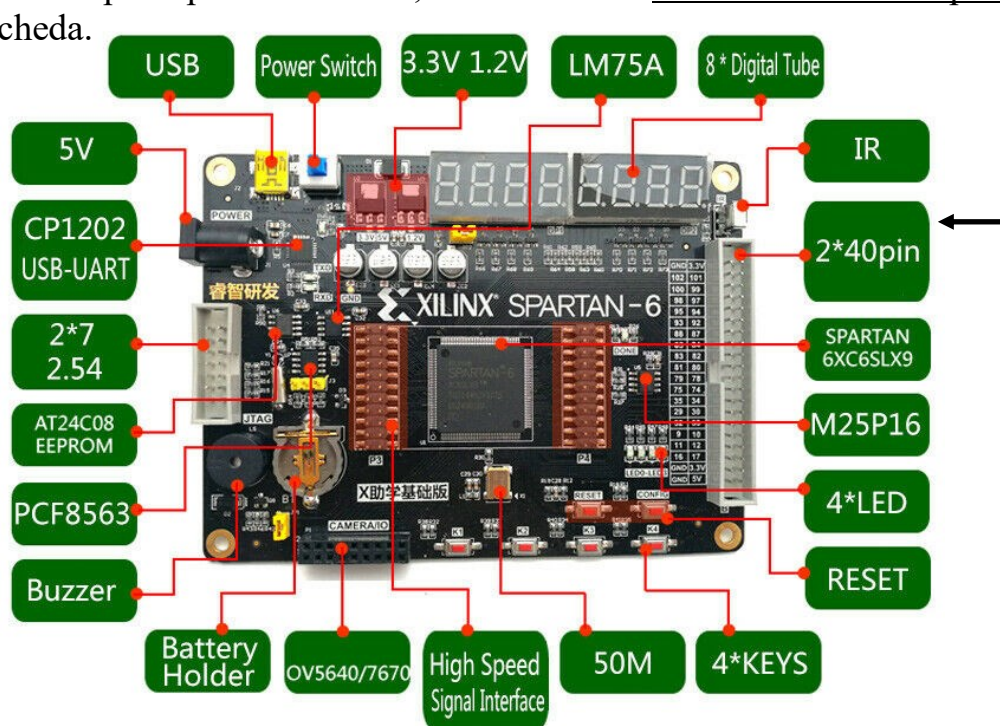
Pinout

Ogni componente è stato collegato alla scheda Arduino Nano seguendo il pinout descritto nell'immagine in alto. Possiamo identificare sulla destra di Arduino Nano i pin Digitali, i quali lavorano su valori digitali (come appunto il nome), ovvero su tensioni di 0V o 5V. A sinistra invece, ci sono i pin Analogici, i quali possono lavorare su valori che variano in un range tra 0V e 5V (sono stati usati questi pin proprio per avere un'uscita di 3.3V adatta alla FPGA). In basso, invece, ci sono i pin di alimentazione (5V e GND).

Per quanto riguarda la Xilinx Spartan6 XC6SLX9 TQG144, i pin non sono stati scelti dal software ISE WebPack, ma sono stati scelti in maniera specifica usando un software secondario (sempre appartenente a Xilinx), il quale è PlanAhead. Questo software permette un'ampia scelta per ogni tipo di dettaglio. In particolare, sono stati impostati i pin della scheda secondo il seguente schema:

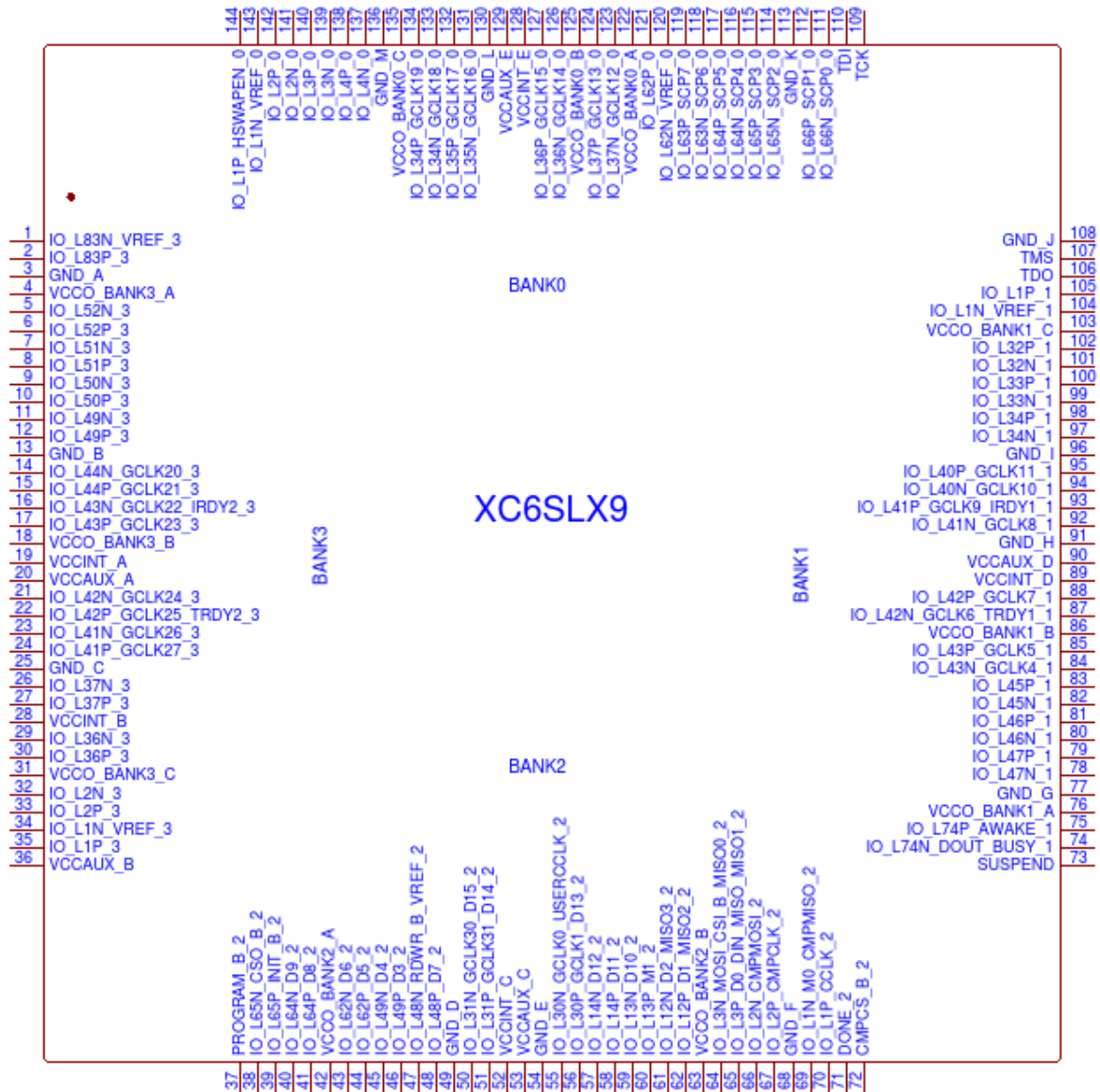
Name	Dir	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type	Off-Chip T...	IN_TERM	OUT_TERM
All ports (10)															
Scalar ports (10)															
clk		Input		P22	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
FEE		Input		P11	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
FEI		Input		P12	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
FUE		Input		P29	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
FUI		Input		P30	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
Pulsante		Input		P9	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
rst		Input		P27	<input checked="" type="checkbox"/>	3	LVCNMOS33*					NONE	NONE	NONE	
SbI		Output		P32	<input checked="" type="checkbox"/>	3	LVCNMOS33*	3.300		12 SLOW	NONE	FP_VTT_50		NONE	
SbU		Output		P33	<input checked="" type="checkbox"/>	3	LVCNMOS33*	3.300		12 SLOW	NONE	FP_VTT_50		NONE	
Sem		Output		P35	<input checked="" type="checkbox"/>	3	LVCNMOS33*	3.300		12 SLOW	NONE	FP_VTT_50		NONE	

Le motivazioni sono varie, ad esempio per il clock è stato scelto il pin 22 (e non 21 come suggerito dall'ISE WebPack) poiché la scheda su cui risiede il chip, ha predisposto un clock (utilizzando un quarzo) di 50MHz (20 ns) proprio su quel pin. Gli altri pin sono stati scelti per praticità nel collegare nella maniera più semplice possibile i cavi, sfruttando dei connettori maschio presenti sulla scheda.



Sul chip, anche se non si hanno gli schemi della scheda acquistata, si possono identificare tutti i pin, con tutte le loro caratteristiche. In seguito viene riportata in grande la codifica dei pin e una parte del datasheet del chip.

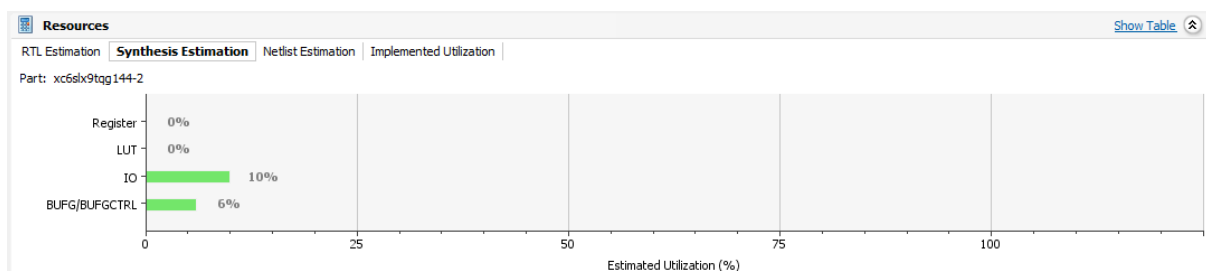
clk	Input	P22
FEE	Input	P11
FEI	Input	P12
FUE	Input	P29
FUI	Input	P30
Pulsante	Input	P9
rst	Input	P27
SbI	Output	P32
SbU	Output	P33
Sem	Output	P35



Successivamente alla codifica dei pin, il codice è stato caricato sulla flash (in questa scheda di 16MB) tramite un altro software chiamato Impact (sempre della Xilinx).

Attenzione! Dato che non si è voluto modificare il concetto dei 15 colpi di clock, nel codice caricato, la parte inerente a ciò è stata commentata.

Dopo caricato il codice, sono stati riportati i seguenti report:





Report utilization



Estimated resources are compared with xc6slx9tqg144-2. Note that these are pre-map netlist estimations.

[Show More Details](#)



Register

Available:  34320
Estimation:  11 (<1% of available) Automa_principale



LUT

Available:  17160
Estimation:  19 (<1% of available) Automa_principale

Global Clock Buffer

Available:  16
Estimation:  1 (6% of available) Automa_principale

IO

Available:  102
Estimation:  10 (10% of available) Automa_principale

Codice Arduino

Oltre al codice sviluppato per la FPGA, è stato sviluppato un ulteriore codice per la scheda Arduino Nano il quale è:

```
#include <Servo.h>

Servo servo_ingresso, servo_uscita;
#define pulsante 11
#define servo_ing 10
#define fotocellula_ingresso_interna 9
#define fotocellula_ingresso_esterna 8
#define led_verde 7
#define led_rosso 6
#define fotocellula_uscita_interna 5
#define fotocellula_uscita_esterna 4
#define servo_usc 3

#define FPGA_pulsante A4
#define FPGA_servo_ing A6
#define FPGA_fotocellula_ingresso_interna A1
#define FPGA_fotocellula_ingresso_esterna A0
#define FPGA_led A5
#define FPGA_fotocellula_uscita_interna A3
#define FPGA_fotocellula_uscita_esterna A2
#define FPGA_servo_usc A7

bool
  FIE=false,
  FII=false,
  FUE=false,
  FUI=false;

const int
  servo_ing_aperto = 0,
  servo_ing_chiuso = 95,
  servo_usc_aperto = 0,
  servo_usc_chiuso = 100;

void setup() {
  pinMode(FPGA_pulsante, OUTPUT);
  pinMode(FPGA_servo_ing, INPUT);
  pinMode(FPGA_fotocellula_ingresso_interna, OUTPUT);
  pinMode(FPGA_fotocellula_ingresso_esterna, OUTPUT);
  pinMode(FPGA_led, INPUT);
  pinMode(FPGA_fotocellula_uscita_interna, OUTPUT);
  pinMode(FPGA_fotocellula_uscita_esterna, OUTPUT);
  pinMode(FPGA_servo_usc, INPUT);

  pinMode(pulsante, INPUT);
  servo_ingresso.attach(servo_ing);
  pinMode(fotocellula_ingresso_interna, INPUT);
```



```

pinMode(fotocellula_ingresso_esterna, INPUT);
pinMode(led_verde, OUTPUT);
pinMode(led_rosso, OUTPUT);
pinMode(fotocellula_uscita_interna, INPUT);
pinMode(fotocellula_uscita_esterna, INPUT);
servo_uscita.attach(servo_usc);
servo_ingresso.write(servo_ing_chiuso);
servo_uscita.write(servo_usc_chiuso);
Serial.begin(9600);
}

void loop() {
  if(analogRead(FPGA_led) > 300)
  {
    digitalWrite(led_verde, HIGH);
    digitalWrite(led_rosso, LOW);
  }
  else
  {
    digitalWrite(led_verde, LOW);
    digitalWrite(led_rosso, HIGH);
  }
  Serial.println(analogRead(FPGA_led));

  if(analogRead(FPGA_servo_ing) > 300)
  {
    servo_ingresso.write(servo_ing_aperto);
  }
  else
  {
    servo_ingresso.write(servo_ing_chiuso);
  }

  if(analogRead(FPGA_servo_usc) > 300)
  {
    servo_uscita.write(servo_usc_aperto);
  }
  else
  {
    servo_uscita.write(servo_usc_chiuso);
  }

  if(digitalRead(fotocellula_ingresso_esterna) != FIE || digitalRead(fotocellula_ingresso_interna) !=
  FII || digitalRead(fotocellula_uscita_esterna) != FUE || digitalRead(fotocellula_uscita_interna) != FUI)
  {
    aggiorna_dati();
    if(FIE)
    {
      analogWrite(FPGA_fotocellula_ingresso_esterna, 650);
    }
    else
    {
      analogWrite(FPGA_fotocellula_ingresso_esterna, 0);
    }
    if(FII)
    {
      analogWrite(FPGA_fotocellula_ingresso_interna, 650);
    }
    else
    {
      analogWrite(FPGA_fotocellula_ingresso_interna, 0);
    }
  }
}

```

```

    if(FUE)
    {
        digitalWrite(led_verde, HIGH);
        analogWrite(FPGA_fotocellula_uscita_esterna, 650);
    }
    else
    {
        digitalWrite(led_verde, LOW);
        analogWrite(FPGA_fotocellula_uscita_esterna, 0);
    }
    if(FUI)
    {
        Serial.println(analogRead(FPGA_led));
        digitalWrite(led_rosso, HIGH);
        analogWrite(FPGA_fotocellula_uscita_interna, 650);
    }
    else
    {
        if(analogRead(FPGA_servo_ing) > 300)
        {
            digitalWrite(led_rosso, LOW);
            analogWrite(FPGA_fotocellula_uscita_interna, 0);
        }
        else
        {
            digitalWrite(led_rosso, HIGH);
        }
    }
    testa_pulsante();
    delay(2);
}

bool stato_pulsante = false;
void testa_pulsante()
{
    int media=0;
    for(int i=0; i<100; i++)
    {
        media += digitalRead(pulsante);
        delay(2);
    }

    if(media>70)
    {
        if(stato_pulsante == false)
        {
            analogWrite(FPGA_pulsante, 650);
            delay(100);
            analogWrite(FPGA_pulsante, 0);
        }
        stato_pulsante=true;
    }
    else
    {
        stato_pulsante=false;
    }
}

int media_FIE,media_FII,media_FUE,media_FUI;

void aggiorna_dati()
{
    media_FIE=0;
    media_FII=0;
    media_FUE=0;
    media_FUI=0;
}

```

```

int media_FIE,media_FII,media_FUE,media_FUI;

void aggiorna_dati()
{
    media_FIE=0;
    media_FII=0;
    media_FUE=0;
    media_FUI=0;

    for(int i=0; i<100; i++)
    {
        media_FIE += digitalRead(fotocellula_ingresso_esterna);
        media_FII += digitalRead(fotocellula_ingresso_interna);
        media_FUE += digitalRead(fotocellula_uscita_esterna);
        media_FUI += digitalRead(fotocellula_uscita_interna);
        delay(5);
    }
    if(media_FIE < 70)
    {
        FIE = true;
    }
    else
    {
        FIE = false;
    }
    if(media_FII < 70)
    {
        FII = true;
    }
    else
    {
        FII = false;
    }
    if(media_FUE < 70)
    {
        FUE = true;
    }
    else
    {
        FUE = false;
    }
    if(media_FUI < 70)
    {
        FUI = true;
    }
    else
    {
        FUI = false;
    }
}

```