Relazione dell’elaborato di Architettura degli Elaboratori

A.A. 2019/2020 Studenti:

Samantha Vanini VR443381
Marco Massagrande VR446285

Sommario

Architettura generale del circuito

Il dispositivo da noi implementato sarebbe un casello per la gestione di un parcheggio con ingresso/uscita automatizzati. Il parcheggio è suddiviso in 3 settori: i settori A e B hanno 31 posti macchina ciascuno, mentre il settore C ha 24 posti macchina. Al momento dell'ingresso l'utente deve dichiarare in quale settore vuole parcheggiare, analogamente al momento dell'uscita l'utente deve dichiarare da quale settore proviene. Il parcheggio rimane libero durante la notte, permettendo a tutte le macchine di entrare e uscire a piacimento.

Il circuito presenta i seguenti ingressi:

- **IN/OUT [2]:** se l'utente è in ingresso il sistema riceve in input la codifica 01, nel caso sia in uscita riceve la codifica 10. Codifiche 00 e 11 vanno interpretate come anomalie di sistema e quella richiesta va ignorata(ovvero non va aperta alcuna sbarra).
- **SECTOR [3]:** i settori sono indicati con codifica one-hot, ovvero una stringa di 3 bit in cui uno solo assume valore 1 e tutti gli altri 0. La codifica sarà pertanto 100-A, 010-B, 001-C. Codifiche diverse da queste vanno interpretate come errori di inserimento e la richiesta va ignorata.

Presenta inoltre le seguenti uscite:

- **SBARRA_(IN/OUT) [1]:** questo bit assume valore 0 se la sbarra è chiusa, 1 se viene aperta. La sbarra rimane aperta per un solo ciclo di clock, dopo di che viene richiusa (anche se la richiesta al ciclo successivo è invalida).
- **SECTOR_(A/B/C) [1]:** questo bit assume valore 1 se tutti i posti macchina nel settore sono occupati, 0 se ci sono ancora posti liberi.

Il controllore è collegato al datapath con tre segnali che hanno il seguente significato:

- **OP_(A/B/C)[2]:** indica se l'auto è in ingresso oppure in uscita e fa scegliere al datapath quale risultato salvare nel registro.
- **MEM_(A/B/C)[1]:** indica se selezionare il valore interno al registro del datapath oppure utilizzare il valore immesso dall'esterno
- **POSTO_(A/B/C)[1]:** indica alla FSM se il parcheggio è al completo (1) oppure no (0)

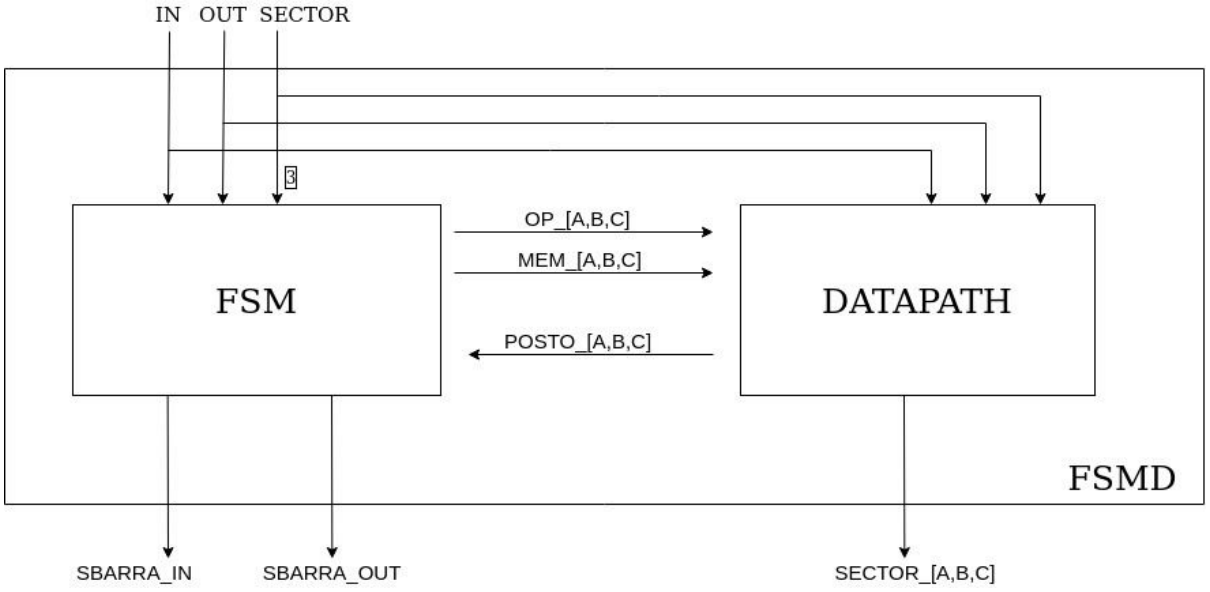


Diagramma degli stati del controllore

Il controllore è una macchina a stati finiti (FSM) di Mealy che presenta 8 ingressi **(IN, OUT, SECTOR_A, SECTOR_B, SECTOR_C, Posto_A, Posto_B, Posto_C**) e 11 uscite (**SBARRA_IN, SBARRA_OUT, MEM_A, MEM_B, MEM_C, OP_A1, OP_A0, OP_B1, OP_B0, OP_C1, OP_C0**).

RESET : E' lo stato di reset, la FSM rimane in questo stato fino a quando non viene inserito in input il codice di attivazione 11111 e si spegne, ovvero inizia la gestione notturna, quando lo stato AUTO riceve in input 00000

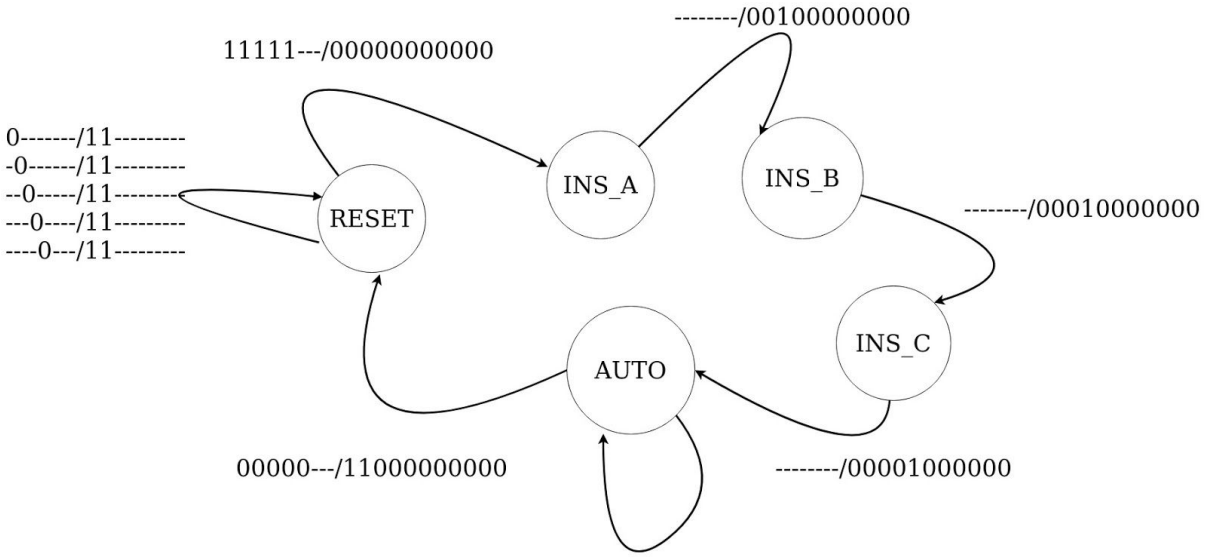
INS_A : E' lo stato in cui il il sistema attende l'inserimento dei posti occupati nel settore A.

INS_B : E' lo stato in cui il il sistema attende l'inserimento dei posti occupati nel settore B.

INS_C : E' lo stato in cui il il sistema attende l'inserimento dei posti occupati nel settore C.

AUTO : E' lo stato di gestione automatica della macchina che riceve in ingresso la posizione in cui si trova l'auto ed il relativo parcheggio. La FSM ricevuto ingresso 00000 si resetta.

Ecco la rappresentazione del nostro controllore :



011000--/10000010000
01010-0-/10000000100
01001--0/10000000001

011001--/00000000000
01010-1-/00000000000
01001--1/00000000000

10100---/01000100000
10010---/01000001000
10001---/01000000010

11------/00000000000
--11----/00000000000
--1-1---/00000000000
---11---/00000000000
001-----/00000000000
00-1----/00000000000
00--1---/00000000000
1-000---/00000000000
-1000---/00000000000

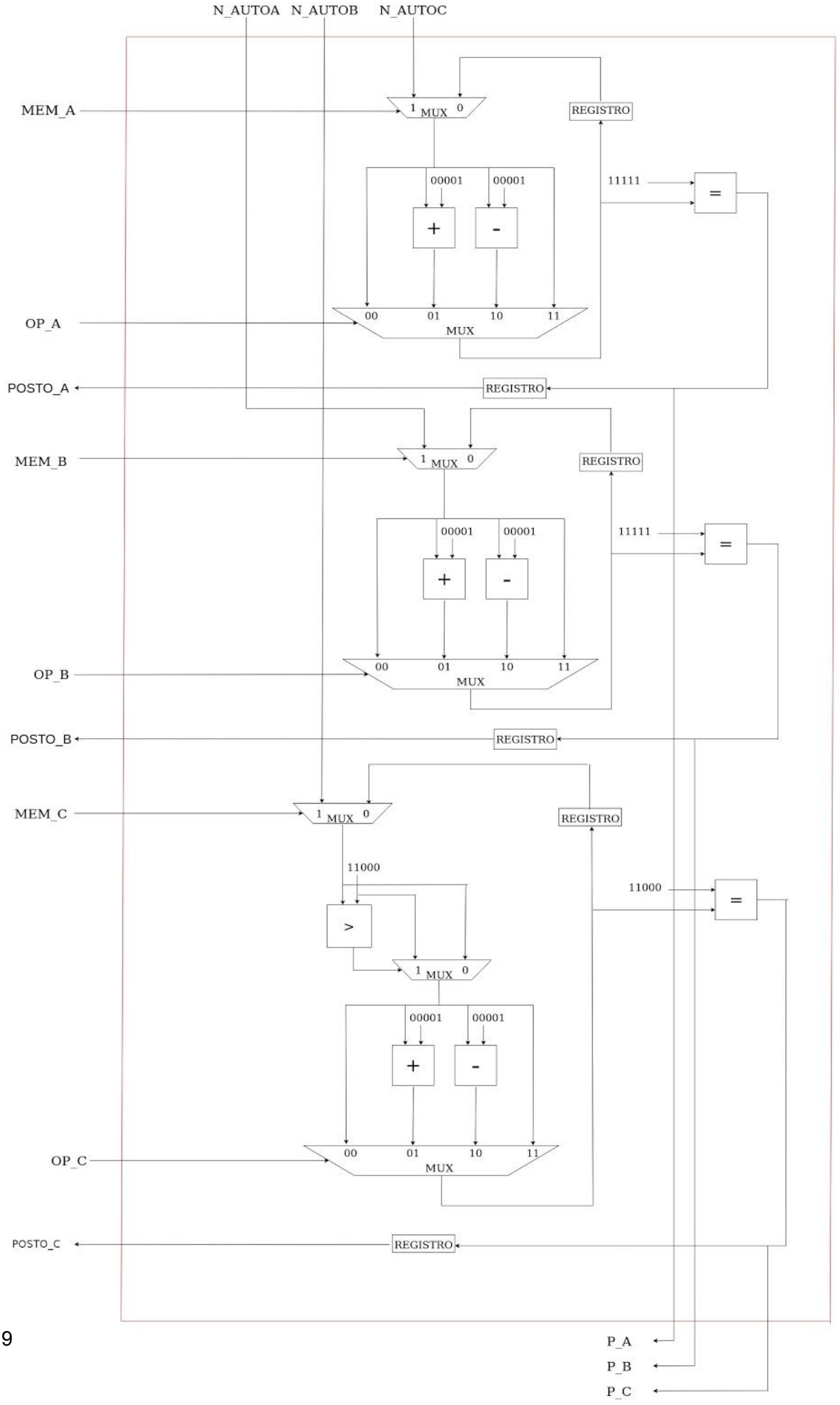
Architettura del datapath

Il datapath è quella parte del circuito che si occupa principalmente di eseguire calcoli.
Nel nostro caso riceve come input esterno il numero di auto presenti all'inizio della gestione (**N_A [5 bit]**), mentre dalla FSM riceve i settori in cui inserire i posti occupati all'accensione della macchina (**MEM_(A/B/C) [1]**) e l'operazione da compiere se l'utente desidera entrare oppure uscire (**OP_(A/B/C) [2]**).Genera come output **POSTO_(A/B/C) [1]** che restituisce alla FSM 1 se il parcheggio è al completo, 0 altrimenti e **P_(A/B/C) [1]** che restituisce come output del datapath 1 se il parcheggio è al completo, 0 altrimenti.

Il circuito riguardante l'elaborazione è stato concepito come 3 unità modulari, ognuna dedicata alla gestione ed al conteggio dei posti occupati di un singolo settore, collegate tra loro dal segnale di clock dei registri utilizzati per conservare il totale dei posti occupati e lo stato del parcheggio.
Il datapath riceve in ingresso dalla FSM **MEM_(A/B/C)[1]**, segnale di selezione del multiplexer a due ingressi iniziale, che si occupa di scegliere se utilizzare per i conteggi il valore inserito da input esterno oppure il valore presente all'interno del registro dopo le operazioni precedenti. Durante i primi 3 cicli di clock il segnale di selezione sarà posto a 1 in base al parcheggio in cui inserire il numero di auto presenti. Collegati al multiplexer si trovano un componente che esegue la somma ed un componente che esegue la sottrazione. La somma avviene tra il valore scelto dal multiplexer e la costante 1, così come la sottrazione. Il componente che calcola la differenza tra due valori è stato realizzato considerando come si calcola il reciproco di un numero in complemento a 2. I risultati delle operazioni precedentemente descritte vengono collegati, assieme al valore uscito dal primo selettore, ad un multiplexer a 4 ingressi, che usa come selettore il segnale **OP_(A/B/C)[2]** proveniente dalla FSM. Così facendo si riescono ad ignorare le codifiche diverse da quelle specificate. Il valore così ottenuto viene confrontato con la capienza massima del parcheggio selezionato, in modo da poter

dare come output esterno **P_(A/B/C)[1]**, che nella FSMD viene chiamato **SECTOR_(A/B/C)[1]**, e come output verso la FSM **POSTO_(A/B/C)[1]** che, dopo essere passato attraverso un registro,indica al controllore lo stato del parcheggio.
Nel modulo riguardante il parcheggio C è stato aggiunto, dopo la selezione del valore su cui compiere i conteggi, un comparatore tra il valore scelto e 24, ovvero la capienza massima del settore, in modo da evitare di compiere conteggi con valori superiori al massimo prestabilito.

Nella prossima pagina è possibile osservare la struttura del datapath da noi realizzato.


N_AUTOA
N_AUTOB
N_AUTOC
MEM_A
1 MUX 0
REGISTRO
00001
00001
11111
=
+
-
00
01
10
11
MUX
OP_A
POSTO_A
REGISTRO
MEM_B
1 MUX 0
REGISTRO
00001
00001
11111
=
+
-
00
01
10
11
MUX
OP_B
POSTO_B
REGISTRO
MEM_C
1 MUX 0
REGISTRO
11000
11000
=
>
1 MUX 0
00001
00001
+
-
00
01
10
11
MUX
OP_C
POSTO_C
REGISTRO
P_A
P_B
P_C

Statistiche del circuito

Vengono ora descritte le statistiche del circuito prima e dopo l'ottimizzazione per area. Per prima cosa abbiamo cercato di minimizzare gli stati della FSM tramite il comando **"state_minimize stamina"**, ma il programma non è riuscito a ridurre il numero di stati.

Abbiamo quindi codificato gli stati con "**state_assign jedi**" ottenendo le seguenti statistiche:

```
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 5
Number of states in minimized machine : 5
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> wl Controllore.blif
sis> print_stats
FSM             pi= 8   po=11   nodes= 14       latches= 3
lits(sop)= 172  #states(STG)=   5
sis>
```

Che dopo l'ottimizzazione diventano:

```
sis> sweep
sis> print_stats
FSM             pi= 8   po=11   nodes= 19       latches= 3
lits(sop)=  71  #states(STG)=   5
sis>
```

Siamo riusciti ad ottenere una diminuzione del numero di letterali proprio come volevamo.

Il datapath da noi scritto presenta le seguenti statistiche:

```
Warning: network `PARK_B', node "C0UT1" does not fanout
Warning: network `PARK_B', node "[6078]" does not fanout
Warning: network `PARK_B', node "[6080]" does not fanout
Warning: network `PARK_B', node "[6081]" does not fanout
Warning: network `PARK_B', node "[6082]" does not fanout
Warning: network `PARK_B', node "[6083]" does not fanout
Warning: network `PARK_C', node "COUT" does not fanout
Warning: network `PARK_C', node "C0UT1" does not fanout
Warning: network `PARK_C', node "[6078]" does not fanout
Warning: network `PARK_C', node "[6080]" does not fanout
Warning: network `PARK_C', node "[6081]" does not fanout
Warning: network `PARK_C', node "[6082]" does not fanout
Warning: network `PARK_C', node "[6083]" does not fanout
Warning: network `DATAPATH', node "COUT" does not fanout
Warning: network `DATAPATH', node "C0UT1" does not fanout
Warning: network `DATAPATH', node "[6078]" does not fanout
Warning: network `DATAPATH', node "[6080]" does not fanout
Warning: network `DATAPATH', node "[6081]" does not fanout
Warning: network `DATAPATH', node "[6082]" does not fanout
Warning: network `DATAPATH', node "[6083]" does not fanout
Warning: network `DATAPATH', node "[6354]" does not fanout
Warning: network `DATAPATH', node "[6369]" does not fanout
Warning: network `DATAPATH', node "[6382]" does not fanout
Warning: network `DATAPATH', node "[6383]" does not fanout
Warning: network `DATAPATH', node "[6384]" does not fanout
Warning: network `DATAPATH', node "[6385]" does not fanout
Warning: network `DATAPATH', node "[6386]" does not fanout
Warning: network `DATAPATH', node "[6444]" does not fanout
Warning: network `DATAPATH', node "[6459]" does not fanout
Warning: network `DATAPATH', node "[6472]" does not fanout
Warning: network `DATAPATH', node "[6473]" does not fanout
Warning: network `DATAPATH', node "[6474]" does not fanout
Warning: network `DATAPATH', node "[6475]" does not fanout
Warning: network `DATAPATH', node "[6476]" does not fanout
sis> print_stats
DATAPATH        pi=14   po= 6   nodes=212       latches=18
lits(sop)= 975
```

I warning non sono preoccupanti, in quanto non interferiscono con il risultato del nostro circuito. Viene richiesta l'ottimizzazione per area, quindi cercheremo di ridurre al minimo il numero di letterali.

A questo punto ripetendo più volte i comandi di ottimizzazione, tra i quali anche "source script.rugged" che tra quelli predefiniti di SIS è quello che in generale offre il miglior risultato, cerchiamo di ridurre il numero di letterali.

Dopo ulteriori ottimizzazioni le statistiche finali del circuito sono le seguenti:

```
sis> print_stats
DATAPATH        pi=14   po= 6   nodes= 50       latches=18
lits(sop)= 288
sis> wl Elaborazione1.blif
```

La FSMD, essendo formata dall'unione dei due circuiti sopra descritti, presenta statistiche che sono la somma di quelle dei suoi componenti:

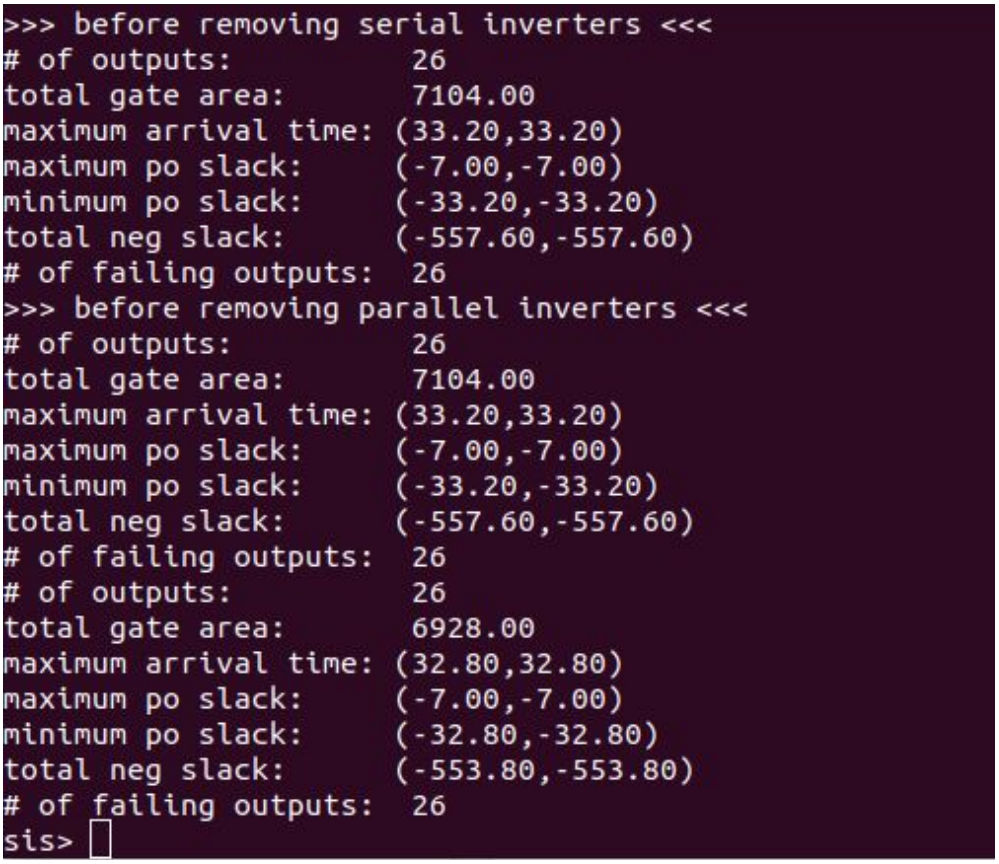
```
sis> rl FSMD_non_ottimizzata.blif
Warning: network `Controllore.blif', node "Posto_A" does not fanout
Warning: network `Controllore.blif', node "Posto_B" does not fanout
Warning: network `Controllore.blif', node "Posto_C" does not fanout
sis> print_stats
FSMD            pi= 5   po= 5   nodes= 69       latches=21
lits(sop)= 359
```

Passiamo quindi all'ottimizzazione:

```
sis> print_stats
FSMD            pi= 5   po= 5   nodes= 67       latches=21
lits(sop)= 340
```

Numero di gates e ritardo dopo la mappatura

Una volta ottimizzato il circuito, dobbiamo mapparlo con una libreria, in modo da avere delle statistiche più verosimili per quanto riguarda area e ritardo. Nel nostro caso la libreria assegnata è la “synch.genlib”. Una volta mappato il circuito presenta le seguenti statistiche:



Il total gate area è 7104.00 mentre il cammino critico è pari a 33.20.

Descrizione delle scelte progettuali

- Il datapath è composto di 3 sottocircuiti, ovvero **PARK_A, PARK_B** e **PARK_C**, che si occupano della gestione di un singolo parcheggio .

- All’interno del data path sono stati introdotti tre latches ulteriori al fine di ritardare di un ciclo di clock l'ingresso dei segnali di **POSTO_(A/B/C)[1]** nella FSM. Questa scelta si è resa necessaria per aggirare il problema dei cicli, che ritardava la chiusura del parcheggio se questo fosse stato totalmente occupato.

- Nello stato di Reset, le sbarre rimarranno aperte fino a quando non verrà digitata la giusta sequenza di apertura, ovvero 11111

- L’underflow è stato ignorato, per cui se si cercherà di uscire da un parcheggio vuoto, questo verrà segnato come occupato