

# ELABORATO SIS

---

Architettura degli Elaboratori  
Laboratorio

## SPECIFICHE

Si progetti un dispositivo 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. La mattina il dispositivo viene attivato manualmente da un operatore inserendo la sequenza di 5 bit 11111. Al ciclo successivo il sistema attende l'inserimento del numero di automobili presenti nel settore A (sempre in 5 bit) e ne memorizza il valore. Nei due cicli successivi avviene lo stesso per i settori B e C. Nel caso in cui il valore inserito superi il numero di posti macchina nel relativo settore si considerino tutti i posti occupati.

A partire dal quarto ciclo di clock il sistema inizia il suo funzionamento autonomo. Ad ogni ciclo un utente si avvicina alla posizione di ingresso o uscita e preme un pulsante relativo al settore in cui intende parcheggiare.

Il circuito ha due ingressi definiti nel seguente ordine:

- IN/OUT [2 bit]: 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 bit]: 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.

Ad ogni richiesta il dispositivo risponde aprendo una sbarra e aggiornando lo stato dei posti liberi nei vari settori. Se un utente chiede di occupare un settore già completo il sistema non deve aprire alcuna sbarra.

Il circuito deve avere tre output che devono seguire il seguente ordine:

- SETTORE\_NON\_VALIDO [1 bit]: se il settore inserito non è valido questo bit deve essere alzato.
- SBARRA\_(IN/OUT) [1 bit]: 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 bit a settore]: questo bit assume valore 1 se tutti i posti macchina nel settore sono occupati, 0 se ci sono ancora posti liberi.

Il dispositivo si spegne quando riceve la sequenza 00000 in input.

Nel caso in cui venga inserito un settore non valido la sbarra deve rimanere chiusa. Anche in questo caso, i valori dei settori devono comunque essere riportati rispetto alla loro situazione attuale.

## ARCHITETTURA GENERALE DEL CIRCUITO

Il dispositivo implementato consente la gestione automatizzata delle operazioni di ingresso e uscita in un parcheggio. Il circuito è composto da due componenti principali: una macchina a stati finiti (FSM) che agisce come controllore e un datapath che funge da elaboratore. Il sistema accetta una serie di ingressi e genera corrispondenti uscite secondo le seguenti specifiche:

Ingressi:

- ⇒ **IN e OUT:** Questi ingressi indicano se l'utente richiede un'operazione di ingresso (IN) o di uscita (OUT) dal parcheggio.
- ⇒ **A, B, C:** Questi ingressi rappresentano le scelte dei settori A, B e C. L'utente attiva uno di questi ingressi per selezionare il settore desiderato. Ogni ingresso è una scelta binaria, attivato (1) o disattivato (0), per il settore corrispondente.

Uscite:

- ⇒ **SETTORE\_NON\_VALIDO:** Questo bit assume valore 1 se l'utente ha inserito un settore non valido, altrimenti rimane a 0.
- ⇒ **SBARRA\_IN e SBARRA\_OUT:** Questi bit indicano se la sbarra relativa all'ingresso (SBARRA\_IN) o all'uscita (SBARRA\_OUT) deve essere aperta (valore 1) o chiusa (valore 0).
- ⇒ **SECTOR\_A, SECTOR\_B, SECTOR\_C:** Questi bit indicano se tutti i posti del settore corrispondente (A, B o C) sono occupati (valore 1) o se ci sono ancora posti liberi (valore 0).

### Comunicazione controllore e datapath:

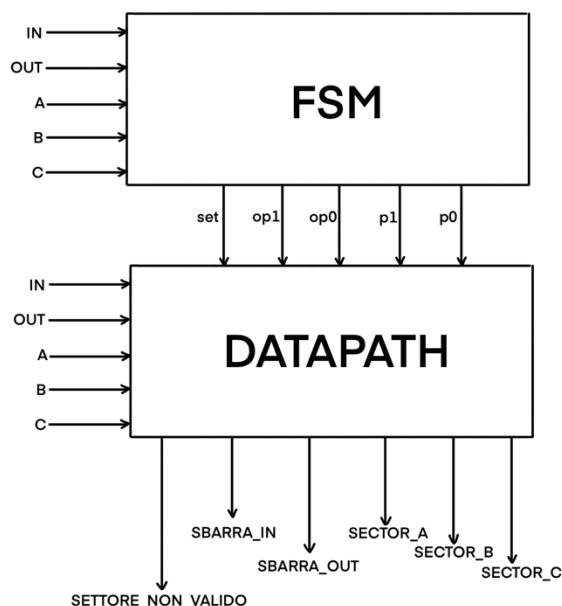
La FSM funge da controllore principale del sistema di gestione del parcheggio; accetta gli input u4, u3, u2, u1, u0 (che corrispondono ad IN, OUT, A, B, C), che rappresentano le operazioni di ingresso e uscita e le scelte dei settori.

Produce gli output set, op1, op0, p1, p0, che guidano il comportamento del sistema.

Il Datapath agisce come l'elaboratore principale del sistema; accetta gli input o1, o0, s2, s1, s0 (che corrispondono a IN, OUT, A, B, C), set, op1, op0, p1, p0 dalla FSM e dagli ingressi del sistema.

Calcola e gestisce le uscite del sistema in base agli input e produce gli output SETTORE\_NON\_VALIDO, SBARRA\_IN, SBARRA\_OUT, SECTOR\_A, SECTOR\_B, SECTOR\_C.

In sintesi, la FSM controlla lo stato e il flusso generale del sistema, mentre il Datapath esegue i calcoli e le operazioni specifiche necessarie per gestire l'ingresso, l'uscita e lo stato dei settori nel parcheggio, generando le risposte appropriate. Insieme, la FSM (Finite State Machine) e il Datapath cooperano per garantire un funzionamento efficiente e coerente del sistema di gestione del parcheggio.

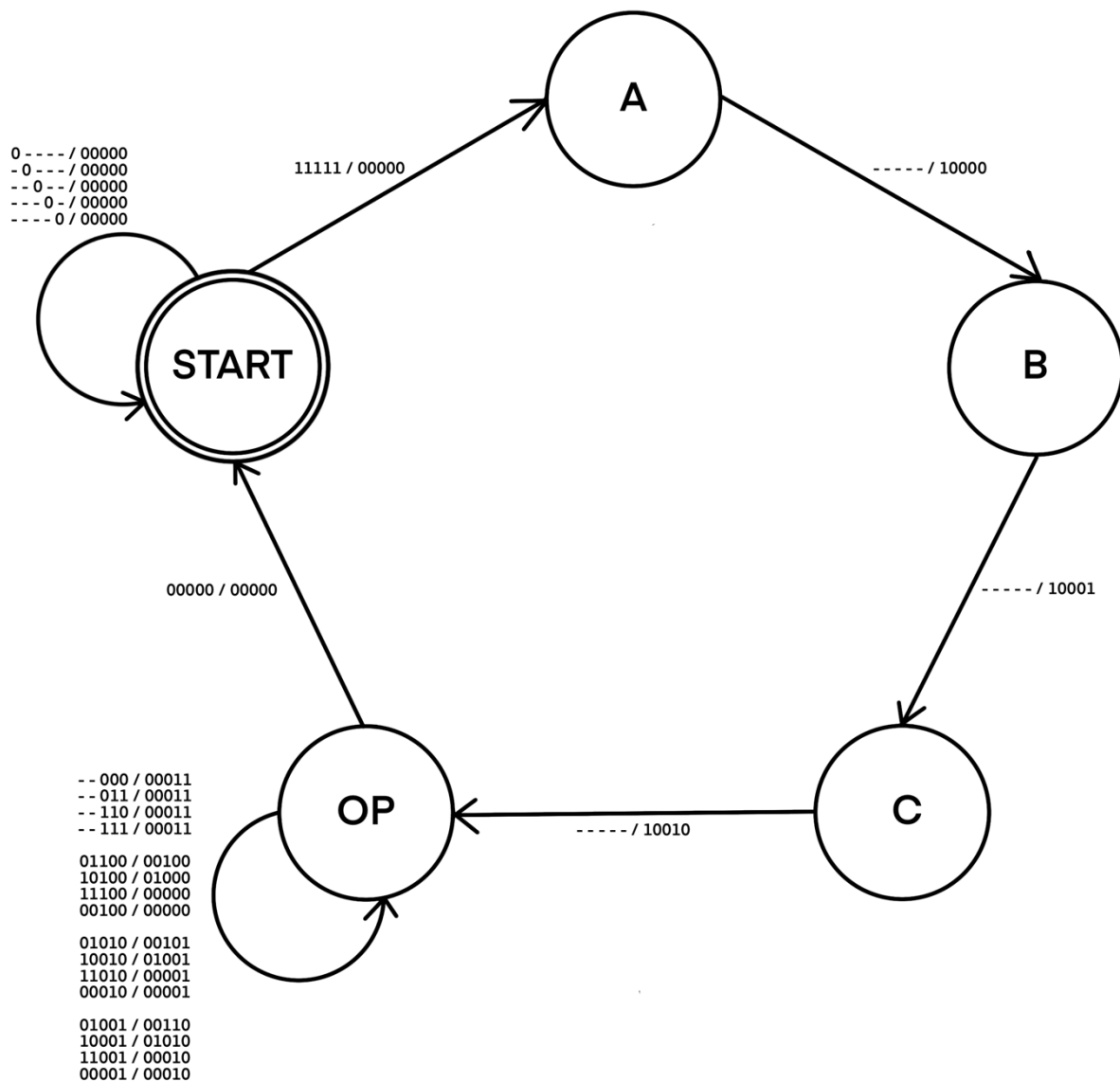


## DIAGRAMMA DI STATI DEL CONTROLLORE

Il controllore è una macchina a stati finiti di Mealy che presenta cinque ingressi (u4, u3, u2, u1, u0, che corrispondono a IN, OUT, A, B, C) e cinque uscite (set, op1, op0, p1, p0).

Il controllore del circuito presenta cinque stati:

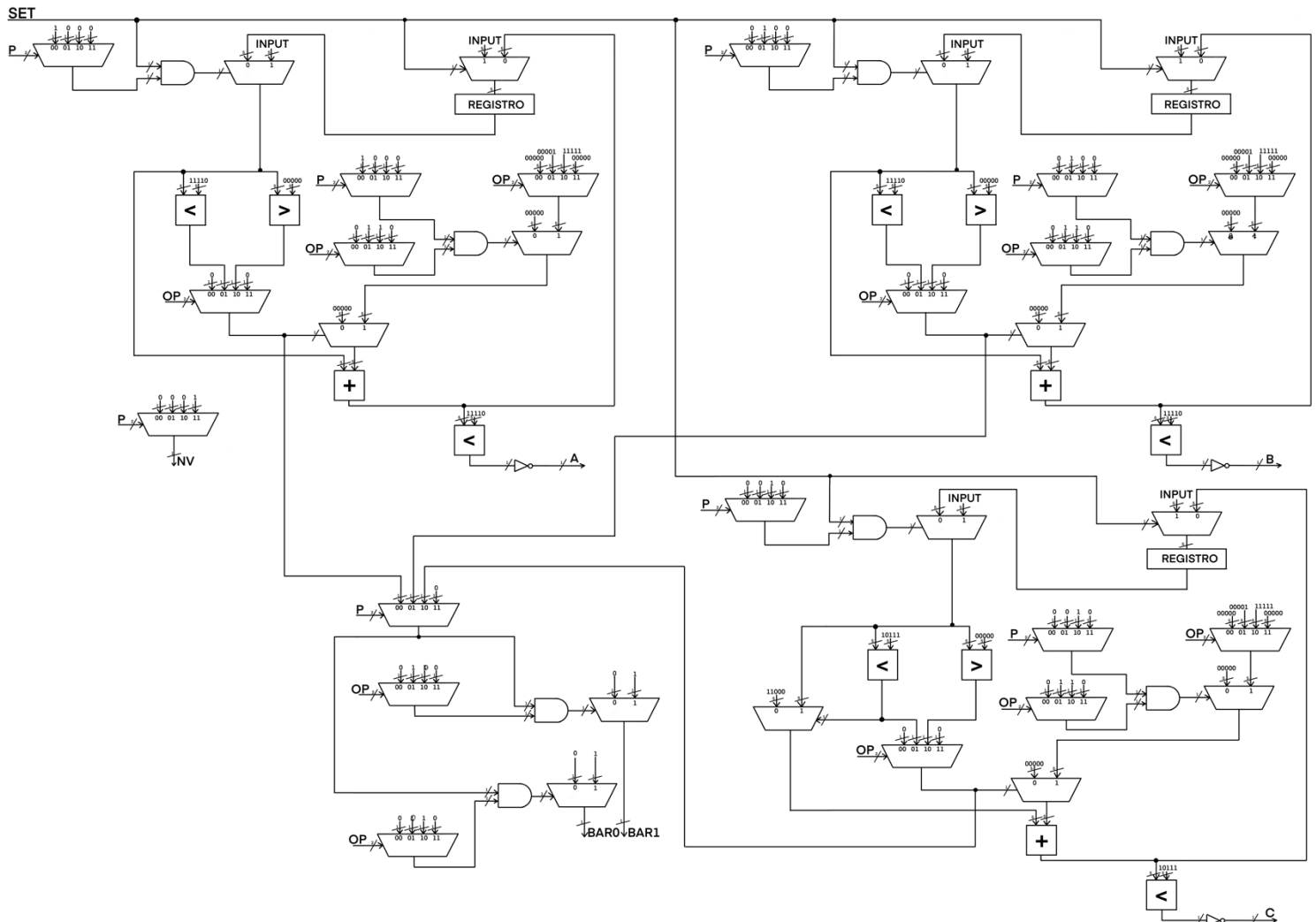
1. **START (Avvio):** Questo stato rappresenta il momento iniziale del dispositivo; dopo aver inserito la sequenza "11111" il dispositivo viene avviato.
2. **A:** Questo stato rappresenta la fase di impostazione del numero di macchine presenti nel parcheggio A.
3. **B:** Questo stato rappresenta la fase di impostazione del numero di macchine presenti nel parcheggio B.
4. **C:** Questo stato rappresenta la fase di impostazione del numero di macchine presenti nel parcheggio C.
5. **OP (Operazioni):** Questo stato è raggiunto dopo che tutti i parcheggi sono stati impostati correttamente. In questo stato il dispositivo è operativo e può eseguire operazioni di ingresso e uscita nei settori del parcheggio su richiesta dell'utente. Inoltre, è possibile spegnere il dispositivo inserendo la sequenza "00000".



## ARCHITETTURA DEL DATAPATH

Il datapath è l'unità di elaborazione del sistema, che riceve gli ingressi dalla FSM e genera le uscite, gestendo operazioni come l'apertura delle sbarre e il monitoraggio dello stato dei settori del parcheggio.

Il Datapath riceve 10 segnali di ingresso (o1, o0, s2, s1, s0 (che corrispondono a IN, OUT, A, B, C), set, op1, op0, p1, p0 - i 5 input (in comune con gli input della FSM) e manda 5 segnali in uscita SETTORE\_NON\_VALIDO, SBARRA\_IN, SBARRA\_OUT, SECTOR\_A, SECTOR\_B, SECTOR\_C (che nel disegno corrispondono a NV, BAR0, BAR1, A, B, C).



Il nostro Datapath è suddiviso in cinque settori:

- I settori **A**, **B** e **C** gestiscono l'ingresso e l'uscita delle macchine attraverso complesse reti di multiplexer e comparatori. Verificano se il parcheggio è pieno o ha spazio rimanente confrontando il numero di macchine con il limite massimo consentito e producono un bit di uscita, 1 se è pieno e 0 se c'è spazio disponibile.
- Il settore **BAR** è costituito da una serie di multiplexer che determinano la validità delle operazioni. In altre parole, questo settore decide se un'operazione è autorizzata o meno.
- Il settore **NV** ha il compito di segnalare se il settore inserito è valido o non valido. In altre parole, verifica se la scelta del settore effettuata è una scelta accettabile o se è considerata non valida all'interno del sistema. Se il settore è non valido, l'output sarà 1.

## STATISTICHE CIRCUITO

Di seguito vengono fornite le statistiche del circuito, prima e dopo l'ottimizzazione per l'area.

Per prima cosa ho tentato di ridurre il numero di stati nella FSM utilizzando il comando "state\_minimize stamina", tuttavia il programma non è riuscito a ottenere una riduzione del numero di stati.

Successivamente ho usato il comando "state\_assign jedi" per ottenere una codifica degli stati ottenendo le statistiche sotto mostrate.

Ho usato quindi il comando "source script.rugged" per ottimizzare la FSM, difatti i letterali sono passati da 196 a 59.

```
Elaborato — root@672115b69417: /esercizio-sis — com.docker.cli ◀ docke...

[sis>
[sis> rl FSMnonottimizzata.blif
[sis> print_stats
fsm          pi= 5  po= 5  nodes= 5      latches= 0
lits(sop)=   0  #states(STG)= 5
[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> print_stats
fsm          pi= 5  po= 5  nodes= 8      latches= 3
lits(sop)= 196  #states(STG)= 5
[sis> source script.rugged
[sis> print_stats
fsm          pi= 5  po= 5  nodes= 12     latches= 3
lits(sop)=  59  #states(STG)= 5
[sis>
```

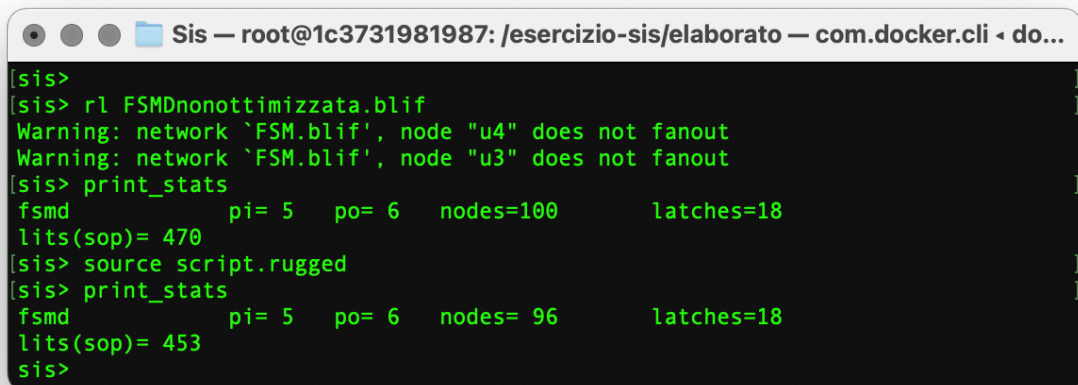
Il datapath invece si presenta così:

```
Elaborato — root@672115b69417: /esercizio-sis — com.docker.cli ◀ docke...

[sis>
[sis> rl datapath.blif
Warning: network `datapath', node "couta" does not fanout
Warning: network `datapath', node "coutb" does not fanout
Warning: network `datapath', node "coutc" does not fanout
[sis> print_stats
datapath     pi=10  po= 6  nodes=198    latches=15
lits(sop)=1259
[sis> source script.rugged
[sis> print_stats
datapath     pi=10  po= 6  nodes= 91    latches=15
lits(sop)= 414
[sis> sweep
[sis> print_stats
datapath     pi=10  po= 6  nodes= 88    latches=15
lits(sop)= 411
[sis>
```

Dopo aver eseguito i comandi "source script.rugged" e sweep per l'ottimizzazione otteniamo un ottimo risultato.

La FSMD (Finite State Machine with Datapath), essendo la combinazione di due circuiti precedentemente descritti, presenta statistiche aggregate che derivano dalla somma delle statistiche dei suoi componenti individuali.



```
Sis — root@1c3731981987: /esercizio-sis/elaborato — com.docker.cli ◀ do...  
[sis>  
[sis> rl FSMDnonottimizzata.blif  
Warning: network `FSM.blif', node "u4" does not fanout  
Warning: network `FSM.blif', node "u3" does not fanout  
[sis> print_stats  
fsmd      pi= 5   po= 6   nodes=100   latches=18  
lits(sop)= 470  
[sis> source script.rugged  
[sis> print_stats  
fsmd      pi= 5   po= 6   nodes= 96   latches=18  
lits(sop)= 453  
sis>
```



## NUMERO DI GATE E RITARDO

Dopo aver ottimizzato il circuito, è necessario associarlo a una libreria, nel nostro caso specifico denominata "synch.genlib". Una volta effettuata questa associazione, il circuito mappato presenta nuove statistiche che offrono una stima più accurata dell'area e del ritardo del circuito.

Il circuito è stato poi minimizzato rispetto all'area.

```
Sis — root@1c3731981987: /esercizio-sis/elaborato — com.docker.cli < do...
[sis>
[sis> rl FSMD.blif
[sis> read_library synch.genlib
[sis> map -m 0 -s
warning: unknown latch type at node '{{[15]}}' (RISING_EDGE assumed)
warning: unknown latch type at node '{{[16]}}' (RISING_EDGE assumed)
warning: unknown latch type at node '{{[17]}}' (RISING_EDGE assumed)
WARNING: uses as primary input drive the value (0.20,0.20)
WARNING: uses as primary input arrival the value (0.00,0.00)
WARNING: uses as primary input max load limit the value (999.00)
WARNING: uses as primary output required the value (0.00,0.00)
WARNING: uses as primary output load the value 1.00
>>> before removing serial inverters <<<
# of outputs:      24
total gate area:    8160.00
maximum arrival time: (44.80,44.80)
maximum po slack:   (-6.40,-6.40)
minimum po slack:   (-44.80,-44.80)
total neg slack:    (-836.20,-836.20)
# of failing outputs: 24
>>> before removing parallel inverters <<<
# of outputs:      24
total gate area:    8128.00
maximum arrival time: (45.00,45.00)
maximum po slack:   (-6.40,-6.40)
minimum po slack:   (-45.00,-45.00)
total neg slack:    (-836.20,-836.20)
# of failing outputs: 24
# of outputs:      24
total gate area:    7504.00
maximum arrival time: (43.80,43.80)
maximum po slack:   (-6.40,-6.40)
minimum po slack:   (-43.80,-43.80)
total neg slack:    (-819.00,-819.00)
# of failing outputs: 24
[sis> print_map_stats
Total Area          = 7504.00
Gate Count          = 211
Buffer Count        = 18
Inverter Count      = 33
Most Negative Slack = -43.80
Sum of Negative Slacks = -819.00
Number of Critical PO = 24
sis> ]
```

Il totale gate area è 8160 e il cammino critico è 44.80.



## **SCELTE PROGETTUALI**

1. Numero limitato di stati nella FSM: Il sistema è stato progettato con una FSM composta da soli 5 stati principali: "Start", "Settaggio A", "Settaggio B", "Settaggio C" e "Operazioni". L'obiettivo di questa scelta progettuale è stato quello di semplificare il controllo generale del sistema, mantenendo allo stesso tempo efficienza. L'implementazione di funzionalità più specifiche e dettagliate è stata affidata al Datapath, consentendo alla FSM di rimanere centrato sui compiti di controllo principali.
2. Creazione di cinque settori nel datapath: suddividere il Datapath in cinque settori distinti, ognuno con un ruolo specifico nel sistema di gestione del parcheggio, ha influenzato in modo significativo l'organizzazione del Datapath.