

**ELABORATO SIS**  
**ARCHITETTURA DEGLI ELABORATORI**  
**UNIVERSITÀ DEGLI STUDI DI VERONA**  
**A.A. 2019-2020**

Bramè Luca VR437445  
Canaia Laura VR437045  
Nikolic Jovan VR422427

# INDICE

1. Architettura generale del circuito
2. Struttura della FSM
3. FSM
  1. Struttura FSM
  2. Spiegazione FSM
4. Datapath
  1. Struttura datapath
  2. Spiegazione datapath
5. Statistiche
6. Scelte progettuali
7. Statistiche e Mappatura
8. Scelte progettuali

## ARCHITETTURA GENERALE DEL CIRCUITO

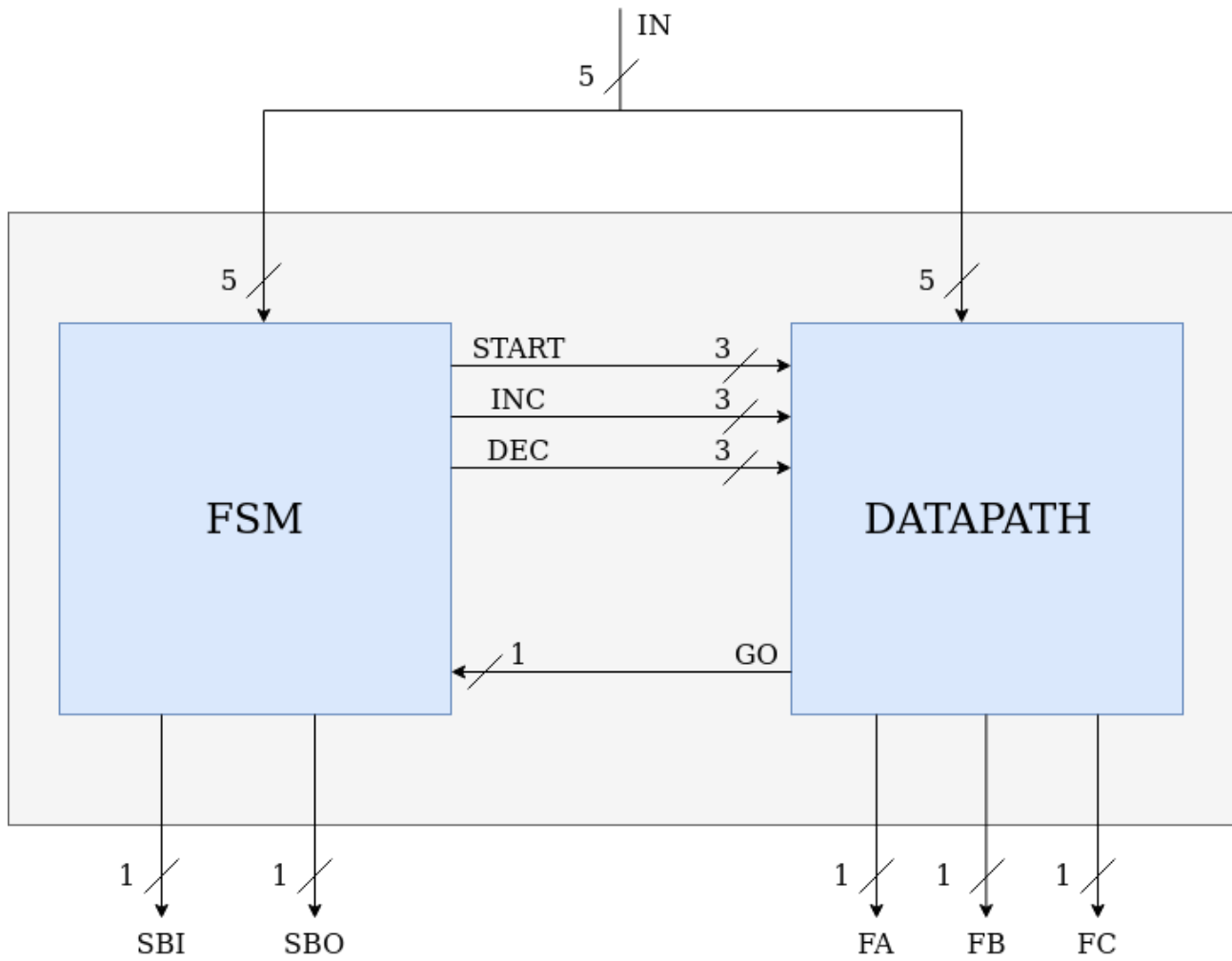
Il presente circuito è preposto alla gestione di un parcheggio con ingresso/uscita automatizzati. Il parcheggio è suddiviso in 3 settori: i settori A e B hanno 31 posti macchina ciascuno, mentre il settore C ha 24 posti macchina. Al momento dell'ingresso l'utente deve dichiarare in quale settore vuole 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.

Gli input del sistema sono:

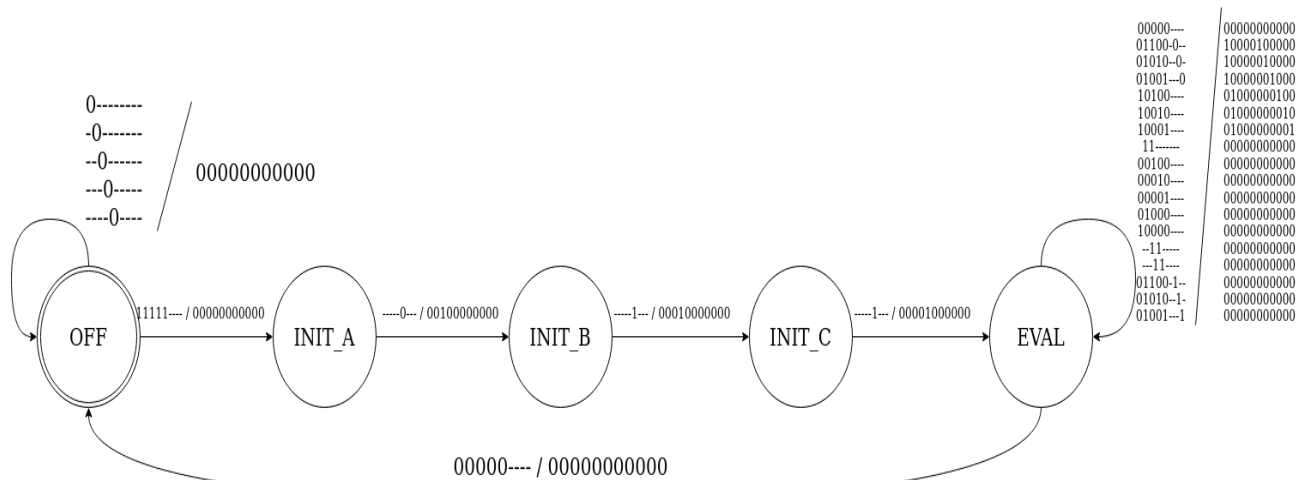
- Input del circuito
  - **IN[5]** - Input generali del circuito, inseriti manualmente da tastierino
- Output del circuito
  - Prodotti dalla FSM:
    - **SBI[1]** - Bit che riporta lo stato della sbarra in ingresso. Vale 1 se la sbarra è alzata, 0 se la sbarra è abbassata.
    - **SB0[1]** - Bit che riporta lo stato della sbarra in uscita. Vale 1 se la sbarra è alzata, 0 se la sbarra è abbassata.
  - Prodotti dal DATAPATH:
    - **FA[1]** - Bit che vale 1 se il settore A è pieno, 0 se ci sono ancora posti macchina
    - **FB[1]** - Bit che vale 1 se il settore B è pieno, 0 se ci sono ancora posti macchina
    - **FC[1]** - Bit che vale 1 se il settore C è pieno, 0 se ci sono ancora posti macchina
- Segnali interni
  - Da FSM a Datapath
    - **START[3]** - Segnale a 3 bit comunicato dalla FSM al Datapath durante la fase di inizializzazione dei tre settori del parcheggio. Segue una codifica one-hot:
      - 100 - Inizializza settore A
      - 010 - Inizializza settore B
      - 001 - Inizializza settore C
    - **INC[3]** - Segnale a 3 bit comunicato dalla FSM al Datapath per incrementare il numero di macchine presenti in un settore. Segue una codifica one-hot:
      - 100 - Incrementa settore A
      - 010 - Incrementa settore B
      - 001 - Incrementa settore C
    - **DEC[3]** - Segnale a 3 bit comunicato dalla FSM al Datapath per decrementare il numero di macchine presenti in un settore. Segue una codifica one-hot:
      - 100 - Decrementa settore A
      - 010 - Decrementa settore B

- 001 - Decrementa settore C
- Da Datapath a FSM
  - **GO[1]** - Segnale da un bit inviato da Datapath a FSM ogni volta che un settore viene inizializzato per commutare al successivo ciclo di clock. Il segnale viene alzato per la prima volta una volta inizializzato il settore A e viene impiegato fino a che lo stato EVAL è raggiunto.

## STRUTTURA DELLA FSMD



# STRUTTURA DELLA FSM



La FSM ha il compito di regolare lo stato di accensione e spegnimento della macchina, nonché l'esecuzione corretta del circuito.

Gli input della FSM sono:

- **IN[ 5 ]** - Input generali del circuito
- **GO[ 1 ]** - Segnale di stato ricevuto dal DATAPATH
- **FA[ 1 ]** - Segnale alzato quando il settore A è pieno
- **FB[ 1 ]** - Segnale alzato quando il settore B è pieno
- **FC[ 1 ]** - Segnale alzato quando il settore C è pieno

Gli output sono:

- **SBI[ 1 ]** - Bit alzato quando la barra in ingresso è alzata
- **SB0[ 1 ]** - Bit alzato quando la barra in uscita è alzata
- **START[ 3 ]** - Segnale a codifica one-hot che informa il DATAPATH del settore da inizializzare
- **INC[ 3 ]** - Segnale a codifica one-hot che informa il DATAPATH del settore da incrementare
- **DEC[ 3 ]** - Segnale a codifica one-hot che informa il DATAPATH del settore da decrementare

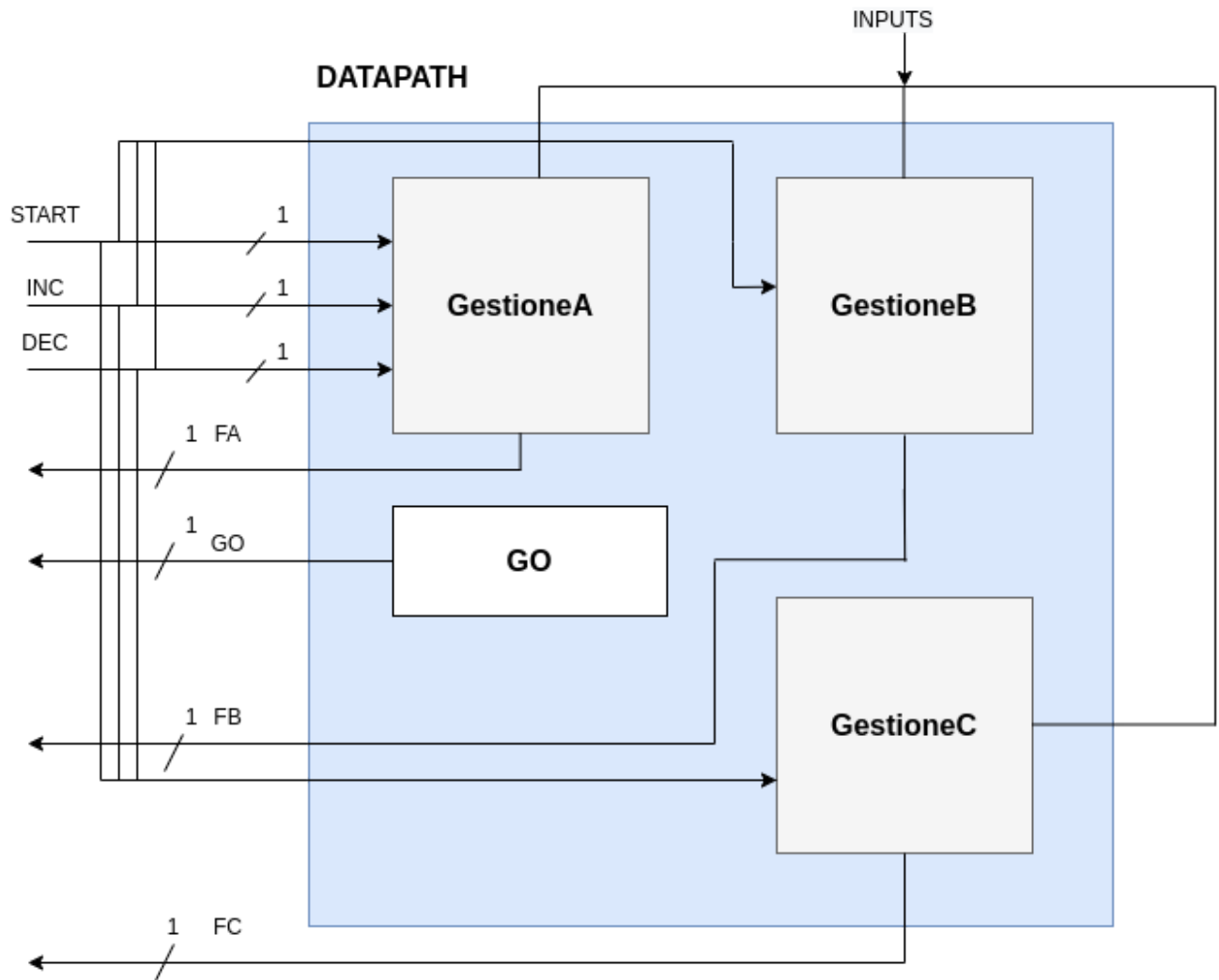
Per quanto riguarda gli stati e il comportamento del controllore:

1. **OFF** - Stato di reset. Il controllore è spento quando si trova su questo stato, viene spento quando commuta verso questo stato e viene acceso quando commuta da questo stato. La FSM cicla su questo stato per qualsiasi sequenza di input diversa da "11111", sequenza che fa accendere la macchina, facendola commutare verso lo stato INIT\_A.
2. **INIT\_A** - Stato in cui viene inizializzato il settore A. Essendo il primo settore da inizializzare, non si aspetta il segnale GO dal DATAPATH. Alza l'output START\_A nella transizione verso lo stato INIT\_B per impartire al DATAPATH il comando di inizializzare il settore A con la sequenza di 5 bit appena mandata in input.
3. **INIT\_B** - Stato in cui viene inizializzato il settore B. Alza l'output START\_A nella transizione verso lo stato INIT\_C per impartire al DATAPATH il comando di inizializzare il settore A con la sequenza di 5 bit appena mandata in input. A partire da questo stato fino alla commutazione verso INIT viene messo in pratica un handshake tra FSM e Datapath tale per cui la FSM invia il segnale START opportuno per impartire il comando di inizializzazione al DATAPATH, che alza il segnale GO a

operazione eseguita per permettere alla FSM di commutare al prossimo ciclo di clock assicurandosi che l'inizializzazione sia stata effettivamente eseguita.

4. **INIT\_C** - Stato in cui viene inizializzato il settore A. Alza l'output START\_A nella transizione verso lo stato INIT\_B per impartire al DATAPATH il comando di inizializzare il settore A con la sequenza di 5 bit appena mandata in input.
5. **EVAL** - Stato in cui, una volta inizializzati i tre settori del parcheggio, interpreta i comandi di ingresso e uscita dai settori alzando opportunamente le sbarre e istruendo il DATAPATH di effettuare eventuali operazioni di incremento e decremento dei settori. L'ingresso e l'uscita dai settori vengono intesi secondo specifica, così come l'output restituito ad ogni transizione. Questo stato si comporta nel seguente modo:
  - a. Se l'input inserito è valido e l'operazione desiderata è possibile, alza in output il bit relativo alla sbarra da alzare e manda al DATAPATH un segnale INC[settore] o DEC[settore] per impartire al DATAPATH il compito di incrementare il numero dei posti macchina occupati all'ingresso di un settore o decrementare tale numero in uscita.
    - i. La FSM riceve in input i tre segnali FA[1] FB[1] FC[1] dal DATPATH, che, se alzati, stanno a significare se il settore è pieno. Questi vengono interpretati per decidere se una transizione in un settore in ingresso è possibile o meno.
  - b. Se l'input inserito è valido ma l'operazione desiderata non è possibile, la FSM manda in output per i primi due bit la sequenza 00, che sta a significare che nessuna delle due sbarre è stata alzata poiché l'operazione desiderata non è possibile
  - c. Se viene inserito un input non valido, viene mandata in output la stessa sequenza.
  - d. Se l'input inserito corrisponde alla sequenza "00000", il controllore commuta verso lo stato OFF, spegnendosi. Successivamente, sarà necessario accendere di nuovo il controllore con la sequenza "11111" e re-inizializzare i tre registri per riprendere l'utilizzo della macchina.

## STRUTTURA DEL DATAPATH





## FUNZIONAMENTO DEL DATAPATH

Il DATAPATH riceve in come input segnali di origine diversa. In primis riceve i cinque bit di input generali `IN[5]`, utilizzati per inizializzare i posti occupati all'accensione della macchina. La seconda categoria di input sono quelli provenienti dalla FSM, ossia i segnali a tre bit in codifica one-hot `START[3]`, `INC[3]` e `DEC[3]` utilizzati rispettivamente per inizializzare, incrementare o decrementare un settore.

Il numero di posti occupati per ogni settore viene preservato per il tempo dell'esecuzione del circuito in tre registri da 5 bit. Questo permette al DATAPATH di controllare lo stato dei tre settori utilizzando i moduli `GestioneA`, `GestioneB` e `GestioneC` ad ogni ciclo di clock conservando lo stesso per più cicli di clock anziché "dimenticare" immediatamente l'informazione. Questo comportamento è fondamentale perché, In fase di inizializzazione, per ognuno dei tre settori, il DATAPATH esegue un controllo tra il valore inserito in input sulla linea `IN[5]` e una costante numerica rappresentante la capienza massima del settore e, in base al risultato, seleziona uno dei due segnali e lo scrive sul registro opportuno. Nel caso in cui un settore inizializzato con un numero di macchine pari o superiore alla capienza massima, viene alzato il segnale `F[settore]` relativo che verrà sia mandato come output della FSM per produrre la configurazione di output secondo specifica sia alla FSM, che ha bisogno di questa informazione quando, trovandosi sullo stato `Eval`, deve decretare se l'operazione in ingresso ad un dato settore sia possibile in virtù del numero di posti macchina attualmente presenti nello stesso. È cruciale notare come questo controllo viene eseguito ad ogni ciclo di clock, sia in fase di inizializzazione, sia in fase di aggiornamento di un settore.

Una volta inizializzato il settore A, fino a quando la FSM non si trova nello stato `Eval`, oltre che i segnali `FA[1]` `FB[1]` `FC[1]`, il DATAPATH manda verso la FSM un segnale `GO[1]` essenziale per l'attuazione del protocollo di *handshake* spiegato sopra.

## STATISTICHE E MAPPATURA

Come da specifica, il presente circuito è stato ottimizzato e mappato sulla libreria tecnologica `synch.genlib`. Per velocizzare le operazioni di minimizzazione e testing del circuito, l'intera sequenza è stata inclusa in uno script generale chiamato `optimize.sis`, che a sua volta richiama lo script `fsm_update.sis`, che si occupa di leggere il prototipo della FSM, eseguire l'assegnazione degli stati con `jedi`, ottimizzare il controllore e scrivere il controllore ottimizzato risultante sul file `fsm.blif`.

Per prima cosa, abbiamo cercato di ottimizzare la FSM quanto più possibile. Si noti che non sono stati ottimizzati gli stati del controllore con `stamina` poiché l'esecuzione dello stesso altera il funzionamento del controllore a tal punto da pregiudicarne il corretto funzionamento.

```
root@823ded275331:/data/sis/elaborato_sis_2019/src# sis
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> read_blif fsm_proto_minimal.blif
sis> state_assign jedi
Running jedi, written by Bill Lin, UC Berkeley
sis> print_stats
FSM          pi= 9   po=11   nodes= 14       latches= 3
lits(sop)= 144 #states(STG)= 5
sis> source script.rugged
sis> print_stats
FSM          pi= 9   po=11   nodes= 21       latches= 3
lits(sop)=  72 #states(STG)= 5
sis> 
```

Nonostante il fatto che gli stati non sono stati ridotti con stamina, abbiamo comunque ottenuto una forte riduzione nel numero di letterali. Ad ogni modo, l'esecuzione dello `script.rugged` ha incrementato leggermente il numero di nodi presenti nel controllore.

Successivamente, ci siamo occupati del DATAPATH. Il circuito è stato mappato sulla libreria tecnologica `synch.genlib` come da specifica.

```
~ : sh — Konsole
root@823ded275331:/data/sis/elaborato_sis_2019/src# sis
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> read_blif datapath.blif
Warning: network `SOTRATTORE5', node "NOT_CONNECTED0" does not fanout
Warning: network `SOTRATTORE5', node "NOT_CONNECTED1" does not fanout
Warning: network `DECREMENTATORE', node "NOT_CONNECTED0" does not fanout
Warning: network `DECREMENTATORE', node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE1', node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE1', node "NOT_CONNECTED0" does not fanout
Warning: network `GESTIONE1', node "[374]" does not fanout
Warning: network `GESTIONE1', node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE1', node "NOT_CONNECTED0" does not fanout
Warning: network `GESTIONE1', node "[330]" does not fanout
Warning: network `GESTIONE1', node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE1', node "NOT_CONNECTED0" does not fanout
Warning: network `GESTIONE1', node "[286]" does not fanout
Warning: network `DATAPATH', node "NOT_CONNECTED1" does not fanout
Warning: network `DATAPATH', node "NOT_CONNECTED0" does not fanout
Warning: network `DATAPATH', node "[374]" does not fanout
Warning: network `DATAPATH', node "[519]" does not fanout
Warning: network `DATAPATH', node "[587]" does not fanout
Warning: network `DATAPATH', node "[330]" does not fanout
Warning: network `DATAPATH', node "[618]" does not fanout
Warning: network `DATAPATH', node "[686]" does not fanout
Warning: network `DATAPATH', node "[286]" does not fanout
sis> 
```

Sebbene la lettura del `datapath.blif` generi immediatamente diversi avvertimenti, ciò non è preoccupante e i comportamenti descritti da `sis` in questo caso sono previsti. In primis, i nodi `NOT_CONNECTED[N]` rappresentano i segnali di OVER delle somme aritmetiche provenienti dai sommatore. Questi segnali sono informazioni superflue per il funzionamento del circuito, quindi si è deliberatamente scelto di non fargli fare fanout. Per quanto riguarda il resto dei nodi, per via del funzionamento interno del `DATAPATH` non tutti i segnali raggiungono effettivamente gli output, ma anche questo è completamente normale.

```
~ : sh — Konsole
Warning: network `GESTIONE' node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE' node "NOT_CONNECTED0" does not fanout
Warning: network `GESTIONE' node "[374]" does not fanout
Warning: network `GESTIONE' node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE' node "NOT_CONNECTED0" does not fanout
Warning: network `GESTIONE' node "[330]" does not fanout
Warning: network `GESTIONE' node "NOT_CONNECTED1" does not fanout
Warning: network `GESTIONE' node "NOT_CONNECTED0" does not fanout
Warning: network `GESTIONE' node "[286]" does not fanout
Warning: network `DATAPATH' node "NOT_CONNECTED1" does not fanout
Warning: network `DATAPATH' node "NOT_CONNECTED0" does not fanout
Warning: network `DATAPATH' node "[374]" does not fanout
Warning: network `DATAPATH' node "[519]" does not fanout
Warning: network `DATAPATH' node "[587]" does not fanout
Warning: network `DATAPATH' node "[330]" does not fanout
Warning: network `DATAPATH' node "[618]" does not fanout
Warning: network `DATAPATH' node "[686]" does not fanout
Warning: network `DATAPATH' node "[286]" does not fanout
sis> print_stats
DATAPATH      pi=14   po= 4   nodes=284      latches=15
lits(sop)=1011
sis> 
```

Queste sono le statistiche prima dell'ottimizzazione del DATAPATH.

```
~ : sh — Konsole
Warning: network `DATAPATH' node "[286]" does not fanout
sis> read_library synch.genlib
sis> map -s
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:      19
total gate area:    5840.00
maximum arrival time: (30.40,30.40)
maximum po slack:   (1073741824.00,1073741824.00)
minimum po slack:   (-30.40,-30.40)
total neg slack:    (-365.40,-365.40)
# of failing outputs: 18
>>> before removing parallel inverters <<<
# of outputs:      19
total gate area:    5840.00
maximum arrival time: (30.40,30.40)
maximum po slack:   (1073741824.00,1073741824.00)
minimum po slack:   (-30.40,-30.40)
total neg slack:    (-365.40,-365.40)
# of failing outputs: 18
# of outputs:      19
total gate area:    5840.00
maximum arrival time: (30.40,30.40)
maximum po slack:   (1073741824.00,1073741824.00)
minimum po slack:   (-30.40,-30.40)
total neg slack:    (-365.40,-365.40)
# of failing outputs: 18
sis> 
```

Queste sono le statistiche relative alla prima mappatura del circuito.

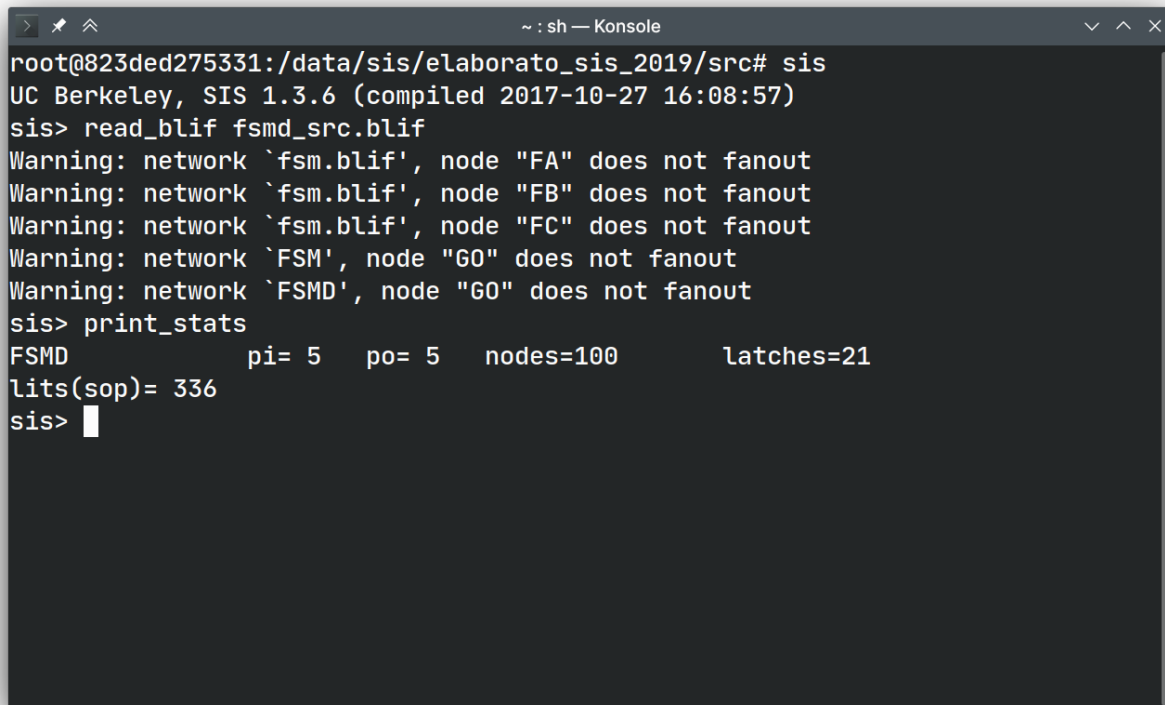
```
~ : sh — Konsole
sis> write_blif datapath_optimized.blif
sis> map -s
>>> before removing serial inverters <<<
# of outputs:      19
total gate area:    5488.00
maximum arrival time: (30.60,30.60)
maximum po slack:   (1073741824.00,1073741824.00)
minimum po slack:   (-30.60,-30.60)
total neg slack:    (-407.20,-407.20)
# of failing outputs: 18
>>> before removing parallel inverters <<<
# of outputs:      19
total gate area:    5360.00
maximum arrival time: (30.60,30.60)
maximum po slack:   (1073741824.00,1073741824.00)
minimum po slack:   (-30.60,-30.60)
total neg slack:    (-393.20,-393.20)
# of failing outputs: 18
# of outputs:      19
total gate area:    5168.00
maximum arrival time: (30.60,30.60)
maximum po slack:   (1073741824.00,1073741824.00)
minimum po slack:   (-30.60,-30.60)
total neg slack:    (-388.00,-388.00)
# of failing outputs: 18
sis> █
```

Questi sono invece i risultati dopo aver ottimizzato il circuito con `script.rugged` ed eseguito una seconda volta il mapping.

```
~ : sh — Konsole
root@823ded275331:/data/sis/elaborato_sis_2019/src# sis
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> read_blif datapath_optimized.blif
sis> print_stats
DATAPATH      pi=14   po= 4   nodes=126   latches=15
lits(sop)= 320
sis> █
```

Queste sono le statistiche del DATAPATH al termine dell'esecuzione. Sono statistiche molto positive: i letterali sono stati più che dimezzati, così come i nodi.

In ultima istanza, abbiamo ottimizzato e mappato la FSMD, ossia il circuito che mette in collegamento il controllore e il circuito di elaborazione.



```
root@823ded275331:/data/sis/elaborato_sis_2019/src# sis
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> read_blif fsmd_src.blif
Warning: network `fsm.blif', node "FA" does not fanout
Warning: network `fsm.blif', node "FB" does not fanout
Warning: network `fsm.blif', node "FC" does not fanout
Warning: network `FSM', node "GO" does not fanout
Warning: network `FSMD', node "GO" does not fanout
sis> print_stats
FSMD          pi= 5   po= 5   nodes=100      latches=21
lits(sop)= 336
sis> 
```

Analogamente a quanto detto per il DATAPATH, gli avvertimenti prodotti da **sis** non sono preoccupanti. I bit di FA FB FC fanno in realtà fanout, l'avvertimento altro non è che un falso positivo dovuto alla modalità che è stata impiegata per lo scambio di segnali tra i due macrocomponenti.

Queste sono le statistiche della FSMD prima di mapping e ottimizzazione.

```
> ✎ ⤴ ~ : sh — Konsole
sis> read_library synch.genlib
sis> map -s
warning: unknown latch type at node '{{[11]}}' (RISING_EDGE assumed)
warning: unknown latch type at node '{{[12]}}' (RISING_EDGE assumed)
warning: unknown latch type at node '{{[13]}}' (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:      26
total gate area:   7008.00
maximum arrival time: (36.80,36.80)
maximum po slack:   (-8.00,-8.00)
minimum po slack:   (-36.80,-36.80)
total neg slack:    (-640.40,-640.40)
# of failing outputs: 26
>>> before removing parallel inverters <<<
# of outputs:      26
total gate area:   6960.00
maximum arrival time: (34.60,34.60)
maximum po slack:   (-8.00,-8.00)
minimum po slack:   (-34.60,-34.60)
total neg slack:    (-625.00,-625.00)
# of failing outputs: 26
# of outputs:      26
total gate area:   6832.00
maximum arrival time: (34.40,34.40)
maximum po slack:   (-8.00,-8.00)
minimum po slack:   (-34.40,-34.40)
total neg slack:    (-623.60,-623.60)
# of failing outputs: 26
sis> █
```

Queste sono le statistiche della FSMD mappata senza alcuna ottimizzazione.

```
~ : sh — Konsole
minimum po slack:      (-36.80,-36.80)
total neg slack:       (-640.40,-640.40)
# of failing outputs:   26
>>> before removing parallel inverters <<<
# of outputs:          26
total gate area:        6960.00
maximum arrival time:   (34.60,34.60)
maximum po slack:       (-8.00,-8.00)
minimum po slack:       (-34.60,-34.60)
total neg slack:        (-625.00,-625.00)
# of failing outputs:   26
# of outputs:          26
total gate area:        6832.00
maximum arrival time:   (34.40,34.40)
maximum po slack:       (-8.00,-8.00)
minimum po slack:       (-34.40,-34.40)
total neg slack:        (-623.60,-623.60)
# of failing outputs:   26
sis> source script.rugged
sis> print_stats
FSMD          pi= 5   po= 5   nodes= 94       latches=21
lits(sop)= 338
sis> 
```

Infine, si può notare che la FSMD mappata e ottimizzata presenta un numero inferiore di nodi.

## SCELTE PROGETTUALI

La priorità nel design del circuito è stata la semplicità dello stesso. Si è preferito utilizzare più segnali interni al circuito che aggiungere componenti al datapath per gestire in maniera differita un numero minore di segnali.

È stato inoltre rispettato il trend della specifica di riferirsi ai settori del parcheggio utilizzando segnali da 3 bit in codifica one-hot.

L'inizializzazione dei tre settori del parcheggio da eseguirsi nei primi tre cicli di clock è stata effettuata commutando tra tre stati diversi e implementando un protocollo di *handshake* tra FSM e DATAPATH caratterizzato dallo scambio tra il segnale `START[sector]` da FSM a DATAPATH e il segnale `G0` da DATAPATH a FSM che segnala l'avvenuta inizializzazione del registro corrispettivo al settore per commutare verso il prossimo stato.

Gli avvertimenti generati alla lettura dell modello FSMD secondo i quali i tre nodi FA FB FC non farebbero fanout sono falsi positivi dovuti al fatto che i tre segnali sono passati tra FSM e Datapath usando tre registri da un bit residenti onde evitare di generare un ciclo nel circuito.



Infine, ad ogni ciclo di clock il DATAPATH esegue il controllo di tutti e tre i settori, non solo di quello in questione, in modo da mandare in output i corretti segnali FA FB FC, ricordandosi dello stato dei settori tra cicli di clock.