PROGETTO SIS

CORSO: ARCHITETTURA DEGLI ELABORATORI

ANNO ACCADEMICO: 2022/2023

PARTECIPANTI :

- **ALEX MERLIN (*VR472547*)**
- **EDOARDO DALL'ORA (*VR489708*)**

CIRCUITO GENERALE

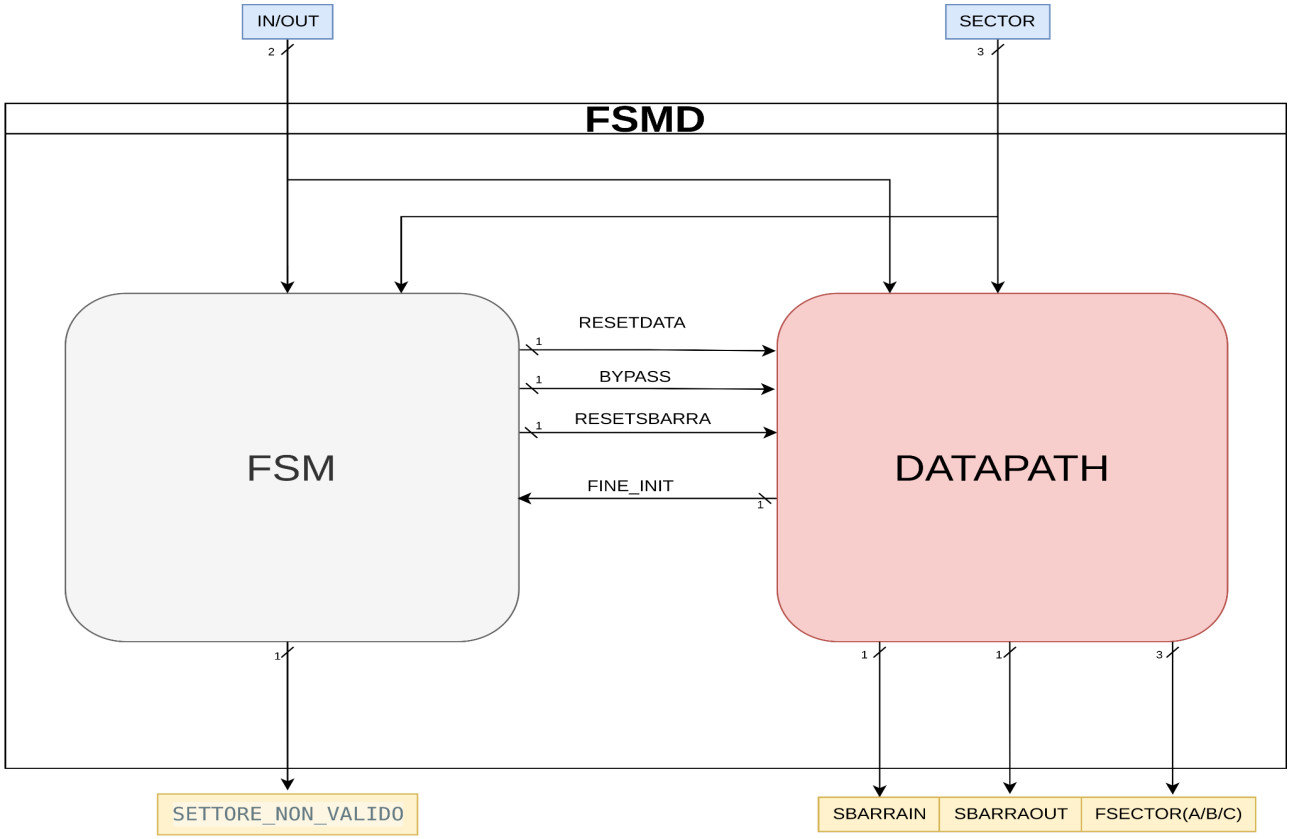
Il circuito generale è costituito da una **FSMD** (*Finite State Machine with Datapath*), che riceve in ingresso 5 bit, definiti nel seguente ordine :

- **IN/OUT [2 bit] :** con la sequenza 01 l'utente richiede di entrare nel parcheggio. Con la sequenza 10 si richiede l'uscita dal parcheggio. Richieste diverse saranno ignorate;
- **SECTOR [3 bit]:** la sequenza 100 indica il parcheggio A, la sequenza 010 indica il parcheggio B e 001 indica il parcheggio C.
  Entrate diverse verranno considerate come errore.

L'output della macchina sarà di 6 bit definiti nel seguente ordine:

- **SETTORE_NON_VALIDO [1 bit]:** il bit è impostato a 1 se l'ingresso **SECTOR [3 bit]** presenta valori diversi da 100, 010 o 001;
- **SBARRAIN [1 bit]:** questo bit assume valore 0 se la sbarra in entrata e chiusa, 1 se aperta;
- **SBARRAOUT [1 bit]:** questo bit assume valore 0 se la sbarra in uscita è chiusa, 1 se aperta;
- **FSECTOR(A/B/C) [1 bit per settore]:** quando il parcheggio è pieno il bit viene alzato.

1



[1] Grafico generale FSMD

FSM

Il controllore è composto da una FSM di Mealy in cui abbiamo identificato 6 input, 3 output e 3 stati: **SPENTO, INIT (*inizializzazione*), STAN (*stand-by*).**

INPUT:

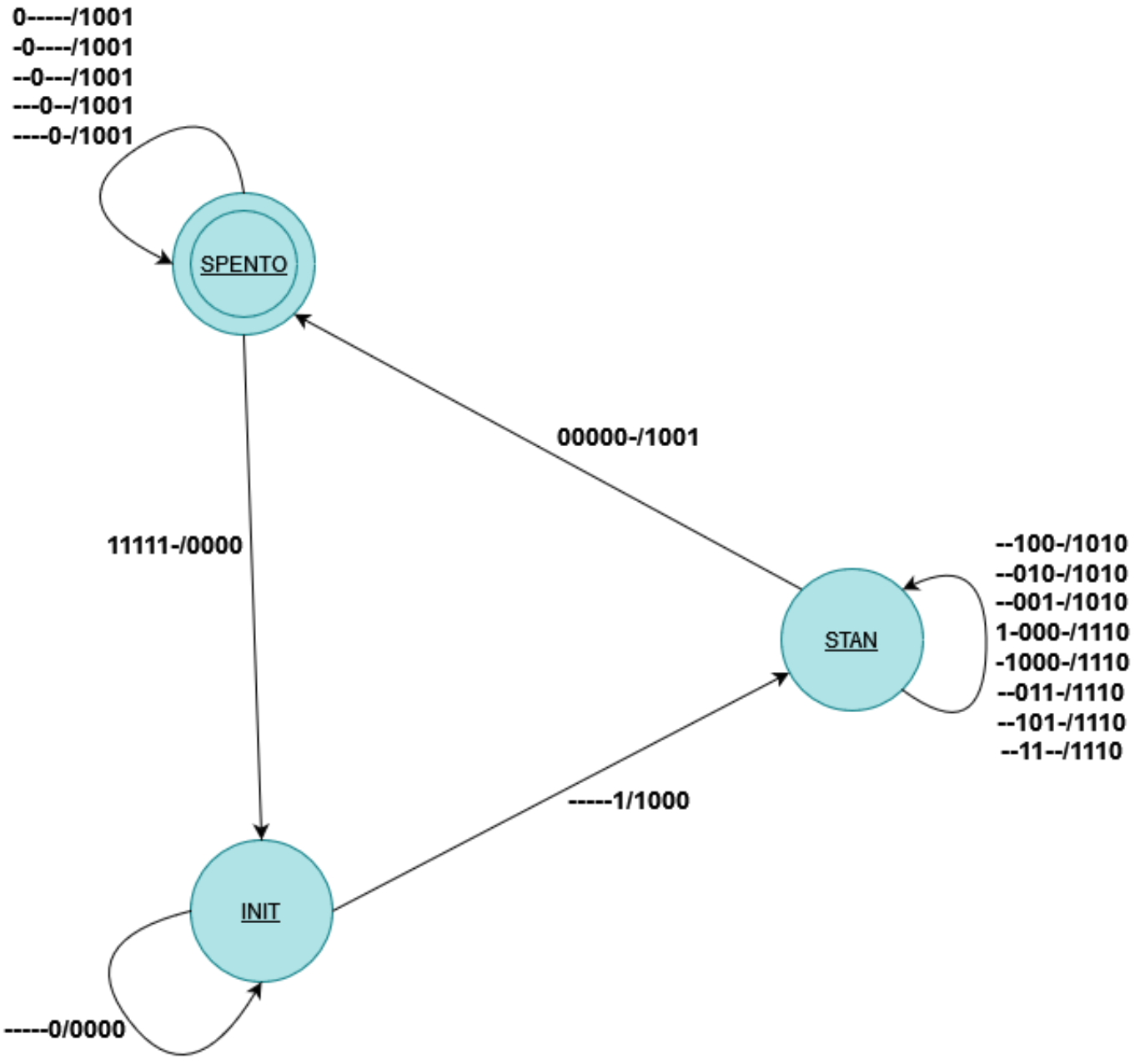
- **IN4/IN3 [2 bit]:** corrispondono alle variabili **IN/OUT** in entrata alla FSMD;
- **IN2/IN1/IN0 [3 bit]:** corrispondono alle variabili **SECTOR** in entrata nella FSMD;
- **FINEINIT [1 bit]:** bit in uscita dal datapath. Indica la fine della modalità operatore. Quando è alzato, se la macchina è nello stato **INIT**, passa allo stato **STAN.**

OUTPUT:

- **RESETDATA [1 bit]:** bit in ingresso al datapath. Il bit è posto a 0 quando la macchina passa allo stato **INIT** ed in questo momento la macchina entra nella modalità operatore;
- **SETTORE_NON_VALIDO [1 bit]:** bit in uscita alla FSMD. Quando il bit è alzato significa che è stato inserito un settore non valido(quindi diverso da 100,010,001);
- **RESETSBARRA [1 bit]:** bit che andrà in ingresso al DATAPATH. Viene impostato a 1 solo nello stato di **STAN**. Gestisce l'apertura delle sbarre.
- **BYPASS [1 bit]:** bit in ingresso al datapath. Quando alzato indica che la FSMD è spenta, comunicando al datapath di azzerare i registri dei 3 settori.

STATI:

- **SPENTO:** stato iniziale della macchina. La macchina rimarrà in questo stato fino a che non verrà inserita la sequenza 11111 nella FSMD, che farà passare la FSM allo stato di INIT;
- **INIT:** stato di inizializzazione in cui l'operatore inserisce manualmente il numero di parcheggi occupati. Nel momento in cui viene inserito il valore dell'ultimo parcheggio la macchina passa allo stato **STAN**;
- **STAN:** in questo stato la macchina è nel suo funzionamento normale, in attesa di un utente che entri o esca dal parcheggio. Inserendo la sequenza 00000 nella FSMD la FSM torna allo stato SPENTO.



2

[2] Grafico STG della FSM

DATAPATH

L'unità di elaborazione (*datapath*) è ottenuta aggregando componenti elementari per la memorizzazione (*registri*) e l'elaborazione (*unità funzionali*).

COMPONENTI:

- **REGISTRI (*REG*):** registri per la memorizzazione dei dati;
- **MULTIPLEXER (*MUX*):** selettori di linee di ingresso;
- **SOMMATORI (+):** restituiscono la somma degli ingressi
- **CLOCK:** genera un clock, alternativo a quello del comando *simulate*, che varia al variare della combinazione di ingressi;
- **COMPARATORI (=):** restituiscono in uscita il risultato della comparazione tra gli elementi in ingresso;
- **CONST SELECT:** componenti che restituiscono una costante diversa al variare delle sue entrate;
- **SELETTORE PARZIALE/INVERTITO:** componente per la gestione dei casi di underflow/overflow;
- **SBARRAINOUT:** componente che controlla l'apertura delle sbarre al variare degli ingressi;
- **PORTE AND, OR**

INPUT:

-dalla FSMD (*vedere capitolo FSMD*):

- **IN/OUT [2 bit];**
- **SECTOR [3 bit];**

-dalla FSM (*vedere capitolo FSM*):

- **RESETDATA [1 bit];**
- **RESETSBARRA [1 bit];**
- **BYPASS [1 bit];**

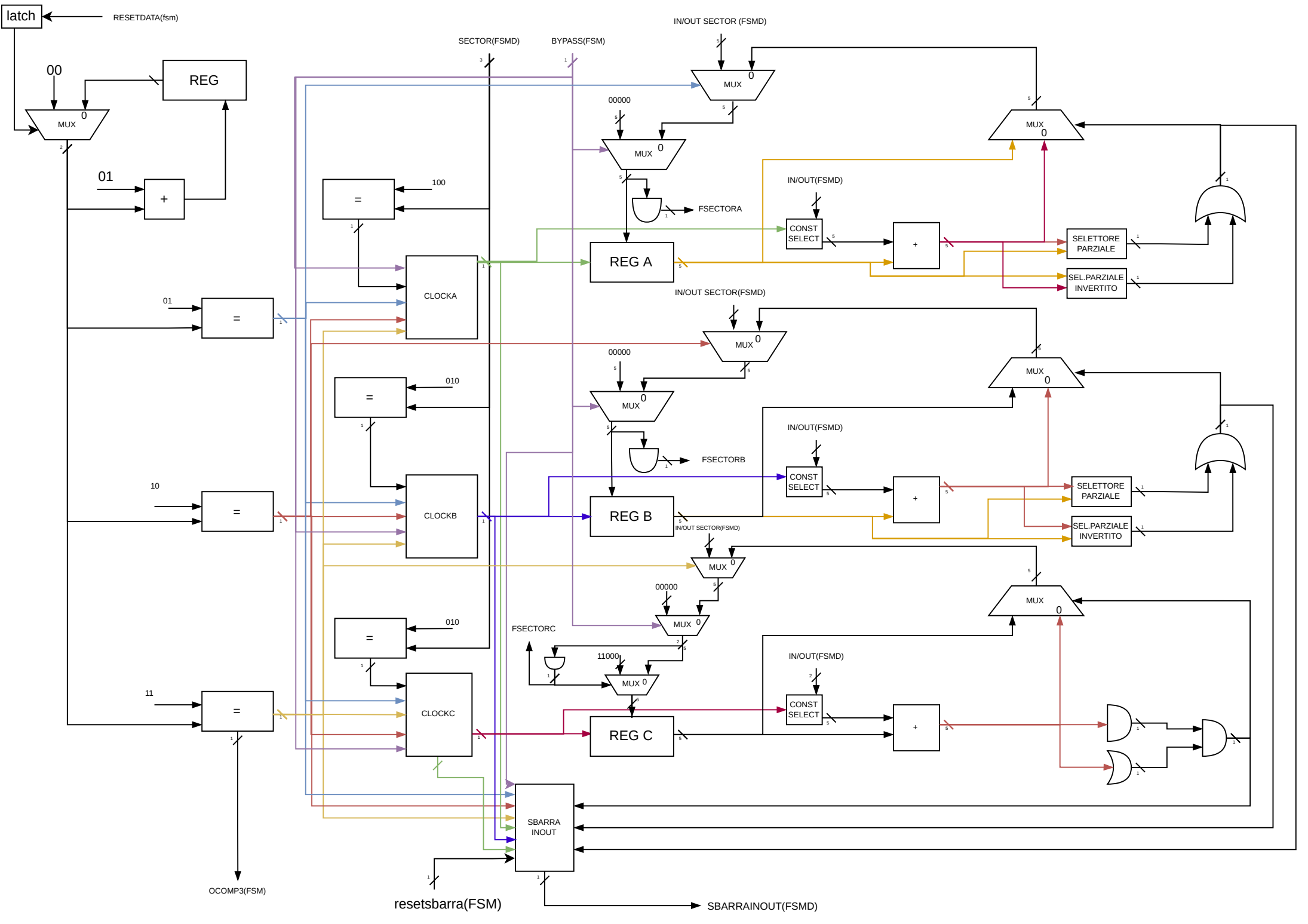
OUTPUT:

-verso la FSMD:

- **FSECTOR (A/B/C) [1 bit per settore]:** indica quando un settore è pieno;
- **SBARRAIN [1 bit]:** questo bit assume valore 0 se la sbarra in entrata e chiusa, 1 se aperta;
- **SBARRAOUT [1 bit]:** questo bit assume valore 0 se la sbarra in uscita è chiusa, 1 se aperta;

-verso la FSM**:**

- **OCOMP3 [1 bit]:** bit in ingresso alla FSM (*variabile **FINEINIT** in entrata all' FSM*). Quando l'uscita è 1 significa che sono passati 3 cicli di clock dall'accensione, quindi finisce la modalità operatore.


latch
RESETDATA(fsm)
00
REG
MUX
0
01
+
SECTOR(FSMD)
BYPASS(FSM)
IN/OUT SECTOR (FSMD)
00000
100
=
FSECTORA
IN/OUT(FSMD)
CONST SELECT
REG A
SELETTORE PARZIALE
SEL.PARZIALE INVERTITO
CLOCKA
01
010
FSECTORB
REG B
10
CLOCKB
FSECTORC
11000
REG C
11
CLOCKC
SBARRA INOUT
OCOMP3(FSM)
resetsbarra(FSM)
SBARRAINOUT(FSMD)

STATISTICHE DEL CIRCUITO

FSM

Statistiche pre-ottimizzazione, ma dopo il comando *state_minimize stamina(dopo il quale non ci sono state variazione del numero di stati)* e *state_assign jedi*:

```
sis> print_stats
FSM             pi= 6   po= 4   nodes=  6       latches= 2
lits(sop)= 139  #states(STG)=   3
sis>
```

Statistiche post ottimizzazione:

```
lits(sop)= 139  #states(STG)=   3
sis> full_simplify
sis> source script.rugged
sis> print_stats
FSM             pi= 6   po= 4   nodes=  9       latches= 2
lits(sop)=  32  #states(STG)=   3
sis>
```

La richiesta dell'elaborato era di ottimizzare per area e la miglior combinazione che abbiamo trovato è risultata con i comandi: *full_simplify* e *script.rugged.*

DATAPATH

Statistiche pre-ottimizzazione:

```
sis> print_stats
DATAPATH        pi= 8   po= 9   nodes=167       latches=18
lits(sop)= 766
sis>
```

Statistiche post-ottimizzazione:

```
sis> source script.rugged
sis> fx
sis> print_stats
DATAPATH        pi= 8   po= 9   nodes= 95       latches=18
lits(sop)= 375
sis>
```

Come nella FSM, oltre allo '*script.rugged'* abbiamo deciso di usare il comando '*fx'* per migliorare l'ottimizzazione per area.

FSMD

Statistiche pre-ottimizzazione:

```
Warning: network `fsm1.blif', node "IN1" does not fanout
Warning: network `fsm1.blif', node "IN0" does not fanout
Warning: network `fsm1.blif', node "FINEINIT" does not fanout
sis> print_stats
FSMD            pi= 5   po= 9   nodes=104       latches=20
lits(sop)= 407
sis>
```

Statistiche post-ottimizzazione:

```
sis> source script.rugged
sis> full_simplify
sis> fx
sis> decomp
sis> print_stats
FSMD            pi= 5   po= 9   nodes=110       latches=20
lits(sop)= 396
sis>
```

Si noti come, anche in questo caso, abbiamo deciso di ridurre al massimo il numero di letterali.
Per questo, oltre allo '*script.rugged*' e '*fx*' visti in precedenza, ci siamo serviti dei comandi *'full_simplify' e 'decomp'.*

LEGENDA COMANDI:

- *script.rugged*: sequenza preordinata di comandi eseguibili come unica operazione.Questo script in particolare itera algoritmi algebrici di ristrutturazione ad algoritmi booleani di minimizzazione. Rimane lo script più efficace eseguibile su SIS;
- *fx*: comando di *estrazione*, utilizzato per trovare una sottoespressione comune di due espressioni che è associata a due nodi diversi, creando un nuovo nodo;
- *full_simplify*: comando di *semplificazione* che riduce la complessità di una funzione
- *decomp*: comando per decomporre i letterali interni a una funzione in più nodi, questo può far diminuire il numero di letterali, a fronte di un aumento dei nodi.

MAPPING

Dopo l'ottimizzazione dobbiamo mappare il progetto, in modo che questo venga sintetizzato con componenti realmente stampabili.
Come richiesto abbiamo utilizzato la libreria *'sync.genlib'* e mappato per area con il comando *'map -m 0'.*

Le statistiche dopo la mappatura sono le seguenti:

```
>>> before removing serial inverters <<<
# of outputs:          29
total gate area:       8208.00
maximum arrival time: (46.00,46.00)
maximum po slack:     (-4.80,-4.80)
minimum po slack:     (-46.00,-46.00)
total neg slack:      (-943.40,-943.40)
# of failing outputs:  29
>>> before removing parallel inverters <<<
# of outputs:          29
total gate area:       8176.00
maximum arrival time: (46.00,46.00)
maximum po slack:     (-4.80,-4.80)
minimum po slack:     (-46.00,-46.00)
total neg slack:      (-944.60,-944.60)
# of failing outputs:  29
# of outputs:          29
total gate area:       7680.00
maximum arrival time: (44.80,44.80)
maximum po slack:     (-4.80,-4.80)
minimum po slack:     (-44.80,-44.80)
total neg slack:      (-921.60,-921.60)
# of failing outputs:  29
```

In particolare notiamo:

- total gate area: **8208.00**
- maximum arrival time: **46.00**

SCELTE PROGETTUALI

In conclusione vorremmo dare delle motivazioni alle scelte fatte precedentemente:

- Visto che durante la notte il parcheggio rimane libero, abbiamo deciso di lasciare aperte entrambe le sbarre. Quando la macchina è spenta l'output della FSMD sarà 011000, in risposta ad ogni sequenza diversa da quella di accensione;

- Per effettuare somma e sottrazione delle macchine in entrata e in uscita, ci siamo serviti di un unico elemento sommatore grazie alle proprietà della codifica complemento a 2;

- Per rendere più chiaro il circuito del datapath abbiamo deciso,nei punti più critici, di colore le frecce che collegano i vari componenti.

- Abbiamo notato che otteniamo ottimizzazioni migliori se lavoriamo sui singoli componenti (FSM, DATAPATH, FSMD) invece di ottimizzare solo la FSMD.