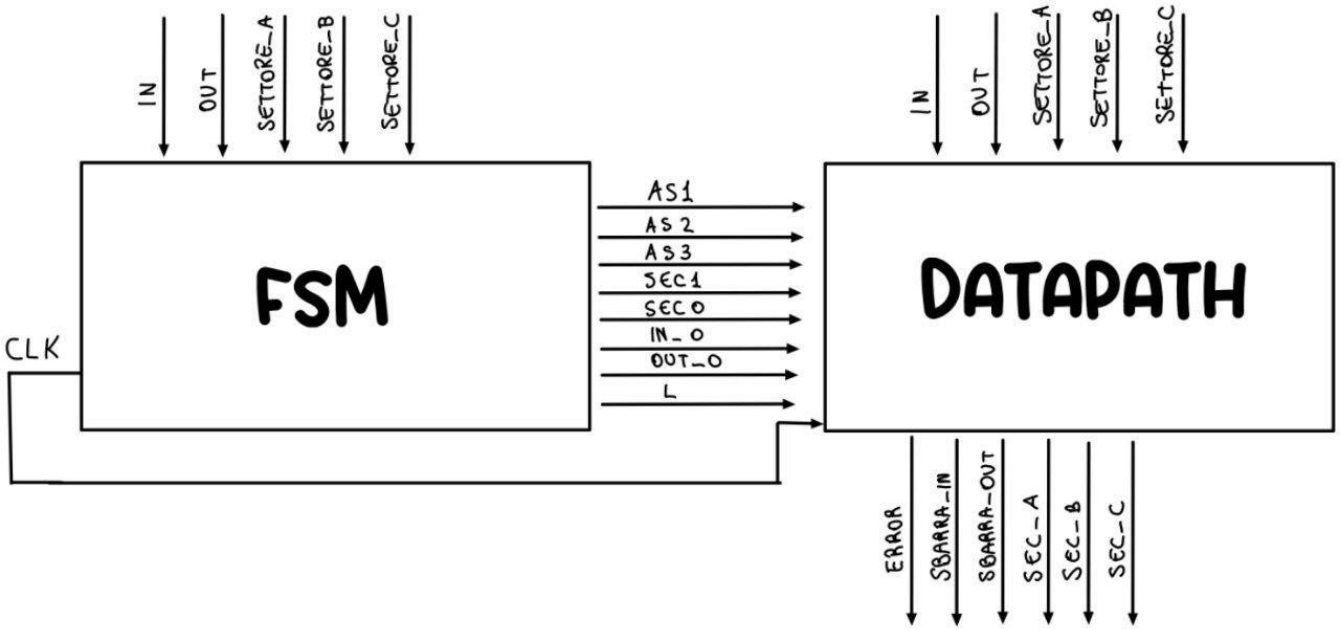
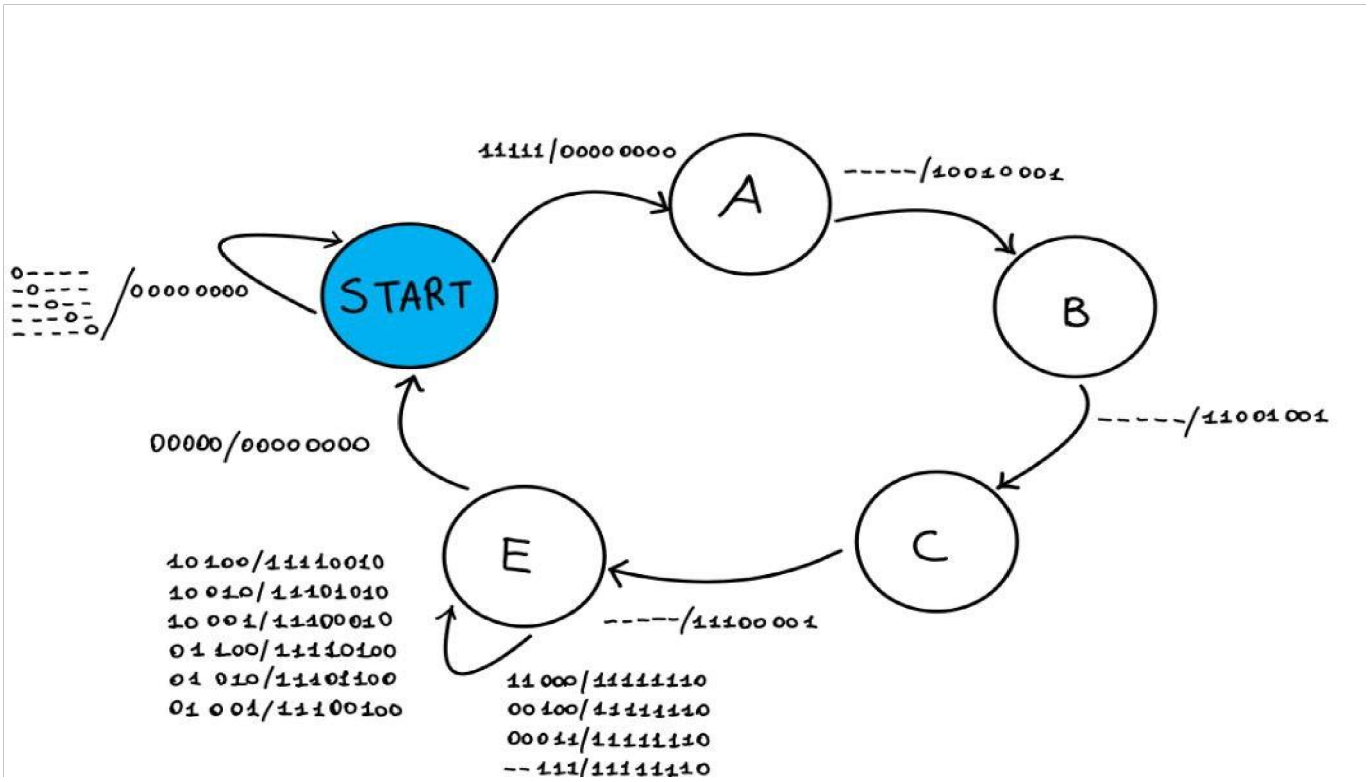
FSMD



I bit di input li mandiamo sia nella fsm che nel datapath. Gli output richiesti usciranno tutti dal datapath.
Per collegare FSM a DATAPATH abbiamo utilizzato 8 bit di controllo: AS1,AS2,AS3 per attivare nel datapath un componente che mi permette di capire se il mio parcheggio è pieno o no. SEC0 SEC1 i quali mi identificano il settore e se la combinazione è valida, IN_O OUT_O per capire la validità del comando di input e output. L per capire se devo scrivere nei registri oppure no.

FSM



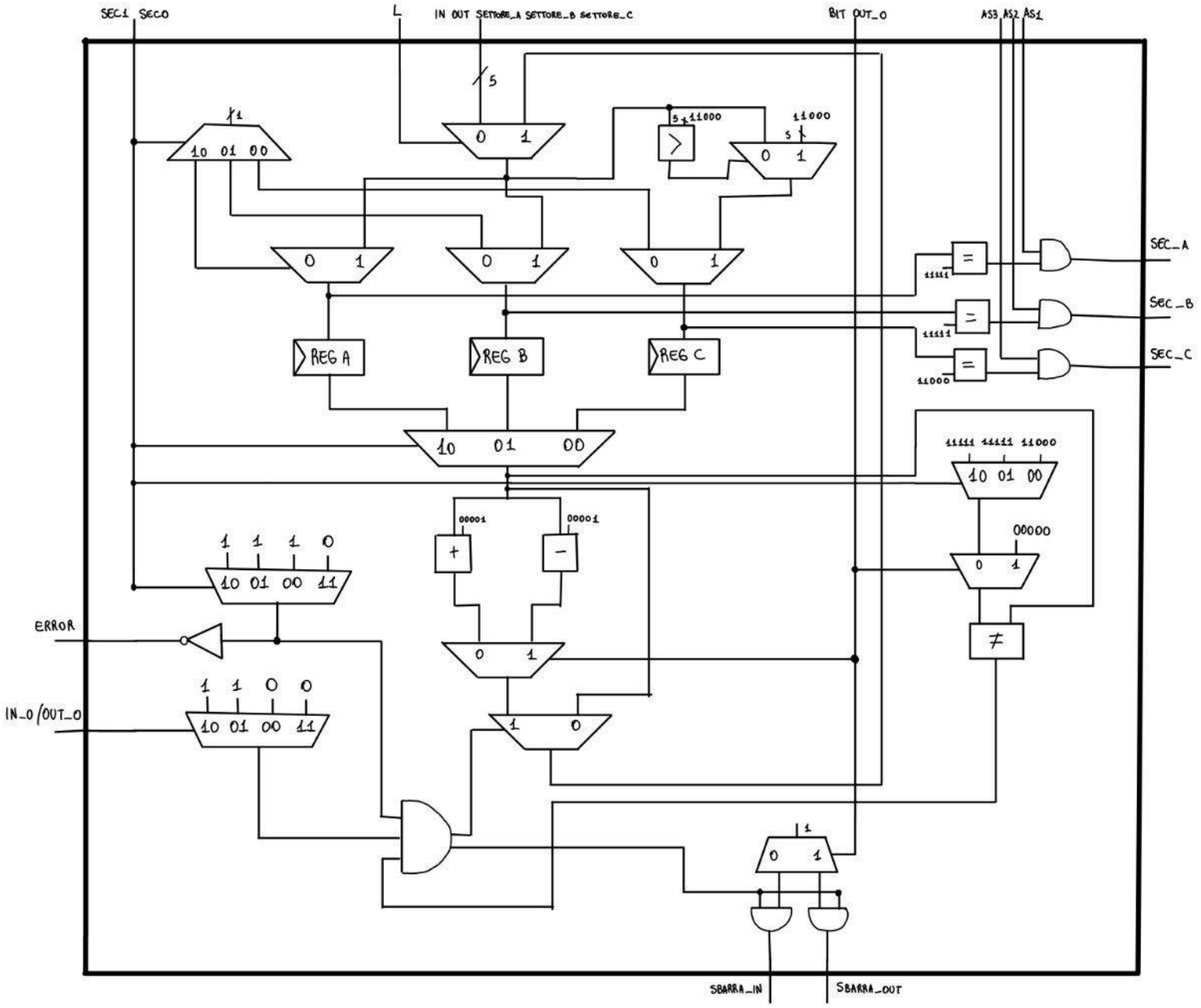
Questa è la STG(State Transition Graph) che illustra graficamente i comportamenti della nostra FSM di tipo Mealy.

La FSM è dotata di 5 stati: **START**, **A**, **B**, **C**, **E**.

Ecco una rapida descrizione del comportamento:

Nello stato **START** solamente l'inserimento della sequenza **11111** permetterà di transitare verso lo stato successivo ovvero **A**, per ogni altra sequenza ci sarà una transizione verso se stesso. Per accedere agli stati successivi ovvero **A - B**, **B - C**, **C - E** viene richiesto di inserire 5 bit corrispondenti al numero di macchine. Questo dato verrà salvato nell'apposito registro del Datapath.
Finito l'inserimento delle macchine si passa allo stato **E** dove vengono fatte le principali operazioni in particolare se una macchina vuole **uscire** o **entrare** e gli eventuali errori sia di input che di settore.
Quando poi viene inserita la sequenza **00000** si ritorna allo stato **START**.

DATAPATH



I bit di controllo **AS1 AS2 AS3 (**tra di loro separati)corrispondenti ai vari settori entrano nelle porte AND e se sono a 0 l'output dei settori è zero (anche se il bit di uguaglianza è 1) altrimenti se è 1 e anche quello di uguaglianza è 1 l'output sarà 1 e quindi da qui potremo verificare se i parcheggi sono vuoti o pieni.

Il primo multiplexer centrale, attraverso il bit di controllo L, determina se devo prendere i bit in ingresso o quelli memorizzati nel registro. Attraverso un demultiplexer con i bit di controllo **SEC1** e **SEC0**, decido in quale settore memorizzarli.

Il mux sotto i registri e il demux che riceve un bit, ed entrambi ricevono come bit di controllo **SEC1** e **SEC0** mi permette di scegliere con quale settore eseguire le mie operazioni mentre gli altri rimangono invariati.

Viene utilizzato un bit della macchina a stati finiti, chiamato bit **OUT_O**, che, attraverso il multiplexer con le due operazioni, determina se sono in fase di ingresso o di uscita. Se è 0, allora sono in fase di ingresso e sommo il valore. Altrimenti, se sono in fase di uscita, sottraggo il valore.

Il risultato ottenuto viene inviato a un altro demultiplexer collegato a una porta AND a tre ingressi che funge da controllo. Se il bit di controllo è 1, accetta il valore del demultiplexer precedente, altrimenti riprende il valore iniziale. In entrambi i casi, il risultato ritorna al primo demultiplexer.

Il bit della porta AND a tre ingressi e il bit **OUT_O** vengono utilizzati nel demultiplexer, che produce in output i segnali **SBARRA_IN** e **SBARRA_OUT**. Il bit **OUT_O** viene utilizzato per determinare se l'operazione è di ingresso o d'uscita, mentre il bit della porta AND è collegato a due porte AND che includono anche il bit del demultiplexer. Pertanto, se non ci sono stati errori e, ad esempio, l'operazione è di ingresso, la porta AND di SBARRA_IN produce in output 1 e l'altra 0. In caso di errori, la porta AND produrrà 0 come risultato, che viene quindi fornito in output a **SBARRA_IN**.

Abbiamo un multiplexer che riceve i bit di controllo **SEC1** e **SEC0** dalla macchina a stati finiti. Se ricevo la sequenza 00, significa che si è verificato un errore nell'inserimento del settore, quindi il risultato sarà 0. Il bit ottenuto, oltre a essere utilizzato nella porta AND, viene negato tramite un NOT e fornito in output con il nome di **ERROR**.

Abbiamo un mux che riceve i bit di controllo della fsm **IN_O** e **OUT_O** il quale controlla se l’inserimento e se ricevo la sequenza 00 vuol dire che c’è stato un errore nell'inserimento e da 0. Il bit che ottengo va a finire nella porta and.

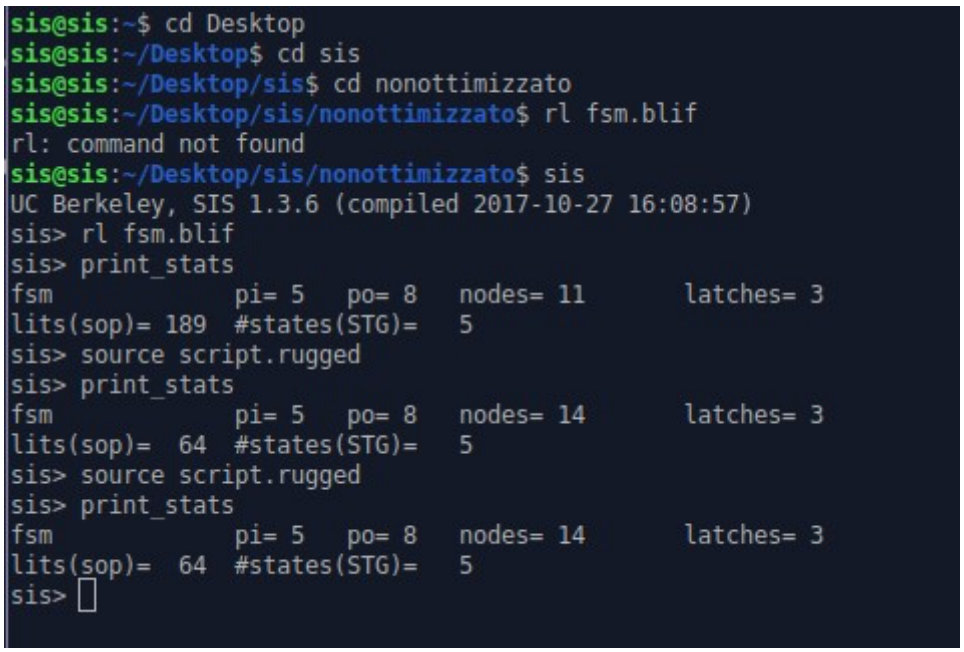
L’ultimo bit della porta and l’ottengo attraverso un operatore di disuguaglianza il quale prende il valore del settore in cui sono e lo confronta o con la costante 00000 nel caso in cui sono in uscita, anche qui viene usato il bit di controllo **OUT_O,** oppure viene confrontato con i valori dei parcheggi, scelti anche qui attraverso un mux con i bit di controllo **SEC1** e **SEC0.**

Nel settore C, per evitare un overflow, utilizziamo un comparatore di maggioranza che confronta il valore ricevuto con la costante 11000. Il bit risultante viene inserito nel

multiplexer associato, in modo che se è 0, viene preso il valore di input, altrimenti viene inserito 11000 nel settore C.

OTTIMIZZAZIONE

FSM



DATAPATH

```
sis> rl datapath.blif
Warning: `datapath', ".subckt mux4": formal input "D4" not driven
Warning: `datapath', ".subckt mux4": formal input "D3" not driven
Warning: `datapath', ".subckt mux4": formal input "D2" not driven
Warning: `datapath', ".subckt mux4": formal input "D1" not driven
Warning: `datapath', ".subckt mux4": formal input "D0" not driven
Warning: `datapath', ".subckt mux4": formal input "D4" not driven
Warning: `datapath', ".subckt mux4": formal input "D3" not driven
Warning: `datapath', ".subckt mux4": formal input "D2" not driven
Warning: `datapath', ".subckt mux4": formal input "D1" not driven
Warning: `datapath', ".subckt mux4": formal input "D0" not driven
Warning: network `and3', node "C" does not fanout
Warning: network `datapath', node "NULL_D" does not fanout
Warning: network `datapath', node "D4" is not driven (zero assumed)
Warning: network `datapath', node "D3" is not driven (zero assumed)
Warning: network `datapath', node "D2" is not driven (zero assumed)
Warning: network `datapath', node "D1" is not driven (zero assumed)
Warning: network `datapath', node "D0" is not driven (zero assumed)
Warning: network `datapath', node "[109]" is not driven (zero assumed)
Warning: network `datapath', node "[110]" is not driven (zero assumed)
Warning: network `datapath', node "[111]" is not driven (zero assumed)
Warning: network `datapath', node "[112]" is not driven (zero assumed)
Warning: network `datapath', node "[113]" is not driven (zero assumed)
Warning: network `datapath', node "[119]" does not fanout
Warning: network `datapath', node "[139]" does not fanout
sis> print_stats
datapath        pi=13   po= 6   nodes=160       latches=15
lits(sop)= 634
sis> source script.rugged
sis> print_stats
datapath        pi=13   po= 6   nodes= 51       latches=15
lits(sop)= 251
sis> source script.rugged
sis> print_stats
datapath        pi=13   po= 6   nodes= 49       latches=15
lits(sop)= 242
sis> source script.rugged
sis> print_stats
datapath        pi=13   po= 6   nodes= 49       latches=15
lits(sop)= 245
sis>
```

FSMD

```
Warning: network `FSMD', node "NULL_D" does not fanout
Warning: network `FSMD', node "[368]" does not fanout
Warning: network `FSMD', node "[388]" does not fanout
sis> print_stats
FSMD            pi= 5   po= 6   nodes=171       latches=18
lits(sop)= 823
sis> source script.rugged
sis> print_stats
FSMD            pi= 5   po= 6   nodes= 62       latches=18
lits(sop)= 303
sis> source script.rugged
sis> print_stats
FSMD            pi= 5   po= 6   nodes= 59       latches=18
lits(sop)= 304
sis> write_blif FSMDott.blif
```

MAPPING

Mappatura FSMD dopo ottimizzazione: