Relazione elaborato del corso di Architettura degli Elaboratori

Anno accademico 2019/2020

Studenti:

Enrico Pachera, matricola VR422511

Alessandro Marconcini, matricola VR421504

SOMMARIO

Architettura generale del circuito

Il dispositivo da noi implementato si occupa della gestione di un parcheggio con ingresso e uscita automatizzati.

Non appena esso viene acceso da un operatore, esso memorizza per tre clicli di clock la quantità di automobili presenti in quell'istante in tre differenti settori, due che contengono un massimo di 31 posti auto e l'ultimo che ne contiente solamente massimo 24, questa fase è stata definita da noi fase di caricamento.

Nel momento in cui la fase di caricamento termina (Sono passati i primi tre cicli di clock) al quarto il dispositivo cambia comportamento e non tornerà più nella fase di caricamento prima di essere

spento. La nuova fase in cui entra è la fase di lavoro vera e propria, dove un utente può decidere di recarsi all'entrata del parcheggio e fare richiesta per entrare, come può decidere di uscire una volta entrato. Il dispositivo si occupa anche di tener conto della scelta del settore in cui l'utente vuole parcheggiare in entrata e il settore dal quale l'utente proviene durante l'uscita.

Il dispositivo reagisce ad entrata e uscita attraverso l'incremento o decremento rispettivo dei registri univcoci legati ai settori. Se il settore in cui si vuole entrare possiede almeno un posto libero, allora la richiesta viene soddisfatta e la sbarra di entrata si apre,vi è una sbarra anche in uscita.

Le sbarre non si aprono quando vi è un inserimento errato oppure nel caso dell'entrata quando il settore scelto è pieno.

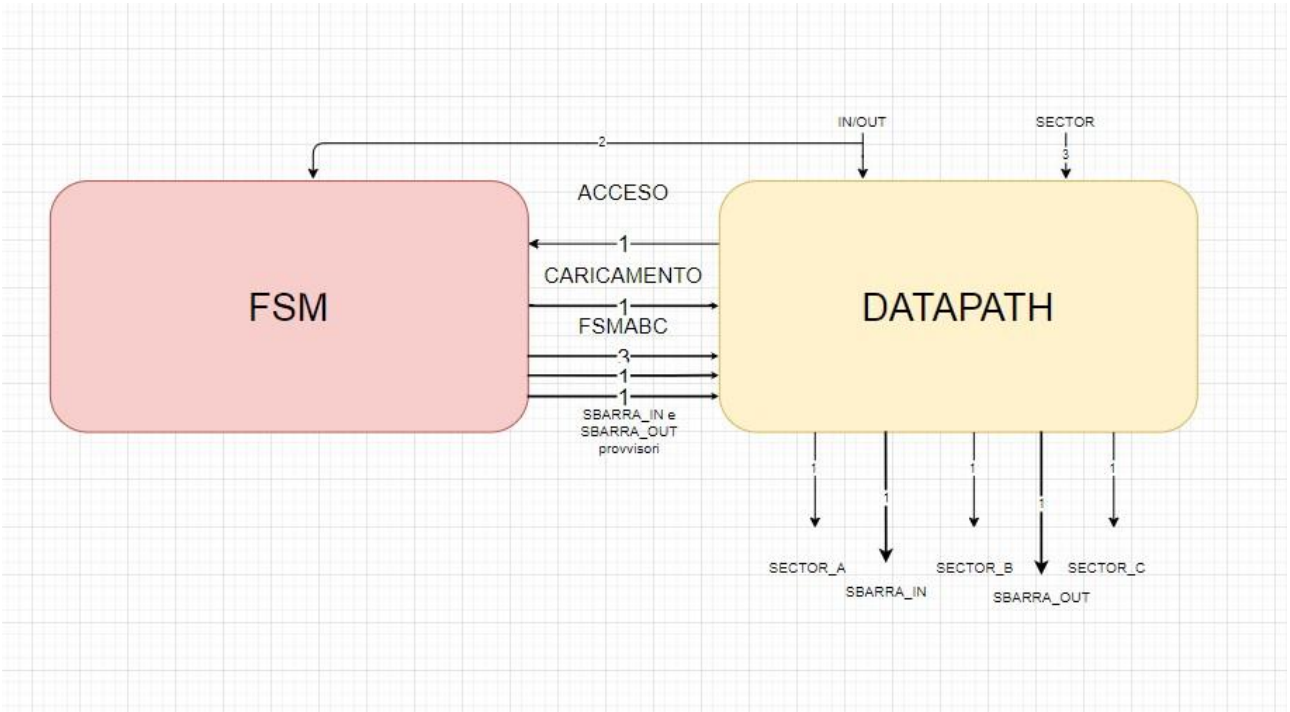
Il circuito presenta i seguenti input e output:

- IN/OUT[2], i quali rappresentano per le codifiche **01** e **10** rispettivamente **entrata** e **uscita** del parcheggio, se questi bit assumono **00** oppure **11,** in entrambi i casi la richiesta viene trattata come anomalia e quindi ignorata, essi vengono forniti sia al componente datapath che al controllore
- SECTOR[3], i quali rappresentano univocamente i segnali dei rispettivi settori A,B e C, in particolare se esso assume la codifica **100** ,allora essa identifica il settore A,**010**, allora in quel caso identifica il settore B e **001** è il caso in cui viene identificato C, tutte le altre codifiche non richiamano a nessun settore fra gli appena citati,anche questo segnale viene fornito al datapath
- SECTOR_A[1], output che si riferisce al registro legato al settore A, esso se è 1 denota il fatto che il settore sia pieno,esce dal datapath
- SECTOR_B[1], output che si riferisce al registro legato al settore B, esso se è 1 denota il fatto che il settore sia pieno,esce dal datapath
- SECTOR_C[1], output che si riferisce al registro legato al settore C, esso se è 1 denota il fatto che il settore sia pieno,esce dal datapath
- SBARRA_IN[1], è il segnale che fa alzare la sbarra in entrata per poter andare a parcheggiare,esso assume 1 solamente se c'è almeno un posto all'interno del parcheggio
- SBARRA_OUT[1],è il segnale che fa alzare la sbarra di uscita del parcheggio

Segnali di comunicazione fra controllore e datapath e viceversa:

- ACCESO[1],questo segnale si occupa di accendere la parte riguardante il controllore, esso viene posto a 1 se il datapath riceve in ingresso una sequenza **11111**,viene posto a 0 se la sequenza è **00000**,in tutti gli altri casi esso rimane invariato rispetto al ciclo di clock precedente
- CARICAMENTO[1],questo segnale proviene dal controllore e diventa 1 per tre cicli di clock successivi, esso è diretto al datapath affinchè avvenga la sovrascrittura del registro del ciclo corrispondente (In ordine A,B e C) e poi viene posto a 0 dal controllore una volta avvenuto il quarto ciclo
- FSMABC[3],questo segnale è un segnale di supporto che abbiamo deciso di far generare al controllore per la gestione della selezione dei registri nella fase di caricamento,esso si compone di tre bit ed essi sostituiscono in quel caso i tre bit di SECTOR[3].
- SBARRA_IN[1] provvisorio, un segnale utilizzato inizialmente per capire il comportamento del controllore , tuttora inutilizzato

- SBARRA_OUT[1] provvisorio, che viene utilizzato per ricavare l'output omonimo definitivo all'interno del datapath



Controllore

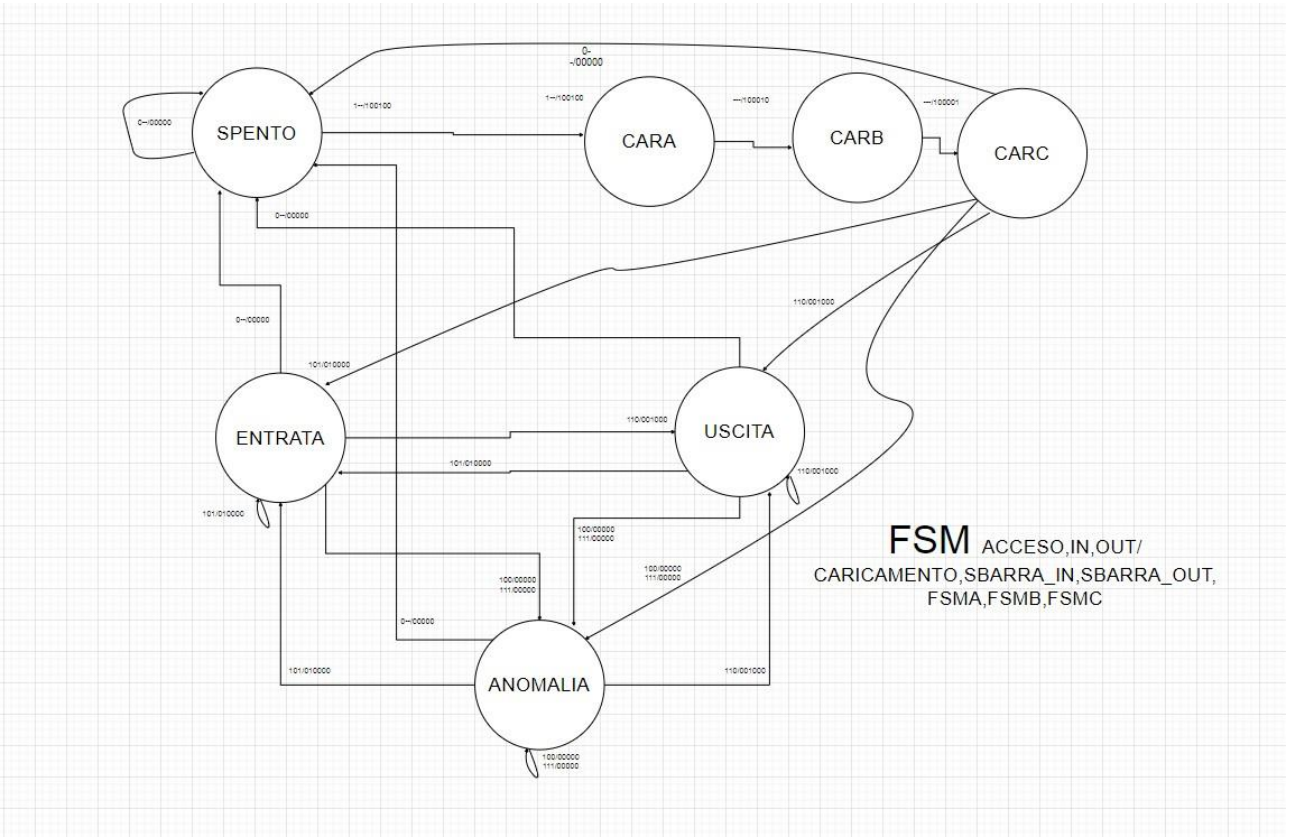
Il controllore è una FSM o Finite State Machine, ovvero una macchina a stati finiti. Il suo funzionamento prevede che a determinati input esso reagisca con un cambiamento o meno di stato attraverso delle transizioni specifiche e che in base alle transizioni produca un output ben specifico.

La macchina a stati finiti utilizzata è una macchina di Mealy.

Il nostro controllore è composto da tre ingressi che sono ACCESO,IN e OUT,mentre produce come uscite CARICAMENTO,SBARRA_IN e SBARRA_OUT provvisori,FSMA,FSMB,FSMC(compongono FSMABC).
Senza minimizzazione esso era formato di 7 stati che riportiamo qui sotto:

- **SPENTO**: lo stato in cui il dispositivo non è funzionante e il parcheggio viene lasciato libero per gli utente che vogliono utilizzarlo,è lo stato di reset e ad esso si può accedere da se stesso oppure dopo la fase di caricamento

- **CARICAMENTO A** : il primo dei tre stati relativi alla fase di caricamento, dove le uscite consentono poi al datapath di registrare quante automobili sono parcheggiate nel settore A
- **CARICAMENTO B:** il secondo dei tre stati relativi della fase di caricamento,esso permette la registrazione del numero di auto in B
- **CARICAMENTO C:** il terzo e ultimo stato della fase di caricamento,esso permette di registrare il numero di auto presenti nel settore C e anche di passare alla fase successiva nel quarto ciclo di clock,a partire da questo stato la macchina può essere spenta in qualunque momento(stato di SPENTO)
- **ENTRATA:** lo stato dedicato alle richieste di entrata,è lo stato in cui l'utente attua una richiesta di entrata verso un qualsiasi settore e il datapath verificherà se aprire le sbarre o meno
- **USCITA:** lo stato in cui avviene una richiesta di uscita da parte dell'utente ed essa è concessa a meno che non ci siano errori anomali di inserimento
- **ANOMALIA:** lo stato in cui l'utente ha fatto una richiesta in entrata o in uscita al parcheggio e questa è stata trovata anomala per un inserimento errato della richiesta, questo stato non apre mai nessuna sbarra

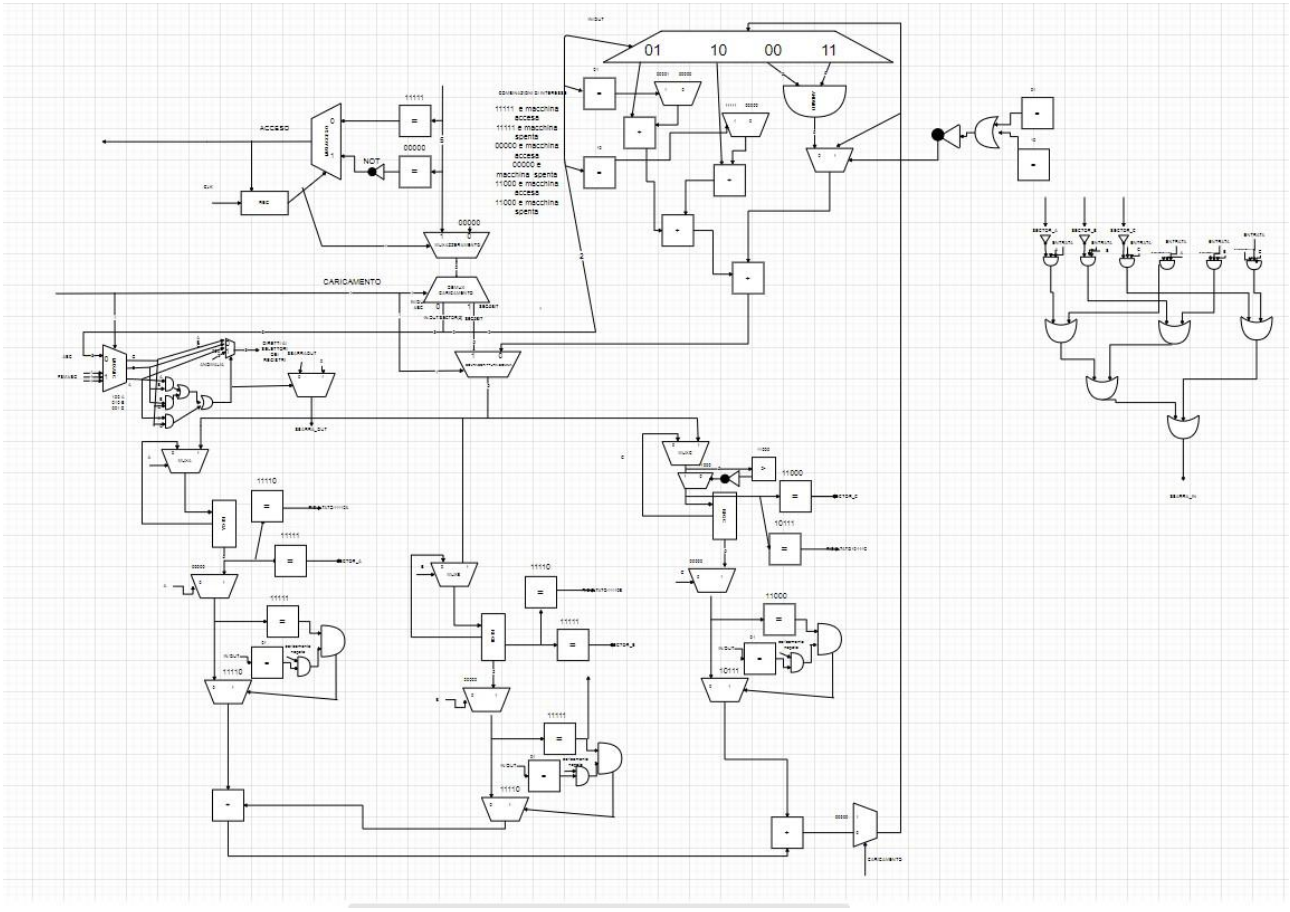


Datapath

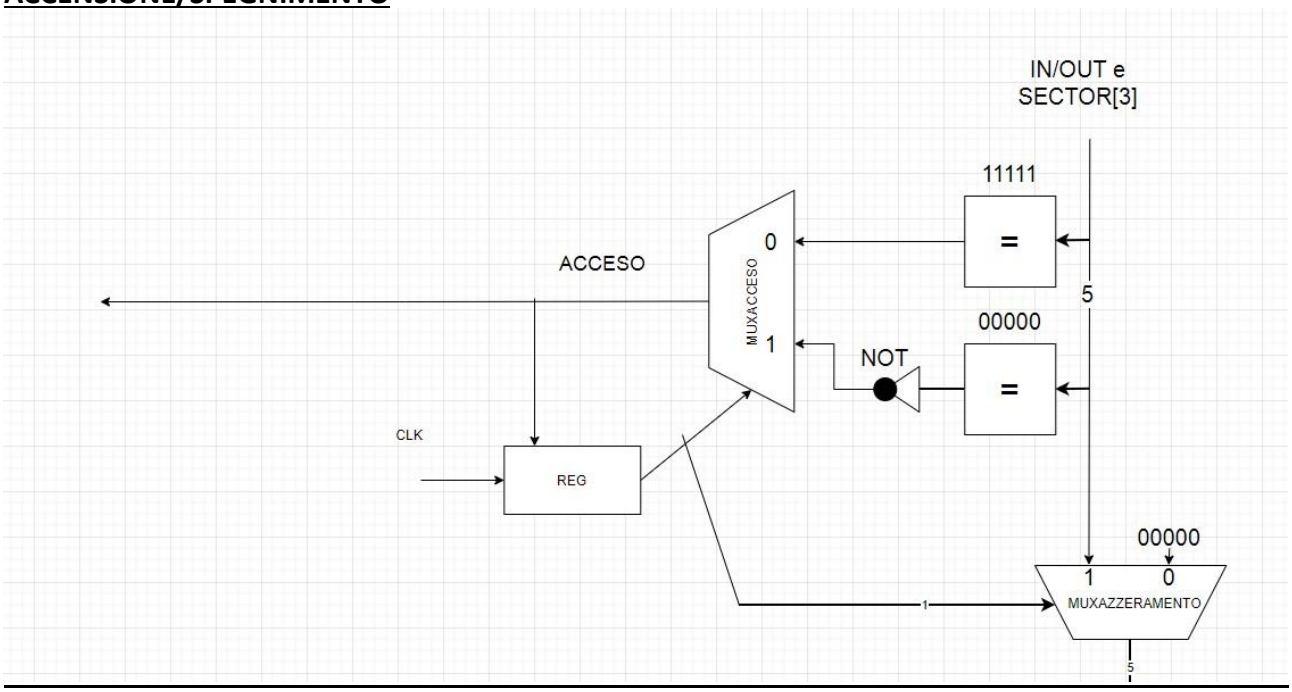
Questo componente ha lo scopo di eseguire perlopiù calcoli, ma non solo esso permette di effettuare scelte attraverso sottocompoenti logici e condizionali, che ci hanno permesso di effettuare la manipolazione dei bit per ottenere i risultati di interesse.

Esso ha come input IN,OUT,SECTOR[3],CARICAMENTO,FSMABC[3],SBARRA_IN e SBARRA_OUT provvisori,mentre come output SECTOR_A,SECTOR_C,SECTOR_C,SBARRA_IN e SBARRA_OUT definitivi.

Schema:



ACCENSIONE/SPEGNIMENTO



In questa sezione del datapath i cinque bit forniti in input vengono utilizzati per effettuare l'accensione e lo spegnimento di tutto il dispositivo.

In caso di inserimento di codifica 11111, il dispositivo si accende se prima era spento e finchè non verrà immessa successivamente una sequenza 00000 esso rimarrà acceso.

Se il dispositivo è spento non può essere acceso dall'inserimento di qualsiasi sequenza causale, deve essere per forza 11111.

Questa parte di datapath si avvale di due comparatori di uguaglianza che confrontano i cinque bit di input con le costanti 11111 e 00000 rispettivamente.

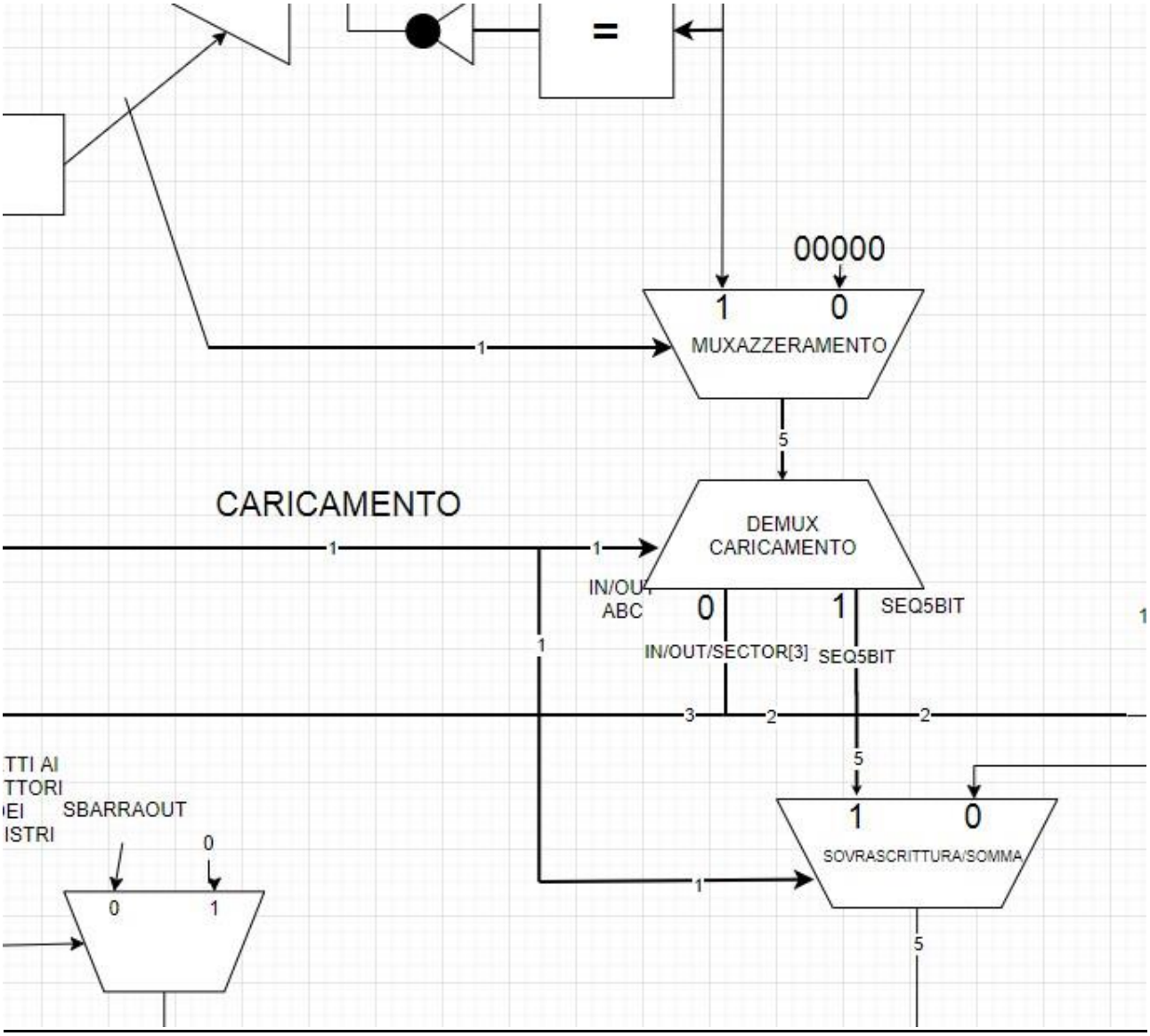
Nel momento in cui viene prodotto il bit di output del comparatore 00000 esso viene negato da un not, un componente che restituisce il bit invertito di quello fornito.

Entrambi i bit ottenuti dai due output vengono immessi come input di MUXACCESO,un multiplexer che si occupa di creare il segnale di ACCESO da fornire al controllore.

Come bit di selezione del mux, è stato scelto di salvare la situazione di accensione precedente tramite l'utilizzo di un registro REG, non collegato concettualmente e logicamente ai registri relativi ai settori,quindi il segnale utilizzato è proprio l'uscita del REG.

Infine, il valore di selezione viene utilizzato anche da un ulteriore componente, ovvero MUXAZZERAMENTO, che si occupa di azzerare i valori in entrata del datapath in caso di spegnimento.

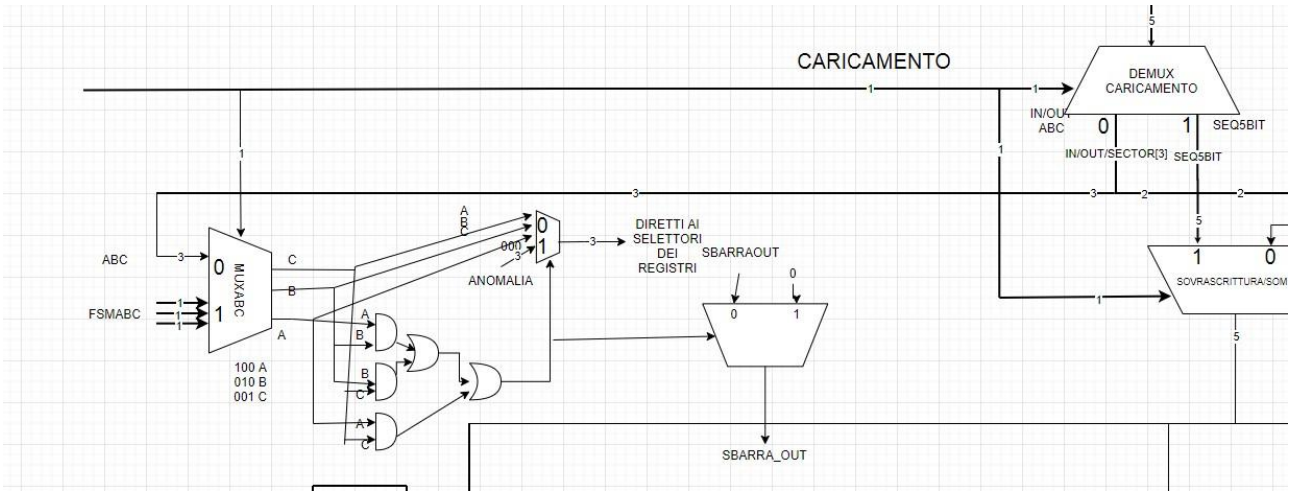
SELEZIONE VALORI PER LA SOVRASCRITTURA DEI REGISTRI



In questa sezione è stato utilizzato il bit di CARICAMENTO proveniente dal controllore per decidere se identificare i cinque bit forniti dalla sequenza di input come valori da sovrascrivere ai registri, oppure per descrivere IN e OUT separati da SECTOR[3], in modo che poi questi valori potessero essere utilizzati in altri punti diversi del dispositivo.

Il componente DEMUX CARICAMENTO, genera due sequenze da cinque bit ciascuna e visto che solo una assume il valore in ingresso, è stato pensato di far risultare IN,OUT e SECTOR[3] in situazione di anomalia in caso essi fossero rimasti tutti a zero, viceversa la sequenza di sovrascrittura di cinque bit verrebbe scambiata con i bit relativi alle operazioni(Vedere sezione operazioni).
A tal proposito interviene il MUX SOVRASCRITTURA/SOMMA ,componente che fornisce bit di sovrascrittura dei registri solo in caso ci sia la fase di caricamento, altrimenti fornisce un esito di un'operazione quale incremento di un registro,decremento oppure un' operazione nulla.

TRATTAMENTO BIT UNIVOCI DEI SETTORI E SBARRA_OUT



In questa sezione seguiamo la sequenza di sinistra di bit emessi del DEMUX CARICAMENTO della sezione precedente, ovvero quella che definisce IN,OUT e SECTOR[3],dove in particolare consideriamo SECTOR[3].

In questa parte avviene la scelta di quale, fra SECTOR[3] e FSMABC[3] viene utilizzata come codifica univoca per i registri. Vengono utilizzate entrambe ma FSMABC[3] viene utilizzata per la fase di caricamento per evitare anomalie,mentre SECTOR[3] per la fase di lavoro.

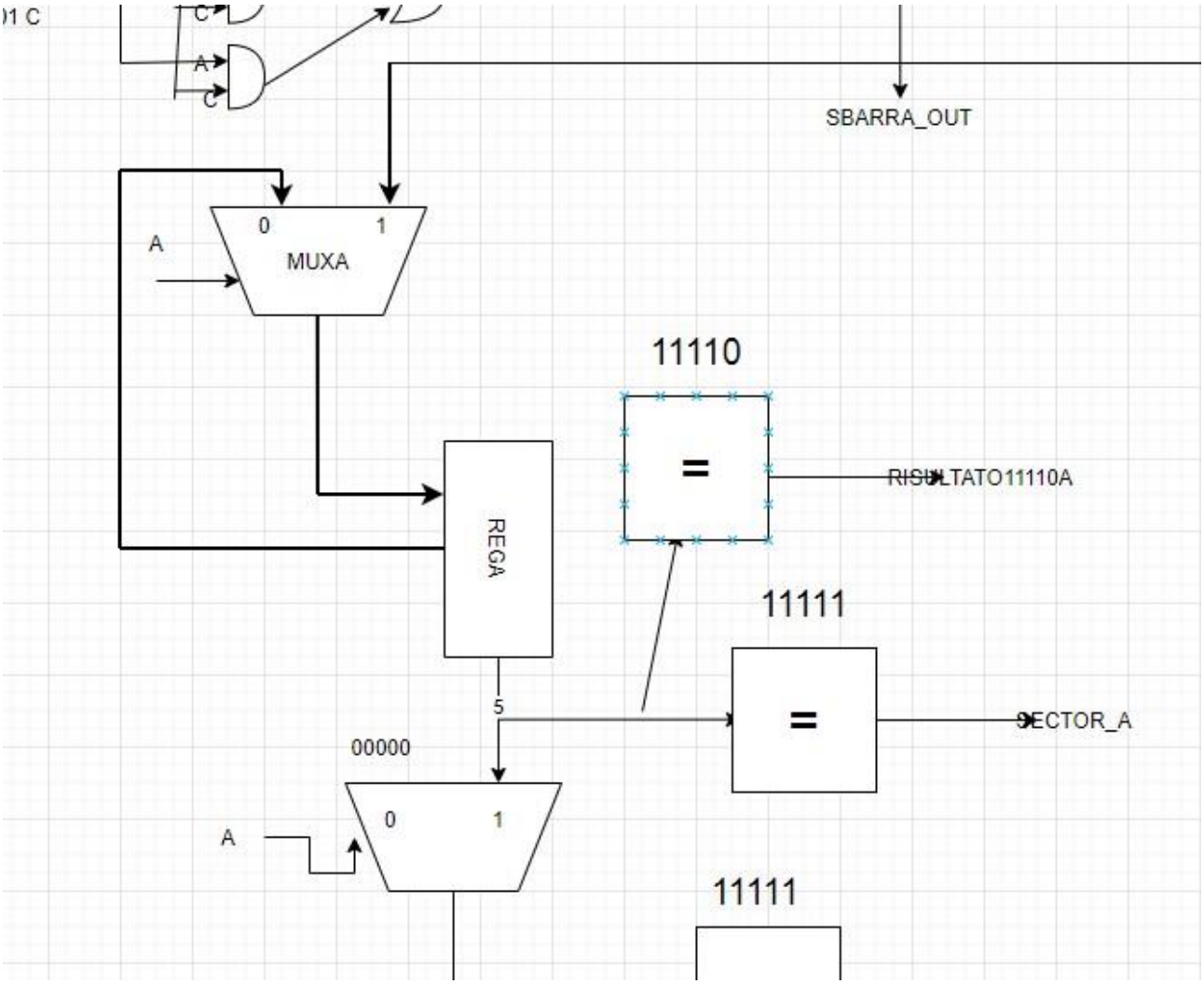
In entrambi i casi viene anche fatta una valutazione sulla codifica per stabilire se al suo interno ci sono più valori posti a 1 (Sequenze **110,011,101,111**) oppure se è presente la singola sequenza a tutti zeri (**000**).

Se vengono riscontrare sequenze come quelle appena citate si è in caso di anomalia e i bit vengono tutti azzerati, questa soluzione è stata ideata allo scopo di non mutare il contenuto dei registri in quel caso.

In questa sezione vi è il trattamento anche del bit SBARRA_OUT, che creava un errore tenendone sono il suo corrispettivo provvisorio,quindi quel bit è stato manipolato in modo che in caso di anomalia non diventasse mai 1.

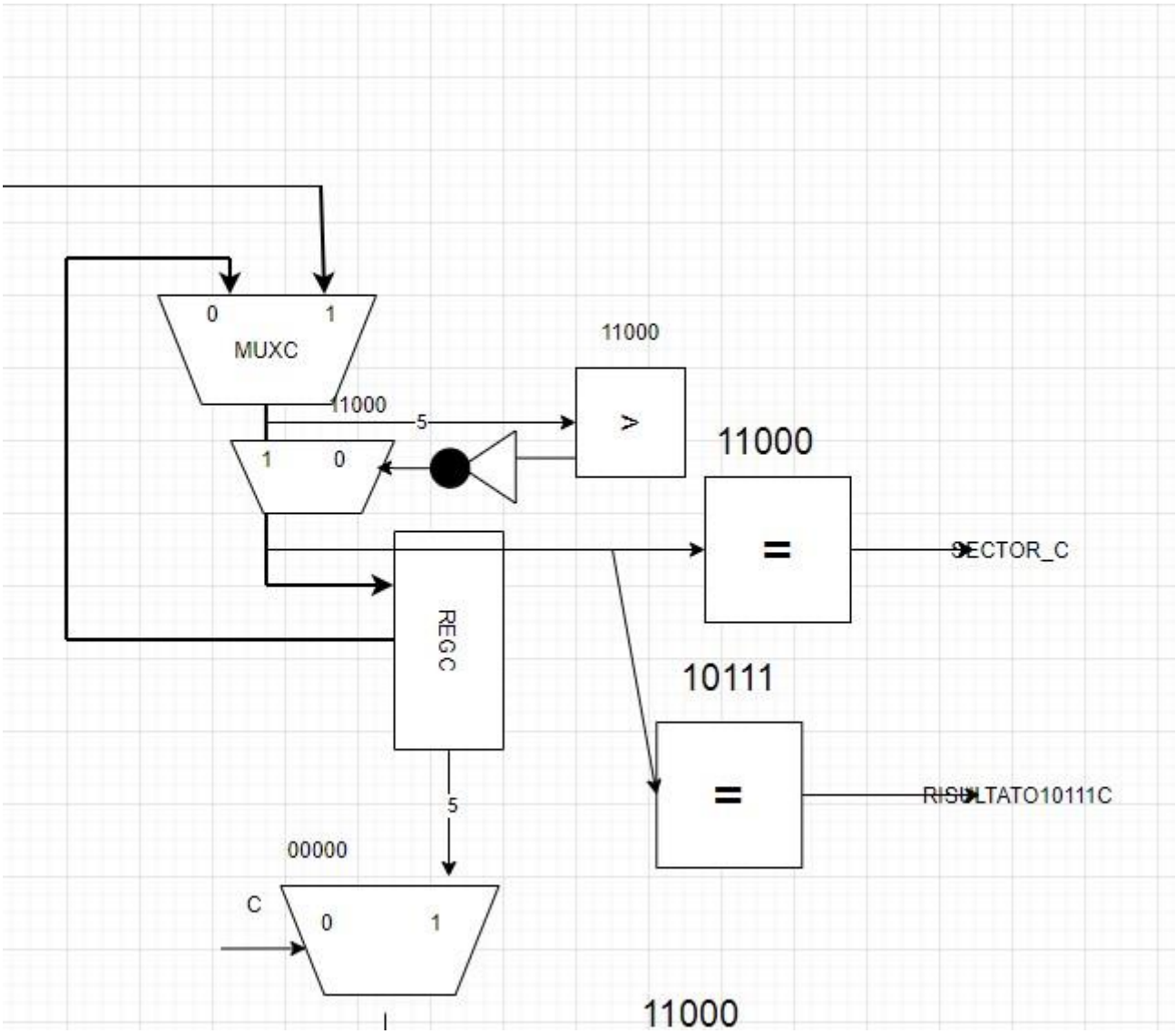
Da questo momento in poi vengono considerati come segnali di codifica univoca dei registri il segnale A,B e C,divisi dalla terzina manipolata.

INSERIMENTO ED ESTRAZIONE DEI VALORI NEI REGISTRI



In questa sezione viene presentato come i registri sono stati sovrascritti.

Il componente MUXA,attraverso il segnale di selezione A, univoco corrispondente al registro A, permette di sovrascrivere o meno il registro con un valore nuovo,altrimenti tiene il precedente. Questa soluzione è stata adottata anche per il registro B (MUXB) e il registro C (MUXC),con qualche modifica e accorgimento in più su quest'ultimo.



Il datapath cambia per il settore C poichè esso contiene un valore massimo di posti auto di 24, perciò è stato applicato un ulteriore multiplexer, con lo scopo di non andare mai oltre al valore binario 11000(24) per il registro C, problematica non da affrontare per B ed A poichè essi raggiungono il valore massimo di 11111(31).

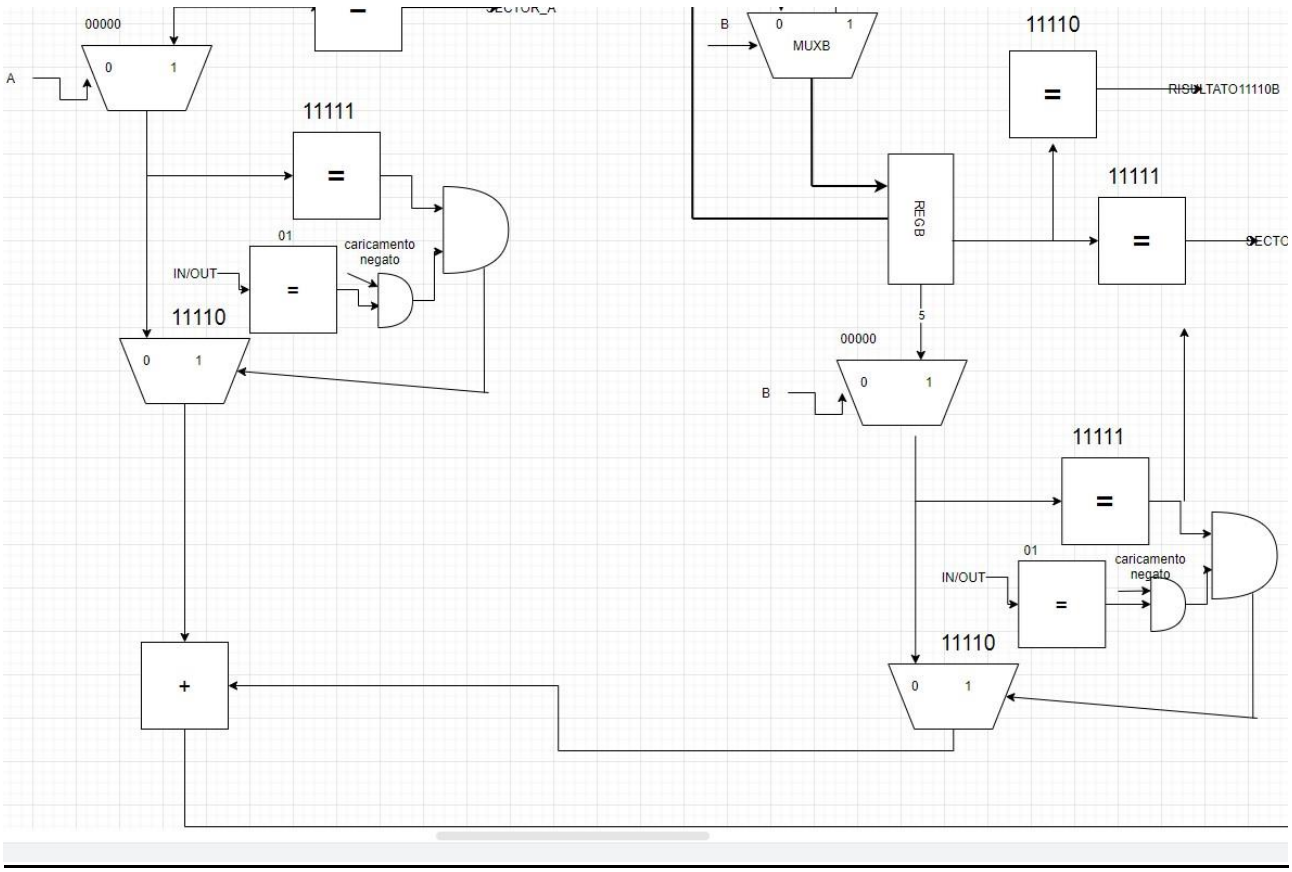
A tale scopo come bit di selezione è stato utilizzato un maggiore di 11000 negato.

Per ogni registro, ad ogni estrazione dei valori viene decretato se esso è pieno oppure no e questo fattore è riportato negli output SECTOR_A,SECTOR_B e SECTOR_C.

Per tutti ancora viene rilevato un bit chiamato RISULTATO11110A/B oppure RISULTATO10111C, questo per risolvere un problema associato a SBARRA_IN che si chiudeva un ciclo di clock in ritardo quando un settore diventava pieno.

Infine tutti e tre i valori prelevati venivano azzerati in caso la richiesta di prelievo del valore non fosse associata a quel determinato registro.

PREPARAZIONE ALLE OPERAZIONI



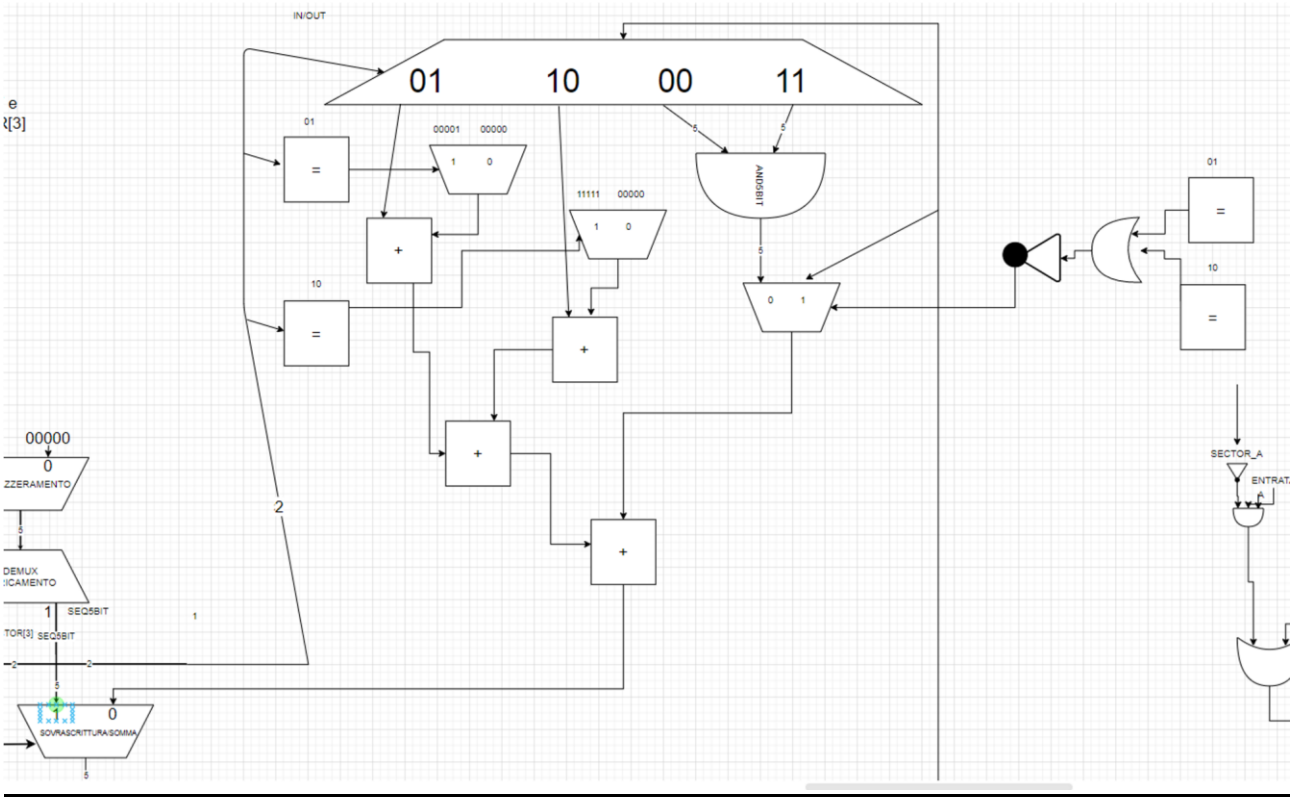
In questa parte il datapath,dopo aver azzerato o meno il valore prelevato dal registro rileva se siamo in situazione di registro pieno e un utente vuole entrare.

In questo caso IN e OUT vengono utilizzati per valutare se siamo in entrata e viene fatta un ulteriore verifica di essere in fase lavoro e non di caricamento.

Se e solo se la situazione che si verifica è quella appena descritta allora il valore prelevato viene sostituito con lo stesso valore decrementato di uno, (11111->11110 per A e B,11000->10111 per C),questo per riottenere il valore iniziale nel registro e quindi non apportare modifiche al valore in esso contenuto.

Infine vengono sommati i tre valori prelevati dai registri, poichè solo uno di loro sarà effettivamente il valore di interessa per applicare un'operazione,poichè non si possono scegliere due settori contemporaneamente,gli altri due valori saranno cinque zeri ciascuno.

OPERAZIONI: INCREMENTO,DECREMENTO,OPERAZIONI NULLE



In questa parte avvengono incremento,decremento oppure operazioni che non toccano il dato di partenza.

IN/OUT sono i due bit di selezioni del DEMUXIO, un demux che genere quattro sequenze di cinque bit, legate alle singole operazioni.

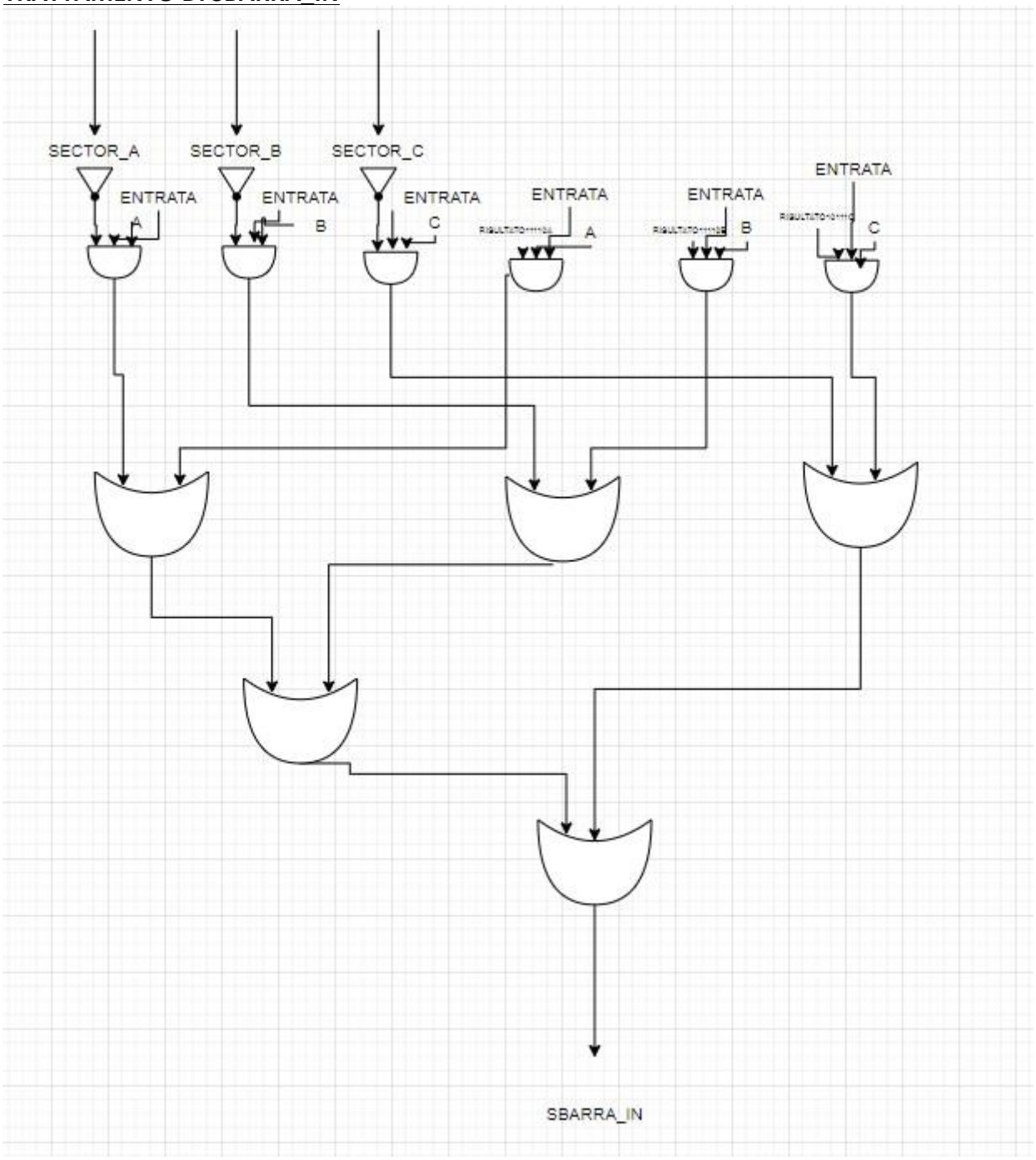
Per incremento e decremento, un rispettivo multiplexer permette ad essi di sommare 00001 in caso di incremento e 11111 in caso di decremento o meno poichè in alternativa entrambi non toccano l'altro addendo in ingresso.

Nel caso delle operazioni nulle invece, gli ultimi dieci bit vengono messi in AND (operazione logica che produrrà in questo caso sempre cinque zeri),poichè nel caso di annullamento attraverso il multiplexer in figura si va a ripescare il valore prelevato dal registro di interesse precedentemente.

Una volta che ogni singola operazione è portata a termine, ogni risultato viene sommato con gli altri affinchè si ottenga un'unica rimanente sequenza di cinque bit, che, tramite il MUX SOVRASCRITTURA/SOMMA verrà scelta come sequenza di sovrascrittura dei registri e in particolare solo del registro di interesse.

Questo unico risultato coincide con il risultato dell'unica operazione che si intendeva svolgere poichè le altre tre, non scelte, conducono a 00000.

TRATTAMENTO DI SBARRA_IN

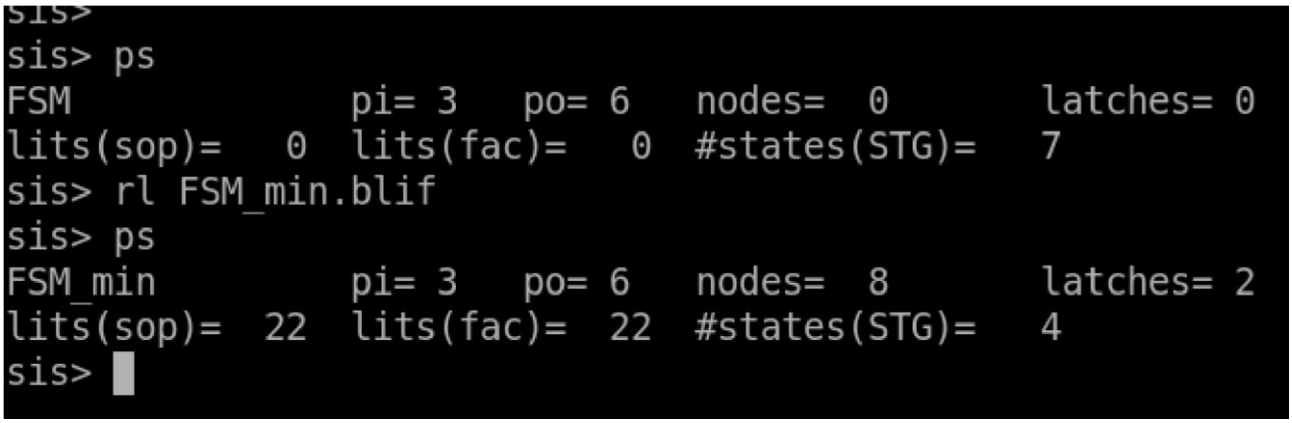


SBARRA_IN definitivo viene trattato per mezzo di molteplici AND poichè la sbarra di entrata deve alzarsi solamente nel caso in cui ci sia spazio nel settore scelto e si sia in caso di una richiesta di entrata.

Sono riutilizzati i segnali RISULTATO11110A,RISULTATO11110B e RISULTATO10111C poichè la sbarra si abbassava quando ancora c'era un posto,con questo stratagemma si ha una soluzione completa.

Statistiche del circuito

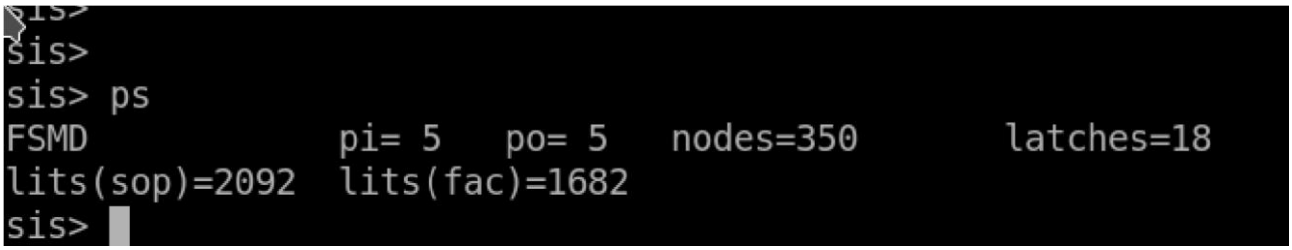
Viene presentato qui di seguito il confronto tra FSM.blif e FSM_min.blif, sorgenti relativi al nostro controllore. Essi presentato le seguenti statistiche



Si può notare come il numero di stati si sia ridotto da 7 a 4.

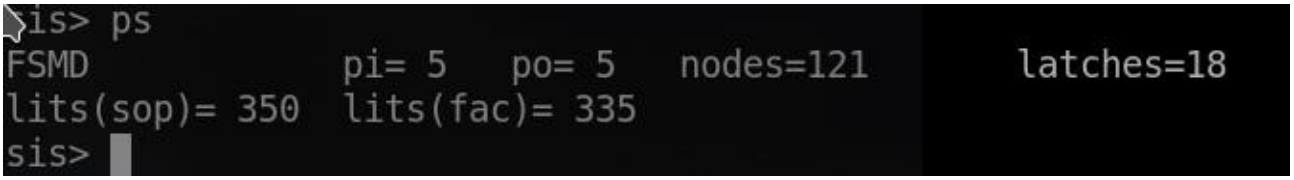
Il controllore è stato minimizzato tramite il comando state_minimize stamina, mentre l'assegnamento della codifica degli stati è avvenuta tramite il comando state_assign jedi.

Vengono presentate di seguito le statistiche dell'FSMD prima dell'ottimizzazione



Si nota un gran numero di letterali e di nodi.

Attraverso la minimizzazione siamo riusciti ad ottenere dei valori molti minori rispetto a questi Il file è stato minimizzato attraverso il comando source script.rugged.



Si può notare come i nodi si siano quasi ridotti ad un terzo dei precedenti e di come i letterali siano diminuiti di varie volte i precedenti.

Di seguito riportiamo infine la mappatura del circuito, dove si possono osservare area e ritardo di quest'ultimo. La mappatura è avvenuta tramite la libreria synch.genlib.

```
sis>
sis> read_library synch.genlib
sis> map -s
warning: unknown latch type at node '{[326]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[327]}' (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:          23
total gate area:       6656.00
maximum arrival time: (42.00,42.00)
maximum po slack:     (-4.60,-4.60)
minimum po slack:     (-42.00,-42.00)
total neg slack:      (-689.00,-689.00)
# of failing outputs:  23
>>> before removing parallel inverters <<<
# of outputs:          23
total gate area:       6592.00
maximum arrival time: (42.20,42.20)
maximum po slack:     (-4.60,-4.60)
minimum po slack:     (-42.20,-42.20)
total neg slack:      (-682.60,-682.60)
# of failing outputs:  23
# of outputs:          23
total gate area:       6240.00
maximum arrival time: (40.00,40.00)
maximum po slack:     (-4.60,-4.60)
minimum po slack:     (-40.00,-40.00)
total neg slack:      (-643.40,-643.40)
# of failing outputs:  23
sis>
```

Il valore dell'area è di 6656.00 e il valore di ritardo è di 42.00 alla termine del procedimento.

Scelte progettuali

1) Abbiamo tenuto i valori di tutti i riporti dei sommatori e SBARRA_IN provvisorio poichè non influivano sul nostro dispositivo alla fine dell'implementazione. Probabilmente se avessimo adottato la scelta di trattarli diversamente avremmo decretato cambiamenti per quanto riguarda area e ritardo finali,da questo punto di vista siamo andati un po' a risparmio. Per questo motivo compaiono le righe di warning alla compilazione. Tali warning non incidono sul dispositivo
2) Abbiamo deciso di inserire le righe di codice del datapath direttamente nell'FSMD anche perchè abbiamo deciso di distinguere a livello di codice la parte di accensione in un file accensione.blif, in modo che ricavasse i segnali necessari all'avvio del controllore mentre le restanti parti sono state implementate nel codice nell'area sottostante la riga riguardante il modello FSM_min.blif (FSM minimizzata).