Relazione SIS

A.A 2022/2023

Indice

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.

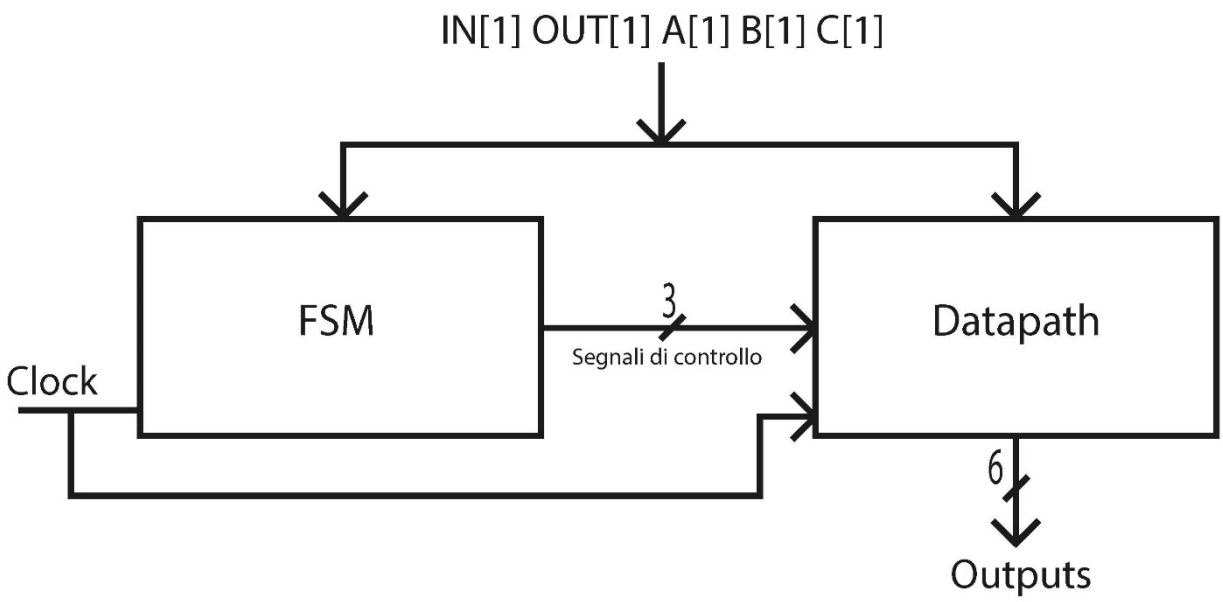
Inputs:

- 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.

Outputs:

- 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.

FSMD



Data una sequenza iniziale di 5 bit tramite una FSMD otteniamo i seguenti output:

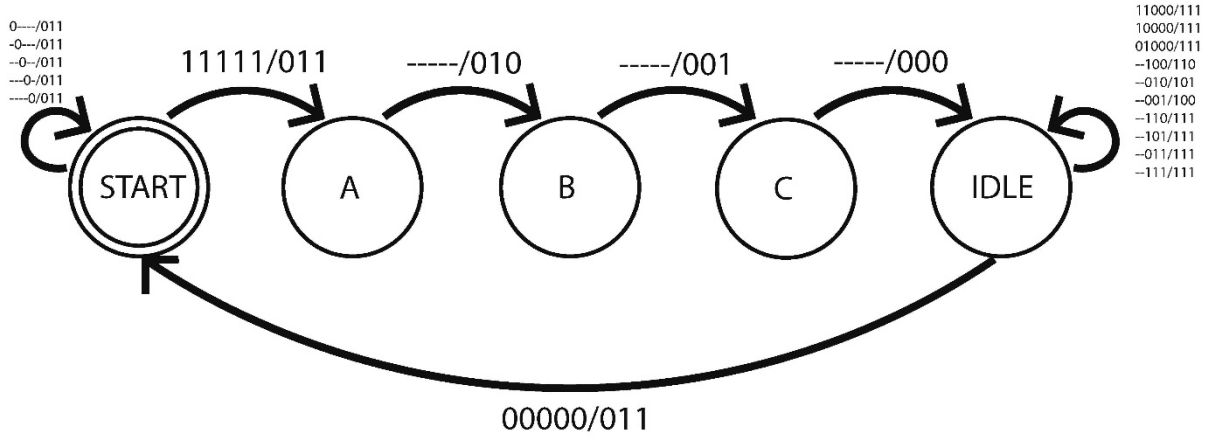
- Settore_non_valido [1]
- Sbarra_IN [1]
- Sbarra_OUT [1]
- Settore_A [1]
- Settore_B [1]
- Settore_C [1]

tramite FSM gestiamo i primi 3 cicli di clock e gestiamo le selezioni.

In seguito controllando le validità delle sequenze in input tramite serie di porte logiche verifichiamo le operazioni richieste ed aggiorniamo i valori.

Le operazioni vengono sempre effettuate. Tramite opportuni controlli di validità del settore e della sequenza i valori vengono sostituiti con i nuovi generati (se validi) o riciclando i valori precedenti (se non validi).

FSM



FSM:

Esclusivamente tramite l'inserimento della sequenza 11111 si passa allo stato A, output 011 (1° bit output = 0: ingresso sequenza, (2° 3°) bit output = i dati non vengono salvati)

Indipendentemente dalla sequenza inserita si passa allo stato B, output 010 (1°bit output = 0: ingresso sequenza, (2° 3°) bit output = i dati vengono salvati nel REG A).

Indipendentemente dalla sequenza inserita si passa allo stato C, output 001 (1°bit output = 0: ingresso sequenza, (2° 3°) bit output = i dati vengono salvati nel REG B).

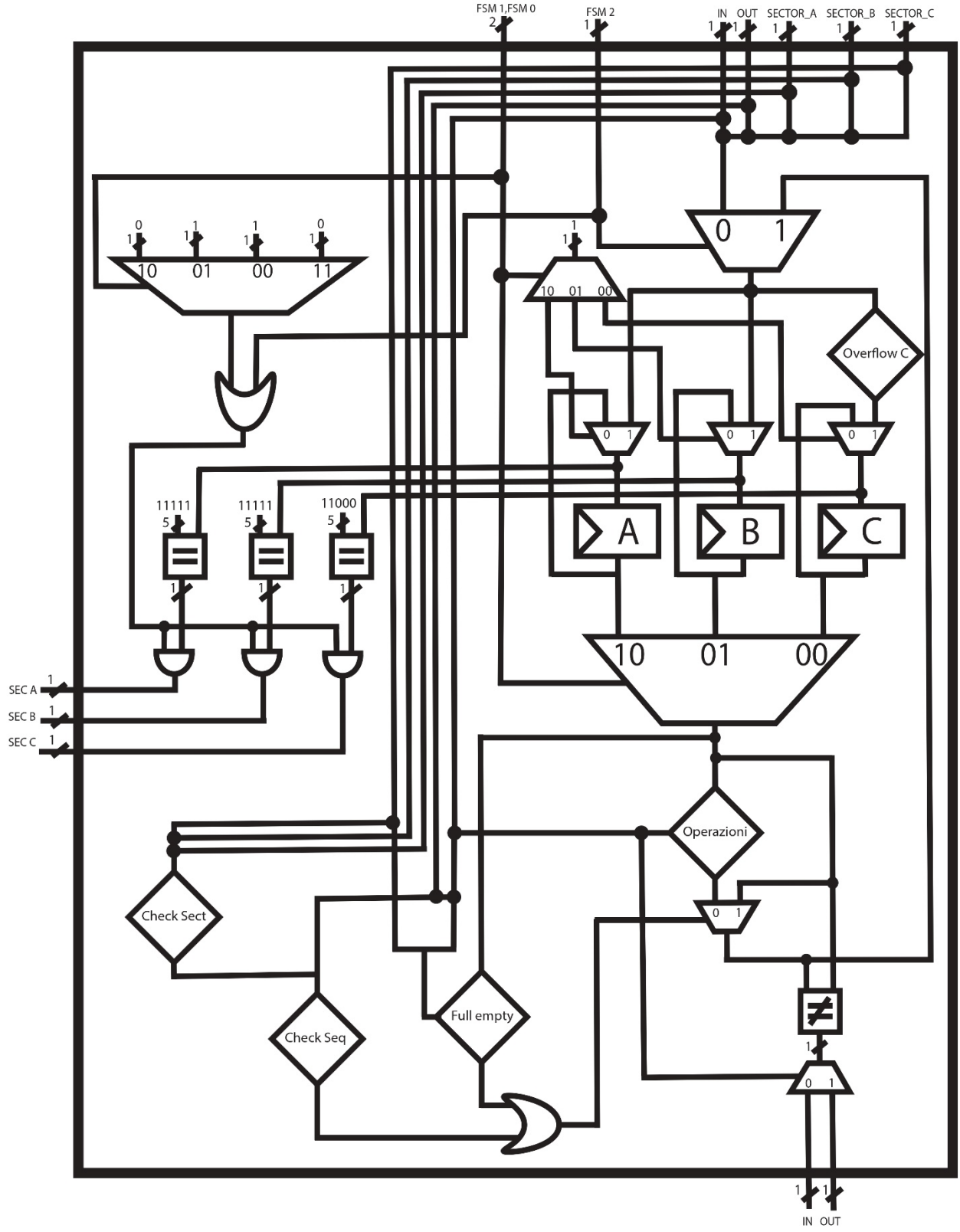
Indipendentemente dalla sequenza inserita si passa allo stato IDLE, output 000 (1°bit output = 0: ingresso sequenza, (2° 3°) bit output = i dati vengono salvati nel REG C).

Dallo stato IDLE con ogni sequenza inserita si rimane in IDLE (reiterazione) (1° bit output=1: mantenimento dei valori esistenti).

La sequenza di output 111 gestisce gli errori (CheckSeq, un componente del datapath, gestirà gli eventuali errori determinati dai primi 2 bit della sequenza).

Tramite sequenza 00000 torniamo allo stato START, output 011 (1° bit output = 0: ingresso sequenza, (2° 3°) bit output = i dati non vengono salvati).

DATAPATH


FSM 1,FSM 0
FSM 2
IN
OUT
SECTOR_A
SECTOR_B
SECTOR_C
10 01 00 11
10 01 00
Overflow C
11111
11111
11000
A
B
C
10 01 00
SEC A
SEC B
SEC C
Operazioni
Check Sect
Check Seq
Full empty
IN OUT

Come input abbiamo la sequenza di 5 bit:

Il primo MUX permette l'ingresso o la reiterazione dei valori:

durante i primi 3 cicli, a seguito dell'attivazione tramite inserimento "11111" la selezione del primo MUX sarà a 0:

abbiamo quindi l'ingresso dei dati dei rispettivi registri grazie alla selezione fornita da un DEMUX ai MUX antecedenti i rispettivi registri A, B, C.

Il DEMUX ha quindi come selezione i bit di FSM che codificano per i rispettivi registri A,B,C.

Il registro non selezionato durante il fronte di salita del clock reitererà i suoi stessi valori che altrimenti andrebbero persi.

NB: alcuni componenti non specificati (rombi) verranno descritti in seguito al paragrafo "Scelte Progettuali".

A questo punto il MUX sottostante ai registri (il quale ha esattamente la stessa selezione qui sopra citata per il DEMUX) permetterà il passaggio dei dati esclusivamente al registro selezionato.

Le operazioni verranno SEMPRE effettuate, sarà il MUX sottostante al blocco "OPERAZIONI" che permetterà il passaggio del valore aggiornato o del valore precedente grazie ad una precisa combinazione di porte logiche spiegata nel dettaglio nella sezione "Scelte progettuali".

A questo punto la semplice differenza (se esiste) tra il nuovo valore ed il precedente al blocco "operazioni" determinerà l'innalzamento o meno della sbarra, la quale viene indirizzata tramite un DEMUX a "sbarra in" o a "sbarra out" in base alla selezione fornita dal 1° bit della sequenza di ingresso.

NB: il DEMUX nel canale non selezionato fornirà output 0.

A seguire quindi il valore viene reiterato all'interno del rispettivo registro, poiché nello stato "IDLE" il segnale "FSM2" sarà costante a 1 e la selezione fornitaci dal DEMUX iniziale permette l'ingresso del valore.

In conclusione abbiamo nel lato sinistro del DATA PATH la gestione dello stato dei registri:

Abbiamo qui un DEMUX con selezione da FSM1, FSM0 (2° e 3° bit dell'output) i quali durante il passaggio di stato: B → C e C → IDLE fornirà output 1.

Il segnale entra poi in una porta "OR", assieme al segnale "FSM2" che sarà ad 1 esclusivamente durante il ciclo IDLE.

Questo segnale entra in una porta "AND" assieme rispettivamente ai distinti valori di controllo tramite uguaglianza del valore aggiornato della reiterazione dei registri, il quale se pari al massimo della capienza fornirà in uscita il bit 1.

Se/quando questo accade il bit in uscita dalla porta "AND" sarà quindi ad 1 ovvero stato settore: pieno.

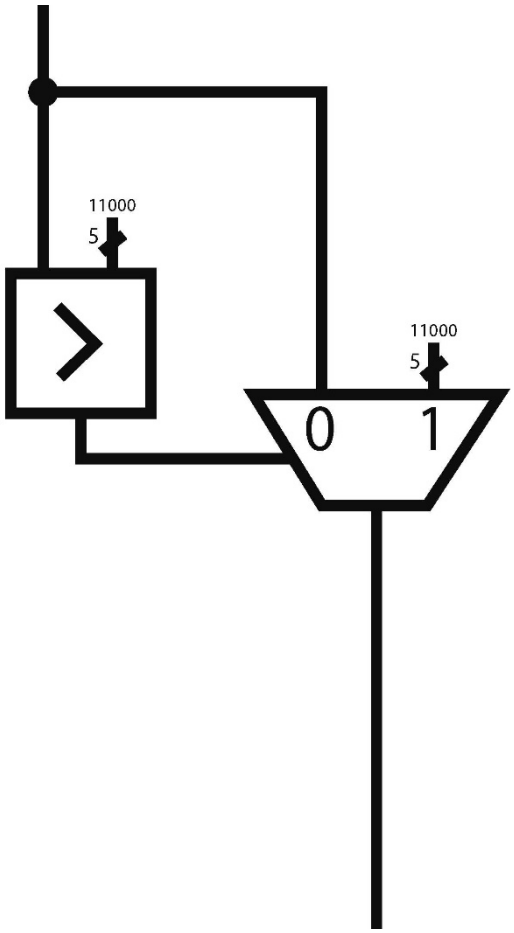
Altrimenti sarà a 0 ovverosia spazio disponibile.

Scelte progettuali

La scelta principale effettuata è la particolare combinazione di porte logiche che assieme determinano la validità o meno della richiesta fornita in input:

Iniziamo però dai passaggi più semplici

Overflow C



Ingressi:

valore registro C.

Componenti:

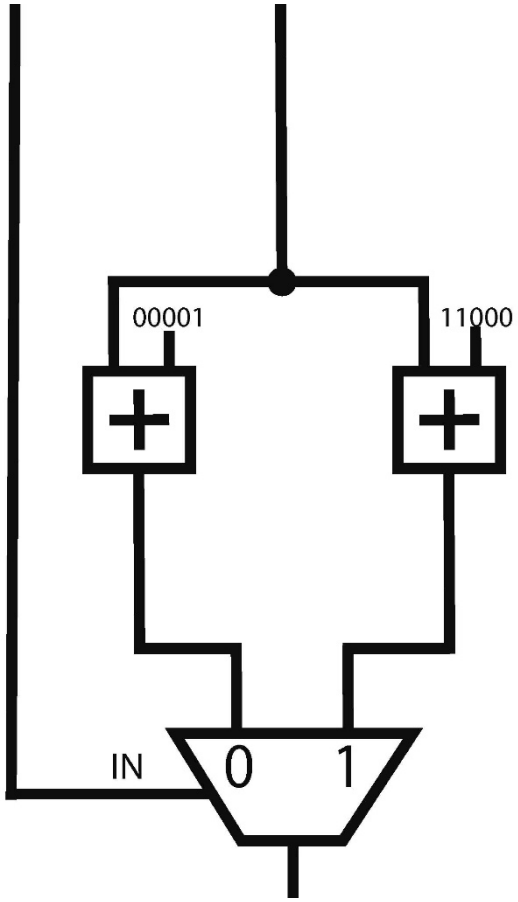
1 blocco di maggioranza

1 mux

Il valore in ingresso nel registro C nella selezione "1" del MUX antecedente il registro viene duplicato e portato in ingresso al blocco di maggioranza, il quale lo compara con il valore "11000".

Se il valore in ingresso è > 24 allora verrà fornito 1 come selezione al MUX sottostante che fornirà in output il valore "11000", altrimenti fornirà 0 e quindi avremo il semplice passaggio del valore di ingresso.

Operazioni



Ingresso:

valore del registro selezionato.

Componenti:

2 sommatori a 5 bit

1 multiplexer

Il valore in uscita dal registro selezionato entra in entrambi i sommatori, dei quali il primo somma "00001" mentre il secondo somma "11111" il cui risultato equivale al sottrarre 1.

Entrambi i valori vengono forniti al MUX, il valore sommato di 1 nell'ingresso 0, mentre il valore sottratto di 1 all'ingresso 1.

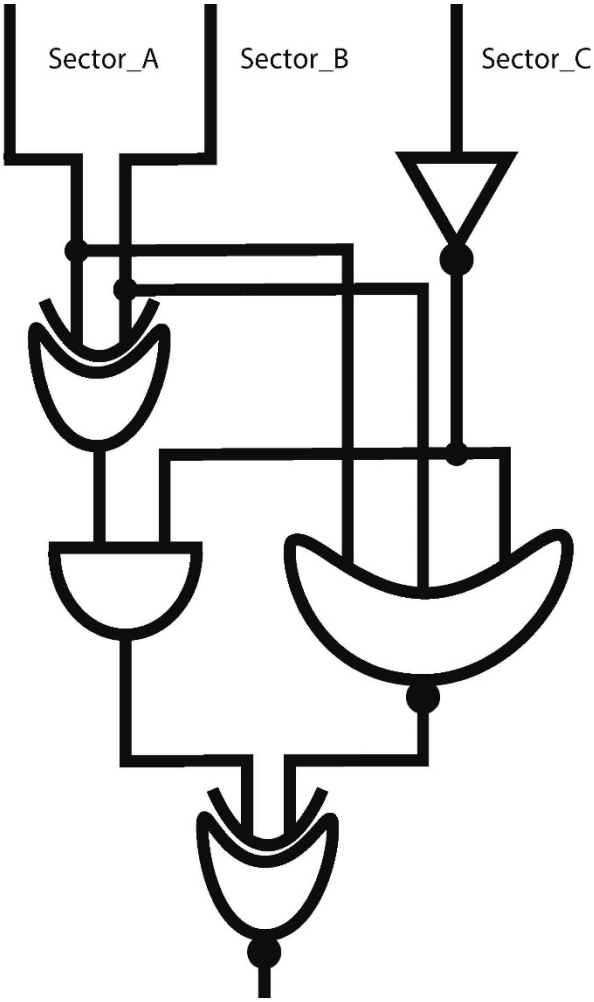
Ora il MUX semplicemente fornirà solo il valore richiesto grazie alla selezione fornita dal bit "IN".

Abbiamo ora la combinazione di porte logiche che valida la richiesta, suddivisa in 3 principali sezioni:

- CheckSect
- CheckSeq
- Full empty

Le descrivo quindi nel rispettivo ordine:

CheckSect: (validità settore)



Ingressi:

"SECTOR_A","SECTOR_B","SECTOR_C"

Componenti:

1 negatore

1 porta XOR

1 porta NOR

1 porta AND

1 porta XNOR

Il bit "SECTOR_A" entra assieme a "SECTOR_B" in una porta XOR.

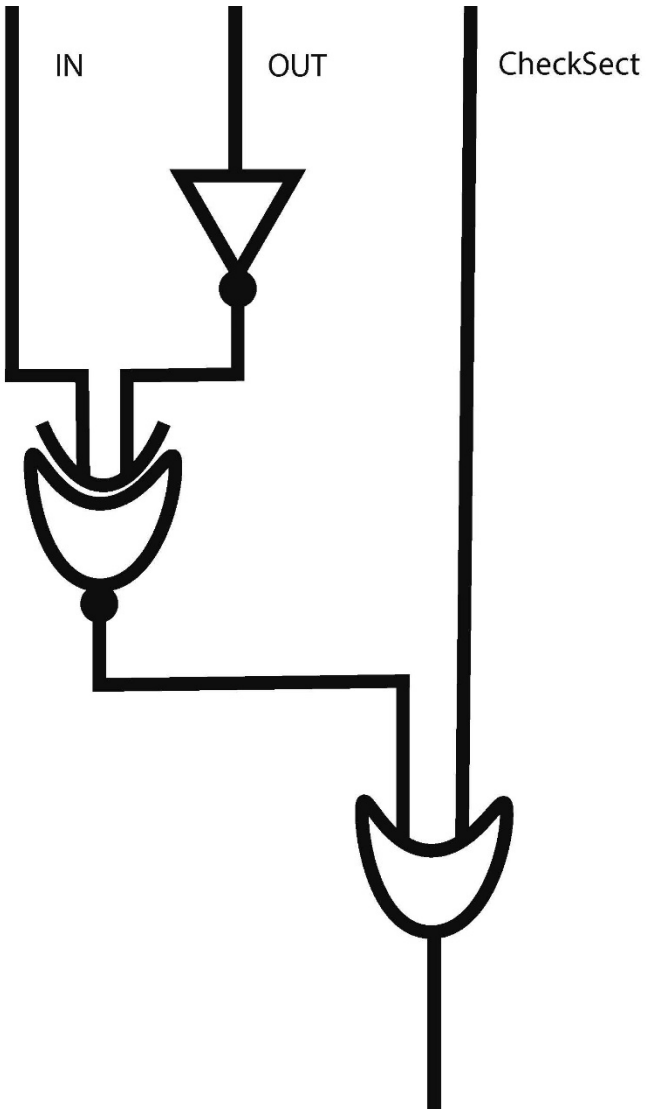
Il suo output entra in una porta AND assieme al valore "SECTOR_C" NEGATO (verifichiamo non ci siano errori di compilazione e quindi che sia selezionato A, B oppure C).

Contemporaneamente i valori:
"SECTOR_A","SECTOR_B",

"SECTOR_C" NEGATO entrano in NOR (3 ingressi) (che fornisce in output 1 solo se input:000)

Il suo output assieme all'output della porta AND entrano in un'ultima XNOR che ci fornisce la validità del settore.

CheckSeq: (validità sequenza)



Ingressi:

"IN", "OUT", "CheckSect"

Componenti:

1 NEGATORE

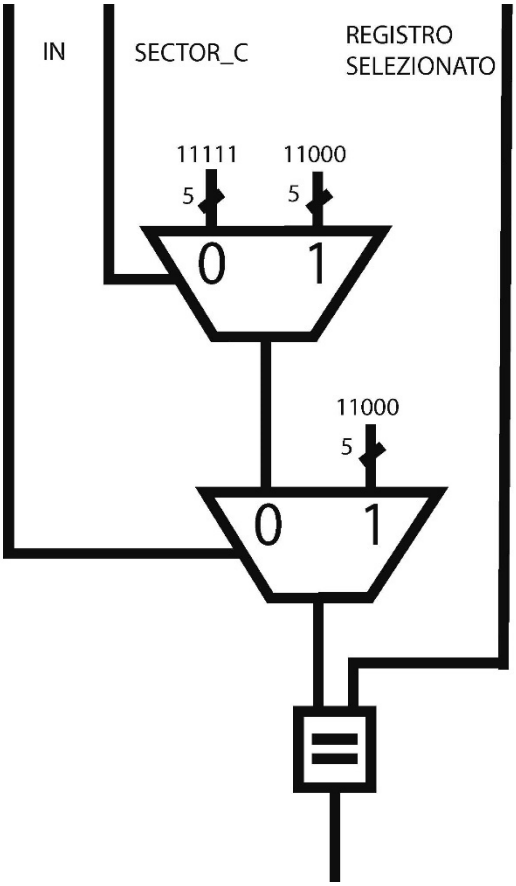
1 porta XNOR

1 porta OR

Il bit "IN" ed il bit "OUT" NEGATO entrano in XNOR (fornirà output 1 solo se la sequenza iniziale inserita è 10 o 01)

In seguito semplicemente questo bit entra in OR con CheckSect, e quindi otteniamo un unico bit che ci autentica la validità del settore e della sequenza complessiva.

Full Empty



Questo componente controlla lo stato dei registri, se fossero pieni NON permette l'ingresso di ulteriori veicoli ma ne permette l'uscita.

Ingressi:

i bit "IN","SECTOR_C" ed il valore del registro selezionato.

Componenti:

2 multiplexer

1 blocco uguaglianza

Il bit "SECTOR_C" funge da selezione di un primo MUX il quale permette l'ingresso tramite selezione 0 di "11111", altrimenti "11000".

Questo valore arriva in ingresso 0 del secondo MUX, mentre in ingresso 1 abbiamo una costante "00000".

Tramite bit "IN" (che funge da selezione) gestiamo la differenza di output:

Se siamo in INGRESSO ("0") oppure se siamo in USCITA ("1").

Il valore in uscita viene poi messo in UGUAGLIANZA con il valore del registro selezionato ed otteniamo quindi:

.se siamo in uscita l'azione sarà sempre consentita (output 0).

.se siamo in ingresso, ed il registro è pieno, (output 1) ingresso negato. Oppure, se il registro non è pieno, (output 0) ingresso consentito.

Ottimizzazione

FSM

Per prima cosa scriviamo la FSM e avviamo il comando per l'assegnamento degli stati:

read_blif FSM.blif

state_assign jedi

write_blif FSM.blif

Ora ottimizziamo la FSM:

```
sis> rl fsm.blif
Warning: network `fsm.blif', node "IN" does not fanout
Warning: network `fsm.blif', node "OUT" does not fanout
Warning: network `fsm.blif', node "SECTOR_A" does not fanout
Warning: network `fsm.blif', node "SECTOR_B" does not fanout
Warning: network `fsm.blif', node "SECTOR_C" does not fanout
sis> print_stats
fsm             pi= 5   po= 3   nodes=  6       latches= 3
lits(sop)=  95  #states(STG)=   5
sis> source script.rugged
sis> print_stats
fsm             pi= 5   po= 3   nodes= 10       latches= 3
lits(sop)=  49  #states(STG)=   5
sis> source script.rugged
sis> print_stats
fsm             pi= 5   po= 3   nodes= 10       latches= 3
lits(sop)=  46  #states(STG)=   5
sis>
```

Datapath

Una volta creato il file del datapath (con tutti i componenti precedentemente descritti) lo possiamo ottimizzare:

```
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: network `operation', node "[94]" does not fanout
Warning: network `operation', node "[114]" does not fanout
Warning: network `datapath', node "O_TRASH_DEMUX_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 "[94]" does not fanout
Warning: network `datapath', node "[114]" does not fanout
sis> print_stats
datapath        pi= 8   po= 6   nodes=172       latches=15
lits(sop)= 627
sis> source script.rugged
sis> print_stats
datapath        pi= 8   po= 6   nodes= 51       latches=15
lits(sop)= 233
sis> source script.rugged
sis> print_stats
datapath        pi= 8   po= 6   nodes= 44       latches=15
lits(sop)= 235
sis>
```

FSMD

Ora una volta collegata la FSM al datapath tramite la FSMD possiamo ottimizzare:

```
sis> source script.rugged
sis> print_stats
FSMD            pi= 5   po= 6   nodes= 63       latches=18
lits(sop)= 286
sis> source script.rugged
sis> print_stats
FSMD            pi= 5   po= 6   nodes= 55       latches=18
lits(sop)= 286
sis> source script.rugged
sis> print_stats
FSMD            pi= 5   po= 6   nodes= 53       latches=18
lits(sop)= 280
sis>
```

Ora salviamo il file ottimizzato.

Queste sono le statistiche del file mappato per area:

```
sis> read_library synch.genlib;map -m 0 -s
warning: unknown latch type at node '{[0]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[1]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[2]}' (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:       6504.00
maximum arrival time: (51.80,51.80)
maximum po slack:     (-6.20,-6.20)
minimum po slack:     (-51.80,-51.80)
total neg slack:      (-921.00,-921.00)
# of failing outputs:  24
>>> before removing parallel inverters <<<
# of outputs:          24
total gate area:       6440.00
maximum arrival time: (49.60,49.60)
maximum po slack:     (-6.20,-6.20)
minimum po slack:     (-49.60,-49.60)
total neg slack:      (-901.20,-901.20)
# of failing outputs:  24
# of outputs:          24
total gate area:       5800.00
maximum arrival time: (47.60,47.60)
maximum po slack:     (-6.20,-6.20)
minimum po slack:     (-47.60,-47.60)
total neg slack:      (-862.80,-862.80)
# of failing outputs:  24
sis>
```