Relazione dell’elaborato di Architettura degli Elaboratori

di Mirco Mattevi VR486372 e Davide Camozzo VR488204

A.A. 2022/2023

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, mentre il settore C ha 24 posti. 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. 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. Il circuito ha 2 ingressi definiti nel seguente ordine:

- 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.
- SECTOR [3 bit]: i settori sono indicati con 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.

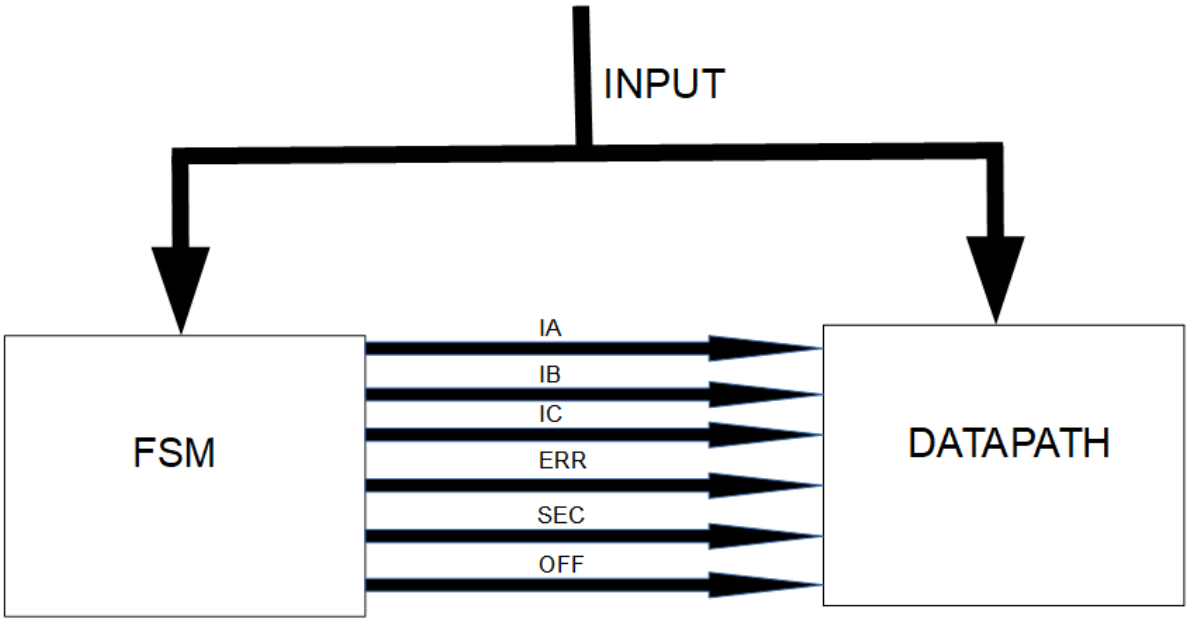
Ad ogni richiesta il dispositivo risponde aprendo una sbarra e aggiornando lo stato dei posti liberi nei vari settori. Se un utente chiede di occupare un settore già completo il sistema non deve aprire alcuna sbarra. Il circuito deve avere 3 output che devono seguire il seguente ordine:

- 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
- 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.

Il dispositivo si spegne quando riceve la sequenza 00000 in input.

Nel caso in cui venga inserito un settore non valido la sbarra deve rimanere chiusa. Anche in questo caso, i valori dei settori devono comunque essere riportati rispetto alla loro situazione attuale.

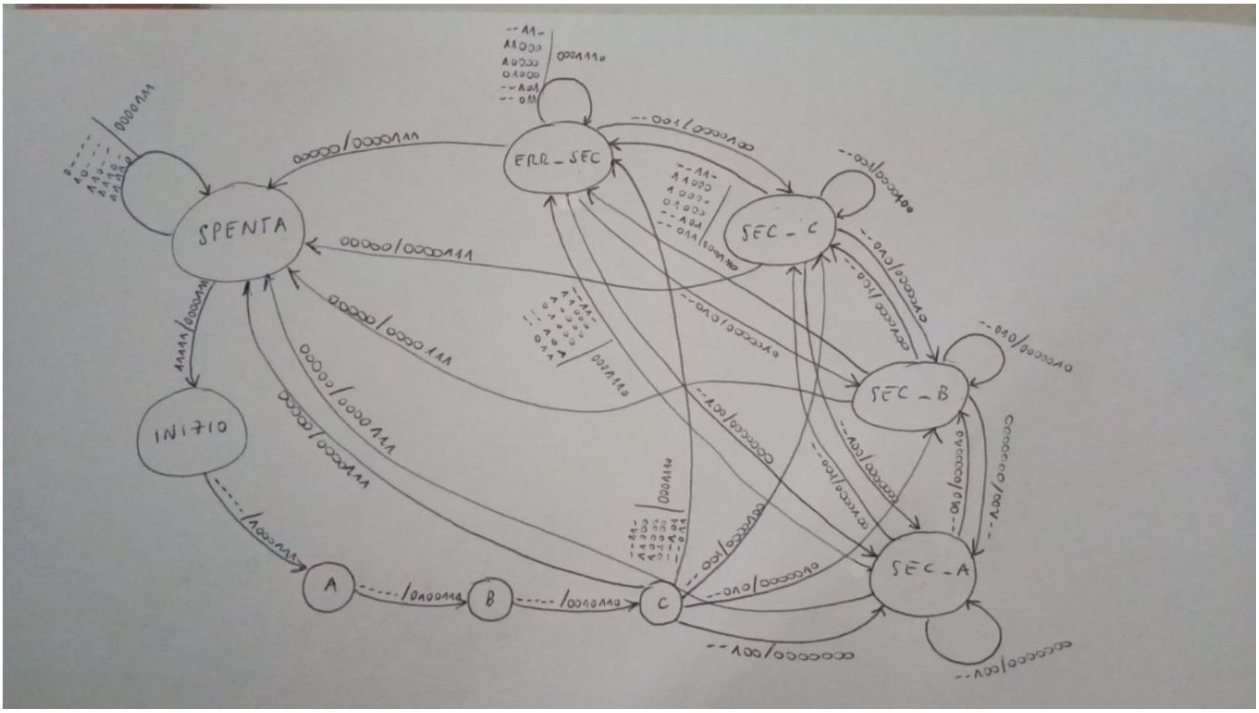
Architettura generale del circuito



Il circuito passa i 5 bit dell'input sia all'FSM che al Datapath. Non ci sono segnali di stato ma solo segnali di controllo, che in totale occupano 7 bit così suddivisi:

- IA, IB e IC [1 bit ciascuno]: i bit vengono alzati uno dopo l'altro quando il circuito si trova nei primi tre cicli, ovvero dove avviene l'inizializzazione dei settori. Ogni segnale corrisponde a un settore
- ERR [1 bit]: il bit viene alzato quando la selezione del settore è diverso da 100, 010 o 001
- SEC [2 bit]: questo segnale ci permette di identificare su quale settore stiamo lavorando: 00 per A, 01 per B, 10 per C e 11 per D (settore "spazzatura" che sarà usato quando l'input non interessa l'aggiornamento dei posti macchina (per esempio in caso di errore))
- OFF [1 bit]: questo bit diventa vero solo a macchina spenta e nei primi tre cicli di clock (quando l'output deve comunque essere identico a quando la macchina è spenta)

Diagramma degli stati del controllore



L' FSM del circuito è composta da 9 stati, suddivisibili in 3 macro-sezioni:

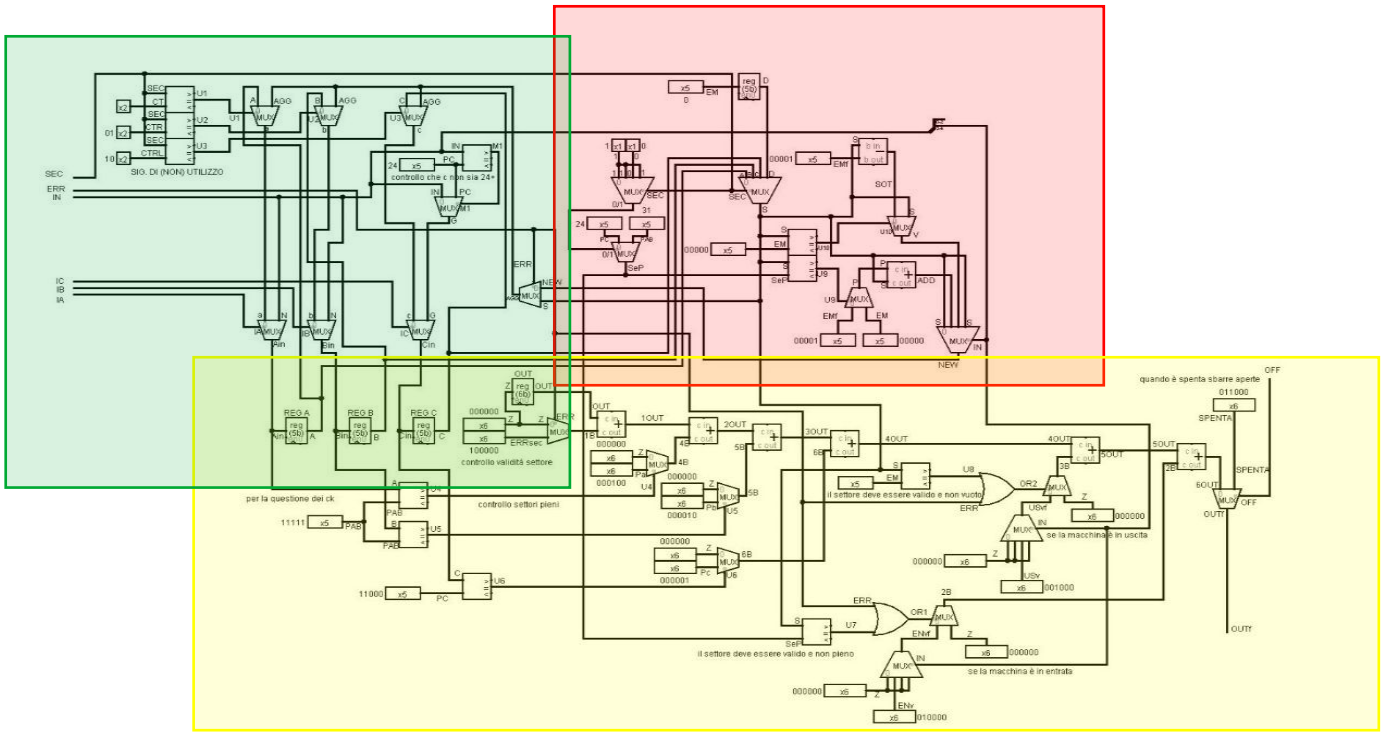
- macchina spenta
- primi tre cicli di clock dopo l'accensione, ovvero quando viene precisato il numero di parcheggi occupati per ogni settore dopo la notte
- controllo dei bit del settore, uno stato per ogni settore e in più uno stato anche per l'errore di codifica del settore (quelli della sbarra vengono, invece, gestiti nel Datapath).

Le tre macro-sezioni sono poi così composte:

- SPENTA: fintanto che l'input non è quello di accensione, ovvero 11111, la FSM resta in questo stato.
- INIZIO: stato prossimo di SPENTA nel caso di input 11111. Nella prima versione della FSM avevamo riscontrato problemi nell'inizializzazione del settore C, che diventava automaticamente sia numero di parcheggi occupati, che input trattato come entrata/uscita di macchine del parcheggio. Abbiamo pertanto introdotto questo stato aggiuntivo per ovviare al problema.

- A, B e C: stati usati per l’inizializzazione delle sezioni del parcheggio. Indipendentemente dall’input lo stato prossimo di A è B, e quello di B è C, da C, invece, l’input verrà trattato normalmente.
- SEC_A, SEC_B, SEC_C e ERR_SEC: ad ogni stato corrisponde una codifica dei bit di settore, 100 per SEC_A, 010 per SEC_B, 001 per SEC_C, qualsiasi altra combinazione per ERR_SEC.

Architettura del Datapath



Il Datapath del circuito è composto da 3 macro-sezioni:

- gestione dell'input (in rosso): è la parte dove viene sommato o sottratto uno al settore interessato, scelto con un multiplexer che usa come selettore il segnale a 2 bit SEC proveniente dalla FSM. Viene fatto un controllo sui posti occupati nel settore, in particolare interessa sapere se è vuoto o se è pieno, in questi due casi viene passato, rispettivamente, come sottraendo o come secondo addendo lo 0 (perché chiaramente non è possibile che una macchina esca da un settore vuoto o entri in uno pieno). Somma e sottrazione vengono fatte a priori da quello che sta realmente facendo la macchina, poi in base ai bit della sbarra (dove vediamo se la macchina entra, esce o se c'è stato un errore nella codifica) facciamo andare avanti solamente il risultato dell'operazione corretto grazie ad un multiplexer
- aggiornamento dei registri (in verde): è la parte dove viene cambiato il numero di posti occupati nei vari settori. È composta da un registro per settore e da due multiplexer in serie per ognuno di essi, il primo ha come input il numero di parcheggi dopo addizione/sottrazione/errore (quello che esce dalla parte di gestione dell'input, per intenderci) e il valore presente nel registro in quel momento, mentre come selettore usiamo il risultato dell'uguaglianza tra il segnale di controllo SEC e la codifica di quest'ultimo corrispondente ad ogni

settore (00 per A, 01 per B, 10 per C), così da salvare il valore modificato solo nel settore dove sta realmente entrando o uscendo la macchina. Il secondo multiplexer viene invece usato per controllare se il rispettivo registro sia o meno in fase di inizializzazione usando come selettore uno dei segnali di controllo IA, IB, IC in base al settore in questione: quando il valore del selettore è 1, l'input viene salvato all'interno del registro (perché ci troviamo nei primi 3 cicli di clock), altrimenti viene salvato il valore uscente dal primo multiplexer. Vi è poi una differenza nel sistema di inizializzazione di C, dato che presenta un sistema di controllo del valore in input, che non deve superare 24, capienza massima del settore, nel caso in cui il valore inserito sia superiore al massimo, nel registro viene salvato 24.

- gestione dell'output (in giallo): parte tutto dal registro OUT, azzerato ad ogni ciclo, che viene modificato bit per bit attraverso delle somme: Il primo bit, usato per segnalare un errore nella codifica del settore, è collegato al segnale ERR della FSM: quando quest'ultimo è 1, anche il bit dell'output sarà 1; il secondo bit (ovvero quello della sbarra di entrata) viene alzato se la macchina è in entrata e in assenza di errori di codifica e di settore non pieno; viene trattato ugualmente il terzo bit (quello della sbarra di uscita); gli ultimi 3 bit vengono trattati allo stesso modo: si effettua un controllo sui singoli settori per verificare se siano pieni o meno, facendo particolare attenzione a usare il segnale in entrata nei registri A,B e C e non quello in uscita per evitare problemi con i clock.

Statistiche del programma

```
sis> rl fsm.blif
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 9
Number of states in minimized machine : 5
```

Grazie alla minimizzazione degli stati nella FSM, siamo riusciti a passare da 9 a 5 stati

```
sis> ps
fsmd            pi= 5   po= 6   nodes=352       latches=29
lits(sop)=2446  lits(fac)=1838
sis> source script.rugged
sis> ps
fsmd            pi= 5   po= 6   nodes= 65       latches=18
lits(sop)= 294  lits(fac)= 281
sis> source script.rugged
sis> ps
fsmd            pi= 5   po= 6   nodes= 60       latches=18
lits(sop)= 289  lits(fac)= 273
sis> fx
sis> ps
fsmd            pi= 5   po= 6   nodes= 64       latches=18
lits(sop)= 284  lits(fac)= 277
```

Per quanto riguarda l'ottimizzazione per area siamo riusciti a passare da 2446 letterali a 284

```
# of outputs:          24
total gate area:       6008.00
maximum arrival time: (51.60,51.60)
maximum po slack:     (-5.80,-5.80)
minimum po slack:     (-51.60,-51.60)
total neg slack:      (-898.00,-898.00)
# of failing outputs:  24
```

Infine, mappando il circuito con la libreria Synch.genlib abbiamo ottenuto un area totale di 6008 e un ritardo massimo di 51.60

Scelte progettuali

Equilibrio FSM-datapath

Inizialmente avevamo progettato il circuito in modo che il datapath gestisse la quasi totalità delle funzioni primarie, quindi oltre al controllo dei bit di sbarra gli avevamo assegnato anche quello dei bit di settore; così facendo, però, la FSM non aveva alcuna funzione degna di nota se non quella di gestire i primi tre cicli di clock e lo spegnimento del dispositivo. Perciò abbiamo deciso, come specificato precedentemente, di far gestire i primi due bit al datapath e gli ultimi tre alla FSM; questo ci ha permesso anche un notevole "alleggerimento" del circuito in quanto abbiamo potuto eliminare svariati controlli (tutti fatti con multiplexer) e di semplificarne alcuni (multiplexer a 4 ingressi invece che 8 dato che in questo modo si fa riferimento al segnale SEC della FSM e non più ai tre bit di settore).

Sottrattore

Nella parte di gestione dell'input abbiamo parlato di sottrazione e somma. Dato che dalle specifiche del progetto si può capire che può uscire una sola macchina per volta non abbiamo implementato un vero e proprio sottrattore ma un componente che ci si avvicina molto ma che ha come sottraendo fisso 1. In questo modo non siamo andati a complicare ulteriormente un circuito già di per se abbastanza pesante.

Output

Dalle specifiche non è ben chiaro quale volesse essere l'output del dispositivo da spento, ma viene detto che le sbarre restano alzate per la notte, perciò abbiamo deciso di mettere come output 011000. Ed è proprio per questo che abbiamo inserito il segnale di controllo OFF, che si collega ad un multiplexer nel datapath in modo da far passare l'output sopra citato nel caso in cui OFF=1.