Laboratorio di Architettura degli Elaboratori-SIS

a cura di Tinazzi Camillo VR459920 e Pasetto Michele VR495361

Descrizione

Il sistema deve gestire l'accesso dei mezzi ad un parcheggio. Il parcheggio, dotato di 2 sbarre, una per l'ingresso dei veicoli ed una per l'uscita del veicolo, è suddiviso in 3 zone, da qui in poi definite A,B,C rispettivamente con una capienza di 31, 31 e 24 posti.

Requisiti

Tratto dal testo dell'elaborato:

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

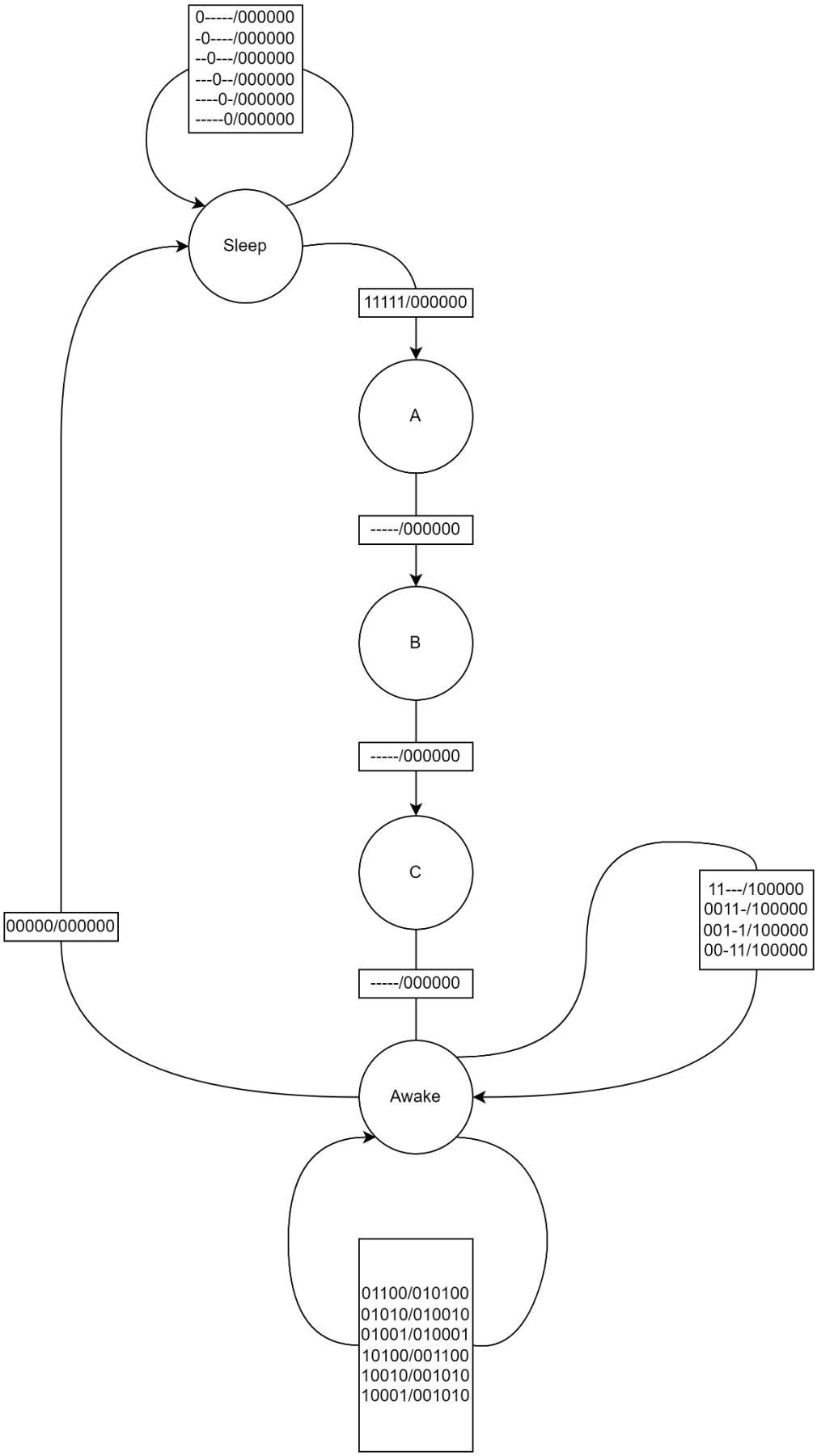
Nel caso in cui venga inserito un settore non valido la sbarra deve rimanere chiusa. Anche in questo caso, i

valori dei settori devono comunque essere riportati rispetto alla loro situazione attuale.

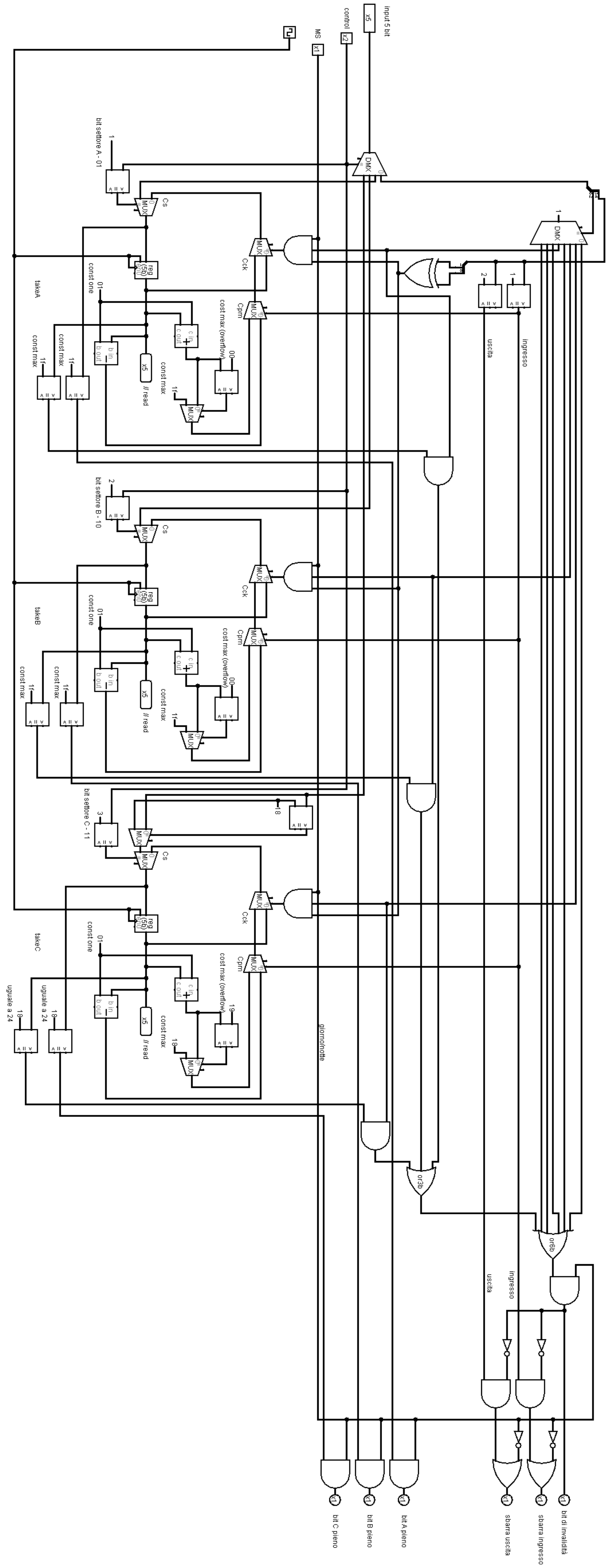
Scelte di implementazione

- 5 stati per l'STG:
  - **SLEEP** : lo stato in cui il sistema è spento. L'output rimane 011000 (sbarre alzate). Input 11111 per entrare nel nuovo stato "A"
  - **A**, **B**, **C** : gli stati sequenziali di inserimento, l'input in ogni stato viene salvato nel relativo registro, controllando che i valori siano coerenti con i posti massimi per ogni parcheggio. Dopo "C", prosegue nello stato di "Awake"
  - **AWAKE** : lo stato in cui il sistema è completamente attivo, l'input deve essere formattato come standard: 2 bit per l'ingresso od uscita (one hot) e 3 bit per la selezione del settore (one hot), tutti gli altri stati risultano in errore, mantenendo le sbarre chiuse ed il primo bit di output a 1 (bit di invalidità dell'input). L'input 00000 spegne il programma, tornando nello stato "Sleep"
- L'errore umano non viene considerato: Ci sarebbero molti casi in cui l'errore umano porterebbe ad un bisogno di ulteriori controlli, in primis l'uscita da un parcheggio già vuoto. Considerando lo scopo educativo del progetto ed il fatto che un buon design presuppone che l'utente sia portato a NON commettere errori (magari lasciando un unico bottone al suo ingresso/uscita invece di una tastiera), il programma viene considerato come un componente di un sistema più grande e complesso e, come tale, presuppone input corretti.
- Il testo dell'elaborato cita*: `La sbarra rimane aperta per un solo ciclo di clock, dopo di che viene richiusa`*, non essendo il nostro circuito dotato di clock a tempo (come ad esempio un quarzo), il clock è definito dall'inserimento di una nuova sequenza di input, pertanto la sbarra rimarrà sollevata fino all'inserimento di un nuovo input, che deciderà se sollevare nuovamente la sbarra o chiuderla

STG


0-----/000000
-0----/000000
--0---/000000
---0--/000000
----0-/000000
-----0/000000
Sleep
11111/000000
A
-----/000000
B
-----/000000
C
-----/000000
00000/000000
11---/100000
0011-/100000
001-1/100000
00-11/100000
Awake
01100/010100
01010/010010
01001/010001
10100/001100
10010/001010
10001/001010

DATAPATH

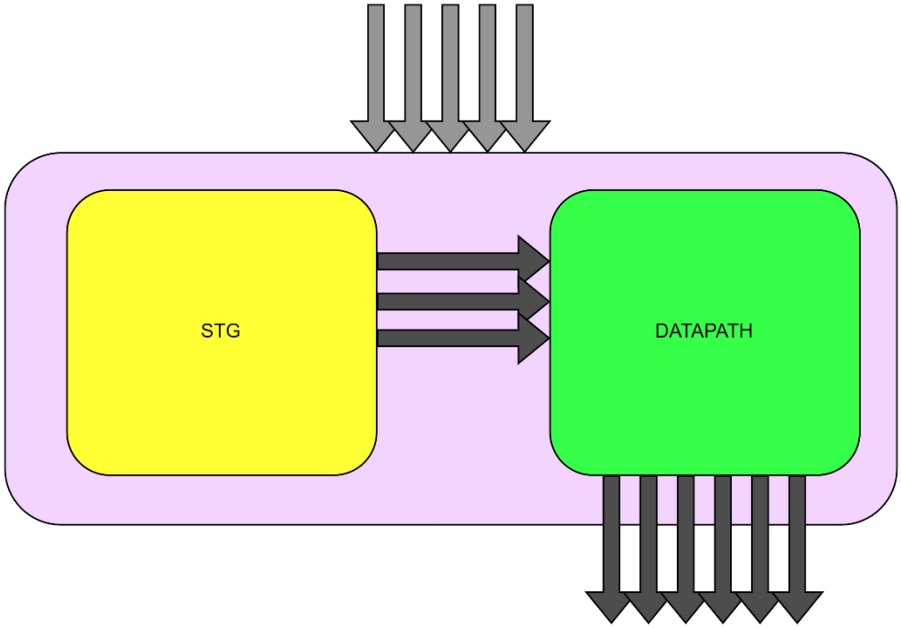


input 5 bit
control
MS
bit settore A - 01
bit settore B - 10
bit settore C - 11
takeA
takeB
takeC
const one
const max
cost max (overflow)
uguale a 24
uscita
ingresso
giorno/notte
bit C pieno
bit B pieno
bit A pieno
sbarra uscita
sbarra ingresso
bit di invalidità

Per il disegno del datapath è stato utilizzato Logisim[1],software gratuito per la progettazione di circuiti logici. Esso ci permette di testare il circuito prima di implementarlo su SIS.
Alcune note per la corretta lettura del datapath:

- Tutte le costanti sono definite come valori esadecimali
- Il diagramma si legge da sinistra verso destra, gli input 5 bit, 2 bit, 1 bit, sono rispettivamente: l'input dell'utente, il controllo dello stato d'inserimento (01: A, 10: B, 11: C) ed il bit di controllo dell'accensione (1: acceso, ovvero stato di "Awake")

FSMD



COMPONENTI

I componenti nella cartella sis/ vengono utilizzati per lo sviluppo dell'applicativo e sono riportati nel codice blif. Questa è una lista di tutti i componenti utilizzati e la loro funzione:

- **And**: porta logica and
- **Awc**: componente per controllare se si è nello stato awake(controlla se i tre bit in ingesso sono uguali a 100)
- **Ccont**: converte i 5 bit in ingresso in 1 1 0 0 0 se maggiori altimenti restituisce i bit in ingresso
- **CK**: porta xor e porta and messe in serie
- **Demulti1b**: un demultiplexer a 4 uscite da un bit regolato da 2 bit
- **Demulti5b**: 5 Demulti1b messi insieme per formare un demultiplexer a 4 uscite da 5 bit regolato da 2 bit
- **Demulti8b**: un demultiplexer a 8 uscite monobit regolato da 3 bit
- **Diff1b**: calcola la differenza fra due bit
- **Eq**: restituisce 1 se i due bit sono uguali
- **Eq2b**: eq ma su due bit
- **Giaf**: serie di porte end e or che descrivono l'ingresso di una vettura in uno spazio pieno
- **Minus1**: utilizando Diff1b riceve 5 bit in entrata e ne restituisce il valore -1

- **Multi5b.blif**: un multiplexer con 2 entrate a 5 bit regolato da un bit
- **Not**: porta logica not
- **one**: rende costante il bit a 1
- **Or**: porta logica or
- **Or6b**: porta logica or a 6 bit
- **Plus1**: utilizzando Somma1b riceve 5 bit in entrata e ne restituisce il valore +1
- **reg1b**: registro a un bit
- **reg5b**: 5 registri a un bit
- **SbarraCK**: condizioni per cui una sbarra si dovrebbe alzare (notte o direzione e settore)
- **Somma1b**: calcola la somma fra due bit
- **stampa**: riceve in input un valore e lo mette come output (usato per uscire dal loop del registro)
- **TakeA/B/C**: insieme di operazioni necessarie per la memorizzazione del numero di auto nel Settore A/B/C
- **Uguale0** : restituisce 1 se i 5 bit in ingresso sono 00000
- **Uguale24**: restituisce 1 se i 5 bit in ingresso sono 11000
- **Uguale25**: restituisce 1 se i 5 bit in ingresso sono 11000
- **Uguale31**: restituisce 1 se i 5 bit in ingresso sono 11111
- **xor**: porta logica xor
- **zero**: rende costante il bit a 0

STATISTICHE ED OTTIMIZZAZIONE

Pre ottimizzazione:

FSMD pi= 5 po= 6 nodes=301 latches=18 lits(sop)=1145

Dopo l'esecuzione delle operazioni:

- source script.rugged
- simplify
- fx

Abbiamo ottenuto:

FSMD pi= 5 po= 6 nodes= 71 latches=18 lits(sop)= 378

Reference

[1]: http://www.cburch.com/logisim/