Elaborato SIS

Laboratorio di Architettura degli Elaboratori

A.A. 2020/2021

Brazzarola David VR457898
Furri Geremia VR456135
Quintarelli Federico VR457997

Sommario

Gestione Del Progetto

Come gruppo abbiamo deciso di gestire il codice servendoci di Github in modo da poter sfruttare i vantaggi offerti dal versioning.

Ogni componente del gruppo ha lavorato su diverse parti del codice in modo da avere una visione a 360 gradi del progetto, anche la relazione è stata redatta da tutti i componenti.

Scelte Progettuali

La consegna dell'elaborato, in alcuni punti, non è specifica nei confronti del funzionamento della macchina, dunque abbiamo creato questa apposita sezione in modo da specificare le nostre ipotesi aggiuntive e scelte progettuali:

- Non è possibile cancellare l'inserimento di una cifra del codice, l'operazione di inserimento della singola cifra è irreversibile.
- Analogamente alle cifre, non è possibile re-inserire l'importo richiesto, in caso di errata digitazione del cash richiesto la macchina processerà le informazioni acquisite, ed in base ad esse, erogherà o annullerà l'operazione. In entrambi i casi la carta del bancomat dovrà essere estratta. Dunque, per eseguire un altro tentativo, si dovrà re-introdurre la carta nel bancomat.
- Anche se il testo fornitoci è chiaro è meglio precisare che la nostra macchina non tratta in alcun modo numeri decimali. Tutte le operazioni sono eseguite su numeri interi, anche quelle di shifting, andando, ovviamente, a perderne in precisione. Per questo motivo la codifica utilizzata è quella in modulo. (es.: se l'utente chiede 1023 euro e, all'interno della macchina, ne sono presenti 4095, l'importo non verrà erogato in quanto 4095/4, codificato in modulo, risulta 1023)
- Sono stati aggiunti i bit INIT e BLOCCARE tra FSM e Datapath in modo da gestire il conteggio dei tentativi, capitolo correlato.
- Il bancomat non permette l'estrazione della carta di credito fino a che non è terminata la transazione, che essa sia fallita o meno. Perciò nel caso in cui venga inserita la carta gli unici modi per liberare il bancomat sono: sapere il codice oppure sbagliare 3 volte consecutive e farsi ritirare la carta dal dispositivo. Per questo motivo il bit REINSERIRE_BANCOMAT è a don't care in qualunque caso tranne quando viene inserita la carta nel dispositivo dal cliente.

Funzionamento

La macchina si divide in due parti principali:

- **FSM**, il controllore
- **DATAPATH**, l'elaboratore

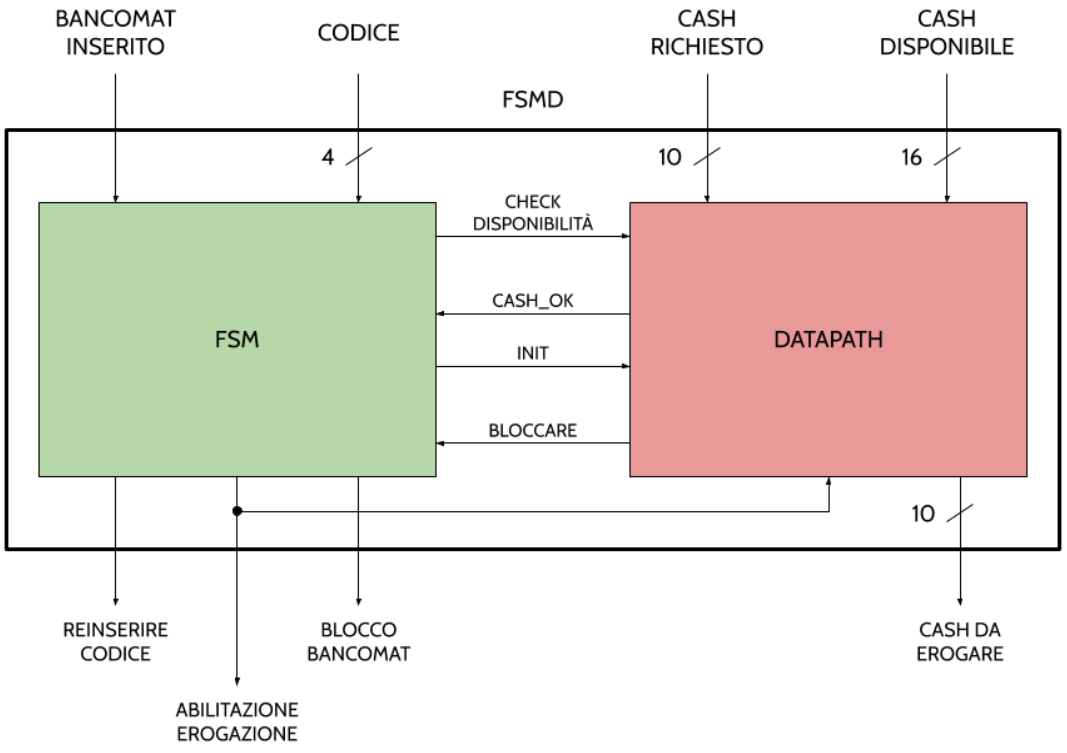
Questi due componenti interagiscono tra di loro in modo da permettere alla macchina di funzionare correttamente.

Il dispositivo che abbiamo costruito implementa le funzionalità base di un bancomat. L'utente, giunto al dispositivo, inserisce la propria carta che verrà riconosciuta da un sensore apposito, a questo punto il bancomat si aspetta un PIN di tre cifre in input dall'utente per verificare la validità della carta.

Se tale PIN risulterà corretto si proseguirà. L'utente ha a disposizione tre tentativi per immettere correttamente il PIN. Nel caso in cui il PIN venga sbagliato tre volte consecutive il dispositivo ritirerà la carta ponendola in una cassetta di sicurezza al suo interno e mostrando sul display una segnalazione di blocco della carta. Alla fine di ciò il bancomat tornerà disponibile per altri clienti.

Una volta digitato correttamente il PIN l'utente potrà inserire l'importo desiderato. Se tale quantità risulterà valida, dopo il controllo eseguito dal sistema, l'ammontare richiesto verrà erogato e la carta potrà essere estratta, liberando il bancomat. Se, invece, al termine dei controlli non sarà abilitata l'erogazione l'utente deve recuperare la carta e, in caso, riprovare dall'inizio. Anche in quest'ultimo caso il bancomat verrà liberato.

Schema della FSMD



Lo schema della FSMD del nostro dispositivo si differenzia da quella specificata nella consegna per 2 bit tra FSM e DATAPATH utili per alcune funzionalità, questi 2 bit non sono riportati né in ingresso né in uscita della FSMD.

Inputs e Outputs

La FSMD, nei diversi istanti, prende in input 31 bit divisi in questo modo:

- `BANCOMAT_INSERITO[1]`: segnale generato da un sensore che individua la carta nell'apposito inserto
- `CODICE[4]`: corrisponde ad una singola cifra del PIN
- `CASH_RICHIESTO[10]`: l'ammontare richiesto dal cliente
- `CASH_DISPONIBILE[16]`: la disponibilità di denaro presente nel bancomat

In output, invece, escono 13 bit ad ogni istante:

- `REINSERIRE_CODICE[1]`: segnala quando è necessario inserire nuovamente il PIN perchè sbagliato
- `ABILITAZIONE_EROGAZIONE[1]`: segnala quando il controllo del cash va a buon fine e dunque l'erogazione è abilitata
- `BLOCCO_BANCOMAT[1]`: segnala quando il PIN è stato inserito in modo errato per 3 volte consecutive e quindi la carta viene bloccata
- `CASH_DA_EROGARE[10]`: l'ammontare erogato se il controllo va a buon fine

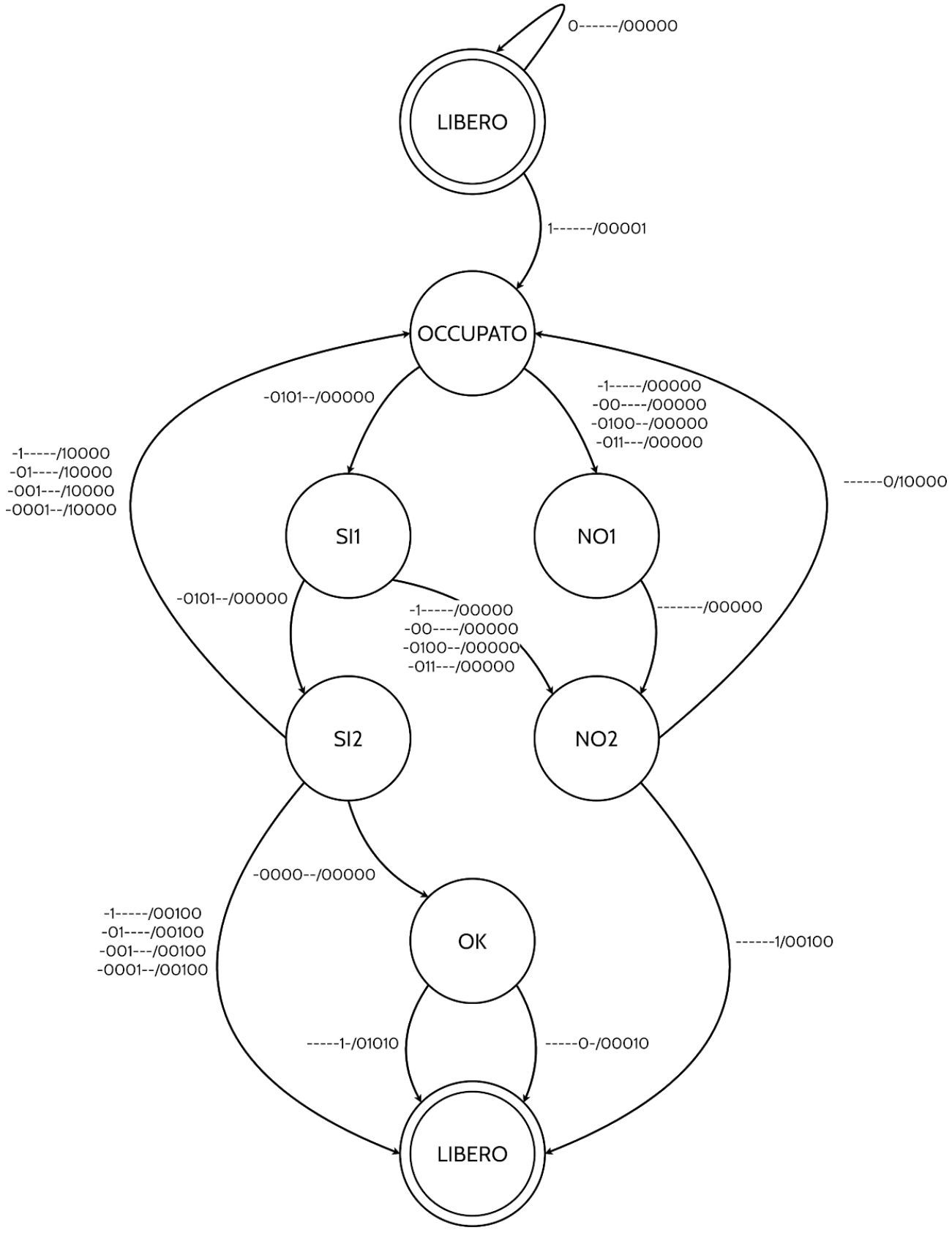
Tra FSM e DATAPATH sono stati inseriti dei bit in modo da poter gestire le operazioni di blocco e controllo del cash. Inoltre, come si vede dallo schema, il bit `REINSERIRE_CODICE` oltre ad essere in output della FSMD è messo in input al DATAPATH.

I bit `CHECK_DISPONIBILITÀ` e `CASH_OK` sono necessari per il controllo del denaro richiesto, quando `CHECK_DISPONIBILITÀ` vale 1 si attiva il controllo dei soldi richiesti e il risultato sarà posto in `CASH_OK`.

I bit `INIT`, `REINSERIRE_CODICE` e `BLOCCARE` sono fondamentali per gestire il controllo dei tentativi del PIN. `INIT` è necessario per resettare il registro che manterrà il contatore dei tentativi nel DATAPATH; solo quando `REINSERIRE_CODICE` vale 1 il codice dovrà essere re-inserito, allo stesso tempo verrà incrementato il valore del registro. Quando il valore del registro corrisponde a 2 (in decimale, 10 in binario) saranno già stati eseguiti 2 tentativi e `BLOCCARE` verrà posto ad 1. In questo modo se l'ultimo tentativo sarà di nuovo errato verrà posto a 1 `BLOCCO_BANCOMAT` e la carta verrà ritirata e bloccata.

Controllore

Questa immagine rappresenta l'STG del controllore.



Il controllore è una FSM (Finite State Machine) di Mealy composta da 7 ingressi e 5 uscite:

Gli ingressi sono i seguenti:

- `BANCOMAT_INSERITO`: vale 1 se è stato rilevato un bancomat, altrimenti 0
- `CODICE3`: cifra binaria tra 0 e 1
- `CODICE2`: cifra binaria tra 0 e 1
- `CODICE1`: cifra binaria tra 0 e 1
- `CODICE0`: cifra binaria tra 0 e 1 (tutte e 4 formano la cifra inserita per il codice)
- `CASH_OK`: l'importo richiesto può essere erogato (fornito dal datapath)
- `BLOCCARE`: se vale 1 sono avvenuti consecutivamente 2 inserimenti errati, se viene sbagliato anche il terzo la carta viene bloccata e posto a 1 il relativo bit in uscita

Le uscite sono le seguenti:

- `REINSERIRE_CODICE`: se vale 1 indica che il codice inserito non è corretto
- `ABILITAZIONE_EROGAZIONE`: se vale 1 indica che può procedere all'erogazione
- `BLOCCO_BANCOMAT`: vale 1 quando sono inseriti 3 codici errati per la stessa sessione
- `CHECK_DISPONIBILITÁ`: vale 1 quando il codice è corretto comunicando al datapath di eseguire i controlli
- `INIT`: quando vale 1 resetta il registro dedicato al conteggio degli errori. Ciò avviene ogni volta che si inizia una nuova sessione

Stati della FSM

La FSM è costituita da 7 stati:

- `LIBERO`: lo stato iniziale, il bancomat aspetta l'inserimento di una carta di credito; è anche lo stato in cui ritorna la macchina alla fine della transazione (qualunque sia il suo esito).
- `OCCUPATO`: è stata inserita una carta di credito nel bancomat;
- `SI1`: la prima cifra del codice è corretta;
- `SI2`: la prima e la seconda cifra del codice sono corrette;
- `NO1`: la prima cifra del codice è errata;
- `NO2`: almeno una delle prime 2 cifre è errata;
- `OK`: codice corretto (550), procede al controllo che, in caso, abilità l'erogazione.

Statistiche del circuito

Prima della minimizzazione:

```
sis> print_stats
CONTROLLORE      pi= 7   po= 5   nodes=  5       latches= 0
lits(sop)=    0  #states(STG)=   7
sis>
```

Dopo la minimizzazione:

Per minimizzare gli stati utilizzo il comando: `state_minimize stamina`

```
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 7
Number of states in minimized machine : 7
```

La codifica degli stati viene eseguita da sis con il comando: `state_assign jedi`

```
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> print_stats
CONTROLLORE      pi= 7   po= 5   nodes=  8       latches= 3
lits(sop)= 159  #states(STG)=   7
```

Ottimizzazione dell'area:

Procediamo con l'utilizzo dello `script.rugged`:

```
sis> source script.rugged
sis> print_stats
CONTROLLORE      pi= 7   po= 5   nodes= 11       latches= 3
lits(sop)=  38  #states(STG)=   7
sis>
```

eseguendo 2 volte lo `script.rugged` otteniamo un'ottimizzazione migliore:

```
sis> source script.rugged
sis> print_stats
CONTROLLORE      pi= 7   po= 5   nodes= 10       latches= 3
lits(sop)=  38  #states(STG)=   7
```

Ulteriori utilizzi di altri comandi o dello stesso script non portano a nessun cambiamento nelle statistiche del circuito.

Mappatura del circuito

I valori interessanti sono il numero di gate ("total gate area") e il ritardo ("maximum arrival time"), in particolare gli ultimi valori.
Il comando utilizzato è: `map -m 0 -s`
L'opzione `-m` richiede a sis di prediligere ottimizzazioni a favore di una diminuzione dell'area, mentre `-s` ci serve per mostrare le statistiche alla fine dell'operazione.

```
sis> map -m 0 -s
warning: unknown latch type at node '{[17]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[18]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[19]}' (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:          8
total gate area:       944.00
maximum arrival time: (11.40,11.40)
maximum po slack:     (-4.60,-4.60)
minimum po slack:     (-11.40,-11.40)
total neg slack:      (-56.00,-56.00)
# of failing outputs:  8
>>> before removing parallel inverters <<<
# of outputs:          8
total gate area:       944.00
maximum arrival time: (11.40,11.40)
maximum po slack:     (-4.60,-4.60)
minimum po slack:     (-11.40,-11.40)
total neg slack:      (-56.00,-56.00)
# of failing outputs:  8
# of outputs:          8
total gate area:       896.00
maximum arrival time: (11.20,11.20)
maximum po slack:     (-4.60,-4.60)
minimum po slack:     (-11.20,-11.20)
total neg slack:      (-55.40,-55.40)
# of failing outputs:  8
sis>
```

Di seguito le statistiche finali del controllore:

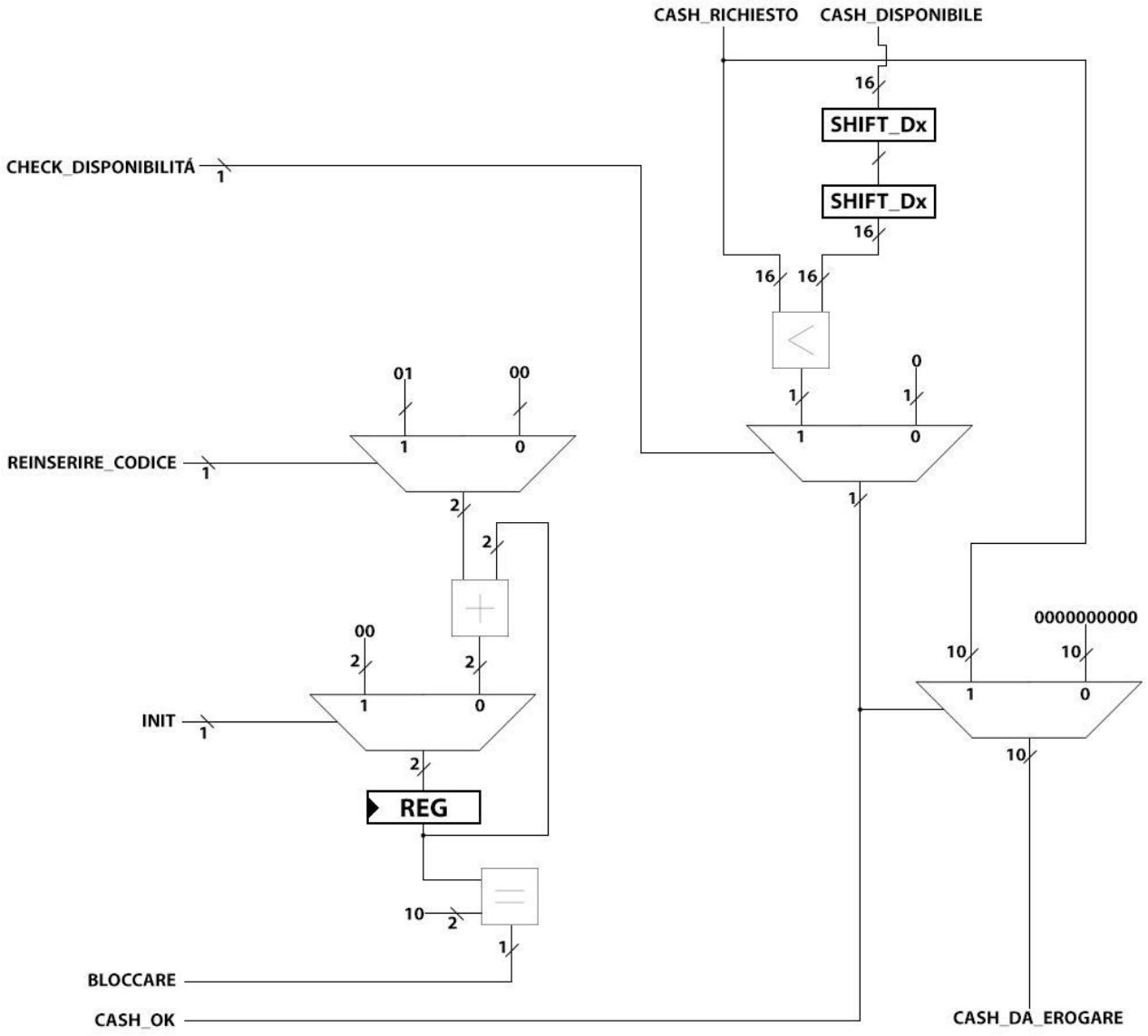
```
sis> print_stats
CONTROLLORE      pi= 7   po= 5   nodes= 24       latches= 3
lits(sop)=  55  #states(STG)=   7
sis>
```

Le condizioni dell'elaborato richiedevano di preferire una riduzione dell'area in fase di ottimizzazione, perciò mostriamo i risultati per altre possibilità di perfezionamento così da capire le scelte effettuate:

- Non è stato possibile effettuare la riduzione del ritardo per il cammino critico con il comando: `reduce_depth` dato che - nonostante il ritardo leggermente minore - ciò avrebbe causato un aumento dell'area andando contro le condizioni imposte.

Datapath

Questa immagine rappresenta lo schema del datapath.



Componenti del datapath

L'unità di elaborazione è un circuito diviso in due parti principali:

Counter tentativi, costituito dai seguenti elementi:

- Un multiplexer a 2 ingressi a 2 bit che tramite il bit di selezione INIT, ricevuto in input dal controllore, resetta il contenuto del registro facendo passare l'input "00" (INIT a 1), oppure aggiorna il registro col numero di tentativi di inserimento del pin dell'utente facendo passare il numero contenuto nel registro sommato ad 1 (INIT a 0)
- Un multiplexer a 2 ingressi a 2 bit che tramite il bit di selezione REINSERIRE_CODICE, ricevuto in input dal controllore, fa passare "01" o "00" a seconda del valore del bit selettore a 1 o 0
- Un registro a 2 bit utile al conteggio dei tentativi di inserimento del pin dell'utente
- Un sommatore a 2 bit che riceve in input il valore del registro e l'uscita del multiplexer che indica se è necessario reinserire il codice, e da in output la somma dei due al multiplexer di conteggio
- Un comparatore a 2 bit che riceve in input il valore del registro ed il valore "10" e pone a 1 il bit BLOCCARE, se quest'ultimo bit vale 1, l'utente ha a disposizione l'ultimo tentativo

Controllore della disponibilità del cash, costituito dai seguenti elementi:

- Uno shifter a 16 bit che prende in input i bit CASH_DISPONIBILE e restituisce in output il valore shiftato di una posizione a destra (equivalente di una divisione per 2)
- Uno shifter a 16 bit che prende in input i bit dello shifter precedente ed esegue la stessa operazione (equivalente di un'ulteriore divisione per 2, in totale una divisione per 4 dei bit CASH_DISPONIBILE)
- Un comparatore a 16 bit che riceve in input i 10 bit di CASH_RICHIESTO (i primi 6 bit di differenza sono posti a 0) e CASH_DISPONIBILE, e restituisce in output 1 o 0 a seconda del fatto che il valore dei primi bit sia minore del valore dei bit del secondo
- Un multiplexer a 2 ingressi da 1 bit che tramite il bit di selezione CHECK_DISPONIBILITA, ricevuto in input dal controllore, fa passare il risultato del comparatore oppure "0" sul bit CASH_OK, che va in input al controllore
- Un multiplexer a 2 ingressi da 10 bit che tramite il bit di selezione corrispondente all'uscita del multiplexer che verifica la disponibilità del cash fa passare i bit CASH_RICHIESTO oppure 10 bit a 0 sui bit CASH_DA_EROGARE

Statistiche del circuito

Prima dell'ottimizzazione, il warning su COUT non è preoccupante, deriva dalla somma:

```
sis@SIS:~/Scrivania/elaborato/elaborato-sis$ sis
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> read_blif datapath_base.blif
Warning: network `DATAPATH', node "COUT" does not fanout
sis> print_stats
DATAPATH        pi=29   po=12   nodes=135       latches= 2
lits(sop)= 492
```

Dopo l'ottimizzazione:

Per ottimizzare il circuito utilizzo il comando: `source script.rugged`

```
sis> source script.rugged
sis> print_stats
DATAPATH        pi=29   po=12   nodes= 22       latches= 2
lits(sop)= 162
```

Eseguendo 2 volte lo `script.rugged` il risultato non cambia

```
sis> source script.rugged
sis> print_stats
DATAPATH        pi=29   po=12   nodes= 22       latches= 2
lits(sop)= 162
sis> source script.rugged
sis> print_stats
DATAPATH        pi=29   po=12   nodes= 22       latches= 2
lits(sop)= 162
```

Ulteriori utilizzi di altri comandi o dello stesso script non portano a nessun miglioramento nelle statistiche del circuito.

Mappatura del circuito

Come per la mappatura del controllore, i valori interessanti sono il numero di gate ("total gate area") e il ritardo ("maximum arrival time"), in particolare gli ultimi valori.
Il comando utilizzato è: `map -m 0 -s`
L'opzione `-m` richiede a sis di prediligere ottimizzazioni a favore di una diminuzione dell'area, mentre `-s` ci serve per mostrare le statistiche alla fine dell'operazione.

```
sis> read_library synch.genlib
sis> map -m 0 -s
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:          14
total gate area:       1816.00
maximum arrival time: (19.40,19.40)
maximum po slack:     (-3.20,-3.20)
minimum po slack:     (-19.40,-19.40)
total neg slack:      (-231.00,-231.00)
# of failing outputs:  14
>>> before removing parallel inverters <<<
# of outputs:          14
total gate area:       1816.00
maximum arrival time: (19.40,19.40)
maximum po slack:     (-3.20,-3.20)
minimum po slack:     (-19.40,-19.40)
total neg slack:      (-231.00,-231.00)
# of failing outputs:  14
# of outputs:          14
total gate area:       1816.00
maximum arrival time: (19.40,19.40)
maximum po slack:     (-3.20,-3.20)
minimum po slack:     (-19.40,-19.40)
total neg slack:      (-231.00,-231.00)
# of failing outputs:  14
```

Di seguito le statistiche finali del controllore:

```
sis> print_stats
DATAPATH        pi=29   po=12   nodes= 63       latches= 2
lits(sop)= 158
```

Analogamente a quanto accade per il controllore, il comando `reduce_depth` non avrebbe senso dato il vincolo di riduzione dell'area.

FSMD

La FSMD è sostanzialmente l'unione tra DATAPATH e CONTROLLORE.

Statistiche del circuito

Una volta mappata ed ottimizzata la FSMD dovrebbe avere caratteristiche molto simili alla somma tra quelle del DATAPATH e del CONTROLLORE, anch'essi ottimizzati.

Le statistiche base della FSMD (non ancora ottimizzata né mappata) sono tali:

```
sis> print_stats
FSMD            pi=31   po=13   nodes= 85       latches= 5
lits(sop)= 211
```

Una volta eseguita l'ottimizzazione possiamo notare come diminuisce il numero di nodi e aumenta, seppur di poco, il numero di letterali. Al gruppo è sembrato un ottimo compromesso.

```
sis> print_stats
FSMD            pi=31   po=13   nodes= 31       latches= 5
lits(sop)= 222
```

Mappatura del circuito

La mappatura è stata eseguita con la libreria `synch.genlib`, questi sono i risultati:

```
>>> before removing serial inverters <<<
# of outputs:          18
total gate area:       2664.00
maximum arrival time: (21.80,21.80)
maximum po slack:     (-7.80,-7.80)
minimum po slack:     (-21.80,-21.80)
total neg slack:      (-320.40,-320.40)
# of failing outputs:  18
>>> before removing parallel inverters <<<
# of outputs:          18
total gate area:       2664.00
maximum arrival time: (21.80,21.80)
maximum po slack:     (-7.80,-7.80)
minimum po slack:     (-21.80,-21.80)
total neg slack:      (-320.40,-320.40)
# of failing outputs:  18
# of outputs:          18
total gate area:       2600.00
maximum arrival time: (21.80,21.80)
maximum po slack:     (-7.80,-7.80)
minimum po slack:     (-21.80,-21.80)
total neg slack:      (-319.80,-319.80)
# of failing outputs:  18
```

I valori fondamentali sono l'area totale di **2600.00** e il ritardo del circuito che vale **21.80**.

Anche in questo caso l'uso del comando `reduce_depth` non avrebbe fatto altro che aumentare considerevolmente l'area totale.

In conclusione le statistiche del circuito finali sono:

```
sis> print_stats
FSMD            pi=31   po=13   nodes= 87       latches= 5
lits(sop)= 214
```

Come è facile notare sono sostanzialmente la somma delle statistiche finali di DATAPATH e CONTROLLORE.

Fanout warning

```
sis> read_blif FSMD.blif
Warning: network `FSMD', node "B1" does not fanout
Warning: network `FSMD', node "B0" does not fanout
```

Questo warning di fanout deriva dall'operazione di shifting contenuta nel datapath, dunque non risulta essere preoccupante ma, bensì, del tutto lecito.

Simulazione

Solo da esempio abbiamo deciso di mostrare la simulazione di una erogazione andata a buon fine.

```
sis> source ./tests/testprelok.script

Network simulation:
Outputs: 0 0 0 0 0 0 0 0 0 0 0 0 0
Next state: 00111

Network simulation:
Outputs: 0 0 0 0 0 0 0 0 0 0 0 0 0
Next state: 00000

Network simulation:
Outputs: 0 0 0 0 0 0 0 0 0 0 0 0 0
Next state: 00001

Network simulation:
Outputs: 0 0 0 0 0 0 0 0 0 0 0 0 0
Next state: 00011

Network simulation:
Outputs: 0 1 0 1 1 1 1 1 0 0 0 0 0
Next state: 00110
```

Lo script utilizzato è il seguente:

```
sim 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sim 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sim 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sim 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sim 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Il primo bit è `BANCOMAT_INSERITO`, i successivi 4 la singola cifra del `CODICE`, i successivi 10 il `CASH_RICHIESTO` e, infine, gli ultimi 16 il `CASH_DISPONIBILE`.

`Next state` indica: nei primi due bit il valore del registro che mantiene il contatore dei tentativi e nei restanti tre il prossimo stato codificato in binario.