UNIVERSITÀ DEGLI STUDI DI VERONA

CORSO DI LAUREA IN INFORMATICA

Elaborato SIS

Laboratorio Architettura degli Elaboratori

Controllo dell'erogazione di denaro di un bancomat

Juri Farruku (VR464742)
Federico Segala (VR457144)
Davide Zamboni (VR455880)

A.A 2020/2021

Indice

Specifiche date

Il dispositivo da noi progettato consente di controllare l'erogazione di denaro di un bancomat. Tale dispositivo é stato modellato come circuito sequenziale composto da una parte di controllo (FSM) ed un'altra adibita all'elaborazione dei dati (Datapath).

Il circuito ha 4 ingressi nel seguente ordine:

- BANCOMAT_INSERITO (1 bit)
- CODICE (4 bit)
- CASH_RICHIESTO (10 bit)
- CASH_DISPONIBILE (16 bit)

Gli output sono nel seguente ordine:

- REINSERIRE_CODICE (1 bit)
- ABILITAZIONE_EROGAZIONE (1 bit)
- BLOCCO_BANCOMAT (1 bit)
- CASH_DA_EROGARE (10bit)

Il meccanismo è guidato come segue:

- Il segnale di ingresso BANCOMAT_INSERITO (da considerare derivante da un circuito esterno che rileva la presenza di un bancomat valido inserito nel macchinario) se uguale a 1 abilita la codifica dei numeri inseriti tramite il segnale di ingresso CODICE. Se uguale a 0, disabilita (pone a zero) tutte le uscite del circuito.

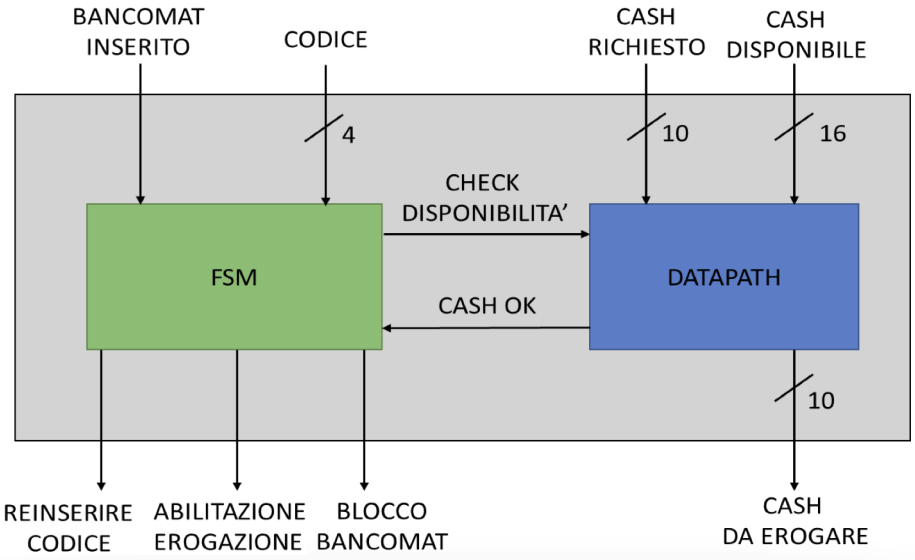
- L'analisi del CODICE inizia soltanto dopo che il BANCOMAT è stato inserito. Non è possibile inserire il BANCOMAT e la prima cifra del codice nello stesso momento.

- Una volta che il bancomat è stato inserito, viene inserito nel circuito il codice di autenticazione tramite il segnale di ingresso CODICE composto da 3 numeri inseriti in 3 istanti consecutivi, con range 0..9, codificati quindi con 4 bit.

- Una volta accertato che la sequenza numerica corrisponde a 5 5 0, il circuito riceve l'ammontare del cash richiesto dall'ingresso CASH_RICHIESTO (ammontare da 0 a 1023 euro, codificato con 10 bit) e attiva il controllo della disponibilità di banconote nella cassaforte, tramite il segnale interno CHECK_DISPONIBILITA (1 bit).
  Il controllo verifica se il cash richiesto è inferiore a 1/4 del cash disponibile in cassaforte, quest'ultimo ricevuto dal segnale di ingresso CASH_DISPONIBILE. Se inferiore allora il circuito abilita il segnale interno CASH_OK (1 bit), il quale fa abilitare il segnale di uscita ABILITAZIONE_EROGAZIONE, e riporta sul segnale di uscita CASH_DA_EROGARE l'importo richiesto. Altrimenti, tutti questi segnali rimangono posti a 0.

- Se il codice viene inserito in modo errato, il circuito abilita l'uscita REINSERIRE_CODICE.
- REINSERIRE_CODICE viene alzato solo al termine dell'inserimento dei codici che compongono il pin. Per esempio, se il codice inserito fosse 123, la porta viene messa ad 1 solo al termine dell'inserimento dell'intero codice e non già alla prima cifra inserita.
- Se il codice viene inserito in modo errato per 3 volte consecutive, il circuito abilita l'uscita BLOCCO_BANCOMAT.

Lo schema generale del circuito viene rappresentato dall'FSMD riportata di seguito



Architettura Generale del Circuito

Come detto precedentemente, il circuito è composto da un controllore (FSM) e da un elaboratore (Datapath) nominati rispettivamente come *FSM.blif* e *datapath.blif*.

Tali componenti sono poi racchiuse nel file principale FSMD.blif che permette di collegare la parte di controllo a quella di Datapath in modo tale che le due parti funzionino in maniera corretta.

Diagramma degli stati del controllore

Il controllore è una macchina a stati finiti (FSM) di Mealy che presenta in ordine:

- Tre **ingressi:** BANCOMAT_INSERITO (1bit), CODICE (4bit), CASH_OK (1bit);
- Quattro **uscite:** REINSERIRE_CODICE (1bit), ABILITAZIONE_EROGAZIONE (1bit), BLOCCO_BANCOMAT (1bit), CHECK_DISPONIBILITA (1bit);

Per permettere la richiesta del PIN fino a tre volte consecutive il controllore è stato ottenuto attraverso l'unione di tre blocchi.

Unità base FSM

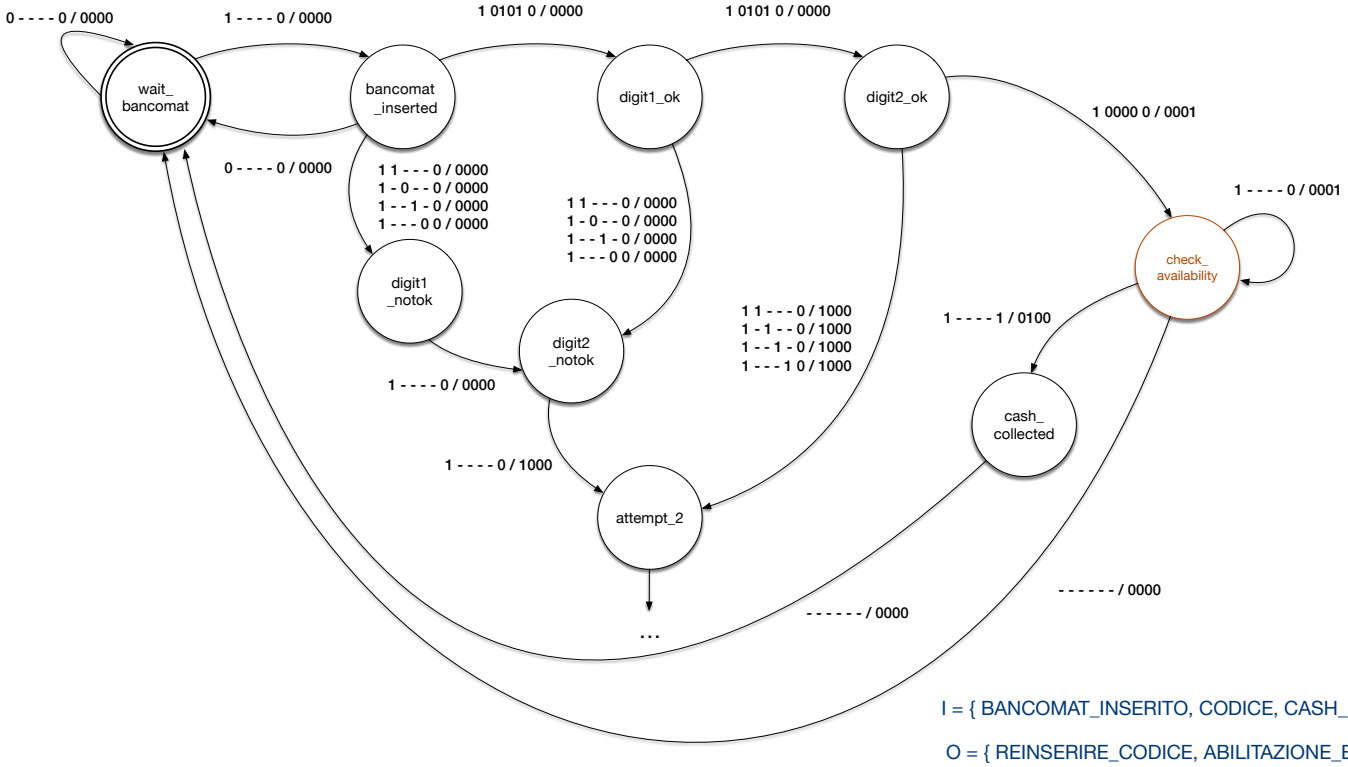
L'unità base di tali blocchi (ovvero quella corrispondente all'inserimento del PIN una sola volta) presenta gli stati illustrati di seguito:

- **wait_bancomat:** corrisponde allo stato di reset. L'FSM resta in questo stato fintantoché non viene inserito il bancomat. La macchina in altre parole è in attesa dell'arrivo di un nuovo utente il quale inserirà il bancomat;
- **bancomat_inserted:** l'FSM raggiunge questo stato quando il bancomat viene inserito. Si ritorna allo stato di reset qualora la tessera venga rimossa.
  Si noti che anche negli stati successivi, quando si andrà a richiedere il PIN, qualora l'input BANCOMAT_INSERITO sia uguale a zero, si ritorna in ogni caso allo stato di reset con output 0000. Per questione di chiarezza grafica all'interno del STG tali percorsi non vengono segnalati (si veda la sezione "Scelte Progettuali");
- **digit1_ok:** si passa a questo stato, partendo da bancomat_inserted, quando viene inserita la prima cifra corretta, cioè 5 (in binario equivale a 0101);
- **digit1_notok:** si passa a questo stato, partendo da bancomat_inserted, quando viene inserita una cifra che non corrisponde a 5 (quando quindi CODICE sarà 1- - -, - 0 - -, - - 1 - oppure - - - 0);
- **digit2_ok:** l'FSM raggiunge questo stato quando viene riconosciuta anche la seconda cifra (anch'essa equivalente a 5);
- **digit2_notok:** si arriva in questo stato quando o la seconda cifra inserita è sbagliata (lavorando in modo analogo a digit1_notok) oppure quando la prima cifra inserita era già sbagliata, indipendentemente dalla cifra inserita successivamente, come da specifica;
- **check_availability:** equivale allo stato di abilitazione dell' erogazione. Si raggiunge tale stato quando viene inserita la terza cifra corretta, cioè 0 (in binario equivale a 0000). Tale stato corrisponde alla parte di Datapath (evidenziata in arancio nello State Transition Graph). Se il Datapath ha verificato che il denaro richiesto è superiore a 1/4 del cash disponibile in cassaforte si ritornerà in questo stato.
  A questo punto, sta all'utente decidere se ritirare il suo bancomat (facendo tornare allo stato di reset l'FSM) oppure richiedere una cifra diversa da erogare. Nella sezione "Architettura del Datapath" tale stato verrà approfondito;

- **cash_collected:** si raggiunge questo stato quando il Datapath ha verificato che il denaro richiesto è inferiore a 1/4 del cash disponibile in cassaforte. Corrisponde dunque allo stato dove verrà abilitata l'erogazione del denaro;

Da trattare in modo particolare è lo stato attempt_2: l'FSM raggiunge questo stato quando o la terza cifra inserita è sbagliata oppure quando la prima e/o la seconda cifra inserita era già sbagliata. Corrisponde allo stato dove comincerà la richiesta del PIN per la seconda volta, dove verrà poi ripetuta l'unità base dell'FSM.

State Transition Graph - Unità base FSM



I = { BANCOMAT_INSERITO, CODICE, CASH_OK }

O = { REINSERIRE_CODICE, ABILITAZIONE_EROGAZIONE, BLOCCO_BANCOMAT, CHECK_DISPONIBLITA }

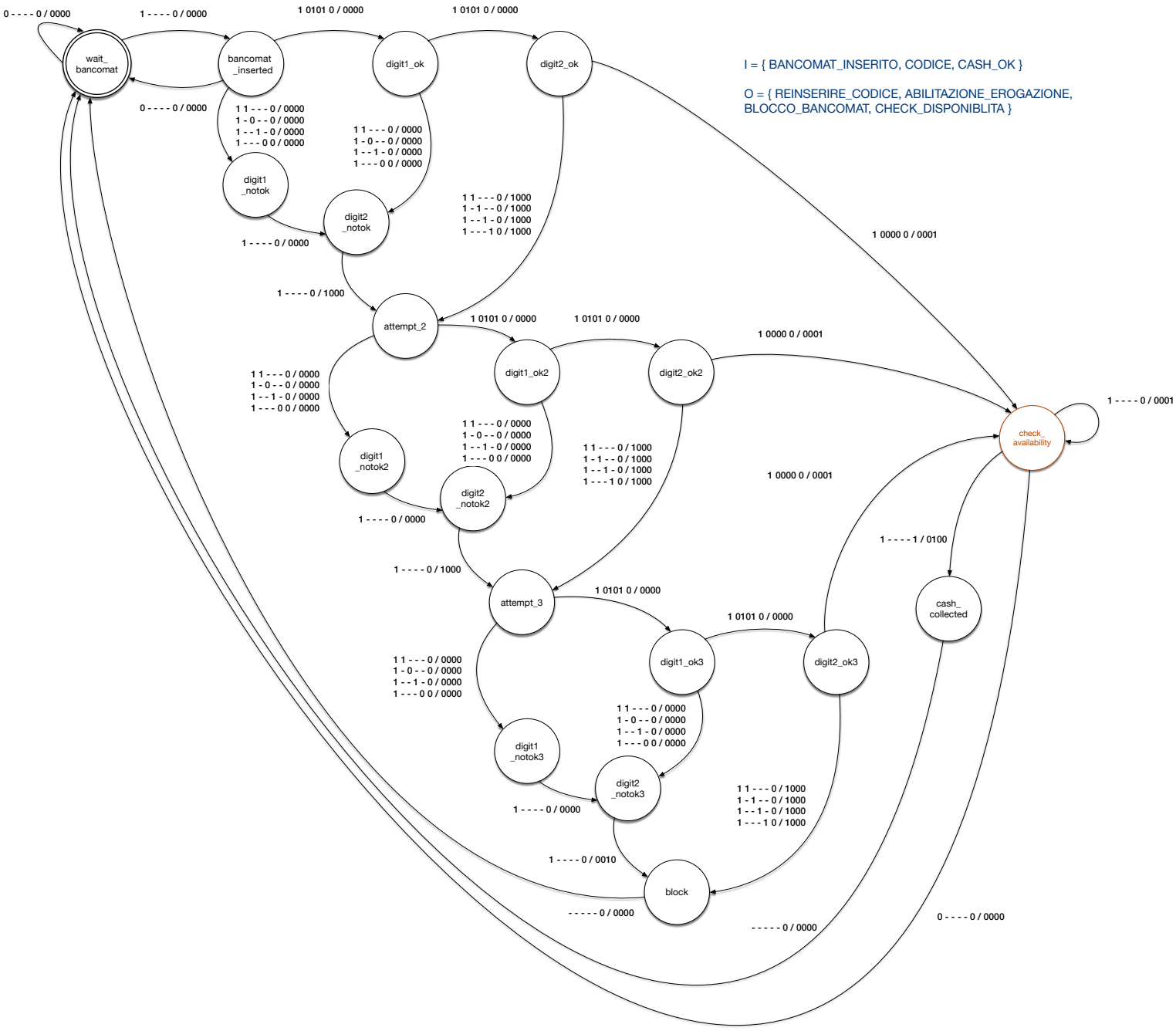
FSM Completa

Dopo aver analizzato l'unità base del controllore, l'FSM finale che si ottiene è composta dalla ripetizione per tre volte dell'unità sessa.

A questo punto restano da analizzare i seguenti stati:

- **attempt_3:** rappresenta lo stato raggiunto nel quale l'utente andrà incontro alla richiesta per la terza e ultima volta del PIN;
- **block:** una volta aver sbagliato il PIN per la terza volta si raggiungerà questo stato, dove il bancomat verrà bloccato abilitando l'uscita BLOCCO_BANCOMAT. Successivamente l'FSM tornerà allo stato di reset.

State Transition Graph - FSM Completa



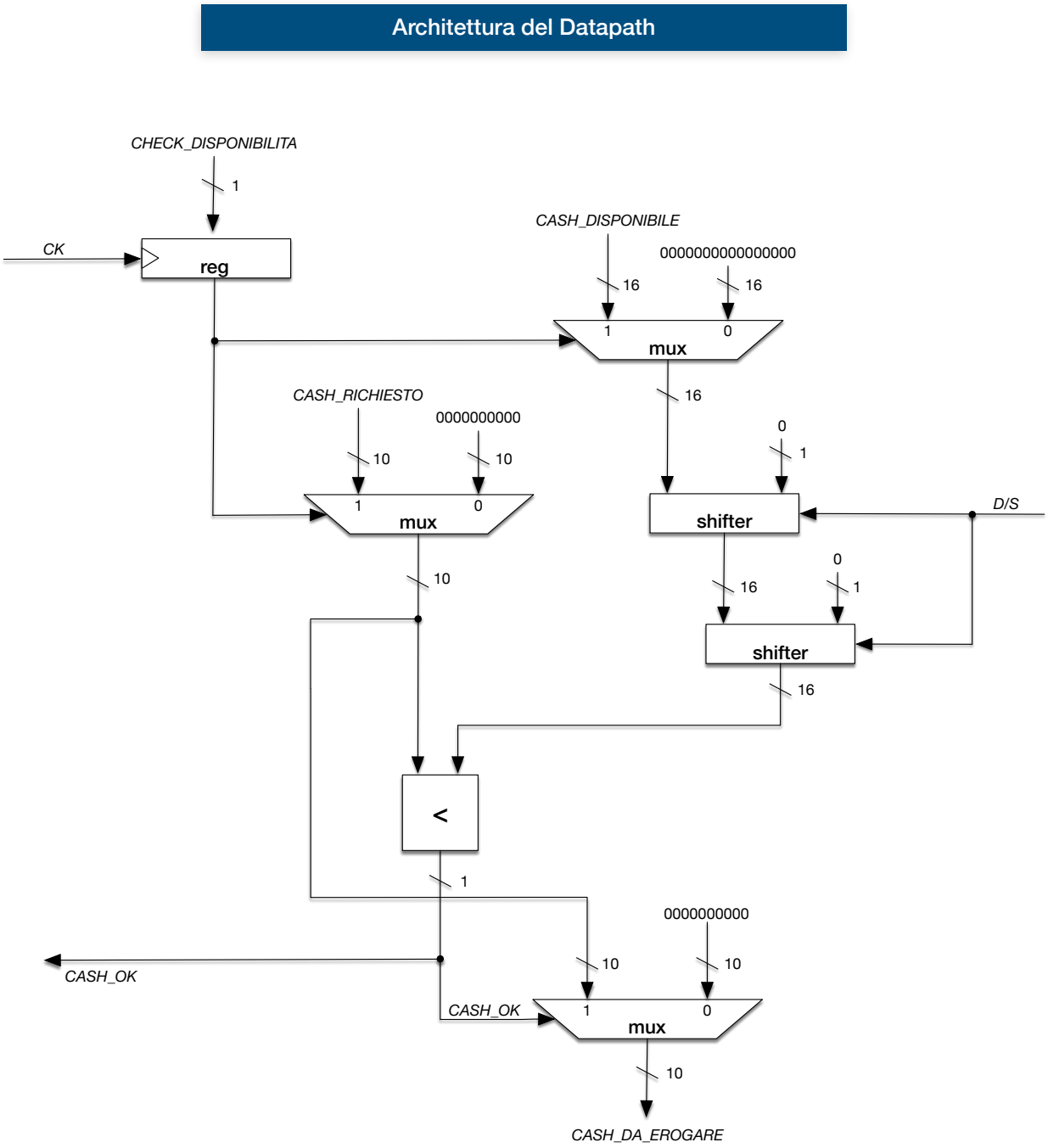
Architettura del Datapath

Per la parte di elaborazione dei dati abbiamo considerato come input *CHECK_DISPONIBILITA* (proveniente dalla FSM), *CASH_RICHIESTO* e *CASH_DISPONIBILE*, mentre come output *CASH_OK* (in uscita verso la FSM) e *CASH_DA_EROGARE*.

Successivamente abbiamo usato le seguenti componenti organizzate come qui descritto:

- Un **registro a 1 bit (reg.blif)** riceve dall'FSM il segnale di *CHECK_DSISPONIBILITA* (il registro è stato usato per evitare errori di cycle);
- Un **multiplexer a 10 bit (mux10.blif)** riceve l'output del registro, se questo è uno seleziona l'input *CASH_RICHIESTO*, altrimenti seleziona il valore 0 codificato su 10 bit;
- Lo stesso valore del registro prima descritto viene selezionato anche da un altro **multiplexer a 16 bit (mux16.blif),** il quale seleziona *CASH_DISPONIBILE* se il selettore è 1, altrimenti seleziona il valore 0 codificato su 16 bit;
- Il valore in uscita del multiplexer appena descritto viene preso come input da uno **shifter a 16 bit (shifter16.blif).** Il quale attraverso uno shift a destra (lo shifter esegue uno shift a destra quando riceve il valore 1) divide per 2 il valore *CASH_DISPONIBILE*. Tale valore viene ulteriormente diviso ancora per due in un ulteriore **shifter a 16 bit**, ottenendo così infine 1/4 del valore *CASH_DISPONIBILE,* necessario in seguito per il confronto;
- Il segnale in uscita dal primo multiplexer descritto e dal secondo shifter vengono confrontati in **minore a 16 bit (minore16.blif)**. L'unità di confronto verifica se il valore del primo multiplexer (quindi *CASH_RICHIESTO,* nel caso in cui il valore di *CHECK_DISPONIBILITA* è 1) è strettamente inferiore ad 1/4 del valore presente in *CASH_DISPONIBILE*. I 6 bit di lunghezza mancanti dal primo valore vengono aggiunti con degli zeri, in modo tale che il comparatore possa lavorare con entrambi i valori a 16 bit;
- Il risultato del confronto, cioè *CASH_OK,* va in output verso la FSM;
- *CASH_OK* infine viene preso in input da un secondo **multiplexer a 10 bit,** il quale produce in uscita *CASH_DA_EROGARE*. Qualora *CASH_OK* sia vero (1), tale valore corrisponderà a *CASH_RICHIESTO*, altrimenti sarà azzerato.

Di seguito la rappresentazione grafica del Datapath:



Statistiche del Circuito

Di seguito vengono illustrate le statistiche del circuito prima e dopo l'ottimizzazione per area.

Statistiche Prima dell'ottimizzazione

Dopo aver assegnato una codifica agli stati del controllore tramite il comando *state_assign jedi*, l'intera FSMD prima dell'ottimizzazione ha le seguenti statistiche:

```
sis> print_stats
FSMD            pi=31   po=13   nodes=185       latches= 6
lits(sop)= 842
```

Statistiche Dopo l'ottimizzazione

Per poter ottenere un circuito minimizzato rispetto l'area, è necessario ridurre il numero di letterali, dunque per prima cosa abbiamo minimizzato gli stati del controllore attraverso il comando *state_minimize stamina*, ottenendo una riduzione degli stati da 19 a 18. Successivamente viene assegnata una codifica con il comando *state_assign jedi:*

```
sis> read_blif FSM.blif
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 19
Number of states in minimized machine : 18
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
```

In seguito siamo passati all'ottimizzazione dell'intero circuito, lanciando uno *script.rugged* e successivamente ottimizzando in particolare l'area attraverso il comado *full_simplify*, ottenendo infine le seguenti statistiche:

```
sis> source script.rugged
sis> full_simplify
sis> print_stats
FSMD            pi=31   po=13   nodes= 34       latches= 6
lits(sop)= 226
```

Si noti come il numero di nodi sia diminuito sensibilmente, portando a buon fine il processo di minimizzazione.

Mapping

In seguito all'ottimizzazione il circuito viene mappato sulla libreria *synch.genlib* come da specifica, in modo tale da produrre delle statistiche più vicine ad un circuito reale per quanto riguarda area e ritardo.

```
sis> read_library synch.genlib
sis> map -s
```

Numero di Gate e Ritardo

Prima della minimizzazione:

```
>>> before removing serial inverters <<<
# of outputs:          19
total gate area:       5248.00
maximum arrival time: (39.20,39.20)
maximum po slack:     (-7.60,-7.60)
minimum po slack:     (-39.20,-39.20)
total neg slack:      (-573.60,-573.60)
# of failing outputs:  19
>>> before removing parallel inverters <<<
# of outputs:          19
total gate area:       5248.00
maximum arrival time: (39.20,39.20)
maximum po slack:     (-7.60,-7.60)
minimum po slack:     (-39.20,-39.20)
total neg slack:      (-573.60,-573.60)
# of failing outputs:  19
# of outputs:          19
total gate area:       5200.00
maximum arrival time: (39.20,39.20)
maximum po slack:     (-7.60,-7.60)
minimum po slack:     (-39.20,-39.20)
total neg slack:      (-573.60,-573.60)
# of failing outputs:  19
```

Dopo la minimizzazione:

```
>>> before removing serial inverters <<<
# of outputs:           19
total gate area:        3032.00
maximum arrival time: (33.00,33.00)
maximum po slack:     (-4.40,-4.40)
minimum po slack:     (-33.00,-33.00)
total neg slack:      (-368.60,-368.60)
# of failing outputs:  19
>>> before removing parallel inverters <<<
# of outputs:           19
total gate area:        3032.00
maximum arrival time: (33.00,33.00)
maximum po slack:     (-4.40,-4.40)
minimum po slack:     (-33.00,-33.00)
total neg slack:      (-368.60,-368.60)
# of failing outputs:  19
# of outputs:           19
total gate area:        3032.00
maximum arrival time: (33.00,33.00)
maximum po slack:     (-4.40,-4.40)
minimum po slack:     (-33.00,-33.00)
total neg slack:      (-368.60,-368.60)
# of failing outputs:  19
```

Scelte Progettuali

La prima questione progettuale alla quale siamo andati incontro risiedeva nello stabilire in che modo contare per tre volte l'inserimento errato del codice. Abbiamo dunque scelto, come descritto precedentemente, di organizzare l'FSM in un' unità base che poi viene ripetuta per tre volte, affinché questa specifica fosse realizzata.

Un'altra scelta progettuale è stata quella del come gestire il bancomat inserito una volta entrati nella fase di richiesta del PIN e successivamente nello stato check_availability o block. Ci è sembrata una soluzione convincente quella di far tornare allo stato di reset la macchina in qualsiasi momento BANCOMAT_INSERITO sia 0. Nello specifico nel momento in cui viene erogato denaro (cash_collected) o quando viene bloccato il bancomat (block) la macchina tornerà allo stato di reset indipendentemente dal valore di BANCOMAT_INSERITO.

Nel caso in cui venga segnalato troppo denaro richiesto rispetto a quello nella cassaforte è stato scelto poi scelto di dare all'utente la possibilità di richiedere una cifra di prelievo diversa, facendo rimanere l'FSM nello stato check_availability. L'utente può comunque, come già descritto, decidere di ritirare la carta, facendo tornare l'FSM allo stato di reset.

Un'ultimo punto da andare a considerare è stato quello di decidere se accettare la transazione di denaro qualora CASH_RICHIESTO fosse stato esattamente uguale ad 1/4 del CASH_DISPONIBILE. È stato scelto in questo caso particolare di non abilitare il segnale interno CASH_OK, in modo tale che vengano accettate solo richieste di denaro strettamente inferiori ad 1/4 di CASH_DISPONIBILE.