

ELABORATO SIS

ARCHITETTURA DEGLI ELABORATORI

A.A. 2020/2021 – Corso di Laurea in Informatica

Bianchini Davide (VR456697)

Bragastini Enrico (VR456374)

Mafficini Andrea (VR462441)

# 

# Sommario

[Sommario 2](#_Toc62745466)

[FSMD 3](#_Toc62745467)

[Controllore FSM 4](#_Toc62745468)

[**1.Interpretazione della specifica in linguaggio naturale** 2](#_Toc62745469)

[**2. Rappresentazione in state-transition-graph (STG)** **Mealy** 2](#_Toc62745470)

[**3. Scelte progettuali** 2](#_Toc62745471)

[**4. Rappresentazione in state-transition-table (STT)** 2](#_Toc62745472)

[Datapath 2](#_Toc62745473)

[**Datapath: Contatentativi** 2](#_Toc62745474)

[**Datapath: Check\_Cash** 3](#_Toc62745475)

[**Rappresentazione Grafica Datapath:** 4](#_Toc62745476)

[Esempio esecutivo 5](#_Toc62745477)

[Ottimizzazione e Mapping 6](#_Toc62745478)

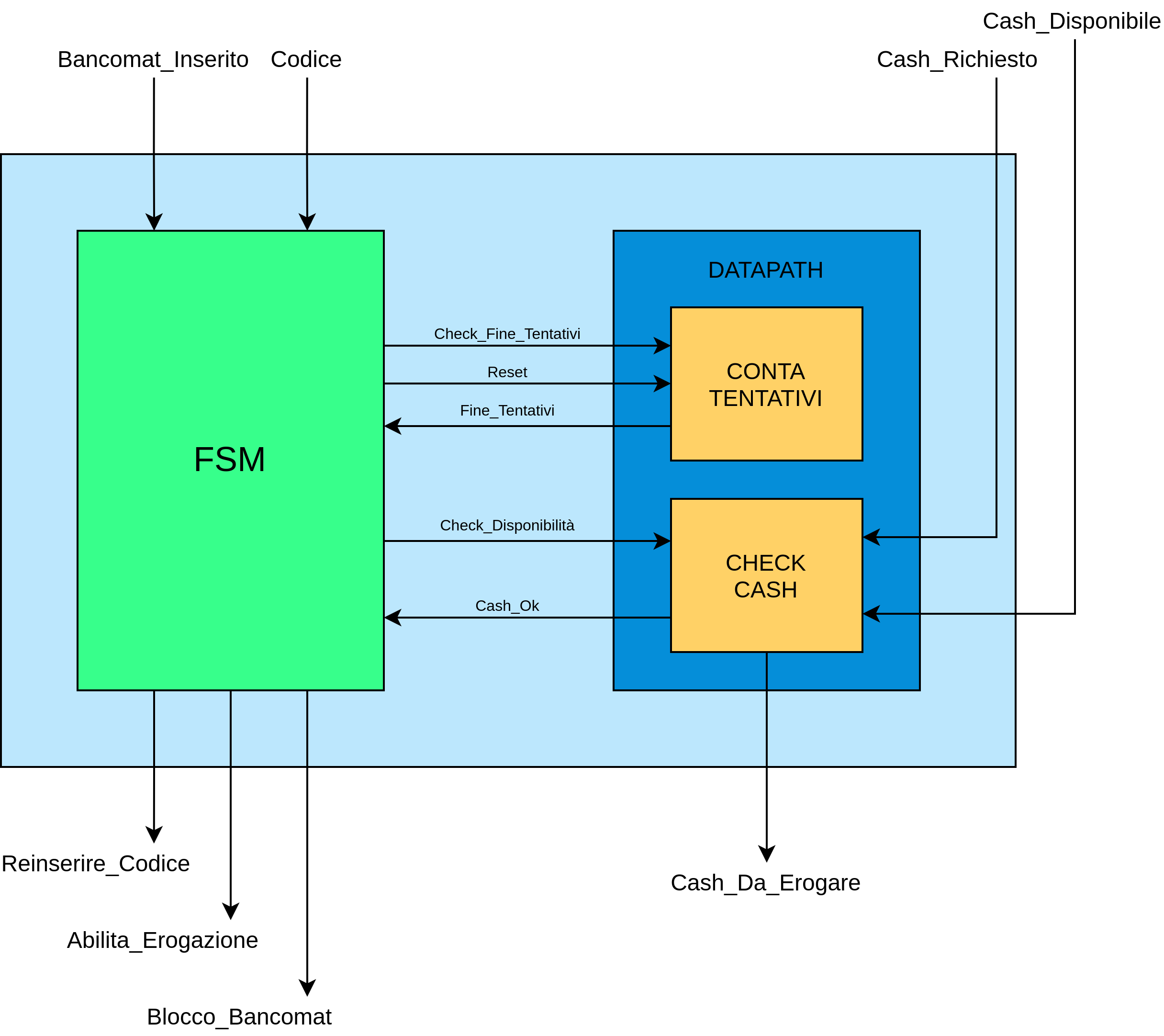
# FSMD

Lo schema generale del circuito da noi utilizzato rispecchia lo schema indicato nelle specifiche del progetto.

Il Datapath, per i motivi successivamente illustrati, viene diviso concettualmente in due circuiti distinti. Questo ha comportato l’aggiunta di alcuni segnali interni di comunicazione tra la FSM e il Datapath.

In particolare è stata fatta la *scelta progettuale* di lasciare il compito del conteggio dei tentativi errati al Datapath, per ridurre così il numero di stati presenti nel controllore. Come si può quindi notare dallo schema seguente, il Datapath è composto di due sezioni: una che si occupa di avvisare la FSM se l’utente ha inserito un codice errato per due volte di seguito e ora sta quindi tentando di inserire il terzo, l’altra sezione si occupa di leggere il cash richiesto dall’utente e il cash disponibile nella cassaforte, verificare che il primo sia minore di ¼ del secondo, e avvisare la FSM se questa condizione è stata rispettata. In caso positivo, scriverà sull’uscita dedicata l’importo richiesto dall’utente che verrà poi erogato.

Oltre ai segnali interni previsti dalla specifica, sono stati aggiunti i segnali che comunicano con la sezione di Datapath che si occupa di gestire contare il numero di tentativi.



# Controllore FSM

**1.Interpretazione della specifica in linguaggio naturale**

Il circuito richiesto, deve controllare l’erogazione di denaro di un bancomat.

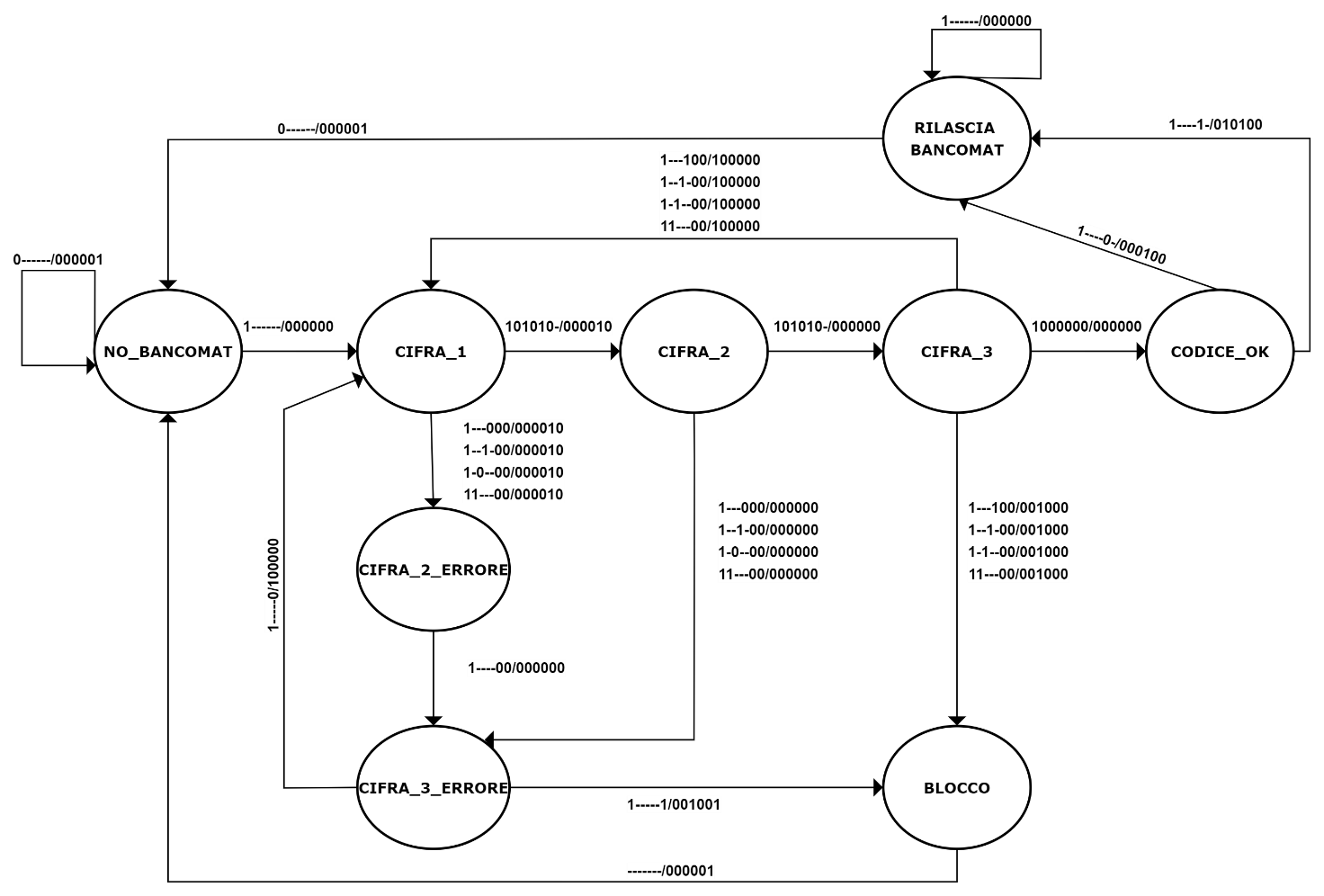
Gli ingressi sono: bancomat\_inserito, codice, cash\_richiesto, cash\_disponibile;   
Le uscite sono: reinserire\_codice, abilitazione\_erogazione, blocco\_bancomat e cash\_da\_erogare.

Dopo aver inserito il bancomat, verrà chiesto di inserire un pin di 3 cifre, una volta inserite tutte le cifre, il circuito deve controllare la correttezza del pin. A questo punto, verrà chiesta la quantità di denaro che si vuole prelevare. Il circuito, una volta controllato che ci siano contanti sufficienti per il prelievo (uscita abilitazione\_erogazione a 1) erogherà i soldi richiesti. (bit di cash\_da\_erogare).

Si hanno 3 tentativi per l’inserimento del pin, se si inserisce il pin scorretto viene richiesto nuovamente di reinserirlo (uscita reinserire\_codice a 1). Se il pin viene inserito errato per 3 volte consecutive, il bancomat si bloccherà (uscita blocco\_bancomat a 1).

Se non ci sono abbastanza soldi per il prelievo, non verrà erogato alcun denaro.

**2. Rappresentazione in state-transition-graph (STG)** **Mealy**



**3. Scelte progettuali**

Uno ad uno, analizziamo gli stati con i rispettivi ingressi e uscite.

**Ingressi** = {Bancomat\_Inserito, Codice3, Codice2, Codice1, Codice0, Cash\_Ok, Fine\_Tentativi}

**Uscite** = {Reinserire\_Codice, Abilitazione\_Erogazione, Blocco\_Bancomat, Check\_Disponibilita, Check\_Fine\_Tentativi, Reset}

**Stato NO\_BANCOMAT**: finché non viene inserito il bancomat, la macchina rimane nello stato NO\_BANCOMAT. Una volta inserito il bancomat, il primo bit di input sale a 1, per permettere di passare allo stato successivo: CIFRA\_1.

**Stato CIFRA\_1**: se nei 4 bit del codice viene inserito il numero 5 (0101) si passerà allo stato CIFRA\_2.  
Per ogni altro numero inserito (input diverso da 0101), si passerà allo stato CIFRA\_2\_ERRORE.   
In entrambi i casi per scelta progettuale si abilita a 1 l’uscita Check\_Fine\_Tentativi, in modo tale da chiedere al datapath se l’attuale tentativo è l’ultimo a disposizione dell’utente.

**Stato CIFRA\_2**: se nei 4 bit del codice viene inserito il numero 5 (0101) si passerà allo stato CIFRA\_3  
Per ogni altro numero inserito (input diverso da 0101), si passerà allo stato CIFRA\_3\_ERRORE.

**Stato CIFRA\_2\_ERRORE**: qualsiasi sia l’input del codice (----), si passerà allo stato CIFRA\_3\_ERRORE.  
Questo perché una cifra è già stata sbagliata, l’inserimento del codice continuerà tenendo conto che è stato inserito errato.

**Stato CIFRA\_3\_ERRORE**: Se il datapath restituisce 1 sulla linea Fine\_Tentativi, l’utente ha terminato i tentativi e quindi si abilita l’uscita blocco che porterà la macchina allo stato BLOCCO.  
Altrimenti, qualsiasi sia l’input del codice (----), si passerà allo stato CIFRA\_1 e verrà abilitato l’output Reinserire\_Codice, per permettere di reinserire il pin.

**Stato CIFRA\_3**: se nei 4 bit del codice viene inserito il numero 0 (0000) si passerà allo stato successivo CODICE\_OK. Nel caso in cui non venga inserito 0, analogamente allo stato CIFRA\_3\_ERRORE, si passerà allo stato CIFRA\_1 oppure allo stato BLOCCO.

**Stato BLOCCO**: la macchina, dopo aver bloccato il bancomat, lo ritira (per scelta progettuale) e torna allo stato NO\_BANCOMAT per ripartire con un nuovo prelievo.

**Stato CODICE\_OK**: stato in cui si inserirà il denaro da prelevare.   
Se la cifra inserita in relazione con i soldi in cassaforte è congrua per essere prelevata, viene settato a 1 l’uscita Abilitazione\_Erogazione. Per effettuare il controllo viene abilitata l’uscita Check\_Disponibilita e allo stesso tempo viene valutata l’entrata Cash\_Ok. (Ai calcoli ci penserà il Datapath)  
Se dopo aver fatto il controllo, la cifra non risulta congrua, il bit Abilitazione\_Erogazione rimane a 0.   
In entrambi i casi, lo stato successivo sarà RILASCIA\_BANCOMAT.

**Stato RILASCIA\_BANCOMAT**: la macchina aspetta che venga espulsa la carta bancomat, ovvero che venga settato a 0 il bit Bancomat\_Inserito. Torna quindi allo stato NO\_BANCOMAT per ripartire con un nuovo prelievo.

**4. Rappresentazione in state-transition-table (STT)**

Legenda (Ingressi e Uscite):

* i6: Bancomat\_Inserito
* i5-i2: Codice
* i1: Cash\_Ok
* i0: Fine\_Tentativi
* o5: Reinserire\_Codice
* o4: Abilitazione\_Erogazione
* o3: Blocco\_Bancomat
* o2: Check\_Disponibilita
* o1: Check\_Fine\_Tentativi
* o0: Reset

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STATO ATTUALE | i6 | i5 | i4 | i3 | i2 | i1 | i0 | STATO PROSSIMO | o5 | o4 | o3 | o2 | o1 | o0 |
| NO\_BANCOMAT | 0 | - | - | - | - | - | - | NO\_BANCOMAT | 0 | 0 | 0 | 0 | 0 | 1 |
| NO\_BANCOMAT | 1 | - | - | - | - | - | - | CIFRA\_1 | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_1 | 1 | - | - | - | 0 | 0 | 0 | CIFRA\_2\_ERRORE | 0 | 0 | 0 | 0 | 1 | 0 |
| CIFRA\_1 | 1 | - | - | 1 | - | 0 | 0 | CIFRA\_2\_ERRORE | 0 | 0 | 0 | 0 | 1 | 0 |
| CIFRA\_1 | 1 | - | 0 | - | - | 0 | 0 | CIFRA\_2\_ERRORE | 0 | 0 | 0 | 0 | 1 | 0 |
| CIFRA\_1 | 1 | 1 | - | - | - | 0 | 0 | CIFRA\_2\_ERRORE | 0 | 0 | 0 | 0 | 1 | 0 |
| CIFRA\_1 | 1 | 0 | 1 | 0 | 1 | 0 | - | CIFRA\_2 | 0 | 0 | 0 | 0 | 1 | 0 |
| CIFRA\_2 | 1 | - | - | - | 0 | 0 | 0 | CIFRA\_3\_ERRORE | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_2 | 1 | - | - | 1 | - | 0 | 0 | CIFRA\_3\_ERRORE | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_2 | 1 | - | 0 | - | - | 0 | 0 | CIFRA\_3\_ERRORE | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_2 | 1 | 1 | - | - | - | 0 | 0 | CIFRA\_3\_ERRORE | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_2 | 1 | 0 | 1 | 0 | 1 | 0 | - | CIFRA\_3 | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | - | - | - | 1 | 0 | 0 | CIFRA\_1 | 1 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | - | - | 1 | - | 0 | 0 | CIFRA\_1 | 1 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | - | 1 | - | - | 0 | 0 | CIFRA\_1 | 1 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | 1 | - | - | - | 0 | 0 | CIFRA\_1 | 1 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | - | - | - | 1 | 0 | 0 | BLOCCO | 0 | 0 | 1 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | - | - | 1 | - | 0 | 0 | BLOCCO | 0 | 0 | 1 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | - | 1 | - | - | 0 | 0 | BLOCCO | 0 | 0 | 1 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | 1 | - | - | - | 0 | 0 | BLOCCO | 0 | 0 | 1 | 0 | 0 | 0 |
| CIFRA\_3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | CODICE\_OK | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_2\_ERRORE | 1 | - | - | - | - | 0 | 0 | CIFRA\_3\_ERRORE | 0 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3\_ERRORE | 1 | - | - | - | - | - | 0 | CIFRA\_1 | 1 | 0 | 0 | 0 | 0 | 0 |
| CIFRA\_3\_ERRORE | 1 | - | - | - | - | - | 1 | BLOCCO | 0 | 0 | 1 | 0 | 0 | 0 |
| CODICE\_OK | 1 | - | - | - | - | 0 | - | RILASCIA\_BANCOMAT | 0 | 0 | 0 | 1 | 0 | 0 |
| CODICE\_OK | 1 | - | - | - | - | 1 | - | RILASCIA\_BANCOMAT | 0 | 1 | 0 | 1 | 0 | 0 |
| RILASCIA\_BANCOMAT | 1 | - | - | - | - | - | - | RILASCIA\_BANCOMAT | 0 | 0 | 0 | 0 | 0 | 0 |
| RILASCIA\_BANCOMAT | 0 | - | - | - | - | - | - | NO\_BANCOMAT | 0 | 0 | 0 | 0 | 0 | 1 |
| BLOCCO | - | - | - | - | - | - | - | NO\_BANCOMAT | 0 | 0 | 0 | 0 | 0 | 1 |

# Datapath

**Datapath: Contatentativi**

Il Datapath Contatentativi, è un componente aggiuntivo che abbiamo deciso di aggiungere, per rendere possibile una funzionalità necessaria per il completamento e il funzionamento della macchina.

Questo Datapath viene utilizzato per contare ciclicamente quante volte viene sbagliato il codice d’ingresso, tramite un registro. Quando il registro avrà assunto il valore 3 e il codice viene nuovamente sbagliato, significa che è stato sbagliato per la terza volta. La FSMD manda quindi in blocco il bancomat e non lo rilascia.

In seguito saranno elencati i componenti utilizzati per la creazione di questa sezione di datapath con le relative specifiche di ogni componente:

* **Multiplexer a 2 Bit:** Il multiplexer a due bit d’ingresso viene utilizzato concettualmente per passare una costante al sommatore. Il bit di Check\_Fine\_Tentativi entra come input di selezione nel multiplexer, il quale lo riceve e se quest’ultimo è a 1 manda in output la costante 01, se invece il valore di Check\_Fine\_Tentativi è a 0 manderà in output la costante 00.

Per semplificazione pratica, questo MUX è stato sostituito mettendo in AND il segnale Check\_Fine\_Tentativi e la cifra meno significativa della costante da mandare al sommatore.

* **Sommatore a 2 Bit:** La funzionalità del sommatore è di prendere in ingresso l’output del **multiplexer** e sommarlo al valore interno del Registro, una volta fatto questo l’output diventerà l’input del **registro**, quindi il suo nuovo valore.
* **Multiplexer a 2 Bit (Reset):** questo multiplexer si intrapone tra il sommatore e il registro. Se il segnale di selezione di questo MUX, ovvero il segnale di Reset vale 0, allora l’output del sommatore viene lasciato entrare nel Registro, altrimenti viene inserito 00 nel Registro, resettandolo.
* **Registro a 2 Bit:** Il registro viene utile per memorizzare un valore preso in ingresso, in questo caso dal **sommatore**. Viene poi utilizzato sia in un passaggio successivo di comparazione, sia come ingresso al **sommatore**.
* **Comparatore a 2 bit:** Il comparatore è l’ultimo componente del nostro Datapath contatentativi. Il suo scopo è di prendere in ingresso il valore a due bit del **Registro** e compararlo con il valore binario 11. Nel caso in cui i due valori coincidessero, viene abilitato il segnale Fine\_Tentativi in ingresso alla FSM, il quale poi persiste finché il Registro non viene resettato. In caso contrario, invece, lascia a 0 il bit di Fine\_Tentativi.

**Datapath: Check\_Cash**

Questa sezione del datapath è quella che si occupa di verificare che il quantitativo di denaro richiesto dall’utente (Cash\_R) sia inferiore di ¼ del denaro presente nella cassaforte del Bancomat (Cash\_D).

La verifica da effettuare deve rispettare la seguente disequazione:

→

L’importo di Cash\_R viene moltiplicato per 4 utilizzando uno *Shifter Register* a 12 bit che esegue lo spostamento dei bit verso sinistra di due posizioni. L’input a questo shifter è di 10 bit e l’output è di 12, in quanto per quadruplicare Cash\_R si spostano verso sinistra di due posizioni tutti i suoi bit, aggiungendo due zeri a destra.

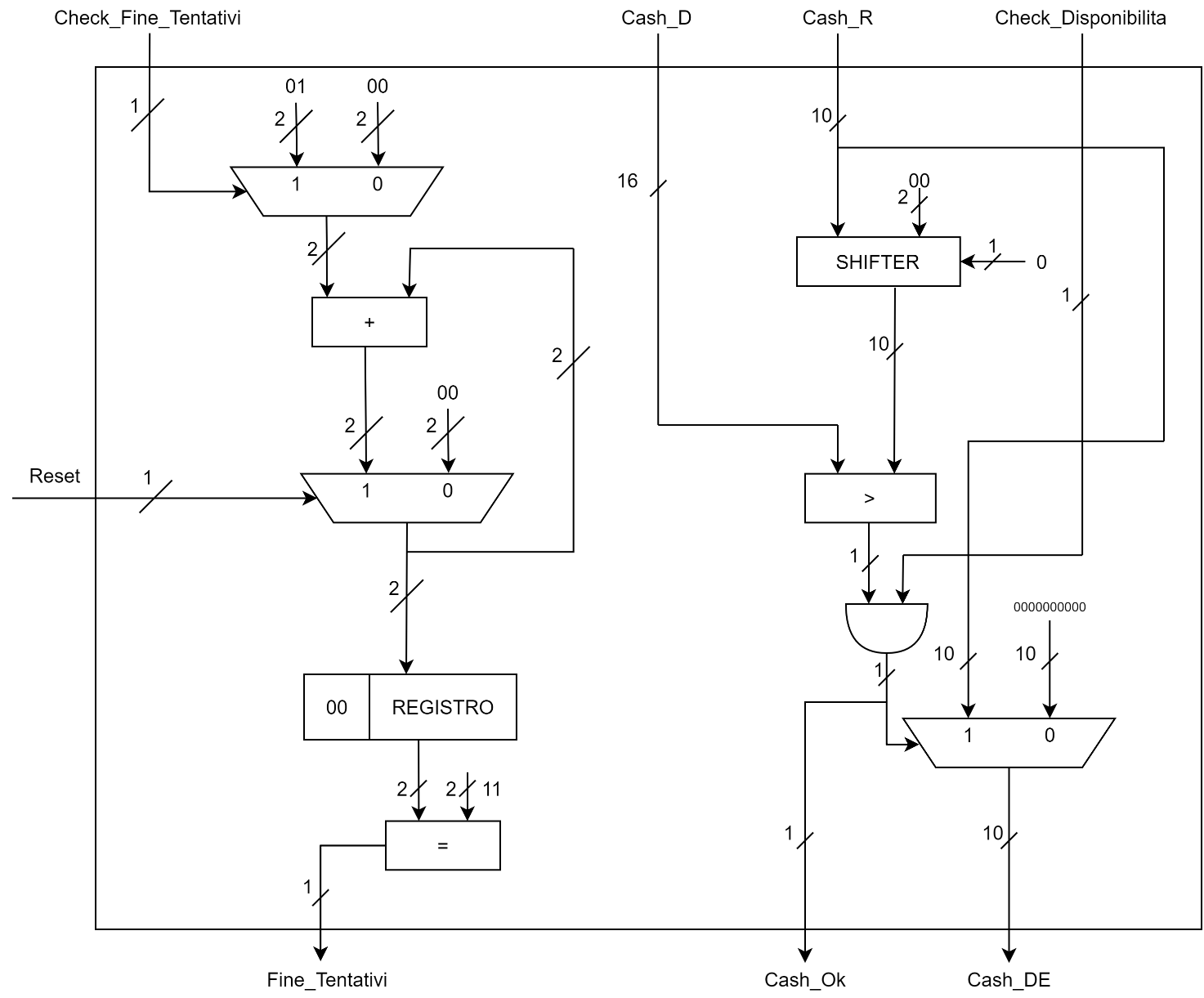
Esempio: Cash\_R: 01101101012 (= 43710) → Cash\_R (shift): 0110110101002 (= 174810)

L’importo che esce dallo Shifter viene quindi confrontato con Cash\_D utilizzando un comparatore di maggioranza. Quindi se il comparatore mette 1 in output. Infine se l’AND tra l’uscita del comparatore e l’input Check\_Disponibilità risulta vera, Cash\_Ok viene messo a 1 e su Cash\_DE (cash da erogare) viene riportato il cash richiesto dall’utente.

Componenti utilizzati:

* **Shifter a SX di 2 Bit**: Il componente utilizza 12 registri. L’input da 10 bit viene trascritto sui primi 10 registri “a sinistra”, ovvero i più significativi, moltiplicando così il valore per 4. I due bit restanti, i meno significativi, vengono riportati a 0. L’entrata di 10 corrisponderà quindi a un’uscita a 12 bit.
* **Comparatore di maggioranza**: Il componente prende in ingresso due valori a 16 bit (Cash\_R di 12 bit, viene portato a 16 bit aggiungendo 4 zeri a sinistra). Il comparatore sfrutta l’operatore xor per verificare se il primo valore inserito (Cash\_D) è maggiore del secondo (Cash\_R\*4). Restituisce 1 in uscita se la condizione è verificata.
* **Porta AND**: Viene sfruttata per determinare se abilitare l’uscita Cash\_Ok. In ingresso alla porta ci sono il segnale d’uscita del comparatore e il segnale Check\_Disponibilita. Cash\_Ok viene quindi messo a 1 solo se la FSM chiede di fare il controllo (mediante il segnale Check\_Disponibilita) e se i valori di Cash\_R e Cash\_D rispettano la condizione richiesta.
* **Multiplexer**: viene usato da un punto di vista schematico per far in modo che il segnale Cash\_DE riporti il cash da erogare solamente quando Cash\_Ok vale 1, altrimenti riporta tutti zeri. Per una questione di semplicità, è stato scelto di sostituirlo nel relativo file *.blif* con una serie di controlli AND tra Cash\_Ok e ogni singola cifra di Cash\_R.

**Rappresentazione Grafica Datapath:**

****

# Ottimizzazione, Mapping e Statistiche

In seguito a una minimizzazione per area, effettuata mediante la libreria synch.genlib, con i comandi:

* source script.rugged Per minimizzare
* map -m 0 Per lanciare l’ottimizzazione
* map -s Per visualizzare e valutare i risultati ottenuti

Abbiamo ottenuto i seguenti parametri:

*Risultati del mapping*

sis> map -s

>>> before removing serial inverters <<<

# of outputs: 19

**total gate area: 2952.00**

**maximum arrival time: (27.40,27.40)**

maximum po slack: (-5.20,-5.20)

minimum po slack: (-27.40,-27.40)

total neg slack: (-369.40,-369.40)

# of failing outputs: 19

>>> before removing parallel inverters <<<

# of outputs: 19

**total gate area: 2952.00**

**maximum arrival time: (27.40,27.40)**

maximum po slack: (-5.20,-5.20)

minimum po slack: (-27.40,-27.40)

total neg slack: (-369.40,-369.40)

# of failing outputs: 19

# of outputs: 19

**total gate area: 2952.00**

**maximum arrival time: (27.40,27.40)**

maximum po slack: (-5.20,-5.20)

minimum po slack: (-27.40,-27.40)

total neg slack: (-369.40,-369.40)

# of failing outputs: 19

*Statistiche prima dell’ottimizzazione*

sis>print\_stats

FSMD pi=31 po=13 nodes= 60 latches= 6

lits(sop)= 336

*Statistiche dopo l’ottimizzazione*

sis>print\_stats

FSMD pi=31 po=13 nodes= 33 latches= 6

lits(sop)= 228