

ELABORATO SIS

ARCHITETTURA DEGLI ELABORATORI

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

Bianchini Davide (matricola)

Bragastini Enrico (matricola)

Mafficini Andrea (VR462441)

# Sommario

[Sommario](#_heading=h.gjdgxs) **2**

[FSMD](#_heading=h.30j0zll) **3**

[Controllore FSM](#_heading=h.1fob9te) **4**

[Datapath Contatentativi](#_heading=h.3znysh7) **7**

[Datapath Check\_Cash](#_heading=h.2et92p0) **9**

[**Esempio esecutivo**](#_heading=h.bpe98oedxwr3) **11**

[**Ottimizzazione e Mapping**](#_heading=h.j1p67vcok6jy) **11**

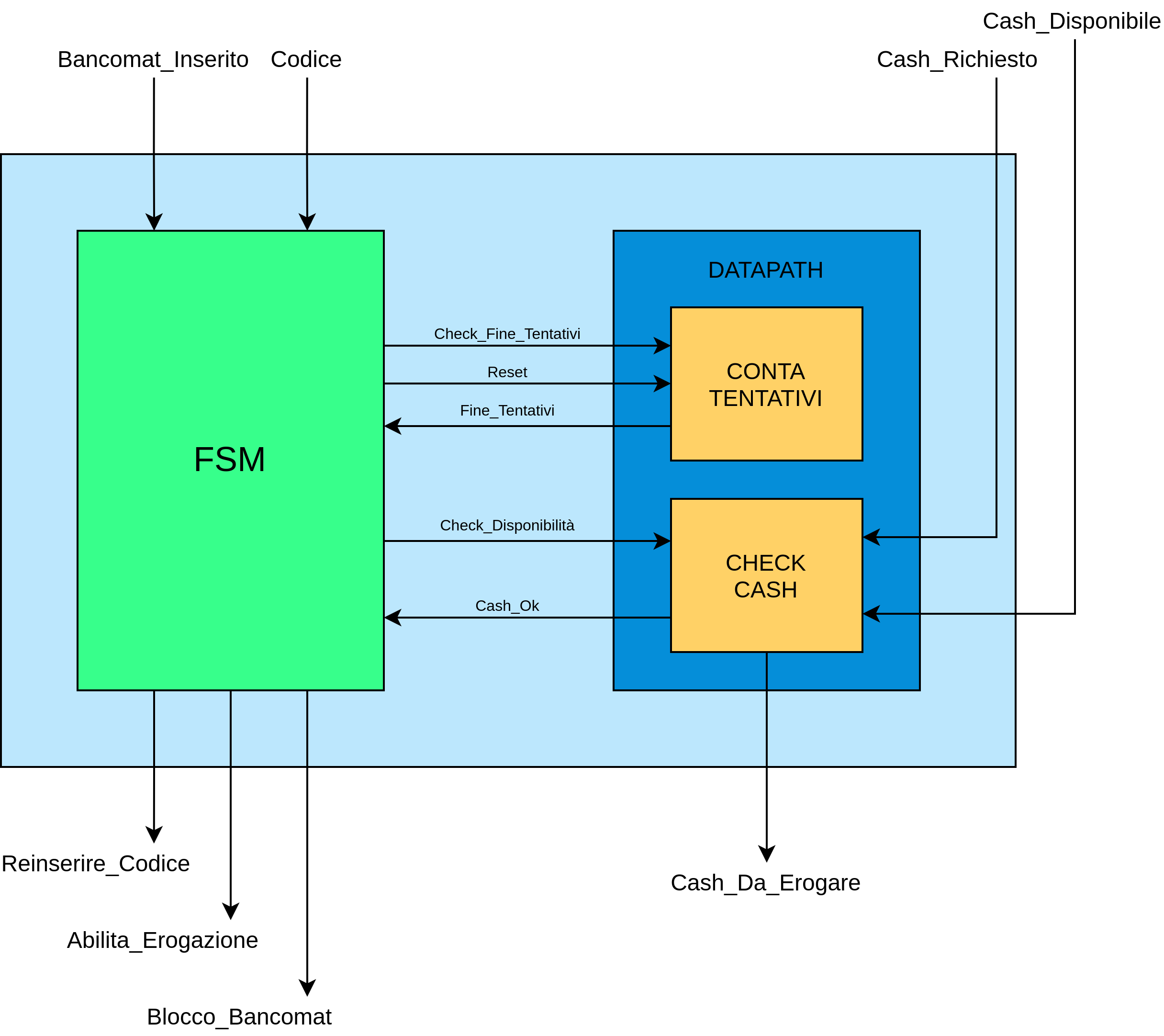
# FSMD

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**

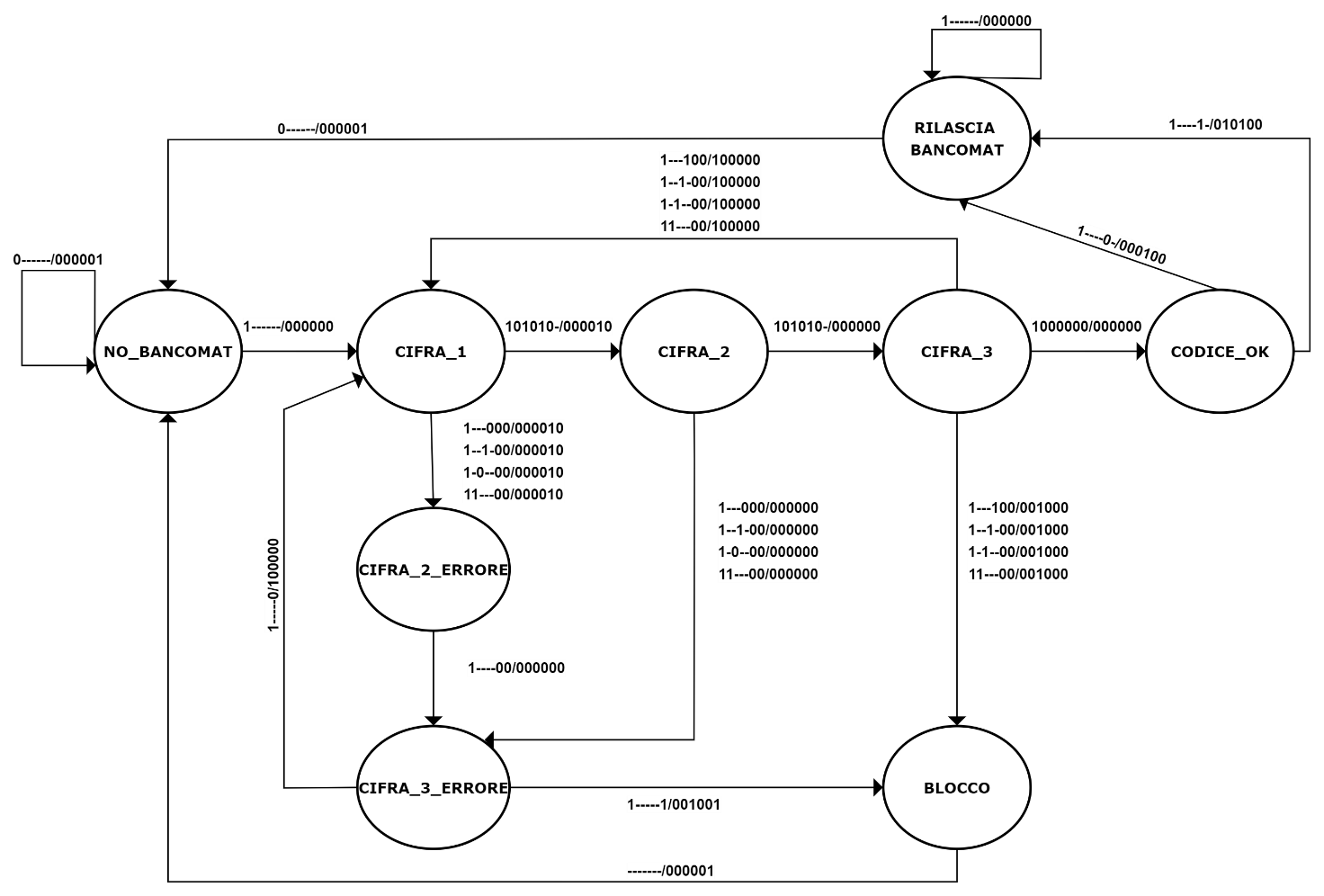
Il circuito richiesto, deve controllare l’erogazione di denaro di un bancomat.

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 cifra che si vuole prelevare. Il circuito, una volta controllato che ci siano contanti sufficienti per il prelievo, ovvero 4 volte la cifra richiesta, (uscita abilitazione\_erogazione a 1) erogherà i soldi. (uscita cash\_da\_erogare a 1).

Si hanno 3 tentativi per l’inserimento del pin (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 il denaro.

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

**3. Scelte progettuali**

Uno ad uno, analizziamo gli stati determinate da 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 viene inserito il numero 5 (0101) si passerà allo stato CIFRA\_2, per scelta progettuale si abilita a 1 l’uscita Check\_Fine\_Tentativi, in modo tale che il conteggio dei tentativi sia fatto nella maniera corretta (\*spiegare bene perché la penultima uscita è a 1\*).  
In tutti gli altri casi (input diverso da 0101), si passerà allo stato CIFRA\_2\_ERRORE.   
Per gli stessi motivi sopra citati l’uscita Check\_Fine\_Tentativi sarà settata a 1

Stato CIFRA\_2: se viene inserito il numero 5 (0101) si passerà allo stato CIFRA\_3 con le uscite settate a 0.   
In tutti gli altri casi (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.

Stato CIFRA\_3: se viene inserito il numero 0 (0000) si passerà allo stato successivo CODICE\_OK, uscite a 0.

Stato CIFRA\_3\_ERRORE: 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.  
Nel caso in cui sia il terzo tentativo, si abilita l’uscita blocco che porterà allo stato di blocco.

Stato BLOCCO: la macchina ritira dopo aver bloccato il bancomat e torna tornare allo stato NO\_BANCOMAT per ripartire con un nuovo prelievo.

Stato CODICE\_OK: stato in cui si inserirà la cifra da prelevare richiesta.   
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 il controllo sale a 1 l’uscita Check\_Disponibilita e l’ingresso Cash\_Ok).  
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\_: lo stato aspetta che venga disinserita la carta bancomat, in modo tale da settare a 0 il bit Bancomat\_Inserito e tornare allo stato NO\_BANCOMAT per ripartire con un nuovo prelievo.

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 Contatentativi

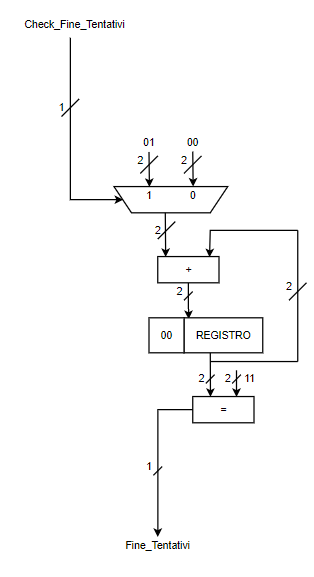
Il Datapath Contatentativi, è un componente aggiuntivo che abbiamo deciso di aggiungere per rendere possibile una funzionalità, necessaria per il completamento e il funzionamento della macchina.

Questo Datapath viene utilizzato per contare ciclicamente quante volte viene sbagliato il codice d’ingresso per l’attivazione del servizio di erogazione tramite un registro. Nel momento in cui il registro assumerà il valore 3, ovvero quando il codice è stato sbagliato per la terza volta, la FSMD manda in blocco il bancomat e non lo rilascia.

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

* **MULTIPLEXER A 2 BIT:** Il multiplexer a due bit d’ingresso viene utilizzato come costante da sommare ai bit del registro e prende come ingressi i bit 01 e 00. Il bit di Check\_Fine\_Tentativi va a 1 nel momento in cui il codice viene sbagliato e entra come input del multiplexer, che lo riceve e se è a 1 manda in output l’ingresso 01, se invece il valore di Check\_Fine\_Tentativi è a 0 manderà in output l’ingresso 00.
* **SOMMATORE A 2 BIT:** La funzionalità del sommatore è di prendere in ingresso l’output del **multiplexer** e sommarlo al valore interno al registro, una volta fatto questo l’output diventerà l’input del **registro** quindi il suo nuovo valore.
* **REGISTRO A 2 BIT:** Il registro viene utile per memorizzare un valore preso in ingresso, in questo caso dal **sommatore**, per poi riutilizzarlo in un passaggio successivo di comparazione e per passarlo al **sommatore** come valore per essere incrementato oppure rimanere tale.
* **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, abiliterebbe a 1 il bit della FSM Fine\_Tentativi che porterebbe il bancomat nello stato di blocco; in caso contrario, invece, lascerebbe a 0 il bit di Fine\_Tentativi e porterebbe la FSM allo stato di inserimento della prima cifra del codice.

**Rappresentazione grafica del datapath Contatentativi**



# 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.

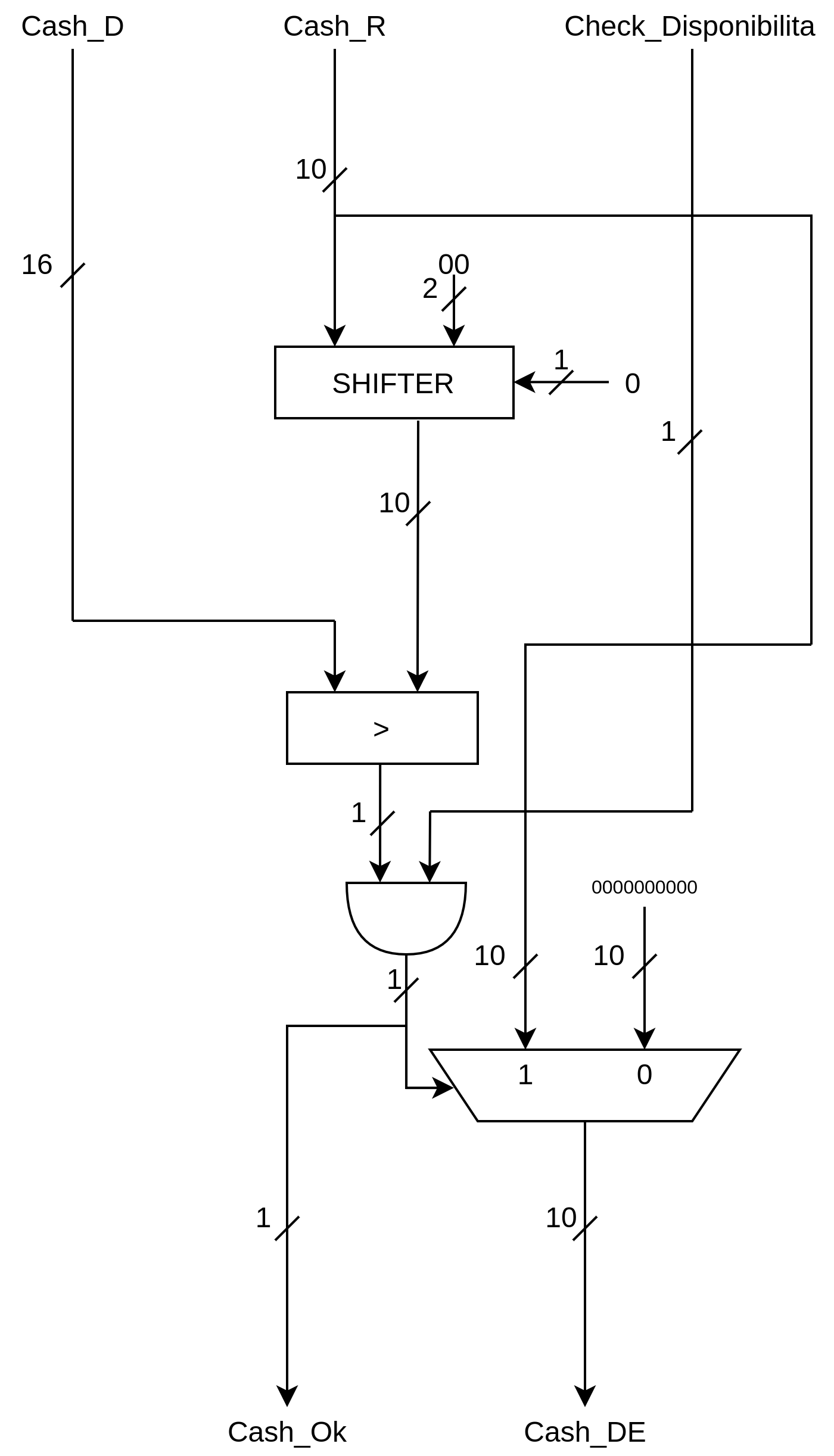
Esempio: Cash\_R: 01101101012 (= 43710) → Cash\_R (shift): 0110110101002 (= 174810)

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. Quindi se l’AND tra l’uscita del comparatore e l’imput 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 SX di 2bit**: 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
* **Comparatore di maggioranza**: Il componente prende in ingresso due valori a 16 bit (Cash\_R che ora vale 12 bit, viene portato a 16 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 mettere a 1 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 se la FSM chiede effettivamente di fare il controllo (mediante il segnale Check\_Disponibilita) e se i valori di Cash\_R e Cash\_D rispettano la condizione richiesta.
* **Multiplexer**: Il MUX 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 del datapath Check\_Cash**



# 

# 

# 

# 

# 

# Esempio esecutivo

# 

# 

# 

# 

# Ottimizzazione e Mapping

* statistiche del circuito prima e dopo l’ottimizzazione
* specifiche mapping