UNIVERSITÀ
di VERONA

Dipartimento di Informatica

Elaborato SIS e Verilog
Laboratorio di Architettura degli Elaboratori

Progettazione dispositivo per partite di morra cinese

A.A. 2023/2024

Lavoro di gruppo:

Antongiulio Donno (VR502818)
Samy El Ansari (VR501144)

INDICE

SPECIFICHE

Il dispositivo in questione permette di gestire partite di morra cinese. I due giocatori (indicati come PRIMO e SECONDO) inseriscono la mossa, la quale può essere sasso, carta o forbice. Una volta eseguita questa scelta , il vincitore della manche verrà decretato in base a queste regole:

- Sasso batte forbici;
- Forbici batte carta;
- Carta batte sasso.

Nel caso i due giocatori inserissero la stessa mossa, la manche finirebbe in pareggio.

Inoltre, ci sono altre regole da ricordare durante la partita e durante le manche:

- Si devono effettuare un minimo di 4 manche affinché un giocatore possa vincere;
- Il numero massimo di manche possibili è 19. Esso viene settato al ciclo di clock in cui viene iniziata la partita;
- Il giocatore vince quando riesce a vincere due manche in più del suo avversario a patto che abbia giocato già almeno quattro manche;
- Ad ogni manche ,il vincitore della manche precedente non può ripetere l'ultima mossa effettuata. Se dovesse farlo, la manche risulterebbe non valida e non verrebbe conteggiata;
- Se la manche dovesse finire in pareggio, quest'ultima viene conteggiata . I giocatori, alla manche successiva, possono usare tutte le mosse.

Il circuito presenta tre ingressi principali:

- **PRIMO [2 bit]** : mossa selezionata dal primo giocatore. Le mosse presentano i seguenti codici:
    - 00 : nessuna mossa;
    - 01 : Sasso;
    - 10 : Carta;
    - 11 : Forbice.

- **SECONDO [2 bit]** : mossa scelta dal secondo giocatore. I codici delle mosse sono gli stessi di quelli del primo giocatore.

- **INIZIA [1 bit]** : quando questo ingresso vale 1, riporta il sistema alla configurazione iniziale. In aggiunta, la concatenazione degli ingressi di PRIMO e SECONDO viene usata per indicare il numero massimo di manche oltre le quattro obbligatorie. Quindi, al numero ottenuto dalla concatenazione dei due ingressi bisogna aggiungere quattro per trovare il vero numero di partite totali: ad esempio se il valore di PRIMO=01 e quello di SECONDO=01, il numero di manche totali è 9 (5 ottenute dalla

concatenazione dagli ingressi dei due giocatori più le 4 partite obbligatorie). Quando vale 0, la manche prosegue normalmente.

Il circuito presenta 2 uscite:

- **MANCHE [2 bit]**: fornisce il risultato dell'ultima manche giocata con la seguente codifica:
    - 00 : manche non valida;
    - 01 : manche vinta dal giocatore 1;
    - 10 : manche vinta dal giocatore 2;
    - 11 : manche pareggiata.

- **PARTITA [2 bit]**: fornisce il risultato della partita con la seguente codifica:
    - 00 : la partita non è terminata;
    - 01 : la partita è terminata, ed ha vinto il giocatore 1;
    - 10 : la partita è terminata, ed ha vinto il giocatore 2;
    - 11 : la partita è terminata in pareggio.

ARCHITETTURA GENERALE DEL CIRCUITO

Il circuito presenta una FSM (controllore) e DATAPATH (elaboratore) che, comunicando tra loro grazie ai segnali di controllo e di stato, in base ai valori di ingresso di PRIMO, SECONDO ed INIZIA danno come uscita due valori a due bit, cioè MANCHE e PARTITA.
Il circuito si avvia dando come input sia all' FSM sia al Datapath gli ingressi PRIMO, SECONDO e INIZIA. Per quanto riguarda il Datapath , i tre input entrano all'interno del circuito e, a seconda della situazione di gioco in cui ci troviamo, ricaviamo tre valori (NP, PMAX, VALIDA) che saranno usati successivamente come segnali di stato, dunque come input nell' FSM oltre ai 3 ingressi di default. Una volta ottenuti tutti gli input per far partire la macchina a stati finiti, a seconda della codifica ottenuta e in base allo stato in cui ci troviamo, andremo allo stato successivo e otterremo 3 output: PM che diventerà un segnale di controllo, dunque un input del datapath, PARTITA che è uno degli output del circuito e infine MANCHE che fa sia da segnale di controllo sia da output del circuito.

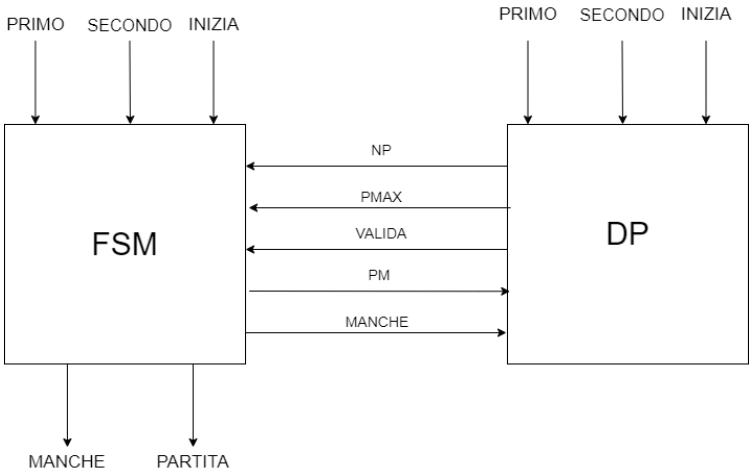


DIAGRAMMA DEGLI STATI DEL CONTROLLORE

Il controllore (FSM) è una macchina a stati finiti del tipo Mealy che presenta 8 segnali in ingresso (PRIMO1,PRIMO0,SECONDO1,SECONDO0,INIZIA,NP,PMAX,VALIDA) e 5 in uscita (MANCHE1,MANCHE0,PARTITA1,PARTITA0,PM).

INPUT

- PRIMO1
- PRIMO0
- SECONDO1
- SECONDO0
- INIZIA
- NP
- PMAX
- VALIDA

OUTPUT

- MANCHE1
- MANCHE0
- PARTITA1
- PARTITA0
- PM

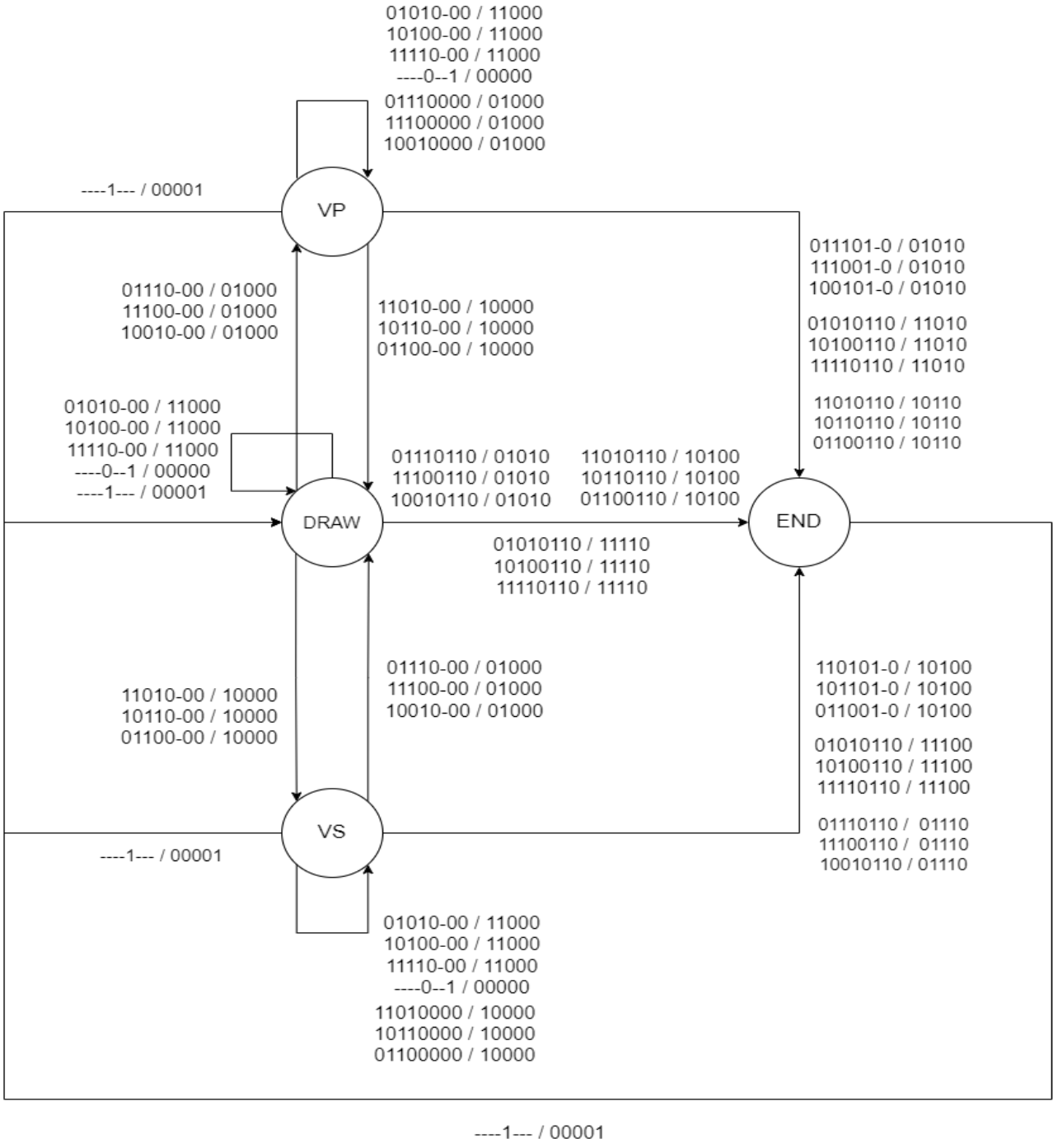
In questa FSM sono presenti 4 stati:

- **DRAW** : è lo stato di reset e indica che la partita tra i due giocatori è in una situazione di pareggio;

- **VP** : si arriva in questo stato quando il primo giocatore presenta un vantaggio di un uno nel conteggio delle manche vinte. Se lo scarto dei giocatori dovesse essere maggiore di uno ma il numero di manche giocate è minore di 4, cioè il numero minimo di manche da giocare, si rimarrà sempre in questo stato;

- **VS** : si arriva in questo stato quando il secondo giocatore presenta un vantaggio di un uno nel conteggio delle manche vinte. Se lo scarto dei giocatori dovesse essere maggiore di uno ma il numero di manche giocate è minore di 4, cioè il numero minimo di manche da giocare, si rimarrà sempre in questo stato;

- **END** : stato finale che preannuncia la fine della partita. Una volta finita, si ritornerà allo stato di reset e quindi ne inizierà una nuova.


01010-00 / 11000
10100-00 / 11000
11110-00 / 11000
----0--1 / 00000
01110000 / 01000
11100000 / 01000
10010000 / 01000
----1--- / 00001
VP
011101-0 / 01010
111001-0 / 01010
100101-0 / 01010
01010110 / 11010
10100110 / 11010
11110110 / 11010
11010110 / 10110
10110110 / 10110
01100110 / 10110
01110-00 / 01000
11100-00 / 01000
10010-00 / 01000
11010-00 / 10000
10110-00 / 10000
01100-00 / 10000
01010-00 / 11000
10100-00 / 11000
11110-00 / 11000
----0--1 / 00000
----1--- / 00001
DRAW
01110110 / 01010
11100110 / 01010
10010110 / 01010
11010110 / 10100
10110110 / 10100
01100110 / 10100
END
01010110 / 11110
10100110 / 11110
11110110 / 11110
11010-00 / 10000
10110-00 / 10000
01100-00 / 10000
01110-00 / 01000
11100-00 / 01000
10010-00 / 01000
110101-0 / 10100
101101-0 / 10100
011001-0 / 10100
01010110 / 11100
10100110 / 11100
11110110 / 11100
01110110 / 01110
11100110 / 01110
10010110 / 01110
VS
----1--- / 00001
01010-00 / 11000
10100-00 / 11000
11110-00 / 11000
----0--1 / 00000
11010000 / 10000
10110000 / 10000
01100000 / 10000
----1--- / 00001

ARCHITETTURA DEL DATAPATH

Il datapath è composto da 8 input (PRIMO1,PRIMO0,SECONDO1,SECONDO0,INIZIA, PM,MANCHE1,MANCHE0) e 3 output (VALIDA, NP, PMAX).
I 3 output diventeranno successivamente segnali di stato per l' FSM e vanno ad indicare queste condizioni:

- **VALIDA** : se vale 1 vuol dire che la manche non è valida (quando almeno una delle due mosse vale 00 oppure quando il giocatore usa la mossa vincente della manche precedente); invece se vale 0 la manche è valida;

- **NP** : se vale 1 indica che il contatore delle manche è uguale o superiore al valore 4, cioè il numero minimo di manche da giocare per ogni partita, e quindi qualsiasi giocatore che presenta uno scarto di due rispetto al suo avversario può vincere; invece se vale 0 nessuno dei due giocatore può vincere anche se uno di essi presenta uno scarto di due manche vinte;

- **PMAX** : se vale 1 indica che il registro che conta le manche giocate è uguale al numero massimo di manche massime da giocare (ottenuto con la concatenazione di PRIMO e SECONDO), di conseguenza sancisce la fine della partita; invece se vale 0 indica che le manche giocate sono inferiori alle manche massime e la partita può continuare se non ci sono altri presupposti affinché uno dei due giocatori possa vincere.

Per quanto riguarda il datapath, per spiegare il suo funzionamento al meglio, lo si può dividere in 4 sottoparti:
- la prima si occupa della concatenazione delle mosse per comprendere quante sono le manche massime per quella determinata partita;
- la seconda controlla se gli input di PRIMO e SECONDO rendono la manche valida oppure no;
- la terza gestisce i registri in cui vengono salvate le mosse vincenti di ogni manche ;
- la quarta si occupa di contare le manche avvenute, salvandole in un registro, e di verificare se il numero di manche giocate è maggiore di tre e se è uguale al numero di partite massime.

La prima parte si verifica solamente quando INIZIA è uguale a 1. Infatti PRIMO e SECONDO entrano in demultiplexer regolato dal segnale INIZIA e, quando vale 1, le mosse dei due giocatori diventano dei segnali di controllo per due multiplexer, uno per ogni giocatore:
- Nel multiplexer regolato da PRIMO, i quattro ingressi assumono dei valori ben definiti: all'ingresso 00 viene assegnato il valore 00000 (0 a 5 bit), all'ingresso 01 viene assegnato il valore 00100 (4 a 5 bit), all'ingresso 10 viene assegnato il valore 01000 (8 a 5 bit) e all'ingresso 11 viene assegnato il valore 01100 (12 a 5 bit);

- Nel multiplexer regolato da SECONDO, i quattro ingressi assumono dei valori definiti: all'ingresso 00 viene assegnato il valore 00000 (0 a 5 bit), all'ingresso 01 viene assegnato il valore 00001 (1 a 5 bit), all'ingresso 10 viene assegnato il valore 00010 (2 a 5 bit) e all'ingresso 11 viene assegnato il valore 00011 (3 a 5 bit);

Successivamente gli output dei 2 multiplexer,tramite un sommatore, si sommano tra di loro e l'uscita di questo componente entra in un altro sommatore che aggiunge il valore 4, cioè quello delle manche minime da giocare. Il risultato che si ottiene da questi due sommatori viene introdotto e salvato all'interno del registro PARTITE che indica quante manche massime si possono fare in quella singola partita. Infine è presente un multiplexer che consente di mantenere il valore dentro quel registro e un altro multiplexer per resettarlo.

La seconda parte si articola in due controlli i quali arriveranno alla fine in un componente OR che ci dirà effettivamente se le mosse inserite da PRIMO e SECONDO rendono la manche valida oppure no:

- il primo controllo verifica se le due mosse sono uguali a 00. Infatti se un giocatore non inserisce nessuna mossa (indicata con la codifica 00), la manche non è valida e non viene conteggiata. Per verificare che nessun giocatore abbia eseguito una mossa nulla c'è un comparatore per ogni giocatore che confronta la propria mossa con il valore 00: se la mossa corrisponde al valore comparato allora il comparatore da come output il valore 1, mentre se i due valori non combaciano il valore in output è 0. Infine i risultati dei due comparatori vengono confrontati all'interno del componente OR: se almeno uno dei due valori è 1 allora l'output del componente vale 1, altrimenti 0;
- il secondo controllo verifica se la mossa di ogni singolo giocatore è uguale alla mossa vincente della manche precedente. Per ogni mossa sono presenti un demultiplexer e un multiplexer, entrambi regolati da PM. Quando PM vale 1 la mossa attuale viene confrontata tramite un comparatore con un valore nullo,perché all'altro ingresso del componente non è stato inserito nessun valore, invece quando PM vale 0 la mossa attuale viene confrontata tramite il comparatore con il valore presente nel registro della mossa vincente. Poi il risultato dei comparatori dei due giocatori (1 quando sono uguali,0 quando sono diversi) entrano nel componente OR dove, se almeno uno dei valori dei comparatori vale 1, il valore di uscita dell'ultimo componente vale 1, altrimenti vale 0.

La terza parte, che si occupa di salvare in un registro la mossa del vincitore della manche, presenta 3 componenti:il registro per salvare la mossa (UP se registra quella di PRIMO, US se registra quella di SECONDO), un multiplexer a 4 entrate che ha MANCHE come segnale, un multiplexer , che ha INIZIA come segnale, che permette di resettare il registro.
Il multiplexer,che presenta il segnale MANCHE, ha quattro entrate in cui la seconda (indicata dal valore 01) riceve il valore della mossa di PRIMO, mentre le altre tre ricevono il valore 00. L'output del multiplexer, il cui valore viene dall'entrata corrispondente al valore di MANCHE, diventa l'input di un altro multiplexer che viene regolato da INIZIA: se quest'ultimo vale 1 allora il registro riceve in entrata il valore 00, se invece vale 0 il registro

riceve in input il valore derivante dal multiplexer gestito dal segnale MANCHE. Infine l'uscita del registro viene collegata con un multiplexer che, quando il segnale PM vale 0, porta al comparatore il valore del registro dove verrà confrontato con la mossa della manche successiva. Questo procedimento indica il salvataggio della mossa quando PRIMO vince la manche. Lo stesso meccanismo vale quando vince SECONDO solo che, nel multiplexer regolato dal segnale MANCHE, l'entrata che riceve la mossa del secondo giocatore è la terza (indicata con il valore 10).

Infine la quarta parte presenta un multiplexer regolato dal segnale VALIDA(ottenuto dalla seconda parte). L'output di questo componente è 1 quando la manche è valida (quando il segnale VALIDA vale 0) oppure zero quando la manche non è valida (quando il segnale VALIDA vale 1) e va a sommarsi tramite un sommatore al valore del registro che salva quante manche sono state giocate(CONT PARTITE). L'uscita del sommatore è collegata a 3 diversi componenti:

- ad un multiplexer regolato dal segnale INIZIA che mi permette di resettare il registro una volta che la partita ricomincia;
- ad un comparatore che controlla se il numero di manche giocate è maggiore di tre e dà in output il valore di NP;
- ad un altro comparatore che confronta se il numero di partite giocate è uguale al numero di partite massime ottenuto dalla concatenazione delle mosse dei due giocatori e dà in uscita il valore di PMAX.

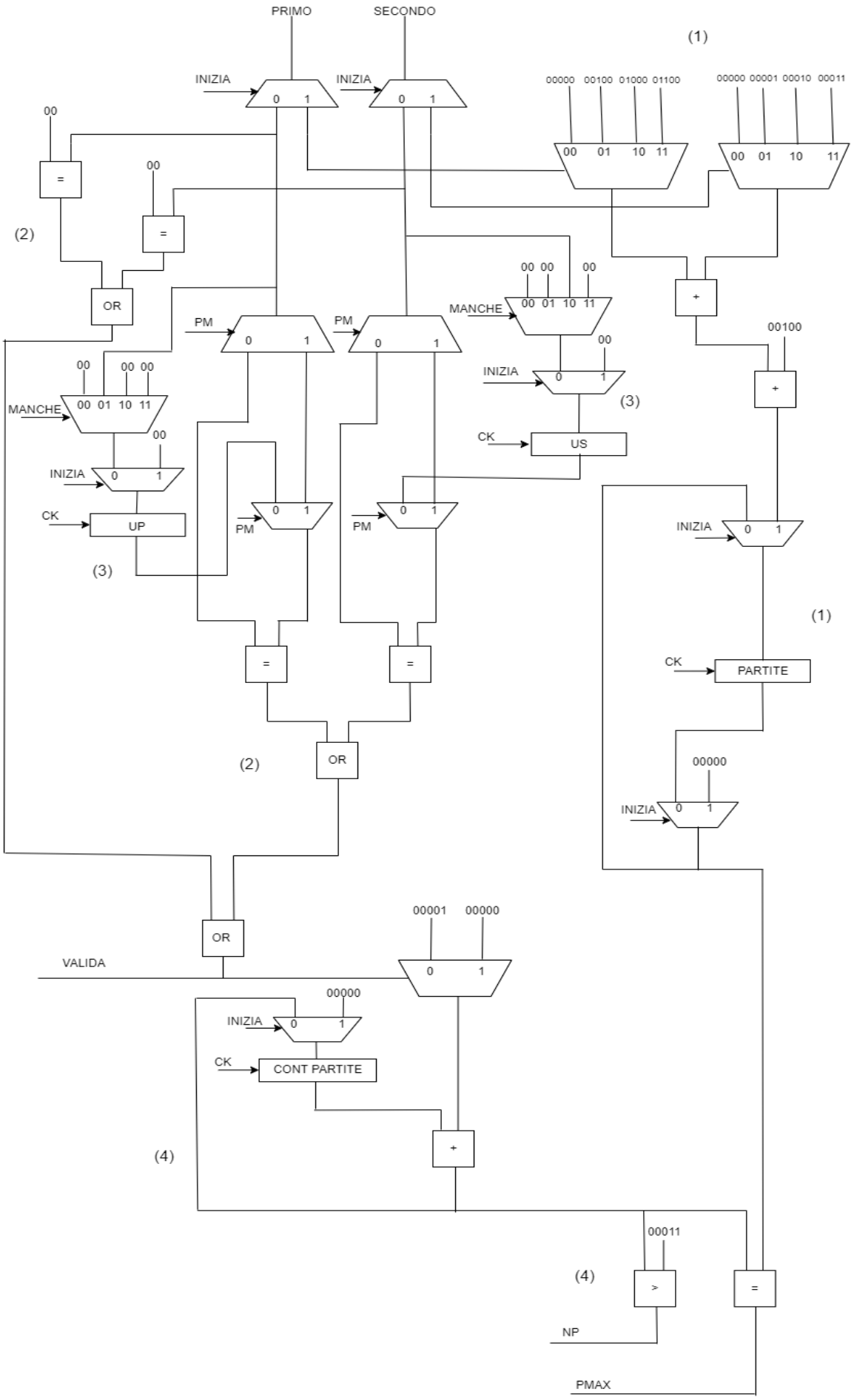
All'interno di questo datapath sono presenti 4 registri, ciascuno regolato dal segnale di clock,utili a salvare 4 valori differenti:

- **UP** = salva la mossa vincente del primo giocatore;

- **US** = salva la mossa vincente del secondo giocatore;

- **PARTITE** = salva il numero massimo di manche per quella singola partita;

- **CONT PARTITE** = salva il numero di manche giocate durante la partita.


PRIMO
SECONDO
(1)
INIZIA
INIZIA
00000 00100 01000 01100
00000 00001 00010 00011
00 01 10 11
00 01 10 11
00
=
00
=
(2)
OR
PM
PM
00 00 00
MANCHE
00 01 10 11
00
INIZIA
0 1
CK
UP
(3)
PM
PM
=
=
(2)
OR
00 00 00
MANCHE
00 01 10 11
00
INIZIA
0 1
(3)
CK
US
+
00100
+
INIZIA
0 1
(1)
CK
PARTITE
00000
INIZIA
0 1
OR
VALIDA
00001 00000
0 1
00000
INIZIA
0 1
CK
CONT PARTITE
+
(4)
00011
(4)
>
=
NP
PMAX

STATISTICHE PRIMA E DOPO L'OTTIMIZZAZIONE

Andando ad analizzare la FSMD, si può notare come essa cambia prima e dopo l'ottimizzazione:

PRIMA DELL'OTTIMIZZAZIONE

```
sis> read_blif morra_cinese.blif
Warning: network `ELABORAZIONE', node "COUT0" does not fanout
Warning: network `ELABORAZIONE', node "COUT1" does not fanout
Warning: network `ELABORAZIONE', node "COUT2" does not fanout
Warning: network `MORRA_CINESE', node "COUT0" does not fanout
Warning: network `MORRA_CINESE', node "COUT1" does not fanout
Warning: network `MORRA_CINESE', node "COUT2" does not fanout
sis> print_stats
MORRA_CINESE    pi= 5   po= 4   nodes=179       latches=16
lits(sop)= 962
```

DOPO L' OTTIMIZZAZIONE

```
sis> read_blif FSMD.blif
sis> print_stats
MORRA_CINESE    pi= 5   po= 4   nodes= 47       latches=16
lits(sop)= 318
```

L'ottimizzazione è avvenuta utilizzando i comandi "***full_simlify***" e "***source script.rugged***" sul file morra_cinese.blif , cioè il file non ottimizzato della FSMD.

MAPPING

Nella fase di ottimizzazione e mappatura, abbiamo adoperato il comando "***read_library synch.genlib***", in modo da caricare la libreria preposta per l'esecuzione del comando. Successivamente si inserisce il comando "***map -m 0 -s***" che permette di ottimizzare il circuito rispetto all'area e di visualizzare le informazioni relative all'area e al ritardo dopo il mapping. Le due informazioni principali ottenute sono:

- **TOTAL GATE AREA**: valore dell'area come numero di celle standard della libreria tecnologica;
- **MAXIMUM ARRIVAL TIME**: indica il ritardo.

```
sis> read_library synch.genlib
sis> map -m 0 -s
warning: unknown latch type at node '{[32]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[33]}' (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:          20
total gate area:       4836.00
maximum arrival time: (43.40,43.40)
maximum po slack:     (-3.60,-3.60)
minimum po slack:     (-43.40,-43.40)
total neg slack:      (-452.60,-452.60)
# of failing outputs:  20
>>> before removing parallel inverters <<<
# of outputs:          20
total gate area:       4836.00
maximum arrival time: (43.40,43.40)
maximum po slack:     (-3.60,-3.60)
minimum po slack:     (-43.40,-43.40)
total neg slack:      (-452.60,-452.60)
# of failing outputs:  20
# of outputs:          20
total gate area:       4708.00
maximum arrival time: (43.00,43.00)
maximum po slack:     (-3.60,-3.60)
minimum po slack:     (-43.00,-43.00)
total neg slack:      (-441.80,-441.80)
# of failing outputs:  20
```

SCELTE PROGETTUALI

Durante la progettazione del circuito abbiamo deciso di dare la vittoria al giocatore che, al termine delle manche massime, ha ottenuto più vittorie, nonostante non abbia raggiunto lo scarto di due richiesto nelle regole. Inoltre abbiamo deciso che, se un giocatore inserisce una mossa non valida, come ad esempio 00 oppure la mossa vincente del turno precedente, la manche non viene contata e entrambi i giocatori possono usare tutte le mosse.