Laboratorio di Architettura degli Elaboratori

Elaborato SIS e Verilog

A.A. 2023/2024

Martini Anna (VR504166)
Zanotelli Riccardo (VR474635)

Testo dell’elaborato

Si progetti un dispositivo per la gestione di partite della morra cinese, conosciuta anche come sasso carta-forbici.
Due giocatori inseriscono una mossa, che può essere carta, sasso, o forbici. Ad ogni manche, il giocatore vincente è decretato dalle seguenti regole:
• Sasso batte forbici;
• Forbici batte carta;
• Carta batte sasso.
Nel caso in cui i due giocatori scelgano la stessa mossa, la manche finisce in pareggio. Per renderla più avvincente, ogni partita si articola di più manche, con le seguenti regole:
• Si devono giocare un minimo di quattro manche;
• Si possono giocare un massimo di diciannove manche. Il numero massimo di manche, viene settato al ciclo di clock in cui viene iniziata la partita;
• Vince il primo giocatore a riuscire a vincere due manche in più del proprio avversario, a patto di aver giocato almeno quattro manche;
• Ad ogni manche, il giocatore vincente della manche precedente non può ripetere l’ultima mossa utilizzata. Nel caso lo facesse, la manche non sarebbe valida ed andrebbe ripetuta (quindi, non conteggiata);
• Ad ogni manche, in caso di pareggio la manche viene conteggiata. Alla manche successiva, entrambi i giocatori possono usare tutte le mosse.
Il circuito ha tre ingressi:
• PRIMO [2 bit];
• SECONDO [2bit];
• INIZIA [1 bit];
Il circuito ha due uscite:
• MANCHE [2 bit];
• PARTITA [2 bit];

Architettura generale del circuito

Il nostro dispositivo è composto da un’ unità di controllo (FSM) e da un’unità di elaborazione (DATAPATH). L’interazione tra questi due componenti va a creare un modello FSMD.
Gli ingressi della FSMD sono:

- INIZIA: input a un bit, che specifica quando una partita inizia.
- PRIMO: input a due bit, che rappresenta la mossa del primo giocatore.
- SECONDO: input a due bit, che rappresenta la mossa del secondo giocatore.

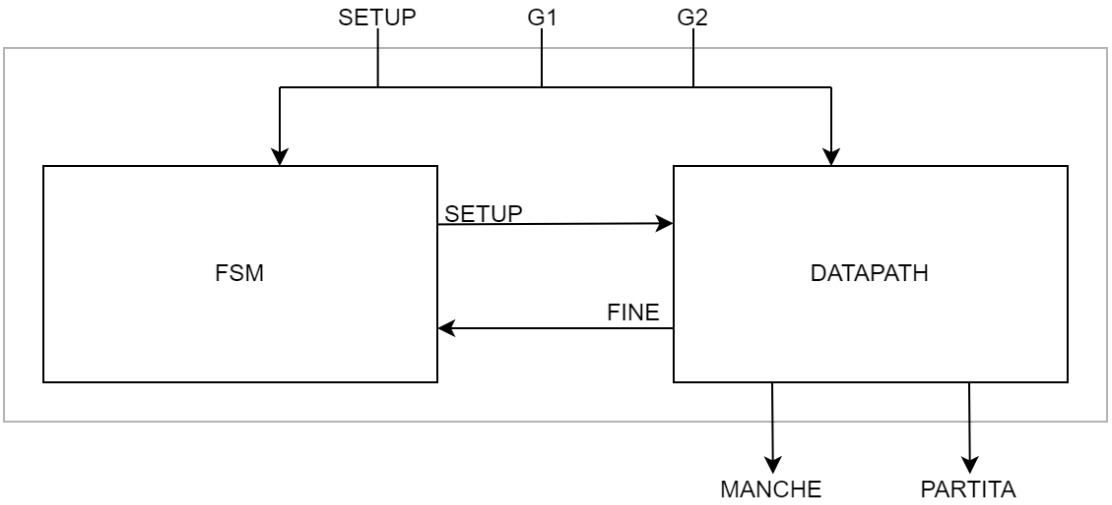
Qualora il bit INIZIA sia ad 1, i 4 bit di PRIMO e SECONDO vengono utilizzati per rappresentare un numero intero, che sommato alla costante 4, rappresenterà il numero dei turni da giocare.
Le uscite della FSMD sono:

- MANCHE: output a due bit, che rappresenta il risultato della giocata attuale.
- PARTITA: output a due bit, che rappresenta lo stato della partita, ossia se è finita o ancora in atto.

La FSM e il DATAPATH interagiscono tra di loro attraverso i seguenti segnali:
SEGNALI DI CONTROLLO:

- FINE: a un bit, assume valore 1 solo quando la partita corrente è terminata.

  SEGNALI DI STATO:
- SETUP: a un bit, indica quando resettare i registri per una nuova partita.

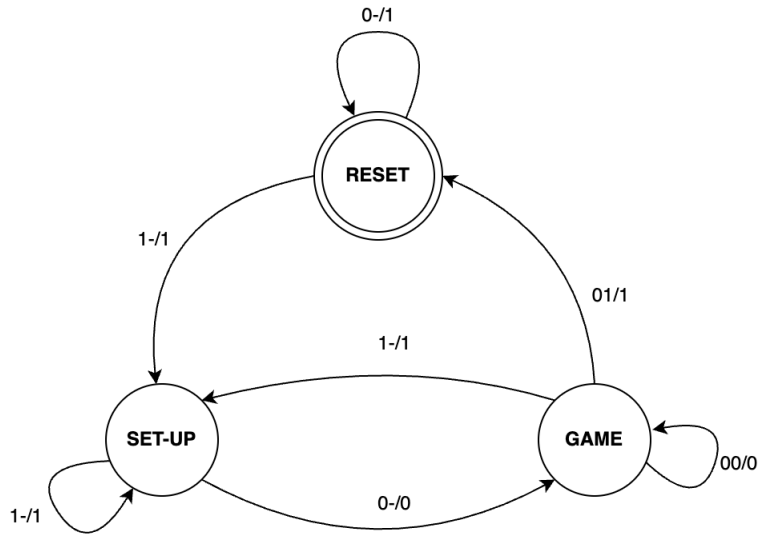


FSM

Il controllore è una FSM di Mealy con due bit d'ingresso (INIZIA e FINE) e un solo bit di output (SETUP).
Gli stati del controllore sono i seguenti:

- RESET: è lo stato iniziale della macchina, che attende l'input di INIZIA.
- SETUP: è lo stato in cui vengono preparati i registri per la partita corrente.
- GAME: è lo stato in cui la macchina rimane in attesa della fine della partita o di un reset dei registri.

Riportiamo in seguito il grafico di transizione degli stati (STG):



DATAPATH

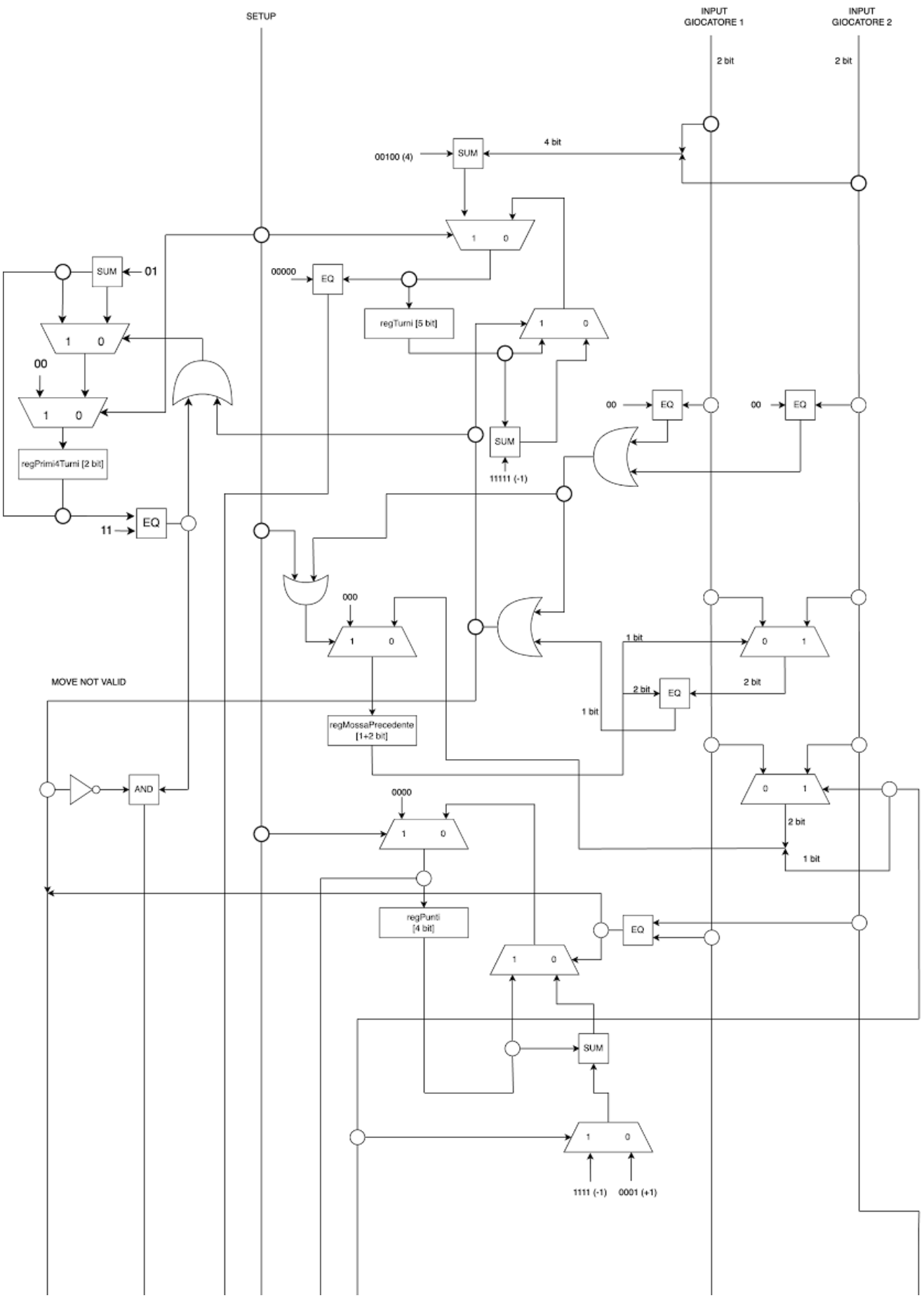
Il datapath riceve in input:

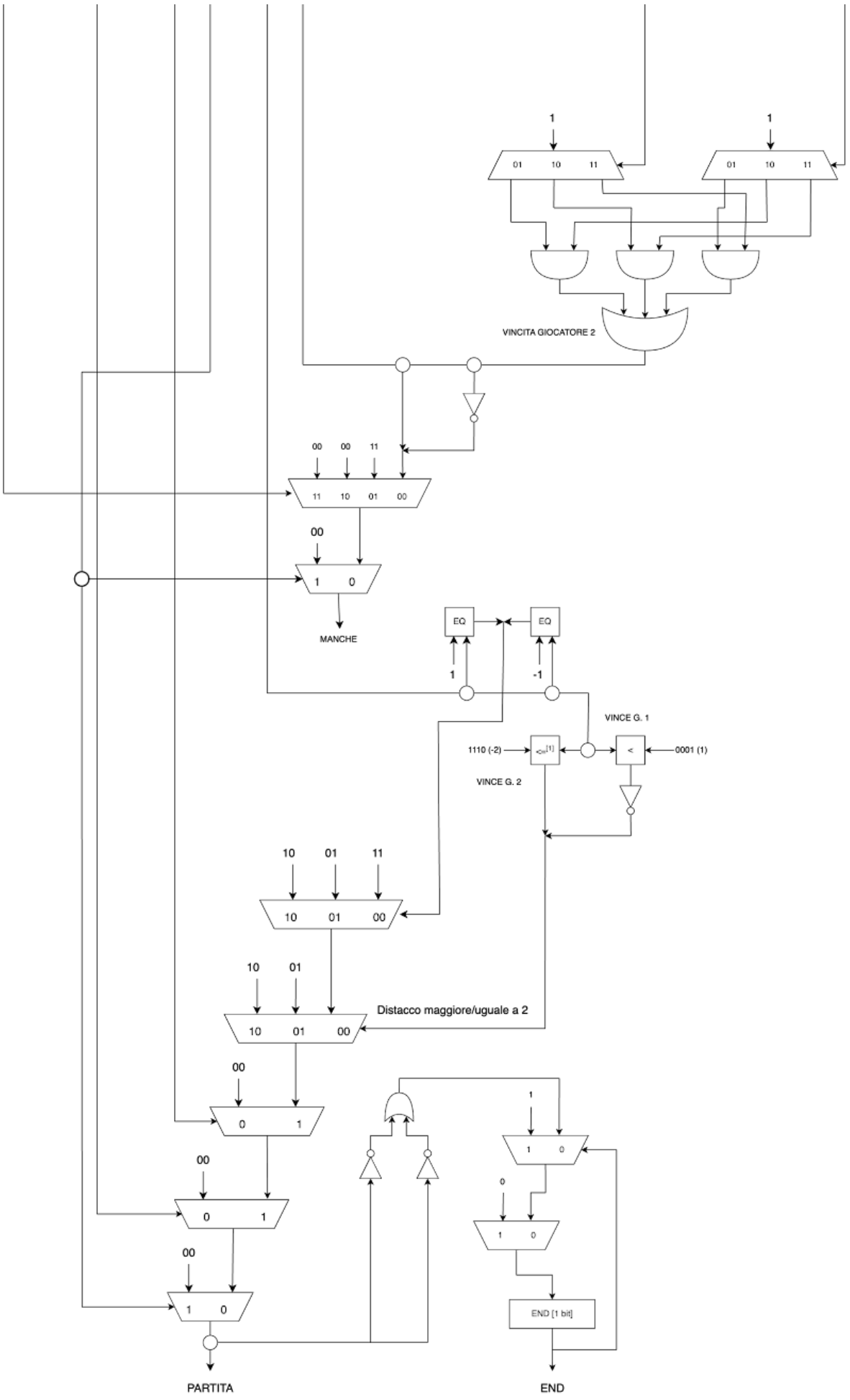
- SETUP: segnale di stato, che resetta i registri per una nuova partita.
- G1: segnale a due bit che rappresenta la mossa giocata dal primo giocatore.
- G2: segnale a due bit che rappresenta la mossa giocata dal secondo giocatore.

Il DATAPATH è formato dalle seguenti componenti:

- AND (and): porta logica.
- DECODER (decoder): componente composto da un input a *n* bit e output a 2^n bit, che alza al valore 1 solamente il bit specificato dall'input.
- DEMULTIPLEXER (dmux): componente composto da più ingressi diversi e un'unica uscita, pilotata da un segnale di selezione.
- EQUAL (equal): componente aritmetico.
- MAGGIORE UGUALE (ge): componente aritmetico.
- MAGGIORE (gt): componente aritmetico.
- MINORE UGUALE (le): componente aritmetico.
- MULTIPLEXER (mux): componente ad un unico input e più output, pilotato da un segnale di selezione.
- NOT (not): componente logico.
- OR (or): componente logico.
- REGISTRO (reg): utilizzato per memorizzare dati intermedi. Nello specifico, nel circuito sono utilizzati i seguenti registri:
    - REG5 (RegTurni): utilizzato per tenere traccia dei turni.
    - REG4 (RegPunti): utilizzato per tenere traccia del vantaggio tra un giocatore e l'altro. Infatti, piuttosto che creare due registri separati per i punteggi di ogni giocatore da confrontare, abbiamo deciso di utilizzare un registro unico che contenesse la differenza continuamente aggiornata delle vincite di ogni giocatore, in complemento a due.
    - REG3 (RegMossaPrecedente): utilizzato per memorizzare la mossa precedente del giocatore vincente.
    - REG2 (RegPrimi4Turni) : utilizzato per contare i primi 4 turni di una partita.
    - REG (END) : utilizzato, nel momento in cui la partita è terminata, per mantenere un output più pulito.
- SOMMA (sum): componente aritmetico.
- XOR (xor): componente logico.
- XNOR (xnor): componente logico.

Riportiamo il grafico del DATAPATH:


SETUP
INPUT GIOCATORE 1
INPUT GIOCATORE 2
2 bit
2 bit
00100 (4)
SUM
4 bit
1 0
SUM
01
00000
EQ
regTurni [5 bit]
1 0
1 0
00
1 0
regPrimi4Turni [2 bit]
00
EQ
00
EQ
SUM
11111 (-1)
11
EQ
000
1 0
1 bit
0 1
MOVE NOT VALID
2 bit
EQ
2 bit
1 bit
regMossaPrecedente [1+2 bit]
AND
0000
1 0
0 1
2 bit
1 bit
regPunti [4 bit]
EQ
1 0
SUM
1 0
1111 (-1)
0001 (+1)


1
01 10 11
1
01 10 11
VINCITA GIOCATORE 2
00 00 11
11 10 01 00
00
1 0
MANCHE
EQ
EQ
1
-1
VINCE G. 1
1110 (-2)
<=[1]
<
0001 (1)
VINCE G. 2
10 01 11
10 01 00
10 01
Distacco maggiore/uguale a 2
10 01 00
00
0 1
00
0 1
00
1 0
1
1 0
0
1 0
END [1 bit]
PARTITA
END

Statistiche del circuito: SIS

- Comando: “state_assign jedi” sulla FSM:

```
sis> print_stats
FSM             pi= 2   po= 1   nodes=  3       latches= 2
lits(sop)=  12  #states(STG)=   3
```

- Comando: “source script.rugged” sulla FSM:

```
sis> print_stats
FSM             pi= 2   po= 1   nodes=  2       latches= 2
lits(sop)=   7  #states(STG)=   3
```

- Statistiche pre-ottimizzazione DATAPATH

```
sis> print_stats
DATAPATH        pi= 5   po= 6   nodes=218       latches=15
lits(sop)= 796
```

- Statistiche del DATAPATH dopo il comando “script.rugged”

```
sis> print_stats
DATAPATH        pi= 5   po= 6   nodes= 35       latches=14
lits(sop)= 200
```

- Statistiche pre-ottimizzazione FSMD

```
sis> read_blif morra_cinese.blif
[ Warning: ... ]
sis> print_stats
MORRA_CINESE    pi= 5   po= 5   nodes=220       latches=17
lits(sop)= 807
```

- Statistiche della FSMD con il comando comando “script.rugged”.
  Ottimizzata 4 volte per ottenere una delle possibili versione più ottimizzabili:

```
sis> read_blif morra_cinese.blif
[ Warning: ... ]
sis> print_stats
MORRA_CINESE    pi= 5   po= 5   nodes=220       latches=17
lits(sop)= 807
sis> full_simplify
sis> source script.rugged
sis> print_stats
MORRA_CINESE    pi= 5   po= 5   nodes= 37       latches=16
lits(sop)= 206
sis> full_simplify
sis> source script.rugged
sis> print_stats
MORRA_CINESE    pi= 5   po= 5   nodes= 34       latches=16
```

```
lits(sop)= 201
sis> full_simplify
sis> source script.rugged
sis> print_stats
MORRA_CINESE    pi= 5   po= 5   nodes= 34       latches=16
lits(sop)= 200
sis> full_simplify
sis> source script.rugged
sis> print_stats
MORRA_CINESE    pi= 5   po= 5   nodes= 34       latches=16
lits(sop)= 200
```

Mapping

Una volta terminata l'ottimizzazione, dobbiamo mapparlo ottimizzando per area. Con le seguenti statistiche:

```
sis> map -m 0
sis> print_map_stats
Total Area              = 4128.00
Gate Count              = 110
Buffer Count            = 16
Inverter Count          = 22
Most Negative Slack     = -35.00
Sum of Negative Slacks  = -410.60
Number of Critical PO   = 21
```

Scelte progettuali

Abbiamo deciso di estendere il più possibile il DATAPATH per essere facilmente ottimizzabile.
In seguito abbiamo abbiamo utilizzato il segnale di controllo FINE per comunicare alla FSM che la partita è giunta al termine.
Inoltre, nel momento in cui la FSM si trova nello stato di RESET, il segnale SETUP del DATAPATH viene portato ad 1, creando così un output più pulito del circuito (“0000”).
Un’ulteriore scelta progettuale è stata quella di non tenere traccia dei punteggi singoli per ogni giocatore, come spiegato sopra, ma di tenere traccia della differenza nel registro numero 5. Infatti, tenendo conto della differenza, qualora sia stata superata la soglia dei 4 turni minimi, se il vantaggio di due punti è positivo, la vincita è da attribuirsi al primo giocatore, se negativo al secondo giocatore. Allo stesso modo, se si arriva alla fine dei turni, se il vantaggio di un punto è positivo, vince il primo giocatore, se negativo il secondo.