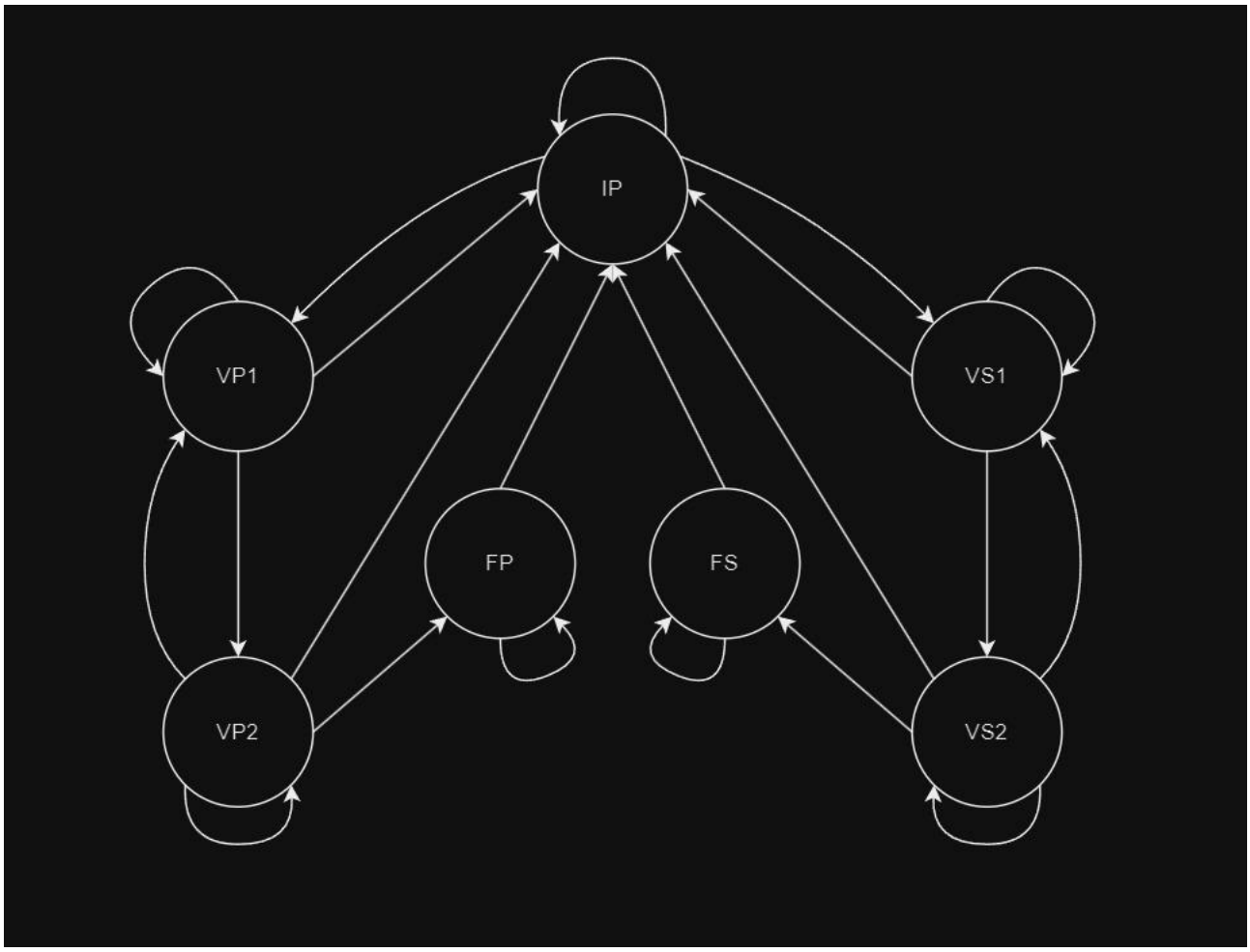
ANGELO VACCARO (480616)

MARWAN CHTINI (510527)

Elaborato SIS/Verilog

FSM



Le transizioni mostrate nel disegno vengono spiegate nella tabella sottostante:

STATE TRANSITION TABLE

	1----	000--	0010-	00110	00111	0100-	01010	01011	0110-	01110	01111
IP	IP/00	IP/00	VP1/00	VP1/00	FP/01	VS1/00	VS1/00	FS/10	IP/00	IP/00	IP/11
VP1	IP/00	VP1/00	VP2/00	VP2/01	VP2/01	IP/00	IP/00	IP/11	VP1/00	VP1/00	FP/01
VP2	IP/00	VP2/00	FP/00	FP/01	FP/01	VP1/00	VP1/00	FP/01	VP2/00	FP/01	FP/01
FP	IP/00	FP/01	FP/01	FP/01	FP/01	FP/01	FP/01	FP/01	FP/01	FP/01	FP/01
VS1	IP/00	VS1/00	IP/00	IP/00	IP/11	VS2/00	VS2/10	VS2/10	VS1/00	VS1/00	FS/10
VS2	IP/00	VS2/00	VS1/00	VS1/00	FS/10	FS/00	FS/10	FS/10	VS2/00	FS/10	FS/10
FS	IP/00	FS/10	FS/10	FS/10	FS/10	FS/10	FS/10	FS/10	FS/10	FS/10	FS/10

LEGENDA:

- ***STRUTTURA TABELLA:***

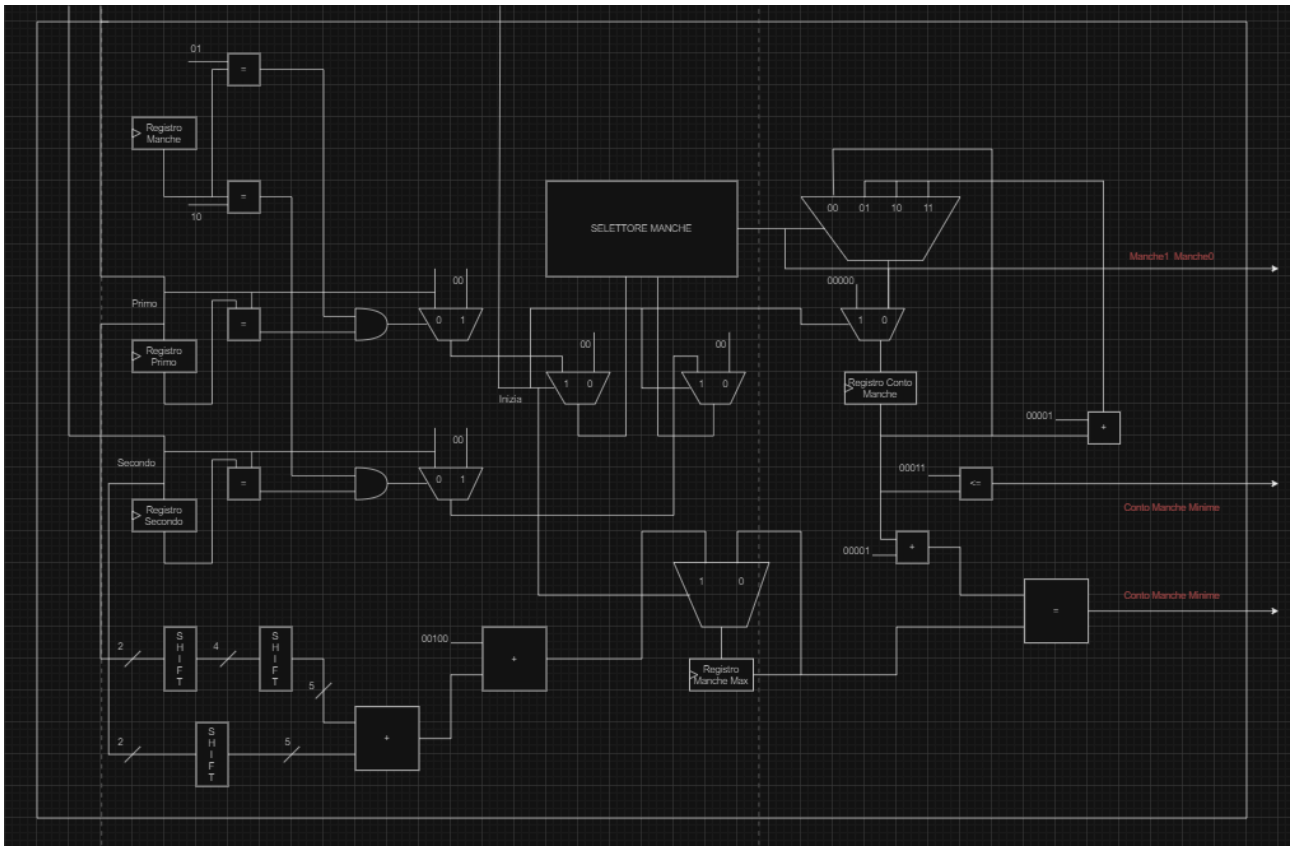
	5 INPUT
STATO CORRENTE	*STATO PROSSIMO / 2 OUTPUT*

- ***STATI:***

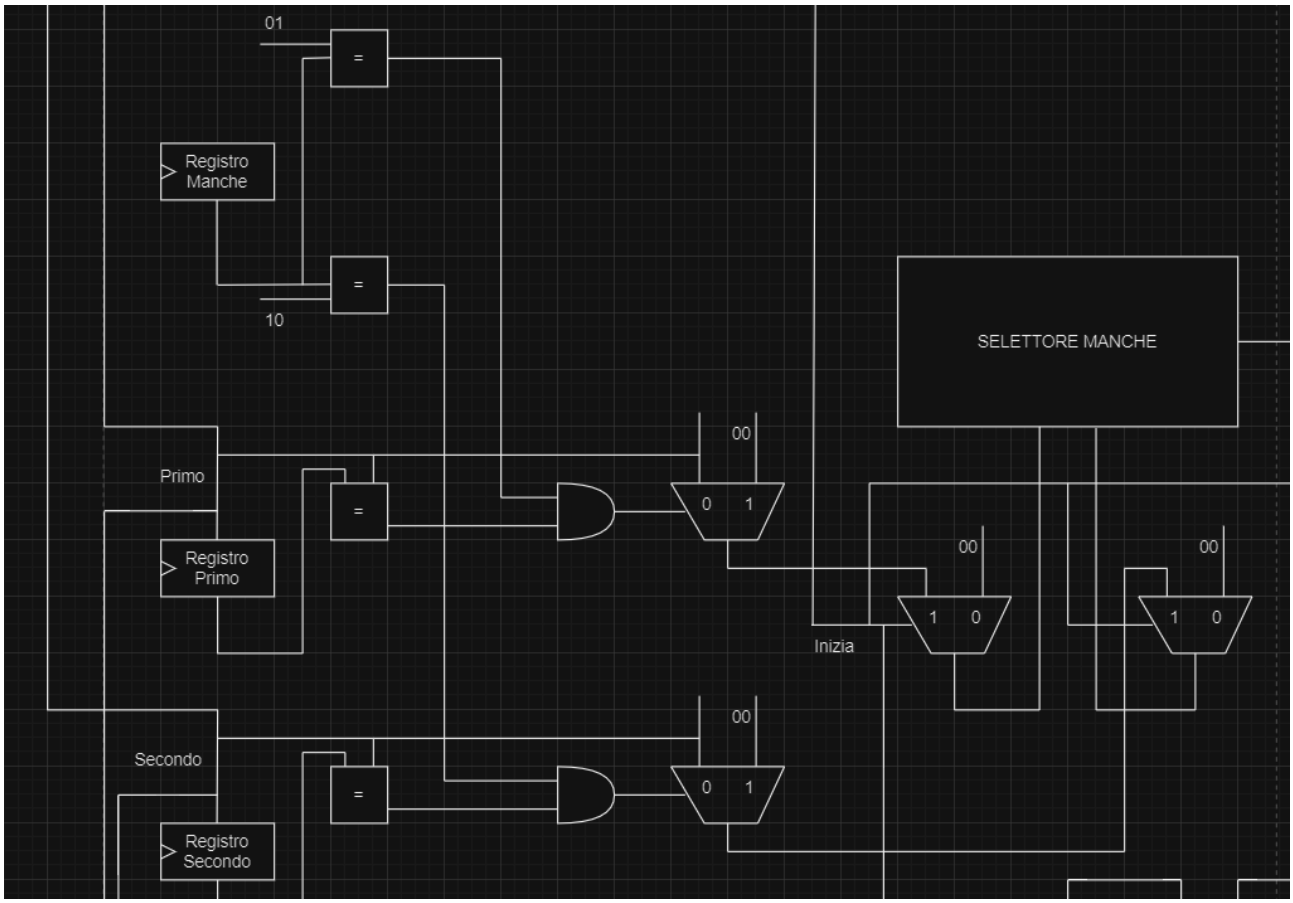
NOME	*SIGNIFICATO*
IP	Inizio o Pareggio
VP1	Vantaggio Primo giocatore = 1
VP2	Vantaggio Primo giocatore = 2
FP	Fine – vittoria Primo giocatore
VS1	Vantaggio Secondo giocatore = 1
VS2	Vantaggio Secondo giocatore = 2
FS	Fine – vittoria Secondo giocatore

DATAPATH

Il nostro datapath avrà come outputs i bit manche1/manche0/cmin(conto delle manche minime)/cmax(conto delle manche massime) e come inputs inizia/primo1 e primo0/secondo1 e secondo0:

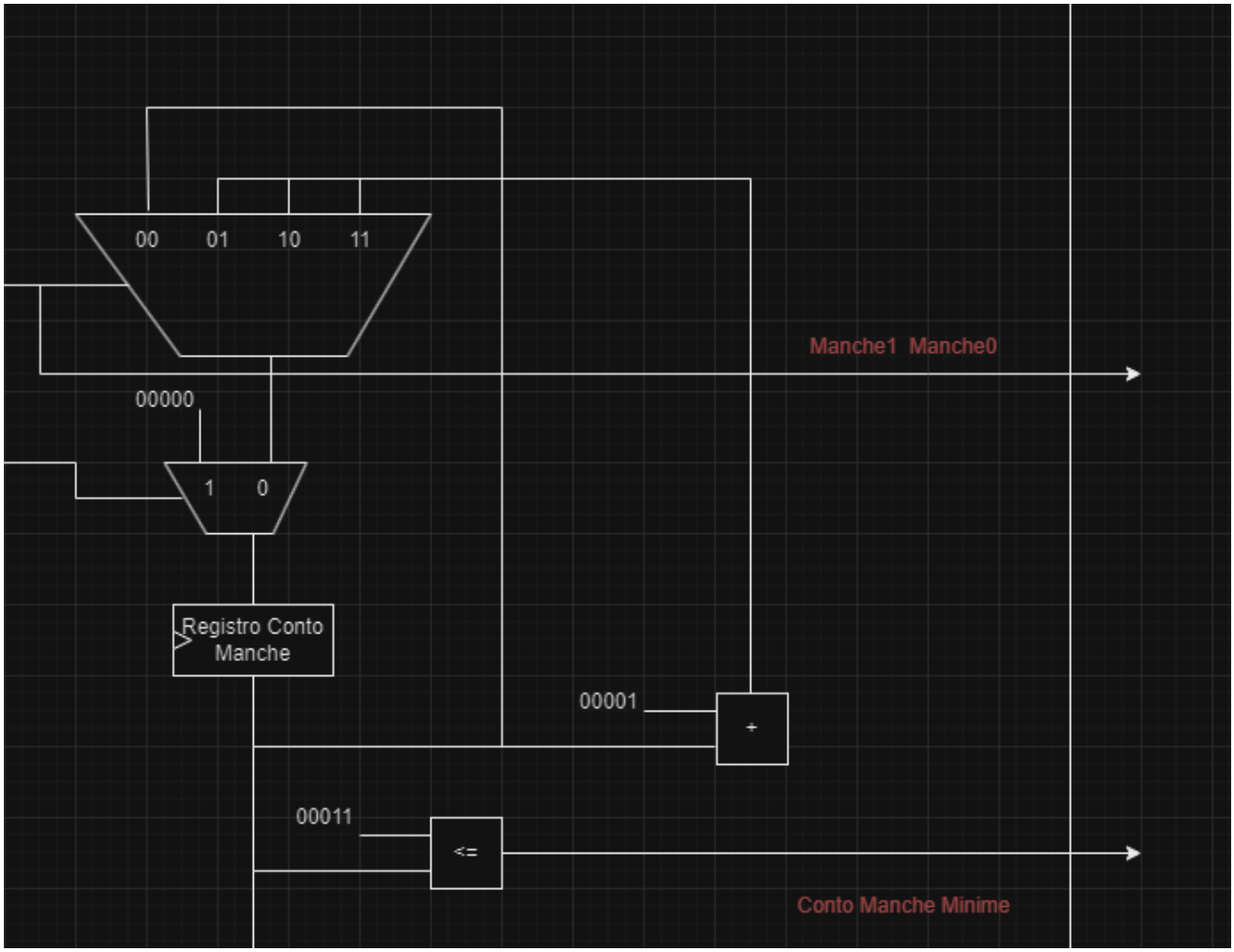

01
=
Registro Manche
=
10
SELETTORE MANCHE
00 01 10 11
Manche1 Manche0
00000
Primo
=
Registro Primo
00
0 1
1 0
Inizia
Registro Conto Manche
00001
+
Secondo
=
Registro Secondo
00011
<=
Conto Manche Minime
00001
+
1 0
=
Conto Manche Minime
2
SHIFT
4
SHIFT
5
00100
+
Registro Manche Max
2
SHIFT
5
+

Le manche vengono calcolate come riportato nell'immagine sottostante:



Come prima cosa andremo ad inserire il valore di primo nel registro, l'output del registro andrà in un uguale il quale comparerà il valore di primo del registro e il valore di primo attuale. Questo serve per vedere se il giocatore ha giocato la stessa mossa della manche precedente, la stessa verifica avviene anche per il giocatore 2. Il registro che vediamo, chiamato registro manche, riceverà in input il valore finale di manche1 e manche0 e tra poco vedremo come verranno calcolati questi bit. Il valore del registro finirà in due uguali, uno che confronterà con i bit 01 (manche vinta dal giocatore 1) e l'altro con i bit 10 (manche vinta dal giocatore 2). I valori che escono dai due uguali andranno in un AND, il bit uscito dall'uguale con 01 andrà con il valore uscito dall'uguale del giocatore 1, e 10 per il giocatore 2. Questo AND ci serve perché se l'output sarà 1 vorrà dire che il giocatore ha giocato la stessa mossa, e la manche era uguale nella manche precedente. I bit usciti dall'AND serviranno come bit di selezione che farà uscire da un multiplexer: 00 in caso di bit di selezione ad uno e primo1/primo0 in caso di bit a 0, stessa cosa per il giocatore 2. I due multiplexer che vediamo di fianco a quelli appena descritti servono per fare in modo che la manche con inizia a 1 non venga calcolata. I valori andranno in questo

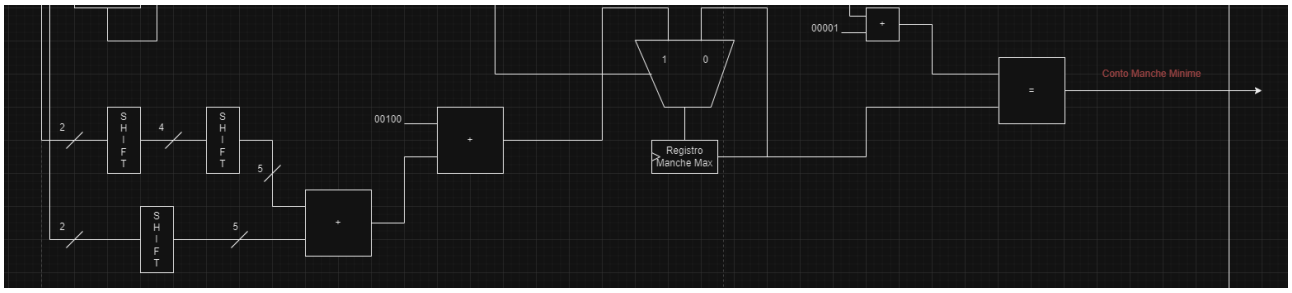
“componente” chiamato SELETTORE, quest’ultimo è composto da una tabella di verità che segue la logica dell’output previsto dall’elaborato delle manche e avrà quindi come uscita manche1/manche0. La nostra FSM richiedeva anche il bit di manche minime, il quale viene calcolato in questo modo:



Nel multiplexer più grande avremo come bit di selezione manche1/manche0, quest’ultimo avrà come ingressi: 00→valore del registro conto manche non aggiornato, quindi quando la manche non è valida non verrà aggiornato il registro, 01/10/11→valore del registro aggiornato e quindi manche valida. L’output del multiplexer finisce in un altro multiplexer, il quale ha come bit di selezione inizia, se inizia è a 1 fa andare nel registro del conto 00000 quindi lo resetta, in caso contrario il valore in aggiornamento. Come spiegato prima il valore del registro va in un sommatore, il quale output finisce nel multiplexer (manche valide), o direttamente nel multiplexer (manche non valida). L’output del registro, come vediamo nell’immagine sopra, finisce in un minore o uguale, quest’ultimo confronterà il

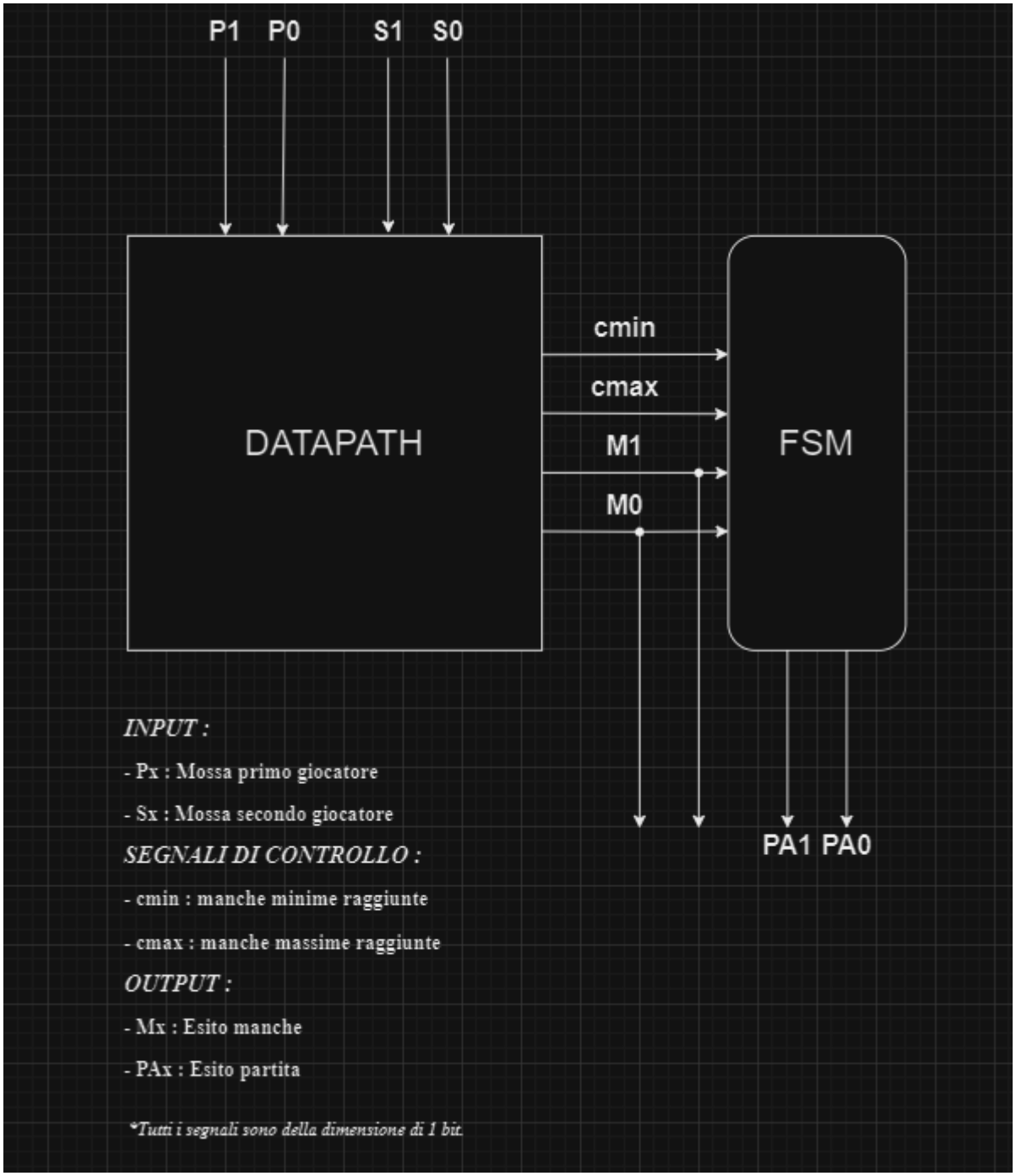
valore del registro con 00011 quindi 3 (utilizziamo minore o uguale e non minore per ovviare al problema di ritardo dei registri).

Il calcolo delle manche massime viene invece calcolato in questo modo:



I primi 3 componenti che vediamo sono degli shifter, il primo prende in ingresso primo1/primo0 e gli aggiunge 2 zeri a destra (es. i→p1/p0=10 o→1000). Il valore finirà in un altro shifter che aggiunge uno zero davanti al valore quindi se il valore è1000 diventerà 01000. Il terzo shifter che vediamo serve per avere il valore di secondo in 5 bit (es. i→s1/s0=11 o→00011). Modificati i bit di primo e secondo li sommeremo e successivamente aggiungeremo 4, così facendo avremo un minimo di 4 manche ed un massimo di 19. Il valore andrà in un multiplexer, il quale, oltre al valore uscito dal sommatore, avrà in ingresso l'output del registro delle manche massime. Questo multiplexer avrà come bit di selezione inizia e ci servirà per avere per tutto il corso della partita lo stesso valore di manche massime, in caso di inizia ad 1 avremo il valore del sommatore, in caso di inizia a 0 quello del registro. Il valore del registro andrà in un uguale, il quale confronterà il valore manche massime con il valore di conto manche visto prima (vediamo che al valore conto manche viene aggiunto 1, questo serve sempre per rimediare al ritardo ed avere il valore attuale del registro nell'uguale). Il valore uscito dall'uguale è il bit di manche massime il quale si alzerà ad 1 se sono state raggiunte.

FSMD



Ottimizzazione per area del circuito

sis> rl DATAPATH.blif

Warning: network `DATAPATH', node "CIN" is not driven (zero assumed)

Warning: network `DATAPATH', node "COUT" does not fanout

Warning: network `DATAPATH', node "pr2" does not fanout

Warning: network `DATAPATH', node "pr1" does not fanout

Warning: network `DATAPATH', node "pr0" does not fanout

Warning: network `DATAPATH', node "CiN" is not driven (zero assumed)

Warning: network `DATAPATH', node "CoUT" does not fanout

Warning: network `DATAPATH', node "cin" is not driven (zero assumed)

Warning: network `DATAPATH', node "cout" does not fanout

Warning: network `DATAPATH', node "ci" is not driven (zero assumed)

Warning: network `DATAPATH', node "co" does not fanout

sis> print_stats

DATAPATH pi= 5 po= 4 nodes=152 latches=16

lits(sop)= 714

sis> source script.rugged

sis> print_stats

DATAPATH pi= 5 po= 4 nodes= 32 latches=16

lits(sop)= 230

sis> source script.rugged

sis> print_stats

DATAPATH pi= 5 po= 4 nodes= 31 latches=16

lits(sop)= 234

Ulteriori utilizzi dello script.rugged non portano ad ulteriori miglioramenti del datapath.

sis> write_blif

(trascrivo il datapath ottimizzato)

Mapping tecnologico del circuito

sis> rl FSMDPREOTT.blif

Warning: network `DATAPATH', node "CIN" is not driven (zero assumed)

Warning: network `DATAPATH', node "COUT" does not fanout

Warning: network `DATAPATH', node "pr2" does not fanout

Warning: network `DATAPATH', node "pr1" does not fanout

Warning: network `DATAPATH', node "pr0" does not fanout

Warning: network `DATAPATH', node "CiN" is not driven (zero assumed)

Warning: network `DATAPATH', node "CoUT" does not fanout

Warning: network `DATAPATH', node "cin" is not driven (zero assumed)

Warning: network `DATAPATH', node "cout" does not fanout

Warning: network `DATAPATH', node "ci" is not driven (zero assumed)

Warning: network `DATAPATH', node "co" does not fanout

Warning: network `FSMD', node "COUT" does not fanout

Warning: network `FSMD', node "pr2" does not fanout

Warning: network `FSMD', node "pr1" does not fanout

Warning: network `FSMD', node "pr0" does not fanout

Warning: network `FSMD', node "CoUT" does not fanout

Warning: network `FSMD', node "cout" does not fanout

Warning: network `FSMD', node "co" does not fanout

sis> print_stats

FSMD pi= 5 po= 4 nodes=165 latches=19

lits(sop)= 790

sis> source script.rugged

sis> print_stats

FSMD pi= 5 po= 4 nodes= 41 latches=19

lits(sop)= 314

sis> source script.rugged

sis> print_stats

FSMD pi= 5 po= 4 nodes= 40 latches=19

lits(sop)= 315

Ulteriori utilizzi dello script.rugged non portano miglioramenti della fsmd non ottimizzata.

sis> write_blif

Scrivo la fsmd ottimizzata e procedo col mapping:

sis> rl FSMD.blif

sis> read_library synch.genlib

sis> map -m 0 -s

warning: unknown latch type at node '{[227]}' (RISING_EDGE assumed)

warning: unknown latch type at node '{[228]}' (RISING_EDGE assumed)

warning: unknown latch type at node '{[229]}' (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

I warning riguardanti il tipo di latch sono dovuti al fatto che i registri utilizzati nel datapath sono resettabili. In ogni caso sis suppone che siano di tipo rising-edge, quindi possiamo ignorarli.

>>> before removing serial inverters <<<

# of outputs: 23

total gate area: 5092.00

maximum arrival time: (33.60,33.60)

maximum po slack: (-2.20,-2.20)

minimum po slack: (-33.60,-33.60)

total neg slack: (-365.40,-365.40)

# of failing outputs: 23

>>> before removing parallel inverters <<<

# of outputs: 23

total gate area: 5060.00

maximum arrival time: (31.40,31.40)

maximum po slack: (-2.20,-2.20)

minimum po slack: (-31.40,-31.40)

total neg slack: (-361.00,-361.00)

# of failing outputs: 23

# of outputs: 23

total gate area: 4948.00

maximum arrival time: (31.40,31.40)

maximum po slack: (-2.20,-2.20)

minimum po slack: (-31.40,-31.40)

total neg slack: (-361.00,-361.00)

# of failing outputs: 23

Dopo l'ottimizzazione siamo passati da un'area totale di 5092 ad un'area di 4948.