UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

RELAZIONE ELABORATO SIS
ARCHITETTURA DEGLI ELABORATORI

Studenti:

MARZARI LUCA VR421483

PINTANI DEBORAH VR422805

Docente: SETTI FRANCESCO

ANNO ACCADEMICO 2017/2018

INDICE

1. Specifiche

Si progetti un dispositivo per la gestione intelligente del consumo di energia elettrica all'interno di un sistema domotico. Il dispositivo è basato su un circuito sequenziale che riceve in ingresso lo stato acceso/spento di un numero finito di dispositivi di cui è noto il consumo istantaneo a priori, e fornisce in uscita la fascia di consumo ad ogni ciclo di clock. Qualora l'assorbimento istantaneo sia superiore al limite di 4.5kW per più di 5 cicli di clock consecutivi, il sistema deve disattivare l'interruttore generale. Al fine di prevenire questa situazione, il dispositivo può disattivare la lavatrice e la lavastoviglie (in questo ordine di priorità).
Il circuito è composto da un controllore e un datapath con i seguenti ingressi e uscite (nel seguente ordine!).

INPUTS:
• RES_GEN [1]: quando vale 1 e INT_GEN=0, gli interruttori vengono armati (INT_GEN, INT_WM e INT_DW commutano a 1) ed il sistema si accende. Quando INT_GEN=1, il valore di RES_GEN non ha rilevanza.
• RES_WM [1]: quando vale 1 e INT_WM=0, l'interruttore della lavatrice deve essere riarmato (INT_WM commuta a 1) ed il carico relativo alla lavatrice deve essere nuovamente preso in considerazione. Quando INT_WM=1, il valore di RES_WM non ha rilevanza.
• RES_DW [1]: quando vale 1 e INT_DW=0, l'interruttore della lavatrice deve essere riarmato (INT_DW commuta a 1) ed il carico relativo alla lavatrice deve essere nuovamente preso in considerazione. Quando INT_DW=1, il valore di RES_DW non ha rilevanza.
• LOAD [10]: stato di accensione (1=ON, 0=OFF) dei carichi elettrici. Ogni carico ha un suo consumo istantaneo associato. Il carico complessivo del circuito è dato dalla somma di tutti i carichi accesi contemporaneamente.

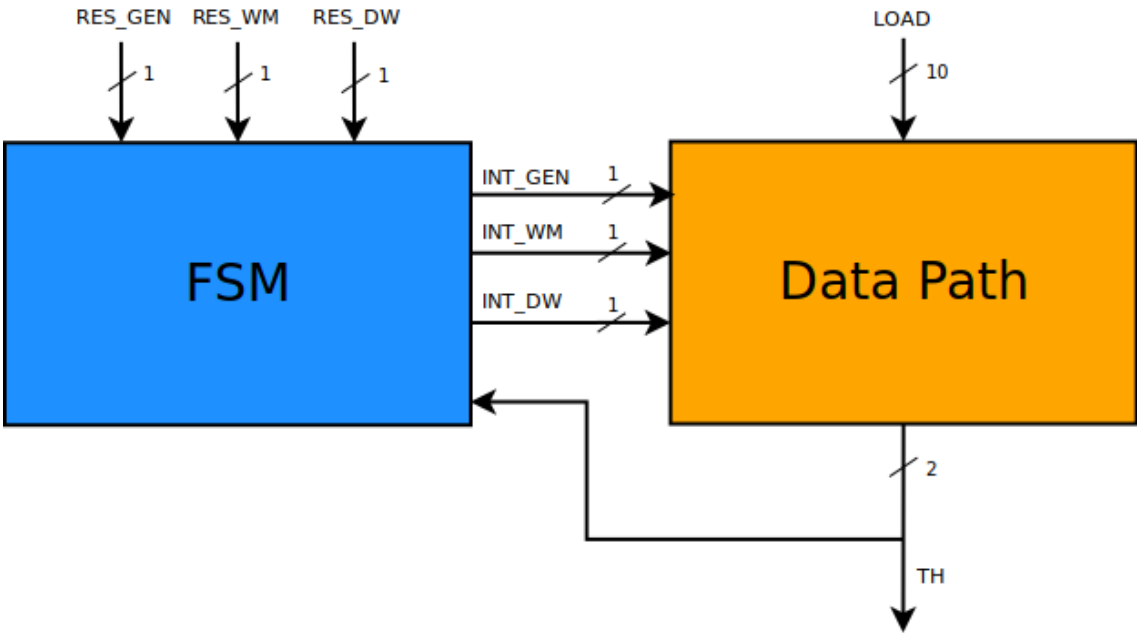
OUTPUTS:
• INT_GEN [1]: indica lo stato di attivazione (1=ON, 0=OFF) dell'interruttore generale. Inizialmente è sempre posto a 0.
• INT_WM [1]: indica lo stato di attivazione (1=ON, 0=OFF) dell'interruttore relativo alla lavatrice. Inizialmente, e dopo ogni spegnimento del sistema, è sempre posto a 0.
• INT_DW [1]: indica lo stato di attivazione (1=ON, 0=OFF) dell'interruttore relativo alla lavastoviglie. Inizialmente, e dopo ogni spegnimento del sistema, è sempre posto a 0.
• TH [2]: indica la fascia di consumo istantanea secondo la seguente codifica: F1=00, F2=01, F3=10, OL=11.

2. Progettazione iniziale

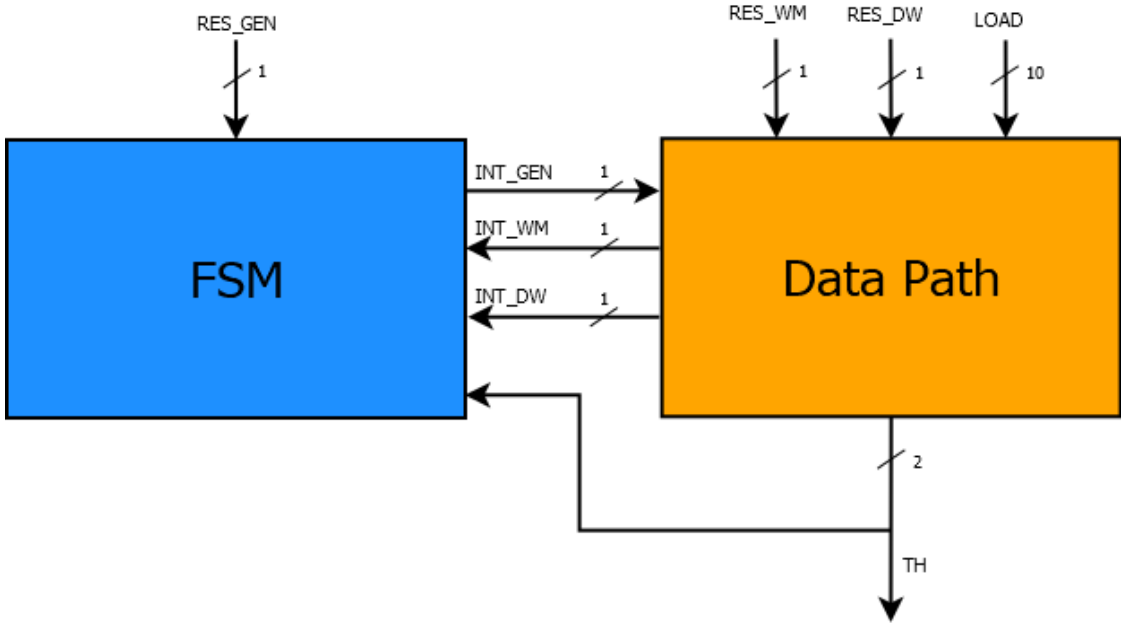
Sono qui di seguito presentati gli schemi intermedi dei componenti del circuito.

2.1. FSMD

Inizialmente avevamo pensato erroneamente di gestire gli *INT_GEN*, *INT_WM* e *INT_DW* nell'*FSM*, ma successivamente ci siamo resi conto che era necessario l'utilizzo del *DATAPATH*.



Dopo aver costruito l'*FSM* e il *DATAPATH*, siamo giunti alla conclusione di spostare i bit di *RES_WM*, *RES_DW* nel *DATAPATH*.

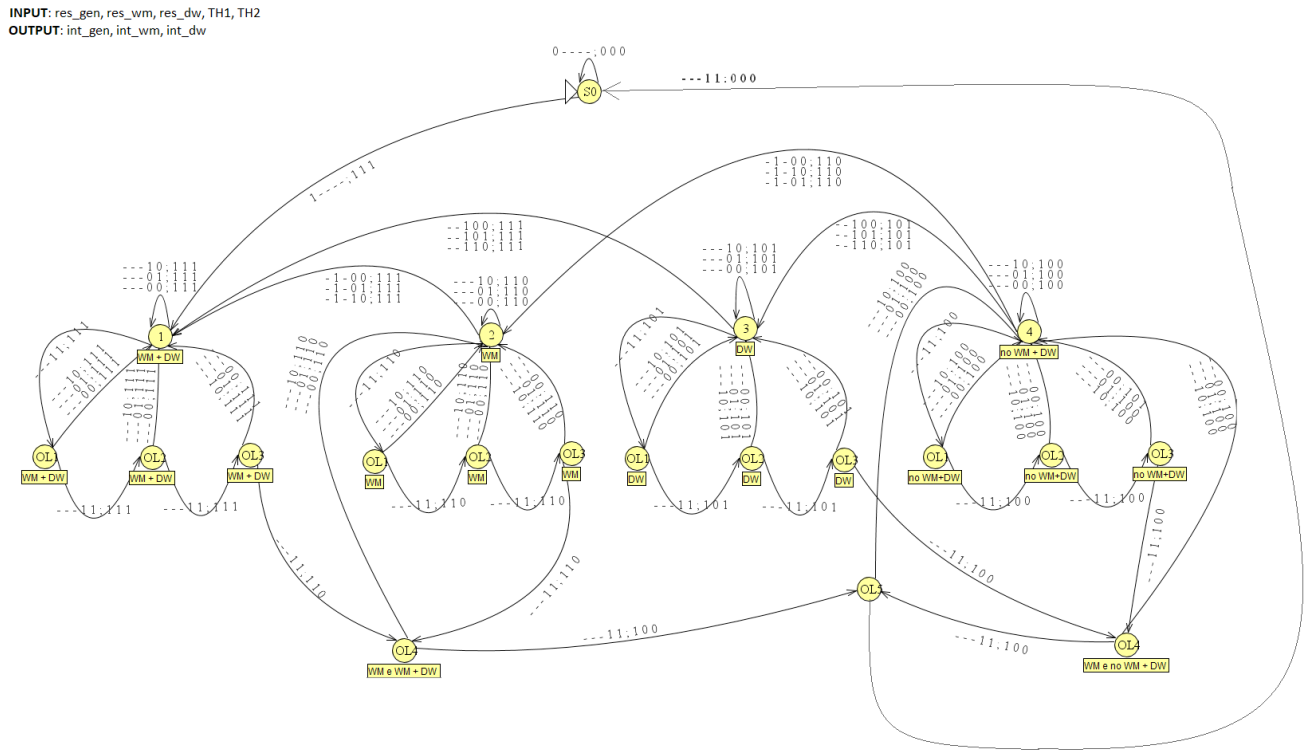


La versione finale è presentata al paragrafo 3.

2.2. FSM

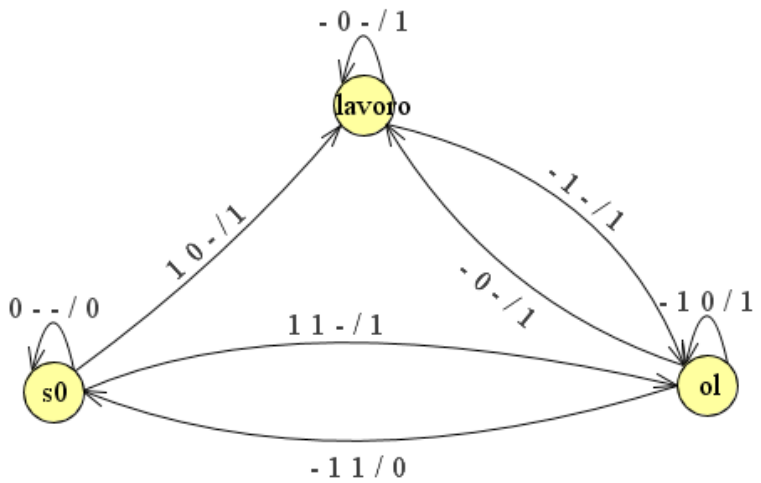
In un primo momento, la nostra *FSM* era molto complessa e presentava alcuni errori, come la presenza degli *INT* in output. Qui sotto è presentato lo schema.

L'idea iniziale era quella di contare i cicli di *OL* nella *FSM*, cosa che risultava complicata.



Una seconda versione si può vedere sotto. Questa è parzialmente corretta, in quanto soggetta a ritardi di comunicazione con il datapath. La versione definitiva è presentata al paragrafo 4.

Inputs: res_gen, ol, off
Output: int_genFSM



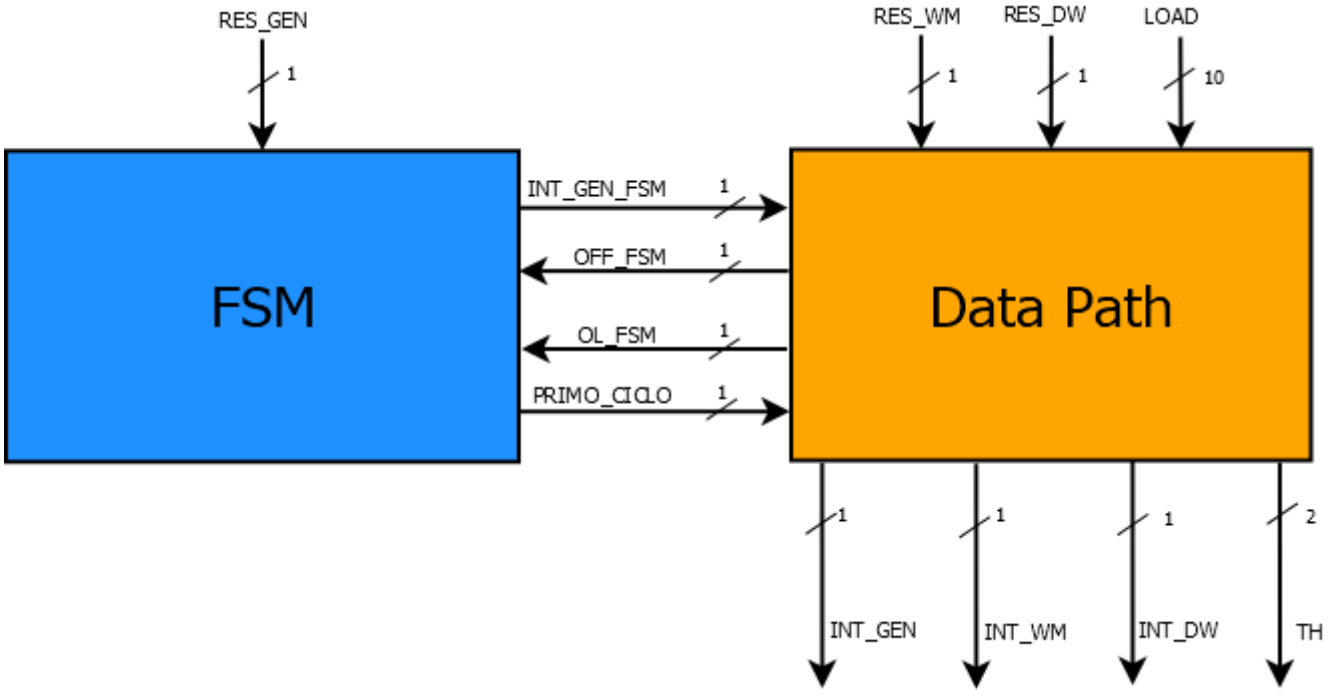
3. Architettura generale del circuito

Dopo aver modificato i bit di *INT_DW* e *INT_WM,* non più in output alla FSM ma direttamente in output dal datapath, abbiamo aggiunto 3 bit di controllo (*OFF_FSM*, *OL_FSM*, *PRIMO_CICLO*) che permettono all'*FSM* di comunicare con il datapath, ed eventualmente cambiare stato della macchina.

Il nostro circuito è composto da una *FSM* (macchina a stati finiti) che presenta solamente 3 stati e un DATAPATH (elaboratore). Questi, collegati, formano la *FSMD* che permette di eseguire tutte le richieste presentate nel capitolo 1.

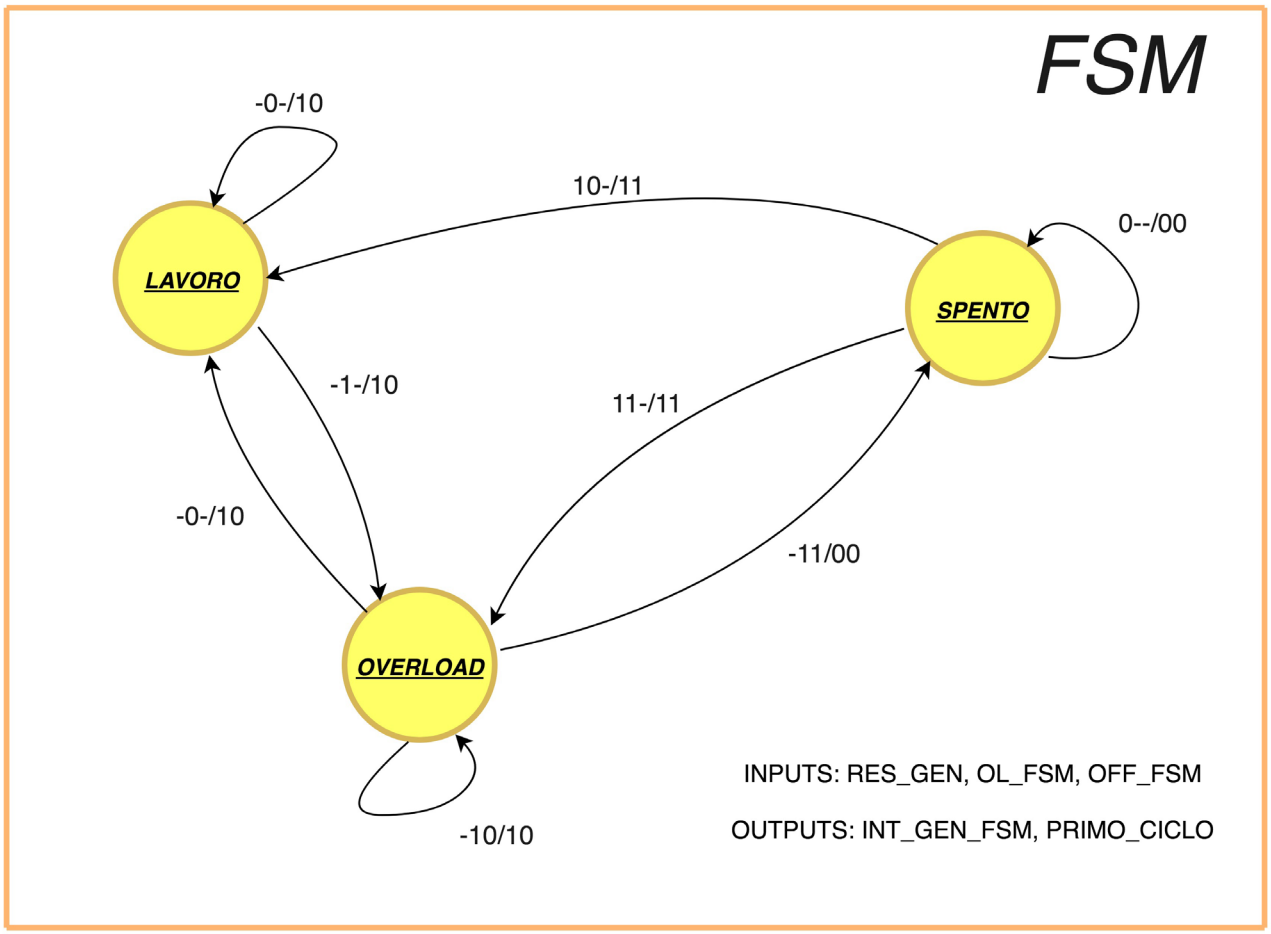
Tra *FSM* e *DATAPATH* sono utilizzati alcuni bit di ausilio:

- *PRIMO_CICLO:* indica se la macchina si trova al primo ciclo. È utilizzato per calcolare la fascia *DW* e *WM* all'avvio della macchina. Avendo modificato la macchina in modo tale che mantenesse lo stato precedente, senza questo bit avrebbe calcolato male la fascia all'accensione.
- *INT_GEN_FSM*: passa al datapath l'accensione o meno della macchina.
- *OFF_FSM*: passa all'*FSM* il segnale di spegnimento della macchina.
- *OL_FSM*: passa all'*FSM* il segnale di overload per il cambio stato.



4. Diagramma degli stati del controllore

La nostra scelta finale è stata quella di semplificare il più possibile il circuito mediante l'utilizzo principale del *DATAPATH*, semplificando così al massimo la *FSM*.
Infatti, quest'ultima presenta solamente 3 stati come mostrato nello schema seguente:



Un primo stato è rappresentato dallo stato *SPENTO*, che è la situazione in cui il circuito si trova inizialmente e rimane in questo stato finché il *RES_GEN[1]* non viene commutato a 1.
Una volta che quest'ultimo viene armato, la macchina passa allo stato di *LAVORO* e a quel punto il valore di *RES_GEN* non ha più alcuna importanza (a meno che la macchina non ritorni allo stato di spento).

La macchina rimane nel secondo stato fintanto che l'assorbimento istantaneo non superi il valore di 4.5kW, qualora si verificasse quest'ultima situazione, allora il bit di *OL,* calcolato nel *DATAPATH*, commuterebbe a 1 e una volta inviato il segnale alla *FSM*, porterebbe la macchina nel terzo stato ovvero di *OVERLOAD*.

Possiamo notare questo passaggio grazie allo schema sopra riportato: la macchina si trova nello stato di *LAVORO* e i bit di input che la *FSM* riceve sono *RES_GEN*, il bit di *OL*, che arriva dal *DATAPATH*, e il bit di *OFF* che arriva anch'esso dal datapath. Quindi con input [- 1 -] la macchina passa da *LAVORO* a *OVERLOAD*.
Il circuito rimane in questo terzo e ultimo stato fintanto che il bit di *OL* rimane a 1, oppure il bit di *OFF* rimane a 0.

Qualora quest'ultimo commutasse a 1, è necessario spegnere la macchina poiché è stato superato il limite di 6 cicli di clock consecutivi in stato di *OVERLOAD*.
Se invece il segnale di *OL* dovesse tornare a 0, la macchina torna allo stato precedente ovvero quello di semplice *LAVORO*.

Dallo schema precedente possiamo notare che, se siamo nello stato di *OVERLOAD* e come bit di input viene ricevuto il segnale [- 0 -], il bit di *OL* è tornato a 0, e di conseguenza si ritorna allo stato di lavoro.
Viceversa se in ingresso si presenta [- 1 1] la macchina commuta allo stato di *SPENTO*.

5. Architettura del Datapath

Il nostro datapath prevede i seguenti input:

- INT_GEN_FSM[1]
- PRIMO_CICLO[1]
- RES_DW[1]
- RES_WM[1]
- LOAD[10]

e genera come output da restituire alla FSM o alla FSMD:

- INT_GEN[1]
- INT_DW[1]
- INT_WM[1]
- TH[2]
- OL_FSM[1]
- OFF_FSM[1]

Per realizzare il datapath ovvero l'elaboratore del circuito abbiamo utilizzato:

1) Moltiplicatori a 10 bit: lo scopo di questi moltiplicatori è moltiplicare la costante di consumo dell'elettrodomestico per il valore di *LOAD* ricevuto in input ad ogni giro di clock.

2) Sommatori a 10 bit: i risultati dei moltiplicatori diventano gli input dei sommatori a 10 bit. Questi, insieme, restituiscono il consumo complessivo. Quest'ultimo viene poi moltiplicato per il valore del bit *INT_GEN_FSM*, ovvero il bit che viene dato in input dalla *FSM* al *DATAPATH*, in base allo stato in cui ci si trova.

3) Comparatore Maggiore a 10 bit e Minore Uguale a 10 bit: utilizzati per calcolare, qualora la macchina fosse accesa, il valore preciso del consumo istantaneo ad ogni ciclo di clock.

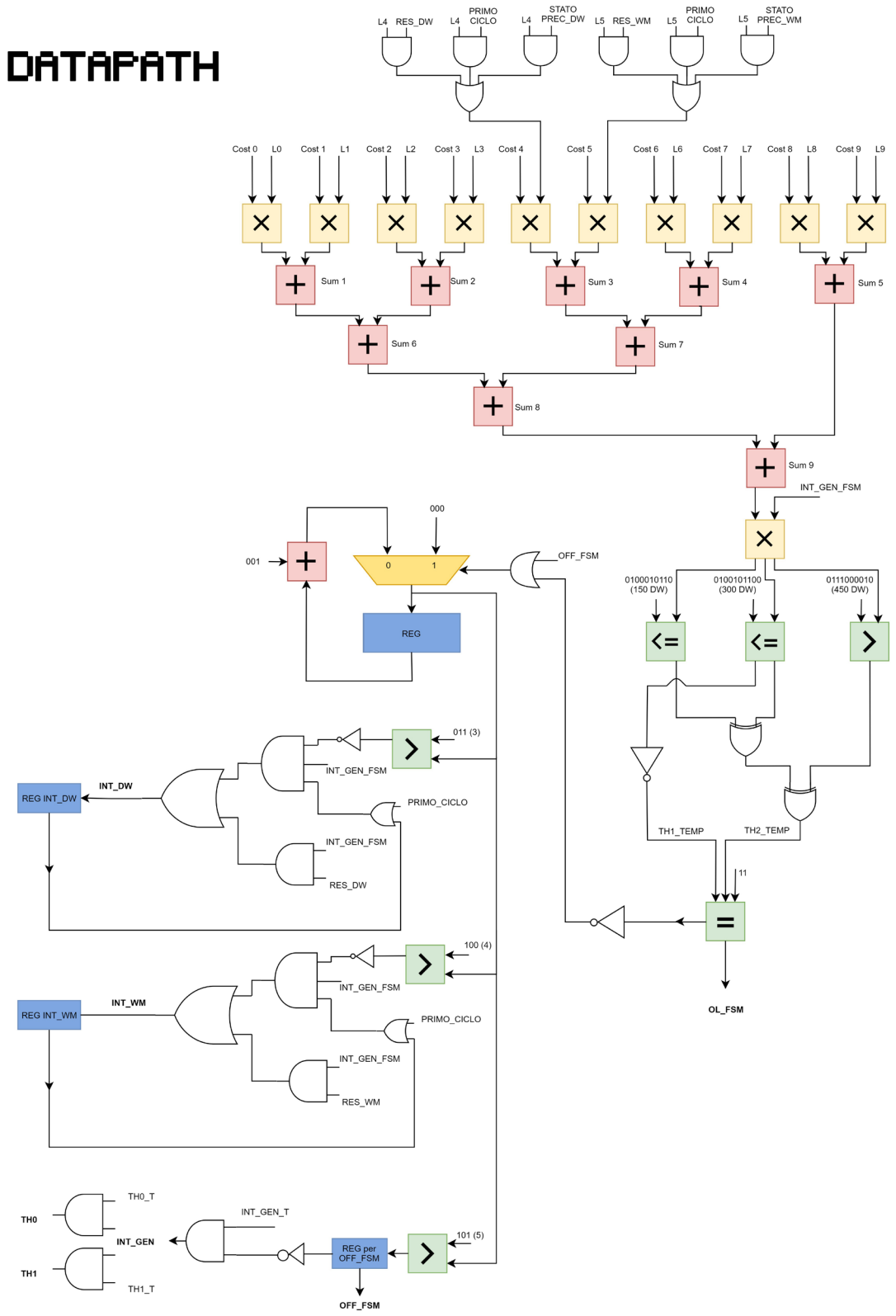
4) Porte logiche AND, OR, NOT, XOR a 2 bit/3 bit e *MUX*: utilizzate sempre al fine di determinare il *TH*.

5) Comparatore uguale a 4 bit per determinare il valore di *OL*.

6) Dei registri a 3 bit: utilizzati per mantenere il conteggio dei cicli di clock durante lo stato di overload e per mantenere lo stato precedente dei bit di *OL*, *OFF*, *INT_DW* e *INT_WM*.

7) Sommatore 3 bit: per conteggiare il numero di cicli di clock per l'overload.

Viene qui di seguito inserito lo schema generale del datapath che esemplifica al meglio il funzionamento del circuito:



5.1. Criticità del circuito

Il nostro *DATAPATH* è stato più volte testato al fine di garantire l'effettiva e corretta esecuzione di ogni passaggio. I tester forniti dal Professor Setti, in questo caso specifico, sono stati adattati ai bit richiesti in input dal datapath e l'elaboratore funziona perfettamente.

L'unica criticità si presenta quando viene formata la *FSMD*, in quanto per far comunicare *FSM* e *DATAPATH*, sono necessari dei registri all'interno del *DATAPATH*, che servono per memorizzare i valori di *OL* e *OFF* da restituire alla *FSM* e, successivamente, per effettuare o meno il cambio di stato della macchina e ricevere così i successivi nuovi bit di input per il *DATAPATH*.

La presenza di questi registri crea un ritardo di comunicazione tra elaboratore e controllore: prendiamo in considerazione, ad esempio, il caso in cui la macchina sia in overload per 5 cicli di clock successivi; al sesto ciclo di clock, se la macchina è ancora in overload, sarà spenta, e si porterà quindi dallo stato di *OVERLOAD* a *SPENTO*. (immagine dimostrativa **PUNTO A**)

Riprendendo lo schema della *FSM* presentato nel capitolo 4, notiamo che la macchina passa da *OVERLOAD* a stato *SPENTO* con conseguenti bit di input per il *DATAPATH* [0 0].

Ora, se volessimo far passare la macchina da *SPENTO* direttamente a *OVERLOAD,* quest'ultima dovrà ricevere dalla *FSM* i seguenti bit di input: come *INT_GEN_FSM* = 1 e come *PRIMO_CICLO* = 1.
Tuttavia, essendo il circuito in ritardo di un ciclo di clock, riceverà come bit di input i bit di stato di SPENTO ovvero *INT_GEN_FSM* = 0, *PRIMO_CICLO* = 0, generando cosi degli output sbagliati. (immagine dimostrativa **PUNTO B**).

Abbiamo cercato varie soluzioni concrete per risolvere il problema: per esempio pensare ai bit che la *FSM* genera in output già modificati in funzione del ritardo, ma modificare questi bit, equivaleva poi a ottenere risultati sbagliati in altre situazioni, come nel caso in cui la macchina è in stato *SPENTO* per più di un ciclo di clock consecutivo.

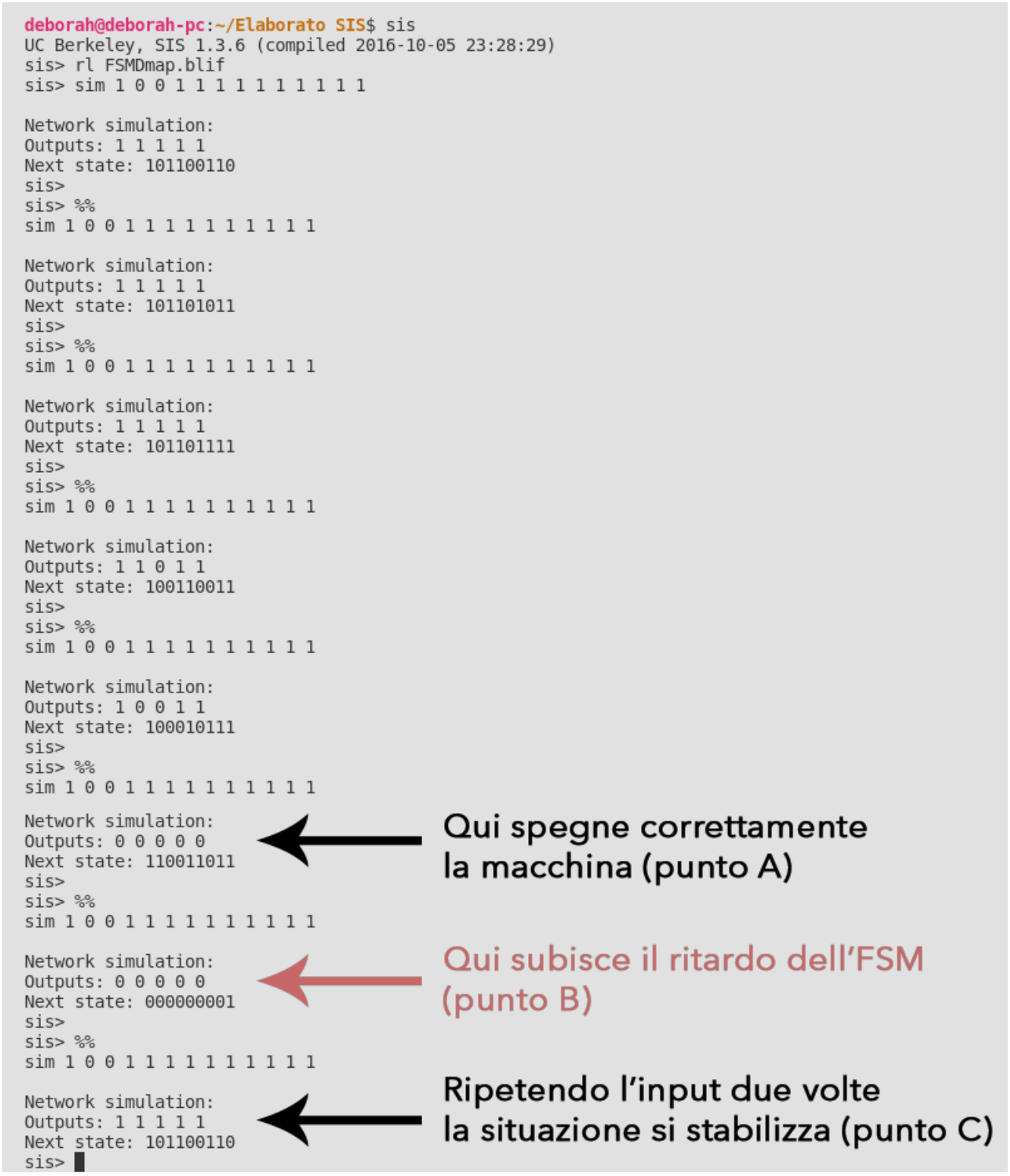
In questa situazione se modificavamo il bit di *PRIMO_CICLO* o *INT_GEN_FSM*, come risultato ottenevamo uno spegnimento effettivo al primo ciclo in cui la macchina è spenta, ma al successivo ciclo la macchina si accedeva anche se doveva rimanere spenta.

Abbiamo provato ad inserire nella *FSM* un bit di output specifico, che cercasse di risolvere questo tipo di problema del ritardo, ma in ogni caso il problema sussisteva.
Infine abbiamo poi provato ad inserire uno stato "intermedio" nella *FSM*, ma rischiava di complicare di più la *FSMD* e non risolveva il problema dei ritardi.

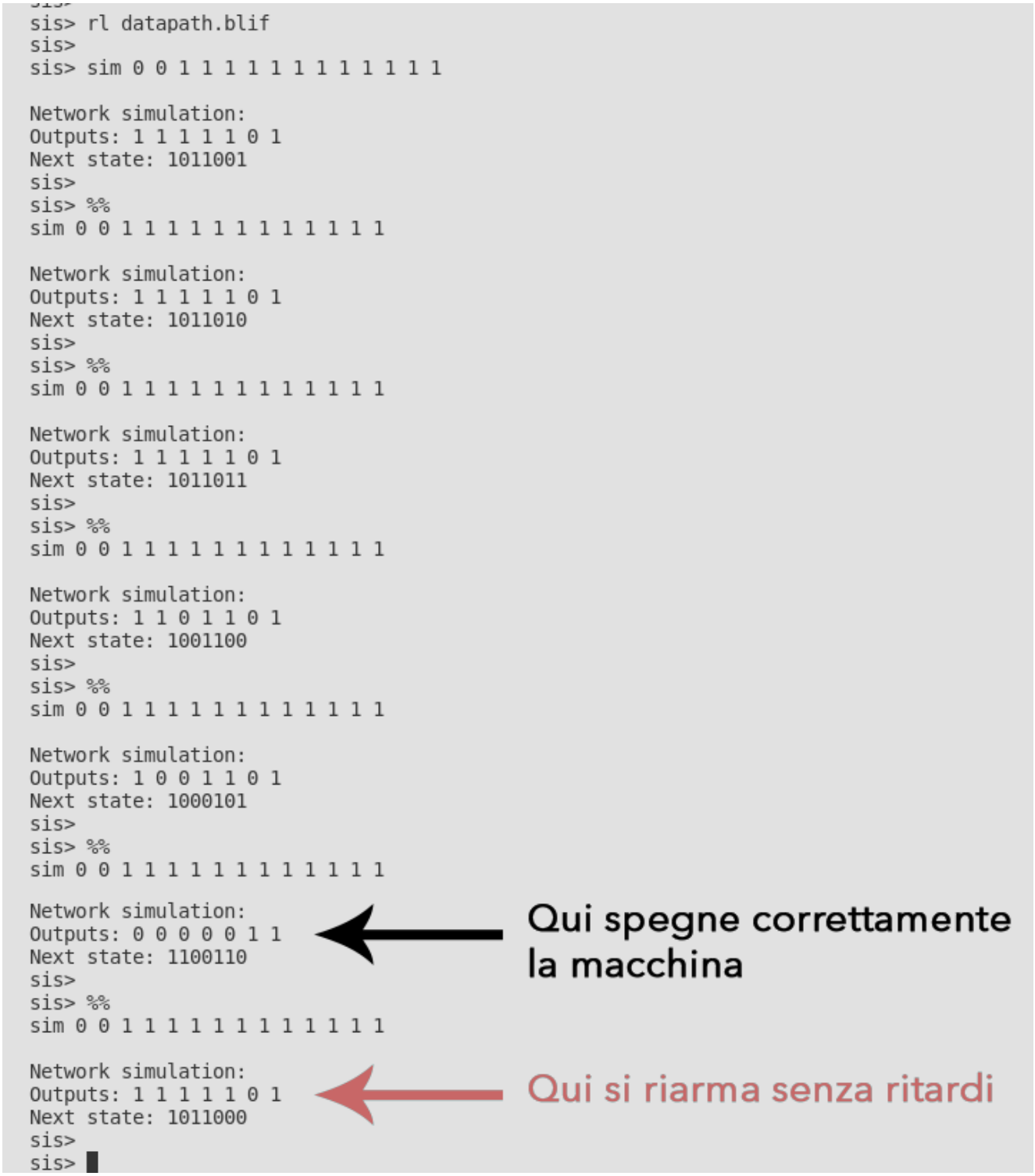
L'unica "soluzione" che abbiamo trovato è quella ripetere due volte la stessa istruzione che genera la criticità, in quanto il circuito ricevendo per due volte gli stessi bit di input riesce a compensare il ritardo e restituire i bit giusti al *DATAPATH* che a sua volta genera gli output corretti. (immagine dimostrativa **PUNTO C**)

5.2. Simulazioni

Vengono qui di seguito inserite delle immagine esemplificative del problema esposto nel paragrafo precedente.



Tuttavia, se il medesimo test viene eseguito semplicemente all'interno del DATAPATH, aggiustando i bit richiesti in input, il risultato è il seguente:



Si può notare dall'immagine precedentemente inserita, che effettivamente il *DATAPATH* esegue correttamente le operazioni richieste nelle specifiche presentate nel capitolo 1.

Ciò nonostante l'utilizzo necessario di registri per permettere la comunicazione con la *FSM* crea quel ritardo che non siamo riusciti a gestire diversamente, se non ripetendo l'istruzione due volte.

Come conferma che il circuito da noi progettato funziona in tutte le altre combinazioni, viene qui di seguito inserita un'immagine con i risultati del file *test_in.txt* fornito dal docente Setti:

```
sis>
sis> rl FSMDmap.blif
sis>
sis>
sis> source test_in.txt

Network simulation:
Outputs: 0 0 0 0 0
Next state: 000000001

Network simulation:
Outputs: 0 0 0 0 0
Next state: 000000001

Network simulation:
Outputs: 1 1 1 0 0
Next state: 001100010

Network simulation:
Outputs: 1 1 1 0 1
Next state: 001100010

Network simulation:
Outputs: 1 1 1 1 0
Next state: 001100010

Network simulation:
Outputs: 1 1 1 1 1
Next state: 101100110

Network simulation:
Outputs: 1 1 1 1 1
Next state: 101101011

Network simulation:
Outputs: 1 1 1 1 1
Next state: 101101111
```

```
Network simulation:
Outputs: 1 1 0 1 1
Next state: 100110011

Network simulation:
Outputs: 1 0 0 1 1
Next state: 100010111

Network simulation:
Outputs: 1 0 0 1 0
Next state: 000000011

Network simulation:
Outputs: 1 1 0 1 1
Next state: 100100110

Network simulation:
Outputs: 1 1 0 1 1
Next state: 100101011

Network simulation:
Outputs: 1 1 0 1 1
Next state: 100101111

Network simulation:
Outputs: 1 1 0 1 1
Next state: 100110011

Network simulation:
Outputs: 1 0 0 1 1
Next state: 100010111

Network simulation:
Outputs: 0 0 0 0 0
Next state: 110011011

Network simulation:
Outputs: 0 0 0 0 0
Next state: 000000001

Network simulation:
Outputs: 1 1 1 1 1
Next state: 101100110

Network simulation:
Outputs: 1 1 1 1 1
Next state: 101101011
sis>
```

Possiamo notare che effettivamente in tutti gli altri casi i test risultano corretti.
L'unica situazione al limite è per l'appunto quella presentata sopra nel paragrafo 5.1.

6. Statistiche del circuito prima e dopo l'ottimizzazione

Vengono ora qui di seguito inserite le immagine relative alle varie ottimizzazioni del circuito diviso nelle varie componenti:

6.1. Minimizzazione stati FSM

Per l'ottimizzazione della FSM abbiamo usato i seguenti comandi:

```
sis>
sis> rl FSMnew.blif
sis> print_stats
FSM             pi= 3   po= 2   nodes=  2       latches= 0
lits(sop)=   0  #states(STG)=   3
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 3
Number of states in minimized machine : 3
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> print_stats
FSM             pi= 3   po= 2   nodes=  4       latches= 2
lits(sop)=  19  #states(STG)=   3
sis> source script.rugged
sis> print_stats
FSM             pi= 3   po= 2   nodes=  3       latches= 2
lits(sop)=  10  #states(STG)=   3
sis> extract_seq_dc

number of latches = 2     depth = 2     states visited =  3
sis> wl FSMmin.blif
sis>
```

6.2. Ottimizzazione DATAPATH

Per l'ottimizzazione del DATAPATH abbiamo usato i seguenti comandi:

```
luca@luca-VirtualBox:~/Scrivania/progetto sis/DATAPATH$ sis
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> rl DATAPATH.blif
Warning: network `selettoref3', node "A0" does not fanout
Warning: network `selettoref3', node "B0" does not fanout
Warning: network `DATAPATH', node "ZERO" is not driven (zero assumed)
Warning: network `DATAPATH', node "t0" does not fanout
Warning: network `DATAPATH', node "t1" does not fanout
Warning: network `DATAPATH', node "t2" does not fanout
Warning: network `DATAPATH', node "t3" does not fanout
Warning: network `DATAPATH', node "t4" does not fanout
Warning: network `DATAPATH', node "t5" does not fanout
Warning: network `DATAPATH', node "t6" does not fanout
Warning: network `DATAPATH', node "t7" does not fanout
Warning: network `DATAPATH', node "t8" does not fanout
Warning: network `DATAPATH', node "t9" does not fanout
sis>
sis>
sis> print_stats
DATAPATH        pi=14   po= 7   nodes=672       latches= 7
lits(sop)=2421
sis> source script.rugged
sis>
sis> print_stats
DATAPATH        pi=14   po= 7   nodes= 96       latches= 7
lits(sop)= 387
sis> wl DATAPATH_min.blif
sis> rl DATAPATH_min.blif
Warning: network `DATAPATH', node "Som_out_79" does not fanout
sis> extract_seq_dc

number of latches = 7     depth = 8     states visited = 32
sis> wl DATAPATH_min.blif
```

È stato necessario utilizzare *extract_seq_dc* a pagina 210 del libro di testo perché compariva il messaggio: "Warning: network 'DATAPATH', node "Som_out_79" does not fanout".

6.3. Minimizzazione area FSMD

Per l'ottimizzazione della FSMD abbiamo usato i seguenti comandi:

```
sis> rl FSMDmin.blif
sis>
sis>
sis> print_stats
FSMD            pi=13   po= 5   nodes= 94       latches= 9
lits(sop)= 387
sis> rlib mcnc.genlib
sis> map -W -m 0 -s
library error: no D-type edge-triggered flip-flops in the library
sis>
sis>
sis> rlib synch.genlib
sis> map -W -m 0 -s
>>> before removing serial inverters <<<
# of outputs:          14
total gate area:       7104.00
maximum arrival time: (62.00,62.00)
maximum po slack:     (-4.00,-4.00)
minimum po slack:     (-62.00,-62.00)
total neg slack:      (-719.20,-719.20)
# of failing outputs:  14
>>> before removing parallel inverters <<<
# of outputs:          14
total gate area:       7056.00
maximum arrival time: (62.00,62.00)
maximum po slack:     (-4.00,-4.00)
minimum po slack:     (-62.00,-62.00)
total neg slack:      (-719.20,-719.20)
# of failing outputs:  14
# of outputs:          14
total gate area:       6416.00
maximum arrival time: (62.00,62.00)
maximum po slack:     (-4.00,-4.00)
minimum po slack:     (-62.00,-62.00)
total neg slack:      (-719.20,-719.20)
# of failing outputs:  14
sis> wl FSMDmap.blif
sis> print_delay
 ... using library delay model
```

7. Numero di gates e ritardo dopo la mappatura

Queste sono le statistiche dopo la mappatura per area eseguita con le istruzioni mostrate nell'immagine precedente:

```
sis> wl FSMDmap.blif
sis> print_delay
 ... using library delay model
RES_GEN    : arrival=( 0.40  0.40) required=(-60.20 -60.20) slack=(-60.60 -60.60)
RES_WM     : arrival=( 0.40  0.40) required=(-53.80 -53.80) slack=(-54.20 -54.20)
RES_DW     : arrival=( 0.40  0.40) required=(-57.20 -57.20) slack=(-57.60 -57.60)
L0         : arrival=( 1.20  1.20) required=(-52.40 -52.40) slack=(-53.60 -53.60)
L1         : arrival=( 1.00  1.00) required=(-53.80 -53.80) slack=(-54.80 -54.80)
L2         : arrival=( 0.80  0.80) required=(-51.20 -51.20) slack=(-52.00 -52.00)
L3         : arrival=( 1.40  1.40) required=(-46.80 -46.80) slack=(-48.20 -48.20)
L4         : arrival=( 0.20  0.20) required=(-57.20 -57.20) slack=(-57.40 -57.40)
L5         : arrival=( 0.20  0.20) required=(-53.80 -53.80) slack=(-54.00 -54.00)
L6         : arrival=( 0.80  0.80) required=(-57.40 -57.40) slack=(-58.20 -58.20)
L7         : arrival=( 0.80  0.80) required=(-57.60 -57.60) slack=(-58.40 -58.40)
L8         : arrival=( 1.00  1.00) required=(-33.80 -33.80) slack=(-34.80 -34.80)
L9         : arrival=( 0.60  0.60) required=(-35.60 -35.60) slack=(-36.20 -36.20)
[4276]     : arrival=( 0.20  0.20) required=(-29.00 -29.00) slack=(-29.20 -29.20)
[4275]     : arrival=( 0.40  0.40) required=(-27.60 -27.60) slack=(-28.00 -28.00)
STATO_PREC_DW: arrival=( 0.20  0.20) required=(-58.60 -58.60) slack=(-58.80 -58.80)
STATO_PREC_WM: arrival=( 0.20  0.20) required=(-55.20 -55.20) slack=(-55.40 -55.40)
REG2       : arrival=( 0.60  0.60) required=(-9.20 -9.20) slack=(-9.80 -9.80)
REG1       : arrival=( 0.20  0.20) required=(-11.00 -11.00) slack=(-11.20 -11.20)
REG0       : arrival=( 0.60  0.60) required=(-9.60 -9.60) slack=(-10.20 -10.20)
LatchOut_v3: arrival=( 0.60  0.60) required=(-61.40 -61.40) slack=(-62.00 -62.00)
LatchOut_v4: arrival=( 0.20  0.20) required=(-26.40 -26.40) slack=(-26.60 -26.60)
[8239]     : arrival=( 3.40  3.40) required=(-44.80 -44.80) slack=(-48.20 -48.20)
[8240]     : arrival=( 4.60  4.60) required=(-33.20 -33.20) slack=(-37.80 -37.80)
[8659]     : arrival=( 5.80  5.80) required=(-32.00 -32.00) slack=(-37.80 -37.80)
[4539]     : arrival=( 4.60  4.60) required=(-37.00 -37.00) slack=(-41.60 -41.60)
[8241]     : arrival=( 6.40  6.40) required=(-35.20 -35.20) slack=(-41.60 -41.60)
[8663]     : arrival=( 7.60  7.60) required=(-32.00 -32.00) slack=(-39.60 -39.60)
[8243]     : arrival=( 2.60  2.60) required=(-44.80 -44.80) slack=(-47.40 -47.40)
[9170]     : arrival=( 4.60  4.60) required=(-43.60 -43.60) slack=(-48.20 -48.20)
[8244]     : arrival=( 6.00  6.00) required=(-42.20 -42.20) slack=(-48.20 -48.20)
[8671]     : arrival=( 7.60  7.60) required=(-40.60 -40.60) slack=(-48.20 -48.20)
[8242]     : arrival=( 2.40  2.40) required=(-52.40 -52.40) slack=(-54.80 -54.80)
[8245]     : arrival=( 4.00  4.00) required=(-49.40 -49.40) slack=(-53.40 -53.40)
[8234]     : arrival=( 2.40  2.40) required=(-45.80 -45.80) slack=(-48.20 -48.20)
[8246]     : arrival=( 3.60  3.60) required=(-44.60 -44.60) slack=(-48.20 -48.20)
[7682]     : arrival=( 4.00  4.00) required=( 0.00  0.00) slack=(-4.00 -4.00)
[8326]     : arrival=(59.00 59.00) required=(-1.20 -1.20) slack=(-60.20 -60.20)
{INT_GEN}  : arrival=(60.20 60.20) required=( 0.00  0.00) slack=(-60.20 -60.20)
{INT_WM}   : arrival=(60.80 60.80) required=( 0.00  0.00) slack=(-60.80 -60.80)
{INT_DW}   : arrival=(59.80 59.80) required=( 0.00  0.00) slack=(-59.80 -59.80)
{TH1}      : arrival=(60.20 60.20) required=( 0.00  0.00) slack=(-60.20 -60.20)
{TH2}      : arrival=(60.20 60.20) required=( 0.00  0.00) slack=(-60.20 -60.20)
sis>
```

8. Scelte progettuali

Le scelte progettuali sono ampiamente discusse nei capitoli precedenti.

Le principali soluzioni adottate sono:

- **Gestire la maggior parte delle operazioni all'interno del DATAPATH,** e utilizzare la *FSM* solamente per inviare bit di controllo.

- **Inserire dei registri per memorizzare lo stato precedente della macchina**, utile per verificare se ad esempio un elettrodomestico il ciclo precedente era acceso o spento e di conseguenza calcolare correttamente la fascia nei punti critici.

- **Conteggiare i cicli di Overload direttamente all'interno del DATAPATH***,* in modo da gestire i ritardi della *FSM*.

- **Inserire il bit di PRIMO_CICLO all'interno della FSM**: utilizzato per calcolare la fascia *DW* e *WM* all'avvio della macchina. Avendo modificato la macchina in modo tale che mantenesse lo stato precedente, senza questo bit avrebbe calcolato male la fascia all'accensione.

- **Utilizzo di porte logiche al posto dei MUX**, al fine di semplificare ulteriormente il *DATAPATH.*

- **Utilizzo del bit di INT_GEN_FSM come fattore di annullamento** del consumo nel caso l'interruttore fosse a 0.