

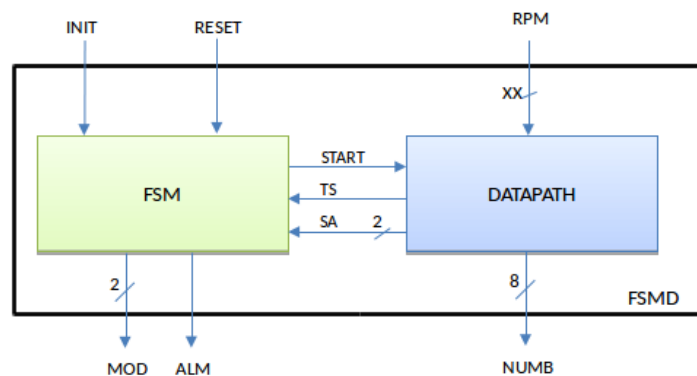
Relazione Elaborato

Architettura degli Elaboratori

Anno Accademico 2015/2016

Breve descrizione:

FSMD di impostazione di soglia RPM motore + allarme fuori giri (dopo 15 cicli di clock)



Studenti

Bertoncelli Giovanni - VR399929

Girelli Alberto - VR397173

Righi Edoardo - VR398499

Sommario

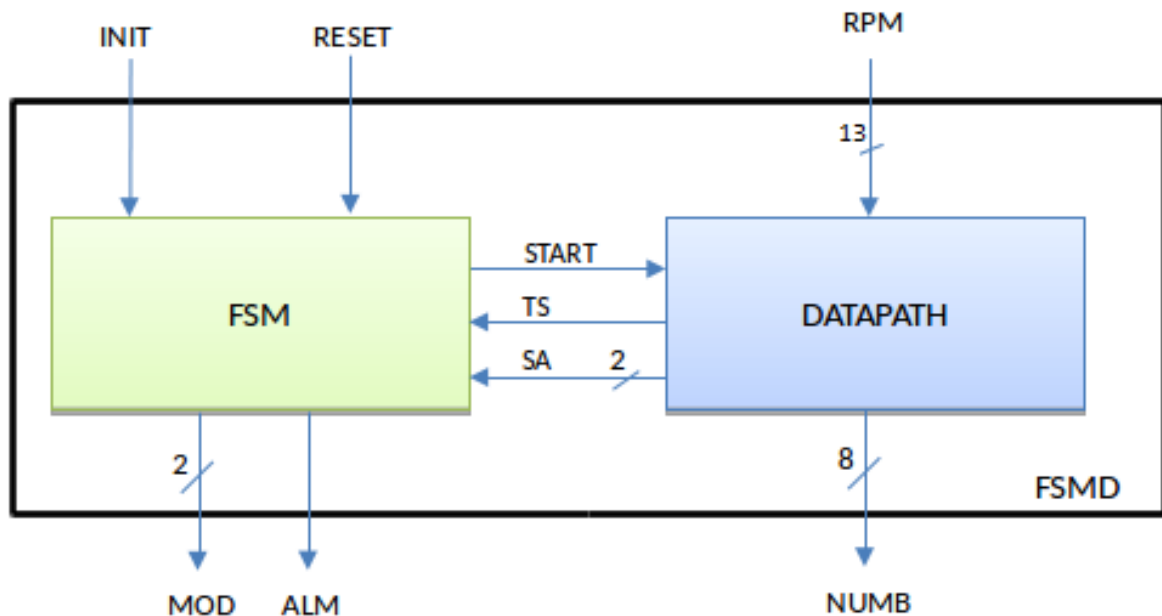
Schema generale del circuito (FSMD).....	3
Controllore (FSM).....	5
Elaboratore (DataPath).....	7
SIS – Valutazione progetto tramite software.....	9
<i>Statistiche del circuito.....</i>	<i>9</i>
<i>Mapping.....</i>	<i>11</i>
<i>Simulazione.....</i>	<i>12</i>
Scelte Progettuali.....	13

Legenda:

INIT → Nome linea di ingresso o uscita utilizzato per la simulazione in SIS

read_blif DATAPATH.blif → Sintassi per comandi in SIS

Schema generale del circuito (FSMD) *finite state machine + datapath*



↑ Fig. 01 Schema generale della FSMD – con entrate e uscite annesse







Il circuito ha come obiettivo quello di monitorare l'andamento dei giri di un motore a combustione interna. Il circuito riceve in ingresso i giri del motore (RPM) su 13 bits che vengono elaborati dal datapath per definire successivamente lo stato di soglia attuale, così suddiviso:

- ⚙ **OFF (OFF):** unicamente quando abbiamo 0 RPM;
- ⚙ **SOTTOGIRI (SG):** Tra 1 e 1999 giri;
- ⚙ **REGIME OTTIMALE (OPT):** Tra 2000 e 4000 giri;
- ⚙ **FUORI GIRI (FG):** Oltre i 4000 giri (con un massimo di 6500 giri);



Inoltre il circuito ha il compito di contare i cicli di clock, corrispondenti ai secondi trascorsi, in cui il motore rimane in un determinato stato di soglia; quindi, quando questo viene alterato, il contatore si resetta. Se il motore rimane nello stato di soglia FG per più di 14 secondi (o meglio 14 cicli di clock) viene dato in uscita un segnale di allarme.

Gli ingressi e le uscite sono elencati e descritti come segue.


Ingressi/uscite principali FSMD

-  **INIT[1]** *INIT*
Quando questo è posto a 1 la macchina restituisce lo stato di soglia e comincia a contare i cicli di clock, e quindi è in funzione. Quando invece è 0, nonostante i secondi trascorsi vengano registrati, non viene restituito né lo stato di soglia attuale né i secondi memorizzati;
-  **RESET[1]** *RESET*
Questo ingresso se vale 1 determina il reset del circuito, quindi il conteggio viene azzerato e viene rivalutato lo stato di soglia attuale. Se è impostato su 0 non influisce sul funzionamento della macchina;
-  **RPM[13]** *RPM12 RPM11 RPM10 RPM9 RPM8 RPM7 RPM6 RPM5 RPM4 RPM3 RPM2 RPM1 RPM0*
Ingresso del numero di giri attuali del motore, basato su 13 bits, per ottenere un massimo di circa 6500 giri/secondo con un certo margine di conteggio;
-  **MOD[2]** *MOD1 MOD0*
È l'uscita che restituisce lo stato di soglia attuale su 2 bits;
-  **ALM[1]** *ALM*
Se restituisce 1 allora il motore è rimasto **fuori giri** per più di 15 secondi;
-  **NUMB[8]** *NUMB7 NUMB6 NUMB5 NUMB4 NUMB3 NUMB2 NUMB1 NUMB0*
Restituisce il numero di secondi trascorsi in un determinato stato di soglia, su 8 bits;

Segnali di stato

-  **TS[1]** *FG*
Segnala il superamento del tempo limite nello stato di soglia FG;
-  **SA[2]** *SA1 SA0*
Restituisce lo stato di soglia attuale da impostare nella FSM;

Segnali di controllo

-  **START[1]** *START*
Messo a 1 fa iniziare al datapath la lettura dei secondi trascorsi altrimenti continua il conteggio dei secondi registrati.

Il circuito quindi per ogni ciclo di clock controlla la soglia attuale, imposta lo stato di soglia corrispondente e inizia il conteggio. Quando cambia stato di soglia il conteggio ricomincia. Se **INIT** è 0 la macchina si pone in uno stato di spegnimento se **RESET** invece vale 1 in contatore viene resettato e lo stato di soglia rivalutato.

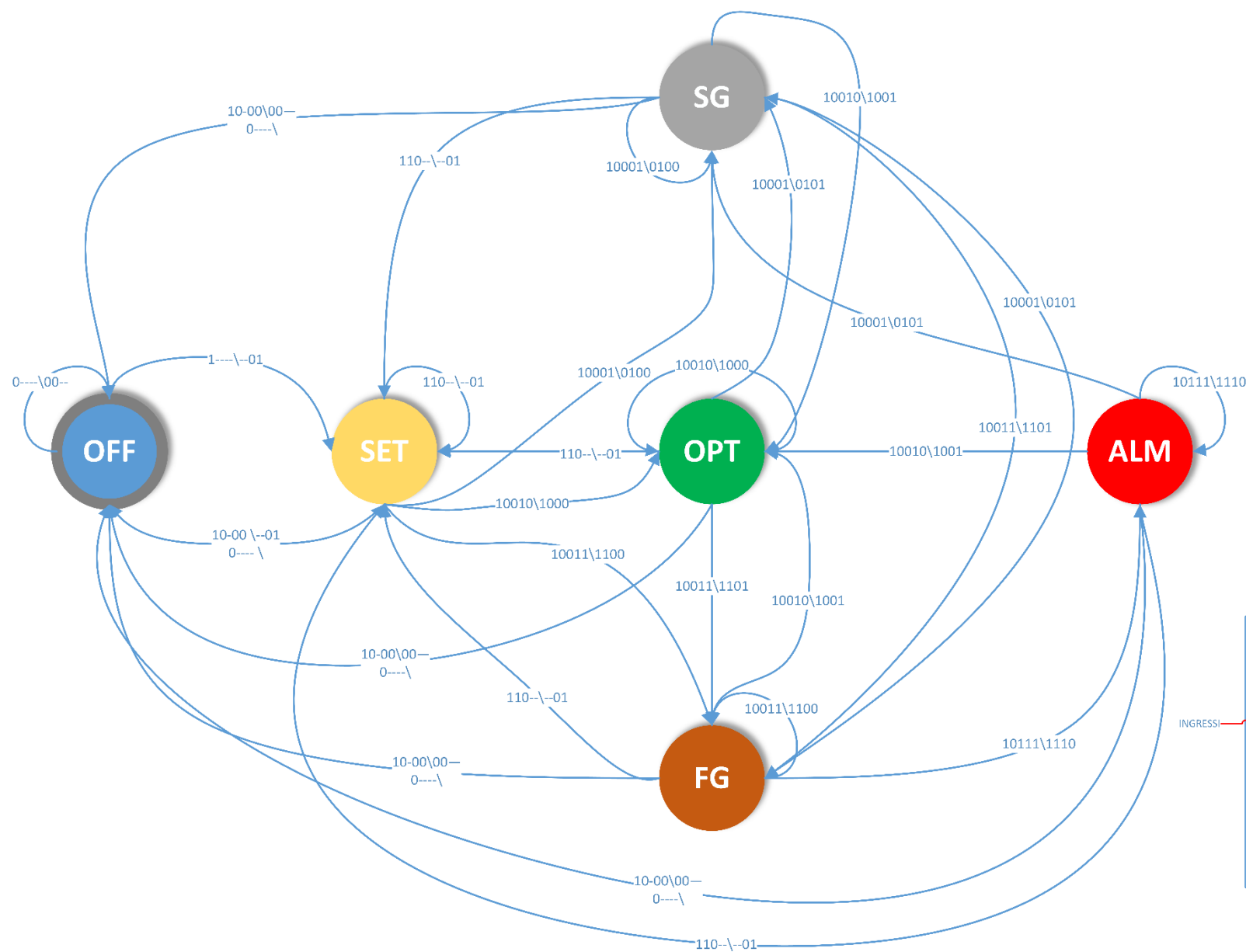
Controllore (FSM) *finite state machine*

Il controllore è una macchina a stati finiti (FSM) di tipo **Mealy** che presenta **quattro ingressi** / **tre uscite** nel seguente ordine: **INIT[1]**, **RESET[1]**, **TS[1]**, **SA[2]** / **MOD[2]**, **ALM[1]**, **START[1]**.

Nonostante l'ottimizzazione tramite SIS riduca di uno il numero degli stati, il nostro progetto prevedeva cinque stati principali:

- ⚙ **OFF**: selezionato come stato di reset, la macchina rimane o ritorna in questo stato se INIT è uguale a 0 oppure (scelta progettuale) mentre il motore restituisce 0 RPM (in questo caso sarà assegnato 00 come stato di soglia attuale);
- ⚙ **SET**: in questo stato la macchina è attiva, quindi INIT è 1, e viene valutato lo stato di soglia attuale. Essendo uno stato puramente di transizione è stato rimosso da SIS. Da questo stato la macchina passerà allo stato di soglia corrispondente al numero degli RPM;
- ⚙ **OPT**: se la macchina è in questo stato allora gli RPM sono tra 2000 e 4000. Da questo stato si può passare agli stati SG o FG se gli RPM cambiano (azzerando il contatore, con START uguale a 1), a SET se RESET uguale a 1, a OFF se INIT vale 0 oppure gli RPM si azzerano. Se gli RPM non cambiano soglia il controllore rimane in questo stato e il contatore continua il conteggio dei secondi trascorsi;
- ⚙ **SG**: se la macchina è in questo stato allora gli RPM sono tra 1 e 1999. Da questo stato si può passare agli stati OPT o FG se gli RPM cambiano (azzerando il contatore, con START uguale a 1), a SET se RESET uguale a 1, a OFF se INIT vale 0 oppure gli RPM si azzerano. Se gli RPM non cambiano soglia il controllore rimane in questo stato e il contatore continua il conteggio dei secondi trascorsi;
- ⚙ **FG**: se la macchina è in questo stato allora gli RPM sono oltre 4000. Da questo stato si può passare agli stati SG o OPT se gli RPM cambiano (azzerando il contatore, con START uguale a 1), a SET se RESET uguale a 1, a OFF se INIT vale 0 oppure gli RPM si azzerano. Se gli RPM non cambiano soglia il controllore rimane in questo stato e il contatore continua il conteggio dei secondi trascorsi. Se rimane in questo stato per più di 15 secondi si passa allo stato di allarme ALM, senza azzerare il conteggio e assegnando all'uscita ALM 1;
- ⚙ **ALM**: la macchina rimane in questo stato solamente se vengono memorizzati più di 15 secondi nello stato FG. Da questo stato si può passare agli stati SG, OPT, SET o OFF, a seconda delle condizioni sopra citate.

Si veda la STG (*state transition graph*) in **Fig. 02** alla pagina seguente.



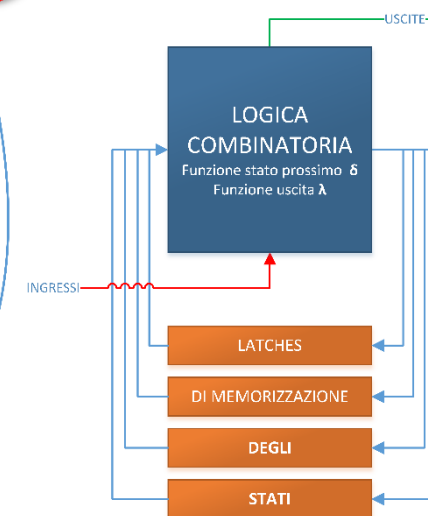
← Fig. 02

STG (state transition graph) FSM

Disegno di descrizione del circuito di controllo

↓ Fig. 02b

Realizzazione "fisica" di una FSM – Modello di Huffman



Elaboratore (DP) *datapath*

Il datapath ha il compito di elaborare le informazioni ricevute dal circuito per ottenere gli output desiderati. In questo caso riceviamo come input principale i 13 bits degli RPM. Come uscita principale invece avremo 8 bits che restituiscono i secondi trascorsi nello stato attuale (NUMB). Come segnali di stato avremo TS e SA[2]. L'unico segnale di controllo è START.

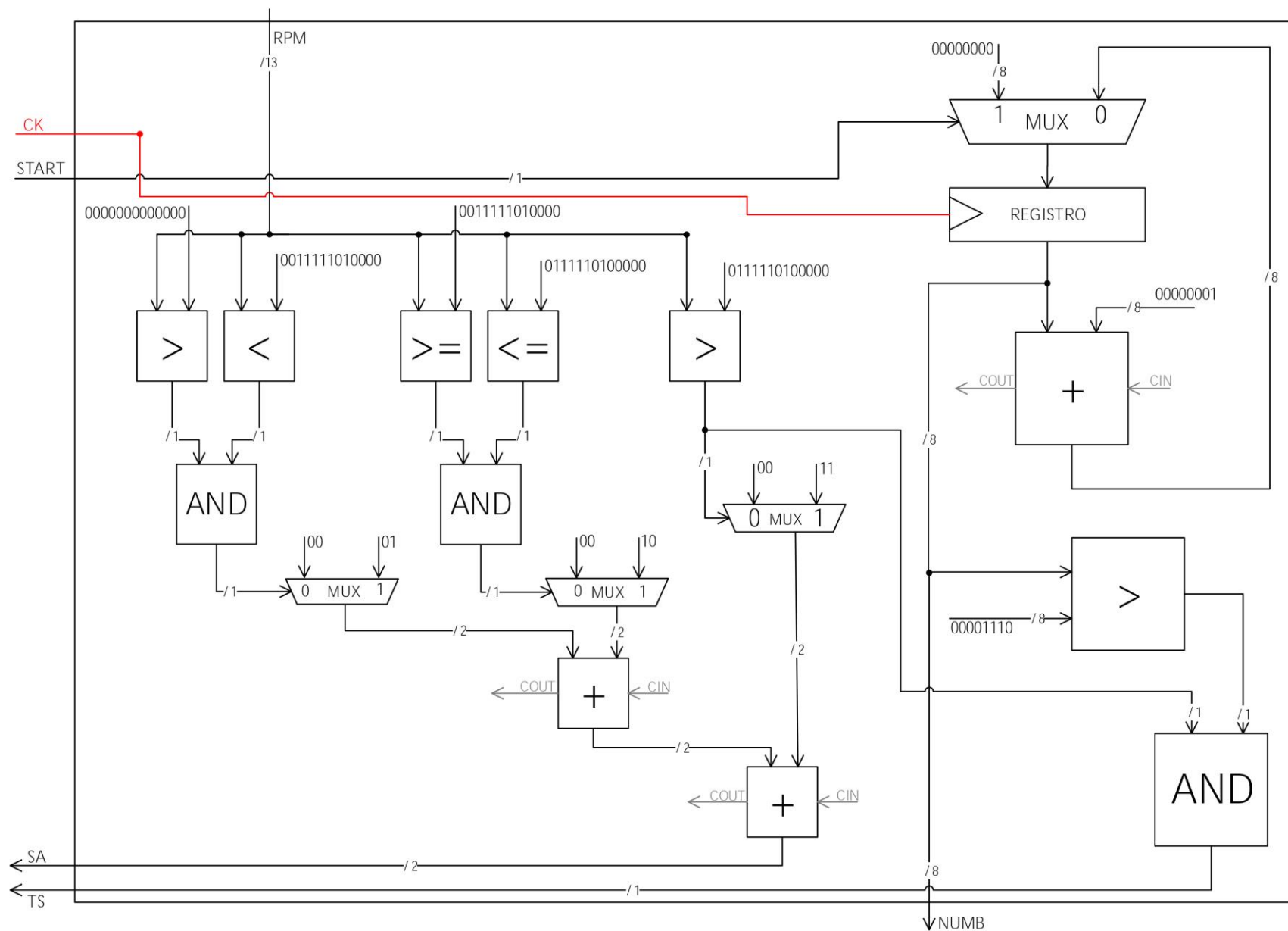
Si veda la Fig. 03 per lo schema generale del datapath, pagina seguente.

Elaborazione degli RPM – valutazione stato di soglia SA

Gli RPM in ingresso vengono inviati a una serie di confrontatori a 13 bits. La prima coppia di confrontatori valuta se gli RPM sono compresi tra 1 e 1999, la seconda coppia se sono compresi tra 2000 e 4000, mentre l'ultimo confrontatore se è maggiore di 4000. Questi valori vengono inseriti su 13 bits come costanti. Le coppie di confrontatori convergono il loro segnale in due differenti porte AND che restituiscono 1 solamente se entrambi i confrontatori valutano vera la condizione corrispondente. I segnali in uscita dalle porte AND e dall'ultimo confrontatore fungono da selettori per tre differenti multiplexer (due ingressi a 2 bits ciascuno). Ogni multiplexer (MUX) riceve in ingresso due costanti: una nulla e una corrispondente allo stato di soglia correlato, selezionato solamente se il selettore è uguale a 1. Si verificherà dunque che, dato un numero di RPM, venga selezionato dai MUX solamente la costante corrispondente allo stato di soglia. I tre segnali di uscita dei differenti MUX vengono sommati tramite due sommatore a 2 bits, generando l'uscita di stato SA.

Conteggio cicli di clock e allarme TS

Un multiplexer, che riceve come selettore il segnale di controllo START, seleziona, se START è uguale a 1, una costante nulla su 8 bits altrimenti il segnale elaborato dal sommatore sottostante. Il valore selezionato dal MUX viene memorizzato in registro con il suo corrispondente segnale di sincronismo. Come uscita dal registro avremo NUMB su 8 bits. Questo segnale viene anche ripreso da un sommatore che gli aggiunge la costante 1, appunto per contare i cicli di clock. L'uscita del sommatore viene posta nell'ingresso 0 del MUX prima citato. Inoltre il segnale NUMB viene ripreso in un confrontatore e viene valutato se esso è maggiore di 14 (scelta progettuale). Infine il risultato del confrontatore converge in una porta AND assieme al risultato del confronto $RPM > 4000$. Se entrambi danno 1, allora TS è uguale a 1.



← Fig. 03

DP (datapath)
 Disegno di
 descrizione del
 circuito di
 elaborazione

SIS - Valutazione del progetto tramite software

Per simulare il circuito abbiamo utilizzato il software in dotazione UC Berkeley, SIS 1.3.6 su sistema operativo *Linux-based Ubuntu 15.10* (64 bit). Per realizzare la FSM ci siamo serviti di BVE.

Statistiche del circuito

FSM

Abbiamo anzitutto provveduto a inserire da terminale il file FSM.blif della macchina a stati finiti nella shell di SIS; come da Fig. 04 abbiamo cercato con successo di ridurre il numero degli stati tramite l'algoritmo offerto dal comando `state_minimize stamina`, che ci ha permesso di ridurre di uno il numero degli stati (da 6 a 5). Successivamente abbiamo inizializzato il nostro file blif creando le opportune codifiche degli stati e le funzioni λ e δ tramite il comando `state_assign jedi`. Solo dopo queste operazioni (obbligatorie per il funzionamento) abbiamo potuto visualizzare le prime statistiche tramite il comando `print_stats`. Da questa prima analisi di SIS abbiamo notato che il nostro circuito constava di **5 stati**, 7 nodi, 3 memorie e 39 letterali.

```
sis> read_blif FSM.blif
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 6
Number of states in minimized machine : 5
sis> state_assign jedi
Running jedi, written by Bill Lin, UC Berkeley
sis> print_stats
FSM          pi= 5   po= 4   nodes= 7       latches= 3
lits(sop)= 39 #states(STG)= 5
```

↑ Fig. 04 FSM: lettura del file, minimizzazione e assegnamento degli stati, prime statistiche

Il passaggio successivo è stato quello dell'ottimizzazione del circuito combinatorio interno alla FSM tramite le operazioni eseguite dallo script `script.rugged` (Fig. 05):

```
sis> print_stats
FSM          pi= 5   po= 4   nodes= 7       latches= 3
lits(sop)= 39 #states(STG)= 5
sis> source script.rugged
sis> print_stats
FSM          pi= 5   po= 4   nodes= 8       latches= 3
lits(sop)= 28 #states(STG)= 5
```

↑ Fig. 05 FSM: Esecuzione del comando `source script.rugged` e ottimizzazione combinatoria

È sicuramente da notare la riduzione dei letterali da 39 a 28 con l'aumento dei nodi da 7 a 8.

DATAPATH

Come per la FSM anche per il datapath è stata eseguita la lettura del file corrispondente e la sua ottimizzazione (Fig. 06).

```
sis> read_blif DATAPATH.blif
Warning: network `DATAPATH', node "COUT1" does not fanout
Warning: network `DATAPATH', node "COUT0" does not fanout
Warning: network `DATAPATH', node "COUT2" does not fanout
sis> print_stats
DATAPATH      pi=14   po=11   nodes=206     latches= 8
lits(sop)=1088
```

↑ Fig. 06 DP: Lettura del file DATAPATH.blif e primissime statistiche

I segnali di errori datati da SIS non sono preoccupanti dal momento che, per scelte progettuali i COUT* non sono stati collegati ad alcuna uscita, essendo i riporti dei sommatori (per dettagli vedere sezione *Scelte Progettuali*). Le prime statistiche, abbastanza elevate, ci riportano un numero di **letterali pari a 1088 e di nodi pari a 206**. L'ottimizzazione tramite lo script `script.rugged` migliora di molto la situazione (Fig. 07):

```
sis> source script.rugged
sis> print_stats
DATAPATH      pi=14   po=11   nodes= 16     latches= 8
lits(sop)= 118
```

↑ Fig. 07 DP: Ottimizzazione tramite script.rugged

Abbiamo infatti ottenuto una notevole riduzione dei **letterali da 1088 a 118** e dei **nodi da 206 a 16**.

FSMD

Per ultimo abbiamo caricato la FSMD, stampato le sue statistiche e ottimizzato ulteriormente il circuito questa volta completo (Fig. 08):

```
sis> read_blif FSMD.blif
sis> print_stats
FSMD_RPM      pi=15   po=11   nodes= 24     latches=11
lits(sop)= 146
sis> source script.rugged
sis> print_stats
FSMD_RPM      pi=15   po=11   nodes= 23     latches=11
lits(sop)= 149
```

↑ Fig. 08 FSMD: Ottimizzazione tramite script.rugged

Questa volta abbiamo ottenuto una riduzione dei nodi a discapito di tre letterali in più.

Mapping

L'operazione di *Mapping* ci permette, tramite un'apposita libreria di componenti elettronici già semi-assemblati, di ottenere una simulazione fisica di come si dovrebbe presentare il circuito, stimando quindi con più precisione il ritardo dovuto alle varie componenti e lo spazio occupato. Tramite SIS abbiamo utilizzato la libreria `synch.genlib` e mappato il nostro circuito con relativo ritardo e area (**Fig. 09**):

```

sis> read_library synch.genlib
sis> map -s
warning: unknown latch type at node '{{[347]}}' (RISING_EDGE assumed)
warning: unknown latch type at node '{{[348]}}' (RISING_EDGE assumed)
warning: unknown latch type at node '{{[349]}}' (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:          22
total gate area:       3000.00
maximum arrival time:  (24.00,24.00)
maximum po slack:      (-0.60,-0.60)
minimum po slack:      (-24.00,-24.00)
total neg slack:       (-214.80,-214.80)
# of failing outputs:  22
>>> before removing parallel inverters <<<
# of outputs:          22
total gate area:       2984.00
maximum arrival time:  (21.60,21.60)
maximum po slack:      (-0.60,-0.60)
minimum po slack:      (-21.60,-21.60)
total neg slack:       (-212.40,-212.40)
# of failing outputs:  22
# of outputs:          22
total gate area:       2904.00
maximum arrival time:  (21.60,21.60)
maximum po slack:      (-0.60,-0.60)
minimum po slack:      (-21.60,-21.60)
total neg slack:       (-212.40,-212.40)
# of failing outputs:  22

```

↑ Fig. 09 MAPPING: importazione della libreria e statistiche di mappatura

La mappatura del nostro circuito ci ha permesso di ottenere i seguenti risultati: cammino critico di 24.00 (*maximum arrival time*) e area totale a 3000.0 (*total gate area*). I *warning* presenti sono gli adattamenti che SIS ha dovuto effettuare per permettere una corretta mappatura del circuito.

Simulazione

Abbiamo voluto lasciare dello spazio nella nostra relazione anche per dimostrare il funzionamento del circuito tramite la simulazione in SIS (**Fig. 10**):

```
simulate * * * * * * * * * * * * * *
```

Ricordiamo l'ordine degli ingressi:

INIT RESET RPM12 RPM11 RPM10 RPM9 RPM8 RPM7 RPM6 RPM5 RPM4 RPM3 RPM2 RPM1 RPM0

E delle uscite:

MOD1 MOD0 ALM NUMB7 NUMB6 NUMB5 NUMB4 NUMB3 NUMB2 NUMB1 NUMB0

```
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 0 0
Next state: 00000001110
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 0 0 1
Next state: 00000010110
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 0 1 0
Next state: 00000011110
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 0 1 1
Next state: 00000100110
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 1 0 0
Next state: 00000101110
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 1 0 1
Next state: 00000110110
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 1 1 0
Next state: 00000111110
```

Simulazione del conteggio dei cicli di clock –
da notare le ultime cifre degli outputs si
incrementano ogni volta di 1

```
sis> simulate 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Network simulation:
Outputs: 0 0 0 0 0 0 0 0 0 0 0 0
Next state: 00000000011
```

Simulazione dello stato di OFF 00 / RPM = 0

```
sis> simulate 1 0 0 0 1 0 1 1 1 0 1 1 1 0 0
Network simulation:
Outputs: 0 1 0 0 0 0 0 0 0 0 0 0
Next state: 00000001101
```

Simulazione dello stato di SG 01 / RPM = 1500

```
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
Network simulation:
Outputs: 1 0 0 0 0 0 0 0 0 0 0 1
Next state: 00000000110
```

Simulazione dello stato di OPT 10 / RPM = 3000

```
sis> simulate 1 0 1 0 0 1 1 1 0 0 0 1 0 0 0
Network simulation:
Outputs: 1 1 0 0 0 0 0 0 0 1 1 1
Next state: 00000000100
sis> simulate 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0
```

Simulazione dello stato di FG 11 / RPM = 5000

```
Network simulation:
Outputs: 1 1 1 0 0 0 0 1 1 1 1 1
Next state: 00010000100
```

Simulazione dello stato di ALLARME
RPM = 5000 / NUMB= 15

↑ Fig. 10 Simulazioni varie del circuito in SIS

Scelte Progettuali - conclusioni

Durante l'implementazione di questo circuito ci siamo spesso posti di fronte a delle scelte, magari per particolarità mancanti nelle specifiche o per gusto personale o per efficienza del progetto.

Elenchiamo le principali:

- ⚙ Essendoci nelle specifiche un apposito stato di OFF 00, abbiamo pensato di realizzare lo stato corrispondente nella FSM e per convenzione di sceglierlo solamente se INIT fosse stato uguale a 0 oppure se il DP avesse dato 0 RPM. Questo ha facilitato anche le condizioni di progettazione del datapath da momento che, se tutte le uscite dei confrontatori risultano false, allora successivamente si sommerebbero zeri a zeri per poi concludere con l'uscita SA=00;
- ⚙ Abbiamo scelto di inserire i riporti per i sommatore anche se questi non sarebbero stati associati a nessuna uscita, generando così il warning *does not fanout*. Questo warning è poi scomparso dopo le varie ottimizzazioni;
- ⚙ Nella FSM non abbiamo preso in considerazione in fatto che un motore non possa passare da fuori giri a sotto giri senza transitare nel regime ottimale (o viceversa). Questo tuttavia non contribuisce eccessivamente al dispendio del circuito occupando solamente alcuni archi di transizione;
- ⚙ Nonostante il progetto iniziale lo avesse previsto, non abbiamo inserito un registro per la memorizzazione degli RPM. Durante le prime simulazioni sperimentali del circuito abbiamo infatti notato che questo provocava un notevole ritardo nell'identificazione dello stato di soglia oltre che un dispendio materiale maggiore e altre (gravi) disfunzioni per la macchina. Il registro quindi è stato rimosso;
- ⚙ Il registro dei secondi trascorsi ritarda di un ciclo di clock le risposte a segnali come RESET o START. Come si può immaginare quindi, nella simulazione di SIS se RESET è posto a 1 allora solamente nello stato prossimo il contatore verrà azzerato; lo stesso accade anche per la transizione da uno stato di soglia a un altro;
- ⚙ Nella realizzazione della FSM abbiamo realizzato gli stati SET e OFF in modo separato, anche se (come avevamo intuito) SIS li ha ridotti ad uno solo. Questo è stato fatto per distinguere meglio lo stato di OFF (con RPM a 0 o INIT a 0) e lo stato di transizione SET, stato di valutazione o di rivalutazione (nel caso RESET fosse stato uguale a 1) dello stato di soglia attuale.
- ⚙ Essendo stato posto dalle specifiche un massimo di 6500 RPM abbiamo utilizzato una codifica a 13 bit, ottenendo comunque un margine di circa 2000 RPM (con 13 bit possiamo arrivare a 8191).
- ⚙ Nelle specifiche quando START valeva 1 bisognava iniziare il conteggio e anche a valutazione degli RPM in ingresso. Per semplicità nel datapath abbiamo affidato questo compito alla FSM che con START uguale a 1 avrebbe azzerato solamente il contatore e dato stato di soglia nullo. Quindi non dovendo fare ottimizzazioni di risparmio energetico la valutazione degli RPM nel datapath viene effettuata in continuazione, mentre è la FSM a annullare lo stato di soglia su MOD* se necessario.
- ⚙ Abbiamo scelto come convenzione di rendere ALM uguale a 1 solamente dopo 14 cicli di clock (quindi all'incirca 15 secondi come da specifiche).