RELAZIONE ELABORATO SIS

Architettura degli Elaboratori

A.A. 2015-2016

Daniele Capponi – VR380495
Abdelkader Yärdi – VR380687

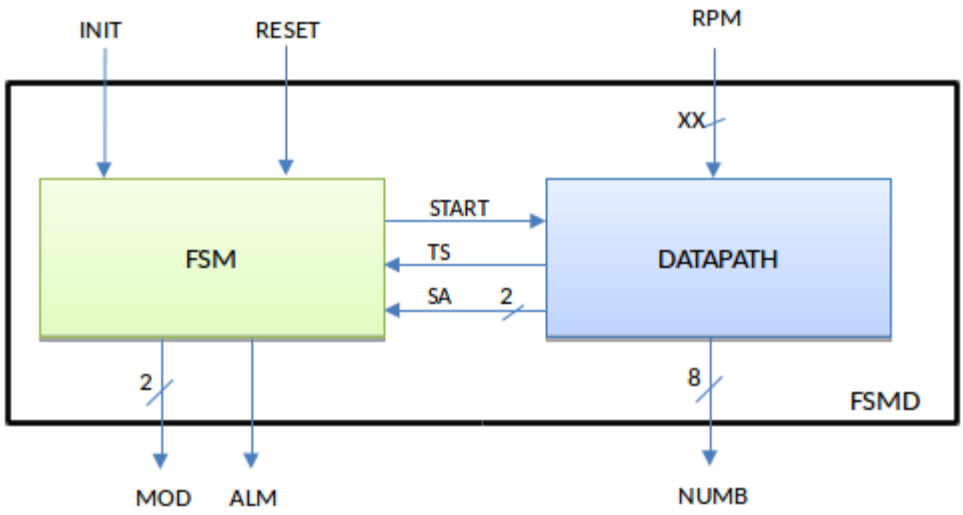
Indice

Specifiche del circuito

Il dispositivo da noi implementato serve come monitoraggio del funzionamento di un motore a combustione di un camion; è quindi basato su un circuito sequenziale che riceve come input il numero di giri al minuto del motore (RPM) e, date delle soglie minime e massime per il funzionamento ottimale, fornisce in uscita una modalità di funzionamento dello stesso: sotto-giri (SG), ottimale (OPT) o fuori-giri (FG). Si vuole inoltre avere un'uscita che rappresenta il tempo trascorso nello stato di "soglia" attuale ed un ulteriore output d'allarme che vale 1 se e soltanto se il sistema è in stato di fuori-giri da più di 15 secondi (cicli di clock).

Il circuito è composto da un controllore e da un datapath con i seguenti ingressi e uscite:

- INIT[1]: quando vale 1 indica che il circuito ha iniziato la rilevazione del numero di giri. Il controllore deve passare nello stato di conteggio dei secondi, trascorsi nell'attuale modalità. Finché vale 0 non deve essere fatto alcun conteggio né indicato alcun valore in uscita.
- RESET[1]: se posto a 1 il controllore deve essere resettato, ovvero il contatore dei secondi deve essere posto a 0.
- RPM[XX]: valore del numero di giri ricevuto dal rilevatore (massimo 6500).
- MOD[2]: indica in quale modalità di funzionamento si trova l'apparecchio al momento (00 – spento, 01 – SG, 10 – OPT, 11 – FG).
- NUMB[8]: indica i secondi trascorsi nell'attuale modalità.
- ALM[1]: segnala il superamento del tempo limite in FG.
- Il controllore è collegato al datapath con tre segnali che hanno il seguente significato:
  - START[1]: messo a 1 fa iniziare al datapath la lettura dei RPM, il set dello stato di soglia e l'inizio del conteggio dei secondi.
  - SA[2]: imposta il controllore nello stato di soglia attuale.
  - TS[1]: segnala al controllore il superamento della soglia di tempo.



Dal momento che il rilevatore manda un valore RPM al secondo, questo deve essere dunque simulato e deve far parte degli ingressi impostati durante la simulazione. Ad ogni valore RPM in ingresso, il circuito controlla l'attuale soglia, imposta lo stato corrispondente e inizia a contare o, nel caso in cui la soglia sia stazionaria, incrementa il valore attuale del contatore. Nel momento in cui il valore RPM non fa parte della soglia attualmente impostata, il circuito cambia stato e inizia da zero il conteggio. Ad ogni inserimento di RPM, se INIT è attivo e non vi è reset, il circuito riporta i valori aggiornati delle uscite. I valori delle soglie sono i seguenti:

- RPM < 2000 → SG
- 2000 ≤ RPM ≤ 4000 → OPT
- RPM > 4000 → FG

Progetto Controllore

Il controllore è una macchina a stati finiti (FSM) di Mealy che presenta cinque ingressi (INIT, RESET, TS, SA2, SA1) e quattro uscite (START, MOD2, MOD1, ALM).

Sono stati individuati quattro stati che verranno di seguito brevemente descritti:

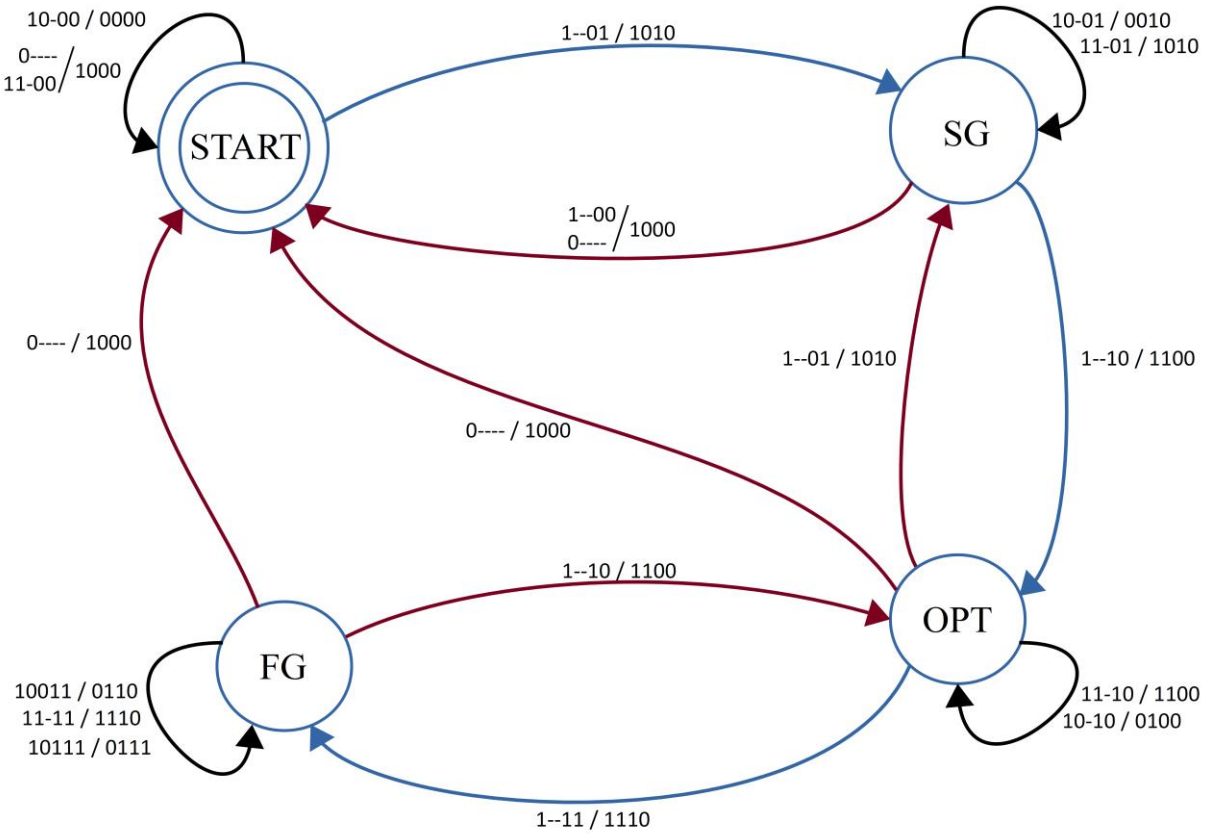
- **START**: stato di reset. La FSM resta in questo stato o fintanto che il segnale di INIT è posto a 0, o quando l'input dei RPM ha come valore 0. Rappresenta quindi lo stato in cui il veicolo è spento oppure quello in cui questo non ha ancora acceso il motore.

- **SG**: stato di "sotto-giri". La FSM arriva in questo stato quando (a patto che INIT sia posto a 1) il datapath rileva un numero di giri superiore a 0 e inferiore a 2000; resta in questo stato se la soglia calcolata dal datapath rimane entro questi due valori.

- **OPT**: stato "ottimale". La FSM arriva in questo stato quando (a patto che INIT sia posto a 1) il datapath rileva un numero di giri superiore a 1999 e inferiore a 4001; resta in questo stato se la soglia calcolata dal datapath rimane entro questi due valori.

- **FG**: stato "fuori-giri". La FSM arriva in questo stato quando (a patto che INIT sia a posto a 1) il datapath rileva un numero di giri superiore a 4000; resta in questo stato se la soglia calcolata dal datapath rimane oltre questo valore. Se l'input TS dovesse segnalare il superamento dei 15 secondi nello stato attuale, l'output ALM verrà posto a 1.

Di seguito, lo State Transition Graph del controllore.



Progetto Datapath

Il datapath è quella parte del circuito che si occupa principalmente di eseguire i calcoli.

Nel nostro caso riceve un input (RPM) codificato su 14 bit, che corrisponde al numero di giri del motore, e un input (START) proveniente dal controllore. In output vi sono SA (2 bit), indicatore dello stato di soglia, TS (1 bit), indicatore di superamento dei 15 secondi, e infine NUMB (8 bit) che mostra il numero dei secondi trascorsi nella soglia corrente.

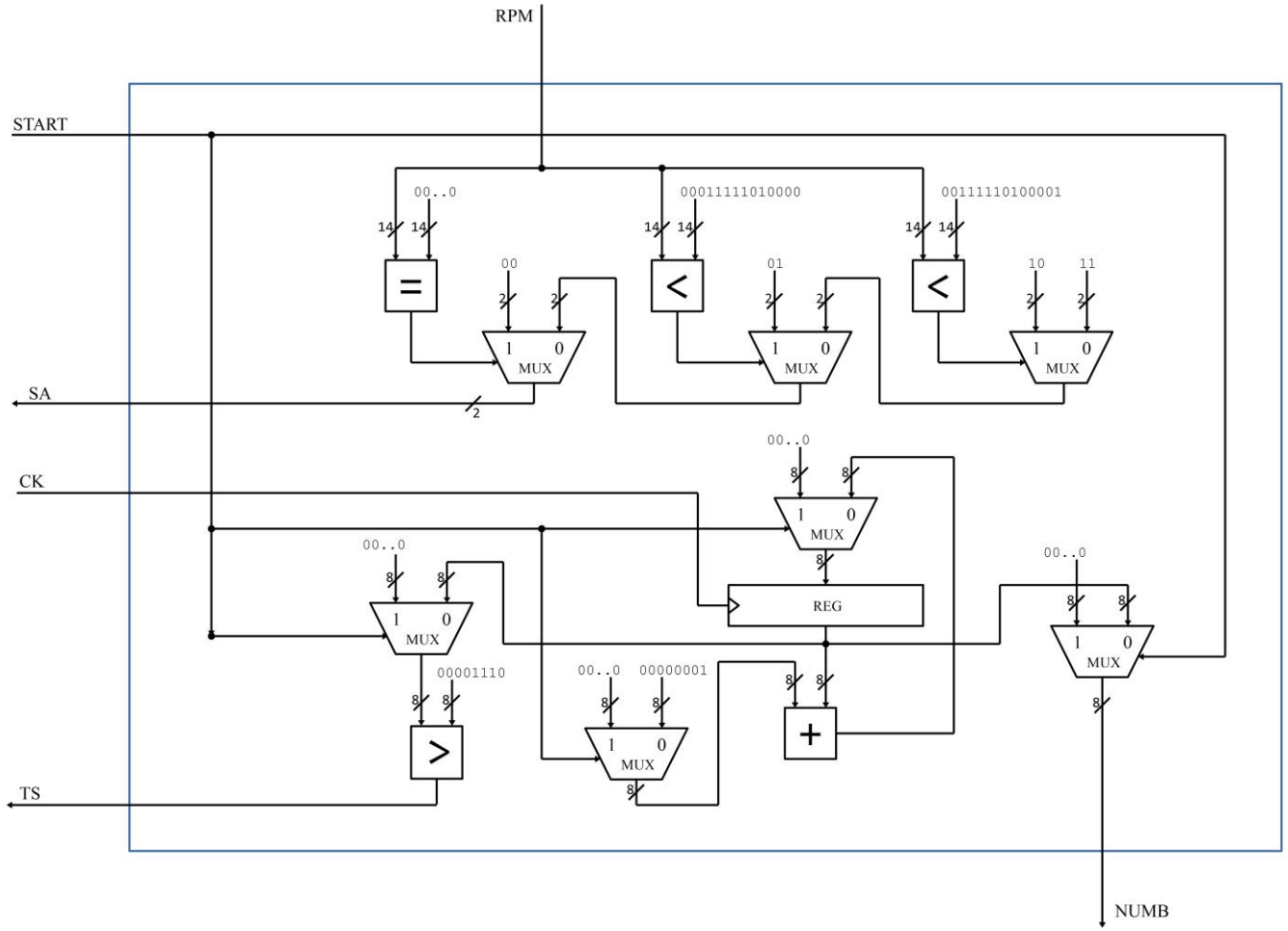
La prima parte del circuito si occupa di calcolare lo "stato di soglia" attuale. Inizialmente i 14 bit relativi ai RPM vengono immessi nel circuito e vengono parallelamente confrontati con i valori delle soglie per ottenere l'uscita 'SA'. A questo proposito utilizziamo tre comparatori: il primo di uguaglianza, il secondo e il terzo di minoranza; il primo confronta i 14 bit sopra specificati con il valore 0, il secondo confronta gli stessi con il valore 2000, l'ultimo con il valore 4001. Il bit risultante dal terzo comparatore farà da selettore di un multiplexer che metterà sulla propria uscita 10 (codifica per "ottimale") se il confronto va a buon fine, altrimenti 11 (codifica per "fuori-giri"); i due bit di segnale appena ottenuti saranno input del multiplexer, insieme all'input 01 (codifica per "sotto-giri"), che ha come selettore il bit risultante dal secondo comparatore; i due bit di segnale appena ottenuti verranno messi in input ad un altro multiplexer, assieme a 00, che ha come selettore il bit risultante dal primo comparatore; i due bit risultanti comporranno l'uscita 'SA' del datapath. La parte del circuito appena descritta non dipende dall'input 'START' proveniente dal controllore, perché l'uscita che produce interessa solo controllore stesso.

La seconda parte del circuito si occupa di produrre l'output a 8 bit 'NUMB' (numero dei secondi) e 'TS' (posto a 1 se NUMB è maggiore o uguale a 15). A questo proposito viene utilizzato un registro, inizialmente settato al valore 0, che viene incrementato di uno ad ogni ciclo di clock fintanto che START vale 0; l'input appena citato infatti, quando vale 1 seleziona il valore 0 di un multiplexer che verrà caricato nel registro, altrimenti seleziona il valore uscente da un sommatore, il quale a sua volta ha come input il valore contenuto nel registro e l'uscita di un altro multiplexer, il quale seleziona il valore 0 se START vale 1, o 1 altrimenti.

Il contenuto del registro viene posto su una linea di input di un multiplexer assieme all'input 0; se START vale 1 in uscita vi sarà quindi 0, altrimenti l'uscita 'NUMB' mostrerà il valore contenuto nel registro.

Nel frattempo l'uscita del registro è collegata a quella parte del circuito che si occupa di verificare che il valore contenuto al suo interno sia maggiore di 14. A questo scopo ci serviamo di un multiplexer che ha come input il valore 0 (posto in uscita se il selettore START vale 1) e il valore in uscita del registro (posto in uscita se START vale 0); il risultato di questa selezione sarà il primo input di un comparatore di maggioranza, mentre il secondo avrà valore 14 appunto. La linea d'uscita da questo componente costituirà l'output TS.

Di seguito lo schema del circuito del datapath.



Statistiche del circuito

Come prima cosa abbiamo minimizzato la FSM col comando "`state_minimize stamina`" e dopo aver assegnato una codifica binaria agli stati con "`state_assign nova`", abbiamo ottenuto le seguenti statistiche.

```
sis> read_blif controller.blif
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 4
Number of states in minimized machine : 2
sis> state_assign nova
Running nova, written by Tiziano Villa,  UC Berkeley
sis> print_stats
controller      pi= 5   po= 4   nodes=  5       latches= 1
lits(sop)=  18  #states(STG)=   2
```

A questo punto ottimizziamo il circuito del controllore, ottenendo tre letterali in meno.

```
sis> print_stats
controller      pi= 5   po= 4   nodes=  4       latches= 1
lits(sop)=  15  #states(STG)=   2
```

Facciamo lo stesso con il circuito del datapath. Quest'ultimo viene descritto nel file "datapath.blif" in cui sono linkati e utilizzati i file di tutti i suoi componenti. Dopo aver provveduto alla minimizzazione quindi, in "main.blif" verrà linkato il nuovo file che descrive il datapath chiamato "minimized_datapath.blif", creato automaticamente tramite il comando "`write_blif`". Di seguito le statistiche pre e post ottimizzazione:

```
sis> read_blif datapath.blif
sis> print_stats
datapath        pi=15   po=11   nodes=147       latches= 8
lits(sop)= 750
```

```
sis> source script.lits; print_stats
datapath        pi=15   po=11   nodes= 21       latches= 8
lits(sop)= 102
```

Il notevole miglioramento è stato ottenuto grazie all'esecuzione dello script creato ad hoc "script.lits".

Caricando in memoria il file principale (main.blif) che costruisce il circuito finale del nostro dispositivo, abbiamo l'opportunità di fare qualche simulazione.

Simulazioni

Per controllare il corretto funzionamento del nostro dispositivo simuliamo degli input "chiave" per il nostro circuito.

La prima immagine mostra la situazione in cui siamo nello stato di soglia sotto-giri (01) da quattro secondi, e arriva il segnale di RESET posto a 1, che produce l'azzeramento del registro. La seconda invece mostra gli output prodotti dall'arrivo del segnale INIT posto a 0.

```
sis> simulate 1 0 0 0 0 0 0 1 0 1 1 1 0 1 1 0

Network simulation:
Outputs: 0 1 0 0 0 0 0 0 1 0 0
Next state: 000001001
sis> simulate 1 1 0 0 0 0 0 1 0 1 1 1 0 1 1 1

Network simulation:
Outputs: 0 1 0 0 0 0 0 0 0 0 0
Next state: 000000001
```

```
sis> simulate 0 0 0 0 1 1 1 1 1 0 1 0 1 0 0 1

Network simulation:
Outputs: 0 0 0 0 0 0 0 0 0 0 0
Next state: 000000000
```

Per simulare il funzionamento del dispositivo in maniera esaustiva è stato creato uno script ad hoc chiamato "simulation" in cui vengono proposti al circuito degli input progressivi a partire dallo stato di accensione della centralina del motore, passando dall'accensione dello stesso e dal suo stato di esercizio normale, arrivando infine allo stato di spegnimento. Abbiamo quindi fatto in modo che il motore passi attraverso tutte le modalità, e durante il suo funzionamento arrivino anche i segnali di reset del conteggio dei secondi e di spegnimento del dispositivo.

Mapping Tecnologico

Una volta che il circuito del nostro dispositivo è stato ottimizzato, ne facciamo il mapping utilizzando la libreria *synch.genlib*, con lo scopo di ottenere delle statistiche verosimili di area e ritardo.

```
sis> map -W -m 0 -s
>>> before removing serial inverters <<<
# of outputs:          20
total gate area:       2728.00
maximum arrival time: (29.00,29.00)
maximum po slack:     (-7.00,-7.00)
minimum po slack:     (-29.00,-29.00)
total neg slack:      (-394.80,-394.80)
# of failing outputs:  20
```

Possiamo notare che da una prima mappatura, volta ad ottimizzare l'area piuttosto che il ritardo, otteniamo che il total-gate-area ha valore 2728, mentre il cammino critico 29.

Attuando delle ottimizzazioni per area riusciamo a ridurne il valore fino a 2648 mentre aumentiamo il cammino critico (32.40). Per fare ciò abbiamo uno script creato ad hoc chiamato "script.mapped"; di seguito la mappatura dopo la sua esecuzione.

```
sis> source script.mapped
>>> before removing serial inverters <<<
# of outputs:          20
total gate area:       2648.00
maximum arrival time: (32.40,32.40)
maximum po slack:     (-7.40,-7.40)
minimum po slack:     (-32.40,-32.40)
total neg slack:      (-463.00,-463.00)
# of failing outputs:  20
```

Scelte Progettuali

Durante la creazione del nostro progetto abbiamo riscontrato alcuni punti che non vengono trattati dalle specifiche; abbiamo perciò fatto alcune scelte ad esse coerenti per quanto riguarda il funzionamento del nostro circuito, che elenchiamo di seguito:

1. Il nostro circuito conta i secondi anche durante lo stato di START, a patto che il segnale di INIT valga 1. Abbiamo supposto infatti che la centralina del veicolo possa essere attiva prima che il motore venga acceso meccanicamente (per esempio se il conducente gira la chiave senza accendere il motore), e che quindi possa rilevare 0 RPM. La scelta non influisce sul corretto funzionamento del circuito, infatti in base alle specifiche, solo quando il segnale di INIT vale 0 non deve essere mostrato alcun output significativo.
2. La nostra FSM esegue le transizioni in modo progressivo, ossia ogni stato può evolvere nel successivo, nel precedente o in START (nel caso di spegnimento). Questa scelta è stata dettata dall'ipotesi che un motore a combustione fornisce dei RPM che nel tempo crescono o decrescono, e che perciò sarà improbabile che per esempio dalla soglia "sotto-giri" passi repentinamente alla soglia "fuori-giri". Nonostante ciò, se dovesse presentarsi questa situazione particolare, il problema viene risolto automaticamente da SIS grazie alla minimizzazione degli stati; simulando infatti questo evento il circuito reagisce in maniera corretta.
3. Dal momento che il numero di bit dedicati al segnale di input relativo ai RPM non viene dato dalle specifiche, abbiamo optato per 14 bit, che sono ampiamente sufficienti ai fini del conteggio, visto che il numero massimo a cui il motore può arrivare è 6500 RPM.
4. Nell'assegnamento automatico della codifica degli stati del controllore abbiamo deciso di usare l'algoritmo "nova" piuttosto che "jedi"; in una prima implementazione del controllore infatti abbiamo notato essere preferibile perché più efficiente. Nonostante allo stato attuale non faccia più differenza, abbiamo mantenuto la scelta.