UNIVERSITA' DEGLI STUDI DI VERONA

ELABORATO SIS

ARCHITETTURA DEGLI ELABORATORI

TONINI FRANCESCO (VR408686)
CAPOGROSSO LUIGI (VR408776)

SOMMARIO

SPECIFICHE 1
SCHEMA GENERALE DEL CIRCUITO 3

CONTROLLORE: DIAGRAMMA DEGLI STATI 4
DATAPATH: ARCHITETTURA E COMPONENTI 6
STATISTICHE DEL CIRCUITO 9
NUMERO DI GATES E RITARDO DOPO LA MAPPATURA 11
SCELTE PROGETTUALI 12

SPECIFICHE

Si progetti un dispositivo per il monitoraggio di un impianto chimico industriale basato su un circuito sequenziale che riceve come input il PH di una soluzione

contenuta in un serbatoio, e fornisce in uscita lo stato della soluzione (compatibilmente con delle soglie pre-impostate) in termini di acido (A), basico (B) e neutro (N).
Il sistema deve portare sempre la soluzione allo stato neutro, accettando un transitorio di 5 cicli di clock; pertanto si richiede che al sesto ciclo di clock in cui il sistema sia allo stato A, venga aperta una valvola BS che riporti il sistema a N, e analogamente se allo stato B si apra la valvola AS. Il sistema deve inoltre fornire in uscita il numero di cicli di clock da cui si trova nello stato attuale.
Il circuito è composto da un controllore e un datapath con i seguenti ingressi e uscite (nel seguente ordine!).

- INPUTS:
  - INIT (1): quando vale 1 il sistema è acceso; quando vale 0 il sistema è spento e deve restituire 0 per tutti i bit di output;
  - RESET (1): quando posto a 1 il controllore deve essere resettato, ovvero tutte le uscite devono essere poste a 0 e il sistema riparte;
  - PH (8): valore del PH misurato dal rilevatore. Il range di misura è compreso tra 0 e 14 con risoluzione di 0,1.
- OUTPUTS:
  - ST (2): indica in quale stato si trova la soluzione al momento corrente (01-A, 10-N, 11-B). Si considera la soluzione acida (A) quando PH<6 e basica (B) se PH>8;
  - NCK (8): indica il numero di cicli di clock trascorsi nello stato corrente
  - VLV (2): indica quale valvola aprire per riportare la soluzione allo stato neutro nel caso in cui la soluzione si trovi da più di 5 cicli di clock in stato A o B (01-BS, 10-AS).

Si richiede inoltre che il circuito sia mappato sulla libreria tecnologica synch.genlyb e che venga ottimizzato per area.

SCHEMA GENERALE DEL CIRCUITO

Il circuito è composto da una parte di controllo e una parte di elaborazione, modellate rispettivamente attraverso il modello FSM e datapath. Si ottiene quindi, il modello FSMD in figura:

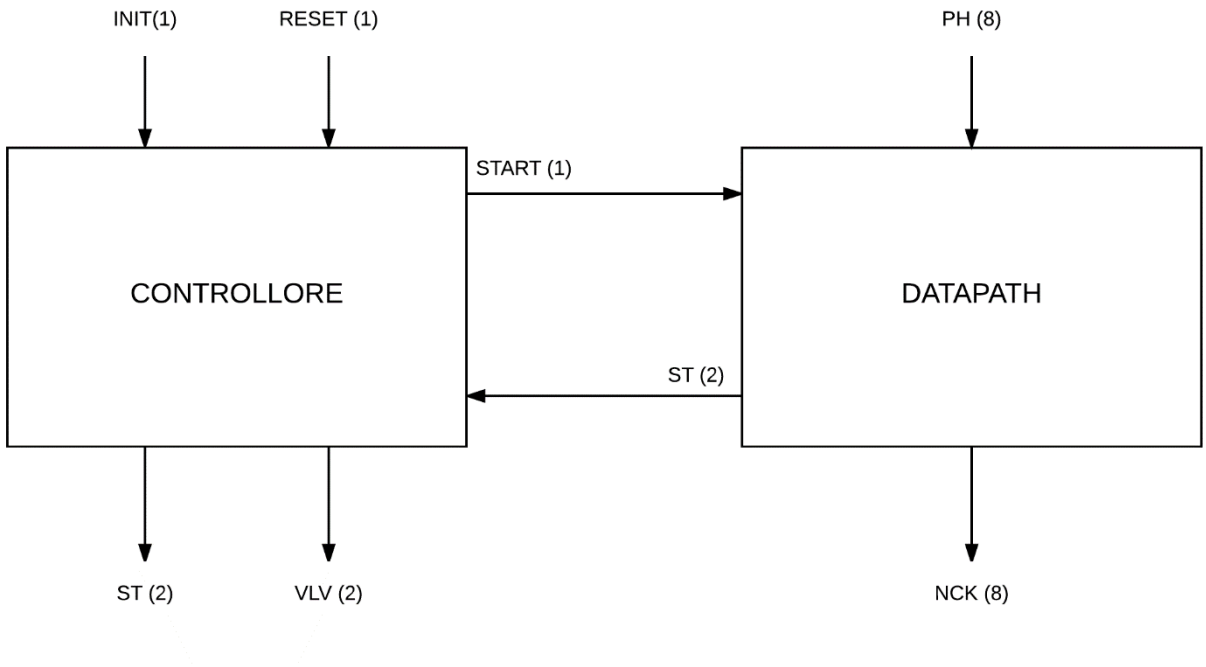

Figura 1 Modello FSMD

I file che contengono l'implementazione di tale dispositivo sono presenti nella cartella allegata.

Andiamo ora a vedere nello specifico le varie componenti.

CONTROLLORE: DIAGRAMMA DEGLI STATI

Il controllore (FSM) è una macchina a stati finiti di tipo deterministica, cioè caratterizzata da un'evoluzione deterministica della macchina da stato presente a stato prossimo sulla base della condizione fornita dagli ingressi primari. La nostra macchina appartiene al tipo Mealy.
Gli ingressi del controllore sono rispettivamente, in ordine, i seguenti cinque:

- INIT (1 bit): rappresenta lo stato della macchina: accesa (1) o spenta (0)
- RESET (1 bit): quando vale 1 le uscite devono essere azzerate;
- ST1 e ST0 (2 bit): indica in quale stato si trova la soluzione al momento corrente.
- TO (1 bit): quando vale 1 indica che il sistema si trova nello stato attuale da più di 5 cicli di clock.

Le uscite sono rispettivamente, in ordine, le seguenti cinque:

- ST0 e ST1 (2 bit): rappresenta lo stato della soluzione. Gli stati acido, basico e neutro sono stati codificati rispettivamente in 10, 11 e 01;
- VLV1 e VLV0 (2 bit): rappresenta la valvola da aprire per modificare lo stato della soluzione. Può assumere due valori significativi: quando assume valore 10 il sistema apre una valvola, detta BS, che porta la soluzione da uno stato acido ad uno stato neutro; quando assume valore 01 il sistema apre una valvola, detta AS, che porta la soluzione da uno stato basico ad uno stato neutro;
- START (1 bit): quando vale 1 segnala al datapath di azzerare il contatore.

La macchina presenta quattro stati, rispettivamente:

- INIT/RESET: rappresenta lo stato di “riposo” o reset. Il sistema permane in questo stato finché il segnale di inizializzazione (INIT) ha valore 0 e/o il segnale di reset (RESET) ha valore 1;
- ACIDO: rappresenta lo stato in cui la soluzione rilevata assume PH acido. Il sistema permane in questo stato finché lo stato della soluzione è acida;
- BASICO: rappresenta lo stato in cui la soluzione rilevata assume PH basico. Il sistema permane in questo stato finché lo stato della soluzione è basica;

- NEUTRO: rappresenta lo stato in cui la soluzione rilevata assume PH neutro, ma può a sua volta ritornare ad uno stato acido o basico, come da specifiche ipotizzato. Questo punto sarà spiegato in maniera più approfondita nel paragrafo riguardante le scelte progettuali.

(inserire STG)

DATAPATH: ARCHITETTURA E COMPONENTI

La parte di progettazione del datapath, ovvero la porzione del circuito che si occupa sostanzialmente di eseguire i calcoli, si basa sul collegamento incrementale fra loro di componenti di libreria.

Nel nostro caso esso riceve in ingresso il valore del PH (8 bit), misurato da un rilevatore, e rilascia in uscita tre valori, due dei quali diretti alla FSM. Nello specifico:

- ST (2 bit): indica lo stato della soluzione. Può assumere quattro valori, di cui tre significativi (acido, basico, neutro);

- TO (1 bit): quando vale 1 indica che il sistema si trova nello stato attuale da più di 5 cicli di clock;
- NCK (8 bit): indica il numero di cicli di clock trascorsi nello stato corrente. Questo segnale fa parte delle uscite del circuito specificate nel problema.

Vediamo ora lo schema del circuito:

Il PH ricevuto in ingresso viene inviato a due operatori di confronto che ne verificano la maggiorazione rispetto a 80 e 60 (vedasi *scelte progettuali)*. Il risultato di queste due operazioni è collegato all'entrata di selezione di due multiplexer come in figura

(figura)

Il risultato di pH > 80 è collegato al segnale di selezione di MUX1. Se il confronto ha esito positivo, quindi uscita 1, MUX1 restituirà 11, ovvero la codifica dello stato basico, oppure 01, codifica dello stato neutro.
Allo stesso modo, il risultato di pH > 60 è collegato al segnale di selezione di MUX2. Se il confronto ha esito positivo, quindi valore 1 sull'uscita, MUX2 restituirà il valore in uscita da MUX1, oppure 10, codifica dello stato acido. L'uscita di MUX2 corrisponderà al nostro ST, presente nello schema generale del circuito.

(img)

Le specifiche del problema ci impongo di aggiungere un circuito che incrementi una variabile per un valore costante (1) ad ogni ciclo di clock e informi il controllore, tramite un opportuno segnale, quando il conteggio ha superato il valore preimpostato di 5. Tale circuito è stato implementato usando un registro ad 8 bit di tipo flip-flop, un sommatore, un comparatore ed un mux, tutti a 8 bit.

Il circuito continua ad incrementare il valore contenuto nel registro fino a quando il valore in ingresso START non assume valore 1. In questa situazione il multiplexer azzera le uscite e il valore contenuto nel registro. Il segnale in uscita TO assume valore 1 quando il confronto tra il valore contenuto ottenuto dal sommatore è maggiore del valore costante 5.

STATISTICHE DEL CIRCUITO

Vengono riportate qui di seguito, accompagnate dai corrispettivi dati, le statistiche del circuito.

Per la parte del controllore abbiamo ottimizzato il circuito riducendo il numero di stati (comando “state_minimize stamina”) e assegnato la codifica per ogni stato (“state_assign jedi”). Abbiamo usato il comando “source -x script.rugged” per minimizzare la logica combinatoria e, una volta collegato il controllore e il datapath, effettuato la mappatura sulla libreria tecnologica *synch.genlib* preferendo l’ottimizzazione per area.

```
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 : 4
Number of states in minimized machine : 3
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> print_stats
FSM             pi= 5   po= 5   nodes=  7       latches= 2
lits(sop)=  45  #states(STG)=   3
```

Figura 6 Riduzione e codifica degli stati, statistiche prima dell'ottimizzazione

Come ci si aspettava vi è stata una riduzione degli stati passando da 4 a 3. Abbiamo poi lanciato uno script di ottimizzazione della logica combinatoria.

```
sis> print_stats
FSM             pi= 5   po= 5   nodes=  8       latches= 2
lits(sop)=  26  #states(STG)=   3
sis>
```

Figura 7 Statistiche del controllore dopo l'ottimizzazione

L'ottimizzazione della logica combinatoria ci ha permesso di passare da 45 letterali a 26.
Il datapath è stato ottimizzato con la stessa tecnica adottata per la FSM. Dopo aver lanciato il comando "source -x script.rugged" il nostro datapath è passato da 330 a 87 letterali.

```
sis> print_stats
DATAPATH        pi= 9   po=11   nodes= 89       latches= 8
lits(sop)= 330
```

Figura 8 Statistiche datapath prima dell'ottimizzazione

```
sis> print_stats
DATAPATH        pi= 9   po=11   nodes= 25       latches= 8
lits(sop)=  87
```

Figura 9 Statistiche datapath dopo l'ottimizzazione

NUMERO DI GATES E RITARDO DOPO LA MAPPATURA

Dopo aver unito le datapath, FSM ed aver creato la nostra FSMD finale, abbiamo eseguito nuovamente il comando di ottimizzazione della logica combinatoria nonché mappato il circuito sulla libreria *synch.genlib* preferendo la riduzione dell'area a discapito del ritardo. Il risultato finale è di una FSMD con area pari a circa 2600 e 144 letterali.

```
sis> map -W -s -m 0
>>> before removing serial inverters <<<
# of outputs:          22
total gate area:       2816.00
maximum arrival time: (29.20,29.20)
maximum po slack:     (-8.80,-8.80)
minimum po slack:     (-29.20,-29.20)
total neg slack:      (-432.60,-432.60)
# of failing outputs:  22
>>> before removing parallel inverters <<<
# of outputs:          22
total gate area:       2816.00
maximum arrival time: (29.20,29.20)
maximum po slack:     (-8.80,-8.80)
minimum po slack:     (-29.20,-29.20)
total neg slack:      (-432.60,-432.60)
# of failing outputs:  22
# of outputs:          22
total gate area:       2624.00
maximum arrival time: (28.20,28.20)
maximum po slack:     (-8.80,-8.80)
minimum po slack:     (-28.20,-28.20)
total neg slack:      (-417.40,-417.40)
# of failing outputs:  22
sis> print_stats
FSMD            pi=10   po=12   nodes= 70       latches=10
lits(sop)= 144
```

Figura 20 Mappatura e statistiche circuito finale

SCELTE PROGETTUALI

Leggendo e ponendo molta attenzione alle specifiche del problema, confrontandoci sul lavoro da svolgere, e su quali potessero essere le soluzioni più adeguate, abbiamo definito alcune scelte progettuali riguardo il funzionamento del circuito, che verranno ora argomentate:

- Codifica in ingresso del pH: il sistema, che riceve come input il pH di una soluzione contenuta in un serbatoio, con un range di misura compreso tra 0 e 14 con risoluzione di 0,1, ammette 140 possibili combinazioni differenti. La soluzione da noi adottata è stata quella di moltiplicare per 10 il valore in input, in modo tale da ottenere sempre un numero intero positivo. Tale operazione ci ha permesso di adottare una codifica in modulo, la quale risulta sufficientemente esaustiva, poiché con gli 8 bit d'ingresso ci permette di codificare 256 situazioni differenti.
- Variazione di pH: come da specifiche indicato, "*Il sistema deve portare sempre la soluzione allo stato neutro*" è stato da noi interpretato come la possibilità che una soluzione allo stato neutro possa tornare ad uno stato basico o acido.
- Condizione di reset o init: un altro caso da noi gestito è relativo al comportamento in qualunque condizione della macchina. Infatti è possibile utilizzare i segnali di reset e init in qualunque momento, con corrispettivi comportamenti della macchina spiegati in precedenza.
- Calcolo e conteggio: il datapath è stato suddiviso in due "macro blocchi", il primo addetto al calcolo del pH e il secondo al conteggio. Questo ci ha permesso di evitare errori di programmazione e quindi velocizzare lo sviluppo