Università degli Studi di Verona
Dipartimento di Informatica
AA 2016/2017
Corso di laurea in Informatica

Elaborato SIS

Laboratorio di Architettura degli Elaboratori

Agresti Nicola VR407685
Piccinelli Marco VR407621

Indice

Schema generale del circuito

Il dispositivo da noi implementato monitora il pH di una soluzione contenuta nei serbatoi di un impianto chimico industriale. È basato su un circuito sequenziale che riceve come input il pH e fornisce in uscita lo stato della soluzione (compatibilmente con delle soglie pre-impostate) in termini di acido (A), basico (B) e neutro (N). Se il sistema si trova per 5 cicli di clock nello stato A, al sesto ciclo apre una valvola BS che riporta il sistema a N, e analogamente se si trova nello stato B aprirà la valvola AS. Il sistema deve inoltre fornire in uscita il numero di cicli di clock da cui si trova nello stato attuale.

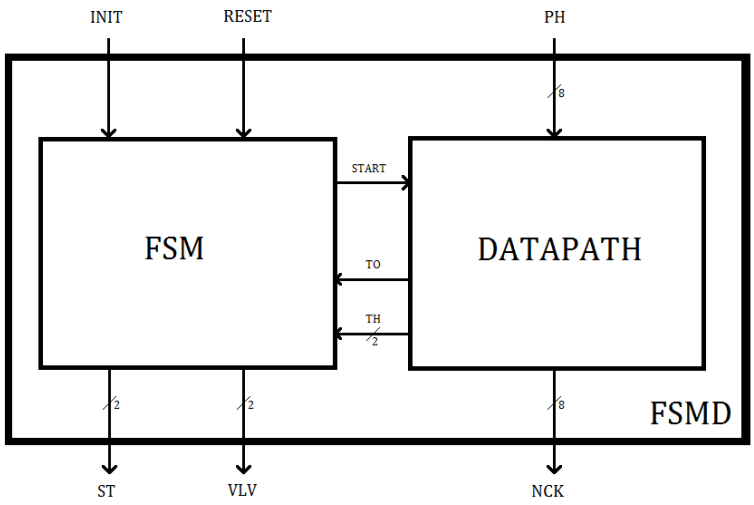
Il circuito è composto da un controllore e un datapath con i seguenti ingressi e uscite.

INPUTS:
• INIT [1]: quando vale 1 il sistema è acceso; quando vale 0 il sistema è spento e restituisce 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 (01-BS, 10-AS)

Il controllore inoltre è collegato al datapath con tre segnali.

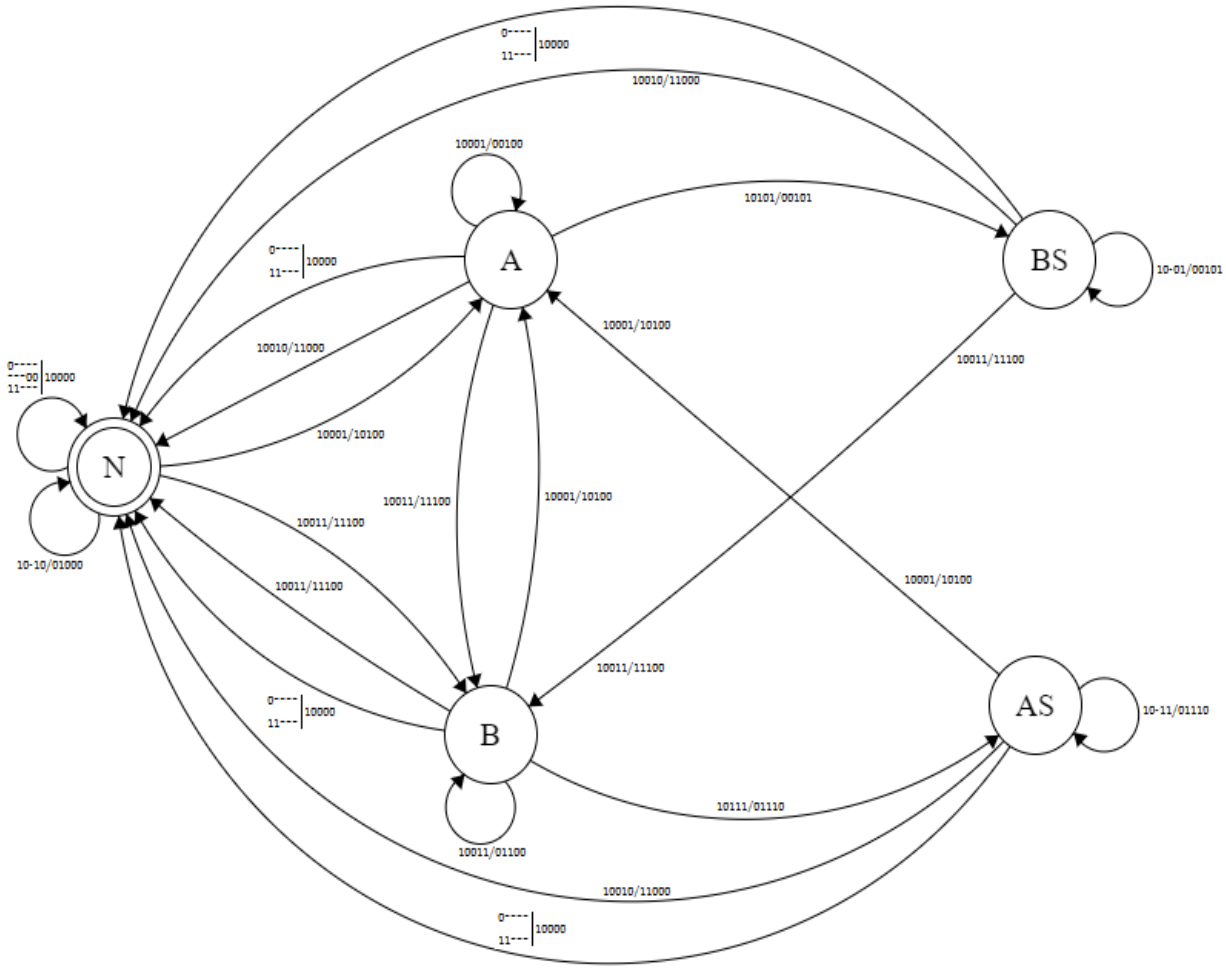
INPUTS:
• TO [1]: segnale di timeout che indica quando il sistema si trova in uno stesso stato da più di 5 cicli di clock
• TH [2]: stato in cui si trova il sistema
OUTPUTS:
• START [1]: segnale di start che comanda il datapath in funzione degli ingressi del controllore



Controllore

Il controllore è una macchina a stati finiti che presenta cinque ingressi (INIT, RESET, TO, TH1, TH0) e cinque uscite (START, ST1, ST0, VLV1, VLV0). Di seguito sono descritti i 5 stati della macchina:

• N: è lo stato di reset e la FSM resta in questo stato finché la macchina è spenta oppure finché il pH è neutro (ST1=1 ST0=0).
• A: la FSM arriva in questo stato quando il pH misurato è acido (ST1=0 STt0=1). Rimane in questo stato finché il segnale di timeout (TO) è 0
• BS: la FSM arriva in questo stato solamente quando il segnale di timeout (TO) è 1 nello stato A. La valvola BS viene aperta (VLV1=0, VLV0=1) e rimane aperta finché non viene misurato un Ph non acido.
• B: la FSM arriva in questo stato quando il pH misurato è acido (St1=0 St0=1). Rimane in questo stato finché il segnale di timeout (TO) è 0.
• AS: la FSM arriva in questo stato solamente quando il segnale di timeout (TO) è 1 nello stato B. La valvola AS viene aperta (VLV1=1, VLV0=0) e rimane aperta finché non viene misurato un Ph non basico.



Datapath

Il datapath è la parte che si occupa di eseguire i calcoli, nel nostro caso riceveremo in input il segnale di START (1 bit) e il pH (8 bit).

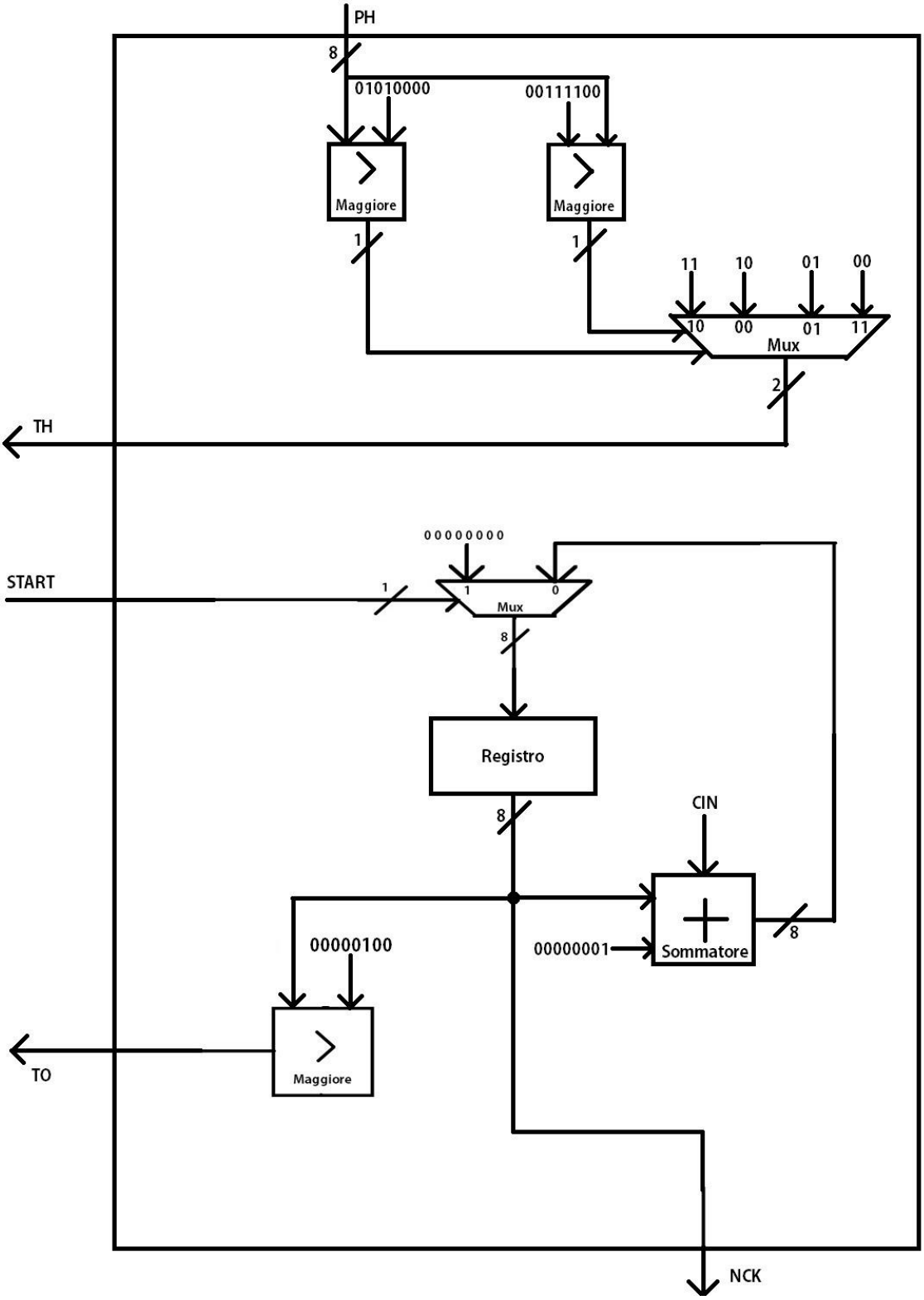
Per determinare la tipologia di TH in output abbiamo implementato 3 componenti:

1. Maggiore: nel primo componente di confronto andremo a determinare se il pH in input sarà maggiore di 80, verrà generato 1 in caso positivo altrimenti 0.
2. Analogamente nel secondo componete andremo a confrontare se 60 è maggiore del pH inserito, per poi ottenere in output 1 oppure 0.
3. Mux: l'altro componete essenziale è il multiplexer, questo riceverà in input i 2 bit uscenti dagli operatori di confronto. Questi bit saranno selezionati e verrà dato in output la sequenza di bit corrispondente. Nel nostro caso:
    a. Input 10 → Output 11 (TH basico)
    b. Input 00 → Output 10 (TH neutro)
    c. Input 01 → Output 01 (TH acido)
    d. Input 11 → Output 00 (TH non valido)

Per calcolare l'output del numero dei cicli di clock (NCK) e del segnale di timeout (TO) abbiamo implementato 4 componenti:

1. Mux: riceverà in input il segnale di START, e verrà assegnata la corrispondente sequenza di bit:
    a. Input 1 → Output 00000000
    b. Input 0 → Output sarà il risultato prodotto dal sommatore
2. Reg: il registro parallelo/parallelo si occuperà ad ogni ciclo di clock di mandare in output la sequenza di bit ricevuta dal mux precedente.
3. Sommatore: questo componente aritmetico riceve l'NCK in input, ci permetterà di aumentarlo di 00000001 ad ogni ciclo di clock in modo da avere un conteggio sempre accurato.
4. Maggiore: questo componente aritmetico invece ci permetterà di determinare se il TO è nello stesso stato da più di 5 cicli di clock o meno, in caso positivo l'output sarà 1 altrimenti 0. Questo grazie al confronto del NCK in input con il valore 4.

Schema del datapath



Statistiche del circuito

Per ottimizzare l'area andremo a ridurre il numero dei letterali (lits) a scapito di un aumento dei nodi (nodes) e quindi di un aumento del ritardo massimo. Per ottimizzare il circuito abbiamo usato vari comandi tra cui eliminate, fx, simplify e full_simplify. Inoltre abbiamo utilizzato anche lo script "script.rugged", il quale contiene una serie di comandi di ottimizzazione molto performante.

Di seguito riportiamo le statistiche del circuito prima e dopo l'ottimizzazione per area.

Abbiamo provato ad ottimizzare la FSM con il comando "state_minimize stamina" ma l'algoritmo non ha ridotto il numero degli stati.

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl fsm_raw.blif
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 5
Number of states in minimized machine : 5
```

In seguito codifichiamo gli stati con "state_assign jedi".

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl fsm_raw.blif
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> print_stats
FSM            pi= 5   po= 5   nodes=  8       latches= 3
lits(sop)= 110  #states(STG)=   5
```

Queste sono le statistiche ottenute dopo l'ottimizzazione della FSM.

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl fsm.blif
sis> print_stats
FSM            pi= 5   po= 5   nodes= 11       latches= 3
lits(sop)=  38  #states(STG)=   5
sis>
```

Prima di procedere con il datapath abbiamo ottimizzati tutti i componenti utilizzati con lo script "script.rugged". Queste sono le statistiche prima e dopo l'ottimizzazione del datapath.

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl datapath.blif
sis> print_stats
datapath        pi= 9   po=11   nodes=108       latches= 8
lits(sop)= 412
sis>
```

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl datapath.blif
sis> print_stats
datapath        pi= 9   po=11   nodes= 25       latches= 8
lits(sop)=  86
sis>
```

Abbiamo poi proseguito ottimizzando la FSMD.

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl FSMD.blif
sis> print_stats
FSMD            pi=10   po=12   nodes=116       latches=11
lits(sop)= 522
sis>
```

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> rl FSMD.blif
sis> psf
FSMD            pi=10   po=12   nodes= 33       latches=11
lits(sop)= 112
sis>
```

Mapping tecnologico

Come da consegna, dopo aver ottimizzato il circuito per area, abbiamo mappato la nostra FSMD sulla libreria tecnologica *synch.genlib* in modo da avere statistiche più veritiere riguardo l'area ed il ritardo. L'area finale è 2760 e il ritardo massimo è 27,40.

```
UC Berkeley, SIS 1.3.6 (compiled 2016-11-15 17:22:37)
sis> read_library synch.genlib
sis> rl FSMD.blif
sis> map -s
>>> before removing serial inverters <<<
# of outputs:           23
total gate area:        3000.00
maximum arrival time: (27.60,27.60)
maximum po slack:     (-7.20,-7.20)
minimum po slack:     (-27.60,-27.60)
total neg slack:      (-448.60,-448.60)
# of failing outputs:  23
>>> before removing parallel inverters <<<
# of outputs:           23
total gate area:        3000.00
maximum arrival time: (27.60,27.60)
maximum po slack:     (-7.20,-7.20)
minimum po slack:     (-27.60,-27.60)
total neg slack:      (-448.60,-448.60)
# of failing outputs:  23
# of outputs:           23
total gate area:        2760.00
maximum arrival time: (27.40,27.40)
maximum po slack:     (-7.20,-7.20)
minimum po slack:     (-27.40,-27.40)
total neg slack:      (-443.40,-443.40)
# of failing outputs:  23
sis>
```

Scelte progettuali

Abbiamo adottato una codifica diversa per il pH. È moltiplicato per 10 in modo da gestire al meglio la risoluzione richiesta al decimo (0.1). Quindi pH acido sarà strettamente minore di 60 mentre pH basico sarà strettamente maggiore di 80. Durante lo sviluppo della fsm abbiamo tentato di minimizzarla a 3 stati ma abbiamo riscontrato problemi con il datapath. Confrontando le aree, però, non c'erano differenze sostanziali. Quindi abbiamo scelto di adottare una fsm a 5 stati perché molto più chiara e pulita dal punto di vista grafico. Abbiamo inoltre eliminato il COUT dal sommatore del datapath in quanto il resto finale era superfluo. Infine, abbiamo deciso di inserire nello stato N di reset della fsm anche la combinazione ---00 nonostante essa non potrà mai essere soddisfatta in quanto dipende a sua volta da una combinazione di TH[2] che non potrà mai presentarsi. Questa combinazione è 11 (un numero non può essere sia maggiore di 80 che minore di 60). L'abbiamo aggiunta per impedire errori nel circuito in seguito ad un possibile malfunzionamento del mux presente nel datapath.