UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

Elaborato SIS
Laboratorio di Architettura degli Elaboratori

Progettazione macchinario chimico

A.A. 2021/2022

Filippo Barbieri (VR472408)
Alessio Brighenti (VR471509)

Indice

1. Specifiche

Si progetti il circuito sequenziale che controlla un macchinario chimico il cui scopo è portare una soluzione iniziale a pH noto, ad un pH di neutralità. Il valore del pH viene espresso in valori compresi tra 0 e 14.

Il circuito controlla due valvole di erogazione: una di soluzione acida e una di soluzione basica.

Se la soluzione iniziale è acida, il circuito dovrà procedere all'erogazione della soluzione basica fintanto che la soluzione finale non raggiunga la soglia di neutralità (pH compreso tra 7 e 8).

Analogamente, se la soluzione iniziale è basica, il circuito procederà all'erogazione di soluzione acida fino al raggiungimento della soglia di neutralità.

Per pH acido si intende un valore strettamente inferiore a 7, mentre per basico si intende una soluzione con pH strettamente maggiore a 8.

Il pH viene codificato in fixed-point, con 4 bit riservati per la parte intera e gli altri per la parte decimale.

Le due valvole hanno flussi differenti di erogazione.

La valvola relativa alla soluzione basica eroga una quantità di soluzione che permette di alzare il pH della iniziale di 0.25 ogni ciclo di clock.

La valvola relativa alla soluzione acida eroga una quantità di soluzione che permette di abbassare il pH della soluzione iniziale di 0.5 ogni ciclo di clock.

Il circuito ha tre ingressi nel seguente ordine:

- RST (1 bit)
- START (1 bit)
- pH (8 bit, 4 parte intera e 4 per la parte decimale)

Gli output sono i seguenti e devono seguire il seguente ordine:

- FINE_OPERAZIONE (1 bit)
- ERRORE_SENSORE (1 bit)
- VALVOLA_ACIDO (1 bit)
- VALVOLA_BASICO (1 bit)
- PH_FINALE (8 bit)
- NCLK (8 bit)

Input e output devono essere definiti nell'ordine sopra specificato (da sinistra verso destra).

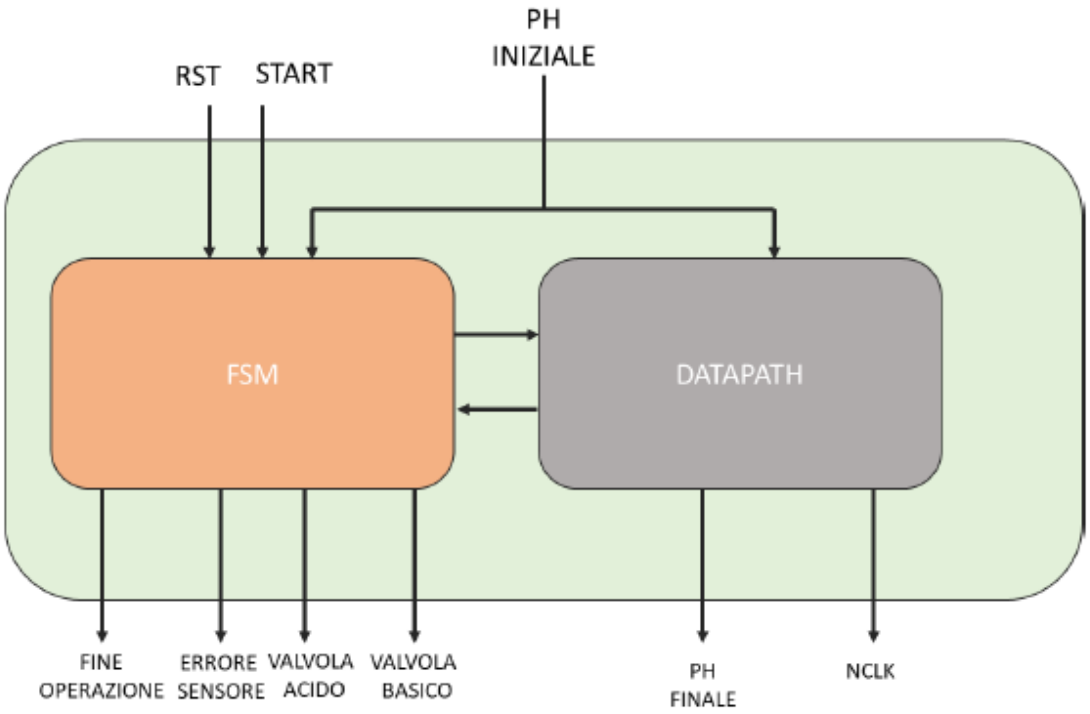
Le porte con più bit devono essere descritte utilizzando la codifica con il bit più significativo a sinistra.

Il meccanismo è guidato come segue:

- Quando il segnale RST viene alzato, il sistema torna da un qualsiasi stato allo stato di Reset, mettendo tutte le porte in output a zero.

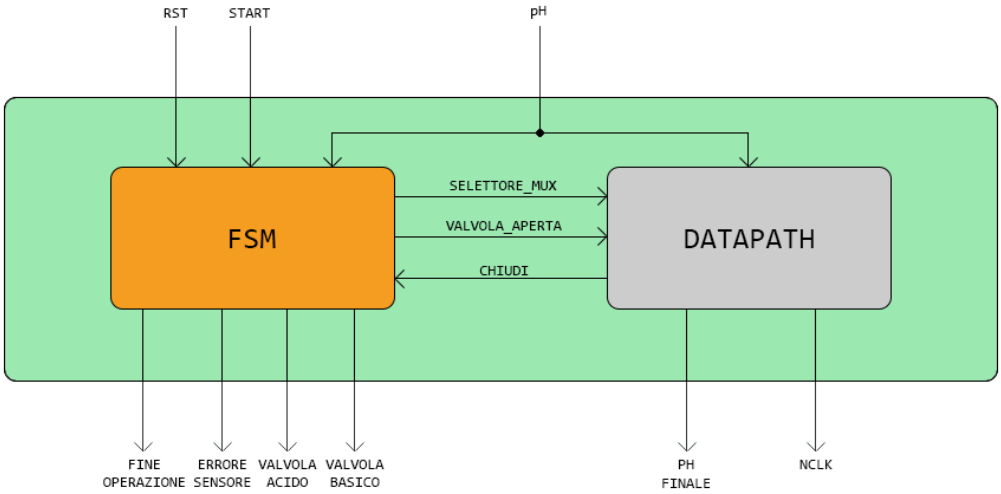
- Per procedere, Il sistema riceve in input il segnale di START, con valore 1, e il segnale del pH iniziale per un solo ciclo di clock. Il sistema potrà quindi procedere con la fase di elaborazione.
- Se la soluzione iniziale è acida, viene aperta la valvola della soluzione basica, mettendo a 1 il relativo output. Analogamente, se la soluzione iniziale è basica, viene aperta la valvola della soluzione acida mettendo a 1 la porta VALVOLA_ACIDO.
- Il sistema mantiene aperte le valvole per il tempo necessario al raggiungimento della soglia di neutralità (calcolata dal sistema).
- Una volta terminata l'operazione, il sistema deve chiudere tutte le valvole aperte, riportare il pH finale sulla porta in output PH_FINALE e alzare la porta di FINE_OPERAZIONE.
- La porta NCLK riporta quanti cicli di clock sono stati necessari per portare la soluzione a neutralità.
- Se il valore del pH non è valido (> 14) il sistema deve riportare l'errore alzando l'output ERRORE_SENSORE.

Lo schema generale del circuito deve rispettare la FSMD riportata di seguito:



- È possibile aggiungere degli ulteriori segnali interni per la comunicazione tra FSM e DATAPATH
- Le porte di input e di output devono rispettare l'ordine definito ed essere collegate al rispettivo sotto modulo
- Il DATAPATH deve essere unico: se volete definire più DATAPATH, questi devono essere inglobati in un unico modello.
- <u>È compito della FSM identificare se il pH della soluzione iniziale sia acido o basico!</u>

2. Architettura generale del circuito



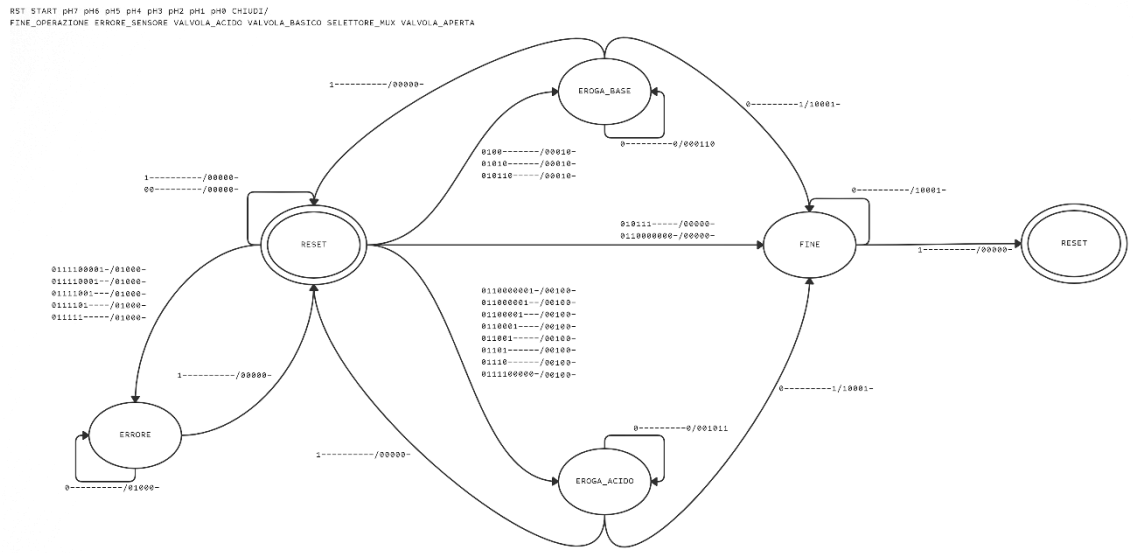
Il circuito si articola in due parti: una FSM, che funge da unità di controllo, e un datapath, che provvede al calcolo dei dati necessari.

Le due parti comunicano tra di loro tramite due segnali di controllo (SELETTORE_MUX e VALVOLA_APERTA) e un segnale di stato (CHIUDI).

Come da specifiche, il segnale pH a 8 bit viene codificato in fixed point utilizzando quattro bit per la parte intera e quattro per la parte decimale; altrettanto vale per il segnale PH_FINALE. Invece, il segnale NCLK viene codificato in modulo.

I file che descrivono singolarmente FSM e datapath si trovano nella cartella “non_ottimizzato” del progetto, rispettivamente con i nomi “fsm.blif” e “datapath.blif”.

3. FSM



Il controllore del macchinario chimico è una macchina a stati finiti di tipo Mealy, ovvero la cui funzione d'uscita dipende sia dallo stato corrente che dai valori in ingresso.

I segnali di input e output della macchina sono i seguenti:

INPUT	**OUTPUT**
RST [1] START [1] pH [8] CHIUDI [1]	FINE_OPERAZIONE [1] ERRORE_SENSORE [1] VALVOLA_ACIDO [1] VALVOLA_BASICO [1] SELETTORE_MUX [1] VALVOLA_APERTA [1]

Gli stati della macchina sono cinque e rappresentati come segue:

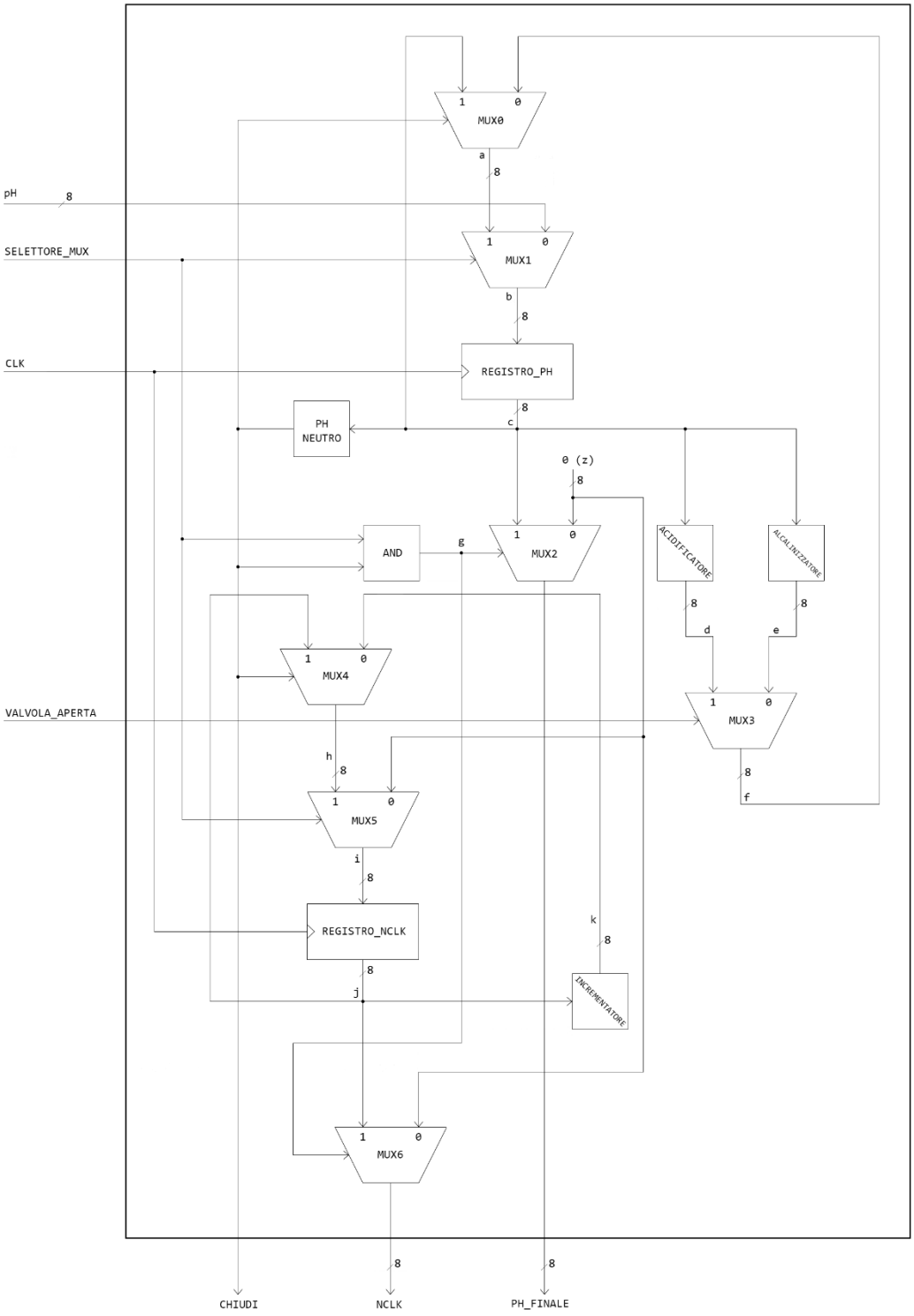
- RESET (stato iniziale): tutti gli output sono a 0 e la macchina è pronta a ricevere un valore di pH in ingresso. La macchina rimane nello stesso stato quando RST è alzato, oppure quando RST è abbassato insieme a START. Quando START è a 1, la macchina passa in stato di ERRORE (alzando il relativo bit d'uscita) se viene inserito un pH non valido (superiore a 14), altrimenti può passare in uno dei due stati di erogazione (con corrispondente bit alzato) o direttamente nello stato di FINE se il pH inserito è già neutro.
- ERRORE: stato in cui la macchina si trova dopo l'inserimento di un pH invalido e in cui permane, lasciando a 1 l'uscita ERRORE_SENSORE, finché non viene alzato RST.
- EROGA_BASICO: stato in cui la macchina si trova dopo l'inserimento di un pH acido e in cui rimane, lasciando a 1 l'uscita VALVOLA_BASICO, finché dal datapath non arriva il segnale CHIUDI a 1. Quando CHIUDI è alzato, si passa nello stato di FINE alzando la relativa uscita. Se viene alzato RST, la macchina torna nello stato di RESET con tutti gli output a 0.
- EROGA_ACIDO: stato in cui la macchina si trova dopo l'inserimento di un pH basico e in cui rimane, lasciando a 1 l'uscita VALVOLA_ACIDO, finché dal datapath non arriva il segnale CHIUDI a 1. Quando CHIUDI è alzato, si passa nello stato di FINE alzando la relativa uscita. Se viene alzato RST, la macchina torna nello stato di RESET con tutti gli output a 0.

- FINE: stato in cui la macchina si trova dopo l'inserimento di un valore di pH neutro o dopo il raggiungimento dell'intervallo di neutralità a seguito dell'erogazione di soluzione acida/basica. La macchina rimane nello stato, con uscita FINE_OPERAZIONE alzata, finché non viene alzato RST che la porta nello stato di RESET.

Quanto sopra descritto si può visivamente verificare nel grafo delle transizioni riportato. Le transizioni sono in totale 30, così ottenute dopo aver ridotto le combinazioni di ingresso nello stato di RESET utilizzando i don't care.

Si può notare che i bit corrispondenti al pH vengono considerati solo nello stato di RESET, in quanto compito della FSM identificare l'entità della soluzione iniziale; negli altri stati la gestione è demandata al datapath.

4. Datapath


MUX0
MUX1
REGISTRO_PH
PH NEUTRO
0 (z)
AND
MUX2
ACIDIFICATORE
ALCALINIZZATORE
MUX4
MUX3
MUX5
REGISTRO_NCLK
INCREMENTATORE
MUX6
pH
SELETTORE_MUX
CLK
VALVOLA_APERTA
CHIUDI
NCLK
PH_FINALE

Il datapath ha i seguenti segnali di input e output:

INPUT	**OUTPUT**
SELETTORE_MUX [1] VALVOLA_APERTA [1] pH [8]	CHIUDI [1] PH_FINALE [8] NCLK [8]

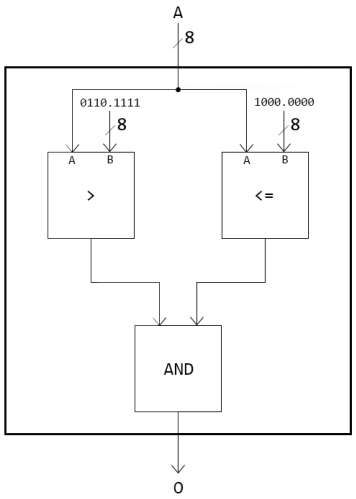
Il datapath può considerarsi come composto da due sotto-circuiti: il primo atto a calcolare il pH della soluzione dopo ogni aggiunta di soluzione acida o basica, il secondo per contare quanti cicli di clock sono necessari per portare la soluzione a pH neutro.

Si può notare come le due parti abbiano un design molto simile, differente solo nel calcolo.

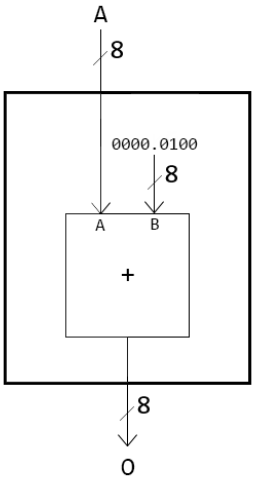
4.1 Calcolo pH

Segue, per ogni componente, una descrizione del proprio funzionamento e ruolo;

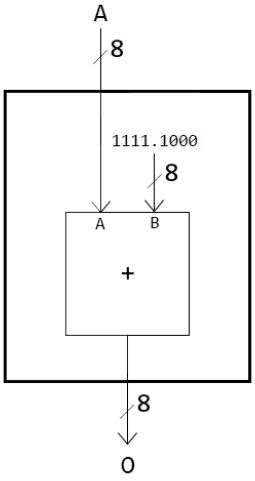
- REGISTRO_PH: registro a 8 bit inizializzati a 0, attivo sul fronte di salita del clock, che contiene il valore di pH in fixed point.
- PH_NEUTRO: unità che riceve in ingresso un segnale a 8 bit in fixed point, corrispondente al pH, e pone l'unico bit di uscita a 1 se il pH è nell'intervallo di neutralità. Il pH viene confrontato con le costanti 6.9375_{10} (0110.1111_2) e 8_{10} (1000.0000_2) rispettivamente tramite gli operatori > e <=, i quali risultati vengono moltiplicati (moltiplicazione booleana) per fornire il risultato finale.



- ALCALINIZZATORE: unità che riceve in ingresso un segnale a 8 bit in fixed point, corrispondente al pH, e produce come risultato un segnale con il valore aumentato di 0.25_{10} (0000.0100_2).



- ACIDIFICATORE: unità che riceve in ingresso un segnale a 8 bit in fixed point, corrispondente al pH, e produce come risultato un segnale con il valore diminuito di 0.5_{10} (0000.1000_2). La sottrazione viene fatta con un sommatore in cui la costante 0.5_{10} viene rappresentata in complemento a 2 (1111.1000_2).

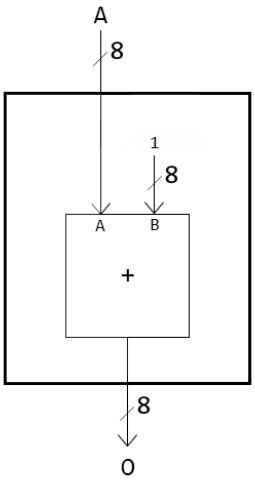


- MUX0: multiplexer con due ingressi a 8 bit che fa passare il valore di pH salvato nel registro, se questo è neutro, altrimenti lascia passare quello calcolato a seguito dell'erogazione di soluzione acida o basica.
- MUX1: multiplexer con due ingressi a 8 bit che fa passare il valore di pH in ingresso se il controllore si trova in stato di RESET o ERRORE, altrimenti lascia passare il valore di pH risultante da MUX0.
- MUX2: multiplexer con due ingressi a 8 bit che fa passare il valore di pH nel registro una volta terminata l'elaborazione, altrimenti lascia passare la costante 0.
- MUX3: multiplexer con due ingressi a 8 bit che fa passare il valore di pH calcolato dall'acidificatore o dall'alcalinizzatore a seconda della valvola che è stata aperta dal controllore.

4.2 Conteggio cicli di clock

Segue, per ogni componente, una descrizione del proprio funzionamento e ruolo;

- REGISTRO_NCLK: registro a 8 bit inizializzati a 0, attivo sul fronte di salita del clock, che mantiene il conteggio dei cicli di clock passati (codificati in modulo).
- INCREMENTATORE: unità che riceve in ingresso un segnale a 8 bit in modulo e vi somma 1_{10} (00000001_2).



- MUX4: multiplexer con due ingressi a 8 bit che fa passare il numero di cicli salvato nel registro, quando il pH è già neutro, altrimenti lascia passare quello calcolato a seguito dell'erogazione di soluzione acida o basica.
- MUX5: multiplexer con due ingressi a 8 bit che fa passare la costante 0 se controllore si trova in stato di RESET o ERRORE, altrimenti lascia passare il numero di cicli risultante da MUX4.
- MUX6: multiplexer con due ingressi a 8 bit che fa passare il numero di cicli nel registro una volta terminata l'elaborazione, altrimenti lascia passare la costante 0.

5. Statistiche del circuito

	PRIMA DELL'OTTIMIZZAZIONE		***DOPO L'ESECUZIONE DI SCRIPT.RUGGED***		***DOPO LA NOSTRA MINIMIZZAZIONE***	
	NODI	*LETTERALI*	*NODI*	*LETTERALI*	*NODI*	*LETTERALI*
FSM	9	229	11	61	11	51
					+22.22%	-77.73%
DATAPATH	200	717	46	212	44	201
					-78.00%	-71.97%
FSMD	56	253	54	254	61	245
					+08.92%	-03.16%

Nota 1: le statistiche della FSMD prima dell'ottimizzazione si riferiscono al componente derivato dall'unione di FSM e datapath già ottimizzati; le statistiche finali derivano da un'ulteriore ottimizzazione effettuata sull'insieme
Nota 2: le percentuali sono espresse in relazione ai valori prima dell'ottimizzazione

- Numero di nodi: 61
- Numero di letterali: 245

L'ottimizzazione è stata eseguita dapprima singolarmente su FSM e datapath e, una volta creato il circuito generale con la versione minima delle due parti, abbiamo ottimizzato ulteriormente l'insieme.

Le sequenze di comandi SIS utilizzate per minimizzare i circuiti sono state ottenute tramite uno script python che abbiamo scritto per eseguire randomicamente sul circuito un numero definito di comandi di ristrutturazione/ottimizzazione e rilevarne le statistiche.

Per la minimizzazione della FSM sono state tentate diverse sequenze di comandi dopo la minimizzazione degli stati con algoritmo *stamina* e dopo la codifica di questi in alcuni casi con algoritmo *jedi*, in altri con algoritmo *nova*, al fine di capire quale dei due permettesse di arrivare a un risultato migliore.

Si può notare che per il datapath si è notevolmente ridotto sia il numero di nodi che di letterali, mentre per FSM e FSMD è normale che aumenti un po' il ritardo per poter diminuire l'area.

Gli script di ottimizzazione implementati ci hanno permesso di migliorare le già ottime statistiche ottenute con l'esecuzione di script.rugged, seppur leggermente aumentando il numero di nodi finale, ma favorendo una minore area (come da specifiche).

6. Mapping: area e ritardo

Una volta eseguita la minimizzazione del circuito, questo viene mappato con le componenti della libreria tecnologica *synch.genlib* per poter aver un modello del circuito che possa essere fisicamente realizzato.

Dopo la mappatura si possono visualizzare le statistiche del circuito risultante:

Total Area	*5656.00*
Gate Count	*157*
Buffer Count	*19*
Inverter Count	*27*
Maximum arrival time	*29.60*
Most Negative Slack	*-29.60*
Sum of Negative Slacks	*-601.00*
Number of Critical PO	*39*

Abbiamo ottenuto un circuito che utilizza 157 gate logici con un ritardo massimo di 29.6.

Quanto sopra descritto è conseguente alla mappatura eseguita con il comando *map -m 0*, tuttavia, utilizzando il comando *map -m 0 -AF* (come suggerito dal manuale di SIS) le statistiche che si ottengono sono le seguenti:

Total Area	5864.00
Gate Count	177
Buffer Count	25
Inverter Count	53
Maximum arrival time	27.80
Most Negative Slack	0.00
Sum of Negative Slacks	0.00
Number of Critical PO	0

La differenza è dovuta al fatto che l'opzione F esegue l'ottimizzazione dei fanout, disabilitando la gestione interna di questi. Successivamente con l'opzione A (pensata appositamente per essere eseguita dopo la F) viene recuperata area ridimensionando i buffer e gli invertitori, con un leggero o nullo aumento del ritardo.

Si nota che aumenta il numero di gate logici (e l'area), ma vengono azzerati il numero di cammini critici e si riduce lievemente il ritardo.

7. Scelte progettuali

7.1 Segnali interni

Abbiamo deciso di utilizzare tre segnali interni:

- SELETTORE_MUX: segnale di controllo necessario al datapath per indirizzare correttamente il flusso di dati a seconda dello stato in cui si trova la FSM.
- VALVOLA_APERTA: segnale di controllo che indica al datapath se "attivare" l'alcalinizzatore o l'acidificatore finché viene erogata soluzione basica/acida; quando la FSM non è in uno dei due stati di erogazione, questo bit viene messo a don't care per poter ottimizzare il circuito, in quanto il segnale risultante viene bloccato da MUX0.
- CHIUDI: segnale di stato corrispondente all'uscita dell'unità PH_NEUTRO che comunica alla FSM di passare da uno stato di erogazione allo stato di FINE.

7.2 Latenza del circuito

Per semplicità di progettazione del datapath, è stato deciso di far eseguire la parte di calcolo e di controllo del risultato con il dato uscente dal registro e non con quello uscente da MUX1, quindi il risultato finale in output si vede con un ciclo di clock di ritardo. Questo perché dal registro esce il valore immagazzinato nel ciclo di clock precedente e il primo ciclo di clock dopo l'inserimento di un pH valido serve per inizializzare il registro.

Sempre per quanto sopra spiegato, quando viene inserito all'avvio un valore di pH neutro, l'output non è immediato e nella transizione da RESET a FINE non viene alzato il bit FINE_OPERAZIONE; invece, nelle transizioni da uno degli stati di erogazione allo stato FINE viene subito alzata l'uscita FINE_OPERAZIONE perché nel registro è già disponibile il valore corretto da mettere in output.

7.3 Segnale g del datapath

Il segnale *g* del datapath serve per portare sugli output NCLK e PH_FINALE il valore calcolato solo e solamente quando viene terminata l'operazione, altrimenti tutti i bit vengono posti a 0.

Per realizzare un segnale di selezione tale, il segnale uscente da PH_NEUTRO non è sufficiente, perché in stato di ERRORE potrebbe essere inserito un valore di pH nell'intervallo di neutralità che entra nel registro e al ciclo di clock successivo verrebbe portato in output visto che PH_NEUTRO restituirebbe 1. Per ovviare a ciò, *g* viene calcolato come risultato dell'and logico tra il segnale PH_NEUTRO e il segnale SELETTORE_MUX, il quale vale 0 in stato di ERRORE e 1 in stato di FINE.