Elaborato SIS

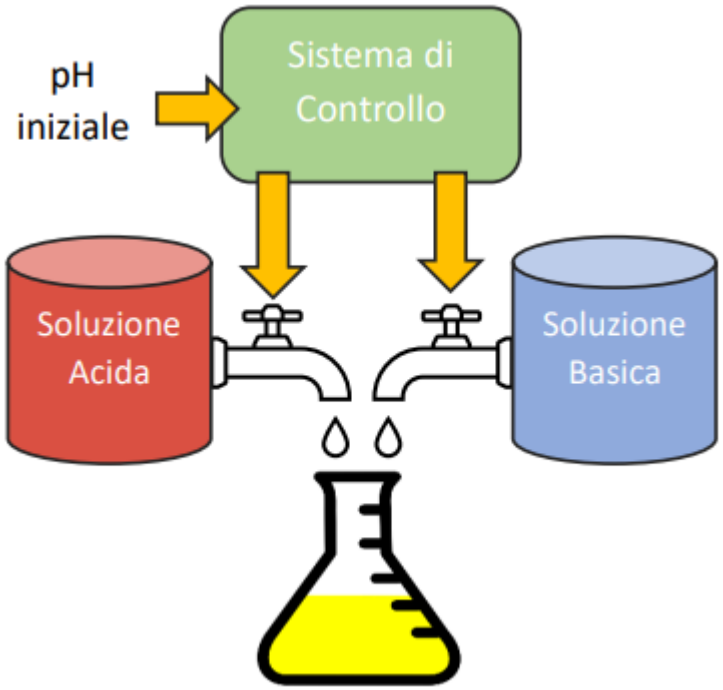
Matricole:
Javed Abdullah (VR471543)
Lucato Giacomo (VR472083)
Ruggiero Mattia (VR471519)

Anno 2021/2022

Sommario:

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.

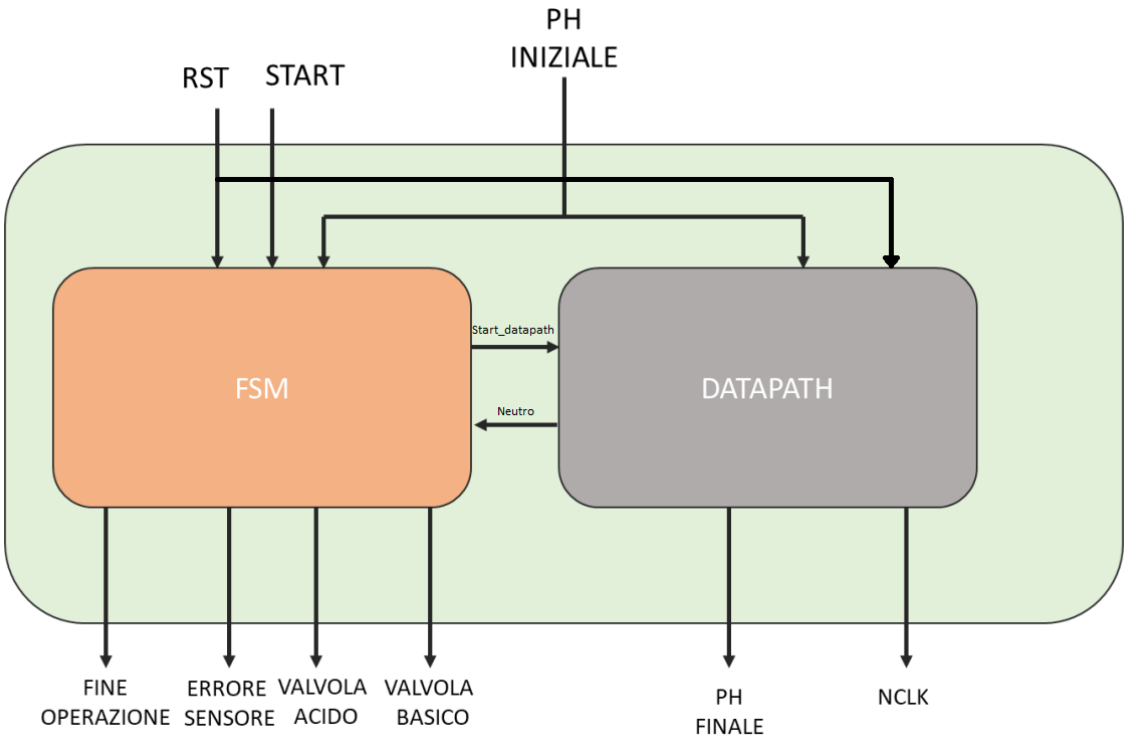


Il dispositivo presenterà seguenti ingressi e uscite:

RST [1 bit]= Se impostato a 1, ogni stato torna allo stato di Reset impostando tutti gli output a zero. Se posto a 0 entra nello stato di Start.
START [1 bit]= Se posto a 1 procede con l'elaborazione, altrimenti no.
pH [8 bit]= Esprime il valore del pH, con parte intera e parte frazionaria composte entrambe da 4 bit. Avrà dunque un valore minimo 0 e un valore massimo 15+15/16.
NEUTRO [1 bit]= (Ingresso aggiunto) Se posto a 1 il pH corrisponde a un valore neutro, se posto a 0 corrisponde a un valore acido o basico.

FINE_OPERAZIONE [1 bit]= Se posto a 1 termina il ciclo, se posto a 0 continua a ciclare.
ERRORE_SENSORE [1 bit]=Se posto a 1 significa che il valore del pH è superiore a 14. Se posto a 0, il valore del pH è compreso tra 0 e 14, estremi inclusi.
VALVOLA_ACIDO [1 bit]= Se posto a 1, il pH inserito ha valore (8,14] e viene attivata, se posto a 0 rimane disattivata.
VALVOLA_BASICO [1 bit]= Se posto a 1, il pH inserito ha valore [0, 7) e viene attivata, se poso a 0 rimane disattivata.

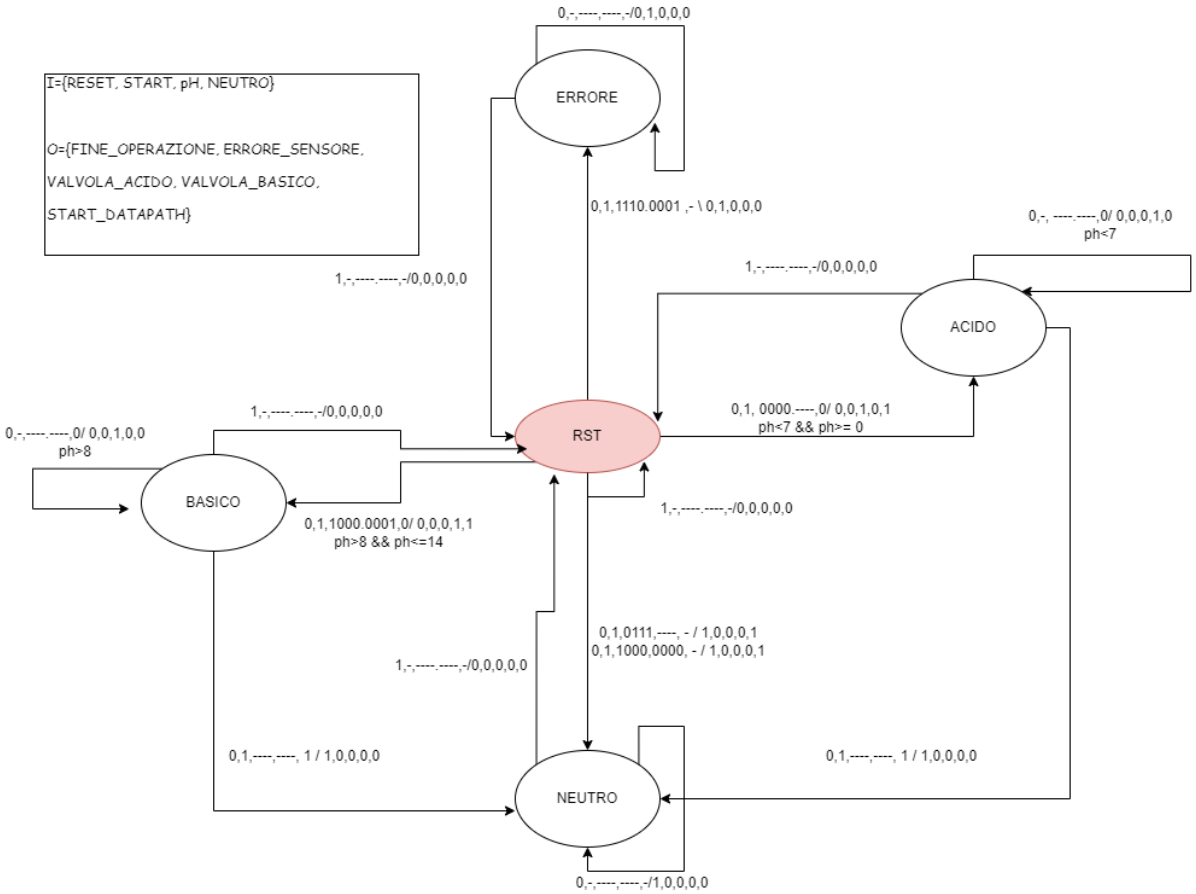
pH_FINALE [8 bit]= Composizione di bit identica a pH[8 bit], avrà sicuramente valore [7, 8], sarà il valore restituito dal circuito.
NCLK [8 bit]= Con valore massimo di 255, indica il numeri di cicli di clock avvenuti per rendere neutro il pH inserito dall'utente.



Controllore:

Per controllore intendiamo la macchina a stati finiti di Melay progettata, che presenta 4 ingressi {RESET, START,, pH, NEUTRO} e 5 uscite {FINE_OPERAZIONE, ERRORE_SENSORE, VALVOLA_ACIDO, VALVOLA_BASICO, START_DATAPATH}
Abbiamo individuato 5 stati:

- RST: che corrisponde allo stato iniziale di reset, dal quale comincia tutto se posto a 1. Da qui viene passato il valore del pH che viene dunque direzionato in base alla sua dimensione.
- ERRORE: In tale stato si arriva nel momento in cui viene inserito un pH dal valore maggiore di 14 (111-.---1) e viene dunque posto a 1. Se posto a 0, il valore del pH rientra nell'intervallo [0, 14]≈[0000.0000, 1110.0000].
- ACIDO: E' posto a 1 quando il pH inserito è acido [0, 7)≈[0000.0000, 0111.0000). Se posto a 0, potrebbe essere in ordine neutro o basico.
- BASICO: E' posto a 1 quando il pH inserito è basico (8, 14]≈(1000.0000, 1110.0000]. Se posto a 0, potrebbe essere neutro o acido.
- NEUTRO: E' posto a 1 quando il pH inserito è neutro [7, 8]≈[0111.0000, 1000.0000]. Se posto a 0, potrebbe essere acido o basico.



Datapath:

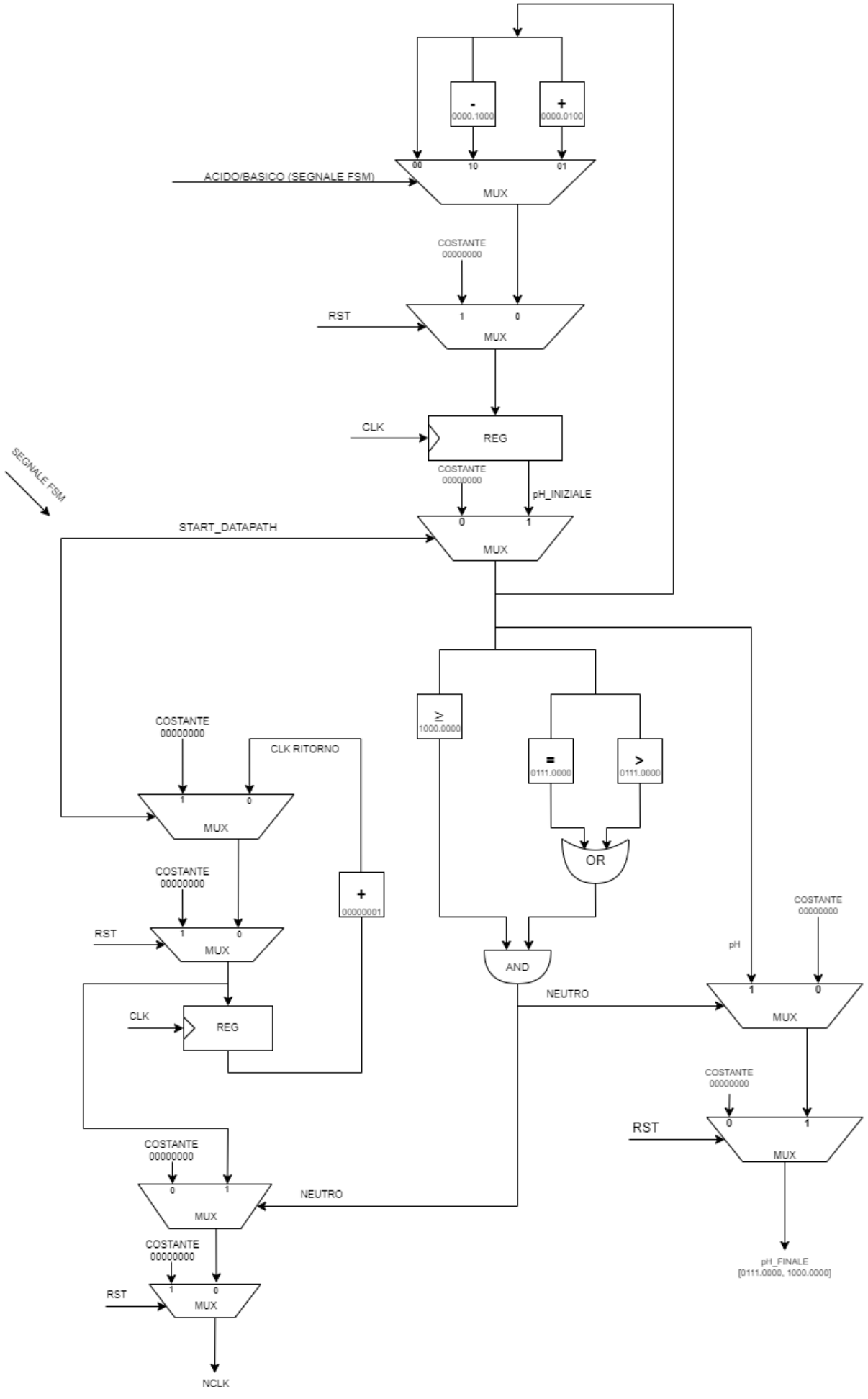
Si tratta della parte che essenzialmente svolge l'elaborazione.
Il nostro datapath riceve come input il segnale START_DATAPATH, il quale inizierà il conteggio dei cicli di clock e se necessario aumenterà o diminuirà il pH per renderlo neutro. Specifico il "se necessario": Se venisse inserito fin da subito un valore neutro, non c'è bisogno di fare alcuna operazione. Il segnale dunque finisce in due zone distinte del datapath.

1) Percorso calcolo: Il segnale entra nel datapath e in base al valore del pH, sceglie quale operazione fare o non fare. Se si tratta di un valore acido, verrà aggiunta una soluzione basica la quale incrementa il pH di 0,50 (0000.1000) a ogni ciclo di clock (aumentando NCLK a ogni ciclo) finché il pH si stabilizza ad un valore neutro. Ragionamento identico qualora venisse immesso un pH basico, aggiungendo una soluzione acida la quale decrementerà il pH di 0,25 (0000.0100) a ogni ciclo. Se invece il pH iniziale fosse già neutro o lo fosse diventato neutro in seguito all'azione delle valvole, non verrà fatta alcun'altra operazione, ma verrà controllato dal componente "comparatore", il quale entrerà nel MUX per dare in output il pH_FINALE.
2) Percorso clock: Il segnale START_DATAPATH entra anche nel 'reparto' di conteggio clock. Se però il valore inserito inizialmente fosse neutro, non ci sarebbe bisogno di

alcun ciclo di clock. Invece quando viene inserito un valore di pH acido o neutro, il conteggio dei cicli di clock comincerà, incrementando ovviamente di 1 a ogni ciclo (ogni operazione avvenuta). Inoltre tale reparto di conteggio è strettamente collegato all'altro reparto di calcolo, poiché deve costantemente controllare che il valore del pH sia diventato neutro o meno per interrompere o continuare il conteggio.

Per effettuare il controllo sul valore del pH neutro, sono stati utilizzati dei comparatori, quali ≥ e ≤. Il primo viene diviso a sua volta in due comparatori, = e >, i quali vengono collegati a un OR. Inoltre il ≥ e il ≤ sono collegati a un AND, dal quale si esce se e solo se il pH ha o ha assunto dunque valore compreso tra 7 e 8.

Al tutto abbiamo aggiunto dei registri i quali devono memorizzare i bit, quali quello del pH acido e basico durante il calcolo di somma o sottrazione, quello del pH finale e quello del contatore dei cicli di clock. Abbiamo inoltre fatto in modo che una volta terminata la sequenza di operazione e restituito il valore neutro, venga resettato tutto affinchè sia possibile svolgere più simulazioni, una dopo l'altra.


-
0000.1000
+
0000.0100
00
10
01
MUX
ACIDO/BASICO (SEGNALE FSM)
COSTANTE
00000000
RST
1
0
MUX
CLK
REG
COSTANTE
00000000
pH_INIZIALE
SEGNALE FSM
START_DATAPATH
0
1
MUX
≥
1000.0000
=
0111.0000
>
0111.0000
OR
AND
COSTANTE
00000000
CLK RITORNO
1
0
MUX
COSTANTE
00000000
+
00000001
RST
1
0
MUX
CLK
REG
pH
COSTANTE
00000000
1
0
MUX
NEUTRO
COSTANTE
00000000
RST
0
1
MUX
pH_FINALE
[0111.0000, 1000.0000]
COSTANTE
00000000
0
1
MUX
NEUTRO
COSTANTE
00000000
RST
1
0
MUX
NCLK

Statistiche circuito:

Di seguito le statistiche individuate rispettivamente nella FSM, nel datapath e nella FSMD.

1) FSM: Le statistiche seguenti sono quelle della macchina a stati finiti non ottimizzata, con una riduzione di 1 stato
Successivamente, abbiamo eseguito l'ottimizzazzione con il comando "source script.rugged", il quale ha permesso una riduzione drastica di letterali, con un leggero aumento di nodi.

```
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> rl fsm.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 : 4
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> print_stats
fsm             pi=11   po= 5   nodes=  7       latches= 2
lits(sop)= 242  #states(STG)=   4
```

```
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> rl fsm.blif
Warning: network `fsm.blif', node "RST" does not fanout
Warning: network `fsm.blif', node "START" does not fanout
Warning: network `fsm.blif', node "pH7" does not fanout
Warning: network `fsm.blif', node "pH6" does not fanout
Warning: network `fsm.blif', node "pH5" does not fanout
Warning: network `fsm.blif', node "pH4" does not fanout
Warning: network `fsm.blif', node "pH3" does not fanout
Warning: network `fsm.blif', node "pH2" does not fanout
Warning: network `fsm.blif', node "pH1" does not fanout
Warning: network `fsm.blif', node "pH0" does not fanout
Warning: network `fsm.blif', node "NEUTRO" does not fanout
sis> print_stats
fsm             pi=11   po= 5   nodes= 16       latches= 3
lits(sop)=  59  #states(STG)=   5
```

Possiamo dunque ritenerci soddisfatti di tale risultato.

2) DATAPATH: Il nostro file datapath.blif è in realtà composto da un collegamento tra due datapath "contatore.blif" e "rendeNeutro.blif". Quindi riporteremo le statistiche dei due file che compongono il principale, prima e dopo la loro ottimizzazione.

```
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> rl contatore.blif
Warning: network `sommatore_clk', node "B0" does not fanout
Warning: network `sommatore_clk', node "COUT" does not fanout
Warning: network `contatore', node "Uno" does not fanout
Warning: network `contatore', node "COUT" does not fanout
sis> print_stats
contatore       pi= 3   po= 8   nodes= 52       latches= 8
lits(sop)= 272
sis> rl rendeNeutro.blif
Warning: network `sommatore_ph', node "TEMP7" does not fanout
Warning: network `fammiSomma', node "TEMP7" does not fanout
Warning: network `rendeNeutro', node "ignore_sottrattore" does not fanout
Warning: network `rendeNeutro', node "TEMP7" does not fanout
sis> print_stats
rendeNeutro     pi=12   po= 9   nodes=161       latches= 8
lits(sop)= 780
sis>
```

Dopo l'ottimizzazione, come per l'FSM, il numero di letterali diminuisce drasticamente.

```
UC Berkeley, SIS 1.3.6 (compiled 2017-10-27 16:08:57)
sis> rl contatore.blif
Warning: network `sommatore_clk', node "B0" does not fanout
Warning: network `sommatore_clk', node "COUT" does not fanout
Warning: network `contatore', node "Uno" does not fanout
Warning: network `contatore', node "COUT" does not fanout
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> print_stats
contatore       pi= 3   po= 8   nodes= 20       latches= 8
lits(sop)=  75
sis> rl rendeNeutro.blif
Warning: network `sommatore_ph', node "TEMP7" does not fanout
Warning: network `fammiSomma', node "TEMP7" does not fanout
Warning: network `rendeNeutro', node "ignore_sottrattore" does not fanout
Warning: network `rendeNeutro', node "TEMP7" does not fanout
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> print_stats
rendeNeutro     pi=12   po= 9   nodes= 32       latches= 8
lits(sop)= 147
sis>
```

3) FSMD: L'FSMD, collegamento tra FSM e Datapath, riporta le seguenti caratteristiche:

```
sis> rl FSMD.blif
Warning: network `fsm.blif', node "RST" does not fanout
Warning: network `fsm.blif', node "START" does not fanout
Warning: network `fsm.blif', node "pH7" does not fanout
Warning: network `fsm.blif', node "pH6" does not fanout
Warning: network `fsm.blif', node "pH5" does not fanout
Warning: network `fsm.blif', node "pH4" does not fanout
Warning: network `fsm.blif', node "pH3" does not fanout
Warning: network `fsm.blif', node "pH2" does not fanout
Warning: network `fsm.blif', node "pH1" does not fanout
Warning: network `fsm.blif', node "pH0" does not fanout
Warning: network `fsm.blif', node "NEUTRO" does not fanout
sis> print_stats
FSMD            pi=10   po=20   nodes= 66       latches=19
lits(sop)= 281
```

In seguito all'ottimizzazione, le statistiche sono rimaste praticamente invariate, poichè l'FSM e il Datapath sono già stati ottimizzati

.

```
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> source script.rugged
sis> print_stats
FSMD            pi=10   po=20   nodes= 61       latches=19
lits(sop)= 282
sis>
```

Non ci preoccupiamo dei warning con la dicitura "does not fanout", poichè non influiscono minimamente sulla funzionalità dell'intero progetto.

Mapping:

In seguito all'ottimizzazione del circuito, avviene la mappatura per definire statistiche importanti quali l'area massima e il ritardo:

```
>>> before removing serial inverters <<<
# of outputs:           39
total gate area:        6272.00
maximum arrival time: (48.40,48.40)
maximum po slack:     (-7.20,-7.20)
minimum po slack:     (-48.40,-48.40)
total neg slack:      (-1118.80,-1118.80)
# of failing outputs:  39
>>> before removing parallel inverters <<<
# of outputs:           39
total gate area:        6048.00
maximum arrival time: (47.20,47.20)
maximum po slack:     (-7.20,-7.20)
minimum po slack:     (-47.20,-47.20)
total neg slack:      (-1042.60,-1042.60)
# of failing outputs:  39
# of outputs:           39
total gate area:        5824.00
maximum arrival time: (47.00,47.00)
maximum po slack:     (-7.20,-7.20)
minimum po slack:     (-47.00,-47.00)
total neg slack:      (-1034.40,-1034.40)
# of failing outputs:  39
```

Ne risulta una Total Gate Area di indicativamente quasi 6300.00 e un cammino critico di 48.40.

Scelte progettuali:

1) Abbiamo aggiunto in input e un output alla FSM, rispettivamente NEUTRO e START_DATAPATH. Il primo verifica lo stato di neutralità del pH inserito e il secondo serve per dare il segnale di start al datapath affinchè venga “attivato” e cominci la sequenza di calcoli.
2) Segnale di reset nel datapath per azzerare registri. Abbiamo inoltre fatto in modo che il segnale di reset andasse anche nel Datapath, in ogni registro, affinchè venisse azzerato ogni volta per dare in output i risultati interi, come quello del numero di cicli di clock e il pH finale.
3) Infine abbiamo diviso il controllo ≥ in un agglomerato composto dal controllo = e > con l’utilizzo di porte logiche quali OR e AND.