Università degli Studi di Verona

Dipartimento di Informatica

Laboratorio di Architettura degli Elaboratori

A.A. 2014/15

Elaborato SIS

COGNOME: Amore

NOME: Benedetta

MATRICOLA: 387279

COGNOME: Imola

NOME: Davide

MATRICOLA: 386238

INDICE:

TESTO DEL PROGETTO

Si progetti un dispositivo per il monitoraggio dei battiti cardiaci di un atleta sotto sforzo aerobico.

Il circuito sequenziale riceve in input il numero di battiti cardiaci per minuto (BPM), imposta i valori delle tre soglie:

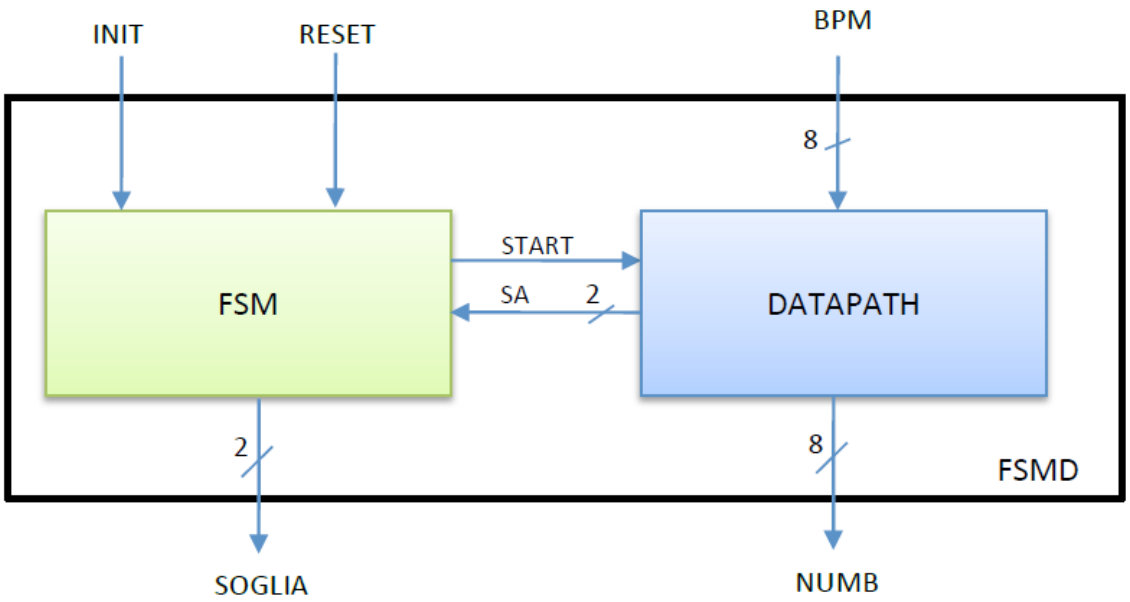
• 0 ≤ BPM < 160: sotto-soglia;

• 160 ≤ BPM ≤180: in-soglia;

• 180 < BPM < 256: oltre-soglia.

e presenta, in uscita, il livello di soglia in cui si trova attualmente il battito cardiaco e da quanti secondi.
Il circuito è composto da un controllore (FSM) e da un'unità di elaborazione (datapath).

SCHEMA GENERALE



Schema generale del FSMD.

Fonte: fornito dal docente

VINCOLI

- La lettura dei BPM dal datapath inizia solo con START posto a 1;
- La lettura dei BPM è fatta ogni secondo;
- Il valore dei BPM deve essere simulato;
- Ogni colpo di clock corrisponde a una botta di simulate.

SVOLGIMENTO RICHIESTE

IPOTESI AGGIUNTIVE

- Ogni secondo, in cui avviene una rilevazione del BPM eseguita dal rilevatore toracico e una lettura del medesimo BPM svolta dal datapath, corrisponde ad un ciclo di clock.

ARCHITETTURA GENERALE DEL CIRCUITO

Il circuito è una FSMD, cioè una macchina a stati finiti con datapath, composta da un controllore (FSM) e da un'unità di elaborazione (datapath).

IL CONTROLLORE
È la Finite State Machine,
Gli ingressi al controllore sono:
• INIT[1]
Quando INIT vale 1, il circuito inizia la rilevazione dei battiti cardiaci, il controllore deve passare nello stato conteggio dei secondi trascorsi nell'attuale soglia.
Finché INIT vale 0, non deve essere fatto alcun conteggio né indicato alcun valore in uscita.
• RESET[1]
Quando RESET vale 1, il controllore deve essere resettato, ovvero il contatore dei secondi deve essere posto a zero.
• SA[2]:
è lo State Signal che esce dal datapath, SA contiene il valore su due bit della soglia attuale (00 spento, 01 sotto-soglia,10 in-soglia, 11 oltre-soglia) e lo imposta sul controllore.

Le uscite dal controllore sono:
• SOGLIA[2]:
è il valore della soglia che il controllore riceve da SA, e lo riporta come output per indicare in quale soglia si trova il battito al momento corrente (00 spento, 01 sotto-soglia, 10 in-soglia, 11 oltre-soglia).

IL DATAPATH
Gli ingressi al datapath sono:
• BPM[8]:
è il valore dei battiti per minuto, codificato in binario.

Le uscite dal datapath sono:
• NUMB[8]:
indica i secondi trascorsi nell'attuale soglia.
• START[1]:
è il Control Signal tra controllore e datapath;
Quando START vale 1, il datapath inizia la lettura dei BPM, individua a quale stato di soglia si trova il battito dell'atleta e inizia il conteggio dei secondi trascorsi in quella soglia.

FUNZIONAMENTO
Si consideri che il rilevatore toracico misura ogni secondo un valore BPM.

Per accendere il controllore viene impostato INIT a 1; solo quando il Control Signal (START[1]) è posto a 1, il datapath avvia la lettura dei BPM, individua in quale livello di soglia collocare lo sforzo dell'atleta e inizia a contare i secondi in cui i BPM rientrano nella soglia attuale. Il datapath, mediante lo State Signal (SA[2]), fornisce al controllore il valore della soglia attuale e presenta come output (NUMB[8]) il tempo trascorso dall'inizio del conteggio, cioè il numero di secondi in cui il BPM si trova nella soglia attuale. Nel momento in cui il valore BPM non fa parte della soglia attuale, il datapath cambia stato, che inoltra al controllore, e inizia da zero il conteggio.
Il controllore presenta in uscita lo stato di SOGLIA[2] attuale, in corrispondenza del tempo trascorso in quello stato.

IL COTROLLORE

È un circuito sequenziale, ossia un circuito il cui comportamento non dipende solo dai valori assunti dai segnali in ingresso nel momento presente, ma anche dai loro valori negli istanti passati, cioè dalla storia del circuito stesso; pertanto esso deve tener traccia degli eventi antecedenti all'istante presente, che prendono il nome di stato.
Ogni circuito sequenziale è rappresentato con una FSM (Finite State Machine) in grado di mantenere le informazioni relative allo stato.
La FSM è definita dalla 6-upla M:< S, I, O, δ, λ, *s* >

S è l'insieme finito degli STATI (PRESENT STATE, NEXT STATE):
SPENTO, ATTESA, SOTTOSOGLIA, INSOGLIA, OLTRESOGLIA
Descritti su k bit

I è l'alfabeto in ingresso, ossia tutte le configurazioni ammissibili per le variabili primarie di ingresso (PRIMARY INPUT):
INIT RESET SA1 SA0
Descritti su n bit

O è l'alfabeto in uscita, ossia, date le configurazioni ammissibili per ogni variabile primaria in ingresso, sono i valori che si ottengono sulle variabili primarie in uscita (PRIMARY OUTPUT):
START SOGLIA1 SOGLIA0
Descritti su m bit

δ FUNZIONE DI STATO PROSSIMO:
δ = S x I -> S
E' una funzione del tipo $f(B^{k,n})->B^{k}$
Descrive l'evoluzione della macchina nell'istante presente in funzione dello stato e degli ingressi.

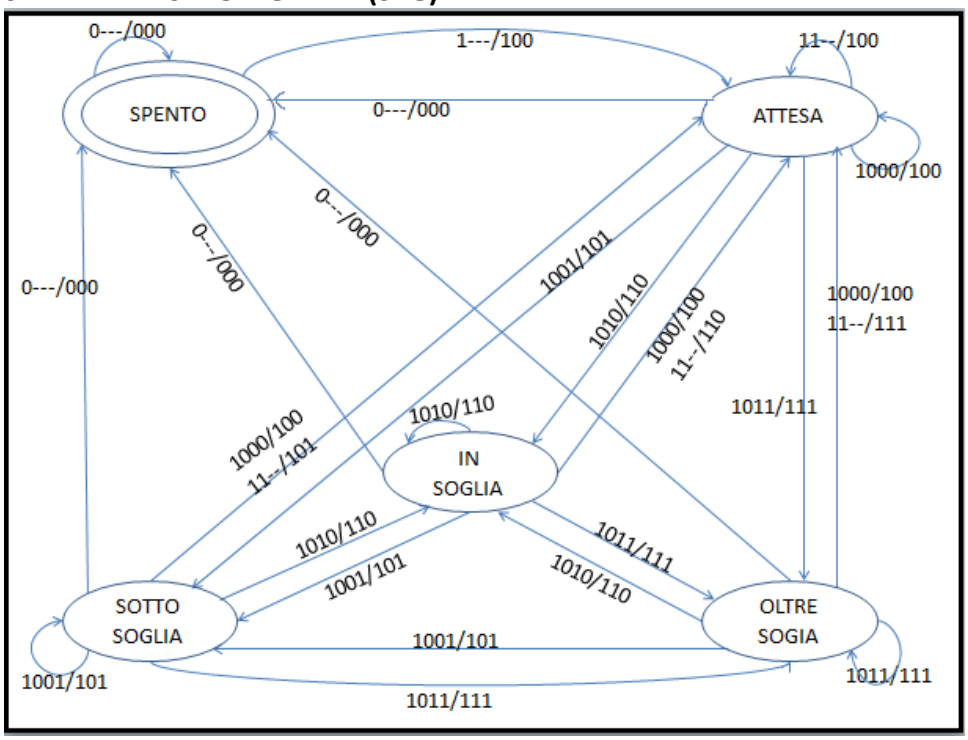
λ FUNZIONE D'USCITA
λ = S x I -> O (Mealy)
E' una funzione del tipo $f(B^{k,n})->B^{m}$
Determina il simbolo dell'alfabeto in uscita che la macchina deve produrre istante per istante, il valore in uscita dipende dallo stato presente in cui si trova la FSM e dalla combinazione in ingresso.

***s* STATO NOTO**
E' lo stato iniziale, da cui la macchina parte necessariamente al fine di fornire dei valori appropriati a ogni uscita.

RAPPRESENTAZIONE DELLA FSM

DIAGRAMMA DEGLI STATI DEL CONTROLLORE

STATE TRANSITION GRAPH (STG)



FSM: Schema del Controllore.
BIT INPUT/BIT OUTPUT:
INIT RESET SA1 SA0/
START SOGLIA1 SOGLIA0

Fonte:
riproduzione manuale

DESCRIZIONE DELL'IMMAGINE:
Lo State Transition Graph è costituito dagli archi e dai cinque nodi che rappresentano, rispettivamente, le transizioni tra uno e l'altro stato, alle quali corrisponde un insieme di valori in ingresso e un insieme di valori in uscita, e i cinque stati in cui, il circuito si sposta istante dopo istante. I cinque stati sono:

- SPENTO è lo stato tra spento e acceso, quando il controllore è spento si trova nello stato di SPENTO con il bit posto a 0, quando il controllore è acceso si trova nello stato SPENTO con il bit posto a 1;
- ATTESA è lo stato di attesa, quando il controllore viene acceso, mediante il bit START, fornisce tale informazione al datapath e attende di ricevere i valori di soglia SA1 SA0 percepiti dal rilevatore di battiti;
- SOTTOSOGLIA, INSOGLIA, OLTRESOGLIA sono gli stati che individuano entro quale soglia rientrano i battiti, quando il controllore riceve un valore di soglia SA1 SA0 si sposta in un dei tre stati e fornisce in uscita il valore della soglia in cui si trova.

SEQUENZA DEI BIT INPUT/OUTPUT SCRITTI SUGLI ARCHI: INIT RESET SA1 SA0 / START SOGLIA1 SOGLIA0

DA	A	bit I/O	Descrizione
SPENTO	SPENTO	0---/000	SPENTO ==0 per qualsiasi valore in ingresso, controllore è spento, uscite a 0
SPENTO	ATTESA	1---/100	INIT==1 per qualsiasi valore in ingresso, il controllore viene acceso, START==1 controllore attende di ricevere un valore di soglia, SA1 SA0 = 00
ATTESA	ATTESA	11--/100	RESET==1, il controllore rimane acceso, in attesa e mostra SA1 SA0 = 00
ATTESA	ATTESA	1000/100	RESET==0, il controllore rimane acceso, in attesa e mostra SA1 SA0 = 00
ATTESA	SPENTO	0---/000	INIT==0 per qualsiasi valore in input, il controllore viene spento, output a 0
ATTESA	SOTTOSOGLIA	1001/101	Il datapath riceve il valore SOTTOSOGLIA=01, transizione dallo stato ATTESA allo stato SOTTOSOGLIA, il controllore mostra SA1 SA0 = 01
ATTESA	INSOGLIA	1010/110	Il datapath riceve il valore INSOGLIA = 10, transizione dallo stato ATTESA allo stato INSOGLIA, il controllore mostra SA1 SA0 = 10
ATTESA	OLTRESOGLIA	1011/111	Il datapath riceve il valore OLTRESOGLIA=11, transizione dallo stato ATTESA allo stato OLTRESOGLIA, il controllore mostra SA1 SA0 = 11
SOTTOSOGLIA	SOTTOSOGLIA	1001/101	Il datapath riceve il valore SOTTOSOGLIA = 01, rimane nello stato SOTTOSOGLIA, il controllore mostra SA1 SA0 = 01
SOTTOSOGLIA	ATTESA	1000/100	Il datapath non riceve alcun valore di soglia = 00, transizione dallo stato SOTTOSOGLIA allo stato ATTESA il controllore mostra SA1 SA0 = 00
SOTTOSOGLIA	ATTESA	11--/101	Il controllore viene resettato, per qualsiasi valore di soglia, torna in ATTESA e mostra l'ultimo stato di soglia in cui si trova il datapath
SOTTOSOGLIA	SPENTO	0---/000	INIT==0 per qualsiasi valore in input, il controllore viene spento, output a 0
SOTTOSOGLIA	OLTRESOGLIA	1011/111	Il datapath riceve il valore di soglia = 11, transizione dallo stato SOTTOSOGLIA allo stato OLTRESOGLIA, il controllore mostra SA1 SA0 = 11
SOTTOSOGLIA	INSOGLIA	1010/110	Il datapath riceve il valore di soglia = 10, transizione dallo stato SOTTOSOGLIA allo stato INSOGLIA il controllore mostra SA1 SA0 = 10
INSOGLIA	INSOGLIA	1010/110	Il datapath riceve il valore INSOGLIA = 10, rimane nello stato INSOGLIA, il controllore mostra SA1 SA0 = 10
INSOGLIA	ATTESA	1000/100	Il datapath non riceve alcun valore di soglia = 00, transizione dallo stato INSOGLIA allo stato ATTESA il controllore mostra SA1 SA0 = 00
INSOGLIA	ATTESA	11--/110	Il controllore viene resettato, per qualsiasi valore di soglia, torna in ATTESA e mostra l'ultimo stato di soglia in cui si trova il data path
INSOGLIA	SPENTO	0--/000	INIT==0 per qualsiasi valore in input, il controllore viene spento, output a 0
INSOGLIA	SOTTOSOGLIA	1001/101	Il datapath riceve il valore di soglia = 01, transizione dallo stato INSOGLIA allo stato SOTTOSOGLIA il controllore mostra SA1 SA0 = 01
INSOGLIA	OLTRESOGLIA	1011/111	Il datapath riceve il valore di soglia = 11, transizione dallo stato INSOGLIA allo stato OLTRESOGLIA il controllore mostra SA1 SA0 = 11
OLTRESOGLIA	OLTRESOGLIA	1011/111	Il datapath riceve il valore OLTRESOGLIA = 11, rimane nello stato OLTRESOGLIA, il controllore mostra SA1 SA0 = 11
OLTRESOGLIA	ATTESA	1000/100	Il datapath non riceve alcun valore di soglia = 00, transizione dallo stato OLTRESOGLIA allo stato ATTESA il controllore mostra SA1 SA0 = 00
OLTRESOGLIA	ATTESA	11--/111	Il controllore viene resettato, per qualsiasi valore di soglia, torna in ATTESA e mostra l'ultimo stato di soglia in cui si trova il data path
OLTRESOGLIA	SPENTO	0---/000	INIT==0 per qualsiasi valore in input, il controllore viene spento, output a 0
OLTRESOGLIA	SOTTOSOGLIA	1001/101	Il datapath riceve il valore di soglia = 01, transizione dallo stato OLTRESOGLIA allo stato SOTTOSOGLIA il controllore mostra SA1 SA0 = 00
OLTRESOGLIA	INSOGLIA	1010/110	Il datapath riceve il valore di soglia = 10, transizione dallo stato OLTRESOGLIA allo stato INSOGLIA il controllore mostra SA1 SA0 = 10

STATE TRANSITION TABLE (STT)

La STT non codificata è la rappresentazione in forma tabulare del grafo delle transizioni.

S \ I	INIT=0		INIT=1 RESET=1		INIT=1 RESET=0					
	---		--		SOTTOSOGLIA		INSOGLIA		OLTRESOGLIA	
SPENTO	INIT	000	ATTESA	100	0000	000	0000	000	0000	000
ATTESA	INIT	000	ATTESA	100	SOTTOSOGLIA	101	INSOGLIA	110	OLTRESOGLIA	111
SOTTOSOGLIA	INIT	000	ATTESA	101	SOTTOSOGLIA	101	INSOGLIA	110	OLTRESOGLIA	111
INSOGLIA	INIT	000	ATTESA	110	SOTTOSOGLIA	101	INSOGLIA	110	OLTRESOGLIA	111
OLTRESOGLIA	INIT	000	ATTESA	111	SOTTOSOGLIA	101	INSOGLIA	110	OLTRESOGLIA	111

MODELLAZIONE E MINIMIZZAZIONE DELLA STT IN SIS

Per modellare, in SIS, il circuito sequenziale, relativo alla STT che abbiamo prodotto, seguiamo questi passaggi:

1. La State Transition Table è descritta nella sezione delimitata dalle keyword .start_kiss .end_kiss, dove
   Le prime cinque righe indicano:
   .i 4 #quattro ingressi
   .o 3 #tre uscite
   .p 26 #ventisei transizioni
   .s 5 #cinque stati
   .r SPENTO #stato noto: SPENTO
   Le transizioni sono specificate come un insieme di righe che riportano in ordine il valore degli ingressi, lo stato presente, lo stato prossimo, il valore delle uscite;
2. Gli stati sono minimizzati con il comando sis>state_minimize stamina, che cerca gli stati "raggiungibili", ossia tutti i Next State che sono tra loro equivalenti e, per ogni combinazione stato "raggiungibile"- Pirmay Input, calcola l'XOR bit a bit dei Primary Output generati, i due stati non sono equivalenti se almeno un bit non è 0; per effettuare detta ricerca usa una struttura BDD.

Nel nostro caso, dopo aver lanciato il comando di minimizzazione, il programma non è riuscito a ridurre il numero degli stati:

```
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 : 5
Number of states in minimized machine : 5
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> print_stats
fsm             pi= 4   po= 3   nodes=  6       latches= 3
lits(sop)=  35  #states(STG)=   5
```

Risultato del comando di minimizzazione degli stati.

Fonte: prodotto da SIS.

3. Gli stati sono codificati automaticamente con la keyword sis>state_assign jedi;
4. Il comando sis>state_assign jedi genera automaticamente le funzioni d'uscita e di stato prossimo;

5. Le funzioni sono minimizzate con lo script.rugged, che si basa su un algoritmo NP completo, e adotta una struttura BDD (Binary Decision Diagram), un grafo che riduce le dimensioni delle strutture dati. E' adatto per la sintesi approssimata multilivello, per ridurre l'area (calcolata sul numero di letterali presenti nella funzione) e il ritardo dei circuiti. A tale scopo lo script dispone di algoritmi di ristrutturazione e minimizzazione del DAG (Direct Aciclic Graph), con cui effettua sia trasformazioni algebriche, come sweep, eliminate, fx, resub, simplify e full simplify, sia trasformazioni booleane: individua l'External Don't Care Set, ossia le condizioni di indifferenza esterna suddivise in Controllability Don't Care Set e Observability Don't Care Set. Il primo insieme include tutte le configurazioni che non potranno mai presentarsi all'ingresso di un nodo; il secondo insieme include tutti i valori in uscita da un nodo che, a partire dalle configurazioni accettate in input, non potranno mai essere generati.
   In generale, lo script.rugged fornisce il miglior risultato, in termini di minimizzazione, tra tutti gli script predefiniti di SIS.

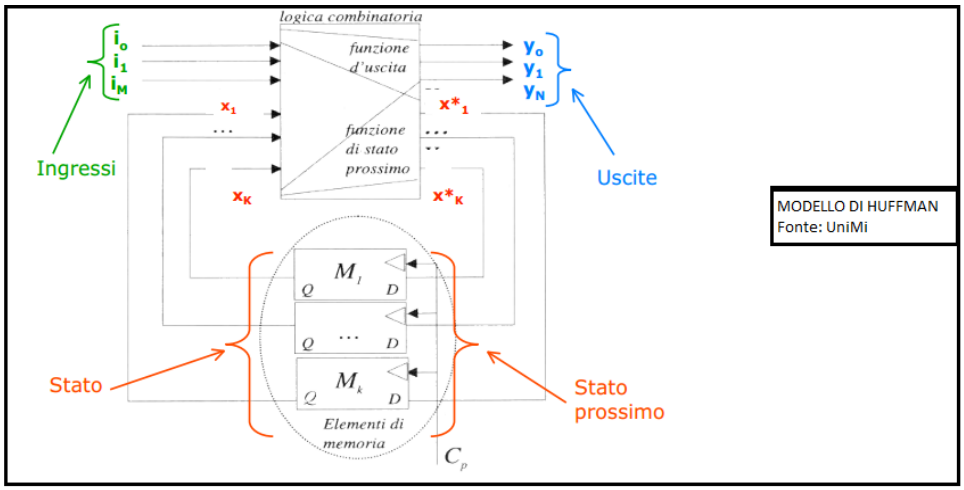
Lanciando lo script.rugged sulla FSM, siamo riusciti ad ottenere questa riduzione sui letterali:

```
sis> source script.rugged
sis> print_stats
fsm              pi= 4   po= 3   nodes=  6       latches= 3
lits(sop)=  19   #states(STG)=   5
```

Risultato del comando di minimizzazione dei letterali.

Fonte: prodotto da sis.

CIRCUITO SUL MODELLO DI HUFFMAN



MODELLO DI HUFFMAN
Fonte: UniMi

Il modello di Huffman divide la logica combinatoria dall'unità di memorizzazione.

MAPPING TECNOLIGICO

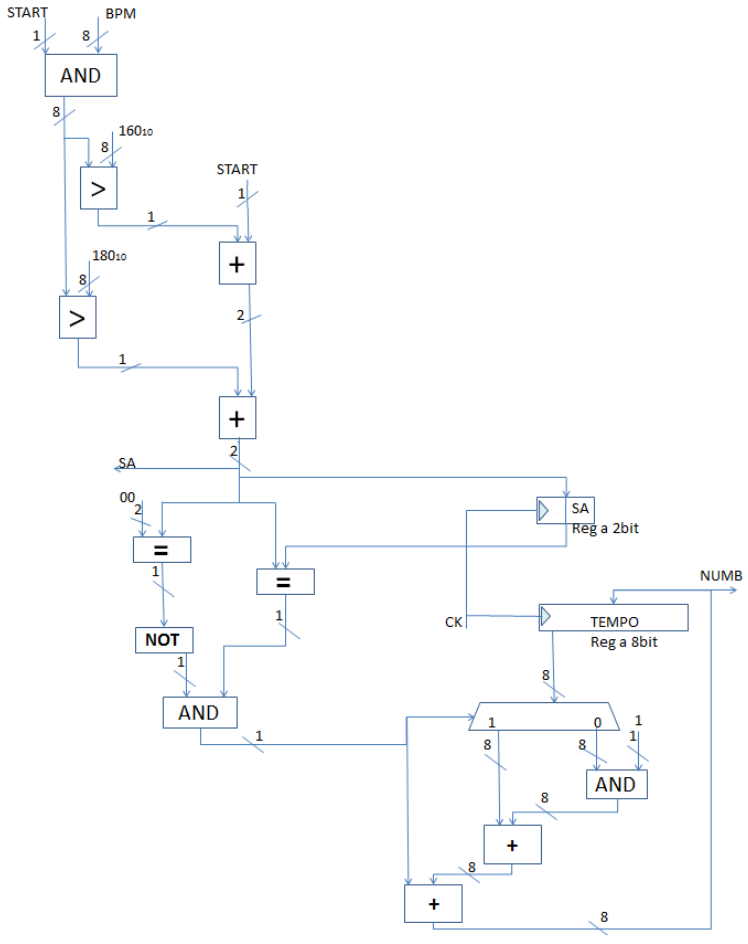
Il mapping tecnologico è necessario per ridurre i tempi di risposta e l'area della parte di logica combinatoria, infatti l'insieme di porte logiche, che implementano funzioni diverse e hanno tempi di risposta diversi, si trasforma in una foresta di alberi di porte logiche NAND, i quali hanno lo stesso ritardo. Inoltre il tree mapping, cioè la mappatura della foresta di NAND è svolta con una tecnica greedy che, partendo dalla sezione più vicina agli output, cerca di ristrutturare il circuito con il minor numero di alberi NAND contenuti nella libreria.
Una volta ottimizzato il circuito, dobbiamo mapparlo con una libreria, in modo da avere delle statistiche più verosimili per quanto area-ritardo. La libreria a noi richiesta è synch.genlib.
Una volta mappato, il circuito presenta le seguenti statistiche:

```
>>> before removing serial inverters <<<
# of outputs:           20
total gate area:        2344.00
maximum arrival time: (31.20,31.20)
maximum po slack:     (-0.40,-0.40)
minimum po slack:     (-31.20,-31.20)
total neg slack:      (-253.80,-253.80)
# of failing outputs:  20
>>> before removing parallel inverters <<<
# of outputs:           20
total gate area:        2344.00
maximum arrival time: (31.20,31.20)
maximum po slack:     (-0.40,-0.40)
minimum po slack:     (-31.20,-31.20)
total neg slack:      (-253.80,-253.80)
# of failing outputs:  20
# of outputs:           20
total gate area:        2296.00
maximum arrival time: (31.20,31.20)
maximum po slack:     (-0.40,-0.40)
minimum po slack:     (-31.20,-31.20)
total neg slack:      (-253.80,-253.80)
# of failing outputs:  20
```

Statistiche relative alla mappatura del circuito con la libreria synch.genlib: l'area complessiva è 2296.00, il cammino critico è 31.20

Fonte: realizzato da SIS.

IL DATAPATH (ARCHITETTURA)



Unità di elaborazione: DATAPATH.

Fonte:
riproduzine manuale.

Il datapath è un grafo diretto e ciclico, costituito da nodi di ingresso, da nodi di uscita, da nodi di elaborazione e da archi direzionali, ossia che terminano con una freccia che indica il verso in cui si spostano i dati, e ciclici, infatti un datapath, al suo interno, può presentare dei cicli che sono risolti con i componenti combinatori che eseguono e risolvono i cicli.
Ogni arco viene contrassegnato con un'etichetta: SEGNALE (nome dell'arco o della transizione cui si riferisce) AMPIEZZA (numero di bit trasportati), in questo modo:
S/A; S[A];

I nodi di elaborazione appartengono a librerie di componenti atte a formalizzare i datapath; i componenti si dividono in:

1. Sequenziali, come le memorie;
2. Combinatori, di selezione, aritmetici e logici.

DESCRIZIONE DELL'IMMAGINE

I componenti sequenziali che abbiamo usato per creare il datapath sono:

- Registro parallelo-parallelo a 2 bit per memorizzare il livello di soglia (SA[2]);
- Registro parallelo-parallelo a 8 bit per memorizzare il numero di secondi in cui i BPM rientrano nel livello di soglia rilevato (TEMPO[8]).

Entrambi comandati da clock.

I componenti combinatori che abbiamo usato per creare il datapath sono:

- Operatore logico AND bit a bit (A[1]/B[8]-O[8]);
- Operatore logico AND bit a bit (A[1]/B[1]-O[1])
- Operatore logico NOT bit a bit;
- Operatore aritmetico di confronto "maggiore" > (A[8]/B[8]-O[1]);
- Operatore aritmetico di confronto "uguale" = (A[2]/B[2]-O[1]);
- Operatore aritmetico sommatore + (A[2]/B[2]-O[2]);
- Operatore aritmetico sommatore + (A[8]/B[8]-O[8]);
- Operatore aritmetico sommatore + (A[8]/B[1]-O[8]);
- Componente di selezione demultiplexer (I[8]-O_1[8]/O_2[8]-sel[1]).

Il primo operatore logico esegue un'AND bit a bit tra il valore di START[1], in uscita dal controllore e ricevuto sul Control Signal del datapath, e il BPM[8] rilevato, se START[1] = 0, significa che il controllore è spento e solo in questo caso il datapath fornirebbe un livello di soglia (SA1 SA0) pari a 00, perché in uscita della AND si avrebbero 8 bit a zero; se START[1] = 1, significa che il controllore è acceso e in attesa di ricevere dei segnali BPM, quindi in uscita dalla AND si avrà il valore a 8 bit del BPM.
Per ottenere il livello di soglia, il BPM entra in due operatori di confronto "maggiore" >, il primo valuta BPM > 160 (codificato in binario su 8 bit), il secondo valuta BPM > 180 (codificato in binario su 8 bit), entrambi con un'uscita da un bit, che vale 1 se BPM è maggiore del valore con cui è confrontato, oppure vale 0 se BPM non è maggiore del valore con cui è confrontato.
Il livello di SOTTOSOGLIA (SA1 SA0 = 01) si ottiene quado il primo confronto BPM > 160 = 0, significa che BPM non è maggiore di 160, si somma 0+START[1] = 01_2, anche il secondo confronto BPM > 180 = 0, significa che BPM non è maggiore di 180, la somma tra questo bit e il valore calcolato dal primo sommatore $0+01_2 = 01_2$ risulta essere l'effettivo livello di soglia, cioè SOTTOSOGLIA.
Il livello INSOGLIA (SA1 SA0 = 10) si ottiene quado il primo confronto BPM > 160 = 1, significa che BPM è maggiore di 160, si somma 1+START[1] = 10_2, invece il secondo confronto BPM > 180 = 0,

significa che BPM non è maggiore di 180, la somma tra questo bit e il valore calcolato dal primo sommatore $0+10_2 = 10_2$ risulta essere l'effettivo livello di soglia, cioè INSOGLIA.
Il livello di OLTRESOGLIA (SA1 SA0 = 11) si ottiene quado il primo confronto BPM > 160 = 1, significa che BPM è maggiore di 160, si somma 1+START[1] = 10_2, anche il secondo confronto BPM > 180 = 1, significa che BPM è maggiore di 180, la somma tra questo bit e il valore calcolato dal primo sommatore $1+10_2 = 11_2$ risulta essere l'effettivo livello di soglia, cioè OLTRESOGLIA.
È necessario sommare il primo valore di confronto con START[1] perché il livello sottosoglia comprende l'intervallo chiuso [0..160], pertanto solo se il controllore è spento (START = 0) o il datapath stesso ha qualche malfunzionamento, il valore di soglia SA1 SA0 = 00.
Il livello di soglia è indirizzato verso il controllore, sullo State Signal SA[2].
In seguito, il livello di soglia entra in due operatori di confrontato "uguale" =, con un bit in uscita che vale 1 solo se SA è uguale ad uno dei valori con cui è confrontato.
Il primo blocco valuta se SA è uguale a 00, uscita a 1 (successivamente negata), se ciò avviene il livello di soglia sicuramente non è uguale al livello salvato nel registro SA[2] (uscita a 0), perciò il risultato della AND è 0, il quale, a sua volta, seleziona nel DEMUX l'uscita 0 (su 8 bit) che resetta il valore del tempo memorizzato nel registro TEMPO[8].
Altrimenti se SA non è uguale a 00, uscita a 0 (successivamente negata), se ciò avviene il secondo blocco valuta se il livello di soglia è uguale al valore memorizzato nel registro SA[2] al colpo di clock precedente. Se SA è uguale alla soglia precedente, l'uscita è posta a 1, perciò il risultato della AND è 1, il quale, a sua volta, selezione nel DEMUX l'uscita 1 che riporta il numero di secondi memorizzati nel registro NUMB[8] e lo incrementa di 1. Se SA non è uguale alla soglia precedente, l'uscita è posta a 0, perciò il risultato della AND è 0, il quale, a sua volta, selezione nel DEMUX l'uscita 0 (su 8 bit) che riporta il numero di secondi memorizzati nel registro NUMB[8] e lo resetta nel blocco AND con 0, le somme successive sono tra due valori 0 quindi nel registro del tempo verrà memorizzato 0 e NUMB = 0.

MODELLAZIONE E MINIMIZZAZIONE DEL DATAPATH IN SIS

Il programma relativo al datapath è diviso in diversi file, ognuno dei quali contiene un .model, cioè una funzione parametrizzata che descrive un elemento del circuito; i principali sono i registri, ossia i componenti di memorizzazione dei dati e le unità di elaborazione o unità funzionali. Nel programma principale sono chiamate sia le funzioni per creare la struttura fisica dell'unità di elaborazione sia quelle che implementano le sue funzionalità.
Per includere un componente all'interno di un file .blif abbiamo usato le seguenti istruzioni:

.subckt nomeComponente parametroFormaleInput1 = parametroAttualeInput1… parametroFormaleInputN = parametroAttualeInputN parametroFormaleOutput1 = parametroAttualeOutput1 … parametroFormaleOutputN = parametroAttualeOutputN

.search nomeFileComponente.blif

La prima istruzione permette di includere un componente di nome "nomeComponente" e di collegare i suoi parametri formali con quelli attuali. La seconda istruzione indica a SIS quale sia il file che contiene la definizione del componente "nomeComponente".

Il datapath generato presenta le seguenti statistiche:

```
sis> read_blif data_path.blif
"zero.blif", line 17: .endrch (null)
"zero.blif", line 17: .endrch (null)
Warning: network `data_path', node "one" does not fanout
Warning: network `data_path', node "COut160" does not fanout
Warning: network `data_path', node "COut180" does not fanout
Warning: network `data_path', node "co" does not fanout
Warning: network `data_path', node "co1" does not fanout
sis> print_stats
data_path       pi= 9   po=10   nodes=136       latches=10
lits(sop)= 510
```

Statistiche relative al datapath prima della minimizzazione.

Fonte: realizzato da SIS.

I warning non sono preoccupanti, in quanto non influiscono con il risultato del nostro circuito.
A noi interessa ridurre l'area del circuito, dopo alcuni tentativi abbiamo ottenuto questi risultati:

```
sis> print_stats
data_path       pi= 9   po=10   nodes= 20       latches=10
lits(sop)=  98
```

Statistiche relative al datapath dopo la minimizzazione.

Fonte: realizzato da SIS.

FSMD

Per unificare i due componenti abbiamo creato un .model main, in cui è chiamata la funzione calcStart.blif che calcola il valore dello START (START=1 when INIT=1 and RESET=0) e attiva il datapath alla rilevazione dei BPM. Successivamente sono chiamate le funzioni relative al datapath e al controllore, data_path.blif e fsm.blif; solo in questo modo la FSMD fornisce un output corretto.
Applicando la minimizzazione sul model main otteniamo:

```
sis> print_stats
main            pi=10   po=10   nodes=141       latches=10
lits(sop)= 513
sis> source script.rugged
sis> print_stats
main            pi=10   po=10   nodes= 21       latches=10
lits(sop)= 100
```

Minimizzazione del model main.

Fonte: realizzato da SIS.

OSSERVAZIONI

1. Il circuito lavora a 1 Hz, infatti per ogni colpo di clock BPM, si ha che Tck = 1/f (rapporto 1:1);
2. La combinazione 1000/100, indica che il controllore è acceso e non resettato con livello di soglia 00, ciò si verifica solo nel caso in cui si scollega il rilevatore di battiti dal datapath o il datapath stesso ha qualche malfunzionamento, in quanto il livello sottosoglia è compreso nell'intervallo chiuso [0..160];

3. Abbiamo scelto di definire delle costanti, come zero.blif e one.blif, utilizzate più volte nel codice.

IMPLEMENTAZIONE IN LINGUAGGIO C

Forniamo in allegato l'implementazione in linguaggio C, che abbiamo usato come bozza e spunto da cui iniziare per fornire la giusta soluzione del progetto, per questo motivo il programma non corrisponde interamente alla soluzione finale del nostro progetto, ma risponde correttamente alle richieste fatte.

MATERIALE USATO PER LA REALIZZAZIONE

IMMAGINI: create con Power Point, modellate e modificate con Photo Shop;
IMMAGINI: screenshot di schermata per esecuzione in SIS;
TESTO: Microsoft Word;
CODICE .blif: programma di editing di (Sublime Text), compilatore SIS;
CODICE .c: programma di editing di (Sublime Text), compilatore gcc.

Per la maggior parte del tempo abbiamo lavorato in coppia, nei laboratori dell'università, abbiamo scelto di condividere il materiale su Google Drive per poter lavorare da casa mantenendoci in contatto costante.

BIBLIOGRAFIA

LIBRO DI TESTO: "*Progettazione digitale*" Seconda edizione, F.Fummi, M. Sami, C. Silvano, McGraw-Hill, Milano, 2007.

FOTOCOPIE E MATERIALE FORNITI DAL DOCENTE.

SITO: dispensa online UNIMI (immagine).