Laboratorio di Architettura degli Elaboratori

A.A. 2017/18

RELAZIONE
ELABORATO SIS

dispositivo per la gestione intelligente del consumo di energia elettrica all'interno di un sistema domotico

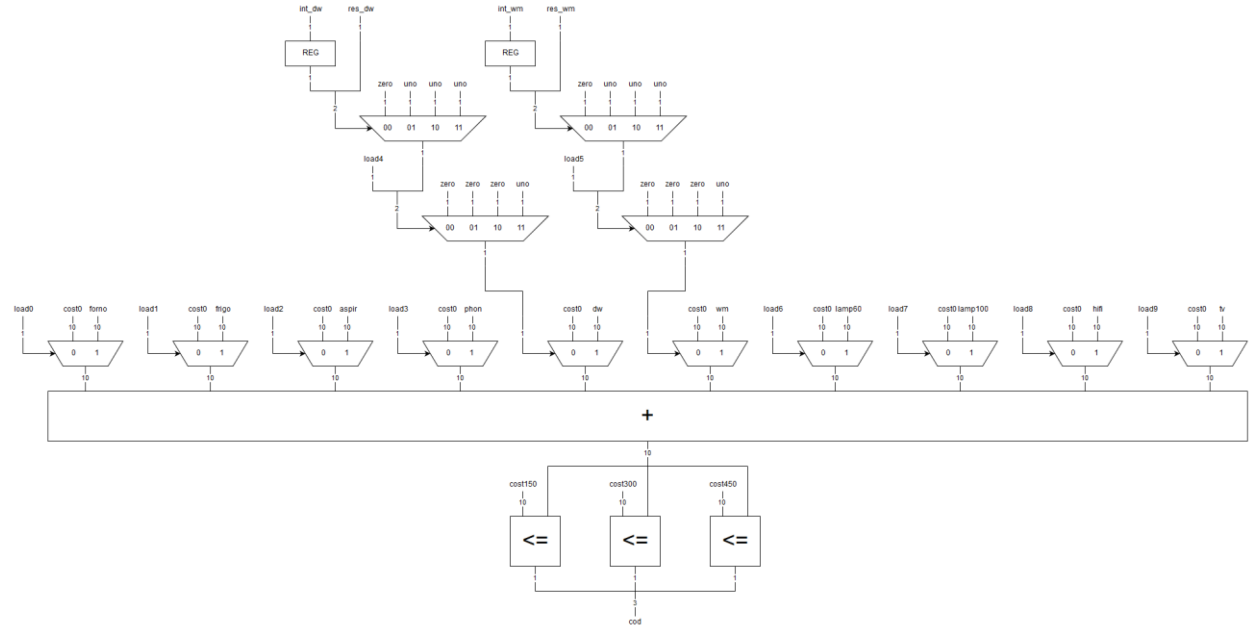
Membri del gruppo

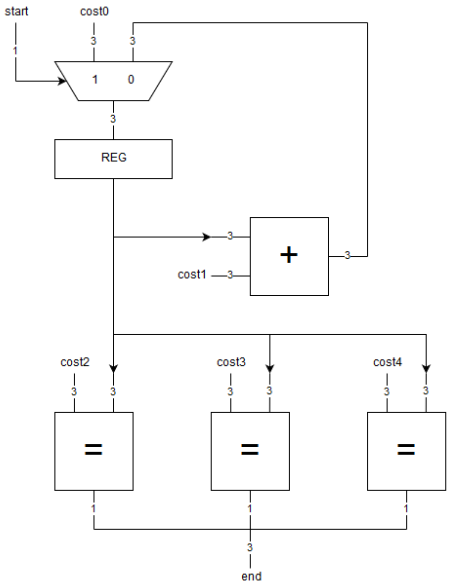
Bonfante Luca – VR

Bordeianu Bogdan – VR

Zampieri Davide – VR421649

ARCHITETTURA DEL DATA-PATH





Composizione:

- Un *codificatore*, che ha il compito di sommare, in base allo stato acceso/spento dei dieci dispositivi (segnale di load), i consumi istantanei e comparare la somma ottenuta con gli estremi delle fasce.
- Un *contatore*, che ha il compito di iniziare a contare da 0 quando riceve 1 come bit di start.

Funzionamento del codificatore:

- I segnali di load forniti in ingresso diventano i selettori di dieci multiplexer a due ingressi. Ogni multiplexer restituisce una costante di tutti zeri (se il selettore vale 0) o una costante che rappresenta il consumo istantaneo del dispositivo (se il selettore vale 1).

- La lavastoviglie e la lavatrice hanno un ulteriore multiplexer che, tenendo conto anche del valore dell'interruttore corrispondente, restituisce un nuovo segnale di load secondo la seguente tabella.

LOAD	INTERRUTTORE	NEW_LOAD
0	0	0
0	1	0
1	0	0
1	1	1

- Per determinare il valore dei suddetti interruttori è necessario implementare due ulteriori multiplexer che hanno come segnali di selezione il valore dell'interruttore al ciclo precedente e il rispettivo segnale di reset (dato in input dall'utente nel ciclo corrente). Ogni multiplexer restituisce un nuovo valore per ognuno dei due interruttori secondo la seguente tabella.

INTERRUTTORE (ciclo precedente)	RESET	INTERRUTTORE (ciclo corrente)
0	0	0
0	1	1
1	0	1
1	1	1

- Dopodiché le dieci costanti in uscita dai dieci multiplexer vengono sommate utilizzando nove sommatori in cascata (il primo sommatore somma le prime due costanti, il secondo somma il risultato del primo con la terza costante e così via).
- Poi il risultato dell'ultimo sommatore viene mandato in ingresso a tre comparatori, i quali controllano se la somma è minore o uguale a 150 daW, 300 daW o 450 daW.
- In uscita avremo tre bit (dai comparatori) che serviranno successivamente al *controllore* per determinare la codifica della fascia di consumo istantanea, più altri due bit degli interruttori.

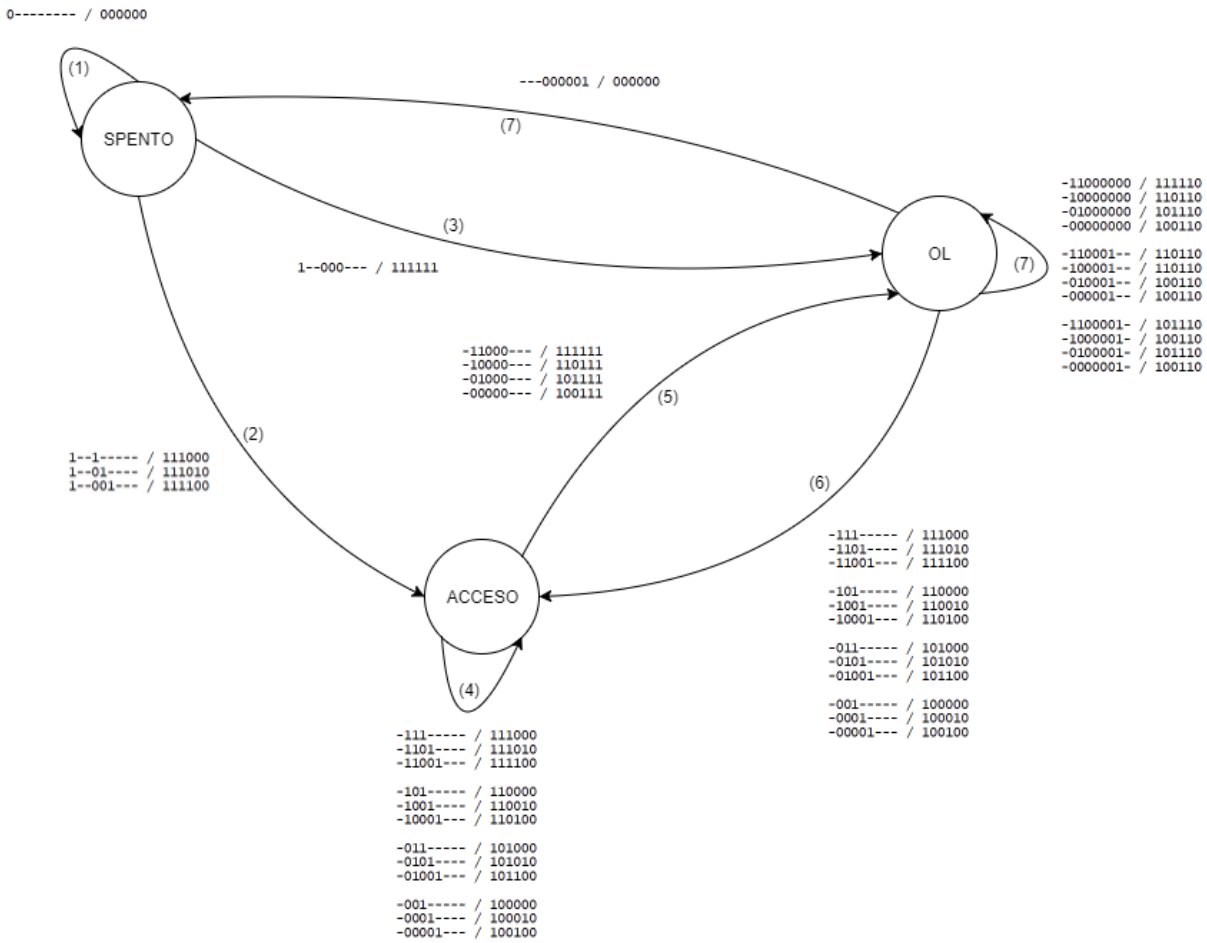
Funzionamento del contatore:

- Un multiplexer seleziona tra 0 e il valore corrente del contatore in base al bit di start.
- Dopodiché memorizza il valore del contatore in un registro, dal quale raggiunge i tre comparatori (che restituiscono ognuno un bit di end a seconda che il valore sia uguale a 2, 3 o 4) e allo stesso tempo un sommatore che incrementa di 1.

Scelte progettuali:

- Considerando che il consumo massimo è quello relativo al forno e alla lavastoviglie (2000 W) e che la somma massima è quella che si ottiene sommando tutti i consumi (9540 W), servirebbero 14 bit per rappresentare le costanti e i risultati dei sommatori. Ma se scaliamo da Watt (W) a decaWatt (daW), il consumo massimo diventa 200 daW e la somma massima diventa 954 daW, ottenendo un risparmio di 4 bit in quanto, dopo questa scelta, ne bastano 10.
- Il valore degli interruttori della lavastoviglie e della lavatrice passano prima da un registro (con valore di default pari a 1) per evitare che si presenti un errore dovuto alla presenza di un ciclo nella FSMD, in quanto al primo ciclo di clock i suddetti valori non sono ancora specificati.

DIAGRAMMA DEGLI STATI DEL CONTROLLORE



Funzionamento della macchina a stati:

(1). Fintanto che l'utente non mette RES_GEN a 1 il sistema rimane SPENTO

(2). Se l'utente mette RES_GEN a 1 e la codifica della fascia non corrisponde a 11, il sistema passa da SPENTO a ACCESO

(3). Se l'utente mette RES_GEN a 1 e la codifica della fascia corrisponde a 11, il sistema passa da SPENTO a OVERLOAD e inizia a contare

(4). Se il sistema è ACCESO e la codifica della fascia non corrisponde a 11, il sistema rimane ACCESO

(5). Se il sistema è ACCESO e la codifica della fascia corrisponde a 11, il sistema passa da ACCESO a OVERLOAD e inizia a contare

(6). Se il sistema è in OVERLOAD e, in qualsiasi momento, la codifica della fascia non corrisponde più a 11, il sistema ritorna ACCESO

(7). Se il sistema è in OVERLOAD e la codifica della fascia corrisponde ancora a 11, il sistema rimane in OVERLOAD e spegne mano a mano INT_DW, INT_WM e infine INT_GEN (ritornando SPENTO)

Scelte progettuali:

- Tre dei segnali di output del *codificatore*, ovvero i risultati dei comparatori, vengono dati in input alla FSM. Quindi, in base alla tabella sottostante, la FSM restituirà in output la codifica della fascia di consumo istantaneo.

	OUTPUT DEI COMPARATORI				
SOMMA (x)	<= 150	<= 300	<= 450	FASCIA	CODIFICA
x <= 150	1	1	1	F1	00
150 < x <= 300	0	1	1	F2	01
300 < x <= 450	0	0	1	F3	10
x > 450	0	0	0	OL	11

NOTA: nella FSM vengono utilizzati i don't care per coprire tutte le combinazioni possibili.

- Anche i segnali di output del *contatore* vengono dati in input alla FSM che, similmente a prima, deciderà se è il momento di cominciare a spegnere gli interruttori (vedere la tabella sottostante).

	OUTPUT DEI COMPARATORI			
VALORE DEL CONTATORE (x)	= 2	= 3	= 4	FSM
0 <= x < 2	0	0	0	invariato
x = 2	1	0	0	spegne INT_DW
x = 3	0	1	0	spegne INT_WM
x = 4	0	0	1	spegne INT_GEN

NOTA: nella FSM vengono utilizzati i don't care per coprire tutte le combinazioni possibili.

- Quando gli interruttori della lavastoviglie e della lavatrice vengono spenti e il sistema esce dallo stato di overload, lo stato dei suddetti interruttori viene preservato attraverso l'implementazione di tutte le combinazioni possibili nelle transizioni ACCESO-ACCESO, ACCESO-OL, OL-ACCESO e OL-OL. Lo stesso accade anche se l'utente decide di riarmare un interruttore spento. Il principio alla base è che la FSM legge lo stato degli interruttori al ciclo precedente (che riceve in input dal data-path) e restituisce in output lo stesso valore, il quale poi andrà a sua volta in input al data-path.
- Le codifiche degli stati della FSM sono state assegnate mediante l'algoritmo *jedi* in quanto, dopo la minimizzazione multi-livello utilizzando lo *script.rugged*, il circuito risulta maggiormente ottimizzato rispetto a quello che si otterrebbe minimizzando la FSM avente le codifiche degli stati assegnate dall'algoritmo *nova*.

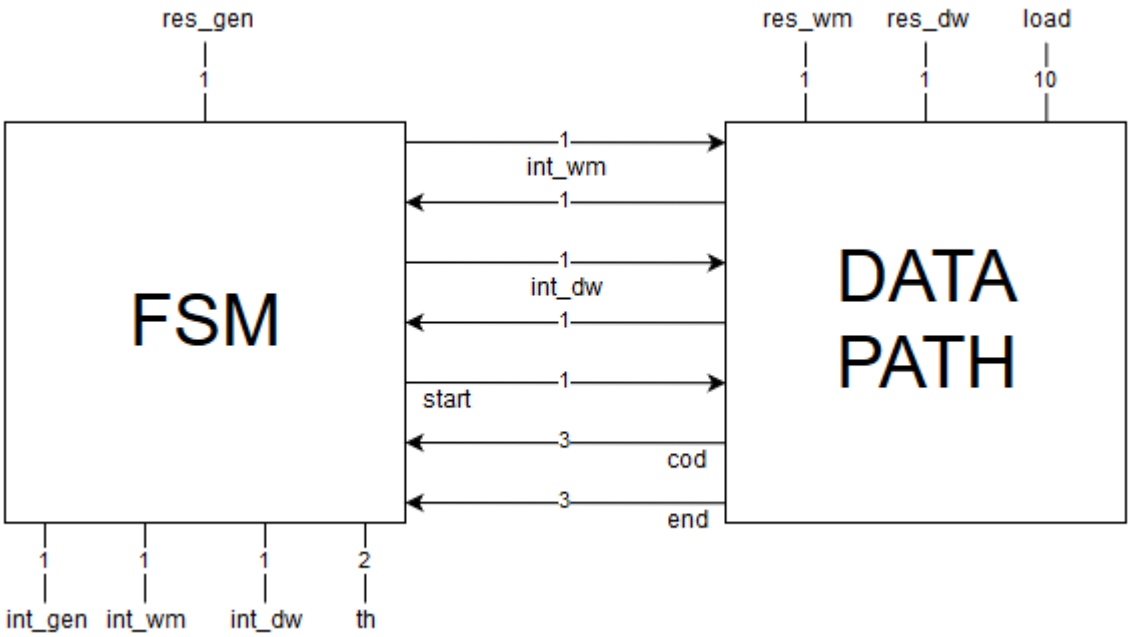
```
sis> rl fsm.blif
sis> psf
CONTROLLO       pi= 9   po= 6   nodes=  6       latches= 0
lits(sop)=   0  #states(STG)=   3
sis> state_assign nova
Running nova, written by Tiziano Villa,  UC Berkeley
Warning: network `SISGAAa13685', node "v0" does not fanout
Warning: network `SISGAAa13685', node "v1" does not fanout
Warning: network `SISGAAa13685', node "v2" does not fanout
Warning: network `SISGAAa13685', node "v3" does not fanout
Warning: network `SISGAAa13685', node "v4" does not fanout
Warning: network `SISGAAa13685', node "v5" does not fanout
Warning: network `SISGAAa13685', node "v6" does not fanout
Warning: network `SISGAAa13685', node "v7" does not fanout
Warning: network `SISGAAa13685', node "v8" does not fanout
sis> psf
CONTROLLO       pi= 9   po= 6   nodes=  8       latches= 2
lits(sop)= 255  #states(STG)=   3
sis> source script.rugged
sis> psf
CONTROLLO       pi= 9   po= 6   nodes= 10       latches= 2
lits(sop)=  50  #states(STG)=   3
```

assegnazione delle codifiche degli stati con l'algoritmo nova

```
sis> rl fsm.blif
sis> psf
CONTROLLO       pi= 9   po= 6   nodes=  6       latches= 0
lits(sop)=   0  #states(STG)=   3
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
sis> psf
CONTROLLO       pi= 9   po= 6   nodes=  8       latches= 2
lits(sop)= 181  #states(STG)=   3
sis> source script.rugged
sis> psf
CONTROLLO       pi= 9   po= 6   nodes= 11       latches= 2
lits(sop)=  46  #states(STG)=   3
sis> wl fsmMin.blif
sis>
```

assegnazione delle codifiche degli stati con l'algoritmo jedi

ARCHITETTURA GENERALE DEL CIRCUITO



Funzionamento del circuito:

Il circuito sequenziale riceve in ingresso 3 segnali di *reset* e 10 segnali di *load* (stato di accensione dei dispositivi). Per fornire in uscita il valore dei 3 *interruttori* e la *fascia* di consumo istantanea (composta da 2 bit), il circuito deve far scambiare alcuni segnali tra FSM e data-path; questi segnali sono:

- I valori degli *interruttori* della lavatrice e della lavastoviglie, i quali dalla FSM vengono mandati al data-path, dove verranno poi ricalcolati e rimandati alla FSM.
- Un segnale *start* che viene inviato dalla FSM al data-path per far sì che il contatore cominci a tenere traccia del numero di cicli di clock trascorsi da quando il sistema è entrato in overload.
- Un segnale *cod* composto da 3 bit che viene inviato dal data-path alla FSM, la quale lo userà per determinare la fascia di consumo istantanea da restituire in output.
- Un segnale *end* composto da 3 bit che viene inviato dal data-path alla FSM, la quale lo userà per decidere se è arrivato il momento di spegnere un determinato interruttore.

MAPPING TECNOLOGICO

Una volta eseguita l'ottimizzazione del circuito è possibile effettuare il mapping tecnologico, che ci permette di passare da porte logiche virtuali a porte logiche reali (appartenenti ad una certa libreria).

```
sis> psf
FSMD            pi=13   po= 5   nodes=603       latches= 7
lits(sop)=2171
sis> source script.rugged
sis> psf
FSMD            pi=13   po= 5   nodes=128       latches= 7
lits(sop)= 461
sis> wl FSMDtoMapp.blif
```

statistiche del circuito prima e dopo l'ottimizzazione

I passaggi da svolgere per effettuare il mapping tecnologico sono:

- Caricare la libreria tecnologica, in questo caso *synch.genlib*.
- Eseguire l'operazione di mapping, in questo caso usando il comando *map -m 0* per ottenere un circuito ottimizzato rispetto all'area.
- Visualizzare le informazioni relative ad area e ritardo del circuito dopo la mappatura lanciando il comando *map -s*. In particolare, *total gate area* indica il numero di gates, mentre *maximum arrival time* indica il ritardo.

```
sis> read_library synch.genlib
sis> map -m 0
warning: unknown latch type at node '{[58]}' (RISING_EDGE assumed)
warning: unknown latch type at node '{[59]}' (RISING_EDGE assumed)
WARNING: uses as primary input drive the value (0.20,0.20)
WARNING: uses as primary input arrival the value (0.00,0.00)
WARNING: uses as primary input max load limit the value (999.00)
WARNING: uses as primary output required the value (0.00,0.00)
WARNING: uses as primary output load the value 1.00
sis> map -s
>>> before removing serial inverters <<<
# of outputs:          12
total gate area:       7608.00
maximum arrival time: (79.20,79.20)
maximum po slack:     (-67.00,-67.00)
minimum po slack:     (-79.20,-79.20)
total neg slack:      (-896.40,-896.40)
# of failing outputs:  12
>>> before removing parallel inverters <<<
# of outputs:          12
total gate area:       7560.00
maximum arrival time: (78.40,78.40)
maximum po slack:     (-66.20,-66.20)
minimum po slack:     (-78.40,-78.40)
total neg slack:      (-886.80,-886.80)
# of failing outputs:  12
# of outputs:          12
total gate area:       6968.00
maximum arrival time: (77.20,77.20)
maximum po slack:     (-65.60,-65.60)
minimum po slack:     (-77.20,-77.20)
total neg slack:      (-875.20,-875.20)
# of failing outputs:  12
sis> psf
FSMD            pi=13   po= 5   nodes=233       latches= 7
lits(sop)= 552
sis>
```

numero di gates e ritardo dopo la mappatura