Laboratorio di Progettazione Digitale

Lezione 3:
Minimizzazione di circuiti combinatori Multilivello

In questa lezione vengono riassunti i concetti fondamentali della minimizzazione approssimata multi. In particolare viene mostrato come utilizzare SIS per effettuare tali operazioni.

Progetto di circuiti modulari in SIS

I modelli che descrivono sistemi complessi vengono spesso descritti con un approccio *gerarchico*. Ossia, solitamente vengono descritti organizzando il sistema in sotto-componenti, ossia *istanze* di modelli che specificano le componenti del sistema. Per includere un componente all'interno di un modello blif usare le seguenti istruzioni:

```
.subckt nomecomponente parametroformale=parametroattuale …

.search nomefilecomponente.blif
```

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

Minimizzazione approssimata multi-livello

La minimizzazione multi-livello consente al progettista di bilanciare area e ritardo di un circuito con un maggior grado di libertà rispetto alla minimizzazione a 2 livelli. Tuttavia, non esistono tecniche esatte efficienti che portino alla realizzazione di configurazioni ottime usando la minimizzazione multi-livello; pertanto si ricorre a tecniche euristiche che garantiscono buone soluzioni in tempi di calcolo ragionevoli.

Riassumiamo di seguito i concetti principali relativi alle tecniche di sintesi applicate durante la minimizzazione multi-livello. Si consideri che il circuito viene rappresentato come un insieme di nodi interconnessi tra loro, e che **ad ogni *nodo* corrisponde una funzione booleana a una sola uscita** (si veda la sezione `nodes` dell'output di `print_stats`). Ogni nodo, pertanto, rappresenta una piccola porzione dell'intero sistema. Le tecniche descritte nei seguenti punti vengono solitamente ripetute secondo un determinato ordine fino al raggiungimento di una configurazione che soddisfi le aspettative del progettista.

- ***Sweep:*** eliminazione dei nodi con un'unica linea di ingresso e di nodi con valore costante.

- ***Eliminazione:*** eliminazione di un nodo interno alla rete. Si consideri che il nodo N rappresenti la funzione y = (a + b)*c. Gli ingressi di N sono pertanto a, b, c mentre l'uscita è y. Si consideri, inoltre, che la variabile di uscita y venga utilizzata come input in alcuni nodi successivi. L'eliminazione di N prevede la sostituzione

della variabile y in tutti i nodi che la utilizzano con l'espressione booleana (a + b) * c.

- ***Scomposizione:*** sostituzione di un nodo interno con un'insieme di nodi la cui funzionalità sia equivalente a quella del nodo sostituito. L'operazione viene effettuata per diminuire la complessità di un nodo.

- ***Estrazione:*** estrazione di una sottoespressione comune a più nodi che viene rappresentata con un nuovo nodo.

- ***Semplificazione:*** riduzione della complessità di ogni singolo nodo con algoritmo di Quine-McCluskey.

Minimizzazione di circuiti combinatori multi-livello tramite SIS

Una volta descritto il circuito desiderato nel formato BLIF è possibile utilizzare i seguenti comandi per effettuare la minimizzazione. Per ulteriori approfondimenti sull'uso dei comandi di ottimizzazione e sulle opzioni disponibili, si veda il relativo help fornito da SIS tramite il comando `help comando`.

- sweep	Esegue l'operazione di sweep;
- eliminate *n*	Esegue l'operazione di eliminazione rimuovendo i nodi tali che la loro rimozione non aumenti il numero di letterali di una quantità superiore a "n" (numero intero). Per eliminare i nodi che sono utilizzati una sola volta utilizzare il valore -1;
- resub *lista*	Esegue l'operazione di scomposizione dei nodi indicati nella lista. Se la lista non viene specificata, la sostituzione viene eseguita per tutti i nodi della rete. I nodi nella lista devono essere specificati con il nome della loro uscita e vanno intervallati tra loro da uno spazio;
- fx	Esegue l'operazione di estrazione;
- full_simplify	Esegue l'operazione di semplificazione su ogni nodo della rete;
- source *script*	Carica lo script ed esegue tutti i comandi contenuti al suo interno. Lo script che fornisce generalmente i risultati migliori è `script.rugged;`
- set autoexec *comando*	Stampa automaticamente il risultato del comando specificato dopo l'esecuzione di un qualunque altro comando.

Mapping tecnologico

La realizzazione pratica di un circuito digitale può avvenire in due modi:

1) costruzione di un chip mediante stampa del circuito su un piccolo quadrato di semiconduttore (ad es. Silicio);

2) utilizzo di circuiti integrati riprogrammabili (es. FPGA), ovvero un sistema composto da un numero fisso di porte logiche elementari che possono essere interconnesse a piacimento per realizzare la funzione desiderata. L'interconnessione tra i vari dispositivi è decisa semplicemente scrivendo in una memoria e può avvenire un elevato numero di volte.

Nel primo caso si possono utilizzare solo le porte logiche di base che è possibile fondere sul semiconduttore mentre nel secondo caso si possono utilizzare solo quelle che sono contenute nell'FPGA; in entrambi i casi quindi occorre ricondurre il circuito da realizzare a tale insieme di elementi base. Per fare questo si effettua il ***mapping tecnologico*** su una determinata **libreria tecnologica** fornita dal costruttore di semiconduttori o dall'FPGA.

Le porte logiche di base contenute nella libreria hanno determinate caratteristiche di area e ritardo perché si riferiscono a *dispositivi fisici concreti*; quindi dopo il mapping tecnologico è possibile conoscere l'area e il ritardo del proprio circuito non più in termini di numero di porte logiche ma in modo esatto.

Mapping tecnologico di circuiti combinatori tramite SIS

Una volta descritto il circuito desiderato nel formato blif, ed una volta eseguite le minimizzazione del caso è possibile utilizzare i seguenti comandi per effettuare il mapping tecnologico.

- read_library *libreria*	Carica la libreria tecnologica di nome libreria. Le librerie sono specificate nel formato genlib (estensione .genlib, da esempio synch.genlib e mcnc.genlib);
- print_library	Visualizza informazioni inerenti la libreria caricata;
- map	Esegue l'operazione di mapping. Vedere l'help di SIS per ulteriori informazioni relative alle opzioni di mapping tramite il comando "help map". L'opzione –m 0 permette di ottenere un circuito minimizzato rispetto all'area. L'opzione –n 1 permette di ottenere un circuito minimizzato rispetto al ritardo. L'opzione –s permette di visualizzare alcune informazioni relative ad area e ritardo dopo il mapping;
- write_blif -n	Mostra la rappresentazione del circuito associata alle porte della libreria;
- print_delay	Stampa informazioni relative al ritardo del circuito.
- reduce_depth	Riduce la lunghezza dei cammini critici;

Il comando `map -s` permette di visualizzare le informazioni relative ad area e ritardo del circuito dopo il mapping tecnologico; in particolare `total gate area` fornisce il valore dell'area come numero di celle standard della libreria tecnologica mentre `maximum arrival time` indica il ritardo.

Esercizi

Esercizio 1: Eseguire la minimizzazione multi-livello usando lo script script.rugged su tutti i circuiti realizzati durante la seconda esercitazione ("Ottimizzazione esatta a 2 livelli"). Quale dei 4 dispositivi viene maggiormente ottimizzato?

Esercizio 2: Effettuare il mapping tecnologico usando la libreria `mcnc.genlib` su tutti i circuiti realizzati durante la seconda esercitazione ("Ottimizzazione esatta a 2 livelli") dopo aver eseguito la minimizzazione multilivello richiesta per l'esercizio precedente. Quale dei 4 dispositivi ha area maggiore? Quale ha ritardo maggiore? Effettuare questa analisi prima e dopo aver eseguito il comando `reduce_depth` per diminuire il ritardo dei cammini critici.

Esercizio 3: Descrivere nel formato blif il circuito rappresentato dalla funzione booleana (x, v, w, z) = f(a, b, c, d, e) descritta dai seguenti nodi:

$$f = \overline{a}be + cd + a\overline{c}d$$
$$g = de + abc + a\overline{c}d$$
$$h = ae + cd + ab$$
$$i = g + \overline{h}$$
$$l = ag + bc\overline{g} + ae$$
$$m = f + i + bc$$
$$n = l + a\overline{i}$$
$$o = m + a + \overline{b}e$$
$$x = f$$
$$v = o$$
$$w = n$$
$$z = l$$

Eseguire la minimizzazione multi-livello in due modi:

1. usando lo script `script.rugged`.
2. usando una sequenza greedy user-defined in modo da migliorare ulteriormente il risultato ottenuto con lo script.rugged, distinguendo l'ottimizzazione per area o ritardo.

Esercizio 4: Effettuare il mapping tecnologico della funzione dell'esercizio precedente usando la libreria `mcnc.genlib`. Qual è il cammino con maggior ritardo prima e dopo aver eseguito il comando `reduce_depth` per diminuire il ritardo dei cammini critici?