

ritardi). Il problema può sembrare computazionalmente molto pesante; la pratica sperimentale ha però dimostrato che – per problemi reali – può essere in realtà risolto in tempi accettabili.

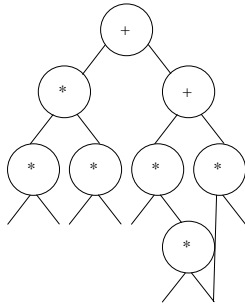


Figura 8.6 Grafo corrispondente all'associazione effettuata.

8.2.1 Esercizi con strumenti automatici

Nel Capitolo 4 sono state presentate le tecniche per la minimizzazione di funzioni combinatorie a due o più livelli. Questo tipo di algoritmi è completamente indipendente dalla libreria tecnologica di porte logiche con la quale sarà realizzato il circuito reale e necessitano quindi di una fase finale di associazione a una libreria di celle. SIS esegue questa operazione con il comando `map` che deve essere eseguito dopo aver caricato in memoria una libreria di porte. Il comando `read_library` permette di effettuare questa operazione leggendo le caratteristiche delle porte da un file in formato *.genlib*. Questo tipo di file riporta le caratteristiche di ogni porta in termini di equazioni, area e ritardi. Una volta caricato un file di libreria è possibile esaminare le caratteristiche di ogni porta eseguendo il comando `print_library`. Il lettore interessato può far riferimento alla documentazione generale di SIS per trovare la sintassi di un file *.genlib*.

Si provi ora a eseguire l'operazione di mapping tecnologico sul circuito descritto alla fine del Capitolo 4 e già ottimizzato. Questa rete è descritta nel file `rete5in4outmin.blif` che si vuole associare alle porte della libreria `mcnc.genlib`².

² I file delle librerie sono presenti nella directory `sis_lib` in cui sono presenti anche gli script di minimizzazione e gli `help`.

```
sis> read_library mcnc.genlib
sis> read_blif rete5in4outmin.blif
sis> print_stats
rete5in4outmin pi= 5 po= 4 nodes= 6 latches= 0
lits(sop)= 21
sis> map -W -m 0 -s
```

l'uso di questi parametri permette di effettuare un mapping il cui obiettivo è la minimizzazione dell'area del circuito. Si ottiene:

```
# of outputs:      4
total gate area:   28.00
maximum arrival time: (7.00,7.00)
maximum po slack:  (-5.80,-5.80)
minimum po slack:  (-7.00,-7.00)
total neg slack:   (-25.50,-25.50)
```

Si noti che stampando le statistiche:

```
sis> print_stats
rete5in4outmin pi= 5 po= 4 nodes= 14 latches= 0
lits(sop)= 32
```

le dimensioni in letterali del circuito sono aumentate poiché la fase di mapping tecnologico ha modificato la scomposizione del circuito trovata alla fine della minimizzazione. Questo fenomeno si verifica sempre, a fronte di una fase di mapping, poiché le funzionalità presenti in una libreria tecnologica impongono dei vincoli alle possibili minimizzazioni.

Per ricavare una rappresentazione del circuito associato alle porte di libreria basta eseguire il comando:

```
sis> write_blif -n
.model rete5in4outmin
.inputs a b c d e
.outputs w x z y
.gate inv1 a=b O=[221]
.gate nand3 a=c b=[221] c=a O=[255]
.gate aoi22 a=[255] b=d c=a d=e O=[157]
.gate inv1 a=[157] O=w
.gate inv1 a=c O=[219]
.gate inv1 a=d O=[218]
.gate nand2 a=[219] b=[218] O=[249]
.gate nand2 a=[249] b=e O=[259]
.gate nand3 a=a b=[259] c=b O=x
.gate inv1 a=a O=[220]
.gate nand2 a=[220] b=[221] O=[225]
.gate or2 a=c b=[225] O=z
.gate aoi21 a=[249] b=[225] c=e O=[160]
```

```
.gate inv1 a=[160] 0=y
.end
```

L'algoritmo di associazione può anche lavorare cercando di minimizzare il ritardo massimo a scapito di una minore minimizzazione del circuito. Cambiando opzioni del comando map si ottiene:

```
sis> read_library mcnc.genlib
sis> read_blif rete5in4outmin.blif
sis> map -W -n 1 -s
# of outputs: 4
total gate area: 36.00
maximum arrival time: (6.60,6.60)
maximum po slack: (-5.50,-5.50)
minimum po slack: (-6.60,-6.60)
total neg slack: (-23.50,-23.50)
```

Si vede che a fronte di un ritardo massimo inferiore (6.6), l'area del circuito è passata da 28 a 36. Ancora una volta è necessario identificare un compromesso tra i due parametri di ottimizzazione: area e ritardo.

8.3 Il problema dell'associazione per componenti programmabili

Negli ultimi anni, la crescente densità di integrazione, il livello di complessità e le prestazioni raggiunte dai dispositivi programmabili o *PLD (Programmable Logic Device)* hanno contribuito da un lato ad allargare la loro diffusione sul mercato dei dispositivi a semiconduttore e dall'altro a cambiare radicalmente il flusso di progettazione dell'hardware. Le soluzioni programmabili più avanzate, come *FPGA (Field Programmable Gate Array)* e *CPLD (Complex PLD)*, realizzate con avanzate tecnologie *CMOS* sub-micrometriche possono costituire in molti casi una valida alternativa ai dispositivi "semicustom"³, e non rappresentare più soltanto una soluzione da adottare in fase di definizione dei prototipi.

Sebbene *FPGA* e *CPLD* siano basati su soluzioni architetturali diverse, caratteristica comune è quella di essere costituiti da un array di circuiti logici elementari e da risorse di interconnessione entrambi configurabili *on-field* dal progettista, che permettono di ottenere una completa programmabilità del dispositivo.

I *PLD* si contrappongono ai dispositivi *semicustom* le cui caratteristiche sono definite durante la progettazione e non sono più modificabili dopo la fase di fabbricazione. In particolare, i dispositivi *semicustom* di tipo *gate array* sono anche

³ Dispositivi costituiti da un insieme di sottocircuiti predefiniti – per esempio, porte logiche con un numero fisso di ingressi – fra i quali l'utente finale realizza una rete di interconnessione che permette di realizzare la funzione voluta. Casi tipici sono i *gate array*, matrici di porte logiche (per esempio *NAND* a due ingressi).

con la rete di Figura 8.9, nella quale la prima cella implementa $\gamma = (fg)(ec)$, la seconda implementa $\beta = cde$, la terza $\alpha = ab$ e la quarta $F = (\alpha + \beta) + \gamma$ (chiaramente, questa non è l'unica copertura possibile). Un secondo approccio mira invece a minimizzare il ritardo di propagazione lungo il percorso critico; dato che il ritardo di propagazione è in questo caso identico per tutte le celle, il problema si traduce nella minimizzazione del numero di look-up tables lungo il percorso critico.

8.3.1 Esercizi con strumenti automatici

Il programma SIS mette a disposizione un certo numero di comandi per vincolare un circuito minimizzato in maniera indipendente dalla tecnologia a essere implementato da un *FPGA*. La famiglia di *FPGA* considerata è prodotta da Xilinx. I comandi a disposizione verificano che ogni nodo della rete corrente sia realizzabile da una cella Xilinx, in caso positivo lo realizzano, altrimenti fattorizzano ulteriormente la rete.

Il comando per effettuare questa associazione è `x1_cover` che accetta come parametro il numero massimo di ingressi di ogni cella di *FPGA*. Questo numero massimo è caratteristico di una particolare famiglia di *FPGA*. Si consideri, per esempio, nuovamente per la rete usata nel Paragrafo 8.2.1 e si assuma che il numero massimo di ingressi di ogni cella possa essere 3 (in realtà questo è un numero molto piccolo). Si ottiene:

```
sis> read_blif rete5in4outmin.blif
sis> print_stats
rete5in4outmin pi= 5 po= 4 nodes= 6 latches= 0
lits(sop)= 21
sis> write_eqn
INORDER = a b c d e;
OUTORDER = w x z y;
w = a*e + d!*c + d*b + !a*d;
x = e*[4] + !b + !a;
z = [5] + c;
y = [4]*[5] + e;
[4] = d + c;
[5] = b + a;
sis> x1_cover -n 3
Error: The network has a node with > 3 fanins
Run x1_imp (x1_ao) -n 3 to decompose these nodes
```

il comando di copertura segnala un errore poiché i nodi *w* e *x* hanno più di tre ingressi. Eseguendo una fattorizzazione *ad hoc* con il comando `x1_imp` si ottiene:

```
sis> x1_imp -n 3
sis> print_stats
rete5in4outmin pi= 5 po= 4 nodes= 9 latches= 0
lits(sop)= 24
sis> write_eqn
INORDER = a b c d e;
```

```
OUTORDER = w x z y;  
w = a*e + [68];  
x = [71] + !b;  
z = [5] + c;  
y = [4]*[5] + e;  
[4] = d + c;  
[5] = b + a;  
[67] = b*d + !a*d;  
[68] = !c*d + [67];  
[71] = e*[4] + !a;
```

in cui tutti i nodi hanno ora al massimo tre ingressi (si noti che il numero di letterali è aumentato). Il comando di copertura eseguito su questa nuova rete:

```
sis> xl_cover -n 3  
sis> print_stats  
rete5in4outmin pl= 5 po= 4 nodes= 9 latches= 0  
lits(sop)= 24
```

non produce ulteriori modifiche alla rete. Questa rete è ora pronta per essere realizzata con un *FPGA* Xilinx: è effettivamente possibile farlo interfacciandosi con i programmi *CAD* della particolare famiglia di *FPGA* che possono leggere un file salvato con il comando `write_pds`.