Lez. 3 – Ottimizzazione di circuiti combinatori multilivello

Laboratorio di Architettura degli Elaboratori

Stefano Centomo

19-29 Novembre 2021

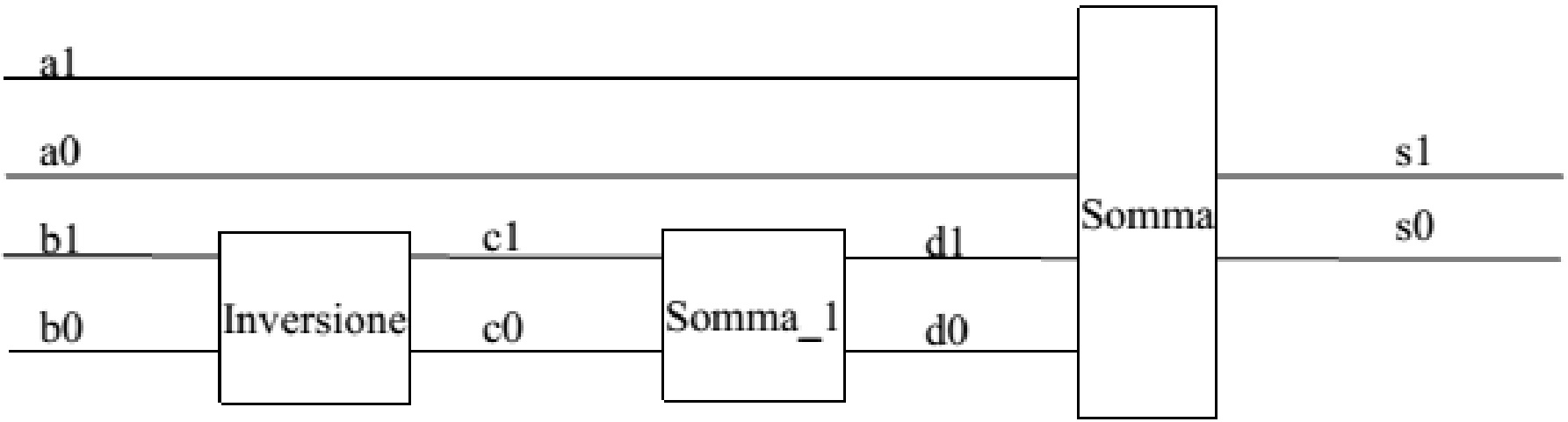
Esercizio 2 – Sottrattore binario

Descrivere in formato `blif` il circuito digitale che esegue la sottrazione di 2 numeri binari su 2 bit rappresentati in complemento a 2 con risultato ancora su 2 bit in complemento a 2. Il circuito corrispondente avrà quindi 4 ingressi e 2 uscite. Si parta scrivendo la tabella di verità per poi scrivere il file `.blif`.
Eseguire l'ottimizzazione con SIS. Visualizzare l'espressione booleana corrispondente al circuito prima e dopo l'ottimizzazione e fornire il grado di ottimizzazione confrontando il numero di letterali del circuito prima e dopo l'esecuzione del comando `full_simplify`.

Esercizio 2

a_1	a_0	b_1	b_0	s_1	s_0
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	-	-
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	-	-
0	1	1	1	-	-
1	0	0	0	1	0
1	0	0	1	-	-
1	0	1	0	0	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	1	0	0

Esercizio 2



Esercizio 2

Inversione

b_1	b_0	c_1	c_0
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

Esercizio 2

Somma_1

c_1	c_0	d_1	d_0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Esercizio 2

Inversione + Somma_1

b_1	b_0	c_1	c_0	d_1	d_0
0	0	1	1	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	1	0	0	0	1

$b_{\{10\}}$		$d_{\{10\}}$
0	$\rightarrow$	0
1	$\rightarrow$	-1
-2	$\rightarrow$	-2
-1	$\rightarrow$	1

Esercizio 2

Somma

a_1	a_0	d_1	d_0	s_1	s_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	0	0
1	1	1	0	0	1
1	1	1	1	1	0

Esercizio 2

				Diff		Sum	
a_1	a_0	b_1	b_0	s_1	s_0	s_1	s_0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	-	-	1	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	0	1
0	1	0	1	0	0	0	0
0	1	1	0	-	-	1	1
0	1	1	1	-	-	1	0
1	0	0	0	1	0	1	0
1	0	0	1	-	-	0	1
1	0	1	0	0	0	0	0
1	0	1	1	1	1	1	1
1	1	0	0	1	1	1	1
1	1	0	1	1	0	1	0
1	1	1	0	0	1	0	1
1	1	1	1	0	0	0	0

Ottimizzazione approssimata multilivello

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.

Ottimizzazione approssimata multilivello

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**.

Ottimizzazione approssimata multilivello

sweep eliminazione dei nodi con un'unica linea di ingresso e di nodi con valore costante

eliminate eliminazione di un nodo interno alla rete. Si consideri che il nodo N rappresenti la funzione $y = (a + b) * c$, l'eliminazione di N prevede la sostituzione della variabile y in tutti i nodi che la utilizzano con l'espressione booleana $(a + b) * c$

resub 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

Ottimizzazione approssimata multilivello

extract estrazione di una sottoespressione comune a più nodi che viene rappresentata con un nuovo nodo

simplify riduzione della complessità di ogni singolo nodo con algoritmo di Quine-McCluskey

Sweep (sweep)

Elimina nodi con un'unica linea di ingresso (`k`) e nodi con valore costante (`x`, dopo aver eliminato `k`).

- **Nodi iniziali**:

```
k = 1;
x = f*k;
f = !a*b*e + a*!c*d + c*d;
m = b*c + i + x;
```

- **Post sweep**:

```
f = !a*b*e + a*!c*d + c*d
m = b*c + i + f;
```

Sweep (Cont.)

ESECUZIONE PASSO PASSO:

1. **Nodi Iniziali:**

```
k = 1;
x = f*k;
f = !a*b*e + a*!c*d + c*d;
m = b*c + i + x;
```

2. **Eliminazione nodo k perchè costante:**

```
x = f;
f = !a*b*e + a*!c*d + c*d;
m = b*c + i + x;
```

3. **Eliminazione del nodo x perchè con una sola linea di ingresso:**

```
f = !a*b*e + a*!c*d + c*d;
m = b*c + i + x;
```

4. **Sostituzione di x con f, visto che x non esiste più:**

```
f = !a*b*e + a*!c*d + c*d;
m = b*c + i + f;
```

Estrazione (fx)

Estrazione di sottoespressioni comuni a più nodi (`b*c`) e creazione di un nuovo nodo (`[8]`).

- **Nodi iniziali**:

```
z = b*c*!g + a*g + a*e;
m = b*c + i + x;
```

- **Post fx**:

```
z = !g*[8] + a*g + a*e;
m = [8] + i + x;
[8] = b*c;
```

Eliminazione (eliminate)

Elimina un nodo (`h`) sostituendo la sua espressione in tutti gli altri nodi (`i`)

- **Nodi iniziali**:

```
g = a*!c*d + a*b*c + d*e;
h = a*e + c*d + a*b;
i = !h + g;
```

- **Post eliminate**:

```
g = a*!c*d + a*b*c + d*e;
i = !b*!d*!e + !b*!c*!e + !a*!d + !a*!c + g;
```

Eliminazione (eliminate) (Cont.)

ESECUZIONE PASSO PASSO:

1. **Nodi Iniziali:**

```
g = a*!c*d + a*b*c + d*e;
h = a*e + c*d + a*b;
i = !h + g;
```

2. **Sostituzione in i di h:**

```
i = !(a*e + c*d + a*b) + g;
```

3. **Espansione not:**

```
 i = (!a+!e) * (!c+!d) * (!a+!b) + g;
```

4. **Rifattorizzazione::**

```
i = (!a*!c + !a*!d + !c*!e + !d*!e) * (!a+!b) + g;
```

5. **Ottengo:**

```
i = !a*!c + !a*!d + !a*!c*!e + !a*!d*!e + !a*!b*!c +
!a*!b*!d + !b*!c*!e + !b*!d*!e + g;
```

6. **Tengo solo i minimi:**

```
i = !a*!c + !a*!d + !b*!c*!e +!b*!d*!e + g;
```

Esercizio 1

Eseguire la minimizzazione multilivello usando lo script `script.rugged` su tutti i circuiti realizzati durante la scorsa esercitazione.
Quale dei 4 dispositivi viene maggiormente ottimizzato?

Esercizio 2

Descrivere nel formato `.blif` il circuito rappresentato dalla funzione booleana $(x, v, w, z) = \mathbf{f}(a, b, c, d, e)$ descritta dai seguenti nodi:

$$f = \overline{a}\, b\, e + c\, d + a\, \overline{c}\, d \qquad l = a\, g + b\, c\, \overline{g} + a\, e \qquad x = f$$
$$g = d\, e + a\, b\, c + a\, \overline{c}\, d \qquad m = f + i + b\, c \qquad v = o$$
$$h = a\, e + c\, d + a\, b \qquad n = l + a\, \overline{i} \qquad w = n$$
$$i = g + \overline{h} \qquad o = m + a + \overline{b}\, e \qquad z = l$$

Eseguire la minimizzazione multi-livello in due modi:

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

script.rugged

```
sweep; eliminate -1
simplify -m nocomp
eliminate -1

sweep; eliminate 5
simplify -m nocomp
resub -a

fx
resub -a; sweep

eliminate -1; sweep
full_simplify -m nocomp
```