

# Sintesi ad alto livello per sistemi pipelined

---

# Tipi di pipelining

- Sintesi con risorse di tipo pipelined
- Sintesi con risorse non pipelined => **functional pipelining**

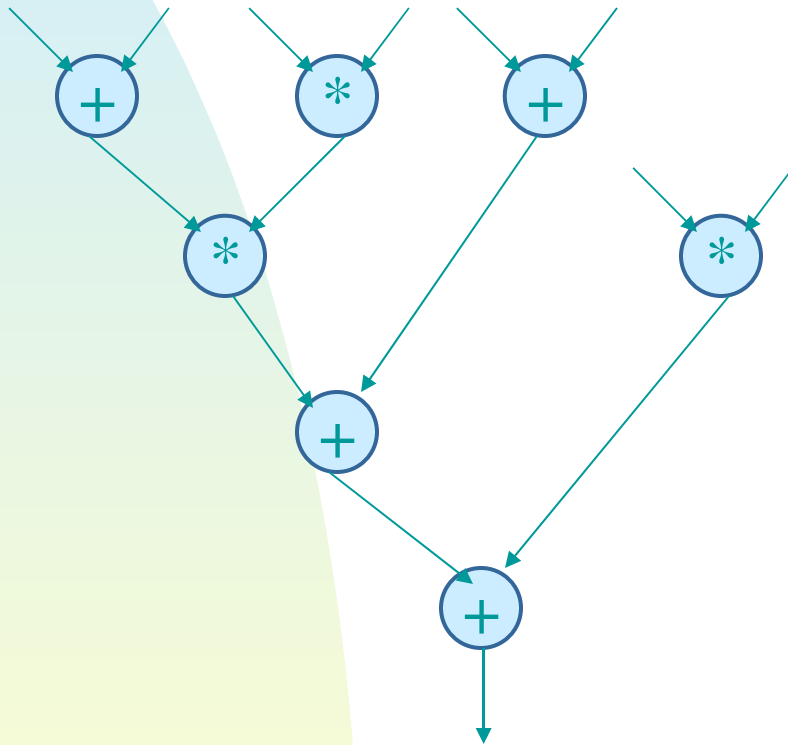
# Specifiche

- Specifiche: rateo di introduzione dei dati
- $\rho=1/\delta$
- Nel caso non pipelined deve essere  $\delta > \text{latenza}$
- Il rateo di introduzione dei dati determina la quantità di risorse da utilizzare => maggiore è  $\rho$ , maggiore è il numero di risorse da utilizzare
- **Di quante risorse si ha bisogno per un dato valore di  $\rho$  e un dato DFG?**

# Numero di risorse

- Ipotesi di ritardi unitari degli operatori (né multicycling, né chaining)
- $n(k)$  = numero di operazioni di tipo  $k$
- $a(k)$  = utilizzo di risorse per tali operazioni
- limite inferiore per il numero di risorse da usare
$$a(k) = \text{round}(n(k)/\delta)$$
- cosa succede se  $\delta=1$  ?

# Esempio



- $n(+)=4$   $n(*)=3$
- Si supponga  $\delta=1$ 
  - ◆  $a(+)=4$ ,  $a(*)=3$
- Si supponga  $\delta=2$ 
  - ◆  $n(+)=2$ ,  $n(*)=2$
- Non è detto che con tali risorse si riesca a soddisfare i vincoli di progetto

# Sintesi ed analisi

- Gli algoritmi di sintesi possono essere modificati per soddisfare i vincoli sul rate di ingresso
- Ci limiteremo a vedere un algoritmo di analisi per verificare se lo scheduling e l'allocazione di un DFG soddisfano le specifiche

# Analisi

- Latenza  $d$
- Per una risorsa  $j$ , sia  $o(j,i)$  una funzione che denota l'impegno della risorsa  $j$  nell'intervallo  $i=1\dots d$
- $o(j,i)=$   $\begin{cases} 1 & \text{se la risorsa è impegnata} \\ 0 & \text{altrimenti} \end{cases}$

# Analisi

- La rete può funzionare con un rate  $\delta$  se
- Per ogni risorsa  $j$  e per ogni  $i=1 \dots d$ , la seguente relazione è soddisfatta

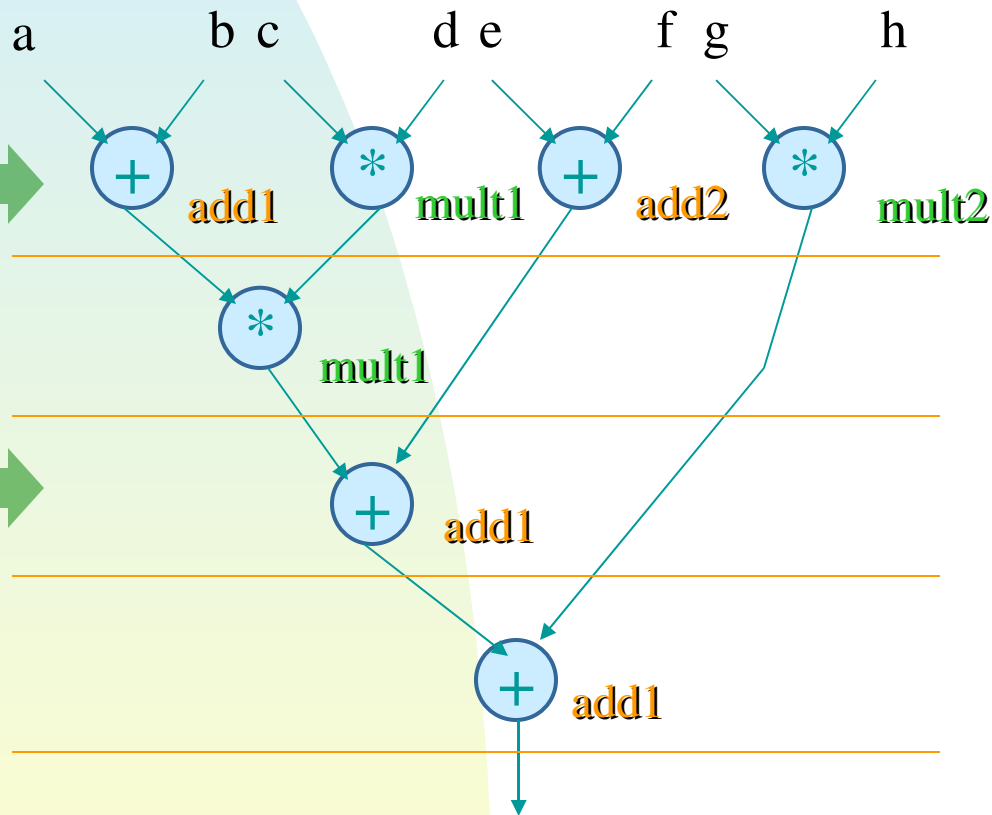
$$o(j,i) + o(j,i + \delta) < 2$$

- In pratica non si devono avere conflitti



# Esempio I

- Scheduling ASAP (con il numero minimo di risorse) con  $\delta=2$



**conflitto**

$$o(add1,i) \quad 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ \dots$$

$$o(add1,i+2) \quad \dots \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ \dots$$

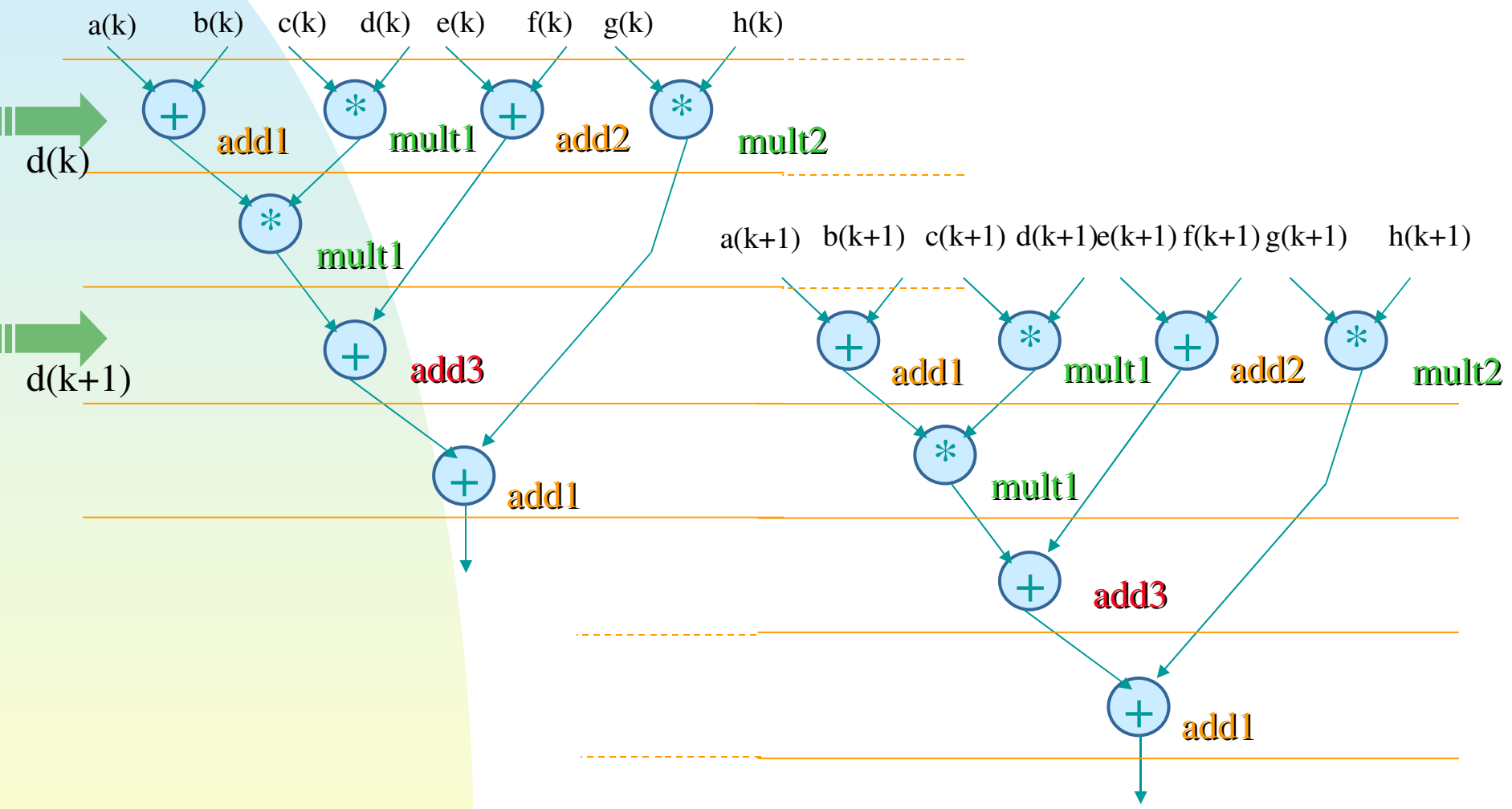
$$o(mult1,i) \quad 1 \ 1 \ 0 \ 0 \ 1 \ \dots$$

$$o(mult1,i+2) \quad \dots \ 1 \ 1 \ 0 \ 0 \ 1 \ \dots$$

**OK**

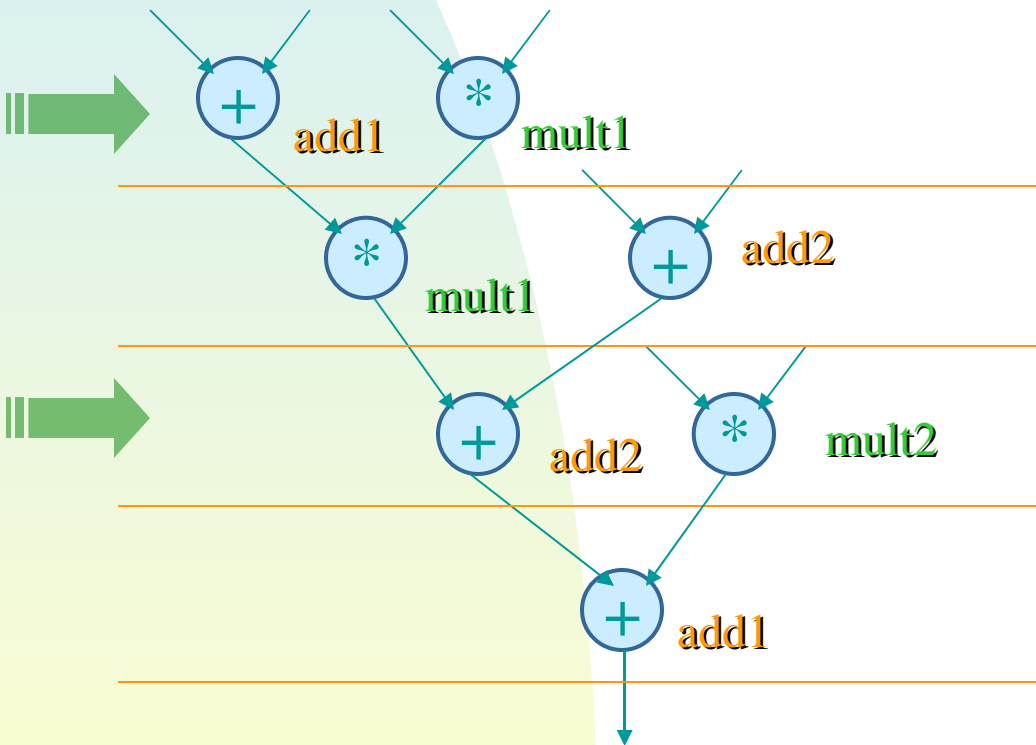
dobbiamo aggiungere un nuovo addizionale

# Esempio I



# Esempio II

- Scheduling ALAP (con il numero minimo di risorse) con  $\delta=2$



$o(add1,i)$  1 0 0 1 1 0 ...

$o(add1,i+2)$  ... 1 0 0 1 1 0 ...

**OK**

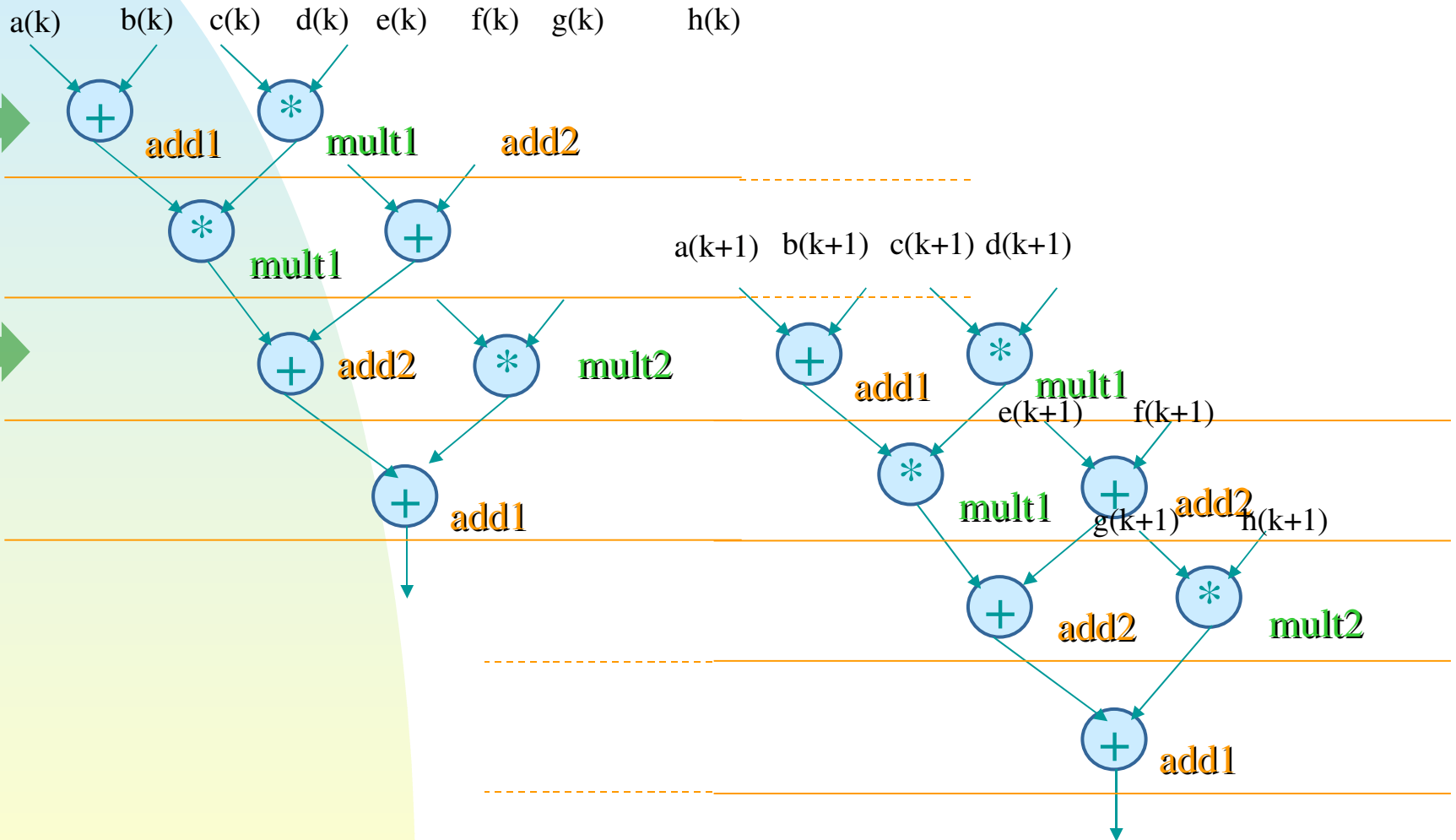
$o(mult1,i)$  1 1 0 0 1 ...

$o(mult1,i+2)$  ... 1 1 0 0 1 ...

$o(add2,i)$  0 1 1 0 0 1 ...

$o(add2,i+2)$  ... 0 1 1 0 1 1 ...

# Esempio II



# Conclusioni

- Il pipelining è una tecnica fondamentale nel caso in cui il rate dei dati in ingresso sia superiore alla latenza minima
- Il pipelining richiede un numero maggiore numero di risorse di un implementazione convenzionale
- I ragionamenti visti per le risorse funzionali devono essere ripetuti per registri e steering logic