# Esempi di reti dataflow in Gezel e in VHDL

## Esercitazione 05 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2016-17



Indice

1. Esempi di reti dataflow in Gezel e in VHDL
2. argomenti dell'esercitazione
3. grafi SDF single-rate in hardware
4. esempio: algoritmo GCD di Euclide, analisi del grafo SDF
5. realizzazione hardware dell'algoritmo GCD di Euclide
6. hardware pipelining
7. pipelining in grafi SDF con cicli
8. esperienza di laboratorio
9. riferimenti

## argomenti dell'esercitazione

in questa esercitazione si trattano:

- realizzazione hardware di modelli SDF single-rate
- esempio, algoritmo GCD di Euclide:
  - analisi del grafo SDF
  - realizzazione hardware in Gezel
- hardware pipelining:
  - miglioramento del throughput
  - cautela con il pipelining di grafi SDF con cicli
- esperienza di laboratorio
  - realizzazioni hardware di modelli dataflow in Gezel e VHDL






## grafi SDF single-rate in hardware

ipotesi per la realizzazione hardware:

grafi SDF single-rate, tutti gli attori operano alla stessa frequenza di clock

tre regole per la realizzazione:

1. gli attori sono realizzati da circuiti combinatori
2. le code di comunicazione sono realizzate da connessioni (prive di memoria)
3. ogni token iniziale sulle code di comunicazione è rimpiazzato da un registro

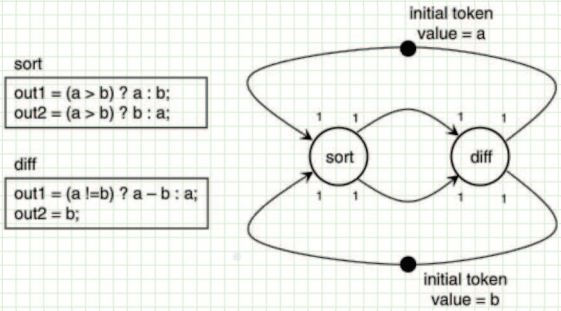
due definizioni:

- *cammino combinatorio* nel grafo SDF: un cammino aciclico privo di token iniziali
- *cammino critico* nel grafo SDF: cammino combinatorio di massima latenza

frequenza di clock massima per il circuito: reciproco della latenza del cammino critico

algoritmo: a ogni passo rimpiazza (a, b) con (|a-b|, min(a,b))

la coppia converge a (GCD(a,b), GCD(a,b))




Schaumont, Figure 3.10 - Euclid's greatest common divisor as an SDF graph

analisi PASS:

$$G = \begin{bmatrix} +1 & -1 \\ +1 & -1 \\ -1 & +1 \\ -1 & +1 \end{bmatrix} \begin{matrix} \leftarrow edge(sort,diff) \\ \leftarrow edge(sort,diff) \\ \leftarrow edge(diff,sort) \\ \leftarrow edge(diff,sort) \end{matrix} \qquad \text{rango}(G) = 1 \qquad q_{PASS} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



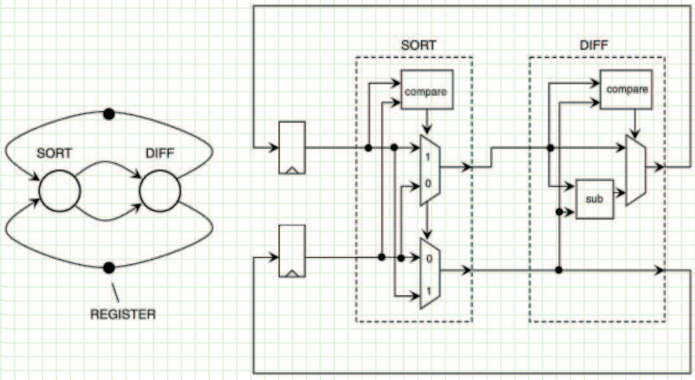





applichiamo le tre regole indicate per la realizzazione hardware del modello SDF:

- due circuiti combinatori realizzano gli attori
- un registro è posto su ciascuna delle due connessioni da diff a sort

la realizzazione degli attori è semplice, con pochi moduli di uso comune (multiplatori, comparatori e un sottrattore)

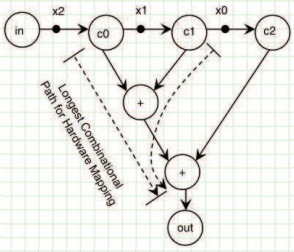



Schaumont, Figure 3.11 - Hardware implementation of Euclid's algorithm

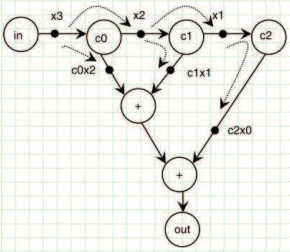
## hardware pipelining

esempio di miglioramento del throughput mediante pipelining:

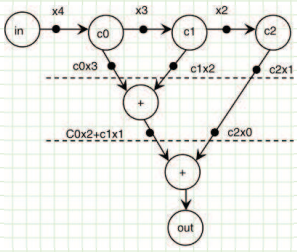
filtro digitale di somma pesata: x0·c2+x1·c1+x2·c0



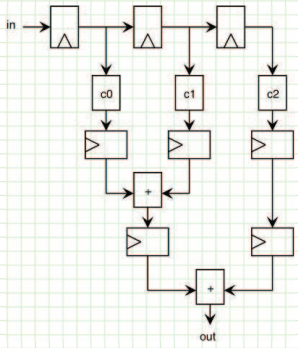

Schaumont, Figure 3.12 - SDF graph of a simple moving-average application




Schaumont, Figure 3.13 - Pipelining the moving-average filter by inserting additional tokens (1)




Schaumont, Figure 3.14 - Pipelining the moving-average filter by inserting additional tokens (2)




Schaumont, Figure 3.15 - Hardware implementation of the moving-average filter

da notare:

- i token iniziali qui hanno il ruolo dei *delay* buffer nella definizione della trasformazione di pipelining
- il grafo SDF è aciclico... (v. appresso)

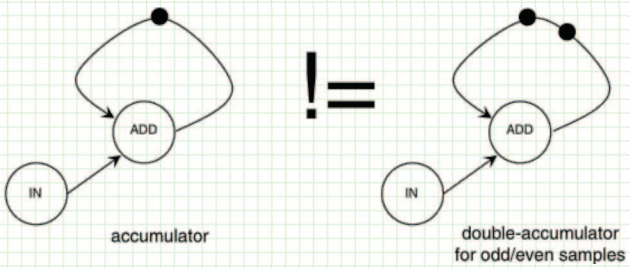





## pipelining in grafi SDF con cicli

il pipelining, con l'aggiunta di token, può alterare il comportamento di un grafo SDF

in particolare, ciò può accadere se si aggiungono token all'interno di un ciclo, come dimostra questo esempio:




Schaumont, Figure 3.16 - Loops in SDF graphs cannot be pipelined

per applicare il pipelining senza alterare il comportamento funzionale del grafo, quando questo contiene cicli, i token aggiuntivi vanno posti al di fuori di qualsiasi ciclo nel grafo

per esempio, sulle linee di ingresso o uscita

## esperienza di laboratorio

il circuito mostrato in figura 3.11 realizza il nucleo computazionale dell'algoritmo di Euclide per il calcolo del GCD, tuttavia non presenta elementi di controllo atti a segnalare l'inizio e la fine del calcolo né a distinguere ingressi e uscita; gli obiettivi di questa esperienza sono: estendere il circuito a tal fine, descriverlo in Gezel, tradurlo in VHDL e simularne il funzionamento

1. estendere lo schematico del circuito di figura 3.11 con tre segnali di input e due di output:
   - `a`, `b`: i dati in ingresso, da 16 bit ciascuno
   - `start`: input da 1 bit, per segnalare la presenza dei dati in ingresso
   - `gcd`: il risultato in uscita, da 16 bit
   - `done`: output da 1 bit, per segnalare la fine del calcolo e la presenza del risultato in uscita

   e con elementi aggiuntivi (multiplatori, eventuale comparatore) utili a realizzare l'obiettivo proposto
2. descrivere in Gezel il circuito prodotto al passo 1
3. tradurre la descrizione Gezel in VHDL mediante il programma `fdlvhd`
4. lanciare Quartus 13.1 (Web edition), crearvi un progetto `EuclidGCD` e assegnargli i file VHDL prodotti al passo 3
5. compilare e simulare il modello VHDL per diverse coppie di dati in ingresso






## riferimenti

letture raccomandate:

Schaumont (2012) Cap. 3, Sez. 3.2