Scheduling Parametric Data Flow Graphs

Vagelis Bebelis
vagelis.bebelis@inria.fr

CPC 2013

Alain Girault (INRIA)
Pascal Fradet (INRIA)
Bruno Lavigueur (STMircoelectronics)

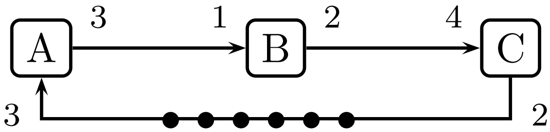
Introduction

Scheduling parametric data flow applications on many core platforms

- Data flow
  - Data flow models
  - Data flow scheduling
- Scheduling framework
  - Scheduling model - STHORM platform (ex. P2012)
  - Scheduling framework

Outline

SDF - Synchronous Data Flow



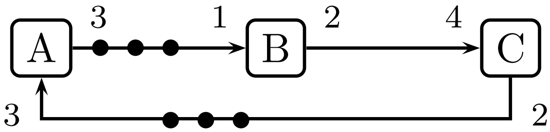
A Synchronous Data Flow graph

- Actors (A, B, C) & edges (AB, BC, CA)
  Function units & Communication links (FIFOs)
- Port rates:
  Number of tokens transferred through a port
- Graph State: $S_i = \begin{bmatrix} 0 & 0 & 6 \end{bmatrix}$
  Number of tokens on the graph's edges

SDF[1]: Port rates are fixed and known at compile time

[1] Lee and Messerschmitt 1987

SDF - Synchronous Data Flow



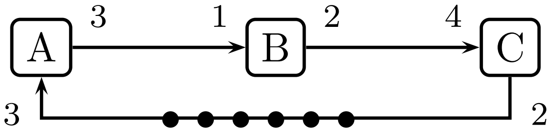
A Synchronous Data Flow graph

- Actors (A, B, C) & edges (AB, BC, CA)
  Function units & Communication links (FIFOs)
- Port rates:
  Number of tokens transferred through a port
- Graph State: $S_i = \begin{bmatrix} 0 & 0 & 6 \end{bmatrix}$
  Number of tokens on the graph's edges

SDF[1]: Port rates are fixed and known at compile time

[1] Lee and Messerschmitt 1987

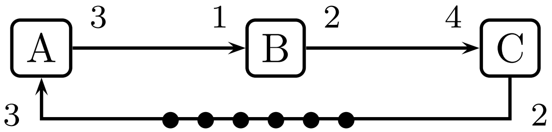
SDF - Analysis



A Synchronous Data Flow graph

- Actor solutions $(\#A, \#B, \#C)$ & repetition vector (r) :
  $\#A = 2\ ,\ \#B = 6\ ,\ \#C = 3\ \Rightarrow\ r = \begin{bmatrix}2 & 6 & 3\end{bmatrix}$
- Iteration: Sequence of firings that return the graph to S_i
- Schedule: Execution of a complete iteration e.g. $A^2; B^6; C^3$
- Liveness: Enough initial tokens on each directed cycle

SDF - Scheduling



A Synchronous Data Flow graph

- Repetition vector: $\begin{bmatrix} 2 & 6 & 3 \end{bmatrix}$
- **Sequential schedules:**
  - $A^2; B^6; C^3$ - Single appearance schedule
  - $A; B^2; C; B; A; B; C; B^2; C$ - Minimum buffer size schedule
- **Parallel schedules:**
  - $A; (A \| B); [B; (B \| C)]^2; B; C$ - As Soon As Possible schedule (ASAP)

SDF - Advantages & Disadvantages

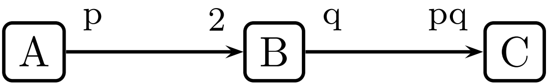
Advantages

+ Modular and reusable design, suitable for DSP
+ Parallelism Exposure
+ Boundedness and liveness guaranteed at compile time
+ Static scheduling - Timing guaranteed

Disadvantages

– Too restrictive to express more advanced applications (e.g. video codec applications)

PDF - Parametric Data Flow



A parametric data flow graph

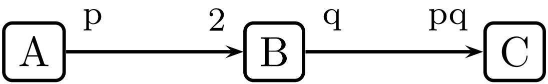
- Simplified version of SPDF [2] and PSDF [3] models
- Parameters change between iterations
- Symbolic analysis of the graph
  Repetition vector: $\begin{bmatrix} 2 & p & 1 \end{bmatrix}$

PDF: Parametric port rates that change between iterations

[2] P.Fradet et al. 2012

[3] B.Bhattacharya and S.Bhattacharyya 2001

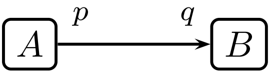
PDF - Scheduling



A parametric data flow graph

- Repetition vector: $\begin{bmatrix} 2 & p & 1 \end{bmatrix}$
- **Sequential schedules:**
  - $A^2; B^p; C$ - Single appearance schedule
- **Parallel schedules:**
  - Difficult to express ASAP schedule

PDF - ASAP scheduling



A PDF edge

- Case $p \geq q$:
  - $A; (A|B)^{q-1}; B^{p-q+1}$
- Case $q > p$:
  - Subcase $q = kp$:
    - $A; (A^{k-1}; (A|B))^{p-1}; A^{k-1}; B$
  - Subcase $q = kp + r,\ 0 < r < p$:
    - Needs iterative comparison of the values of p and q

PDF - Advantages & Disadvantages

Advantages

+ Modular and reusable design, suitable for streaming applications
+ Parallelism Exposure
+ Boundedness and liveness guaranteed at compile time
+ Increased expressiveness

Disadvantages

– (Quasi -) static parallel scheduling possible is more involved (e.g. ASAP scheduling is a challenge)

Outline

STHORM platform

Platform Features

- Many - core platform designed by STMicroelectronics
- 1-32 clusters with 1-16 cores:
  - Software cores: General Purpose Processors (GPP)
  - Hardware cores: HardWare Processing Elements (HWPE)

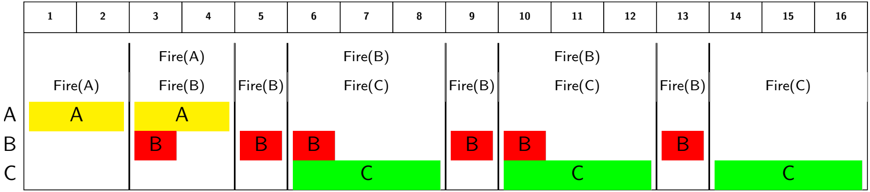
Mapping assumptions

- Application fits in a single cluster
- Each actor is executed by a GPP or implemented as a HWPE
- The schedule is executed by a GPP

Slotted scheduling model

- Compatible with the scheduling model of STHORM
- Uses a slot notion like in blocked scheduling [4]
  - + Actors synchronize after each execution
  - + Reduces complexity of parallel scheduling
  - + Compatible with other parallel programming models (CUDA, OpenGL)
  - − May introduce slack

Repetition vector: $\begin{bmatrix} 2 & 6 & 3 \end{bmatrix}$



[4] S.Ha et al. 1991

Scheduling framework features

The framework should

- Automatically produce ASAP schedules
- Be expressive and flexible for different
  - Platforms
  - Optimization criteria
  - Scheduling strategies

Main idea: Production of different schedules with the same (ASAP) algorithm

Scheduling framework overview

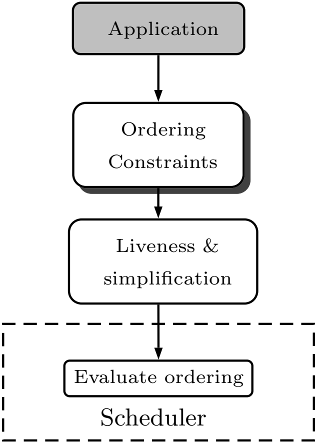


Figure : Scheduling framework

Scheduling framework overview

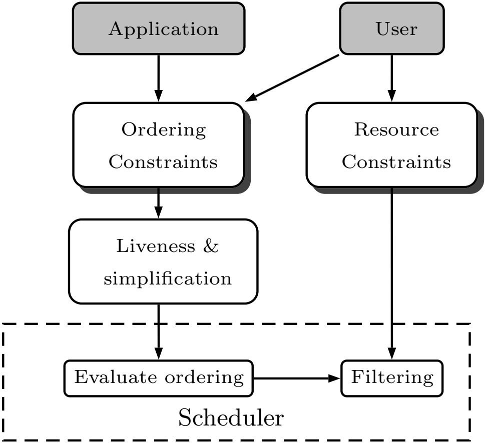


Figure : Scheduling framework

Scheduling constraints

- Ordering Constraints: Express the partial ordering of the firings

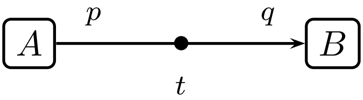
$$\boldsymbol{X_i > Y_{f(i)}}$$

- Resource Constraints: Control the parallel execution

$$\textbf{replace } S_A \textbf{ by } S_B \textbf{ if } \textit{condition}$$

$$\text{where } S_B \subseteq S_A \text{ and } S_B \neq \emptyset$$

Constraint Examples



Graph Constraint: Data dependency

$$B_i > A_{f(i)} \quad \text{with} \quad f(i) = \left\lceil \frac{q \cdot i - t}{p} \right\rceil$$

User Contraint: Buffer capacity restriction to k

$$A_i > B_{g(i)} \quad \text{with} \quad g(i) = \left\lceil \frac{p \cdot i + t - k}{q} \right\rceil$$

Resource Constraint: Mutual exclusion of A and B

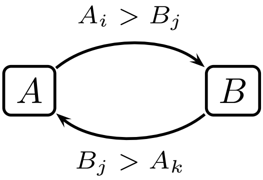
replace $\{A, B\}$ **by** $\{A\}$

Constraint deadlock Detection

Deadlock

A set of ordering constraints deadlocks when it implies (by transitivity) a constraint of the form:

$$\exists A, i, j, (A_i > A_j) \wedge (i \leq j)$$



$\Rightarrow \; A_i > A_k$

$\forall$ cycle $A_i > A_k$

check if $\; i > k$

Deadlock detection example

Constraints:

$$B_i > A_{f(i)}$$
$$A_i > B_{g(i)}$$

Cycle:

$$A_i > A_{f(g(i))}$$

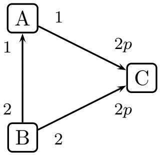
Deadlock free condition:

$$i > f(g(i))$$

Solution:

$$\begin{aligned} i > f(g(i)) &\Leftrightarrow i > \left\lceil \frac{q \cdot \lceil \frac{p \cdot i - k}{q} \rceil}{p} \right\rceil \\ &\Leftarrow i > \frac{q \cdot (\frac{p \cdot i - k}{q} + 1)}{p} + 1 \\ &\Leftarrow i > i + \frac{q - k}{p} + 1 \\ &\Leftarrow k > p + q \\ &\Leftarrow k > p_{max} + q_{max} \end{aligned}$$

Constraint simplification



Constraints:

$$A_i > B_{\lceil \frac{i}{2} \rceil}$$

$$C_i > B_{\lceil \frac{2pi}{2} \rceil}$$

$$C_i > A_{pi}$$

Firing Function:

$$B_i = i \quad , \quad i \in [1 \cdots p]$$

$$A_i = max(B_{\lceil \frac{i}{2} \rceil}, A_{i-1}) + 1 \quad , \quad i \in [1 \cdots 2p]$$

$$\Rightarrow A_i = 2i + 1$$

$$C_i = max(B_{pi}, A_{pi}, C_{i-1}) + 1 \quad , \quad i = 1$$

$$\Rightarrow C_i = max(A_{pi}, C_{i-1}) + 1$$

$$\Rightarrow C_i = 2p + 1$$

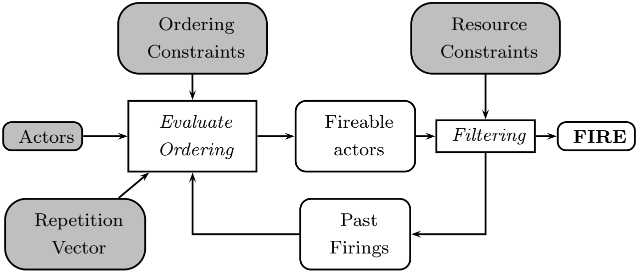
Schedules:

$$B : \ \mathcal{F}^p$$

$$A : \ \mathcal{E}; (\mathcal{E}; \mathcal{F})^{2p}$$

$$C : \ \mathcal{E}^{2p}; \mathcal{F}$$

Run-time scheduler



Overall small overhead:

- Concurrent execution with actors
- Coarse - grain graph
- Optimization of static parts of the graph

Conclusions

We presented a **scheduling framework for PDF** applications that:

- Flexible constraint framework for PDF graphs:
- Modular way to adjust the schedule
- Expressive power to optimize the schedule
- Automatically generates of ASAP schedules
- Statically guarantees boundness and liveness of the schedule

On-going work

- Implementation and integration within ST's SDK
- Evaluation of the scheduler with real world applications

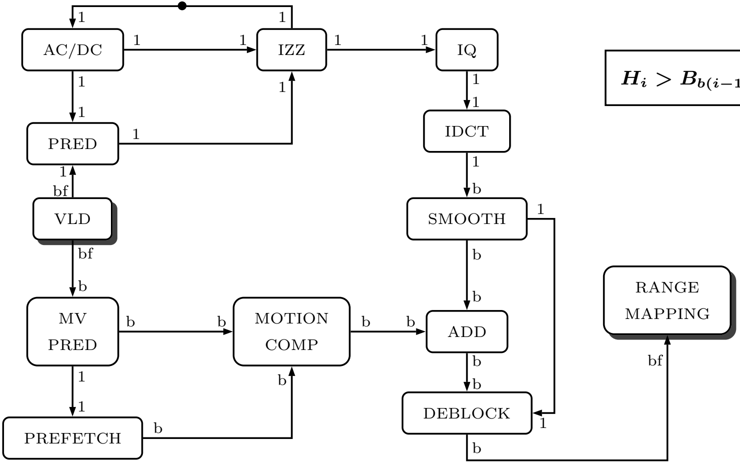
Future work

- Introduction of timing information
- Flexible slotted scheduling model already used on the platform

Questions

Thank you for your attention!

Use case example: VC1 decoder



$H_i > B_{b(i-1)}$

VC-1 capture in PDF

Resource constraint examples

- Mutual Exclusion (A and B: specific actors)

$$\textbf{replace } A,\ B \textbf{ by } A$$

- Bounded Parallelism (x,y,z: variables - can be any actor)

$$\textbf{replace } x,\ y,\ z \textbf{ by } x,\ y$$

- Timing optimization

$$\textbf{replace } x,\ y \textbf{ by } x \textbf{ if } short(x) \ \wedge\ long(y)$$

- Power optimization

$$\textbf{replace } x,\ y \textbf{ by } x \textbf{ if } high(x) \ \wedge\ high(y)$$

$$\textbf{replace } x,\ y,\ z \textbf{ by } x,\ y \textbf{ if } high(x) \ \wedge\ low(y) \ \wedge\ low(z)$$