Parametric Dataflow Programming: Model of Computation and Many-Core Scheduling

Alain Girault (alain.girault@inria.fr)

The 9th Scheduling for Large Scale Systems Workshop, July 2014

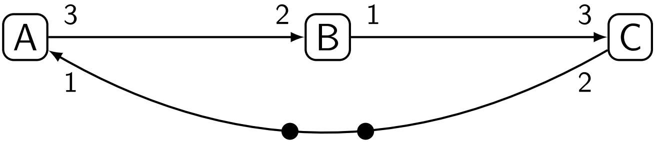
Joint work with
Vagelis Bebelis & Pascal Fradet (INRIA)

Presentation Overview

Outline

1. Data Flow Models of Computation
   - Synchronous Data Flow
   - Boolean Parametric Data Flow
2. Scheduling Framework
3. Experiments
4. Conclusions

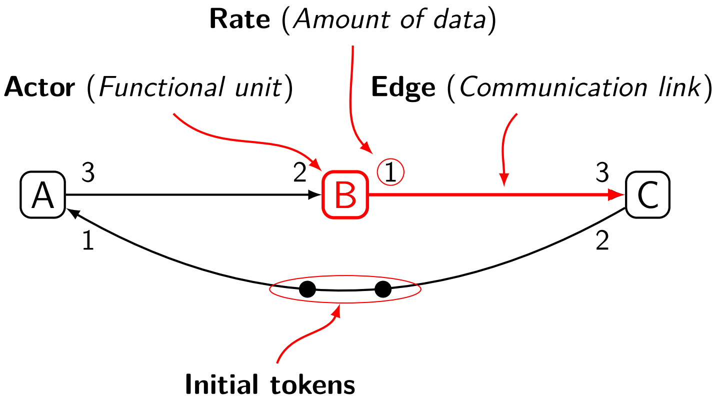
SDF - Synchronous Data Flow [1]



An SDF graph

[1] E.A.Lee and D.G.Messerschmitt, Proc. of the IEEE, 1987

SDF - Synchronous Data Flow [1]

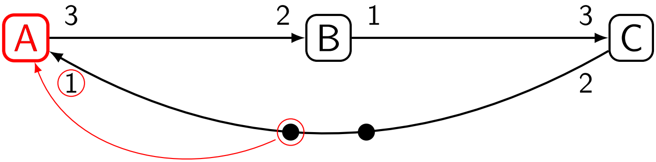


In SDF all rates are **fixed and known** at compile time

[1] E.A.Lee and D.G.Messerschmitt, Proc. of the IEEE, 1987

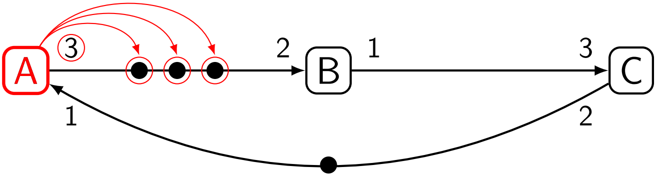
SDF – Firing

Firing of actor A**:** Consumes 1 token

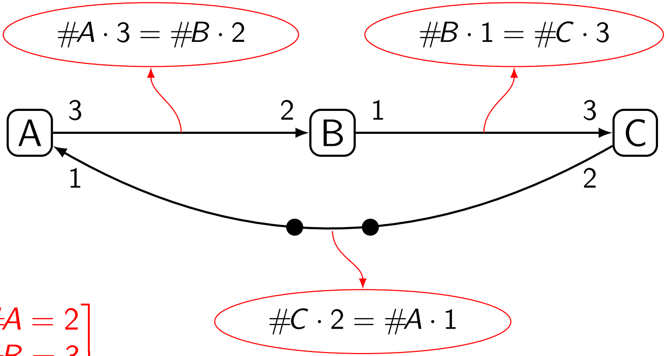


SDF – Firing

Firing of actor A**:** Produces 3 tokens



SDF – Balance Equations

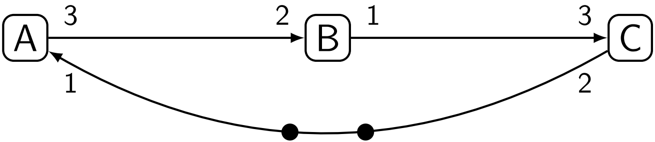


$$\begin{bmatrix} \#A = 2 \\ \#B = 3 \\ \#C = 1 \end{bmatrix} \longrightarrow [A^2\ B^3\ C]$$

SDF – Scheduling

Schedule: Series of firings that complete one iteration

Here: $[A^2\ B^3\ C]$



Sequential Single Appearance	$A^2; B^3; C$
Sequential Minimum Buffer Size	$A; B; A; B^2; C$
Parallel ASAP	$A; (A \| B); B^2; C$

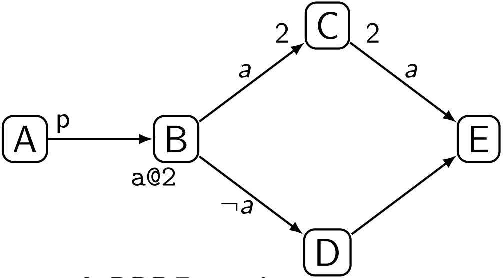
Parametric data flow models

Need for more expressive data flow models

- SDF is not expressive enough for complex applications.
- **More expressive models are employed that use**
  - Parametric rates
  - Dynamic graph topology
- Both features make scheduling more difficult.
- Such a model is Boolean Parametric Data Flow (BPDF)[1].

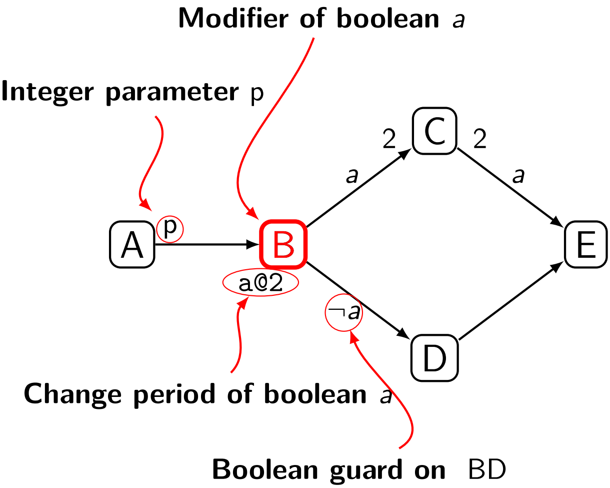
[1] V.Bebelis, P.Fradet, A.Girault and B.Lavigueur, EMSOFT'13, 2013

Boolean Parametric Data Flow - BPDF



A BPDF graph

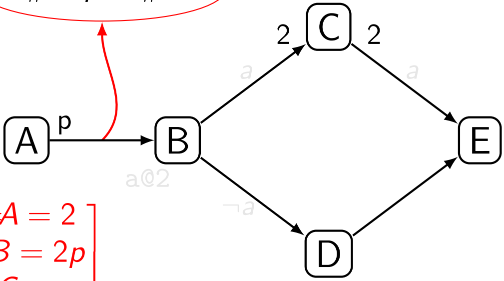
Boolean Parametric Data Flow - BPDF



BPDF - Balance Equations

BPDF analysis: Balance Equations

$$\#A \cdot p = \#B$$

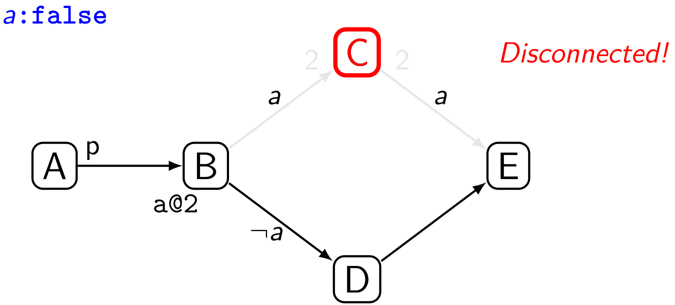


$$\begin{bmatrix} \#A = 2 \\ \#B = 2p \\ \#C = p \\ \#D = 2p \\ \#E = 2p \end{bmatrix} \longrightarrow [A^2 \; B^{2p} \; C^p \; D^{2p} \; E^{2p}]$$

Parametric solution of balance equations

BPDF - Balance Equations

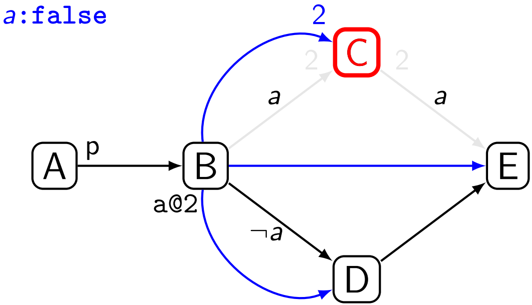
BPDF analysis: Balance Equations



Actor **C** fires despite being disconnected

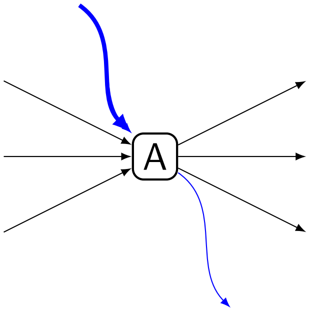
BPDF - Balance Equations

BPDF analysis: Balance Equations



There are implicit boolean propagation links

BPDF - Actor firing



(1) Read boolean parameters

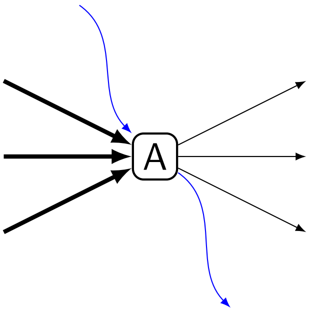
(2) Read data from connected inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Write data to connected outputs

BPDF - Actor firing



(1) Read boolean parameters

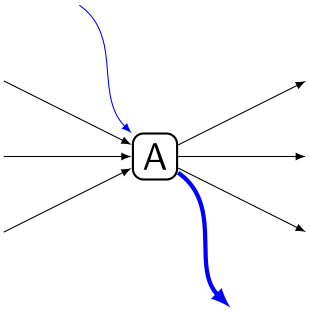
(2) Read data from connected inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Write data to connected outputs

BPDF - Actor firing



(1) Read boolean parameters

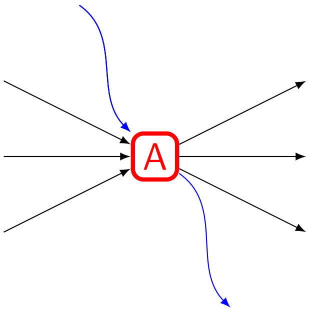
(2) Read data from connected inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Write data to connected outputs

BPDF - Actor firing



(1) Read boolean parameters

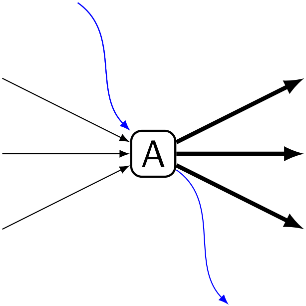
(2) Read data from connected inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Write data to connected outputs

BPDF - Actor firing



(1) Read boolean parameters

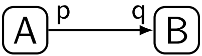
(2) Read data from connected inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Write data to connected outputs

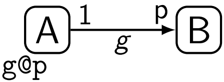
BPDF Scheduling - Integer parameters



Sequential Single Appearance	$A^q; B^p$
Sequential Minimum Buffer Size	Difficult to express
Parallel ASAP	Difficult to express

Needs iterative comparison of the values of p and q

BPDF Scheduling - Boolean parameters



Parallel ASAP schedule when $g = \mathit{true}$	$A^p; B$
Parallel ASAP schedule when $g = \mathit{false}$	$A; (A \| B); A^{p-2}$

Also needs constant checking of the boolean values

Outline

STHORM platform

Platform Features

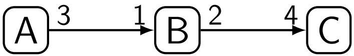
- Many - core platform designed by STMicroelectronics
- 1-32 clusters with 1-16 cores:
  - Software cores: General Purpose Processors (GPP)
  - Hardware cores: HardWare Processing Elements (HWPE)

Mapping assumptions

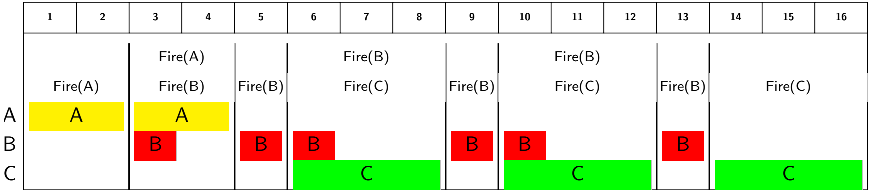
- Application fits in a single cluster
- Each actor is executed on a GPP or implemented as a HWPE
- The schedule is executed on a GPP

Slotted scheduling model

- Compatible with the scheduling model of STHORM.
- Uses a slot notion like in blocked scheduling [2]
  - \+ Actors synchronize after each execution
  - \+ Reduces complexity of parallel scheduling
  - \+ Compatible with other parallel programming models (CUDA, OpenGL)
  - – May introduce slack



$[A^2\ B^6\ C^3]$



[2] S.Ha et al., IEEE Trans. On Computers, 1991

Scheduling framework features

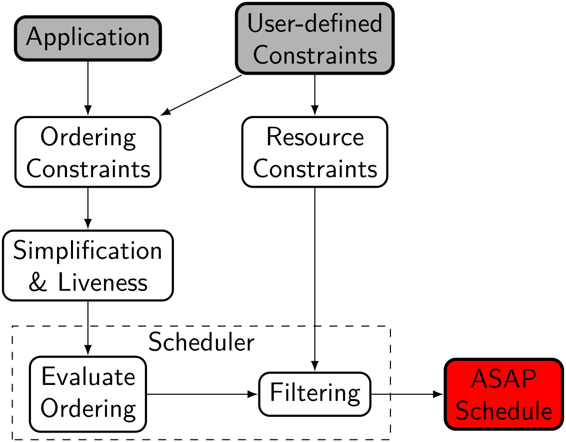
The framework should

- Automatically produce ASAP schedules
  - Best strategy when timing is unknown [3]
- Be expressive and flexible for different
  - Platforms
  - Optimization criteria
  - Scheduling strategies

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

[3] S.Sriram and S.S. Btathacharyya. Embedded Multiprocessors: Scheduling and Synchronization. 2000

Scheduling framework overview



Scheduling constraints

- Ordering Constraints: Express the partial ordering of the firings

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

where X_i is the ith firing of actor X

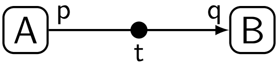
- Resource Constraints: Control the parallel execution

replace S_A **by** S_B **if** *condition*

where S_A and S_B are subsets of actors such that $S_A \supseteq S_B \neq \emptyset$

(used for filtering the preliminary schedule)

Application Constraints



Graph constraint: Data dependency

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

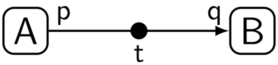
$$\text{because } t + f(i) \cdot p \geq q \cdot i \Longleftrightarrow f(i) \geq \frac{q \cdot i - t}{p}$$

Modifier to user constraint: Boolean dependency

$$\boldsymbol{U_i > M_{f(i)}} \qquad \text{with} \qquad f(i) = \pi_w \cdot \left\lfloor \frac{i-1}{\pi_r} \right\rfloor + 1$$

where π_r (resp. π_w) is the reading (resp. writing) period of U (resp. M)

User Constraint Examples



Buffer Contraint: Buffer capacity restriction to k

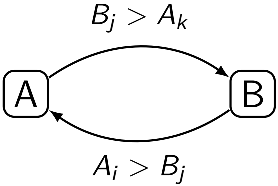
$$\boldsymbol{A_i} > \boldsymbol{B_{g(i)}} \qquad \text{with} \qquad g(i) = \left\lceil \frac{p \cdot i + t - k}{q} \right\rceil$$

Resource Constraint: Mutual exclusion of A and B

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

Constraint liveness condition

User ordering constraints may introduce deadlocks



$$\Rightarrow A_i > A_k$$

Liveness condition:

$\forall$ cycle $A_i > A_k$

we need $i > k$

Deadlock detection example

Constraints:

Application: $B_i > A_{f(i)}$

Buffer: $A_i > B_{g(i)}$

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

Liveness condition:

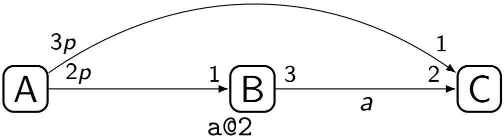
$$i > f(g(i))$$

Solution:

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

Constraint simplification

$[A\ B^{2p}\ C^{3p}]$



Constraints

$$B_i > A_{\left\lceil \frac{i}{2p} \right\rceil}$$
$$C_i > B_{\left\lceil \frac{2i}{3} \right\rceil}$$
$$C_i > A_{\left\lceil \frac{i}{3p} \right\rceil}$$
$$C_i > B_{2\left\lceil \frac{i}{3} \right\rceil - 1}$$

$$A_1 = 1$$

$$B_i = max(A_1, B_{i-1}) + 1 \quad \text{for } i \in [1..2p]$$
$$B_i = i + 1$$

$$C_i = max(A_1, B_{\left\lceil \frac{2i}{3} \right\rceil}, B_{2\left\lceil \frac{i}{3} \right\rceil - 1}, C_{i-1}) + 1 \quad \text{for } i \in [1..3p]$$
$$C_i = i + 2$$

ASAP Schedule: $A; B; (B \| C)^{2p-1}; C^{p+1}$

Run-time scheduler

If simplification is not possible then a run-time scheduler is employed

Small overhead:

- Concurrent execution with actors
- Coarse - grain graphs (small number of actors)
- Simplification of constraints at compile time
- Optimization of static parts of the graph

Outline

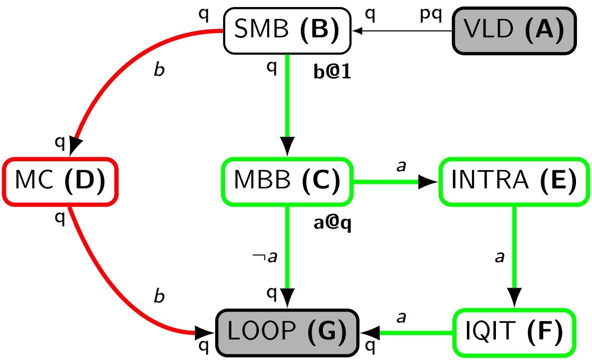
Scheduler Overhead

- Implementation of TNR (Temporal Noise Reduction) application on STHORM platform
- Overhead comparison between
  - Dynamic scheduler
  - Simplified scheduler
  - Manually optimized schedule

	Best Actor Performance	Dynam. sched.	Simpl. sched.	Manual Sched.
Cycles / frame	2.140.000	1.100.000	360.000	340.000

Schedule overhead for different schedules of TNR

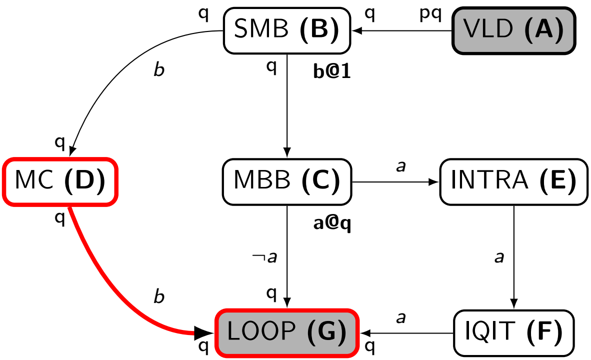
Use Case: VC-1 decoder



- Inter pipeline
- Intra pipeline

Repetition vector:
$[A\ B^p\ C^{pq}\ D^p\ E^{pq}\ F^{pq}\ G^p]$

Buffer restriction



Token accumulation on edge DG

- Maximum buffer estimation: $\sim pq - p$
- Buffer restriction to: q

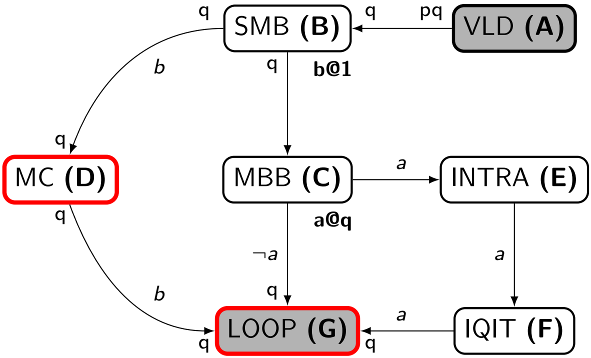
$$D_i > G_{\left\lceil \frac{q \cdot i - q}{q} \right\rceil}$$

- Increase of 2% in total schedule duration

Repetition vector:

$[A\ B^p\ C^{pq}\ D^p\ E^{pq}\ F^{pq}\ G^p]$

Timing Optimization



When timing is known, the goal is to minimize slack

- Clustering of actors D and G in the same slot with:

replace $D, \ E$ **by** E **if** $\neg$`fireable`(G)

- Improvement of 15% in the schedule duration.

Repetition vector:
$[A\ B^p\ C^{pq}\ D^p\ E^{pq}\ F^{pq}\ G^p]$

Outline

Conclusions

We proposed a scheduling framework for BPDF applications that

- is flexible and modular through constraints
- is expressive to optimize the schedule
- automatically generates ASAP parallel schedules
- Statically guarantees the boundness and liveness of the produced schedule

Ongoing and future work

- Use the framework with a non-slotted scheduling model
- Formalization of the constraint simplification procedure
- Use the framework to optimize bi-criteria scheduling, specifically power consumption vs. throughput
- Design a high-level language to express scheduling policies that can be automatically compiled into constraints
- Implementation of BPDF within Ptolemy II
- Formal comparison between BPDF and SADF [4]

[4] Stuijk et al, IC-SAMOS'11

Thank you for your attention!

Questions?

Run-time scheduler