Boolean Parametric Data Flow

Vagelis Bebelis
vagelis.bebelis@inria.fr

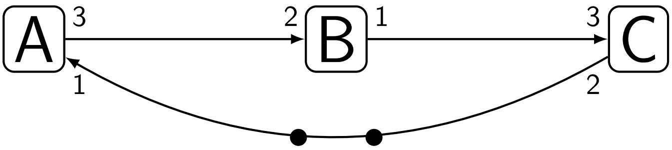
Streaming Day

Joint work with
Pascal Fradet
Alain Girault

Outline

1. Data Flow Models of Computation
   - Synchronous Data Flow
   - Motivation
   - Boolean Parametric Data Flow
   - Related Models
   - Conclusions
2. Scheduling
3. Current work

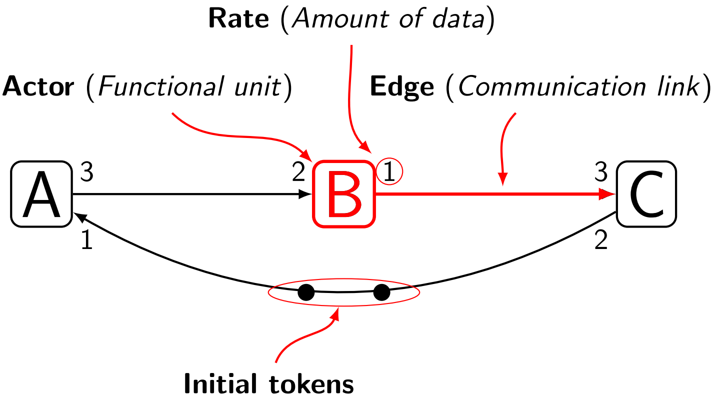
Synchronous Data Flow [1] - SDF



An SDF graph

[1] Lee and Messerschmitt 1987

Synchronous Data Flow [1] - SDF

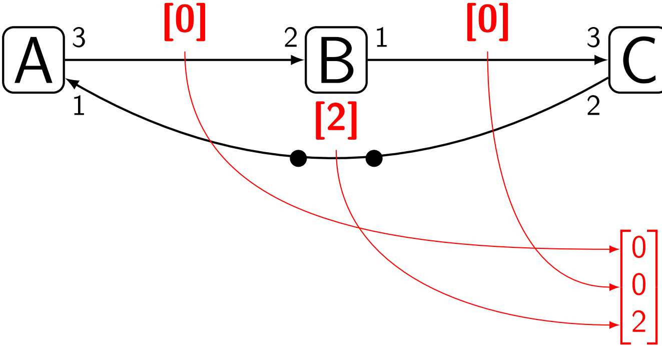


In SDF all rates are **fixed and known** at compile time

[1] Lee and Messerschmitt 1987

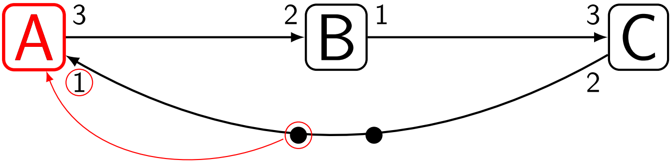
Synchronous Data Flow - Graph State

Graph state: Data stored on its edges



Synchronous Data Flow - Firing

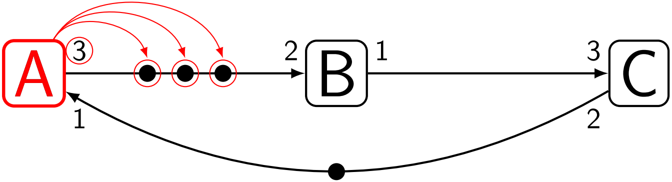
Firing of actor A**:** Consumes 1 token



$$\begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

Synchronous Data Flow - Firing

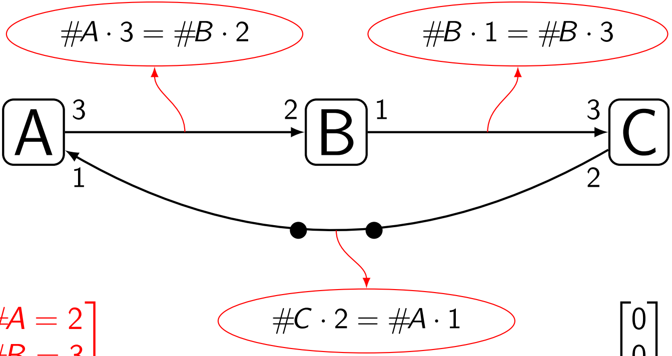
Firing of actor ***A*****:** Produces 3 tokens



$$\begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$$

Synchronous Data Flow - Consistency

SDF analysis: Consistency

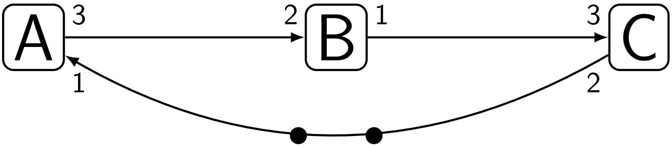


$$\begin{bmatrix} \#A = 2 \\ \#B = 3 \\ \#C = 1 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

Synchronous Data Flow - Boundedness

SDF analysis: Boundedness

There is **no** accumulation of tokens as the graph returns to its initial state

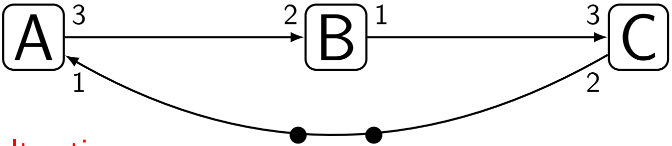


$$\begin{bmatrix} \#A = 2 \\ \#B = 3 \\ \#C = 1 \end{bmatrix} \qquad \begin{array}{ccccccccccccc} & & \text{(A)} & & \text{(A)} & & \text{(B)} & & \text{(B)} & & \text{(B)} & & \text{(C)} \\ \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} & \blacktriangleright & \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} & \blacktriangleright & \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} & \blacktriangleright & \begin{bmatrix} 4 \\ 1 \\ 0 \end{bmatrix} & \blacktriangleright & \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} & \blacktriangleright & \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} & \blacktriangleright & \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \end{array}$$

Synchronous Data Flow - Iteration

SDF analysis: Boundedness

There is **no** accumulation of tokens as the graph returns to its initial state



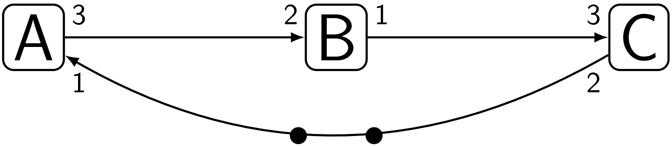
One Iteration

$$\begin{bmatrix} \#A = 2 \\ \#B = 3 \\ \#C = 1 \end{bmatrix} \qquad \begin{bmatrix}0\\0\\2\end{bmatrix} \xrightarrow{(A)} \begin{bmatrix}3\\0\\1\end{bmatrix} \xrightarrow{(A)} \begin{bmatrix}6\\0\\0\end{bmatrix} \xrightarrow{(B)} \begin{bmatrix}4\\1\\0\end{bmatrix} \xrightarrow{(B)} \begin{bmatrix}2\\2\\0\end{bmatrix} \xrightarrow{(B)} \begin{bmatrix}0\\3\\0\end{bmatrix} \xrightarrow{(C)} \begin{bmatrix}0\\0\\2\end{bmatrix}$$

Synchronous Data Flow - Liveness

SDF analysis: Liveness

There exists a schedule completing one iteration
or *Are there enough initial tokens?*



If there exists, it can be repeated **indefinitely**
and the graph is live

Synchronous Data Flow - Conclusions

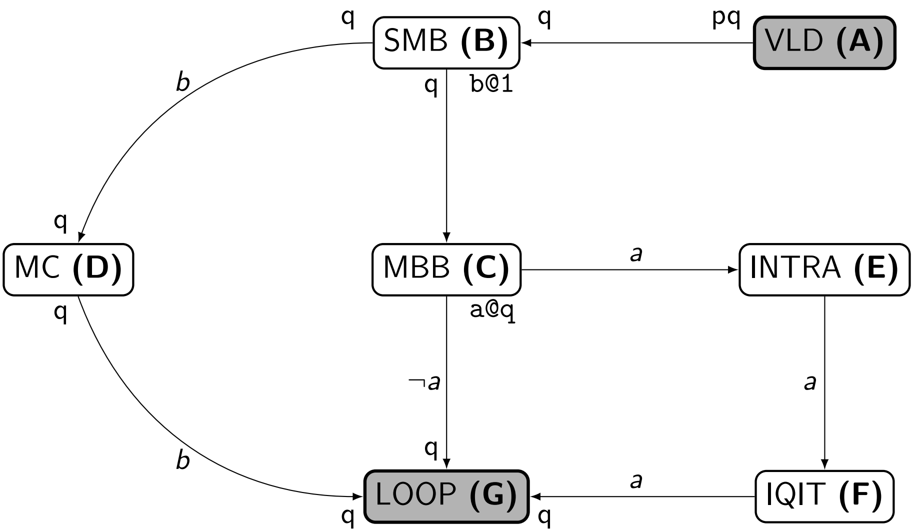
Advantages

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

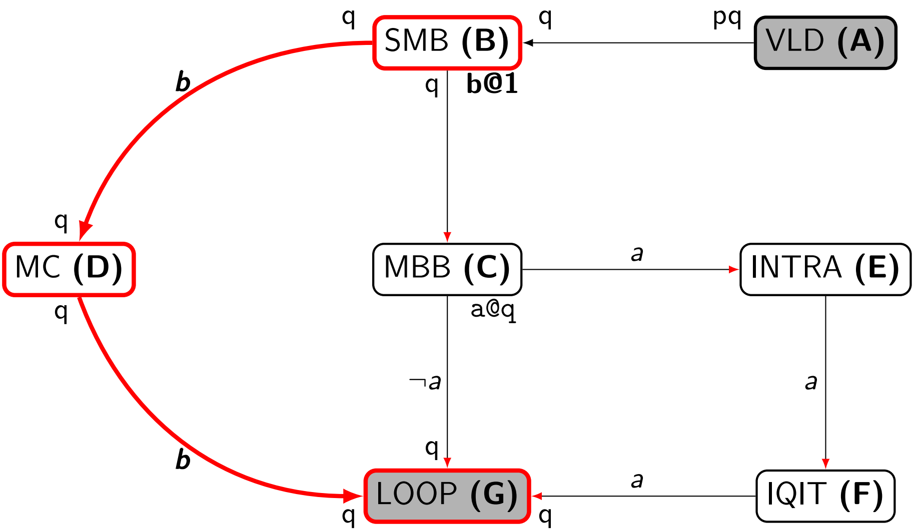
Disadvantages

– Too restrictive to express more **advanced applications**

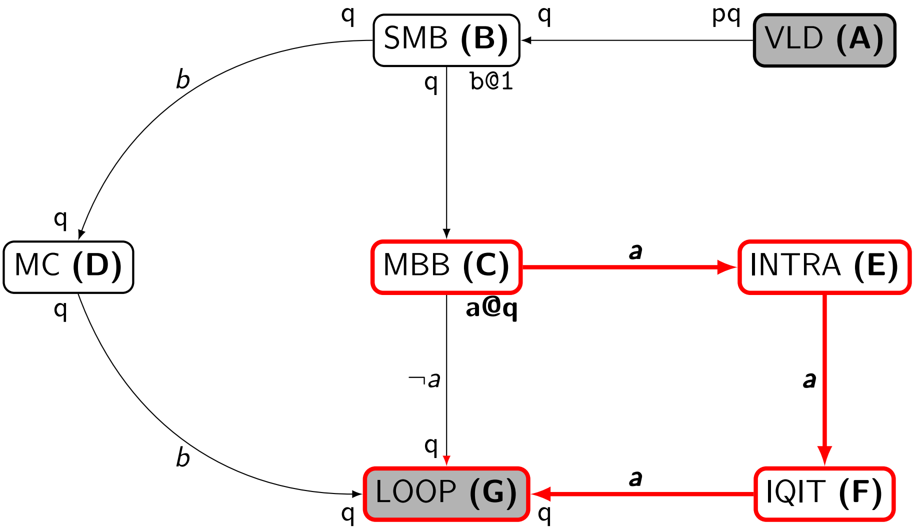
Motivation - VC-1 decoder



Motivation - Inter pipeline



Motivation - Intra pipeline



Motivation

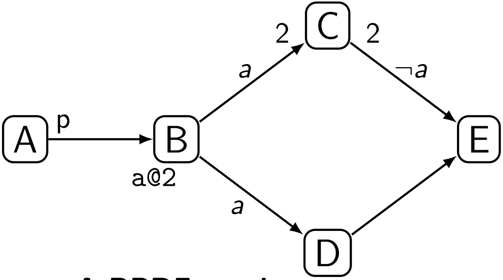
SDF is not expressive enough for more complex applications.

We want to increase SDF expressiveness with

- Parametric rates
- Dynamic graph topology

... while keeping all the static guarantees

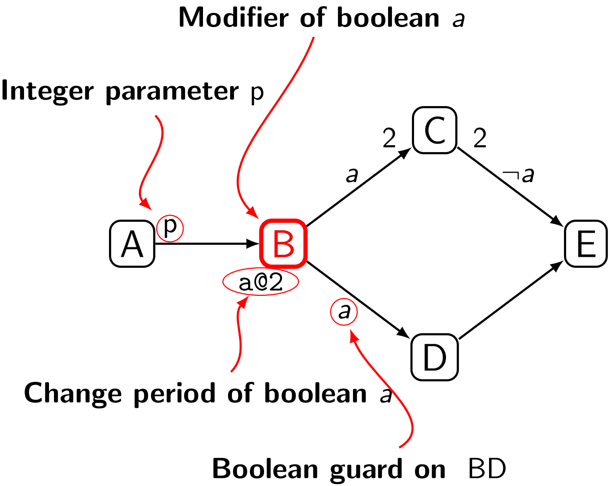
Boolean Parametric Data Flow [1] - BPDF



A BPDF graph

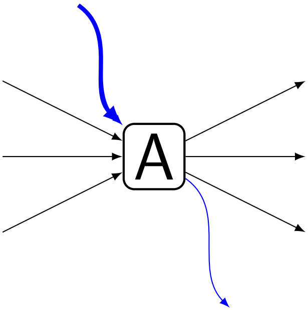
[1] Bebelis, Fradet and Girault 2013

Boolean Parametric Data Flow [1] - BPDF



[1] Bebelis, Fradet and Girault 2013

BPDF - Actor firing



(1) Read boolean parameters

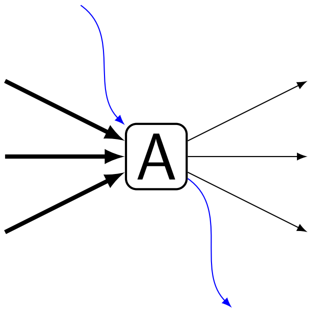
(2) Read data from inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Produce data on outputs

BPDF - Actor firing



(1) Read boolean parameters

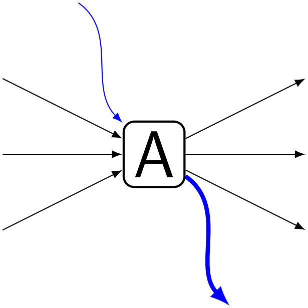
(2) Read data from inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Produce data on outputs

BPDF - Actor firing



(1) Read boolean parameters

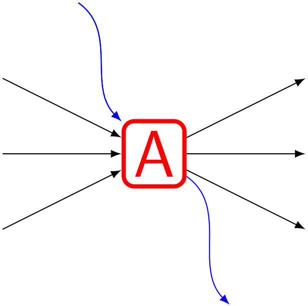
(2) Read data from inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Produce data on outputs

BPDF - Actor firing



(1) Read boolean parameters

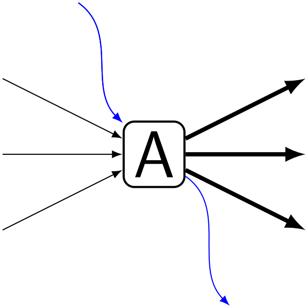
(2) Read data from inputs

(3) Set boolean parameters

(4) ... Compute ...

(5) Produce data on outputs

BPDF - Actor firing



(1) Read boolean parameters

(2) Read data from inputs

(3) Set boolean parameters

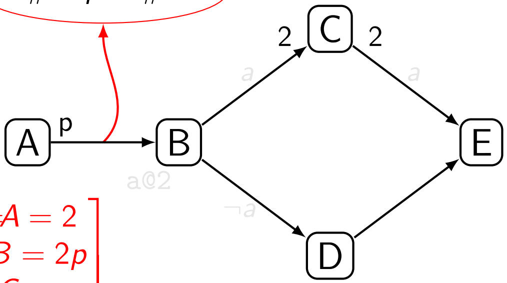
(4) ... Compute ...

(5) Produce data on outputs

BPDF - Consistency

BPDF analysis: Consistency

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

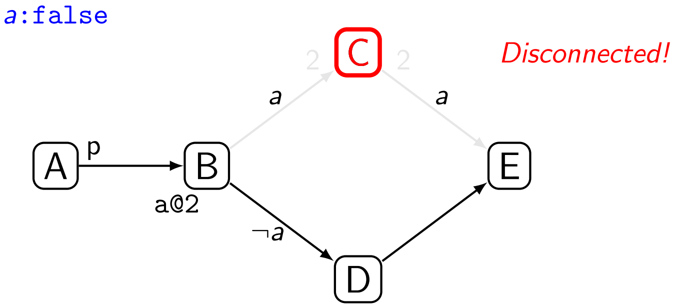


$$\begin{bmatrix} \#A = 2 \\ \#B = 2p \\ \#C = p \\ \#D = 2p \\ \#E = 2p \end{bmatrix} \rightarrow [A^2 \;\; B^{2p} \;\; C^p \;\; D^{2p} \;\; E^{2p}]$$

Parameteric soluton of balance equations

BPDF - Consistency

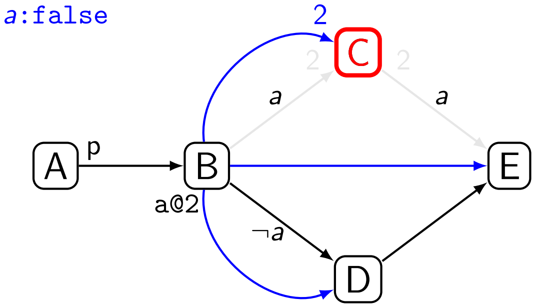
BPDF analysis: Consistency



C although disconnected still fires

BPDF - Consistency

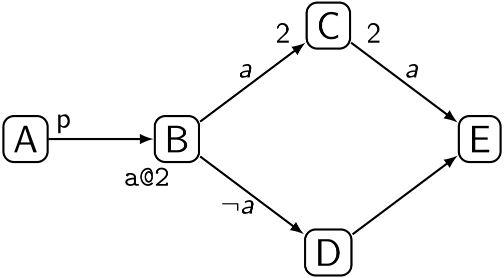
BPDF analysis: Consistency



There are boolean propagation links

BPDF - Boundedness

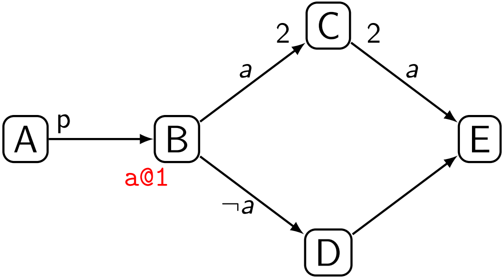
BPDF analysis: Boundedness



In SDF, **consistency** suffices

BPDF - Boundedness

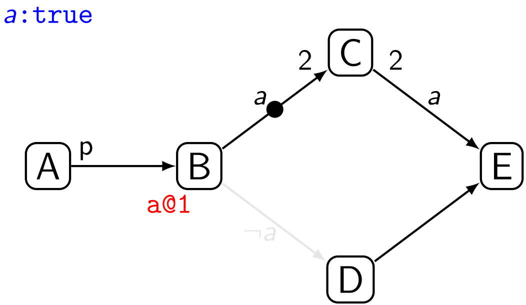
BPDF analysis: Boundedness



What happens if the period of a changes to 1?

BPDF - Boundedness

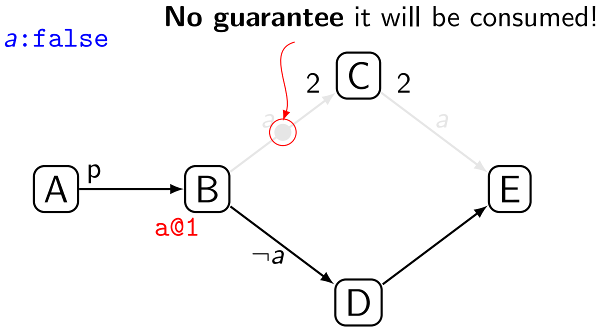
BPDF analysis: Boundedness



B produces a token on **BD**

BPDF - Boundedness

BPDF analysis: Boundedness

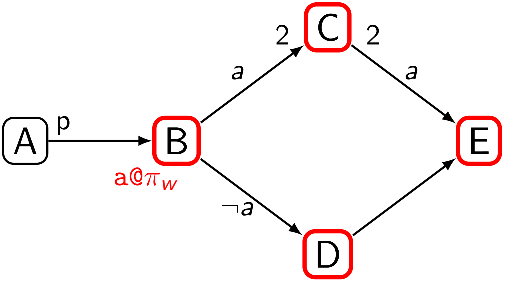


Not all periods are safe and should be checked.

BPDF - Boundedness

BPDF analysis: Boundedness

$[B^{2p}\ \ C^{p}\ \ D^{2p}\ \ E^{2p}]$



Region of boolean a and solutions

BPDF - Boundedness

BPDF analysis: Boundedness

$[B^{2p}\ C^{p}\ D^{2p}\ E^{2p}]$

Boolean **cannot** change during a local iteration

Can be factorized by $f = \boldsymbol{p}\ or\ \mathbf{1}$

$$\pi_w = \frac{\#B}{f} \Rightarrow \pi_w = \mathbf{2}\ or\ \mathbf{2}\boldsymbol{p}$$

Period Safety Criterion:

A BPDF graph is period safe if and only if, for each boolean parameter $b \in \mathcal{P}_b$ and each actor $X \in \mathfrak{R}(b)$,

$$\exists k \in \mathbb{N},\ \#X = k \cdot \frac{M(b)}{\alpha(b)}$$

BPDF - Boundedness

BPDF analysis: Boundedness

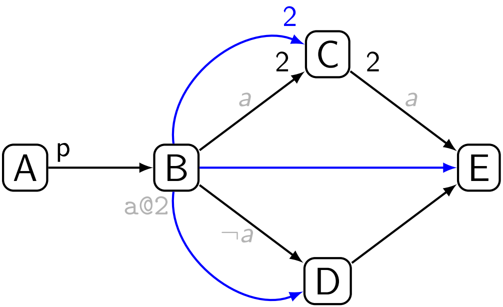
A BPDF graph is **bounded** if

- it is consistent and
- all its boolean parameters

satisfy the period safety criterion

BPDF - Liveness

BPDF analysis: Liveness



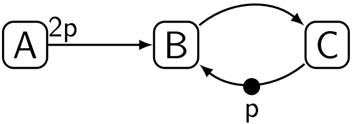
For **liveness** analysis we consider
the boolean propagation links
while disregarding the boolean parameters

BPDF - Liveness

BPDF analysis: Liveness

A BPDF graph is **live** when a schedule of an iteration exists.

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



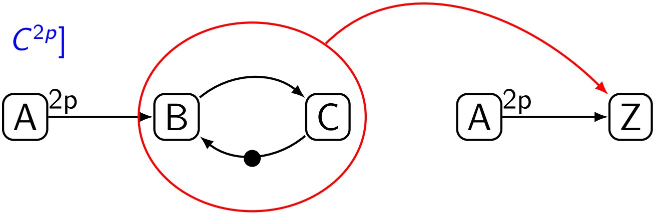
Parametric SDF-Like Liveness Checking (PSLC)
The PSLC algorithm finds the schedule $A(B^p C^p)^2$

BPDF - Liveness

BPDF analysis: Liveness

A BPDF graph is **live** when a schedule of an iteration exists.

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



Clustering cycles + PSLC

Cluster B and C into Z with local schedule BC

PSLC finds the schedule AZ^{2p}

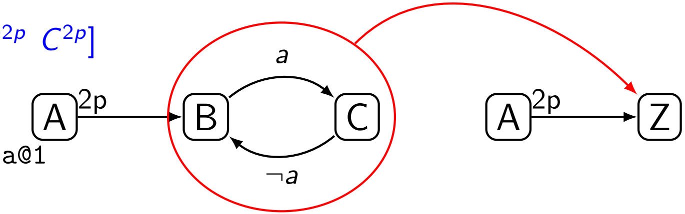
which unfolds to $A(BC)^{2p}$

BPDF - Liveness

BPDF analysis: Liveness

A BPDF graph is **live** when a schedule of an iteration exists.

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



False cycles + Clustering + PSLC

Cluster B and C into Z with conditional schedule

$A(\text{if } a \text{ then } BC \text{ else } CB)^2 p$

PSLC finds the schedule AZ^{2p}

Related models - Integer Parameters

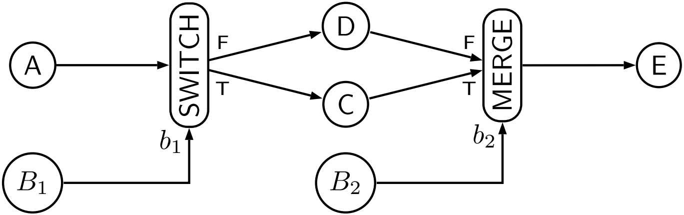
- Parametric Synchronous Data Flow - **PSDF** [2]
  - PSDF uses hierarchy and two auxiliary actors to introduce integer parameters.
  - The model does not provide formal guarantees
  - Does not use boolean parameters
- Schedulable Parametric Data Flow - **SPDF** [3]
  - SPDF is very expressive model that allows change of integer parameters during an iteration
  - It is really complex to schedule and combine with boolean parameters

[2] Bhattacharya and Bhattacharyya 2001

[3] Fradet *et al.* 2012

Related models - Boolean Parameters

- Boolean Data Flow - **BDF** [4]
- Integer Data Flow - **IDF** [5]

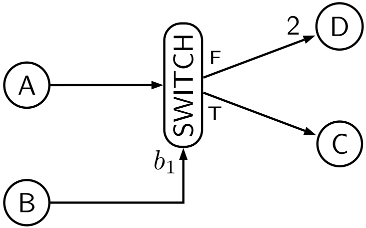


BDF graph with SWITCH and MERGE actors

[4] Buck 1993
[5] Buck 1995

BDF - Undecidability

- Both BDF abd IDF models are Turing complete models
- They suffer from the undecidability of the Halting Problem

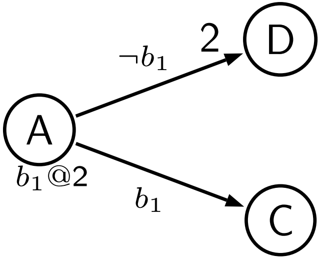


Example of undecidable BDF graph

- Firing of actor D is not guaranteed.
- The graph is not guaranteed to return to its initial state.

BPDF Restrictions

- BPDF restricts the expressiveness of the Boolean parameters to obtain static guarantees
- The period safety criterion guarantees that D will always fire and finish the iteration
- This renders BPDF independent of the values of the boolean parameters.



BPDF - Conlcusions

Boolean Parameteric Data Flow

- combines integer and boolean parameters to allow
  - change of port rates at run time
  - change of graph topology at run time
- while being statically analyzable with
  - Boundedness guaranteed at compile time
  - Liveness guaranteed at compile time

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)

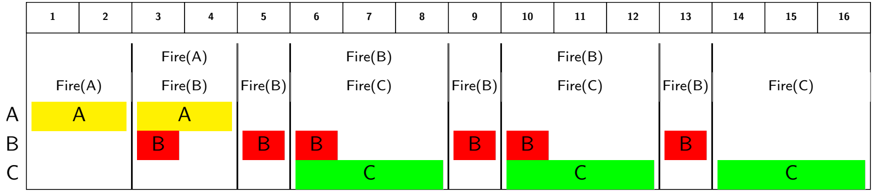
Mapping assumptions

- 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 [6]
  - + Actors synchronize after each execution
  - + Reduces complexity of parallel scheduling
  - + Compatible with other parallel programming models (CUDA, OpenGL)
  - − May introduce slack

$A \xrightarrow{3 \qquad 1} B \xrightarrow{2 \qquad 4} C$ Rep. vector: $\left[A^2 \quad B^6 \quad C^3\right]$



[6] S.Ha et al. 1991

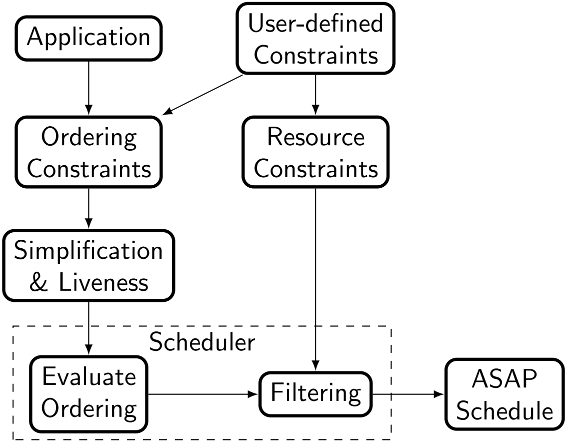
Scheduling framework features

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



Scheduling constraints

- Ordering Constraints: Express the partial ordering of the firings

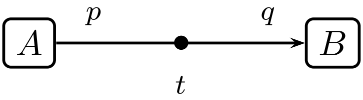
$$\boldsymbol{X_i > Y_{f(i)}}$$

- Resource Constraints: Control the parallel execution

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

where $S_B \subseteq S_A$ and $S_B \neq \emptyset$

Application Constraints



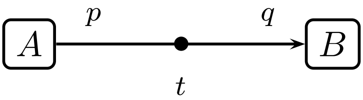
Graph Constraint: Data dependency

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

Modifier - User Constraint: Boolean dependency

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

User Constraint Examples



User Contraint: Buffer capacity restriction to k

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

Resource Constraint: Mutual exclusion of A and B

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:

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

$$\begin{tikzcd} \boxed{A} \arrow[rr, bend left, "A_i > B_j"] & & \boxed{B} \arrow[ll, bend left, "B_j > A_k"] \end{tikzcd}$$

$$\Rightarrow \; A_i > A_k$$

$\forall$ 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:

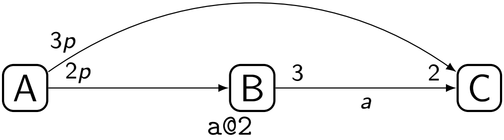
$$i > f(g(i))$$

Solution:

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

Constraint simplification

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



Constraints

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

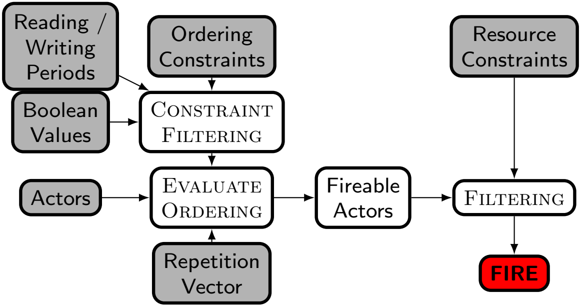
$$A_1 = 1$$

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

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

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

Run-time scheduler



Small overhead:

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

Conclusions

- Flexible constraint framework for BPDF 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

Outline

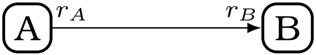
Throughput Calculation

Throughput of an SDF graph is the number of iterations that the graph finishes per time unit. To calculate:

- Conversion to HSDF and examination of the critical cycle
  - The cycle with the maximal cycle mean
- Simulating self-time execution and finding steady state execution [7]
  - Can be formulated using (max,+) algebra. [8]
- Both approaches do not support parameters

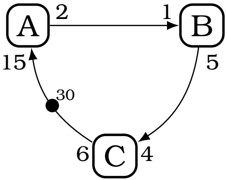
[7] Ghamarian, 2006, Throughput Analysis of Synchronous Data Flow Graphs
[8] Geilen, 2010, Synchronous Dataflow Scenarios

Throughput Calculation - Our approach



Nominal throughput of an actor: $T_{A_N} = \frac{1}{t_A}$
Maximum throughput of an actor:

$$T_B = \left(\min T_{B_N}, T_A \cdot K_{BA} \right) \quad where \quad K_{BA} = \frac{r_A}{r_B}$$



$$\#A \cdot T_A = \#B \cdot T_B$$
$$\#B \cdot T_B = \#C \cdot T_C$$
$$\#C \cdot T_C = \#A \cdot T_A$$
$$T_A \leq T_{A_N}$$
$$T_B \leq T_{B_N}$$
$$T_C \leq T_{C_N}$$