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 - SDF

An SDF graph

1 Lee and Messerschmitt 1987
Synchronous Data Flow - SDF

Rate (Amount of data)

Actor (Functional unit)

Edge (Communication link)

Initial tokens

In SDF all rates are fixed and known at compile time

---

1Lee and Messerschmitt 1987
Graph state: Data stored on its edges
Firing of actor $A$: Consumes 1 token
Firing of actor A: Produces 3 tokens

An SDF graph

Actor (Functional unit)

Rate (Amount of data)

Edge (Communication link)

Initial tokens

In SDF all rates are fixed and known at compile time

Graph state: Data stored on its edges

Firing of actor A:
Consumes 1 token

Firing of actor A:
Produces 3 tokens

SDF analysis:
Consistency

SDF analysis:
Boundedness

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

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 - Consistency

SDF analysis: Consistency

\[ #A \cdot 3 = #B \cdot 2 \]
\[ #B \cdot 1 = #B \cdot 3 \]

Initial tokens:

\[
\begin{bmatrix}
0 \\
0 \\
2
\end{bmatrix}
\]

Firing of actor A:
Consumes 1 token

\[
\begin{bmatrix}
0 \\
0 \\
2
\end{bmatrix}
\]

Firing of actor A:
Produces 3 tokens

\[
\begin{bmatrix}
3 \\
0 \\
1
\end{bmatrix}
\]

SDF analysis:
Liveness

There exists a schedule completing one iteration
Are there enough initial tokens?
If there exists, it can be repeated indefinitely and the graph is live
SDF analysis: Boundedness

There is no accumulation of tokens as the graph returns to its initial state.
**Synchronous Data Flow - Iteration**

**SDF analysis: Boundedness**

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

One Iteration

- **#A = 2**
- **#B = 3**
- **#C = 1**

- **Initial tokens:**
  - A: 0
  - B: 0
  - C: 2

- **After firing A:**
  - A: 3
  - B: 6
  - C: 4

- **After firing B:**
  - A: 2
  - B: 1
  - C: 2

- **After firing C:**
  - A: 0
  - B: 0
  - C: 1

- Final state:
  - A: 0
  - B: 0
  - C: 0

Graph state:

\[
\begin{bmatrix}
0 & 3 & 6 \\
0 & 0 & 0 \\
2 & 1 & 0 \\
\end{bmatrix}
\]

SDF analysis: Liveness

There exists a schedule completing one iteration or if there exists, it can be repeated indefinitely and the graph is live.
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*
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

VLD (A) -> SMB (B) -> MBB (C) -> MC (D) -> LOOP (G) -> IQIT (F) -> INTRA (E)

Motivation - Intra pipeline

VLD (A) -> SMB (B) -> MBB (C) -> MC (D) -> INTRA (E) -> IQIT (F) -> LOOP (G) -> MC (D)

Diagram:

- VLD (A)
- SMB (B)
- MBB (C)
- MC (D)
- LOOP (G)
- IQIT (F)
- INTRA (E)

Arrows indicate flow direction:
- pq to VLD (A)
- b to SMB (B)
- q to MBB (C)
- a to INTRA (E)
- b to MC (D)
- a to LOOP (G)
- a to IQIT (F)
- b to MC (D)
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 - BPDF

A BPDF graph

1 Bebelis, Fradet and Girault 2013
Boolean Parametric Data Flow - BPDF

Modifier of boolean $a$

Integer parameter $p$

Change period of boolean $a$

Boolean guard on BD

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 analysis: Consistency

#A \cdot p = #B

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

Parameteric solution of balance equations

\( [A^2 \ B^{2p} \ C^p \ D^{2p} \ E^{2p}] \)
BPDF analysis: Consistency

\[ A \cdot p = B \]

\[ C \]

\[ a: false \]

\[ D \]

\[ E \]

\[ C \] although disconnected still fires

Disconnected!
BPDF analysis: Consistency

There are boolean propagation links

A: false
**BPDF analysis:** Boundedness

In SDF, **consistency** suffices
BPDF analysis: Boundedness

What happens if the period of $a$ changes to 1?
BPDF analysis: Boundedness

a: true

B produces a token on BD
BPDF analysis: Boundedness

No guarantee it will be consumed!

a: false

Not all periods are safe and should be checked.
BPDF analysis: Boundedness

Region of boolean a and solutions

[B^{2p} C^p D^{2p} E^{2p}]
BPDF analysis: Boundedness

Boolean cannot change during a local iteration

Can be factorized by $f = p$ or 1

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

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 $X \in \mathcal{R}(b)$,

$$\exists k \in \mathbb{N}, \#X = k \cdot \frac{M(b)}{\alpha(b)}$$
A BPDF graph is **bounded** if

• it is **consistent** and

• all its boolean parameters satisfy the period safety criterion
**BPDF analysis:** Liveness

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

For the given graph, the BPDF analysis shows that the graph is live when a schedule of an iteration exists. The PSLC algorithm finds the schedule, and the result unfolds to the specified form.

Clustering cycles + PSLC
- Cluster B and C into Z with local schedule BC
- PSLC finds the schedule AZ
- Which unfolds to A(BC)2p

False cycles + Clustering + PSLC
- Cluster B and C into Z with conditional schedule
- A(if a then BC else CB)2p
- PSLC finds the schedule AZ
- Which unfolds to A(B2p

V.BEBELIS (INRIA)
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 analysis:** Liveness

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

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

Cluster B and C into Z with local schedule \(BC\)

PSLC finds the schedule \(AZ^{2p}\) which unfolds to \(A(BC)^{2p}\)
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
  \[A(\text{if } a \text{ then } BC \text{ else } CB)^{2p}\]
  PSLC finds the schedule \[AZ^{2p}\]
Related models - Integer Parameters

- **Parametric Synchronous Data Flow - PSDF**
  - 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**
  - 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** \(^5\)

---

\(^4\) Buck 1993
\(^5\) Buck 1995
BDF - Undecidability

- Both BDF and 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.
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

1. Data Flow Models of Computation

2. Scheduling
   - STHORM platform
   - Scheduling framework

3. Current work
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

**Rep. vector:** \[
\begin{bmatrix}
A^2 & B^6 & C^3
\end{bmatrix}
\]

---

\(A \rightarrow B \rightarrow C\)

---

\(^6\)S. Ha et al. 1991
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

1. Application
2. Ordering Constraints
3. Simplification & Liveness
4. User-defined Constraints
5. Resource Constraints
6. Scheduler
   - Evaluate Ordering
   - Filtering
7. ASAP Schedule
Scheduling constraints

- **Ordering Constraints:** Express the partial ordering of the firings
  \[ X_i > Y_{f(i)} \]

- **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

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

Modifier - User Constraint: Boolean dependency

\[ U_i > M_{f(i)} \quad \text{with} \quad f(i) = \pi_w \cdot \left\lfloor \frac{i - 1}{\pi_r} \right\rfloor + 1 \]
User Constraint: 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 $\{A\}$
Deadlock

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

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

$$A_i > B_j$$

$$B_j > A_k$$

$$\Rightarrow A_i > A_k$$

$$\forall$$ 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:

\[ i > f(g(i)) \iff i > \left[ \frac{q \cdot \left\lfloor \frac{p \cdot i - k}{q} \right\rfloor}{p} \right] \]
\[ \iff i > \frac{q \cdot (\frac{p \cdot i - k}{q} + 1)}{p} + 1 \]
\[ \iff i > i + \frac{q - k}{p} + 1 \]
\[ \iff k > p + q \]
\[ \iff k > p_{\text{max}} + q_{\text{max}} \]
Constraint simplification

\[
\begin{bmatrix}
A & B^{2p} & C^{3p}
\end{bmatrix}
\]

Constraints:

\[
\begin{align*}
B_i & > A\left\lceil \frac{i}{2p} \right\rceil \\
C_i & > B\left\lceil \frac{2i}{3} \right\rceil \\
C_i & > A\left\lceil \frac{i}{3p} \right\rceil \\
C_i & > B_2\left\lceil \frac{i}{3} \right\rceil - 1
\end{align*}
\]

\[
A_1 = 1 \\
B_i = \max(B_{i-1}, 1) \text{ for } i \in [1..2p] \\
B_i = i + 1 \\
C_i = \max(A_1, B_{\left\lceil \frac{2i}{3} \right\rceil}, B_2\left\lceil \frac{i}{3} \right\rceil - 1, C_{i-1}) \text{ for } i \in [1..3p] \\
C_i = i + 2
\]

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

1. Data Flow Models of Computation
2. Scheduling
3. Current work
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
  - Can be formulated using \((\text{max},+)\) algebra.
- 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: \( T_{AN} = \frac{1}{t_A} \)

Maximum throughput of an actor:

\[
T_B = \left( \min T_{BN}, T_A \cdot K_{BA} \right) \quad \text{where} \quad K_{BA} = \frac{r_A}{r_B}
\]