Statically Analyzable Programming Model for Dynamic Streaming

Peter Poplavko\textsuperscript{1,2}

joint work with: Pascal Fradet\textsuperscript{1}, Alain Girault\textsuperscript{1}, Ali Erdem Ozcan\textsuperscript{3}

\textsuperscript{1} INRIA Rhône-Alpes, \textsuperscript{2} CRI Grenoble, France
\textsuperscript{3} STMicroelectronics Inc., Ottawa, Canada

29 June 2011, Map2MPSoC
plan

• motivation

• basic model of computation

• status
  - proposed model of computation
  - tools

• summary
motivation

• the advent many-core SoCs:
  tens / hundreds of processors
  high-performance applications e.g. HDTV
  =$\Rightarrow A$ and $B$

$A$: thread-level parallelism
  - data parallelism (clone threads for different data)
  - functional parallelism (different threads in a pipeline)

$B$: point-to-point FIFO connections

$A + B = \text{dataflow}$
requirements

dataflow model of computation (MoC):
  expressive
    - data-dependent communication rates
    wide dynamic range

pure dataflow execution
  - fully distributed
  - where no central controllers intervene

compile-time verifiable for:
  - absence of deadlock / liveness
  - bounded memory requirements
state of the art

- static-rate and static-structure streaming
  - video sample rate conversion, noise reduction, FFT, ...
  - **Synchronous Dataflow – SDF** [E.A.Lee and D.G.Messerschmitt 1987]

- dynamic-rate and dynamic-structure streaming
  - MP3/MPEG4 codecs, video surveillance, interactive TV ...
  - **which model of computation (MoC) ???**

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPN</td>
<td>Kahn Process Networks</td>
<td>G.Kahn 1974</td>
</tr>
<tr>
<td>HDF</td>
<td>heterochronous</td>
<td>A.Girault et al 1999</td>
</tr>
<tr>
<td>PSDF</td>
<td>parametric</td>
<td>B.Bhattacharyya and S.S.Bhattacharyya 2001</td>
</tr>
<tr>
<td>SADF</td>
<td>scenario-aware</td>
<td>B.D.Theelen et al 2006</td>
</tr>
<tr>
<td>VRDF</td>
<td>variable-rate</td>
<td>M.H.Wiggers et al 2008</td>
</tr>
</tbody>
</table>

... no complete answer to our goals
starting point: SDF

SDF graph (initial state):

![SDF graph](image)

2 initial tokens in FIFO

one balance equation per edge:

\[ #_A \cdot 4 = #_B \cdot 3 \]

minimal solution = SDF iteration:

\[ #_A = 3; \quad #_B = 4; \]

after iteration → initial state

schedule 1:  A A A B B B B = A³ B⁴

schedule 2:  A B B A B A B = A B² (AB)²

a larger SDF:

![Larger SDF graph](image)

system of balance equations

a schedule:  A B⁴ C²
proposed MoC: SPDF

Schedulable Parametric Dataflow (SPDF)
- rates: parametric or constant
- a parameter is set by an actor
  - parameter changes every “period” actor executions
  - notation: set $\text{parameter}_{[\text{period}]}$

A \hspace{1cm} B \hspace{1cm} C \hspace{1cm} D

\begin{align*}
\text{set } q_{[1]} \\
\text{set } r_{[3]} \\
\text{set } r
\end{align*}

\[ \text{a schedule: } A (B^3 C^r D)^q \]

quasi-static schedule
**SPDF details**

- **parameter expressions**
  - polynomials with positive integer coefficients
  - Boolean expressions

- no fixed rule on which actors may set a given parameter
- the tools check inconsistencies in parameter communication

Diagram:

```
A
  set q_{[1]}
  set p_{[1]}

3q  1  r
B  set r_{[3]}

C
  1  3
   r

E
  1
  p+1

3p+3
D
```

8/16
implement parameter communication

static analysis:
- rate consistency
  - solutions of balance equations must exist
- parameter change safety
  - rates may change only at certain points
- liveness of the cycles

compute a quasi-static schedule

compile time!

rate consistency

do balance equations have a solution for any parameter value?

turn all directed edges into undirected edges

consider every cycle:

\[ \begin{align*}
A \quad r & \quad 3 & \quad C \quad 1 & \quad D \\
p+1 & \quad 1 & \quad E \quad 1 & \quad 3p+3 \\
\end{align*} \]

\[ \begin{align*}
& \swarrow 3 \cdot (p+1) \\
\end{align*} \]

**theorem:**

we have solutions

iff

the factors are balanced in every cycle
parameter change safety

check the correctness of the **periods** algorithm:

1. cover the graph by a hierarchy of subgraphs.
2. for every subgraph bottom-up:
   - solve local balance equations
     - #A, #B... - safe periods for setting parameters at A, B, ...

"regions of influence" of parameters: e.g. region r, region p
parameter change safety (2)

for example, region $r$

during an iteration of subgraph, rates should not change
Liveness of cyclic paths

Enough initial tokens

Sufficient condition:

\[ \exists k : m_k \geq i_k \cdot \#^c(A_k) \]

The above is easy to verify

- \( m_k \) - a compile-time constant
- \( i_k \), \( \#^c \) - expressed by Boolean functions and positive-coefficient polynomials
- Max values of parameters should be provided
video decoder example

(rates equal to 1 are omitted here)
the regions of parameters
summary

- dynamic streaming model of computation
  - schedulable at compile-time
  - preserves many advantages of SDF

future work

- memory minimization
- reconfiguration
- MPSoC mapping and scheduling
- verify the performance analytically
  - worst-case, average-case analysis
questions?