Fast and Optimal Throughout Evaluation of Cyclo-Static Dataflow

Bruno Bodin\textsuperscript{1}  Alix Munier-Kordon\textsuperscript{2}
Benoit Dupont de Dinechin \textsuperscript{3}

\textsuperscript{1}University of Edinburgh, United Kingdom
\textsuperscript{2}LIP6, University Pierre et Marie Curie, France
\textsuperscript{3}Kalray Corp, France

DAC’16, Austin, TX.
Context  Embedded systems and streaming applications

Computationally intense applications change fast

Heterogeneous hardwares are numerous
Streaming languages imply explicit parallelism and modularity
How to select which core for which task?
- Very Hard problem,
- dataflow compilation.
Each of these steps can require throughput evaluation method.
A set of processes \((\mathcal{T})\) communicating through channels \((\mathcal{A})\)
- Channels are unbound FIFO buffers with blocking read
- Tasks are divided in \(\varphi(t)\) phases
- \(in_a(k)\), the production rate of \(t_k\) the \(k^{th}\) phase of \(t\)
- \(out_a(k')\), the consumption rate of the \(k'^{th}\) phase of \(t'\)
- The initial quantity of token is \(M_0(a)\).
**Throughput**  
*As soon as possible scheduling*

- Task duration:
  - $d(A_1) = 3$, $d(A_2) = 1$  
  - $d(B_1) = 2$, $d(B_2) = 1$  
  - $d(B_3) = 2$, $d(C_1) = 1$
- No resource constraint

---

**Context**

- Throughput
- K-Periodic
- Conclusion
**Throughput Definition**

**Functional frequency**

$$Th_t^S = \lim_{n \to \infty} \frac{n}{S(t, n)}$$

with $S(t, n)$ the starting time of the first phase of $t$.

**Normalized period**

When a CSDFG has bounded memories, a balance exists between tasks frequency.

$$\Omega^S_G = \frac{N^G_t}{Th_t^S} \quad \forall t \in \mathcal{T}$$

---

**Example**

$Th_A = 6/21 = 2/7$, $\Omega^S_G = 21$
### Throughput  State of the art

**Exact methods ([GGS^+06, SGB08])**:

![Execution pattern, $Th_A = 6/21 = 2/7$](image)

- **Optimal** ✓
- **Exponential complexity** ✗

**Approximate methods ([BHMMK12, BKdD13])**:

![Period, $Th_A = 1/4$](image)

- **Polynomial** ✓
- **Lower bound** ✗
**K-Periodic**  Definition of K-periodic scheduling

- **ASAP (✓ Optimal ✗ Exponential):**
  
  \[
  \begin{array}{cccccccccccc}
  A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 \\
  B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 \\
  C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 \\
  \end{array}
  \]

- **Periodic (✓ Polynomial ✗ Non optimal):**
  
  \[
  \begin{array}{cccccccccccc}
  A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 \\
  B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 \\
  C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 \\
  \end{array}
  \]

- **K-periodic (✓ Flexible):**
  
  \[
  \begin{array}{cccccccccccc}
  A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 \\
  B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 \\
  C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 \\
  \end{array}
  \]

- **Context Throughput K-Periodic Conclusion**

- **K-periodicity vector, \( K = [1, 1, 2] \)
Property 1 - Optimal solution

Let \( G = (\mathcal{T}, \mathcal{A}) \) be a CSDFG, and considering a \( K \)-periodic schedule with \( K^G = N^G \) its periodicity vector. This schedule reaches the maximal throughput of \( G \).

**asap schedule:**

\[
\begin{array}{cccccccccccc}
A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 \\
B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 \\
C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 \\
\end{array}
\]

**Vector** \( K^G = N^G = [3, 3, 4] \):

\[
\begin{array}{cccccccccccc}
A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & A_1 \\
B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 & B_1 \\
C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 & C_1 \\
\end{array}
\]

Context Throughput K-Periodic Conclusion
Property 2 - Non linearity

The throughput does not necessarily increase while periodicity factors increase.

Vector $K^G = [1, 1, 2]$

Vector $K^G = [1, 1, 3]$
How to explore these vectors?

- Each node is a possible vector.
- The darker is the node, the faster to compute is the schedule.
- The bigger is the node, best is the solution.
- A white node is an optimal solution.
To compute the throughput of a K-periodic schedule, we build a dependency bi-graph between every required starting time (defined by the vector K). The MCR of this graph is the solution.

\[
MCR = \max_{c \in G} \frac{\sum_{e \in c} l(e)}{\sum_{e \in c} r(e)}
\]

With \(K=[1,1,1,1]\), MCR is reached by the circuit \(\{A_1, D_1, C_1\}\) and is equal to \(\frac{1+1+1}{\frac{1}{36} + \frac{1}{18} - \frac{1}{18}} = 108\).
Compute the **local repetition factors of the critical tasks** $(A,C,D)$ by ignoring other tasks.

\[
8 \times N_A = 24 \times N_D \\
4 \times N_A = 2 \times N_C \\
6 \times N_C = 36 \times N_D
\]

Then we **update the vector** $K$ such as factors of critical tasks are multiple to these local repetition factors. $K=[1,6,1,6]$. 

$N_A = 6$, $N_C = 6$, $N_D = 1$
We continue iteratively until the same set of task are critical.

Thus, we compute ...

\[
K = [1,1,1,1] \quad \Omega^S_G = 108 \quad c = [C,A,D] \\
K = [6,1,6,1] \quad \Omega^S_G = 96 \quad c = [A,B,C] \\
K = [6,2,6,1] \quad \Omega^S_G = 84 \quad c = [A,B,C,A,D,C] \\
K = [6,12,6,1] \quad \Omega^S_G = 78 \quad c \text{ is the same.}
\]
# K-Periodic Experimentations

<table>
<thead>
<tr>
<th>Application</th>
<th>Act.</th>
<th>Buff.</th>
<th>( \text{sum}(N_t^G) )</th>
<th>periodic</th>
<th>Kiter</th>
<th>symbolic exec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlackScholes</td>
<td>41</td>
<td>40</td>
<td>11895</td>
<td>100%</td>
<td>0.28ms</td>
<td>100% 22.43ms</td>
</tr>
<tr>
<td>Echo</td>
<td>240</td>
<td>703</td>
<td>802971540</td>
<td>100%</td>
<td>0.12ms</td>
<td>100% 37.57ms</td>
</tr>
<tr>
<td>JPEG2000</td>
<td>38</td>
<td>82</td>
<td>336024</td>
<td>100%</td>
<td>1.02ms</td>
<td>100% 4sec</td>
</tr>
<tr>
<td>Pdetect</td>
<td>58</td>
<td>76</td>
<td>3883200</td>
<td>100%</td>
<td>6.15ms</td>
<td>100% 117ms</td>
</tr>
<tr>
<td>H264 Enc.</td>
<td>665</td>
<td>3128</td>
<td>24094980</td>
<td>100%</td>
<td>3.83ms</td>
<td></td>
</tr>
<tr>
<td>BlackScholes</td>
<td>41</td>
<td>80</td>
<td>11895</td>
<td>98%</td>
<td>0.36ms</td>
<td>100% 4sec</td>
</tr>
<tr>
<td>Echo</td>
<td>240</td>
<td>1406</td>
<td>802971540</td>
<td>33%</td>
<td>0.14ms</td>
<td>100% 188sec</td>
</tr>
<tr>
<td>JPEG2000</td>
<td>38</td>
<td>164</td>
<td>336024</td>
<td>N/S</td>
<td>2.37ms</td>
<td>100% 4928sec</td>
</tr>
<tr>
<td>Pdetect</td>
<td>58</td>
<td>152</td>
<td>3883200</td>
<td>100%</td>
<td>10.98ms</td>
<td>100% &gt; 1d</td>
</tr>
<tr>
<td>H264 Enc.</td>
<td>665</td>
<td>6256</td>
<td>24094980</td>
<td>100%</td>
<td>6.76ms</td>
<td>100% &gt; 1d</td>
</tr>
<tr>
<td>graph1</td>
<td>90</td>
<td>617</td>
<td>752976</td>
<td>0.1%</td>
<td>3 ms</td>
<td>100% 646sec</td>
</tr>
<tr>
<td>graph2</td>
<td>70</td>
<td>473</td>
<td>2479863720</td>
<td>??%</td>
<td>4 ms</td>
<td></td>
</tr>
<tr>
<td>graph3</td>
<td>154</td>
<td>671</td>
<td>3705826224</td>
<td>??%</td>
<td>9 ms</td>
<td></td>
</tr>
<tr>
<td>graph4</td>
<td>2426</td>
<td>2900</td>
<td>615612</td>
<td>96%</td>
<td>218 ms</td>
<td></td>
</tr>
<tr>
<td>graph5</td>
<td>2767</td>
<td>4894</td>
<td>1874910</td>
<td>2%</td>
<td>600 ms</td>
<td></td>
</tr>
</tbody>
</table>

N/S : No periodic solution. ?? : unknown optimality.

Source code available : https://github.com/bbodin/kiter
We proposed an optimal scheduling method for CSDFG
- It is fast.
- But still not enough with pathological cases.

The complexity of this problem is unknown
- The K-periodic equations provided a new point of view
- Recent works are going to the same direction[GHKS15], we need to compare those techniques.

Kiter may be applied to Scenario-aware dataflows.
Thanks for your attention
Conclusion

References


[IEE]