Mapping to Parallel Hardware

Bruno Bodin

University of Edinburgh, United Kingdom
- ARM big-LITTLE (Heterogeneous),
- $4 \times$ ARM A15,
- $4 \times$ ARM A7,
- 2-16 cores GPU,
- 5 Watt.
Kalray MPPA256

- 256 VLIW cores,
- 16 clusters of 16 cores,
- Torus NoC,
- 10 Watts.
**Context - Problematic**

**Mapping problem**

- How to select which core for which task?
- Very Hard problem!
- Usually (automatically) solved for streaming apps.
- Ex. StreamIt, SigmaC
The SigmaC dataflow language [Goubier et al., 2011]

agent Reader(N) {
    interface {
        spec {/* N outputs */}
        void start() {
            /* f => out */
        }
    }
}

agent Writer(N) {
    interface {
        spec {/* N inputs */}
        void start() {
            /* in => f */
        }
    }
}

agent Worker {
    interface {
        spec {in[1]; out[1]}
        void start(in,out) {
            /* in => out */
        }
    }
}

subgraph root {
    map {
        agent inA = new Reader(N+1);
        agent outA = new Writer(N+1);
        agent w[N+1] = new Worker();
        for(int i=0;i<=N;i++){
            connect(inA.out[i],w[i].in);
            connect(w[i].out, outA.in[i]);
        }
    }
}

What is a dataflow model!
Context - Dataflow models

Kahn Process Networks [Kahn, 1974]

- A set of processes ($T$) communicating through channels ($A$)
- Channels are unbound FIFO buffers
- Reading is a blocking operation
- A deterministic model

Undecidable analysis problems

Kahn Networks don’t allow buffer sizing [Buck et Lee, 1993].
**Context - Dataflow models**

**Synchronous Dataflow [Lee et Messerschmitt, 1987]**

\[ I_a \text{ is the number of tokens produced,} \]
\[ O_a \text{ is the number of tokens consumed,} \]
\[ \text{The initial quantity of tokens is } M_0(a). \]
\[ \text{Synchronous because } N_t^G \times I_a = N_{t'}^G \times O_a \forall a \in A \]

**A static model**

It is possible to study its behavior at compile-time: this model is static. Several fundamental problems become decidable.
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 $t'$. 
Context - Solution

The dataflow compilation

Fission and fusion
Buffer sizing

Throughput evaluation
Buffer sizing
Mapping & Routing

MPPA™-256ES
HFCBGA 1156 V1 00
F1 A11 F12 XXXXXX.XXX-X
TWN TF YY WW
Liveness -

Liveness and Consistency

Liveness

A dataflow is alive, if there exists a task execution sequence which can be repeated infinitely often.

Consistency

A dataflow is consistent if there exists a task execution sequence which can be repeated infinitely often with bounded memory constraint.
Liveness -

Liveness and Consistency - Example

Example

\[ S = [T_i, T_j, T_k, T_k, T_i] \]

3: Alive and consistent.
**Throughput - Problem definition**

As soon as possible scheduling

- Task duration (WCET):
  - $d(A_1) = 3$, $d(A_2) = 1$
  - $d(B_1) = 2$, $d(B_2) = 1$
  - $d(B_3) = 2$, $d(C_1) = 1$

- No resource constraint

---

<table>
<thead>
<tr>
<th>Transient state</th>
<th>Steady state</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$ $A_2$</td>
<td>$A_1$ $A_2$</td>
</tr>
<tr>
<td>$B_1$ $B_2$ $B_3$ $B_1$ $B_2$</td>
<td>$B_1$ $B_2$ $B_3$ $B_1$ $B_2$</td>
</tr>
<tr>
<td>$C_1$ $C_1$</td>
<td>$C_1$ $C_1$</td>
</tr>
</tbody>
</table>
**Throughput - Problem definition**

**Normalized period**

Functional frequency: \( Th_t^S = \lim_{n \to \infty} \frac{n}{S(t, n)} \).

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

\[ \Omega_G^S = \frac{N_t^G}{Th_t^S}, \quad \forall t \in T \]

---

\[ Th_A = \frac{6}{21} = \frac{2}{7}, \quad \Omega_G^S = 21 \]
### Throughput - State of the art

#### Existing methods

**Exact methods ([Ghamarian et al., 2006, Stuijk et al., 2008]):**

![Diagram of execution pattern]

Execution pattern, $Th_A = 6/21 = 2/7$

- **Optimal**
- **Exponential complexity**

**Approximate methods ([Benabid et al., 2012, Bodin et al., 2013]):**

![Diagram of execution pattern]

Period, $Th_A = 1/4$

- **Polynomial**
- **Lower bound**

---

Context | Liveness | Throughput | Memory | Mapping
Throughput - Periodic scheduling of CSDFG

Definition of precedence relations
Definition (Valid schedule)

If a precedence relation exists between two executions \( \langle t_k, n \rangle \) and \( \langle t'_k, n' \rangle \), a valid schedule must check

\[
S\langle t'_k, n' \rangle \geq S\langle t_k, n \rangle + d(t_k).
\]
**Throughput - Periodic scheduling of CSDFG**

*A precedence relation between two executions*

**Lemma (existence condition)**

Let a buffer $a = (t, t')$. A precedence relation exists between $\langle t_k, n \rangle$ and $\langle t'_k, n' \rangle$ if and only if

$$in_a(k) > M_0(a) + I_a\langle t_k, n \rangle - O_a\langle t'_k, n' \rangle \geq \max\{0, in_a(k) - out_a(k')\}.$$
Lemma (Existence condition between two phases)

A buffer \( a = (t, t') \) induce a precedence relation between \( \langle t_k, n \rangle \) and \( \langle t'_k, n' \rangle \) if and only if

\[
\alpha_{a}^{\text{min}}(k, k') \leq \alpha(n, n') \leq \alpha_{a}^{\text{max}}(k, k').
\]
Throughput - Periodic scheduling of CSDFG

Periodic constraints

We finally get the following constraint:

Theorem (Periodic constraints)

Precedence relations induced by a buffer \( a = (t, t') \) are checked by a periodic schedule \( S \) if and only if

\[
S\langle t'_{k'}, 1 \rangle - S\langle t_k, 1 \rangle \geq d(t_k) + \Omega^S_G \times \frac{\alpha_{\text{max}}(k, k')}{N^G_t \times i_a}
\]

\( \forall (k, k') \in \{1, \cdots, \varphi(t)\} \times \{1, \cdots, \varphi(t')\} \) with \( \alpha_{\text{min}}(k, k') \leq \alpha_{\text{max}}(k, k') \).
Throughput - Periodic scheduling of CSDFG

Computation of a periodic schedule

These constraints can be used to compute a periodic schedule with maximal throughput:

Minimize Ω^S with

\[
\forall a = (t, t') \in A, \forall (k, k') \in \mathcal{Y}(a),
\]

\[
S\langle t'_{k'}, 1 \rangle - S\langle t_k, 1 \rangle \geq d(t_k) + \Omega^S_G \times \frac{\alpha^\text{max}_a(k, k')}{i_a \times N_t^G}
\]

\[
\forall t \in T, \forall k \in \{1, \cdots, \varphi(t)\}, S\langle t_k, 1 \rangle \in \mathbb{R}^+\]

\[
\Omega^S_G \in \mathbb{R}^+ - \{0\}
\]

✓ A linear program
✓ Polynomial (Maximum Cycle Ratio Problem)
✓ Maximal periodic throughput
✗ Upper bound of the maximal throughput
Memory - Definition

Buffer sizing

- Buffer sizing impact liveness
- But also performances

![Diagram of buffer sizing with tokens]

3 tokens max:

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t1</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
</tr>
</tbody>
</table>

4 tokens max:

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
<td>t1</td>
</tr>
</tbody>
</table>
Memory - State of the art

Existing solutions

- Exact method [Stuijk et al., 2008]
  - Exploring a (combinatorial) solution space
  - Maximal throughput evaluation (ASAP)
  - Too long for existing cases

- Approximative methods [Benazouz et Munier-Kordon, 2013]
  - A maximal throughput is fixed
  - Only periodic schedules are considered
  - Phases execution pattern is fixed.

\[
\begin{align*}
\text{Phases fixées} & \quad \langle t_1, 1 \rangle \quad \langle t_1, 2 \rangle \quad \langle t_1, 3 \rangle \\
\text{période d’exécution} & \quad \langle t_2, 1 \rangle \quad \langle t_2, 2 \rangle \quad \langle t_2, 3 \rangle
\end{align*}
\]
We adapt the linear formulation:

\[
\text{Minimize } \sum_{a \in Fb(A)} M_0(a) \text{ with }
\]

\[
\begin{align*}
\forall a = (t, t') \in A, \forall (k, k') \in \mathcal{Y}(a), \\
S\langle t', k' \rangle - S\langle t, k \rangle &\geq d(t) + \Omega^S_G \times \frac{\alpha^\max_a(k, k')}{i_a \times N^G_t} \\
\forall a' = (t, t') \in Fb(A), \forall (k, k') \in \mathcal{Y}(a'), \\
S\langle t', k' \rangle - S\langle t, k \rangle &\geq d(t) + \Omega^S_G \times \frac{\alpha^\max_{a'}(k, k')}{i_{a'} \times N^G_t} \\
\forall t \in T, \forall k \in \{1, \ldots, \varphi(t)\}, S\langle t, k \rangle &\in \mathbb{R}^+ 
\end{align*}
\]
The buffer sizes are known, we need now to select which processor for which task. This is hard, the problem is **splitted into sub-problems**.
Mapping - Partitioning

Partitioning

Producing task groups which will be executed on the same core. Usually solved off-line using meta-heuristics (genetic, taboo, machine learning, etc.).
This is the actual association of task groups with computation units. This can be solved at compile-time, but also at run-time using task migration.
For some platforms this is required to determine the actual communication in the NoC. This may be optimally solved with a deterministic NoC.
Conclusion

- Mapping is a hard problem
- Depends on how parallelism and tasks represented
- Wide open research area
- Large research interest at ICSA Edinburgh


