Formal architectural modelling

Opportunities and challenges

Julien Schmaltz

joint work with
Perry van Wesel and Alexander Fedotov
Architectural model example: TornadoNoC (1)
Additional mechanisms:

- **Spillover** – Travel an additional lap to avoid blocking other traffic.

- **Sequence numbering** – Limit the number of laps for one packet.

- **Injection constraint** – Limit number of packets in a ring.

- **Inflation mechanism** - Avoid protocol deadlock and starvation.

Architectural model example: TornadoNoC (II)
MaDL basic principles - Originates from Intel’s xMAS

Notion of a “dead” channel: F (c.irdy & G ! c.trdy)
MaDL Design and Verification Environment

```plaintext
param int QSIZE;
const pkt;
chan src := Source(pkt);
chan one, two := Fork(src);
chan long := Queue(QSIZE,Queue(QSIZE, one));
chan short := Queue(QSIZE, two);
Sink(CtrlJoin(long,short));
```

https://github.com/MaDL-DVT/madl-dvt
MaDL Ring Extension - GuardQueue primitive

Modelling the injection constraint with the GuardQueue: ‘Only inject if the local buffer is empty’

\[
\begin{align*}
  i_b.\text{trdy} &= \neg \text{Full}(gq) \land \text{sel} == 0 \\
  i_g.\text{trdy} &= o.\text{trdy} \land \text{Empty}(gq) \land \text{sel} == 1 \\
  o.\text{irdy} &= (\text{sel} == 0 \land \neg \text{Empty}(gq)) \lor (\text{Empty}(gq) \land i_g.\text{irdy} \land \text{sel} == 1)
\end{align*}
\]

\[
\begin{align*}
  \text{block}(i_b, c) &= \text{Full}(gq) \land \text{block}(o, gq.\text{head}) \\
  \text{block}(i_g, c) &= \text{block}(o, c) \lor (\text{sel} == 0 \land \text{Full}(gq) \land \text{block}(o, gq.\text{head})) \\
  &\quad \lor \text{never empty}(gq) \\
  \text{idle}(o, c) &= \left(\text{idle}(i_b, c) \land \text{idle}(i_g, c)\right) \lor \left(\text{idle}(i_b, c) \land \text{never empty}(gq)\right) \\
  &\quad \lor \exists_{c' \in C \setminus c} \text{block}(o, c')
\end{align*}
\]
Ring $R = (D,X,I,O)$
- $D$: data colours
- $X$: Channels
- $I$: Inputs
- $O$: Outputs

Ring invariant for ring $R$:

$$\text{occ}(R) \leq \text{cap}(R) - x$$

where $x$ is the smallest capacity over all guard queues in the ring.

$$(\{\text{red}\}, \{\star \star \star \}, \{R_1_{-}\text{Inject}\}, \{R_0_{-}\text{Out}\})$$

$$(\{\text{blue}\}, \{\star \star \star \}, \{R_0_{-}\text{Inject}\}, \{R_1_{-}\text{Out}\})$$

$$(\{\text{red,blue}\}, \{\star \star \star \star \}, \{R_0_{-}\text{Inject}, R_1_{-}\text{Inject}\}, \{R_0_{-}\text{Out}, R_1_{-}\text{Out}\})$$
MaDL model of the TornadoNoC blocks
MaDL model and analysis results

<table>
<thead>
<tr>
<th>Layout</th>
<th>#Components</th>
<th>Rings</th>
<th>Deadlock</th>
<th>Verification time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 × 1</td>
<td>98</td>
<td>1</td>
<td>No</td>
<td>1.4s</td>
</tr>
<tr>
<td>2 × 2</td>
<td>196</td>
<td>4</td>
<td>No</td>
<td>2.8s</td>
</tr>
<tr>
<td>4 × 2</td>
<td>392</td>
<td>6</td>
<td>No</td>
<td>8.0s</td>
</tr>
<tr>
<td>4 × 4</td>
<td>784</td>
<td>8</td>
<td>No</td>
<td>59.7s</td>
</tr>
<tr>
<td>4 × 8</td>
<td>1568</td>
<td>12</td>
<td>No</td>
<td>970s</td>
</tr>
<tr>
<td>8 × 8</td>
<td>3136</td>
<td>16</td>
<td>No</td>
<td>3691s</td>
</tr>
</tbody>
</table>

Individual MaDL steps

Time required (seconds)

- Reading
- Livelocks
- Rings
- Invariants
- Full Queues
- SMT
Conclusion:
• Ring networks can now be analysed without finding the trivial all-queues-are-full deadlock.
• TornadoNoC contains a potential deadlock when using credit-flow control.
• The MaDL GuardQueue avoids this deadlock.

Future Work
- More detailed model of TornadoNoC, including Inflation Mechanism
- Proof of other properties like message ordering, starvation and livelock freedom, etc.
Architectural modelling and analysis - Reflection

specification

Architectural Model (AM)

Early modelling / prototyping

RTL
Architectural modelling and analysis - Reflection

Architectural Model (AM)

Specification

RTL

(automatic) abstractions
Architectural modelling and analysis - Reflection

Architectural Model (AM)

Specification

RTL

Model refinements
Architectural modelling and analysis - Reflection

Model Driven Document Generation

Model Driven Code Generation
Thanks!