Static Scheduling for Time-Predictable Networks-on-Chip

Embedded Systems Engineering Section
Institut for Informatik og Matematisk Modellering
Danmarks Tekniske Universitet

Florian Brandner

The work presented here is supported by the T-CREST project.
Real-Time Systems

Strict timing guarantees

- Critical tasks have to be completed in time
Real-Time Systems

Strict timing **guarantees**

- Critical tasks have to be completed in time
- Bound worst-case execution time (WCET)
Real-Time Systems

Strict timing guarantees

- Critical tasks have to be completed in time
- Bound worst-case execution time (WCET)
What Impacts Worst Case Behavior?

- Data structures
- Algorithms
- Input data
What Impacts Worst Case Behavior?

**Program Structure**
- Data structures
- Algorithms
- Input data

**Machine-Level Code**
- Compiler
- Assembly code
- Code generation tools
What Impacts Worst Case Behavior?

- Program Structure
  - Data structures
  - Algorithms
  - Input data

- Machine-Level Code
  - Compiler
  - Assembly code
  - Code generation tools

- Execution Platform
  - Processor state
  - Shared resources
  - Clock frequency
How Optimize the Worst Case Behavior?

Original System

WCET-unaware Opt.

WCET-aware Opt.

WCET-driven Opt.
How Optimize the Worst Case Behavior?
How Optimize the Worst Case Behavior?

Original System

WCET-unaware Opt.

WCET-aware Opt.

WCET-driven Opt.
### How Optimize the Worst Case Behavior?

<table>
<thead>
<tr>
<th></th>
<th>BCET</th>
<th>AVG</th>
<th>WCET</th>
<th>WCET Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution Time</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Original System</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WCET-unaware Opt.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WCET-aware Opt.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WCET-driven Opt.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| **Execution Time**  |      |     |      |            |
| **Original System** |      |     |      |            |
| **WCET-unaware Opt.** |      |     |      |            |
| **WCET-aware Opt.**  |      |     |      |            |
| **WCET-driven Opt.** |      |     |      |            |
The T-CREST Platform

Design of a time-predictable platform

• Simplify calculation of WCET bounds
The T-CREST Platform

Design of a time-predictable platform

- Simplify calculation of WCET bounds
- Explore WCET-driven system design and optimization
The T-CREST Platform

Design of a time-predictable platform
- Simplify calculation of WCET bounds
- Explore WCET-driven system design and optimization

- Project scope
  - Processor / Chip-multi-processor (DTU)
  - Memory hierarchy
  - Network-on-Chip (DTU)
  - Compiler (DTU, Vienna UT)
  - WCET analysis
The T-CREST Platform

Design of a time-predictable platform

- Simplify calculation of WCET bounds
- Explore WCET-driven system design and optimization

- Project scope
  - Processor / Chip-multi-processor (DTU)
  - Memory hierarchy
  - **Network-on-Chip (DTU)**
  - Compiler (DTU, Vienna UT)
  - WCET analysis

- Project mantra
  Evaluate everything with regard to WCET analyzability.
Networks-on-Chip

What is a Network-on-Chip (NoC)?

- Nodes: processors, specialized accelerators, memories, ...
- Nodes connected to routers
- Routers interconnected by links
Networks-on-Chip

What is a Network-on-Chip (NoC)?

- Nodes: processors, specialized accelerators, memories, ...
- Nodes connected to routers
- Routers interconnected by links
- Provide means to exchange messages between nodes
Network Topologies

Define structure and layout of nodes, routers, and links:

Mesh

Torus

Bidirectional Torus (intertwined)
Time-predictable Networks-on-Chip

Why is there a problem in real-time systems?

- The NoC is shared by all nodes
- Plenty of interaction between nodes through the network
Time-predictable Networks-on-Chip

Why is there a problem in real-time systems?

• The NoC is shared by all nodes
• Plenty of interaction between nodes through the network
• WCET analysis for one task depends on
  • Nodes involved in the completion of the task
  • Message delivery times between involved nodes
Time-predictable Networks-on-Chip

Why is there a problem in real-time systems?

- The NoC is shared by all nodes
- Plenty of interaction between nodes through the network
- WCET analysis for one task depends on
  - Nodes involved in the completion of the task
  - Message delivery times between involved nodes

⇒ All nodes of the NoC have to be analyzed!!
Time-predictable Networks-on-Chip

Why is there a problem in real-time systems?

- The NoC is shared by all nodes
- Plenty of interaction between nodes through the network
- WCET analysis for one task depends on
  - Nodes involved in the completion of the task
  - Message delivery times between involved nodes

⇒ All nodes of the NoC have to be analyzed!!

Clear separation of independent tasks w.r.t. the NoC needed.
Make it time-predictable ...

Two options

1. Guarantee throughput by buffering
2. Guarantee exclusive access by arbitration
Make it time-predictable ...

Two options

1. Guarantee throughput by buffering
2. Guarantee exclusive access by arbitration

Buffers:

- Incur hardware overhead
- How determine buffer sizes?
- How determine bandwidth per router/link?
- Requirements are application-specific
Make it time-predictable ...

Two options

1. Guarantee throughput by buffering
2. Guarantee exclusive access by arbitration

Buffers:

- Incur hardware overhead
- How determine buffer sizes?
- How determine bandwidth per router/link?
- Requirements are application-specific
- Not suited for a generic, cost-effective platform
The S4 Design

Minimalistic time-multiplexed NoC

• Fixed, static schedules
  providing periodic, all-to-all communication
The S4 Design

Minimalistic time-multiplexed NoC

• Fixed, static schedules
  providing periodic, all-to-all communication
• Time-multiplexing of routers and links
The S4 Design

Minimalistic time-multiplexed NoC

• Fixed, static schedules
  providing periodic, all-to-all communication
• Time-multiplexing of routers and links
• Guarantees
  • Fixed, well-known timing
  • Separation of independent tasks
  • Analyzability of worst-case behavior
The S4 Design

Minimalistic time-multiplexed NoC

- Fixed, static schedules providing periodic, all-to-all communication
- Time-multiplexing of routers and links
- Guarantees
  - Fixed, well-known timing
  - Separation of independent tasks
  - Analyzability of worst-case behavior

- Minimal hardware requirements
  - Routers are simple multiplexers
  - Static schedule tables at routers
The S4 Router

- A multiplexer and a register per out-going link
  - L connects the router to its node
  - N to the north neighbor
  - ...

- One in-coming/out-going link per direction
- Word-sized link width (e.g., 16 bits)
- Schedule table (ST) controls multiplexers
Schedule Construction

Some assumptions

• Every node sends one message to every other node
• Every node receives one message from every other node
• A node can each send and receives one message
Schedule Construction

Some assumptions

- Every node sends one message to every other node
- Every node receives one message from every other node
- A node can each send and receives one message
- Message = packet = flit = phit = 1 data word
Schedule Construction

Some assumptions

• Every node sends one message to every other node
• Every node receives one message from every other node
• A node can each send and receives one message
• Message = packet = flit = phit = 1 data word
• A hop from one router to the next takes one cycle
Schedule Construction

Some assumptions

• Every node sends one message to every other node
• Every node receives one message from every other node
• A node can each send and receives one message
• Message = packet = flit = phit = 1 data word
• A hop from one router to the next takes one cycle

• Network given as a graph \( G = (N \cup R, E) \)
  \( N \ldots \text{nodes}, \ R \ldots \text{router}, \ E \ldots \text{links} \)
Schedule Construction (2)

Solve a multi-commodity flow problem over time:

- Commodities correspond to messages
- Minimize time to deliver all commodities/messages
Schedule Construction (2)

Solve a multi-commodity flow problem over time:

- Commodities correspond to messages
- Minimize time to deliver all commodities/messages
- Modeled as Integer Linear Program (ILP)
  - Given a network $G$ construct a time-expanded network $G^T$
  - $G^T$ consists of copies of $G$: $g_1, \ldots, g_T$
  - Edges lead from some copy $g_i$ to $g_{i+1}$
  - Solve a standard flow problem on $G^T$
Example: Time-Expanded $3 \times 3$ Torus

The time-expanded network represents the state of the NoC on all time instants during the schedule construction.
Structure of the Linear Program

Variables

• \( \ell_{l,n}^t \): Use link \( l \) to send a message to node \( n \) at time instant \( t \)
Structure of the Linear Program

Variables

- $\ell_{l,n}^t$: Use link $l$ to send a message to node $n$ at time instant $t$

Constraints

- $\sum_{n \in N} \ell_{l,c}^t = 1$
  
  Use every link to send at most one message per time instant.
Structure of the Linear Program

Variables

- $\ell_{l,n}^t$: Use link $l$ to send a message to node $n$ at time instant $t$

Constraints

- $\sum_{n \in N} \ell_{l,c}^t = 1$
  Use every link to send at most one message per time instant.

- $\sum_{i \in \text{In}(r,t)} \ell_{i,n}^t - \sum_{o \in \text{Out}(r,t+1)} \ell_{o,n}^{t+1} = 0$
  All messages flowing into a router have to flow out again.
Structure of the Linear Program

Variables

- $\ell_{l,n}^t$: Use link $l$ to send a message to node $n$ at time instant $t$

Constraints

- $\sum_{n \in N} \ell_{l,n}^t = 1$
  Use every link to send at most one message per time instant.

- $\sum_{i \in \text{In}(r,t)} \ell_{i,n}^t \sum_{o \in \text{Out}(r,t+1)} \ell_{o,n}^{t+1} = 0$
  All messages flowing into a router have to flow out again.

- $\sum_{l \in \text{In}(n,t)} \ell_{l,n}^t = |N| - 1$
  Receive a message from every other node.
Structure of the Linear Program

Variables

- $\ell_{l,n}^t$: Use link $l$ to send a message to node $n$ at time instant $t$

Constraints

- $\sum_{n \in N} \ell_{l,c}^t = 1$
  Use every link to send at most one message per time instant.

- $\sum_{i \in \text{in}(r,t)} \ell_{i,n}^t \cdot n \in N - \sum_{o \in \text{out}(r,t+1)} \ell_{o,n}^{t+1} = 0$
  All messages flowing into a router have to flow out again.

- $\sum_{l \in \text{in}(n,t)} \ell_{l,n}^t = |N| - 1$
  Receive a message from every other node.

- $\sum_{l \in \text{out}(s,t)} \ell_{l,d}^t = 1$
  Send once from source node $s$ to destination node $d$. 
Solving the Linear Program

Using a generic ILP solver

- CPLEX 12.3 academic
- DTU’s hms2 server
  8 Quad-Core AMD Opteron 8356 (32 cores), 256 GB RAM
Solving the Linear Program

Using a generic ILP solver

- CPLEX 12.3 academic
- DTU's hms2 server
  - 8 Quad-Core AMD Opteron 8356 (32 cores), 256 GB RAM
- Network sizes of up to 25 nodes feasible
  (no initial solution, generous upper bound for schedule length)
  - Topologies: mesh, torus, bidir. torus (also: tree, fat-tree)
Solving the Linear Program

Using a generic ILP solver

- CPLEX 12.3 academic
- DTU’s hms2 server
  - 8 Quad-Core AMD Opteron 8356 (32 cores), 256 GB RAM
- Network sizes of up to 25 nodes feasible
  - (no initial solution, generous upper bound for schedule length)
    - Topologies: mesh, torus, bidir. torus (also: tree, fat-tree)
- Considerable improvements possible
## Some results

<table>
<thead>
<tr>
<th>Topology</th>
<th>Nodes</th>
<th>Routers</th>
<th>Links</th>
<th>Schedule</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh</td>
<td>4</td>
<td>4</td>
<td>8+8</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td>24+18</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>48+32</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>25</td>
<td>80+50</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Torus</td>
<td>4</td>
<td>4</td>
<td>8+8</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td>18+18</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>32+32</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>25</td>
<td>50+50</td>
<td></td>
<td>54*</td>
</tr>
<tr>
<td>Bidir. Torus</td>
<td>4</td>
<td>4</td>
<td>16+8</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td>36+18</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>64+32</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>25</td>
<td>100+50</td>
<td></td>
<td>27</td>
</tr>
</tbody>
</table>

* still running ... (lower bound from CPLEX: 52)
Dealing with Larger Networks

Observations from ILP solutions

• Paths taken for message routes
  • One straight horizontal or vertical segment, or
  • One horizontal and one vertical segment (X-Y, Y-X)
  • No detours, shortest routes only
Dealing with Larger Networks

Observations from ILP solutions

- Paths taken for message routes
  - One straight horizontal or vertical segment, or
  - One horizontal and one vertical segment (X-Y, Y-X)
  - No detours, shortest routes only

- We can exploit regularities of network topologies
  - Construct symmetric schedules
  - Synchronize all routers
  - Use the same schedule table everywhere
Example: Partial, Optimal Solution for the $3 \times 3$-Torus

This schedule cannot be replicated everywhere, due to conflicts, e.g., at cycle 2.
Example: Symmetric Solution for the $3 \times 3$-Torus

This schedule can be replicated everywhere.
A Heuristic Algorithm

Some assumptions

- Schedule relative to **one** reference node, then replicate
  - Restrict paths considered as routes
  - Only shortest routes from reference node
  - Example: only X-Y and Y-X routes
A Heuristic Algorithm

Some assumptions

• Schedule relative to **one** reference node, then replicate
  • Restrict paths considered as routes
  • Only shortest routes from reference node
  • Example: only X-Y and Y-X routes

• The NoC topology allows for symmetric schedules
  • **Yes:** torus, bidirectional torus, . . .
  • So-so: 2D-mesh
  • **No:** tree, fat-tree, star, . . .
A Heuristic Algorithm

Some assumptions

- Schedule relative to **one** reference node, then replicate
  - Restrict paths considered as routes
  - Only shortest routes from reference node
  - Example: only X-Y and Y-X routes

- The NoC topology allows for symmetric schedules
  - **Yes**: torus, bidirectional torus, . . .
  - So-so: 2D-mesh
  - **No**: tree, fat-tree, star, . . .

- As before
  - all-to-all communication
  - single-cycle hops
  - Message = 1 flit = 1 word
A Heuristic Algorithm (2)

1. Compute set of candidate routes
   - Set of routes from the reference node to all other nodes
   - Example: shortest, X-Y and Y-X routes

2. Select a good candidate
   - Example: a longest remaining route

3. Schedule the candidate route
   - Avoid conflicts with already scheduled routes
   - Example: schedule as early as possible

4. Remove all equivalent candidate routes

5. Repeat step 2-4 until no candidate route remains
Routes and Conflicts

- Represent routes as strings over an alphabet
  - Each symbol represents the direction of the next hop
  - Example: eesss – two hops to the east, three to the south

- Conflict when hops at the same time have identical symbols
  - invalid: eess
  - valid: eesssww

- Routes are equivalent when they lead to the same target node
  - Example: eesss ≡ sssee
Routes and Conflicts

• Represent routes as strings over an alphabet
  • Each symbol represents the direction of the next hop
  • Example: eesss – two hops to the east, three to the south

• Conflict when hops at the same time have identical symbols
  invalid: eesss
  valid: eesss
  wwww
  wwww

• Routes are equivalent when they lead to the same target node
  Example:
  eesss ≡ sssee
Routes and Conflicts

• Represent routes as strings over an alphabet
  • Each symbol represents the direction of the next hop
  • Example: eesss – two hops to the east, three to the south

• Conflict when hops at the same time have identical symbols
  invalid: eesss
  valid: eesss

• Routes are equivalent when they lead to the same target node
  Example: eesss ≡ sssee
Experiments

Several heuristic configurations

• Implemented in C++ (relatively untuned)
Experiments

Several heuristic configurations

• Implemented in C++ (relatively untuned)

• This netbook
  1 Dual-Core AMD Fusion E-350 (1 core), 2 GB RAM
Experiments

Several heuristic configurations

• Implemented in C++ (relatively untuned)

• This netbook
  1 Dual-Core AMD Fusion E-350 (1 core), 2 GB RAM

• Network size up to 900 nodes
  • Network topologies: torus, bidirectional torus (also: mesh)
Experiments

Several heuristic configurations
  • Implemented in C++ (relatively untuned)
  • This netbook
    1 Dual-Core AMD Fusion E-350 (1 core), 2 GB RAM
  • Network size up to 900 nodes
    • Network topologies: torus, bidirectional torus (also: mesh)
  • Compare schedule lengths and execution times
    • Theoretical bounds: network capacity, bisection bandwidth
    • Optimal results (as far as available)
Configurations

Candidate selection

- Sht: Shortest routes first
- Rnd: Select random route
- Lng: Longest routes first
- Cnfl: Longest routes first, avoid conflict with last candidate
Configurations

Candidate selection
- Sht: Shortest routes first
- Rnd: Select random route
- Lng: Longest routes first
- Cnfl: Longest routes first, avoid conflict with last candidate

Scheduling
- Schedule as early as possible
- Naive checking for conflicts
Schedule Lengths – Torus

- Heuristic is within 15-20% of theoretical lower bound.
Heuristic is within 15-20% of theoretical lower bound.
Schedule Lengths – Bidirectional Torus

![Graph showing schedule lengths on a bidirectional torus. The graph includes lines for Sht, Rnd, Lng, Cnfl, and Bound, with nodes on the x-axis and cycles on the y-axis. The graph shows that the schedule lengths are again within 15-20% of the theoretical lower bound.](image-url)
Again within 15-20% of theoretical lower bound.
# Heuristic vs. Optimal Solutions

<table>
<thead>
<tr>
<th>Topology</th>
<th>Nodes</th>
<th>Optimal</th>
<th>Heuristic</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torus</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>11</td>
<td>11</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>26</td>
<td>27</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>54*</td>
<td>56</td>
<td>1.04</td>
</tr>
<tr>
<td>Bidir. Torus</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>18</td>
<td>19</td>
<td>1.06</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>27</td>
<td>28</td>
<td>1.04</td>
</tr>
</tbody>
</table>

* still: still running ...
Execution Time – Bidirectional Torus

Polynomial curve, can clearly be improved by more efficient tracking of conflicts.
Execution Time – Bidirectional Torus

Polynomial curve, can clearly be improved by more efficient tracking of conflicts.
Conclusion

S4 network-on-chip

- Minimal hardware requirements NoC
- Static schedules at routers
Conclusion

S4 network-on-chip

- Minimal hardware requirements NoC
- Static schedules at routers

Optimal scheduling

- Integer linear programming infeasible for large networks
- NP-hard in general
Conclusion

S4 network-on-chip
- Minimal hardware requirements NoC
- Static schedules at routers

Optimal scheduling
- Integer linear programming infeasible for large networks
- NP-hard in general

Heuristic scheduling
- Exploit regularity of NoC topology
- Simple, yet efficient
- 15-20% from lower bounds
- Even closer to optimal solutions
Future Work

Optimal scheduling

- Start from heuristic solution
- Prune search space
- Restrict freedom of routes

Conjecture: Optimal symmetric schedules are globally optimal.
Future Work

Optimal scheduling

- Start from heuristic solution
- Prune search space
- Restrict freedom of routes

Conjecture: Optimal symmetric schedules are globally optimal.

Heuristic scheduling

- Explore other network topologies
- Improve complexity of algorithm (tracking of conflicts)
- Relax some assumptions
  - worm-hole routing, ...