Static Scheduling for Time-Predictable Networks-on-Chip

Florian Brandner
Unité d'Informatique et d'Ing. des Systèmes
ENSTA-ParisTech

J. Sparsø, M. Schoeberl, E. Kasapaki
Embedded Systems Engineering Sect.
Technical University of Denmark

This work is partially supported by the EC project T-CREST.

The T-CREST Platform

Design of a time-predictable multi-core platform

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

The T-CREST Platform

Design of a time-predictable multi-core platform

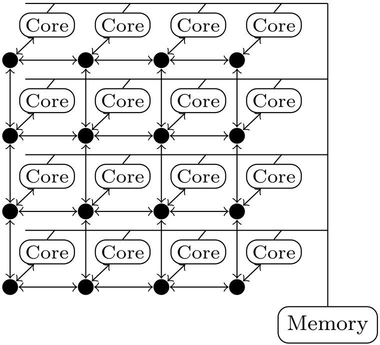
- Simplify calculation of WCET bounds
- Explore WCET-driven system design and optimization
- Project scope
  - Processor / Chip-multi-processor (DTU)
  - Memory hierarchy (U. York, TU/e)
  - **Network-on-Chip (DTU)**
  - Compiler (DTU, Vienna UT)
  - WCET analysis (Absint)
  - Use cases (GMV, Intecs)

The T-CREST Platform

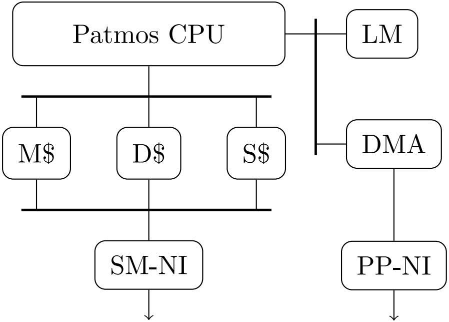
Design of a time-predictable multi-core platform

- Simplify calculation of WCET bounds
- Explore WCET-driven system design and optimization
- Project scope
  - Processor / Chip-multi-processor (DTU)
  - Memory hierarchy (U. York, TU/e)
  - **Network-on-Chip (DTU)**
  - Compiler (DTU, Vienna UT)
  - WCET analysis (Absint)
  - Use cases (GMV, Intecs)
- Project mantra
  Evaluate everything with regard to WCET analyzability.

Platform Overview

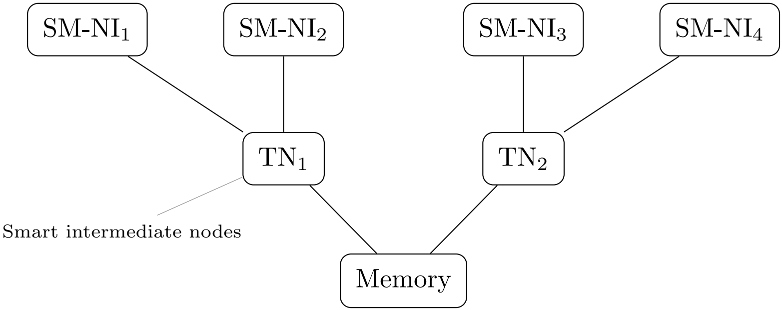


(a) Overall Platform



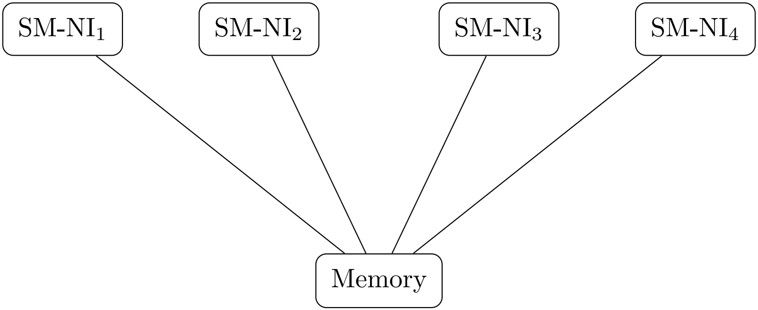
(b) Processing Core

Shared Memory – Tree



Work-preserving tree

Shared Memory – Serial Lines



Point-to-point serial lines

Point-to-Point Communication

Realized using a Network-on-Chip (NoC)

- Consisting of several Patmos cores
- Each core is connected to a router
- Routers interconnected by links
- DMA-based point-to-point exchange of messages

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

All nodes of the NoC have to be analyzed

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

- Not suited for a generic, cost-effective platform

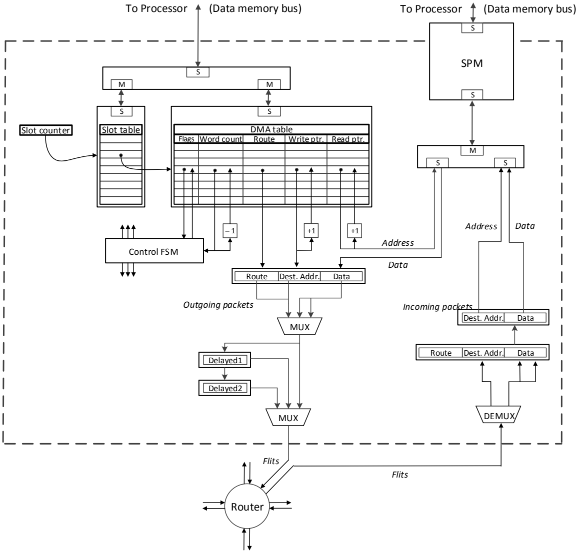
The T-CREST NoCs

- S4
  - Simple, Stupid, Small, ...
  - Minimal NoC aiming at all-to-all communication
  - Fully synchronous design
- Argo
  - Globally asynchronous locally synchronous (GALS) design
  - Asynchronous handshaking at link-level
  - Implicit *clock* driven by message injection

The T-CREST NoCs

- S4
  - Simple, Stupid, Small, ...
  - Minimal NoC aiming at all-to-all communication
  - Fully synchronous design
- Argo
  - Globally asynchronous locally synchronous (GALS) design
  - Asynchronous handshaking at link-level
  - Implicit *clock* driven by message injection
- Common ideas
  - Time-division multiplexing at all levels of the NoC
  - TDM at Network Interfaces (NI)
  - TDM on NoC links and routers

The T-CREST Network Interface



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

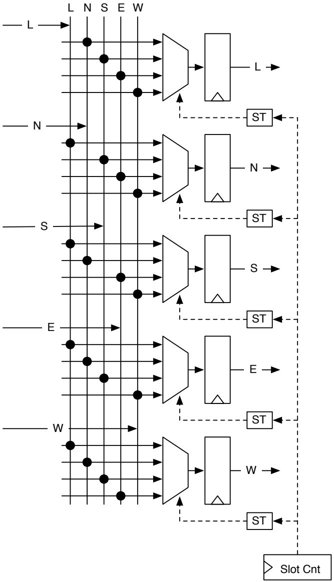
The S4 Design

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 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 ... nodes, R ... router, E ... links

Schedule Construction (2)

Solve a multi-commodity flow problem over time:

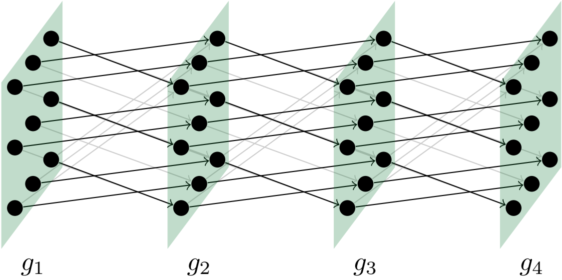
- Commodities correspond to messages
- Minimize time to deliver all commodities/messages

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: $g_1, \dots g_T$
  - Edges lead from some copy g_i to g_{i+1}
  - Solve a standard flow problem on G^T

Example: Time-Expanded 3×3 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

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

Constraints

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

Structure of the Linear Program

Variables

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

Constraints

- $\sum_{n \in N} \ell^t_{l,c} = 1$
  Use every link to send at most one message per time instant.
- $\sum_{i \in In(r,t)} \ell^t_{i,n \in N} - \sum_{o \in Out(r,t+1)} \ell^{t+1}_{o,n \in N} = 0$
  All messages flowing into a router have to flow out again.
- $\sum_{l \in In(n,t)} \ell^t_{l,n} = |N| - 1$
  Receive a message from every other node.

Structure of the Linear Program

Variables

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

Constraints

- $\sum_{n \in N} \ell^t_{l,c} = 1$
  Use every link to send at most one message per time instant.
- $\sum_{i \in In(r,t)} \ell^t_{i,n \in N} - \sum_{o \in Out(r,t+1)} \ell^{t+1}_{o,n \in N} = 0$
  All messages flowing into a router have to flow out again.
- $\sum_{l \in In(n,t)} \ell^t_{l,n} = |N| - 1$
  Receive a message from every other node.
- $\sum_{l \in Out(s,t)} \ell^t_{l,d} = 1$
  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

Topology	Nodes	Routers	Links	Schedule Length
Mesh	4	4	8+8	5
	9	9	24+18	10
	16	16	48+32	18
	25	25	80+50	34
Torus	4	4	8+8	5
	9	9	18+18	11
	16	16	32+32	26
	25	25	50+50	54*
Bidir. Torus	4	4	16+8	4
	9	9	36+18	10
	16	16	64+32	18
	25	25	100+50	27

* DNF ... (lower bound from CPLEX: 52)

Dealing with Larger Networks

Observations from ILP solutions

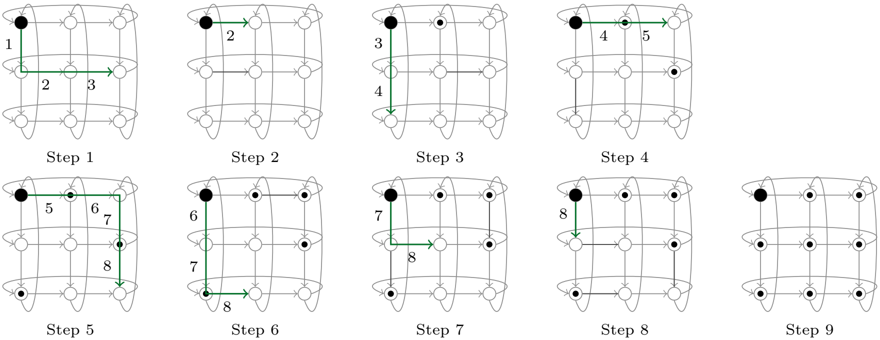
- Paths taken for message routes
    - One straight horizontal or vertical segment, or
    - One horizontal and one vertical segment (X-Y, Y-X)
    - No detours, shortest routes only

Dealing with Larger Networks

Observations from ILP solutions

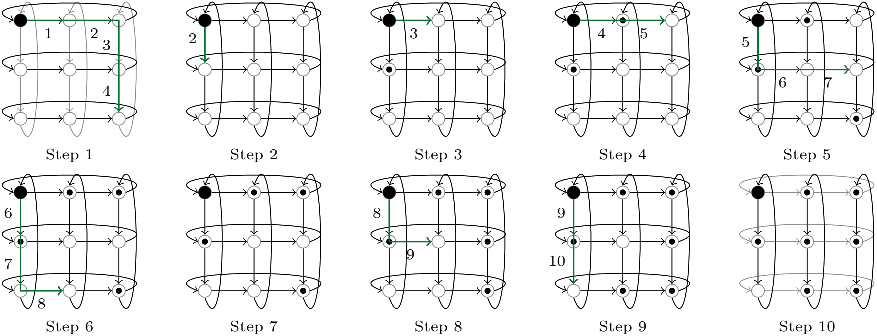
- Paths taken for message routes
  - One straight horizontal or vertical segment, or
  - One horizontal and one vertical segment (X-Y, Y-X)
  - No detours, shortest routes only
- We can exploit regularities of network topologies
  - Construct symmetric schedules
  - Synchronize all routers
  - Use the same schedule table everywhere

Example: Partial, Optimal Solution for the 3×3-Torus



This schedule cannot be replicated everywhere, due to conflicts, e.g., at cycle 2.

Example: Symmetric Solution for the 3×3-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

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
           wwss                  wwss
```

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
             wwss                  wwss
```

- 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)
- Executed on my previous netbook
  1 Dual-Core AMD Fusion E-350 (1 core), 2 GB RAM

Experiments

Several heuristic configurations

- Implemented in C++ (relatively untuned)
- Executed on my previous 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)
- Executed on my previous 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

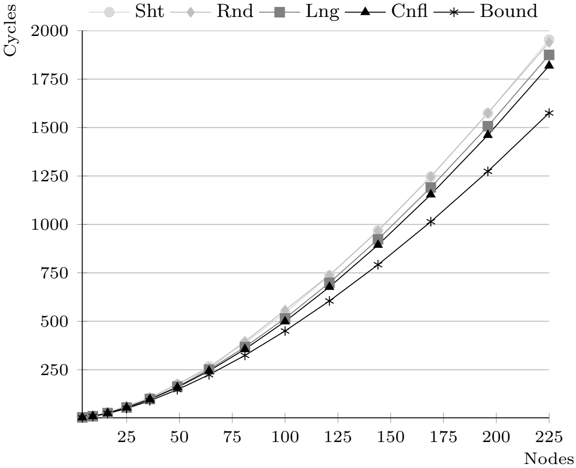
Candidate selection

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

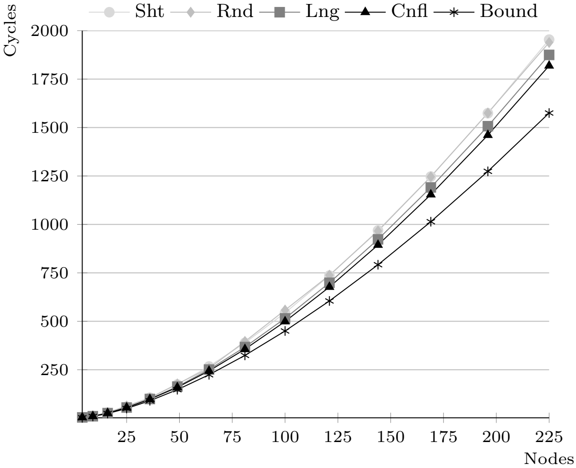
Scheduling

- Schedule as early as possible
- Naive checking for conflicts

Schedule Lengths – Torus

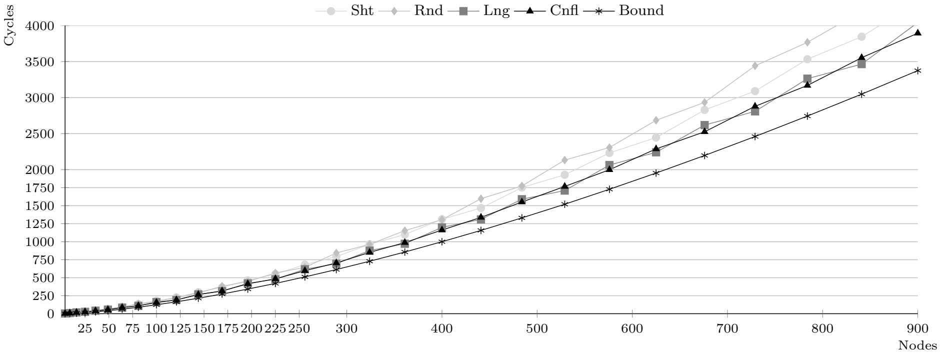


Schedule Lengths – Torus

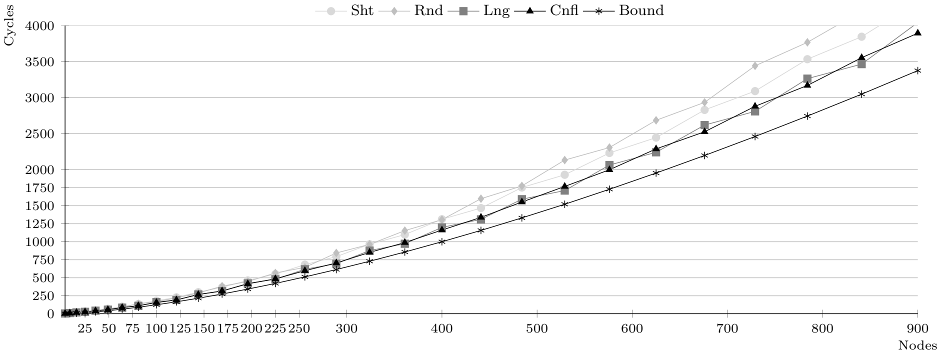


Heuristic is within 15-20% of theoretical lower bound.

Schedule Lengths – Bidirectional Torus



Schedule Lengths – Bidirectional Torus



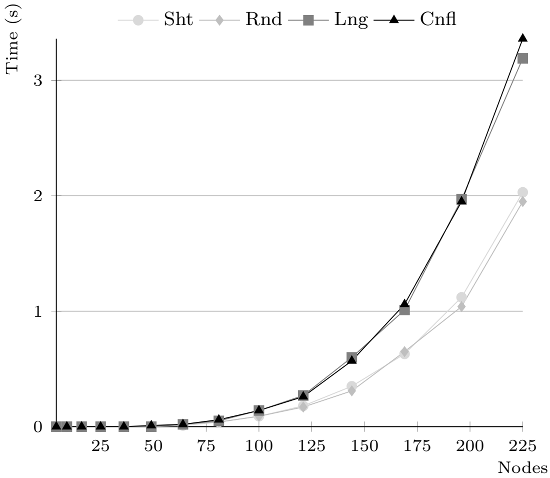
Again within 15-20% of theoretical lower bound.

Heuristic vs. Optimal Solutions

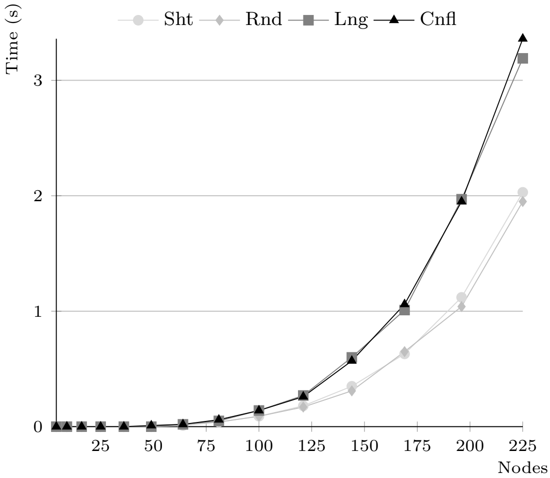
Topology	Nodes	Optimal	Heuristic	Ratio
Torus	4	5	5	1.00
	9	11	11	1.00
	16	26	27	1.04
	25	54*	56	1.04
Bidir. Torus	4	4	5	1.25
	9	10	10	1.00
	16	18	19	1.06
	25	27	28	1.04

* DNF

Execution Time – Bidirectional Torus



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