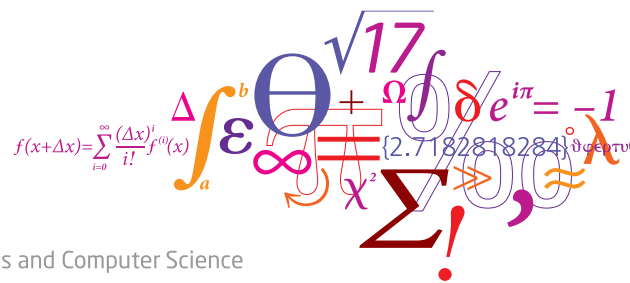
Time-Division Multiplexing vs Network Calculus: A Comparison

Wolfgang Puffitsch, Rasmus Bo Sørensen, Martin Schoeberl

RTNS '15, Lille, France



DTU Compute
Department of Applied Mathematics and Computer Science

Motivation

- Modern multiprocessors use networks-on-chip
- Congestion leads to excessive worst-case latencies
- Real-time systems require guaranteed service
- Static arbitration with predefined schedule
  - Time-division multiplexing
- Dynamic arbitration with rate control
  - Network calculus
  - (Response-time analysis)

Time-Division Multiplexing

- Static schedule
- Packets delayed at source, then flow freely
  - No collisions, no flow control, no buffering
- Fits bottom-up approach
  - Time-predictable computer architectures

TDM Latency

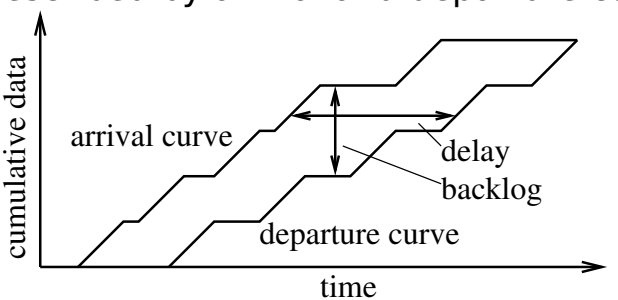
One packet per round, over n hops:

$$L_{\max}^{\text{TDM}} = T_r - 1 + (n-1)p + nd + l_{\max}$$

T_r . . . length of TDM round
p . . . router traversal time
d . . . link traversal time
$l_{\max}$ . . . maximum packet length

Network Calculus

- Mathematical results for dynamically scheduled networks
- Fits top-down approach
  - Analysis of COTS platforms
- Traffic described by arrival and departure curves



Network Calculus Latency

- Approach by Dupont de Dinechin for Kalray NoC [1]
- Bandwidth ρ and burstiness σ

  ρ number of packets N_{max} that may be injected over sliding time window T_w

  $\sigma = \rho(1-\rho)T_w$

Over n hops:

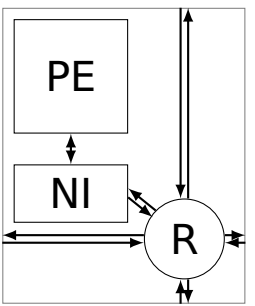
$$L_{max}^{RC} = \frac{\sigma}{\rho} + \frac{(n-1)l_{max}}{\rho} + n(d + l_{max})$$

d . . . link traversal time

l_{max} . . . maximum packet length

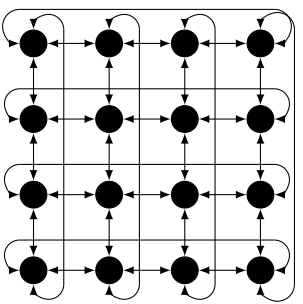
Network-on-Chip Model

- TDM NoC modeled after Argo NoC [2]
- Rate-controlled NoC like Kalray MPPA NoC [1]
- Tiles
  - Processing element PE
  - Network interface NI
  - Router R
- Source routing, push communication



Network-on-Chip Model

- TDM NoC modeled after Argo NoC [2]
- Rate-controlled NoC like Kalray MPPA NoC [1]
- 4×4 bi-torus (max. 6 hops)
- Link trav. time $d = 1$
- Router trav. time $p = 2$
- 3-word packets $l_{max} = 3$
- Queue size $Q_{size} = 401$



Evaluation Setup

- Meta-heuristic scheduler for TDM [3]
  - Shortest-path routing
  - Multiple packets on different routes
  - Optimizes for round length T_r
- Network calculus approach by Dupont de Dinechin
  - Assume fixed routes (one per flow)
  - Generate bandwidth constraints
  - We use ILOG OPL constraint solver to solve for bandwidths ρ and time window T_w
  - Maximize bandwidths or minimize latencies

All-to-All Schedule

- Every node sends one packet to every other node
- Time-division multiplexing
  - Solution with $T_r = 19 \times l_{max} = 57$ cycles
  - Optimal solution $T_r = 18 \times l_{max} = 54$ cycles
  - L_{max}^{TDM}: 66 to 75 cycles
- Network calculus approach
  - Solution with $T_w = 15 \times l_{max} = 45$ cycles
  - Optimal
  - L_{max}^{RC}: 144 to 291 cycles

First Observations

- 26% higher bandwidth with network calculus
- Significantly lower latencies with TDM
- Number of hops critical in network calculus
  - Application mapping important for network calculus, not so much so for TDM

Application-Specific Schedules

- All-to-all-schedule maybe not representative
- Communication benchmarks
  - FFT, encoders/decoders, robot control, . . .
  - 28 to 226 flows, varying bandwidths
- Worst-case latencies and bandwidths
- Minimum, maximum, average
  - Different flows have different latencies
- Detailed tables in paper

Example: Reed-Solomon Decoder

	Worst-Case Latency			Bandwidth		
	min	avg	max	min	avg	max
TDM	72	100.7	108	.0323	.0579	.0690
NC L	30	176.0	285	.0667	.1325	.7333
NC B	42	218.9	338	.0645	.1332	.7419

- Latencies in cycles, bandwidths as fraction of link capacity
- “NC L”: Network calculus, optimize for average latency
- “NC B”: Network calculus, optimize for bandwidth

Example: Reed-Solomon Decoder

	Worst-Case Latency			Bandwidth		
	min	avg	max	min	avg	max
TDM	72	100.7	108	.0323	.0579	.0690
NC L	30	176.0	285	.0667	.1325	.7333
NC B	42	218.9	338	.0645	.1332	.7419

- Lower average latency with TDM

Example: Reed-Solomon Decoder

	Worst-Case Latency			Bandwidth		
	min	avg	max	min	avg	max
TDM	72	100.7	108	.0323	.0579	.0690
NC L	30	176.0	285	.0667	.1325	.7333
NC B	42	218.9	338	.0645	.1332	.7419

- Higher average bandwidth with network calculus approach

Example: Reed-Solomon Decoder

	Worst-Case Latency			Bandwidth		
	min	avg	max	min	avg	max
TDM	72	100.7	108	.0323	.0579	.0690
NC L	30	176.0	285	.0667	.1325	.7333
NC B	42	218.9	338	.0645	.1332	.7419

- Network calculus approach makes use of spare capacities
- Some flows are assigned a very high bandwidth

Example: Reed-Solomon Decoder

	Worst-Case Latency			Bandwidth		
	min	avg	max	min	avg	max
TDM	72	100.7	108	.0323	.0579	.0690
NC L	30	176.0	285	.0667	.1325	.7333
NC B	42	218.9	338	.0645	.1332	.7419

- For some flows, network calculus can guarantee lower latencies than TDM
- TDM scheduler only optimizes T_r, not distribution of slots

Example: Reed-Solomon Decoder

	Worst-Case Latency			Bandwidth		
	min	avg	max	min	avg	max
TDM	72	100.7	108	.0323	.0579	.0690
NC L	30	176.0	285	.0667	.1325	.7333
NC B	42	218.9	338	.0645	.1332	.7419

- For other flows, network calculus leads to much higher latencies than TDM

Hardware Costs I

- Hardware costs depend on many factors
  - Feature size: 130nm, 65nm, 32nm, . . .
  - Optimizing for area/speed/power
  - Network interfaces, routers, links

  $\Rightarrow$ Published results not comparable
- Main functionality of routers similar
- Rate-controlled NoC needs buffers, TDM does not
- Relate size of TDM router to size of SRAM cell

Hardware Costs II

- Argo router in 65 nm: 8000 μm^2
- SRAM cell in 65 nm: $\approx$0.5 μm^2

$\Rightarrow$ 500 32-bit words same size as TDM router

$\Rightarrow$ Buffers in Kalray NoC $\approx$16$\times$ larger than Argo router

- Increasing bandwidth through wider links may be better investment than buffers

Best-Effort/Mixed Criticality

- TDM NoC cannot accomodate best-effort traffic without adding buffers
- Rate-controlled NoC only needs adding flow control
  - Relatively minor change
- Network calculus can handle priorities
  - Guaranteed service vs best-effort traffic
  - High-criticality vs low-criticality traffic

Future Work

- More aggressive scheduling for TDM
  - Minimize latencies, not only schedule length
  - Use spare capacities
- More advanced network calculus techniques
  - Use different windows T_w for different flows
- Extend comparison to response-time analysis
  - Combine low latencies with high bandwidth?

Conclusion

- TDM provides better latencies
  - Potential for even better latencies
- Network calculus leads to higher bandwidths
  - Use of spare capacities
- TDM NoC hardware smaller
  - Wider links vs buffers?
- Network calculus more flexible

Bibliography

- B. Dupont de Dinechin, Y. Durand, D. van Amstel, and A. Ghiti.
  Guaranteed services of the NoC of a manycore processor.
  In *International Workshop on Network on Chip Architectures (NoCArc)*, pages 11–16, New York, NY, USA, Dec. 2014. ACM.
- E. Kasapaki, M. Schoeberl, R. B. Sørensen, C. T. Müller, K. Goossens, and J. Sparsø.
  Argo: A real-time network-on-chip architecture with an efficient GALS implementation.
  IEEE Transactions on Very Large Scale Integration (VLSI) Systems, PP, 2015.
- R. B. Sørensen, J. Sparsø, M. R. Pedersen, and J. Højgaard.
  A metaheuristic scheduler for time division multiplexed network-on-chip.
  In *Software Technologies for Future Embedded and Ubiquitous Systems (SEUS), 2014*. IEEE, 2014.