A Low-Latency and Flexible TDM NoC for Strong Isolation in Security-Critical Systems

Davide Bertozzi, Mariem Turki
University of Ferrara (Italy)

Miguel Gorgues Alonso, José Flich
Universidad Politécnica de Valencia (Spain)

Motivation

• REQUIREMENT: STRONG ISOLATION
Even cycle-level variations of communication performance induced by interference between domains should be prevented.

Sharing & Security

• The unprecedented levels of interconnection among computing devices and of aggregation of IoT services makes the protection from security threats from malicious adversaries of paramount importance.

It’s mainly a network-on-chip business!

With irregular partition shapes, packet routes can «invade» other partitions

• Link-level QoS support can already prevent DoS and bandwidth depletion attacks from other domains.
• QoS cannot easily avoid an information leak associated with latency and throughput variations of communication flows.

Baseline TDM (Global schedule)
Local schedules exist...

One fundamental issue is that communication performance in time-multiplexed NoCs is highly sensitive to the scheduling methodology of time slots.

Statically schedule domains on the network over time
Time-division multiplexing

Runtime reconfiguration of the generic global schedule is hard to make fast and cost-effective.
Goal of the Work

The main goal of our proposal is to deliver the strong isolation property...

- for a runtime-configurable number of domains
- optimizing the latency of TDM communication flows
- providing scalable latency across the entire configuration space without major architectural modifications

**Howto**

TDM schedule and programming mechanism inspired by the Channel Dependency Graph of the network

---

### Synchronized Token Propagation

**Why not using tokens to carry scheduling commands to local router-level domain schedulers?**

At cycle t:

- Token
  - Domain ID: D1

Effect: Only packets of domain D1 can move to switch allocation

**Why not injecting a token at every cycle so to schedule NoC resources uninterruptedly?**

At cycle t:

- D2
  - Token
- D1
  - Token

Effect: Resource allocation requires the delay of token propagation to be synchronized so that every router can schedule the same domain at each input port at each cycle.

---

### Synchronized Token Propagation: HOW?

**Basic Idea**

We can span the CDG by propagating a token that fires on an output port when all inputs with routing dependencies that have the output have token as with.

If the CDG is acyclic (i.e., deadlock-free) each port of each switch will receive the token exactly once.


---

**Requirement for Router-Level Strong Isolation:**

Relative latencies should be a multiple of the number of domains (number of domains is the MCD of all relative latencies)

---

**Requirement for Network-Level Strong Isolation:**

A: compute the MCD for each router
B: The MCD of the router-level MCDs (if any) is the ideal number of domains, for which the network as a whole works with strong isolation of domains.
C: With SR routing on a 2D mesh, such on MCD is also the latency of the smallest cyclic path (SCL) spanned by a token to reach two different ports of the same router (SCL=4).
Perfect Scheduling

If this property holds, then as a side effect we have the onset of unstoped
propagating waves of synchronized same-domain tokens throughout the network

Let us set up SCL domains

- A 2D-mesh with periodic horizontal
  segment-based routing and single-cycle
  switches and links fulfills the property.
- AT REGIME, it enables perfect
  scheduling with SCL=4 domains.

Supporting a Higher Number of Domains

How to handle Number Domains \( D > SCL \) while preserving strong isolation
and perfect scheduling and without major changes of the architecture?

1. Split the critical path of token propagation throughout the CDG into subnets of length SCL
   clock cycles: all I/Os of these subnets will be in the same domain at a given time slot.

2. Place domain propagation DELAYS selectively within subnets, in the same position, to realign token
   propagation.

3. The number of delays in each subnet should be set to \( D - SCL \).

Experimental Results

- In our experimental test we used:
  - SystemC-based VirtualSoC
  - Both Intra-domain traffic
    and memory controller traffic

- To test flexibility/scalability, four mechanisms were evaluated across a different number
  of running domains: Global-TDM, Optimized Global TDM, PhaseNoC, Token-based TDM

- The PhaseNoC router architecture also delivers perfect
  scheduling and strong isolation...

- ...but not for all system configurations
  (i.e., number of domains).

- Major architecture change, higher latency
  Increasing the pipeline depth

- No perfect scheduling across
  subnets. No strong isolation
  Splitting the network into subnetworks
  (e.g., \( X \times Y \times X \times Y \))

Experimental Results

- Local traffic in a 64-tile scenario
- MC traffic in a 64-tile scenario

- The improvement of Token-based TDM
  oscillates between 20% (5 domains) and 9%
  (15 domains).

- Token-Based TDM reaches the best performance improving up to 30% the
  PhaseNoC network latency.

- Token-based TDM offers better
  latency scalability over the whole configuration space
  Token-based TDM is fully re-programmable while PhaseNoC architecture
  should be changed at each configuration.
Experimental Results

➢ Local schedules consistently perform better than global schedules from both a latency and a throughput viewpoints.
➢ Token-based TDM reduces PhaseNoC latency by roughly 10% and 20% for local and MC traffic, respectively, while matching the saturation bandwidth.

Conclusions

• Emerging applications increasingly call for strongly isolated & high performance NoCs
• TDM NoCs have to be brought into new ground:
  ✓ Scheduling of slots should be optimized for latency
  ✓ Reconfigurability becomes a must
• We propose a CDG-inspired approach to derive TDM slot scheduling and a reprogramming framework for such scheduling
• We consistently provide generalized perfect scheduling across the whole configuration space.
• Each configuration can be supported without major architectural modifications, unlike state-of-the-art.

Thank you for your attention!