# A Complete Network-On-Chip Emulation Framework


**Nicolas Genko, LSI/EPFL Switzerland**

**David Atienza, DACYA/UCM Spain**

**G. De Micheli, LSI/EPFL Switzerland**

**J.M. Mendias, DACYA/UCM Spain**

**R. Hermida, DACYA/UCM Spain**

**F. Catthoor, IMEC Belgium**





# Outline

- Introduction
- NoC Emulation architecture
- NoC Emulation Flow
- Experimental results
- Conclusion

# Motivations

- Growing interest in NoCs.
  - Buses run out of requirements.
  - Need for structured and reliable interconnect.
- NoC Design tools
  - Synthesis tools like Xpipes allow us to quickly generate NoC models.
  - NoC models need to be accurately tested to see how well they fit each concrete application.
    - Simulation (cycle accurate).
    - Emulation on FPGA.

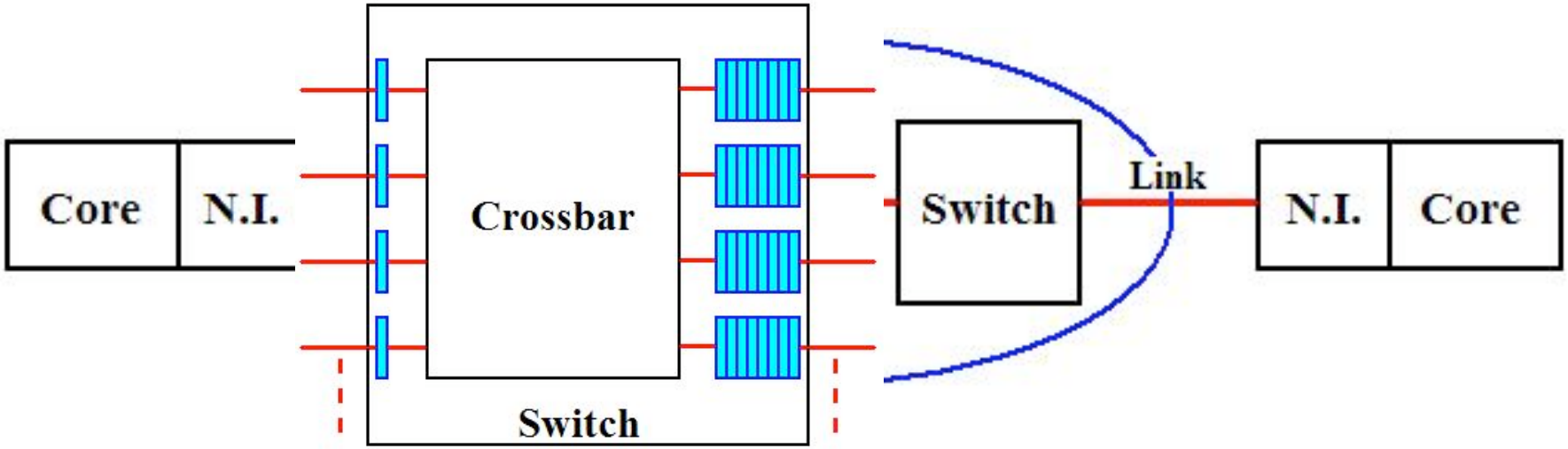
# Previous Work

- **NoC software simulation:**
  - **High level models in C/C++.**
  - **To evaluate the latency of NoCs.**
  - **To evaluate the throughput of NoCs.**
- **NoC implementation on FPGAs:**
  - **For functional validation.**
  - **To show the effectiveness of NoCs.**
  - **To validate NoCs features.**

# NoC Emulation on FPGA

- The emulation can achieve important speed-ups compared to cycle accurate simulations:
  - Up to four orders of magnitude faster.
  - Real inputs with millions of packets can be emulated.
- Emulation on FPGA enables functional validation of NoC systems.
  - FPGA emulation is a way to run NoCs accurately.
- Extensive profiling of statistics and NoC emulation features are provided.

# NoC Architecture to be emulated




- **Our focus:**
  - **Switch topology**
  - **Switch parameters**
    - **Number of inputs**
    - **Number of outputs**
    - **Size of buffers**

# Outline

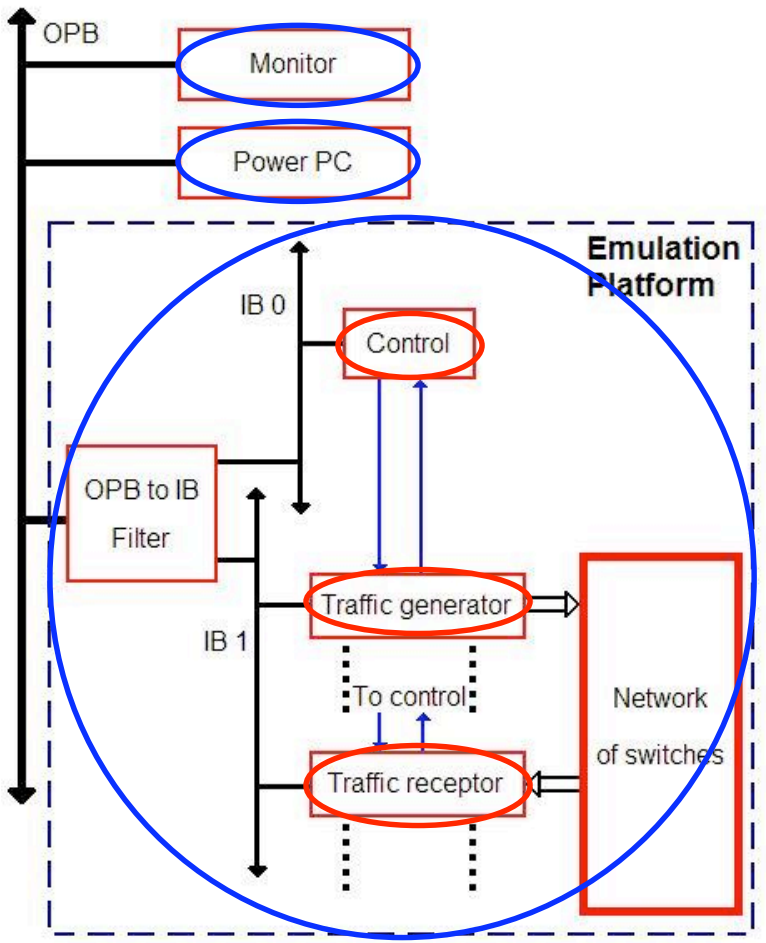
- Introduction
- NoC Emulation architecture
- NoC Emulation Flow
- Experimental results
- Conclusion

# NoC Emulation architecture

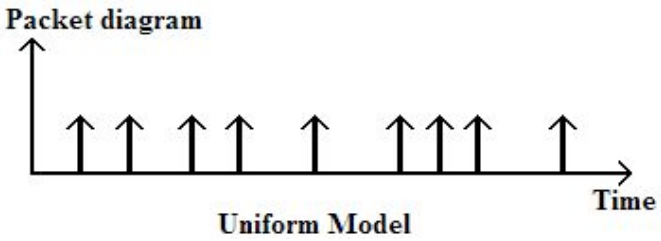
• A Processor (i.e. PowerPC): Orchestrates the whole process.
• A monitor: Display on the screen of a PC the information extracted from NoC emulation components.
• The emulation platform.

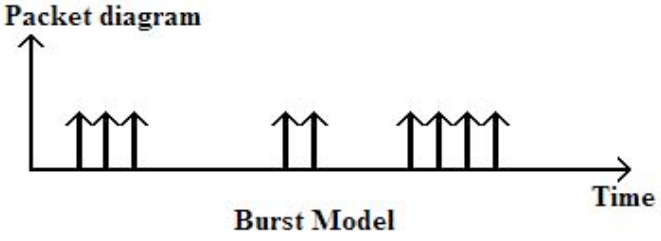
• The processor can access each component by accessing their specific addresses.

• In our design, we allow up to 4 internal busses and 1024 devices in each internal bus.



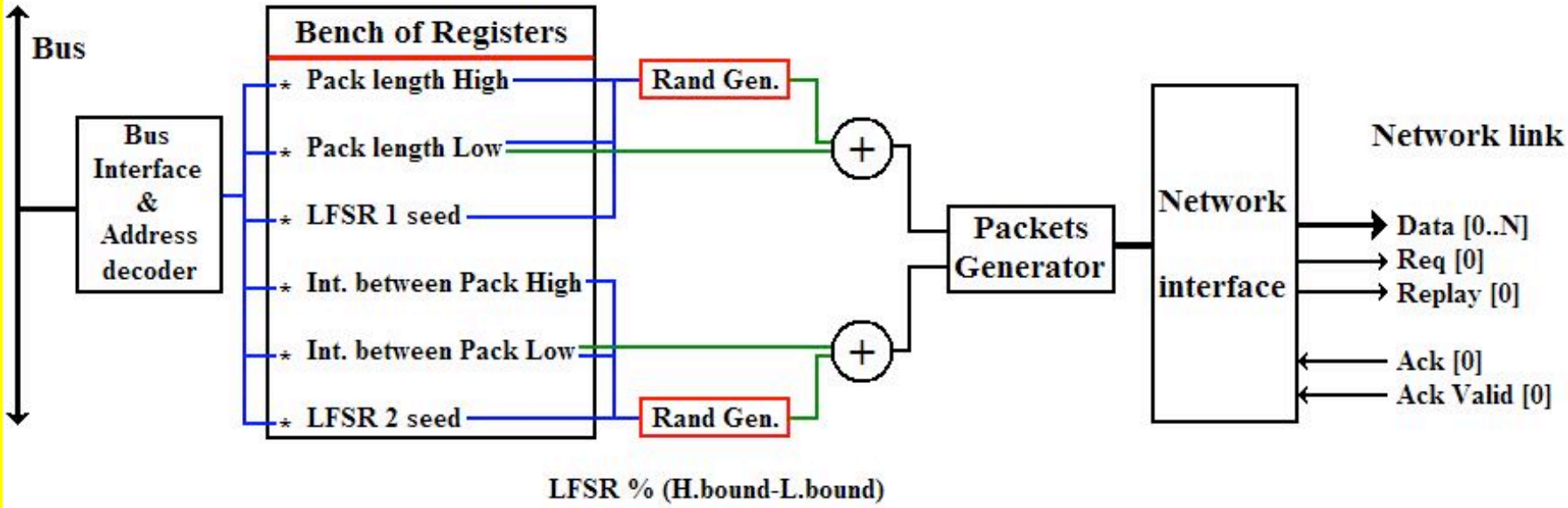
# Stochastic Traffic




- **Uniform Model; Parameters:**
  - **Length of packets.**
  - **Interval between packets.**
- **Burst Model; Parameters:**
  - **Transition probabilities in a 2-state Markov chain.**
- **Other models possible (i.e. Poisson…).**
- **Trace driven traffic generators:**
  - **Generates traffic from a trace recorded on a real life application.**

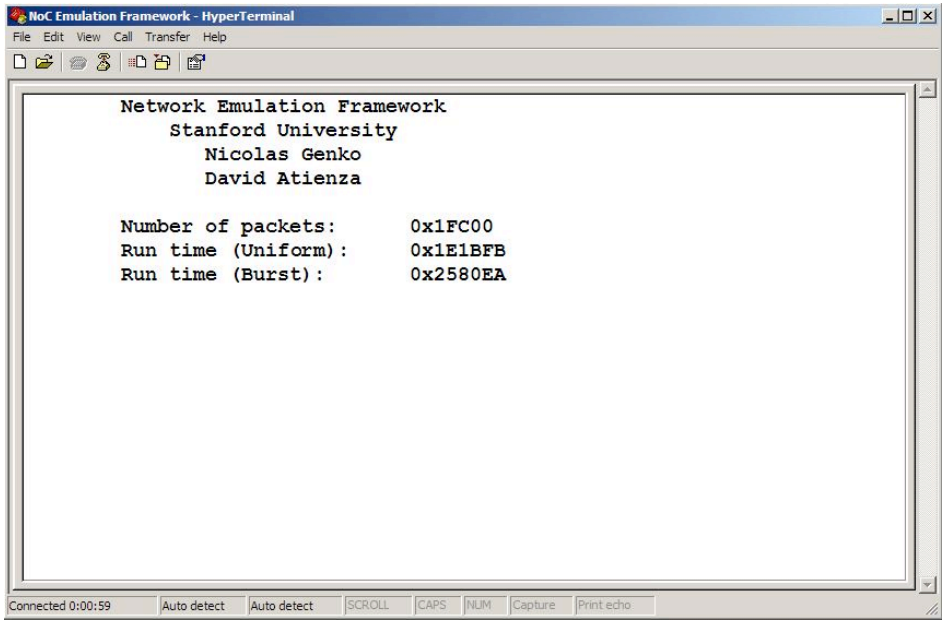
# Example of TG structure




- A bench of registers:
  - For traffic parameterization.
  - For random initialization.
- A packet generator which generates various traffic patterns.
- A Network interface:
  - Converts a traffic pattern in flits for NoC.
  - Can be adapted for any type of NoC.

# Statistics reports and analysis

- **Stochastic receptors:**
  - **Histograms, which show an image of the received traffic.**
  - **Total running time.**
- **Trace driven receptors:**
  - **Latency analyzer.**
  - **Congestion counter.**



# Outline

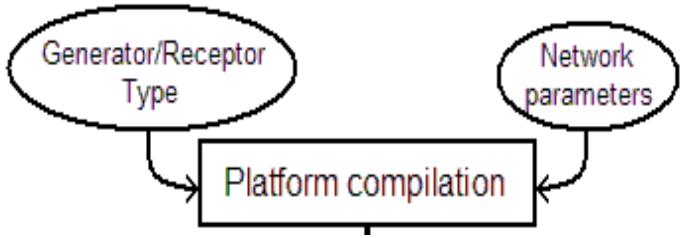
- Introduction
- NoC Emulation architecture
- NoC Emulation Flow
- Experimental results
- Conclusion

# NoC Emulation flow: Approach

- **Objective: A HW/SW emulation environment.**
  - **Provide a versatile emulation platform.**
  - **Avoids often hardware re-synthesis.**
- **HW part: network of switches to emulate any NoC packet-switching intercommunication scheme.**
  - **It can emulate different types of NoC and compare their features.**
- **SW part: A processor configures and rules the NoC emulation platform features to emulate and statistics acquisition.**

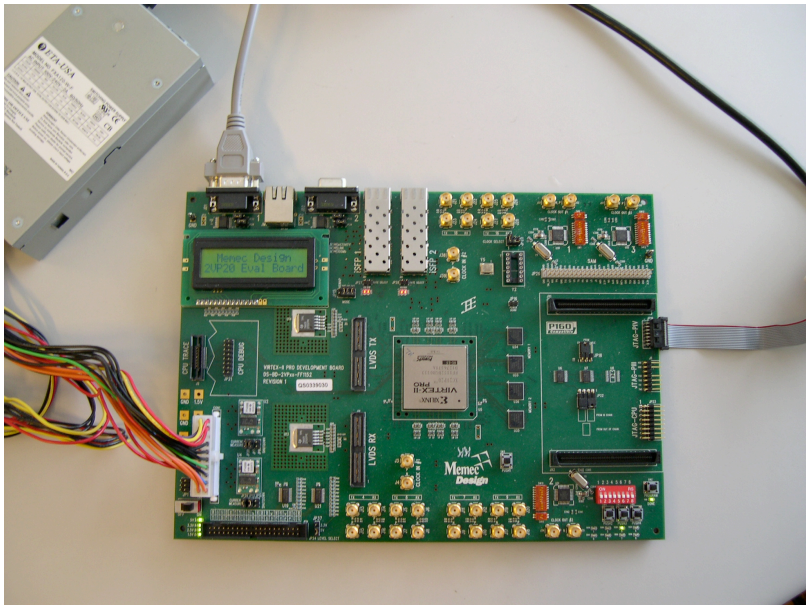
# NoC Emulation flow: Overview

1) Platform compilation: Setup of NoC parameters, type of TG/TR.
2) Physical synthesis.
3) Platform initialization: Setup the software with emulation parameters.
4) Software compilation.
5) Emulation on FPGA: The emulation runs according to the user-specific setup.
6) Final report: The user visualizes the results of the emulation on the screen of his/her PC.



# Emulation setting

- **Platform settings:**
  - **Topology, type of generators…**
- **Software settings:**
  - **Traffic definition, orchestration of the emulation…**
- **Ease of use:**
  - **Simplicity of the flow.**
  - **Speed of the emulation.**



# Outline

- Introduction
- NoC Emulation architecture
- NoC Emulation Flow
- Experimental results
- Conclusion

# FPGA reports

- **Platform with:**
  - **4 TG, 4 TR**
  - **6 switches**

→ **7387 Xilinx slices (80%)**

| Device | Number of slices | FPGA percentage (%) |
|---|---|---|
| TG stochastic | 719 | 7.8 |
| TG trace driven | 652 | 7.0 |
| TR stochastic | 371 | 4.0 |
| TR trace driven | 690 | 7.4 |
| Control module | 18 | 0.2 |

# FPGA reports

- **Platform speed:**

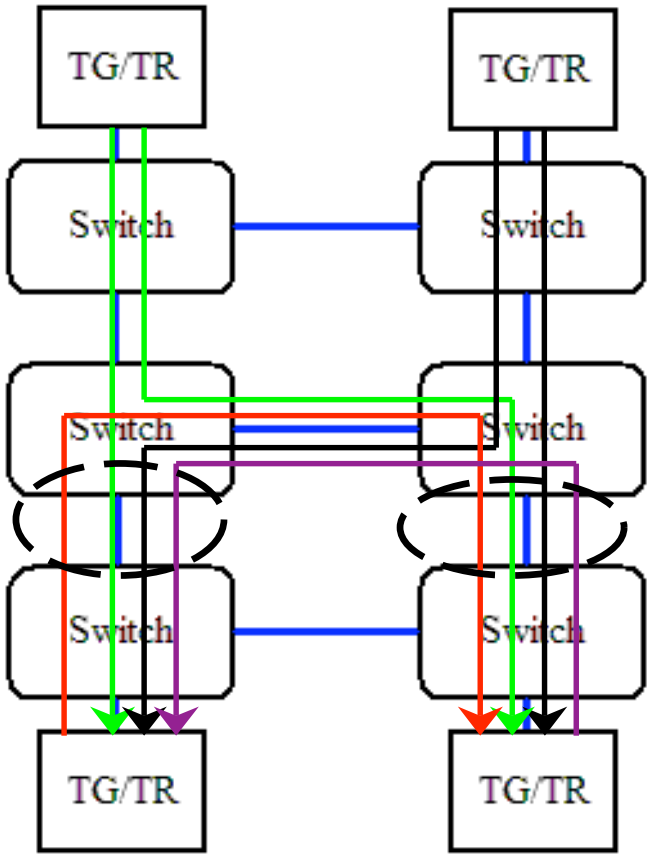
→ **50 MHz**

- **The speed has been chosen regarding the possibilities of our Virtex 2 Pro FPGA.**

| Simulation mode | Speed(cycles/sec) | Simulation time For 16 Mpackets | Simulation time For 1000 Mpackets |
|---|---|---|---|
| Verilog (ModelSim) | 3.2K | 13h53’ | 36 days 4h |
| SystemC (MPARM) | 20K | 2h13’ | 5 days 19h |
| Our Emulation | 50M | 3.2 sec | 3’20’’ |

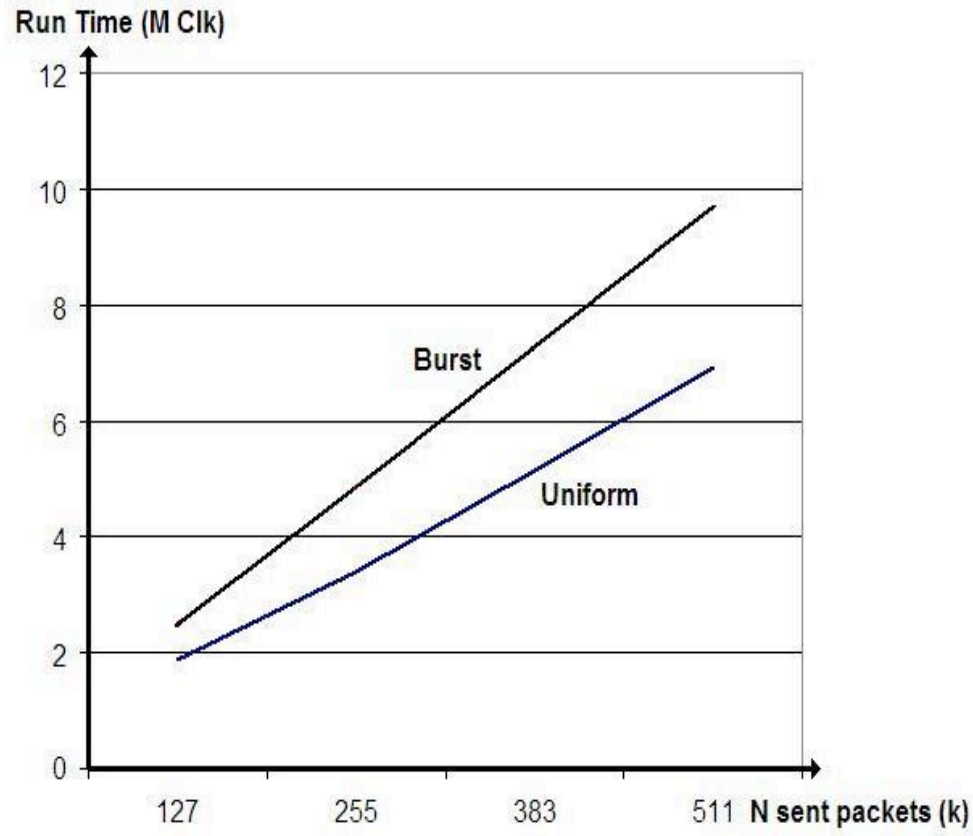
# Experimental setup

- Each TG generates some traffic at 45% of the maximum bandwidth with two routing possibilities in two cases.
- Two inter-switch links are loaded with 90% of traffic.



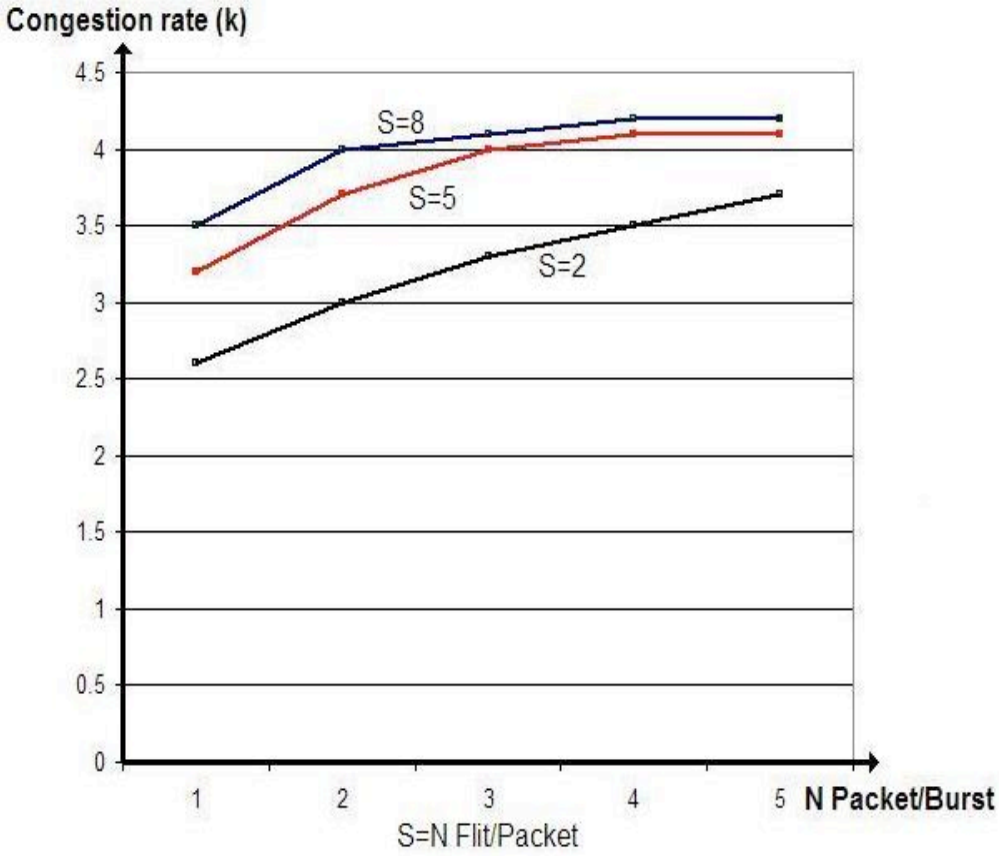
# Experimental results

- With stochastic traffic devices.
- Run-time vs. Number of sent packets.
- Burst traffic creates more congestion on the NoC than uniform traffic.



# Experimental results

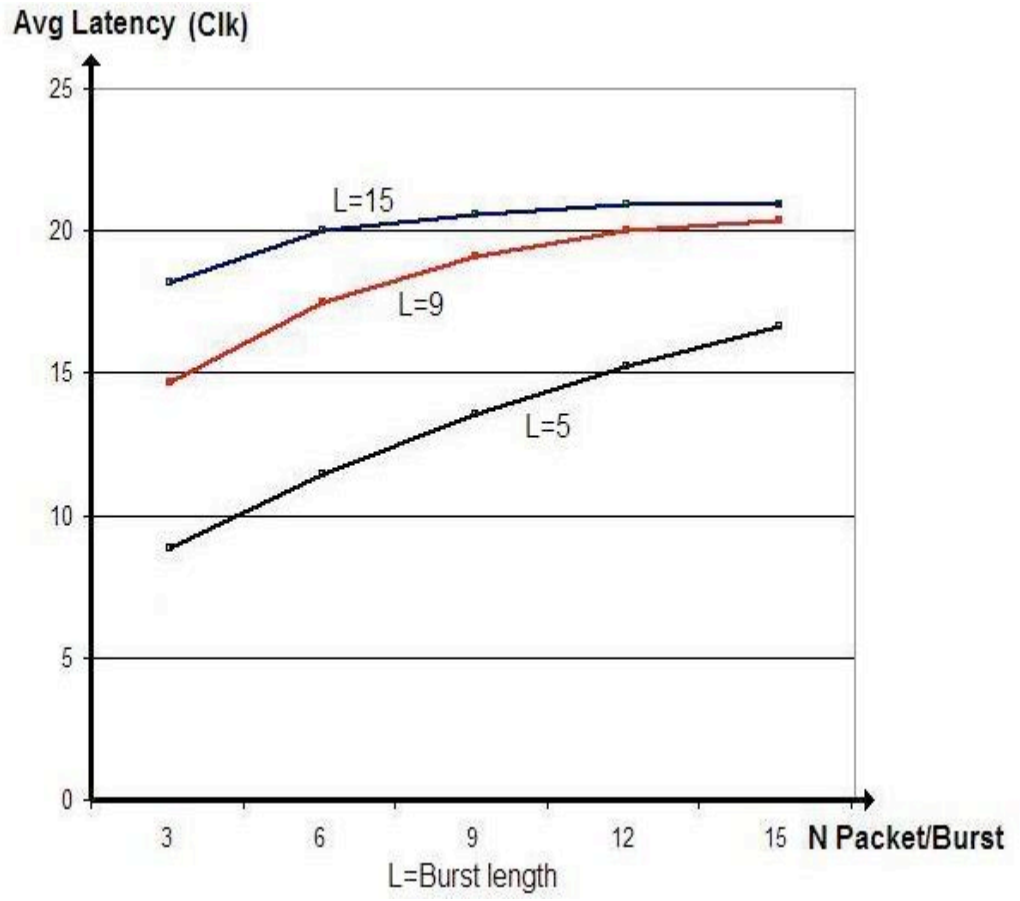
- **With trace driven traffic devices.**
- **Congestion rate vs. Number of packet/Burst with a variation of the Number of Flit/Packet.**




Measure of congestion according to burst's length in flits.

# Experimental result

- With trace-driven traffic devices.
- Average latency vs. N of packets/Burst.
- The latency reaches a maximum.
- The maximum is a function of the congestion rate (90%).



# Outline

- Introduction
- NoC Emulation architecture
- NoC Emulation Flow
- Experimental results
- Conclusion

# Conclusion

- **Important speed-ups in NoC studies are possible with our NoC emulation platform.**
- **Our HW/SW emulation solves efficiently the time-consuming problem of HW re-synthesis in case of many NoC parameters changes.**
- **With larger FPGAs, it will be possible to emulate very large NoCs (tens of switches in the next generation of FPGAs).**