Mapping and Physical Planning of Networks-on-Chip Architectures with Quality-of-Service Guarantees

Srinivasan Murali [1], Prof. Luca Benini [2], Prof. Giovanni De Micheli [1]

[1] Computer Systems Lab, Stanford University

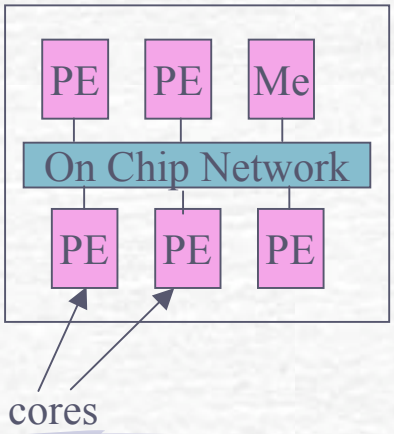
[2] DEIS, University of Bologna

Outline of Talk

- Introduction
- Objective of the work
- Previous Work
- Design Methodology
- On-Chip Traffic Modeling
- Problem Formulation
- Mapping and Physical Planning Algorithm
- Experiments and Case studies
- Conclusions and Future Work

Introduction

- Present & Future SoCs designed using pre-existing components, called *Cores*
- Future SoCs will have
  - Increased functionality: many applications on a chip
  - Increased speed of cores
- Scalable Networks on Chips (NoCs) needed to accommodate the high communication requirements
- NoCs provide:
  - Higher Performance, and
  - Better Structure
  - Modularity
  - Compatible with core design & reuse

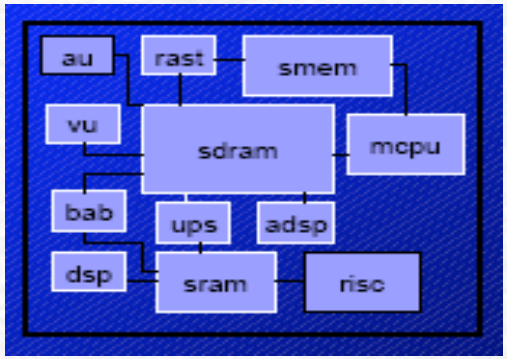


Why on-chip networking ?

- Tame the complexity of SoC design
  - Separation of computation and communication
    - Component-based design
    - Organic realization of on-chip communication
- Address limitations of DSM technologies
  - Relatively-long wire delays
  - Unreliable electrical-level transfer
- Provide a structured methodology for realizing on-chip communication schemes
  - Address performance and energy issues

NoC Mapping

- SoCs aggressively designed to meet performance requirements
- Mostly the cores in the SoCs are heterogeneous in nature:
  - each core has different functionality, size & communication requirements.
  - maximizes performance and satisfy design constraints such as *Quality-of-Service (QoS)* for the applications.



MPEG4 Decoder

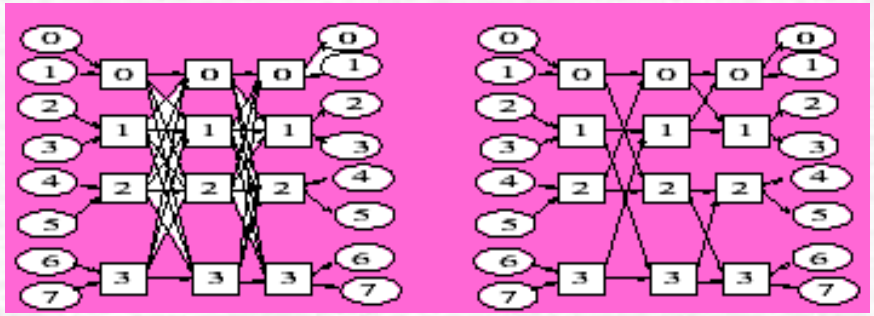
- Some of the cores are *hard cores*, with size fixed during design & some of the cores are *soft cores*.
- Application-specific mapping of cores onto suitable NoCs is an important phase in NoC design

QoS Requirements

- Another important design consideration for SoCs is to guarantee *Quality-of-Service (QoS)* for the application.
- The network should support QoS requirements of applications
  - satisfy delay constraints of traffic streams.
  - provide support for real-time communication.
- These QoS guarantees need to be considered during the mapping process.
- Moreover the burstiness in the traffic streams (that makes providing QoS guarantees harder) needs to be considered.

Objective of this Work

- Traditionally, mapping is followed by Floorplanning where position, size of cores & network components computed.
- We present an integrated approach to mapping of cores onto NoC topologies and physical planning of NoCs
- Take NoC specific features into physical planning of the topology during the mapping process.



Clos and Butterfly topologies

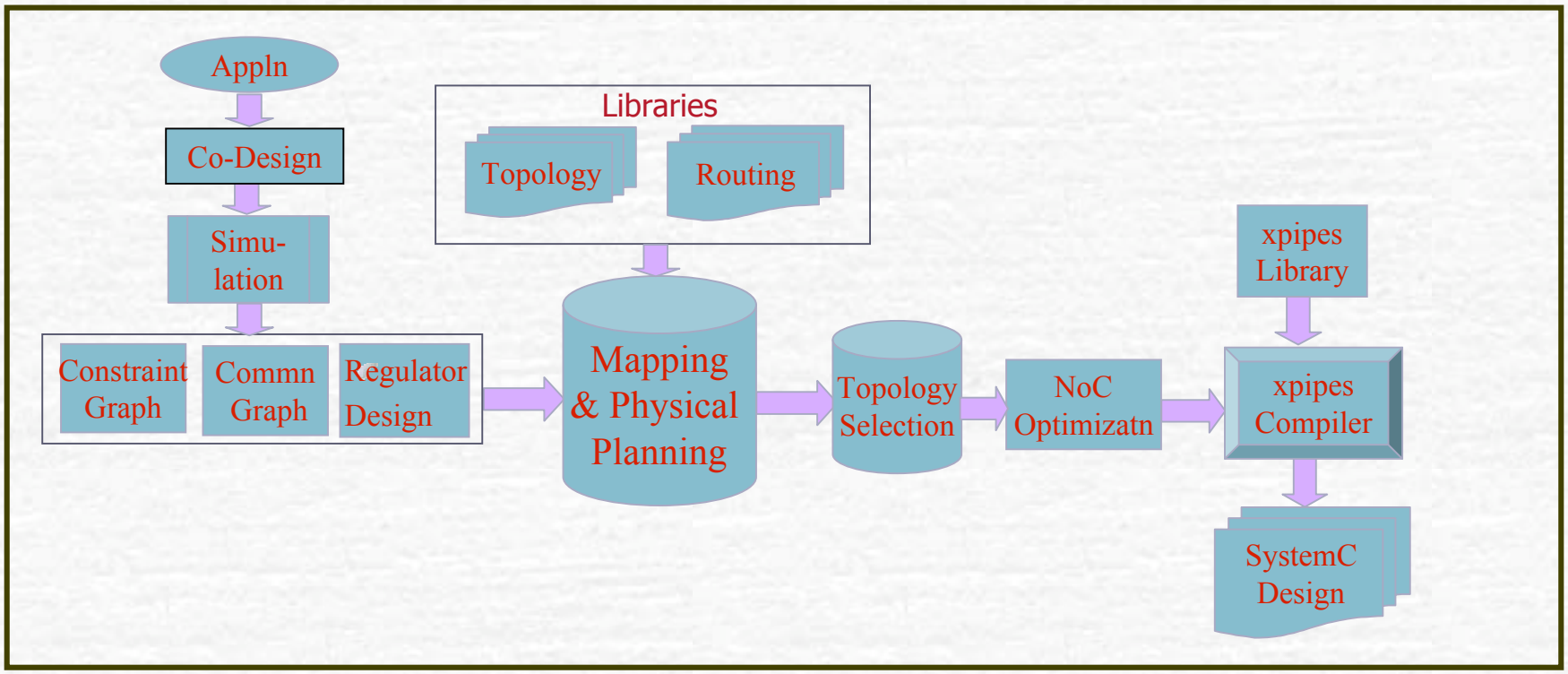
Objective of this work

- We present a method to provide QoS guarantees for application during mapping-physical planning phase.
- For QoS guarantees, we consider:
  - the burstiness in the application traffic
  - delay/jitter constraints of the individual traffic streams
  - provide support for real-time communication
- Once the mappings onto NoC topologies are obtained, we size the NoCs according to traffic characteristics.
- We integrate these features to *Netchip* design flow, automating mapping, physical planning, NoC optimization and generation.

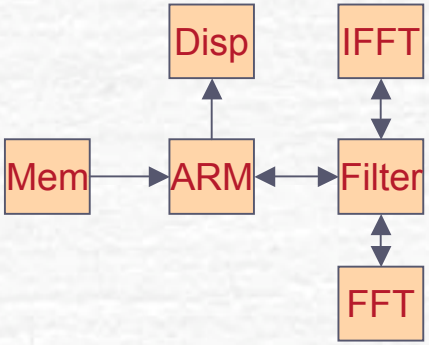
Previous Work

- Large body of research focuses on the design methodologies for NoCs such as *Nostrum*, *SPIN*, *Aetheral, aSoC, Proteo, Netchip, …*
- NoC Topology mapping for applications has been presented by W.H.Ho et al. (HPCA 03), A.Pinto et al. (ICCD 03), J.Hu et al. (ASPDAC 03, DATE 03), S.Murali et al. (DAC 04).
- Physical planning for NoCs presented by T.T.Ye et al. (ASAP 03).
- QoS guarantees to applications is provided by efficient router design in the *Aethereal NoC* (DATE 03).
- All these works don't provide integrated mapping and physical planning.
- They do not consider QoS guarantees during mapping process.

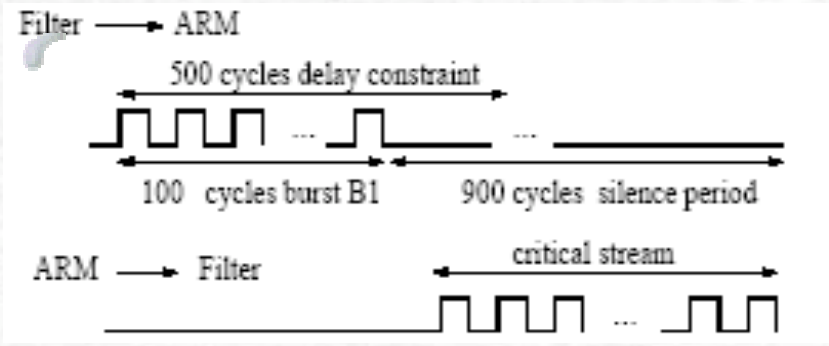
Netchip Design Flow



Traffic Modeling



Filter Application



Sample Traffic Trace

Traffic Characteristics:

- The application traffic from *Filter* to *ARM* core is bursty in nature, peak bandwidth is much higher than average bandwidth.
- Each burst from the *Filter* core has a delay constraint by which it should reach the *ARM* core.
- The *ARM* core issues a control stream to the *Filter* which is assumed to be critical and needs to reach the *Filter* as quickly as possible.

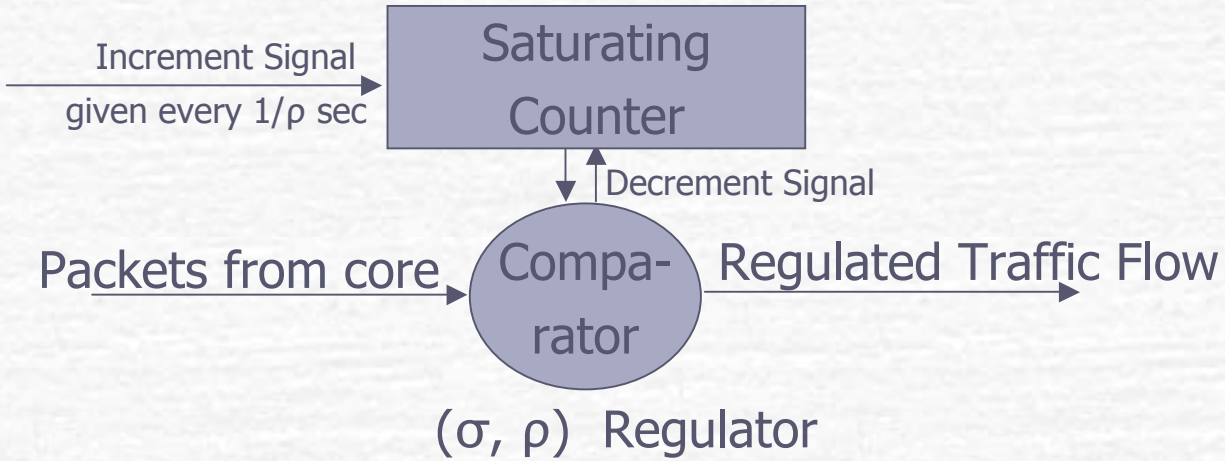
Traffic Modeling

- Consider three implementations of link between Filter and ARM core.

Scheme		Delay (cycles)
Average BW		1000
Peak BW		100
Optimal BW		500

- Communication links should be designed optimally to support traffic flowing through them, satisfying the delay/jitter constraints.
- Moreover, there should be a mechanism to ensures that- each core sends traffic that links can support & delay constraints are met.
- To satisfy these complementing objectives, we propose use of traffic regulators for NoCs, that are traditionally used in ATM networks.

Traffic Regulator Design



- The (σ, ρ) regulator is added to each core or network interface.
- The parameter ρ represents bandwidth required, so that the delay constraints are met.
- The parameter σ represents the variations permitted over the value.
- A packet is transmitted only if the counter is non-zero and when a packet is transmitted the counter is decremented by 1.
- The counter is incremented at rate ρ.

Traffic Regulator Design

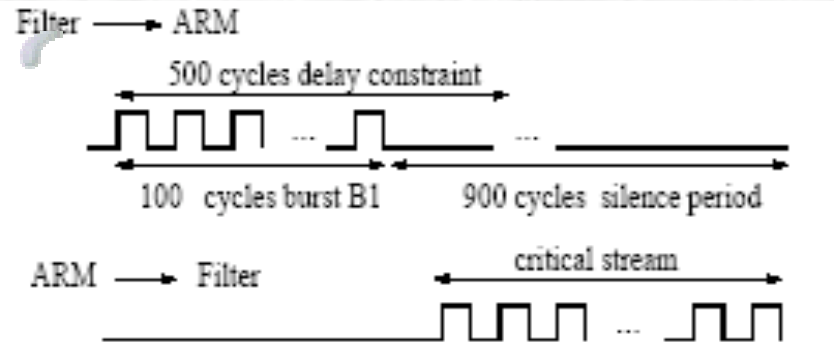
- Regulator ensures that traffic rate transmitted by source matches the rate of links.
- The ρ values for the traffic flow between a source (*i*) to destination (*i*) is obtained by:

$$\rho_{i,j} = max_{\forall k \in M_{i,j}} \left(\frac{blen_{i,j,k}}{blen_{i,j,k} + blat_{i,j,k}} \right)$$

  - Where: $M_{i,j}$ are the set of bursts from i to *j*.
  - $blen_{i,j,k}$ is the length of the k^{th} burst, in cycles.
  - $blat_{i,j,k}$ is the slack of the k^{th} burst, in cycles.

Traffic Regulator Design

- As an example,



- $blen_{i,j,1}$ is 100, $blat_{i,j,1}$ is 400 cycles.
- Thus $\rho_{i,j,1}$ is 0.2 packets/cycle.

Problem Formulation

- The communication between the cores can be abstracted by the weighted core graph G.
- The edge weights of G are function of criticality of streams & amount of traffic communicated.
- The traffic constraints are abstracted by the constraint graph CG.
- The edge weights in the CG are ρ (x packet size/ cycle time) values for the corresponding traffic flows.
- The NoC topology is defined by adjacency information of nodes in the topology and by the capacity of the links by the graph P.
- The mapping of cores onto NoC topologies is a graph mapping problem such that:
  - Each link in the mapped NoC should satisfy the bandwidth constraints corresponding to the constraint graph CG.
  - The design objective (area, power or hop delay) of a mapping is obtained from the physical planning of the mapping.

Mapping & Physical Planning Algorithm

- NoC mapping is intractable, and is a special case of the *Quadratic Assignment Problem (QAP).*
- *QAP* is well studied in the literature with many heuristic algorithms available.
- *Robust tabu search* is shown to be most effective for many classes of QAP and we use this to solve our mapping problem.
- In the first step an initial greedy mapping of the cores onto the topology is obtained.
- Then for each iteration of the *robust tabu search*, we perform the following computations:
  - Compute the routes for the traffic flowing between the cores, based upon the routing function chosen from the library.
  - Physical planning for this mapping. This includes computing the positions of the cores and the switches, sizes of the switches & soft cores.
  - Check whether the mapping satisfies the delay/jitter and area constraints.
  - optimize the design objective (area, power or hop delay) satisfying the QoS and criticality constraints.

Physical Planning

- We modify an Mixed Integer Linear Program (MILP) floorplanner by J.G. Kim et al. (IEEE TCAD 03) by considering NoC specific features.
- As the cores are pre-designed components, assume the area and power values of the cores as an input.
- Assume type of core (hard or soft) & aspect ratio constraints as input.
- We use area, power libraries for switches based on xpipes architecture.
- For a given mapping, the relative position of the cores with respect to each other is obtained from the tabu search.
- Relative position of the switches is unknown.
- For direct topologies (like mesh, torus, hypercube), we assume the switch to lie in a region surrounding the core.
- For indirect topologies, we distribute the switches along the cores in a 2D plane, based on their connectivity to the cores and to other switches.

NoC Design

- This mapping/physical planning is applied to all topologies in the library.
- Our library currently has *mesh, torus, hypercube, Clos, butterfly, spidergon* topologies.
- The best topology is selected & switches, links are optimized to match the application characteristics.
- The links are sized according to the traffic flowing through them.
- Then, the SystemC design of the NoC is generated automatically using the ×pipesCompiler.

Experiments and case studies

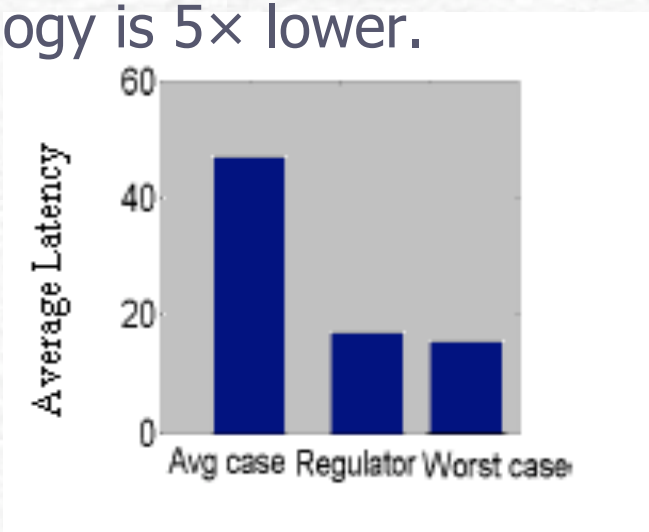
- Effect of Integrated Mapping and Physical Planning

Applica-tion	Area 1 (mm2)	Area 2 (mm2)	Ratio
VOPD	20.25	18.01	1.12
MPEG	36	20.25	1.19
PIP	20.25	10.565	1.92
MWA	33	10.565	1.32
Avg			1.39

- In the traditional scheme, we assume that the relative position of the cores is maintained as per the topology.
- The integrated scheme gives large savings as we incorporate the area computation at each step of the mapping process.

Design for QoS Guarantees

- Traditionally, QoS is guaranteed by designing the network to support worst-case bandwidth.
- This leads to an over-design of network components.
- By using traffic regulators for guaranteeing QoS, we achieve large network component savings.
- For *Filter* application, the minimum bandwidth needed for our design methodology is 5× lower.



Video Object Plane Decoder

Topology	Power (in mW)
Butterfly	No Mapping
Mesh	542
Torus	930
Hypercube	960
Clos	753

- With out considering bursty traffic, we had done an analysis of VOPD in our eariler work (DAC 04).
- Butterfly topology was found to be the best choice for VOPD
- Actual traffic for this application is bursty.
- Butterfly topology doesn't have enough bandwidth to support the bursty traffic of VOPD.
- Current analysis gives Mesh topology to be the best choice.

Buffer Sizing and Network Optimization

- During physical planning, the number of buffers needed for switches is automatically computed based on the link lengths.
- When the number of buffers is lower than the required number, throughput of the network is low.
- In heterogeneous SoC, the number of buffers can be different different for different inputs of the same switch as the link lengths are non-uniform in nature.
- As this optimum buffer count is automatically computed by the physical planner we get large buffer reduction.
- For VOPD, compared to average-case design (all switches have same buffer count), get 2.2× reduction.

Buffer Sizing and Network Optimization

- After the topology selection phase, the network components (switches and links) are optimized based on the traffic flowing through them.

Components	Savings
Buffer	2.2×
Wire count	3.77×
Ports	1.6×

Conclusions

- SoCs are aggressively designed with many heterogeneous processor/memory cores to maximize system performance.
- Mapping of heterogeneous cores onto NoCs, physical planning, topology selection, optimization & instantiation are important phases in designing application-specific NoCs.
- In this work we have presented a design methodology that automates all these steps.
- Mapping and physical planning phases are integrated together, resulting in better NoC design than traditional methodology.
- Presented method for guaranteeing QoS during mapping/physical planning phase.
- Our design approach results in large network component savings.
- In future, we plan to take dynamic variations in input during mapping/physical planning phase.

Thank You

Why Standard Topologies ?

- The use of application specific topology (irregular topology) may not ensure connectivity of design
  - Connectivity useful when some dynamic relocation/scheduling is done along with static scheduling
  - For spreading out traffic, fault-tolerance
- For mostly static with partially dynamic scheduling our tool works well: used more in SoCs
- Starting point for building automatic heterogeneous topology:
  - SUNMAP gives information on which network element is the bottleneck
  - Use this information to design application-specific topology
- Application-specific topology has its own problems: deadlocks, routing
- Well, our tool can let u explore the application-specific topology if given as an input (or defined in the topology library)