Mapping and Configuration Methods for Multi-Use-Case Networks on Chips

Srinivasan Murali, Stanford University

Martijn Coenen, Andrei Radulescu, Kees Goossens, Philips Research

Giovanni De Micheli, Ecole Polytechnique Federal de Lausanne (EPFL)

Contact: smurali@stanford.edu

Introduction

- Systems On Chips have multiple components, cores
- Communication between cores rapidly increasing
- Wire scaling not on par with transistor scaling
- Communication architecture becomes major bottleneck
  - Scalability, delay, power
- Networks on Chips (NoCs) needed for SoCs
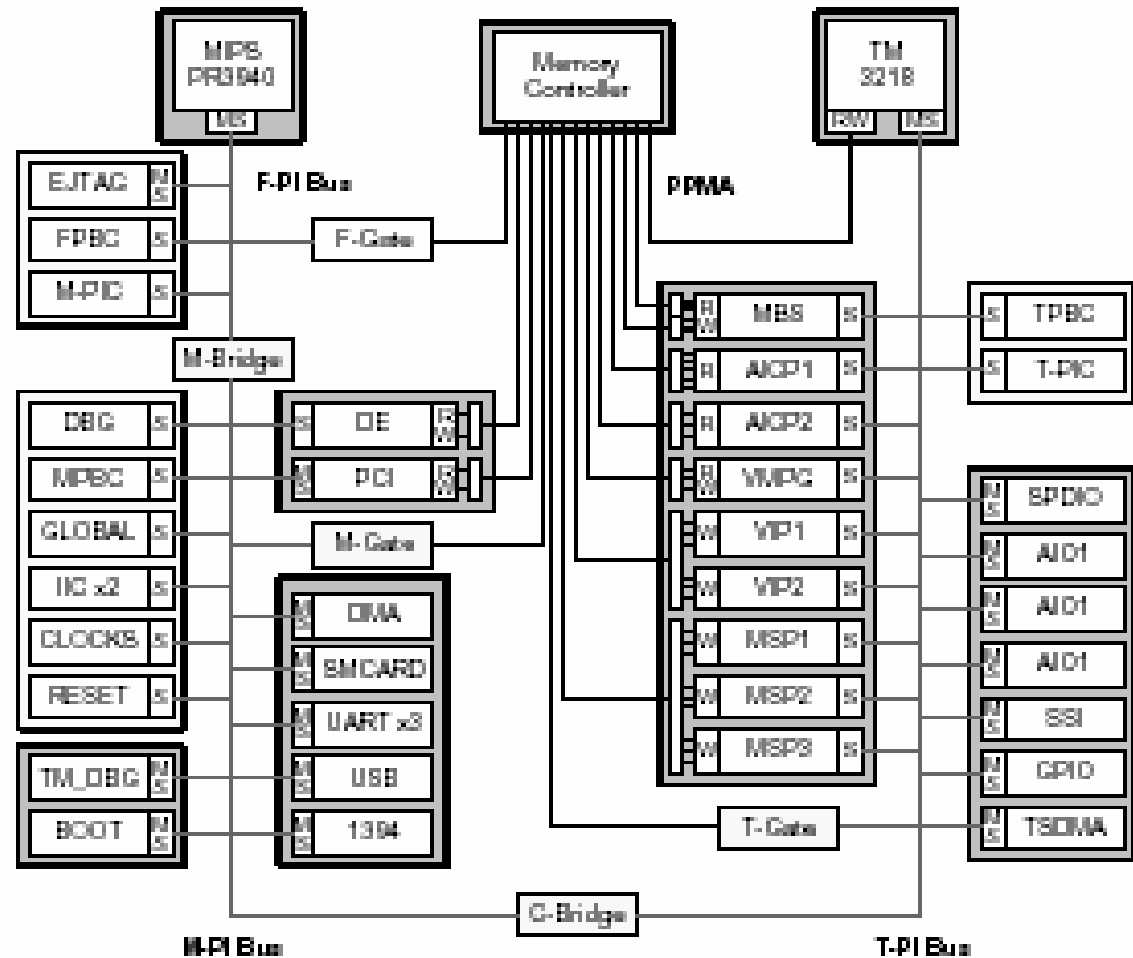
Philips Nexperia (Viper) SoC

Philips Research
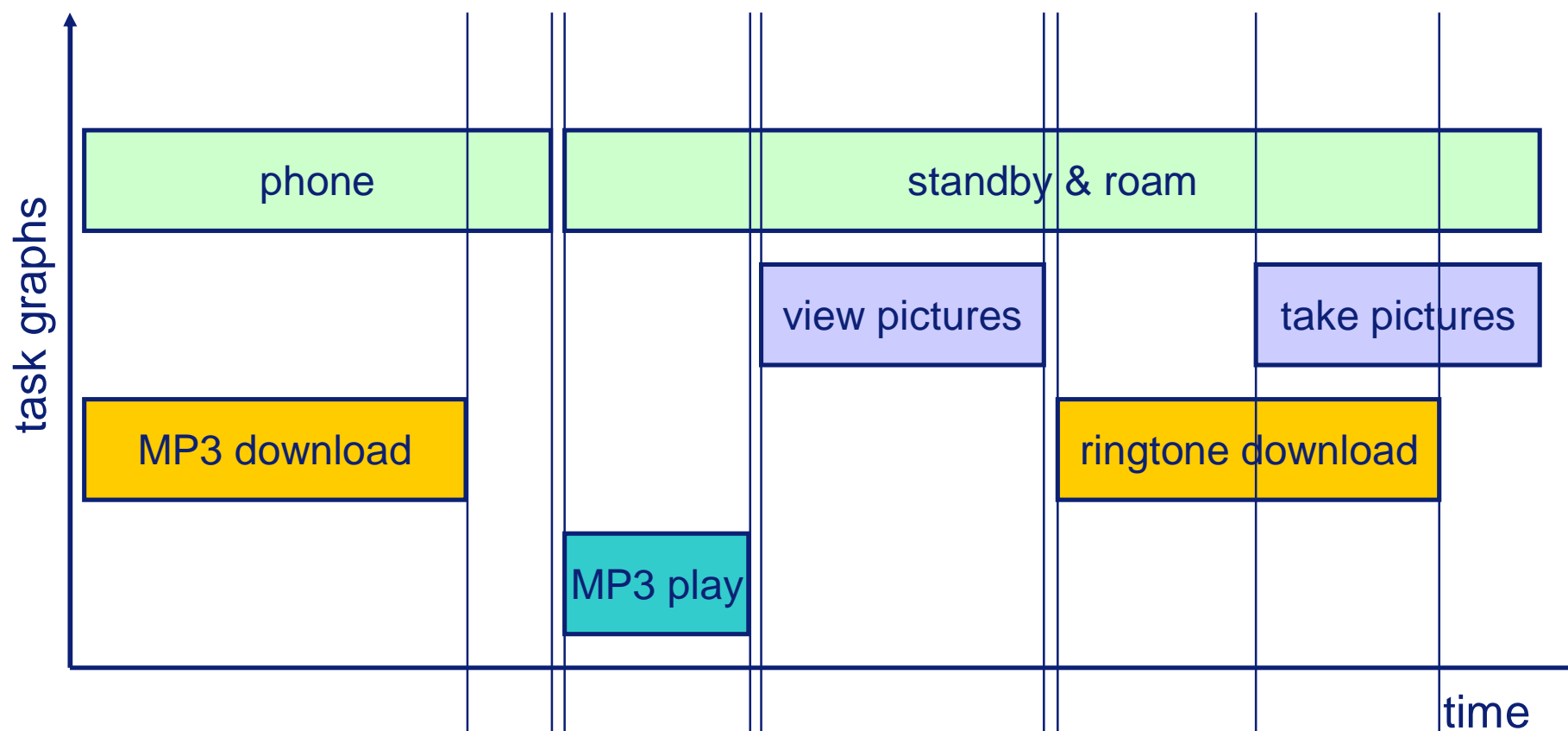
NoC Design Challenges

- Design Methods
  - Performance, design constraints
  - Minimum area, power overhead
- Several Phases
  - Capture application behavior
  - Find topology & map cores
  - Find paths for traffic flows
  - Set architectural parameters
  - Simulation, verification
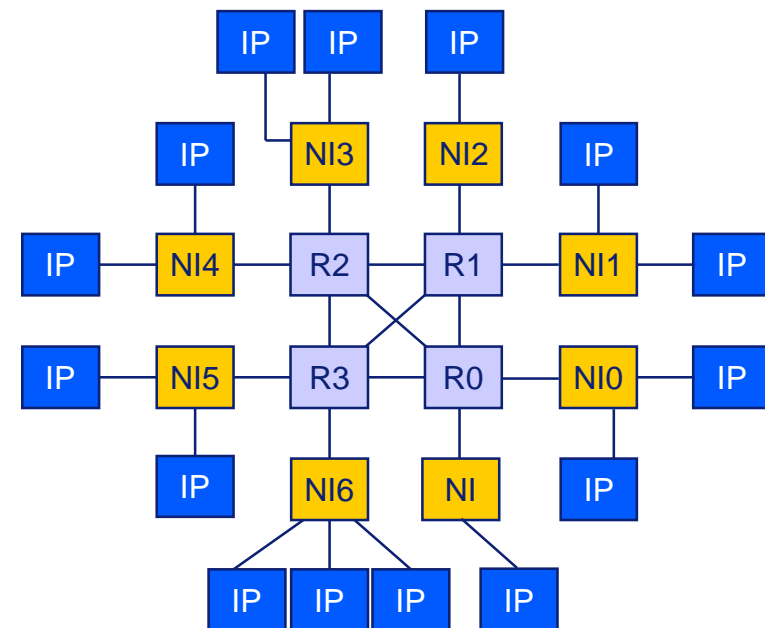
Integrated Approach is Key for Proper System Design

Philips Research

Motivating Example

- A single SoC supports large number of use-cases

task graphs

phone | standby & roam

view pictures | take pictures
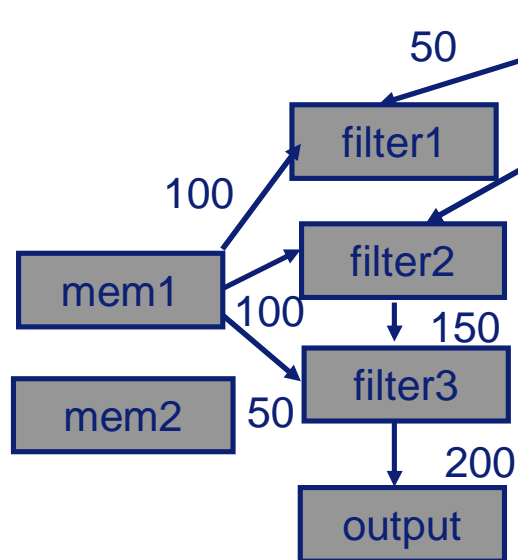
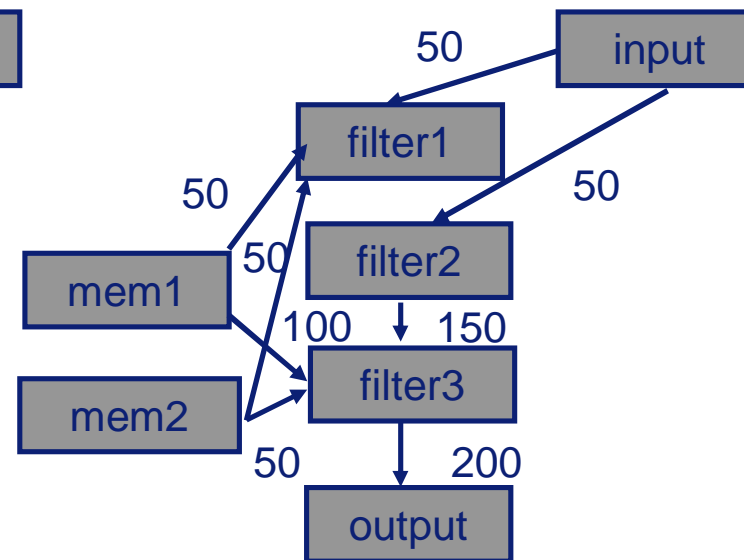MP3 download | ringtone download

MP3 play

time

c1　　　　　cadgroup, 2005/12/13

Multiple Use-Cases

- Communication characteristics differ over time
  - Each use-case has different traffic patterns
  - Different constraints: bandwidth, latency
- Small example from Philips Nexperia (Viper 2) set-top box
- Big challenge: Design one architecture to support all use-cases

Use-case 1

Use-case 2

Contributions of this work

- Design NoCs that satisfy multiple use-cases
  - Map cores onto topologies
  - Find paths, reserve time-slots (resources)
- Satisfy constraints of all use-cases
- Dynamically configure paths, resources across use-cases
- Dynamic voltage, frequency scaling to reduce NoC power consumption
- Integrate with existing Aetheral tool chain

Previous Work

- Several works on NoC mapping & topology design
  - Hu et al. (DATE '03), Murali et al. (DAC '04)
  - Hansson et al. (ISSS '05)
  - Pinto et al. (ICCD '04)
- Time-window based bus design, Murali et al. (DATE '05)
- All existing works consider only single application (trace driven)
- Few works on network re-configuration
  - MIT RAW (compiler assisted)
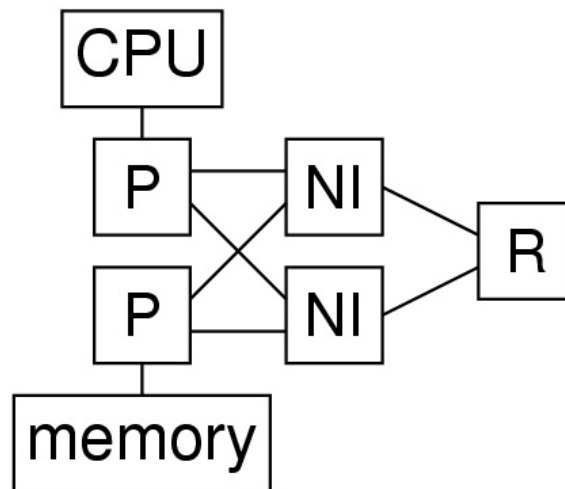 - Flexbus , Sekar et al.,(DAC '05)

Unified MApping, Routing and Slot allocation (UMARS)

- For single use-case
- key ideas
  - routing uses slot allocation algorithm to evaluate paths
  - mapping is implicitly determined during routing
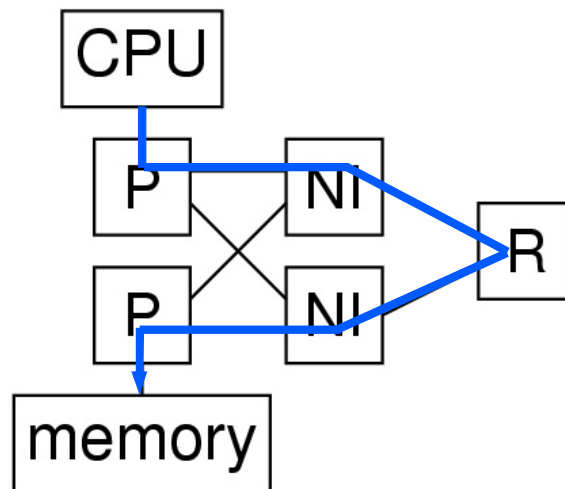- hierarchical decomposition
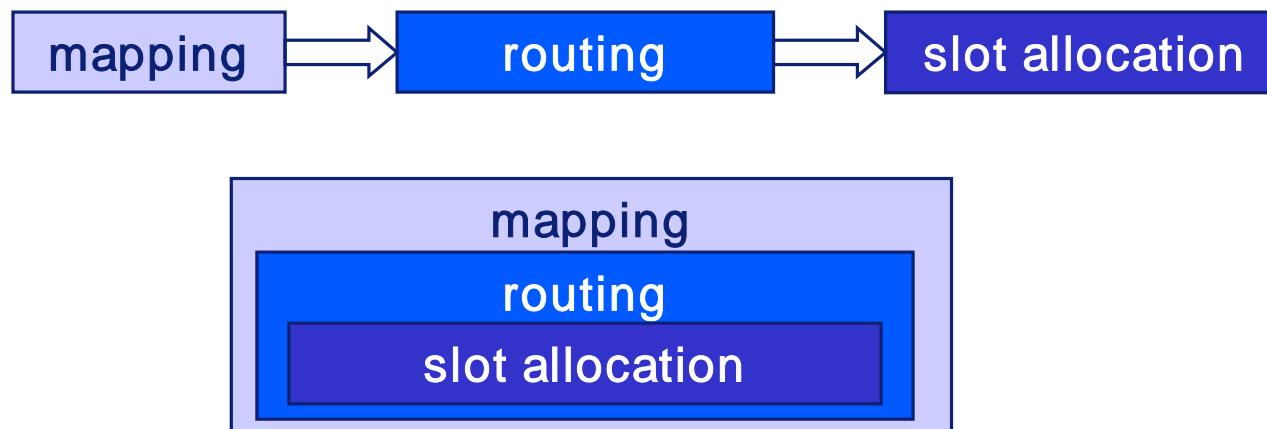
algorithm outline

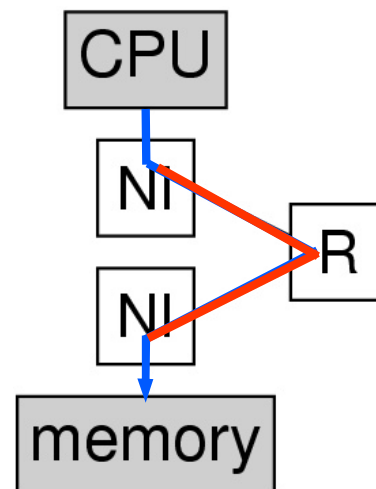- initially all cores are mapped to dummy mapping nodes (P)

algorithm outline

- initially all cores are mapped to dummy mapping nodes (P)
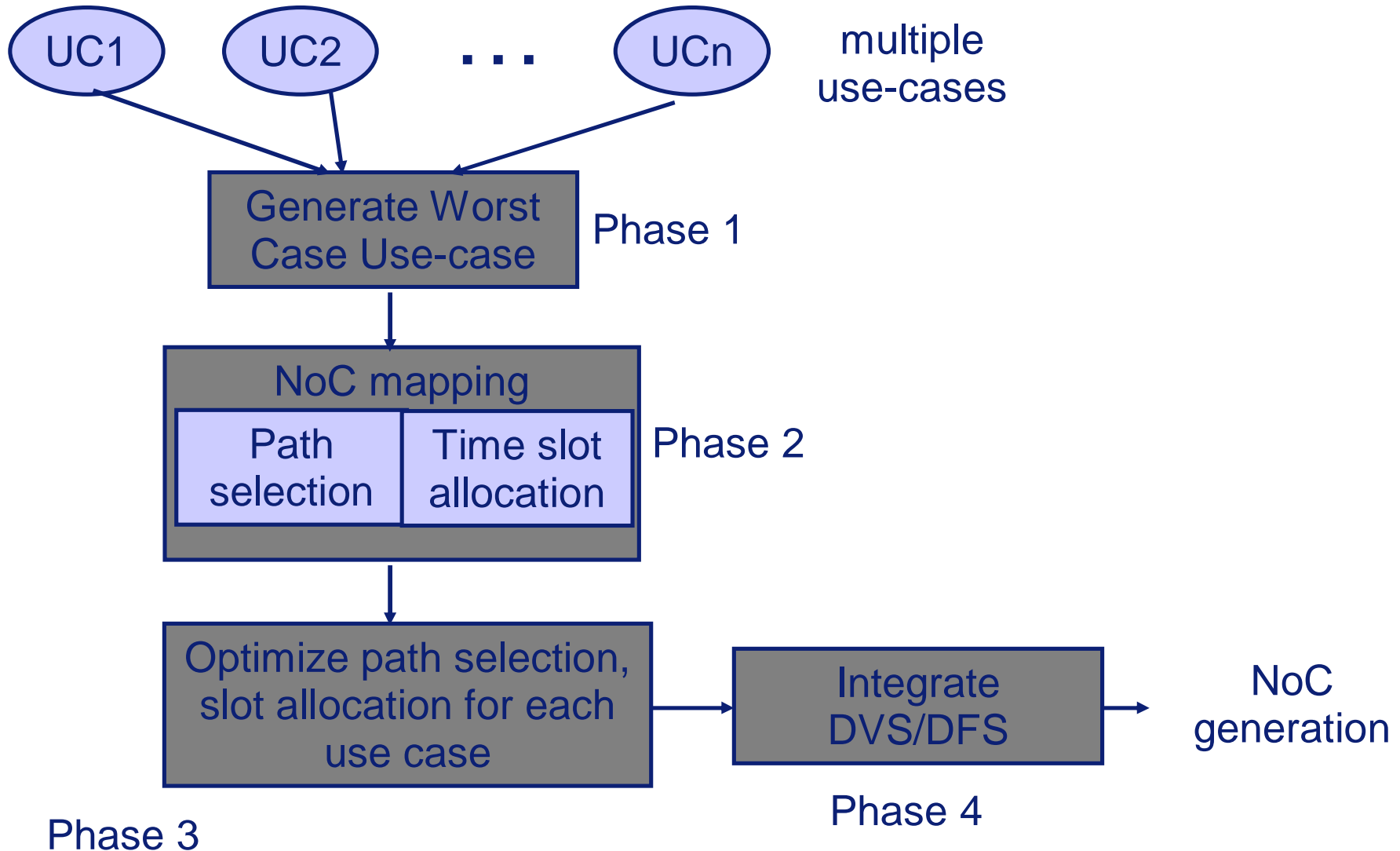- a route is selected for the next unallocated flow

algorithm outline

- initially all cores are mapped to dummy mapping nodes (P)
- a route is selected for the next unallocated flow
- this route determines also
  - the mapping (first & last link)
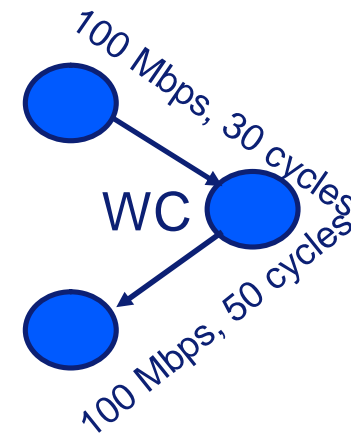 - the set of usable time-slots

"mapping" path
"normal" path

Philips Research

Multi Use-Case Mapping Design Flow

UC1   UC2   **. . .**   UCn   multiple use-cases

Generate Worst Case Use-case   Phase 1

NoC mapping
| Path selection | Time slot allocation |
Phase 2

Optimize path selection, slot allocation for each use case

Integrate DVS/DFS   NoC generation

Phase 3

Phase 4

Philips Research

Phases 1, 2: Generating Worst Case (WC) use-case

- Captures worst-case bandwidth, latency constraints
- For each communicating pair of cores
  – Take largest bandwidth value across all use-cases
  – Take the tightest latency constraint

UC1   100 Mbps, 50 cycles   10 Mbps, 50 cycles

UC2   50 Mbps, 30 cycles   100 Mbps, 100 cycles

WC   100 Mbps, 30 cycles   100 Mbps, 50 cycles
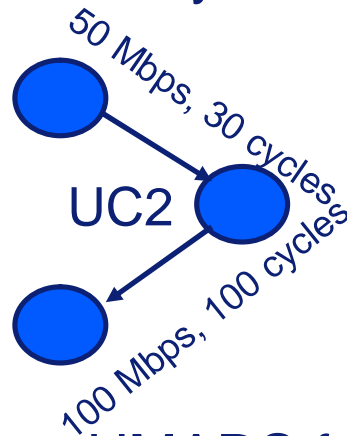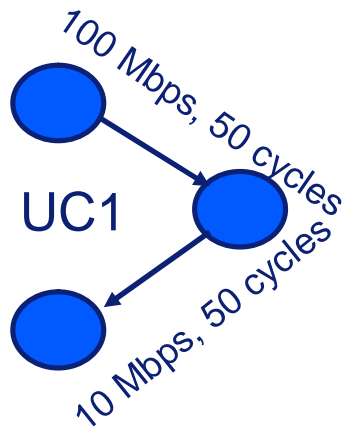
- Mapping, paths, slots: UMARS for WC use-case (phase 2)
- The design satisfies constraints of all use-cases
- However, network can be over-designed

Phase 3: Optimize path, slot tables

- Observation:
  - Each use-case only needs support for its flows
  - Bandwidth & slots allocated are more than needed
- Fix mapping from WC use-case
- Re-run the network configuration step for each use-case individually
- Two options:
  - Use different paths, slot-tables for each use-case
  - Same paths, slot-tables
- Find those use-cases that support NoC re-configuration

NoC re-configuration

Transitions between use-cases takes time depending on use-case
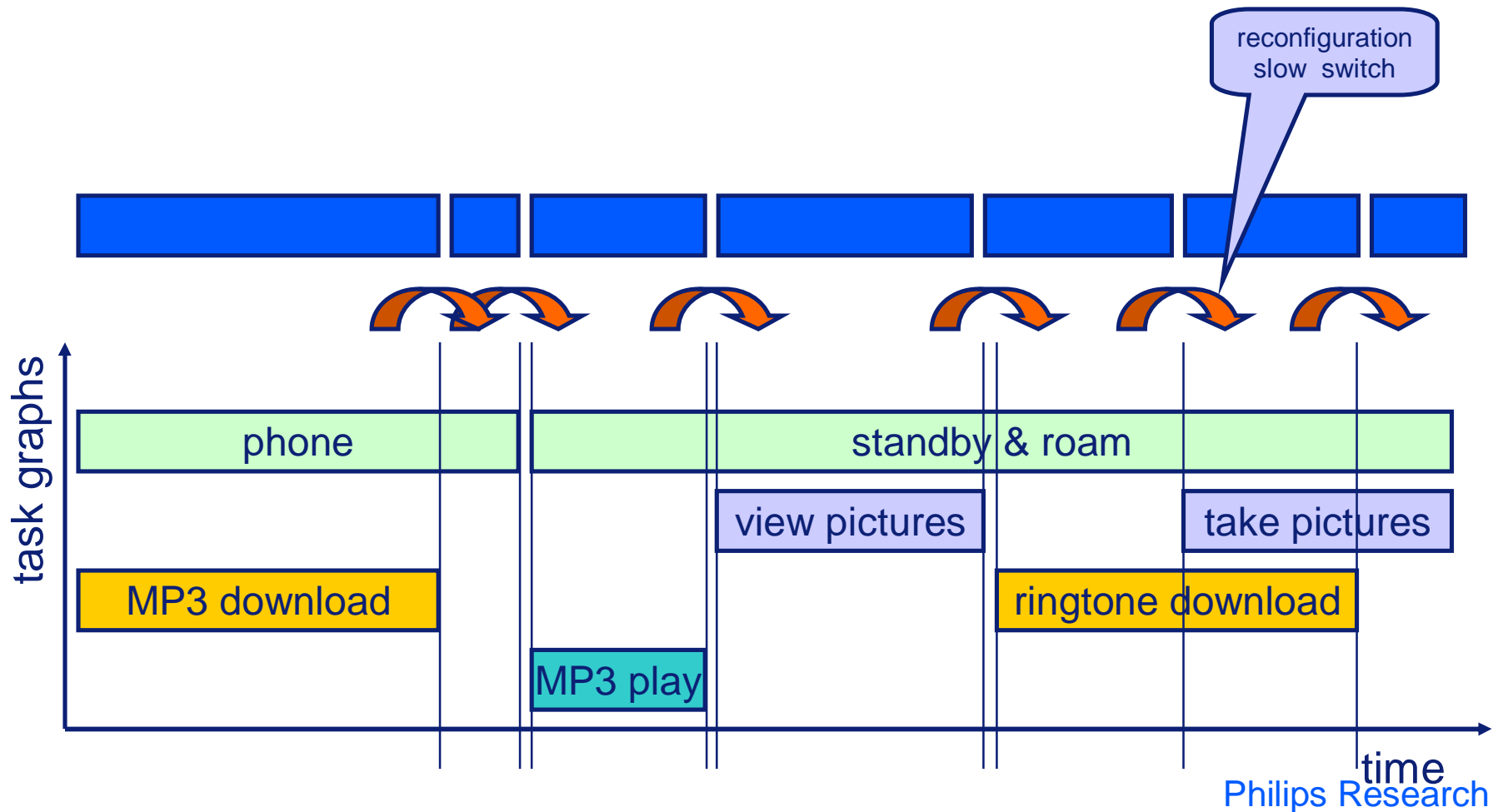
NoC Re-Configuration

- Use-case switching times depend on
  - Underlying architecture
  - Type of use-cases (critical or not), amount of data needed
- Switching times vary widely: from 100ns to hundreds of ms
- Three NoC configurations:
  - Smooth switching: use-cases use WC configuration
  - Switching time of micro-seconds: change paths, slot-tables
  - Switching time ms: change NoC frequency, voltage
- Paths, slot tables stored in external memory
- Overhead for re-configuration
  - Few KBs of memory (for 4 use-cases)
  - Micro-joules of energy, micro-seconds for loading data
  - Use network itself to spread the information

Mapping Results

	map one uc & configure each uc (1)	map & configure WC uc (2a)	map WC & configure each uc (2b)
convergence guaranteed	no	yes	yes
configure uc	sequential	same	sequential
# configurations	many	1	many
Use case switching	slow	fast between all uc	fast within WC slow between
work well for	Doesn't necessarily work	few or similar uc	large # of ucs
freq / power cost	-	high	low
area cost	-	high	medium

Philips Research

Simulation Results

Experimental Benchmarks

- Applied to two real SoC systems (simplified versions)
  - Viper Set-top box SoC (2 designs with 2, 1 with 4 use-cases)
  - In-house TV processor (8 use-cases)
- Communication characteristics very different in the two
- Viper uses single shared memory (bottleneck communication)
- TV processor uses multiple local memories (spread communication)
- Such varied examples to show the generality of the methods

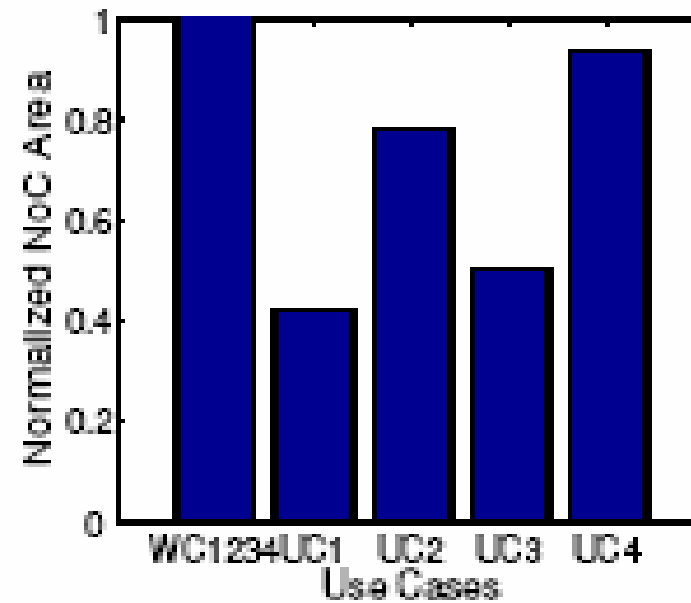# Comparisons with WC use-case
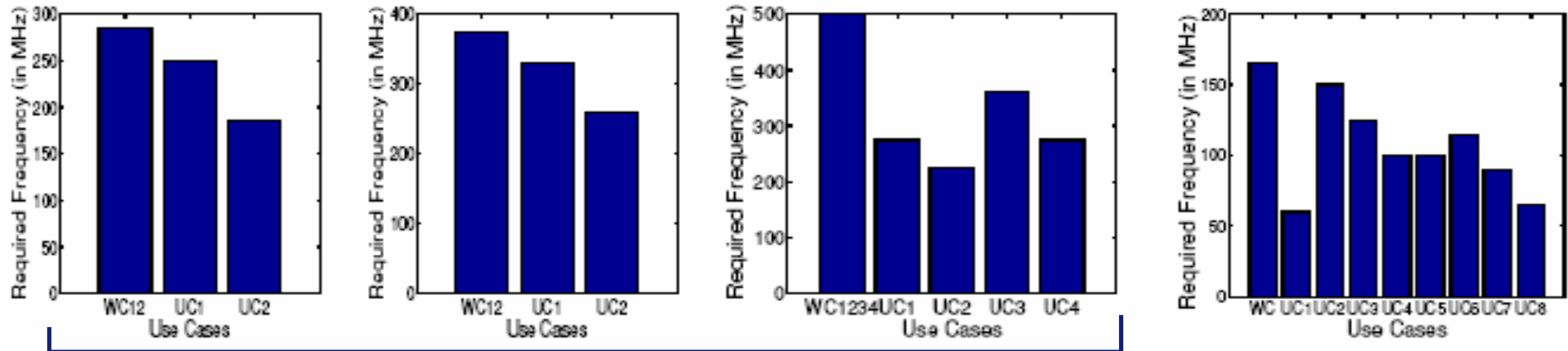
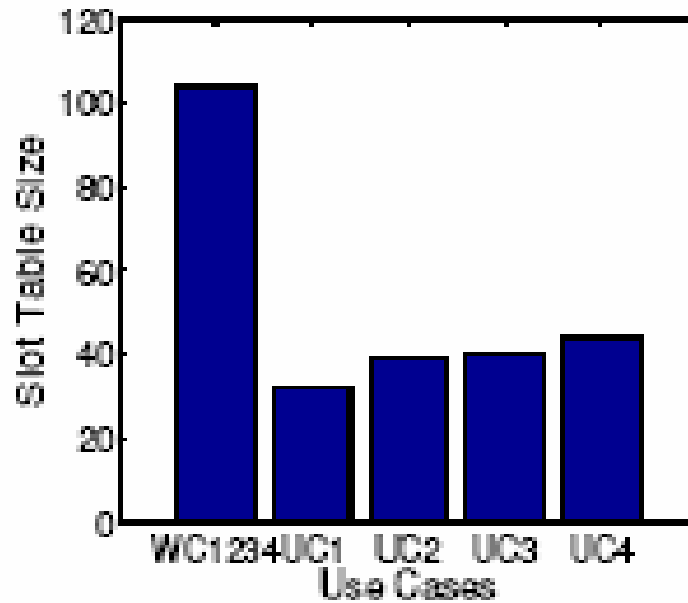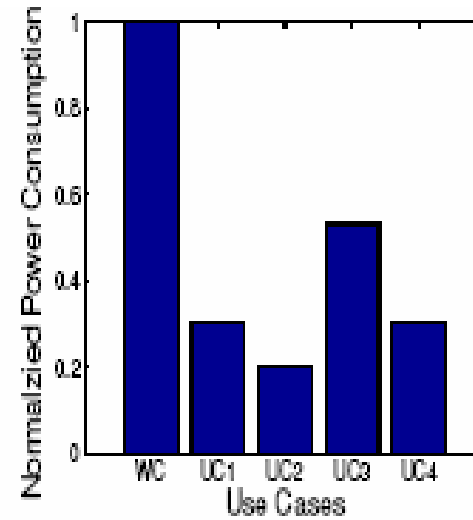Viper SoC designs   TV processor

- With re-configuration, frequency is maximum of the individual use-cases

- Re-configuration results in 9% to 38% reduction in required frequency

- Results are within 10% of the minimum possible frequencies, when each use-case has its own best mapping
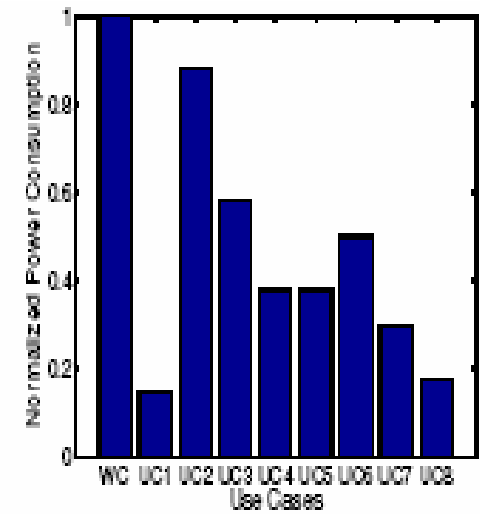
Area, Slot-table Size for Viper

- Re-applying path, slot-table allocation leads up to to 58% reduction in slots

- Area savings of 10% due to reduction in slots

Dynamic Frequency and Voltage Scaling
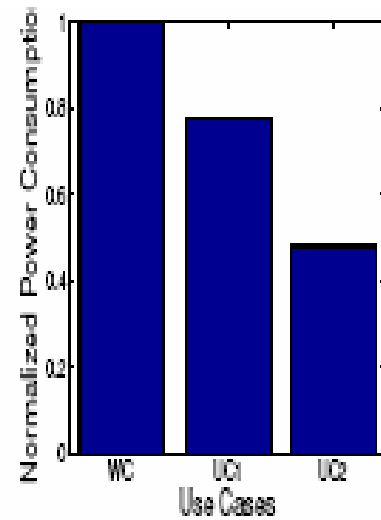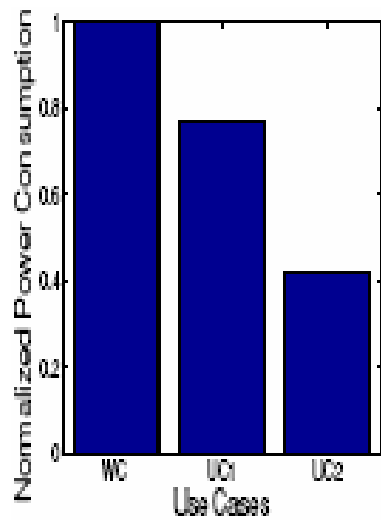
Viper SoC designs

TV processor

- •Apply DVS/DFS to match the frequency needs of use-cases
- •On average leads to 59.2% reduction in power consumption

Philips Research

Conclusions & Future Work

- Designing NoCs that multiple applications or use-cases
  - Realistic problem
  - Non-trivial to obtain a design that works well for all use-cases
- We presented methods to design NoCs to support multiple use-cases
  - Least cost network
  - Satisfy all design (bandwidth, latency) constraints
- Explored mechanisms for re-configuring paths, time slots
- Explored DVS/DFS

- Future work
  - Look at dynamic mapping
  - Topology synthesis

Philips Research

Philips Research