Shared-clock methodology for time-triggered multi-cores

Keith F. Athaide

Project supervisor: Michael J. Pont
Technical supervisor: Devaraj Ayavoo

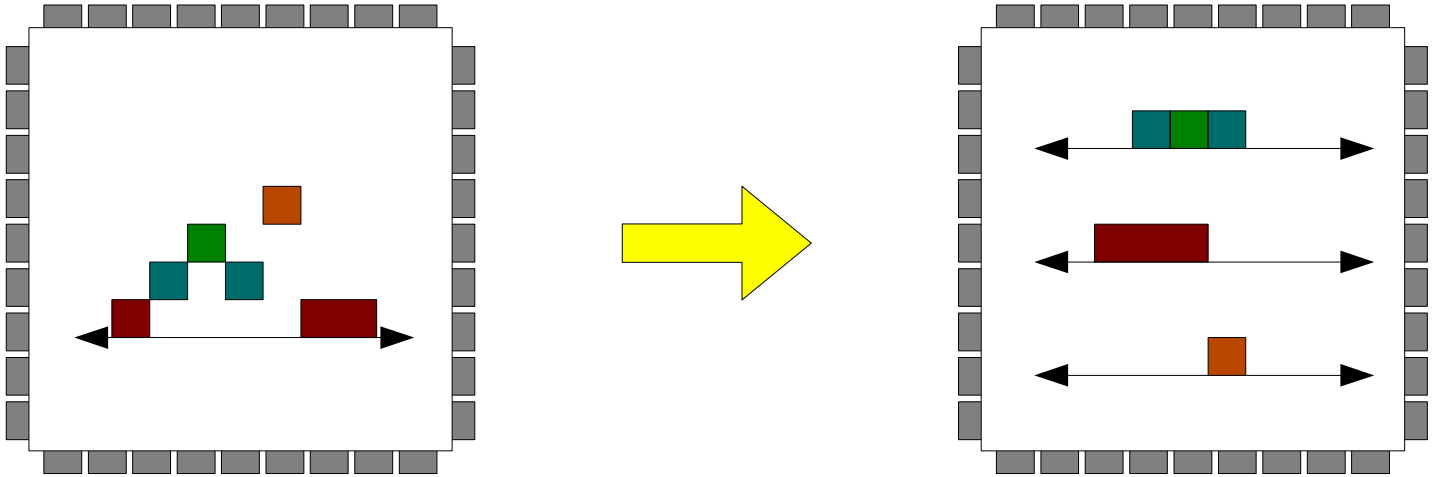
Communicating Process Architectures (CPA) 2008
8th-10th September 2008

Overview

- Aims
- Execution policies
  - Co-operative
  - Pre-emptive
- Execution architectures
- Shared-clock architecture
  - Algorithm for non-broadcast topologies
- Multi-processor microcontroller architecture
- Case study description
- Results
- Conclusions

Aims

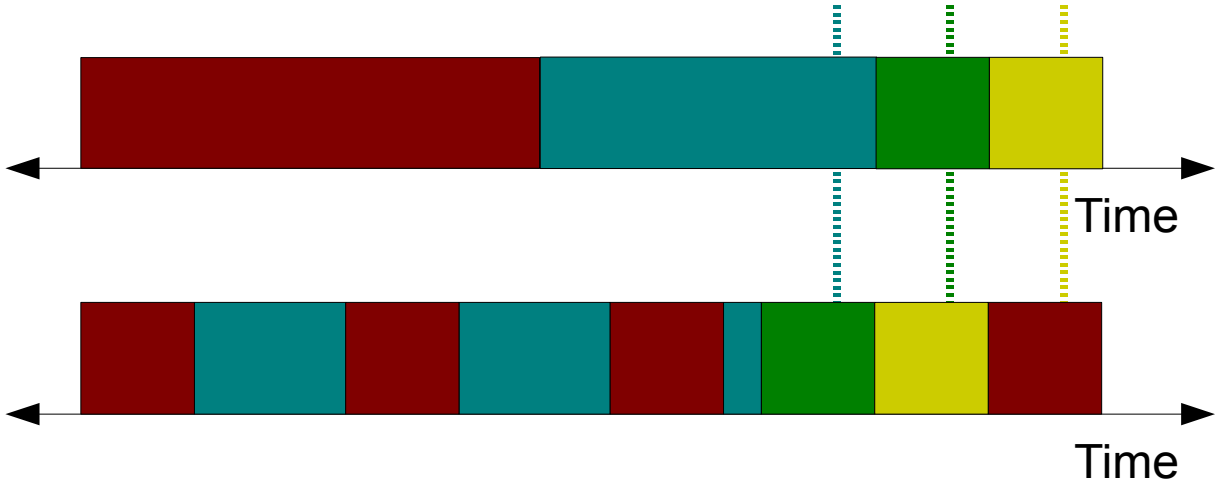
- Maintain the predictability and robustness of co-operative single-processor systems
  - Custom system-on-chip (SoC)
  - Time-triggered applications
- Heterogeneous processors
- How to synchronise the different processors?



Execution policy

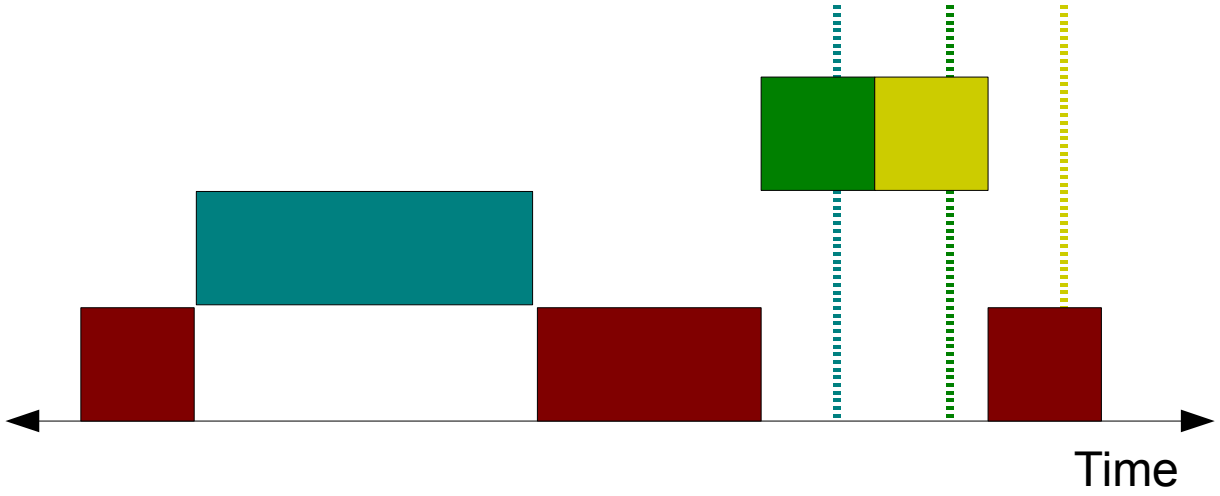
- System has many functions
- Functions often decomposed into discretely executing blocks called tasks
  - Periodic or aperiodic tasks
- Periodic tasks may have static or dynamic periods
- Tasks have deadlines
- Tasks are executed according to a policy
  - Co-operative execution policy
  - Pre-emptive execution policy

Co-operative execution policy



- Tasks must yield control when required
- Resource sharing needs no complex locking mechanisms
  - Same processor, one execution thread
- System responsiveness inversely related to longest task execution time

Pre-emptive execution policy

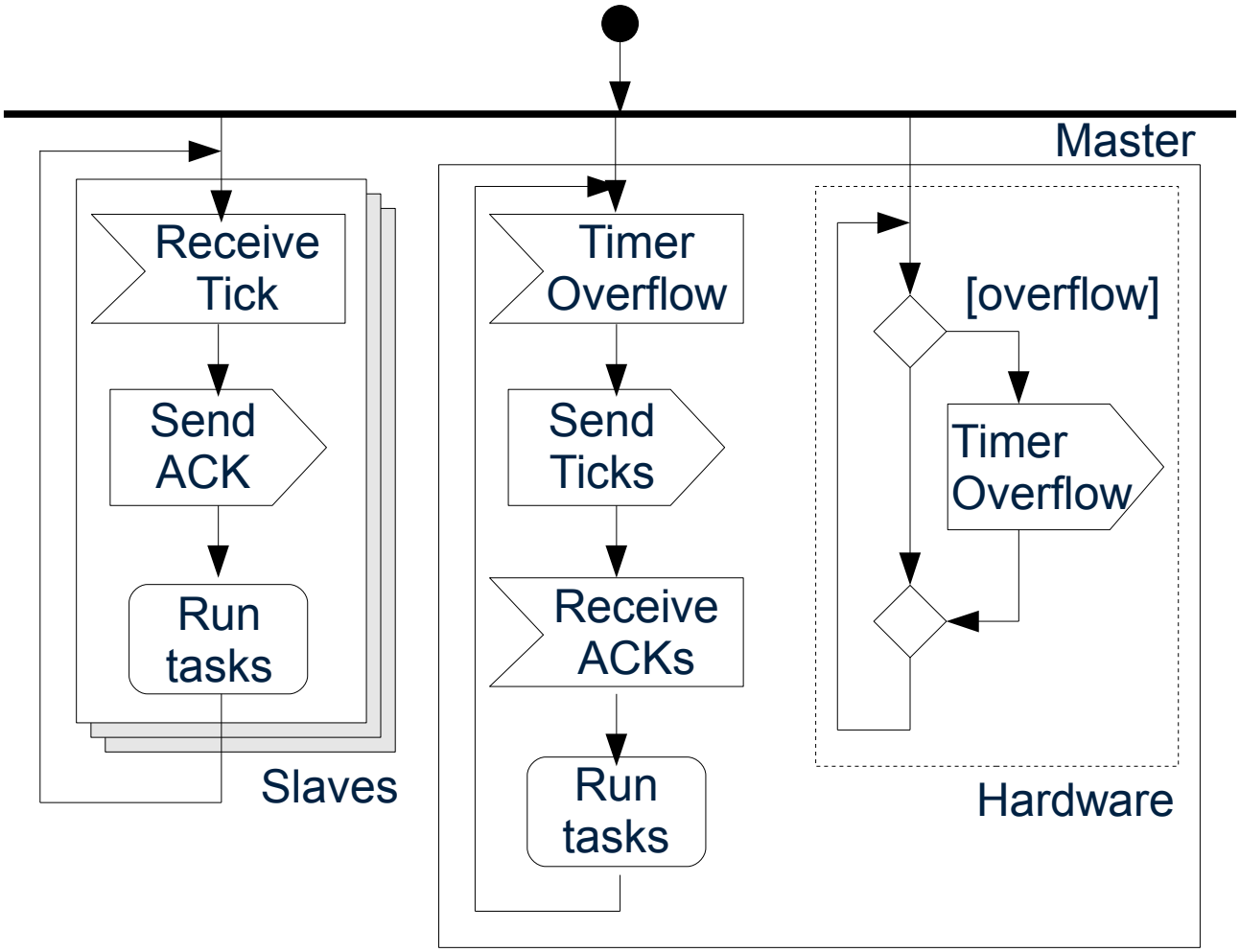


- Tasks can interrupt each other
- Interruption controlled by priorities
- Predictability dependent on uniformity in pre-empting instructions
- Problems such as priority inversion

Scheduler architectures

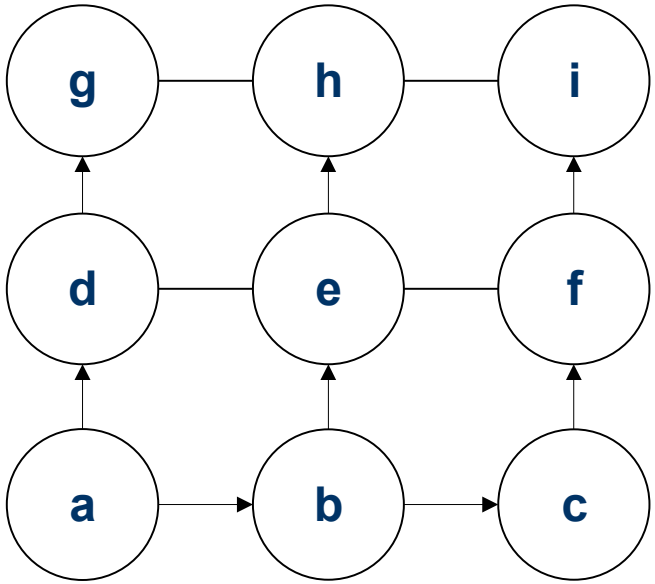
- Event-triggered
  - Multiple events
  - Feasibility depends on
    - the number of events expected
    - the number of events serviceable by hardware
  - “Construct by correction”
- Time-triggered
  - Single event
  - Other events sensed by polling
  - “Correct by construction”
  - Can be power hungry

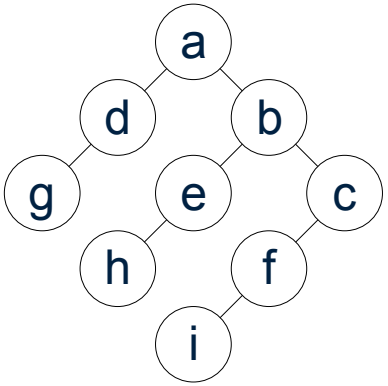
Shared-clock architecture



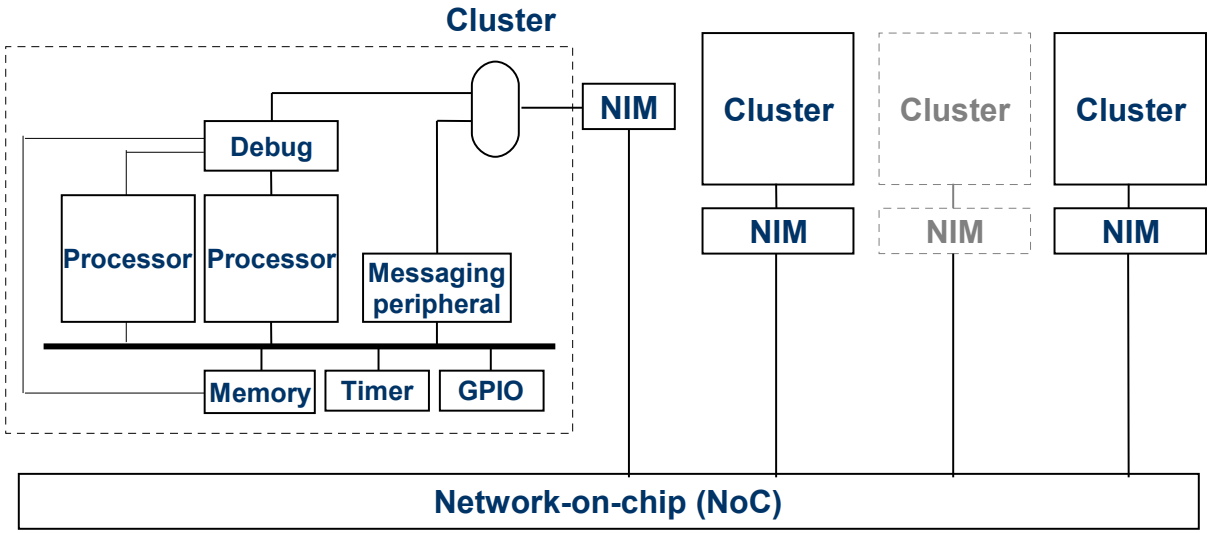
Shared-clock non-broadcast topology

- Existing implementations need communication topologies supporting broadcasts
  - Buses like CAN
- Can be simulated by point-to-point transmissions
  - Hardware or software
- Tree broadcast
  - MPI collective communication algorithm
- Lag due to point-to-point transmissions





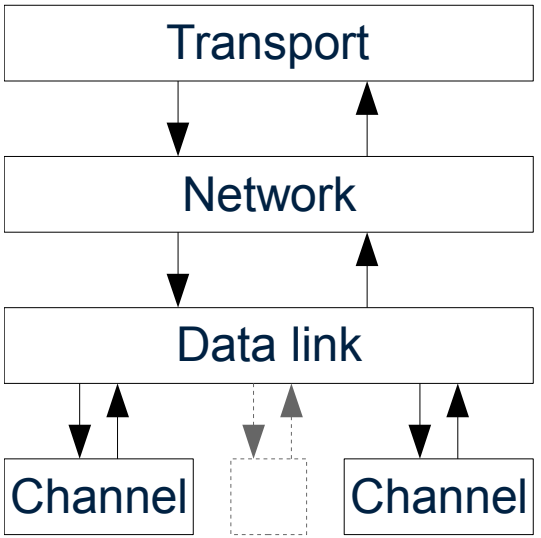
Multiprocessor architecture



- Network Interface Module (NIM)
  - Messaging component as peripheral or co-processor
- Debug cluster
  - Write to memories
  - Set breakpoints, stepping, etc.

Network interface modules (NIMs)

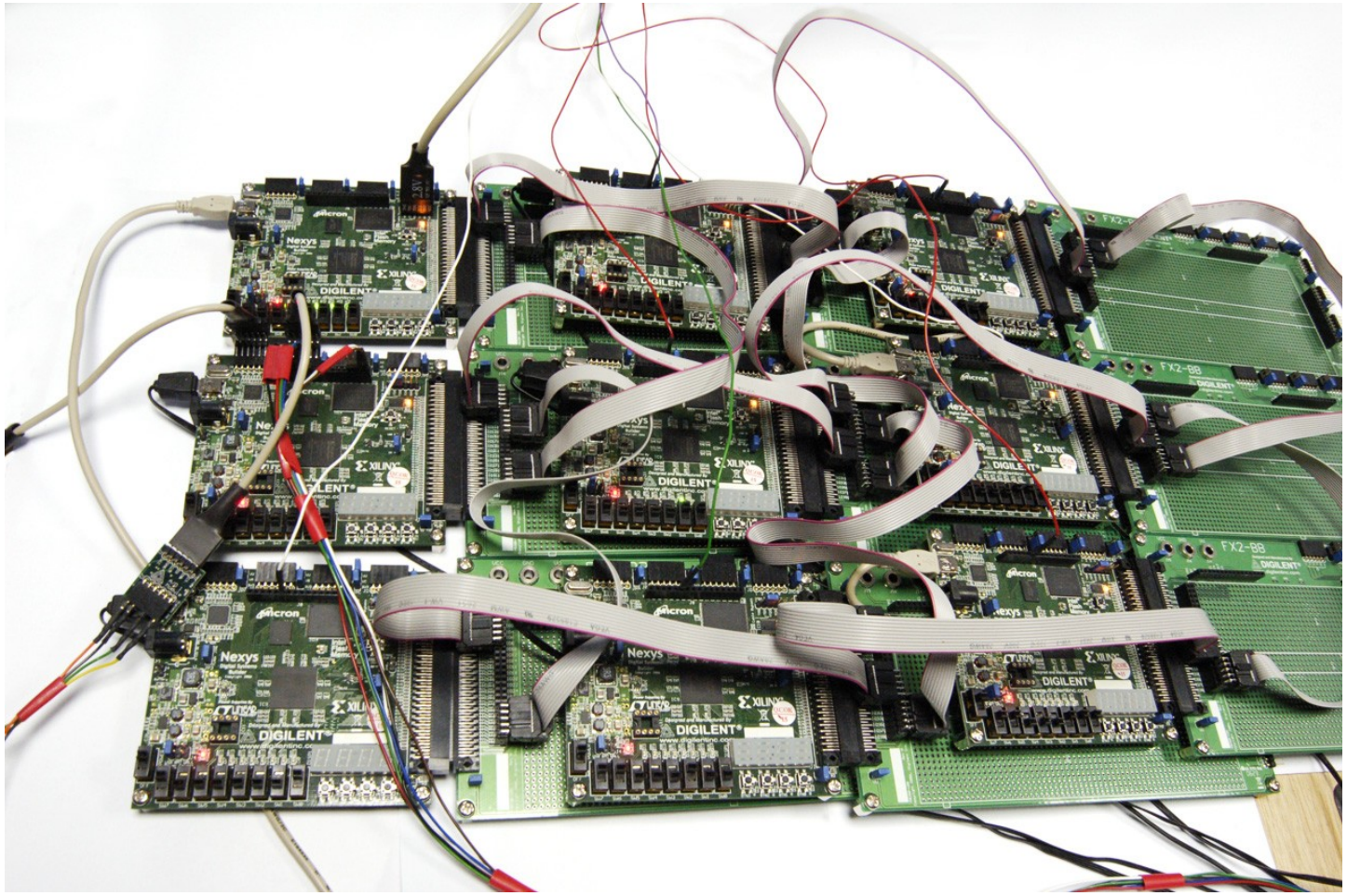
- Asynchronous communication
- Error detection
  - 12-bit checksums (CRCs)
- No automatic error correction
  - Errors cause no extra communication
  - Software notes and corrects errors
- Static routing
- Serial-parallel communication
- Variable number of channels
- Lack of predictability in communication latency might affect overall predictability of the shared-clock system



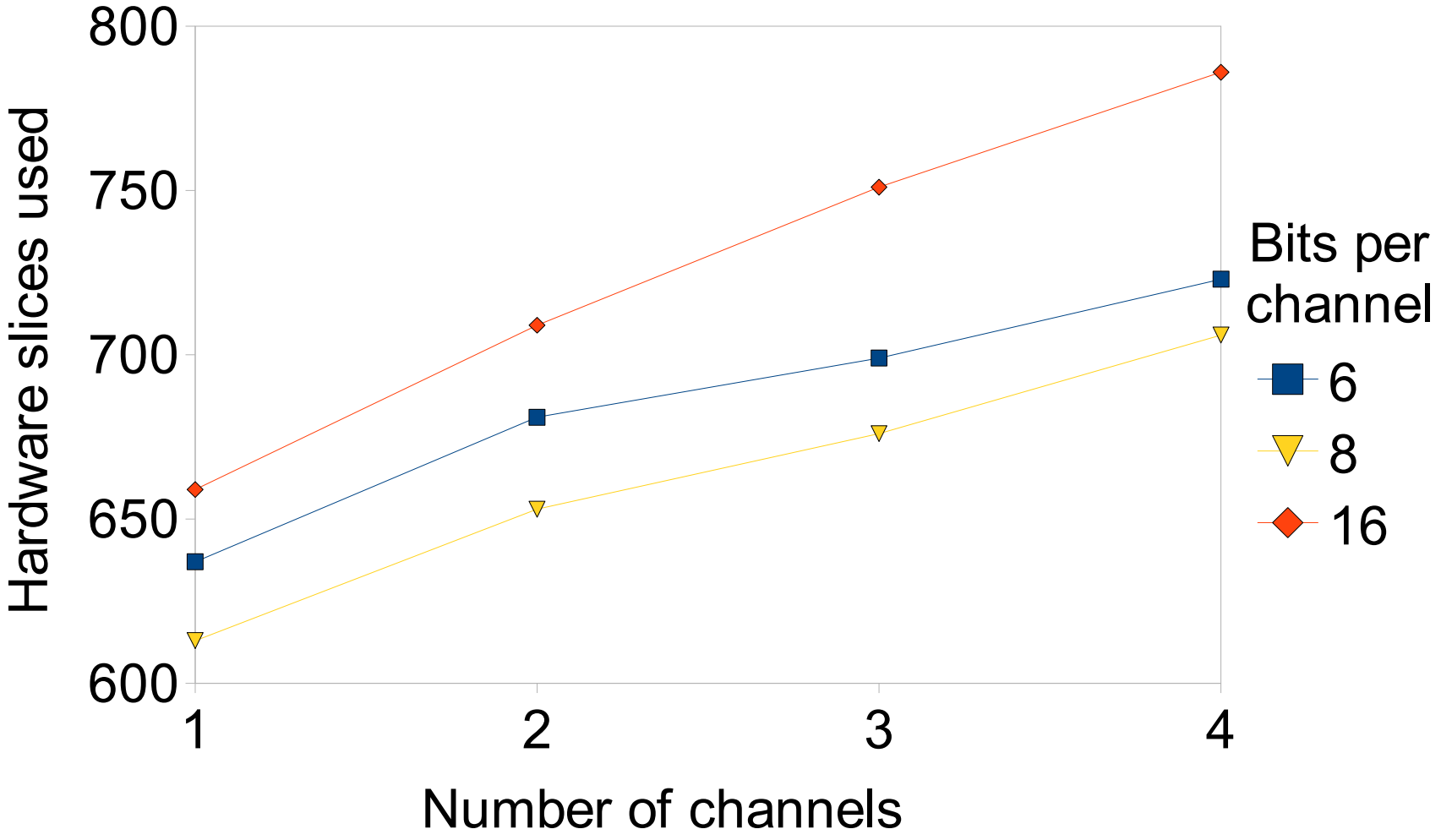
PH Processor

- Single interrupt
  - Built for time-triggered applications
  - Multiplexed from any number of sources
- Soft-core processor (VHDL source available)
- 32-bit reduced instruction set computer (RISC)
- MIPS I ISA (excluding patented instructions)
- Harvard architecture
- 32 registers
- 5-stage pipeline

Hardware implementation

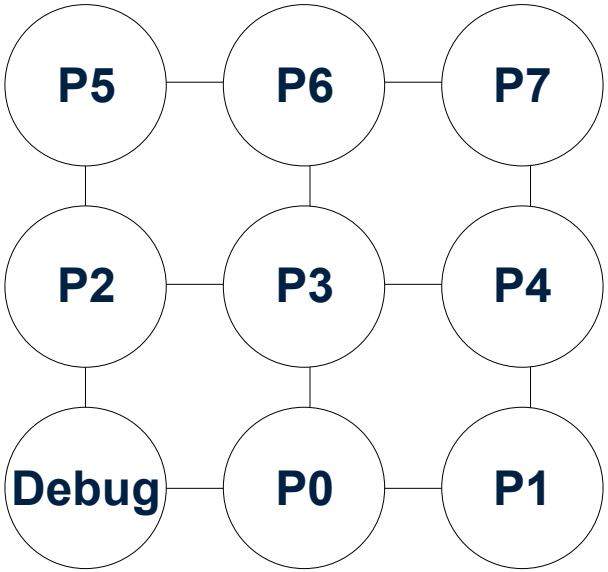


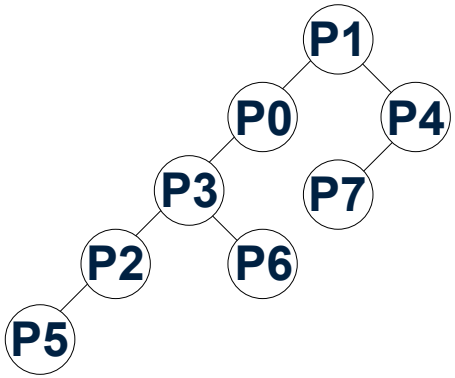
Hardware usage of NIMs



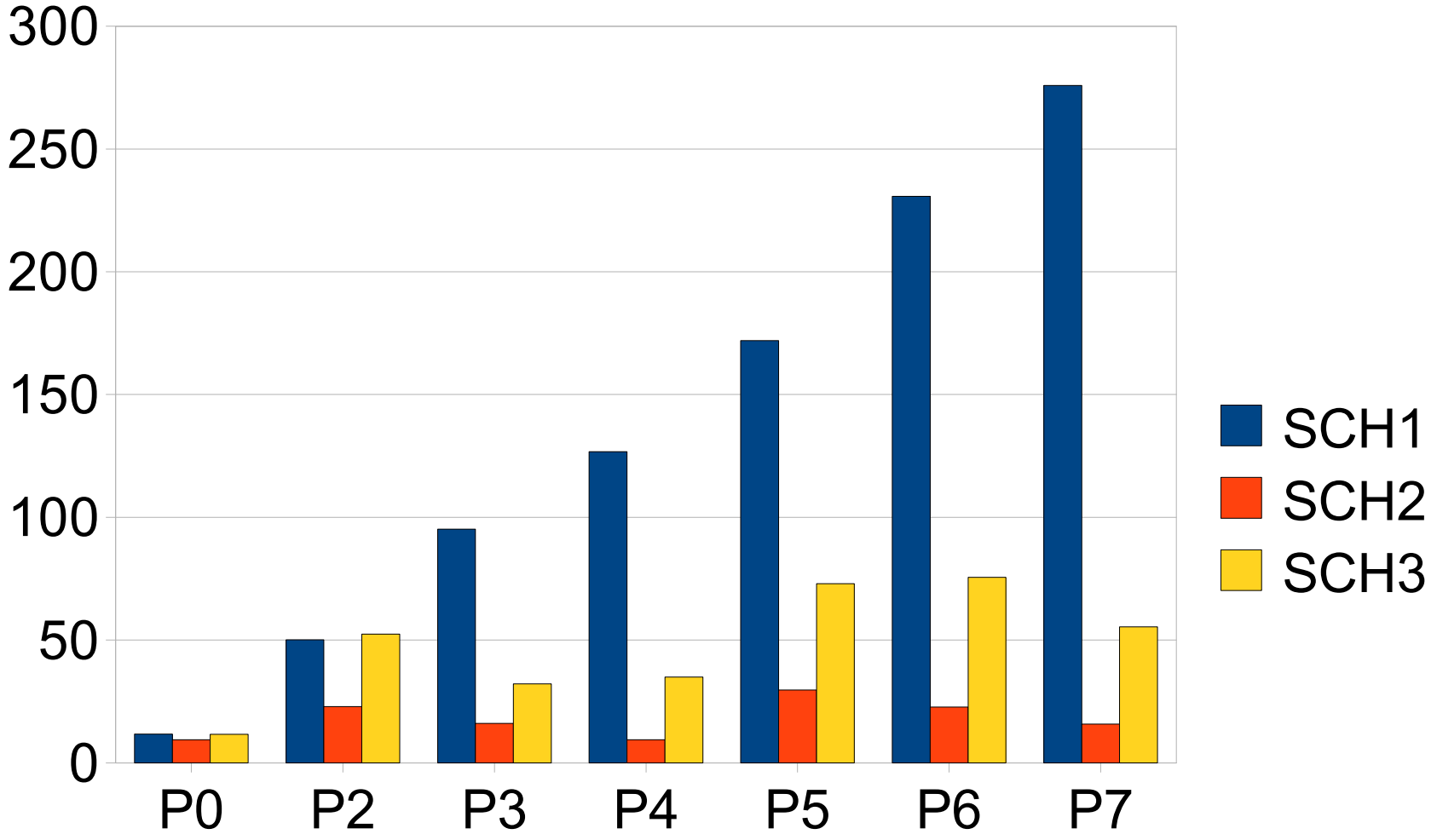
Case study description

- Nine nodes
  - Mesh topology
- Three scheduler types
  - **SCH1**: P1 as master; P1 sends Ticks only when previous is acknowledged
  - **SCH2**: P1 as master; P1 sends Ticks in turn
  - **SCH3**: Tree broadcast
- Relative times measured

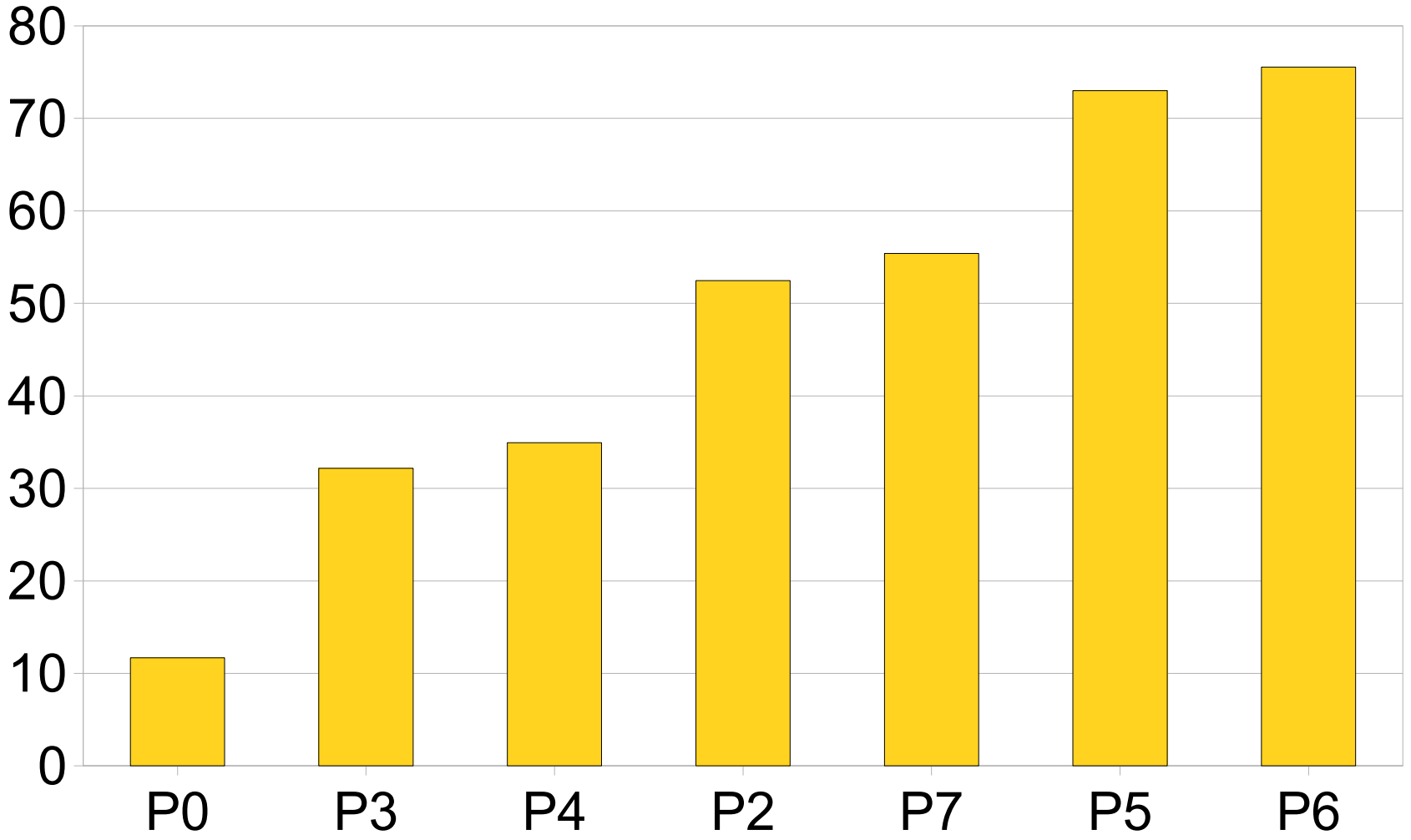




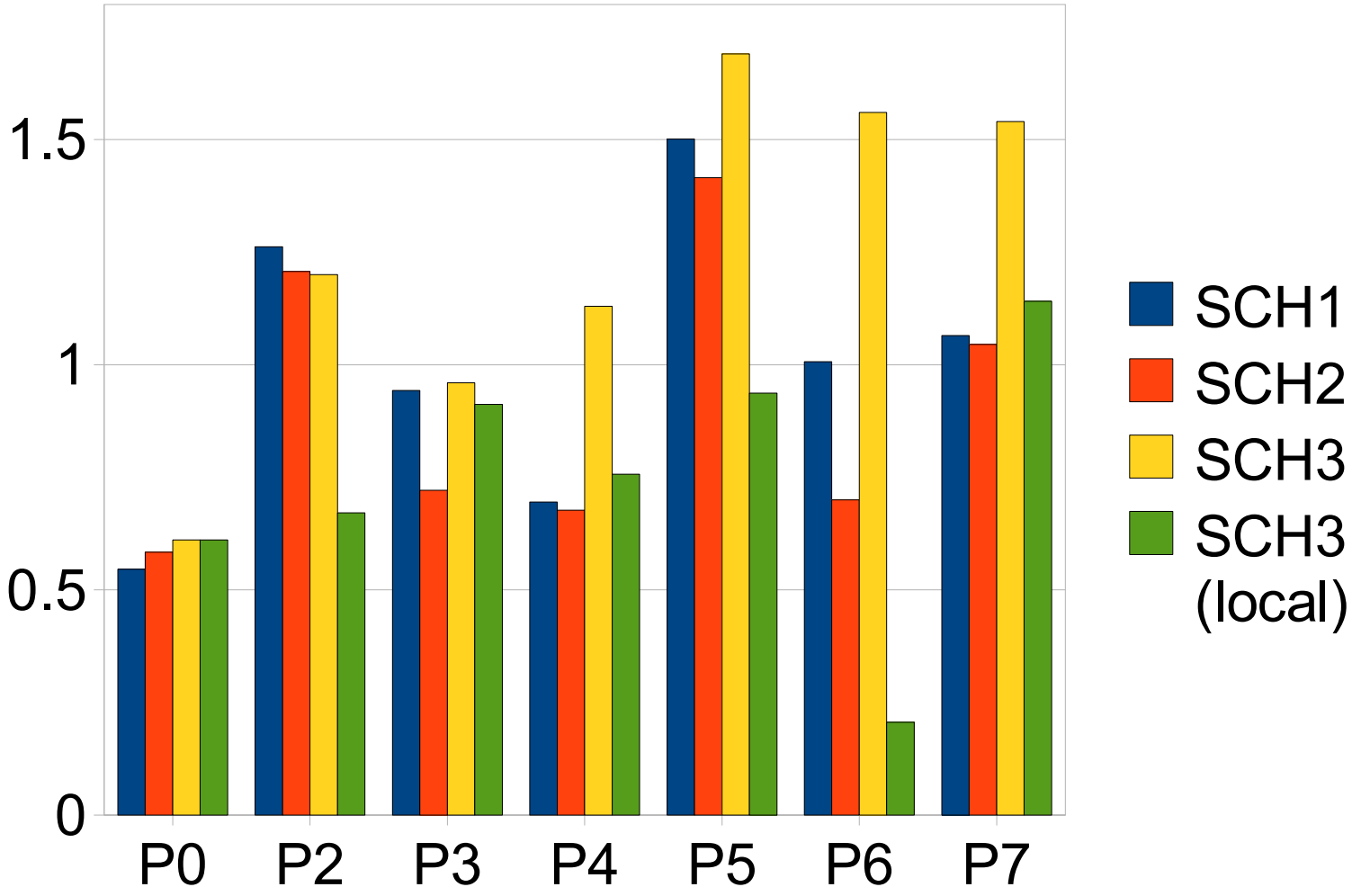
Timer sense times (microseconds)



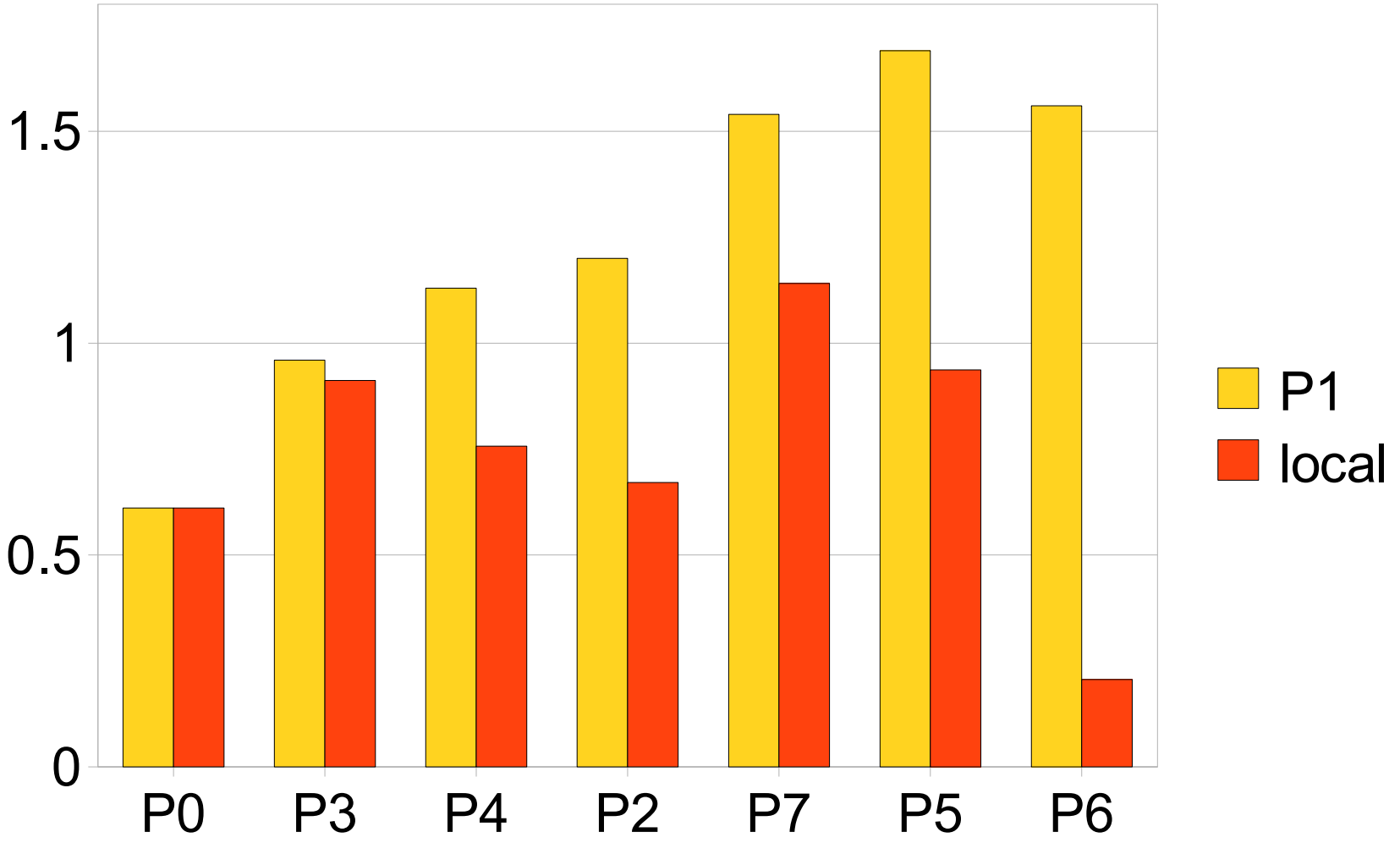
Timer sense times for SCH3 (microseconds)



Timer sense time jitter (microseconds)



Timer sense time jitter in SCH3 (microseconds)



Conclusions

- A custom multiprocessor microcontroller was developed for time-triggered applications
- The shared-clock protocol was employed on a 9 node mesh version of this microcontroller using a broadcast simulation algorithm
- Absorption of the broadcast simulation algorithm into software allows the node sending the ticks to worry only about the ones it is connected to – a **scalable** situation
- The delay and jitter in SCH3 could be improved