Unified Design of Hardware and Software Components

Tiago Rogério Mück

Current scenario

- System-level design requires a unified design approach for hardware and software

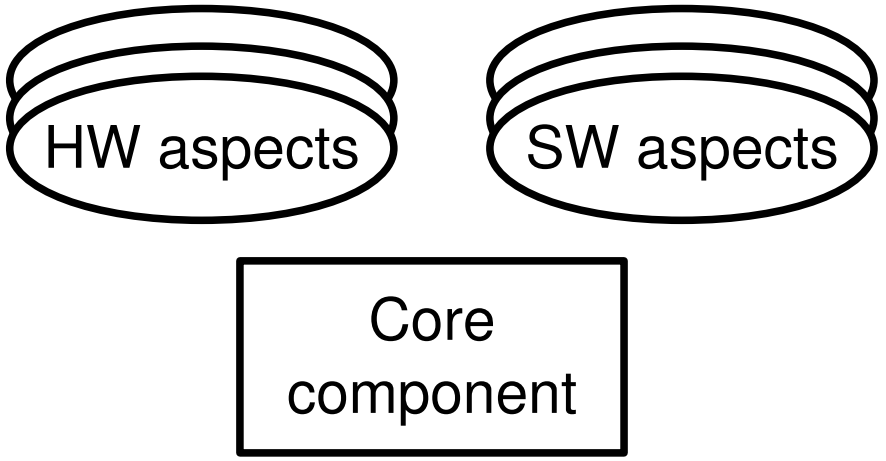
- State-of-art EDA tools already enable hardware synthesis from software-like implementation in C/C++
  - Such tools focus only in hardware synthesis
  - A clear methodology to synthesize both hardware and software from the same description is still missing

Goal

- Providing design guidelines to implement both hardware and software from a single C++ description

Proposed approach

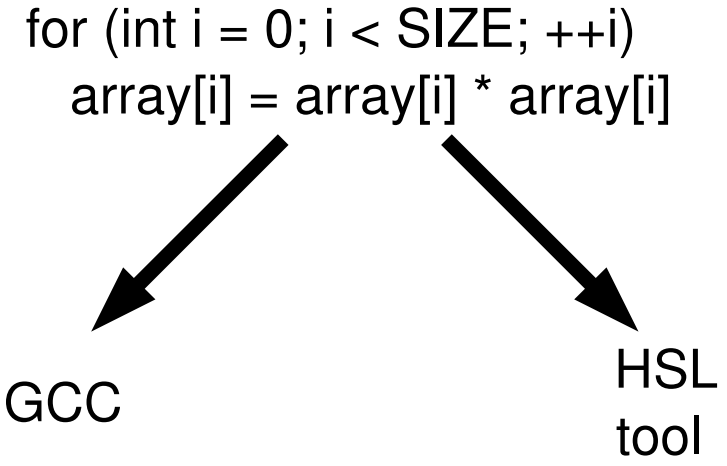
- Separate the core implementation of a component from *aspects* which are specific of hardware and software



- Adapt the component to the target scenario only during the final system generation

Hardware and software aspects

- Implementation of core behavior not differ significantly between hardware and software



- However, there are differences at structural level
  - Communication interface
  - Resource allocation

Communication interface

- Hardware and software have different communication patterns

Communication interface

- Hardware and software have different communication patterns
  - SW → SW
    - Function/Method call interface

Communication interface

- Hardware and software have different communication patterns
  - SW → SW
    - Function/Method call interface
  - HW → HW
    - Signals/Handshaking

Communication interface

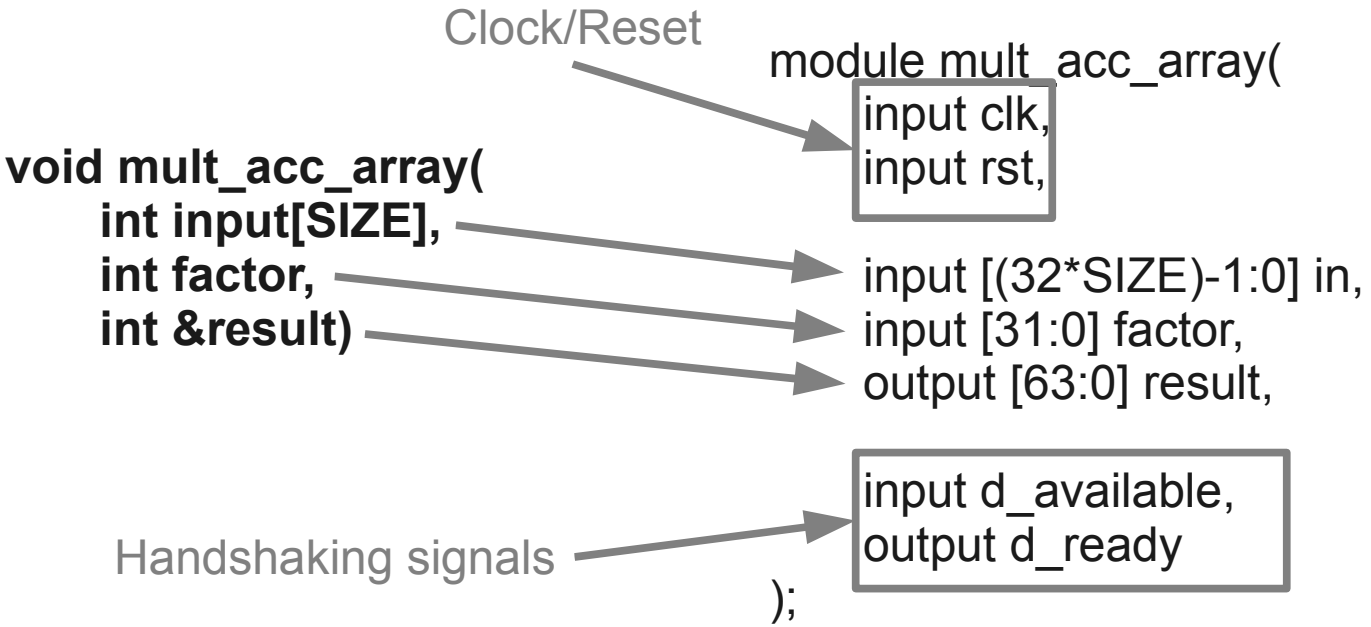
- Hardware and software have different communication patterns
  - SW → SW
    - Function/Method call interface
  - HW → HW
    - Signals/Handshaking
  - SW → HW
    - HAL sends commands using a communication infrastructures (e.g. a bus or a NoC). HW parses commands and triggers the operations

Communication interface

- Hardware and software have different communication patterns
  - SW → SW
    - Function/Method call interface
  - HW → HW
    - Signals/Handshaking
  - SW → HW
    - HAL sends commands using a communication infrastructures (e.g. a bus or a NoC). HW parses commands and triggers the operations
  - HW → SW
    - HW interrupts software. An ISR calls the requested operations and then signalizes hardware

HW interfaces from C/C++

- HLS tools require the definition of a single entry point for the component (e.g. a single function or method)

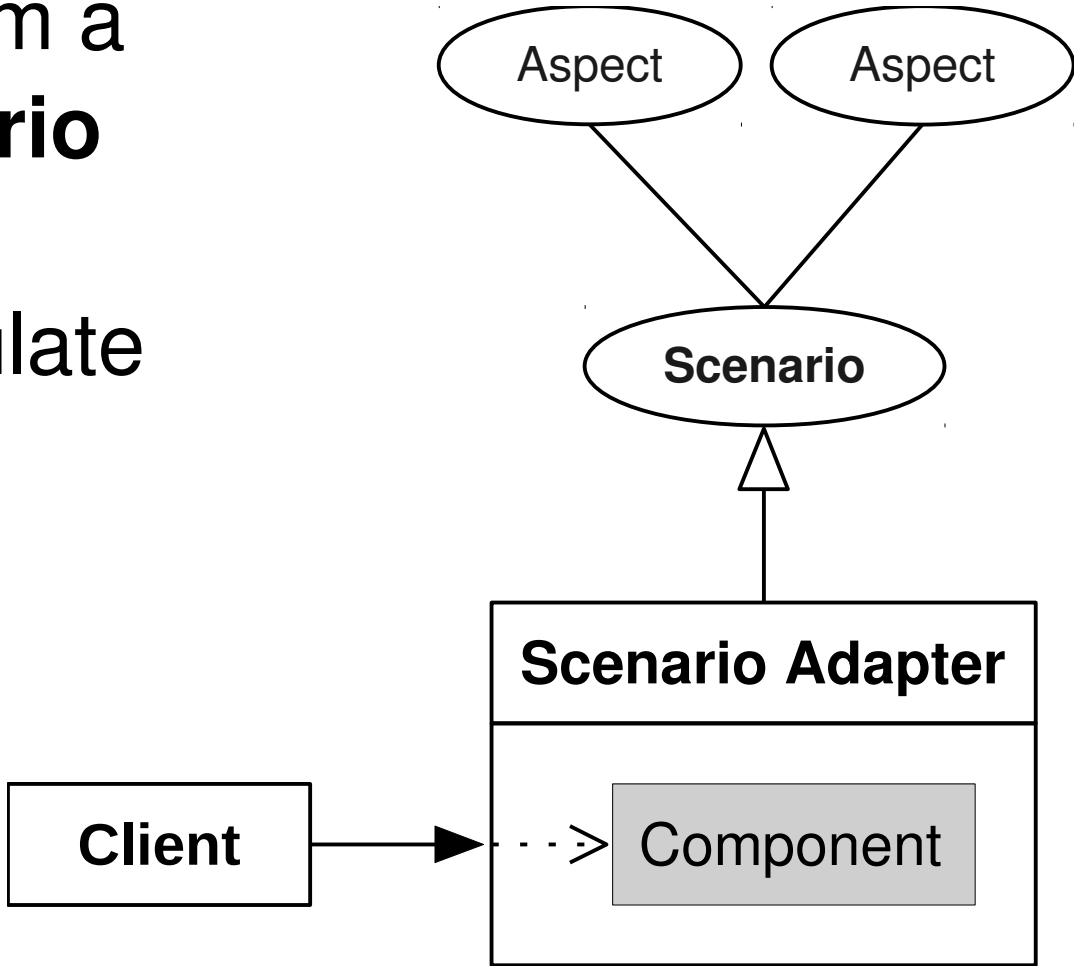


Resource allocation

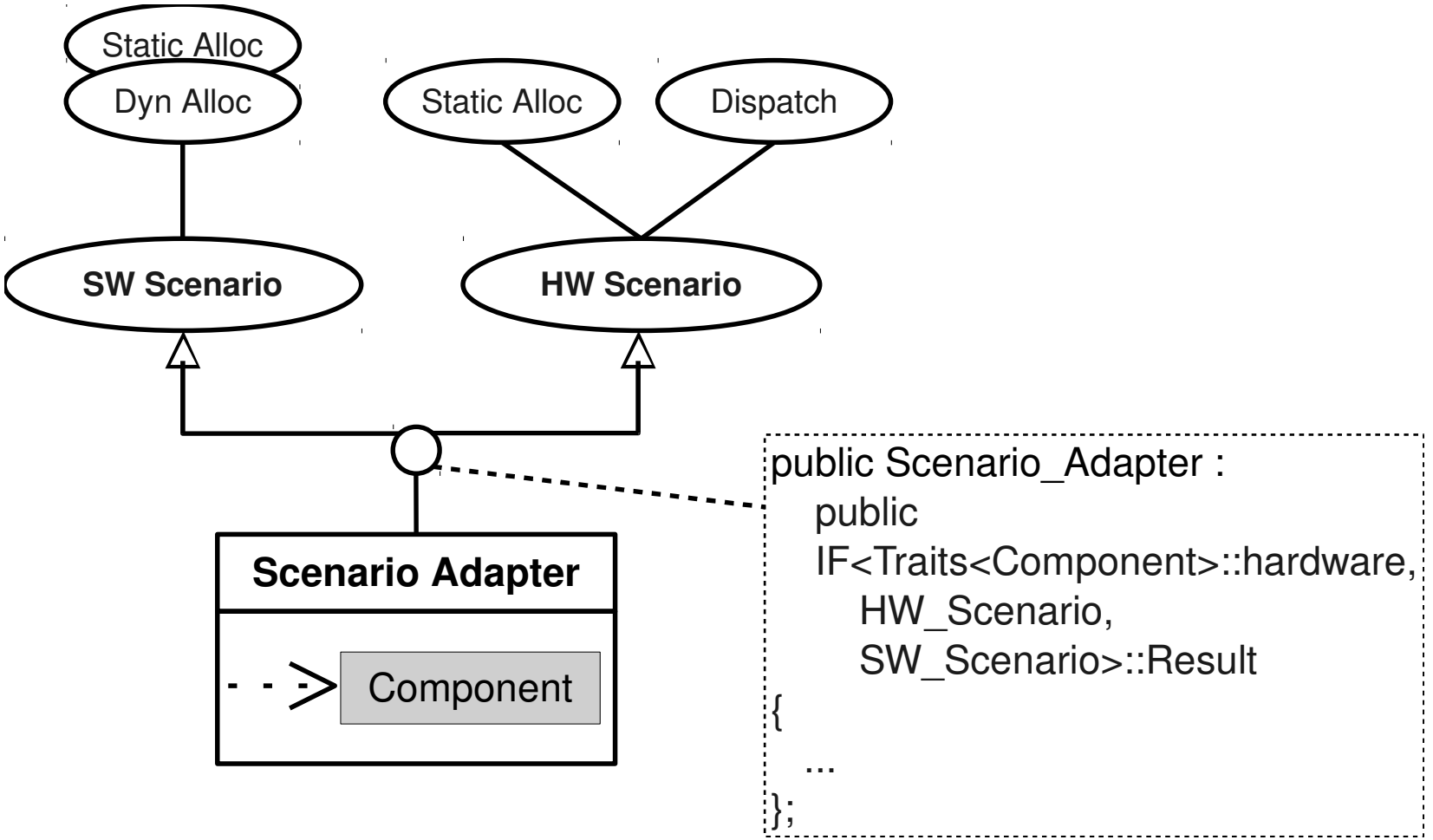
- Hardware is "frozen"
- Dynamic resource allocation is not available
  - e.g. **new** operator, malloc() …
- All data structures must reside in statically allocated memory

Scenario adapters

- Components are designed independently from a **execution scenario**
- **Aspects** encapsulate common features



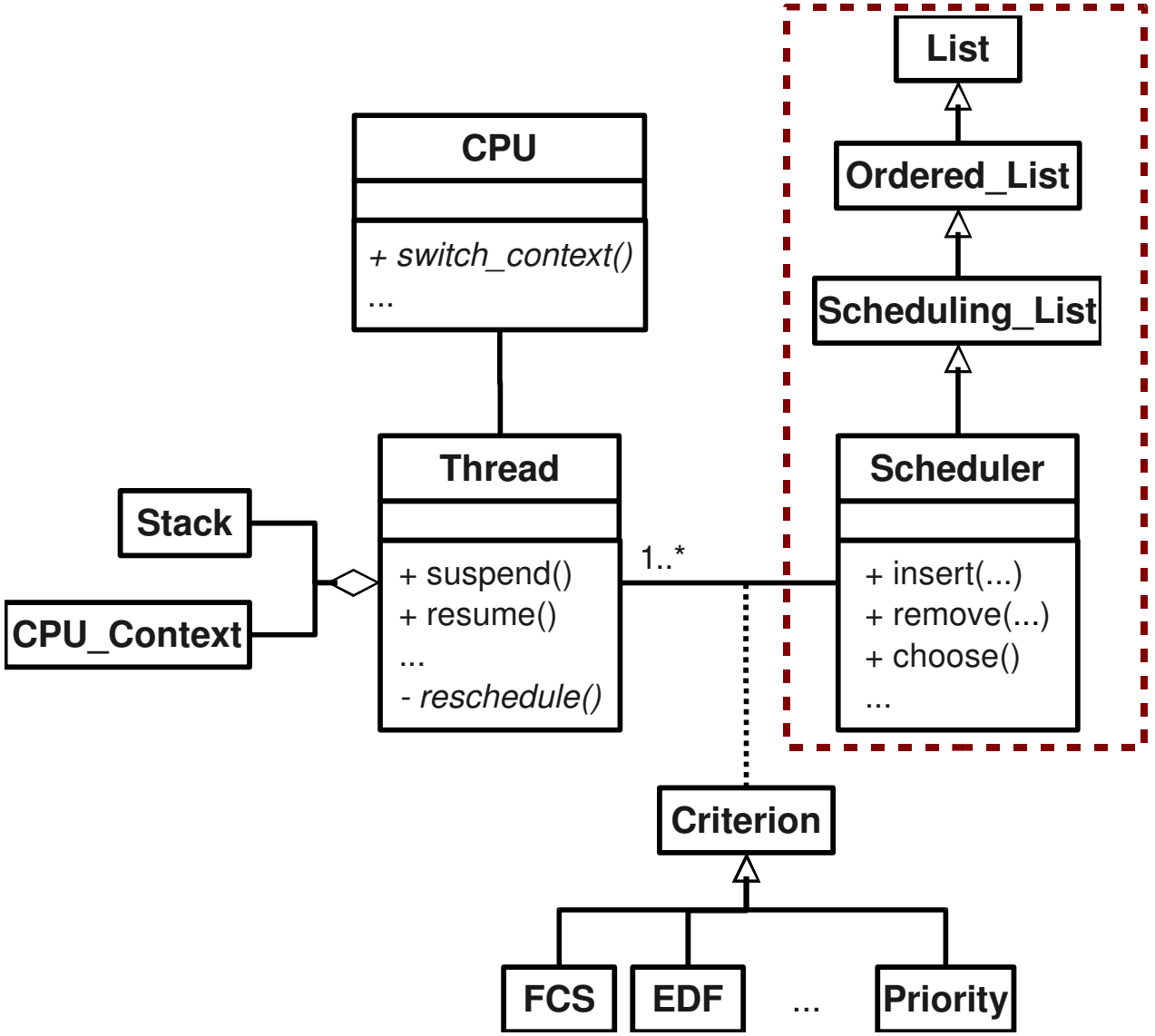
Hardware and software scenarios



Component adaptation

- Scenario adapters do not modify the inner structure of components
- Behavior is only added **before** and **after** the execution of each operation
- Components design must follow a careful domain decomposition process

Case study: EPOS Scheduler

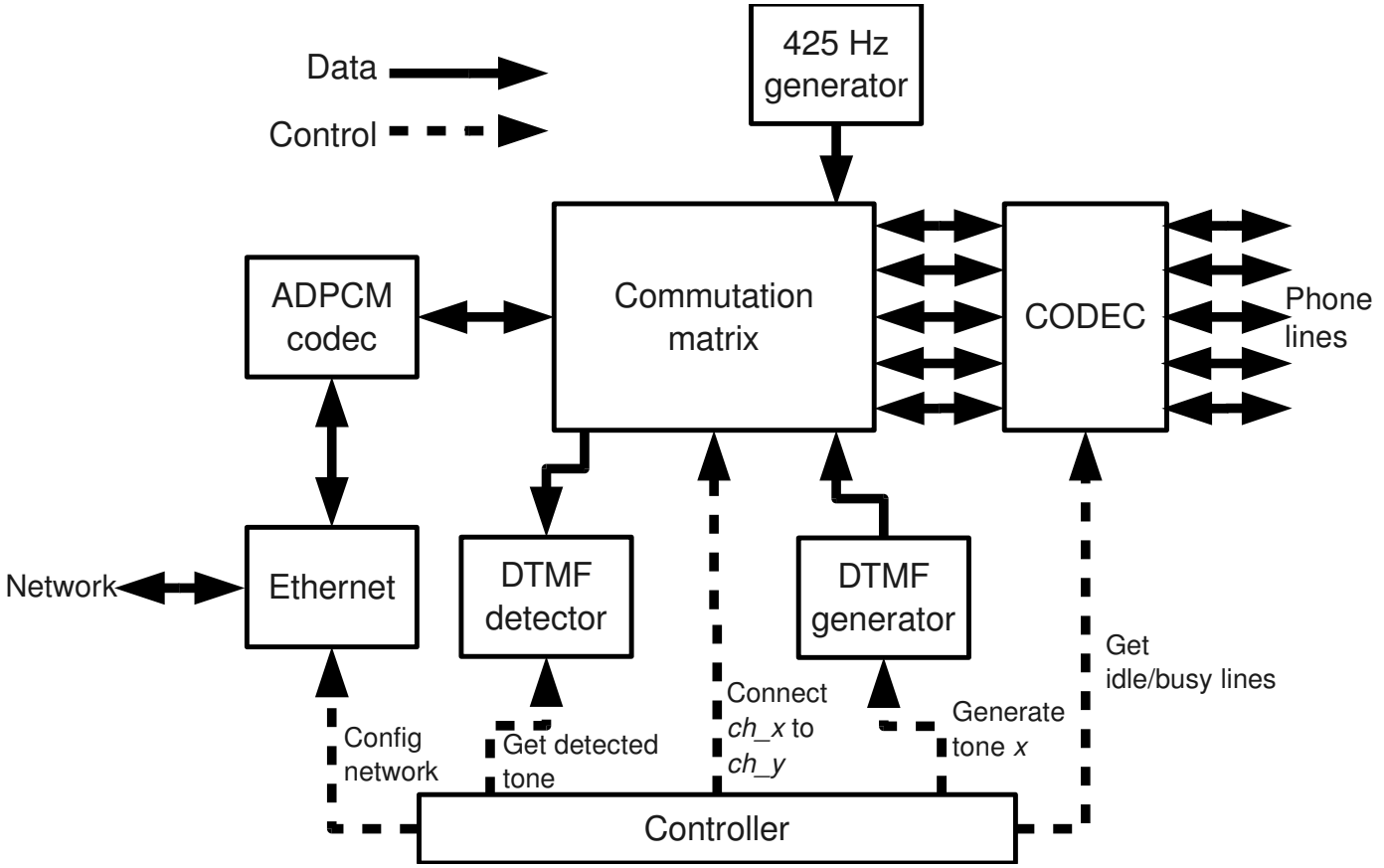


External allocation of links

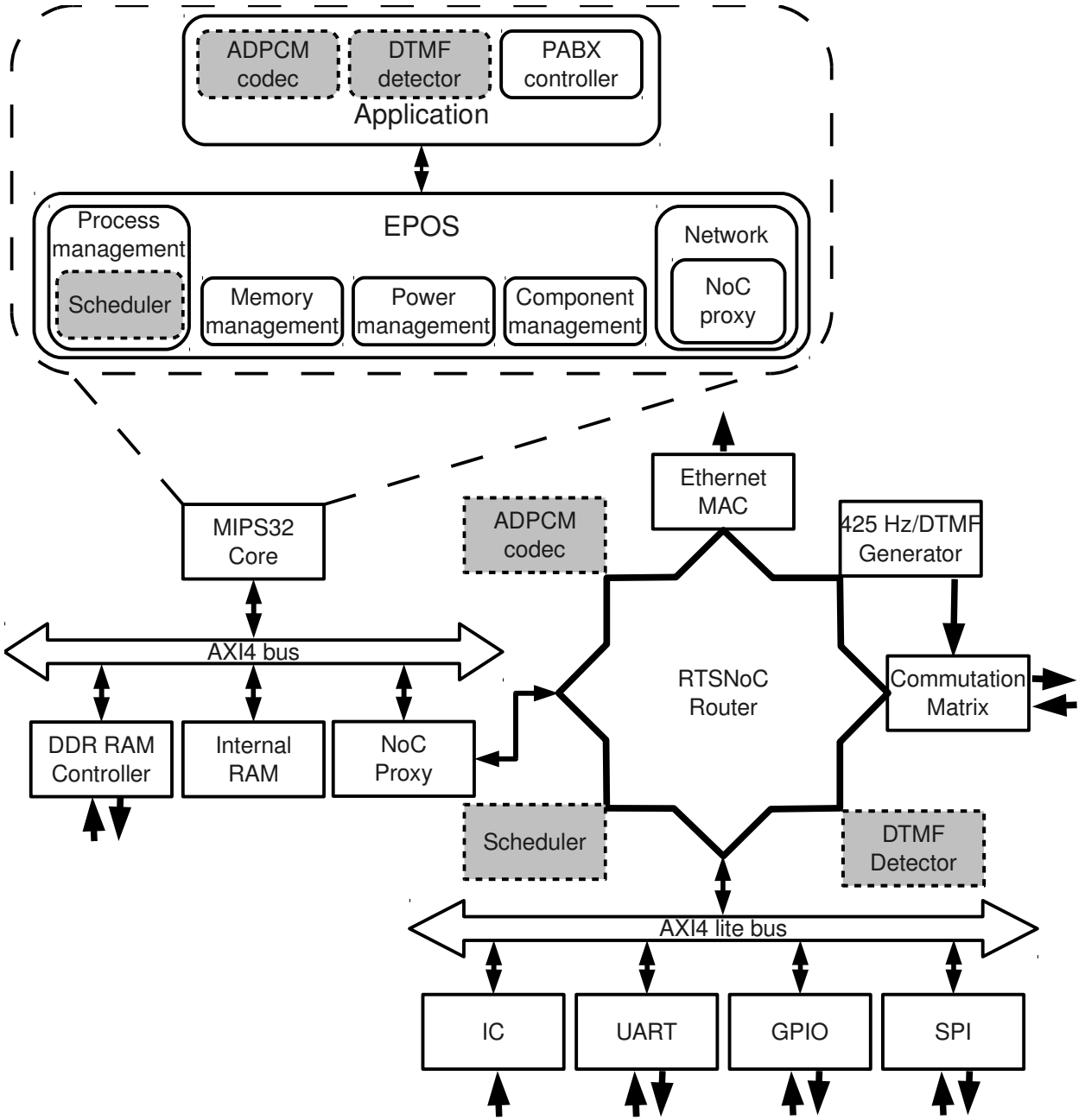
```
Link insert(Object obj) {
    Link link = Scenario::allocate(obj);
    Scheduler::insert(link);
    return link;
}

Object remove(Link link) {
    Object obj = Scenario::get(link);
    Scheduler::remove(link);
    Scenario::free(link);
    return obj;
}
```

Case study

- PABX application



PABX SoC


ADPCM codec
DTMF detector
PABX controller
Application
Process management
Scheduler
EPOS
Memory management
Power management
Component management
Network
NoC proxy
MIPS32 Core
ADPCM codec
Ethernet MAC
425 Hz/DTMF Generator
AXI4 bus
RTSNoC Router
Commutation Matrix
DDR RAM Controller
Internal RAM
NoC Proxy
Scheduler
DTMF Detector
AXI4 lite bus
IC
UART
GPIO
SPI

Results

- SW-only *vs* SW-adapted

- HW-only *vs* HW-adapted

Results

- SW-only *vs* SW-adapted
  - Memory footprint overhead = 5 %
  - Execution time overhead = 3 %

- HW-only *vs* HW-adapted

Results

- SW-only *vs* SW-adapted
  - Memory footprint overhead = 5 %
  - Execution time overhead = 3 %

- HW-only *vs* HW-adapted
  - Area overhead = 20 %
  - Speed overhead = 23 %

Conclusions

- It is possible to provide unified implementations of components through the isolation of specific HW/SW aspects and careful design
- Despite the current results, C++ implementations can be used at system-level in a straightforward way