Using a Dataflow abstracted Virtual Prototype for HdS-Design

ASPDAC 2009 Special Session

Hardware-dependent Software
for Multi- and Many-Core Embedded Systems

Wolfgang Ecker Stefan Heinen Michael Velten

Infineon Technologies AG

Germany

Never stop thinking

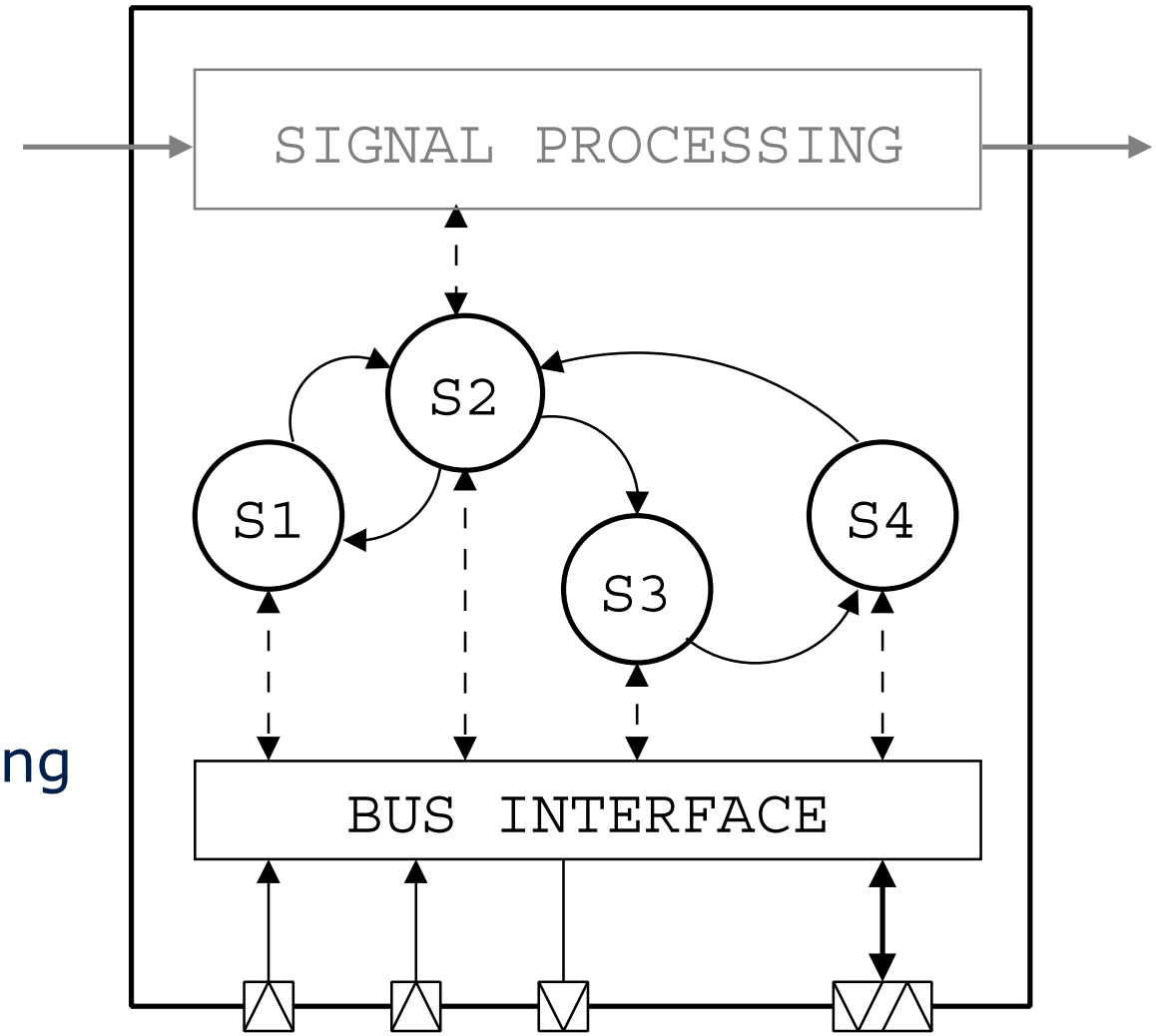
Motivation

- Improvement of execution speed of current VP solution of more than factor 10, i.e., reduce simulation time from 1 h ➔ 5 min
- Provide higher abstracted TLM+ modeling style (to achieve performance) which is compatible and interchangeable with today's VP modeling
- Ease the integration and reuse of algorithm models

Table of contents

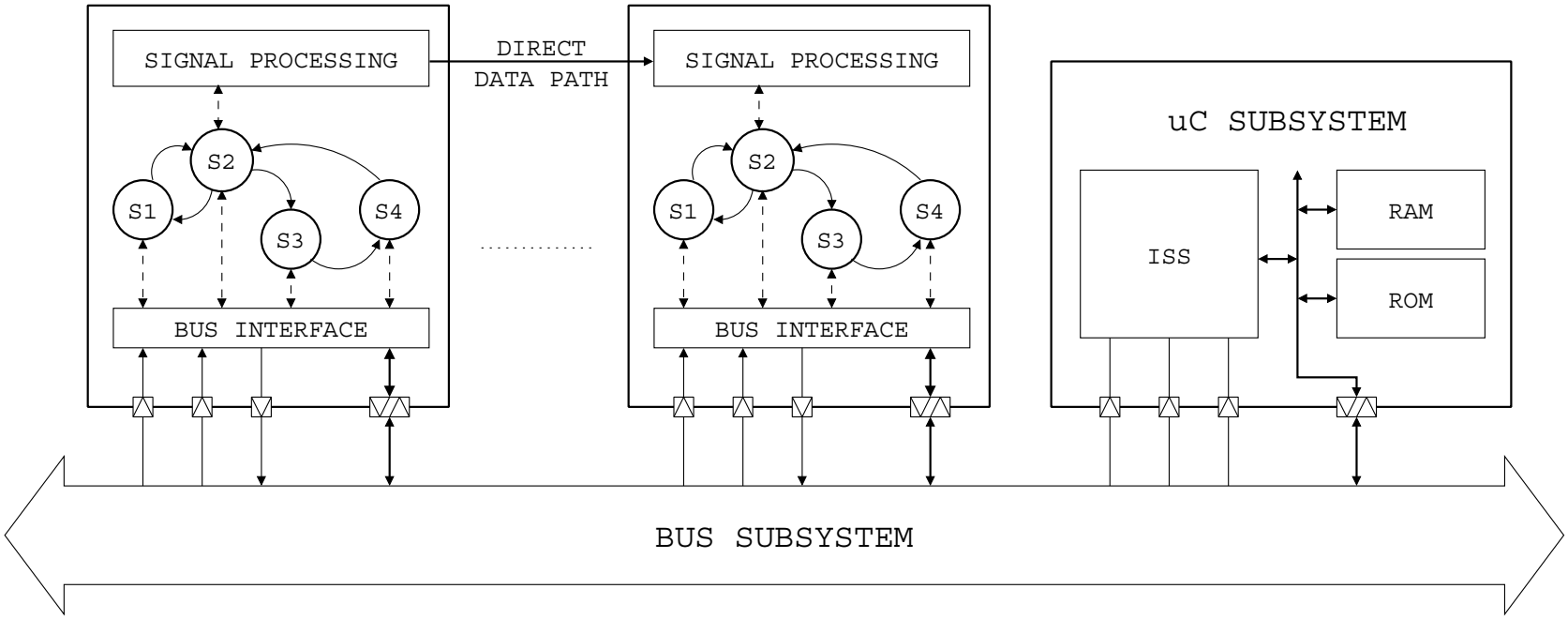
Introductive 3G Modem VP
Basic Structure of the VP Models

- Bus interface handling, register accesses, reset, clock, interrupts
- State machine (S1-S4) controlled via the bus interface
- Optionally, a signal processing unit containing the reused algorithm models



Introductive 3G Modem VP
Structure of the 3G Modem VP

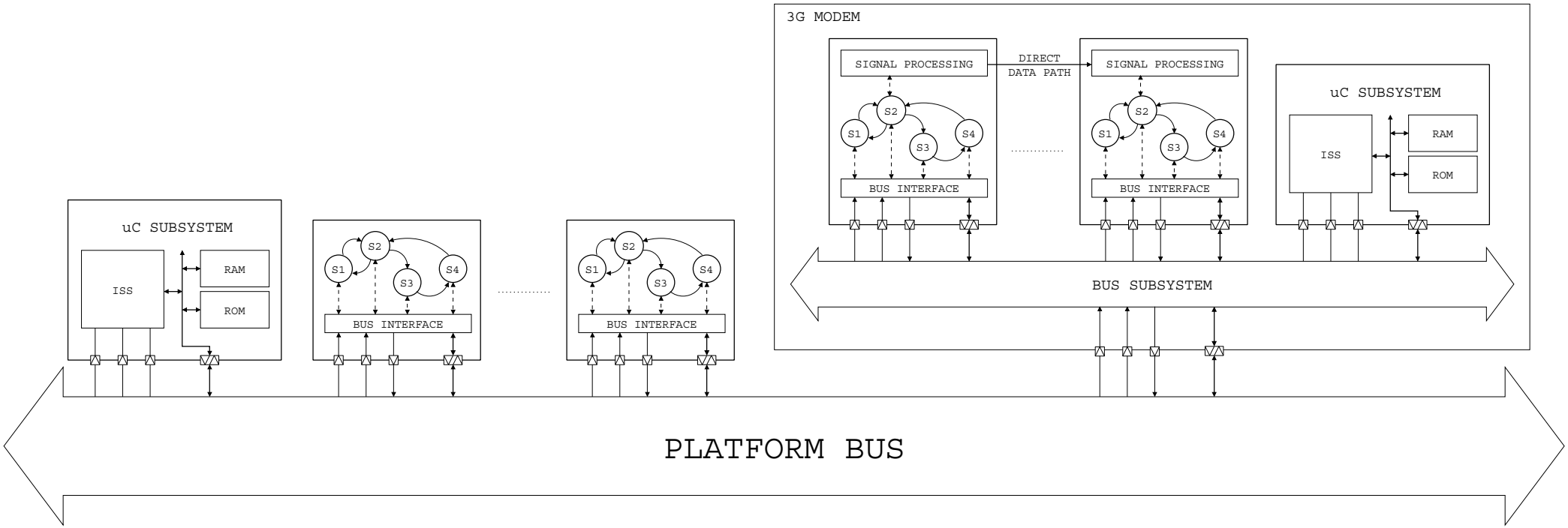
- Backbone components: ISS, memories, bridges, and busses
- System control: SCU, ICU, memory control, and timer
- High cycle accuracy, bit exact data path simulation
- 3G physical layer testbench and regression suite
- VP simulation speed: 1/300 (max simulation time 5 minutes)



Introductive 3G Modem VP
VP Platform

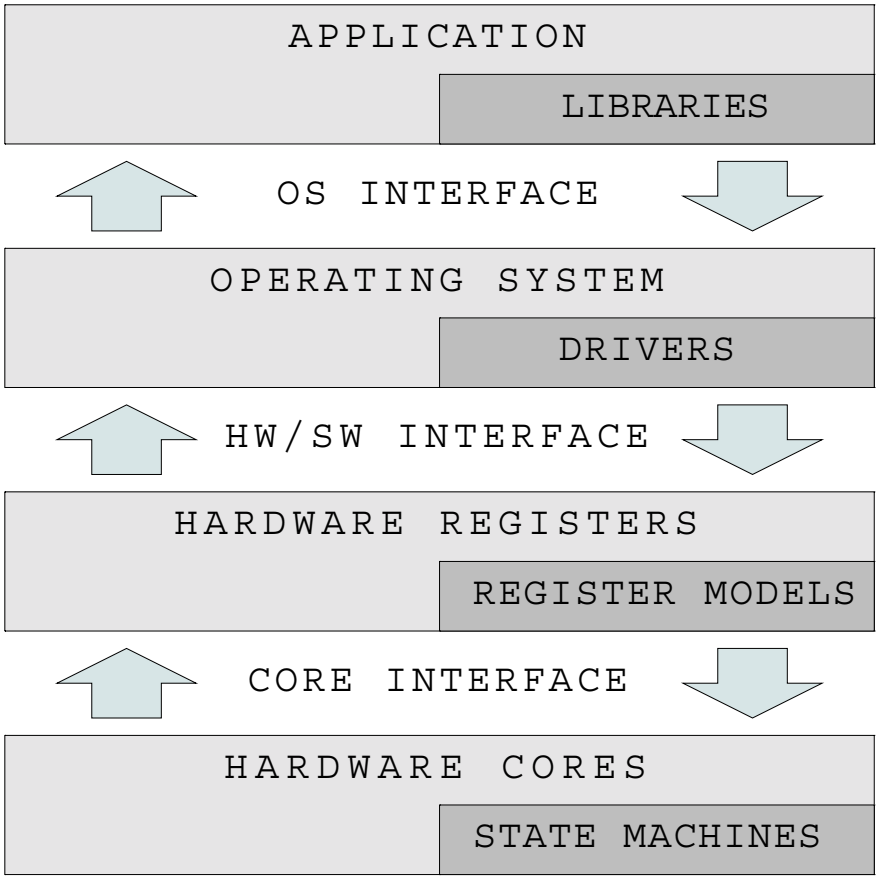
- 3G modem integrated with application HW into one SoC
- Execution of 3G protocol stack and further application SW
- Drawbacks: High cycle accuracy and bit exact data path
- Resulting simulation time up to 1 hour is not acceptable

➔ A new methodology beyond TLM is required

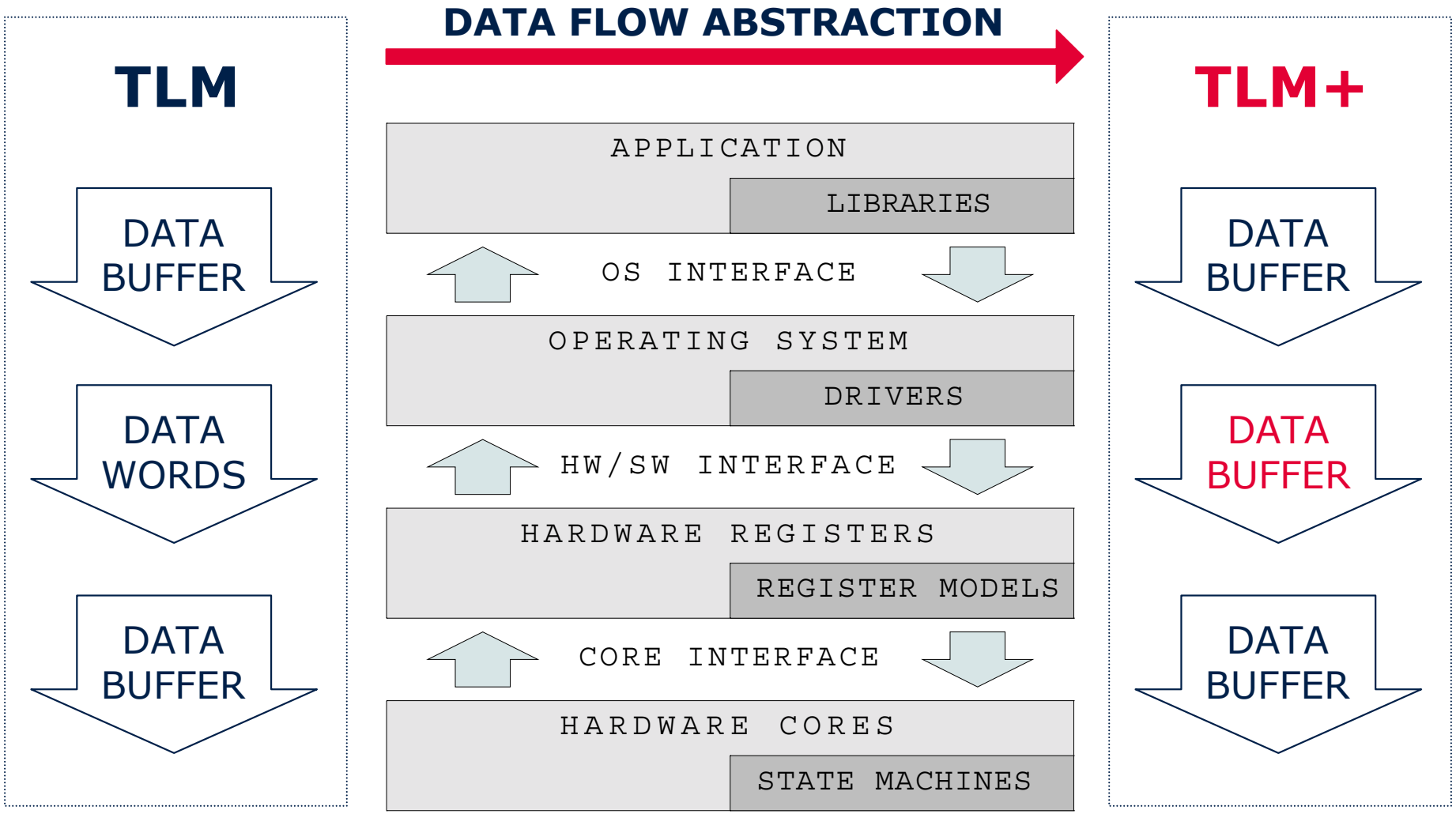


Embedded HW/SW System Abstract View (Bottom-Up)

- Hardware core incorporates functionality of a HW device
- Core interface offers internal access to the HW registers
- Hardware registers enable software access to the hardware
- HW/SW interface provides SW access to the HW registers
- OS offers an IO subsystem with a device driver interface
- OS interface offers with its IO subsystem access to HW devices
- Application SW runs on top of the operation system



Embedded HW/SW System
Dataflow Abstraction

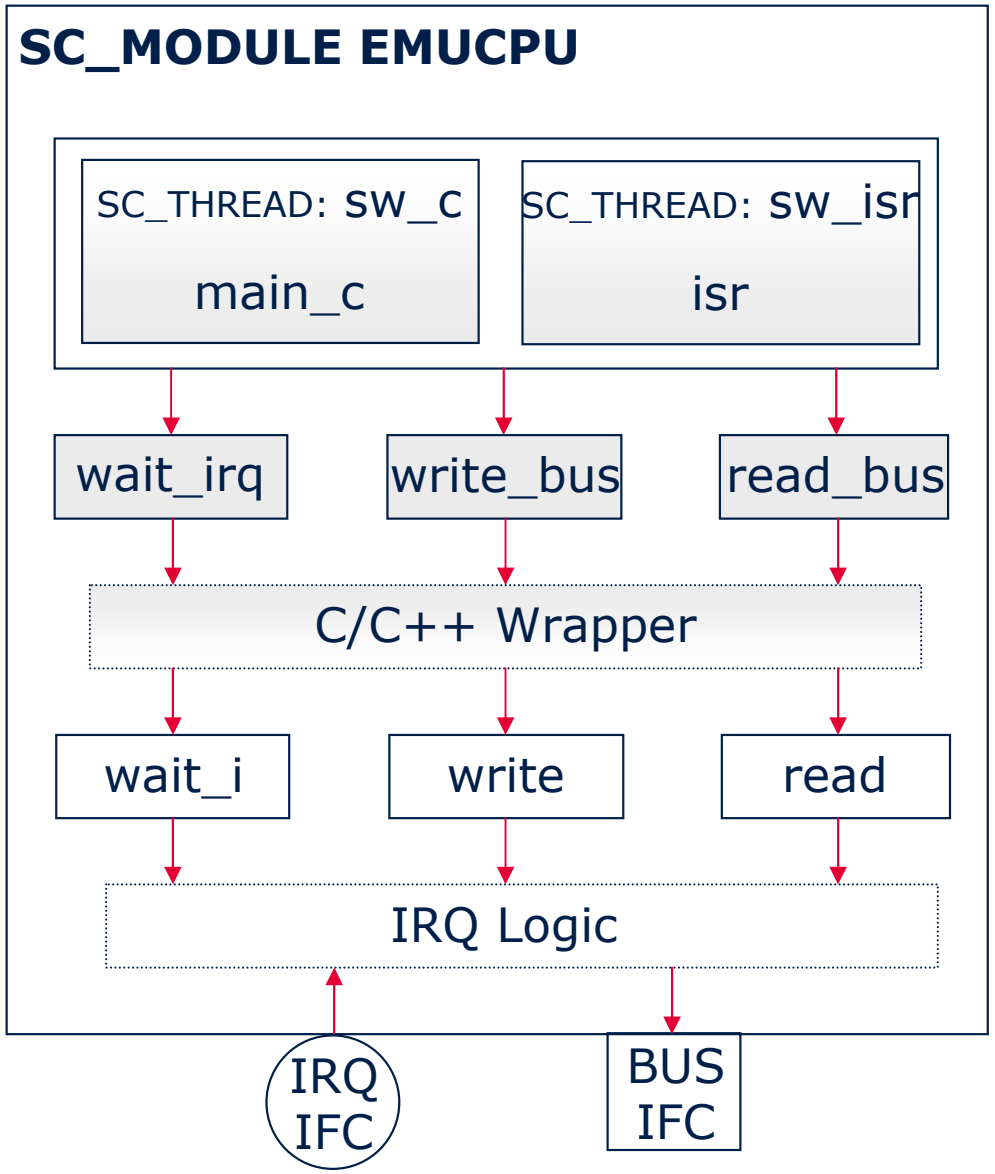


Dataflow Abstraction Methodology Requirements

- ISS including its compiler does not provide HW/SW interface abstraction ➔ native software execution required
- Development of a CPU model which offers the same interface for register access and IRQ handling as the ISS
- Extension of the interfaces to enable buffer transfer but also provide word access for TLM backward compatibility
- The HW/SW interface need to provide an abstract register interface for package transfer

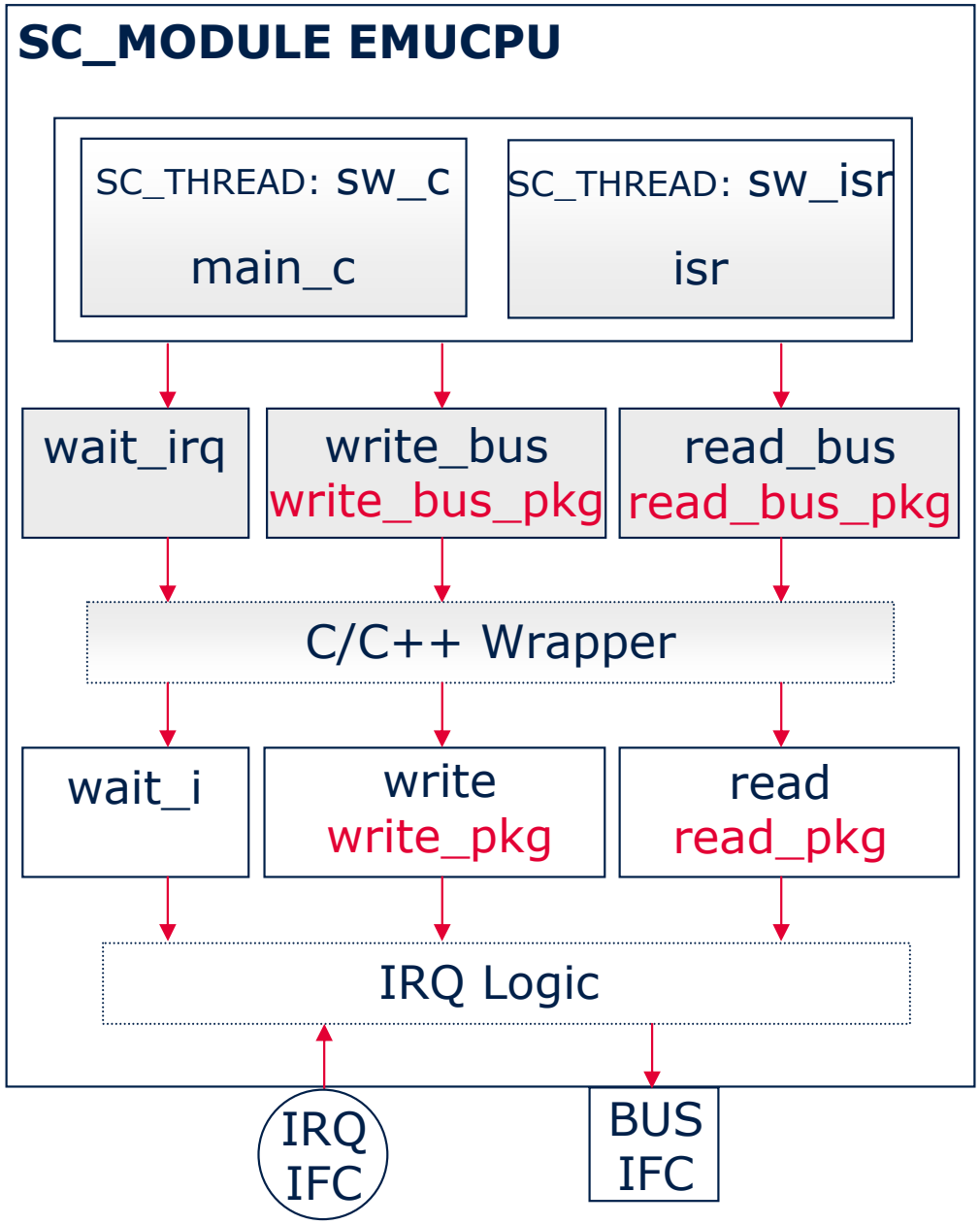
Dataflow Abstraction Methodology
Native Software Execution – EMU CPU

- The main program of the C software and the ISR is started in the context of an SC_THREAD
- Contains a C/C++ wrapper to provide bus access functions for the C software (read_bus, write_bus)
- The bus access blocks the main_c software execution in case of active interrupts



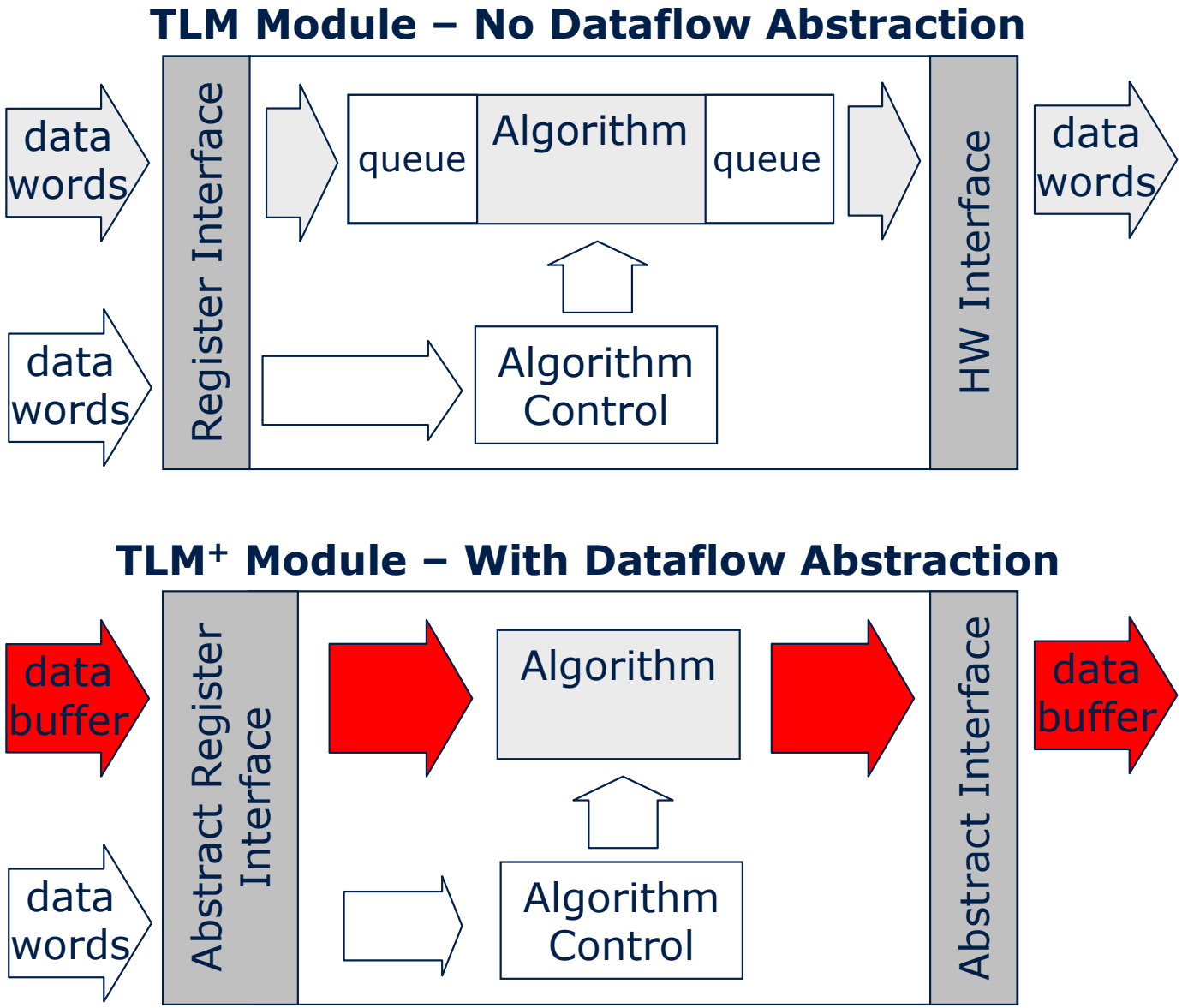
Dataflow Abstraction Methodology
HW/SW Interface Abstraction

- EMU CPU model is extended by the write_bus_pkg and read_bus_pkg functions for buffer access
- Request and response classes extended by a package pointer and a package size member
- The HW/SW interface is extended by get_REG_pkg and set_REG_pkg functions
- The OS device drivers directly pass the buffers of the application SW to the HW by calling the functions for package transfer



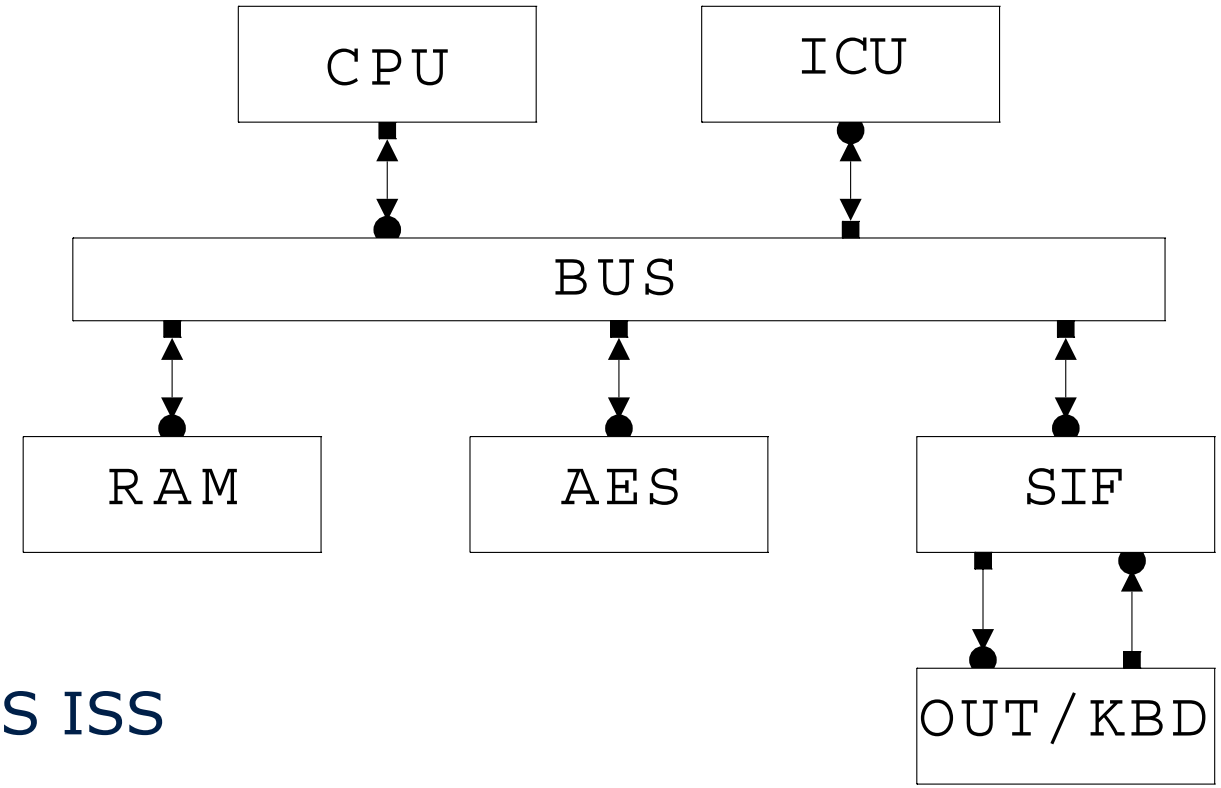
Dataflow Abstraction Methodology
Core Interface Abstraction

- Currently queues are used for algorithm synchronization
- Now algorithms can process the data buffers directly
- Payload of HW interfaces is extended by the package pointer and a the package size



Dataflow Abstraction Methodology Demonstration Model

- Development of three different virtual prototypes
- Each VP contains a CPU, BUS, ICU, AES encoder/decoder, and a serial console connected to a SIF



- ISS_VP
  - CPU is a SystemC MIPS ISS
- EMU_VP
  - CPU is an EMU CPU without package interface
- HWSW_VP
  - CPU is an EMU CPU with package interface

Experimental Results
Test Applications

- Hardware peripherals:
  - AES encoder/decoder supports packages with the maximum of 16 bytes
  - SIF has no package size restriction, packages are passed completely to the serial console

- Software applications:
  - IO Test: large buffers, no SW- and HW algorithms
  - AES Test: small buffers, HW algorithms, no SW algorithms
  - MIXED Test: SW- and HW algorithms, small and large buffers

Experimental Results
Simulation Performance Results

	IO [s]	AES [s]	MIXED [s]
ISS_VP	1048,6	1632,7	1206,5
EMU_VP	49,9 (21)	84,9 (19)	48,0 (25)
HWSW_VP	0,2 (249)	32,5 (2,6)	3,6 (13)

- ISS vs. native software execution (EMU_VP) shows a performance gain factor of about 20x
- IO test shows fast performance of the HWSW_VP using large buffer sizes (compiler can optimize)
- AES test shows a small performance gain, because of the small buffers and the dominant AES algorithm
- The MIXED test can be considered as a common use case

➔ The HWSW_VP is more than 10x faster than the EMU_VP

Summary

- The 3G modem virtual prototype was introduced as an industrial use-case example to show the need for a new methodology
- An abstract view of an embedded HW/SW system was used to introduce the data flow abstraction methodology
- A SystemC EMU CPU model was introduced which enables further HW/SW interface abstractions
- The extension of the HW/SW, register and core interfaces was explained to provide package transfer for the application buffers
- Finally, experimental results were presented which show that the data flow abstracted VP is more than 10x faster than the EMU_VP and more than 200x faster than the ISS_VP

Outlook & Current Work

- The abstraction of the control flow to ease and speed-up the configuration of HW algorithms is in progress
- A prototype of a central and configurable performance model to provide timing information has been developed
- Future work will focus on the development of a methodology for automated timing annotations of the native software execution

Never stop thinking