NANYANG TECHNOLOGICAL UNIVERSITY

TUM CREATE

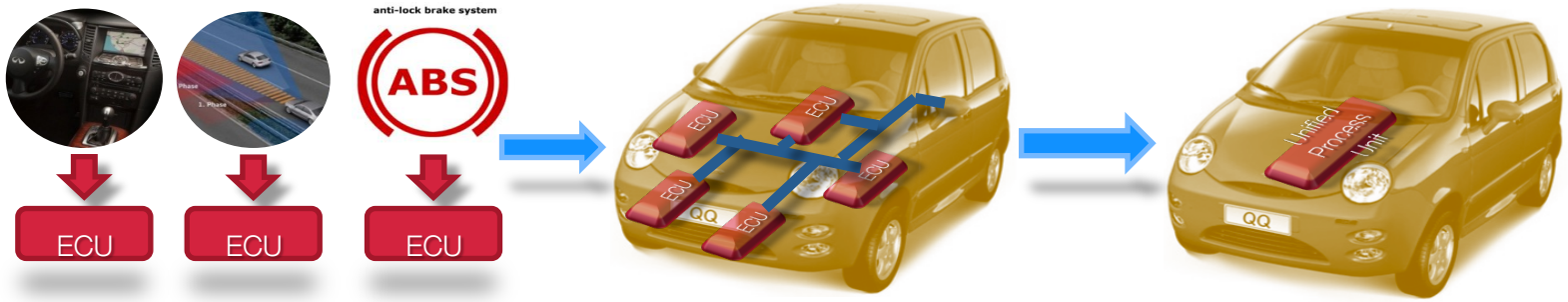
Microkernel Hypervisor for a Hybrid ARM-FPGA Platform

Khoa D. Pham, Abhishek K. Jain, Jin Cui, **Suhaib A. Fahmy**, Douglas L. Maskell
School of Computer Engineering
Nanyang Technological University, Singapore
(in collaboration with TUM CREATE, Singapore)

Int. Conf. Application-specific Systems, Architectures and Processors (ASAP)
5-7 June 2013, Washington, USA

Motivation

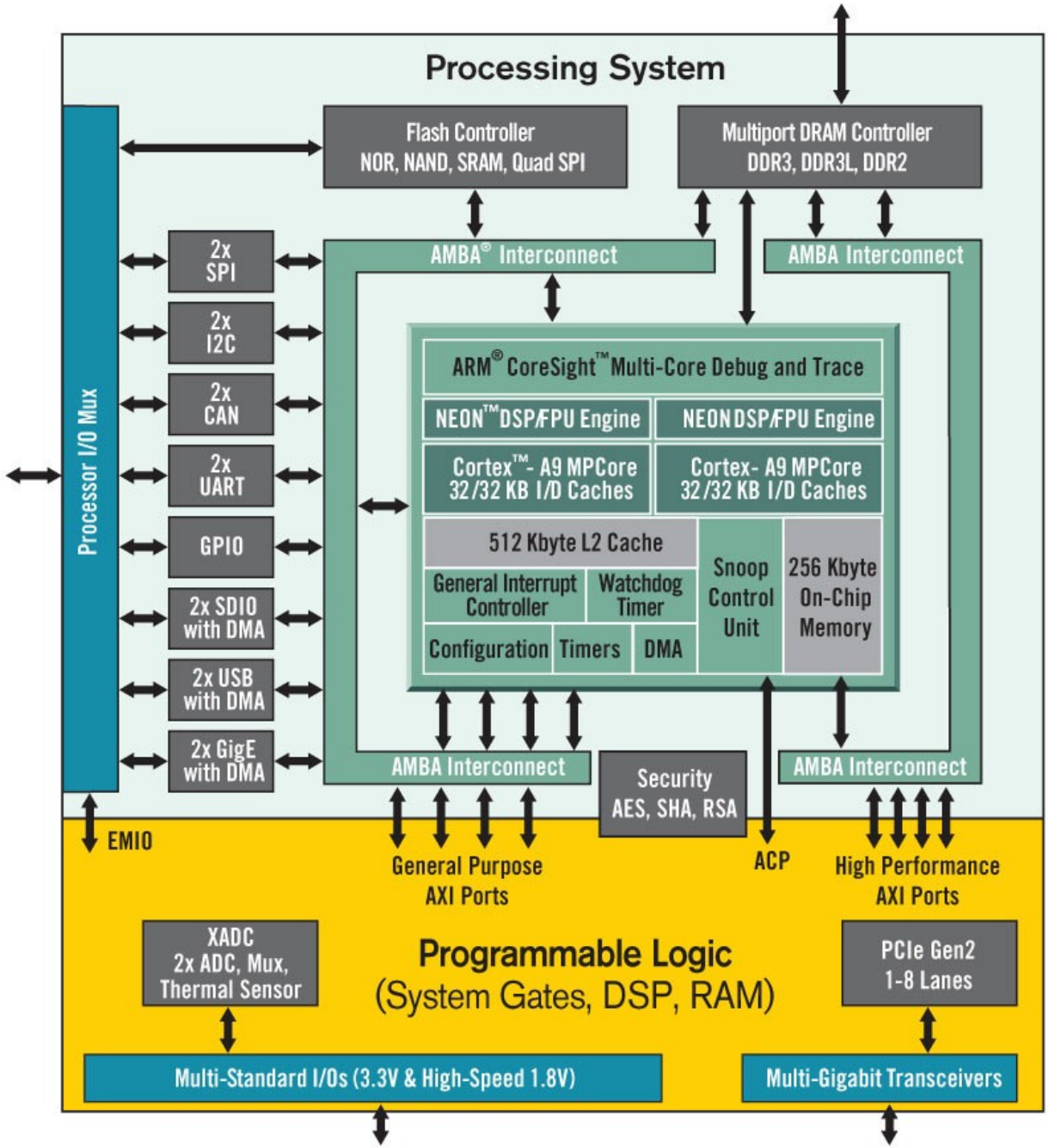
- Increased computing in vehicles through increased number of compute nodes
- Isolation essential for safety → complex network
- Desire to consolidate compute on fewer nodes
- New hybrid architectures provide ideal platform



NANYANG TECHNOLOGICAL UNIVERSITY

Hybrid Platform

- New hybrid FPGAs with ARM cores provide:
  - Processor-first view of device, independently functional
  - A core of comparable performance to existing SoCs
  - High throughput between core and fabric
- Offers us the software-programming view but with hardware performance
- How can we take advantage of hardware isolation while still offering a software interface?
- This is still a key difficulty in design for these hybrid architectures (design time)



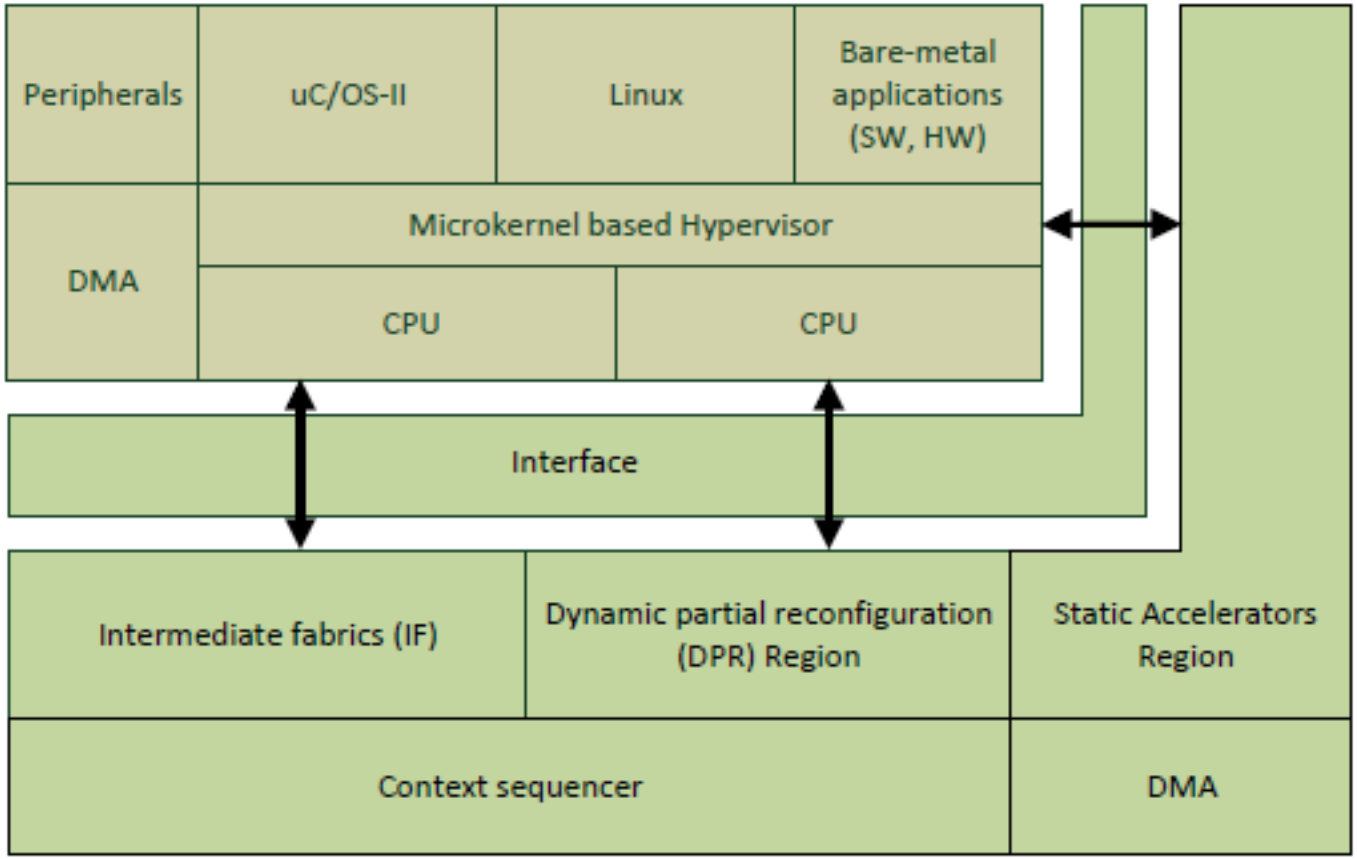
Courtesy Xilinx

NANYANG TECHNOLOGICAL UNIVERSITY

Proposed Approach

- A hypervisor to virtualise access to all resources
  - Software, including bare metal applications, full OS, realtime OS
  - Hardware:
    - Static accelerators
    - Virtual fabric for ease of programming
    - Partially reconfigurable regions
- Task management across resources, with low latency communication and context switch

Proposed Approach

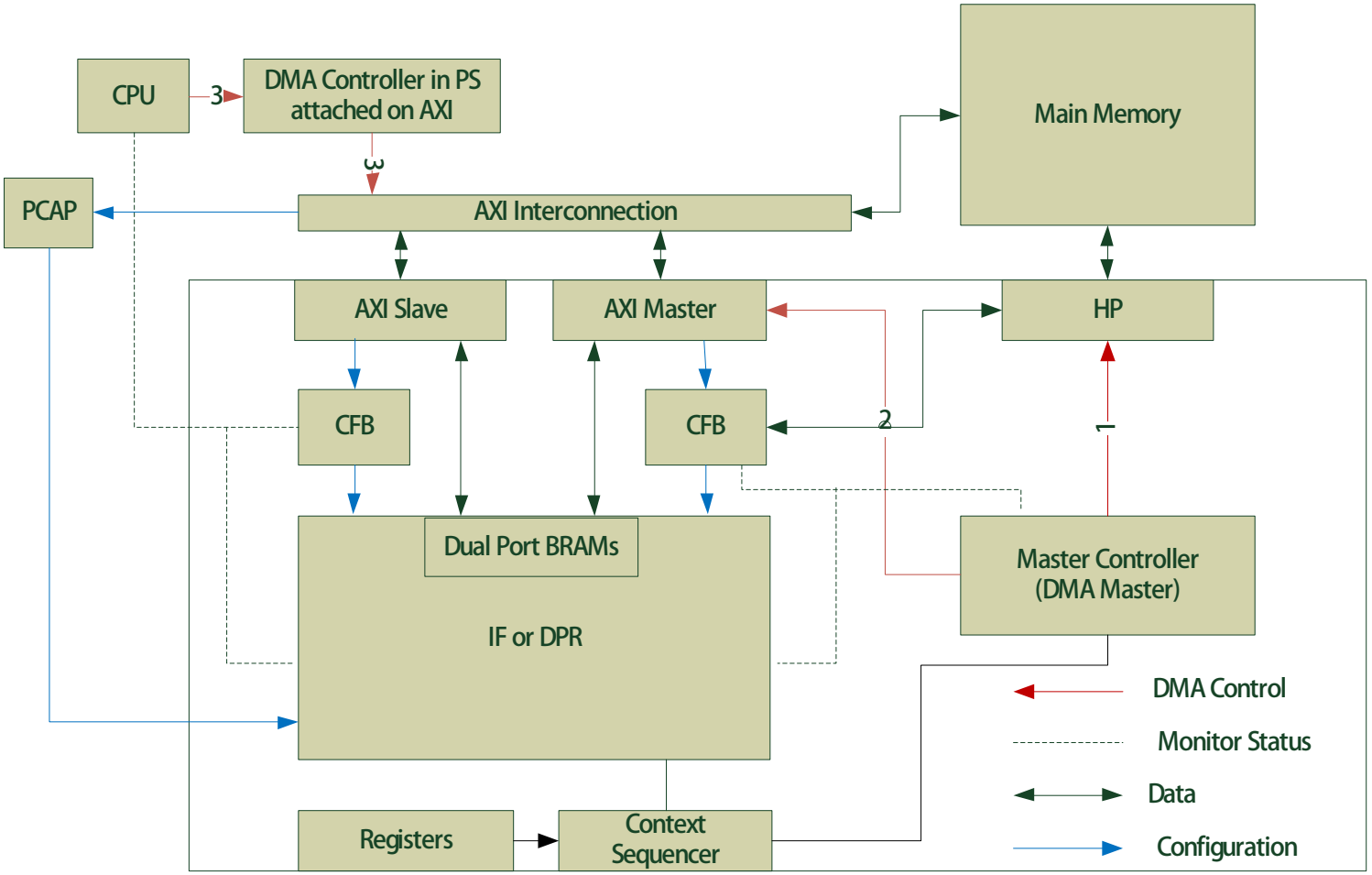


NANYANG TECHNOLOGICAL UNIVERSITY

Hardware Support

- Communication:
  - Zynq provides high performance AXI interface between processor and fabric
- Context Frame Buffer
  - Hardware tasks can be decomposed into multiple contexts
  - Storing contexts off-chip is more scalable
  - A buffer in Block RAMs makes access faster
- Intermediate Fabric
  - A way of using the logic fabric at a higher layer of abstraction
  - Communicate through dual ported Block RAMs

Hardware Support



NANYANG TECHNOLOGICAL UNIVERSITY

Porting the Hypervisor

The CODEZERO hypervisor from B-Labs was modified:

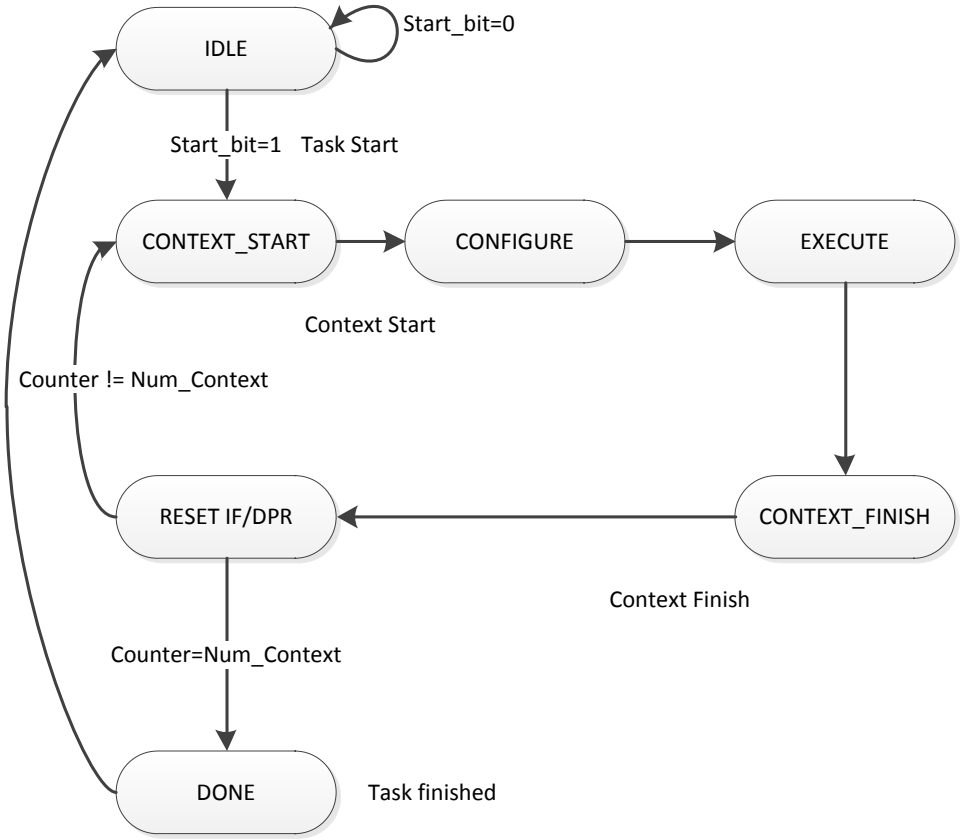
- Rewriting drivers for the Zynq ARM (PCAP, timers, interrupt controller, etc.)
- FPGA initialisation (clocks, pin mapping, interrupts)
- Hardware task management and scheduling
- DMA transfer support

- All scheduling and management is managed by the hypervisor

Context Sequencer

- Manages hardware tasks
- Loads context frames (parts of a task)
- Memory mapped register interface in fabric
- Control register to control how many frames and base address for configuration
- Status register indicates hardware task status like completion

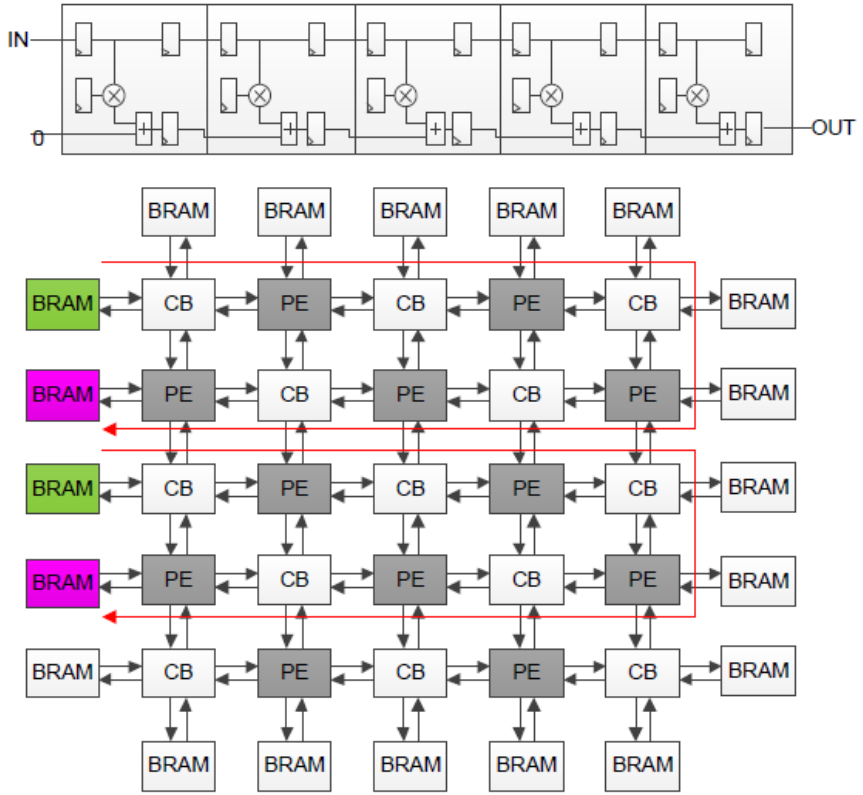
Context Sequencer



NANYANG TECHNOLOGICAL UNIVERSITY

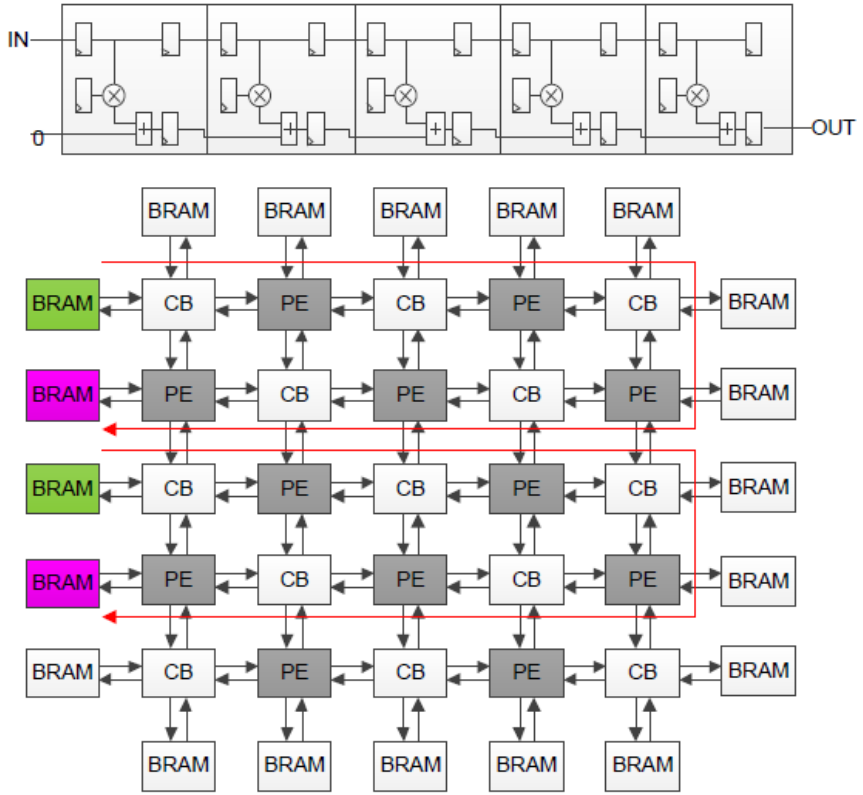
Intermediate Fabric

- Allows more coarse grained use of FPGA logic fabric
  - Simple compilation
  - Reduced configuration time
  - Predictable timing



Intermediate Fabric

- A simple fabric with DSP block-based processing elements
- Configurable nearest neighbour connections
- Map two applications:
  - Matrix multiplication
  - FIR filter
- Fabric not optimised, but proof of concept



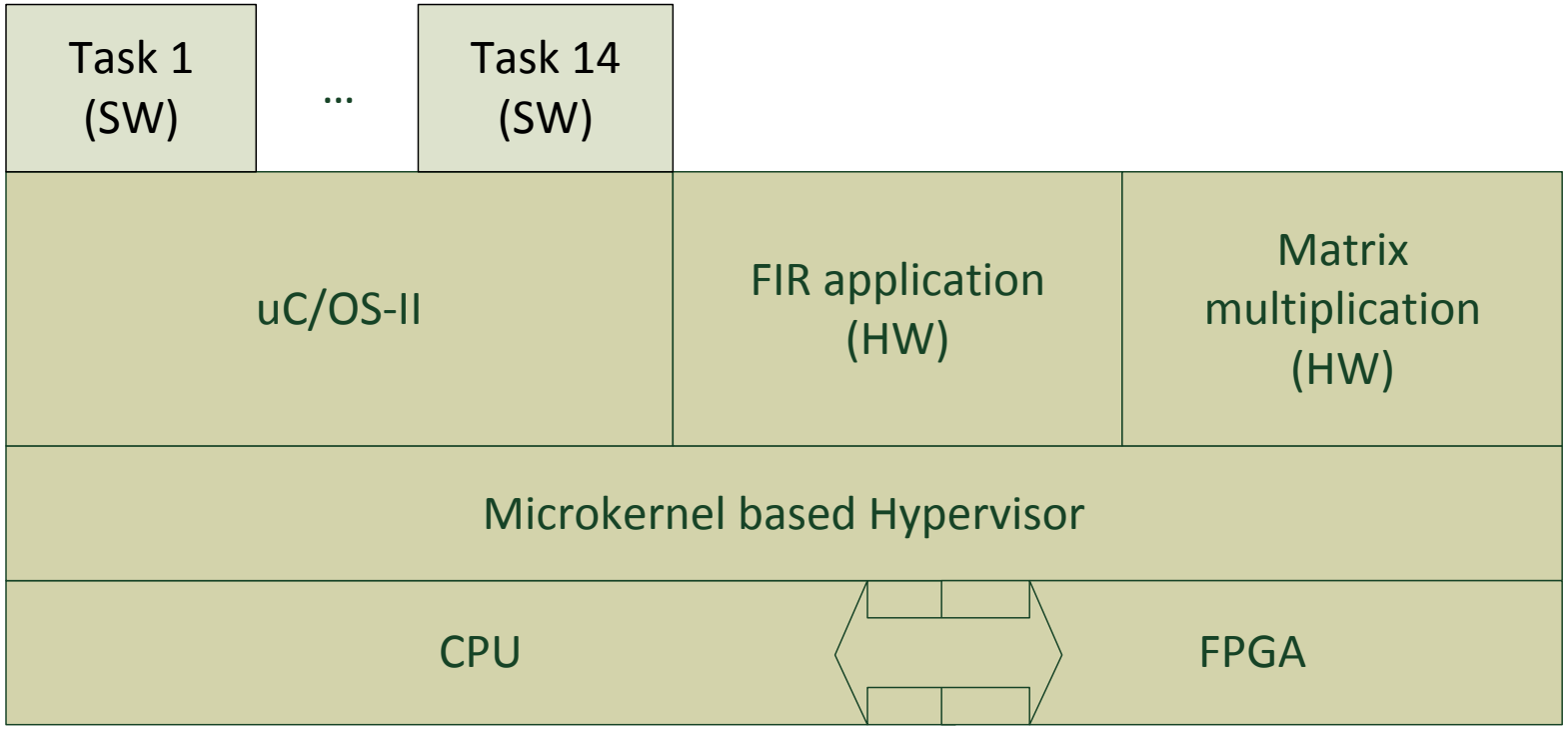
Hardware task management

- Non-preemptive switching
  - Hypervisor mutex mechanism used to block access to hardware
  - On completion of a context, lock is released to allow switch
  - No need for context save and restore
  - Minimal modifications to hypervisor required
- Preemptive switching
  - Must be able to store and load contexts
  - Modifications to user thread control block and context switch
  - Can provide faster response time at cost of overhead

Case Study

- Proof of concept with three containers:
  - Real-time OS container with 14 software tasks
  - A bare metal application that runs a hardware FIR filter task
  - A bare metal application that runs a hardware matrix multiplication task
  - The hardware tasks use the same intermediate fabric
- FIR filter uses single context frame
- Matrix mult requires 3 context frames

Case Study



Case Study

- Context switch time:

Clock cycles (time)	Non-preemptive	Preemptive
T_{lock} (no contention)	214 (0.32μs)	NA
T_{lock} (with contention)	7738 (11.6μs)	
T_{C0_switch}	3264 (4.9μs)	3140 (4.7μs)

- Configuration and response times:

Clock cycles (time)	Non-preemptive		Preemptive	
	FIR	MM	FIR	MM
T_{conf}	2150 (3.2μs)	3144 (4.7μs)	3392(5.1μs)	5378 (8.1μs)
T_{hw_resp}	(8.5μs-19.7μs)	(9.9μs-20.3μs)	(9.8μs)	(12.8μs)

Future Work

- Porting Linux to be para-virtualised on top of CODEZERO
- A detailed comparison with hardware managed by Linux threads on the same hypervisor
- Direct support for partial reconfiguration
- Improved intermediate fabric
- Optimisation of communication between hypervisor, hardware, and software tasks