Preemptive Hardware Multitasking in ReconOS

Markus Happe, Andreas Traber, Ariane Trammell

Communication Systems Group, ETH Zürich, Switzerland

ARC'15 Symposium

Bochum, 15-17.04.2015

ETH Zürich

Adding preemptive hardware multitasking to a reconfigurable system-on-chip (rSoC).

Preemptive hardware multitasking

multiple tasks execute on a single slot

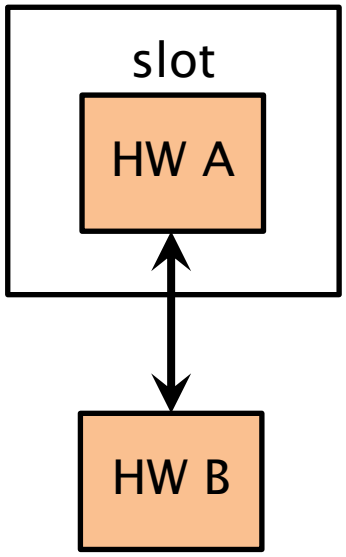
enable rSoCs to use smart schedulers

fairness, responsiveness, avoid starvation

Unmodified hardware source code

task preemption is transparent to developer

For context restoration, we can use task- or FPGA-specific methods.



How to switch between tasks A & B?

(1) stop task A, save context A

(2) reconfigure task B, restore context B, run B

What is the context?

content of all state-holding elements

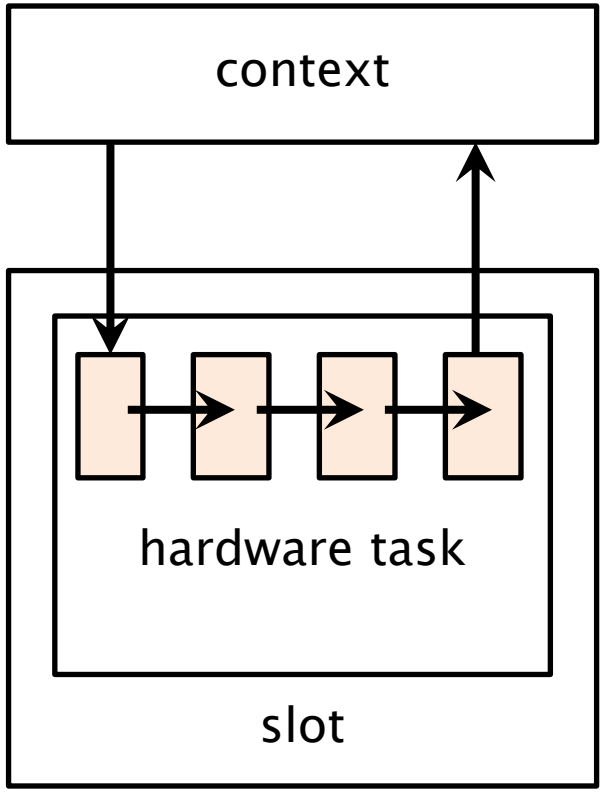
i.e. flip-flops, BRAMs, LUT-RAMs

Which approaches exist?

(i) task-specific preemption (scan-chains)

(ii) FPGA-specific bitstream readback

Task-specific preemption techniques extract/insert only the actual context.



Add scan-chains to hardware tasks

extract/insert context from/to scan-chain

Drawbacks

modifications at source code- or netlist-level

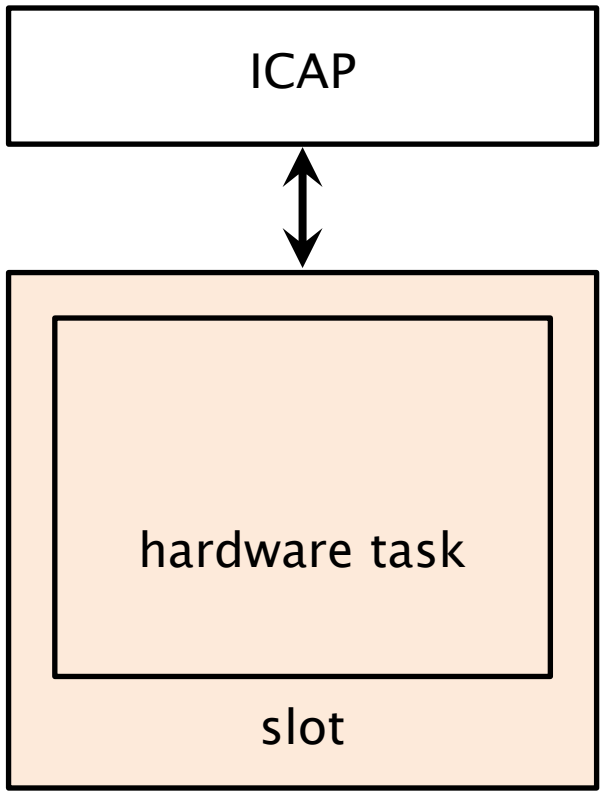
considerable hardware overhead

Advantages

approach indepependent of FPGA architecture

only captures/restores used resources

Bitstream readback methods read/ write configuration of the entire slot.



Read back bitstream of a partial region

using internal configuration access port (ICAP)

read contents of all flip-flops, BRAMs, etc.

Drawbacks

bitstream format tailored to FPGA family

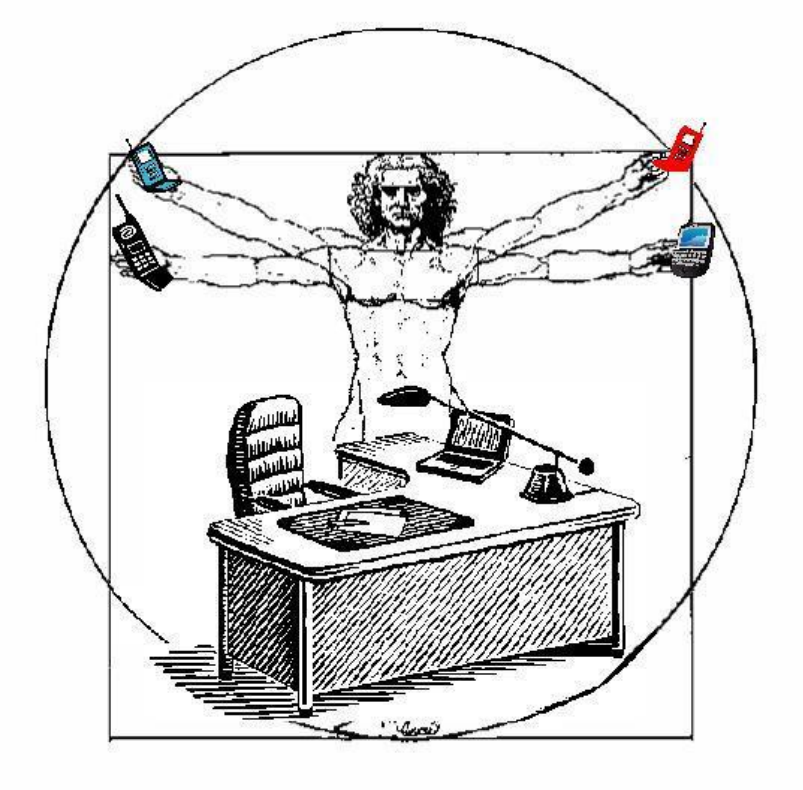
time-inefficient (capture/restore all resources)

Advantages

no modifications at task-level

no hardware overhead

Preemptive Hardware Multitasking in ReconOS



Methodology

bitstream readback

ReconOS Integration

ICAP controller, scheduler

Virtex-6 Evaluation

reconfiguration overhead

Our methodology is based on Xilinx PlanAhead PR toolflow.

full bitstream

system_A.bit

partial bitstreams

partial_A.bit

partial_B.bit

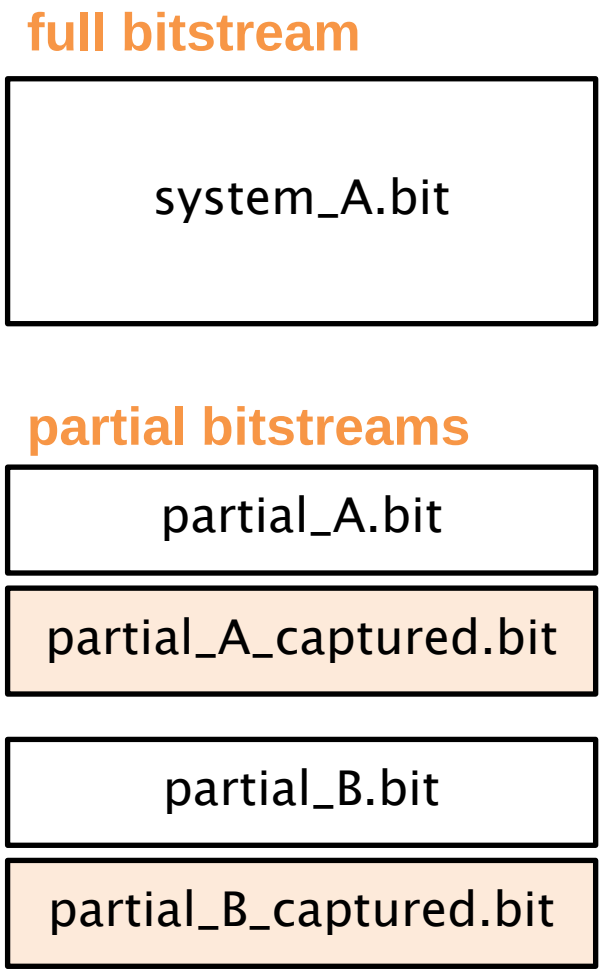
Generate bitstreams with PlanAhead

reset_after_reconfiguration attribute: 1

Partial bistreams: in external memory

e.g. on compact flash card

Our methodology is based on Xilinx PlanAhead PR toolflow.



Generate bitstreams with PlanAhead

reset_after_reconfiguration attribute: 1

Partial bistreams: in external memory

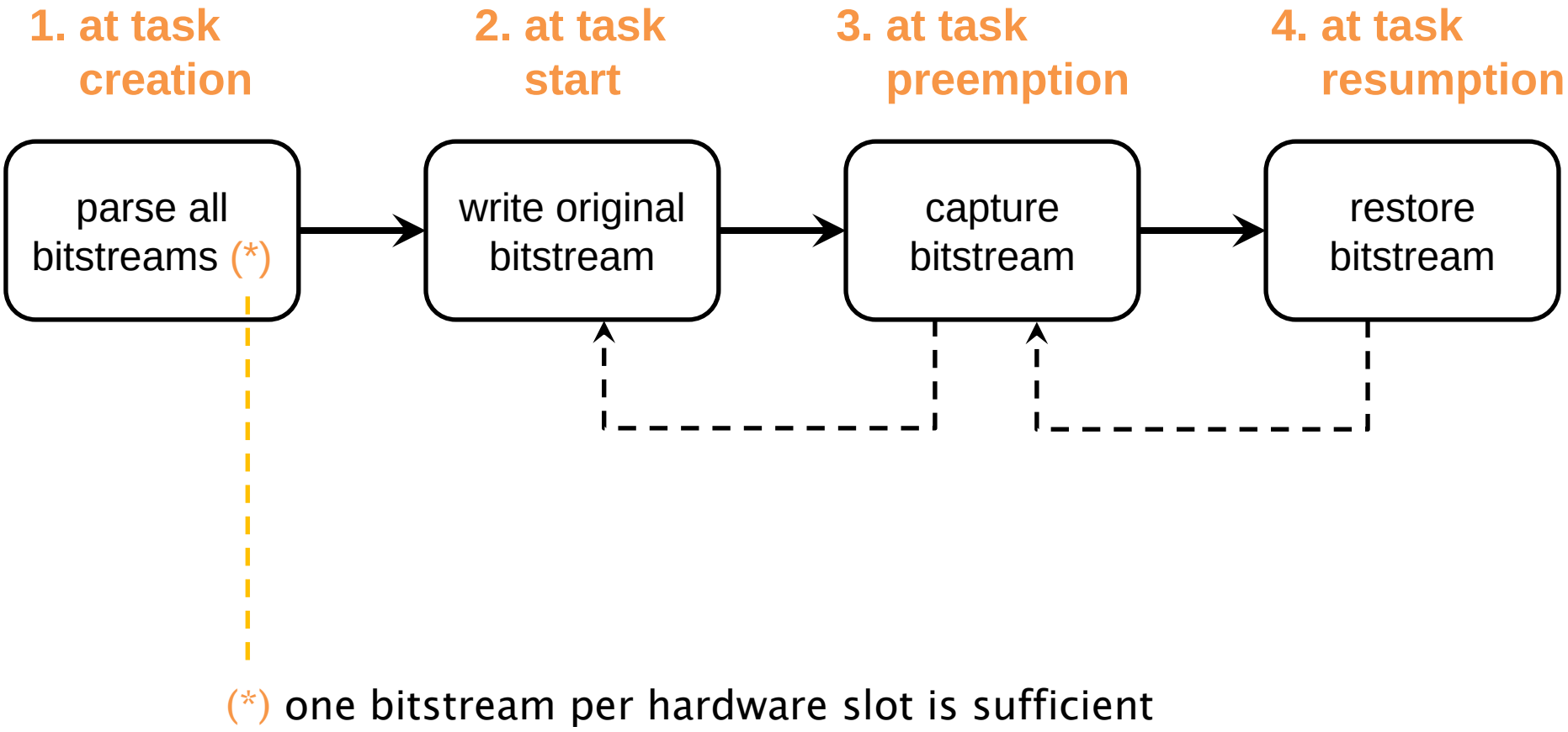
e.g. on compact flash card

Working copy of partial bitstreams

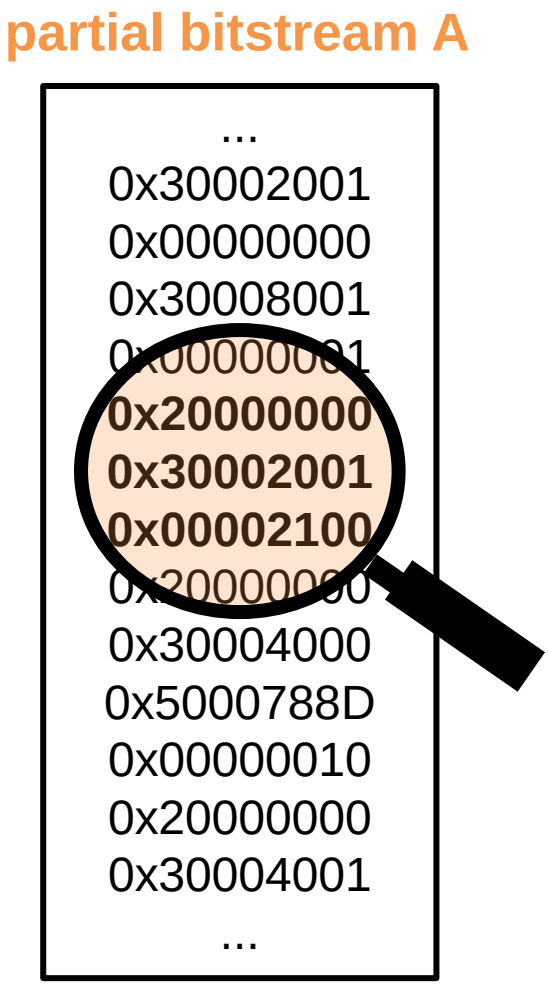
also contain the context of preempted task

initially, working copy identical to original file

Our approach initially parses bitstreams to extract metadata about configuration frames.



Xilinx bitstreams store three types of configuration frames.



Type 1: FFs, LUTs, etc.

CLBs, input/output blocks, clocks, ...

Type 2: BRAMs

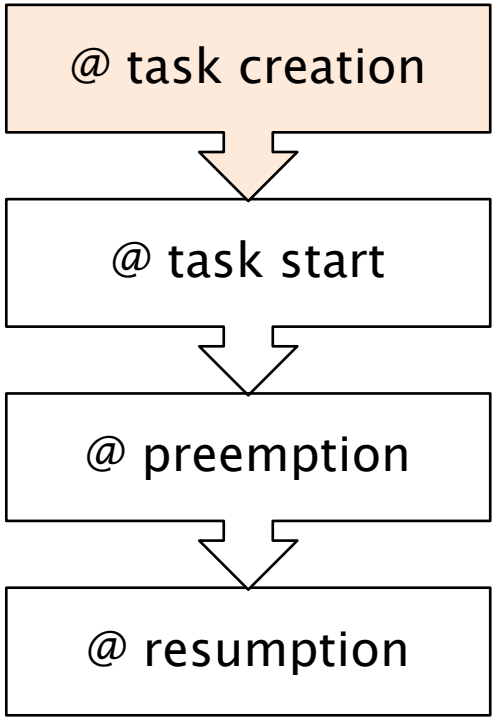
configuration of BRAM contents

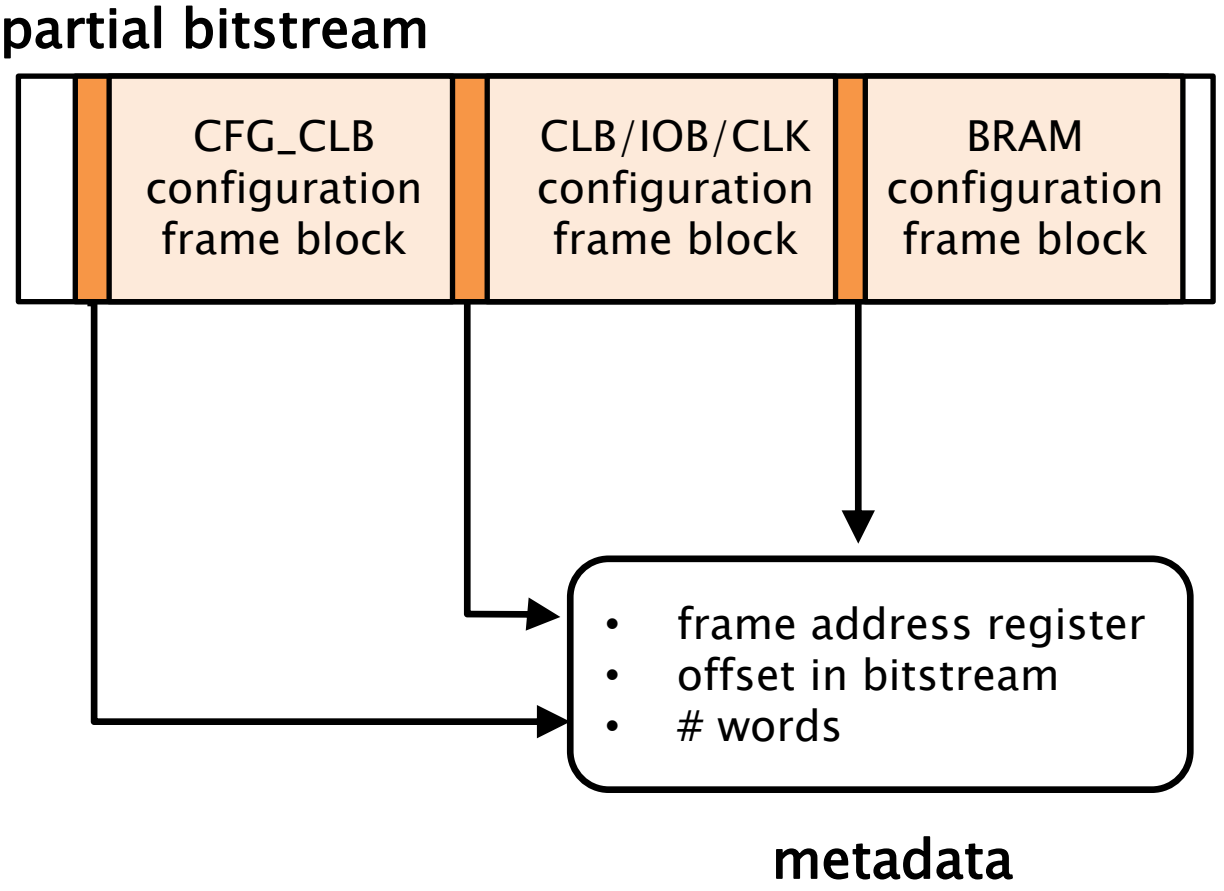
Type 3: CFG_CLB

selects which part of the FPGA is reconfigured

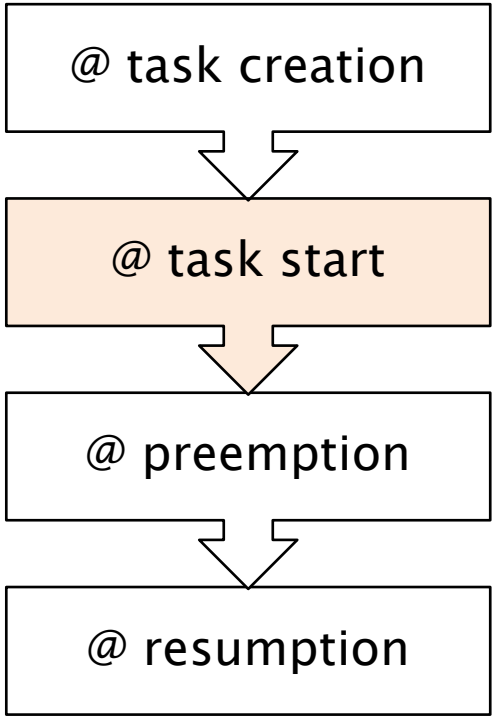
only present if «reset_after_reconfiguration=1»

We extract a list of metadata for all configuration frame blocks.





At task start, we configure the original bitstream using ICAP write commands.



No context at task start

configure original bitstream from PlanAhead

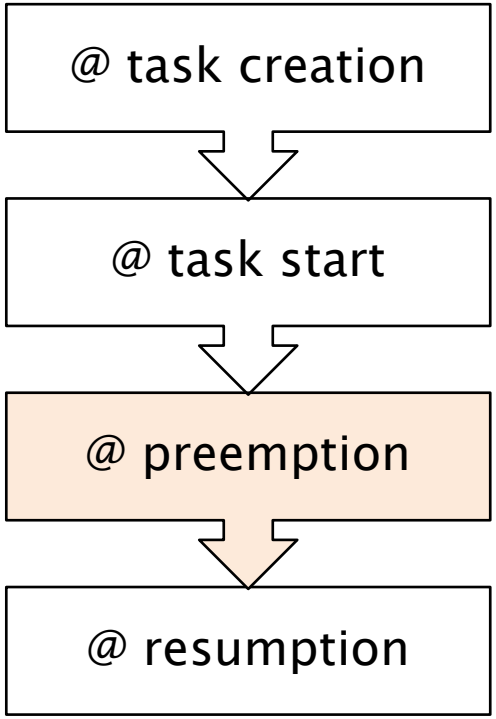
Configuration

using ICAP write commands

During reconfiguration

reset interface to/from hardware task

At task preemption, we need to capture the flip-flop states of the slot to hidden registers.



1. Prepare hardware slot

deactivate clock

2. Constrain capture command to slot

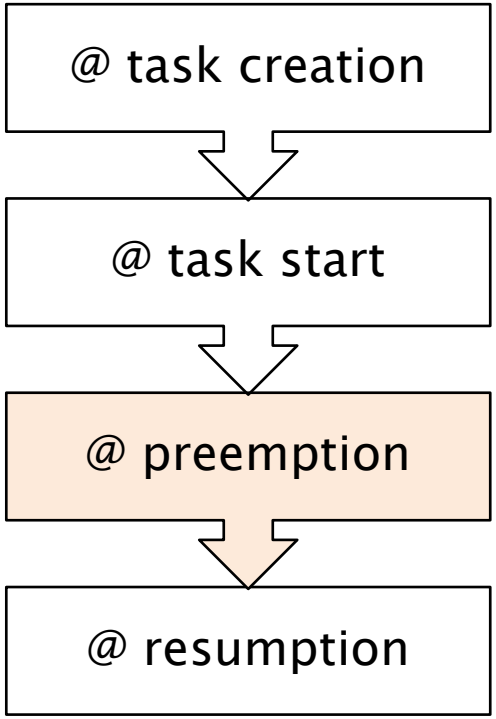
write CFG_CLB block over ICAP

3. Capture states of flip-flops

send gcapture command over ICAP

states captured to hidden registers INT0/INT1

After reading back all frames, we modify certain configuration bits for the BRAMs.



4. Read-back configuration frames

as defined by metadata (except CFG_CLB block)

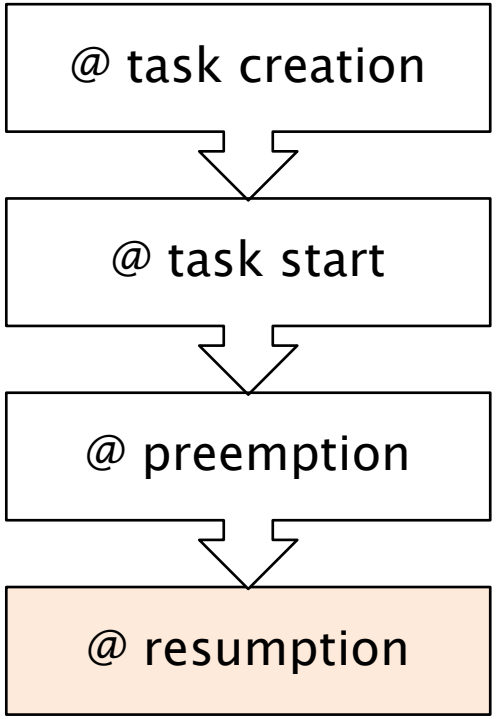
we update the working copy for these blocks

5. Modify BRAM configuration bits

otherwise BRAM contents will not be restored

not described in official Xilinx documentation

At task resumption, we write the modified captured bitstream to the hardware slot.



1. Prepare hardware slot

disable clock, reset slot and interfaces

2. Download captured bitstream

using ICAP write commands

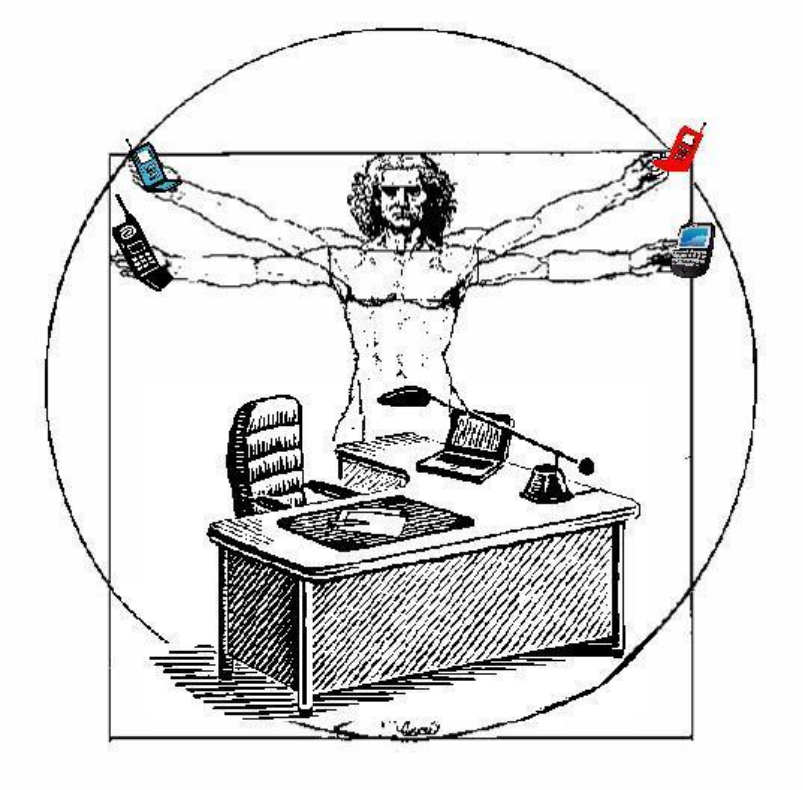
3. Trigger global/set reset (GSR)

manually trigger GSR port of startup_virtex6

grestore command in bitstream does not work

4. Resume clock and hardware thread

Preemptive Hardware Multitasking in ReconOS



Methodology

bitstream readback

ReconOS Integration

ICAP controller, scheduler

Virtex-6 Evaluation

reconfiguration overhead

ReconOS extends multithreaded programming model to reconfigurable hardware.

reconOS
Operating System for Reconfigurable Computing

www.reconos.de

Autonomous hardware threads

similar to POSIX softwarte threads

access shared memory, semaphores, mqueue

Software delegate threads

represent hardware threads

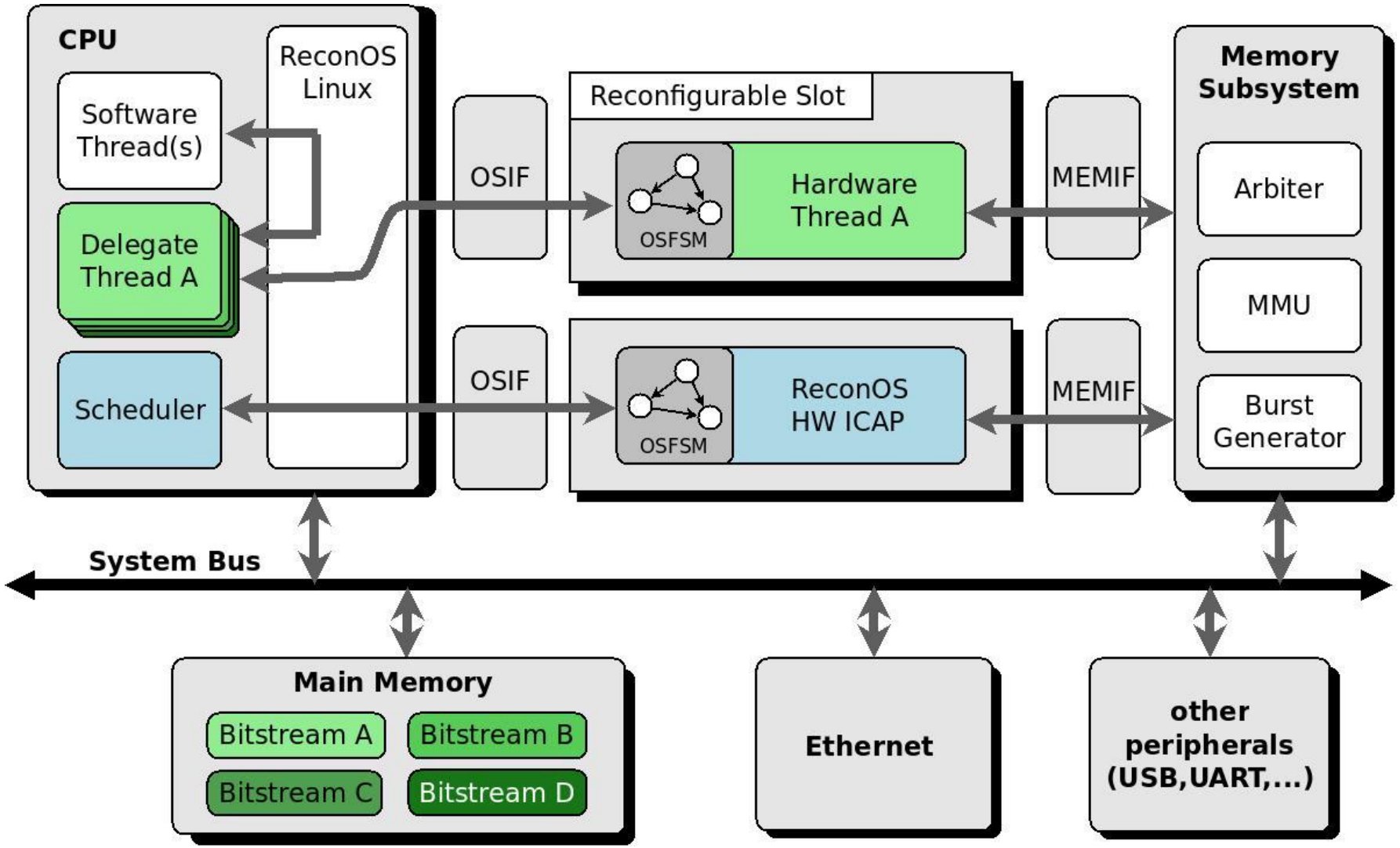
access OS services on behalf of hw threads

ReconOS @ Linux

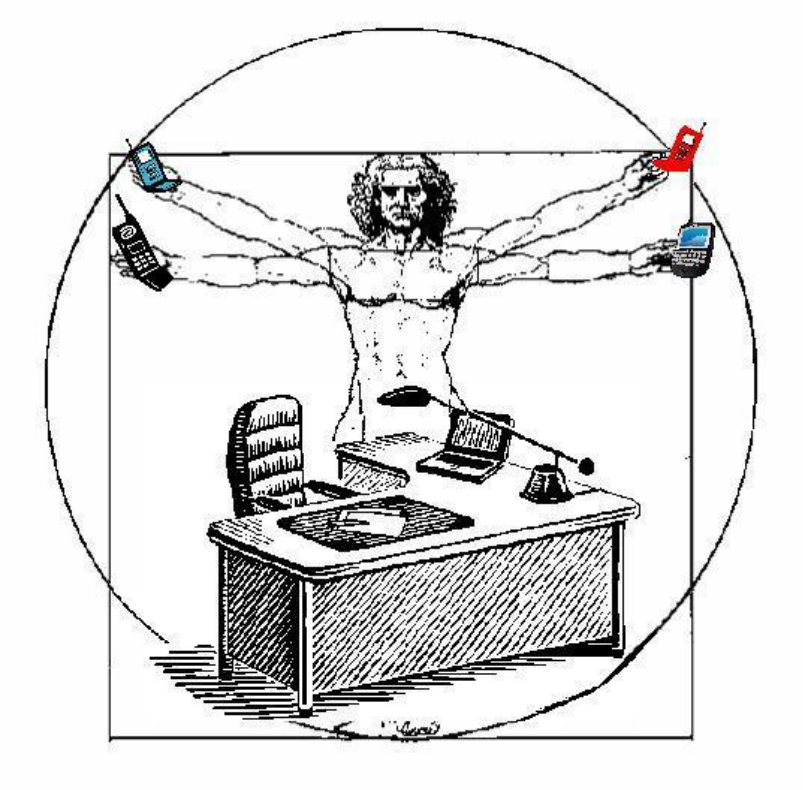
extensions implemented in user-space

support for virtual memory

Our ReconOS Architecture has one slot and one ICAP core.



Preemptive Hardware Multitasking in ReconOS



Methodology

bitstream readback

ReconOS Integration

ICAP controller, scheduler

Virtex-6 Evaluation

reconfiguration overhead

We have performed our experiments on a Xilinx Virtex-6 FPGA for two slot sizes.

Xilinx Virtex-6 ML605 board

implements ReconOS architecture

300k flip-flops, 0.4k BRAMs (36 Kbit)

Tested slot sizes

2% of FPGA area: 361 KB bitstreams

6% of FPGA area: 741 KB bitstreams

Four hardware threads

ADD, SUB, MUL, LFSR

We implemented four hardware threads to validate the context restoration.

$R_3 = R_1 \pm R_2$

ADD/SUB-Threads

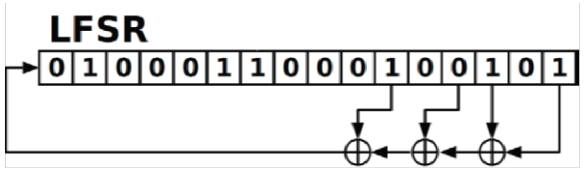
add/substract sw-defined registers

$R_3 = \sum_1^{R_1} R_2$

MUL-Thread

multiplies sw-defined registers

validate cycle-true restoration of flip-flops



LFSR-Thread

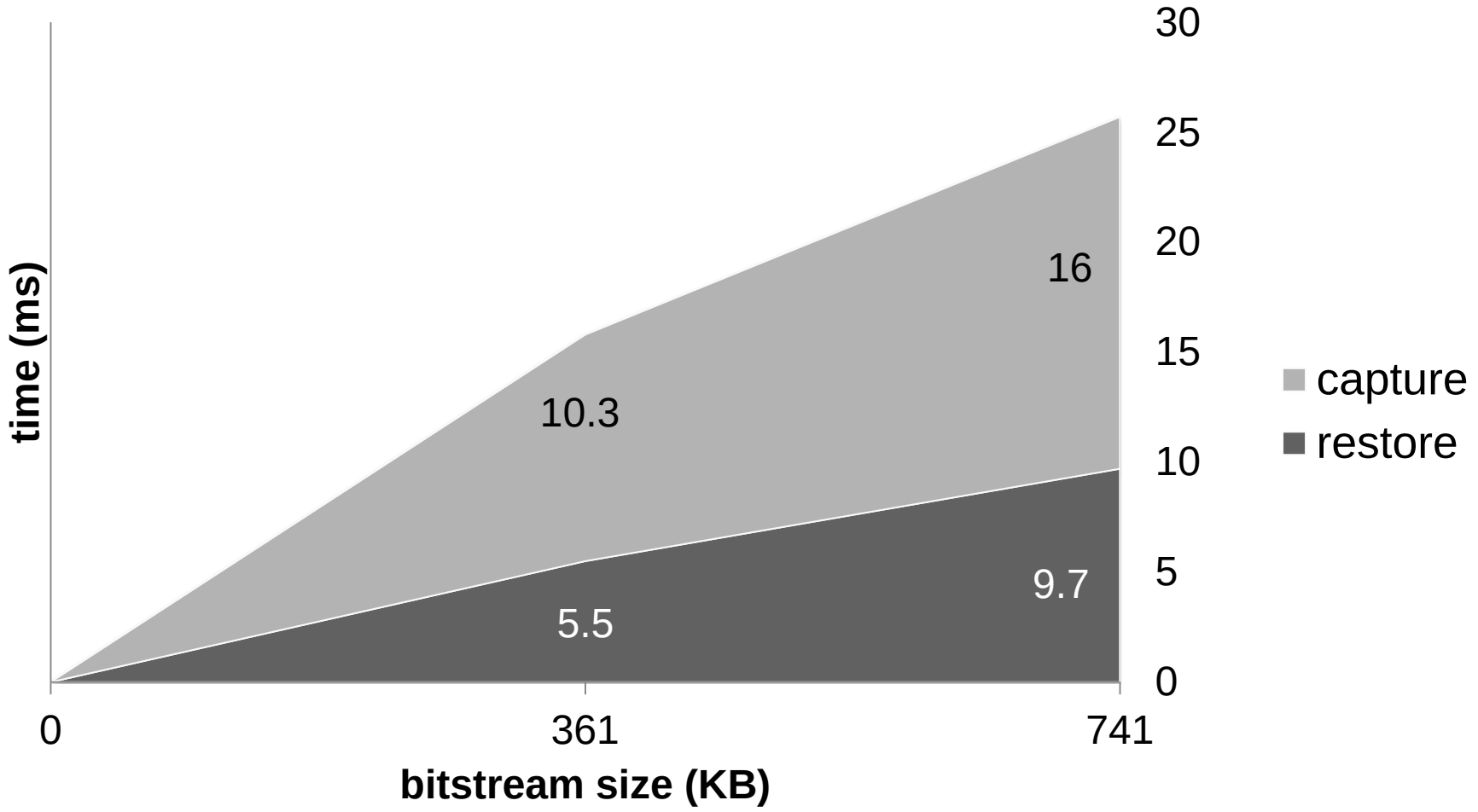
stores several LFSRs in BRAMs, shifts one LFSR

validate cycle-true restoration of BRAMs

The hardware threads only occupy a fraction of the slots, but all FFs/BRAMs are restored.

component	#FFs	#LUTs	#BRAMs
HW slot (~2%)	5.8k	2.9k	8
HW slot (~6%)	17k	8.6k	32
ADD/SUB	0.4k	0.6k	2
MUL	0.5k	0.8k	2
LFSR	0.5k	0.8k	2
ReconOS ICAP	0.3k	0.7k	2
XPS_HWICAP	0.8k	0.8k	1

We support context switches in the order of milliseconds.



The maximum number of context switches per second depends on the slot size.

Max. context switches / second

361KB: 63 switches/s

741KB: 38 switches/s

Capture/restore bandwidth

361KB: 22 MB/s

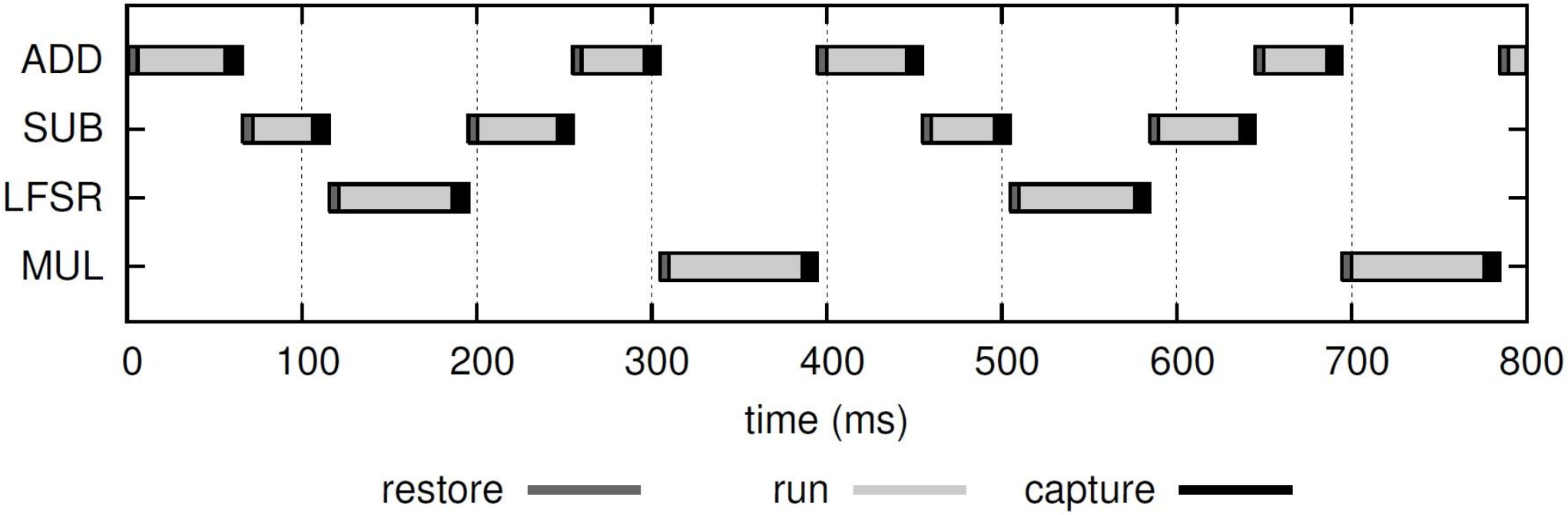
741KB: 28 MB/s

Performance bottleneck

ReconOS memory interface

Periodic tasks can be executed using preemptive scheduling algorithms.

Example measurement with fixed periodic schedule



In summary, we integrated preemptive hardware multitasking to ReconOS.

Novel methodology

preemptive hardware multitasking on Virtex-6

restore states of flip-flops and BRAMs

ReconOS integration

customized ICAP controller, sw scheduler

Experimental results

achieved bandwidth: 22-28 MB/s

multiple task swaps per second

In future work, we want to support more FPGA families and relocate tasks.

Restore further FPGA resources

LUT-RAMs, DSP blocks

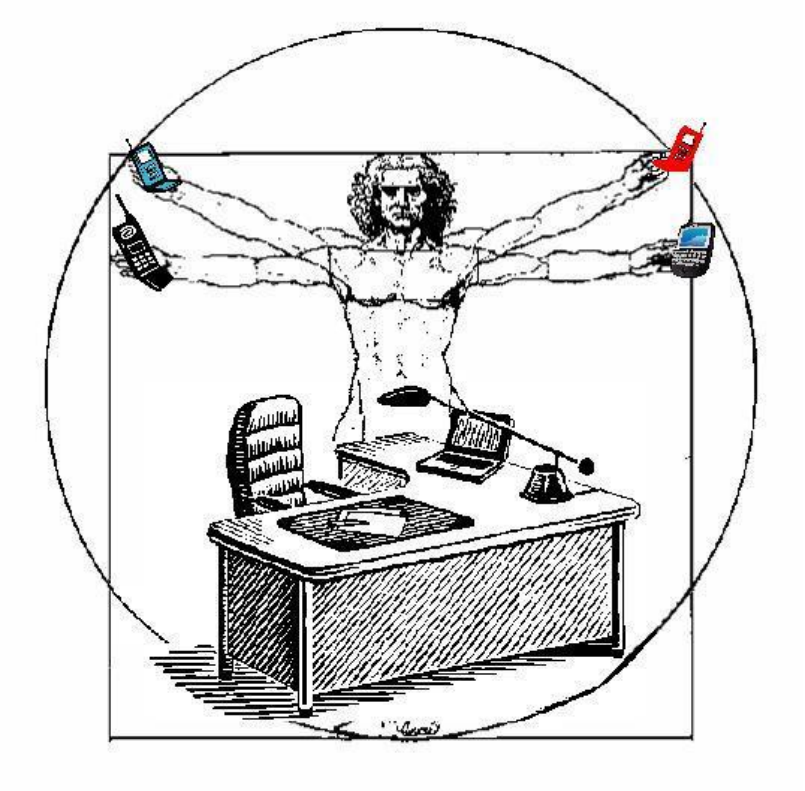
Support more FPGA families

e.g. Virtex-7, Zynq

Task relocation

migrate threads between slots

Preemptive Hardware Multitasking in ReconOS



Markus Happe

mhappe@ethz.ch

ETH Zürich

ARC'15, Bochum

15th April 2015