Fast dynamic and partial reconfiguration Data Path with low Hardware overhead on Xilinx FPGAs

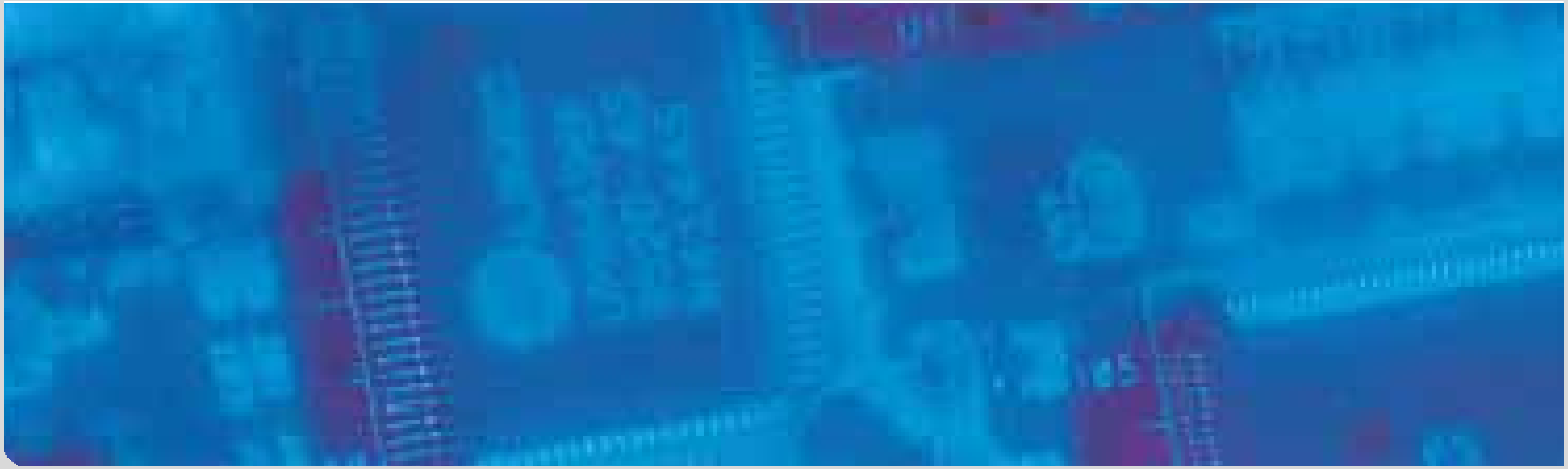
Michael Hübner[1], Diana Göhringer[2], Juanjo Noguera[3], Jürgen Becker[1]

[1] Karlsruhe Institute of Technology (KIT), Germany

[2] Fraunhofer IOSB, Germany

[3] Xilinx Inc., Dublin

Institut für Technik der Informationsverarbeitung (ITIV)

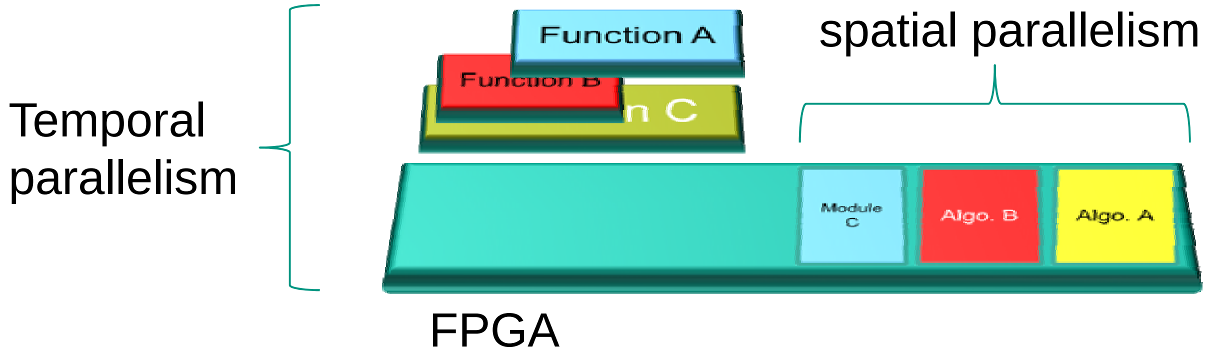


Outline

- Introduction and motivation
- Related work
- Concept of Fast Simplex Link (FSL) internal configuration access port (ICAP)
- Realization and results
- Conclusion and future work

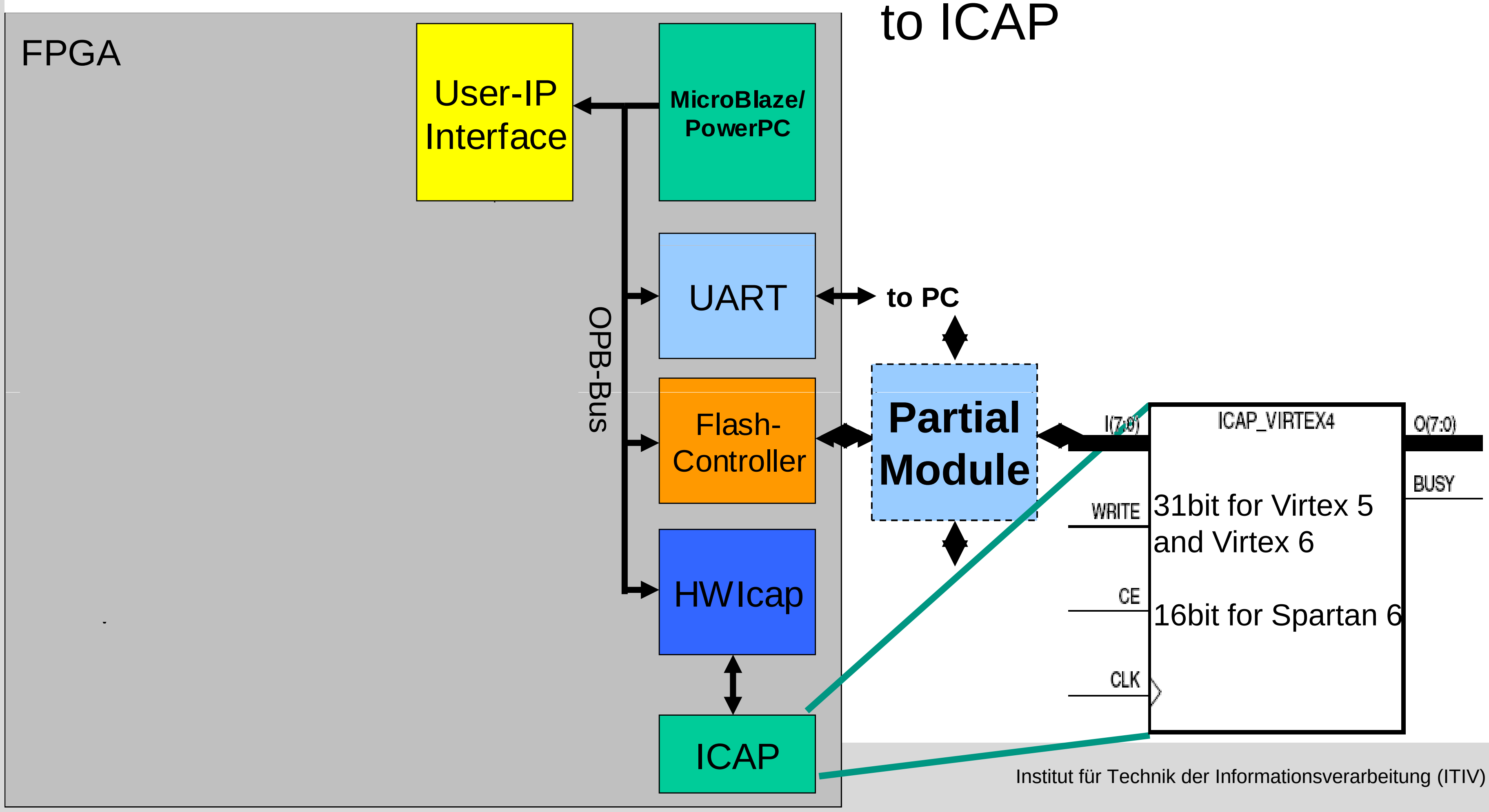
Introduction and motivation

- Dynamic and partial reconfiguration: *"parts of a configuration can be substituted while other parts stay operative without any disturbance"*
- Spatial and temporal partitioning exploitation to increase performance and to reduce power consumption
- In a processor based design (MicroBlaze), the configuration access port is one of the "devices" on the OPB or PLB bus
  → Why is it not a part of the processor's microarchitecture?



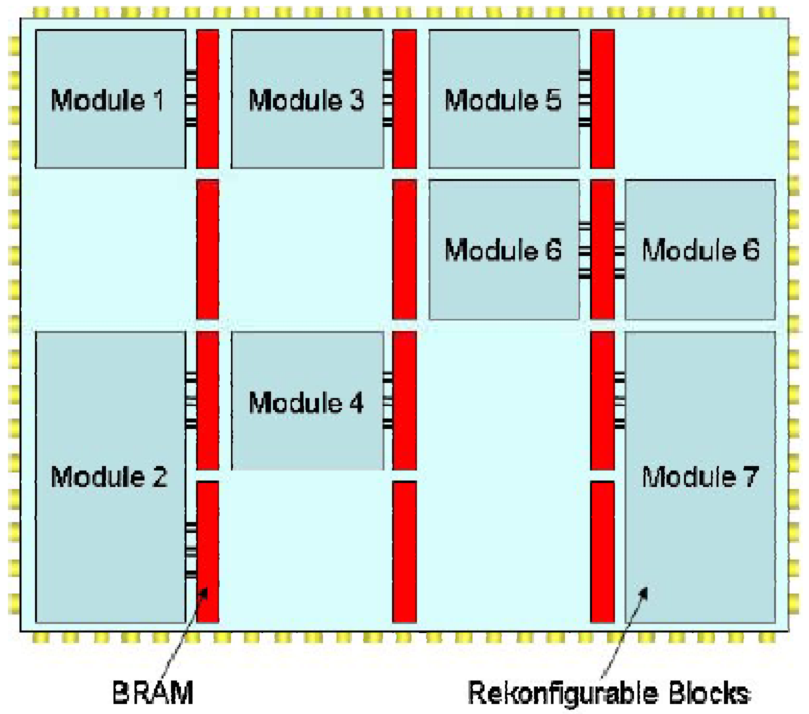
Traditional usage of the ICAP: 1. Dynamic Reconfiguration

- ICAP was traditionally used for run-time adaptive systems:
  → Loading partial bitreams from external memory, transfer to ICAP

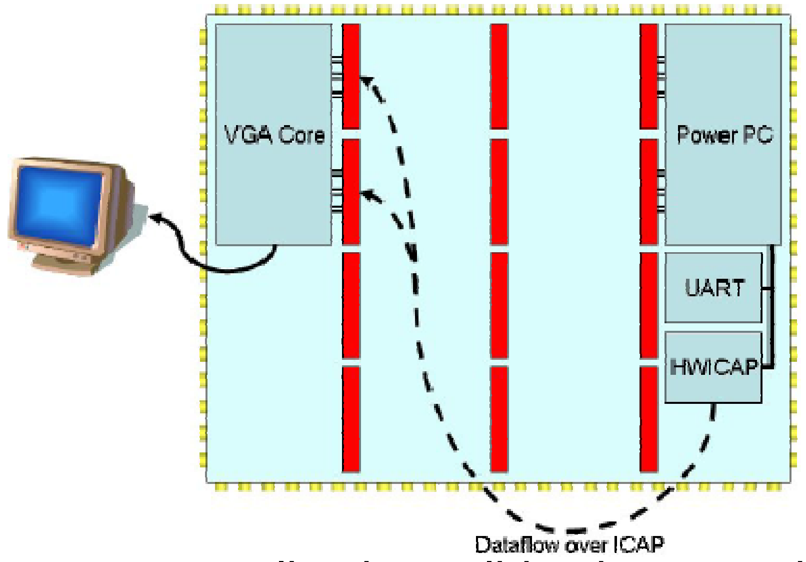


Traditional usage of the ICAP (cont) : Data transfer through read- and writeback

- ICAP was used to transfer data from one BRAM to another
  → Reduction of signal line utilization, novel degree of freedom



Example with 7 encapsulated modules

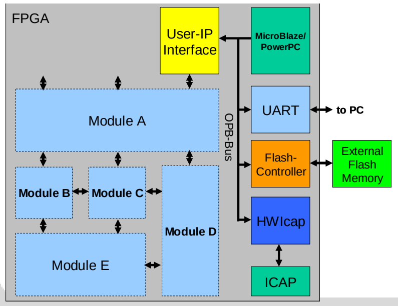


Test application: Slide show on Virtex 2
VGA core has no connection via signal line to PPC

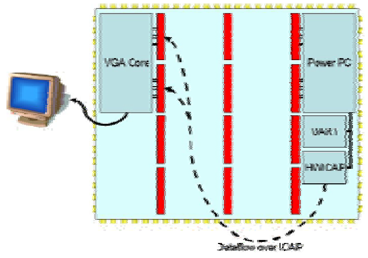
This example: Sander et. Al.: „ Data Reallocation by Exploiting FPGA Configuration Mechanisms", RAW 2008, April
Very nice extension:
Shelbourne et. Al.: "MetaWire: Using FPGA Configuration Circuitry to Emulate a Network-on-Chip", FPL 2008, September

ICAP is more than a configuration port…

- ICAP can be used in different modes:
  - Access port to the reconfigurable logic, consuming configuration data
  - Access port to the reconfigurable logic, producing configuration data (e.g. Readback of configuration data for safety reasons (bit flips etc.)
  - Access port to processing elements already configured on the FPGA, consuming (write mode) data to be processed
  - Access port to processing elements already configured on the FPGA, producing (read mode) data which were processed

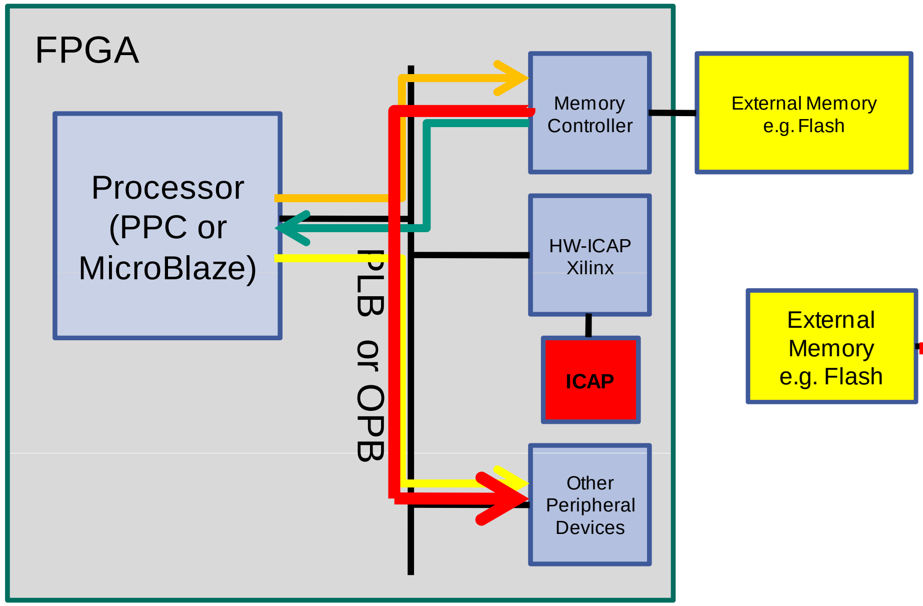


In general two modes of operation:
1. for hardware reconfiguration purposes
2. for data transfer purposes



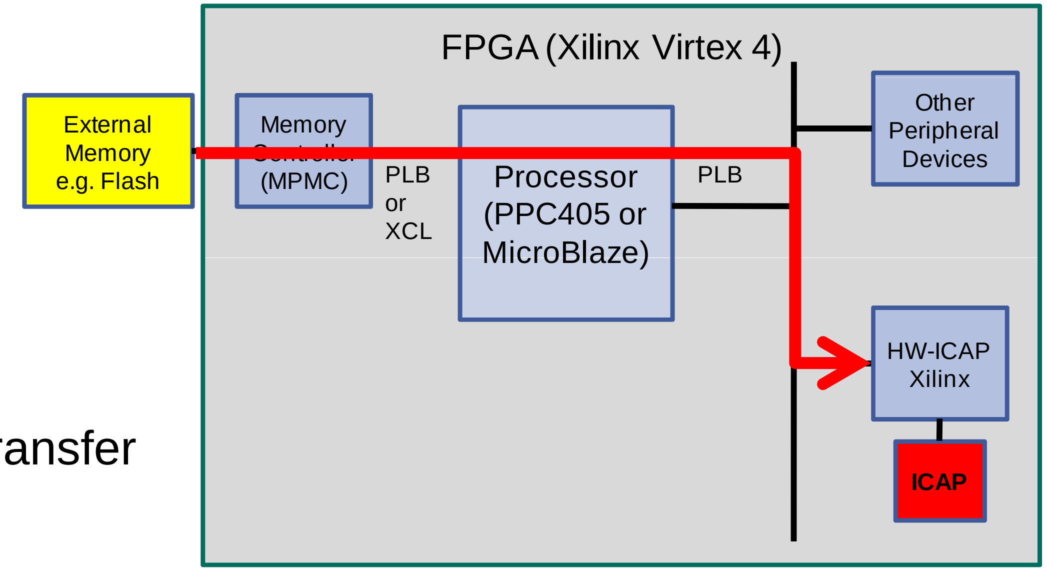
Realization alternatives for processor – ICAP connection

- Several approaches exist where a processor triggers the reconfiguration of an accelerator



Single bus version enables DMA transfer for bistreams to the ICAP

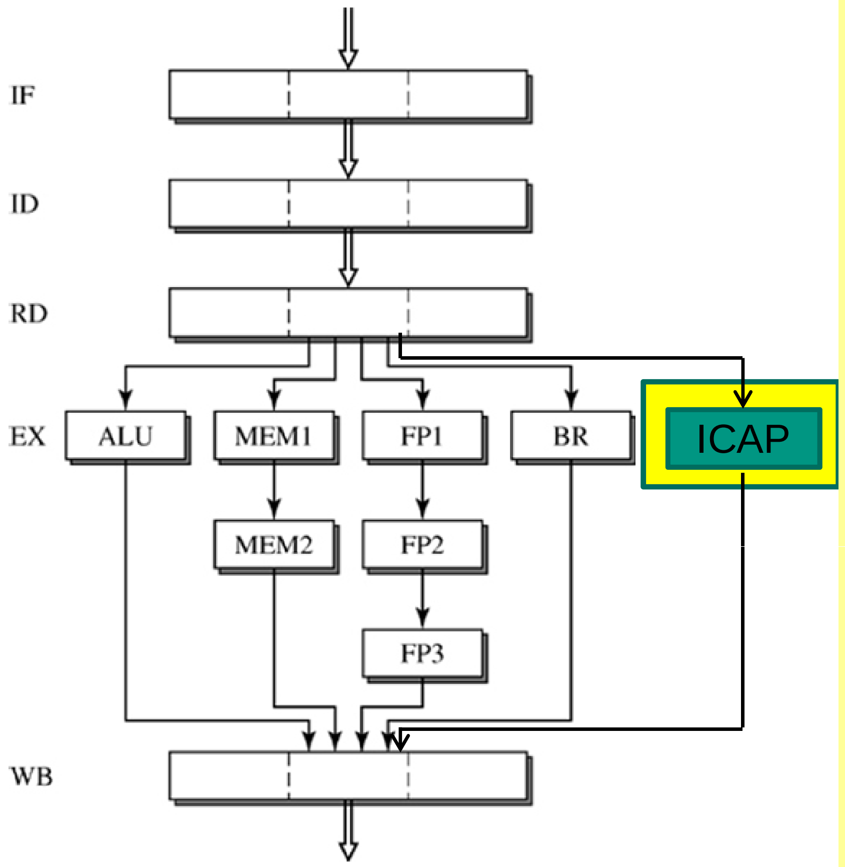
Dual bus version (newest version)
Efficient usage for MicroBlaze



Numerous previous work, e.g.:
Blodget et. Al.:"A Lightweight Approach for Embedded Reconfiguration of FPGAs", DATE 2003
Claus et. Al.: „A multi-platform controller allowing for maximum dynamic partial reconfiguration throughput", FPL 2008

Novel exploitation possibilities of the ICAP in adaptive microprocessor architectures: The i-Core

- Lets assume ICAP is integrated into the processor pipeline



Extended version based on the picture used by Prof. Lizy Kurian John, Univ. Austin, Texas

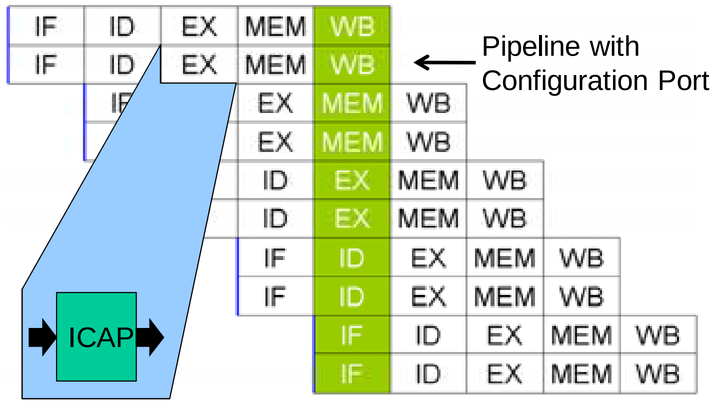
That would mean:

- Processor commands are reserved for the ICAP:
  Reconfiguration mode:
- ICAP write config.
- ICAP read config.
  Data transfer mode:
- ICAP write process data
- ICAP read process data

- **The ICAP is included directly into the data path of the processor**
  → lowest delay for data transfer
  → see ICAP from „the software point of view“ and write simple programs for accessing it

Exploitation of the novel concept

- <u>The novel concept increases the flexibility of a FPGA based processor tremendously</u>
  - The ICAP as **data sink and source** can be seen as a **multipurpose ALU**
  - From the user **(programmer)** point of view the hardware complexity is hidden through the provided libraries
  - Accessible with standard C construction
  - Further hardware abstraction which definitely will increase the acceptance of run-time adaptive hardware



Legend:	
-IF:	Instruction fetch
-ID:	Instruction decode
-EX:	Execute
-MEM:	Memory access
-WB:	Writeback

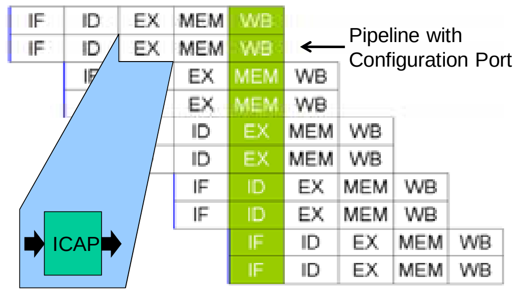
Exploitation of the novel concept (cont)

- <u>The novel concept enables the run-time adaptation of the processors microarchitecture:</u>
  - Realized instruction (within the ISA) be reconfigured at runtime and realizes therefore a dynamic reconfigurable instruction set processor
  - In general: An adaptive microarchitecture is possible:
    - **Power and energy reduction via pipeline balancing**
    - **Using ipc (instruction per cycles) variation reduce power consumption**
    - **Dynamic instruction level parallelism pipeline adaptation**
    - **Adaptive issue queue for reduced power at high performance**

    (Please see in the our paper the references, they did not use this novel approach!)
  - Decentralized processor approach: ICAP connects cores on any position of the chip

→ Novel quality of processors: The **i-Core** provides the run-time adaptation of the microarchitecture

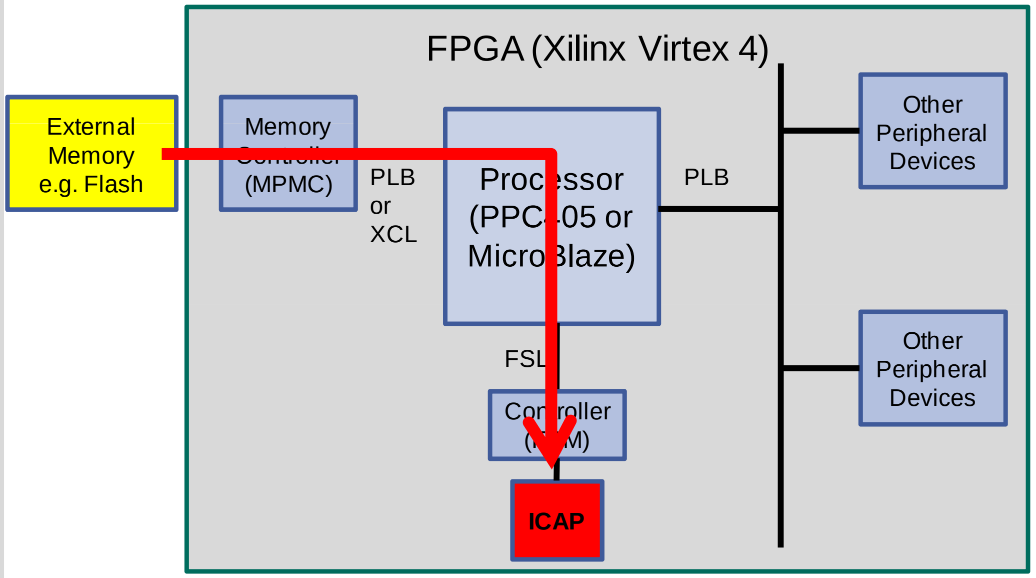
An example from a real experiment:
adaptation of pipeline from 5 to 3 stages
reduction of **90mW** power consumption!
(Publication under review: ReCoSoC 2010)



Legend:
-IF: Instruction fetch
-ID: Instruction decode
-EX: Execute
-MEM: Memory access
-WB: Writeback

(One of the) First steps to the i-Core: FSL-ICAP Hardware view

- Connecting the ICAP **as near as possible** to the processor core of MicroBlaze: the FSL connection provides a connection with the latency of **only one clock cycle**



Concept:
Transfer the configuration as well as data to be processed by IP cores through the processor. Simple programming model, no hardware knowledge required.

Approach does not target highest perfromance in data throughput. It focuses to embedd the ICAP into C world.
But side effect: 2-3x speedup in comparison to XPS ICAP (through reduction of required clock cycles)

(One of the) First steps to the i-Core: FSL-ICAP Software view

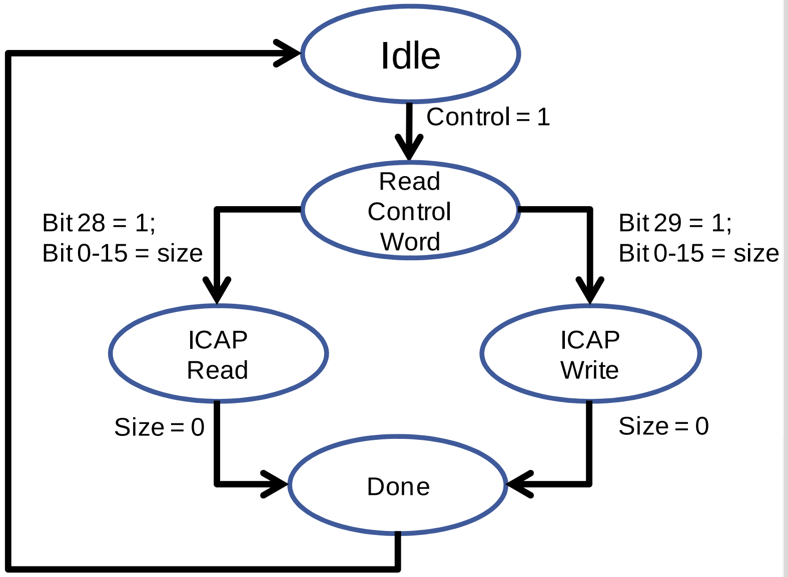
- Accessible ICAP with reserved commands (e.g. CPUTFSL and PUTFSL) out of C code: Hiding the hardware complexity increases development efficiency

```
/* Function for Bitstream transfer from external memory to FSL- ICAP */
        int loadBitfromSRAM(Xuint32 baseaddr, Xuint32 size)
        {
                Xuint32     conf_word, size4, output_0, i;

                /*point to Addr in Ext. Mem    */
                Xuint32 *pointword = (Xuint32*) baseaddr;

                size4 = size >> 2; // Get size in words
                output_0 = (size4<<16) | command;

               /* write bitstream to FSL_HW_ICAP */
                cputfsl(output_0, 1); //bit 0 to 15 = size,
                                                    bit 28 to 31 = command

                for (i = 0; i < size4; i++)
                {/* write memory content to FSL */
                                conf_word = pointword[i];
                                putfsl(conf_word, 1);}
          return 0;}
```

Sample code for reconfiguration access
Similar for data transfer mode

FSL ICAP commands

Bit 28..31	Mode	Description
0001	Reset	Back to Reset state
0010	ICAP Status	Send status of ICAP to processor
0100	ICAP write	Write configuration data
1000	ICAP read	Readback data from configuration memory



FSM for ICAP control purposes realized within the FSL ICAP controller

Implementation results: Utilization and performance

- We compared the results with already exisiting ICAP hardware drivers (see references in the paper)
- **Goal was not to gain in performance, the benefit of a *“new ICAP thinking”* is in the foreground**

Technical Data	FSL-ICAP	XPS ICAP [6]	ICAP [8]
Utilized Slices	78 + 44 (for 2 FSLs)	2868	637
Utilized BRAM	0	0	2
FPGA Board	Xilinx ML405	Xilinx ML405	Xilinx ML410
External Memory	DDR SDRAM (32-bit interface)	DDR SDRAM (32-bit interface)	DDR2 SDRAM (64-bit interface)
Processor	µBlaze, PPC	µBlaze, PPC	PPC
Throughput with µBlaze (MByte/s)	25,89	12,79	n/a
Throughput with PPC (MByte/s)	28,28	8,60	295,4

→ FSL ICAP is easy to use and comparably fast

Conclusion and future work

- The ICAP can be used in different modes of operation
- It depends from the perspective, what impact these modes have to a system’s realization:
  - The ICAP in the processors data path, enables new degrees of freedom for the adaptivity of the processor while run time
  - Previous work shows, what optimizations of a processor’s microarchitecture enable in terms of reduced power consumption and increased perfomance
  - The reconfigurable Application-specific instruction-set processor (ASIP) enables to provide a “multipurpose” but “application tailored” processor core
  - We will call this approach i-Core (related to the latest results in programming paradigm: Invasic Computing (see reference in the paper)

Conclusion and future work con’t

- One of the first steps to the i-Core was the FSL ICAP
- FSL ICAP targets the idea of hiding hardware complexity from the developer
  - Simple C libraries enable the access to ICAP as sink and source for configuration data as well as for data to be processed by IP cores
  - The IP has a very low footprint in comparison to other solutions
  - FSL ICAP can easily be adapted to other Xilinx FPGAs:
    e.g. we adapted the FSL ICAP quickly and successfully to the requirements of the Virtex 5 and Spartan 6 FPGAs for our demonstrators
- Next steps are the exploration of the possible adaptation mechanisms related to the processor microarchitecture
  (e.g. manipulation of the pipeline width and depth…etc.)
- A further step is to use the processor at the FSL ICAP to run an “intelligent” ICAP related OS (don’t miss talk of Mrs. Göhringer later ☺)

Thanks a lot for your attention!

- I hope that you are interested in this work.
- Please share your ideas with me

- Contact:
- Dr.-Ing. Michael Hübner
- Karlsruhe Institute of Technology (KIT)
- Institut für Technik der Informationsverarbeitung (ITIV)
- Email: michael.huebner@kit.edu
- Skype: huebner_michael