RUHR-UNIVERSITÄT BOCHUM

RUB

Adaptive processor architectures for detector applications

Prof. Dr.-Ing. habil. Michael Hübner

Chair for Embedded Systems in Information Technology (ESIT)

Faculty of Electrical Engineering and Information Technology, Ruhr-University of Bochum, Germany

E S I T

www.esit.rub.de

Content

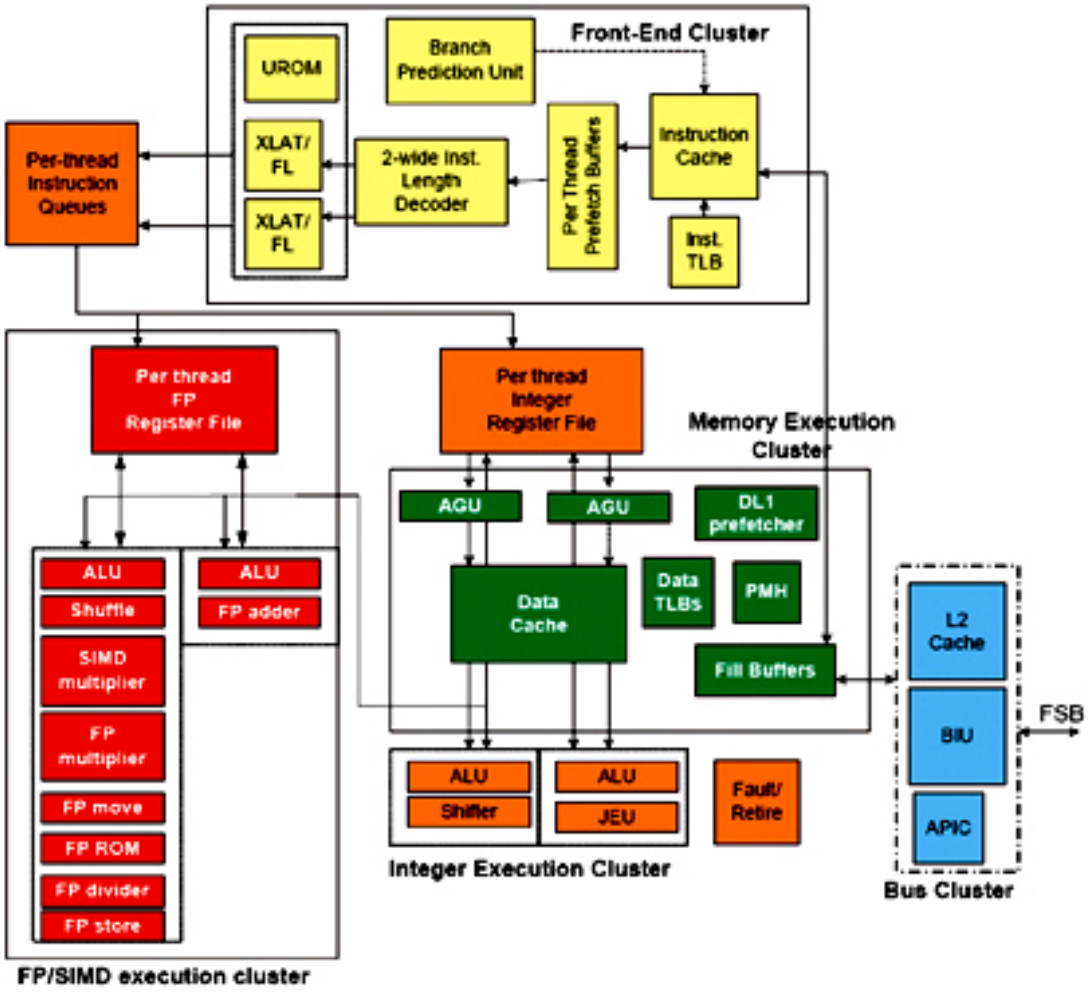
- Introduction of traditional and new processor architectures
    - Standard RISC processor ATOM
    - Standard DSP processor from Texas Instruments
    - Novel heterogeneous processor including RISC and DSP
- Motivation of an adaptive processor concept
    - Requirements using the processor for detector application
    - Integration of the sensor data into the processor datapath
    - Description of the extended datapath
- How to develop and simulate an adaptive processor
    - High level design flow for processors
    - Benchmarking and emulation
- Conclusion and outlook

(Non) adaptive processor architectures RISC / CISC

- Traditional proce **dth of application wit ive**
- Example: Intel A
  - 45nm tec 4Watt
  - Was deve (here the as performance)
  - Supersca ipeline stages

  → Lower p t

  → Less po eordering unit
- Other process slide!



From: Taking a closer look at Intel's Atom multicore processor architecture, Stephen Blair-Chappell

(Non) adaptive processor architectures DSP

- Traditional processor, developed for **dataflow oriented applications**
- Example: Texas Instruments DSP Processor
    - Developed for data flow oriented applications
      (here the data throughput plays a important role, also power consumption)
    - VLIW architecture (256bit), deep pipeline stages (can be over 20)

    → Data dependencies are solved by the compiler during design time
    → Application scenario with low control flow (therefore parallelization during design time by compiler) **but**
    → **every control flow, reduces the performance tremendously**

- So why not an adaptive processor, or at least providing a heterogeneous architecture

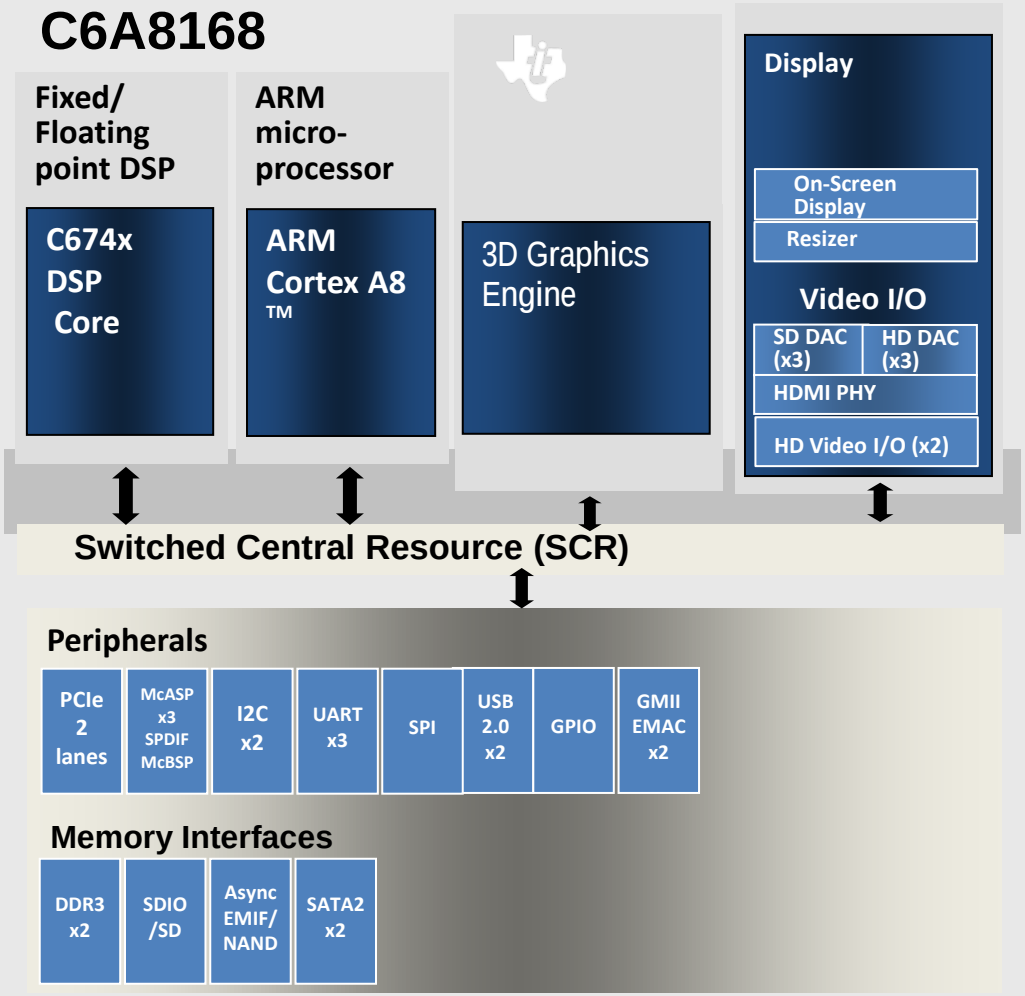
The hybrid way: C6A816x Integra™ DSP + ARM processor

Cores
- C674x™ Programmable, Floating/Fixed Point DSP Core up to 1.5 GHz
- ARM Cortex A8™ (MPU) up to 1.5 GHz
- 3D Graphics Engine – up to 27M polygons/s (C6A8168 only)
- Display Subsystem – interface to multiple, simultaneous HD displays

Memory
- ARM: 32KB L1I-Cache, 32KB L1 D-Cache, 256K L2
- DSP: 32KB L1I-Cache, 32KB L1 D-Cache, 256K L2
- External Interfaces: Two DDR3-1600 Controllers and NAND

Peripherals
- Gigabit EMAC x2
- USB 2.0 Ctlr/PHY x 2
- PCIe 2.0 – x1; Supports 2 lanes
- SATA 3.0Gbps supports 2 external drives
- HDMI 1.3 Tx
- SD/SDIO
- McASP x3, McBSP
- SPI, GPIO, I2C, UART, EMAC

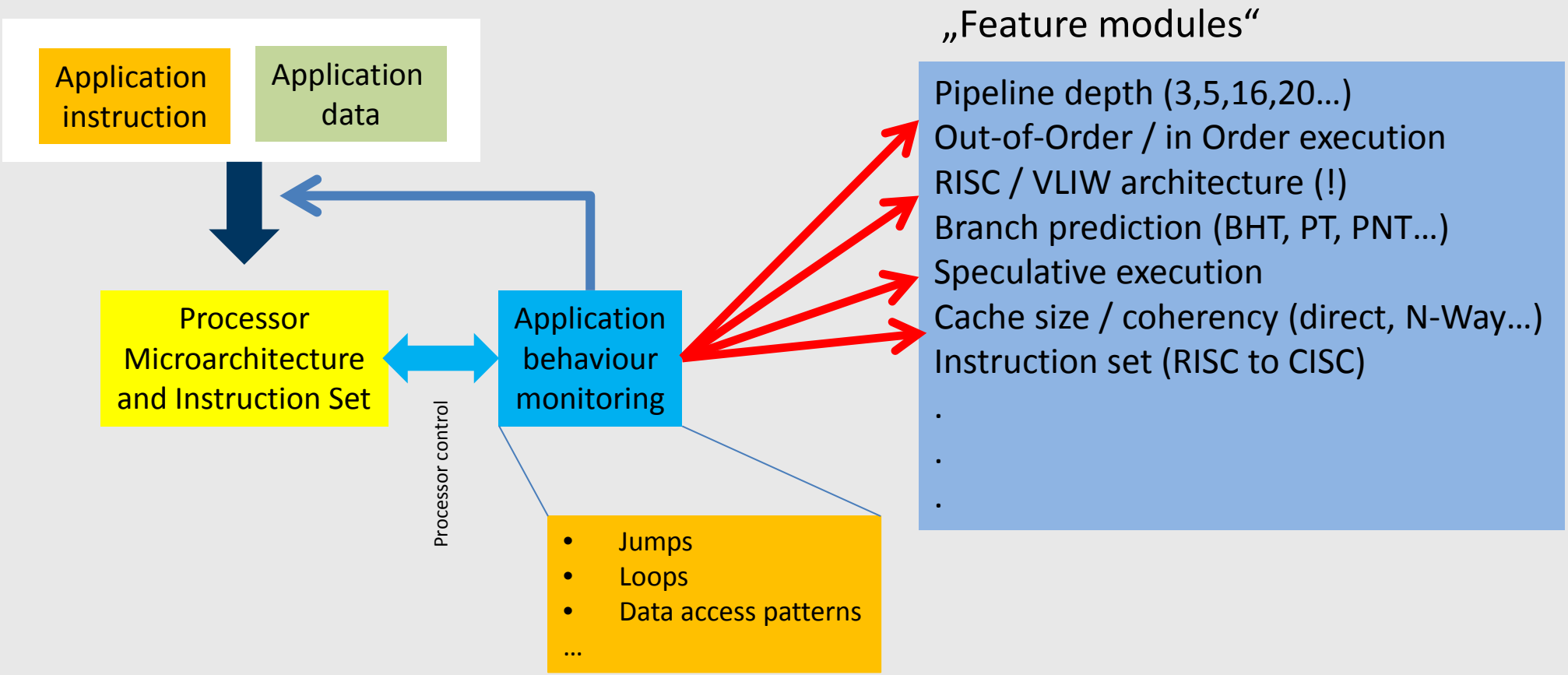
Power
- Total Power – Typical 5-6W



From: http://newscenter.ti.com/index.php?s=32902&item=632#Documents:86

The adaptive processor

- Application depend on the “position” of the processor: e.g. ATLAS (trigger level)
- …from billions of events to hundreds, from petabytes to hundreds of megabytes…
- Different “requirements” of the application with different control / data flow overhead, or even both in separate phases of the application

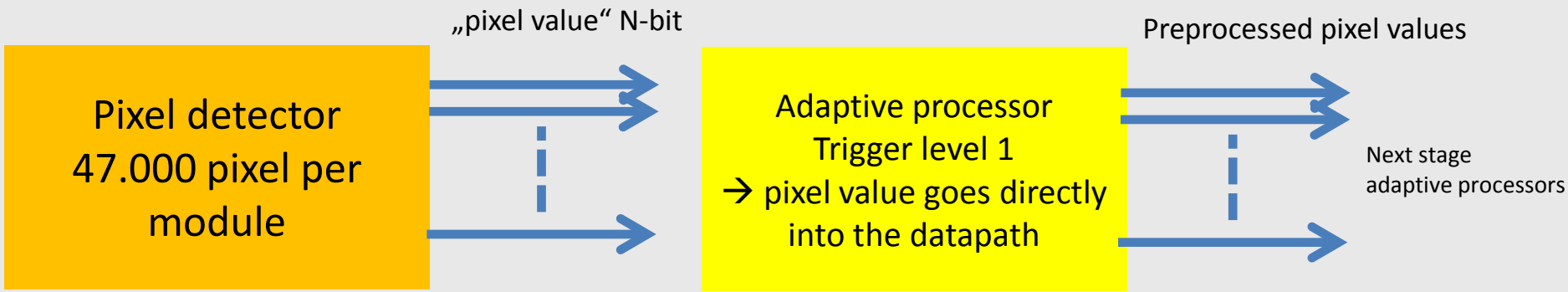


The adaptive processor: Advantages

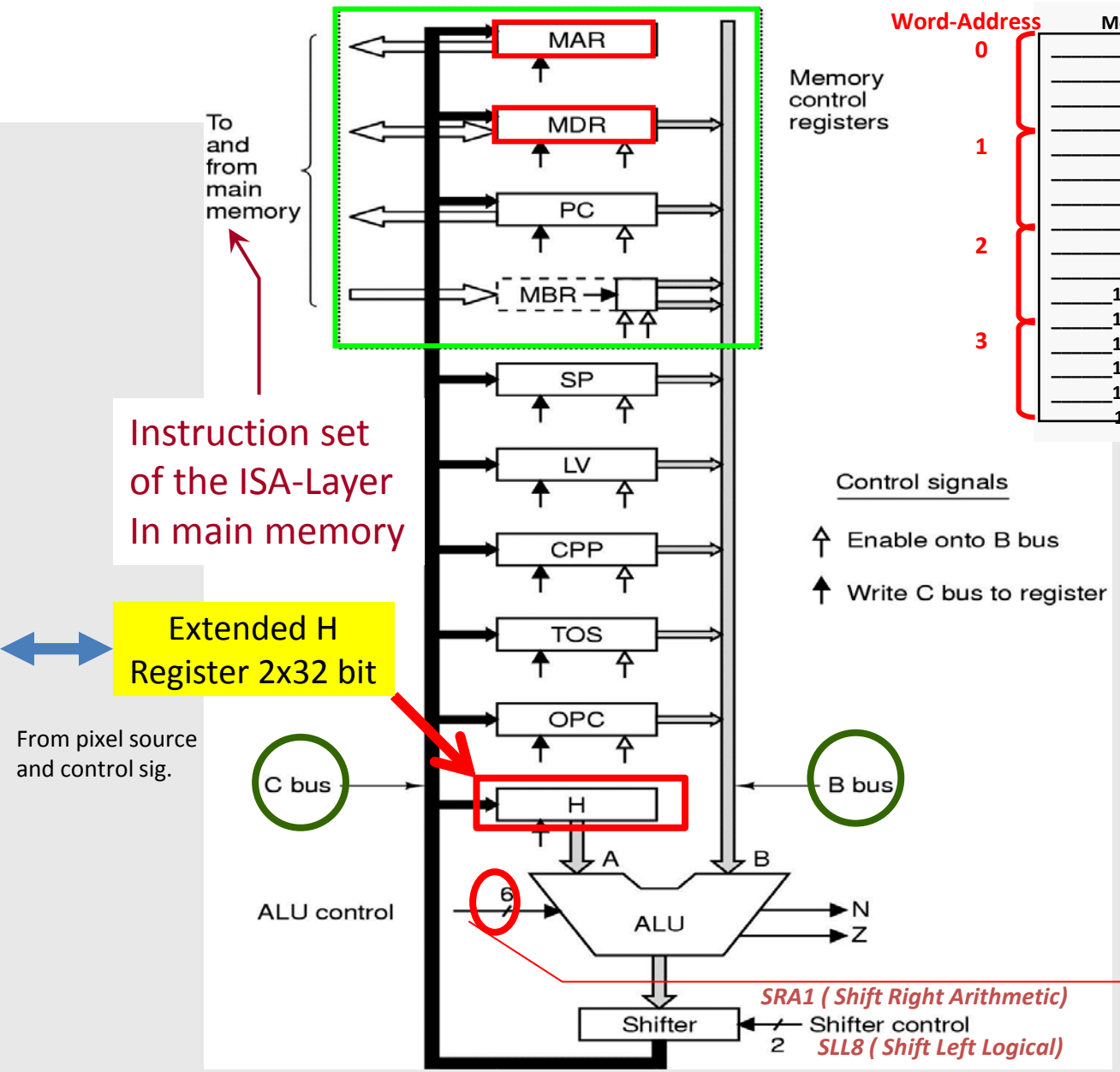
- The adaptive processor is able to “react” to application requirements
- It can be deployed without modification in many application (it starts as “general purpose processor and ends as application specific processor”
- The monitoring can be adapted to many signatures, even a “history” can be stored and reused (keyword case based reasoning from AI)

- **And:** it combines the methods of embedded computing with the ones from supercomputing (keyword multicore, power saving modes etc.)

- **BUT:** How can a processor be as near as possible to the place, where data are produced?
- **E.g. ATLAS level 1: Tight coupling of the processor to the sensor**



Data path of a processor



ALU Arithmetic Logic Unit
CPP Constant Pool Pointer
H Hold Register
LV Local Variable Pointer
MAR Memory Adress Register
MBR Memory Buffer Register
MDR Memory Data Register
N N=1 bei ALU-Inhalt <0
OPC Old Program Counter
PC Program Counter
SP Stack Pointer
TOS Top of Stack
Z Z=1 bei ALU-Inhalt =0

F_0	F_1	ENA	ENB	INVA	INC	Funktion
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	0	1	0	0	$\overline{B}$
1	1	1	1	0	0	A+B
1	1	1	1	0	1	A+B+1
1	1	1	0	0	1	A+1

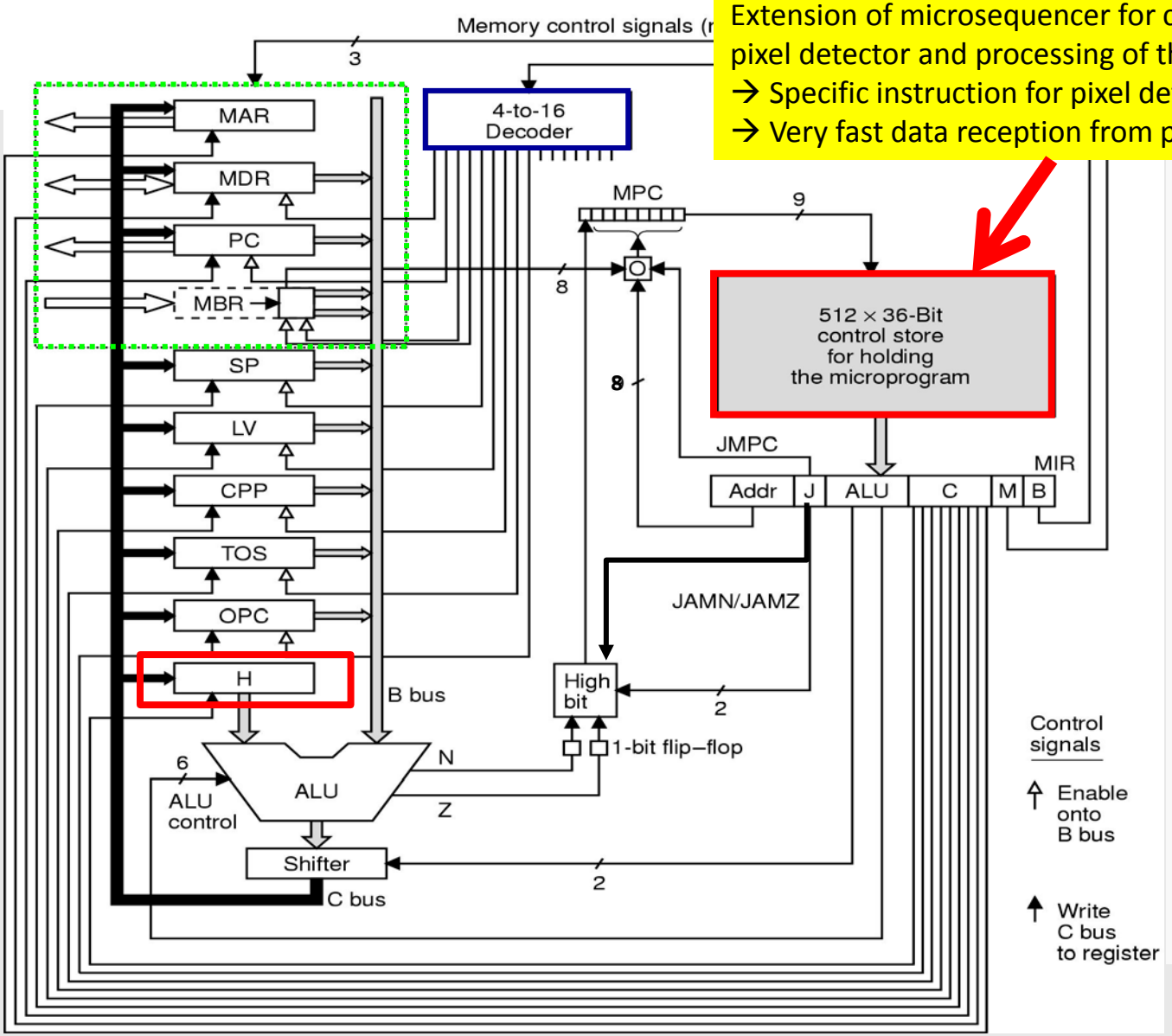
Specific ALU operations

1	1	0	1	1	1	B–1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	–1

Bildquelle: Tanenbaum, Structured Computer Organization

Example pixel detector specific microarchitecture

Extension of microsequencer for controlling the
pixel detector and processing of the pixel data directly in the data path
→ Specific instruction for pixel detector hardware
→ Very fast data reception from pixel sensor



(Constant Pool Pointer)

H Hold Register

LV Pointer für Basisadresse im Stack *(Local Variables)*

MAR Memory Adress Register

MBR Memory Buffer Register

MDR Memory Data Register

N N=1 bei ALU-Inhalt <0

OPC Temporäres Register (=Arbeitsregister) *Old Program Counter*

PC Program Counter

SP Pointer für höchste Adresse im Stack *(Stack Pointer)*

TOS Top of Stack (enthält den Wert der Zelle, auf die SP zeigt; Sinn: schnellerer Zugriff)

Z Z=1 bei ALU-Inhalt =0

MIR Micro Instruction Register

MPC Micro Program Counter

Bildquelle: Tanenbaum, Structured Computer Organization

(Modern) processor design

Instr. Set
Micro Arch.
Cache
Mem

Abstract architecture description langauge (e.g. LISA)

Update

Processor Desinger Tool (e.g. Synopsys)

Generate

Application (C, C++)

Compiler

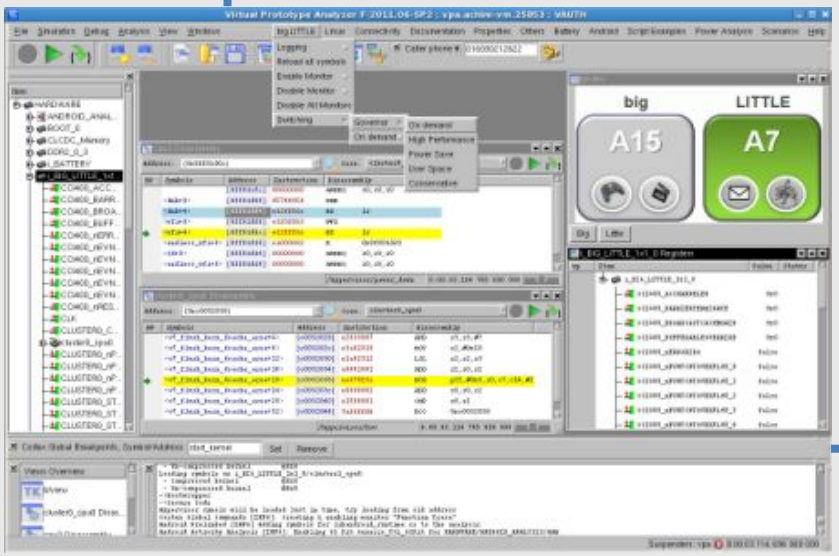
Assembler

Linker

Simulator

RTL, System C

To virtual plattform



Debugging, Profiling, Performance analysy

(Modern) processor design

SystemC model of new processor

Library with other processors and peripherals

Platform development

Fast I/O Interfaces

DSP

Network on Chip

Shared Memory

HW accelerator (reconf. HW)

RISC

$

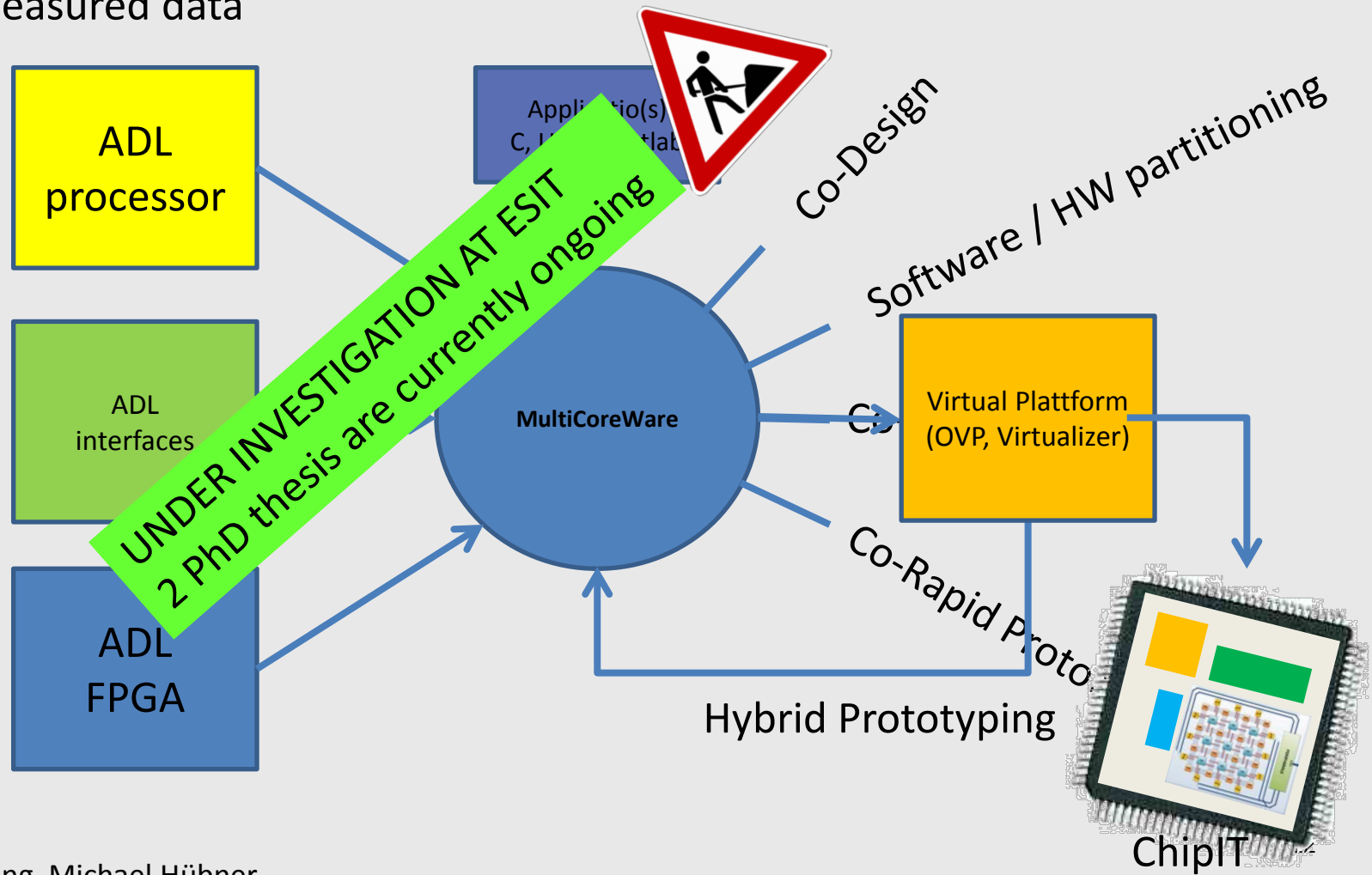
- Fast simultation through virtual platform
- Efficient HW / SW Codesign by fast update with new HW architectures
- Usage of real physical interfaces
  → real measured data can be processed to test the hardware

Co-simulation

ASIC /FPGA Emulation

Conclusion and outlook

- Co-Description of processors as well as the other architecture from high level of abstraction (model based)
- Fast simulation by using virtual prototyping platforms and FPGA based emulators
- Processing real measured data



- Instruction set
- Microarch. adaptivity
- …

- Businterface
- Pipeline-integration
- …

- LUT-width
- Routing channels
- …

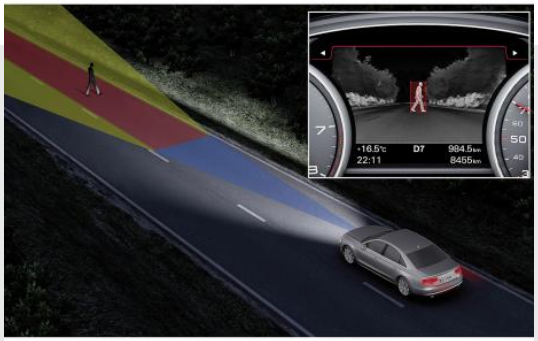
Thanks for your interest!

Contact:

Prof Dr.-Ing. habil. Michael Hübner
Chair for Embedded Systems in Information Technology (ESIT)
Ruhr-University of Bochum (RUB)
Building ID/1 Room 341
Tel.: +49 234 32 25975
Email: michael.huebner@rub.de

Collaboration welcome!

Embedded Systems: Examples from daily life: „Ubiquitous Computing“



→ Interaction with the environment and via network leads to the term Cyber Physical Systems