Laboratory: “Customized Embedded Processor Design for IoT”

Hussam Amrouch, Lars Bauer, Sajjad Hussain

(Lehrstuhl Prof. Dr. J. Henkel)

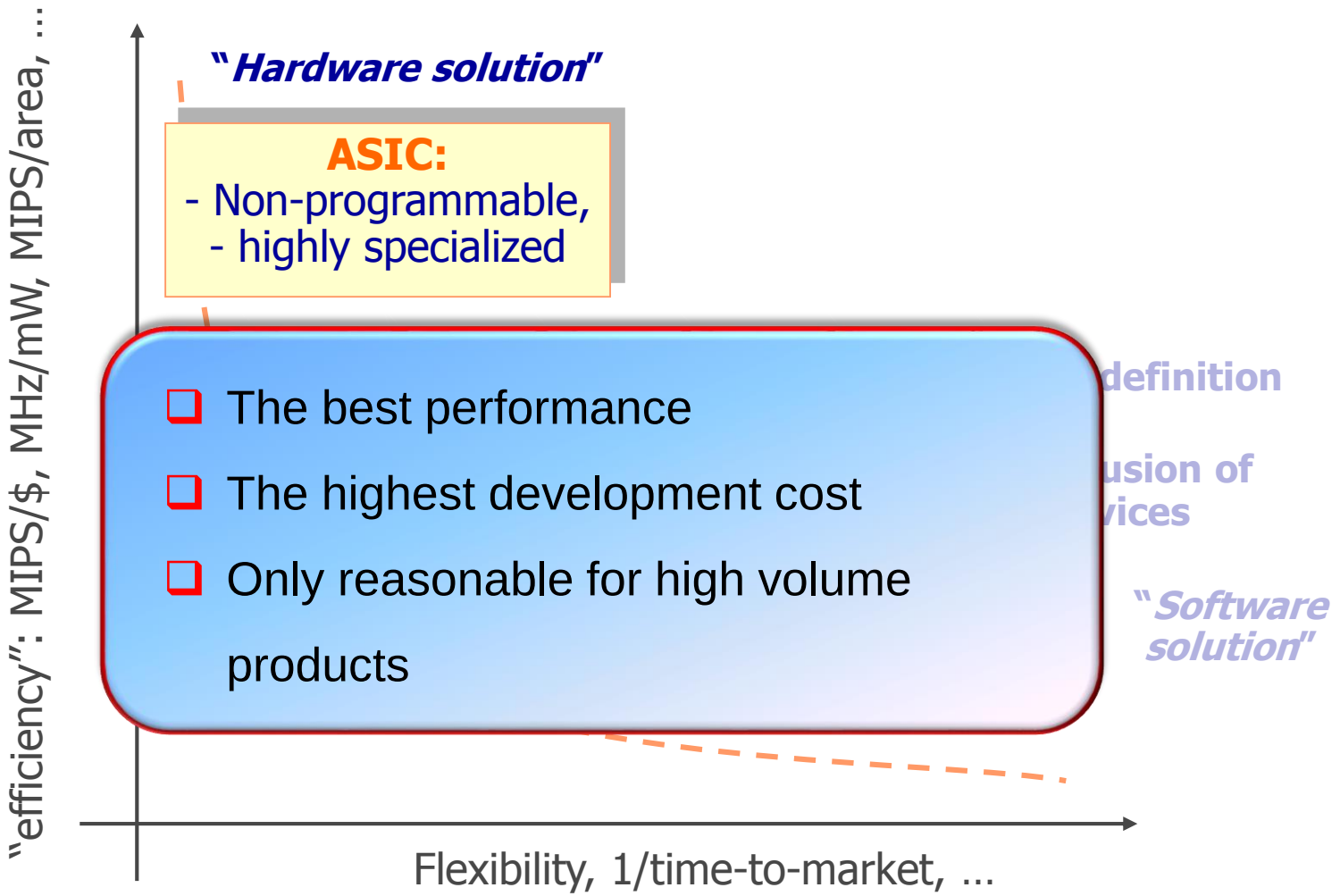
CES - Chair for Embedded Systems

KIT - Karlsruhe Institute of Technology, Germany

Internet of Things (IoT)

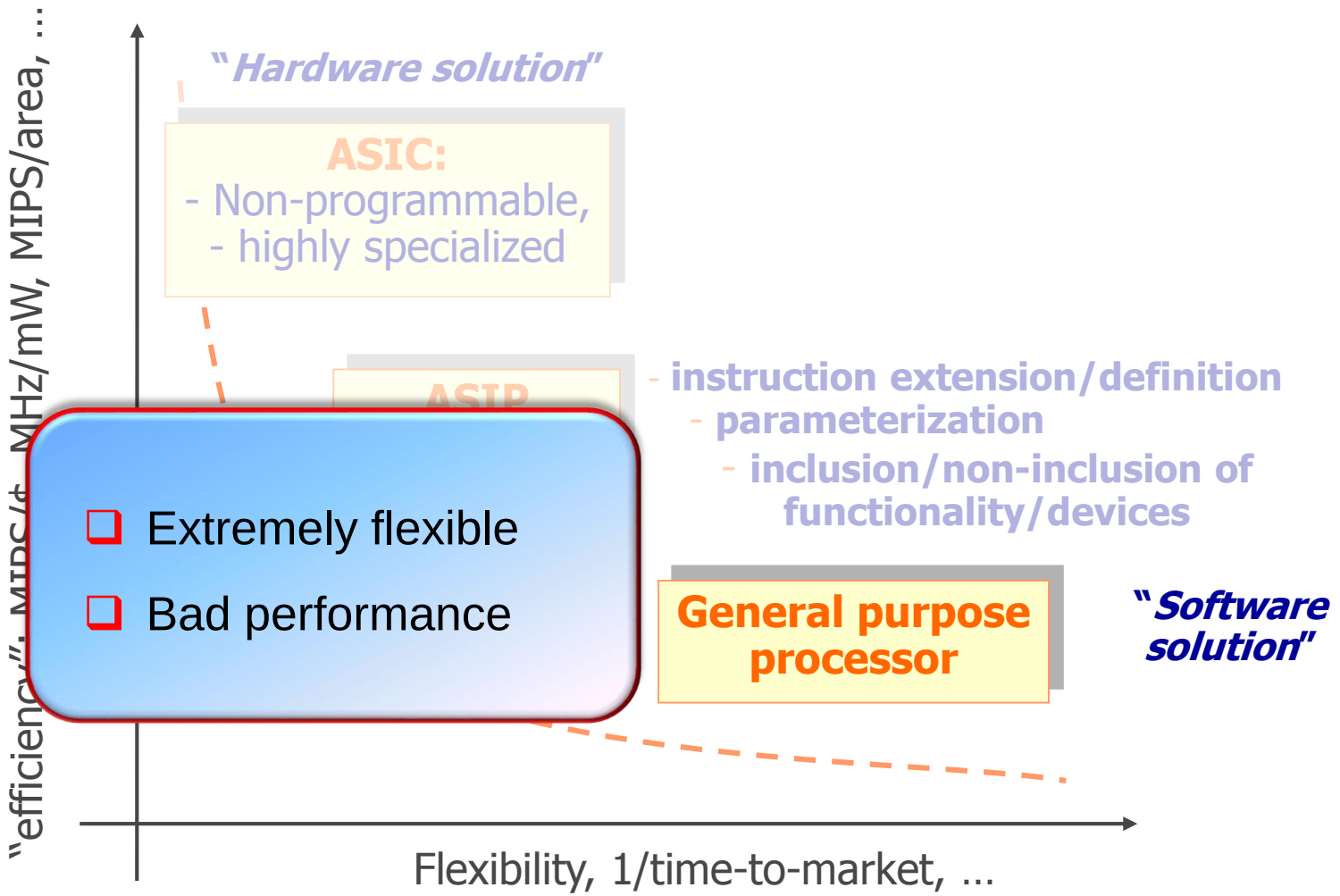
- Internet of Things (IoT) covers an ever-increasing range of applications.
- The design of embedded processors, especially for IoT, has experienced significant progress.
- This development has been characterized by the increasing demand for application-specific solutions for IoT in order to fulfil the diverse and contradictory requirements of low power consumption, high performance, low cost and most importantly an efficient time-to-market deployment of those processors.

ASIPs: efficiency vs. flexibility



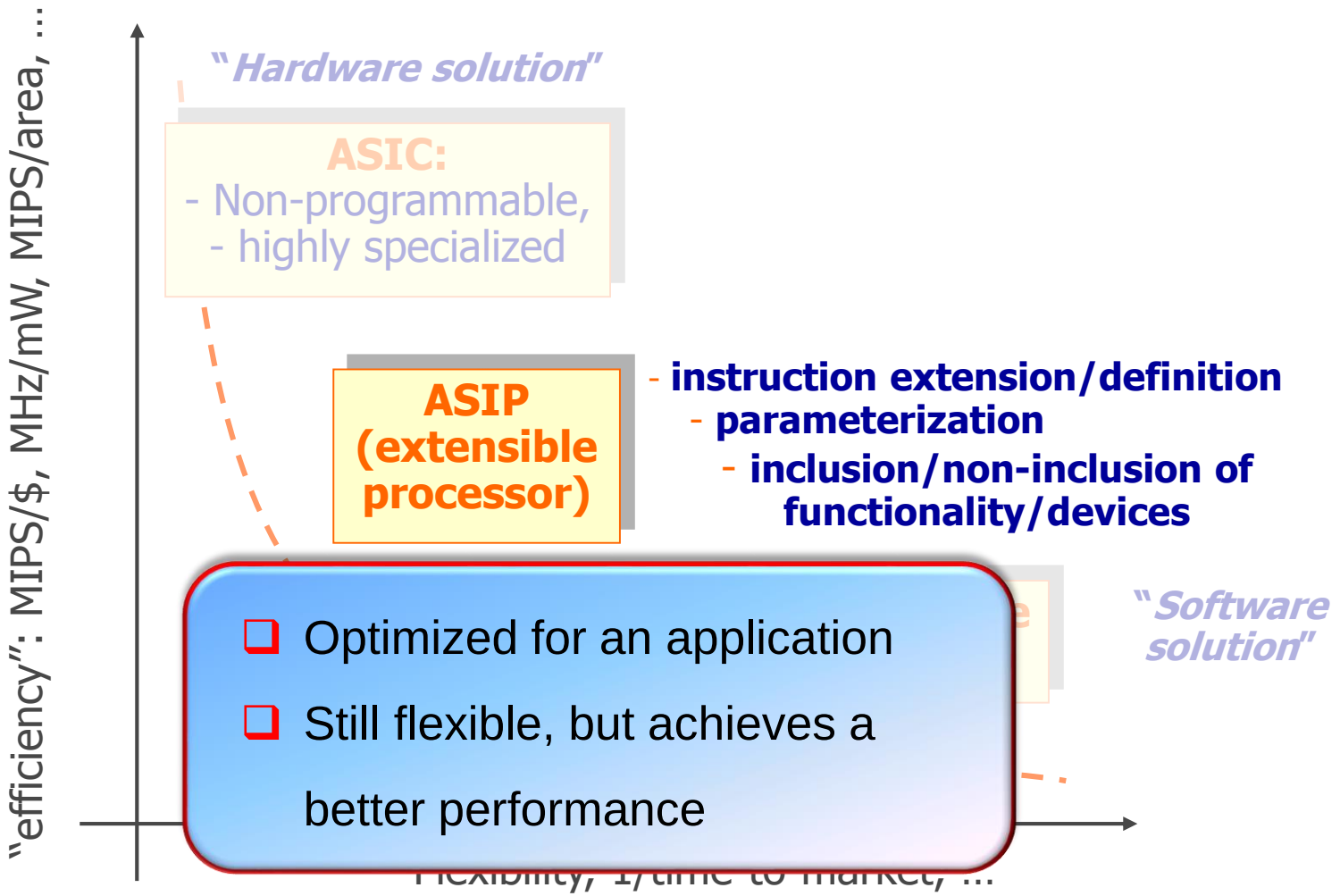
src: Henkel "Design and Architectures for Embedded Systems (ESII)"

ASIPs: efficiency vs. flexibility



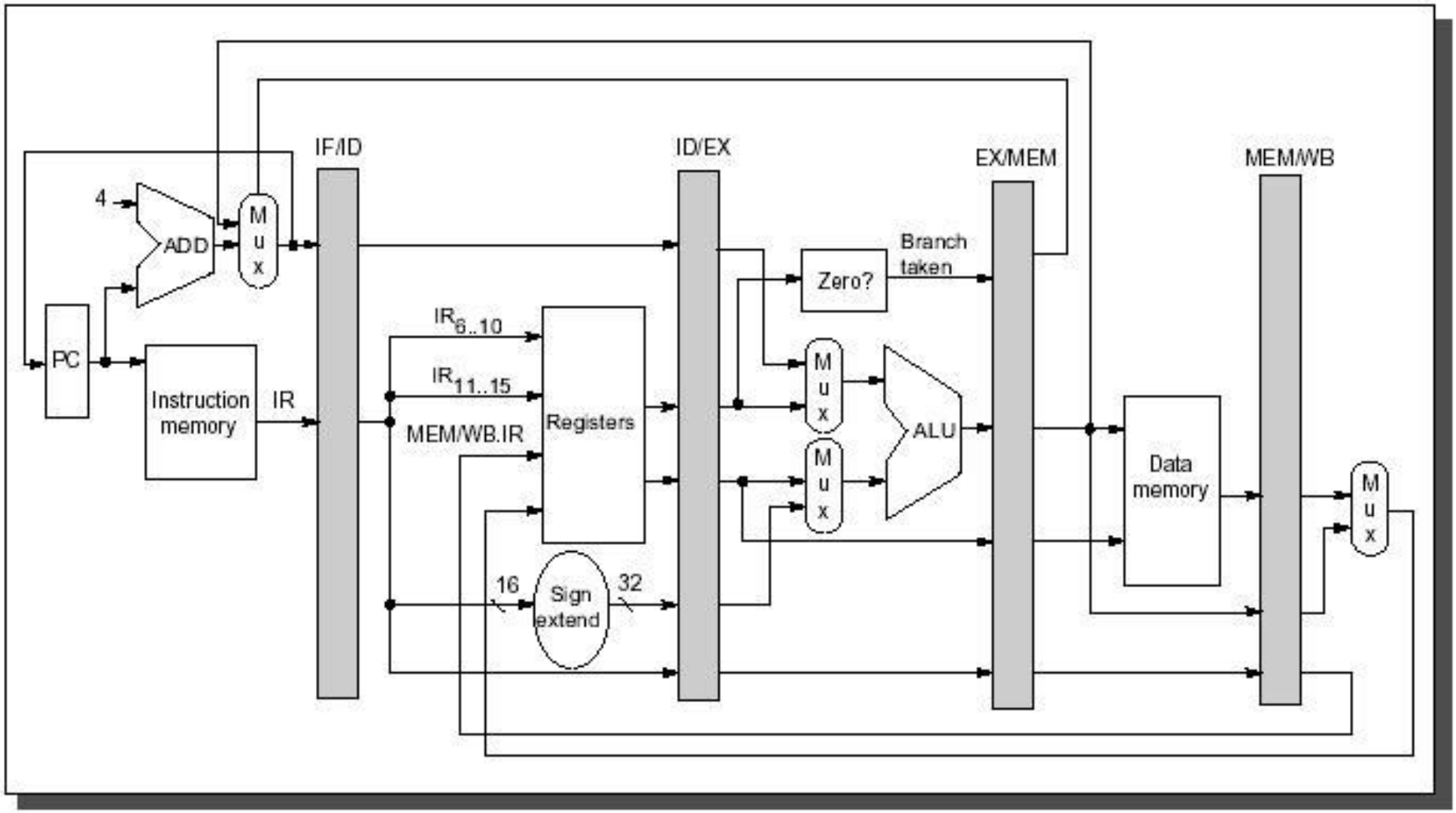
src: Henkel "Design and Architectures for Embedded Systems (ESII)"

ASIPs: efficiency vs. flexibility



src: Henkel "Design and Architectures for Embedded Systems (ESII)"

Pipeline Processor

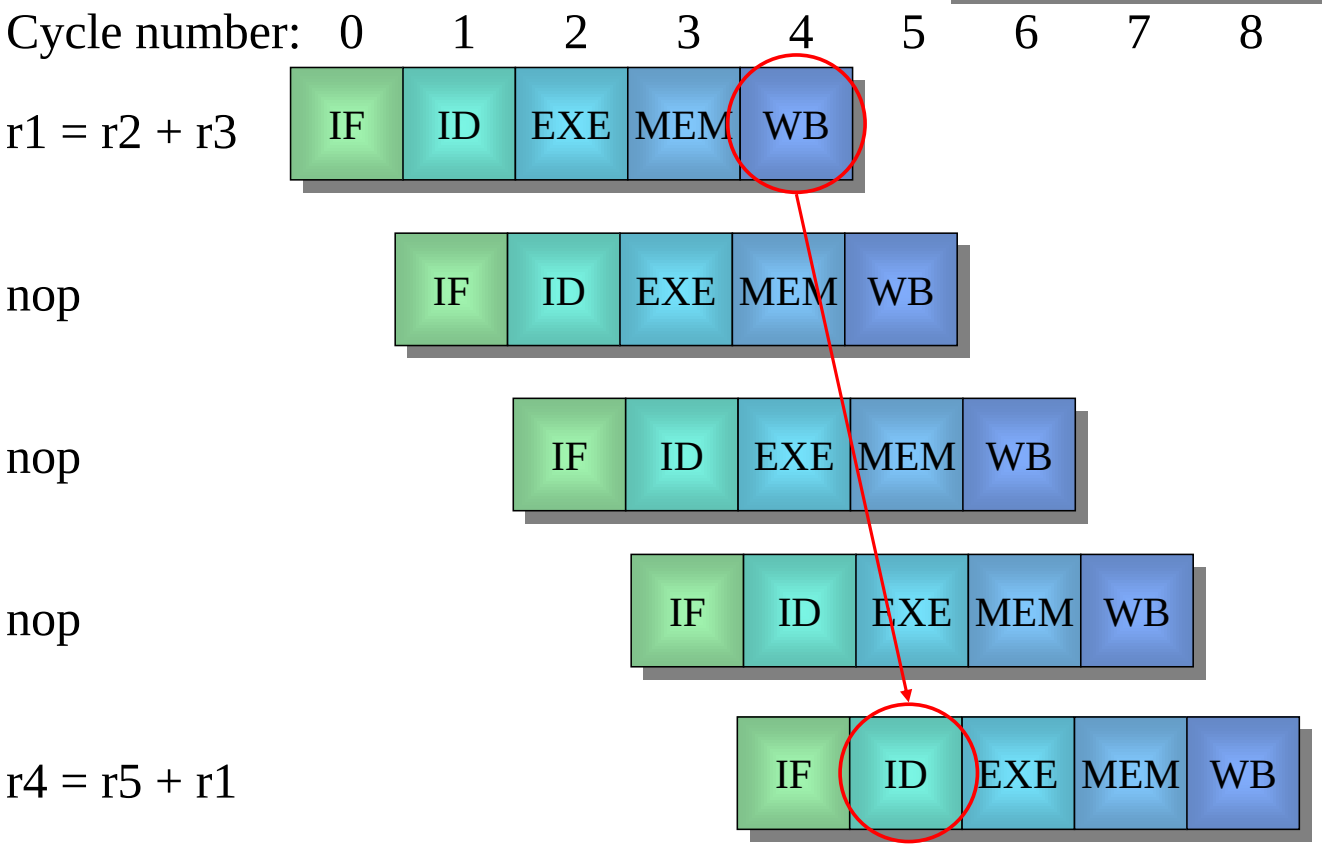


src: Muhammad Shaaban “The DLX Architecture”, lecture on ‘Computer Architecture’

Pipelining

Overlapping Execution of multiple Commands with a data dependency

IF	Instruction Fetch
ID	Instruction Decode
EXE	Execute
MEM	Memory Access
WB	Write Back



Pipeline Processor Used

- Brownie STD 32 is a RISC-type pipeline processor architecture. The Brownie architecture is designed for an easy and fast pipeline processor. It is a Load-/Store-architecture,

Parameters	Architecture
Basic architecture	RISC
Memory architecture	Harvard
Instruction length	32
Data length	32
Addressing	Byte address
The number of general-purpose registers	32
The number of pipeline stages	4
The number of delayed branch slot	0
Floating-point unit	N/A
Forwarding	full forwarding
Alignment	1-byte, 2-byte and 4-byte
Endian-ness	Big Endian
Interrupts	Reset, Internal, External

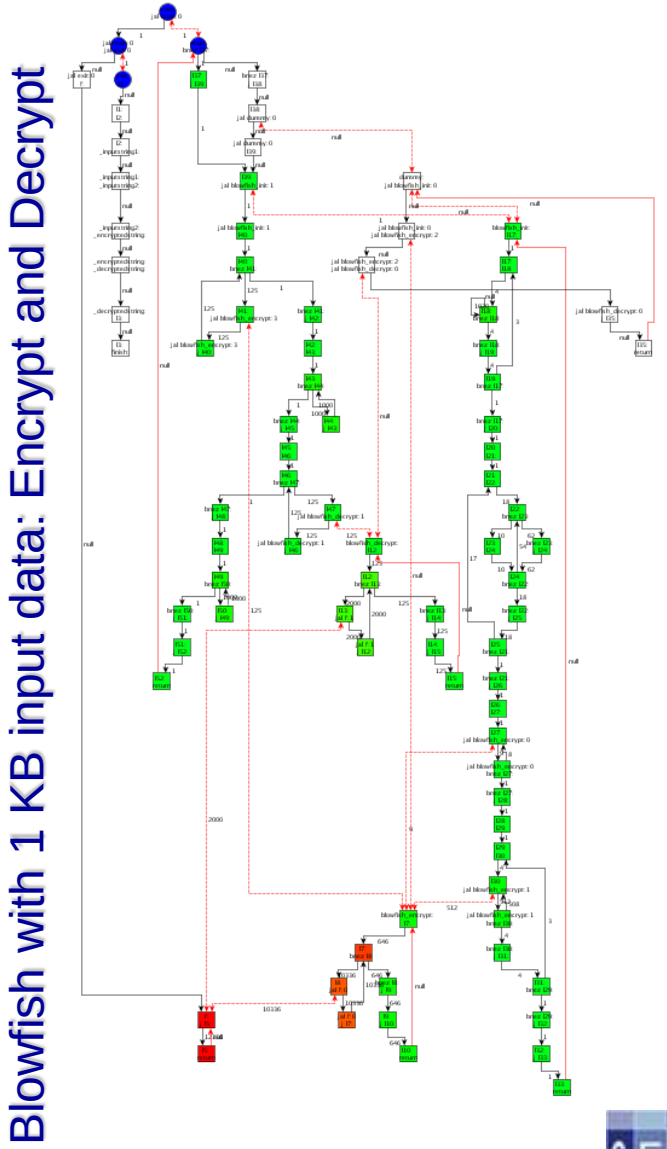
Goal of ASIP Lab

- Creating new CPUs with new instructions and implementing these new instructions in Hardware and Software with evaluation and testing

- The main goals are:
  - creating new ASIPs for special applications
  - benchmark those ASIPs to find out their benefits and drawbacks
  - and finally to interpret the benchmark results

Profiling on Control-Flow Graph

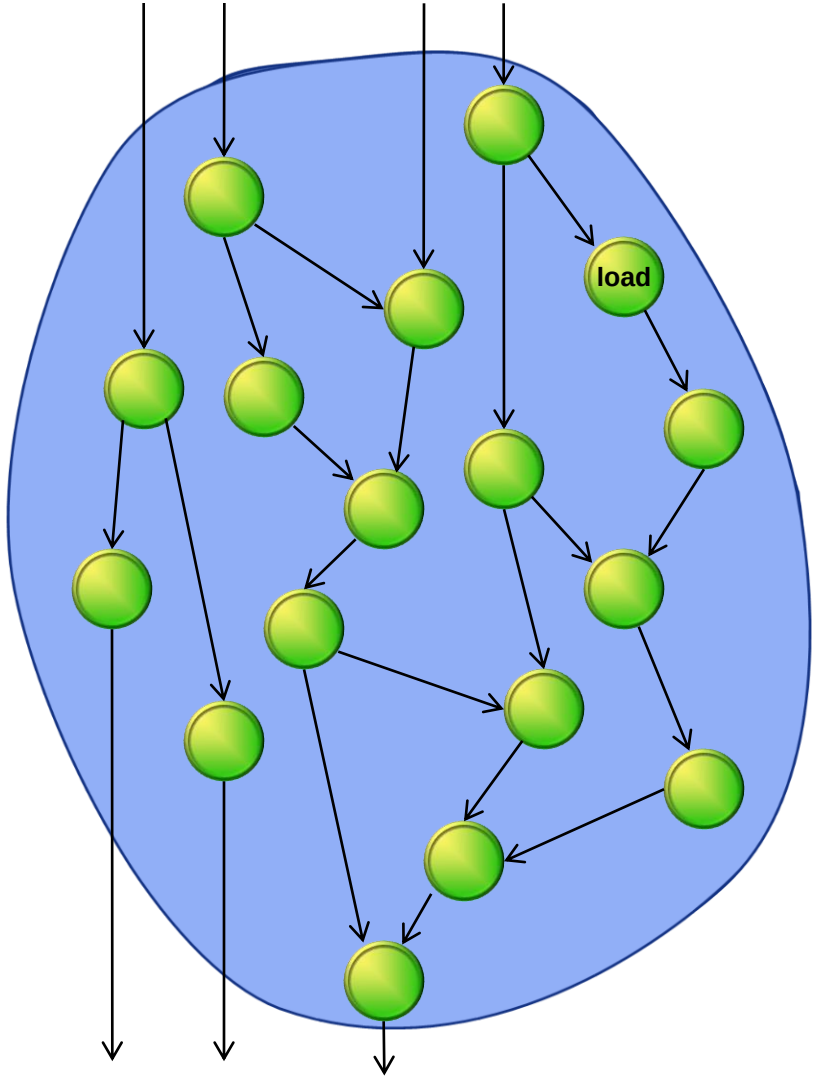
- Each node is a **Base Block**:
  - The instructions in a Base Block are always executed together
  - ➔ No jumps in a Base Block, except the end
  - ➔ No jump targets in a Base Block, except the beginning



Blowfish with 1 KB input data: Encrypt and Decrypt

Constraints for Designing Custom Instructions

- Number of Input-/Output-Values to Custom Instruction
  - Limited by register file, e.g. 4 read and 2 write ports
- ‘Forbidden Nodes’, e.g. Load/Store
- Convex Graph
- Limited Frequency, Area, Power, and/or Energy Budget
  - An ASIP that consumes more power may consume less energy, if the application terminates much earlier



Tasks in ASIP Lab

- Programming in assembly language
- Implementing new instructions with ASIP Meister (+Simulation, +Hardware)
- Evaluating the results from the new CPU
- designing new instructions for large applications with testing
- Creating different CPU versions

Fig.1 ASIPmeister Software

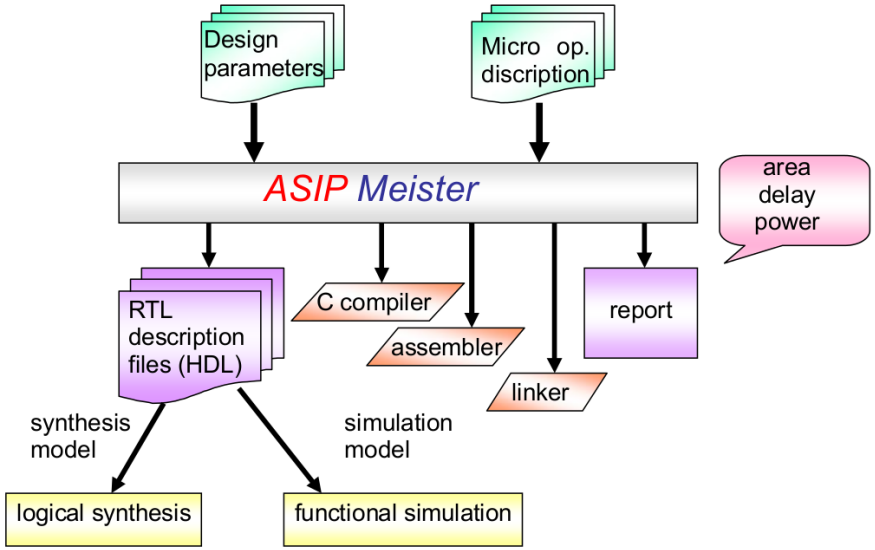


Fig. 2 ASIP Meister Input and Output

Lab Environments

- **dlxsim**: Simulator for DLX-Assembler
- **ASIPMeister**: To create new processor
- **Extended GCC compiler:** compiler out of an processor architecture description in ASIPMeister.
- **ModelSim**: simulator for VHDL-Code
- **Xilinx ISE**: Synthesis of VHDL-Code
- **XPower**: To estimate power consumption

Virtex-5 FPGA Evaluation Board

Important Directories

- /home/asip00/epp:
  - The required programs for the Lab are often used automatically by scripts
- /home/asip00/ASIPMeisterProjects/TEMPLATE_PROJECT
  - New version of Lab scripts
  - Project example with application example
- /home/asip00/Sessions/SessionX: The requirements for each session
  - For instance, you can find the slides and the Lab Script in ~/epp/Sessions/Session0
  - Before each session is started, the corresponding material will be copied to its folder.
- /home/asip00/Documents/: The relevant tutorials and manuals for the Lab.

Important Documents

Following is used as reference material. In each session, only the relevant chapters are asked to read. No need to go through all these.

- Laboratory Script
- ASIPMeister Users' Manual (AM_usersmanual_en.pdf)
- ASIPMeister Tutorial (AM_tutorial_en.pdf)
- Brownie STD 32 Reference Manual (BrownieSTD32_Spec_en.pdf)
- Brownie STD 32 Package ReadMe.txt (README_en.txt)
- Using the GNU Compiler Collection (Compiler.pdf)
- The GCC Assembler (GCC Assemble.pdf)
- The GNU linker (linker.pdf)

Important Note

- Generally, the sessions need to be executed on one of the following Lab PCs:

  i80labpc01, *i80labpc02*, … *i80labpc10*

- ASIPmeister is ALWAYS executed from i80pc57. This PC can be SSH-ed from any of above PCs i.e. i80labpcXX or from your laptop.
- FPGA JTAG driver is setup on *i80labpc10*, and FPGA board can be connected to this PC only.

To avoid compatibility issues between the Software tools and the OS

For remote access use
ssh username@i80labpc01.ira.uka.de

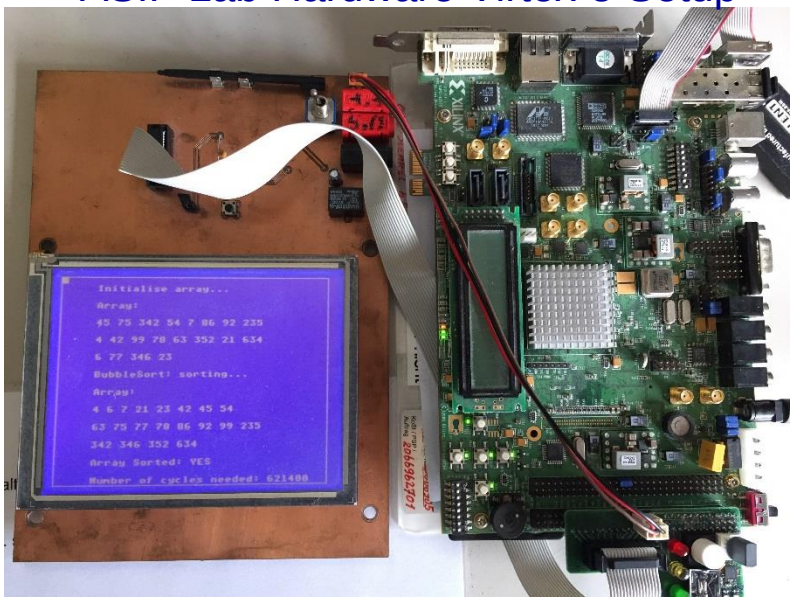
General Information

- Computer-Accounts (Ubuntu 12.04)
- ASIPmeister PC i.e. i80pc57 (CentOS 5)
- For the first session, read chapter 3 (dlxsim; without 3.2.2, 3.2.3 & 3.3.2)
- For the second session, read chapter 2 (working environment), 8 (using the compiler) and 5 (ModelSim)

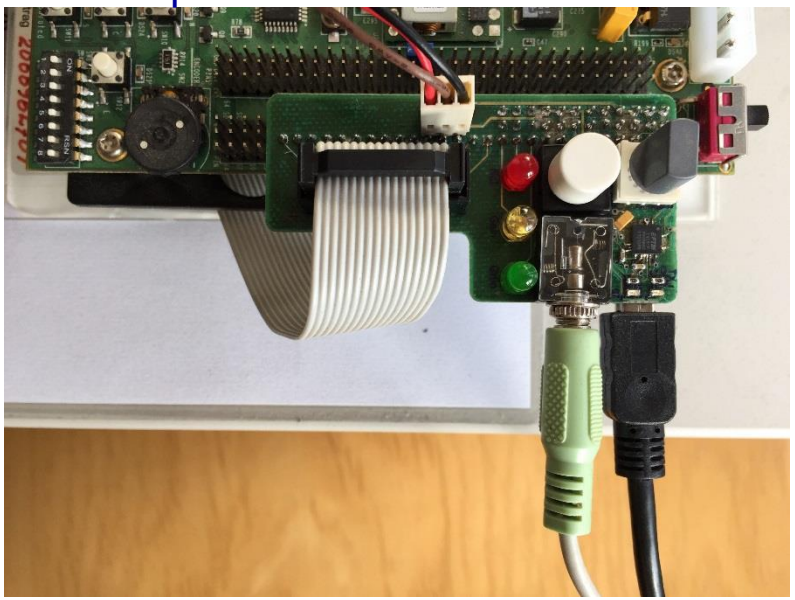
- **Now:** Accounts & Linux-Tutorial (chapter 2.2)

Bubble Sort Example

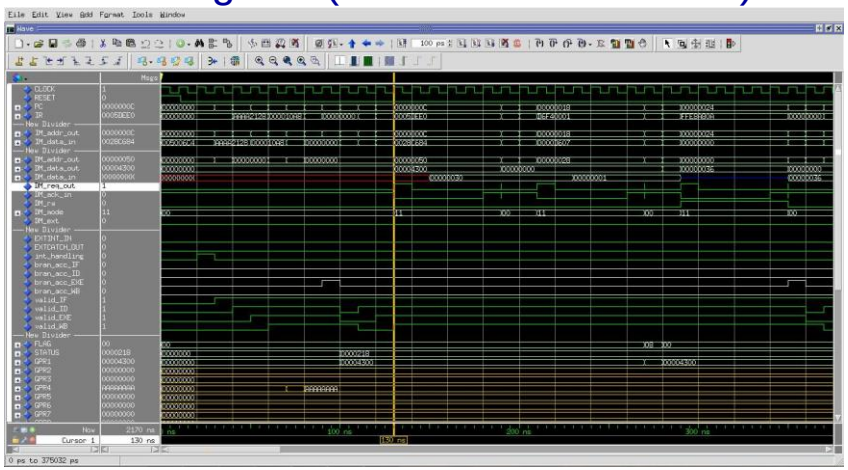
ASIP Lab Hardware Virtex-5 Setup



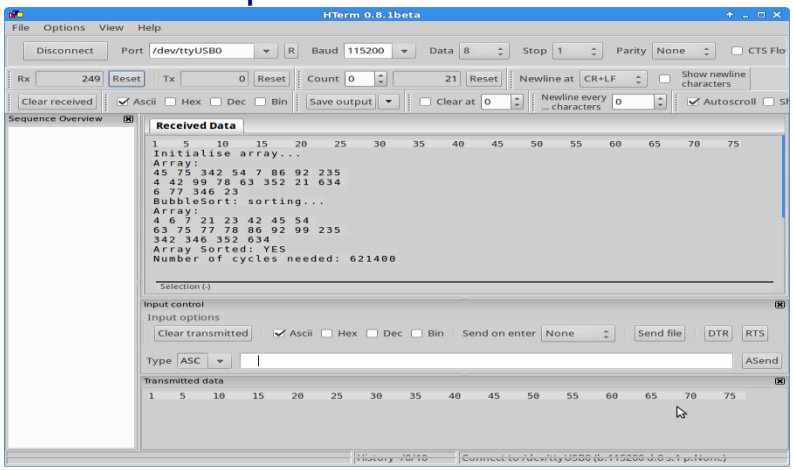
Peripheral Board Connections



CPU Signals (ModelSim Simulation)



Output to UART ***hterm***



Supervisors and the regular meeting

- Dr. Hussam Amrouch
- Dr. Lars Bauer
- Sajjad Hussain

- Regular Meeting:
  what about Monday 14:00 ?

Questions??