Course Organization and Introduction

Today's Agenda

Intended topics for today's session

- General Course Information
- Lecture Preview
- Course Goals and Intended Learning Outcomes
- Prerequisites
- Exam and Evaluation
- Recommended Readings
- FSoC WS22/23 in a Nutshell

FPGA-based SoC Design

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

General Course Information

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

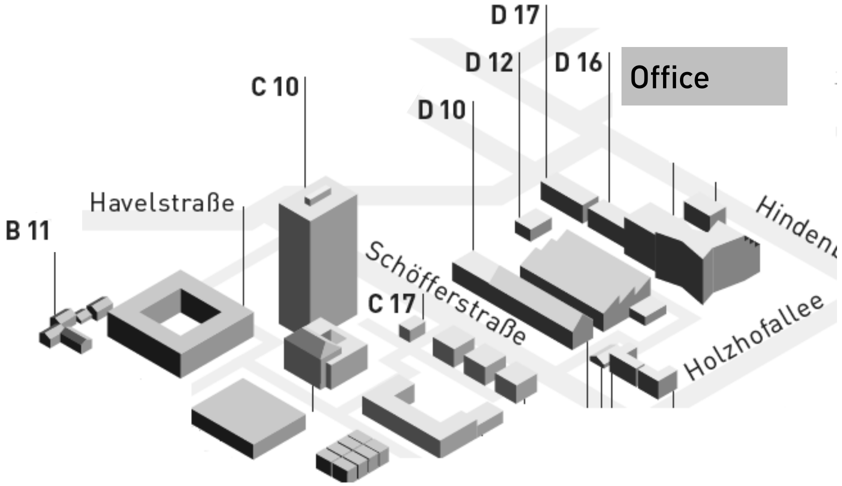
fbeit

General Course Information

h_da – fbeit - FPGA-based SoC Design

Contact Information

- Name: Prof. Dr. Christian Jakob
- Office: 311/D16, Birkenweg 8, 64295 Darmstadt
- Email: christian.jakob@h-da.de – Use your official h_da address to contact me
- Phone: +49.6151.16-37706 (Rarely reachable by phone)
- Office hour: Tuesday, 9.00am (this might change again)
  no registration necessary, first come
  first serve …



Moodle

- Moodle is the main landing page
- Course notes, additional materials, assignments, readings, example code …

  … will be posted on moodle
- Use the moodle forum to ask your questions. Something that is not clear to you is most probably not clear to others as well!

Lecture Preview

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

Lecture Preview

h_da – fbeit - FPGA-based SoC Design

Intended Roadmap WS22/23

- **Design and Verification of FPGA based System using SystemVerilog**

  Intel based FPGA Design, SystemVerilog based Design Entry, Build Automation using Makefiles, …

- **The Architecture of modern FPGAs – How to map HDL to HW …**

  Fundamental building blocks of modern FPGAs (Logic cells, memory, DSP cells, PLLs, …) and how to efficiently use them in HDL

- **Designing Data and Control-Path Architectures using SystemVerilog**

  Designing Data and Control Path Architectures, State Machines, …

- **Designing FPGA based Softcore Processor Systems**

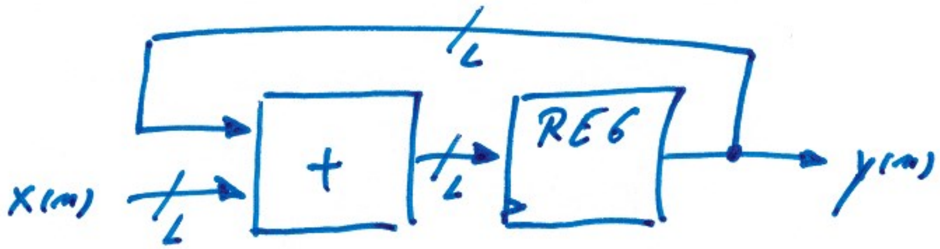
  Processor Design and Programming of FPGA based Softcore Architectures

- **Embedded Linux and SoC FPGAs**

  Architecture of SoC FPGAs, core concepts – from booting up to writing user space applications

- **Design and Interfacing of Custom Accelerators**

  Implementing custom Accelerators, ISA Customization and Extensions, Design of Customized HW/SW Interfaces



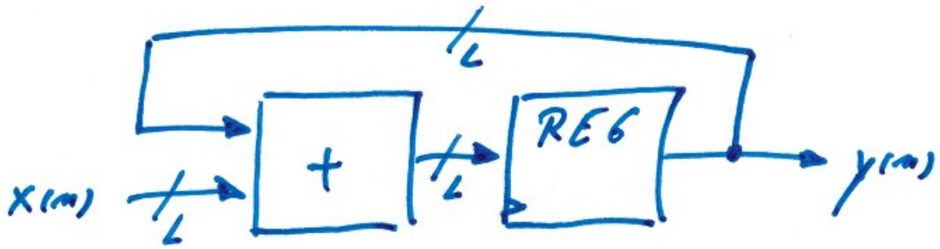
Lecture Preview

h_da – fbeit - FPGA-based SoC Design

Intended Roadmap WS22/23

- **High-Level-Synthesis**

  Algorithm and interface synthesis, design evaluation and optimization



Course Goals and Intended Learning Outcomes

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

Intended Learning Outcomes

Students should understand

- the structure of SoC FPGAs and their role in the design of modern electronic systems.
- the tools and methodologies for FPGA-based SoC design.
- the basic principles of hardware/software co-design and co-verification.

Course Goals and Learning Outcomes

Intended Learning Outcomes

Students should be able to apply the gained knowledge

- to partition simple software programs into dedicated hard- and software components.
- to analyse existing HW/SW architectures, to identify possible performance bottlenecks and to optimize them by finding the optimal HW/SW mapping.
- to optimize existing HW/SW architectures by transformations on hardware and software components.
- to evaluate the implementation results (e.g. timing, resource usage, power consumption) and correlate them with the corresponding high level design.

Prerequisites

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

Prerequisites

- I assume that students are familiar with the concepts of “hardware oriented” C programming and the fundamentals of digital logic design.
- Catch up on untreated material as soon as possible.

Laboratory Sessions

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

FSoC Lab WS22/23

- In order to participate in the module exam, it is required to successfully finish the lab part of the module.
- Successful fulfilment of prerequisites are measured by:
  - Lab attendance
  - Lab entry test
  - Lab progress
- 25% of the module grade is obtained by the laboratory.
- There are four lab sessions in total.
- Lab seat assignments will be published shortly.

FSoC Lab WS22/23

- The lab focuses on teaching practical skills related to FPGA based SoC design using C and SystemVerilog:
  - Design and implementation of custom hardware accelerators and ISA extensions.
  - HW/SW integration of custom accelerators into existing FPGA based SoC architectures followed by profiling and benchmarking of the respective solutions.

FSoC Lab WS22/23 – Hardware Accelerator Design

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

FSoC Lab WS22/23 – Hardware Accelerator Design

- General-purpose CPUs implement a fixed set of instructions, able to execute nearly any kind of computation.
- This great flexibility offered by a CPU based computing platform also represents one of its major drawbacks.
- The sequential execution of algorithms, often characterized as ‘temporal computing’ allows the implementation of almost all existing algorithms, however, not always with their best potential performance.

FSoC Lab WS22/23 – Hardware Accelerator Design

- Wikipedia: Hardware acceleration is the use of computer hardware designed to perform specific functions more efficiently when compared to software running on a general-purpose central processing unit (CPU).
- The slowdown and the nearing end of transistor scaling play a vital role in the rise of domain-specific architectures.
- Domain-specific architectures are based on "accelerators": Specialized hardware structures designed to efficiently execute dominating computation patterns and improve performance under a given transistor budget.

FSoC Lab WS22/23 – Hardware Accelerator Design

- Application areas
  - Networking - Security, monitoring, function virtualization
  - Streaming analytics
  - Financial technology (Fintech) - Banking, investment, risk management
  - Genomic sequencing
  - Video transcoding, media processing
  - Machine learning
  - Anything with significant parallelism
  - …

FSoC Lab WS22/23 – Hardware Accelerator Design

- This trend is not limited to large-scale computing platforms. Specialized and application specific computing units have become common place in modern embedded computer architectures:
  - The ST Microelectronics **STM32G4** series combines a 32-bit Arm® Cortex®-M4 core (with FPU and DSP instructions) running at 170 MHz combined with **mathematical accelerators**: one for trigonometric calculations and the other for filtering functions.
  - The ST Microelectronics **STM32F479XX** series integrates various cryptographic hardware accelerators (HW accelerator for AES 128, 192, 256, Triple DES, HASH (MD5, SHA-1, SHA-2) and HMAC)
  - Texas Instruments **AWR1443** is an automotive single-chip FMCW radar sensor capable of operation in the 76- to 81-GHz band. The device comes with an integrated ARM R4F processor and a **hardware accelerator** for radar data processing …
  - The Renesas **RX72T** series of 32-bit MCUs are optimized for motor control. They have dedicated hardware accelerators to perform the complex, high-speed computations required for motor control in robots and other industrial applications.
  - …
- There will be MCUs with integrated AI accelerators in the near future to enable intelligent IOT/AI at the edge devices …

FSoC Lab WS22/23 – Cryptographic Hardware Accelerator Design

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

FSoC Lab WS22/23 – Cryptographic Hardware Accelerator Design

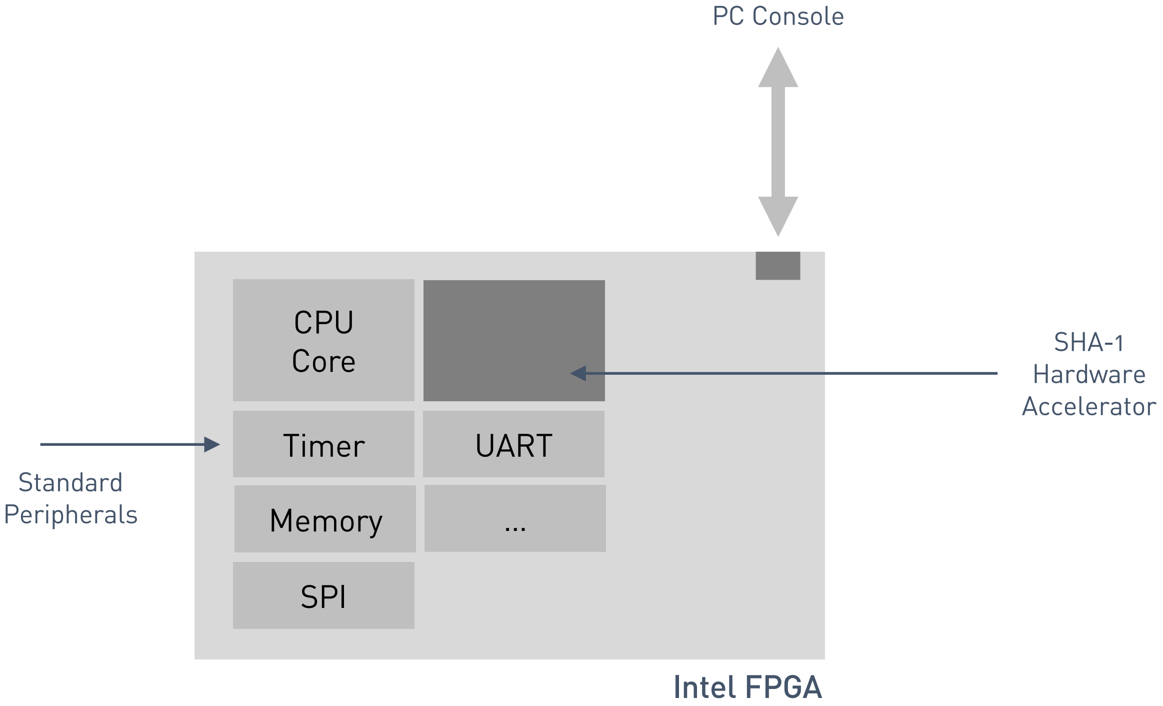
- Our intention is to design an FPGA based Embedded System similar to the ones described before: The task is to analyze, design and implement a cryptographic hash function (SHA-1) in hardware (SystemVerilog) and software (C/C++ bare-metal on the Intel Nios II softcore processor or on the ARM Cortex A9 running Linux).
- The hardware unit finally acts as dedicated accelerator placed in the peripheral-set of the Nios II CPU to accelerate the computation of the SHA-1 algorithm.
- Hash functions are based on simple mathematical recipes. They map data of arbitrary size to a fixed length output (the so called hash).

Data of arbitrary length ...

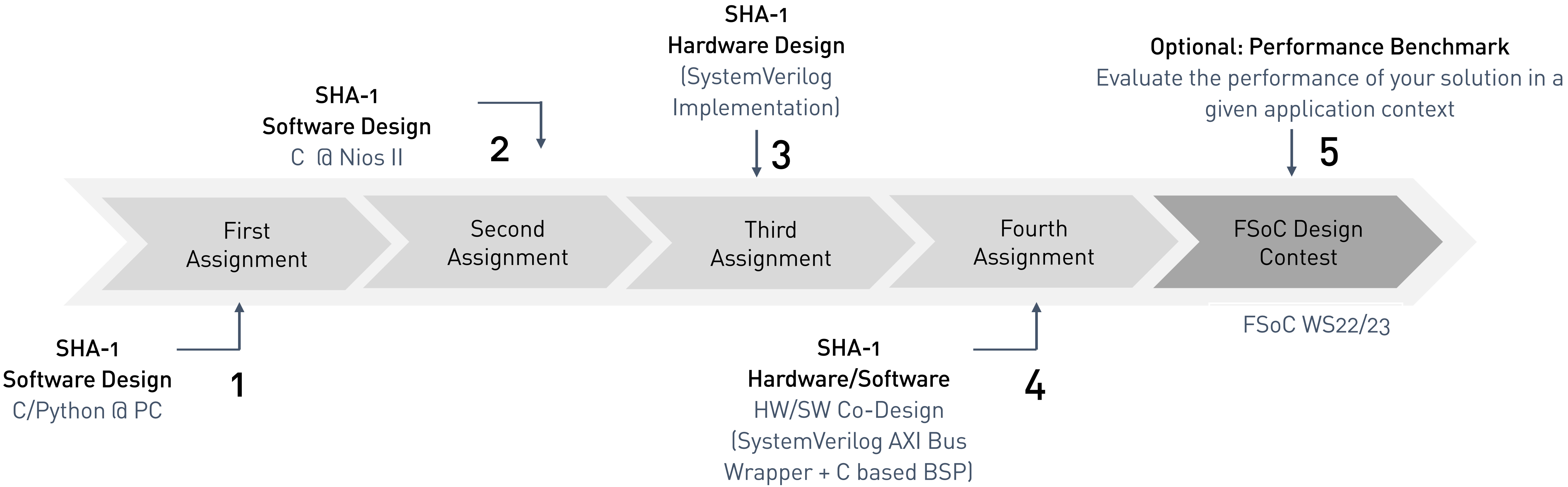
Hash Function

Fixed-length output

FSoC Lab WS22/23 – Cryptographic Hardware Accelerator Design



FSoC Lab WS22/23 – Cryptographic Hardware Accelerator Design



SHA-1 Software Design
C/Python @ PC

SHA-1 Software Design
C @ Nios II

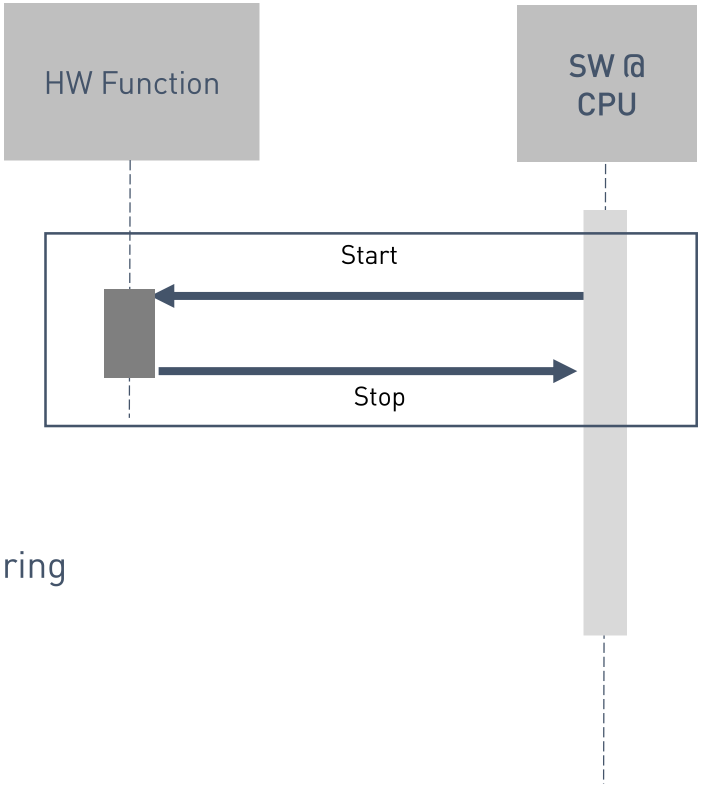
SHA-1 Hardware Design
(SystemVerilog Implementation)

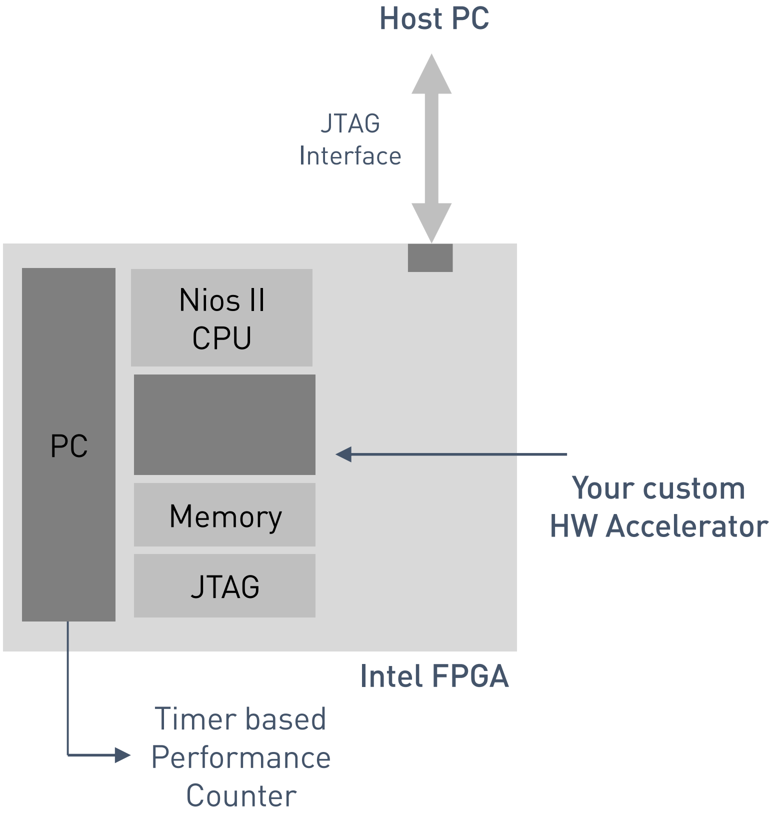
SHA-1 Hardware/Software
HW/SW Co-Design
(SystemVerilog AXI Bus Wrapper + C based BSP)

Optional: Performance Benchmark
Evaluate the performance of your solution in a given application context

The FSoC Design Challenge WS22/23

- Use and benchmark your design in a dedicated application context.





- Inspired by Patrick R. Schaumont annual design challenge running at the Professor Bradley Department of Electrical and Computer Engineering Virginia Tech.

Exam and Evaluation

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

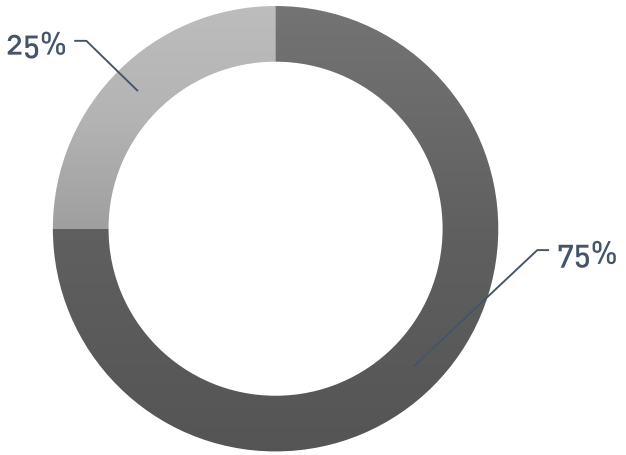
fbeit

FSoC Examination WS22/23

- WS22/23 semester grades will be based on the following weights:

Final Grade

- Exam (75%)
- Lab part (25%)



- There is the possibility to get extra credit for the completion of homework assignments.

Recommended Readings

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

Recommended Readings

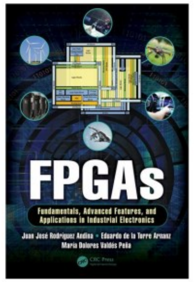
h_da – fbeit - FPGA-based SoC Design

Course related Literature

- Kilts, Steven, “Advanced FPGA Design – Architecture, Implementation, and Optimization”, Wiley-IEEE Press, 2007

  - Addressing advanced issues of FPGA (Field-Programmable Gate Array) design and implementation, Advanced FPGA Design: Architecture, Implementation, and Optimization

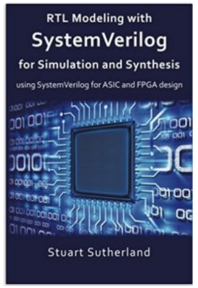
- Andina, J., Arnanz, E., Valdes, M., “FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics”, CRC Press, 2017


FPGAs
Fundamentals, Advanced Features, and Applications in Industrial Electronics

  - Explains how and why field-programmable gate arrays (FPGAs) may be used for high-end applications in a variety of domains.
  - Analyzes fundamental architectures of FPGA devices, providing case studies and simple design examples
  - Presents a state-of-the-art vision of FPGA technologies as well as a mid-term forecast for the future

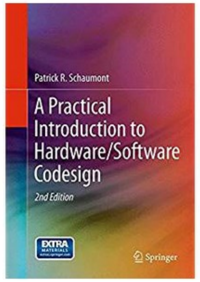
Course related Literature

- Sutherland, S., “RTL Modeling with SystemVerilog for Simulation and Synthesis: Using SystemVerilog for ASIC and FPGA Design”, CreateSpace Independent Publishing Platform, 2017


RTL Modeling with
SystemVerilog
for Simulation and Synthesis
using SystemVerilog for ASIC and FPGA design
Stuart Sutherland

  - This book is both a tutorial and a reference for engineers who use the SystemVerilog Hardware Description Language (HDL) to design ASICs and FPGAs.
  - The book shows how to write SystemVerilog models at the Register Transfer Level (RTL) that simulate and synthesize correctly, with a focus on proper coding styles and best practices.

- Schaumont, P. R., “A Practical Introduction to Hardware/Software Codesign”, Springer, 2nd Edition, 2014



  - This textbook serves as an introduction to the subject of embedded systems design, with emphasis on integration of custom hardware components with software.
  - The key problem addressed in the book is the following: how can an embedded systems designer strike a balance between flexibility and efficiency?

Recommended Readings

Online Resources

Intel FPGA Technical Training

- https://community.intel.com/

All about tooling, FPGAs and HDLs

Intel FPGA University Program

- https://software.intel.com/content/www/us/en/develop/topics/fpga-academic.html

Tutorials for Digital Logic Design, Computer Organization and Embedded Systems

RocketBoards

- https://rocketboards.org/

Community home for open source collaboration aimed at Linux development for SoC FPGAs

FSoC WS22/23 in a Nutshell ...

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

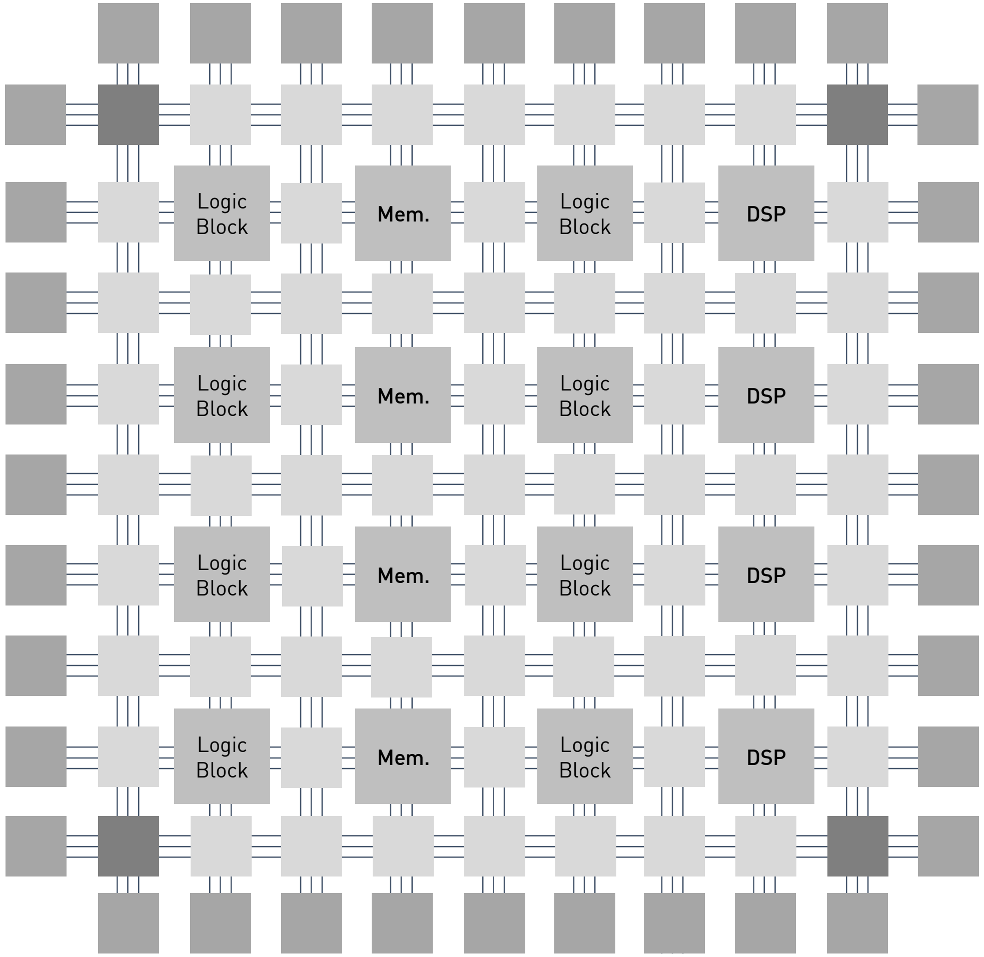
FSoC WS22/23 in a Nutshell …

- Everything starts with …

Intel FPGA

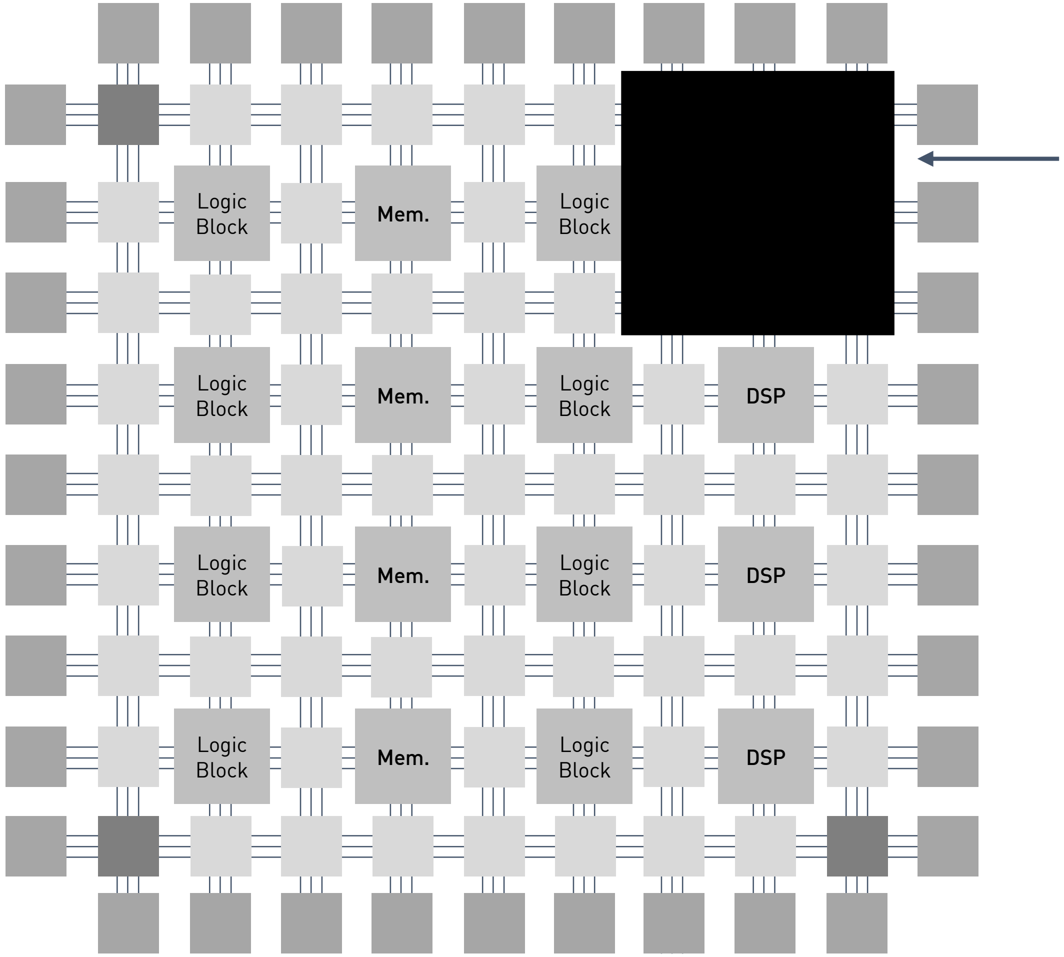
FSoC WS22/23 in a Nutshell ...

- Simplified FPGA (logic-centric) Structure

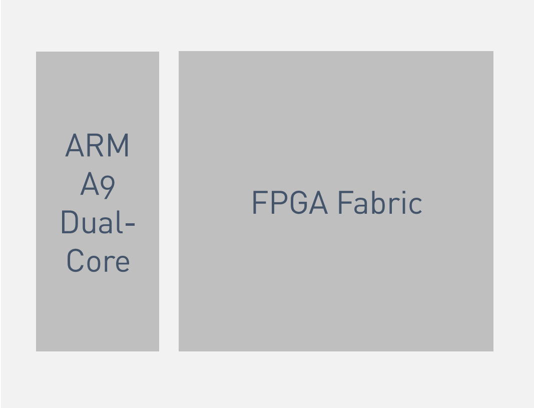


FSoC WS22/23 in a Nutshell ...

- Simplified FPGA (SoC-centric) Structure



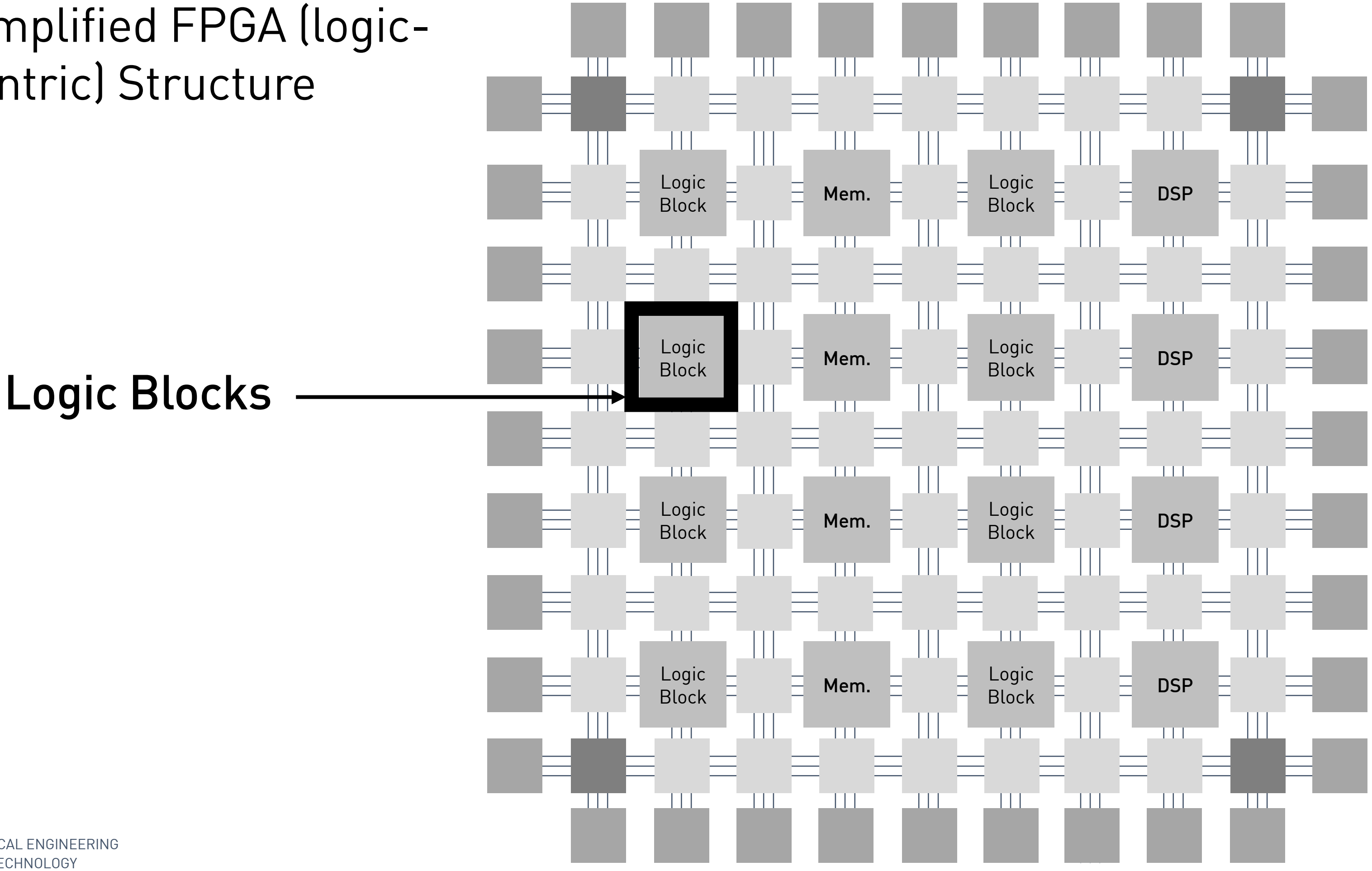
Hard Processor Systems



Intel SoC FPGA

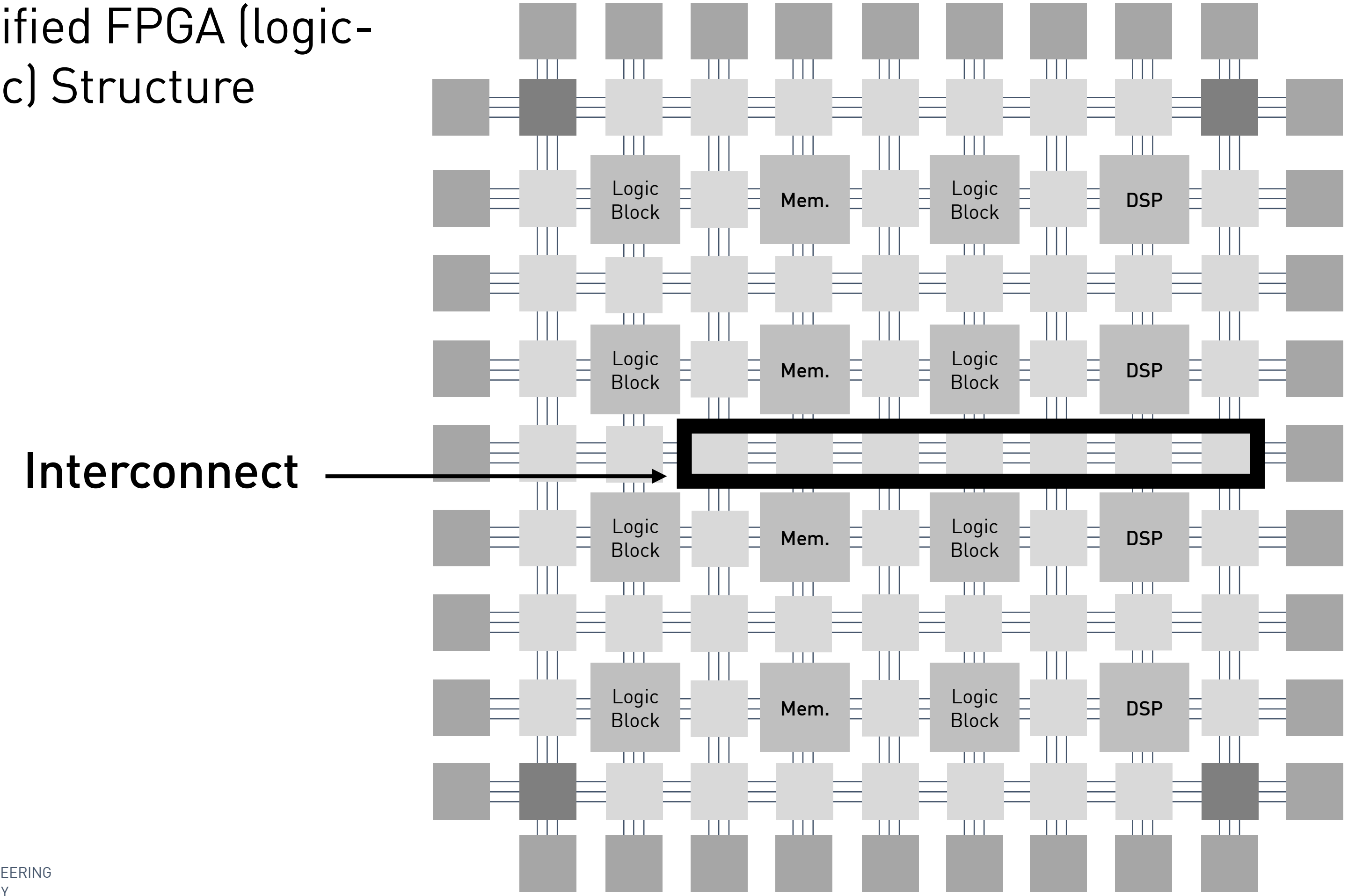
FSoC WS22/23 in a Nutshell ...

- Simplified FPGA (logic-centric) Structure



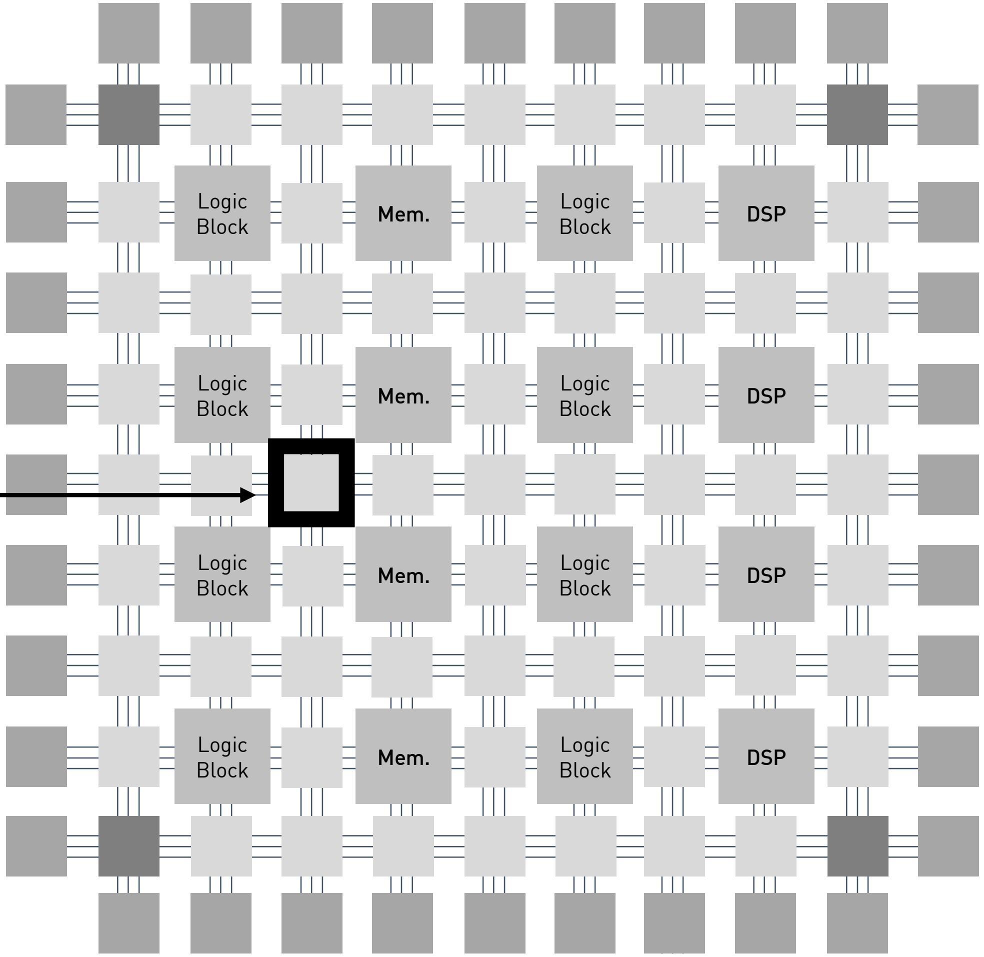
FSoC WS22/23 in a Nutshell ...

- Simplified FPGA (logic-centric) Structure



FSoC WS22/23 in a Nutshell ...

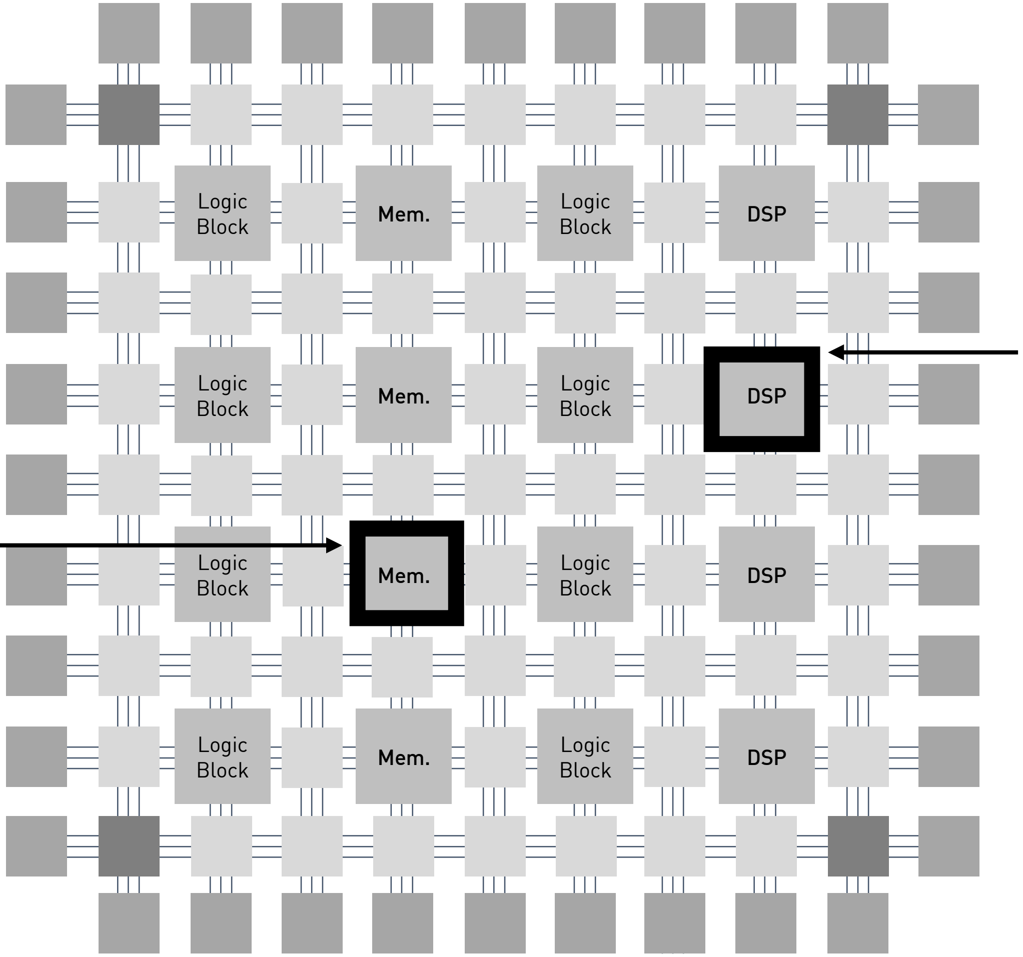
- Simplified FPGA (logic-centric) Structure



Switch Matrix Blocks

FSoC WS22/23 in a Nutshell ...

- Simplified FPGA (logic-centric) Structure

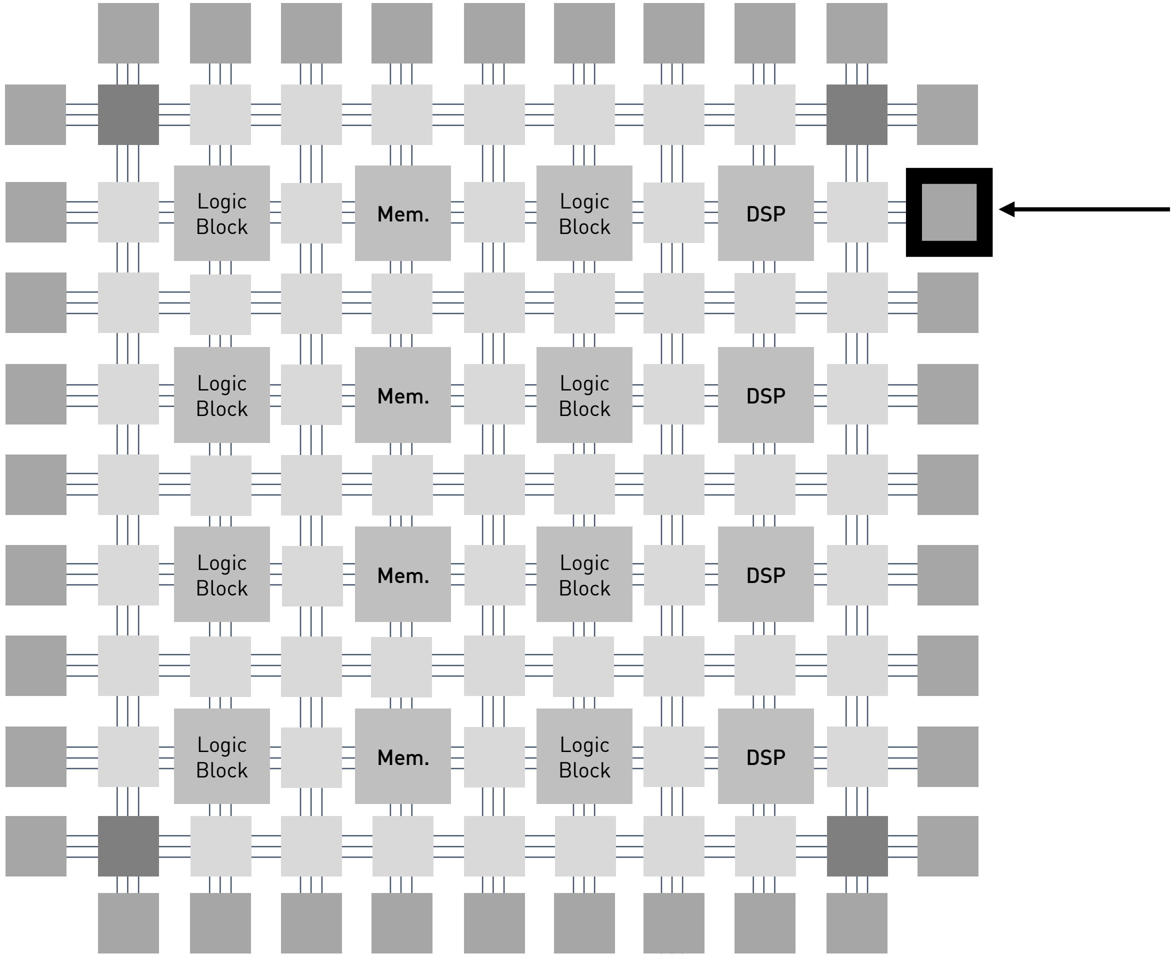


Dedicated Functionality (Multiplier)

Dedicated Functionality (Memory)

FSoC WS22/23 in a Nutshell …

- Simplified FPGA (logic-centric) Structure

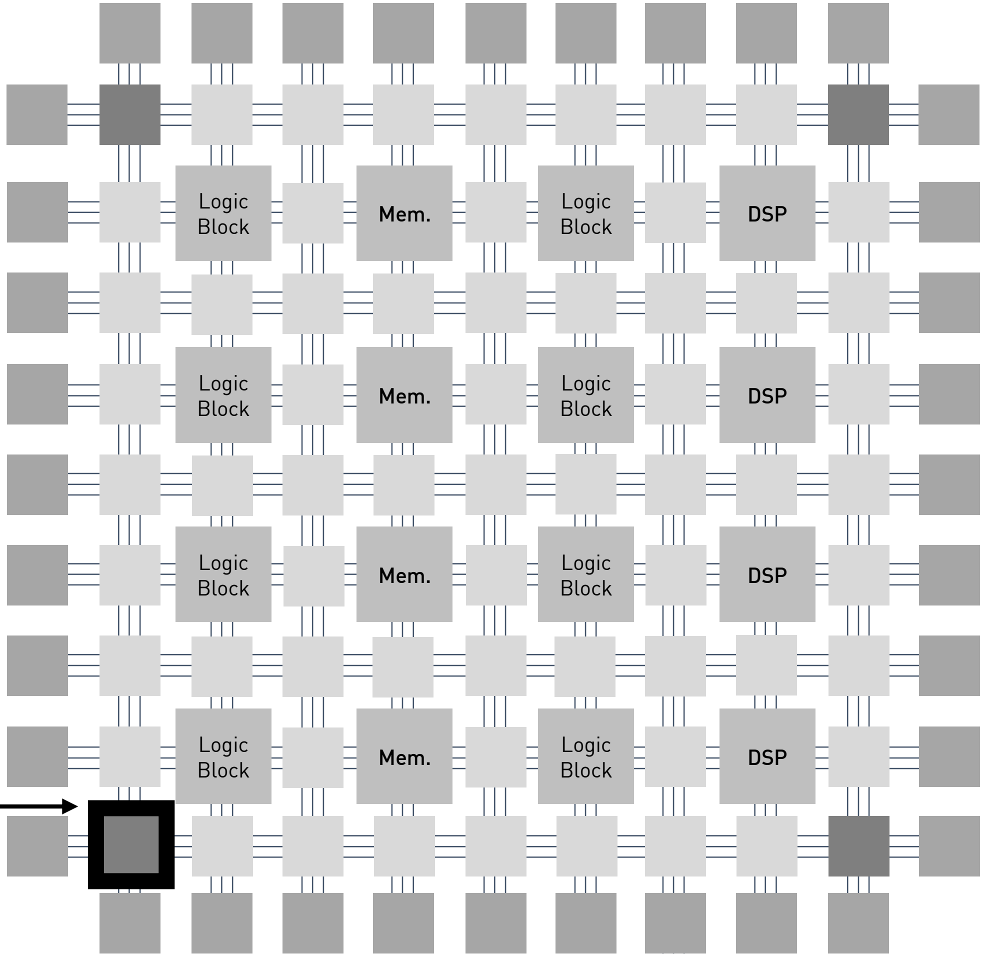


I/O Blocks

- Supporting various I/O standards

FSoC WS22/23 in a Nutshell ...

- Simplified FPGA (logic-centric) Structure



PLL

FSoC WS22/23 – Mapping HDL to FPGA logic cells

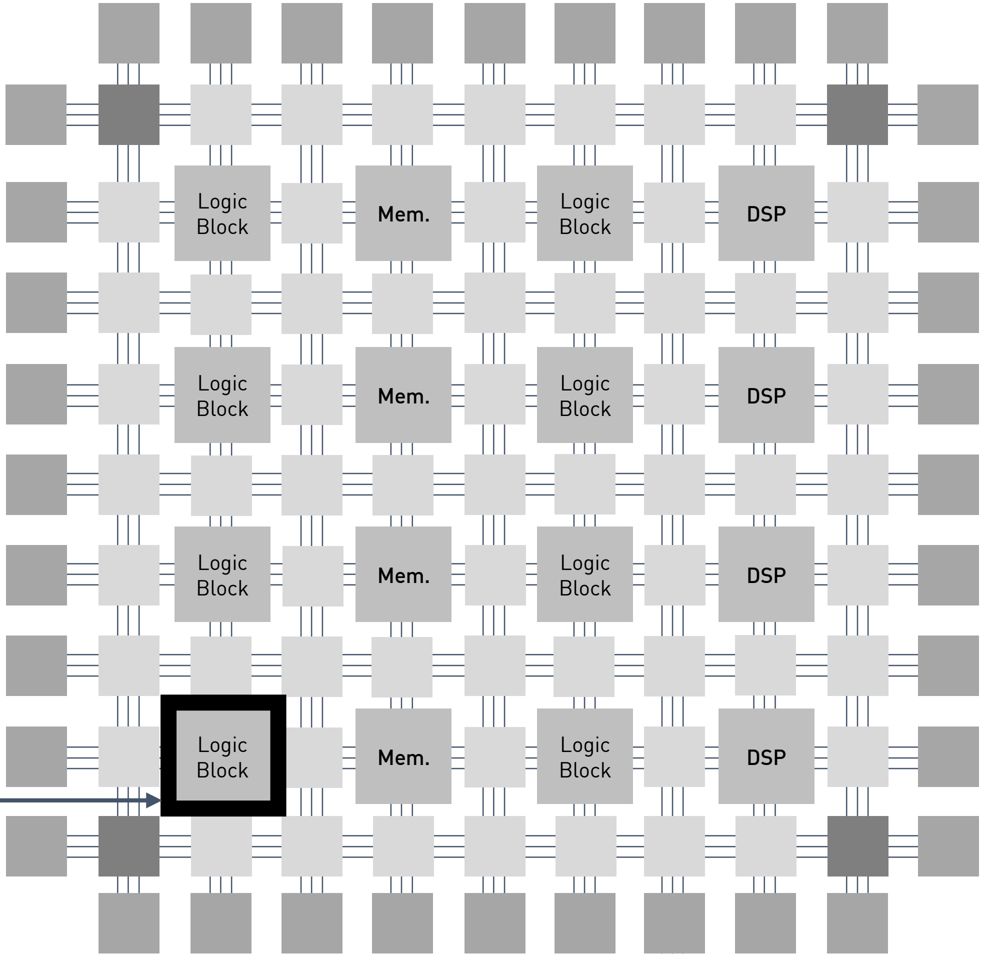
International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt
h_da
Faculty of Electrical Engineering and Information Technology
fbeit

Basic FPGA Logic Cell Structure

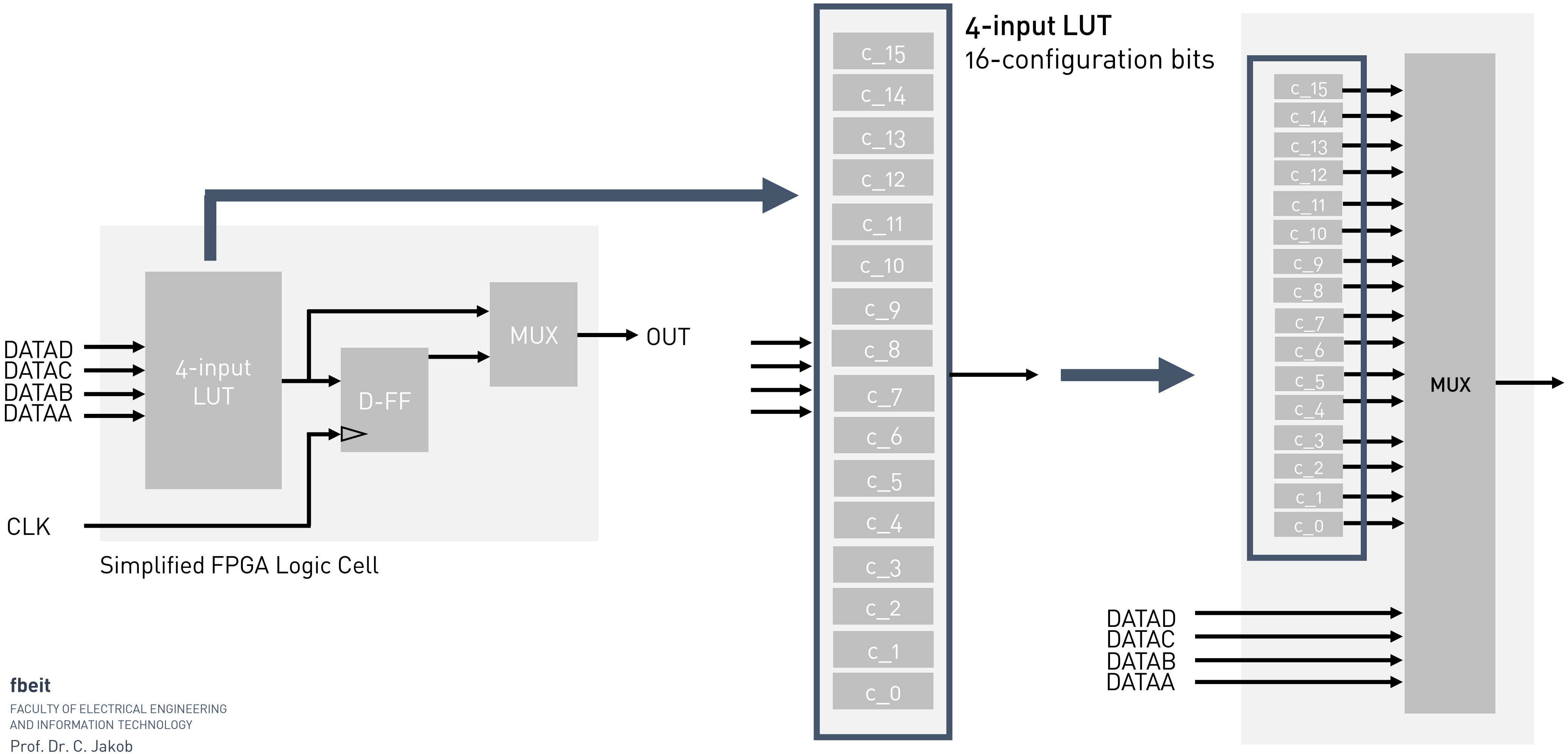
Notes

- Combinational and sequential logic is mapped to dedicated logic cells within the FPGA fabric



Logic Blocks

Basic FPGA Logic Cell Structure

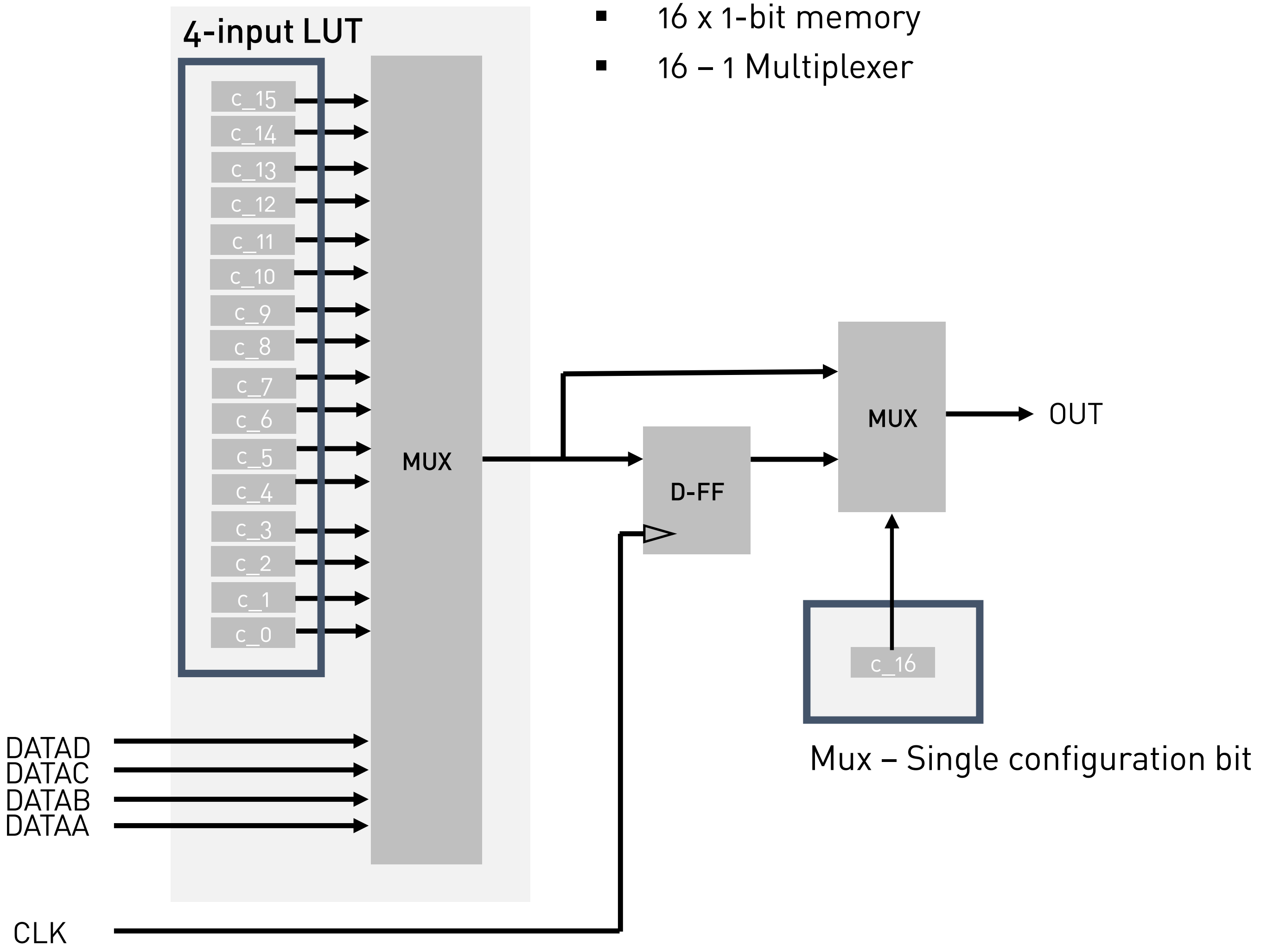


Simplified FPGA Logic Cell

4-input LUT
16-configuration bits

- 16 x 1-bit memory
- 16 – 1 Multiplexer

Basic FPGA Logic Cell Structure



- 16 x 1-bit memory
- 16 – 1 Multiplexer

Mux – Single configuration bit

Mapping simple logic functions to the Intel Cyclone IV/X FPGA

FPGA-based SoC Design

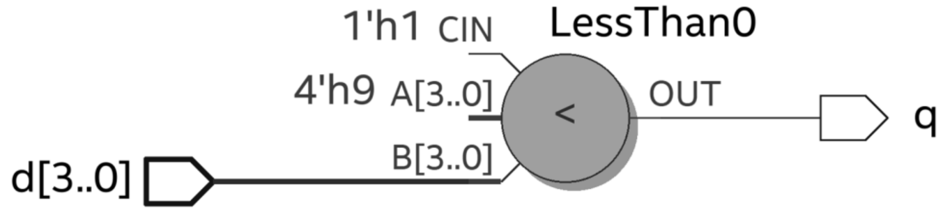
Greater than or equal to 9 ...

1/1

```
module bool_function_v1(
  input logic [3:0] d,
  output logic q
  );
  // ### logic function implementation ...
  assign q = (d >= 9) ? 1‘b1 : 1‘b0;

endmodule
```

[SystemVerilog] Source Code: bool_function_v1.sv - logic function >= 9



Synthesized circuit

Use case:

- Logic function implementation: Proof if a 4-bit logic input is greater than or equal to 9

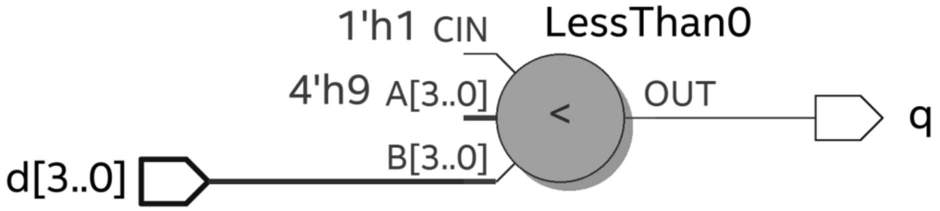
Greater than or equal to 9 ...

1/1

```
module bool_function_v2(
  input logic [3:0] d,
  output logic q
  );
  // ### logic function implementation ...
  always_comb
     if(d >= 9)
         q = 1‘b1;
     else
         q = 1‘b0;

endmodule
```

[SystemVerilog] Source Code: bool_function_v2.sv - logic function >= 9



Synthesized circuit

Notes

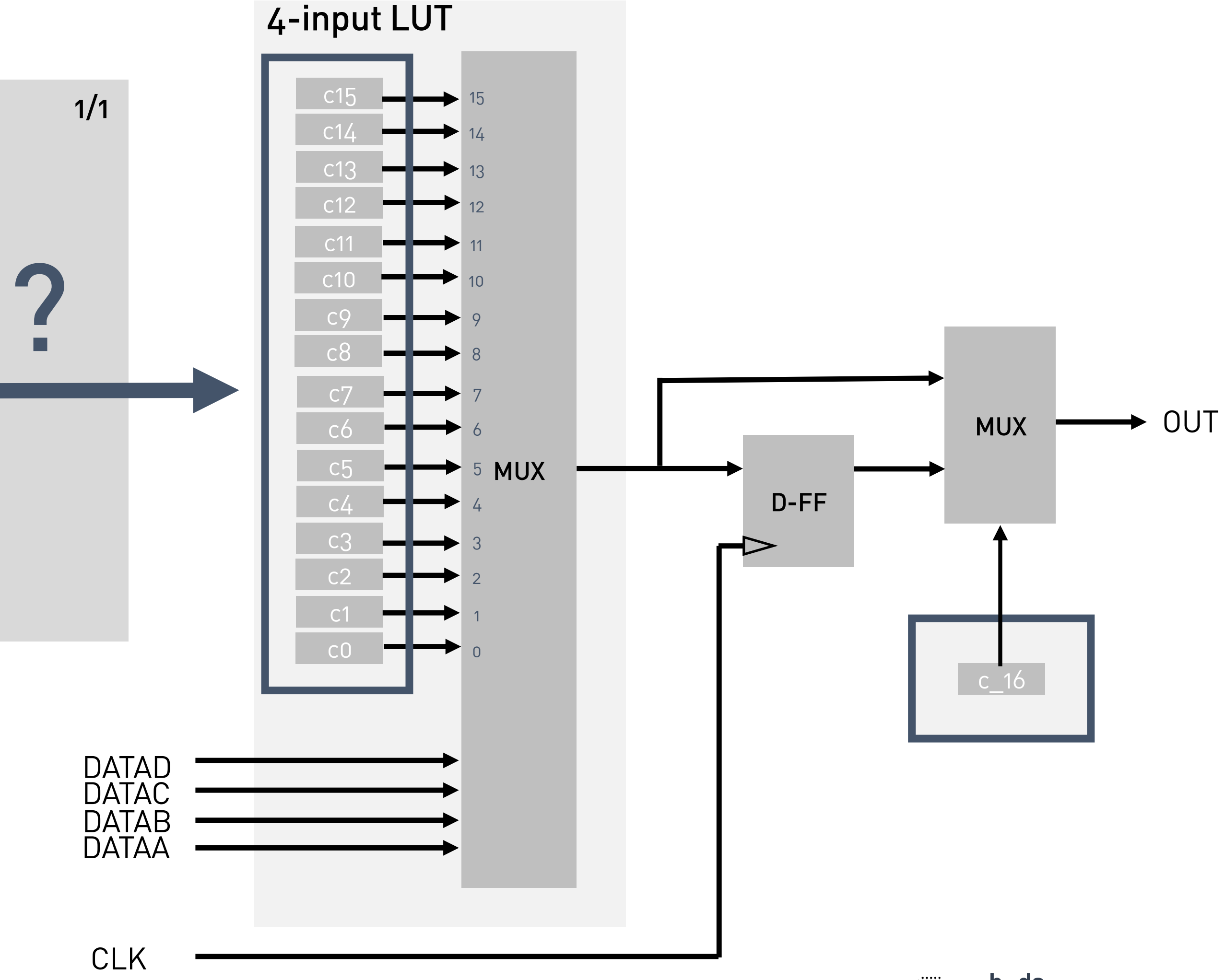
- Leads to the exact same synthesis result ...

Greater than or equal to 9 …

1/1

```
module bool_function_v2(
  input logic [3:0] d,
  output logic q
  );
  // ### logic function implementation ...
  always_comb
     if(d >= 9)
         q = 1‘b1;
     else
         q = 1‘b0;

endmodule
```



[SystemVerilog] Source Code: bool_function_v2.sv - logic function >= 9

Notes

- Leads to the exact same synthesis result …

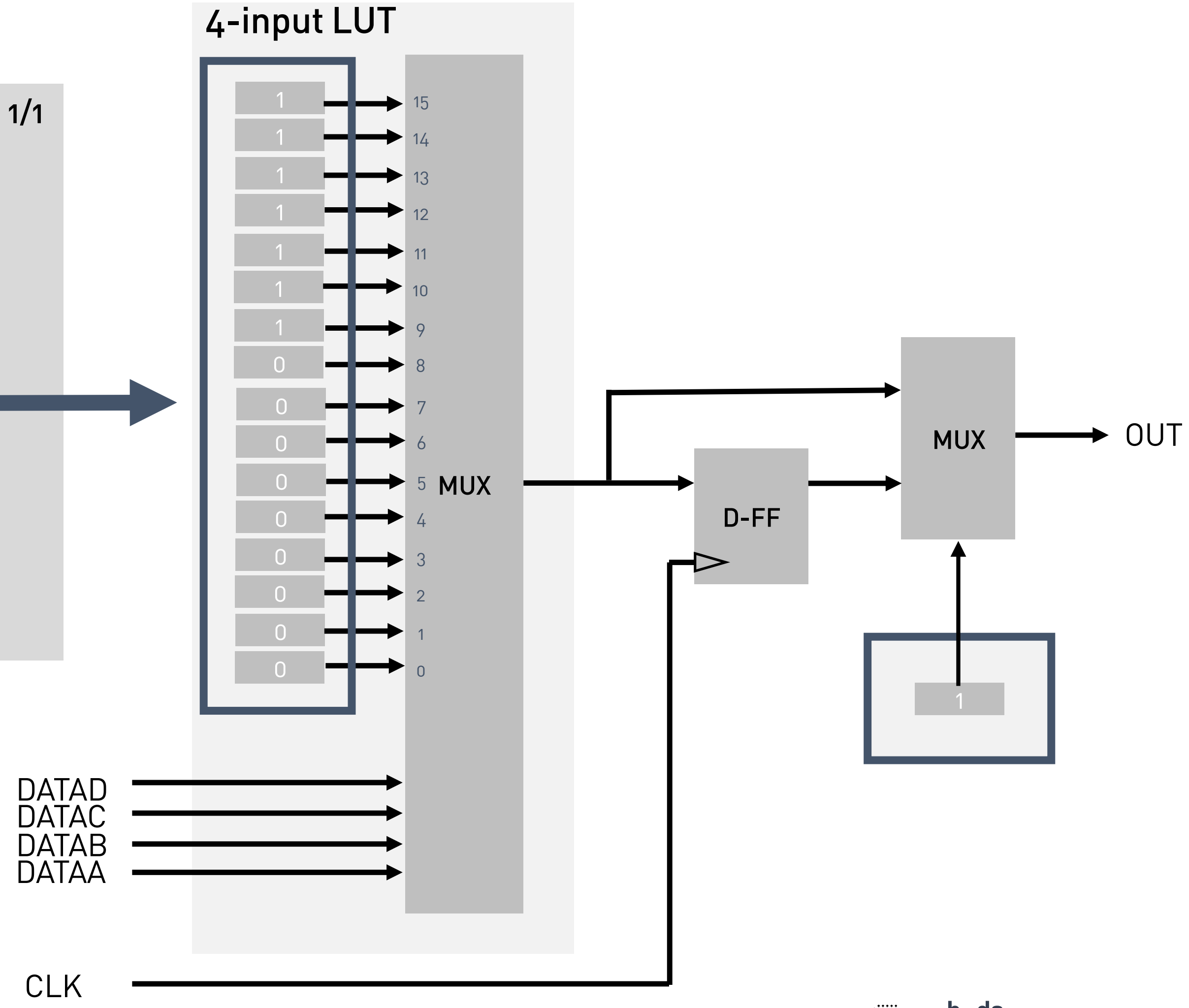
Greater than or equal to 9 ...

1/1

```
module bool_function_v2(
  input logic [3:0] d,
  output logic q
  );
  // ### logic function implementation ...
  always_comb
     if(d >= 9)
         q = 1‘b1;
     else
         q = 1‘b0;

endmodule
```

[SystemVerilog] Source Code: bool_function_v2.sv - logic function >= 9



Notes

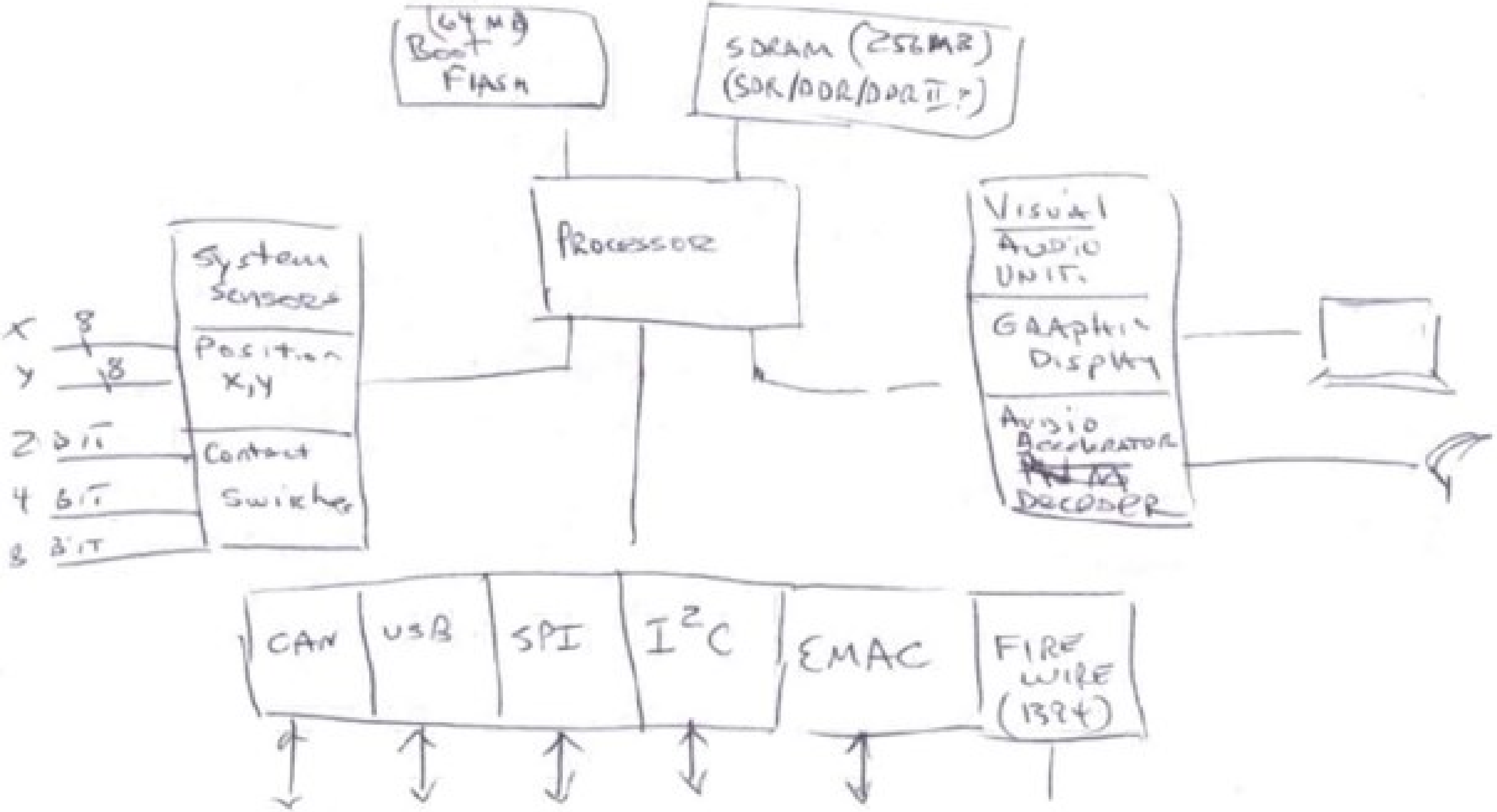
- Leads to the exact same synthesis result ...

SoC FPGA based System Design

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell …

- Everything starts with pencil and paper …

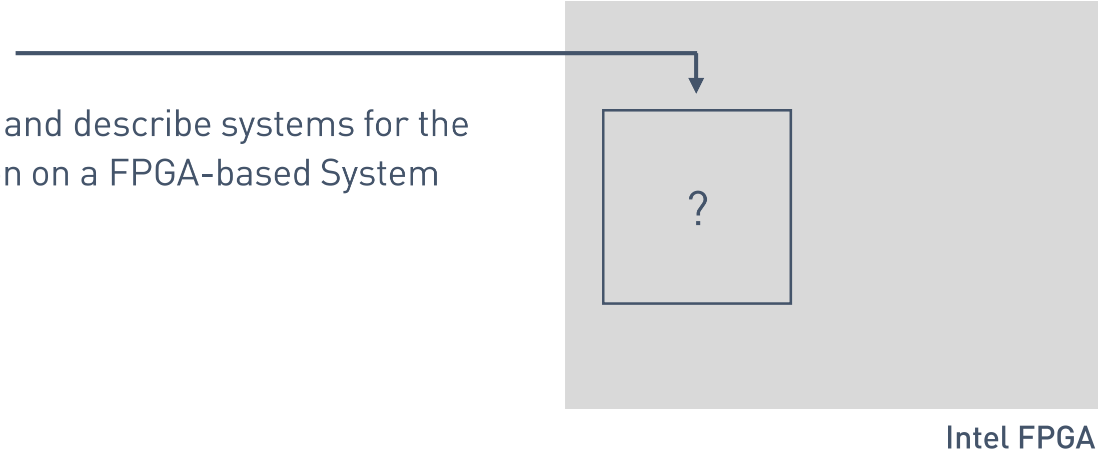


FSoC WS22/23 in a Nutshell ...

- This lecture addresses questions such as ...

First steps

- How to design and describe systems for the implementation on a FPGA-based System platform ...



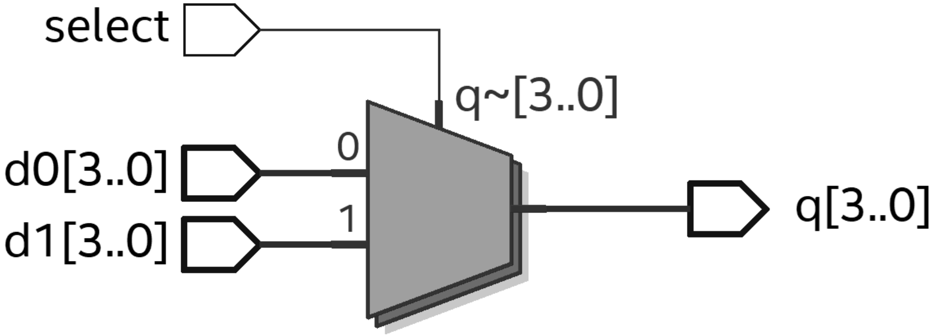
FSoC WS22/23 in a Nutshell ...

- Everything starts with ...

1/1

```
module mux_2_4bit(
  input  logic [3:0] d0, d1,
  output logic [3:0] q
  );
  always_comb
     if(select == 1’b1)
        q = d1;
     else
        q = d0;
endmodule
```

[SystemVerilog] Source Code: mux_2_4bit_comb.sv



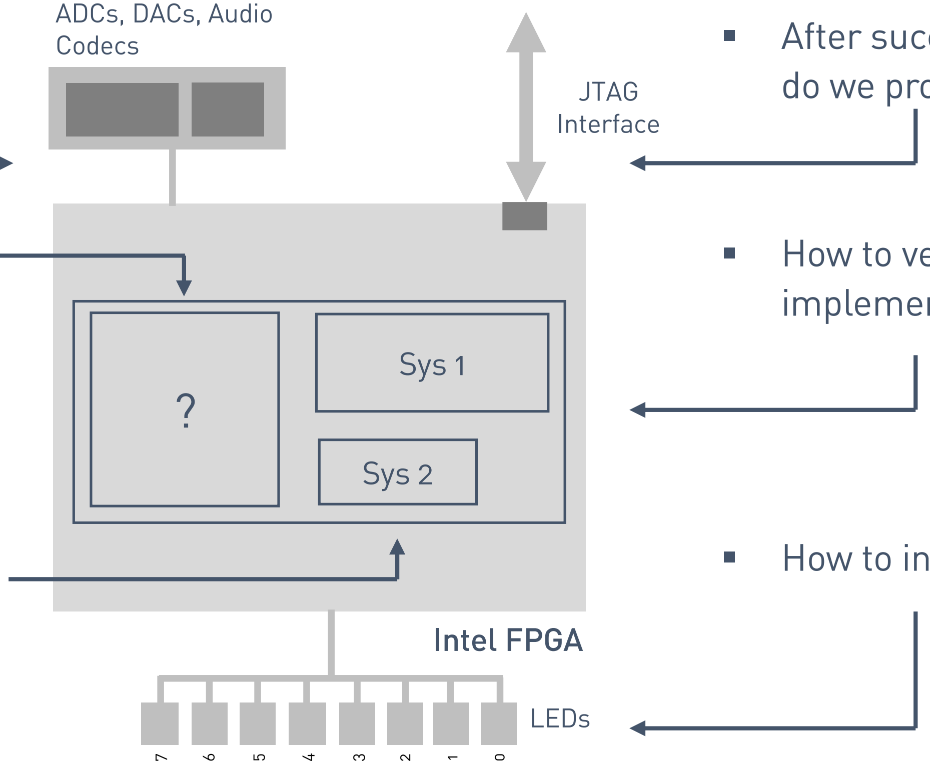
- How to design and describe systems in SystemVerilog for the implementation on a FPGA-based System platform ...

FSoC WS22/23 in a Nutshell ...

- This lecture addresses questions such as ...
  - How to interface “more complex” external devices ...

First steps

- How to design and describe systems for the implementation on a FPGA-based System platform ...
- How to create Systems with multiple hardware instances ...



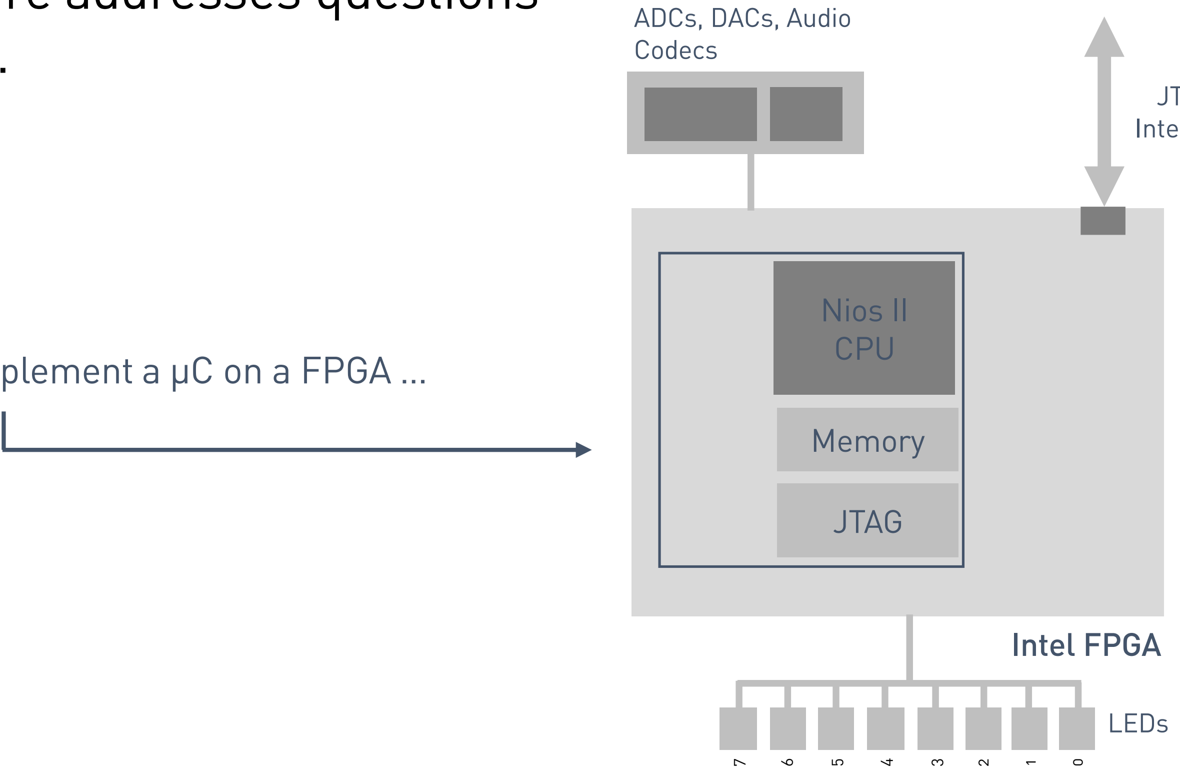
- After successful functional verification, how do we program the FPGA
- How to verify the correct functionality implemented?
- How to interface simple external systems ...

Softcore CPU based FPGA Embedded Systems

FPGA-based SoC Design

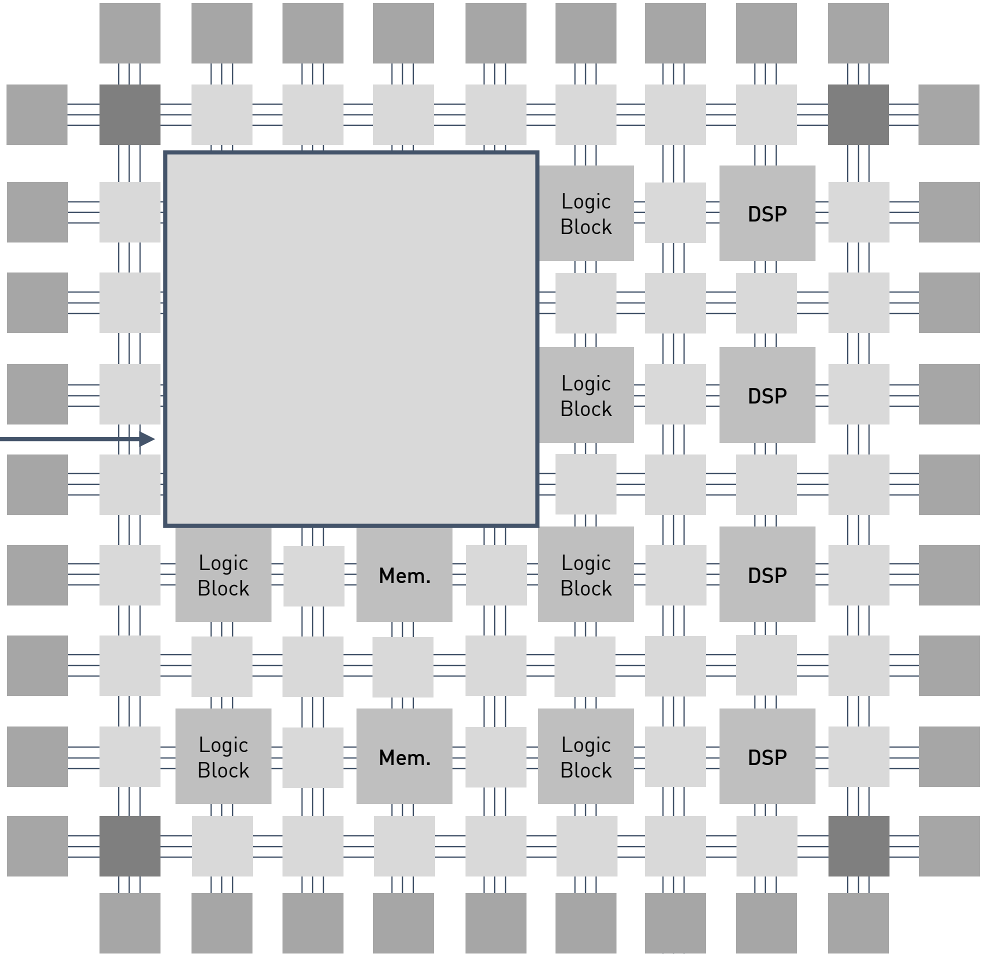
FSoC WS22/23 in a Nutshell ...

- This lecture addresses questions such as ...
  - How to implement a µC on a FPGA ...



FSoC WS22/23 in a Nutshell ...

- FPGA Softcore Processor

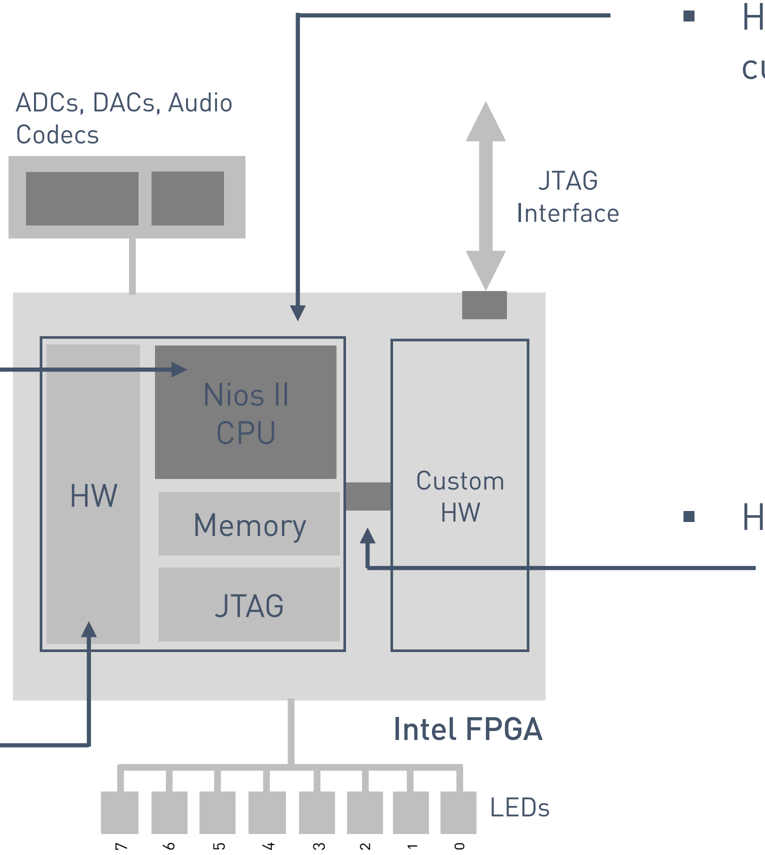


Softcore Processor

Mapped to FPGA Logic Blocks

FSoC WS22/23 in a Nutshell ...

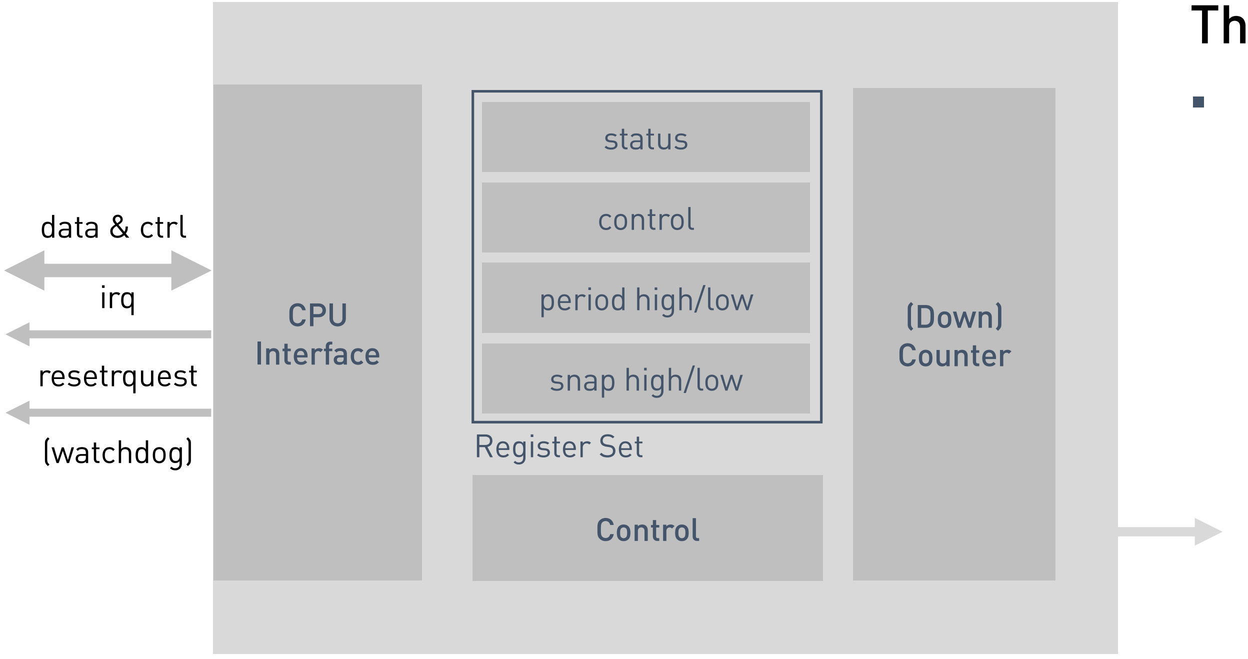
- This lecture addresses questions such as ...
  - How to program a µC implemented on a FPGA ...
  - How to implement a µC on a FPGA ...
  - How to integrate custom hardware into the peripheral set of the processor ...
  - How to write software drivers for accessing custom HW peripherals
  - How to interface custom hardware ...



Designing Hardware Peripherals for Softcore CPU Systems

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell ...



Custom Hardware Peripheral
(Timer)

The HW part ...

- How to design custom peripherals ...

FSoC WS22/23 in a Nutshell ...

1/1

```
#include “system.h”

typedef struct {
   __IO unsigned int status;
    __IO unsigned int control;
    __IO unsigned int peridl;
    __IO unsigned int period;
    __IO unsigned int snapl;
    __IO unsigned int snaph;
   } timer_t;

#define timer_0  (*((volatile timer_t *)  (0x8000000 | 0xA0001200 )));

...

timer_0.periodl = 0xFFFF;
```

[C] Source Code: Nios II hardware timer

The SW part ...

- How to access them from SW ...

FPGA Design - Timing Analysis and Closure

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell ...

- This lecture also addresses questions such as ...

  ... how to verify the timing of a designed system

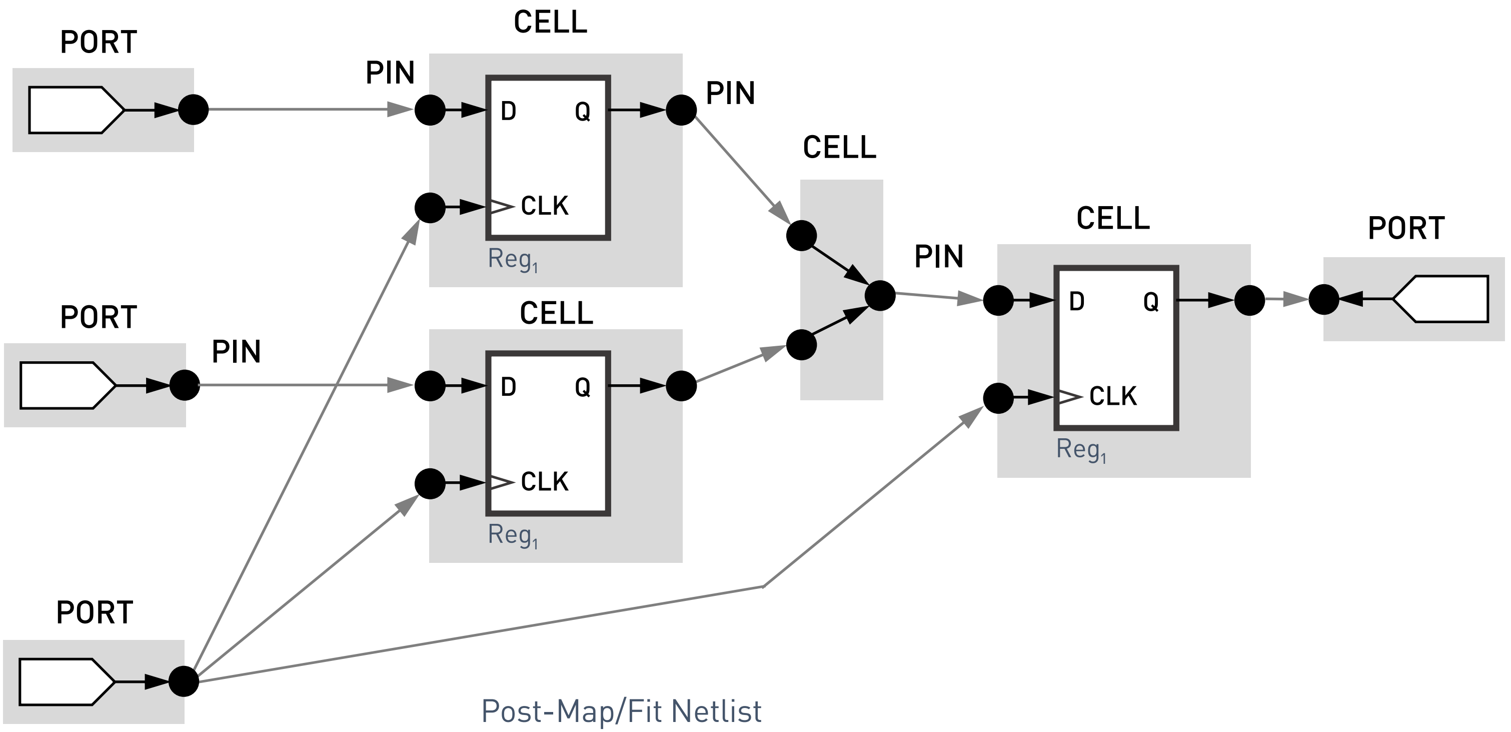
FSoC WS22/23 in a Nutshell ...

- How to verify timing ...

```
module sys_proc(
   input  logic [3:0] d0, d1,
   ...
   );
   always_ff@(posedge clk)
      if(reset_n == 1’b1)
         q = 1’b0;
      ...

endmodule
```

[SystemVerilog] Source Code Excerpt: sys_proc.sv

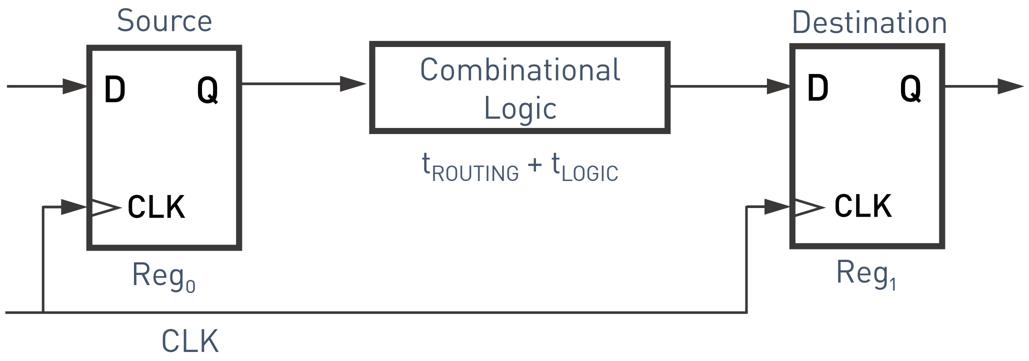


Post-Map/Fit Netlist

- The synthesis tool translates the initial HDL design into a netlist which consists of basics components such as cells (logic and/or DFFs), (in and output) ports and pins.

FSoC WS22/23 in a Nutshell …

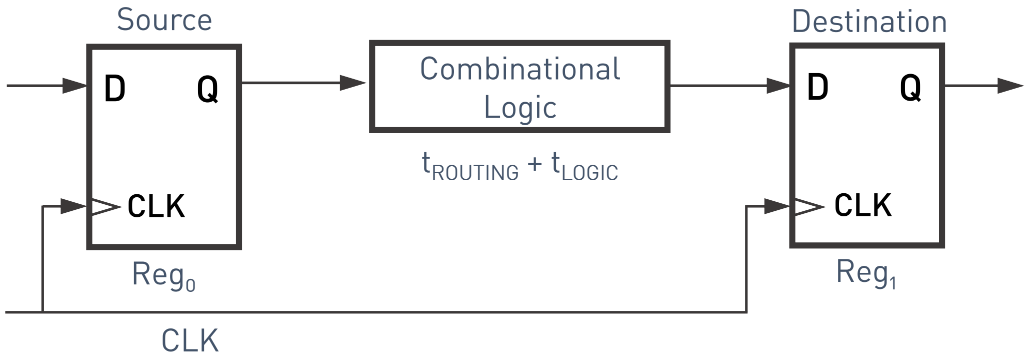
- Starting point



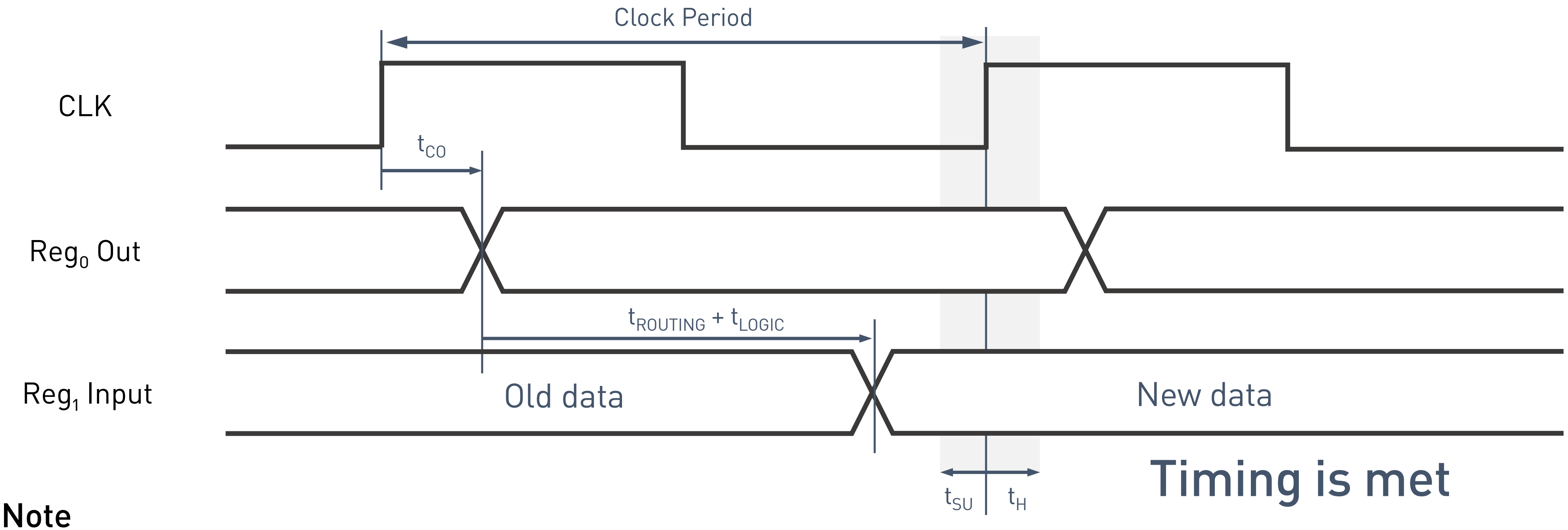
Scenario #1 – Timing is met …

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell ...



Scenario: Register transfer takes place within one clock cycle



Timing is met

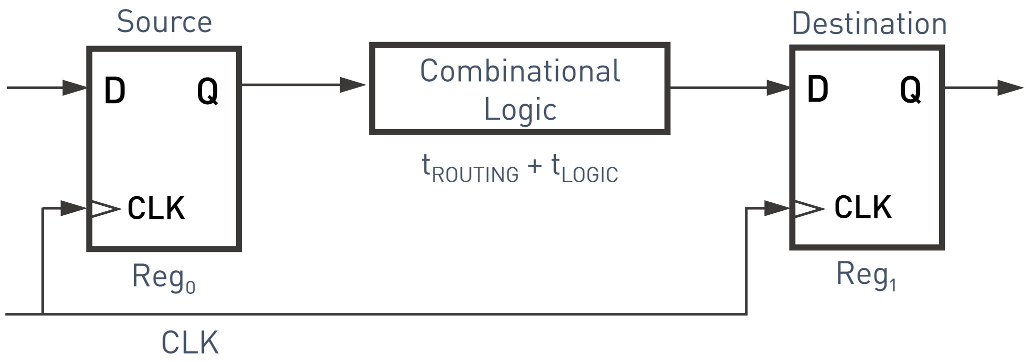
Note

- Short path delay: $T_{CLK} > t_{CO} + t_{LOGIC} + t_{ROUTING} + t_{SU}$

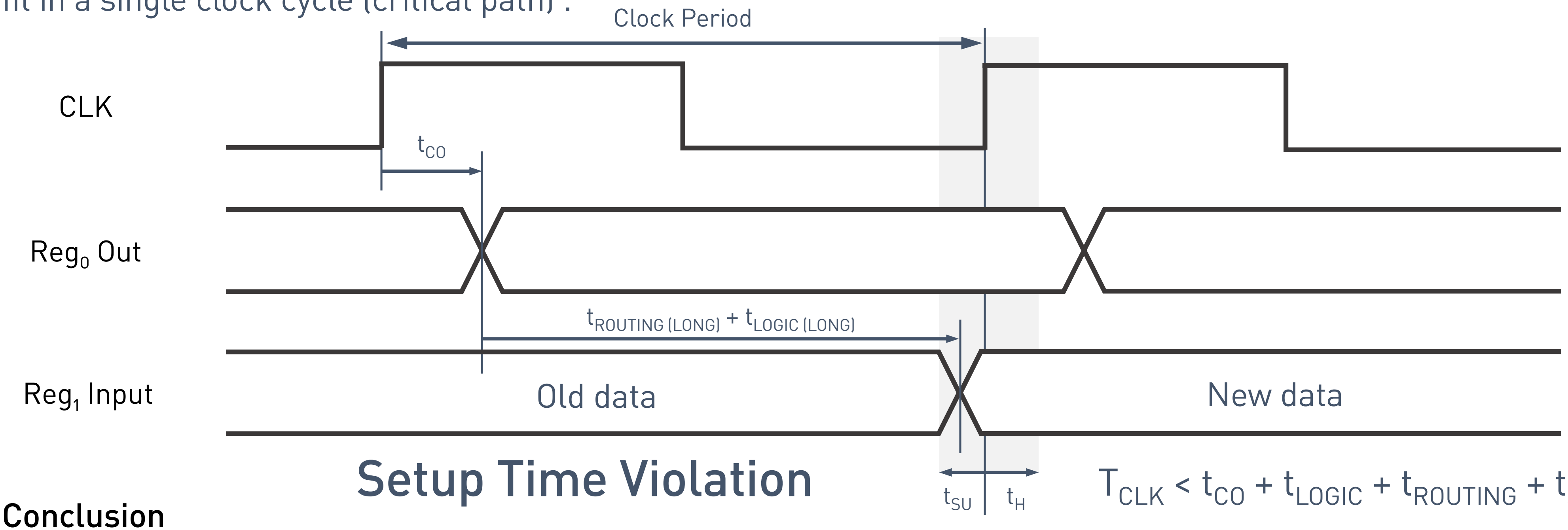
Scenario #2 – Timing is not met ...

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell …



Scenario: We are considering the longest register to register delay in a design. In the present case, we assume that the register transfer does fit in a single clock cycle (critical path) .



Setup Time Violation

$$T_{CLK} < t_{CO} + t_{LOGIC} + t_{ROUTING} + t_{SU}$$

Conclusion

- In order to meet timing, the clock period must be short enough so that the data can reach the destination register before the next rising edge of the clock.

FPGA Design – Resource and Power Optimization

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell …

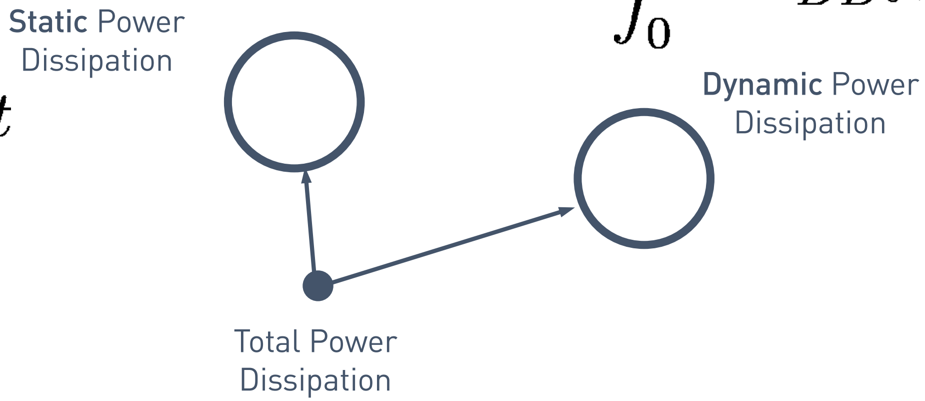
- This lecture also addresses questions such as …

… how to verify the timing of a designed system

… how to analyze and optimize the power consumption of a designed system

$$\int_0^t CV_{DD}^2 f_S dt$$

$$\int_0^t V_{DD} I_L dt$$



FSoC WS22/23 in a Nutshell ...

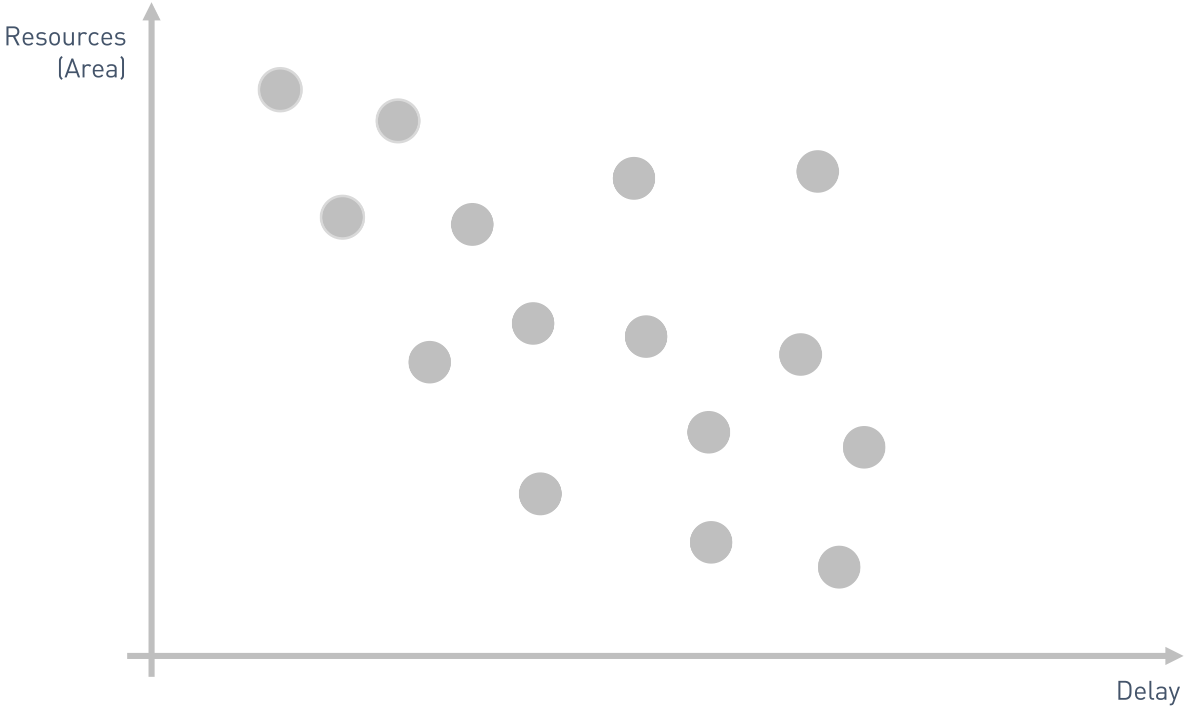
- This lecture also addresses questions such as ...

  ... how to verify the timing of a designed system

  ... how to analyze and optimize the power consumption of a designed system

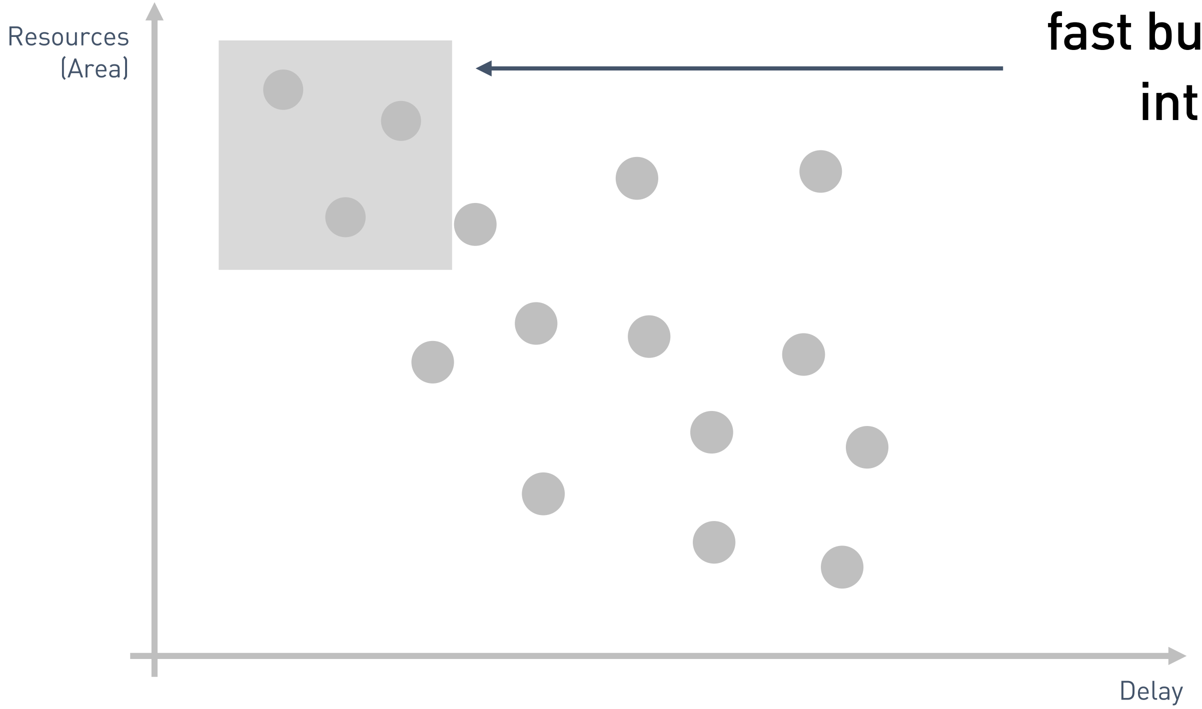
  ... how to efficiently use the design tools in order to achieve **timing**, **resource** or **power** related system constraints

FSoC WS22/23 in a Nutshell ...



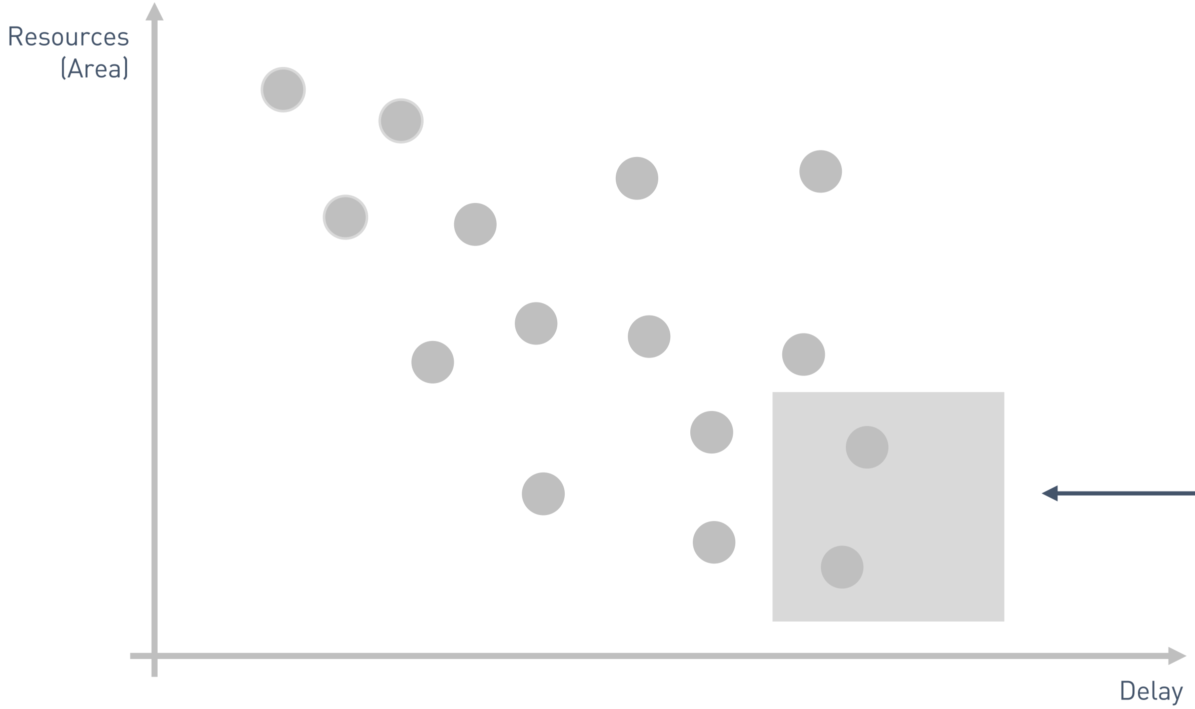
- 15 different designs, each implementing the same functionality
- Which one to choose?

FSoC WS22/23 in a Nutshell ...



fast but resource intensive

FSoC WS22/23 in a Nutshell ...

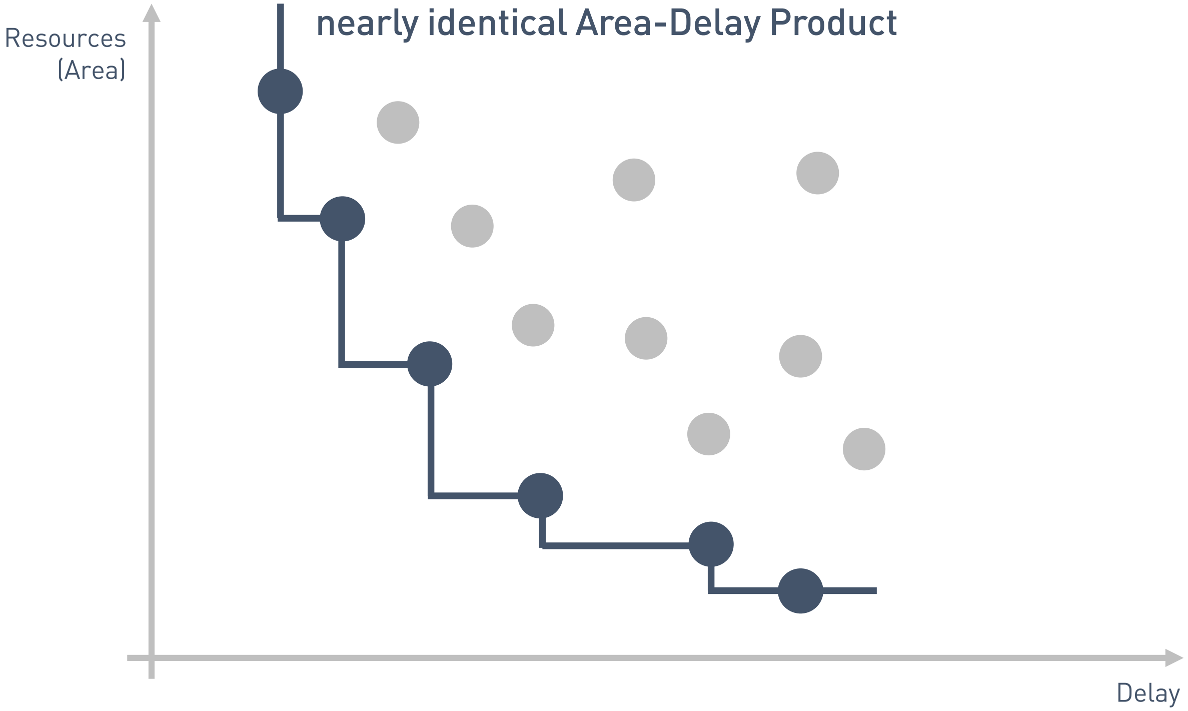


low resource usage but pretty slow ...

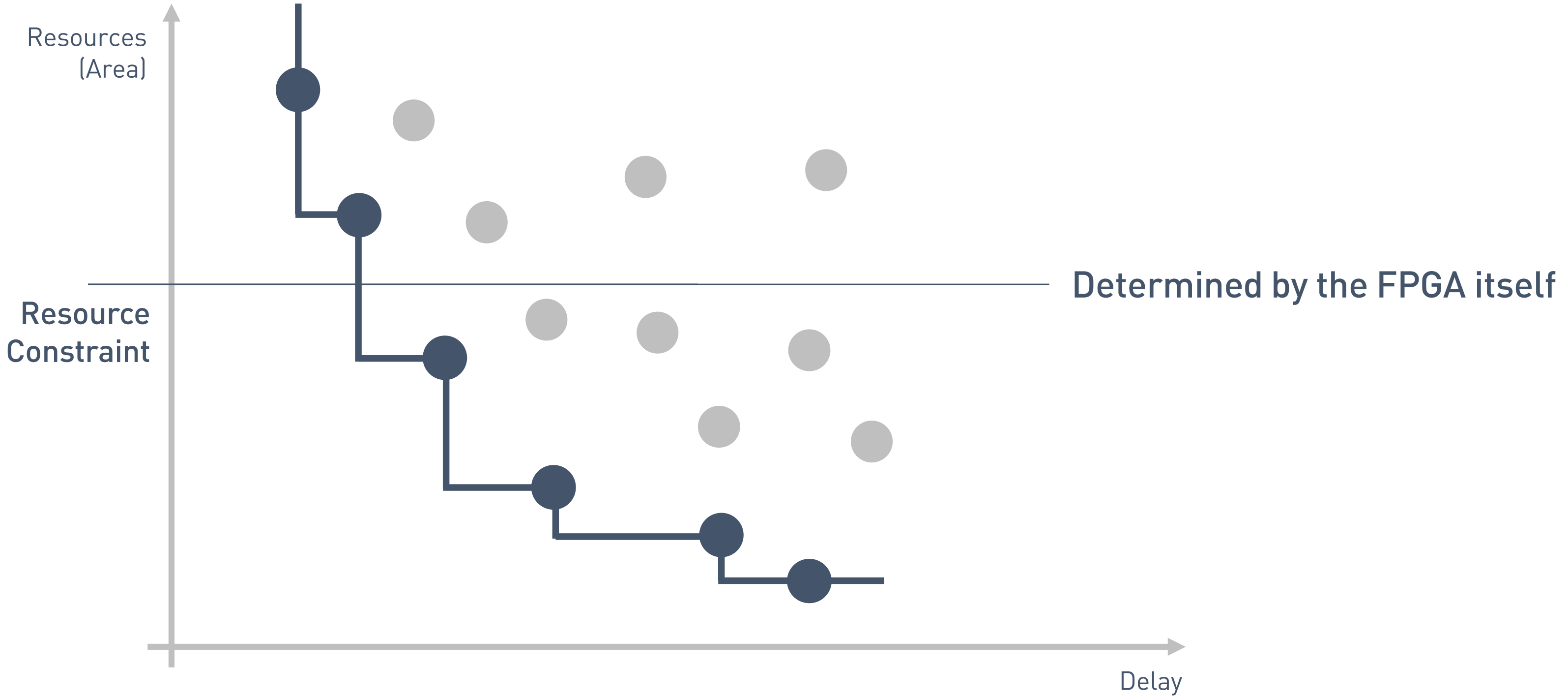
FSoC WS22/23 in a Nutshell ...

- Which one to use ...?

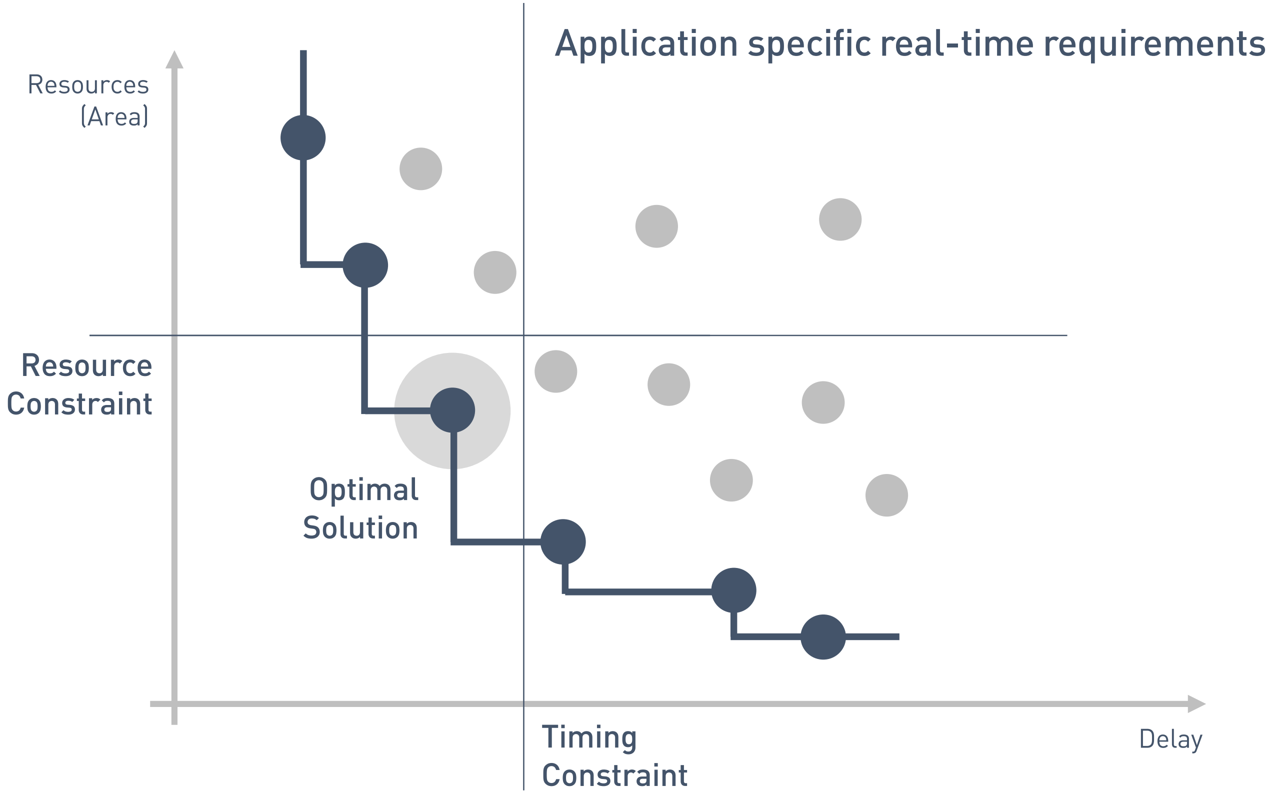
FSoC WS22/23 in a Nutshell ...



FSoC WS22/23 in a Nutshell ...



FSoC WS22/23 in a Nutshell ...



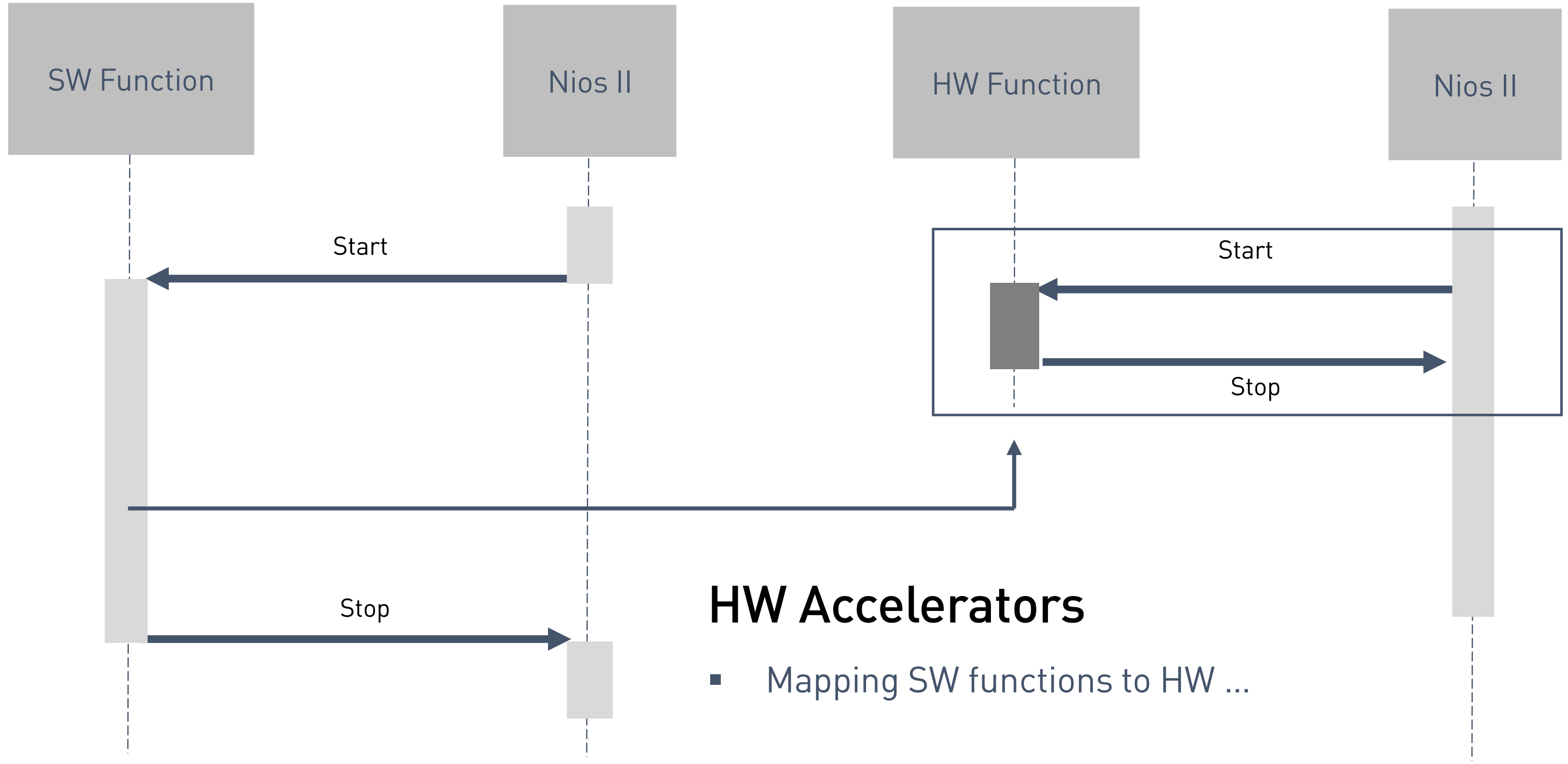
How to Speed up Software?

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell ...

- How to speed up software ?

FSoC WS22/23 in a Nutshell ...



- Parallel execution ...

HW Accelerators

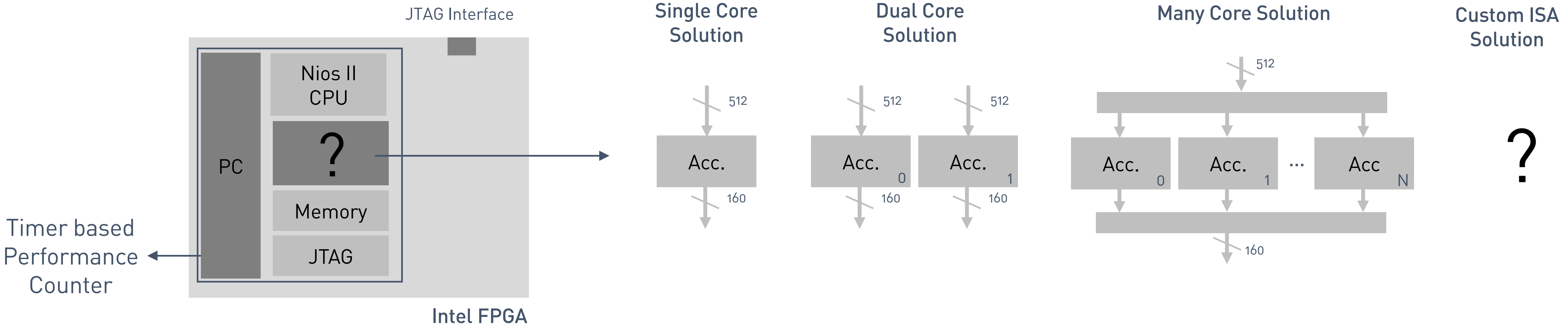
- Mapping SW functions to HW ...

How to Design Custom Hardware Accelerators

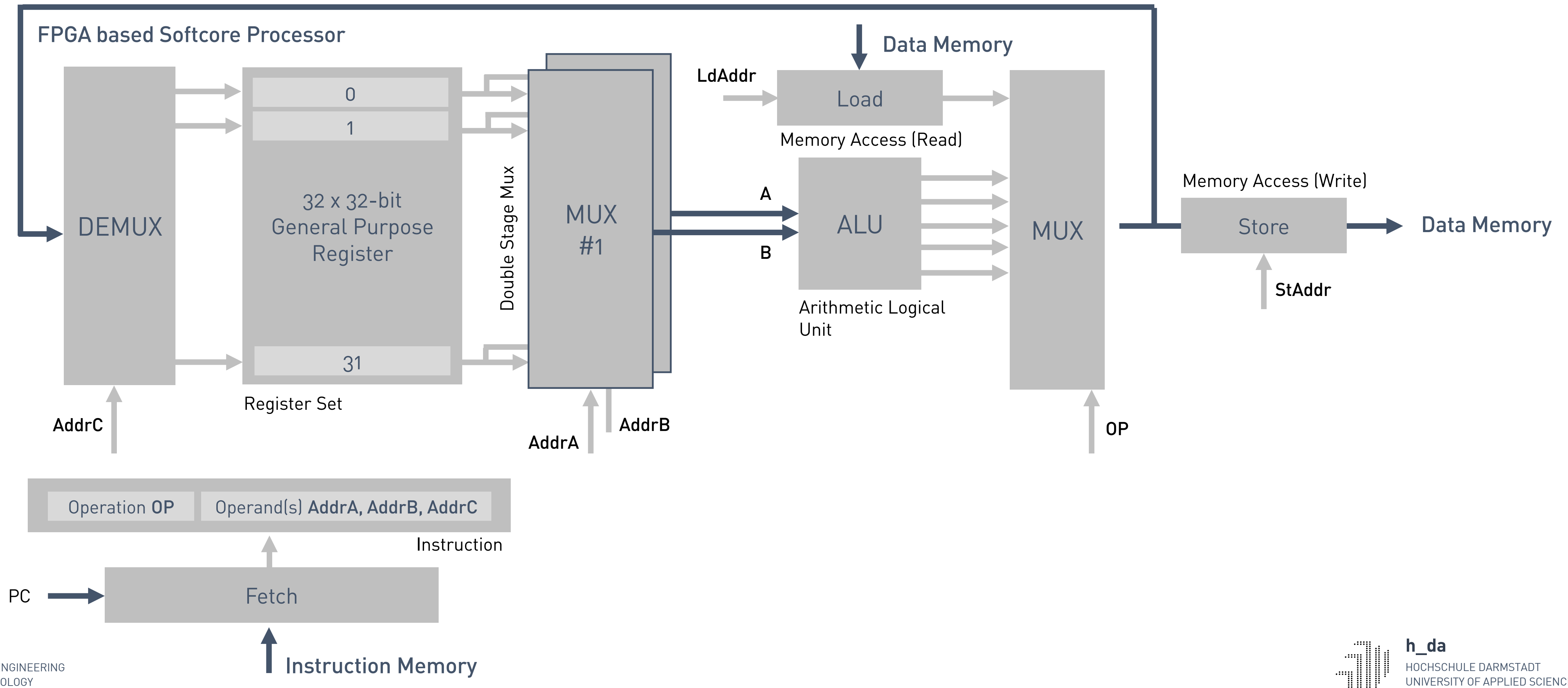
FPGA-based SoC Design

FSoC WS22/23 in a Nutshell ...

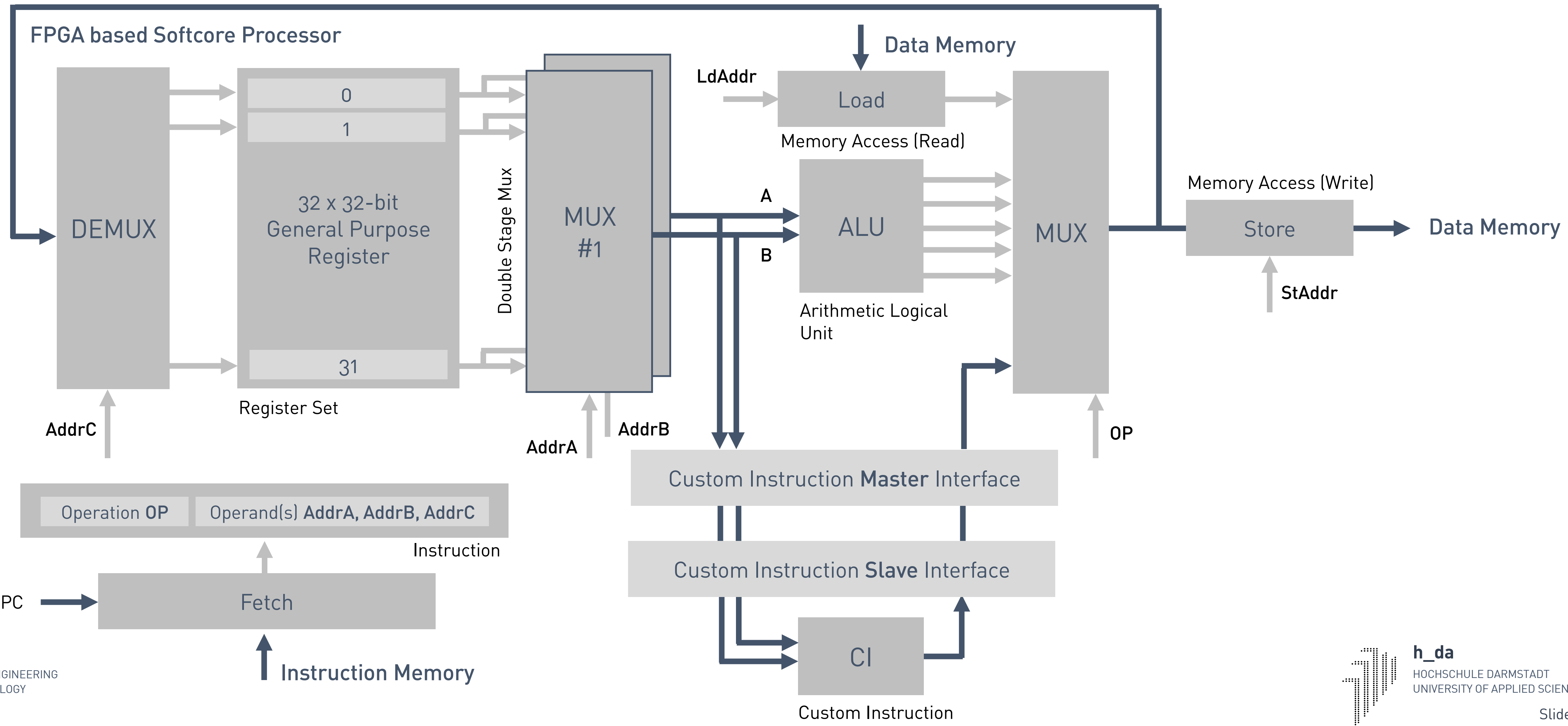
- How to design custom HW Accelerators ...



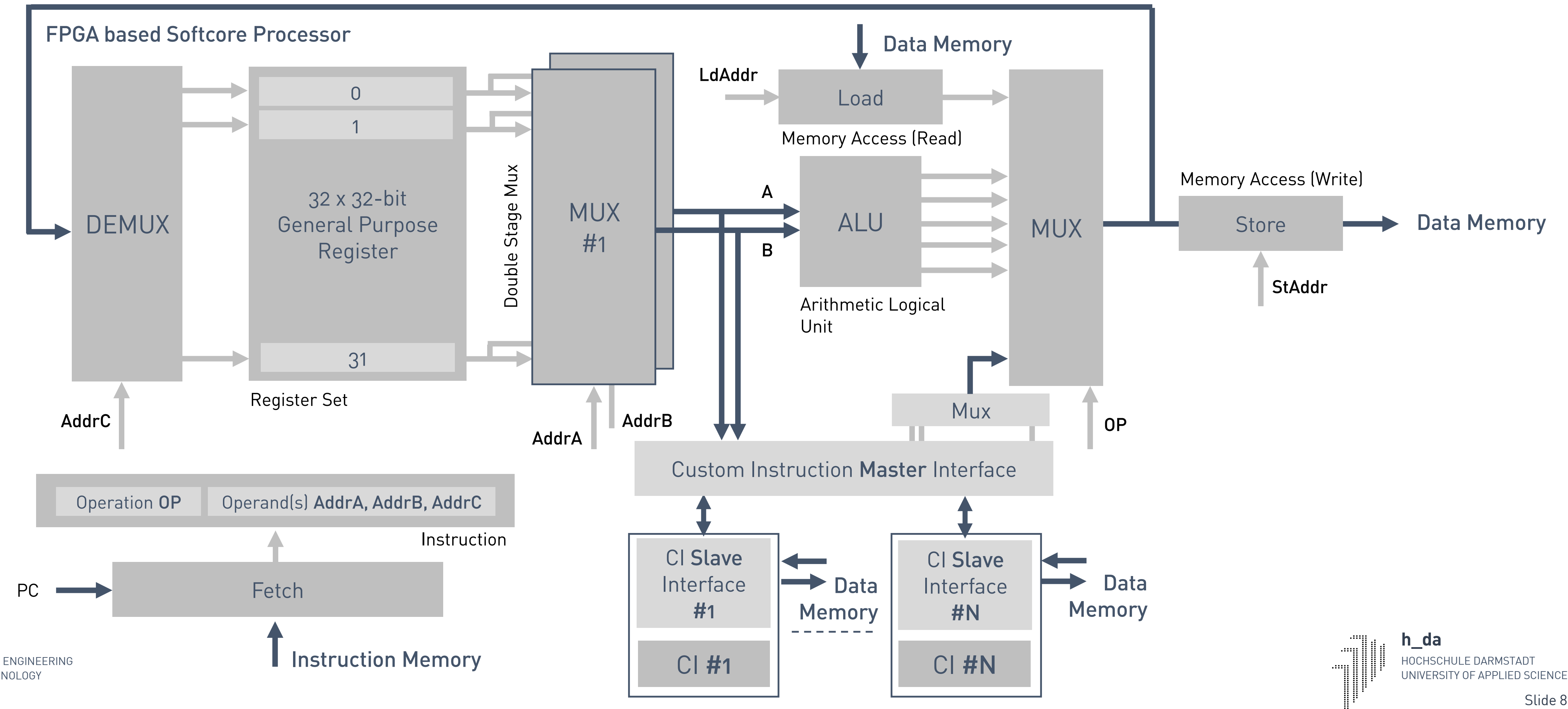
FSoC WS22/23 in a Nutshell ...



FSoC WS22/23 in a Nutshell ...



FSoC WS22/23 in a Nutshell ...

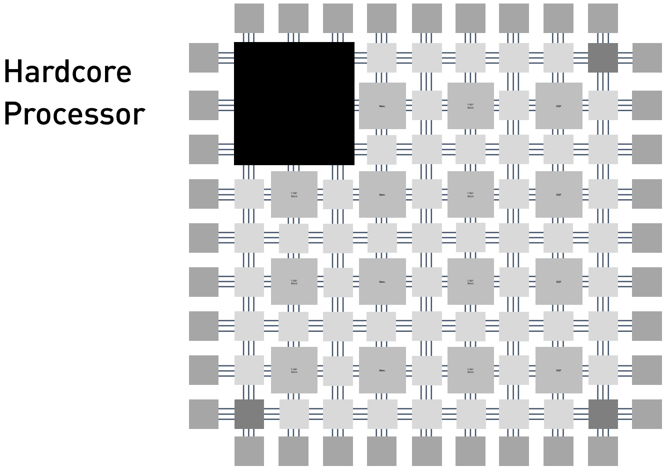


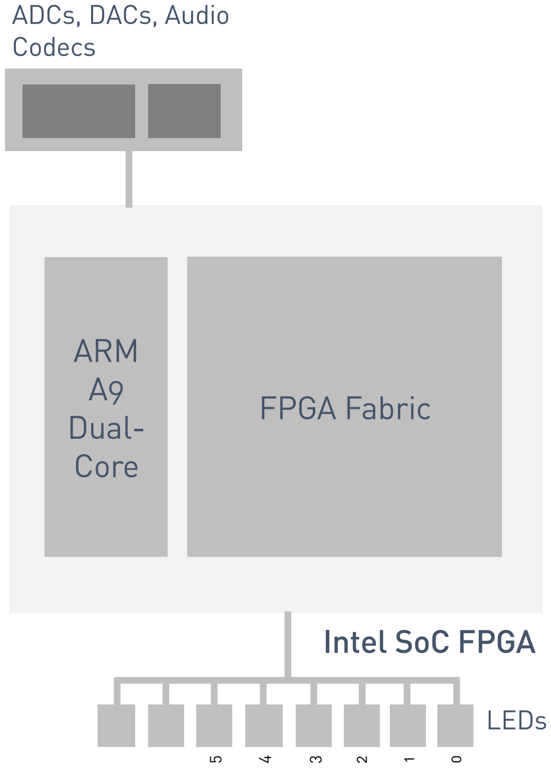
Configuring the SoC FPGA to run the Linux OS

FPGA-based SoC Design

FSoC WS22/23 in a Nutshell …

- Hard Processor Systems using Embedded Linux
  - How to design FPGA based SoC with integrated ARM A9 processor





The HPS integrates further peripherals:

- Several system control modules for clock and reset
- Debug support
- Timers (periodic and watchdog)
- Direct Memory Access Engine
- Ethernet Interface
- USB Interface
- UART
- CAN bus Controller
- SPI master and slave
- GPIO

Roadmap for next week

International Master of Science in Electrical Engineering

Prof. Dr. C. Jakob

University of Applied Sciences Darmstadt

h_da

Faculty of Electrical Engineering and Information Technology

fbeit

Roadmap for next week

- Lecture #1 - Monday, 24th October, 10.15am
- Lecture #2 - Tuesday, 25th October, 12.30pm – Room tba.