System on Chip Design

Jil Zerndt HS 2025

Summary

Introduction

Course Information

Course: System on Chip Design (SCD)
Institution: Zürcher Hochschule für Angewandte Wissenschaften (ZHAW)
Instructors:
- Tobias Welti (welo@zhaw.ch, +41 58 934 67 30)
- Dominique Cachin (cacd@zhaw.ch, +41 79 45559 01)

Course Materials and Schedule

- **Platform:** Moodle - https://moodle.zhaw.ch/course/view.php?id=25948
- **Script:** Available on Moodle (scd_script.pdf)
- **Lab Instructions:** https://github.zhaw.ch/pages/hpmm/scd-labs/index.html

Assessment and Grading

Grading Components

- **Electronic Quiz:** 15% (November 11, 2025, Moodle test)
- **Lab Exercises:** 15% (6 labs during semester, graded by lecturer)
- **Written Exam:** 70% (January 2026, Moodle test)

Lab Grading System

Seven labs (four lessons each) contribute to the lab grade.
Credits per Lab:
- Not done: 0 points
- Required tasks done with small errors: 1 point
- Required tasks done without errors: 2 points

Lab Grade Formula:

$$\text{Lab Grade} = \frac{\text{Sum of Points}}{12} \times 5 + 1$$

Exam Guidelines:
- Open book: lecture and lab notes, personal notes, books allowed
- No generative AI such as ChatGPT
- Calculators allowed

Course Objectives

Target Audience

This course is designed for engineers who want to:
- Design high-performance digital circuits with SoC-FPGAs, beyond writing VHDL code
- Gain in-depth background knowledge of SoC and FPGA (for software engineers)
- Design systems with Linux on SoC-FPGA
- Obtain introduction and basic knowledge of Integrated Circuit design
- Design general high-speed digital systems with complex peripherals (DDRAM)

Learning Goals

By the end of this course, students will be able to:
- Work with FPGA block memory
- Configure a FPGA-SoC with ARM hardcore processor
- Configure the I/O and computer peripherals (DRAM) of a FPGA
- Port Yocto Linux to SoC-FPGA
- Configure and analyse timing to drive synthesis
- Configure clock generators in FPGA and route clocks on PCB
- Check signal integrity of clock and data lines on PCB
- Explain differences between different signaling standards
- Connect high-speed FPGA peripherals with differential signals
- Realize a project with video and audio output (Pacman game)

Lab Setup

Laboratory Environment

Location: Lab TE 519
Equipment:
- Lab PCs with required software setup running on Linux
- DE1-SoC Development Board with Intel Cyclone V SoC FPGA
- Hardware only available in lab (not distributed to students)

Work Organization:
- Students work in teams of two
- Lab instructions available on GitHub

Introduction to System on Chip Design

Fundamentals of Sequential Digital Circuits

General Representation of Sequential Circuits

Sequential circuits consist of two main components:

- **D-Flip-Flops:** Storage elements that hold state
- **Combinatorial Logic:** Logic circuits that compute next state and outputs
- **Common Clock:** Synchronizes all flip-flops

FPGA Architecture

SRAM-based FPGAs

SRAM-based FPGAs use **Lookup Tables (LUTs)** to generate logic functions.
Key Concept: A LUT is essentially a small RAM (e.g., 4x1 RAM) that can implement any Boolean function of its inputs.

Logic Cells (ALM)

ALM: Adaptive Logic Module (Altera/Intel terminology)
Different FPGA manufacturers use different names:

- **Altera (Intel):** Adaptive Logic Module (ALM)
- **Xilinx:** Configurable Logic Block (CLB) or Slice

Each logic cell typically contains:

- Lookup tables (LUTs)
- Flip-flops
- Multiplexers
- Carry logic

Classic FPGA Architecture

A traditional FPGA consists of:

- **FPGA Fabric:** Array of configurable logic cells
- **Interconnection Network:** Programmable routing between logic cells
- **Logic Cells:** Containing lookup tables and flip-flops

Additional FPGA Resources

Modern FPGA fabric may also contain:

- **SRAM Blocks:** Embedded memory blocks for data storage
- **DSP Blocks:** Dedicated hardware for digital signal processing
- **Hard IP Blocks:** Pre-designed functional blocks (e.g., PCIe, memory controllers)

FPGA Applications

Home Electronics: FPGAs are used in consumer devices for video processing, display controllers, and smart home automation.

Medical Diagnostics: FPGAs provide real-time signal processing for medical imaging equipment and diagnostic devices.

Industrial Automation: FPGAs enable precise control and monitoring in industrial systems.

Studio Equipment: Professional audio and video equipment uses FPGAs for real-time processing and effects.

Products where FPGAs are NOT ideal:

- Battery-operated mobile devices (high power consumption)
- Very high-volume products like mobile phones (cost per unit too high)
- Trivial control applications (washing machine, bike computer - overkill)
- Automotive applications (reliability and certification requirements)
- Some aerospace applications (radiation hardness requirements)

Evolution from FPGA to SoC

System on Chip (SoC) FPGA

Modern SoC FPGAs combine:

- **Hard Processor System (HPS):** Fixed CPU subsystem
- **FPGA Fabric:** Programmable logic
- **Bridges:** Connections between processor and FPGA

This integration enables software and hardware co-design on a single chip.

SoC FPGA Components

CPU Portion (Hard Processor System):

- Cortex-A9 CPU Subsystem
- Flash Controllers
- SDRAM Controller Subsystem
- On-Chip Memories
- Support Peripherals
- PLLs (Phase-Locked Loops)
- Debug Interface
- Peripherals Control Block

FPGA Portion:

- User I/O
- FPGA Fabric (LUTs, RAMs, Multipliers & Routing)
- HSSI Transceivers (High-Speed Serial Interface)
- PLLs
- Hard PCIe
- Hard Memory Controllers

Processor-FPGA Bridges: Enable communication between CPU and FPGA fabric

Topics Covered in SCD Course

Course Topics Overview

The following topics will be discussed throughout the course:

1. Bringing Linux to SoC
2. FPGA System Builder (Platform Designer)
3. Bootloader for Linux
4. RAM and FPGA on-chip Memory
5. DRAM (Dynamic Random Access Memory)
6. JTAG (Joint Test Action Group)
7. Timing Analysis and Synthesis
8. Clock Distribution and PLL
9. High-Speed I/O Interfaces
10. FPGA Configuration

Bringing Linux to SoC-FPGA

Linux on SoC-FPGA

Running Linux on SoC-FPGA enables:

- High-level software development using familiar tools
- Access to vast ecosystem of Linux software
- Network connectivity and modern protocols
- File systems and standard I/O
- Dynamic hardware reconfiguration from software

FPGA System Builder

Platform Designer

Platform Designer (formerly known as Qsys) is Intel's system integration tool for SoC FPGAs.

Functions:

- Graphical system design interface
- Integration of IP cores
- Automatic interconnect generation
- System-level simulation
- Export to Quartus for compilation

Bootloader for Linux

Boot Process

The bootloader is responsible for:

- Initializing hardware (CPU, memory, peripherals)
- Loading the Linux kernel into memory
- Passing boot parameters to the kernel
- Transferring control to the operating system

Common bootloaders: U-Boot, GRUB

Memory Systems

RAM and FPGA On-Chip Memory

On-Chip Memory Types:

- **Block RAM (BRAM):** Embedded memory blocks in FPGA fabric
- **Distributed RAM:** Memory implemented using LUTs
- **On-Chip RAM in HPS:** Fast memory integrated in processor system

Characteristics:

- Very fast access (single cycle or few cycles)
- Limited size (kilobytes to few megabytes)
- No refresh needed (SRAM-based)
- Higher cost per bit than external memory

Dynamic RAM (DRAM)

Inventor: Robert H. Dennard (invented at IBM in 1967)

Key Characteristics:

- **Storage Mechanism:** Stores data as charge in capacitors
- **Refresh Required:** Must be periodically refreshed (capacitors leak)
- **High Density:** Much higher storage density than SRAM
- **Lower Cost:** Cheaper per bit than SRAM
- **Slower Access:** Requires complex timing protocols

Common Types:

- DDR3, DDR4, DDR5: Double Data Rate SDRAM
- LPDDR: Low Power DDR for mobile devices

JTAG (Joint Test Action Group)

JTAG Interface

JTAG is a standard for testing and debugging electronic systems.

Primary Uses:

- Boundary scan testing of PCBs
- Programming FPGAs and flash memory
- Debugging processors (via debug interface)
- In-system programming

Signal Lines:

- **TDI:** Test Data In
- **TDO:** Test Data Out
- **TCK:** Test Clock
- **TMS:** Test Mode Select
- **TRST:** Test Reset (optional)

JTAG State Machine

The JTAG interface operates through a state machine with the following main states:

- **Test-Logic-Reset:** Initial state
- **Run-Test/Idle:** Idle state
- **Select-DR-Scan:** Select data register scan
- **Select-IR-Scan:** Select instruction register scan
- **Capture-DR/IR:** Capture data or instruction
- **Shift-DR/IR:** Shift data or instruction
- **Update-DR/IR:** Update data or instruction register

Timing Analysis and Synthesis

Static Timing Analysis

Static timing analysis verifies that a digital circuit meets all timing constraints without requiring simulation of the circuit's operation.

Key Concepts:

- **Setup Time (t_{su}):** Minimum time data must be stable before clock edge
- **Hold Time (t_h):** Minimum time data must remain stable after clock edge
- **Clock-to-Q Delay (t_{co}):** Time for data to appear at flip-flop output after clock
- **Propagation Delay:** Time for signals to travel through logic and routing

Timing Paths and Slack

Data Arrival Time is composed of:

1. Clock network delay from clock node to launch D-FF: 3.00 ns
2. Clock-to-Q delay from launch D-FF: 2.00 ns
3. Logic delay (cells, I/O pins, routing): 5.00 ns

Total Data Arrival Time: 10.00 ns

Data Required Time is determined by:

1. Clock period: 40.00 ns
2. Clock network delay to latch D-FF: 2.00 ns
3. Setup time of latch D-FF: -0.50 ns

Total Required Time: 41.50 ns

Setup Slack = Required Time - Arrival Time = 41.50 ns - 10.00 ns = 31.50 ns

Positive slack indicates timing is met. Negative slack indicates timing violation.

Clock Distribution and PLL

Phase-Locked Loop (PLL)

A **PLL** is a control system that generates an output signal whose phase is related to the phase of an input signal.

Key Functions:

- **Frequency Multiplication:** Generate higher frequency clocks
- **Frequency Division:** Generate lower frequency clocks
- **Phase Alignment:** Align clock phases
- **Jitter Reduction:** Clean up noisy clock signals
- **Clock Deskew:** Compensate for clock distribution delays

Clock Distribution Network

FPGAs have dedicated clock distribution networks:

- **Global Clock Networks (GCLK):** Low-skew distribution across entire device
- **Regional Clock Networks:** Lower skew within regions
- **Dual-Purpose Clock Networks (DPCLK):** Flexible routing
- **Clock Control Blocks (CDPCLK):** Gating and multiplexing

Proper clock distribution is critical for achieving timing closure in high-speed designs.

High-Speed I/O Interfaces

Differential Signaling

High-speed interfaces use **differential signaling** where information is transmitted using two complementary signals.

Advantages:

- Better noise immunity (common-mode noise rejection)
- Higher data rates possible
- Lower EMI emissions
- Better signal integrity over long distances

Common Standards:

- LVDS (Low-Voltage Differential Signaling)
- TMDS (Transition-Minimized Differential Signaling)
- SerDes (Serializer/Deserializer)

FPGA Configuration

FPGA Configuration Process

FPGAs are volatile devices that must be configured at power-up.

Configuration Sources:

- **Configuration Flash:** Non-volatile memory storing bitstream
- **JTAG:** Programming via debugger (Byte Blaster)
- **Serial Configuration:** Via SPI or similar interface
- **Parallel Configuration:** Faster configuration method

Configuration Modes:

- **Active Serial (AS):** FPGA controls configuration process
- **Passive Serial (PS):** External controller drives configuration
- **JTAG Mode:** For development and debugging

Configuration Pins:

- **nCE:** Active-low chip enable
- **nCONFIG:** Configuration control
- **CONF_DONE:** Configuration complete signal
- **DATA:** Configuration data lines

Typical pull-up/pull-down resistors: 10kΩ