SYSTEM MODELLING AND SYNTHESIS WITH HDL

DTEK0078
2022 Lecture 1

UNIVERSITY OF TURKU

Course Information

- Lecturer:
  - Associate Professor Tomi Westerlund
    tomi.westerlund@utu.fi

- Credits: 5 ects

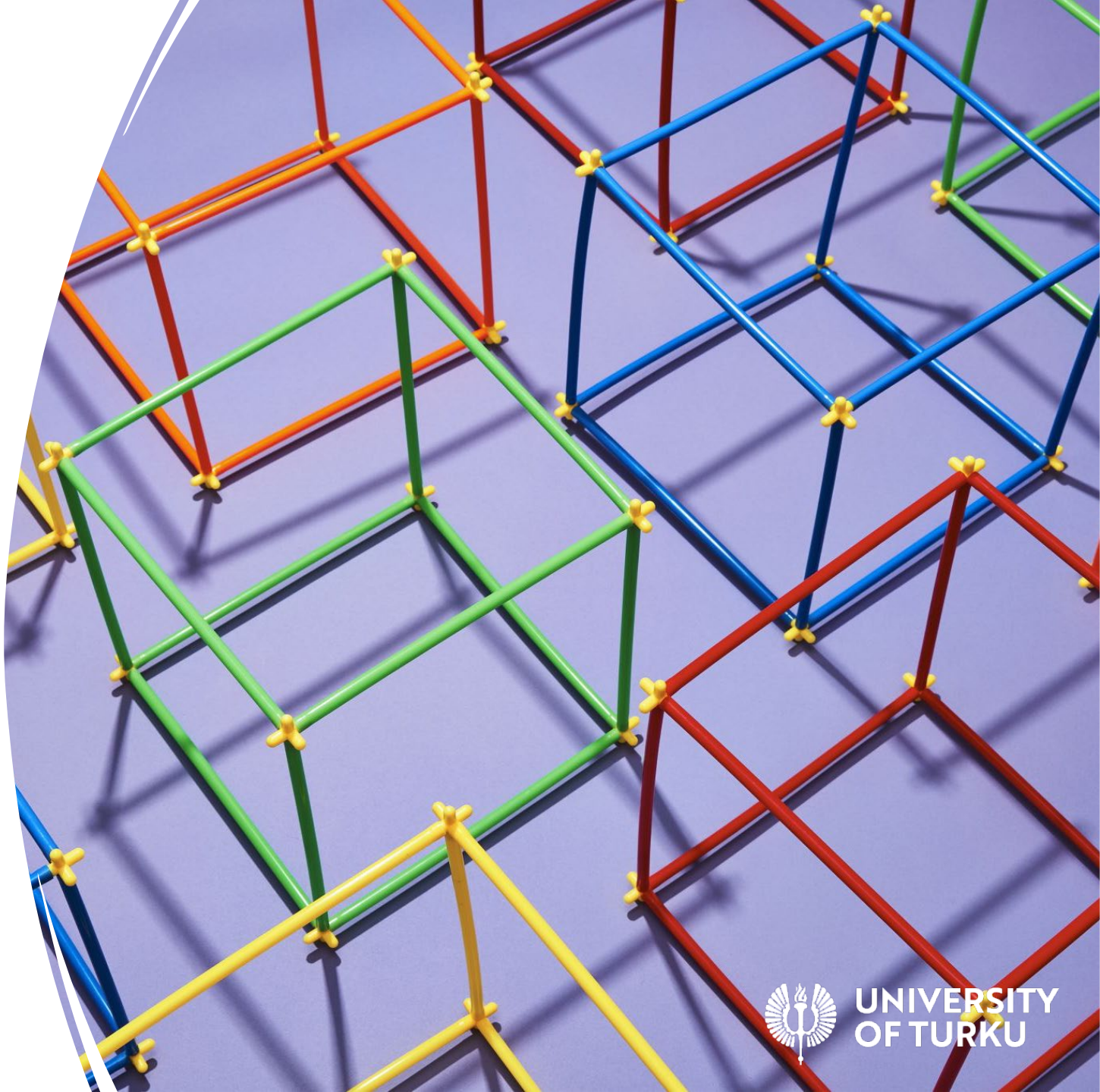
- Course Requirement:
  - Exercises

- Pre-Requirements: none
- Helpful knowledge: digital system design, basic knowledge of HDL and object-oriented programming

UNIVERSITY OF TURKU

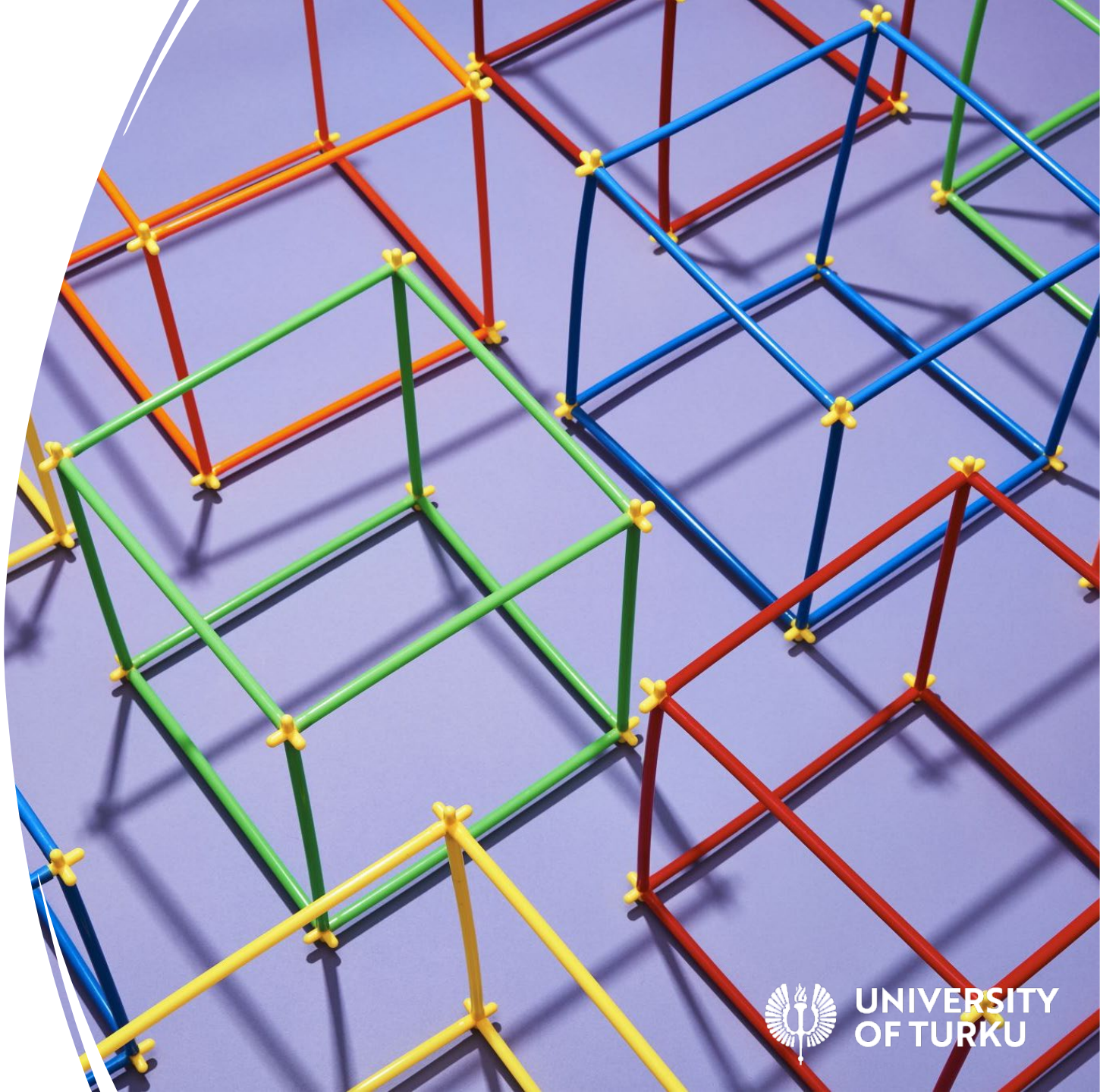
Lectures

- Intro, modelling, very basic VHDL syntax
- Main building blocks, Coding Styles, objects and data types,
- Hierarchical design, components, testbenches
- Statements, processes, control structures
- Statements, signals
- Simulation, timing, synchronous design,
- Storage
- Finate state machines, generics
- Routines (functions and procedures)



Labs

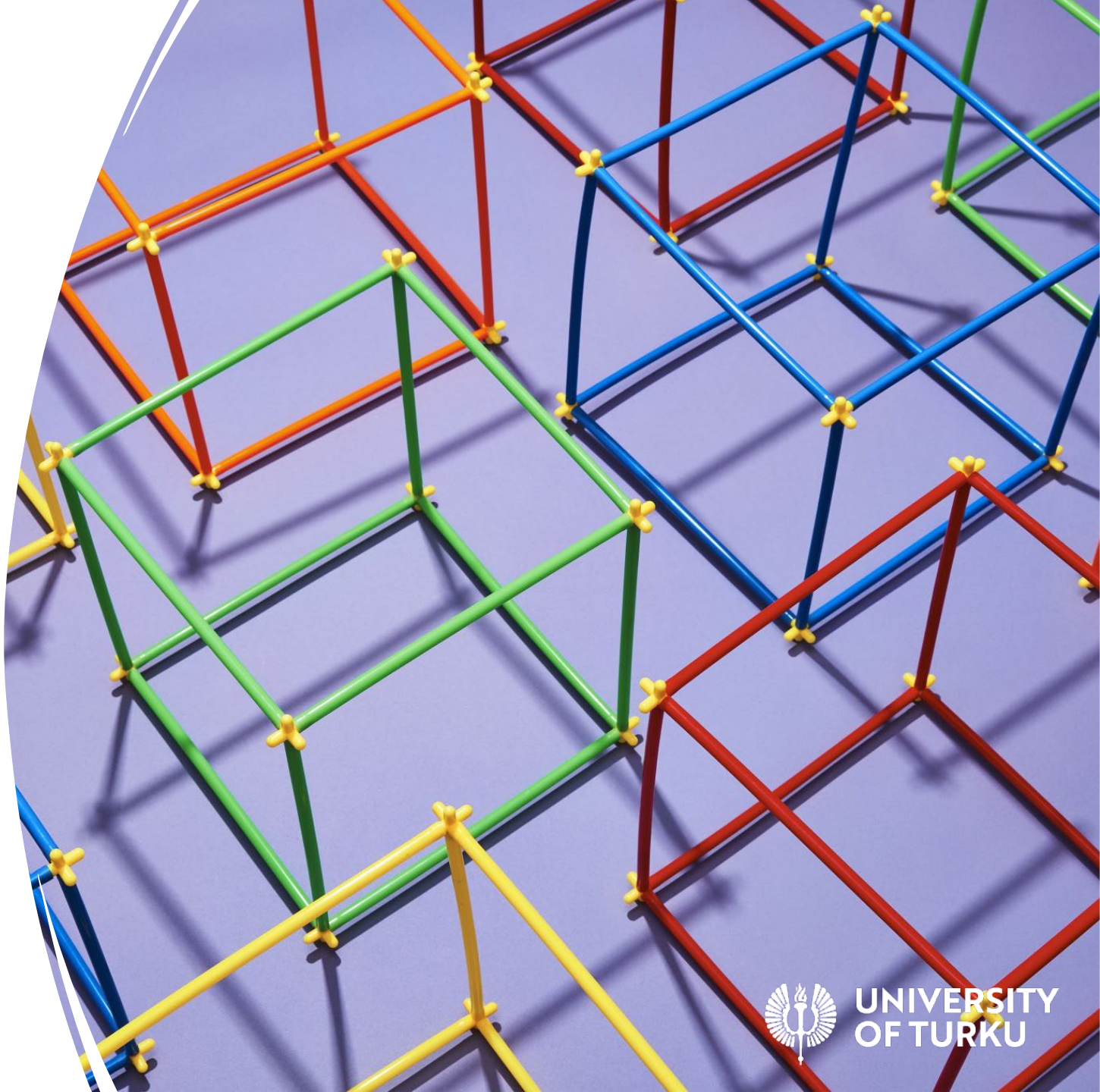
- Modelling Concepts
- Encoders, Decoders and Memories
- Behavioural and Timing Constraints
- Latch and FlipFlop
- Registers and Counters
- Finite State Machines
- Functions and Procedures



Exercises

- All exercises are obligatory
  - Totally 5 (as discussed during the lecture, can be 4-6 exercises)

- Always remember to **comment** your code
  - Insert comment blocks on files, classes and class members (variables and member functions)
  - Comment should descripe the **intent** of the code



Exercises

- In the exercises, we will use Xilinx's tools and FPGA development boards. Therefore, **download Vivado ML Standard Edition (free)**

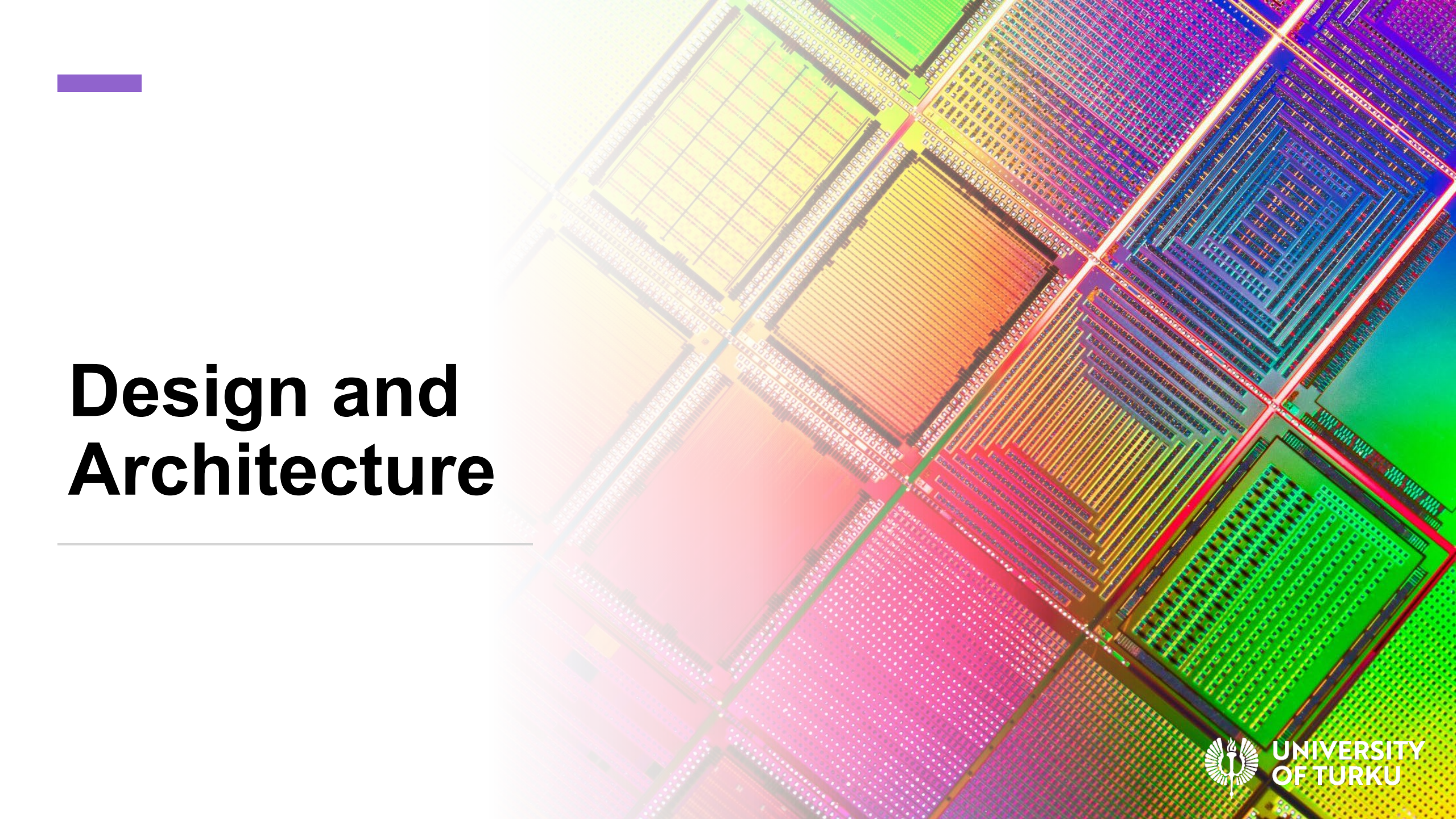
- https://www.xilinx.com/products/design-tools/vivado/vivado-ml.html

Passing the course

- Do ALL the exercises

UNIVERSITY OF TURKU

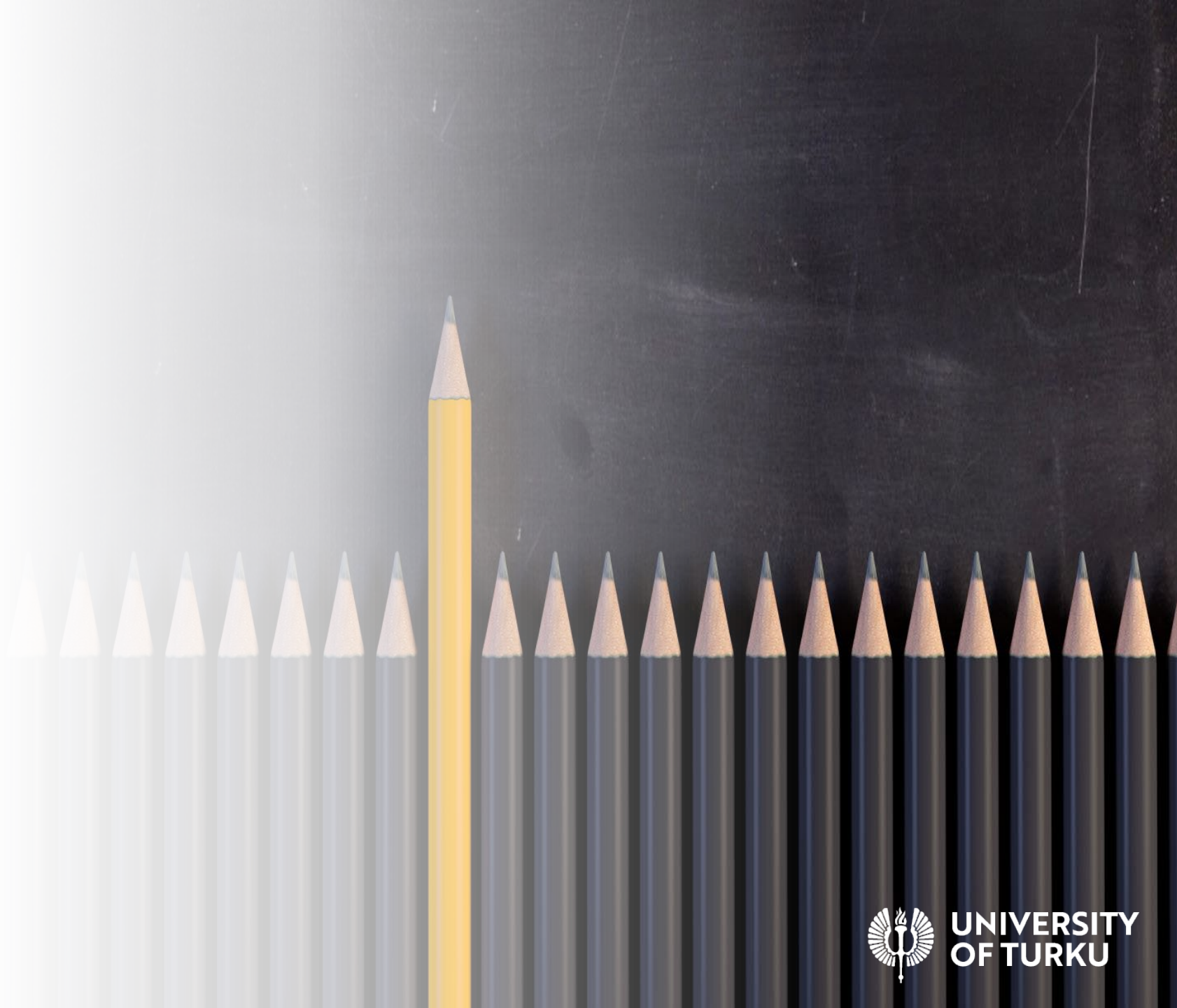
Design and Architecture



UNIVERSITY OF TURKU

Quality

The quality of the code we write has a deep influence on the quality of the finished system



Design Qualities

- Simplicity
  - It does not come naturally; it must be designed
- Understandability
  - In every realistic situation, code is read much more often than it is written
  - The clever code we write today may be hard to understand tomorrow
- Modifiability
  - A highly modifiable system has two desirable characteristics: it enables localized changes, and it prevents ripple effects
- Testability
  - There is a strong correlation between ease of testing and good design

Design Qualities

- Extensibility
  - How easy it is to accommodate new functionality in a system
- Reusability
  - A measure of how easy it is to use the code in a place other than where it was designed for
- Portability
  - How easy it is to migrate a system to a different environment
  - To increase portability, do not use nonstandard language features
- Maintainability
  - Combined result of other qualities, such as modifiability, understandability, expensibility and portability

Perfomance

The best way to write code that helps achieve performance goals is to have a modular design and keep the code clean and modifiable

Efficiency

Efficiecny means to make good use of scarce recourses

- Energy-efficient
- Area-efficient

UNIVERSITY OF TURKU

What is Design?

- As a verb, **to design** means to create a plan for turning specifications into a working solution
- As a noun, **design** can mean either a set of such plans or the way in which something has been made
  - the sum of all design decisions made while creating the solution

- We need to do some design before coding because a good plan improves the odds of achieving our goals
  - The quality of the source code deeply influences the design qualities of the finished system

In Digital System Development …

- … you can have a role as a
- Design Engineer

- Verification Engineer

In Digital System Development …

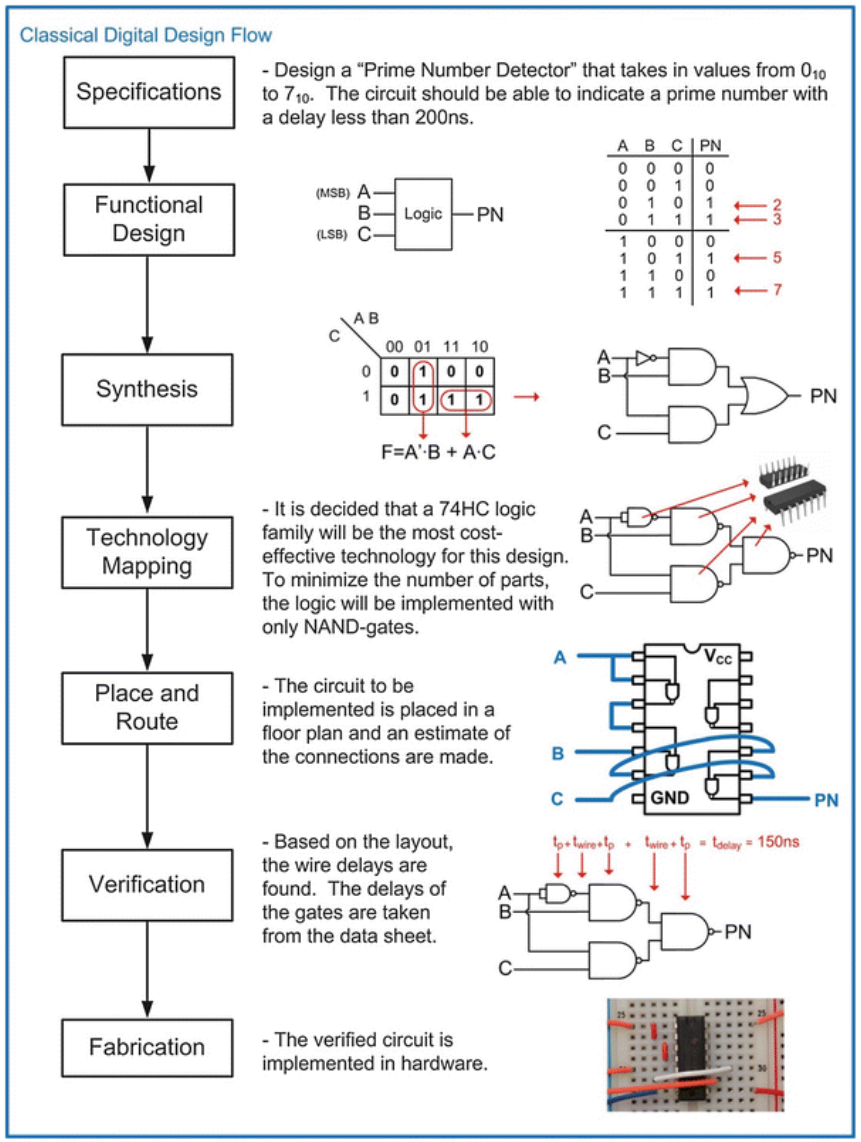
- … you can have a role as a
- Design Engineer
  - To create a device that performs a particular task based on a design specification

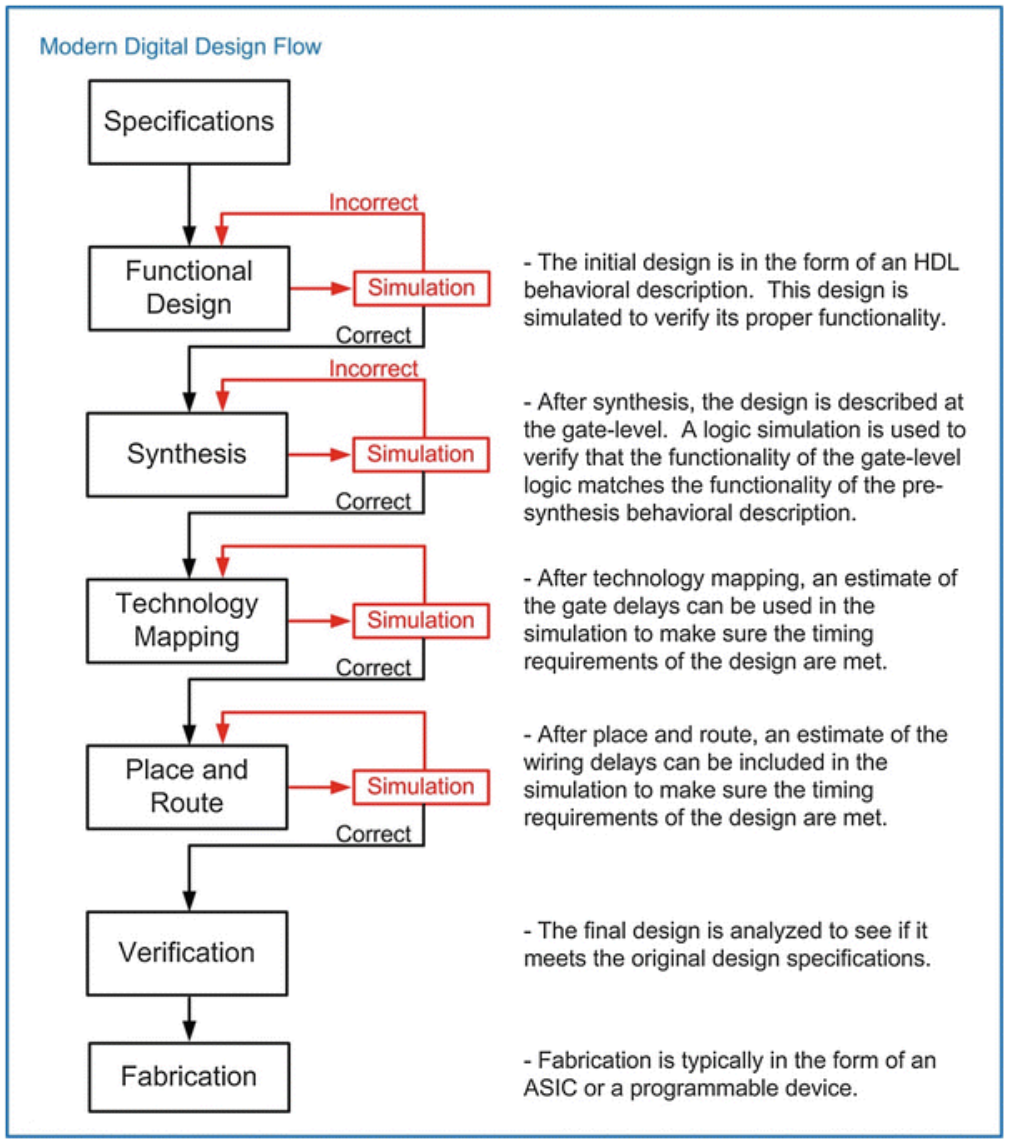
- Verification Engineer

In Digital System Development …

- … you can have a role as a
- Design Engineer
  - To create a device that performs a particular task based on a design specification
- Verification Engineer
  - To ensure that the model / design works correctly and succesfully based on a design specification

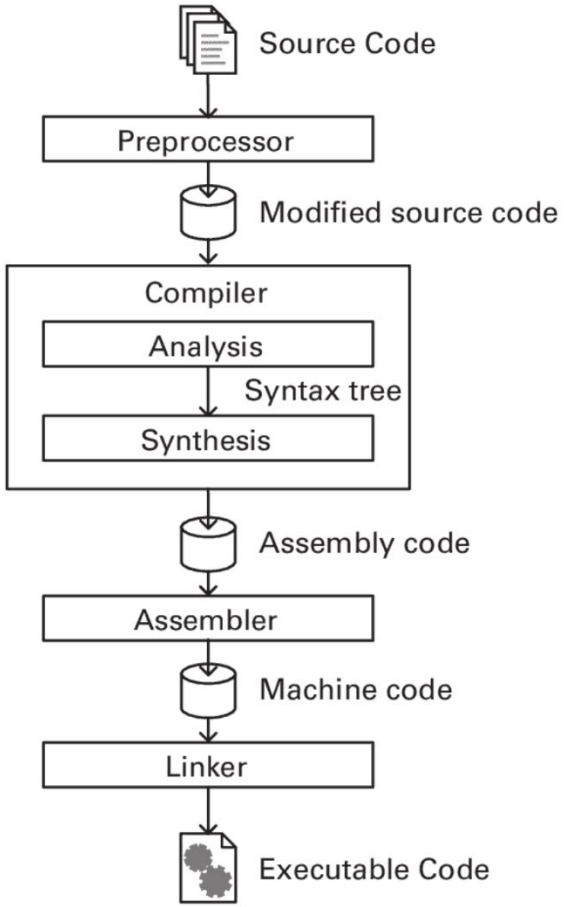
Digital Design Flow



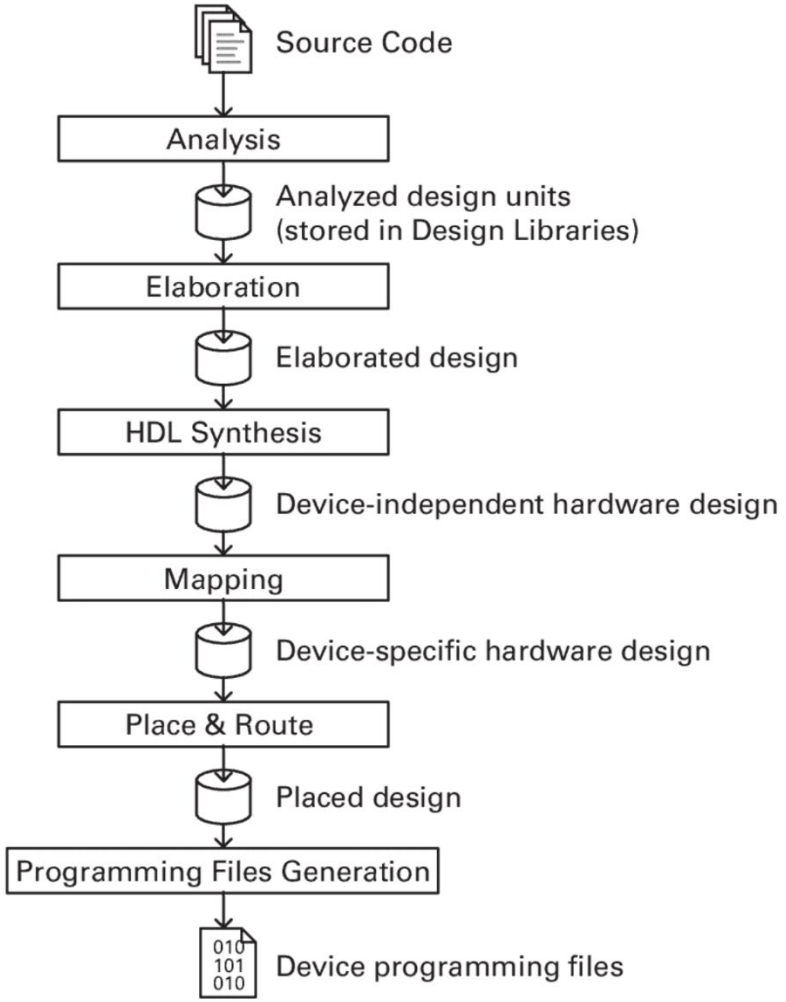


UNIVERSITY OF TURKU

Digital Design Flow



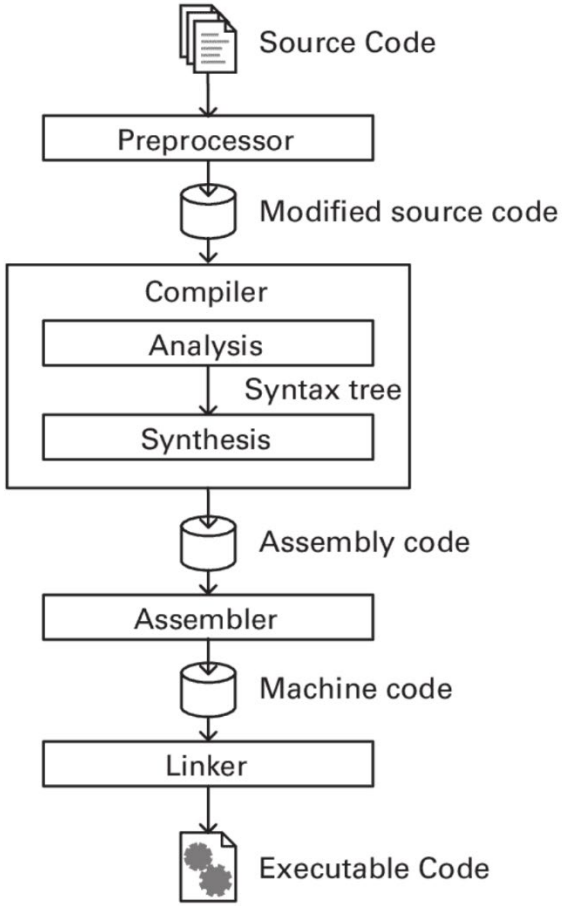
(a) Software case (programming language).



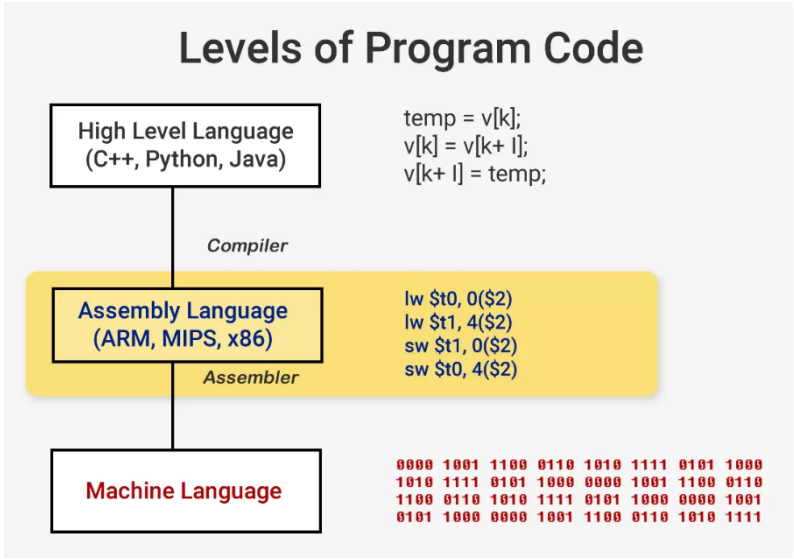
(b) Hardware case (hardware design language).

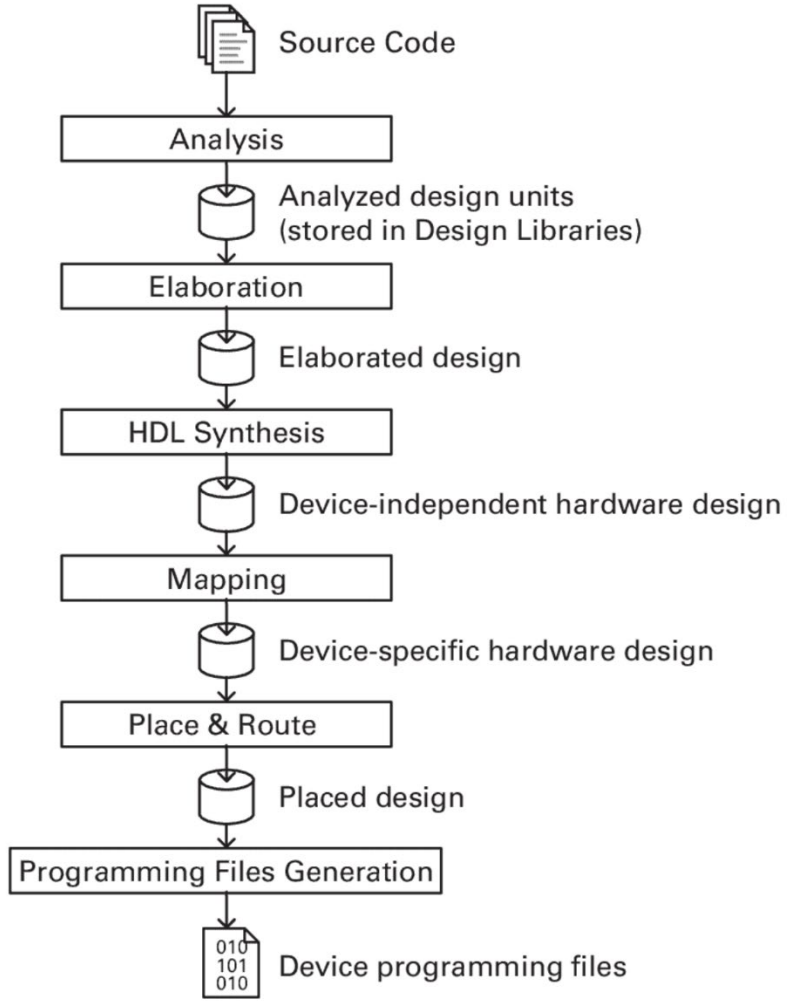
UNIVERSITY OF TURKU

Digital Design Flow



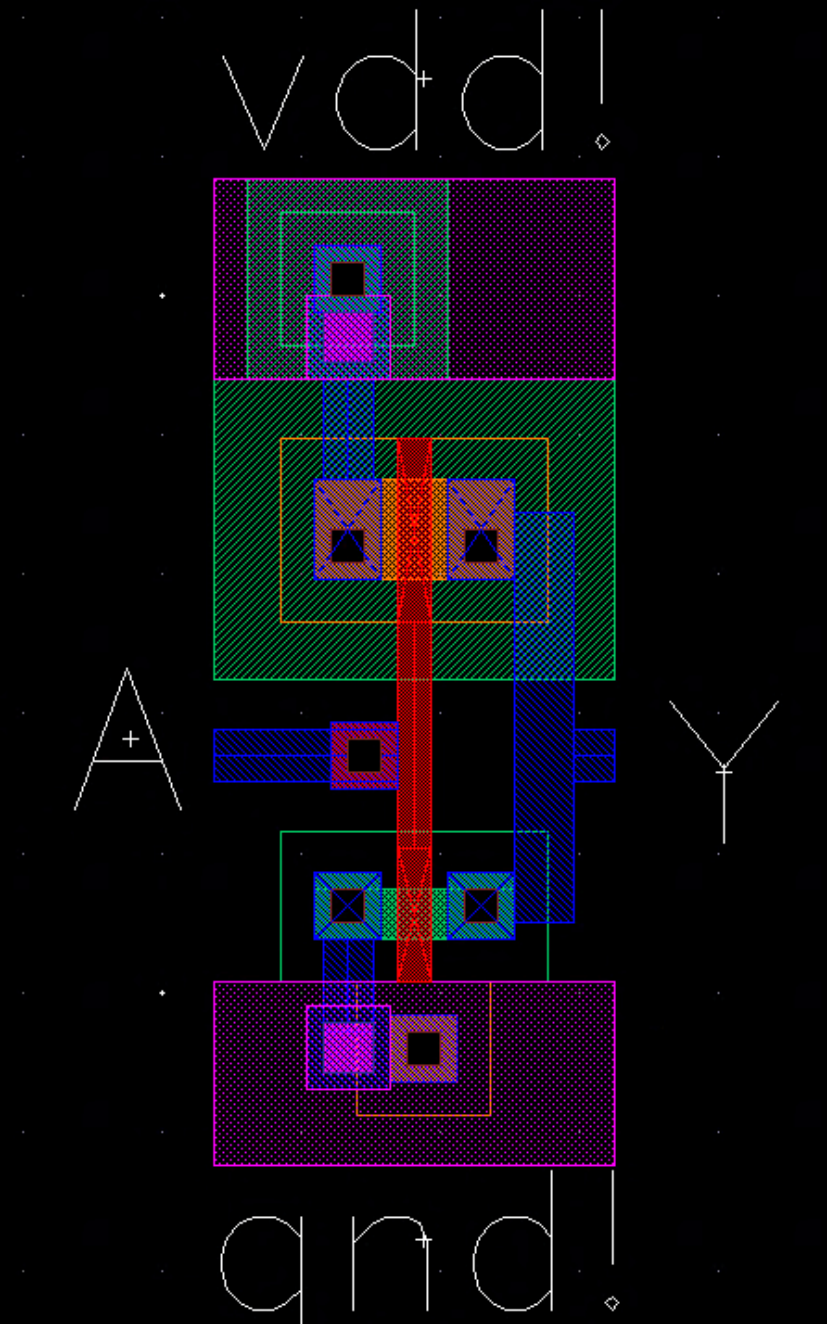
(a) Software case (programming language).



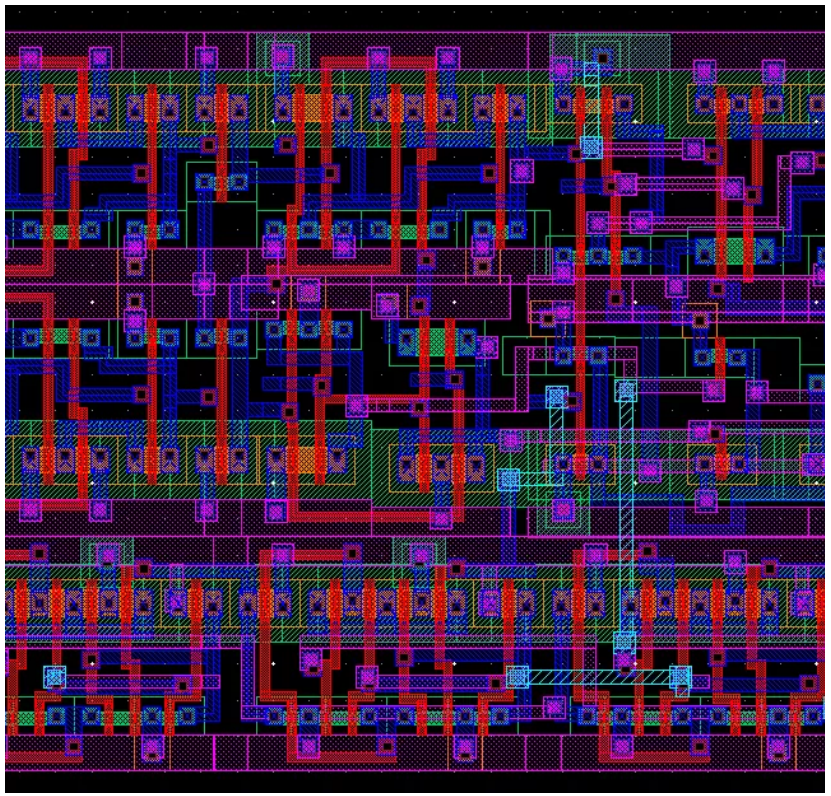


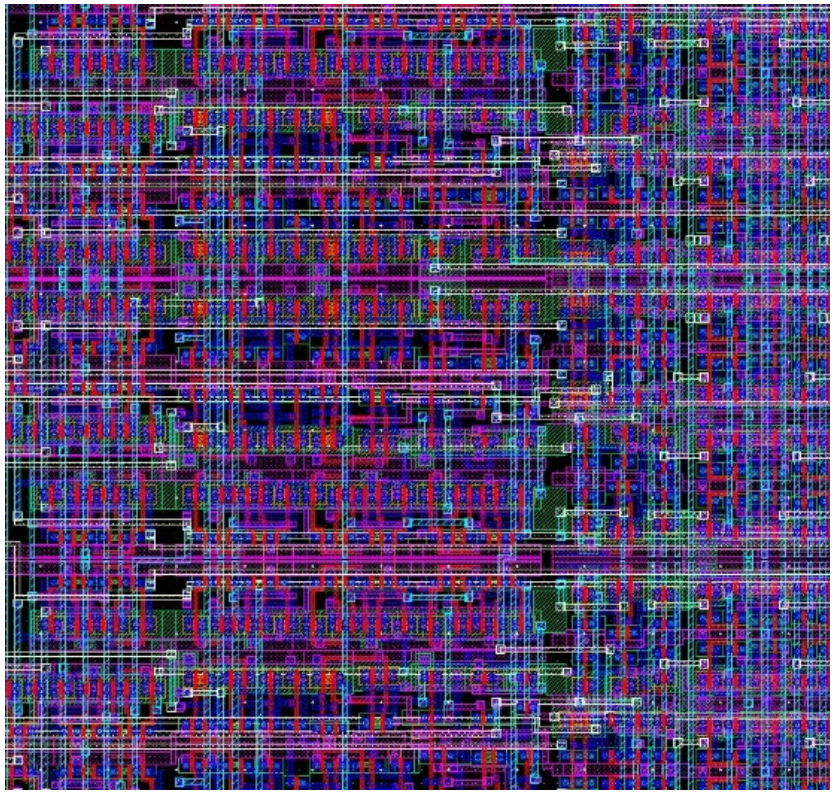
(b) Hardware case (hardware design language).

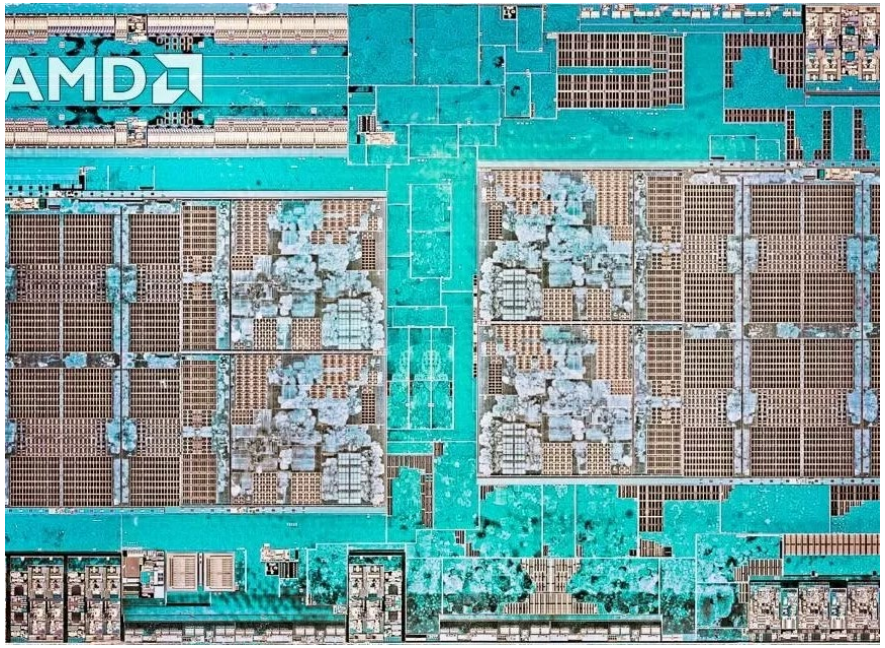
UNIVERSITY OF TURKU

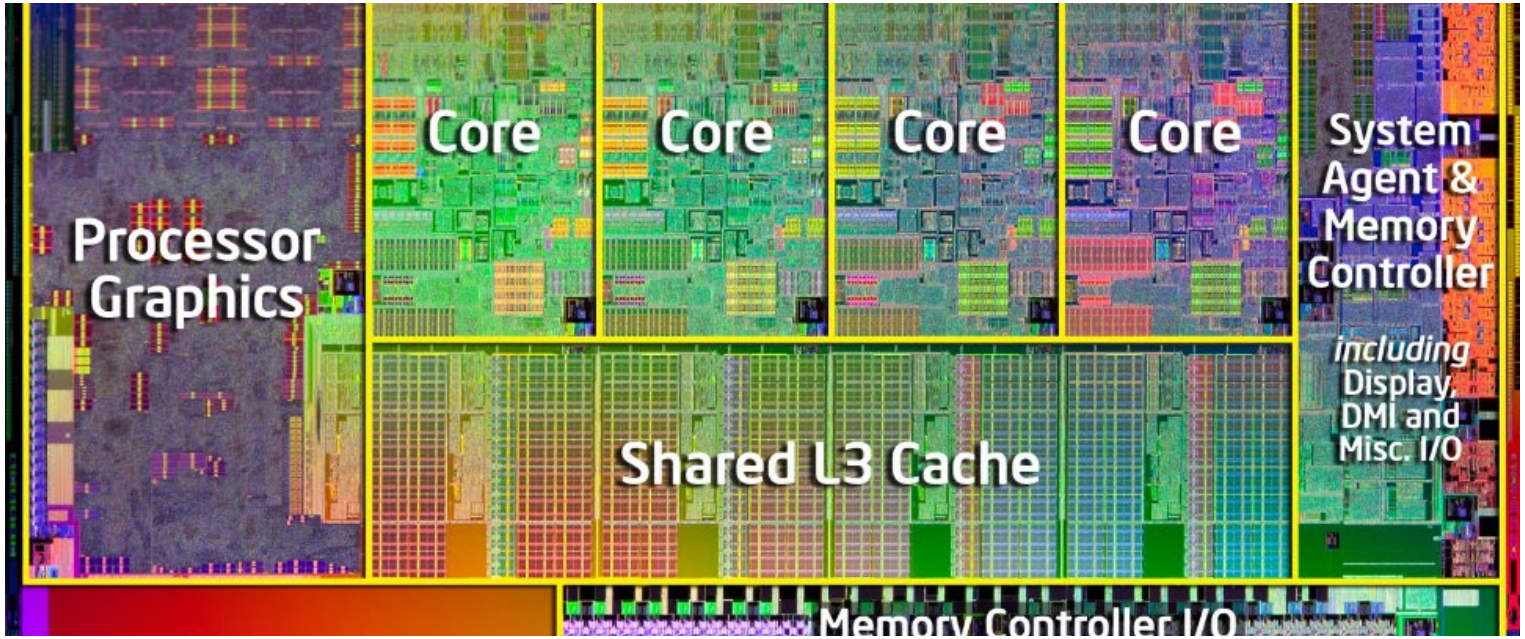


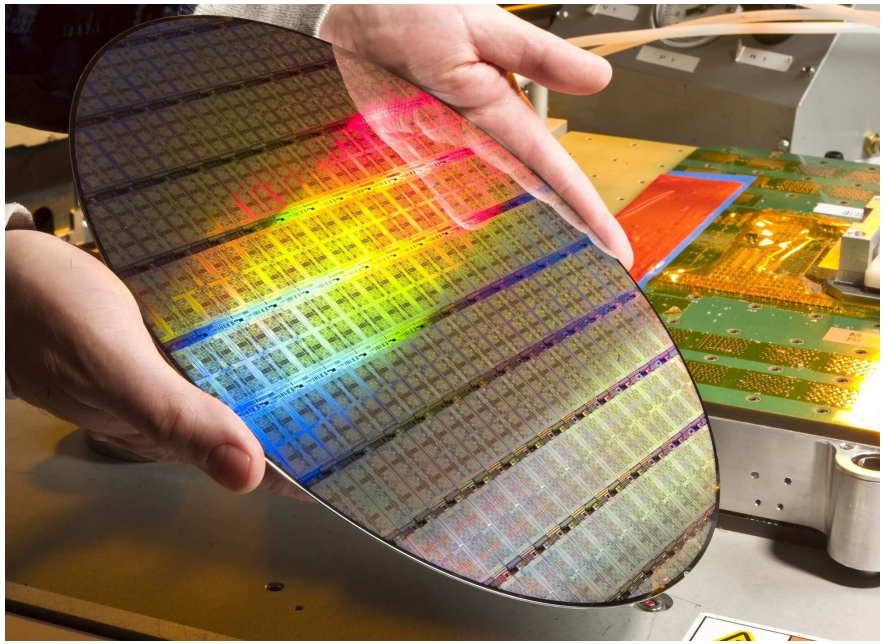
Technology Mapping / Place and Route







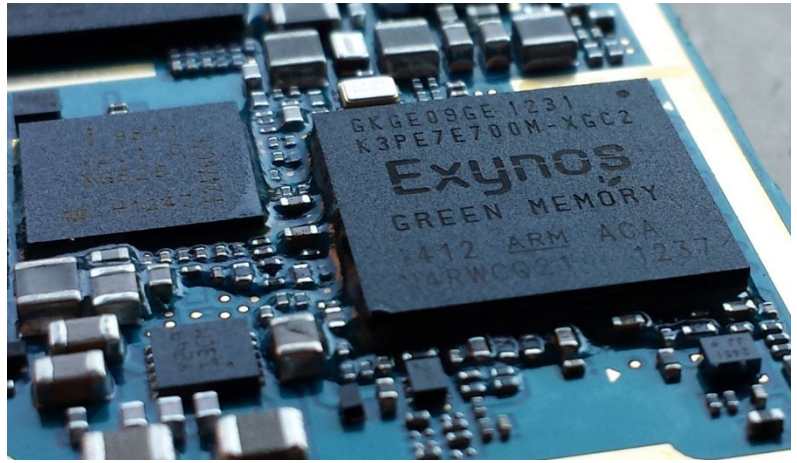


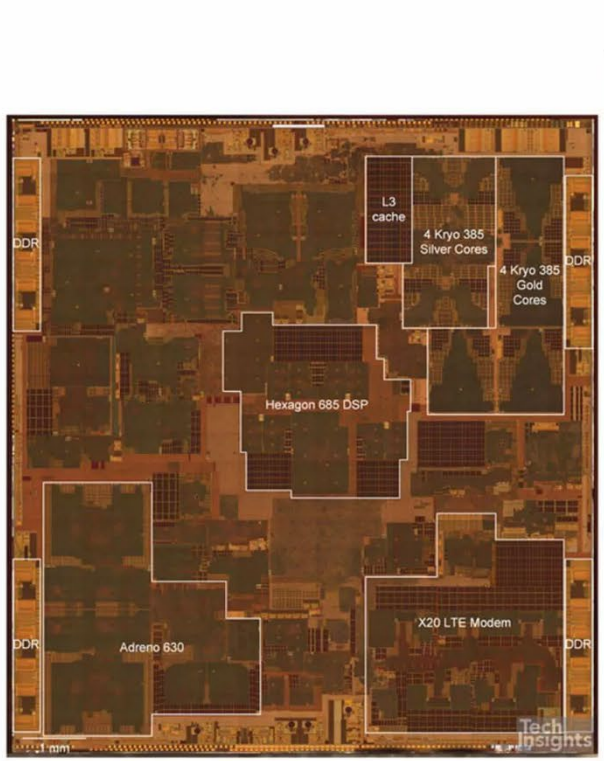


Fabrication

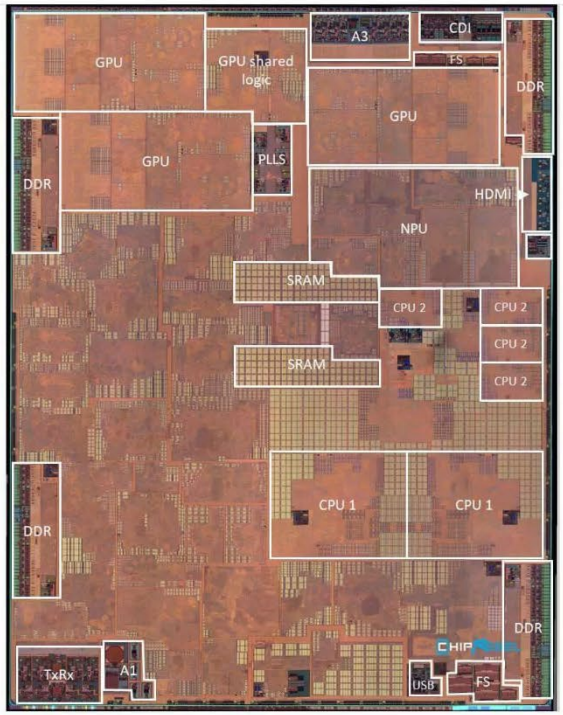
UNIVERSITY OF TURKU

Fabrication

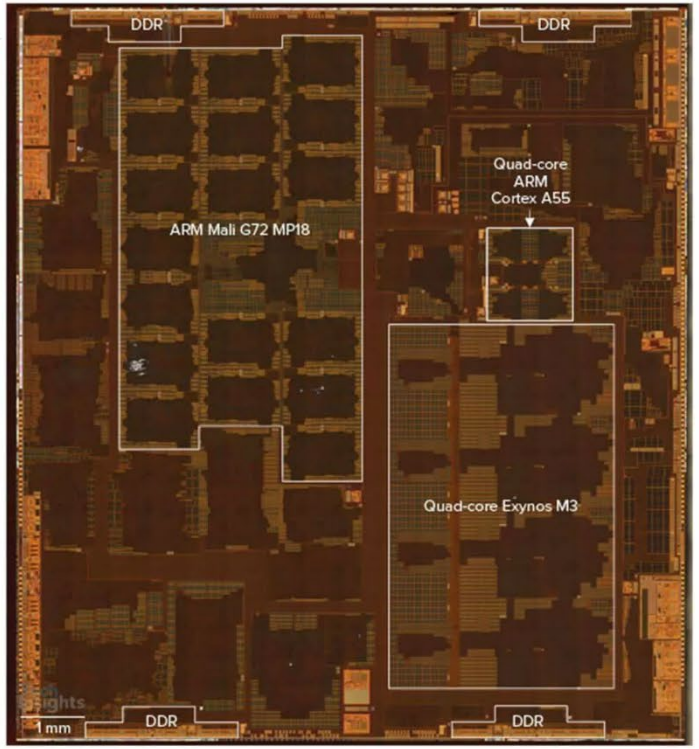




Snapdragon 845
(~90 mm²)



A11 Bionic
(87.66 mm²)



Exynos 9810
(118.94 mm²)

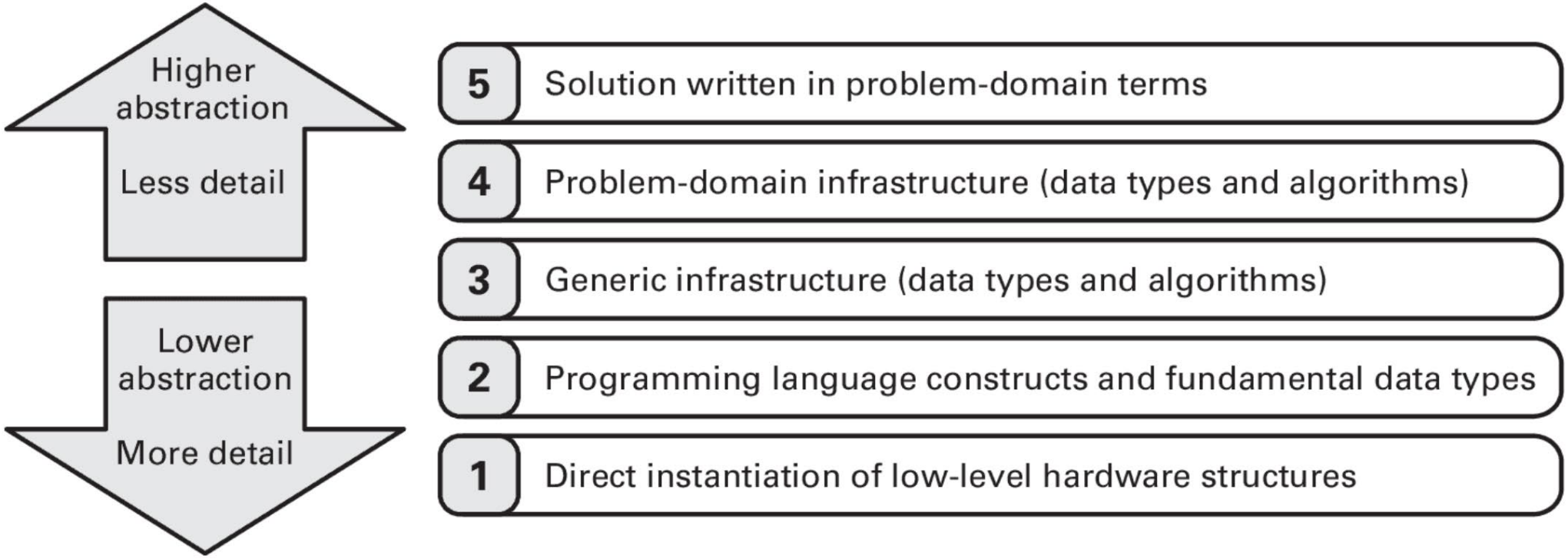
UNIVERSITY OF TURKU

Three Main Design Principles

- Abstraction
- Modularity
- Hierarchy

UNIVERSITY OF TURKU

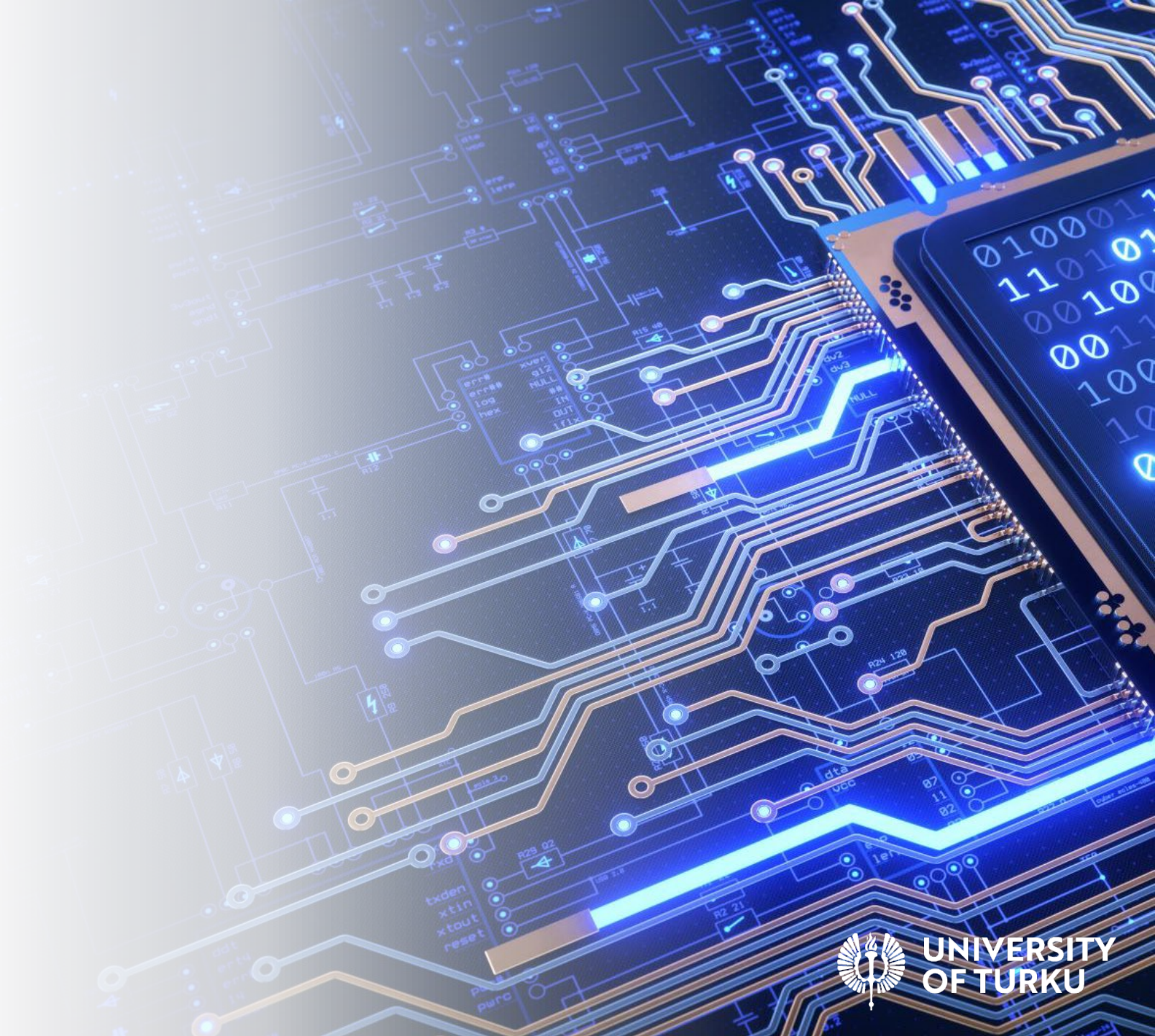
Layers of Abstraction



Levels of Abstraction

Level	Components	Representation
Behavioral	Algorithms, FSMs	Algorithmic representation of system functions
System (architectural)	Processors, memories, functional subsystems	Interconnection of major functional units
Register Transfer (dataflow)	Registers, adders, comparators, ALUs, counters, controllers, bus	Data movement and transformation and required sequential control
Gate (logic level)	Gates, flip-flops	Implementation of register level components using gates and flip-flops
Transistor	Transistors, resistors, capacitors	Implementation of gates and flip-flops using transistors, capacitors, and resistors

Modelling



Modelling

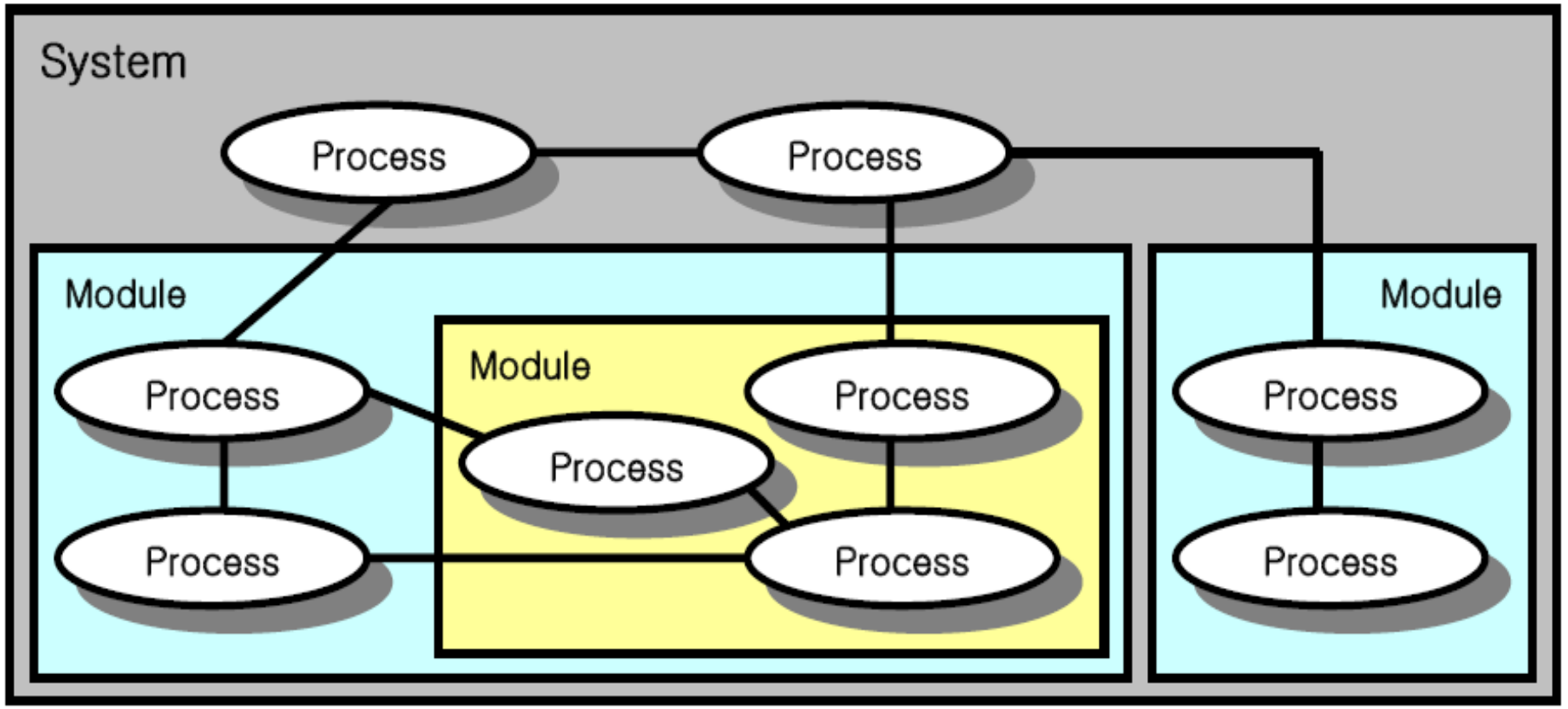
- A model helps us to understand the important aspects of a complex system *before* its implementation
- A model attempts to simulate an *abstract* model of a particular system
- The abstraction level of a model defines how much low-level details are included in the model

Can't see the forest for the tree

In Finnish: Nähdä metsä puilta

UNIVERSITY OF TURKU

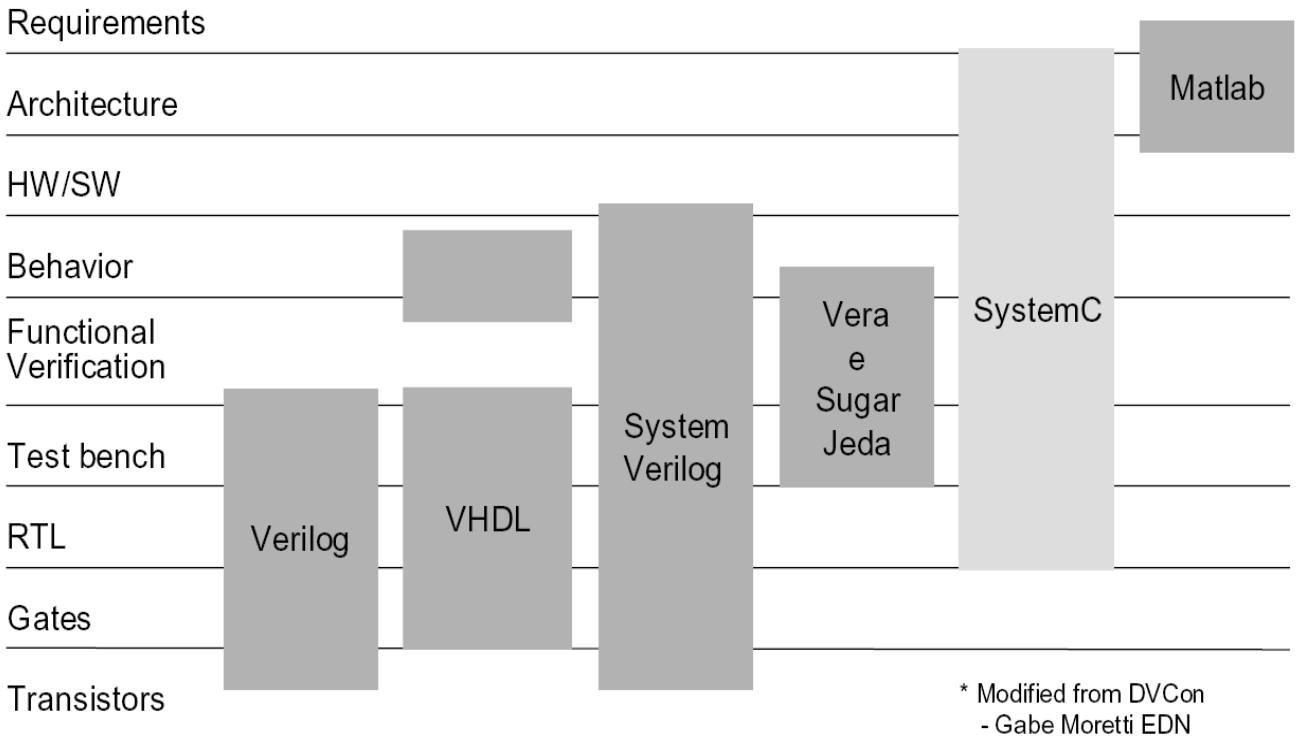
What is the forest?



UNIVERSITY OF TURKU

Language Comparison

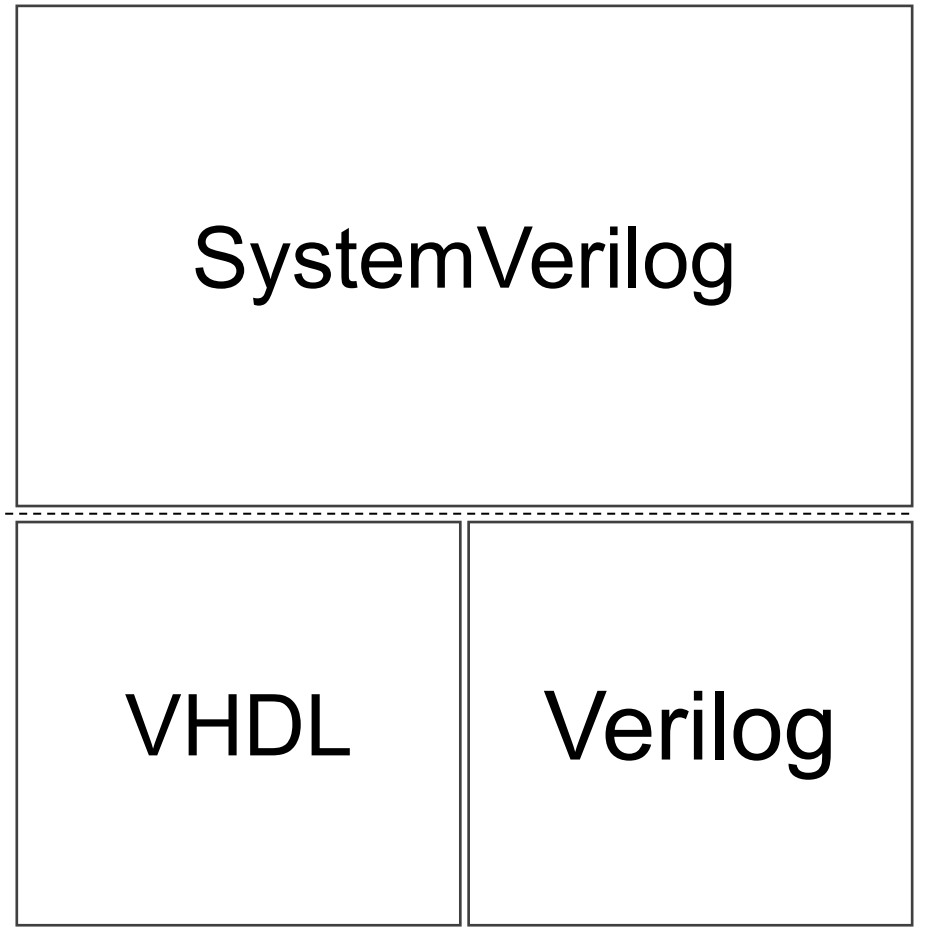
- What kind of modelling languages there are?



UNIVERSITY OF TURKU

Language Comparison

- Verification
  - Object Oriented
  - Randomisation
  - Verification support
  - Verification methodology
- RTL
  - Multi-threaded
  - Hardware data types
  - Netlisting



Scope of VHDL

- VHDL is designed for the specification, design and description of digital hardware systems

- VHDL stands for VHSIC HDL

Very High Speed Integrated Circuits
Hardware Description Language

- https://ieeexplore.ieee.org/document/5981354/

History of VHDL

1981	Initiated by US DoD to address hardware life-cycle crisis
1983-85	Development of baseline language by Intermetrics, IBM and TI
1986	All rights transferred to IEEE
1987	Publication of IEEE Standard
1987	Mil Std 454 requires comprehensive VHDL descriptions to be delivered with ASICs
1994	Revised standard (named VHDL 1076-1993)
2000	Revised standard (named VHDL 1076 2000, Edition)
2002	Revised standard (named VHDL 1076-2002)
2007	VHDL Procedural Language Application Interface standard (VHDL 1076c-2007)
2009	Revised Standard (named VHDL 1076-2008)

IEEE 1076-2019
(approved 5.9.2019)

Some Key Features of VHDL

- VHDL can be used throughout the entire design cycle
- VHDL supports different levels of abstraction
- VHDL inherently supports parallelism
- VHDL also supports sequential programming
- VHDL supports different modeling styles

- **VHDL is continuously (but slowly) evolving**
  - **newest standard from 2019**

Some Issues of VHDL

- VHDL is hard to master
  - A good knowledge of the language requires understanding its execution model and concepts that are not apparent in the code
- VHDL still lacks higher level programming feature
  - NO object-oriented features
- VHDL code is still verbose


UNIVERSITY
OF TURKU