Budapest University of Technology and Economics
Department of Electron Devices

Lec. 1
Introduction to system-level design

Osama Ali

osamaalisalman.khafajy@edu.bme.hu

2 of May 2022

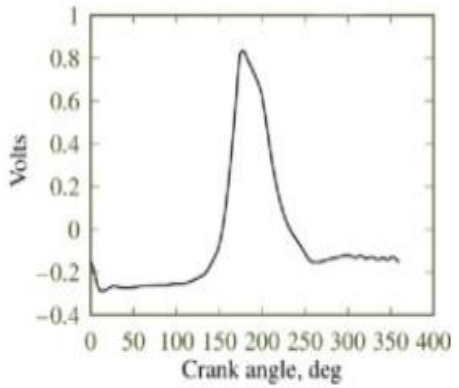
Outline

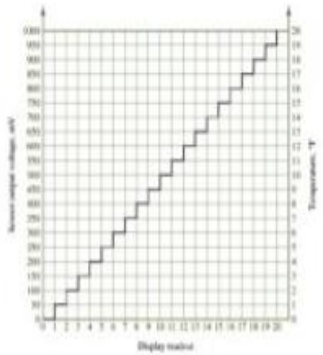
- Analog Vs. Digital
- Synchronous and asynchronous
- What is a System
- Specification and Implementation of digital system
- System Level design
- Managing a complex design
- Abstraction Levels
- Design flow of digital systems
- Small intro about VHDL

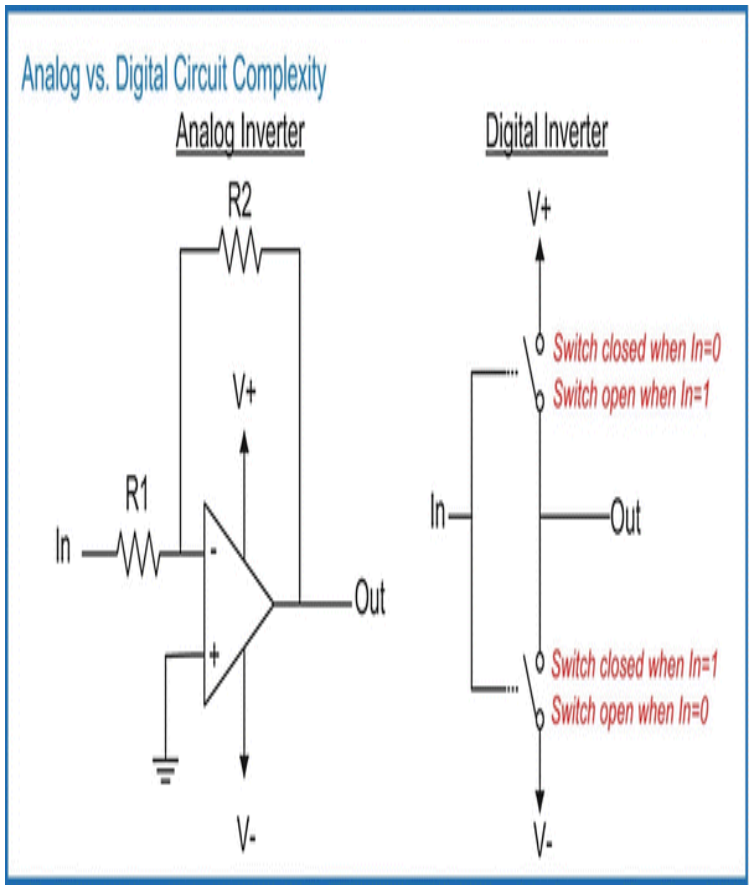
Analog Computer Vs Digital Computer

Analog Circuits vs. Digital Circuits

- An **analog** signal is an electric signal whose value varies continuously over time
- A **digital** signal can take on only finite values as the input varies over time



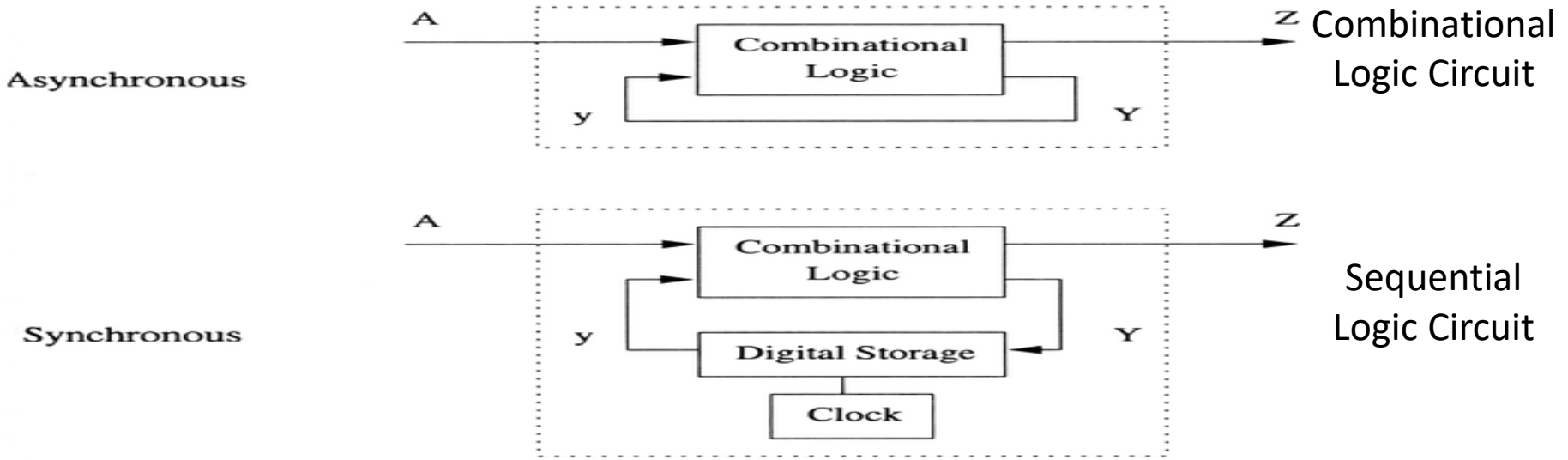




Analog Circuits	Digital Circuits
Analog circuits operate on **continuously variable** signals also known as Analog Signals.	Digital Circuits operate on **discretely variable** signals or Digital signals i.e. the signal exists only in two levels: 0 and 1 (binary digital signaling).
Depending the efficiency and precision, it is quite difficult to design Analog Circuits.	Digital Circuits are relatively easy to design with many automated tools available for various stages of design and analysis.
When interacting with the physical world, analog circuits can **directly accept the signals from outside** as the data is already analog.	If a digital circuit has to acquire data from physical world, the analog signals must be converted to digital signals first.
As there is no need for data conversion, there is ideally **no loss of information**.	During the process of converting analog signals to digital signals, **there might a significant amount of data loss**, which can result in loss of information.
If **precision and accuracy are not a criterion**, then analog circuits can be simple and inexpensive.	Even with simple design techniques and at low cost, the digital circuits can **provide good accuracy and precision**.
Due to the lack of skilled engineers and the complexity of the designs, **analog circuits can turnout to be quite expensive**.	Advanced Integrated Circuits technologies and many other factors help the digital circuits to be reliable, lower in cost and smaller in size.

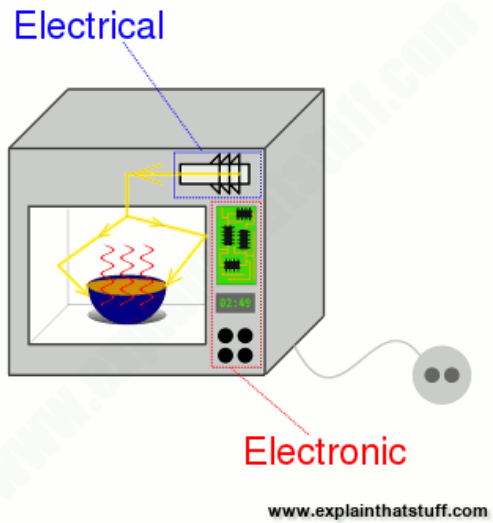
(Synchronous and asynchronous)

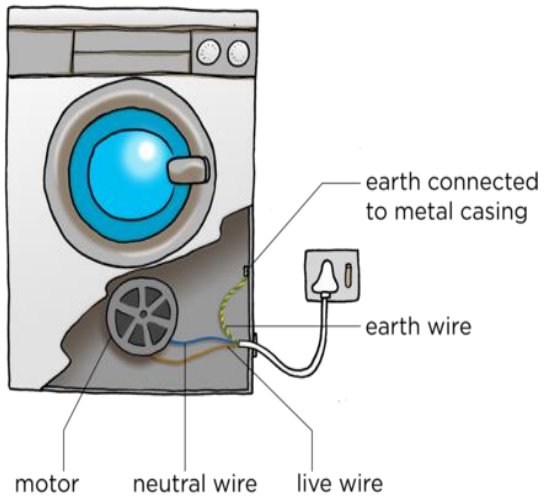
- Sequential logic is a type of logic circuit whose output depends not only on the present value of its input signals but on the sequence of past inputs, the input history as well.
- On the other hand, the combinational logic, whose output is a function of only the present input.
- That is, sequential logic has state (memory) while combinational logic does not.



What is a System

- A system is a group of **interacting** or **interrelated** elements that perform according to a set of **rules**.
- A system, **surrounded** and **influenced** by its **environment**, is described by its **boundaries**, **structure** and **purpose** and expressed in its functioning





Digital system specification and Implementation

- **Specification** of a system is the description of its **function** and other **characteristics required** for it, for example **speed**, **cost** and **power**. On other hand you need to define **input** and **output port** and **signals**.

- **Implementation** means how the system is constructed from **smaller** and **simpler components** called **modules**. The modules can vary from simple **Gates** to complex **processors**

System Level design

- In same cases a system can be a **Heterogenous/Mixed system** (sub-systems.
- It could be Multi-Domain or Multi Physics System like System that contains **analog** and **digital** components and **non-electrical** components
- Domain/ Areas
  - electrical - digital and analog
  - magnetic - also related to electrical domain
  - mechanical rotational
  - mechanical translational
  - hydraulic/fluidic
  - radiation/optical
  - thermal

Cont.

- System level design is a **methodology** where the designer (engineer) accounts for all the components of a system.
- Traditionally, you may have a specialized designer for each domain of the system each one of them know little about others domain.
- This can lead to **long** and **frustrating** design iterations to reach a successful design
- The solution for that a tool not only provides multiple levels of abstraction but also it have the ability for integration between hardware and software that will be a very good system-level design tool.

Cont.

- This tool must have ability to express
  - mathematical equations
  - execute them in a real-time

Operation:	OR (logical sum)	AND (logical product)	NOT (negation)
Algebraic symbols:	$X + Y$ $X \vee Y$ $X \cup Y$ X or Y	$X \cdot Y = XY$ $X \wedge Y$ $X \cap Y$ X and Y	$\overline{X}$ $!X$ $-X$ not(X)

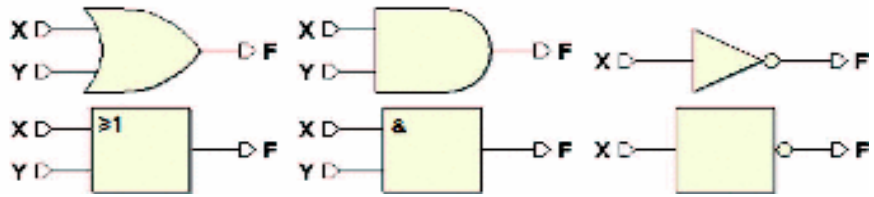
Truth table:

X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

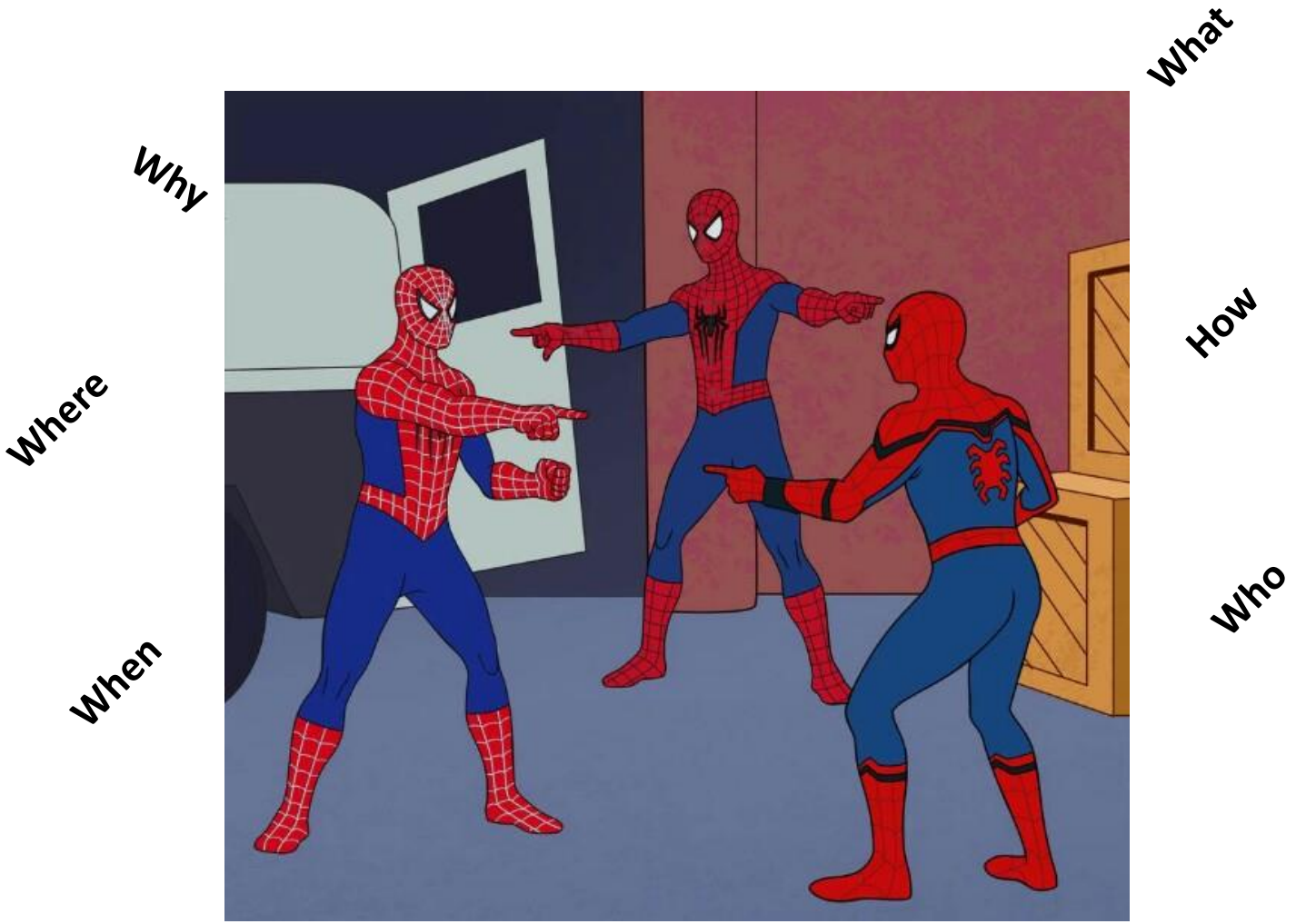
X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	$\overline{X}$
0	1
1	0

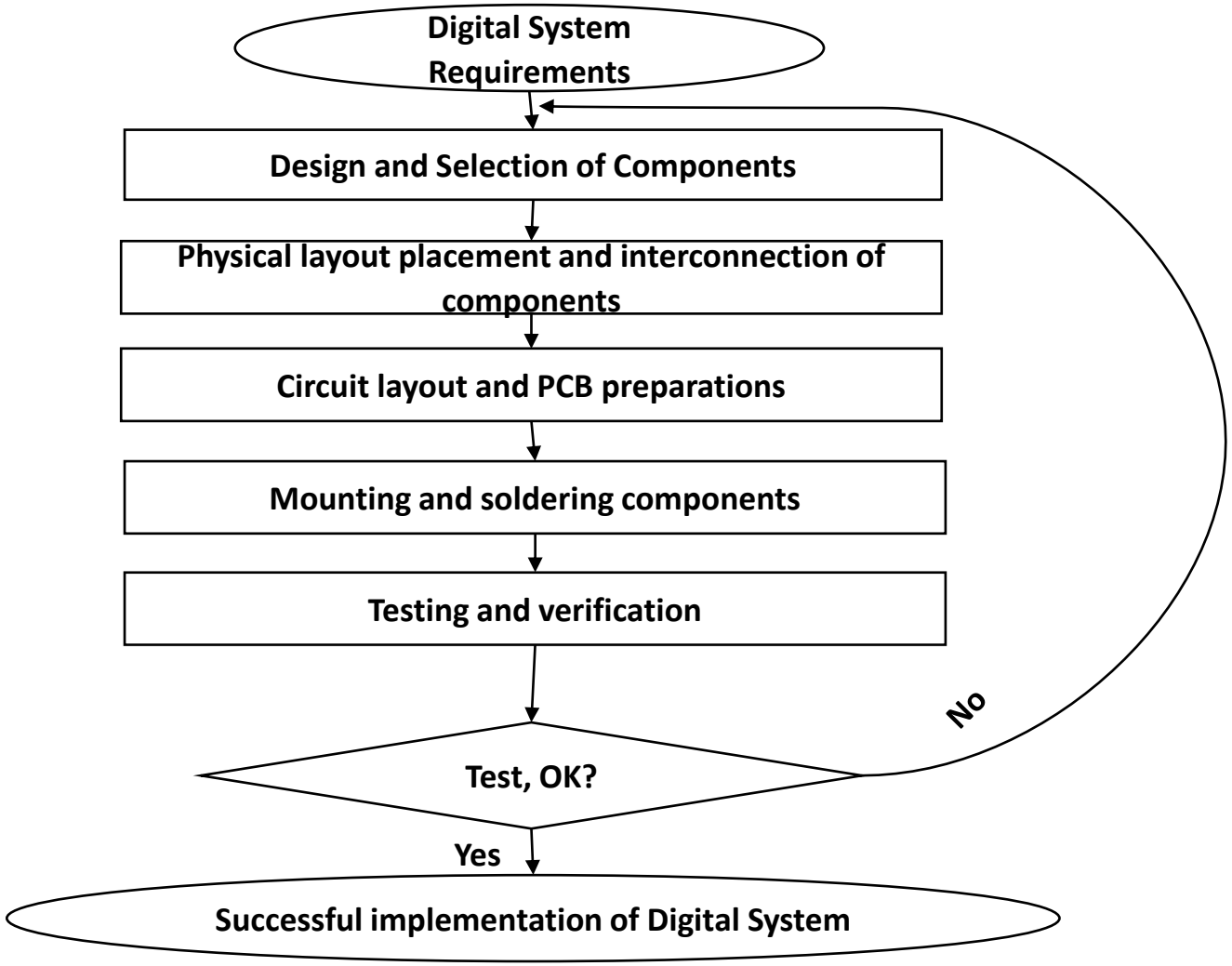
Circuit diagram symbols:



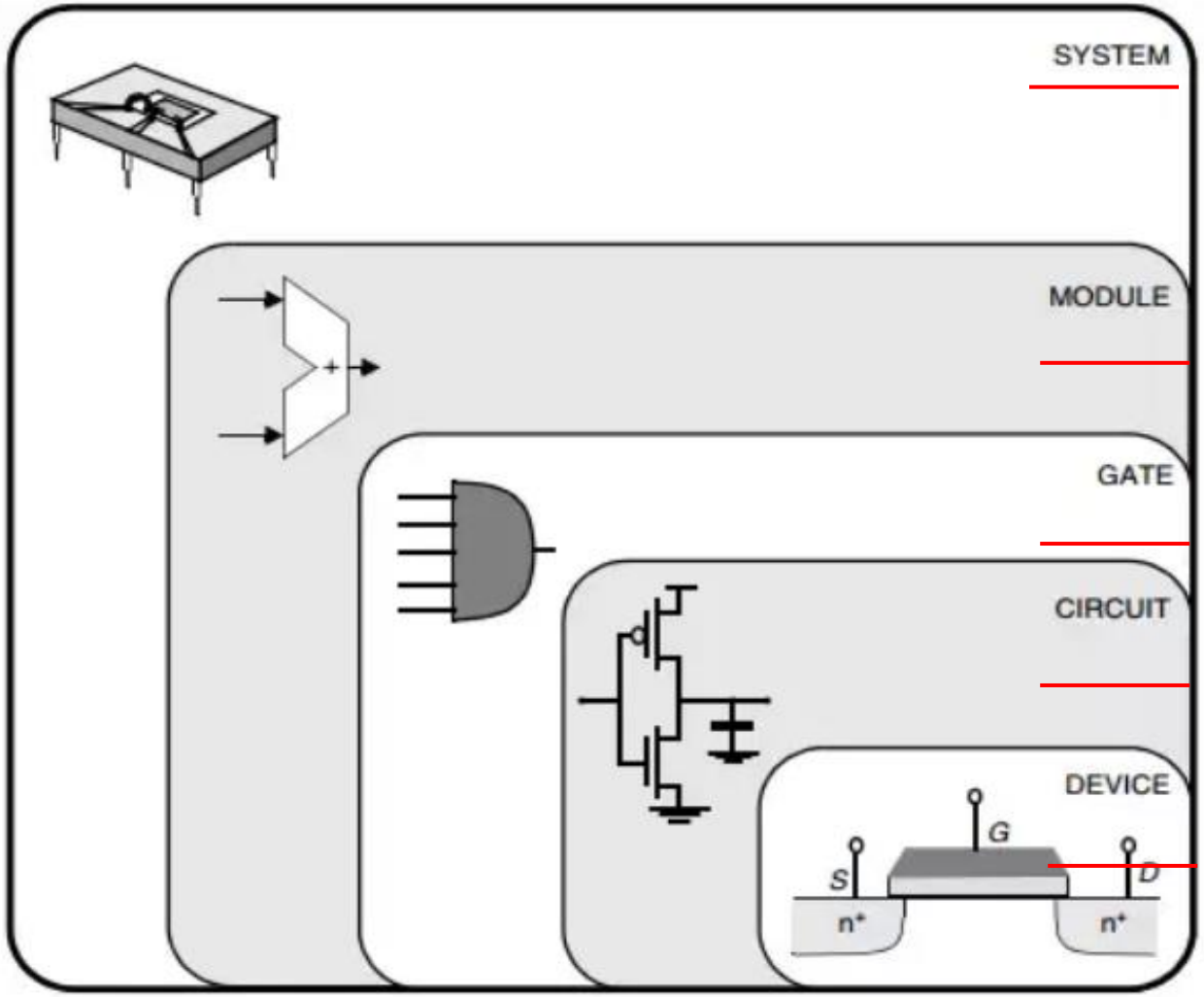
Managing a complex design



Sequence of steps in conventional digital design

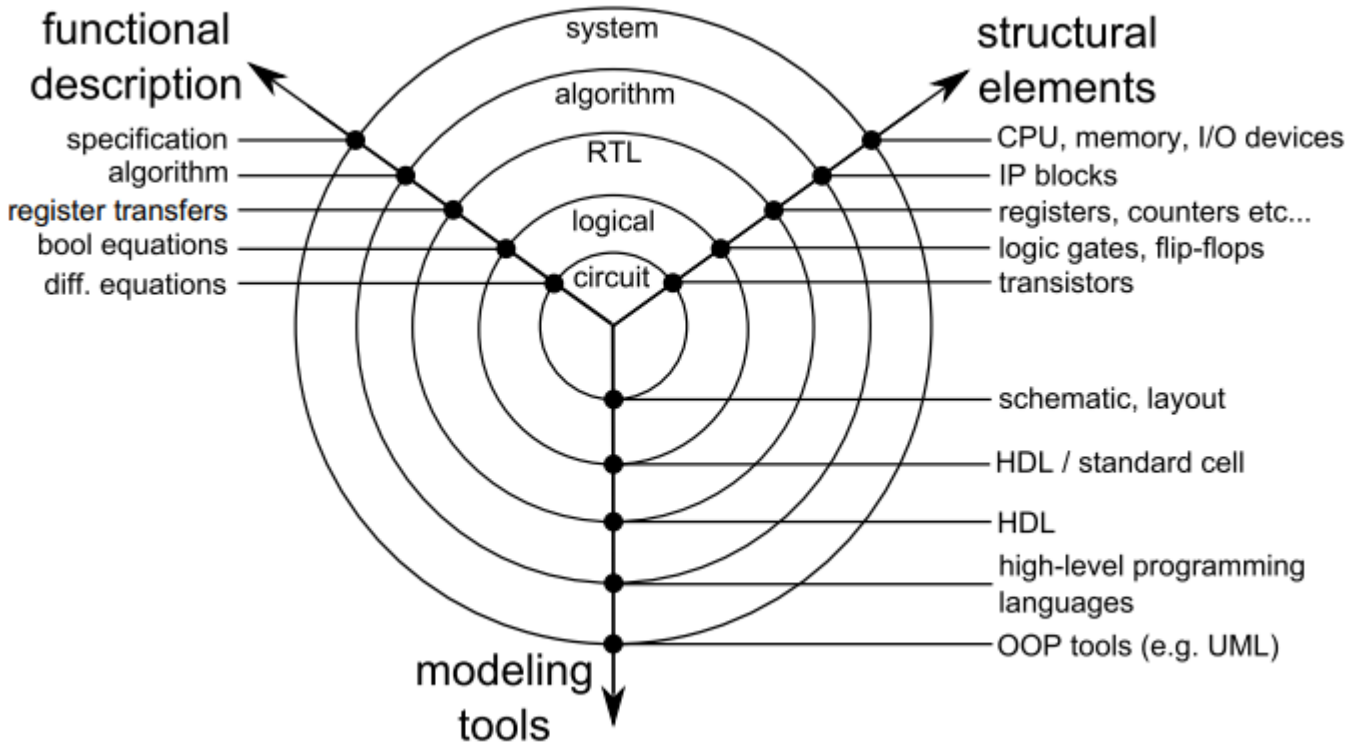


Abstraction Levels in Digital System



Abstraction Levels\Gajski-Kuhn Y-diagram

- The GK diagram is an expressive representation of the abstraction levels.



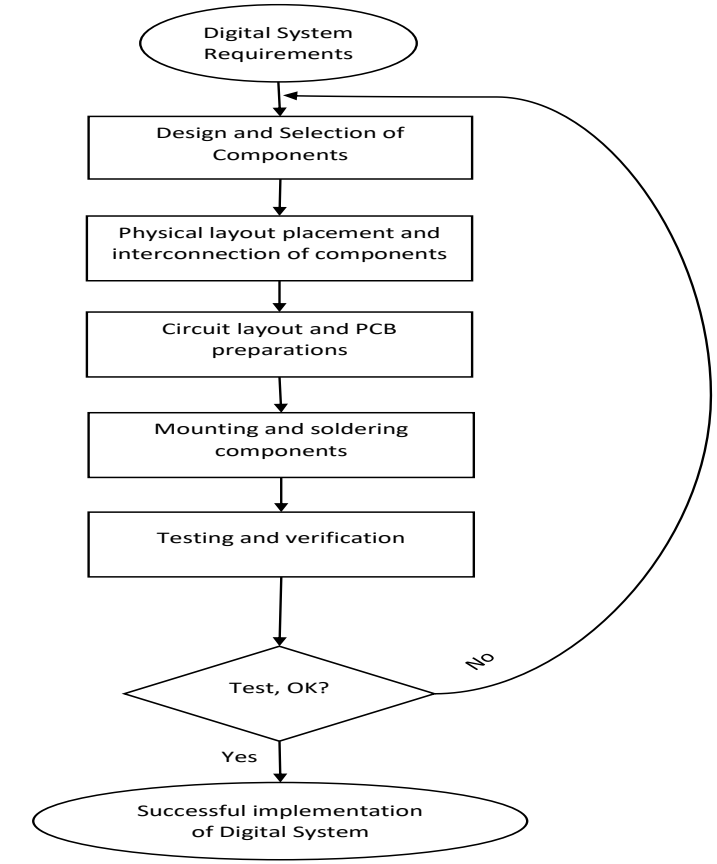
Ref: Dr. Péter Horváth

Different abstraction levels

- ***System*** – defining design **partitions** and their **interfaces.** In the case of VLSI design, these can be description languages like VHDL, Verilog or SystemC.
- ***Algorithm*** – **behavioral** modeling with high-level programming languages. The tool that could be used in here C\C++.
- ***RTL*** *(register-transfer level)* – defining "**microarchitecture**“. So, it capture the functionality in the ships of register-transfer operations on ALUs, registers, multiplexers
- ***Gate*** – defining the behavior of **RTL** components with **Boolean Equations**
- ***Circuit*** – implementing the behavior of the logic gates with **transistor-based structures**

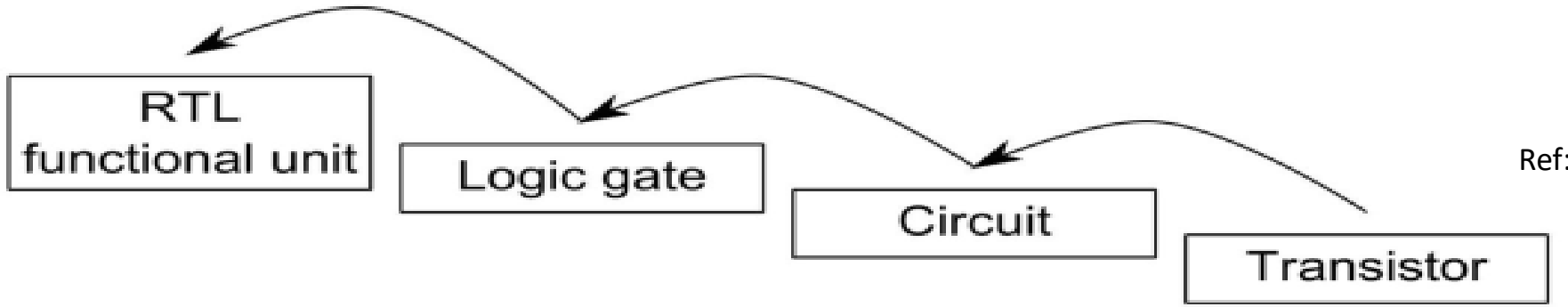
Design flow of digital systems

- Not the conventional one
- Mainly there are two approaches that can be taken out in designing a digital system
  - Bottom-up design method
  - Top-down design method



Bottom-up design method

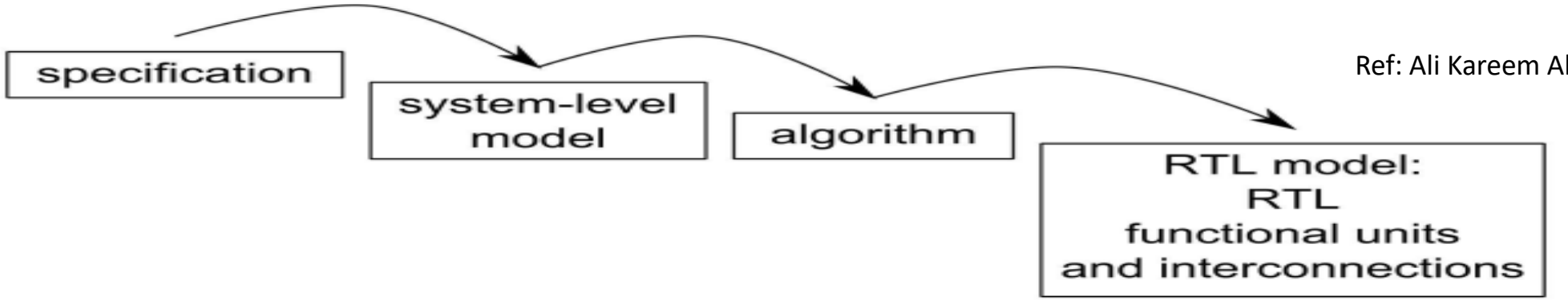
- In the bottom-up approach the designer creates basic functional units with very simple tasks. Once a sufficient set of elementary functionalities is constructed, a more complex model can be prepared with the combination of the simple ones. The design process stops when the increasingly complex model is able to implement the desired functionality defined in the specification.



Ref: Ali Kareem Abdulrazzaq

Top-down design method

- The design process starts with a high-level representation of the system.
- The high-level model includes partitions (subsystems) with a specific task.
- During the design process the implementations of the subsystems are elaborated; they are split into components with more specific sub-tasks and more detailed implementations.
- The process stops when the components of the refined design are simple enough to substitute them with an existing model (practically with an RTL functional unit).



Ref: Ali Kareem Abdulrazzaq

Small introduction in the descriptive language (VHDL)

Introduction to VHDL

Languages for designing hardware

- Higher-level computer languages are used to describe algorithms
  - Sequential execution
- Hardware Description Languages (HDL) are used to describe hardware
  - Not for programming, but for designing hardware
  - Most popular: VHDL, Verilog
  - Parallel (concurrent) execution
    - **Instructions are all executed at the same time**

Abstraction levels in VHDL

- VHDL is rich in language abstractions, in addition to which the language can be used to describe different abstraction levels
  - Dataflow
  - Structural
  - Behavioural

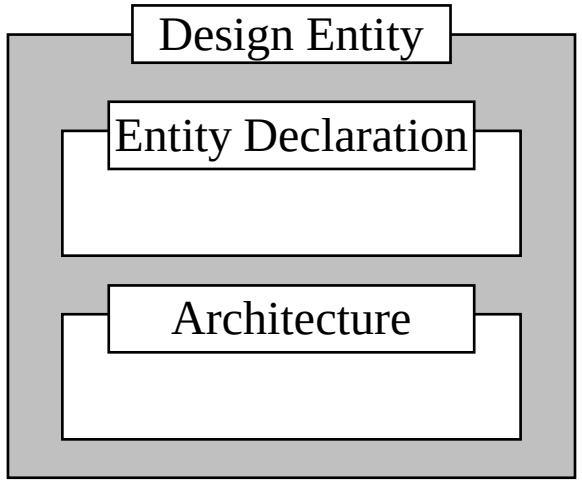
- Abstraction levels are means of concealing/hidden details
- The design of VHDL components described on higher abstraction levels can be **technology-independent**
  - It is usually a requirement to determine the abstraction level at which the information is to be described
  - If a short development time is required, a high abstraction level should be chosen as the way of modelling

- **Dataflow** : The dataflow view describes a network of signals in which the flow of signal values is supervised by a set of control elements

- **Structural** : Structural design is the closest to schematic capture and utilizes simple building blocks to compose logic functions

- **Behavioural** : It describes hardware behaviour in terms of **circuits** and **signal**
  - It accurately models what happens on the inputs and outputs of the black box
    - No matter what is inside and how it works
    - Function is defined algorithmically with timing and node loading largely ignored

Primary language abstraction



- The primary abstraction is the design entity
- It is the basic unit of hardware description
- The design entity can represent a **cell, chip, board, or subsystem**

- Aspects of modelling a system: *interface* and *function*
- An entity declaration defines the interface between the entity and the environment outside of the design entity
- An entity can be linked to several architectures
  - E.g., one architecture may model an entity at a behavioural level, while another architecture may be a structural model

Design entity and component

- Entity is a component of a design
  - Component reusability
    - A component can be saved in a component library this will enabling it to be copied as many times as required
  - Ports
    - The inputs and outputs of the circuits
    - They are special programming objects
    - Ports are signals
    - Ports are the means used by the circuit to communicate with the external world, or with other circuits
    - Each port must be declared to be of a particular type

Fundamental terms

Architecture

- It is a code that specifies the behaviour of a component
- There can be more than one architecture for an entity
- Each architecture would specify some different modelling levels

Configuration

- It specifies the entities, architectures to use for components within a particular model
- In hardware design tools there is not configuration at all, or the integrated tool has configuration functionality

Package

- A collection of type definitions, procedures, functions, and component declarations

Library

- A collection of entities, architectures, configurations, packages
- Most designs import library modules

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY HALFADDER IS
    PORT(U,   V     : IN  STD_LOGIC;
         SUM, CARRY : OUT STD_LOGIC);
END HALFADDER;

ARCHITECTURE RTL_HALFADDER OF HALFADDER IS
    BEGIN
      SUM   <= U XOR V;
      CARRY <= U AND V;
END RTL_HALFADDER;
```

Example of entity declaration and architecture

```
entity andGate is
            port (a, b: in bit;
                  q: out bit);
end andGate;

architecture str of andGate is
        -- declarations here
begin
        -- statements here
end str;
```

- Two names are specified in the architecture declaration
  - *Component name* describes which entity the architecture belongs to
  - *Architecture name*
- The entity name in the architecture has to be the same as the identifier (entity name) of the corresponding entity declaration

- The highlighted (bold face) words are key words in VHDL
- The other words are user given