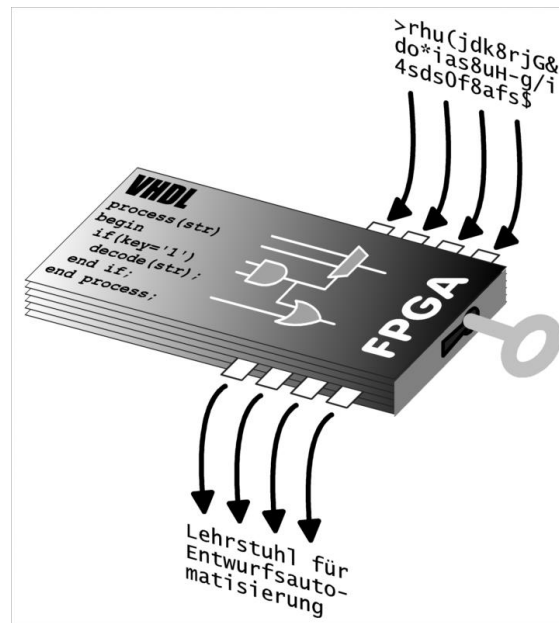


VHDL System Design Laboratory



Institute for Electronic Design Automation
Technische Universität München

Outline

- Organization
- Cryptography
- IDEA Algorithm and FPGA
- VHDL Introduction
- Design Flow
- VHDL Example: XOR

Outline

- **Organization**
- Cryptography
- IDEA Algorithm and FPGA
- VHDL Introduction
- Design Flow
- VHDL Example: XOR

Laboratory Tasks

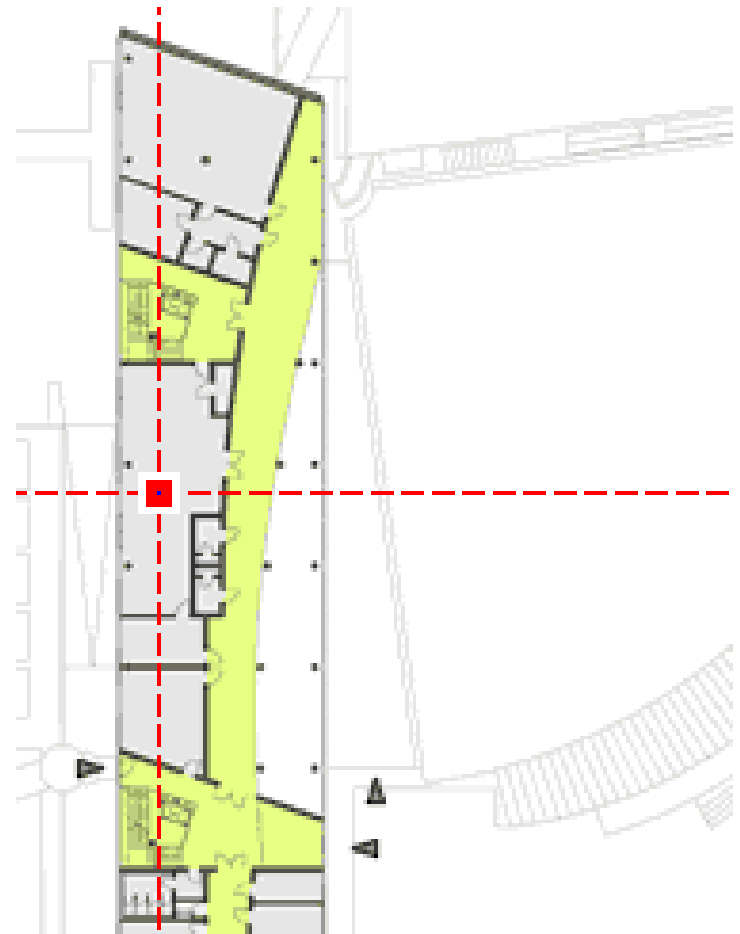
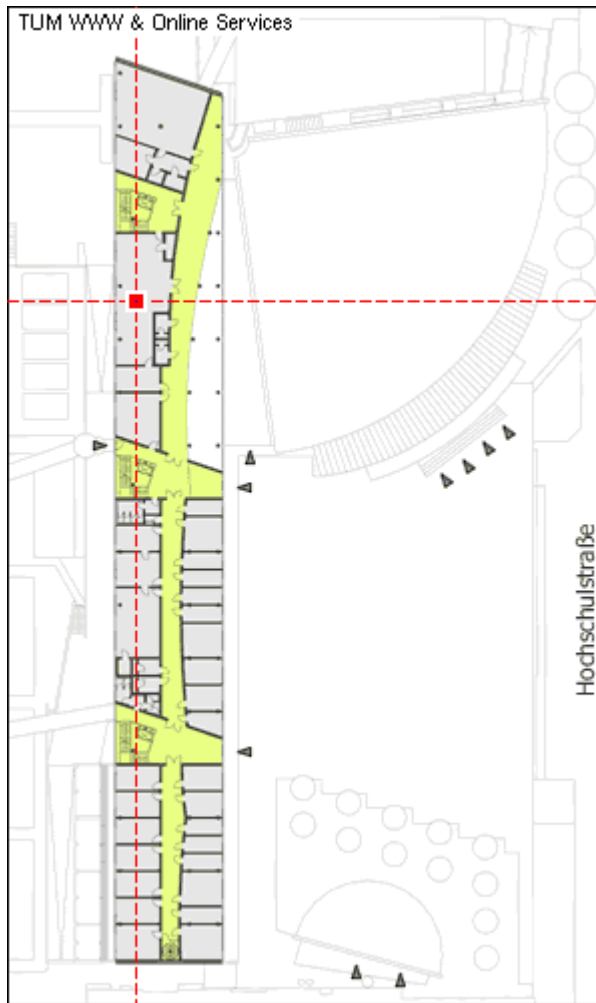
- Implementation of an encryption and decryption algorithm (IDEA algorithm) in VHDL
- 3 different implementations in order to optimize the used HW resources
- Simulation and synthesis of different implementations
- Program and test the different implementations on the FPGA (Field Programmable Gate Array)

Organizational Matters

- Independent work with submission deadlines (Lab rooms or at home on your PC)
- Login: Your LRZ/TUMOnline-Login
- Software: Xilinx ISE (Windows, Linux)
 - Download available on Xilinx Webpage (Link will be provided in Moodle)
- 4 Introduction Classes (Room 1260)
- Tutorhours starting from 10/19 (Room 2977):
 - MON 18:00 – 20:00, TUE 09:00 – 13:00 and WED 11:15 – 13:15
 - No presentation of new stuff
 - Just provided help to do the project
- Materials are available in Moodle

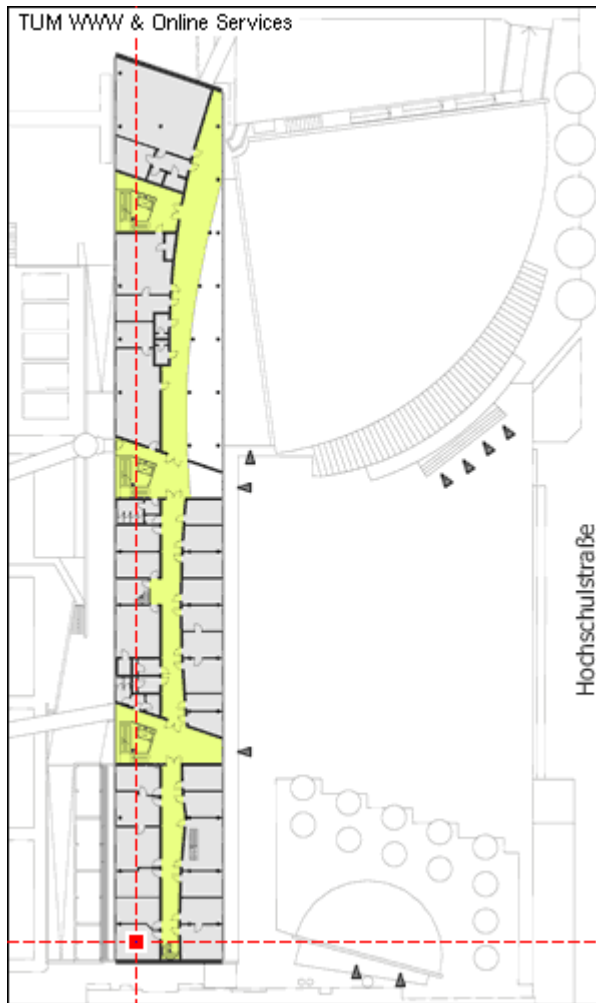
Lab Rooms

2977 (Access with Student ID from staircase)



Lab Rooms

3905a (Access by asking the supervisor)



Contact Information

- Mailing List (Email with your login on title)
eda-hdl-support@lists.lrz-muenchen.de
Tutor: Korbinian Weigl
- Tutorhour: Shrivathsan Narayanan



- Supervisor: Dipl.-Ing. Andreas Herrmann (Room 4909)
andreas.herrmann@tum.de

Schedule

Date	Task
Fr. 10/16 11:30	Class: Project Introduction
Fr. 10/23 11:30	Class: VHDL 1
Thu. 11/05 24:00	Submit Code: Direct Implementation
Fr. 11/06 11:30	Class: VHDL 2 + Board Distribution
Thu 11/26 24:00	Submit Code: Resource Constraint Scheduling 1
Fr. 11/27 11:30	Class: Hardware Debugging
Thu. 01/28 24:00	Submit Code: Resource Constraint Scheduling 2/+
Fr. 01/29 15:00	Submit E-Test in Moodle + Return Board
Fr. 05/02 13:30	Final Written Exam (1h) (Room: N1080 + 2750)

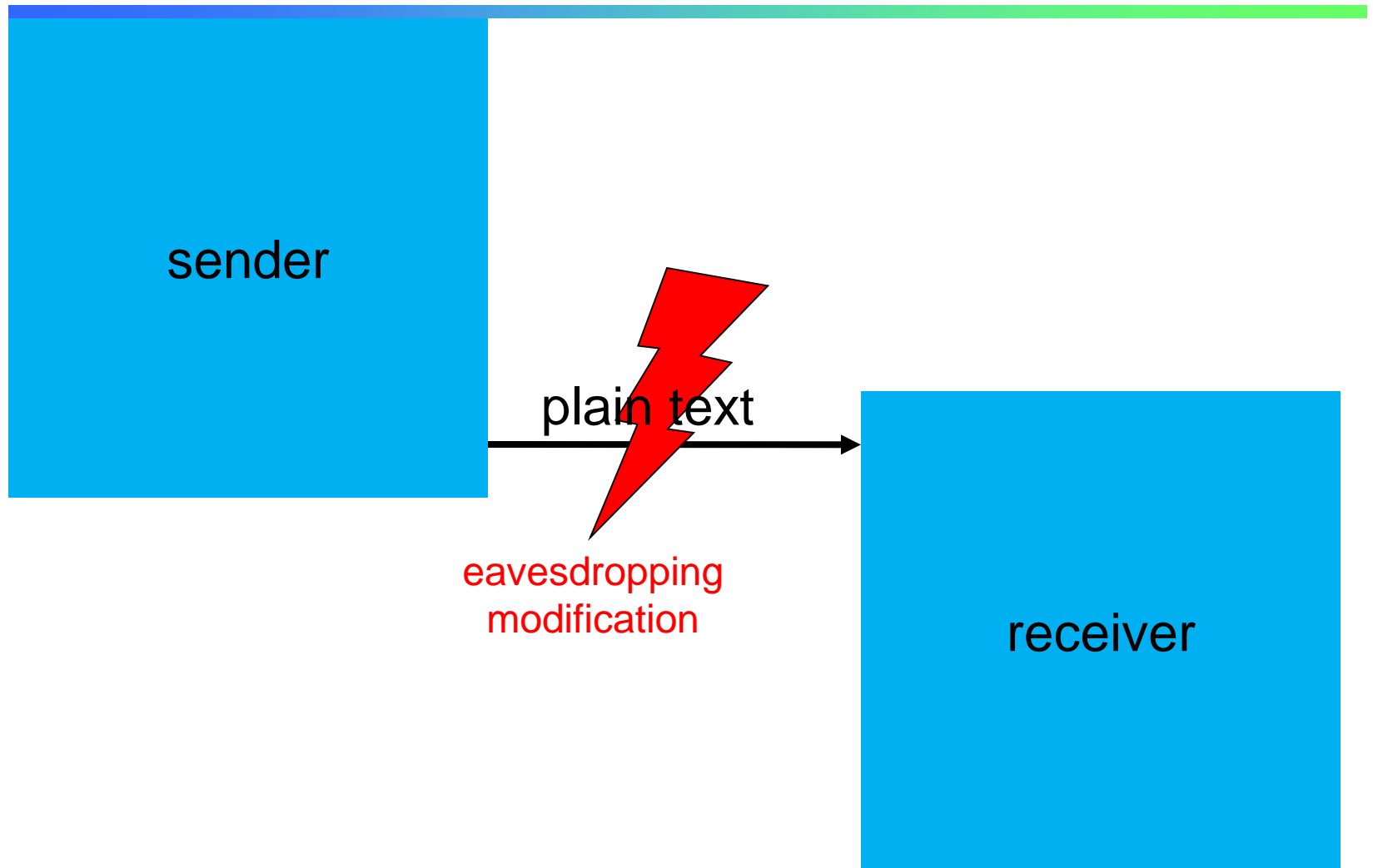
Grading

- Project (60%)
 - Answered preparation questions in E-Test in Moodle
 - Project Code (VHDL files + Project files in the Submission Directory)
- Written Exam (40%)
 - VHDL – in general
 - Tasks performed in laboratory
 - Slides shown in the introduction lessons
 - Closed book exam

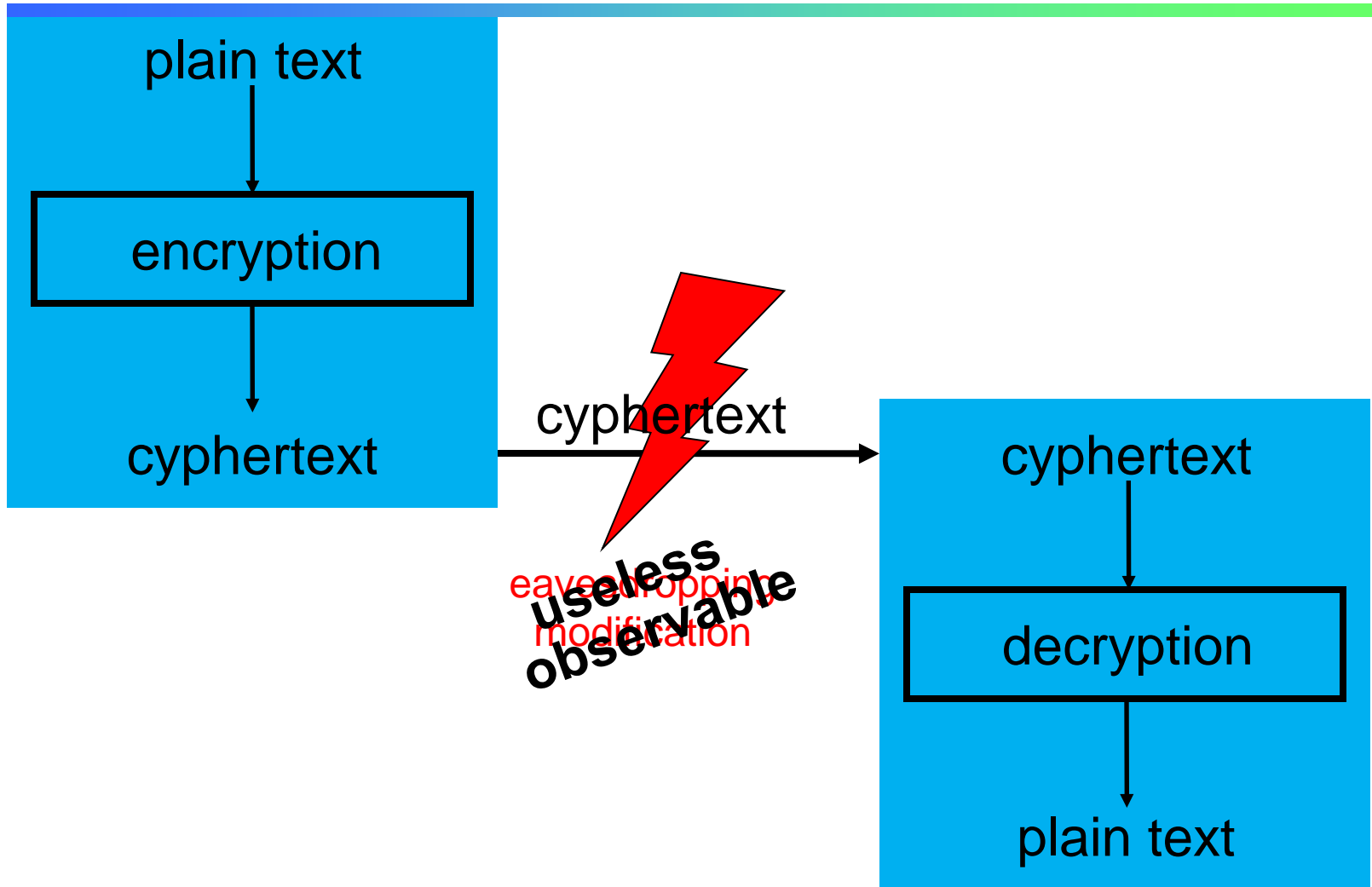
Outline

- Organization
- **Cryptography**
- IDEA Algorithm and FPGA
- VHDL Introduction
- Design Flow
- VHDL Example: XOR

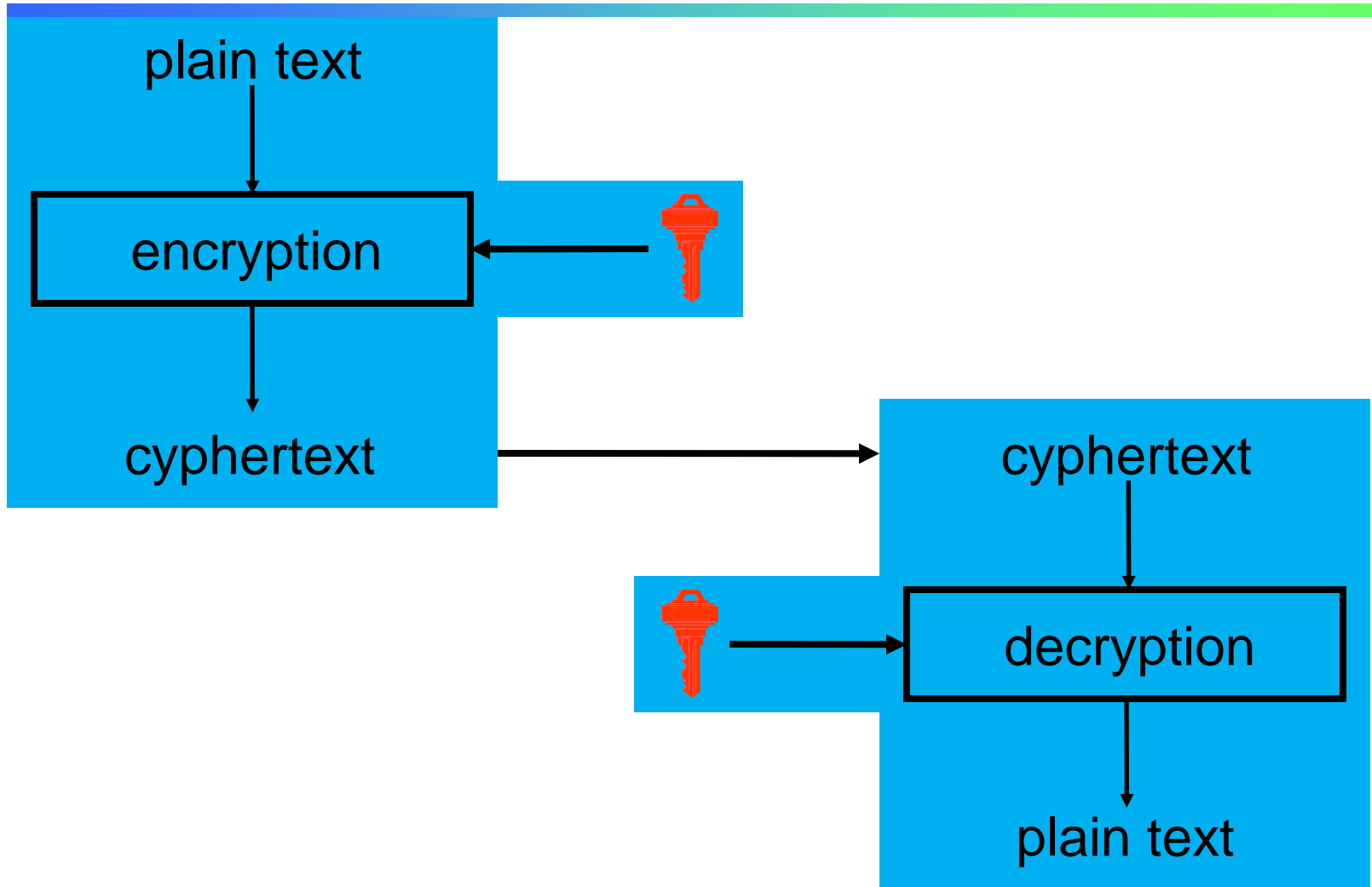
Message Transmission



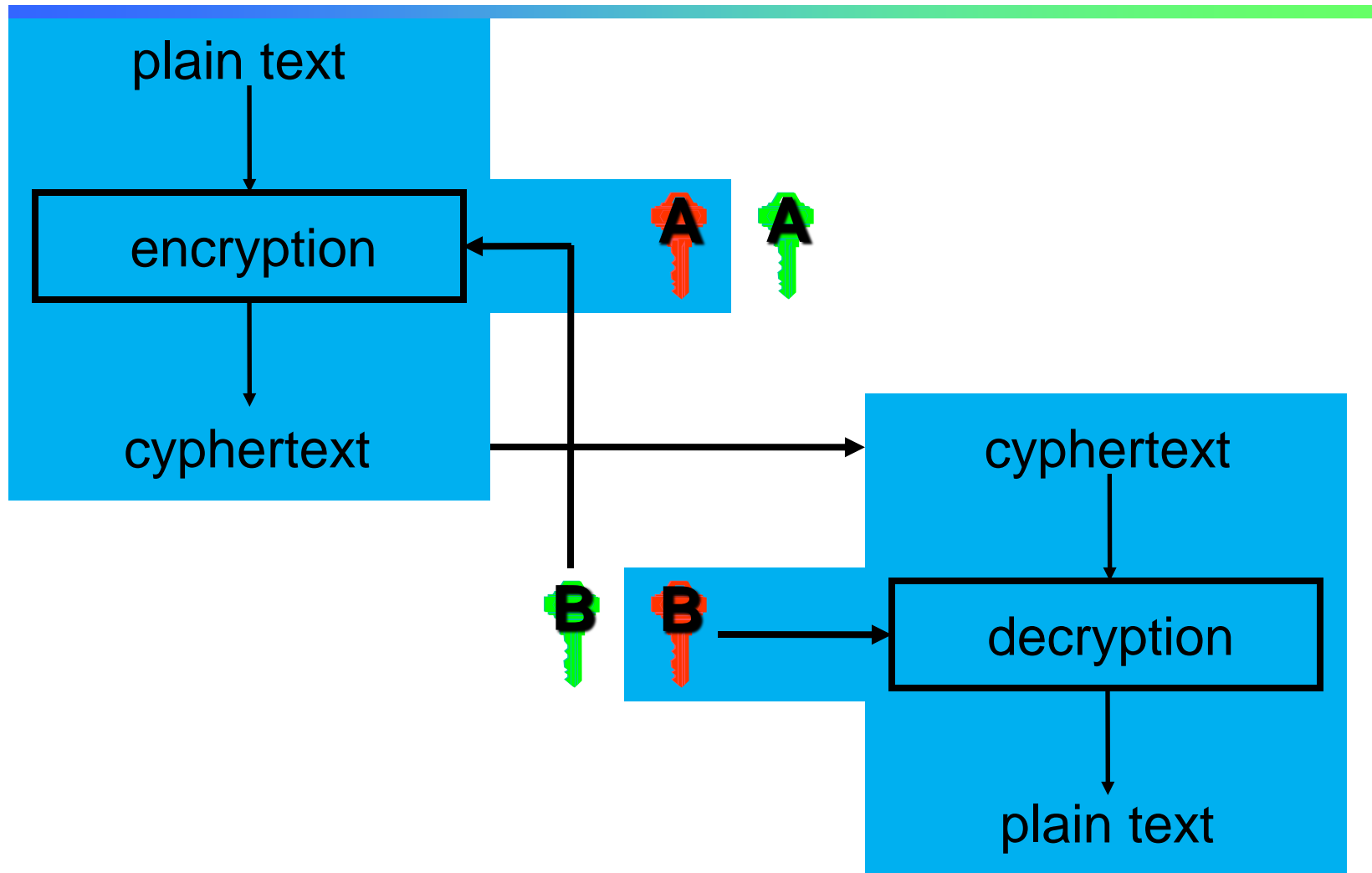
Message Transmission using Cryptography



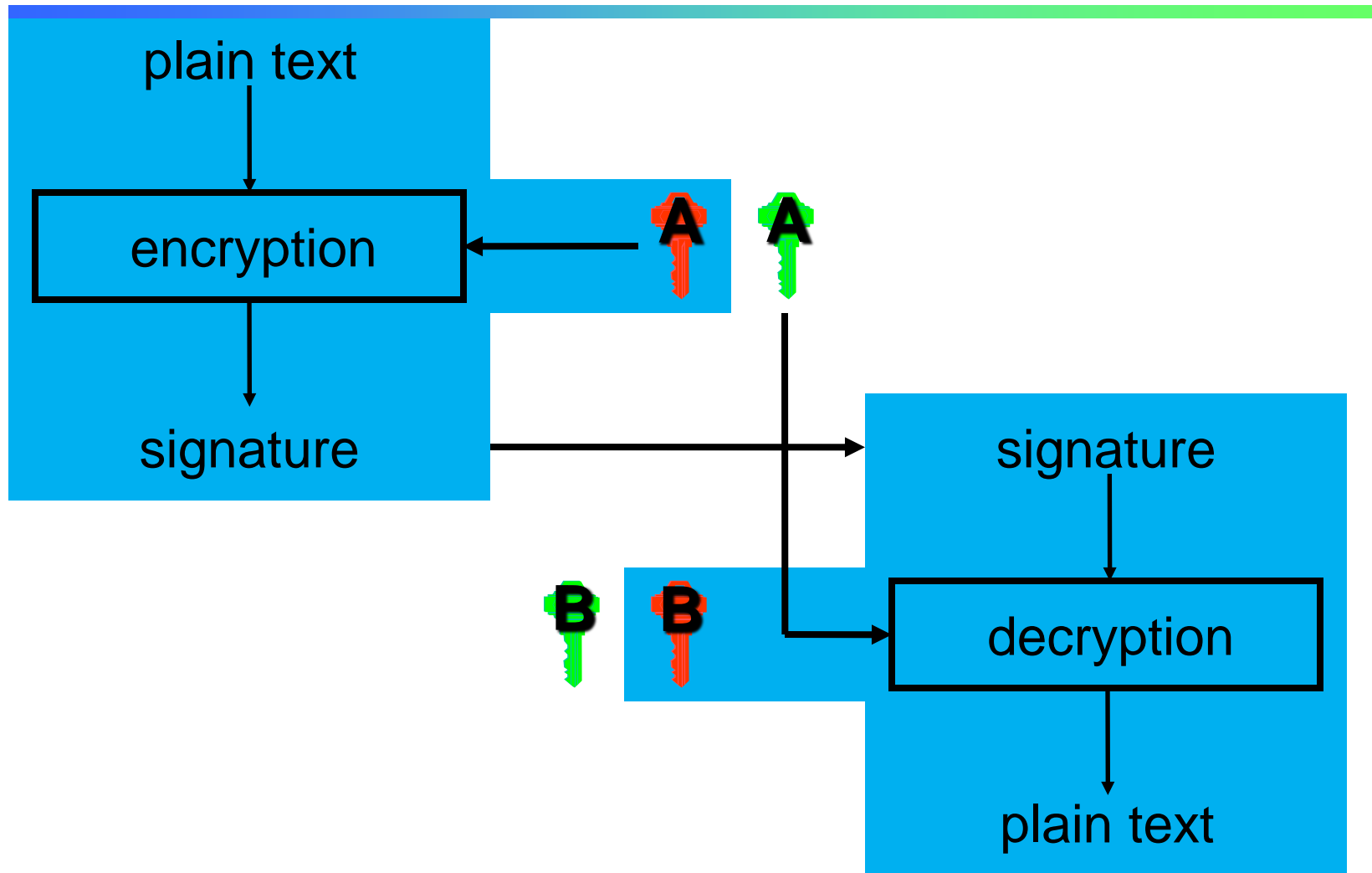
Symmetric Encryption Techniques



Asymmetric Encryption Techniques



Digital Signature



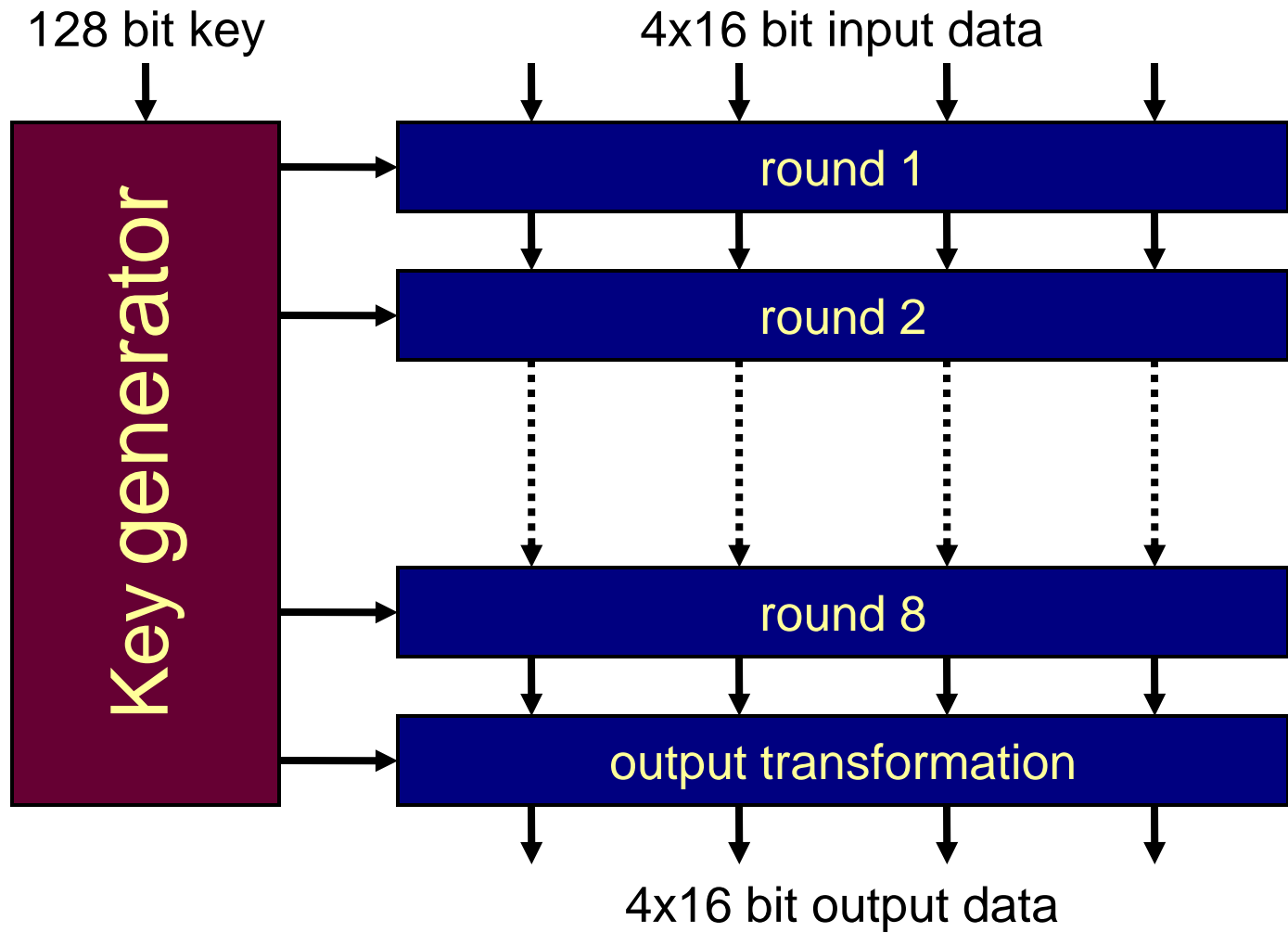
Outline

- Organization
- Cryptography
- **IDEA Algorithm and FPGA**
- VHDL Introduction
- Design Flow
- VHDL Example: XOR

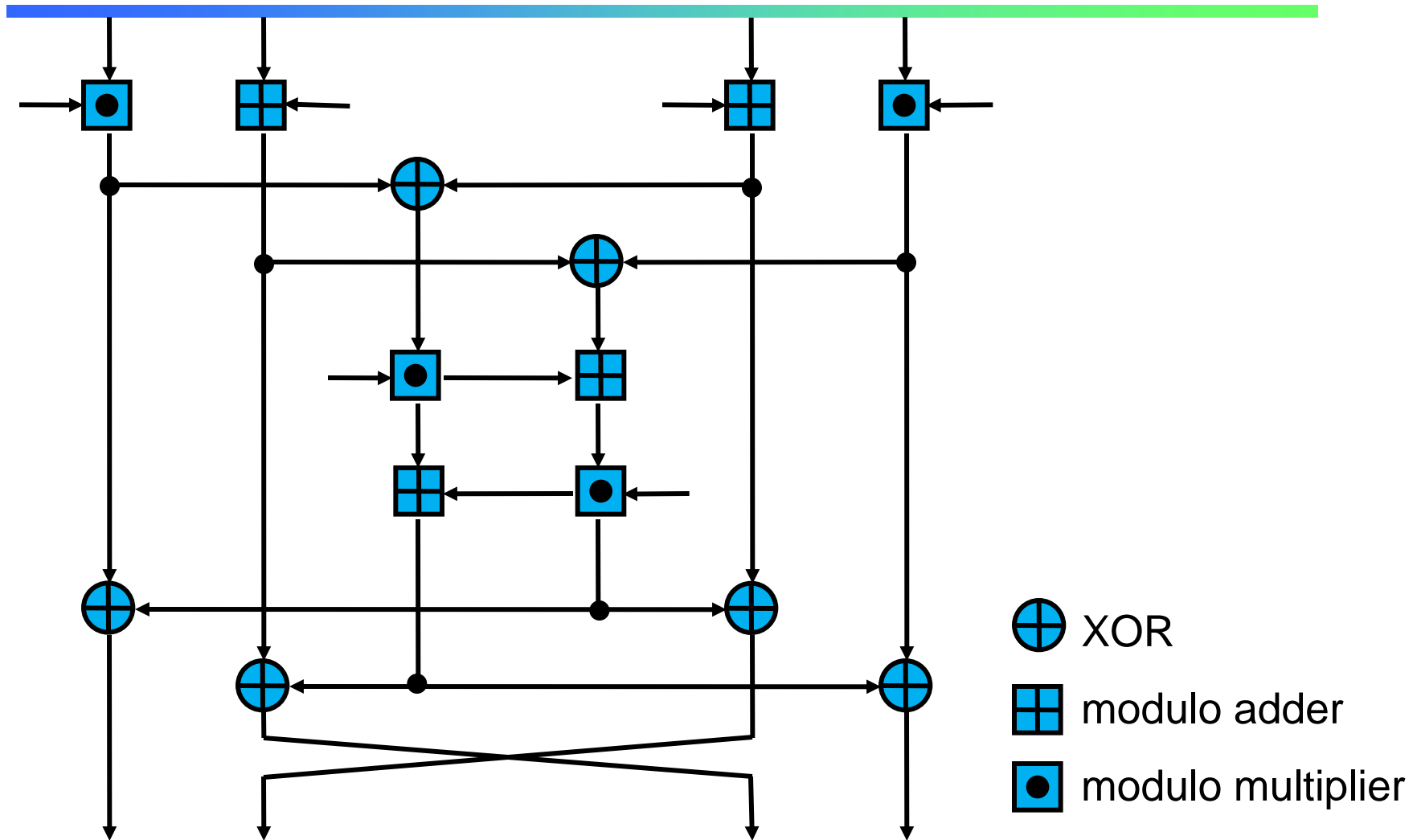
IDEA Algorithm

- symmetric
- block cipher, 4x16 bit input- / output data
- 128 bit key (52 partial keys of 16 bit length)
- 16 bit operations
 - XOR
 - addition modulo 2^{16}
 - multiplication modulo $2^{16} + 1$

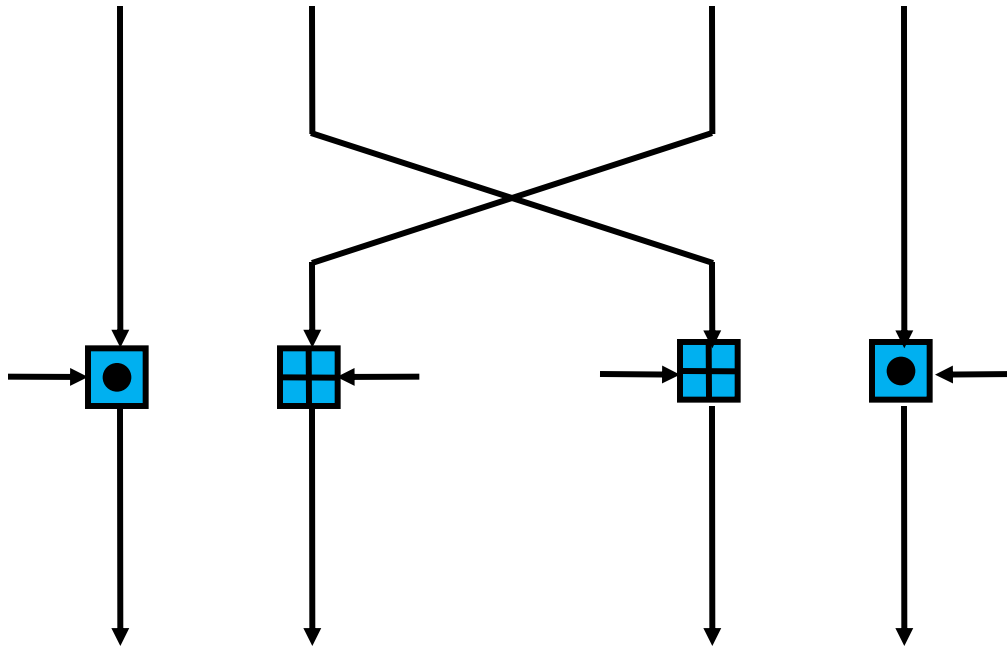
IDEA Algorithm





Round Module

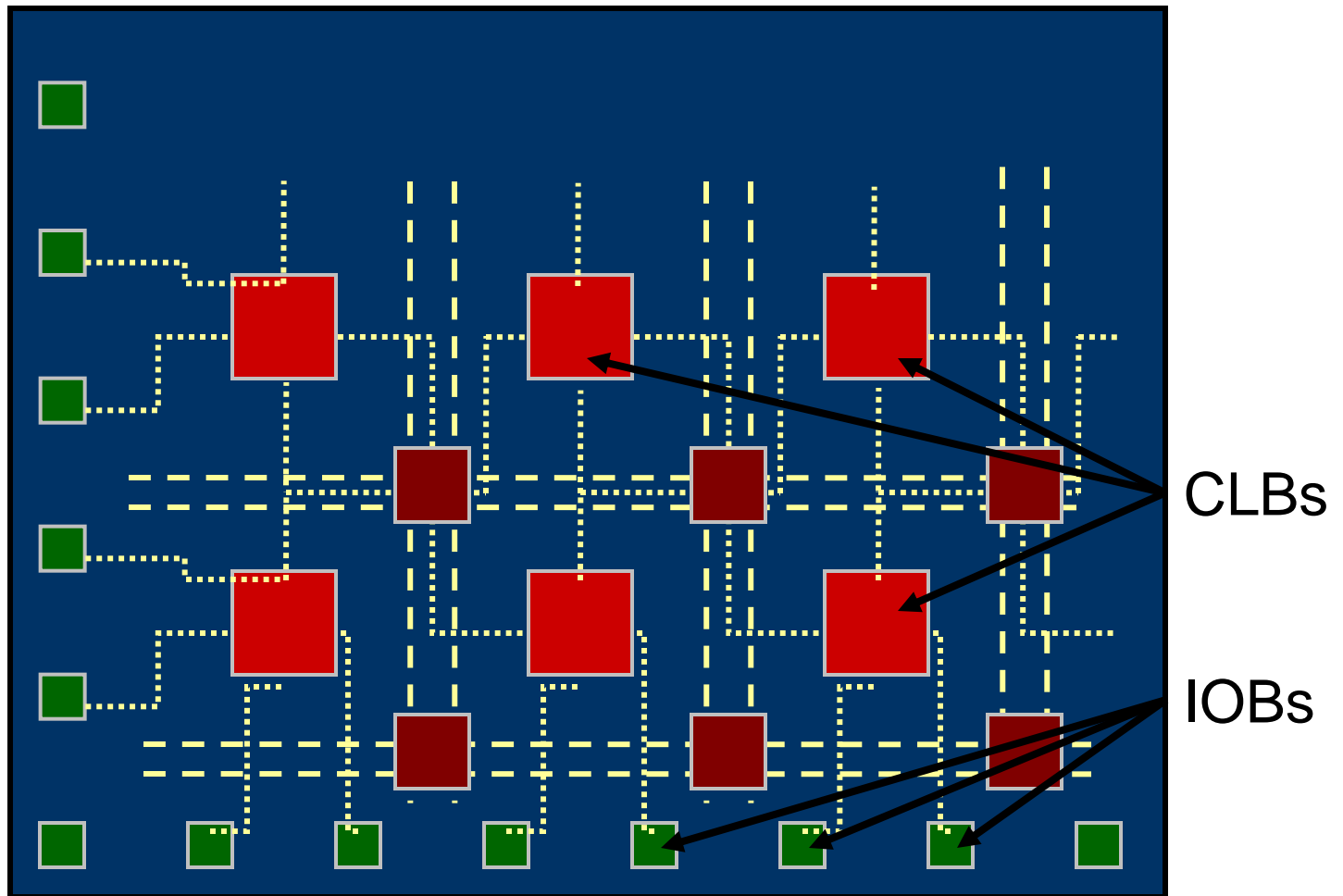


Output Transformation



 modulo adder
 modulo multiplier

Xilinx FPGA



XC3S500 Ressources

available:

required:

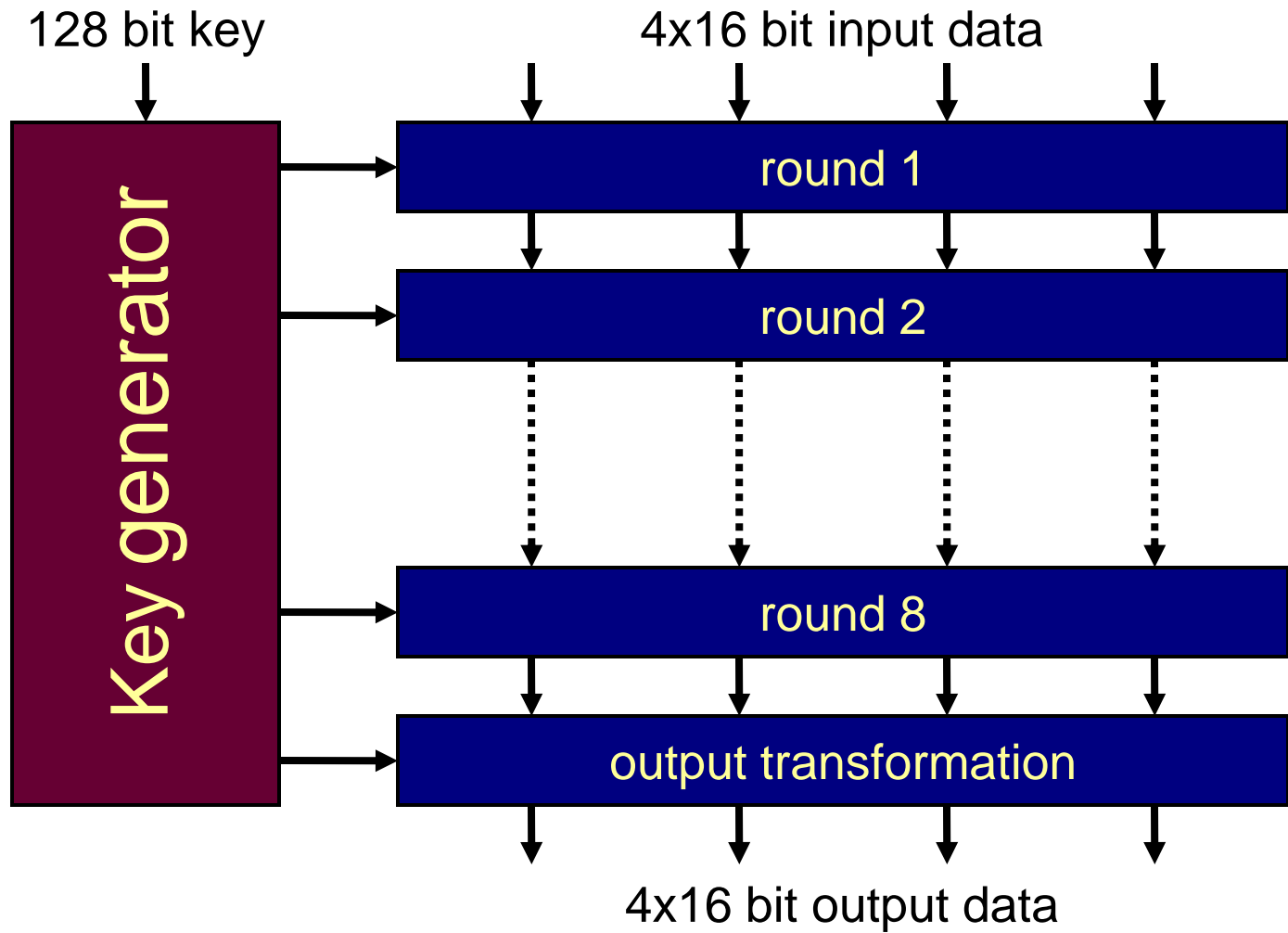
CLBs:

1164

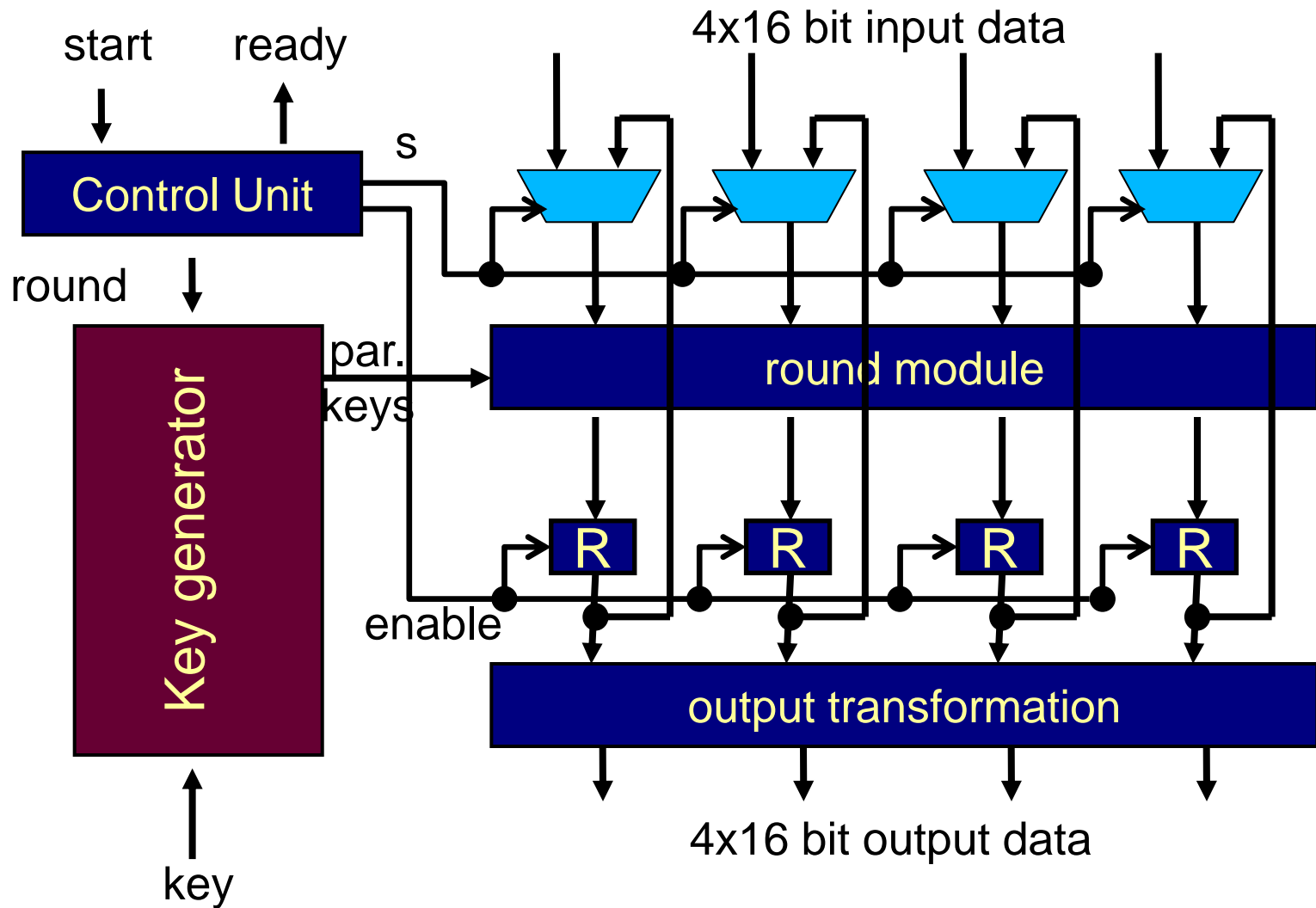
≈ 1750



Direct Implementation



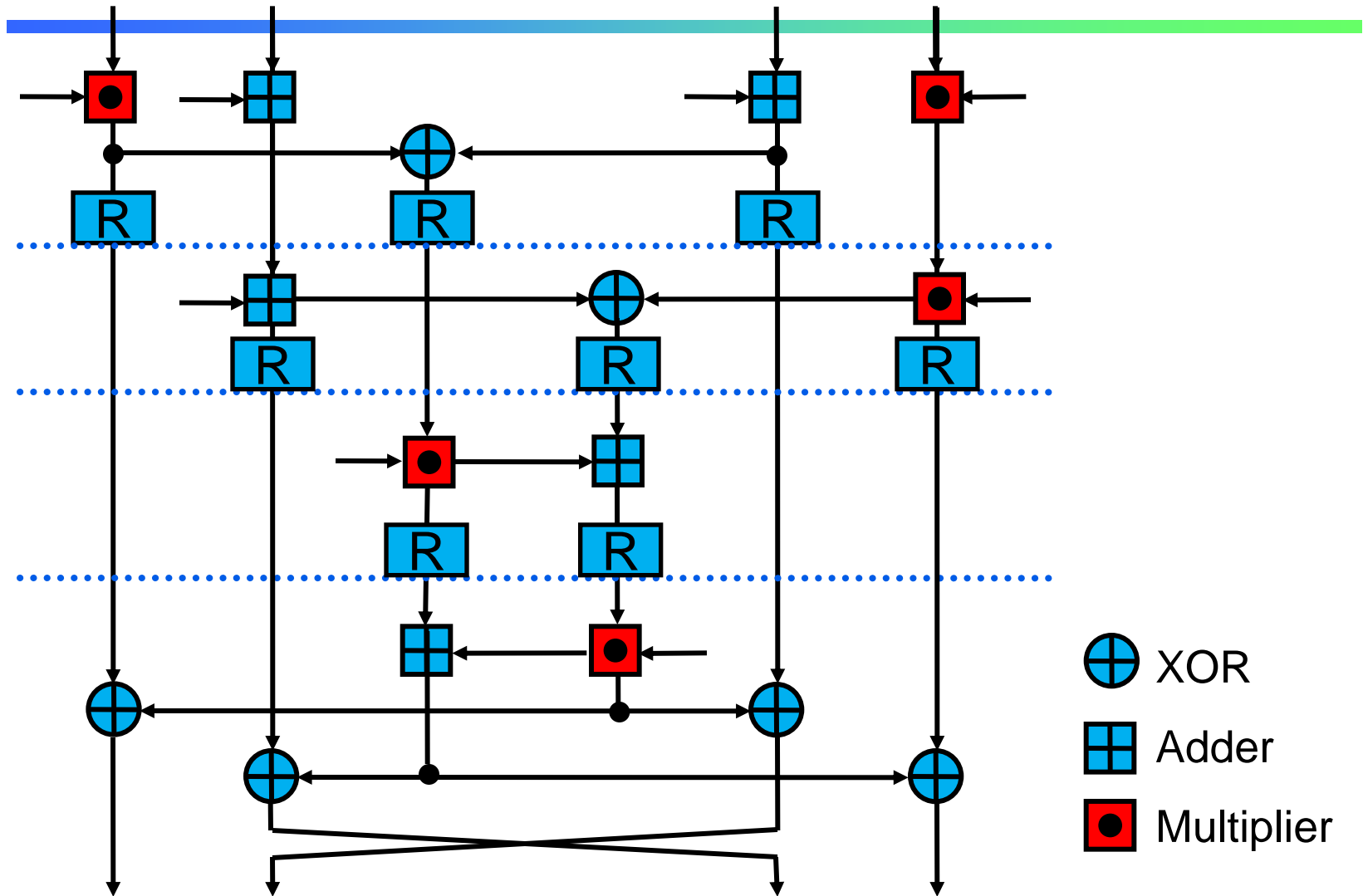
Resource Constrained Scheduling I



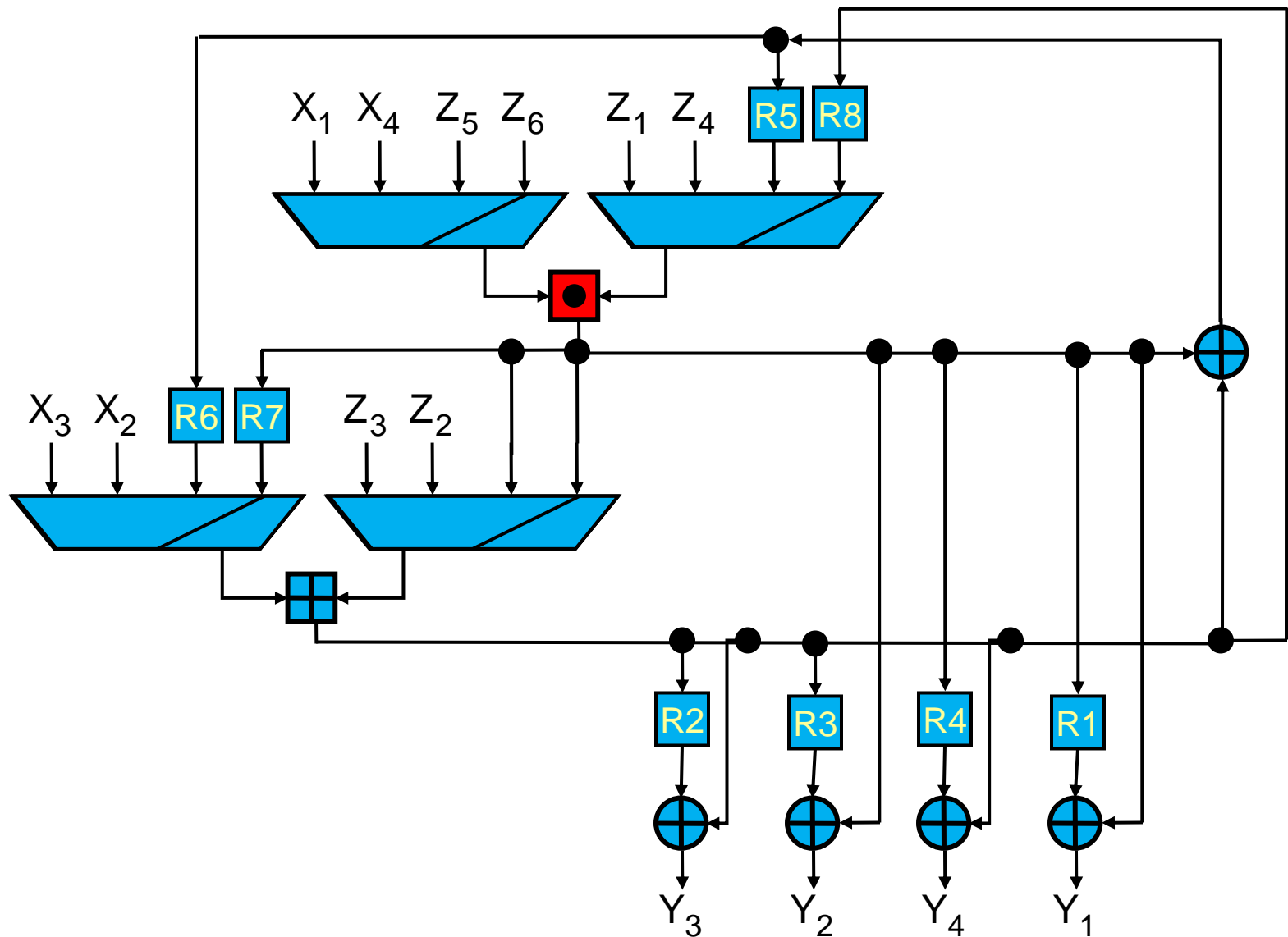
Think now ...

- Could we reduce the resource use further?
- Go into the round module (round module needs still many CLBs)

Round Module



Resource Constrained Scheduling II : Data Path



Outline

- Organization
- Cryptography
- IDEA Algorithm and FPGA
- **VHDL Introduction**
- Design Flow
- VHDL Example: XOR

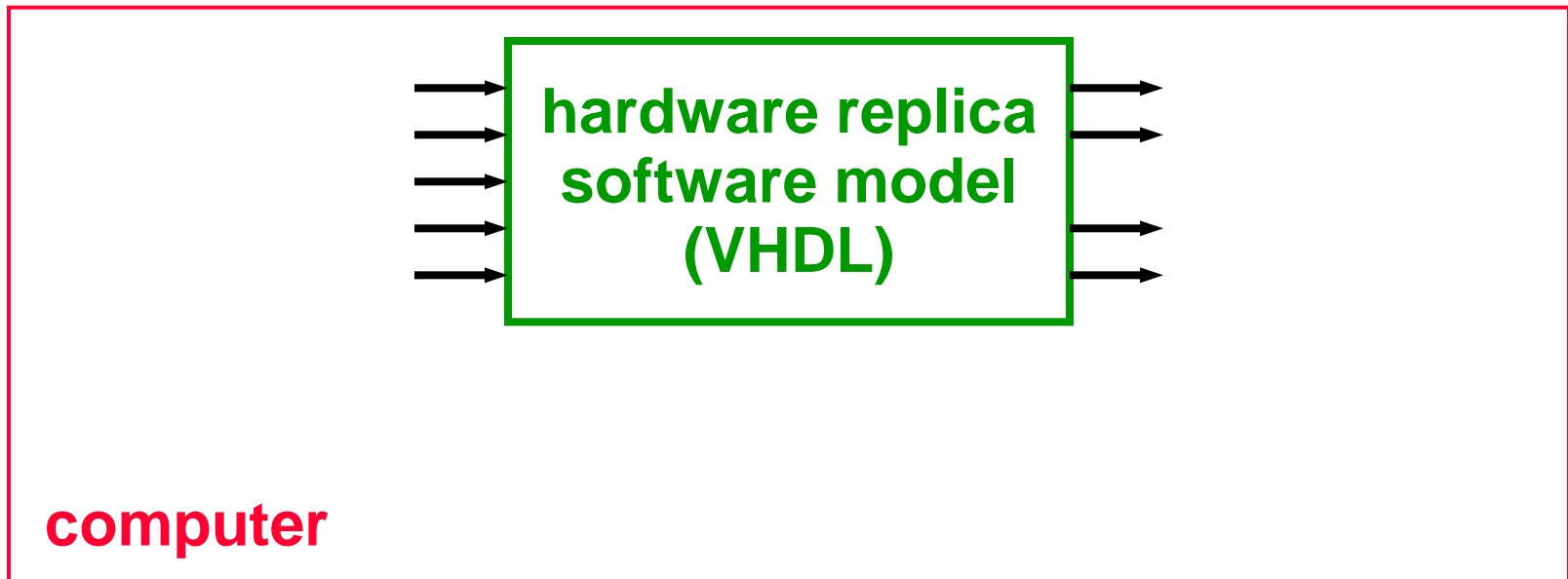
VHDL

- **V** VHSIC (Very High Speed Integrated Circuit) or VLSI
- **H** Hardware
- **D** Description or Design
- **L** Language
- Common **programming language** to describe hardware
- Standardized by IEEE and ANSI since 1987 (VHDL-87).
In this lab VHDL-93 is used
- **Concurrent** Modeling
- **Hierarchical** Modeling
- Support of **different design styles** (Behavioral ↔ Structural)
- Verilog is another HDL widely used in the U.S.A.
- Extension for describing analog and mixed-signal circuits:
VHDL-AMS

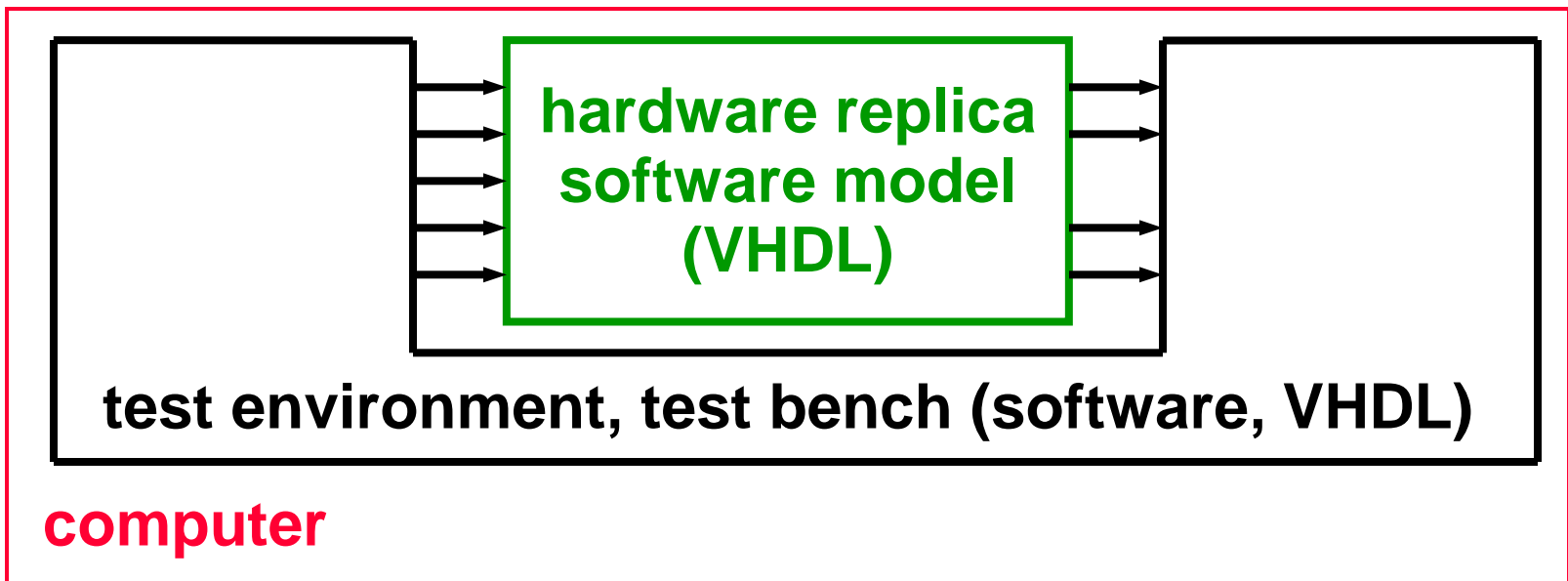
Literature

- Peter J. Ashenden
The Designer's Guide to VHDL
Morgan Kaufman Publishers, 2008
- Peter J. Ashenden
VHDL Tutorial, 2004 (see Moodle)

Hardware Emulation



Hardware Emulation



Modeling of Digital Systems

- **What is the purpose of Hardware Description Languages ?**
 - Describing hardware circuits and systems by using software models
- **What is a model?**
 - idealized description (an abstraction) of a real object
 - expresses important properties and neglects minor properties
 - models described with formal languages like VHDL have well-described syntax and semantics and can thus be interpreted unambiguously => **formal models**

Application Domains of HDL Models

- **What are the main applications of HDL models?**
 - validation of the correct function and the timing behavior by simulation
 - automated synthesis
 - design reuse (reuse of VHDL modules)
 - data transfer between vendors and designers and between design systems
 - prototyping and design specification
- **What are the major objectives for using HDLs?**
 - avoidance of design errors requiring redesigns
 - minimization of design cost and time

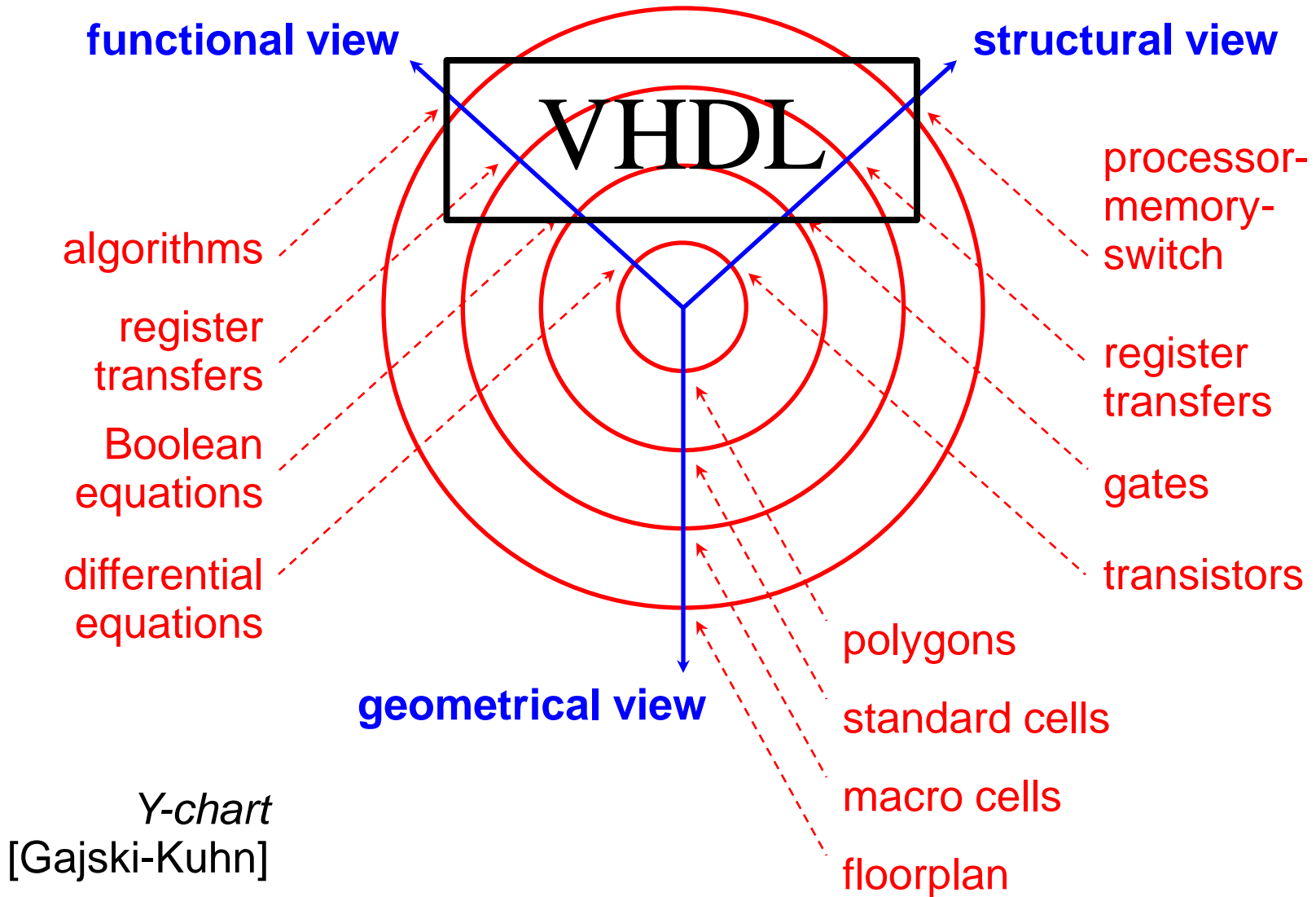
The Design Space

Where do we use VHDL?

		<i>design view</i>		
		<i>behavior</i>	<i>structure</i>	<i>geometry</i>
<i>abstraction level</i>	<i>architecture</i>	system specification	system partitioning	floorplan
	<i>register transfer</i>	algorithms	module netlist (alu, mux, register)	macro cells (IP blocks)
	<i>logic</i>	Boolean equations	gate netlist (gates, flipflops)	standard cells library cells
	<i>circuit</i>	differential equations	transistor netlist	mask data, polygons

VHDL

Modeling Space



Outline

- Organization
- Cryptography
- IDEA Algorithm and FPGA
- VHDL Introduction
- **Design Flow**
- VHDL Example: XOR

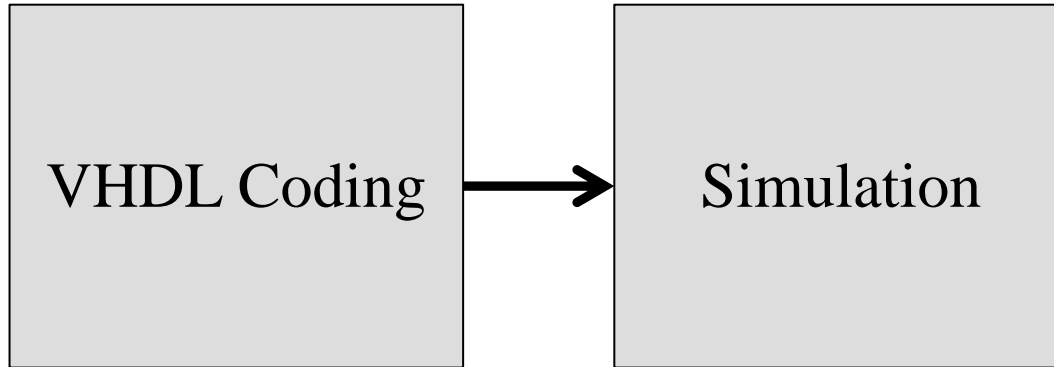
Design Flow

VHDL Coding

VHDL Coding

- Describes our model
 - Behaviour
 - Structure
 - Dataflow
- More information in the next introduction classes

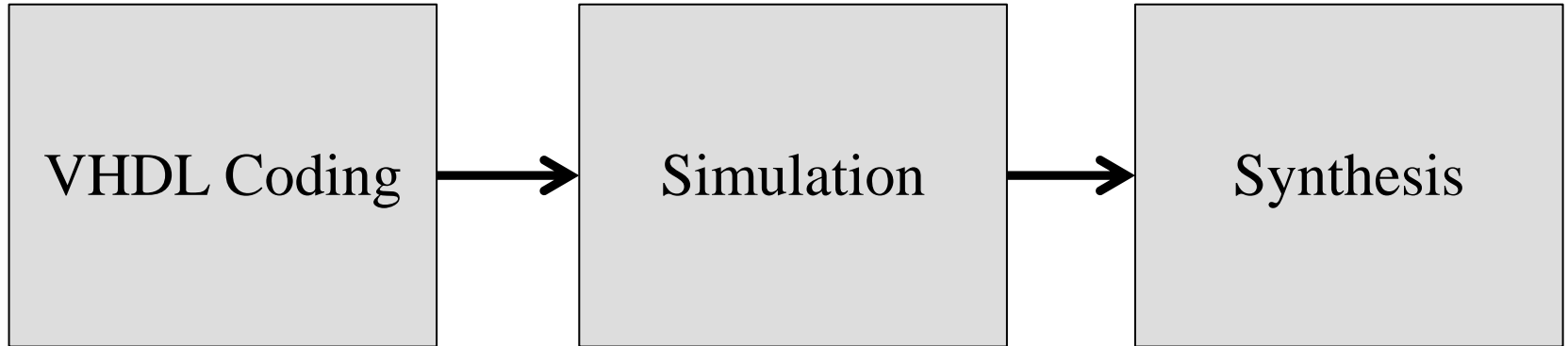
Design Flow



Simulation

- Check syntax
- Testbench including the stimuli
- Output: Waveforms
- Logical verification of the model
 - Fulfill specification?
 - Timing requirements fulfilled?

Design Flow



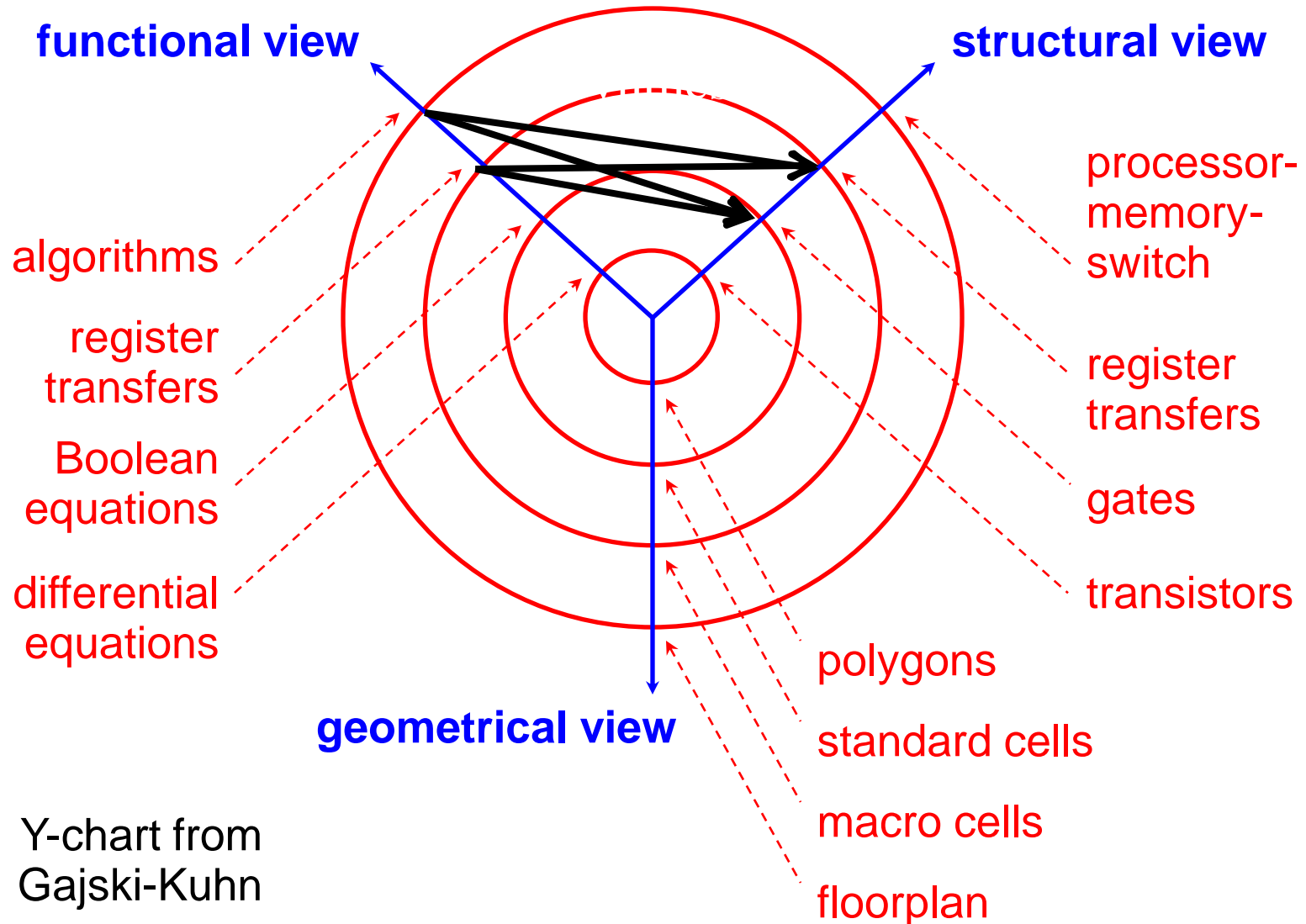
Synthesis

- Input: working structural or behavioral VHDL model
- Output: Gate / RTL Netlist
- No information about the later Hardware

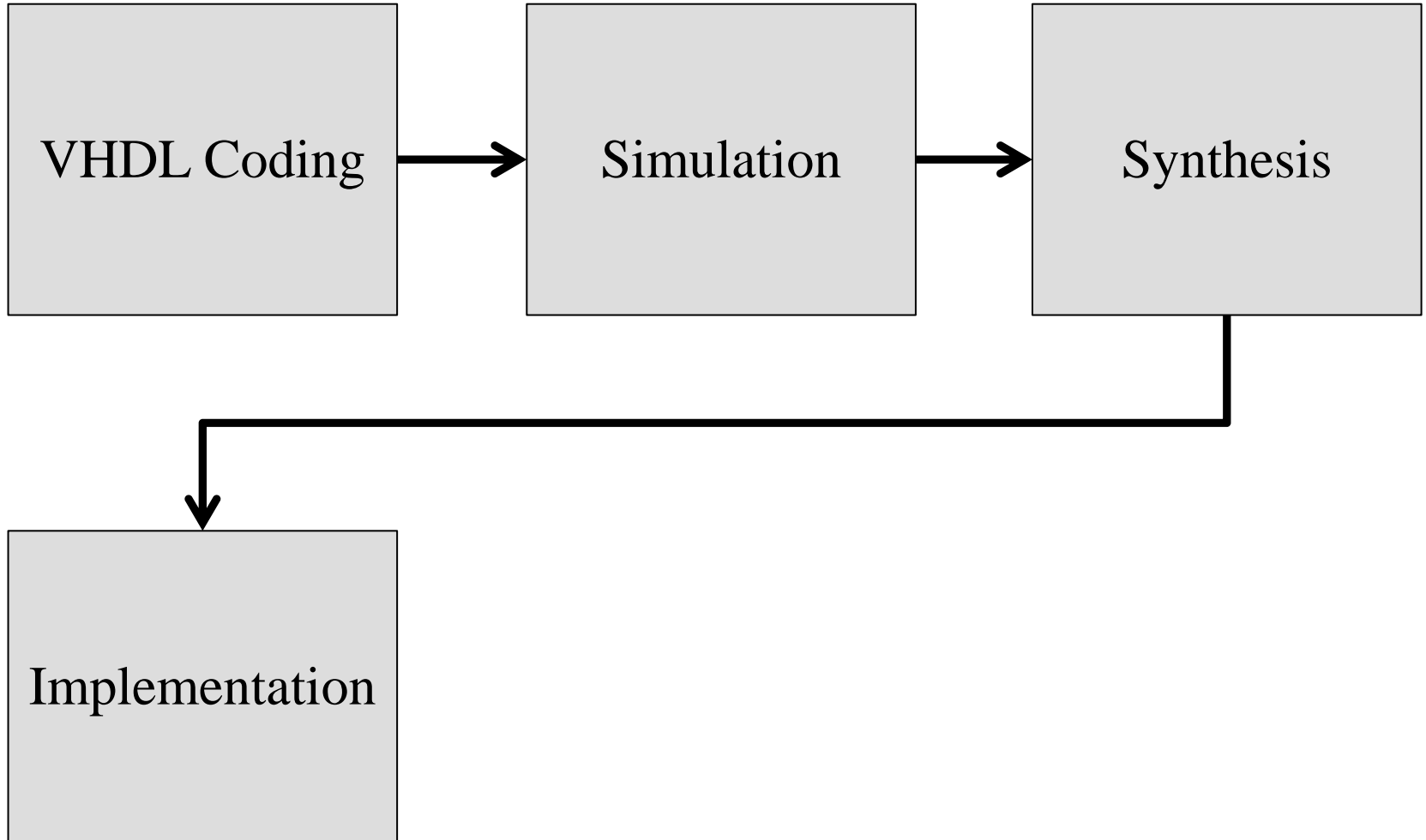
Synthesis

		<i>design view</i>		
		<i>behavior</i>	<i>structure</i>	<i>geometry</i>
<i>abstraction level</i>	<i>architecture</i>	system specification	system partitioning	floorplan
	<i>register transfer</i>	algorithms	module netlist (alu, mux, register)	macro cells (IP blocks)
	<i>logic</i>	Boolean equations	gate netlist (gates, flipflops)	standard cells library cells
	<i>circuit</i>	differential equations	transistor netlist	mask data, polygons

Synthesis



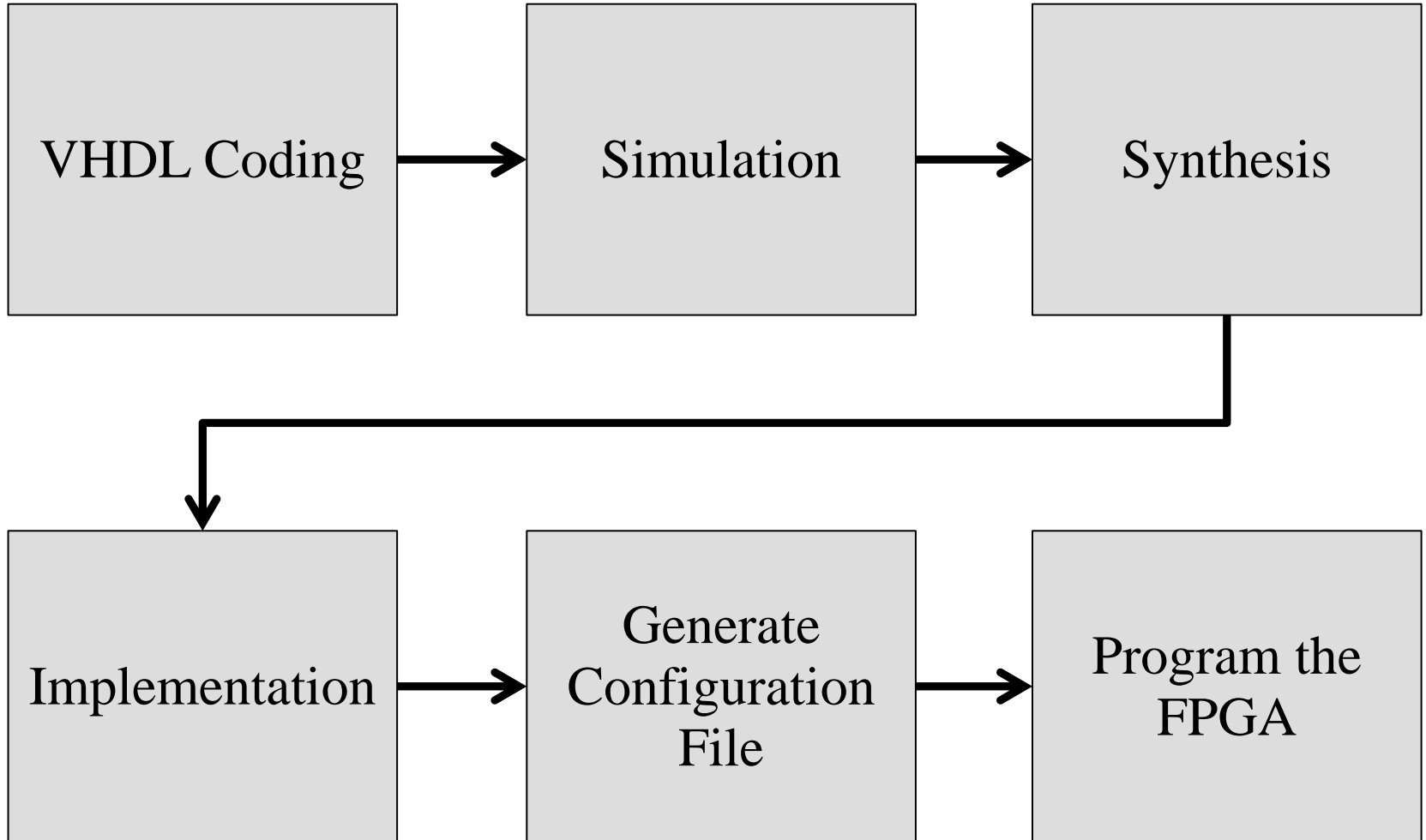
Design Flow



Implementation

- Translation
 - Synthesized netlist is translated into a Xilinx file
- Map
 - Devices of the netlist are mapped to the type of HW resources
- Place & Route
 - Exact placement and wiring is defined

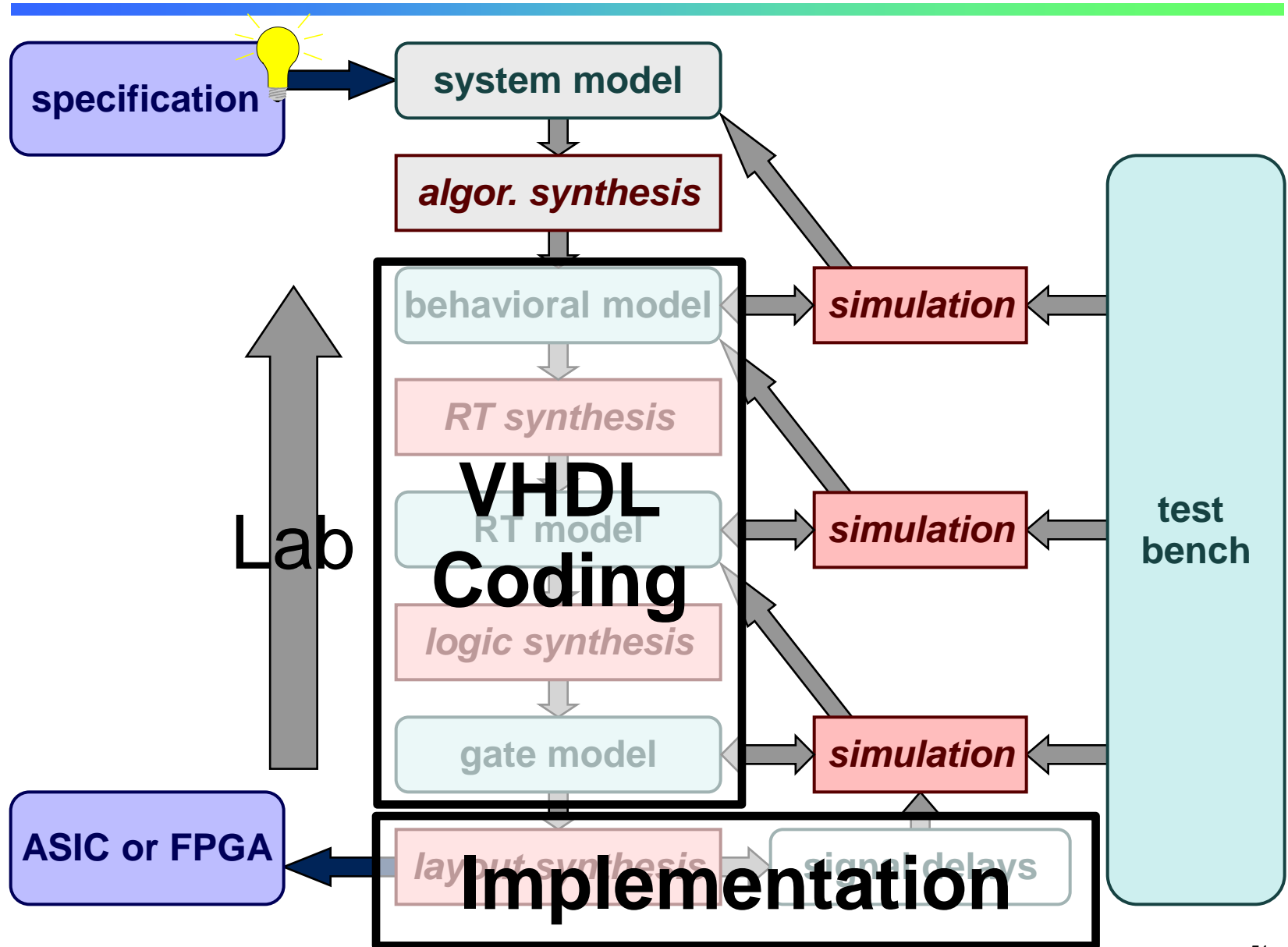
Design Flow



Generation of the Configuration File & Programming

- Based on the Place & Route step the configuration file (here: *.bit) is generated
- This file is used to program (flash) the FPGA via the USB port

Outlook: The Design Process

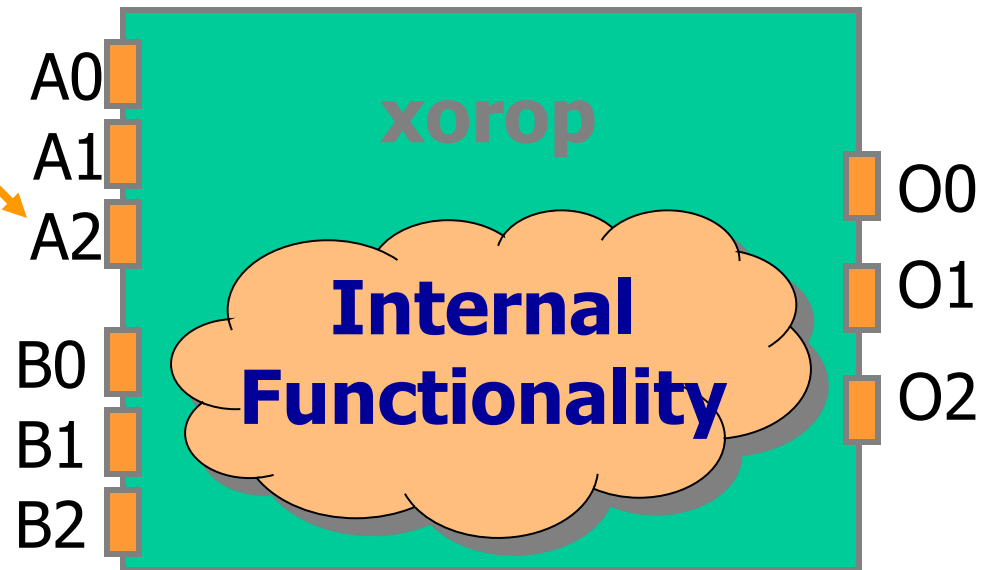


Outline

- Organization
- Cryptography
- IDEA Algorithm and FPGA
- VHDL Introduction
- Design Flow
- **VHDL Example: XOR**

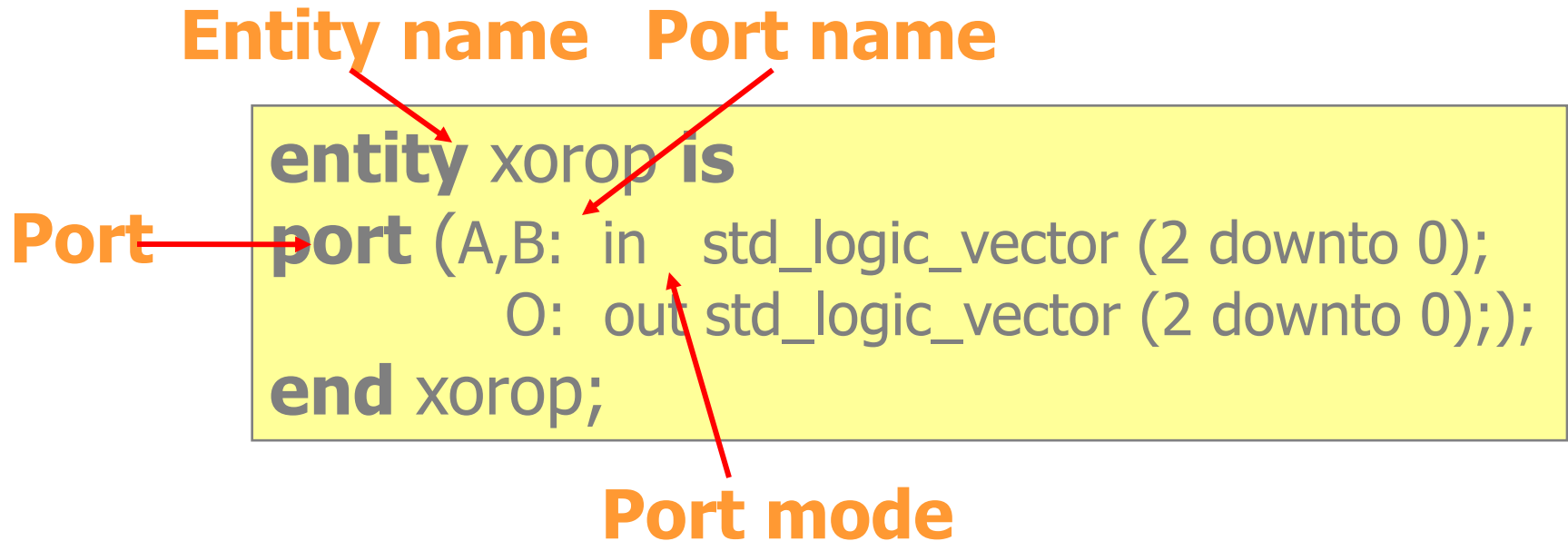
Basic Modeling Concepts: Device

External
Interface



Basic Modeling Concepts: Entity

External Interface modeled by “entity” VHDL construct.



VHDL “port” construct models data input/output.

Basic Modeling Concepts: Architecture

Internal Functionality modeled by “architecture”
VHDL construct

Architecture name

Entity name



```
architecture arch1 of xorop is  
begin  
.....  
end arch1;
```

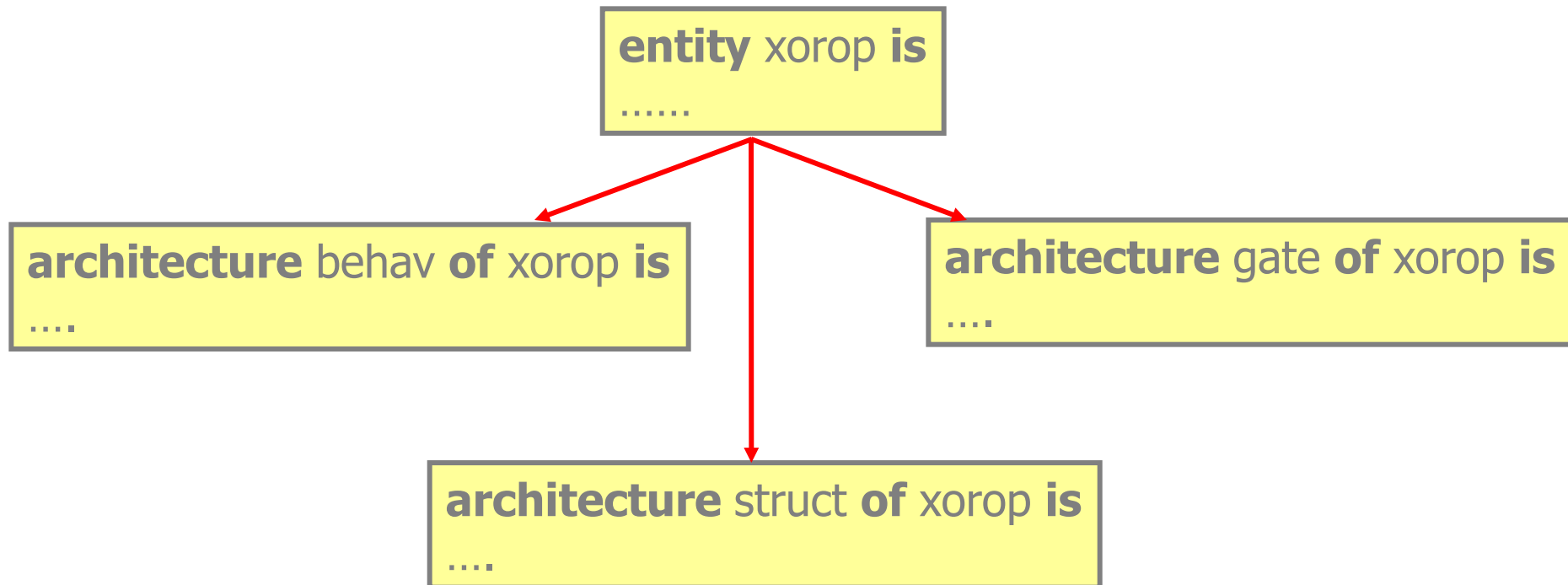
Architecture: Introduction

```
architecture arch1 of xorop is
begin
.....
end arch1;
```

- Between “**begin**” and “**end**” there are concurrent statements.
- A pure behavioral description contains only **process statements** and signal assignments.
- A pure structural description contains only **component instantiation** statements.

Using several Architectures for one Entity

There can be more than one “architecture” or internal functionality descriptions associated with one “entity” declaration.



Architecture Example: XOR

Process Sensitivity List:

combinational logic:
all input signals
(right side of assignments)

sequential logic:
only clock and reset

Signal
Assignments

```
architecture arch1 of xorop is  
begin
```

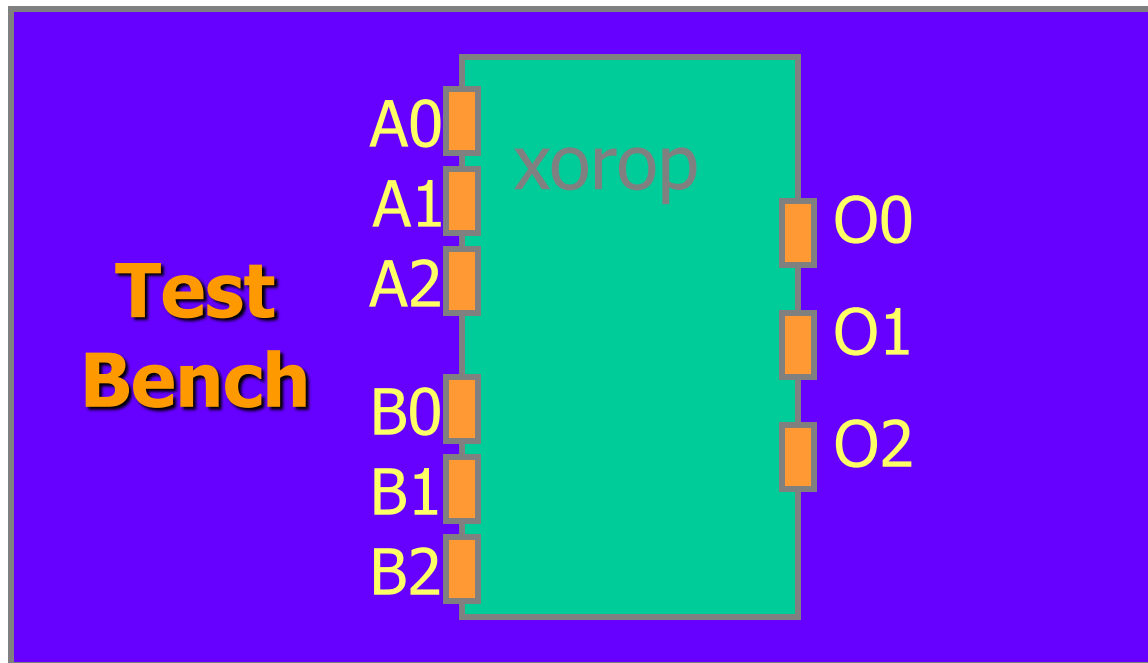
```
  process (A,B)  
  begin  
    O <= A xor B;  
  end process;
```

```
end arch1;
```

Input-/Output signals defined through entity

Testbench

Testbench: module that is used for testing the functionality of a design module by simulation.



Entity declarations of testbench modules have **no input and output ports**.

Testbench: XOR

```
entity tb_xorop is  
end;
```

no I/O Ports

```
architecture tb of tb_xorop is
```

```
component xorop
```

```
  port(  
    A : in std_logic_vector(2 downto 0);  
    B : in std_logic_vector(2 downto 0);  
    O : out std_logic_vector(2 downto 0)  );
```

make module
xorop known
to testbench

```
end component;
```

```
signal A, B, O : std_logic_vector(2 downto 0):="0000000000000000";
```

define test-
bench signals

```
begin
```

```
  my_xorop: xorop port map(A, B, O);
```

instantiation

```
  A<="000", "101" after 100 ns;
```

```
  B<="000", "101" after 200 ns;
```

apply stimulus
to input signals

```
end tb;
```