Introduction to VHDL

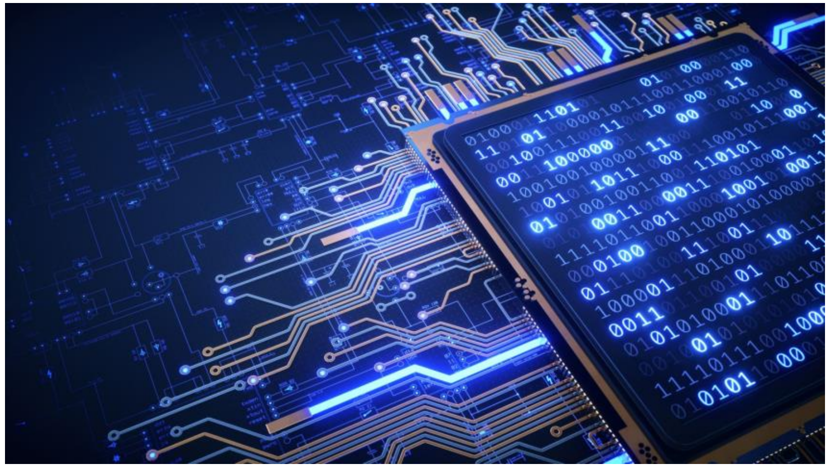
IAS0060 Digital Systems Design with VHDL

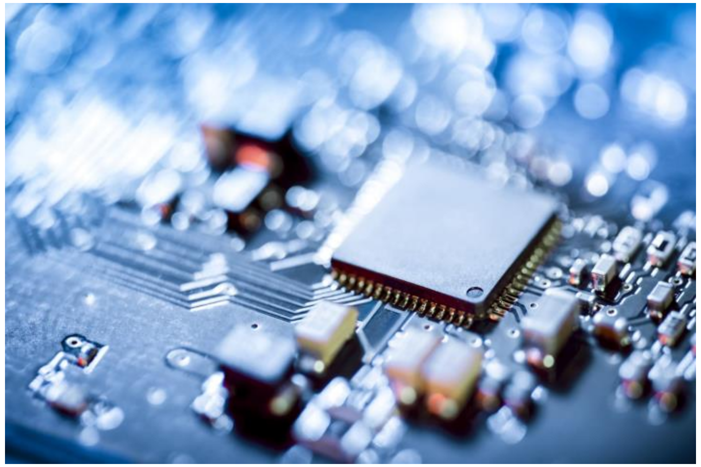
Natalia Cherezova

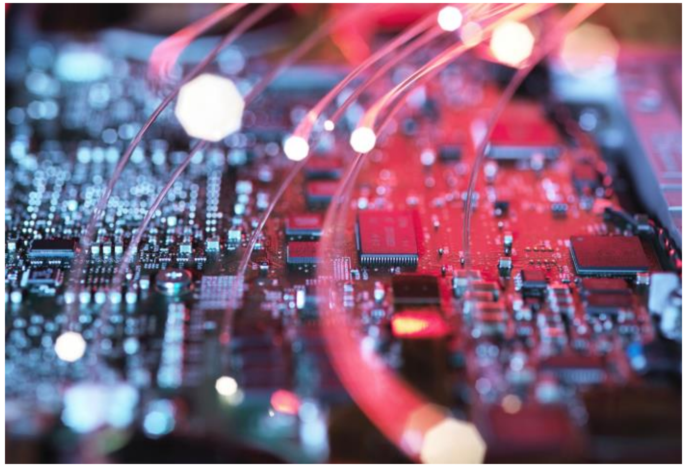
07.09.22

Digital systems

- A **discrete system** is a system in which signals have a finite number of **discrete** values
  - In contrast to analog systems, in which signals have continuous values from an infinite set
- **Digital systems** process signals that take only two values: '0' and '1' (Low and High)

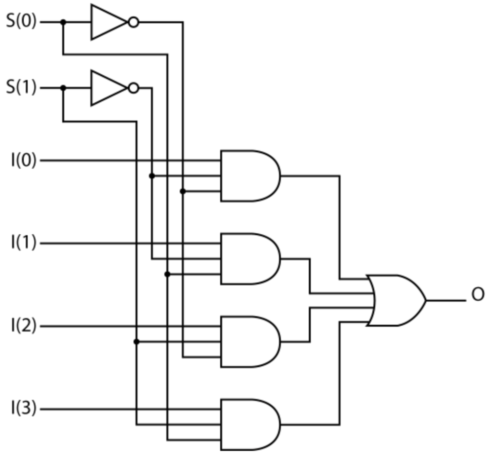




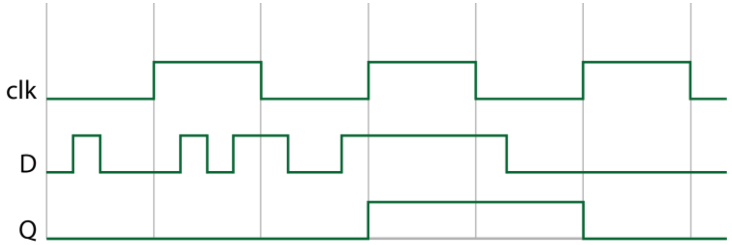


Representation

Schematic diagram



Timing diagram



Boolean equation

$$O = \overline{s1 \cdot s0} \cdot \mathrm{in0} + s1 \cdot \overline{s0} \cdot \mathrm{in1} + \overline{s1} \cdot s0 \cdot \mathrm{in2} + s1 \cdot s0 \cdot \mathrm{in3}$$

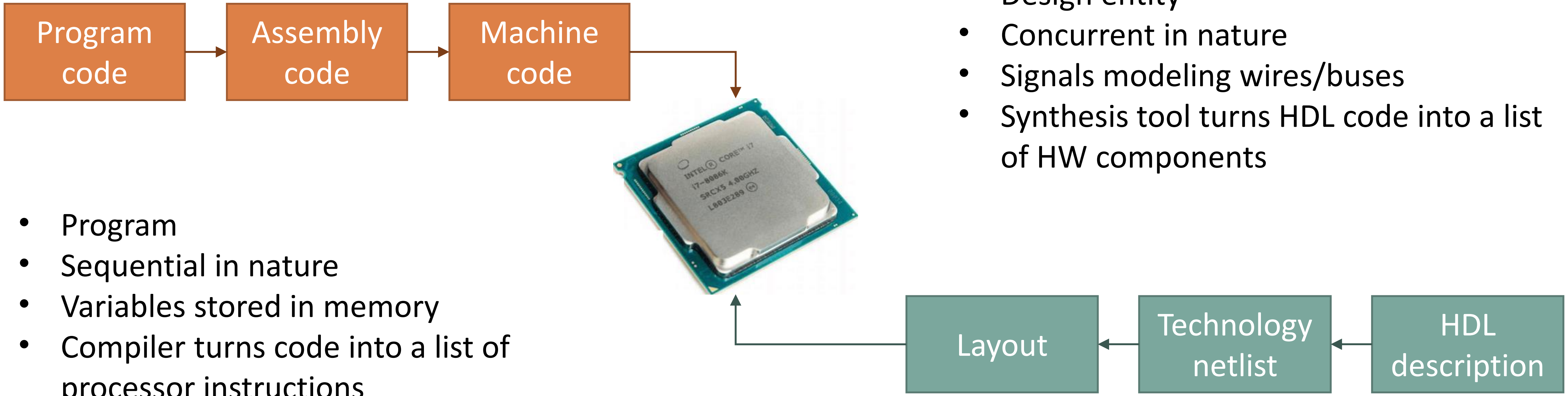
Truth table

s1	s0	in3	in2	in1	in0	O
0	0	x	x	x	1	1
0	1	x	x	1	x	1
1	0	x	1	x	x	1
1	1	1	x	x	x	1

Hardware description language

```
entity MUX is
port (I : in std_logic_vector(3 downto 0);
      S : in std_logic_vector(1 downto 0);
      O : out std_logic);
end MUX;
architecture model of MUX is
begin
  O <= I(0) when S = "00" else
       I(1) when S = "01" else
       I(2) when S = "10" else
       I(3);
end model;
```

HDL vs SW programming languages



- Program
- Sequential in nature
- Variables stored in memory
- Compiler turns code into a list of processor instructions

- Design entity
- Concurrent in nature
- Signals modeling wires/buses
- Synthesis tool turns HDL code into a list of HW components

VHDL: brief history

- VHDL — **V**HSIC **H**ardware **D**escription **L**anguage
  - VHSIC — Very High Speed Integrated Circuit
- Developed in the early 1980s as a research project funded by the U.S. Department of Defense (DoD) for the **documentation and later simulation**
  - Situation in 1980:
    - Multiple design entry methods and hardware description languages in use
    - Limited portability of designs between CAD tools from different vendors
- In 1987 VHDL became IEEE Standard 1076-1987
- In 1993 the standard was revised, new features added
- In 1999 IEEE issued a standard describing an official **subset** of language suitable for **synthesis**
- Latest standard revision in 2008

VHDL: overview

- Technology/vendor independent
- Industrial standard
- Based on ADA language
- Strongly typed
  - Conversion functions are required for type cast
- Case-insensitive

VHDL: synthesizable vs non-synthesizable

- HDL key feature — possibility to describe the design and an environment for simulation and testing
- Synthesizable subset — design description
- Non-synthesizable — testbench and simulation
    - Time-related statements
    - Files
    - Print statements, assertions
    - Dynamic loops

VHDL: entity structure

- Design entity is the basic building block in VHDL
- One file — one entity
- The name of the file should be the same as the entity name

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

Library declaration

```
entity and_gate is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           y : out STD_LOGIC);
end and_gate;
```

Entity declaration (interface)

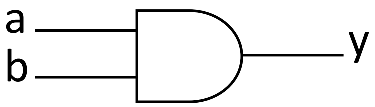
```
architecture model of and_gate is
begin
    y <= a AND b;
end model;
```

Architecture body (functionality)

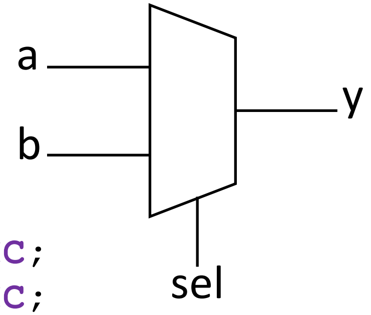
VHDL: entity declaration

- Entity declaration describes the interface of the component, its input and output ports

```
entity <entity_name> is
    Port ( <port_name> : <mode> <data_type>;
           <port_name> : <mode> <data_type>);
end <entity_name>;
```

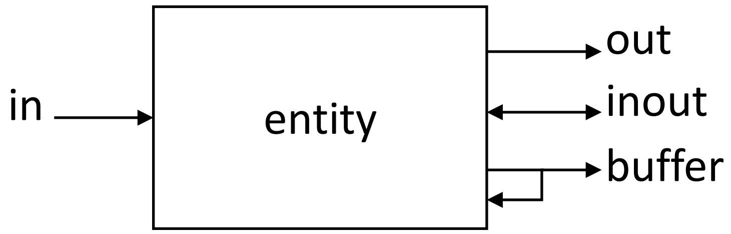


```
entity and_gate is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           y : out STD_LOGIC);
end and_gate;
```



```
entity mux is
    Port ( a   : in STD_LOGIC;
           b   : in STD_LOGIC;
           sel : in STD_LOGIC;
           y   : out STD_LOGIC);
end mux;
```

VHDL: port modes



- Four modes:
  - IN, OUT (unidirectional)
  - INOUT (bidirectional)
  - BUFFER (signal is sent out but can also be read internally)
- INOUT is useful to implement memories
  - That use the same data bus for writing and reading
- The use of BUFFER is **not recommended**
  - A port of type BUFFER can only be connected to ports of the same type
  - Create an internal signal to hold the output value and assign its value to the out port

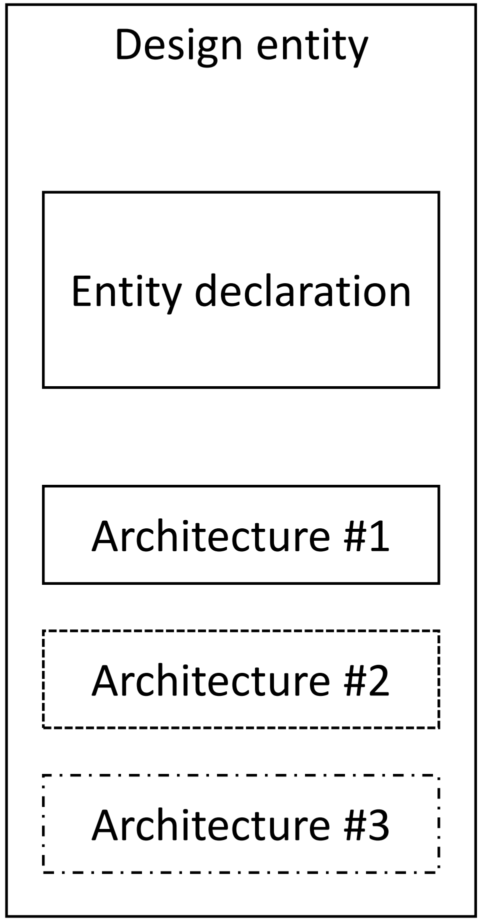
VHDL: architecture

- Architecture describes the functionality of the component

```
architecture <arch_name> of <entity_name> is
    <declarative_part>
begin
    <functionality>
end <arch_name>;
```

- Declarative part is used to declare internal signals, custom data types, used components
- Functionality can be described using
  - Concurrent assignment statements
  - Interconnected components
  - Sequential assignment statements (in a process)

VHDL: design entity



- One entity can have several architectures
- It is useful if one wants to compare different implementations of the same design

VHDL: objects

- The main object in VHDL is a **signal**
  - Model physical wires (buses) for communications or physical storage of information

```
signal <name>: <type> [range] [:= default_value];
```

- Variable is a programming construct to model temporary storage
  - Used only inside processes

```
variable <name>: <type> [range] [:= default_value];
```

- Constant is an object whose value cannot be changed

```
constant <name>: <type> := <value>;
```

VHDL: data types

- INTEGER
- REAL
  - Floating-point
  - Limited synthesis support
- CHARACTER, STRING
- BOOLEAN
- TIME
  - Not synthesizable

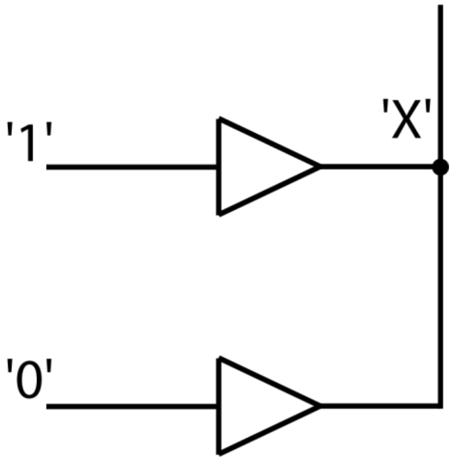
VHDL: data types

- BIT
  - Two possible values: '0' and '1'
- STD_LOGIC
  - Nine possible values: '0', '1', 'U', 'X', 'L', 'H', 'W', 'Z', '–'
  - Better model actual hardware
  - Industry standard for ports
  - 'U', 'X', 'L', 'H', 'W' are only used during **simulation**
  - Only '0', '1', 'Z', '–' are **synthesizable**
- UNSIGNED, SIGNED
  - Based on STD_LOGIC
  - Numeric types

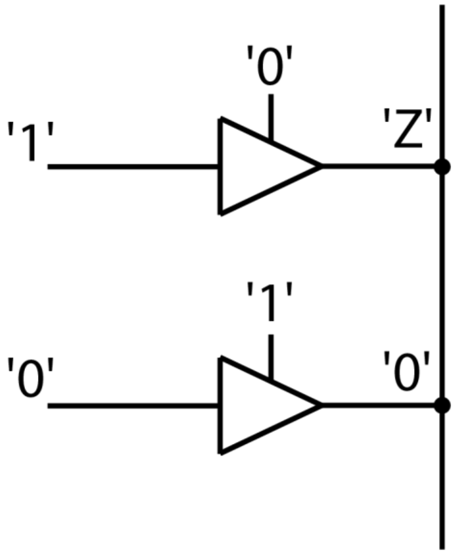
std_logic values	
'0'	LOW
'1'	HIGH
'U'	Uninitialized
'X'	Unknown
'L'	Weak LOW
'H'	Weak HIGH
'W'	Weak unknown
'Z'	High impedance
'–'	Don’t care

VHDL: std_logic

- 'X' means contention on the bus



- High impedance 'Z' means that there is no current flow through the wire
- Used in tri-state buffers



VHDL: wires and buses

- Single-bit signal (wire)

```
signal x: std_logic := '0';
```

- Multi-bit signal (bus)
  - Size is defined as a range
  - Little-endian order is preferred

```
signal y: std_logic_vector(3 downto 0) := "0001";  -- little-endian order
signal z: std_logic_vector(0 to 3) := "0001";      -- big-endian order
```

VHDL: operators

Operator type	Operators	Supported data types
Assignment	<=, :=	All
Logical	NOT, AND, NAND, OR, NOR, XOR, XNOR	bit, bit_vector, boolean, std_logic, std_logic_vector, (un)signed*
Arithmetic	+, –, *, /, **, ABS, REM, MOD	integer, (un)signed, std_logic_vector**
Comparison	=, /=, >, <, >=, <=	bit, bit_vector, boolean, integer, string, (un)signed*, std_logic_vector**
Shift***	SLL, SRL, SLA, SRA, ROL, ROR	bit_vector, std_logic_vector**, (un)signed*
Concatenation	&	bit_vector, string, std_logic_vector, (un)signed*

* Requires package numeric_std

** Requires package std_logic_(un)signed (non-standard)

*** Recommended approach for shifting data is with the concatenation operator

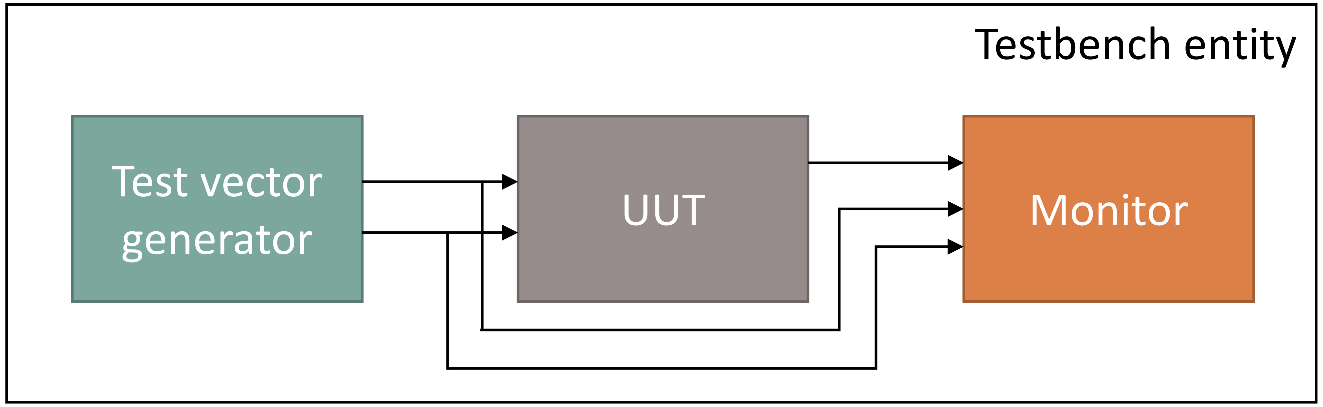
VHDL: basic identifiers

- Consist of Latin letters (a…z), underscores (_), and digits (0…9)
- Must start with a letter
- Cannot have two consecutive underscores
- Cannot end with an underscore
- Special characters are not allowed

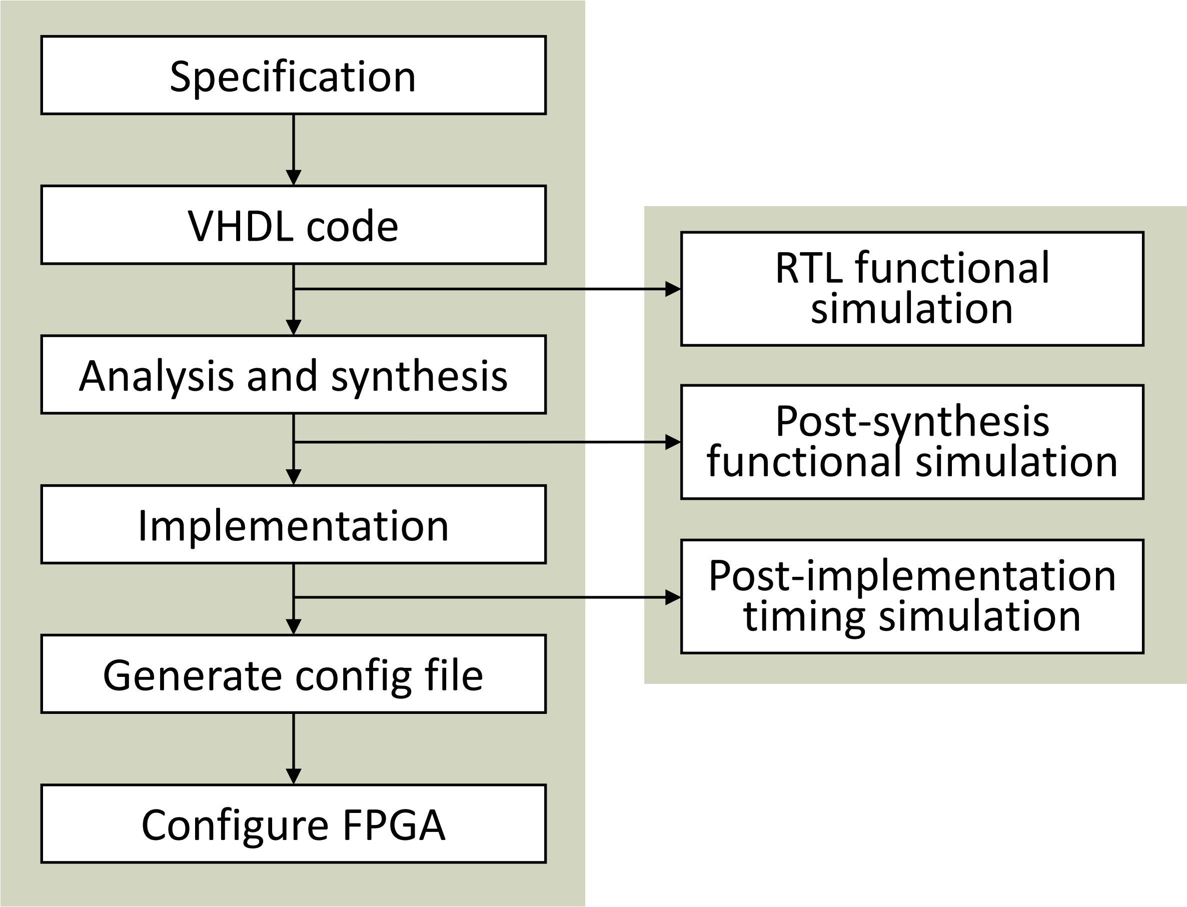
```
Comp_1, LED_OUT, counter         -- legal
_comp_, reg-01, 2bitAdder        -- illegal
```

Testbench

- When a design is implemented, it should be simulated and tested
- In order to simulate the design, we need a testbench
- VHDL can be used both for describing hardware and the environment
  - Testbench is also written in VHDL
- During the simulation and testing, design inputs are driven by test vectors, and outputs are monitored and checked



Design flow for FPGA


Specification
VHDL code
Analysis and synthesis
Implementation
Generate config file
Configure FPGA
RTL functional simulation
Post-synthesis functional simulation
Post-implementation timing simulation

For labs 1–5, functional simulation is enough

For lab 6, timing simulation might be necessary

Coding guide

- Use the coding style suggested in the guide in Moodle
  - For readability and proper synthesis results
- Based on
  - Xilinx coding guidelines
  - OpenCores coding guidelines
  - Textbooks

> “The task of synthesis tool is to analyze VHDL description and infer what hardware elements are represented and how they are connected. A tool cannot infer hardware from any arbitrarily written VHDL model. Instead, we need to write models in a synthesis style that is recognized by the tool.”
>
> -- *The Designer’s Guide to VHDL*, P. J. Ashenden