POWER & ENERGY CONVERSION LABORATORY

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Code Structure
Library, Entity, Architecture

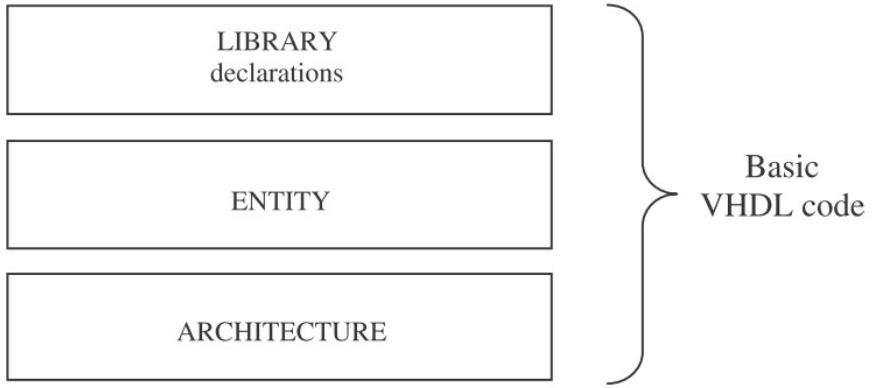
Definition, operators, functions, procedures components, constans, types.

VHDL
VHSIC HARDWARE DESCRIPTION LANGUAGE

Code Structure
Design Flow

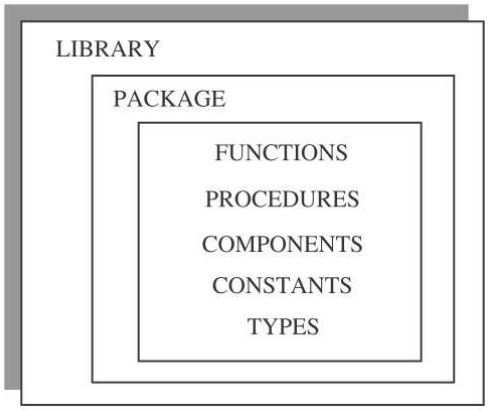
```
-------------------------------------
 LIBRARY ieee;
 USE ieee.std_logic_1164.all;
 -------------------------------------
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
s <= a XOR b XOR cin;
cout <= (a AND b) OR (a AND cin) OR
       (b AND cin);
END dataflow;
```

- A minimum standalone code of VHDL is composed of **Library**, **Entity** and **Architecture**.
  - **LIBRARY** declarations: Contains a list of all libraries to be used in the design.
  - **ENTITY**: Specifies the I/O pins of the circuit.
  - **ARCHITECTURE:** Contains the VHDL code proper, which describes how the circuit should behave (function).



Code Structure

Library



A **Library** is structured based on Functions, procedures or components which are placed inside ***PACKAGES*** and then compiled into the destination library.

LIBRARY declarations: Contains a list of all libraries to be used in the design.

Syntax:

```
LIBRARY library_name;
USE library_name.package_name.package_parts;
```

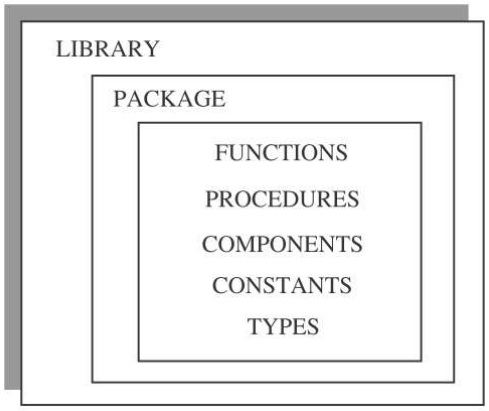
Libraries usually required:

```
LIBRARY ieee;                    -- A semi-colon (;) indicates
USE ieee.std_logic_1164.all;     -- the end of a statement or

LIBRARY std;                     -- declaration, while a double
USE std.standard.all;            -- dash (--) indicates a comment.

LIBRARY work;
USE work.all;
```

Note that libraries *std* and *work* are available by default and therefore there is no need to declare them.

Code Structure

Library



A **Library** is structured based on Functions, procedures or components which are placed inside ***PACKAGES*** and then compiled into the destination library.

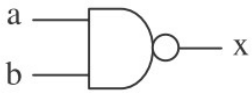
- **std** is a resource library (data types, text i/o, etc.) for the VHDL design environment;
- **work** library is where we save our design (the .vhd file, plus all files created by the compiler, simulator, etc.)

In addition, we have several packages from ieee library as:

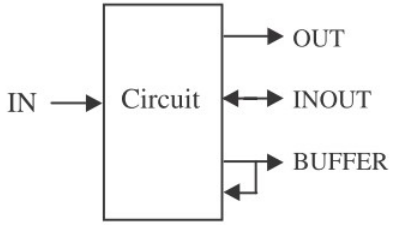
- **std_logic_1164**: Specifies a multi-level logic system. The STD_LOGIC (8 levels) and STD_ULOGIC (9 levels) multi-valued logic systems.
- **std_logic_arith**: Specifies the SIGNED and UNSIGNED data types and related arithmetic and comparison operations. It also contains several data conversion functions, which allow one type to be converted into another: conv_integer(p), conv_unsigned(p, b), conv_signed(p, b), conv_std_logic_vector(p, b).
- **std_logic_signed**: Contains functions that allow operations with STD_LOGIC_VECTOR data to be performed as if the data were of type SIGNED.
- **std_logic_unsigned**: Contains functions that allow operations with STD_LOGIC_VECTOR data to be performed as if the data were of type UNSIGNED

Code Structure

Entity



NAND Gate




```
ENTITY nand_gate IS
   PORT (a, b : IN BIT;
         x : OUT BIT);
END nand_gate;
```

ENTITY is a list with specifications of all input and output pins (PORTS) of the circuit.
Syntax:

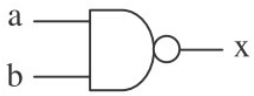
```
ENTITY entity_name IS
PORT (
port_name : signal_mode signal_type;
port_name : signal_mode signal_type;
...);
END entity_name;
```

Signal Mode can be IN, OUT, INOUT (bidirectional) or BUFFER (for output signals that are used (read) internally).

Signal Type can be BIT, STD_LOGIC, INTEGER, etc. (we see this later).

Entity Name can be any name, except VHDL. It is the name of your entity **and .vhd file**

Code Structure
Architecture



NAND Gate

ARCHITECTURE is the description of the functionality of the circuit.
Syntax (it has two parts, declarations and code)

```
ARCHITECTURE architecture_name OF entity_name IS
[declarations]
BEGIN
(code)
END architecture_name;
```

- The declarative part (optional) declares signals and constants (among others).
- The code part define the function of the code.

```
ARCHITECTURE myarch OF nand_gate IS
BEGIN
   x <= a NAND b;
END myarch;
```

´<=´Assign the result of “a AND b” to x.

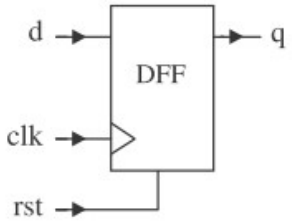
There are three different modelling styles for architecture body:
Data flow : The circuit is described using concurrent statements
Behavioral : The circuit is described using sequential statements
Structural : The circuit is described using different interconnected components

Mixed style is also allowed, using two or the three styles.

Code Structure

Exercises –FlipFlop Source Code

Implement a D-type flip-flop (DFF) and simulate its behavior to validate its code.



- The flip-flop is triggered at rising Edge of the clock signal clk
- It possesses an asynchronous reset
- If reset is high, output must be low regardless of clk.
- If reset is low, the input d is copied to the output q at the clock raising

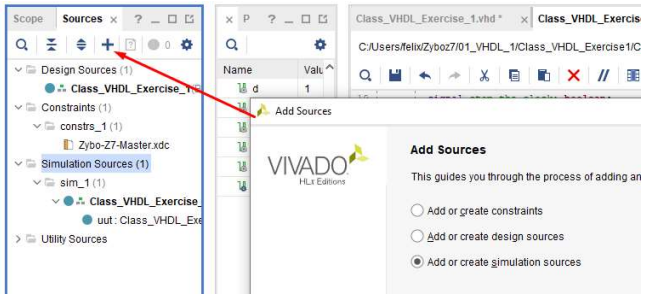
```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Class_VHDL_Exercise_1 is
    Port ( d, clk, rst : in STD_LOGIC;
           q : out STD_LOGIC);
end Class_VHDL_Exercise_1;

architecture Behavioral of Class_VHDL_Exercise_1 is

begin
    PROCESS (rst, clk)
    BEGIN
        IF(rst='1') THEN
            q <= '0';
        ELSIF (clk'EVENT AND clk='1') THEN
            q <= d;
        END IF;
    END PROCESS;

end Behavioral;
```

Code Structure

Exercises-FlipFlop Test Bench For Simulation

In order to simulate our design, we need to create a simulation testbench.

For doing that, we need to add a new simulation source here:



Once created, you need to copy the text on the right. That will create the environment and conditions of the simulation.

For creating the code use this website.

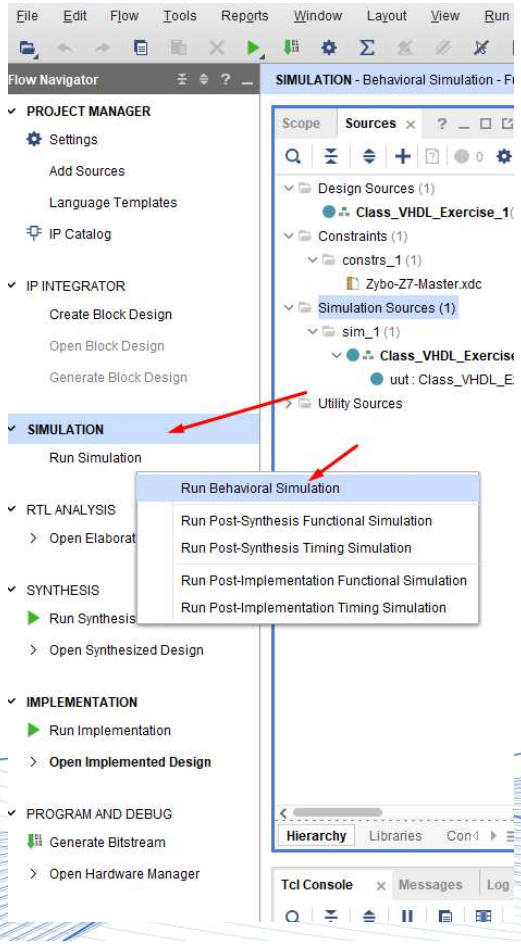
A future complete lecture will be dedicated to create testbenchs!
No worries for now.

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity Class_VHDL_Exercise_1_tb is
end;

architecture bench of Class_VHDL_Exercise_1_tb is

  component Class_VHDL_Exercise_1
      Port ( d, clk, rst : in STD_LOGIC;
             q : out STD_LOGIC);
  end component;

  signal d, clk, rst: STD_LOGIC;
  signal q: STD_LOGIC;

  constant clock_period: time := 10 ns;
  signal stop_the_clock: boolean;

begin

  uut: Class_VHDL_Exercise_1 port map ( d   => d,
                                        clk => clk,
                                        rst => rst,
                                        q   => q );

  stimulus: process
  begin

    -- Put initialisation code here
    d<='1';
    wait for 50 ns;
    d<='1';
    wait for 50 ns;
    -- Put test bench stimulus code here
    stop_the_clock <= true;
    wait;
  end process;

  clocking: process
  begin
    while not stop_the_clock loop
      clk <= '0', '1' after clock_period / 2;
      wait for clock_period;
    end loop;
    wait;
  end process;

end;
```

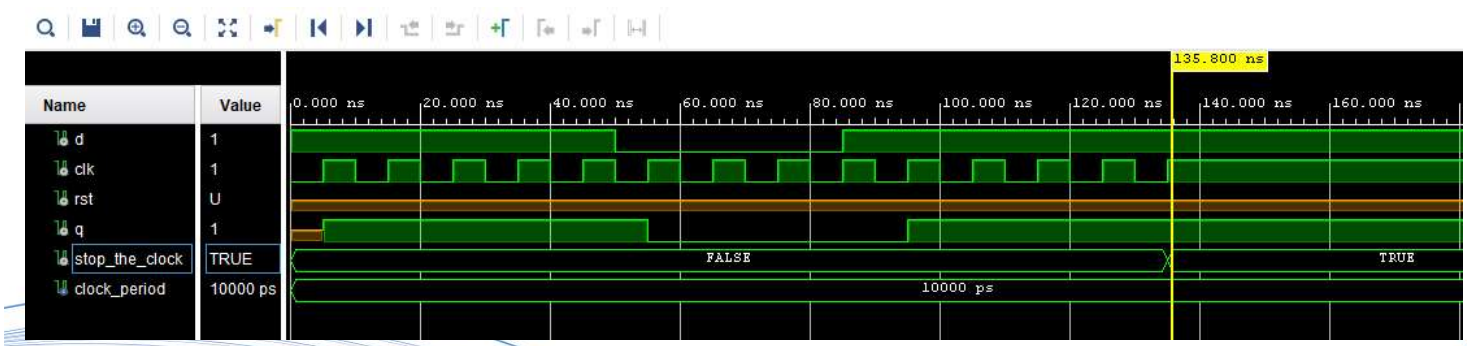
Code Structure

Exercises-Simulation

Once the simulation file is ready, we can run the simulation:




```vhdl
stimulus: process
begin

  -- Put initialisation code here
  d<='1';
  wait for 50 ns;
  d<='0';
  wait for 35 ns;
  d<='1';
  wait for 50 ns;
  -- Put test bench stimulus code here
  stop_the_clock <= true;
  wait;
end process;
```



END-Code Structure
Library, Entity, Architecture

Definition, operators, functions, procedures components, constans, types.

VHDL
VHSIC HARDWARE DESCRIPTION LANGUAGE