Digital Design Using VHDL
Using Xilinx’s Tool for Synthesis and ModelSim for Verification

Ahmed Abu-Hajar, Ph.D.
abuhajar@digitavid.net
Digitavid, Inc
San Jose, CA
Session One Outline

- Introducing VHDL and Top-Down Design
- Our First VHDL code
- Introducing Xilinx Project Navigator
- Introducing ModelSim
Introduction to VHDL

- **VHDL**: Very Large Scale Integrated Circuit Hardware Description Language
- Initially sponsored by DOD then standardized by IEEE in 1987 and was improved in 1993.
- VHDL is used for **Synthesis** to construct and implement a design on silicon.
- VHDL is used for **Simulation** to imitate real world scenarios for verification.

Sum of product Example

\[ f = ab + cd \]
Types of HDL Designs

- **Structural Level Design**: Detailed description of the interconnection between components within a module.

- **Behavioral Level Design**: Provides behavioral description of the module
  - Three types:
    - **Algorithmic**: procedural to model the behavior of the module
    - **Data Flow**: Describe how data moving between registers.
    - **Mixed**: combines both.
Top-Down Design

- Very efficient
- Utilizes library reuse
Our First VHDL Code

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity my_first_code is
    Port ( a : in std_logic;
          b : in std_logic;
          f : out std_logic);
end my_first_code;
architecture Behavioral of my_first_code is
begin
    f <= (a and b) or (c and d);
end Behavioral;
```

\[ f = ab + cd \]
Dissecting our First VHDL Code

- **IEEE library** is pre-defined library with many packages. Here we are using the std_logic_1164
- Other predefined libraries will be used
- You may create your own libraries
- We tend to use libraries supplied by third party vendors

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

entity my_first_code is
  Port ( a : in std_logic;
        b : in std_logic;
        f : out std_logic);
end my_first_code;

architecture Behavioral of my_first_code is
begin
  f <= a and b;
end Behavioral;
```
Dissecting our First VHDL Code

Entity indicates the beginning of the entity. (Required)

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

```
entity entity_name is
  generic ( passed parameters);
  port ( inputs and outputs);
  Begin
    -- only if statements used
    statements
    -- passive statements
    -- (generally not used)
  end entity_name;
```

End indicates the end of this entity. (Optional)

```
my_first_code
```

Port defines inputs and output ports (pins) of this entity. (Optional)

```
entity my_first_code is
  Port ( a : in std_logic;
         b : in std_logic;
         f : out std_logic);
  end my_first_code;
```

“Is” is a keyword indicates the beginning of the content of entity “in this case”. (Required)

End indicates the end of this entity. (Optional)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity my_first_code is
  Port ( a : in std_logic;
         b : in std_logic;
         f : out std_logic);
  end my_first_code;
```

Generic passes parameters from local higher level modules and it uses them as generic parameters. Handy for delay Modeling. Why? (Optional)

```
entity my_first_code is
  Port ( a : in std_logic;
         b : in std_logic;
         f : out std_logic);
  end my_first_code;
```

```
More on Entity (Example)

Created an entity called `my_and_3_inputs`.

The port has three inputs “in” and one output “out”.

Ports may be defined as:

1. In
2. Out
3. inout

Two generic parameters `Time_delay1` & `Time_delay2` of data type `TIME` and set equal to 25ns and 10ns.

Most basic data type “bit” has binary values “0” and “1”. Other data types defined by IEEE standard such as `std_logic`, and `std_logic_vector`, which have seven values.
Dissecting our First VHDL Code Architecture Syntax

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

entity my_first_code is
  Port ( a : in std_logic;
         b : in std_logic;
         f : out std_logic);
end

architecture Behavioral of my_first_code is
begin
  f <= (a and b) or (c and d);
end Behavioral;
```

- **Library**: `library IEEE;` indicates the beginning of the architecture.
- **Use**: `use IEEE.STD_LOGIC_1164.ALL;` binds the library to the entity.
- **Entity**: `entity my_first_code is` defines the entity.
  - `Port` declarations: `Port ( a : in std_logic;
                                b : in std_logic;
                                f : out std_logic);` define the inputs and output of the entity.
- **Architecture**: `architecture Behavioral of my_first_code is` begins the architecture.
  - `begin`: Starts the block of statements.
    - `f <= (a and b) or (c and d);` is a signal assignment.
- **End**: `end Behavioral;` indicates the end of the architecture.
More on Architecture (Example)

Architecture srtuct of my_and_three_inputs is
signal W: bit;
begin
  W <= a and b;
  f  <= W and c;
End struct;

Pay attention to ;, <= and )
Xilinx Example: Create New Project

- We would like to design a full adder using Xilinx’s Project Navigator.
- Open Project Navigator
  - Create new Project (Workshop_Day_One)
Xilinx Example: Create New Project

Open Project Navigator

- FILE ➔ new Project
- Enter project name: Workshop_Day_One
Xilinx Example: Create New Project

Open Project Navigator

- FILE → new Project
- Enter project name: Workshop_Day_One
- Simulator select ModelSimXE VHDL
- The preferred language select VHDL
Xilinx Example: Create New Project

Open Project Navigator
- FILE → new Project
- Enter project name: Workshop_Day_One
- Simulator select ModelSimXE
- The preferred language select VHDL
- Source name enter New Source
- Select VHDL Module
- Enter module name MY_Full_Adder
Xilinx Example: Create New Project

Source

Editor

Process

Console
In the architecture section enter this code:

```vhdl
architecture Behavioral of My_Full_Adder is
  signal W, X, Y, Z : std_logic;
begin
  -- Sum
  W <= a xor b;
  sum <= W xor cin;
  --
  X <= a and b;
  Y <= a and cin;
  Z <= b and cin;
  -- Cout
  Cout <= X or Y or Z;
end Behavioral;
```
Xilinx Example: Create Test Bench

- Go to process window and double click \texttt{synthesize-XST}
- If your code is error free \texttt{synthesize-XST} shows a check sign, else it shows an X sign
- Expand the \texttt{synthesize} and click on the View RTL Schematic
- Project Navigator have synthesized the code as shown
Xilinx Example: Create Test Bench

- Go to Project → New Source
- Select VHDL Test Bench
- In the file name enter Testing_my_full_adder
- Select MY_FULL_Adder to be the associated source
- You are ready to verify your design
Xilinx Example: Verification

- ModelSim is a simulation tool used to verify design.
- Test bench files used to verify designs.
- New terms:
  - **Module Under Test** (MUT) is the module needed to be tested.
  - MUT is a component embedded within the test bench
  - **Input Test Vector** is sent to MUT
  - The **Output Test Vector** is verified against ideal output
Xilinx Example: Test Bench

- **Component declaration in the test bench file for the MUT is needed**
- **In our example the MUT is my_full_adder**

```vhdl
ARCHITECTURE behavior OF Testing_my_full_adder_vhd IS
  -- Component Declaration for the Unit Under Test (UUT)
  COMPONENT My_Full_Adder
    PORT(
      a, b, cin : IN std_logic;
      Sum, Cout : OUT std_logic);
  END COMPONENT;

  -- test vectors (input and output)
  SIGNAL a,b,cin, sum, cout : std_logic := '0';
BEGIN
  -- Architecture Declaration
  ALIGNMENT OF Architecture Loop

  -- Component Declaration
  COMPONENT Declaration Loop

  -- Signal Declaration
  Signal Declaration Loop

  Note: Component Declaration must use the same port name as the original. The component may be called as many times in the body “hierarchical structure”
```
Xilinx Example: Test Bench

- The body of the architecture calls the (MUT) Component
- Assigns the input test vector to the module under test using PORT MAP
- The values of the input test vectors are assigned

BEGIN
  -- Instantiate the Unit Under Test (UUT)
  uut: My_Full_Adder PORT MAP (a => a1, b => b, cin => cin, Sum => Sum, Cout => Cout);
  -- Test input vector assignment
  a1 <= '1' after 10ns, '1' after 20ns;
  b <= '1' after 5ns, '0' after 10ns, '1' after 15ns, '0' after 20ns;
  cin <= '1' after 2.5ns, '0' after 5ns, '1' after 7.5ns, '0' after 10ns,'1' after 12.5ns, '0' after 15ns, '1' after 17.5ns, '0' after 20ns;
END;
Xilinx Example: Test Bench

- Save your Test bench File
- Go to Source For → Behavioral Simulation
  "Behavioral is the architecture name"
- Double click on ModelSim Simulator and then on Simulate Behavioral Model
- ModelSim is automatically loaded.
- If no errors were found in your test bench file, the simulation graph is already loaded