INTRODUCTION TO FPGA PROGRAMMING

LESSON 02: VHDL FUNDAMENTALS

Dr. Davide Cieri[1]

[1]Max-Planck-Institut für Physik, Munich

August 2024

MAX-PLANCK-INSTITUT FÜR PHYSIK

VHDL BASICS

- VHDL is a **strongly typed language** (like C)
  - All objects must have a type and contain values of that type
  - Operations between objects must be of the same type

```
signal a : std_logic; -- std_logic is the type of the signal named a
signal b : integer;   -- integer is the type of the signal named a
```

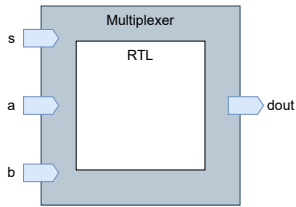
- VHDL is **case insensitive**

```
-- These two lines are equivalent
signal a : std_logic;
SIGNAL A : STD_LOGIC;
```

- Every VHDL statement ends with a **semicolumn ;**
- Not sensitive to white spaces between statements
- **Empty lines** ignored by compiler
- **Comments** start with two dashes −−

VHDL BUILDING BLOCKS

- `library` and `package` import data types and functions from other files
- The `entity` declares the digital block and defines the interface with the outside world through `port` statements.
- The `architecture` implements the functionality of the block (inner working).




```
library ieee ;
use ieee.std_logic_1164.all ;

entity multiplexer is
Port( a    : in  STD_LOGIC;
      b    : in  STD_LOGIC;
      s    : in  STD_LOGIC;
      dout : out STD_LOGIC);
end multiplexer;

architecture RTL of multiplexer is
begin
  out <= a when s = '0' else b;
end RTL;
```

VHDL TYPES

- **Built-in data types**
  - `integer`: whole numbers
  - `real`: floating point numbers
  - `time`: used to specify delays - important in simulation
  - `bit`: scalar 1-bit signal (allowed values `'0'` or `'1'`)
  - `bit_vector`: multiple bits (buses) signals
  - `boolean`: boolean number (`true` or `false`)
- Built-in data-types are not enough to describe an actual circuit
  - E.g. `bit` can not describe high-impedance state
- Types can be extended using external libraries (IEEE or users)

LIBRARY STATEMENTS

- To import definitions (types, functions, etc...) from a library, use the `library` statement

```
library <library_name>;
```

- In addition to entity and architectures, also packages can be stored in a library
- Packages are used to define types or functions, that can be used in multiple part of the design (more in another lecture)
- The `use` statement is used to include a package from a library

```
use <library_name>.<package_name>.all; -- "all" import all objects in the package
```

-

THE IEEE LIBRARY

- One of the most common library used in VHDL designs
- Despite being called `IEEE` is actually property of a company called Synopsys
- Two main packages to import

```
library IEEE;
use IEEE.std_logic_1164.all; -- Extended logic types
use IEEE.numeric_std.all;    -- Numeric data types and arithmetic functions
```

IEEE PACKAGES

- Two extended VHDL logic types
  - `std_logic` for single bit signals
  - `std_logic_vector` for multiple-bit signals (buses)
  - `signed`: vector representation of signed binary numbers
  - `unsigned`: vector representation of unsigned binary numbers
- Additional logic values
  - `'U'`: unitialised
  - `'X'`: unknown logic value
  - `'Z'`: high-impedance
  - `'W'`: weak signal
  - `'L'`: weak low
  - `'H'`: weak high
  - `'-'`: don't care

BUSES AND ENDIANESS

- The ordering of bits within a bus is fundamental to know the Most Significant Bit (MSB) and Least Significant Bit (LSB)
- Two options: big endian or little endian

```
signal be_signal : std_logic_vector(11 downto 0); -- Big Endian. (11) is the MSB
signal le_signal : std_logic_vector(0 to 11);     -- Little Endian. (0) is the MSB
```

- Tip: define always your vectors using `downto` so the binary value follow the two's complement representation (unsigned and signed).
- MSB in a signed vector represents the minus.

STRAIGHT BINARY CODE

- Binary strings interpreted as power of 2 integer numbers

Unsigned

```
"000"; -- 0
"001"; -- 1
"010"; -- 2
"011"; -- 3
"100"; -- 4
"101"; -- 5
"110"; -- 6
"111"; -- 7
```

Signed

```
"000" -- 0
"001" -- 1
"010" -- 2
"011" -- 3
"100" -- -4
"101" -- -3
"110" -- -2
"111" -- -1
```

ENTITY AND ARCHITECTURE SYNTAX

Entity

```
entity MyModule is
-- optional generic list: More in another
    lesson
generic(...);
-- port list
port(...);
```

Architecture

```
architecture rtl of MyModule is
  -- Declarative region local to the
      architecture (e.g. local signals)
  begin
  -- Implementation of the architecture (
      Functionalities)
  end rtl;
```

PORTS

```
entity MyModule is
port(
  -- <port_name> : <port mode> <port type>,
  data_in    : in  std_logic; -- input port
  data_out   : out std_logic; -- output port
  data_inout : inout std_logic -- bidirectional port
);
```

- Common port modes: `in`, `out` and `inout`
- Reading value from an out port is allowed only from VHDL-2008
- N.B. No semicolumn (;) after the last port definition

SIGNAL DECLARATION

```
-- signal <signal_name> : <signal type> := <default_value>
signal A : std_logic := '0';
signal B : std_logic_vector(11 downto 0) := (others => '0'); -- initialises all bits to
    zero
```

- Signals must be defined inside the entity's architecture
- Optionally declare an initial value
  - If no initial value, derived from type definition
- Signal assignement is done with the <= operator, in the architecture implementation space

```
A <= B;
```

ADVANCED SIGNAL ASSIGNEMENT

Multiple Assignment[1]

```
A <= '1' after 5ns, '0' after 15 ns;
```

Conditional Signal Assignment (implicit if)

```
A <= '1' when B='1' else '0';
```

Selected Signal Assignment (implicit case)

```
with sel select
A <= B when "00",
C when "01",
D when "10",
E when "11";
```

[1] `after 5ns` is not a syntesisable statement, since the FPGA does not have a concept of time

CONCURRENT EXECUTION

- Signal assignments are evaluated concurrently to each VHDL statement, when placed outside of a process block (Sequential execution, more in another lesson)
- Evaluated at the same time. The order in which they are written does not count
- The following code snippets are equivalent.

```
architecture Behavioral of MyModule is

begin
  S <= A;
  B <= C;
end Behavioral;
```

```
architecture Behavioral of MyModule is

begin
  B <= C;
  S <= A;
end Behavioral;
```

MODULE INSTANTIATION

- You can instantiate other HDL modules inside your architecture
- Two options available:
  - **Component Instantiation**
  - **Entity or Direct Instantiation**
- As an example, we want to instantiate the module `mux` inside our module `MyModule`.

```
entity mux is
Port( a    : in  STD_LOGIC;
      b    : in  STD_LOGIC;
      s    : in  STD_LOGIC;
      dout : out STD_LOGIC);
end mux;
```

COMPONENT INSTANTIATION

1. Declare the component in the declarative region of your architecture
2. Instantiate the component in the implementation region

```
-- Component Declaration
component <component name>
port (
  <port 0 name> : <mode> <type>;
  ...
  <port N name> : <mode> <type>
);
end component;
```

```
-- Component Instantiation
<instantiation name> : <component name>
port map (
  <component port 0 name> => <parent
      module signal>,
  ...
  <component port N name> => <parent
      module signal>
);
```

COMPONENT INSTANTIATION EXAMPLE

```
entity MyModule is
Port( a, b, s : in  STD_LOGIC;
      dout    : out STD_LOGIC);
end MyModule;

architecture RTL of MyModule is
  -- Component Declaration
  component mux is
  Port( a    : in  STD_LOGIC;
        b    : in  STD_LOGIC;
        s    : in  STD_LOGIC;
        dout : out STD_LOGIC);
  end component;
begin
  -- Component Instantiation
  mux_inst : mux
  port map (
    a    => a,
    b    => b,
    s    => s,
    dout => dout
  );
end RTL;
```

ENTITY INSTANTIATION

- Valid only if instantiating another VHDL module[2]
- No declaration needed. Instantiate directly in the implementation region.

```
-- Entity Instantiation
<instantiation name> : <library_name>.<component name>
port map (
  <entity port 0 name> => <parent module signal>,
  ...
  <entity port N name> => <parent module signal>
);
```

[2] instantiating a verilog module inside VHDL is possible only with the component syntax.

COMPONENT INSTANTIATION EXAMPLE

```
entity MyModule is
Port( a, b, s : in  STD_LOGIC;
      dout    : out STD_LOGIC);
end MyModule;

architecture RTL of MyModule is
begin
  -- Entity Instantiation
  -- work is the a special keyword in VHDL. It always refers to the same library as the current module.
  -- i.e. mux and MyModule should be in the same library to function
  mux_inst : entity work.mux
  port map (
    a    => a,
    b    => b,
    s    => s,
    dout => dout
  );
end RTL;
```

LESSON 02 REVIEW

VHDL Fundamentals

- VHDL basics
- VHDL Built-in types
- VHDL libraries and IEEE library
- Entity and Architecture
- Ports and Signals
- Concurrent Execution
- Hierarchical models

LAB 03: WIRING SWITCHES TO LEDS