Behavioral VHDL

Last updated 12/17/24

Behavioral VHDL

- Behavioral VHDL describes the behavior of the circuit
  - Concurrent logic
  - Sequential Logic
  - Memories
  - Finite State Machines
  - …

Behavioral VHDL

- Behavioral VHDL file components
  - Header
    - Description of who created the file
    - Description of the purpose of the block
    - Description of inputs and outputs
  - Inclusions
    - Any libraries that will be referenced in the design
  - Entity
    - Formal definition of the inputs and outputs
    - Any generic parameters are defined here also
  - Architecture
    - Internal signal declarations
    - HDL description of the desired functionality
      - Concurrent, sequential logic

Behavioral VHDL

- Behavioral VHDL file components - example

comments begin with 2 dashes
--
continue to the end of the current line

```
----------------------------
--
-- behavioral_vhdl.vhdl
-- Created: 12/17/24
-- By: johnsontimoj
-- For: CPE 1500
--

-- File Overview ---
--
-- This file demonstrates behavioral VHDL file structure

-- File Details ---
----------------------------
```

Header Information

Document
who, when, what and how

Should be sufficient to allow a co-worker to use/modify the design

Behavioral VHDL

- Behavioral VHDL file components - example

```
-- Library inclusions
library IEEE;
use ieee.std_logic_1164.all;
```

Library inclusions

statements end with a semi-colon

Include the basic library definitions

Behavioral VHDL

- Behavioral VHDL file components - example

```
-- Entity definition
entity behavioral_vhdl is
   port(    i_a:  in   std_logic;
            i_b:  in   std_logic;
            i_c:  in   std_logic;
            i_d:  in   std_logic_vector(7 downto 0);
            o_x:  out  std_logic;
            o_y:  out  std_logic_vector(7 downto 0)
   );
end entity;
```

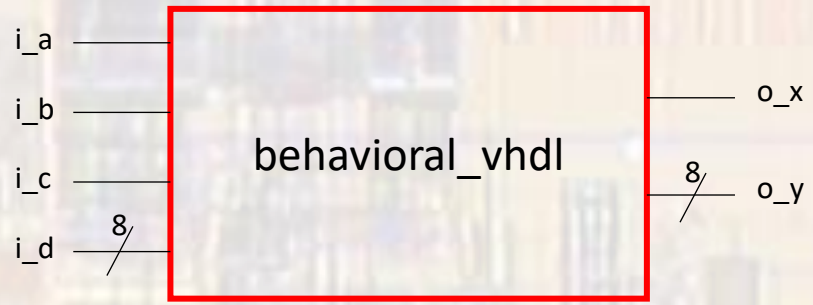
Entity

the entity name must match the file name

port format:
name : direction signal_type ;

directions: in, out, bidir
signal types: std_logic
std_logic_vector(…)

We will prepend ports with
i_ or o_



The Entity is the external view of the block
Ports are the i/o connections of the block
Ports have a direction but no interpretation
Wires or groups of wires

Behavioral VHDL

- Behavioral VHDL file components - example

```
-- Behavioral Architecture Definition
architecture behavioral of behavioral_vhdl is

   -- internal signals
   signal e:   std_logic;
   signal f:   std_logic;
   signal g:   std_logic;
   signal h:   std_logic_vector(7 downto 0);
   signal i:   std_logic_vector(7 downto 0);

   begin
      e <= i_a and i_b;
      f <= i_c nor i_a;
      g <= e xor f;
      o_x <= g or i_a;

      h <= not i_d;
      i <= i_a & i_b & i_c & i_d(6 downto 2);
      o_y <= h and i;

end architecture;
```

Architecture

Internal signals

Architectural Definition (operation)

signal format:
signal name : signal_type ;

signal types: std_logic
std_logic_vector(…)

Before the begin
- Internal signals
- Constants used for readability

After the begin
- HDL for the specific application

signal assignment:
<=
result <= logical statement

the result “takes the value of” …
the result “is assigned the value of” …

Behavioral VHDL

- Behavioral VHDL file components - example

```
----------------------------
--
-- behavioral_vhdl.vhdl
-- Created: 12/17/24
-- By: johnsontimoj
-- For: CPE 1500
--

-- File Overview ---
--
-- This file demonstrates behavioral VHDL file structure

-- File Details ---
-----------------------------

-- Library inclusions
library IEEE;
use ieee.std_logic_1164.all;

-- Entity definition
entity behavioral_vhdl is
   port(    i_a:  in    std_logic;
            i_b:  in    std_logic;
            i_c:  in    std_logic;
            i_d:  in    std_logic_vector(7 downto 0);
            o_x:  out   std_logic;
            o_y:  out   std_logic_vector(7 downto 0)
   );
end entity;

-- Behavioral Architecture Definition
architecture behavioral of behavioral_vhdl is

   -- internal signals
   signal e:   std_logic;
   signal f:   std_logic;
   signal g:   std_logic;
   signal h:   std_logic_vector(7 downto 0);
   signal i:   std_logic_vector(7 downto 0);

   begin
      e <= i_a and i_b;
      f <= i_c nor i_a;
      g <= e xor f;
      o_x <= g or i_a;

      h <= not i_d;
      i <= i_a & i_b & i_c & i_d(6 downto 2);
      o_y <= h and i;

end architecture;
```

Resulting synthesized logic

It may not look like how you would draw it because the tool has a large number of mathematical models (gates) that it can use