

Module 10 - Basic VHDL Tutorial and Exercises For the VeriBest Simulator

Copyright 1995-1999 SCRA

All rights reserved. This information is copyrighted by the SCRA, through its Advanced Technology Institute (ATI), and may only be used for non-commercial educational purposes. Any other use of this information without the express written permission of the ATI is prohibited. Certain parts of this work belong to other copyright holders and are used with their permission. All information contained, may be duplicated for non-commercial educational use only provided this copyright notice and the copyright acknowledgements herein are included. No warranty of any kind is provided or implied, nor is any liability accepted regardless of use.

The United States Government holds “Unlimited Rights” in all data contained herein under Contract F33615-94-C-1457. Such data may be liberally reproduced and disseminated by the Government, in whole or in part, without restriction except as follows: Certain parts of this work to other copyright holders and are used with their permission; This information contained herein may be duplicated only for non-commercial educational use. Any vehicle, in which part or all of this data is incorporated into, shall carry this notice .

See the [RASSP Disclaimer file](#) for additional RASSP Disclaimer, Warranty and Limitation of Liability Information concerning the material, VHDL code and software developed under the RASSP programs or incorporated in RASSP material.

1. Getting Started

- 1.1. For each tutorial module, you will be using sample VHDL files. These files are provided along with these modules. You will also create some VHDL files of your own.
- 1.2. The first VHDL files that you will need are **package.vhdl** and **and2.vhdl**.

2. Examine and compile the code for this lab

- 2.1. Open the file **package.vhdl** using a text editor or a VHDL editing environment. This file defines the enumerated data types that will be used throughout this tutorial.

```
PACKAGE resources IS
    -- user defined enumerated type
    TYPE level IS ('X', '0', '1', 'Z');
    -- type for vectors (buses)
    TYPE level_vector IS ARRAY (NATURAL RANGE <>) OF level;
    -- subtype used for delays
    SUBTYPE delay IS time;
END resources;
```

- 2.2. Compile the VHDL code. Consult your compiler's documentation for more information on choosing compile settings. **package.vhdl** should compile without any errors. Note that compiled VHDL is placed into a directory (or folder) of files called a *library*. Typical default names for VHDL libraries are "work" and "worklib". Once VHDL is successfully compiled into a library, it can be referenced (*used*) by other VHDL files. This capability will be explored further in the Module 11 lab.
- 2.3. Open the file **and2.vhdl** using a text editor or a VHDL editing environment. This file defines an entity and architecture of a two-input AND gate. The input and output ports of the device are defined in the entity declaration. The behavior of the gate is described in the architecture declaration.

```

-----
--          Standard 2 input AND gate example          --
--    RASSP E&F Module # 10 Basic VHDL                --
--    Robert Klenke UVa 19 April 1996                  --
-----

USE work.resources.all;

ENTITY and2 is

    GENERIC(trise : delay := 10 ns;
            tfall : delay := 8 ns);

    PORT(a : IN level;
          b : IN level;
          c : OUT level);

END and2;

ARCHITECTURE behav OF and2 IS

    BEGIN

        one : PROCESS (a,b)

            BEGIN
                IF (a = '1' AND b = '1') THEN
                    c <= '1' AFTER trise;
                ELSIF (a = '0' OR b = '0') THEN
                    c <= '0' AFTER tfall;
                ELSE
                    c <= 'X' AFTER (trise+tfall)/2;
                END IF;
            END PROCESS one;

        END behav;

```

2.4. Compile the VHDL code. **and2.vhdl** should compile without any errors.

2.5. A mechanism for testing the two-input AND gate is needed. A test bench is used for this purpose. A test bench is a VHDL model which (1) contains an instance of the component under test, and (2) exercises the input signals of the component under test. Open the file **and2_test.vhdl**, which contains a test bench for the two-input AND model.

```

-----
-- Standard 2 input AND gate example testbench           --
-- RASSP E&F Module # 10 Basic VHDL                     --
-- Thomas Egolf Georgia Tech 21 May 1998                --
-----
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;
USE work.resources.all;
USE work.and2;

ENTITY testbnch IS
END testbnch;

ARCHITECTURE stimulus OF testbnch IS

COMPONENT and2 IS
  GENERIC(trise : delay;
          tfall : delay);
  PORT(a : IN level;
        b : IN level;
        c : OUT level);
END COMPONENT;

SIGNAL a_sig, b_sig, c_sig : level;

BEGIN
  DUT: and2
    GENERIC MAP (10 ns, 8 ns)
    PORT MAP ( a => a_sig, b => b_sig, c => c_sig );

  STIMULUS1: PROCESS
  BEGIN

    -- Sequential stimulus goes here...
    --
    a_sig <= '0' AFTER 5 ns;
    WAIT FOR 20 ns;
    a_sig <= '1', '0' AFTER 20 ns;
    b_sig <= '1';

    --
    -- Enter more stimulus here...
    --

    WAIT;           -- Suspend simulation
  END PROCESS STIMULUS1;

END stimulus;

```

2.6. Compile the VHDL code. **and2_test.vhdl** should compile without any errors.

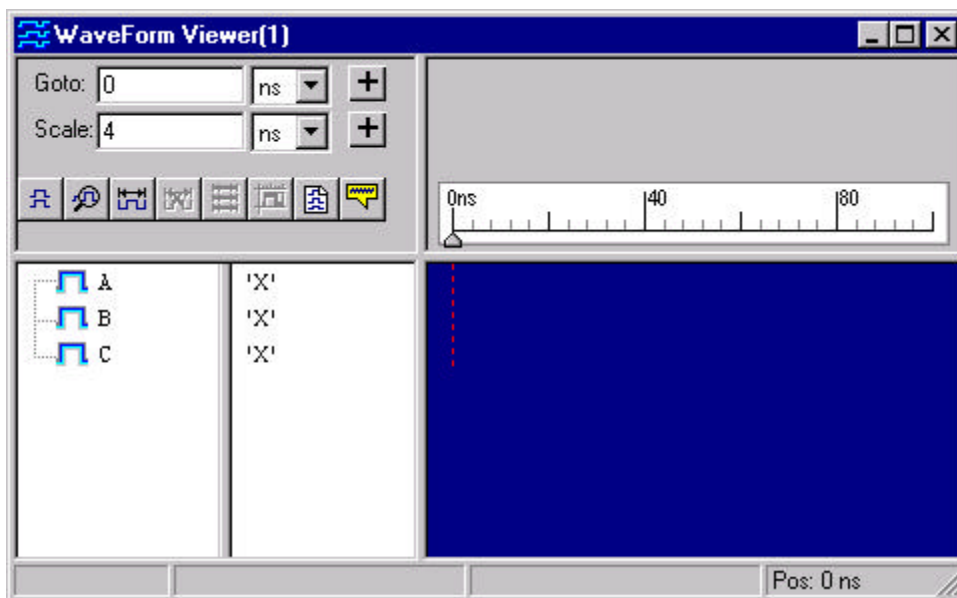
3. Simulate the compiled code

Copyright ©1995-1999 SCRA

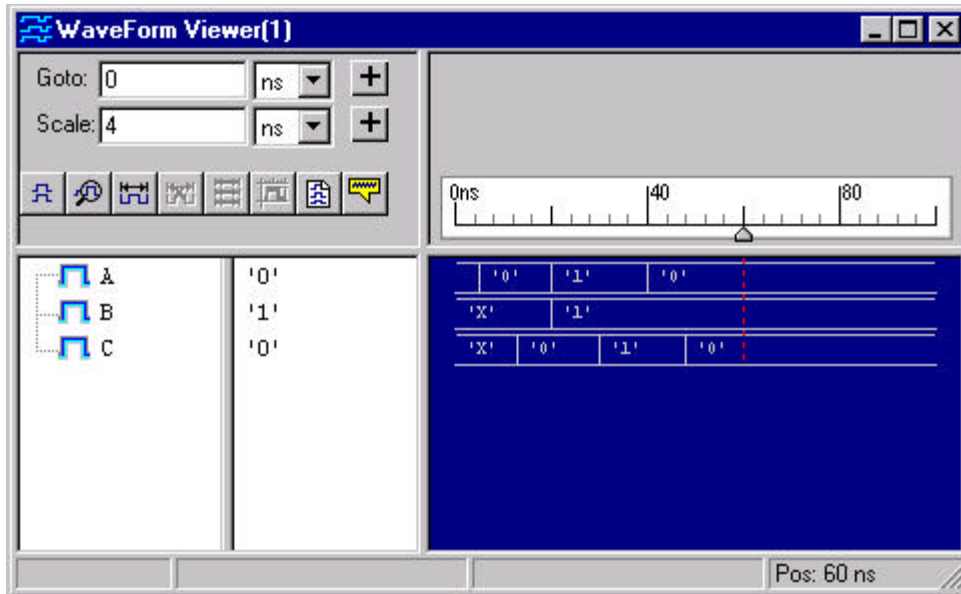
See first page for copyright notice,

Distribution restrictions and disclaimer

- 3.1. Start the VHDL simulator. Consult your simulator's documentation for more details. Note that some additional settings may be required. For example, you may be required to select the entity **testbnch** inside the file **and2_test.vhdl** as the "design root" in order for the simulation to work properly. The simulator should start without any errors.
- 3.2. Open a waveform window, and add the signals "A", "B", and "C" to the window. (Note: some simulation packages may require that the signals be selected before the waveform window is opened.) The purpose of the waveform window is to show the signal values over some period of time, with the signal values shown along the vertical axis and time represented along the horizontal axis. Here is an example of a waveform window:

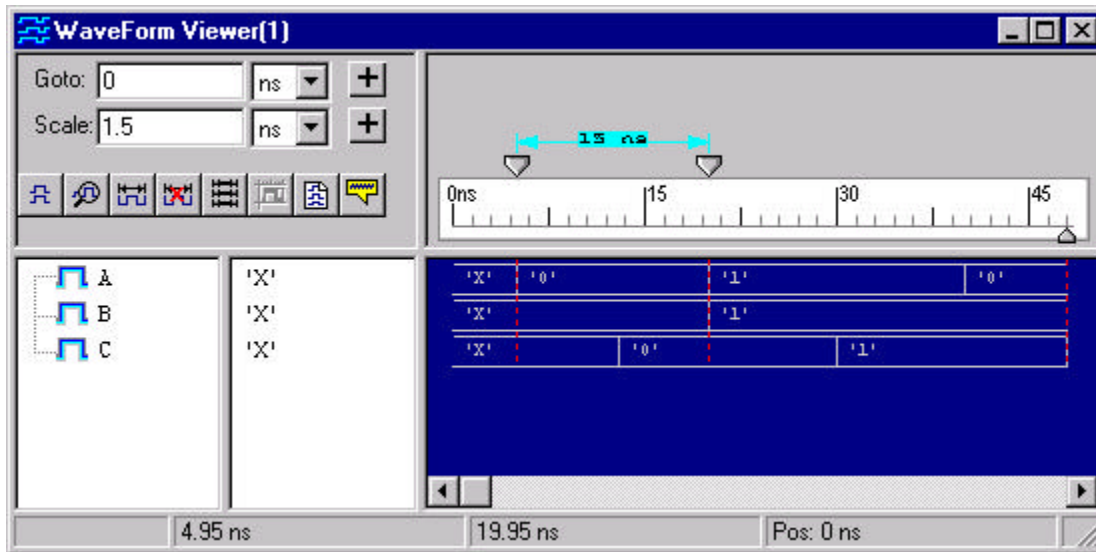


- 3.3. The simulation time begins at zero nanoseconds; no circuit activity has yet taken place. Run the simulator for sufficient time to fully exercise the circuit (at least 50 nanoseconds in this example). Signal waveforms will be shown:



- 3.4. If the simulation results do not entirely fit within the waveform window, adjust the scale (i.e., zoom) so that the entire simulation is shown. Signals can take on the values '0', '1', or 'X' as shown in the waveform window. '0' represents logical low, '1' represents logical high, and 'X' represents an unknown value.
- 3.5. Notice that the test bench drives the input signals "A" and "B". Output signal "C" responds according to the behavioral description of the AND gate.

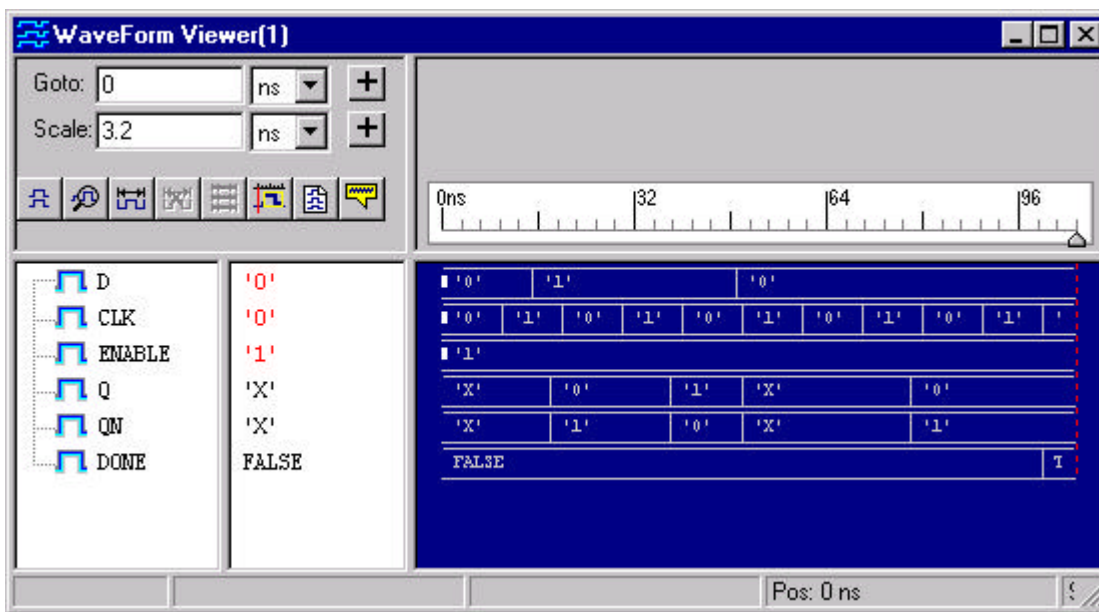
- 3.6. To measure the time between two events, it is useful to have adjustable time cursors. Add time cursors to the waveform window. Consult the simulator's documentation for more information on adding cursors. These cursors are typically moved by dragging them with the mouse. This can be a useful feature for checking the timing characteristics of a circuit, as shown below:



- 3.7. Quit the simulator. Be sure to quit the simulator after each lab exercise.
- 3.8. Modify **and2_test.vhdl** such that signal "B" returns to '0' at 50 ns and signal "A" returns to '1' at 60 ns. Recompile **and2_test.vhdl** and simulate the new testbench.

4. Examine, compile and simulate the D Flip Flop model

- 4.1. Open the file **dff.vhdl** using a text editor or a VHDL editing environment. This file contains a behavioral description of a D Flip Flop.
- 4.2. Compile the VHDL code. **dff.vhdl** should compile without any errors.
- 4.3. A test bench for the flip flop is provided in the file **dff_test.vhdl**. Open and compile this test bench file.
- 4.4. Simulate the D Flip Flop model using the test bench, as before. The resulting waveforms should be similar to the ones shown here:



Module 10 Exercise

Assignment:

In addition to the **and2** gate and **dff** module, you will need the following gates for the designs you will build and simulate in the lab for Module 11:

Entity Name	Architecture Name	Generics	Input Ports	Output Ports	Description
Or2	behav	trise tfall	A b	c	two input or gate
Inv	behav	trise tfall	A	b	single input inverter
Xor2	behav	trise tfall	A b	c	two input exclusive or gate
Mux2	behav	tprop	A b sel	c	Two input multiplexor
Mux4	behav	tprop	A b c d sel(2)	e	Four input multiplexor

Develop a VHDL entity and architecture description for each of these modules and compile them into the default work library. Simulate each design as necessary to ensure proper operation. Hint - copy the VHDL source code from a similar model and modify it to suit your needs; this is often easier than writing the VHDL code from scratch.