

Module 10 - Basic VHDL Tutorial and Exercises For the Mentor Graphics 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. The first VHDL files that you will need are **package.vhdl** and **and2.vhdl**. You should create separate directories for the lab material for each module's labs. Create one now for the Module 10 lab material, e.g:

```
mkdir m10_ex
cd m10_ex
```

- 1.2 Copy the source files for the VHDL that you will compile and simulate from the appropriate source directory, e.g:

```
cp $VHDL_SRC/m10_ex/package.vhdl package.vhdl
cp $VHDL_SRC/m10_ex/and2.vhdl and2.vhdl
```

- 1.3 The Mentor Graphics QuickVHDL simulator needs a work directory for the compiled VHDL files. Create this directory with the appropriate command for the version you are running, e.g:

```
qhlib work
```

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. The specific command may vary depending on the specific Mentor Graphics VHDL simulator, e.g:

```
qvhcom package.vhdl
```

The compiler should display a message similar to the following with no errors:

```
// Compiling for QuickHDL
// QuickHDL qvhcom v8.5_4.5a Mar 28 1996 SunOS 4.1.3
//
// Copyright (c) Mentor Graphics Corporation, 1982-1995, All Rights Reserved.
// UNPUBLISHED, LICENSED SOFTWARE.
```

```
// CONFIDENTIAL AND PROPRIETARY INFORMATION WHICH IS THE
//PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS
//LICENSORS.
//
// Copyright (c) Model Technology Incorporated 1990-1995, All Rights Reserved.
//
-- Loading package standard
-- Compiling package resources
```

- 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 IF;

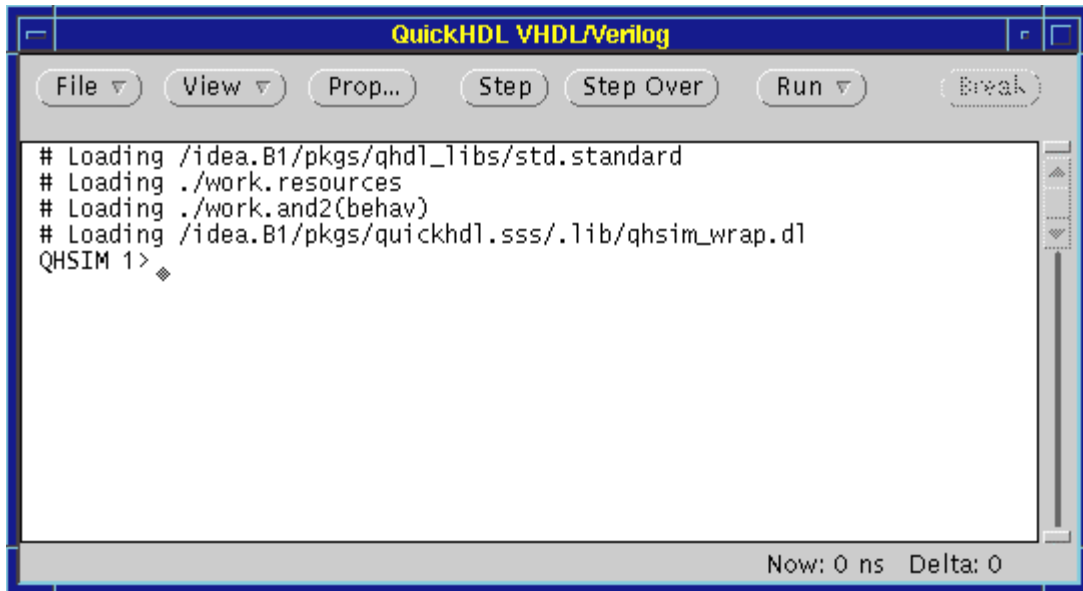
        END PROCESS one;
    END behav;
```

- 2.4 Compile the VHDL code. **and2_test.vhdl** should compile without any errors, e.g:
qvhcom and2_test.vhdl

3 Simulate the compiled code

- 3.1 Start up the Mentor Graphics VHDL simulator. The specific command may vary depending on the version you are using, e.g.:
- ```
qhsim and2
```

This will bring up a window that should look something like this:

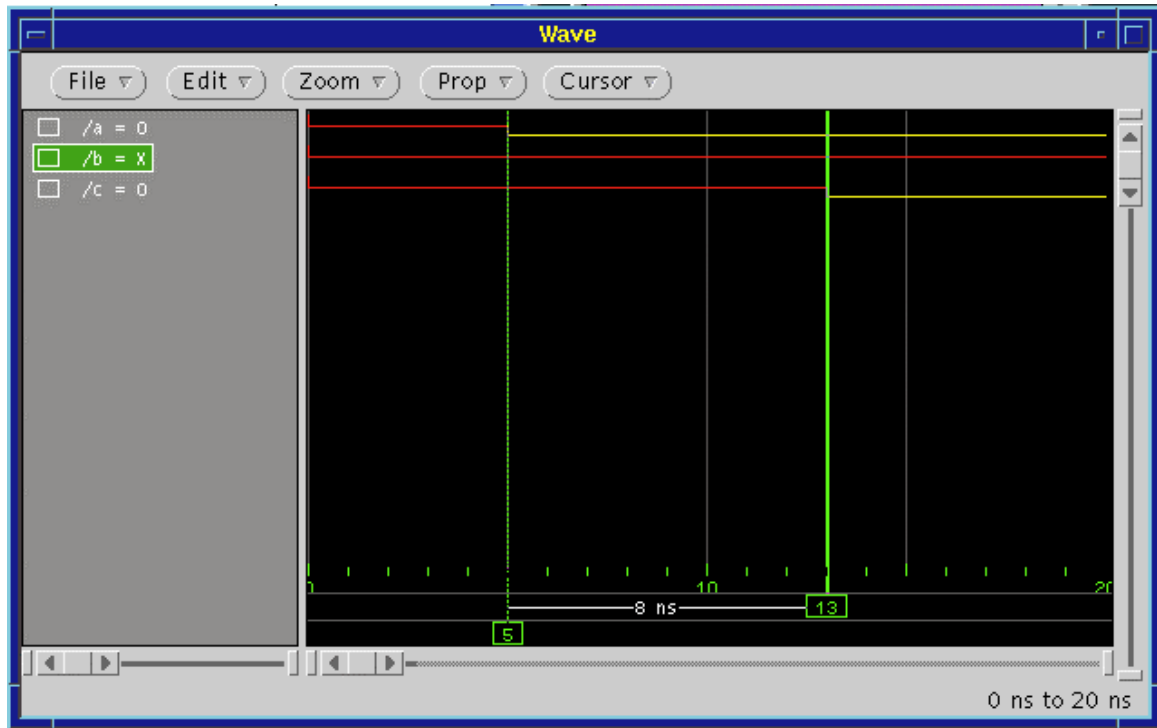


Next, select signals **A**, **B**, and **C** for viewing. Consult the documentation for your specific simulator version for instructions on selecting signals for viewing.

- 3.2 Stimulate the inputs to view the circuit's response. Using the *force* utility in the Mentor Graphics VHDL simulator, set the value of A to 0 after 5 ns. You may want to refer to the manual for your specific simulator for detailed instructions. Note that in general, QuickVHDL commands can be entered through menu options, entered directly through the command line, or read from a command file.
- 3.3 Now run the simulator for sufficient time, e.g. 20 ns. The specific instruction may vary depending on your particular VHDL simulator version, e.g:

```
QHSIM 2> run 20
```

This command will update the Wave window. You may want to zoom the window to full size and add two cursors so that transition times can be observed easily. The resulting window should look something like this:



- 3.4 Add some additional forces and run the simulation. Force signal A to be 1 at time 0 and 0 and 20 ns, and force signal B to be 1 starting at time 0. Simulate for 40 ns. The commands should be similar to the following:

```
QHSIM 3> force -freeze /a 1 0, 0 20
QHSIM 4> force -freeze /b 1 0
QHSIM 5> run 40
```

After zooming to full size, the wave window should look something like this:



3.5 Quit the simulator. Note: Be sure to quit the simulator after each lab.

## 4 Copy, compile and simulate the D Flip Flop code

4.1 Copy the VHDL source file for the D Flip Flop that you will compile from the appropriate source directory and simulate it, e.g.:

```
cp $VHDL_SRC/m10_ex/dff.vhdl dff.vhdl
```

4.2 Compile the **dff.vhdl** file. The specific command may vary depending on the specific Mentor Graphics VHDL simulator, e.g:

```
qvhcom dff.vhdl
```

4.3 Simulate the DFF, e.g:

```
qhsim dff
```

4.4 Select all the signals in the design to be viewed. Consult the documentation for your specific simulator version for instructions on selecting signals for viewing.

4.5 Use the *force* mechanism to generated a clock signal, e.g:

```
QHSIM 6> force -freeze /clk 0 -repeat 20
```

```
QHSIM 7> force -freeze /clk 1 10 -repeat 20
```

## Module 10 - Basic VHDL Lab Tutorial

This will setup a clock that is '0' for 10 ns, '1' for the next 10 ns, and then repeats.

- 4.6 Use the *force* mechanism to drive the *enable* and *d* inputs and run the simulation:

```
QHSIM 8> force -freeze /enable 1
```

```
QHSIM 9> force -freeze /d 0, 1 15, 0 49
```

```
QHSIM 10> run 100
```

- 4.7 After using zooming the waveform viewing window to full size and adding some cursors, the result should look similar to this:



## 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:

## Module 10 - Basic VHDL Lab Tutorial

| Entity Name | Architecture Name | Generics       | Input Ports                | Output Ports | Description                 |
|-------------|-------------------|----------------|----------------------------|--------------|-----------------------------|
| or2         | behav             | trise<br>tfall | a<br>b                     | c            | two input or gate           |
| inv         | behav             | trise<br>tfall | a                          | b            | single input inverter       |
| xor2        | behav             | trise<br>tfall | a<br>b                     | c            | two input exclusive or gate |
| mux2        | behav             | tprop          | a<br>b<br>sel              | c            | two input multiplexor       |
| mux4        | behav             | tprop          | a<br>b<br>c<br>d<br>sel(2) | e            | four input multiplexor      |

Develop a VHDL entity and architecture description for these modules and compile them into the work library. Simulate each design as necessary to ensure proper operation.

Hint - it is often easier to copy the VHDL source file from a similar model and modify it to generate the model you desire instead of writing the VHDL code from scratch