

## **Module 13 - Advanced Concepts in VHDL Tutorial and Exercises**

### **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.**

## Module 13 Tutorial

### 1. Getting started

#### 1.1. Create a directory for the Module 13 lab material:

```
>> mkdir m13_ex
>> cd m13_ex
```

#### 1.2. Copy the source files for the VHDL that you will compile and simulate:

```
>> cp $VHDL_SRC/package.vhdl .
>> cp $VHDL_SRC/test_and2.vhdl .
>> cp $VHDL_SRC/test_mult.vhdl .
```

**These files are also in m13\_lab.tar. This includes behavioral descriptions of the testbench for the and2 gate used in Module 10 and the multiplier used in Module 12.**

#### 1.3. There are also two files that contain the input vectors used with these testbenches. Copy them now:

```
>> cp $VHDL_SRC/test_and2.vec .
>> cp $VHDL_SRC/test_mult.vec .
```

#### 1.4. QuickVHDL needs a work directory for the compiled VHDL files. Create this directory with the following command:

```
>> qhlib work
```

#### 1.5. The full multiplier uses the package from the Module 10 lab, the datapath constructed for the Module 11 lab, and the behavioral descriptions from the Module 12 lab. Map the work libraries from these exercises into the gate\_lib, mult\_lib, and behav\_lib logical libraries:

```
>> qhmap gate_lib ../m10_ex/work
>> qhmap mult_lib ../m11_ex/work
>> qhmap bahav ../m12_ex/work
```

### 2. Examine and compile the package containing some functions to be used in this lab

#### 2.1. Using your favorite text editor, examine the *package.vhdl* file:

```
LIBRARY gate_lib;
USE gate_lib.resources.ALL;

PACKAGE resources IS
```

```

FUNCTION bit2level (bit_in : BIT) RETURN level;

FUNCTION bit2level (bit_vector_in : BIT_VECTOR)
    RETURN level_vector;

END resources;

PACKAGE BODY resources IS

    FUNCTION bit2level (bit_in : BIT) RETURN level IS

        BEGIN

        CASE bit_in IS
            WHEN '0' => RETURN '0';
            WHEN '1' => RETURN '1';
            WHEN OTHERS => ASSERT FALSE
                                REPORT 'Cannot do a conversion to BIT'
                                SEVERITY ERROR;

            END CASE;
        RETURN '0';
        END bit2level;

    FUNCTION bit2level (bit_vector_in : BIT_VECTOR) RETURN
    level_vector IS

        variable I : integer;
        variable level_vector_out : level_vector(bit_vector_in'RANGE);

        BEGIN

        FOR I IN bit_vector_in'RANGE LOOP
            CASE bit_vector_in(I) IS
                WHEN '0' => level_vector_out(I) := '0';
                WHEN '1' => level_vector_out(I) := '1';
                WHEN OTHERS => ASSERT FALSE
                                    REPORT 'Cannot do a conversion to BIT'
                                    SEVERITY ERROR;

                END CASE;
            END LOOP;
        RETURN level_vector_out;
        END bit2level;

    END resources;

```

## 2.2. Compile the VHDL code.

```
>> qvhcom package.vhdl
```

## 3. Examine and compile the testbench for the and2 gate

### 3.1. Using your favorite text editor, examine the *test\_and2.vhdl* file:

```
LIBRARY gate_lib;
```

```
USE gate_lib.resources.ALL;
USE work.resources.ALL;
USE std.textio.ALL;

ENTITY test_and2 IS END;

ARCHITECTURE test OF test_and2 IS
    FILE datain: TEXT OPEN READ_MODE IS 'test_and2.vec';
    SIGNAL a, b, c : level;

BEGIN

    Gate0: ENTITY gate_lib.and2(behav)
    PORT MAP (a,b,c);

    Action: PROCESS

        VARIABLE L : LINE;
        VARIABLE okay : BOOLEAN := TRUE;

        VARIABLE sim_time : TIME;
        VARIABLE a_in,b_in : BIT;
        VARIABLE delay : TIME;

    BEGIN

        WHILE NOT(ENDFILE(datain)) LOOP
            READLINE(DATAIN,L);
            READ(L,sim_time,okay);
            READ(L,a_in,okay);
            READ(L,b_in,okay);
            IF okay THEN
                delay := sim_time-now;
                a <= bit2level(a_in) AFTER delay;
                b <= bit2level(b_in) AFTER delay;
            WAIT FOR delay;
            END IF;
        END LOOP;
        WAIT UNTIL FALSE;
    END PROCESS;

END test;
```

### 3.2. Compile the VHDL code.

```
>> qvhcom -93 test_and2.vhdl
```

## 4. Simulate the compiled code

### 4.1. Start up the Quicksim tool:

```
>> qhsim test_and2
```

### 4.2. Next, select the signals for viewing. Using the right mouse button, select

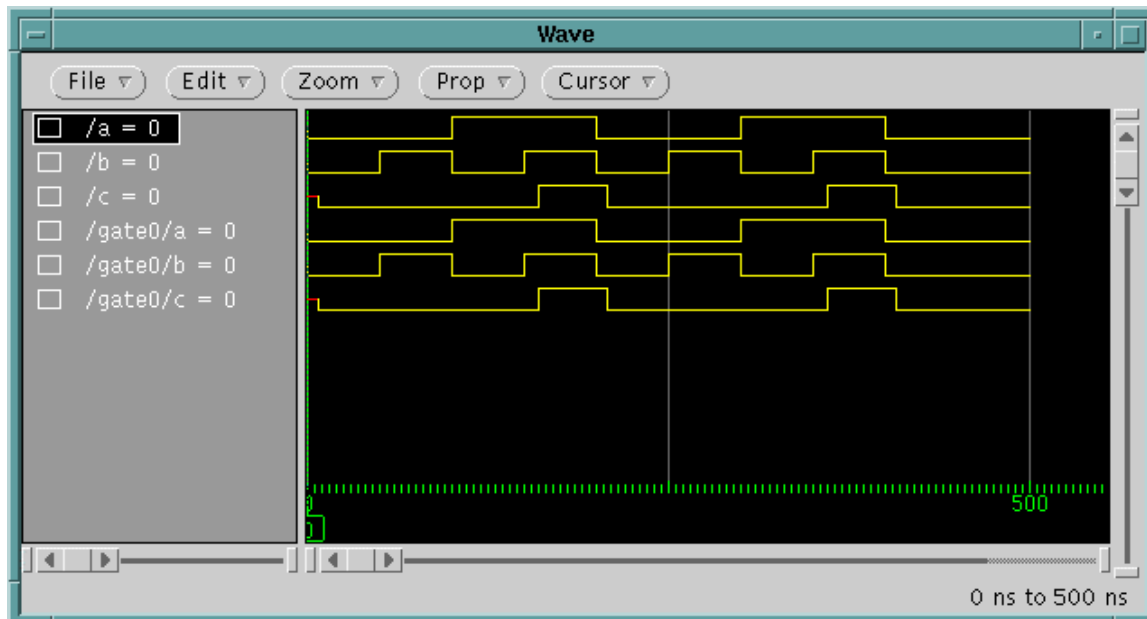
**View->Signals from the menus at the top of the window.**

**4.3. View the waveforms of the selected signals by selecting under the Signals menu, Wave->Signals in design.**

**4.4. Now run the simulation for 500 ns**

```
QHSIM 1> run 500
```

**You should see an output that looks something like this:**



**By editing the contents of the *test\_and2.vec* file, you may change the input values and the times at will be assigned.**

## **5. Testing the Booth Multiplier**

**Using a similar procedure to the one just used to test the and2 gate, we may test the Booth multiplier from the Module 12 lab using a testbench approach.**

### **5.1. Compile the VHDL code.**

```
>> qvhcom -93 test_mult.vhdl
```

### **5.2. Start up the Quicksim tool:**

```
>> qhsim test_mult
```

### **5.3. Select the signals for viewing. Using the right mouse button, select**

**View->Signals from the menus at the top of the window.**

**5.4. For this simulation, it may be more convenient to view the signals in a list format rather than in the waveform format that has been used thus far. From the Signals menu, select**

**List->Signals in design.**

**5.5. Now run the simulation for 5000 ns**

```
QHSIM 2> run 5000
```

**5.6. Again, the input values for this simulation may be changed by editing *test\_mult.do*.**