M.C. Smith

Parts Adapted from:
"VHDL Quick Start" by Peter J. Ashenden

VHDL INTRO (PART III)

Using Tri-State Logic

```
ENTITY test_three IS
      PORT( oe      :   IN std_logic;
            data    :   OUT std_logic_vector(0 to 7));
END test_three;
ARCHITECTURE archtest_three OF test_three IS
BEGIN
  PROCESS (oe)
  BEGIN
      IF (oe = '1')
        THEN data <= "01100100";
        ELSE data <= "ZZZZZZZZ";
      END IF;
  END PROCESS;
END archtest_three;
```

Will synthesize as a tri-state buffer

Simulation

- Execution of the processes in the model
- Discrete event simulation
  - time advances in discrete steps
  - when signal values change—events
- A processes is sensitive to events on input signals
  - specified in wait statements
  - resumes and schedules new values on output signals
    - schedules transactions
    - event on a signal if new value different from old value

Test Benches

CLEMSON UNIVERSITY

- Testing a design by simulation
- Use a *test bench* model
  - Entity is empty
  - Architecture body includes an instance of the design under test
  - Applies sequences of test values to inputs (all external stimulus including clock and reset)
  - Monitors values on output signals
    - either using simulator
    - or with a *process* that verifies correct operation

Test Bench Example 1

```
entity test_bench is
end entity test_bench;

architecture test_reg4 of test_bench is
    signal d0, d1, d2, d3, en, clk, q0, q1, q2, q3 : bit;
begin
    dut : entity reg4
        port map ( d0=>d0, d1=>d1, d2=>d2, d3=>d3, en=>en, clk=>clk,
                   q0=>q0, q1=>q1, q2=>q2, q3=>q3 );
    stimulus : process is
    begin
        d0 <= '1';  d1 <= '1';  d2 <= '1';  d3 <= '1';  wait for 20 ns;
        en <= '0';  clk <= '0';  wait for 20 ns;
        en <= '1';  wait for 20 ns;
        clk <= '1';  wait for 20 ns;
        d0 <= '0';  d1 <= '0';  d2 <= '0';  d3 <= '0';  wait for 20 ns;
        en <= '0';  wait for 20 ns;
        …
        wait;
    end process stimulus;
end architecture test_reg4;
```

empty

device under test

simulation stimulus

Test Bench Example 2

```
Library IEEE;
  use IEEE.STD_LOGIC_1164.all;
  use IEEE.STD_LOGIC_ARITH.all;
  use IEEE.std_logic_unsigned.all;
Library WORK;
  use WORK.all;

entity testbench is
end testbench;

architecture test_adder of testbench is
  signal clk : std_logic:= '0';
  signal rst : std_logic:= '0';

  signal a : std_logic_vector(4 downto 0);
  signal b : std_logic_vector(4 downto 0);
  signal c : std_logic_vector(4 downto 0);
```

empty

Test Bench Example 2 (cont)

```
begin
  dut : entity adder
    port map(
      in1 => a, in2 => b, out => c );
-- ********************************
-- process for simulating the clock
process
begin
  clk <= not(clk);
  wait for 20 ns;
end process;
-- ********************************
-- This process does the RESET
process
begin
  rst <= '1';
  wait for 53 ns;
  rst <= '0';
  wait until(rst'event and rst = '1');
-- stops this process from happening again (this is an initial )
end process;
```

device under test

simulation stimulus

Parallel processes

Test Bench Example 2 (cont)

```
process
  variable i : integer;
  variable j : integer;
  begin
    for i in 1 to 16 loop
      for j in 1 to 16 loop
        a <= CONV_STD_LOGIC_VECTOR(i, 5);
        b <= CONV_STD_LOGIC_VECTOR(j, 5);
        wait until (clk'event and clk='1');
      end loop;
    end loop;
end process;
--*****************************************
end test_adder;
```

simulation stimulus

Parallel processes

converts i or j to std_logic_vector of size 5 bits

Regression Testing

CLEMSON UNIVERSITY

- Test that a refinement (revision) of a design is correct
  - That lower-level structural model does the same as a behavioral model (e.g.)
- Test bench includes two instances of design under test
  - Behavioral and lower-level structural (e.g.)
  - Stimulates both with same inputs
  - Compares outputs for equality
- May need to account for timing differences

Regression Test Example

```
architecture regression of test_bench is
    signal d0, d1, d2, d3, en, clk : bit;
    signal q0a, q1a, q2a, q3a, q0b, q1b, q2b, q3b : bit;
begin
    dut_a : entity work.reg4(struct)
        port map ( d0, d1, d2, d3, en, clk, q0a, q1a, q2a, q3a );
    dut_b : entity work.reg4(behav)
        port map ( d0, d1, d2, d3, en, clk, q0b, q1b, q2b, q3b );
    stimulus : process is
    begin
        d0 <= '1'; d1 <= '1'; d2 <= '1'; d3 <= '1'; wait for 20 ns;
        en <= '0'; clk <= '0'; wait for 20 ns;
        en <= '1'; wait for 20 ns;
        clk <= '1'; wait for 20 ns;
        …
        wait;
    end process stimulus;
```

Regression Test Example

```
…
verify : process is
begin
    wait for 10 ns;
    assert q0a = q0b and q1a = q1b and q2a = q2b and q3a = q3b
        report ”implementations have different outputs”
        severity error;
    wait on d0, d1, d2, d3, en, clk;
end process verify;
end architecture regression;
```

If condition following ASSERT is false, the default message "Assertion violation" is printed. The REPORT statement replaces the default with a “string”. The SEVERITY statement defines the fault class. Four fault classes are defined in severity_level: note, warning, error and failure. If SEVERITY statement is omitted, error is the default.

Some Final Syntax Thoughts

- Reserved keywords (typically lower case, bold font) may not be used as identifiers
- Comments start with two dashes "--"
- Multi-line comments require double dashes on each line
- Semicolons separate stmts and embedded stmts are indented

Final VHDL Definitions?

Very Hard Difficult Language

-or-

VHSIC Hardware Description Language

VHSIC -- Very High Speed Integrated Circuits