Testbenches

Dr. Shubhajit Roy Chowdhury,

Centre for VLSI and Embedded Systems Technology,

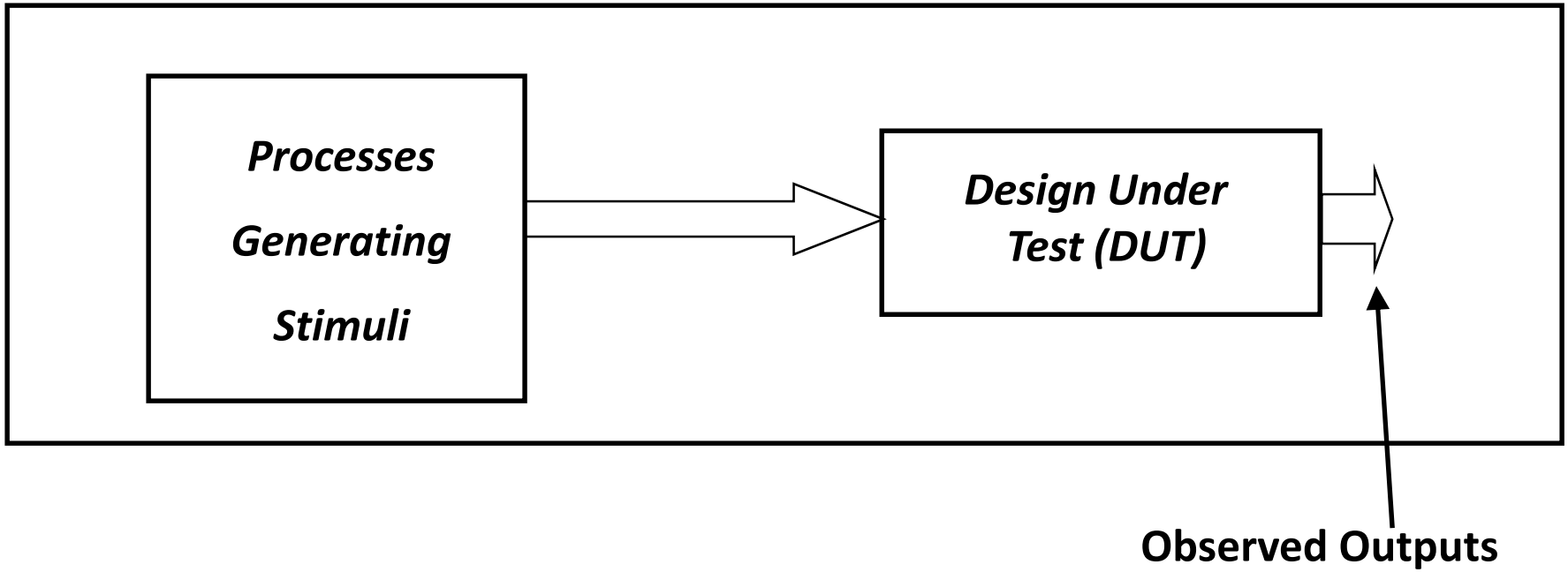
IIIT Hyderabad, India

Email: src.vlsi@iiit.ac.in

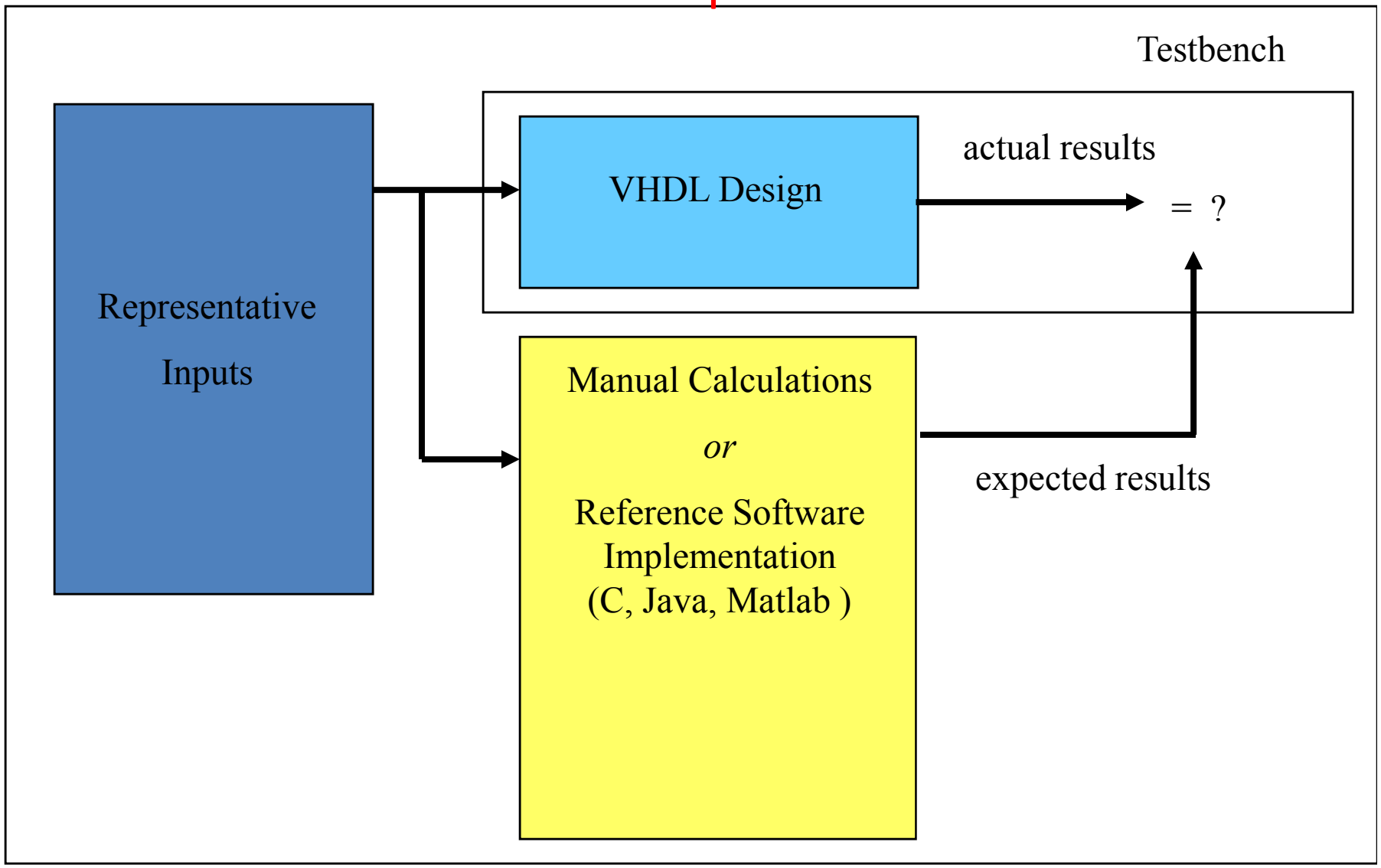
Test Benches

- Testing a design by simulation
- Use a *test bench* model
  - an architecture body that includes an instance of the design under test
  - applies sequences of test values to inputs
  - monitors values on output signals
    - either using simulator
    - or with a process that verifies correct operation

Simple Testbench

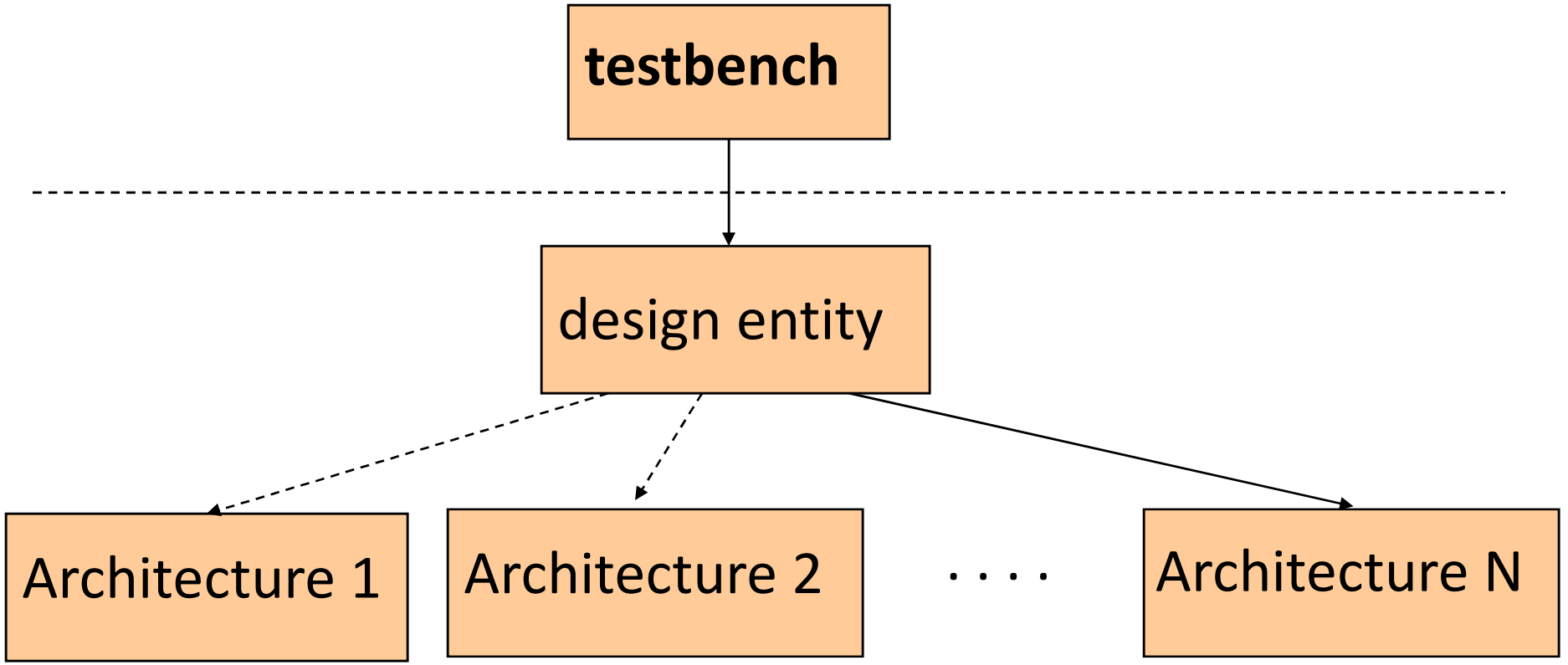


Possible sources of expected results used for comparison



Testbench

The same testbench can be used to
test multiple implementations of the same circuit
(multiple architectures)



VHDL : Test Benches

- We need to stimulate our designs in order to test their functionality

- Stimulus in a real system is from an external source, not from our design

- We need a method to test our designs that is not part of the design itself

- This is called a "Test Bench"

- Test Benches are VHDL entity/architectures with the following:

    - We instantiate the design to be tested using components

    - We call these instantiations "Unit Under Test" (UUT) or "Device Under Test".

    - The entity has no ports

    - We create a stimulus generator within the architecture

    - We can use reporting features to monitor the expected outputs

Objectives of Testbench

The main objectives of TB is to:

– Instantiate the design under test (DUT)

– Generate stimulus waveforms for DUT

– Generate reference outputs and compare them with the outputs of DUT

– Automatically provide a pass or fail indication

Test bench is a part of the circuits specification.

Its a good idea to design the test bench before the DUT, why?

Stimulus and Response

Three ways how TB can generate the stimulus:

– Generate them “on-the-fly”

– Read vectors stored as constants in an array

– Read vectors stored in a separate system file

Response is produced in the test bench.

Response can be stored into file for further processing.

Example:

– Stimulus can be generated with Matlab and TB feeds it into DUT.

– DUT generates the response and TB stores it into file.

– Result can be compared to Matlab simulations.

VHDL : Test Benches

- Test Benches are for Verification, not for Synthesis!!!

  - this allows us to use constructs that we ordinarily wouldn't put in a design because they are not synthesizable

```
entity  Mux_2to1 is

      port (A, B, Sel          : in     STD_LOGIC;
            Y                  : out    STD_LOGIC);

entity Mux_2to1;
```

VHDL : Test Benches

```
entity  Test_Mux is
end entity Test_Mux;                                        -- the test bench entity has no ports

architecture Test_Mux_arch of Test_Mux is

        signal      In1_TB, In2_TB      : STD_LOGIC;        -- setup internal Test Signals
        signal      Sel_TB              : STD_LOGIC;        -- give descriptive names to make
        signal      Out_TB              : STD_LOGIC;        --  apparent they are test signals

        component Mux_2to1                                  -- declare any used components
            port (A, B, Sel         : in        STD_LOGIC;
              Y                     : out       STD_LOGIC);
        end component;

        begin

          UUT : Mux_2to1                                    -- instantiate the design to test
            port map ( A    => In1_TB,
                       B    => In2_TB,
                       Sel => Sel_TB,
                       Y    => Out_TB);
```

VHDL : Test Benches

```
STIM : process                                          -- create process to generate stimulus
  begin
    In1_TB <= '0'; In2_TB <= '0'; Sel_TB <= '0' wait for 10ns      -- we can use wait
    In1_TB <= '0'; In2_TB <= '1'; Sel_TB <= '0' wait for 10ns      -- statements to control
    In1_TB <= '1'; In2_TB <= '0'; Sel_TB <= '0' wait for 10ns      -- the speed of the stim

                              :
                              :
                              :
    In1_TB <= '1'; In2_TB <= '1'; Sel_TB <= '1' wait for 10ns      -- end with a wait…

end process STIM;


end architecture Test_Mux_2to1;
```

Test Bench Example

```
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 work.reg4(behav)
        port map ( d0, d1, d2, d3, en, clk, q0, q1, q2, 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;
```

Assert

Assert is a **non-synthesizable** statement whose purpose is to write out messages on the screen when problems are found during simulation.

Depending on the **severity of the problem**, The simulator is instructed to continue simulation or halt.

Assert - syntax

ASSERT condition
[REPORT "message“]
[SEVERITY severity_level];

The message is written when the condition is FALSE.

Severity_level can be:
Note, Warning, Error (default), or Failure.

Assert - Examples

```
assert initial_value <= max_value
  report "initial value too large"
  severity error;

assert packet_length /= 0
  report "empty network packet received"
  severity warning;

assert false
  report "Initialization complete"
  severity note;
```

Report - syntax

REPORT "message"

[SEVERITY severity_level];

The message is always written.

Severity_level can be:

Note (default), Warning, Error, or Failure.

Report - Examples

```
report "Initialization complete";

report  "Current time = " & time'image(now);

report  "Incorrect branch"  severity error;
```

Report - Examples

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity example_1_tb is
end example_1_tb;

architecture behavioral of example_1_tb is
    signal clk : std_logic := '0';
begin
    clk <= not clk after 100 ns;
    process
        begin
            wait for 1000 ns;
            report "Initialization complete";
            report  "Current time = " & time'image(now);
            wait for 1000 ns;
            report "SIMULATION COMPLETED"  severity failure;
    end process;
end behavioral;
```

Generating reports in the message window

```
reports: process(clk_trigger) begin
  if (clk_trigger = '0' and clk_trigger'EVENT) then
    case segments is
            when seg_0 => report time'image(now) & ": 0 is displayed" ;
            when seg_1 => report time'image(now) & ": 1 is displayed" ;
            when seg_2 => report time'image(now) & ": 2 is displayed" ;
            when seg_3 => report time'image(now) & ": 3 is displayed" ;
            when seg_4 => report time'image(now) & ": 4 is displayed" ;
            when seg_5 => report time'image(now) & ": 5 is displayed" ;
            when seg_6 => report time'image(now) & ": 6 is displayed" ;
            when seg_7 => report time'image(now) & ": 7 is displayed" ;
            when seg_8 => report time'image(now) & ": 8 is displayed" ;
            when seg_9 => report time'image(now) & ": 9 is displayed" ;
    end case;
  end if;
end process;
```

Questions?