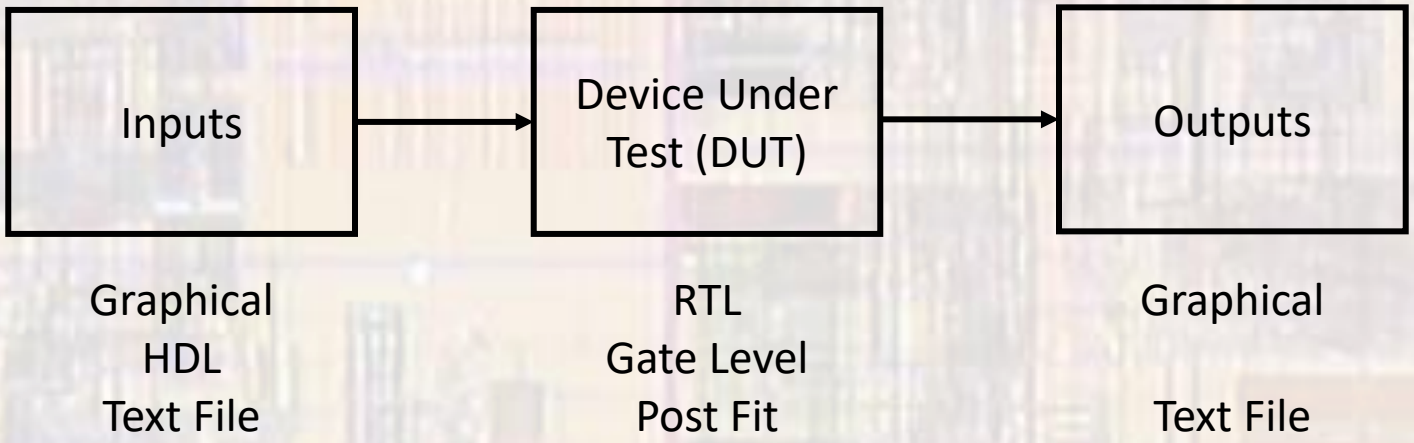
Testbench Basics

Last updated 7/14/23

Testbench Basics

- Testbenches are used to simulate designs
  - RTL level
  - Gate level

- General Testbench Structure



Testbench Basics

- Manual Testbench
  - Inputs
    - Hand coded
    - HDL generated
  - Expected Outputs
    - User generated
  - Results
    - Compare waveforms to expectations
  - Limited to
    - Very small systems
    - Systems with very few inputs and outputs

Testbench Basics

- Automated Testbench
  - Inputs
    - HDL generated
    - Read from a file
  - Expected Outputs
    - HDL generated
      - Be careful not to use the same code
    - Read from a file
      - Be careful not to use the same code
  - Results
    - Identify errors and document
      - Test #, Time, Expected Value, Actual Value, …

Testbench Basics

- VHDL Testbench Code
  - General structure
    - Define input and output signals
    - Connect inputs and outputs to the DUT (device under test)
    - Generate input waveforms
    - Simulate and verify outputs
  - Test benches can contain un-synthesizable code
    - Variables
    - Time
      - After
      - Wait for
    - Loops
      - For
      - While-loop

Testbench Basics

- Quartus Interaction and ModelSim
  - To use Quartus to start/run your ModelSim simulation you must setup the simulation in Quartus
    - Assignments → Settings → EDA Tool Settings → Simulation → Test Benches :
  - Quartus will generate errors on compilation if you set the testbench to the top-level entity
    - Use the device under test as the top-level entity

Testbench Basics

- Documentation, includes and entity
  - No ports – testbench is self-contained
  - Generics are allowed

```
----------------------------------
--
--  testbench_stim_tb.vhdl
--
--  by: johnsontimoj
--
--  created: 7/20/18
--
--  version: 0.0
--
----------------------------------
----------------------------------
--
--  Testbench stimulus generation example
--
--  inputs: None
--
--  outputs: stimulus signals, counter output
--
----------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity testbench_stim_tb is
   generic(
            NUM_BITS: natural := 4
   );
   -- No I/O (ports)
end entity;
```

Testbench Basics

- Signal Definitions
  - Use all caps
  - Setup a constant for the clk (PER)
    - Only need to change this 1 place when changing the clk frequency

```
architecture testbench of testbench_stim_tb is

   -- Input signals
   signal   CLK:          std_logic;
   signal   RSTB:         std_logic;
   signal   X_IN:         std_logic;
   signal   Y_IN:         std_logic_vector(7 downto 0);
   signal   Z_IN:         std_logic;
   signal   CNT_IN:       std_logic_vector(6 downto 0);

   -- Output signals
   signal   CNT_OUT:      std_logic_vector((NUM_BITS - 1) downto 0);

   -- clock constant - 50MHz
   constant PER:  time := 20 ns;
```

Testbench Basics

- Device under test
  - Component + instantiation

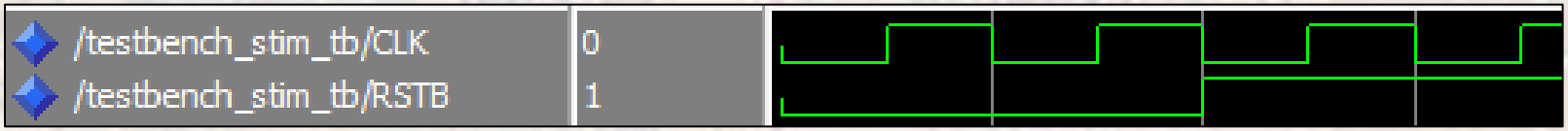
```
   ---------------------------------
   -- Component prototype
   ---------------------------------
   component counter_unsigned_n_bit
      generic(
                N: natural := 8
      );
      port (
            i_clk :     in std_logic;
            i_rstb :    in std_logic;

            o_cnt :     out std_logic_vector((N - 1) downto 0)
      );
   end component;
   ---------------------------------

begin

   ---------------------------------
   -- Device under test (DUT)
   ---------------------------------
   DUT: counter_unsigned_n_bit
      generic map(
                  N => NUM_BITS
      )
      port map(
               i_rstb => RSTB,
               i_clk => CLK,
               o_cnt => CNT_OUT
      );
   ---------------------------------
```

Overwriting the generic

Testbench Basics

- Signal generation
  - Clk and ResetB

```
--
-- Clock process
--
clock: process     -- no sensitivity list allowed
begin
   CLK <= '0';
   wait for PER/2;
   infinite: loop
      CLK <= not CLK;
      wait for PER/2;
   end loop;
end process;

--
-- Reset process
-- active low, changes on falling edge
--
resetB: process    -- no sensitivity list allowed
      begin
   RSTB <= '0';
   wait for 2*PER;
   RSTB <= '1';
   wait;           -- waits forever
end process;
```



Testbench Basics

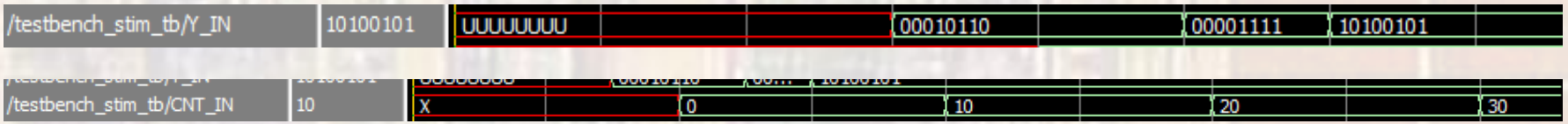
- Signal generation
  - std_logic

```
--
-- Concurrent signal assignment
-- note "after" uses absolute timing within a single statement
--
Z_IN <=  '1',
         '0' after 20 ns,
         '1' after 25 ns,
         '0' after 35 ns;

--
-- fixed pattern process
--
fixed: process     -- no sensitivity list allowed
   constant x_values: std_logic_vector(11 downto 0) := "110101100101";
   begin
      for i in x_values'range loop
         X_IN <= x_values(i);
         wait for 10 ns;
      end loop;
      wait for 20 ns;    -- executes repeatedly
end process;
```

Testbench Basics

- Signal generation
  - std_logic_vector

```
--
-- vector non-periodic process
--
vnp: process       -- no sensitivity list allowed
   begin
      wait for 30 ns;
      Y_IN <= std_logic_vector(to_unsigned(22, Y_IN'length));
      wait for 20 ns;
      Y_IN <= std_logic_vector(to_unsigned(15, Y_IN'length));
      wait for 10 ns;
      Y_IN <= "10100101";
      wait;        -- executes only once
end process;

--
-- counting process
--
count: process     -- no sensitivity list allowed
   begin
      for i in 0 to 9 loop
         wait for 2*PER;
         CNT_IN <= std_logic_vector(to_unsigned((i * 10), 7));
      end loop;
end process;
```



Testbench Basics

- Additional Information
  - Simulation Setup
    - Class Website/Labs : Modelsim Testbench Setup
  - Simulation Tips
    - Restart, Add signals, …
    - Class Website/Labs : Modelsim Simulation Tips