Vivado Tutorial

This tutorial demonstrates how to use Vivado to create, simulate, synthesis, and implement a hardware model (based on Vivado 2020.2 version). It consists of project creation, model simulation, design synthesis and implementation for a combinational logic model in VHDL.

1. Create a project

To create a project, start Vivado from the Start menu or double click Vivado icon on the desktop. This brings up a start page, as shown in Figure 1. Select Create Project in the Quick Start section, then click Next on the Create a New Vivado Project page, which leads to a window shown in Figure 2. After providing the project name and related location, click Next to continue.

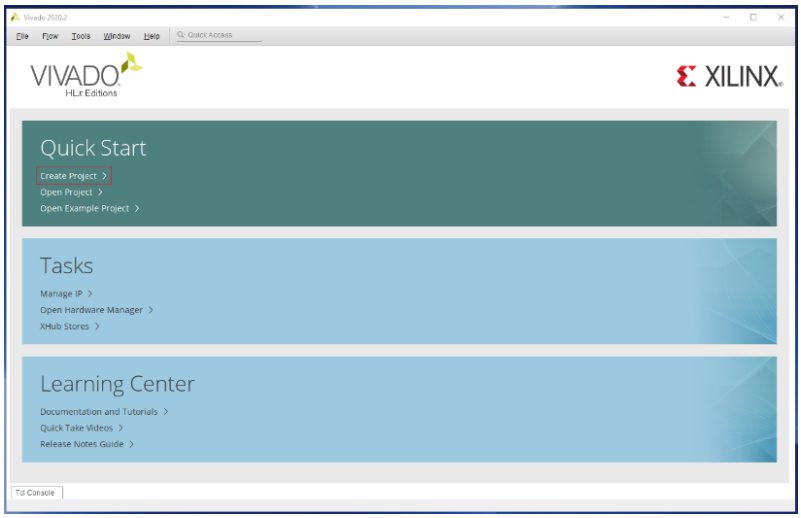

Figure 1

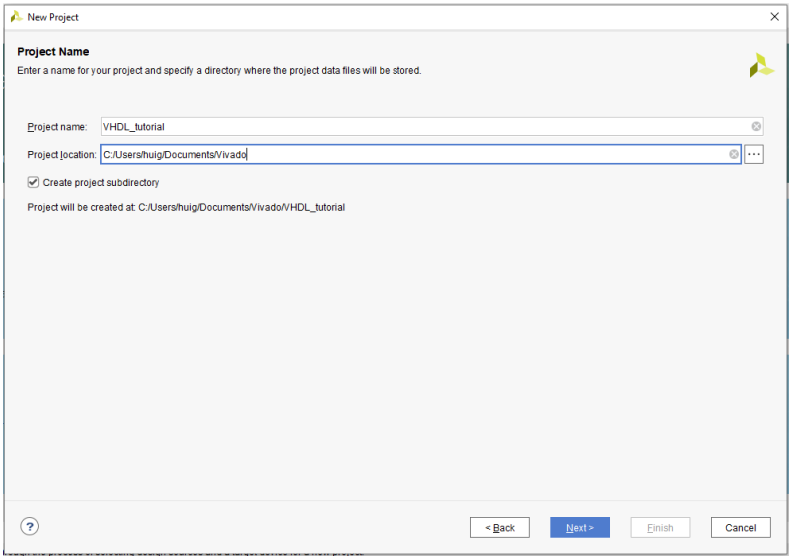

Figure 2

Next select RTL Project as the project type (see Figure 3).

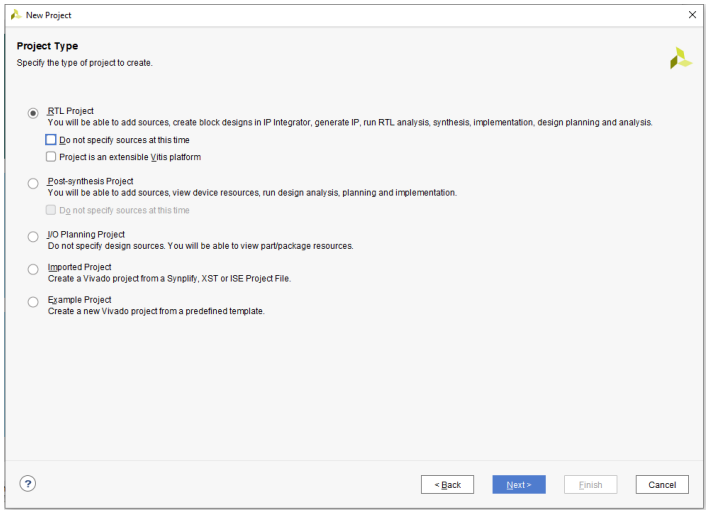

Figure 3

Here we first build a VHDL project template for a target device, as shown Figures 4-8.

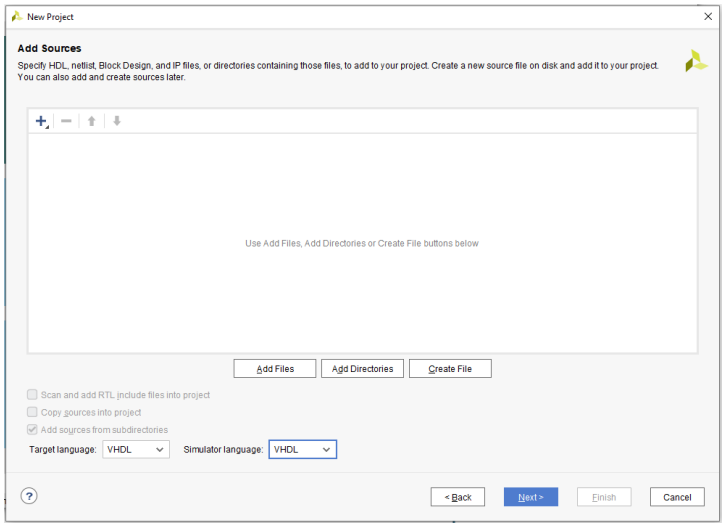

Figure 4

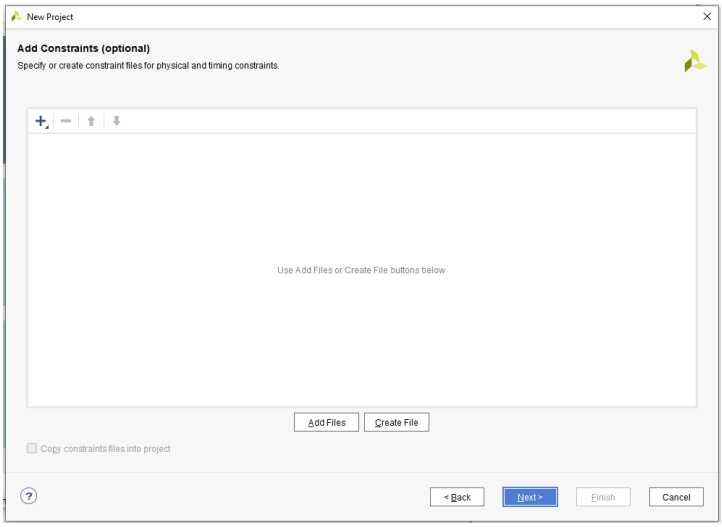

Figure 5

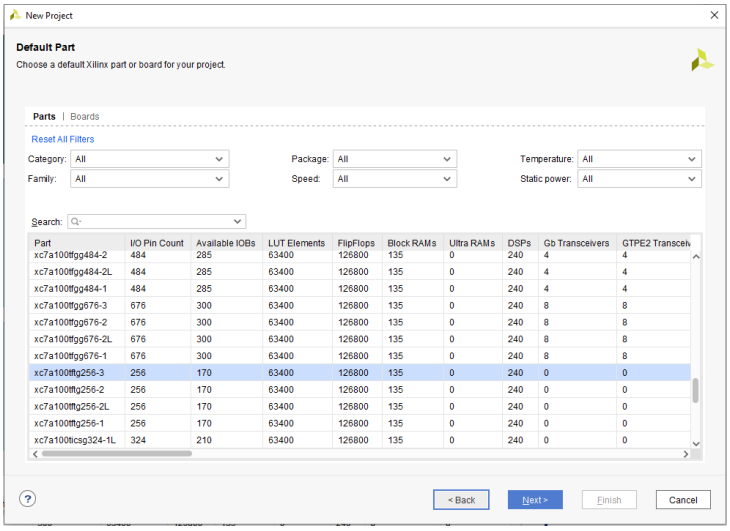

Figure 6

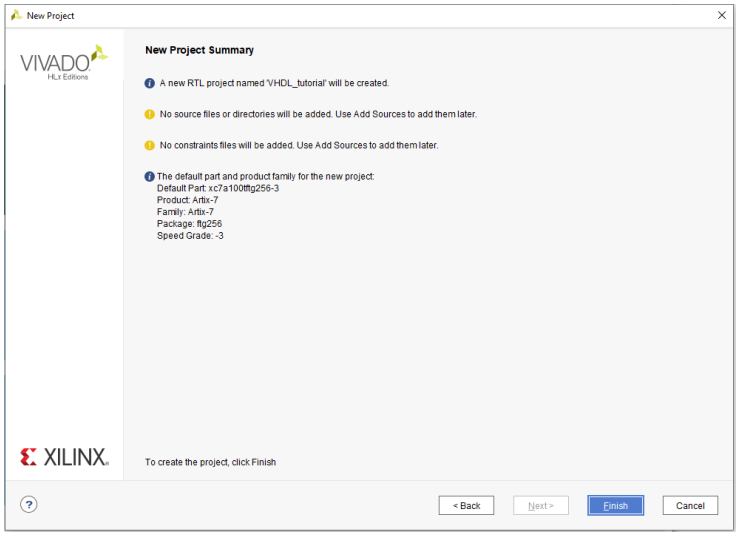

Figure 7

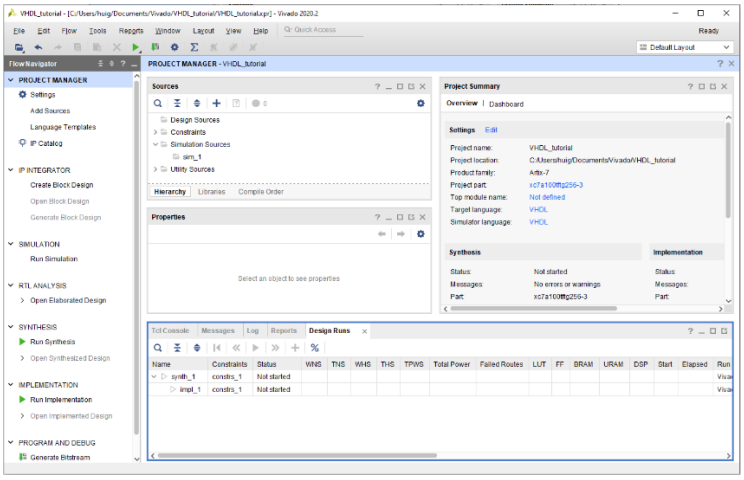

Figure 8

With this template, we can then add our design (a combinational logic for three-input addition) as follows.

In the Add Sources window, create a VHDL file called add3.vhd, and provide the I/O port definitions, as shown in Figures 9 &10. (Note: You can also add existing files to the project through the Add Files option.)

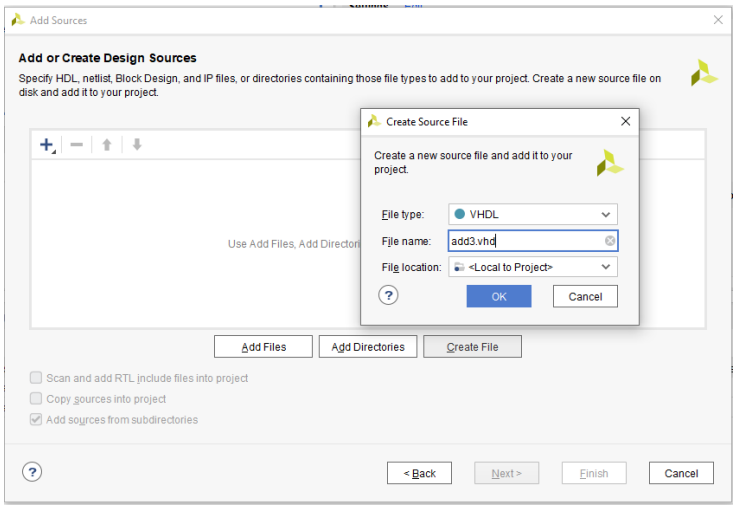

Figure 9

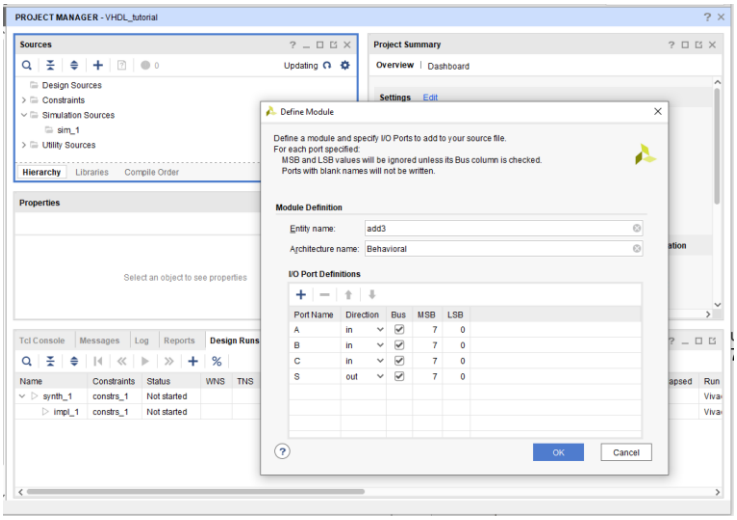

Figure 10

Click add3.vhd in the Project Manager window to open the file. In the file edit window, insert the following two lines to the model, as shown in Figure 11. Save the file.

```
use IEEE.std_logic_unsigned.all;
S <= A + B +C;
```

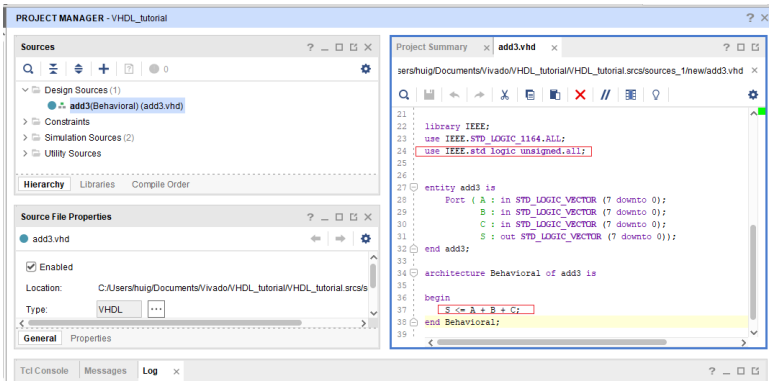

Figure 11

2. Run simulation

2.1 Create Testbench

To simulate the hardware model, we need first to create a testbench. Click Add Sources in the Project Manager panel, then select “Add or create simulation sources”, as shown in Figure 12. Click Next and type “add3_TB” for the name of the testbench, as shown in Figure 13.

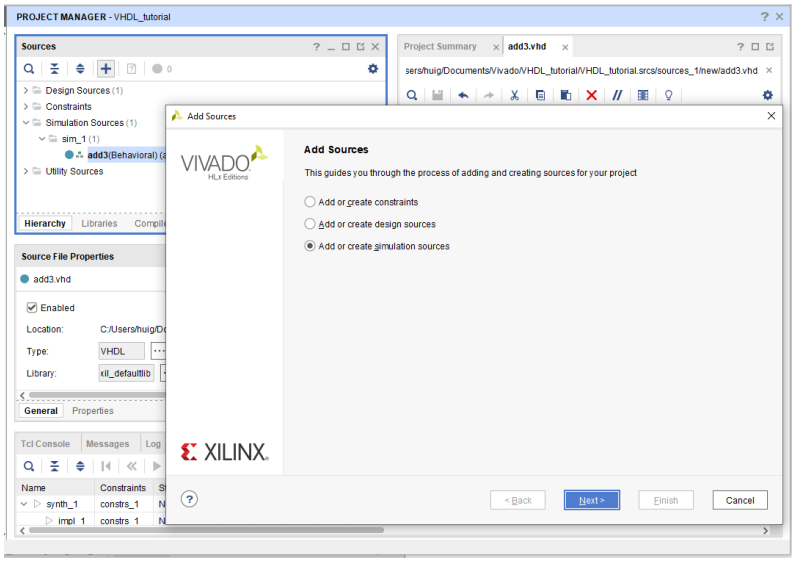

Figure 12

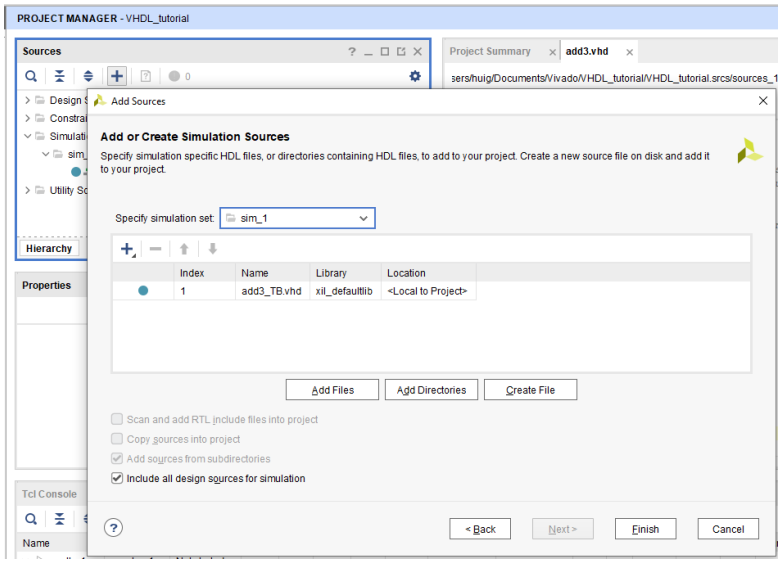

Figure 13

Next in the project Manger window, click add3_TB.vhd to open the file and copy lines in the appendix add3_TB.vhd to the file, as shown in Figure 14. Save the file.

(Note: For your convenience, the txt file of add3_TB is also available on the same Resources page as the tutorial.)

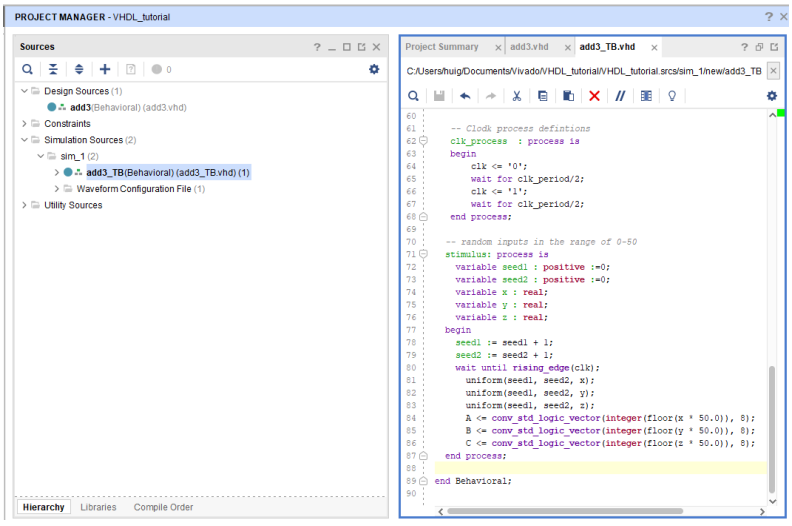

Figure 14

2.2 Simulation

To simulate the model, click on Run Simulation → Run Behavioral Simulation. The simulation result will be generated as shown in Figure 10.

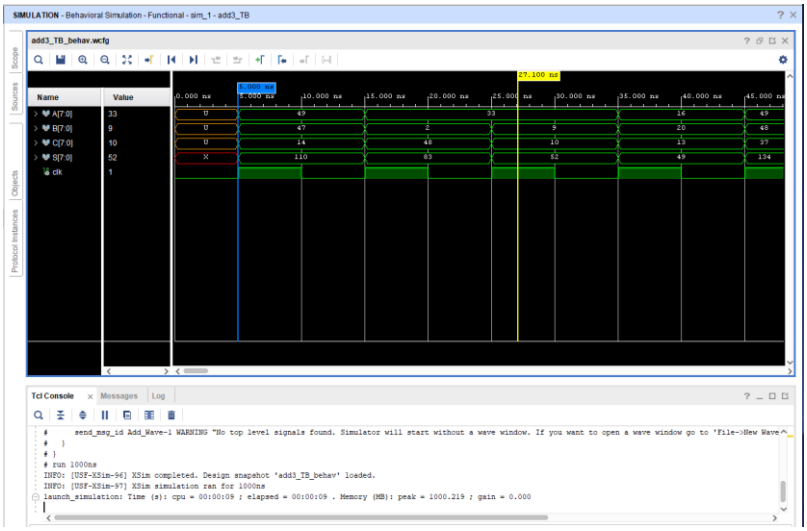

Figure 15

If there is any syntax error in the design or testbench, it will be shown on **Tcl Console** and the waveform window.

3. Synthesize design

To obtain the estimated area cost and time, we need to synthesize the model by selecting “Run Synthesis". When the synthesis is completed, the FPGA resource utilization of the design is available, as shown in Figure 16, which shows that 8 LUTs are used for the design.

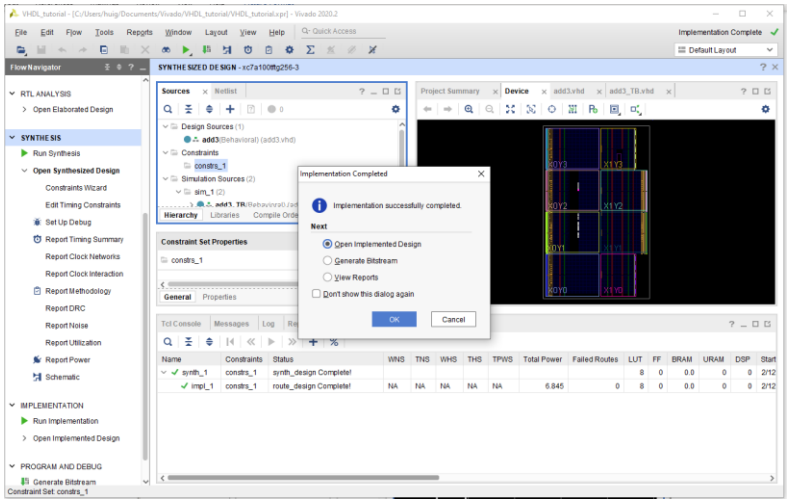

Figure 16

To find the delay of the combinational logic, click “Report Timing Summary” to generate a timing report. You can then find the net delays, logic delays under "Unconstrained paths", as shown in Figure 17, which gives the latency of the whole design is 5.59 ns (the max sum of logic delay and the net delay).

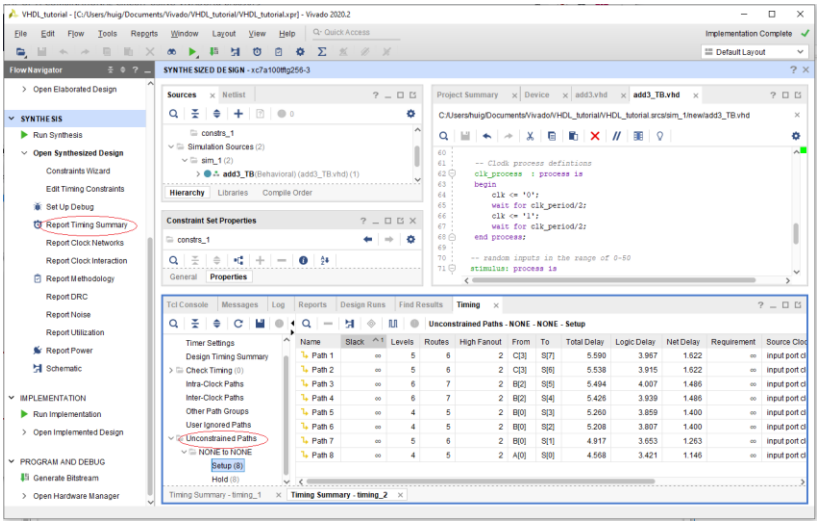

Figure 17

Note: the costs will vary from device to device and from implementation to implementation. In this course, let us use the estimations from the synthesis based on device xc7a100ftg256-3 as used in the tutorial (see Figure 6).

Appendix: add3_TB.vhd

```
Add3_TB.vhd
  library IEEE;
  use IEEE.STD_LOGIC_1164.ALL;
  use ieee.math_real.uniform;
  use ieee.math_real.floor;
  use ieee.std_logic_arith.all;
  use ieee.numeric_std.all;

  entity add3_TB is
  end add3_TB;

  architecture Behavioral of add3_TB is

     component add3
     PORT(
        A: in std_logic_vector(7 downto 0);
        B: in std_logic_vector(7 downto 0);
        C: in std_logic_vector(7 downto 0);
        S: out std_logic_vector(7 downto 0));
     end component;

     --Inputs
     signal A : std_logic_vector(7 downto 0);
     signal B : std_logic_vector(7 downto 0);
     signal C : std_logic_vector(7 downto 0);
     --Outputs
     signal S : std_logic_vector(7 downto 0);

     signal clk: std_logic;
     constant clk_period : time := 10 ns;

  begin

     -- Instantiate the Unit Under Test (UUT)
     uut:   add3 PORT map (
        A => A,
        B => B,
        C => C,
        S => S
       );

    -- Clodk process defintions
    clk_process  : process is
    begin
       clk <= '0';
       wait for clk_period/2;
       clk <= '1';
       wait for clk_period/2;
    end process;

   -- random inputs in the range of 0-50
   stimulus: process is
     variable seed1 : positive :=0;
     variable seed2 : positive :=0;
     variable x : real;
     variable y : real;
     variable z : real;
   begin
     seed1 := seed1 + 1;
     seed2 := seed2 + 1;
     wait until rising_edge(clk);
       uniform(seed1, seed2, x);
       uniform(seed1, seed2, y);
       uniform(seed1, seed2, z);
       A <= conv_std_logic_vector(integer(floor(x * 50.0)), 8);
       B <= conv_std_logic_vector(integer(floor(y * 50.0)), 8);
       C <= conv_std_logic_vector(integer(floor(z * 50.0)), 8);
   end process;

  end Behavioral;
```