CSU22022 Computer Architecture I

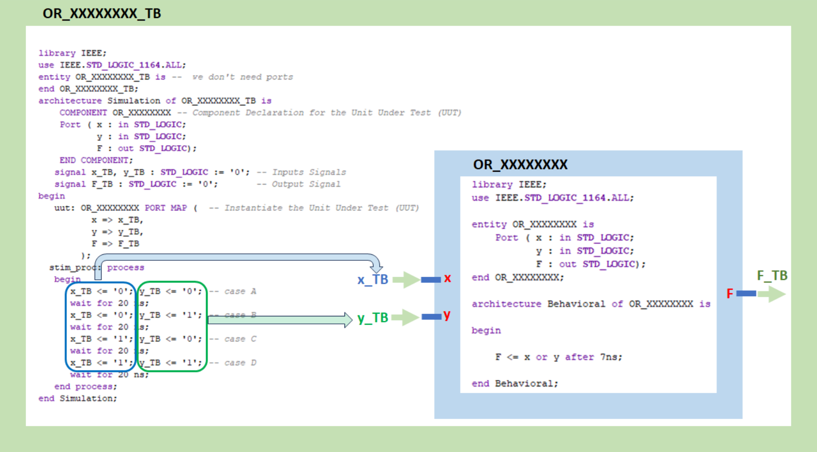
Ninth Lecture - Testbench

Michael Manzke

2023-2024

Trinity College Dublin

Simulation



OR_XXXXXXXX_TB

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR_XXXXXXXX_TB is --  we don't need ports
end OR_XXXXXXXX_TB;
architecture Simulation of OR_XXXXXXXX_TB is
    COMPONENT OR_XXXXXXXX -- Component Declaration for the Unit Under Test (UUT)
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           F : out STD_LOGIC);
    END COMPONENT;
   signal x_TB, y_TB : STD_LOGIC := '0'; -- Inputs Signals
   signal F_TB : STD_LOGIC := '0';       -- Output Signal
begin
   uut: OR_XXXXXXXX PORT MAP (  -- Instantiate the Unit Under Test (UUT)
          x => x_TB,
          y => y_TB,
          F => F_TB
        );
  stim_proc: process
   begin
      x_TB <= '0'; y_TB <= '0'; -- case A
      wait for 20 ns;
      x_TB <= '0'; y_TB <= '1'; -- case B
      wait for 20 ns;
      x_TB <= '1'; y_TB <= '0'; -- case C
      wait for 20 ns;
      x_TB <= '1'; y_TB <= '1'; -- case D
      wait for 20 ns;
   end process;
end Simulation;
```

OR_XXXXXXXX

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_XXXXXXXX is
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           F : out STD_LOGIC);
end OR_XXXXXXXX;

architecture Behavioral of OR_XXXXXXXX is

begin

    F <= x or y after 7ns;

end Behavioral;
```

Figure 1: The Unit Under Test (UUT) OR_XXXXXXXX and the testbench OR_XXXXXXXX_TB

Design Entity - OR_XXXXXXXX

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_XXXXXXXX is
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           F : out STD_LOGIC);
end OR_XXXXXXXX;

architecture Behavioral of OR_XXXXXXXX is

begin

    F <= x or y after 7ns;

end Behavioral;
```

Listing 1: The design entity describes the hardware

Testbench Entity - OR_XXXXXXXX_TB - One

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_XXXXXXXX_TB is
--  we don't need paorts
end OR_XXXXXXXX_TB;

architecture Simulation of OR_XXXXXXXX_TB is

   -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT OR_XXXXXXXX
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           F : out STD_LOGIC);
    END COMPONENT;
```

Listing 2: The testbench entity instantiates the UUT and generates into to the UUT.

Testbench Entity - OR_XXXXXXXX_TB - Two

```
   --Inputs Signals
   signal x_TB : STD_LOGIC := ’0’;
   signal y_TB : STD_LOGIC := ’0’;
   --Output Signal
   signal F_TB : STD_LOGIC := ’0’;
   -- StudentID e.g. 26 33 57 25(DEC) = 1 91 D9 ED(HEX)
   constant StudentID : STD_LOGIC_VECTOR (27 downto 0) := x"191D9ED";
 begin
  -- Instantiate the Unit Under Test (UUT)
   uut: OR_XXXXXXXX PORT MAP (
          x => x_TB,
          y => y_TB,
          F => F_TB
        );
```

Listing 3: The testbench entity instantiates the UUT and generates into to the UUT.

Testbench Entity - OR_XXXXXXXX_TB - Three

```

  stim_proc: process
   begin
      x_TB <= ’0’; y_TB <= ’0’; -- case A
      wait for 20 ns;
      x_TB <= ’0’; y_TB <= ’1’; -- case B
      wait for 20 ns;
      x_TB <= ’1’; y_TB <= ’0’; -- case C
      wait for 20 ns;
      x_TB <= ’1’; y_TB <= ’1’; -- case D
      wait for 20 ns;
   end process;
end Simulation;
```

Listing 4: The testbench entity instantiates the UUT and generates into to the UUT.

Testbench

- We require a testbench for every entity
- These entities may have instantiated other entities.
    - Instantiated entities will have their own testbench entities
- A testbench allows us to generate input to the design entity (UUT)
- We can change the input to he UUT over time

Testbench Entity - Ports

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_XXXXXXXX_TB is
--  we don’t need ports
end OR_XXXXXXXX_TB;
```

Listing 5: The testbench entity has no ports

Testbench Entity - Component Declaration

```
   -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT OR_XXXXXXXX
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           F : out STD_LOGIC);
    END COMPONENT;
```

Listing 6: The testbench entity must declare the component under test in its architecture

The component name and the component ports are those of the Unit Under Test (UUT)

Testbench Entity - signal Declaration

```
    --Inputs Signals
    signal x_TB : STD_LOGIC := ’0’;
    signal y_TB : STD_LOGIC := ’0’;
    --Output Signal
    signal F_TB : STD_LOGIC := ’0’;

    -- StudentID e.g. 26 33 57 25(DEC) = 1 91 D9 ED(HEX)
    constant StudentID : STD_LOGIC_VECTOR (27 downto 0) := x"191D9ED";
```

Listing 7: The testbench entity must declare input and out signals for all the ports of the UUT in its architecture

For this cousework, we also define a constant with our studentID. This constant must be shown simulation output.

Testbench Entity - Instantiate the Unit Under Test (UUT)

```
begin
  -- Instantiate the Unit Under Test (UUT)
   uut: OR_XXXXXXXX PORT MAP (
          x => x_TB,
          y => y_TB,
          F => F_TB
        );
```

Listing 8: In the architecture after the begin we instantiate the previously declared component and connect the simulation signals to it.

Testbench Entity - stim_proc: process

```
  stim_proc: process
   begin
      x_TB <= ’0’; y_TB <= ’0’; -- case A
      wait for 20 ns;
      x_TB <= ’0’; y_TB <= ’1’; -- case B
      wait for 20 ns;
      x_TB <= ’1’; y_TB <= ’0’; -- case C
      wait for 20 ns;
      x_TB <= ’1’; y_TB <= ’1’; -- case D
      wait for 20 ns;
   end process;
```

Listing 9: The stim_proc: process runs all the time and generates intput signals for the Unit Under Test (UUT)

Testbench Entity - For Clocked Designs

```
   Clk_TB <= not CLK_TB after PERIOD/2;
   stim_proc: process
   begin
      wait until CLK_TB’event and CLK_TB=’1’;
      D_TB <= ’0’ after PERIOD/4;                 -- Case A
      wait until CLK_TB’event and CLK_TB=’1’;
      D_TB <= ’0’ after PERIOD/4;                 -- Case B
      wait until CLK_TB’event and CLK_TB=’1’;
      D_TB <= ’1’ after PERIOD/4;                 -- Case C
      wait until CLK_TB’event and CLK_TB=’1’;
      D_TB <= ’0’ after PERIOD/4;                 -- Case D
      wait until CLK_TB’event and CLK_TB=’1’;
      D_TB <= ’1’ after PERIOD/4;                 -- Case E
      wait until CLK_TB’event and CLK_TB=’1’;
   end process;
end Simulation;
```

Listing 10: Simulation for a clocked design