Lab 1: A Introduction to Xilinx Vivado IDE and VHDL

Digital Circuit Design UESTC3020
School of Engineering, University of Glasgow

1. Objective

The objective of this lab is to learn how to use Xilinx Vivado IDE and go through some of the VHDL examples on Vivado IDE.

2. Get familiar with Xilinx Vivado IDE

The software used in this lab is Vivado 2018.2 WebPACK for Windows 10.

2. 1 Create Project

Here are the steps to create a project:

1) Open Vivado 2018.2

2) From "Quick Start" click "Create Project >"

3) Click "Next"

4) Give your project a name and a location

5) Click "Next"

6) Click "Next"

7) Click "Create File"

8) Choose File Type "VHDL" and give it a name and click OK

9) Set "Target Language" to "VHDL" and Simulator Language to "VHDL"

10) Click "Next"

11) Click "Next"

12) Click "Next"

13) Click "Finish" and wait for the project to Open

14) As we learned in class, create I/O ports according to the example file ssmach.vhd and then click "OK"

15) Double Click your VHDL file in the "Sources" window

16) Type the code from ssmach.vhd

2.2 Simulation

The following steps show to how to generate the simulation for the example file ssmach.vhd in 04a_Examples.pdf

1) Click Run Simulation

2) Choose "Run Behavioral Simulation"
   a. Wait for the simulation window
   --objects window to appear
   -- All values are "U" for "Uninitialized"

3) From the "Run" menu, choose "Restart"

4) In the Objects Window, right click on the object named "input"
   a. Click "Force Constant"
   b. Set "Value radix" to "binary"
   c. Set Force Value to 0
   d. Click OK

5) In the Objects Window, right click on the object named "reset"
   a. Click "Force Constant"
   b. Set "Value radix" to "binary"
   c. Set Force Value to 0
   d. Click OK

6) In the Objects Window, right click on the object named "clk"
   a. Click "Force Clock..."
   b. Set "Value radix" to "binary"
   c. Set "Leading edge value" to 1
   d. Set "Trailing edge value" to 0
   e. Set "Period" to 200ns
   f. Click OK

7) From Run menu choose "Run For" 200 ns

8) Following same steps as 6, change the "reset" value to 1

9) From Run menu choose "Run For" 200 ns

10) Following same steps as 6, change the "reset" value to 0

11) From Run menu choose "Run For" 200 ns

12) Following same steps as 5, change the "input" value to 1

13) From Run menu choose "Run For" 200 ns

14) Following same steps as 5, change the "input" value to 0

15) From Run menu choose "Run For" 1000 ns

16) Click "Zoom Fit" on the simulation window

17) Compare the waveforms with the Plot of SSMACH.VHD in 04a_Examples.pdf

2.3 Useful Links to Vivado Instructions

1) https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug893-vivado-ide.pdf
2) http://blog.dev-flow.com/en/5-Behavioral-Simulation-with-the-Vivado-Simulator/

3. Demonstration

You need to demonstrate your work to one of the TAs during your lab time slot. For full credit, you need to demonstrate the waveform generated by Vivado simulator e.g., for SSMACH.VHD on your computer, also to be able to answer the TA's questions.

Appendix: VHDL Example Code

```
-------------------------------------------------------------------------------
-- Title         : Simple State Machine Implementation in VHDL
-- Project       : Digital Design IV
-------------------------------------------------------------------------------
-- File          : ssmach.vhd
-- Author        :   <Craig Slorach@HANDEL>
-- Created       : 1999/02/17
-- Last modified : 1999/02/17
-------------------------------------------------------------------------------
-- Description :
-- Implements a simple state machine with no outputs in VHDL
-------------------------------------------------------------------------------
-- Modification history :
-- 1999/02/17 : created
-------------------------------------------------------------------------------

-- NOTE THIS DESIGN IS FOR DEMONSTRATION/ SIMULATION  PURPOSES ONLY- IT CANNOT
-- BE SYNTHESISED SINCE THERE ARE NO SYSTEM OUTPUTS AND AS SUCH THE
-- OPTIMISATION STEP IN SYNTHESIS WILL YIELD NO CIRCUIT !


library ieee;
use ieee.std_logic_1164.all;

entity SSMACH is

    port (input : in std_logic;         -- system input to start the system
          clk,reset : in std_logic);    -- clock and reset inputs

end SSMACH;

-- purpose: Implement main architecture for SSMACH
architecture BEHAVIOR of SSMACH is

    type STATES is (START, OPENED, CLOSED);  -- possible states
    signal PRESENT_STATE : STATES;      -- present state

begin  -- BEHAVIOR

    -- purpose: Main process
    process (clk, reset)

    begin  -- process

        -- activities triggered by asynchronous reset (active high)
        if reset = ’1’ then

            PRESENT_STATE <= START;     -- default state

        -- activities triggered by rising edge of clock
        elsif clk’event and clk = ’1’ then

            case PRESENT_STATE is

                when START =>
                    if INPUT=’1’ then
                        PRESENT_STATE <= OPENED;
                    else
                        PRESENT_STATE <= START;
                    end if;

                when OPENED =>
                    PRESENT_STATE <= CLOSED;

                when CLOSED =>
                    PRESENT_STATE <= START;

                when others =>
                    PRESENT_STATE <= START;

            end case;
        end if;
    end process;
end BEHAVIOR;
```

```
-------------------------------------------------------------------------------
-- Title         : Simple State Machine Implementation (with outputs) in VHDL
-- Project       : Digital Design IV
-------------------------------------------------------------------------------
-- File          : ssmach2.vhd
-- Author        :   <Craig Slorach@HANDEL>
-- Created       : 1999/02/17
-- Last modified : 1999/02/17
-------------------------------------------------------------------------------
-- Description :
-- Implements a simple state machine with outputs in VHDL
-------------------------------------------------------------------------------
-- Modification history :
-- 1999/02/17 : created
-- 2000/01/13 : Updated for default output behaviour
-------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity SSMACH2 is

  port (input      : in  std_logic;     -- system input to start the system
        output     : out std_logic;     -- system output
        clk, reset : in  std_logic);    -- clock and reset inputs

end SSMACH2;

-- purpose: Implement main architecture for SSMACH2
architecture BEHAVIOR of SSMACH2 is

  type STATES is (START, OPENED, CLOSED);  -- possible states
  signal PRESENT_STATE : STATES;           -- present state

begin  -- BEHAVIOR

  -- purpose: Main process
  process (clk, reset)
  begin  -- process

    -- activities triggered by asynchronous reset (active high)
    if reset = ’1’ then
      PRESENT_STATE <= START;           -- default state
      OUTPUT        <= ’0’;

      -- activities triggered by rising edge of clock
    elsif clk’event and clk = ’1’ then

      OUTPUT        <= ’0’;             -- default output
      PRESENT_STATE <= OPENED;          -- default state

      case PRESENT_STATE is

        when START =>
          if INPUT = ’1’ then
            PRESENT_STATE <= OPENED;
            OUTPUT        <= ’1’;
          else
            PRESENT_STATE <= START;
            OUTPUT        <= ’0’;
          end if;

        when OPENED =>
          PRESENT_STATE <= CLOSED;
          OUTPUT        <= ’0’;

        when CLOSED =>
          PRESENT_STATE <= START;
          OUTPUT        <= ’0’;

        when others =>
          PRESENT_STATE <= START;
          OUTPUT        <= ’0’;

      end case;
    end if;
  end process;
end BEHAVIOR;
```

```
--------------------------------------------------------------------------------
-- Title      : Simple counter in VHDL
-- Project    : Digital Design IV
--------------------------------------------------------------------------------
-- File       : upcounter.vhd
-- Author     :   <Craig Slorach@HANDEL>
-- Company    :
-- Last update: 2000/01/13
-- Platform   :
--------------------------------------------------------------------------------
-- Description: Implements a simple counter in VHDL which counts up only
--             when an input ’inc’ = ’1’
--------------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2000/01/13  1.0      Craig Slorach   Created
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity UPCOUNTER is

  port (
    inc        : in  std_logic;                      -- increment input
    count      : out std_logic_vector (3 downto 0);  -- output
    clk, reset : in  std_logic);                     -- clock and reset

end UPCOUNTER;

architecture BEHAVIOR of UPCOUNTER is

  signal internal_count : unsigned (3 downto 0);  -- internal counter

begin  -- BEHAVIOR

  -- purpose: Main process
  -- type   : sequential
  -- inputs : clk, reset, inc
  -- outputs: internal_count
  process (clk, reset)
  begin  -- process
    if reset = ’1’ then                 -- asynchronous reset (active high)

      internal_count <= "0000";

    elsif clk’event and clk = ’1’ then  -- rising clock edge

      if inc = ’1’ then

        if internal_count = "0100" then
          internal_count <= "0000";     -- reset back

        else
          internal_count <= internal_count + 1;  -- increment

        end if;

      else
        null;

      end if;

    end if;

    -- drive the counter output with the internal value
    count <= std_logic_vector(internal_count);

  end process;


end BEHAVIOR;
```