ECEN302 : Embedded Systems
Lab 2 Submission

Daniel Eisen : 300447549

August 7, 2020

1 Objectives

This lab was to familiarise ourselves with setting up a Vivado project, adding sources, constraints etc and to understand the basic step by step of progressing through the design flow of FPGA programming. It then focused on an introduction to writing, simulating and 'executing' a source written in VHDL; learning the syntax, encountering and overcoming the problems and quirk in the syntax. This is the introduction/basis that all future labs projects were to be based.

2 Methodology

2.1 Setup

To begin with, a project had to be created. This required following specific steps in selecting the correct board etc. The more important steps in this procedure was:

- The initial I/O port definitions, taking care to note if it was a bus, and its direction.
- Including the master constraints file that defines the pinout/switch mappings vital to actually using the device.
- Importing the nesacery template, in this case a simple binary counter construct so we have a basis to modify and build on.

2.2 Part 1 : Counter

Modify template

The provided counter is a 4 bit counter, but is provided with standin variable naming, these had to be mapped to the correct names (clk, direction, count_int) and link the internal signal to the output bus. Learning the new syntax of VHDL was frustrating at first, not sure if its the conflicting legacy stuff like bit vs std_logic or just lack of familiarity but the cheat sheet got frequent use.

Simulate

To test if the module was functioning, a testbench source was loaded in. This allowed the use of the simulation tools within Vivado.

Over a 1000ns simulation period, seen above, we can confirm that the counter incrementing per clk cycle, steps are even has overflows/resets correctly.

Implement and Program

To run the implementation and prepare for writing the bitstream the package pins must be selected in I/O planning. The direction was mapped to a switch, each bit of the output to an LED and connecting the clk.
One hiccup was the I/O std was initially on default, but displayed the correct value. This causes failure unless it was reselected.
Once this was completed, the schematics and device can be viewed. When connecting the board via USB the board can be written to with a generated bitstream.

2.3 Part 2 : Slow it down

Visually the progression of the LED count is unreadable due to speed. To slow the display down, the internal count signal was increased to a 28bit vector incrementing at the same speed, but outputting the most significant 4 bits (LSB @ 5HZ).

2.4 Part 3 : 7-seg and FLASH

The binary output of the counter is hard to directly read from the LED, so we can use the 7-segment on the board to display the encoded character. This was done with a separate decoder module that translated the 4bit binary to a 7 bit binary coded decimal fit for sending to the displays. While this is simple in concept, in the lab there was lots of opportunity to mistype and this caused multiple errors.
The current process only writes the bitstream live to the fabric, the final part of this lab was writing the bitstream to the onboard FLASH so the fabric is programmed on bo

3 Questions

1. We are using the Xilinx Artix 7 FPGA (on the NexysDDR board) and has 324 pins (ie the defined package of the chip).
2. The onboard clock runs a 100Mhz crystal oscillator
3. 100Mhz / 2^{24} = 5.96046448 hertz, assuming a flip per clk
4. 100Mhz / 2^{15} = 3.05175781 kilohertz
5. This line links the value of the internal signal that represents the current counter value the the count output of the module.
6. A testbench defines a testing procedure for use in simulation to debug and confirm the behaviour of the code.

Appendix

counter.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 23.07.2020 10:33:44
-- Design Name:
-- Module Name: counter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
    Port ( clk : in std_logic ;
           direction : in std_logic;
           count_out : out std_logic_vector (3 downto 0);
           Seg_Out : out std_logic_vector (6 downto 0));
end counter;

architecture Behavioral of counter is
    signal count_int : unsigned(27 downto 0) := (others => '0');

    component ssd_decoder is
        port (Bin : in std_logic_vector (3 downto 0);
              Seg_out : out std_logic_vector (6 downto 0));
    end component;

begin
    ssd: ssd_decoder port map (
        Bin => std_logic_vector(count_int(27 downto 24)),
        Seg_Out => Seg_Out);

    process (clk)
    begin
       if clk='1' and clk'event then
          if direction='1' then
             count_int <= count_int + 1;
          else
             count_int <= count_int - 1;
          end if;
       end if;
    end process;

    count_out <= std_logic_vector(count_int(27 downto 24));

end Behavioral;
```

ssd_decoder.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 23.07.2020 12:09:29
-- Design Name:
-- Module Name: ssd_decoder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ssd_decoder is
    Port ( Bin : in std_logic_vector (3 downto 0);
           Seg_Out : out std_logic_vector (6 downto 0));
end ssd_decoder;

architecture Behavioral of ssd_decoder is

begin
    with Bin select
        Seg_Out <= "1111001" when "0001", --1
                   "0100100" when "0010", --2
                   "0110000" when "0011", --3
                   "0011001" when "0100", --4
                   "0010010" when "0101", --5
                   "0000010" when "0110", --6
                   "1111000" when "0111", --7
                   "0000000" when "1000", --8
                   "0010000" when "1001", --9
                   "0001000" when "1010", --a
                   "0000011" when "1011", --b
                   "1000110" when "1100", --c
                   "0100001" when "1101", --d
                   "0000110" when "1110", --e
                   "0001110" when "1111", --Ff
                   "1000000" when others; --0
end Behavioral;
```