COEN 313
Digital Systems Design II Winter 2019
LAB 4: Clocked Processes and registers in VHDL

The purpose of this lab is to become acquainted with clocked processes and registers in VHDL.

Introduction

In this lab, a shift register file together with logic to determine the maximum and minimum values contained within the 4 registers comprising the shift register will be designed using clocked processes in VHDL. Two output registers will be used to store the maximum and minimum values which have been entered into the shift register file via the data input port. Figure 1 gives the block diagram of the system.

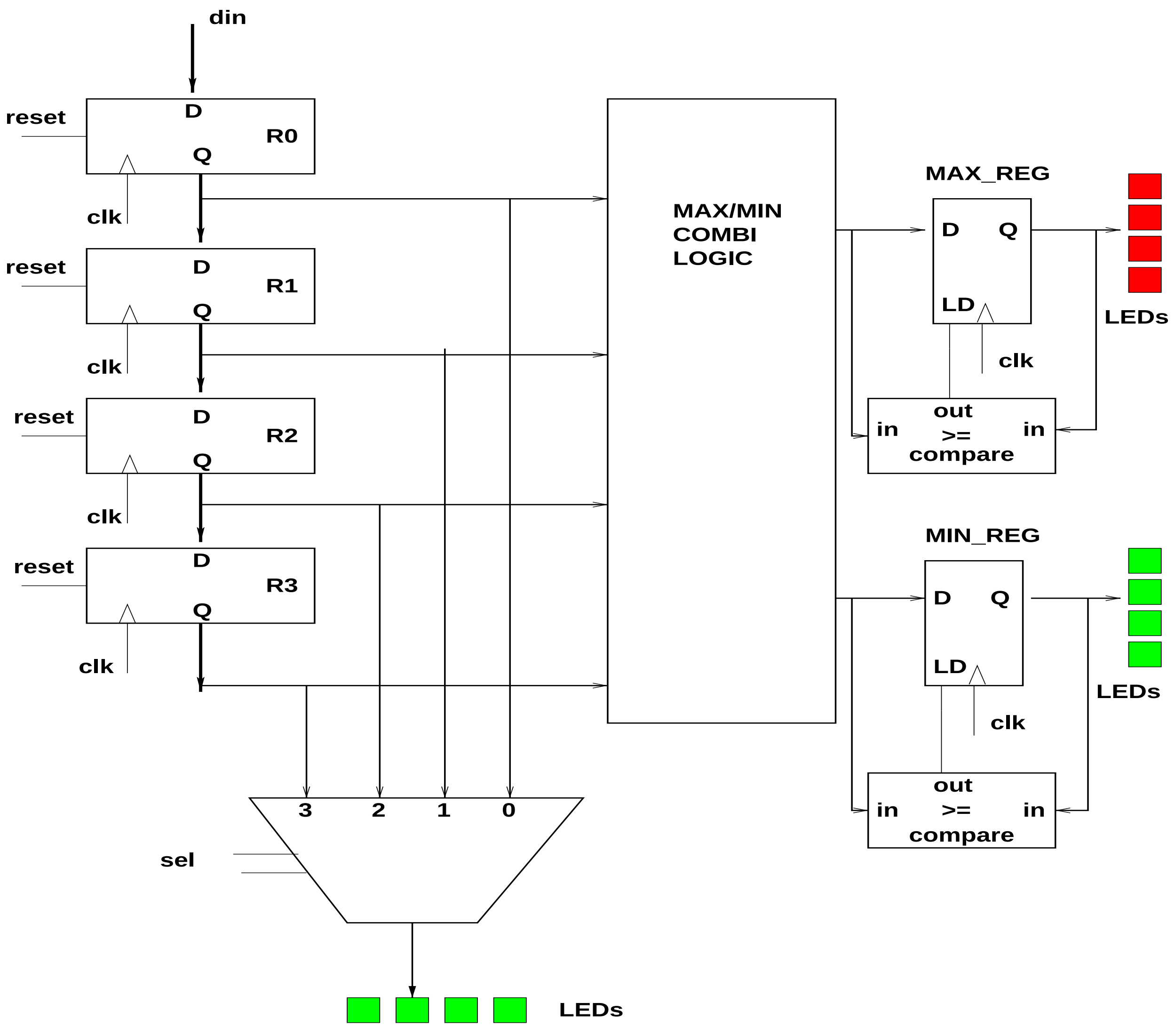

Figure 1: Shift registers with max/min logic and output registers.

The shift register file consists of 4 (R0, R1, R2, R3) registers with each register consisting of 4 bits. These registers have a common reset signal. On every clock cycle, new data on the din port is stored into R0, and the old contents of R0 are shifted into R1, and the old contents of R1 are shifted into R2, etc. In this manner, the shift registers store the past 4 values which have been input.

A combinational logic block is used the determine the maximum and minimum values contained within the 4 shift registers. The output registers (max_reg and min_reg) are used to store the maximum and minimum values encountered during the operation of the circuit. Theses two registers have a load input (LD) which controls the selective loading of new data depending on whether the current values of the maximum and minimum signals (as produced by the max/min logic block) are greater than or less then the current values stores in the two output registers. The values contained within the output registers are displayed on the LEDs of the FPGA board.

A 4-1 multiplexer is used to select any of the 4 registers within the shift register file to display the selected register value on the 4 remaining LEDS. A two bit select input is used to select one of the 4 registers.

Procedure

Employ VHDL **clocked processes** to design the shift register file and the two output registers You may also make use of combinational processes, concurrent signal assignment statements, etc. for the max/min combinational logic block. Simulate your design with the Modelsim simulator to verify correct functioning for several typical values of the input. Synthesize your VHDL code with Precision RTL and obtain the RTL schematic diagram as produced by the synthesis tool. Program the FPGA board with the Xilinx ISE tool. Demonstrate the operation of the design by downloading your synthesized code to the FPGA demonstration board. Use the following VHDL entity specification:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity registers_min_max is
port( din   : in std_logic_vector(3 downto 0);
      reset : in std_logic;
      clk   : in std_logic;
      sel   : in std_logic_vector(1 downto 0);
      max_out : out std_logic_vector(3 downto 0);
      min_out : out std_logic_vector(3 downto 0);
      reg_out : out std_logic_vector(3 downto 0));
end registers_min_max ;
```

Implementation Notes

Due to the large number of input and outputs, it is recommended that you use the following switches and LEDs:

`din`: use the 4 **leftmost** DIP switches on the expansion I/O board
`sel`: use 2 **(third and second rightmost)** of the DIP switches on the expansion I/O board
`reset:` use the **rightmost** switch of the expansion I/O board
`max_out:` use 4 of the LEDS on the expansion I/O board
`min_out:` use the remaining 4 LEDS on the expansion I/O board
`reg_out:` use the 4 LEDS onto the XUPV2Pro board

For convenience, here is the UCF file :

```
NET din(3) LOC = N5;
NET din(2) LOC = L4;
NET din(1) LOC = N2;
NET din(0) LOC = R9;

NET clk LOC = T4;

NET reset LOC = N3 ;
NET sel(1) LOC = P1 ;
NET sel(0) LOC = P7;

NET max_out(3) LOC = P2;
NET max_out(2) LOC = R7;
NET max_out(1) LOC = P4;
NET max_out(0) LOC = T2;

NET min_out(3) LOC = R5;
NET min_out(2) LOC = R3;
NET min_out(1) LOC = V1;
NET min_out(0) LOC = T6;


NET reg_out(3) LOC = AA5;
NET reg_out(2) LOC = AA6;
NET reg_out(1) LOC = AC3;
NET reg_out(0) LOC = AC4;
```

Be sure to account for the active low operation of the LEDs on both the expansion I/0 and the Xilinx development board.

The reset is asynchronous and in the interest of simplicity, the 4 registers within the shift register will be reset to “1000”. This is so that we may shift in a value such as “0010” in order to test that the max/min logic is working correctly. Had we reset the register to “0000” it would be difficult to ascertain whether the minimum logic is working correctly. It is up to you to decide what the max_out and min_out registers should be reset to.

The VHDL **must** contain at a minimum two clocked processes: one for the 4 registers comprising the shift register organization, and another for the max/min output registers. You may use one clocked process for each of the output register for a total of 3 clocked processes.

You are to write the VHDL code such that the two comparators which are used to control the selective load of the output registers are to be **inferred** by the synthesis tool. Hint: use an if statement in the clocked process for the output register to determine whether the condition for updating the value stored within the output register is satisfied or not.

Take advantage of the full power of the VHDL language to write well structured, easily expandable code. Some suggestions which spring to mind are: array types, for loops, etc. Keep in mind which changes would be necessary in your code if the shift register file were to expand from a paltry 4 registers to 4096 registers.

Questions

1. To which signals should a clocked process be sensitive to if the register is to have a synchronous reset?

2. What will happen if the following VHDL code is simulated?

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity registers is
port( din1, din2   : in std_logic_vector(3 downto 0);
      reset : in std_logic;
      clk   : in std_logic;
      mick,keith : out std_logic_vector(3 downto 0));
end registers ;

architecture rtl of registers is

begin

process(clk, reset)
begin
if  reset = ’1’ then
  mick <= "0000";
  keith <= "0000";
elsif clk’event and clk = ’1’ then
```

```
  mick <= din1;
end if;
end process;

process(clk, reset)
begin
if  reset = ’1’ then
  mick <= "0000";
  keith <= "0000";
elsif clk’event and clk = ’1’ then
  keith  <= din2;
end if;
end process;

end rtl;
```

What will happen if the above code is synthesised?