Digital Design Principles

Shift Register

Name: Rajkaran Singh Grewal
Student No: 8882386
Email: rgrewal2386@conestogac.on.ca

Objective:

- Design a parallel load, left and right shift register.

VHDL Introduction:

To make the program in VHDL we will need to initialize the library and entity. Within the entity we will have the following inputs, a standard logic vector of 6 bits for the switch inputs, a standard logic vector of 2 elements for the choice of null, load, shift right and shift left, a standard logic for the push button. For the outputs we will have 4 standard logic vector of a total length of seven elements for the seven segment display, and we will have a standard logic for the led light to let us know that the push button is pressed. Within the architecture we will have the following logic, we will initialize two elements one will be an integer to store the counter value. And the next value will be a standard logic vector which stores 4 elements will represent the four elements we will be manipulating. In the process we will check if the push button is pressed and if it is pressed than we will light up the led, and then check if the counter value is less than one. If it is than we will increment the value of count by one and use a case statement to check if the choice is one of the following options where when choice represents both 0 than nothing will be done. If the second element is high than we will shift the load element to the left by one, and add the value in switch 0 to the last element. In the case of the first element is only high than we will shift the load Element by one to the right and add the value in switch 5 to the first element. For the case of both elements are high than we will load the values in switch 4 to 1 to the load Element. After the if statement that checks if the count is less than one, we will set the seven-segment display with the correct representation of the value so if it’s a 0 it will show a zero, and if it is a one than it will show a one. And after the if statement where we check if the push button is pressed, we will finish the program with an else statement where we set the led to low and the count back to zero.

VHDL program screenshot:

Lab 8 > VHDL > rgrewallab8VHDL.vhd

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rgrewallab8VHDL is
    port(
        S: in std_logic_vector(5 downto 0);
        choice: in std_logic_vector(1 downto 0);
        P: in std_logic;
        hex1: out std_logic_vector(6 downto 0);
        hex2: out std_logic_vector(6 downto 0);
        hex3: out std_logic_vector(6 downto 0);
        hex4: out std_logic_vector(6 downto 0);
        led: out std_logic
    );
end rgrewallab8VHDL;

architecture shiftRegister of rgrewallab8VHDL is
    signal count: integer := 0;
    signal loadValue: std_logic_vector(3 downto 0) := "0000";
begin
    process(S,choice,count,loadValue,P) is
    begin
        if(P = '0') then
            led <= '1';
            if(count < 1) then
                count <= count + 1;
                case choice is
                    when "00" =>
                        NULL;
                    when "01" =>
                        loadValue(3) <= loadValue(2);
```

```
                    loadValue(2) <= loadValue(1);
                    loadValue(1) <= loadValue(0);
                    loadValue(0) <= S(0);
                when "10" =>
                    loadValue(0) <= loadValue(1);
                    loadValue(1) <= loadValue(2);
                    loadValue(2) <= loadValue(3);
                    loadValue(3) <= S(5);
                when "11" =>
                    loadValue(0) <= S(1);
                    loadValue(1) <= S(2);
                    loadValue(2) <= S(3);
                    loadValue(3) <= S(4);
                when others =>
                    NULL;
            end case;
        end if;
        case loadValue(0) is
            when '0' =>
                hex1 <= "0000001";
            when '1' =>
                hex1 <= "1001111";
            when others =>
                hex1 <= "0000000";
        end case;
        case loadValue(1) is
            when '0' =>
                hex2 <= "0000001";
            when '1' =>
                hex2 <= "1001111";
```

```
                when others =>
                    hex2 <= "0000000";
            end case;
            case loadValue(2) is
                when '0' =>
                    hex3 <= "0000001";
                when '1' =>
                    hex3 <= "1001111";
                when others =>
                    hex3 <= "0000000";
            end case;
            case loadValue(3) is
                when '0' =>
                    hex4 <= "0000001";
                when '1' =>
                    hex4 <= "1001111";
                when others =>
                    hex4 <= "0000000";
            end case;
        else
            led <= '0';
            count <= 0;
        end if;
    end process;
end shiftRegister;
```

VHDL compilation report:

Compilation Report - rgrewallab8VHDL

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Mar 23 16:25:55 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	rgrewallab8VHDL
Top-level Entity Name	rgrewallab8VHDL
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	45 / 32,070 (< 1 %)
Total registers	0
Total pins	38 / 457 (8 %)
Total virtual pins	0
Total block memory bits	Total pins 65,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Pin Planner screenshot:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
choice[1]	Input	PIN_AC9	3A	B3A_N0	PIN_AC9	2.5 V
choice[0]	Input	PIN_AE11	3A	B3A_N0	PIN_AE11	2.5 V
hex1[6]	Output	PIN_AE26	5A	B5A_N0	PIN_AE26	2.5 V
hex1[5]	Output	PIN_AE27	5A	B5A_N0	PIN_AE27	2.5 V
hex1[4]	Output	PIN_AE28	5A	B5A_N0	PIN_AE28	2.5 V
hex1[3]	Output	PIN_AG27	5A	B5A_N0	PIN_AG27	2.5 V
hex1[2]	Output	PIN_AF28	5A	B5A_N0	PIN_AF28	2.5 V
hex1[1]	Output	PIN_AG28	5A	B5A_N0	PIN_AG28	2.5 V
hex1[0]	Output	PIN_AH28	5A	B5A_N0	PIN_AH28	2.5 V
hex2[6]	Output	PIN_AJ29	5A	B5A_N0	PIN_AJ29	2.5 V
hex2[5]	Output	PIN_AH29	5A	B5A_N0	PIN_AH29	2.5 V
hex2[4]	Output	PIN_AH30	5A	B5A_N0	PIN_AH30	2.5 V
hex2[3]	Output	PIN_AG30	5A	B5A_N0	PIN_AG30	2.5 V
hex2[2]	Output	PIN_AF29	5A	B5A_N0	PIN_AF29	2.5 V
hex2[1]	Output	PIN_AF30	5A	B5A_N0	PIN_AF30	2.5 V
hex2[0]	Output	PIN_AD27	5A	B5A_N0	PIN_AD27	2.5 V
hex3[6]	Output	PIN_AB23	5A	B5A_N0	PIN_AB23	2.5 V
hex3[5]	Output	PIN_AE29	5B	B5B_N0	PIN_AE29	2.5 V
hex3[4]	Output	PIN_AD29	5B	B5B_N0	PIN_AD29	2.5 V

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
hex3[3]	Output	PIN_AC28	5B	B5B_N0	PIN_AC28	2.5 V
hex3[2]	Output	PIN_AD30	5B	B5B_N0	PIN_AD30	2.5 V
hex3[1]	Output	PIN_AC29	5B	B5B_N0	PIN_AC29	2.5 V
hex3[0]	Output	PIN_AC30	5B	B5B_N0	PIN_AC30	2.5 V
hex4[6]	Output	PIN_AD26	5A	B5A_N0	PIN_AD26	2.5 V
hex4[5]	Output	PIN_AC27	5A	B5A_N0	PIN_AC27	2.5 V
hex4[4]	Output	PIN_AD25	5A	B5A_N0	PIN_AD25	2.5 V
hex4[3]	Output	PIN_AC25	5A	B5A_N0	PIN_AC25	2.5 V
hex4[2]	Output	PIN_AB28	5B	B5B_N0	PIN_AB28	2.5 V
hex4[1]	Output	PIN_AB25	5A	B5A_N0	PIN_AB25	2.5 V
hex4[0]	Output	PIN_AB22	5A	B5A_N0	PIN_AB22	2.5 V
led	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V
P	Input	PIN_Y16	3B	B3B_N0	PIN_Y16	2.5 V
S[5]	Input	PIN_AD12	3A	B3A_N0	PIN_AD12	2.5 V
S[4]	Input	PIN_AD11	3A	B3A_N0	PIN_AD11	2.5 V
S[3]	Input	PIN_AF10	3A	B3A_N0	PIN_AF10	2.5 V
S[2]	Input	PIN_AF9	3A	B3A_N0	PIN_AF9	2.5 V
S[1]	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
S[0]	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V

Verilog introduction:

For the program in Verilog we will need to initialize the module with the parameters having the following inputs, first of all we will have an array of 6 elements that will represent are switches, than we will have array of two element which will store the value for our choices, that can be null, shift left, shift right, load. We will have the input for the push button. Next for the outputs we will have four array which can store seven elements, this will represent our four seven segment displays. And finally, we will have an output to store our led value.

For the logic we will initialize an array of four elements to store our load-value. We will also initialize an integer to store our count value. Within the always, we will check if the push button is pressed using a if statement, for the condition we will light up the led and increment the count by one. We will check if the count is equal to 0 with an if statement. Then with the if block we will use a case statement to check what choice has been provided. If both choices are zero than we will do nothing, if the last element is high and the other element is low than we will shift the load-value to the left and add the value in switch zero to the last element in load-value. If the case is high for the first element and low for the second element, we will shift the elements to the right and insert the value of switch 5 to the first element. For the case where both element is high than we load the value of switch 4 to 1 to the load-value. After the if block which checks if the count value is less than zero we set the values of the seven segment display according to the load-value. After the if block of which checks if the push button is pressed, we will have an else block where the led is set to low and the count is set to zero.

We will also include a function where the value which is return is an array of seven segments and the input is a character. The character will be check using a case statements where the output will be the highs and lows value to set the seven segment displayed.

Verilog program screenshot:

```verilog
module rgrewallab8verilog(input [5:0] Switch,
                          input [1:0] choice,
                          input PushButton,
                          output reg [6:0] hex1,
                          output reg [6:0] hex2,
                          output reg [6:0] hex3,
                          output reg [6:0] hex4,
                          output reg led);
    reg [3:0] loadValue = 4'b0000;
    integer count = 0;
    always @(Switch,choice) begin
        if(!PushButton) begin
            led = 1;
            count = count + 1;
            if(count == 0) begin
                case(choice)
                    2'b00: begin
                    end
                    2'b01: begin
                        loadValue[3] = loadValue[2];
                        loadValue[2] = loadValue[1];
                        loadValue[1] = loadValue[0];
                        loadValue[0] = Switch[0];
                    end
                    2'b10: begin
                        loadValue[0] = loadValue[1];
                        loadValue[1] = loadValue[2];
                        loadValue[2] = loadValue[3];
                        loadValue[3] = Switch[5];
                    end
```

```
                    2'b11: begin
                        loadValue[0] = Switch[1];
                        loadValue[1] = Switch[2];
                        loadValue[2] = Switch[3];
                        loadValue[3] = Switch[4];
                    end
                    default: begin
                        loadValue = 4'b0000;
                    end
                endcase
            end
            hex1 = displaySevenSegment(loadValue[0]);
            hex2 = displaySevenSegment(loadValue[1]);
            hex3 = displaySevenSegment(loadValue[2]);
            hex4 = displaySevenSegment(loadValue[3]);
        end
        else begin
            led = 0;
            count = 0;
        end
    end
    function [6:0]displaySevenSegment(input character);
        case(character)
            0: begin
                displaySevenSegment = 7'b0000001;
            end
            1: begin
                displaySevenSegment = 7'b1001111;
            end
            default: begin
                displaySevenSegment = 7'b0000000;
                end
        endcase
    endfunction
endmodule
```

Verilog compilation report:

Compilation Report - rgrewallab8verilog

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Mar 23 16:30:44 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	rgrewallab8verilog
Top-level Entity Name	rgrewallab8verilog
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	35 / 32,070 (< 1 %)
Total registers	0
Total pins	38 / 457 (8 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Verilog pin planner screenshot:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
PushButton	Input	PIN_Y16	3B	B3B_N0	PIN_Y16	2.5 V
Switch[5]	Input	PIN_AD12	3A	B3A_N0	PIN_AD12	2.5 V
Switch[4]	Input	PIN_AD11	3A	B3A_N0	PIN_AD11	2.5 V
Switch[3]	Input	PIN_AF10	3A	B3A_N0	PIN_AF10	2.5 V
Switch[2]	Input	PIN_AF9	3A	B3A_N0	PIN_AF9	2.5 V
Switch[1]	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
Switch[0]	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
choice[1]	Input	PIN_AC9	3A	B3A_N0	PIN_AC9	2.5 V
choice[0]	Input	PIN_AE11	3A	B3A_N0	PIN_AE11	2.5 V
hex1[6]	Output	PIN_AE26	5A	B5A_N0	PIN_AE26	2.5 V
hex1[5]	Output	PIN_AE27	5A	B5A_N0	PIN_AE27	2.5 V
hex1[4]	Output	PIN_AE28	5A	B5A_N0	PIN_AE28	2.5 V
hex1[3]	Output	PIN_AG27	5A	B5A_N0	PIN_AG27	2.5 V
hex1[2]	Output	PIN_AF28	5A	B5A_N0	PIN_AF28	2.5 V
hex1[1]	Output	PIN_AG28	5A	B5A_N0	PIN_AG28	2.5 V
hex1[0]	Output	PIN_AH28	5A	B5A_N0	PIN_AH28	2.5 V
hex2[6]	Output	PIN_AJ29	5A	B5A_N0	PIN_AJ29	2.5 V
hex2[5]	Output	PIN_AH29	5A	B5A_N0	PIN_AH29	2.5 V
hex2[4]	Output	PIN_AH30	5A	B5A_N0	PIN_AH30	2.5 V
hex2[3]	Output	PIN_AG30	5A	B5A_N0	PIN_AG30	2.5 V
hex2[2]	Output	PIN_AF29	5A	B5A_N0	PIN_AF29	2.5 V
hex2[1]	Output	PIN_AF30	5A	B5A_N0	PIN_AF30	2.5 V
hex2[0]	Output	PIN_AD27	5A	B5A_N0	PIN_AD27	2.5 V
hex3[6]	Output	PIN_AB23	5A	B5A_N0	PIN_AB23	2.5 V
hex3[5]	Output	PIN_AE29	5B	B5B_N0	PIN_AE29	2.5 V
hex3[4]	Output	PIN_AD29	5B	B5B_N0	PIN_AD29	2.5 V
hex3[3]	Output	PIN_AC28	5B	B5B_N0	PIN_AC28	2.5 V
hex3[2]	Output	PIN_AD30	5B	B5B_N0	PIN_AD30	2.5 V
hex3[1]	Output	PIN_AC29	5B	B5B_N0	PIN_AC29	2.5 V
hex3[0]	Output	PIN_AC30	5B	B5B_N0	PIN_AC30	2.5 V
hex4[6]	Output	PIN_AD26	5A	B5A_N0	PIN_AD26	2.5 V
hex4[5]	Output	PIN_AC27	5A	B5A_N0	PIN_AC27	2.5 V
hex4[4]	Output	PIN_AD25	5A	B5A_N0	PIN_AD25	2.5 V
hex4[3]	Output	PIN_AC25	5A	B5A_N0	PIN_AC25	2.5 V
hex4[2]	Output	PIN_AB28	5B	B5B_N0	PIN_AB28	2.5 V
hex4[1]	Output	PIN_AB25	5A	B5A_N0	PIN_AB25	2.5 V
hex4[0]	Output	PIN_AB22	5A	B5A_N0	PIN_AB22	2.5 V
led	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V

Conclusion:

Thus we can conclude that by both the programming languages we will check if the push button is pressed and if the counter is less than 1 than we will use case statement and set the load-element according the requirement of the objective. We will finially set the seven segment display to the load-element to show us how the shift register works.