Digital Design Principles

Counters

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

Objective:

- Design some simple counters.
- Learn to use Asynchronous Reset in VHDL

VHDL Introduction:

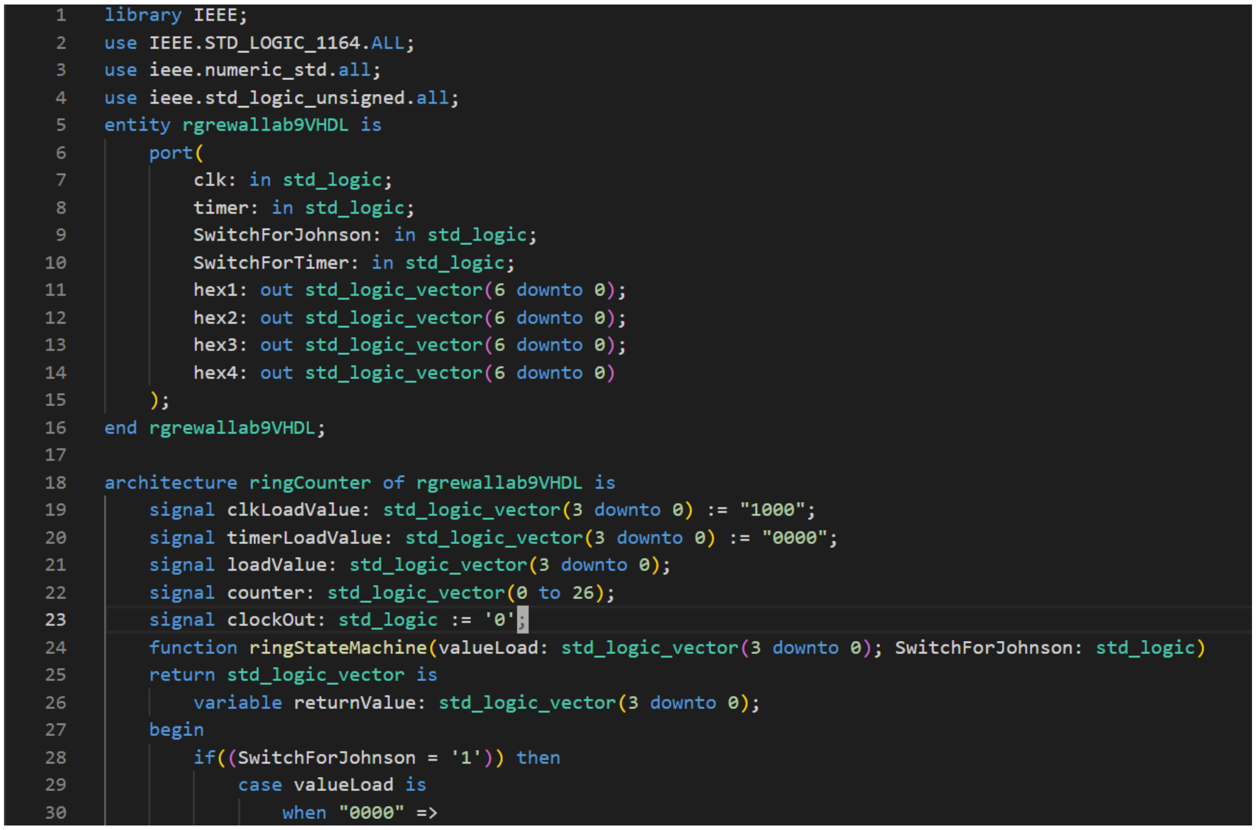
To make the program in VHDL we will need to initialize the library and entity. Within the entity we will have the following inputs, we will have a standard logic with clock, a timer as a standard logic, a Switch to switch for the Johnson ring counter using a standard logic, a switch for to use the clock or timer using a standard logic. For the output we will use 4 standard logic vector of total length of seven units which will be used for the Seven Segment Display.

Within the Architecture of the program we will define a signal to store the clock load Value which is a standard logic vector of 4 units. We will also have a timer load value which is a standard logic vector which stores 4 units, a load value which is a standard logic vector which stores 4 units, we will also initialize counter which also a standard logic vector which stores 27 units. We will also have standard logic variable to store clock out. Then we will initialize a function to change the state of the ring. We will also initialize a function to get the seen segment value according to the value given in the function. We than will initialize the first process which is checking on the clock which is the push button. We will write an if statement where the rising edge of the clock will be checked if its true and within the block. We will check the if statement where we will check if the clock load value has the value 0000 and we the switch for Johnson is not turned on, then we will change the clock load value to 1000. Than after this block we will have another if statement where we will check if the switch for timer is not turned on. And than we will assign the clock load value to the return value of function where we assign the pattern according to the pattern sent. And we are done that process.

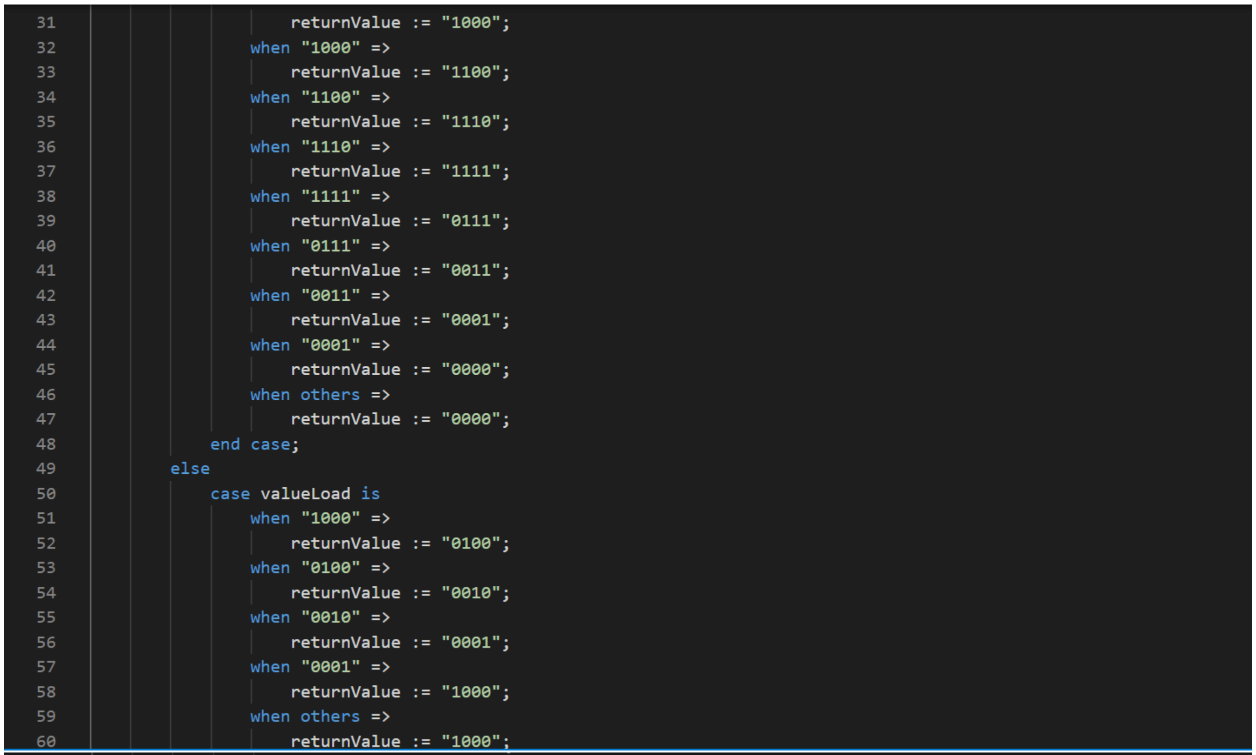
In the next process we will check for the changes in value of timer, we will check if the timer is in rising edge, and then we will increment the counter. We will check if the counter will equal to the decimal value of 49,999,999 and if that is true we will reset the counter back to zero, and assign clock out to zero. Else if we will check if the counter is equal to the decimal value of 49,999,998 and if true we will assign clock out to one. Then we will check if the switch for timer is turned on and if true we will check if the clock out is one if that is true we will check if timer load value is equal to 0000 and the switch for Johnson is not turned on we will set the value of timer load value to 1000. After wards we will set the value of timer load value to the value returned from the function where the pattern is set by supplying the value of the current load Value. And we end that process.

With the final function we will check for the changes in the value of clock load value or timer load value and set the value of load value to either clock load value or timer load value if the switch for timer is turned off or on respectively. Then we will set the value of the seven segment display from the return value of the function which is written to set the seven segment display.

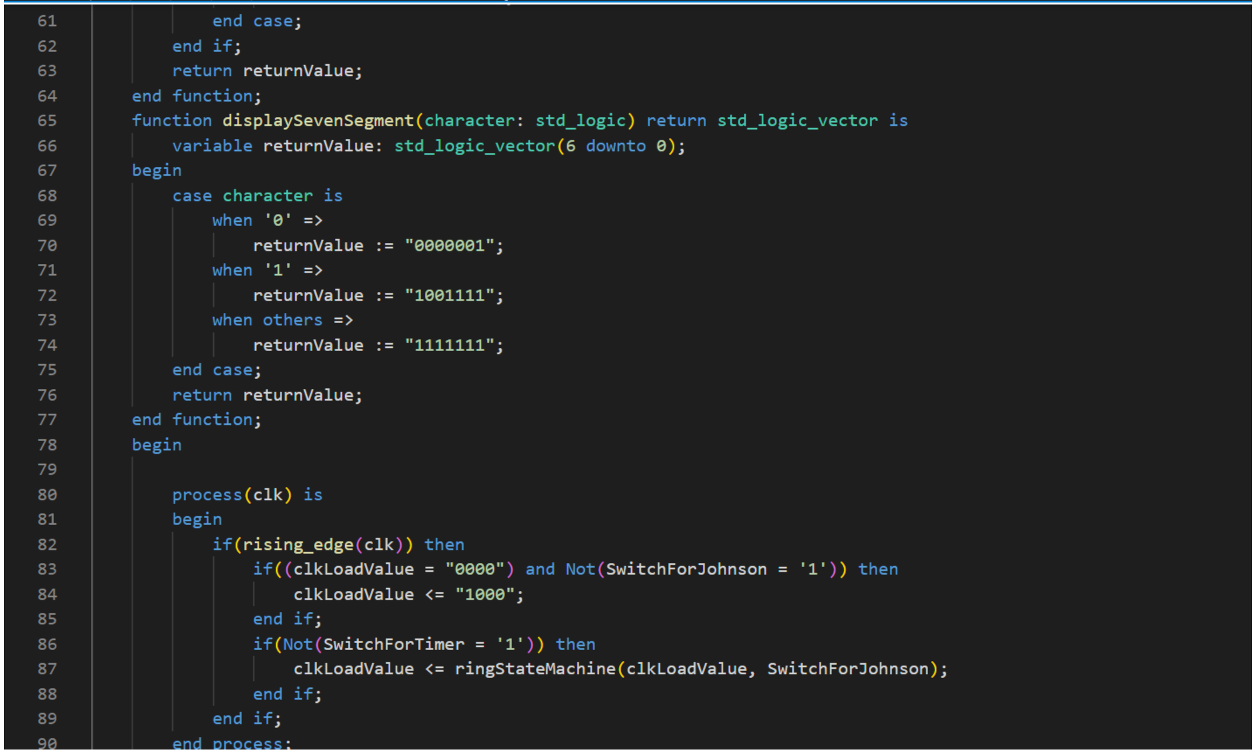
VHDL program screenshot:



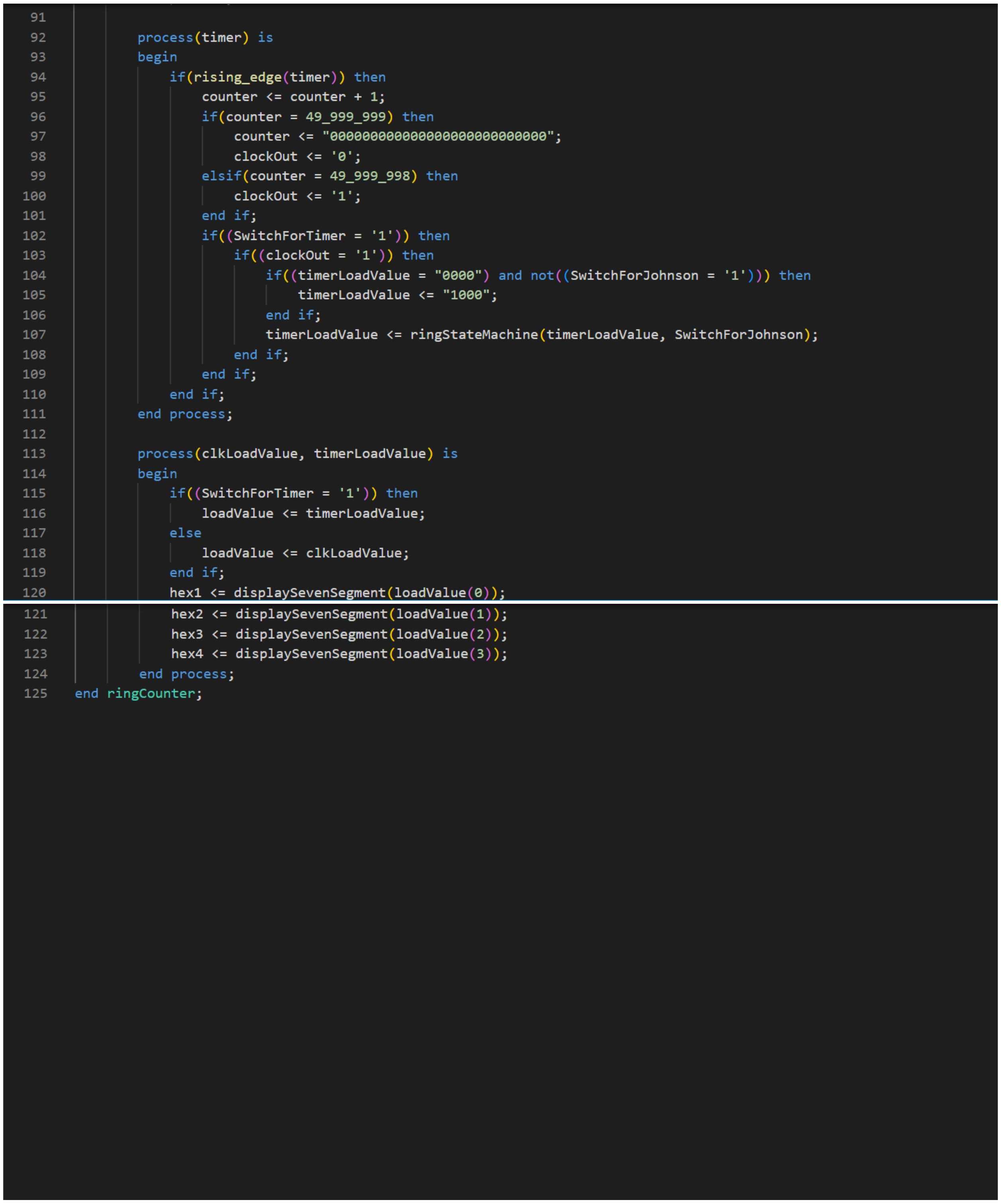
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
entity rgrewallab9VHDL is
    port(
        clk: in std_logic;
        timer: in std_logic;
        SwitchForJohnson: in std_logic;
        SwitchForTimer: in std_logic;
        hex1: out std_logic_vector(6 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)
    );
end rgrewallab9VHDL;

architecture ringCounter of rgrewallab9VHDL is
    signal clkLoadValue: std_logic_vector(3 downto 0) := "1000";
    signal timerLoadValue: std_logic_vector(3 downto 0) := "0000";
    signal loadValue: std_logic_vector(3 downto 0);
    signal counter: std_logic_vector(0 to 26);
    signal clockOut: std_logic := '0';
    function ringStateMachine(valueLoad: std_logic_vector(3 downto 0); SwitchForJohnson: std_logic)
    return std_logic_vector is
        variable returnValue: std_logic_vector(3 downto 0);
    begin
        if((SwitchForJohnson = '1')) then
            case valueLoad is
                when "0000" =>
```



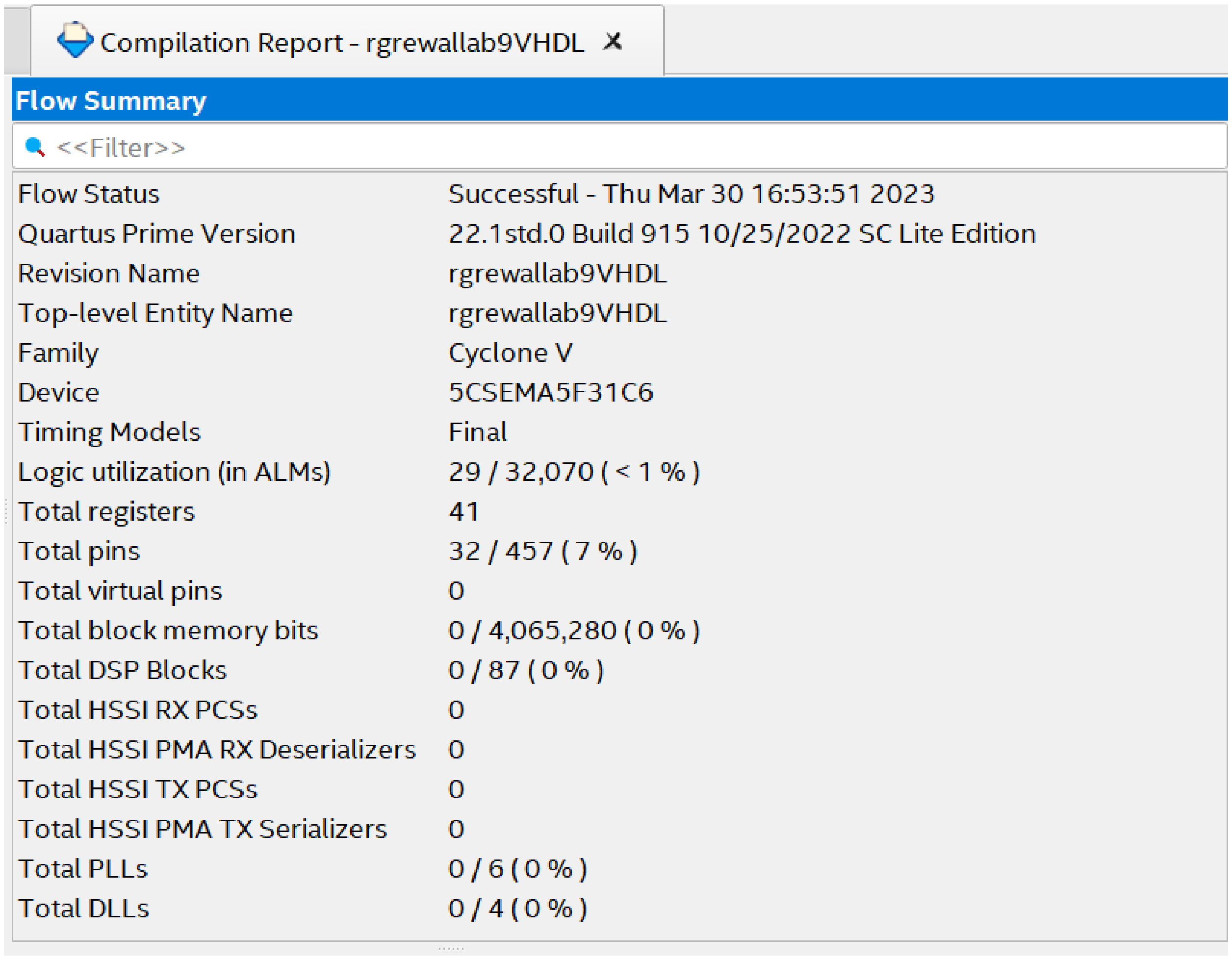
```
                returnValue := "1000";
            when "1000" =>
                returnValue := "1100";
            when "1100" =>
                returnValue := "1110";
            when "1110" =>
                returnValue := "1111";
            when "1111" =>
                returnValue := "0111";
            when "0111" =>
                returnValue := "0011";
            when "0011" =>
                returnValue := "0001";
            when "0001" =>
                returnValue := "0000";
            when others =>
                returnValue := "0000";
        end case;
    else
        case valueLoad is
            when "1000" =>
                returnValue := "0100";
            when "0100" =>
                returnValue := "0010";
            when "0010" =>
                returnValue := "0001";
            when "0001" =>
                returnValue := "1000";
            when others =>
                returnValue := "1000";
```



```
        end case;
    end if;
    return returnValue;
end function;
function displaySevenSegment(character: std_logic) return std_logic_vector is
    variable returnValue: std_logic_vector(6 downto 0);
begin
    case character is
        when '0' =>
            returnValue := "0000001";
        when '1' =>
            returnValue := "1001111";
        when others =>
            returnValue := "1111111";
    end case;
    return returnValue;
end function;
begin

    process(clk) is
    begin
        if(rising_edge(clk)) then
            if((clkLoadValue = "0000") and Not(SwitchForJohnson = '1')) then
                clkLoadValue <= "1000";
            end if;
            if(Not(SwitchForTimer = '1')) then
                clkLoadValue <= ringStateMachine(clkLoadValue, SwitchForJohnson);
            end if;
        end if;
    end process;
```


```

        process(timer) is
        begin
            if(rising_edge(timer)) then
                counter <= counter + 1;
                if(counter = 49_999_999) then
                    counter <= "00000000000000000000000000";
                    clockOut <= '0';
                elsif(counter = 49_999_998) then
                    clockOut <= '1';
                end if;
                if((SwitchForTimer = '1')) then
                    if((clockOut = '1')) then
                        if((timerLoadValue = "0000") and not((SwitchForJohnson = '1'))) then
                            timerLoadValue <= "1000";
                        end if;
                        timerLoadValue <= ringStateMachine(timerLoadValue, SwitchForJohnson);
                    end if;
                end if;
            end if;
        end process;

        process(clkLoadValue, timerLoadValue) is
        begin
            if((SwitchForTimer = '1')) then
                loadValue <= timerLoadValue;
            else
                loadValue <= clkLoadValue;
            end if;
            hex1 <= displaySevenSegment(loadValue(0));
            hex2 <= displaySevenSegment(loadValue(1));
            hex3 <= displaySevenSegment(loadValue(2));
            hex4 <= displaySevenSegment(loadValue(3));
        end process;
end ringCounter;
```

VHDL compilation report:



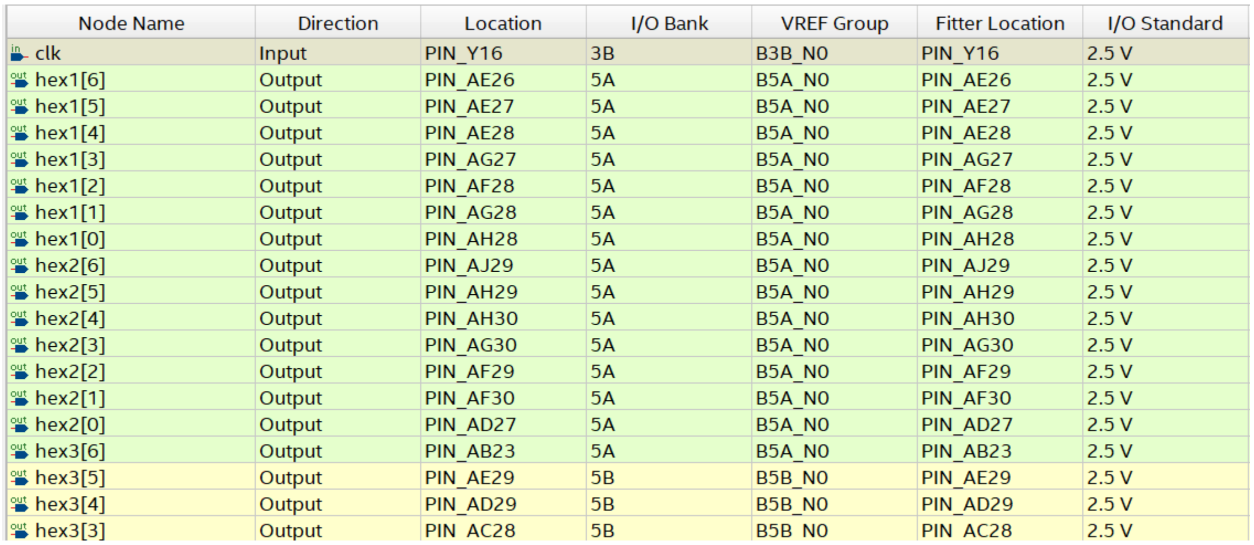
Compilation Report - rgrewallab9VHDL

Flow Summary

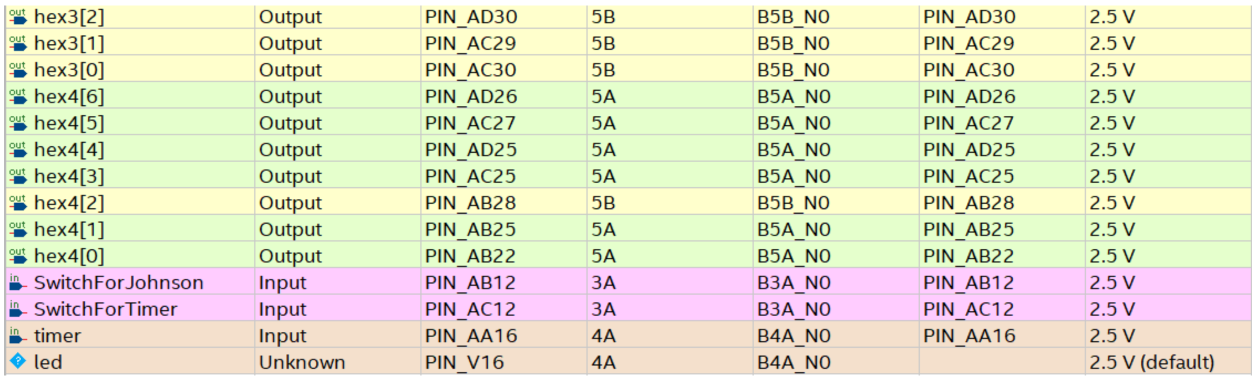
<<Filter>>

Flow Status	Successful - Thu Mar 30 16:53:51 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	rgrewallab9VHDL
Top-level Entity Name	rgrewallab9VHDL
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	29 / 32,070 (< 1 %)
Total registers	41
Total pins	32 / 457 (7 %)
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 %)

Pin Planner screenshot:



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
clk	Input	PIN_Y16	3B	B3B_N0	PIN_Y16	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
SwitchForJohnson	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
SwitchForTimer	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
timer	Input	PIN_AA16	4A	B4A_N0	PIN_AA16	2.5 V
led	Unknown	PIN_V16	4A	B4A_N0		2.5 V (default)

Verilog introduction:

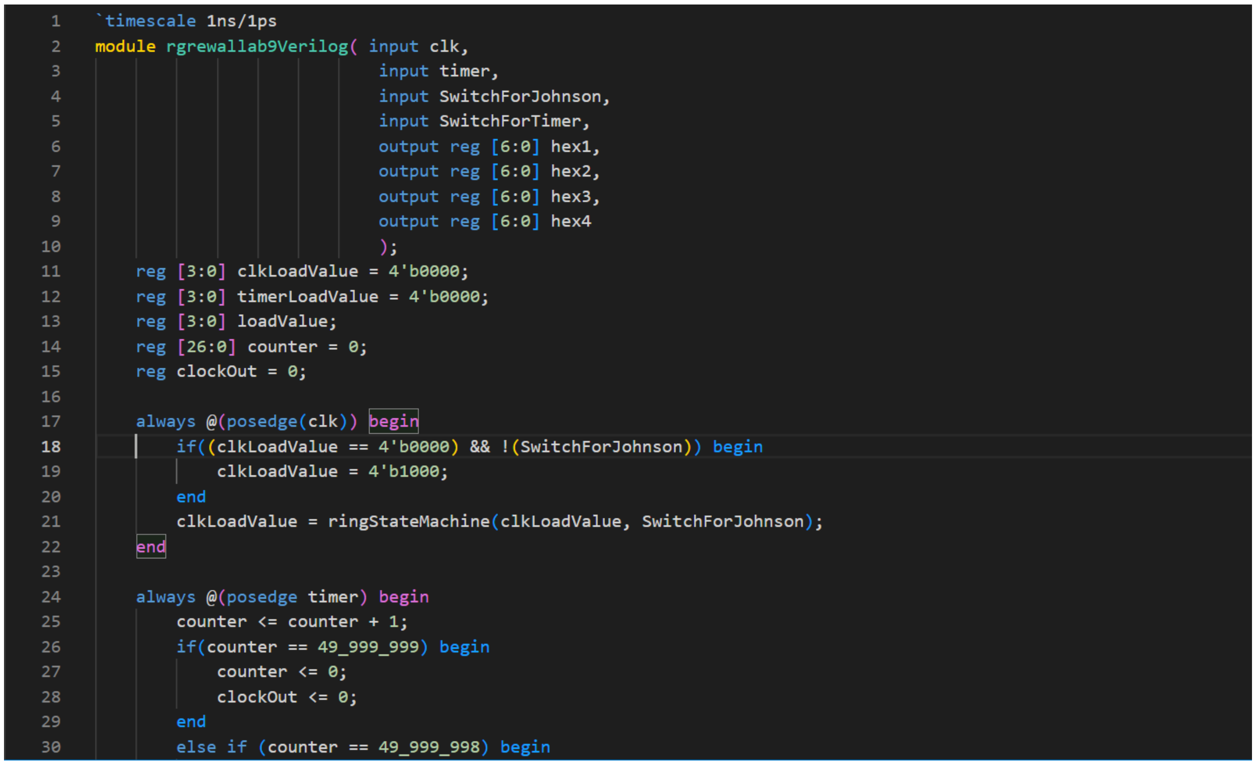
For the program in Verilog, we will need to initialize the module with the parameters having the following inputs, first we will have a clock, timer, switch for Johnson, switch for timer. For the output we will have seven element arrays for the seven-segment display. Within the module we will have the array of four element clock load value, we will have the array of four element for the timer load value, and we will have an array of four element that stores the load value. We will have array of 26 element of counter. We will also initialize a variable clock out.

Then we will initialize always where the positive edge of the clock is checked. And we will check if the clock load value is equal to 0000 and the switch for johnson is not turned on. And if that is true than we will set the clock load value to 1000. Afterwards we will assign the value of clock load value to the return value of the function which sets the next value of the current value of clock load value.

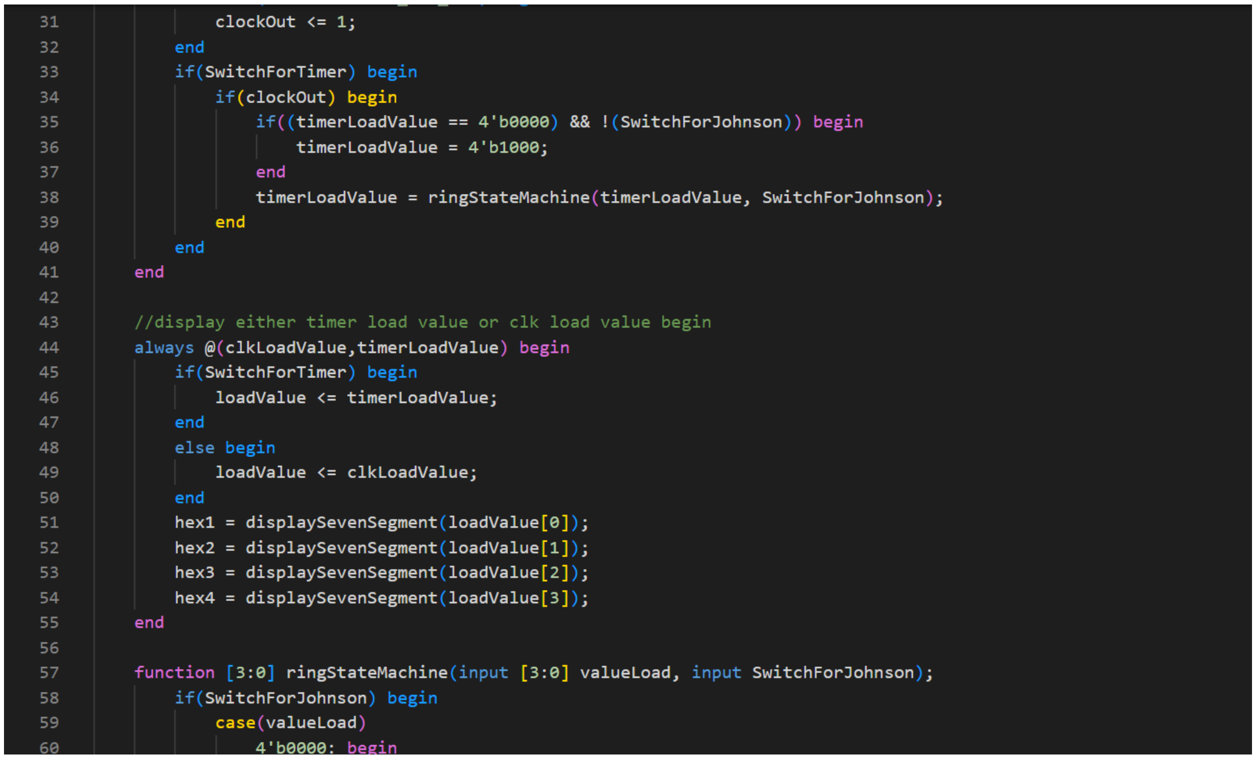
In the next process where we will check of the positive edge of the timer. We will increment the counter by one. And if the counter value equals the decimal value of 49,999,999 we will set the value of counter to zero and clock out to the value of zero. Else if we will check if the counter will equal to 49,999,998 we will set the clock out value to 1. Than we will check of switch for timer is turned on we will than check if the vale of clock out is true , we will finally check if the value of timer load value is equal to 0000, and if the switch for johnson is turned off. And if its true we will assign the value of timer load value to equal 1000. And afterwards we will assign the timer load value to the return of the function which is returned to the next value.

In the final process we will check for the changes in either value clock load value or timer load value and we will set the load value to either timer load value or clock load value if the switch for the timer is turned on. And than we will set the seven segment display to the value of the return value of function of the display seven segment which sets the pattern for the character provided. Than we will write the function afterwards.

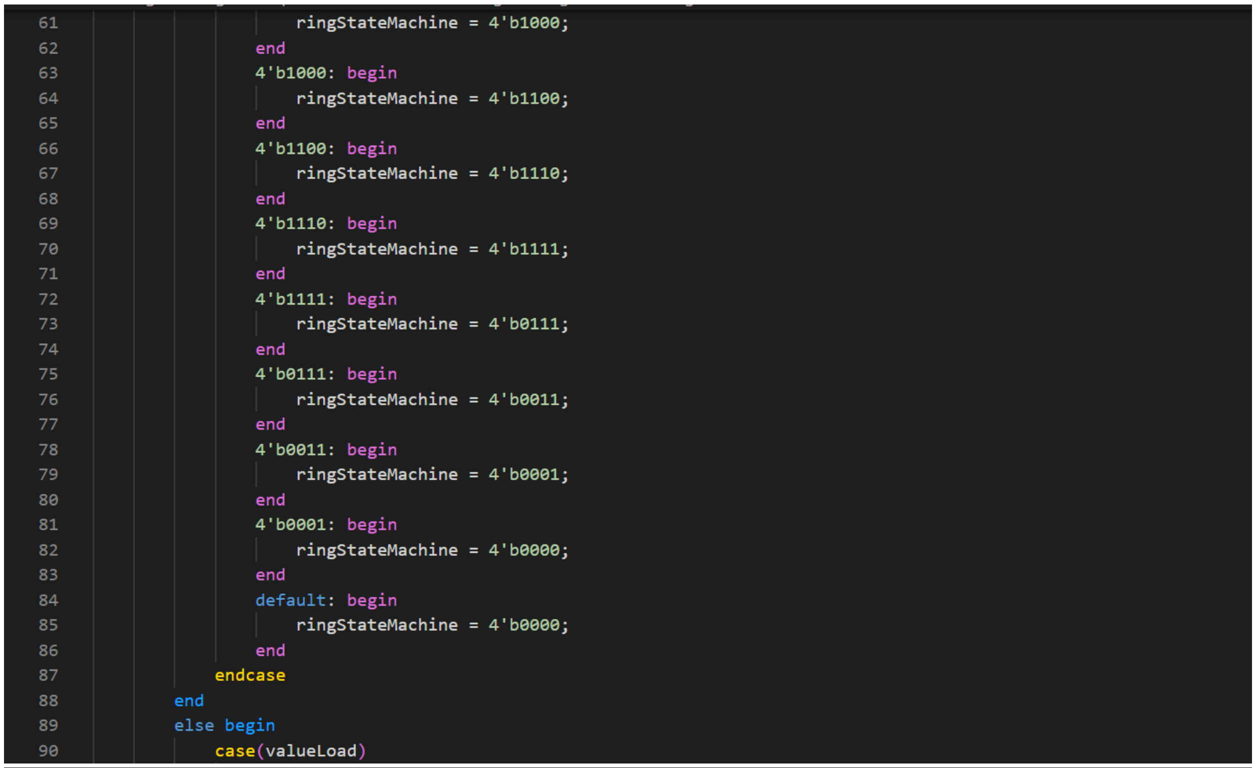
Verilog program screenshot:



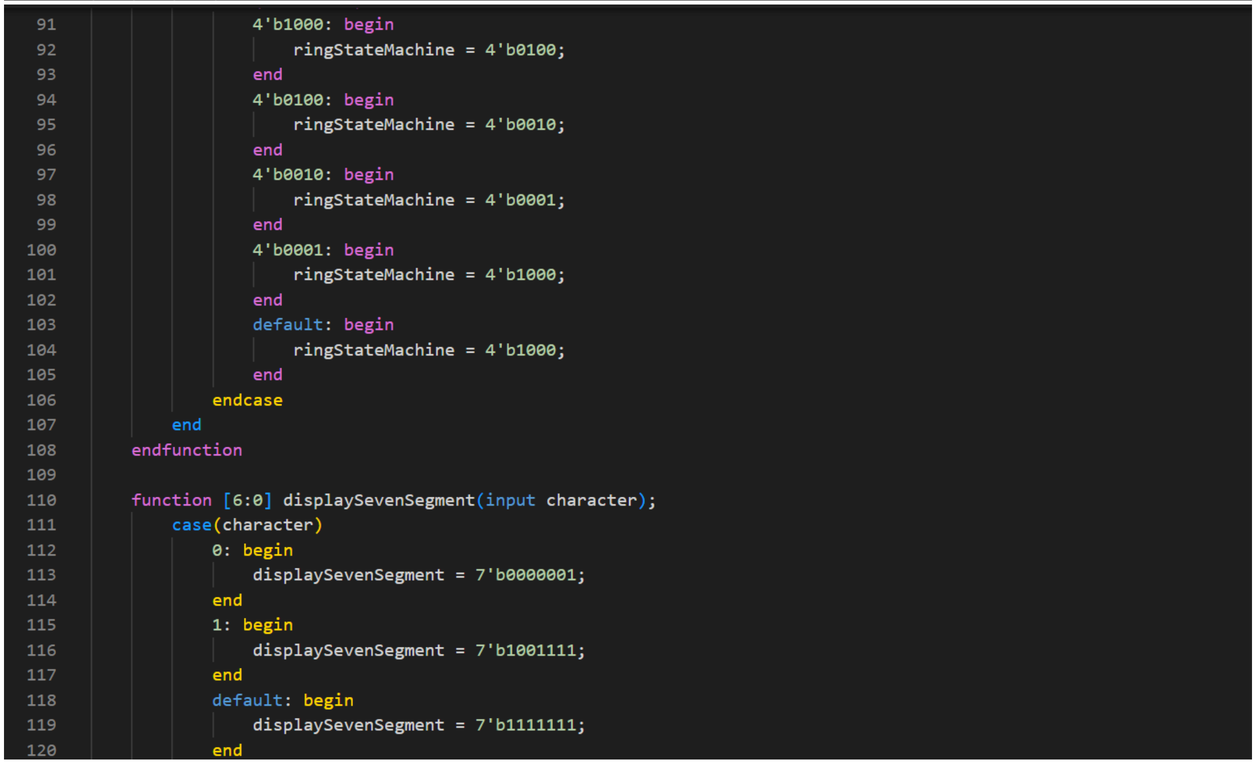
```verilog
`timescale 1ns/1ps
module rgrewallab9Verilog( input clk,
                           input timer,
                           input SwitchForJohnson,
                           input SwitchForTimer,
                           output reg [6:0] hex1,
                           output reg [6:0] hex2,
                           output reg [6:0] hex3,
                           output reg [6:0] hex4
                           );
    reg [3:0] clkLoadValue = 4'b0000;
    reg [3:0] timerLoadValue = 4'b0000;
    reg [3:0] loadValue;
    reg [26:0] counter = 0;
    reg clockOut = 0;

    always @(posedge(clk)) begin
        if((clkLoadValue == 4'b0000) && !(SwitchForJohnson)) begin
            clkLoadValue = 4'b1000;
        end
        clkLoadValue = ringStateMachine(clkLoadValue, SwitchForJohnson);
    end

    always @(posedge timer) begin
        counter <= counter + 1;
        if(counter == 49_999_999) begin
            counter <= 0;
            clockOut <= 0;
        end
        else if (counter == 49_999_998) begin
```



```verilog
            clockOut <= 1;
        end
        if(SwitchForTimer) begin
            if(clockOut) begin
                if((timerLoadValue == 4'b0000) && !(SwitchForJohnson)) begin
                    timerLoadValue = 4'b1000;
                end
                timerLoadValue = ringStateMachine(timerLoadValue, SwitchForJohnson);
            end
        end
    end

    //display either timer load value or clk load value begin
    always @(clkLoadValue,timerLoadValue) begin
        if(SwitchForTimer) begin
            loadValue <= timerLoadValue;
        end
        else begin
            loadValue <= clkLoadValue;
        end
        hex1 = displaySevenSegment(loadValue[0]);
        hex2 = displaySevenSegment(loadValue[1]);
        hex3 = displaySevenSegment(loadValue[2]);
        hex4 = displaySevenSegment(loadValue[3]);
    end

    function [3:0] ringStateMachine(input [3:0] valueLoad, input SwitchForJohnson);
        if(SwitchForJohnson) begin
            case(valueLoad)
                4'b0000: begin
```



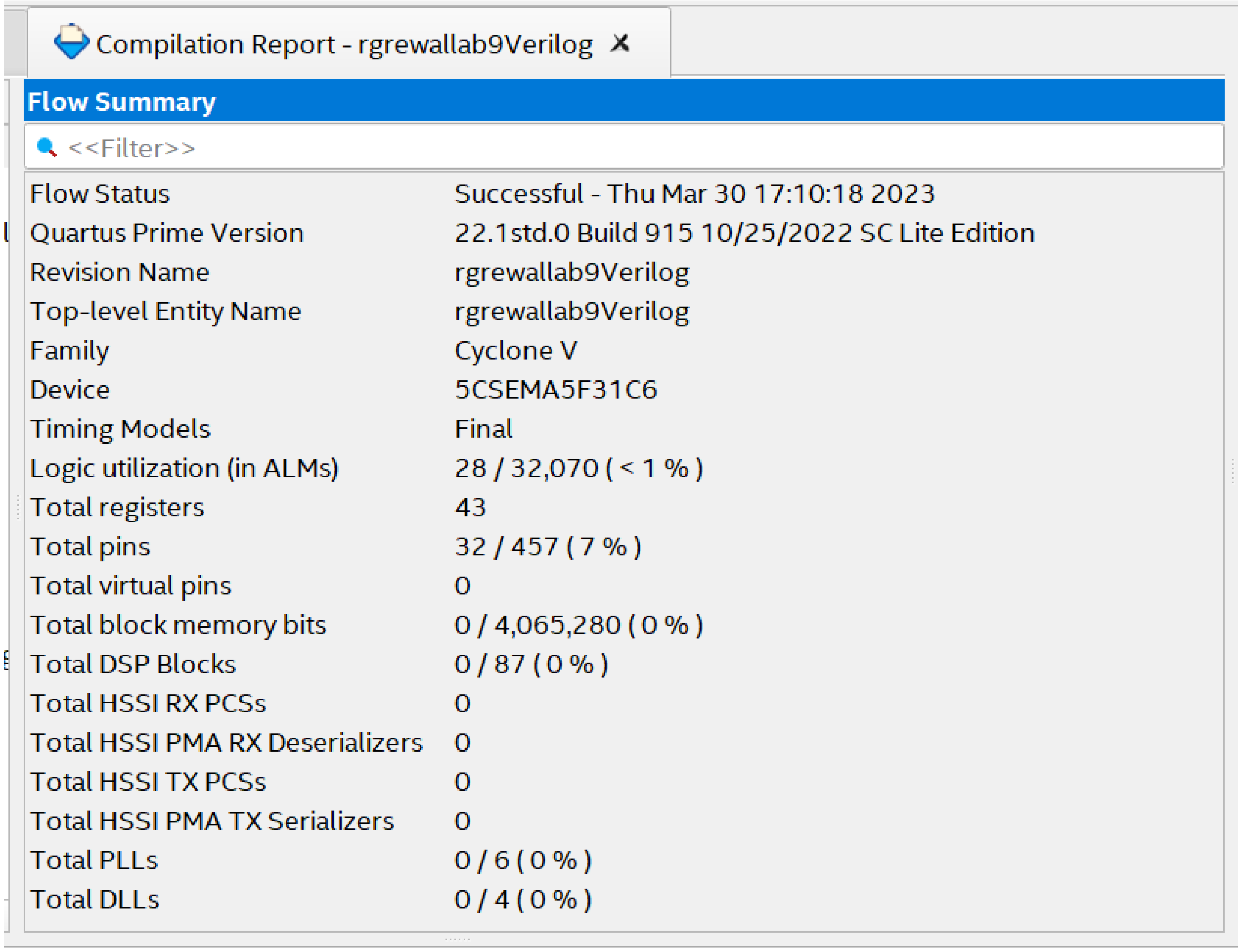
```
                    ringStateMachine = 4'b1000;
                end
                4'b1000: begin
                    ringStateMachine = 4'b1100;
                end
                4'b1100: begin
                    ringStateMachine = 4'b1110;
                end
                4'b1110: begin
                    ringStateMachine = 4'b1111;
                end
                4'b1111: begin
                    ringStateMachine = 4'b0111;
                end
                4'b0111: begin
                    ringStateMachine = 4'b0011;
                end
                4'b0011: begin
                    ringStateMachine = 4'b0001;
                end
                4'b0001: begin
                    ringStateMachine = 4'b0000;
                end
                default: begin
                    ringStateMachine = 4'b0000;
                end
            endcase
        end
        else begin
            case(valueLoad)
```



```
                4'b1000: begin
                    ringStateMachine = 4'b0100;
                end
                4'b0100: begin
                    ringStateMachine = 4'b0010;
                end
                4'b0010: begin
                    ringStateMachine = 4'b0001;
                end
                4'b0001: begin
                    ringStateMachine = 4'b1000;
                end
                default: begin
                    ringStateMachine = 4'b1000;
                end
            endcase
        end
    endfunction

    function [6:0] displaySevenSegment(input character);
        case(character)
            0: begin
                displaySevenSegment = 7'b0000001;
            end
            1: begin
                displaySevenSegment = 7'b1001111;
            end
            default: begin
                displaySevenSegment = 7'b1111111;
            end
```

```verilog
        endcase
    endfunction
endmodule
```

Verilog compilation report:



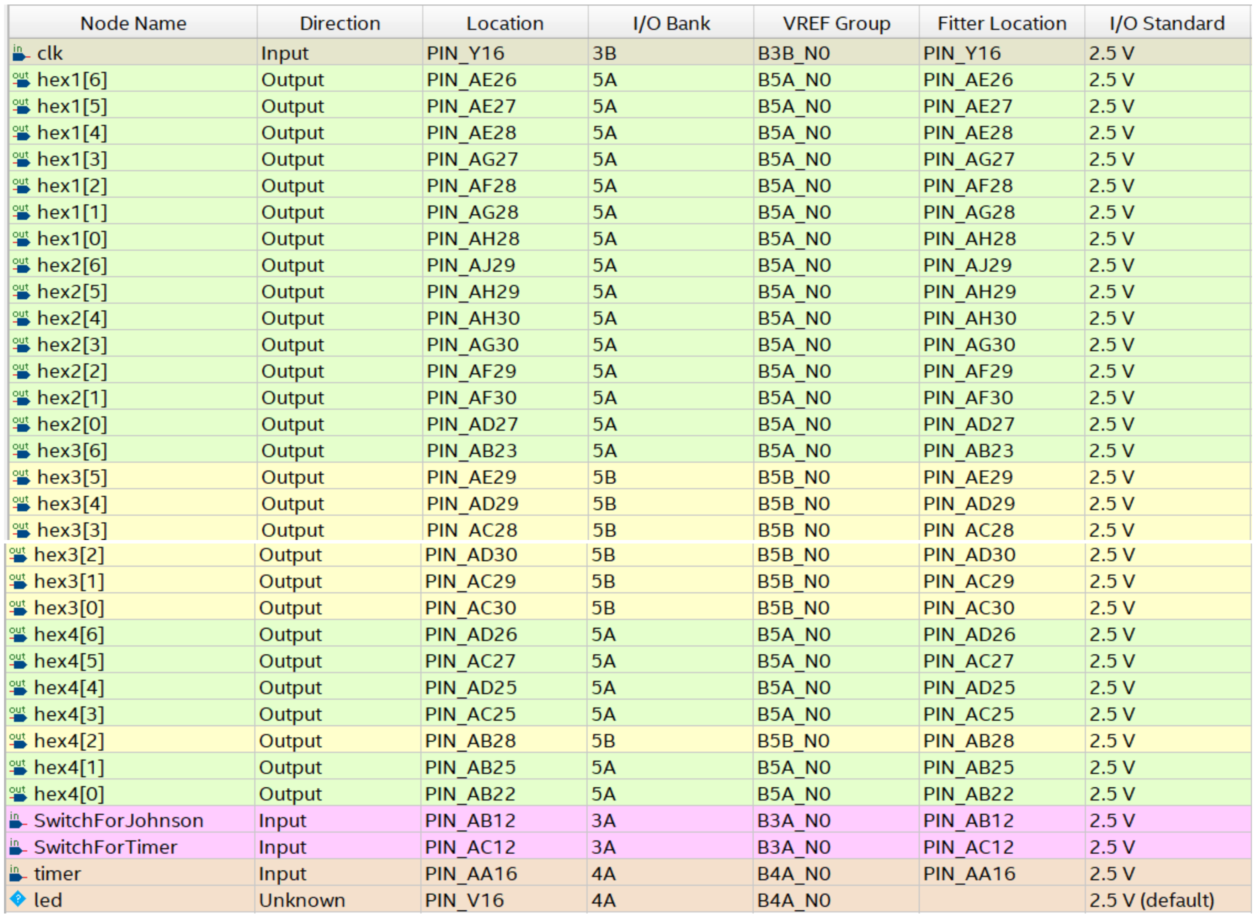
Compilation Report - rgrewallab9Verilog

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Mar 30 17:10:18 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	rgrewallab9Verilog
Top-level Entity Name	rgrewallab9Verilog
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	28 / 32,070 (< 1 %)
Total registers	43
Total pins	32 / 457 (7 %)
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
clk	Input	PIN_Y16	3B	B3B_N0	PIN_Y16	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
SwitchForJohnson	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
SwitchForTimer	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
timer	Input	PIN_AA16	4A	B4A_N0	PIN_AA16	2.5 V
led	Unknown	PIN_V16	4A	B4A_N0		2.5 V (default)

Conclusion:

Thus we can conclude that by both the programming languages we can change the pattern of the ring counter or the johnson ring counter either using the inbuilt clock or a push button by a switch case that we inbuilt into a function to get the next pattern symbol. The program can be consider a state machine where by the input of the clock or the push button can change the state to the next state.