Digital Design Principles

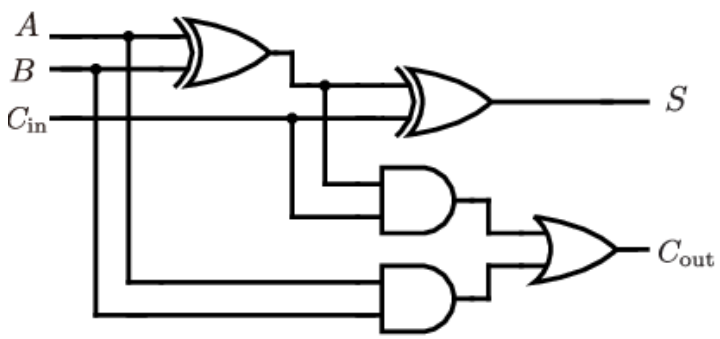
Full Adder

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

Objective:

- I use VHDL to implement simple 4-bit Full Adder.
- Set up a project in quartus || targeting your FPGA board.

Truth Table:



Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VHDL Introduction:

To make the program in VHDL we will need to initialize the library and entity. Within the entity we will state the vector of 4 switch for a and b. We will also add a switch for carry in and another switch to change from half adder and full adder. For the output we will need a vector to store for the two 7-segment led, and the vector of 4 led for Sum and Carry. Within the architecture of the program, we will have a case where the state switch is checked if it's on and if it's on the code of the full adder will run and if it's not on the code for the half adder will one. If the full adder code is running the first seven segment led will light up F and the second one

will light up A. When the half adder is running the first seven segment led will light up H and the second one will light up A. The rest of the program will calculate the Sum and the Carry according to the way the half adder/ Full adder works.

VHDL program screenshot:

VHDL > rgrewallab6vhdl.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rgrewallab6vhdl is
    port(
        A: in std_logic_vector(3 downto 0);
        B: in std_logic_vector(3 downto 0);
        C: in std_logic;
        State: in std_logic;
        Hex1: out std_logic_vector(6 downto 0);
        Hex2: out std_logic_vector(6 downto 0);
        Sum: out std_logic_vector(3 downto 0);
        Carry: out std_logic_vector(3 downto 0)
    );
end rgrewallab6vhdl;

architecture adder of rgrewallab6vhdl is
    signal carryin: std_logic_vector(3 downto 0);
    begin
        process(A,B,C,State) is
            begin
                case State is
                    when '1' =>
                        Hex1 <= "0111000";
                        Hex2 <= "0001000";
                        carryin(0) <= C;
                        Sum(0) <= (A(0) XOR B(0)) XOR carryin(0);
                        carryin(1) <= ((A(0) XOR B(0)) and carryin(0)) or (A(0) and B(0));

                        Sum(1) <= (A(1) XOR B(1)) XOR carryin(1);
                        carryin(2) <= ((A(1) xor B(1)) and carryin(1)) or (A(1) and B(1));

                        Sum(2) <= (A(2) XOR B(2)) XOR carryin(2);
                        carryin(3) <= ((A(2) xor B(2)) and carryin(2)) or (A(2) and B(2));

                        Sum(3) <= (A(3) XOR B(3)) XOR carryin(3);
                        Carry(0) <= ((A(3) xor B(3)) and carryin(3)) or (A(3) and B(3));


                    when '0' =>
                        Hex1 <= "1001000";
                        Hex2 <= "0001000";

                        Sum(0) <= A(0) XOR B(0);
                        Carry(0) <= A(0) AND B(0);

                        Sum(1) <= A(1) XOR B(1);
                        Carry(1) <= A(1) AND B(1);

                        Sum(2) <= A(2) XOR B(2);
                        Carry(2) <= A(2) AND B(2);

                        Sum(3) <= A(3) XOR B(3);
                        Carry(3) <= A(3) AND B(3);
                    when others =>
                        Sum <= "0000";
                        Carry <= "0000";
                end case;
        end process;
end adder;
```

VHDL compilation report:

Compilation Report - rgrewallab6vhdl

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Mar 9 17:43:50 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	rgrewallab6vhdl
Top-level Entity Name	rgrewallab6vhdl
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	10 / 32,070 (< 1 %)
Total registers	0
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:

A[3]	Input	PIN_AE12	3A	B3A_N0	PIN_AE12	2.5 V
A[2]	Input	PIN_AD10	3A	B3A_N0	PIN_AD10	2.5 V
A[1]	Input	PIN_AC9	3A	B3A_N0	PIN_AC9	2.5 V
A[0]	Input	PIN_AE11	3A	B3A_N0	PIN_AE11	2.5 V
B[3]	Input	PIN_AD12	3A	B3A_N0	PIN_AD12	2.5 V
B[2]	Input	PIN_AD11	3A	B3A_N0	PIN_AD11	2.5 V
B[1]	Input	PIN_AF10	3A	B3A_N0	PIN_AF10	2.5 V
B[0]	Input	PIN_AF9	3A	B3A_N0	PIN_AF9	2.5 V
C	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
Carry[3]	Output	PIN_V18	4A	B4A_N0	PIN_V18	2.5 V
Carry[2]	Output	PIN_V17	4A	B4A_N0	PIN_V17	2.5 V
Carry[1]	Output	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V
Carry[0]	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V
Hex1[6]	Output	PIN_V25	5B	B5B_N0	PIN_V25	2.5 V
Hex1[5]	Output	PIN_AA28	5B	B5B_N0	PIN_AA28	2.5 V
Hex1[4]	Output	PIN_Y27	5B	B5B_N0	PIN_Y27	2.5 V
Hex1[3]	Output	PIN_AB27	5B	B5B_N0	PIN_AB27	2.5 V
Hex1[2]	Output	PIN_AB26	5A	B5A_N0	PIN_AB26	2.5 V
Hex1[1]	Output	PIN_AA26	5B	B5B_N0	PIN_AA26	2.5 V
Hex1[0]	Output	PIN_AA25	5A	B5A_N0	PIN_AA25	2.5 V
Hex2[6]	Output	PIN_AA24	5A	B5A_N0	PIN_AA24	2.5 V
Hex2[5]	Output	PIN_Y23	5A	B5A_N0	PIN_Y23	2.5 V
Hex2[4]	Output	PIN_Y24	5A	B5A_N0	PIN_Y24	2.5 V
Hex2[3]	Output	PIN_W22	5A	B5A_N0	PIN_W22	2.5 V
Hex2[2]	Output	PIN_W24	5A	B5A_N0	PIN_W24	2.5 V
Hex2[1]	Output	PIN_V23	5A	B5A_N0	PIN_V23	2.5 V
Hex2[0]	Output	PIN_W25	5B	B5B_N0	PIN_W25	2.5 V
State	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
Sum[3]	Output	PIN_Y21	5A	B5A_N0	PIN_Y21	2.5 V
Sum[2]	Output	PIN_W21	5A	B5A_N0	PIN_W21	2.5 V
Sum[1]	Output	PIN_W20	5A	B5A_N0	PIN_W20	2.5 V
Sum[0]	Output	PIN_Y19	4A	B4A_N0	PIN_Y19	2.5 V

Verilog introduction:

For the program in Verilog we will need to initialize the module with the parameters having we will state the vector of 4 switches for a and b. We will also add a switch for carry in and another switch to change from half adder and full adder. For the output we will need a vector to store for the two 7-segment led, and the vector of 4 led for Sum and Carry. We will have a case where the state switch is checked if it's on and if it's on the code of the full adder will run and if it's not on the code for the half adder will one. If the full adder code is running the first seven segment led will light up F and the second one will light up A. When the half adder is running the first seven segment led will light up H and the second one will light up A. The rest of the program will calculate the Sum and the Carry according to the way the half adder/ Full adder works.

.

Verilog program screenshot:

Verilog > rgrewallab6verilog.v

```
module rgrewallab6verilog(input [3:0]A,
                          input [3:0]B,
                          input C,
                          input State,
                          output reg [6:0] hex1,
                          output reg [6:0] hex2,
                          output reg[3:0] sum,
                          output reg[3:0] carry);
        integer i;
        reg carryin;
        reg [1:0] adder;
        always @(A,B,C,State) begin
            carryin = C;
            case(State)
                1'b1:begin
                    hex1 = 7'b0111000;
                    hex2 = 7'b0001000;
                    for(i = 0;i<4;i =  i+1) begin
                        adder =  fullAdder(A[i],B[i],carryin);
                        carryin = adder[1];
                        sum[i] = adder[0];
                    end
                    carry[0] = carryin;
                end
                1'b0: begin
                    hex1 = 7'b1001000;
                    hex2 = 7'b0001000;
                    for(i =0;i<4;i=i+1) begin
                        adder = halfAdder(A[i],B[i]);
                        sum[i] = adder[0];
```

```
                    carry[i] = adder[1];
                end
            end
            default: begin
                sum = 4'b0000;
                carry = 4'b0000;
            end
        endcase
    end
function [1:0] halfAdder(input A,B);
    reg Sum = A ^ B;
    reg Carry = A && B;
    halfAdder[0] = Sum;
    halfAdder[1] = Carry;
endfunction
function [1:0] fullAdder(input A,B,Carryin);
    reg Sum = A ^ B ^ Carryin;
    reg Carryout = (A ^ B && Carryin) || (A && B);
    fullAdder[0] = Sum;
    fullAdder[1] = Carryout;
endfunction
endmodule
```

Verilog compilation report:

Compilation Report - rgrewallab6verilog

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Mar 9 17:41:14 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	rgrewallab6verilog
Top-level Entity Name	rgrewallab6verilog
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	9 / 32,070 (< 1 %)
Total registers	0
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:

A[3]	Input	PIN_AE12	3A	B3A_N0	PIN_AE12	2.5 V
A[2]	Input	PIN_AD10	3A	B3A_N0	PIN_AD10	2.5 V
A[1]	Input	PIN_AC9	3A	B3A_N0	PIN_AC9	2.5 V
A[0]	Input	PIN_AE11	3A	B3A_N0	PIN_AE11	2.5 V
B[3]	Input	PIN_AD12	3A	B3A_N0	PIN_AD12	2.5 V
B[2]	Input	PIN_AD11	3A	B3A_N0	PIN_AD11	2.5 V
B[1]	Input	PIN_AF10	3A	B3A_N0	PIN_AF10	2.5 V
B[0]	Input	PIN_AF9	3A	B3A_N0	PIN_AF9	2.5 V
C	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
State	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
carry[3]	Output	PIN_V18	4A	B4A_N0	PIN_V18	2.5 V
carry[2]	Output	PIN_V17	4A	B4A_N0	PIN_V17	2.5 V
carry[1]	Output	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V
carry[0]	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V
hex1[6]	Output	PIN_V25	5B	B5B_N0	PIN_V25	2.5 V
hex1[5]	Output	PIN_AA28	5B	B5B_N0	PIN_AA28	2.5 V
hex1[4]	Output	PIN_Y27	5B	B5B_N0	PIN_Y27	2.5 V
hex1[3]	Output	PIN_AB27	5B	B5B_N0	PIN_AB27	2.5 V

hex1[4]	Output	PIN_Y27	5B	B5B_N0	PIN_Y27	2.5 V
hex1[3]	Output	PIN_AB27	5B	B5B_N0	PIN_AB27	2.5 V
hex1[2]	Output	PIN_AB26	5A	B5A_N0	PIN_AB26	2.5 V
hex1[1]	Output	PIN_AA26	5B	B5B_N0	PIN_AA26	2.5 V
hex1[0]	Output	PIN_AA25	5A	B5A_N0	PIN_AA25	2.5 V
hex2[6]	Output	PIN_AA24	5A	B5A_N0	PIN_AA24	2.5 V
hex2[5]	Output	PIN_Y23	5A	B5A_N0	PIN_Y23	2.5 V
hex2[4]	Output	PIN_Y24	5A	B5A_N0	PIN_Y24	2.5 V
hex2[3]	Output	PIN_W22	5A	B5A_N0	PIN_W22	2.5 V
hex2[2]	Output	PIN_W24	5A	B5A_N0	PIN_W24	2.5 V
hex2[1]	Output	PIN_V23	5A	B5A_N0	PIN_V23	2.5 V
hex2[0]	Output	PIN_W25	5B	B5B_N0	PIN_W25	2.5 V
sum[3]	Output	PIN_Y21	5A	B5A_N0	PIN_Y21	2.5 V
sum[2]	Output	PIN_W21	5A	B5A_N0	PIN_W21	2.5 V
sum[1]	Output	PIN_W20	5A	B5A_N0	PIN_W20	2.5 V
sum[0]	Output	PIN_Y19	4A	B4A_N0	PIN_Y19	2.5 V

Conclusion:

Thus we can conclude that by both the programming languages we will have the led light up according to if the full adder is followed or the half adder is followed. The two seven-segment led will light up according to which state the fpga is in.