EE587 - GATE-LEVEL IMPLEMENTATION OF FULL ADDER AND SR LATCH USING VERILOG

G.A.Y.L. DHARMARATHNE

E/19/075

EE.19.587.02

SEMESTER 7

18/12/2024

Exercise 1 : Implementing a 1-bit full adder in Verilog

1. Truth table of the full adder

Table 01:Truth table of the 1bit full adder

A	B	Carry_in	Sum	Carry_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

2. Logic gate diagram of the full adder

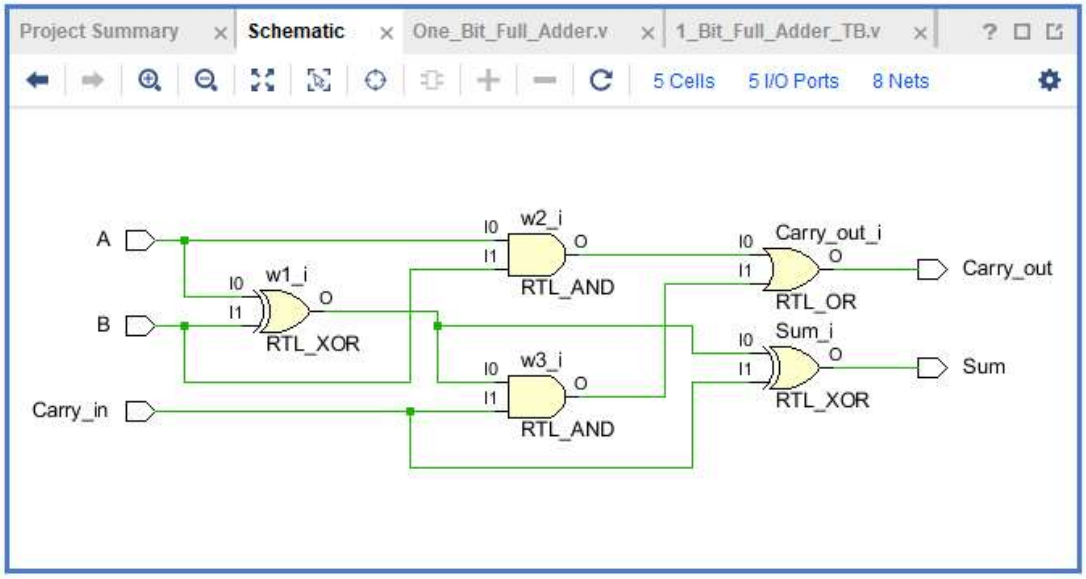

Figure 01: Logic gate diagram of the 1bit full adder

3. Module code and testbench code for the full adder

Module code

```
`timescale 1ns / 1ps


module full_adder(
    input A,
    input B,
    input Carry_in,
    output Sum,
    output Carry_out
);

// Internal wires
wire w1, w2, w3;

// Gate-level implementation
xor (w1, A, B);
xor (Sum, w1, Carry_in);
and (w2, A, B);
and (w3, w1, Carry_in);
or  (Carry_out, w2, w3);

endmodule
```

Test Bench code

```
`timescale 1ns / 1ps

module full_adder_TB;

reg A, B, Carry_in;
wire Sum, Carry_out;

full_adder dut (
    .A(A),
    .B(B),
    .Carry_in(Carry_in),
    .Sum(Sum),
    .Carry_out(Carry_out)
);

initial begin
    A = 1; B = 1; Carry_in = 1; #10;
    A = 0; B = 0; Carry_in = 0; #10;
    A = 1; B = 1; Carry_in = 0; #10;
    A = 0; B = 0; Carry_in = 1; #10;
    $finish;
end
endmodule
```

4. Waveforms of the behavioral simulation

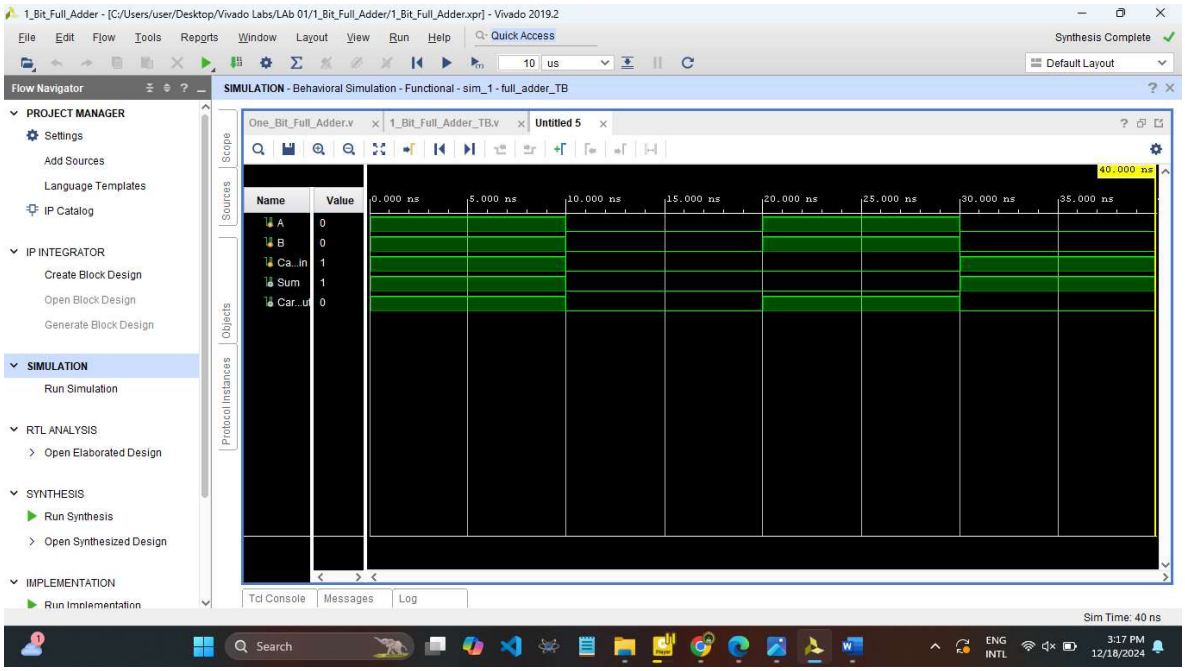

Figure 02:Waveforms of the 1bit full adder behavioral simulation

Exercise 2 : Implementing a SR Latch in Verilog

1. Module code and testbench code for the SR flipflop

Module code

```
`timescale 1ns / 1ps

module sr_latch(
    input S,
    input R,
    output Q,
    output Q_bar
);

// Gate-level implementation
wire S_bar, R_bar;
not(R_bar, R);
not(S_bar, S);
nand (Q, S_bar, Q_bar);
nand (Q_bar, R_bar, Q);

endmodule
```

Test bench

```
`timescale 1ns / 1ps


module sr_latch_TB;


reg S, R;

wire Q, Q_bar;


sr_latch dut (

    .S(S),

    .R(R),

    .Q(Q),

    .Q_bar(Q_bar)
```

```
);

initial begin
    S = 1; R = 1; #10;
    S = 1; R = 0; #10;
    S = 1; R = 1; #10;
    S = 0; R = 1; #10;
    $finish;
end

endmodule
```

2. Results of the Behavioral simulation

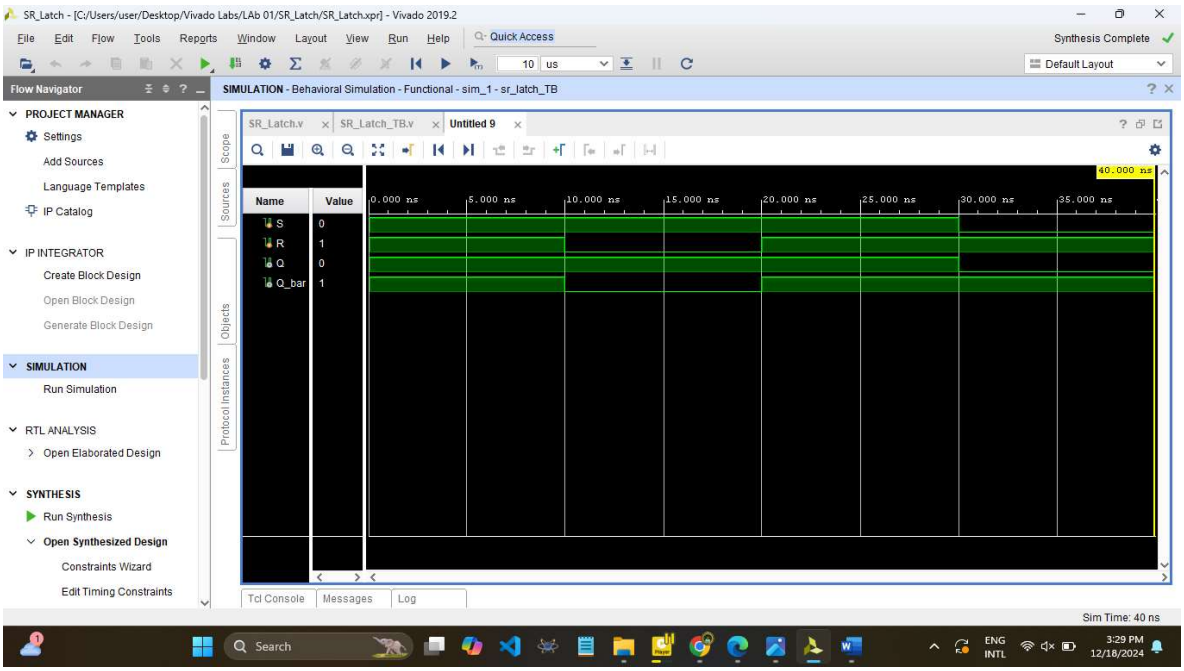

Figure 03:Waveforms of the SR flip flop behavioral simulation

Exercise 3: Implementation of 4 bit full adder

1. Module code of the 4 bit adder

Module code of the 1bit Full Adder. For the designing 4 bit full adder we have used 4 ,1 bit full adders.

```
`timescale 1ns / 1ps

module fa0(
    input A,
    input B,
    input Carry_in,
    output Sum,
    output Carry_out
);

// Internal wires
wire w1, w2, w3;

// Gate-level implementation
xor (w1, A, B);
xor (Sum, w1, Carry_in);
and (w2, A, B);
and (w3, w1, Carry_in);
or  (Carry_out, w2, w3);

endmodule
```

Module code of the **FullAdder_4bit**

```
`timescale 1ns / 1ps


module FullAdder_4bit (
    input [3:0] A,  // 4-bit input A
    input [3:0] B,  // 4-bit input B
    input Carry_in,        // Carry-in
    output [3:0] Sum, // 4-bit Sum output
    output Carry_out       // Final Carry-out
);
    wire c1, c2, c3;  // Internal carry wires
```

```
    // Instantiate 1-bit full adders
    fa0 fa0 (.A(A[0]), .B(B[0]), .Carry_in(Carry_in),    .Sum(Sum[0]), .Carry_out(c1));
    fa1 fa1 (.A(A[1]), .B(B[1]), .Carry_in(c1),     .Sum(Sum[1]), .Carry_out(c2));
    fa2 fa2 (.A(A[2]), .B(B[2]), .Carry_in(c2),     .Sum(Sum[2]), .Carry_out(c3));
    fa3 fa3 (.A(A[3]), .B(B[3]), .Carry_in(c3),     .Sum(Sum[3]), .Carry_out(Carry_out));

endmodule
```

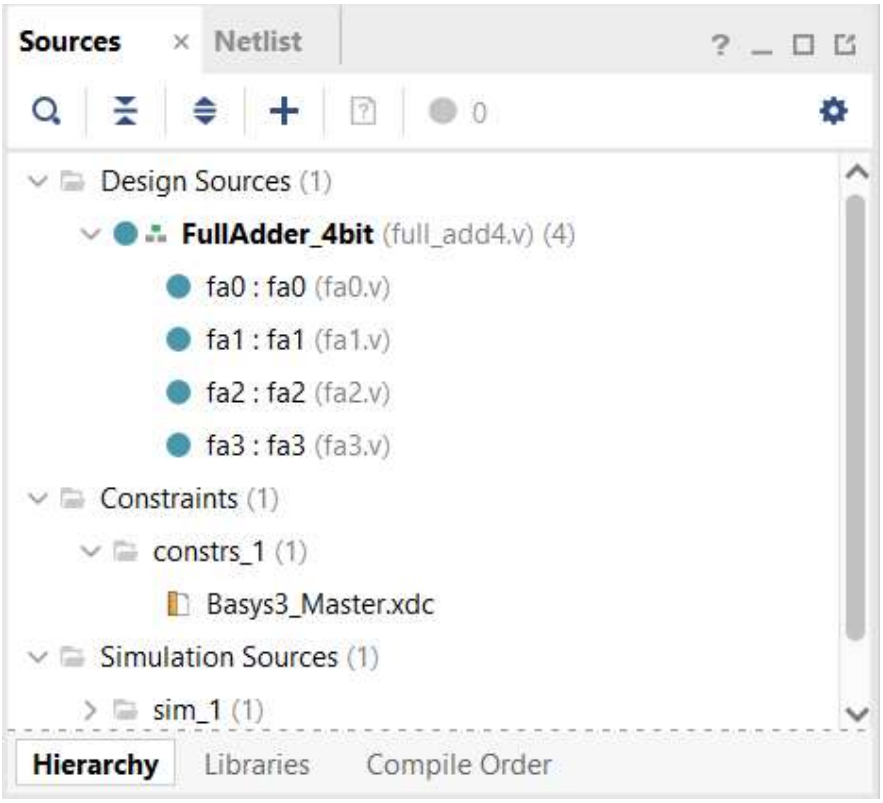

Figure 04 :Hierarchical View of 4bit full adder the design files