Due Date: Sunday September 1, 11:59pm

/ 38

EEDG/CE 6370
Design and Analysis or Reconfigurable Systems
Homework 1
Design of Combinational Logic Designs using Schematic entry and Verilog/VHDL using Intel Quartus Prime

Student Name: Prathamesh Sanjay Gadad
Student email: psg220003@utdallas.edu

Part I: Schematic Entry (1-bit adder)

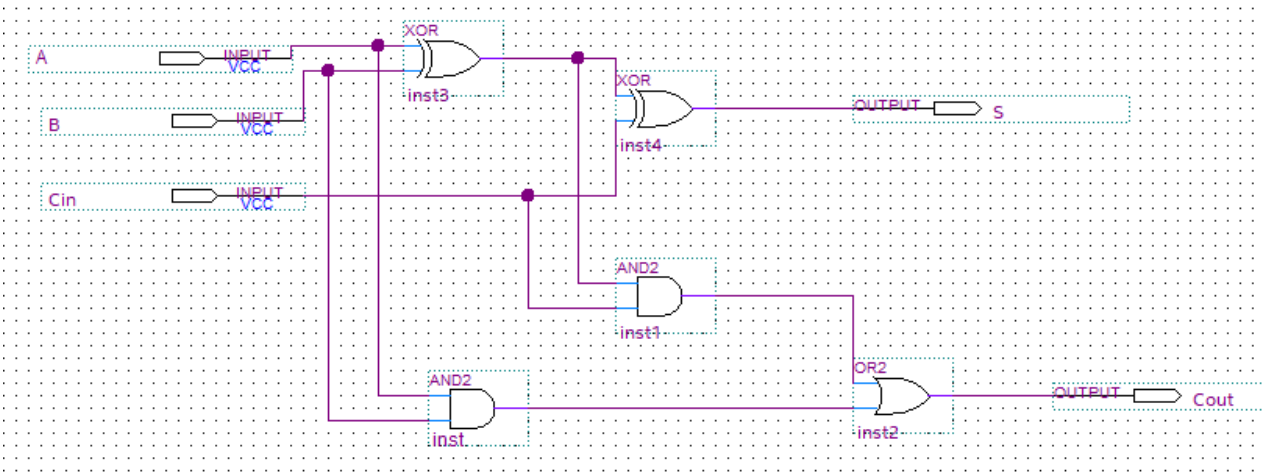
a.) Follow the instructions in the homework sheet and design a 1-bit full adder using schematic entry show here the generated schematic (screenshot of the circuit)

(4 marks)

Ans:

- I have designed the circuit with two '2 – input XOR' gates two '2 – input AND' gates and one '2 – input OR' gate.
- A, B and Cin are the Inputs and S, Cout are Output.
- All the gates are joined by wires.

Following is the circuit:



b.) Simulate the 1-bit full adder showing that it works. Include here the test vectors used and the simulation results (waveform). Clearly indicate why it is working.

(4 marks)

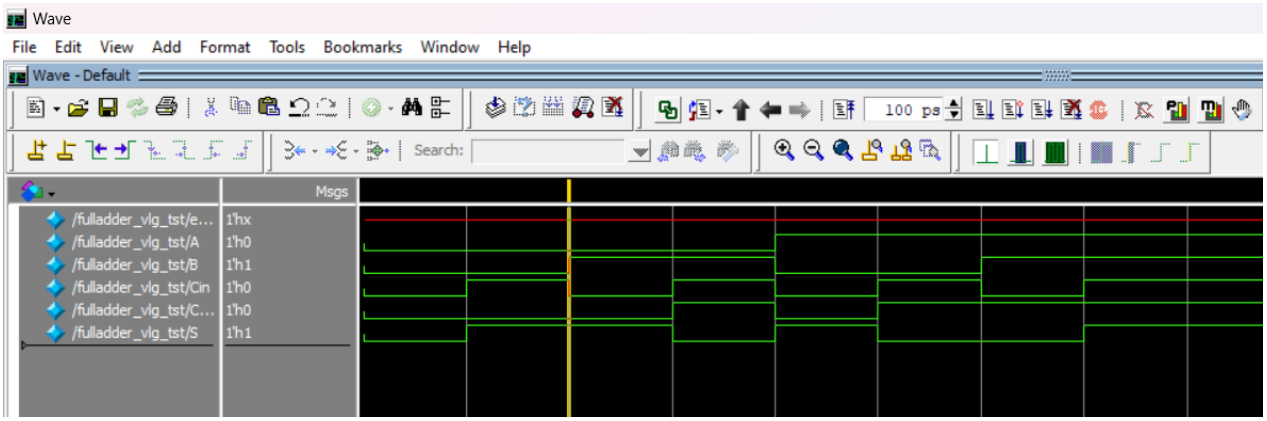
Ans:

We have created a Verilog file and then generated the testbench for the Verilog based on the schematic and verified the design, following is the testbench and generated waveform.

```
// Test Bench for 1-bit Full Adder
module tb_full_adder_4bit;

// Testbench signals
reg [3:0] A;
reg [3:0] B;
reg Cin;
wire [3:0] S;
wire Cout;

// Instantiate the full adder module (use behavioral or structural here)
FullAdder4bit UUT (
    .A(A),
    .B(B),
    .Cin(Cin),
    .S(S),
    .Cout(Cout)
);

// Initialize and apply test vectors
initial begin
    // Display results
    $monitor("A=%b B=%b Cin=%b | S=%b Cout=%b", A, B, Cin, S, Cout);

// Test vector 1
    A = 4'b0010; // Binary 2
    B = 4'b0101; // Binary 5
    Cin = 1'b0;  // Decimal 0
    #1;
```

```
// Test vector 2
A = 4'b0000; // Binary 0
B = 4'b0010; // Binary 2
Cin = 1'b1;  // Decimal 1
#1;

// Test vector 3
A = 4'b0000; // Binary 0
B = 4'b0011; // Binary 3
Cin = 1'b0;  // Decimal 0
#1;

// Test vector 4
A = 4'b0100; // Binary 4
B = 4'b0001; // Binary 1
Cin = 1'b1;  // Decimal 1
#1;

// Test vector 5
A = 4'b0010; // Binary 2
B = 4'b0000; // Binary 0
Cin = 1'b0;  // Decimal 0
#1;

// Test vector 6
A = 4'b0101; // Binary 5
B = 4'b0000; // Binary 0
Cin = 1'b1;  // Decimal 1
#1;

// Test vector 7
A = 4'b0101; // Binary 5
B = 4'b0001; // Binary 1
Cin = 1'b0;  // Decimal 0
#1;

// Test vector 8
A = 4'b0110; // Binary 6
```

```
    B = 4'b0001; // Binary 1
    Cin = 1'b1;  // Decimal 1
    #1;


    // End simulation
    $finish;
end

endmodule
```



Observation:

For the given inputs A=1'h0, B=1'h1, Cin=1'h0 the output obtained is S=1'h1, Cout=1'h0, thus we can say that 1-bit full adder schematic along with testbench is working correctly.

c.) What test vectors would you use in the case of having to test a 4-bit adder and an 8-bit adder? Do you foresee any problems verifying larger adders?

(2 marks)

Ans:

1. For 4-bit Adder we must use two 4bit as input, following are the test vectors:

- A = 4'b0000, B = 4'b0000, Cin = 0: Test adding zero.
- A = 4'b0110, B = 4'b0011, Cin = 1: Test with a carry-in affecting the result.
- A = 4'b1111, B = 4'b1111, Cin = 1: Maximum value addition with carry-in, expect S = 4'b1111, Cout = 1.
- A = 4'b1010, B = 4'b0101, Cin = 1: Test a mix of bits with carry-in.

2. For 8-bit Adder we must use two 8-bit as input, following are the test vectors:

- A = 8'b00000000, B = 8'b00000000, Cin = 0: Test adding zero.
- A = 8'b01100110, B = 8'b00110011, Cin = 1: Test with a carry-in affecting the result.
- A = 8'b11111111, B = 8'b11111111, Cin = 1: Maximum value addition with carry-in, expect Sum = 8'b11111111, Cout = 1.
- A = 8'b10101010, B = 8'b01010101, Cin = 1: Test a mix of bits with carry-in.

Problems verifying large adders:

- Timing and Propagation Delay
- Increased Complexity
- Simulation Timing

Apart from these there are many other issues, however these concern during design.

d.) Report the number of ALMs, ALUTs and critical path of the design from the synthesis report. Compute the maximum frequency.

(2 marks)

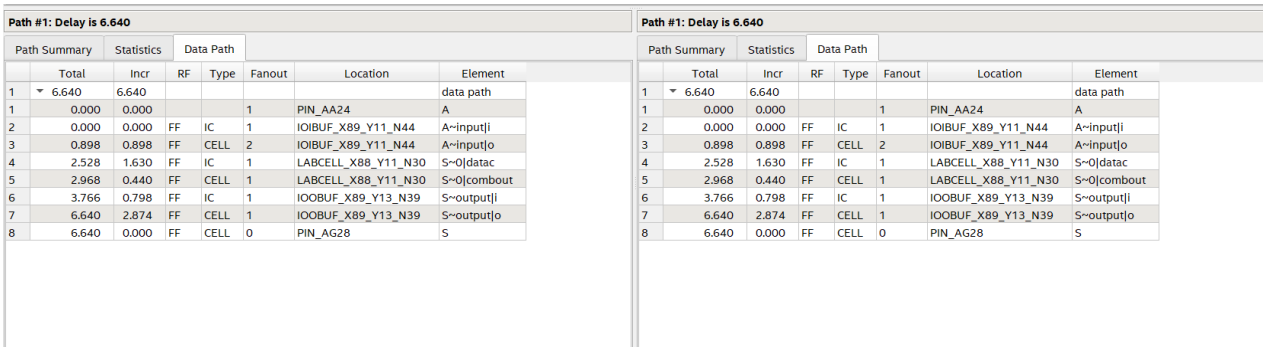
	ALMS/ALUTs	FFs	IOs	DSPs	BRAM
1-bit adder schematic	2	0	5	0	0

Critical Path:
The critical path is from A (Input) to S (Output; Sum)

Maximum Frequency:

= 1/6.640

= 150Mhz



Path #1: Delay is 6.640

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	6.640	6.640					data path
1	0.000	0.000			1	PIN_AA24	A
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y11_N44	A~input|i
3	0.898	0.898	FF	CELL	2	IOIBUF_X89_Y11_N44	A~input|o
4	2.528	1.630	FF	IC	1	LABCELL_X88_Y11_N30	S~0|datac
5	2.968	0.440	FF	CELL	1	LABCELL_X88_Y11_N30	S~0|combout
6	3.766	0.798	FF	IC	1	IOOBUF_X89_Y13_N39	S~output|i
7	6.640	2.874	FF	CELL	1	IOOBUF_X89_Y13_N39	S~output|o
8	6.640	0.000	FF	CELL	0	PIN_AG28	S

Path #1: Delay is 6.640

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	6.640	6.640					data path
1	0.000	0.000			1	PIN_AA24	A
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y11_N44	A~input|i
3	0.898	0.898	FF	CELL	2	IOIBUF_X89_Y11_N44	A~input|o
4	2.528	1.630	FF	IC	1	LABCELL_X88_Y11_N30	S~0|datac
5	2.968	0.440	FF	CELL	1	LABCELL_X88_Y11_N30	S~0|combout
6	3.766	0.798	FF	IC	1	IOOBUF_X89_Y13_N39	S~output|i
7	6.640	2.874	FF	CELL	1	IOOBUF_X89_Y13_N39	S~output|o
8	6.640	0.000	FF	CELL	0	PIN_AG28	S

Part II : RTL Entry (Verilog/VHDL) (1-bit adder)

a.) Re-do the full adder example specifying the 1-bit adder directly in Verilog or VHDL. Enter here the clean source code with comments.

(4 marks)

Ans:

- RTL code in Verilog:

```
//Design code for 1-bit adder
module fulladder_rtl(
        input A,         // First input bit
        input B,         // Second input bit
        input Cin,       // Carry-in bit
        output Cout,     // Carry-out bit
        output S         // Sum bit
    );

        // Sum Calculation
        // The sum S is calculated using the XOR operation.
        // It adds the input bits A, B, and Cin.
        assign S = A ^ B ^ Cin;

        // Carry-out Calculation
        // The carry-out Cout is calculated using the following logic:
        // A carry is generated if any two of the inputs are 1.
        assign Cout = (A & B) | (B & Cin) | (Cin & A);

    endmodule
```

- Test Bench:

```
//Test bench for 1-bit full adder
module fulladder_tb();
// constants
// general purpose registers
reg eachvec;
// test vector input registers
reg A;
reg B;
reg Cin;
// wires
wire Cout;
wire S;

// assign statements (if any)
fulladder_rtl i2 (
// port map - connection between master ports and signals/registers
   .A(A),
   .B(B),
   .Cin(Cin),
   .Cout(Cout),
   .S(S)
);
initial
begin
// code that executes only once
// insert code here --> begin

// --> end
$display("Running testbench");
end
always

// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// Test Case 0: A=0, B=0, Cin=0
```

```
A = 0; B = 0; Cin = 0;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 1: A=0, B=0, Cin=1
A = 0; B = 0; Cin = 1;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 2: A=0, B=1, Cin=0
A = 0; B = 1; Cin = 0;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 3: A=0, B=1, Cin=1
A = 0; B = 1; Cin = 1;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 4: A=1, B=0, Cin=0
A = 1; B = 0; Cin = 0;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 5: A=1, B=0, Cin=1
A = 1; B = 0; Cin = 1;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 6: A=1, B=1, Cin=0
A = 1; B = 1; Cin = 0;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);

// Test Case 7: A=1, B=1, Cin=1
A = 1; B = 1; Cin = 1;
#1; // 1 ns delay
$display("%b %b   %b  | %b   %b", A, B, Cin, S, Cout);
```

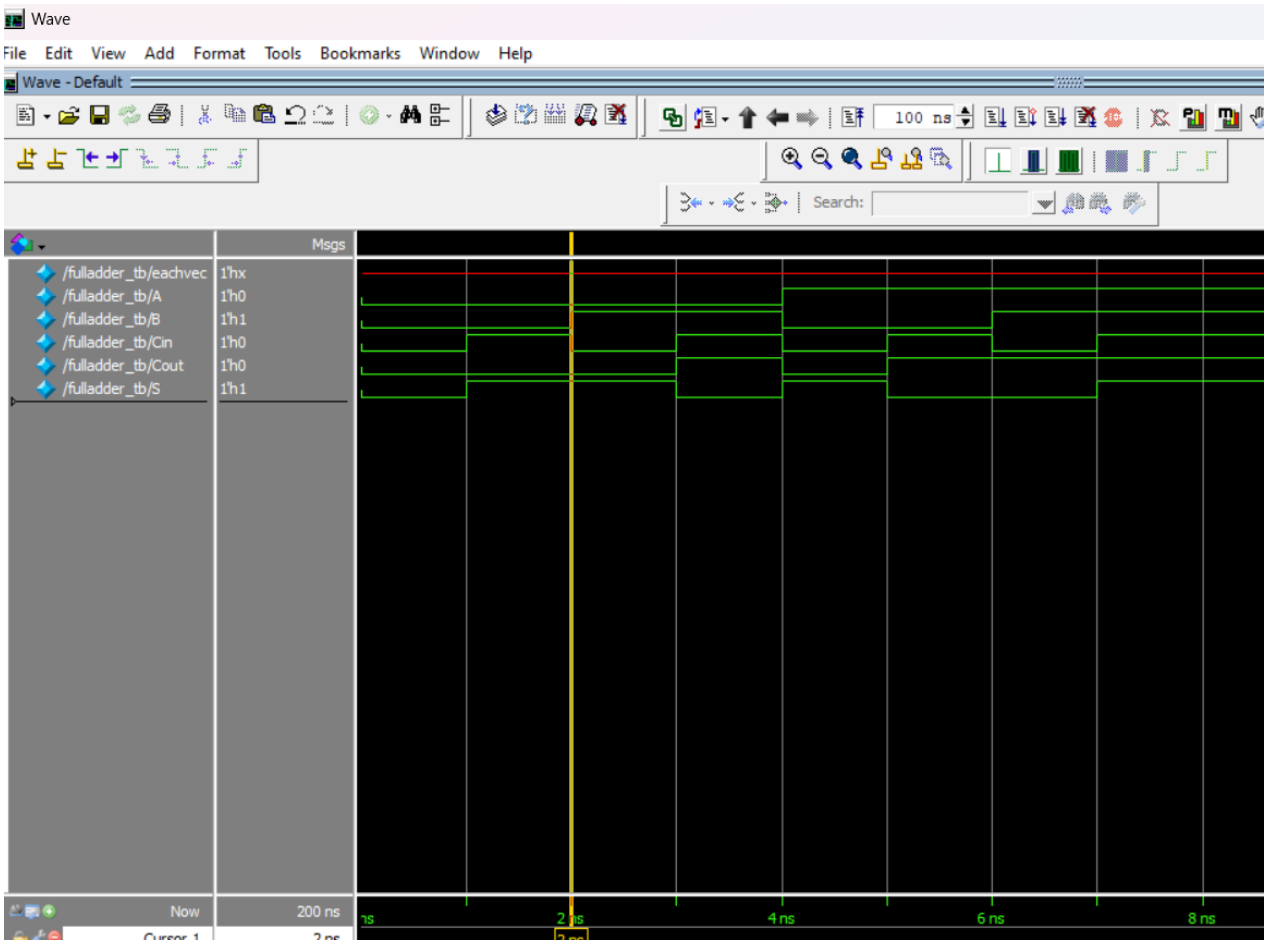
```
// code executes for every event on sensitivity list
// insert code here --> begin

@eachvec;
// --> end
end
endmodule
```

b.) Show the simulation results of the working adder.

(4 marks)

Ans:

Waveform:



Observation:

From the above waveform we can observe that for inputs, A=1'h0, B=1'h1, Cin=1'h0 the output obtained is S=1'h1, Cout=1'h0, thus we can say that 1-bit full adder RTL code along with testbench is working correctly.

c.) Report the number of ALMs, ALUTs and critical path of the design from the synthesis report. Compute the maximum frequency.

(2 marks)

	ALMs/ALUTs	FFs	IOs	DSPs	BRAM
1-bit adder RTL	2	0	5	0	0

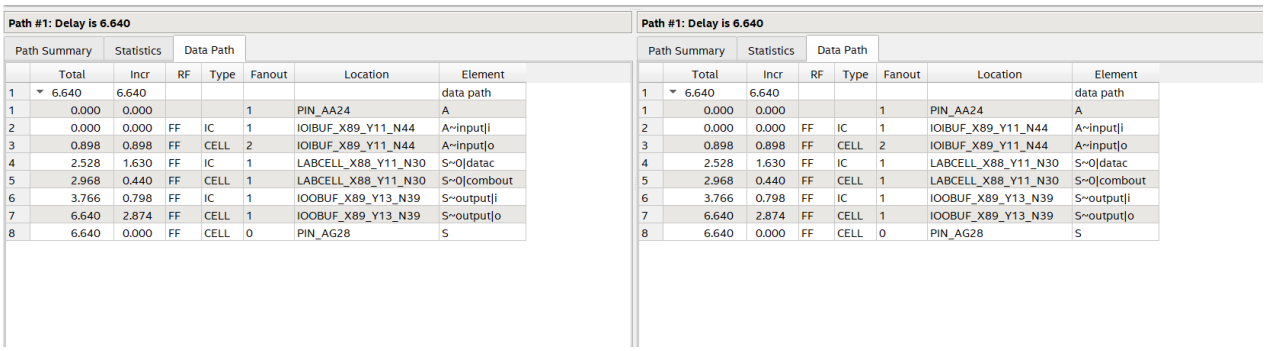
Critical Path:

From A (input) to S (output/ Sum)

Maximum Frequency:

= 1/6.640

= 150MHz



Path #1: Delay is 6.640

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	6.640	6.640					data path
1	0.000	0.000			1	PIN_AA24	A
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y11_N44	A~input|i
3	0.898	0.898	FF	CELL	2	IOIBUF_X89_Y11_N44	A~input|o
4	2.528	1.630	FF	IC	1	LABCELL_X88_Y11_N30	S~0|datac
5	2.968	0.440	FF	CELL	1	LABCELL_X88_Y11_N30	S~0|combout
6	3.766	0.798	FF	IC	1	IOOBUF_X89_Y13_N39	S~output|i
7	6.640	2.874	FF	CELL	1	IOOBUF_X89_Y13_N39	S~output|o
8	6.640	0.000	FF	CELL	0	PIN_AG28	S

Path #1: Delay is 6.640

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	6.640	6.640					data path
1	0.000	0.000			1	PIN_AA24	A
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y11_N44	A~input|i
3	0.898	0.898	FF	CELL	2	IOIBUF_X89_Y11_N44	A~input|o
4	2.528	1.630	FF	IC	1	LABCELL_X88_Y11_N30	S~0|datac
5	2.968	0.440	FF	CELL	1	LABCELL_X88_Y11_N30	S~0|combout
6	3.766	0.798	FF	IC	1	IOOBUF_X89_Y13_N39	S~output|i
7	6.640	2.874	FF	CELL	1	IOOBUF_X89_Y13_N39	S~output|o
8	6.640	0.000	FF	CELL	0	PIN_AG28	S

d.) Compare the ALUT usage and critical path delay from the schematic adder and RTL adder do the results match? Yes/no? Explain why?

(2 marks)

	ALMs/ALUTs	FFs	IOs	DSPs	BRAM	Delay
Schematic	2	0	5	0	0	6.640
RTL	2	0	5	0	0	6.640

Observation:

From the above two tasks we can observe that the ALUTs/ ALMs, critical path and delay are same for both schematic and RTL code, which implies that RTL is the higher abstraction of the digital circuits and we can completely be dependent on RTL code without investing time in designing schematics to observe the functionality and performance of lower-level schematic circuits.

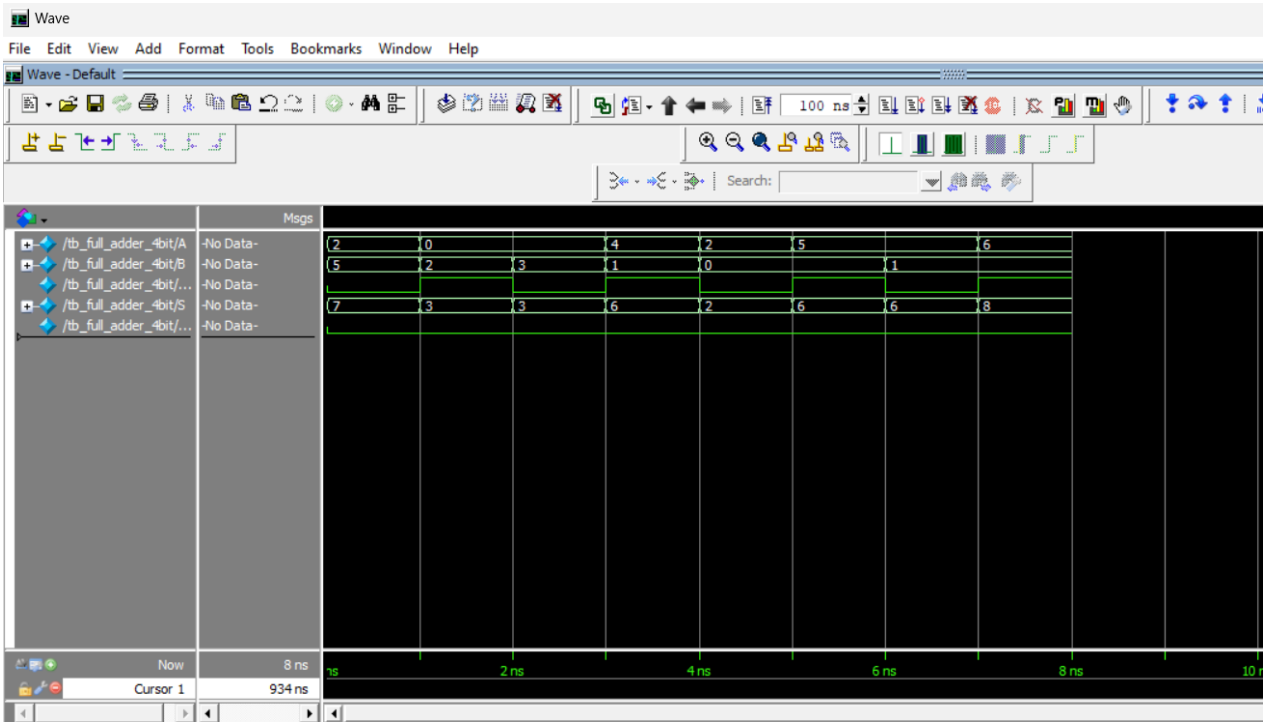
Part III : RTL Entry (Verilog/VHDL) (4-bit adder)

a.) Create a 4-bit adder using structural and behavioral RTL (Verilog or VHDL). Annotate their resource usage and critical path in the table below:"

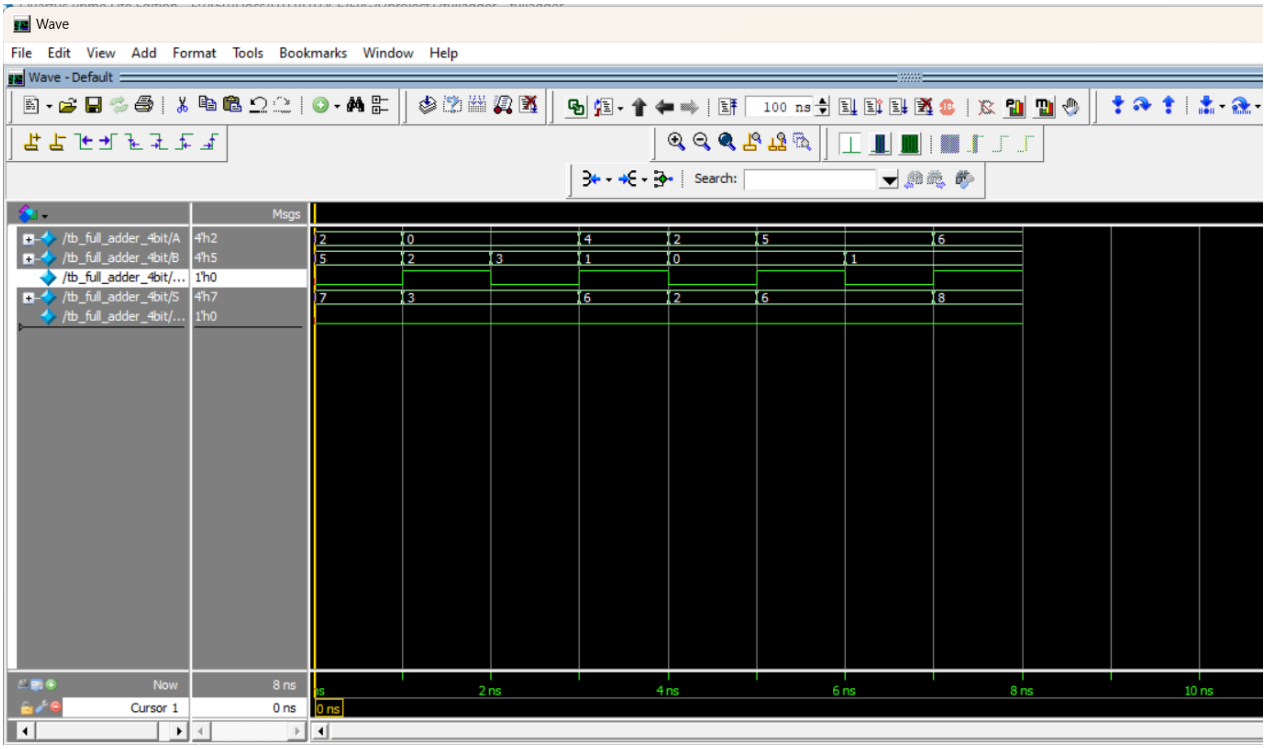
(4 marks)

Ans:

For Structural



For Behavioral



	ALMs/ALUTs	Critical path
Structural	4/6	B[1] – Cout
Behavioral	3/5	A[0] – S[2]

For Structural

Path #1: Delay is 7.868

Path Summary | Statistics | Data Path

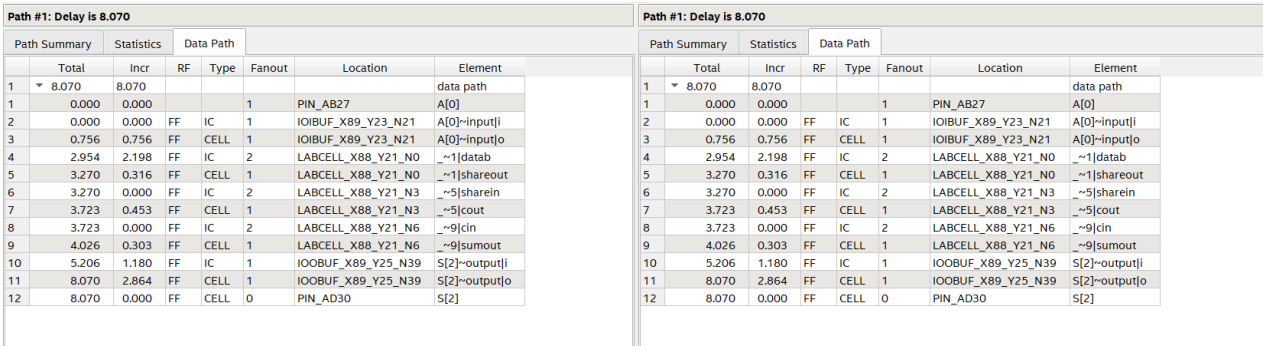
	Total	Incr	RF	Type	Fanout	Location	Element
1	7.868	7.868					data path
1	0.000	0.000			1	PIN_AB27	B[1]
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y23_N21	B[1]~input|i
3	0.756	0.756	FF	CELL	2	IOIBUF_X89_Y23_N21	B[1]~input|o
4	2.989	2.233	FF	IC	1	LABCELL_X88_Y21_N42	FA1|Cout|datab
5	3.467	0.478	FF	CELL	3	LABCELL_X88_Y21_N42	FA1|Cout|combout
6	3.660	0.193	FF	IC	1	LABCELL_X88_Y21_N33	FA3|Cout|datac
7	4.094	0.434	FF	CELL	1	LABCELL_X88_Y21_N33	FA3|Cout|combout
8	5.004	0.910	FF	IC	1	IOOBUF_X89_Y25_N39	Cout~output|i
9	7.868	2.864	FF	CELL	1	IOOBUF_X89_Y25_N39	Cout~output|o
10	7.868	0.000	FF	CELL	0	PIN_AD30	Cout

Path #1: Delay is 7.868

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	7.868	7.868					data path
1	0.000	0.000			1	PIN_AB27	B[1]
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y23_N21	B[1]~input|i
3	0.756	0.756	FF	CELL	2	IOIBUF_X89_Y23_N21	B[1]~input|o
4	2.989	2.233	FF	IC	1	LABCELL_X88_Y21_N42	FA1|Cout|datab
5	3.467	0.478	FF	CELL	3	LABCELL_X88_Y21_N42	FA1|Cout|combout
6	3.660	0.193	FF	IC	1	LABCELL_X88_Y21_N33	FA3|Cout|datac
7	4.094	0.434	FF	CELL	1	LABCELL_X88_Y21_N33	FA3|Cout|combout
8	5.004	0.910	FF	IC	1	IOOBUF_X89_Y25_N39	Cout~output|i
9	7.868	2.864	FF	CELL	1	IOOBUF_X89_Y25_N39	Cout~output|o
10	7.868	0.000	FF	CELL	0	PIN_AD30	Cout

For Behavioral



Path #1: Delay is 8.070

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	8.070	8.070					data path
1	0.000	0.000			1	PIN_AB27	A[0]
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y23_N21	A[0]~input|i
3	0.756	0.756	FF	CELL	1	IOIBUF_X89_Y23_N21	A[0]~input|o
4	2.954	2.198	FF	IC	2	LABCELL_X88_Y21_N0	_~1|datab
5	3.270	0.316	FF	CELL	1	LABCELL_X88_Y21_N0	_~1|shareout
6	3.270	0.000	FF	IC	2	LABCELL_X88_Y21_N3	_~5|sharein
7	3.723	0.453	FF	CELL	1	LABCELL_X88_Y21_N3	_~5|cout
8	3.723	0.000	FF	IC	2	LABCELL_X88_Y21_N6	_~9|cin
9	4.026	0.303	FF	CELL	1	LABCELL_X88_Y21_N6	_~9|sumout
10	5.206	1.180	FF	IC	1	IOOBUF_X89_Y25_N39	S[2]~output|i
11	8.070	2.864	FF	CELL	1	IOOBUF_X89_Y25_N39	S[2]~output|o
12	8.070	0.000	FF	CELL	0	PIN_AD30	S[2]

Path #1: Delay is 8.070

Path Summary | Statistics | Data Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	8.070	8.070					data path
1	0.000	0.000			1	PIN_AB27	A[0]
2	0.000	0.000	FF	IC	1	IOIBUF_X89_Y23_N21	A[0]~input|i
3	0.756	0.756	FF	CELL	1	IOIBUF_X89_Y23_N21	A[0]~input|o
4	2.954	2.198	FF	IC	2	LABCELL_X88_Y21_N0	_~1|datab
5	3.270	0.316	FF	CELL	1	LABCELL_X88_Y21_N0	_~1|shareout
6	3.270	0.000	FF	IC	2	LABCELL_X88_Y21_N3	_~5|sharein
7	3.723	0.453	FF	CELL	1	LABCELL_X88_Y21_N3	_~5|cout
8	3.723	0.000	FF	IC	2	LABCELL_X88_Y21_N6	_~9|cin
9	4.026	0.303	FF	CELL	1	LABCELL_X88_Y21_N6	_~9|sumout
10	5.206	1.180	FF	IC	1	IOOBUF_X89_Y25_N39	S[2]~output|i
11	8.070	2.864	FF	CELL	1	IOOBUF_X89_Y25_N39	S[2]~output|o
12	8.070	0.000	FF	CELL	0	PIN_AD30	S[2]

b.) Discuss if the results obtained are what you expected and why yes or no.

(2 marks)

Ans:

Considering that 4-bit adder is more complex than 1-bit adder and assuming that resource utilization is more important, than we can consider Behavioral RTL over Structural RTL, however if we consider delay, then it is high for Behavioral RTL due to the usage of more optimized constructs.

So, there is a tradeoff between delay and resource utilization.

Part IV: Homework reflection

(4 marks)

What do you think were the objectives of this homework?

Ans:

The objective of this homework is get familiarized with Quartus Prime Lite and Questa simulator, apart from this to learn the design structure and how to work optimized, as we worked on the schematic and then directly using RTL coding, observed the difference, later got to analyze the resource (ALMs, ALUTs) utilizations, delay, critical path and tried to analyze which one to select for better design.

What conclusions have you extracted from the results obtained?

Ans:

Based on the above tasks, we can conclude that, using of RTL abstraction provides the same result as schematic does, thus we can start from RTL level for further tasks and save time. And usage of behavioral model would be better considering the tradeoff as it enables us to use functionality of the digital circuit.

Part V: Create a short YouTube video that shows that the designs work and the results obtained. Include the link to the video here.

(4 marks)

https://youtu.be/h_7rpPd9S6w

Attachments

Include the Verilog/VHDL code here. Make sure it is well formatted and include comments. Marks might get deducted if not.

For Behavioral:

```
// 1-bit Full Adder in Behavioral Style
module fulladder_behavioural_4bit (
    input [3:0] A,
    input [3:0] B,
    input Cin,
    output wire [3:0] S,
    output wire Cout
);

// Sum calculation
assign {Cout, S} = A + B + Cin;

endmodule
```

For Structural:

```
// 1-bit Full Adder in Structural Style
module FullAdder (
    input A, B, Cin,
    output S, Cout
);
```

```
    wire S1, C1, C2;

    xor (S1, A, B);
    xor (S, S1, Cin);
    and (C1, A, B);
    and (C2, S1, Cin);
    or (Cout, C1, C2);
endmodule

module FullAdder4bit (
    input [3:0] A, B,
    input Cin,
    output [3:0] S,
    output Cout
);
    wire C1, C2, C3;

    FullAdder FA0 (.A(A[0]), .B(B[0]), .Cin(Cin), .S(S[0]), .Cout(C1));
    FullAdder FA1 (.A(A[1]), .B(B[1]), .Cin(C1), .S(S[1]), .Cout(C2));
    FullAdder FA2 (.A(A[2]), .B(B[2]), .Cin(C2), .S(S[2]), .Cout(C3));
    FullAdder FA3 (.A(A[3]), .B(B[3]), .Cin(C3), .S(S[3]), .Cout(Cout));
Endmodule
```

Test Bench:

```
// Test Bench for 1-bit Full Adder
module tb_full_adder_4bit;

// Testbench signals
reg [3:0] A;
reg [3:0] B;
reg Cin;
wire [3:0] S;
wire Cout;

// Instantiate the full adder module (use behavioral or structural here)
FullAdder4bit UUT (
```

```
    .A(A),
    .B(B),
    .Cin(Cin),
    .S(S),
    .Cout(Cout)
);

// Initialize and apply test vectors
initial begin
    // Display results
    $monitor("A=%b B=%b Cin=%b | S=%b Cout=%b", A, B, Cin, S, Cout);

// Test vector 1
    A = 4'b0010; // Binary 2
    B = 4'b0101; // Binary 5
    Cin = 1'b0;  // Decimal 0
    #1;

    // Test vector 2
    A = 4'b0000; // Binary 0
    B = 4'b0010; // Binary 2
    Cin = 1'b1;  // Decimal 1
    #1;

    // Test vector 3
    A = 4'b0000; // Binary 0
    B = 4'b0011; // Binary 3
    Cin = 1'b0;  // Decimal 0
    #1;

    // Test vector 4
    A = 4'b0100; // Binary 4
    B = 4'b0001; // Binary 1
    Cin = 1'b1;  // Decimal 1
    #1;

    // Test vector 5
    A = 4'b0010; // Binary 2
```

```
    B = 4'b0000; // Binary 0
    Cin = 1'b0;  // Decimal 0
    #1;

    // Test vector 6
    A = 4'b0101; // Binary 5
    B = 4'b0000; // Binary 0
    Cin = 1'b1;  // Decimal 1
    #1;

    // Test vector 7
    A = 4'b0101; // Binary 5
    B = 4'b0001; // Binary 1
    Cin = 1'b0;  // Decimal 0
    #1;

    // Test vector 8
    A = 4'b0110; // Binary 6
    B = 4'b0001; // Binary 1
    Cin = 1'b1;  // Decimal 1
    #1;


    // End simulation
    $finish;
end

endmodule
```