EEL2020 Digital Design Lab Report

Sem II AY 2023-24

Experiment No.	:	**02**
Name	:	**Lavangi Parihar**
Roll No.	:	**B22EE044**
Partner Name (Partner Roll No.)	:	**Mansi Choudhary (B22EE045)**

Objective

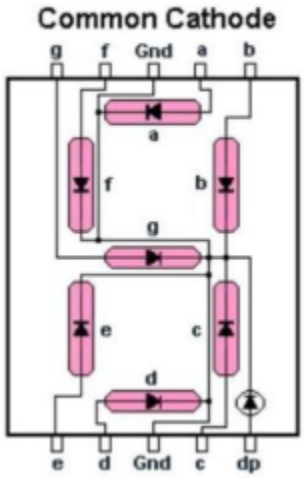
a. Implement a BCD-to-7 segment decoder on the PYNQ Z2 FPGA board using Verilog through the Vivado Design Suite.
b. Write a full adder module and use it to simulate a 4-bit adder.

A)

Logic Design

A BCD to 7-segment display decoder is a special decoder which can convert a 4-bit binary-coded decimal (BCD) input, say ABCD, into a form that displays the corresponding decimal number on a seven-segment display (SSD).

CIRCUIT DIAGRAM:



TRUTH TABLE:

Binary Inputs				Decoder Outputs							7-Segment Display Outputs
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	**0**	1	1	1	1	1	1	0	0
0	0	0	**1**	0	1	1	0	0	0	0	**1**
0	0	**1**	0	1	1	0	1	1	0	1	**2**
0	0	1	1	1	1	1	1	0	0	1	3
0	**1**	0	0	0	1	1	0	0	1	1	**4**
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	**8**
1	0	0	1	1	1	1	1	0	1	1	9

VERILOG REPRESENTATION:-

```
module sevensegdisplay(input A3, A2, A1, A0, output a, b, c, d, e, f, g);
        assign a = ~(A3 | A1 | (A2 & A0) | (~A2 & ~A0));
        assign b = ~(~A2 | (~A1 & ~A0) | (A1 & A0));
        assign c = ~(A2 | ~A1 | A0);
        assign d = ~(A3 | (~A2 & A1) | (~A2 & ~A0) | (A1&~A0) | (A2 & ~A1 & A0));
        assign e = ~((A1&~A0) | (~A2 & ~A0));
        assign f = ~(A3 | (~A1&~A0) | (A2 & ~A1) | (A2 & ~A0));
        assign g = ~(A3 | (A2&~A1) | (~A2 & A1) | (A1 & ~A0));
endmodule
```

Source Description

Design Source:- Verilog(or VHDL) files containing the logic description of the implemented circuit.
Constraint file (XDC):- Specifies pin assignments, I/O standards, and timing constraints for the FPGA

- Design source

Module name: BCD.v
Input ports: A0,A1,A2,A3
Output ports: a,b,c,d,e,f,g

- Constraint file

The XDC file was updated with the following changes:

Ports (from Verilog module)	**Designation** (Input/Output)	PYNQ Component Type (Button/LED/Switch etc. along with number, eg. LD01, BTN2, etc.)	Pin Configuration (from the PYNQ User Manual)
A0	input	BTN1	D19
A1	input	BTN2	D20
A2	input	BTN3	L20
A3	input	BTN4	L19
a	output	PmodB1	W14
b	output	PmodB2	Y14
c	output	PmodB3	T11
d	output	PmodB4	T10
e	output	PmodB5	V16
f	output	PmodB6	W16
g	output	PmodB7	V12

Elaborated Design

(Insert the Logic diagram created on Vivado)

The design in Vivado matches the planned half-adder circuit.it correctly uses the XOR gate for sum(S) and the AND gate for carry(C).the connections follows the Verilog code,ensuring the circuit operates as expected .this verification aligns with the truth table,confirming the accuracy of the design.

PYNQ Working Video (to be recorded during lab session)

(with voiceover briefly explaining the working, not exceeding 1-2 minutes)
Video link : link

List of Attachments

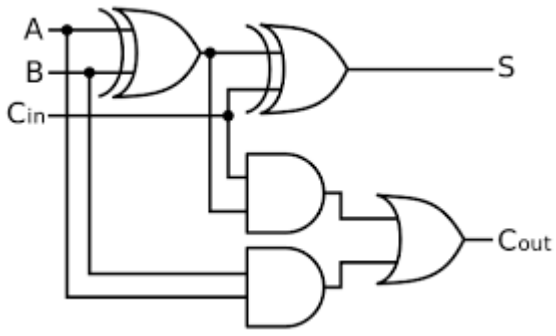
Video link: link
Link for the whole project: link

B)

Logic Design

A full adder adds three one-bit inputs. Given the positional nature of addition of n-bit binary numbers, full adders can be connected in cascade to generate sum of individual bit positions and pass on the carry output.

CIRCUIT DIAGRAM



:

LOGIC EXPRESSIONS:-
1) Sum(S):- The sum is obtained from the XOR operation of A and B
2) Carry(C):- The carry is obtained from the AND operation of A and B C=A.B

VERILOG REPRESENTATION:-

```
module full_adder(input X, input Y, input Cin, output Sum, output Cout);
assign {Cout, Sum} = X + Y + Cin;
endmodule


module four_bit_adder(input [3:0] A, input [3:0] B, output
[3:0] S, output Co);
wire [3:0] C;
full_adder fa0(A[0], B[0], 0, S[0], C[0]);
full_adder fa1(A[1], B[1], C[0], S[1], C[1]);
full_adder fa2(A[2], B[2], C[1], S[2], C[2]);
full_adder fa3(A[3], B[3], C[2], S[3], Co);
endmodule
```

Source Description

Design Source:- Verilog(or VHDL) files containing the logic description of the implemented circuit.
Constraint file (XDC):- Specifies pin assignments, I/O standards, and timing constraints for the FPGA

- Design source

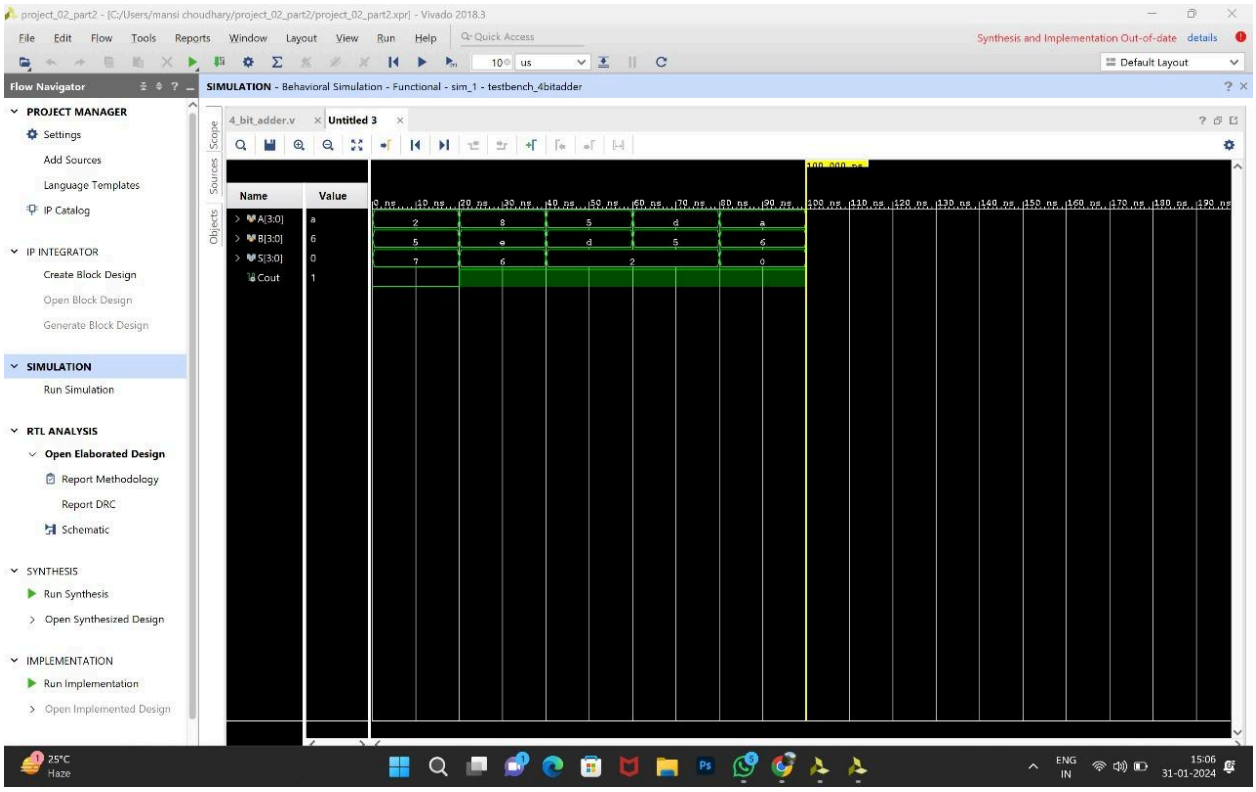
Module name: bit_adder.v
Input ports: [3:0]A,[3:0]B,
Output ports: [3:0]S,Co

- Testbench

```
//Testbench
module testbench_4bitadder;
reg [3:0] A, B;
wire [3:0] S;
wire Cout;

four_bit_adder B4add(A, B, S, Cout);
// Test values
initial begin

A= 4'b0010; B=4'b0101; #20;
A= 4'b1000; B= 4'b1110; #20;
A=4'b0101; B=4'b1101; #20;
A= 4'b1010; B=4'b0110; #20;

// Stop simulation
$finish;
end

endmodule
```

- Simulation source

Simulation Results (Timing diagram)



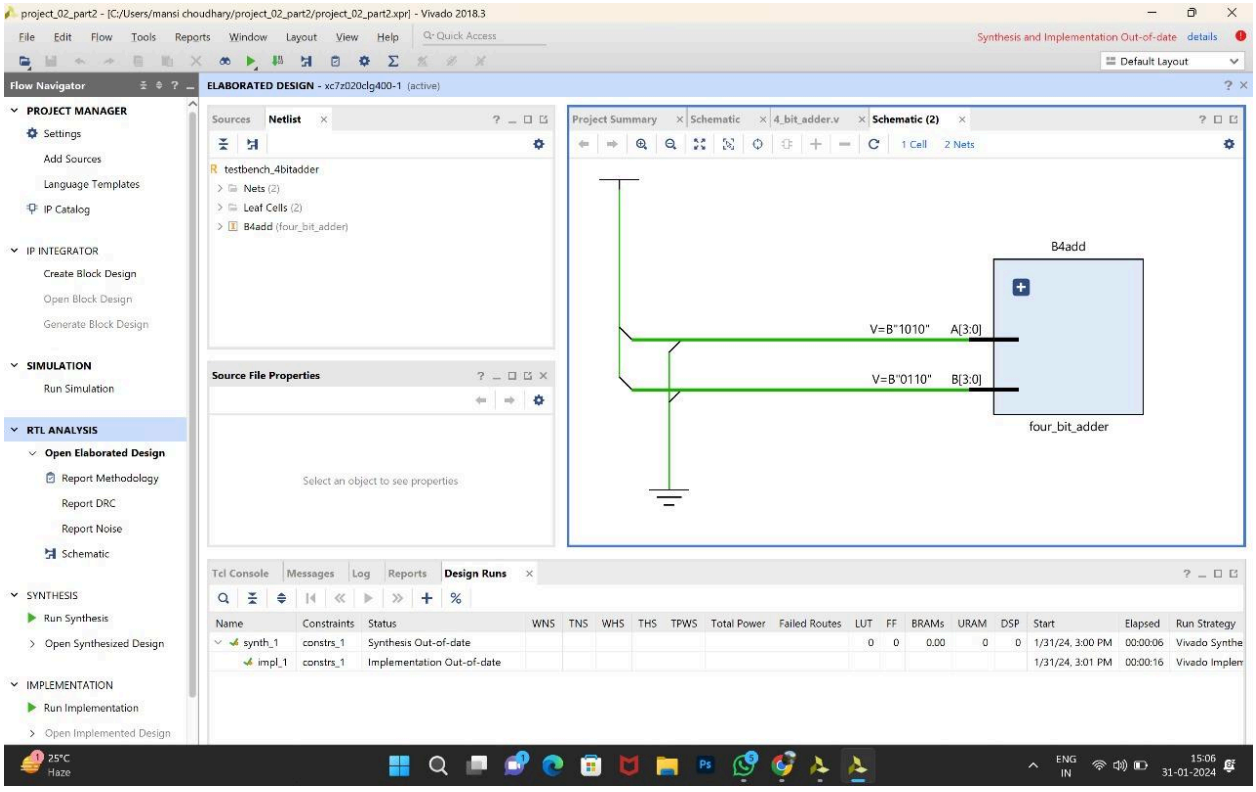
Insert a table verifying the truth table with that observed in the timing diagram.

A	*B*	*S*	*C*	*A*	*B*	*S*	*C*
0010	*0101*	*1000*	*0*	*0011*	*0101*	*1000*	*0*
1000	*0001*	*1001*	*1*	*1000*	*0001*	*1001*	*1*
0101	*1101*	*0010*	*1*	*0101*	*1101*	*0010*	*1*
1010	*0110*	*1000*	*1*	*1010*	*0110*	*1000*	*1*

Hence I have verified that the expected addition is the same as the 4-bit addition found here.

Elaborated Design

(Insert the Logic diagram created on Vivado)



The design in Vivado matches the planned design of 4-bit adder according to the Verilog code,ensuring the circuit operates as expected, this verification aligns with the truth table,confirming the accuracy of the design.

List of Attachments

Link for the whole project: link