Dept. of CSE, Bennett University
ECSE108L – Digital Design
Lab Assignment – 06

Q 1. A half-adder is used to add two single bit inputs. It produces a single bit output and a possible carry bit. Below is the truth table for the same.

Input		**Output**	
A	B	Carry	Output
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

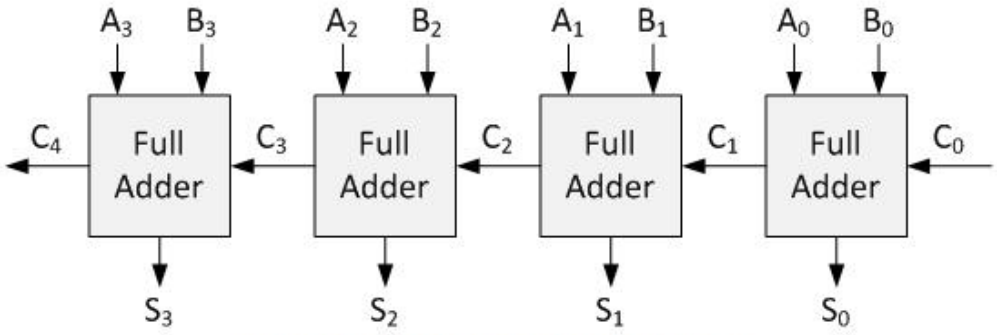
(i) Develop the Boolean expression and logic circuit for the given truth table.
(ii) Write a Verilog module for Universal NOR gate. Utilize the **instances of NOR gate** only to write a structural Verilog code for Half adder.
(iii) Verify it with respective testbench code.

Q 2. The half adder in the previous question can only add two one bit numbers when there is no carry bit, which is not sufficient in many cases. While a full-adder has two one-bit inputs, a carry-in input, a sum output, and a carry-out output. Below truth table represents a full-adder.

Full Adder Truth Table				
Input			**Output**	
A	B	Cin	Cout	Sum
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

(i) Utilize the above truth table to design the Boolean expression and digital circuit for the full adder.
(ii) Write the behavioral Verilog code for the full adder.
(iii) Verify it with respective Testbench code.

Q 3. The full adder designed in previous question is able to add only two one bit inputs. We can advance our design by adding N number of full adders to create a N bit Ripple carry adder. A block diagram for 4-bit Ripple carry adder can be expressed as:



Ripple Carry Adder (4-bit) Block Diagram

(i) By using the instances of full adder module created in Question 2, now create a 4-bit Ripple carry adder Verilog module.
(ii) Write a Testbench code to verify it. (you may take 1101 and 1011 as input values)

Q 4. We can also utilize a adder module to perform the subtraction of binary numbers with the help of 2's complement. Use the Ripple adder module developed in previous question to develop a binary subtraction module.

Verify your code by a suitable Testbench code and test it for following inputs.
(a) 1011 - 1001
(b) 1010 - 1110

Note:
The syntax for multi-bit port declaration is as follows:
port_direction data_type [port_size] port_name

Example:
input [1:0] a_in // a two bit [1, 0] names a_in
output reg [2:0] b_in // a three bit register type output port named as b_in