Assignment-1

1. Write a Verilog code module to implement one-bit full adder. Write another module to implement eight-bit ripple carry adder that instantiates eight one-bit full adders and connects them properly. Now, add a test bench to test the eight-bit ripple carry adder.
Write a test bench to test the eight-bit adder. Make sure to display your inputs, sum, and carry out. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs. Place one module in one Verilog file i.e., you will have three Verilog files.

2. Write a Verilog code to implement an eight-bit comparator using eight one-bit comparators. Follow the similar nomenclature as discussed in question 1. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs. Place one module in one Verilog file i.e., you will have three Verilog files.

3. Write a Verilog code to implement a two-to-four decoder. Then a top module for three-to-eight decoder using two-to-four decoder. Write a test bench to test three-to eight decoder for all possible input with five-time delay.

4. write a Verilog code to implement an 8-to-3 priority encoder where priority is given to the least significant input position with a 1. For example, if the input is 11000000, the output would be 110 encoding the position of the least significant 1 in the input (leftmost bit is the most significant bit). Write a test bench to test encoder for at least five test cases. Put five-time unit delay between consecutive inputs.

Assignment-2

1. Write a detailed description of 8 bit Carry Look-Ahead Adder and its working with the proper circuit diagram in a PDF file. Then write the Verilog code module to implement Carry Look-Ahead Adder. Now, add a test bench to test the Carry Look-Ahead Adder. Make sure to display your inputs, sum, and carry out. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs.

2. Write a detailed description of 8-bit Johnson Counter and its working with the proper circuit diagram and truth table in a PDF file. Then write the Verilog code module to implement 8-bit Johnson Counter. Now, add a test bench to test all the states of the 8-bit Johnson Counter.

Assignment-3

1. Construct a finite state machine for a sequence detector that detects the sequence 1010. The FSM should permit overlapping too. For example if the input sequence is 01101010, the corresponding output sequence is 00000101.
a) Construct the state diagram, Transition and output table, excitation table.
b) Then build the K-map for the same and design the circuitry or logic diagram.
c) Write the Verilog code to implement the sequence detector.
d) Add a test bench to test the sequence detector given fifteen different inputs. Put five-time unit delay between consecutive inputs.

2. Construct a finite state machine for the 3-bit odd parity bit generator. A serial parity-bit generator is a two-terminal circuit that receives coded messages and adds a parity bit to every m bits of the message so that the resulting outcome is an error detecting code. The inputs are assumed to arrive in strings of three symbols(m=3) and the string is spaced apart by single time units (i.e., the fourth place is blank). The parity bits are inserted in the appropriate spaces so that the resulting outcome is a continuous string of symbols without spaces. For even parity, a parity bit 1 is inserted, if and only if the numbers of 1s in the preceding string of three symbols are odd. For odd parity, a parity bit 1 is inserted, if and only if the numbers of 1s in the preceding string of three symbols is even. Here we will focusing on designing only odd parity generator.
a) Construct the state diagram, state table, transition and output table, excitation table.
b) Then build the K-map for the same and design the circuitry or logic diagram.
c) Then write the Verilog code module to implement the 3-bit odd parity bit generator.
d) Now, add a test bench to test the 3-bit odd parity bit generator given 8 different inputs. Put five-time unit delay between consecutive inputs.
The students have to submit a pdf file containing the details of the FSM constructions and the verilog code.

Assignment-4

1. Construct a finite state machine for the 3-bit Gray code counter. The counter is to be designed with one input terminal (which receives pulse signals) and one output terminal. It should be capable of counting in the Gray system up to 7 and producing an output pulse for every 8 input pulses. After the count 7 is reached, the next pulse will reset the counter to its initial state, i.e., to a count of zero.
The state transitions will be: S0: 000, S1:001, S2:011, S3:010, S4:110, S5:111, S6:101, S7:100 --> S0
Output will be high only from S7-->S0.
a) Construct the state assignment, the state diagram, state table, transition and output table, excitation table.
b) Then build the K-map for the same and design the circuitry or logic diagram.
c) Write the Verilog code module to implement the 3-bit Gray code counter.
d) Now, add a test bench to test the 3-bit Gray code counter.

2. Write a detailed description of eight-bit adder/subtracter to add/subtract two eight-bit two's complement numbers. and it's working with the proper circuit diagram in a PDF file. Then write the Verilog code module to implement an eight-bit adder/subtracter. It will be implemented in two modules. First, module implements a one-bit adder/subtracter with four inputs a, b, cin, and opcode, and two outputs sum and carry. For the addition operation the input opcode will be 0 and 1 for subtraction operation. The top module implements the eight-bit adder/subtracter using the one-bit adder/subtracter module. There will be two inputs for this module the two input numbers and the opcode and produces the sum and whether there is an overflow as the outputs. Now, add a test bench to test the eight-bit adder/subtracter. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs.

Assignment-5

1. Write a Verilog code to design a hardware for calculating the GCD of two 8-bit numbers.
Next, write a test bench to test the design. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs.

2. Write an assembly language program for insertion sort of 10 integers given input by the user in the console. Display the final result in the console.

3. Write an assembly language program for printing fibonacci numbers upto 500. Display the series in the console.

4. Take input a number p and a single-precision floating point vector B which has p elements, b0,b1,⋯,bp
Calculate the following
$q=(\text{sum from } i = 0 \text{ to } p\text{-}1)(-1)^i b_i$

Display the final result in the console.