Verilog Lab-1

1. Write the code for single bit full adder using
2. Data flow modeling
3. Extend the above approach to create a 4-bit full adder.
4. Extend the above approach to create a 8-bit full adder.
5. Gate level modeling
6. Create 4-bit full adder using single bit full adders (designed at gate level).
7. Create 8-bit full adder using 4-bit full adders (designed above).
8. Write code for 2:1 multiplexer using
9. Gate level modeling
10. Create 4:1 mux using 2:1 designed above.
11. Ternary Operator
12. Extend the concept to build 4:1 mux using Ternary operation only.
13. Design D Latch using logic gate. (Hint: design D Latch from SR Latch).
14. NAND Gates only.
15. NOR Gates only.
16. Design T Flip Flop using Latches having asynchronous set and clear.
17. Design frequency divide by 8 circuit using T Flip Flop.
18. Design and verify following circuits using Verilog operators:
19. NAND Gate.
20. XOR Gate
21. 4:1 Multiplexer.
22. 2-bit Magnitude Comparator.
23. 2:4 Decoder with enable.
24. 1-bit Full Adder
25. Tri-State Buffer.
26. Design XNOR Gate using 4:1 Multiplexer.
27. Design 4-bit Full Adder using 1-bit Full Adder.
28. Design 3:8 Decoder with enable using 2:4 Decoder.
29. Design following using Behavioural style of coding.
30. Design D-Latch with enable.
31. Design D-FF with synchronous reset and enable.
32. MOD-5 counter with asynchronous reset and synchronous load.
33. 7-bit PIPO shift register with asynchronus reset and synchronous load.
34. 6-bit PISO shift register with synchronous load and enable.
35. 16 X 4 RAM with clk(clock),wr\_en(write enable), rd\_en(read enable), wr\_addr(write address), rd\_addr(read address), din(data input) and dout(data output).
36. 32 X 5 FIFO with clk(clock), wr\_en(write enable), rd\_en(read enable), din(data input) , dout(data output), empty(empty flag) and full(full flag).
37. Design state machine to detect sequence 10110 (Overlapping). You may use Mealy model.
38. Design state machine to detect sequence 01010 (Non-Overlapping). You may use Moore model.
39. Design a Traffic Light controller which contains following states Red, Red-Yellow, Green, Green-Yellow. The state machine stays in Red state for 25 time units, Red-Yellow state for 2 time units, Green state for 15 time units and Green-Yellow for 2 time units.

Verilog Lab-2

1. Design Decoders with following specifications

a. 2:4 using gate level modeling.

b. 4:16 using shift operator.

2. Design 8:3 priority encoder using dataflow modeling.

3. Design configurable full adder using parameter construct (Test it for 8 bit and 16 bit full adder). Use $Strobe for Displaying result on transcript.

4. Implement Y= A|B|C + A.B.E + |B.C + C|D using delay value 3 for and gates, 2 for or gates and 4 for not gates. Use two inputs and, or gates only. Use $Monitor for displaying result on transcript, also turn off display for 50-time units and then continue displaying the results.

**Note:** |B|C means (not B) and (not c)

5. Design parallel in parallel out 8-bit rotational right shift register with a clock, synchronous load input using

a. Blocking

b. Non-Blocking.

6. Design 16:1 mux using case statements.

7. Design 8:3 priority encoder using casez statements.

8. Design the following circuits

a. D latch with reset using if-else statement.

b. D flip-flop using latch designed above. Verify whether flip-flop has

**synchronous** asynchronous reset.

c. D flip-flop with asynchronous reset using If-else Statement.

Verilog Lab-3

1. Design 16:4 Priority Encoder using if-else Statement. Use $write to display the output with respect to current input on new line of the transcript.
2. 2. Design an 8-bit ALU. Features include:

a. Two 8-bit inputs A and B.

b. One 8-bit output C.

c. Five 1-bit outputs Borrow, Carry, Equal, less (A<B) and more (A>B).

d. Supports Addition, Subtraction, XOR, AND, NOR, NAND and Comparison.

Also Display Inputs, Outputs and Operation performed on transcript. Output

(8-bit) will be zero for comparison operation. Use case equality for comparison

of A and B.

1. Design a 16 bit parallel in parallel out universal Shift Register which performs

following operations:

a. Load, Shift\_enable.

b. Left Shift.

c. Right Shift.

d. Rotational Left Shift.

e. Rotational Right Shift.

1. Design a circuit which accepts two 3-bit inputs (A and B) and produces 5 1-bit outputs (gray, Excess-3, More, Less, no\_relation). The outputs are set high when:

a. Gray: A and B are gray codes to each other

b. Excess-3: A and B differ by 3.

c. More: A is one more than B.

d. Less: A is one less than B.

e. no\_relation: If none of the above relation is true.

1. Write a function to perform XOR operation on two inputs A and B (1- bit each). Using

this function, write a parity function which accepts m-bit input and returns parity. Design a module which accepts 16-bit input din and returns 1-bit output Parity using parity function.

Verilog Lab-4

1. Design a circuit which is capable of detecting “10110” pattern in an incoming

sequence (serial input first bit is '1’ then ‘0’ then ‘1’ and so on) using:

a. Non-Overlapping Mealy model

b. Overlapping Mealy model

2. Use Gate Level modeling to design a sequence detector for “1010011” pattern using:

a. Overlapping Mealy

b. Non-Overlapping Mealy

3. Design 16 x 16 bidirectional memory. Use gate primitive bufif1or bufif0 to design bidirectional data bus. Use Tri-state buffer only at the data output from the memory, it should not be present at data input to memory.

Verilog Lab-5

1. Design positive edge triggered D Flip Flop, with asynchronous active low clear and active high preset using **procedural continuous assignments** (assign and deassign). Also verify concept of force and release using stimulus.

2. Design a 4-bit counter with asynchronous reset, synchronous load and enable inputs. On reset the counter is set to 1000 and counts sequence in the order specified below (top to bottom and loop back). Counter can be loaded with any value present in the sequence. Counter should go back to reset state if loaded with some invalid count. Enable pin can be used to halt the counter if set to 0. Use $monitor to verify the functionality of the counter.

1000

0111

1011

0100

1001

0010

0101

1100

0110

0011

1111

0001

1110

1101

3. Design an 8x8 sequential multiplier. The multiplier has asynchronous reset, synchronous load and output valid signal. Use the concept of self-testing Test Bench to verify that multiplication result is correct.

Verilog Lab-6

Q1. Parity Generator:

Accept a serial stream of 8 bits on the assertion of a ‘Valid\_in’ signal. Convert the serial stream into parallel. Output the byte along with the even parity of the 8 bits on the next clock. The output should be qualified with a output ‘Valid\_out’ signal.



Q2. Write a Verilog program using the loop statements which counts the number of ZEROS in the ODD indices of 32 bits input.

Q3. Design a flow control block which takes 8-bits at i/p when valid\_in is high. The block gives 16-bits at o/p in next clock edge (8 bit previous input with 8 bit current input) with valid\_out high.



Q4. Chirp counter:

The output of the chirp counter is the following waveform.



The chirp output is high for 16 clocks, low for 15, high for 14 and so on. Then the cycle repeats.