Assignment 1

1. **Wire and Reg Demonstration**
   Write a Verilog module where:
   - net1 and net2 are wires.
   - net1 is assigned a constant value and net2 = net1.
   - Display their values using $display.
2. **Comparison of Wire and Reg**
   Implement two separate modules:
   - One using wire and assign
   - Another using reg and procedural assignment
     Compare their simulation behavior.
3. **Net Types Exploration**
   Write a Verilog module using:
   - wand, wor, tri, triand
   - Drive them using assign and simulate with #delay and $monitor.
4. **Register Assignment**
   Create a module that uses:
   - reg [7:0] data
     Assign and display a binary value.
5. **Signed Integer Simulation**
   Use integer i;
   Assign positive and negative values and simulate signed behavior.
6. **Time Simulation Variable**
   Create a time t; variable
   Store and display $time at different events using delays.
7. **Floating-Point Handling**
   Use real delta; and assign values like 4e10, 2.18.
   Observe behavior when assigned to an integer.
8. **Realtime for Simulation Time**
   Use realtime current_time = $realtime;
   Display floating-point simulation time.
9. **Bit Select Access**
   Use a vector reg [7:0] data
   Assign data = 8'b10101100
   Access specific bit using data[3].
10. **Part Select Operation**
    Extract lower nibble from reg [7:0] bus using bus[3:0].
11. **Manipulate Vector Using Part Select**
    Assign and modify specific bits using both busA[i] = bus[i]; and busA = bus[3:0];

12. **1D Memory Model**
    Declare reg [7:0] mem[7:0];
    Write and read values to/from it.
13. **2D Memory Access**
    Declare reg [7:0] mem[3:0][3:0];
    Write nested loops to assign values and print specific bytes like mem[3][2].
14. **1-Bit Full Adder Using Gates**
    Write structural code using xor, and, or gates for a full adder.
15. **4:1 Multiplexer**
    Design using gates and simulate all s1, s0 combinations.
16. **Tristate 2:1 MUX**
    Use bufif0 and bufif1 to design and test.
17. **CMOS Inverter with NMOS/PMOS**
    Implement a CMOS inverter using supply1, supply0, nmos, pmos.
18. **Switch-Level NAND**
    Implement a 2-input NAND using nmos and pmos switches manually.
19. **Combinational UDP for AND**
    Write and simulate a UDP for 2-input AND gate.
20. **UDP for Custom Logic**
    Write a UDP using the given table:
    f(x,y,z) = ~(~(x|y) | ~x & z);
21. **Sequential UDP – D Latch**
    Model a level-sensitive latch and simulate behavior.
22. **Sequential UDP – T Flip-Flop**
    Create a UDP for a T flip-flop triggered on posedge of clk.
23. **Ripple Counter Using UDP**
    Instantiate 4 T flip-flop UDPs to make a ripple counter.
24. **Array of Gate Instantiations**
    Declare 4-bit buses and instantiate an array of NAND gates.
25. **Waveform Visualization**
    Use $dumpfile and $dumpvars in all testbenches and view in GTKWave.
26. **Bit Slicing Operations**
    Perform operations on different slices of a vector (data[7:4] vs data[3:0]) and verify outputs.
27. **Advanced Memory Assignment**
    Simulate a memory block with part-select (mem[4][7:4]) and bit-select (mem[5][3]).