ECE 426/516 Implementation of VLSI Systems with HDL
Lab 5 Design of Memory with Verilog
Date: Oct 24 (Monday), 2016
Due date: Oct 31 (Monday), 2016
Late lab is partially graded (80% of full points)

Section A

In this section, you will design a ROM block, which is designed to realize a simple **squaring circuit**. Implement such a circuit using Verilog for squaring numbers up to 15 if unsigned and -8 to +7 if signed (that means 4-bit input number).

Lab Procedure

1. Use the following module template for this ROM design.

    ```
    // This ROM stores the square of a number n.
    module square_ROM (n, sign, square);

    input [3:0] n ; // signed number.
    input sign ;  // signed number if it is high and unsigned otherwise.
    output [7:0] square ; // Result = n x n.
    reg    [7:0] square ;

    always @ (n or sign)
    begin
    if (sign == 0)  // Output "square" if the input is unsigned.
      begin
      case (n)
              0 :    square <= 0 ;
              1 :    square <= ___ ;
              2 :    square <= ___ ;
    ```

```verilog
        3 :    square <= ___ ;
        4 :    square <= ___ ;
        5 :    square <= ___ ;
        6 :    square <= ___ ;
        7 :    square <= ___ ;
        8 :    square <= ___ ;
        9 :    square <= ___ ;
       10 :    square <= ___ ;
       11 :    square <= ___ ;
       12 :    square <= ___ ;
       13 :    square <= ___ ;
       14 :    square <= ___ ;
       15 :    square <= 255 ;
        default :    square <= 0 ;        // Clear the result.
  endcase
  end

else
  begin
  case (n)    // Output "square" if the input is unsigned.
        0 :    square <= 0 ;
        1 :    square <= 1 ;
        2 :    square <= 4 ;
        3 :    square <= ___ ;
        4 :    square <= ___ ;
        5 :    square <= ___ ;
        6 :    square <= ___ ;
        7 :    square <= ___ ;
        8 :    square <= ___ ;
        9 :    square <= ___ ;
       10 :    square <= ___ ;
       11 :    square <= ___ ;
       12 :    square <= ___ ;
       13 :    square <= ___ ;
       14 :    square <= ___ ;
       15 :    square <= ___ ;
        default :    square <= 0 ;        // Clear the result.
  endcase
  end
end
```

endmodule

2. Write a test bench to test your behavioral ROM design be simulating the functionality. Test your unsigned and signed inputs in your test bench.

Section B

In this section, you will design a ROM based **square root circuit** for unsigned numbers in the range of 0 to 15. Provide three digits after decimal point. Such as if the input number is a 2, the output should be the square root of 2, which is 1.414. Complete the following Verilog code and write a test bench and verify your results.

```
/ This ROM stores the square root of a number n.

module square_root_ROM (n, sq_root) ;

input [3:0] n ; // Unsigned number.
output [11:0]  sq_root ; // Result = Square root of n.
reg  [11:0] sq_root  ;  // Msb 2 bits (sq_root_int)is the integer part and the
                       // other bits (sq_root_dp) are after the decimal point.
wire [1:0] sq_root_int ;
wire [9:0] sq_root_dp ;

always @ (n)

begin

  case (n) // Output "square root" .

          0 :   sq_root  <= 0 ;
          1 :   sq_root <= _____ ;
          2 :   sq_root <= _____ ;
          3 :   sq_root <= _____ ;
          4 :   sq_root <= _____ ;
          5 :   sq_root <= _____ ;
          6 :   sq_root <= _____ ;
          7 :   sq_root <= _____ ;
          8 :   sq_root <= _____ ;
          9 :   sq_root <= _____ ;
```

```
10 :   sq_root <= _____ ;
11 :   sq_root <= _____ ;
12 :   sq_root <= _____ ;
13 :   sq_root <= _____ ;
14 :   sq_root <= _____ ;
15 :   sq_root <= _____ ;

default :    sq_root <= 0 ;  // Clear the result.

  endcase

end

assign sq_root_int = _____ ;
assign sq_root_dp  = _____ ;

endmodule
```

Demo

You should demo the following aspects of your design to TA.
1. Verilog code for ROM based square and square-root of circuits.
2. Simulation waveforms demonstrating correct functionality of both designs.