ECE 426/516 **Implementation of VLSI Systems with HDL**

Homework #7 Design of Memory

Due date: April 8th, 2024, by 11:30pm

Please upload your solution to D2L

Question: A ROM can be used to multiply two binary numbers by splitting the address lines to accommodate the two numbers. Implement using Verilog such as a multiplier for multiplying two signed numbers, each of size 4 bits.

(a) Part of the Verilog code has been provided below. Complete the Verilog code and verify your results by simulation.
(b) Will this be an efficient implementation if used for two 8-bit, unsigned numbers? Discuss your reason.

```
// This ROM stores the product of two numbers n1 and n2. Both the numbers
and "result" are in twos complement.

module rom (n1, n2, result);

input   [3:0]   n1 ;          // Signed first number.
input   [3:0]   n2 ;          // Signed second number.
output  [7:0]  result ;       // Result = n1 x n2.

wire    [7:0]   result ;
wire    [3:0]   n1_mag ;
wire    [3:0]   n2_mag ;

reg     [7:0]   product ;

assign n1_mag = _____ ; // get magnitude of n1
assign n2_mag = _____ ; // get magnitude of n2

always @ (n1_mag or n2_mag)

begin

  case ({n1_mag, n2_mag})

            17 :    product = ____ ;
```

18 : product = ____ ;
19 : product = ____ ;
20 : product = ____ ;
21 : product = ____ ;
22 : product = ____ ;
23 : product = ____ ;
24 : product = ____ ;

33: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;

49 : product = ____ ;
___: product = ___ ;
___: product = ____;
___: product = ___ ;
___: product = ___ ;
___: product = ___ ;
___: product = ___ ;
___: product = ___ ;

65 : product = ___ ;
___ : product = ___ ;
___ : product = ___ ;
___ : product = ___ ;
____: product = ___ ;
___ : product = ___ ;
___ : product = ___ ;
____: product = ___ ;

81 : product = _____;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;
___: product = ____ ;

```
            ___:    product = _____ ;

             97 :    product = ___ ;
            ___:    product =  ___ ;
            ___:    product = _____;
            ___:    product = ___ ;
            ___:    product = ___ ;
            ___:    product = ___ ;
            ___:    product = ___ ;
            ___:    product = ___ ;

             113:    product = ___ ;
            ___ :    product = ___ ;
            ___ :    product = ___ ;
            ___ :    product = ___ ;
            _____:   product = ___ ;
            ___ :    product = ___ ;
            ___ :    product = ___ ;
            _____:   product = ___ ;

             129:    product = ___ ;
            ___ :    product = ___ ;
            ___ :    product = ___ ;
            ___ :    product = ___ ;
            _____:   product = ___ ;
            ___ :    product = ___ ;
            ___ :    product = ___ ;
            _____:   product = ___ ;

        default :       product = 0 ;                   // Clear the result.

    endcase

end

assign result = _____;

endmodule
```