Solutions to Quiz 1

Question 1

There were four versions of this question with two different bit widths, and two different output signal names and behaviour.

Write a Verilog module named **check** with two 15 (or 7)-bit inputs named **s1** and **s2** and a 1-bit output named **both** (or **either**). **both** (or **either**) should be set to **1** if both **s1** and **s2** (or either **s1** or **s2**) are non-zero. **both** (or **either**) should be set to zero otherwise. Declare arrays with decreasing bit order.

Answer

Some correct and incorrect solutions are shown below. Incorrect solutions include examples that the give incorrect results.

```
module check
   (
    input logic [6:0] s1, s2,
    // input logic [14:0] s1, s2,
    // output logic both ) ;
    output logic either ) ;

   /* correct: */

   // assign both = s1 && s2 ;
   // assign both = s1 && s2 ? '1 : '0 ;
   // assign both = s1 ? ( s2 ? '1 : '0 ) : '0 ;

   assign either = s1 || s2 ;
   // assign either = s1 || s2 ? '1 : '0 ;
   // assign either = s1 ? '1 : ( s2 ? '1 : '0 ) ;

   /* incorrect: */

   // incorrect: 2&1 = 0
   // assign both = s1 & s2 ;

   // incorrect: 7'd64*7'd02 = 0
   // assign both = s1 * s2 ? '1 : '0 ;

   // incorrect: 2|2 = 0 (truncated to 1 bit)
   // assign either = s1 | s2 ;

   // incorrect: b7'd64+7'd64 = 0 (overflow)
   // assign either = s1 + s2 ? '1 : '0 ;

endmodule
```

A "testbench" module that tests the solution is shown in Appendix A.

Question 2

Given the following Verilog declarations:

```
logic [11:0] x = 12'd32 ;
logic  [7:0] y = 8'b101 ;
```

or

```
logic [11:0] x = 12'd64 ;
logic  [7:0] y = 8'b111 ;
```

fill in in the table below with the value of the each expression as a hexadecimal (base 16) number (Verilog format not necessary) and the length (number of bits) as a decimal (base 10) number:

expression	value (in hexadecimal)	length (in bits, decimal)
{y,y}		
x[7:4]		
x && y		
x + y		

Answer

For the first version of the question these are:

```
  {y,y}: value: 505 bits: 16
 x[7:4]: value: 2 bits:  4
 x && y: value: 1 bits:  1
  x + y: value: 25 bits: 12
```

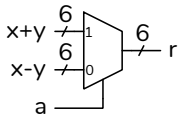
and for the second:

```
  {y,y}: value: 707 bits: 16
 x[7:4]: value: 4 bits:  4
 x && y: value: 1 bits:  1
  x + y: value: 47 bits: 12
```

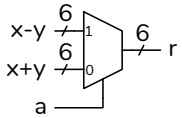
A Verilog module in Appendix A displays the values and lengths of each expression.

Question 3

Write a Verilog statement that would implement the schematic on the right. You can assume it would replace the "missing statement" comment in the **quiz1** module:



or




```
module quiz1
   ( input logic [5:0] x, y,
     input logic a,
     output logic [5:0] r ) ;

   // missing statement

endmodule
```

Answer

The statements for the two schematics are:

```
assign r = a ? x+y : x-y ;
```

and

```
assign r = a ? x-y : x+y ;
```

respectively.

The synthesis results are shown in Appendix A.

Appendix A

Question 1 Testbench

```
module test ;

   logic a, b, c, d ;

   check c0 (0,0,a) ;
   check c1 (0,2,b) ;
   check c2 (64,0,c) ;
   check c3 (64,2,d) ;

   initial begin
      #1 ;
      $display(a,b,c,d) ;
   end

endmodule
```

The result for **both** should be 0001 (only the last test case outputs a 1) and the result for **either** should be 0111 (only the first test case outputs a 0).

Question 2 Testbench

```
`define ans(expr) \
   $display("%16s: value: %0h bits: %2d", \
            `"expr`",expr,$bits(expr)) ;
module test ;
   logic [11:0] x = 12'd32 ;
   logic  [7:0] y = 8'b101 ;
   // logic [11:0] x = 12'd64 ;
   // logic [7:0]  y = 8'b111 ;
   initial begin
      `ans({y,y})
      `ans(x[7:4])
      `ans(x && y)
      `ans(x + y)
   end
endmodule
```

Question 3 Synthesis Results

In the schematics below subtraction is implemented by adding the two's complement (inverting the bits and adding 1).