Solutions to Midterm Exam

Question 1

The two versions of this question used the same expressions arranged in different orders. The following code computes the size and value of each expression:

```
// solutions for midterm exam question 1
module midterm ;

   logic [7:0] a = 8'h08 ;
   logic [3:0] b [3:0]  = '{ 4'bxxxx, 4'h1, 4'd2, 4'b0011 } ;
   logic signed [7:0] c = 8'b1 ;

`define ans(expr) \
   $display("%16s: bits: %2d value: %8b", \
            `"expr`",$bits(expr),expr) ;

   initial begin
      `ans( 1 && 2 ) ;
      `ans( {4{c[0]}} ) ;
      `ans( b[0][1:0] ) ;
      `ans( 4'sb1000 >>> 3 ) ;
      `ans( ^c ) ;
      `ans( {a[0],b[0]} ) ;
      `ans( c[0] ? b[2] : b[1] ) ;
      `ans( 2*c ) ;
      `ans( a|c ) ;
   end

endmodule
```

and the output is:

```
          1 && 2: bits:  1 value: 00000001
       {4{c[0]}}: bits:  4 value: 00001111
       b[0][1:0]: bits:  2 value: 00000011
  4'sb1000 >>> 3: bits:  4 value: 00001111
              ^c: bits:  1 value: 00000001
     {a[0],b[0]}: bits:  5 value: 00000011
c[0] ? b[2]:b[1]: bits:  4 value: 00000001
             2*c: bits: 32 value: 00000010
             a|c: bits:  8 value: 00001001
```

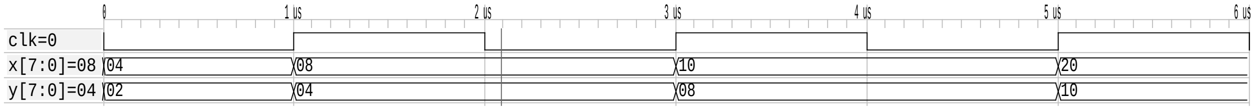
Question 2

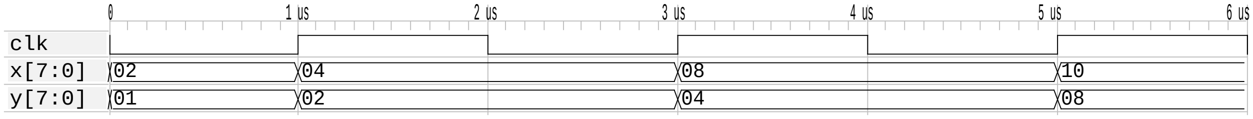
(a) There were two versions of the question. In both the clock signal is initialized to 0 and the simulation runs for six delays of 1 μs (6 μs total). There are two rising clock edges (at 1, 3 and 5 μs).

(b) In both versions of the question `x` is always set to `y+y` in an `always_comb` block so the values are always twice those of `y`. In one case the initial value of `y` is 2 and so the initial value of `x` is 4. In the one case the initial value of `y` is 1 and so the initial value of `x` is 2.

(c) The `always_ff` block executes and updates `y` on each rising clock edge. The value of `y` is set to twice the current value. So in the first version `y` is set to (2,4,8,'h10); in the second it is set to (1,2,4,8) as shown below.

The simulation results are as follows:


0
1 us
2 us
3 us
4 us
5 us
6 us
clk=0
x[7:0]=08
y[7:0]=04
04
08
10
20
02
04
08
10


0
1 us
2 us
3 us
4 us
5 us
6 us
clk
x[7:0]
y[7:0]
02
04
08
10
01
02
04
08