Assignment 1

Due Tuesday, February 13. Submit your assignment using the appropriate dropbox on the course web site. Assignments submitted after the solutions are made available will be given a mark of zero.

Question 1

This questions asks you to explore different ways of describing combinational logic functions in System Verilog.

A half adder has two one-bit inputs and two one-bit outputs: the sum and the carry. It can be described by the following truth table:

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

where `a` and `b` are the inputs and `sum` and `carry` the outputs.

Write a System Verilog module that computes the sum and carry in six different ways:

- using only logic operators (&, |, ^)
- using only ternary operators (`?:`)
- using only if/else statements
- using only a case statement
- using only concatenation (`{,}`), addition (+) and bit subscript (slicing) (`[:]`) operators
- using only initialized array(s), concatenation (`{,}`) and the indexing operator (`[]`)

Question 2

You can implement multiplication as addition of shifted versions of a multiplicand. The additions are controlled by the bits of the multiplier. For example, to multiply `a=4'b1010 (10)` by `b=4'b011 (3)` we would add `c= (a<<0)*b[0]+ (a<<1)*b[1]+ (a<<2)*b[2]= 1010+10100+000000=11110 (30)`.

Complete the following System Verilog 8 × 8 multiplier module. Use a for-loop so your design has only combinational logic.

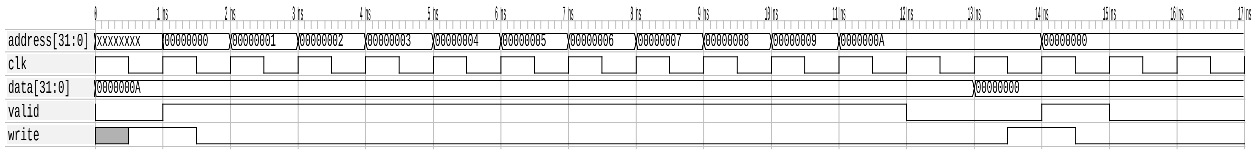
```
module mul8 ( input logic [7:0] a,
              input logic [7:0] b,
              output logic[15:0] c ) ;

    always_comb begin
       // your code here
    end

endmodule

module mul8_tb  ;

    logic [15:0] res ;

    mul8 m0 (8'd12, 8'd34, res ) ;

    initial begin
       #1 $display ( "The product is %d\n", res ) ;
       $finish ;
    end

endmodule
```

Test you design using the last four digits of your BCIT ID as the two two-digit test values. For example, if your BCIT ID number is A00123456 then your example should multiply 34 and 56.

Your answer should include your code and a screen capture of the transcript window showing the correct result.

Question 3

Design a controller called `asg1` with the following interface:



bits	name	in/out	notes
1	`write`	in	active-high
32	`data`	in	
1	`clk`	in	
32	`address`	out	
1	`valid`	out	active-high

The controller operates as follows:

The outputs change only on the rising edge of `clk`. When `write` is asserted `address` is set to 0, the value on the `data` input is stored, and `valid` is asserted. On each subsequent clock pulse the `address` output increases by 1 until it reaches the value stored. `valid` is de-asserted on the next clock cycle.

(a) Write the state transition table.

(b) Draw the state transition diagram.

(c) Write a synthesizable System Verilog description. Your design should use only one `always_comb` block and one `always_ff` block.

(d) Test your design using the testbench supplied on the course web site.

The simulation results are shown above.