Solutions to Quiz 1

There were two versions of each question. The values and the answers for the two versions are given below.
Revision 2: Added **`reset!=0`** *transition conditions in Question 5.*

Question 1

Define a Verilog lookup table named **`square`** (or **`double`**) that can be used to determine the square (or double the value) of a number between 0 and 7. The result should be a 6-bit value between between 0 and 49 (or 14). For example, the expression **`square[5]`** should have the value **`6'd25`** (or **`double[5]`** should have the value **`6'd10`**). Write only the lookup table definition, not a complete module.

Solution

Following the example in the "Lookup Tables" section of Lecture 1:

```
                        // 0  1  2  3  4  5  6  7
logic [0:7][5:0] square = '{ 0, 1, 4, 9,16,25,36,49} ;
logic [0:7][5:0] double = '{ 0, 2, 4, 6, 8,10,12,14} ;
```

Question 2

What minimum size of counter is needed to implement a timer that has a duration of 10 (or 20) ms using a 2.5 MHz clock? Your answer should be the number of bits. Show your work.
If this timer's counter counts down to zero, what range of values will the counter take on?

Solution

Following the method in Exercise 3 of Lecture 2:

- For a clock frequency of $f_{clock} = 2.5$ MHz, the period is $T_{clock} = 1/2.5\times10^{6}$ Hz $= 400$ ns.
- A timer duration of 10 ms requires $N = 10\times10^{-3}/400\times10^{-9} = 25000$ clock cycles.
- The smallest number of bits that can represent this number of values is $m \geq \log_2 N = \log_2 25000$ so we need $\boxed{15}$ bits.
- If the timer counts down to zero then it must count from $N-1$ to 0 or $\boxed{24999 \text{ to } 0}$.

For a timer duration of 20 ms $N = 20\times10^{-3}\,\text{s}/400\times10^{-9}\,\text{s} = 50000$ clock cycles, $\lceil\log_2 50000\rceil = \boxed{16}$ is the smallest number of bits that can represent values from 0 to 49999 and the counter will count down from $\boxed{49999 \text{ to } 0}$.

Question 3

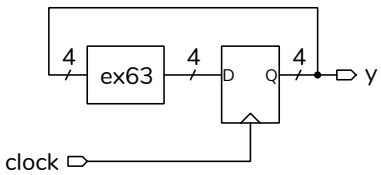
Write a Verilog module named **`ymod`** corresponding to the block diagram below. The block labelled **`ex63`** represents a module named **`ex63`** defined as:

```
module ex63
  ( input  logic [3:0] a,
    output logic [3:0] b ) ;
   //...
endmodule
```

Write only the module **`ymod`**. The module inputs and outputs should be as shown in the diagram. The module should include definitions for any additional signal(s) required to instantiate the **`ex63`** module.

Solutions

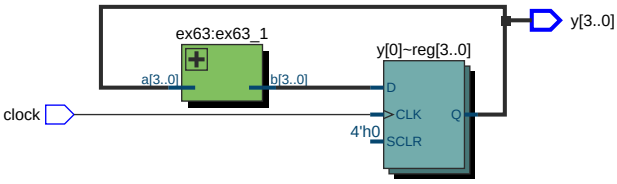
For the block diagram:



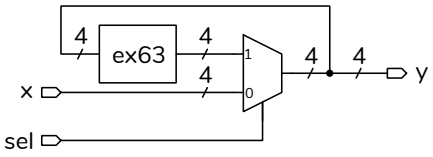
the corresponding Verilog is:

```
module ymod
  ( input logic clock,
    output logic [3:0] y ) ;

   logic [3:0] t ; // output of ex63

   ex63 ex63_1 (y, t) ;

   always_ff @(posedge clock)
     y <= t ;

endmodule
```

which Quartus synthesizes to:



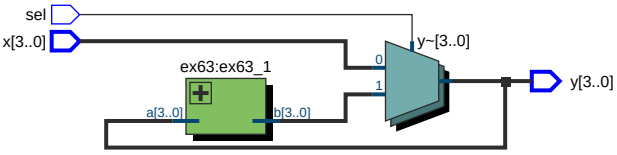
For the block diagram:



the corresponding Verilog is:

```
module ymod
  ( input logic [3:0] x,
    input logic sel,
    output logic [3:0] y ) ;

   logic [3:0] t ; // output of ex63

   ex63 ex63_1 (y, t) ;

   assign y = sel ? t : x ;

endmodule
```

which Quartus synthesizes to:



Question 4

A state machine has a an input named **rst**, a clock input named **clk** and an output named **s**. **s** can take on the values **000**, **001**, **011**, and **111**. The state machine operates as follows:

- If **rst** is asserted (has the value 1) then **s** is set to **000** (or **111**).
- If **rst** is not asserted then **s** takes on the values **001**, **011** and **111** (or **011**, **001**, and **000**), in that order, and then stays at the value **111** (or **000**) until **rst** is asserted again.

Write a state transition table for this state machine. Include columns for the current state, the **rst** input and the next state.

Solutions

For the Verilog description:

state	**rst**	next state
xxx	1	000
000	0	001
001	0	011
011	0	111
111	0	111

For the Verilog description:

state	**rst**	next state
xxx	1	111
111	0	011
011	0	001
001	0	000
000	0	000

Question 5

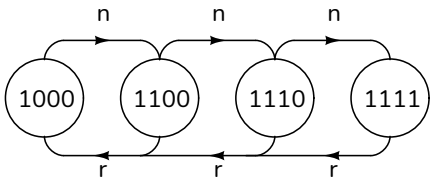
Below is the Verilog code for a state machine. Draw the state transition diagram. Label all states. Label all transition conditions using Verilog expressions.

Solutions

For this code:

```
module blink
  ( input  logic clock, reset,
    output logic [3:0] leds ) ;

   always_ff @(posedge clock)
     leds <= reset ? 4'b1000 :
             leds == 4'b1000 ? 4'b1100 :
             leds == 4'b1100 ? 4'b1110 :
             leds == 4'b1110 ? 4'b1111 : leds ;

endmodule
```

the state variable is **leds**, the input is **reset** and the transition diagram would be:



r: reset == 0
n: reset != 0

For this version of the question:

```
module blink
  ( input  logic clock, reset,
    output logic [3:0] leds ) ;

   always_ff @(posedge clock)
     leds <= reset ? 4'b1111 :
             leds == 4'b1111 ? 4'b0011 :
             leds == 4'b0011 ? 4'b0001 :
             leds == 4'b0001 ? 4'b0000 : leds ;

endmodule
```

the state transition diagram would be: