Solutions to Midterm Exam 1

Revision 3.

Question 1

The columns in each row of the following tables match. There were two versions of this question:

(a)

width (bits)	base	value (decimal)	Verilog literal
9	decimal (10)	19	9'd19
8	binary (2)	4	8'b100
16	hex (16)	17	16'h11
8	hex (16)	20	8'h14

(b)

width (bits)	base	value (decimal)	Verilog literal
19	decimal (10)	9	19'd9
8	binary (2)	4	8'b100
12	hex (16)	17	12'h11
8	hex (16)	21	8'h15

Question 2

Module definitions for:

(a) a module named **uart** with two single-bit **logic** inputs named **clock** and **sin**, and an 8-bit **logic** output named **data**; and

(b) a module named **dac** with a 10-bit **logic** input named **val**, a one-bit **logic** input named **sclk** and a one-bit **logic** output named **err**.

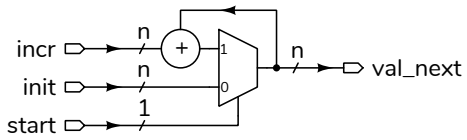
are:

```
// ELEX 2117 Fall 2021
// Midterm 1 Question 2
// Ed. Casas 2021-10-17

module uart
  (
   input logic clock, sin,
   output logic [7:0] data
   ) ;

endmodule

module dac
  (
   input logic [9:0] val,
   input logic sclk,
   output logic err
   ) ;

endmodule
```

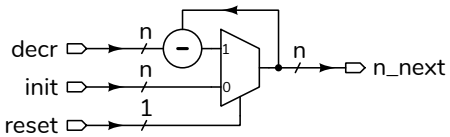
Question 3

Verilog **assign** statements that implement the following schematics are given below

(a)



(b)




```
// ELEX 2117 Fall 2021
// Midterm 1 Question 3
// Ed. Casas 2021-10-17

module test ;
   logic [7:0] incr, decr, init,
               val_next, n_next ;
   logic start, reset ;

   // (a)
   assign val_next = start ? val_next + incr : init ;

   // (b)
   assign n_next = reset ? n_next - decr : init ;

endmodule
```

Note that these are not realistic examples because the assign statements set up combinational logic feedback loops and these could not be synthesized.

Question 4

The values in each row below are consistent.

In these tables the correspondence is between Verilog values **`0/1`** and the logic levels (**`H/L`**) as would be the case in the context of an input or output signal rather than between **`0/1`** and the truth value (**`T/F`**) as would be the case in the context of a logical expression.

(a)

signal name	truth value (T/F)	logic level (H/L)	Verilog value (0/1)
`off`	**`T`**	**`H`**	**`1`**
`open*`	**`T`**	**`L`**	**`0`**
`closed`	**`F`**	**`L`**	**`0`**
$\overline{\texttt{off}}$	**`F`**	**`H`**	**`1`**

(b)

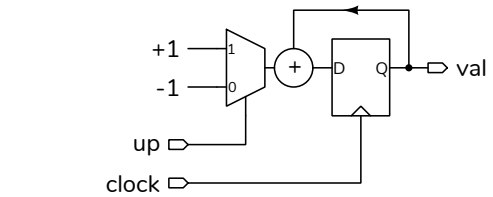
signal name	truth value (T/F)	logic level (H/L)	Verilog value (0/1)
`on`	**`T`**	**`H`**	**`1`**
`closed*`	**`T`**	**`L`**	**`0`**
`open`	**`F`**	**`L`**	**`0`**
$\overline{\texttt{on}}$	**`F`**	**`H`**	**`1`**

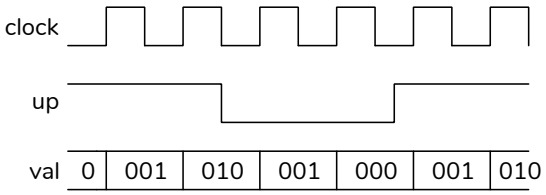
Question 5

The following schematics show 3-bit registers whose values, **`val`** or **`num`**, increase or decrease by 1 under control of an **`up`** input. The initial value of the register is zero.

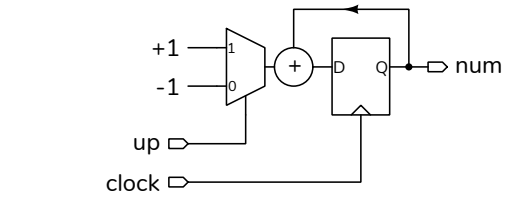
The vertical lines show where the output changes — on the rising edge of the clock. The output value is shown as a binary number in-between each change.

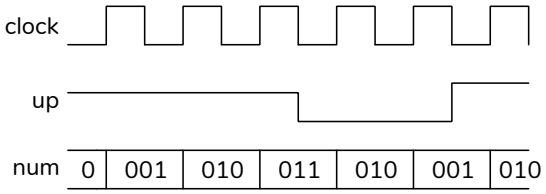
(a)





(b)





Question 6

A Verilog module implementing a state machine is declared as follows:

```
// Version (i)

module bitgrow
  (
   input logic clock, reset, enable,
   output logic [2:0] bits
   ) ;

  // ... your code goes here ...

endmodule

// Version (ii)

module bitshrink
  (
  input logic clock, reset, enable,
  output logic [2:0] bits
  ) ;

  // ... your code goes here ...

endmodule
```

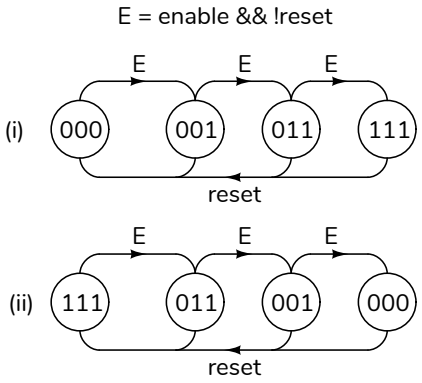
The **`bits`** output changes on the rising edge of **`clock`** – a periodic clock input. The **`bits`** output can be **`3'b000`**, **`3'b001`**, **`3'b011`**, or **`3'b111`**. The **`reset`**

and **enable** inputs control the state machine as follows for the two versions of this question:

(i)
- if reset is asserted (H), the output is set to **3'b000**,
- otherwise, if enable is not asserted (L), the output doesn't change,
- otherwise, if enable is asserted (H) then the value with one more bit set to **1** is output unless the output is already **3'b111**; in this case the output doesn't change.

(ii)
- if reset is asserted (H), the output is set to **3'b111**,
- otherwise, if enable is not asserted (L), the output doesn't change,
- otherwise, if enable is asserted (H) then the value with one less bit set to **1** is output unless the output is already **3'b000**; in this case the output doesn't change.

(a) The state transition diagrams below label each state with the corresponding value of **bits**. The state transitions are labelled with the values of **reset** and **enable** required for that transition. Transitions that do not result in a change of state are not shown.



(b) The Verilog code required to implement and test these state machine is shown below.

```
// ELEX 2117 Fall 2021
// Midterm 1 Question 6
// Ed. Casas 2021-10-7

module bitgrow
  (
   input logic clock, reset, enable,
   output logic [2:0] bits
   ) ;

   logic [2:0] bits_next ;

   assign bits_next
     = reset ? 3'b000 :
       !enable ? bits :
       {bits,1'b1} ;

   always @(posedge clock) bits = bits_next ;

endmodule

module bitshrink
  (
   input logic clock, reset, enable,
   output logic [2:0] bits
   ) ;

   logic [2:0] bits_next ;

   assign bits_next
     = reset ? 3'b111 :
       !enable ? bits :
       bits[2:1] ;

   always @(posedge clock) bits = bits_next ;

endmodule

// testbench

module midterm1_q6_tb ;

   logic clock, reset, enable ;
   logic [2:0] growbits, shrinkbits ;

   bitgrow   m0 (.bits(growbits),   .* ) ;
   bitshrink m1 (.bits(shrinkbits), .* ) ;

   logic [1:0] tv []
               = '{ 2'b11, 2'b11, 2'b10, 2'b00,
                    2'b01, 2'b01, 2'b01, 2'b01,
                    2'b01 } ;

   initial begin
      $dumpfile("midterm1_q6.vcd") ;
      $dumpvars ;
      clock = '0 ;
      for ( int i=0; i<$size(tv) ; i++ ) begin
         {reset,enable} = tv[i] ;
         #100ns ;
      end
      $finish ;
   end

   always #50ns clock = ~clock ;

endmodule
```

The simulation output waveforms in 1 demonstrate the required behaviour.

These designs assume the state variable, **bits**, always has a valid value. If this is not a good assumption then it would be necessary to check for invalid values and take appropriate action (e.g. reset to **3'b000**).

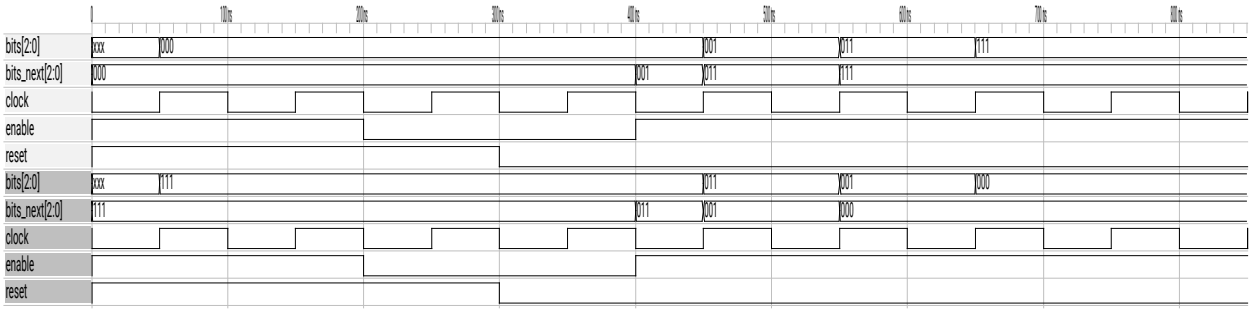


Figure 1: Question 6 Simulation Results.

Question 7

Given the following Verilog signal declarations and initial values the following tables show the lengths (in bits) and values (in binary) of each of four expressions. Verilog-syntax separators (_) have been used in the binary numeric literals to demonstrate how the results were computed.

(a)
```
logic x[7:0] = 8'h12 ; // 0001_0010.
logic y[3:0] = 4'd2 ;  // 0010.
```

expression	length (bits)	value (binary)
`x + y`	`max(8,4) = 8`	`0001_0100`
`x << 4'b2`	`8`	`01_0010_00`
`x[3:0]`	`4`	`0010`
`{x,y}`	`8+4 = 12`	`0001_0010_0010`

(b)
```
logic x[7:0] = 8'h14 ; // 0001_0100.
logic y[3:0] = 4'd1 ;  // 0001.
```

expression	length (bits)	value (binary)
`x + y`	`max(8,4) = 8`	`0001_0101`
`x << 4'b2`	`8`	`01_0100_00`
`x[3:0]`	`4`	`0100`
`{x,y}`	`8+4 = 12`	`0001_0100_0001`