Problem 3: [2 points] Drill problem
Filename: library.sv
AndrewID: xinyew

```
`default_nettype none
module Decoder
  (input  logic en,
   input  logic [2:0] I,
   output logic [7:0] D);

  always_comb begin
    unique case (I)
      3'b000: D = 8'b00000001;
      3'b001: D = 8'b00000010;
      3'b010: D = 8'b00000100;
      3'b011: D = 8'b00001000;
      3'b100: D = 8'b00010000;
      3'b101: D = 8'b00100000;
      3'b110: D = 8'b01000000;
      3'b111: D = 8'b10000000;
    endcase
    D = en ? D : 0;
  end

endmodule : Decoder

module BarrelShifter
  (input  logic [15:0] v,
   input  logic [3:0] by,
   output logic [15:0] s);

  always_comb begin
    s = by[3] ? v << 8 : v;
    s = by[2] ? s << 4 : s;
    s = by[1] ? s << 2 : s;
    s = by[0] ? s << 1 : s;
  end

endmodule : BarrelShifter

module Multiplexer
  (input  logic [7:0] I,
   input  logic [2:0] s,
   output logic Y);

  always_comb
    case (s)
      3'b000: Y = I[0];
      3'b001: Y = I[1];
      3'b010: Y = I[2];
      3'b011: Y = I[3];
      3'b100: Y = I[4];
      3'b101: Y = I[5];
      3'b110: Y = I[6];
      3'b111: Y = I[7];
    endcase

endmodule: Multiplexer

module Mux2to1
  (input  logic [6:0] I0,
   input  logic [6:0] I1,
   input  logic S,
   output logic [6:0] Y);

  always_comb
    case (S)
      0: Y = I0;
      1: Y = I1;
    endcase

endmodule : Mux2to1

```

```
module MagComparator
  (input  logic [7:0] A,
   input  logic [7:0] B,
   output logic AltB,
   output logic AeqB,
   output logic AgtB);

   assign AltB = A < B;
   assign AgtB = A > B;
   assign AeqB = A == B;

endmodule : MagComparator

module Comparator
  (input  logic [3:0] A,
   input  logic [3:0] B,
   output logic AeqB);

   assign AeqB = A == B;

endmodule : Comparator

module Adder
  (input logic [4:0] A,
   input logic [4:0] B,
   input logic cin,
   output logic cout,
   output logic [4:0] sum);

  assign {cout, sum} = A + B + cin;
endmodule : Adder
```

Problem 3: [2 points] Drill problem
Filename: library_tests.sv
AndrewID: xinyew

```
`default_nettype none
module Decoder_test;
  logic en;
  logic [2:0] I;
  logic [7:0]D;

  Decoder DUT(.en(en),
              .I(I),
              .D(D));

  initial begin
    $monitor($time,,
      "en = %b, I = %b, D = %b", en, I, D);
    en = 1;
    for (I = 3'b000; I < 3'b111; I++)
      #1;
    en = 0;
    for (I = 3'b000; I < 3'b111; I++)
      #1;
    #1 $finish;
  end

endmodule: Decoder_test

module BarrelShifter_test;
  logic [15:0] v;
  logic [3:0] by;
  logic [15:0] s;

  BarrelShifter DUT(.v(v),
                    .by(by),
                    .s(s));

  initial begin
    $monitor($time,,
      "v = %b, by = %b, s = %b", v, by, s);
    v = 16'b0101010101010101;
    for (by = 4'b0000; by < 4'b1111; by++)
      #1;
    #1 $finish;
  end

endmodule : BarrelShifter_test

module Multiplexer_test;
  logic [7:0] I;
  logic [2:0] s;
  logic Y;

  Multiplexer DUT(.I(I),
                  .s(s),
                  .Y(Y));

  initial begin
    $monitor($time,,
      "I = %b, s = %b, Y = %b", I, s, Y);
    I = 8'b01010101;
    for (s = 3'b000; s < 3'b111; s++)
      #1;
    #1 $finish;
  end

endmodule : Multiplexer_test

module Mux2to1_test;
  logic [6:0] I0;
  logic [6:0] I1;
  logic S;
  logic [6:0] Y;
  Mux2to1 DUT(.I0(I0),
              .I1(I1),
              .S(S),
              .Y(Y));

  initial begin
    $monitor($time,,
      "I0 = %b, I1 = %b, S = %b, Y = %b", I0, I1, S, Y);
    I0 = 7'b0000000;
    I1 = 7'b1111111;
    S = 0;
    #1 S = 1;
    #1 $finish;
  end

endmodule : Mux2to1_test

module MagComparator_test;
  logic [7:0] A;
  logic [7:0] B;
  logic AltB;
  logic AgtB;
  logic AeqB;

  MagComparator DUT (.A(A),
                     .B(B),
                     .AltB(AltB),
                     .AgtB(AgtB),
                     .AeqB(AeqB));

  initial begin
    $monitor($time,,
    "A = %b, B = %b, AltB = %b, AgtB = %b, AeqB = %b", A, B , AltB, AgtB, AeqB);
    A = 8'b00000000;
    B = 8'b11111111;
    #2 A = 8'b11111111;
    #2 B = 8'b00000000;
    #2 $finish;
  end

endmodule : MagComparator_test

module Comparator_test;
  logic [3:0] A;
  logic [3:0] B;
  logic AeqB;

  MagComparator DUT (.A(A),
                     .B(B),
                     .AeqB(AeqB));

  initial begin
    $monitor($time,,
    "A = %b, B = %b, AeqB = %b", A, B, AeqB);
    A = 8'b00000000;
    B = 8'b11111111;
    #2 A = 8'b11111111;
    #2 $finish;
  end

endmodule : Comparator_test

module Adder_test;
  logic [4:0] A;
  logic [4:0] B;
  logic [4:0] sum;
  logic cin;
  logic cout;

  Adder DUT (.A(A),
             .B(B),
             .cin(cin),
             .cout(cout),
             .sum(sum));

  initial begin
    $monitor($time,,
    "A = %b, B = %b, cin = %b, sum = %b, cout = %b",
    A, B, cin, sum, cout);
       A = 5'b00000;
       B = 5'b00000;
       cin = 1'b0;
    #2 cin = 1'b1;
    #2 A = 5'b00001;
    #2 B = 5'b00001;
    #2 A = 5'b11111;
       cin = 1'b0;
    #2 cin = 1'b1;
    #2 $finish;
  end

endmodule : Adder_test
```

Problem 10: [10 points]
Filename: hw3prob10.sv
AndrewID: xinyew

```
`default_nettype none
module hw3prob10
  (input logic [7:0] a, b,
   output logic [7:0] sum,
   output logic overflow);

  logic [6:0] mag_a, mag_b , max, min;
  logic [6:0]mag_sum;
  logic sign_a, sign_b, sign_sum;

  assign mag_a = a[6:0] ;
  assign mag_b = b[6:0] ;
  assign sign_a = a[7];
  assign sign_b = b[7] ;

  always_comb begin
    if (mag_a > mag_b) begin
      max = mag_a;
      min = mag_b;
      sign_sum = sign_a;
    end
    else begin
      max = mag_b;
      min = mag_a;
      sign_sum = sign_b;
    end

    if (sign_a==sign_b) begin
      {overflow, mag_sum} = max + min;
    end
    else begin
      mag_sum = max - min;
      overflow = 0;
    end


    sum = {sign_sum, mag_sum};

  end
endmodule : hw3prob10

module hw3prob10_test;
  logic [7:0] a, b, sum;
  logic overflow;

  hw3prob10 DUT (.a(a),
                 .b(b),
                 .sum(sum),
                 .overflow(overflow));

  initial begin
    $monitor($time,,
      "a = %b, b = %b, sum = %b, overflow = %b",
      a, b, sum, overflow);
    a = 8'b00000000;
    b = 8'b11111111;
    #1 a = 8'b00000001;
    #1 a = 8'b10000001;
    #1 a = 8'b11111111;
    #1 a = 8'b01111111;
       b = 8'b10000000;
    #1 b = 8'b10000001;
    #1 b = 8'b00000001;
    #1 b = 8'b11111111;
    #1 b = 8'b01111111;
    #1 $finish;
  end
endmodule : hw3prob10_test
```