Assignment 2

1)
State table:

Current state Yp	X (size 1)	Z (size 1)	Next state Yn
00 (remainder = 0)	0	0	00 (remainder = 0)
00 (remainder = 0)	1	0	01 (remainder = 1)
01 (remainder = 1)	0	0	10 (remainder = 2)
01 (remainder = 1)	1	1	00 (remainder = 0)
10 (remainder = 2)	0	1	01 (remainder = 1)
10 (remainder = 2)	1	1	10 (remainder = 2)

For Z

Yp \ X	**0**	**1**
00	0	0
01	0	1
10	1	1

Z = (Yp1 && !Yp0) || (!Yp1 && Yp0 && X)

For Yn1

Yp \ X	**0**	**1**
00	0	0
01	1	0
10	0	1

Yn1 = (!Yp1 && Yp0 && !X) || (Yp1 && !Yp0 && X)

For Yn0

Yp \ X	**0**	**1**
00	0	1
01	0	0
10	1	0

Yn0 = !Yp0 && [(!Yp1 && X) || (Yp1 && !X)]

Code:

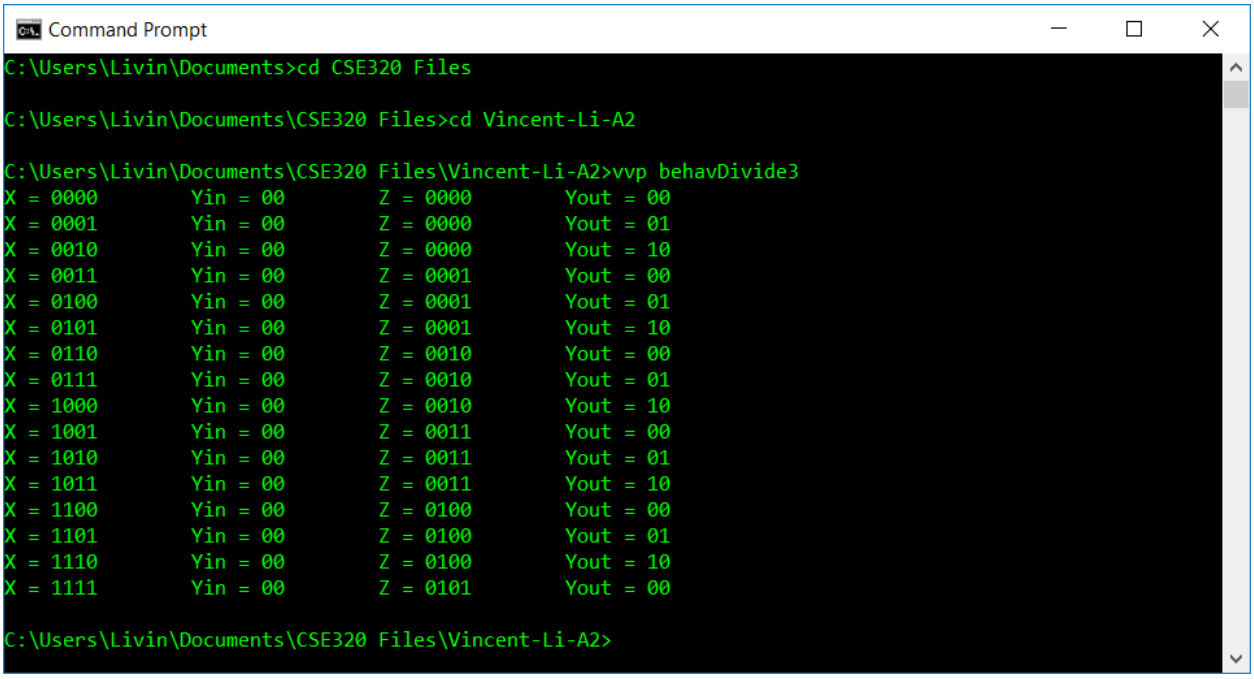
```
// Filename: Vincent-Li-A2-Q1c-behavDivide3.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Behavioral verilog program that creates a linear iterative array.
    Data flow is from left to right. It takes in an input of an
    unsigned binary number x, and outputs z, which are the bits
    of the quotient when x is divided by 3.
*/

module behavDivide3_1 (
```

```
    input wire X,
    output wire Z,
    input wire Yp1, Yp0,
    output wire Yn1, Yn0
    );
    assign Z = (Yp1 & !Yp0) | (!Yp1 & Yp0 & X);
    assign Yn1 = (!Yp1 & Yp0 & !X) | (Yp1 & !Yp0 & X);
    assign Yn0 = !Yp0 & ((!Yp1 & X) | (Yp1 & !X));

endmodule

module behavDivide3 #(parameter integer WIDTH = 4) (
    input wire [WIDTH - 1 : 0] X,
    output wire [WIDTH - 1 : 0] Z,
    input wire [1 : 0] Yin,
    output wire [1 : 0] Yout
    );

    wire [WIDTH - 1 : 0] Yp1, Yp0, Yn1, Yn0;

    behavDivide3_1 M[WIDTH - 1 : 0] (.X(X), .Z(Z), .Yp1(Yp1), .Yp0(Yp0),
.Yn1(Yn1), .Yn0(Yn0));

    assign Yp1[WIDTH - 1]  = Yin[1];
    assign Yp0[WIDTH - 1] = Yin[0];

    assign Yp1[WIDTH - 2 : 0] = Yn1[WIDTH - 1 : 1];
    assign Yp0[WIDTH - 2 : 0] = Yn0[WIDTH - 1 : 1];

    assign Yout[1] = Yn1[0];
    assign Yout[0] = Yn0[0];

endmodule
```

```
// Filename: Vincent-Li-A2-Q1c-behavDivide3_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Test bench for Vincent-Li-A2-Q1c-behavDivide3.v
*/

module behaveDivide3_tb;
    parameter integer WIDTH = 4;
    reg [WIDTH - 1 : 0] X;
```

```
    reg [1 : 0] Yin;
    wire [WIDTH - 1 : 0] Z;
    wire [1 : 0] Yout;

    behavDivide3 #(.WIDTH(WIDTH)) M(.X(X), .Yin(Yin), .Z(Z), .Yout(Yout));

    initial
        begin
            Yin = 2'b00;
            X = 0;
            repeat(WIDTH * WIDTH) begin
                //repeat(3) begin
                    #1 $display("X = %b\tYin = %b\tZ = %b\tYout = %b", X, Yin, Z,
Yout);
                    //#1 Yin = Yin + 1;
                //end

                // increment X
                X = X + 1;
                // Reset Yin
                Yin = 2'b00;
                #1;

            end

        end

endmodule
```

Results:



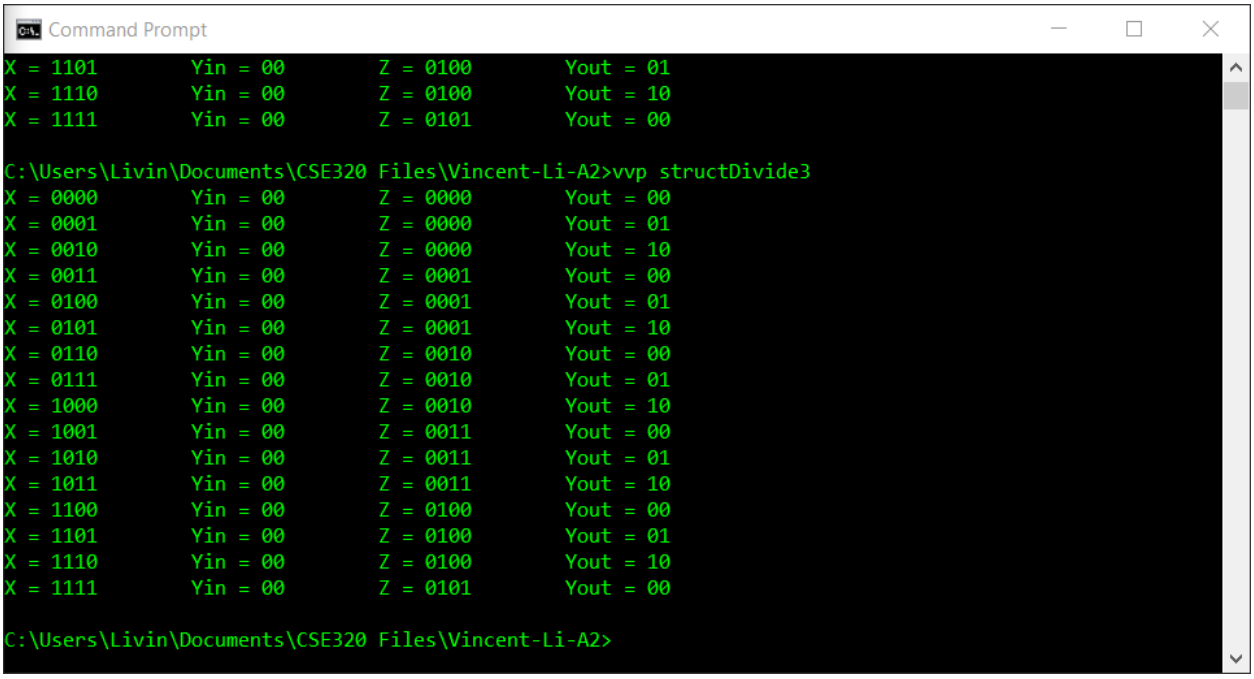

```
// Filename: Vincent-Li-A2-Q1c-structDivide3.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Structural version of Vincent-Li-A2-Q1c-behavDivide3.v
*/

module structDivide3_1 (
    input wire X,
    output wire Z,
    input wire Yp1, Yp0,
    output wire Yn1, Yn0
    );
    wire Z_bar, Yp1_bar, Yp0_bar, X_bar;
    wire w1, w2, w3, w4, w5;
    // nots
    not n0(Z_bar, Z);
    not n1(Yp1_bar, Yp1);
    not n2(Yp0_bar, Yp0);
    not n3(X_bar, X);
    // for Z
    and a0(w1, Yp1, Yp0_bar);
    and a1(w2, Yp1_bar, Yp0, X);
    or o0(Z, w1, w2);
    // for Yn1
```

```
    and a2(w3, Yp1_bar, Yp0, X_bar);
    and a3(w4, Yp1, Yp0_bar, X);
    or o1(Yn1, w3, w4);
    // for Yn0
    xor xo1(w5, Yp1, X);
    and a4(Yn0, Yp0_bar, w5);

endmodule

module structDivide3 #(parameter integer WIDTH = 4) (
    input wire [WIDTH - 1 : 0] X,
    output wire [WIDTH - 1 : 0] Z,
    input wire [1 : 0] Yin,
    output wire [1 : 0] Yout
    );

    wire [WIDTH - 1 : 0] Yp1, Yp0, Yn1, Yn0;

    structDivide3_1 M[WIDTH - 1 : 0] (.X(X), .Z(Z), .Yp1(Yp1), .Yp0(Yp0),
.Yn1(Yn1), .Yn0(Yn0));

    assign Yp1[WIDTH - 1]  = Yin[1];
    assign Yp0[WIDTH - 1] = Yin[0];

    assign Yp1[WIDTH - 2 : 0] = Yn1[WIDTH - 1 : 1];
    assign Yp0[WIDTH - 2 : 0] = Yn0[WIDTH - 1 : 1];

    assign Yout[1] = Yn1[0];
    assign Yout[0] = Yn0[0];

endmodule
```

```
// Filename: Vincent-Li-A2-Q1c-behavDivide3_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Test bench for Vincent-Li-A2-Q1c-structDivide3.v
*/

module structDivide3_tb;
    parameter integer WIDTH = 4;
    reg [WIDTH - 1 : 0] X;
    reg [1 : 0] Yin;
    wire [WIDTH - 1 : 0] Z;
```

```
    wire [1 : 0] Yout;

    structDivide3 #(.WIDTH(WIDTH)) M(.X(X), .Yin(Yin), .Z(Z), .Yout(Yout));

    initial
        begin
            Yin = 2'b00;
            X = 0;
            repeat(WIDTH * WIDTH) begin
                //repeat(3) begin
                    #1 $display("X = %b\tYin = %b\tZ = %b\tYout = %b", X, Yin, Z,
Yout);
                    //#1 Yin = Yin + 1;
                //end

                // increment X
                X = X + 1;
                // Reset Yin
                Yin = 2'b00;
                #1;

            end

        end

endmodule
```

Results:



2) (didn't complete)
State table: (same as before)

Current state	X (size 1)	Z (size 1)	Next state
00 (remainder = 0)	0	0	00 (remainder = 0)
00 (remainder = 0)	1	0	01 (remainder = 1)
01 (remainder = 1)	0	0	10 (remainder = 2)
01 (remainder = 1)	1	1	00 (remainder = 0)
10 (remainder = 2)	0	1	01 (remainder = 1)
10 (remainder = 2)	1	1	10 (remainder = 2)

Code:

```
// Filename: Vincent-Li-A2-Q1c-seqDivide3.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Moore type finite state machine that has a 1-bit primary input X
    and 1-bit primary output Z.  It can recieve an arbitrarily long
    input sequence of bits as X (MSB first).
*/

module seqDivide3_1 (
    input X,
    input reset,
    input clock,
    output reg Z
    );
```

```
    reg [1 : 0] Sp, Sn;
    // state parameters
    parameter s0 = 2'b00;
    parameter s1 = 2'b01;
    parameter s2 = 2'b10;

    // next state logic
    always @(*) begin
        if(Sp == s0 && X == 0) Sn <= s0;
        else if(Sp == s0 && X == 1) Sn <= s1;
        else if(Sp == s1 && X == 0) Sn <= s2;
        else if(Sp == s1 && X == 1) Sn <= s0;
        else if(Sp == s2 && X == 0) Sn <= s1;
        else if(Sp == s2 && X == 1) Sn <= s2;
    end

    // state update
    always @(posedge clock, posedge reset) begin
        if(reset) Sp <= s0;
        else Sp <= Sn;
    end

    // output logic
    always @(*) begin
        if(Sp == s0 && X == 0) Z = 0;
        else if(Sp == s0 && X == 1) Z = 0;
        else if(Sp == s1 && X == 0) Z = 0;
        else if(Sp == s1 && X == 1) Z = 1;
        else if(Sp == s2 && X == 0) Z = 0;
        else if(Sp == s2 && X == 1) Z = 1;
    end

endmodule

module seqDivide3 #(parameter integer WIDTH = 4) (
    input [WIDTH - 1 : 0] X,
    input reset,
    input clock,
    output reg [WIDTH - 1 : 0] Z
    );
    integer i;
    reg x;
    reg z;

    seqDivide3_1 M(.X(x), .reset(reset), .clock(clock), .Z(z));
```

```
    for(i = WIDTH - 1; i >= 0; i = i - 1) begin
        assign x = X[i];
        assign Z[i] = z;
    end

endmodule
```

```
// Filename: Vincent-Li-A2-Q1c-seqDivide3_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Moore type finite state machine that has a 1-bit primary input X
    and 1-bit primary output Z.  It can recieve an arbitrarily long
    input sequence of bits as X (MSB first).
*/

module seqDivide3_tb;
    parameter integer WIDTH = 4;
    reg [WIDTH - 1 : 0] X;
    reg clock, reset;
    reg [WIDTH -1 : 0] Z;

    integer i;

    initial
        begin
            X = 0;
            clock = 0;
            reset = 1;
            #1;
            reset = 0;
            forever
                #1 clock = ~clock;
        end

    seqDivide3 M(.X(X), .clock(clock), .reset(reset), .Z(Z));

    initial
        begin
            x = #1 0;
            repeat(WIDTH * WIDTH) begin
                #1 $display("X = %b\tZ = %b", X, Z);
                X = X + 1;
```

```
                reset = 1;
                #1;
                reset = 0;
            end
            $finish;
        end

endmodule
```