Part 1: BCD and Binary Adders

b) Behavioral Verilog Code

```verilog
`timescale 1ns / 1ps

module bin_add(A, B, S, C_out);
  input [3:0] A;
  input [3:0] B;
  output [3:0] S;
  output C_out;

  reg [3:0] S;
  reg C_out;

  always @(*)
    begin
      {C_out, S} = A + B;
    end

endmodule
```
module bcd_add(A, B, S, C_out);
    input [3:0] A;
    input [3:0] B;
    output [3:0] S;
    output C_out;

    reg [3:0] S;
    reg [3:0] temp;
    reg c_temp;
    reg chuck;

    always @(*)
    begin
        {c_temp, temp} = A + B;
        if (C_out == 1'b1)
            {chuck,S} = temp + 4'b0110;
        else
            {chuck,S} = temp + 4'b0000;
    end
endmodule

module l2p5(A, B, sel, S, C_out);
    input [3:0] A;
    input [3:0] B;
    input sel;
    output [3:0] S;
    output C_out;

    reg [3:0] S;
    reg C_out;

    wire [3:0] binsum;
    wire [3:0] bcdsum;
    wire bincar, bcdcar;

    bin_add bin1(A,B,binsum,bincar);
    bcd_add bcd1(A,B,bcdsum,bcdcar);
always @(*)
begin
  case (sel)
    1'b0: {C_out,S} = {bcdcar,bcdsum};
    1'b1: {C_out,S} = {bincar,binsum};
    default: {C_out,S} = 5'b11001;
  endcase
end
endmodule

c) Simulation results:
A = 8; b = 7; S = 0 and S = 1

![Simulation Result 1]

A = 8; b = 7; S = 0 and S = 1

![Simulation Result 2]
Part 2 State Reduction:

a)

b)

`timescale 1ns / 1ps

module l3p3(clk, rst, in, out);
    input clk;
    input rst;
input in;
output out;

reg out;

parameter A = 0;
parameter B = 1;
parameter C = 2;
parameter D = 3;

reg [1:0] state;
reg [1:0] state_nxt;

always @ (posedge clk)
begin
if (rst) state = A;
else state = state_nxt;
end

always @ (in,state)


case (state)

    A: begin
        if (in) state_nxt = C;
        else state_nxt = A;
        end

    B: begin
        if (in) state_nxt = B;
        else state_nxt = D;
        end
C: begin
if (in) state_nxt = B;
else state_nxt = D;
end

D: begin
if (in) state_nxt = C;
else state_nxt = B;
end
endcase
always @ (state)
case (state)
A: out = 0;
B: out = 1;
C: out = 0;
D: out = 0;
endcase
endmodule
Part 3:

a)

b) `timescale 1ns / 1ps

module l3p5(left_in, right_in, rst, clk, left_most, right_most);
  input left_in, right_in, rst, clk;
  output right_most, left_most;

  reg right_most, left_most;
  parameter S0 = 0;
  parameter S1 = 1;
parameter S2 = 2;
parameter S3 = 3;

parameter S4 = 4;

parameter S5 = 5;
parameter S6 = 6;
parameter S7 = 7;
parameter S8 = 8;

reg [3:0] state;
reg [3:0] state_nxt;

always @ (posedge clk)
begin
if (rst) state = S3;
else state = state_nxt;
end

always @ (posedge clk)
case (state)
  S8: begin
    if (left_in) state_nxt = S7;
    else state_nxt = S8;
    right_most = 1'b1;
    left_most = 1'b0;
  end
  S7: begin
    if (left_in) state_nxt = S6;
    else if (right_in) state_nxt = S8;
    else state_nxt = S7;
    right_most = 1'b0;
    left_most = 1'b0;
  end
  S6: begin
    if (left_in) state_nxt = S5;
    else if (right_in) state_nxt = S7;
    else state_nxt = S6;
    right_most = 1'b0;
    left_most = 1'b0;
  end
  S5: begin
        ...
  end
endcase;
if (left_in) state_nxt = S4;
else if (right_in) state_nxt = S6;
else state_nxt = S5;
right_most = 1'b0;
left_most = 1'b0;
end
S4: begin
  if (left_in) state_nxt = S3;
  else if (right_in) state_nxt = S5;
  else state_nxt = S4;
  right_most = 1'b0;
  left_most = 1'b0;
end
S3: begin
  if (left_in) state_nxt = S2;
  else if (right_in) state_nxt = S4;
  else state_nxt = S3;
  right_most = 1'b0;
  left_most = 1'b0;
end
S2: begin
  if (left_in) state_nxt = S1;
  else if (right_in) state_nxt = S3;
  else state_nxt = S2;
  right_most = 1'b0;
  left_most = 1'b0;
end
S1: begin
  if (left_in) state_nxt = S0;
  else if (right_in) state_nxt = S2;
  else state_nxt = S1;
  right_most = 1'b0;
  left_most = 1'b0;
end
S0: begin
  if (right_in) state_nxt = S1;
  else state_nxt = S0;
  right_most = 1'b0;
  left_most = 1'b1;
end
endcase
Part 4. Communicating FSMs: Gray code counter

\`
\texttt{\textcolor{gray}{\timescale 1ns / 1ps}}
\\
\begin{verbatim}
\texttt{//\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n\end{verbatim}
\`
module l3p2(up, dn, clk, rst, out);
  input up;
  input dn;
  input clk;
  input rst;
  output [2:0] out;

  reg [2:0] out;
  parameter S0 = 0;
  parameter S1 = 1;
  parameter S2 = 2;
  parameter S3 = 3;
  parameter SA = 4;

  parameter S4 = 5;
  parameter S5 = 6;
  parameter S6 = 7;
  parameter S7 = 8;
  parameter SB = 9;

  reg [3:0] state_L;
  reg [3:0] state_nxt_L;
reg [3:0] state_R;
reg [3:0] state_nxt_R;
reg [7:0] all_st;

/*
initial
begin
    state_L = S0;
    state_R = SB;
    all_st = 8'b1000_0000;
end
*/

always @(posedge clk)
begin
    if (rst)
    begin
        state_L = S0;
        state_R = SB;
        all_st = 8'b1000_0000;
    end
    else
    begin
        state_L = state_nxt_L;
        state_R = state_nxt_R;
    end
end

always @(clk or state_L)
begin
    case (state_L)
    S0: begin
        if (up == 1'b1) state_nxt_L = S1;
        else if (dn == 1'b1) state_nxt_L = SA;
        else state_nxt_L = S0;
    end
end
end
S1: begin
    if (up == 1'b1) state_nxt_L = S2;
    else if (dn == 1'b1) state_nxt_L = S0;
    else state_nxt_L = S1;
end
S2: begin
    if (up == 1'b1) state_nxt_L = S3;
    else if (dn == 1'b1) state_nxt_L = S1;
    else state_nxt_L = S2;
end
S3: begin
    if (up == 1'b1) state_nxt_L = SA;
    else if (dn == 1'b1) state_nxt_L = S2;
    else state_nxt_L = S3;
end
SA: begin
    if (up & all_st[0]) state_nxt_L = S0;
    else if (dn & all_st[3]) state_nxt_L = S3;
    else if (~((up & all_st[0])+(dn & all_st[3]))) state_nxt_L = SA;
end

endcase

always @ (clk or state_R)
case (state_R)
    S4: begin
        if (up) state_nxt_R = S5;
        else if (dn) state_nxt_R = SB;
        else state_nxt_R = S4;
    end
    S5: begin
        if (up) state_nxt_R = S6;
        else if (dn) state_nxt_R = S4;
        else state_nxt_R = S5;
    end
    S6: begin
        if (up) state_nxt_R = S7;
        else if (dn) state_nxt_R = S5;
        else state_nxt_R = S6;
    end
    S7: begin
        if (up) state_nxt_R = SB;
    end
else if (dn) state_nxt_R = S6;
else state_nxt_R = S7;
end
SB: begin
if (up & all_st[4]) state_nxt_R = S4;
else if (dn & all_st[7]) state_nxt_R = S7;
else if (~((up & all_st[4])+(dn & all_st[7]))) state_nxt_R = SB;
end
endcase

always @(state_L)
  case (state_L)
    S0: begin
      out = 3'b000;
      all_st = 8'b1000_0000;
    end
    S1: begin
      out = 3'b001;
      all_st = 8'b0100_0000;
    end
    S2: begin
      out = 3'b011;
      all_st = 8'b0010_0000;
    end
    S3: begin
      out = 3'b010;
      all_st = 8'b0001_0000;
    end
  endcase

always @(state_R)
  case (state_R)
    S4: begin
      out = 3'b110;
      all_st = 8'b0000_1000;
    end
    S5: begin
      out = 3'b111;
      all_st = 8'b0000_0100;
    end
  endcase
endcase

dendmodule

d)

<table>
<thead>
<tr>
<th>Current Simulation Time: 10000 ns</th>
<th>0</th>
<th>500</th>
<th>1000</th>
<th>1500</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
</tr>
</tbody>
</table>


e)

<table>
<thead>
<tr>
<th>Current Simulation Time: 10000 ns</th>
<th>1500</th>
<th>2000</th>
<th>2500</th>
<th>3000</th>
<th>3500</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
<td><img src="image1.png" alt="Waveform 1" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
<td><img src="image2.png" alt="Waveform 2" /></td>
</tr>
</tbody>
</table>
Part 5. Sequence Recognizer:

a)

b) `timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 02:25:36 04/27/2010
// Design Name:
// Module Name: l3p4
// Project Name:
// Target Devices:
// Tool versions:
// Description:

//
// Dependencies:

//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:

//
module l3p4(x, clk, rst, y);
  input x;
  input clk;
  input rst;
  output [1:0] y;
  reg [1:0] y;
  parameter A = 0;
  parameter B = 1;
  parameter C = 2;
  parameter D = 3;
  parameter E = 4;

  reg [2:0] state;
  reg [2:0] state_nxt;

  always @ (posedge clk)
  begin
    if (rst) state = A;
    else state = state_nxt;
  end

  always @ (posedge clk)
  case (state)
    A: if (x) begin
      state_nxt = B;
      y = 2'b00;
    end else begin
      state_nxt = A;
      y = 2'b00;
    end
    B: if (x) begin
      state_nxt = B;
      y = 2'b00;
    end else begin
      state_nxt = C;
      y = 2'b00;
    end
    C: if (x) begin
      state_nxt = E;
    end
  endcase
endmodule
y = 2'b00;
end
else begin
  state_nxt = D;
y = 2'b00;
end

D: if (x) begin
  state_nxt = B;
y = 2'b01;
end
else begin
  state_nxt = A;
y = 2'b00;
end

E: if (x) begin
  state_nxt = B;
y = 2'b11;
end
else begin
  state_nxt = C;
y = 2'b10;
end

// default: y = 2'b00;
endcase
defaultcase
endmodule
c)