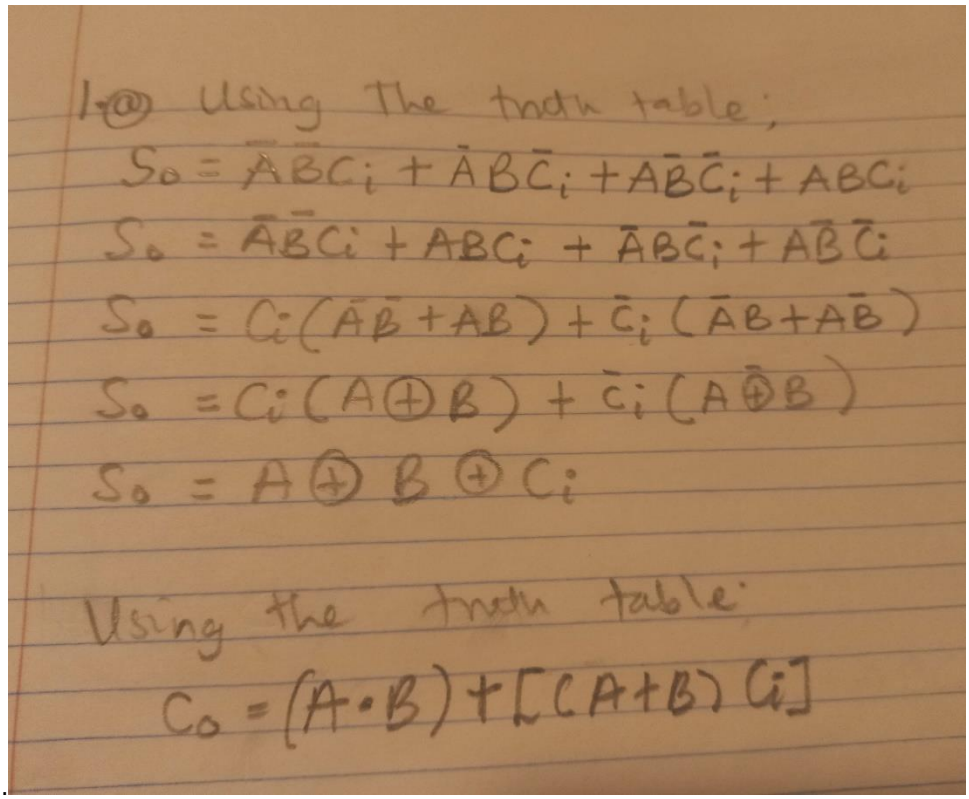
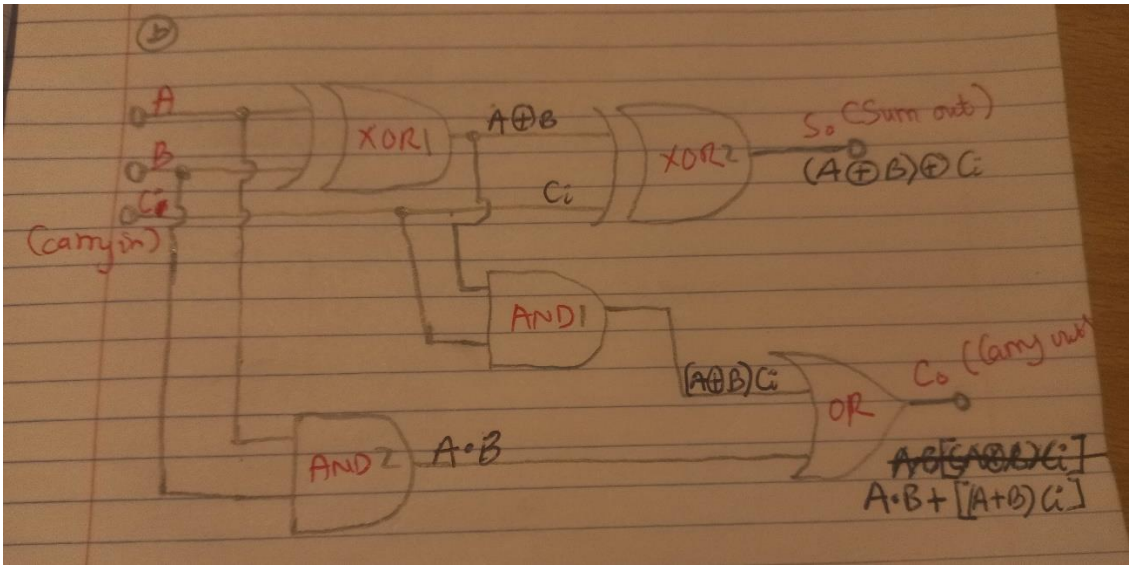
PROBLEM 1 - FULL ADDER

A.



.

B.



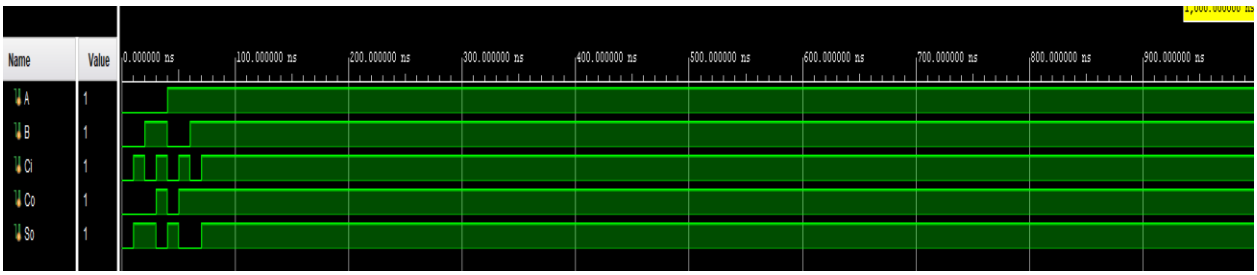
C.

```
`timescale 1ps/1fs


module XORX1 (A, B, C);
  input A, B;
  output C;
  assign C = A ^ B ;
endmodule

module ANDX1 (A, B, C);
  input A, B;
  output C;
  assign C = A & B ;
endmodule

module ORX1 (A, B, C);
  input A, B;
  output C;
  assign C = A | B ;
endmodule

module Cout (A, B, Ci,Co);
  input A, B, Ci;
  output Co;
  logic x1, x2 , x3, x4 ;
  ANDX1 and1 (A, B, x1);
  ANDX1 and2 (A, Ci, x2);
  ANDX1 and3 (B, Ci, x3);
  ORX1 or1 (x1, x2, x4);
  ORX1 or2 (x4, x3, Co);
endmodule

module Sout (A, B, Ci, So);
  input A, B,Ci;
  output So;
  logic x1;
  XORX1 xor1 (A, B, x1 );
  XORX1 xor2 (Ci, x1, So);

endmodule

module fulladder (A, B, Ci, Co, So);
    input  A, B, Ci;
```

```
    output  Co, So;

    Cout CO (A, B, Ci, Co);
    Sout SO ( A, B, Ci, So);

    specify
    (A, B, Ci => Co) = (500, 781);
    (A, B, Ci => So) = (245, 237);
    endspecify

endmodule
```

D. TESTBENCH WITH PLOT SHOWING ALL POSSIBLE SCENARIOS(2^3 =8)

```
`timescale 1ns/1ps
module fulladder_tb();
logic A,B,Ci,Co,So;

fulladder DUT (A,B,Ci,Co,So);

initial begin
    A = 0; B = 0 ; Ci = 0; # 10;
    A = 0; B = 0 ; Ci = 1; # 10;
    A = 0; B = 1 ; Ci = 0; # 10;
    A = 0; B = 1 ; Ci = 1; # 10;
    A = 1; B = 0 ; Ci = 0; # 10;
    A = 1; B = 0 ; Ci = 1; # 10;
    A = 1; B = 1 ; Ci = 0; # 10;
    A = 1; B = 1 ; Ci = 1; # 10;
end
endmodule
```



PROBLEM 2 - STATE MACHINE

A. The FSM is a Mealy Machine. The output of each state is dependent on the input(w) of the FSM and the current state.

B.

```
`timescale 1ns/1ps

module MealyMachine (
    input logic clock,
    input logic reset,
    input logic w,
    output logic out
);
    typedef enum logic [2:0] { A, B, C, D, E } state;
    state c_state , n_state ;
    always_ff @( c_state, w ) begin
        case (c_state)
          A  : if (w) n_state = B;
               else n_state = A ;
          B  : if (w) n_state = C;
               else n_state = A ;
          C  : if (w) n_state = D;
               else n_state = A;
          D  : if (w) n_state = E;
               else n_state = A;
          E  : if (w) n_state = E;
               else n_state = A;
            default: n_state = A;
        endcase
    end
    always_ff @( posedge clock ) begin
        if (reset) c_state = A ;
        else c_state = n_state ;
    end
    assign out = (c_state == A | c_state == E);
endmodule
```

C.

```
`timescale 1ns/1ps
module MealyMachine_tb ();

logic w, clock, reset, out;

MealyMachine DUT (
   .w(w),
   .clock(clock),
   .reset(reset),
   .out(out)
);

initial
 clock = 1'b0;
always #5
 clock = ~clock;

initial begin
    w = 1'b0 ; #20;
    reset = 1'b1 ;#20;
    reset = 1'b0 ;#40;
    w = 1'b1 ;#30;
    w = 1'b1 ;#10;
    w = 1'b0 ;#30;
    w = 1'b1 ;#40;
    w = 1'b1 ;#25;
  $finish();
end
endmodule
```