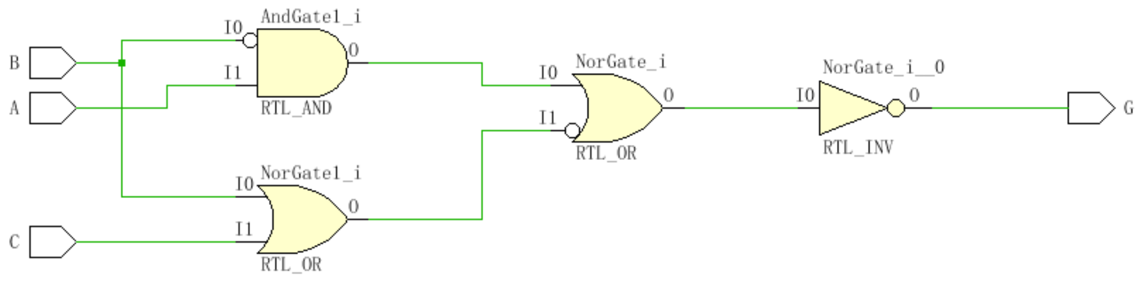
HOMEWORK 1

Cunyang Liu U201811446

PROBLEM 1

Figure



PROBLEM 2

Code

- Problem2.v

```
`timescale 1ns / 1ps
//Problem 2 Homework #1 summer 2021 HUST
module p2hw1summer2021HUST(input I1,I2,I3,I4,output Straight,Tum)
    wire NotOut,OUT1,OUT3,OUT4;
    not U1(NotOut,I2);
    nand U2(OUT1,NotOut,I1);
    nor U3(OUT3,I3,I4);
    nand U4(OUT4,I3,I4);
    and U5(Straight,OUT1,OUT3);
    not U6(Tum,OUT4);
endmodule
```

PROBLEM 3

Code

- MultiplierBehavior.v

```
module MultiplierBehavior(
    input [1:0] A,B,
    output reg [2:0] Product
);
    always @(A,B) begin
        Product = A * B;
    end
endmodule
```

- MultiplierExpression.v

```
module MultiplierExpression(
    input [1:0] A,B,
    output reg [2:0] Product
);
    assign Product[2] = (A[1]&(!A[0])&B[1]) | (A[1]&B[1]&(!B[0]));
    assign Product[1] = ((!A[1]&A[0]&B[1]) | (A[0]&B[1]&(!B[0])) | (A[1]&
(!A[0])&B[0]) | (A[1]&(!B[1])&B[0]));
    assign Product[0] = A[0] & B[0];
endmodule
```

- MultiplierGates.v

```
module MultiplierGates(
    input [1:0] A,B,
    output reg [2:0] Product
);
    or P2(Product[2], (A[1]&(!A[0])&B[1]), (A[1]&B[1]&(!B[0])));
    or P1(Product[1], (!A[1]&A[0]&B[1]), (A[0]&B[1]&(!B[0])), (A[1]&
(!A[0])&B[0]), (A[1]&(!B[1])&B[0]));
    and P0(Product[0], A[0], B[0]);
endmodule
```

- MultiplierTruthTable.v

```
module MultiplierTruthTable(
    input [1:0] A,B,
    output reg [2:0] Product
);
    always @(A,B) begin
        case ({A,B})
            4'b0000: Product = 3'b000;
            4'b0001: Product = 3'b000;
            4'b0010: Product = 3'b000;
            4'b0011: Product = 3'b000;
            4'b0100: Product = 3'b000;
            4'b0101: Product = 3'b001;
            4'b0110: Product = 3'b010;
            4'b0111: Product = 3'b011;
            4'b1000: Product = 3'b000;
            4'b1001: Product = 3'b010;
            4'b1010: Product = 3'b100;
            4'b1011: Product = 3'b110;
            4'b1100: Product = 3'b000;
            4'b1101: Product = 3'b011;
            4'b1110: Product = 3'b110;
            4'b1111: Product = 3'b001;
        endcase
    end
endmodule
```

- p3hw1TestFixture.v

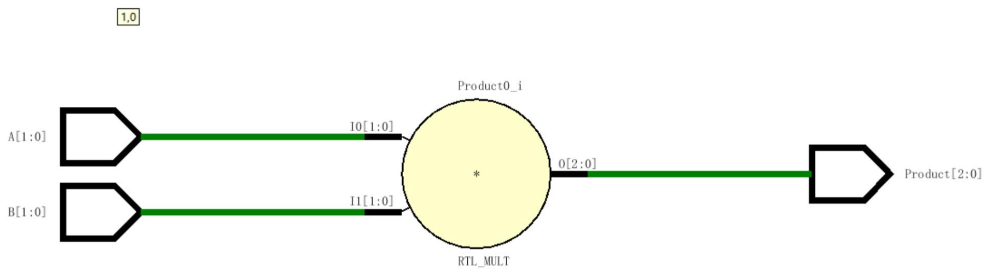
```
`timescale 1ns / 1ps
```

```
//Multiplier test bench
module p3hw1TestFixture;
reg A[1:0], B[1:0];
wire ProductBgates[2:0], ProductExpressions [2:0], ProductTruthTable[2:0], ProductBbehavior[2:0];
//insert four circuits
initial begin
A[0]=0;A[1]=0; B[0]=0;B[1]=0; #10;
A[0]=0;A[1]=0; B[0]=1;B[1]=0; #10;
A[0]=0;A[1]=0; B[0]=0;B[1]=1; #10;
A[0]=0;A[1]=0; B[0]=1;B[1]=1; #10;

A[0]=1;A[1]=0; B[0]=0;B[1]=0; #10;
A[0]=1;A[1]=0; B[0]=1;B[1]=0; #10;
A[0]=1;A[1]=0; B[0]=0;B[1]=1; #10;
A[0]=1;A[1]=0; B[0]=1;B[1]=1; #10;

A[0]=1;A[1]=1; B[0]=0;B[1]=0; #10;
A[0]=1;A[1]=1; B[0]=1;B[1]=0; #10;
A[0]=1;A[1]=1; B[0]=0;B[1]=1; #10;
A[0]=1;A[1]=1; B[0]=1;B[1]=1; #10;

A[0]=0;A[1]=1; B[0]=0;B[1]=0; #10;
A[0]=0;A[1]=1; B[0]=1;B[1]=0; #10;
A[0]=0;A[1]=1; B[0]=0;B[1]=1; #10;
A[0]=0;A[1]=1; B[0]=1;B[1]=1; #10;
end

MultiplierBehavior Unit1({A[1],A[0]},{B[1],B[0]},{ProductBbehavior[2],ProductBbehavior[1],ProductBbehavior[0]});
MultiplierExpression Unit2({A[1],A[0]},{B[1],B[0]},{ProductExpressions[2],ProductExpressions[1],ProductExpressions[0]});
MultiplierGates Unit3({A[1],A[0]},{B[1],B[0]},{ProductBgates[2],ProductBgates[1],ProductBgates[0]});
MultiplierTruthTable Unit4({A[1],A[0]},{B[1],B[0]},{ProductTruthTable[2],ProductTruthTable[1],ProductTruthTable[0]});

//generate test patterns
endmodule
```

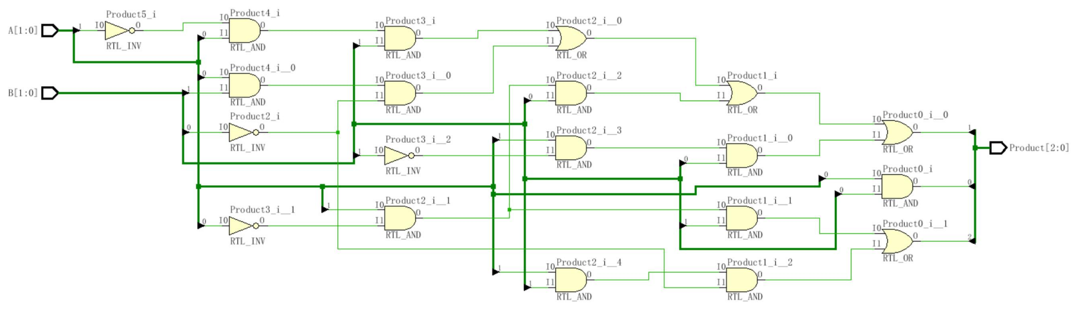
Figure

Schematic

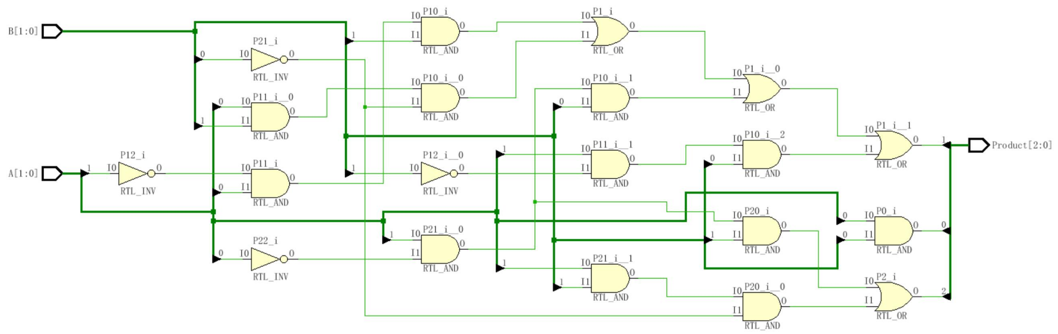
- MultiplierBehavior



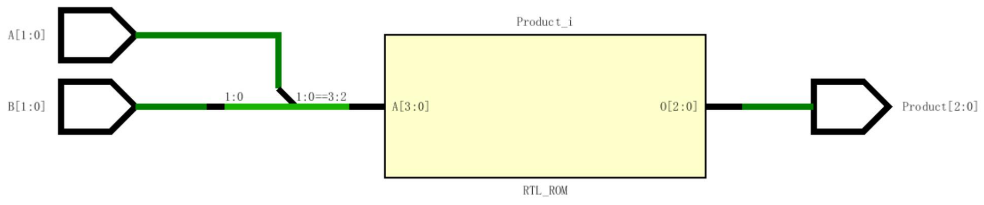
- MultiplierExpression



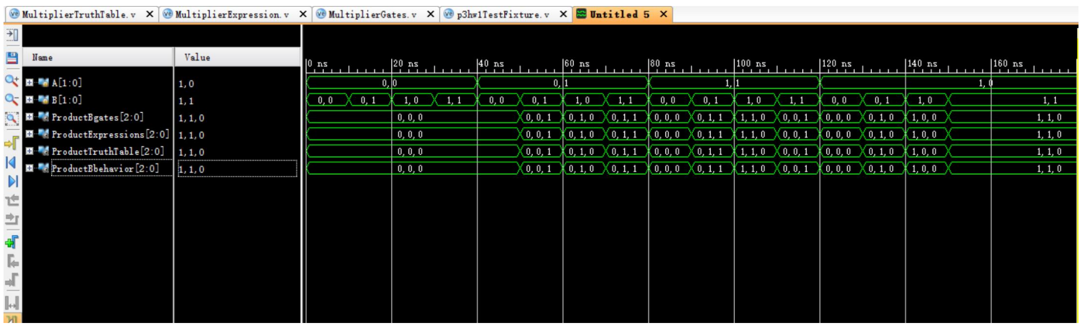
- MultiplierGates



- MultiplierTruthTable



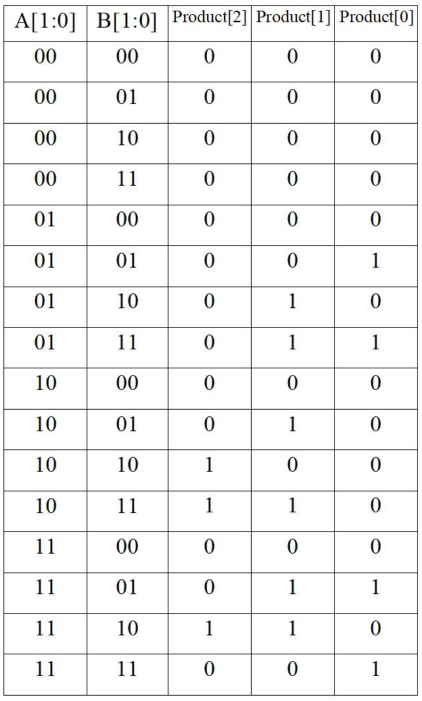
- Waveforms



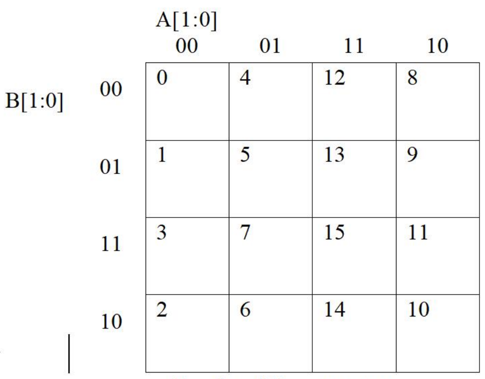
```
Name                      Value
A[1:0]                    1,0
B[1:0]                    1,1
ProductBgates[2:0]        1,1,0
ProductExpressions[2:0]   1,1,0
ProductTruthTable[2:0]    1,1,0
ProductBbehavior[2:0]     1,1,0
```

```
A[1:0]                   0,0 | 0,1 | 1,1 | 1,0
B[1:0]                   0,0 | 0,1 | 1,0 | 1,1 | 0,0 | 0,1 | 1,0 | 1,1 | 0,0 | 0,1 | 1,0 | 1,1 | 0,0 | 0,1 | 1,0 | 1,1
ProductBgates[2:0]       0,0,0 | 0,0,1 | 0,1,0 | 0,1,1 | 0,0,0 | 0,1,1 | 1,1,0 | 0,0,1 | 0,0,0 | 0,1,0 | 1,0,0 | 1,1,0
ProductExpressions[2:0]  0,0,0 | 0,0,1 | 0,1,0 | 0,1,1 | 0,0,0 | 0,1,1 | 1,1,0 | 0,0,1 | 0,0,0 | 0,1,0 | 1,0,0 | 1,1,0
ProductTruthTable[2:0]   0,0,0 | 0,0,1 | 0,1,0 | 0,1,1 | 0,0,0 | 0,1,1 | 1,1,0 | 0,0,1 | 0,0,0 | 0,1,0 | 1,0,0 | 1,1,0
ProductBbehavior[2:0]    0,0,0 | 0,0,1 | 0,1,0 | 0,1,1 | 0,0,0 | 0,1,1 | 1,1,0 | 0,0,1 | 0,0,0 | 0,1,0 | 1,0,0 | 1,1,0
```

- Table and Logic Expression

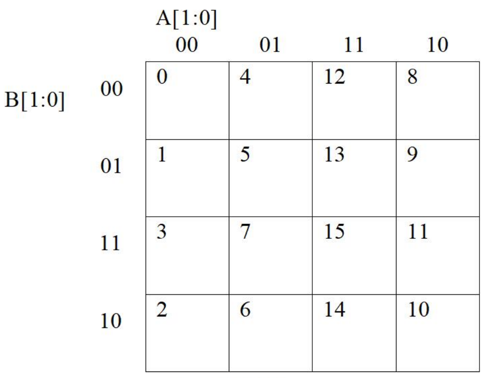


A[1:0]	B[1:0]	Product[2]	Product[1]	Product[0]
00	00	0	0	0
00	01	0	0	0
00	10	0	0	0
00	11	0	0	0
01	00	0	0	0
01	01	0	0	1
01	10	0	1	0
01	11	0	1	1
10	00	0	0	0
10	01	0	1	0
10	10	1	0	0
10	11	1	1	0
11	00	0	0	0
11	01	0	1	1
11	10	1	1	0
11	11	0	0	1



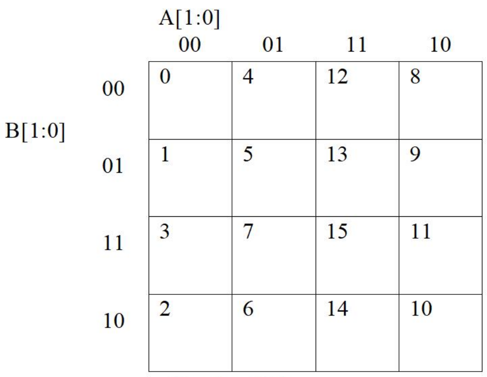
B[1:0] \ A[1:0]	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Product[0]= $A[0]B[0]$



B[1:0] \ A[1:0]	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Product[2]=
$A[1]\overline{A[0]}B[1]+A[1]B[1]\overline{B[0]}$



B[1:0] \ A[1:0]	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Product[1]=
$\overline{A[1]}A[0]B[1]+A[0]B[1]\overline{B[0]}+A[1]\overline{A[0]}B[0]+A[1]\overline{B[1]}B[0]$

PROBLEM 4

4.(a)

- B;Active low

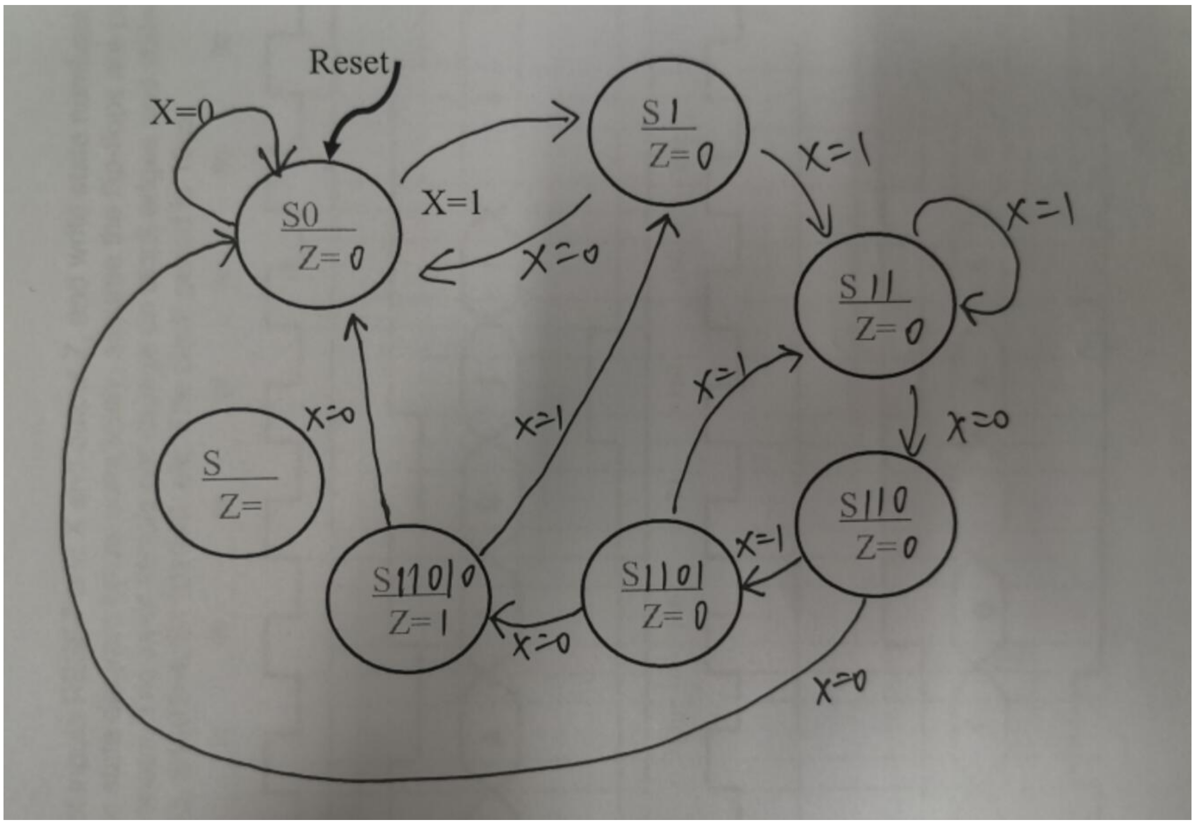
4.(b)

- A;Active high;Asynchronous

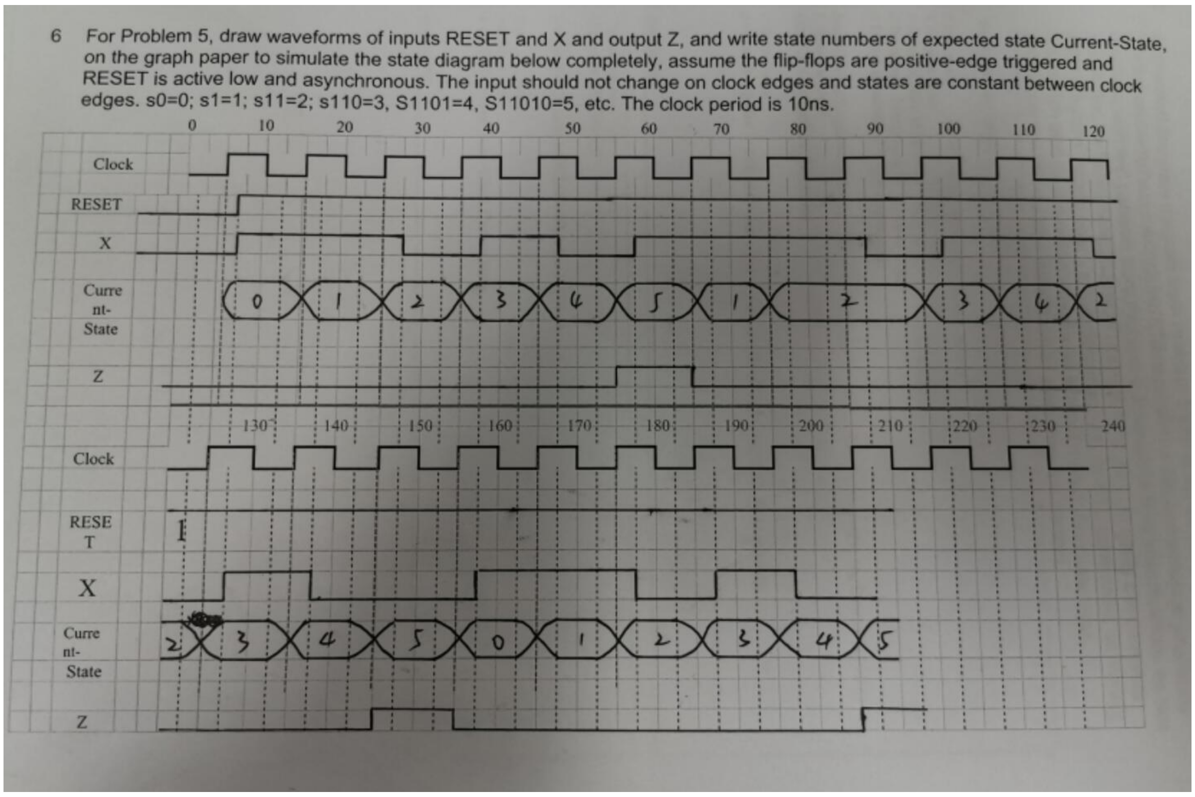
4.(c)

- C;Active low;Synchronous

PROBLEM 5



PROBLEM 6



6 For Problem 5, draw waveforms of inputs RESET and X and output Z, and write state numbers of expected state Current-State, on the graph paper to simulate the state diagram below completely, assume the flip-flops are positive-edge triggered and RESET is active low and asynchronous. The input should not change on clock edges and states are constant between clock edges. s0=0; s1=1; s11=2; s110=3, S1101=4, S11010=5, etc. The clock period is 10ns.

PROBLEM 7

Code

- hw1p7summer2020HUSTdetect11010.v

```
`timescale 1ns / 100ps
// File name : hw1p7summer2020HUSTdetect11010.v
// Cunyang Liu
```

```verilog
// Summer 2021 HUST
// Problem 7, Homework #1, summer 2021
// Detect sequence of 11010 recursively.
module hw1p5summer20201HUSTdetect11010(input InputBit, CLK, Reset, output reg Detected11010);
// State variables
reg [2:0] CurrentState, NextState;
// State codes
parameter SInitial = 3'd0,S1 = 3'd1, S11 = 3'd2, S110= 3'd3, S1101=3'd4, S11010=3'd5;

always @ (posedge CLK or negedge Reset)
begin
if (!Reset)
  CurrentState <= SInitial;
else
  CurrentState <= NextState;
end

always @ (*) begin
case (CurrentState)
  SInitial:
    if (InputBit == 1)
        NextState <= S1;
    else
        NextState <= SInitial;
  S1:
    if (InputBit == 1)
        NextState <= S11;
    else
        NextState <= SInitial;
  S11:
    if (InputBit == 1)
        NextState <= S11;
    else
        NextState <= S110;
  S110:
    if (InputBit == 1)
        NextState <= S1101;
    else
        NextState <= SInitial;
  S1101:
    if (InputBit == 1)
        NextState <= S11;
    else
        NextState <= S11010;
  S11010:
    if (InputBit == 1)
        NextState <= S1;
    else
        NextState <= SInitial;
  default:
        NextState <= SInitial;
endcase
end

//the output depends on the current state
always @ (*) begin
```

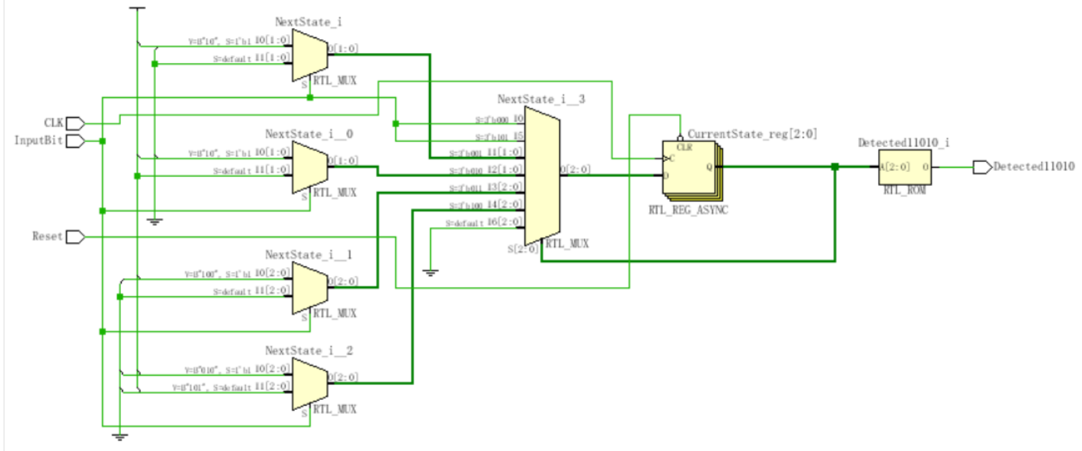
```
    if (CurrentState == S11010)
        Detected11010 <= 1;
    else
        Detected11010 <= 0;
end
endmodule
```

- hw1p7summer2021HUSTdetect11010_tb.v

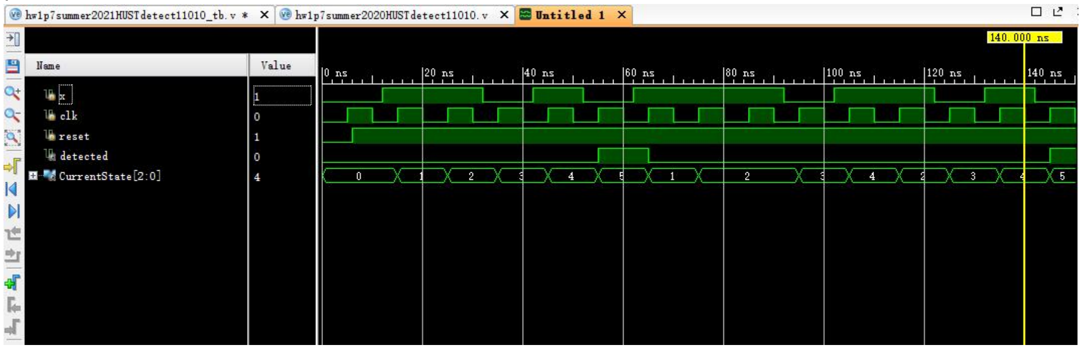
```
`timescale 1ns / 1ps
//Summer 2021 HUST
module hw1p7summer2021HUSTdetect11010_tb;
reg x, clk, reset;
wire detected;
wire [2:0] CurrentState=Unit1.CurrentState;

always #5 clk = ~clk;
    initial begin
        clk = 0;
        reset = 0;
        x = 0;
        #8 reset=1;
        #8 x=1;
        #15 x = 1;
        #15 x = 0;
        #15 x = 1;
        #15 x = 0;
        #15 x = 1;
        #15 x = 1;
        #15 x = 1;
        #15 x = 0;
        #15 x = 1;
        #15 x = 1;
        #15 x = 0;
        #15 x = 1;
        #15 x = 0;
        #15 x = 0;
        #15 x = 1;
        #15 x = 1;
        #15 x = 0;
        #15 x = 1;
        #15 x = 0;
    end
    hw1p5summer20201HUSTdetect11010 Unit1(x, clk, reset,detected);
endmodule
```

Figure

- problem7_RTL_schematic



- problem7_waveform



PROBLEM 8

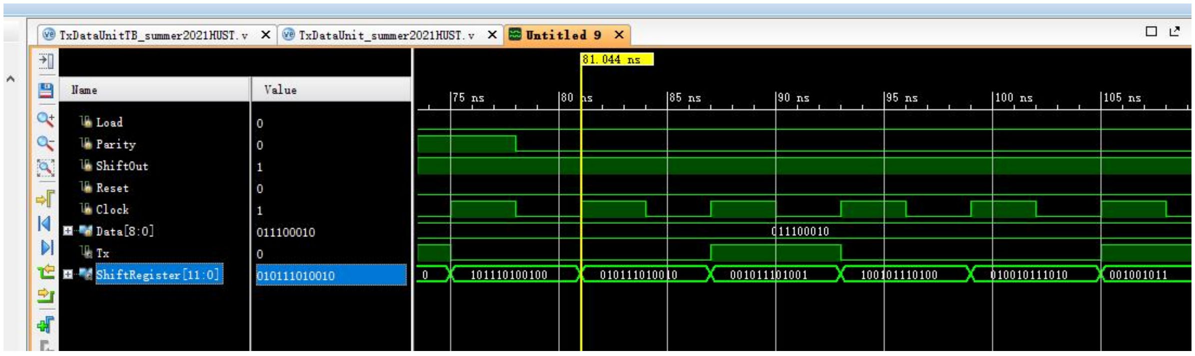
- TxDataUnit_summer2021HUST.v

```verilog
`timescale 1ns / 1ps
module TxDataUnit_summer2021HUST #(parameter DataLength=9)(
    input [DataLength-1:0] Data,
    input Load, ShiftOut, Parity, Reset, Clock,
    output Tx);
    reg [11:0] ShiftRegister;
    wire ParityBit;
    assign Tx=ShiftRegister[0];
    assign ParityBit=Parity^Data[0]^Data[1]^Data[2]^Data[3]^Data[4]^Data[5]^Data[6]^Data[7]^Data[8];
    always@(negedge Clock)
        if(Reset==1)
            ShiftRegister<={ParityBit, Data, 2'b01};
    else if(Load == 1)
            ShiftRegister <= {1'b1,ParityBit,Data,1'b0};
    else if(ShiftOut==1)
            ShiftRegister<={ShiftRegister[0], ShiftRegister[11:1]};
    else ShiftRegister<=ShiftRegister;
endmodule
```

- TxDataUnitTB_summer2021HUST.v

```verilog
`timescale 1ns / 1ps
module TxDataUnitTB_summer2021HUST;
    reg Load, Parity, ShiftOut, Reset, Clock;
    reg [8:0] Data;
    wire Tx;
    wire [11:0] ShiftRegister=uut.ShiftRegister;

    TxDataUnit_summer2021HUST uut (.Load(Load), .Data(Data), .Parity(Parity),.Tx(Tx),
 .ShiftOut(ShiftOut), .Reset(Reset), .Clock(Clock));

    initial begin
        Load = 0; Data = 0; Parity = 0; ShiftOut = 0; Reset = 0; Clock = 0;
    end
    always #3 Clock=~Clock;
    initial fork
        #0 Load = 0; #21 Load = 1; #32 Load = 0; #56 Load = 0; #152 Load = 1;
        #165 Load = 0;
        #0 Data = 8'b010010111; #56 Data = 8'b011100010;
        #0 Parity = 1; #34 Parity = 1; #78 Parity = 0; #134 Parity = 1;
        #0 ShiftOut = 0; #38 ShiftOut = 1; #148 ShiftOut = 0; #167 ShiftOut = 1;
        #284 ShiftOut = 0;
        #0 Reset = 1; #12 Reset = 0;
        #300 $stop;
    join
endmodule
```

- Simulation Waveforms



```
Name                 Value
Load                 0
Parity               0
ShiftOut             1
Reset                0
Clock                1
Data[8:0]            011100010
Tx                   0
ShiftRegister[11:0]  010111010010
```

- TxModule_Toplevel_summer2021HUST.v

```verilog
`timescale 1ns / 1ps
module TxModule_Toplevel_summer2021HUST(input Start, Parity, Reset, Clock,
                                        input [8:0] Data,
                                        input [3:0] Speed, // baud in the number of clock cycles
                                        output Tx);
    wire Load, ShiftOut;
    TxController_summer2021HUST ControlUnit(Start, Reset, Clock, Speed, Load, ShiftOut);
    TxDataUnit_summer2021HUST DataUnit(Data, Load, ShiftOut, Parity, Reset, Clock, Tx);

endmodule
```

- TxController_summer2021HUST.v

```verilog
`timescale 1ns / 1ps
```

```verilog
module TxController_summer2021HUST(
    input Start, Reset, Clock,
    input [3:0] Speed,
    output reg Load, reg ShiftOut
);
    // State variables
    reg [2:0] CurrentState;

    // Counter
    reg StartDelay;
    reg [3:0] DataCounter;
    wire delay_timeout;

    // State codes
    parameter InitialState = 3'd0, LoadState = 3'd1, DelayState = 3'd2, ShiftState= 3'd3;

    // module DelayTime(Start, Speed, Timeout,Reset, Clock);
    DelayTime_summer2021HUST DelayUnit(StartDelay, Speed, delay_timeout, Reset, Clock);

    initial begin
        CurrentState = InitialState;
        StartDelay = 0;
        DataCounter = 0;
    end

    always @(posedge Clock or posedge Reset) begin
        if (Reset) begin
            // reset
            CurrentState <= InitialState;
        end
        else begin
            case (CurrentState)
                InitialState:begin
                    CurrentState = (Start) ? LoadState : InitialState;
                end
                LoadState:begin
                    CurrentState = DelayState;
                end
                DelayState:begin
                    CurrentState = (delay_timeout) ? ShiftState : DelayState;
                end
                ShiftState:begin
                    CurrentState = (DataCounter < 12) ? DelayState : InitialState;
                end
            endcase
        end
    end

    always @(posedge Clock) begin
        case (CurrentState)
            InitialState:begin
                StartDelay <= 0;
                DataCounter <= 0;
                ShiftOut <= 0;
```

```
                    Load <= 0;
                    StartDelay = 0;
                end
                LoadState:begin
                    DataCounter <= 1;
                    Load <= 1;
                    ShiftOut <= 0;
                    StartDelay <= 0;
                end
                DelayState:begin
                    DataCounter <= DataCounter;
                    Load <= 0;
                    ShiftOut <= 0;
                    StartDelay <= 1;
                end
                ShiftState:begin
                    DataCounter <= DataCounter + 1;
                    Load <= 0;
                    ShiftOut <= 1;
                    StartDelay <= 0;
                end
            endcase
        end
endmodule
```

- DelayTime_summer2021HUST.v

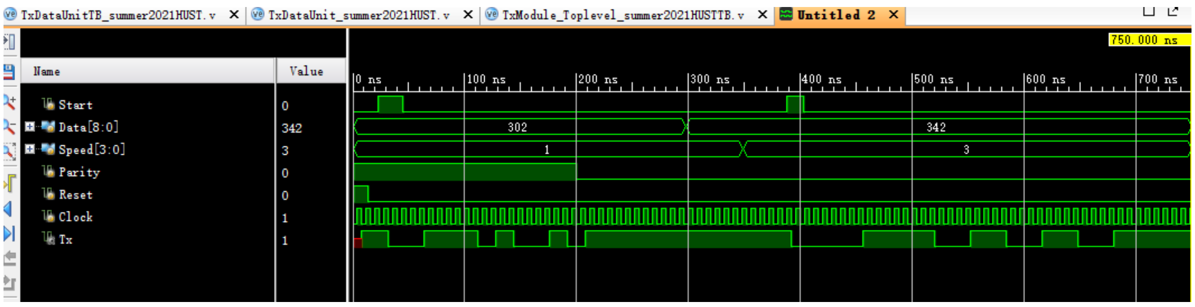
```
`timescale 1ns / 1ps
module DelayTime_summer2021HUST(Start, Speed, Timeout,Reset, Clock);
    //delay time in number of clock cycles as speficied by Speed
    parameter   NumberOfBits = 4;
    input       Start, Reset, Clock;
    input       [NumberOfBits-1:0] Speed;
    output reg  Timeout;
    reg         [NumberOfBits-1:0] count;

    always @ (count or Speed)
        if (count == (Speed-1'b1))
            Timeout <= 1'b1;
        else
            Timeout <= 0;

    always @ (posedge Clock)
        if(Reset == 1)
            count <= 4'd0;
        else if(Start == 0)
            count <= 4'd0;
        else if (count >= (Speed-1'b1))
            count <= 4'd0;
        else
            count <= count + 1'b1;
endmodule
```

- TxModule_Toplevel_summer2021HUSTTB.v

```
`timescale 1ns / 1ps
```

```verilog

module TxModule_Toplevel_summer2021HUSTTB;
// Inputs
    reg Start;
    reg [8:0] Data;
    reg [3:0] Speed;
    reg Parity, Reset, Clock;
    wire Tx;

    TxModule_Toplevel_summer2021HUST TopLevel (Start, Parity, Reset, Clock,
Data, Speed, Tx);
    initial begin
        Start = 0; Data = 0; Speed = 0; Parity = 0; Reset = 0; Clock = 0;
    end
    always
        #4 Clock=~Clock;
    initial fork
        #0 Reset = 1;
        #0 Start = 0;
        #0 Data = 9'b100101110;
        #0 Speed = 1;
        #0 Parity = 1;
        #14 Reset = 0;
        #23 Start = 1;
        #45 Start = 0;
        #200 Parity = 0;
        #298 Data = 9'b101010110;
        #349 Speed = 3;
        #388 Start = 1;
        #403 Start = 0;
        #750 $stop;
    join
endmodule
```

- waveform


TxDataUnitTB_summer2021HUST.v
TxDataUnit_summer2021HUST.v
TxModule_Toplevel_summer2021HUSTTB.v
Untitled 2
750.000 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
302
342
1
3

Name	Value
Start	0
Data[8:0]	342
Speed[3:0]	3
Parity	0
Reset	0
Clock	1
Tx	1