Name ____________________ ID ________ Due date: Thursday, Aug. 12, 2021

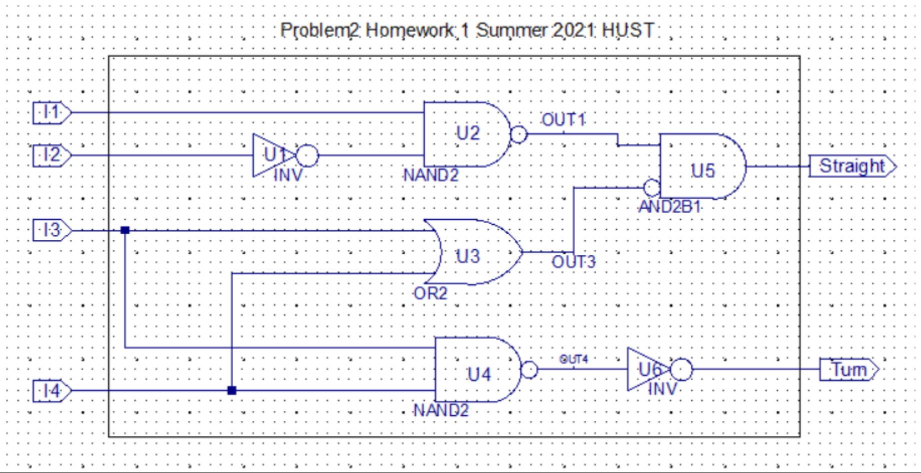
S&H Co-Design Homework #1 Handout Summer 2021

(Combinational and Sequential Circuits)

1 Draw the circuit schematic for the following Verilog gate-level circuit module named Problem1.

```
//Problem 1 HW1 Summer 2021 HUST
module p1hw1summer2020(input A,B,C, output G);

wire NotOut, NorOut, AndOut;
              not NotGate1(NotOut,B);
   and AndGate1(AndOut, NotOut, A);
              nor NorGate1(NorOut, B,C);
              nor NorGate(G, AndOut, NorOut);
endmodule
```

2 Translate the following schematic into a gate-level text module in Verilog called p2hw1summer2021HUST.v.



```
`timescale 1ns / 1ps
//Problem 2 Homework #1 summer 2021 HUST
module p2hw1summer2021HUST(





endmodule
```

3 Design a 2-bit multiplier that has two 2-bit numbers as inputs to obtain a 3-bit product. Create a single Vivado project to complete the following assignments. All implementation files should be in the same project folder and will be simulated with one test bench file. Attach all your design files in Verilog or their RTL schematics. These files can be submitted as pdf copies.

3.(a) *Write simplified sum-of-product (SOP) logic expressions for the circuit. Use the K-maps on next page to help you derive the expressions.*

3.(b) *Create a Vivado project and write a Verilog gate-level module based on the two-level schematic, entitled MultiplierGates.v. Submit the RTL schematic generated from the Vivado project. The external behavior of the circuit should have two inputs A[1:0], B[1:0], and three outputs, Product[2:0], i.e., Product=A*B.*

3.(c) *Write a Verilog logic-expression-level module based on the SOP logic expressions, entitled MultiplierExpressions.v. Submit the RTL schematic generated. The external behavior of the circuit should be the same as in 3.(b).*

3.(d) *Write a Verilog module based on the truth table of the problem, entitled MultiplierTruthTable.v. Submit the Technology schematic generated. The external behavior of the circuit should be the same as in 3.(b).*

3.(e) *Write a Verilog module based on behavior of the problem, entitled MultiplierBehavior.v. Submit the Technology schematic generated. The external behavior of the circuit should be the same as in 3.(b).*

3.(f) *Simulate the four circuits with one test bench that has the following I/O definition. Combine 2-bit input A[1:0], and 2-bit B[1:0] to form virtual buses and display the bus value in decimal. Align the outputs so that the related outputs are adjacent to each other. Submit the annotated waveforms.*

```
//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
//generate test patterns
Endmodule
```

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

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

Product[0]=

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

Product[2]=

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

Product[1]=

4 Answer the following questions about module p4hw1summer2021() by circling correct terms. Reset is to make state to be State0 and set to be State3.

4.(a) *Which wire is the clock signal? Is the clock signal active on positive or negative edge?*

o A	o B	o C
o D	o E	o Active low
o Active high	o Asynchronous	o Synchronous

4.(b) *Which wire is the reset wire? Is the reset signal active high or active low, synchronous or synchronous?*

o A	o B	o C
o D	o E	o Active low
o Active high	o Asynchronous	o Synchronous

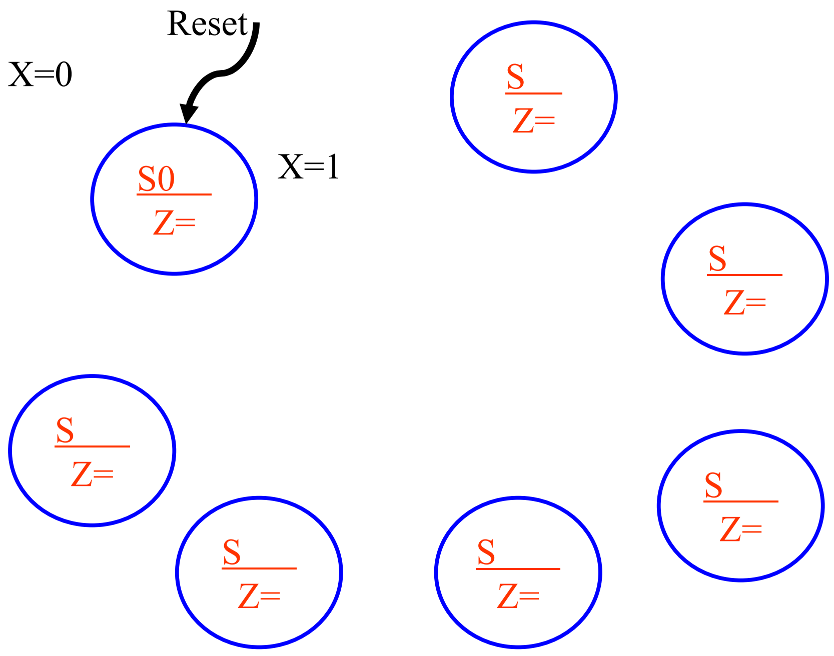
4.(c) *Which wire is the set wire? Is the set signal active high or active low, synchronous or synchronous?*

o A	o B	o C
o D	o E	o Active low
o Active high	o Asynchronous	o Synchronous

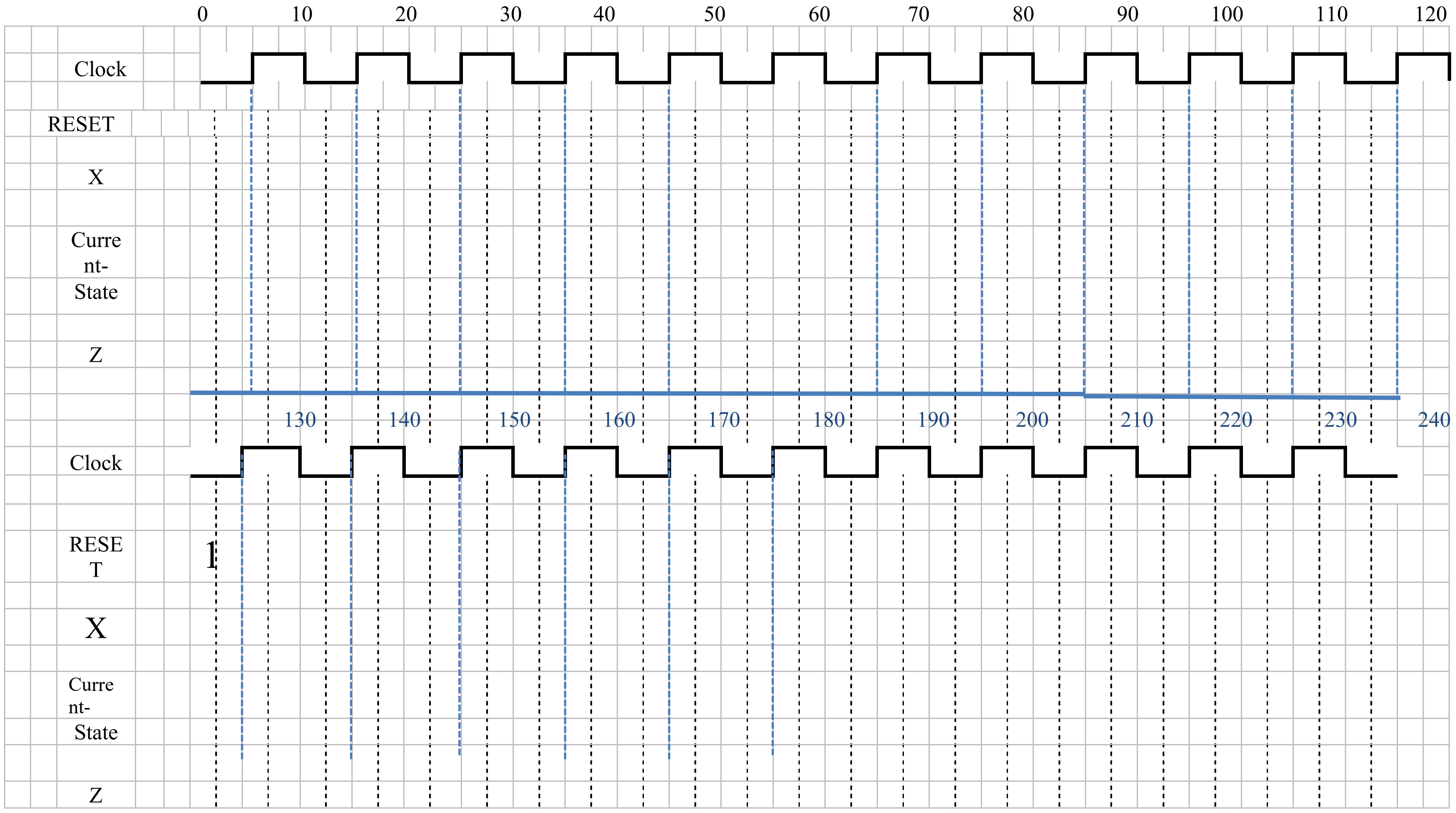
```
//Set is to make CurrentState State0
//Reset is to make CurrentState State3
//Jianjian Song
module  p4hw1summer2021(input A, B,C , D, output reg E);
reg [1:0] NextState, CurrentState;
parameter  State0 = 2'b00, State3=2'd3, State1 = 2'd1, State2 = 2'd2;
always @ (CurrentState)
            if (CurrentState == State1) E <= 1;   else E <= 0;
always @ (posedge A or negedge B)
            begin  if (A==1)   CurrentState <= State0;    else if (C==0)  CurrentState <= State3;
            else       CurrentState<=NextState;
            end
always@(CurrentState or D)
            case (CurrentState)
            State1:   if (D==0) NextState <= State1;   else NextState <= State2;
            State2:   if (D==0) NextState <= State2;   else NextState <= State3;
            endcase
endmodule
```

5 Draw a state diagram for a sequence detector that will output a logic “1” on Z if and only if it receives 11010 consecutively from an input X. The detection is done recursively. Assume the reset state is State “Initial”. A sample timing sequence is given below. The state diagram has an asynchronous reset pin Reset to bring the state to the initial state. Name your states as S0, S1, S11 and S110, S1101, S11010, etc. You may not need all of the state bubbles.

Clock Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
X	0	1	1	0	1	0	1	1	1	0	1	1	0	1	0	0	1	1	0	1	0
Z	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1



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.

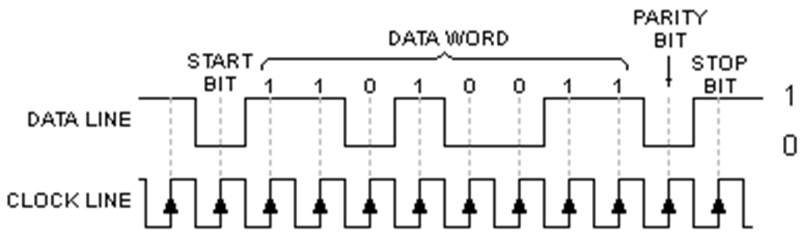


7 Implement the state diagram from Problem 5 with a Moore state machine in Verilog as hw1p7summer2020HUSTdetect11010.v file and simulate your state machine with test patterns from Problem 6 in absolute time specifications as test bench file, hw1p7summer2021HUSTdetect11010_tb.v, to verity the complete state diagram. Attach your Verilog state machine, its RTL schematic, its Verilog test bench and its simulation waveforms in a pdf memo. Annotate your simulation waveforms and initial them to state they are correct.

```
`timescale 1ns / 100ps

// File name     : hw1p7summer2020HUSTdetect11010.v
// Jianjian Song
// 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;




endmodule
```

```
``timescale 1ns / 1ps
//Summer 2021 HUST

module hw1p7summer2021HUSTdetect11010_tb;

reg x, clk, reset;
wire detected;
wire [2:0] CurrentState=Unit1.CurrentState;


endmodule
```

8 (Controller Design) Given the data unit and top level circuit files, design a controller for 9-bit serial data transmission with parity, one start bit and one stop bit as shown below. Parity control bit is an input: Parity=0 for Even Parity and Parity=1 for Odd Parity. The speed of transmission is specified by one 4-bit input parameter Speed as the number clock cycles per bit. For example, if Speed=3, then it takes 3 clock cycles for each bit to be sent. The bit rate is controlled by the controller. The following example shows 8-bit data transmission. You will need to assume 9-bit data transmission. The total number of bits is therefore 12 bits for each transmission.



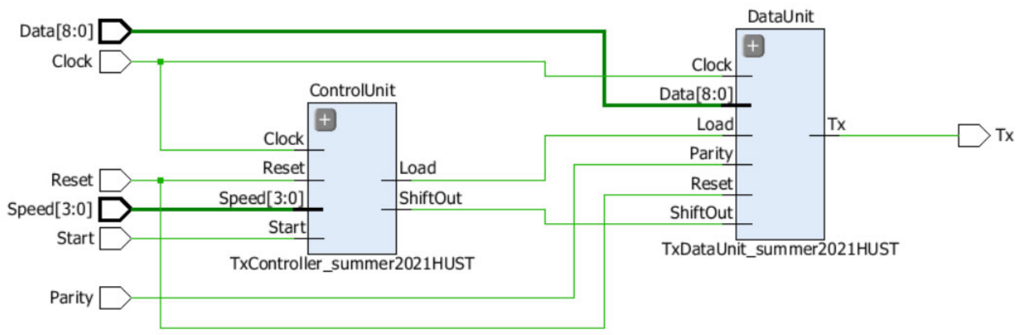
8.(a) *Obtain the data unit and its test bench as well as the overall circuit and test bench files from the instructor. Debug it for errors and verify its functionality by simulation. The data unit generates Start bit, Data bits, Parity, and Stop bit with a 12-bit shift register as seen in the Verilog TxDataUnit module. Its output is Tx serial output. The least significant bit of 9-bit data is transmitted first after Start bit.*

8.(b) *Draw an ASM chart of your controller by hand or with PowerPoint. An example ASM chart in PowerPoint is available from Moodle. The controller may have four states as follows: parameter InitialState=2'd0, LoadState=2'd1, DelayState=2'd2, ShiftState=2'd3. It will generate two control signals: Load and ShiftOut. The bit rate is specified by 4-bit input Speed as a number of clock cycles per bit ranging from 2clock cycles to 16 clock cycles when Speed goes from 1 to 15 (0 is not valid). Signal Load is active first to load data into the Tx data unit and then eleven pulses should be generated on ShiftOut to transmit bits through Tx.*

8.(c) *Design a controller with a Verilog state machine to interface to this data unit TxDataUnit.v so that transmission will start when Start=1. Simulate the controller to verify its correctness. Attach your annotated simulation waveforms as well as your test bench.*

8.(d) *Combine the data unit and controller. Simulate the whole system to show its correctness. Attach your annotated simulation waveforms.*

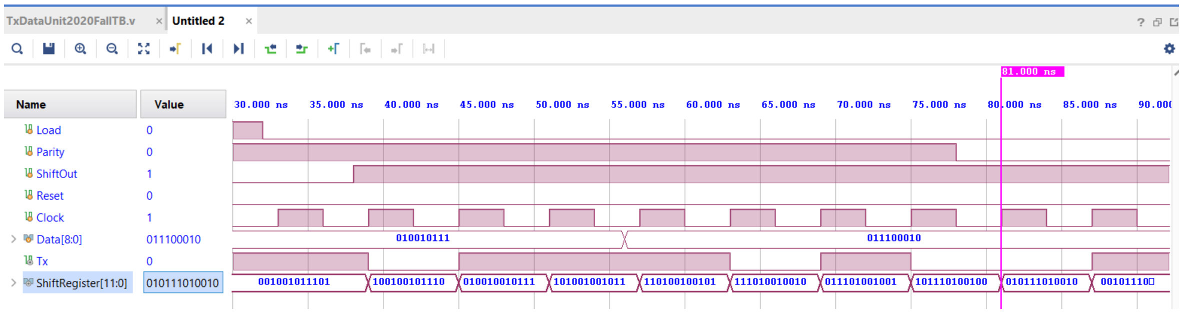
Here is the external behavior of the data unit, the controller and the overall circuit.



Here are TxDataUnit_summer2021HUST.v, its Simulation Test Bench and Simulation Waveforms.

```
`timescale 1ns / 1ps
//Tx Data Unit of serial port transmission unit
//Author: Jianjian Song
//Date: August 2021
//Purpose: Problem #8 Homework #1, Summer 2021 HUST

//Even parity: Parity=0;
//Odd parity: Parity=1;
//1-bit start, 9-bit data, 1-bit parity, 1-bit stop
//data is sent with the least significant bit first
//Load = 1 to load Data into shift register
//shiftOut = 1 to send data to Rx one bit at a time

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@(posedge Clock)
		if(Reset==1 || Load==1) ShiftRegister<={ParityBit, Data, 2'b01};
		else
		if(ShiftOut==1) ShiftRegister<={ShiftRegister[0], ShiftRegister[11:1]};
		else ShiftRegister<=ShiftRegister;
endmodule
```

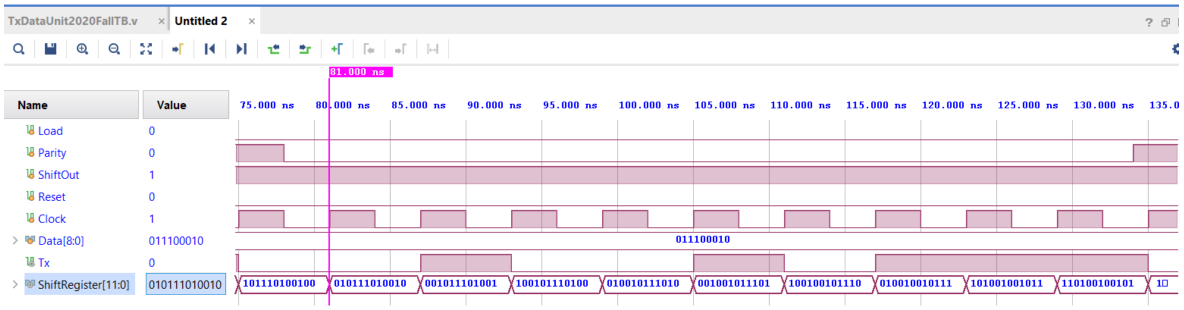
```
`timescale 1ns / 1ps
//Author: Jianjian Song
//Date: August 2021
//Purpose: Problem #8 Homework #1, summer 2021 HUST

module TxDataUnitTB_summer2021HUST;

		reg Load, Parity, ShiftOut, Reset, Clock;
		reg [8:0] Data;
		wire Tx;
		wire [11:0] ShiftRegister=uut.ShiftRegister;
//module TxDataUnit_summer202`HUST #(parameter DataLength=9)(
//    input [DataLength-1:0] Data,
//    input Load, ShiftOut, Parity, Reset, Clock,
//    output Tx);

            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'b101010101;   #56             Data = 8'b111100010;
            #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
```



TxDataUnit2020FallTB.v × Untitled 2 ×

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

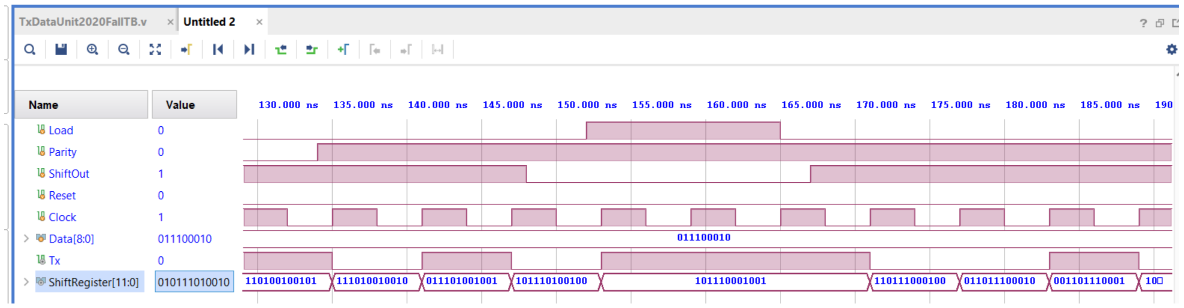
```
81.000 ns
30.000 ns  35.000 ns  40.000 ns  45.000 ns  50.000 ns  55.000 ns  60.000 ns  65.000 ns  70.000 ns  75.000 ns  80.000 ns  85.000 ns  90.00
Data[8:0]:           010010111 | 011100010
ShiftRegister[11:0]: 001001011101 | 100100101110 | 010010010111 | 101001001011 | 110100100101 | 111010010010 | 011101001001 | 101110100100 | 010111010010 | 00101110
```



TxDataUnit2020FallTB.v × Untitled 2 ×

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

```
81.000 ns
75.000 ns  80.000 ns  85.000 ns  90.000 ns  95.000 ns  100.000 ns  105.000 ns  110.000 ns  115.000 ns  120.000 ns  125.000 ns  130.000 ns  135.0
Data[8:0]:           011100010
ShiftRegister[11:0]: 101110100100 | 010111010010 | 001011101001 | 100101110100 | 010010111010 | 001001011101 | 100100101110 | 010010010111 | 101001001011 | 110100100101 | 10
```



TxDataUnit2020FallTB.v × Untitled 2 ×

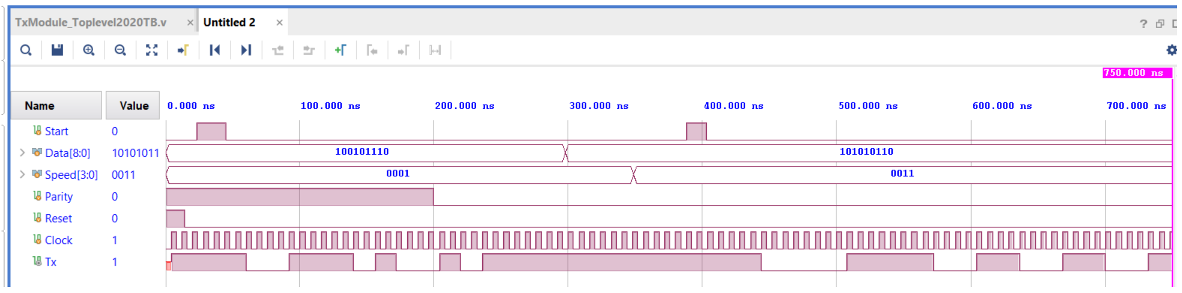
Name	Value
Load	0
Parity	0
ShiftOut	1
Reset	0
Clock	1
Data[8:0]	011100010
Tx	0
ShiftRegister[11:0]	010111010010

```
130.000 ns  135.000 ns  140.000 ns  145.000 ns  150.000 ns  155.000 ns  160.000 ns  165.000 ns  170.000 ns  175.000 ns  180.000 ns  185.000 ns  190
Data[8:0]:           011100010
ShiftRegister[11:0]: 110100100101 | 111010010010 | 011101001001 | 101110100100 | 101110001001 | 110111000100 | 011011100010 | 001101110001 | 10
```

Here is the top-level circuit that combines the data unit and controller. The whole system with your controller should be simulated to show its correctness as this following test waveforms show.

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

```
//Author: Jianjian Song
//Date: August 2021
//Purpose: Problem #8 Homework #1, summer 2021 HUST

// TxModule_Toplevel_summer2021HUST.v
//uart transmission unit with data unit and controller
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;

//module TxController_summer2021HUST(input Start, Reset, Clock,
//input [3:0] Speed,
//output reg Load, ShiftOut,
//reg [4:0] BitCount);


TxController_summer2021HUST ControlUnit(Start, Reset, Clock, Speed, Load, ShiftOut);

//module TxDataUnit_summer2021HUST #(parameter DataLength=9)(
//input [DataLength-1:0] Data,
//input Load, ShiftOut, Parity, Reset, Clock,
//output Tx);

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

endmodule
```



Name	Value
Start	0
Data[8:0]	10101011
Speed[3:0]	0011
Parity	0
Reset	0
Clock	1
Tx	1