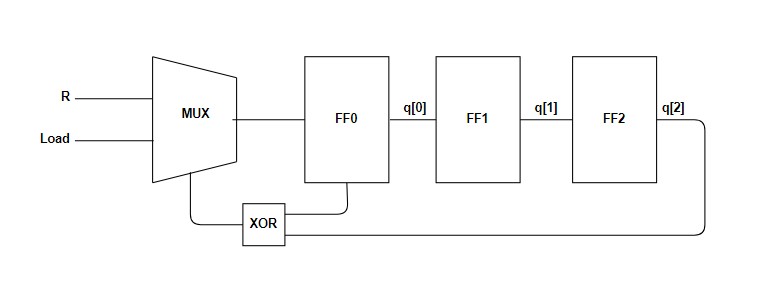
Nang Thiri Wutyi

20113

12/4/2024

**Quiz #2**

1. **a. Diagram**

****

**b. If 1002 is loaded into the LFSR initially,**

**Initial State (t=0):**

q[2] = 1, q[1] = 0, q[0] = 0

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 1 XOR 0 = 1

The LFSR holds the value 100.

**Cycle 1 (t=1):**

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 0

q[1] gets the value of q[0] → q[1] = 0

q[0] gets the feedback value from the XOR gate → q[0] = 1

**New State**:

q[2] = 0, q[1] = 0, q[0] = 1

The LFSR holds the value 001.

**Cycle 2 (t=2):**

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 0 XOR 1 = 1

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 0

q[1] gets the value of q[0] → q[1] = 1

q[0] gets the feedback value from the XOR gate → q[0] = 1

**New State**:

q[2] = 0, q[1] = 1, q[0] = 1

The LFSR holds the value 011.

**Cycle 3 (t=3):**

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 0 XOR 1 = 1

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 1

q[1] gets the value of q[0] → q[1] = 1

q[0] gets the feedback value from the XOR gate → q[0] = 1

**New State**:

q[2] = 1, q[1] = 1, q[0] = 1

The LFSR holds the value 111.

**Cycle 4 (t=4):**

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 1 XOR 1 = 0

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 1

q[1] gets the value of q[0] → q[1] = 1

q[0] gets the feedback value from the XOR gate → q[0] = 0

**New State**:

q[2] = 1, q[1] = 1, q[0] = 0

The LFSR holds the value 110.

**Cycle 5 (t=5):**

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 1 XOR 0 = 1

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 1

q[1] gets the value of q[0] → q[1] = 0

q[0] gets the feedback value from the XOR gate → q[0] = 1

**New State**:

q[2] = 1, q[1] = 0, q[0] = 1

The LFSR holds the value 101.

**Cycle 6 (t=6):**

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 1 XOR 1 = 0

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 0

q[1] gets the value of q[0] → q[1] = 1

q[0] gets the feedback value from the XOR gate → q[0] = 0

**New State**:

q[2] = 0, q[1] = 1, q[0] = 0

The LFSR holds the value 010.

**Cycle 7 (t=7):**

**Feedback Calculation**:

XOR Feedback = q[2] XOR q[0] = 0 XOR 0 = 0

**Shift Operation**:

q[2] gets the value of q[1] → q[2] = 1

q[1] gets the value of q[0] → q[1] = 0

q[0] gets the feedback value from the XOR gate → q[0] = 0

**New State**:

q[2] = 1, q[1] = 0, q[0] = 0

The LFSR holds the value 100 (back to the original state).

### **Generated Sequence:**

100, 001, 011, 111, 110, 101, 010, 100, ...

1. **UART Receiver Design Module**

`timescale 1ns / 1ps

module uart\_receiver (

input clk, // System clock

input rst, // Reset signal

input rx, // Serial data input

input baud\_clk, // Baud rate clock (generated externally)

output reg [7:0] data\_out, // Parallel data output

output reg data\_ready, // Data ready flag

output reg framing\_error // Framing error flag

);

reg [3:0] bit\_count; // Bit counter

reg [7:0] shift\_reg; // Shift register for storing received bits

reg [1:0] state, next\_state;

localparam IDLE = 2'b00,

START\_BIT = 2'b01,

RECEIVE = 2'b10,

STOP\_BIT = 2'b11;

always @(posedge baud\_clk or posedge rst) begin

if (rst) begin

state <= IDLE;

end else begin

state <= next\_state;

end

end

always @(\*) begin

case (state)

IDLE: begin

if (~rx) begin

next\_state = START\_BIT;

end else begin

next\_state = IDLE;

end

end

START\_BIT: begin

if (~rx) begin

next\_state = RECEIVE;

end else begin

next\_state = IDLE;

end

end

RECEIVE: begin

if (bit\_count == 8) begin

next\_state = STOP\_BIT;

end else begin

next\_state = RECEIVE;

end

end

STOP\_BIT: begin

next\_state = IDLE;

end

default: next\_state = IDLE;

endcase

end

always @(posedge baud\_clk or posedge rst) begin

if (rst) begin

shift\_reg <= 8'b0;

bit\_count <= 4'b0;

data\_ready <= 0;

framing\_error <= 0;

end else begin

case (state)

IDLE: begin

data\_ready <= 0;

framing\_error <= 0;

end

START\_BIT: begin

bit\_count <= 4'b0;

end

RECEIVE: begin

shift\_reg <= {rx, shift\_reg[7:1]};

bit\_count <= bit\_count + 1;

end

STOP\_BIT: begin

if (rx == 1) begin

data\_out <= shift\_reg;

data\_ready <= 1;

framing\_error <= 0;

end else begin

framing\_error <= 1;

end

bit\_count <= 0; // Reset bit count

end

endcase

end

end

endmodule

**Testbench Module**

`timescale 1ns / 1ps

module tb\_uart\_receiver;

reg clk;

reg rst;

reg rx;

reg baud\_clk;

wire [7:0] data\_out;

wire data\_ready;

wire framing\_error;

// Instantiate the UART Receiver module

uart\_receiver uut (

.clk(clk),

.rst(rst),

.rx(rx),

.baud\_clk(baud\_clk),

.data\_out(data\_out),

.data\_ready(data\_ready),

.framing\_error(framing\_error)

);

// Clock generation for system clock (50 MHz)

initial begin

clk = 0;

forever #10 clk = ~clk; // 50 MHz clock (20 ns period)

end

// Baud clock generation (9600 baud rate)

initial begin

baud\_clk = 0;

forever #5208 baud\_clk = ~baud\_clk; // 9600 baud (10417 ns period divided by 2 for toggle)

end

// Stimulus generation

initial begin

rst = 1;

rx = 1; // Line idle state (high)

#100; // Hold reset high for 100 ns

rst = 0;

// Send a valid UART frame: Start bit (0), 8 data bits, Stop bit (1)

// Data to send: 8'b10101010 (0xAA)

send\_uart\_frame(8'b10101010);

// Send another frame: Start bit (0), 8 data bits, Stop bit (1)

// Data to send: 8'b11001100 (0xCC)

#100000; // Wait some time before sending next frame

send\_uart\_frame(8'b11001100);

// Finish simulation after some delay

#500000;

$finish;

end

// UART frame generator task

task send\_uart\_frame(input [7:0] data);

integer i;

begin

// Start bit

rx = 0;

#104170; // Wait 1 baud period

// Data bits (LSB first)

for (i = 0; i < 8; i = i + 1) begin

rx = data[i];

#104170; // Wait 1 baud period

end

// Stop bit

rx = 1;

#104170; // Wait 1 baud period

// Idle state (line high)

#104170;

end

endtask

// Monitoring

initial begin

$monitor("Time=%0t | State=%b | rx=%b | data\_out=%b | data\_ready=%b | framing\_error=%b",

$time, uut.state, rx, data\_out, data\_ready, framing\_error);

end

// Waveform dump for debugging

initial begin

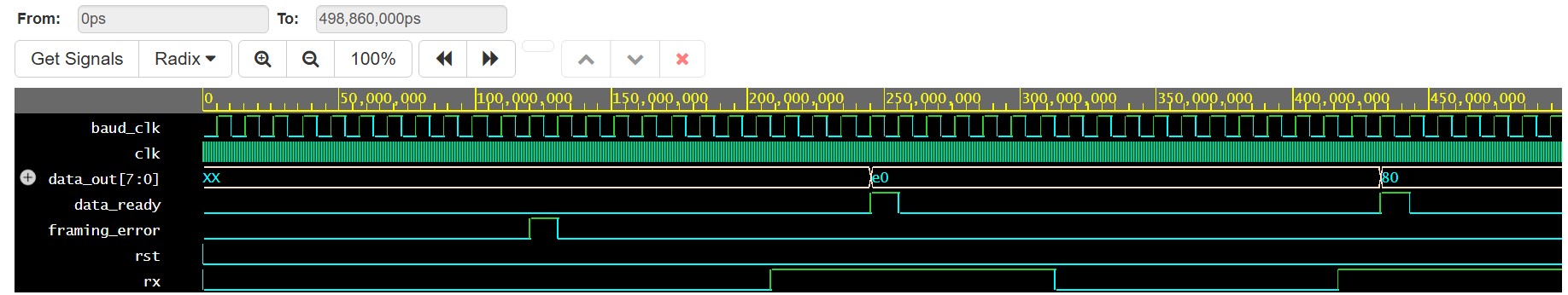
$dumpfile("uart\_receiver.vcd");

$dumpvars(1, tb\_uart\_receiver);

end

endmodule

**Results**



**FSM Diagram**