**Assignment 9**

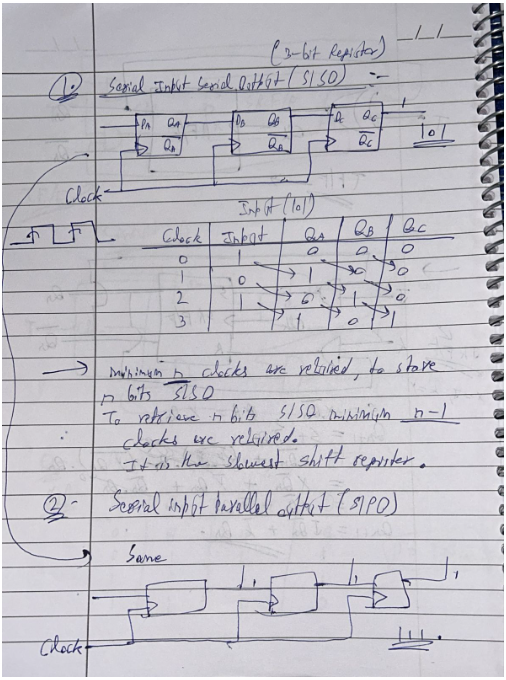
**Name-Aditya Agrawal**

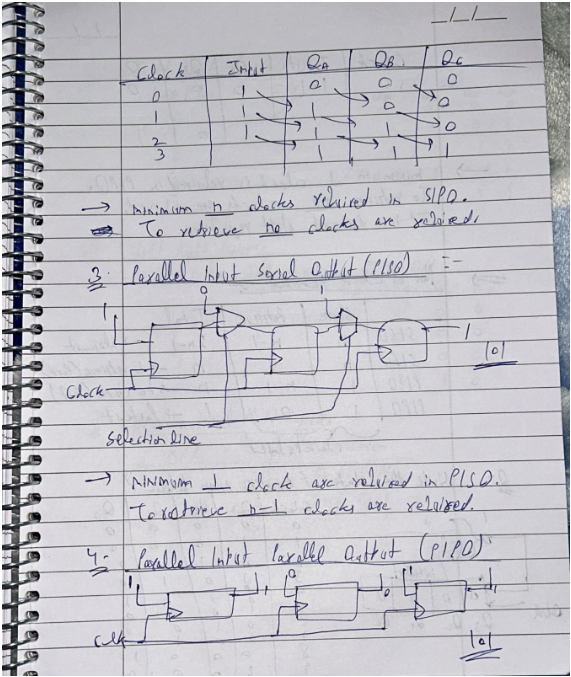
**Enroll no-S24CSEU1321**

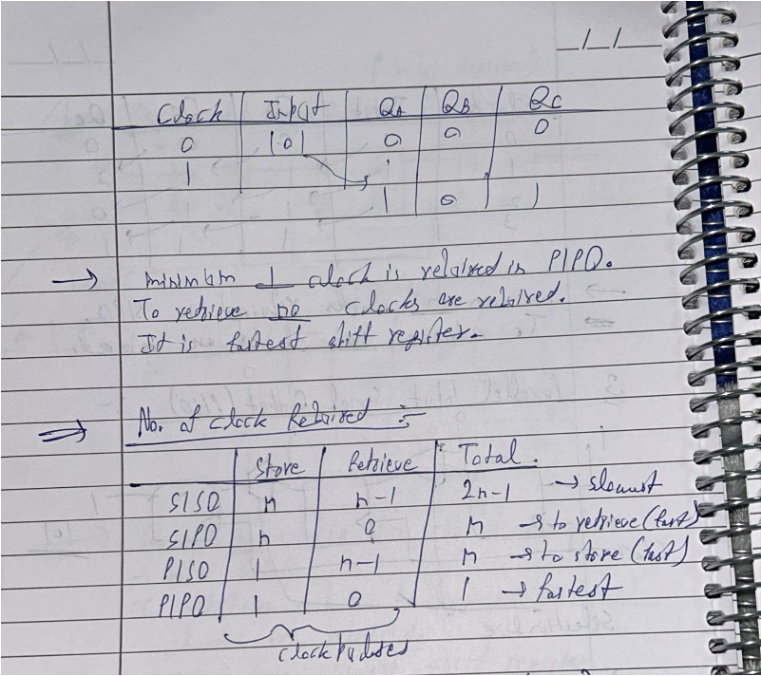
1. A shift register is a sequential logic circuit that stores and shifts binary data (1s and 0s) in a linear fashion. It consists of a cascade of flip-flops (usually D-type), where the output of one flip-flop is connected to the input of the next. Data can be shifted left or right, depending on the design.

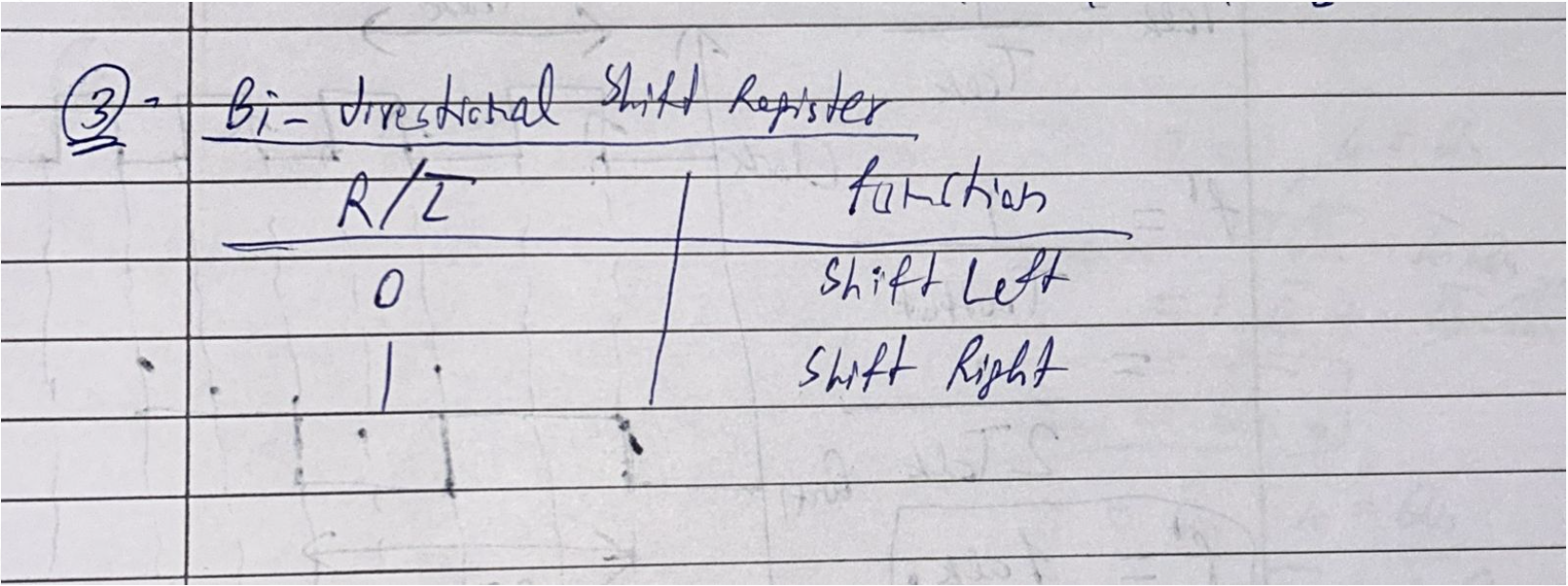
**Types of Shift Registers:**

1. Serial-In Serial-Out (SISO) Shift Register
2. Serial-In Parallel-Out (SIPO) Shift Register
3. Parallel-In Serial-Out (PISO) Shift Register
4. Parallel-In Parallel-Out (PIPO) Shift Register
5. Bidirectional Shift Register









**Applications of Shift Registers:**

Data Storage & Transfer, Time Delay Circuits, Arithmetic Operations etc.

1. **2\_verilog:**

**//Aman Chaudhary**

**//S24CSEU1348**

module Register\_4bit (

input wire clk, // Clock signal

input wire reset, // Asynchronous reset (active high)

input wire load, // Load control signal (1 = load, 0 = hold)

input wire [3:0] d, // 4-bit input data

output reg [3:0] q // 4-bit output (registered value)

);

// Register behavior

always @(posedge clk or posedge reset) begin

if (reset) begin

q <= 4'b0000; // Reset to 0 (asynchronous)

end

else if (load) begin

q <= d; // Load new data on clock edge if load=1

end

// Else, retain previous value (implicit)

end

endmodule

**2\_testbench:**

`timescale 1ns / 1ps

module Register\_4bit\_tb;

// Inputs

reg clk;

reg reset;

reg load;

reg [3:0] d;

// Outputs

wire [3:0] q;

// Instantiate the Unit Under Test (UUT)

Register\_4bit uut (

.clk(clk),

.reset(reset),

.load(load),

.d(d),

.q(q)

);

// Clock generation (10ns period)

initial begin

clk = 0;

forever #5 clk = ~clk; // Toggle every 5ns (100MHz clock)

end

// Stimulus generation

initial begin

// Initialize Inputs

reset = 1;

load = 0;

d = 4'b0000;

// Dump waveform (for GTKWave/Vivado/ModelSim)

$dumpfile("register\_4bit\_wave.vcd");

$dumpvars(0, Register\_4bit\_tb); // Dump all variables

// Release reset after 10ns

#10 reset = 0;

// Test case 1: Load data (1010)

d = 4'b1010;

load = 1;

#10 load = 0;

// Test case 2: Change input but don't load (q should not change)

d = 4'b1111;

#10;

// Test case 3: Load new data (1111)

load = 1;

#10 load = 0;

// Test case 4: Reset again

reset = 1;

#10 reset = 0;

// Test case 5: Load new data (0101)

d = 4'b0101;

load = 1;

#10 load = 0;

// End simulation

#20 $finish;

end

// Monitor changes (optional)

initial begin

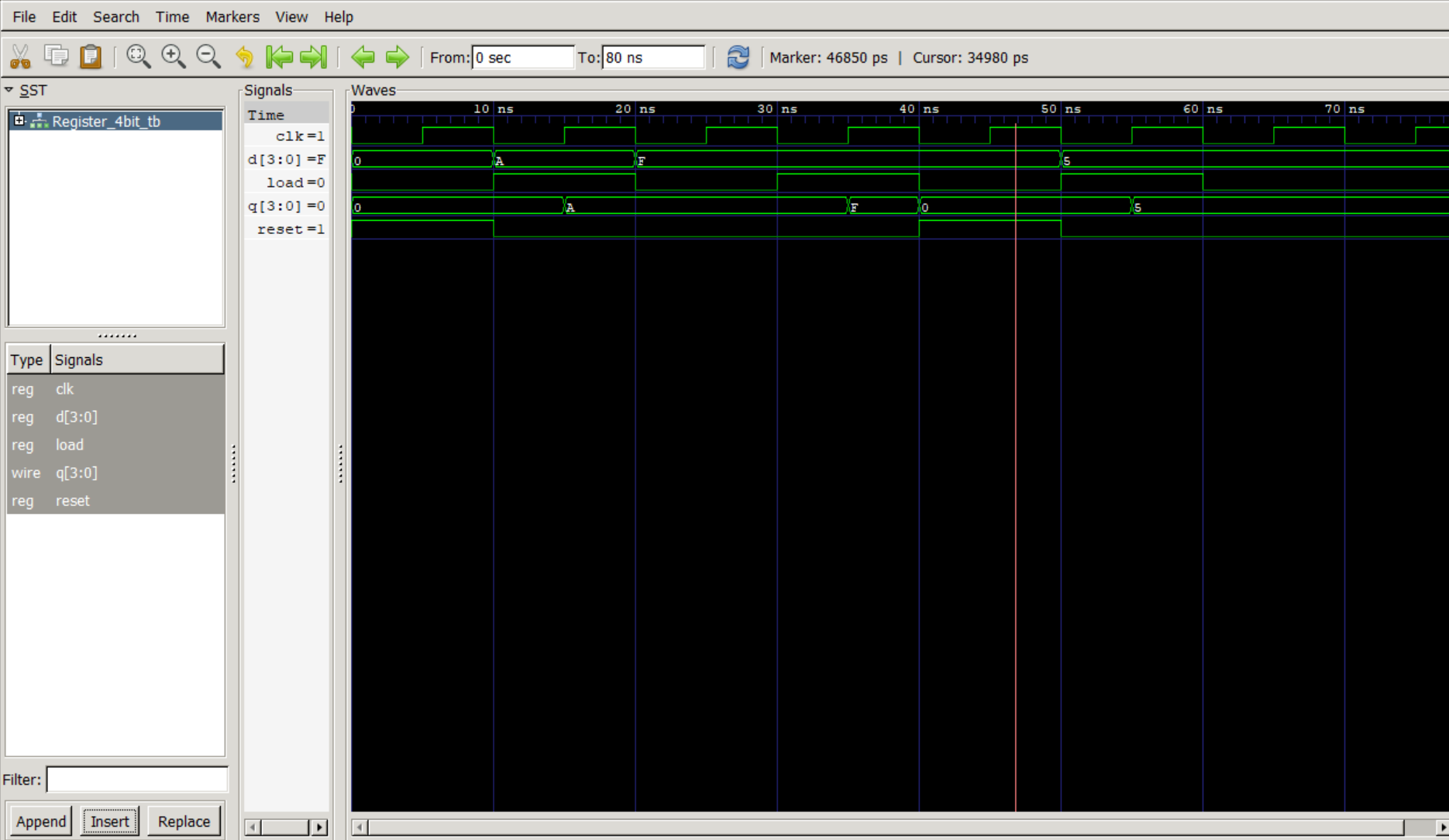
$monitor("Time = %0t ns | clk=%b reset=%b load=%b d=%4b q=%4b",

$time, clk, reset, load, d, q);

end

endmodule

**2\_waveform:**

****

1. **3\_verilog:**

**//Aman Chaudhary**

**//S24CSEU1348**

module ShiftRegister\_8bit (

input wire clk, // Clock signal

input wire reset, // Asynchronous reset (active high)

input wire ser\_in, // Serial input (shifted in LSB-first)

output reg [7:0] q // 8-bit parallel output

);

// Shift register behavior

always @(posedge clk or posedge reset) begin

if (reset) begin

q <= 8'b00000000; // Reset all bits to 0

end

else begin

q <= {ser\_in, q[7:1]}; // Right shift (MSB gets ser\_in, others shift right)

end

end

endmodule

**3\_testbench:**

`timescale 1ns / 1ps

module ShiftRegister\_8bit\_tb;

// Inputs

reg clk;

reg reset;

reg ser\_in;

// Outputs

wire [7:0] q;

// Instantiate the shift register

ShiftRegister\_8bit uut (

.clk(clk),

.reset(reset),

.ser\_in(ser\_in),

.q(q)

);

// Clock generation (10ns period)

initial begin

clk = 0;

forever #5 clk = ~clk; // Toggle every 5ns (100MHz clock)

end

// Stimulus generation

initial begin

// Initialize

reset = 1;

ser\_in = 0;

// Dump waveform (for GTKWave/Vivado/ModelSim)

$dumpfile("shift\_register\_8bit\_wave.vcd");

$dumpvars(0, ShiftRegister\_8bit\_tb);

// Release reset after 10ns

#10 reset = 0;

// Shift in data: 11010010 (LSB first)

ser\_in = 0; #10; // Bit 0

ser\_in = 1; #10; // Bit 1

ser\_in = 0; #10; // Bit 2

ser\_in = 0; #10; // Bit 3

ser\_in = 1; #10; // Bit 4

ser\_in = 0; #10; // Bit 5

ser\_in = 1; #10; // Bit 6

ser\_in = 1; #10; // Bit 7

// Check output (should be 11010010)

#10;

// Reset and test again

reset = 1;

#10 reset = 0;

ser\_in = 1;

#80; // Shift in 1s (result: 11111111)

// End simulation

#10 $finish;

end

// Monitor changes

initial begin

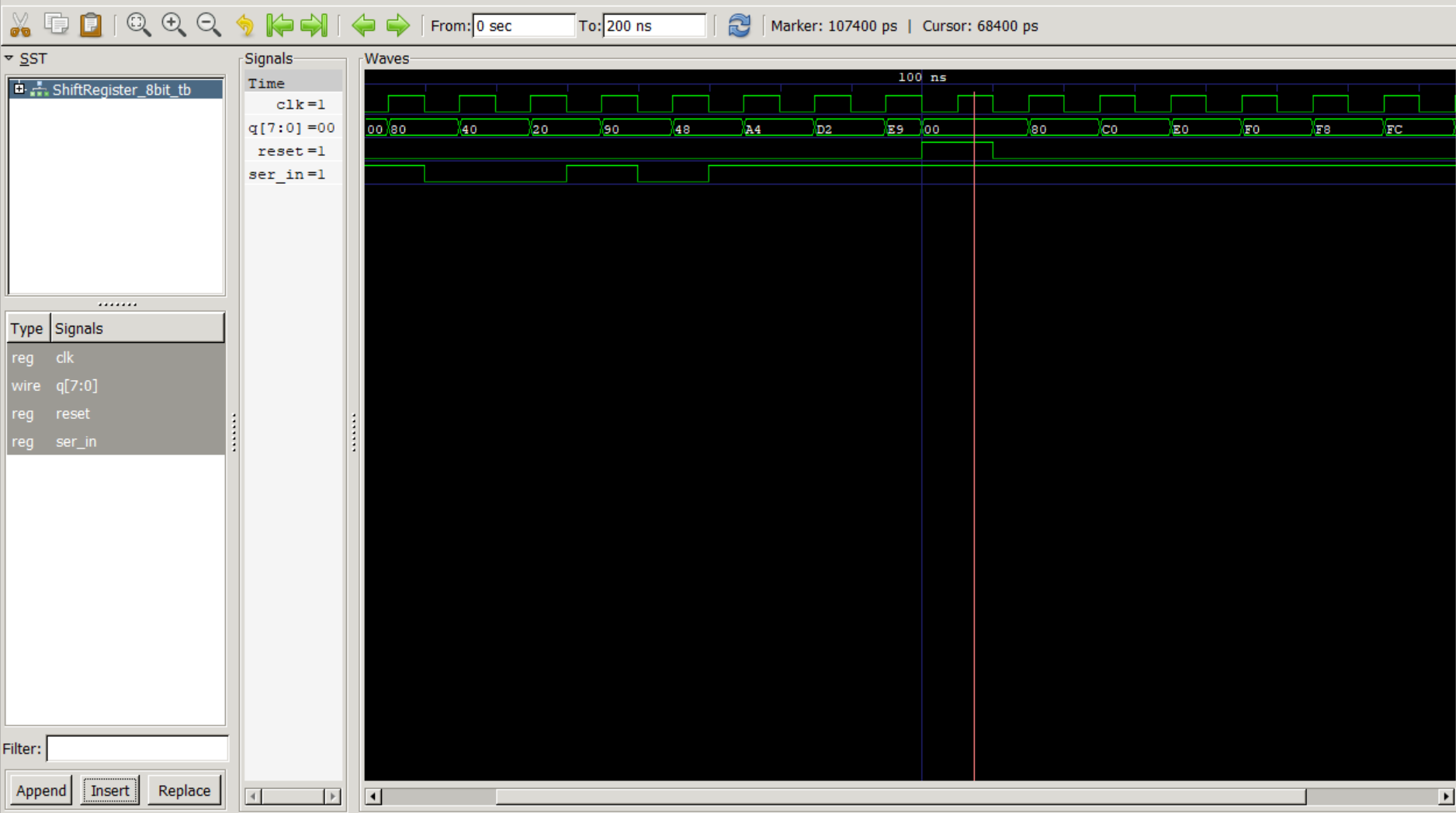
$monitor("Time = %0t ns | clk=%b reset=%b ser\_in=%b q=%8b",

$time, clk, reset, ser\_in, q);

end

endmodule

**3\_waveform:**

****