**Module: R3: DLD + DSD**

**Section:** Combinational Circuits **Task:** Design Problem

**Design Problem**

**Combination Circuits**

* **Verilog Code:**

//Author: Noman Rafiq

//Date: July 11, 2024

module shifter\_circuit (input [7:0] data\_in, input [2:0] shift\_amount, input shift\_direction, output reg [7:0] result);

//Function Definition

function [7:0] shifted\_value (input [7:0] data\_in, input [2:0] shift\_amount, input shift\_direction);

begin

if(shift\_direction)

shifted\_value = (data\_in << shift\_amount);

else

shifted\_value = (data\_in >> shift\_amount);

end

endfunction

always @(\*)

begin

result = shifted\_value(data\_in, shift\_amount, shift\_direction);

end

endmodule

* **Testbench:**

module tb\_shifter\_circuit;

reg [7:0] data\_in;

reg [2:0] shift\_amount;

reg shift\_direction;

wire [7:0] result;

integer results\_file;

// Instantiating the shifter\_circuit module

shifter\_circuit dut (

.data\_in(data\_in),

.shift\_amount(shift\_amount),

.shift\_direction(shift\_direction),

.result(result)

);

initial begin

$dumpfile("wavetrace.vcd");

$dumpvars(0, tb\_shifter\_circuit);

results\_file = $fopen("results.txt", "w");

// Generate 10 random test cases

repeat (10) begin

data\_in = $random;

shift\_amount = $random;

shift\_direction = $random;

#10;

// Self-Checks

if (result == shifted\_value(data\_in, shift\_amount, shift\_direction))

$display("Test passed for data\_in=%d, shift\_amount=%d, shift\_direction=%b", data\_in, shift\_amount, shift\_direction);

else

$display("Test failed for data\_in=%d, shift\_amount=%d, shift\_direction=%b", data\_in, shift\_amount, shift\_direction);

$fwrite(results\_file, "data\_in=%d, shift\_amount=%d, shift\_direction=%b, result=%d\n", data\_in, shift\_amount, shift\_direction, result);

end

$fclose(results\_file);

$finish;

end

function [7:0] shifted\_value (input [7:0] data\_in, input [2:0] shift\_amount, input shift\_direction);

begin

if(shift\_direction)

shifted\_value = (data\_in << shift\_amount);

else

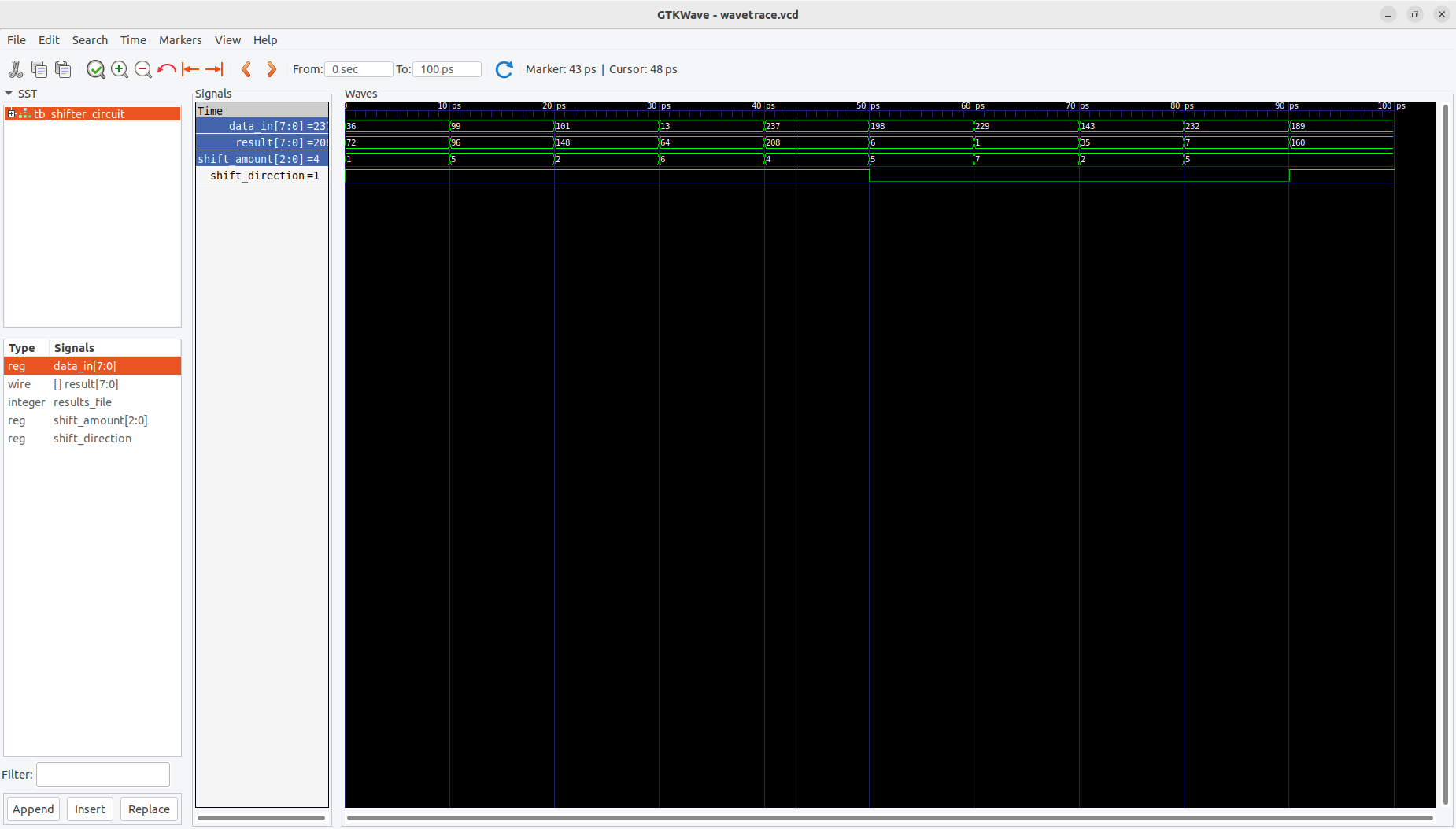
shifted\_value = (data\_in >> shift\_amount);

end

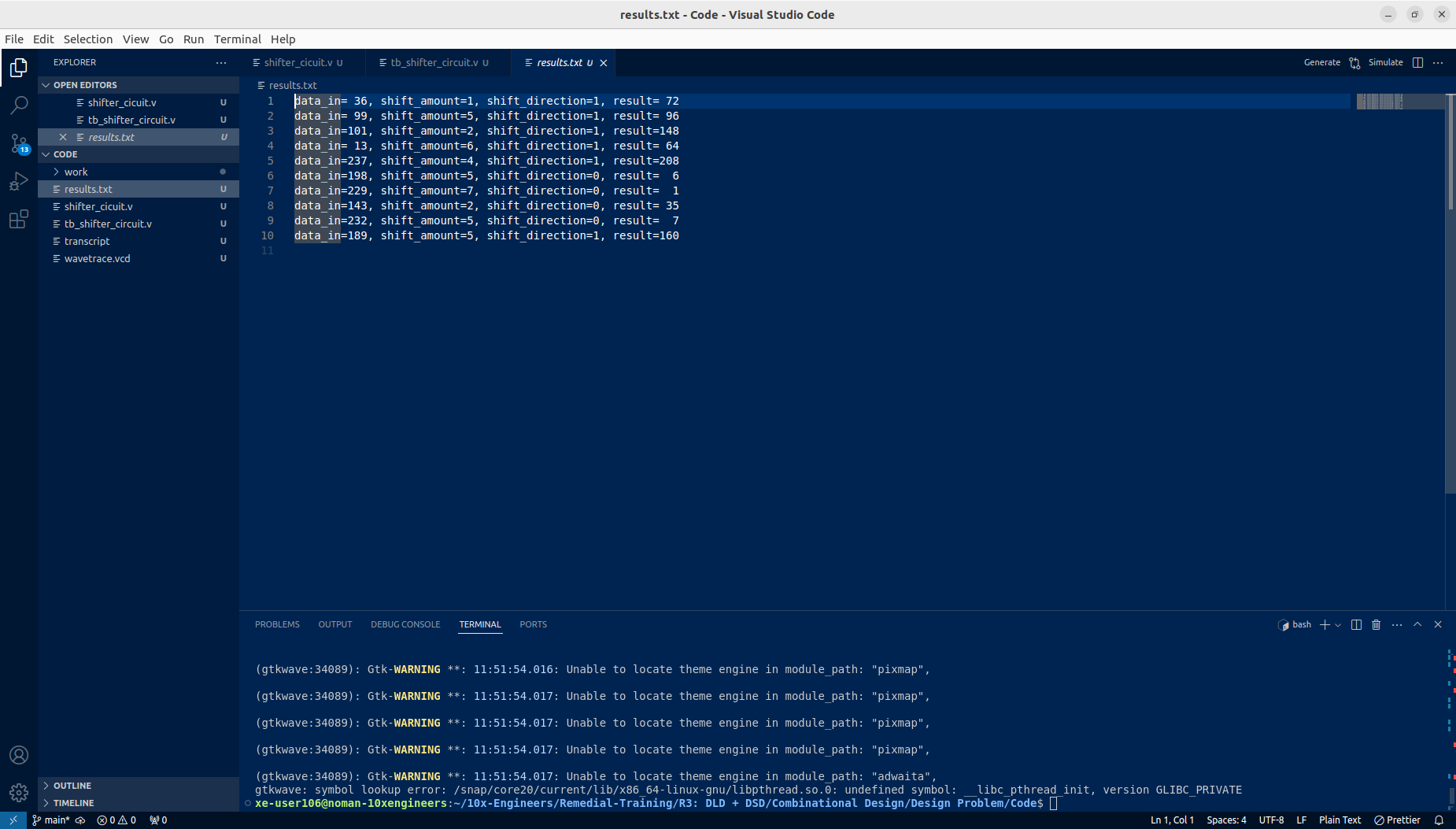
endfunction

endmodule

* **Output:**

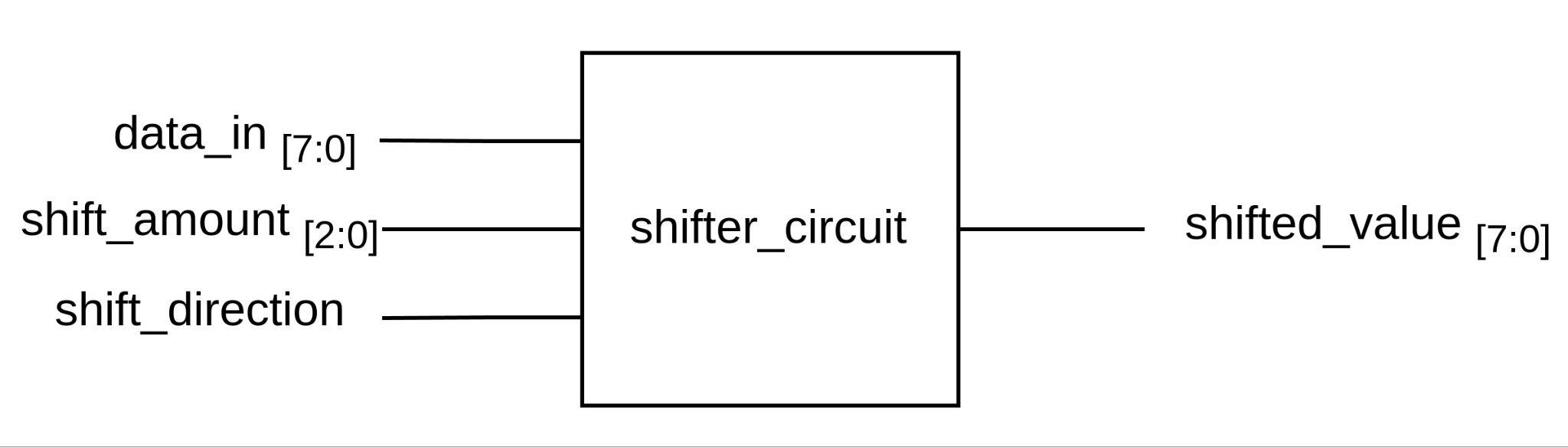
****

The results have been recorded in **results.txt,** as shown below:



* **Schematics:**

The block-level implementation will be as follows:



The block will be utilizing left and right shift blocks that take **data\_in** and **shift\_amount** as inputs and then shifts the value by the n bits provided by the **shift\_amount.** The **shift\_direction** is used to choose one of the two shift operations. It is achieved through a mux as given below:

