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
```

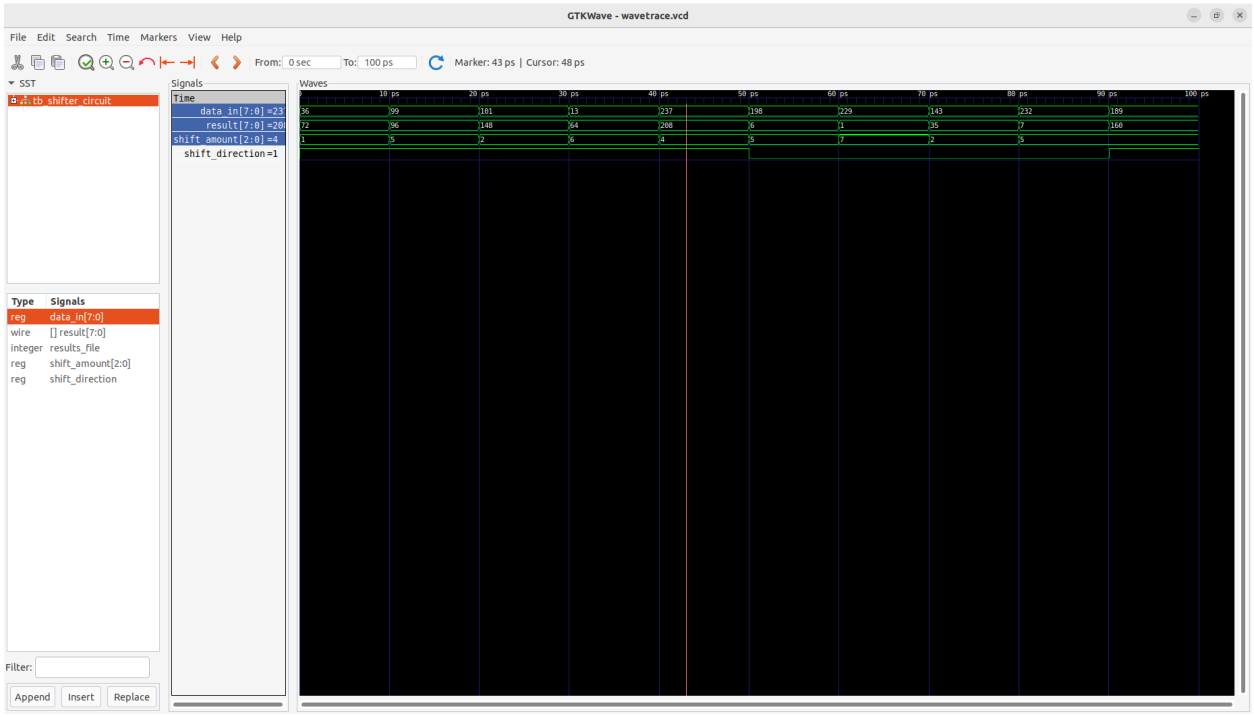
➢ **Testbench:**

```
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
```

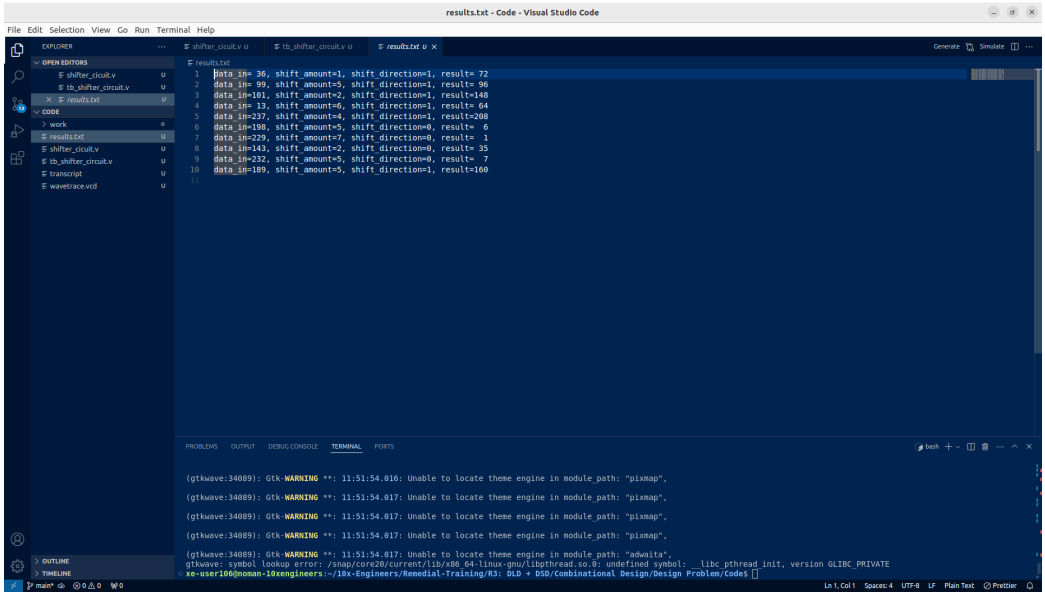
```
  shifter_circuit dut (
    .data_in(data_in),
    .shift_amount(shift_amount),
    .shift_direction(shift_direction),
    .result(result)
  );


  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:**

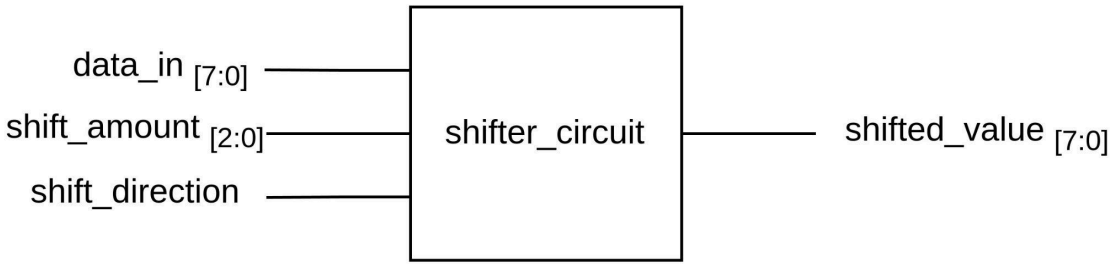


The results have been recorded in **results.txt,** as shown below:



➢ **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: