3 Kogge-Stone Adder (Athul P R)

3.1 Design

```
module gp_calc (
    input wire a,
    input wire b,
    input wire cin,
    output wire g,
    output wire p
);
    assign g = a & b;
    assign p = a ^ b;
endmodule

// Prefix computation module for Kogge-Stone
module prefix_op (
    input wire gi,
    input wire pi,
    input wire gj,
    input wire pj,
    output wire go,
    output wire po
);
    assign go = gi | (pi & gj);
    assign po = pi & pj;
endmodule

// 4-bit Kogge-Stone Adder Core
module kogge_stone_4bit (
    input wire [3:0] a,
    input wire [3:0] b,
    input wire cin,
    output wire [3:0] sum,
    output wire cout
);

    // Level 0: Initial G and P computation
    wire [3:0] g0, p0;

    gp_calc gp0 (.a(a[0]), .b(b[0]), .cin(cin), .g(g0[0]), .p(p0[0]));
    gp_calc gp1 (.a(a[1]), .b(b[1]), .cin(1'b0), .g(g0[1]), .p(p0[1]));
    gp_calc gp2 (.a(a[2]), .b(b[2]), .cin(1'b0), .g(g0[2]), .p(p0[2]));
    gp_calc gp3 (.a(a[3]), .b(b[3]), .cin(1'b0), .g(g0[3]), .p(p0[3]));

    // Level 1: First prefix computation
    wire [3:0] g1, p1;

    assign g1[0] = g0[0];
    assign p1[0] = p0[0];
```

```
    prefix_op pre1_1 (.gi(g0[1]), .pi(p0[1]), .gj(g0[0]), .pj(p0[0]), .go(g1[1]), .po(p1[1]));

    assign g1[2] = g0[2];
    assign p1[2] = p0[2];

    prefix_op pre1_3 (.gi(g0[3]), .pi(p0[3]), .gj(g0[2]), .pj(p0[2]), .go(g1[3]), .po(p1[3]));

    // Level 2: Second prefix computation
    wire [3:0] g2, p2;

    assign g2[0] = g1[0];
    assign p2[0] = p1[0];
    assign g2[1] = g1[1];
    assign p2[1] = p1[1];

    prefix_op pre2_2 (.gi(g1[2]), .pi(p1[2]), .gj(g1[1]), .pj(p1[1]), .go(g2[2]), .po(p2[2]));
    prefix_op pre2_3 (.gi(g1[3]), .pi(p1[3]), .gj(g1[1]), .pj(p1[1]), .go(g2[3]), .po(p2[3]));

    // Carry computation
    wire c0, c1, c2, c3;
    assign c0 = cin;
    assign c1 = g2[0] | (p2[0] & c0);
    assign c2 = g2[1] | (p2[1] & c0);
    assign c3 = g2[2] | (p2[2] & c0);
    assign cout = g2[3] | (p2[3] & c0);

    // Sum computation
    assign sum[0] = a[0] ^ b[0] ^ c0;
    assign sum[1] = a[1] ^ b[1] ^ c1;
    assign sum[2] = a[2] ^ b[2] ^ c2;
    assign sum[3] = a[3] ^ b[3] ^ c3;

endmodule
```

3.2 1-bit D Flip-Flop

```
module dff (
    input wire clk,
    input wire rst,
    input wire d,
    output reg q
);

    always @(posedge clk or posedge rst) begin
        if (rst)
            q <= 1'b0;
        else
            q <= d;
    end

endmodule
```

3.3 Top Module

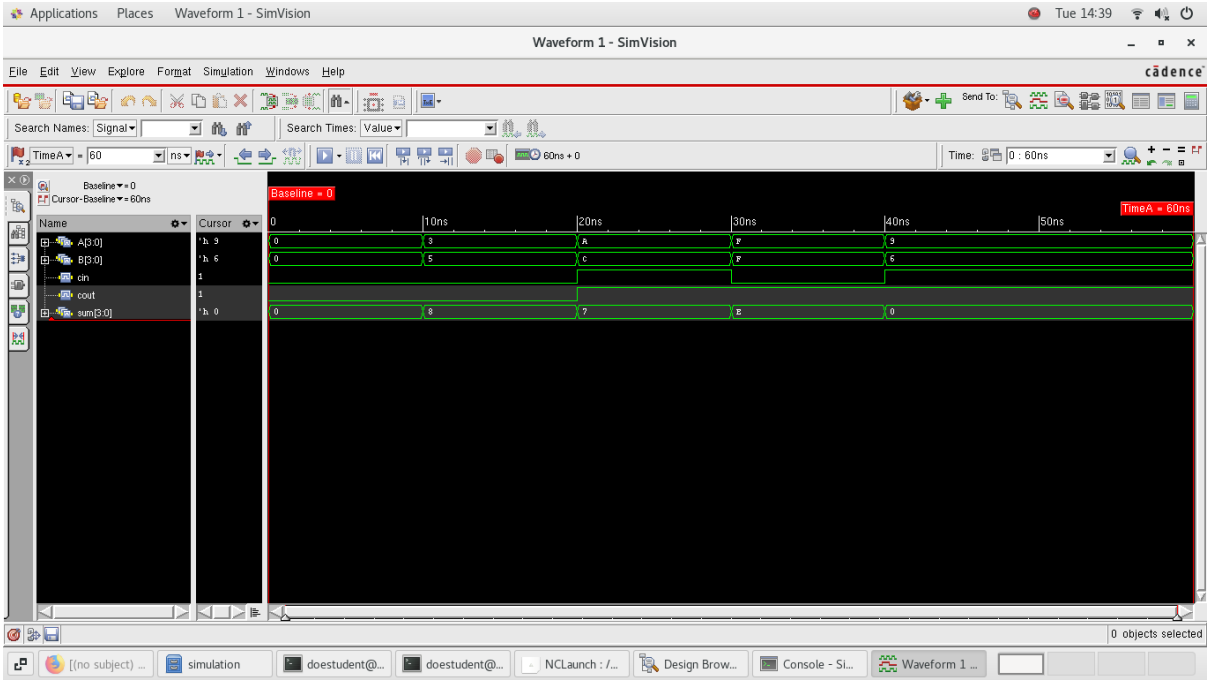
```
module kogge_stone_top (
    input wire clk,
    input wire rst,
    input wire [3:0] a_in,
    input wire [3:0] b_in,
    input wire cin_in,
    output wire [3:0] sum_out,
    output wire cout_out
);

    // Input registers
    wire [3:0] a_reg, b_reg;
    wire cin_reg;

    // Output registers
    reg [3:0] sum_reg;
    reg cout_reg;
    // Input D Flip-Flops
    dff dff_a0 (.clk(clk), .rst(rst), .d(a_in[0]), .q(a_reg[0]));
    dff dff_a1 (.clk(clk), .rst(rst), .d(a_in[1]), .q(a_reg[1]));
    dff dff_a2 (.clk(clk), .rst(rst), .d(a_in[2]), .q(a_reg[2]));
    dff dff_a3 (.clk(clk), .rst(rst), .d(a_in[3]), .q(a_reg[3]));

    dff dff_b0 (.clk(clk), .rst(rst), .d(b_in[0]), .q(b_reg[0]));
    dff dff_b1 (.clk(clk), .rst(rst), .d(b_in[1]), .q(b_reg[1]));
    dff dff_b2 (.clk(clk), .rst(rst), .d(b_in[2]), .q(b_reg[2]));
    dff dff_b3 (.clk(clk), .rst(rst), .d(b_in[3]), .q(b_reg[3]));

    dff dff_cin (.clk(clk), .rst(rst), .d(cin_in), .q(cin_reg));
    // Adder core
    wire [3:0] sum_wire;
    wire cout_wire;

    kogge_stone_4bit adder_core (
        .a(a_reg),
        .b(b_reg),
        .cin(cin_reg),
        .sum(sum_wire),
        .cout(cout_wire)
    );
    // Output D Flip-Flops
    dff dff_sum0 (.clk(clk), .rst(rst), .d(sum_wire[0]), .q(sum_out[0]));
    dff dff_sum1 (.clk(clk), .rst(rst), .d(sum_wire[1]), .q(sum_out[1]));
    dff dff_sum2 (.clk(clk), .rst(rst), .d(sum_wire[2]), .q(sum_out[2]));
    dff dff_sum3 (.clk(clk), .rst(rst), .d(sum_wire[3]), .q(sum_out[3]));

    dff dff_cout (.clk(clk), .rst(rst), .d(cout_wire), .q(cout_out));

endmodule
```

3.4 Testbench

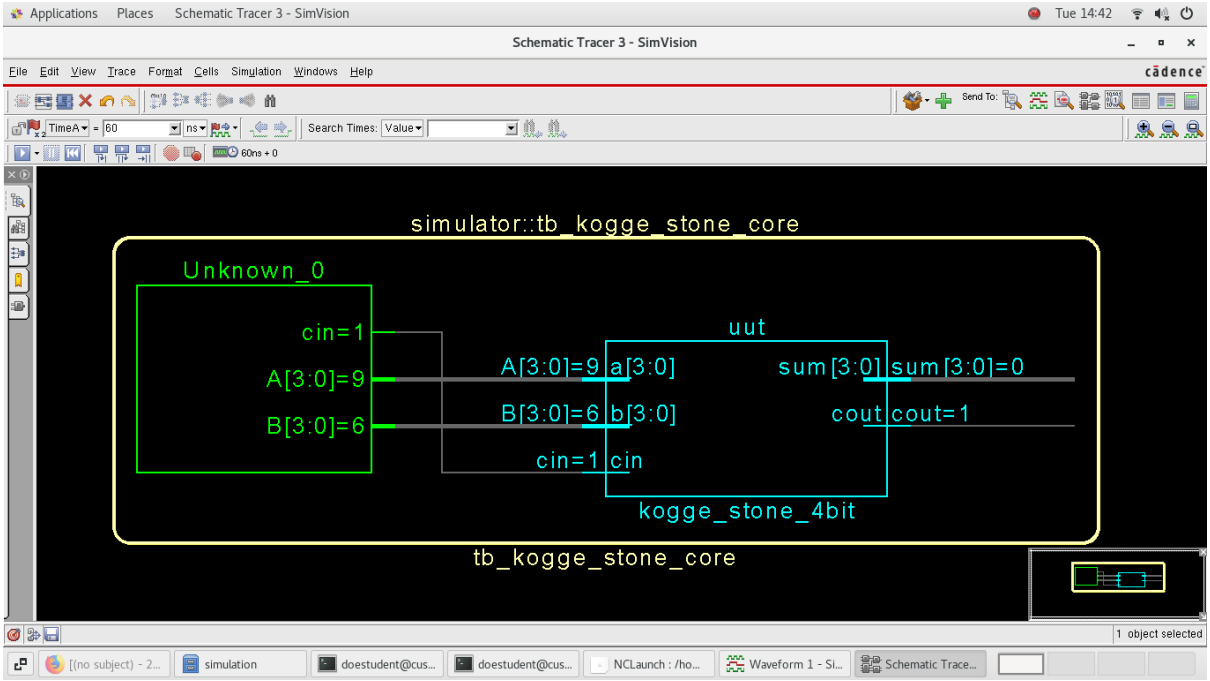
```
module tb_kogge_stone_adder;

    reg clk;
    reg rst;
    reg [3:0] a_in;
    reg [3:0] b_in;
    reg cin_in;
    wire [3:0] sum_out;
    wire cout_out;

    // Instantiate the top module
    kogge_stone_top uut (
        .clk(clk),
        .rst(rst),
        .a_in(a_in),
        .b_in(b_in),
        .cin_in(cin_in),
        .sum_out(sum_out),
        .cout_out(cout_out)
    );

    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk;  // 10ns period
    end

    // Test stimulus
    initial begin
        $dumpfile("kogge_stone_adder.vcd");
        $dumpvars(0, tb_kogge_stone_adder);

        // Initialize
        rst = 1;
        a_in = 4'b0000;
        b_in = 4'b0000;
        cin_in = 1'b0;
```

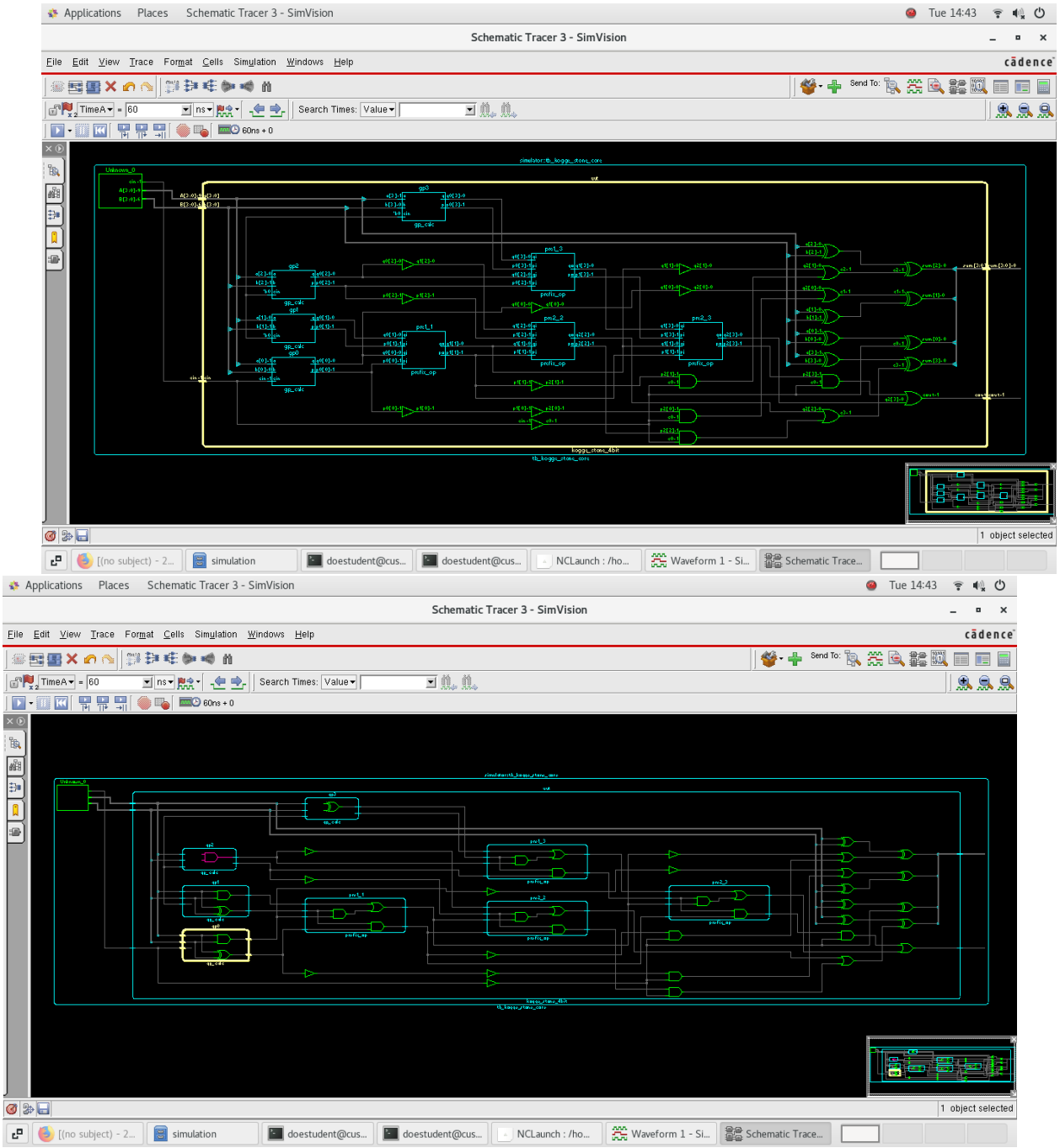
```
        // Reset
        #10;
        rst = 0;
        #10;

        // Test cases
        $display("Starting Kogge-Stone Adder Tests");
        $display("Time\tA\tB\tCin\tSum\tCout\tExpected");

        // Existing test cases
        a_in = 4'b0000; b_in = 4'b0000; cin_in = 1'b0; #20;
        a_in = 4'b0001; b_in = 4'b0001; cin_in = 1'b0; #20;
        a_in = 4'b0101; b_in = 4'b0011; cin_in = 1'b1; #20;
        a_in = 4'b0111; b_in = 4'b1000; cin_in = 1'b0; #20;
        a_in = 4'b1111; b_in = 4'b1111; cin_in = 1'b1; #20;
        a_in = 4'b1001; b_in = 4'b0110; cin_in = 1'b0; #20;
        a_in = 4'b1100; b_in = 4'b0100; cin_in = 1'b1; #20;
        a_in = 4'b0000; b_in = 4'b1111; cin_in = 1'b1; #20;

        // Additional test cases
        a_in = 4'b0011; b_in = 4'b0111; cin_in = 1'b1; #20;
        a_in = 4'b0010; b_in = 4'b0101; cin_in = 1'b0; #20;
        a_in = 4'b1110; b_in = 4'b0001; cin_in = 1'b1; #20;
        a_in = 4'b0110; b_in = 4'b1001; cin_in = 1'b1; #20;
        a_in = 4'b1101; b_in = 4'b0010; cin_in = 1'b0; #20;
        a_in = 4'b1011; b_in = 4'b0100; cin_in = 1'b1; #20;
        a_in = 4'b1000; b_in = 4'b0111; cin_in = 1'b1; #20;
        a_in = 4'b0101; b_in = 4'b1010; cin_in = 1'b0; #20;
        a_in = 4'b0010; b_in = 4'b1101; cin_in = 1'b1; #20;
        a_in = 4'b0111; b_in = 4'b0101; cin_in = 1'b1; #20;
        a_in = 4'b0001; b_in = 4'b1110; cin_in = 1'b0; #20;

        // Newly added test cases
        a_in = 4'b0100; b_in = 4'b1011; cin_in = 1'b0; #20;
        a_in = 4'b0111; b_in = 4'b0110; cin_in = 1'b1; #20;
        a_in = 4'b1000; b_in = 4'b1001; cin_in = 1'b0; #20;
        a_in = 4'b1010; b_in = 4'b0010; cin_in = 1'b1; #20;
        a_in = 4'b1110; b_in = 4'b0011; cin_in = 1'b0; #20;
        a_in = 4'b0101; b_in = 4'b1100; cin_in = 1'b1; #20;
        a_in = 4'b1101; b_in = 4'b0111; cin_in = 1'b0; #20;
        a_in = 4'b0011; b_in = 4'b1100; cin_in = 1'b1; #20;
        a_in = 4'b1001; b_in = 4'b0101; cin_in = 1'b1; #20;
        a_in = 4'b0010; b_in = 4'b1110; cin_in = 1'b0; #20;

        $display("Test completed");
        #20;
        $finish;
    end

    // Monitor changes
    initial begin
        $monitor("At time %t: A=%b B=%b Cin=%b -> Sum=%b Cout=%b",
                 $time, a_in, b_in, cin_in, sum_out, cout_out);
    end

endmodule
```

3.5 Waveform

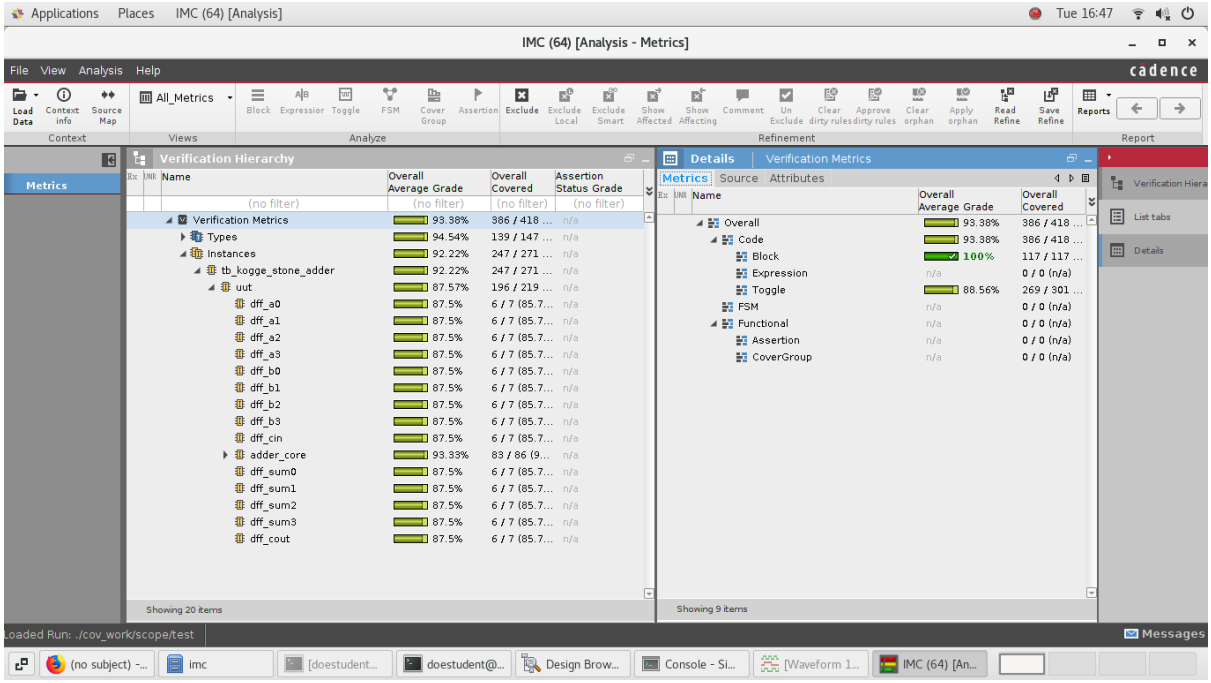


3.6 Schematic Diagram




Applications Places Schematic Tracer 3 - SimVision
Tue 14:43
Schematic Tracer 3 - SimVision
File Edit View Trace Format Cells Simulation Windows Help
cadence
Send To:
Search Times:
Value
60ns + 0
1 object selected
simulation
doestudent@cus...
NCLaunch : /ho...
Waveform 1 - Si...
Schematic Trace...

3.7 Coverage Analysis



3.8 Constraints File

```
create_clock -name clk -period 1 -waveform {0 0.5} [get_ports clk]
set_clock_transition -rise 0.1 [get_clocks clk]
set_clock_transition -fall 0.1 [get_clocks clk]
set_clock_uncertainty 0.01 [get_ports clk]

set_input_delay -max 0.4 -clock clk [get_ports {a_in[0] a_in[1] a_in[2] a_in[3] b_in[0] b_in[1] b_in[2] b_in[3] cin_in}]
set_output_delay -max 0.4 -clock clk [get_ports {sum_out[0] sum_out[1] sum_out[2] sum_out[3] cout_out}]
```

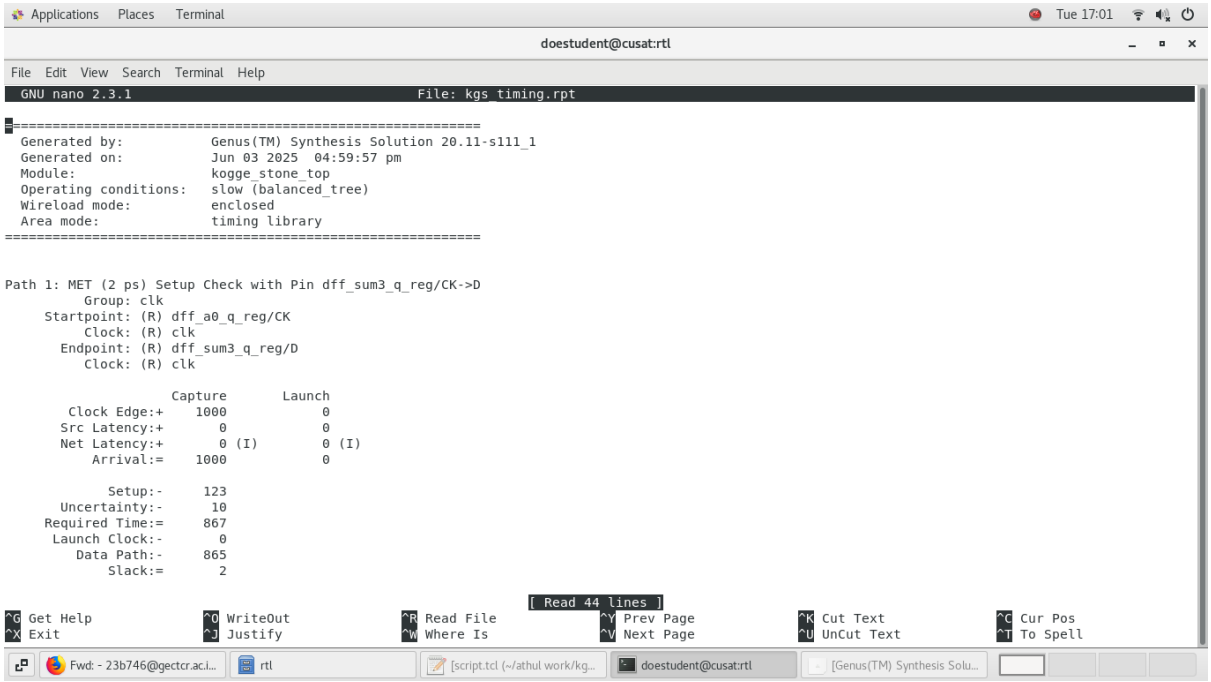
3.9 Script File

```
read_libs /home/installs/FOUNDRY/digital/90nm/dig/lib/slow.lib
read_hdl kgs.v kgstm.v dff.v
elaborate
read_sdc constraints.sdc
syn_generic
syn_map
syn_opt
gui_show
write_hdl > kgs_netlist.v
write_sdc > output_constraints.sdc
report_area > kgs_area.rpt
report_power > kgs_power.rpt
report_timing > kgs_timing.rpt
report_gates > kgs_gates.rpt
gui_show
```

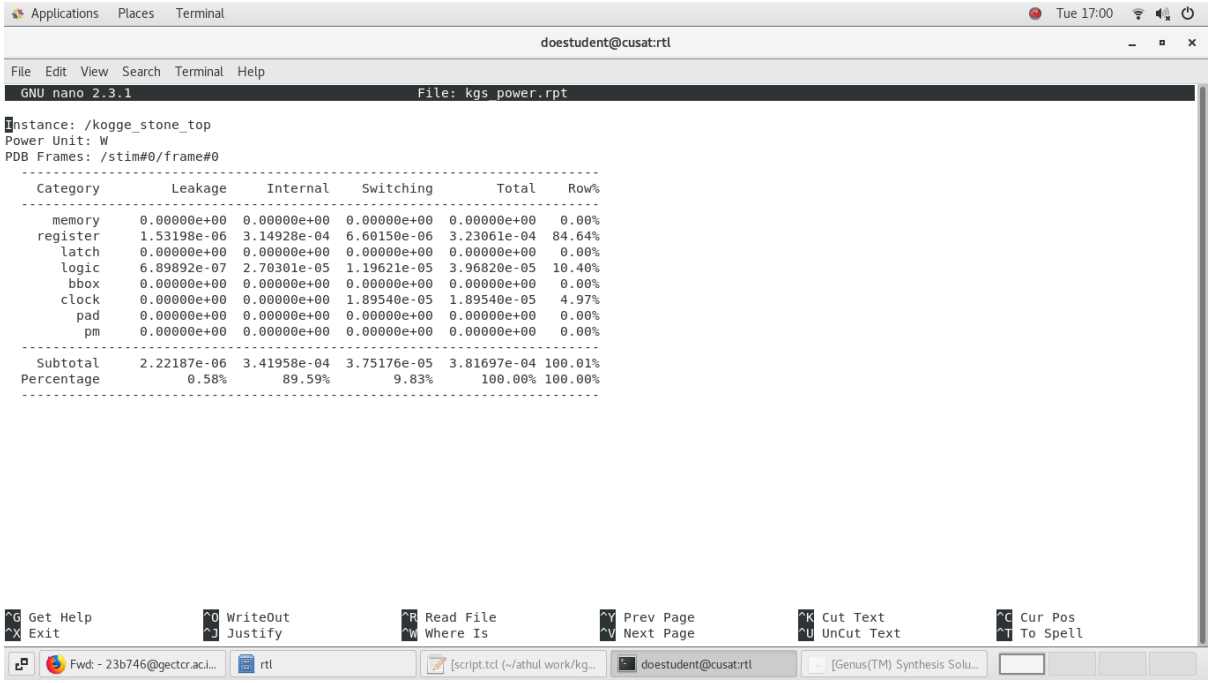
3.10 Reports

Area Report

Timing Report



Power Report



Gate Report