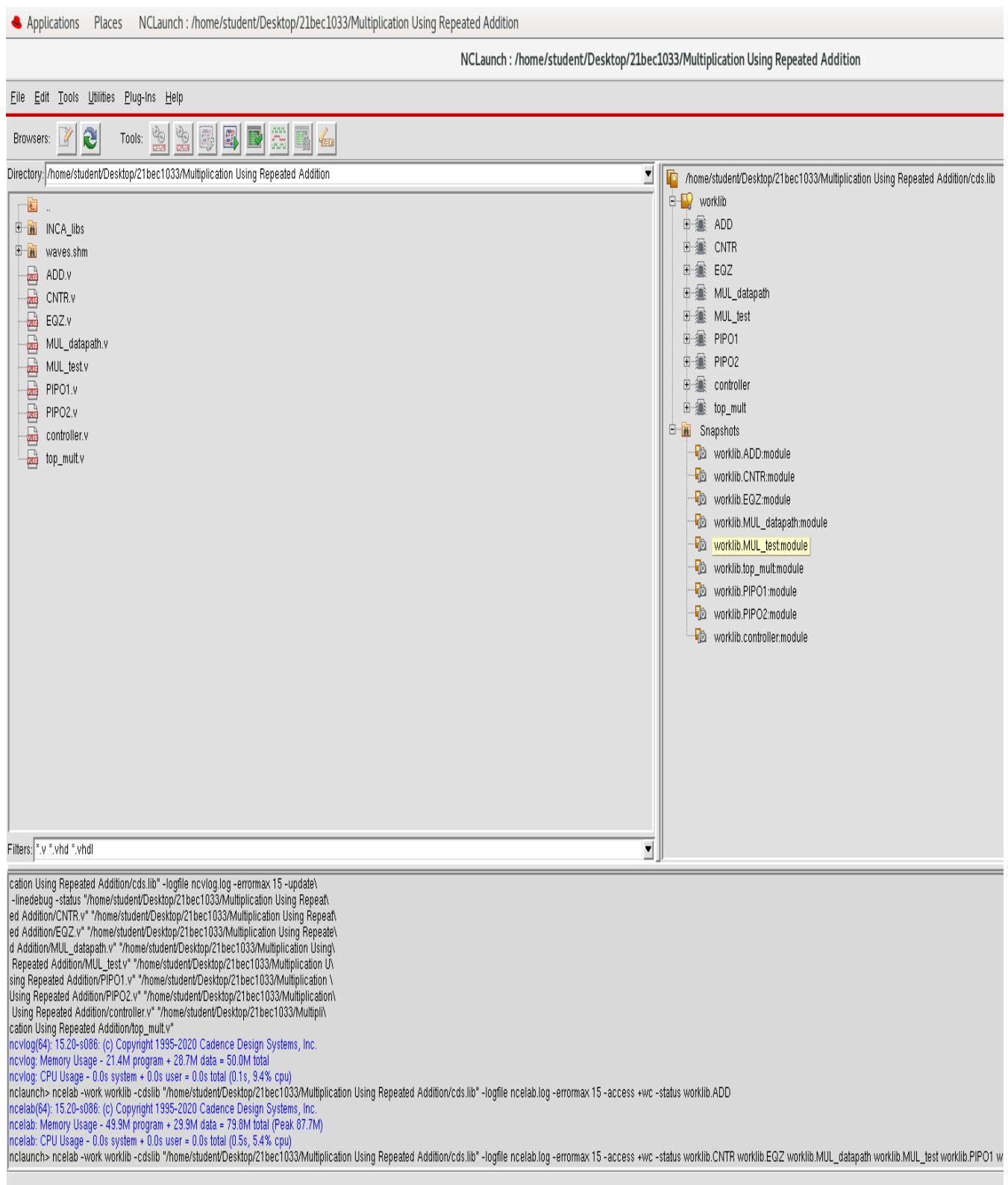


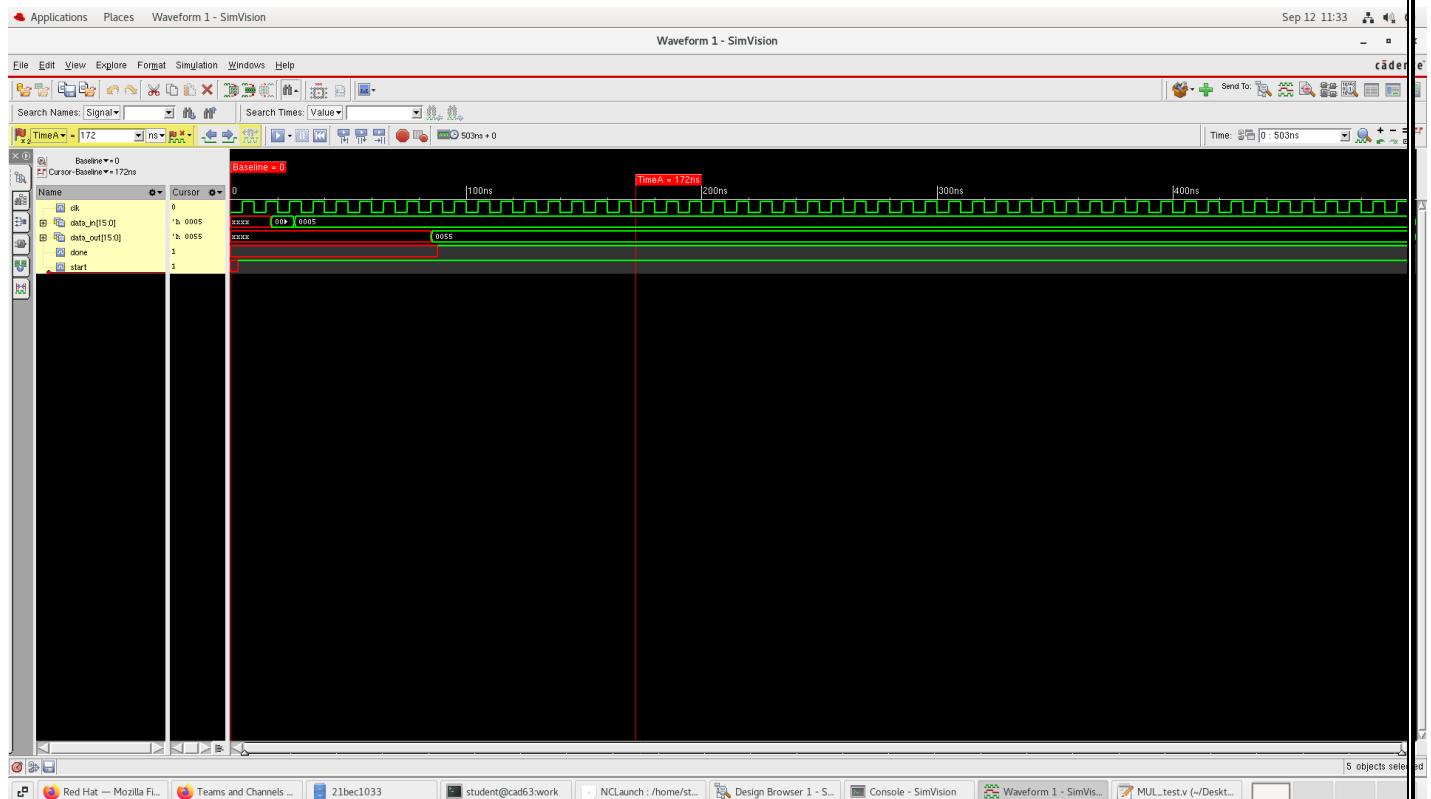
Lab4 1: Design and simulation of a multiplier by repeated addition using Datapath and Control path logic

NCLaunch Compilation and Elaboration:



Lab4 1: Design and simulation of a multiplier by repeated addition using Datapath and Control path logic

Observations/Simulation Waveforms:



Inference: In the above output waveform, the data_in is provided with 5 & 11 values and the data_out is obtained as 55 which is the multiplied result after repeated addition. Output “done” is equal to 1 whenever multiplication result is obtained.

Result: Hence, a multiplier is designed and the simulation and verification of behavior of multiplier by repeated addition using data path and Control path logic is performed for the given stimuli in the test bench.

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

Aim: To synthesize a multiplier by repeated addition using Datapath and Control path logic and to get the gate level netlist with timing, area, and power reports.

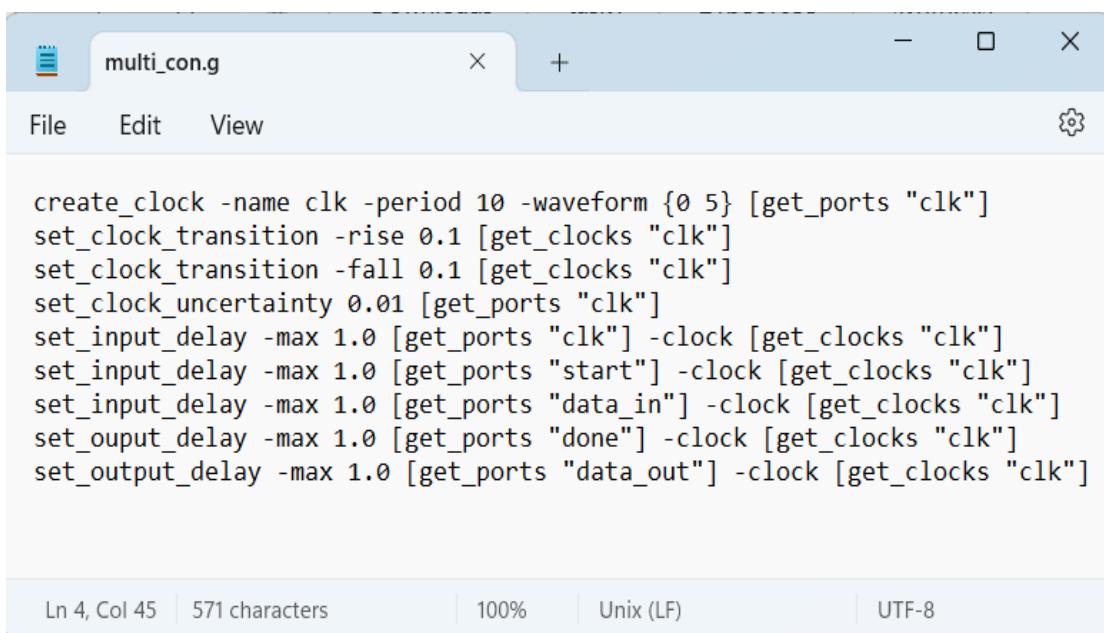
EDA Tools Used: Cadence Genus

Description:

Datapath is the hardware that performs all the required operations. Control is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc. Simple datapath components include memory (stores the current instruction), PC or program counter (stores the address of current instruction), and ALU (executes current instruction). Control path is the Finite state machine designed by using Moore or Mealy models.

Procedure:

1. Copy the fast.lib and slow.lib files into the folder where (.v) files are located.
2. Create a Synopsys design constraint file (.g file) by entering the design constraints required for the synthesis.



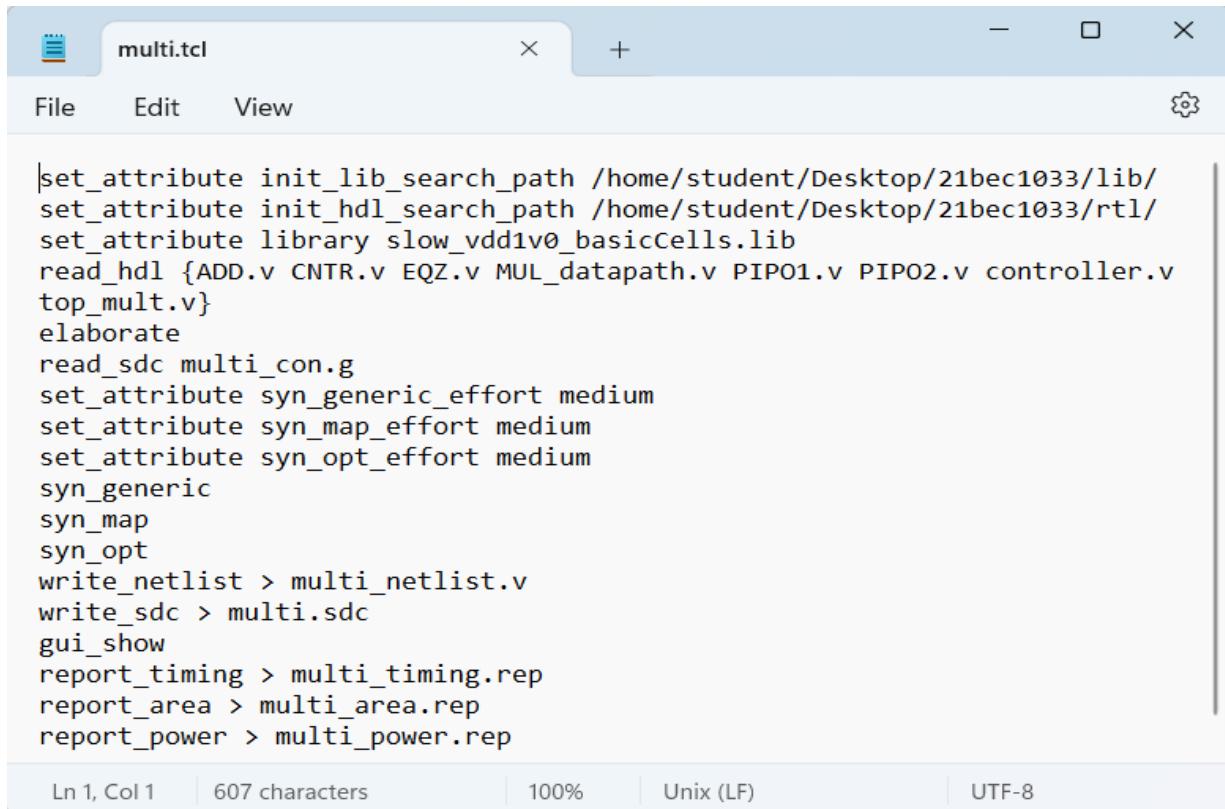
The screenshot shows a text editor window with the title bar 'multi_con.g'. The menu bar includes 'File', 'Edit', 'View', and a gear icon. The main text area contains the following Synopsys design constraint code:

```
create_clock -name clk -period 10 -waveform {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 1.0 [get_ports "clk"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "start"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "data_in"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "done"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "data_out"] -clock [get_clocks "clk"]
```

At the bottom of the window, status indicators show 'Ln 4, Col 45', '571 characters', '100%', 'Unix (LF)', and 'UTF-8'.

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

3. Create a (.tcl) file containing all the commands for performing the logic synthesis. Synthesis effort can be medium or high.



```
|set_attribute init_lib_search_path /home/student/Desktop/21bec1033/lib/
|set_attribute init_hdl_search_path /home/student/Desktop/21bec1033/rtl/
|set_attribute library slow_vdd1v0_basicCells.lib
|read_hdl {ADD.v CNTR.v EQZ.v MUL_datapath.v PIPO1.v PIPO2.v controller.v
|top_mult.v}
|elaborate
|read_sdc multi_con.g
|set_attribute syn_generic_effort medium
|set_attribute syn_map_effort medium
|set_attribute syn_opt_effort medium
|syn_generic
|syn_map
|syn_opt
|write_netlist > multi_netlist.v
|write_sdc > multi.sdc
|gui_show
|report_timing > multi_timing.rep
|report_area > multi_area.rep
|report_power > multi_power.rep
```

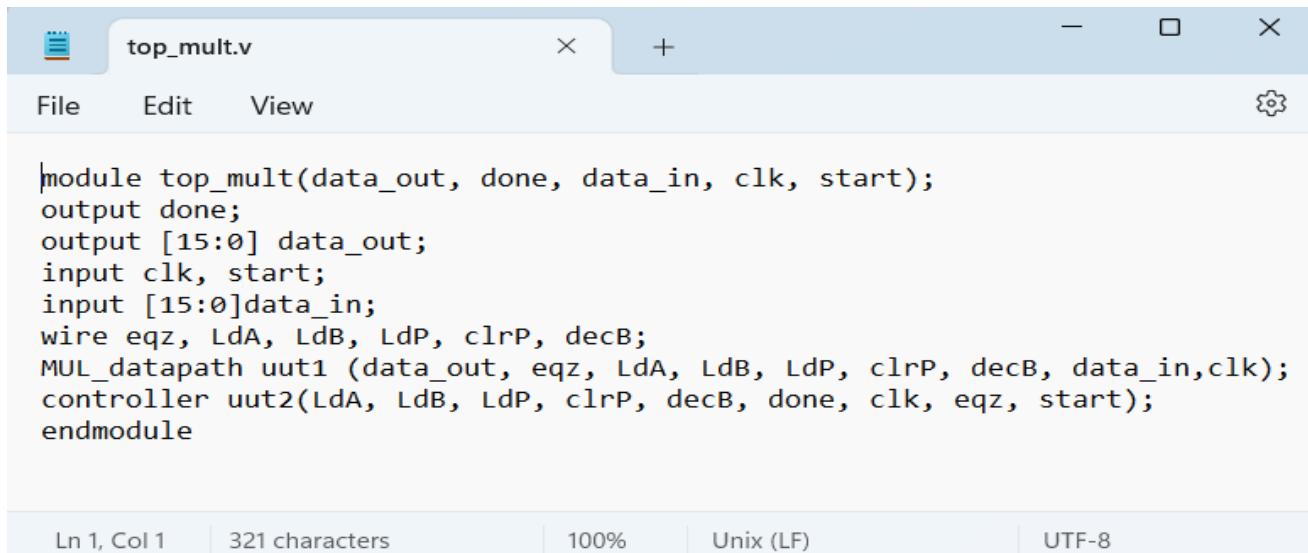
Ln 1, Col 1 | 607 characters | 100% | Unix (LF) | UTF-8

4. Invoke the C shell and launch the Genus tool by entering the below commands.
‘csh’
‘source /home/install/cshrc’
‘genus -legacy -ui -f multi.tcl’
5. Execute the commands in the (.tcl file) by entering **source multiplier.tcl** command in the command line window.
6. Check for the area, timing and power reports generated in the respective multiplier folder. Also check the gate level netlist generated in the Genus synthesis solution window.

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

Verilog Programs:

//Verilog Program of top-level module containing data path and control path

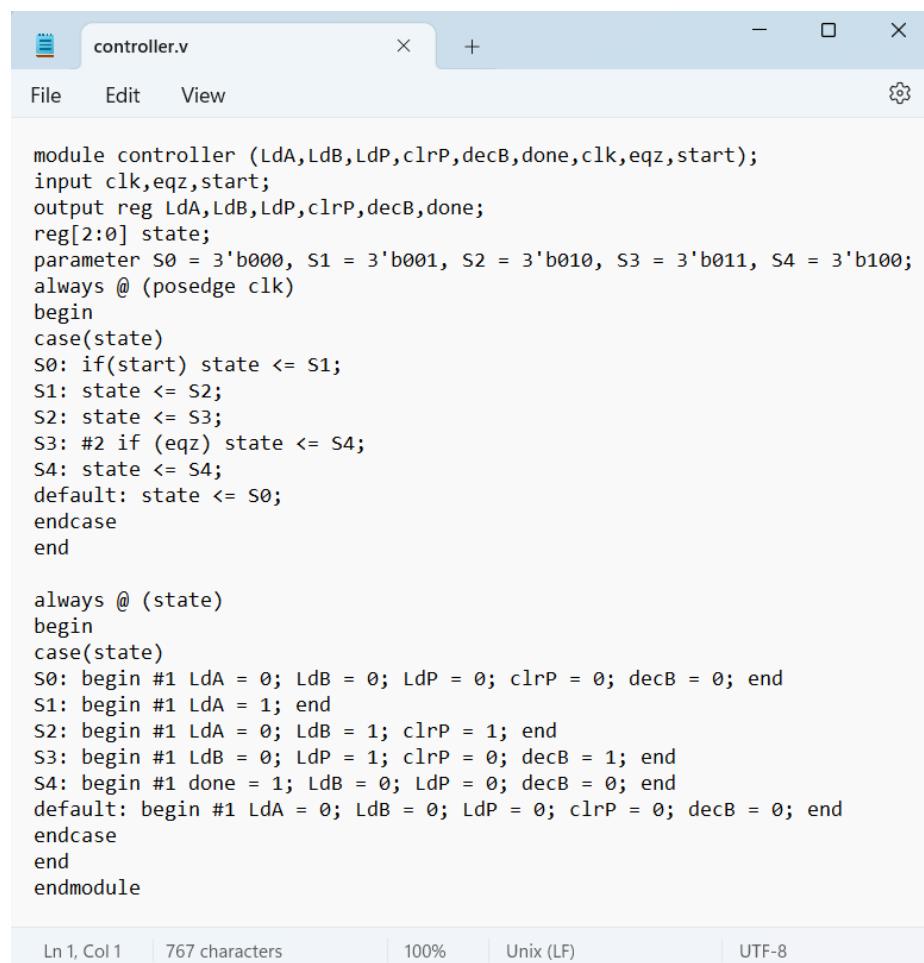


The screenshot shows a Verilog code editor window titled "top_mult.v". The code defines a module "top_mult" with the following interface:

```
module top_mult(data_out, done, data_in, clk, start);
output done;
output [15:0] data_out;
input clk, start;
input [15:0]data_in;
wire eqz, LdA, LdB, LdP, clrP, decB;
MUL_datapath uut1 (data_out, eqz, LdA, LdB, LdP, clrP, decB, data_in,clk);
controller uut2(LdA, LdB, LdP, clrP, decB, done, clk, eqz, start);
endmodule
```

The status bar at the bottom indicates: Ln 1, Col 1 | 321 characters | 100% | Unix (LF) | UTF-8

//Verilog Program of control path module



The screenshot shows a Verilog code editor window titled "controller.v". The code defines a module "controller" with the following interface:

```
module controller (LdA,LdB,LdP,clrP,decB,done,clk,eqz,start);
input clk,eqz,start;
output reg LdA,LdB,LdP,clrP,decB,done;
reg[2:0] state;
parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;
always @ (posedge clk)
begin
  case(state)
    S0: if(start) state <= S1;
    S1: state <= S2;
    S2: state <= S3;
    S3: #2 if (eqz) state <= S4;
    S4: state <= S0;
  endcase
end

always @ (state)
begin
  case(state)
    S0: begin #1 LdA = 0; LdB = 0; LdP = 0; clrP = 0; decB = 0; end
    S1: begin #1 LdA = 1; end
    S2: begin #1 LdA = 0; LdB = 1; clrP = 1; end
    S3: begin #1 LdB = 0; LdP = 1; clrP = 0; decB = 1; end
    S4: begin #1 done = 1; LdB = 0; LdP = 0; decB = 0; end
  endcase
end
endmodule
```

The status bar at the bottom indicates: Ln 1, Col 1 | 767 characters | 100% | Unix (LF) | UTF-8

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

//Verilog Program of data path module

The screenshot shows a code editor window titled "MUL_datapath.v". The code defines a module "MUL_datapath" with various inputs and outputs, including control signals like LdA, LdB, LdP, clrP, decB, and data_in, clk. It includes an always block for handling eqz, and several PIP01, PIP02, CNTR, ADD, and EQZ COMP components. The code is written in standard Verilog syntax.

```
module MUL_datapath(data_out,eqz,LdA,LdB,LdP,clrP,decB,data_in,clk);
input LdA,LdB,LdP,clrP,decB,clk;
input [15:0] data_in;
output eqz;
output reg [15:0] data_out;
wire [15:0] X,Y,Z,Bout;
always @ (eqz)
begin
if(eqz == 1)
data_out <= Y;
else
data_out <= data_out;
end

PIPO1 A (X,data_in,LdA,clk);
PIPO2 P (Y,Z,LdP,clrP,clk);
CNTR B (Bout,data_in,LdB,decB,clk);
ADD AD (Z,X,Y);
EQZ COMP (eqz, Bout);
endmodule
```

Ln 1, Col 1 | 411 characters | 100% | Unix (LF) | UTF-8

//Verilog Program of PIP01

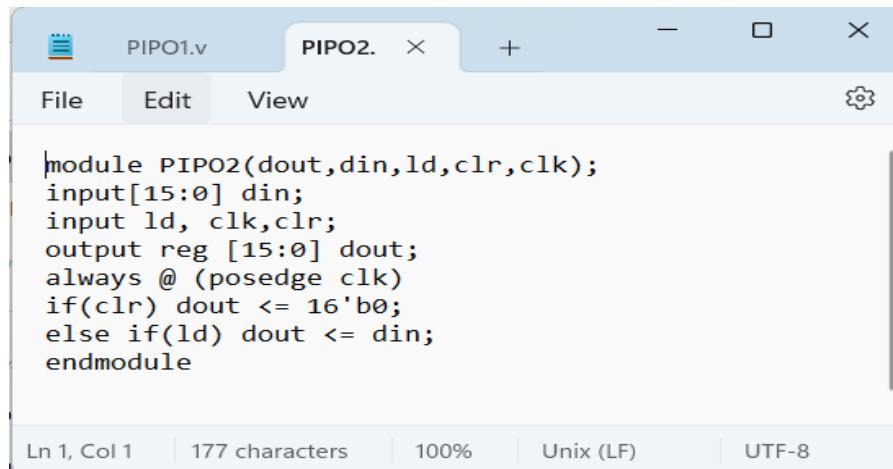
The screenshot shows a code editor window titled "PIP01.v". The code defines a module "PIP01" with inputs din, ld, and clk, and an output dout. It uses a simple if-else statement to assign dout to din if ld is asserted. The code is written in standard Verilog syntax.

```
module PIP01(dout,din,ld,clk);
input[15:0] din;
input ld, clk;
output reg [15:0] dout;
always @ (posedge clk)
if(ld) dout <= din;
endmodule
```

Ln 1, Col 1 | 141 characters | 100% | Unix (LF) | UTF-8

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

//Verilog Program of PIPO2

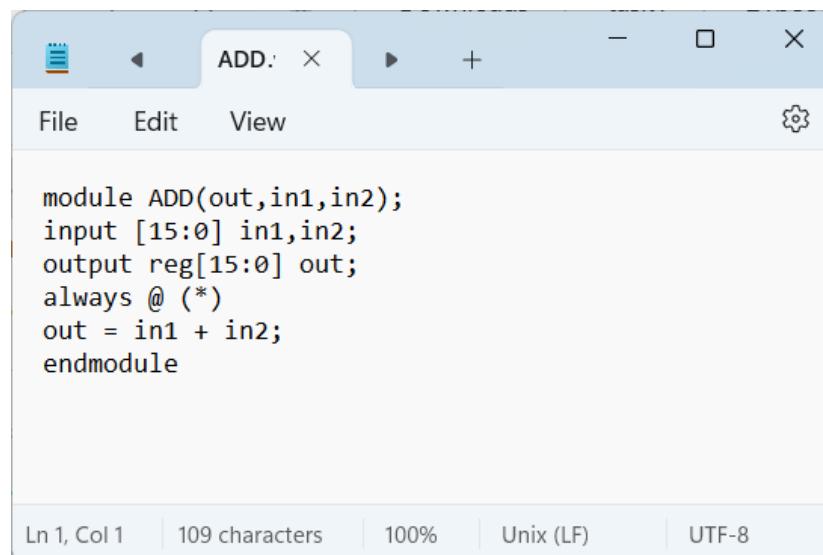


A screenshot of a Verilog editor window titled "PIPO2.v". The code in the editor is:

```
module PIPO2(dout,din,ld,clr,clk);
  input[15:0] din;
  input ld, clk,clr;
  output reg [15:0] dout;
  always @ (posedge clk)
    if(clr) dout <= 16'b0;
    else if(ld) dout <= din;
endmodule
```

The status bar at the bottom shows: Ln 1, Col 1 | 177 characters | 100% | Unix (LF) | UTF-8

//Verilog Program of Adder

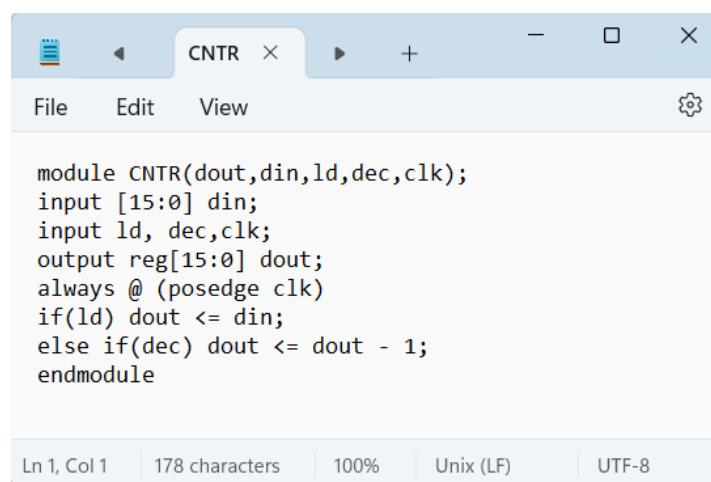


A screenshot of a Verilog editor window titled "ADD.v". The code in the editor is:

```
module ADD(out,in1,in2);
  input [15:0] in1,in2;
  output reg[15:0] out;
  always @ (*)
    out = in1 + in2;
endmodule
```

The status bar at the bottom shows: Ln 1, Col 1 | 109 characters | 100% | Unix (LF) | UTF-8

//Verilog Program of Counter



A screenshot of a Verilog editor window titled "CNTR.v". The code in the editor is:

```
module CNTR(dout,din,ld,dec,clk);
  input [15:0] din;
  input ld, dec,clk;
  output reg[15:0] dout;
  always @ (posedge clk)
    if(ld) dout <= din;
    else if(dec) dout <= dout - 1;
endmodule
```

The status bar at the bottom shows: Ln 1, Col 1 | 178 characters | 100% | Unix (LF) | UTF-8

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

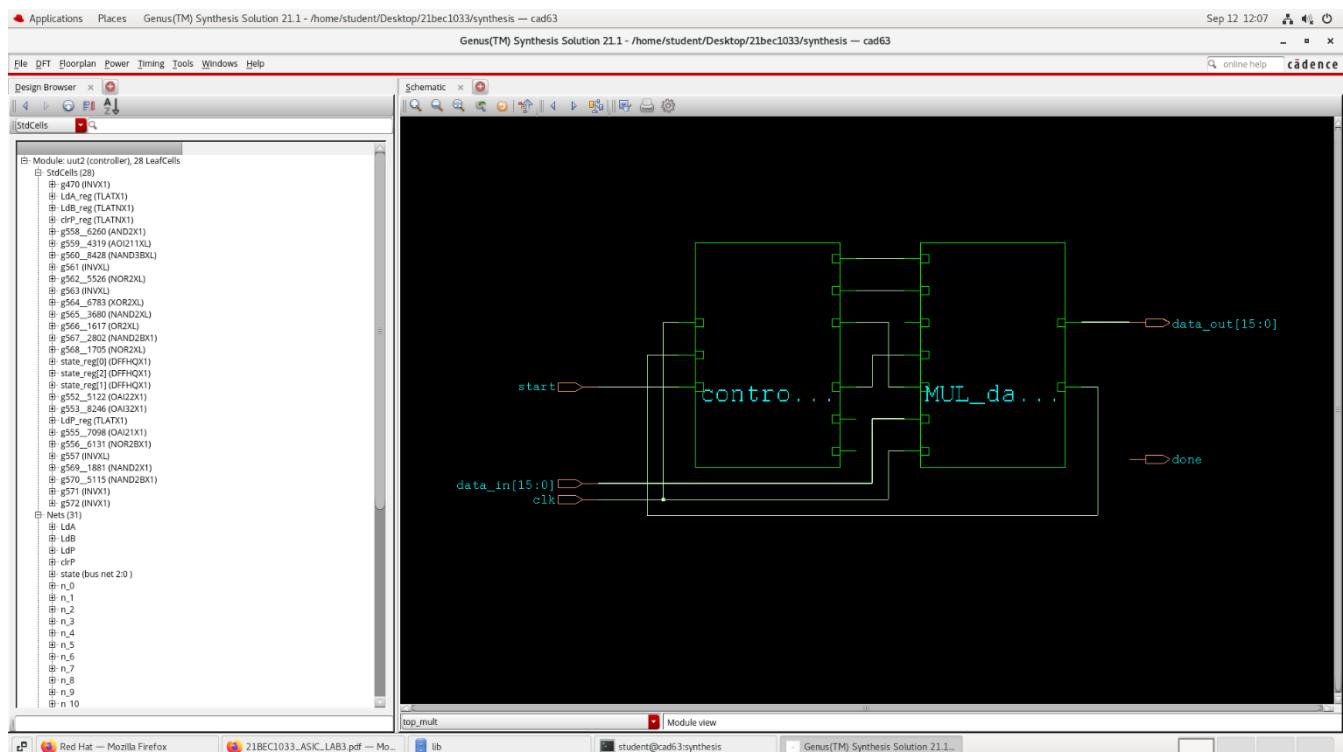
//Verilog Program of Comparator

The screenshot shows a Verilog code editor window titled "EQZ.v". The code defines a module EQZ with an input "data" of width 15:0 and an output "eqz". The output is assigned the value of 1 if the input is 0, and 0 otherwise. The editor interface includes tabs for File, Edit, View, and a gear icon for settings. At the bottom, status bars show "Ln 1, Col 1", "89 characters", "100%", "Unix (LF)", and "UTF-8".

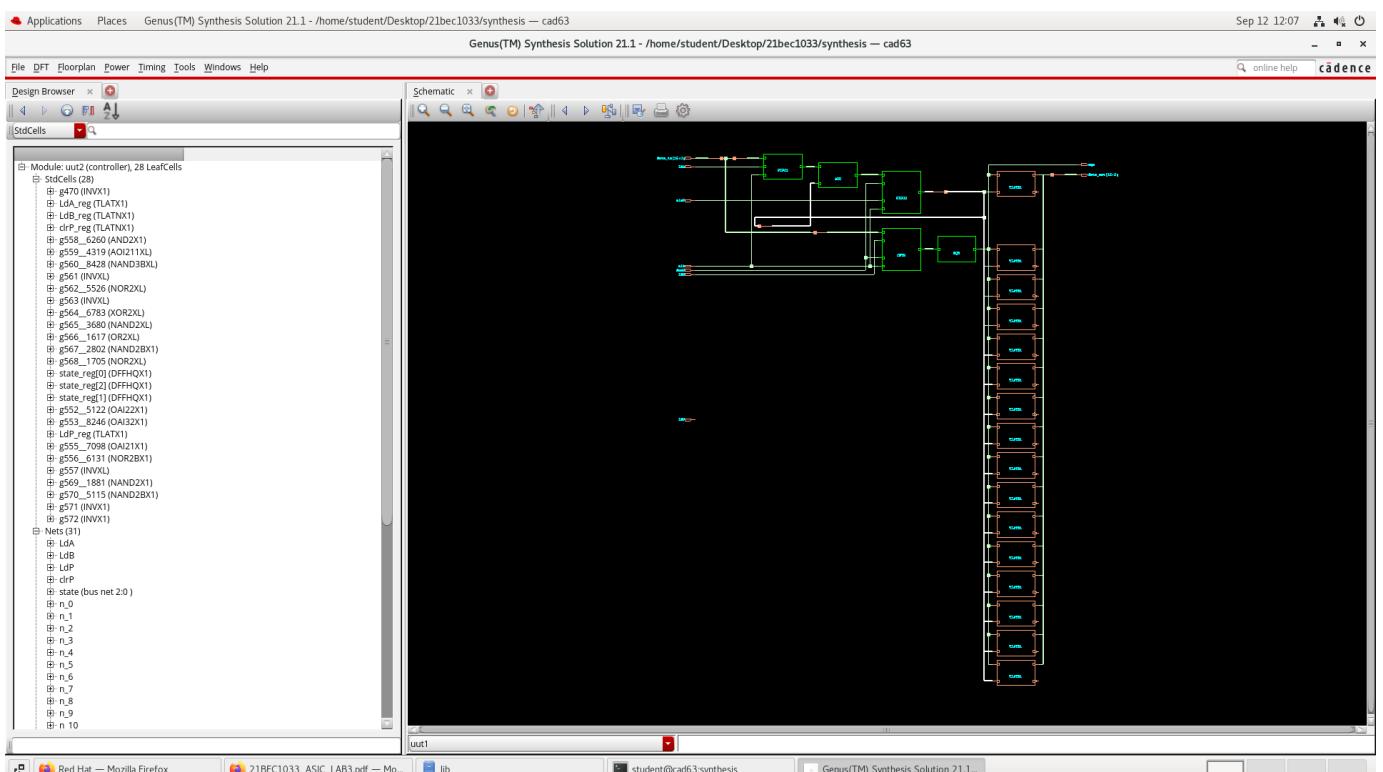
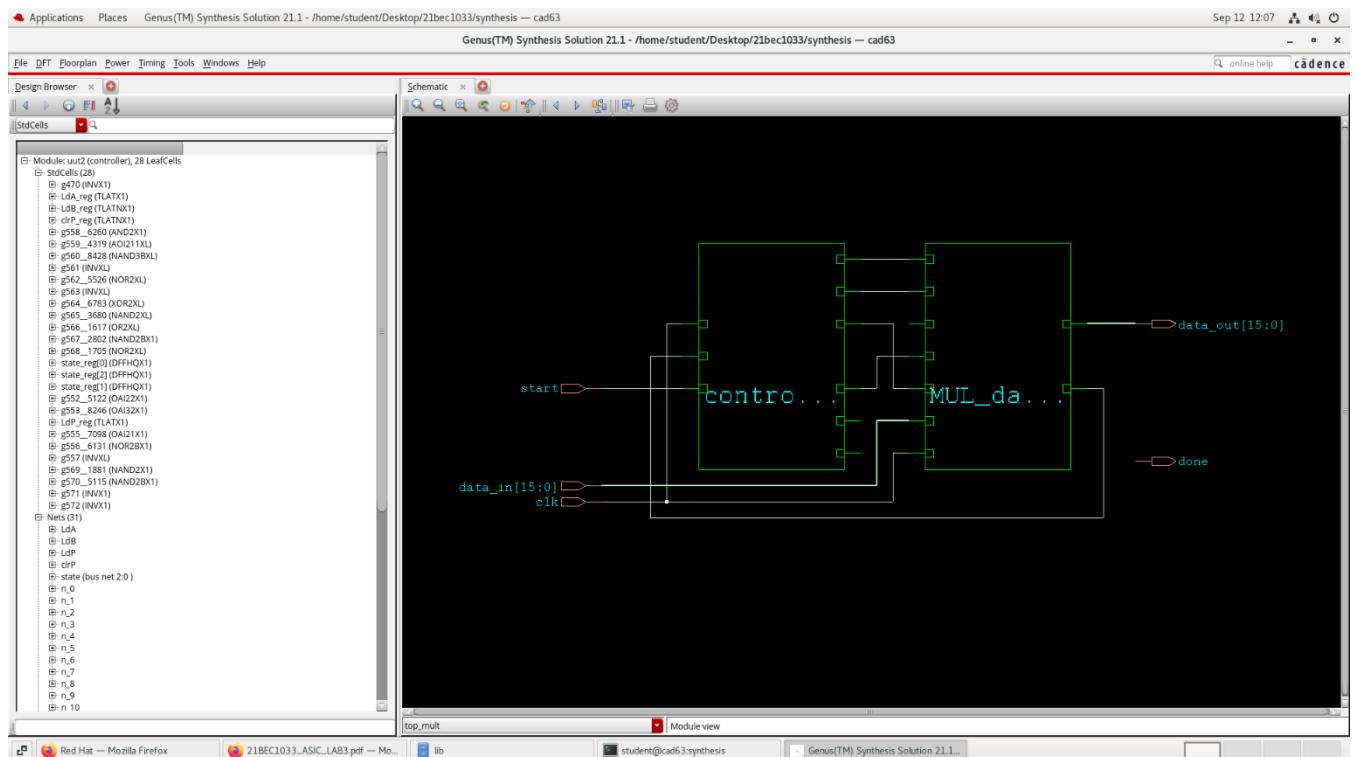
```
module EQZ(eqz,data);
  input [15:0] data;
  output eqz;
  assign eqz = (data == 0);
endmodule
```

Gate level Netlist:

Synthesis effort: Medium

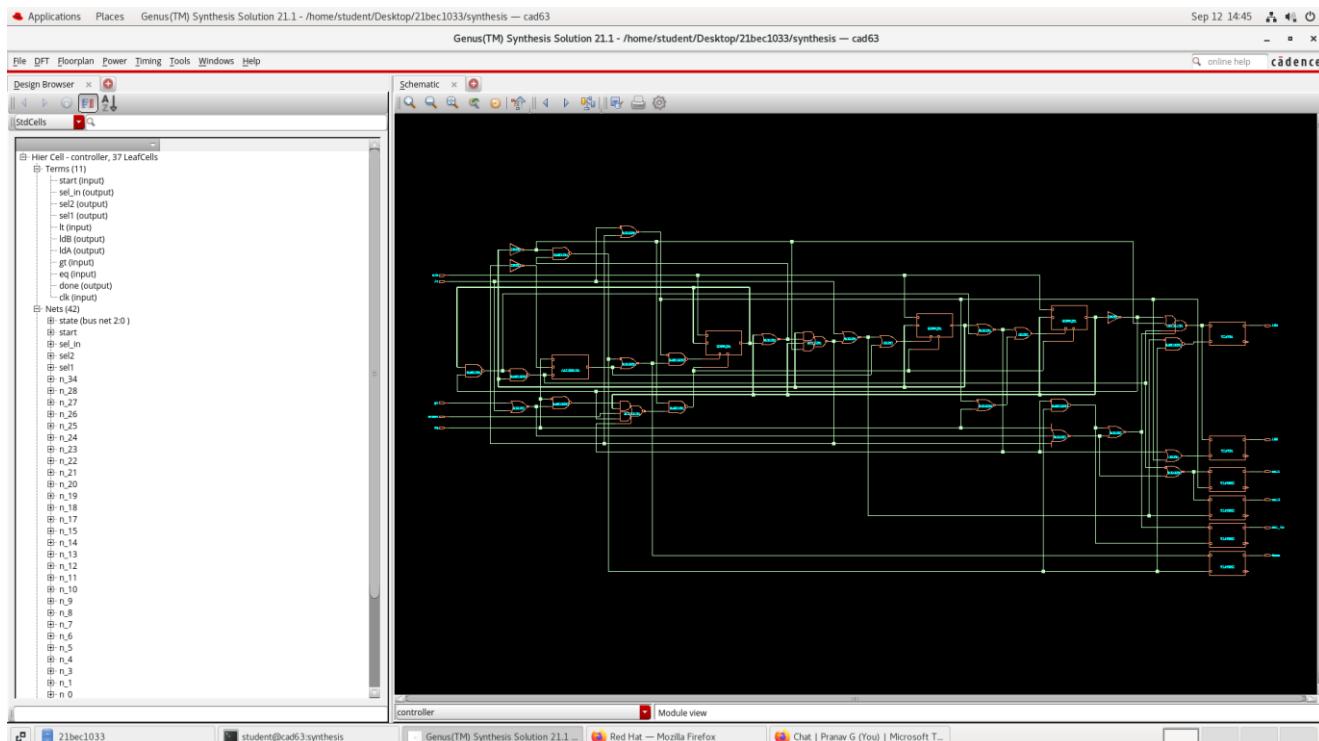


LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic



LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

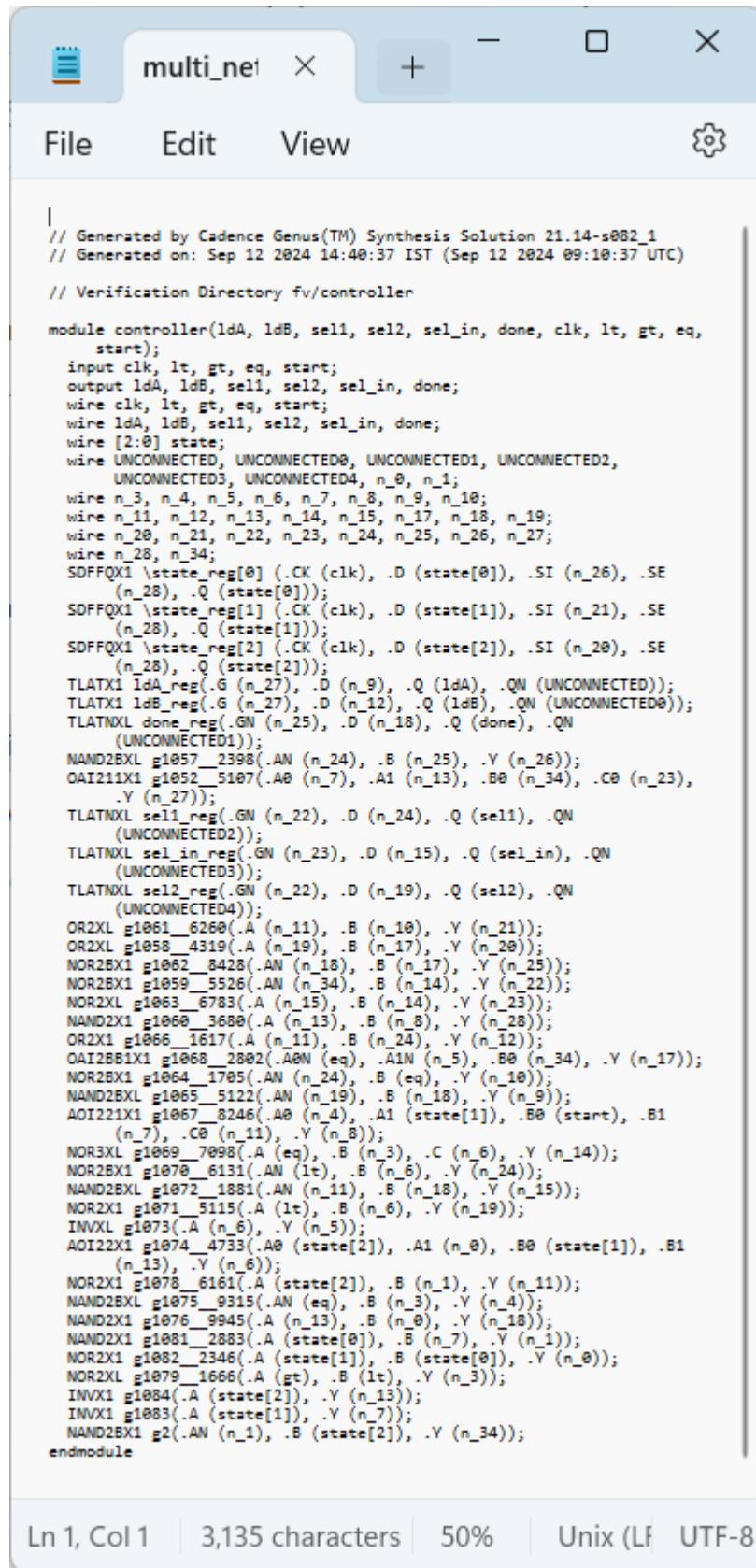
Synthesis effort : High



LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

Observations:

a) multiplier_netlist_created file



The screenshot shows a Cadence Genus Synthesis Solution window titled "multi_net". The window contains a code editor with the following content:

```
// Generated by Cadence Genus(TM) Synthesis Solution 21.14-s082_1
// Generated on: Sep 12 2024 14:40:37 IST (Sep 12 2024 09:10:37 UTC)

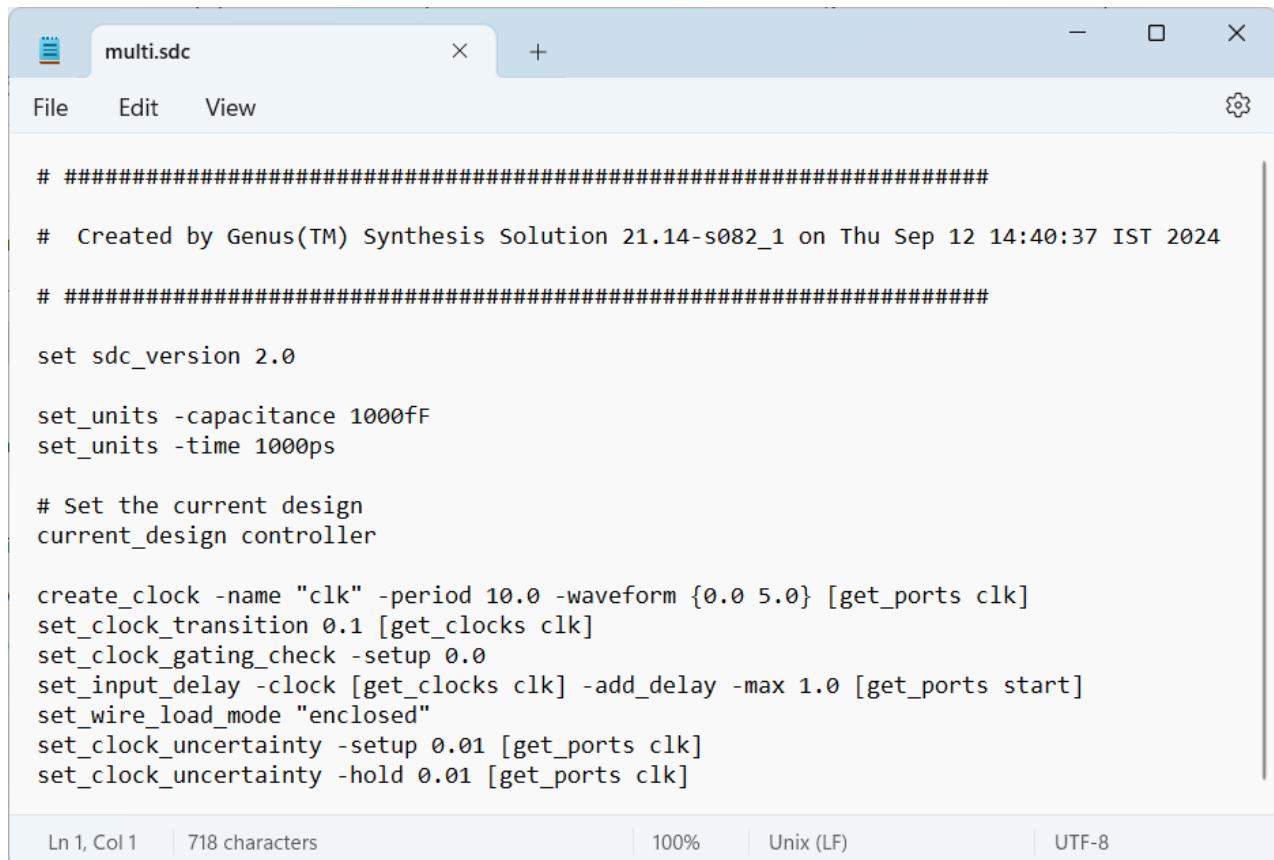
// Verification Directory fv/controller

module controller(1dA, 1dB, sel1, sel2, sel_in, done, clk, lt, gt, eq,
    start);
    input clk, lt, gt, eq, start;
    output 1dA, 1dB, sel1, sel2, sel_in, done;
    wire clk, lt, gt, eq, start;
    wire 1dA, 1dB, sel1, sel2, sel_in, done;
    wire [2:0] state;
    wire UNCONNECTED, UNCONNECTED0, UNCONNECTED1, UNCONNECTED2,
        UNCONNECTED3, UNCONNECTED4, n_0, n_1;
    wire n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_10;
    wire n_11, n_12, n_13, n_14, n_15, n_17, n_18, n_19;
    wire n_20, n_21, n_22, n_23, n_24, n_25, n_26, n_27;
    wire n_28, n_34;
    SDFFQX1 \state_reg[0] (.CK (clk), .D (state[0]), .SI (n_26), .SE
        (n_28), .Q (state[0]));
    SDFFQX1 \state_reg[1] (.CK (clk), .D (state[1]), .SI (n_21), .SE
        (n_28), .Q (state[1]));
    SDFFQX1 \state_reg[2] (.CK (clk), .D (state[2]), .SI (n_20), .SE
        (n_28), .Q (state[2]));
    TLATX1 1dA_reg(.G (n_27), .D (n_9), .Q (1dA), .QN (UNCONNECTED));
    TLATX1 1dB_reg(.G (n_27), .D (n_12), .Q (1dB), .QN (UNCONNECTED0));
    TLATNX1 done_reg(.GN (n_25), .D (n_18), .Q (done), .QN
        (UNCONNECTED1));
    NAND2BXL g1057_2398(.AN (n_24), .B (n_25), .Y (n_26));
    OAI211X1 g1052_5107(.A0 (n_7), .A1 (n_13), .B0 (n_34), .C0 (n_23),
        .Y (n_27));
    TLATNX1 sel1_reg(.GN (n_22), .D (n_24), .Q (sel1), .QN
        (UNCONNECTED2));
    TLATNX1 sel_in_reg(.GN (n_23), .D (n_15), .Q (sel_in), .QN
        (UNCONNECTED3));
    TLATNX1 sel2_reg(.GN (n_22), .D (n_19), .Q (sel2), .QN
        (UNCONNECTED4));
    OR2XL g1061_6260(.A (n_11), .B (n_18), .Y (n_21));
    OR2XL g1058_4319(.A (n_19), .B (n_17), .Y (n_28));
    NOR2BX1 g1062_8428(.AN (n_18), .B (n_17), .Y (n_25));
    NOR2BX1 g1059_5526(.AN (n_34), .B (n_14), .Y (n_22));
    NOR2XL g1063_6783(.A (n_15), .B (n_14), .Y (n_23));
    NAND2X1 g1060_3680(.A (n_13), .B (n_8), .Y (n_28));
    OR2X1 g1066_1617(.A (n_11), .B (n_24), .Y (n_12));
    OAI2BB1X1 g1068_2882(.A0N (eq), .A1N (n_5), .B0 (n_34), .Y (n_17));
    NOR2BX1 g1064_1785(.AN (n_24), .B (eq), .Y (n_18));
    NAND2BXL g1065_5122(.AN (n_19), .B (n_18), .Y (n_9));
    AOI221X1 g1067_8246(.A0 (n_4), .A1 (state[1]), .B0 (start), .B1
        (n_7), .C0 (n_11), .Y (n_8));
    NOR3XL g1069_7098(.A (eq), .B (n_3), .C (n_6), .Y (n_14));
    NOR2BX1 g1070_6131(.AN (lt), .B (n_6), .Y (n_24));
    NAND2BX1 g1072_1881(.AN (n_11), .B (n_18), .Y (n_15));
    NOR2X1 g1071_5115(.A (lt), .B (n_6), .Y (n_19));
    INVXL g1073(.A (n_6), .Y (n_5));
    AOI22X1 g1074_4733(.A0 (state[2]), .A1 (n_8), .B0 (state[1]), .B1
        (n_13), .Y (n_6));
    NOR2X1 g1078_6161(.A (state[2]), .B (n_1), .Y (n_11));
    NAND2BXL g1075_9315(.AN (eq), .B (n_3), .Y (n_4));
    NAND2X1 g1076_9945(.A (n_13), .B (n_8), .Y (n_18));
    NAND2X1 g1081_2883(.A (state[0]), .B (n_7), .Y (n_1));
    NOR2X1 g1082_2346(.A (state[1]), .B (state[0]), .Y (n_0));
    NOR2XL g1079_1666(.A (gt), .B (lt), .Y (n_3));
    INVX1 g1084(.A (state[2]), .Y (n_13));
    INVX1 g1083(.A (state[1]), .Y (n_7));
    NAND2BX1 g2(.AN (n_1), .B (state[2]), .Y (n_34));
endmodule
```

At the bottom of the window, the status bar shows: Ln 1, Col 1 | 3,135 characters | 50% | Unix (LF | UTF-8)

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

b) multiplier_constraint_created file



The screenshot shows a text editor window titled "multi.sdc". The file contains SDC (Setup/Constraints) script code used for logic synthesis. The code includes comments about the creation by Genus(TM) Synthesis Solution, setting the SDC version to 2.0, defining units for capacitance and time, setting the current design to "controller", creating a clock named "clk" with a period of 10.0, and specifying various timing constraints like setup and hold times for the "start" port.

```
# #####  
# Created by Genus(TM) Synthesis Solution 21.14-s082_1 on Thu Sep 12 14:40:37 IST 2024  
# #####  
  
set sdc_version 2.0  
  
set_units -capacitance 1000fF  
set_units -time 1000ps  
  
# Set the current design  
current_design controller  
  
create_clock -name "clk" -period 10.0 -waveform {0.0 5.0} [get_ports clk]  
set_clock_transition 0.1 [get_clocks clk]  
set_clock_gating_check -setup 0.0  
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports start]  
set_wire_load_mode "enclosed"  
set_clock_uncertainty -setup 0.01 [get_ports clk]  
set_clock_uncertainty -hold 0.01 [get_ports clk]
```

Ln 1, Col 1 | 718 characters | 100% | Unix (LF) | UTF-8

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

c) multiplier_timing_created file

Synthesis effort: Medium

```
multi_tim X - + ⚙
File Edit View ⚙

=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 12:05:27 pm
Module: top_mult
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Pin Type Fanout Load Slew Delay Arrival
----- (ff) (ps) (ps) (ps)
-----
```

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(ff)	(ps)	(ps)	(ps)	
(clock clk)	launch					0 R
uut1						
A						
dout_reg[0]/CK						
dout_reg[0]/Q	SDFFQX1	3	0.6	24	+225	225 R
A/dout[0]						
AD/in1[0]						
add_5_11_g383_7482/A						+0 225
add_5_11_g383_7482/Y	AND2XL	2	1.0	49	+124	349 R
add_5_11_g380_6131/CI						+0 349
add_5_11_g380_6131/CO	ADDFX1	1	0.6	39	+191	540 R
add_5_11_g379_7098/CI						+0 540
add_5_11_g379_7098/CO	ADDFX1	1	0.6	39	+186	726 R
add_5_11_g378_8246/CI						+0 726
add_5_11_g378_8246/CO	ADDFX1	1	0.6	39	+186	912 R
add_5_11_g377_5122/CI						+0 912
add_5_11_g377_5122/CO	ADDFX1	1	0.6	39	+186	1098 R
add_5_11_g376_1705/CI						+0 1098
add_5_11_g376_1705/CO	ADDFX1	1	0.6	39	+186	1284 R
add_5_11_g375_2802/CI						+0 1284
add_5_11_g375_2802/CO	ADDFX1	1	0.6	39	+186	1470 R
add_5_11_g374_1617/CI						+0 1470
add_5_11_g374_1617/CO	ADDFX1	1	0.6	39	+186	1656 R
add_5_11_g373_3680/CI						+0 1656
add_5_11_g373_3680/CO	ADDFX1	1	0.6	39	+186	1842 R
add_5_11_g372_6783/CI						+0 1842
add_5_11_g372_6783/CO	ADDFX1	1	0.6	39	+186	2028 R
add_5_11_g371_5526/CI						+0 2028
add_5_11_g371_5526/CO	ADDFX1	1	0.6	39	+186	2214 R
add_5_11_g370_8428/CI						+0 2214
add_5_11_g370_8428/CO	ADDFX1	1	0.6	39	+186	2400 R
add_5_11_g369_4319/CI						+0 2400
add_5_11_g369_4319/CO	ADDFX1	1	0.6	39	+186	2586 R
add_5_11_g368_6268/CI						+0 2586
add_5_11_g368_6268/CO	ADDFX1	1	0.6	39	+186	2772 R
add_5_11_g367_5107/CI						+0 2772
add_5_11_g367_5107/CO	ADDFX1	1	0.4	35	+184	2956 R
add_5_11_g366_2398/B						+0 2956
add_5_11_g366_2398/Y	XNOR2X1	1	0.4	26	+125	3081 R
AD/out[15]						
P/din[15]						
g80_5477/B1						+0 3081
g80_5477/Y	AOI22X1	1	0.3	105	+57	3138 F
g64_5122/B						+0 3138
g64_5122/Y	NOR2X1	1	0.3	49	+84	3223 R
dout_reg[15]/D	<<< DFFHQX1					+0 3223
dout_reg[15]/CK	setup				100	+101 3323 R
(clock clk)	capture					10000 R
	uncertainty					-10 9990 R
Cost Group : 'clk' (path_group 'clk')						
Timing slack : 6667ps						
Start-point : uut1/A/dout_reg[0]/CK						
End-point : uut1/P/dout_reg[15]/D						

Ln 1, Col 1 | 4,446 characters | 50% | Unix (LF) | UTF-8

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

Synthesis effort: High

```
multi_timing.rep
File Edit View
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 02:40:37 pm
Module: controller
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Pin Type Fanout Load Slew Delay Arrival
(fF) (ps) (ps) (ps)
-----
(clock clk) launch 0 R
(multi_con.g_line_6) ext delay +1000 1000 F
start in port 1 0.4 0 +0 1000 F
g1067_8246/B0 +0 1000
g1067_8246/Y AOI221X1 1 0.4 101 +110 1110 R
g1060_3680/B +0 1110
g1060_3680/Y NAND2X1 3 1.2 106 +121 1230 F
state_reg[2]/SE <<< SDFFQX1 +0 1230
state_reg[2]/CK setup 100 +233 1464 R
----- (clock clk) capture 10000 R
uncertainty -10 9990 R
-----
Cost Group : 'clk' (path_group 'clk')
Timing slack : 8526ps
Start-point : start
End-point : state_reg[2]/SE

Ln 1, Col 1 | 1,740 characters | 100% | Unix (LF) | UTF-8
```

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

d) multiplier_area_created file

Synthesis effort: Medium

```
multi_area.rep * + File Edit View
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 12:05:27 pm
Module: top_mult
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----
top_mult 180 671.688 0.000 671.688 <none> (D)
  uut1 MUL_datapath 152 604.998 0.000 604.998 <none> (D)
    A PIPO1 16 120.384 0.000 120.384 <none> (D)
    AD ADD 18 80.028 0.000 80.028 <none> (D)
    B CNTR 48 169.632 0.000 169.632 <none> (D)
    COMP EQZ 5 9.918 0.000 9.918 <none> (D)
    P PIPO2 49 137.484 0.000 137.484 <none> (D)
  uut2 controller 28 66.690 0.000 66.690 <none> (D)
(D) = wireload is default in technology library
Ln 1, Col 1 | 1,311 characters | 100% | Unix (LF) | UTF-8
```

Synthesis effort: High

```
multi_area.rep * + File Edit View
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 02:40:37 pm
Module: controller
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----
controller 37 91.656 0.000 91.656 <none> (D)
(D) = wireload is default in technology library
Ln 1, Col 1 | 734 characters | 100% | Unix (LF) | UTF-8
```

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

e) multiplier_power_created file

Synthesis effort: Medium

```
multi_power.rep multi_power.rep × + ⌂
File Edit View ⌂
Instance: /top_mult
Power Unit: W
PDB Frames: /stim#0/frame#0

Category Leakage Internal Switching Total Row%
-----
memory 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
register 6.03735e-09 9.83431e-06 2.56520e-07 1.00969e-05 76.15%
latch 2.28749e-09 4.92714e-07 1.81550e-07 6.76552e-07 5.10%
logic 4.86856e-09 1.31326e-06 3.40639e-07 1.65877e-06 12.51%
bbox 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
clock 0.00000e+00 0.00000e+00 8.26200e-07 8.26200e-07 6.23%
pad 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
pm 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
-----
Subtotal 1.31934e-08 1.16403e-05 1.60491e-06 1.32584e-05 99.99%
Percentage 0.10% 87.80% 12.10% 100.00% 100.00%
-----
```

Ln 7, Col 76 | 1,202 characters | 100% | Unix (LF) | UTF-8

Synthesis effort: High

```
multi_power.rep multi_power.rep × + ⌂
File Edit View ⌂
Instance: /controller
Power Unit: W
PDB Frames: /stim#0/frame#0

Category Leakage Internal Switching Total Row%
-----
memory 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
register 3.54301e-10 1.40330e-06 1.10603e-07 1.51426e-06 40.98%
latch 5.57475e-10 5.95999e-07 0.00000e+00 5.96557e-07 16.14%
logic 8.32462e-10 1.04767e-06 4.87399e-07 1.53590e-06 41.56%
bbox 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
clock 0.00000e+00 0.00000e+00 4.86000e-08 4.86000e-08 1.32%
pad 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
pm 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
-----
Subtotal 1.74424e-09 3.04697e-06 6.46602e-07 3.69531e-06 100.00%
Percentage 0.05% 82.45% 17.50% 100.00% 100.00%
-----
```

Ln 1, Col 1 | 1,204 characters | 100% | Unix (LF) | UTF-8

LAB4 1: Logic synthesis of a multiplier by repeated addition using Datapath and Control path logic

Inference:

Synthesis effort: Medium

A total of 28 leaf instance count is present in the gate level netlist with total area of 671.688, total power of 1.32584e-05 W.

Synthesis effort: High

A total of 37 leaf instance count is present in the gate level netlist with total area of 91.656, total power of 3.69531e-06 W.

Result: Hence a multiplier by repeated addition using Datapath and Control path logic is synthesized and the gate level netlist with timing, area and power report has been generated.

Lab4 2: Design and simulation of a GCD computation using Datapath and Control path logic

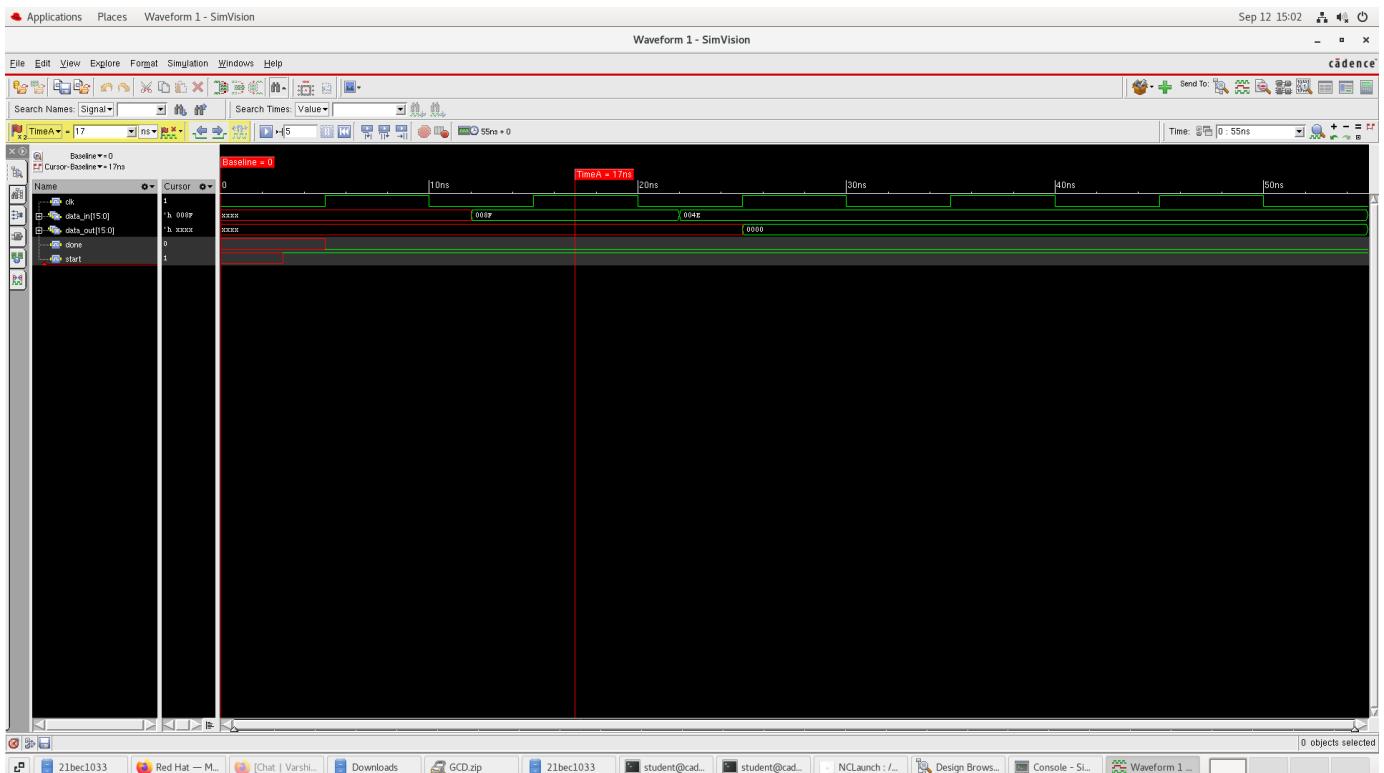
NCLaunch Compilation and Elaboration:

The screenshot shows the NCLaunch graphical user interface. The top menu bar includes File, Edit, Tools, Utilities, Plug-Ins, and Help. Below the menu is a toolbar with icons for Browsers, Tools, and various design functions. The main workspace displays a file tree on the left and a hierarchical module structure on the right. The terminal window at the bottom shows the command-line output of the ncclab session, detailing memory usage, CPU usage, and the execution of various modules (COMPARE, GCD_datapath, GCD_mult, MUX, PIP0, SUB) and their sub-modules (e.g., worklib.GCD_test.module). The bottom status bar shows open applications like 21bec1033, Red Hat - Mozilla Fi..., Chat, Downloads, GCD.zip, 21bec1033, and student@cad63:w.

```
ncelab: Memory Usage - 49.9M program + 29.9M data = 79.8M total (Peak 87.7M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.2% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.GCD_datapath
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.7M data = 82.6M total (Peak 87.6M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.3s, 9.7% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.GCD_mult
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.8M data = 82.7M total (Peak 87.7M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 10.2% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.GCD_test
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.8M data = 82.7M total (Peak 87.7M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 10.1% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.MUX
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.8M data = 82.8M total (Peak 87.7M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.5% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.PIP0
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.5M data = 82.4M total (Peak 87.7M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.5% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.SUB
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 29.9M data = 79.9M total (Peak 87.7M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 8.9% cpu)
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.controller
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.7M data = 82.6M total (Peak 87.6M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.8% cpu)
nclaunch> ncsm -gui -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncsm.log -errormax 15 -status worklib.GCD_testmodule
nclaunch> ncsm(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
```

Lab4 2: Design and simulation of a GCD computation using Datapath and Control path logic

Observations/Simulation Waveforms:



Inference: In the above output waveform, the data_in is provided with 143 & 78 values and the data_out is obtained as 13 which is the GCD. Output “done” is equal to 1 whenever gcd result is obtained.

Result: Hence, a GCD computer is designed and the simulation and verification of behavior of GCD computer using data path and Control path logic is performed for the given stimuli in the test bench.

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

Aim: To synthesize a GCD computer using Datapath and Control path logic and to get the gate level netlist with timing, area, and power reports.

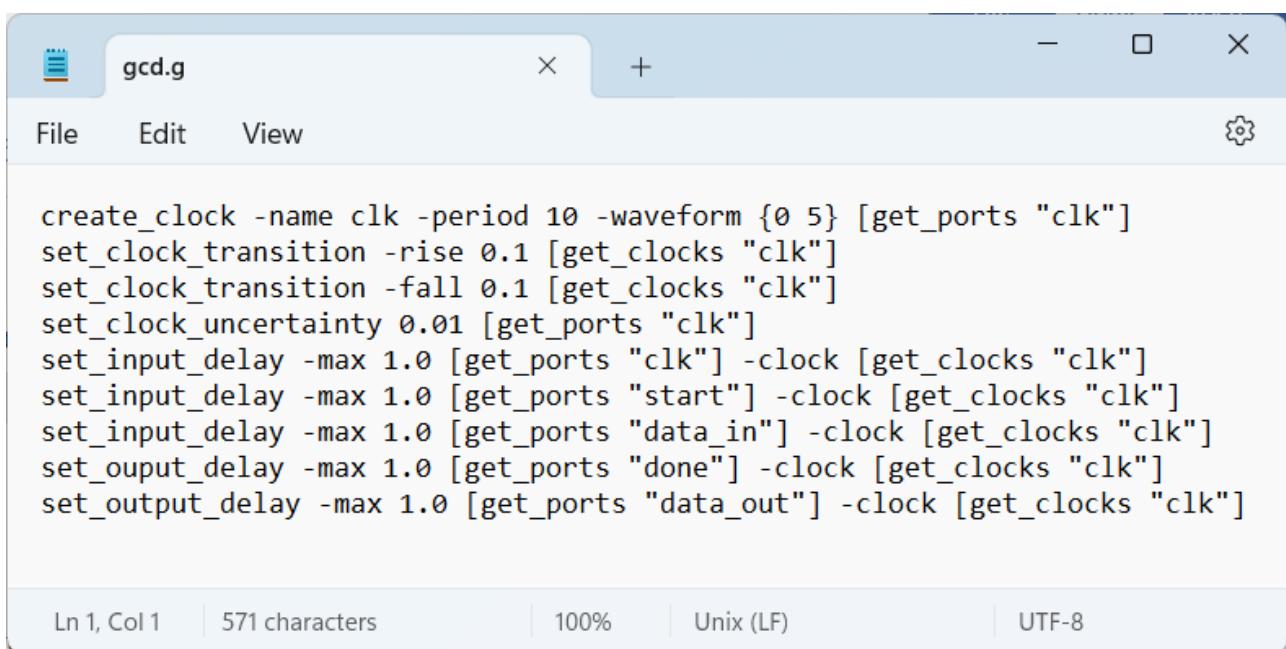
EDA Tools Used: Cadence Genus

Description:

Datapath is the hardware that performs all the required operations. Control is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc. Simple datapath components include memory (stores the current instruction), PC or program counter (stores the address of current instruction), and ALU (executes current instruction). Control path is the Finite state machine designed by using Moore or Mealy models.

Procedure:

1. Copy the fast.lib and slow.lib files into the folder where (.v) files are located.
2. Create a Synopsys design constraint file (.g file) by entering the design constraints required for the synthesis.

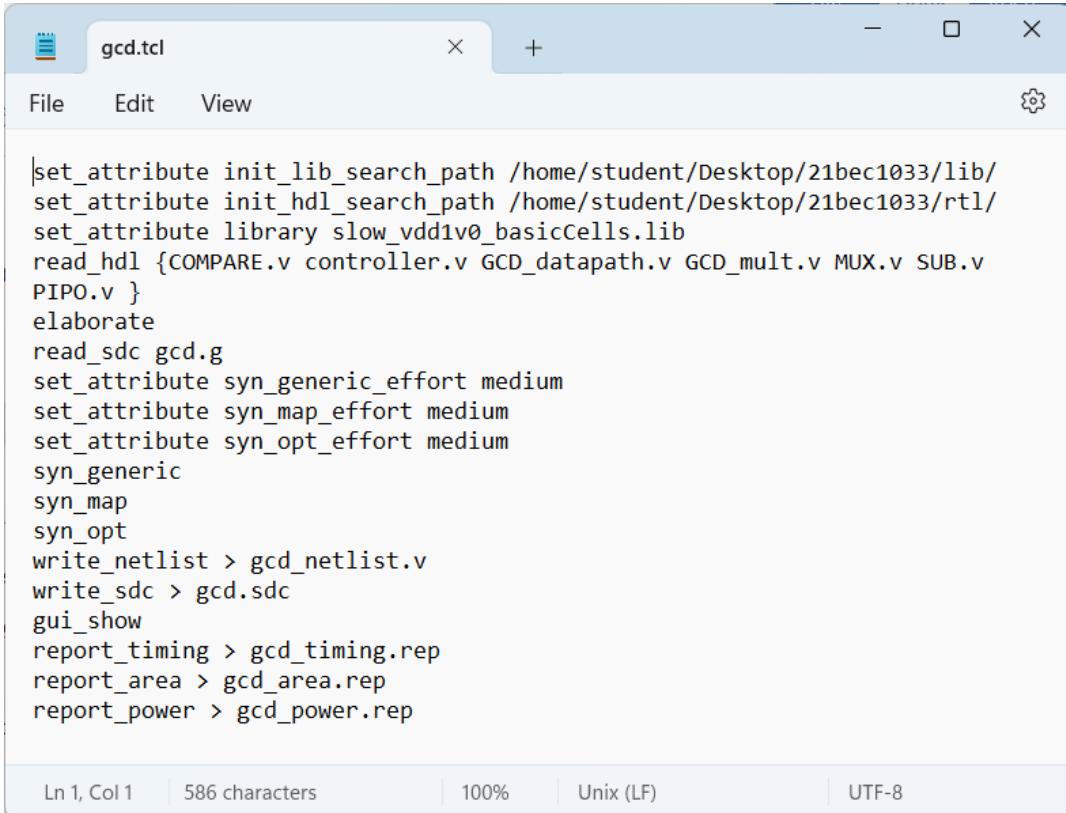


```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 0.01 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "clk"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "start"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "data_in"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "done"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "data_out"] -clock [get_clocks "clk"]
```

Ln 1, Col 1 | 571 characters | 100% | Unix (LF) | UTF-8

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

3. Create a (.tcl) file containing all the commands for performing the logic synthesis. Synthesis effort can be medium or high.



```
gcd.tcl

File Edit View
set_attribute init_lib_search_path /home/student/Desktop/21bec1033/lib/
set_attribute init_hdl_search_path /home/student/Desktop/21bec1033/rtl/
set_attribute library slow_vdd1v0_basicCells.lib
read_hdl {COMPARE.v controller.v GCD_datapath.v GCD_mult.v MUX.v SUB.v
PIPO.v }
elaborate
read_sdc gcd.sdc
set_attribute syn_generic_effort medium
set_attribute syn_map_effort medium
set_attribute syn_opt_effort medium
syn_generic
syn_map
syn_opt
write_netlist > gcd_netlist.v
write_sdc > gcd.sdc
gui_show
report_timing > gcd_timing.rep
report_area > gcd_area.rep
report_power > gcd_power.rep

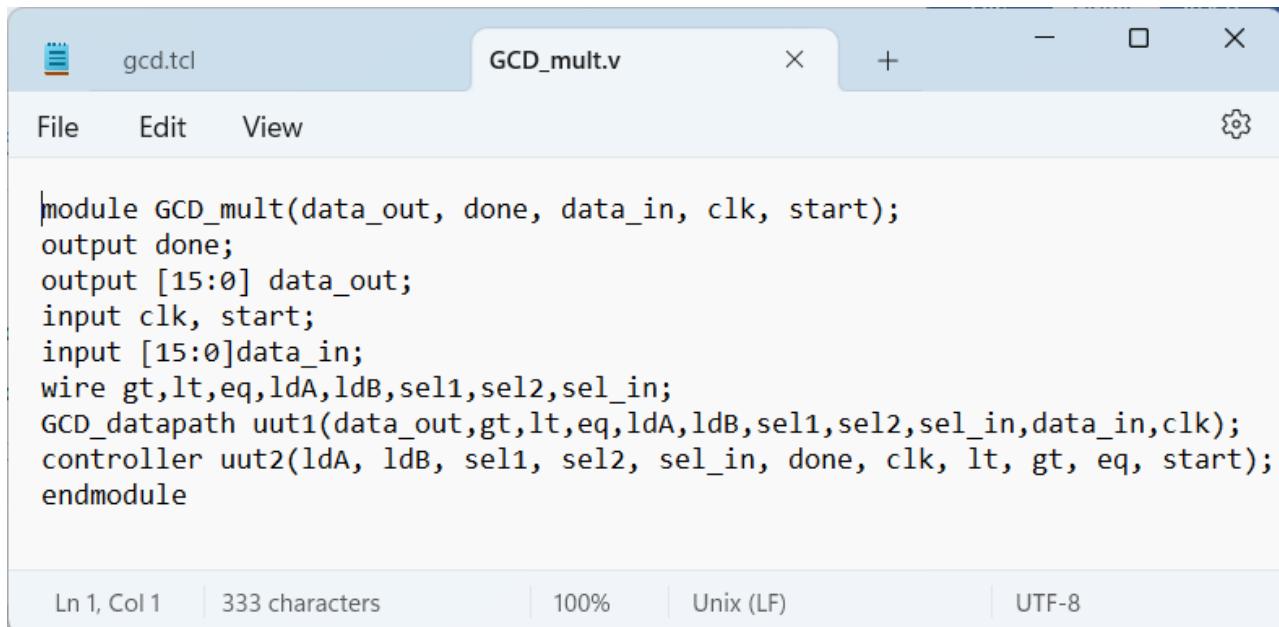
Ln 1, Col 1 | 586 characters | 100% | Unix (LF) | UTF-8
```

4. Invoke the C shell and launch the Genus tool by entering the below commands.
‘csh’
‘source /home/install/cshrc’
‘genus -legacy -ui -f gcd.tcl’
5. Execute the commands in the (.tcl file) by entering **source gcd.tcl** command in the command line window.
6. Check for the area, timing and power reports generated in the respective multiplier folder. Also check the gate level netlist generated in the Genus synthesis solution window.

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

Verilog Programs:

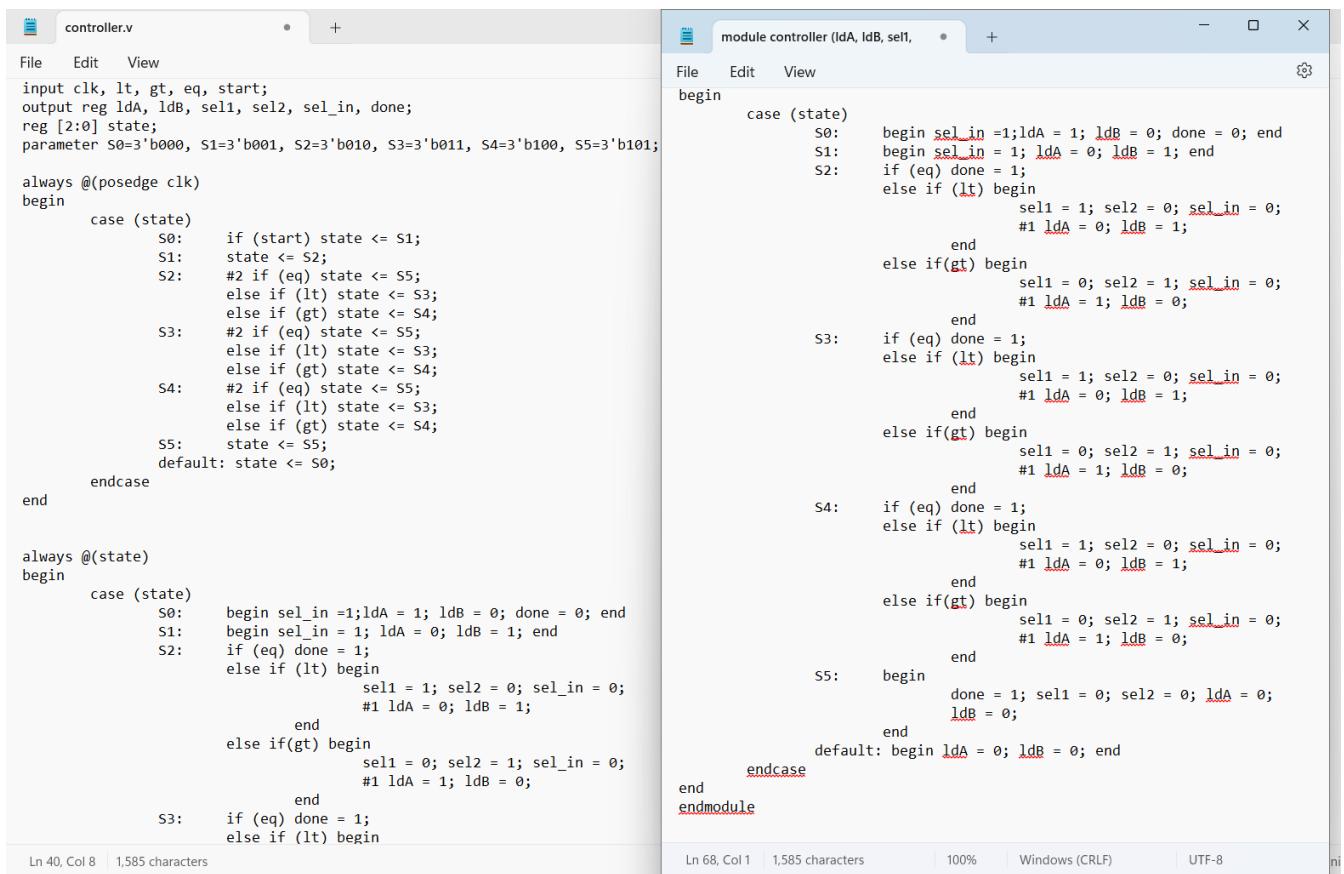
//Verilog Program of top-level module containing data path and control path



```
module GCD_mult(data_out, done, data_in, clk, start);
output done;
output [15:0] data_out;
input clk, start;
input [15:0]data_in;
wire gt,lt,eq,ldA,ldB,sel1,sel2,sel_in;
GCD_datapath uut1(data_out,gt,lt,eq,ldA,ldB,sel1,sel2,sel_in,data_in,clk);
controller uut2(ldA, ldB, sel1, sel2, sel_in, done, clk, lt, gt, eq, start);
endmodule
```

Ln 1, Col 1 | 333 characters | 100% | Unix (LF) | UTF-8

//Verilog Program of control path module



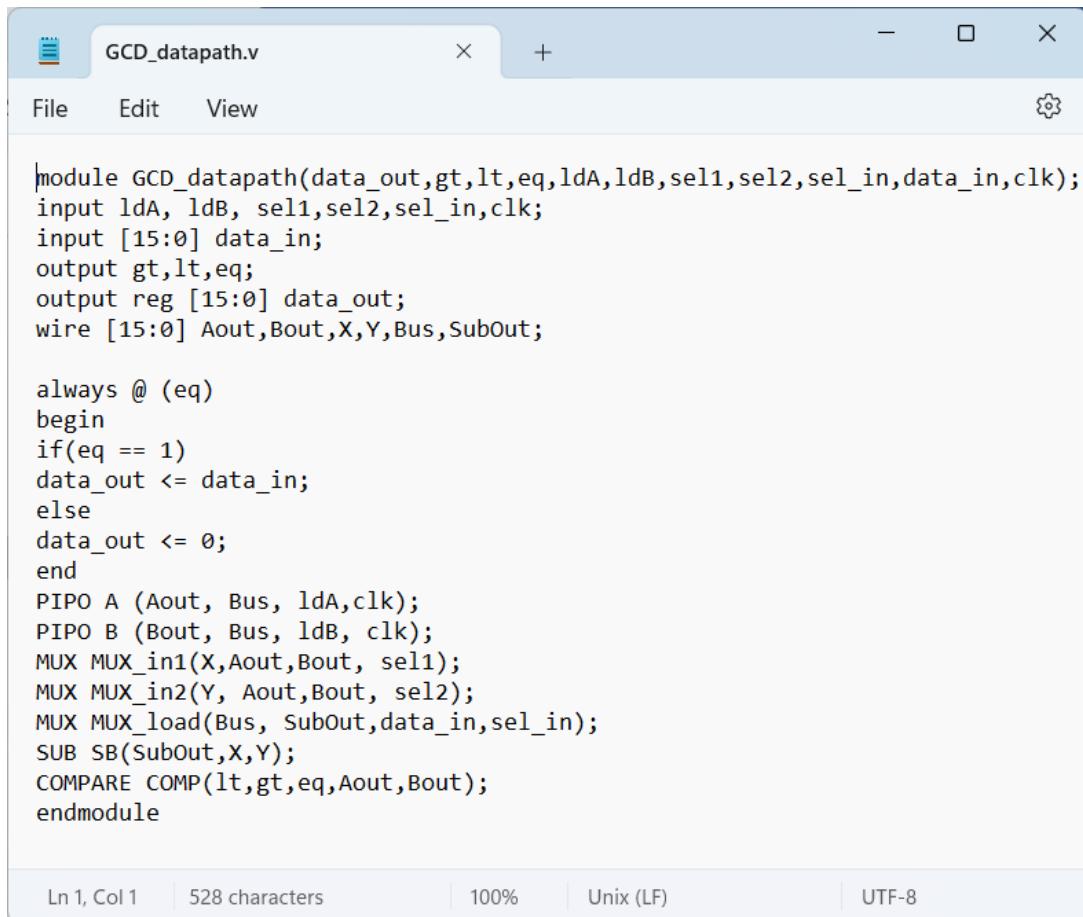
```
controller.v
```

```
module controller (ldA, ldB, sel1, sel2, sel_in, done);
input ldA, ldB;
output sel1, sel2, sel_in, done;
parameter S0=3'b000, S1=3'b001, S2=3'b010, S3=3'b011, S4=3'b100, S5=3'b101;
reg [2:0] state;
begin
    case (state)
        S0: begin sel_in = 1; ldA = 1; ldB = 0; done = 0; end
        S1: begin sel_in = 1; ldA = 0; ldB = 1; end
        S2: if (eq) done = 1;
            else if (lt) begin
                sel1 = 1; sel2 = 0; sel_in = 0;
                #1 ldA = 0; ldB = 1;
            end
            else if(gt) begin
                sel1 = 0; sel2 = 1; sel_in = 0;
                #1 ldA = 1; ldB = 0;
            end
        S3: if (eq) done = 1;
            else if (lt) begin
                sel1 = 1; sel2 = 0; sel_in = 0;
                #1 ldA = 0; ldB = 1;
            end
            else if(gt) begin
                sel1 = 0; sel2 = 1; sel_in = 0;
                #1 ldA = 1; ldB = 0;
            end
        S4: if (eq) done = 1;
            else if (lt) begin
                sel1 = 1; sel2 = 0; sel_in = 0;
                #1 ldA = 0; ldB = 1;
            end
            else if(gt) begin
                sel1 = 0; sel2 = 1; sel_in = 0;
                #1 ldA = 1; ldB = 0;
            end
        S5: begin
            if (eq) done = 1;
            else if (lt) begin
                sel1 = 0; sel2 = 0; sel_in = 0;
                #1 ldA = 0; ldB = 0;
            end
            else if(gt) begin
                sel1 = 0; sel2 = 1; sel_in = 0;
                #1 ldA = 1; ldB = 0;
            end
        end
    endcase
end
endmodule
```

Ln 40, Col 8 | 1,585 characters | 100% | Windows (CRLF) | UTF-8

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

//Verilog Program of data path module



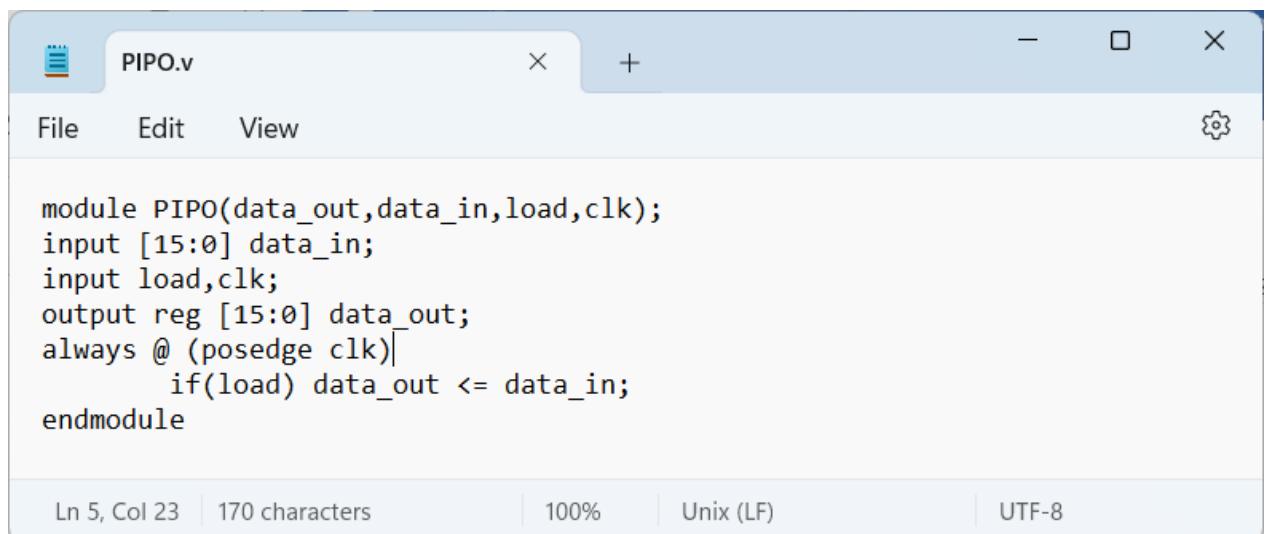
The screenshot shows a Verilog code editor window titled "GCD_datapath.v". The code defines a module GCD_datapath with various inputs (ldA, ldB, sel1, sel2, sel_in, data_in, clk) and outputs (gt, lt, eq, data_out). It includes a control logic section with an always block and a series of logic blocks: PIPO A, PIPO B, MUX MUX_in1, MUX MUX_in2, MUX MUX_load, and SUB SB. The code ends with a COMPARE block. The status bar at the bottom indicates the code length (528 characters), zoom level (100%), and encoding (UTF-8).

```
module GCD_datapath(data_out,gt,lt,eq,ldA,ldB,sel1,sel2,sel_in,data_in,clk);
input ldA, ldB, sel1,sel2,sel_in,clk;
input [15:0] data_in;
output gt,lt,eq;
output reg [15:0] data_out;
wire [15:0] Aout,Bout,X,Y,Bus,Subout;

always @ (eq)
begin
if(eq == 1)
data_out <= data_in;
else
data_out <= 0;
end
PIPO A (Aout, Bus, ldA,clk);
PIPO B (Bout, Bus, ldB, clk);
MUX MUX_in1(X,Aout,Bout, sel1);
MUX MUX_in2(Y, Aout,Bout, sel2);
MUX MUX_load(Bus, SubOut,data_in,sel_in);
SUB SB(SubOut,X,Y);
COMPARE COMP(lt,gt,eq,Aout,Bout);
endmodule

Ln 1, Col 1 | 528 characters | 100% | Unix (LF) | UTF-8
```

//Verilog Program of PIPO



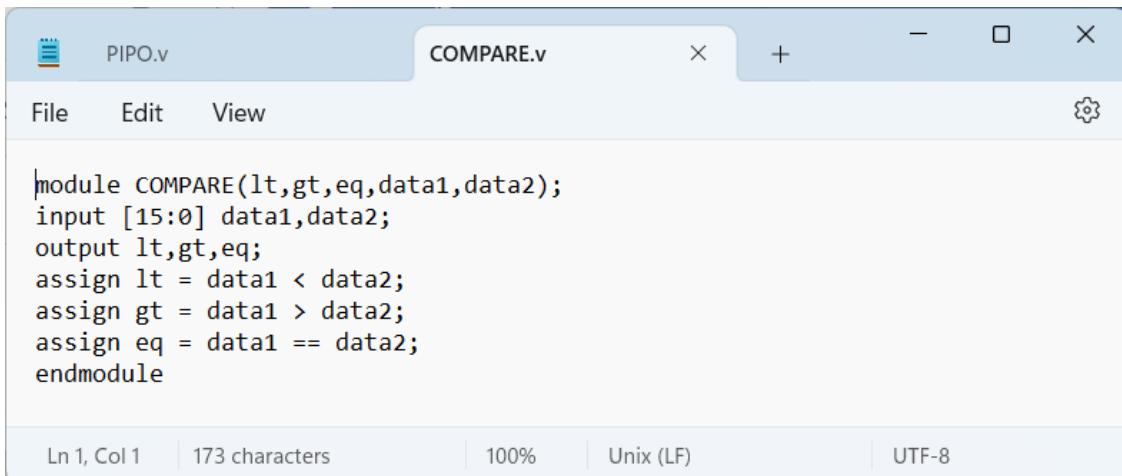
The screenshot shows a Verilog code editor window titled "PIPO.v". The code defines a module PIPO with inputs (data_in, load, clk) and output (data_out). It features an always block that updates data_out to data_in if load is asserted. The status bar at the bottom indicates the code length (170 characters), zoom level (100%), and encoding (UTF-8).

```
module PIPO(data_out,data_in,load,clk);
input [15:0] data_in;
input load,clk;
output reg [15:0] data_out;
always @ (posedge clk)
if(load) data_out <= data_in;
endmodule

Ln 5, Col 23 | 170 characters | 100% | Unix (LF) | UTF-8
```

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

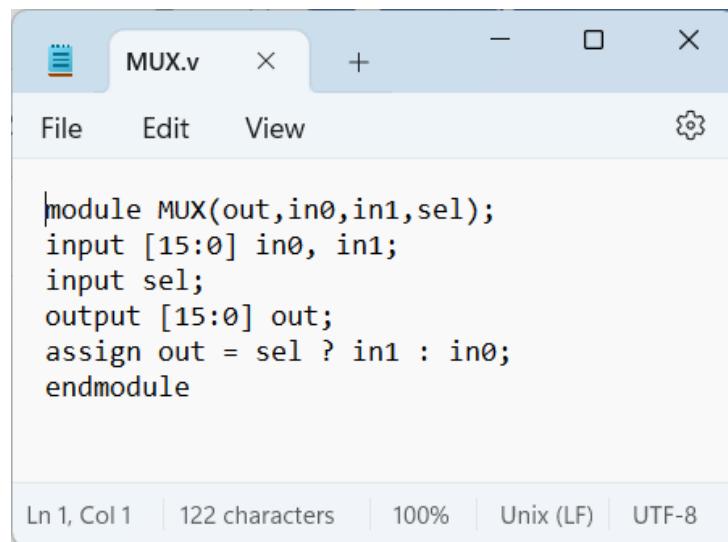
//Verilog Program of COMPARE



The screenshot shows a Verilog editor window titled "COMPARE.v". The code defines a module named "COMPARE" with four inputs: "lt", "gt", and "eq", and two output ports: "data1" and "data2". The logic is implemented using assignments based on comparison operators. The status bar at the bottom indicates the file is at Line 1, Column 1, with 173 characters, at 100% zoom, using Unix (LF) line endings, and in UTF-8 encoding.

```
module COMPARE(lt,gt,eq,data1,data2);
  input [15:0] data1,data2;
  output lt,gt,eq;
  assign lt = data1 < data2;
  assign gt = data1 > data2;
  assign eq = data1 == data2;
endmodule
```

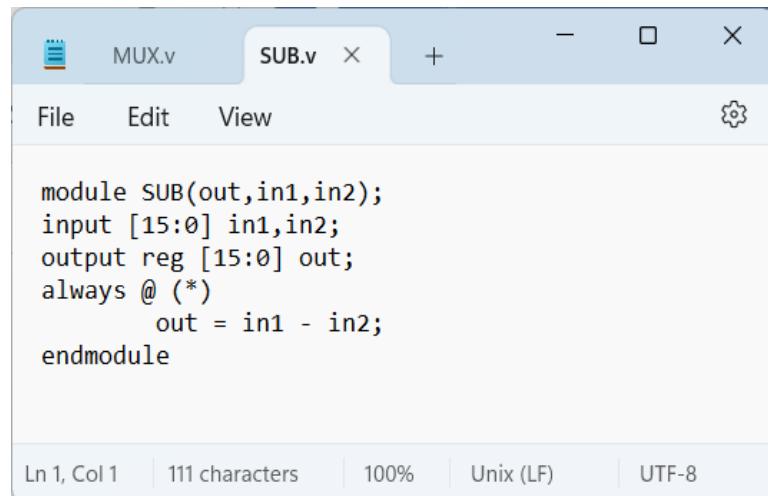
//Verilog Program of MUX



The screenshot shows a Verilog editor window titled "MUX.v". The code defines a module named "MUX" with three inputs: "in0", "in1", and "sel", and one output "out". The output is assigned based on the value of "sel". The status bar at the bottom indicates the file is at Line 1, Column 1, with 122 characters, at 100% zoom, using Unix (LF) line endings, and in UTF-8 encoding.

```
module MUX(out,in0,in1,sel);
  input [15:0] in0, in1;
  input sel;
  output [15:0] out;
  assign out = sel ? in1 : in0;
endmodule
```

//Verilog Program of SUB



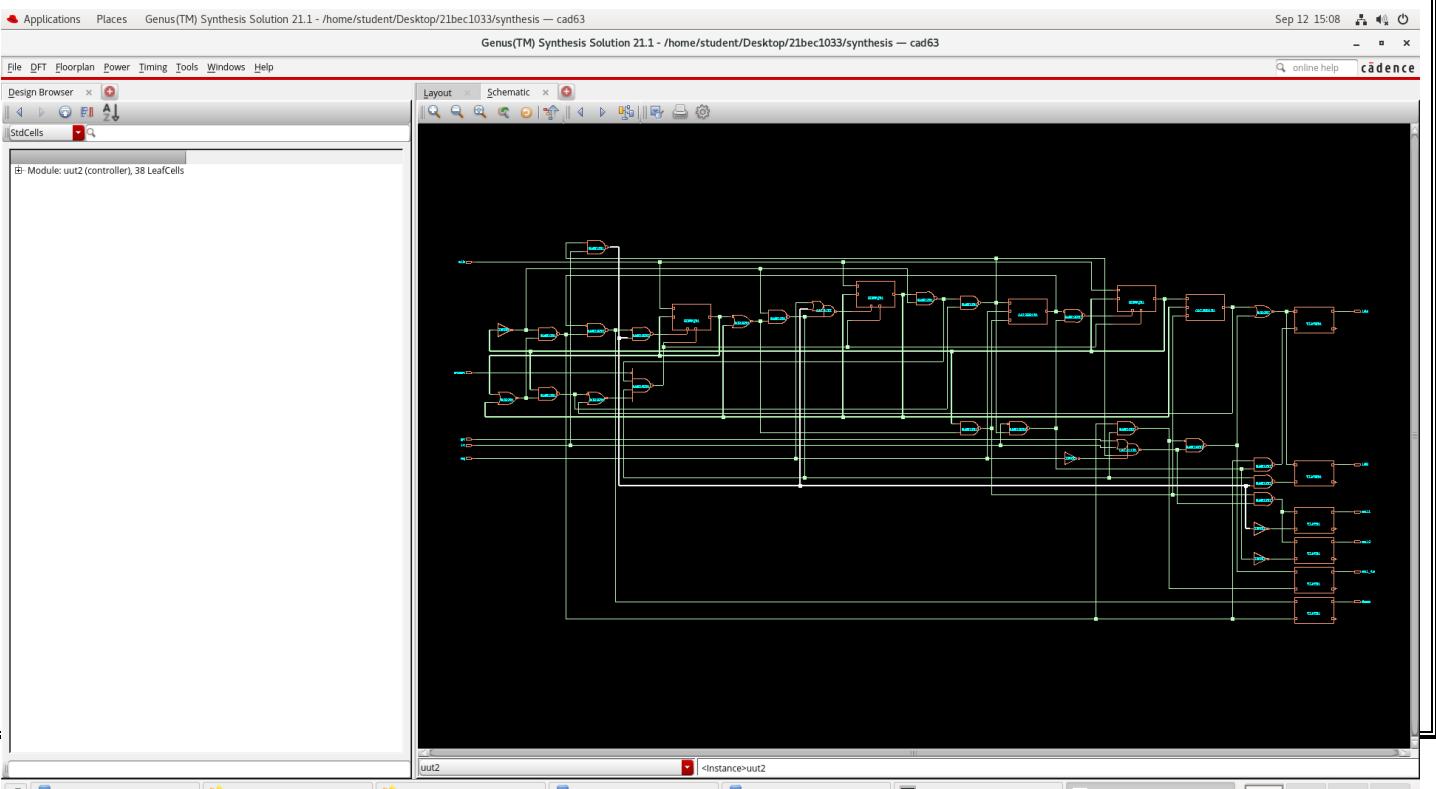
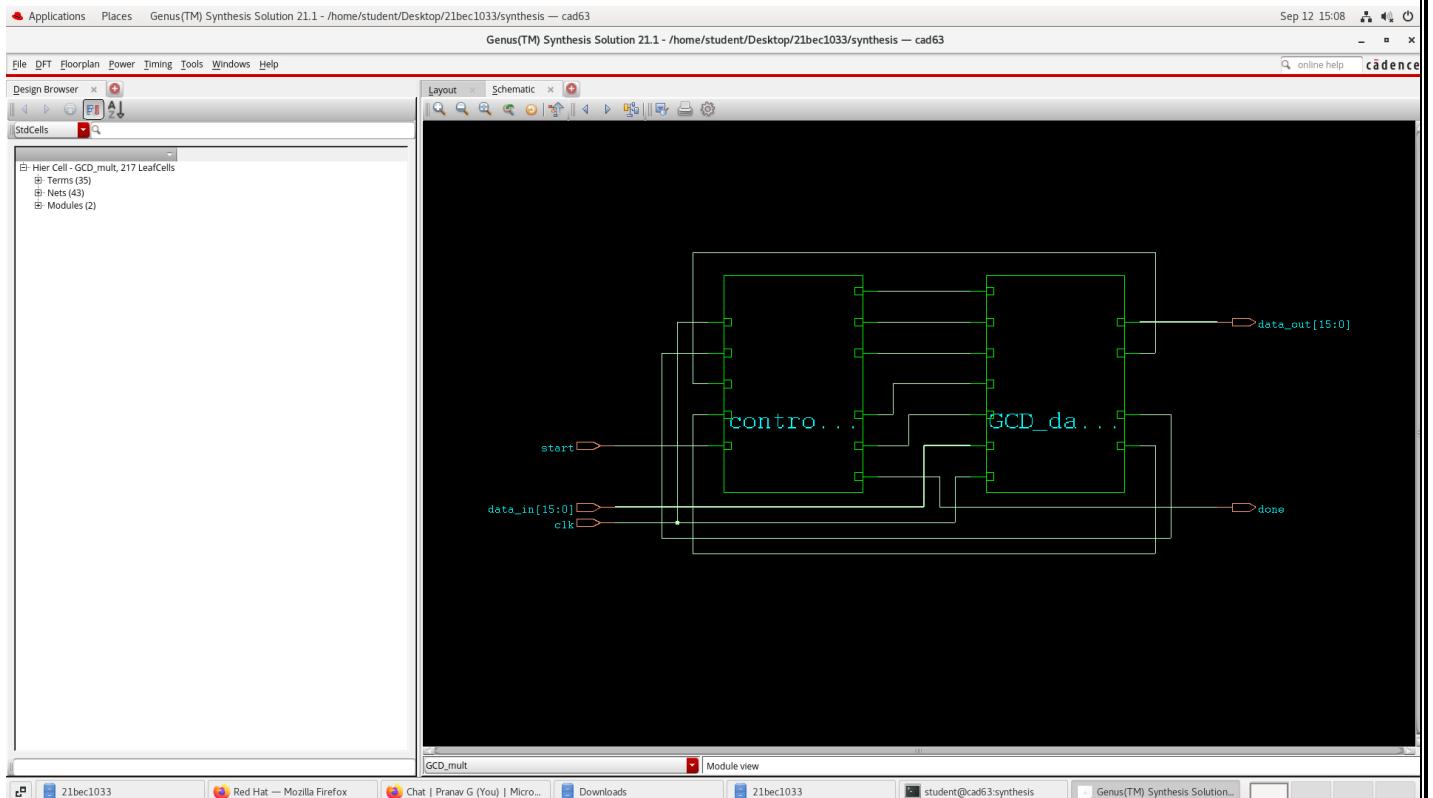
The screenshot shows a Verilog editor window titled "SUB.v". The code defines a module named "SUB" with two inputs: "in1" and "in2", and one output "out". The output is calculated as the difference between "in1" and "in2" using an always block. The status bar at the bottom indicates the file is at Line 1, Column 1, with 111 characters, at 100% zoom, using Unix (LF) line endings, and in UTF-8 encoding.

```
module SUB(out,in1,in2);
  input [15:0] in1,in2;
  output reg [15:0] out;
  always @ (*)
    out = in1 - in2;
endmodule
```

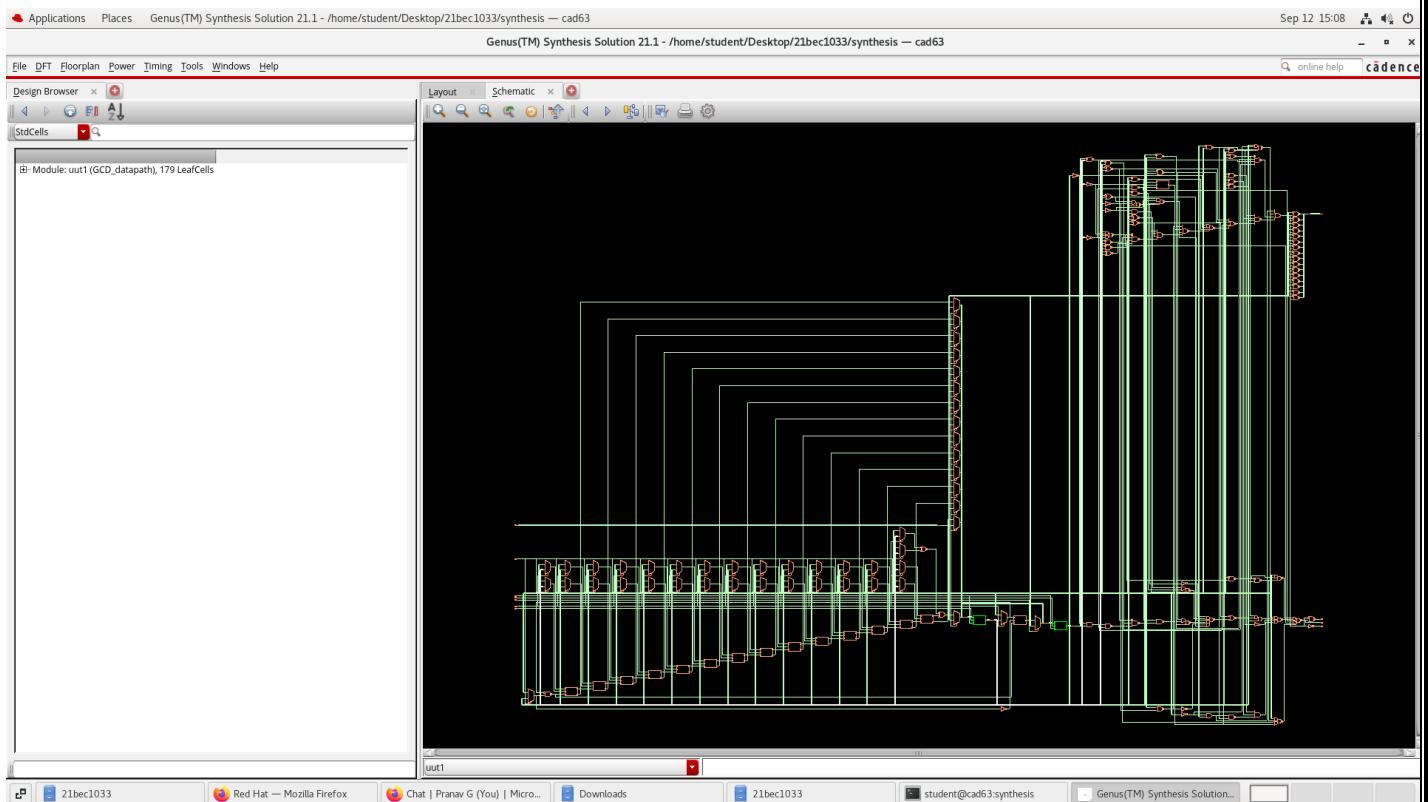
LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

Gate level Netlist:

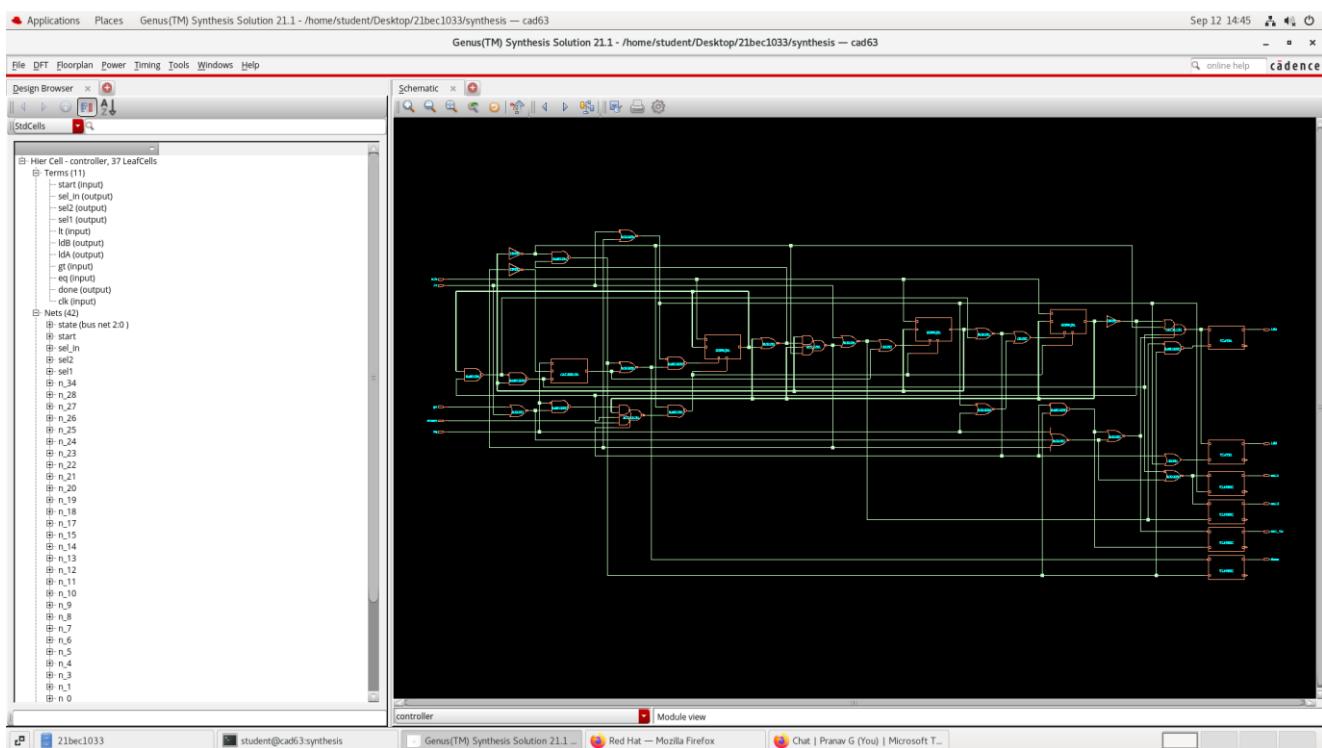
Synthesis effort: Medium



LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic



Synthesis effort : High



LAB4_2: Logic synthesis of a GCD computation using Datapath and Control path logic

Observations:

a) GCD computer_netlist_created file

```
// Generated by Cadence Genus(TM) Synthesis Solution 21.14-s882_1
// Generated on: Sep 12 2024 15:08:20 IST (Sep 12 2024 09:38:20 UTC)

// Verification Directory fv/GCD_mult

module PIP0_38(data_out, data_in, load, clk);
    input [15:0] data_in;
    input load, clk;
    output [15:0] data_out;
    wire [15:0] data_in;
    wire [15:0] data_out;
    wire [15:0] data_out;
    SDFFFX1(data_out, reg[9].(clk.(clk), .D(data_out[9]), .SI(data_in[9]), .SE(load), .Q(data_out[9])));
    SDFFFX1(data_out, reg[11].(clk.(clk), .D(data_out[11]), .SI(data_in[11]), .SE(load), .Q(data_out[11])));
    SDFFFX1(data_out, reg[7].(clk.(clk), .D(data_out[7]), .SI(data_in[7]), .SE(load), .Q(data_out[7])));
    SDFFFX1(data_out, reg[8].(clk.(clk), .D(data_out[8]), .SI(data_in[8]), .SE(load), .Q(data_out[8])));
    SDFFFX1(data_out, reg[10].(clk.(clk), .D(data_out[10]), .SI(data_in[10]), .SE(load), .Q(data_out[10])));
    SDFFFX1(data_out, reg[18].(clk.(clk), .D(data_out[18]), .SI(data_in[18]), .SE(load), .Q(data_out[18])));
    SDFFFX1(data_out, reg[2].(clk.(clk), .D(data_out[2]), .SI(data_in[2]), .SE(load), .Q(data_out[2])));
    SDFFFX1(data_out, reg[11].(clk.(clk), .D(data_out[11]), .SI(data_in[11]), .SE(load), .Q(data_out[11])));
    SDFFFX1(data_out, reg[4].(clk.(clk), .D(data_out[4]), .SI(data_in[4]), .SE(load), .Q(data_out[4])));
    SDFFFX1(data_out, reg[13].(clk.(clk), .D(data_out[13]), .SI(data_in[13]), .SE(load), .Q(data_out[13])));
    SDFFFX1(data_out, reg[14].(clk.(clk), .D(data_out[14]), .SI(data_in[14]), .SE(load), .Q(data_out[14])));
    SDFFFX1(data_out, reg[3].(clk.(clk), .D(data_out[3]), .SI(data_in[3]), .SE(load), .Q(data_out[3])));
    SDFFFX1(data_out, reg[12].(clk.(clk), .D(data_out[12]), .SI(data_in[12]), .SE(load), .Q(data_out[12])));
    SDFFFX1(data_out, reg[15].(clk.(clk), .D(data_out[15]), .SI(data_in[15]), .SE(load), .Q(data_out[15])));
    SDFFFX1(data_out, reg[5].(clk.(clk), .D(data_out[5]), .SI(data_in[5]), .SE(load), .Q(data_out[5])));
    SDFFFX1(data_out, reg[6].(clk.(clk), .D(data_out[6]), .SI(data_in[6]), .SE(load), .Q(data_out[6])));
endmodule

module PIP0_38(data_out, data_in, load, clk);
    input [15:0] data_in;
    input load, clk;
    output [15:0] data_out;
    wire [15:0] data_out;
    wire [15:0] data_out;
    SDFFFX1(data_out, reg[9].(clk.(clk), .D(data_out[9]), .SI(data_in[9]), .SE(load), .Q(data_out[9])));
    SDFFFX1(data_out, reg[11].(clk.(clk), .D(data_out[11]), .SI(data_in[11]), .SE(load), .Q(data_out[11])));
    SDFFFX1(data_out, reg[7].(clk.(clk), .D(data_out[7]), .SI(data_in[7]), .SE(load), .Q(data_out[7])));
    SDFFFX1(data_out, reg[8].(clk.(clk), .D(data_out[8]), .SI(data_in[8]), .SE(load), .Q(data_out[8])));
    SDFFFX1(data_out, reg[10].(clk.(clk), .D(data_out[10]), .SI(data_in[10]), .SE(load), .Q(data_out[10])));
    SDFFFX1(data_out, reg[12].(clk.(clk), .D(data_out[12]), .SI(data_in[12]), .SE(load), .Q(data_out[12])));
    SDFFFX1(data_out, reg[11].(clk.(clk), .D(data_out[11]), .SI(data_in[11]), .SE(load), .Q(data_out[11])));
    SDFFFX1(data_out, reg[4].(clk.(clk), .D(data_out[4]), .SI(data_in[4]), .SE(load), .Q(data_out[4])));
    SDFFFX1(data_out, reg[13].(clk.(clk), .D(data_out[13]), .SI(data_in[13]), .SE(load), .Q(data_out[13])));
    SDFFFX1(data_out, reg[14].(clk.(clk), .D(data_out[14]), .SI(data_in[14]), .SE(load), .Q(data_out[14])));
    SDFFFX1(data_out, reg[3].(clk.(clk), .D(data_out[3]), .SI(data_in[3]), .SE(load), .Q(data_out[3])));
    SDFFFX1(data_out, reg[12].(clk.(clk), .D(data_out[12]), .SI(data_in[12]), .SE(load), .Q(data_out[12])));
    SDFFFX1(data_out, reg[15].(clk.(clk), .D(data_out[15]), .SI(data_in[15]), .SE(load), .Q(data_out[15])));
    SDFFFX1(data_out, reg[5].(clk.(clk), .D(data_out[5]), .SI(data_in[5]), .SE(load), .Q(data_out[5])));
    SDFFFX1(data_out, reg[6].(clk.(clk), .D(data_out[6]), .SI(data_in[6]), .SE(load), .Q(data_out[6])));
endmodule

module GCD_datapath(data_out, gt, lt, eq, ldA, ldB, sell1, sel2, sel_in, data_in, clk);
    input ldA, ldB, sell1, sel2, sel_in, clk;
    input [15:0] data_in;
    output [15:0] data_out;
    output gt, lt, eq;
    wire ldA, ldB, sell1, sel2, sel_in, clk;
    wire [15:0] data_in;
    wire [15:0] data_out;
    wire gt, lt, eq;
    wire [15:0] Bus;
    wire [15:0] Bout;
    wire SB_sub_5_12_n_10, SB_sub_5_12_n_15, SB_sub_5_12_n_16,
    SB_sub_5_12_n_18, SB_sub_5_12_n_20, SB_sub_5_12_n_22,
    SB_sub_5_12_n_24, SB_sub_5_12_n_26;
    wire SB_sub_5_12_n_28, SB_sub_5_12_n_30, SB_sub_5_12_n_32,
    SB_sub_5_12_n_34, SB_sub_5_12_n_36, SB_sub_5_12_n_38,
    SB_sub_5_12_n_40, SB_sub_5_12_n_42;
    wire SB_sub_5_12_n_44, n_8, n_1, n_3, n_4, n_5, n_6,
    n_7, n_8, n_9, n_10, n_11, n_12, n_13, n_14;
    wire n_15, n_16, n_17, n_18, n_19, n_20, n_21, n_22,
    wire n_23, n_24, n_25, n_26, n_27, n_28, n_29, n_30;
    wire n_31, n_32, n_33, n_34, n_35, n_36, n_37, n_38;
    wire n_39, n_40, n_41, n_42, n_43, n_44, n_45, n_46;
    wire n_47, n_48, n_49, n_50, n_51, n_52, n_53, n_54;
    wire n_55, n_56, n_57, n_58, n_59, n_61, n_62, n_63;
    wire n_64, n_65, n_66, n_67, n_68, n_69, n_70, n_71;
    wire n_72, n_73, n_74, n_75, n_76, n_77, n_78, n_79;
    wire n_96, n_97, n_98, n_99, n_100, n_101;
    wire n_102, n_103, n_104, n_105, n_106, n_107, n_108, n_109;
    wire n_150, n_151, n_152, n_153, n_154, n_155, n_156, n_157;
    wire n_158, n_159, n_160, n_161, n_162, n_163;
    PIP0_38(A,load, Bus, ldA, clk);
    PIPE_38(B,out, Bus, ldB, clk);
    INVI14x56(A(.n_61), Y(.eq));
    MX2L2023(A(.A[12]), B(.Bout[12]), .S0(sell1), .Y(.n_196));
    MX2L2023(A(.A[12]), B(.Bout[12]), .S0(sell1), .Y(.n_103));
    MX2L2023(A(.A[12]), B(.Bout[12]), .S0(sell1), .Y(.n_90));
    MX2L2023(A(.A[12]), B(.Bout[12]), .S0(sell1), .Y(.n_96));
    MX2L2023(A(.A[12]), B(.Bout[12]), .S0(sell1), .Y(.n_79));
    MX2L2040_5526(A(.A[12]), B(.Bout[8]), .S0(sell1), .Y(.n_102));
    MX2L2041_6783(A(.A[11]), B(.Bout[1]), .S0(sell1), .Y(.n_95));
    MX2L2042_3688(A(.A[10]), B(.Bout[8]), .S0(sell1), .Y(.n_94));
    MX2L2043_1617(A(.A[11]), B(.Bout[11]), .S0(sell1), .Y(.n_105));
    MX2L2044_2802(A(.A[7]), B(.Bout[7]), .S0(sell1), .Y(.n_101));
    MX2L2045_1705(A(.A[15]), B(.Bout[15]), .S0(sel2), .Y(.n_93));

```

File Edit View

MX2XL_k4299_0480(.A_(H'000101), .B_(H'000101), .C_(H'000101), .D_(H'000101), .E_(H'000101))

MX2XL_g2050_1881(.A_(Out[15]), .B_(Bout[15]), .S0_(sel1), .Y_(n_109));

MX2XL_g2052_7482(.A_(Out[13]), .B_(Bout[13]), .S0_(sel1), .Y_(n_107));

MX2XL_g2055_9315(.A_(Out[10]), .B_(Bout[10]), .S0_(sel1), .Y_(n_104));

MX2XL_g2056_9945(.A_(Out[5]), .B_(Bout[5]), .S0_(sel1), .Y_(n_99));

MX2XL_g2060_7410(.A_(Out[4]), .B_(Bout[4]), .S0_(sel1), .Y_(n_98));

MX2XL_g2064_5107(.A_(Out[3]), .B_(Bout[3]), .S0_(sel1), .Y_(n_97));

MX2XL_g2066_4319(.A_(Out[14]), .B_(Bout[14]), .S0_(sel1), .Y_(n_108));

NOR2XL_g2480_8428(.A_(n_61), .B_(gt), .Y_(1t));

ORAX1_g2477_5526(.A_(n_19), .B_(n_39), .C_(n_41), .D_(n_59), .Y_(n_61));

NAND4XL_g2478_6783(.A_(n_22), .B_(n_27), .C_(n_13), .D_(n_58), .Y_(n_59));

A0T21XL_g2479_3680(.A0_(n_47), .A1_(n_57), .B0_(n_29), .B1_(n_34), .C_(n_49), .Y_(gt));

NORAX1_g2480_1617(.A_(n_14), .B_(n_30), .C_(n_12), .D_(n_55), .Y_(n_58));

OA122XL_g2481_2802(.A0_(n_39), .A1_(n_56), .B0_(n_18), .B1_(n_44), .Y_(n_57));

OA21XL_g2482_1705(.A0_(n_10), .A1_(n_54), .B0_(n_13), .Y_(n_56));

OA121XL_g2483_5122(.A0_(Out[0]), .A1_(n_3), .B0_(n_16), .C0_(n_53), .Y_(n_55));

NOR2XL_g2484_8246(.A_(n_7), .B_(n_52), .Y_(n_54));

NORAX4_g2485_7089(.A_(n_23), .B_(n_37), .C_(n_43), .D_(n_51), .Y_(n_53));

OA122XL_g2486_6131(.A0_(n_33), .A1_(n_38), .B0_(n_43), .B1_(n_50), .C0_(n_24), .CI_(n_9), .Y_(n_52));

ORAX1_g2487_1881(.A_(n_11), .B_(n_28), .C_(n_7), .D_(n_46), .Y_(n_51));

A0T21XL_g2488_5115(.A0_(n_11), .A1_(n_48), .B0_(n_30), .Y_(n_50));

OA122XL_g2489_7482(.A0_(Out[12]), .A1_(n_45), .B0_(n_25), .B1_(n_37), .Y_(n_49));

NAND2XL_g2490_4733(.A_(n_16), .B_(n_42), .Y_(n_48));

OA121XL_g2491_6161(.A0_(Out[12]), .A1_(n_40), .B0_(n_45), .Y_(n_47));

NAND4XL_g2492_9315(.A_(n_34), .B_(n_25), .C_(n_15), .D_(n_31), .Y_(n_46));

AND3XL_g2493_9945(.A_(n_22), .B_(n_27), .C_(n_36), .Y_(n_44));

NAND2XL_g2494_2883(.A0_(n_40), .B_(Bout[12]), .Y_(n_45));

NAND2XL_g2495_2346(.A0_(n_14), .B_(n_8), .C_(n_35), .Y_(n_42));

NAND4XL_g2496_1666(.A_(n_10), .B_(n_11), .C_(n_6), .D_(n_8), .Y_(n_41));

OA121XL_g2497_7410(.A0_(Bout[4]), .A1_(n_1), .B0_(n_26), .C0_(n_32), .Y_(n_43));

NAND2XL_g2498_6417(.A0_(n_37), .B_(n_15), .Y_(n_40));

OA121XL_g2499_5477(.A0_(n_21), .A1_(n_26), .B0_(n_28), .Y_(n_38));

File Edit View

A0T12XL_g2499_5477(.A0_(n_21), .A1_(n_26), .B0_(n_28), .Y_(n_38));

OA121XL_g2500_2398(.A0_(Bout[3]), .A1_(n_8), .B0_(n_20), .C0_(n_17), .Y_(n_19));

NAND2XL_g2501_5107(.A_(n_19), .B_(n_20), .Y_(n_36));

NAND2XL_g2502_6260(.A_(n_12), .B_(n_6), .Y_(n_35));

OA12BBL1XL_g2506_4319(.A0N_(Out[14]), .A1N_(n_4), .B0_(n_29), .Y_(n_37));

INVA1XL_g2507_6783(.A0_(Out[6]), .A1_(n_5), .B0_(n_9), .Y_(n_32));

INVA1XL_g2508_1193(.A_(n_23), .Y_(n_24));

NOR2XL_g2509_3680(.A0_(Bout[3]), .B_(Out[3]), .Y_(n_30));

NAND2XL_g2510_1617(.A_(n_15), .B_(n_2), .Y_(n_29));

NOR2XL_g2511_2802(.A0_(Bout[5]), .B_(Out[5]), .Y_(n_28));

NAND2XL_g2512_1785(.A0_(Out[18]), .B_(Bout[18]), .Y_(n_27));

NAND2XL_g2513_5122(.A0_(Bout[5]), .B_(Out[5]), .Y_(n_26));

NAND2XL_g2514_8246(.A0_(Out[13]), .B_(Bout[13]), .Y_(n_25));

NOR2XL_g2515_7098(.A_(Out[5]), .B_(n_5), .Y_(n_23));

NAND2XL_g2516_6131(.A0_(Out[11]), .B_(Bout[11]), .Y_(n_22));

AND2XL_g2517_1881(.A_(n_1), .B_(Bout[4]), .Y_(n_21));

NAND2XL_g2518_5115(.A0_(Bout[18]), .B_(Out[18]), .Y_(n_20));

AND2XL_g2519_7482(.A_(n_8), .B_(Bout[9]), .Y_(n_19));

INVA1XL_g2520_1173(.A_(n_17), .Y_(n_18));

NAND2XL_g2521_4733(.A0_(Bout[11]), .B_(Out[11]), .Y_(n_17));

NAND2XL_g2522_6161(.A0_(Out[2]), .B_(Bout[2]), .Y_(n_16));

NAND2XL_g2523_9315(.A0_(Bout[13]), .B_(Out[13]), .Y_(n_15));

NOR2XL_g2524_9945(.A0_(Out[2]), .B_(Bout[2]), .Y_(n_14));

NAND2XL_g2525_2883(.A0_(Out[8]), .B_(Bout[8]), .Y_(n_13));

AND2XL_g2526_2346(.A_(n_3), .B_(Out[9]), .Y_(n_12));

NAND2XL_g2527_1666(.A0_(Bout[3]), .B_(Out[3]), .Y_(n_11));

NOR2XL_g2528_7418(.A0_(Out[8]), .B_(Bout[8]), .Y_(n_10));

NOR2XL_g2529_6417(.A0_(Out[7]), .B_(Bout[7]), .Y_(n_9));

NAND2XL_g2530_5477(.A0_(Bout[1]), .B_(Out[1]), .Y_(n_8));

NOR2XL_g2531_2398(.A0_(Bout[7]), .B_(Out[7]), .Y_(n_7));

NAND2XL_g2532_5187(.A0_(Out[1]), .B_(Bout[1]), .Y_(n_6));

INVA1XL_g2533(.A_(Bout[6]), .Y_(n_5));

INVA1XL_g2534(.A_(Bout[4]), .Y_(n_4));

INVA1XL_g2535(.A_(Bout[8]), .Y_(n_3));

INVA1XL_g2536(.A_(Bout[15]), .Y_(n_2));

INVA1XL_g2537(.A_(Out[4]), .Y_(n_1));

INVA1XL_g2538(.A_(Out[9]), .Y_(n_0));

MX2XL_g2601_6260(.A_(n_67), .B_(data_in[5]), .S0_(sel_in), .Y_(Bout[5]));

MX2XL_g2602_4319(.A_(n_76), .B_(data_in[4]), .S0_(sel_in), .Y_(Bout[4]));

MX2XL_g2603_8428(.A_(n_69), .B_(data_in[7]), .S0_(sel_in), .Y_(Bout[7]));

File Edit View

MX2XL_g1863_8428(.A_(n_69), .B_(data_in[7]), .S0_(sel_in), .Y_(Bout[7]));

MX2XL_g1864_5526(.A_(n_68), .B_(data_in[6]), .S0_(sel_in), .Y_(Bout[6]));

MX2XL_g1865_6783(.A_(n_77), .B_(data_in[5]), .S0_(sel_in), .Y_(Bout[5]));

MX2XL_g1866_3680(.A_(n_66), .B_(data_in[4]), .S0_(sel_in), .Y_(Bout[4]));

MX2XL_g1867_1617(.A_(n_75), .B_(data_in[3]), .S0_(sel_in), .Y_(Bout[3]));

MX2XL_g1868_2802(.A_(n_65), .B_(data_in[3]), .S0_(sel_in), .Y_(Bout[3]));

MX2XL_g1869_1705(.A_(n_73), .B_(data_in[11]), .S0_(sel_in), .Y_(Bout[11]));

MX2XL_g1870_5122(.A_(n_74), .B_(data_in[12]), .S0_(sel_in), .Y_(Bout[12]));

MX2XL_g1871_8246(.A_(n_63), .B_(data_in[1]), .S0_(sel_in), .Y_(Bout[1]));

MX2XL_g1872_7098(.A_(n_62), .B_(data_in[8]), .S0_(sel_in), .Y_(Bout[8]));

MX2XL_g1873_6131(.A_(n_64), .B_(data_in[2]), .S0_(sel_in), .Y_(Bout[2]));

MX2XL_g1874_1881(.A_(n_72), .B_(data_in[10]), .S0_(sel_in), .Y_(Bout[10]));

MX2XL_g1875_5115(.A_(n_71), .B_(data_in[9]), .S0_(sel_in), .Y_(Bout[9]));

MX2XL_g1876_7482(.A_(n_70), .B_(data_in[8]), .S0_(sel_in), .Y_(Bout[8]));

NOR2XL_g1877_4733(.A_(data_in[14]), .B_(n_61), .Y_(data_out[14]));

NOR2XL_g1878_6161(.A_(data_in[8]), .B_(n_61), .Y_(data_out[8]));

NOR2XL_g1879_9315(.A_(data_in[15]), .B_(n_61), .Y_(data_out[15]));

NOR2XL_g1880_9945(.A_(data_in[8]), .B_(n_61), .Y_(data_out[8]));

NOR2XL_g1881_2883(.A_(data_in[12]), .B_(n_61), .Y_(data_out[12]));

NOR2XL_g1882_2346(.A_(data_in[7]), .B_(n_61), .Y_(data_out[7]));

NOR2XL_g1883_1666(.A_(data_in[11]), .B_(n_61), .Y_(data_out[11]));

NOR2XL_g1884_7418(.A_(data_in[6]), .B_(n_61), .Y_(data_out[6]));

NOR2XL_g1885_6417(.A_(data_in[5]), .B_(n_61), .Y_(data_out[5]));

NOR2XL_g1886_5477(.A_(data_in[10]), .B_(n_61), .Y_(data_out[10]));

NOR2XL_g1887_2398(.A_(data_in[4]), .B_(n_61), .Y_(data_out[4]));

NOR2XL_g1888_5107(.A_(data_in[13]), .B_(n_61), .Y_(data_out[13]));

NOR2XL_g1889_2680(.A_(data_in[5]), .B_(n_61), .Y_(data_out[5]));

NOR2XL_g1890_4319(.A_(data_in[9]), .B_(n_61), .Y_(data_out[9]));

NOR2XL_g1891_8428(.A_(data_in[2]), .B_(n_61), .Y_(data_out[2]));

NOR2XL_g1892_5526(.A_(data_in[1]), .B_(n_61), .Y_(data_out[1]));

XOR2XL_g5b_512_1323_6783(.A_(SB_sub_5_12_n_16), .B_(SB_sub_5_12_n_17), .Y_(SB_sub_5_12_n_4), .Y_(n_77));

ADD1F1XL_g5b_512_1324_3680(.A_(n_108), .B_(n_161), .CI_(n_160));

ADD1F1XL_g5b_512_1325_1617(.A_(n_107), .B_(n_156), .CI_(n_156));

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

File Edit View

```

0AI211X1 g2483_512(.A0(Aout[8]), .A1(n_3), .B0(n_16), .C0
(n_53); NOR2X1 g2484_8246(.A(n_7), .B(n_52), .Y(n_54));
NOR4X1 g2485_7098(.A(n_23), .B(n_37), .C(n_43), .D(n_51), .Y
(n_53)); OR1222X1 g2486_6131(.A0(n_33), .A1(n_38), .B0(n_43), .B1(n_50),
.C0(n_24), .C1(n_9), .Y(n_52)); OR4X1 g2487_1881(.A(n_21), .B(n_28), .C(n_7), .D(n_46), .Y
(n_51)); A0I211X1 g2488_5115(.A0(n_11), .A1(n_48), .B0(n_38), .Y(n_58));
A0I222X1 g2489_7403(.A0(Aout[7]), .A1(n_45), .B0(n_25), .B1
(n_37), .Y(n_49)); NAND2X1 g2490_4733(.A(n_16), .B(n_42), .Y(n_48));
OAI211X1 g2491_6161(.A0(Aout[12]), .A1(n_40), .B0(n_45), .Y
(n_47)); NAND4BX1 g2492_9315(.AN(n_34), .B(n_25), .C(n_15), .D(n_31),
.Y(n_46)); AND3X2 g2493_9945(.A(n_22), .B(n_27), .C(n_44), .D(n_36));
NAND2BX1 g2494_2883(.AN(n_40), .B(Bout[12]), .Y(n_45)); NAND3BX1
g2495_2346(.AN(n_42), .B(Bout[11]), .C(n_6), .D(n_8), .Y
(n_41)); NAND4BX1 g2496_1666(.AN(n_10), .B(n_11), .C(n_6), .D(n_8), .Y
(n_41)); A0I211X1 g2497_7140(.A0(Bout[4]), .A1(n_1), .B0(n_26), .C0
(n_32), .Y(n_43)); NAND2X1 g2498_6417(.A(n_15), .B(n_36), .Y(n_40));
A0I211X1 g2499_5477(.A0(n_21), .A1(n_26), .B0(n_28), .Y(n_38));
OAI211X1 g2500_2398(.A0(Bout[9]), .A1(n_0), .B0(n_20), .C0
(n_17), .Y(n_39)); NAND2X1 g2501_5107(.A(n_19), .B(n_20), .Y(n_36));
NAND2X1 g2502_6260(.A(n_12), .B(n_6), .Y(n_35)); OAI2B1X1
g2503_4319(.A0(Aout[14]), .A1(n_4), .B0(n_29), .Y
(n_34)); INVX1 g2504_1_A(n_32), .Y(n_33)); XMR2X1 g2505_8428(.Aout[12]), .B(Bout[12]), .Y(n_31));
OAI222X1 g2506_5526(.A0(Aout[14]), .A1(n_4), .B0(Aout[15]), .B1
(n_2), .Y(n_34)); A0I211X1 g2507_6783(.A0(Aout[6]), .A1(n_5), .B0(n_9), .Y
(n_32)); INVX1 g2508_1_A(n_23), .Y(n_24)); NOR2X1 g2509_3680(.Aout[3]), .B(Aout[3]), .Y(n_30));
NAND2X1 g2510_1617(.A0(Aout[15]), .B(n_2), .Y(n_29)); NOR2X1
g2511_2892(.A0(Bout[5]), .B(Aout[5]), .Y(n_28)); NOR2X1
g2512_1921(.A0(Bout[9]), .B(Bout[10]), .Y(n_27)); NOR2X1
g2513_2532(.A0(Bout[11]), .B(Aout[11]), .Y(n_26)); NOR2X1
g2514_2466(.A0(Aout[10]), .B(Bout[10]), .Y(n_25)); NOR2X1
g2515_7098(.A0(Aout[6]), .B(n_8), .Y(n_23)); NAND2BX1
g2516_6131(.AN(n_11), .B(Bout[11]), .Y(n_22)); AND2X1
g2517_1881(.A(n_1), .B(Bout[4]), .Y(n_21)); NAND2BX1
g2518_5115(.AN(Bout[18]), .B(Aout[18]), .Y(n_20)); AND2X1
g2519_7482(.A(n_0), .B(Bout[9]), .Y(n_19)); TMUX1
g2520_1_A(n_17), .V(n_18));

```

Ln 210, Col 39 20,377 characters

File Edit View

```

AND2X1 g2519_7482(.A(n_0), .B(Bout[9]), .Y(n_19));
NAND2BX1 g2520_6131(.AN(Bout[11]), .B(Aout[11]), .Y(n_17));
NAND2BX1 g2522_6161(.AN(Bout[2]), .B(Bout[2]), .Y(n_26));
NOR2BX1 g2523_9315(.AN(Bout[13]), .B(Aout[13]), .Y(n_15));
NOR2BX1 g2524_9945(.AN(Aout[2]), .B(Bout[2]), .Y(n_14));
NAND2BX1 g2525_2883(.AN(Aout[8]), .B(Bout[8]), .Y(n_13));
AND2X1 g2526_2346(.A(n_3), .B(Aout[8]), .Y(n_12));
NAND2BX1 g2527_1666(.AN(Bout[3]), .B(Aout[3]), .Y(n_11));
NOR2BX1 g2528_7410(.AN(Aout[8]), .B(Bout[8]), .Y(n_10));
NAND2BX1 g2529_6417(.AN(Aout[7]), .B(Bout[7]), .Y(n_9));
NAND2BX1 g2530_5477(.AN(Bout[1]), .B(Aout[1]), .Y(n_8));
NOR2BX1 g2531_2398(.AN(Bout[7]), .B(Bout[7]), .Y(n_7));
NAND2BX1 g2532_5526(.AN(Bout[6]), .B(Bout[6]), .Y(n_6));
INVX1 g2533(.A(Bout[6]), .Y(n_5));
INVX1 g2534(.A(Bout[14]), .Y(n_4));
INVX1 g2535(.A(Bout[0]), .Y(n_3));
INVX1 g2536(.A(Bout[15]), .Y(n_2));
INVX1 g2537(.A(Bout[4]), .Y(n_1));
INVX1 g2538(.A(Bout[9]), .Y(n_0));
INVX1 g1861_6260(.A(n_67), .B(data_in[5]), .S0(sel_in), .Y
(SB_sub_5_12_n_44));
INVX1 g1862_4319(.A(n_76), .B(data_in[14]), .S0(sel_in), .Y
(SB_sub_5_12_n_45));
INVX1 g1863_3680(.A(n_66), .B(data_in[4]), .S0(sel_in), .Y
(SB_sub_5_12_n_46));
INVX1 g1864_5477(.A(n_68), .B(data_in[6]), .S0(sel_in), .Y
(SB_sub_5_12_n_47));
INVX1 g1865_8246(.A(n_77), .B(data_in[15]), .S0(sel_in), .Y
(SB_sub_5_12_n_48));
INVX1 g1866_3808(.A(n_66), .B(data_in[4]), .S0(sel_in), .Y
(SB_sub_5_12_n_49));
INVX1 g1867_1617(.A(n_75), .B(data_in[13]), .S0(sel_in), .Y
(SB_sub_5_12_n_50));
INVX1 g1868_2892(.A(n_65), .B(data_in[3]), .S0(sel_in), .Y
(SB_sub_5_12_n_51));
INVX1 g1869_6131(.A(n_64), .B(data_in[2]), .S0(sel_in), .Y
(SB_sub_5_12_n_52));
INVX1 g1870_1705(.A(n_73), .B(data_in[11]), .S0(sel_in), .Y
(SB_sub_5_12_n_53));
INVX1 g1871_5122(.A(n_74), .B(data_in[12]), .S0(sel_in), .Y
(SB_sub_5_12_n_54));
INVX1 g1872_8246(.A(n_63), .B(data_in[1]), .S0(sel_in), .Y
(SB_sub_5_12_n_55));
INVX1 g1873_7098(.A(n_62), .B(data_in[0]), .S0(sel_in), .Y
(SB_sub_5_12_n_56));
INVX1 g1874_6131(.A(n_64), .B(data_in[2]), .S0(sel_in), .Y
(SB_sub_5_12_n_57));
INVX1 g1875_5115(.A(n_71), .B(data_in[9]), .S0(sel_in), .Y
(SB_sub_5_12_n_58));

```

Ln 258, Col 65 20,377 characters

File Edit View

```

(Bus[9]);
M2K2X1 g1876_7482(.A(n_78), .B(data_in[8]), .S0(sel_in), .Y
(SB_sub_5_11_n_11));
NOR2BX1 g1877_4733(.AN(data_in[14]), .B(n_61), .Y(data_out[14]));
NOR2BX1 g1878_6161(.AN(data_in[8]), .B(n_61), .Y(data_out[8]));
NOR2BX1 g1879_9315(.AN(data_in[15]), .B(n_61), .Y(data_out[15]));
NOR2BX1 g1880_9945(.AN(data_in[8]), .B(n_61), .Y(data_out[8]));
NOR2BX1 g1881_2883(.AN(data_in[12]), .B(n_61), .Y(data_out[12]));
NOR2BX1 g1882_2346(.AN(data_in[7]), .B(n_61), .Y(data_out[7]));
NOR2BX1 g1883_1666(.AN(data_in[11]), .B(n_61), .Y(data_out[11]));
NOR2BX1 g1884_7410(.AN(data_in[6]), .B(n_61), .Y(data_out[6]));
NOR2BX1 g1885_6417(.AN(Aout[7]), .B(Bout[7]), .Y(n_9));
NAND2BX1 g2530_5477(.AN(Bout[1]), .B(Aout[1]), .Y(n_8));
NOR2BX1 g2531_2398(.AN(Bout[7]), .B(Bout[7]), .Y(n_7));
NAND2BX1 g2532_5526(.AN(Bout[6]), .B(Bout[6]), .Y(n_6));
INVX1 g2533(.A(Bout[6]), .Y(n_5));
INVX1 g2534(.A(Bout[14]), .Y(n_4));
INVX1 g2535(.A(Bout[0]), .Y(n_3));
INVX1 g2536(.A(Bout[15]), .Y(n_2));
INVX1 g2537(.A(Bout[4]), .Y(n_1));
INVX1 g2538(.A(Bout[9]), .Y(n_0));
INVX1 g2539(.A(Bout[14]), .Y(n_13));
INVX1 g2540(.A(Bout[13]), .Y(n_12));
INVX1 g2541(.A(Bout[14]), .Y(n_11));
INVX1 g2542(.A(Bout[16]), .Y(n_10));
INVX1 g2543(.A(Bout[10]), .Y(n_15));
INVX1 g2544(.A(Bout[13]), .Y(n_16));
INVX1 g2545(.A(Bout[11]), .Y(n_17));
INVX1 g2546(.A(Bout[12]), .Y(n_18));
INVX1 g2547(.A(Bout[3]), .Y(n_19));
INVX1 g2548(.A(Bout[5]), .Y(n_20));
INVX1 g2549(.A(Bout[14]), .Y(n_21));
INVX1 g2550(.A(Bout[2]), .Y(n_22));
INVX1 g2551(.A(Bout[11]), .Y(n_23));
endmodule

```

Ln 306, Col 60 20,377 characters

File Edit View

```

(SB_sub_5_12_n_16);
NAND2BX1 SB_sub_5_12_n_16_g2346(.AN(n_94), .B(n_78), .Y
(SB_sub_5_12_n_15));
INVX1 SB_sub_5_12_n_16_g2345(.A(n_78), .Y(SB_sub_5_12_n_10));
INVX1 g2345(.A(Bout[17]), .B(Bout[17]), .S0(sel12), .Y(n_150));
INVX1 g2346(.A(Bout[18]), .B(Bout[18]), .S0(sel12), .Y(n_151));
INVX1 g2347(.A(Bout[19]), .B(Bout[19]), .S0(sel12), .Y(n_152));
INVX1 g2348(.A(Bout[20]), .B(Bout[20]), .S0(sel12), .Y(n_153));
INVX1 g2349(.A(Bout[21]), .B(Bout[21]), .S0(sel12), .Y(n_154));
INVX1 g2350(.A(Bout[22]), .B(Bout[22]), .S0(sel12), .Y(n_155));
INVX1 g2351(.A(Bout[23]), .B(Bout[23]), .S0(sel12), .Y(n_156));
INVX1 g2352(.A(Bout[24]), .B(Bout[24]), .S0(sel12), .Y(n_157));
INVX1 g2353(.A(Bout[1]), .B(Bout[1]), .S0(sel12), .Y(n_158));
INVX1 g2354(.A(Bout[2]), .B(Bout[2]), .S0(sel12), .Y(n_159));
INVX1 g2355(.A(Bout[3]), .B(Bout[3]), .S0(sel12), .Y(n_160));
INVX1 g2356(.A(Bout[4]), .B(Bout[4]), .S0(sel12), .Y(n_161));
INVX1 g2357(.A(Bout[5]), .B(Bout[5]), .S0(sel12), .Y(n_162));
INVX1 g2358(.A(Bout[6]), .B(Bout[6]), .S0(sel12), .Y(n_163));
endmodule

```

File Edit View

```

module controller(l1d, l1b, sel1, sel2, sel_in, done, clk, lt, gt, eq,
start;
input clk, lt, gt, eq, start;
output l1d, l1b, sel1, sel2, sel_in, done;
wire clk, lt, gt, eq, start;
wire l1d, l1b, sel1, sel2, sel_in, done;
wire [2:0] state;
wire UNCONNECTED, UNCONNECTED0, UNCONNECTED1, UNCONNECTED2,
UNCONNECTED3, UNCONNECTED4, n_0, n_2;
wire n_3, n_4, n_6, n_8, n_9, n_10, n_11;
wire n_12, n_13, n_14, n_15, n_16, n_17, n_18, n_19;
wire n_20, n_21, n_22, n_23, n_24, n_25, n_26, n_27;
TLATX1 l1d_reg(.GN(n_21), .D(n_17), .Q(l1d), .QN(UNCONNECTED0));
TLATX1 l1b_reg(.GN(n_21), .D(n_19), .Q(l1b), .QN(UNCONNECTED0));
TLATX1 done_reg(.G(n_27), .D(n_11), .Q(done), .QN(UNCONNECTED1));
TLATX1 sel1_reg(.G(n_20), .D(n_16), .Q(sel1), .QN(UNCONNECTED2));
TLATX1 sel2_reg(.G(n_19), .D(n_12), .Q(sel2), .QN(UNCONNECTED3));
TLATX1 sel2z_reg(.G(n_20), .D(n_14), .Q(sel2z), .QN(UNCONNECTED4));
NAND2BX1 g21_1666(.AN(n_24), .B(n_11), .Y(n_27));
NAND2BX1 g21_6417(.A(n_29), .B(n_19), .Y(n_21));
NAND2BX1 g21_5477(.A(n_10), .B(n_15), .Y(n_20));
NAND2BX1 g21_2398(.A(n_11), .B(n_26), .Y(n_17));
NAND2BX1 g22_5107(.AN(n_12), .B(n_15), .Y(n_19));
NAND2BX1 g22_6260(.AN(n_13), .A1(eq), .B0(n_10), .Y(n_24));
INVX1 g2251(.A(n_25), .Y(n_16));
NAND2X1 g23_419(.A(n_13), .B(lt), .Y(n_25));
NAND2X1 g277(.A(n_14), .Y(n_14));

```

File Edit View

```

INVX1 g278_7410(.A(n_29), .B(n_19), .Y(n_21));
NAND2BX1 g279_5526(.AN(lt), .B(n_13), .Y(n_26));
NAND2X1 g280_6783(.A(n_23), .B(n_22), .Y(n_13));
NAND2BX1 g281_3680(.A(n_11), .B(n_28), .Y(n_12));
OAI2B1X1 g282_8428(.A1(lt), .B0(n_13), .C0(n_7), .Y
(n_15));
NAND2BX1 g283_7479(.AN(lt), .B(n_13), .Y(n_26));
NAND2X1 g284_5107(.AN(lt), .B(n_13), .Y(n_25));
NAND2BX1 g285_6131(.A(state[1]), .B(n_5), .C(n_9), .Y
(n_23));
NOR2BX1 g286_1781(.A(state[0]), .B(state[1]), .Y(n_8));
INVX1 g287_1_A(eq), .Y(n_7));
INVX1 g288_2882(.A(state[2]), .B(n_8), .Y(n_22));
NAND2X1 g289_7475(.A(n_25), .B(n_13), .Y(n_26));
NAND2BX1 g290_6245(.A(state[2]), .B(n_9), .C(n_8), .Y
(n_28));
NAND2BX1 g291_4733(.A(state[2]), .B(n_9), .C(n_10));
NOR2BX1 g292_7098(.AN(state[0]), .B(state[1]), .Y(n_9));
NAND2X1 g293_6131(.A(state[1]), .B(n_5), .C(n_9), .Y
(n_23));
NOR2BX1 g294_1781(.A(state[0]), .B(state[1]), .Y(n_8));
INVX1 g295_7408(.A(eq), .Y(n_7));
SDFFFX1 state_res[0].(clk, lck, .D(state[0]), .SI(n_0), .SE
(n_5));
SDFFFX1 state_res[1].(clk, lck, .D(state[2]), .SI(n_3), .SE
(n_6));
SDFFFX1 state_res[2].(clk, lck, .D(state[1]), .SI(n_4), .SE
(n_5));
SDFFFX1 state_res[3].(clk, lck, .D(state[2]), .SI(n_1), .SE
(n_4));
OAI211X1 g296_7473(.A(n_24), .B(n_24), .Y(n_26));
NAND2BX1 g297_6556(.AN(lt), .B(n_24), .Y(n_26));
NOR2BX1 g298_6161(.AN(n_22), .B(n_29), .Y(n_2));
NAND2BX1 g299_9315(.AN(n_27), .B(n_25), .Y(n_9));
NAND4BX1 g300_2(.AN(start), .B(n_23), .C(n_28), .D(n_2), .Y
(n_35));
endmodule

```

File Edit View

```

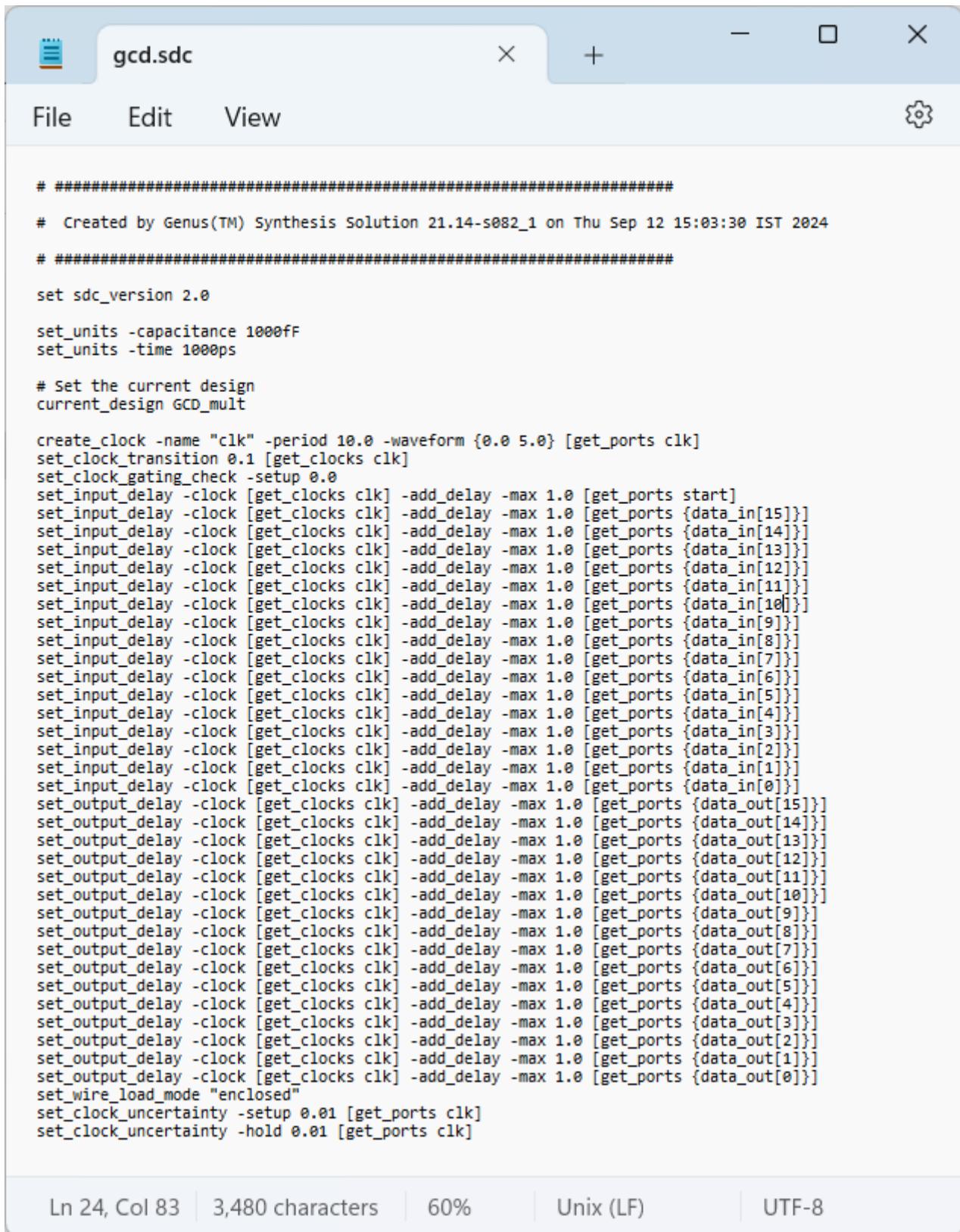
input GCD0(data_out, done, data_in, clk, start);
input [15:0] data_in;
input clk, start;
output [15:0] data_out;
wire [15:0] data_in;
wire clk, start;
wire [15:0] data_out;
wire done;
wire eq, st, l1d, l1b, lt, sel1, sel2, sel_in;
GCD0 module uut(data_out, done, data_in, clk, start);
controller uut(l1d, l1b, sel1, sel2, sel_in, done, clk, lt, gt, eq,
start);
endmodule

```

Ln 357, Col 71 20,377 characters

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

b) GCD computer_constraint_created file



The screenshot shows a text editor window titled "gcd.sdc". The menu bar includes "File", "Edit", "View", and a gear icon. The code in the editor is a System Design Constraints (SDC) file. It starts with a header indicating it was created by Genus(TM) Synthesis Solution 21.14-s082_1 on Thu Sep 12 15:03:30 IST 2024. It sets the SDC version to 2.0 and specifies units for capacitance (1000fF) and time (1000ps). It identifies the current design as "GCD_mult". The file contains numerous "set_input_delay" and "set_output_delay" commands for various ports, all configured to have a maximum delay of 1.0. These commands are grouped by clock edge (negedge or posedge) and target specific data input and output ports. The file concludes with setting the wire load mode to "enclosed" and specifying clock uncertainty for setup and hold times.

```
# #####
# Created by Genus(TM) Synthesis Solution 21.14-s082_1 on Thu Sep 12 15:03:30 IST 2024
# #####
set sdc_version 2.0
set_units -capacitance 1000fF
set_units -time 1000ps
# Set the current design
current_design GCD_mult
create_clock -name "clk" -period 10.0 -waveform {0.0 5.0} [get_ports clk]
set_clock_transition 0.1 [get_clocks clk]
set_clock_gating_check -setup 0.0
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports start]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[15]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[14]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[13]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[12]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[11]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[10]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[9]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[8]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[7]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[6]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[5]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[4]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[3]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[2]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[1]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[0]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[15]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[14]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[13]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[12]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[11]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[10]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[9]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[8]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[7]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[6]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[5]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[4]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[3]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[2]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[1]}]
set_output_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_out[0]}]
set_wire_load_mode "enclosed"
set_clock_uncertainty -setup 0.01 [get_ports clk]
set_clock_uncertainty -hold 0.01 [get_ports clk]
```

Ln 24, Col 83 | 3,480 characters | 60% | Unix (LF) | UTF-8

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

c) GCD computer_timing_created file

Synthesis effort: Medium

```
gcd_netlist.v
```

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(fF)	(ps)	(ps)	(ps)	
(clock clk)	launch					0 R
uut1						
A						
data_out_reg[0]/CK						
data_out_reg[0]/Q	SDFFQX1	5	1.2	37	+232	232 R
A/data_out[0]						
g2042_3680/A						+0 232
g2042_3680/Y	MX2XL	2	0.4	29	+152	384 R
SB_sub_5_12_g1340_2346/AN	NAND2BX1	2	1.0	40	+78	461 R
SB_sub_5_12_g1340_2346/Y						+0 461
SB_sub_5_12_g1337_9315/CI						
SB_sub_5_12_g1337_9315/CO	ADDFX1	1	0.6	39	+186	648 R
SB_sub_5_12_g1336_6161/CI						+0 648
SB_sub_5_12_g1336_6161/CO	ADDFX1	1	0.6	39	+186	834 R
SB_sub_5_12_g1335_4733/CI						+0 834
SB_sub_5_12_g1335_4733/CO	ADDFX1	1	0.6	39	+186	1020 R
SB_sub_5_12_g1334_7482/CI						+0 1020
SB_sub_5_12_g1334_7482/CO	ADDFX1	1	0.6	39	+186	1206 R
SB_sub_5_12_g1333_5115/CI						+0 1206
SB_sub_5_12_g1333_5115/CO	ADDFX1	1	0.6	39	+186	1392 R
SB_sub_5_12_g1332_1881/CI						+0 1392
SB_sub_5_12_g1332_1881/CO	ADDFX1	1	0.6	39	+186	1578 R
SB_sub_5_12_g1331_6131/CI						+0 1578
SB_sub_5_12_g1331_6131/CO	ADDFX1	1	0.6	39	+186	1764 R
SB_sub_5_12_g1330_7098/CI						+0 1764
SB_sub_5_12_g1330_7098/CO	ADDFX1	1	0.6	39	+186	1950 R
SB_sub_5_12_g1329_8246/CI						+0 1950
SB_sub_5_12_g1329_8246/CO	ADDFX1	1	0.6	39	+186	2136 R
SB_sub_5_12_g1328_5122/CI						+0 2136
SB_sub_5_12_g1328_5122/CO	ADDFX1	1	0.6	39	+186	2322 R
SB_sub_5_12_g1327_1705/CI						+0 2322
SB_sub_5_12_g1327_1705/CO	ADDFX1	1	0.6	39	+186	2508 R
SB_sub_5_12_g1326_2802/CI						+0 2508
SB_sub_5_12_g1326_2802/CO	ADDFX1	1	0.6	39	+186	2694 R
SB_sub_5_12_g1325_1617/CI						+0 2694
SB_sub_5_12_g1325_1617/CO	ADDFX1	1	0.6	39	+186	2880 R
SB_sub_5_12_g1324_3680/CI						+0 2880
SB_sub_5_12_g1324_3680/CO	ADDFX1	1	0.4	35	+184	3063 R
SB_sub_5_12_g1323_6783/B						+0 3063
SB_sub_5_12_g1323_6783/Y	XOR2XL	1	0.2	22	+121	3184 R
g1865_6783/A						+0 3184
g1865_6783/Y	MX2XL	2	0.4	29	+144	3328 R
B/data_in[15]						
data_out_reg[15]/SI	<< SDFFQX1					+0 3328
data_out_reg[15]/CK	setup					100 +204 3532 R
(clock clk)						
	capture					10000 R
	uncertainty					-10 9990 R
Cost Group : 'clk' (path_group 'clk')						
Timing slack : 6458ps						
Start-point : uut1/A/data_out_reg[0]/CK						
End-point : uut1/B/data_out_reg[15]/SI						

Ln 1, Col 1 | 4,532 characters | 60% | Unix (LF) | UTF-8

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

Synthesis effort: High

```
gcd_timing.rep
```

File Edit View

```
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:03:31 pm
Module: GCD_mult
Technology library: slow_vddiv0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
```

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	launch					0 R
uut1_A_data_out_reg[0]/CK	SDFFXQX1	5	1.0	32	+230	230 R
uut1_A_data_out_reg[0]/Q					+0	230
g3083_2398/A	MX2XL	2	0.4	29	+150	379 R
g3083_2398/Y					+0	379
uut1_SB_sub_5_12_g448_1881/AN	NAND2BX1	2	1.0	40	+77	457 R
uut1_SB_sub_5_12_g448_1881/Y					+0	457
uut1_SB_sub_5_12_g445_8246/CI	ADDFX1	1	0.6	39	+186	643 R
uut1_SB_sub_5_12_g445_8246/CO					+0	643
uut1_SB_sub_5_12_g444_5122/CI	ADDFX1	1	0.6	39	+186	829 R
uut1_SB_sub_5_12_g444_5122/CO					+0	829
uut1_SB_sub_5_12_g443_1705/CI	ADDFX1	1	0.6	39	+186	1015 R
uut1_SB_sub_5_12_g443_1705/CO					+0	1015
uut1_SB_sub_5_12_g442_2802/CI	ADDFX1	1	0.6	39	+186	1201 R
uut1_SB_sub_5_12_g442_2802/CO					+0	1201
uut1_SB_sub_5_12_g441_1617/CI	ADDFX1	1	0.6	39	+186	1387 R
uut1_SB_sub_5_12_g441_1617/CO					+0	1387
uut1_SB_sub_5_12_g440_3680/CI	ADDFX1	1	0.6	39	+186	1573 R
uut1_SB_sub_5_12_g440_3680/CO					+0	1573
uut1_SB_sub_5_12_g439_6783/CI	ADDFX1	1	0.6	39	+186	1759 R
uut1_SB_sub_5_12_g439_6783/CO					+0	1759
uut1_SB_sub_5_12_g438_5526/CI	ADDFX1	1	0.6	39	+186	1945 R
uut1_SB_sub_5_12_g438_5526/CO					+0	1945
uut1_SB_sub_5_12_g437_8428/CI	ADDFX1	1	0.6	39	+186	2131 R
uut1_SB_sub_5_12_g437_8428/CO					+0	2131
uut1_SB_sub_5_12_g436_4319/CI	ADDFX1	1	0.6	39	+186	2317 R
uut1_SB_sub_5_12_g436_4319/CO					+0	2317
uut1_SB_sub_5_12_g435_6260/CI	ADDFX1	1	0.6	39	+186	2503 R
uut1_SB_sub_5_12_g435_6260/CO					+0	2503
uut1_SB_sub_5_12_g434_5107/CI	ADDFX1	1	0.6	39	+186	2689 R
uut1_SB_sub_5_12_g434_5107/CO					+0	2689
uut1_SB_sub_5_12_g433_2398/CI	ADDFX1	1	0.6	39	+186	2875 R
uut1_SB_sub_5_12_g433_2398/CO					+0	2875
uut1_SB_sub_5_12_g432_5477/CI	ADDFX1	1	0.4	35	+184	3059 R
uut1_SB_sub_5_12_g432_5477/CO					+0	3059
uut1_SB_sub_5_12_g431_6417/B					+0	3180
uut1_SB_sub_5_12_g431_6417/Y	XOR2XL	1	0.2	22	+121	3180 R
g1599_4733/A					+0	3180
g1599_4733/Y	MX2XL	2	0.4	29	+144	3324 R
uut1_A_data_out_reg[15]/SI	<<	SDFFXQX1			+0	3324
uut1_A_data_out_reg[15]/CK	setup				100 +204	3527 R
(clock clk)	capture uncertainty				-10	9990 R

```
Cost Group : 'clk' (path_group 'clk')
Timing slack : 6463ps
Start-point : uut1_A_data_out_reg[0]/CK
End-point   : uut1_A_data_out_reg[15]/SI
```

Ln 1, Col 1 | 4,585 characters | 60% | Unix (LF) | UTF-8

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

d) GCD computer_area_created file

Synthesis effort: Medium

```
=====
Generated by:          Genus(TM) Synthesis Solution 21.14-s082_1
Generated on:         Sep 12 2024 03:08:21 pm
Module:               GCD_mult
Technology library:   slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode:        enclosed
Area mode:            timing library
=====

Instance     Module      Cell Count  Cell Area  Net Area  Total Area  Wireload
-----+-----+-----+-----+-----+-----+-----+-----+
GCD_mult           217       642.276    0.000    642.276 <none> (D)
  uut1   GCD_datapath  179       551.304    0.000    551.304 <none> (D)
    A    PIPO             16       120.384    0.000    120.384 <none> (D)
    B    PIPO_38          16       120.384    0.000    120.384 <none> (D)
  uut2   controller      38       90.972    0.000     90.972 <none> (D)
(D) = wireload is default in technology library

Ln 4, Col 1 | 1 of 1,068 characters | 60% | Unix (LF) | UTF-8
```

Synthesis effort: High

```
=====
Generated by:          Genus(TM) Synthesis Solution 21.14-s082_1
Generated on:         Sep 12 2024 03:03:31 pm
Module:               GCD_mult
Technology library:   slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode:        enclosed
Area mode:            timing library
=====

Instance Module  Cell Count  Cell Area  Net Area  Total Area  Wireload
-----+-----+-----+-----+-----+-----+-----+
GCD_mult          210       633.384    0.000    633.384 <none> (D)
(D) = wireload is default in technology library

Ln 14, Col 5 | 726 characters | 60% | Unix (LF) | UTF-8
```

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

e) GCD computer_power_created file

Synthesis effort: Medium

gcd_power.rep

File Edit View

Instance: /GCD_mult
Power Unit: W
PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	4.14102e-09	1.44255e-05	9.77810e-07	1.54074e-05	53.95%
latch	6.80845e-10	5.96203e-07	5.03486e-07	1.10037e-06	3.85%
logic	6.43104e-09	8.80464e-06	2.67392e-06	1.14850e-05	40.21%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	5.67000e-07	5.67000e-07	1.99%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.12529e-08	2.38263e-05	4.72222e-06	2.85598e-05	100.00%
Percentage	0.04%	83.43%	16.53%	100.00%	100.00%

Ln 1, Col 1 | 1,202 characters | 60% | Unix (LF) | UTF-8

Synthesis effort: High

gcd_power.rep

File Edit View

Instance: /GCD_mult
Power Unit: W
PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	4.13550e-09	1.35555e-05	6.97469e-07	1.42571e-05	53.58%
latch	6.81032e-10	7.98210e-07	5.72306e-07	1.37120e-06	5.15%
logic	6.91522e-09	7.99946e-06	2.40811e-06	1.04145e-05	39.14%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	5.67000e-07	5.67000e-07	2.13%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.17318e-08	2.23532e-05	4.24488e-06	2.66098e-05	100.00%
Percentage	0.04%	84.00%	15.95%	100.00%	100.00%

Ln 1, Col 1 | 1,202 characters | 80% | Unix (LF) | UTF-8

LAB4 2: Logic synthesis of a GCD computation using Datapath and Control path logic

Inference:

Synthesis effort: Medium

A total of 217 leaf instance count is present in the gate level netlist with total area of 642.276, total power of 2.66098e-05 W.

Synthesis effort: High

A total of 210 leaf instance count is present in the gate level netlist with total area of 633.384, total power of 2.85598e-05W.

Result: Hence a gcd computer using Datapath and Control path logic is synthesized and the gate level netlist with timing, area and power report has been generated.

Lab4 3: Design and simulation of a Booth Multiplier using Datapath and Control path logic

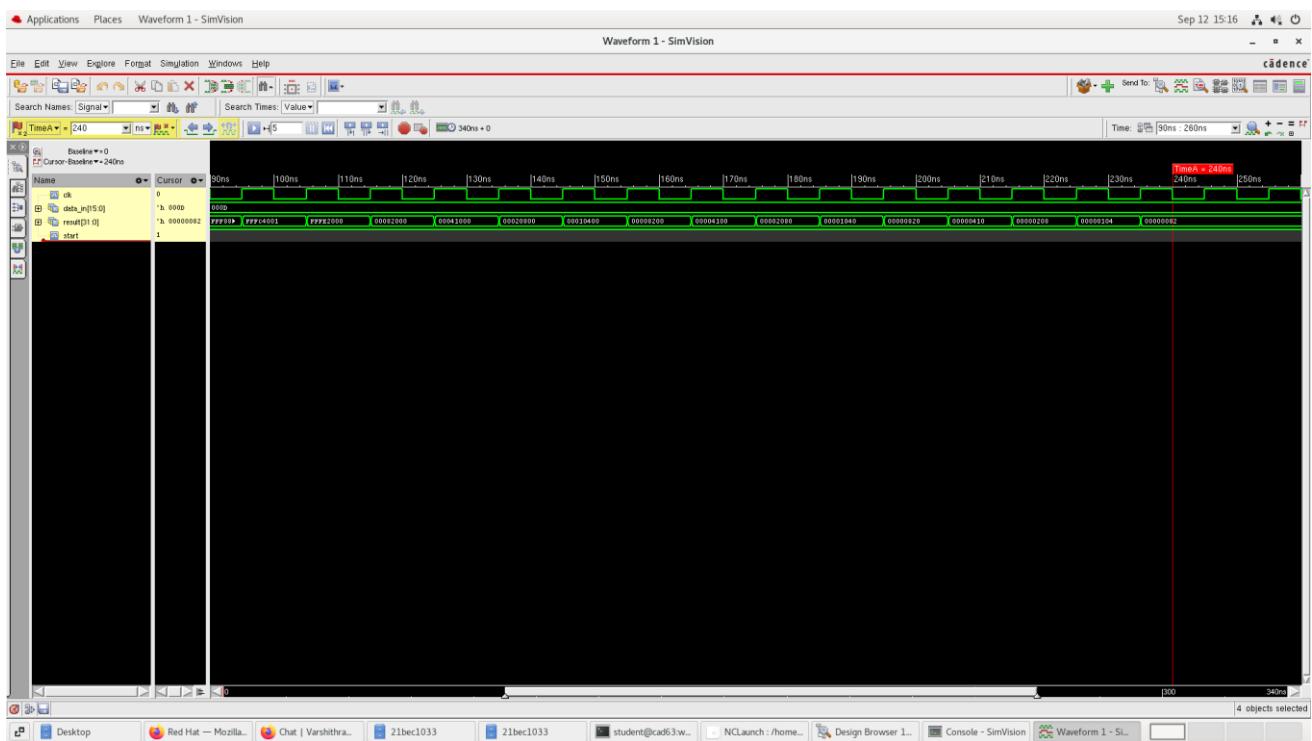
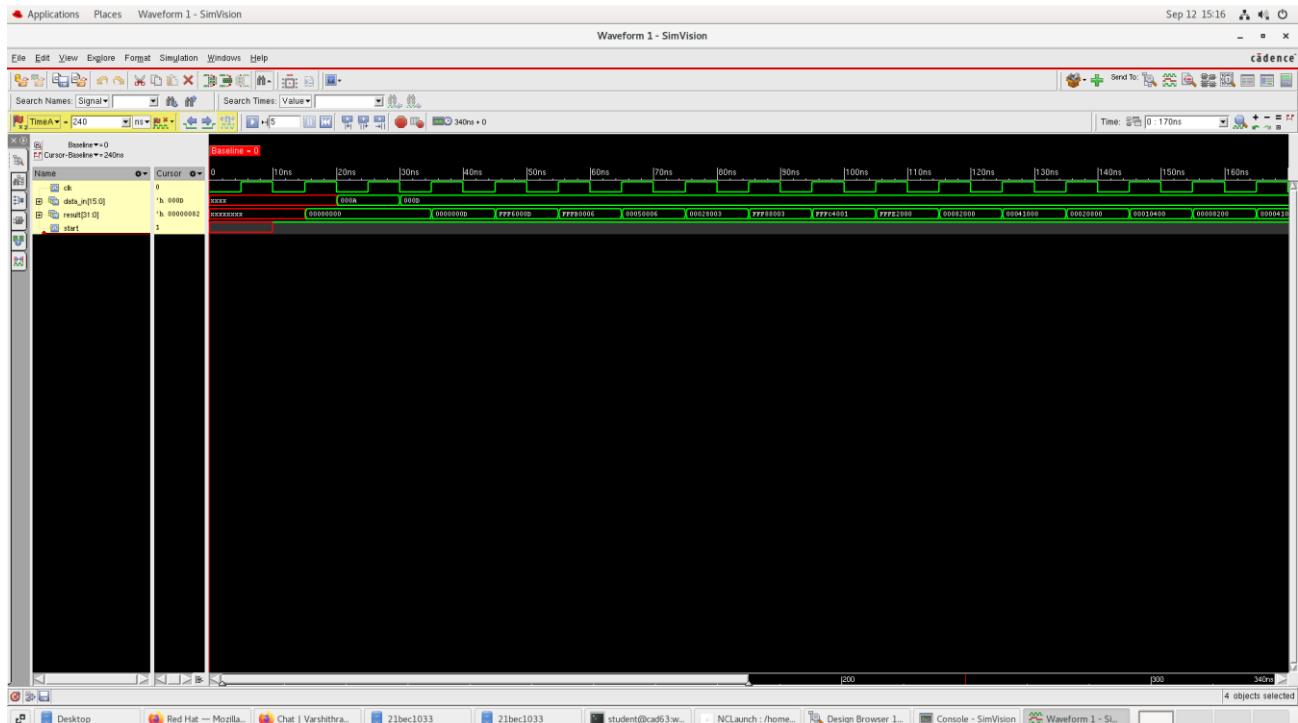
NCLaunch Compilation and Elaboration:

The screenshot shows the NCLaunch graphical user interface. The top menu bar includes File, Edit, Tools, Utilities, Plug-Ins, and Help. Below the menu is a toolbar with various icons for file operations. The main workspace has two panes: a left pane showing a file tree with files like waves.shm, ALU.v, BOOTH.v, BOOTH_test.v, PIP0.v, controller.v, counter.v, dff.v, and shiftreg.v; and a right pane showing a hierarchical view of modules under worklib, such as ALU, BOOTH, BOOTH_test, PIP0, controller, counter, dff, and shiftreg, with sub-modules like worklib.ALU:module and worklib.BOOTH:module highlighted. At the bottom, a terminal window displays the command-line logs for the compilation and elaboration process, showing Cadence nclog and ncelab commands for each module.

```
nclaunch> nclog -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile nclog.log -errormax 15 -update -linedebug -status /home/student/Desktop/21bec1033/rtl/ALU.v /home/student/Desktop/21bec1033/rtl/ALU.v  
nclog(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
nclog: Memory Usage - 21.4M program + 28.7M data = 50.1M total  
nclog: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.1s, 7.0% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.ALU  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.5M data = 82.4M total (Peak 87.7M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.7% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.BOOTH  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.8M data = 82.7M total (Peak 87.8M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 11.4% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.BOOTH_test  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.8M data = 82.7M total (Peak 87.8M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 10.0% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.PIP0  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.5M data = 82.4M total (Peak 87.7M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.7% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.controller  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.5M data = 82.4M total (Peak 87.6M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.1s, 19.4% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.counter  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.6M data = 82.6M total (Peak 87.7M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.4% cpu)  
nclaunch> ncelab -work worklib -cdslib /home/student/Desktop/21bec1033/work/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.dff  
ncelab(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.  
ncelab: Memory Usage - 49.9M program + 32.6M data = 82.5M total (Peak 87.7M)  
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 9.5% cpu)
```

Lab4 3: Design and simulation of a Booth Multiplier using Datapath and Control path logic

Observations/Simulation Waveforms:



Lab4 3: Design and simulation of a Booth Multiplier using Datapath and Control path logic

Inference: In the above output waveform, the data_in is provided with 13 & 10 values and the data_out is obtained as 130 which is the multiplied value. Output “done” is equal to 1 whenever booth multiplier result is obtained.

Result: Hence, a Booth multiplier is designed and the simulation and verification of behavior of booth multiplier using data path and Control path logic is performed for the given stimuli in the test bench.

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

Aim: To synthesize a booth multiplier using Datapath and Control path logic and to get the gate level netlist with timing, area, and power reports.

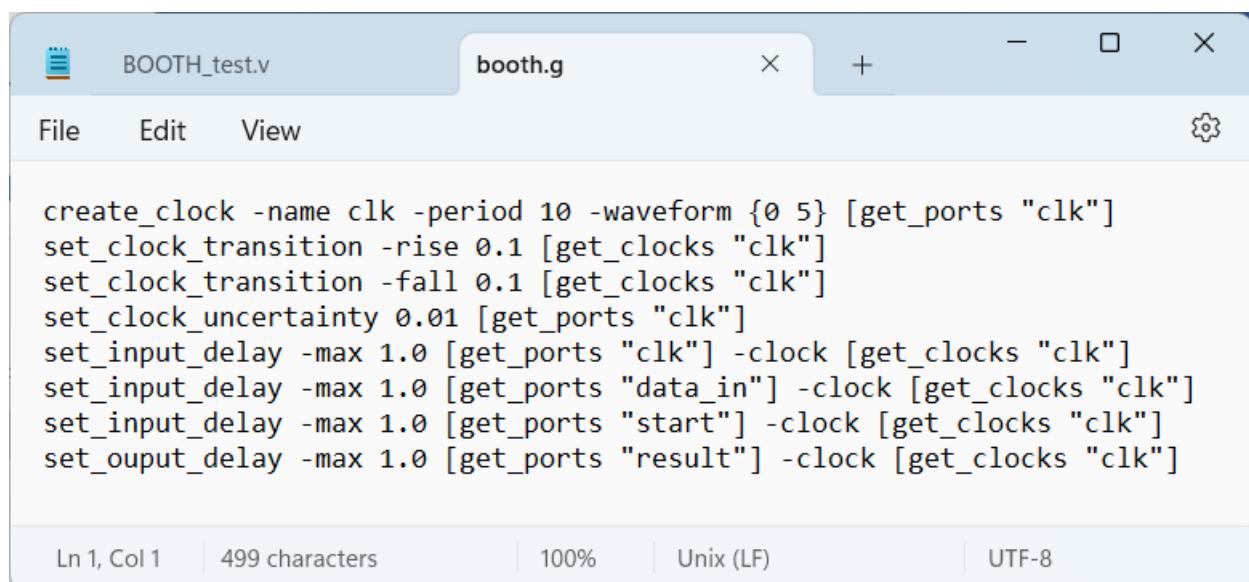
EDA Tools Used: Cadence Genus

Description:

Datapath is the hardware that performs all the required operations. Control is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc. Simple datapath components include memory (stores the current instruction), PC or program counter (stores the address of current instruction), and ALU (executes current instruction). Control path is the Finite state machine designed by using Moore or Mealy models.

Procedure:

1. Copy the fast.lib and slow.lib files into the folder where (.v) files are located.
2. Create a Synopsys design constraint file (.g file) by entering the design constraints required for the synthesis.

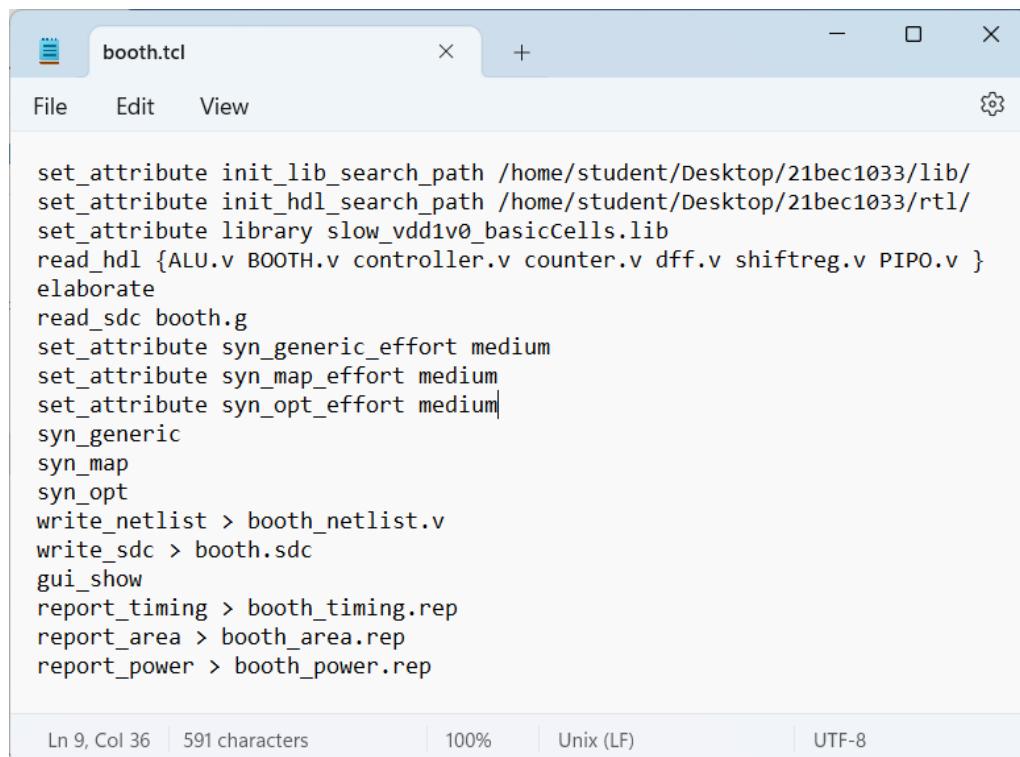


```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 0.01 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "clk"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "data_in"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "start"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "result"] -clock [get_clocks "clk"]
```

Ln 1, Col 1 | 499 characters | 100% | Unix (LF) | UTF-8

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

3. Create a (.tcl) file containing all the commands for performing the logic synthesis. Synthesis effort can be medium or high.



The screenshot shows a Windows-style text editor window with the title bar 'booth.tcl'. The menu bar includes 'File', 'Edit', and 'View'. The main area contains the following Tcl script:

```
set_attribute init_lib_search_path /home/student/Desktop/21bec1033/lib/
set_attribute init_hdl_search_path /home/student/Desktop/21bec1033/rtl/
set_attribute library slow_vdd1v0_basicCells.lib
read_hdl {ALU.v BOOTH.v controller.v counter.v dff.v shiftreg.v PIPO.v }
elaborate
read_sdc booth.sdc
set_attribute syn_generic_effort medium
set_attribute syn_map_effort medium
set_attribute syn_opt_effort medium
syn_generic
syn_map
syn_opt
write_netlist > booth_netlist.v
write_sdc > booth.sdc
gui_show
report_timing > booth_timing.rep
report_area > booth_area.rep
report_power > booth_power.rep
```

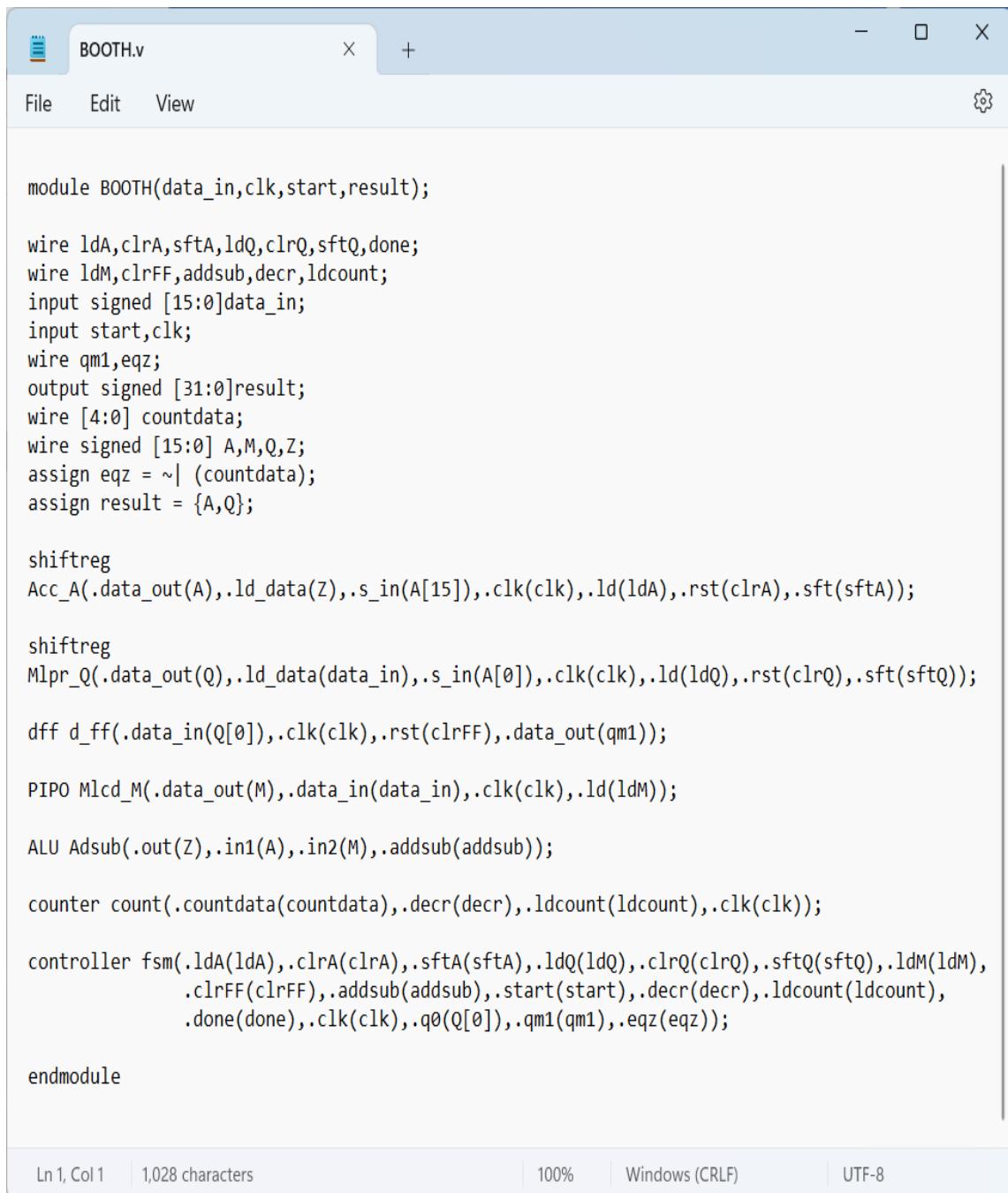
At the bottom of the window, status indicators show 'Ln 9, Col 36 | 591 characters', '100%', 'Unix (LF)', and 'UTF-8'.

4. Invoke the C shell and launch the Genus tool by entering the below commands.
‘csh’
‘source /home/install/cshrc’
‘genus -legacy -ui -f booth.tcl’
5. Execute the commands in the (.tcl file) by entering **source booth.tcl** command in the command line window.
6. Check for the area, timing and power reports generated in the respective multiplier folder. Also check the gate level netlist generated in the Genus synthesis solution window.

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

Verilog Programs:

//Verilog Program of top-level module containing data path and control path



The screenshot shows a Verilog code editor window titled "BOOTH.v". The code defines a module named "BOOTH" with inputs "data_in", "clk", and "start", and output "result". It uses several components: Acc_A, Mlpr_Q, dff, PIP0, and controller fsm. The code is as follows:

```
module BOOTH(data_in,clk,start,result);

    wire ldA,clrA,sftA,ldQ,clrQ,sftQ,done;
    wire ldM,clrFF,addsub,decr,ldcount;
    input signed [15:0]data_in;
    input start,clk;
    wire qm1,eqz;
    output signed [31:0]result;
    wire [4:0] countdata;
    wire signed [15:0] A,M,Q,Z;
    assign eqz = ~| (countdata);
    assign result = {A,Q};

    shiftreg
    Acc_A(.data_out(A),.ld_data(Z),.s_in(A[15]),.clk(clk),.ld(ldA),.rst(clrA),.sft(sftA));

    shiftreg
    Mlpr_Q(.data_out(Q),.ld_data(data_in),.s_in(A[0]),.clk(clk),.ld(ldQ),.rst(clrQ),.sft(sftQ));

    dff d_ff(.data_in(Q[0]),.clk(clk),.rst(clrFF),.data_out(qm1));

    PIP0 Mlcd_M(.data_out(M),.data_in(data_in),.clk(clk),.ld(ldM));

    ALU Adsub(.out(Z),.in1(A),.in2(M),.addsub(addsub));

    counter count(.countdata(countdata),.decr(decr),.ldcount(ldcount),.clk(clk));

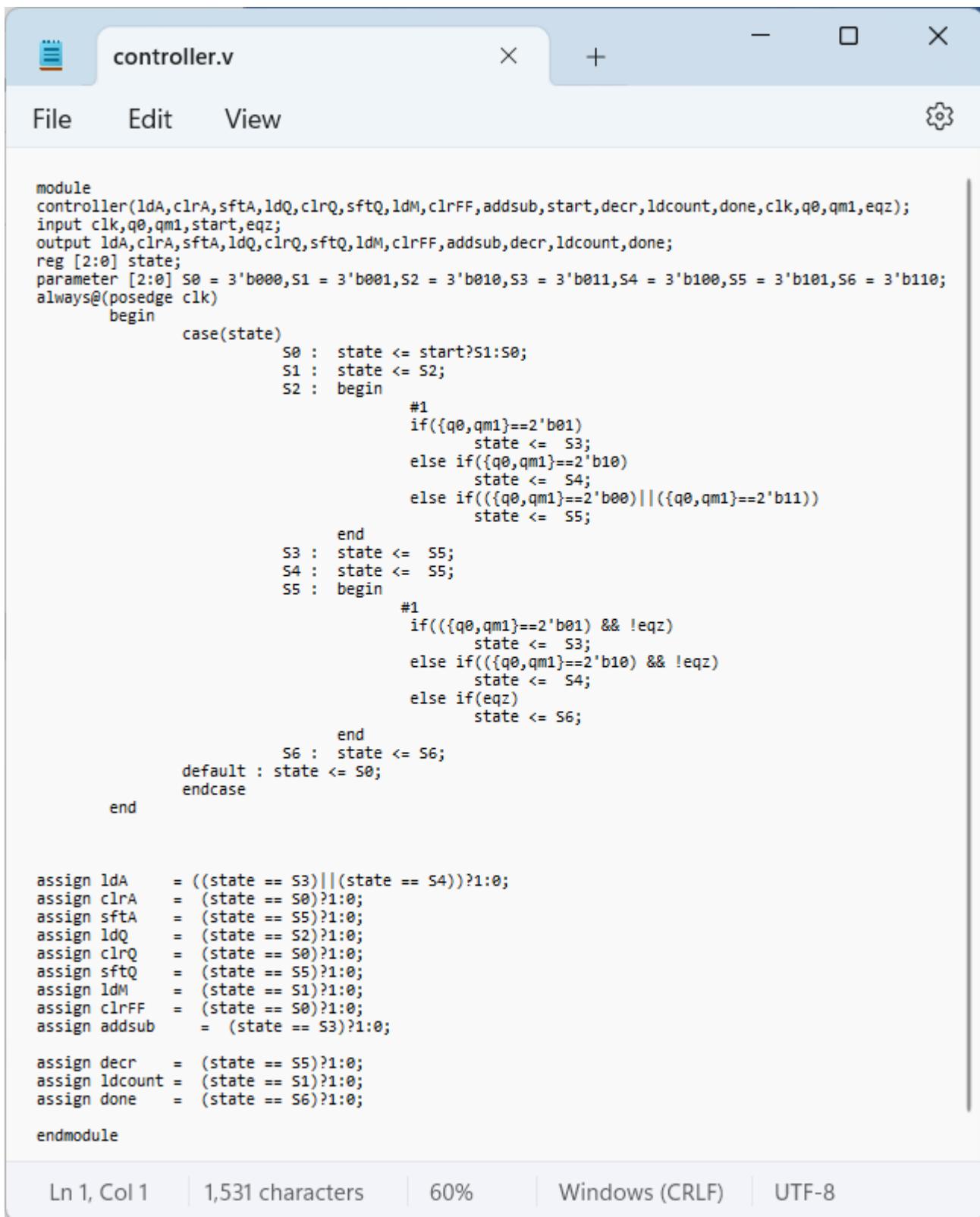
    controller fsm(.ldA(ldA),.clrA(clrA),.sftA(sftA),.ldQ(ldQ),.clrQ(clrQ),.sftQ(sftQ),.ldM(ldM),
                  .clrFF(clrFF),.addsub(addsub),.start(start),.decr(decr),.ldcount(ldcount),
                  .done(done),.clk(clk),.q0(Q[0]),.qm1(qm1),.eqz(eqz));

endmodule
```

At the bottom of the editor, status information includes: Ln 1, Col 1 | 1,028 characters | 100% | Windows (CRLF) | UTF-8.

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

//Verilog Program of control path module



The screenshot shows a code editor window titled "controller.v". The file contains a Verilog module definition for a controller. The module has inputs clk, q0, qm1, start, eqz; and outputs ldA, clrA, sftA, ldQ, clrQ, sftQ, ldM, clrFF, addsub, start, decr, ldcount, done. It features a parameter section with states S0 through S6. The main body includes a begin-end block with a case statement for state. The case block handles transitions based on q0 and qm1 values, and includes assignments for ldA, clrA, etc. The code ends with an endmodule statement. The status bar at the bottom shows "Ln 1, Col 1", "1,531 characters", "60%", "Windows (CRLF)", and "UTF-8".

```
module controller(ldA,clrA,sftA,ldQ,clrQ,sftQ,ldM,clrFF,addsub,start,decr,ldcount,done,clk,q0,qm1,eqz);
input clk,q0,qm1,start,eqz;
output ldA,clrA,sftA,ldQ,clrQ,sftQ,ldM,clrFF,addsub,decr,ldcount,done;
reg [2:0] state;
parameter [2:0] S0 = 3'b000,S1 = 3'b001,S2 = 3'b010,S3 = 3'b011,S4 = 3'b100,S5 = 3'b101,S6 = 3'b110;
always@(posedge clk)
begin
    case(state)
        S0 : state <= start?S1:S0;
        S1 : state <= S2;
        S2 : begin
            #1
            if({q0,qm1}==2'b01)
                state <= S3;
            else if({q0,qm1}==2'b10)
                state <= S4;
            else if(({q0,qm1}==2'b00)||({q0,qm1}==2'b11))
                state <= S5;
            end
        S3 : state <= S5;
        S4 : state <= S5;
        S5 : begin
            #1
            if(({q0,qm1}==2'b01) && !eqz)
                state <= S3;
            else if(({q0,qm1}==2'b10) && !eqz)
                state <= S4;
            else if(eqz)
                state <= S6;
            end
        S6 : state <= S6;
        default : state <= S0;
    endcase
end

assign ldA    = ((state == S3)|| (state == S4))?1:0;
assign clrA   = (state == S0)?1:0;
assign sftA   = (state == S5)?1:0;
assign ldQ    = (state == S2)?1:0;
assign clrQ   = (state == S0)?1:0;
assign sftQ   = (state == S5)?1:0;
assign ldM    = (state == S1)?1:0;
assign clrFF  = (state == S0)?1:0;
assign addsub = (state == S3)?1:0;

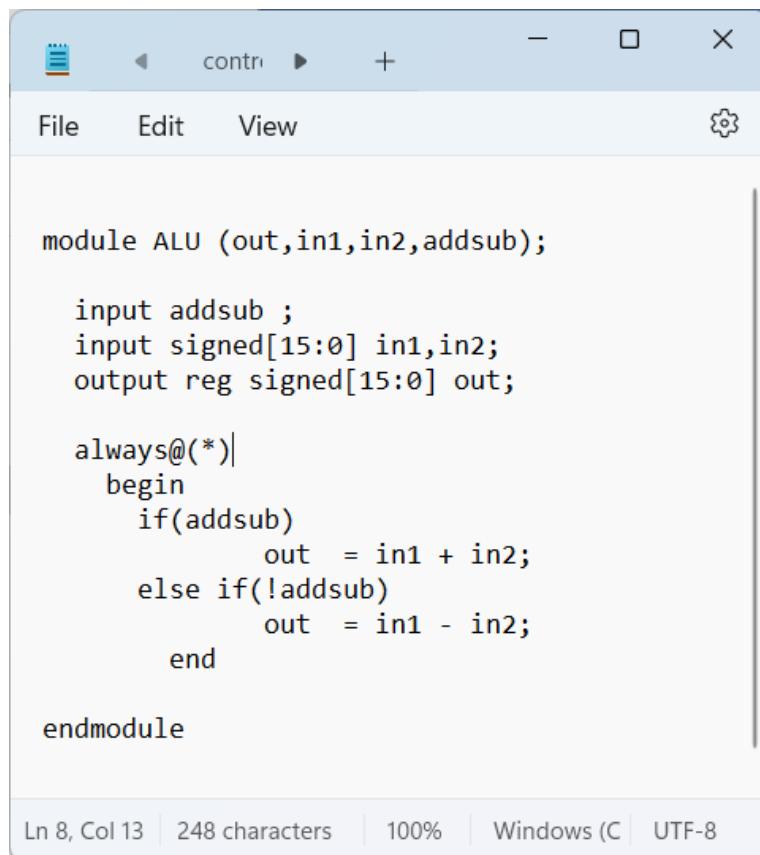
assign decr   = (state == S5)?1:0;
assign ldcount = (state == S1)?1:0;
assign done    = (state == S6)?1:0;

endmodule
```

Ln 1, Col 1 | 1,531 characters | 60% | Windows (CRLF) | UTF-8

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

//Verilog Program of ALU



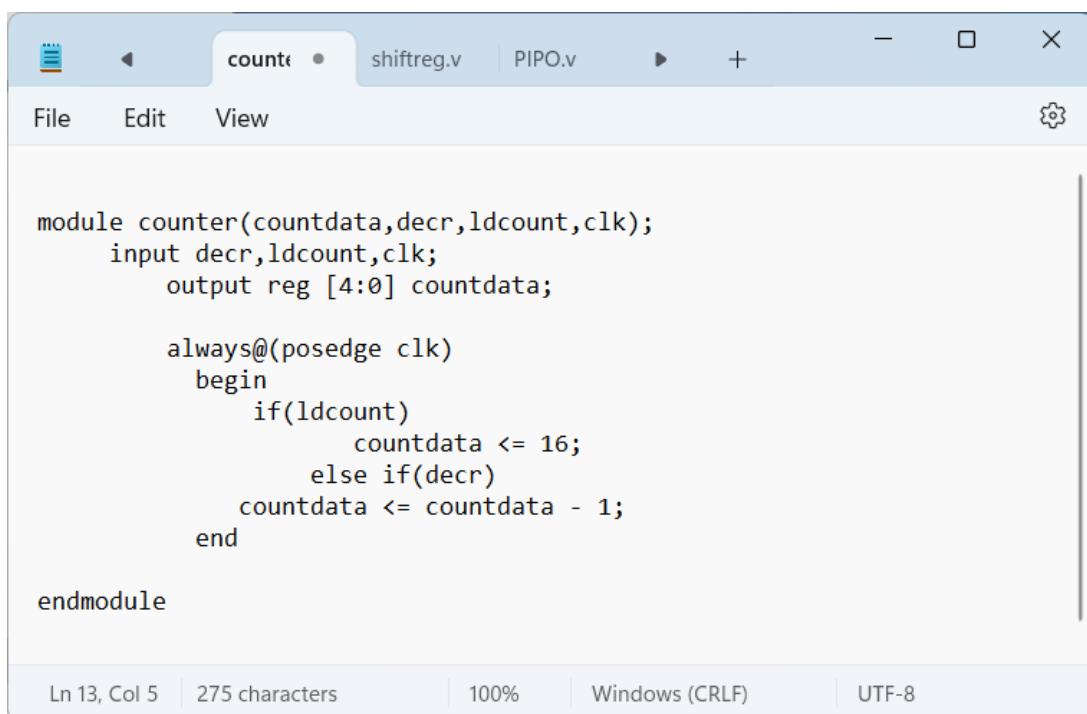
```
module ALU (out,in1,in2,addsub);
    input addsub ;
    input signed[15:0] in1,in2;
    output reg signed[15:0] out;

    always@(*)
        begin
            if(addsub)
                out = in1 + in2;
            else if(!addsub)
                out = in1 - in2;
        end

    endmodule
```

Ln 8, Col 13 | 248 characters | 100% | Windows (C) | UTF-8

//Verilog Program of counter



```
module counter(countdata,decr,ldcount,clk);
    input decre,ldcount,clk;
    output reg [4:0] countdata;

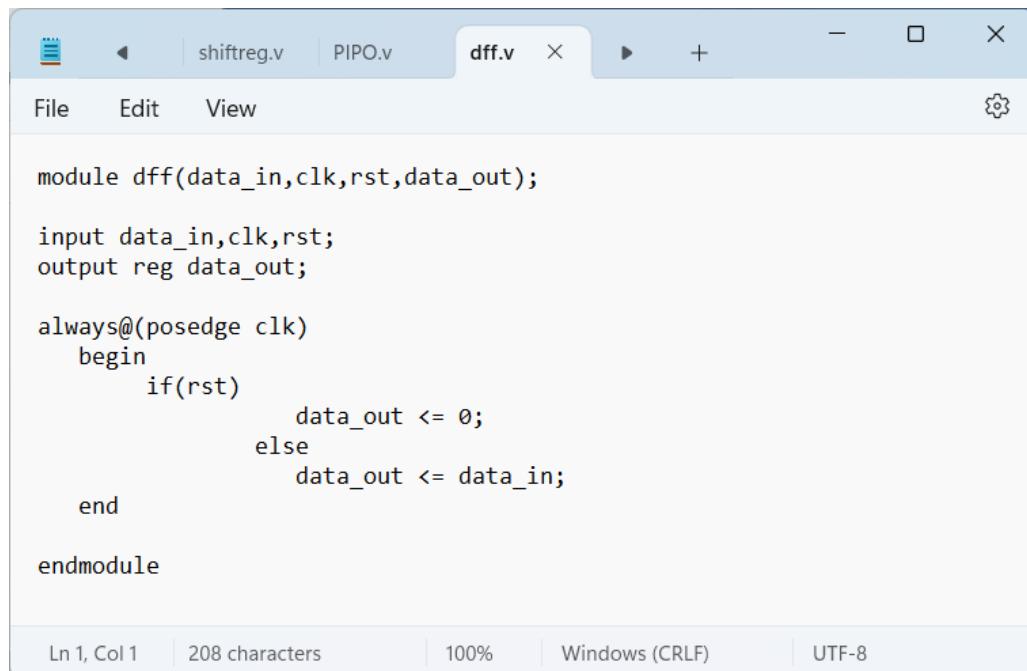
    always@(posedge clk)
        begin
            if(ldcount)
                countdata <= 16;
            else if(decr)
                countdata <= countdata - 1;
        end

    endmodule
```

Ln 13, Col 5 | 275 characters | 100% | Windows (CRLF) | UTF-8

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

//Verilog Program of D-FF

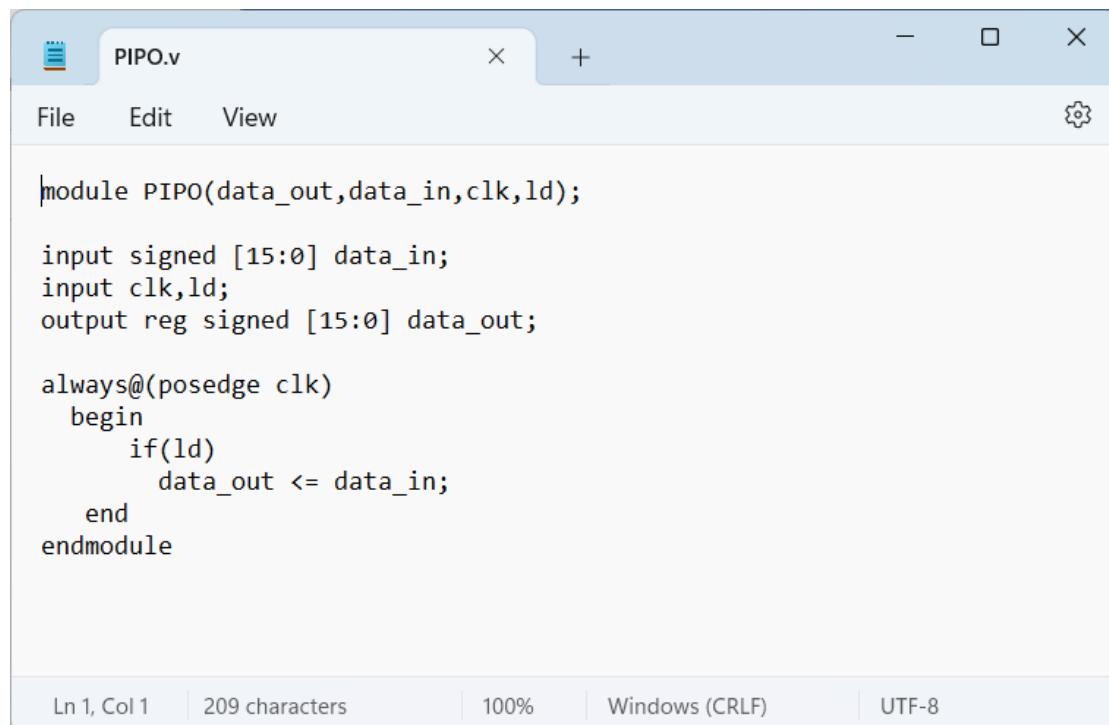


A screenshot of a Verilog code editor window titled "dff.v". The code defines a D flip-flop (D-FF) module with inputs data_in, clk, and rst, and an output data_out. It uses an always@(posedge clk) block to update the output based on the input data_in and the reset signal rst.

```
module dff(data_in,clk,rst,data_out);
    input data_in,clk,rst;
    output reg data_out;
    always@(posedge clk)
        begin
            if(rst)
                data_out <= 0;
            else
                data_out <= data_in;
        end
    endmodule
```

Ln 1, Col 1 | 208 characters | 100% | Windows (CRLF) | UTF-8

//Verilog Program of PIPO



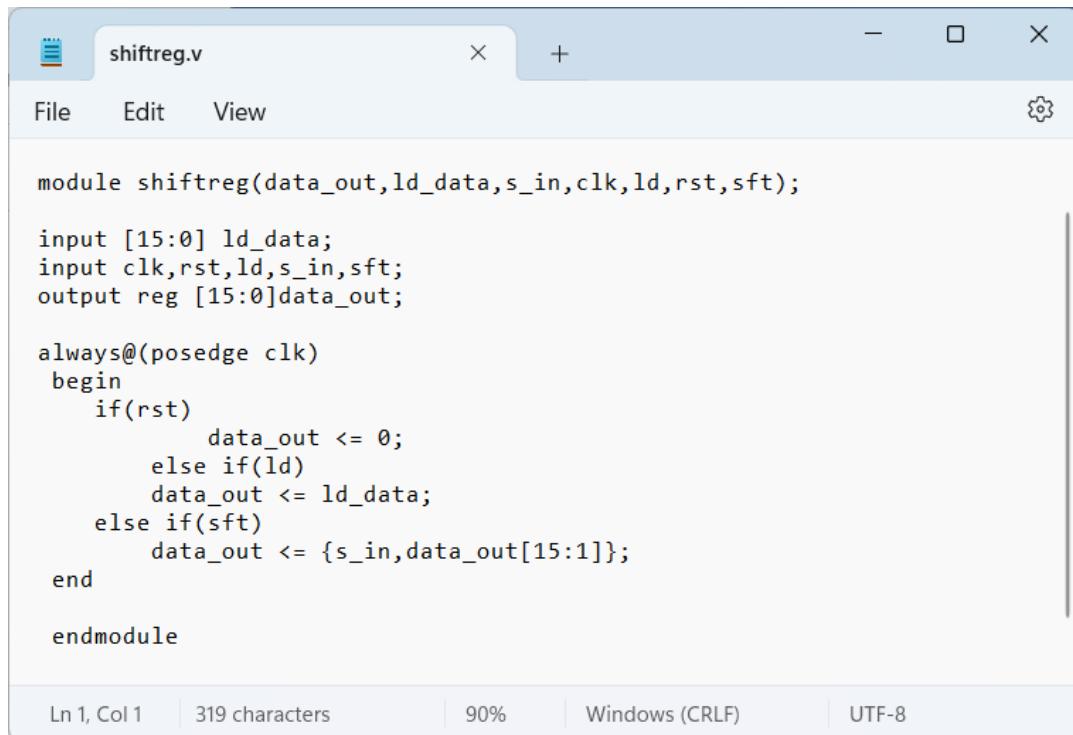
A screenshot of a Verilog code editor window titled "PIPO.v". The code defines a PIPO (Pipeline Register) module with inputs data_in, clk, and ld, and an output data_out. It uses an always@(posedge clk) block to update the output based on the input data_in and the load signal ld.

```
module PIPO(data_out,data_in,clk,ld);
    input signed [15:0] data_in;
    input clk,ld;
    output reg signed [15:0] data_out;
    always@(posedge clk)
        begin
            if(ld)
                data_out <= data_in;
        end
    endmodule
```

Ln 1, Col 1 | 209 characters | 100% | Windows (CRLF) | UTF-8

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

//Verilog Program of shiftregister



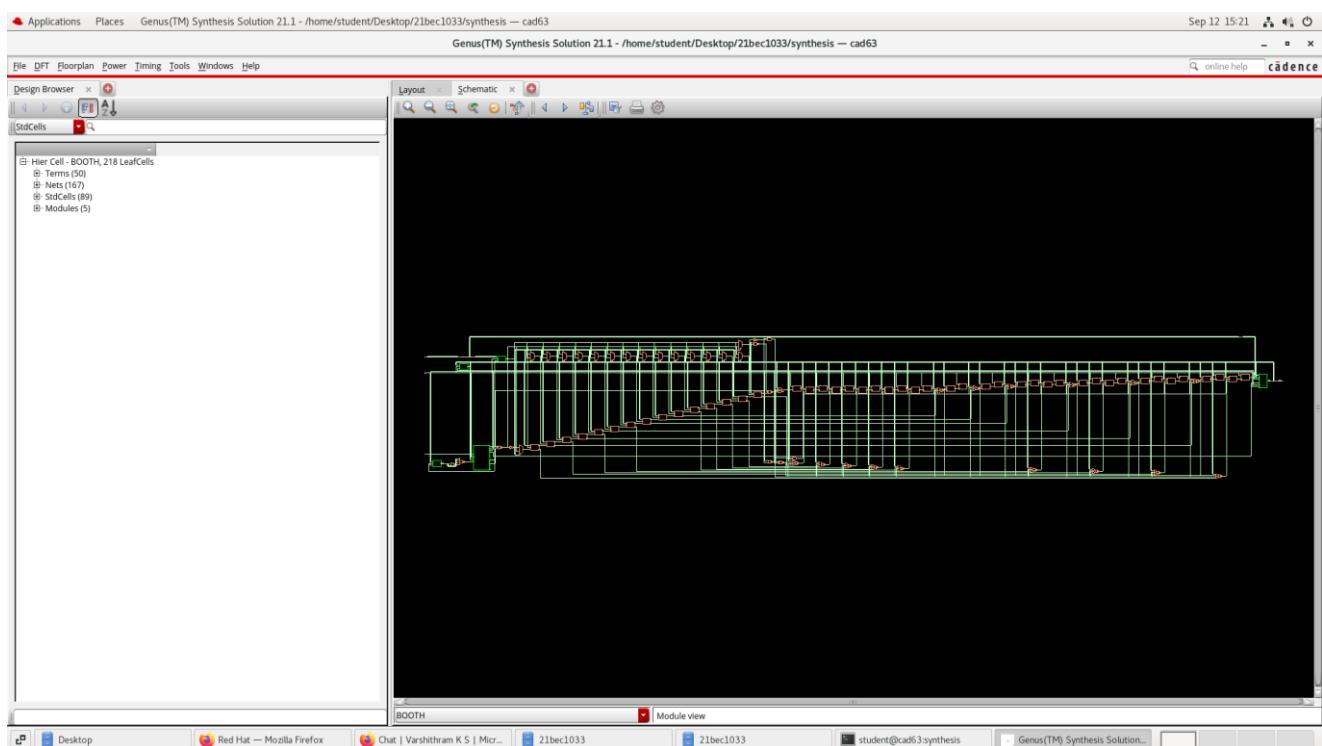
```
shiftreg.v
```

```
module shiftreg(data_out,ld_data,s_in,clk,ld,rst,sft);  
    input [15:0] ld_data;  
    input clk,rst,ld,s_in,sft;  
    output reg [15:0]data_out;  
  
    always@(posedge clk)  
    begin  
        if(rst)  
            data_out <= 0;  
        else if(ld)  
            data_out <= ld_data;  
        else if(sft)  
            data_out <= {s_in,data_out[15:1]};  
    end  
  
endmodule
```

Ln 1, Col 1 | 319 characters | 90% | Windows (CRLF) | UTF-8

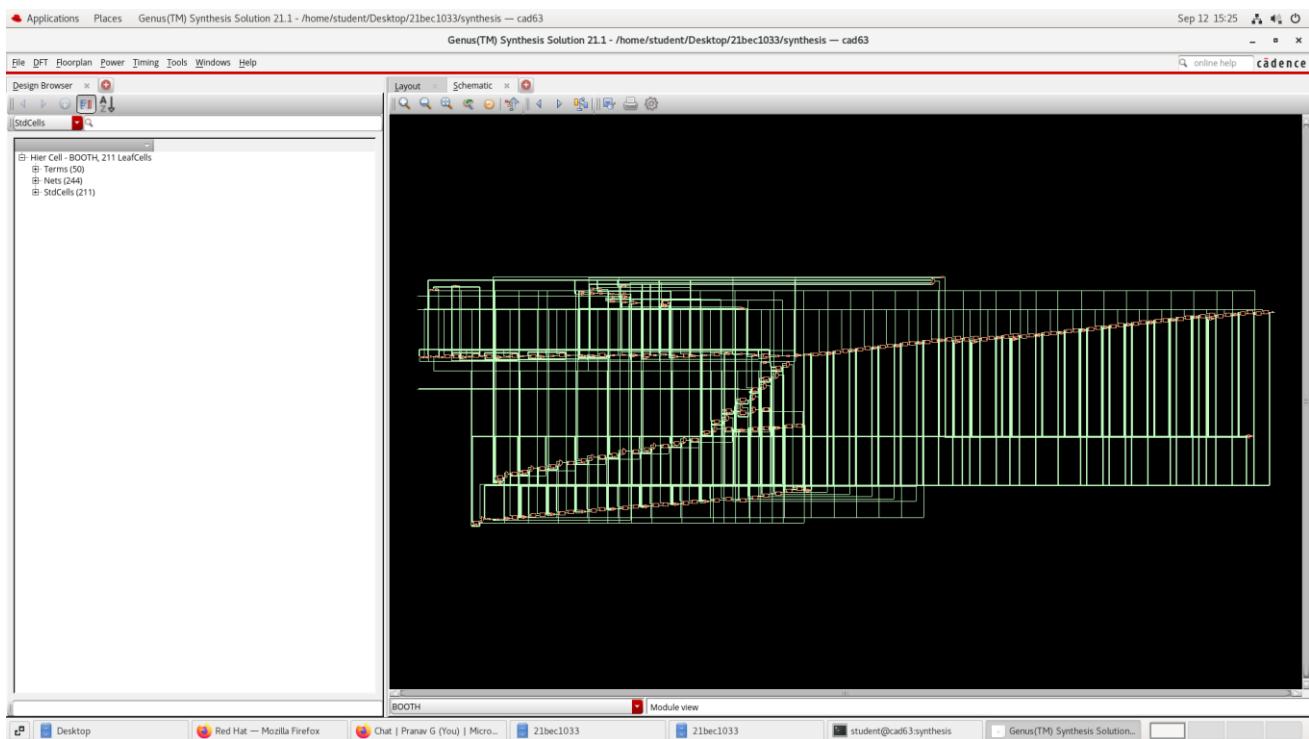
Gate level Netlist:

Synthesis effort: Medium



LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

Synthesis effort : High



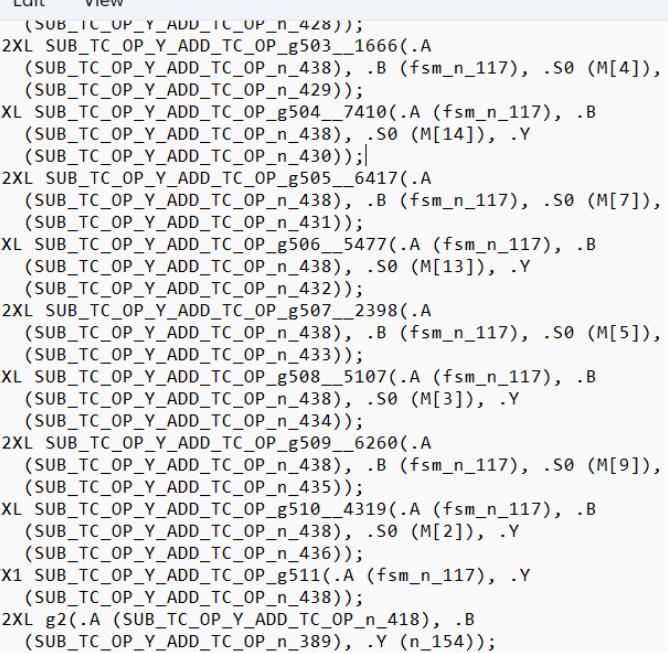
LAB4_2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

Observations:

a) Booth_multiplier_netlist_created file

LAB4_2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

The screenshot shows a dual-monitor setup. The left monitor displays a file editor window with code in a programming language, likely C or C++, with syntax highlighting. The right monitor also displays a file editor window with the same code, but it appears to be a decompiled or interpreted version, as the syntax is less structured. Both monitors have standard Windows taskbars at the bottom.

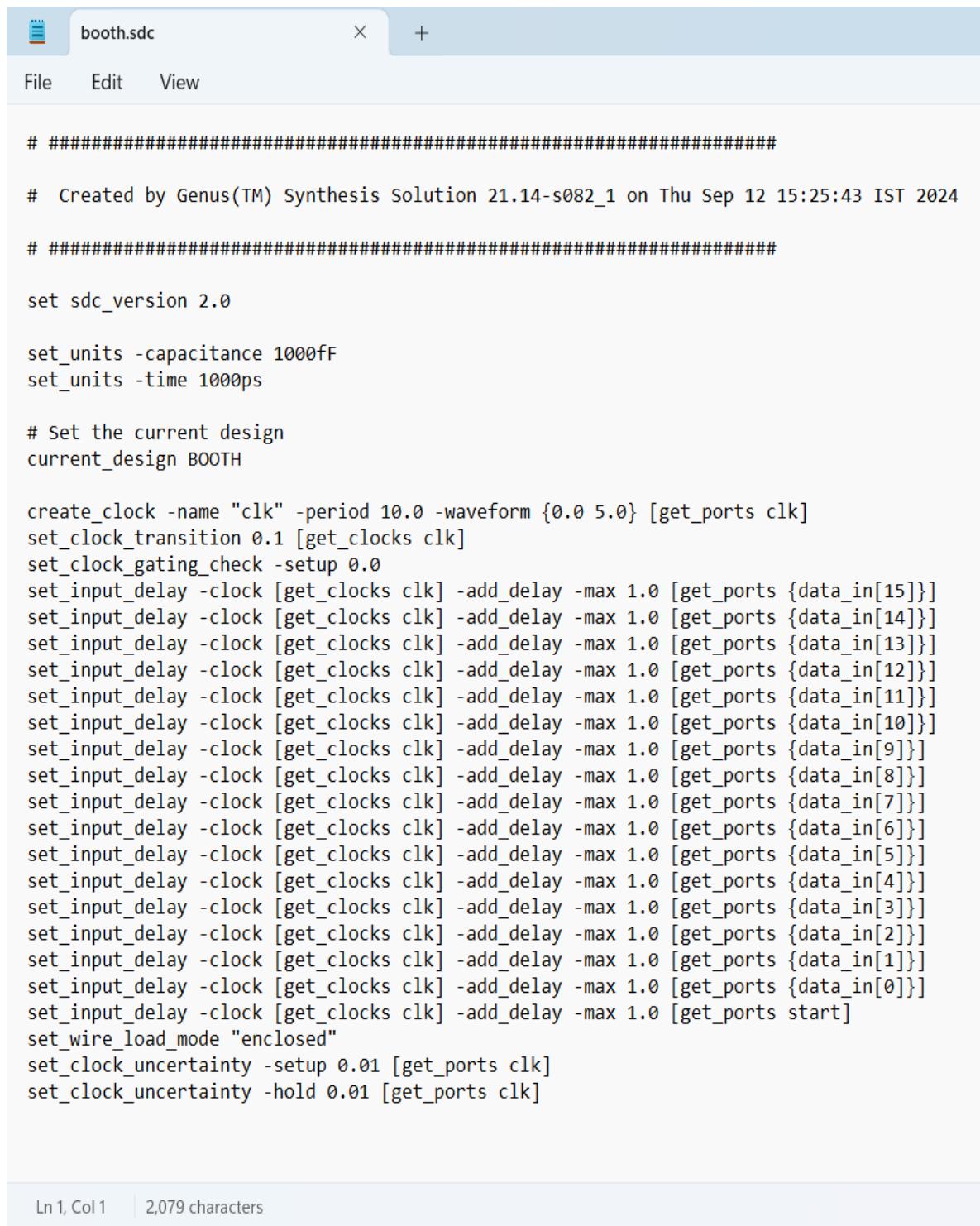


The screenshot shows a text editor window with the title "booth_netlist.v". The menu bar includes "File", "Edit", and "View". The status bar at the bottom indicates "Ln 455, Col 39 | 23,005 characters | 90% | Unix (LF) | UTF-8". The main content area contains Verilog code for a booth multiplier module:

```
(SUB_TC_OP_Y_ADD_TC_OP_n_428));
MXI2XL SUB_TC_OP_Y_ADD_TC_OP_g503_1666(.A
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .B (fsm_n_117), .S0 (M[4]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_429));
MX2XL SUB_TC_OP_Y_ADD_TC_OP_g504_7410(.A (fsm_n_117), .B
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .S0 (M[14]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_430));
MXI2XL SUB_TC_OP_Y_ADD_TC_OP_g505_6417(.A
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .B (fsm_n_117), .S0 (M[7]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_431));
MX2XL SUB_TC_OP_Y_ADD_TC_OP_g506_5477(.A (fsm_n_117), .B
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .S0 (M[13]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_432));
MXI2XL SUB_TC_OP_Y_ADD_TC_OP_g507_2398(.A
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .B (fsm_n_117), .S0 (M[5]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_433));
MX2XL SUB_TC_OP_Y_ADD_TC_OP_g508_5107(.A (fsm_n_117), .B
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .S0 (M[3]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_434));
MXI2XL SUB_TC_OP_Y_ADD_TC_OP_g509_6260(.A
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .B (fsm_n_117), .S0 (M[9]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_435));
MX2XL SUB_TC_OP_Y_ADD_TC_OP_g510_4319(.A (fsm_n_117), .B
    (SUB_TC_OP_Y_ADD_TC_OP_n_438), .S0 (M[2]), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_436));
INVX1 SUB_TC_OP_Y_ADD_TC_OP_g511(.A (fsm_n_117), .Y
    (SUB_TC_OP_Y_ADD_TC_OP_n_438));
XOR2XL g2(.A (SUB_TC_OP_Y_ADD_TC_OP_n_418), .B
    (SUB_TC_OP_Y_ADD_TC_OP_n_389), .Y (n_154));
endmodule
```

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

b) Booth_multiplier_constraint_created file



The screenshot shows a software window titled "booth.sdc". The menu bar includes "File", "Edit", and "View". The main area contains the following Genus SDC code:

```
# #####
# Created by Genus(TM) Synthesis Solution 21.14-s082_1 on Thu Sep 12 15:25:43 IST 2024
# #####
set sdc_version 2.0

set_units -capacitance 1000fF
set_units -time 1000ps

# Set the current design
current_design BOOTH

create_clock -name "clk" -period 10.0 -waveform {0.0 5.0} [get_ports clk]
set_clock_transition 0.1 [get_clocks clk]
set_clock_gating_check -setup 0.0
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[15]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[14]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[13]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[12]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[11]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[10]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[9]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[8]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[7]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[6]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[5]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[4]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[3]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[2]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[1]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports {data_in[0]}]
set_input_delay -clock [get_clocks clk] -add_delay -max 1.0 [get_ports start]
set_wire_load_mode "enclosed"
set_clock_uncertainty -setup 0.01 [get_ports clk]
set_clock_uncertainty -hold 0.01 [get_ports clk]
```

At the bottom left, it says "Ln 1, Col 1" and "2,079 characters".

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

c) Booth_multiplier_timing_created file

Synthesis effort: Medium

```
booth_timing.rep                                booth_timing.rep

File      Edit      View

=====
Generated by:      Genus(TM) Synthesis Solution 21.14-s082_1
Generated on:     Sep 12 2024  03:21:21 pm
Module:          BOOTH
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode:    enclosed
Area mode:       timing library
=====

Pin           Type      Fanout Load Slew Delay Arrival
              (fF)   (ps)  (ps)  (ps)
=====
(clock clk)      launch          0 R
fsm
state_reg[0]/CK
state_reg[0]/Q      SDFFQX1      7  1.6  53  +257  257 F
g574_5526/AN
g574_5526/Y      NAND3BXL      2  0.8  200 +201  458 F
g568_5477/A
g568_5477/Y      NAND2X1       4  1.2  73  +142  600 R
fsm/lda
g571_1666/AN
g571_1666/Y      NOR2BX1       18 4.4  218 +215  815 R
SUB_TC_OP_Y_ADD_TC_OP_g1628/A
SUB_TC_OP_Y_ADD_TC_OP_g1628/Y      INVX1       16 3.2  127 +184  998 F
SUB_TC_OP_Y_ADD_TC_OP_g1619_5107/A
SUB_TC_OP_Y_ADD_TC_OP_g1619_5107/Y      MXI2XL      1  0.6  95  +139  1138 R
SUB_TC_OP_Y_ADD_TC_OP_g1610_9315/CI
SUB_TC_OP_Y_ADD_TC_OP_g1610_9315/CO      ADDFX1      1  0.6  39  +215  1352 R
SUB_TC_OP_Y_ADD_TC_OP_g1609_6161/CI
SUB_TC_OP_Y_ADD_TC_OP_g1609_6161/CO      ADDFX1      1  0.6  39  +186  1538 R
SUB_TC_OP_Y_ADD_TC_OP_g1608_4733/CI
SUB_TC_OP_Y_ADD_TC_OP_g1608_4733/CO      ADDFX1      1  0.6  39  +186  1724 R
SUB_TC_OP_Y_ADD_TC_OP_g1607_7482/CI
SUB_TC_OP_Y_ADD_TC_OP_g1607_7482/CO      ADDFX1      1  0.6  39  +186  1910 R
SUB_TC_OP_Y_ADD_TC_OP_g1606_5115/CI
SUB_TC_OP_Y_ADD_TC_OP_g1606_5115/CO      ADDFX1      1  0.6  39  +186  2096 R
SUB_TC_OP_Y_ADD_TC_OP_g1605_1881/CI
SUB_TC_OP_Y_ADD_TC_OP_g1605_1881/CO      ADDFX1      1  0.6  39  +186  2282 R
SUB_TC_OP_Y_ADD_TC_OP_g1604_6131/CI
SUB_TC_OP_Y_ADD_TC_OP_g1604_6131/CO      ADDFX1      1  0.6  39  +186  2468 R
SUB_TC_OP_Y_ADD_TC_OP_g1603_7098/CI
SUB_TC_OP_Y_ADD_TC_OP_g1603_7098/CO      ADDFX1      1  0.6  39  +186  2654 R
SUB_TC_OP_Y_ADD_TC_OP_g1602_8246/CI
SUB_TC_OP_Y_ADD_TC_OP_g1602_8246/CO      ADDFX1      1  0.6  39  +186  2840 R
SUB_TC_OP_Y_ADD_TC_OP_g1601_5122/CI
SUB_TC_OP_Y_ADD_TC_OP_g1601_5122/CO      ADDFX1      1  0.6  39  +186  3026 R
SUB_TC_OP_Y_ADD_TC_OP_g1600_1705/CI
SUB_TC_OP_Y_ADD_TC_OP_g1600_1705/CO      ADDFX1      1  0.6  39  +186  3212 R
SUB_TC_OP_Y_ADD_TC_OP_g1599_2802/CI
SUB_TC_OP_Y_ADD_TC_OP_g1599_2802/CO      ADDFX1      1  0.6  39  +186  3398 R
SUB_TC_OP_Y_ADD_TC_OP_g1598_1617/CI
SUB_TC_OP_Y_ADD_TC_OP_g1598_1617/CO      ADDFX1      1  0.6  39  +186  3584 R
SUB_TC_OP_Y_ADD_TC_OP_g1597_3680/CI
SUB_TC_OP_Y_ADD_TC_OP_g1597_3680/CO      ADDFX1      1  0.6  39  +186  3770 R
SUB_TC_OP_Y_ADD_TC_OP_g1596_6783/CI
SUB_TC_OP_Y_ADD_TC_OP_g1596_6783/CO      ADDFX1      1  0.4  35  +184  3954 R
SUB_TC_OP_Y_ADD_TC_OP_g1595_5526/B
SUB_TC_OP_Y_ADD_TC_OP_g1595_5526/Y      XNOR2X1      1  0.2  22  +123  4077 R
g1094_2398/B1
g1094_2398/Y      AO22XL       1  0.3  28  +128  4205 R
Acc_A_data_out_reg[15]/D
Acc_A_data_out_reg[15]/CK      DFHQX1      100 +90  4295 R
=====
(clock clk)      capture          10000 R
                                  uncertainty      -10  9990 R
=====

Cost Group : 'clk' (path_group 'clk')
Timing slack : 5695ps
Start-point : fsm/state_reg[0]/CK
End-point   : Acc_A_data_out_reg[15]/D

Ln 1, Col 1 | 5,419 characters
```

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

Synthesis effort: High

```
booth_timing.rep
File Edit View

=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:25:43 pm
Module: BOOTH
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Pin Type Fanout Load Slew Delay Arrival
----- (ff) (ps) (ps) (ps)

(clock clk) launch
fsm_state_reg[1]/CK
fsm_state_reg[1]/Q SDFFQX1 7 2.3 70 +267 267 R
g1206_4319/B
g1206_4319/Y NOR2X1 2 0.6 58 +73 340 R
g1205_6260/A
g1205_6260/Y AND2X1 22 5.8 135 +197 536 R
SUB_TC_OP_Y_ADD_TC_OP_g511/A INVX1 16 3.2 112 +134 670 F
SUB_TC_OP_Y_ADD_TC_OP_g511/Y
SUB_TC_OP_Y_ADD_TC_OP_g496_7482/A MXI2XL 1 0.6 96 +131 802 R
SUB_TC_OP_Y_ADD_TC_OP_g496_7482/Y
SUB_TC_OP_Y_ADD_TC_OP_g493_6131/CI ADDFX1 1 0.6 39 +215 1017 R
SUB_TC_OP_Y_ADD_TC_OP_g493_6131/CO
SUB_TC_OP_Y_ADD_TC_OP_g492_7098/CI ADDFX1 1 0.6 39 +186 1203 R
SUB_TC_OP_Y_ADD_TC_OP_g492_7098/CO
SUB_TC_OP_Y_ADD_TC_OP_g491_8246/CI ADDFX1 1 0.6 39 +186 1389 R
SUB_TC_OP_Y_ADD_TC_OP_g491_8246/CO
SUB_TC_OP_Y_ADD_TC_OP_g490_5122/CI ADDFX1 1 0.6 39 +186 1575 R
SUB_TC_OP_Y_ADD_TC_OP_g490_5122/CO
SUB_TC_OP_Y_ADD_TC_OP_g489_1705/CI ADDFX1 1 0.6 39 +186 1575 R
SUB_TC_OP_Y_ADD_TC_OP_g489_1705/CO
SUB_TC_OP_Y_ADD_TC_OP_g488_2802/CI ADDFX1 1 0.6 39 +186 1761 R
SUB_TC_OP_Y_ADD_TC_OP_g488_2802/CO
SUB_TC_OP_Y_ADD_TC_OP_g487_1617/CI ADDFX1 1 0.6 39 +186 1947 R
SUB_TC_OP_Y_ADD_TC_OP_g487_1617/CO
SUB_TC_OP_Y_ADD_TC_OP_g486_3680/CI ADDFX1 1 0.6 39 +186 2133 R
SUB_TC_OP_Y_ADD_TC_OP_g486_3680/CO
SUB_TC_OP_Y_ADD_TC_OP_g485_6783/CI ADDFX1 1 0.6 39 +186 2319 R
SUB_TC_OP_Y_ADD_TC_OP_g485_6783/CO
SUB_TC_OP_Y_ADD_TC_OP_g484_5526/CI ADDFX1 1 0.6 39 +186 2505 R
SUB_TC_OP_Y_ADD_TC_OP_g484_5526/CO
SUB_TC_OP_Y_ADD_TC_OP_g484_8428/CI ADDFX1 1 0.6 39 +186 2691 R
SUB_TC_OP_Y_ADD_TC_OP_g483_8428/CO
SUB_TC_OP_Y_ADD_TC_OP_g482_4319/CI ADDFX1 1 0.6 39 +186 2877 R
SUB_TC_OP_Y_ADD_TC_OP_g482_4319/CO
SUB_TC_OP_Y_ADD_TC_OP_g481_6260/CI ADDFX1 1 0.6 39 +186 3063 R
SUB_TC_OP_Y_ADD_TC_OP_g481_6260/CO
SUB_TC_OP_Y_ADD_TC_OP_g480_5107/CI ADDFX1 1 0.6 39 +186 3249 R
SUB_TC_OP_Y_ADD_TC_OP_g480_5107/CO
SUB_TC_OP_Y_ADD_TC_OP_g479_2398/CI ADDFX1 1 0.4 37 +257 3435 R
SUB_TC_OP_Y_ADD_TC_OP_g479_2398/S ADDFX1 1 0.4 108 +102 3692 R
g2815_8246/A1
g2815_8246/Y AOI22X1 1 0.4 +0 +0 3794 F
g2740_2398/B0
g2740_2398/Y OAI2BB1X1 1 0.3 40 +80 3874 R
Acc_A_data_out_reg[14]/D << DFFHQX1 100 +96 3970 R
Acc_A_data_out_reg[14]/CK setup
----- capture 10000 R
uncertainty -10 9990 R

Cost Group : 'clk' (path_group 'clk')
Timing slack : 6020ps
Start-point : fsm_state_reg[1]/CK
End-point : Acc_A_data_out_reg[14]/D

Ln 1, Col 1 | 5,342 characters
```

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

d) multiplier_area_created file

Synthesis effort: Medium

```
booth_area.rep      +  -  X  ☰
File Edit View
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:21:21 pm
Module: BOOTH
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----
BOOTH 218 677.160 0.000 677.160 <none> (D)
  Mlcd_M PIPO 16 120.384 0.000 120.384 <none> (D)
  Mlpr_Q shiftreg_1 54 154.584 0.000 154.584 <none> (D)
  count counter 24 56.772 0.000 56.772 <none> (D)
  d_ff dff 2 6.840 0.000 6.840 <none> (D)
  fsm controller 33 60.876 0.000 60.876 <none> (D)
(D) = wireload is default in technology library
Ln 1, Col 1 | 1,130 characters | 100% | Unix (LF) | UTF-8
```

Synthesis effort: High

```
booth_area.rep      +  -  X  ☰
File Edit View
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:25:43 pm
Module: BOOTH
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----
BOOTH 211 667.242 0.000 667.242 <none> (D)
(D) = wireload is default in technology library
Ln 1, Col 1 | 723 characters | 100% | Unix (LF) | UTF-8
```

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

e) multiplier_power_created file

Synthesis effort: Medium

```
booth_power.rep
File Edit View
Instance: /BOOTH
Power Unit: W
PDB Frames: /stim#0/frame#0
-----
Category Leakage Internal Switching Total Row%
-----
memory 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
register 6.79047e-09 1.60959e-05 8.38848e-07 1.69416e-05 56.35%
latch 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
logic 7.16841e-09 9.07615e-06 3.11648e-06 1.21998e-05 40.58%
bbox 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
clock 0.00000e+00 0.00000e+00 9.23400e-07 9.23400e-07 3.07%
pad 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
pm 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
-----
Subtotal 1.39589e-08 2.51721e-05 4.87873e-06 3.00647e-05 100.00%
Percentage 0.05% 83.73% 16.23% 100.00% 100.00%
-----
Ln 1, Col 1 | 1,199 characters | 100% | Unix (LF) | UTF-8
```

Synthesis effort: High

```
booth_area.rep booth_power.rep
File Edit View
Instance: /BOOTH
Power Unit: W
PDB Frames: /stim#0/frame#0
-----
Category Leakage Internal Switching Total Row%
-----
memory 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
register 6.79055e-09 1.63542e-05 8.64179e-07 1.72251e-05 58.48%
latch 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
logic 6.88496e-09 8.53146e-06 2.76721e-06 1.13056e-05 38.38%
bbox 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
clock 0.00000e+00 0.00000e+00 9.23400e-07 9.23400e-07 3.14%
pad 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
pm 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00%
-----
Subtotal 1.36755e-08 2.48856e-05 4.55479e-06 2.94541e-05 100.00%
Percentage 0.05% 84.49% 15.46% 100.00% 100.00%
-----
Ln 14, Col 47 | 1,199 characters | 100% | Unix (LF) | UTF-8
```

LAB4 2: Logic synthesis of a Booth Multiplier using Datapath and Control path logic

Inference:

Synthesis effort: Medium

A total of 218 leaf instance count is present in the gate level netlist with total area of 677.160, total power of 3.00647e-05W.

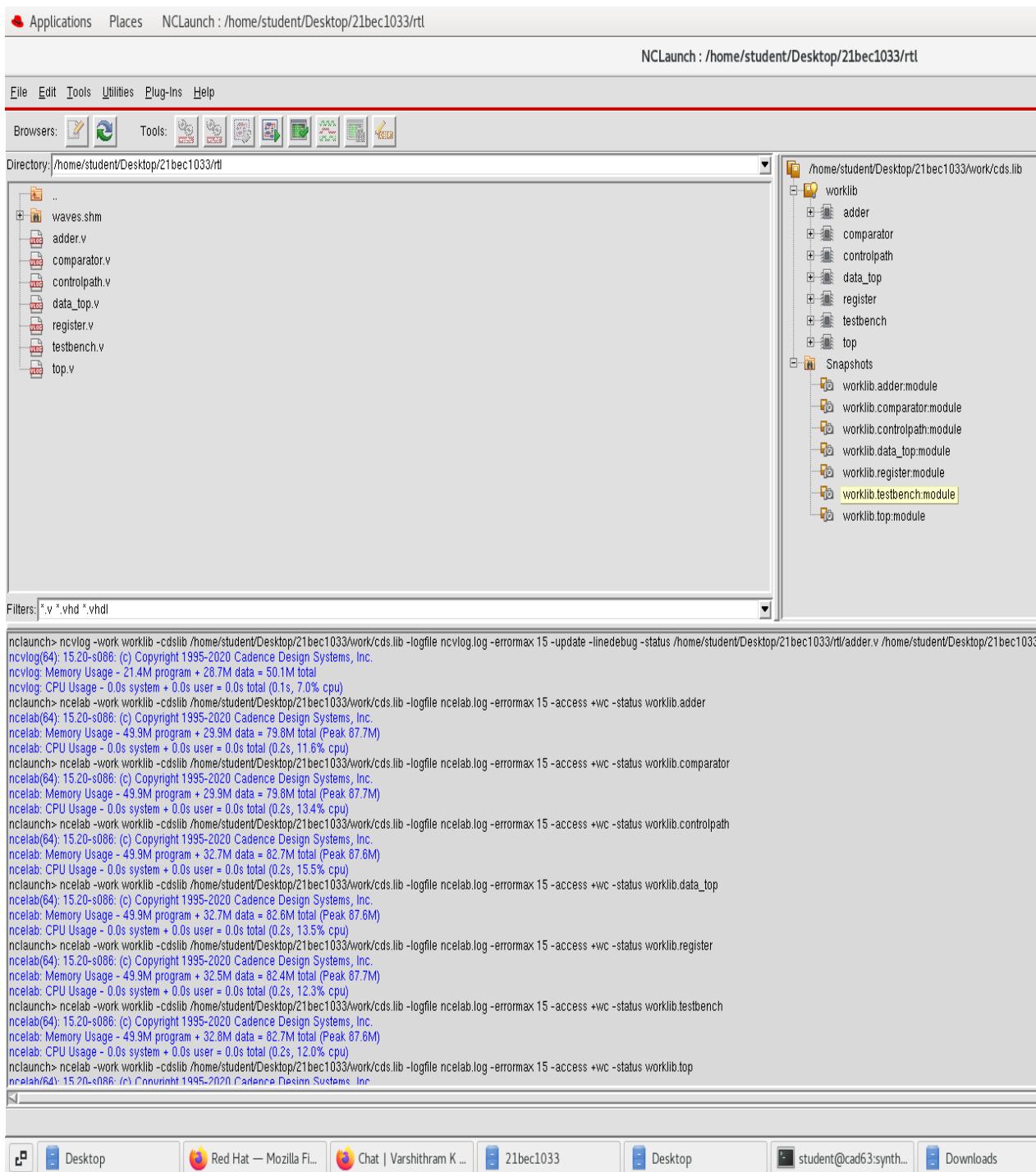
Synthesis effort: High

A total of 211 leaf instance count is present in the gate level netlist with total area of 667.242 , total power of 2.94541e-05W.

Result: Hence a Booth multiplier using Datapath and Control path logic is synthesized and the gate level netlist with timing, area and power report has been generated.

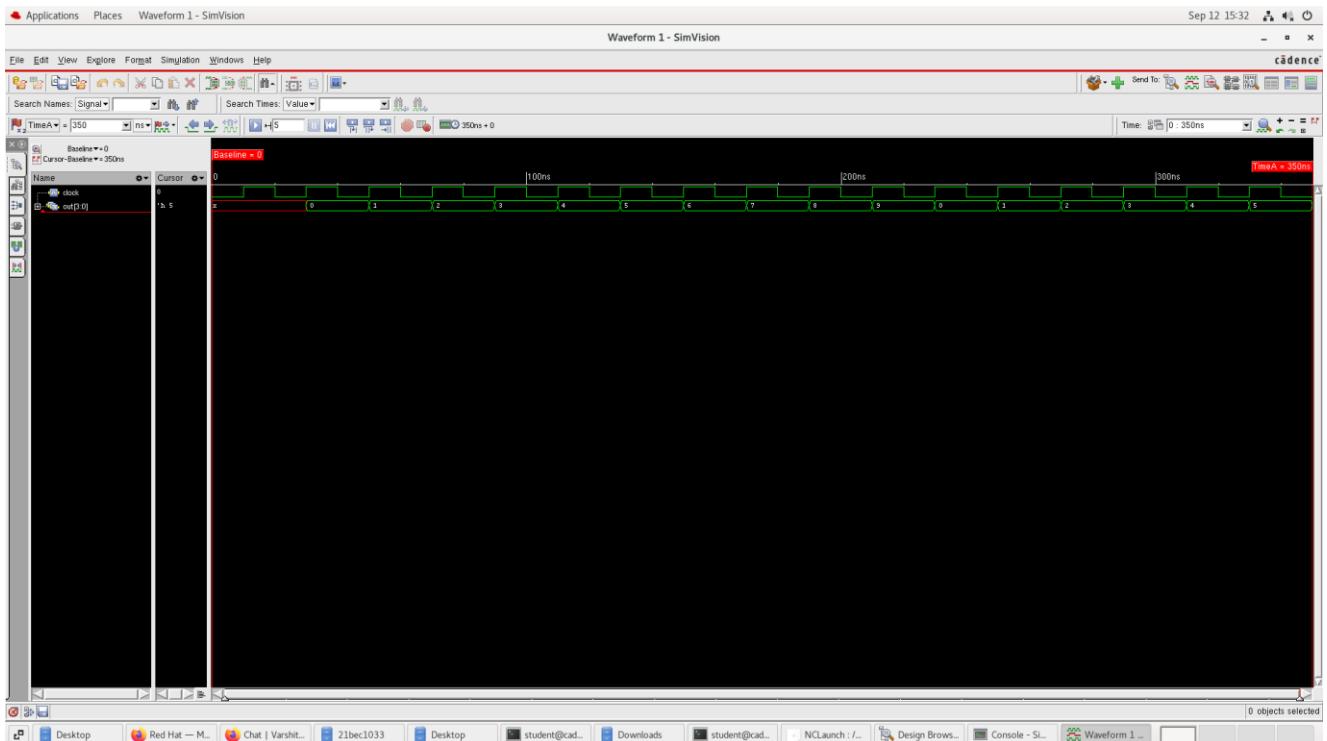
Lab4_4: Design and simulation of a Binary Counter using Datapath and Control path logic

NCLaunch Compilation and Elaboration:



Lab4_4: Design and simulation of a Binary Counter using Datapath and Control path logic

Observations/Simulation Waveforms:



Inference: In the above output waveform, the “out” signal increases by 1 as and when a posedge of the clock occurs. If the out has reached the maximum value then it become zero when it encounters a posedge of the clock.

Result: Hence, a Binary_counter is designed and the simulation and verification of behavior of Binary_counter using data path and Control path logic is performed for the given stimuli in the test bench.

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

Aim: To synthesize a Binary_counter using Datapath and Control path logic and to get the gate level netlist with timing, area, and power reports.

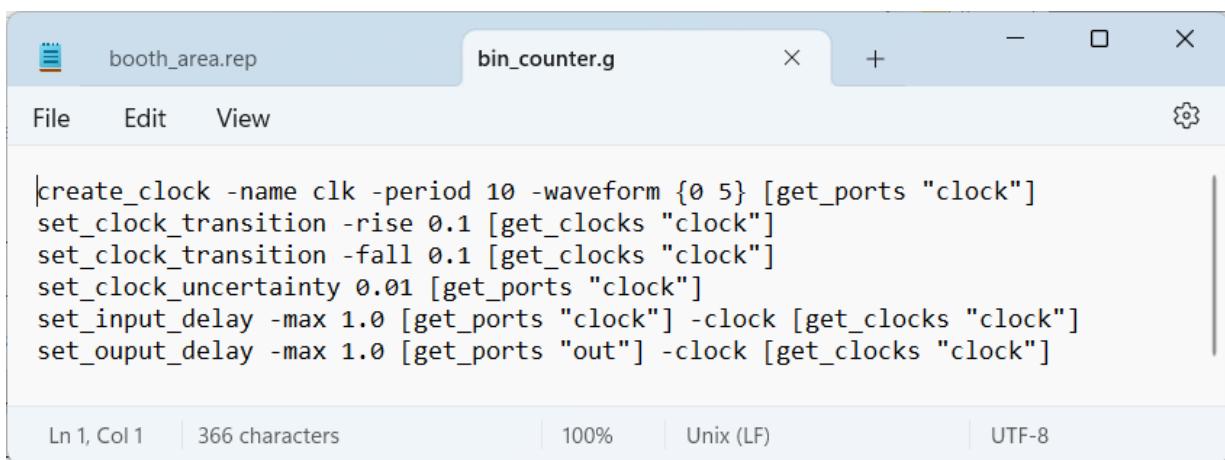
EDA Tools Used: Cadence Genus

Description:

Datapath is the hardware that performs all the required operations. Control is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc. Simple datapath components include memory (stores the current instruction), PC or program counter (stores the address of current instruction), and ALU (executes current instruction). Control path is the Finite state machine designed by using Moore or Mealy models.

Procedure:

1. Copy the fast.lib and slow.lib files into the folder where (.v) files are located.
2. Create a Synopsys design constraint file (.g file) by entering the design constraints required for the synthesis.

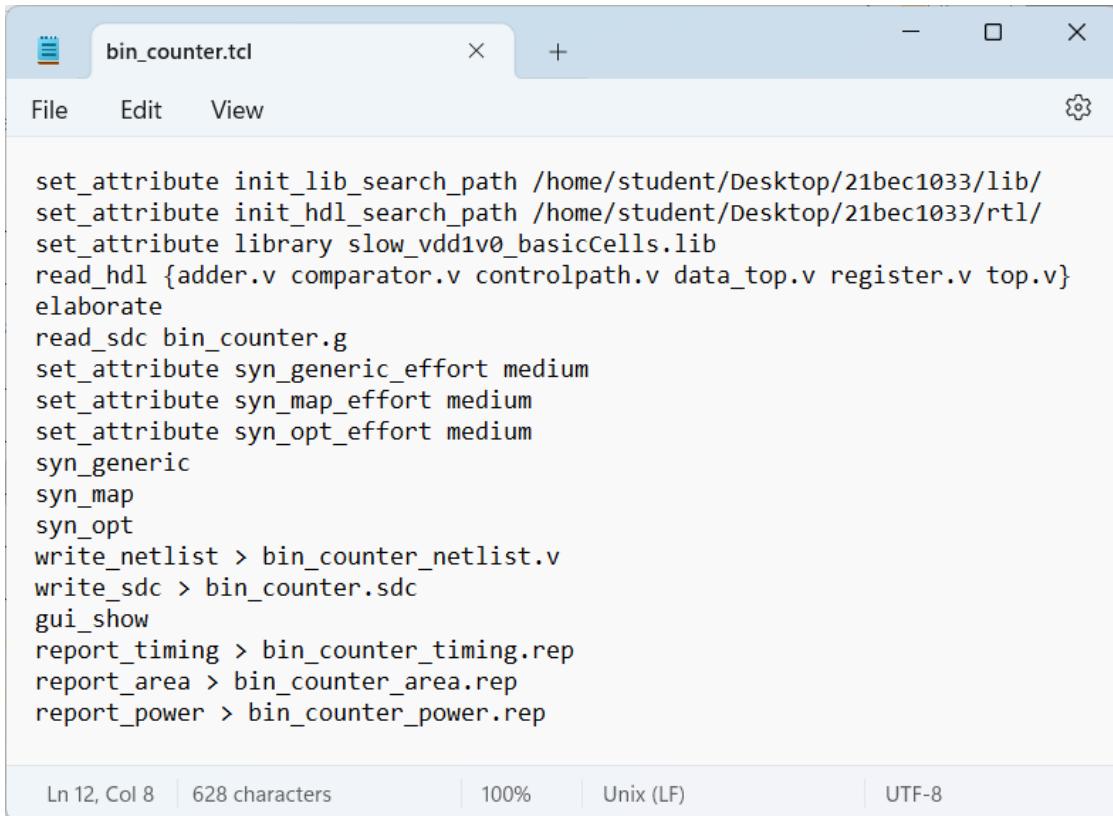


```
booth_area.rep bin_counter.g
File Edit View
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clock"]
set_clock_transition -rise 0.1 [get_clocks "clock"]
set_clock_transition -fall 0.1 [get_clocks "clock"]
set_clock_uncertainty 0.01 [get_ports "clock"]
set_input_delay -max 1.0 [get_ports "clock"] -clock [get_clocks "clock"]
set_output_delay -max 1.0 [get_ports "out"] -clock [get_clocks "clock"]

Ln 1, Col 1 | 366 characters | 100% | Unix (LF) | UTF-8
```

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

3. Create a (.tcl) file containing all the commands for performing the logic synthesis. Synthesis effort can be medium or high.



The screenshot shows a text editor window with the title bar 'bin_counter.tcl'. The menu bar includes 'File', 'Edit', 'View', and a settings gear icon. The main text area contains the following Tcl script:

```
set_attribute init_lib_search_path /home/student/Desktop/21bec1033/lib/
set_attribute init_hdl_search_path /home/student/Desktop/21bec1033/rtl/
set_attribute library slow_vdd1v0_basicCells.lib
read_hdl {adder.v comparator.v controlpath.v data_top.v register.v top.v}
elaborate
read_sdc bin_counter.sdc
set_attribute syn_generic_effort medium
set_attribute syn_map_effort medium
set_attribute syn_opt_effort medium
syn_generic
syn_map
syn_opt
write_netlist > bin_counter_netlist.v
write_sdc > bin_counter.sdc
gui_show
report_timing > bin_counter_timing.rep
report_area > bin_counter_area.rep
report_power > bin_counter_power.rep
```

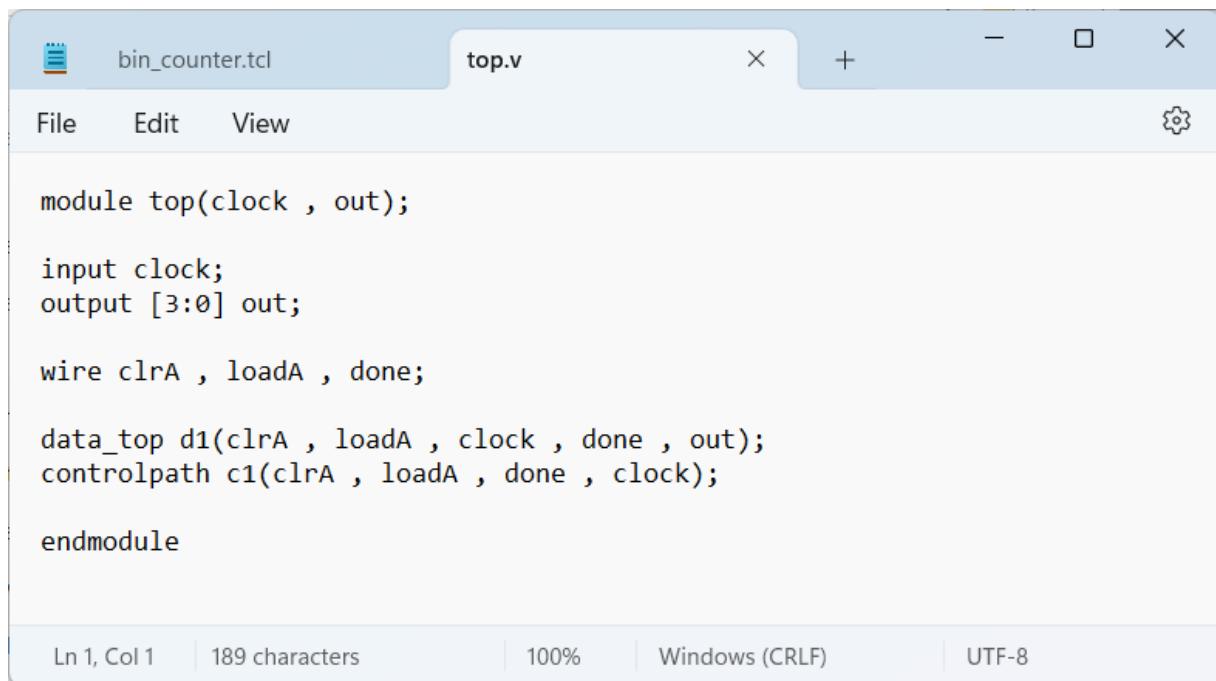
At the bottom of the window, status indicators show 'Ln 12, Col 8', '628 characters', '100%', 'Unix (LF)', and 'UTF-8'.

4. Invoke the C shell and launch the Genus tool by entering the below commands.
‘csh’
‘source /home/install/cshrc’
‘genus -legacy -ui -f bin_counter.tcl’
5. Execute the commands in the (.tcl file) by entering **source bin_counter.tcl** command in the command line window.
6. Check for the area, timing and power reports generated in the respective multiplier folder. Also check the gate level netlist generated in the Genus synthesis solution window.

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

Verilog Programs:

//Verilog Program of top-level module containing data path and control path



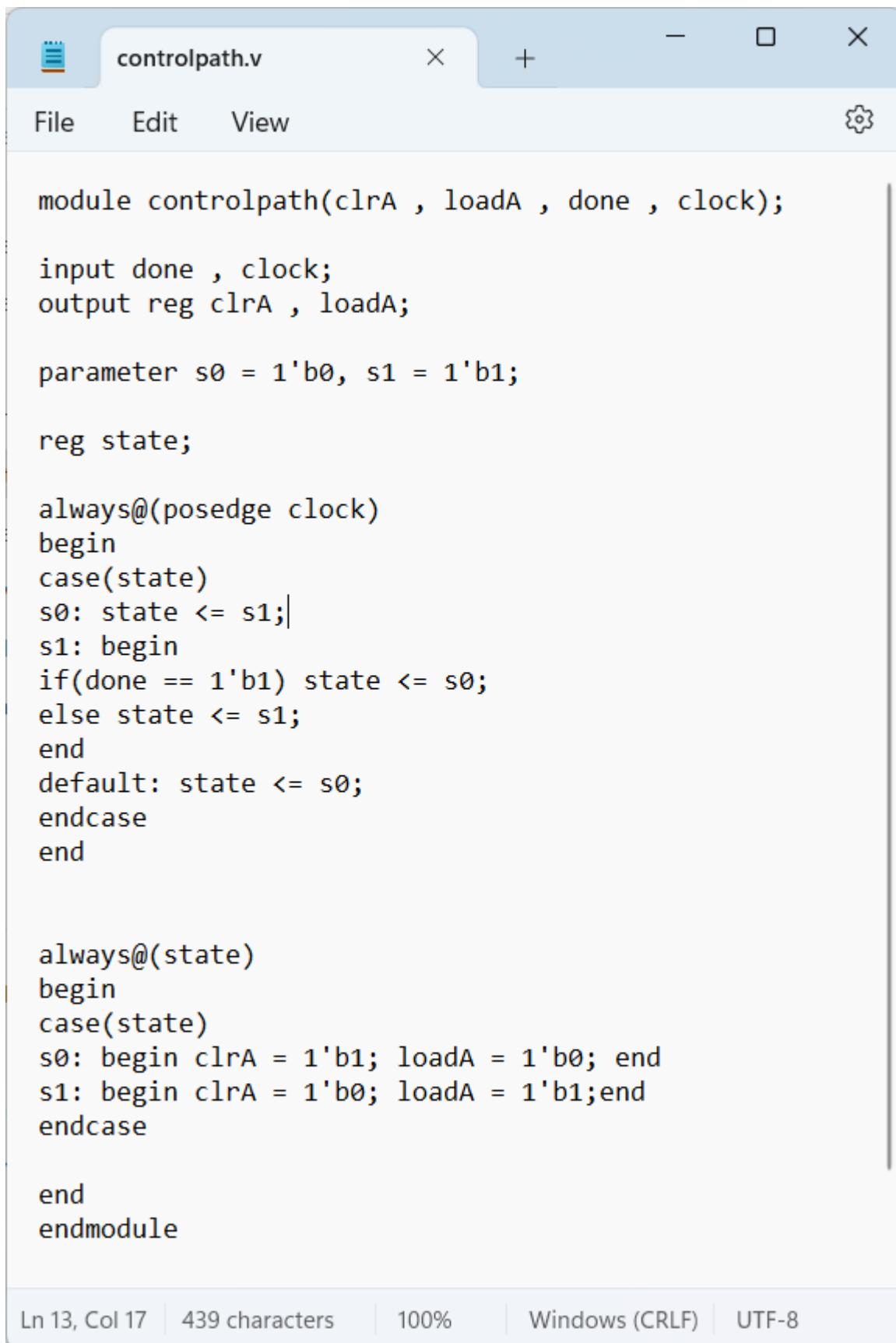
The screenshot shows a code editor window with the following details:

- File tab: bin_counter.tcl
- Active tab: top.v
- Toolbar: File, Edit, View, settings gear icon.
- Code content:

```
module top(clock , out);  
  input clock;  
  output [3:0] out;  
  
  wire clrA , loadA , done;  
  
  data_top d1(clrA , loadA , clock , done , out);  
  controlpath c1(clrA , loadA , done , clock);  
  
endmodule
```
- Status bar: Ln 1, Col 1 | 189 characters | 100% | Windows (CRLF) | UTF-8

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

//Verilog Program of control path module



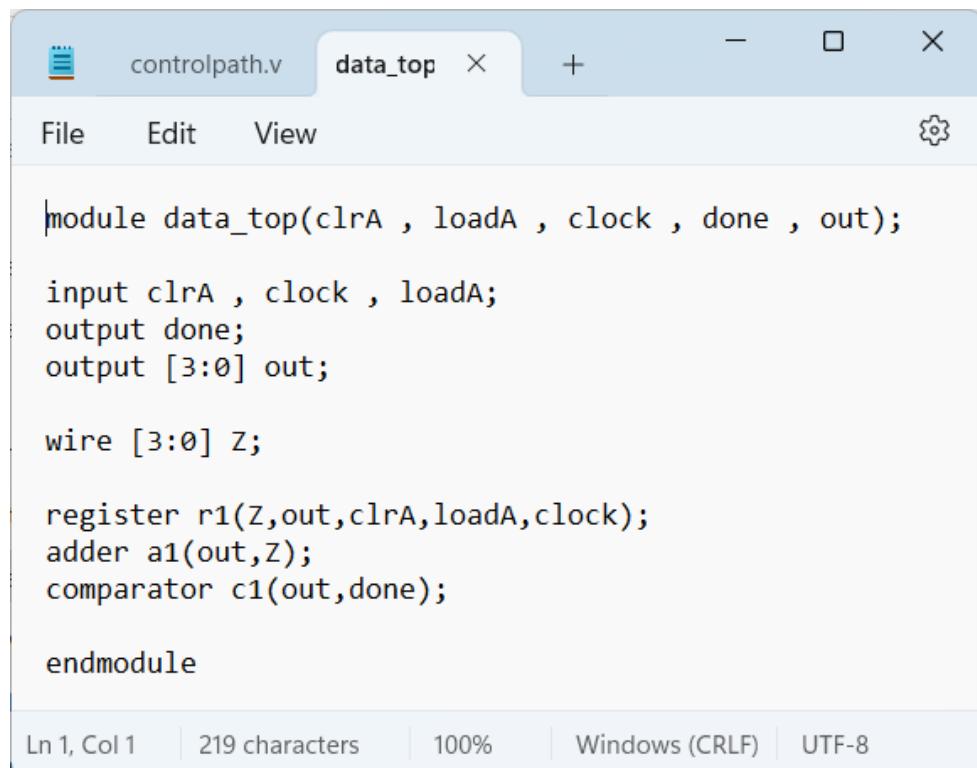
The screenshot shows a code editor window titled "controlpath.v". The window has standard operating system window controls at the top right. The menu bar includes "File", "Edit", "View", and a settings gear icon. The code itself is Verilog, defining a module "controlpath" with inputs "done" and "clock", and outputs "clrA" and "loadA". It uses parameters s0 and s1, and a register "state". The module contains two always blocks: one for state transitions based on the clock edge, and another for generating control signals based on the current state.

```
module controlpath(clrA , loadA , done , clock);  
  input done , clock;  
  output reg clrA , loadA;  
  
  parameter s0 = 1'b0, s1 = 1'b1;  
  
  reg state;  
  
  always@(posedge clock)  
  begin  
    case(state)  
      s0: state <= s1; |  
      s1: begin  
        if(done == 1'b1) state <= s0;  
        else state <= s1;  
      end  
      default: state <= s0;  
    endcase  
  end  
  
  
  always@(state)  
  begin  
    case(state)  
      s0: begin clrA = 1'b1; loadA = 1'b0; end  
      s1: begin clrA = 1'b0; loadA = 1'b1; end  
    endcase  
  end  
endmodule
```

Ln 13, Col 17 | 439 characters | 100% | Windows (CRLF) | UTF-8

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

//Verilog Program of datapath



The screenshot shows a Verilog code editor window with the title bar "controlpath.v" and the active tab "data_top". The menu bar includes File, Edit, View, and a settings gear icon. The code area contains the following Verilog module definition:

```
module data_top(clrA , loadA , clock , done , out);

    input clrA , clock , loadA;
    output done;
    output [3:0] out;

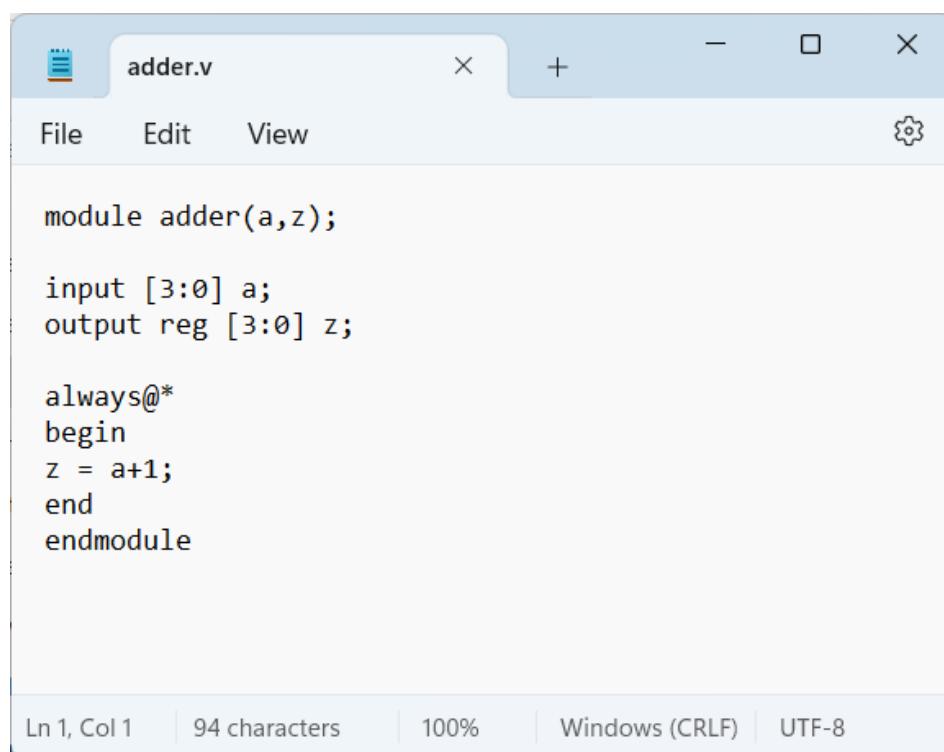
    wire [3:0] Z;

    register r1(Z,out,clrA,loadA,clock);
    adder a1(out,Z);
    comparator c1(out,done);

endmodule
```

The status bar at the bottom shows "Ln 1, Col 1 | 219 characters | 100% | Windows (CRLF) | UTF-8".

//Verilog Program of adder



The screenshot shows a Verilog code editor window with the title bar "adder.v". The menu bar includes File, Edit, View, and a settings gear icon. The code area contains the following Verilog module definition:

```
module adder(a,z);

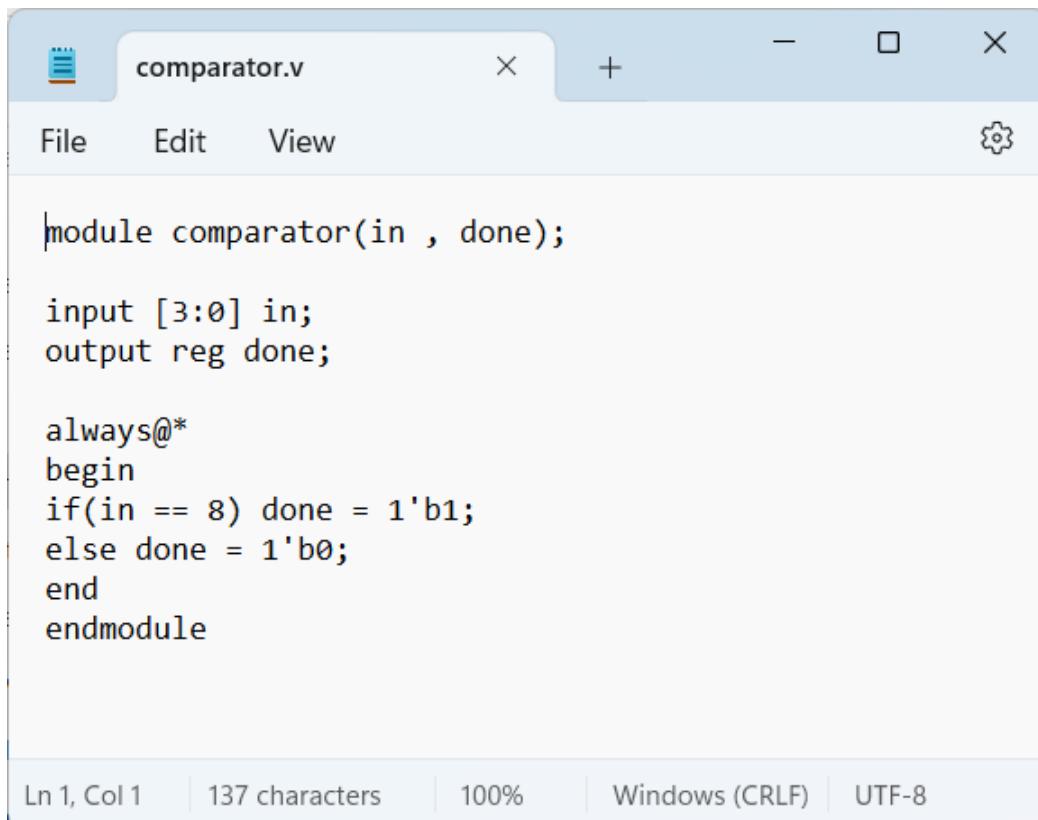
    input [3:0] a;
    output reg [3:0] z;

    always@*
    begin
        z = a+1;
    end
endmodule
```

The status bar at the bottom shows "Ln 1, Col 1 | 94 characters | 100% | Windows (CRLF) | UTF-8".

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

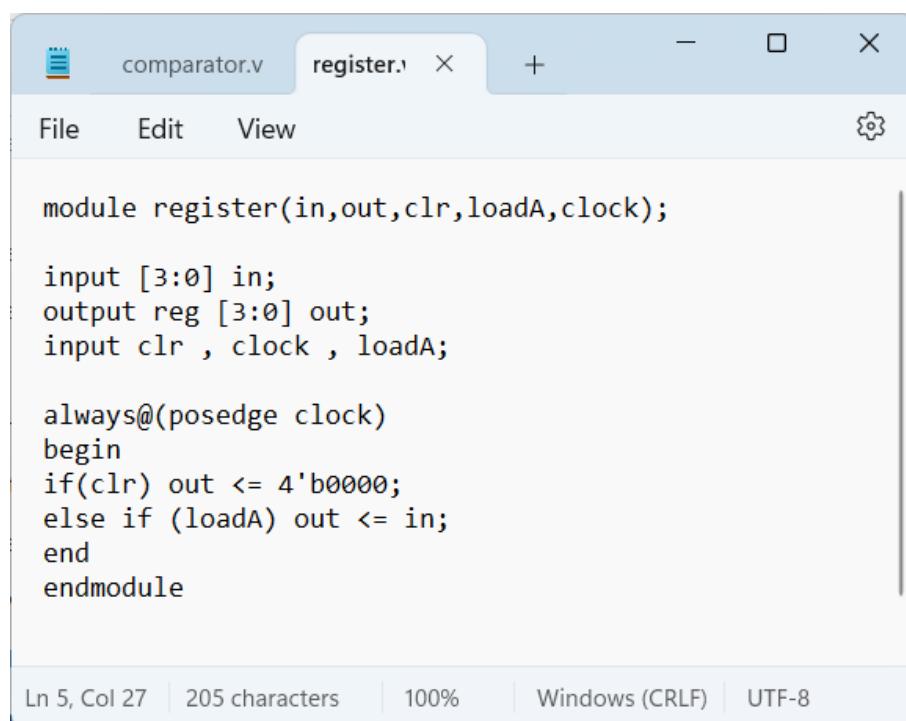
//Verilog Program of Comparator



A screenshot of a text editor window titled "comparator.v". The code defines a module "comparator" with an input "in" of type [3:0] and an output "done" of type reg. An "always@*" block checks if "in" is equal to 8, setting "done" to 1'b1; otherwise, it sets "done" to 1'b0. The editor interface includes tabs for "File", "Edit", "View", and a gear icon. Status bar at the bottom shows "Ln 1, Col 1", "137 characters", "100%", "Windows (CRLF)", and "UTF-8".

```
module comparator(in , done);  
  input [3:0] in;  
  output reg done;  
  
  always@*  
  begin  
    if(in == 8) done = 1'b1;  
    else done = 1'b0;  
  end  
endmodule
```

//Verilog Program of register



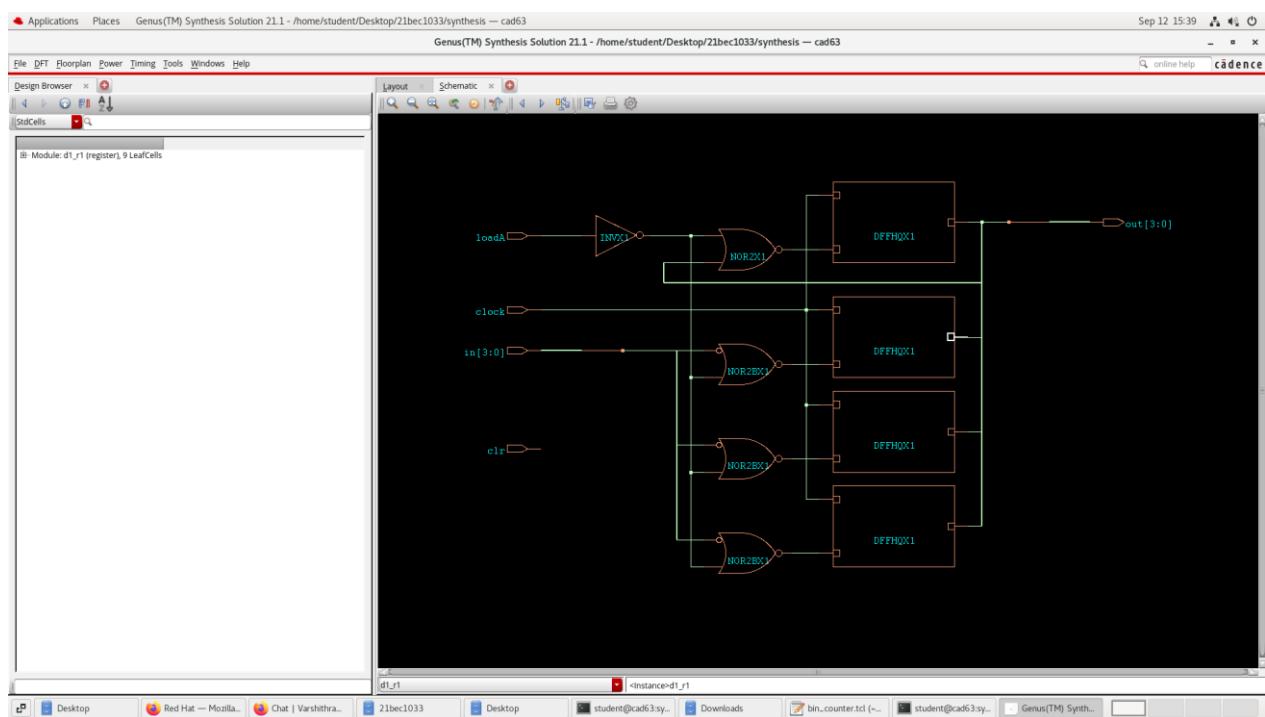
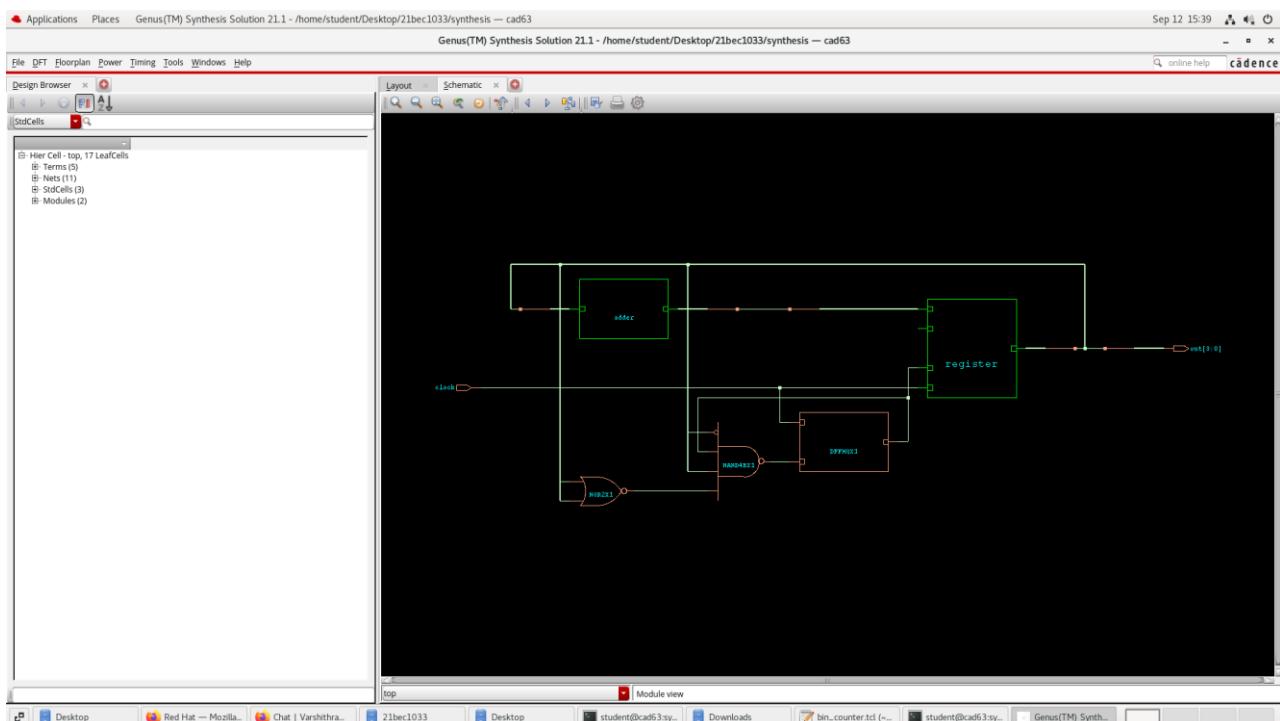
A screenshot of a text editor window titled "comparator.v" with a tab for "register.v". The code defines a module "register" with inputs "in" and "out" of type [3:0], and control signals "clr", "clock", and "loadA". An "always@(posedge clock)" block updates the register "out" based on "in", "clr", and "loadA". The editor interface includes tabs for "File", "Edit", "View", and a gear icon. Status bar at the bottom shows "Ln 5, Col 27", "205 characters", "100%", "Windows (CRLF)", and "UTF-8".

```
module register(in,out,clr,loadA,clock);  
  input [3:0] in;  
  output reg [3:0] out;  
  input clr , clock , loadA;  
  
  always@(posedge clock)  
  begin  
    if(clr) out <= 4'b0000;  
    else if (loadA) out <= in;  
  end  
endmodule
```

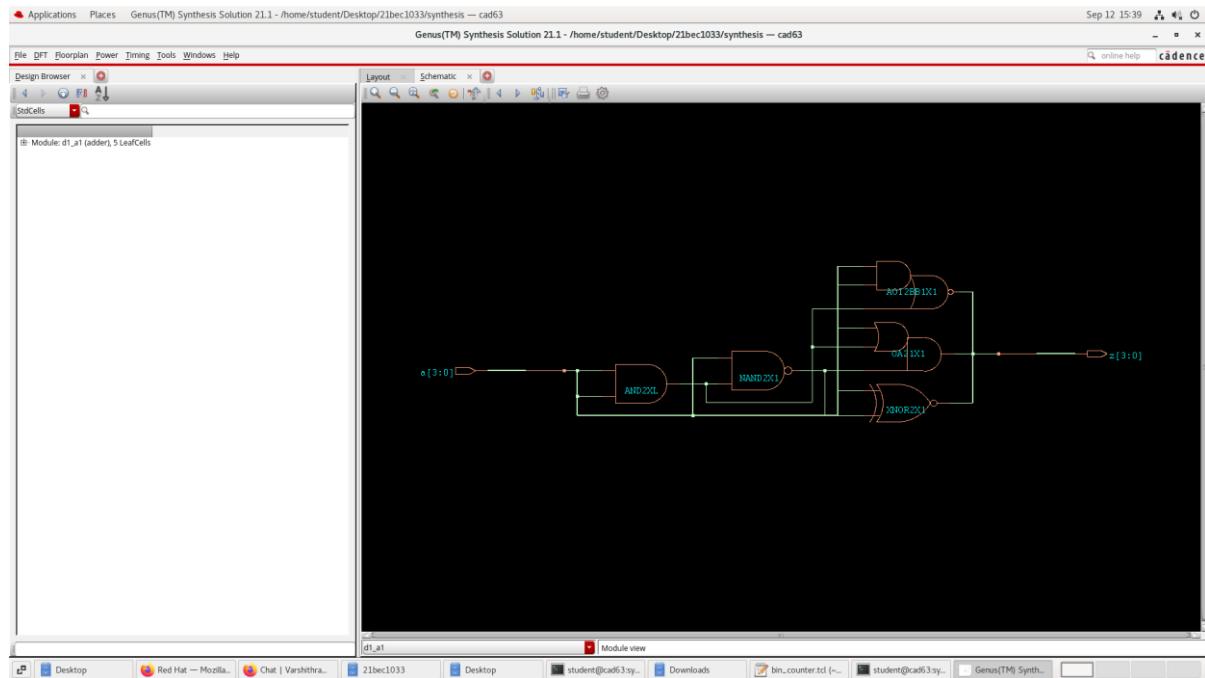
LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

Gate level Netlist:

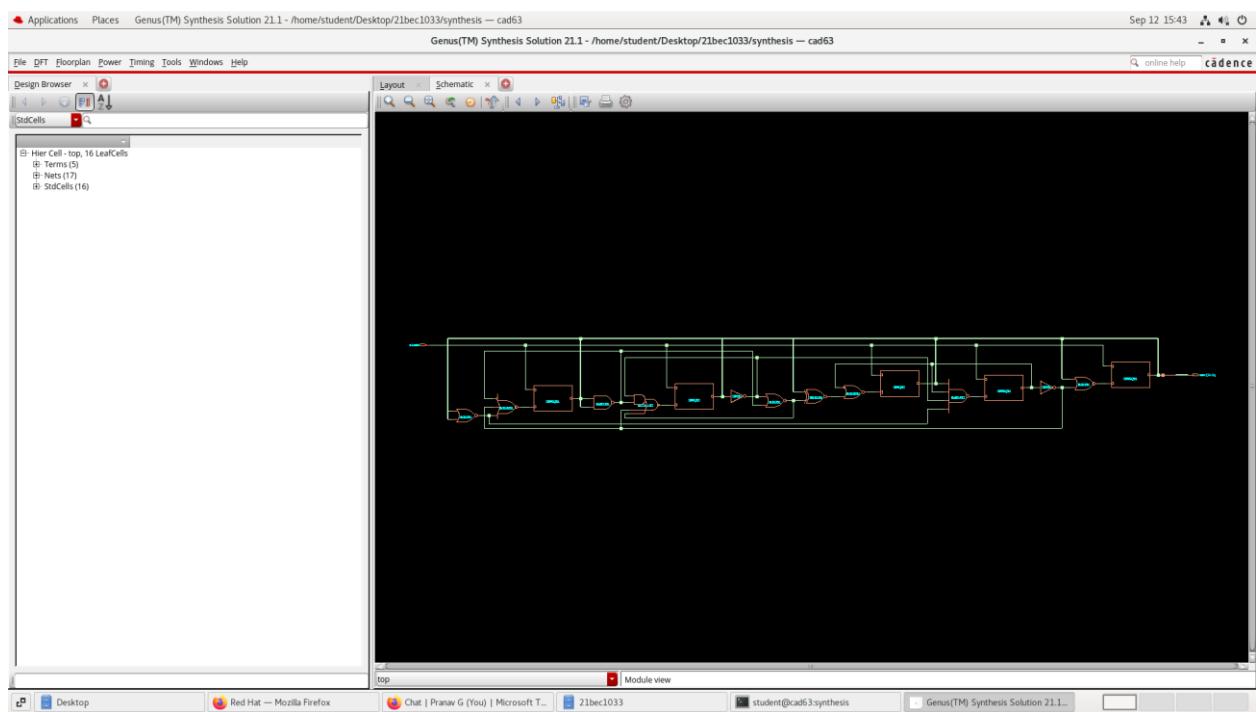
Synthesis effort: Medium



LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic



Synthesis effort : High



LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

Observations:

- Binary_counter_netlist_created file

```
// Generated by Cadence Genus(TM) Synthesis Solution 21.14-s082_1
// Generated on: Sep 12 2024 15:43:07 IST (Sep 12 2024 10:13:07 UTC)

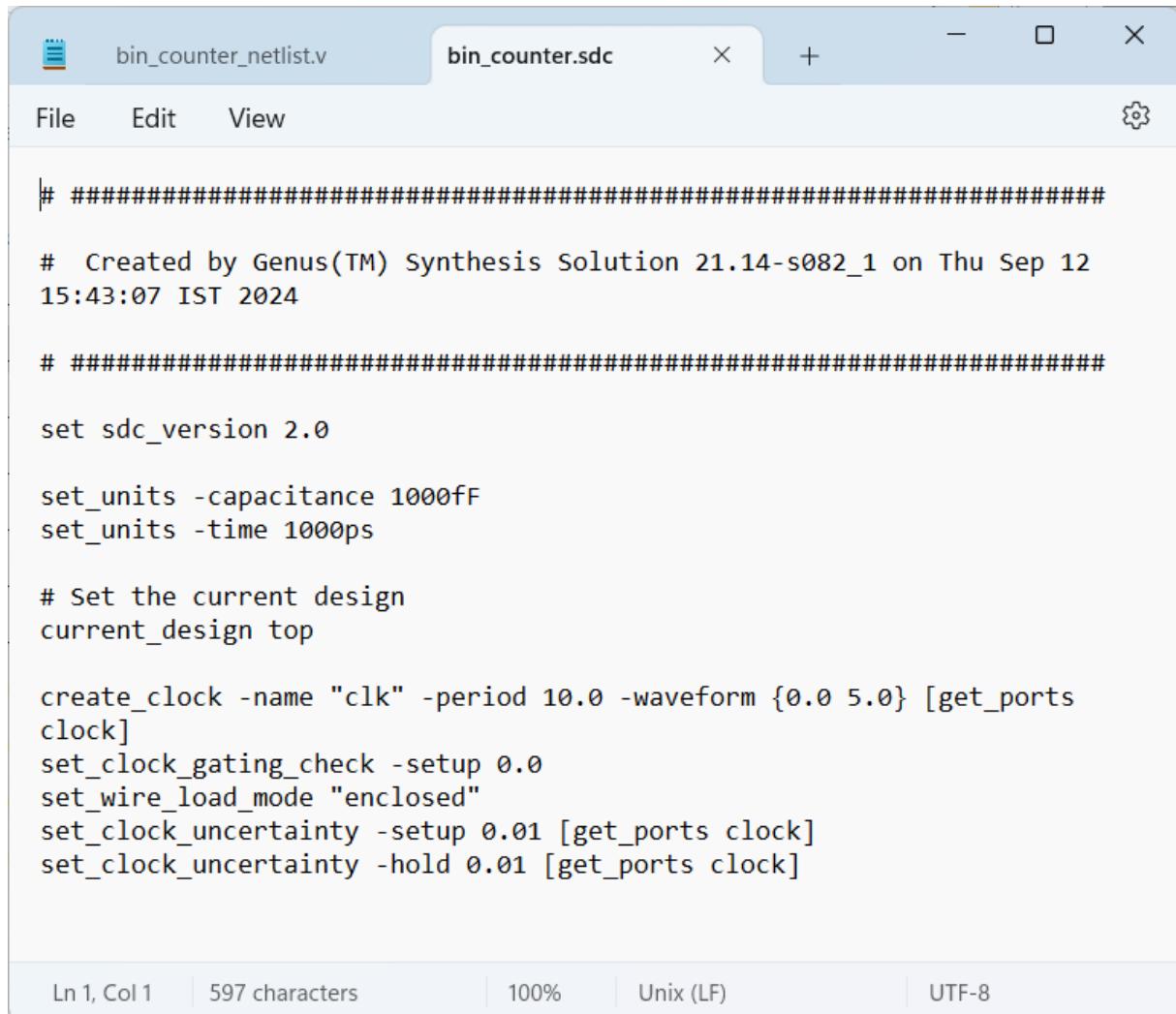
// Verification Directory fv/top

module top(clock, out);
    input clock;
    output [3:0] out;
    wire clock;
    wire [3:0] out;
    wire loadA, n_0, n_1, n_2, n_3, n_4, n_5, n_6;
    wire n_7, n_8, n_9, n_10;
    DFFQXL \d1_r1_out_reg[3] (.CK (clock), .D (n_10), .Q (out[3]));
    NOR2BX1 g163_2398(.AN (loadA), .B (n_9), .Y (n_10));
    DFFQXL \d1_r1_out_reg[2] (.CK (clock), .D (n_8), .Q (out[2]));
    XNOR2X1 g165_5107(.A (out[3]), .B (n_7), .Y (n_9));
    AOI211XL g166_6260(.A0 (n_2), .A1 (n_5), .B0 (n_4), .C0 (n_7), .Y
        (n_8));
    DFFHQX1 \d1_r1_out_reg[1] (.CK (clock), .D (n_6), .Q (out[1]));
    DFFHQX1 c1_state_reg(.CK (clock), .D (n_3), .Q (loadA));
    NOR3BX1 g170_4319(.AN (n_5), .B (n_4), .C (n_1), .Y (n_6));
    NAND4XL g169_8428(.A (out[3]), .B (loadA), .C (n_2), .D (n_1), .Y
        (n_3));
    DFFHQX1 \d1_r1_out_reg[0] (.CK (clock), .D (n_0), .Q (out[0]));
    NOR2X1 g172_5526(.A (n_2), .B (n_5), .Y (n_7));
    NOR2X1 g173_6783(.A (out[0]), .B (n_4), .Y (n_0));
    NAND2X1 g175_3680(.A (out[1]), .B (out[0]), .Y (n_5));
    NOR2X1 g174_1617(.A (out[1]), .B (out[0]), .Y (n_1));
    INVX1 g176(.A (out[2]), .Y (n_2));
    INVX1 g177(.A (loadA), .Y (n_4));
endmodule
```

Ln 1, Col 1 | 1,308 characters | 100% | Unix (LF) | UTF-8

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

b) Binary_counter_constraint_created file



The screenshot shows a text editor window titled "bin_counter.sdc". The file content is a System Design Constraints (SDC) script used for logic synthesis. It includes comments about the creation by Genus(TM) Synthesis Solution, specifies units for capacitance and time, sets the current design to "top", and creates a clock named "clk" with a period of 10.0 units. The script also configures clock gating, wire load mode, and clock uncertainty.

```
# #####
# Created by Genus(TM) Synthesis Solution 21.14-s082_1 on Thu Sep 12
# 15:43:07 IST 2024
#
set sdc_version 2.0

set_units -capacitance 1000fF
set_units -time 1000ps

# Set the current design
current_design top

create_clock -name "clk" -period 10.0 -waveform {0.0 5.0} [get_ports
clock]
set_clock_gating_check -setup 0.0
set_wire_load_mode "enclosed"
set_clock_uncertainty -setup 0.01 [get_ports clock]
set_clock_uncertainty -hold 0.01 [get_ports clock]
```

Ln 1, Col 1 | 597 characters | 100% | Unix (LF) | UTF-8

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

c) Binary_counter_timing_created file

Synthesis effort: Medium

```
bin_counter_timing.rep * + ⚙
File Edit View
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:39:19 pm
Module: top
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Pin Type Fanout Load Slew Delay Arrival
(fF) (ps) (ps) (ps)
-----
(clock clk) launch 0 R
d1_r1
  out_reg[0]/CK 0 +0 0 R
  out_reg[0]/Q DFFHQX1 5 1.2 36 +181 181 R
d1_r1/out[0]
d1_a1/a[0]
  g86_8428/A +0 181
  g86_8428/Y AND2XL 3 1.0 49 +130 311 R
  g84_6260/B +0 311
  g84_6260/Y NAND2X1 2 0.6 70 +74 385 F
  g82_2398/B +0 385
  g82_2398/Y XNOR2X1 1 0.2 22 +176 561 R
d1_a1/z[3]
d1_r1/in[3]
  g31/AN +0 561
  g31/Y NOR2BX1 1 0.3 43 +86 647 R
  out_reg[3]/D <<< DFFHQX1 +0 647
  out_reg[3]/CK setup 0 +126 773 R
-----
(clock clk) capture 10000 R
uncertainty -10 9990 R
-----
Cost Group : 'clk' (path_group 'clk')
Timing slack : 9217ps
Start-point : d1_r1/out_reg[0]/CK
End-point   : d1_r1/out_reg[3]/D
Ln 1, Col 1 | 1,999 characters | 100% | Unix (LF) | UTF-8
```

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

Synthesis effort: High

```
bin_counter_timing.rep
```

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	launch					0 R
d1_r1_out_reg[0]/CK				0	+0	0 R
d1_r1_out_reg[0]/Q	DFFHQX1	4	1.2	36	+181	181 R
g175_3680/B					+0	181
g175_3680/Y	NAND2X1	3	0.7	74	+70	251 F
g172_5526/B					+0	251
g172_5526/Y	NOR2X1	2	0.6	58	+75	326 R
g165_5107/B					+0	326
g165_5107/Y	XNOR2X1	1	0.3	26	+139	465 F
g163_2398/B					+0	465
g163_2398/Y	NOR2BX1	1	0.2	36	+39	504 R
d1_r1_out_reg[3]/D <<<	DFFQXL				+0	504
d1_r1_out_reg[3]/CK	setup			0	+125	629 R
(clock clk)	capture					10000 R
	uncertainty				-10	9990 R
Cost Group : 'clk' (path_group 'clk')						
Timing slack : 9361ps						
Start-point : d1_r1_out_reg[0]/CK						
End-point : d1_r1_out_reg[3]/D						

Ln 1, Col 1 | 2,022 characters | 100% | Unix (LF) | UTF-8

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

d) Binary_counter_area_created file

Synthesis effort: Medium

```
bin_counter_timing.rep bin_counter_area.rep × + ⌂
File Edit View ⌂
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:39:19 pm
Module: top
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----
top 17 45.486 0.000 45.486 <none> (D)
d1_a1 adder 5 8.892 0.000 8.892 <none> (D)
d1_r1 register 9 27.702 0.000 27.702 <none> (D)
(D) = wireload is default in technology library
=====

Ln 1, Col 1 | 881 characters | 100% | Unix (LF) | UTF-8
```

Synthesis effort: High

```
bin_counter_area.rep × + ⌂
File Edit View ⌂
=====
Generated by: Genus(TM) Synthesis Solution 21.14-s082_1
Generated on: Sep 12 2024 03:43:07 pm
Module: top
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----
top 16 42.408 0.000 42.408 <none> (D)
(D) = wireload is default in technology library
=====

Ln 1, Col 1 | 721 characters | 100% | Unix (LF) | UTF-8
```

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

e) Binary_counter_power_created file

Synthesis effort: Medium

bin_counter_power.rep

File Edit View

```
Instance: /top
Power Unit: W
PDB Frames: /stim#0/frame#0
```

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	6.01724e-10	1.07533e-06	2.65163e-08	1.10245e-06	79.10%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	4.18816e-10	1.73367e-07	3.65876e-08	2.10374e-07	15.09%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	8.10000e-08	8.10000e-08	5.81%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.02054e-09	1.24870e-06	1.44104e-07	1.39383e-06	100.00%
Percentage	0.07%	89.59%	10.34%	100.00%	100.00%

Ln 1, Col 1 | 1,197 characters | 100% | Unix (LF) | UTF-8

Synthesis effort: High

bin_counter_area.rep

bin_counter_power.rep

File Edit View

```
Instance: /top
Power Unit: W
PDB Frames: /stim#0/frame#0
```

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	5.27962e-10	9.85531e-07	2.27229e-08	1.00878e-06	81.83%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	3.40689e-10	1.05315e-07	3.72653e-08	1.42921e-07	11.59%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	8.10000e-08	8.10000e-08	6.57%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	8.68651e-10	1.09085e-06	1.40988e-07	1.23270e-06	99.99%
Percentage	0.07%	88.49%	11.44%	100.00%	100.00%

Ln 1, Col 1 | 1,197 characters | 100% | Unix (LF) | UTF-8

LAB4 4: Logic synthesis of a Binary Counter using Datapath and Control path logic

Inference:

Synthesis effort: Medium

A total of 17 leaf instance count is present in the gate level netlist with total area of 45.486, total power of 1.39383e-06 W.

Synthesis effort: High

A total of 16 leaf instance count is present in the gate level netlist with total area of 42.408, total power of 1.23270e-06 W.

Result: Hence a Binary_counter using Datapath and Control path logic is synthesized and the gate level netlist with timing, area and power report has been generated.