BÀI 1: THIẾT KẾ VÀ VIẾT TESTBECH

1. MỤC ĐÍCH

-Giúp Sinh viên thiết kế các mạch và hệ thống đơn giản dùng ngôn ngữ Verilog.

-Giúp Sinh viên thực hành viết testbench kiểm tra mạch thiết kế.

-Giúp Sinh viên thực hành công cụ mô phỏng.

2. THỰC HÀNH

2.1. Thực hiện mạch đếm và quan sát mô phỏng dùng công cụ VCS:

File : Counter.v

```
module counter ( out, clk, reset ) ;
    input clk, reset;
    output [3:0] out;
    reg [3:0] out;
    wire [3:0] next;
    // This statement implements reset and increment
    assign next = reset ? 4'b0 : (out + 4'b1);
    // This implements the flip-flops
    always @ ( posedge clk ) begin
    out <= #1 next;
    end
endmodule // counter
```

File : Counter_tb.v

```
// This stuff just sets up the proper time scale and format for the simulation, for now
do not modify.
`timescale 1ns/10ps
module counter_testbench ( ) ;
    wire [3:0] out;
    reg clk;
    reg reset;
    counter dut (.out (out[3:0]),  .reset (reset),  .clk (clk));
```

```
always #10 clk = ~clk;
initial begin
    $monitor ("time=%5d ns, clk=%b, reset=%b, out=%b", $time, clk, reset,
out[3:0]);
    clk = 1'b0;
    reset = 1'b0;
    reset = 1'b1;
    @(posedge clk);#1;
    reset = 1'b0;
    @(posedge clk);#1;
    @(posedge clk);#1;
    @(posedge clk);#1;
    @(posedge clk);#1;
    @(posedge clk);#1;
    @(posedge clk);#1;
    @(posedge clk);#1;
    if (out != 4'b0110) begin
        $display("ERROR 1: Out is not equal to 4'b0110");
        $finish;
    end
     $display("All tests completed sucessfully\n\n");
     $finish;
end
 initial begin
     $vcdplusfile ("counter.vpd");
     $vcdpluson;
end
endmodule // counter_testbench
```

Chú ý:

✓ Để chạy chương trình mô phỏng VCS vào linux, mở TERMINAL và gõ lệnh sau để biên dịch:

vcs –R –debug_pp <tên_testbench> <tên_code>

Ví dụ: vcs –R –debug_pp Counter_tb.v Counter.v

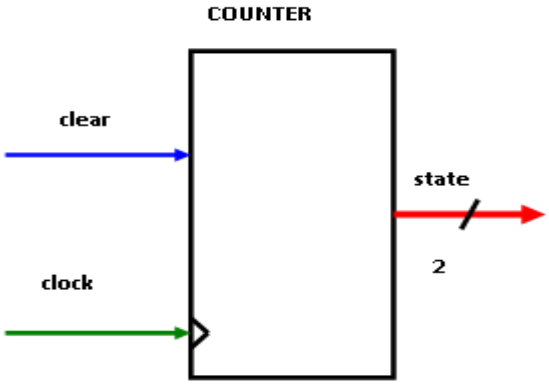
✓ Quan sát dạng sóng sử dụng công cụ DVE:

dve

2.2. Thiết kế và viết testbech

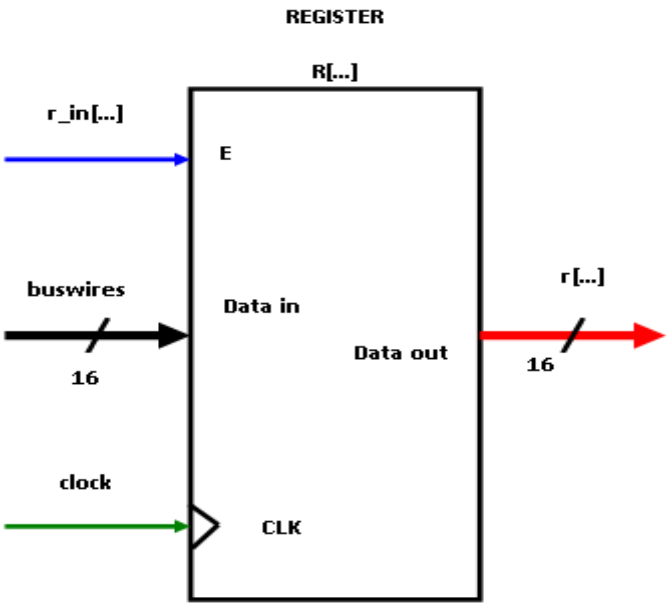
$$Y = \overline{A}.\overline{B}.\overline{C} + A.\overline{B}.\overline{C} + A.\overline{B}.C.$$

2.3. Thiết kế và viết testbech cho bộ counter



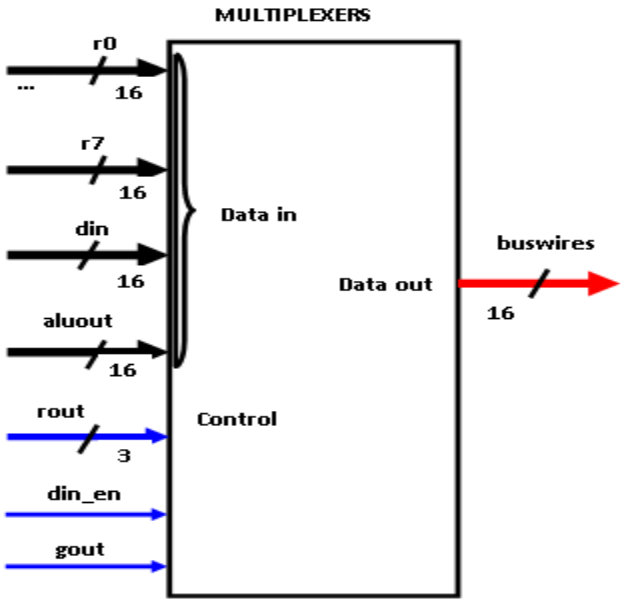
STT	Tên pin	Số bit	Chiều	Chức năng
1	clear	1	input	
2	clock	1	input	
3	state	2	output	data_out

2.4. Thiết kế và viết testbech cho bộ thanh ghi 16 bit



STT	Tên pin	Số bit	Chiều	Chức năng
1	rin[...]	1	input	enable
2	clock	1	input	
3	buswires	16	input	data_in
4	r[...]	16	output	data_out

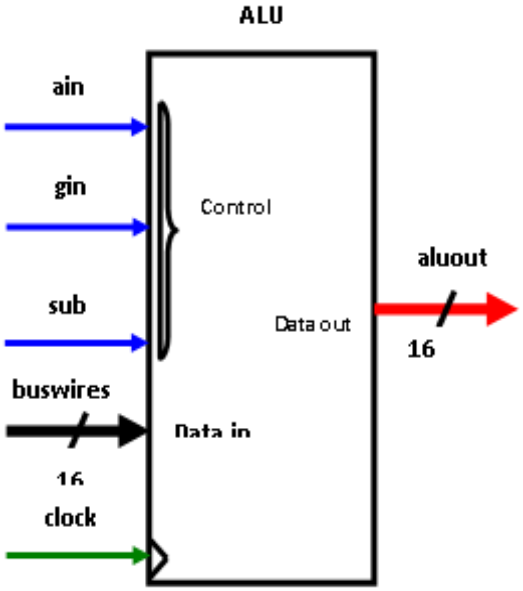
2.5. Thiết kế và viết testbench cho bộ multiplexer

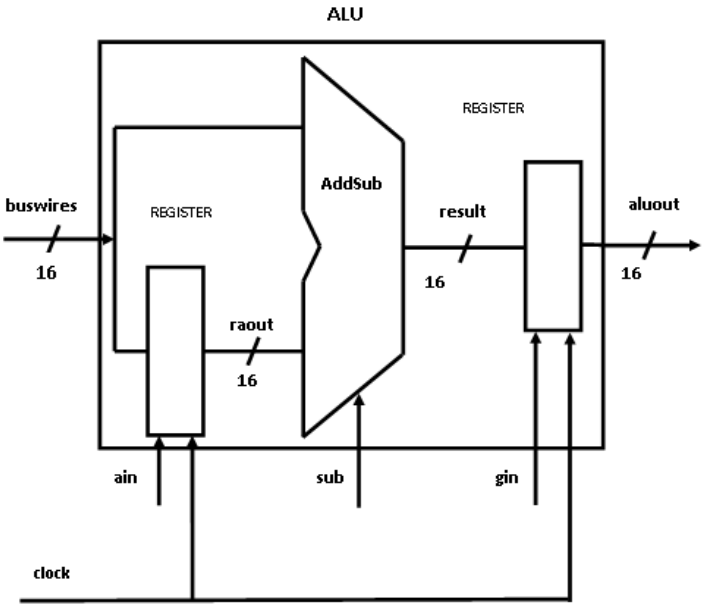


STT	Tên pin	Số bit	Chiều	Chức năng
1	din	16	input	data_in
2	r0	16	input	data_in
3	r1	16	input	data_in
4	r2	16	input	data_in
5	r3	16	input	data_in
6	r4	16	input	data_in
7	r5	16	input	data_in
8	r6	16	input	data_in
9	r7	16	input	data_in
11	aluout	16	input	data_in
12	buswires	16	output	data_out
13	rout	3	i	selection
14	din_en	1	i	selection

15	gout	1	i	selection

2.6. Thiết kế và viết testbench cho bộ ALU bao gồm ALU và khối điều khiển





STT	Tên Pin	Số bit	Chiều	Chức năng
1	buswires	16	input	data_in
2	clock	1	input	
3	ain	1	input	control_in
4	gin	1	input	control_in
5	sub	1	input	control_in

6	aluout	16	output	data_out

Mô tả hoạt động của khối ALU

Khối ALU thực hiện 2 phép toán cộng và trừ 2 số (khác nhau) A và B 16 bit trong hai chu kì:

Chu kì 1: A được đưa vào thanh ghi ain.

Chu kì 2: B được đưa trực tiếp đến ALU và thực hiện phép cộng hoặc trừ. Kết quả được lưu vào thanh ghi gin và đưa ra lối ra.

BÀI 1: THIẾT KẾ VÀ VIẾT TESTBECH (tiếp theo)

1. MỤC ĐÍCH

-Giúp Sinh viên thiết kế các hệ thống phức tạp dùng ngôn ngữ Verilog.

-Giúp Sinh viên thực hành viết testbench kiểm tra mạch thiết kế.

-Giúp Sinh viên thực hành công cụ mô phỏng thiết kế VCS của Synopsys.

2. THỰC HÀNH

2.1. Thực hiện thiết kế khối CU - Control Unit và viết testbench

STT	Tên Pin	Độ rộng	Chiều	Chức năng
1	Run	1	i	Run
2	resetn	1	i	Reset
3	Ir	9	i	inst
4	state	2	i	Inst cycle
5	ain	1	O	Enable A
6	gin	1	O	Enable G
7	sub	1	O	Select add or sub
8	rin	8	O	Enable r0-r7
9	rout	3	O	-> mux
10	din_en	1	O	-> mux
11	ir_en	1	O	Enable ir
12	clear	1	O	Clear counter
13	done	1	O	complete

Mô tả hoạt động của khối CU:

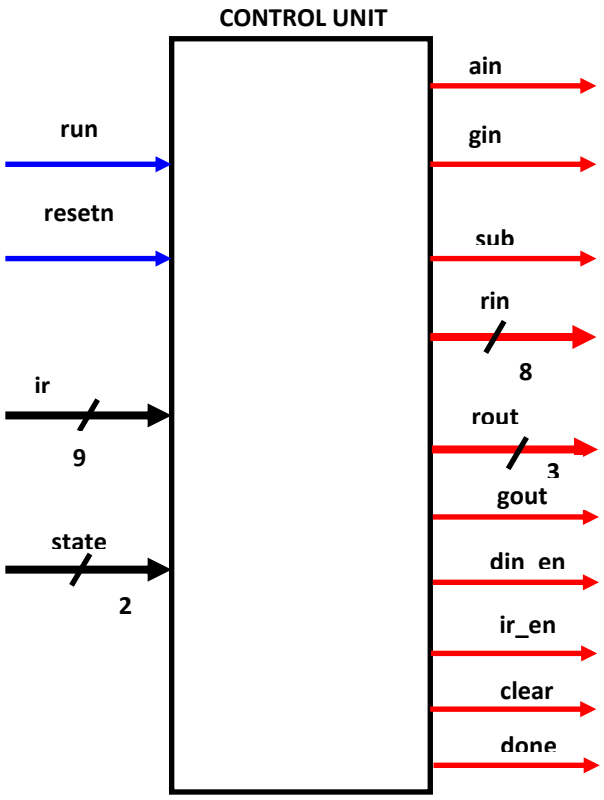
Khối CU điều khiển hoạt động các khối khác trong thiết kế CPU (phần sau). Chu kì lệnh lớn nhất của CPU là 4 chu kì nên bộ đếm chương trình cần 2 bit. Mỗi trạng thái, CU đưa tín hiệu điều khiển đến từng khối (qui ước các tín hiệu điều khiển tác động mức cao)

- state [1:0]: 4 trạng thái
- ir [8:0]: 9-bit.

ir [8:6] : mã lệnh

3'b 000 : dịch chuyển (move) dữ liệu từ ry đến rx (mv)

3'b 001 : dịch chuyển dữ liệu từ din đến rx (mvi)



3'b 010 : cộng 2 thanh ghi rx và ry, kết quả lưu ở thanh ghi rx (add)

3'b 011 : trừ 2 thanh ghi rx và ry, kết quả lưu ở thanh ghi rx (sub)

3'b 100 : dịch chuyển dữ liệu từ rx ra dout (mvo)

ir [5:3] : địa chỉ rx

ir [2:0] : địa chỉ ry

- reset : reset tất cả trở về trạng thái ban đầu

a. State 0 : Thanh ghi ir được kích hoạt để đọc giá trị từ din

→ ir_en = 1

b. State 1 :

- Nếu mã lệnh là mv:

  rout = ry

  rin = rx

  done = 1

- Nếu mã lệnh là mvi:

  -> rin = rx

  -> done

- Nếu mã lệnh là add or sub:

  Chuyển dữ liệu vào thanh ghi A từ rx

  -> rout = rx

  -> ain =1

c. State 2 :

- Nếu mã lệnh là add/sub:

  buswires = ry

  alu thực hiện cộng và giá trị lưu vào thanh ghi G

  -> rout = ry

  -> sub=1/ 0

  -> g_in=1

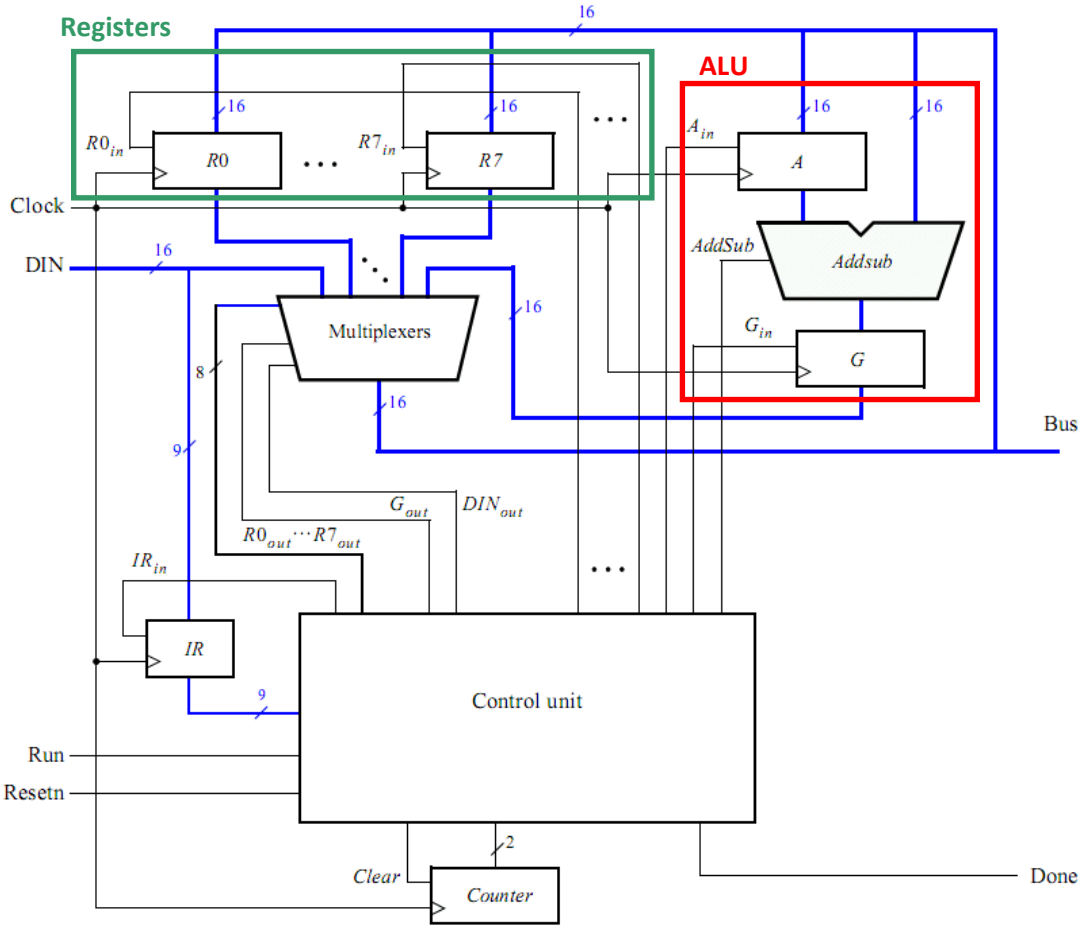
d. State 3 :

  -> rin = rx

  -> gout=1

  -> din=1

2.2. Thực hiện thiết kế CPU theo kiến trúc sau:



CPU bao gồm:

7 thanh ghi 16 bit R0, R1,…, R7: lưu trữ dữ liệu cần thiết

1 bộ mux 9 lối vào, 1 lối ra: điều khiển luồng dữ liệu trong CPU

1 bộ ALU: thực hiện phép toán cộng trừ

1 thanh ghi IR 9 bit giải mã lệnh: giúp CPU hiệu lệnh từ ngoài vào

1 bộ đếm chương trình (2 bit): giúp khối điều khiển quản lý tiến trình đang thực hiện

1 bộ điều khiển (CU – Control Unit): điều tiết toàn bộ hoạt động của CPU

Qui trình hoạt động của CPU

Bảng 1 mô tả mã lệnh của CPU. Thanh ghi IR 9 bit có chuẩn III_XXX_YYY, với:

III: tượng trưng cho mã lệnh

XXX: địa chỉ thanh ghi Rx

YYY: địa chỉ thanh ghi Ry

Lệnh	Chức năng
mv Rx,Ry	Rx ← [Ry]
mvi Rx,#D	Rx ← D
add Rx,Ry	Rx ← [Rx]+[Ry]
sub Rx,Ry	Rx ← [Rx] − [Ry]

Bảng 1: Lệnh thực thi của CPU

➢ Chú ý:

- ✓ Lệnh move chỉ thực hiện trong 1 chu kì
- ✓ Lệnh add/sub thực hiện nhiều hơn 1 chu kì
- ✓ Kết thúc mỗi lệnh CPU xuất ra tín hiệu Done (Bảng 2)

	T1	T2	T3
(mv): I_0	Ry_{out}, Rx_{in},Done		
(mvi): I_1	DIN_{out}, Rx_{in}, Done		
(add): I_2	Rx_{out}, A_{in}	Ry_{out}, G_{in}	G_{out}, Rx_{in}, Done
(sub): I_3	Rx_{out}, A_{in}	Ry_{out},G_{in}, AddSub	G_{out}, Rx_{in}, Done

Bảng 2: Các bước thực thi lệnh

2.3. Thực hiện mô phỏng và quan sát dạng sóng dùng công cụ VCS thiết kế CPU trên

BÀI 3: TỔNG HỢP THIẾT KẾ

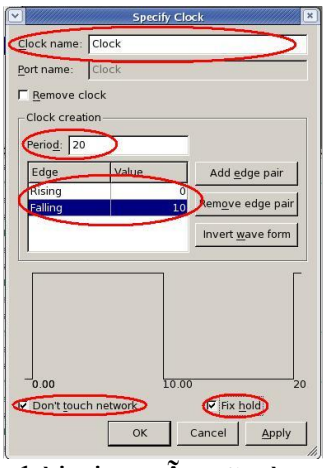
1. MỤC ĐÍCH

- Giúp sinh viên tổng hợp và tối ưu hóa thiết kế.
- Giúp sinh viên sử dụng công cụ tổng hợp và tối ưu mạch
- Giúp sinh viên đọc và hiểu tập tin sau khi tổng hợp mạch

2. HƯỚNG DẪN THỰC HÀNH

Bước 1. Khởi động công cụ Design Compiler
Cách 1: Khởi động giao diện hình ảnh
dc_shell –gui
Cách 2: Khởi động giao diện dùng lệnh
dc_shell
Bước 2: Thiết đặt thư viện
Cách 1: Vào File->Setup
Cách 2: Dùng lệnh
set link_library: Các thư viện cần cho việc tổng hợp thiết kế
set target_library: Thư viện đích để tối ưu tổng hợp thiết kế
set symbol_library: Thư viện biểu tượng cho thiết kế sau khi tổng hợp thiết kế
Cú pháp:
set link_library [list đường_dẫn/tên_thư_viện_1.db đường_dẫn/tên_thư_viện_2.db]
set target_library [list đường_dẫn/tên_thư_viện_1.db
đường_dẫn/tên_thư_viện_2.db]
set symbol_library [list đường_dẫn/tên_thư_viện_1.db
đường_dẫn/tên_thư_viện_2.sb]
Bước 3: Đọc thiết kế
Cách 1:
File ->Read->verilog file

Cách 2:
read_file -format verilog { đường_dẫn/thiết_kế_1.v đường_dẫn/thiết_kế_2.v }
analyze -library WORK -format Verilog { đường_dẫn/thiết_kế_1.v
đường_dẫn/thiết_kế_2.v }
elaborate -architecture verilog -library WORK module_top
check_design
Bước 4: Thiết lập thông số để tổng hợp mạch
Cách 1:
Tạo xung clock: Attributes->Specify clock

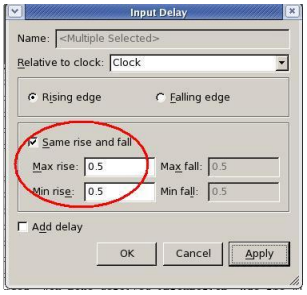


Thiết lập các thông số khác như thời gian trễ ngõ vào, ra, cường độ lái tải, tải, môi trường hoạt động, ….**:** Attributes-> Optimization Contraints

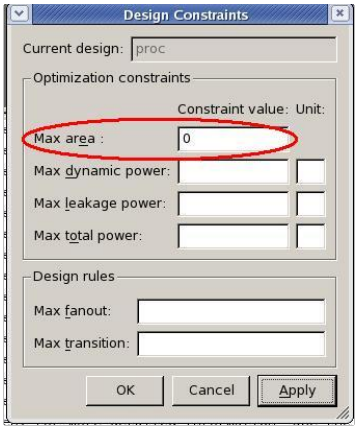
- Thời gian trễ ngõ vào/ra (Input/Output delay)

Select → Ports → Pins →Input ports

Attributes → Operating Environment → Input/Output delay



- Diện tích



Cách 2:

Tạo clock:

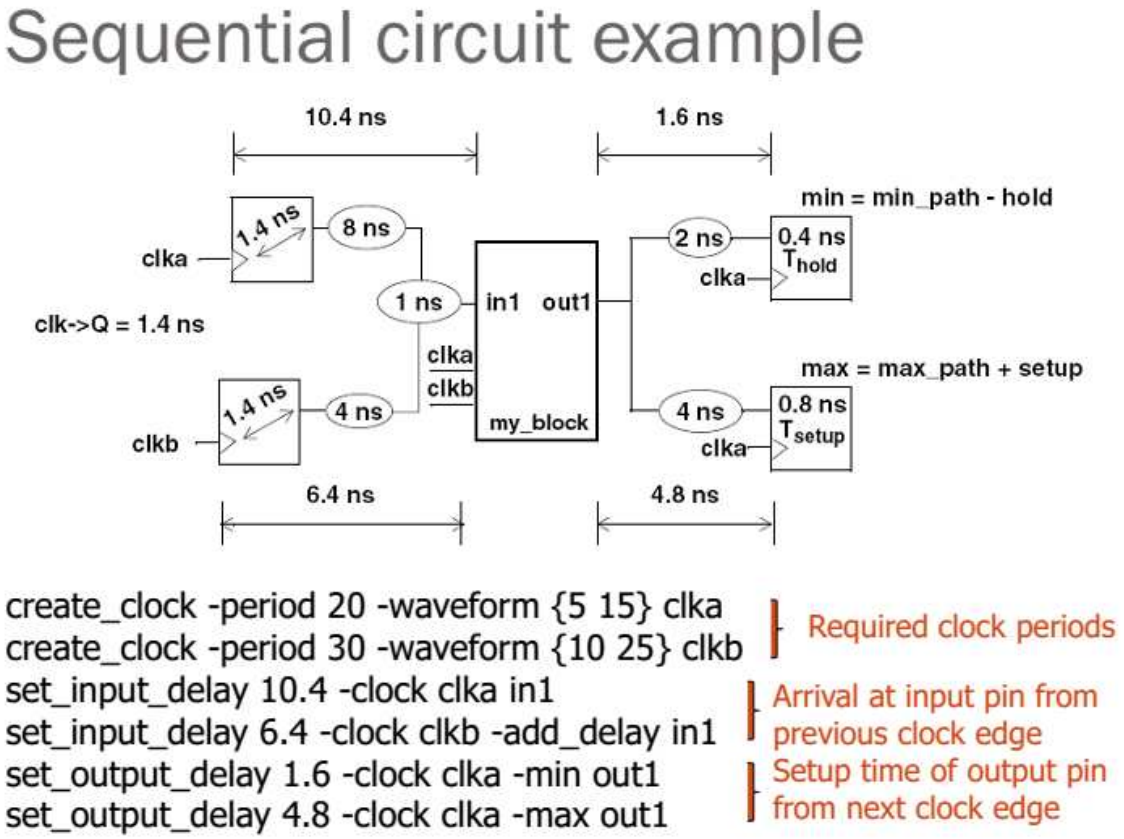
create_clock -period thời_gian -name design_clk ***-waveform*** *{T1 T2}* tên_clock

Ví dụ:

set_clock master -period 10 -waveform {0 5} CLK

Thời gian trễ ngõ vào/ra:

set_input_delay thời_gian [get_ports tên_ngõ_vào] -clock tên_clock
set_output_delay thời_gian [get_ports tên_ngõ_ra] -clock tên_clock



Diện tích thiết kế
set_max_area 0 : chọn cell có diện tích tối ưu nhất
Môi trường hoạt động:
set_operating_conditions -min_library tên_thư_viện \
-min điều_kiện_tương_ứng \
-max_library tên_thư_viện
-max điều_kiện_tương_ứng
Tương tự thiết đặt thông số còn lại

- Clock uncertainly
  set_clock_uncertainty
- Clock latency
  set_clock_latency
- Clock transition
  set_clock_transition
- Set load on all outpus

set_load

- Set maximum trasition limit

set_max_transition

Bước 5: Tổng hợp mạch
Compile

Compiling the design

- Compile (and optimize) the design
  - *compile –map_effort $mapEffort1*
    - *design hierarchy preserved*
    - *map_effort = medium(default) or high*
  - *compile –ungroup_all –map_effort $mapEffort1*
    - *design "flattened" (ungrouped – all levels collapsed)*
  - *compile –incremental_mapping –map_effort $mapEffort2*
    - *work on it some more – incremental improvements*
- High-effort compile
  - *compile_ultra*
  - use on high-performance designs, tight time constraints
  - specify *–no_autoungroup* to preserve hierarchy

Bước 6: Báo cáo kết quả tổng hợp

report_area
report_cell
report_qor
report_resources
report_timing

Bước7: Lưu lại kết quả báo cáo

write_sdc
write

Synthesis Output Files

- Design.v – verilog structural netlist
  - *change_names –rules verilog*
  - *write –format verilog –output file.v*
- Design.sdf – std delay file for timing simulation
  - *write_sdf –version 1.0 file.sdf*
- Design.rep – synthesis report (timing,area,etc)
  - *redirect file.rep {report_timing}*
  - *redirect –append file.rep {report_area –hier }*
- Design.ddc – Synopsys database format (to view in DV)
  - *write –format ddc –hierarchy -o file.ddc*
- Design.sdc – constraints for Encounter place/route
  - *write_sdc file.sdc*
- Design.pow – power estimate
  - *redirect file.pow { report_power }*

Bước 8: Mô phỏng netlist

vcs –R –debug_pp <tên_testbench> <tên_code> –v lib.v

3. THỰC HÀNH

3.1 Thực hiện tổng hợp thiết kế mạch đếm (counter) ở bài 1.

3.2 Thực hiện tổng hợp các thiết ở bài 1 và 2

3.3 Thực hiện tổng hợp thiết kế CPU ở bài 2 và tối ưu thiết kế theo yêu cầu sau:

a/ Tạo clock có chu kỳ 20ns

- Thời gian trễ lối vào/ra bằng 70% chu kỳ clock
- Latency, transition và uncertainty 5% chu kỳ clock
- set_load, set_max_capacitance and operation conditions tham khảo trong thư viện được cung cấp.
- Báo cáo timing và nhận xét

b/ Tương tự (a) nhưng *Thời gian trễ lối vào/ra* 60% chu kỳ clock

c/ Thực hiện tương tự (a), tìm tần số tối đa thiết kế có thể thực hiện.

SCRIPT Mẫu

```
lappend search_path
define_design_lib WORK -path "work"

set link_library [ list …]
set target_library [ list …]


## read the verilog files
read_verilog -rtl { counter.v }

analyze -library WORK -format verilog  counter.v
elaborate -architecture verilog -library WORK counter

## check if design is consistent
uniquify

check_design

## set enviroment
set_min_library "lib_max.db" -min_version "lib_min.db"

## create constraints

## create constraints
set_operating_conditions -min fast -max slow
set_load 0.1  [all_outputs]
```

```
set_max_capacitance 1.5 [all_outputs]
set_max_transition 1.5 [all_inputs]
set time_scale  2
set tran [expr (0.05*$time_scale)]
set delay_in [expr (0.6*$time_scale)]
set delay_out_min [expr (0.6*$time_scale)]

#set delay_out_max [expr (0.5*$time_scale)]

create_clock -period $time_scale -waveform {0 1} {clock}
set_clock_uncertainty $tran clock
set_clock_latency $tran clock
set_clock_transition $tran clock

set_input_delay $delay_in [all_inputs] -clock clock
set_output_delay $delay_out_min [all_outputs] -clock clock

## output delay nam trong khoang
## compilation

compile -area_effort medium -map_effort medium


## below commands report area, cell, and timing information needed to analysis

 report_area > synth_area.rpt
 report_cell > synth_cells.rpt
 report_qor  > synth_qor.rpt
 report_resources > synth_resources.rpt
 report_timing -max_paths 20 > synth_timing.rpt


write_sdc counter.sdc
## dump out the synthesis
write -f ddc -hierarchy -output counter.ddc
write -hierarchy -format verilog -output counter.v
```

BÀI 4: PHÂN TÍCH THỜI GIAN TĨNH

1. MỤC ĐÍCH

- Giúp sinh viên phân tích thời gian thiết kế.
- Giúp sinh viên sử dụng công cụ phân tích thời gian
- Giúp sinh viên đọc và hiểu tập tin sau khi phân tích thời gian

2. HƯỚNG DẪN THỰC HÀNH

2.1. Thiết kế bộ đếm Gray 2 bit

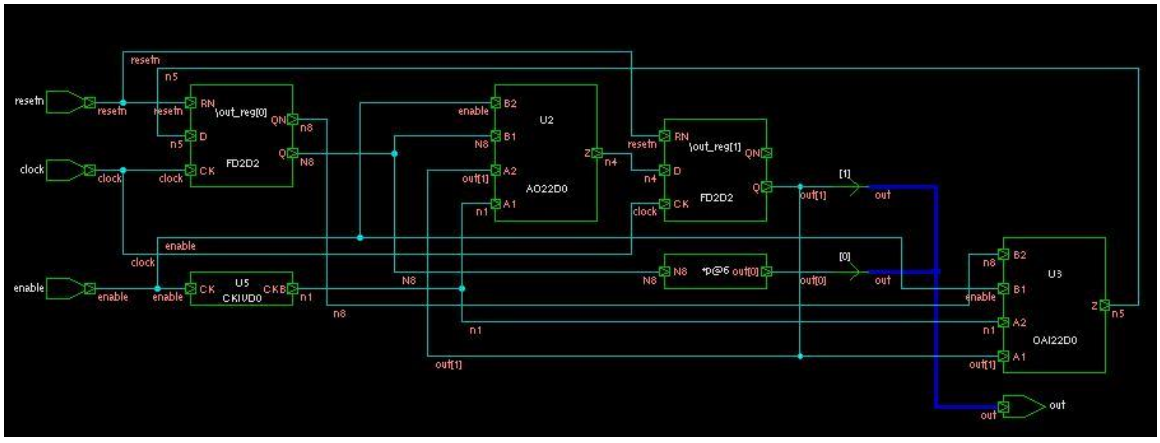
2.1.1. Mã Gray

State	Gray code
1	00
2	01
3	11
4	10

2.1.2. Thiết kế và viết testbench sử dụng ngôn ngữ verilog và lưu graycounter.v và tb_graycounter.v

2.2. Tạo script TCL (dc_script.tcl) với rang buộc sau:

- Tạo clock 2ns
- Delay inputs 60% của chu kỳ clock
- Delay outputs 60% của chu kỳ clock
- Latency, transition và uncertainty 5% của chu kỳ clock
- set_load, set_max_capacitance và operation conditions theo thư viện.

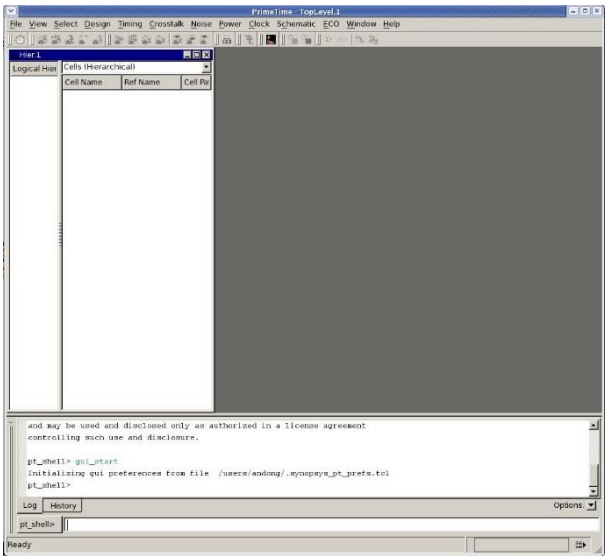


So sánh kết quả mô phỏng giữa RTLvà nettlist .

Khởi động công cụ Primetime (PT)

 # softPT

 # pt_shell -gui &



Hình : Primetime graphic

- Thiết đặt thư viện
  set link_library *[list /.../lib/ saed90nm_typ.db* \
  .../lib/saed90nm_max.db \
  /.../lib/ saed90nm_min.db]

  set target_library *[list /.../lib/ saed90nm_max.db* \
  /.../lib/ saed90nm_min.db \
  /.../lib/ saed90nm_typ.db]
- Đọc thiết kế (netlist)
  read_verilog -rtl */.../graycounter.v*
- Thiết lập module top
  current_design *graycounter*
- Đọc SDC file
  source */.../ graycountersdc.sdc*

➢ Xuất báo cáo 6 loại phân tích của thiết kế:

✓ Từ inputs đến tất cả flipflop tong thiết kến

report_timing -max_paths 20 -from *[all_inputs]* **to** *[all_registers -data_pins]*

> */.../reports/inputstoflops.rpt*

✓ Từ flipflop đến flipflop

report_timing -max_paths 20 -from *[all_registers -clock_pins]* **to** *[all_registers -data_pins]* **>** */.../reports/floptoflop.rpt*

✓ Từ flipflop đến output

report_timing -max_paths 20 -from *[all_registers -clock_pins]* **to** *[all_registers -data_pins]* **>** */.../reports/floptooutputs.rpt*

✓ Từ inputs đến outputs

report_timing -max_paths 20 -from *[all_inputs]* **to** *[all_outputs]*

> */.../reports/inputstouotputs.rpt*

✓ Báo cáo setup time và hold time

report_timing -from *[all_registers -clock_pins]* **-to** *[all_registers -data_pins] -delay_type max* **>** */.../reports/setuptiming.rpt*

report_timing -from *[all_registers -clock_pins]* **-to** *[all_registers -data_pins] -delay_type min* **>** */.../reports/holdtiming.rpt*

✓ Báo cáo thời gian chuyển mạch transition và tụ của các path

report_timing -transition_time -capacitance -nets -input_pins -from

[all_registers -clock_pins] ***-to*** *[all_registers -data_pins]* > reports/tran_captiming.rpt

Chú ý: Nếu thời gian bị vi phạm bạn phải tổng hợp hoặc thiết kế để đảm bảo thiết kế chuẩn bị cho bước back-end

➢ Giải thích các báo cáo

3. THỰC HÀNH

3.1. Thực hành tại lớp các mục 2

3.2. Bài tập về nhà các mục 2 với thiết kế ở bài 1, 2 và 3

3.3. Tạo script TCL cho PrimeTime script (pt_script.tcl).
Sinh viên làm báo cáo theo mẫu và nộp lại cho GV sau mỗi buổi học

Chú ý: **max-paths** với tham số tối thiểu là 20 path cho thiết kế các bài 1, 2 và 3.

BÀI 5: THIẾT KẾ BACK-END

1. MỤC ĐÍCH

- Giúp sinh viên layout thiết kế
- Giúp sinh viên sử dụng công cụ để layout thiết kế

2. YÊU CẦU

Để thực hiện bài thực hành này, Sinh viên cần hoàn thành các bài thực hành trước và chuẩn bị các tập tin sau:

- Netlist

- Tập tin chứa ràng buộc timing được tạo ra từ phần mềm tổng hợp mạch (SDC)

3. HƯỚNG DẪN THỰC HÀNH

3.1. Thư viện logical và định nghĩa VDD, VSS

#Prepair data

lappend search_path */users/andong/Desktop/lib*

set link_library "lib.db"

set target_library "lib.db"

set_min_library lib.db ***-min_version*** lib.db

Reading parasitic information from tlu+ (.tlup) files

set_tlu_plus_files \

-tech2itf_map lib.map \

-max_tluplus lib_max.tlup \

-min_tluplus lib_min.tlup

set mw_logic0_net *VSS*

set mw_logic1_net *VDD*

3.2. Tạo thư viện Milkway (nếu chưa có)

##create milkyway lib //File --> Create lib

#create_mw_lib -technology lib.tf ***mw_reference_library*** lib.mw

3.3. Mở thư viện Milkyway vừa tạo

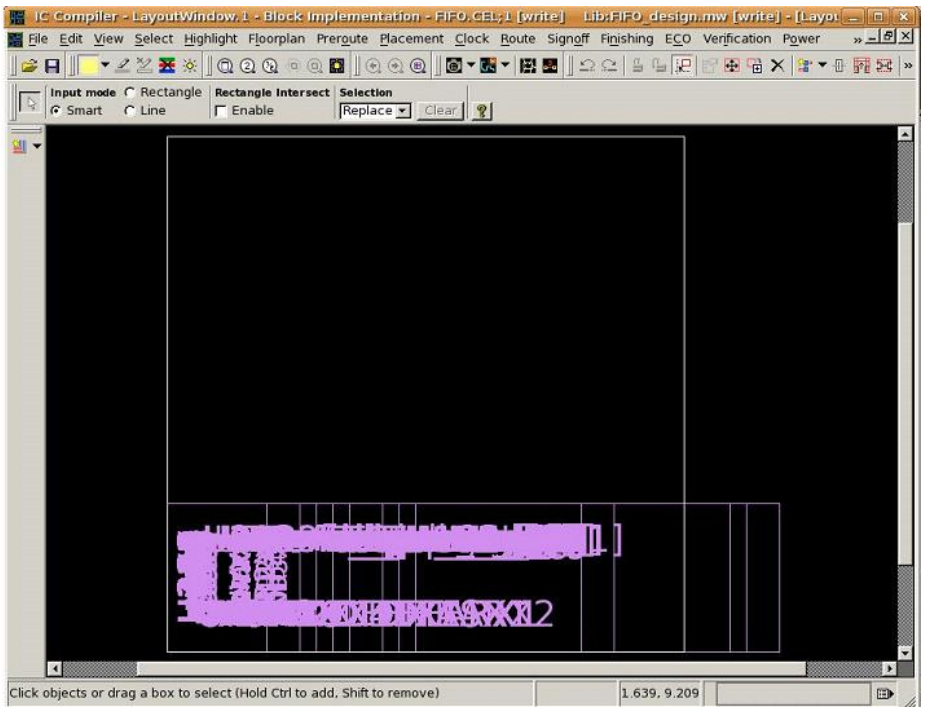
3.4. Đọc netlist

##open milkyway lib //File --> Open lib

open_mw_lib lib.mw

##open netlist design //File --> Import

read_verilog -top top_cpu *{netlist.v}*



3.5. Tạo liên kết

uniquify_fp_mw_cel

##Link the design

link

current_design *top_cpu*

3.6. Đọc tập tin SDC

read_sdc -version Latest "file.sdc"

3.7. Lưu thiết kế

##save the design //File -->Save design

save_mw_cel -as "top_cpu"

3.8. Floorplanning

3.8.1. Khởi tạo floorplan

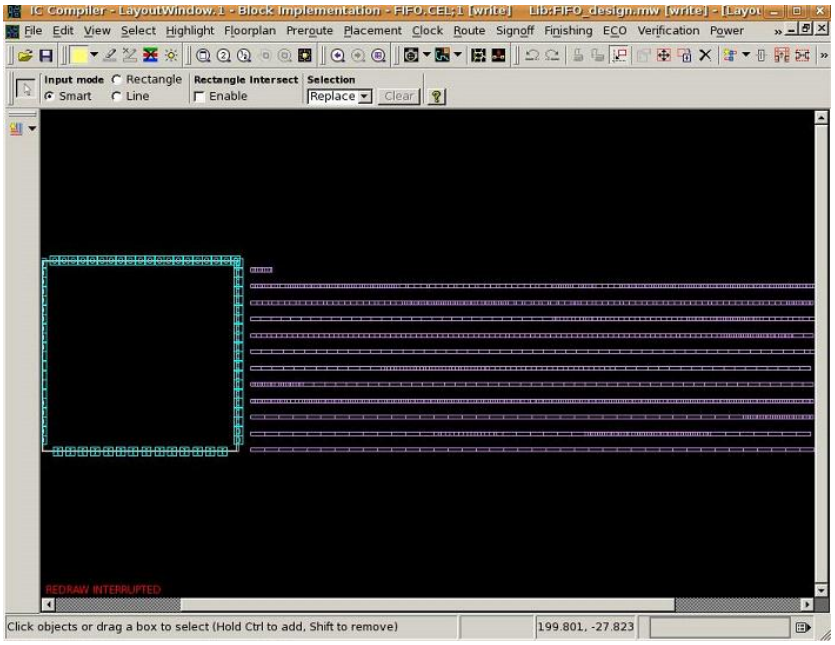
initialize_floorplan -control_type \

width_and_height -core_width 200 ***-core_height*** 200 \

-left_io2core 20 ***-bottom_io2core*** 20 \

-right_io2core 20 ***-top_io2core*** 20 \

-pin_snap -keep_macro_place



3.8.2. Tạo các pin power và ground

##connect power and ground

derive_pg_connection -power_net VDD ***-ground_net*** VSS

derive_pg_connection -power_net VDD ***-ground_net*** VSS ***-tie***

3.8.3. Power Planning

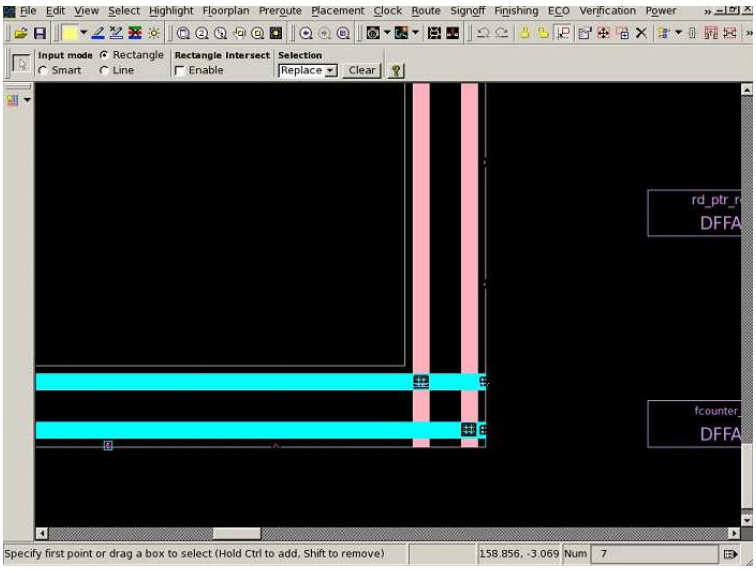
+++ create power-rings +++

create_rectangular_rings -nets {VSS} \

-left_offset 0.5 -left_segment_layer M3 \

-left_segment_width 4 \

```
                    -right_offset  0.5 -right_segment_layer   M3 \
                    -right_segment_width  4 \
                    -bottom_offset        0.5 -bottom_segment_layer  M4 \
                    -bottom_segment_width 4 \
                    -top_offset    0.5 -top_segment_layer     M4 \
                    -top_segment_width    4
create_rectangular_rings -nets {VDD} \
                    -left_offset    1.8 -left_segment_layer        M3 \
                    -left_segment_width   3 \
                    -right_offset  1.8 -right_segment_layer   M3 \
                    -right_segment_width  3 \
                    -bottom_offset        1.8 -bottom_segment_layer  M4 \
                    -bottom_segment_width 3 \
                    -top_offset    1.8 -top_segment_layer     M4 \
                    -top_segment_width    3
##Power strap
create_power_strap -nets "VDD" -layer M3 \
        -direction horizontal -start_at 55 -width 2 -step 50 -num_group 15
create_power_strap -nets "VSS" -layer M3 \
        -direction horizontal -start_at 50 -width 4 -step 50 -num_group 15
create_power_strap -nets "VDD" -layer M4 \
        -direction vertical   -start_at 55 -width 2 -step 50 -num_group 15
create_power_strap -nets "VSS" -layer M4 \
        -direction vertical   -start_at 50 -width 4 -step 50 -num_group 15
# create power-rail
preroute_standard_cells -mode rail -nets "VSS VDD"
```



3.8.4. Lưu thiết kế

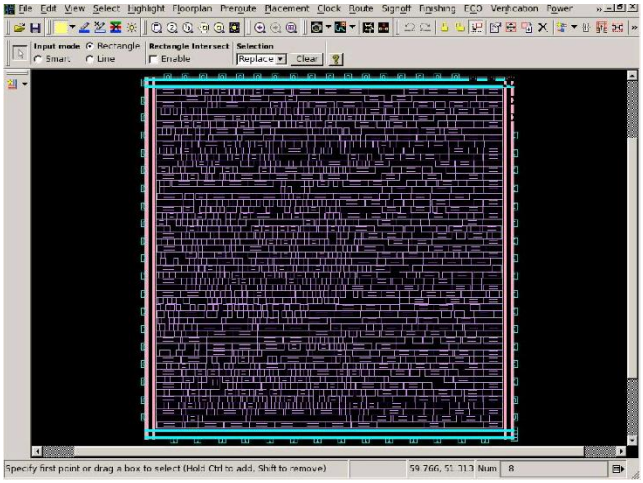
save_mw_cel -as "top_cpu_floorplan"

3.9. **Placement**

Placement and optimization

place_opt -effort high

place_opt -congestion



3.10. Clock Tree Synthesis (CTS)

3.10.1. Clock Tree Synthesis

clock_opt -effort high

3.10.2. Lưu thiết kế và cuất báo cáo timing

save_mw_cel -as "top_cpu_cts"

report_placement_utilization > reports/cpu_cts_util.rpt

report_qor_snapshot > reports/cpu_cts_qor_snapshot.rpt

report_qor > reports/cpu_cts_qor.rpt

report_timing –max_paths 20 –delay max > reports/cpu_cts.setup.rpt

report_timing –max_paths 20 –delay min > reports/cpu_cts.hold.rpt

3.10.3. Tối ưu CTS và lưu thiết kế

clock_opt -no_clock_route -only_psyn -fix_hold_all_clocks -only_hold_time

save_mw_cel -as "top_cpu_cts_op

3.11. Routing

3.11.1. Routing global first or detail

route_opt

3.11.2. Lưu thiết kế và xuất báo cáo timing

save_mw_cel -as "top__rout"

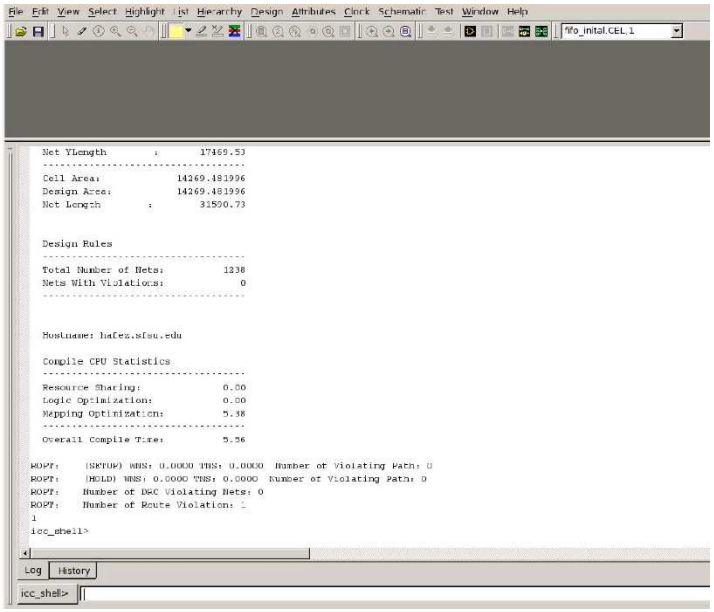
report_placement_utilization > reports/cpu_route_util.rpt

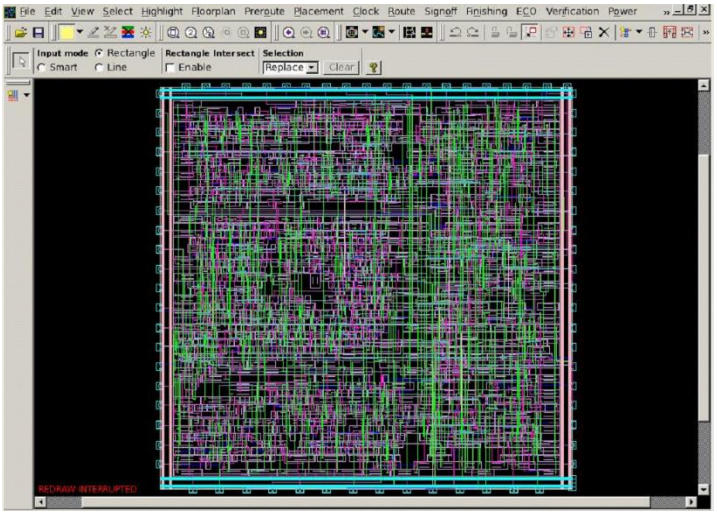
report_qor_snapshot > reports/cpu_route_qor_snapshot.rpt

report_qor > reports/cpu_route_qor.rpt

report_timing –max_paths 20 –delay max > reports/cpu_route.setup.rpt

report_timing –max_paths 20 –delay min > reports/cpu_route.hold.rpt





3.11.3.Tối ưu Post route và lưu thiết kế

route_opt -skip_initial_route

save_mw_cel -as "top_cpu_rout_op"

4. THỰC HÀNH

4.1. Thực hành tại lớp các mục 1 đến mục 6 cho các thiết kế ở bài 1

4.2. Bài tập về nhà tương tự 6.1 cho bài 2

4.3. Tạo script TCL cho ICC script (icc_script.tcl).

Sinh viên làm báo cáo theo mẫu và nộp lại cho GV sau mỗi buổi học

BÁO CÁO THỰC TẬP

Nhóm: ……..

- ……MSSV:……
- ……MSSV:……..
- ……MSSV:……..

Bài.....

Thiết kế

.....

.....

.....

.....

Testbench

.....

.....

.....

.....

Mô phỏng (phải có dạng sóng)

.....

.....

.....

.....

.....