Chương 4

Bài thực hành-thí nghiệm 2: Thiết kế bộ đếm

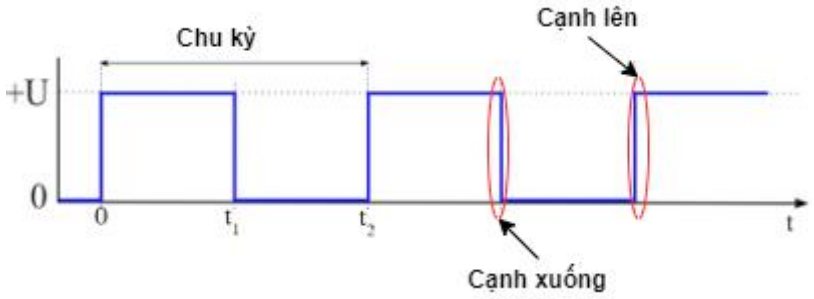
4.1. Lý thuyết

4.1.1. Mạch tổ hợp và mạch tuần tự

Trước khi vào thiết kế bộ đếm, ta cần phải làm rõ điểm khác biệt giữa mạch tổ hợp và mạch tuần tự. Mạch tổ hợp là mạch mà trạng thái đầu ra của mạch chỉ phụ thuộc vào trạng thái đầu vào ở cùng thời điểm mà không phải trạng thái đầu vào ở thời điểm trước đó. Ngược lại, mạch tuần tự có trạng thái đầu ra phụ thuộc vào trạng thái trước đó. Nói cách khác, mạch tuần tự cần có tính nhớ, cần lưu giữ giá trị một hoặc nhiều tín hiệu để dùng vào thời điểm phía sau.

Như vậy, mạch tuần tự hoạt động thay đổi theo thời gian. Để xây dựng được mạch tuần tự, ta cần hai yếu tố quan trong sau đây:

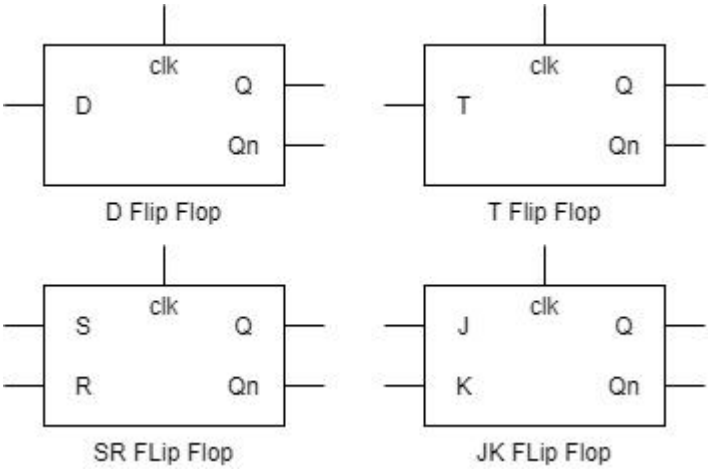
Clock: Đây là tín hiệu dùng để định các khoảng thời gian mà mạch tuần tự sẽ hoạt động. Clock có dạng thường thấy là xung vuông (hình 4.1), trong đó các thời điểm cạnh lên của xung thường được chọn làm thời điểm mà mạch tuần tự thay đổi trạng thái.



Hình 4.1. *Tín hiệu clock.*

Flip-flop: Đây là các phần tử nhớ (hình 4.2), ở đó ngõ ra của các flip-flop này sẽ thay đổi theo cạnh lên (hoặc cạnh xuống) của tín hiệu clock.

Thực tế mà nói, mạch tuần tự chính là chứa mạch tổ hợp bên trong nó, là sự kết hợp giữa mạch tổ hợp cùng các phần tử nhớ, trong đó các phần tử nhớ thay đổi trạng thái tùy theo tín hiệu clock. Và trong sử dụng hiện nay, gần như tất cả các vi mạch đều là mạch tuần tự.



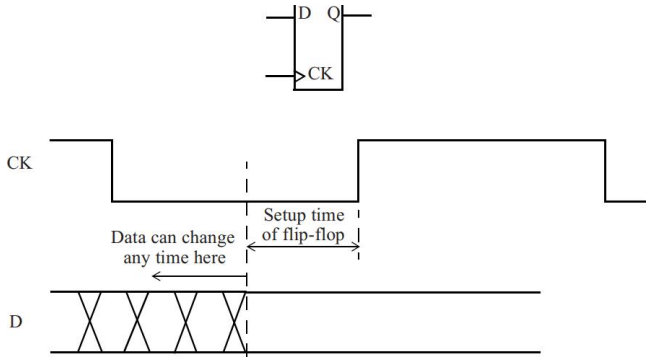
Hình 4.2. *Các loại flip flop thông dụng.*

4.1.2. Setup time và hold time

Phân tích timing là một trong những phân tích quan trọng nhất đối với vi mạch nói chung và mạch tuần tự nói riêng. Ràng buộc về timing sẽ cho ta biết liệu rằng vi mạch có hoạt động ổn định ở tần số mà mình mong muốn. Và nói đến tần số ở đây chính là nói đến tần số (nghịch đảo của chu kỳ) của tín hiệu clock. Tần số càng cao (chu kỳ càng nhỏ) đồng nghĩa với việc mạch hoạt động càng nhanh; do chu kỳ rút ngắn xuống thì vi mạch làm được nhiều việc hơn trong cùng một khoảng thời gian

Tuy nhiên, muốn tăng được tần số mà vẫn giữ được vi mạch hoạt động bình thường, ta cần phải chú ý xem xét đến cấu trúc các mạch tổ hợp bên trong vi mạch. Và để đạt được một phân tích có tính định lượng, người ta đặt ra hai đại lượng là setup time và hold time.

Trước hết, ta sẽ nói về setup time. Đây là khái niệm dùng để chỉ thời gian mà một flip flop cần ngõ vào của nó giữ nguyên giá trị để nó có thể lấy đúng giá trị vào. Nói cách khác, giá trị ngõ vào của flip flop có thể thay đổi bất kì trước khi bước vào thời gian setup của flip flop và phải giữ nguyên đến khi flip flop lấy được giá trị của nó. (hình 4.3)



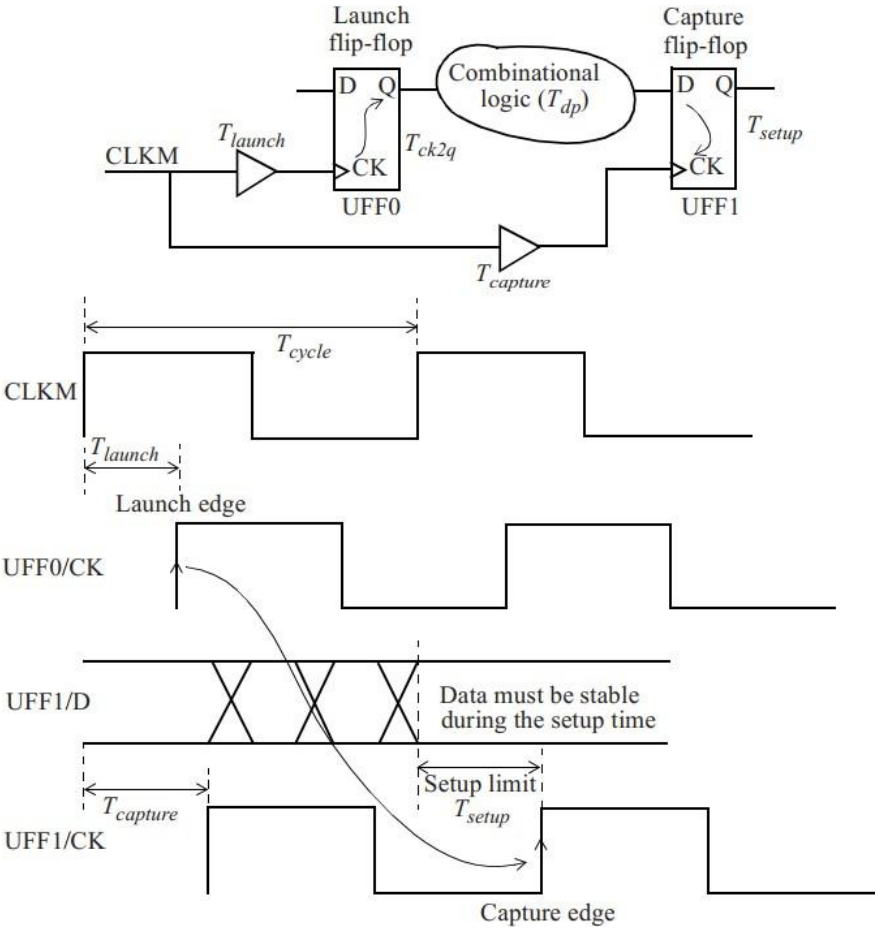
Hình 4.3. *Yêu cầu về setup time của flip flop.*

Thông thường mà nói, ta sẽ có 2 flip flop, một flip flop cho ra dữ liệu, dữ liệu này sẽ đi qua một mạch tổ hợp, và đến một flip flop thứ hai có nhiệm

vụ lấy dữ liệu đó. Khi đó, dữ liệu khi đi qua mạch tổ hợp cần đến được flip flop thứ hai trước khi nó bắt đầu vào setup time (hình 4.4). Cần lưu ý rằng trong thực tế, dữ liệu khi đi qua các phần tử linh kiện, thậm chí dây dẫn cũng đều tiêu tốn một lượng thời gian. Công thức nhằm đảm bảo cho setup time trong trường hợp là

$$Tlaunch + Tck2q + Tdp \leq Tcapture + Tcycle - Tsetup \quad (1)$$

trong đó Tlaunch là thời gian từ clock nguồn đến chân clock của flip flop thứ nhất, Tck2q là thời gian từ khi flip flop thứ nhất có clock đến khi có ngõ ra của flip flop này, Tdp là thời gian dành cho mạch tổ hợp xử lý dữ liệu; Tcapture là thời gian từ clock nguồn đến chân clock của flip flop thứ 2, Tcycle là chu kỳ của clock, Tsetup là thời gian setup của flip flop thứ 2. Cần lưu ý rằng, giả sử dữ liệu A được flip flop thứ nhất lấy được trong chu kỳ thứ n, thì nó mất một chu kỳ nữa dữ liệu A được xử lý qua mạch tổ hợp mới đến được flip flop thứ hai trong chu kỳ n + 1. Đó là lý do mà ta phải cộng thêm Tcycle trong công thức ở trên.

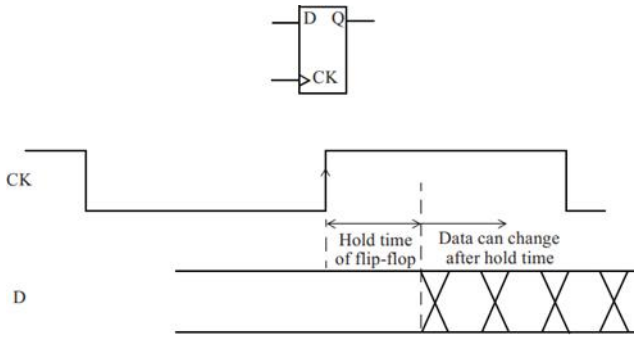


Hình 4.4. *Ví dụ cho kiểm tra điều kiện setup của flip flop.*

Ta cần chú ý hai điều quan trọng sau. Thứ nhất, nếu ta tăng tần số của clock, tức giảm Tcycle, vế phải của (1) sẽ trở nên nhỏ đi và có khả năng bất

đẳng thức không còn đúng; đó là lý do ta không thể cứ mãi tăng tần số của clock. Thứ hai, nếu ta tối ưu mạch tổ hợp, tức làm giảm Tdp, vế trái của (1) cũng đồng thời giảm đi và làm tăng khả năng vượt qua được kiểm tra setup.

Tiếp theo ta xét đến hold time. Hold time là khoảng thời gian mà dữ liệu phải được giữ nguyên đủ lâu để flip flop có thể bắt được dữ liệu đúng với chu kỳ của nó. Nói cách khác, nếu dữ liệu đi quá nhanh, nó có thể đè lên dữ liệu cũ trước khi mà flip flop bắt được cho đúng. (hình 4.5)

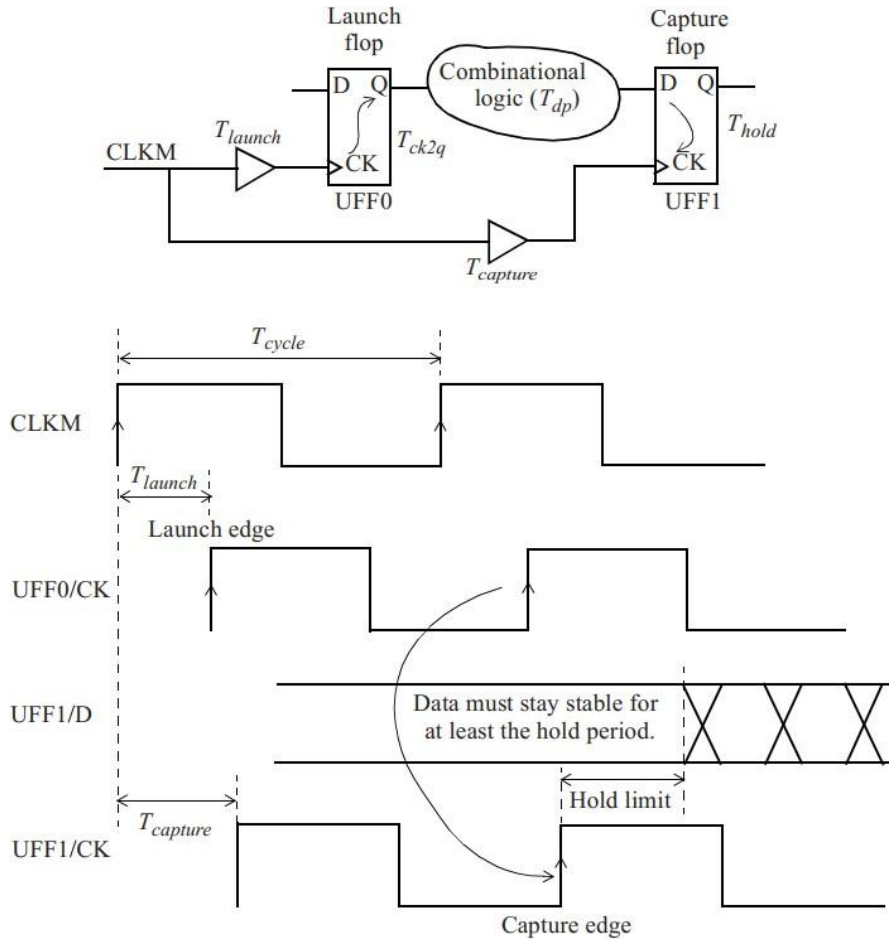


Hình 4.5. *Yêu cầu về hold time của flip flop.*

Lấy ví dụ cho 2 flip flop như ở trên (hình 4.6). Ta sẽ xét đến cạnh lên thứ 2 của CLKM trong hình, dữ liệu bắt đầu từ cạnh lên này sẽ mất Tlaunch +Tck2q + Tdp để đến được chân D của flip flop thứ hai. Cũng cùng cạnh lên đó, từ CLM đến chân clock của flip flop thứ hai sẽ mất thời gian Tcapture. Mong muốn của chúng ta, là trong cạnh lên clock này, flip flop thứ hai sẽ bắt đượcdữ liệu xuất ra từ flip flop thứ nhất của chu kì trước, tức là dữ liệu xuất ra của flip flop thứ nhất trong cạnh lên clock này sẽ được nhận bởi flip flop thứ hai trong chu kì tiếp theo. Nếu dữ liệu xuất ra của flip flop thứ nhất trong cạnh lên clock này đi quá nhanh, nó sẽ ghi đè lên dữ liệu xuất ra của flip flop thứ nhất trong chu kỳ trước và flip flop thứ hai lúc này sẽ bắt sai dữ liệu. Khoảng thời gian hold time là khoảng thời gian để đảm bảo dữ liệu mà flip flop thứ hai bắt được sẽ không bị ghi đè mất. Công thức để kiểm tra điều kiện hold time, nhằm đảm bảo khoảng thời gian chênh lệch giữa thời điểm dữ liệu đến flip flop và thời điểm clock đến flip flop phải lớn hơn thời gian hold time của flip flop đó, để đảm bảo dữ liệu cũ đến flip flop không bị ghi đè và được bắt lại bởi flip flop trên

$$Tlaunch + Tck2q + Tdp \geq Tcapture + Thold \quad (2)$$

Đối với hold time, ta cũng cần chú ý đến hai điều. Thứ nhất, kiểm tra điều kiện về hold không liên quan đến chu kỳ của clock. Thứ hai, nếu mạch tổ hợp quá nhanh dẫn đến vi phạm về điều kiện hold, ta cần thêm các khối đệm (có thể là hai cổng NOT nối tiếp) nhằm làm tăng thời gian xử lý của mạch tổ hợp, hay tăng vế trái của (2).



Hình 4.6. *Ví dụ cho kiểm tra điều kiện hold của flip flop.*

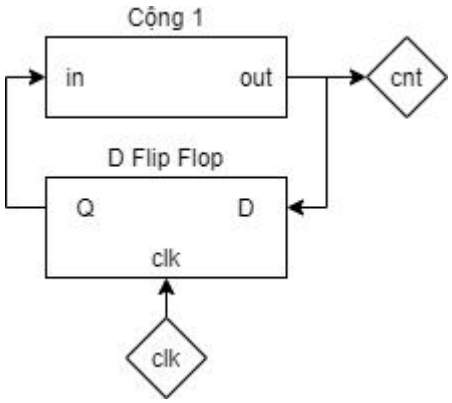
4.1.3. Thiết kế bộ đếm lên

Bài lab thứ 2 này sẽ hướng dẫn sinh viên thực hiện một mạch tuần tự rất đơn giản là bộ đếm lên. Ở đó:

- Ngõ vào: tín hiệu clock tên clk, độ rộng 1 bit.

- Ngõ ra: số đếm tên cnt, có giá trị tăng dần sau mỗi cạnh lên của clock, độ rộng 16 bit.

Hình 4.7 cho ta mô tả về bộ đếm lên. Ở đó khối cộng 1 có nhiệm vụ là mạch tổ hợp mà ngõ ra lớn hơn ngõ vào một đơn vị. D flip flop có nhiệm vụ cập nhật ngõ vào của mạch tổ hợp theo clock với đúng giá trị mà bộ cộng 1 xuất ra trong chu kỳ trước đó.



Hình 4.7. *Thiết kế bộ đếm lên.*

4.2. Thực hiện thiết kế

Bảng 4.1: Mô tả bước thiết kế cấp độ hệ thống.

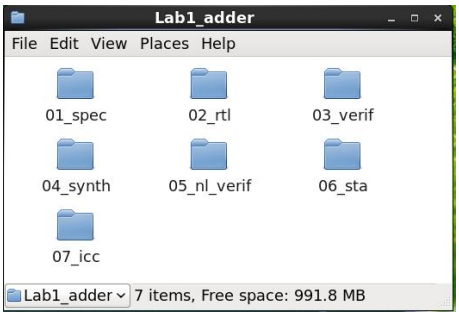
Đầu vào	**Đầu ra**	**Các công đoạn**
Ý tưởng thiết kế	- Mô tả cụ thể thiết kế (gọi chung là Specifi- cation), có thể là file Word, hình vẽ sơ đồ khối,…	- Nắm rõ mục đích của thiết kế - Xây dựng thiết kế nhằm phục vụ mục đích trên - Sửa chữa lỗi và hoàn thiện

Bảng 4.1 là mô tả khái quát của bước thiết kế cấp độ hệ thống.

Phần này ta đã mô tả trong phần lý thuyết.

4.2.1. Tạo dựng môi trường và các chuẩn bị khác

Về mặt cấu trúc thư mục, ta sẽ trình bày giống hệt bài lab 1. (hình 4.8). Cách chia sẻ thư mục thông qua kênh thư mục chia sẻ cũng được thực hiện tương tự bài lab trước.



Hình 4.8. *Cấu trúc thư mục trong lab 2.*

4.2.2. Mô tả thiết kế bằng Verilog

Bảng 4.2: Mô tả bước thiết kế cấp độ RTL.

Đầu vào	Đầu ra	Các công đoạn
Specification	Các file mã lập trình đuôi .v, trong đó mô tả các khối trong thiết kế cùng kết nối giữa chúng thông qua ngôn ngữ Verilog	- Dùng Verilog mô tả lần lượt các khối trong thiết kế - Kết nối các khối con và instance chúng trong các khối lớn hơn - Kết nối các khối lớn hơn cho đến khi đến cấp độ cao nhất

Bảng 4.2 là mô tả khái quát của bước thiết kế cấp độ RTL.

Ở mục này, ta sẽ thực hiện thiết kế 3 file Verilog:

add_1.v: Khối cộng 1 mô tả ở trên.

DFF.v: D flip flop

counter.v: bộ đếm tổng hợp hai file trên.

Nội dung cũng như giải thích ngắn gọn các file trên được thể hiện trong hình 4.9.

Bảng 4.3: Mô tả bước kiểm tra thiết kế cấp độ RTL (Verification).

Đầu vào	Đầu ra	Các công đoạn
- Các file mã lập trình đuôi .v - Các file testbench cũng đuôi .v hay .sv (SystemVerilog)	- Các báo cáo (report) về quá trình mô phỏng - Các file dạng sóng (waveform) nhằm kiểm tra mô phỏng thiết kế	- Xây dựng môi trường cho việc kiểm tra - Kiểm tra lỗi cú pháp của RTL (các file .v) và testbench - Chạy mô phỏng và xuất ra dạng sóng - Kiểm tra các file báo cáo (report) và dạng sóng nhằm đảm bảo mô phỏng RTL sạch lỗi

Bảng 4.3 là mô tả khái quát của bước kiểm tra thiết kế cấp độ RTL (Verification).

add_1.v

```
module add_1(in,out);
input [15:0] in;
output [15:0] out;

assign out  = in + 1;

endmodule
```

← Mạch tổ hợp cộng thêm 1

DFF.v

```
module DFF(in,clk,out);
input [15:0] in;
input clk;
output reg [15:0] out;

initial begin out <=0; end

always @(posedge clk) begin
        out <= in;
end

endmodule
```

← Khởi tạo giá trị bằng 0

← Khối always hoạt động theo clock

counter.v

```
module counter(clk,out);
input clk;
output [15:0] out;

wire [15:0] cnt_add1;
wire [15:0] cnt;

add_1 add1_inst(cnt,cnt_add1);
DFF ff_inst(cnt_add1,clk,cnt);

assign out = cnt_add1;

endmodule
```

← Gọi các khối con

Hình 4.9. *Code Verilog của lab 2.*

Về cơ bản, ta chỉ cần sửa nội dung file testtop.v trong bài lab trước để áp dụng cho bài lab này. Nội dung của file trên được thể hiện trong hình 4.10.

Sau khi thêm đường dẫn của testtop.v vào file lab_ben.flist, ta thực hiện chạy giống hệt bài lab trước. Lưu ý là cần sửa lại đường dẫn trong file set_env.bash để trỏ tới thư mục của bài lab hiện tại.

testtop.v

```
module testtop;

reg clk;
wire [15:0] out;

counter counter_inst(clk,out);

always begin #5 clk=~clk; end

initial begin
        clk =0;
        #100;
        $finish;
end

always @(out) begin
        #1;
        $display("out = %d",out);
end

endmodule
```

← Tạo clock chu kỳ 10s

← Delay 100s để bộ đếm đếm 10 lần

← In giá trị ngõ ra sau mỗi clock

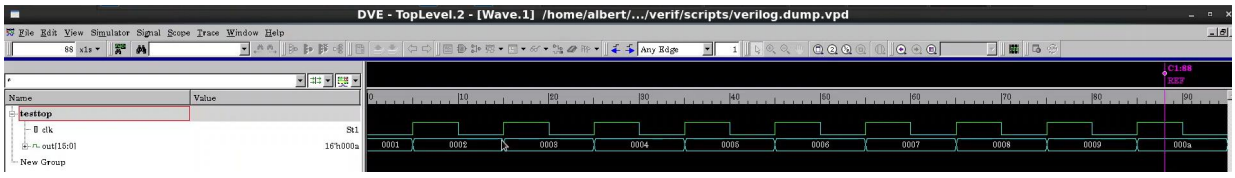
Hình 4.10d. *Nội dung file* ***testtop.v****.*

Kết quả sau khi chạy mô phỏng được thể hiện trong hình 4.11.

```
Compiler version M-2017.03-SP2_Full64; Runtime version M-2017.03-SP2_Full64;  Aug  4 02:52 2024
out =     1
out =     2
out =     3
out =     4
out =     5
out =     6
out =     7
out =     8
out =     9
out =    10
out =    11
$finish called from file "/home/albert/Desktop/Lad2_counter/03_verif/verif/sv/testtop.v", line 13.
$finish at simulation time                  100
          V C S   S i m u l a t i o n   R e p o r t
```

Hình 4.11. *Kết quả mô phỏng của bài lab 2.*

Tương tự, kết quả dạng sóng được thể hiện trong hình 4.12.



Hình 4.12. *Dạng sóng mô phỏng của bài lab 2.*

4.2.3. Tổng hợp (Synthesis)

Bảng 4.4: Mô tả bước Synthesis (tổng hợp) thiết kế.

Đầu vào	Đầu ra	Các công đoạn
- Các file mã code đuôi .v - Các file thư viện đuôi .db nhằm cung cấp các cell tham khảo cho quá trình chuyển RTL sang các cell này - Một số ràng buộc về timing, area hay power	- Các report báo cáo về quá trình tổng hợp - Các file dùng cho các công đoạn tiếp theo trong quy trình thiết kế (.sdc, .sdf, .ddc,...) - File RTL (.v) đã tổng hợp (đã chuyển về các cell trong thư viện thao khảo), thường gọi là netlist	- Viết script cho quá trình tổng hợp, bao gồm việc lựa chọn thư viện, lựa chọn RTL, thêm các ràng buộc, tổng hợp và xuất kết quả - Chạy script tổng hợp - Kiểm tra và sửa lỗi sau khi tổng hợp (nếu có)

Bảng 4.4 là mô tả khái quát của bước Synthesis (tổng hợp) thiết kế.

Trong phần tổng hợp của bài lab này, ta sẽ thêm vào một số ràng buộc về timing nhằm tạo điều kiện cho tool DC có thể kiểm tra về các điều kiện của setup time và hold time (hình 4.13).

Các ràng buộc được thêm vào ở mục CONSTRAINT FOR DESIGN. Trong đó:

Tạo clock có tên clk (trùng tên với tín hiệu trong file counter.v) có chu kì là 1000ns (Đơn vị dùng trong DC là ns).

Thời gian delay của dữ liệu trước khi đến được input của bộ đếm sẽ nằm trong khoảng [1;10] ns.

Thời gian delay của dữ liệu khi ra ngoài output của bộ đếm cũng tương tự nằm trong khoảng [1;10] ns.

Lưu ý rằng đây là các điều kiện ràng buộc về timing của thiết kế mà người thiết kế tự thiết lập. Về setup time và hold time của các flip flop, các con số này được quy định bởi thư viện

- Tới đây ta chỉ cần chỉnh lại file dc_command.src theo các yêu cầu có sẵn rồi tiến hành chạy tool DC trên Terminal:

```
set link_library "* $target_library"
set synthesis_library standard.sldb

#============= ANALYSE DESIGN ==================
analyze -format verilog "./../../02_rtl/add_1.v"
analyze -format verilog "./../../02_rtl/DFF.v"
analyze -format verilog "./../../02_rtl/counter.v"
elaborate counter
current_design counter

#============= CONSTRAINT FOR DESIGN ============
create_clock -name clk -period 10 {clk}
set_input_delay -max 10 -clock clk  [all_inputs]
set_input_delay -min 1 -clock clk  [all_inputs]
set_output_delay -max 10 -clock clk [all_outputs]
set_output_delay -min 1 -clock clk [all_outputs]

#============= SYNTHESIZE=======================
compile_ultra

#============= REPORT PERFORMANCE ===============
report_area > ./../report/report.area
report_timing > ./../report/report.timing
report_constraint > ./../report/report.constraint
report_qor > ./../report/report.qor
write -f ddc -o ./../report/report.ddc
write -format verilog -hierarchy -output ./../report/lab_synth.netlist.v
write_sdf ./../report/report.sdf
write_sdc ./../report/report.sdc
```

Hình 4.13a. *File* ***dc_command.src*** *của bài lab 2.*

Sau khi kết thúc, ta sẽ mở file report.timing và lab_synth.netlist.v trong thư mục 04_synth/report nhằm quan sát kết quả (hình 4.14).

```
////////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : L-2016.03-SP1
// Date      : Sun Aug  4 03:22:06 2024
////////////////////////////////////////////////////////////////


module counter ( clk, out );
  output [15:0] out;
  input clk;
  wire   n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16,
         n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28, n29, n30,
         n31, n32, n33, n34, n35, n36, n37, n38, n39, n40, n43, n44, n45, n46,
         n47, n48, n49, n50, n51, n52, n53, n54, n55;
  wire   [15:0] cnt;

  DFF_X1 \ff_inst/out_reg[0]  ( .D(out[0]), .CK(clk), .Q(cnt[0]), .QN(out[0])
        );
  DFF_X1 \ff_inst/out_reg[1]  ( .D(out[1]), .CK(clk), .Q(cnt[1]), .QN(n55) );
  DFF_X1 \ff_inst/out_reg[2]  ( .D(out[2]), .CK(clk), .Q(cnt[2]), .QN(n53) );
  DFF_X1 \ff_inst/out_reg[3]  ( .D(out[3]), .CK(clk), .Q(cnt[3]), .QN(n54) );
  DFF_X1 \ff_inst/out_reg[4]  ( .D(out[4]), .CK(clk), .Q(cnt[4]), .QN(n50) );
  DFF_X1 \ff_inst/out_reg[5]  ( .D(out[5]), .CK(clk), .Q(cnt[5]), .QN(n51) );
  DFF_X1 \ff_inst/out_reg[6]  ( .D(out[6]), .CK(clk), .Q(cnt[6]), .QN(n52) );
  DFF_X1 \ff_inst/out_reg[7]  ( .D(out[7]), .CK(clk), .Q(cnt[7]), .QN(n49) );
  DFF_X1 \ff_inst/out_reg[8]  ( .D(out[8]), .CK(clk), .Q(cnt[8]), .QN(n47) );
  DFF_X1 \ff_inst/out_reg[9]  ( .D(n44), .CK(clk), .Q(cnt[9]) );
  DFF_X1 \ff_inst/out_reg[11]  ( .D(n46), .CK(clk), .Q(cnt[11]) );
  DFF_X1 \ff_inst/out_reg[13]  ( .D(out[13]), .CK(clk), .Q(cnt[13]) );
```

```
Operating Conditions: typical   Library: NangateOpenCellLibrary_typ
Wire Load Model Mode: top

  Startpoint: ff_inst/out_reg[6]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: out[12] (output port clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  counter            5K_hvratio_1_1        NangateOpenCellLibrary_typ

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  ff_inst/out_reg[6]/CK (DFF_X1)           0.00       0.00 r
  ff_inst/out_reg[6]/QN (DFF_X1)           0.06       0.06 f
  U19/ZN (INV_X1)                          0.03       0.09 r
  U20/ZN (NAND2_X1)                        0.03       0.12 f
  U21/ZN (NOR2_X1)                         0.04       0.16 r
  U22/ZN (NAND2_X1)                        0.04       0.19 f
  U23/Z (BUF_X1)                           0.04       0.24 f
  U27/ZN (OR2_X1)                          0.05       0.29 f
  U28/ZN (XNOR2_X1)                        0.05       0.34 f
  out[12] (out)                            0.00       0.35 f
  data arrival time                                   0.35

  clock clk (rise edge)                   10.00      10.00
  clock network delay (ideal)              0.00      10.00
  output external delay                  -10.00       0.00
  data required time                                  0.00
  -----------------------------------------------------------
  data required time                                  0.00
  data arrival time                                  -0.35
  -----------------------------------------------------------
  slack (VIOLATED)                                   -0.35
```

Hình 4.14. *Report về netlist và timing sau khi chạy xong DC.*

Ở đây, ta thấy được một phép thử timing duy nhất, có Path Type là max, có nghĩa là kiểm tra điều kiện setup time (min là kiểm tra điều kiện hold time). Phép thử được chia làm 3 phần rõ ràng:

Từ clock clk đến data arrival time là thời gian dữ liệu đến được flip flop, hay vế trái của (1). Từ clock clk tiếp theo đến data required time là vế phải của 1.

Lấy hiệu hai đại lượng trên (slack) để kiểm tra điều kiện setup time. Slack đạt MET có nghĩa là điều kiện về setup time đã thỏa. Nếu vi phạm sẽ báo là VIOLATED.

Đến đây là kết thúc công đoạn Synthesis, ta copy toàn bộ thư mục Lab 2 bỏ vào thư mục chia sẻ.

4.2.4. Kiểm tra netlist

Bảng 4.5: Mô tả bước kiểm tra netlist.

Đầu vào	Đầu ra	Các công đoạn
- Các file mã lập trình RTL đuôi .v - Các file RTL đã tổng hợp (netlist) đuôi .v	- Các báo cáo (report) về quá trình Matching - Các báo cáo (report) về quá trình Verify	- Lựa chọn các file thiết kế RTL - Lựa chọn file RTL đã tổng hợp cùng thư viện tham khảo đi kèm - Thực hiệm kiểm tra Matching và sửa lỗi (nếu có) - Thực hiệm kiểm tra Verify và sửa lỗi (nếu có)

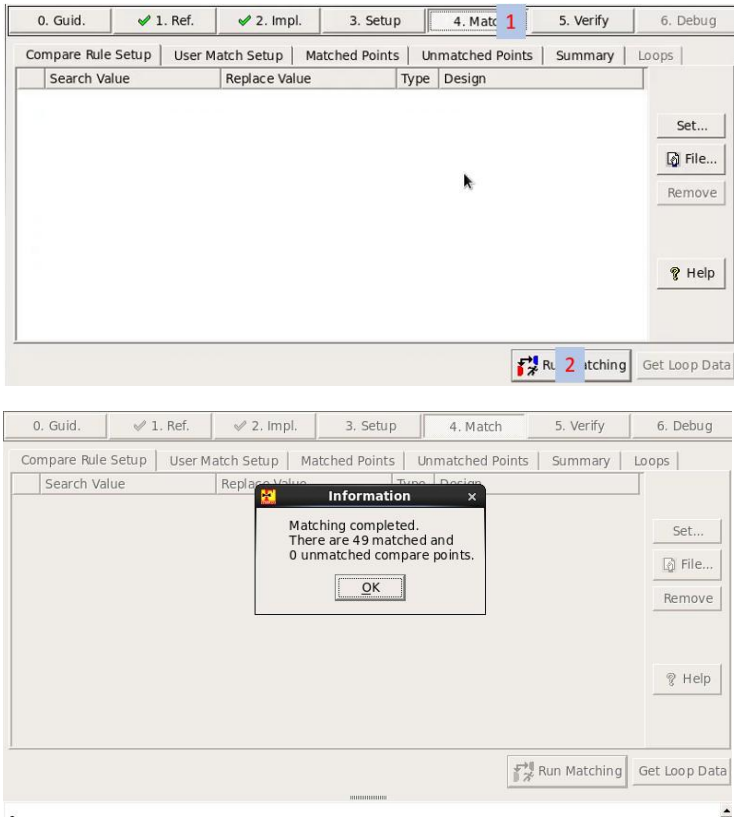
Bảng 4.5 là mô tả khái quát của bước kiểm tra netlist.

Việc thực hiện kiểm tra netlist thông qua tool Formality, ta sẽ thực hiện lại tương tự bài thí nghiệm trước, các bước thực hiện sẽ được tóm tắt dưới đây:

1. Nhấn chọn tab **1.Ref**., nhấn nút **Verilog**.... Chọn tất cả các file Verilog trong thư mục 02_rtl. Sau đó nhấn **Load Files**. Chọn sang tab **3.Set Top Design** (ở dưới) và chọn module counter là top design (thiết kế tổng), rồi nhấn **Set Top.**

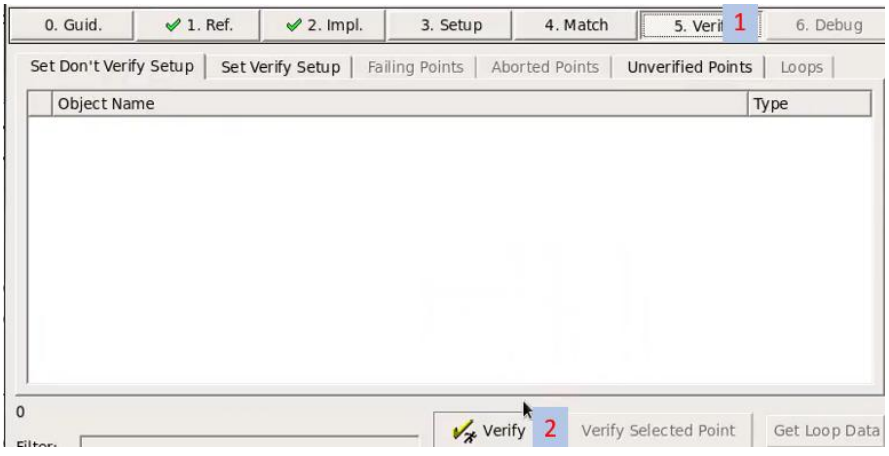
2. Nhấn chọn tab **2.Impl**., nhấn nút **Verilog**.... Chọn file **lab_synth.netlist.v trong thư mục 04_synth/report. Sau đó nhấn Load Files**. Chọn sang tab **2.Read DB** Libraries, nhấn nút DB..., chọn thư viện dùng cho Synthesize trong thư mục 04_synth/lib (ở đây là thư viện **NangateOpenCellLibrary_typical.db**). Sau đó nhấn **Load Files**. Chọn sang tab **3.Set Top Design** (ở dưới) và chọn **module counter** là top design (thiết kế tổng), rồi nhấn Set Top.

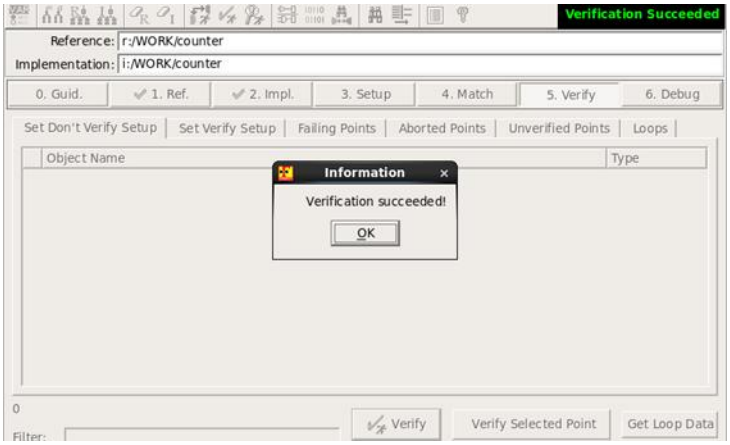
3. Nhấn chọn tab **4.Match**., nhấn chọn **Run Matching** để kiểm chứng sự tương đồng của RTL khi thiết kế và sau khi Synthesis.



Hình 4.15. *Kiểm tra tương đồng giữa RTL trước và sau khi Synthesize.*

4. Nhấn chọn tab **5.Verify**., nhấn chọn **Verify** để kiểm chứng chức năng của RTL sau khi Synthesis.





Hình 4.16. *Kiểm tra chức năng RTL sau khi Synthesize.*

5. Nhấn chọn tab **6.Debug..** Chức năng này được sử dụng khi có lỗi xảy ra trong quá trình so sánh RTL trước và sau Synthesize.

4.2.5. Phân tích thời gian tĩnh STA

Bảng 4.6: Mô tả bước phân tích thời gian tĩnh (STA).

Đầu vào	Đầu ra	Các công đoạn
- Các file RTL đã tổng hợp (netlist) đuôi .v - Các ràng buộc đã sử dụng trong qúa trình tổng hợp (file .sdc) - Các ràng buộc mới thêm vào	- Các báo cáo (report) về kiểm tra Design - Các báo cáo (report) về kiểm tra STA	- Lựa chọn thư viện tham khảo đuôi .db (thư viện dùng trong Synthesis) - Liên kết thiết kế (file RTL đã tổng hợp) - Thêm các ràng buộc cho STA - Thiết lập chế độ hoạt động - Chạy kiểm tra STA và xuất các báo cáo (report) - Kiểm tra các báo cáo và sửa lỗi (nếu có)

Bảng 4.6 là mô tả khái quát của bước phân tích thời gian tĩnh (STA).

Để thực hiện phân tích thời gian tĩnh (thông qua tool PrimeTime của Syn- opsys), thư mục làm việc cùng thư viện đi kèm cần được tải xuống máy ảo thông qua kênh chia sẻ thư mục. Thư mục 06_sta sẽ là thư mục làm việc chính ở bước này, với cấu trúc nội dung như sau:

- File **sta_command.src**: File này có nội dung tương tự file dc_command.src và dùng để chạy các lệnh STA (hình 4.17).

- Thư mục **report_design**: chứa các report liên quan về thiết kế như phân cấp, cell, dây,...

- Thư mục **report_sta**: chứa các report liên quan về STA, đặc biệt là về timing (điều kiện setup/hold).

Ta dùng lệnh **source sta_command.src** để chạy các lệnh phân tích thời gian tĩnh chứa trong file này. Mô tả và giải thích các lệnh này được thể hiện trong hình 4.18. Sau khi chạy xong quá trình STA, ta có thể **quan sát các report trong các thư mục report_design và report_sta** để đánh giá hoạt động hiệu quả của vi mạch và nếu có lỗi xảy ra, ta cần xem xét lại thiết kế hoặc chỉnh sửa về ràng buộc.

```
#!/bin/bash

#========================== SET DIRECTORY ===============
set search_path "/home/hoangtrang/Desktop/icc_lab/logical_lib"
set osearch_path [ concat $search_path \
                ]

#========================== ADD THE LIBRARY =============
set target_library "NangateOpenCellLibrary_typical.db"
set link_library "* $target_library"

#======================== LINK DESIGN ================
read_verilog "./../04_synth/report/lab_synth.netlist.v"
current_design counter
link

#======================== CONSTRAINT FOR STA ==========
set_units -time ns -resistance kOhm -capacitance pF -voltage V -current mA
set_max_area 0
create_clock -name clk -period 20 {clk}
set_clock_uncertainty 0.1 [get_clocks clk]
set_clock_latency 2 [get_clocks clk]
set_clock_transition -max -rise 0.1 [get_clocks clk]
set_clock_transition -max -fall 0.1 [get_clocks clk]
set_clock_transition -min -rise 0.1 [get_clocks clk]
set_clock_transition -min -fall 0.1 [get_clocks clk]
set_input_delay -max 10 -clock clk [all_inputs]
set_input_delay -min 1 -clock clk [all_inputs]
set_output_delay -max 10 -clock clk [all_outputs]
set_output_delay -min 10 -clock clk [all_outputs]
set_fanout_load 8 [all_outputs]

#====================== OPERATING CONDITIONS =========
set_operating_conditions -analysis_type on_chip_variation

#==================== REPORT DESIGN ===============
report_design > report_design/report.design
report_net > report_design/report.net
report_cell > report_design/report.cell
report_hierarchy > report_design/report.hierarchy
report_clock > report_design/report.clock
report_path_group > report_design/report.pg

#====================== REPORT STA ================
report_timing -delay_type max > report_sta/report.timing
report_timing -delay_type min >> report_sta/report.timing
report_port > report_sta/report.port
report_constraint > report_sta/report.constraint
report_global_slack > report_sta/report.glbslk
report_analysis_coverage > report_sta/report.anacov
report_qor > report_sta/report.qor

quit
```

Thêm đường dẫn và liên kết thư viện logic

Liên kết với thiết kế đã tổng hợp

Các ràng buộc dành cho input, output, clock, diện tích,...

Thiết đặt chế độ hoạt động

Xuất các report cho thiết kế

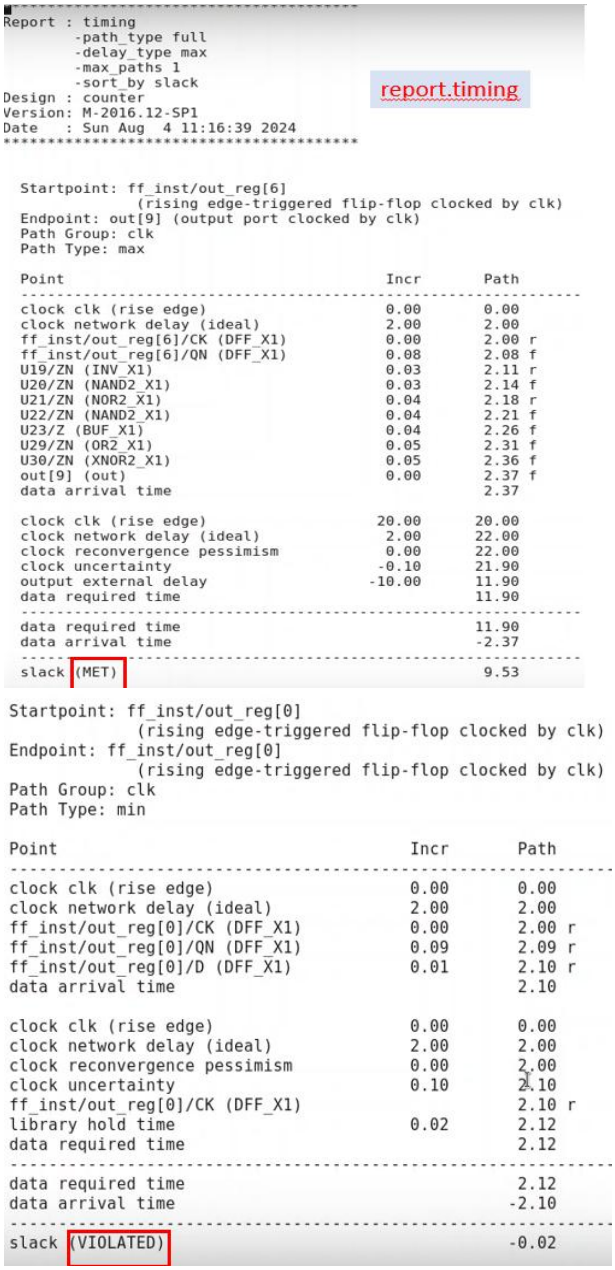
Xuất các report cho STA

Hình 4.17. *File* ***sta_command.src.***

```
****************************************
Report : constraint
Design : counter
Version: M-2016.12-SP1
Date   : Sun Aug  4 11:16:39 2024
****************************************

                                                  Weighted
Group (max_delay/setup)       Cost      Weight      Cost
--------------------------------------------------------
clk                           0.00       1.00       0.00
--------------------------------------------------------
max_delay/setup                                     0.00

                                                  Weighted
Group (min_delay/hold)        Cost      Weight      Cost
--------------------------------------------------------
clk                           0.02       1.00       0.02
--------------------------------------------------------
min_delay/hold                                      0.02

Constraint                                          Cost
--------------------------------------------------------
max_delay/setup                                     0.00  (MET)
min_delay/hold                                      0.02  (VIOLATED)
sequential_clock_pulse_width                        0.00  (MET)
max_capacitance                                     0.00  (MET)
max_transition                                      0.00  (MET)
max_area                                          136.19  (VIOLATED)
```

report.constraint

Hình 4.18. *Kết quả kiểm tra constraint*



Hình 4.19. Kết quả kiểm tra timing.

***Sau khi check tất cả các report, ta thấy có 2 lỗi violated là hold timing và max area. Ta có thể sửa constraints trong file sta_command.src để MET hết các lỗi này theo như gợi ý của sách.

4.2.6. Place and Route

Bảng 4.7: Mô tả bước Place & Route.

Đầu vào	Đầu ra	Các công đoạn
- Các file RTL đã tổng hợp (netlist) đuôi .v - Thư viện tham khảo ở cấp độ vật lý (phải	- Các báo cáo (report) về kiểm tra DRC - Các báo cáo (report) về kiểm tra LVS	- Tạo thư viện Milkyway và import thiết kế - Floor Planning: đặt các thành phần nền tảng lên bề

trùng khớp với thư viện dùng cho Synthesis) - File công nghệ đuôi .tf đi theo thư viện trên - Các ràng buộc cung cấp bởi nhà máy (các file rule về DRC và LVS)	- File GDSII đuôi .gds để gửi đến nhà máy sản xuất	mặt thiết kế (I/O, nguồn, đất,...) - Placement: đặt các cell (trong thư viện tham khảo) ứng với netlist lên Floor vừa tạo - Clock Tree Synthesis: tổng hợp, đi dây cho toàn bộ hệ thống clock, kiểm tra các vi phạm về timing và sửa lỗi (nếu có) - Route: nối dây cho tất cả các kết nối còn lại trong thiết kế - Xuất các file GDSII, .sdf nhằm kiểm tra các bước cuối cùng - Kiểm tra DRC (Design Rule Check) nhằm tìm ra vi phạm với ràng buộc của nhà máy sản xuất (nếu có) - Kiểm tra LVS (Layout vs Schematic) nhằm tìm ra khác biệt giữa Layout (sau khi Place & Route) và netlist trước đó - Nếu không có vấn đề gì, file GDSII sẽ được gửi đến nhà máy để tiến hành sản xuất

Bảng 4.7 là mô tả khái quát của bước Place & Route.

Tương tự như bài thí nghiệm 1, thư mục **07_icc** sẽ là thư mục làm việc chính ở bước này. Trong thư mục này ta sẽ tạo 1 file có tên **icc_setup.TCL** (hình 4.20) nhằm setup cho bài thực hành-thí nghiệm trước khi đi vào thực hiện Place & Route. Kết quả thể hiện trong các hình ảnh sau (từ hình 4.21 đến hình 5.17).

```
#SETUP LIB
set_app_var search_path "/home/hoangtrang/Desktop/icc_lab/logical_lib"
set_app_var target_library "NangateOpenCellLibrary_typical.db"
set_app_var link_library "* $target_library"

#REMOVE OLD LIB
sh rm -rf CHIP

#CREATE MILKYWAY LIB
create_mw_lib -tech "/home/hoangtrang/Desktop/icc_lab/tech/NangateOpenCellLibrary.tf"   \
-mw_reference_library                                                                    \
{/home/hoangtrang/Desktop/icc_lab/physical_lib/NangateOpenCellLibrary                    \
/home/hoangtrang/Desktop/icc_lab/physical_lib/RF_2P_ADV64_16                             \
/home/hoangtrang/Desktop/icc_lab/physical_lib/tpz}                                       \
-open CHIP

set_tlu_plus_files              \
        -max_tluplus "/home/hoangtrang/Desktop/icc_lab/tluplus/NangateOpenCellLibrary.tluplus"  \
        -min_tluplus "/home/hoangtrang/Desktop/icc_lab/tluplus/NangateOpenCellLibrary.tluplus" \
        -tech2itf_map "/home/hoangtrang/Desktop/icc_lab/tluplus/NangateOpenCellLibrary.map"

import_design "./../04_synth/report/lab_synth.netlist.v" -format "verilog" -top "adder_4bit" -cel "adder_4bit"

read_sdc "./../04_synth/report/report.sdc"
```

Setup thư viện logic

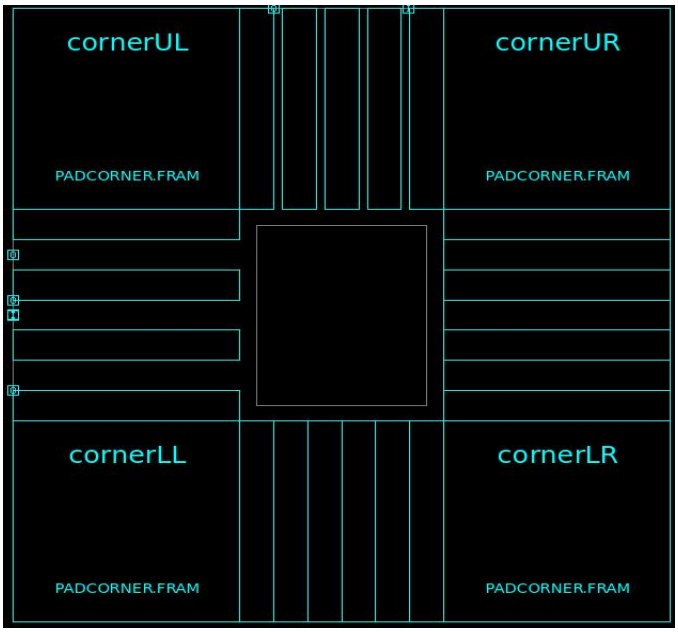
Xóa thư viện MilkyWay cũ (nếu có)

Tạo thư viện MilkyWay mới tên CHIP với file công nghệ sau option -tech và các thư viện vật lý sau option -mw_reference_library

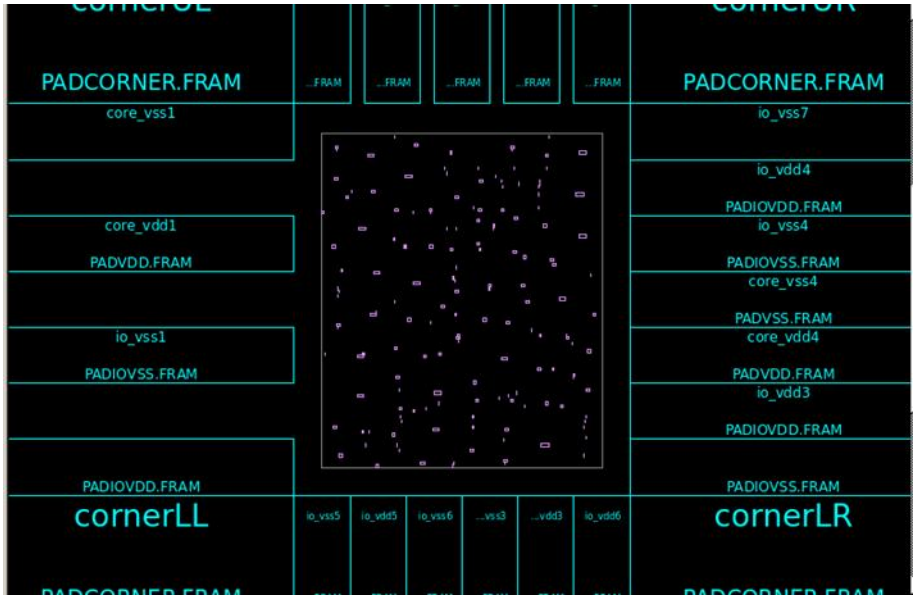
Thiết lập các file TLU+, chứa thông tin nhằm chiết xuất tụ ký sinh

Import thiết kế đã tổng hợp cùng file sdc (ràng buộc) đi kèm

Hình 4.20. *Mô tả nội dung file* ***icc_setup.TCL***



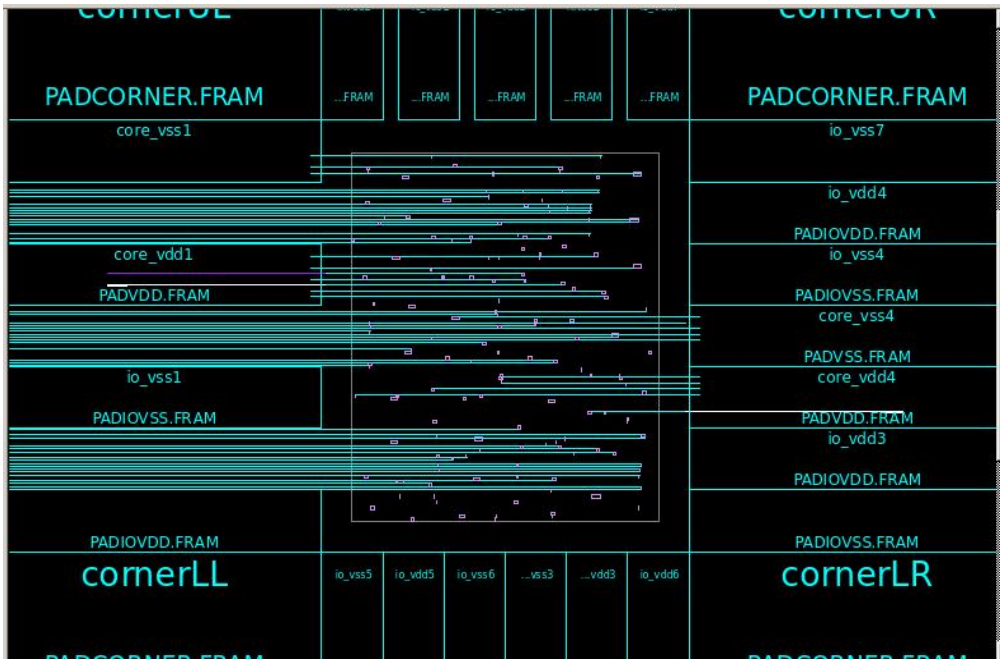
Hình 4.21. Kết quả chạy Floorplanning.



Hình 4.22. *Kết quả chạy Placement.*

```
icc_shell> derive_pg_connection -power_net {VDD} -ground_net {VSS} -power_pin {VDD} -ground_pin {VSS}
Information: connected 169 power ports and 169 ground ports
```

Hình 4.23. *Kết quả chạy nhận tín hiệu nguồn/đất.*

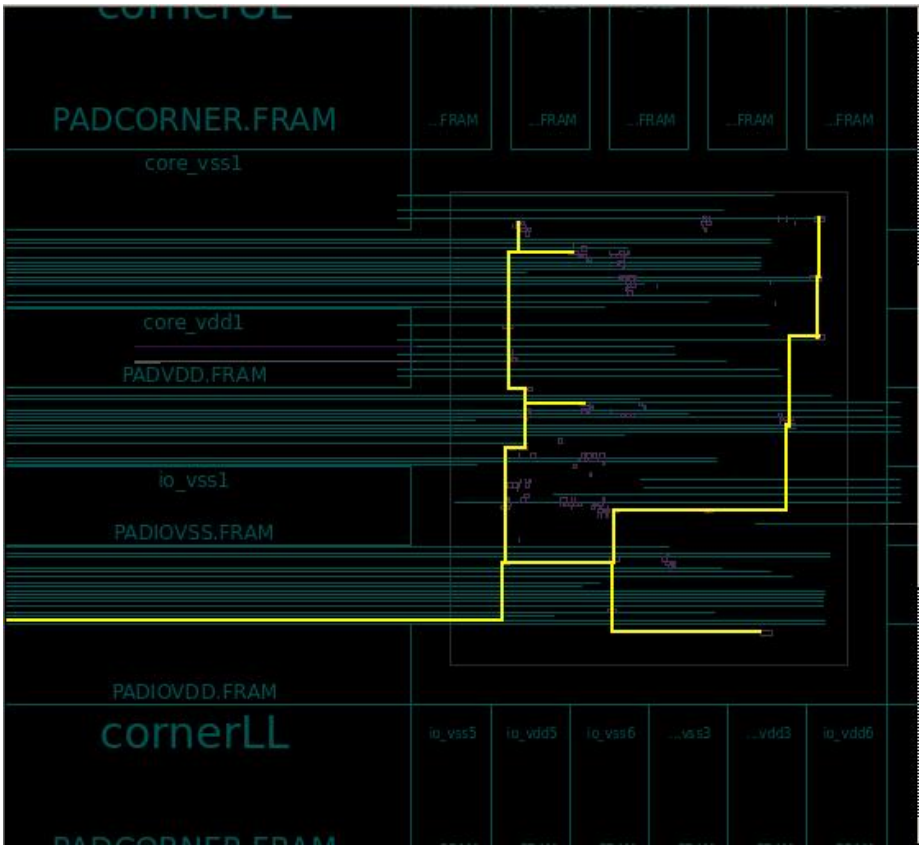


Hình 4.24. *Kết quả nối trước tín hiệu nguồn/đất.*

Công việc tiếp theo là tổng hợp cây clock (Clock Tree Synthesis). Trong công đoạn này, các tín hiệu clock sẽ được nối, đồng thời các điều kiện timing sẽ được kiểm tra lại (dựa trên file sdc có từ quá trình Synthesis hay STA). Để

thực hiện tổng hợp cây clock, ta chọn **Clock→Core CTS and Optimization**..., Trong mục Clock tree names, ta chọn tín hiệu clk như trong thiết kế (hình 4.23) và nhấn chọn **Add» và nhấn OK**. Sau đó ta nhấn chọn các box Fix hold timing only after CTS và Fix hold time violation for all clocks nhằm sửa tất cả các lỗi về hold time (hình 4.24). Kết quả tổng hợp clock được thể hiện trong hình 4.25.

Sau khi thực hiện tổng hợp cây clock, bước Routing và xuất file GDSII được thực hiện giống như lab trước.



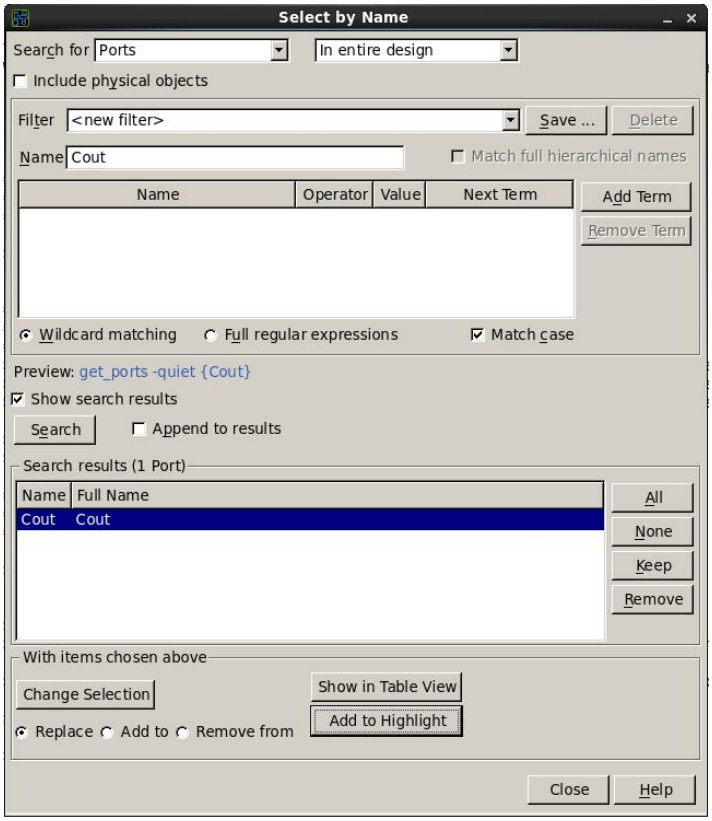
Hình 4.25. *Kết quả chạy tổng hợp cây clock.*

Công đoạn cuối cùng trong Place & Route là Routing, hay nói cách khác là nối tất cả các dây còn lại:

Trước khi tiến hành Routing, ta cần kiểm tra khả năng cho phép đi dây thông qua chọn **Route→Check Routability...**, sau đó chọn **OK** và xem kết quả kiểm tra trong Terminal

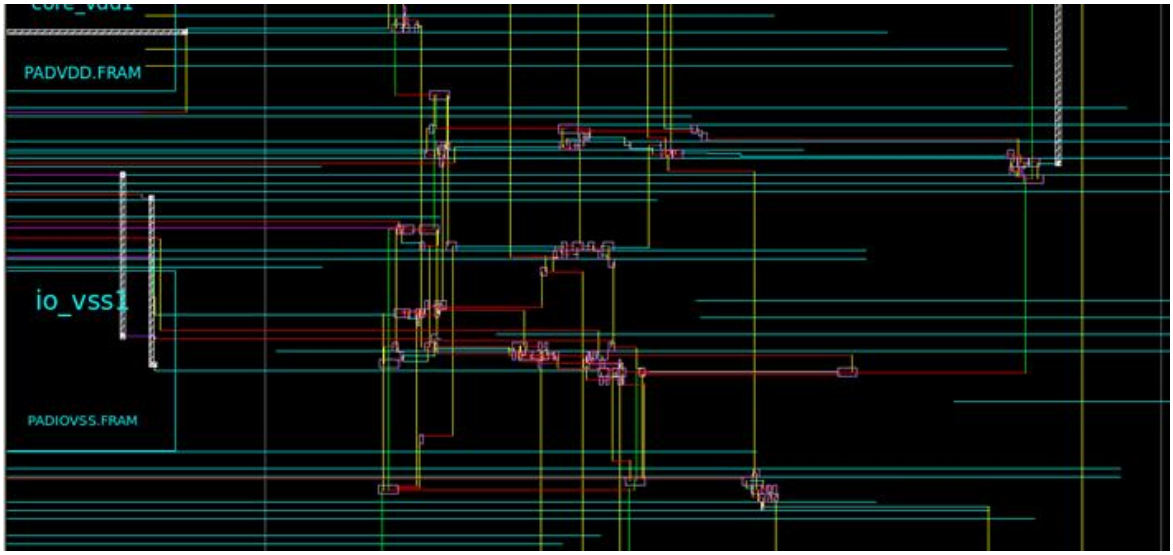
Trong trường hợp port hay cell bị blocked (tức bị cản trở), ta cần thay đổi vị trí của chúng. Để thuận tiện cho việc tìm **Port hay Cell bị lỗi**, chọn **Select→By Name**..., sau đó chọn loại tìm kiếm, tên→nhấn Search→chọn tín hiệu cần quan sát→nhấn **Add to Highlight** (hình 4.26) Cell hay Port bị lỗi sẽ được thể hiện sáng lên giúp ta dễ dàng tìm kiếm được chúng trên Floor. Để thay đổi vị trí của chúng, nhấn chuột phải lên LayoutWindow, chọn Window Stretch, sau đó kéo thả sao cho bao phủ Cell hay Port cần di chuyển, sau đó kéo

thả chính Cell hay Port đó đến vị trí thích hợp.



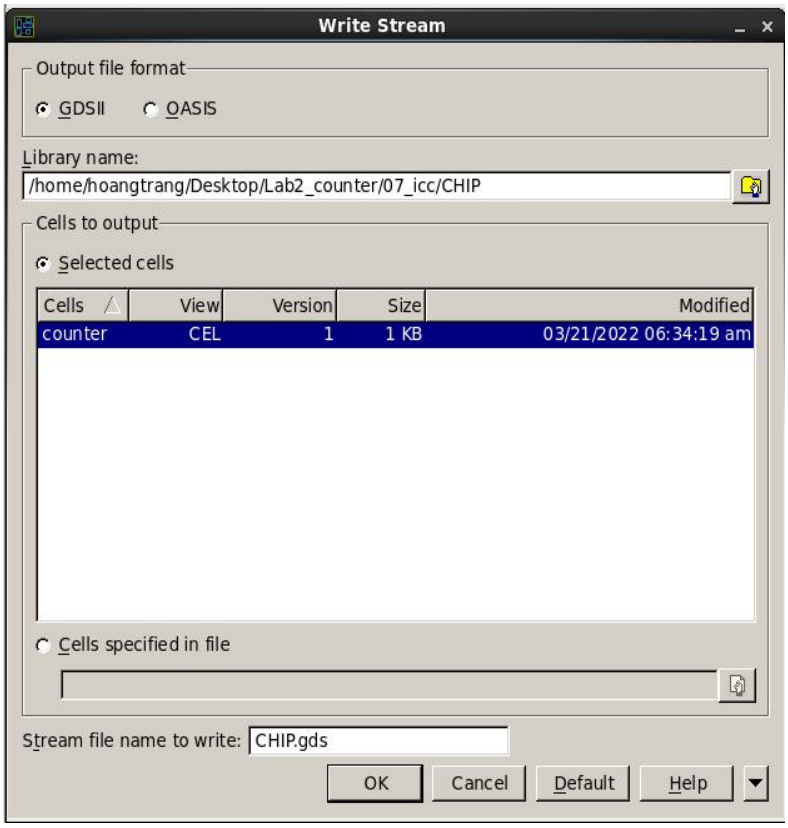
Hình 4.26. *Highlight port/cell bị lỗi để chỉnh sửa.*

Nếu kiểm tra Routing cho kết quả khả thi, nhấn chọn **Route→Core Routing and Optimization**.



Hình 4.27. *Kết quả chạy quá trình Routing.*

Để gửi thông tin về vi mạch đến nhà máy sản xuất, ta sẽ xuất file có định dạng GDSII (đuôi .gds) bằng cách chọn **File→Export→WriteStream**, sau đó chọn thư viện **MilkyWay, Cell** và tên file output như hình sau và nhấn **OK**



Hình 4.28. *Xuất file GDSII để gửi đến nhà máy.*

4.3 Tự thực hành

Bài thực hành-thí nghiệm dành cho người đọc cuối chương này có nội dung như sau:

- Tên: Thiết kế bộ đếm lên và xuống.
- Ngõ vào:

– Một số 16 bit cnt_in là giá trị đếm khởi điểm

– Một tín hiệu 1 bit tên load là tín hiệu báo hiệu bắt đầu load giá trị khởi điểm

– Một tín hiệu clock 1 bit tên clk

– Một tín hiệu 1 bit nhằm chọn chế độ đếm tên mode, với 0 là đếm xuống và 1 là đếm lên.

- Ngõ ra: 1 số 16 bit cnt_out là số đếm ngõ ra có giá trị thay đổi theo thời gian.
- Hoạt động: Khi có xung cạnh lên của clk, bộ đếm sẽ kiểm tra giá trị của tín hiệu load, nếu bằng 1 thì cnt_out sẽ bằng giá trị của tín hiệu cnt_in±1 tùy theo

giá trị của mode, nếu bằng 0 thì cnt_out sẽ bằng giá trị cnt_out trong chu kỳ clock trước đó ±1 tùy theo giá trị của mode.

Ví dụ: Để dễ hiểu hoạt động của bộ đếm trên, quan sát hình 4.28, trong đó:

- Ở xung cạnh lên clock thứ nhất, load=1, mode=1 (đếm lên), cnt_in=25, suy ra cnt_out=25+1=26.
- Ở xung cạnh lên clock thứ hai, load vẫn bằng 1, mode=1 (đếm lên), cnt_in=25, suy ra cnt_out=25+1=26 như cũ.
- Ở xung cạnh lên clock thứ ba, load chuyển sang bằng 0, mode=1 (đếm lên), cnt_in=25 lúc này không cần quan tâm do hết phải load, suy ra cnt_out=26+1=27 (lấy giá trị cnt_out cũ bằng 26).
- Ở xung cạnh lên clock thứ tư, load=0, mode=1 (đếm lên), cnt_in=25 lúc này không cần quan tâm do hết phải load, suy ra cnt_out=27+1=28 (lấy giá trị cnt_out cũ bằng 27).