

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**  
**KHOA ĐIỆN-ĐIÊN TỬ**  
**BỘ MÔN ĐIỆN TỬ**

-----o0o-----



**THIẾT KẾ VI MẠCH**  
**BÁO CÁO TKVM SỐ LAB 2**  
**THIẾT KẾ BỘ ĐẾM**

GVHD: PGS.TS Hoàng Trang

TGHD: Đỗ Quang Thịnh

NTH : Lớp L05 – Nhóm 6

DANH SÁCH THÀNH VIÊN:

STT	Họ và tên	MSSV	Email
1	Nguyễn Phan Vĩnh Khang (0915 282 427)	1910242	khang.nguyennpvk@hcmut.edu.vn
2	Nguyễn Quang Minh	1911612	minh.nguyen711@hcmut.edu.vn
3	Phạm Hồng Thái	1915119	thai.phamhongthai10@hcmut.edu.vn
4	Nguyễn Văn Thăng	1910543	thang.nguyen28@hcmut.edu.vn
5	Huỳnh Phú Cường	1912821	cuong.huynh922035@hcmut.edu.vn

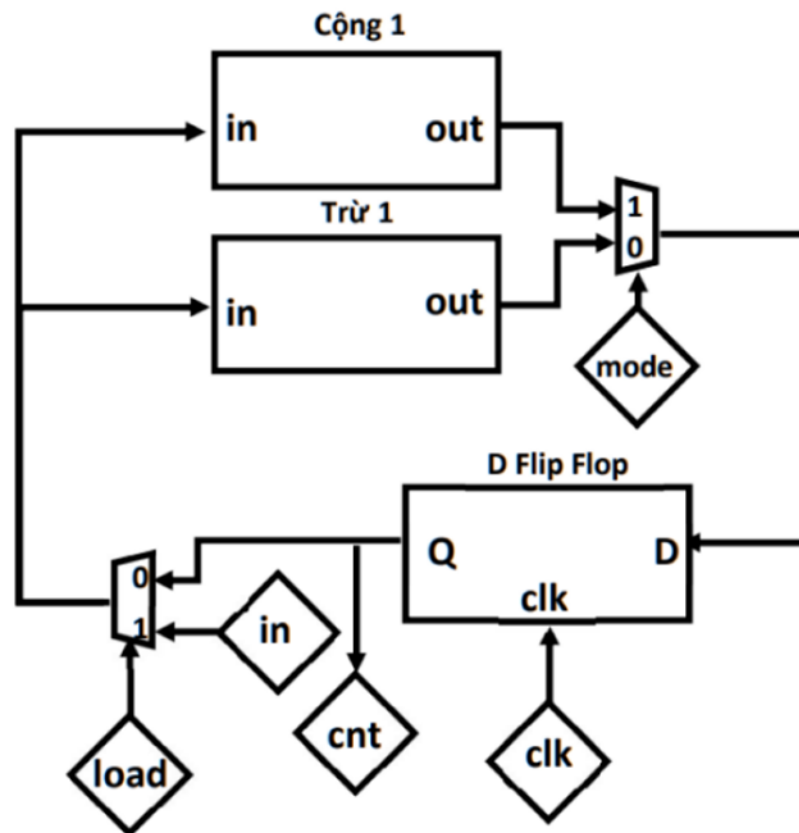
## MỤC LỤC

<b>1. Tổng Quan Lý Thuyết.</b>	4
<b>1.1. Thiết kế cấp độ Specification</b>	4
a) Sơ đồ khối bộ đếm lên/xuống	4
b) Cách thức hoạt động của bộ đếm lên/xuống	4
c) Ý tưởng thiết kế.	5
<b>2. Thực hiện thiết kế</b>	5
<b>2.1. Mạch thiết kế cấp cổng dùng Verilog</b>	5
a) Khối bộ đếm tổng thể	5
b) Các khối tính toán	6
c) Khối D Flip Flop	7
<b>2.2. Kiểm định RTL</b>	8
a) File testtop.v	8
b) Kết quả hiển thị ở terminal sau khi make	9
c) Dạng sóng của thiết kế	10
<b>2.3. Synthesis (Tổng hợp).</b>	11
a) File counter sau khi Synthesis	11
b) File report.timing.	12
<b>2.4. Kiểm định netlist.</b>	14
a) Màn hình thông báo của Formality sau khi chạy Matching	14
b) Màn hình thông báo của Formality sau khi chạy Verify.	15
<b>3. Kết luận</b>	15
<b>3.1. Nhận xét</b>	15
<b>3.2. Kết luận</b>	15
<b>4. Phụ lục</b>	16
<b>5. Tài liệu tham khảo (nếu có).</b>	16
<b>6. Trả lời một số câu hỏi lý thuyết.</b>	16

## 1. Tổng Quan Lý Thuyết.

### 1.1. Thiết kế cấp độ Specification

a) Sơ đồ khối bộ đếm lên/xuống



Hình 1-1: Sơ đồ khối bộ đếm lên/xuống.

b) Cách thức hoạt động của bộ đếm lên/xuống

- Khối input:

- + in: dữ liệu đầu vào 16 bit.
- + load, mode: tín hiệu điều khiển.
- + clk : tín hiệu xung clk.

- Khối xử lý:

- + Khối cộng 1 bit.

+ Khởi trừ 1 bit.

+ Khởi D Flip Flop.

- Khởi output:

+ load =1, mode =1:  $cnt = in + 1$ .

+ load =1, mode =0:  $cnt = in - 1$ .

+ load =0, mode =1:  $cnt = cnt + 1$ .

+ load =0, mode =0:  $cnt = cnt - 1$ .

c) Ý tưởng thiết kế.

- Hình thành sơ đồ thiết kế.

- Xây dựng các khối xử lý add\_1, sub\_1, D Flip Flop.

- Xây dựng bộ đếm lên/xuống dựa vào input, chọn dữ liệu nạp vào khối từ tín hiệu điều khiển load và chọn output từ tín hiệu điều khiển mode.

## 2. Thực hiện thiết kế

### 2.1. Mạch thiết kế cấp cổng dùng Verilog

a) Khối bộ đếm tổng thể

- input:

+ cnt\_in : 16 bit.

+ clk, load, mode.

- output:

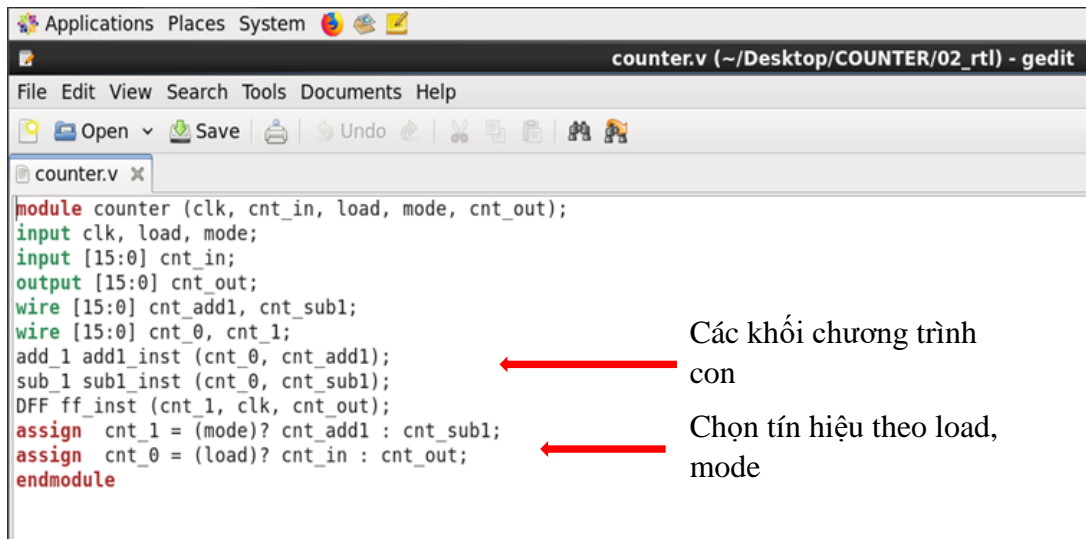
+ cnt\_out: 16 bit.

- biến tạm:

+ cnt\_add, cnt\_sub : cho giá trị ra 2 khối cộng 1 và trừ 1.

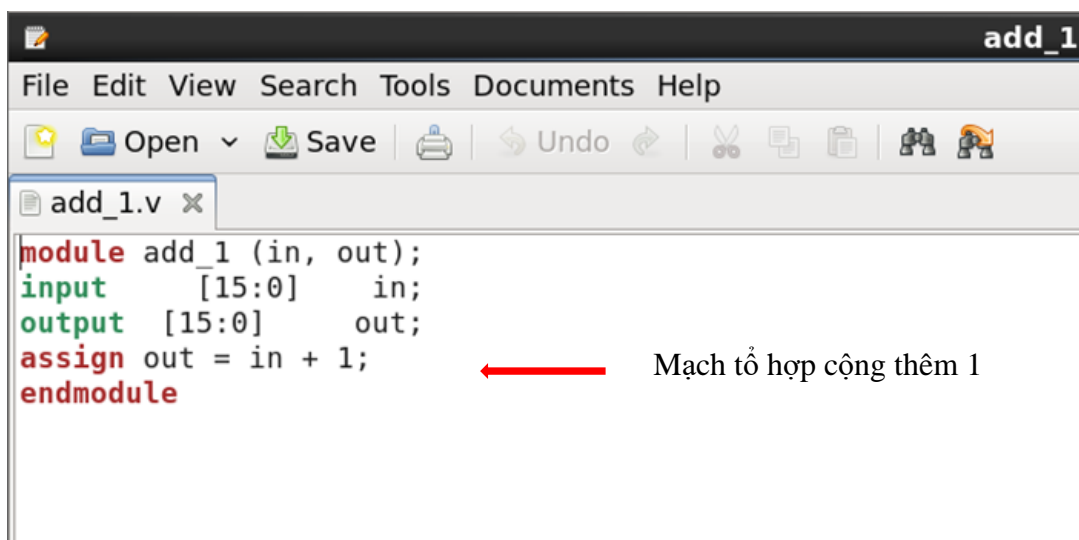
+ cnt\_0 : cho giá trị vào của khối cộng 1 và trừ 1.

+ cnt\_1 : cho giá trị vào của khối D Flip Flop.

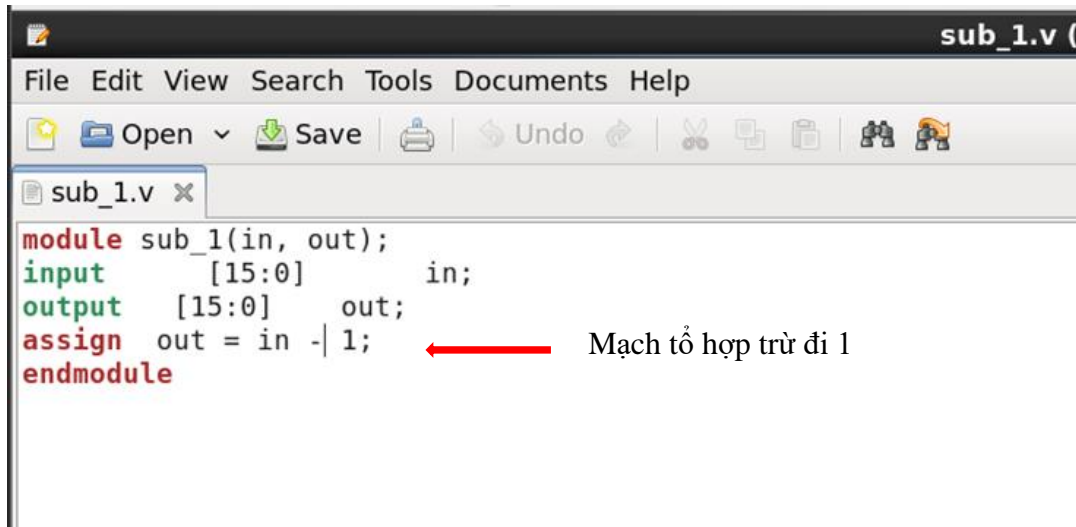


Hình 2-1 : Khối counter

b) Các khối tính toán.

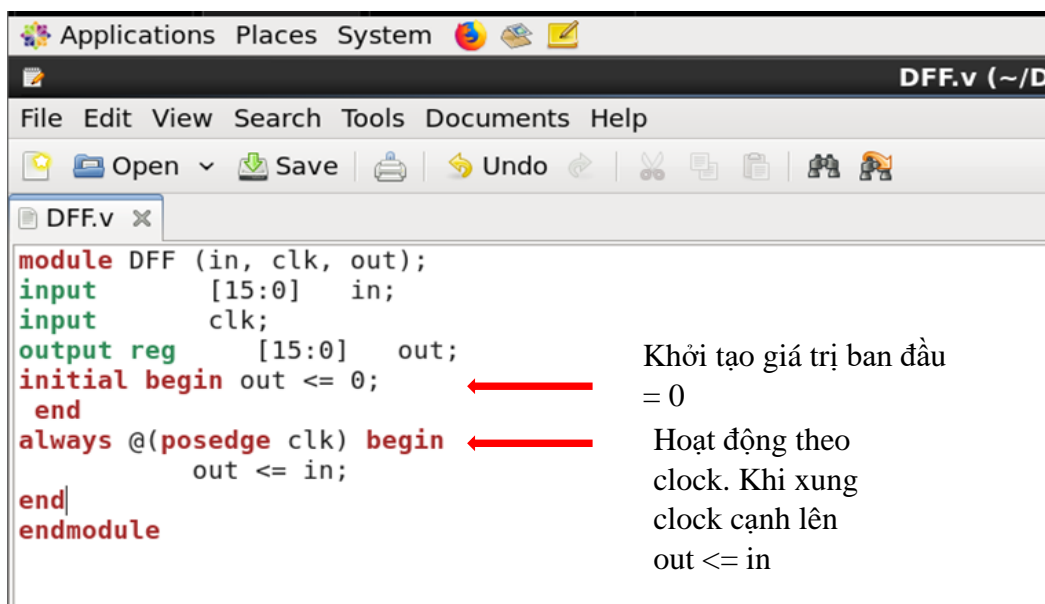


Hình 2-2 : Khối cộng 1.



Hình 2-3 : Khởi trừ 1.

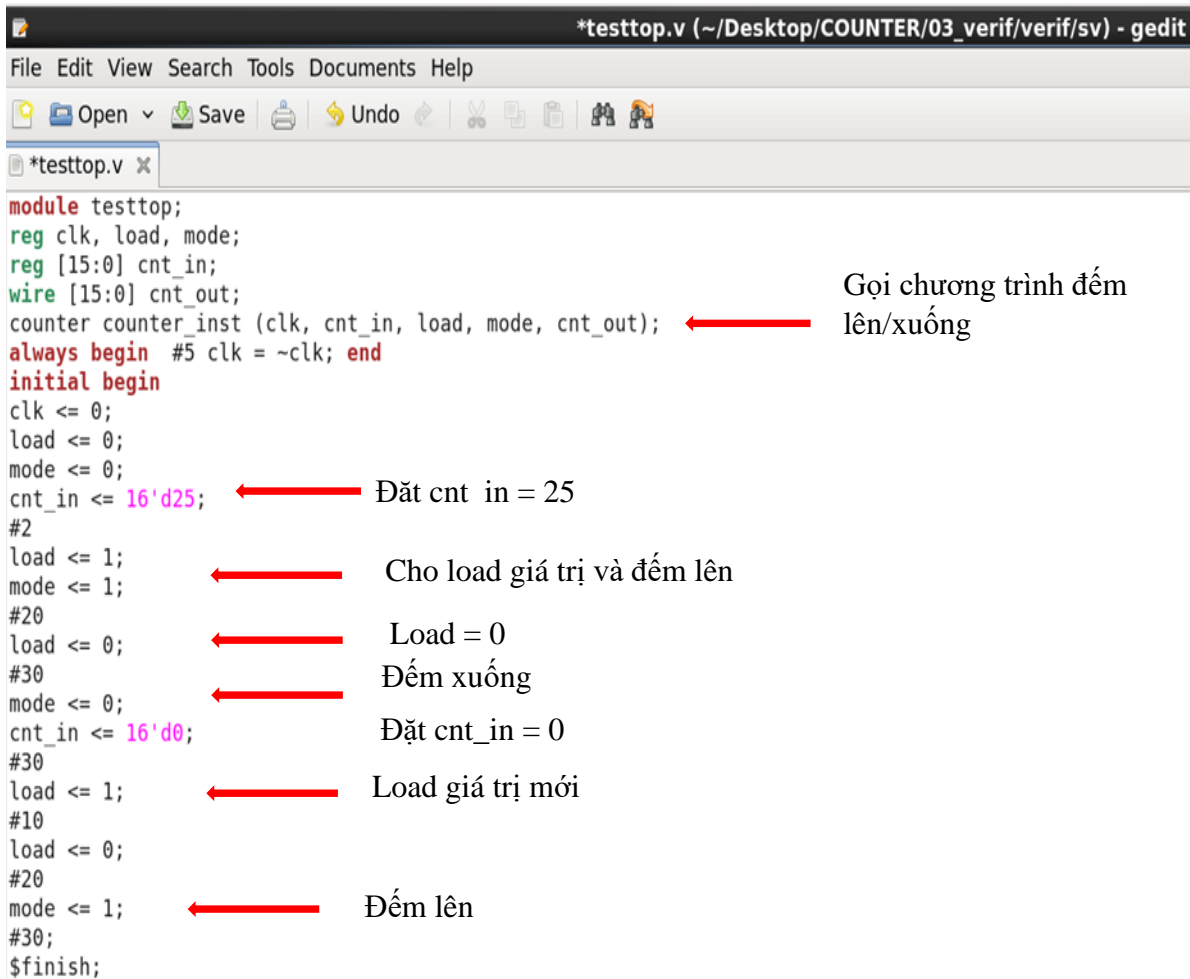
c) Khối D Flip Flop



Hình 2-4 : Khối D Flip Flop.

## 2.2. Kiểm định RTL

### a) File testtop.v



```
*testtop.v (~/Desktop/COUNTER/03_verif/verif/sv) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
*testtop.v
module testtop;
reg clk, load, mode;
reg [15:0] cnt_in;
wire [15:0] cnt_out;
counter counter_inst (clk, cnt_in, load, mode, cnt_out);
always begin #5 clk = ~clk; end
initial begin
clk <= 0;
load <= 0;
mode <= 0;
cnt_in <= 16'd25;
#2
load <= 1;
mode <= 1;
#20
load <= 0;
#30
mode <= 0;
cnt_in <= 16'd0;
#30
load <= 1;
#10
load <= 0;
#20
mode <= 1;
#30;
$finish;
endmodule
```

Gọi chương trình đếm  
lên/xuống

Đặt cnt\_in = 25

Cho load giá trị và đếm lên

Load = 0

Đếm xuống

Đặt cnt\_in = 0

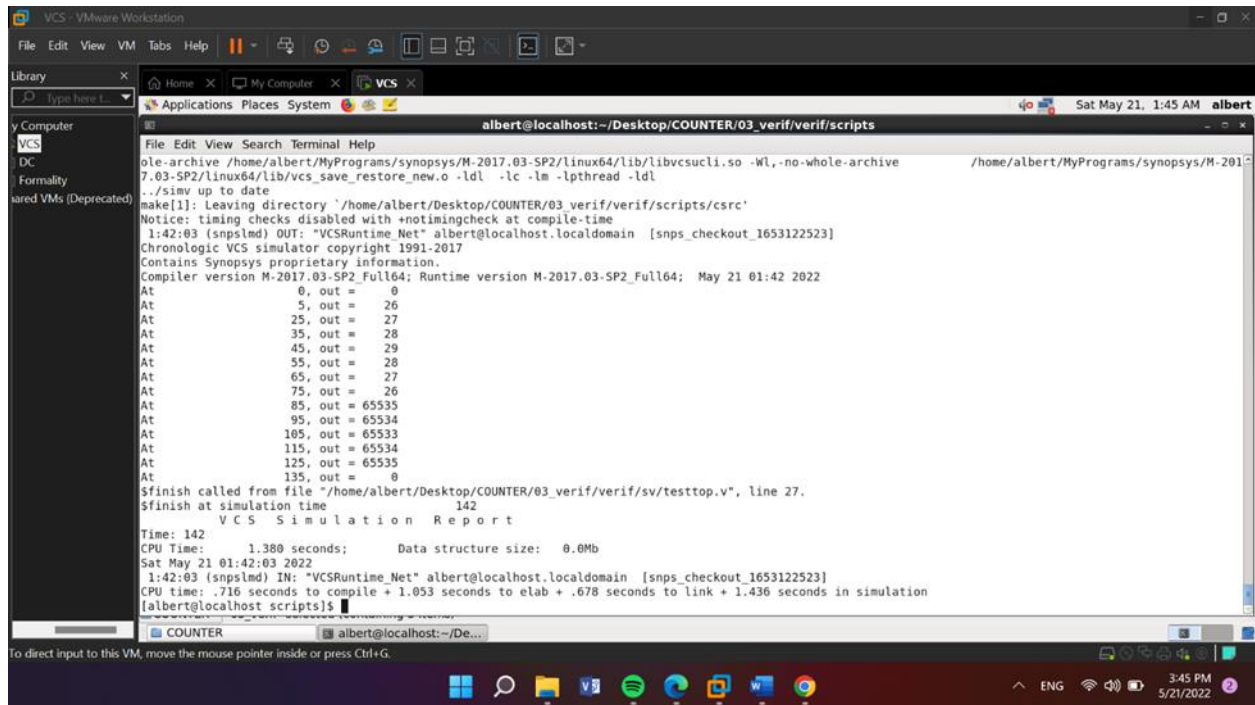
Load giá trị mới

Đếm lên

Hình 2-5: File testtop.v



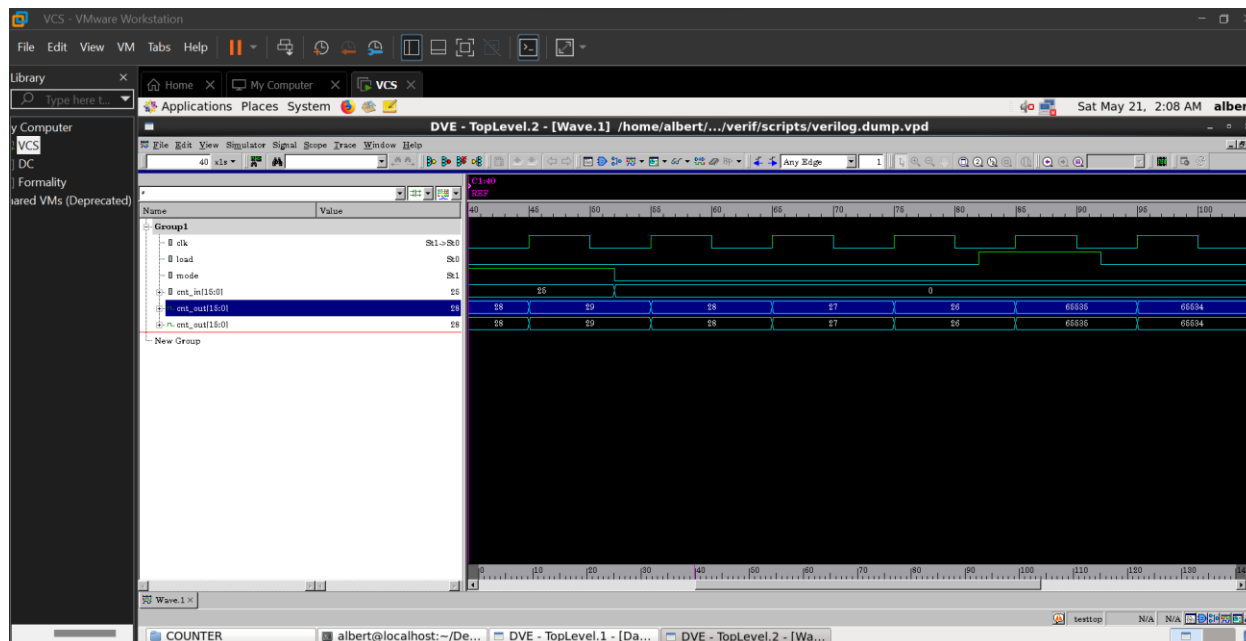
b) Kết quả hiển thị ở terminal sau khi make



```
File Edit View Search Terminal Help
ole-archive /home/albert/MyPrograms/synopsys/M-2017.03-SP2/linux64/lib/libvcsucli.so -Wl,-no-whole-archive /home/albert/MyPrograms/synopsys/M-201
7.03-SP2/linux64/lib/vcs_save_restore_new.o -ldl -lc -lm -lpthread -ldl
../simv up to date
make[1]: Leaving directory `/home/albert/Desktop/COUNTER/03_verif/verif/scripts/csrc'
Notice: timing checks disabled with +notimingcheck at compile-time
1:42:03 (snpslmd) OUT: "VCSRuntime Net" albert@localhost.localdomain [snps_checkout_1653122523]
Chronologic VCS simulator copyright 1991-2017
Contains Synopsys proprietary information.
Compiler version M-2017.03-SP2_Full64; Runtime version M-2017.03-SP2_Full64; May 21 01:42 2022
At 0, out = 0
At 5, out = 26
At 25, out = 27
At 35, out = 28
At 45, out = 29
At 55, out = 28
At 65, out = 27
At 75, out = 26
At 85, out = 65535
At 95, out = 65534
At 105, out = 65533
At 115, out = 65534
At 125, out = 65535
At 135, out = 0
$finish called from file "/home/albert/Desktop/COUNTER/03_verif/verif/sv/testtop.v", line 27.
$finish at simulation time 142
VCS Simulation Report
Time: 142
CPU Time: 1.380 seconds; Data structure size: 0.0Mb
Sat May 21 01:42:03 2022
1:42:03 (snpslmd) IN: "VCSRuntime Net" albert@localhost.localdomain [snps_checkout_1653122523]
CPU times: .716 seconds to compile + 1.053 seconds to elab + .678 seconds to link + 1.436 seconds in simulation
[albert@localhost scripts]$
```

Hình 2-6: Kết quả ở terminal sau khi make

- |           |              |  |
|-----------|--------------|--|
| - At 0,   | out = 26;    | Đặt cnt_in = 25, load giá trị và đếm lên |
| - At 55,  | out = 28;    | Đếm xuống                                |
| - At 85,  | out = 65535; | Đặt cnt_in = 0, đếm xuống                |
| - At 115, | out = 65534; | Đếm lên                                  |



- Kết quả mô phỏng dạng sóng được hiển thị ở hình bên dưới.
- Phần bên trái là các biến của thiết kế.
- Phần bên phải là kết quả dạng sóng biểu diễn ở dạng số thập phân không dấu.

**2.3. Synthesis (Tổng hợp).**

## a) File counter sau khi Synthesis

```

lab_synthesis.v
// Created by: Synopsys Design Compiler(R)
// Version : L-2016.03-SP1
// Date : Sat May 28 08:23:25 2022
//

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

ADD UNS_OP add_4 ( .A(in), .B(1'b1), .Z(out) );
endmodule

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

SUB UNS_OP sub_4 ( .A(in), .B(1'b1), .Z(out) );
endmodule

module OFF ( in, clk, out );
input [15:0] in;
output [15:0] out;
input clk;

\\**SEQUENCE** \out_reg[15] ( .clear(1'b0), .preset(1'b0), .next_state(in[15]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[15]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[14] ( .clear(1'b0), .preset(1'b0), .next_state(in[14]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[14]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[13] ( .clear(1'b0), .preset(1'b0), .next_state(in[13]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[13]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[12] ( .clear(1'b0), .preset(1'b0), .next_state(in[12]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[12]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[11] ( .clear(1'b0), .preset(1'b0), .next_state(in[11]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[11]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[10] ( .clear(1'b0), .preset(1'b0), .next_state(in[10]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[10]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[9] ( .clear(1'b0), .preset(1'b0), .next_state(in[9]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[9]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[8] ( .clear(1'b0), .preset(1'b0), .next_state(in[8]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[8]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[7] ( .clear(1'b0), .preset(1'b0), .next_state(in[7]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[7]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[6] ( .clear(1'b0), .preset(1'b0), .next_state(in[6]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[6]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[5] ( .clear(1'b0), .preset(1'b0), .next_state(in[5]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[5]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );

```

```

lab_synthesis.v
// Created by: Synopsys Design Compiler(R)
// Version : L-2016.03-SP1
// Date : Sat May 28 08:23:25 2022
//

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

ADD UNS_OP add_4 ( .A(in), .B(1'b1), .Z(out) );
endmodule

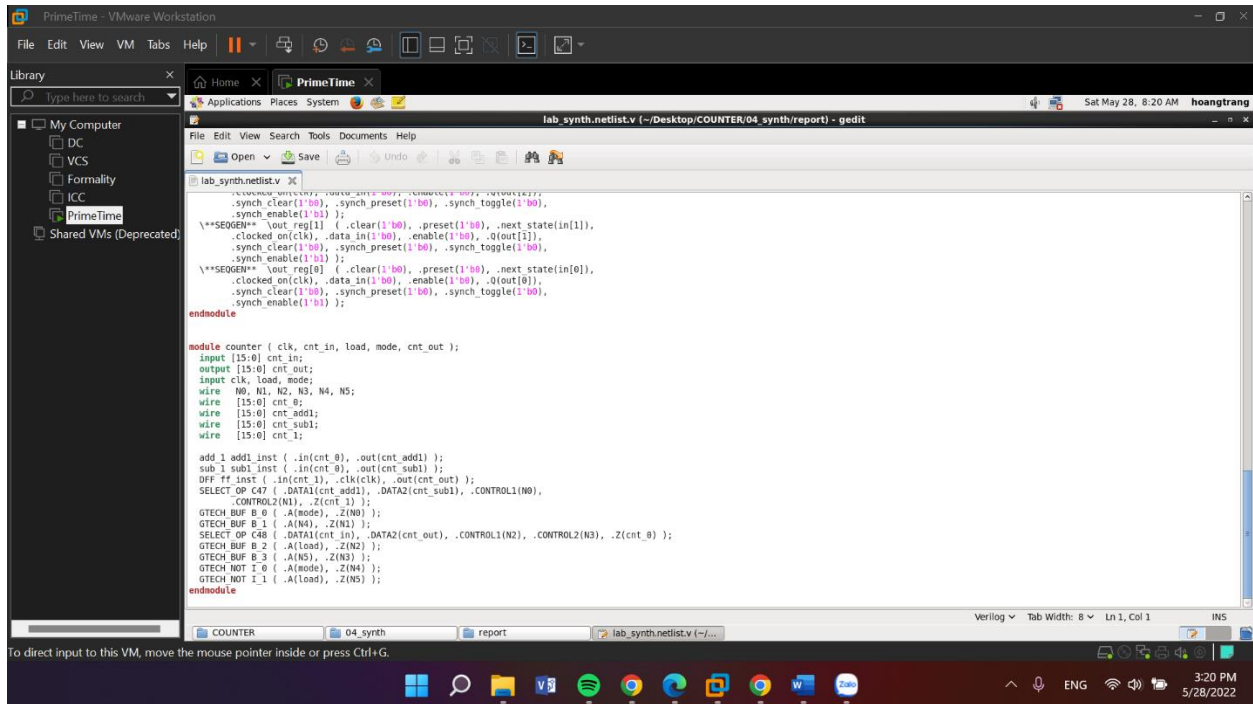
module sub ( in, out );
input [15:0] in;
output [15:0] out;

SUB UNS_OP sub_4 ( .A(in), .B(1'b1), .Z(out) );
endmodule

module OFF ( in, clk, out );
input [15:0] in;
output [15:0] out;
input clk;

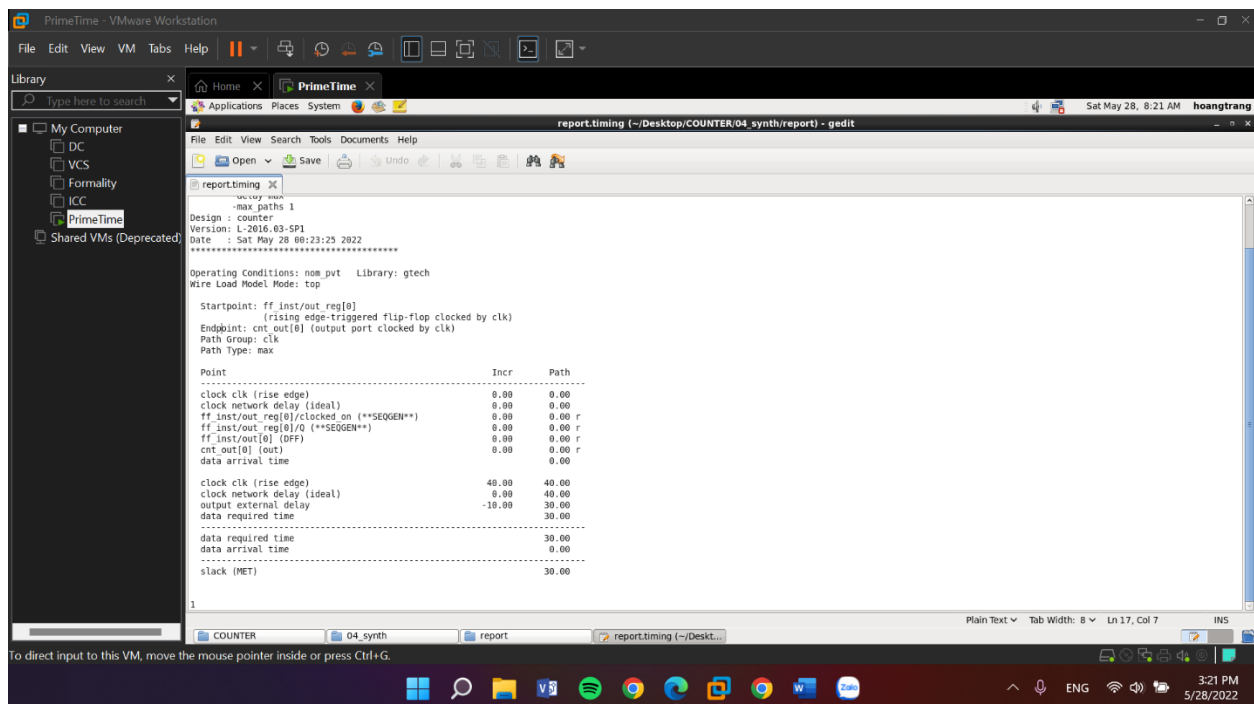
\\**SEQUENCE** \out_reg[15] ( .clear(1'b0), .preset(1'b0), .next_state(in[15]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[15]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[14] ( .clear(1'b0), .preset(1'b0), .next_state(in[14]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[14]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[13] ( .clear(1'b0), .preset(1'b0), .next_state(in[13]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[13]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[12] ( .clear(1'b0), .preset(1'b0), .next_state(in[12]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[12]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[11] ( .clear(1'b0), .preset(1'b0), .next_state(in[11]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[11]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[10] ( .clear(1'b0), .preset(1'b0), .next_state(in[10]), .clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[10]), .synch_clear(
1'b0), .synch_preset(1'b0), .synch_toggle(1'b0), .synch_enable(1'b1)
);
\\**SEQUENCE** \out_reg[9] ( .clear(1'b0), .preset(1'b0), .next_state(in[9]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[9]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[8] ( .clear(1'b0), .preset(1'b0), .next_state(in[8]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[8]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[7] ( .clear(1'b0), .preset(1'b0), .next_state(in[7]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[7]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[6] ( .clear(1'b0), .preset(1'b0), .next_state(in[6]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[6]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );
\\**SEQUENCE** \out_reg[5] ( .clear(1'b0), .preset(1'b0), .next_state(in[5]),
.clocked_on(clk), .data_in(1'b0), .enable(1'b0), .Q(out[5]),
.synch_clear(1'b0), .synch_preset(1'b0), .synch_toggle(1'b0),
.synch_enable(1'b1) );

```



Hình 2-8: File lab\_synth.netlist.v

b) File report.timing.



Hình 2-9: File report.timing

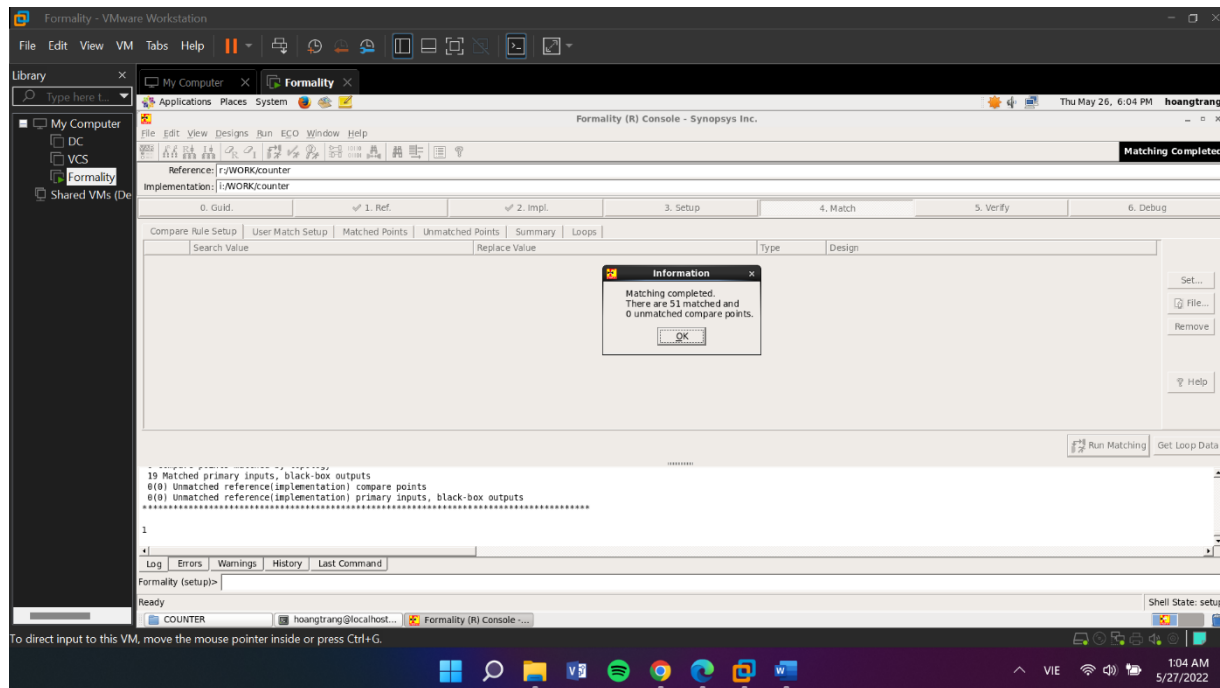
- Ở file này ta thấy có 1 phép thử timing duy nhất, có Path Type là max, có nghĩ là kiểm tra điều kiện setup time.
- Phép thử được chia làm 3 phần rõ ràng:
- 

$$T_{launch} + T_{ck2q} + T_{dp} \leq T_{capture} + T_{cycle} - T_{setup}$$

- Từ clock clk đến data arrival time là thời gian dữ liệu từ cnt\_in đến được flip flop, hay về trái.
- Từ clock clk tiếp theo đến data required time là về phải.
- Lấy hiệu hai đại lượng trên (slack) để kiểm tra điều kiện setup time. Slack đạt MET có nghĩ điều kiện setup time đã thỏa. Nếu vi phạm báo VIOLATED.

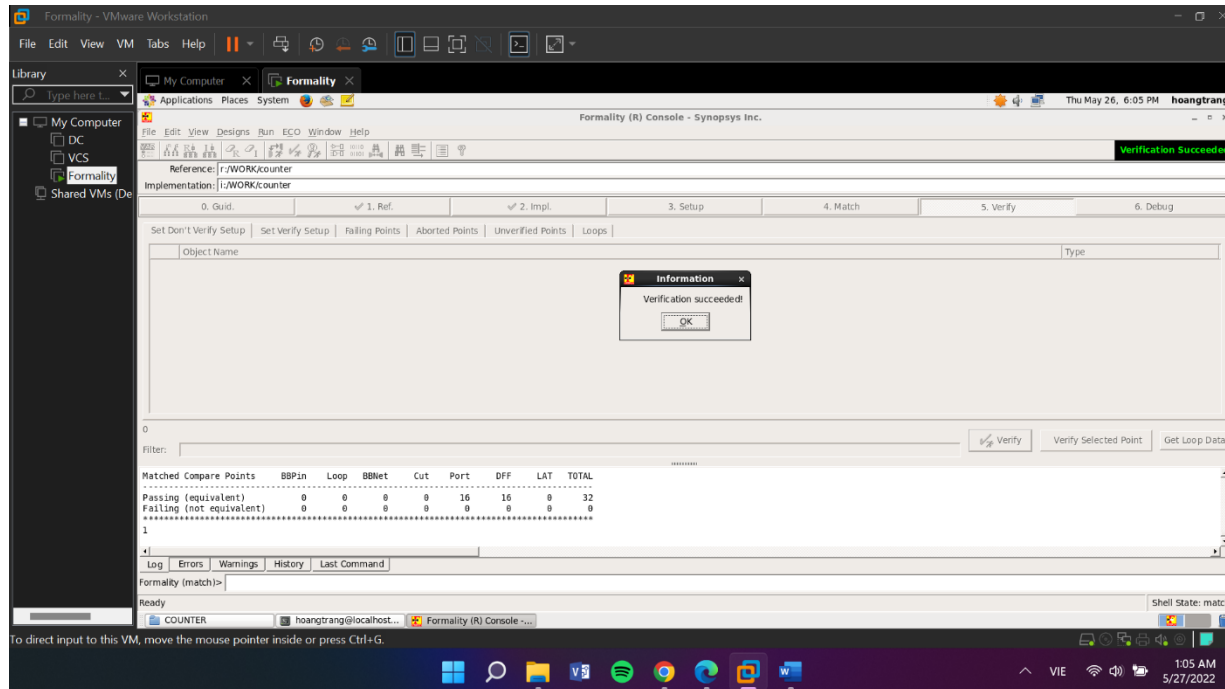
## 2.4. Kiểm định netlist.

a) Màn hình thông báo của Formality sau khi chạy Matching.



Hình 2-10: Màn hình sau khi chạy Matching.

b) Màn hình thông báo của Formality sau khi chạy Verify.



Hình 2-11: Màn hình sau khi chạy Verify

### 3. Kết luận

#### 3.1. Nhận xét

- Mạch thiết kế bộ đếm lên/ xuống là mạch cơ bản sử dụng D Flip Flop, các chức năng đếm lên xuống cơ bản, các khối tính toán đơn giản.
- Việc xây dựng khối dựa trên các khối cơ bản và các bộ mux.
- Sinh viên chưa làm quen với Linux sẽ gặp khó khăn.

#### 3.2. Kết luận.

- Nắm được các thao tác sử dụng tool trên Linux.
- Giúp khái quát được một số quá trình thiết kế vi mạch.
- Sau khi thực hiện bộ đếm lên/ xuống giúp sinh viên ôn lại kiến thức Verilog và làm quen với môi trường sử dụng tool.

#### 4. Phụ lục

#### 5. Tài liệu tham khảo (nếu có).

- Synthesis Tool Commands (DC\_Tool\_Commands) [1]
- VCS® MX/VCS MXi™ User Guide [2]
- Tài liệu hướng dẫn thí nghiệm Quy trình thiết kế vi mạch số.

#### 6. Trả lời một số câu hỏi lý thuyết.

*Cách khắc phục khi điều kiện setup time bị vi phạm:*

Giảm tần số, tức tăng Tcycle, tăng Tdp.

*Cách khắc phục khi điều kiện hold time bị vi phạm:*

Kiểm tra điều kiện về hold không liên quan đến chu kỳ của clock, 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.

*So sánh mạch tổ hợp và 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.

**Đường link file chạy mô phỏng:**

<https://drive.google.com/drive/folders/1dTSaEsADl6Z1lNAcp8SqHOqthmxWd1Q0?fbclid=IwAR00zvP--3kcIJdQSS0Y50qJh-g51cxBPU3GuKWhYdjUN8LOrPoijpIrh-8>