Chương 5
Bài thực hành-thí nghiệm 3: Thiết kế máy trạng thái

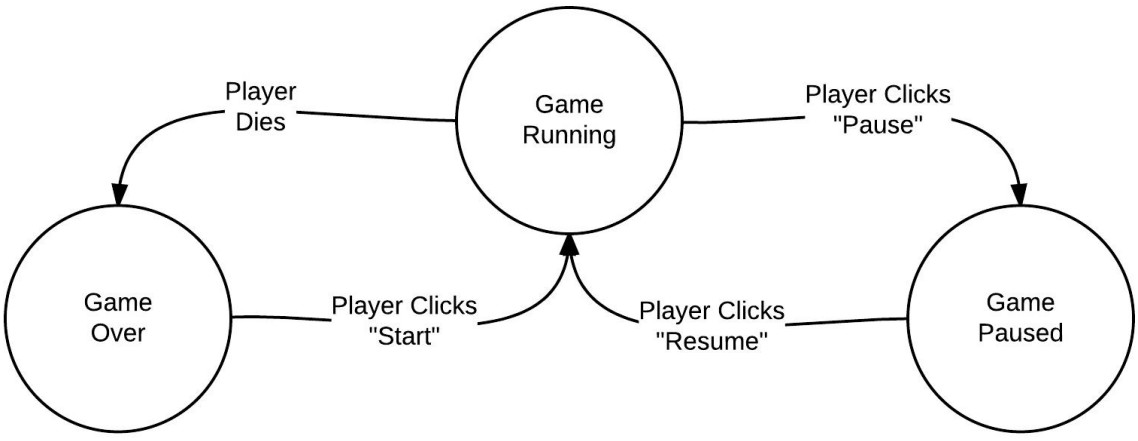
Tóm tắt về chương 5

Bài thực hành-thí nghiệm này sẽ là bài cuối cùng trong số các bài thực hành thí nghiệm căn bản của tài liệu này. Do đó, trong bài thực hành-thí nghiệm 3, các bước thực hiện quy trình sẽ được nhấn mạnh, và đi từ đầu đến cuối (từ ý tưởng thiết kế đến file gửi nhà máy sản xuất). Trong các chương tiếp theo (nâng cao), một số kĩ năng, kiến thức mới sẽ được đề cập đến nhằm làm phong phú hơn kinh nghiệm về thiết kế vi mạch.

5.1. Lý thuyết

*5.*1.1. Máy trạng thái

Trong các hệ thống lớn, thường sẽ bao gồm các khối điều khiển với nhiệm vụ kiểm soát các khối khác, đảm bảo vận hành giữa các khối này ăn khớp với nhau. Các khối điều khiển này thường hoạt động trong nhiều trạng thái khác nhau, tùy thuộc vào từng thời điểm. Khái niệm máy trạng thái dùng để mô tả cách vận hành, luân chuyển giữa các trạng thái mà khối điều khiển hoạt động. Như minh họa ở *hình 5.1.* (gọi là sơ đồ máy trạng thái), cần có các điều kiện nhất định để máy chuyển trạng thái, và một trạng thái cũng không nhất thiết phải chuyển sang mọi trạng thái có thể. Sơ đồ máy trạng thái cho ta một cái nhìn trực quan, do đó thường được người thiết kế dùng để tạo dựng máy trạng thái họ mong muốn, cũng như giúp người khác dễ hiểu hơn khi tìm hiểu máy trạng thái trên. Sơ đồ máy trạng thái thường có dạng vòng kín, nhằm giữ máy có khả năng duy trì hoạt động; ngược lại, nếu là vòng hở, tức khi mà có một trạng thái không chuyển đổi được, máy có khả năng bị "treo" và nằm "chết cứng" tại trạng thái trên.



Hình 5.1. *Ví dụ về máy trạng thái.*

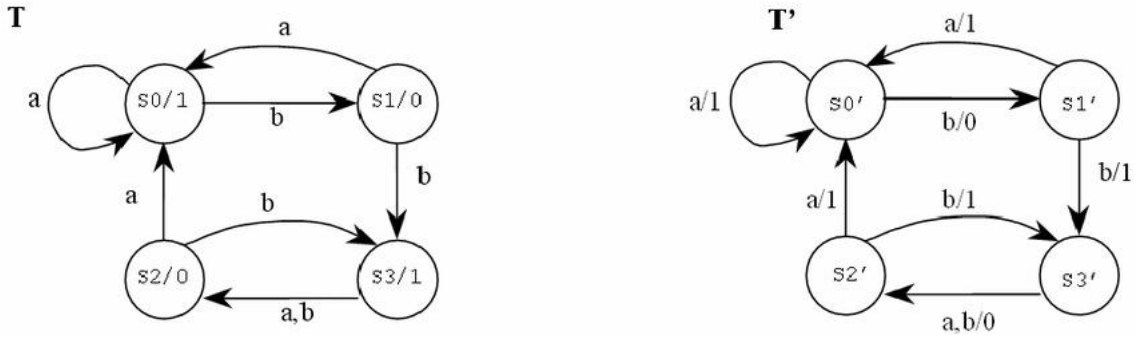
Bảng 5.1. *So sánh máy trạng thái Mealy và Moore.*

Mealy	Moore
Ngõ ra phụ thuộc vào trạng thái và ngõ vào hiện tại	Ngõ ra chỉ phụ thuộc vào trạng thái hiện tại
Nếu ngõ vào thay đổi giữa hai cạnh lên clock, ngõ ra cũng thay đổi	Nếu ngõ vào thay đổi giữa hai cạnh lên clock, ngõ ra không thay đổi
Cần ít trạng thái hơn	Cần nhiều trạng thái hơn
Yêu cầu phần cứng khó khăn hơn	Yêu cầu phần cứng ít khó khăn hơn
Phản ứng nhanh với ngõ vào	Phản ứng chậm với ngõ vào (1 chu kì sau)
Ngõ ra bất đồng bộ	Ngõ ra đồng bộ

5.1.2. Máy trạng thái Mealy và Moore

Có hai loại máy trạng thái mà người ta thường sử dụng: Mealy và Moore. Mỗi loại này có ưu/nhược điểm riêng của chúng, và người thiết kế có thể tự do lựa chọn loại mong muốn khi thiết kế máy trạng thái. Điểm khác biệt cơ bản và lớn nhất của chúng nằm ở ngõ ra của máy trạng thái. Đối với máy Mealy, ngõ ra của máy trạng thái phụ thuộc vào trạng thái hiện tại của máy, cũng như ngõ vào khi đó. Ngược lại, ở máy Moore, ngõ ra của máy chỉ phụ thuộc vào trạng thái hiện tại của máy mà thôi. Những ưu/nhược điểm giữa hai loại máy này được thể hiện trong *bảng 5.1*.

Sơ đồ máy trạng thái của hai loại máy trên cũng khác nhau (*hình 5.2.*). Đối với máy Moore, ngõ ra đi liền với trạng thái nên được thể hiện bên trong các ô tròn trạng thái; còn ở máy Mealy, do ngõ ra phụ thuộc vào cả ngõ vào nên chúng được thể hiện trên các đường mũi tên cùng với ngõ vào.



Hình 5.2. *Sơ đồ máy trạng thái của Mealy và Moore.*

5.1.3. Thiết kế máy bán nước ngọt tự động

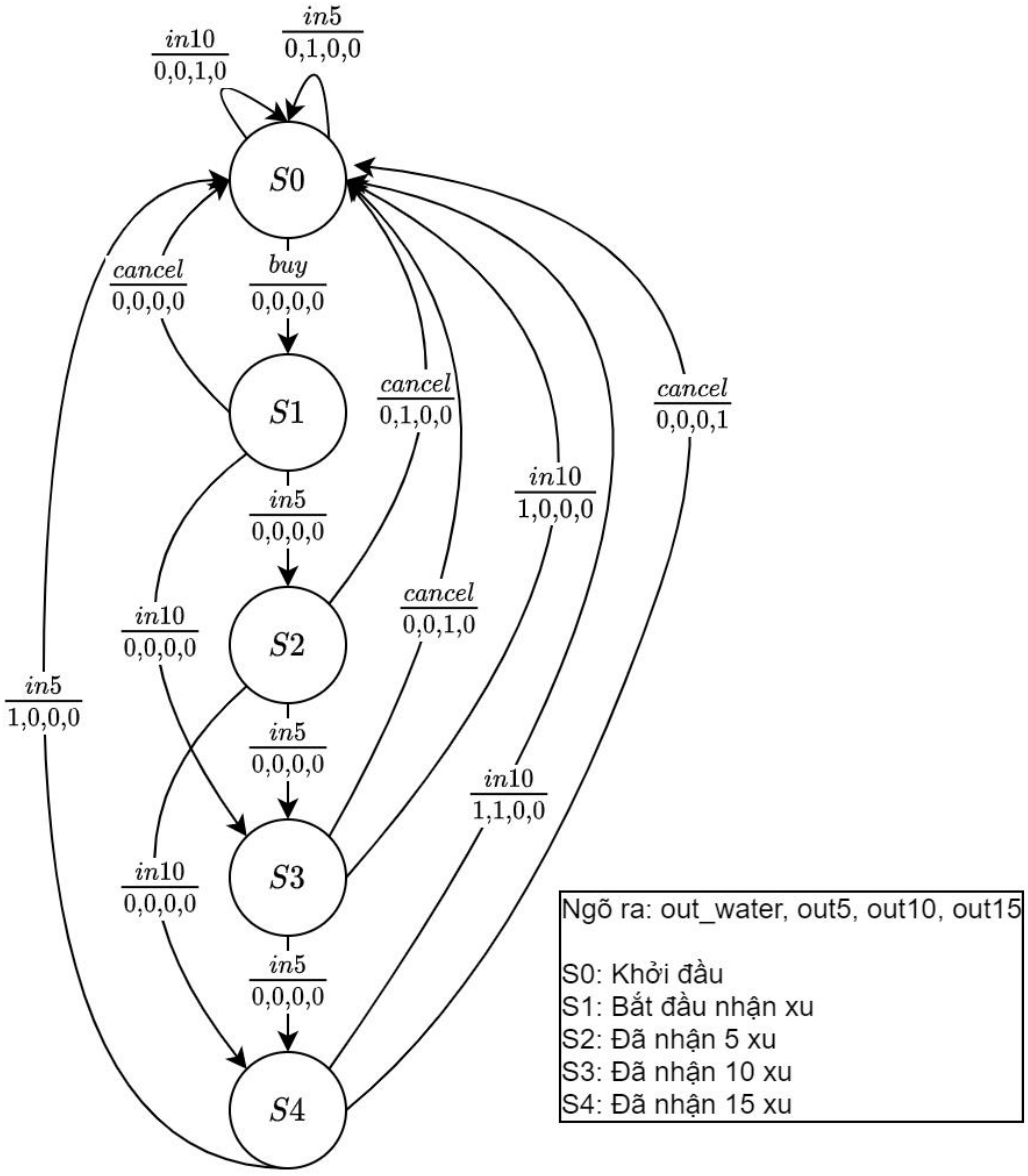
Bài thực hành-thí nghiệm 3 sẽ trình bày thiết kế một máy trạng thái đơn giản dành cho các máy bán nước tự động thường thấy trên đường phố. Để máy hoạt động, nó phải luân chuyển giữa các trạng thái khác nhau như chờ, chọn sản phẩm, đã nhận tiền, trả nước,... Trong bài thực hành-thí nghiệm này, máy trạng thái sẽ được thực hiện cho bài toán như sau:

> Máy bán một loại nước ngọt có giá 20 xu. Máy nhận hai loại tiền có mệnh giá lần lượt là 5 và 10 xu. Trước khi máy nhận xu, khách hàng phải nhấn nút Mua, nếu không thì trả lại xu nếu có xu được cho vào máy. Sau khi khách hàng nhét xu vào mà đổi ý không muốn mua nữa thì có thể nhấn nút Hủy, số tiền sẽ được trả lại. Nếu số tiền khách hàng cho vào máy nhiều hơn giá nước, trả nước cùng số tiền thối tương ứng.

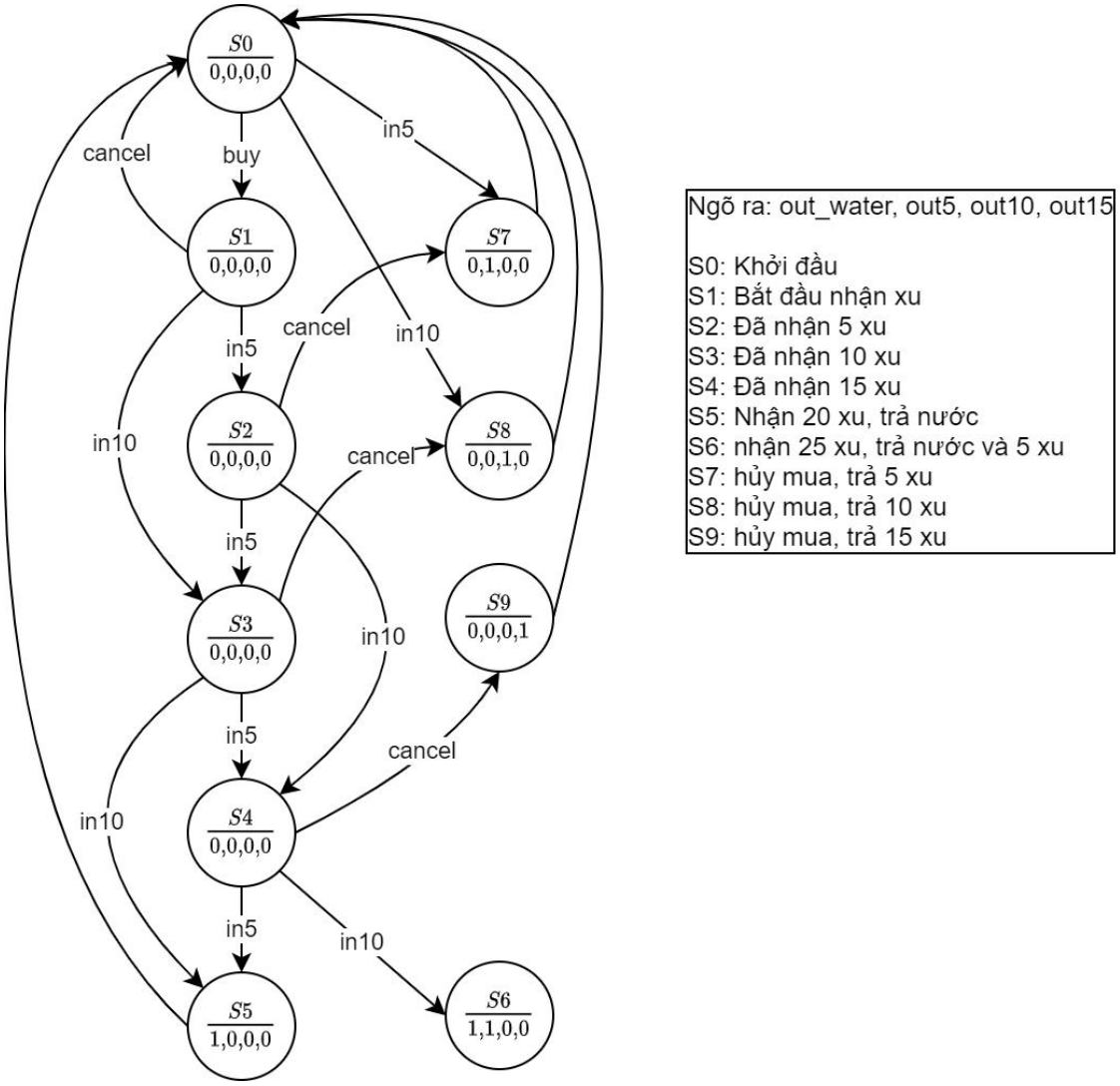
Trước hết, ta cần làm rõ ngõ ra và ngõ vào của máy:

- Ngõ vào
  - Tín hiệu Mua, 1 bit, tên là **buy**.
  - Tín hiệu Hủy, 1 bit, tên là **cancel**.
  - Tín hiệu nhận 5 xu, 1 bit, tên là **in5**.
  - Tín hiệu nhận 10 xu, 1 bit, tên là **in10**.
  - Tín hiệu clock, 1 bit, tên là **clk**.
  - Tín hiệu reset giá trị dương, 1 bit, tên là **rst**.
- Ngõ ra
  - Tín hiệu trả nước, 1 bit, tên là **out_water**.
  - Tín hiệu trả 5 xu, 1 bit, tên là **out5**.
  - Tín hiệu trả 10 xu, 1 bit, tên là **out10**.
  - Tín hiệu trả 15 xu, 1 bit, tên là **out15**.

Để vẽ sơ đồ máy trạng thái cho máy trên, ta cần nắm được số trạng thái cần dùng và các liên kết chuyển đổi giữa chúng. Để bắt đầu, ta sẽ chọn một trạng thái khởi đầu làm gốc, thường là trạng thái nghỉ (hay trạng thái khởi động), sau đó nhận xét ảnh hưởng của ngõ vào để xác định và định nghĩa các trạng thái tiếp theo. Trong bài toán ở trên, ta chọn trạng thái chờ khách hàng nhấn nút Mua là trạng thái khởi đầu, kí hiệu là **S0**. Sau khi khách hàng nhấn nút Mua, máy chuyển sang trạng thái chờ nhận xu, kí hiệu là **S1**. Tương tự như vậy, ta thu được sơ đồ máy trạng thái như trong *hình 5.3.* và *hình 5.4.*, theo Mealy và Moore. Thực tế khi thiết kế thì ta chỉ cần chọn một loại Mealy hay Moore tùy theo nhu cầu và hoàn cảnh hiện tại lúc đó. Có thể để ý thấy số trạng thái của máy Mealy trong trường hợp này ít hơn hẳn con số của máy Moore (5 so với 10).

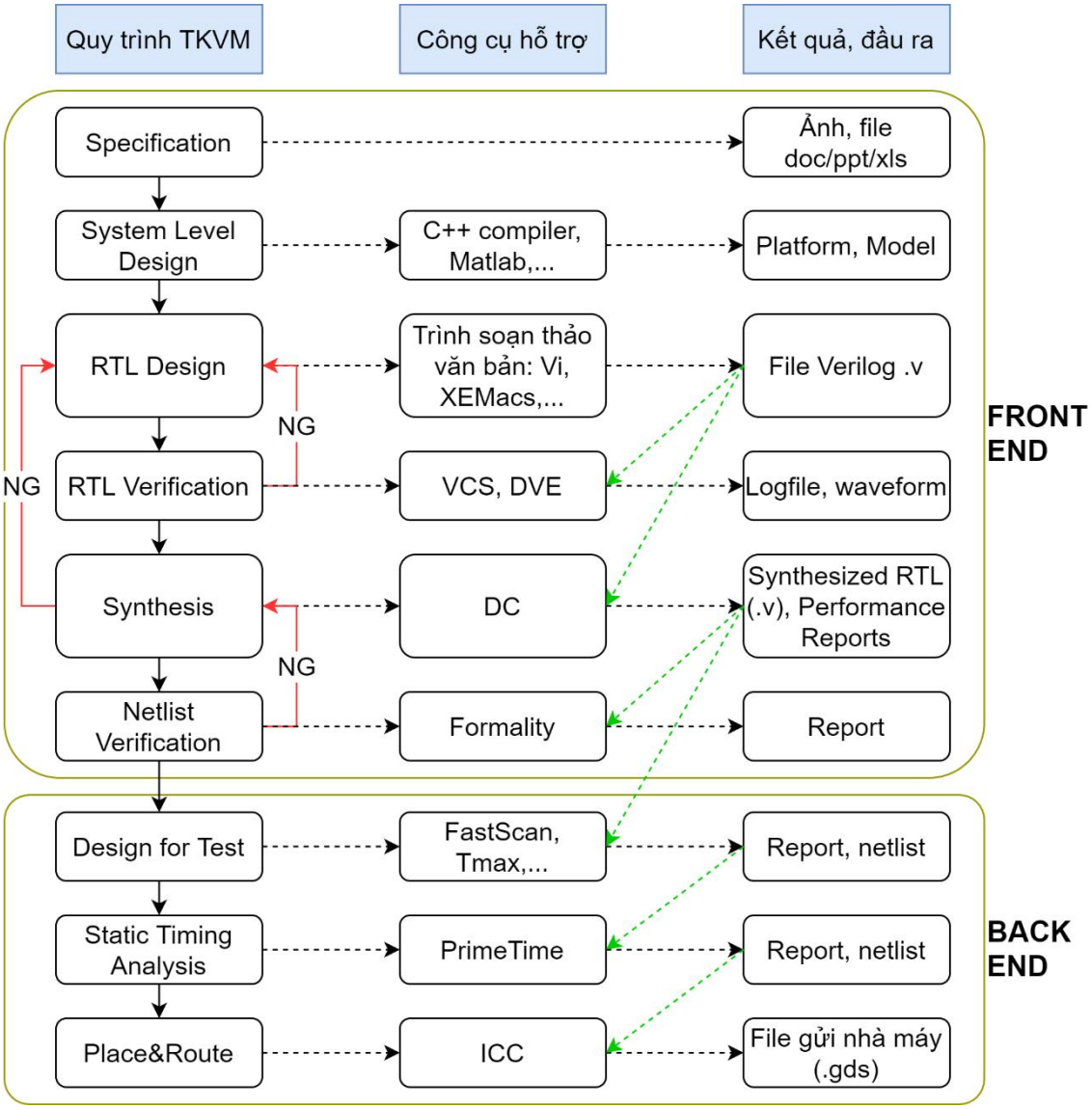


Hình 5.3. *Sơ đồ máy trạng thái của máy bán nước ngọt trong bài thực hành-thí nghiệm 3 theo Mealy.*



Hình 5.4. *Sơ đồ máy trạng thái của máy bán nước ngọt trong bài thực hành-thí nghiệm 3 theo Moore.*

5.2. Thực hiện thiết kế



Hình 5.5. *Nhắc lại các bước trong quy trình thiết kế một vi mạch.*

5.2.1. Thiết kế cấp độ hệ thống

Bảng 5.2. là mô tả khái quát của bước thiết kế cấp độ hệ thống.

Phần này ta đã mô tả trong phần lý thuyết.

5.2.2. Tạo dựng môi trường và các chuẩn bị khác

Phần này được thực hiện tương tự các bài thực hành-thí nghiệm trước.

Bảng 5.2. *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 Specification), 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.

5.2.3. Mô tả thiết kế bằng Verilog

Bảng 5.3. là mô tả khái quát của bước thiết kế cấp độ RTL.

Bảng 5.3. *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.

Thiết kế trong bài thực hành-thí nghiệm này chỉ bao gồm một file Verilog duy nhất, trong đó có mô tả máy trạng thái theo đúng sơ đồ đã vẽ ở phần trước. *Hình 5.6.* cho ta một mô tả cách lập trình máy trạng thái kiểu Moore ứng với bài toán đã cho, trong đó các biến dạng **reg** là **state** và **next_state** nhằm thể hiện trạng thái hiện tại cũng như trạng thái kế tiếp sẽ được cập nhật trong chu kỳ clock tới.

Lập trình cho các máy trạng thái sử dụng Verilog có thể được chi làm 2 phần chính:

- Phần chuẩn bị trạng thái kế tiếp: ở phần này, những thay đổi của ngõ vào hay trạng thái hiện tại sẽ ảnh hưởng đến việc chọn trạng thái kế tiếp của máy (biến **next_state**).
- Phần cập nhật trạng thái theo clock: ở phần này, trạng thái của máy sẽ thay đổi và đồng bộ theo clock. Trong trường hợp có reset, máy quay về trạng thái khởi đầu.

Vậy làm thế nào để phân biệt máy Moore và Mealy? Ta cần đọc lại về định nghĩa của chúng. Máy Mealy có ngõ ra phụ thuộc ngõ vào và trạng thái hiện tại, do đó ngõ ra sẽ được cập nhật trong phần chuẩn bị trạng thái kế tiếp, khi xét đến ảnh hưởng ngay lập tức của ngõ ra theo ngõ vào. Ngược lại, ở máy

Moore, ngõ ra chỉ phụ thuộc vào trạng thái hiện tại, vốn đã được đồng bộ theo clock, do đó ngõ ra cũng sẽ được cập nhật ở phần cập nhật trạng thái theo clock. Và cũng chính vì lẽ đó mà ngõ ra của máy Moore thường chậm hơn so với máy Mealy (ít nhất 1 chu kỳ clock).

```
module vending_machine(
        clk,
        rst,
        buy,
        cancel,
        in5,
        in10,
        out_water,
        out5,
        out10,
        out15);
input   clk;
input   rst;
input   buy;
input   cancel;
input   in5, in10;
output reg      out_water;
output reg      out5, out10, out15;

reg [3:0] state, next_state;

always @(state or buy or cancel or in5 or in10) begin
        next_state = 4'd0;
        case (state)
                4'd0: begin //start
                if (buy) next_state = 4'd1;
                else if (in5) next_state = 4'd7;
                else if (in10) next_state = 4'd8;
                else next_state = 4'd0;
                end
                4'd1: begin //bat dau nhan xu
                if (cancel) next_state = 4'd0;
                else if (in5) next_state = 4'd2;
                else if (in10) next_state = 4'd3;
                else next_state = 4'd1;
                end
                4'd2: begin //da nhan 5
                if (cancel) next_state = 4'd7;
                else if (in5) next_state = 4'd3;
                else if (in10) next_state = 4'd4;
                else next_state = 4'd2;
                end
                4'd3: begin //da nhan 10
                if (cancel) next_state = 4'd8;
                else if (in5) next_state = 4'd4;
                else if (in10) next_state = 4'd5;
                else next_state = 4'd3;
                end
                4'd4: begin //da nhan 15
                if (cancel) next_state = 4'd9;
                else if (in5) next_state = 4'd5;
                else if (in10) next_state = 4'd6;
                else next_state = 4'd4;
                end
                default: begin
                next_state = 4'd0;
                end
        endcase
end
```

```
always @(posedge clk) begin
        if (rst)
                state <= 4'd0;
        else
                state <= next_state;
        case (state)
                4'd5: begin //nhan 20, tra nuoc
                out_water = 1;
                out5 = 0;
                out10 = 0;
                out15 = 0;
                end
                4'd6: begin //nhan 25, tra nuoc va 5 xu
                out_water = 1;
                out5 = 1;
                out10 = 0;
                out15 = 0;
                end
                4'd7: begin //cancel, tra 5 xu
                out_water = 0;
                out5 = 1;
                out10 = 0;
                out15 = 0;
                end
                4'd8: begin //cancel, tra 10 xu
                out_water = 0;
                out5 = 0;
                out10 = 1;
                out15 = 0;
                end
                4'd9: begin //cancel, tra 15 xu
                out_water = 0;
                out5 = 0;
                out10 = 0;
                out15 = 1;
                end
                default: begin
                out_water = 0;
                out5 = 0;
                out10 = 0;
                out15 = 0;
                end
        endcase
end

endmodule
```

Hình 5.6. *Lập trình Verilog cho máy trạng thái của bài thực hành-thí nghiệm 3 (kiểu Moore).*

5.2.4. Thực hiện viết testbench

Bảng 5.4. *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 (System Verilog)	- 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 5.4. là mô tả khái quát của bước kiểm tra thiết kế cấp độ RTL (Verification).

File **testtop.v** trong bài thực hành-thí nghiệm này đơn giản làm 4 công việc sau (*Hình 5.7.*).

- Gọi máy trạng thái dưới dạng instance.
- Tạo clock.
- Tạo các chuỗi ngõ vào theo từng clock.
- In các giá trị cần quan sát ra Terminal. Chú ý rằng biến **state** của máy trạng thái là biến nội, không thể nhìn trực tiếp trong testbench mà phải thông qua gọi phân cấp (**vm_inst.state**).

Thực hiện các lệnh trong Terminal để chạy thiết kế và mô phỏng RTL.

Hình 5.8. Ta được kết quả mô phỏng RTL.

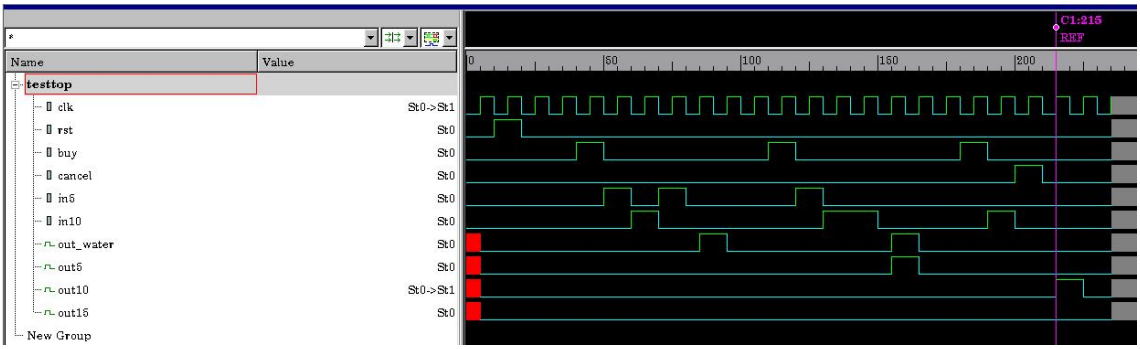
Hình 5.9. hiển thị kết quả quan sát dạng sóng bằng phần mềm DVE.

```
module testtop;

reg     clk;
reg     rst;
reg     buy;
reg     cancel;
reg     in5, in10;
wire    out_water;
wire    out5, out10, out15;

vending_machine vm_inst(
        clk,
        rst,
        buy,
        cancel,
        in5,
        in10,
        out_water,
        out5,
        out10,
        out15);

always begin #5 clk=~clk; end

initial begin
        clk = 0;
        //reset
        buy = 0; in5 = 0; in10 = 0; cancel = 0;
        rst = 0; #10
        rst = 1; #10
        rst = 0; #20
        //mua du 20 xu
        buy = 1; #10;
        buy = 0; in5 = 1; #10
        in5 = 0; in10 = 1; #10
        in10 = 0; in5 = 1; #10
        in5 = 0; #30
        //mua 25 xu
        buy = 1; #10;
        buy = 0; in5 = 1; #10
        in5 = 0; in10 = 1; #10
        in10 = 1; #10
        in10 = 0; #30
        //cancel, tra 10
        buy = 1; #10;
        buy = 0; in10 = 1; #10
        in10 =0; cancel = 1; #10
        cancel = 0; #30
        $finish;
end
always @(posedge clk) begin
        $display("At %t, state = %d, rst = %b, buy = %b, cancel = %b, in5 = %b, in10 = %b",$time, vm_inst.state, rst, buy, cancel, in5, in10);
        $display ("Water out = %b, out5 = %b, out10 = %b, out15 = %b",out_water, out5, out10, out15);
end

endmodule
```

Hình 5.7. *Lập trình testbench cho máy trạng thái của bài thực hành-thí nghiệm 3.*

```
Compiler version M-2017.03-SP2_Full64; Runtime version M-2017.03-SP2_Full64;  Jul 17
 03:42 2024
At                    5, state =  x, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = x, out5 = x, out10 = x, out15 = x
At                   15, state =  0, rst = 1, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   25, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   35, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   45, state =  0, rst = 0, buy = 1, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   55, state =  1, rst = 0, buy = 0, cancel = 0, in5 = 1, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   65, state =  2, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 1
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   75, state =  4, rst = 0, buy = 0, cancel = 0, in5 = 1, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   85, state =  5, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                   95, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 1, out5 = 0, out10 = 0, out15 = 0
At                  105, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  115, state =  0, rst = 0, buy = 1, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  125, state =  1, rst = 0, buy = 0, cancel = 0, in5 = 1, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  135, state =  2, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 1
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  145, state =  4, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 1
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  155, state =  6, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  165, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 1, out5 = 1, out10 = 0, out15 = 0
At                  175, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  185, state =  0, rst = 0, buy = 1, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  195, state =  1, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 1
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  205, state =  3, rst = 0, buy = 0, cancel = 1, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  215, state =  8, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
At                  225, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 1, out15 = 0
At                  235, state =  0, rst = 0, buy = 0, cancel = 0, in5 = 0, in10 = 0
Water out = 0, out5 = 0, out10 = 0, out15 = 0
$finish called from file "/home/albert/Desktop/Lab3_vending_machine/03_verif/verif/s
v/testtop.v", line 49.
$finish at simulation time                 240
           V C S   S i m u l a t i o n   R e p o r t
Time: 240
CPU Time:      2.020 seconds;       Data structure size:   0.0Mb
Wed Jul 17 03:42:36 2024
CPU time: 1.957 seconds to compile + 1.028 seconds to elab + 1.054 seconds to link +
 2.109 seconds in simulation
[albert@localhost scripts]$
```

Hình 5.8. *Kết quả mô phỏng RTL cho máy trạng thái của bài thực hành-thí nghiệm 3.*



Hình 5.9. *Quan sát dạng sóng bằng phần mềm DVE cho máy trạng thái (kiểu Moore).*

5.2.5 Tổng hợp (Synthesis)

Bảng 5.5. là mô tả khái quát của bước Synthesis (tổng hợp) thiết kế.

Quá trình Synthesis diễn ra tương tự các bài thực hành-thí nghiệm trước (*Hình 5.10.*). Lưu ý rằng trong bài thực hành-thí nghiệm này, thư viện sử dụng cần phải hỗ trợ cho các cell ở cả hai mảng logic và vật lý. Trong đó, thư viện logic dùng cho các quá trình Front-End, trong khi ở Back-End, cụ thể là Place&Route, thư viện vật lý sẽ được sử dụng.

Bảng 5.5. *Mô tả bước Synthesis (tổng hợp) thiết kế.*

Đầu vào	Đầu ra	Các công đoạn
- Các file mã lập trình 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 báo cáo (report) 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 tham khảo), thường gọi là nettlist.	- 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).

```
File  Edit  View  Search  Terminal  Help
#!/bin/bash

#============== SET DIRECTORY =================
#set search_path "./../lib"
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"
set synthesis_library standard.sldb

#============= ANALYSE DESIGN ==================
analyze -format verilog -vcs "./../../02_rtl/vending_machine.v"
elaborate vending_machine
current_design vending_machine

#============= CONSTRAINT FOR DESIGN ===========
create_clock -name clk -period 40 {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]
set_fanout_load 8 [all_outputs]

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

#============= REPORT PERFORMANCE ==============
report_area > ./../report/report.area
report_timing > ./../report/report.timing
report_power > ./../report/report.power
report_port > ./../report/report.port
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

quit
```

Hình 5.10. *Chỉnh sửa thư viện tham khảo cho file dc_command.src.*

```
module vending_machine ( clk, rst, buy, cancel, in5, in10, out_water, out5,
        out10, out15 );
  input clk, rst, buy, cancel, in5, in10;
  output out_water, out5, out10, out15;
  wire   N72, N73, N74, N75, N101, N102, N103, N104, n35, n36, n37, n38, n39,
         n40, n41, n42, n43, n45, n46, n47, n48, n49, n50, n51, n52, n53, n54,
         n55, n56, n57, n58, n59, n60, n61, n62, n63, n64, n65, n66, n67, n68,
         n69, n70;
  wire   [3:0] state;

  DFF_X1 \state_reg[0]  ( .D(N72), .CK(clk), .Q(state[0]), .QN(n70) );
  DFF_X1 \state_reg[3]  ( .D(N75), .CK(clk), .Q(state[3]), .QN(n68) );
  DFF_X1 out10_reg ( .D(N103), .CK(clk), .Q(out10) );
  DFF_X1 \state_reg[2]  ( .D(N74), .CK(clk), .Q(state[2]), .QN(n69) );
  DFF_X1 \state_reg[1]  ( .D(N73), .CK(clk), .Q(state[1]), .QN(n67) );
  DFF_X1 out5_reg ( .D(N102), .CK(clk), .Q(out5) );
  DFF_X1 out15_reg ( .D(N104), .CK(clk), .Q(out15) );
  DFF_X1 out_water_reg ( .D(N101), .CK(clk), .Q(out_water) );
  NAND2_X1 U45 ( .A1(state[2]), .A2(n68), .ZN(n35) );
  AOI221_X1 U46 ( .B1(state[0]), .B2(state[1]), .C1(n70), .C2(n67), .A(n35),
        .ZN(N101) );
  NOR3_X1 U47 ( .A1(state[3]), .A2(n69), .A3(n67), .ZN(N102) );
  NAND2_X1 U48 ( .A1(n69), .A2(state[0]), .ZN(n52) );
  NOR3_X1 U49 ( .A1(state[1]), .A2(n68), .A3(n52), .ZN(N104) );
  NOR2_X1 U50 ( .A1(state[2]), .A2(state[0]), .ZN(n36) );
  NAND2_X1 U51 ( .A1(n36), .A2(n67), .ZN(n47) );
  NOR2_X1 U52 ( .A1(n47), .A2(n68), .ZN(N103) );
  NOR2_X1 U53 ( .A1(in5), .A2(cancel), .ZN(n40) );
  OR2_X1 U54 ( .A1(rst), .A2(state[3]), .ZN(n65) );
  NOR4_X1 U55 ( .A1(state[0]), .A2(state[2]), .A3(n67), .A4(n65), .ZN(n56) );
  INV_X1 U56 ( .A(n56), .ZN(n49) );
  NOR3_X1 U57 ( .A1(state[1]), .A2(state[0]), .A3(n69), .ZN(n61) );
  INV_X1 U58 ( .A(n40), .ZN(n42) );
  INV_X1 U59 ( .A(n52), .ZN(n46) );
  INV_X1 U60 ( .A(in5), .ZN(n41) );
  INV_X1 U61 ( .A(buy), .ZN(n37) );
  AOI21_X1 U62 ( .B1(n41), .B2(n37), .A(n47), .ZN(n38) );
  AOI221_X1 U63 ( .B1(n61), .B2(n42), .C1(n46), .C2(n40), .A(n38), .ZN(n39) );
  OAI22_X1 U64 ( .A1(n40), .A2(n49), .B1(n39), .B2(n65), .ZN(N72) );
  INV_X1 U65 ( .A(in10), .ZN(n55) );
  NOR2_X1 U66 ( .A1(n55), .A2(n42), .ZN(n48) );
  AOI21_X1 U68 ( .B1(n41), .B2(n55), .A(cancel), .ZN(n53) );
  INV_X1 U69 ( .A(n53), .ZN(n43) );
  OAI33_X1 U70 ( .A1(1'b0), .A2(state[1]), .A3(n43), .B1(n67), .B2(in10), .B3(
        n42), .ZN(n45) );
  AOI22_X1 U71 ( .A1(n61), .A2(n48), .B1(n46), .B2(n45), .ZN(n51) );
  NOR3_X1 U72 ( .A1(buy), .A2(n47), .A3(n65), .ZN(n62) );
  NAND2_X1 U73 ( .A1(in5), .A2(n62), .ZN(n58) );
  OR2_X1 U74 ( .A1(n49), .A2(n48), .ZN(n50) );
  OAI211_X1 U75 ( .C1(n51), .C2(n65), .A(n58), .B(n50), .ZN(N73) );
  INV_X1 U76 ( .A(cancel), .ZN(n54) );
  NOR2_X1 U77 ( .A1(n67), .A2(n52), .ZN(n60) );
  AOI22_X1 U78 ( .A1(n61), .A2(n54), .B1(n53), .B2(n60), .ZN(n59) );
  NOR2_X1 U79 ( .A1(in5), .A2(n55), .ZN(n63) );
  OAI21_X1 U80 ( .B1(cancel), .B2(n63), .A(n56), .ZN(n57) );
  OAI211_X1 U81 ( .C1(n59), .C2(n65), .A(n58), .B(n57), .ZN(N74) );
  OAI21_X1 U82 ( .B1(n61), .B2(n60), .A(cancel), .ZN(n66) );
  NAND2_X1 U83 ( .A1(n63), .A2(n62), .ZN(n64) );
  OAI21_X1 U84 ( .B1(n66), .B2(n65), .A(n64), .ZN(N75) );
endmodule
```

Hình 5.11. *Kết quả RTL sau khi Synthesis cho máy trạng thái (kiểu Moore).*

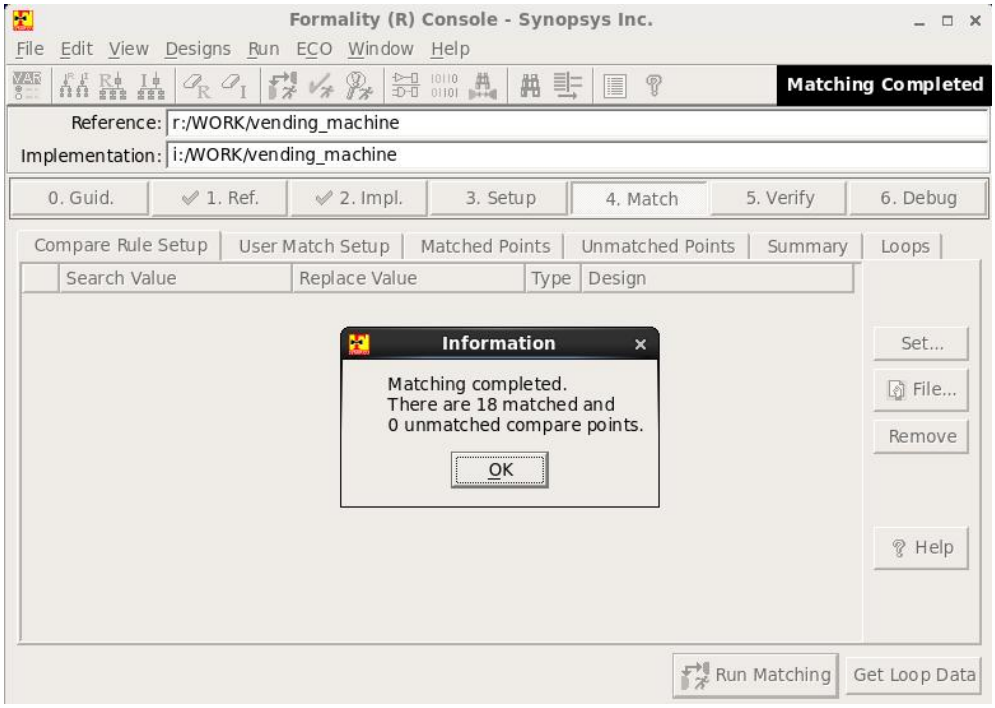
5.2.6 Kiểm tra netlist

Bảng 5.6. *Mô tả bước kiểm tra netlist.*

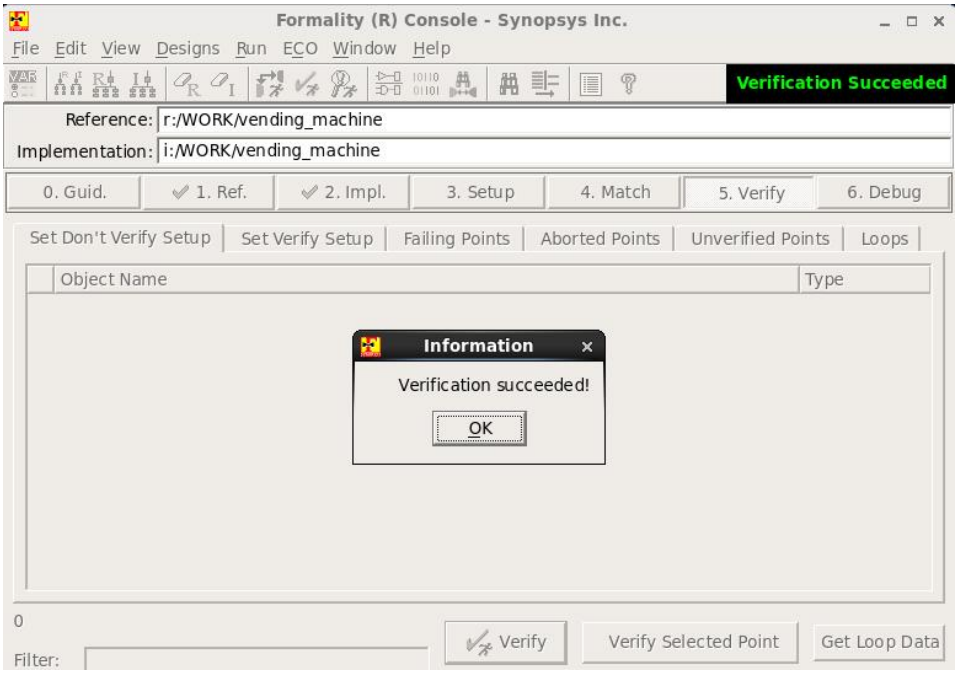
Đầu vào	Đầu ra	Các công đoạn
- Các file mã lập trình RTL đuôi .v - Các file RTL đã tổng hợp (netlist) đuôi .v	- Các báo cáo (report) về quá trình Matching. - Các báo cáo (report) về quá trình Verify.	- Lựa chọn các file thiết kế RTL. Lựa chọn file RTL đã tổng hợp cùng thư viện tham khảo đi kèm. - Thực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

Bảng 5.6. là mô tả khái quát của bước kiểm tra netlist.

Tương tự bài thực hành-thí nghiệm trước, quá trình kiểm tra netlist (bao gồm kiểm tra Matching và Verifying) cũng theo các bước đã mô tả thông qua phần mềm **Formality**. Lưu ý, khi chọn thư viện tham khảo cho netlist, cần chọn đúng thư viện dùng trong Synthesis.



Hình 5.12. *Kiểm tra tương đồng giữa RTL trước và sau Synthesize.*



Hình 5.13. *Kiểm tra chức năng RTL sau Synthesize.*

5.2.7 Phân tích thời gian tĩnh (STA)

Bảng 5.7. là mô tả khái quát của bước phân tích thời gian tĩnh (STA).

Bảng 5.7. *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 quá 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).

Việc thực hiện STA ta thực hiện tương tự bài thực hành-thí nghiệm trước. File **sta_command.src** và kết quả kiểm tra timing được thể hiện trong *Hình 5.14. và Hình 5.15.*

```
#!/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 "/home/hoangtrang/Desktop/Lab3_vending_machine/04_synth/report/lab_synth.netlist.v"
current_design vending_machine
link

#============= CONSTRAINT FOR STA ============
set_units -time ns -resistance MOhm -capacitance fF -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 1 -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/report_design/report.design
report_net > report/report_design/report.net
report_cell > report/report_design/report.cell
report_hierarchy > report/report_design/report.hierarchy
report_clock > report/report_design/report.clock
report_path_group > report/report_design/report.pg

#============= REPORT STA ==============
report_timing -delay_type max > report/report_sta/report.timing
report_timing -delay_type min >> report/report_sta/report.timing
report_port > report/report_sta/report.port
report_constraint > report/report_sta/report.constraint
report_global_slack > report/report_sta/report.glbslk
report_analysis_coverage > report/report_sta/report.anacov
report_qor > report/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 5.14. *File* ***sta_command.src.***

```
Startpoint: rst (input port clocked by clk)
Endpoint: state_reg[1]
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              2.00       2.00
input external delay                    10.00      12.00 f
rst (in)                                 0.00      12.00 f
U54/ZN (OR2_X1)                          0.07      12.07 f
U55/ZN (NOR4_X1)                         0.13      12.19 r
U56/ZN (INV_X1)                          0.03      12.22 f
U74/ZN (OR2_X1)                          0.06      12.28 f
U75/ZN (OAI211_X1)                       0.03      12.31 r
state_reg[1]/D (DFF_X1)                  0.01      12.32 r
data arrival time                                  12.32

clock clk (rise edge)                   20.00      20.00
clock network delay (ideal)              2.00      22.00
clock reconvergence pessimism            0.00      22.00
clock uncertainty                       -0.10      21.90
state_reg[1]/CK (DFF_X1)                           21.90 r
library setup time                      -0.03      21.87
data required time                                 21.87
-----------------------------------------------------------
data required time                                 21.87
data arrival time                                 -12.32
-----------------------------------------------------------
slack (MET)                                         9.55
```

```
Startpoint: state_reg[0]
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: out_water_reg
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: min

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              2.00       2.00
state_reg[0]/CK (DFF_X1)                 0.00       2.00 r
state_reg[0]/QN (DFF_X1)                 0.08       2.08 f
U46/ZN (AOI221_X1)                       0.04       2.13 r
out_water_reg/D (DFF_X1)                 0.01       2.13 r
data arrival time                                   2.13

clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              2.00       2.00
clock reconvergence pessimism            0.00       2.00
clock uncertainty                        0.10       2.10
out_water_reg/CK (DFF_X1)                           2.10 r
library hold time                        0.03       2.13
data required time                                  2.13
-----------------------------------------------------------
data required time                                  2.13
data arrival time                                  -2.13
-----------------------------------------------------------
slack (MET)                                         0.00
```

Hình 5.15. *Kết quả kiểm tra timing trong bước STA.*

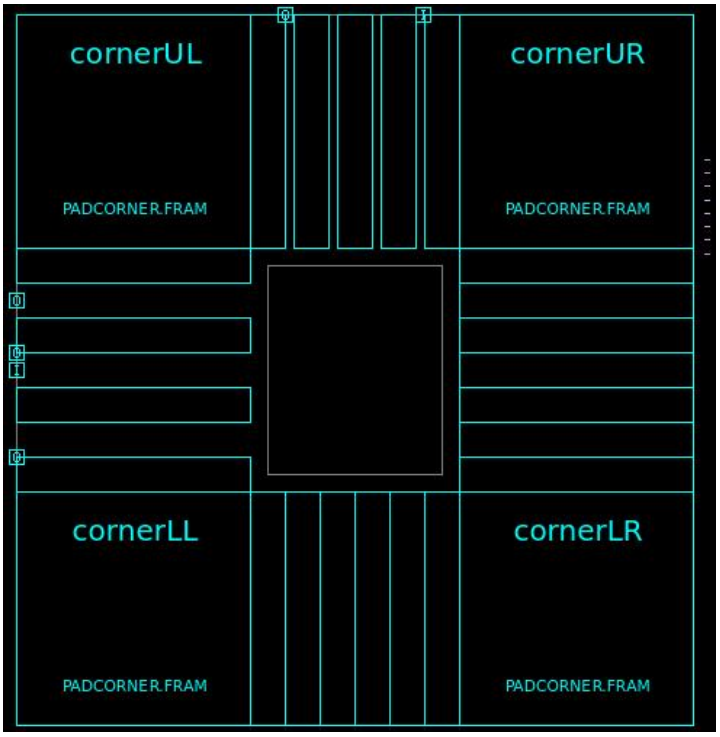
5.2.8 Place & Route

Bảng 5.8. là mô tả khái quát của bước Place & Route.

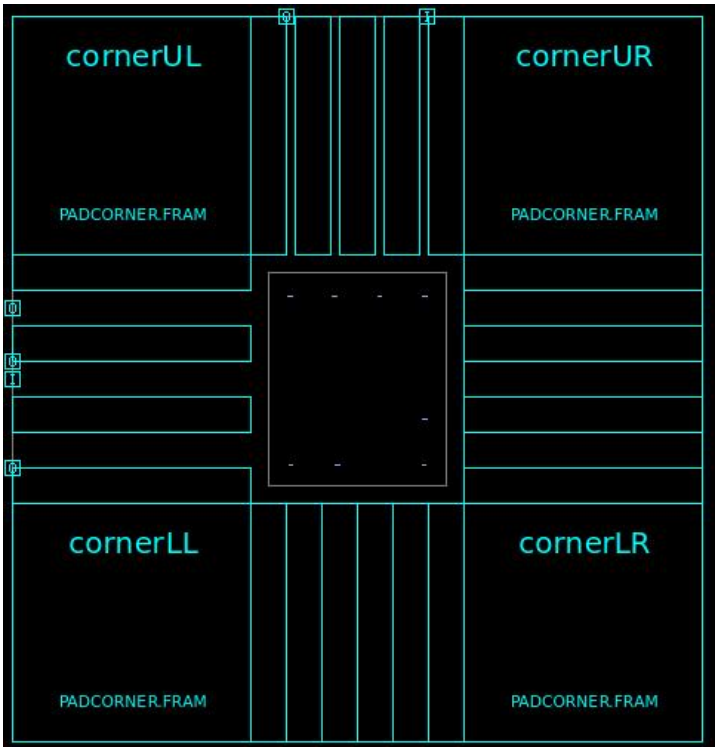
Bảng 5.8. *Mô tả bước Place & Route.*

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 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).	- Các báo cáo (report) về kiểm tra DRC. - Các báo cáo (report) về kiểm tra LVS. - File GDSII đuôi .gds để gửi đến nhà máy sản xuất.	- Tạo thư viện Milkyway và import thiết kế. - Floor Planning: đặt các thành phần nền tảng lên bề 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 (De sign Rule Check) nhằm tìm ra vi phạm với ràng buộc của nhà máy sản xuất (nếu có). - Kiểm tra LVS (Lay out 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.

Các bước thực hiện Place&Route thực hiện tương tự bài thực hành-thí nghiệm trước, với kết quả thể hiện trong các hình ảnh sau (từ *Hình 5.16.* đến *Hình 5.22.*).



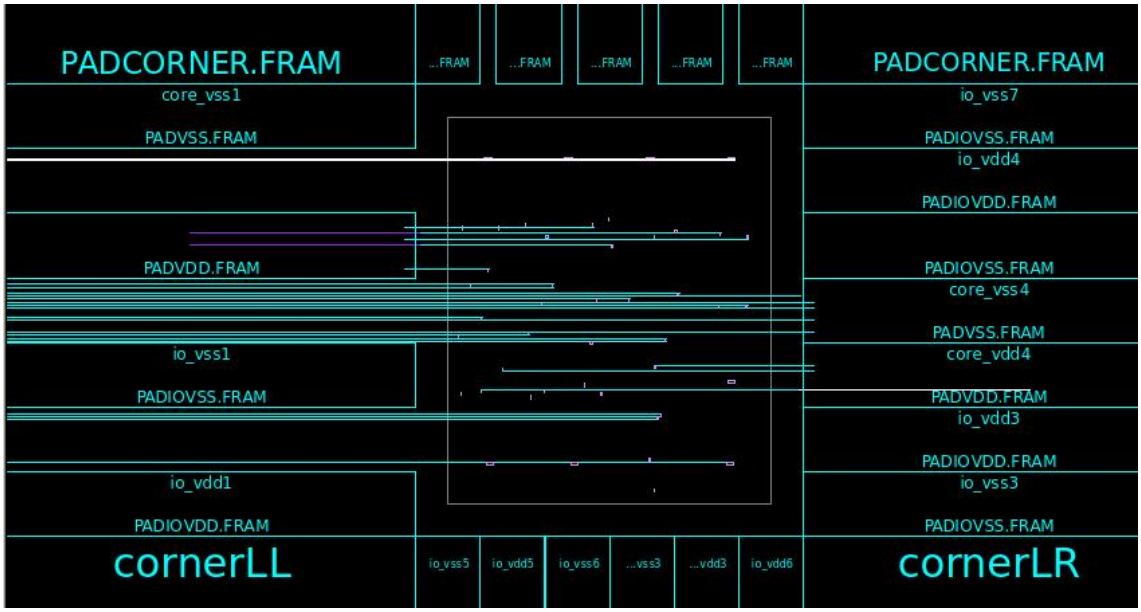
Hình 5.16. *Kết quả chạy Floorplanning.*



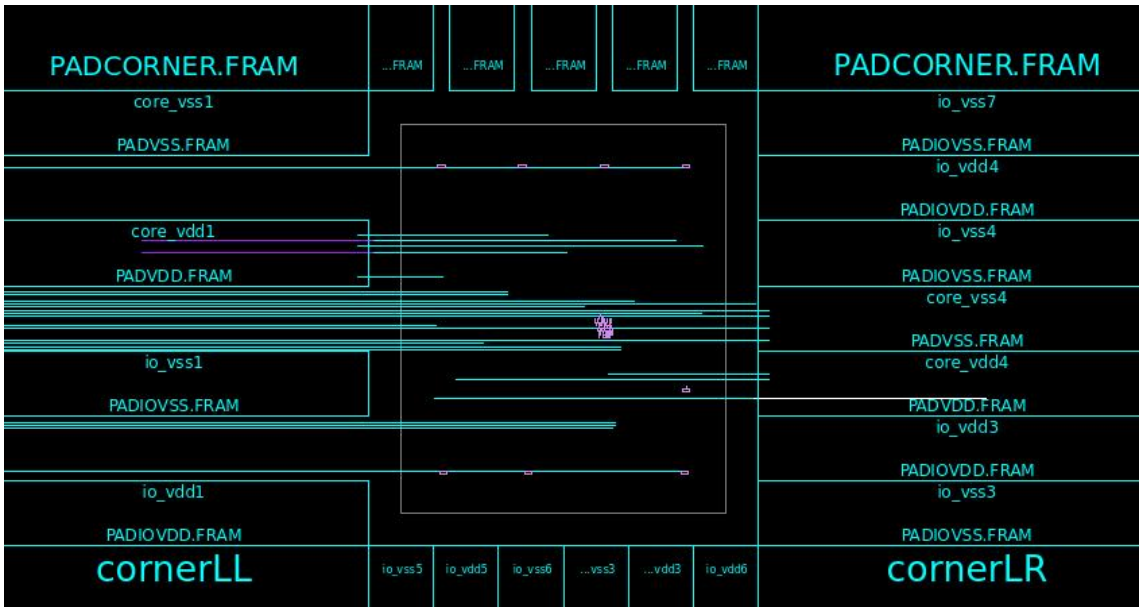
Hình 5.17. *Kết quả chạy Placement.*

```
icc_shell> derive_pg_connection -power_net {VDD} -ground_net {VSS} -power_pin {VDD} -ground_pin {VSS}
Information: connected 51 power ports and 51 ground ports
```

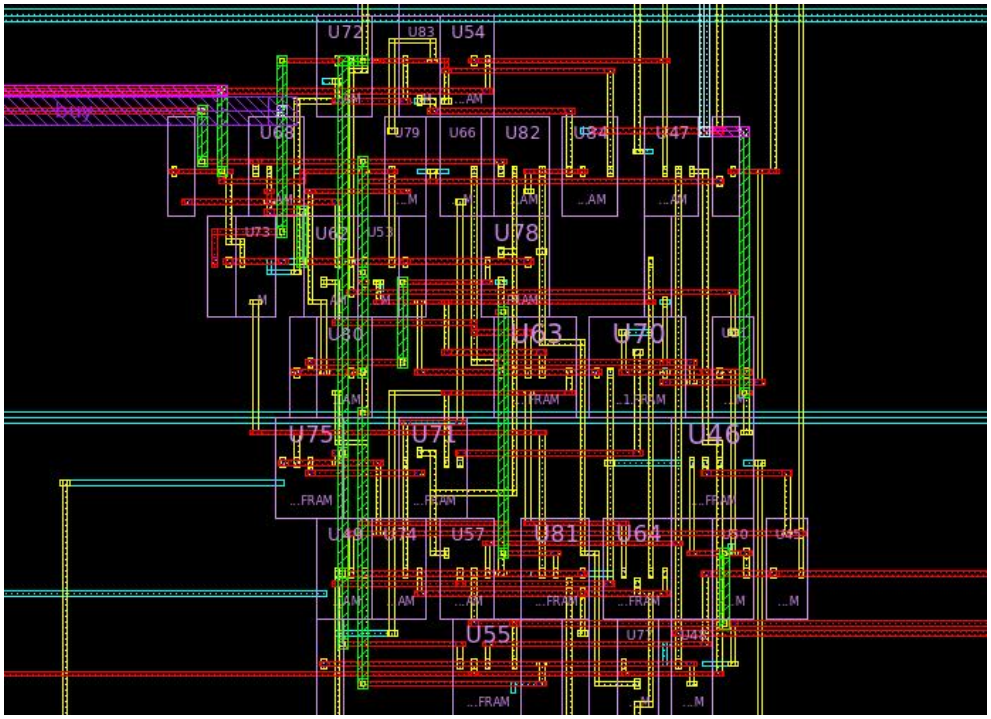
Hình 5.18. *Kết quả chạy nhận tín hiệu nguồn/đất.*



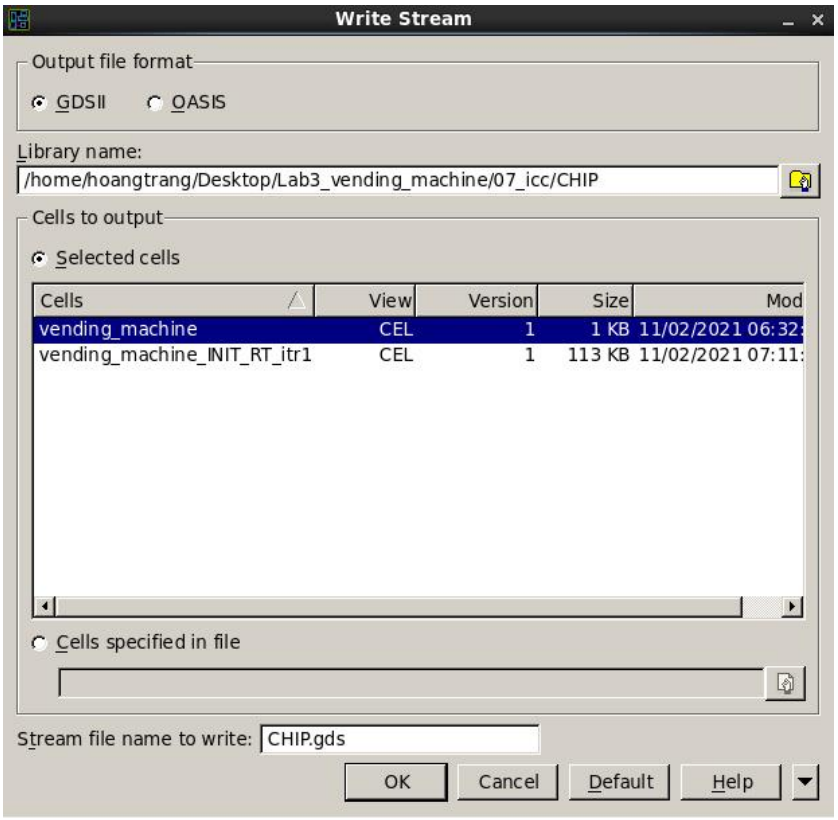
Hình 5.19. *Kết quả nối trước tín hiệu nguồn/đất.*



Hình 5.20. *Kết quả chạy tổng hợp cây clock.*



Hình 5.21. *Kết quả chạy quá trình Routing.*



Hình 5.22. *Xuất file GDSII để gửi đến nhà máy.*

Đến đây là kết thúc bài thực hành-thí nghiệm 3.