|  |  |  |  |
| --- | --- | --- | --- |
| Họ tên sinh viên | MSSV | Lớp (thứ - tiết) |  |
| Hoàng Ngọc Dung | 23139006 | Thứ 7 - Tiết 7 - 9 |

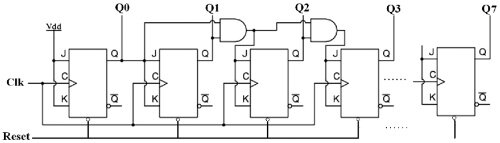
Chú ý: Sinh viên thay bằng QR code của mã số sinh viên (ví dụ: 23119012), có thể tham khảo tại https://barcode.tec-it.com)

Quick question : chapter 6

*Lưu ý: Trong mỗi thiết kế yêu cầu sinh viên thực hiện*

* *Sơ đồ khối (nguyên lý, cấu trúc)*
* *Bảng trạng thái*
* *Mô tả bằng ngôn ngữ Verilog cho module cần thiết kế,*
* *Mô tả Verilog cho module dùng để kiểm tra thiết kế*
* *Kết quả mô phỏng quá trình kiểm tra, có phân tích*
* *Module test được đặt tên theo cú pháp: tensv\_testbench\_tenmodule, ví dụ để test module encoder, sinh viên Nguyen Van An phải đặt tên module test như sau: An\_testbench\_encoder. Các kết quả mô phỏng phải được chụp màn hình bao gồm cả tên của module test trong đó có tên sinh viên thì mới hợp lệ*

1. Thiết kế và mô phỏng mạch đếm đống bộ 8 bit sử dụng phương pháp thiết kế mạch tuần tự



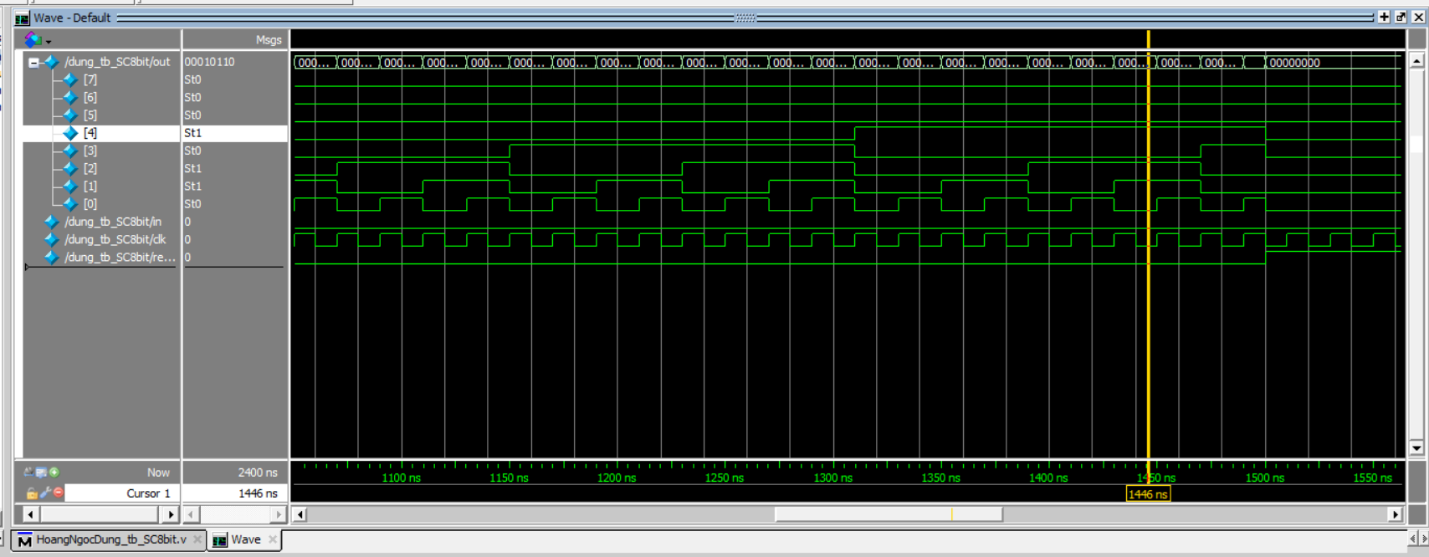
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CLK | Q7 (MSB) | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 (LSB) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 254 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 255 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Mô tả bằng ngôn ngữ Verilog

|  |
| --- |
| module Counter  #(parameter N = 8)  ( input wire clk, reset,  output wire [N-1:0] q );  // signal declaration  reg [N-1:0] r\_reg;  wire [N-1:0] r\_next;  // body, register  always @(posedge clk, posedge reset)  if (reset)  r\_reg <= 0;  else  r\_reg<=r\_next; // <= is non-blocking statement  // next state logic  assign r\_next = r\_reg + 1;  // output logic  assign q=r\_reg;  endmodule |

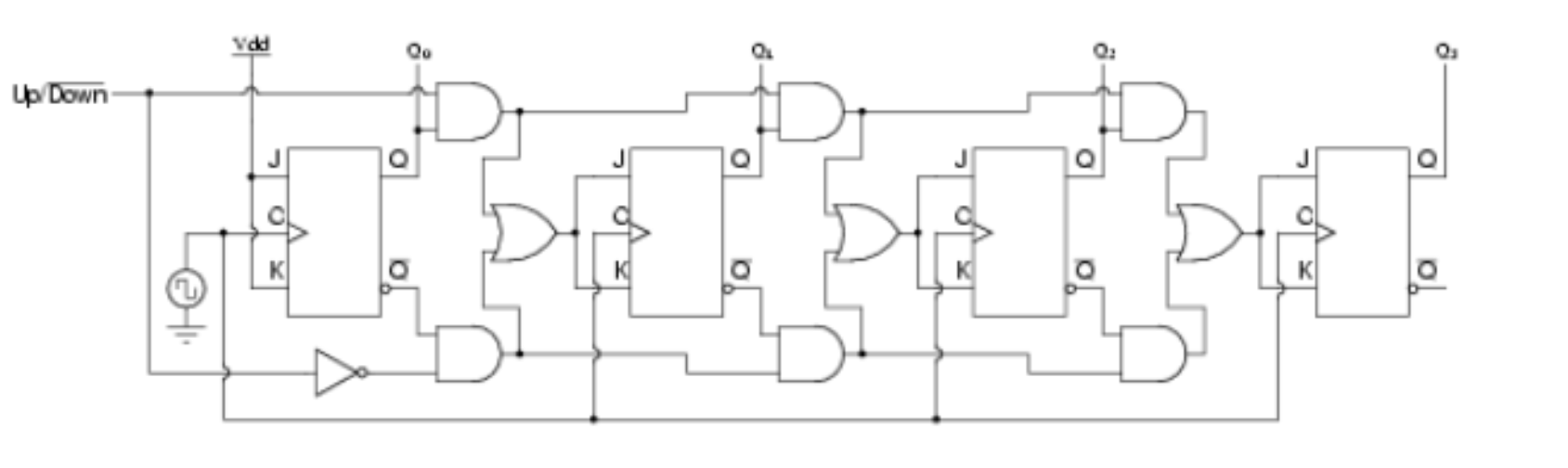
Mô tả Verilog cho module dùng để kiểm tra thiết kế

|  |
| --- |
| `timescale 1ns/1ns  module dung\_tb\_SC8bit;  wire [7:0] out;  reg in, clk, reset;  initial begin  in = 0;  clk = 0;  reset = 0;  end  always forever #10 clk = ~clk;  always forever #500 reset = ~reset;  Counter m0(clk, reset, out );  endmodule |



Hình 1: Hình ảnh mô tả kết quả cấp xung đầu vào in và xung Clk. Tại thời điểm 1100ns xung ngõ ra 0000101 sau 10ns output: 000110 .. đến thời điểm 1445ns output: 00010110 => phù hợp với bảng trạng thái của mạch tuần tự đồng bộ 8 bit

1. Thiết kế và mô phỏng kiểm chứng mạch đếm đồng bộ 8 bit, có tín hiệu UD cho phép đếm lên/đếm xuống



Bảng trạng thái

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Up/Down | Current State (Q7-Q0) | Next State (Q7-Q0) | J7 K7 | J6 K6 | ... | J0 K0 |
| 1 (Up) | 0 | 1 | 0 X | 0 X | ... | 1 X |
| 1 (Up) | 1 | 10 | 0 X | 0 X | ... | 1 X |
| 1 (Up) | 1111111 | 10000000 | 1 X | 1 X | ... | 1 X |
| 1 (Up) | 11111111 | 0 | X 1 | X 1 | ... | X 1 |
| 0 (Down) | 11111111 | 11111110 | X 0 | X 0 | ... | X 1 |
| 0 (Down) | 10000000 | 1111111 | X 1 | X 1 | ... | X 1 |
| 0 (Down) | 1 | 0 | X 0 | X 0 | ... | X 1 |
| 0 (Down) | 0 | 11111111 | 1 X | 1 X | ... | 1 X |

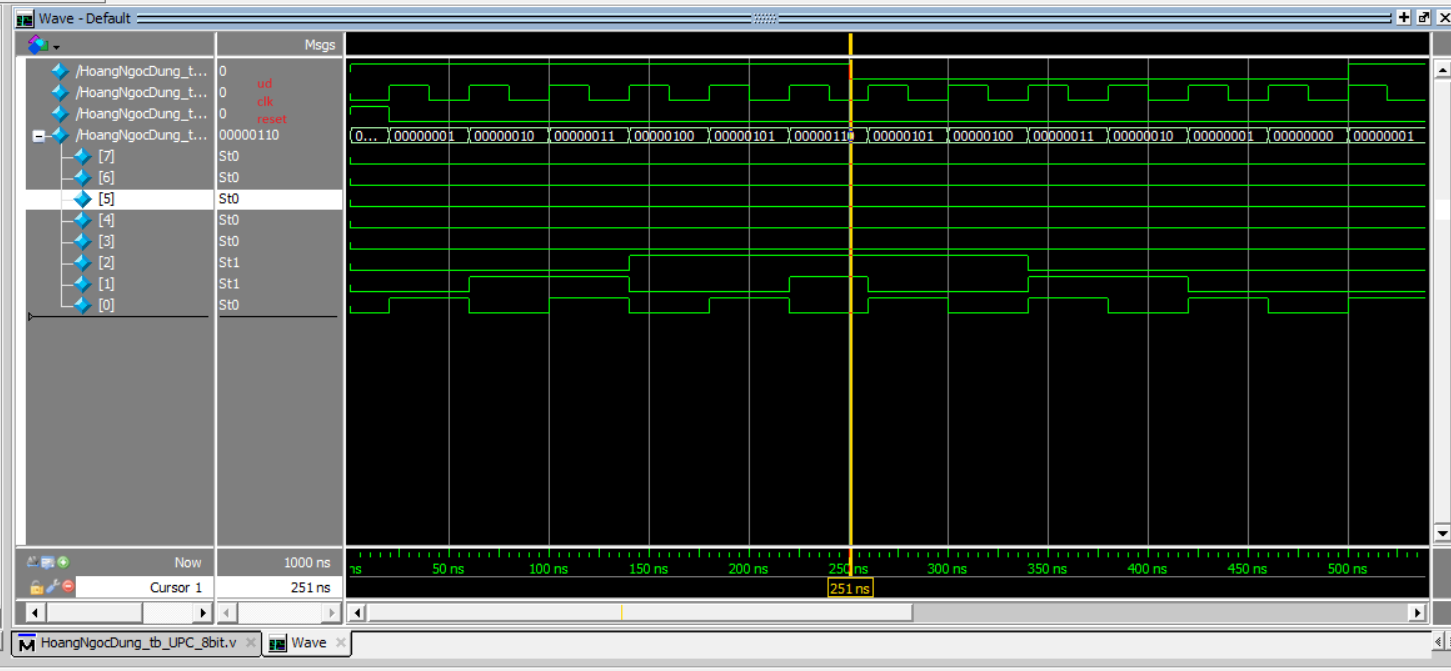
Mô tả bằng ngôn ngữ Verilog

|  |
| --- |
| module CounterUD  ( input wire clk,reset,ud,  output wire [7:0] q );  // signal declaration  reg [7:0] r\_reg;  wire [7:0] r\_next;  // body, register  always @(posedge clk, posedge reset)  if (reset)  r\_reg<=0;  else  r\_reg<=r\_next;  // next state logic  assign r\_next = (ud==1)?r\_reg + 1:r\_reg - 1;  // output logic  assign q=r\_reg;  endmodule |

Mô tảVerilog cho module dùng để kiểm tra thiết kế

|  |
| --- |
| `timescale 1ns/1ns  module HoangNgocDung\_tb\_UPC\_8bit;  wire [7:0] out;     // Đường tín hiệu đầu ra 8-bit từ bộ đếm  reg ud, clk, reset; // Các tín hiệu điều khiển:                      // ud (đếm lên/đếm xuống), clk (xung đồng hồ), reset (đặt lại)  initial begin      ud = 1;         // Khởi tạo chế độ đếm lên (1)      clk = 0;        // Xung đồng hồ bắt đầu ở mức thấp (0)      reset = 1;      // Đặt lại bộ đếm (reset = 1)      #20 reset = 0;  // Sau 20ns, tắt reset để bộ đếm bắt đầu hoạt động  end  always forever #20 clk = ~clk;   // Tạo xung đồng hồ với chu kỳ 40ns  always forever #250 ud = ~ud;    // Đổi chế độ đếm (lên/xuống) mỗi 250ns  always forever #800 reset = ~reset; // Reset bộ đếm mỗi 800ns  // Kết nối module CounterUD (bộ đếm) với các tín hiệu trong testbench  CounterUD m0(      .clk(clk),     // Xung đồng hồ      .reset(reset), // Tín hiệu đặt lại      .ud(ud),       // Tín hiệu đếm lên/đếm xuống      .out(out)      // Đầu ra của bộ đếm  );  endmodule |

Kết quả mô phỏng



Hình 2: Dạng sóng hiển thị sự thay đổi của clk, ud, reset và out (giá trị bộ đếm 8 bit) theo thời gian.

Reset (0ns đến 20ns): Tín hiệu reset ở mức cao (1).

Giá trị của out hiển thị 00000000. Giai đoạn Đếm Lên (từ 20ns đến 250ns): Tại 20ns, tín hiệu reset chuyển xuống mức thấp (0), cho phép bộ đếm hoạt động bình thường. Tín hiệu ud đang ở mức cao (1). Bộ đếm đang ở chế độ đếm lên.

Quan sát dạng sóng out, giá trị của bộ đếm tăng lên tại mỗi cạnh lên của tín hiệu clk (xảy ra tại khoảng 20ns, 60ns, 100ns, 140ns, 180ns, 220ns,...):

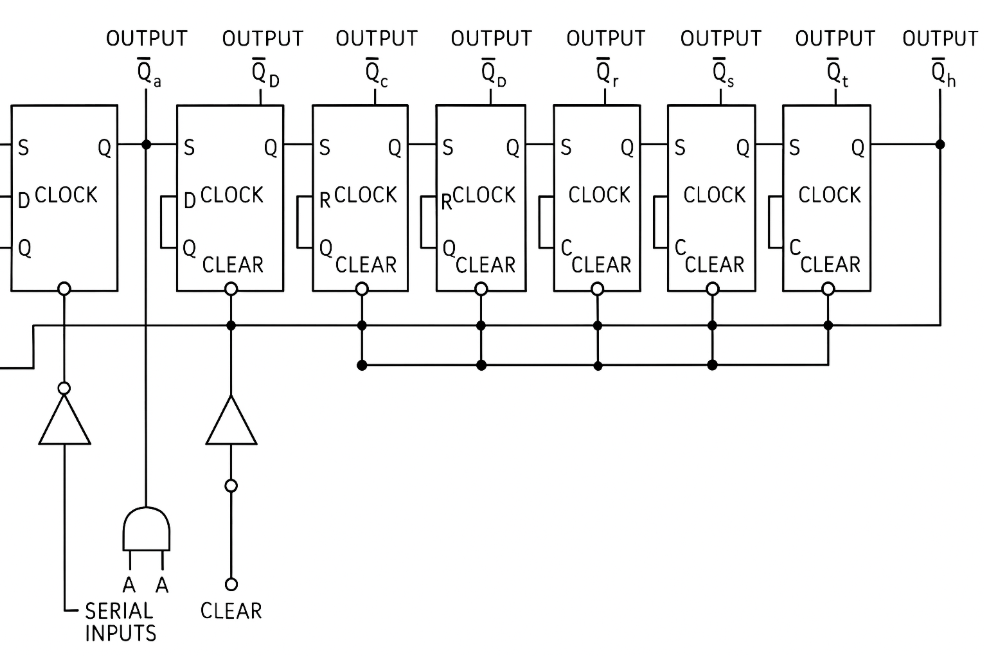
* Tại 20ns (hoặc ngay sau khi reset kết thúc và có cạnh lên clock đầu tiên): out là 00000000.
* Tại ~60ns (cạnh lên clock đầu tiên sau reset): out chuyển lên 00000001 (1 thập phân).
* Tại ~100ns: out chuyển lên 00000010 (2 thập phân).
* Tại ~140ns: out chuyển lên 00000011 (3 thập phân).
* Tại ~180ns: out chuyển lên 00000100 (4 thập phân).
* Tại ~220ns: out chuyển lên 00000101 (5 thập phân).

Giai đoạn Đếm Xuống (từ 250ns trở đi): Tại thời điểm 250ns, tín hiệu ud chuyển từ mức cao (1) xuống mức thấp (0). Bộ đếm chuyển sang chế độ đếm xuống. Tại các cạnh lên của clock sau 250ns (xảy ra tại khoảng 260ns, 300ns, 340ns, v.v.), giá trị của bộ đếm bắt đầu giảm xuống.

Quan sát dạng sóng out từ 250ns trở đi:

* Tại 250ns (hoặc ngay trước cạnh lên clock tiếp theo): out đang là 00000101 (5).
* Tại ~260ns (cạnh lên clock đầu tiên sau khi ud xuống 0): out chuyển xuống 00000100 (4).
* Tại ~300ns: out chuyển xuống 00000011 (3).
* Tại ~340ns: out chuyển xuống 00000010 (2).
* Tại ~380ns: out chuyển xuống 00000001 (1).
* Tại ~420ns: out chuyển xuống 00000000 (0).
* Tại ~460ns: out chuyển xuống 11111111 (255 thập phân - đếm xuống từ 0 sẽ quay vòng về giá trị lớn nhất).

1. Thiết kế và mô phỏng kiểm chứng Thanh ghi dịch 8 bit, vào nối tiếp ra song song

Bảng trạng thái 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chu kỳ Clock | s\_in (giải sử) | r\_reg (Binary) | r\_reg (Hex) | Giải Thích |
| 0 (Khởi tạo) | X | 0 | 0 | Trạng thái ban đầu |
| 1 | 1 | 10000000 | 80 | Bit 1 dịch vào MSB |
| 2 | 0 | 1000000 | 40 | Bit 0 dịch vào, các bit khác dịch phải |
| 3 | 1 | 10100000 | A0 | Bit 1 dịch vào |
| 4 | 1 | 11010000 | D0 | Bit 1 dịch vào |
| 5 | 0 | 1101000 | 68 | Bit 0 dịch vào |
| 6 | 0 | 110100 | 34 | Bit 0 dịch vào |
| 7 | 1 | 10011010 | 9A | Bit 1 dịch vào |

Mô tả bằng ngôn ngữ Verilog

|  |
| --- |
| module ShiftSIPO (      input wire clk,      input wire s\_in,      output wire [7:0] q\_out);  reg [7:0] r\_reg;  wire [7:0] r\_next;  always @(negedge clk)      r\_reg <= r\_next;  assign r\_next = {s\_in, r\_reg[7:1]};  assign q\_out = r\_reg;  endmodule |

Mô tả Verilog cho module dùng để kiểm tra thiết kế

|  |
| --- |
| `timescale 1ns/1ns  module HoangNgocDung\_tb\_UPC\_8bit;  wire [7:0] out;  reg ud, clk, reset;  initial begin      ud = 1;      clk = 0;        // Xung đồng hồ bắt đầu ở mức thấp (0)      reset = 1;      // Đặt lại bộ đếm (reset = 1)      #20 reset = 0;  // Sau 20ns, tắt reset để bộ đếm bắt đầu hoạt động  end  always forever #20 clk = ~clk;  always forever #250 ud = ~ud;  always forever #800 reset = ~reset;  CounterUD m0(      .clk(clk),      .reset(reset),      .ud(ud),      .out(out)  );  endmodule |

Kết quả mô phỏng

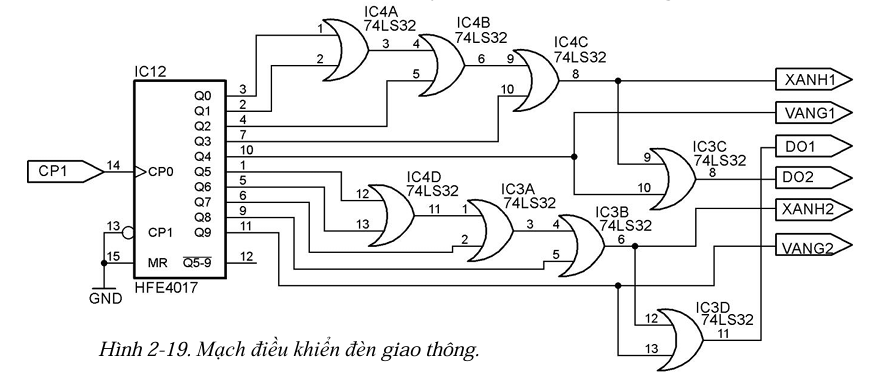


Hình : Dạng sóng hiển thị hoạt động với các tín hiệu clk, i\_in và out[7:0]. Tại cạnh lên clock khoảng 100ns: out chuyển từ 00001xxx sang 100001xx. Bit i\_in (lúc đó là 1) đã được dịch vào. Dựa trên cách các bit cũ (1, 0, 0, 0) dịch chuyển và bit mới vào vị trí LSB (out[0])

* Tại cạnh lên clock khoảng 140ns: i\_in là 0. out chuyển từ 100001xx sang 0100001x. Bit 0 từ i\_in vào out[0], nội dung cũ dịch sang trái.
* Tại cạnh lên clock khoảng 180ns: i\_in là 1. out chuyển từ 0100001x sang 10100001. Bit 1 từ i\_in vào out[0], nội dung cũ dịch trái.

Quá trình này tiếp tục, các bit từ i\_in lần lượt được dịch vào out[0], đẩy các bit cũ sang trái qua out[1], out[2], ..., out[7]. Sau 8 cạnh lên clock kể từ khi thanh ghi rỗng hoặc reset, toàn bộ 8 bit dữ liệu từ i\_in sẽ được nạp đầy vào thanh ghi và hiển thị ở đầu ra out[7:0].

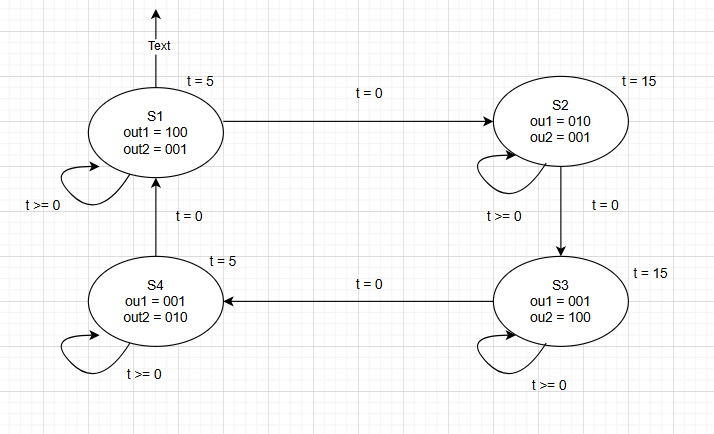
1. Thiết kế và mô phỏng mô hình máy trạng thái cho bài toán điều khiển đèn giao thông, các tín hiệu ngõ ra giải sử chỉ là các ngõ điều khiển đèn xanh, vàng đỏ



- Thiết kế đèn giao thông có 2 ngõ ra 2 trụ đèn tín hiệu thời gian từ Xanh 15s, Vàng 5S và đỏ 20s

Bảng trạng thái

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trạng thái | Đầu ra 1 (out1) | Đầu ra 2 (out2) | Thời gian (giây) | Trạng thái | Đầu ra 1 (out1) | Đầu ra 2 (out2) |
| S1 | 100 | 001 | 15 | S1 | 100 | 001 |
| S2 | 010 | 001 | 5 | S2 | 010 | 001 |
| S3 | 001 | 100 | 15 | S3 | 001 | 100 |
| S4 | 001 | 010 | 5 | S4 | 001 | 010 |



Biểu đồ 1: Sơ đồ nguyên lý dùng mô hình máy trạnh thái đèn giao thông

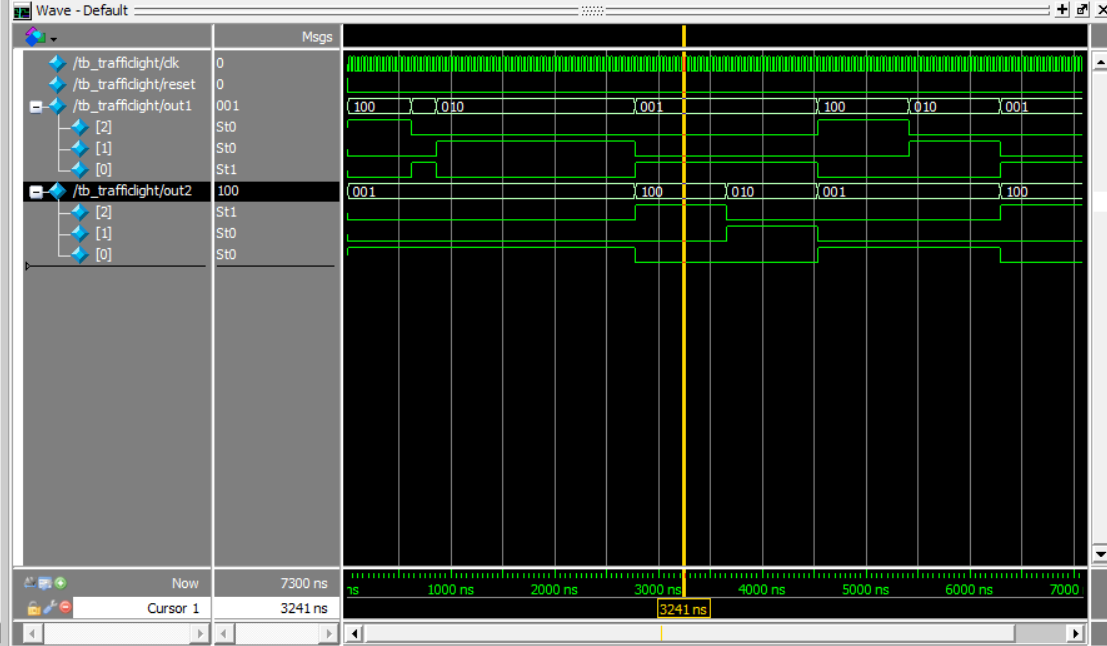
Mô tả bằng ngôn ngữ Verilog

|  |
| --- |
| module traffic\_light(      input clk,          // Clock input      input reset,        // Reset signal      output reg [2:0] out1,  //  1: [Red, Yellow, Green]      output reg [2:0] out2 //  2: [Red, Yellow, Green]  );  parameter S1 = 2'b00, S2 = 2'b01, S3 = 2'b10, S4 = 2'b11;  reg [1:0] state, next\_state;  reg [4:0] timer;  parameter S1\_TIME = 15, // Xanh 15s            S2\_TIME = 5,  // Vàng 5s            S3\_TIME = 15, // Xanh 15s            S4\_TIME = 5;  // Vàng 5s  always @(posedge clk or posedge reset) begin      if (reset) begin          state <= S1;          timer <= S1\_TIME;      end      else begin          timer <= timer - 1;          if (timer == 0) begin              case(state)                  S1: begin                      next\_state <= S2;                      timer <= S2\_TIME;                  end                  S2: begin                      next\_state <= S3;                      timer <= S3\_TIME;                  end                  S3: begin                      next\_state <= S4;                      timer <= S4\_TIME;                  end                  S4: begin                      next\_state <= S1;                      timer <= S1\_TIME;                  end              endcase              state <= next\_state;          end      end  end  always @(state) begin      case(state)          S1: begin              out1 = 3'b100;              out2 = 3'b001;          end          S2: begin              out1 = 3'b010;              out2 = 3'b001;          end          S3: begin              out1 = 3'b001;              out2 = 3'b100;          end          S4: begin              out1 = 3'b001;              out2 = 3'b010;          end          default: begin              out1 = 3'b001;              out2 = 3'b001;          end      endcase  end  endmodule |

Mô tả Verilog cho module dùng để kiểm tra thiết kế

|  |
| --- |
| `timescale 1ns/1ns  module tb\_trafficlight;  reg clk, reset;  wire [2:0] out1,out2;  initial begin  clk = 0;  reset = 1;  #10 reset = 0;  end  always forever #20 clk = ~clk;  traffic\_light uut(clk, reset, out1, out2 );  endmodule |

Kết quả mô phỏng

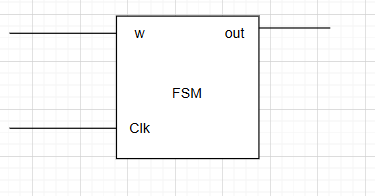


Hình 4: Hình ảnh mô tả kết quả cấpxung Clk và tính hiệu reset. Theo hình ta thấy trước 1000ns out1 = 100 (xanh 1) và ou2 = 001 (đỏ 2). Giai đoạn Reset (0ns đến 10ns). Hoạt động Bình thường (từ 10ns trở đi)

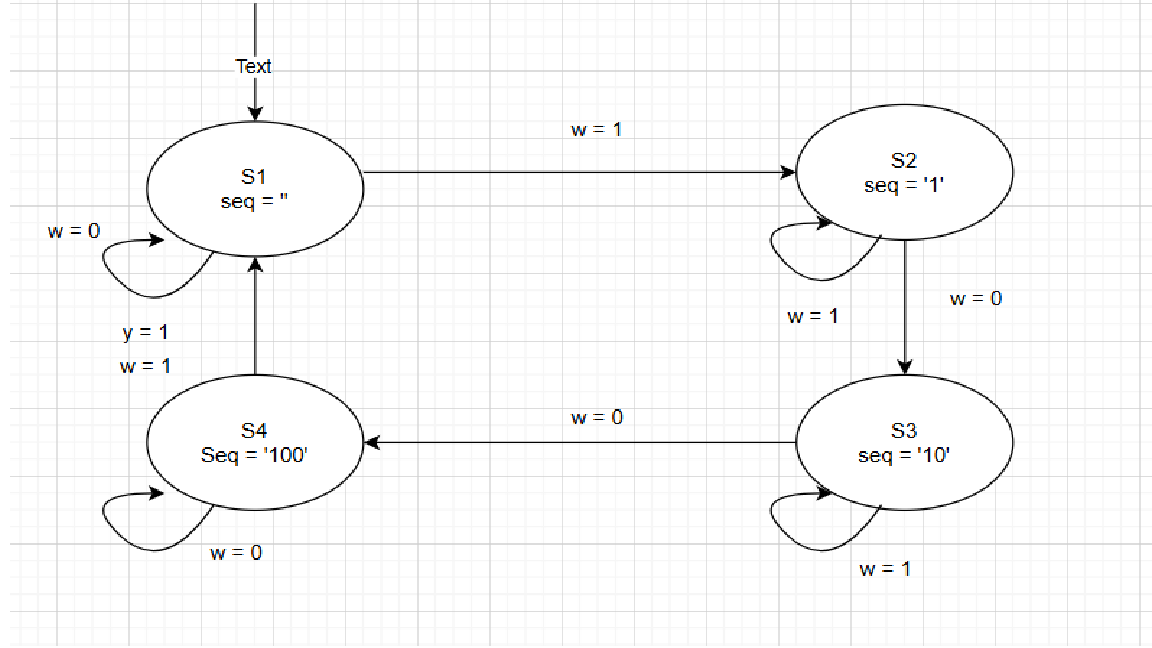
Tại 10ns, tín hiệu reset chuyển xuống mức thấp (0), cho phép module đèn giao thông hoạt động theo chu kỳ.

* Trạng thái 1: Đèn 1 Đỏ, Đèn 2 Xanh out1 = 100 (Đỏ), out2 = 001 (Xanh). Trạng thái này bắt đầu sau reset (khoảng 10ns) và kéo dài đến khoảng 3000ns trên dạng sóng. Thời gian mô phỏng này tương ứng với thời gian Đèn Xanh 15s cho trụ 2 (và Đèn Đỏ 20s cho trụ 1 trong một chu kỳ hoàn chỉnh, nhưng chu kỳ Đỏ dài hơn Xanh). Dựa trên dạng sóng, Đèn Xanh của trụ 2 kéo dài khoảng 3000ns - 10ns ≈ 2990ns.
* Trạng thái 2: Đèn 1 Đỏ, Đèn 2 Vàng , out1 = 100 (Đỏ), out2 = 010 (Vàng). Trạng thái này bắt đầu từ khoảng 3000ns và kéo dài đến khoảng 3500ns. Thời gian mô phỏng này ≈ 500ns, tương ứng với thời gian Đèn Vàng 5s cho trụ 2. Trụ 1 vẫn giữ Đỏ trong giai đoạn này.
* Trạng thái 3: Đèn 1 Xanh, Đèn 2 Đỏout1 = 001 (Xanh), out2 = 100 (Đỏ).Trạng thái này bắt đầu từ khoảng 3500ns và kéo dài đến khoảng 5500ns. Thời gian mô phỏng này ≈ 2000ns, tương ứng với thời gian Đèn Xanh 15s cho trụ 1. Trụ 2 chuyển sang Đỏ trong giai đoạn này (kết thúc Đỏ của trụ 2 ở khoảng 6000ns).
* Trạng thái 4: Đèn 1 Vàng, Đèn 2 Đỏ, out1 = 010 (Vàng), out2 = 100 (Đỏ). Trạng thái này bắt đầu từ khoảng 5500ns và kéo dài đến khoảng 6000ns. Thời gian mô phỏng này ≈ 500ns, tương ứng với thời gian Đèn Vàng 5s cho trụ 1. Trụ 2 vẫn giữ Đỏ trong giai đoạn này.
* Trạng thái 5: Đèn 1 Đỏ, Đèn 2 Xanh, out1 = 100 (Đỏ), out2 = 001 (Xanh). Trạng thái này bắt đầu từ khoảng 6000ns và lặp lại chu kỳ.

1. Thiết kế mô phỏng phát hiện chuỗi 1001 trong chuỗi dữ liệu nối tiếp. Trong do, W: dữ liệu nối tiếp vào theo xung Clk cạnh xuống. Nếu phát hiện 1001 thì output y = 1 ngược lại y = 0



Hình : Sơ đồ khối



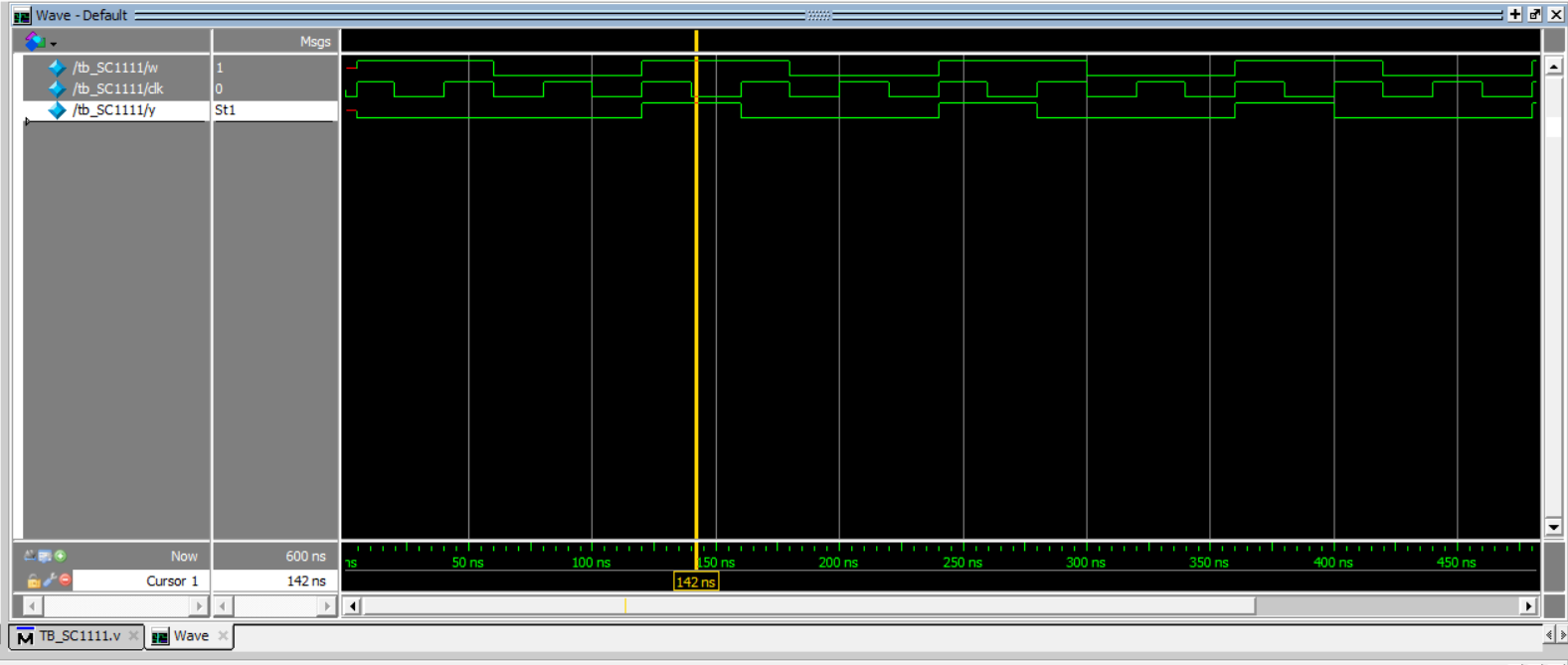
Hình : Sơ đồ nguyên lý dùng mô hình máy trạnh thái phát hiện chuỗi số

Mô tả bằng ngôn ngữ Verilog

|  |
| --- |
| module seq4\_detect\_moore(x, clk, y);    input x, clk;    output reg y;    reg [2:0] state, nstate;    parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011;    // State register (flip-flop)    always @(posedge clk) begin      state <= nstate;    end    // Next state logic (state transitions)    always @(state or x) begin      case(state)        S0: nstate = (x) ? S1 : S0;  // Looking for '1'        S1: nstate = (x) ? S1 : S2;  // Looking for '10'        S2: nstate = (x) ? S0 : S3;  // Looking for '100'        S3: nstate = (x) ? S1 : S0;  // Looking for '1001'        default: nstate = S0;  // Default case (reset to S0)      endcase    end    always @(state) begin      if(state == S3)        y = 1;      else        y = 0;    end  endmodule |

Mô tả Verilog cho module dùng để kiểm tra thiết kế

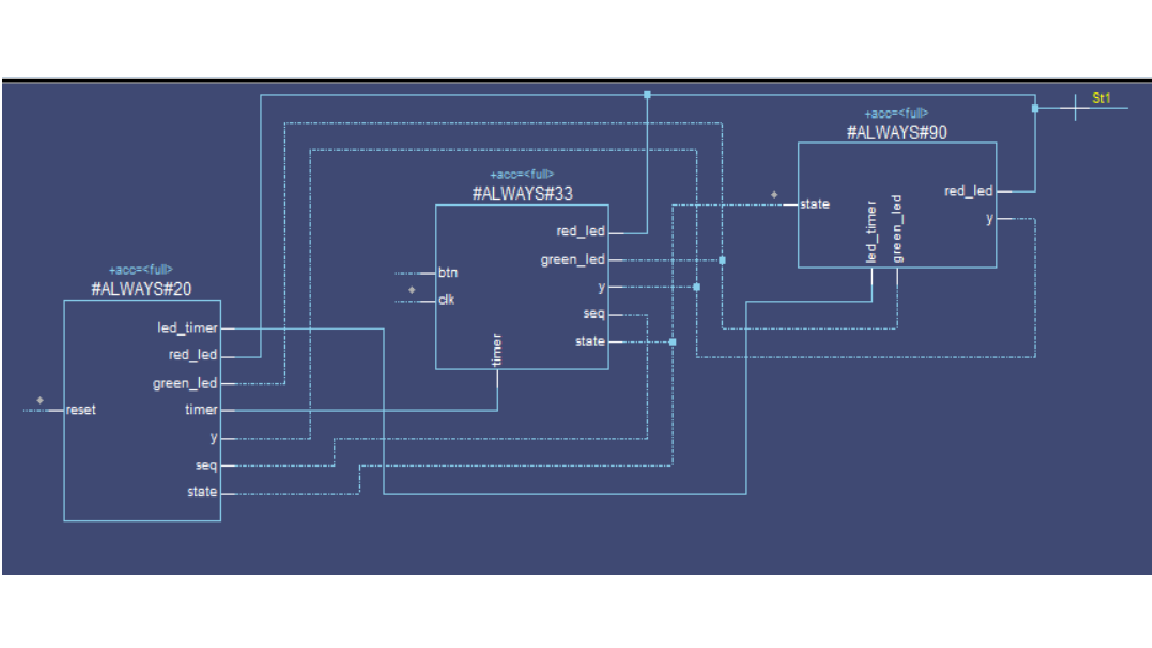
|  |
| --- |
| `timescale 1ns/1ns  module tb\_SC1111;  reg w, clk;  wire  y;  initial begin  clk = 0;  #5;  clk = 1;  w = 1;  end  always forever #20 clk = ~clk;  always forever #60 w = ~w;  seq4\_detect\_moore  uut(w,clk, y );  endmodule |



Hình :Khi xung clk kích cạnh lên , trạng thái ban đầu w = 1 sau 60ns đổi thành 0và tiếp tujv. Ta thấy khi chuỗi nhập đúng bằng 1001 thì y (output = 1). Ngược lại = 0.

1. Thiết kế hệ thống khóa cửa đơn giản hệ thống sử dụng một bàn phím giả lập bang 4 nút nhấn tương ứng với 1,2,3,4

* Mật khẩu mở cửa là 1,3,2,4
* Mỗi người khi dung cửa hệ thống kiểm tra xem nhập dung hay không
* Nếu đúng , đènn led xanh bậc sáng 3s de báo mở cửa
* Nếu sai, den đỏ nhấp nháy 3 lần de báo sai mật khẩu sau do he thong quay lai trạng thái ban đầu.
* Nếu không có thao tác trong vòng 10s quay lai trạng thái ban dau.



Hình 8: Sơ đồ khối của hệ thống

Mô tả bằng ngôn ngữ Verilog (phân tích máy trạng thái)

* **Yêu cầu**: Nhập mật khẩu từ nút nhấn. Kiểm tra xem mật khẩu đúng hay không. Mật khẩu đúng là chuỗi **"1,3,2,4"**
* Nếu không có thao tác trong vòng 10s quay lai trạng thái ban dau.
* Khi nhập sai quay về trạng thái ban đầu

|  |
| --- |
| module Door\_System(  input wire [3:0] btn,  input wire clk, reset,  output reg y  );  reg [2:0] state, next\_state;  reg [11:0] seq; *// input seq*  reg [3:0] timer; *// 10 time count*  parameter S0 = 0 , S1 = 1 , S2 = 2 , S3 = 3, S4 = 4, error = 5;  parameter TIME\_OUT = 10;  *// reset*  always @(posedge reset)  if (reset)  begin  state <= S0;  seq <= 0;  y <= 0;  timer = 0;  end  *// time count*  always @(posedge clk)  if (timer == TIME\_OUT)  begin  state <= S0;  seq <= 0;  y <= 0;  timer = 0;  end  else if (btn != 0)  timer <= 0;  else  timer <= timer + 1;    *// FSM : thay đổi trạng thái*  always @(posedge clk)  case(state)  S0: if (btn[0] == 1)  begin  seq[2:0] = 1;  next\_state <= S1;  end else  next\_state <= S0;  S1: if (btn[2] == 1)  begin  seq[5:3] = 3;  next\_state <= S2;  end  else if (btn[1] || btn[3] || btn[0])  next\_state <= error;  S2: if (btn[1] == 1)  begin  seq[8:6] = 2;  next\_state <= S3;  end  else if (btn[2] || btn[3] || btn[0])  next\_state <= error;  S3: if (btn[3] == 1)  begin  seq[11:9] = 4;  next\_state <= S4;  end  else if (btn[1] || btn[2] || btn[0])  next\_state <= error;  error:  next\_state <= S0;  default: next\_state <= S0;  endcase  *// cập nhật trạng thái*  always @(posedge clk)  state <= next\_state;  *// output*  always @(state) begin  if(state == S3)  y = 1;  else  y = 0;  end  endmodule |



Hình : Ta thấy sau khi nhấn button 1,3,2,4 thì tín hiệu mở cửa (y = 1)

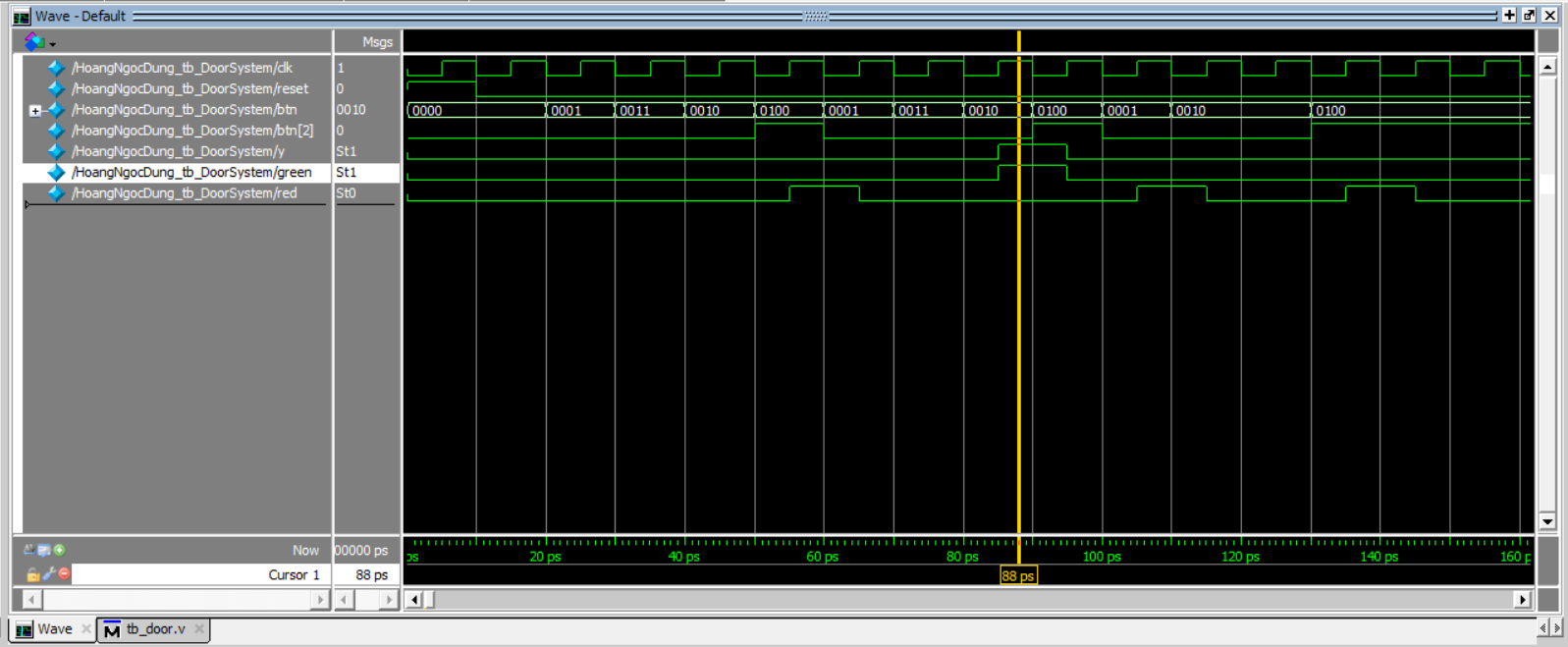
Mô tả bằng ngôn ngữ Verilog (full code door\_system)

|  |
| --- |
| module Door\_System(  input wire [3:0] btn,  input wire clk, reset,  output reg y,  output reg green\_led,  output reg red\_led  );  reg [2:0] state, next\_state;  reg [11:0] seq; // input seq  reg [3:0] timer; // 10 time count  reg [3:0] led\_timer; // 3s  parameter S0 = 0 , S1 = 1 , S2 = 2 , S3 = 3, S4 = 4, error = 5;  parameter TIME\_OUT  = 10, led\_on = 3, led\_blink = 1;  // reset  always @(posedge reset)  if (reset)  begin      state <= S0;      seq <= 0;      y <= 0;      timer <= 0;      green\_led <= 0;      red\_led <= 0;      led\_timer <= 0;  end  // time count  always @(posedge clk)  if (timer == TIME\_OUT)  begin      state <= S0;      seq <= 0;      y <= 0;      timer = 0;      green\_led <= 0;      red\_led <= 0;  end  else if (btn != 0)      timer <= 0;  else      timer <= timer + 1;    // change state  always @(posedge clk)  case(state)  S0: if (btn[0] == 1)  begin      seq[2:0] = 1;      next\_state <= S1;  end else      next\_state <= S0;  S1: if (btn[2] == 1)  begin      seq[5:3] = 3;      next\_state <= S2;  end  else if (btn[1] || btn[3] || btn[0])      next\_state <= error;  S2: if (btn[1] == 1)  begin      seq[8:6] = 2;      next\_state <= S3;  end  else if (btn[2] || btn[3] || btn[0])      next\_state <= error;  S3: if (btn[3] == 1)  begin      seq[11:9] = 4;      next\_state <= S3;  end  else if (btn[1] || btn[2] || btn[0])      next\_state <= error;  error: next\_state  <= S0;  default: next\_state  <= S0;  endcase  always @(posedge clk)      state <= next\_state;  // output  always @(state) begin  if (state == S3) begin // correct pass -> green light      y = 1;      green\_led = 1;          led\_timer <= 0;  end else begin      y = 0;      green\_led = 0;      led\_timer <= 0;  end  if (green\_led == 1) begin      if (led\_timer < led\_on)                  led\_timer <= led\_timer + 1;      else begin                  green\_led = 0;                  led\_timer <= 0;              end  end  if (state == error) begin              if (led\_timer < led\_blink) begin                  red\_led = (led\_timer % 2 == 0) ? 1 : 0;                  led\_timer <= led\_timer + 1;              end else begin                  red\_led = 0;                  led\_timer <= 0;              end          end else begin              red\_led = 0;              led\_timer <= 0;          end      end  endmodule |

Mô tả Verilog cho module dùng để kiểm tra thiết kế

|  |
| --- |
| module HoangNgocDung\_tb\_DoorSystem();  reg clk, reset;  reg [3:0] btn;  wire y, green, red;    initial begin  clk = 0;  btn = 0;  reset = 1;  #10 reset = 0;  #10 btn = 1;  #10 btn = 2;  #10 btn = 4;  #10 btn = 3;  #5 reset = 1;  #5 reset = 0;  #10 btn = 1;  #10 btn = 3;  #10 btn = 2;  #10 btn = 4;  #10 btn = 1;  #10 btn = 2;  #10 btn = 2;  #10 btn = 4;  end  always forever #5 clk = ~clk;  always forever #200 reset = ~ reset;  Door\_System uut (btn , clk, reset, y, green, red);  endmodule |

Kết quả mô phỏng



Hình : Sau khi reset. Hình ảnh mô tả tín hiệu đèn xanh sáng khi nhập đúng password = ‘1 3 2 4’. Thì y = 1 => green light sáng 3s. test Nhấn nút 1 rồi 3,2,4 => green\_led = 1 (Đèn xanh sáng). Và y = 1 (Mở cửa thành công).

Đèn xanh sẽ sáng trong 3 giây rồi tắt. Tuy nhiên trường hoejp đèn đỏ khi nhập sai vẫn chưa xử lý đúng nhấp nháy 3 lần