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

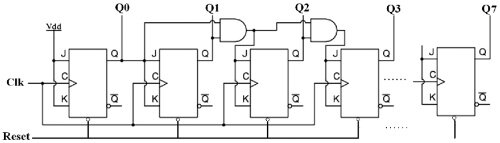
Chú ý: Sinh viên thay bằng QR code của mã số sinh viên (ví dụ: 23119012), có thể tham khảo tại https://barcode.tec-it.com)

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*

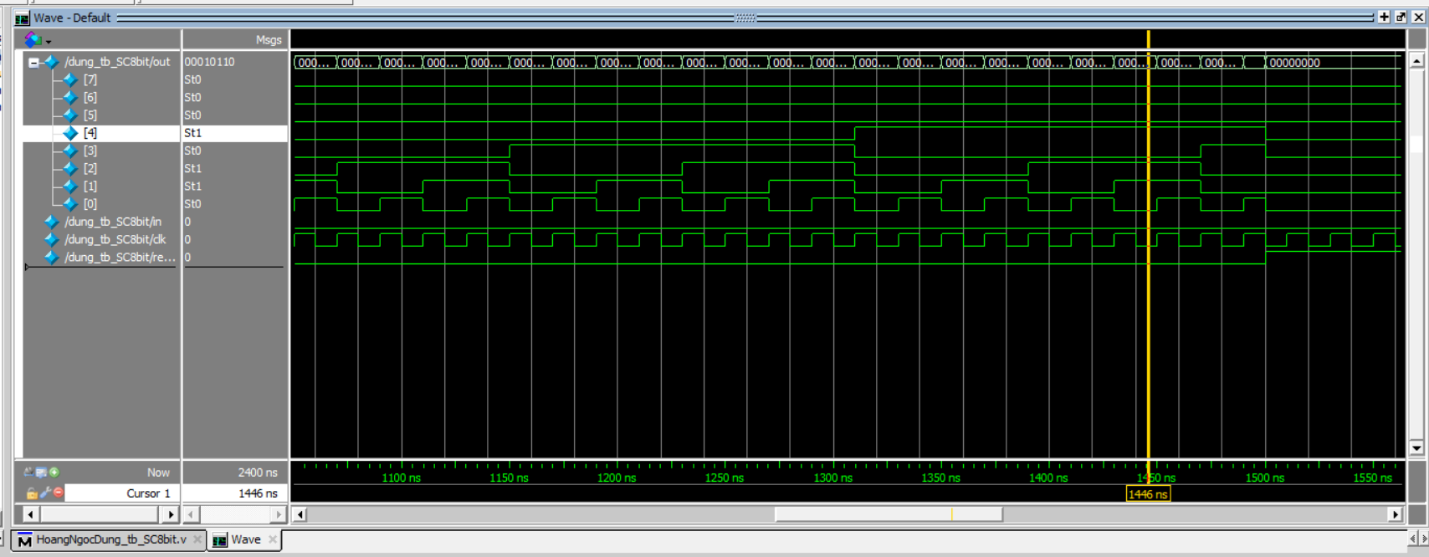
1. Thiết kế và mô phỏng mạch đếm đống bộ 8 bit sử dụng phương pháp thiết kế mạch tuần tự



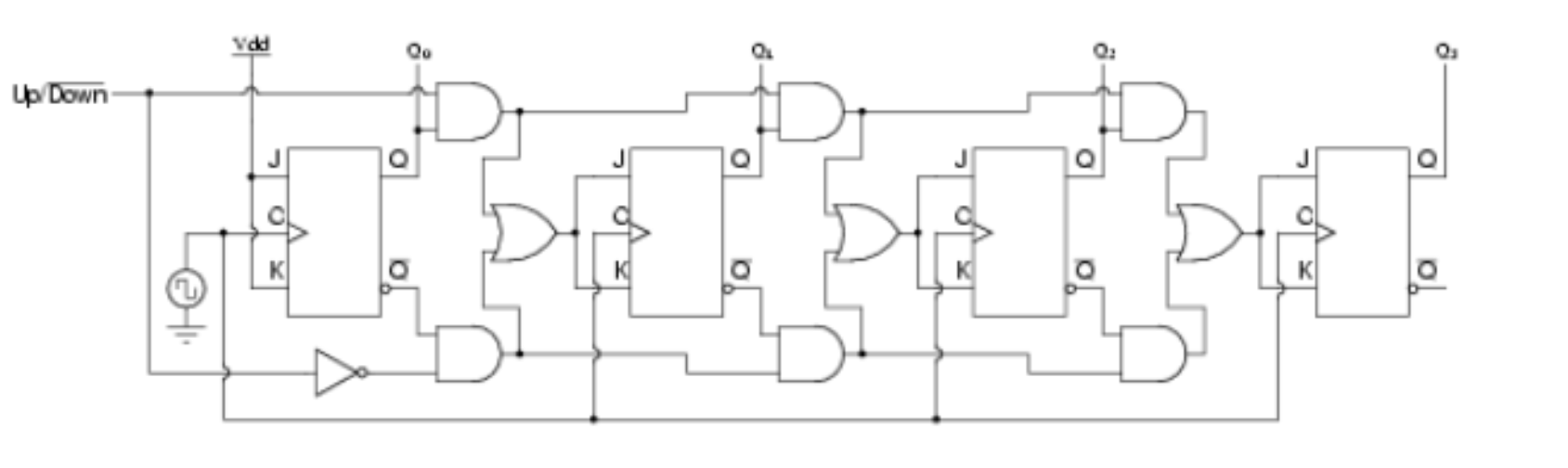
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CLK | Q7 (MSB) | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 (LSB) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ... | ... | ... | ... | ... | ... | ...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

|  |
| --- |
| 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 |

|  |
| --- |
| `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 |



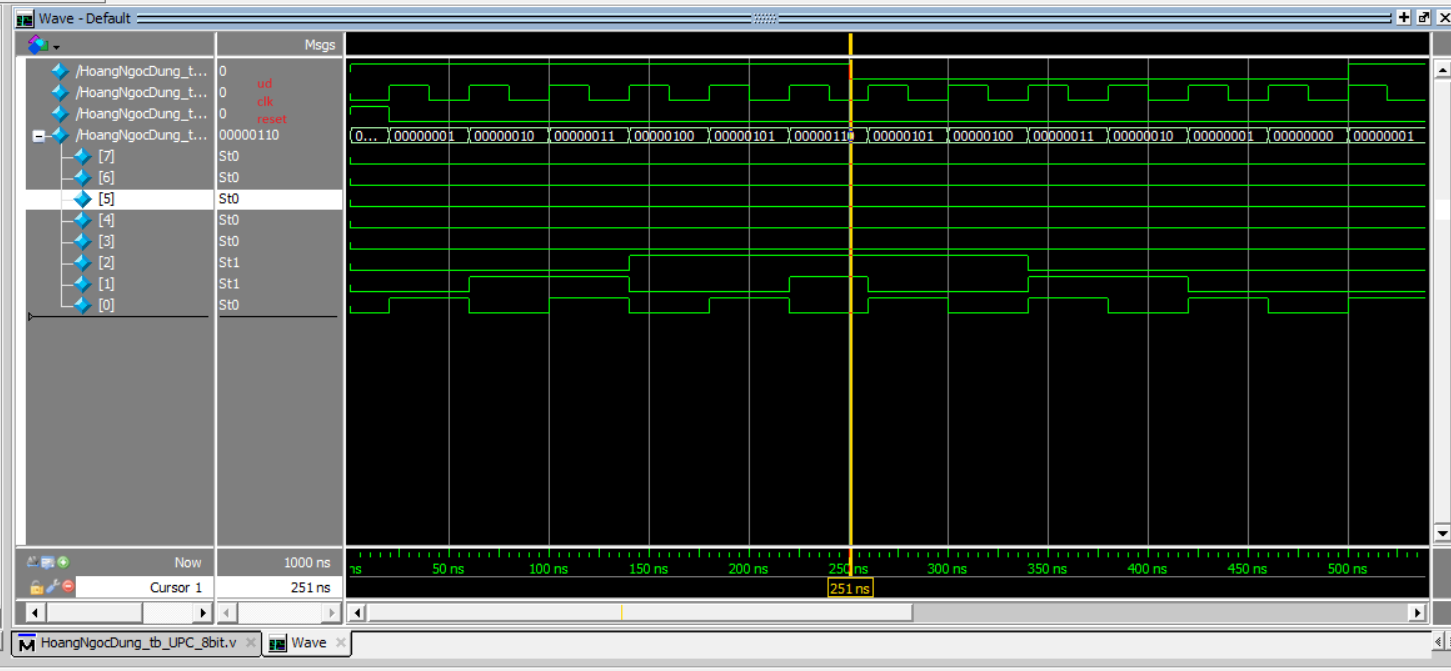
1. Thiết kế và mô phỏng kiểm chứng mạch đếm đồng bộ 8 bit, có tín hiệu UD cho phép đếm lên/đếm xuống



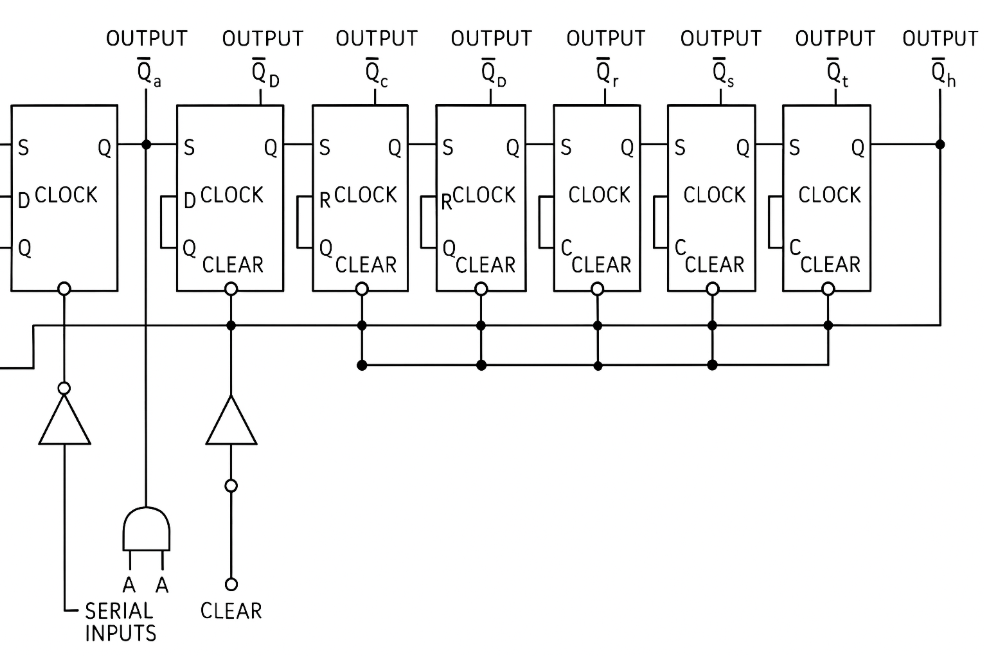
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Up/Down | Current State (Q7-Q0) | Next State (Q7-Q0) | J7 K7 | J6 K6 | ... | J0 K0 |
| 1 (Up) | 0 | 1 | 0 X | 0 X | ... | 1 X |
| 1 (Up) | 1 | 10 | 0 X | 0 X | ... | 1 X |
| 1 (Up) | 1111111 | 10000000 | 1 X | 1 X | ... | 1 X |
| 1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

|  |
| --- |
| 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 |

|  |
| --- |
| `timescale 1ns/1ns  // Định nghĩa đơn vị thời gian là 1ns, độ chính xác 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 |



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



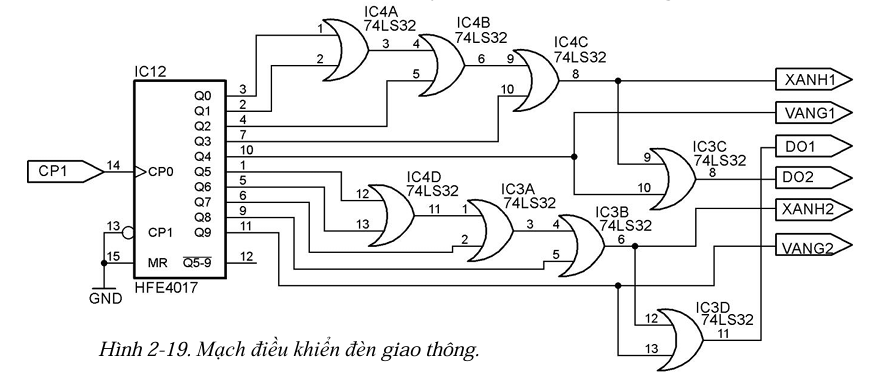
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chu kỳ Clock | s\_in (giải sử) | r\_reg (Binary) | r\_reg (Hex) | Giải Thích |
| 0 (Khởi tạo) | X | 0 | 0 | Trạng thái ban đầu |
| 1 | 1 | 10000000 | 80 | Bit 1 dịch vào MSB |
| 2 | 0 | 1000000 | 40 | Bit 0 dịch vào, các bit khác dịch phải |
| 3 | 1 | 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 |

|  |
| --- |
| 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 |

|  |
| --- |
| `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 |

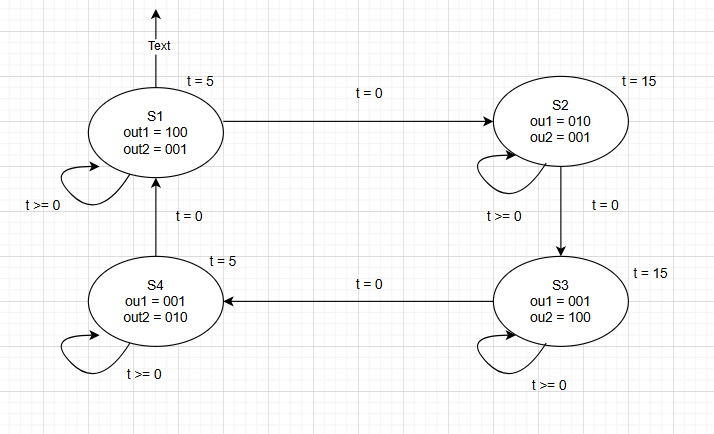


1. Thiết kế và mô phỏng mô hình máy trạng thái cho bài toán điều khiển đèn giao thông, các tín hiệu ngõ ra giải sử chỉ là các ngõ điều khiển đèn xanh, vàng đỏ



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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 : Sơ đồ nguyên lý dùng mô hình máy trạnh thái

|  |
| --- |
| 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 |

|  |
| --- |
| `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 |

