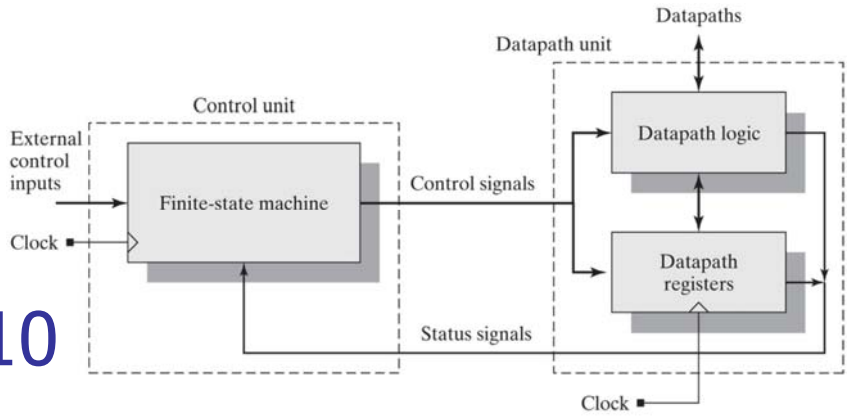


Topic 10

Finite State Machines & (ASM Charts)-(2/2)

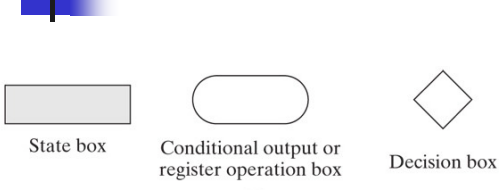
國立高雄大學電機工程學系

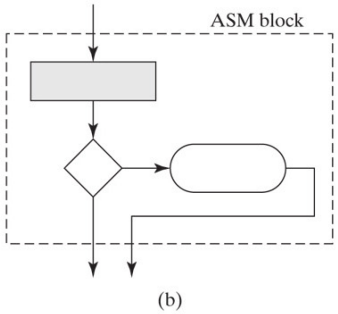
陳春僥

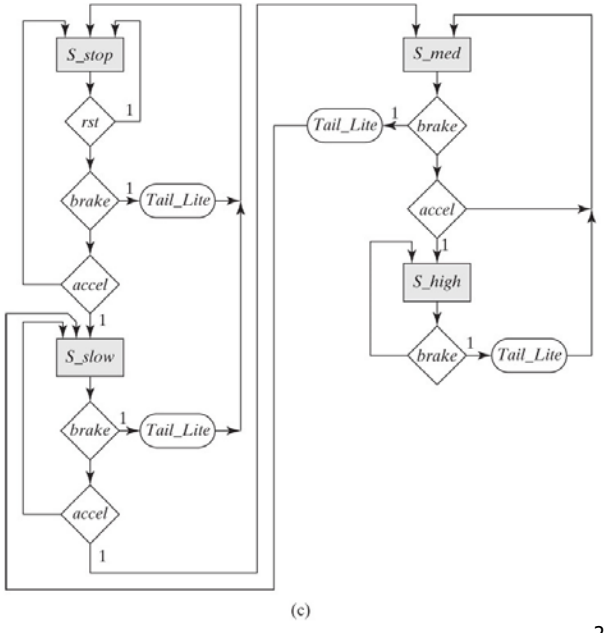
cychen@ieee.org

June 17, 2021

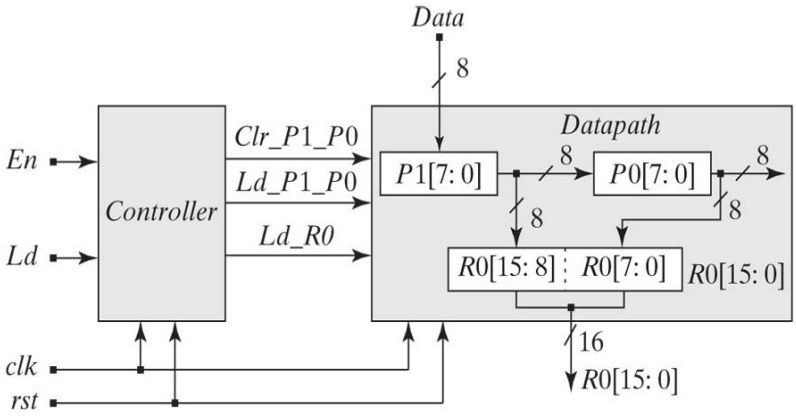
Algorithmic State Machine (ASM) Chart



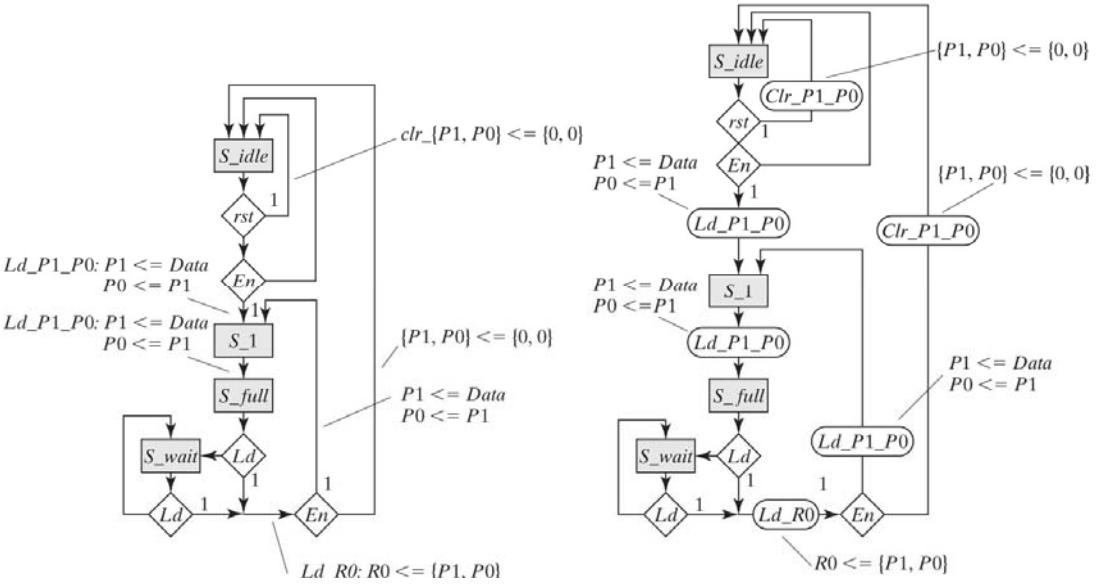




Two-Stage Pipelined Register



Two-Stage Pipelined Register



Two-Stage Pipelined Register


```
module pipe_2stage (R0, Data, En, Ld, clk, rst);
  output R0;
  input Data;
  input En, Ld, clk, rst;

  reg [1:0] state, next_state;
  parameter S_idle = 0;
  parameter S_1 = 1;
  parameter S_full = 2;
  parameter S_wait = 3;

  wire [7:0] Data;
  reg  Ld_R0;
  reg  Clr_P1_P0;
  reg  Ld_P1_P0;
  reg  [7: 0] P0, P1;
  reg  [15:0] R0;

  always @ (posedge clk)
     if (rst)
        state <= S_idle;
     else
        state <= next_state;

  always @ (state, Ld, En, rst)
  begin
     Ld_R0 = 0;
     Clr_P1_P0 = 0;
     Ld_P1_P0 = 0;
```

1

```
  case (state)
     S_idle:   if (rst)
               begin
                  next_state = S_idle;
                  Clr_P1_P0 = 1;
               end
               else if (En)
               begin
                  next_state = S_1;
                  Ld_P1_P0 = 1;
               end
               else
                  Clr_P1_P0 = 1;
     S_1:      begin
                  next_state = S_full;
                  Ld_P1_P0 = 1;
               end
     S_full: if (Ld == 0)
                  next_state = S_wait;
               else
               begin
                  Ld_R0 = 1;
                  if (En)
                  begin
                     next_state = S_1;
                     Ld_P1_P0 = 1;
                  end
                  else
                  begin
                     next_state = S_idle;
                     Clr_P1_P0 = 1;
                  end
               end
```

2

```
     S_wait:   if (Ld)
               begin
                  Ld_R0 = 1;
                  if (En)
                  begin
                     next_state = S_1;
                     Ld_P1_P0 = 1;
                  end
                  else
                  begin
                     next_state = S_idle;
                     Clr_P1_P0 = 1;
                  end
               end
        default:  next_state = 2'bx;
     endcase
  end
```

3

```
  always @ (posedge clk)
  begin
     if (Clr_P1_P0)
        begin
           P1 <= 0;
           P0 <= 0;
        end
     if (Ld_R0)
        begin
           R0 <= {P1, P0};
        end
     if (Ld_P1_P0)
        begin
           P1 <= Data;
           P0 <= P1;
        end
  end
endmodule
```

4

```
module t_pipe_2stage;
  wire [15: 0] R0;
  reg [7: 0] Data;
  reg  En, Ld, clk, rst;

  pipe_2stage M0 (R0, Data, En,
                  Ld, clk, rst);

  initial #500 $finish;
  initial
  begin
     clk = 0;
     forever #5 clk = ~clk;
  end
  initial
  begin
     Data = 8'H55;
     forever @ (negedge clk)
        Data = Data + 1;
   end
  initial
  fork
    #10 rst = 0;
    #20 rst = 1;
    #40 rst = 0;
    #300 rst = 1;
    #350 rst = 0;
    #20 En = 0;
    #70 En = 1;
    #80 En = 0;
    #200 En = 1;
    #20 Ld = 0;
    #140 Ld = 1;
    #150 Ld = 0;
    #400 Ld = 1;
  join
endmodule
```

5

HDL Design & Simulation — Topic 10: Finite-State Machines & (ASM Charts)

Two-Stage Pipelined Register