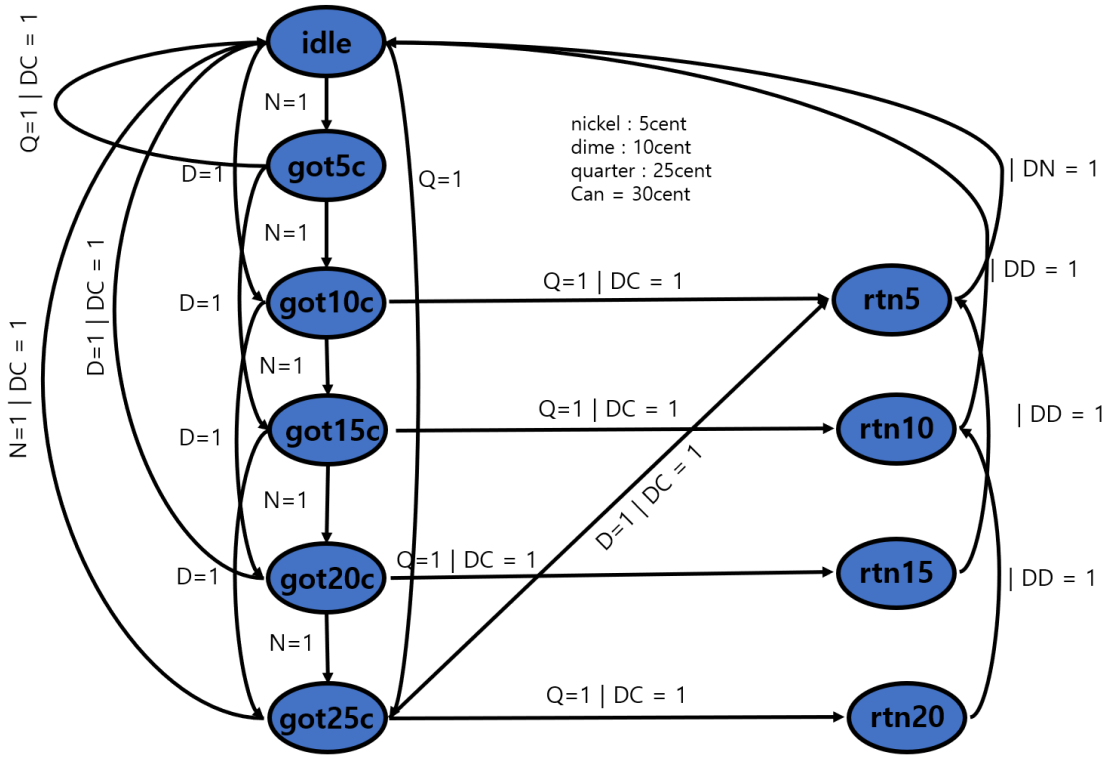
# Microprocessor Interface 시험범위

## 1. 강의 6 Design Vending Machine (자판기) [p15]

- **state diagram 그리기**




## 2. 강의 6 Mealy FSM(Finity State Machine) [p16~17]

- **always 2개 구문 외우기**

```
    // 순차적으로 state를 정기적으로 업데이트 해준다
// posedge : 0에서 1이 되는 순간
// negedge : 1에서 0이 되는 순간
always @ (posedge clk or negedge reset) // Clock 마다, reset
    if (!reset) state <= IDLE; //reset이 0이되면 state를 IDLE로 바꿔줌
        else state <= next; //나머지의 경우에는 state를 다음으로 진행시켜줌

//state업데이트에 대한 비동기 초기화
```

```
//state Diagram 대로 코드짜기
always @ (state or N or D or Q) //무엇이 무엇이 들왔을까?
    begin
        DC = 0; DN = 0; DD = 0; // 초기화
        case (state)
            IDLE: //아무일없으면 다음 상황 확인
                if (Q)
                    next = GOT_25c; // Q=1이면 quarter가 들어온것
                else if (D)
                    next = GOT_10c; // D=1이면 dime이 들어온것
                else if (N)
```

```
                next = GOT_5c; // N=1이면 quarter가 들어온것
            else
                next = IDLE; // 아니면 넌 백수임
        GOT_5c: // Nickel이 들어온 상황이다
            if (Q)
                begin //25센트가 들어왔으면 음료를 주고 Idle;
                    DC = 1; next = IDLE;
                end
            else if (D)
                next = GOT_15c;
            else if (N)
                next = GOT_10c;
            else
                next = GOT_5c;
        GOT_10c:
            if (Q)
                begin
                  DC = 1; next = RETURN_5c;
                end
             else if (D)
                 next = GOT_20c;
             else if (N)
                 next = GOT_15c;
             else
                 next = GOT_10c;
         GOT_15c:
             if (Q)
                 begin
                    DC = 1; next = RETURN_10c;
                end
            else if (D)
                next = GOT_25c;
            else if (N)
                next = GOT_20c;
            else
                next = GOT_15c;
         GOT_20c:
             if (Q)
                 begin
                    DC = 1; next = RETURN_15c;
                end
              else if (D)
                  begin
                     DC = 1; next = IDLE;
                  end
              else if (N)
                  next = GOT_25c;
              else
                  next = GOT_20c;
        GOT_25c:
            if (Q)
                begin
                    DC = 1; next = RETURN_20c;
                end
            else if (D)
                begin
                    DC = 1; next = RETURN_5c;
            end
```

```
                else if (N)
                    begin
                        DC = 1; next = IDLE;
                    end
                else
                    next = GOT_25c;
            RETURN_20c:
                begin
                    DD = 1;
                    next = RETURN_10c;
                end
            RETURN_15c:
                begin
                    DD = 1;
                    next = RETURN_5c;
                end
            RETURN_10c:
                begin
                    DD = 1;
                    next = IDLE;
                end
            RETURN_5c:
                begin
                    DN = 1;
                    next = IDLE;
                end
            default:
                next = IDLE;
            endcase
        end
endmodule
```

## 3. 강의 7 Non blocking, Blocking [과제]

- **과제 내용 숙지**
- > Blocking : 해당 구문이 등장하는 시점에 할당이 이루어짐(=)
  >
  > Non Blocking : CLK 마지막 주기에 할당이 이루어짐 (<=)

```
module block_swap(x_out,y_out,clk,x,y);
  input clk,x,y;
  output x_out,y_out;
  reg x_val,y_val, temp;

  always @(posedge clk) begin
     x_val = x;
     y_val = y;
       temp = x_val;
       x_val = y_val;
       y_val = temp;
  end

  assign x_out = x_val;
  assign y_out = y_val;
```

```
endmodule;
//Blocking때는 순차적으로 바로 적용되어 교차가 이루어지지 않는다. 그래서 임시변수를 만들어 swap해야한다.
```

```
//Non Blocking때는 정상적으로 교차가 이루어진다.
module nonblock_swap(x_out,y_out,clk,x,y);
  input clk,x,y;
  output x_out,y_out;
  reg x_val,y_val;

  always @(posedge clk) begin
     x_val = x;
     y_val = y;
       x_val <= y_val;
       y_val <= x_val;
  end

  assign x_out = x_val;
  assign y_out = y_val;

endmodule;
//Non Blocking때는 일이 다 끝난후 할당되어 정상적으로 교차가 이루어진다.
```

```
module block_swap_tb;
  reg x,y,clk;
  wire x_out,y_out;


  block_swap bswap(x_out,y_out,clk,x,y);

  always #1 clk = !clk;

  initial begin
    x=0;y=0;clk=0;
    #4 x=0;y=1;
    #4 x=1;y=0;
    #4 x=1;y=1;
    #4 x=0;y=1;
  end

endmodule
```

## 4. 강의 7 Delay-based timing [과제]

- Regular Delay Control

```
//Delay 실행 -> 우측 계산후 좌변에 할당
parameter latency = 20;
parameter delta = 2;
reg x, y, z, p, q;

initial
    begin
```

```
        x = 0;
        #10 y = 1;
        #latency z = 0;
        #(latency + delta) p = 1;
        #y x = x + 1;
        #(4:5:6) q = 0;
```

- Intra assignment Delay

```
//우측 계산후 -> Delay -> 좌측에 할당
reg x, y, z;
initial
    begin
        x = 0;   z = 0;
        y = #5  x + z;
    end
initial
    begin
        x = 0; z = 0;
        temp_xz = x + z;
        #5    y = temp_xz;
    end
//intra delay가 변수로 이루어질경우 delay가 시작하고 나서의 변수값을 가져온다
```

## 5. 강의 7 Generate Blocks, Parallel block[과제]

```
module BeginEnd(clk,y_out,Z_out,W_out);
  input clk;
  output y_out,Z_out,W_out;
  reg y_val,Z_val,W_val;
  always @(posedge clk)
    begin
      #10 y_val = 1'b1;
      #20 Z_val = 1'b1;
      #40 W_val = 1'b1;
    end
  assign y_out = y_val;
  assign Z_out = Z_val;
  assign W_out = W_val;
endmodule;
# Sequential Delay이므로 10 -> 30(+20) -> 70(+40) 시점에서 1로 증가한다.
```

```
module forkJoin(clk,y_out,Z_out,W_out);
  input clk;
  output y_out,Z_out,W_out;
  reg y_val,Z_val,W_val;
  always @(posedge clk)
  fork
    #10 y_val = 1'b1;
    #20 Z_val = 1'b1;
    #40 W_val = 1'b1;
```

```
  join
  assign y_out = y_val;
  assign Z_out = Z_val;
  assign W_out = W_val;
endmodule;
# Parellar Delay이므로 10 -> 20 -> 40 시점에서 1로 증가한다.
```

```
module forkJoin_tb;
  reg a,b,clk;
  wire y_out,Z_out,W_out;
  always #1 clk = !clk;
  initial begin
    a=0;b=0;clk=0;
  end
  forkJoin _forkJoin(clk,y_out,Z_out,W_out);
endmodule
```

## 6. 강의 8 Function Task [p10]

- **Example 8-7: Parity Calculation**

```
//패리티 비트 계산
module parity;
    ...
    reg [31:0] addr;
    reg parity;
    always @(addr)
        begin
            parity = calc_parity(addr);
            $display("Parity calculated = %b", calc_parity(addr) );
        end
    function calc_parity;
        input [31:0] address;//input 매개변수
        begin// begin으로 시작
            calc_parity = ^address;//XOR계산
        end
    endfunction
    ...
endmodule
```

## 7. 강의 8 Automatic Function [p13]

- **Example 8-10: Recursive (Automatic) Functions**

```
module top;
    ...
    function automatic integer factorial;//재귀 함수 선언 호출시 반환값이 자동으로 넘
어감
        input [31:0] oper;//input 매개변수
        integer i;
        begin
            if (oper >= 2)
```

```
                factorial = factorial(oper -1) * oper;// 재귀함수 호출
            else
                factorial = 1 ;
        end
    endfunction
    integer result;
    initial
        begin
            result = factorial(4);//Recursizr function 호출
            $display("Factorial of 4 is %0d", result);
        end
    ...
endmodule
```

## 8. 강의 8-9 assign구문 왜 있을까? [p23 밑에서 5번째줄]

- **왜 저 구문이 필요할까?**

```
// UART_TxD
//indicated the rising edge of bit clock
// 정기적으로 bclk_delayed주기로 bclk_rising 바꾸어줘 always문에 신호를 전송한다.
// bit 전송 속도를 synchronize해주기 위해 일정한 주기로 Bclk이 tick되도록한다.
// ~bclk_delayed: 비트 뒤집기
assign  bclk_rising = bclk & (~bclk_delayed);
```

## 9. 강의 8-9 UART Transmitter [p23,25]

- **always구문 2개 숙지**

한개는 State를 업데이트하고 모니터링하기위해 존재하고

한개는

## 10. 강의 8-9 UART Receiver assign구문 외우기 [p31 마지막줄]

- **왜 저 구문이 필요할까?**

```
// UART_RxD
//indicated the rising edge of bit clock
// bit 전송 속도를 synchronize해주기 위해 일정한 주기로 Bclk이 tick되도록한다.
// ~BclkX8_Dlayed: 비트 뒤집기
assign BclkX8_Rising = BclkX8 & (~BclkX8_Dlayed);
```

## 11. 강의 8-9 UART Receiver [p32,35]

- **always구문 2개 숙지 type구분**

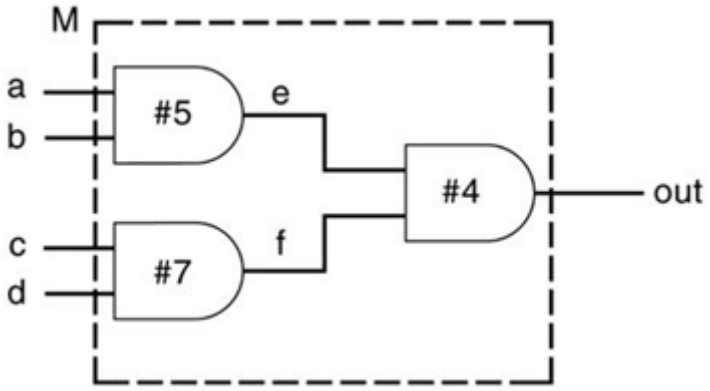
**always @ (state or RxD or RDRF or ct1 or ct2 or BclkX8_Rising)**

은 bit의 수신상태에 따라 state를 바꿔주눈 역할을 한다.

**always @ (posedge sysclk or negedge rst_b)**

은 receive된 bit가 있는지 detect하고 수신한다.

## 12. 강의 10 Distribute Delay



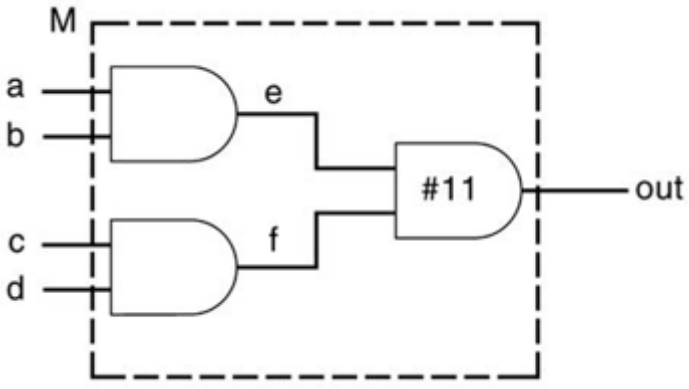

```
//Distribute Delay by gate module
module M (out, a, b, c, d);
    output out;
    input a, b, c, d;
    wire e, f;

    and #5 a1(e, a, b);
    and #7 a2(f, c, d);
    and #4 a3(out, e, f);
endmodule
```

```
//Distribute Delay by data flow
module M (out, a, b, c, d);
    output out;
    input a, b, c, d;
    wire e, f;

    assign #5 e = a & b;
    assign #7 f = c & d;
    assign #4 out = e & f;
endmodule
```

## 13.강의 10 Lumped Delay



```
module M (out, a, b, c, d);
    output out;
    input a, b, c, d;
    wire e, f;

    and a1(e, a, b);
    and a2(f, c, d);
    and #11 a3(out, e, f);
    //delay only on the output gate
endmodule
```