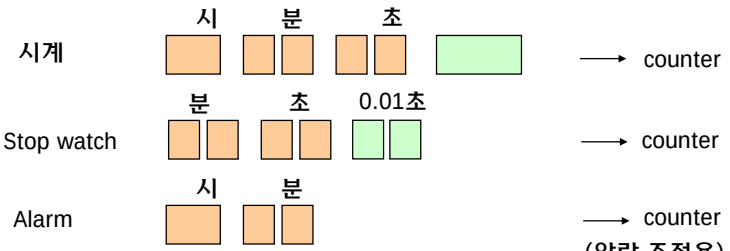
Design Example

Alarm 시계

디지털 시계

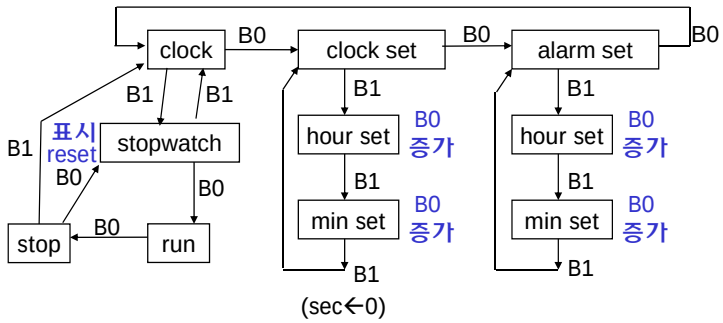
- 기능
  - 시계 — 시, 분, 초 표시 (날짜는 선택사항)
  - stop watch — 0.01초까지 측정 (분, 초, 1/100초 표시)
  - alarm — 시, 분 지정
- 입출력
  - 출력: 7-segment LED를 사용하여 출력
  - 입력: 버튼을 사용하여 모드 설정, 시간 설정

- 동작
  - 시계와 Stop Watch는 동작모드에서 counter를 사용하여 증가시킴
  - 시계와 alarm 시간 설정도 counter를 사용하여 조정
  - 시계와 Alarm의 시, 분을 비교하여 일치하면 alarm 신호 표시 (소리를 울리거나 LED 표시에 변화를 주어서 표시함)
  - 시계, Alarm은 시, 분을 지정할 수 있으며 시계설정이 완료되면 초는 0부터 시작



상태도

- 상태도 (입력버튼을 2개 사용)



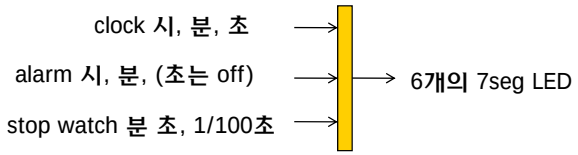
- stop watch 동작
  - 버튼 B0: stop/reset/run
  - 버튼 B1: clock으로 되돌아감(run상태에서는 적용되지 않음)

- 시계, 알람 시간 조정
  - 현재 조정되는 단계(시 또는 분)는 출력을 깜박이게 하여 구분이 되도록 함
  - 버튼을 길게 누르더라도 1번만 증가하도록 함
    (또는 일정시간이 경과하면 계속적으로 증가되도록 함)


```
/* state machine */
    always @(posedge clock or posedge reset) begin
        if (reset) state <= S0;
        else begin
            case (state)
             S0: if (B0) state <= S10;          // S0: watch
                 else if (B1) state <= S30;

             S10: if (B0) state <= S20;         // S10: set clock
                  else if (B1) state <= S11;
             S11: if (B1) state <= S12;         // S11: adjust clock hour
             S12: if (B1) state <= S10;         // S12: adjust clock min

             S20: if (B0) state <= S0;          // S20: set clock alarm
                  else if (B1) state <= S21;
             S21: if (B1) state <= S22;         // S21: adjust alarm hour
             S22: if (B1) state <= S20;         // S22: adjust alarm min

             S30: if (B0) state <= S31;         // S30: stop watch (reset)
                  else if (B1) state <= S0;
             S31: if (B0) state <= S32;         // S31: start stop-watch
             S32: if (B0) state <= S30;         // S32: stop stop-watch
                  else if (B1) state <= S0;
             default : state <= S0;
            endcase
        end
    end
```


- 출력
  - 분과 초는 그대로 출력
  - 시는 0-23까지의 2진수를 2자리 BCD코드로 변환 후에 출력

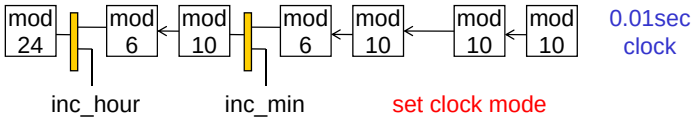
  - 7seg LED decoder를 사용




```
/* binary2bcd - for hour */
always @(hour) begin
   if (hour < 10) begin hr10 = 0; hr=hour; end
   else if (hour < 20) begin hr10 = 1; hr=hour+6; end
   else begin hr10 = 2; hr=hour+12; end
end
```

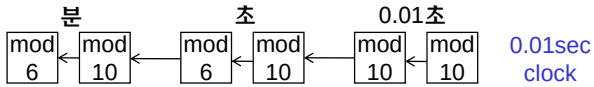
데이터 경로

- clock



- alarm register

- stop watch



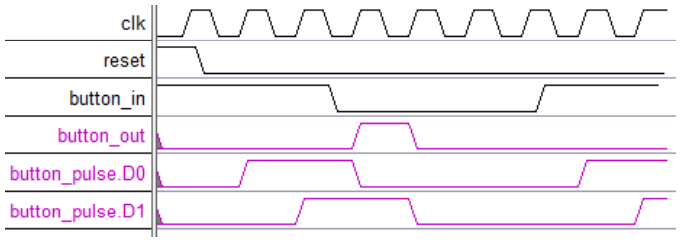

```
module counterN(clk, reset, enable, qout, tc);
    parameter N=24, M=5;
    input clk, reset;
    input enable;
    output [M-1:0] qout;
    output tc;

    reg [M-1:0] qout;

    assign tc = (qout==N-1) & enable;
    always @(posedge clk or posedge reset) begin
        if (reset) qout <= 0;
        else if (enable) begin
            if (qout==N-1) qout = 0;
            else qout <= qout + 1;
        end
    end
endmodule
```



```
counterN #(10,4) u1 (clk, reset_sec, en_sec, sec, tc_sec);
counterN #(6,3)  u2 (clk, reset, en_sec10, sec10, tc_sec10);
counterN #(10,4) u3 (clk, reset, en_min, min, tc_min);
counterN #(6,3)  u4 (clk, reset, en_min10, min10, tc_min10);
counterN #(24,5) u5 (clk, reset, en_hour, hour, );

assign en_sec = (set_clock) ? 0 : 1;
assign en_sec10 = tc_sec ;
assign en_min = (set_clock) ? inc_min : tc_sec10;
assign en_min10 = tc_min;
assign en_hour = (set_clock) ? inc_hour : tc_min10;
assign reset_sec = reset | set_clock;
```


Button 입력을 pulse로 변환

- button이 눌려진 동안 button 입력이 여러 clock 지속될 경우에도 1번의 button 입력이 들어온 것으로 처리해야 함
  - button 입력이 1 clock 동안만 제공되도록 변환하는 회로가 필요함
- button 입력을 1 clock 폭을 갖는 pulse로 변환하는 회로



- 지속되는 button 입력을 일정 간격마다 pulse를 발생시키는 회로
  - 한 번 button을 눌러서 연속적으로 적용하는 데 사용 가능

module button_pulse

```
module button_pulse(clk, reset, button_in, button_out);
	input clk, reset;
	input button_in;
	output button_out;

	reg D0, D1;

	always @(posedge clk or posedge reset) begin
		if (reset) begin D0 <= 0; D1 <= 0; end
		else begin D0 <= button_in; D1 <= D0; end
	end
	assign button_out = ~D0 & D1;
endmodule
```