Finite State Machine & Miniwatch Example

Hsi-Pin Ma

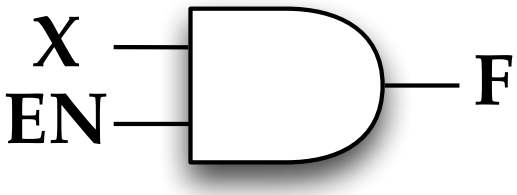
https://eeclass.nthu.edu.tw/course/18498
Department of Electrical Engineering
National Tsing Hua University

Clocks

- Use signal (100MHz) from crystal (W5)
- Do not use gated clock
  - ex: en & clk (X)
  - For large blocks, use PLL IP
- Various clock frequencies in labs
  - 100-Hz for Debounce circuits
  - 1-Hz for second clock display
  - faster clock for push-button-controlled FSM (~10-Hz) - one pulse generation

"Turn off" a Block

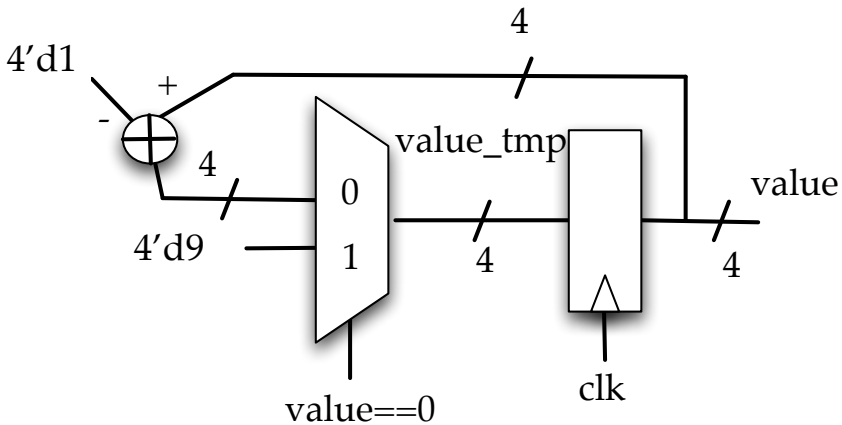
- Enable control (with AND gate) for combinational logics (block input)



$$F = \mathrm{EN} \cdot X$$

E	X	F
0	0	0
0	1	0
1	0	0
1	1	1

- Use MUXs in front of DFFs for unchanged states
  - Clock gating is NOT preferred



Finite State Machine

- Derive the state diagram
- Determine # of DFFs (for N states)
  - Use $\lceil \log_2 N \rceil$ numbers of DFFs for binary-coded state
  - If FSM is Moore model, use DFFs for outputs
  - Write Verilog codes for DFFs
- Use combinational logics for state transitions and output functions (Use ***case*** statement)

Bad Coding Style: Inferred Latches in Combinational Logics

- Incomplete case statement
- To avoid
  - Make sure to have ***default*** case
  - Or always specify the default value in the beginning of the always block

```
always @*
begin
  case (alu_control)
    2'd0: y = x + z;
    2'd1: y = x - z;
    2'd2: y = x * z;
 endcase
end
```

```
always @*
begin
  case (alu_control)
    2'd0: y = x + z;
    2'd1: y = x - z;
    2'd2: y = x * z;
    default: y = 0;
  endcase
end
```

```
always @*
begin
  y=0;
  case (alu_control)
    2'd0: y = x + z;
    2'd1: y = x - z;
    2'd2: y = x * z;
  endcase
end
```

Bad Coding Style: Inferred Latches in Combinational Logics

```
always @*
begin
  if (alu_control==2'b00)
    y=x+z;
  else if (alu_control==2'b01)
    y=x-z;
  else if (alu_control==2'b10)
    y=x*z;
end
```

```
always @*
begin
  y=0;
  if (alu_control==2'b00)
    y=x+z;
  else if (alu_control==2'b01)
    y=x-z;
  else if (alu_control==2'b10)
    y=x*z;
end
```

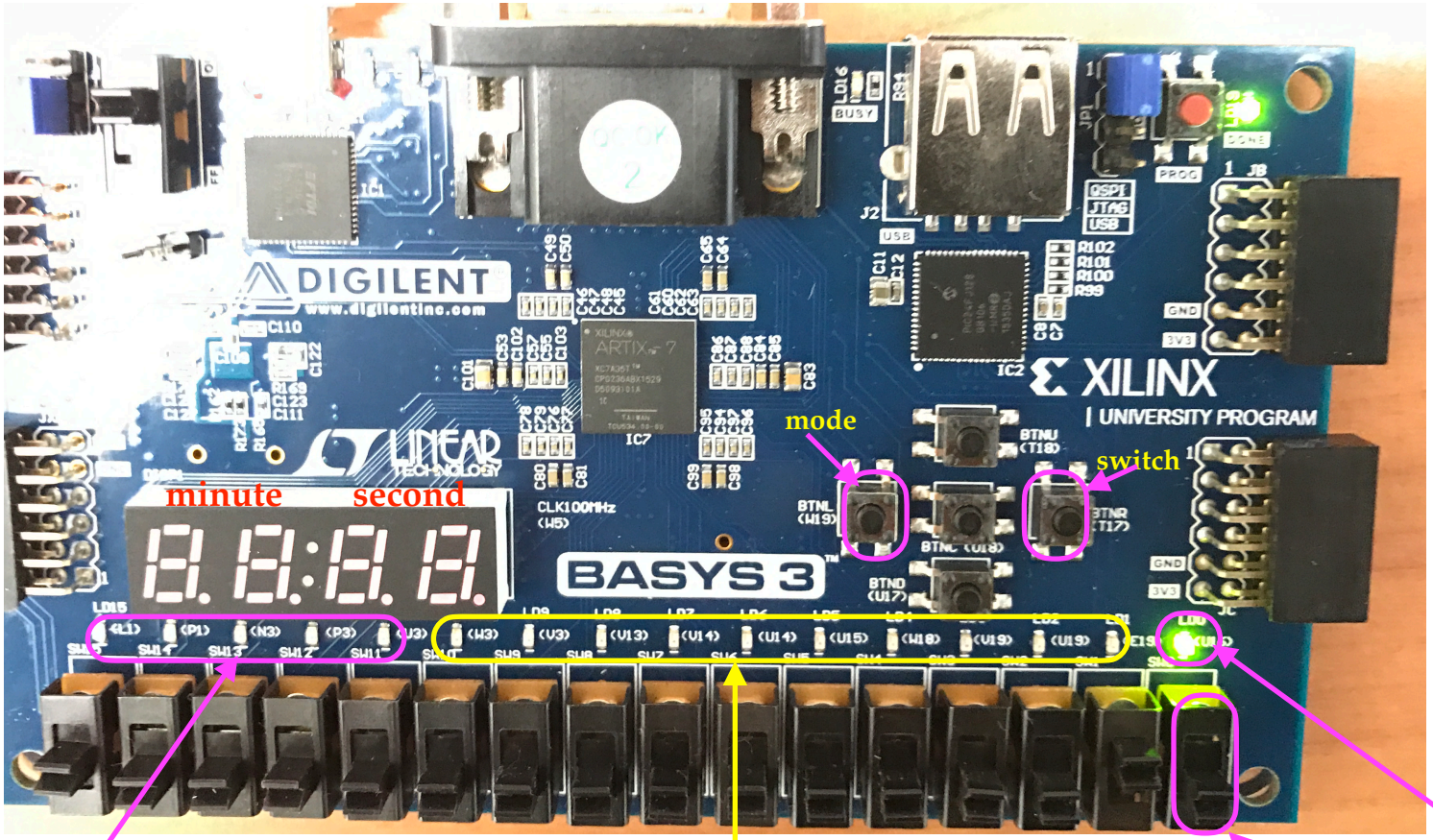
```
always @*
  if (alu_control==2'b00)
    y=x+z;
  else if (alu_control==2'b01)
    y=x-z;
  else if (alu_control==2'b10)
    y=x*z;
  else
    y=0;
```

Miniwatch Example

Specification

- Function
  - Normal function
    - minute-second time display
    - timer supporting up to 59:59
    - alarm for a certain minute
  - Tuning function for all normal function
  - Using two push buttons for control (one additional reset with DIP switch)

I/O Description



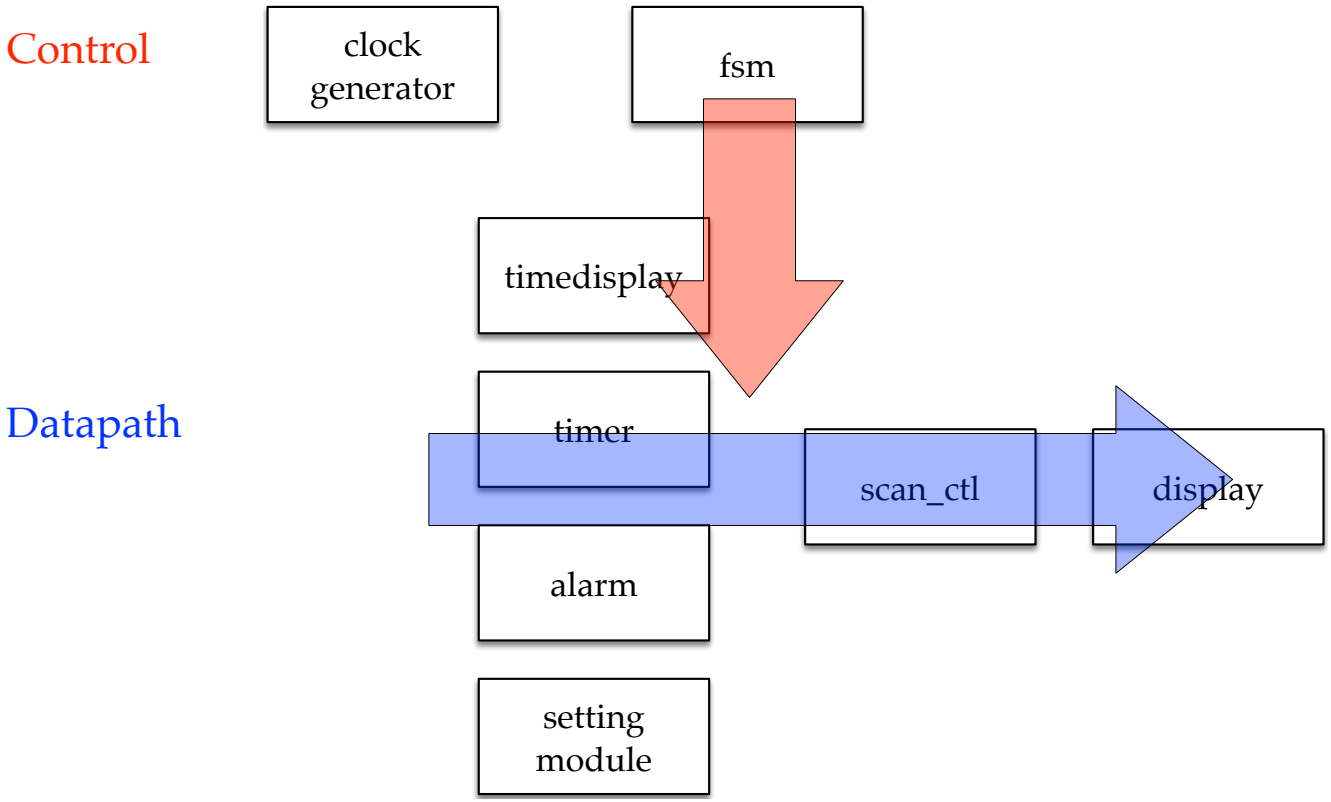
state

10 leds lighted for alarm or stopwatch ending

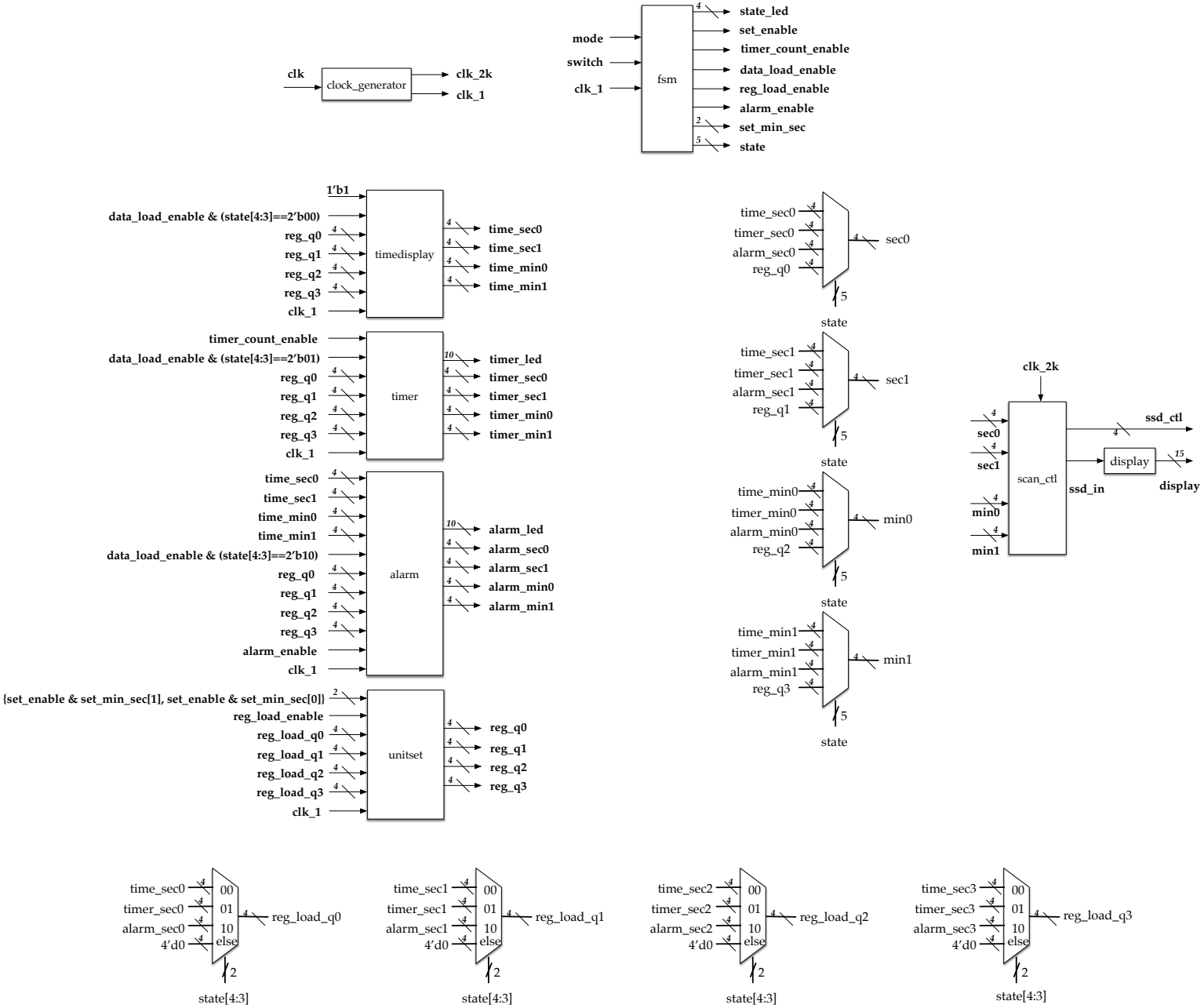
reset

alarm mode

Function Sketch

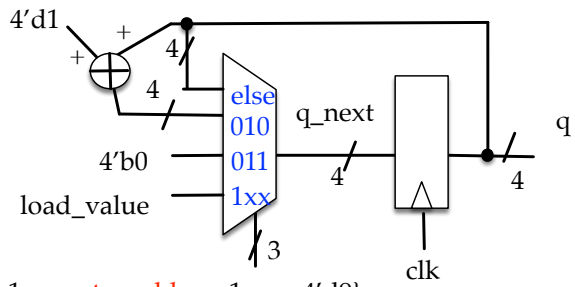


Block Diagram

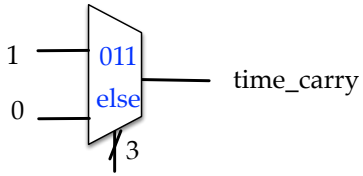


Timedisplay

- Based on counterx

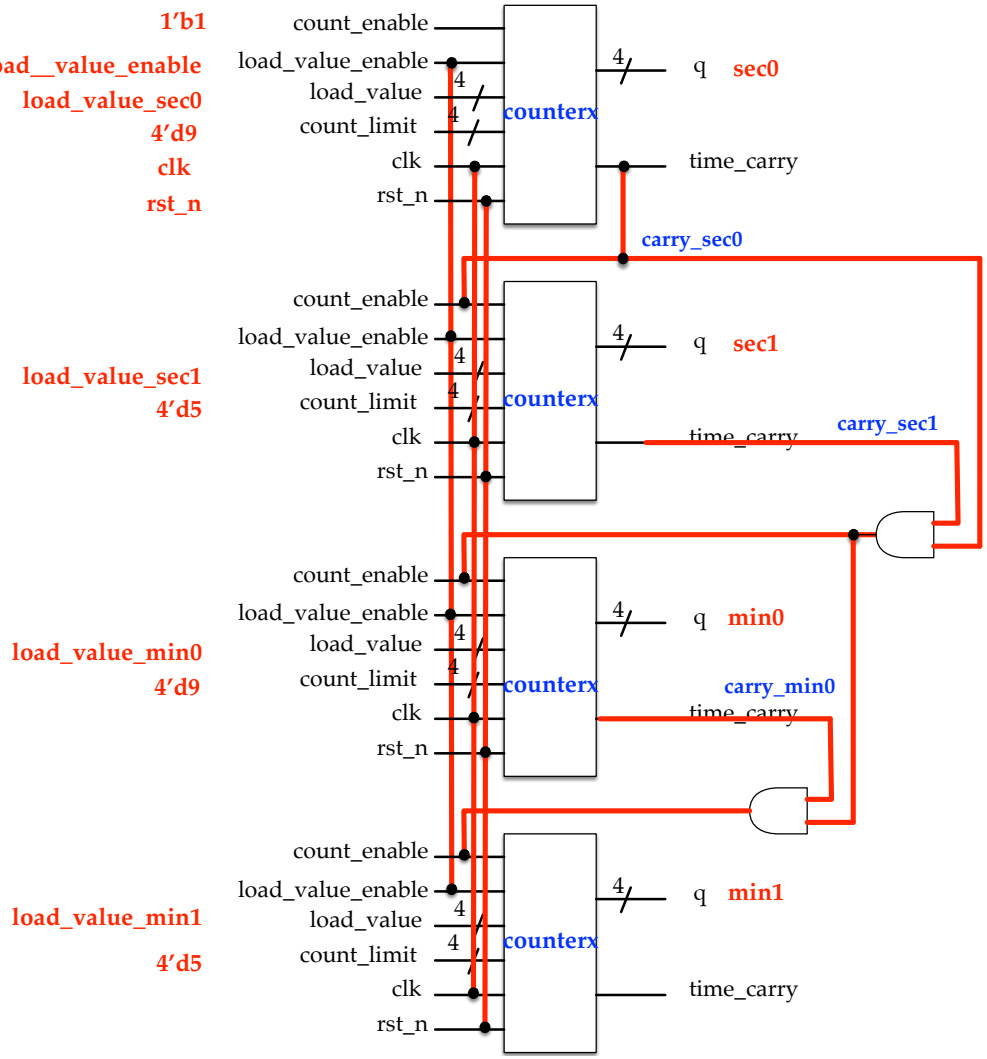


{load_value_enable==1, count_enable ==1, q==4'd9}



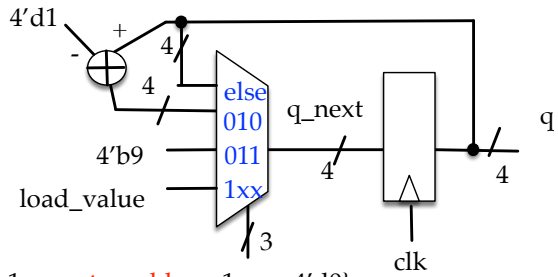
{load_value_enable==1, count_enable ==1, q==4'd9}

counterx

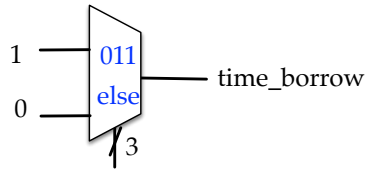


Timer

- Based on downcounter

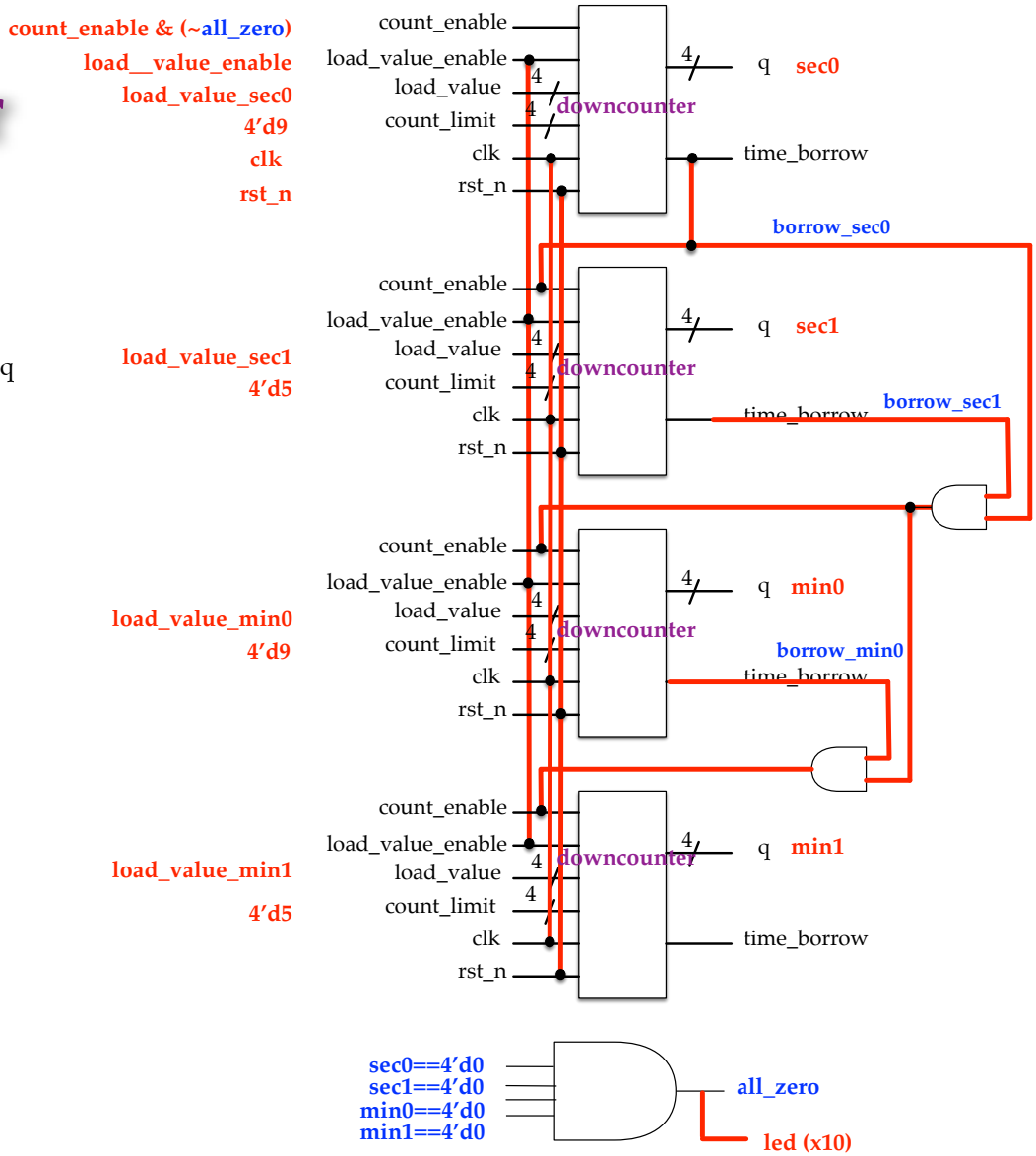


{load_value_enable==1, count_enable ==1, q==4'd0}

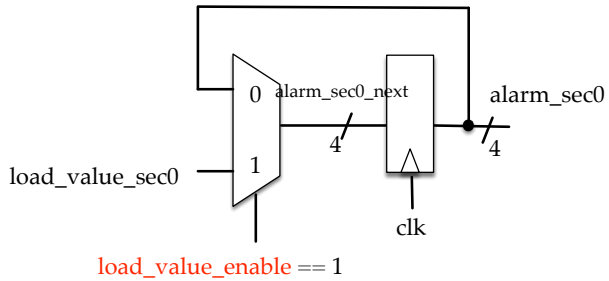


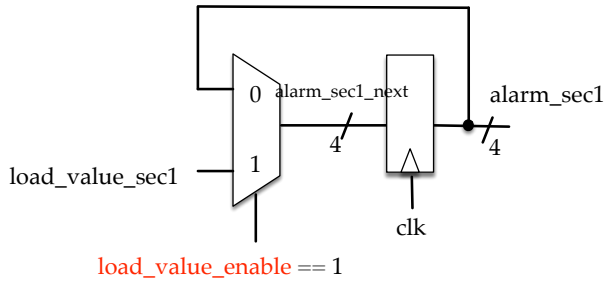
{load_value_enable==1, count_enable ==1, q==4'd1}

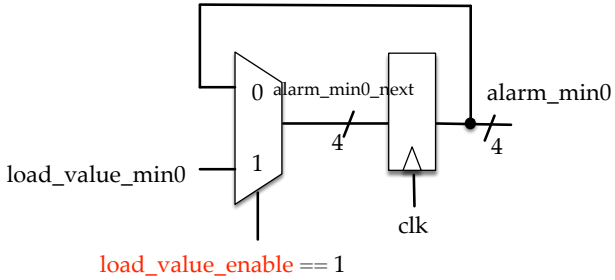
downcounter

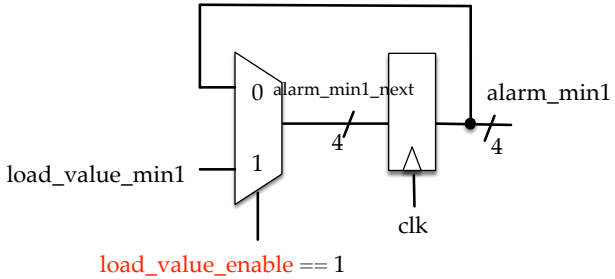


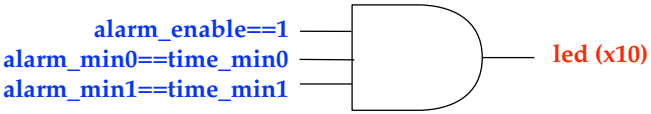
Alarm


0
1
alarm_sec0_next
4
alarm_sec0
4
load_value_sec0
clk
load_value_enable == 1


0
1
alarm_sec1_next
4
alarm_sec1
4
load_value_sec1
clk
load_value_enable == 1

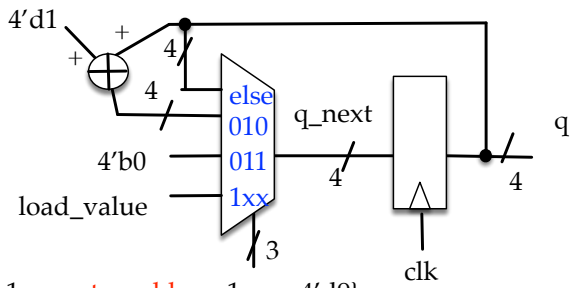

0
1
alarm_min0_next
4
alarm_min0
4
load_value_min0
clk
load_value_enable == 1


0
1
alarm_min1_next
4
alarm_min1
4
load_value_min1
clk
load_value_enable == 1

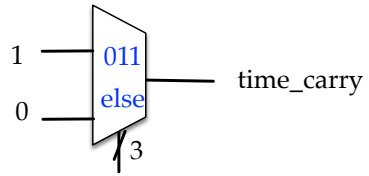

alarm_enable==1
alarm_min0==time_min0
alarm_min1==time_min1
led (x10)

Unitset (Setting Function)

- Based on counterx

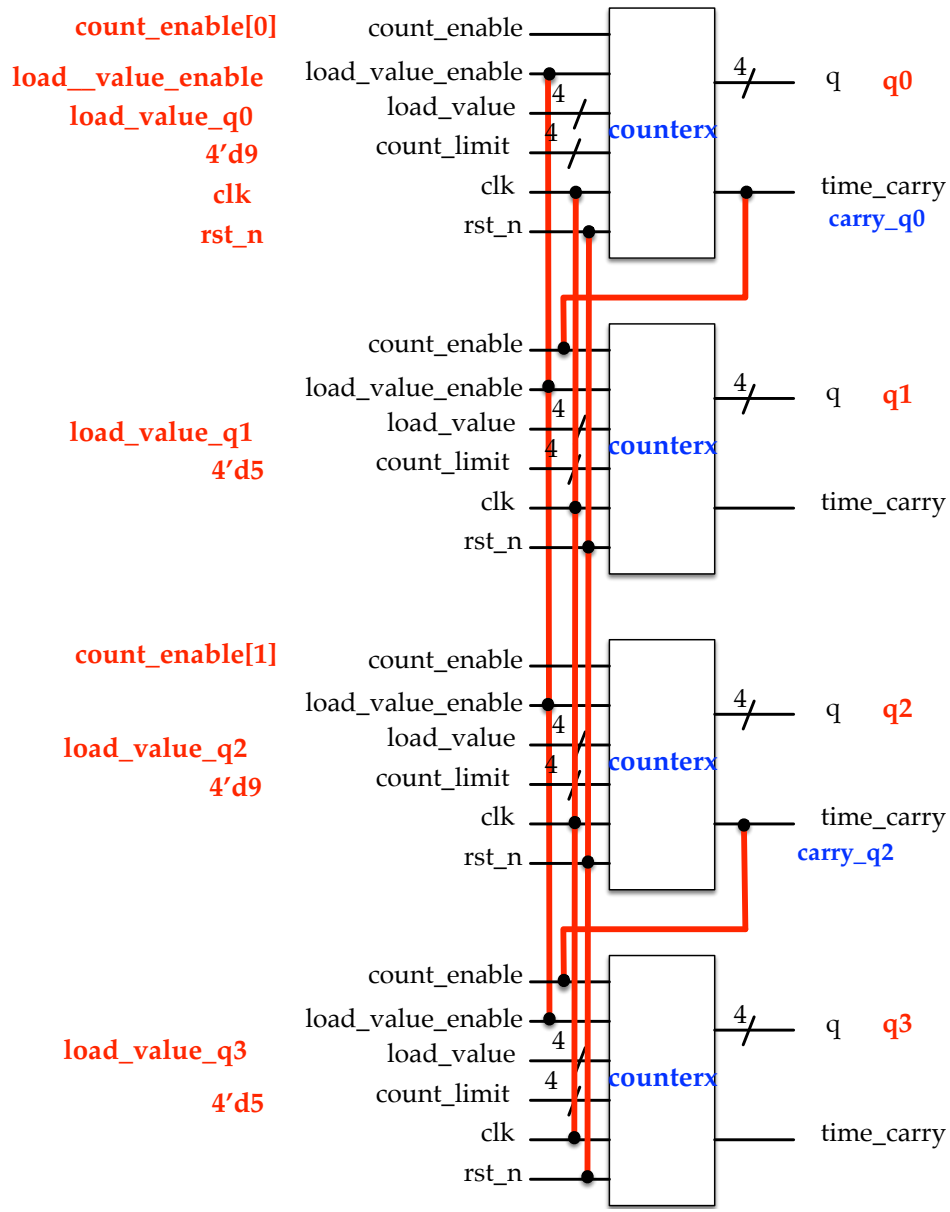


{load_value_enable==1, count_enable ==1, q==4'd9}



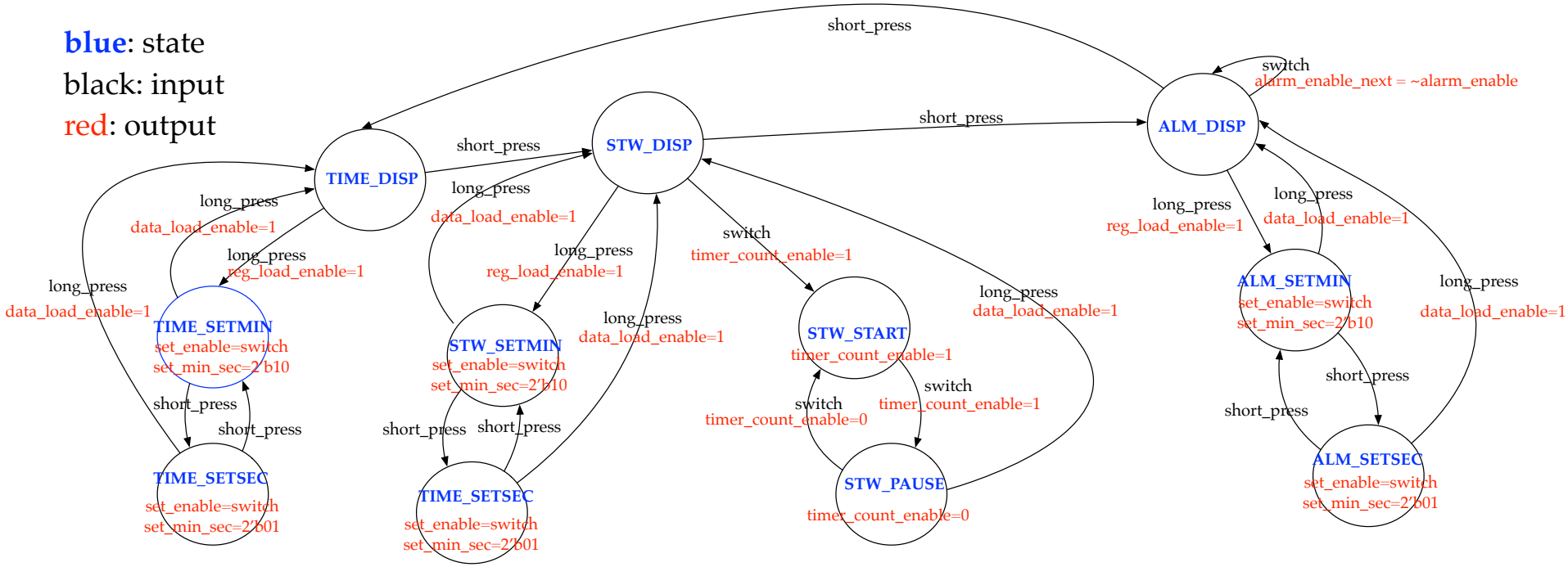
{load_value_enable==1, count_enable ==1, q==4'd9}

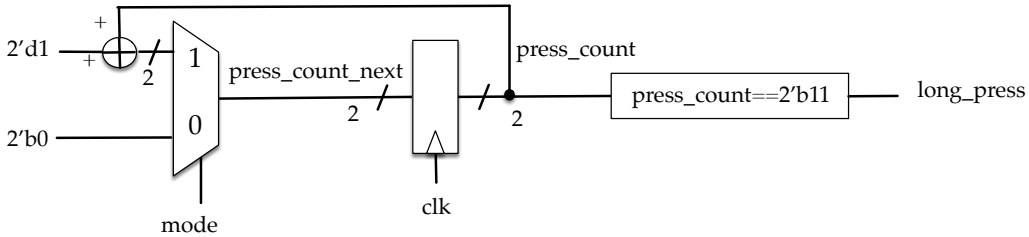
counterx

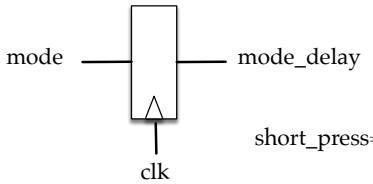


FSM

blue: state
black: input
red: output







short_press=(~mode) & mode_delay & (~(press_count[1]&press_count[0]))

scan_ctl

use 2kHz for scan frequency