Finite-State Machine Miniwatch

黃元豪
Yuan-Hao Huang

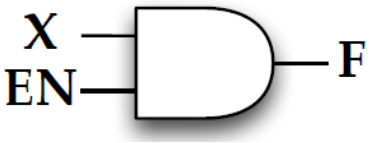
國立清華大學電機工程學系
Department of Electrical Engineering
National Tsing-Hua University

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

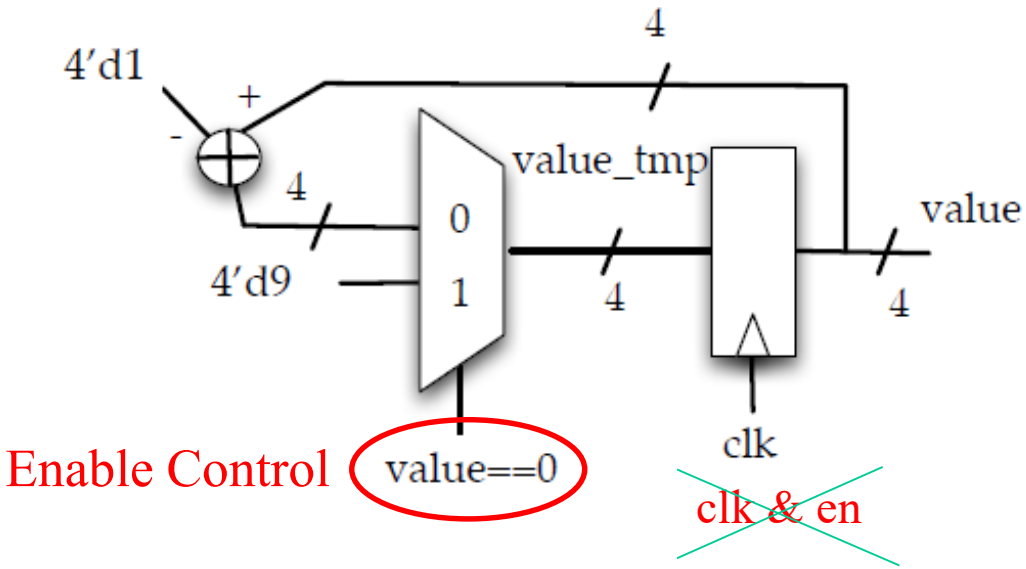
- Enable control (with AND gate) for combinational logics (block input)



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

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 Design Procedure

- Derive the state diagram
- Determine # of DFFs (for N states)
  - Use [$\log_2 N$] 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
```

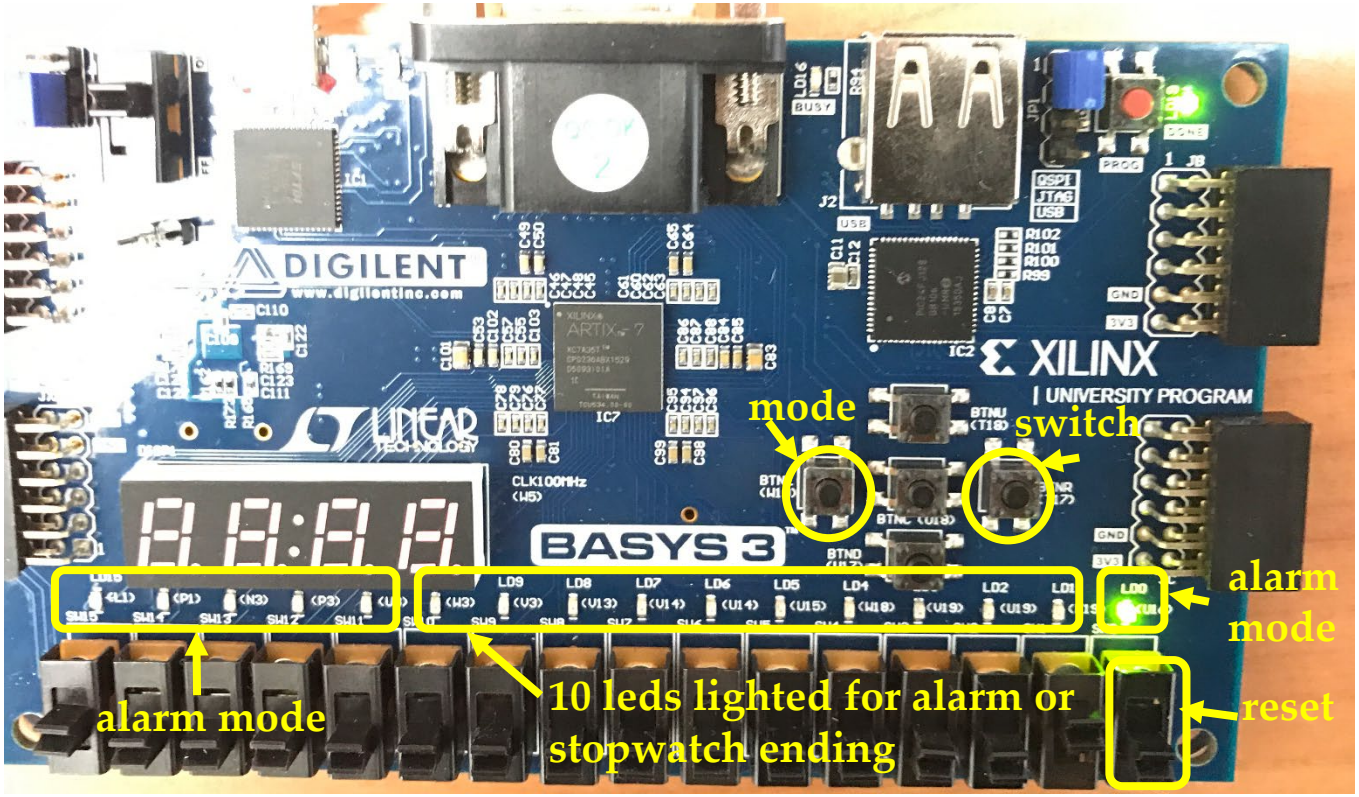
```
always @*
begin
  y=0;
  if (alu_control==2'b00)
    y=x+z;
  else if (alu_control==2'b01)
    y=x-z;
  else if (alu_control==2'b10)
    y=x*z;
end
```

```
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;
```

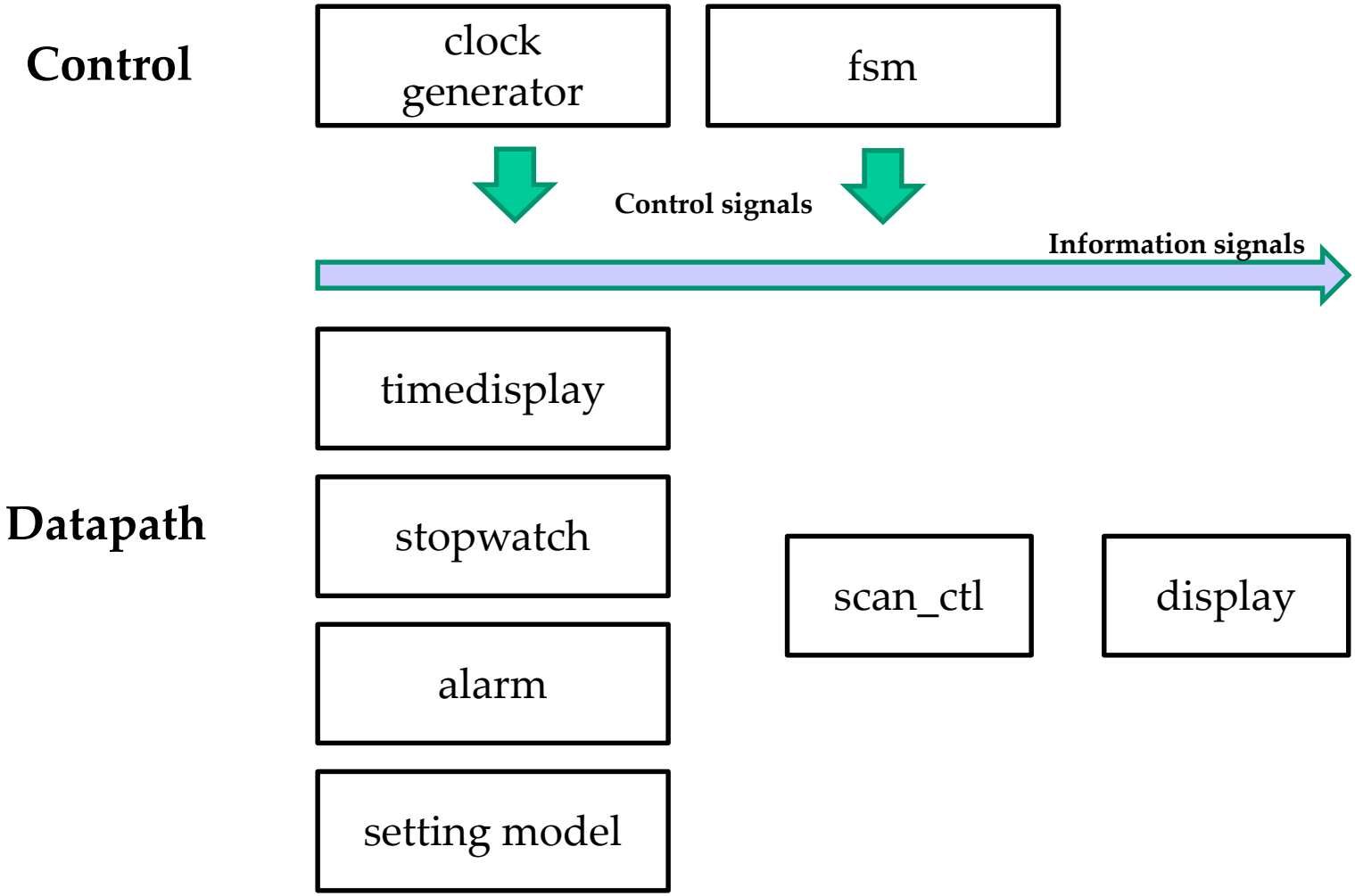
Miniwatch Example

- Specifications
  - Miniwatch
    - minute-second time display
    - stopwatch supporting up to 59:59
    - alarm for a certain time
  - FSM controller for all normal functions
  - Two push buttons for external control (one additional reset with DIP switch)

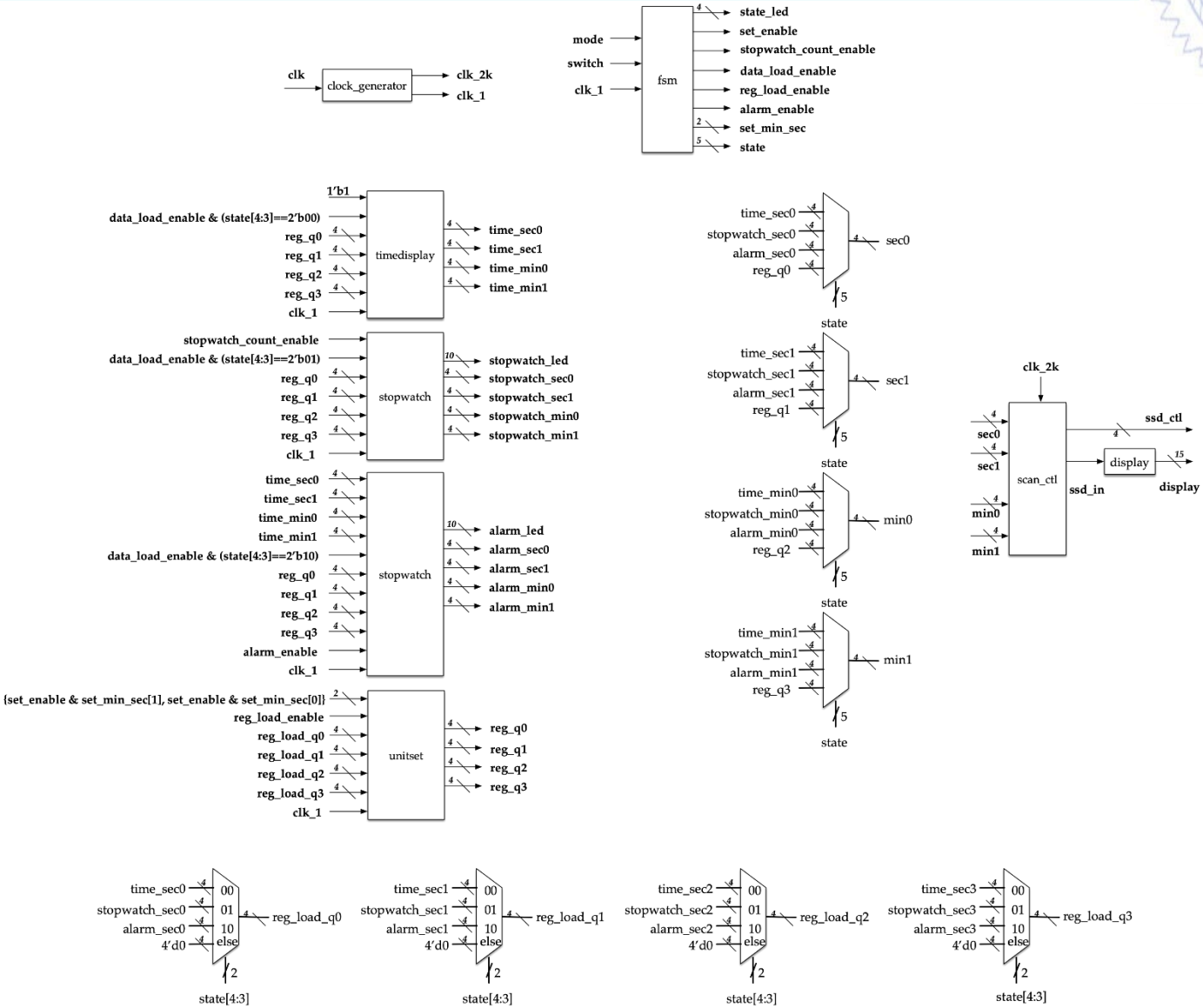
I/O Description for Demonstration



Hardware Architecture

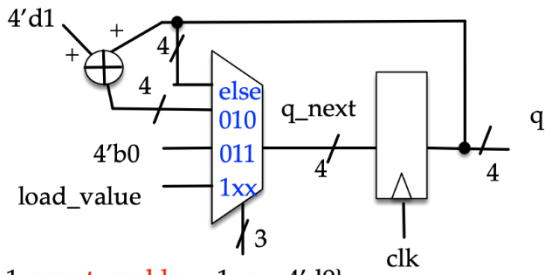


Block Diagram

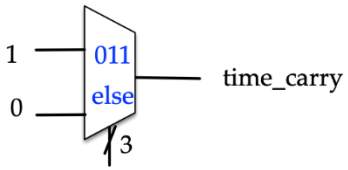


timedisplay module

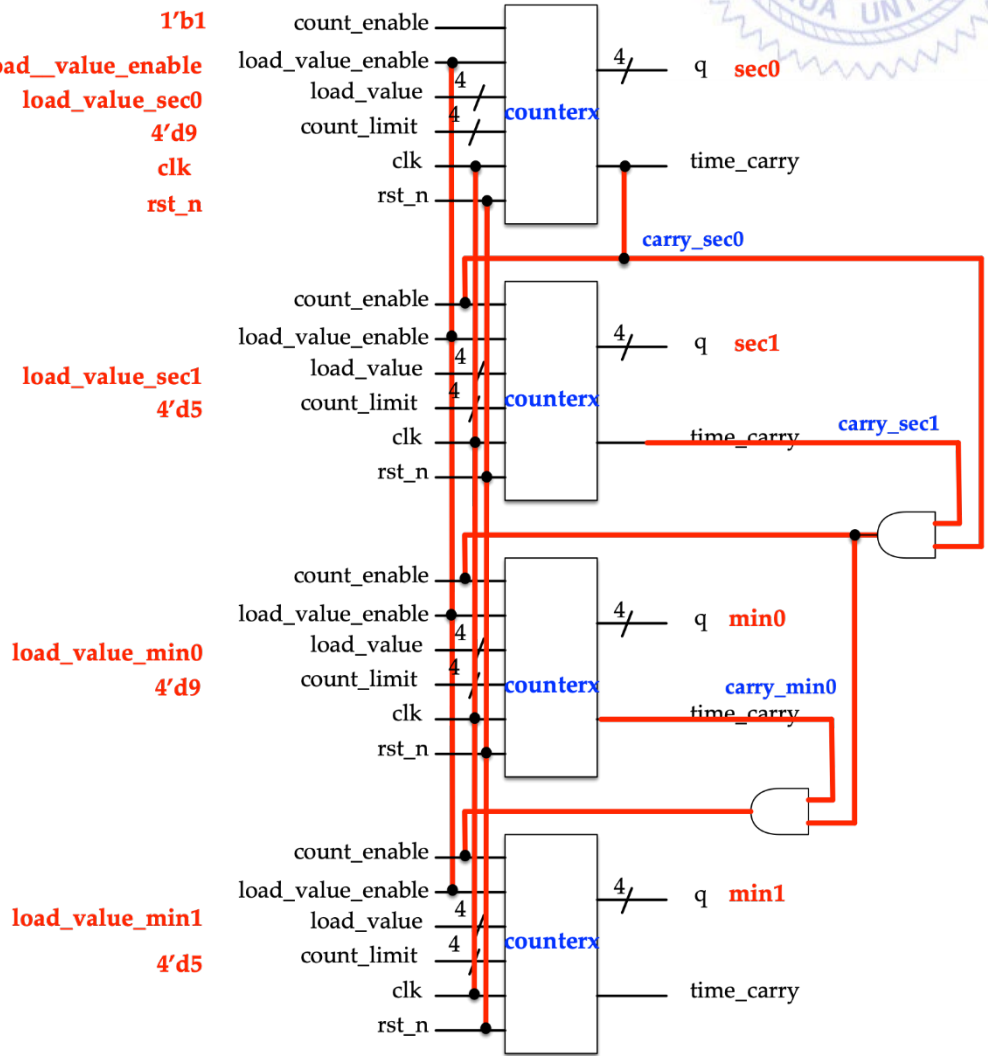
- counterx



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

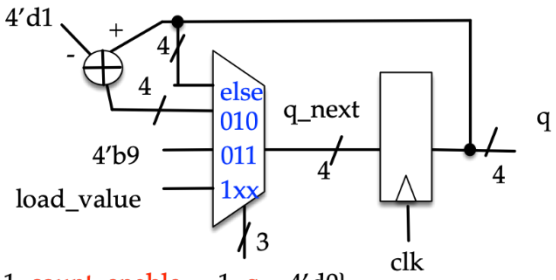


{load_value_enable==1, count_enable ==1, q==4'd9}

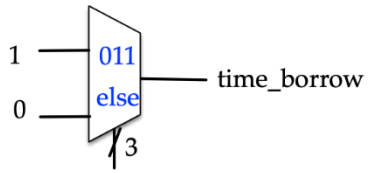


stopwatch module

- downcounter



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



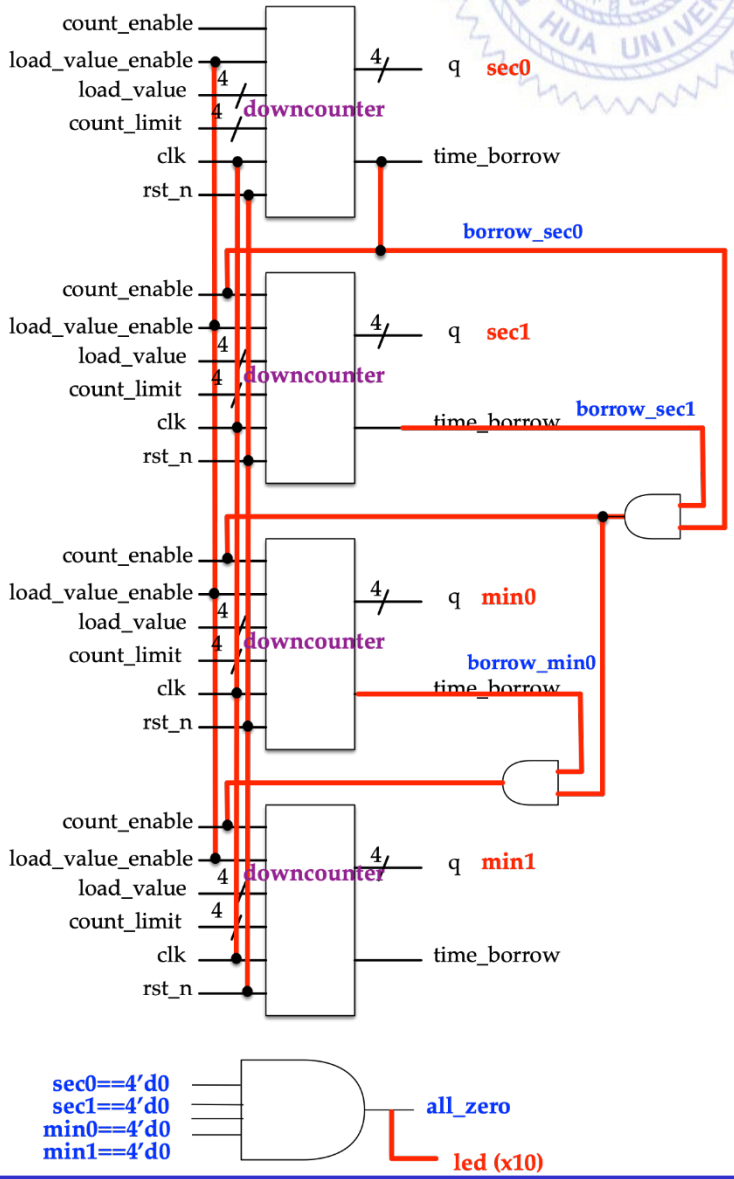
{load_value_enable==1, count_enable ==1, q==4'd1}

count_enable & (~all_zero)
load__value_enable
load_value_sec0
4'd9
clk
rst_n

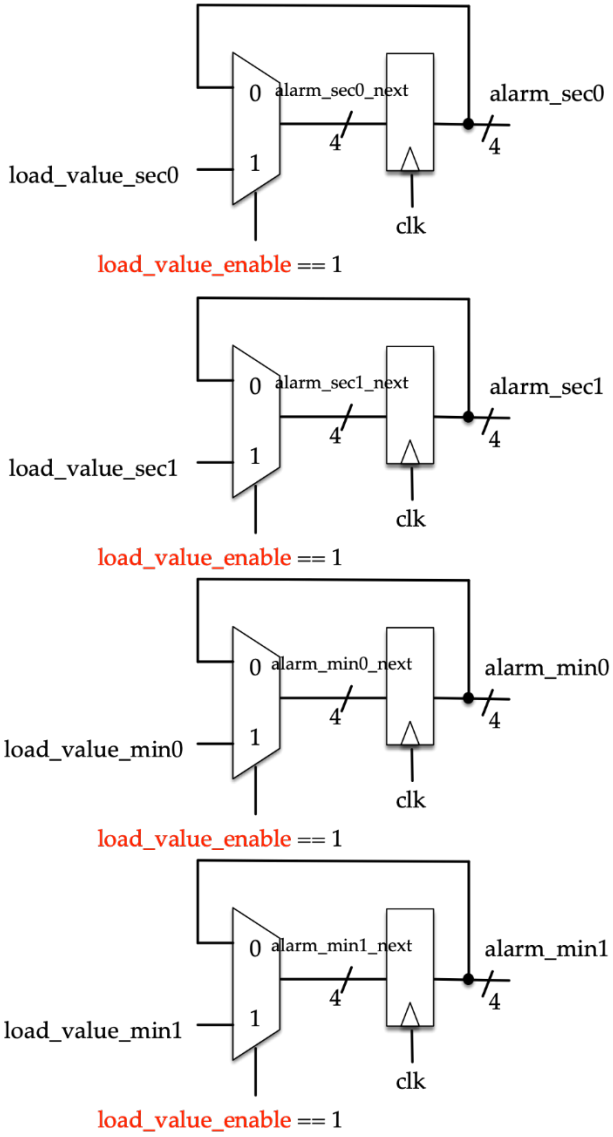
load_value_sec1
4'd5

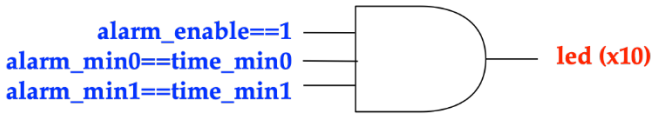
load_value_min0
4'd9

load_value_min1
4'd5

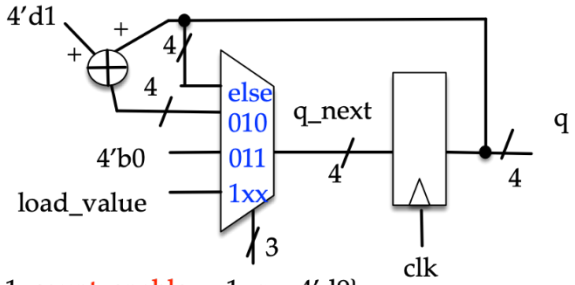


alarm module

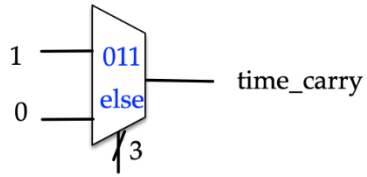




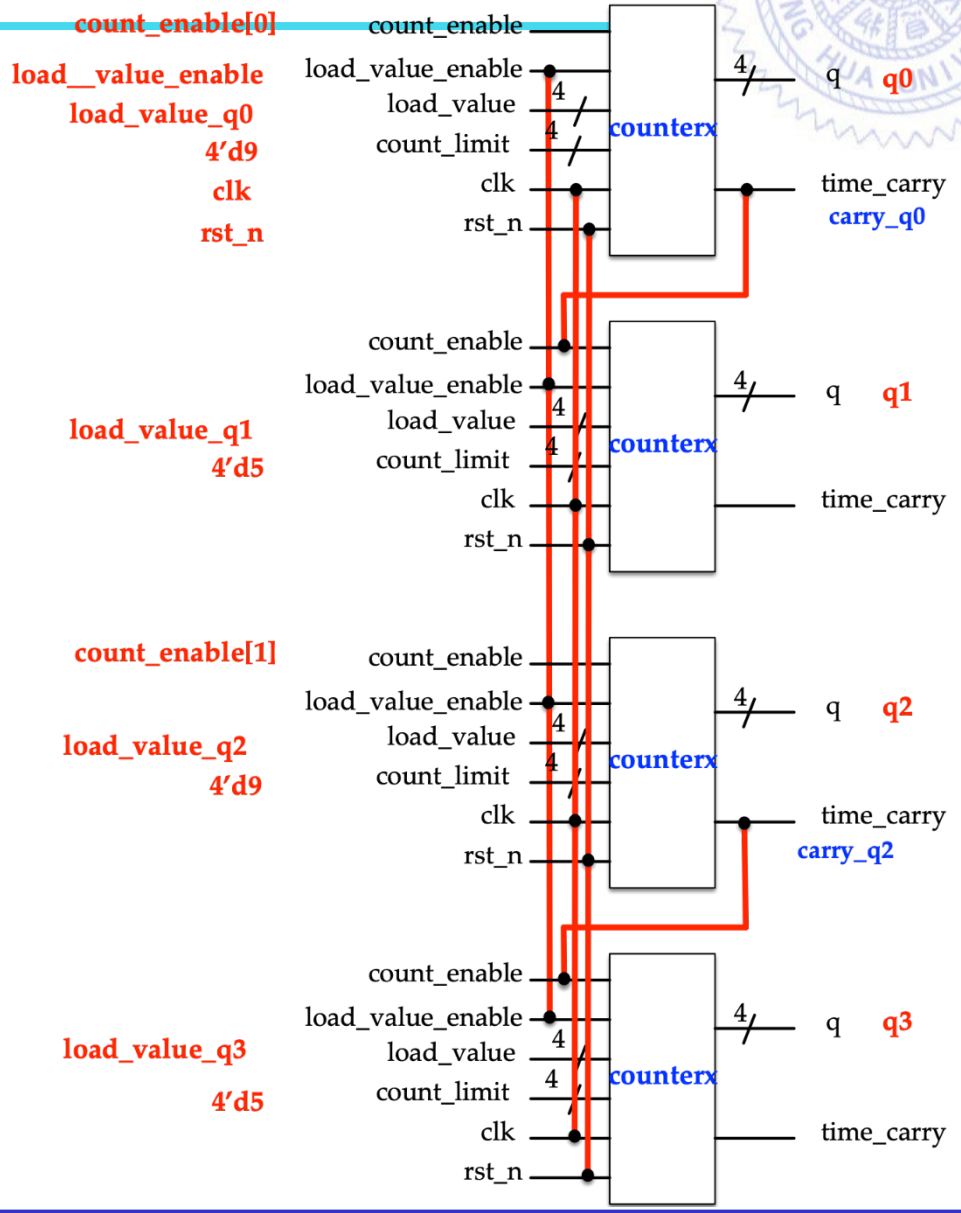
Unitset module (setting functions)



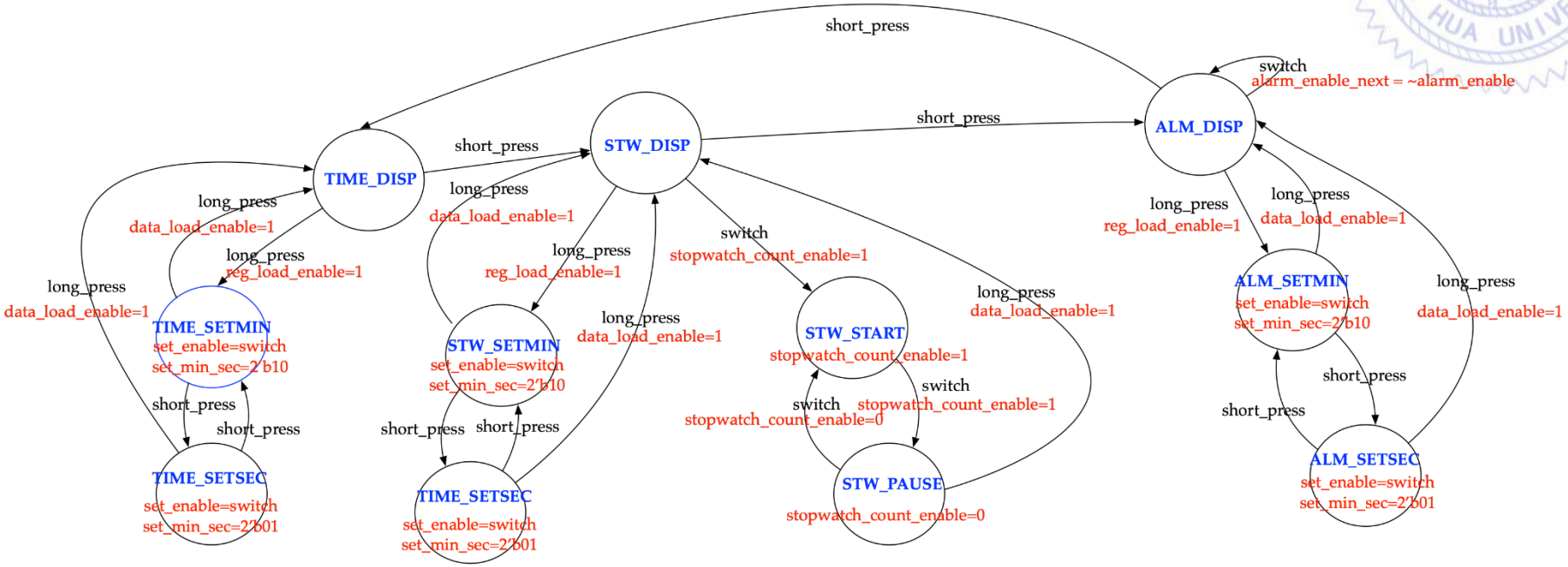
{load_value_enable==1, count_enable ==1, q==4'd9}

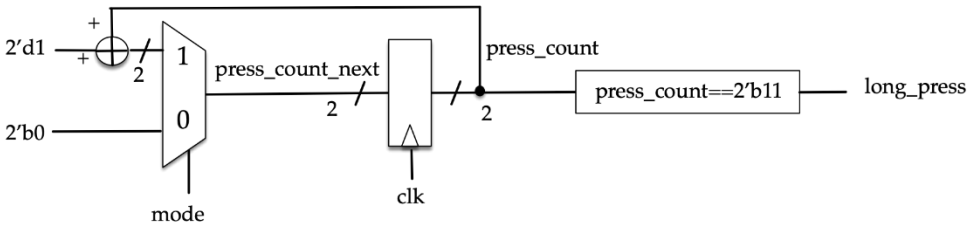


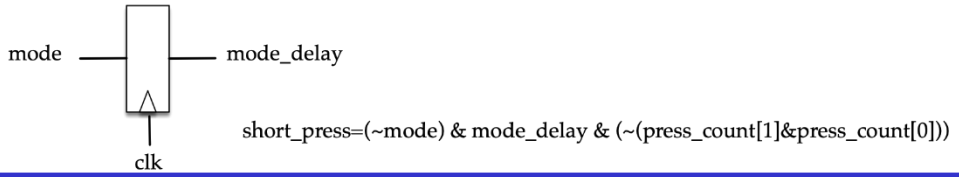
{load_value_enable==1, count_enable ==1, q==4'd9}



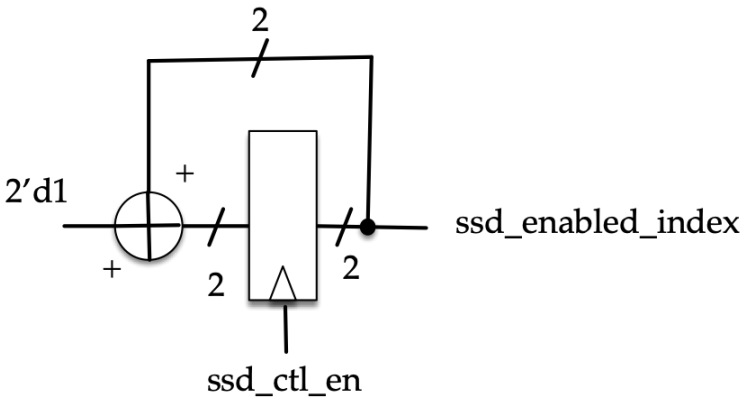
FSM module

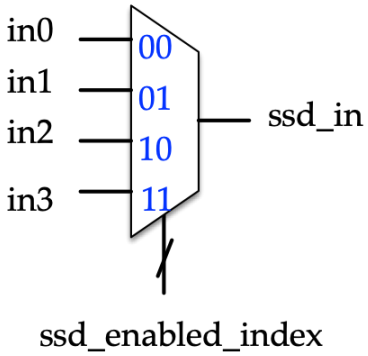

short_press
switch
alarm_enable_next = ~alarm_enable
ALM_DISP
short_press
STW_DISP
short_press
TIME_DISP
long_press
data_load_enable=1
long_press
reg_load_enable=1
long_press
data_load_enable=1
long_press
reg_load_enable=1
switch
stopwatch_count_enable=1
long_press
data_load_enable=1
long_press
reg_load_enable=1
long_press
data_load_enable=1
long_press
data_load_enable=1
ALM_SETMIN
set_enable=switch
set_min_sec=2'b10
long_press
data_load_enable=1
TIME_SETMIN
set_enable=switch
set_min_sec=2'b10
STW_SETMIN
set_enable=switch
set_min_sec=2'b10
STW_START
stopwatch_count_enable=1
short_press
short_press
short_press
short_press
switch
stopwatch_count_enable=0
switch
stopwatch_count_enable=1
short_press
TIME_SETSEC
set_enable=switch
set_min_sec=2'b01
TIME_SETSEC
set_enable=switch
set_min_sec=2'b01
STW_PAUSE
stopwatch_count_enable=0
ALM_SETSEC
set_enable=switch
set_min_sec=2'b01

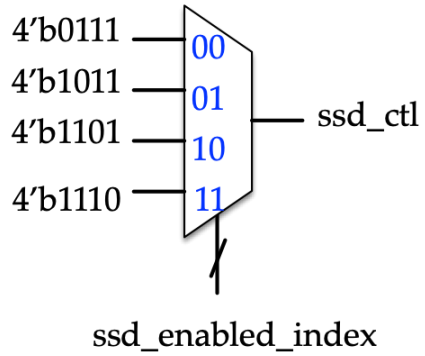

2'd1
+
+
2
1
0
2'b0
press_count_next
2
press_count
2
press_count==2'b11
long_press
mode
clk


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

scan_ctrl module







2kHz for scan frequency