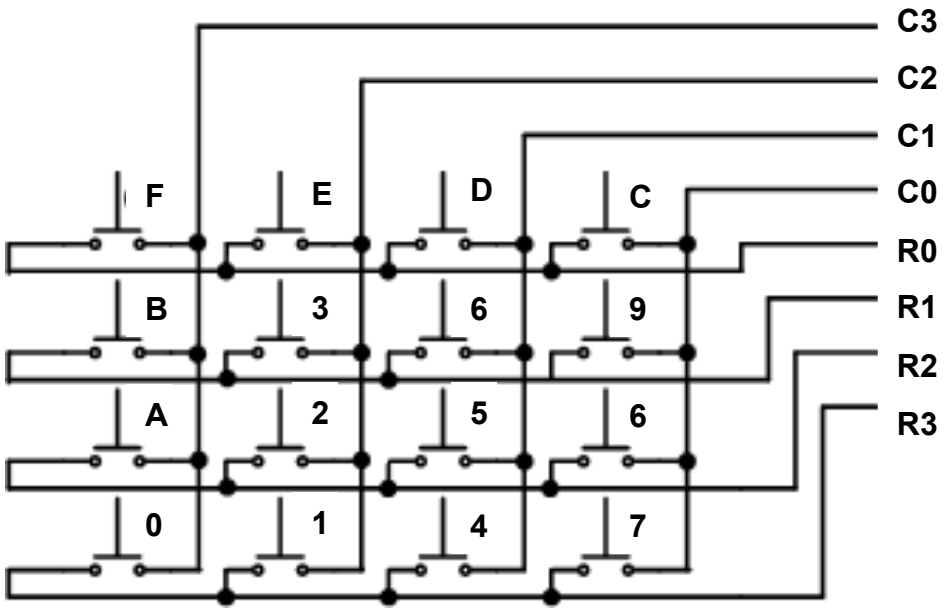
Using Keypad

Hsi-Pin Ma

http://lms.nthu.edu.tw/course/24953
Department of Electrical Engineering
National Tsing Hua University

4x4 Keypad

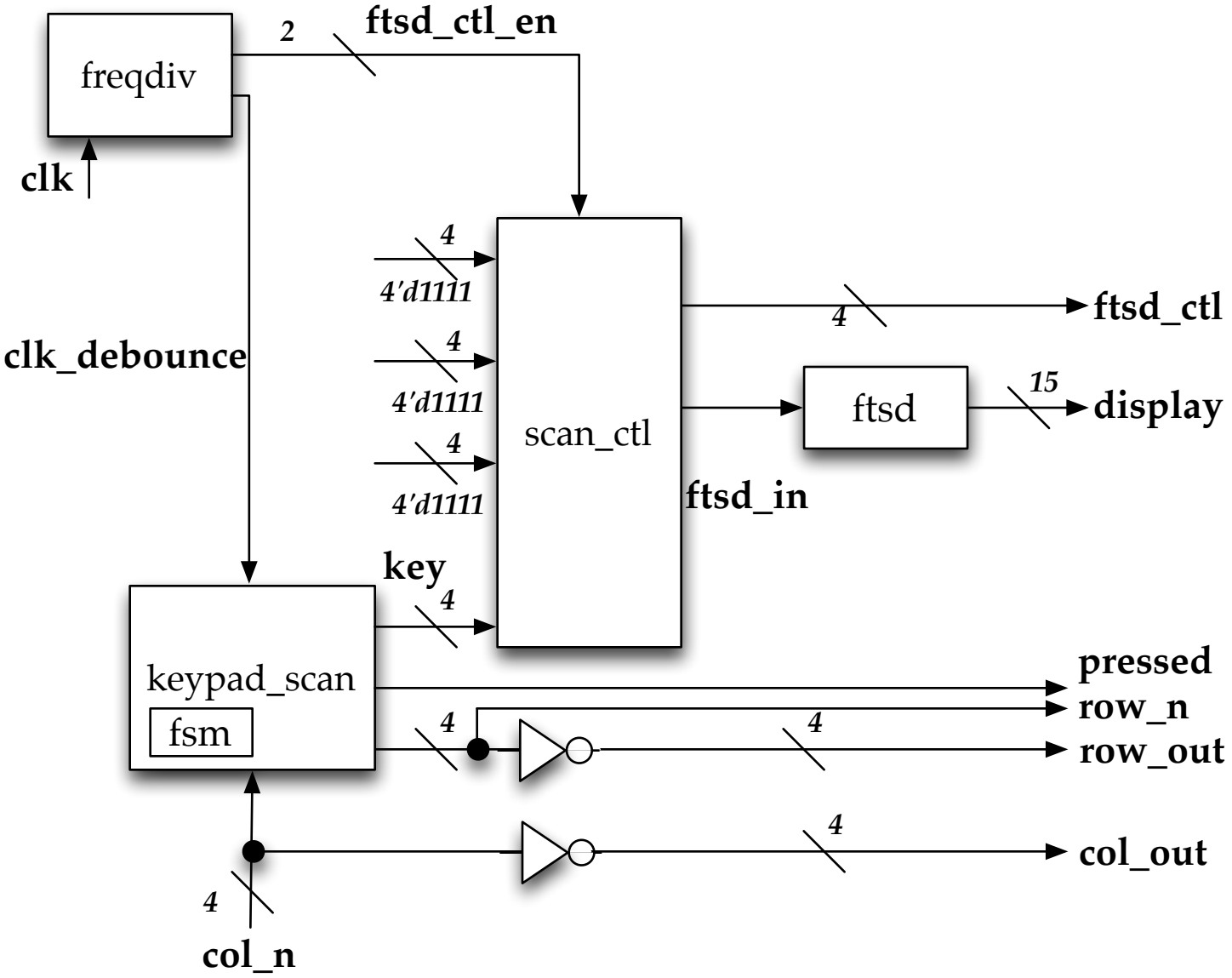


Row[3]	Row[2]	Row[1]	Row[0]
L3	L4	K1	K2
Col[3]	Col[2]	Col[1]	Col[0]
H1	H2	J1	J3

Note

- column, row are all low active
  - 0111 for first row / column
  - 1101 for third row / column
- Use 'row scan' + 'key press (column)' to locate the real key pressed (2D addressing)
- See the demo (keypad_scan) and example for details

Keypad Scan


freqdiv
2
ftsd_ctl_en
clk
clk_debounce
4
4'd1111
4
4'd1111
4
4'd1111
scan_ctl
4
ftsd_ctl
ftsd
15
display
ftsd_in
key
4
keypad_scan
fsm
pressed
row_n
row_out
4
4
4
col_out
4
col_n

keypad_scan.v

1. Row scan (row_n: 0111->1011->1101->1110, loop every 4 clocks)

2. For {row_n,col_n}, the mapping:

```
// 1st row
0111_0111 => F
0111_1011 => E
0111_1101 => D
0111_1110 => C

// 2nd row
1011_0111 => B
1011_1011 => 3
1011_1101 => 6
1011_1110 =>9

// 3rd row
1101_0111 => A
1101_1011 => 2
1101_1101 => 5
1101_1110 => 8

// 4th row
1110_0111 => 0
1110_1011 => 1
1110_1101 => 4
1110_1110 =>7
```

Just one-level case. AVOID using two level (nested) cases.

keypad_scan.v

```
`include "global.v"
module keypad_scan(
  clk, // scan clock
  rst_n, // active low reset
  col_n, // pressed column index
  row_n, // scanned row index
  key, // returned pressed key
  pressed // whether key pressed (1) or not (0)
);

// Declare I/Os
input clk; // scan clock
input rst_n; // active low reset
input [`KEYPAD_COL_WIDTH-1:0] col_n;
output [`KEYPAD_ROW_WIDTH-1:0] row_n;
output [3:0] key; // returned pressed key
output pressed; // whether key pressed (1) or not (0)

// Declare internal nodes
reg [1:0] sel, sel_next;
reg [`KEYPAD_ROW_WIDTH-1:0] row_n;
reg [3:0] key;
reg [3:0] key_detected;
reg [3:0] key_next;
reg keypad_state, keypad_state_next;
```

```
reg [`KEYPAD_PAUSE_PERIOD_BIT_WIDTH-1:0]
    pause_delay, pause_delay_next;
reg pressed_detected;
reg pressed_next;
reg pressed

// A repetitive counter for row-wise scan
always @(posedge clk or negedge rst_n)
  if (~rst_n)
    sel = 2'b00;
  else
    sel = sel_next;

always @*
  sel_next = sel + 1'b1;

// row-wise scan
always @*
  case (sel)
    2'd0: row_n = 4'b0111;
    2'd1: row_n = 4'b1011;
    2'd2: row_n = 4'b1101;
    2'd3: row_n = 4'b1110;
    default: row_n=4'b1111;
  endcase
```

keypad_scan.v

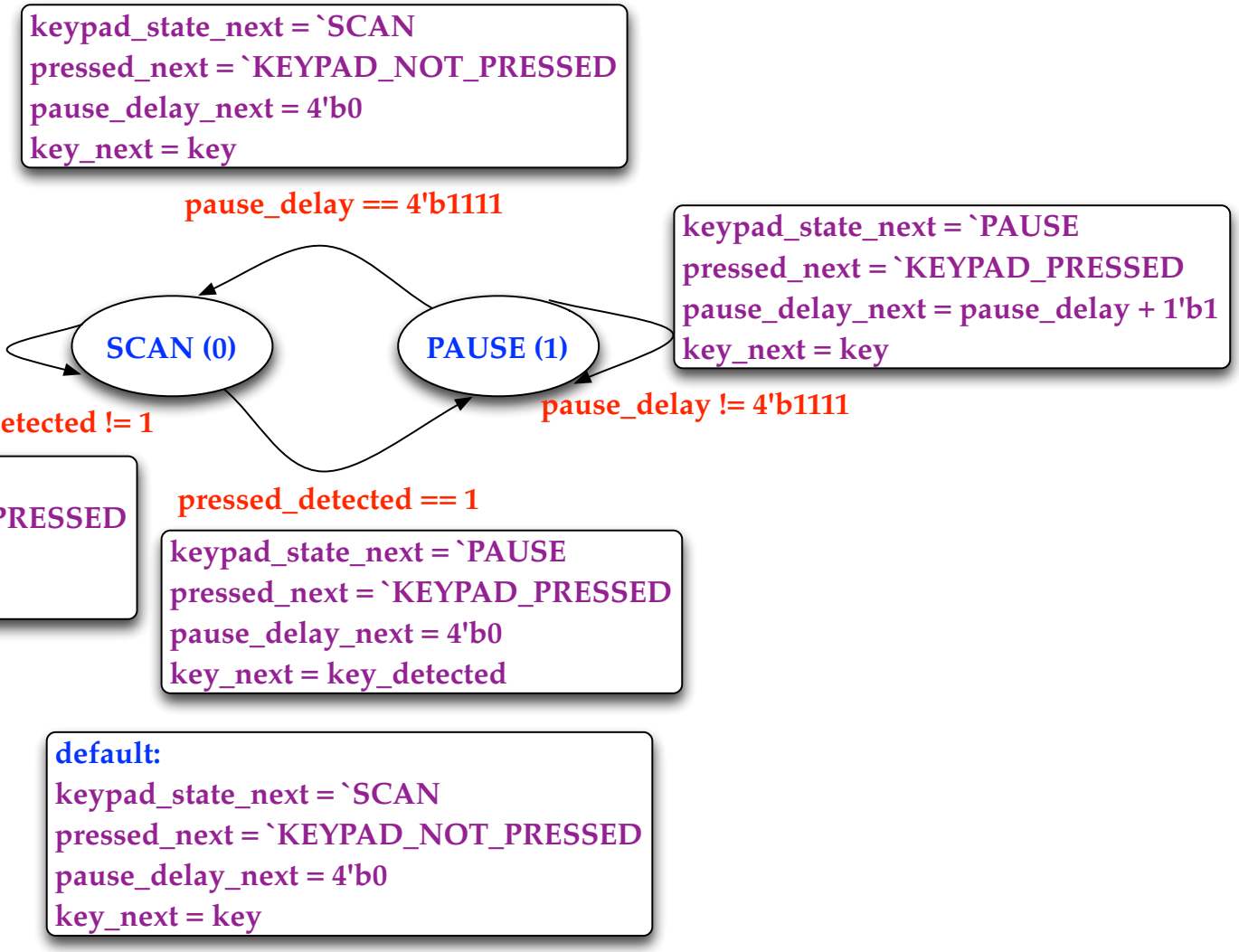
```
// column-wise readout
always @*
begin
  case ({row_n,col_n})
    `KEYPAD_DEC_WIDTH'b0111_0111: // press 'F'
    begin
      key_detected = `KEY_F;
      pressed_detected = `KEYPAD_PRESSED;
    end
    `KEYPAD_DEC_WIDTH'b0111_1011: // press 'E'
    begin
      key_detected = `KEY_E;
      pressed_detected = `KEYPAD_PRESSED;
    end
    `KEYPAD_DEC_WIDTH'b0111_1101: // press 'D'
    begin
      key_detected = `KEY_D;
      pressed_detected = `KEYPAD_PRESSED;
    end

.....
```

```
    `KEYPAD_DEC_WIDTH'b1110_1011: // press '1'
    begin
      key_detected = `KEY_1;
      pressed_detected = `KEYPAD_PRESSED;
    end
    `KEYPAD_DEC_WIDTH'b1110_1101: // press '4'
    begin
      key_detected = `KEY_4;
      pressed_detected = `KEYPAD_PRESSED;
    end
    `KEYPAD_DEC_WIDTH'b1110_1110: // press '7'
    begin
      key_detected = `KEY_7;
      pressed_detected = `KEYPAD_PRESSED;
    end
    default:
    begin
      pressed_detected = `KEYPAD_NOT_PRESSED;
      key_detected = `KEY_0;
    end
  endcase
```

Just one-level case. AVOID using two level (nested) cases.

FSM for Keypad Scan

Red: input
Purple: output



keypad_scan.v

```
// ************************
// FSM for keypad scan
// ************************
// FSM state transition
always @*
  case (keypad_state)
    `SCAN:
     begin
      if (pressed_detected == `KEYPAD_PRESSED)
      begin
        keypad_state_next = `PAUSE;
        pressed_next = `KEYPAD_PRESSED;
        pause_delay_next = `KEYPAD_PAUSE_PERIOD_BIT_WIDTH'b0;
        key_next = key_detected;
      end
      else
      begin
        keypad_state_next = `SCAN;
        pressed_next = `KEYPAD_NOT_PRESSED;
        pause_delay_next = `KEYPAD_PAUSE_PERIOD_BIT_WIDTH'b0;
        key_next = key;
      end
     end
```

keypad_scan.v

```
`PAUSE:
begin
 if (pause_delay==`KEYPAD_PAUSE_PERIOD_BIT_WIDTH'b1111)
 begin
  keypad_state_next = `SCAN;
  pressed_next = `KEYPAD_NOT_PRESSED;
  pause_delay_next = `KEYPAD_PAUSE_PERIOD_BIT_WIDTH'b0;
  key_next = key;
 end
 else
 begin
  keypad_state_next = `PAUSE;
  pressed_next = `KEYPAD_PRESSED;
  pause_delay_next = pause_delay + 1'b1;
  key_next = key;
 end
end
default:
begin
 keypad_state_next = `SCAN;
 pressed_next = `KEYPAD_NOT_PRESSED;
 pause_delay_next = `KEYPAD_PAUSE_PERIOD_BIT_WIDTH'b0;
 key_next = key;
end
endcase
```

keypad_scan.v

```
// FSM state register
always @(posedge clk or negedge rst_n)
  if (~rst_n)
    keypad_state <= 1'b0;
  else
    keypad_state <= keypad_state_next;

// Keypad Pause state counter
always @(posedge clk or negedge rst_n)
  if (~rst_n)
    pause_delay <= `KEYPAD_PAUSE_PERIOD_BIT_WIDTH'd0;
  else
    pause_delay <= pause_delay_next;

// pressed indicator
always @(posedge clk or negedge rst_n)
  if (~rst_n)
    pressed <= `KEYPAD_NOT_PRESSED;
  else
    pressed <= pressed_next;
```

```
// pressed key indicator
always @(posedge clk or negedge rst_n)
  if (~rst_n)
    key <= `KEY_0;
  else
    key <= key_next;

endmodule
```

Modification for lab6

- You should modify the keypad_scan.v to get the press value correctly.
  - Hint: One pulse pressed?
- Steps for exp.2 demo:
  - After reset
  - Press first number (0-9) and the number will be displayed in ftsd0 (addend)
  - Press A for addition
  - Press second number (0-9) and the number will be displayed in ftsd1(augend)
  - Press E, and the result will be shown on ftsd2, ftsd3