課題23-2

コード

matrix_key.v

```
module m_matrix_key(
    input             clk, rst,    // クロック，リセット
    input      [3:0]  row,          // 4bit入力 行
    output reg [3:0]  col,          // 4bit出力 列
    output reg [15:0] key,          // 16bitキー出力
    output            tc             // 出力カウント
    ) ;

    reg [2:0]  index ;
    reg [15:0] tmp ;

    always @(posedge rst or posedge clk) begin
        if(rst == 1'b1)begin
           tmp   <= 16'hFFFF ;
            key   <= 16'h0000 ;
            index <= 3'd0 ;
        end
        else begin
            // LSB=0 のとき：col を1列ずつ Low にしてスキャン
            if (index[0] == 1'b0) begin
                case (index[2:1])
                    2'd0: begin
                                col <= 4'b1110 ;
                                key <= ~tmp ;     // 1周期分の結果を key に反
映
                                tmp <= 16'hFFFF ; // 一時バッファ初期化
                            end
                    2'd1: col <= 4'b1101 ;
                    2'd2: col <= 4'b1011 ;
                    2'd3: col <= 4'b0111 ;
                endcase
            end
            // LSB=1 のとき：row を読み取り tmp に保存
            else begin
                tmp[{2'd0, index[2:1]}] <= row[0] ;
                tmp[{2'd1, index[2:1]}] <= row[1] ;
                tmp[{2'd2, index[2:1]}] <= row[2] ;
                tmp[{2'd3, index[2:1]}] <= row[3] ;
            end

            index <= index + 3'd1 ;
        end
    end
```

```
    // index が一周したタイミングを示すフラグ
    assign tc = (index == 3'd0) ? 1'b1 : 1'b0 ;

endmodule

// 押下されているキーを4bitで出力（16bit→4bitデコーダ）
module m_dec16to4 (
    input [15:0] key,         // 16bit入力
    output [3:0] out,         // 4bit出力
    output       pushed       // 打鍵検出
) ;

    function [4:0] f ;
        input [15:0] in ;
        case(in)
              16'h0001: f = { 1'b1, 4'h0 } ;
              16'h0002: f = { 1'b1, 4'h1 } ;
              16'h0004: f = { 1'b1, 4'h2 } ;
              16'h0008: f = { 1'b1, 4'h3 } ;
              16'h0010: f = { 1'b1, 4'h4 } ;
              16'h0020: f = { 1'b1, 4'h5 } ;
              16'h0040: f = { 1'b1, 4'h6 } ;
              16'h0080: f = { 1'b1, 4'h7 } ;
              16'h0100: f = { 1'b1, 4'h8 } ;
              16'h0200: f = { 1'b1, 4'h9 } ;
              16'h0400: f = { 1'b1, 4'hA } ;
              16'h0800: f = { 1'b1, 4'hB } ;
              16'h1000: f = { 1'b1, 4'hC } ;
              16'h2000: f = { 1'b1, 4'hD } ;
              16'h4000: f = { 1'b1, 4'hE } ;
              16'h8000: f = { 1'b1, 4'hF } ;
              default:  f = { 1'b0, 4'h0 } ; // 同時押し・未押下は無効
        endcase
    endfunction

    assign { pushed, out } = f(key) ;

endmodule

module m_convert_num (
    input [3:0] key,
    output [3:0] num
);
    function [3:0] f;
        input [3:0] in;
        case(in)
            4'h0: f = 4'h1;
            4'h1: f = 4'h2;
            4'h2: f = 4'h3;
            4'h3: f = 4'hA;
            4'h4: f = 4'h4;
            4'h5: f = 4'h5;
            4'h6: f = 4'h6;
            4'h7: f = 4'hB;
```

```
            4'h8: f = 4'h7;
            4'h9: f = 4'h8;
            4'hA: f = 4'h9;
            4'hB: f = 4'hC;
            4'hD: f = 4'h0;
            4'hF: f = 4'hD;
            default: f = 4'hF;
        endcase
    endfunction

    assign num = f(key);

endmodule
```

SevenSegment.v

```
// 分周器(100Hz)
module m_prescale(input clk,output c_out);
    reg [19:0] cnt;
    wire wcout;

    assign wcout=(cnt==20'd499999) ? 1'b1 : 1'b0;
    assign c_out=wcout;

    always @(posedge clk) begin
        if(wcout==1'b1)
            cnt=0;
        else
            cnt=cnt+1;
    end
endmodule

// マトリクスキー用7セグLED表示
module m_mat7segment(input [3:0] idat, input pushed, output [7:0] odat);

    function [7:0] LedDec;
      input [3:0] num;
      begin
         case (num)
           4'h0:        LedDec = 8'b11000000;
           4'h1:        LedDec = 8'b11111001;
           4'h2:        LedDec = 8'b10100100;
           4'h3:        LedDec = 8'b10110000;
           4'h4:        LedDec = 8'b10011001;
           4'h5:        LedDec = 8'b10010010;
           4'h6:        LedDec = 8'b10000010;
           4'h7:        LedDec = 8'b11111000;
           4'h8:        LedDec = 8'b10000000;
           4'h9:        LedDec = 8'b10011000;
           4'ha:        LedDec = 8'b10001000;
           4'hb:        LedDec = 8'b10000011;
```

```
            4'hc:       LedDec = 8'b10100111;
            4'hd:       LedDec = 8'b10100001;
            4'he:       LedDec = 8'b10000110;
            4'hf:       LedDec = 8'b10001110;
            default:    LedDec = 8'b11111111;
          endcase
      end
    endfunction

    // キーが押されているときのみ表示
    assign odat= (pushed) ? LedDec(idat) : 8'b11111111;

endmodule
```

TopModule.v

```
module TopModule(
    input                CLK1,
    input                CLK2,
    output       [7:0]    HEX0,
    output       [7:0]    HEX1,
    output       [7:0]    HEX2,
    output       [7:0]    HEX3,
    output       [7:0]    HEX4,
    output       [7:0]    HEX5,
    input        [1:0]    BTN,
    output       [9:0]    LED,
    input        [9:0]    SW,
    input        [3:0]    KEY_ROW,
    output       [3:0]    KEY_COL
    );

    wire clk;
    wire [3:0]  wq;         // 押下キー(4bit)
    wire [3:0]  num;        // 表示用変換後データ
    wire [15:0] key;        // 押下キー(16bit)
    wire [7:0]  dec_pat0;   // 7セグ表示パターン
    wire pushed;            // 打鍵検出

    // 100Hz クロック生成
    m_prescale(CLK1, clk);

    // マトリクスキーのスキャン
    m_matrix_key(clk, SW[0], KEY_ROW, KEY_COL, key, tc);

    // 16bitキーを4bitに変換
    m_dec16to4(key, wq, pushed);

    // 配列番号 → 表示番号変換
    m_convert_num(wq, num);
```

```
    // 7セグ表示
    m_mat7segment(num, pushed, dec_pat0);

    assign LED  = {6'd0, wq};
    assign HEX0 = dec_pat0;
    assign HEX1 = 8'hff;
    assign HEX2 = 8'hff;
    assign HEX3 = 8'hff;
    assign HEX4 = 8'hff;
    assign HEX5 = 8'hff;

endmodule
```

動作確認

- m_convert_num モジュールのシミュレーションを行い、キー番号と表示番号の対応が正しいことを確認した。
- 実際にボタンを押して問題ないことを確認した

解説

4×4 マトリクスキーをスキャンして押下されたキーを検出し、その結果を 7 セグメント LED に表示している。

m_matrix_key は、列を 1 本ずつ Low にして行入力を読み取ることでキーの状態を取得する。index を用いて「列出力」と「行読み取り」を交互に行い、1 周期分の結果を tmp に蓄積する。1 周すると key に反映され、16bit のワンホットデータとして出力される。

m_dec16to4 では、16bit のワンホット信号から押されているキーを 4bit の番号に変換する。同時押しや未押下の場合は pushed を 0 にし、無効として扱う。

m_convert_num は、マトリクスキーの物理配置と表示したい数値の対応を変換するためのモジュールで、キー番号を 7 セグ表示用の値に並び替えている。

m_prescale は入力クロックを分周し、キー入力を安定して読み取るための低速クロック（100Hz）を生成する。

m_mat7segment は、押下されたキーがあるときのみ対応する 7 セグメントの点灯パターンを出力し、キーが押されていない場合は消灯する。

TopModule では、これらの各モジュールを接続し、マトリクスキー入力から 7 セグメント LED 表示までの一連の処理をまとめている。