課題22-5

コード

TopModule.v

```
module TopModule(
    ////////////// CLOCK //////////
    input                       CLK1,
    input                       CLK2,
    ////////////// SEG7 //////////
    output      [7:0]       HEX0,
    output      [7:0]       HEX1,
    output      [7:0]       HEX2,
    output      [7:0]       HEX3,
    output      [7:0]       HEX4,
    output      [7:0]       HEX5,
    ////////////// Push Button //////////
    input       [1:0]       BTN,
    ////////////// LED //////////
    output      [9:0]       LED,
    ////////////// SW //////////
    input       [9:0]       SW

    );

    wire w_we;          // RAM 書き込みイネーブル信号
    wire w_btn0;        // チャタリング除去後の BTN[0]
    wire [3:0] rdata;   // RAM からの読み出しデータ

    // BTN[0] のチャタリングを除去する
    m_chattering u0(CLK1, BTN[0], w_btn0);

    // ボタンが押されている間だけ書き込みを有効にする
    assign w_we = ~w_btn0;

    // RAM モジュール
    // アドレスは SW[9:4]
    // 書き込みデータは SW[3:0]
    m_ram u1(SW[9:4], SW[3:0], w_we, rdata);

    // LED は使用しないため全消灯
    assign LED = 10'h0;

    // 7 セグメント LED 表示
    // HEX0 には RAM から読み出したデータを表示
    m_seven_segment s0(rdata, HEX0);

    // HEX1 には書き込みデータを表示
    m_seven_segment s1(SW[3:0], HEX1);
```

```
    // HEX2 と HEX3 にはアドレスを 2 桁で表示
    m_seven_segment_2 s2(SW[9:4], HEX2, HEX3);

    // 使用しない 7 セグは消灯
    assign HEX4 = 8'hff;
    assign HEX5 = 8'hff;

endmodule
```

RAM.v

```
module m_ram(input [5:0] adr, input [3:0] wdata, input we, output [3:0]
rdata);
    reg [3:0] mem[0:63];   // 64 ワード分の 4bit RAM

    // 指定されたアドレスの内容を常に出力
    assign rdata = mem[adr];

    // 書き込みイネーブルの立ち上がりでデータを書き込む
    always @(posedge we) begin
        mem[adr] = wdata;
    end
endmodule

// チャタリング除去回路
module m_chattering(input clk, input sw_in, output sw_out);
    reg [15:0] cnt;   // 分周用カウンタ
    reg swreg;        // スイッチ状態保持用レジスタ
    wire iclk;        // 低速クロック

    assign sw_out = swreg;

    // カウンタを回し続ける
    always @(posedge clk) begin
        cnt = cnt + 1;
    end

    // カウンタの上位ビットを低速クロックとして使用
    assign iclk = cnt[15];

    // 一定周期ごとにスイッチ入力を取り込む
    always @(posedge iclk) begin
        swreg = sw_in;
    end
endmodule

// 7 セグメントデコーダ
module m_seven_segment(input [3:0] idat, output [7:0] odat);
    parameter dot = 1'b1;
```

```
    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

    wire [7:0] tdat;
    assign tdat = LedDec(idat);
    assign odat = {dot, tdat[6:0]};
endmodule

// 7 セグメントデコーダ 2 桁表示
module m_seven_segment_2(input [5:0] idat, output [7:0] odat1, output [7:0]
odat2);
    parameter dot = 1'b1;

    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

    wire [7:0] tdat1;
    wire [7:0] tdat2;

    assign tdat1 = LedDec(idat[3:0]);
    assign tdat2 = LedDec(idat[5:4]);
    assign odat1 = {dot, tdat1[6:0]};
    assign odat2 = {dot, tdat2[6:0]};
endmodule
```

動作確認

- スイッチでアドレスとデータを指定し，押しボタンを押すと RAM にデータが書き込まれることを確認した。
- 値が更新されていることが確認できた

解説

スライドスイッチの上位 6 ビットで RAM のアドレスを指定し，下位 4 ビットで書き込みデータを決める。押しボタンを押すと，その瞬間に指定したアドレスへデータが書き込まれる。

押しボタンにはチャタリングが発生するため，そのまま RAM の制御に使うと意図しない複数回の書き込みが起きてしまう。そこでクロックを分周し，一定周期ごとにスイッチ状態を取り込むことで入力を安定させている。この安定化された信号を反転させ，書き込みイネーブルとして使用している。

RAM は 64 ワード，1 ワード 4 ビットの構成で，アドレスが指定されると常にその内容が読み出される。読み出したデータは HEX0 に表示され，現在書き込もうとしているデータは HEX1 に表示される。また，指定しているアドレスは 2 桁の 7 セグメント LED に分けて表示されるため，どのアドレスを操作しているのかを視覚的に確認できる。

このように，入力，記憶，出力の流れをすべて目で確認できる構成になっており，RAM の動作原理を理解するための実験回路として適している。