# 課題22-3

## 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

    );

    //=============================
    // 内部信号定義
    //=============================
    reg  [3:0] cnt;     // 表示位置を制御する4bitカウンタ
    wire       clk_1s;  // 1秒周期クロック

    //=============================
    // 50MHz → 1Hz 分周回路
    //=============================
    // CLK1(50MHz)を入力し、1秒ごとに1パルス出力
    m_prescale50M tim(
        .CLK1(CLK1),
        .c_out(clk_1s)
    );

    //=============================
    // 1秒ごとにカウンタ更新
    //=============================
    always @(posedge clk_1s) begin
        cnt = cnt + 1;   // 4bitなので 0～15 で循環
    end

    //=============================
    // LED表示
```

```
    //==============================
    assign LED[0] = SW[0];  // スイッチ0の状態をLED0に表示

    //==============================
    // ROMを使った7セグ表示
    // cnt にオフセットを加えて
    // 文字が横にスクロールするように表示
    //==============================
    m_rom u1(cnt+5, SW[0], HEX0);
    m_rom u2(cnt+4, SW[0], HEX1);
    m_rom u3(cnt+3, SW[0], HEX2);
    m_rom u4(cnt+2, SW[0], HEX3);
    m_rom u5(cnt+1, SW[0], HEX4);
    m_rom u6(cnt+0, SW[0], HEX5);

endmodule
```

Rom.v

```
//==============================
// 7セグ用ROMモジュール
//==============================
// adr : 表示文字アドレス
// sw  : 表示切替スイッチ
// dat : 7セグ出力データ
//==============================
module m_rom(
    input  [3:0] adr,   // 文字位置アドレス
    input        sw,    // 表示切替用スイッチ
    output [7:0] dat    // 7セグメントLED出力
);

    reg [7:0] data;     // 内部保持用レジスタ
    assign dat = data;  // 出力に接続

    //==============================
    // アドレスが変化したら即時更新
    //==============================
    always @(adr) begin
        //--------------------------
        // sw = 1 のとき
        // "GOOdbyE " を表示
        //--------------------------
        if(sw) begin
            case (adr)
                4'h0: data = 8'b11000010;  // G
                4'h1: data = 8'b11000000;  // O
                4'h2: data = 8'b11000000;  // O
                4'h3: data = 8'b10100001;  // d
                4'h4: data = 8'b10000011;  // b
```

```
                4'h5: data = 8'b10010001;  // y
                4'h6: data = 8'b10000110;  // E
                4'h7: data = 8'b11111111;  // スペース
                4'h8: data = 8'b11000010;  // G
                4'h9: data = 8'b11000000;  // O
                4'ha: data = 8'b11000000;  // O
                4'hb: data = 8'b10100001;  // d
                4'hc: data = 8'b10000011;  // b
                4'hd: data = 8'b10010001;  // y
                4'he: data = 8'b10000110;  // E
                4'hf: data = 8'b11111111;  // スペース
                default: data = 8'hff;
            endcase
        end
        //-------------------------
        // sw = 0 のとき
        // "HELLO   " を表示
        //-------------------------
        else begin
            case (adr)
                4'h0: data = 8'b10001001;  // H
                4'h1: data = 8'b10000110;  // E
                4'h2: data = 8'b11000111;  // L
                4'h3: data = 8'b11000111;  // L
                4'h4: data = 8'b11000000;  // O
                4'h5: data = 8'b11111111;  // スペース
                4'h6: data = 8'b11111111;  // スペース
                4'h7: data = 8'b11111111;  // スペース
                4'h8: data = 8'b10001001;  // H
                4'h9: data = 8'b10000110;  // E
                4'ha: data = 8'b11000111;  // L
                4'hb: data = 8'b11000111;  // L
                4'hc: data = 8'b11000000;  // O
                4'hd: data = 8'b11111111;  // スペース
                4'he: data = 8'b11111111;  // スペース
                4'hf: data = 8'b11111111;  // スペース
                default: data = 8'hff;
            endcase
        end
    end
endmodule

//==============================
// 50MHz → 1Hz 分周回路
//==============================
module m_prescale50M(
    input  clk,    // 50MHz クロック
    output c_out   // 1秒ごとのパルス出力
);

    reg  [25:0] cnt;   // 50,000,000 まで数えるカウンタ
    wire        wcout; // カウンタ満了フラグ

    // カウンタが 49,999,999 に達したら 1 を出力
```

```
    assign wcout = (cnt == 26'd49999999) ? 1'b1 : 1'b0;
    assign c_out = wcout;

    // クロック立ち上がりでカウント
    always @(posedge clk) begin
        if (wcout)
            cnt <= 26'd0;       // 1秒経過でリセット
        else
            cnt <= cnt + 26'd1;
    end
endmodule
```

## 動作確認

- スイッチのON/OFFで "HELLO " と "GOOdbyE " の表示を切り替え可能
- 7セグLEDに文字が左にスクロールして表示される

## 解説

m_prescale50M を用いて 50MHz のクロックを分周し 1 秒に 1 回だけ High になるパルスを生成している。このパルスを表示更新のタイミングとして使用することで 表示速度を人間が読み取りやすい速さにしている。

cnt は 1 秒ごとに 1 ずつ増加する 4 ビットのカウンタで 0 から 15 まで循環する。この値は 現在どの文字を表示するかを決めるためのスクロール位置として使われている。

文字の表示には m_rom を使用しており これは文字パターンを格納した ROM である。入力として 表示したい文字番号を示す adr と 表示内容を切り替えるためのスイッチ信号 sw を受け取り 出力として 7 セグメント LED の点灯パターンを生成する。

各 7 セグメント LED には cnt に異なるオフセットを加えた値を adr として与えている。1 秒ごとに cnt が増加することで 各表示器が参照する文字アドレスが順番にずれていき その結果 文字列が左方向へ流れているように見える。