導入課題

作成者：Pyii Phyo Maung
Student ID: 1029322149

April 20, 2023

課題の概要

7SEG LED(下図)は16進数のA～Fを含む数値の表示のためによく利用されるデバイスである． このデバイスは，8つのLEDから構成される．

7SEG LED 通常，7SEG LEDはデコーダとセットで利用され，1桁の16進数(4ビット)の入力を受けて0～Fを表示する． しかし，実験3ハードウェアで使うPowerMedusaボード上の7SEG LEDは，16進数の表現以外の用途にも利用できるように，あえてデコーダは準備されていない． そのため，FPGAの出力は7SEG LED中の8個のLEDそれぞれにつながっている．

本課題では，この1桁の16進数(4ビット)の入力を受け，MU500-RK上（MU500-7SEG上ではない）の8個のLEDを適切に点灯させて0～Fを表現する組合せ回路を設計する． また，この回路を部品として用いてカウント値を表示する順序回路を設計する．

1　課題1

PowerMedusa上の入力装置から4ビットのデータを入力し，MU500-RK上の7SEG LEDに1桁の16進数を表示する回路を設計し，ボード上で動作を確認せよ．

1.1　コードとその説明

```
module intro7seg(
input [3:0] number,
output [7:0] signal,
output [3:0] selector
);

function [7:0] display;
input [3:0] a;
begin
    case(a)
      4'h0: display = 8'b11000000;
      4'h1: display = 8'b11111001;
      4'h2: display = 8'b10100100;
      4'h3: display = 8'b10110000;
      4'h4: display = 8'b10011001;
```

```
      4'h5: display = 8'b10010010;
      4'h6: display = 8'b10000010;
      4'h7: display = 8'b11111000;
      4'h8: display = 8'b10000000;
      4'h9: display = 8'b10010000;
      4'hA: display = 8'b10001000;
      4'hB: display = 8'b10000011;
      4'hC: display = 8'b11000110;
      4'hD: display = 8'b10100001;
      4'hE: display = 8'b10000110;
      4'hF: display = 8'b10001110;
      default: display = 8'b11111111;
    endcase
  end
 endfunction
assign signal = display (number);
assign selector = 4'b0000;
endmodule
```

本コードは、0からFまでの16進数を7セグメントディスプレイに表示するためのモジュールです。入力は4ビットの2進数で表される。

まず、intro7segモジュールが定義されています。このモジュールは、入力ポート番号、出力ポート信号、セレクタを備えています。

次に、表示機能を定義します。この関数は、4ビット入力aを受け取り、対応する7セグメント表示パターンを8ビット2進数で返します。入力が16進数の0Fの場合、それぞれに対応する7セグメント表示パターンが割り当てられる。デフォルトでは、全セグメントを消したパターンが返される。

最後に、assign文を使って、信号とセレクタを適切な値に設定する。これにより、モジュールは正しく動作し、入力番号に応じて7セグメント・ディスプレイに正しい値を表示するようになります。

signalはdisplay関数を使って番号に対応する7セグメント表示パターンを取得し、それをsignalに代入する。これにより、7セグメント表示器に必要なセグメントが点灯し、入力された数値が表示されます。

selectorには4ビットバイナリ値4'b0000が割り当てられています。このデモコードではセレクタを使用していませんが、複数のディスプレイがある場合に、どのディスプレイを表示するかを選択するために使用できます。

以上が、このVerilogのコードが行っていることの説明です。このコードを使うことで、7セグメントディスプレイを制御して、0からFまでの16進数を表示することができます。

2 課題2

10進数4桁の数字を表示して一定の間隔で1ずつカウントアップする回路を設計し，ボード上で動作を確認せよ．設計した回路のサイズ(使用Logic Element数)，動作可能速度(最大クロック周波数)などの性能を調べよ．

(ヒント) 課題1の回路をモジュールとして呼び出して用いるとよい 順序回路になるのでクロックとalways文の出番である MU500-RK上の7SEG LEDでは一度に1つの数字しか表示できないため，ダイナミック点灯が必要である MU500-RKのクロックには20MHzのものと，ロータリースイッチによって周波数を変えられるものがある 回路性能の見方はFAQのページを参照すること．レポートでは，課題1と同様に，仕様の決定，論理設計，CAD設計，動作結果の検証，および，性能評価とその考察について説明する．

2.1 コードとその説明

2.1.1 intro7seg2 module

```
    module intro7seg2 (
    input clock,
    output reg [7:0] signal_out,
    output reg [3:0] selector
);
    wire clk_div;
 wire [31:0] signal;

    // Clock Divider
    clock_divider clk_div_inst (
        .clock(clock),
        .clk_div(clk_div)
    );

    wire [13:0] count;
    wire [3:0] n1, n2, n3, n4;
```

```
    // Counter for selector
    reg [1:0] selector_counter;

    always @(posedge clock) begin
        selector_counter <= selector_counter + 1;
        case (selector_counter)
            2'b00: selector <= 4'b1110;
            2'b01: selector <= 4'b1101;
            2'b10: selector <= 4'b1011;
            2'b11: selector <= 4'b0111;
        endcase
    end

 always @ (posedge clock) begin
case (selector_counter)
      2'b00: signal_out <= signal[7:0];
            2'b01: signal_out <= signal[15:8];
            2'b10: signal_out <= signal[23:16];
            2'b11: signal_out <= signal[31:24];
        endcase
    end


    // Counter
    counter100000000 counter_inst (
        .clock(clk_div),
        .count(count)
    );

    // Decimal converter
    convertdecimal decimal_inst (
        .binary(count),
        .num1(n1),
        .num2(n2),
        .num3(n3),
        .num4(n4)
```

```
    );

    digitdisplay dd1 (
        .number(n1),
        .signal(signal[7:0])
    );

    digitdisplay dd2 (
        .number(n2),
        .signal(signal[15:8])
    );

    digitdisplay dd3 (
        .number(n3),
        .signal(signal[23:16])
    );

    digitdisplay dd4 (
        .number(n4),
        .signal(signal[31:24])
    );

endmodule
```

intro7seg2 モジュールは、4 桁の 7 セグメント LED ディスプレ イを駆動するように設計されています。ディスプレ イには、各桁に8つのLEDセグメントがあり、各セグメントは特定の文字を表示するためにオンまたはオフになります。このモジュールは、各桁のどのセグメントをオンにして正しい数字を表示させるかを制御する信号を生成します。

このモジュールは、クロック入力信号を受け取り、3つの出力信号を生成します：

signal_out： signal_out：選択された桁のLEDを駆動する8ビット信号。 セレクター： 表示する桁を選択する4ビット信号。 clk_div： 入力クロック信号の分周版。 wire clk_div と wire [31:0] 信号線は、モジュールで使用される2つのワイヤーを宣言します。clk_div線はクロック分周器の出力を格納するために使用され、信号線は4桁すべてのLED信号を結合したものを格納するため

に使用されます。

clock_dividerモジュールは、次のコードでインスタンス化されます：

.clock(clock)、 .clk_div(clk_div));

このモジュールは、入力クロック信号 clock を受けて、より遅いクロック信号 clk_div を生成し、この信号は counter100000000 モジュールを駆動するために使用される。

counter100000000モジュールは、クロックサイクル数をカウントし、現在のカウント値を表示するために使用される14ビットバイナリカウント信号（count）を生成するために使用されます。convertdecimalモジュールは、この2進カウント信号を受け取り、4つのBCD（2進符号化10進）信号（n1, n2, n3, n4）に変換し、それぞれが表示の1桁を表現する。

digitdisplayモジュールは、対応するBCD信号に基づいて、4つの桁のそれぞれのLED信号を生成します。digitdisplayモジュールは4つのインスタンスがあり、各桁に1つずつ対応する。

always @(posedge clock) begin で始まる always ブロックは、クロック信号 (posedge clock) の立ち上がりエッジでトリガされます。このブロックは、2ビットのセレクタカウンタ (selector_counter) をインクリメントし、カウンタの現在値に基づいて selector 出力を設定します。always ブロックの case 文は、2クロックサイクルごとに selector 出力を異なる値に設定し、これにより各桁を順次表示することができます。

always @ (posedge clock) begin で始まる2番目の always ブロックも、クロック信号の立ち上がりエッジでトリガされる。このブロックは、セレクタカウンタの値に基づいてsignal出力の適切な部分を選択し、signal_outに代入する。これにより、選択された桁に対して正しいLED信号が表示される。

全体として、このモジュールは、4桁の7セグメントLEDディスプレイを制御し、入力クロック信号に基づいて現在のカウント値を表示する信号を生成するように設計されています。このモジュールでは、クロック分周器、カウンタ、BCDコンバータ、桁数表示モジュールを使用して、必要な信号を生成しています。

2.1.2 counter100000000 module

```
module counter100000000 (
    input clock,
 output reg [14:0] count);

 initial begin
```

```
count = 0;
end

 always @(posedge clock)
begin
if (count == 14'd9999)
begin
count <= 14'd0000;
end
else
begin
count <= count + 1;
end
end
endmodule
```

本モジュールは、入力クロック信号（clock）を受けて、クロック信号の立ち上がりエッジごとに1ずつ増加する14ビットバイナリカウント信号（count）を生成します。カウントが最大値である9999に達すると、ゼロにリセットされ、再びカウントを開始します。

入力クロック線は，クロック信号を入力とする入力ポートを宣言しています。output reg [14:0] count lineは、2値のカウント信号を出力するための出力ポートを宣言しています。

initialブロックは、シミュレーション開始時にcount変数を0に初期化するために使用されます。

always @(posedge clock)で始まるalwaysブロックは、クロック信号(posedge clock)の立ち上がりエッジでトリガされます。このブロックは、count変数の現在値をチェックし、最大値である9999に達したかどうかによって、1つインクリメントするか、ゼロにリセットする。

alwaysブロックのif文は、countの現在値が9999に等しいかどうかをチェックする。もしそうなら、count ¡= 14'd0000;という行でcount変数をゼロに設定する。もしcountの現在値が9999に等しくなければ、elseブロックが実行され、count ¡= count + 1;という行でcount変数が1つインクリメントされる。

全体として、このモジュールは、クロック信号の各立ち上がりエッジで1ずつ増加し、最大値である9999に達するとゼロにリセットされるバイナリカウント信号を生成するように設計されています。このタイプのカウンターは、発生したクロックサイクルの数を追跡するために、デジタル回路で

一般的に使用されています。クロック信号の立ち上がりエッジでトリガーされる常時ブロックの使用は、Verilogでカウンターを実装する標準的な方法である。

2.1.3 convertdecimal module

```
  module convertdecimal(
input [13:0] binary,
output [3:0] num1,
output [3:0] num2,
output [3:0] num3,
output [3:0] num4 );

assign num1 = binary%10;
assign num2 = ((binary - num1)/ 10) % 10;
assign num3 = ((binary - num1 - num2 * 10)/ 100) % 10;
assign num4 = ((binary - num1 - num2 * 10 - num3 * 100)/ 1000) % 10;


endmodule
```

convertdecimalモジュールは，14ビット形式の2進数（バイナリ）を4ビットBCD（2進符号化10進数）数値のセット（num1，num2，num3，num4）に変換するように設計されています。出力されるnum信号のそれぞれは、入力された2進数のBCD表現の4桁のうちの1桁を表します。

このモジュールには、outputキーワードを使用して宣言された4つの出力ポートが含まれています。num1、num2、num3、num4出力信号は、それぞれ4ビット幅で、BCD桁を出力するために使用されます。

このモジュールでは、assignキーワードを使用して、BCD桁を出力信号に割り当てています。BCD変換のロジックは、一連のモジュロ演算と整数分割演算を使用して実装されています。各Num出力信号は、2進数入力値から特定の桁を抽出し、残りの桁を適切な10のべき乗で割ることによって計算されます。

BCD変換のロジックは次のように実装されています：

num1には、2進数値を10で割った余りが代入されます（2進数％10）。
num2は，num1の桁を減算したバイナリ値を10で割った余りを代入する

（（binary - num1） / 10） % 10）。
num3は、num1桁とnum2桁を引いた2進数を100で割った余り
(((binary - num1 - num2 * 10) / 100) % 10)
となります。
num4は、num1、num2、num3の3桁を引いた2進数を1000で割った余り
(((binary - num1 - num2 * 10 - num3 * 100) / 1000) % 10)
となります。

2.1.4 clockdivider module

```
    module clock_divider (
    input clock,
    output clk_div
);

    reg [8:0] clk_counter;
    reg clk_div_reg;

    always @(posedge clock) begin
        clk_counter <= clk_counter + 1;
        if (clk_counter == 9'd150) begin // Divide by 300
            clk_div_reg <= ~clk_div_reg;
            clk_counter <= 9'd0;
        end
    end

    assign clk_div = clk_div_reg;

endmodule
```

clock_dividerモジュールは、入力されたクロック信号から、より遅いクロック信号を生成するために、より大きな回路で使用されます。この低速のクロック信号を用いてカウンタ（counter100000000）を駆動し、桁表示信号の変化速度の300分の1の速度で0から9999までカウントする。

clock_dividerモジュールによって生成されたより遅いクロック信号をカウンターの駆動に使用することで、カウンターは桁表示信号が300回変化するごとに1回だけインクリメントします。これにより、カウンターは、桁表

示信号が変化する速度の300倍遅い速度でカウントすることになり、カウンターは4つのディスプレイすべてにカウントを正しく表示するために必要となる。

カウンターの出力は10進数に変換され（convertdecimalモジュール）、その10進数で4桁すべての7セグメントディスプレイ（digitdisplayモジュール）を駆動します。カウンターは、桁表示信号が変化する速度の300倍遅い速度でカウントしているため、カウンターが再び更新される前に4つのディスプレイすべてに桁が表示され、4つのディスプレイすべてにカウントが正しく表示されることが保証される。

全体として、clock_dividerモジュールは、桁表示信号に同期したより遅いクロック信号を生成することで、カウンターを桁表示信号に同期させるのに役立ちます。これにより、カウンターは桁表示信号が変化する速度の300倍遅い速度でカウントすることになり、カウンターは4つのディスプレイにカウントを正しく表示するために必要なのです。

2.1.5 digitdisplay module

digitdisplayはintro7segモジュールと同じです。

3 課題3

課題2の回路にボード上のスイッチからの入力を追加し，スイッチを押す度にカウントアップを停止／再開できるようにせよ．

(ヒント) スイッチのチャタリングを除去する必要がある

3.1 コードとその説明

```
    module intro7seg3 (
    input clock,
 input switch,
    output reg [7:0] signal_out,
    output reg [3:0] selector
);
    wire clk_div;
 wire [31:0] signal;

    // Clock Divider
    clock_divider clk_div_inst (
```

```
        .clock(clock),
        .clk_div(clk_div)
    );

    wire [13:0] count;
    wire [3:0] n1, n2, n3, n4;
 wire ispressed;
reg enabled;

initial begin
enabled = 1'b1;
end
    // Counter for selector
    reg [1:0] selector_counter;

    always @(posedge clock) begin
        selector_counter <= selector_counter + 1;
        case (selector_counter)
            2'b00: selector <= 4'b1110;
            2'b01: selector <= 4'b1101;
            2'b10: selector <= 4'b1011;
            2'b11: selector <= 4'b0111;
        endcase
    end

 always @ (posedge clock) begin
case (selector_counter)
      2'b00: signal_out <= signal[7:0];
            2'b01: signal_out <= signal[15:8];
            2'b10: signal_out <= signal[23:16];
            2'b11: signal_out <= signal[31:24];
        endcase
    end


    // Counter
    counter100000000 counter_inst (
```

```
        .clock(clk_div & enabled),
        .count(count)
    );
// setup chatter avoidance
chattercounter(.clock(clk_div), .switch(switch), .ispressed(ispressed));

    // Decimal converter
    convertdecimal decimal_inst (
        .binary(count),
        .num1(n1),
        .num2(n2),
        .num3(n3),
        .num4(n4)
    );

    digitdisplay dd1 (
        .number(n1),
        .signal(signal[7:0])
    );

    digitdisplay dd2 (
        .number(n2),
        .signal(signal[15:8])
    );

    digitdisplay dd3 (
        .number(n3),
        .signal(signal[23:16])
    );

    digitdisplay dd4 (
        .number(n4),
        .signal(signal[31:24])
    );

   always @(negedge ispressed) begin
enabled <= !enabled;
```

```
end
endmodule
```

intro7seg3 モジュールは intro7seg2 モジュールを拡張し、カウンタで使用するクロックの停止と開始の機能を追加しています。スイッチという入力が追加されており、これは押しボタン式のスイッチと思われ、押すとイネーブルレジスタが1と0の間で切り替わります。

しかし、スイッチが押されると、スイッチバウンスやチャタリングと呼ばれる現象が発生し、最終状態に落ち着くまでに、スイッチが何度も急激な出力遷移を起こすことがあります。これにより、複数のトリガーが発生し、出力に誤差が生じることがあります。この問題を解決するために、チャタリング・カウンター・モジュールは、スイッチのデバウンスに使用され、一定時間内の急激な状態変化をフィルタリングして安定した出力を提供します。チャタカウンターモジュールの出力は、ispressedという配線に割り当てられています。

enabledレジスタは、counter100000000モジュールのクロック入力に使用され、enabledがHighの時のみクロック信号を通過させるようになっています。これにより、スイッチを押すことでクロックの停止と起動を行うことができる。スイッチを押すと、イネーブルドレジスタがトグルされ、クロック信号がカウンターに到達しないようにブロックされます。そのため、カウンターはカウントを停止し、一定の値を表示します。スイッチを離すと、イネーブルドレジスタが再びトグルされ、クロック信号がカウンターに通過できるようになり、カウントが再開されます。

3.1.1 chattercounter module

```
    module chattercounter(
input clock,
input switch,
output reg ispressed );

reg [3:0] count;

initial begin
count = 4'b0000;
end
```

```
// count switch pressed time
always @(posedge clock) begin
if ( !switch ) begin // if switch is pressed
count <= count + 1;
end
else begin
count <= 4'b0000;
end
end

// if switch is pressed for long
always @(posedge clock) begin
if (count == 4'b0011) begin
ispressed <= 1;
end

if (count == 0) begin
ispressed <= 0;
end
end

endmodule
```

本モジュールは，押しボタン式スイッチの多重トリガーを引き起こすノイズやチャタリングを防止するために，押しボタン式スイッチをデバウンスするために使用します。このモジュールは、クロック信号、スイッチ信号、出力線ispressedの3つの入力を受け取ります。ispressedワイヤーは、スイッチが押されているかどうかを示すために使用されます。

このモジュールには、初期状態で0に設定されているcountというレジスタが含まれています。countレジスタは、スイッチが押されているクロックサイクルの数をカウントします。このモジュールには2つの常時ブロックがあり、どちらもクロック信号のプラスエッジでトリガーされます。

最初の常時ブロックは、スイッチが押さえられているクロックサイクル数をカウントする。スイッチ信号がLow（スイッチが押されていることを意味する）かどうかをチェックし、Lowの場合はカウントレジスターを1つインクリメントします。それ以外の場合、スイッチ信号がHigh（スイッチが押されていないことを意味する）なら、カウントレジスターは0に設定されます。

2番目の常時ブロックは、カウント・レジスタがある閾値に達したかどうかをチェックし、この場合、閾値は3に設定されます。これは、スイッチが一定時間（この場合は3クロックサイクル）押されたままであり、もはやチャタリングとはみなされないことを意味します。カウントレジスタが3になると、ispressedワイヤが1にセットされ、スイッチが押されていることを示します。一方、カウントレジスターが0になると、スイッチが解放されたことを意味し、ispressedワイヤが0に設定され、スイッチが押されていないことを示す。

全体として、このモジュールはスイッチ信号をデバウンスし、ispressedワイヤがスイッチが押されているかどうかを示すクリーンな信号のみを出力することを保証するために使用されます。