平成 29 年度　計算機科学実験及演習 3A
(3 回生前期学生実験　HW)
導入レポート

提出期限：4 月 20 日
提出日：4 月 20 日

1029272870

谷　勇輝

目次

0 概要

0.1 実験目的

7SEG LED 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。7SEG LED 表示のためのデコーダ、及びそれを利用したカウンタの設計を通して、論理システムの講義および実験 2 ハードウェアの復習を行う。また、HDL 設計、CAD ツール使用の習熟を図る。

0.2 実験環境

日時： 2017 年 4 月 13 日(木)、14 日(金)
場所： 京都大学 総合研究 7 号館 計算機演習室 1
環境：

FPGA ボード：PowerMedusa MU500-RX/RK

FPGA：Altera 社 Cyclone IV　EP4CE30F23I7N クロック：20MHz 発信機 拡張ボード：MU500-7SEG（今回は未使用）

CAD ツール： Altera Quartus II 13.0sp1

ハードウェア記述言語(HDL)：Verilog HDL 2001

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、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 原理・方針

1.1.1 7SEG LED

FPGA ボード PowerMedusa の 1 つの 7SEG LED は 8 つの LED から構成される。それぞれの LED は FPGA からの出力を受けて独立に動作する（正論理で点灯）。16 進数を表示するためには、表示する数字に対応する 8 つの出力を制御する必要がある。

表示までの流れを図 1 に示す。

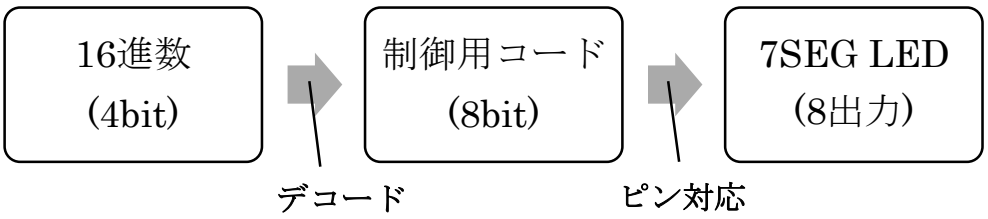


図 1　表示までの流れ

また、PowerMedusa には 8 つの 7SEG LED が搭載されており、どの 7SEG LED に出力を行うかはセレクタ出力（負論理）で制御を行う。

今回の実験の課題 1 では、ボードの最も左の 7SEG LED（セレクタ出力対応ピン：E6）を使用した。

制御用コード[7:0] は各ビットがそれぞれ図 2 に示す位置の LED を制御することとした。

以上の仕様に従って表示を行う。各 16 進数に対応する制御用コード、7SEG LED の表示の対応は表 1 の通りである。

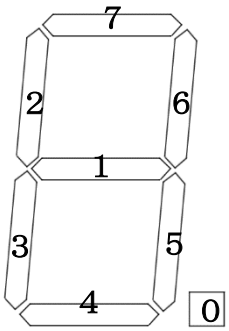
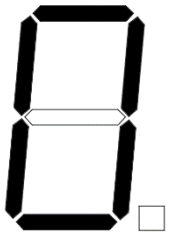
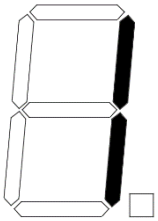
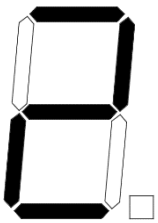
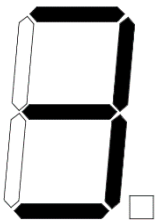
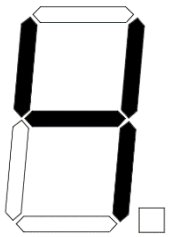
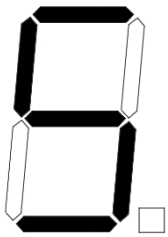
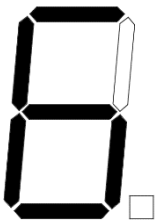
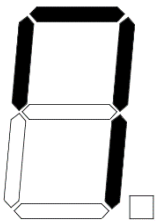
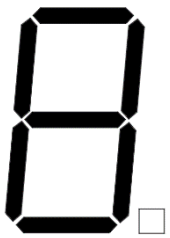
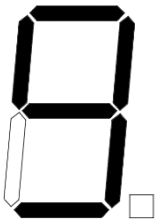
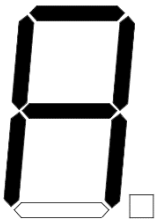
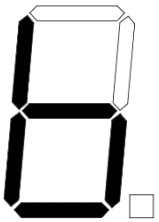
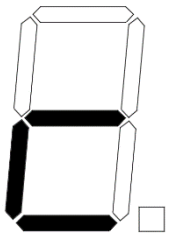
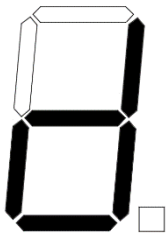
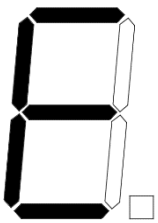
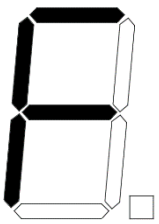


図 2　制御用コード bit 対応

表 1　16 進数,制御用コード,表示の対応

16 進数	0	1	2	3
制御用コード	11111100	01100000	10011010	11110010
表示				
16 進数	4	5	6	7
制御用コード	01100110	10110110	10111110	11100100
表示				
16 進数	8	9	A	B
制御用コード	11111110	11110110	11101110	00111110
表示				
16 進数	C	D	E	F
制御用コード	00011010	01111010	10011110	10001110
表示				

1.2 設計

Verlog HDL を用いて CAD ツール上で論理回路の設計を行った後、FPGA 入出力ピンの割り当てを行った。

課題１で設計したモジュールは以下の１つである。

- LED7segDecoder
  16 進数(4bit)を制御用コード(8bit)に変換するデコーダ

1.2.1 LED7segDecoder の設計

LED7segDecoder は、16 進数(4bit)を制御用コード(8bit)にデコードするモジュールである。HDL コードを図３に示す。

入力は 4bit ポート number の１本である。7SEG LED に表示したい 16 進数を入力する。

出力は 8bit ポート led7seg、1bit ポート gnd の２本である。led7seg はデコード結果である制御用コードを出力する。gnd は常に 0 の値を持つ出力で、セレクタ出力として使用する。

内部的には、入力として受け取った 16 進数によって case 分岐を行い、対応する制御用コードを出力している。入力は 4bit であるので 0 から F までの値しか取らないが、フールプルーフとしてドットのみの表示を default に設定している。

```
module LED7segDecoder(
   input [3:0] number,
   output reg [7:0] led7seg,
   output gnd = 0);

   always @*
   begin
      case(number)
         4'h0 : led7seg = 8'b11111100;
         4'h1 : led7seg = 8'b01100000;
         4'h2 : led7seg = 8'b11011010;
         4'h3 : led7seg = 8'b11110010;
         4'h4 : led7seg = 8'b01100110;
         4'h5 : led7seg = 8'b10110110;
         4'h6 : led7seg = 8'b10111110;
         4'h7 : led7seg = 8'b11100100;
         4'h8 : led7seg = 8'b11111110;
         4'h9 : led7seg = 8'b11110110;
         4'hA : led7seg = 8'b11101110;
         4'hB : led7seg = 8'b00111110;
         4'hC : led7seg = 8'b00011010;
         4'hD : led7seg = 8'b01111010;
         4'hE : led7seg = 8'b10011110;
         4'hF : led7seg = 8'b10001110;
         default : led7seg = 8'b00000001;
      endcase
   end
endmodule
```

図 3　LED7segDecorder

1.2.2 FPGA ピン割当て

FPGA のピン割当てを表２に従って行った。

表 2　ピンの割当て

ポート種類	ポート名	ピン	接続先
入力	number [3:0]	F14,E14,B14,A14	ロータリースイッチ（HEX-A）
出力	led7seg [7:0]	A3,B6,A6,A5,B4,B3,A4,B5	7SEG LED（A）
	gnd	E6	7SEG LED セレクタ

課題１では入力にロータリースイッチを用いた。ロータリースイッチは０～F の 16 進数を直接入力することができ、今回のモジュールの入力として最適である。

1.3 動作確認

シミュレーションと FPGA ボードでの実機テストを行い、設計した論理回路が正しく動作することを確認した。

1.3.1　シミュレーション

テストベンチを作成し、CAD ツール上のシミュレータで論理回路の動作を観察した。

入力 number には 4bit がとりうる全ての入力（０からＦ）を 100ns 間隔で与えた（複合条件網羅）。シミュレータを動作させ、出力と制御用コードの対応を見て、設計した論理回路に誤りが無いことを確認した。

1.3.2　実機確認

FPGA に設計した論理回路を書き込み、実機で動作を観察した。

入力として設定したロータリースイッチが０からＦまで切り替わるに従い、7SEG LED が正しく点灯することが確認できた。

1.4. 考察

今回設計した 7SEG LED 表示用デコーダは、今後ハードウェアの設計を行う上で非常に有用である。LED7segDecoder モジュールの出力に適切な LED 表示用ピンを割当て、入力には表示したい数を示す信号を入れるだけで良い。数字の表示が効率的にできるようになり、設計の工数を削減することに役立つだろう。

2. 課題２

10 進数 4 桁の数字を表示して 1 クロックで 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。

2.1. 原理・方針

2.1.1　UI ボードと拡張ボードの 7SEG LED

今回の実験で使用している FPGA ボードには、UI ボードと拡張ボードの両方に 7SEG LED が配置されている。

UI ボードの 7SEG LED は、横に並んだ４つの表示盤が組を成しており、それが２組、計８つの表示盤がある。組を成した表示盤は同時に同じ点灯パターンしか設定することができず、どの表示盤を点灯させるかはセレクタ出力のパターンが管理している。

拡張ボードの 7SEG LED は縦に並んだ４つの表示盤が組を成しており、それが 16 組、計 64 個の表示盤がある。基本的な仕組みは UI ボードのものと同じであるが、横に並んだ表示盤はそれぞれ他の組に属しているため、同時に別のパターンの表示を行うことができる。

今回の実験では、UI ボード側の 7SEG LED に４桁の数字を表示する仕様とした。拡張ボード側のものを使えば簡単であるが、今後 UI ボード側に 10 進数の数字を表示できるようにしておきたいというのが主な理由である。また、拡張ボードで実現すればピン接続数が 36 となるのに対し、UI ボード上で実現すれば９つのピン接続で表示が可能になる。

４つの表示盤は同時に別パターンを表示することはできない。そこで、細かい周期でセレクタ出力と表示パターンを切り替え、「同時には表示されていないが、人間の目には同時に点灯しているように見える」状態をつくる方針で実装を行った。

2.1.2　分周

PowerMedusa に標準搭載されているクロック(ピン番号 : A12)の周波数は 20MHz である。また、UI ボードに配置されたクロック用ロータリースイッ

チを使用して 1Hz～40MHz の 15 段階のクロック(ピン番号：B12)をボード側で発生させることができる。

しかし、今回のように一つの論理回路で複数種類の周期のクロックが必要となることも少なくない。その際に必要となるのが**分周**である。分周は図４のようにカウンタの溢れ出しを利用することで行うことができる。

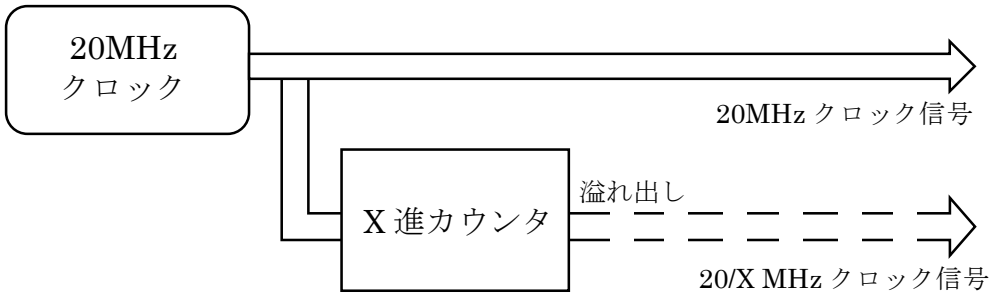


図 4　カウンタを使った分周

課題２では、以下の２種類のクロックが必要となる。

- カウントアップに使用する低周期のクロック
- LED 表示の切り替えに使用する高周期のクロック

2.2. 設計

Verlog HDL を用いて CAD ツール上で論理回路の設計を行った後、FPGA 入出力ピンの割り当てを行った。

課題２で設計したモジュールは以下の５つである。

- counter10x4
  10 進４桁の同期式カウンタ。トップモジュール
- LED4set
  UI ボードの隣り合った 4 つの 7SEG LED にそれぞれ独立した表示をするためのモジュール
- counter4
  ４進カウンタ
- counter10
  10 進カウンタ
- counter2p16
  2^16 進カウンタ

2.2.1　counter4, counter10, counter2p16

counterX は X 進カウンタ回路のモジュールである。counter4 の HDL コードを図 5 に示す。その他のカウンタはコード中の 4 を表す数字を X を表す数字に変更することで実現できる。

入力は 1bit ポート clock と 1bit ポート plus の 2 本である。clock が立ち上がる度に plus だけカウントが上がる。

出力は 2bit ポート count と、1bit ポート overflow の 2 本である。count は現在のカウント値を示し、overflow は溢れが発生した際に立ち上がる。

```
module counter4(
   input clock,
   input plus,
   output reg [1:0] count,
   output reg overflow);

   always @(posedge clock)
   begin
      overflow <= 0;
      if(plus == 1) begin
         if(count == 2'b11) begin
            overflow <= 1;
            count <= 2'b00;
         end else begin
            count <= count + 2'b01;
         end
      end
   end
endmodule
```

図 5　count4

overflow は count が X 回増える度に立ち上がる。従って、周期 A Hz のクロック信号を clock に入力し、plus を常に 1 とすると、overflow は周期 A/X Hz のクロック信号として使用できる。

2.2.2　LED4set

LED4set は UI ボードの隣り合った 4 つの 7SEG LED にそれぞれ独立した表示をするためのモジュールである。HDL コードを図 6 に示す。

```
module LED4set(
   input CK,
   input [7:0] LED3,LED2,LED1,LED0,
   output reg [7:0] LED,
   output reg [3:0] selecters);

   wire vdd;
   assign vdd = 1;
   wire [1:0] num;

   counter4 counter(.clock(CK),.plus(vdd),
                                    .count(num));
   always @(posedge CK)
   begin
      case (num)
         2'b00: begin
                selecters <= 4'b0111;
                LED <= LED3;
            end
         2'b01: begin
                selecters <= 4'b1011;
                LED <= LED2;
            end
         2'b10: begin
                selecters <= 4'b1101;
                LED <= LED1;
            end
         2'b11: begin
                selecters <= 4'b1110;
                LED <= LED0;
            end
         default : begin
                selecters <= 4'b0000;
                LED <= 8'b00000001;
            end
      endcase
   end
endmodule
```

図 6　LED4set

入力は 1bit ポート CK と、8bit ポート LED3、LED2、LED1、LED0 の計５本である。CK には高周波クロック信号を入力する。LED3~0 には、それぞれの 7SEG LED に表示したいパターンの制御用コードを入力する。

出力は 8bit ポート LED と 4bit ポート selecters の２本である。LED は 7SEG LED の最終的な制御用コードを出力する。selecters はセレクタ出力を束ねた出力である。

内部的には４進カウンタの値によって出力する制御用コードとセレクタを切り替えることで実現した。

このモジュールにより設計者は、表示タイミング等を意識することなく、４つの隣り合った 7SEG LED に一見同時に異なった表示を出力することができるようになった。

2.2.3 counter10x4

counter10x4 は 10 進数４桁同期式カウンタのトップモジュールである。HDL コードを図７に示す。

入力は 1bit ポート clock の１本である。高周波クロックを入力する。

出力は 8bit ポート LED、4bit ポート selecters、1bit ポート overflow の３本である。LED と selecters は 7SEG LED の制御用コードとセレクタ出力列であり、overflow はカウントの溢れを検出する。

内部の主な流れを図８に示した。4 つの 10 進カウンタはクロック入力に従って同期的に変化し、LED7segDecoder と LED4set を経て表示に至る。

今回は入力 clock として 40MHz～10kHz 位の高周波クロックを想定して設計を行った。10 進カウンタに入力するクロックは clock を 2 の 16 乗分の１（=1/65,536）で分周したものとした。また、LED4set のクロックとして clock をそのまま入力すると 10MHz を超えた辺りから、文字盤の表示が不明瞭になってしまう。これは表示の切り替わりが高速すぎるあまり、十分に発光する前に入力が切れてしまうためである。そこで、こちらのクロックも 1/4 に分周したものを使用することにした。

```
module counter10x4(
        input clock,
        output [7:0] LED,
        output [3:0] selecters,
        output overflow);

        wire CKx2p16,CKx4;
        wire plus[0:3];
        wire [3:0] num[0:3];
        wire [7:0] digitLED[0:3];

        wire vdd;
        assign vdd = 1;
        counter2p16 dev(.clock(clock),.plus(vdd),.overflow(CKx2p16));
        counter4 dev2(.clock(clock),.plus(vdd),.overflow(CKx4));

        assign plus[0] = vdd;
        assign plus[1] = plus[0] & (num[0] == 4'd9);
        assign plus[2] = plus[1] & (num[1] ==4'd9);
        assign plus[3] = plus[2] & (num[2]==4'd9);

        counter10 c0(.clock(CKx2p16),.plus(plus[0]),.count(num[0]));
        counter10 c1(.clock(CKx2p16),.plus(plus[1]),.count(num[1]));
        counter10 c2(.clock(CKx2p16),.plus(plus[2]),.count(num[2]));
        counter10 c3(.clock(CKx2p16),.plus(plus[3]),.count(num[3]).overflow(overflow);

        LED7segDecoder d0(.number(num[0]),.led7seg(digitLED[0]));
        LED7segDecoder d1(.number(num[1]),.led7seg(digitLED[1]));
        LED7segDecoder d2(.number(num[2]),.led7seg(digitLED[2]));
        LED7segDecoder d3(.number(num[3]),.led7seg(digitLED[3]));

        LED4set gath(.CK(CKx4),.LED3(digitLED[3]),.LED2(digitLED[2]),
                     .LED1(digitLED[1]),.LED0(digitLED[0]),
                     .LED(LED),.selecters(selecters));

endmodule
```

図 7　counter10x4

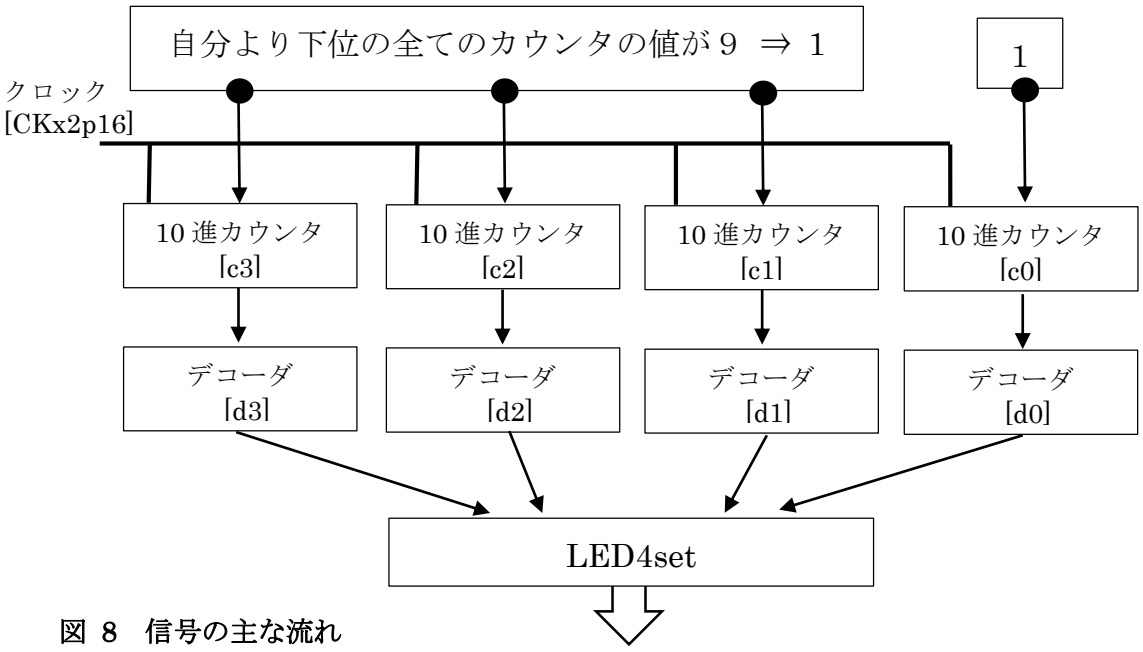


図 8　信号の主な流れ

2.2.4　FPGA ピン割当て

FPGA のピン割当てを表 3 に従って行った。

表 3　FPGA ピン割当て

ポート種類	ポート名	ピン	接続先
入力	clock	B12	UI ボードからのクロック
出力	LED [7:0]	A3,B6,A6,A5,B4,B3,A4,B5	7SEG LED（A~D）
	selecters[3:0]	E6,E5,C4,C3	7SEG LED セレクタ
	overflow	A8	LED ランプ（0）

2.3. 動作確認

コンパイル結果から回路規模と動作可能速度を確認した。また、FPGA ボードでの実機テストを行い、設計した論理回路が正しく動作することを確認した。

2.3.1　性能

- 回路規模：　　総論理素子数 90 個（FPGA 占有率 1%以下）
- 動作可能速度：　367.92 MHz（10ns 周期クロックを入力）

2.3.2　実機確認

FPGA に設計した論理回路を書き込み、実機で動作を観察した。

入力は、クロック用ロータリースイッチを使用して 40MHz (0)～1.22kHz(B)の 12 種類の周波数を順に入力した。

7SEG LED には 4 桁の十進数が表示され、カウントアップも正常に動作した。(図 9)

図 9　7SEG LED 表示

2.4. 考察

UI ボードの組になった LED 表示盤に独立した数を表示できた。出力 overflow を使用すれば、複数個の count10x4 を繋ぎ合わせて 4 桁以上の数の表示も可能である（同期式とするには最上位が 9 かどうかの出力が別途必要である）。

3. 課題３

課題 2 の回路にプッシュ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。(スイッチのチャタリングに対応する必要があるので注意すること。)

3.1. 原理・方針

3.1.1　チャタリング

物理スイッチは電気的接触部の振動等の原因によって、切り替わりの際に ON 状態と OFF 状態が交互に繰り返される**チャタリング**という現象が起こる。そのため、スイッチから送られる信号をそのまま観測して動作を行うと、一回の切り替わりにつき複数回の立ち上がりが観測され、意図しない挙動が発生する場合がある。

チャタリングを阻止するための主な方法として、**ディレイ法**と**サンプリング法**がある。

ディレイ法は立ち上がりを観測した後、一定時間の立ち上がりを無視することによってチャタリングをキャンセルする方式である。

サンプリング法はスイッチからの入力をそのまま使用せず、一定間隔（チャタリングの時間以上の間隔）でサンプリングした信号を入力とすることでチャタリングを除去する方式である。

今回の実験では安定性が高くチャタリング除去として広く使用されている**サンプリング法**で実装を行うことにした。

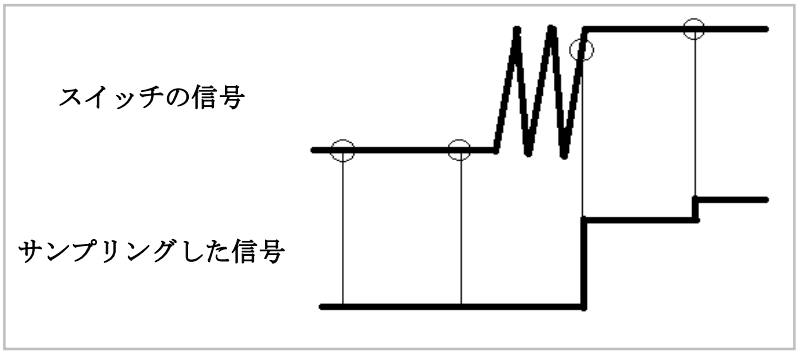


図 10　サンプリングによるチャタリングの除去

サンプリングによって立ち上がりを１回にする

3.2. 設計

Verlog HDL を用いて CAD ツール上で論理回路の設計を行った後、FPGA 入出力ピンの割り当てを行った。

課題２のトップモジュールの counter10x4 に追加実装を行った。

3.2.1　counter10x4 への追加

- 入力に 1bit ポート start_stop を追加した。
- start_stop からチャタリングを除去した信号を保持する reg start_stop_RMch を定義した。
- カウントアップを有効かどうかを判定する reg run を定義した。
- サンプリングを行う以下の always 文を追加した。サンプリング間隔がチャタリング時間より長くなるよう、十分に分周したクロックを用いる必要がある。今回は 2^16 分周クロックを使用した。

```
always @(posedge CKx2p16) begin
   start_stop_RMch <= start_stop;
end
```

- スタート・ストップの信号を受けて動作・停止の切り替えを行う以下の always 節を追加した。

```
always @(negedge start_stop_RMch) begin
   run <= (run==1)? 0 : 1;
end
```

- 最下位の 10 進カウンタ入力である num[0]への接続を vdd から run に変更した。

3.2.2　FPGA ピン割当て

追加した start_stop ポート以外の割当ては、課題２から変更はない。

start_stop ポートへの割当ては表 4 のようにした。

表 4

ポート種類	ポート名	ピン	接続先
入力	start_stop	E15	プッシュスイッチ　（SW4）

3.3. 動作確認

コンパイル結果から回路規模と動作可能速度を確認した。また、FPGA ボードでの実機テストを行い、設計した論理回路が正しく動作することを確認した。

3.3.1　性能

- 回路規模：　　総論理素子数 93 個（FPGA 占有率 1%以下）
- 動作可能速度：　368.46 MHz（10ns 周期クロックを入力）

3.3.2　実機確認

FPGA に設計した論理回路を書き込み、実機で動作を観察した。

入力は、クロック用ロータリースイッチを使用して 40MHz (0)～1.22kHz(B)の 12 種類の周波数を順に入力した。

また、そのそれぞれについてプッシュダウンスイッチを数回押し、カウントアップの停止、再開が滑らかに切り替わることを確認した。

3.4. 考察

実機確認後、今回はチャタリング除去機能を外してプッシュダウンスイッチを操作する試みも行った。チャタリング除去機能が無いと、スイッチを押したのに反応せずに動き続けることが頻繁に起きた。それと比較しても、今回実装したチャタリング除去機能は上手く実装できたことが分かる。

今後もスイッチングを行う機会は多いと予想できるので、チャタリング除去機能を提供するモジュールを別途作成しておきたい。

参考文献

ChaN. (2002 年 1 月 30 日). Technical note /チャタリング対策の仕方. 参照先: Electronic Lives Mfg.: http://elm-chan.org/docs/tec/te01.html

杉山進, 田中克彦, 小西聡. (2014). 電気電子回路　-アナログ・ディジタル回路-. コロナ社.