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 2](#_Toc480410554)

[0.1 実験目的 2](#_Toc480410555)

[0.2 実験環境 2](#_Toc480410556)

[1 課題１ 3](#_Toc480410557)

[1.1 原理・方針 3](#_Toc480410558)

[1.1.1 7SEG LED 3](#_Toc480410559)

[1.2 設計 5](#_Toc480410560)

[1.2.1 LED7segDecoderの設計 5](#_Toc480410561)

[1.2.2 FPGAピン割当て 5](#_Toc480410562)

[1.3 動作確認 6](#_Toc480410563)

[1.3.1 シミュレーション 6](#_Toc480410564)

[1.3.2 実機確認 6](#_Toc480410565)

[1.4. 考察 6](#_Toc480410566)

[2. 課題２ 7](#_Toc480410567)

[2.1. 原理・方針 7](#_Toc480410568)

[2.1.1 UIボードと拡張ボードの7SEG LED 7](#_Toc480410569)

[2.1.2 分周 7](#_Toc480410570)

[2.2. 設計 8](#_Toc480410571)

[2.2.1 counter4, counter10, counter2p16 9](#_Toc480410572)

[2.2.2 LED4set 9](#_Toc480410573)

[2.2.3 counter10x4 10](#_Toc480410574)

[2.2.4 FPGAピン割当て 12](#_Toc480410575)

[2.3. 動作確認 12](#_Toc480410576)

[2.3.1　性能 12](#_Toc480410577)

[2.3.2 実機確認 12](#_Toc480410578)

[2.4. 考察 12](#_Toc480410579)

[3. 課題３ 13](#_Toc480410580)

[3.1. 原理・方針 13](#_Toc480410581)

[3.1.1 チャタリング 13](#_Toc480410582)

[3.2. 設計 14](#_Toc480410583)

[3.2.1 counter10x4への追加 14](#_Toc480410584)

[3.2.2 FPGAピン割当て 14](#_Toc480410585)

[3.3. 動作確認 15](#_Toc480410586)

[3.3.1 性能 15](#_Toc480410587)

[3.3.2 実機確認 15](#_Toc480410588)

[3.4. 考察 15](#_Toc480410589)

[参考文献 15](#_Toc480410590)

# 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



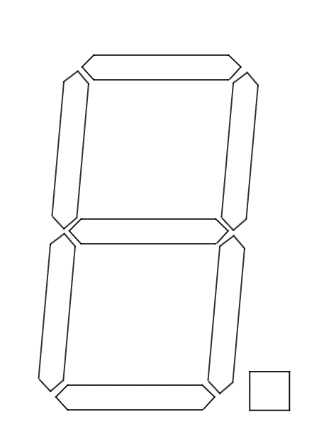
**デコード**

**ピン対応**

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。



**７**

**６**

**５**

**４**

**３**

**２**

**１**

**０**

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 | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber0.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber1.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber2.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber3.png |
| 16進数 | 4 | 5 | 6 | 7 |
| 制御用コード | 01100110 | 10110110 | 10111110 | 11100100 |
| 表示 | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber4.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber5.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber6.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber7.png |
| 16進数 | 8 | 9 | A | B |
| 制御用コード | 11111110 | 11110110 | 11101110 | 00111110 |
| 表示 | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber8.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumber9.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumberA.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumberB.png |
| 16進数 | C | D | E | F |
| 制御用コード | 00011010 | 01111010 | 10011110 | 10001110 |
| 表示 | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumberC.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumberD.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumberE.png | C:\Users\Yuki\Program\projects\le3\Class-le3-HW\reports\intro\7segNumberF.png |

## 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。

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

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。

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。

20MHz

クロック

X進カウンタ

20MHzクロック信号

20/X MHzクロック信号

溢れ出し

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。

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

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。

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。

図 6　LED4set

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

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。

図 7 counter10x4

10進カウンタ[c3]

10進カウンタ[c2]

10進カウンタ[c1]

10進カウンタ[c0]

デコーダ

[d3]

デコーダ

[d2]

デコーダ

[d1]

デコーダ

[d0]

LED4set

自分より下位の全てのカウンタの値が９ ⇒ 1

１

クロック

[CKx2p16]

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 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。

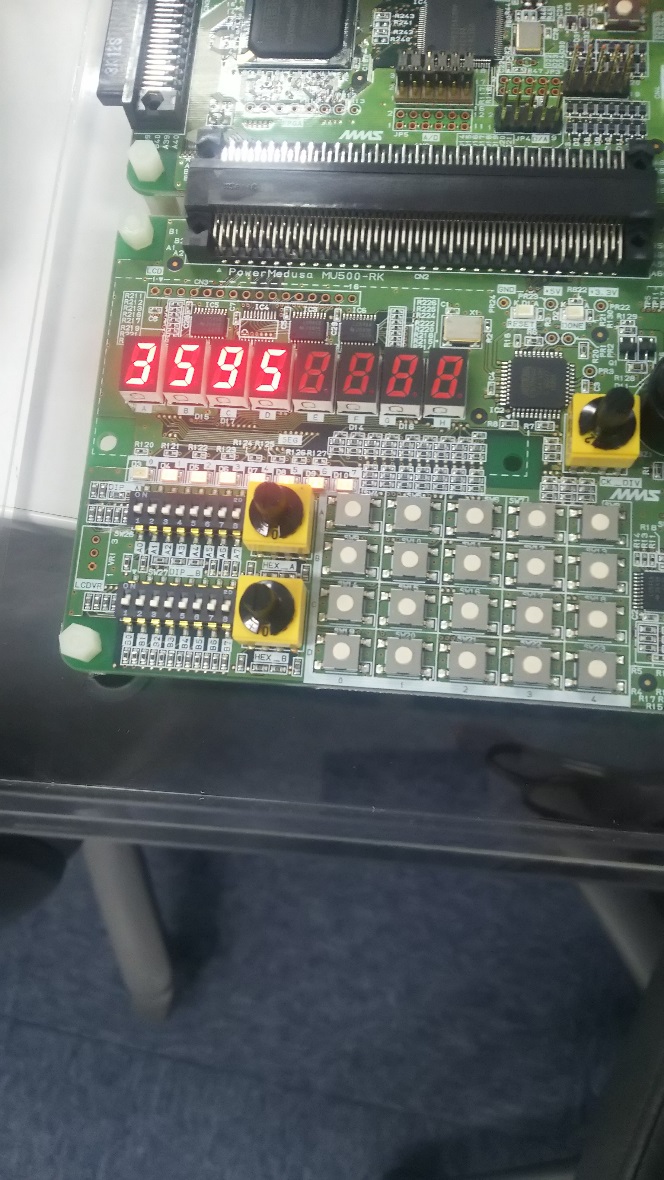
### 2.3.1　性能

* 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)

## 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）。

図 9　7SEG LED表示

# 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。

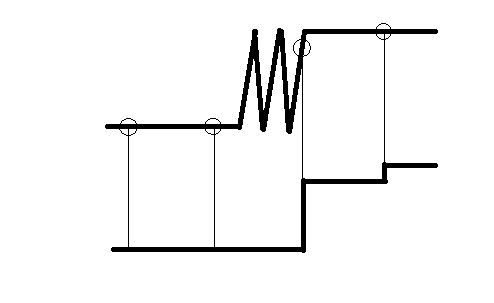
**ディレイ法**は立ち上がりを観測した後、一定時間の立ち上がりを無視することによってチャタリングをキャンセルする方式である。

**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



**スイッチの信号**

**サンプリングした信号**

## 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.