平成 25 年度

卒業研究報告書

研究題目

マイケルソン干渉計型光波長計の改良

富山高等専門学校

電子制御工学科　陣内活実　高松史

指導教官　由井四海

平成 26 年 2 月 28 日提出

目次

1　研究目的

昨年度は、未知波長の半導体レーザーと比較対象のためのHe-Neレーザーの波長を比較して、半導体レーザーの波長を測定するためにマイケルソン干渉計を試作した。

マイケルソン干渉計は、レーザーをビームスプリッターで2つに分岐させ、それぞれの光をミラーで反射させ、その光を重ね合わせてディテクターで受光することで光の波長を測定する。

しかし、振動などによってレーザーがビームスプリッターやミラーに当たる角度が変わると、光学素子間の距離が長いほどずれが大きく生じ、外部からの影響を受けやすい。

また、波長計測では、干渉信号をオシロスコープで測定しこの波形をコンピュータに取り込み、プログラムを利用して波長を算出する処理が必要となっていた。

そこで、今年度は外部からの影響を受けにくくするために、マイケルソン干渉計の小型化するとともに、FPGAボードを用いて干渉信号の処理と波長の算出を目的とした。

図１に今年度の実験手順を示す。マイケルソン干渉計によって生じさせた干渉縞をディテクターによって干渉波に変換する。その波をハイパスフィルタと矩形波変換回路を含む波形整形回路に入力し、変換された矩形波をFPGAボードで作成したカウント表示プログラムに入力して干渉回数を液晶に表示させる。表示された干渉回数を用いて、波長計算を行い、半導体レーザーの未知波長を算出する。

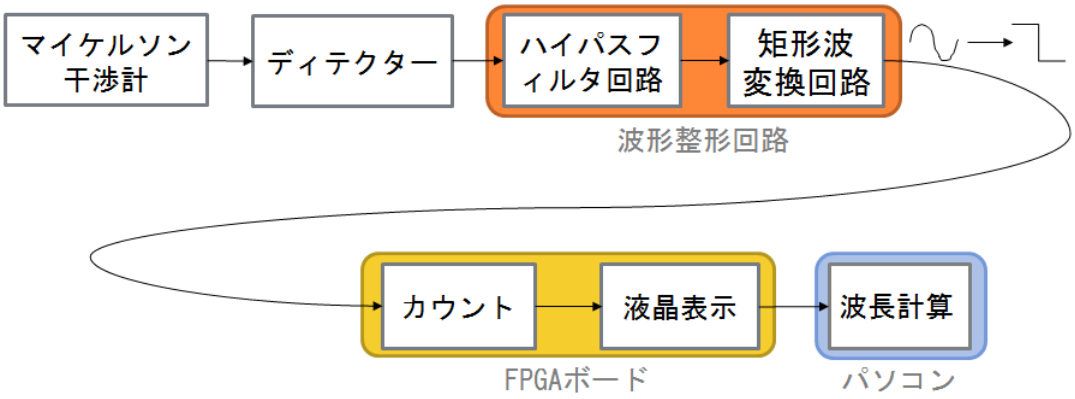

図1　実験手順

2　測定原理

ここではマイケルソン干渉計の原理について説明する。マイケルソン干渉計では波長を測定する際、光が強めあった回数を測定する。

2.1　マイケルソン干渉計の原理

図 2 にマイケルソン干渉計の構成を示す。レーザー光源から出射した光をビームスプリッターで 2 つの光路にわける。その 2 つの光はそれぞれミラーで反射され、再びビームスプリッターで光が交わり、ディテクターに入り、干渉縞ができる。

干渉縞とは、光の波が重なったとき、互いに強めあったり弱めあったりすることによってできる光の明暗の縞模様であり、この縞模様の明暗から光路差がわかる。ミラーの基準位置での光路長を b mm としたとき、光路差は図 2 中のミラーの移動後の光路長 a mm からミラーの基準位置での光路長 b mm の差であらわすことができる。

ここで、光が強めあう回数を n、光路差を x mm とおくと、波長 λ nm は以下の式で示される。

$$\lambda = \frac{x}{n} \quad \cdots\cdots (1)$$

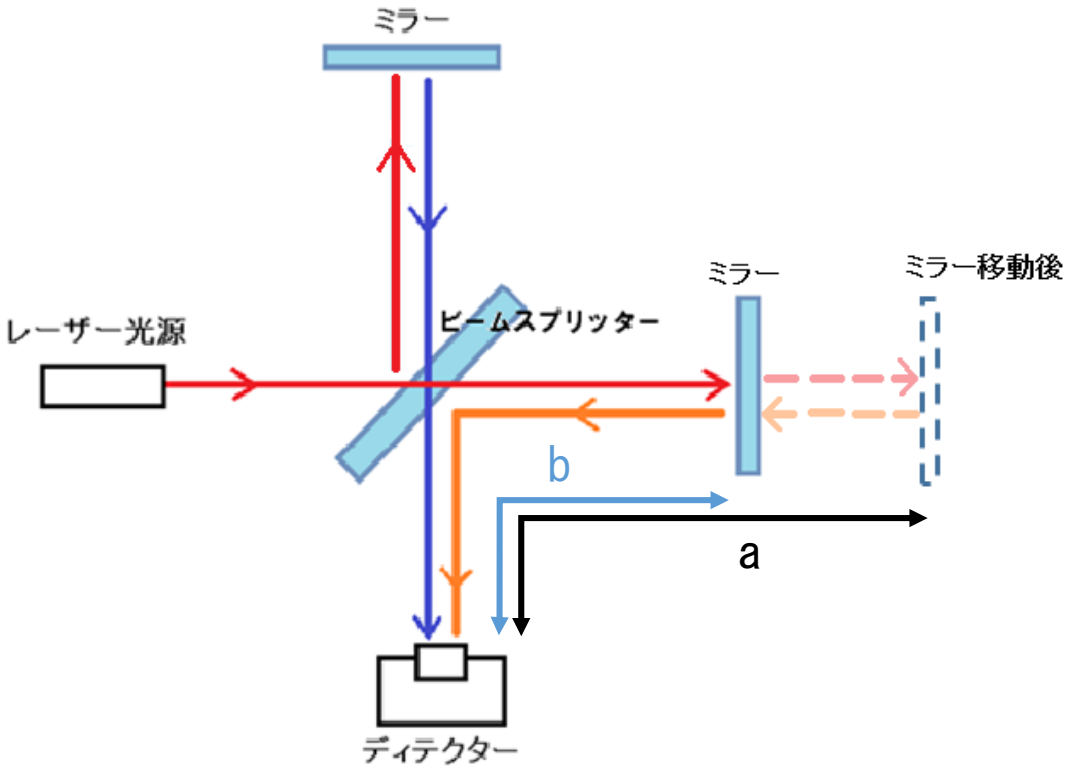

図 2　マイケルソン干渉計の構成

2.2　波の干渉

図 3 に強めあっている波形を、図 4 に弱めあっている波形を示す。

ここで、波長が同じ 2 つの波を考える。なお、この波長は、同じ大きさの振幅をもつ波長とする。図 3 に示すように、2 つの波形の山と山が重なり合うと、2 つの波形の合計により山の高さが 2 倍になり、同じように谷と谷の部分が重なると深さが 2 倍になる。図 4 に示すように、2 つの波形の山と谷が重なり合うと、その部分の合計は 0 になる。振幅の大きさが元の波より大きくなることを「強め合う」といい、元の波の振幅より小さくなることを「弱め合う」といい、これらを総称して干渉という。

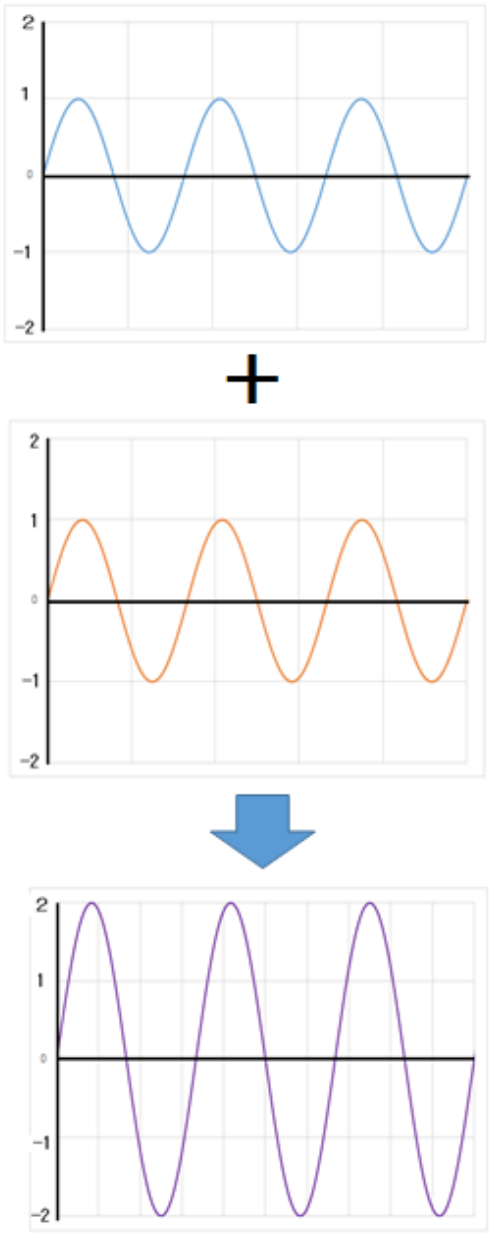

図 3　波形の合成(光路差 0)

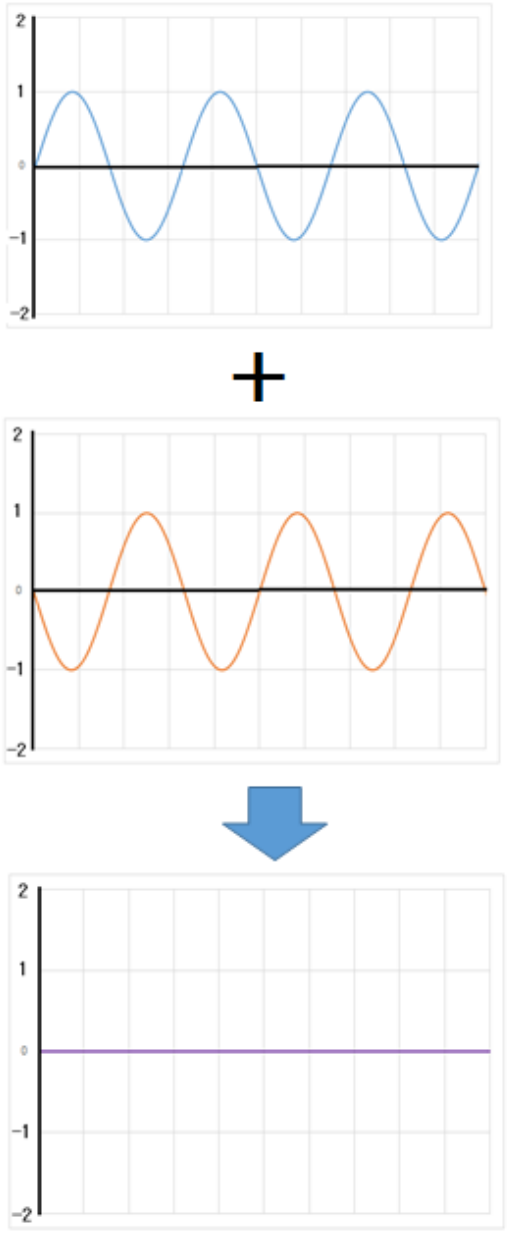

図 4　波形の合成(光路差 λ /2)

(例)　図 5 に任意の光路差をもったマイケルソン干渉計を示す。He-Ne レーザーを照射し、直角プリズムを 4 mm 移動させたときの強め合った回数を求める。ただし、He-Ne レーザーの波長は 633nm とする。

□直角プリズムを移動させる前

$L_A = 2L_2 + 2L_4$

$L_B = 2L_1 + 2L_3$

移動前の光路差 $L_A - L_B = 0$

□直角プリズムを移動させた後（4mm移動させたとき往復で8mm動いたことになる）

$L_A' = 2L_4 + (2L_2 + 8)$

$L_B' = 2L_3 + (2L_1 - 8)$

移動後の光路差 $L_A' - L_B' = 16$

これより光路差は直角プリズムを移動距離の 4 倍になることがわかった。

よって、光路差は16mm変化したことがわかり、(1)式のλと x にそれぞれ代入すると

$$n = \frac{x}{\lambda}$$

$$= \frac{16\times10^{-3}}{633\times10^{-9}}$$

$$= 25276$$

以上より、25276回強め合うことがわかった。

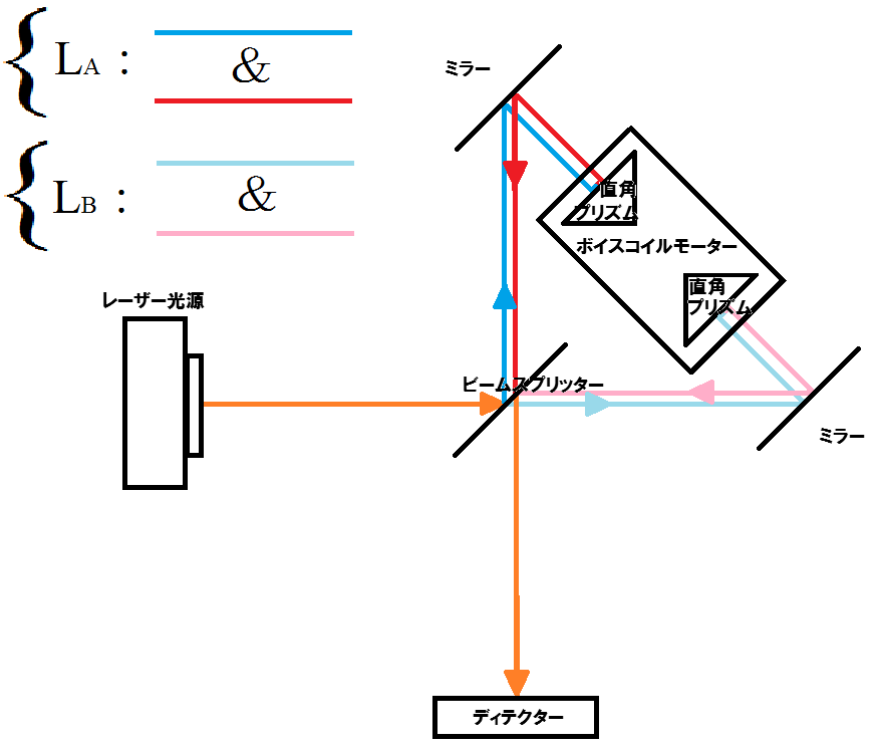

図5　　直角プリズムを移動させたる前のマイケルソン干渉計

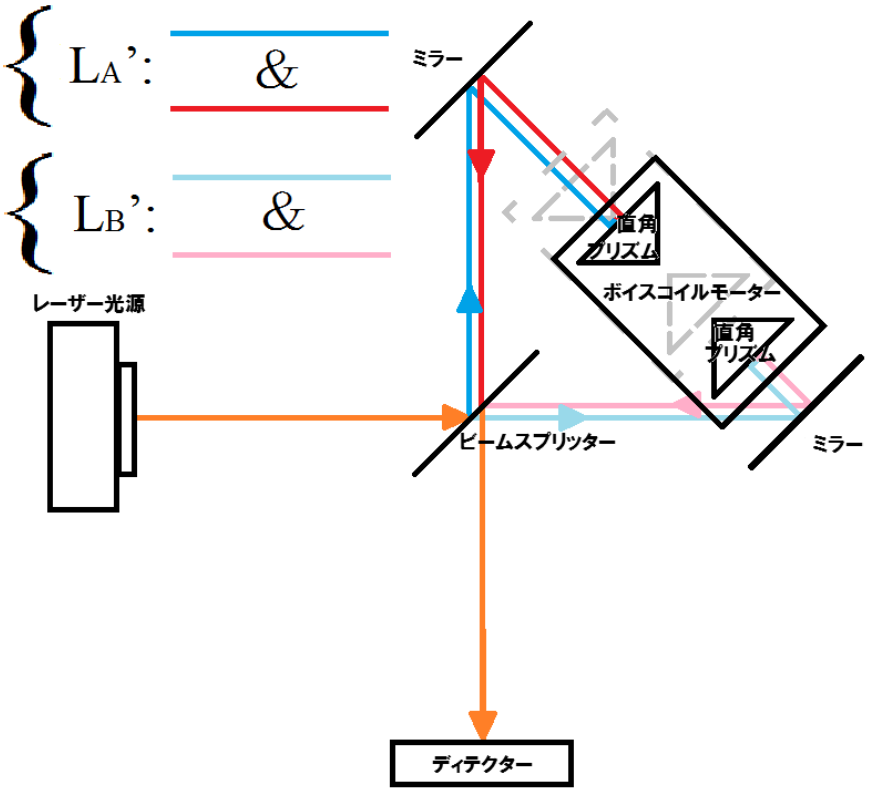

図6　　直角プリズムを移動させた後のマイケルソン干渉計

3　研究内容

3.1　小型マイケルソン干渉計の製作

マイケルソン干渉計は、一辺の長さが 25cm の正方形のアルミ板上に必要な全ての光学部品を配置し、干渉計を構成するビームスプリッターとそれぞれのミラーの距離を 15cm とし、干渉計を製作した。ビームスプリッターとミラー間の距離を短くすることにより、振動によりレーザー光がビームスプリッターやミラーに当たったときの入出射角変化が測定波長に及ぼす影響を小さくすることが可能となる。

図 5 のようにミラーとミラーの間にボイスコイルモータの上に置いた直角プリズムを設置し、ボイスコイルモータを光軸に対して前後に移動させることで光路差を周期的に変化させて干渉縞を生じさせることができる。

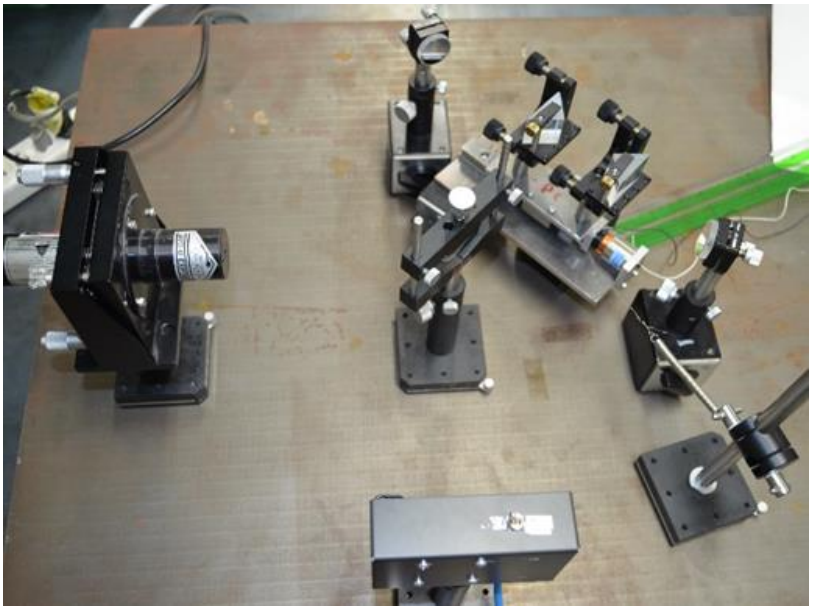

図 7　昨年度のマイケルソン干渉計

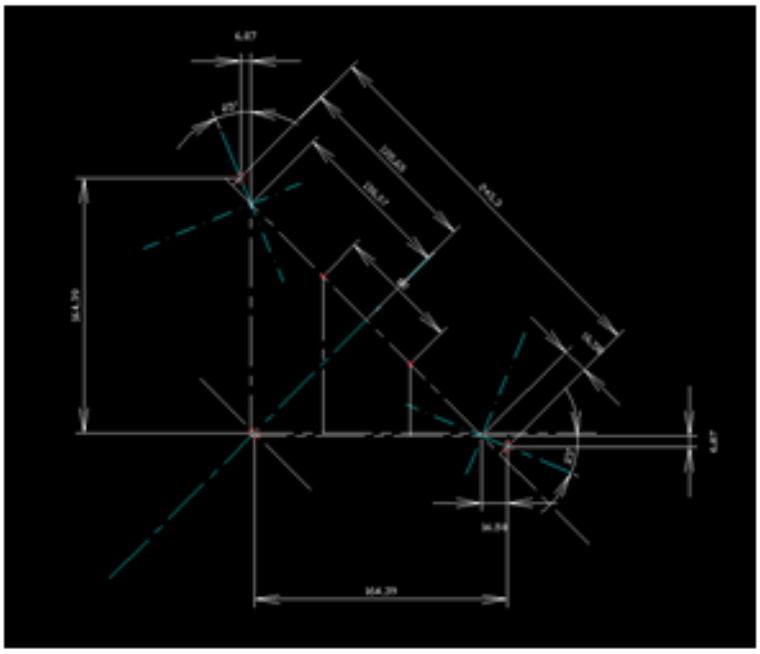

図 8　マイケルソン干渉計 設計図面

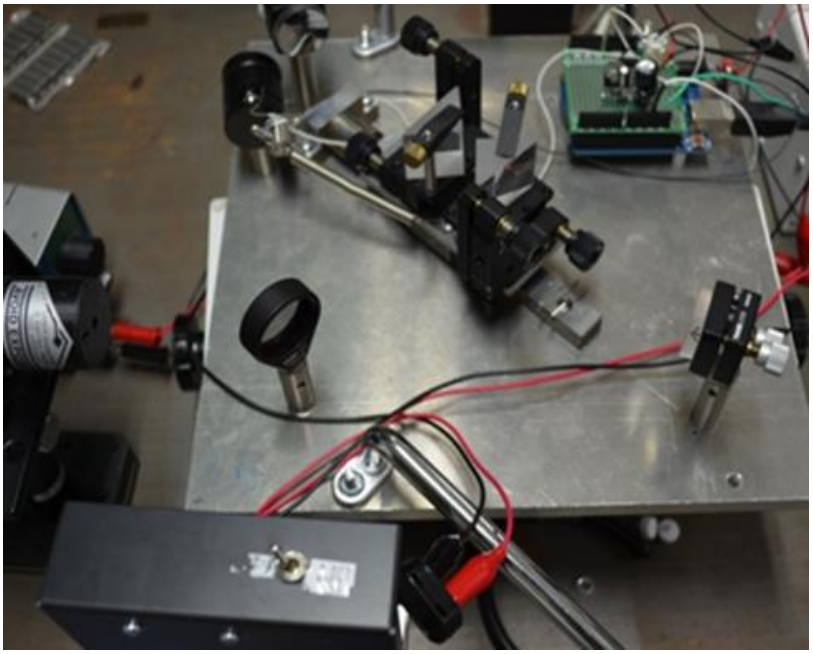

図 9　　小型化したマイケルソン干渉計

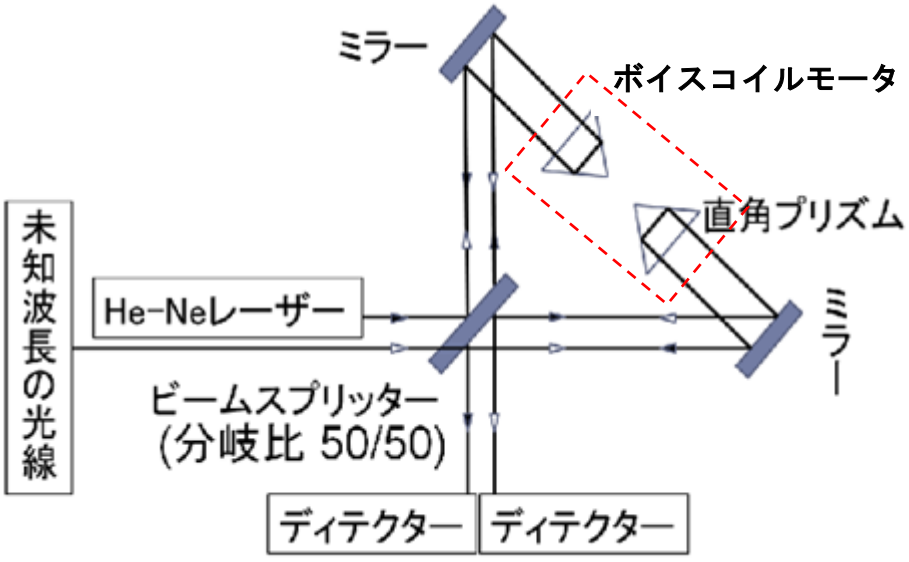

図 10　　マイケルソン干渉計(直角プリズム有)

3.2　直角プリズムの移動方法

図 11 に直角プリズムの移動装置を示す。直角プリズムを図 9 のように配置して光路差を変化させた。直角プリズム(Thorlabs、PS911)はリニアガイド(MiSUMi、SSEB16-230)に乗せ、ボイスコイルモータ(Akribis、AVM20-10)によって移動させた。また、このボイスコイルモータのストローク幅は±5 mm (total 10 mm)である。

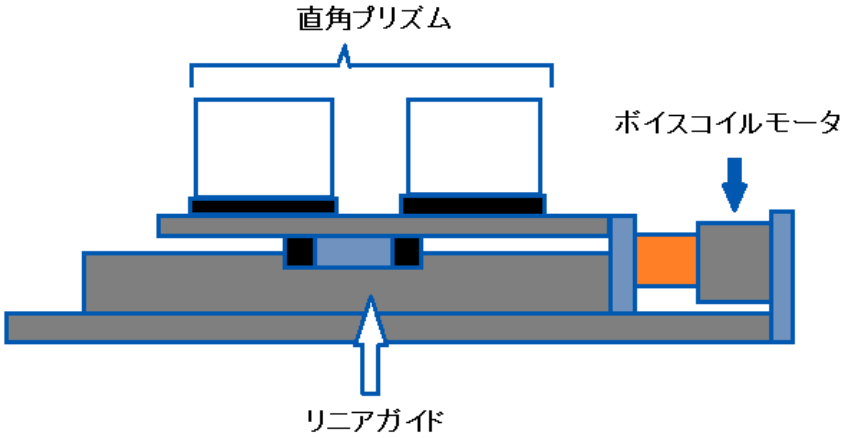

図 11　　直角プリズムの移動装置

ボイスコイルモータの仕組みとしては、ボイスコイルモータに電圧を印加すると永久磁石の磁界の中でボイスコイルに電流が流れ、フレミングの左手の法則によりボイスコイルが直線運動を行う。ボイスコイルモータは、マイコン (Arduino Uno)で作ったプログラムを用いて動かす。このプログラムは、30ms 電流を流し、120ms 電流を流さないという内容にした。電圧を印加していない時間は移動台にとりつけたバネで引っ張ることによってプリズムを光軸に対して前後に動かす。

3.3　小型化した干渉計の動作確認

3.3.1　He-Ne レーザー光を用いた動作確認

図 12 に He-Ne レーザー光を用いた実験装置を示す。He-Ne レーザー光のみを用いて、直角プリズムを動かして光路差を変化させ、ディテクター上に干渉縞を生み出し、出力波形を確認する。

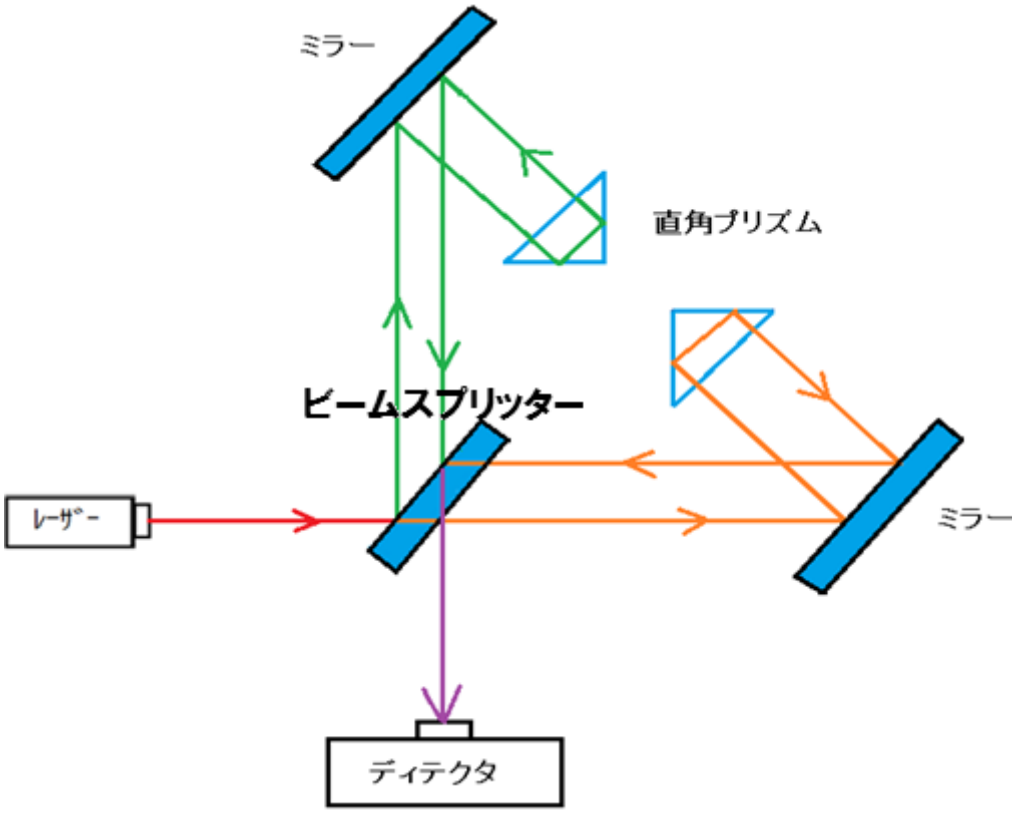

図 12 He-Ne レーザー光を用いた実験装置

3.3.2　He-Ne レーザー光と半導体レーザー光での実験

ボイスコイルモータで台を移動させ、光が干渉することが確認できたので、He-Ne レーザー光と半導体レーザーを使用し、半導体レーザー光の波長を、図 9 の実験装置を用いて出力波形の確認を行う。実験結果については 4.4 に詳細に記す。

3.4　干渉波形の整形方法

昨年度までは、干渉計で得られた出力波形をデータに変換し、プログラムを介してピーク数を求めていたが、本研究では干渉回数をカウントし液晶に表示させることが目的となった。干渉波形をそのままの形で読み取り、ピーク数を液晶に表示させるのはできないため、出力波形を矩形波に変換し、カウントプログラムで正確にピーク数を表示させるために、図 13 の矩形波変換回路を作成する。

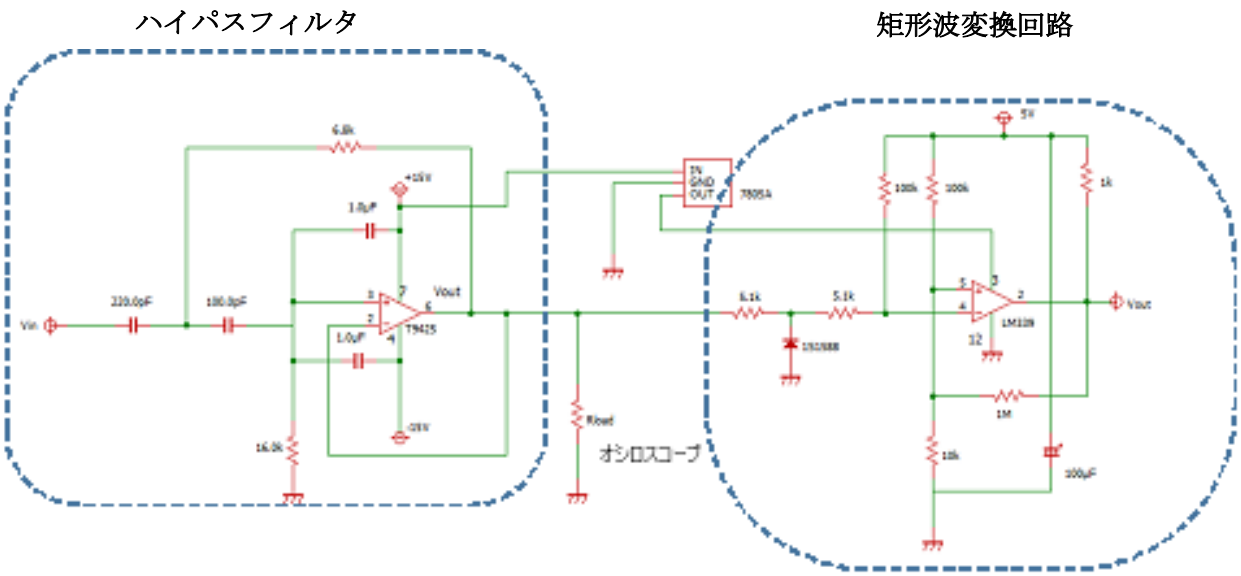

図 13　干渉波形の整形回路

この回路を用いて干渉波形をハイパスフィルタに入力し、その後、矩形波変換回路を通り矩形波に変換する。

3.5　波長測定方法

矩形波変換を用いて、He-Ne レーザー光と半導体レーザー光の出力波形を整形しカウントプログラムで光が強めあった回数(または弱めあった回数)をそれぞれカウントする。カウントした回数を n_1(He-Ne レーザー光)、n_2(半導体レーザー光)とする。

ここで、式(1)より、He-Ne レーザー光の波長 λ_1 は式(2)になる。

$$\lambda_1 = \frac{x}{n_1} \quad \cdots\cdots(2)$$

同様にして、半導体レーザー光の波長 λ_2 は式(3)になる。

$$\lambda_2 = \frac{x}{n_2} \quad \cdots\cdots(3)$$

ここで、移動距離 x[mm]は等しい。 x を消去し次式を得る。

$$\lambda_1 n_1 = \lambda_2 n_2 \quad \cdots\cdots(4)$$

よって、半導体レーザー光の波長 λ_2 は、式(5)より求めることができる。

$$\lambda_2 = \frac{\lambda_1 n_1}{n_2} \quad \cdots\cdots(5)$$

3.6　信号処理

図14にFPGAボードの写真を示す。

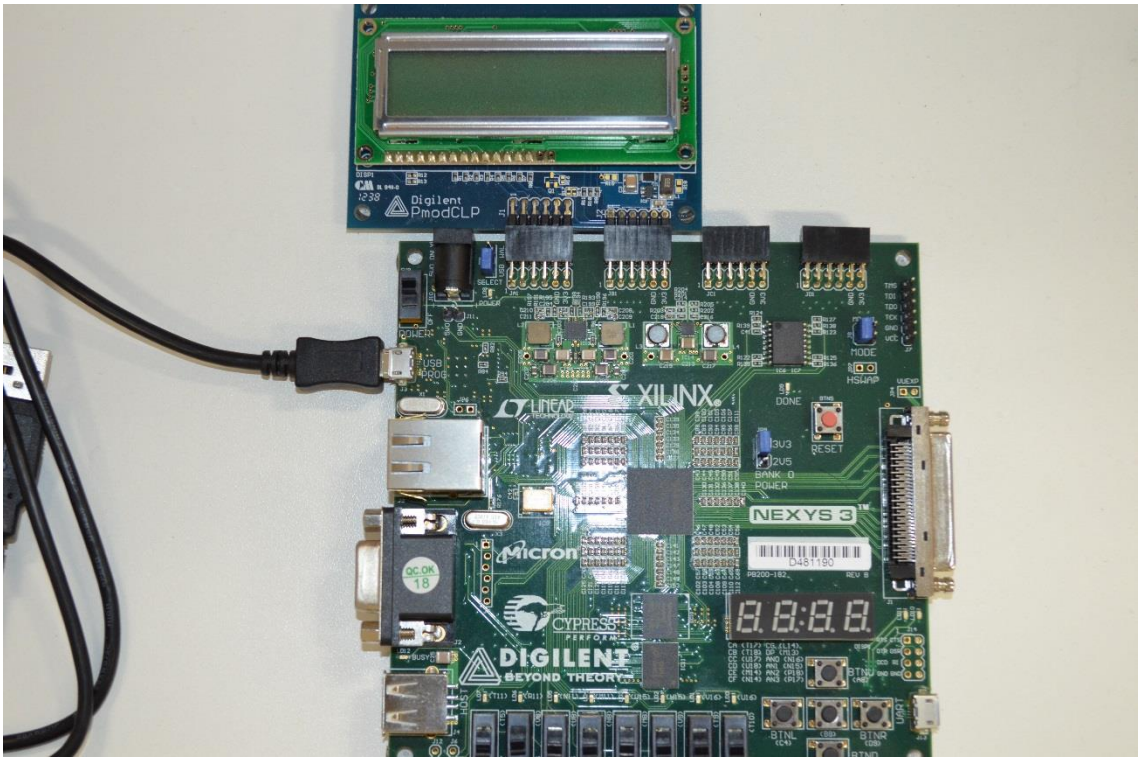

図 14　FPGA　Nexys3 ボード(Spartan-6) Xilinx 社製

3.6.1　FPGA Nexys3 ボード(Sparyan-6)とは

製造後ユーザーがハードウェア構成を設定できる集積回路で、並列処理が可能なため動作が高速である。また、実際に回路シミュレーションを行うことができるため、時間的な問題について検出可能である。外部のインターフェースとして、6ピン入出力ヘッダが4つある。

3.6.2　VHDL 言語

デジタル回路設計用のハードウェア記述言語のひとつである。「ada」という記述言語をもとに開発されたため複雑ではあるが、同時にテストパターンの記述ができるため、早期に回路の欠陥を見つけたり、修正したりすることが可能である。

3.6.3　FPGA と PC の接続方法

FPGA の回路は電源を投入するたびに外部からプログラムするか、PROM に直接書き込み、ダウンロードする必要がある。FPGA やコンフィギュレーション ROM への書き込みは通常、専用のプログラムケーブルが必要だが、この FPGA ボードは通常の USB ケーブル経由でプログラムすることが可能である。また、Nexys3 は比較的小さなボードであるため、別途電源を供給しなくても USB からの電源供給で動作させることができる。

3.6.4　カウント方法

ボイスコイルモータの動きに合わせて、矩形波に変換したレーザーの立ち上がりエッジ回数をカウントして、液晶に表示する。

その際、ボイスコイルモータの速度が速いので、液晶にカウント値を表示する時間の問題も考慮する。

4　研究結果

4.1　He-Ne レーザー光による波長測定結果

図 14 に He-Ne レーザー光による干渉波形を示す。電圧が時間によって周期的に変化していることがわかる。図より、この波形の周期は約 200kHz であり、振幅はおよそ 1V であることが読みとれる。

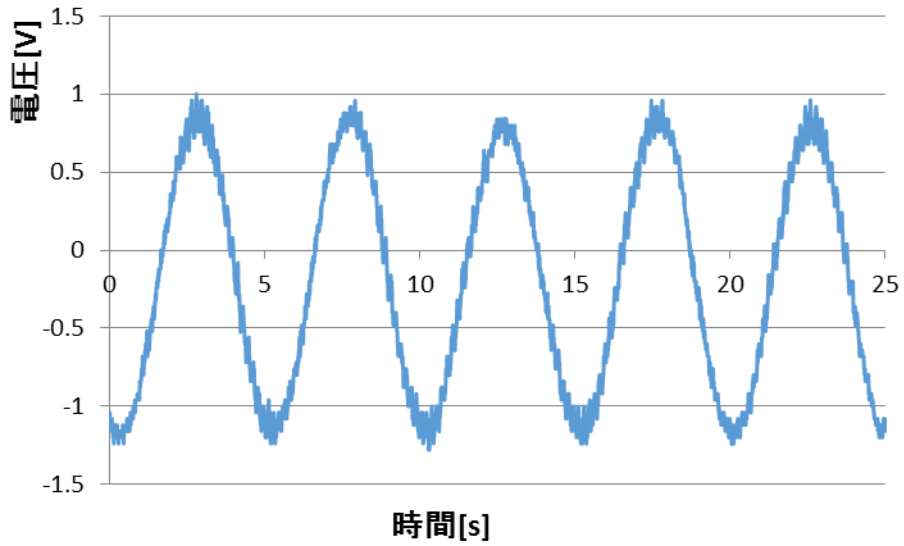

図 14　　He-Ne レーザー光による干渉波形

4.2　He-Ne レーザー光と半導体レーザー光での実験

マイケルソン干渉計を小型化させ、1 つのレーザー光の干渉を確認ができたので、図 8 の実験装置を用いて He-Ne レーザー光と半導体レーザー光の 2 つのレーザーを実際に入射してマイケルソン干渉計の動作確認を行った。

図 15 に He-Ne レーザー光と半導体レーザー光の各干渉波形を示す。

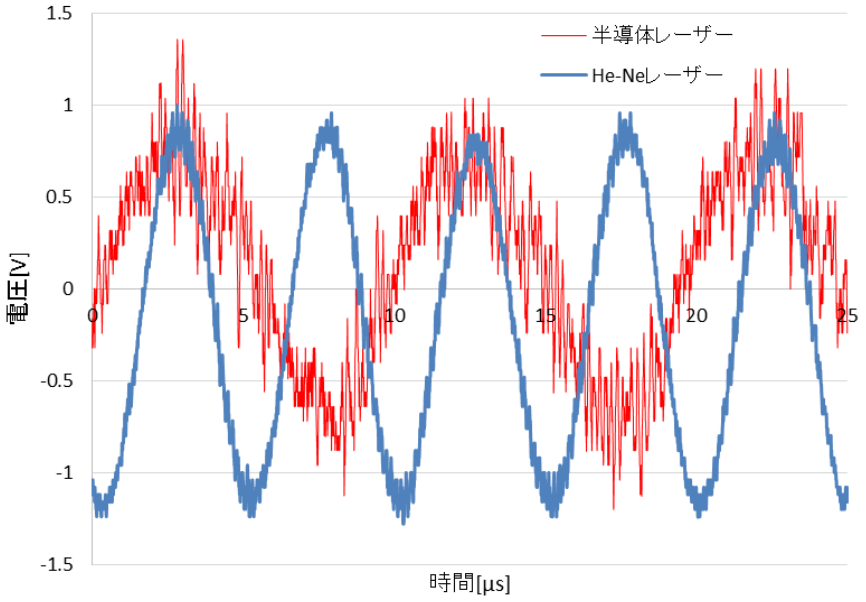

図 15　　He-Ne レーザー光と半導体レーザー光の各干渉波形

この結果をみると、電圧が時間によって周期的に変化していることから、2 つのレーザー光は干渉していると言える。

4.3　出力波形の整形

He-Ne レーザー光による干渉波形をハイパスフィルタおよび矩形波変換回路で整形した結果を図 18 に示す。図より、元の干渉波形の直流成分が除去されて、矩形波に変換されていることが分かる。

また、矩形波変換後の波形は曲線がかっており、予想していた方形波とは大きく異なった。

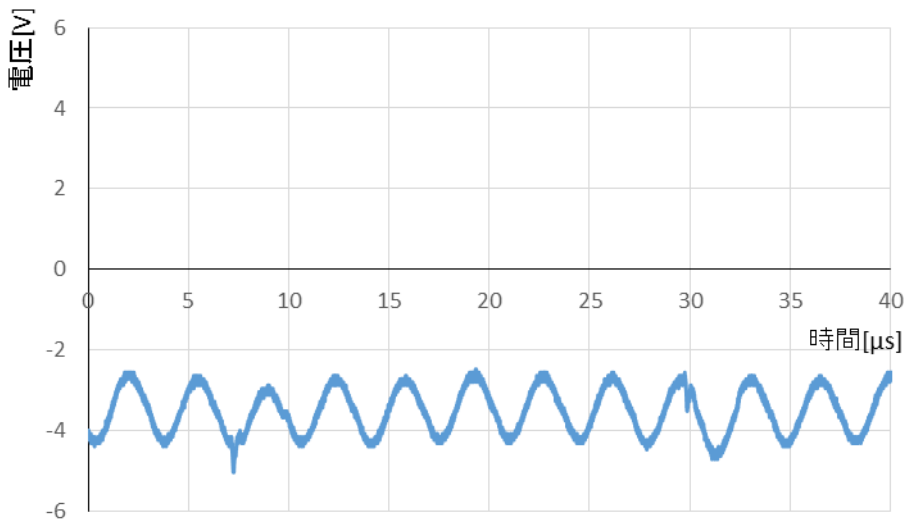

図 16　波形整形回路に入射前の干渉波形

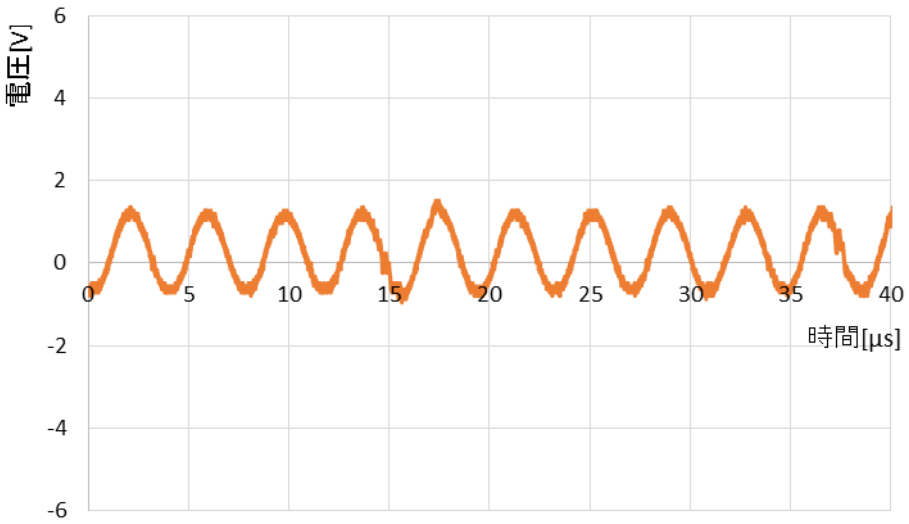

図 17　ハイパスフィルタ通過後の波形

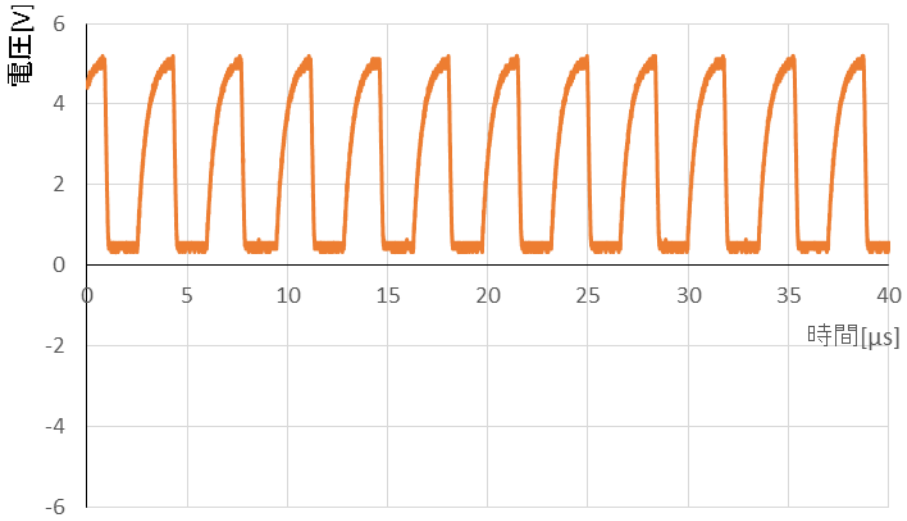

図 18　　矩形波変換回路通過後の波形

4.4　FPGA ボードに書き込んだ干渉回数のカウント表示プログラム

Xilinx 社製 ISE 64-bit Project Navigator を用いて干渉回数のカウントと表示を行うプログラムを作成した。付録 7.2 に作成したプログラムを示す。

まず、液晶表示のための初期設定と変数宣言を書き込む。

次に、外部入力にはボイスコイルモータからの信号と矩形波に変換した He-Ne レーザーと矩形波に変換した半導体レーザーの 3 つの信号が必要となるため、アップカウンターを３つ（FFFFF と GGGG と HHH）用意した。（FFFFF では、numa から numf を用いてアップカウント。GGGG では、numg から numl を用いてアップカウント。HHH では、numflag3 を用いて液晶表示のための時間経過設定を表す。）

その後、液晶に6ビットカウンタを2行表示するため、計12ビットの数値 (numa, numb……numl)を用意する。それぞれの数値は「0」から「9」までカウントアップし、「9」まできたら「0」に戻り、左隣のアップカウンターへ「1」繰り上がるように設定した。また、6ビット全て「9」になったときと、リセットボタンを押したときは、カウンターの値が全て「0」になるように定めた。

液晶にカウンターを表示させる際、一定時間カウント値を表示しなければカウントした回数を読むことができないため、保持変数(numflag1, numflag2, numflag3)を用意した。

4.5　カウンターの速度制御

ボイスコイルモータの速度が速いため、Arduino 信号の立ち上がりエッジが来るたびに矩形波に変換したレーザーのカウントと表示を行うと、一瞬しか表示されない。

そのため、最初の立ち上がりエッジがきてから Arduino 信号の ON 時間分、矩形波に変換したレーザーをカウントし、そこから、Arduino 信号の 5 個目の立ち上がりエッジが来るまで矩形波に変換したレーザーのカウント値を液晶に表示するように設計した。

図 19 に矩形波変換した Arduino の信号、図 20 に矩形波変換したレーザーの信号を示し、図 21 にカウンターのフローチャートを示す。

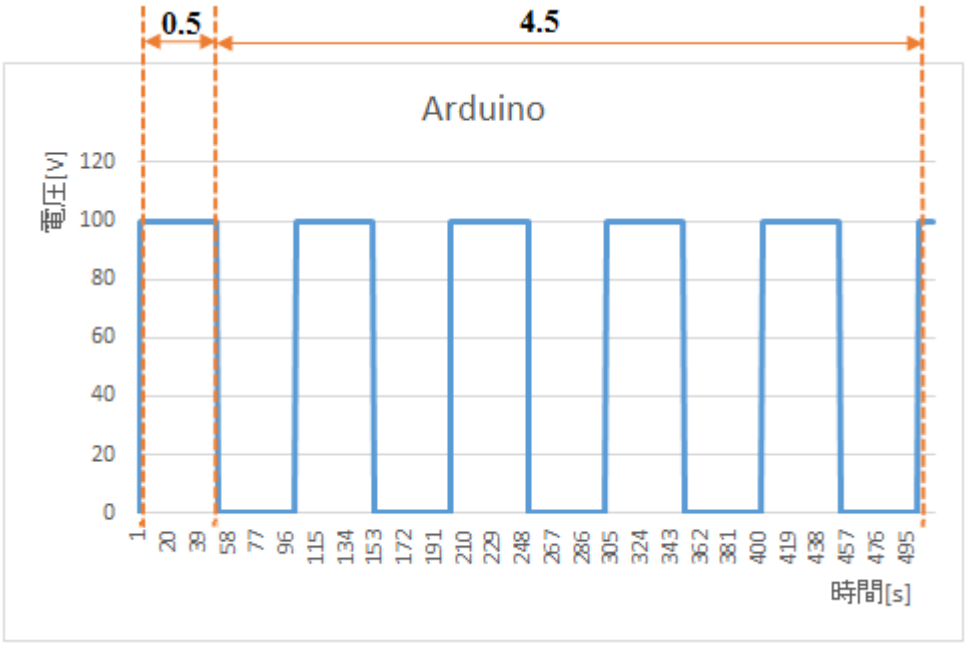

図 19　　Arduino からの信号

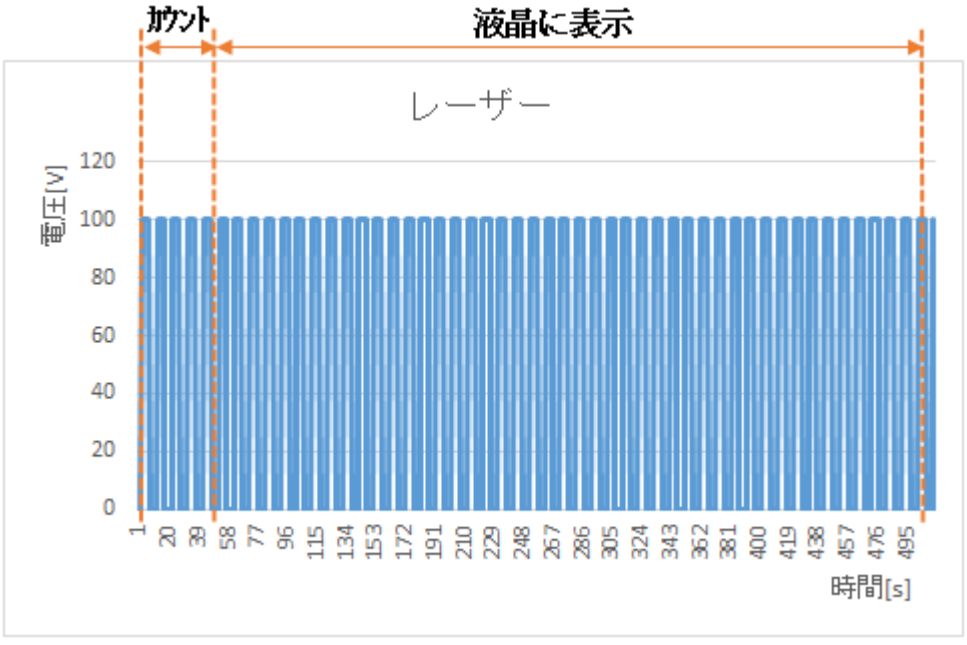

図 20　　レーザーからの信号

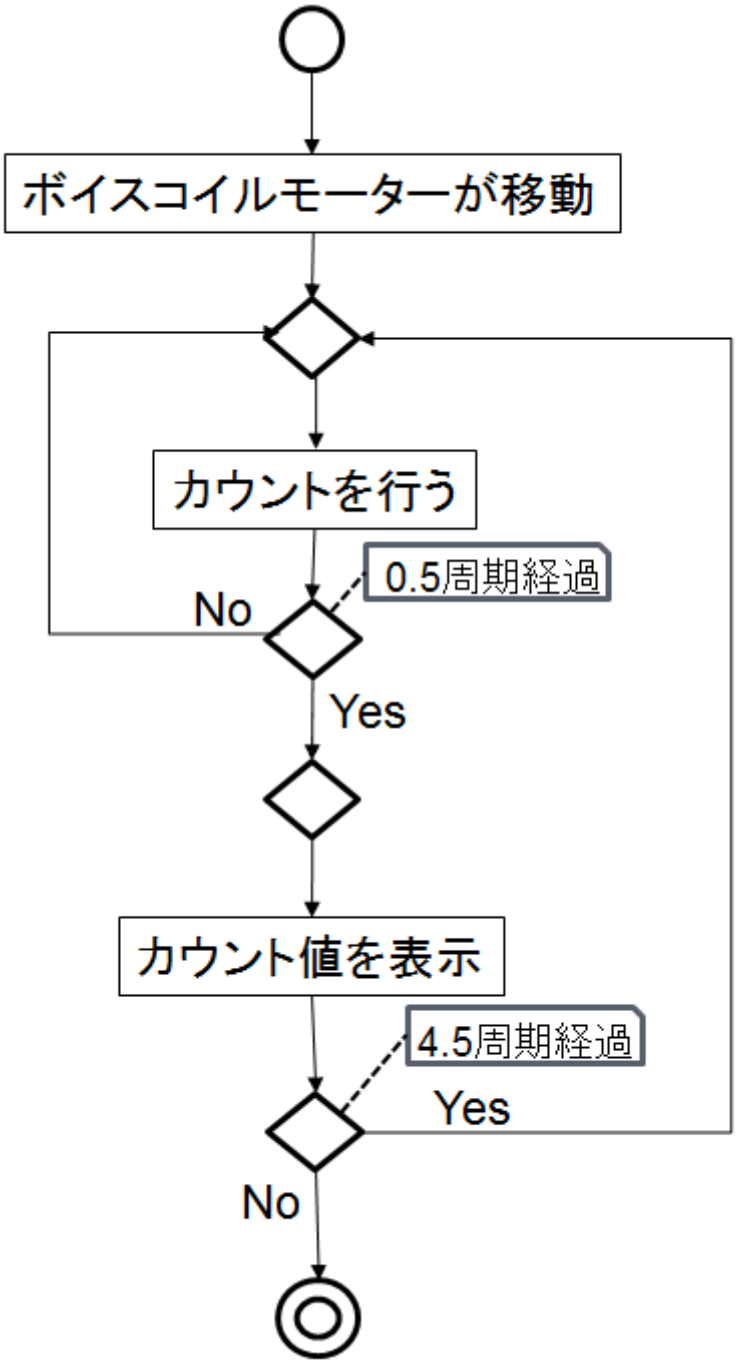

図21　　カウンターのフローチャート

4.6　波長測定結果

図22に測定機器の全体図を示す。

また、表１にピン配置表を示す。

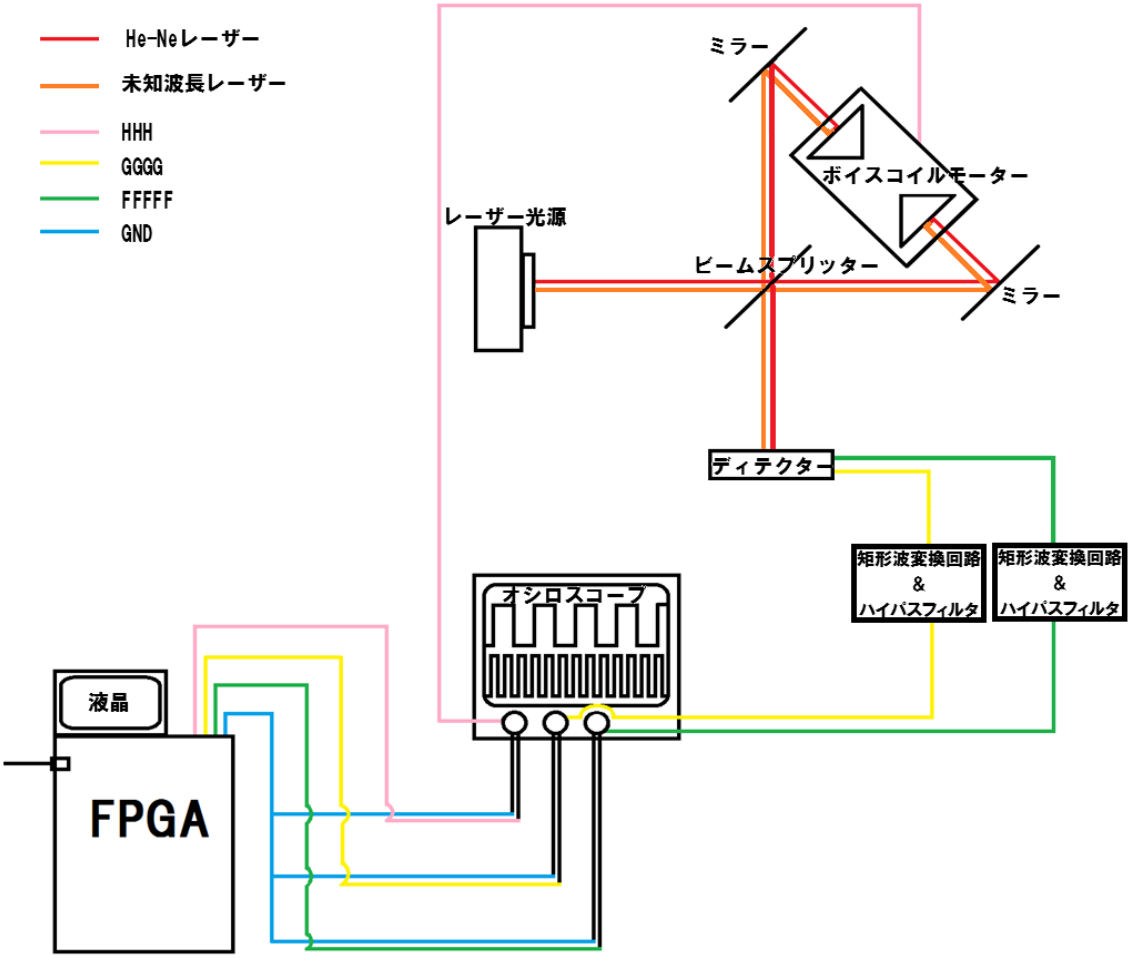

図 22　　測定機器

表 1　　ピン配置表

プログラム	基板	ピン番号
HHH	JD2	F10
GGGG	JD3	F11
FFFFF	JD4	E11

図23に直角プリズムを移動させないときの光路図を示し、図24に直角プリズムを移動させたときの光路図示す。表１のピン配置表に基づき、FPGAボードのピン配置を行い、ディテクターが受光した波長の強め合った回数をFPGAボードでカウントし、液晶に表示させた。作製したマイケルソン干渉計を用いてレーザー光の波長を測定する。ボイスコイルモータを動かすことにより、ボイスコイルモータ上の直角プリズムからミラーまでの距離が変わり、光は明暗を繰り返す。干渉した光をディテクターに通し、その光をFPGAボードに入れ波長をカウントする。波長は半周期で明→暗になり、さらに半周期で暗→明となるため、１周期は明→暗→明である。強めあった回数をｎ、光路長の変化をｘとするとき、波長λは(１)式で求めることができる。

また、既知であるHe-Neレーザーの波長は632.88nmとし、半導体レーザーの公称値は、636.57nmである。

また、測定したときのレーザー以外の外部の光量（蛍光灯）および室内の温度はほぼ一定であり、ボイスコイルモータ以外のものが発生させた振動はなかったため、外部からの影響は無視できるものと思われる。

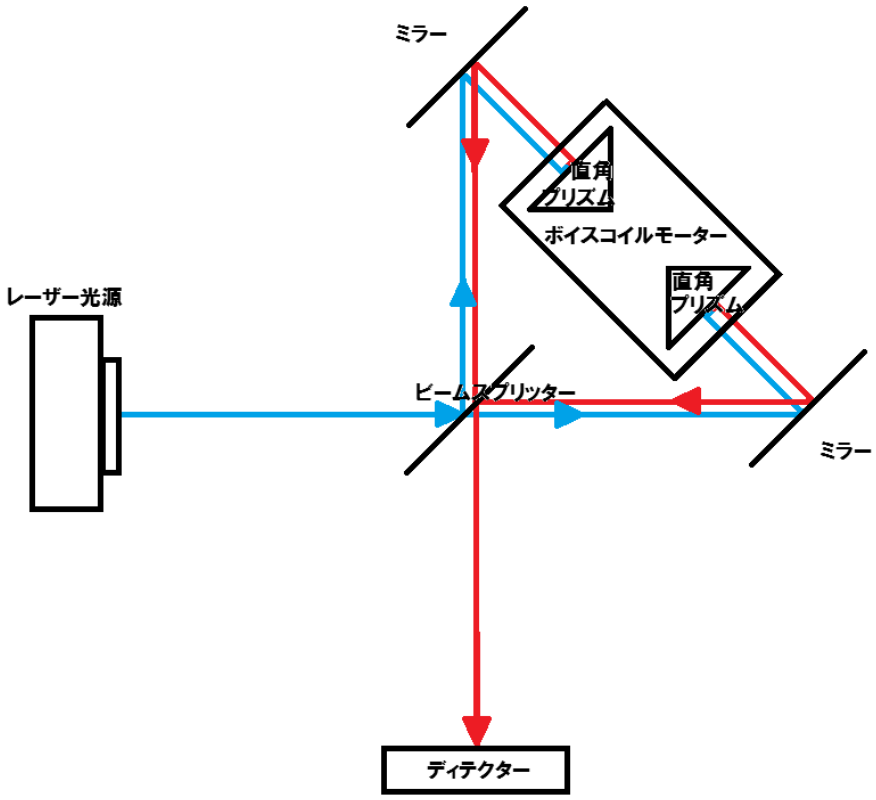

図 23　　直角プリズムを移動させないときの光路図

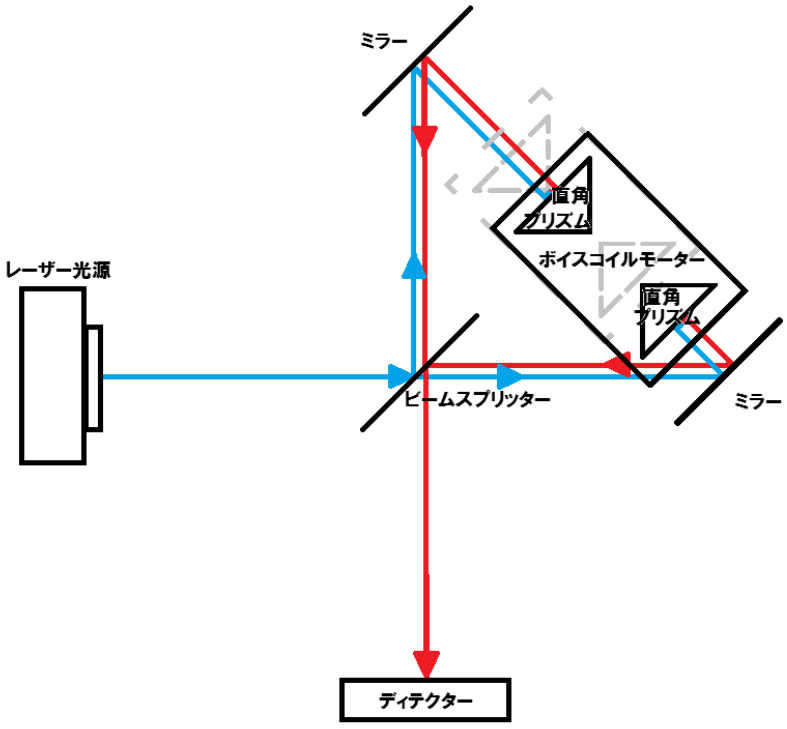

図 24　　直角プリズムを移動させたときの光路図

図25に波長測定結果1を、図26に波長測定結果2を、図27に波長測定結果3を示す。１回目の計測では、上記の方法でE11ピンにHe-Neレーザーの干渉波を繋ぎ、F11ピンに半導体レーザーの干渉波を繋ぎ、25回計測を行った。結果は分光器で測定したλ$_2$の目標値と計測値の間に差が見て取れた。

原因として、矩形波変換回路やハイパスフィルタの回路もしくは、FPGAボードに組み込んだプログラムが考えられ、波長測定結果2では、E11ピンに半導体レーザーの干渉波を繋ぎ、F11ピンにHe-Neレーザーを繋ぎ、25回計測を行った。

しかし、測定結果に目立った変化は見られず、回路やプログラムに異常は無いことが分かった。

３回目の計測では、レーザーの向きを調整し、光路差を変えて25回計測を行った。

すると、図27をみて分かるように、１回目、２回目の計測よりも636.57[nm]に近づいき、ばらつきも少なくなった。

このことより、レーザーの入射角による光路差の変化が計測値に影響を及ぼし、レーザーの入射角による光路差次第で、計測値は、目標値に近い値となることが分かった。

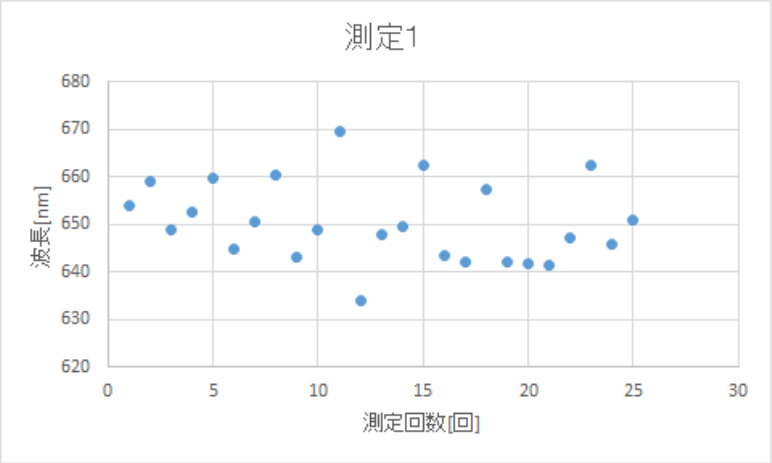

図 25　　波長測定結果 1

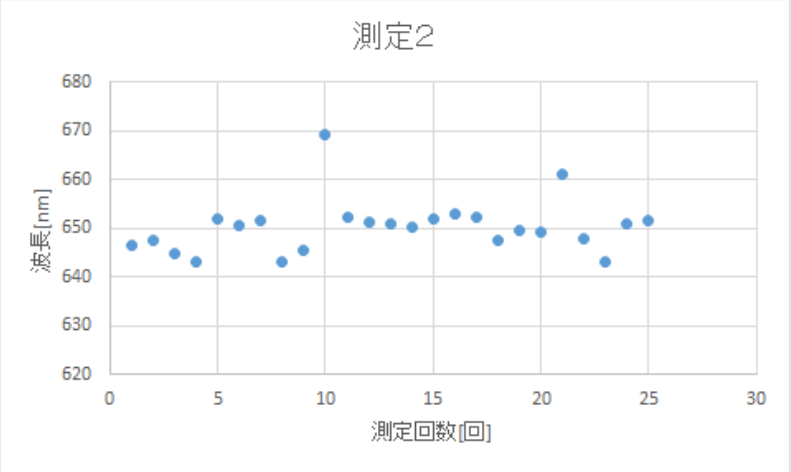

図 26　　波長測定結果 2

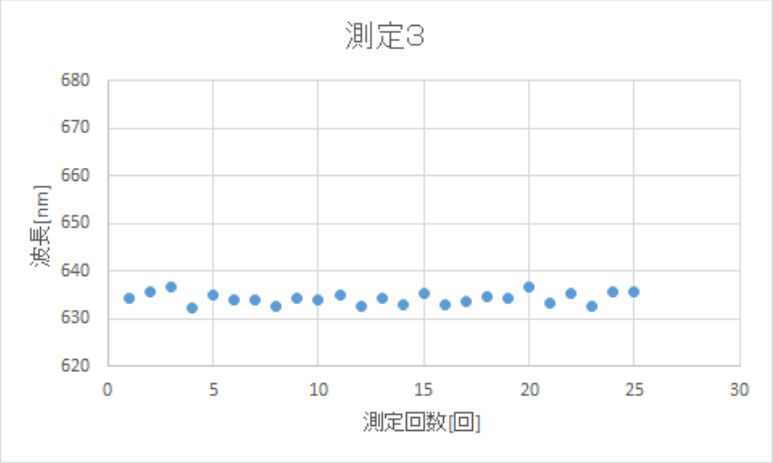

図 27　　波長測定結果 3

5　まとめ

マイケルソン干渉計の小型化と、信号処理系を含んだシステムを製作した。実際に実験を行うと、実験により求めた波長の値と、分光器で測定した値との間には約 11.77[nm] の差があった。この差については、レーザーを干渉計に入射する時に、各々の光路差が変わってしまうことで波長に差が出たと考えられる。

6　参考文献

(１)キットで学ぶ！シリーズNo.06 FPGAチャレンジャーXILINX Spartan3E 版入門編, キットで学ぶ教材委員会, 株式会社アドウィン, 2013

(２)実用HDLサンプル記述集, 鳥海佳考・田原迫仁治・横溝憲治, CQ出版株式会社, 2002

(３)VHDLとCPLDによるロジック設計入門, 中幸政, CQ出版株式会社, 2005

(４)マイケルソン干渉計を用いた波長計の製作, 堺洋貴・村田義光, 2012

7 付録

7.1 以下にボイスコイルモータを動作させる場合にマイコンボードに入力したプログラムを示す。

```
int ledPin1 = 13;     // LED とデジタルピン 13 を接続する
void setup()
{
pinMode(ledPin1, OUTPUT);   //出力のデジタル端子を設定
}
void loop() // 繰り返し
{
digitalWrite(ledPin1, HIGH);   // LED を on にする

delay(30); // 30ms 待つ

digitalWrite(ledPin1, LOW);     //LED を off にする

delay(120); // 120ms 待つ
```

7.2 FPGA ボードに書き込んだプログラム(VHDL)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lcd is
        port(
        HHH, GGGG, FFFFF, clk , reset : in std_logic;
        SF_D : out std_logic_vector(3 downto 0);
        LCD_E, LCD_RS, LCD_RW : out std_logic;
        LED : out std_logic_vector(7 downto 0)
        );
end lcd;

architecture behavior of lcd is

type display_state is (init, function_set, s1, entry_set, s2, set_display, s3,
clr_display, s4, pause,set_adda, s5, updata, s6,
 updatb, s8, updatc, s10, updatd, s12, update, s14, updatf, s16,
 set_addg, s17, updatg, s18, updath, s20, updati, s22, updatj, s24,  updatk,
s26, updatl, s28, done);
signal cur_state : display_state := init;

signal SF_D0, SF_D1 : std_logic_vector(3 downto 0);
signal LCD_E0, LCD_E1 : std_logic;
signal mux : std_logic;

type tx_sequence is (high_setup, high_hold, oneus, low_setup, low_hold, fortyus,
done);
signal tx_state : tx_sequence := done;
signal tx_byte : std_logic_vector(7 downto 0);
signal tx_init : std_logic := '0';
signal tx_rdy : std_logic := '0';

type init_sequence is (idle, fifteenms, s1, s2, s3, s4, s5, s6, s7, s8, done);
```

```
signal init_state : init_sequence := idle;
signal init_init, init_done : std_logic := '0';

signal i : integer range 0 to 1500000 := 0;
signal i2 : integer range 0 to 4000 := 0;
signal i3 : integer range 0 to 164000 := 0;
signal i4 : integer range 0 to 100000000 := 0;

signal numa : std_logic_vector(3 downto 0);
signal numb : std_logic_vector(3 downto 0);
signal numc : std_logic_vector(3 downto 0);
signal numd : std_logic_vector(3 downto 0);
signal nume : std_logic_vector(3 downto 0);
signal numf : std_logic_vector(3 downto 0);

signal numg : std_logic_vector(3 downto 0);
signal numh : std_logic_vector(3 downto 0);
signal numi : std_logic_vector(3 downto 0);
signal numj : std_logic_vector(3 downto 0);
signal numk : std_logic_vector(3 downto 0);
signal numl : std_logic_vector(3 downto 0);

signal numflag1 : std_logic_vector(3 downto 0);
signal numflag2 : std_logic_vector(3 downto 0);
signal numflag3 : std_logic_vector(7 downto 0);


signal numaa : std_logic_vector(3 downto 0);
signal numbb : std_logic_vector(3 downto 0);
signal numcc : std_logic_vector(3 downto 0);
signal numdd : std_logic_vector(3 downto 0);
signal numee : std_logic_vector(3 downto 0);
signal numff : std_logic_vector(3 downto 0);

signal numgg : std_logic_vector(3 downto 0);
signal numhh : std_logic_vector(3 downto 0);
signal numii : std_logic_vector(3 downto 0);
signal numjj : std_logic_vector(3 downto 0);
signal numkk : std_logic_vector(3 downto 0);
signal numll : std_logic_vector(3 downto 0);

begin
LED <= numflag3; --for diagnostic purposes
        LCD_RW <= '0'; --write only

        --when to transmit a command/data and when not to
        with cur_state select
                tx_init <= '1' when function_set | entry_set | set_display |
clr_display | set_adda |
                updata | updatb | updatc | updatd | update| updatf|set_addg |
updatg| updath| updati| updatj| updatk| updatl,'0' when others;

        --control the bus
        with cur_state select
```

```
                mux <= '1' when init,
                        '0' when others;

        --control the initialization sequence
        with cur_state select
                init_init <= '1' when init,
                        '0' when others;

        --register select
        with cur_state select
                LCD_RS <= '0' when s1|s2|s3|s4|s5|s17,
                        '1' when others;

        with cur_state select
                tx_byte <= "00101100" when s1,
                                        "00000110" when s2,
                                        "00001100" when s3,
                                        "00000001" when s4,
                                        "10000000" when s5,
                                        "0011"&numa when s6,
                                        "0011"&numb when s8,
                                        "0011"&numc when s10,
                                        "0011"&numd when s12,
                                        "0011"&nume when s14,
                                        "0011"&numf when s16,
                                        "11000000" when s17,
                                        "0011"&numg when s18,
                                        "0011"&numh when s20,
                                        "0011"&numi when s22,
                                        "0011"&numj when s24,
                                        "0011"&numk when s26,
                                        "0011"&numl when s28,
                                        "00000000" when others;

counter0: process(HHH, reset)
        begin
                if(reset='1')then
                        numflag3 <= (others => '0');
                elsif(HHH'event and HHH = '1')then
                        if(numflag3 = 5)then
                                numflag3 <= (others =>'0');
                        else
                                numflag3 <= numflag3 + '1';
                        end if;
                end if;
        end process counter0;

counter1: process(HHH, GGGG,reset)
        begin
                if(reset = '1') then
                        i4 <= 0;
                        numg <= "0000";
                        numh <= "0000";
                        numi <= "0000";
```

```
                numj <= "0000";
                numk <= "0000";
                numl <= "0000";

        elsif(HHH='0') then
                numg <= numgg;
                numh <= numhh;
                numi <= numii;
                numj <= numjj;
                numk <= numkk;
                numl <= numll;
                numflag1 <= "0001";

        elsif(GGGG'event and GGGG='1' ) then
                if(numflag3="0001")then
                        if(numflag1 = "0001")then
                                        numg <= "0000";
                                        numh <= "0000";
                                        numi <= "0000";
                                        numj <= "0000";
                                        numk <= "0000";
                                        numl <= "0000";
                                        numflag1 <= "0000";

                        end if;

                        if(numg = "1001" and numh = "1001" and numi
= "1001" and numj = "1001" and numk = "1001" and numl = "1001")then
                                numg <= "0000";
                                numh <= "0000";
                                numi <= "0000";
                                numj <= "0000";
                                numk <= "0000";
                                numl <= "0000";
                        elsif(numh = "1001" and numi = "1001" and
numj = "1001" and numk = "1001" and numl = "1001")then
                                numg <= numg + '1';
                                numh <= "0000";
                                numi <= "0000";
                                numj <= "0000";
                                numk <= "0000";
                                numl <= "0000";
                        elsif(numi = "1001" and numj = "1001" and
numk = "1001" and numl = "1001")then
                                numh <= numh + '1';
                                numi <= "0000";

                                numj <= "0000";
                                numk <= "0000";
                                numl <= "0000";
                        elsif(numj = "1001" and numk = "1001" and
numl = "1001")then
                                numi <= numi + '1';
                                numj <= "0000";
```

```
                                        numk <= "0000";
                                        numl <= "0000";
                                elsif(numk = "1001" and numl = "1001")then
                                        numj <= numj + '1';
                                        numk <= "0000";
                                        numl <= "0000";
                                elsif(numl = "1001")then
                                        numk <= numk + '1';
                                        numl <= "0000";
                                else
                                        numl <= numl + '1';
                                end if;
                        numgg <= numg;
                        numhh <= numh;
                        numii <= numi;
                        numjj <= numj;
                        numkk <= numk;
                        numll <= numl;

                        end if;
                end if;
        end process counter1;

counter2: process(HHH,FFFFF,reset)
        begin
                if(reset = '1') then
                        i4 <= 0;
                        numa <= "0000";
                        numb <= "0000";
                        numc <= "0000";
                        numd <= "0000";
                        nume <= "0000";
                        numf <= "0000";

                elsif(HHH='0') then
                        numa <= numaa;
                        numb <= numbb;
                        numc <= numcc;
                        numd <= numdd;
                        nume <= numee;
                        numf <= numff;
                        numflag2 <= "0001";

                elsif(FFFFF'event and FFFFF='1') then
                        if(numflag3="0001")then
                                if(numflag2 = "0001")then
                                        numa <= "0000";
                                        numb <= "0000";
                                        numc <= "0000";
                                        numd <= "0000";
                                        nume <= "0000";
                                        numf <= "0000";

                                        numflag2 <= "0000";
```

```
                                    end if;
                                    if(numa = "1001" and numb = "1001" and numc
= "1001" and numd = "1001" and nume = "1001" and numf = "1001")then
                                            numa <= "0000";
                                            numb <= "0000";
                                            numc <= "0000";
                                            numd <= "0000";
                                            nume <= "0000";
                                            numf <= "0000";
                                    elsif(numb = "1001" and numc = "1001" and
numd = "1001" and nume = "1001" and numf = "1001")then
                                            numa <= numa + '1';
                                            numb <= "0000";
                                            numc <= "0000";
                                            numd <= "0000";
                                            nume <= "0000";
                                            numf <= "0000";
                                    elsif(numc = "1001" and numd = "1001" and
nume = "1001" and numf = "1001")then
                                            numb <= numb + '1';
                                            numc <= "0000";

                                            numd <= "0000";
                                            nume <= "0000";
                                            numf <= "0000";
                                    elsif(numd = "1001" and nume = "1001" and
numf = "1001")then
                                            numc <= numc + '1';
                                            numd <= "0000";
                                            nume <= "0000";
                                            numf <= "0000";
                                    elsif(nume = "1001" and numf = "1001")then
                                            numd <= numd + '1';
                                            nume <= "0000";
                                            numf <= "0000";
                                    elsif(numf = "1001")then
                                            nume <= nume + '1';
                                            numf <= "0000";
                                    else
                                            numf <= numf + '1';
                                    end if;
                            numaa <= numa;
                            numbb <= numb;
                            numcc <= numc;
                            numdd <= numd;
                            numee <= nume;
                            numff <= numf;
                            end if;
                    end if;
            end process counter2;


--main state machine
display: process(clk, reset)
```

```
begin
        if(reset='1') then
                cur_state <= init;
        elsif(clk='1' and clk'event) then
                case cur_state is
                        when init =>
                                if(init_done = '1') then
                                        cur_state <= function_set;
                                else
                                        cur_state <= init;
                                end if;
                        when function_set =>
                                cur_state <= s1;
                        when s1 =>
                                if(tx_rdy = '1') then
                                        cur_state <= entry_set;
                                else
                                        cur_state <= s1;
                                end if;
                        when entry_set =>
                                cur_state <= s2;
                        when s2 =>
                                if(tx_rdy = '1') then
                                        cur_state <= set_display;
                                else
                                        cur_state <= s2;
                        end if;
                        when set_display =>
                                cur_state <= s3;
                        when s3 =>
                                if(tx_rdy = '1') then
                                        cur_state <= clr_display;
                                else
                                        cur_state <= s3;
                                end if;
                        when clr_display =>
                                cur_state <= s4;
                        when s4 =>
                                i3 <= 0;
                                if(tx_rdy = '1') then
                                        cur_state <= pause;
                                else
                                        cur_state <= s4;
                                end if;
                        when pause =>
                                if(i3 = 164000) then
                                        cur_state <= set_adda;
                                        i3 <= 0;
                                else
                                        cur_state <= pause;
                                        i3 <= i3 + 1;
                                end if;

                        when set_adda =>
```

```
        cur_state <= s5;
when s5 =>
        if(tx_rdy = '1') then
                cur_state <= updata;
        else
                cur_state <= s5;
        end if;

when updata =>
        cur_state <= s6;

when s6 =>
        if(tx_rdy = '1') then
                cur_state <= updatb;
        else
                cur_state <= s6;
        end if;

when updatb =>
        cur_state <= s8;

when s8 =>
        if(tx_rdy = '1') then
                cur_state <= updatc;
        else
                cur_state <= s8;
        end if;

when updatc =>
        cur_state <= s10;

when s10 =>
        if(tx_rdy = '1') then
                cur_state <= updatd;
        else
                cur_state <= s10;
        end if;

when updatd =>
        cur_state <= s12;

when s12 =>
        if(tx_rdy = '1') then
                cur_state <= update;
        else
                cur_state <= s12;
        end if;

when update =>
        cur_state <= s14;

when s14 =>
        if(tx_rdy = '1') then
                cur_state <= updatf;
```

```
        else
                cur_state <= s14;
        end if;

when updatf =>
        cur_state <= s16;

when s16 =>
        if(tx_rdy = '1') then
                cur_state <= set_addg;
        else
                cur_state <= s16;
        end if;

when set_addg =>
        cur_state <= s17;
when s17 =>
        if(tx_rdy = '1') then
                cur_state <= updatg;
        else
                cur_state <= s17;
        end if;

when updatg =>
        cur_state <= s18;

when s18 =>
        if(tx_rdy = '1') then
                cur_state <= updath;
        else
                cur_state <= s18;
        end if;

when updath =>
        cur_state <= s20;

when s20 =>
        if(tx_rdy = '1') then
                cur_state <= updati;
        else
                cur_state <= s20;
        end if;

when updati =>
        cur_state <= s22;

when s22 =>
        if(tx_rdy = '1') then
                cur_state <= updatj;
        else
                cur_state <= s22;
        end if;
when updatj =>
        cur_state <= s24;
```

```
                                when s24 =>
                                        if(tx_rdy = '1') then
                                                cur_state <= updatk;
                                        else
                                                cur_state <= s24;
                                        end if;
                                when updatk =>
                                        cur_state <= s26;

                                when s26 =>
                                        if(tx_rdy = '1') then
                                                cur_state <= updatl;
                                        else
                                                cur_state <= s26;
                                        end if;
                                when updatl =>
                                        cur_state <= s28;

                                when s28 =>
                                        if(tx_rdy = '1') then
                                                cur_state <= set_adda;
                                        else
                                                cur_state <= s28;
                                        end if;


                                when done =>
                                        cur_state <= done;
                        end case;
                end if;
        end process display;

        with mux select
                SF_D <= SF_D0 when '0', --transmit
                SF_D1 when others; --initialize
        with mux select
                LCD_E <= LCD_E0 when '0', --transmit
                LCD_E1 when others; --initialize
        with tx_state select
                tx_rdy <= '1' when done,
                        '0' when others;with tx_state select
                LCD_E0 <= '0' when high_setup | oneus | low_setup | fortyus |
done,
                        '1' when high_hold | low_hold;
        with tx_state select
                SF_D0 <= tx_byte(7 downto 4) when high_setup | high_hold |
oneus,
                tx_byte(3 downto 0) when low_setup | low_hold | fortyus |
done;

--specified by datasheet
transmit : process(clk, reset, tx_init)
        begin
```

```
                        if(reset='1') then
                                tx_state <= done;
                        elsif(clk='1' and clk'event) then
                                case tx_state is
                                        when high_setup => --40ns
                                                if(i2 = 4) then
                                                        tx_state <= high_hold;
                                                        i2 <= 0;
                                                else
                                                        tx_state <= high_setup;
                                                        i2 <= i2 + 1;
                                                end if;
                                        when high_hold => --230ns
                                                if(i2 = 24) then
                                                        tx_state <= oneus;
                                                        i2 <= 0;
                                                else
                                                        tx_state <= high_hold;
                                                        i2 <= i2 + 1;
                                                end if;
                                        when oneus =>
                                                if(i2 = 100) then
                                                        tx_state <= low_setup;
                                                        i2 <= 0;
                                                else
                                                        tx_state <= oneus;
                                                        i2 <= i2 + 1;
                                                end if;
                                        when low_setup =>
                                                if(i2 = 4) then
                                                        tx_state <= low_hold;
                                                        i2 <= 0;
                                                else
                                                        tx_state <= low_setup;
                                                        i2 <= i2 + 1;
                                                end if;
                                        when low_hold =>
                                                if(i2 = 24) then
                                                        tx_state <= fortyus;
                                                        i2 <= 0;
                                                else
                                                        tx_state <= low_hold;
                                                        i2 <= i2 + 1;
                                                end if;
                                        when fortyus =>
                                                if(i2 = 4000) then
                                                                tx_state <= done;
                                                                i2 <= 0;
                                                else
                                                                tx_state        <=
fortyus;
                                                                i2 <= i2 + 1;
                                                end if;
                                        when done =>
```

```
                                        if(tx_init = '1') then
                                                tx_state <= high_setup;
                                                i2 <= 0;
                                        else
                                                tx_state <= done;
                                                i2 <= 0;
                                        end if;
                                end case;
                        end if;
                end process transmit;

                with init_state select
                        init_done <= '1' when done,
                                '0' when others;
                with init_state select
                        SF_D1 <= "0011" when s1 | s2 | s3 | s4 | s5 | s6,
                                "0010" when others;
                with init_state select
                        LCD_E1 <= '1' when s1 | s3 | s5 | s7,
                                '0' when others;

        --specified by datasheet
        power_on_initialize: process(clk, reset, init_init) --power on initialization
        sequence
                begin
                        if(reset='1') then
                                init_state <= idle;
                        elsif(clk='1' and clk'event) then
                                case init_state is
                                        when idle =>
                                                if(init_init = '1') then
                                                        init_state <= fifteenms;
                                                        i <= 0;
                                                else
                                                        init_state <= idle;
                                                        i <= i + 1;
                                                end if;
                                        when fifteenms =>
                                                if(i = 1500000) then
                                                        init_state <= s1;
                                                        i <= 0;
                                                else
                                                        init_state <= fifteenms;
                                                        i <= i + 1;
                                                end if;
                                        when s1 =>
                                                if(i = 22) then
                                                        init_state<=s2;
                                                        i <= 0;
                                                else
                                                        init_state<=s1;
                                                        i <= i + 1;
                                                end if;
                                        when s2 =>
```

```
        if(i = 410000) then
                init_state<=s3;
                i <= 0;
        else
                init_state<=s2;
                i <= i + 1;
        end if;
when s3 =>
        if(i = 22) then
                init_state<=s4;
                i <= 0;
        else
                init_state<=s3;
                i <= i + 1;
        end if;
when s4 =>
        if(i = 10000) then
                init_state<=s5;
                i <= 0;
        else
                init_state<=s4;
                i <= i + 1;
        end if;
when s5 =>
        if(i = 22) then
                init_state<=s6;
                i <= 0;
        else
                init_state<=s5;
                i <= i + 1;
        end if;
when s6 =>
        if(i = 4000) then
                init_state<=s7;
                i <= 0;
        else
                init_state<=s6;
                i <= i + 1;
        end if;
when s7 =>
        if(i = 22) then
                init_state<=s8;
                i <= 0;
        else
                init_state<=s7;
                i <= i + 1;
        end if;
when s8 =>
        if(i = 4000) then
                init_state<=done;
                i <= 0;
        else
                init_state<=s8;
                i <= i + 1;
```

```
                                        end if;
                                    when done =>
                                        init_state <= done;
                                end case;
                            end if;
                        end process power_on_initialize;
    end behavior;
```

8　謝辞

最後に本研究を行うにあたり、絶え間ないご指導をいただきました富山高等専門学校制御情報システム工学専攻科 由井四海 教官をはじめ、ご協力を賜りました皆様に心より感謝申し上げます。

ここで教わったことは、これから社会に出て行く上で、お金で買えない価値があると存じます。１年間多大なご協力、本当にありがとうございました。

平成26年 2月　陣内活実、高松 史