

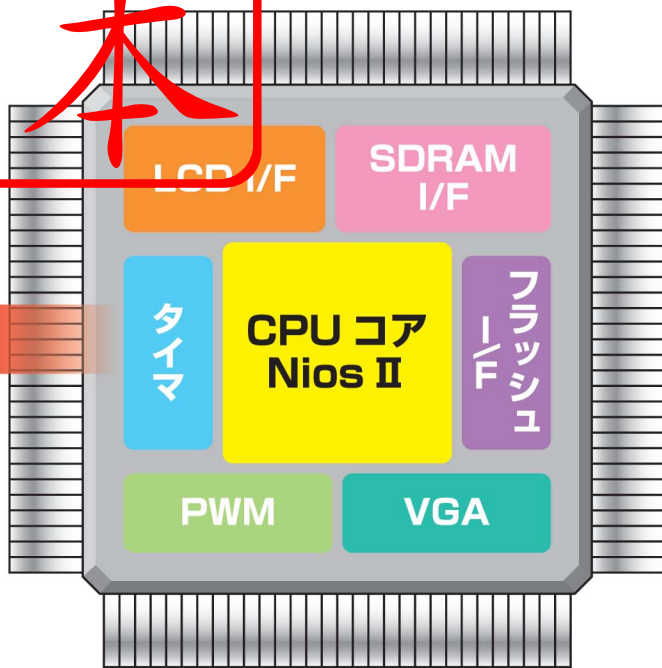
フリーのCPUコアNios II/eと  
高速ロジックで七変化



# FPGAスタータ・キットで初体験! オリジナル・マイコン作り

岩田 利王 著

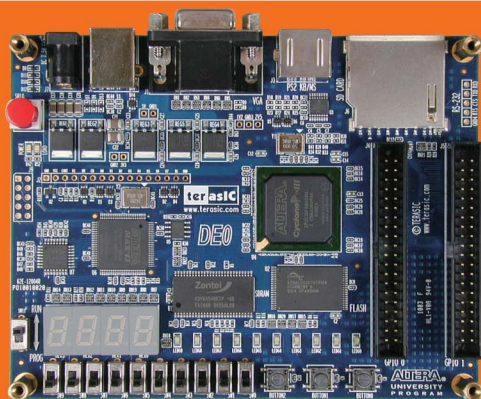
見本



本書の内容を試せる!  
USB対応FPGAキット DEO (別売)

●搭載デバイスと付属品

- ・Cyclone III FPGA (LE数15408個)
- ・8MバイトSDRAM ・4MバイトFLASH
- ・プログラミング/コントロール用USBケーブル
- ・7.5V DC電源 ・DVD-ROM



CQ出版社

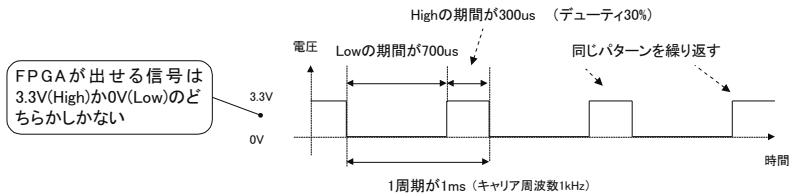
# 第1部 PWM 制御システム編

## プロローグ

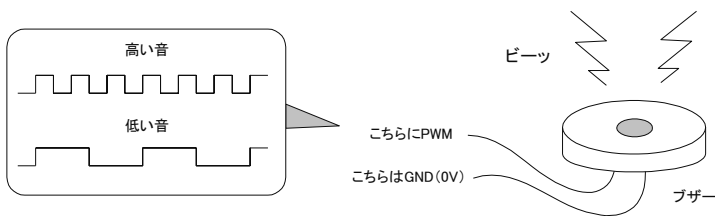
PWM (Pulse Width Modulation, パルス幅変調) は、ブザーや DC モータの制御などによく使われる技術です。そのキャリア周波数を変えることによりブザーから出る音の高さを変えることができ、またデューティを変えることにより DC モータの速度を変えることができます (図 1)。

このように「PWM 制御システム」はいろいろな用途に使われますが、ブザー音の高低、DC モータの速度などはその状況によりさまざまです。従って、PWM のデューティや周波数をフレキシブルに変えられるシステムがあれば便利です。

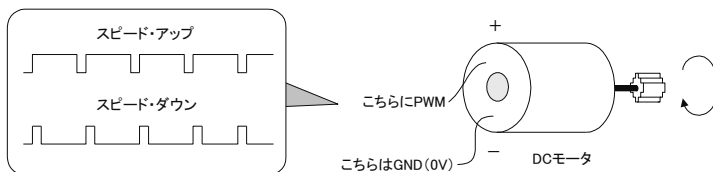
さて、あなたならどのように PWM 信号を発生し、またどのようにそれらのデューティや周波数を制御しますか？マイコンを使う、CPLD を使う…いろいろな技法 (メソッド) がありますが、それぞれに長所、短所があり悩ましいところです。ここでは、4 種類の「設計メソッド」を主張する 4 人の技術者が登場します。



(a) PWM の一例。1周期の逆数をキャリア周波数、Highの期間/1周期をデューティと呼ぶ



(b) PWM のキャリア周波数が高いと高い音、低いと低い音が出る

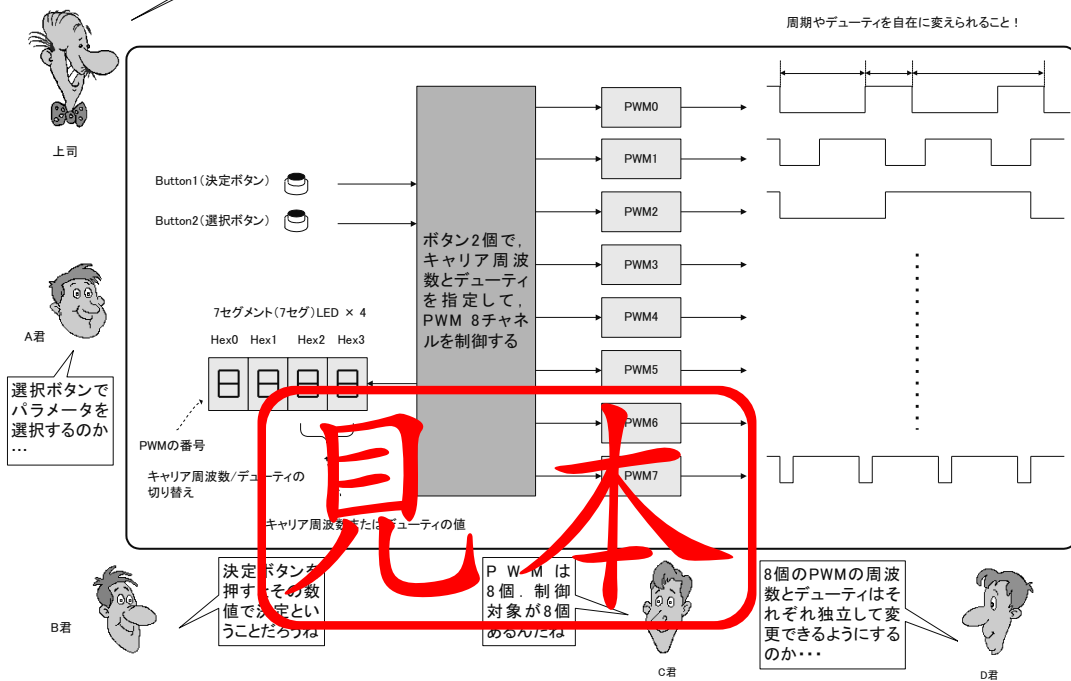


(c) PWM のデューティが大きいと速く回転し、小さいとゆっくり回転する

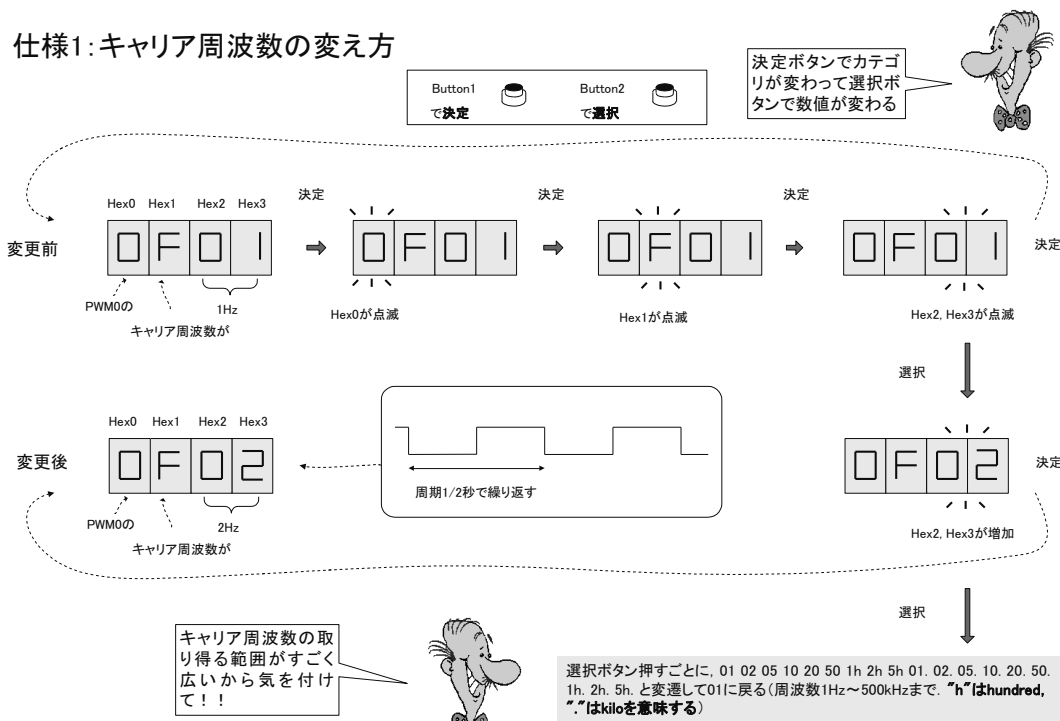
図 1 PWM をブザーや DC モータの制御に使う

指令★このシステムを構築せよ

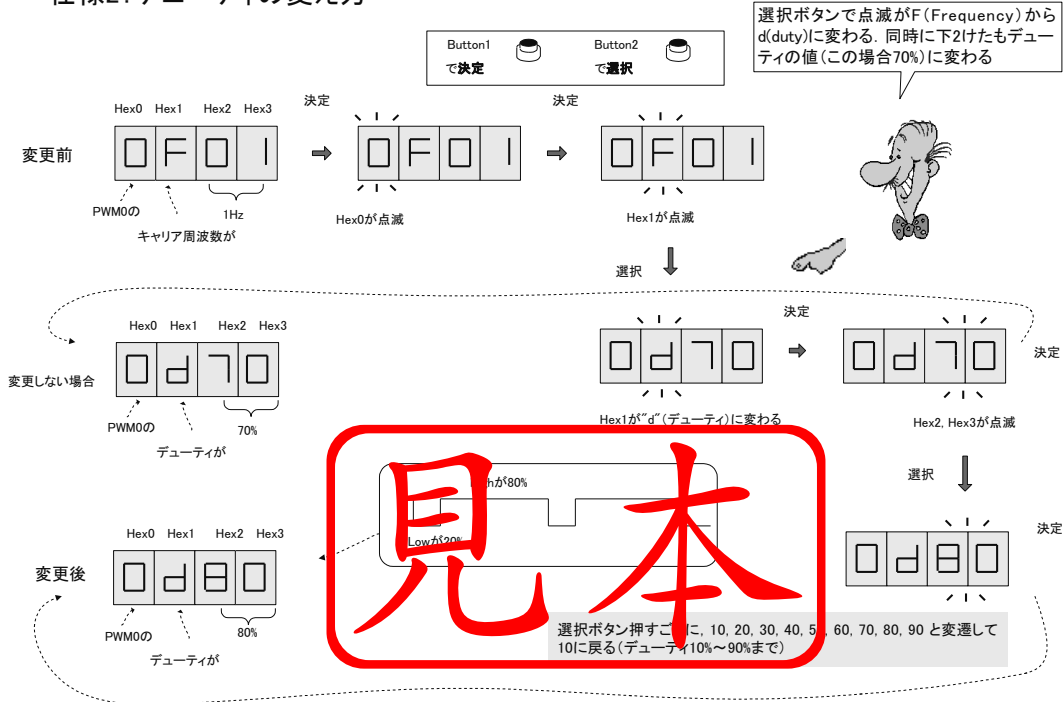
ある日、上司からこんな指令が出ました。さて、あなたならどうしますか？



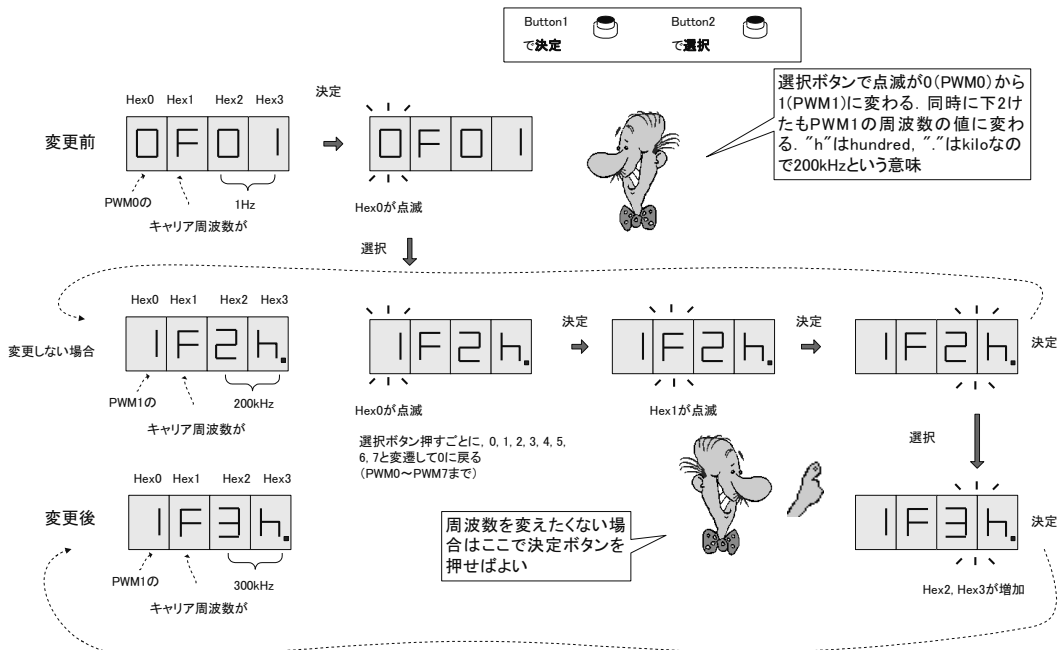
### 仕様1: キャリア周波数の変え方



### 仕様2: デューティの変え方



### 仕様3: 別チャンネルの周波数を変える



## 四者四様のメソドロジ

技術者はそれぞれ自分流のメソドロジ(技法)を持っており、プライドの高い技術者にとってそれはアイデンティティであり、哲学であるためなかなか変えられないものです。ここにいる4人も例外ではありません。

\*:4人に同じものを設計させて一番良いものを採用しようという魂胆でしょうか？  
しかし、「失敗は成功の母」という言葉があるくらいですから、敗者から学ぶことも多いのです。また、ここに出てくる4手法に優劣があるわけではなく、オブジェクト型(目的)に合ったメソドロジ(技法)を採用しましょうということです。

全部マイコンでやる！



A君

このページの下

- 今までマイコンでやってきたので設計資産がある
- C言語だけで済ませたい
- PWMはマイコンのタイマを使って出来そう

やりたいようにやってみなさい



包容力を見せる上司(\*)

次のページの上

- PWMの周波数が速いからロジック向き
- PWMが8chも要るので並列処理が必要
- HDLだけで済ませたい
- ボタンと7セグLEDはステート・マシンで何とかなる

全部CPLD(\*\*)でやる！



B君

マイコン+CPLDでやる！



C君

次のページの下

- PWMなど高速・並列な部分はロジックで処理する
- ボタンや7セグLEDなど条件分岐はマイコンで処理する

次の次のページの上

- 最近話題のNios IIマイコンをFPGAに組み込んでみる
- コスト、基板面積、信頼性などで有利になりそう

Nios IIマイコン入りFPGAでやる！



D君

見本

表1 CPLDとFPGAの違い

ゲート規模	CPLD	FPGA
コンフィギュレーションROM	不要	必要(回路が複雑)
マイコン	実装可能	実装不可

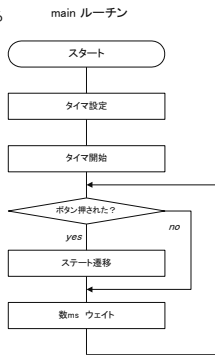
\*\*: Complex Programmable Logic Device, FPGA (Field-Programmable Gate Array)との違いを問われる明確な定義はないが、一般に表1のように言われている。

### A君: 全部マイコンでやる

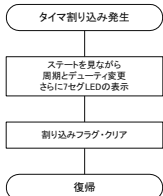
まずフローチャートから見る  
A君(ソフト屋的発想)



ボタンを押すとステート遷移する。ここはmainルーチンで処理。



タイマ割り込み



```

if((reg1 & 0x03) == 0x01) { // button2(選択ボタン)が押された
  if(state[1] == 'N') { // "PNs", Pwm CHN selection
    if(state[2] == '0') {
      state[2] = '1';
    } else if(state[2] == '1') {
      state[2] = '2';
    } else if(state[2] == '2') {
      state[2] = '3';
    }
  }
}

if((reg1 & 0x03) == 0x02) { // button1(決定ボタン)が押された
  if(state[1] == 'U') { // "RUN", free-run state
    if(CHN == 0) state[2] = '0';
    else if(CHN == 1) state[2] = '1';
    else if(CHN == 2) state[2] = '2';
    else if(CHN == 3) state[2] = '3';
  }
}
  
```



ステート遷移をC言語で書くとこんな感じで良いはず！

あっ!!このマイコン、タイマ2個しかない!!



PWMは8chあるし周期もデューティもバラバラだし... どうしよう...



PWMとLEDの変更はタイマ割り込みだな...

問題点を整理してみると...



タイマの数が足りない

並列処理が苦手

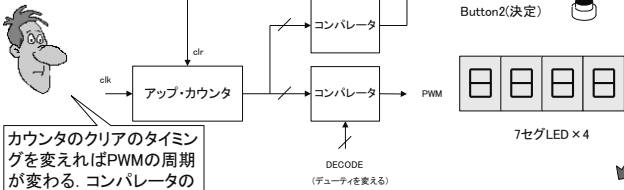
キャリア周波数の取り得る範囲が小さい

基本的にマイコンは1度に一つの処理しかできません。その欠点を解消するために、タイマ割り込みやDMA(Direct Memory Access)などの機能がありますが、今回のようにPWMを8個制御すると使えるマイコンに限られます。

現在ではマイコンが搭載されていない電気製品はなかなか見つけることは出来ないと思います。従って、マイコンを使いこなすことは技術者にとって必須なメソドロジと言えます。しかし、マイコンだけで完結する製品もほとんど存在しないことも事実です。

### B君：全部CPLDでやる

まず回路図から入る  
B君(ハード屋的発想)



カウンタのクリアのタイミングを変えればPWMの周期が変わる。コンパレータのしきい値を変えればデューティが変わる...

PWMをHDLで書くことになる。この部分をコピーすればいくらでもPWMを作れるぞ!

並列処理も楽々!

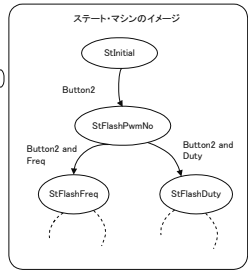
```
always @(posedge CLK or negedge RST_N)
begin
    if (RST_N == 1'b0) begin
        counter0 <= 0;
    end else begin
        if (test0_clr == 1'b1) begin
            counter0 <= 0;
        end else begin
            counter0 <= counter0 + 1;
        end
    end
end

assign counter0_clr = (counter0 == period0) ? 1'b1 : 1'b0;
assign counter0_dec = (counter0 == 0) ? 1'b1 : 1'b0;
```

例えば、マイコン 8個のカウンタを動作させる場合、それらを同時にカウント・アップすることはできないため、マイコンでアップすることになります(順次処理)。それに対しHDL (Hardware Description Language) ではカウンタは並列処理が可能です。並列処理で表現することが出来ます(並列処理)。HDLとは元々回路図だったものを言語で表現するものですからそれは当然と言えます。



問題はボタンと7セグLED...HDLで書くには...ステート・マシンだな



B君は必死でステート・マシンを書きました。その結果は...

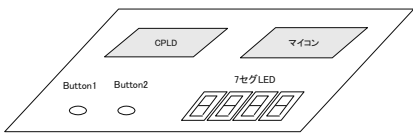
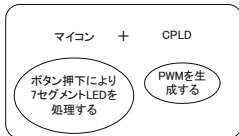


条件分岐の多い処理をHDLで書くとは分りにくくなる(他人には読解不能になりがち)



HDLは複雑な処理を記述するのに優れた言語ですが、複雑な条件分岐の記述は苦手としています。条件分岐をHDLで記述する手法として「ステート・マシン」というものがありますが、どうしてもC言語に比べると分りにくくなってしまいます。幸うじて自分自身では理解出来ていても、他人に理解できない物では「引き継ぎ」が出来なくなってしまいます。今自分がやっている仕事は自分だけのものではない、という認識のもとにコーディングしましょう。

### C君：マイコンとCPLDでやる

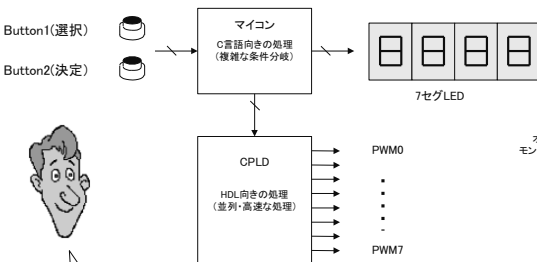


早速実行!

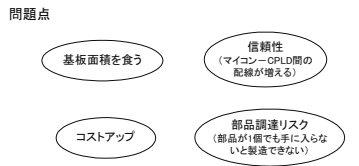
こんな基板を作りたんですけど...

あちゃー。部品が増えるのか...

マイコンとCPLD、適材適所でコーディングすればCもHDLもすっきりしそうだ...



マイコン-CPLD間には、PWMの周期 & デューティを伝えるインターフェースが必要

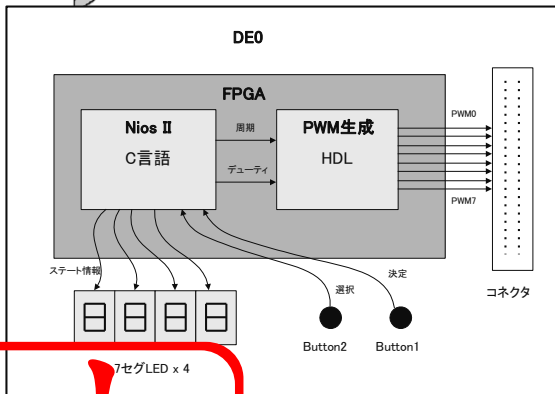
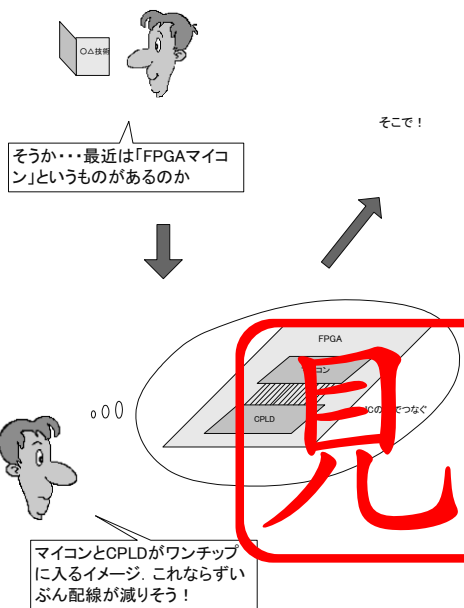


仕様通りに動くものを設計することは技術者にとって最も大事なことは疑う余地がありません。しかし、仕様を達成したからといって、上記のような問題点を残しているとやはり一人前とは見てくれないものです。仕様を満たした上に、安心して使えるもの、安心して製造できるもの、安心してサポートできるもの、さらに儲かるものを世に出すことが出来れば、スーパー技術者として尊敬されると思います。

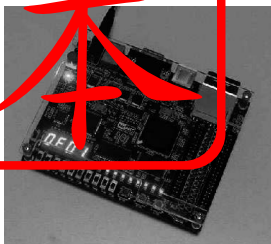
D君: Nios IIマイコン入りFPGAでやる



FPGAスタータ・キットDE0を使えば簡単に実現できる!



見本



こんな感じでDE0にPWM制御システムを実装します!

写真1 完成したPWM制御システム

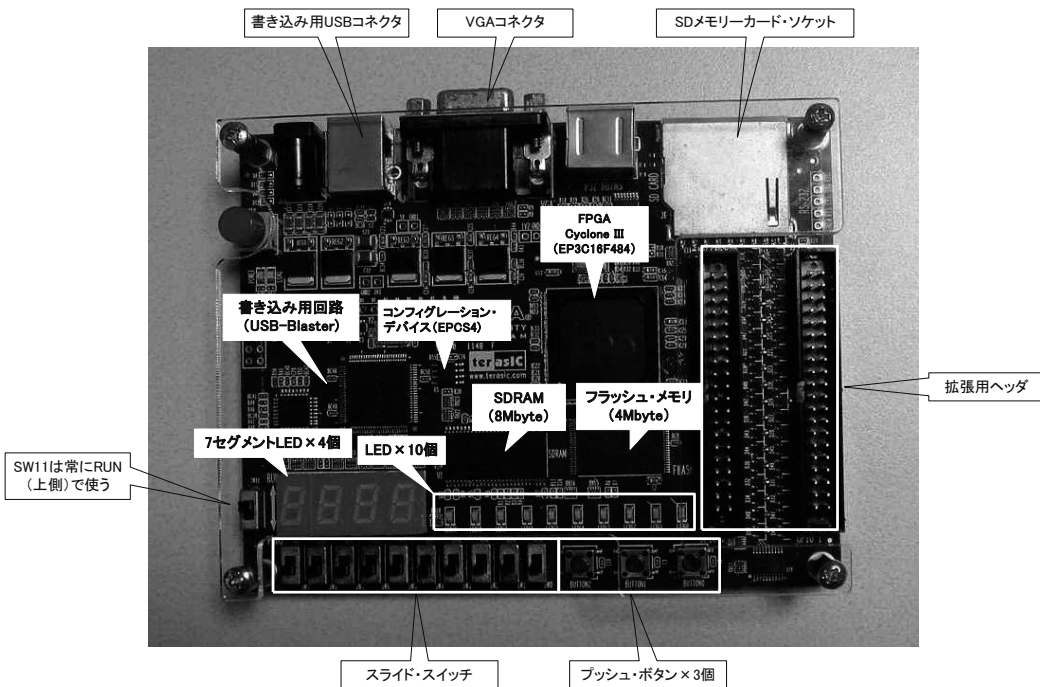
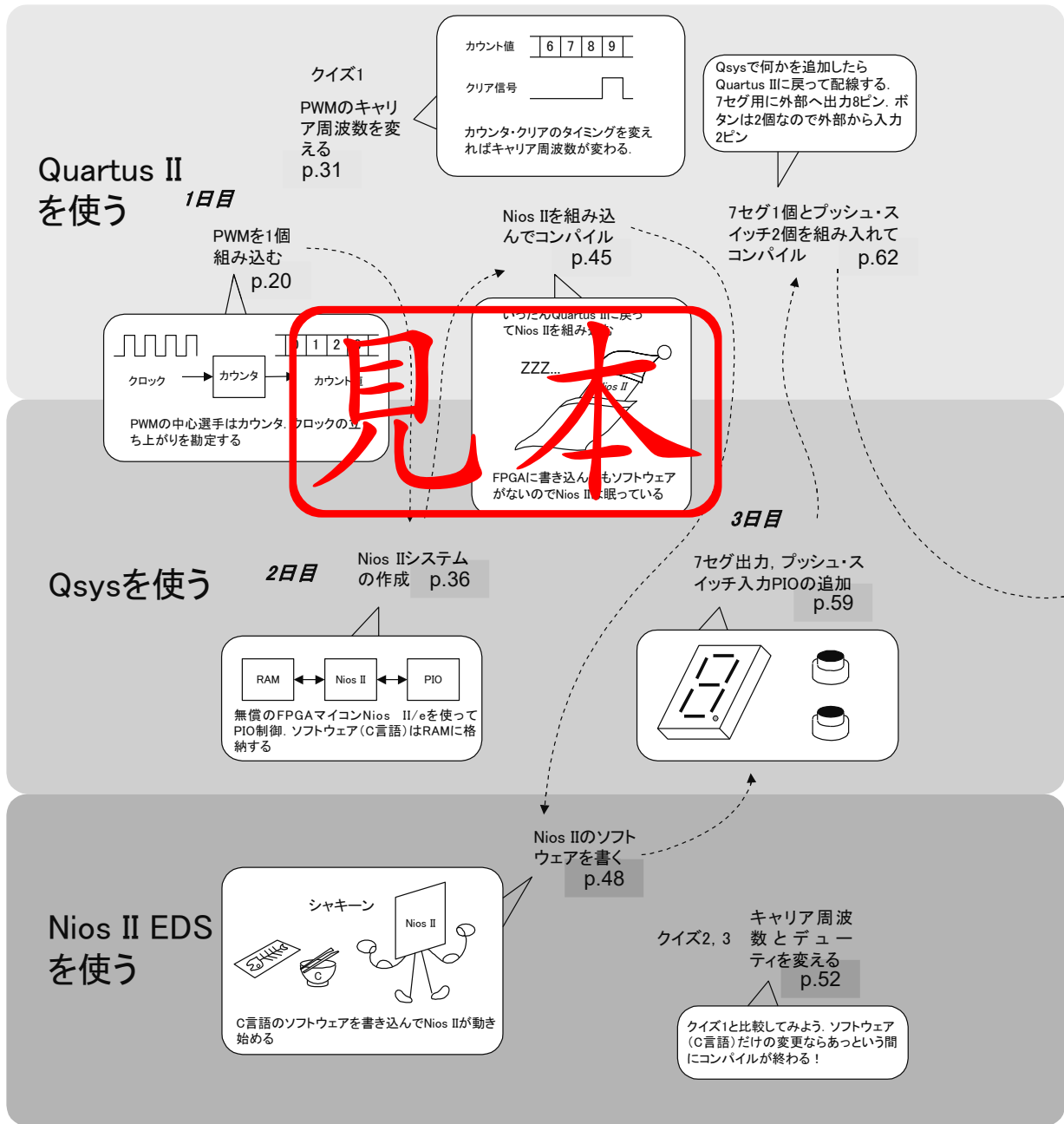


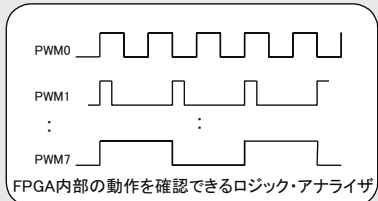
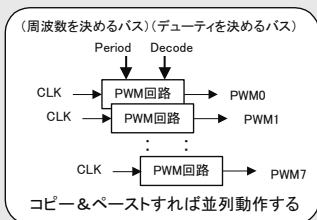
写真2 本書で使用する FPGA スタータ・キット DE0 の外観

第1部ロードマップ (Nios IIマイコン入りFPGAでスマートなPWM制御)





ここまでどうやらD君のメソッドログがベストだということが分かりました。第1章以降ではこのロードマップのように三つのツール(Quartus II, Qsys, Nios II EDS)の間を行き来しながら上司からの指令を達成します。



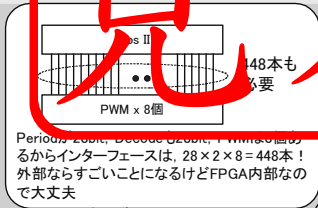
6日目

PWMを8チャンネルに増やしコンパイル (p.74)

7セグを4個に増やしてコンパイル (p.81)

SignalTap IIでPWMの波形を確認 (p.88)

# 見本

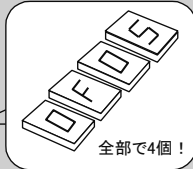


4日目

PWM制御ポートを8個分に増やす (p.69)

5日目

7セグを3個追加 (p.81)

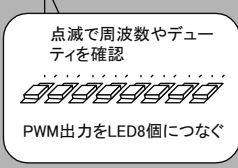


次ページへ続く

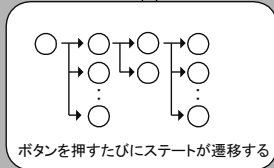
7セグをカウントアップ/ダウン (p.64)



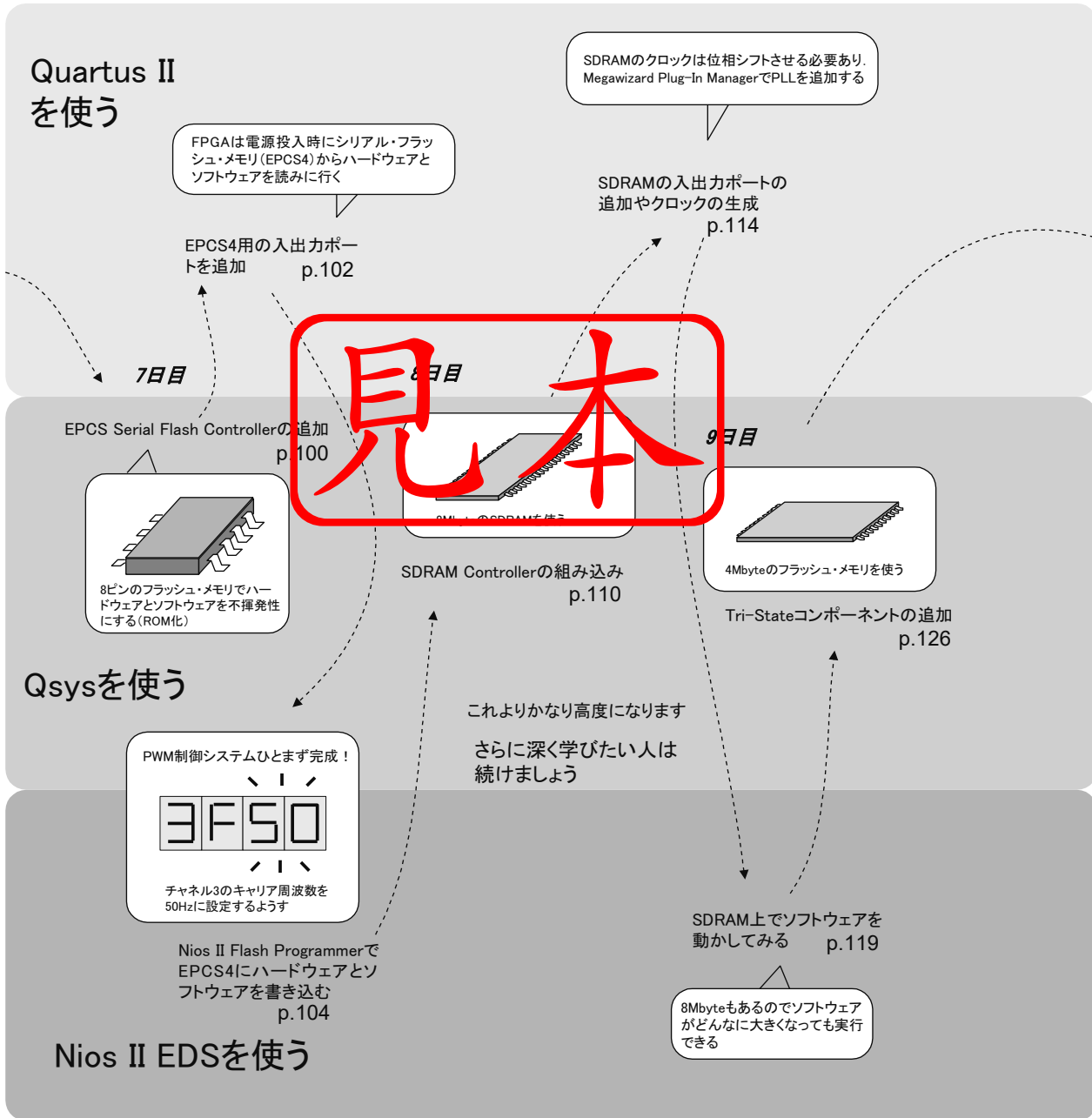
8個のPWMの周波数とデューティを設定 (p.76)



ステート遷移を考える (p.83)



## 第1部ロードマップ（続き）

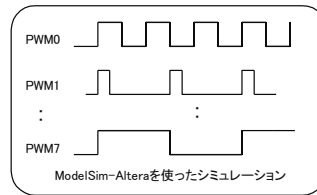


外部フラッシュ・メモリはソフトウェアの格納に使う

モード切り替え用

フラッシュ・メモリ入出力ポートの追加や設定の変更、そしてハードウェアだけをEPCS4にプログラム  
p.132

スライド・スイッチ用の入力ポートを追加  
p.143



11日目

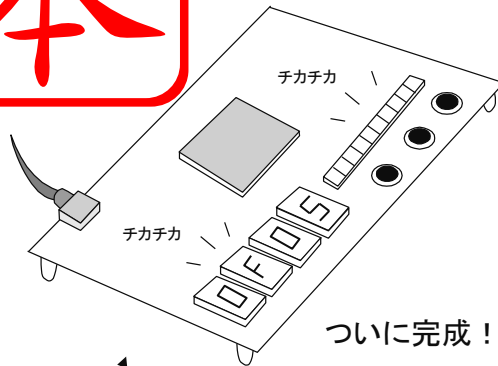
論理シミュレーションでおさらいして終わり!  
p.149

10日目

スライド・スイッチを追加  
p.142

基板の内側に倒すとHighになる

見本



ソフトウェアだけをフラッシュ・メモリにプログラム  
p.136

4Mbyteもあるので大きなソフトウェアでも保存できる

変数をフラッシュ・メモリに書き込んで読み出す  
p.144

設定したPWMのパラメータが電源を切っても保存されている



**FPGAスタータ・キットで初体験!  
オリジナル・マイコン作り**

このPDFは、CQ出版社発売の「FPGAスタータ・キットで初体験！オリジナル・マイコン作り」の一部見本です。

内容・購入方法などにつきましては以下のホームページをご覧ください。

内容 <https://shop.cqpub.co.jp/hanbai/books/48/48191.html>

購入方法 <https://www.cqpub.co.jp/order.htm>