2. 論理設計の基礎

1. 目的

必要な機能をユーザ (回路設計者) 自身がプログラム可能な論理 IC であるプログラマブル・ロジック・デバイス (Programmable Logic Device, 以下 PLD) とハードウェア記述言語 VHDL を用い, 簡単な論理回路を対象 としたディジタル回路の設計法を理解する.

2. プログラム可能な論理 IC ~ PLD ~

2.1. PLD の特徴

PLDは、その名のとおり「ユーザ自身が設計したロジック回路を実チップとして動作させることが可能となる プログラマブル・デバイス」であり、現在のディジタル回路設計の世界では、PLDを抜きに語れない時代になっ ている.比較的最近までは、ASICの試作のため、あるいは少量生産システムでのみ利用されていたが、最近で はあらゆる組み込み機器で利用されていると言っても過言ではない状況になってきている.また、量販店で販売 されているような電気製品にも採用されるようになってきており、多種の PLD が発売され用途に応じて使い分け できるようになってきている. PLD は、大別して CPLD(Complex PLD) と FPGA(Field Programmable Gate Array) がある.

2.2. PLD の基本構造

PLDの内部には、プログラム可能なマクロ・セルと呼ばれる素子と、マクロ・セル間を接続するための配線リ ソースが多数内蔵されている. PLDの内部構造のイメージを図1に示す.



図 1: PLD の内部構造例

● マクロ・セル

マクロ・セルは,組み合わせロジックを実現するプログラム可能なロジック・モジュールと,順序回路を構築 するためのDフリップフロップ¹から構成されている.

ユーザが設計した回路は, PLD 設計・開発用ツールによって最終的にマクロ・セル単位に分割される. 各マク ロ・セル内のロジック・モジュールは, その分割された回路が機能するようにプログラムされる.

¹一種の記憶回路であり、Cp にクロック入力が加えられた瞬間に D 入力の 0/1 の状態が Q 出力にセットされる.

● 配線リソース

分割されたそれぞれの回路を含むマクロ・セル間は, 配線リソースを介して接続される. この配線リソースは あらかじめデバイスの縦横方向に用意されている.

● 1/0セル

デバイスの I / O ピンと直結したセルである. I / O セルではデバイス内部のマクロ・セルとは異なり, 組み 合わせロジックは構成できない.

3. VHDLの概要

3.1. HDL 設計のメリット

ハードウェア記述言語 (Hardware Description Language, 以下 HDL) は、その名のとおりハードウェアの動作 を記述する一種の言語である.

HDL による設計手法は,現在すでに ASIC(Application Specified Integrated Circuit:特定用途向け集積回路) などの大規模集積回路の設計でさかんに利用されており, FPGA や CPLD などを使用する比較的小規模な設計にも様々なメリットをもたらす.表1,図2に今までの回路図入力による設計と HDL による設計の比較を示す.

	回路図入力	HDL 入力
1	回路図入力に時間がかかる	テキストで簡単に入力
2	回路変更が困難	回路変更が容易
3	設計者以外では内容を理解しづら	動作の記述のため内容を理解
	<i>۷</i> ۷	しやすい
4	特定の半導体メーカのライブラリ	半導体メーカのライブラリに
	を考慮して回路図入力	依存せず、どのメーカにも容
		易に対応可

表 1:回路図入力,HDL 記述各入力による設計



ハーフアダー(半加算器)の真理値表

Х	Y	S	С	
0	0	0	0	
0	1	1	0	
1	0	1	0	
1	1	0	1	

図 2: 回路図入力, HDL 記述各入力による設計 (ハーフアダー)

3.2. HDL の種類

HDLには、代表的なものとして VHDL と Verilog-HDL がある. 各々を比較した場合、様々な立場によっ て優劣が左右されるが、本実験では VHDL を使用する.

3.3. VHDL の特徴

VHDLは、さまざまなレベルでの記述が可能である。たとえば、システム全体のアルゴリズムを記述したり (アーキテクチャ・レベルでの記述)、ハードディスクのデータのやり取りや、モータの制御などをモデル化してシ ステム全体を抽象度の高いレベルで記述したり(ビヘイビア・レベルの記述)、ロジック回路生成が可能なレベル で記述したり(RTL: Register Transfer Level での記述)できる。もちろんロジック・ゲートレベルでの記述も可 能である。

4. 実験概要,使用装置

本実験では、教育用端末の Windows 上で動作する ALTERA 社製 Quartus II 開発ソフトウェアを使用し、 ディジタル回路の設計,合成 および論理シミュレーション (Quartus II での動作シミュレーション) を行う. そ の後、合成した回路を同アルテラ社製 Altera DE0 開発・学習ボードに転送し、実際に DE0 ボード上で設計し た回路の動作確認を行う.

また、本実験での回路の設計には、VHDL と呼ばれるハードウェア記述言語を使用する. 今まではディジタル 回路の設計には、設計する回路の論理式などから and や or などのゲートを並べてそれらの間を結線することに より設計されてきた. しかし近年では、VHDL や Verilog-HDL のようなハードウェア記述言語により、ディ ジタル回路をソフトウェア的なスタンスから設計できるようになっている.



図 3: 本実験でのディジタル回路設計の流れ

5. 実験

5.1. 実験内容

本実験では、ディジタル回路の設計方法を理解することを目的として、以下の3種類の回路について 回路設計~回路合成~論理シミュレーション~DE0ボード上での動作確認まで行う.

• 必修課題(上記の一連の作業を行う)

- [1] 真理値表を元に全加算器を設計 (プログラム記述),回路合成した後,設計した回路を DE0 ボードへダ ウンロードし,ボード上で動作確認を行う.
- 【2】 【1】で設計した全加算器を4つ用いた,図4に示す4ビット並列形加算器(4ビットリップル加算器) の設計し,DE0ボードでの動作確認を行う.



図 4:4 ビット並列形加算器

• 検討課題 (実験作業終了後,後日各自でプログラム作成のみ行う)

- 【3】 真理値表を元に7セグメント LED と 8P ディップスイッチを使用した2進-10 進デコーダの設計(VHDL によるプログラムの作成)を行う.
- # 以降,本実験テキスト,「Quartus II の使い方」/「DE0 開発・学習ボードの使い方」を合せて参照しなが ら実験を進めていく.

5.2. 全加算器の設計

(i) 真理値表 · 論理式

全加算器を設計するために,最初に真理値表(表 2)から必要な論理式を導き出す.表 2(a)は,普通使われている 0/1 による '正論理'の真理値表,表 2(b)は,負論理動作も考慮して信号を L/H 表現とした真理値表である.

(a) 0/1 表現 (正論理)						(b)	L/H	表現	1
	Y	CI	S	CO	X	Y	CI	S	
0	0	0	0	0	L	L	L	L	
_	0	0	1	0	Η	L	L	Н	
0	1	0	1	0	L	Η	L	H	
L	1	0	0	1	Н	Η	L	L	
)	0	1	1	0	L	L	Η	H	
1	0	1	0	1	Н	L	Η	L	
0	1	1	0	1	L	Η	Η	L	
1	1	1	1	1	Η	Η	Η	Н	

表 2: 全加算器の真理値表

表2から、全加算器の論理式は次のように導き出される.

以上の論理式および表2での入出力名を使用し、VHDLにより記述(設計)された全加算器の記述例をサンプル1に示す.

```
-- (サンプル1) 全加算器

library IEEE;

use IEEE.std_logic_1164.all;

entity FULLADDER is

port( A, B, CI : in std_logic;

S, CO : out std_logic );

end FULLADDER;

architecture DATAFLOW of FULLADDER is

begin

S <= (A xor B) xor CI;

CO <= (A and B) or ( (A xor B) and CI);

end DATAFLOW;

('--' で始まる行はコメント行)
```

(ii) Quartus II による回路設計手順

前節のサンプルを用い (正しい論理式であれば,上の同じでなくても良い),Quartus II でディジタル回路の設計・合成・シミュレーションを行う.次に述べる設計時の各手順については,別紙「Quartus II の使い方」も参照しながら行う.

【1】作業用フォルダの作成 (「Quartus II の使い方」 P4)

コンピュータの画面の左下の丸い部分 (スタート) から「コンピュータ」を開き, さらにその中の「ネットワークの場所」にある windows ディスクを開き, その中に班のアルファベット名のフォルダを作成する. なお,日本語等の全角文字は使用できません.

[2] Quartus II の起動 (「Quartus II の使い方」 P5)

デスクトップ上の 「Quartus II」という名前の青いアイコンをダブルクリックして起動.

【3】プロジェクトの設定(「Quartus II の使い方」 P7~ P15)

設計する回路の名前を定義する.「Quartus II の使い方」に沿って作業すること.

- (使い方 P6): プロジェクト名は 'fulladder' とする. プロジェクトは, 作成した作業用フォルダ内に 設定する.
- (使い方 P7): そのまま「Next」をクリック
- (使い方 P8): ウインドウ上の Family: は Cyclone III を選択し、その後下の Available devices: で EP3C16F484C6 を選択して「Next」をクリック.
- (使い方 P9): そのまま「Next」をクリック

【4】回路設計 (HDL 記述入力) (「Quartus II の使い方」 P16~ P17)

サンプルプログラムを VHDL 入力ウィンドウ (エディタ) を用いて入力する. 一行目のコメント文 「-(サンプル1) 全加算器」 は記述しない. プログラムは 'fulladder.vhd' として保存する.

【5】設計回路の合成(「Quartus IIの使い方」 P18~ P19)

ここで,設計(入力)したプログラムのコンパイルを行い,プログラム内の入出力宣言や論理式から,実際の回路(and や or などのゲートにより構成される回路)に合成する.何らかのエラーが出た場合には,エラーの箇所を修正しエラーが出なくなるまで再度コンパイルしてチェックする.このとき,幾つかのワーニングが出る場合があるが,この課題ではとりあえずワーニングは無視.

【6】入出カピンの配置 (「Quartus II の使い方」 P20~ P22)

設計(合成)される回路は,最終的には DE0 ボード上の FPGA に書き込まれ,その後動作確認が行われる.しかし動作確認を行うためには,FPGA の適切なピンにより信号の入出力しなければならず,ここではそのためのピンの割り当てを行う.「Quartus II の使い方」には二種類の割り当て方法が記載してあるが,どちらか一方のやり易い方で実行する.ピン配置が終ったら,再度コンパイルを実行する.この課題では,表3に示すピン番号を使用する.

学科 Wiki のハードウェア実験のページにある別紙「動作確認方法」の pdf 資料の2ページ目も確認する。

信号名	ピン番号	機能
А	J6	スライドスイッチ SW0
В	H5	スライドスイッチ SW1
CI	H6	スライドスイッチ SW2
S	J1	LED0
CO	J2	LED1

表 3: 全加算器のピン配置

【7】設計した回路の DE0 ボードへの実装 (「Quartus II の使い方」P33~ P36)

以上の作業が終ったら, Quartus II からパソコンに接続された DE0 ボードへ回路の転送 (ダウンロード) を行う. その後, ピン配置したスライドスイッチと LED を用いて動作確認を行う. ここで目的の動作が行 なわれることが確認されたら, この回路についての一通りの設計が終了.

(注) 「Quartus II の使い方」28ページ冒頭にあるジャンパの設定については無視して下さい.

※実験ボードでの動作確認の際は担当の方に確認してもらうこと.

5.3.4ビット並列加算器の設計

前で設計した全加算器をサブ・モジュールとして利用し,図4に示す4ビット並列加算器を設計する.記述例 をサンプル2に示す.但し,このサンプルはまだ不完全なので,このサンプルを用いて回路を完成させる.

```
-- (サンプル2) 4ビット並列加算器
library IEEE;
use IEEE.std_logic_1164.all;
entity FULLADDER4 is
 port( A0, A1, A2, A3 : in std_logic;
       B0, B1, B2, B3 : in std_logic;
       CI
                   : in std_logic;
       S0, S1, S2, S3 : out std_logic;
                   : out std_logic );
       CO
end FULLADDER4;
architecture STRUCTURE of FULLADDER4 is
component FULLADDER
 port( A, B, CI : in std_logic;
       S, CO : out std_logic );
end component;
signal C1, C2, C3 : std_logic;
begin
 FAO : FULLADDER port map ( AO, BO, CI, SO, C1 ); -- 0 ビット
目の全加算器
 FA1 : ~ 以降, 1ビット目から3ビット目までの全加算器について
 FA2: ~ 0ビット目と同様に記述する.この際,隣の全加算器との
 FA3: ~ 入出力の接続に注意して記述する.
end STRUCTURE;
```

(i) 4 ビット並列加算器の設計手順

Quartus II の起動から DE0 ボードを用いた動作確認までの作業手順は前の全加算器の場合と"ほぼ"同じであるが、異なる点を以下に示す.

- 【1】プロジェクト名,プログラムファイル名,およびプログラムでのエンティティ名は 'FULLADDER4 と する.
- 【2】 プロジェクトは全加算器と同じフォルダに作成するが、このとき「Quartus II の使い方」の P6 で「Next」 をクリックした後に出てくるダイアログ画面では「いいえ」をクリックして次に進む.
- 【3】 「Quartus II の使い方」の P6 で fulladder.vhd を追加する (右側の「Add All」ボタンをクリック)
- 【4】サンプル2の FA1, FA2, FA3 には FA0 と同じ書式の記述になるが, FA0 ~ FA3 は 実験書図5の F0 ~ FA3 に対応する.よって, 桁上りの伝播などに留意して記述していく必要がある.
- 【5】この回路でも動作確認では全加算器と同じスイッチと LED を用いる。
- (注) ただし,使うスイッチと LED の数は全加算器よりも多くなるので,学科 Wiki のハードウェア実験のページにある別紙「動作確認方法」の pdf 資料の3ページ目を参考にピン配置を行なう。

※4ビット並列加算器の動作は必ず担当の方に動作確認をしてもらう事.

以上の作業が終了したら実験室での実験作業は終了

(注)次ページの検討課題は、各自で作業を行いレポートに作成したプログラムを添付する。

5.4. (検討課題) 7 セグメント LED を用いた 2 進-10 進デコーダの設計

「2進-10進デコーダ回路」は、DE0ボードにある7セグメントLED(1つ)と4つのスライドスイッチを用い、 スイッチの4桁2進数を0~9までの10進数表現に変換して表示する回路である.

~この課題は、後日各自でVHDLによるプログラムを作成し、レポートに添付すること、~ ※実験担当者から動作確認してもらう必要はありません

○設計上の注意

この課題は、全加算器と同じ手法で真理値表を用いて回路設計を行う. 但し、DE0 ボードでは LED の 消灯/ 点灯 と信号値の対等が、使う LED の種類によって異なる. 全加算器、4 ビット並列加算器で使用した LED では 「消灯:信号'0'、点灯:信号 '1'」であったが、7 セグメント LED では逆で「消灯:信号 '1'、点灯:信号 '0'」 となっている. そこで、次の表4のような表現の真理値表を用います.

.....

(i) 真理值表

表4は、スライドスイッチが上の時を'**OFF**、下の時を'**ON**' と考え、7セグメント LED が消灯の時を '**OFF**'、 点灯の時を'**ON**' と考えて作成した真理値表である.

	スイッ	チ入力		7 セグメント LED								
I3	I2	I1	I0	a b c d e				f	g			
OFF	OFF	OFF	OFF	ON	ON	ON	ON	ON	ON	OFF		
OFF	OFF	OFF	ON	OFF	ON	ON	OFF	OFF	OFF	OFF		
OFF	OFF	ON	OFF	ON	ON	OFF	ON	ON	OFF	ON		
OFF	OFF	ON	ON	ON	ON	ON	ON	OFF	OFF	ON		
OFF	ON	OFF	OFF	OFF	ON	ON	OFF	OFF	ON	ON		
OFF	ON	OFF	ON	ON	OFF	ON	ON	OFF	ON	ON		
OFF	ON	ON	OFF	ON	OFF	ON	ON	ON	ON	ON		
OFF	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF		
ON	OFF	OFF	OFF	ON	ON	ON	ON	ON	ON	ON		
ON	OFF	OFF	ON	ON	ON	ON	ON	OFF	ON	ON		

表 4: ON/OFF 表現による真理値表

(ii) 論理式の導出

次に,表4から論理式を作成する. DE0 ボードは,前に述べたようにスイッチが上の時が信号'1',下の時が信 号'0',7セグメント LED は信号'1'が消灯,信号'0'の時が点灯なので,入力・出力とも 'OFF = '1', ON = '0' として論理式を導出していく.

例えば出力信号'1'の組み合せで論理式を導出すると 'a'の論理式とその VHDL 表記は次のようになる.

 $a \le (not I0 and I1 and I2 and I3) or (I0 and I1 and not I2 and I3) (\leftarrow VHDL 表記)$

同様に、以下 'b' から 'g' までの論理式を求めていく (必要に応じて論理式の簡単化も可).

(iii) VHDL による 2 進-10 進デコーダ回路の設計

設計方法(プログラムの記述手順)は**全加算器**の時と同じである.

- 6. レポートについて
 - (重要)~ 実験に関する注意事項~ にある 2.実験レポートについて【2】の表紙の記入項 目は忘れずに記入すること.未記入項目がある場合,適正に評価されないことがある.

レポートの提出期限は実験終了日から2週間後まで.提出するレポートには,

- 目的,実験装置
- 実験方法:簡単でいいですが,実際に行なった実験作業について流れがわかるように書く事.
- 実験結果・考察:作成したプログラム
- 検討課題 (結果·考察·課題):

を書いて提出する. なお、レポートに添付する実験結果は

- (必須課題) 4 ビット並列加算器のプログラムファイル
- (検討課題) 7 セグメント LED のプログラムファイル

.....

参考文献

- [1] "VHLDによるハードウェア設計入門",長谷川裕恭, CQ出版社
- [2] "A VHDL Primer VHDL言語入門", Jayaram Bhasker, CQ出版社
- [3] "FPGA活用チュートリアル", CQ出版社
- [4] "FPGA/PLD設計スタートアップ", CQ出版社

参考:7セグメント LED を用いた2進-10進デコーダの設計(実験作業編)

ここには,検討課題の「7 セグメント LED を用いた 2 進-10 進デコーダの設計」について,必修課題の「全加算器」や「4ビット並列加算器」と同じく DE0 ボードを使って動作確認する場合の参考になるような留意点を記載しています.

○設計上の注意

この課題は、全加算器と同じ手法で真理値表を用いて回路設計を行う. 但し, DE0 ボードでは LED の 消灯/ 点灯 と信号値の対等が, 使う LED の種類によって異なる. 全加算器、4 ビット並列加算器で使用した LED では 「消灯:信号'0'、点灯:信号 '1'」であったが, 7 セグメント LED では逆で「消灯:信号 '1'、点灯:信号 '0'」 となっている. そこで、次の表 4 のような表現の真理値表を用います。

.....

(i) 真理值表

前に出た表4は、スライドスイッチが上の時を'**OFF**、下の時を'**ON**' と考え、7セグメント LED が消灯の時 を '**OFF**'、点灯の時を'**ON**' と考えて作成した真理値表である.

7	7 セグメント LED									
I3	I2	I1	I0	a	b	с	d	е	f	g
1	1	1	1	0	0	0	0	0	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	0	1	0	0	1	0	0	1	0
1	1	0	0	0	0	0	0	1	1	0
1	0	1	1	1	0	0	1	1	0	0
1	0	1	0	0	1	0	0	1	0	0
1	0	0	1	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1
0	1	1	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	1	0	0

表 5: ON/OFF 表現による真理値表 (表4の ON/OFF を 0/1 に変更したもの)

(ii) 論理式の導出

次に、表4から論理式を作成する.

DE0 ボードは、前に述べたようにスイッチが上の時が信号'1'、下の時が信号'0'、7 セグメント LED は信号'1' が消灯、信号'0'の時が点灯なので、入力・出力とも '**OFF = '1'、ON = '0'** として論理式を導出していく.す ると、上の表 5 のようになる。

この場合、例えば出力信号'1'の組み合せで論理式を導出すると 'a' の論理式とその VHDL 表記は次のようになる.

 $a = \overline{I0} \cdot I1 \cdot I2 \cdot I3 + I0 \cdot I1 \cdot \overline{I2} \cdot I3 \quad (\leftarrow ``mtext{metric} a = \texttt{I})$

 $a \le ($ not I0 and I1 and I2 and I3) or (I0 and I1 and not I2 and I3) (← VHDL 表記)

同様に、以下 'b' から 'g' までの論理式を求めていく (必要に応じて論理式の簡単化も可)。

(iii) VHDL による 2 進-10 進デコーダ回路の設計

以下、Quartus II を使用して設計を行いますが、設計作業は**全加算器**の時と同じである.ピン配置は、表5を 使用すること.



図 5:7セグメント LED を用いた2進-10進デコーダのピン配置

(重要)回路名 (プロジェクト名 = *entity* 名 = ソースファイル名) は、必ず半角英文字で始まる名前にすること. (7segment など、数字で始まる名前は不可)

(iv) DE0 ボードでの動作確認

4つのスイッチを用いて,真理値表にある 2進数の "1111" ~ "0000"を入力すると、それに対応して 7 セグメント LED が数字の 0 ~ 9の組合せで点灯する.具体的には,例えば入力値が 10進数 '1' 2進数では "1 1100") び場合は,セグメント 'b' と 'c' が点灯するように設計されていれば良い.