Interface増刊 “ゼロ”から学び，プログラミングを楽しもう

# インターフェースZERO

No. 04

http://zero.cqpub.co.jp/

2大メーカXilinx，Alteraのお手軽ボードでチョコッと体験!

# Hello World から始めるFPGA入門

論理回路設計に挑戦！

三好 健文 著

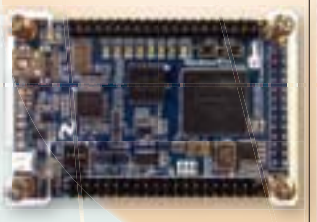
ソフトウェア・プログラマに贈る



動作確認済み

本誌の内容を試せる定番FPGAボード

| Xilinx社の評価ボード | | Altera社の評価ボード | |
|---|---|---|---|
| ①MicroBoard（Avnet社） | ②Atlys（Digilent社） | ③DEO nano（Terasic社） | ④DEO（Terasic社） |



CQ出版社

第1章 ソフトウェアとハードウェアの関係を理解する

# ハードウェア・プログラミングを始める第一歩

## 1.1 もし○○なプログラムが書けたなら…

ソフトウェアの設計をしていて，こんな思いをしたことはありませんか？

- たくさんのスレッドを，並列に低コストで実行できたらいいのになぁ
- 361 ビットの演算を，一発でできたらすっきり書けるのになぁ
- グラフィクスの特別な命令が使えると，処理を高速化できるのになぁ
- 100 個ほど入出力があれば，モータやセンサをたくさんつなげられるのになぁ
- このシステム，乾電池 1 本で動作しないかなぁ

など，ソフトウェアではあと一息痒いところに手が届かず，歯痒い思いをしたことはないでしょうか．

### ● プログラムはプロセッサに束縛されている

パソコンはもちろんのこと，今やテレビや携帯電話，自動車などのあらゆる製品の中でソフトウェアが動作しています．それらのソフトウェアは，実現したいアプリケーションや動作させる環境に応じて，さまざまなプログラミング言語で記述されています．例えば，JavaScript は Web アプリケーションを便利で華やかにしてくれますし，

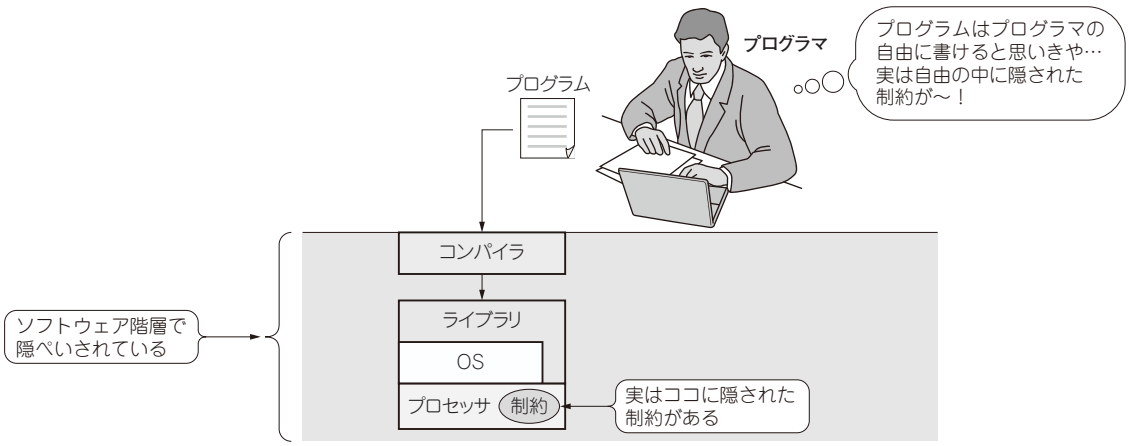



図 1.1　プログラムはさまざまな処理を自由に書けるが，実はハードウェアの制限がある

第2章 Hello World 表示や標準入出力の基本プログラムを書く

# HDL だって，ふつうのプログラミング言語

まずは，ふつうのプログラミング言語のようにHDL を使ってみましょう．

HDL は，ハードウェアを設計するための言語ですが，ハードウェアの動作をパソコンの上で擬似的に動作させることができます．

まずは，この擬似的な動作で，プログラミング言語としての HDL に慣れてみましょう．ここでは，Verilog HDL というハードウェア記述言語を使います．

## 2.1 プログラミングは Hello World から始まる

### ● プログラミング環境の準備

多くのソフトウェア・プログラミングの処理系と同じように，Verilog HDL の処理系にも有償のものや無償のものなどのさまざまなものがあります．ここでは，Icarus Verilog というコマンドライン・ベースで使う Verilog HDL 処理系（Verilog HDL を実行できる環境）を使用します．

Icarus Verilog は，オープン・ソースの Verilog HDL 処理系で，Windows，Linux，MacOS X などの幅広い環境で使うことができます．Linux や MacOS X であればパッケージ・システムを使って簡単にインストールでき，Windows であれば Cygwin を使って自分でビルドしたり，コンパイル済みバイナリをダウンロードしてインストールすることができます．詳しいインストール方法は，**Appendix 1** を参照してください．

### ● Hello World プログラムの作成

それでは，Verilog HDL のプログラミングを始めてみましょう．まず，多くのプログラミング言語の教科書にならって，最初は“Hello World”から始めます．**リスト 2.1** が，Verilog HDL で記述した“Hello Verilog World”をコンソールに表示するソース・コードです．

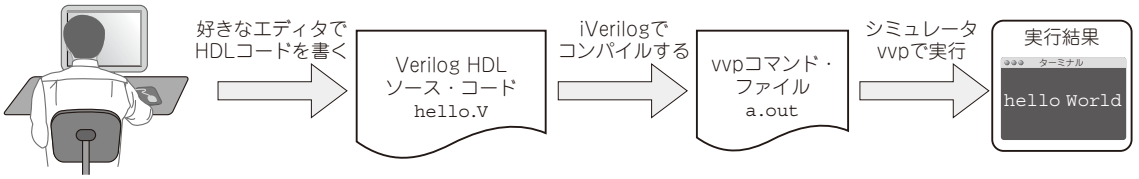



**図 2.1　HDL コードを Icarus Verilog を使ってコンパイルし実行するまでの流れは，C 言語の場合と同じ！**

第3章 HDLだから記述できる並列処理にチャレンジ！

# 画像ファイル解析プログラムを作成しよう！

**図 3.1　256×256 のサイズに縮小した「レナ画像」**
ファイル名は lenna.bmp とした．実際はカラー画像．

本章では**第 2 章**での基本プログラムを応用して，入力された少し複雑なデータを解析する例として，画像ファイルの解析を考えてみましょう．

ここでは，24 ビットのビットマップ画像（レナ画像，**図 3.1**）を読み込み，カラー画像からグレースケール画像に変換する HDL プログラムを作成します．

そして，ハードウェア記述言語 Verilog HDL だから可能な並列処理の記述方法について解説します．

なお，入力に使用した画像は**図 3.1** に示したオリジナルのレナ画像（512×512）を 256×256 のサイズに縮小したファイルです．

## 3.1 画像データ・ファイルを表示する

### ● 画像解析の準備

#### ▶ ビットマップ・ファイルの形式

ビットマップ・ファイルは，Windows でよく使われている無圧縮の画像ファイル形式の一つです．ヘッダ領域とデータ領域から構成され，ヘッダ領域の情報によって，いくつかの形式のデータを扱うことができます．

大きく分けて，Windows ビットマップ形式と OS/2 ビットマップ形式がありますが，ここでは Windows ビットマップ形式のみを扱うことにします．Windows ビットマップ形式のファイル形式を**図 3.2** に示します．

#### ▶ 解析の基本方針

画像データを解析するには，まず，ファイルを読み込まなければいけません．ここで，ファイルを読み込みながら解析をする方法と，全部読み込

第4章 ハードウェアをプログラムする基本ルールを習得する

# VHDLとVerilog HDLの基礎概念と文法

普段コミュニケーションで使用する日本語や英語といった自然言語でも，C言語やJavaのようなプログラミング言語でも，正しく使用するためには文法の知識が欠かせません．同じように，ハードウェアを設計する際にはHDLの文法の知識が必要です．ここで，HDLの基本文法をしっかり押さえておきましょう．

本章では，ハードウェア記述言語（HDL）の中で，よく使用されているVHDLとVerilog HDLの二つを対比しながら基本文法を説明します．両者のちょっとした違いを発見しながら読み進めると面白いでしょう．

## 4.1 ハードウェア記述言語の基本概念

プログラミング言語に多くの種類があるように，ハードウェア記述言語（HDL）にもさまざまな種類があります．その中でもよく利用されているのが，VHDL[注1]とVerilog HDL[注2]です（**図4.1**）．

VHDLとVerilog HDLは，どちらもハードウェアを表現する似たような概念を取り扱うことができる言語です．ただし，似たような概念であっても，それぞれの言語で使用する言葉が違うので注意が必要です．

そこで，両方の言語に共通する概念と，言語の特徴について説明します．

### ● 構造の基本 ―― エンティティ / モジュール

どの言語にも基本的な構造があります．例えば，C言語では関数，Javaではクラスなどです．HDLでは，与えられた入力に対して出力を生成するブロックが基本的な単位となります（**図4.2**）．このブロックをVHDLではエンティティ，Verilog HDLではモジュールと呼びます．

通常のプログラミング言語とHDLの大きな違いは，エンティティ / モジュールは，最初から最後まで与えられた入力に対する出力を生成し続けるということです．C言語などで関数を呼び出す場合は，`main`プログラムからその関数内へ処理

注1：VHDL（VHSIC Hardware Description Language）は，1970年代後半から1980年代前半にかけて，米国国防総省がVHSIC（Very High Speed Integrated Circuit）プロジェクトの一環として，部品の納入業者に対しASICの仕様書を統一させることを目的として開発された．言語仕様は，Adaの影響を受けている．1987年にIEEEの標準規格となり，現在の最新版はIEEE1076-2008である．

注2：Verilog HDLは，1984年に米国Gateway Design Automation社のPhil Moorbyによって，ディジタル回路を設計するための言語およびシミュレータとして開発された．言語仕様は，C言語やPascalの影響を受けている．1995年にIEEEの標準規格となり，現在の最新版はIEEE1364-2005である．

第5章 ハードウェア・プログラミング的な「Hello World!」で始める

# Xilinx&Altera の FPGA 開発ツールの使い方

本章では，実際に FPGA を使って「モノ」を作る手順を習得しましょう．パソコン上での HDL 記述によるバーチャルなハードウェア設計と現実世界を橋渡しする方法を説明していきます．いよいよ FPGA を使った論理回路設計の開発へ一歩を踏み出しましょう．

## 5.1 FPGA 評価ボードを用意しよう

HDL で記述したコードを実際に「モノ」として動かすためには，HDL のソース・コードを合成して FPGA に書き込むのですが，それだけでは不十分です．FPGA を動作させるには，電源やクロックを供給する必要があります．パソコンでいうところのマザー・ボードに相当するものが必要です．

そこで本章では，FPGA の 2 大メーカの定番 FPGA ボードを使用して，実際の開発手順を説明します．使用するボードは Xilinx 社の FPGA を搭載した Avnet 社の MicroBoard（**写真 5.1**）と，Altera 社の FPGA を搭載した Terasic 社の DE0-Nano（**写真 5.2**）です（入手方法は，**Appendix 3** を参照）．**第 4 章**で HDL ソース・コードの文法について説明しましたが，それに基づいて，FPGA 開発の雰囲気を味わってみましょう．

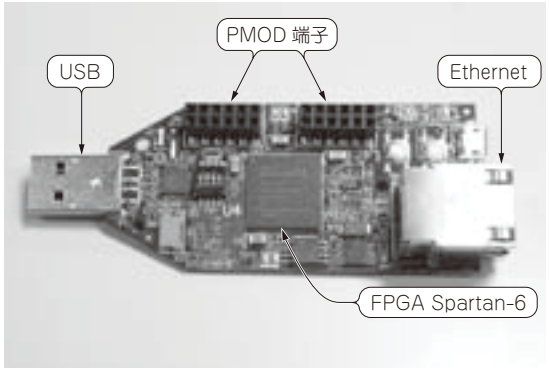



**写真 5.1　Xilinx 社の FPGA Spartan6 を搭載した Avnet 社の MicroBoard**

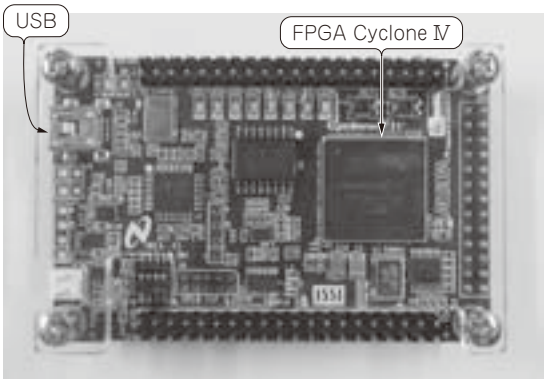



**写真 5.2　Altera 社の FPGA Cyclone IV を搭載した Terasic 社の DE0-Nano**

第6章 ハードウェア設計を意識した回路シミュレーションを体験しよう！

# シミュレータ ISim, ModelSim の使い方

本章では，Windows 上で動作するシミュレータを使って，HDL で設計したモジュールの動作を確認する方法を修得しましょう．シミュレータを利用すると，FPGA 本体や FPGA を動作させるための周辺回路（電源やクロック回路など）のハードウェアを用意する必要がないので，ハードウェアを用意することに抵抗を感じる人でも気軽に試すことができます．

## 6.1 FPGA がなくても HDL モジュールの動作を確認できる

**第 5 章**では，開発ターゲットとして MicroBoard と DE0-Nano の FPGA 評価ボードを使用し，FPGA の開発を始める手順を紹介しました．とはいえ，実際には手元に FPGA のハードウェア環境を用意していない読者も多いことでしょう．ここでは，手軽に HDL でハードウェア設計の雰囲気を試せるシミュレータの使い方を紹介します．

**第 2 章**と**第 3 章**では，プログラミング言語としての HDL を紹介する目的で，ハードウェアの概念をほとんど意識することのない例を紹介しました．逆に本章では，いかに HDL で「ハードウェアの設計」をパソコン上で試すことができるか，に焦点を当てます．

### ● 無償のシミュレータを使う

Xilinx 社の FPGA ユーザは ISE WebPACK に付属している「ISim」を，Altera 社の FPGA ユーザは「ModelSim」を無償で利用できます．基本的な HDL コードの動作をシミュレーションするだけなら，どちらのシミュレータでも大差ありません．ただし，それぞれのデバイスに特化した機能を使ったハードウェアのシミュレーションをするためには，使用するデバイスをサポートするシミュレータを選択する必要があります．

ISim は，Xilinx 社の ISE WebPACK をインストールするときに一緒にインストールされます．ModelSim は，Altera 社の開発環境 Quartus Ⅱ のインストール時に選択するか，Altera 社の Web サイトから別途ダウンロードしてインストールします．

インストールの手順は，一般的な Windows アプリケーションと同様にインストーラの指示に従うだけです．詳しいインストール方法は，**Appendix 2** を参照してください．本章では，ISim と ModelSim で HDL コードをシミュレーションする手順を紹介します．

### ● ISim や ModelSim でできること

**第 2 章**で紹介した Icarus Verilog では，コマンドラインから HDL ソース・コードをコンパイル実行して，普通のプログラミング言語処理系のように使うことができました．

第7章 カウンタ，算術演算，マルチプレクサ，7セグメントLED，乱数生成器，シリアル送受信モジュール

# コピペできる！ VHDL/Verilog HDLの基本プログラム集

新しいプログラミング言語を習得するには，さまざまなサンプル・プログラムを見て実際に動かしたり，コピー&ペーストで自分のプログラムに移植してみたりするのが早道でしょう．それはHDLの場合も同じです．

本章では，HDLコードで記述する基本的なハードウェア・モジュールとして，カウンタ，ビットの数え上げ演算器，7セグメントLEDの点灯制御，乱数生成器，シリアル送受信モジュールをVHDLとVerilog HDLの両方で記述する例を紹介します．

どの記述例も，**第6章**で説明した手順でシミュレーションができます．是非，手元で動作を確認してみてください．

## 7.1 基本的なハードウェア① ―― カウンタ

カウンタは，基本的なハードウェア・モジュールの一つです．入力された信号の変化に応じて内部の変数をインクリメント(1ずつ値を増加)します．例えば，クロックを入力信号として立ち上がりに応じて内部の変数をカウントすることで処理のタイミングを遅らせるためのウェイトを作ったり，定期的に特定の処理を行うためのフラグにしたりといった使い方が考えられます．

VHDLとVerilog HDLで記述したカウンタのソース・コードを，**リスト7.1**に示します．

### ● モジュールの外枠と必要な変数の定義

カウンタ・モジュールの外枠は，カウントの対象となる入力信号(例えば，クロック信号)pCLKと，カウンタを初期値にセットする信号pReset，そしてカウンタ値の出力信号Qで定義できます．

カウントした値は，VHDLではsignal，Verilog HDLではreg型の変数として定義されているcounterです．カウンタの幅は，パラメータで指定できるようになっています．

### ● カウンタの処理内容

リセット信号が入力されたらカウンタ値をクリアし，それ以外のときはクロックが立ち上がるタイミングで，counter変数の値を一つずつ増加させます．この処理を，if文を使用して記述します．

### ● シミュレーション結果

この動作を**リスト7.2**に示すテストベンチを使ってシミュレーションしてみます．シミュレーションすると，**図7.1**に示すような波形が表示されます．矢印の区間がちょうどカウンタの1周期に相当します．クロック信号が立ち上がるたびに，モジュールの出力がカウントアップされていることを確認できます．

第8章 ステート・マシン，モジュール化，階層設計，IP コアの活用で効率的な設計を見指す

# 大規模ハードウェア・プログラミングに挑戦する

ソフトウェア・プログラミングでは，既存の資産の有効活用や再利用性を高めることが求められます．また，よくあるパターンを定型化することにより，きれいな記述を目指します．これらの取り組みにより，ソフトウェアを記述する時間やデバッグ時間を削減できます．

同じように，HDL によるハードウェア・プログラミングでも，データ構造の抽象化やモジュールの再利用などにより，記述やデバッグの負荷を削減できます．

そこで本章では，決められた手順で処理を実行するモジュールを設計するときに有効となるステート・マシンや，すでに実装したモジュールをサブモジュールとして利用する方法，そして既存の IP コアを活用して設計を楽にする手法などについて解説します．

## 8.1 ステート・マシンを実装する

まず，ハードウェア・プログラミングにおいて，逐次的に処理するために必要不可欠な概念であるステート・マシンと，その実装方法について説明します．

ハードウェアでは処理を並列に実行できますが，世の中には順番にしか実行できない物事もたくさんあります．例えば，そうめんを茹でるとき，鍋の水がまだ湯になる前に並行してそうめんを鍋に入れても，とても食べられるものにはなりません．

### ● 料理は逐次的な処理

何かを実行し，その次に何かを実行し，その次

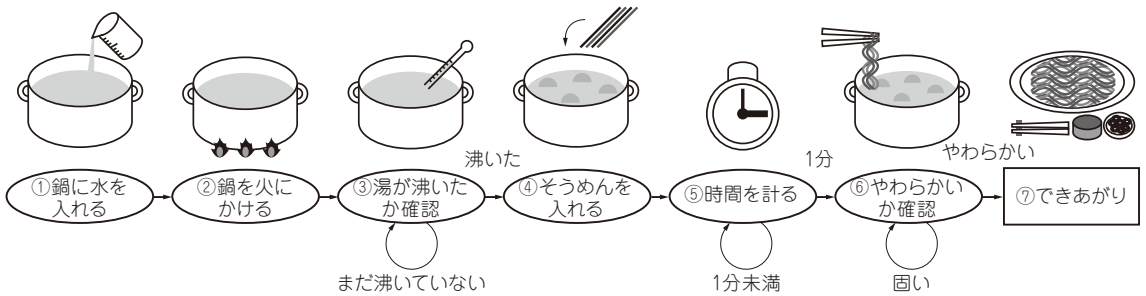



図 8.1 そうめんを茹でる手順をステート・マシンで表すと

第 9 章 製作事例で学ぶ①
小さなモジュールを組み合わせて並列処理する方法

# オセロ・ゲームを FPGA に実装しよう

HDL を用いたハードウェアの開発では，ソフトウェアの開発と同じように小さな回路モジュールを組み合わせることにより，大規模な回路を作成していきます．

そこで本章では，オセロ・ゲームの作成を通じて，FPGA を使用したシステムを開発する過程を解説します．

また，この製作を通して，FPGA の特徴といえる並列処理により，処理を高速化できることを体験してみてください．

## 9.1 今回作成するオセロ・ゲームの仕様

オセロ[注1]は，2 人のプレーヤが白と黒の石を交互に打ち，ゲーム終了時の石の数を競うゲームです．

ここでは，FPGA とパソコンによりシステムを構成します（**図 9.1**）．FPGA には，ゲーム全般の処理（ゲーム盤の表示や石が挟まれたときの処理など），対戦するコンピュータ側の処理（乱数を用いて次の手を決定する処理），パソコンと通信する処理を実装します．

パソコンとの通信には，**第 7 章**で紹介したシリアル通信を利用することにします．例えば，Windows の TeraTerm などのターミナル・ソフトを使用して，パソコンの画面にゲームを表示させて遊ぶことができます．最近は，シリアル・ポートのないパソコンが多くなりましたが，USB で接続するシリアル通信用アダプタや IC もありますから，回路の実用性は十分です．

本章では，Xilinx 社の FPGA Spartan-6（XC6SLX9-2CSG324C）を搭載した FPGA ボード MicroBoard と Altera 社の FPGA Cyclone IV

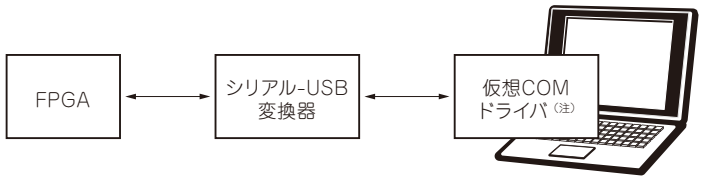



**図 9.1　FPGA とパソコンでオセロ・ゲームを構成する**

注 1：オセロはメガハウスの登録商標である．世界的にはリバーシと呼ばれている．

第10章 製作事例で学ぶ②
画像処理を専用ハードウェア化する方法

# 画像のグレースケール処理をFPGAで動かしてみよう

**第3章**では，Verilog HDLを普通のプログラミング言語として使うことにより，画像のグレースケール処理を記述しました．本章では，同じグレースケール処理を題材にして，実際にFPGAに実装可能なハードウェアとして設計し直してみましょう．

## 10.1 ソフトウェア実装とハードウェア実装の違い

**第3章**では，Verilog HDLという言語に触れてもらうことを主眼にしたため，実際にFPGA上に回路を合成し，ハードウェアとして動作させるために考慮すべき事由が欠けていました．そのため，設計を変更する必要があります．パソコン上のシミュレーションではなく，実際のFPGA上で動作させるには，

(1) 標準入力や標準出力といった手軽なパソコンのI/Oが使えない．
(2) データの最後の判別にEOFのような便利なマークは使えない．
(3) FPGAの持つハードウェア・リソースに収まるような回路にしなければならない．

という課題があります．

(1)は，標準入出力の代わりに**第9章**でも利用したシリアル通信を使うことで解決できそうです．(2)は，ファイル・サイズを何らかの方法でハードウェアに伝えることで解決できます．今回は，ビットマップ・ファイルのフィールドにある，格納ファイル・サイズを使うことにします（詳細は後述）．

問題は(3)です．何も考えないとFPGAのリソースをあっという間に喰いつぶしてしまいます．論理回路や信号遅延を考慮しなければならないという点については，これまでの章でも触れてきましたが，今回考えるハードウェアではさらにメモリの使用量についても考える必要があります．

### ● メモリのサイズは有限 !!

**第3章**の例では，ビットマップ・ファイルを扱うために，

```
reg [7:0] data[2**20-1:0];
// サイズ2^20の8ビット幅のデータ配列
```

と，あまり深く考えずに1Mバイトのメモリを確保しています．しかし，MicroBoardに搭載されているSpartan-6 XC6SLX9やDE0-Nanoに搭載されているCyclone IV EP4CE22F17C6N上では，そんなに多くのメモリを使うことができません．

メモリとして利用できるブロックRAMのサイズは，それぞれ64Kバイト程度です．1Mバイトの容量を確保することは到底できません．

したがって，

(1) より大きなFPGAを利用する．
(2) 扱う画像サイズを小さくする．

Interface増刊 インターフェースZERO No.04

CQ出版社 〒170-8461 東京都豊島区巣鴨1-14-2 TEL.03-5395-2141（営業部）

定価2,310円 本体2,200円

雑誌　01620-1
Ⓛ 2013.2/10