

# プロジェクト実習Ⅲ 論理設計

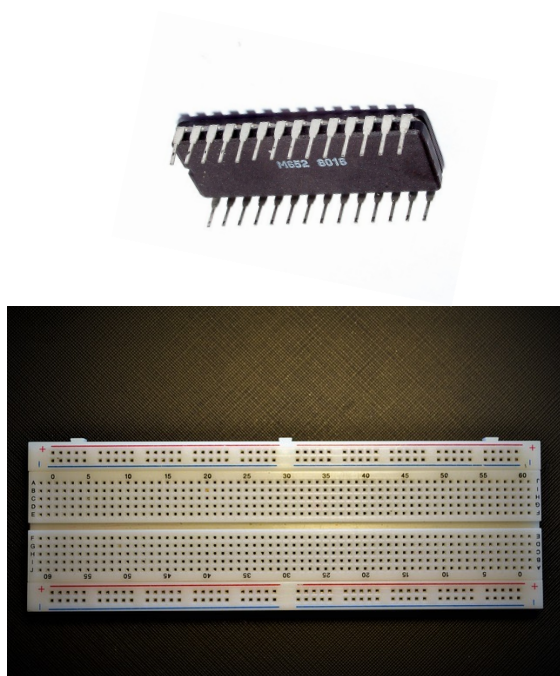
第1週「論理設計の基礎」

# 実習の目的

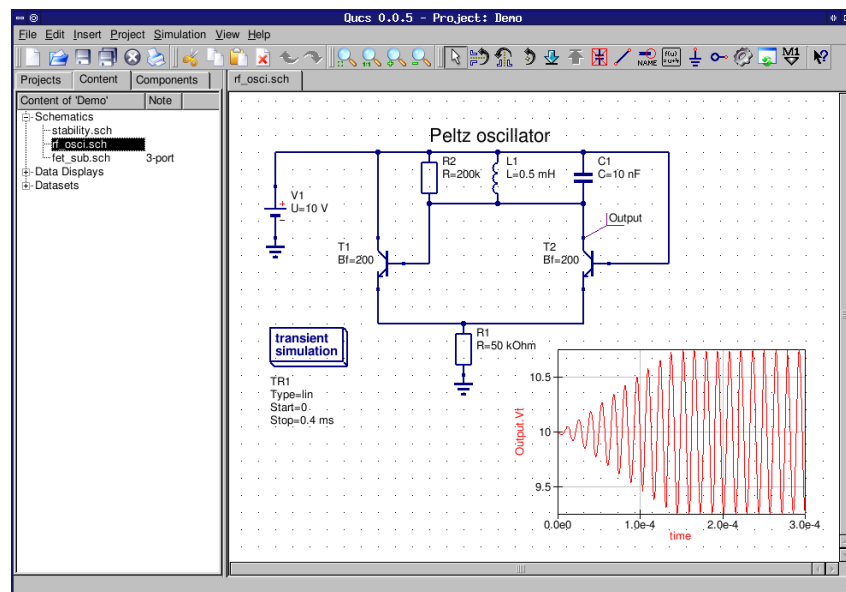
- ハードウェア記述言語（HDL）を用いて論理回路を設計する手法を学ぶ
- 最終的には（簡単な）プロセッサを設計する

# かつての論理設計（の実験）

- ブレッドボードに実体を配置・配線
- PC上のCADツール上で作成・シミュレーション



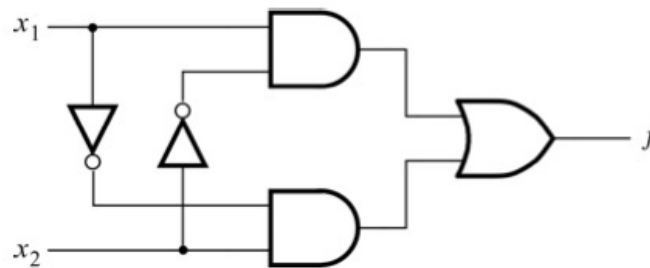
ブレッドボード




QUCS (Quite Universal Circuit Simulator)

# この実習で行う論理設計

- HDL + FPGA
- PC上で設計・シミュレーションは完結するが、  
実機(FPGA)上でも動作確認する



$x_1$	$x_2$	$f$
0	0	0
0	1	1
1	0	1
1	1	0



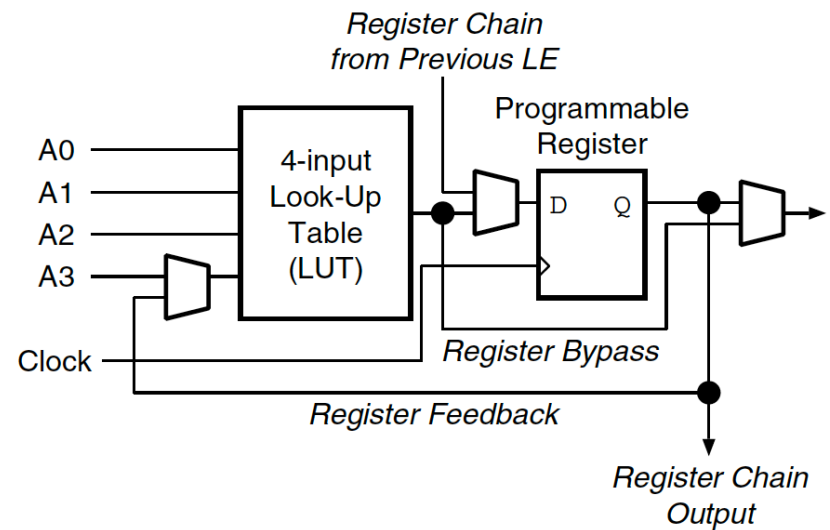
```
module light (x1, x2, f);  
  input x1, x2;  
  output f;  
  assign f = (x1 & ~x2) | (~x1 & x2);  
endmodule
```

# HDLによる論理設計

- HDL（ハードウェア記述言語）
- 設計者がテキストファイル（HDLファイル）を記述
- 既存のライブラリと合わせて**論理合成ツール**がゲート回路に変換
- より高いレベルで回路設計が可能に
- 本実習ではVerilog HDLを使用する
  - （良くも悪くも）C言語とよく似ている記述

# FPGA

- 「現場(Field)でプログラム可能  
(Programmable)なゲート  
アレイ (Gate Array)」  
(現場で書き換えられ  
るハードウェア)
- 入力に応じた出力値を  
表(LUT)を参照して決定
- LUTの内容は変更可能



ロジックエレメント(LE)の構成

# LUTエントリの構成例

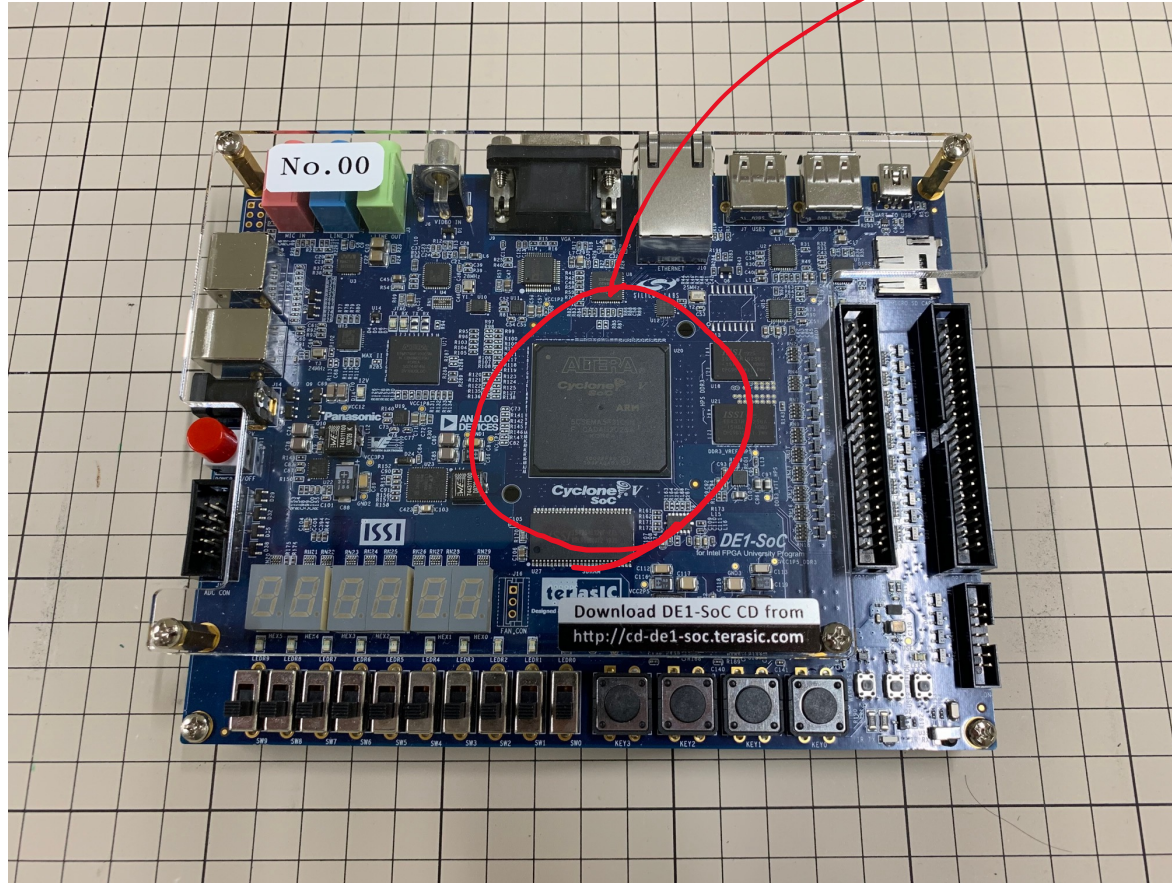
A0	A1	A2	A3	Output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

論理式  $A0 \cdot A1 + A2 \cdot A3$

- すべての入力パターンに対応した出力値をテーブル(LUT)に格納する
- 基本論理ゲートとLEが1対1に対応するわけではない
- LUTの内容は、Verilog HDLでの記述から論理合成ツールが生成してくれる

# FPGAボード DE1-SoC

FPGAデバイス  
(Cyclone V)



約8万5千個のLEを搭載している



# 「CPU」実験との違い

- 「CPU」実験
  - 教育用CPUの命令でプログラム作成
  - ソフトウェアをCPUボードに入力して実行させる
- 「論理設計」実験
  - HDLでハードウェア構成を記述
  - FPGAの構成を変更（ハードウェアを再構成）して動作させる

プログラムっぽいものを記述してボード上で動かすのは似ていても、行っていることの意味はまったく異なることに注意！

# 今回の実習項目

- 実習1：4-to-1マルチプレクサ
- 実習2：D-FF
- 実習3：有限状態機械
- 実習4：カウンタ

Verilog HDLで記述し，シミュレーション（実習1, 2）または実機（実習3, 4）で確認

演習問題1, 2はレポートで解答すること

# 実習の進め方とレポートの書き方

- 「実習○」の直前部分だけ読んでも不十分
- テキスト全体に目を通すこと
- 実習課題「動作を確認せよ」

→レポート「動作することを確認した」ではダメ．第三者に「何を根拠に確認したか」が伝わるように書く


- 何を入力に与え、こういった出力が得られたか
- 何を調べる目的でその入力を選んだか
- 得られた結果から何が言えるのか

# Verilog HDLの文法

- テキスト付録A
- 「信号(wire)」と「変数(reg)」の違い
  - wire：部品間の配線
  - reg：データの記憶素子
- 代入方法の使い分けに注意
  - wireへの代入 → assign文
  - regへの代入 → ブロッキング or ノンブロッキング代入
- ハードウェアなので基本的に並列に動作する

# 論理合成ツールの使い方

- テキスト付録B
- Quartus Prime Lite Edition 20.1.1
  - Windows/Linux対応
  - 自宅環境で使いたい場合は、Moodleの「自習環境について」を参照のこと
  - 実機のFPGAデバイス(Cyclone V)は、バージョン14.0以降でサポート
  - バージョン21.1以降はシミュレータが変更 (ModelSim→Questa) され、使用時にライセンスファイル (別途登録必要, 無料) が必要になった



推奨

# シミュレータの使い方

- テキスト付録C
- ModelSim (Intel FPGA **Starter** Edition)を使う
  - インストール時に選択しておく
  - 「ModelSim - Intel FPGA Edition」の方をインストールしないこと（こちらは有償版）
- シミュレーション開始前に詳細設定を確認すること（テキスト手順8）
- テキスト手順9では，functional simulation（機能シミュレーション）を選択する

# 【おまけ】 iverilog+gtkwave

- テキスト付録D
- フリーのシミュレーション環境
  - iverilog→コマンドラインのシミュレーションツール
  - gtkwave→信号波形の表示ツール(GUI)
- Windows/macOS/Linux/FreeBSDなどに幅広く対応
- Quartusがインストールできない環境で使用する