

実験3 HDLによるハードウェア設計

〔目的〕

ハードウェア記述言語 (HDL: Hardware Description Language) を使用したハードウェア (デジタル論理回路) の設計手法を学習する。また、学習用 FPGA ボードと HDL シミュレータによる回路の動作検証を通して、実際にハードウェアがどのように動作するのか理解する。

〔概要〕

HDL の一つとして広く利用されている **Verilog-HDL** を使用して、デジタル論理回路、すなわち組合せ回路や順序回路を記述する。記述した回路が正しく動作することを、学習用 FPGA ボードおよび Verilog-HDL シミュレータを用いて確認する。

HDL によるデジタル論理回路の設計は、一般的には図 1 に示す設計フロー¹で行われる。

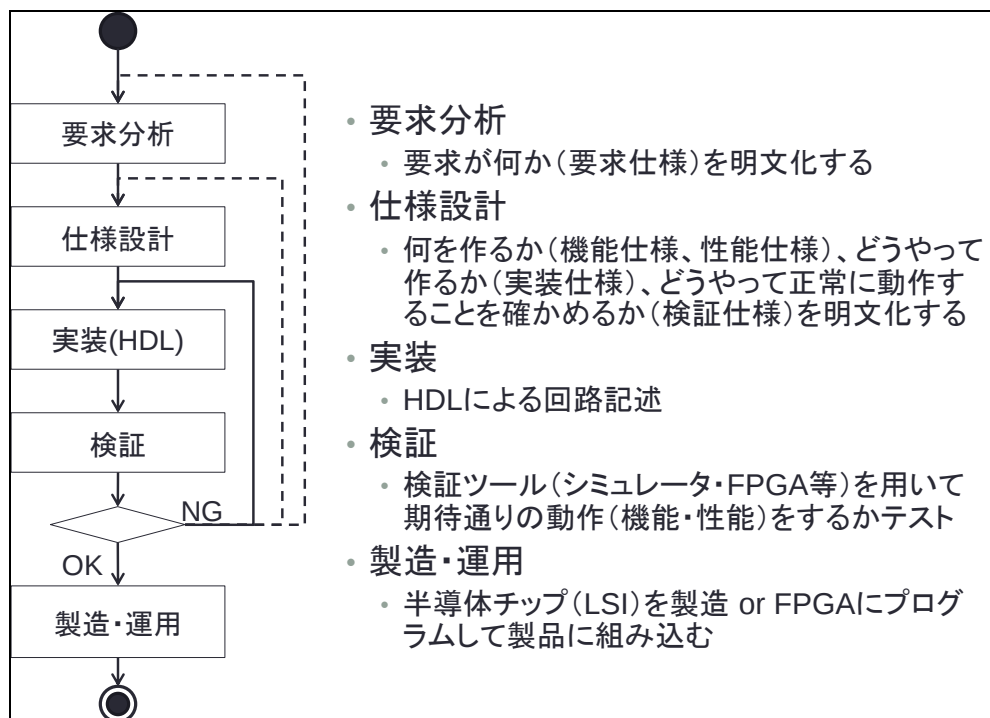


図 1 HDL によるデジタル論理回路設計フローの例 (単純化したもの)

本実験では、仕様に基づいて HDL による記述を行い (実装)、期待通りの動作をすることをシミュレータ・FPGA²を用いてテストする (検証) という工程を、受講生が各自行う。一連の設計フローを経験し、それをレポートという形に明文化することを通じて、HDL によるハードウェア設計の実際を理解する。

¹ この設計フローは、単純化して図示したものであり、実際の設計においては、実装と検証は様々なレベルで繰り返し行われる。また点線のフローは生じないことが望ましい。

² FPGA (Field Programmable Gate Array) は、開発者が任意のデジタル論理回路を、ソフトウェアの様にプログラムして実現することが可能な、半導体 LSI チップである。

[使用する教材]

- ・実験指導書（この資料）
- ・副教材：キットで学ぶ FPGA チャレンジャー Xilinx Artix-7 版

<副教材内容>

- ・テキスト冊子
- ・学習用 FPGA ボード Basys3 ボード
 - ・FPGA： Xilinx 社製 Artix-7 XC7A35T
 - ・ユーザ I/O デバイス：
LED 緑 16 個、スライドスイッチ 16 個、プッシュボタン 5 個、リセットボタン
7 セグメント LED 4 ケタ、VGA ポート、I/O ピン 6pin(Pmod) x 4 個、他

[実験内容]

以下の 3 項目について実験を行い、HDL で記述し合成した回路が期待通りに動作することを、FPGA ボードとシミュレータを用いて確認する。

1. 組合せ回路を FPGA ボードで動かす
2. 順序回路を FPGA ボードで動かす・シミュレーションで何が起きているか確認する
3. 合成した回路の規模を知る・課題回路を作成して動作を検証する

1. 組合せ回路を FPGA ボードで動かす

組合せ回路は、AND/OR/NOT から構成される論理関数に相当するディジタル論理回路である。すなわち、出力はその時の入力によってのみ決まる。入力の組合せで出力が決まるので、組合せ回路という。

本項目においては FPGA 上に組合せ回路を実現して期待通りに動作することを確認する。

手順

- 1-1. STEP01 を読み、学習用 FPGA ボードと FPGA について理解する。
- 1-2. STEP02 を読み、P.22 の Basys3 ボード動作チェックを行う。
- 1-3. STEP03 に従い、Vivado のプロジェクトを作成する。プロジェクトの場所は、各自のマインドキュメント内とする。最も単純な Verilog 記述を作成する。(P.32 ON_Circuit.v)
- 1-4. STEP04 に従い、合成・デバイスの制約・ビットストリーム生成（コンパイル）・デバイスへ書き込み（JTAG プログラミング）、を行うことで、FPGA ボードに作成した Verilog 記述に相当する回路をプログラムし動作を確認する。
- 1-5. STEP05 に従い、**課題 05（AND 回路）**を作成し動作を確認する。
- 1-6. STEP06 に従い、**課題 06（OR 回路）**を作成し動作を確認する。
- 1-7. レポート課題 1「組合せ回路の実装・テスト」を行う。
 - 仕様（何を作ったか）、実装（どうやって作ったか）、およびテスト結果についてレポートに記載すること。
- 1-8. （オプション）STEP07 に従い、セクタ回路を作成し動作を確認する。

2. 順序回路を FPGA ボードで動かす・シミュレーションで何が起きているか確認する

順序回路は、メモリ要素（フリップフロップなど）を含むデジタル論理回路である。すなわち、出力はその時の入力と、以前の状態から決まる。入力の順序により出力が決まるので、順序回路という。本項目においては、FPGA 上に順序回路を実現して、期待通りに動作することを確認する。また、回路の動作は非常に高速（100MHz で動作）であるため、シミュレーションにより動作の詳細な様子を観察して理解する。

手順

- 2-1. STEP08 に従い、**課題 08-1（非同期リセット）** および **課題 08-2（同期リセット）** を実施する。フリップフロップ(DFF)の動作を FPGA ボードにおいて確認する。
- 2-2. STEP09 に従い、**課題 08-1** と **課題 08-2** のフリップフロップ(DFF)の動作をシミュレーション(Vivado のシミュレータ)で確認する。
→ 非同期リセットと同期リセットの違いを理解する。(班内でお互いに説明せよ)
- 2-3. STEP10 に従い、**課題 10-1（カウンタ）**、**課題 10-2（100MHz クロック分周して 1Hz を作成）** および **課題 10-3（1 秒ごとにカウントアップして LED 表示）** を実施する。
各回路の動作を確認する（FPGA ボード、および Vivado のシミュレータ）。
- 2-4. レポート課題 2 「順序回路の実装・テスト」を行う。
→ 仕様（何を作ったか）、実装（どうやって作ったか）、およびテスト結果についてレポートに記載すること。
- 2-5. (オプション)STEP11 に従い、スイッチ入力を数えるカウンタ回路の動作を確認する。

3. 作成した回路の規模を知る・課題回路を作成して動作を検証する

本項目においては、これまで作成した回路の規模を調べる。また、課題回路を作成する。

FPGA は任意の回路をプログラムできる半導体 LSI チップであるが、内部にはメモリ要素である Flip-Flop と、論理関数を実現するための真理値表に相当する LUT(Look Up Table)が数多く存在しており、それらの間をプログラマブルな配線スイッチで接続することで任意の回路を実現する。以上を踏まえて、FPGA 上に作成した回路の規模を調べる。

手順

- 3-1. 組合せ回路（レポート課題 1 など）および順序回路（レポート課題 2、カウンタなど）の、フリップフロップ使用量、LUT 使用量、乗算器使用量を調べる。(レポート中には、これらに関する調査結果を記載すること。)
- 3-2. STEP12 を読み、**課題 12-1：7 セグ LED デコーダ回路**の動作を理解する。
(P.128 課題 12-2 の、7 セグ LED カウンタ回路はオプションとする)
- 3-3. 任意の 4 ケタ数字を 7 セグメント LED に表示する回路を、講義資料 WEB サイトよりダウンロードし、FPGA ボードにおいて動作を確認する。
- 3-4. レポート課題 3 「すこし複雑なデジタルシステムの実装・テスト」を行う。

[課題]

ソースコードなどの参考資料は、講義資料 WEB サイト内の以下の URI を参照すること。

<http://www.ced.is.utsunomiya-u.ac.jp/lecture/2017/jikken2/hdl/>

レポート課題 1 : 組合せ回路の実装・テスト

積和演算回路を Verilog-HDL で実装・テストする。

◆ 機能仕様

入力： FPGA ボード上のスライドスイッチ(SW)の、SW15~8 を入力 A[7:0]、SW7~0 を入力 B[7:0]とする。

出力： A と B を乗算し、本年度の数値 (10 進数の 2017) を加算した値を S (Sum の略)とし、FPGA ボード上の LED に S[15:0]の値を出力する。

◆ 実装仕様

・ファイル名は MulAdd.v とする。

・加算器・乗算器は Verilog-HDL の加算演算子・乗算演算子を使って記述する。

◆ 検証仕様

・FPGA ボードを用いて検証を行う。以下の入力値の組合せに対する出力値を記録し、期待値と一致するかを確認する。結果は OK か NG かで示す。

テスト 番号	入力		出力期待値		出力値	結果
	A[7:0]	B[7:0]	S[15:0]		S[15:0]	OK/NG
1	0	0	17	16'h11		
2	1	0	17	16'h11		
3	0	1	17	16'h11		
4	1	1	18	16'h12		
5	2	3	23	16'h17		
6	100	100				
7	250	250				
8	255	255				

・自分の学籍番号をテスト入力値に設定し、出力値(LED 状態)を、カメラで撮影してレポートに画像として貼る。撮影の際、加算器への入力値は、自分の学籍番号の下二桁の数字を入力 A、入力 B を 100 とし、LED の出力が期待値と同じになることを示す。

例) 学籍番号 : 152985 入力 A = 85, 入力 B =100 出力期待値 S = $85 \times 100 + 2017 = 10517 = 16'h2915$

・フリップフロップ・LUT・乗算器使用量を調べる事。(本資料 P.8 補足 2 を参照)

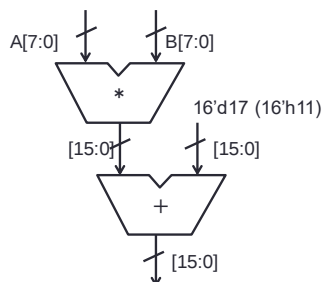


図 2 積和演算器のブロック図

レポート課題 2 : 順序回路の実装・テスト

自分の学籍番号の数字を、順番に 1 秒間に 1 桁ずつ LED に表示する順序回路を Verilog-HDL にて実装・テストする。

◆ 機能仕様

入力： プッシュボタン BTNC(start)を押すことで学籍番号の表示を開始する合図とする。プッシュボタン BTND(reset)を押すと初期状態に戻ることとする。

出力： FPGA ボード上の 4 つの LED に自分の学籍番号を順番に 1 秒間に 1 桁ずつ、4 ビットの 2 進数として表示する。1 秒間の時間は、FPGA ボードの 100MHz クロック信号を 100M 回カウントすることで計測する。

◆ 実装仕様

- ・ファイル名は NumberDisplay.v とする。
- ・以下の状態遷移図および状態遷移表に従った、ステートマシン³を作成する。

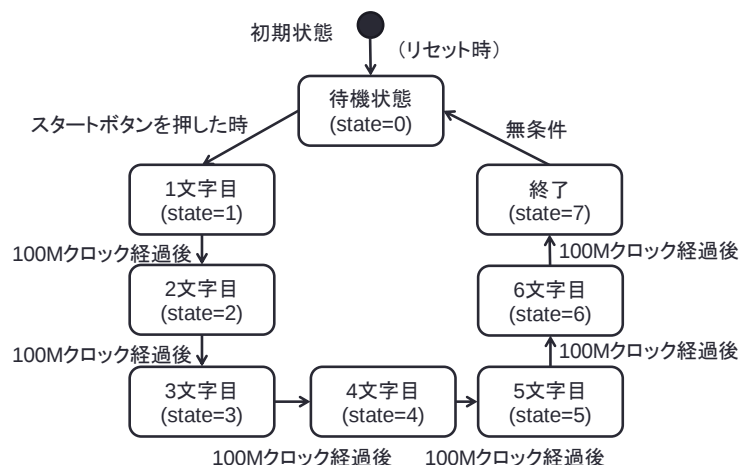


図 3 課題 2 の状態遷移図

表 1 課題 2 の状態遷移表 (学籍番号 : 152985 の場合)

状態 (state)	入力		次状態 (state')	出力	備考
	BTNC	BTND		LED[3:0]	
X	1	X	0	X	-
0	0	0	0	0	-
	0	1	1	0	-
1	0	X	2	1	100M クロック経過後
2	0	X	3	5	100M クロック経過後
3	0	X	4	2	100M クロック経過後
4	0	X	5	9	100M クロック経過後
5	0	X	6	8	100M クロック経過後
6	0	X	7	5	100M クロック経過後
7	0	X	0	0	-

表中の X は、任意の値を示す (Don't care という)。すなわち 1 行目は任意の状態において BTNC の値が 1 だったら、BTND の値に関わらず状態 0 に遷移することを示す。

³ カウンタを用いることで、任意の状態遷移を持つ順序回路を構成することができる。回路の制御部となる順序回路をステートマシンと呼ぶ。

◆ 検証仕様

- ・シミュレータを用いて検証を行う。実時間（数秒間＝数千万クロック）のシミュレーション結果を確認するのは時間がかかるので、シミュレータを用いた検証の際は、2 クロックに 1 桁ずつ LED への出力信号を変化させることとする。
- ・以下の入力値の組合せに対する出力値を記録し、期待値と一致するかを確認する。結果は OK か NG かで示す。

クロック時刻	入力		出力期待値	出力値	結果
	BTN0 (reset)	BTN3 (start)	LED[3:0]	LED[3:0]	OK/NG
0	1	0	0		
1	0	0	0		
2		1	0		
3		0	1		
4			1		
5			5		
6			5		
7			2		
8			2		
9			9		
10			9		
11			8		
12			8		
13			5		
14			5		
15			0		
16			0		

- ・信号の値は、シミュレーション波形の画面キャプチャを行うか、シミュレータのテキスト出力を、レポートに貼り付ける事。（本資料 P.8 補足 1 を参照）
- ・フリップフロップ・LUT 使用量を調べる事。

レポート課題 3 : 少し複雑なデジタルシステムの実装・テスト

以下の 3 つの課題のうち 1 つ以上を選択して、機能仕様・実装仕様・検証仕様を作成し（すなわち設計を行い）、HDL による実装と FPGA ボードもしくはシミュレーションによる検証を行う。

課題 3-1 4 ビット算術論理演算装置(ALU)

課題 3-2 8 ビット簡易電卓（加算もしくは乗算）

課題 3-3 スロットマシンゲーム

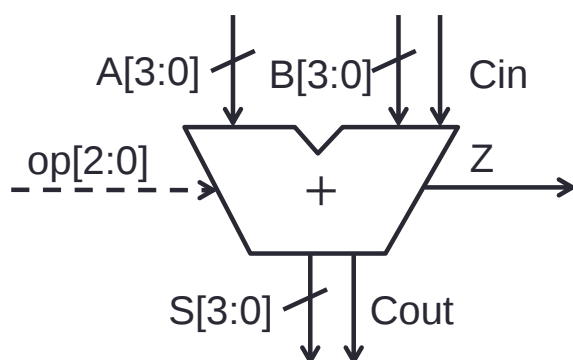
検証結果として、以下の 3 つを含める事。

- ・シミュレーション時の入力と出力の値をテキストで出力、もしくは波形の画面写真
- ・フリップフロップ・LUT・乗算器使用量
- ・動作中の FPGA ボードの写真

課題 3-1 4 ビット算術論理演算装置 (ALU)

図 4に示す 4 ビット算術論理演算装置 (ALU) を設計・実装・テストする。

【必須】この ALU は、4 ビットデータ A、B、及びキャリーCin を入力とし、op で指定される演算を行い、演算結果 S、キャリーCout を出力する。また、S がゼロの時、Z に 1 を出力する。入力は、スライドスイッチを 8 つ使って A[3:0]、B[3:0]を設定し、4 ビットのプッシュスイッチで op[2:0]および Cin を設定する。出力は、S[3:0]を 1 ビットずつ”0”か”1”を 7 セグメント LED の 1 桁ごとに表示し、Cout と Z を LED に表示する。



op	演 算 内 容	Z	
0		*	
1		*	
2		*	
3		*	
4		*	*
5		*	*
6	←	*	*
7	←	*	*

演算結果は * 印の付いた出力に反映

図 4 4 ビット算術論理演算装置

課題 3-2 8 ビット簡易電卓 (加算もしくは乗算)

8 ビットの数字を加算もしくは乗算可能な、簡易電卓を設計・実装・テストする。

【必須】計算結果は、16 進数の数字として 7 セグメント LED に表示する。ユーザは、値を設定するときはスライドスイッチ(8 ビット)を設定する。操作方法は以下の通りである。

- ・ BTNL を押すと値 1 として記憶し、7 セグメント LED に表示する。
- ・ BTNC を押すと値 2 として記憶し、7 セグメント LED に表示する。
- ・ BTNR を押すと、2 つの値の和を 7 セグメント LED に表示する。

【オプション】

- ・ BTND を押すと、2 つの値の積を 7 セグメント LED に表示する。

課題 3-3 スロットマシンゲーム

以下のゲームを設計・実装・テストする

【必須】7 セグメント LED に数字 3 ケタを表示し、時間と共に高速に変化する。ユーザがプッシュボタンを押すと、数字が停止する。

【オプション】システムは各桁の数字が全て一致している場合、「当たり」と判定して、ユーザに示す。(例：LED を賑やかに変化させる。)

[補足資料]

補足 1. Verilog-HDL シミュレーションにおいて信号値を出力する方法

レポート作成時、動作の様子を示すためには、画面写真をキャプチャして貼りつけるか、信号値の変化をテキストとして貼りつける必要がある。シミュレーションの際に、テストベンチの Verilog のモジュール内に図 5 に示す記述を加えると、信号値の変化をテキストで出力することが出来る。(%b: 2 進数、 %d: 10 進数、 %o: 8 進数、 %h: 16 進数)

```
initial
    $monitor($stime, "in0=%b in1=%b out=%b", in0, in1, out);
```

図 5 シミュレーションにおける信号値を出力するための記述例

補足 2. FPGA のコンパイル結果・リソース使用量・性能を知る方法

FPGA のコンパイル（合成・配置配線）を行った後、Vivado の Design Summary を見ること、リソース使用量や性能を知ることが出来る。（画面右の Project Summary ウィンドウ内で、下にスクロールするとグラフが現れる）

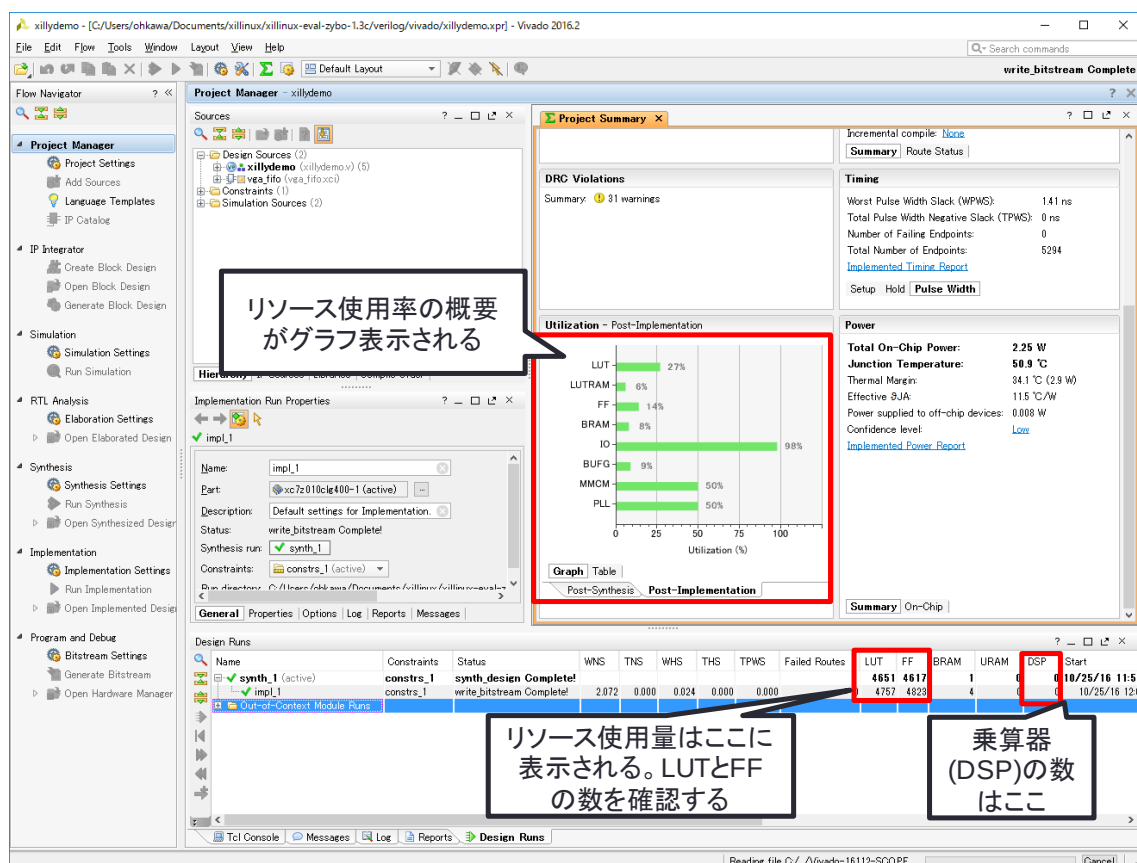


図 6 Vivado の Project Summary の読み方

[参考文献]

- [1] 小林優, “入門 Verilog-HDL 記述”, CQ 出版.
- [2] 小林優, “初めてでも使える HDL 文法概要① Verilog-HDL 編”, デザインウェブマガジン, No.13, pp.150–159.
- [3] 小林優, “初めてでも使える Verilog HDL 文法ガイド —— 記述スタイル編”,
http://www.kumikomi.net/archives/2009/07/verilog_hdl.php
- [4] 内田智久, “Verilog-HDL 入門”,
<http://research.kek.jp/people/uchida/educations/verilogHDL/>

[改訂履歴]

日付	氏名	修正内容
2012 年度 まで	-	実験内容：シミュレータによる論理回路設計
2013/3/27	大川 猛	教育用 FPGA ボード (Basys2) およびシミュレータを用いた実験内容への変更
2013/4/3	大川 猛	語句の間違いなどの微修正
2014/4/7	大川 猛	課題 1 を積和演算の内容に変更 補足 6 に乗算器使用量の調べ方を追加 課題 2 のパラメータを 2014 年度用に修正 課題 3-1 の入力が必要だった問題を修正 課題 3-3 を新規追加
2015/4/7	大川 猛	課題 2 のパラメータを 2015 年度用に修正
2016/4/6	大川 猛	課題 2 のパラメータを 2016 年度用に修正
2017/4/6	大川 猛	Basys3 ボードを用いた実験内容に変更