HDL記述による設計法をマスタする

実験で学ぶロジック回路設計

木村 真也
Shinya Kimura

第5回 HDL記述のメリット

ロジック回路をHDLで設計するメリットはいくつかありますが，そのうちの二つを紹介します．

一つは，同じ回路が簡単に使いまわせることです．何度も使う回路をモジュールとして記述しておけば，ICやツールに依存せずシンプルな記述で何度でも使えます．

もう一つは，よく使う信号処理(演算)があらかじめ用意されていることです．加算のような頻繁に使う演算でも，回路図レベルではとても手間がかかります．HDLならたった1行で記述できます．〈編集部〉

簡単なゲート回路を動作させてみる

● 8種類のゲート回路を作り込む

では，基本ゲートの動作を確認してみましょう．基本ゲートであるAND(2入力と4入力)，OR(2入力と4入力)，NOT以外にNAND，NOR，ExORを実装して動作確認をしてみます．ロジック回路図を**図5-1**に示します．

リスト5-1のようにゲート回路の記述を追加したトップ・モジュールを論理合成してCPLDに書き込

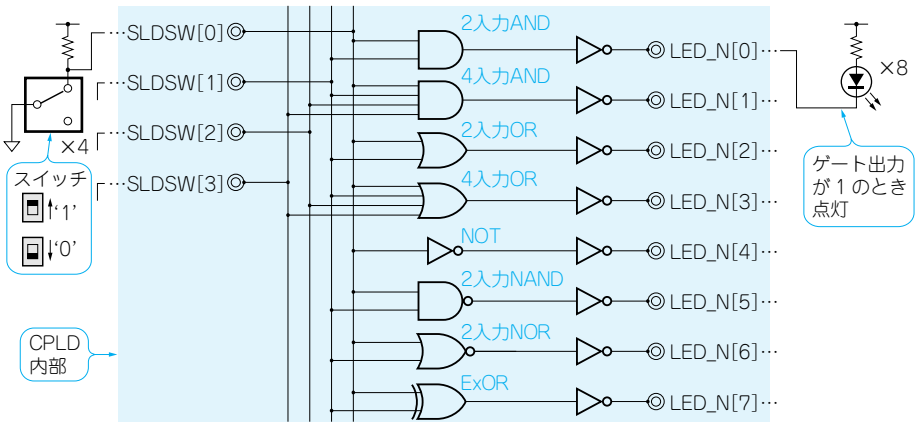

図5-1 基本ゲートの動作確認用ロジック回路
基本ゲートがCPLD内にできたことをスイッチとLEDで確認できる

Keyword 1 基本ゲート

ブール代数の演算を電子回路で実現したものを論理ゲート(あるいは単にゲート)と呼んでいます．

基本演算の論理積，論理和，否定に対応するゲートがANDゲート，ORゲート，NOTゲートでロジック回路を構成する基本部品です(**表5-A**，**図5-A**)．

表5-A 基本ゲートの真理値表

入力		AND出力	OR出力	NOT出力
A	B	A・B	A + B	¯A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

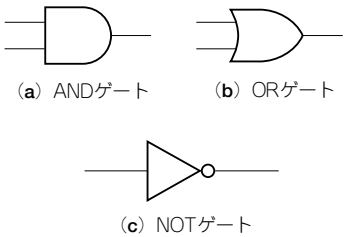

図5-A 基本ゲートの回路記号

```
 1: /* MAX2top2001.v ---------- ロジック回路実験ボード・トップ・モジュール *
 2:  * ゲート→LED 実験                                                    *
 3:  *                                    designed by Shinya KIMURA *
 4:  * ---------------------------- トランジスタ技術連載 BASICS (Digital) */
  :      (途中省略)
51:   // core module instantiation or additional logic
52:      assign led[0] =   SLDSW[0] & SLDSW[1];                        // 2-input AND
53:      assign led[1] =   SLDSW[0] & SLDSW[1] & SLDSW[2] & SLDSW[3]; // 4-input AND
54:      assign led[2] =   SLDSW[0] | SLDSW[1];                        // 2-input OR
55:      assign led[3] =   SLDSW[0] | SLDSW[1] | SLDSW[2] | SLDSW[3]; // 4-input OR
56:      assign led[4] = ~ SLDSW[0];                                   //         NOT
57:      assign led[5] = ~(SLDSW[0] & SLDSW[1]);                       // 2-input NAND
58:      assign led[6] = ~(SLDSW[0] | SLDSW[1]);                       // 2-input NOR
59:      assign led[7] =   SLDSW[0] ^ SLDSW[1];                        // 2-input EXOR
60:
61: endmodule
```

リスト5－1　基本ゲートの動作確認用回路のVerilog HDL記述
トップ・モジュールを直接書き換えている

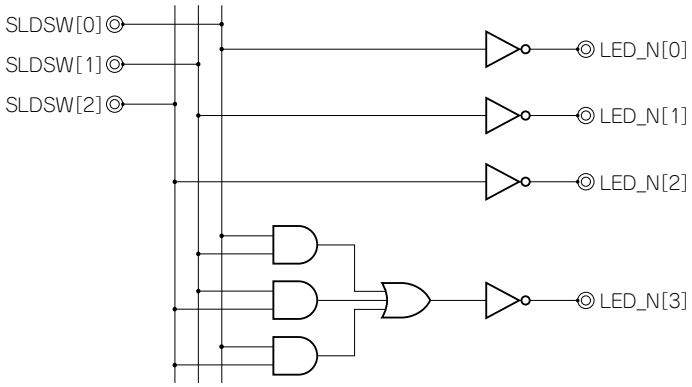

図5－2　3人の多数決結果を得るロジック回路

めば，LEDの点灯で基本ロジックの動作を確認できます．スイッチもLEDも，左端が［7］で，右に［6］，［5］…と並び，右端が［0］です．

● 3人の多数決回路を実装してみる

連載第1回で解説した3人の多数決回路を実装してみましょう．ロジック回路図は簡単なほうの論理式で作っています(**図5－2**)．トップ・モジュールは**リスト5－2**のようになります．

賛成/反対の入力はスライド・スイッチ3個を使用します．3人の投票結果はLED［2：0］（右端から三つ）に表示されます．賛成で点灯し，多数決結果はLED［3］（右端から四つ目）に表示します．

モジュール化と演算子の活用

● 4ビットの加算回路を例にしてみよう

4ビットの2進数を加算する回路を作ってみましょう．ロジック回路で何らかのデータを処理する場合，加算回路は頻繁に必要になります．4ビットを例に考

Keyword 2　演算子

Verilog HDLにはビットごとのブール演算以外に，プログラミング言語で標準的な演算が用意されていて，ほとんどが論理合成可能になっています．

- 算術演算…加算(+)，減算(-)，乗算(*)，除算(/)，剰余演算(%)，べき乗(**)
- ビット演算… AND演算(&)，OR演算(|)，NOT演算(~)，ExOR演算(^)，ExNOR(~^)
- 論理演算…論理積(&&)，論理和(||)，論理否定(!)
- シフト演算…論理左シフト(<<)，論理右シフト(>>)，算術左シフト(<<<)，算術右シフト(>>>)
- 関係演算…小なり(<)，小なりイコール(<=)，大なり(>)，大なりイコール(>=)
- 等号/不等号…一致(==)，不一致(!=)，ZやXも含む一致(===)※，ZやXも含む不一致(!==)※
  ※ === と !== は論理合成できない
- リダクション演算(ビット・ベクタに対するビット演算)…AND演算(&)，OR演算(|)，NAND演算(~&)，NOR演算(~|)，ExOR演算(^)，ExNOR(~^)
- その他
  条件演算子(?:)，連結演算子({})