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7回　定石2　コピーしながら効率良く記述

今回は，if と case の二つの予約語を紹介しましょう．

まったく同じ組み合わせ回路が複数ある場合，コピーしながら効率良く記述できるだけでなく，読みやすくなります．例として**図7－1**に示す7セグメントLED表示回路を複数個HDLで記述することを考えてみます．**写真7－1**に示すのは，ロジック回路学習ボードを使って，2桁表示の7セグメントLEDを点灯させているところです．〈編集部〉

複数の7セグLED駆動回路を作る

● ?や:を使って書くと面倒なうえに読みにくい

?や:は，同じ条件で機能する複数の式を記述する場合，同じ条件を繰り返し書いていかなければなりません．複数の7セグメントLED駆動回路を前回紹介した方法で記述すると，**図7－2(a)**のように，同じ条件を何度も書くことになりスマートではありません．

また論理合成した際に，複数の同じロジック回路が無駄に合成される可能性があります．条件を修正する必要が出てきた場合も，複数の箇所を変更しなければなりません．読みやすさの点でも劣ります．

if 文や case 文も同様に，function の中に記述して，設定した値を条件によって変更したいときに利用できます．両者の違いは，if や case は，同じ条件下で複数の式を書くことができる点です．

?，:の使いどころは，条件が単純な場合です．

● 同じ回路をコピーしながら効率良くスマートに書ける if と case

図7－2(b)に示すように，回路が複雑な場合や同じ条件下で，複数の同じ式をコピーしながら効率良く記

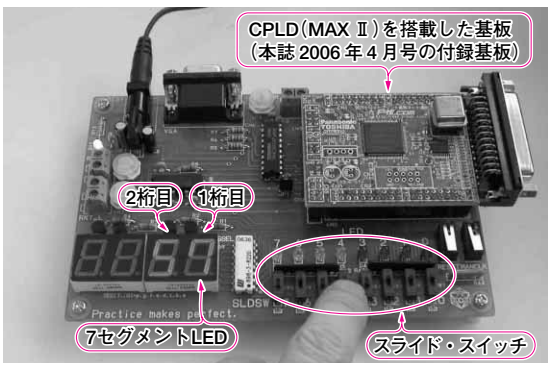


写真7－1　ロジック回路実習ボードを使って2個の7セグメントLEDに数値を表示する回路を動かす
スライド・スイッチ(0～3)の設定値を2桁7セグメントLEDの1桁目に，スライド・スイッチ(0～4)の設定値を2桁目に表示する

Keyword 1　2進数とBCDコード

BCDコード(Binary Coded Decimal code，2進化10進数符号)は，1桁の10進数の値に，4桁の2進数を対応させる数値表現のための符号です．

表7－Aにこの対応を示します．2進数4桁の各桁の重み付けは上位桁から8，4，2，1です．例えば，10進数の1234のBCDコード表現は，

BCDコード：0001001000110100
10進数　　：　1　　2　　3　　4

となります．このようにBCDコードは，2進数で表現するよりも，多くのビット数を必要としますが，10進数との対応がわかりやすい表現です．

表7－A　BCDコードと10進数
1桁の10進数の値に4桁の2進数を対応させる

10進数	BCDコード	10進数	BCDコード
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

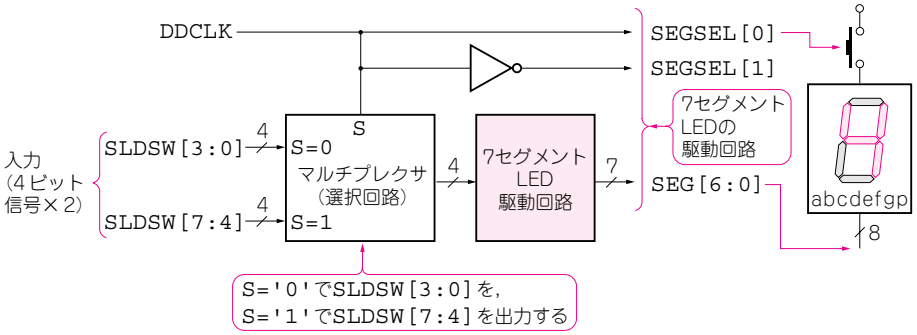
述したい場合は，予約語functionを使ってファンクション化し，if文やcase文を使って機能を記述します．ifやcaseは，同じ条件下で複数の式を書くことができます．

本稿では，このような複雑な組み合わせ回路でもスマートに記述できる，ifとcaseを紹介します．

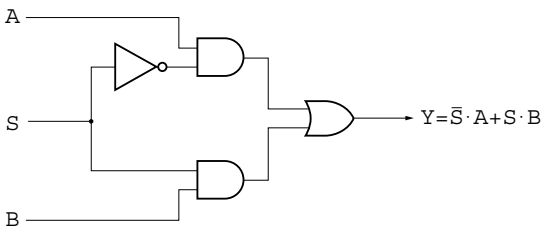
assign文の右辺をファンクション化する

● assignの右辺をモジュール名とモジュールの入出力名に変える

多少の語弊がありますが，ファンクション化とは，assign文の右辺を切り取って独立させることです．



(a) 全体回路（トップ・モジュール）



(b) 1ビット分のマルチプレクサ

図7-1　2桁の7セグメントLED表示回路(1回路分)

```
module bcd7seg2_cond_op(
    input  wire [3:0] bcd1, bcd2, ····· ,
    output wire       a1, b1, c1, d1, e1, f1, g1,
                      a2, b2, c2, d2, e2, f2, g2,
                      ·····················);

    assign {a1,b1,c1,d1,e1,f1,g1} =
                bcd1==4'b0000 ? 7'b1111110 :
                bcd1==4'b0001 ? 7'b0110000 :
                  :
                bcd1==4'b1001 ? 7'b1111011 :
                                7'b0110111 ;

    assign {a2,b2,c2,d2,e2,f2,g2} =
                bcd2==4'b0000 ? 7'b1111110 :
                bcd2==4'b0001 ? 7'b0110000 :
                  :
                bcd2==4'b1001 ? 7'b1111011 :
                                7'b0110111 ;

endmodule
```

必要な個数だけ枠内の記述を作成する

7セグメントLEDに表示する文字を変更する場合は全assign文の右辺を修正しなければならない

(a) ファンクション化しない場合(何度も同じような記述をしている)

```
module bcd7seg2_func_case(
    input  wire [3:0] bcd1, bcd2, ····· ,
    output wire       a1, b1, c1, d1, e1, f1, g1,
                      a2, b2, c2, d2, e2, f2, g2,
                      ·····················);

    function [6:0] bcd7seg_func;
        input [3:0] bcd;
        begin
            case (bcd)
              4'b0000: bcd7seg_func = 7'b1111110;
              4'b0001: bcd7seg_func = 7'b0110000;
                 :
              4'b1001: bcd7seg_func = 7'b1111011;
              default: bcd7seg_func = 7'b0110111;
            endcase
        end
    endfunction

    assign {a1,b1,c1,d1,e1,f1,g1} = bcd7seg_func(bcd1);

    assign {a2,b2,c2,d2,e2,f2,g2} = bcd7seg_func(bcd2);

endmodul
```

functionを使うと，ifやcaseといったなじみのある演算子を使える

表示の変更はfunction内だけ修正すればよい

1行分の記述を必要なだけ作成する

(b) ファンクション化すると可読性やメンテナンス性が向上する

図7-2　図7-1の回路は？や：を使うときれいに記述できない