关于 ABEL PLD 编程笔记的补充

1．关于源文件内容的说明

1）模块开始和结束语句必有。语法：module　模块名 和 end　模块名。开始和结束的模块名要一致；

2）说明段中标志语句 flag 用来给编译器提供参数，这些参数也可通过编译命令提供，但建议使用标志语句。关于可填的参数，见原文“ABEL 3.0 命令行”部分。常用的参数是-Rn 和-Tn 参数分别指明逻辑化简的级别和模拟仿真的方式。如果使用默认参数，或由命令行指定编译参数，标志语句可以省略。

标题语句常用来为设计加一些简短的说明，以增加可读性。由于 flag 和 title 后面的参数都要求字符串所以不要忘了单引号。

3）定义段中，器件定义、管脚定义必不可少，但为了简化书写和增加可读性，程序中往往也少不了常量定义。其余的定义则视需要而定。在器件定义中，器件名用 P 打头。比如 PAL16L8，要定义成 device ‘P16L8’；如果是 GAL16V8，则定义成 device ‘P16V8R’。其中 GAL 器件名后面的后缀，如上例中的 R，按如下方法取：

- 若设计中需要寄存器（时序逻辑设计），用后缀 R。如：P16V8R、P16Z8R、P20V8R 等。
- 若组合逻辑中需用反馈，用后缀 C。如：P16V8C、P16Z8C、P20V8C 等。
- 若组合逻辑中每个输出需 8 个乘积项，用后缀 S。如：P16V8S、P16Z8S、P20V8S 等。

在组合逻辑设计中，如果关系简单用 S 或 C 均可，编不通时，可变换一下后缀试试。

在管脚定义中，所定义的输出输入关系必须符合器件的规定。比如 GAL16V8 只能做输入的管脚有：1—9 和 11 脚；只能做输出的管脚有：12 和 19 脚；既能做输入又能做输出的管脚有：13—18 脚；如需时钟输入只能用 1 脚；如需使能输入只能用 11 脚。

常量定义中类似如下的程序行往往少不了：

H，L，X = 1，0，.X. ；　(让字符常量 H 在程序中代表逻辑 1，… X 在程序中代表任意态.X.)

集合等也在常量定义中说明。

4）描述段是整个程序的核心，你的设计可用逻辑表达式、真值表或状态图描述，对于每个输出用哪种方式描述，完全取决于你的意愿。如用逻辑表达式，你只要把你的设计准确无误的表达出来就行了，程序会替你化简。

5）熔丝段允许直接说明熔丝的取值。如 FUSES　[2200..2222]= ^H24 将 2200 到 2222 号熔丝置为 24H。如果对 PLD 的内部结构不是非常了解，建议不要使用熔丝段。

6）测试段同样不是必须的但往往是必要的。通过测试段可以让编译器在烧录器件之前找出设计的器件是不是真的符合设计要求。

7）除了 module， end， flag ，title 语句以外，所有语句都要向 C 语言一样以分号“;”结尾。

2．设计举例

[例一] 用 GAL16V8 设计一个可控反相器，当控制端为 0 时输出信号 Q0—Q7 分别是输入信号 D0—D7 的反相值；如果控制端为 1，输出信号 D0—D7 分别等于输入信号 D0-D7。希望通过本例说明 ABEL 源程序的基本结构和集合的使用。

源程序如下：

```
module DS000
title 'DS-1 型数据采集系统的 BCD 输入数据适配电路
晓风  1997 年 7 月 31 日    为 ARGS 公司设计'

                        U9      device    'P16V8S';
        D0,D1,D2,D3,D4,D5,D6,D7    pin 1,2,3,4,5,6,7,8;
        Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7    pin 19,18,17,16,15,14,13,12;
        CTRL    pin    9;

        input     = [D7,D6,D5,D4,D3,D2,D1,D0];
        output    = [Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0];

equations

        WHEN CTRL = = 0 THEN output = !(input);
        WHEN CTRL = = 1 THEN output = (input);

test_vectors ( [CTRL,input] -> output)
                [0,0] -> 255;
                [0,255] ->0;
                [1,0] -> 0;
                [1,255] -> 255;

end DS000
```

说明：

1）模块名称为 DS000；

2）省略 flag 语句，所以编译器使用默认值如-R1、-T0 等进行处理；

3）由于只有一个器件，所以管脚定义语句中可不用 in 指明是为 U9 定义；

4）在定义段还定义了两个集合，一个是由输入脚 D0-D7 组成的 input 集合，另一个是由输出脚 Q0-Q7 组成的 output 集合。当然，由于下标的有序性，这两个集合也可以这样定义：

```
input     = [D7..D0];
output    = [Q7..Q0];
```

5）回顾一下布尔方程的语法：

[WHEN **条件** THEN] [！] …… [ENABLE] **元素 = 表达式；**[ELSE 方程]

其中方括号中的内容为任选项，其余为必有项，套用到本例的方程：

WHEN CTRL = = 0 THEN output = !(input);

可见这是一个合法方程，它所表达的意思也很明确，就象我们的自然语言一样：当 CTRL 等于目零时，输出是输入的反相值。这就是 ABEL 带给我们的好处。

当然，如果不使用集合也可以，比如：

```
WHEN CTRL = = 0 THEN Q0 = !D0;
WHEN CTRL = = 0 THEN Q1 = !D1;
……
WHEN CTRL = = 0 THEN Q7 = !D7;

WHEN CTRL = = 1 THEN Q0 = D0;
WHEN CTRL = = 1 THEN Q1 = 1D1;
……
WHEN CTRL = = 0 THEN Q7 = 1D7;
```

显然没有使用集合来得简洁。

6）同样在测试段也可看到集合所带来的好处，这里用到了集合的赋值，当 input = 255 时相当于给 D0-D7 每个输入脚置 1；而 output = 255 则相当于 Q0-Q7 每个脚都输出逻辑 1。

[例二]用 GAL16V8 设计一个带锁存的地址分配器。要求当时钟端 LE 上升沿出现时，四个输出端 E0-E3 要根据此时数据端 D0-D7 的状态把某一个锁存为 1，其余锁存 0。同时 D0 和 D1 的值也要分别锁存在 A0 和 A1 输出端。器件还要有一个使能端 EO，只有 EO=0 时，各输出锁存的状态才能呈现在输出端上，否则均为高阻态。

当 D0-D7 = 0-3 时 E0 锁存 1，其余锁存 0；
当 D0-D7 = 4-7 时 E1 锁存 1，其余锁存 0；
当 D0-D7 = 8-11 时 E2 锁存 1，其余锁存 0；
当 D0-D7 = 12-15 时 E3 锁存 1，其余锁存 0；

希望通过本例说明带寄存器和使能控制的设计。

源程序如下：

```
module ds005
flag '-x1','-t4'
title '16 通道智能 AD 接口卡上通道选择器的地址分配器
晓风 2000 年 8 月 9 日　为 JIANGMIN 公司设计'

        U6      device   'P16V8R';

        D7,D6,D5,D4,D3,D2,D1,D0  pin in U6   9,8,7,6,5,4,3,2;
        CK,OE                Pin in U6   1,11;
        E3,E2,E1,E0                  pin in U6   17,19,16,15;
        A1,A0                        pin in U6   18,14;
```

```
        H,L,X,Z,C      = 1,0,.X.,.Z.,.C.;
        DATA = [D7..D0];
        OUTPUT = [E3,E2,E1,E0,A1,A0];

equations in U6

    E0 := (DATA >= 0) & (DATA <= 3);
    E1 := (DATA >= 4) & (DATA <= 7);
    E2 := (DATA >= 8) & (DATA <= 11);
    E3 := (DATA >= 12) & (DATA <= 15);

    A0 := D0;
    A1 := D1;

    ENABLE !OUTPUT = OE;

test_vectors in U6 ([LE,OE,DATA] -> OUTPUT)
                [C, L, 0 ] -> [L, L, L, H, L, L];
                [C, L, 1 ] -> [L, L, L, H, L, H];
                [C, L, 2 ] -> [L, L, L, H, H, L];
                [C, L, 3 ] -> [L, L, L, H, H, H];
                [C, L, 4 ] -> [L, L, H, L, L, L];
                [C, L, 5 ] -> [L, L, H, L, L, H];
                [C, L, 6 ] -> [L, L, H, L, H, L];
                [C, L, 7 ] -> [L, L, H, L, H, H];
                [C, L, 8 ] -> [L, H, L, L, L, L];
                [C, L, 9 ] -> [L, H, L, L, L, H];
                [C, L, 10] -> [L, H, L, L, H, L];
                [C, L, 11] -> [L, H, L, L, H, H];
                [C, L, 12] -> [H, L, L, L, L, L];
                [C, L, 13] -> [H, L, L, L, L, H];
                [C, L, 14] -> [H, L, L, L, H, L];
                [C, L, 15] -> [H, L, L, L, H, H];
                [C, H, 15] -> [Z, Z, Z, Z, Z, Z];
                [C, H, 0 ] -> [Z, Z, Z, Z, Z, Z];

end ds005
```

说明：

1） flag '-x1','-t4' 中的-x1 让编译器在输出编程器下载文件中把源程序中的测试向量里的任意态（.X.）都用逻辑 1 代表；-t4 让编译器在仿真输出文件（*.sim 文件）中给出仿真波形；

2） 因为数据要锁存，所以要用寄存器，因此器件定义中器件名的后缀为 R；

3） 在管脚定义中时钟脚 LE 一定要是 1 脚，使能脚一定要是 11 脚；
4） 在方程表达式中，因为时序电路要时钟赋值，所以要用“:=” 赋值号；
5） 使能表达式 ENABLE !OUTPUT = OE;是说 OE=1 时输出不使能（等于说 OE=0 输出使能），而不能理解为普通的表达式。事实上，这如果是普通的表达式，那也是个非法表达式，因为单个变量是不能对集合赋值的；
6） 在测试向量中给出了各种输出组合情况下期望得到的输出值。编译器将根据你在描述段中的设计得出的输出和这些期望值进行比较，如果不符就会报出有多少向量没有通过测试。因此测试向量一定要安排合理，能够说明问题。要做到：如果能够通过测试，设计一定符合要求，如果没能通过则设计一定有缺陷。

事实上，只要设计成功一个范例，以后就有了一个基本的结构框架，今后的编程实际上就是粘来贴去，只要有需求、肯实践，你一定会成功的。

3．纠正原文中几处错误

1）第 2 页上关于字符串的叙述中：“字符串是包含在**双**引号内的 ASCII 字符序列”应为“字符串是包含在**单**引号内的 ASCII 字符序列”。“ DMI ‘P16L8’;”应为“ DEVICE ‘P16L8’;”

7）第 3 页上“4 种基数表示方法”表中二进制值^b101 的十进制应为 5；

8）第 4 页上[例 1]中所有双等号= =，均应改为单等号=。如 ChipSel = Addr = = [1，0，1]；应改为 ChipSel = Addr = [1，0，1];

4）第 12 页上 2 和 3 小标题中的“IF - THEN – THEN”应为“IF - THEN – ELSE”。

错误肯定还有，希望大家指正。

晓风 treehouse@sina.com