ABEL语言简介

ABEL语言，全称是高级布尔方程语言（Advanced Boolean Equation Language），它是用于对可编程逻辑器件或复杂可编程逻辑器件进行设计的一种高级程序语言，用ABEL语言编写后缀名为.abl的源文件，在经过相应的ABEL软件编译生成可烧录的文件（.jed文件）后，就可以下载到可编程逻辑器件中。

ABEL语言支持逻辑方程、真值表和状态图三种逻辑描述方式。

ABEL语言与VHDL语言相比，具有简单易学的特点。

例：四位二进制数比较器，如图1所示：

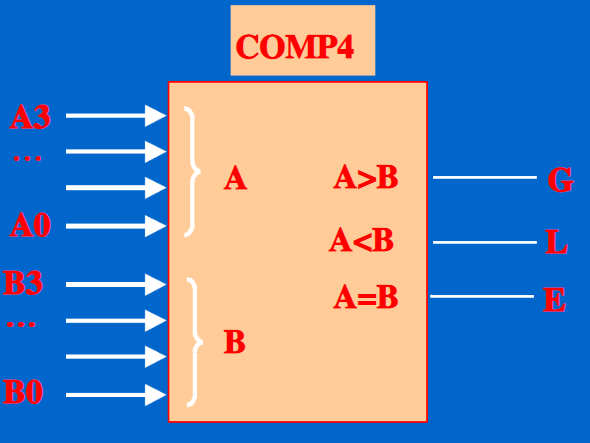


图1 四位二进制数比较器

以四位二进制数比较器为例来说明ABEL语言的程序结构：  
module COMP4 ——模块名

declarations ——变量说明段

A3, A2, A1, A0 pin;

B3, B2, B1, B0 pin;

G, L, E pin istype ‘com’;

A=[A3..A0]; B=[B3..B0];

equations ——逻辑方程段

G = (A>B);

L = (A<B);

E = (A=B);

test\_vectors([A, B] -> [G, L, E]) ——测试向量段

[0, 0] -> [1, 0, 0];

[0, 1] -> [0, 1, 0];

[15, 10] -> [0, 0, 1];

end COMP4

1. ABEL语言基本语法（元素）

1.1标识符与关键字

标识符通常用来表示标识器件名称、器件引脚名称、输入或输出信号的名称、状态名称、集合名称、宏、节点、常量及模块名称等。（注：标识符不能和关键字重名）

字符串：用单引号括起来的一系列字符。

操作数：

·二进制数：^B1010 ^b1010

·八进制数：^O67 ^o67

·十进制数：^D97 ^d97

·十六进制数：^H7EF ^h7EF

·字母：转换成对应的二进制ASCII码

标识符必须以英文字母或下划线“\_”开头，其后可跟数字、英文字母、下划线以及波浪线。

标识符最长不得超过31个字符。

在ABEL语言中标识符区分大小写，如Ac和ac是两个不同的标识符。

关键字是一些具有特殊用途的保留标识符。

关键字不得用来给器件、引脚、节点、常量、集合、宏和信号命名。

关键字可以用大写、小写或大小写混合方式输入，它们表示的含义相同。

1.2 专用常量

以下列出ABEL语言中的专用常量，如表-1所示：

表-1 专用常量

|  |  |  |
| --- | --- | --- |
| 常量 |  | 说明 |
| .C. |  | 时钟脉冲输入（低电平——高电平——低电平转换） |
| .D. |  | 脉冲下降沿（高电平——低电平转换） |
| .F. |  | 悬浮的输入或输出信号 |
| .K. |  | 时钟输入（高电平——低电平——高电平转换） |
| .P. |  | 寄存器预加载 |
| .U. |  | 时钟脉冲上升沿（低电平——高电平转换） |
| .X. |  | 任意值 |
| .Z. |  | 三态值 |

.表示方法是在英文字母左下方及右下方各加一个圆点。

.专用常量字母可以用大写或小写字母输入。

1.3 注释

有两种注释方式：  
（1）用一对””括起来或用”开始，以行结束来结束。

（2）用//开始，以行结束来结束。

1.4运算符和逻辑表达式

算术运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 表达式举例 | 说明 | 运算符优先级 |
| - | -A | 求二进制补码 | 1 |
| — | A—B | 减运算 | 3 |
| + | A + B | 加运算 | 3 |
| \* | A \* B | 乘运算 | 2 |
| / | A / B | 无符号整除 | 2 |
| % | A % B | 取模：无符号数整除的余数 | 2 |
| << | A << B | A左移B位 | 2 |
| >> | A >> B | A右移B位 | 2 |

逻辑运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 表达式举例 | 说明 | 运算符优先级 |
| ! | !A | 逻辑非 | 1 |
| & | A&B | 逻辑与 | 2 |
| # | A#B | 逻辑或 | 3 |
| $ | A$B | 逻辑异或 | 3 |
| !$ | A!$B | 异或非（同或） | 3 |

关系运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 表达式举例 | 说明 | 运算符优先级 |
| = = | A == B | 等于 | 4 |
| != | A != B | 不等于 | 4 |
| < | A < B | 小于 | 4 |
| <= | A <= B | 小于或等于 | 4 |
| > | A > B | 大于 | 4 |
| >= | A >= B | 大于或等于 | 4 |

赋值运算符：

|  |  |  |
| --- | --- | --- |
| 运算符 | 表达式举例 | 说明 |
| = | A = B | 非时钟赋值（把B的值立即赋给A，没有时延） |
| := | A := B | 时钟赋值（在下一个时钟脉冲有效沿来后，把B的值赋给A） |

1.5 块

块是括在大括号“{}”内的一段ASCII码文本，括号内的文本可以是一行，也可以是多行。

块用于方程式、状态图、宏定义和指示字中，块可以嵌套使用，即块中包含块。

1.6 集合

集合是作为一个整体参与运算的一组信号或常量。

它采用一个标识符，用方括号内的一组信号或常量来表示，其中的每一个信号或常量称为集合的元素。

例：

Addr = [A7, A6, A5, A4, A3, A2, A1, A0]; 枚举法

Addr = [A7..A0]; 界限符法

Addr = [A7..A2, A1, A0]; 两种方法结合

集合的赋值和运算

例：①若已定义sign=[A,B,C];

则可以这样赋值：sign=[1,1,0]; 或者sign=6;

也可以这样运算：sign=[1,1,0]&[1,0,1]; 或者sign=6&5;

②[a,b]=^bxx11; 等同于[a,b]=^b11;

③[a,b]=^b1; 等同于[a,b]=^b01;

④[a,b]=c&d; 等同于a=c&d; b=c&d;

1.7 变量及变量代换

哑变量：在宏、模块和指示字中，可以被真实变量替代的标识符。

真实变量：用于宏、模块和指示字中的变量，真实变量可以替代哑变量。

例：在宏定义中有：OR\_EM MACRO(a,b,c){?a#?b#?c}; a,b,c是哑变量，在宏定义体中，哑变量前带有问号

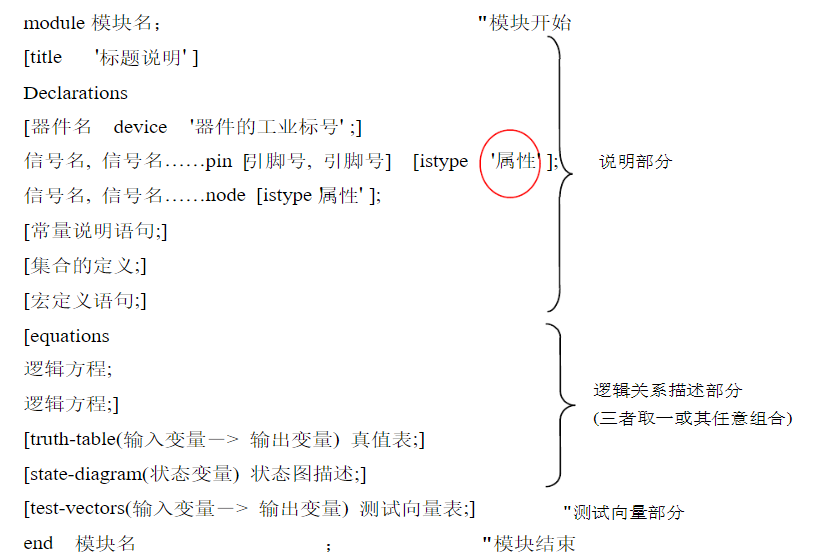
在方程中有：D=OR\_EM(X,Y,A&B); 这个方程中调用了OR\_EM宏，结果为：D=X#Y#(A&B);

1.8 常用的点后缀

表-2 ABEL语言中常用的点后缀

|  |  |
| --- | --- |
| 点后缀 | 含 义 |
| .AP | 寄存器异步置位 |
| .AR | 寄存器异步复位 |
| .CE | 时钟导通触发器的时钟输入 |
| .CLK | 边沿触发器的时钟输入 |
| .D | D触发器的激励输入 |
| .FB | 寄存器反馈信号 |
| .J | JK触发器的J激励输入 |
| .K | JK触发器的K激励输入 |
| .LD | 寄存器并行置数输入 |
| .LE | 锁存器锁存使能输入 |
| .LH | 锁存器的锁存使能 |
| .OE | 输出使能 |
| .PIN | 引脚反馈 |
| .Q | 寄存器输出（书写源文件时可以省略） |
| .R | SR触发器的R激励输入 |
| .RE | 寄存器复位信号（同步或异步） |
| .S | SR触发器的S激励输入 |
| .SP | 寄存器同步置位 |
| .SR | 寄存器同步复位 |
| .T | T触发器的激励输入 |

1. ABEL源文件的基本结构



基本的ABEL程序通常由头段、说明段、逻辑描述段和结束段等部分组成。在说明段中，需要说明选用的信号名称、类型及其管脚分配等。

信号包括I/O信号，它直接与管脚连接（要通过PIN指定IO引脚号）和内部隐埋的节点信号（要用node说明且不能分配IO引脚号）两大类。

信号类型包括组合逻辑类型（用ISTYPE ‘COM’说明）和时序逻辑类型（用ISTYPE ‘REG’说明），未明确说明的信号，默认其类型是组合逻辑。

逻辑描述段由保留字EQUATIONS开始，用于给出每一位IO信号，每一个节点信号的逻辑方程。赋值号’=’用于向组合逻辑信号赋值，赋值号’:=’用于向时序信号赋值。

某些管脚可以分时用于输入和输出，会用到三态逻辑，需要写出单独的控制语句，通常在变量名后接’.OE=控制信号名’的方式指出输出是正常电平还是高阻态。

需要为触发器电路指定时钟脉冲信号，要写出单独的语句，通常用在变量名后接’.CLK=脉冲信号名’的方式指出用到的时钟信号。

每一个赋值语句用（;）结束

程序中的寄存器通常选用D型触发器构成，运行时需用时钟的上升边沿控制接收，要通过点后缀方式为寄存器指定时钟信号，例如pc.clk = CLK

向寄存器赋值的2种方式：  
①在条件语句中使用:=符号，当条件成立时将把输入数据保存到寄存器；当条件不成立时，会把0保存到寄存器，这通常是需要避免的。为此，可以把寄存器的内容作为输入执行一次写操作，例如：when cif then ir:=DB; else ir:=ir; ir.clk=CLK;指令寄存器只在取指周期（cif）接收从内存读来的指令，在其他周期ir接收自己的内容，而不是被清零。

②在条件赋值语句中，可以通过点后缀方式（.ce）说明寄存器有输入使能控制功能，并改用=作为赋值符，例如，when cif then ir=DB; ir.ce=cif; ir.clk=CLK;则ir只在取指周期接收DB，其他情况ir内容将保持不变。

使用@include ‘文件名’语句可以把一段ABEL程序引入到一个程序模块中。

信号属性说明，如表-3

表-3 信号属性

|  |  |
| --- | --- |
| 属性字符串 | 意义 |
| ‘com’ | 组合型输出 |
| ‘reg’ | 寄存器型输出 |
| ‘invert’ | 目标器件中的反相器 |
| ‘buffer’ | 目标器件中的缓冲器 |
| ‘neg’ | 未规定的逻辑为‘1’ |
| ‘pos’ | 未规定的逻辑为‘0’ |
| ‘keep’ | 不将此信号从方程式中简化掉 |
| ‘reg\_d’ | D型触发器 |
| ‘reg\_jk’ | JK型触发器 |
| ‘reg\_sr’ | SR型触发器 |
| ‘reg\_t’ | T型触发器 |

下面以4-2线编码器的例子来讲解ABEL程序设计的方法：

例：设计一个4-2线编码器。

依据真值表来设计逻辑关系，真值表见表-4

表-4 4-2线编码器真值表

|  |  |
| --- | --- |
| 输入 | 输出 |
| I0 I1 I2 I3 | Y1 Y0 |
| 1 0 0 0 | 0 0 |
| X 1 0 0 | 0 1 |
| X X 1 0 | 1 0 |
| X X X 1 | 1 1 |

MODULE CODER24 模块语句

TITLE ‘CODER24’ 标题语句

I3,I2,I1,I0 PIN; 定义段

O1,O0 PIN ISTYPE ‘COM’; 组合型输出

TRUTH\_TABLE 用真值表实现

([I3,I2,I1,I0] -> [O1,O0])

[0,0,0,1] -> [0,0];

[0,0,1,0] -> [0,1];

[0,1,0,0] -> [1,0];

[1,0,0,0] -> [1,1];

END

1. ABEL语言语句

3.1 方程式语句

方程式中的赋值符为“=”或“:=”。

在方程中允许使用WHEN-THEN-ELSE语句。

但不能使用IF-THEN-ELSE。

EQUATIONS

[WHEN 条件 THEN][!] 变量名=表达式；

[ELSE 方程式;]

或者

[WHEN 条件 THEN]方程式;

[ELSE 方程式;]

例：EQUATIONS

X = !(a & b); “与非门

Y = a # b; “或门

!z = a $ b; “异或门

例：设计一个4位左移移位寄存器。

MODULE SHIFT4

TITLE ‘SHIFT4’

DIN,CLK,CLR PIN;

Q3..Q0 PIN ISTYPE ‘REG’; 输入输出信号定义

OUT = [Q3..Q0];

EQUATIONS

OUT.CLK = CLK;

OUT.AR = CLR;

Q0 := DIN;

Q1 := Q0;

Q2 := Q1;

Q3 := Q2;

END

例：设计一个四选一数据选择器。

MODULE MUX4S1

TITLE ‘MUX4S1’

S1,S0 PIN;

D3,D2,D1,D0 PIN;

Y PIN ISTYPE ‘COM’;

S=[S1,S0];

EQUATIONS

WHEN (S==0) THEN Y = D0;

WHEN (S==1) THEN Y = D1;

WHEN (S==2) THEN Y = D2;

WHEN (S==3) THEN Y = D3;

END

例：设计一位十进制加法计数器。

MODULE count10

q3,q2,q1,q0 PIN ISTYPE ‘reg’;

clk,cd PIN;

count = [q3..q0];

EQUATIONS

count.clk = clk;

count.ar = cd;

WHEN(count==9)THEN count:=0;

ELSE count:=(count.fb + 1);

END

3.2 真值表语句

它用表格的形式将输出定义为输入的函数。

以关键字TRUTH\_TABLE为开始，后面跟表头向量和表格。

TRUTH\_TABLE(输入->输出)

输入->输出；

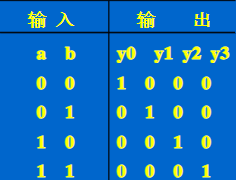
...

或 TRUTH\_TABLE(输入[:>寄存器型输出] -> 输出;)

...

输入[:>寄存器型输出] -> 输出;

...

例：2/4线译码器

MODULE decoder

DECLARATIONS

a, b pin;

y0,y1,y2,y3 pin istype ‘com’;

TRUTH\_TABLE ([a,b] -> [y0,y1,y2,y3])

[0,0] -> [1,0,0,0];

[0,1] -> [0,1,0,0];

[1,0] -> [0,0,1,0];

[1,1] -> [0,0,0,1];

END decoder

例：设计一位十进制加法计数器及七段译码电路

MODULE count

clk pin;

q3..q0 pin istype ‘reg’;

a,b,c,d,e,f,g pin istype ‘com’;

H, L = 1, 0;

count = [q3..q0];

EQUATIONS

[q3..q0].clk = clk;

TRUTH\_TABLE

([count]:>[count]->[a,b,c,d,e,f,g])

[0]:>[1]->[1,1,1,1,1,1,0];

[1]:>[2]->[0,1,1,0,0,0,0];

[2]:>[3]->[1,1,0,1,1,0,1];

[3]:>[4]->[1,1,1,1,0,0,1];

[4]:>[5]->[0,1,1,0,0,1,1];

[5]:>[6]->[1,0,1,1,0,1,1];

[6]:>[7]->[0,0,1,1,1,1,1];

[7]:>[8]->[1,1,1,0,0,0,0];

[8:]>[9]->[1,1,1,1,1,1,1];

[9]:>[0]->[1,1,1,0,0,1,1];

END

3.3 状态图语句

STATE\_DIAGRAM 状态寄存器

[->状态输出]

[STATE状态表达式:[方程式]

[方程式]

...

状态转移描述]

ABEL语言提供三种语句来描述状态转移:

无条件转移语句(GOTO语句)

条件转移语句(IF-THEN-ELSE语句)

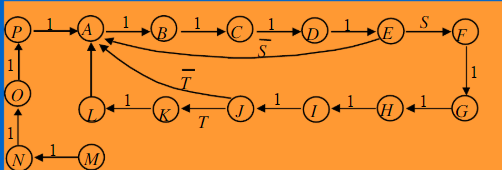
多路转移语句(CASE-ENDCASE)；

此外，状态机在发生转移的同时可能会有相应的数据输出，所以还有条件输出语句(WITH语句)。

例：变模计数器的设计

设计一个变模计数器，在S和T的控制下，实现同步模5或模10或模12的递增计数。

|  |  |  |
| --- | --- | --- |
| S | T | 模式 |
| 0 | 1 | 模5计数 |
| 1 | 0 | 模10计数 |
| 1 | 1 | 模12计数 |

 module MACHINE

title ‘VARIABLE\_MODULO\_COUNTER’

declarations

CP, OE, S, T pin;

Q3, Q2, Q1, Q0 pin istype ‘reg,buffer’;

“set definitions”

Q = [Q3,Q2,Q1,Q0];

A = [0,0,0,0]; B = [0,0,0,1]; C = [0,0,1,0]; D = [0,0,1,1];

E = [0,1,0,0]; F = [0,1,0,1]; G = [0,1,1,0]; H = [0,1,1,1];

I = [1,0,0,0]; J = [1,0,0,1]; K = [1,0,1,0]; L = [1,0,1,1];

M = [1,1,0,0]; N = [1,1,0,1]; O = [1,1,1,0]; P = [1,1,1,1]; 计数器状态定义

equations

[Q3,Q2,Q1,Q0].CLK = CP;

[Q3,Q2,Q1,Q0].OE = !OE; 逻辑方程段

state\_diagram Q

state A: goto B;

state B: goto C;

state C: goto D;

state D: goto E;

state E: if(!S & T) then A

else F;

...

state J: if(S & !T) then A

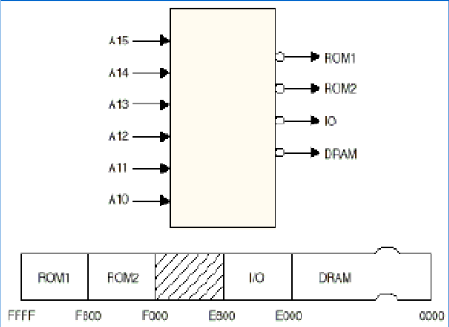
else K;

...

End

1. ABEL语言电路设计举例

|  |  |
| --- | --- |
| 存储器段 | 地址范围（HEX） |
| DRAM | 0000～DFFF |
| IO | E000～E7FF |
| ROM2 | F000～F7FF |
| ROM1 | F800～FFFF |

 存储器地址译码器的设计（地址译码器的设计是PLD的典型应用之一）

module decode

title ‘memory decode’

A15, A14, A13, A12, A11, A10 pin;

ROM1, IO, ROM2, DRAM pin istype ‘com’;

H, L, X = 1, 0, .X.;

Address = [A15, A14, A13, A12, A11, A10, X, X, X, X, X, X, X, X, X, X];

equations

!DRAM = (Address <= ＾Hdfff);

!IO = (Address >= ＾hE000) & (Address <= ＾hE7FF);

!ROM2 = (Address >= ＾hF000) & (Address <= ＾hF7FF);

Test\_vectors

(Address -> [ROM1, ROM2, IO, DRAM])

＾h0000 -> [H, H, H, L];

＾h4000 -> [H, H, H, L];

＾h8000 -> [H, H, H, L];

＾hC000 -> [H, H, H, L];

＾hE000 -> [H, H, L, H];

＾hE800 -> [H, H, H, H];

＾hF000 -> [H, L, H, H];

＾hF800 -> [L, H, H, H];

End

1. ABEL语言编译与综合（附录）

ABEL的编程设计大致可以分为建立源文件、编译和生成JEDEC(可编程逻辑文件)编程器装入文件、下载编程和调试运行四个步骤。

1. 建立源文件，可以使用ABEL或VHDL硬件描述语言提供的文本编辑器，建立或编辑扩展名分别为“.abl”或“.vhd”的源文件。
2. 编译源文件并生成扩展名为“.jed”的结果文件。用ispLEVER软件完成。
3. 下载结果文件到MACH芯片中，可使用Lattice Semiconductor公司的ispLEVER软件或 ispVM System软件完成。
4. 调试、运行完成下载的硬件系统。

下面介绍ispLever软件、ispVM System软件的使用方法和操作过程。

ISPLEVER软件是Lattice Semiconductor公司的产品。ISPLEVER软件将器件选择，源文件的建立或导入，源文件的编辑、编译，功能模拟，生成编程文件等诸多功能都集成在工程项目引导器（ISPLEVER Project Navigator）中。引导器帮助用户完成整个设计的全过程。下面我们将以实验1的内容为例对它的具体使用方法进行说明。

给出的源文件可以使用文本编辑器进行编辑，假设文件被存放在D/shiyan1文件夹中。

1. 在工程项目引导器中创建一个新工程项目

可以按照以下步骤创建一个新的工程项目，首先运行ISPLEVER软件，并启动工程项目引导器（ISPLEVER Project Nevigator）。

a.文件菜单中，单击新工程项目（New Project）命令；

在ISPLEVER中一个工程项目就是一个设计，每一个工程项目对应一个独立的的目录，它包含所有的源文件、中间的数据文件和结果文件。

对于新启动的工程项目引导器，位于左边的源文件(Sources in Project)窗口中一般仍保存有前一次工程项目文件，在执行a步骤前，使用文件菜单中关闭工程项目（Close Project）命令将其清除。

b.在新创建的工程项目对话框中，选择或新建新的工程项目的保存目录，输入工程项目文件名（\*.syn）或使用默认的工程项目名untitled.syn。从四种工程项目类型（Project type）中ABEL、VHDL、Verilog HDL 、EDIF选择一种，例如实验单独MACH芯片的运算器实验中所给例子使用ABEL-HDL描述语言，所以选择ABEL类型，如图-1所示。最后单击“保存”按钮，返回到图-2工程项目引导器窗口。

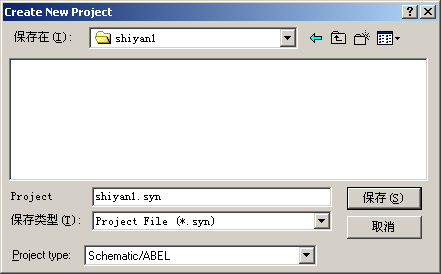
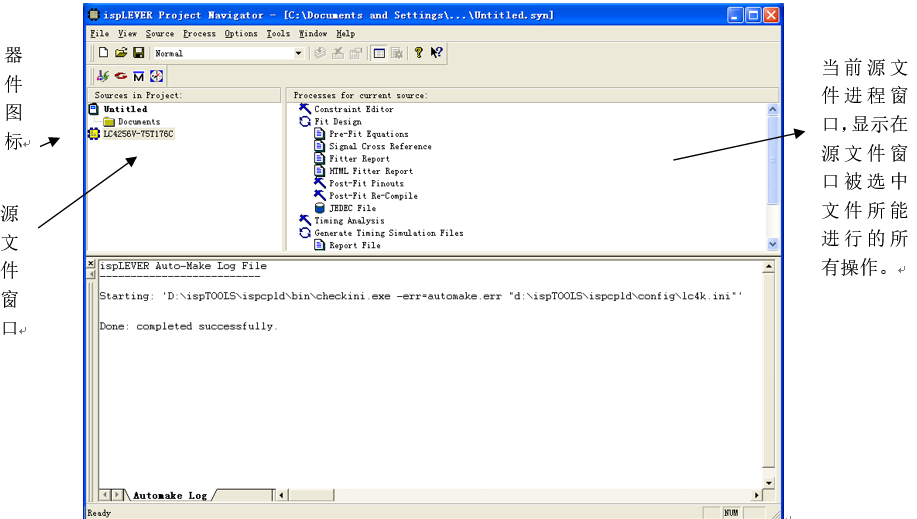
c.双击图-2中的器件图标，出现器件选择(Device Selector)对话框，通过下拉菜单在Family 和 Device两个选项中选择你所使用的器件，其它选项在选好Family 和 Device两个选项后会自动配置，不需选择。我们用的是Lattice LC4256V 器件，因此，Family选项选ispMACH 4000，Device选项选LC4256V。

图-1 创建新工程项目对话框

 图-2 工程项目引导器

1. 导入一个已有的源文件或新建一个源文件

a.在Source菜单中，单击Import命令。出现Import File对话框，双击对话框中的shiyan1.abl文件，则该源文件出现在工程项目引导器源文件窗口中，如图-3所示。

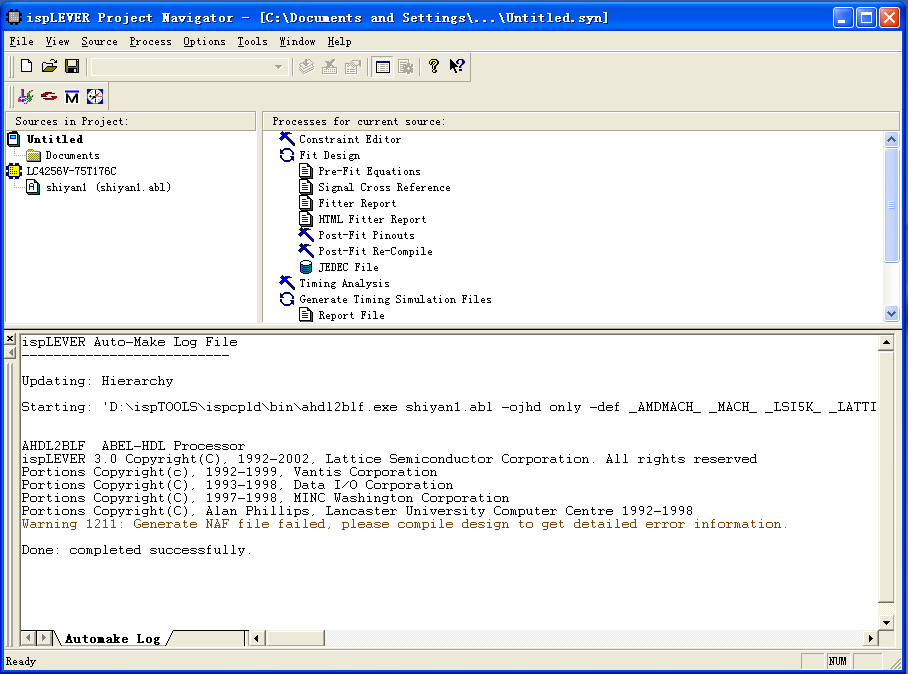
也可使用Source菜单中的New命令，创建一个新的源文件。

图-3 选择已有的或建立新的源文件

b. 双击图3所示的MACH文本文件图标，将运行文本编辑器（Text Editor），被编辑文件的内容将显示在工程项目引导器右侧的文本显示窗口中，接着可以开始编辑操作，编辑结束后需要通过保存选项保存编辑结果，这里的操作过程和屏幕显示内容从略。

1. 编译

单击图3所示的MACH文本文件（shiyan1.abl）图标，则在图3右侧当前源文件进程窗口中将显示Compile Logic任务项，双击该选项则启动对源文件shiyan1.abl编译操作，此项操作只是检查并指出源文件中的语法错误。如有错，就可进入文本编辑器来执行修改操作。如编译通过，系统会在Compile Logic任务项前以绿色的“√”标记。

1. 器件引脚分配

在源文件shiyan1.abl中直接对引脚进行了定义既可。

1. 生成JEDEC编程文件

a. 在图3左侧源文件窗口中点击器件图标，则在图3右侧当前源文件进程窗口中将显示几种可执行的任务项，双击其中的JEDEC File任务项，就启动对原文件的编译和优化操作，这里的编译是针对所选择的器件进行的，要检查并指出器件引脚指定是否有错，芯片资源使用是否全部支持，不合理等会进行检查，如有错，就可进入文本编辑器来执行修改操作。

b. 若无错，结束编译后会在任务项前以红色的“√”标记。若有警告信息，则以红色的“!”标记，警告信息通常并不影响JEDEC文件的使用。

六、 以MACH芯片为例讲述.jed类型文件的下载操作方法

启动Lattice Semiconductor公司的ispVM System软件，弹出主界面。

（1）接好教学计算机上在线MACH编程电缆，打开教学计算机电源。

（2）通过主界面的SCAN按钮找到在线编程器件，如下图-4所示。

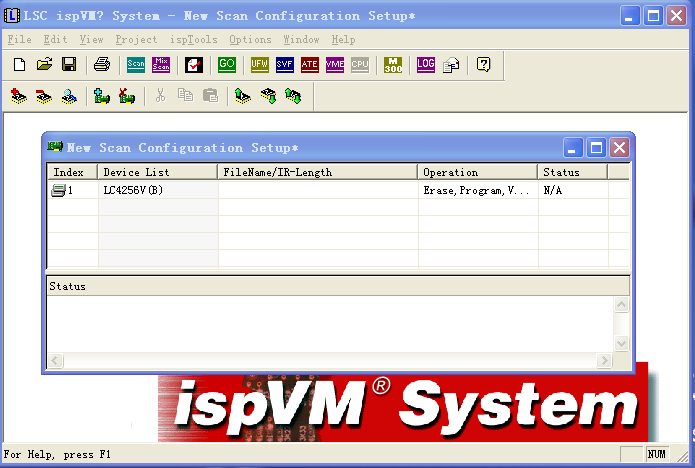


图-4扫描所用的器件型号

（3）双击图-4中所找到的编程器件LC4256V，窗口显示如图-5所示，通过Browse按钮选择已编译好的JED文件（shiyan1.jed），单击OK。

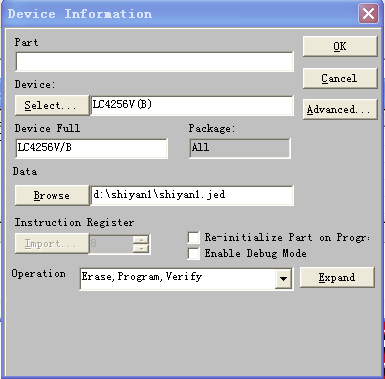


图-5在线编程操作

（4）重新回到主界面，点击go命令启动下载操作。编程完成后，可关闭编程窗口。