深圳市紫光同创电子有限公司
SHENZHEN PANGO MICROSYSTEMS CO.,LTD

IP Compiler

开发者参考手册

Rev. 20231111E

目 录

1. 概述 4
1.1 概念与流程 4
1.2 文件与目录结构 6
1.3 顶层模块单元 8
2. 模型索引文件 10
2.1 顶层元素 10
2.2 头部基本信息 11
2.3 参数列表 13
2.4 操作列表 14
2.4.1 条件属性 15
2.4.2 synthesize 操作 16
2.4.3 run 操作 16
2.5 兼容的软件工具 17
2.6 支持的器件 18
2.7 规格说明 19
3. 标记源文件 20
3.1 顶层元素 20
3.2 参数值的替换 21
3.3 IP 基本属性的替换 22
3.4 循环值的替换 23
3.5 端口声明的替换 24
3.6 对部分文本的显示控制 25
3.6.1 用参数值控制 25
3.6.2 用端口状态控制 25
3.6.3 嵌套使用 25
3.7 不需要标记的文本 26
4. 调用综合工具 27
5. 编程参考 28
5.1 用户界面描述文件 28
5.2 控制流程的入口 30
5.3 图形界面的布局 32
5.4 如何定义和操作模型参数 36
5.5 窗口组件 38
5.5.1 输入类组件 38
5.5.2 显示类组件 46
5.5.3 容器类组件 50
5.5.4 其它组件 54
5.6 对象的控制操作 55
5.7 参数值的获取与设定 61
5.8 IP 基本属性的获取 65
5.9 封装管脚信息的获取 68
5.10 实例符号及管脚的定义与操作 69
5.11 信息区域的操作 74
5.12 综合工具的设置 74
5.13 信息提示 75
6 IP 测试 77
6.1 测试 IP 模型 77
6.2 测试 IP 实例 78

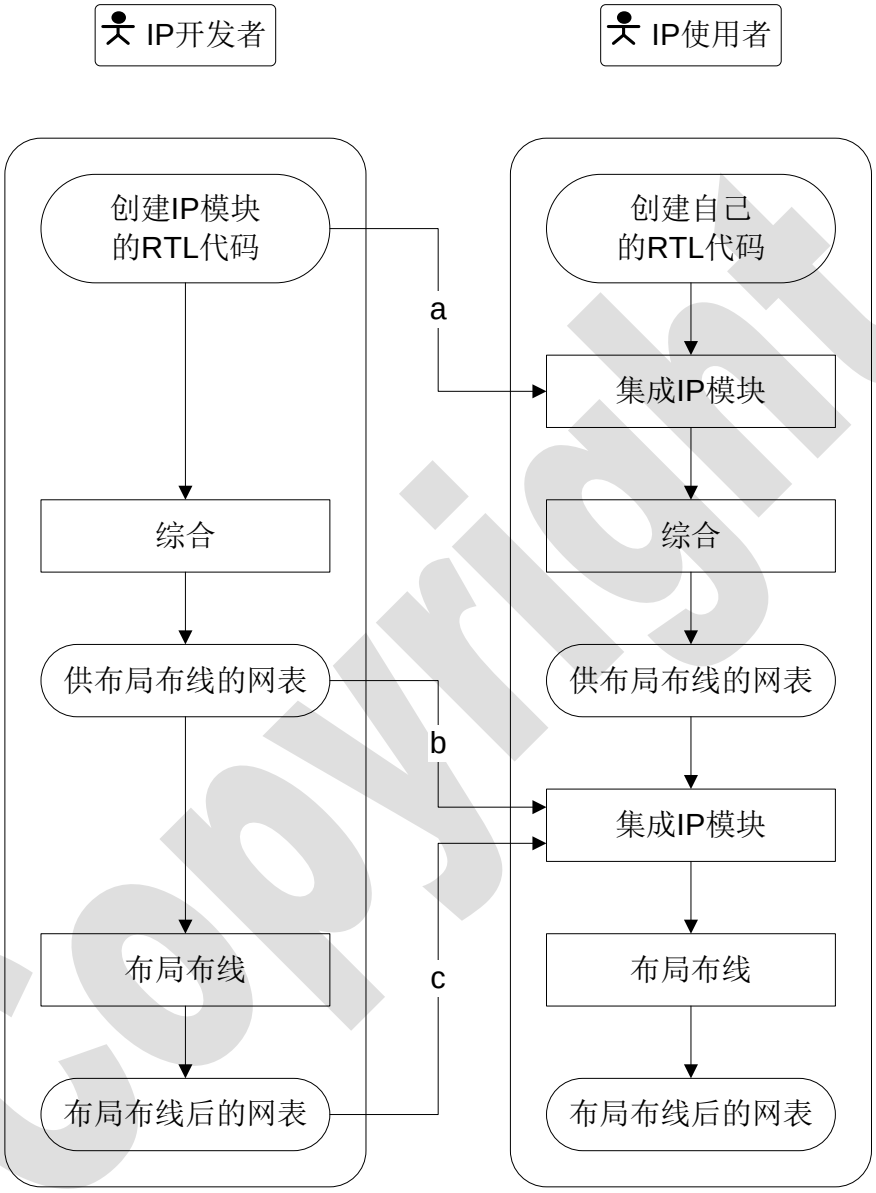
6.3 测试举例 78
7 IP 的升级安装与管理 79
7.1 IP 模块内容的准备 79
7.2 IP 包的制作 80
7.3 IP 包的测试 81
7.4 用户端更新 82
7.5 冲突解决 83
附录：Tcl 过程函数索引 84

本文主要帮助 IP 开发者理解 IP Compiler 系统的使用流程，提供相关的文件格式定义和 Tcl 过程定义，并以此为基础开发和维护 IP 库。

1. 概述

_x0002_.1 概念与流程

任何现有的功能模块都可以称作 IP，IP 可以来源于 FPGA 厂商也可以来自第三方 IP 提供商。如下图所示，FPGA 设计人员可以通过几个途径来使用一个 IP 模块。



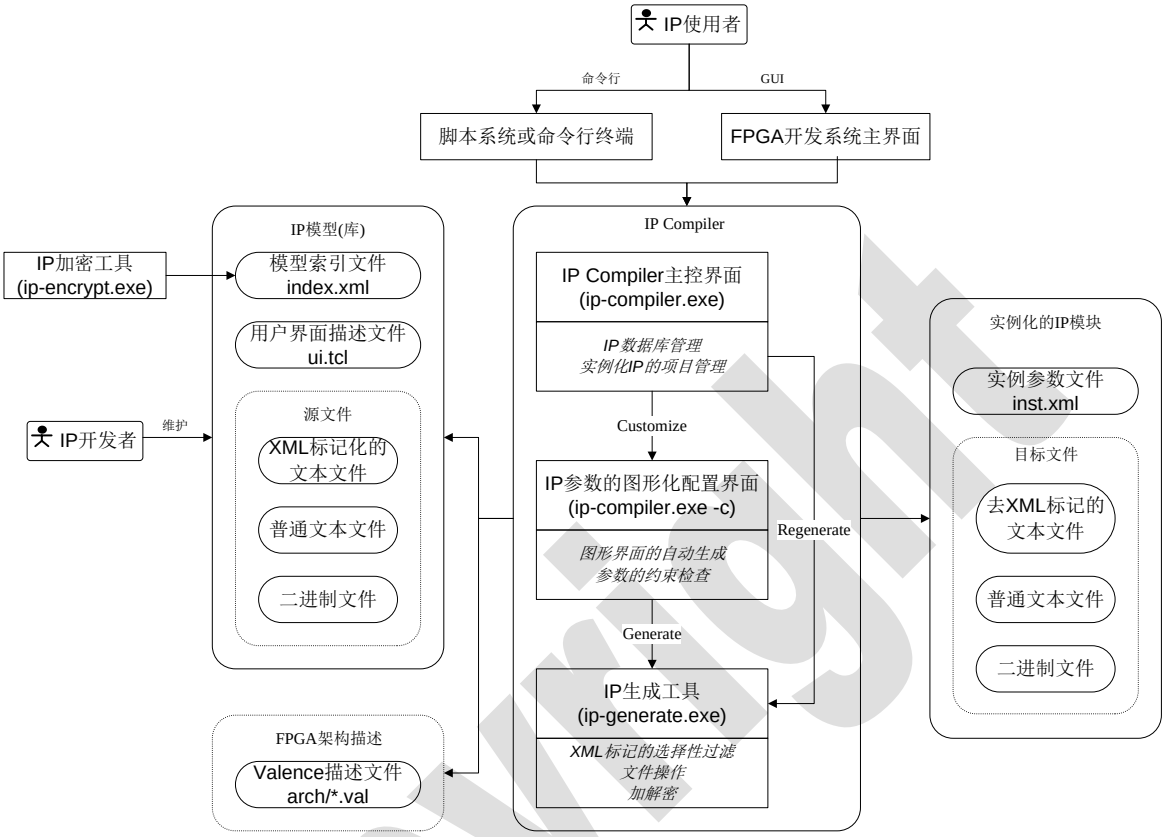
FPGA 设计人员可以购买(加密的)RTL 级 IP 模块，然后将这些 IP 模块集成到设计的 RTL 代码中，如上图中的 a 路径所示；另一种方式是使用未经布局布线的网表级 IP，如上图中的 b 路径所示，这样做的好处是 IP 提供商已经做了许多工作来调整综合引擎并对某些功能进行了优化，以便在资源利用和性能等方面达到最优；还有一种情况是使用布局布线后的网表级 IP，如上图中的 c 路径，这样做是为了得到最高的性能，比如有些 IP 模块有特殊的 I/O 引脚的要求，这时构成 IP 模块的元素的位置就不能任意改变。

FPGA 厂商的普遍做法是提供一个专门的工具作为 IP 模块的生成器，这些生成器软件都是参数化的，可以由用户设定各种参数。本文要讨论的 IP Compiler 就是这样一种软件。

一个 IP 模块在计算机磁盘上通常是以一个单独的目录的形式存放的，该目录下可以有若干

文件，也可以包含子目录，IP 开发者可以自由地将相关文件放置在这个目录中。根据该思路设计的 IP Compiler 可以与具体的 IP 无关，是一个通用的、具备较强扩展性的软件系统，方便 IP 使用者的理解，也力图减轻 IP 开发者的负担。目前阶段 IP Compiler 仅能处理 RTL 级的 IP 模块。

综合以上的信息，我们可以给出 IP Compiler 的系统框图如下：



IP 开发者在对一个 IP 进行组织打包时，需要做以下一些工作：

- 将相关的所有文件(和子目录)存放在一个特定的目录中；
- 编写一个索引文件来描述这个 IP 的属性及配置入口等；
- 编写一个文件来描述 IP 参数和用户界面，以及生成实例时要做的具体工作；
- 给需要进行参数配置的(文本)文件增加特定的标记，而二进制文件则不加处理；

从系统框图中可以看到，IP 开发者除了要掌握 HDL 语言外，还需要了解两种语言格式：一个是 Tcl 语言，这是一种在 IC 设计和 EDA 领域比较流行的脚本语言，在这里用来描述用户界面；另一个是 XML 语言，这是一种互联网时代发展起来的技术，其优点之一是可以很方便地给文本文件加标记从而方便计算机软件的处理，在这里用来描述 IP 模型的基本信息以及对可配置的(HDL)文件进行标记化。

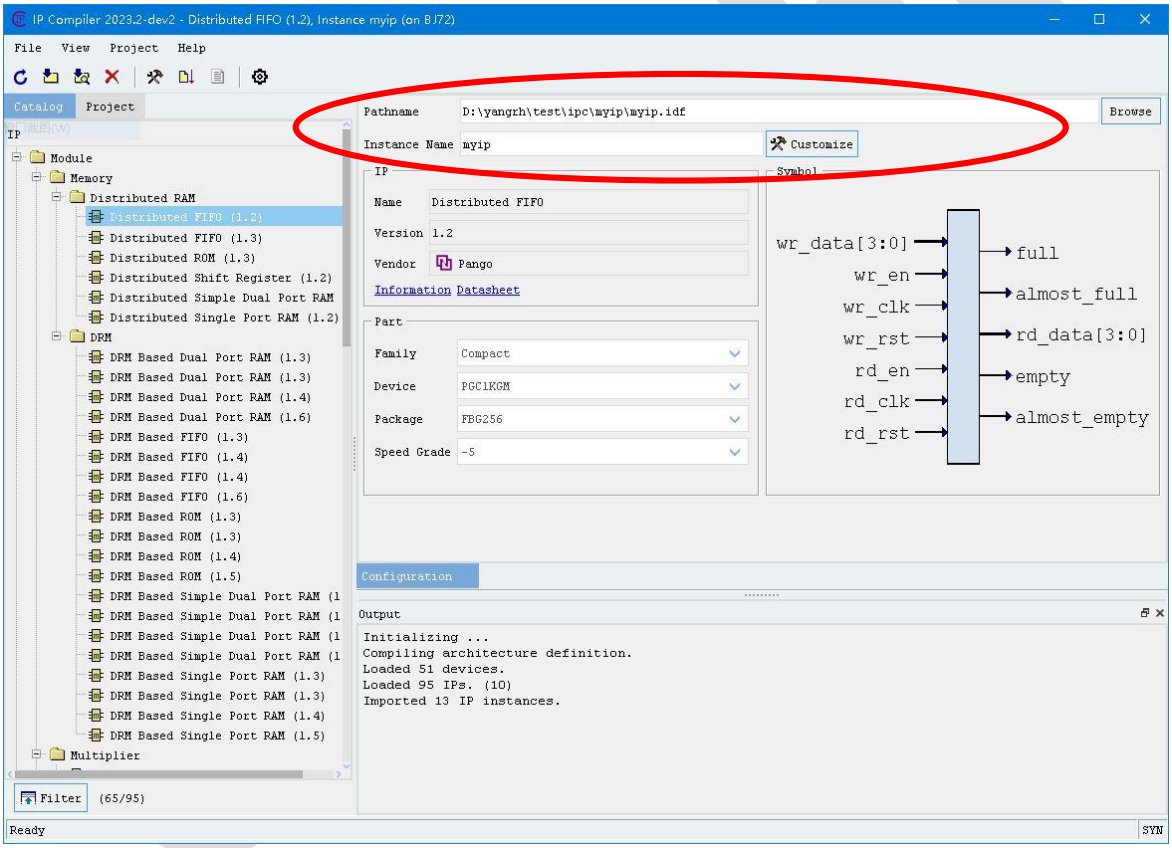
1.2 文件与目录结构

从前面章节的描述中可以看到与 IP 相关的文件与目录位于两个磁盘空间中，IP 开发者在编程时应该使用这两个空间中的相对路径，并不要使用任何绝对路径：

- 发布的 IP 模块放置于单独的目录中(model path)，每个模块位于一个子目录中，由索引文件 *index.xml* 来标识这是一个 IP 模块；通过设置环境变量 *IP_LIBRARY_PATH* 可以让 IP Compiler 找到这些 IP 模块。
- 由 IP Compiler 配置并例化的 IP 位于另一个目录中(instance path)，每一个 IP 实例都有独立的子目录，由数据文件 *inst.idf* 来标识这是一个 IP 实例。

在 IP Compiler 主窗口中指定的 Project Path 实际上是 IP 实例目录的上层目录(见下图)。

例如：输入 Instance Path 为 D:\ipc\a，输入 Instance Name 为 a，则会生成 D:\ipc\a 目录并在该路径下进行各种操作；再配置另一个 IP，输入 Instance Path 为 D:\ipc\b，输入 Instance Name 为 b，则会生成 D:\ipc\b 目录，两者是互不干扰的。



需要注意的是，在 IP Compiler 中，有一些文件名是系统保留的：*index.xml* 是 IP 模型的索引文件，*inst.idf* 是 IP 实例的数据文件等。为了保证得到正确的结果，IP 开发者应避免文件/目录名的冲突，一种比较好的习惯是将所有要被处理的 HDL 文件都放置于 IP 模型/实例路径中的一个子目录中，比如叫作 hdl 或 src 等。

1.3 顶层模块单元

有三种方式可以指定顶层单元的名称。按照优先级从高到低依次如下：

(1) 在 IP 代码中通过调用 *ui::setTopModule* 指定顶层单元的名称，该名称会被保存在 IP 实例的数据文件中。

- ***ui::setTopModule***

ui::setTopModule		
参数	**name,n**	字符串值； 顶层模块单元的名称；如果为空串，则表示使用 IP 模型中的缺省定义；
参见	**ui::getTopModule**	

- ***ui::getTopModule***

ui::getTopModule	
返回值	顶层模块单元的名称；来自于 *ui::setTopModule* 语句或 IP 模型的缺省定义；
参见	**ui::setTopModule**

(2) 在 IP 模型模型索引文件中的头部，可以用***<top_module>***指定一个缺省的顶层单元名，在没有调用过 *ui::setTopModule* 的情况下，该缺省名会被使用。例如：

```
<header>
    ...
    <top_module>add_or_substract</top_module>
    ...
</header>
```

(3) 如果通过前两种方式都不能获得，则会使用 IP 实例的名称作为顶层单元名。

在标记化的 HDL 文件模板中，可以用***<top_module>***或***<iname>***来替换顶层单元名。注意，在指定顶层单元名时，应使其符合相应 HDL 的语法规则，使得可以通过编译和综合。

```
module <top_module/> (
    a, b, op, s);
    input [7:0] a;
    input [7:0] b;
    input op;
    output [7:0] s;

    add_or_substract i0 (
        .a(a),
        .b(b),
        .op(op),
        .s(s)
    );
endmodule
```

在生成 IP 的过程中，IPC 会产生两个例化模板文件<inst-name>*_tmpl.v* 和 <inst-name>*_tmpl.vhdl*，来展示调用顶层模块单元的接口。

2. 模型索引文件

IP 模型索引文件用来标识一个 IP 单元，它位于一个 IP 单元的根目录中，并且名称是固定的，为 ***index.xml***。从文件名可以看出，这是一个 XML 格式的文件。本章主要描述模型索引文件的格式，其中涉及到 license 的内容会在其它章节中讲述。

2.1 顶层元素

IP 模型索引文件的顶层元素为***<ip_index>***，用来标识该文件，从而识别一个有效的 IP 单元。在顶层元素中包含几个子元素：***<header>***，***<license>***，***<tool>***，***<param_list>***，***<action_list>***，***<specs>***等。

```
<ip_index>

    <header>
        ...
    </header>

    <license>
        ...
    </license>

    <tool>
        ...
    </tool>

    <supported>
        ...
    </supported>

    <param_list>
        ...
    </param_list>

    <action_list>
        ...
    </action_list>

    <specs>
        ...
    </specs>

</ip_index>
```

2.2 头部基本信息

在元素***<header>***中包含了一个 IP 的基本信息。

```
<header>
    <id>...</id>
    <display_name>...</display_name>
    <name>...</name>
    <version>...</version>
    <category>...,...,...</category>
    <ui>...</ui>
    <datasheet>...</datasheet>
    <info>...</info>
    <top_module>...</top_module>
    <vendor full_name="name">...</vendor>
    <logo>...</logo>
    <attribute name="value" />
</header>
```

在头部中定义的元素有：

<id>	用来标识一个 IP 的唯一的字符串码
<display_name>	IP 的显示名称；用于在各种界面中显示用，但并不用来标识；
<name> (废弃)	IP 的名称；在**<id>**和**<display_name>**不存在的情况下，可以代替之； 在新开发的 IP 中不要再使用；
<version>	IP 的版本号
<category>	IP 的分类，可以是一个用逗号分割的列表；
<ui>	用户界面描述文件的路径名，是一个 IP 单元根目录的相对路径；缺省为 *ui.tcl*；
<datasheet>	有关该 IP 的数据手册文件，是一个 IP 单元根目录的相对路径；可省略；
<info>	有关该 IP 的简要描述的 HTML 文件，可以显示在 IP Compiler 的主控界面中，是一个 IP 单元根目录的相对路径；可省略；
<top_module>	(缺省的)顶层模块(单元)的名称；可省略；
<vendor>	IP 开发者名称，大小写不敏感；可省略，缺省为 Pango；
<logo>	IP 开发者图标；可省略，缺省为紫光同创的标识；
<attribute>	属性

需要特别说明的是，因为历史原因，在一些早期版本中，只存在***<name>***，它实际上同时起到了***<id>***和***<display_name>***的作用；在一些过渡版本中，有可能同时存在***<name>*** ***<id>***和***<display_name>***，此时***<id>***的优先级要高于***<name>***；而在新开发的版本中，建议不要再使用***<name>***。

目前支持的属性有：

load_defaults_for_new_part	如果该属性被设为真，则在配置 IP 实例时，如果改动了器件设置，就会自动用缺省的参数值替换上次配置的参数值；

2.3参数列表

在元素***<param_list>***中可以定义一系列该 IP 所用到的参数，这些参数以后是可以进行配置的。可选属性 ***decimal*** 定义所有浮点类型参数的缺省显示精度(小数位数)。

每个***<param>***元素定义一个(组)参数。可选属性 ***from*** 和 ***to*** 可以指定一个整数范围，从而一次生成一组参数；

```
<param_list decimal="4">

    <param from="0" to="3">
        <name>...</name>
        <type>...</type>
        <default>...</default>
        <min>...</min>
        <max>...</max>
        <msb>...</msb>
        <lsb>...</lsb>
        <item>...</item>
        <tip>   </tip>
        <decimal>   </decimal>
        <visible>   </visible>
    </param>
        ...
</param_list>
```

在参数列表中可以定义多个参数，每个参数定义中的元素有：

<name>	参数的名称；如果***<param>***元素中 ***from/to*** 属性指定了索引值的范围，则名字中的字符串***%i*** 会被依次替换成索引值
<type>	参数的类型，目前支持的类型有：***string***(字符串)，***bool***(布尔)，***int***(有符号整数)，***float***(双精度浮点数)，***enum***(枚举/多选一)，***list***(列表/多选多)，***logicvec***(逻辑向量)
<default>	参数的缺省值；如果未设定则字符串参数的缺省值为空串，其它类型参数则为未设定状态；
<max>	整数和浮点数的最大值，如果未设定则整数的最大值为 2147483647，浮点数值为 1.79769e+308
<min>	整数和浮点数的最小值，如果未设定则整数的最小值为-2147483648，浮点数值为-1.79769e+308
<msb>	一个整数值，用来定义逻辑向量中的最高数位索引
<lsb>	一个整数值，用来定义逻辑向量中的最低数位索引
<item>	枚举和列表类型的选项，可以有多个
<tip>	关于该参数的简单描述
<decimal>	一个整数值，用来定义浮点类型参数的显示精度(小数位数)；可以覆盖**<param_list>**中的定义
<visible>	一个布尔值，用来控制是否在图形界面中显示本参数，缺省为显示

2.4操作列表

在元素***<action_list>***中可以定义一系列生成该 IP 的实例时所要进行的操作，这些操作在***ip-compiler*** 中调用 ***Generate*** 或者 ***Regenerate*** 时会执行。

```
<action_list>

    <action condition-attributes>
        <type>copy</type>
        <src>...</src>
        <dest>...</dest>
    </action>

    <action condition-attributes>
        <type>mkdir</type>
        <dir>...</dir>
    </action>

    <action condition-attributes>
        <type>compile</type>
        <file>...</file>
        <output>...</output>
    </action>

    <action condition-attributes>
        <type>run</type>
        <file>...</file>
    </action>

    <action condition-attributes>
        <type>synthesize</type>
        <file>...</file>
        <file condition-attributes>...</file>
    </action>

    <action condition-attributes>
        <type>interrupt</type>
    </action>

</action_list>
```

在操作列表中可以定义多个操作，***synthesize*** 操作总是在最后执行，而其它操作都是按照定义顺序执行的。

每个操作中定义的元素有：

<type>	操作的类型，目前支持的类型有：***copy***(拷贝)，***mkdir***(创建目录)，***compile***(编译)，***run***(执行 Tcl 文件)，***synthesize***(综合)，***interrupt***(中断后续的操作)；
<src>	在 ***copy*** 操作中用来指定源文件/目录，必须是一个 IP 模块目录的相对路径，支持含有星号*和问号?的通配符；
<dest>	在 ***copy*** 操作中用来指定目标文件/目录，必须是一个 IP 实例目录的相对路径；可以省略，此时与源文件/目录同名；
<dir>	在 ***mkdir*** 操作中用来指定目标目录，必须是一个 IP 实例目录的相对路径；
<file>	在 ***compile*** 操作中用来指定输入的标记化的 HDL 文件，必须是一个 IP 实例目录的相对路径，支持含有星号*和问号?的通配符；在 ***synthesize*** 操作中用来指定输入的 HDL 文件，必须是一个 IP 实例目录的相对路径，支持含有星号*和问号?的通配符，还可以含有***%iname%***(会被替换为实例的名称)，***%top_module%***则会被替换为顶层模块单元的名称；在 ***run*** 操作中用来指定被执行的 Tcl 文件，是一个 IP 模块目录的相对路径或者一个绝对路径；
<output>	在 ***compile*** 操作中用来指定输出文件名的格式，如果不指定则保持原文件名不变；这个格式必须是一个 IP 实例目录的相对路径，其中可以含有***%iname%***(会被替换为实例的名称)和***%bname%***(会被替换为输入文件不含后缀的名称)，***%top_module%*** 则会被替换为顶层模块单元的名称；

2.4.1 条件属性

对每个***<action>***元素还可以设置可选的条件属性 *condition-attributes*，包括器件属性和参数属性两种，可以混合使用。这些属性之间是“与”的关系，即只有当这些条件都满足时，才会执行该操作。

器件条件属性包括 ***family***、***device***、***pack/package*** 和 ***speed/speedgrade*** 等四种，分别对应于 FPGA 的系列、器件、封装形式和速度等级。注意，如果一个器件条件属性没有设置或者值为空，则等同于该属性没有限制。

```
<action family="Titan" speed="-6">
```

参数条件属性是指参数值满足一定条件，格式为***param:***<参数名>="<条件符号><参数值>"，其中的条件符号可以为**=**（可省略）、**!=** 、**>**、**>=** 、**<**、**<=**等六种（后四种条件仅对整数和浮点数类型的参数有效，对其它类型的参数使用会返回不确定值）。

```
<action param:p1="true" param:p2=">100">
```

对 ***synthesize*** 操作中的***<file>***项也可以使用条件属性。

2.4.2 synthesize 操作

在 ***synthesize*** 操作中指定的 HDL 文件，将会在生成 IP 实例后被加入到文件列表中，从而能被 PDS 等工具所识别并进行加载；注意，即使综合操作被禁止，这个文件列表仍然可以生成。

synthesize 操作总是在最后执行。如果有多个符合条件的 ***synthesize*** 操作，则只有第一个会被执行。

2.4.3 run 操作

调用 ***run*** 操作时，系统会执行一个指定的 Tcl 文件，其路径通常是一个相对于模块路径的相对路径名，也可以是一个绝对路径名(但不建议这样做)。在该文件中除了可以使用 Tcl 的基本语句外，还可以调用用户界面描述文件中定义的大部分与图形界面无关的过程。在这个文件退出时，如果返回 0 则表示有问题需要中断其它操作，否则返回其它非 0 值表示正常。

2.5 兼容的软件工具

在元素***<tool>***中可以指明本 IP 与哪些软件工具的哪些版本相匹配。不匹配的 IP 单元是不能在 IP Compiler 中装载的。如果***<tool>***部分没有定义，或者其内容为空，则表示没有软件工具兼容性的限制。

```
<tool>

    <pds>
        <version>2018.3-SP2</version>                    ①
        <version>2017.*</version>                        ②
        <version from="2016.2" to="2020.3" />            ③
    </pds>

</tool>
```

在工具列表中可以定义多个软件工具，对每个工具项可以定义多个兼容的版本：

<version>	指定一个可以兼容的软件工具版本，该版本号需要与相应软件中显示的一致；有两种定义方法：如①②所示可以具体指定一个版本，支持含有星号*和问号?的通配符；或者如③所示指定一个起止的版本范围，属性 **from** 和 **to** 支持含有星号*和问号?的通配符，如果没有指定则表示该项没有限制；

2.6 支持的器件

在元素***<supported>***中可以指明本 IP 所支持的器件种类。在 IP Compiler 中不能对某个 IP 不支持的器件进行配置。如果***<supported>***或其下的某个***<family>***、***<device>***、***<package>***部分没有定义，或者其内容为空，则表示该项没有限制。

```
<supported>

    <family name="Titan">
        <device name="PGT30">
            <package name="FFBG484" />
        </device>
    </family>

    <load_defaults from="device1" to="device2" />

</supported>
```

在器件列表中可以定义多个系列，对每个系列可以定义多个器件及其封装形式：???yrh

<family>	指定一个器件系列；属性 **name** 支持含有星号*和问号?的通配符；
<device>	指定一个器件类型；属性 **name** 支持含有星号*和问号?的通配符；
<package>	指定一个封装形式；属性 **name** 支持含有星号*和问号?的通配符；

2.7 规格说明

在元素***<specs>***中可以指明本 IP 所用到的一些规格说明文件，比如设计约束 FDC 文件等。这些文件可以是保存在 IP 模型目录中并复制到 IP 实例中的，也可以是由 IP 代码动态生成到 IP 实例中的，所以它们都应该是对于 IP 实例目录的相对路径。在 IPC 执行 Generate 时，记录于此的这些文件路径名会保存到 IP 实例的数据文件中，从而可以被其它工具模块所引用。

```
<specs>

    <fdc>my.fdc</fdc>

</specs>
```

3. 标记源文件

对 IP 单元中那些需要进行参数配置的源文件进行 XML 标记化可以帮助计算机软件进行处理。这里的标记化是指在文本文件中插入一些 XML 的元素和属性，而其原始本质并未加改变，只是加上了一个 XML 的外壳。

3.1 顶层元素

标记化文本文件的顶层元素为***<ipc>***，这个元素需要将整个文件包括在内。该元素有两个属性 ***true*** 和 ***false***，用来在替换布尔类型的参数值时用什么来表示真假，因为对于不同的(硬件描述)语言可能是不同的，在缺省的情况下用 **true** 和 **false** 来表示。

```
<ipc true="t" false="f">
    ...
</ipc>
```

3.2 参数值的替换

参数值的替换是进行配置时最常用的操作，通过插入***<value>***元素可以对要替换的参数进行标记。这种替换有两种方式：

一种是插入一对***<value>/</value>***将一段文字包括起来，该段文字中所有的字符串***%v*** 都会被替换为属性 ***param*** 所指定的参数的值。

```
<value param="myParamName">
    myParamName = %v
</value>
```

另一种是通过自封闭的***<value>***元素实现的简洁写法，该元素会被替换为属性 ***param*** 所指定的参数的值，不过一次只能替换一个值。

```
myParamName = <value param="myParamName" />
```

在上述两个例子中，替换后的结果均为：

```
myParamName = myParamValue
```

对于逻辑向量参数，可以通过属性 ***logicbase*** 指定所需的数值进制，在仅含有 0/1 时可以转化为八或十六进制，缺省为二进制。

```
myParamName = 'b<value param="myParamName" />
myParamName = 'o<value param="myParamName" logicbase="8" />
myParamName = 'h<value param="myParamName" logicbase="16" />
```

注意，如果一个参数值未被设定也没有缺省值，则它的值不能被处理，IPC 此时会提示警告信息。

3.3IP 基本属性的替换

- 实例名称

有的时候可能需要用到所配置的 IP 实例名，比如用来代替作为顶层模块名，则可以通过插入自封闭的***<iname>***元素来实现。

```
module <iname/>
```

- 器件名称

通过插入自封闭的 XML 元素可以获得当前 FPGA 器件的设置信息，见下表。

<family/>	替换为 FPGA 系列名称
<device/>	替换为 FPGA 器件名称
<package/>	替换为封装形式
<speedgrade/>	替换为速度等级

- 顶层模块单元名称

如果在 IP 代码中调用 ui::setTopModule 或在 IP 模型索引文件中指定过顶层模块名，则可以通过插入自封闭的***<top_module>***元素来实现。

```
module <top_module/>
```

3.4循环值的替换

这里定义了一种 for 循环，可以处理源文件中可能出现的数组式的定义，进行一种局部的展开替换。

具体方法是插入一对***<for>/</for>***将一段文字包括起来，该段文字中所有的字符串***%a*** 都会被替换为属性 ***array*** 的值，而所有的字符串***%i*** 都会被替换为从属性 ***from*** 到属性 ***to*** 定义的索引值，这个索引值可以是一个整数常量或者一个整数型参数的名字。属性 ***array*** 是可选的。

```
<for array="myArrayName" from="0" to="3">
    param[%i] = %a[%i]
</for>

<for array="myArrayName" from="myParamName1" to="myParamName2">
    param[%i] = %a[%i]
</for>
```

在这个简单的例子中，替换后的结果为：

```
param[0] = myArrayName[0]
param[1] = myArrayName[1]
param[2] = myArrayName[2]
param[3] = myArrayName[3]
```

3.5端口声明的替换

输出一个端口的声明。如果属性 ***name*** 所指定的端口是可见的，则该端口的 Verilog 格式的说明会输出，否则会隐藏(丢弃)。可选属性 ***type*** 的有效值为 reg 和 wire。常用于在模块 module 定义的开始部分声明端口，以便于 Verilog 代码中的端口与 symbol 中的端口保持一致。

```
<port_declaraion name=”myPortName” type=”myPortType” />
```

例如一个名为 A 的可见的输入端口，其 MSB 为 31，LSB 为 0，则：
<port_declaration name=”A” />会被替换为 input [31:0] A；

参见***<show port=…>***的用法。

3.6 对部分文本的显示控制

通过一个布尔类型的参数或者一个端口的状态，可以对源文件中的一段文字是否显示进行控制，这样做的目的是在不同的配置情况下可以有选择的将某些文字暴露给 IP 使用者。具体的方法插入一对***<show>/</show>***将一段文字包括起来。如果***<show>***元素未指定 ***param*** 或 ***port*** 属性则该段文字会被输出。

3.6.1 用参数值控制

比较常见的是通过判断参数的值来进行控制。

对***<show>***元素可以设置参数条件属性，这些属性之间是“与”的关系，即只有当这些条件都满足时，则该段文字会输出，否则会隐藏(丢弃)。

参数条件属性是指参数值满足一定条件，格式为***param:***<参数名>="<条件符号><参数值>"，其中的条件符号可以为**=**（可省略）、**!=** 、**>**、**>=** 、**<**、**<=**等六种（后四种条件仅对整数和浮点数类型的参数有效，对其它类型的参数使用会返回不确定值）。

```
<show param:p1="true" param:p2=">100">
    ...
</show>
```

3.6.2 用端口状态控制

也可以根据一个端口的状态来进行控制。如果属性 ***port*** 所指定的端口是可见的，则该段文字会输出，否则会隐藏(丢弃)。经常用于在模块的定义或例化的开始部分声明端口，以便于 HDL 代码中的端口与 symbol 中的端口保持一致。

```
<show port="myPortName">
    ...
</show>
```

参见***<port_declaration>***的用法。

3.6.3 嵌套使用

<show>元素的一个特点是它可以再包含除了顶层元素***<ipc>***以外的其它控制元素，包括***<show>***自己，以方便使用并可以形成更复杂的控制逻辑，参见下面的这段例子。

```
<show param="enable_inc">
    <show param="inc_a">
        `include "<iname/>_init_params_a.v"
    </show>
    <show param="inc_b">
        `include "<iname/>_init_params_b.v"
    </show>
</show>
```

3.7 不需要标记的文本

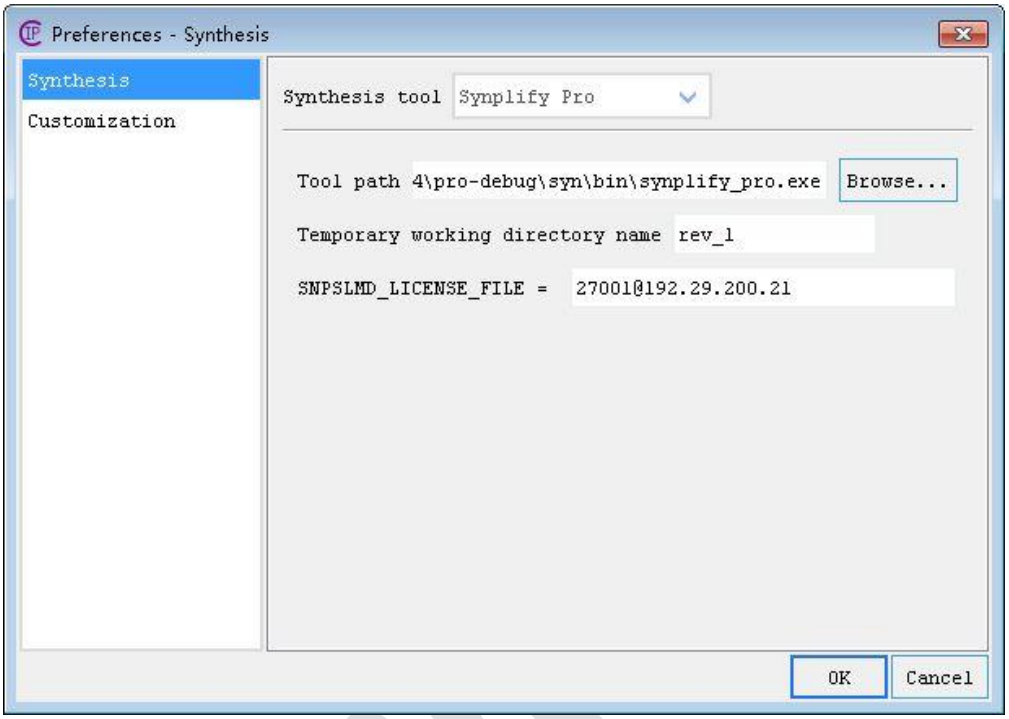
源文件中的大部分可能都是不需要处理的文本，在进行标记化时最简单的做法是对这些文本不做任何处理，而只要放在顶层的**<*ipc*>**元素中即可。

但是一些特殊字符可能会与 XML 产生冲突，比如小于号<和大于号>，可以手工地将小于号改写为**<**，大于号改写为**>**。但如果特殊字符很多或者不想改动原始文本，则可以使用所谓的 CDATA 节，CDATA 代表字符数据，表示它不需要被标记，这是一种避免重复转义字符的方法。

```
<![CDATA[
    ...
]]>
```

4. 调用综合工具

IP Compiler 可以调用不同的综合工具，比如 Synplify Pro 或 ADS 等，来对一个配置好的 IP 进行综合。外部综合工具的路径是可以指定的。如果所调用的综合工具是需要 license 的，则使用者需要自行保证其 license 设置的正确性。下图展示的是对综合工具进行设置的场景：



IP 开发者可以指明一个 IP 是需要综合的，为此需要在模型索引文件 *index.xml* 中增加一个综合操作，并指定需要综合的 HDL 文件，见下面的例子。因为操作列表是按顺序执行的，所以综合通常是最后一个操作。

```
<action_list>

   ...

   <action>
      <type>synthesize</type>
      <file>hdl/*.v</file>
   </action>

</action_list>
```

IP Compiler 在调用综合工具之前会先生成一个工程文件，自动设置输入的 HDL 文件以及输出的结果文件和日志文件等。如果使用 Synplify Pro，IP 开发者也可以进行其它设置，这是通过在用户界面描述文件 *ui.tcl* 中调用过程 *ui::addSynOption* 及 *ui::clearSynOptions* 来实现的。这些设置语句也会自动加入到工程文件中。

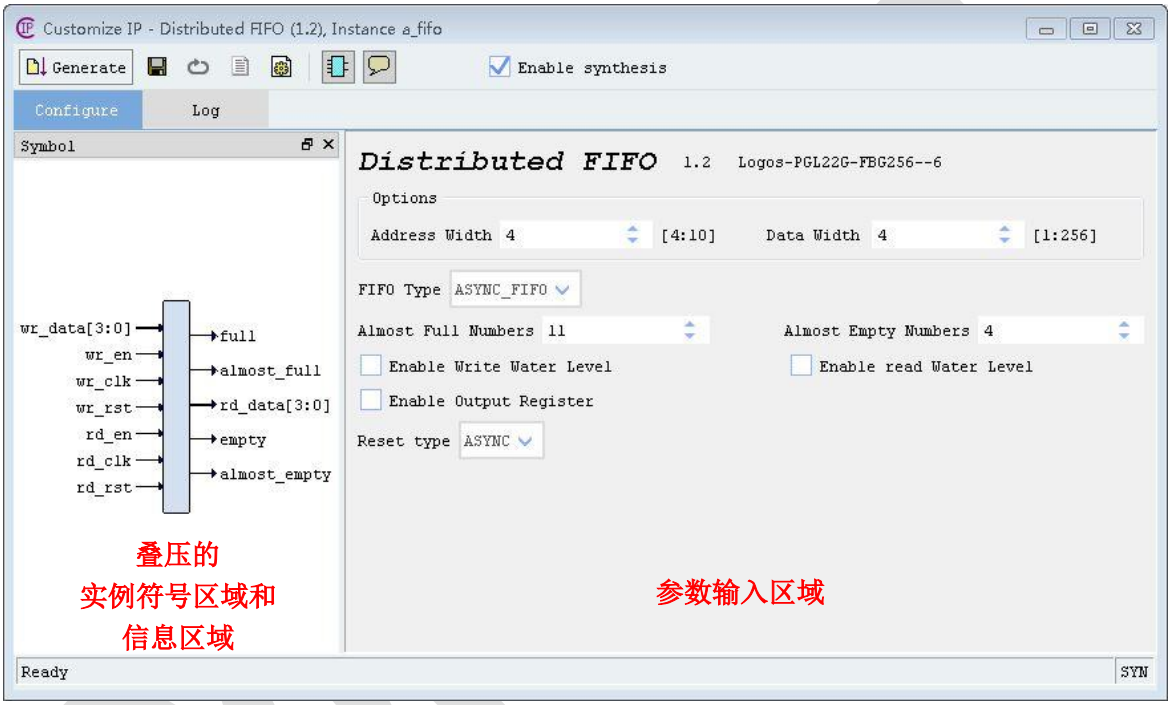
5. 编程参考

对于 IP 开发者来说，有两处可能需要使用 Tcl 语言来进行编程：一个是描述用户界面，一个是在生成 IP 实例时执行一些额外的操作。后者可用的过程较少，是前者的一个子集，主要是去除了那些与图形界面相关的过程。

使用者需要正确安装 Tcl，并设置环境变量 *TCL_LIBRARY* 以使得可以找到 Tcl 的初始化文件 init.tcl。否则，仍然可以使用 Tcl 的大部分功能，但是有某些语句(比如 clock)可能无法识别。

5.1 用户界面描述文件

在 IP 参数配置界面中，IP 开发者可以预先定义好图形界面以方便 IP 使用者理解和操作，这里的图形界面包括三个主要区域：参数输入(input)、实例符号(symbol)和提示信息(information)，后两者不是必需的，可以关闭掉，参见下图。



这些界面的描述都放置在 Tcl 文件中，这个文件的入口在模型索引文件 *index.xml* 中的***<ui>*** 元素中定义，如果没有定义则缺省的入口文件名为 *ui.tcl*，IP 开发者也可以根据 Tcl 语法将这些描述分散在多个文件和过程中。

用户界面描述文件是与模型索引文件中定义的参数配合使用的，它的作用有以下几个：

- 描述了模型参数在图形界面中的表现形式；
- 描述了模型参数之间的依赖关系，并在一个参数改动后进行检查对其它的参数进行相应改动；
- 在生成 IP 实例前对参数进行检查；
- 显示 IP 实例的 symbol；
- 显示当前配置的一些信息或警告；

软件系统预定义了一系列 Tcl 过程，作为这些描述的基础和控制流程的入口，下面这段代码显示了一个基本的用户界面描述文件的框架。

```
# 这是一个用户界面描述文件

proc ui::init {} {
    # 图形界面和 symbol 的描述；参数的初始化等；
}
    ...
proc ui::update { args } {
    # 当参数发生变化时，处理关联的参数并检查有效性；
}

proc ui::check {} {
    # 在保存之前，做参数等的检查；
}
```

这些系统过程都定义在命名空间 ***ui*** 中，即过程名的前缀都为 ***ui::***，IP 开发者要注意避免与这些过程名发生冲突。这些系统过程的具体说明见以下章节。

在这些过程的定义采用了典型的 Tcl 语言风格，使用单减号-开头的一组参数项，比如：

```
ui::addHBox -name hbox1 -parent topVBox
```

我们会提供一些关于参数和返回值定义的表格，在这些表格中，参数名前面的单减号被省略了；如果一个参数名有简写则会在逗号后显示，如果没有显示则表示没有简写；如果参数名后面有一个星号，则表示该参数是必需的，否则是可选的；如果一个过程没有定义参数或返回值则在表格中不再显示。

在编写用户界面描述文件时，开发者需要注意对象的名称，这里的对象包括窗口布局、窗口组件、模型参数和实例符号的管脚。它们分属三个命名空间：位于参数区域的布局和组件对象属于一个命名空间，而参数则往往与所关联的组件同名；位于信息区域的布局和组件对象属于一个命名空间；位于实例符号区域的管脚对象属于一个命名空间。当然，为了避免潜在的冲突，给所有的对象都起一个独特的名字也是一个好方法。

注意，本章节中的一些过程或参数标识为已废弃，这说明在代码中最好不要使用这些过程或参数，但在当前的软件版本中仍然是兼容的。

5.2 控制流程的入口

本节声明的这些系统过程都需要由 IP 开发者来实现，但不是必需的。

- ***ui::init***

初始化过程。主要用来定义各参数的图形界面，以及 symbol 的各管脚等。

ui::init		
参数	**use_defaults**	初始化时是否应使用缺省配置，由软件系统自动填入；通常地，当一个 IP 实例是新创建打开的，或者是处于 IP 模型的预览状态，则本参数被置为 1；当一个 IP 实例是已经配置并保存过参数的，则本参数被置为 0；
参见	**ui::update**	

软件系统在装载一个 IP 模块时，会自动执行用户界面描述文件并调用系统过程 ***ui::init***。根据 Tcl 语法，在执行一个 Tcl 文件时，散放在任何过程之外的语句会被执行，因此严格地讲，本过程不是必需的，但是将这些描述语句归于本过程中可以使这个描述更加整洁从而方便维护。

- ***ui::update***

参数的更新与检查。当一个参数发生变化时，会自动调用本过程，IP 开发者需要检查当前的参数值并更新有依赖关系的其它参数。如果因为编程错误导致递归依赖，则软件系统会报错从而避免系统崩溃。在本过程中，也可以根据当前的参数设置，修改 symbol 的管脚状态，或者给出更多的相关信息，比如所占芯片资源的估算等。

ui::update		
参数	**args**	刚发生变化的参数列表，由软件系统自动填入；
参见	**ui::init**	

需要特别注意的是，***ui::update*** 是由系统在感知有参数变化时自动调用的，以使各参数调整到一个合理状态，最好不要显式地去调用，特别是在初始化过程 ***ui::init*** 中，以免它会干扰已配置保存好的参数值。

- ***ui::check***

在保存一个 IP 实例前，可以对所有参数做最后的检查，在此期间如果发现有错误，则可以通过打印报错信息、弹出警告对话框等形式提醒使用者并中断操作。如果本过程没有定义，则不经检查而继续执行。

ui::check		
参数	**is_new_inst**	当一个 IP 实例是新创建打开的，则本参数被置为 1，否则本参数被置为 0；
返回值	**1**	参数正确，可以进行生成操作；
	0	配置错误，不能进行生成操作；
参见	**ui::popupMessage**	

需要注意的是，虽然对于新创建的 IP 实例也会调用 ***ui::check***，但是不建议用来在其中对参数进行初始化性质的修正。可选参数 *is_new_inst* 可以用来区分是否为新创建的 IP 实例的首次调用，可以如下所示来避免抛出不必要的警告。

```
proc ui::check { is_new_inst } {
    # 是新 IP 实例，直接返回
    if { $is_new_inst } {
        return 1
    }

    # 在保存之前，做参数等的检查
}
```

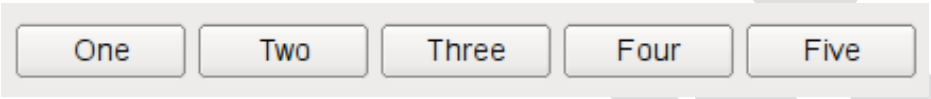
5.3 图形界面的布局

在设计一个图形界面的时候首先要考虑它的布局(layout)。布局用来定义摆放窗口组件(widget)的方式，每个组件都必须给定一个合适的尺寸和位置，使用布局则可以通过布局管理来根据实际情况自动摆放组件。一个布局可以放置在一个组件中，也可以放置在另一个布局中以实现更加复杂的组合。布局通常有三种类型：水平布局、垂直布局、网格布局。

本节声明的这些系统过程用于在输入区域或信息区域中生成一个布局，这些布局一旦生成后就不能再删除或修改了。

- ***ui::addHBox***

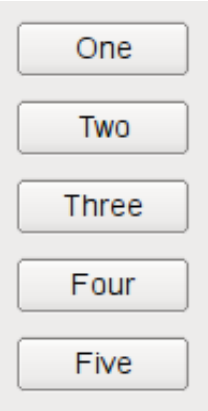
生成一个水平布局(horizontal box, HBox)。如下图所示，若干窗口组件被摆放在一行中。



ui::addHBox		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p	字符串值，缺省为顶层布局； 父对象的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	spacing,s	整数值； 在本布局中，各窗口组件的间距；
返回值	本对象的名称	

- ***ui::addVBox***

  生成一个垂直布局(vertical box, VBox)。如下图所示，若干窗口组件被摆放在一列中。



ui::addVBox		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p	字符串值，缺省为顶层布局； 父对象的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	spacing,s	整数值； 在本布局中，各窗口组件的间距；
返回值	本对象的名称	

- ***ui::addGrid***

生成一个网格布局(Grid)。如下图所示，若干窗口组件被摆放在一个阵列中。

```
label-at-(0,0)          label-at-(0,1)          label-at-(0,2)
label-at-(1,0)          label-at-(1,1)          label-at-(1,2)
        label-at-(2,0)-span(2,2)                label-at-(2,2)
label-at-(4,0)          label-at-(4,1)          label-at-(4,2)
```

ui::addGrid		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在父对象网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在父对象网格布局中列的扩展；
	hspacing,hs	整数值； 在本布局中，各窗口组件的水平间距；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p	字符串值，缺省为顶层布局； 父对象的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在父对象网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在父对象网格布局中行的扩展；
	vspacing,vs	整数值； 在本布局中，各窗口组件的垂直间距；
返回值	本对象的名称	
参见	**ui::setGridColumnStretch, ui::setGridRowStretch**	

- ***ui::setGridColumnStretch***

设置网格布局中各列的伸缩系数。伸缩系数缺省为 0，表示列的宽度由其中的控件自动决定；更大的伸缩系数表示一个列可以占有更大的宽度空间。

ui::setGridColumnStretch		
参数	**column,c**	整数值，大于等于 0； 指定网格布局中的一个列号；如果不指定则表示所有列；
	name,n*	字符串值； 网格布局的名称；
	stretch,s	整数值，缺省为 0； 伸缩系数；
参见	**ui::addGrid, ui::setGridRowStretch**	

- ***ui::setGridRowStretch***

设置网格布局中各行的伸缩系数。伸缩系数缺省为 0，表示行的高度由其中的控件自动决定；更大的伸缩系数表示一个行可以占有更大的高度空间。

ui::setGridGridStretch		
参数	**name,n***	字符串值； 网格布局的名称；
	row,r	整数值，大于等于 0； 指定网格布局中的一个行号；如果不指定则表示所有行；
	stretch,s	整数值，缺省为 0； 伸缩系数；
参见	**ui::addGrid, ui::setGridColumnStretch**	

5.4 如何定义和操作模型参数

IP Compiler 的主要功能就是配置或控制 IP 参数的值，参数共有 7 种类型，分别是：字符串、整数、浮点数、布尔、逻辑向量、枚举和列表。

这些参数通常都是与某个窗口组件相关联的，组件显示的值应该与参数的值保持一致。要特别注意窗口组件的两个重要属性：名称和类型。这些组件是通过名称与一个参数关联起来的，在模型索引文件中定义的参数名应该与组件名一致，这样当窗口组件的状态改变时其所对应的参数值就会自动更新，而参数定义中的缺省值等属性也可以自动赋予这些窗口组件。

一个参数也可以不指定与之关联的组件，而会在配置窗口参数输入区域的最下方自动生成在一个组件列表中。如果该参数在模型索引文件中定义为不可见，则它不会出现在配置窗口中，但还是可以对其通过编程进行读写值等操作。

另一个重要的属性就是类型，一个组件的类型要和它所关联的参数的类型相匹配，否则系统会报错。正确的类型匹配关系见下表：

组件类型	参数类型
文本输入框 line-edit	字符串、整数、浮点数、布尔、逻辑向量
文本编辑器 text-edit	字符串
勾选按钮 check-button	布尔
互斥选项按钮 radio-button	枚举
下拉式列表 combo-box	枚举
列表 list	列表
整数输入框 spin-box	整数
浮点数输入框 double-spin-box	浮点数

不同类型的参数有着不同的特点和操作方法。

- **字符串**

字符串是最通用的类型，可以表示任意信息，并可以通过设置一个正则表达式来限制其格式；单行的字符串可以使用文本输入框(line-edit)组件来表示，多行的文本则可以使用文本编辑器(text-edit)组件；主要相关过程有：*ui::addLineEdit*、*ui::addTextEdit*、*ui::setText*、*ui::setValidator*、*ui::setValue*、*ui::getValue* 等。

- **整数**

通常使用整数输入框(spin-box)组件来表示，也可以使用一个设置了限定表达式的文本输入框(line-edit)组件来表示；主要相关过程有：*ui::addSpinBox*、*ui::addLineEdit*、*ui::setValidator*、*ui::setValue*、*ui::getValue* 等。

- **浮点数**

通常使用浮点数输入框(double-spin-box)组件来表示，也可以使用一个设置了限定表达式的文本输入框(line-edit)组件来表示；主要相关过程有：*ui::addDoubleSpinBox*、*ui::addLineEdit*、*ui::setValidator*、*ui::setValue*、*ui::getValue* 等。

- **布尔**

通常使用勾选按钮(check-button)组件来表示，也可以使用一个设置了限定表达式的文本输入框(line-edit)组件来表示；主要相关过程有：*ui::addCheckBox*、*addLineEdit*、*ui::setValidator*、*ui::setValue*、*ui::getValue* 等。

- **逻辑向量**

使用文本输入框(line-edit)组件来表示，缺省用二进制表示，如果向量值仅含有 0 和 1 也可以指定用八进制或十六进制来显示，；主要相关过程有： *ui::addLineEdit*、*ui::setValue*、*ui::getValue* 等。

- **枚举**

枚举是一种多选一的参数类型，其初始选项在模型索引文件中定义，并且可以通过编程进行修改；使用下拉式列表(combo-box)组件或互斥选项按钮(radio-button)组件来表示；主要相关过程有：*ui::addCombox*、*ui::addRadioButton*、*ui::setItems*、*ui::setText*、*ui::setValue*、*ui::getValue* 等。

- **列表**

列表是一种类似于枚举的参数类型，不同之处在于它可以是多选多的，其初始选项在模型索引文件中定义，并且可以通过编程进行修改；使用列表(list)组件来表示；主要相关过程有：*ui::addList*、*ui::setItems*、*ui::setText*、*ui::setValue*、*ui::getValue* 等。

5.5 窗口组件

窗口组件或者称之为部件，是用来显示或者与使用者交互的对象，有的可以接受键盘或鼠标的输入。一个组件通常必须放置在一个布局中，而不能直接放在另一个组件中(stacked-page 除外)。在本软件系统中，窗口组件可以分为以下几种类型：输入类组件、显示类组件、容器类组件等。

本节声明的这些系统过程用于生成一个窗口组件。

5.5.1 输入类组件

输入类窗口组件用于给使用者提供一个输入信息的接口，在本软件系统中，这是最重要的一类组件，因为对于 IP Compiler 来说，输入类组件主要用来配置或控制 IP 参数的值，所以这些组件只能在 IP 参数配置界面的参数输入区域中生成(只读的 text-edit 除外)。

- ***ui::addLineEdit***

生成一个文本输入框，最为常用的输入组件之一，可以关联字符串、整数、浮点数、布尔、逻辑向量等类型的参数，参见下图。

ui::addLineEdit		
参数	**align**	枚举值，可选项为 ***left, center, right***，缺省为 ***left*** 指定文字的水平对齐方向
	column,c	整数值，大于等于 0，缺省为 0 表示在一个网格布局中的(起始)列号
	columnspan,cs	整数值，大于等于 1，缺省为 1 表示在一个网格布局中列的扩展
	len,l	整数值，大于 0 表示最多允许输入的字符个数，缺省为 32767
	logicbase,lb	枚举值，可选项为 ***2, 8, 16***，缺省为 ***2*** 指定显示逻辑向量时所用的进制
	name,n	字符串值 本对象的名称
	parent,p*	字符串值 所属布局的名称
	row,r	整数值，大于等于 0，缺省为 0 表示在一个网格布局中的(起始)行号
	rowspan,rs	整数值，大于等于 1，缺省为 1 表示在一个网格布局中行的扩展
	text,t	字符串值 初始文本
	tip	字符串值 提供一小段关于本对象的提示
	validator	字符串值，可以提供一个正则表达式，以限制输入文字的格式； 若与一个逻辑向量参数相关联，则可以自动设置输入约束
返回值	本对象的名称	
参见	**ui::addTextEdit, ui::setText, ui::setValidator, ui::setTextLen**	

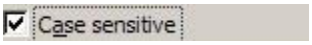
- ***ui::addTextEdit***

生成一个多行文本编辑器，可以关联字符串类型的参数，在将编辑器设为只读后也可以用来仅显示一段文字，参见下图。

ui::addTextEdit		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象，并且自动设为只读；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	readonly,ro	开关值；打开时则只读，缺省为可编辑；当 **info** 开关打开时，会自动设为只读；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	text,t	字符串值； 初始文本；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::addLineEdit, ui::setText**	

- ***ui::addCheckBox***

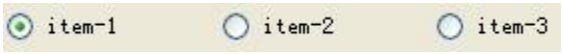
  生成一个勾选按钮，可以关联布尔类型的参数，参见下图。



ui::addCheckBox		
参数	**checked**	开关值； 初始的勾选状态；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	text,t	字符串值； 在按钮旁边显示的文本；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::setText**	

- ***ui::addRadioButton***

生成一个互斥选项按钮，这些按钮可以放置在一个组中以构成一个选项集，可以关联枚举类型的参数，参见下图。



ui::addRadioButton		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	group,g*	字符串值； 按钮组的名称，属于同一组的按钮是互斥的；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	text,t	字符串值； 在按钮旁边显示的文本；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::addComboBox, ui::setItems, ui::setText, ui::count**	

- ***ui::addComboBox***

生成一个下拉式列表，可以从该列表中选取一项，可以关联枚举类型的参数，参见下图。缺省状态为选中列表中的第一项。

ui::addComboBox		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	text,t	字符串列表； 在选项列表中显示的文本；如果本对象与一个枚举类型的参数相关联，则其缺省的文本就是参数选项的值，但如果希望用一种更可读的形式来显示文本时，就会用到本选项，例如：参数选项为 greater/equal/less，可以通过设置本选项使其显示为>/=/<，而参数选项本身并不会被修改；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::addList, ui::addRadioButton, ui::setItems, ui::setText, ui::count**	

- ***ui::addList***

  生成一个可滚动列表，可以从该列表中选取多项，可以关联列表类型的参数，参见下图。

ui::addList		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	mulsel	开关值； 打开时表示可以多选，缺省为只能单选；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	text,t	字符串列表； 在选项列表中显示的文本；在选项列表中显示的文本；如果本对象与一个列表类型的参数相关联，则其缺省的文本就是参数选项的值，但如果希望用一种更可读的形式来显示文本时，就会用到本选项，例如：参数选项为 greater/equal/less，可以通过设置本选项使其显示为>/=/<，而参数选项本身并不会被修改；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::addComboBox, ui::setItems, ui::setText ,ui::count**	

- ***ui::addSpinBox***

  生成一个整数输入框，可以关联整数类型的参数，参见下图。

ui::addSpinBox		
参数	**align**	枚举值，可选项为 ***left, center, right***，缺省为 ***left***； 指定数值(文字)的水平对齐方向；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	intbase,ib	整数值，缺省为 10； 指定显示整数量时所用的进制；
	max	整数值； 表示可以输入的最大值，缺省为 2147483647；
	min	整数值； 表示可以输入的最小值，缺省为-2147483648；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	prefix	字符串值； 在数值前面显示的字符串，比如一个货币符号；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	step,s	整数值； 每点击一次箭头时，数值所变化的步长，缺省为 1；
	suffix	字符串值； 在数值后面显示的字符串，比如一个百分号；
	tip	字符串值； 提供一小段关于本对象的提示；
	validator	字符串值； 可以提供一个正则表达式，以限制输入文字的格式；
	value,v	整数值； 初始值，缺省为 0；
返回值	本对象的名称	
参见	**ui::addDoubleSpinBox**	

- ***ui::addDoubleSpinBox***

  生成一个双精度浮点数输入框，可以关联浮点类型的参数，参见下图。

ui::addDoubleSpinBox		
参数	**align**	枚举值，可选项为 ***left, center, right***，缺省为 ***left***；指定数值(文字)的水平对齐方向；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	decimal	整数值； 数值的显示精度，缺省为小数点后 2 位；
	max	浮点数值； 表示可以输入的最大值，缺省为 1.79769e+308；
	min	浮点数值； 表示可以输入的最小值，缺省为-1.79769e+308；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	prefix	字符串值； 在数值前面显示的字符串，比如一个货币符号；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	step,s	浮点数值； 每点击一次箭头时，数值所变化的步长，缺省为 1.0；
	suffix	字符串值； 在数值后面显示的字符串，比如一个百分号；
	tip	字符串值； 提供一小段关于本对象的提示；
	validator	字符串值； 可以提供一个正则表达式，以限制输入文字的格式；
	value,v	浮点数值； 初始值，缺省为 0.0；
返回值	本对象的名称	
参见	**ui::addSpinBox**	

5.5.2 显示类组件

显示类窗口组件用于在图形界面上显示一些信息或者进行一些装饰或分割，以帮助用户识别和使用那些输入类组件，这些组件可以在 IP 参数配置界面的参数输入区域或信息区域中生成。

- ***ui::addLabel***

生成一个文本标签，是最为常用的显示组件，参见下图。也可以关联任意类型的参数，但仅能显示参数的值，而不能做出修改。

ui::addLabel		
参数	**align**	枚举值，可选项为 ***left, center, right***，缺省为 ***left***； 指定文字的水平对齐方向；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	text,t	字符串值； 显示的文本；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::setText**	

- ***ui::addImage***

生成一个图片标签，并装载一个指定的图片文件，支持的文件格式有 BMP、GIF、JPG、PNG、XPM 等。

ui::addImage		
参数	**align**	枚举值，可选项为 ***left, center, right***，缺省为 ***left***； 指定图片的水平对齐方向；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	file,f*	字符串值； 一个图片文件名，可以是 IP 单元根目录的相对路径；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::setImage**	

- ***ui::addHSep***

  生成一条水平分隔线，用来分割其它组件。

ui::addHSep		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；

- ***ui::addVSep***

  生成一条垂直分隔线，用来分割其它组件。

ui::addVSep		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；

- ***ui::addSpacer***

在布局中增加一个可拉伸的空间，以便使其它组件自动占用一个适当的空间。

ui::addSpacer		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	parent,p*	字符串值； 所属水平或垂直布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
参见	**ui::addSpacing**	

- ***ui::addSpacing***

在布局中增加一个不可拉伸的固定空间，手工指定一个空间以便分隔其它组件。

ui::addSpacing		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	parent,p*	字符串值； 所属水平或垂直布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	size,s	整数值，缺省为 0； 指定一个固定的空间；
参见	**ui::addSpacer**	

5.5.3 容器类组件

容器类窗口组件主要用于容纳其它组件，以帮助用户识别和使用，这些组件可以在 IP 参数配置界面的参数输入区域或信息区域中生成。

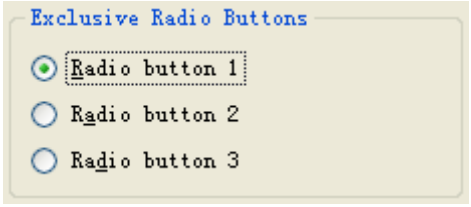
- ***ui::addFrame***

生成一个框架，常用来包含一组其它组件，这个区域的周围可以有一个矩形的边界，参见下图。

ui::addFrame		
参数	**box**	开关值； 打开时表示有一个矩形的边界，缺省为无；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::addGroupBox**	

- ***ui::addGroupBox***

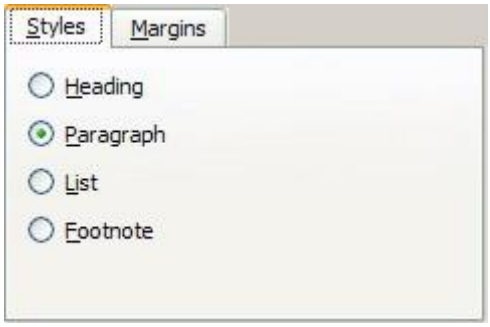
  生成一个有边界和标题的框架，常用来包含一组其它组件，参见下图。



ui::addGroupBox		
参数	**align**	枚举值，可选项为 ***left, center, right***，缺省为 ***left***； 指定标题的水平对齐方向；
	column,c	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	title	字符串值； 标题的文字；
	tip	字符串值； 提供一小段关于本对象的提示；
返回值	本对象的名称	
参见	**ui::addFrame**	

- ***ui::addStackedWidget***

生成一个分页的容器，每一页都可以放置其它的窗口组件，每次只显示一页，通常还有一个标签栏用来切换到某一页，参见下图。



ui::addStackedWidget		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	info	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	tab	枚举值，可选项为 ***left, right, top, bottom, hide***，缺省为 ***top***； 指定标签栏的水平对齐方向，***hide*** 表示不显示标签栏；
返回值	本对象的名称	
参见	**ui::addStackedPage, ui::count, ui::getCurrentIndex, ui::setCurrentIndex**	

- ***ui::addStackedPage***

在一个分页的容器中生成一个新页面，用来放置其它的窗口组件。

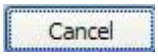
ui::addStackedPage		
参数	**info**	开关值； 打开时表示在信息区域生成本对象；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属页容器的名称；
	tip	字符串值； 提供一小段关于本对象的提示；
	title	字符串值； 页标签的标题；
返回值	本页在页容器中的索引值，从 0 开始；	
参见	**ui::addStackedWidget**	

5.5.4 其它组件

还有一些其它类型的组件，比如按钮(button)，可以用来进行辅助性的操作

- ***ui::addButton***

生成一个普通按钮，点击该按钮可以执行一个指定的命令(**-command** 选项)或者选择文件(**-selfile** 选项)和目录(**-seldir** 选项)，参见下图。



ui::addButton		
参数	**column,c**	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)列号；
	columnspan,cs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中列的扩展；
	command,cmd	字符串值；指定在本按钮被按下时激活一个 Tcl 命令，这个命令所对应的过程通常在模型索引文件中定义；
	name,n	字符串值； 本对象的名称；
	parent,p*	字符串值； 所属布局的名称；
	row,r	整数值，大于等于 0，缺省为 0； 表示在一个网格布局中的(起始)行号；
	rowspan,rs	整数值，大于等于 1，缺省为 1； 表示在一个网格布局中行的扩展；
	seldir	开关值；打开时在会在按钮被按下时打开一个目录选择对话框，并将结果填入 **to** 选项指定的一个文本输入框中；
	selfile	开关值；打开时在会在按钮被按下时打开一个文件选择对话框，并将结果填入 **to** 选项指定的一个文本输入框中；
	nativesep,native	开关值；与选项 **seldir** 和 **selfile** 配合使用；缺省为否； 是否将路径中的分隔符转换为本地操作系统所用的； 在 Windows 系统上为'\'，在 Linux 等系统上为'/'；
	text,t	字符串值； 在按钮上显示的文本；
	tip	字符串值； 提供一小段关于本对象的提示；
	to	字符串值；与选项 **seldir** 和 **selfile** 配合使用； 指定一个文本输入框的名称；
返回值	本对象的名称	
参见	**ui::setText**	

5.6 对象的控制操作

本节声明的这些系统过程用于在配置区域、符号区域和信息区域中控制一个组件或符号管脚的状态或属性。

- ***ui::hide***

隐藏一个窗口组件或实例符号管脚。

ui::hide		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件或实例符号管脚的名称；
	page	字符串值； 分页容器中的一个页面；
参见	**ui::show, ui::isVisible, ui::hideSymbol, ui::hidePin, ui::addStackedPage**	

- ***ui::show***

显示一个窗口组件或实例符号管脚。

ui::show		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件或实例符号管脚的名称；
	page	字符串值； 分页容器中的一个页面；
参见	**ui::hide, ui::isVisible, ui::showSymbol, ui::showPin, ui::addStackedPage**	

- ***ui::isVisible***

检查一个窗口组件是否可见。

ui::isVisible		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件的名称；
	page	字符串值； 分页容器中的一个页面；
返回值	可见时为 1，不可见时为 0；	
参见	**ui::hide, ui::show, ui::addStackedPage**	

- ***ui::disable***

使一个窗口组件无效，即使其无法接收键盘和鼠标输入，并且显示颜色会变灰。

ui::disable		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件或实例符号管脚的名称；
参见	**ui::enable, ui::isEnabled**	

- ***ui::enable***

使一个窗口组件有效，即使其可以接收键盘和鼠标输入，并且显示颜色会变正常。

ui::enable		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件或实例符号管脚的名称；
参见	**ui::disable, ui::isEnabled**	

- ***ui::isEnabled***

检查一个窗口组件是否有效。

ui::isEnabled		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件的名称；
返回值	有效时为 1，无效时为 0；	
参见	**ui::disable, ui::enable**	

- ***ui::setText***

设置一个窗口组件的显示文本。这些组件包括文本标签（label）、文本输入框（line-edit）、文本编辑器(text-edit)、普通按钮(button)、勾选按钮(check-button)、下拉式列表(combo-box)、列表（list）、互斥按钮（radio-button）组等。

ui::setText		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件的名称；
	text,t	字符串值； 新的文本(列表)，缺省为空；
参见	**ui::getText，ui::setValue，ui::setItems，ui::setValidator**	

这里有几点需要特别说明一下：对于文本标签(label、文本输入框(line-edit)、文本编辑器(text-edit)、普通按钮(button) 、勾选按钮(check-button)组件，如果新文本是一个列表，则只有其第一个元素字符串会被使用；对于文本输入框(line-edit)组件，如果它已经设置了限制格式，则新文本只有在符合格式的情况下才会被设置；对于下拉式列表(combo-box)或列表(list)组件，如果新文本列表的长度小于可选项的个数，则组件的表项会用可选项来表示，如果新文本列表的长度大于可选项的个数则会被截断；对于互斥按钮(radio-button)组，新文本列表的长度不能少于按钮的个数。

- ***ui::getText***

获取一个窗口组件的显示文本。这些组件包括文本标签（label）、文本输入框（line-edit）、文本编辑器(text-edit)、普通按钮(button)、勾选按钮(check-button)、下拉式列表(combo-box)、整数输入框(spin-box)、浮点数输入框(double-spin-box)、列表(list)、互斥按钮(radio-button)组等。

ui::getText		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件的名称；
返回值	当前文本	
参见	**ui::setText，ui::setValue**	

- ***ui::setImage***

设置一个图片标签，并装载一个指定的图片文件，支持的文件格式有 BMP、GIF、JPG、PNG、XPM 等。

ui::setImage		
参数	**file,f***	字符串值； 一个图片文件名，可以是 IP 单元根目录的相对路径；
	info	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件的名称；

- ***ui::setTip***

对大多数窗口组件都可以设置一小段关于本对象的提示。

ui::setTip		
参数	**info**	开关值； 打开时表示对象位于信息区域；
	name,n*	字符串值； 一个窗口组件的名称；
	tip	字符串值； 新的提示文本，缺省为无；

- ***ui::setValidator***

设置文本输入框(line-edit)的输入格式限制。

ui::setValidator		
参数	**name,n***	字符串值； 本对象的名称；
	validator	字符串值；可以提供一个正则表达式，以限制输入文字的格式， 未指定时则表示清除所有限制；
参见	**ui::addLineEdit, ui::setTextLen**	

- ***ui::setTextLen***

设置文本输入框(line-edit)可输入字符的长度。

ui::setTextLen		
参数	**name,n***	字符串值； 本对象的名称；
	len,I	整数值；可以输入的字符个数，如果小于等于 0，则会被置为缺 省值 32767；
参见	**ui:addLineEdit, ui::setTextValidator**	

- ***ui::count***

获得一个可下拉式列表(combo-box)或可滚动列表(list)中的可选项的数量，也可以获得一个页容器(stacked-widget)中的总页数，或者一个互斥选项按钮(radio-button)组中的按钮数量。

ui::count		
参数	**name,n***	字符串值； 一个窗口组件的名称；
返回值	数量	

- ***ui::getCurrentIndex***

获得一个页容器(stacked-widget)中的当前页索引号。

ui::getCurrentIndex		
参数	**name,n***	字符串值； 一个页容器组件的名称；
返回值	数量	
参见	**ui::addStackedWidget, ui::setCurrentIndex**	

- ***ui::setCurrentIndex***

设置一个页容器(stacked-widget)中的当前页。

ui::setCurrentIndex		
参数	**index,i***	整数值； 一个页编号；
	name,n*	字符串值； 一个页容器组件的名称；
返回值	数量	
参见	**ui::addStackedWidget, ui::getCurrentIndex**	

- ***ui::setColor***

设置一个组件的正常状态文本颜色，针对的是一个可以显示或输入文本的窗口组件，通常在我们可以改变它的颜色来提示错误。禁用状态文本颜色不受影响。

ui::setColor		
参数	**color,clr***	颜色值；可以是一个颜色的名字，比如 red 或者 blue 等，也可以是一个十六位的 RGB 值，其格式可以为以下几种：#RGB、#RRGGBB、#RRRGGGBBB、#RRRRGGGGBBBB；
	name,n*	字符串值； 一个组件的名称；
参见	**ui::resetColor**	

- ***ui::resetColor***

恢复一个组件或所有组件的正常状态文本颜色，通常是黑色。禁用状态文本颜色不受影响。

ui::resetColor		
参数	**all,a**	开关值；表示所有组件的文本颜色都将被恢复，此时 ***name*** 参数会被忽略；
	name,n	字符串值； 一个组件的名称；
参见	**ui::setColor**	

5.7 参数值的获取与设定

本节声明的这些系统过程用于获取和设定一个参数或输入组件的值。这些参数都是在模型索引文件中定义的。

- ***ui::getValue***

获得一个参数的当前值；如果该参数没有在模型索引文件中定义，并且存在一个同名的窗口组件，则返回该组件的当前值。

ui::getValue		
参数	**name,n***	字符串值； 一个参数或窗口组件的名称；
	keepzeros,z	开关值； 对于一个浮点参数取值时，如果打开本开关，则小数部分尾部的 0 会被保留；缺省会丢掉尾部的 0；
返回值	参数或组件的值或值的列表	
参见	**ui::setValue，ui::getText，<param>，<param_list>**	

- ***ui::setValue***

设置一个参数的当前值；如果该参数没有在模型索引文件中定义，但存在一个同名的窗口组件，则设置该组件的当前值/显示。

ui::setValue		
参数	**name,n***	字符串值； 一个参数或窗口组件的名称；
	value,v*	字符串列表； 参数或组件的新值；
参见	**ui::getValue，ui::setText**	

- ***ui::getDefaultValue***

获得一个参数的缺省值。

ui::getDefaultValue		
参数	**name,n***	字符串值； 一个参数的名称；
返回值	参数的缺省值或值的列表	

- ***ui::getMaxValue***

获得一个整数或浮点数类型参数的最大值，如果该参数没有在模型索引文件中定义，并且存在一个同名的整数输入框(spin-box)或浮点数输入框(double-spin-box)组件，则返回该组件的最大值。

ui::getMaxValue		
参数	**name,n***	字符串值； 一个参数或窗口组件的名称；
返回值	参数的最大值	
参见	**ui::setMaxValue**	

- ***ui::setMaxValue***

设置一个整数或浮点数类型参数的最大值，如果该参数没有在模型索引文件中定义，并且存在一个同名的整数输入框(spin-box)或浮点数输入框(double-spin-box)组件，则设置该组件的最大值。

ui::setMaxValue		
参数	**name,n***	字符串值； 一个参数或窗口组件的名称；
	max,v*	最大值；
参见	**ui::getMaxValue**	

- ***ui::getMinValue***

获得一个整数或浮点数类型参数的最小值，如果该参数没有在模型索引文件中定义，并且存在一个同名的整数输入框(spin-box)或浮点数输入框(double-spin-box)组件，则返回该组件的最小值。

ui::getMinValue		
参数	**name,n***	字符串值； 一个参数或窗口组件的名称；
返回值	参数的最小值	
参见	**ui::setMinValue**	

- ***ui::setMinValue***

设置一个整数或浮点数类型参数的最小值，如果该参数没有在模型索引文件中定义，并且存在一个同名的整数输入框(spin-box)或浮点数输入框(double-spin-box)组件，则设置该组件的最小值。

ui::setMinValue		
参数	**name,n***	字符串值； 一个参数或窗口组件的名称；
	min,v*	最小值；
参见	**ui::getMinValue**	

- ***ui::setItems***

设置一个枚举或列表类型参数的可选项(可选项的初始值是在模型索引文件中定义的)，该参数的当前值也会被重置；如果有对应的下拉式列表(combo-box)或列表(list)组件，则其表项的文本也同时被修改；如果有对应的互斥按钮(radio-button)组，则新的可选项个数必须与按钮的个数相同，并且设置新的可选项后按钮的文本也同时被修改。

ui::setItems		
参数	**name,n***	字符串值； 一个窗口组件的名称；
	items*	字符串值；新的选项值/文本；对于一个列表类型的变量，需要用{*}将其展开；
参见	**uis, ui::setText, ui::addCombox, ui::addList, ui::addRadioButton**	

- ***ui::getItems***

获得一个枚举或列表类型参数的可选项。

ui::getItems		
参数	**name,n***	字符串值； 一个窗口组件的名称；
返回值	可选项列表	
参见	**ui::setItems**	

5.8 IP 基本属性的获取

- ***ui::getModelName***

  返回当前所用 IP 模块的名称。

ui::getModelName	
返回值	IP 模块的名称

- ***ui::getModelVersion***

  返回当前所用 IP 模块的版本。

ui::getModelVersion	
返回值	IP 模块的版本

- ***ui::getModelPath***

  返回当前所用 IP 模块在磁盘中的路径(目录)。

ui::getModelPath		
参数	**nativesep,native**	开关值；缺省为否； 是否将路径中的分隔符转换为本地操作系统所用的； 在 Windows 系统上为'\'，在 Linux 等系统上为'/'；
返回值	路径名	
参见	**ui::getInstPath**	

- ***ui::getInstName***

  返回当前 IP 实例的名称。

ui::getInstName	
返回值	IP 实例的名称

- ***ui::getInstPath***

  返回当前 IP 实例在磁盘中的路径(目录)。

ui::getInstPath		
参数	**nativesep,native**	开关值；缺省为否； 是否将路径中的分隔符转换为本地操作系统所用的； 在 Windows 系统上为'\'，在 Linux 等系统上为'/'；
返回值	路径名	
参见	**ui::getModelPath**	

- ***ui::getFamily***

  返回当前 IP 实例所用的 FPGA 系列。

ui::getFamily	
返回值	系列的名称

- ***ui::getDevice***

  返回当前 IP 实例所用的 FPGA 器件。

ui::getDevice		
参数	**original,o**	返回未经映射的原始器件名称；在特定情况下，软件会将一种器件自动变换成另一种，但通常情况下不会这样做；
返回值	器件的名称	

- ***ui::getPackage***

  返回当前 IP 实例所用的 FPGA 封装形式。

ui::getPackage	
返回值	封装形式的名称

- ***ui::getSpeedGrade***

  返回当前实例所用器件的速度等级。

ui::getSpeedGrade	
返回值	速度等级的名称

- ***ui::getInstallPath***

  返回工具的安装路径，即可执行程序所在路径的上一级目录。因为可执行程序通常放在 bin 目录下，所以在正常安装的情况下返回的路径名与安装路径是一致的。

ui::getInstallPath		
参数	**nativesep,native**	开关值；缺省为否； 是否将路径中的分隔符转换为本地操作系统所用的； 在 Windows 系统上为'\'，在 Linux 等系统上为'/'；
返回值	路径名	

- ***ui::getArchPath***

  返回 arch 的安装路径，通常为安装路径下的 arch 子目录。

ui::getArchPath		
参数	**nativesep,native**	开关值；缺省为否； 是否将路径中的分隔符转换为本地操作系统所用的； 在 Windows 系统上为'\'，在 Linux 等系统上为'/'；
返回值	路径名	

- ***ui::getToolVersion***

  返回软件工具的版本。

ui::getToolVersion	
返回值	版本号

5.9封装管脚信息的获取

本节声明的这些系统过程用于从硬件封装规格说明中获取管脚信息。

- ***ui::getPkgPins***

获取当前封装定义的管脚。可以指定 bank 或/和 group，如果没有指定则返回所有管脚。

ui::getPkgPins		
参数	**bank,b**	字符串值； 指定一个 bank 的名称；
	group,g	字符串值； 指定一个 group 的名称；
返回值	管脚名称的列表	

- ***ui::getPkgPinSpec***

获取一个指定管脚的具体规格说明。

ui::getPkgPinSpec		
参数	**name,n***	字符串值； 管脚的名称；
返回值	规格说明的列表，每一项包含一个属性名和一个属性值；	

- ***ui::getPkgPinBank***

获取一个指定管脚所属的 bank 名称。

ui::getPkgPinBank		
参数	**name,n***	字符串值； 管脚的名称；
返回值	bank 的名称	

- ***ui::getPkgPinGroup***

获取一个指定管脚所属的 group 名称。

ui::getPkgPinGroup		
参数	**name,n***	字符串值； 管脚的名称；
返回值	group 的名称	

- ***ui::isPkgPinDQS***

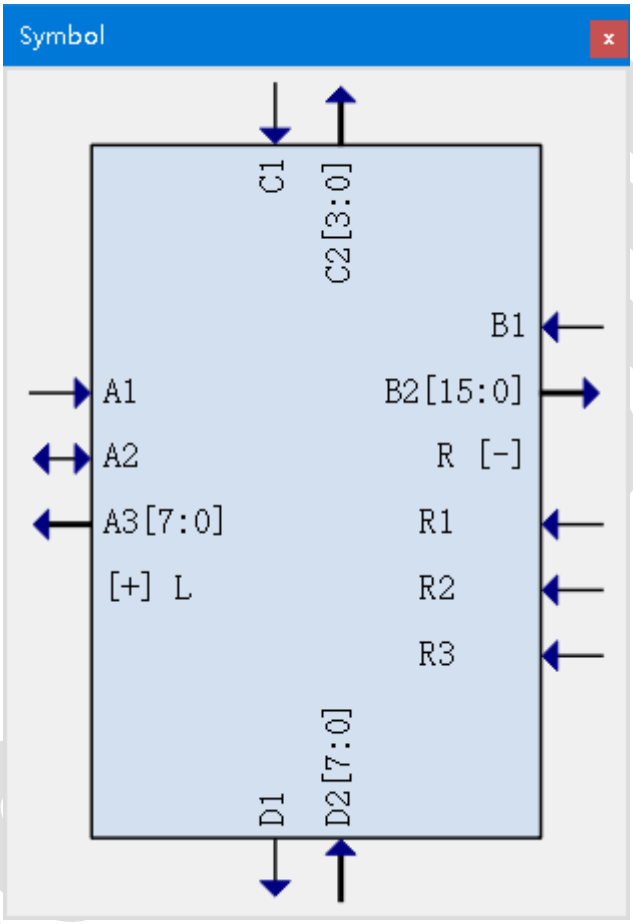
检查一个指定管脚是否为 DQS。

ui::isPkgPinDQS		
参数	**name,n***	字符串值； 管脚的名称；
返回值	是为 1，其它为 0；	

5.10 实例符号及管脚的定义与操作

本节声明的这些系统过程用于控制实例符号区域(Symbol)，并在其中定义和控制符号的管脚。符号区域在缺省状态下是打开的，可以通过编程将其关闭。实例符号区域可以借助于鼠标进行缩放和拖动等操作。

参见下图，实例符号通常有一个矩形外框，管脚可以放置在四条边上，管脚可以定义输入输出方向并用箭头表示出来；如果一个管脚的索引下标的高低值不相同，则认为它是一个是 bus 类型，在图中用粗线表示并且名称后面会显示索引下标的范围。若干个管脚可以放置在一个组中，每个管脚组处都显示有[+]或[-]来提示可以展开或折叠。



- ***ui::hideSymbol***

  隐藏整个实例符号区域以及 *View Symbol* 开关按钮。

- ***ui::showSymbol***

  显示实例符号区域以及 *View Symbol* 开关按钮。

- ***ui::addPin***

在实例符号区域中增加一个管脚。

ui::addPin		
参数	**dir**	枚举值，可选项为 ***input, output, inout***； 表示管脚的输入输出方向，缺省为 ***input***；
	edge,e	枚举值，可选项为 ***left, right, top, bottom***； 表示管脚位于符号的哪条边上，缺省为根据输入输出方向自动确定；如果指定了所属的管脚组，则会跟随组的位置而放置；
	group,g	字符串值； 可以将该管脚放置在一个组中；
	lsb	整数值； 表示 bus 的最低有效位，缺省为 0；
	msb	整数值； 表示 bus 的最高有效位，缺省为 0；
	name,n*	字符串值； 管脚的名称；
	text,t	字符串值；管脚的显示文本，对于 bus 不需包括其有效索引值，缺省为无；如果本参数为空值，则在绘制的时候显示管脚的名称；
返回值	本管脚的名称	
参见	**ui::addPinGroup, ui::addPinSpacing, ui::setPin**	

- ***ui::addPinGroup***

在实例符号区域中增加一个管脚组。管脚组应该在管脚之前生成。如果一个组内的所有管脚都是不可见的，则整个管脚组也是不可见的。

ui::addPinGroup		
参数	**edge,e**	枚举值，可选项为 ***left, right, top, bottom***； 表示管脚组位于符号的哪条边上，缺省为其中所加入的（第一个）管脚所在的位置；
	name,n*	字符串值； 管脚组的名称；
	text,t	字符串值；管脚组的显示文本；如果本参数为空值，则在绘制的时候显示管脚组的名称；
返回值	本管脚组的名称	
参见	**ui::addPin, ui::hidePin**	

- ***ui::addPinSpacing***

在实例符号区域的管脚之间增加一个空白区域，以便在管脚较多时将其分组显示。

ui::addPinSpacing		
参数	**edge,e**	枚举值，可选项为 ***left, right, top, bottom***； 表示位于符号的哪条边上，缺省与前一个管脚相同，或者为 ***left***；
	size,s	整数值，有效范围为 1 至 65535； 表示留下相当于几个管脚的空白，缺省为 1；
参见	**ui::addPin**	

- ***ui::showPin***

显示一个实例符号管脚。

ui::showPin		
参数	**name,n***	字符串值； 一个管脚的名称；
参见	**ui::hidePin**	

- ***ui::hidePin***

隐藏一个实例符号管脚。

ui::hidePin		
参数	**name,n***	字符串值； 一个管脚的名称；
	shrink, s	开关值； 打开时则不会保留原来管脚所占的位置；缺省为保留；
参见	**ui::showPin**	

- ***ui::enablePin***

使一个管脚有效，其在实例符号中的显示颜色会变正常。

ui::enablePin		
参数	**name,n***	字符串值； 一个管脚的名称；
参见	**ui::disablePin**	

- ***ui::disablePin***

使一个管脚无效，其在实例符号中的显示颜色会变灰。

ui::disablePin		
参数	**name,n***	字符串值； 一个管脚的名称；

参见	**ui::enablePin**

- ***ui::setPin***

  设置一个管脚的属性。

ui::setPin		
参数	**edge,e**	枚举值，可选项为 ***left, right, top, bottom***； 表示位于符号的哪条边上；
	lsb	整数值； 表示 bus 的最低有效位，缺省为保持不变；
	msb	整数值； 表示 bus 的最高有效位，缺省为保持不变；
	name,n*	字符串值； 管脚的名称；
	text,t	字符串值； 管脚的显示文本，对于 bus 不需包括其有效索引值，缺省为无；
参见	**ui::addPin**	

- ***ui::setPinIndex***

  设置 bus 管脚的有效索引值的范围。已废弃，请用 **ui::setPin** 代替。

ui::setPinIndex		
参数	**lsb**	整数值； 表示 bus 的最低有效位，缺省为保持不变；
	msb	整数值； 表示 bus 的最高有效位，缺省为保持不变；
	name,n*	字符串值； 管脚的名称；

- ***ui::setPinPos***

  设置管脚的在符号上的位置。已废弃，请用 **ui::setPin** 代替。

ui::setPinPos		
参数	**name,n***	字符串值； 管脚的名称；
	pos	枚举值，可选项为 ***left, right, top, bottom***；表示管脚位于符号的哪条边上，缺省为保持不变；

- ***ui::setPinText***

  设置管脚的显示文本。已废弃，请用 **ui::setPin** 代替。

ui::setPinText		
参数	**name,n***	字符串值； 管脚的名称；
	text,t	字符串值； 管脚的显示文本，对于 bus 不需包括其有效索引值，缺省为无；

5.11 信息区域的操作

本节声明的这些系统过程用于控制信息区域。信息区域在缺省状态下是关闭的，需要通过编程才能将其打开。

布局和某些窗口组件可以通过增加 **info** 选项在信息区域中生成。

- ***ui::hideInfo***

  隐藏整个信息区域以及 *View Information* 开关按钮。

- ***ui::showInfo***

  显示信息区域以及 *View Information* 开关按钮。

- ***ui::clearInfo***

  清空信息区域中的内容。

5.12 综合工具的设置

- ***ui::addSynOption***

  添加一个综合工具支持的脚本语句。

ui::addSynOption		
参数	**text,t**	字符串值； 综合工具支持的一个脚本语句；
参见	**ui::clearSynOptions**	

- ***ui::clearSynOptions***

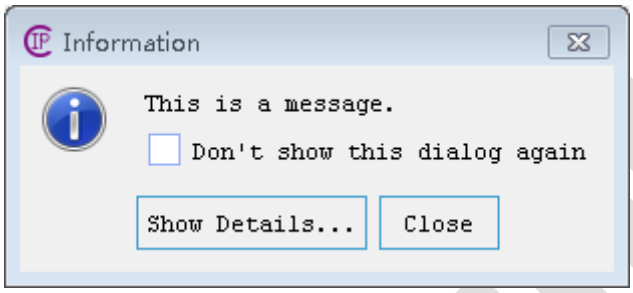
  清除所有用 *ui::addSynOption* 加入的综合工具设置脚本。

5.13 信息提示

通常在一个 IP 模块的界面设计完毕后，对用户的提示可以通过文本标签等显示类组件和窗口组件的提示(tip)来显示一些短格式的信息，比较详细的(动态)信息比如当前占用资源等可以在信息区域中显示，而对于警告或错误信息则可以弹出一个信息对话框。对于 puts 等 Tcl 语句则仅限于开发者调试代码时使用。

- ***ui::popupMessage***

  弹出一个信息对话框，如下图所示。



ui::popupMessage		
参数	**beep**	开关值； 打开时表示会在弹出对话框时发出一声提示音；
	button,b	字符串列表； 可以自行指定一组按钮以代替缺省的 *Close* 按钮；
	default	字符串值； 指定一个缺省按钮；
	detail	字符串值；可以指定一些更详细的信息内容，这样在对话框中会出现一个 *Show Details* 按钮以显示这些信息，通常这些信息是有关错误的细节；
	error,e	开关值； 表示这是一个错误信息；
	message,m*	字符串值； 信息内容；
	notshowagain,nsa	开关值； 打开时在对话框中会出现一个 *Don't show this dialog again* 的选项，勾选它则该对话框在配置当前 IP 实例过程中不再弹出；
	title,t	字符串值；可以指定一个对话框的窗口标题，如果未指定则显示为当前模型的名称和版本号；
	warning,w	开关值； 表示这是一个警告信息；
返回值	点击的按钮文本	
参见	**ui::print**	

- ***ui::popupBalloon***

弹出一条(气球)提示，如下图所示。通常是对一个窗口组件的用法或错误进行提示。可以指定一个显示时间，时间到后会自动关闭这个提示；通过鼠标点击也可以关闭该提示；在窗口组件移动或发生其它状态变化时也会自动关闭提示。

ui::popupBalloon		
参数	**color,clr**	颜色值；可以是一个颜色的名字，比如 red 或者 blue 等，也可以是一个十六位的 RGB 值，其格式可以为以下几种：#RGB、#RRGGBB、#RRRGGGBBB、#RRRRGGGGBBBB；
	message,m*	字符串值； 信息内容；
	name,n*	字符串值； 一个窗口组件的名称；
	timeout	整数值； 提示显示的时间，缺省为 5000(毫秒)；
	repeat	开关值； 下次光标再次进入到窗口组件中时会自动弹出提示，缺省为否；

- ***ui::print***

打印一条信息。与 *ui::popupMessage* 不同的是，*ui::print* 不会因为弹出对话框而影响流程的进行，而是将信息打印到标准输出或者被重定向到图形界面的信息区域中。

ui::print		
参数	**error,e**	开关值； 表示这是一个错误信息；
	message,m*	字符串值； 信息内容；
	warning,w	开关值； 表示这是一个警告信息；
参见	**ui::popupMessage**	

6 .IP 测试

IP 模型的参数较多，这些参数的正确组合及配置界面的正确性通常需要 IP 开发者手工反复测试。*ip-compiler* 有一种预览模式(preview)，这种模式可以在一定程度上提高 IP 的开发效率。

IP Compiler 还提供了一种测试模式，可以对单个的或批量的 IP 模型进行快速测试。需要注意的是，这种测试因为无法配置参数值，所以只能根据初始参数的配置进行测试，并不能进行覆盖式测试。即便如此，它可以方便地对各种支持的 arch 配置以及软件兼容性等进行测试，适合于在软件发布前对 IP 库进行批量测试，发现那些比较初级的错误，以便进行进一步测试修改。

测试模式针对两个目标分别进行，一个是对 IP 模型用户界面的测试，一个是对 IP 实例（生成）的测试。

在终端窗口中输入 *ip-test* 可以开始测试。

6.1 测试 IP 模型

软件可以自动地依次对指定的 IP 模型在各种有效的 arch 配置下进行快速测试。期间可能会有配置窗口自动打开并关闭，注意此时的配置窗口是处于预览模式下，可以点击各组件，但是结果不能保存。此外，该测试还可以对模型索引文件 *index.xml* 中**<datasheet>**和**<info>**等元素指定的文件进行存在性检查。主要命令行参数见下表：

-m *<DIR>*	指定一个 IP 模型 I 所在的路径；或者某些 IP 模型所在的上层路径(需与**-r** 选项配合使用)；如果没有指定任何路径，则会选择当前安装路径下的 ip 目录(需与**-r** 选项配合使用)；
-r	递归地去搜索指定路径下的所有 IP 模型
-win	显示 IP 配置窗口
-no-dialog	自动关闭出现的对话框，返回缺省按钮值，并打印对话框的信息内容；
-continue	如果测试中遇到错误则会继续测试其它 IP 模型，否则会立即停止；
-wait *<NUMBER>*	使得配置窗口在打开后保持一段时间，缺省为 3 秒；
-family *<STRING>*	指定一个需要测试的器件系列，缺省为没有限制；
-device *<STRING>*	指定一个需要测试的器件类型，缺省为没有限制；
-pack *<STRING>*	指定一个需要测试的封装形式，缺省为没有限制；
-speed *<STRING>*	指定一个需要测试的速度等级，缺省为没有限制；

6.2 测试 IP 实例

如果已经有了一个配置好的 IP 实例，希望用它配合新的 IP 模型进行测试，则可以按照下表中的命令行参数进行设置，同时还可以使用综合器的设置选项。

-i *<FILE>*	指定一个 IP 实例的数据文件路径
-generate	对 IP 实例执行生成操作
-win	显示 IP 配置窗口
-no-dialog	自动关闭出现的对话框，返回缺省按钮值，并打印对话框的信息内容；
-wait *<NUMBER>*	使得配置窗口在打开后保持一段时间，缺省为 3 秒；

6.3 测试举例

对安装路径下 IP 库中的所有模型进行测试

```
ip-test -r -continue
```

对安装路径下 IP 库中的所有模型针对 Titan 系列进行测试

```
ip-test -r -continue -family Titan
```

对某路径下 IP 库中的所有模型进行测试

```
ip-test -r -continue -m <A-PATH>
```

对某路径下的单个 IP 模型进行测试

```
ip-test -m <A-MODEL-PATH>
```

对某路径下的单个 IP 实例进行测试

```
ip-test -i <AN-INST-PATHNAME>
```

7 IP 的升级安装与管理

前面已经提到过，对于 IP Compiler，一个 IP 模块在计算机磁盘上是以一个单独的目录的形式存放的，所有这些模块都被放置在工具安装路径下的一个固定位置，即*<INSTALL-PATH>/ip* 及其子目录中。如果不能确定 IP 模块的安装位置，可以在 IP Compiler 主窗口中点击 *File->Preferences->Update* 进行查看。更新一个 IP 模块时，可以直接手工更换或复制一个磁盘目录，但更好的方法是利用打包工具 *ip-tar* 来制作和发布 IP 包。

7.1 IP 模块内容的准备

IP 开发者在对一个 IP 模块进行组织打包时，首先需要将相关的所有文件(和子目录)存放在一个单独的目录中。因为在用户方面进行更新的时候不能指定解包的路径，所以一个 IP 模块所在的位置最好是位于一些有含义的路径中以便于管理和识别，比较好的习惯是可以按照发行商和功能等建立一组路径，比如 pango/ram/an_ip_name。

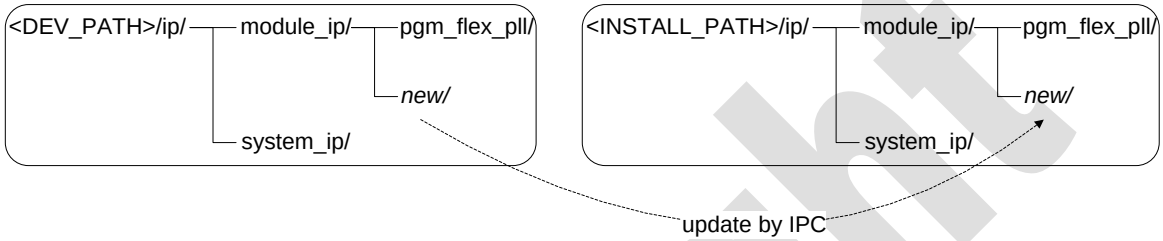
然后，需要检查 IP 索引文件 *index.xml* 中是否包了以下这些基本属性：IP 的名称、版本、发行商、工具兼容性等，它们会在更新时用到，错误的属性可能会导致更新失败。

7.2 IP 包的制作

通过下面的命令行，指定某个 IP 模块所在的路径以及一个打包文件名，就可以生成一个 IP 包。一个 IP 包文件名中最好包含 IP 的发行商、名称和版本信息，以便快速识别。

```
ip-tar -c <PACKAGE-FILE-NAME> -from <MODEL-PATH>
```

注意，IP 开发者应该建立一个与用户环境相同的 IP 目录结构，比如*<DEV-PATH>/ip/*，它应该与用户环境下的*<INSTALL-PATH>/ip/*目录结构是相同的。应该先将当前工作目录换到*<DEV-PATH>/ip/*中，再调用 *ip-tar* 来制作 IP 包时，这样命令行中的参数*<MODEL-PATH>*是一个相对于开发目录的路径，会被记录到 IP 包文件中，从而在用户处解包时可以被释放到指定的位置。



举例说明，上图左边的*<DEV-PATH>/ip/*是开发目录，右边的*<INSTALL-PATH>/ip/*安装目录，两者的结构是相同的。现在准备新增一个叫作 *new* 的 IP，它的磁盘位置在 *ip/module_ip/new/*，则执行以下操作：

```
cd <DEV_PATH>/ip
ip-tar -c new.iar -from module_ip/new
```

注意命令行参数*-from* 后面用的是一个相对路径，那么在用户端通过 Update 操作来更新这个 IP 包 *new.iar* 时，它会自动被释放到*<INSTALL_PATH>/ip/module/ip/new/*目录，而不需用户干预。

在制作 IP 包时，可以使用命令行参数*-encrypt* 对 IP 包内容进行加密。注意，这并不同于使用 IEEE 1735 等协议对 HDL 文件的内容进行的加密，而仅是对 IP 包整体进行加密，从而不暴露 Tcl 代码等实现细节。

```
ip-tar -c new.iar -from module_ip/new -encrypt
ip-tar -c new.iar -from module_ip/new -encrypt vendor-public-key.txt
```

注意命令行参数*-encrypt* 后面有一个可选的公钥文件，如果开发者可能会自己手动解包，则需要用一个私钥文件；如果不需要，则不必指定该文件。

```
ip-tar -x new.iar -to somewhere -decrypt vendor-private-key.txt
```

公钥和私钥可以用 *ip-tar* 自行生成。注意保管好私钥文件。

```
ip-tar -genpkey
```

7.3 IP 包的测试

通过*ip-tar*自带的测试功能，可以检测一个IP包的完整性。如果在测试过程中发现问题，*ip-tar*会打印错误信息，并以一个非 0 值退出；否则会以表示正常的 0 值退出。

```
ip-tar -test <PACKAGE-FILE-NAME>
```

检查的内容包括：

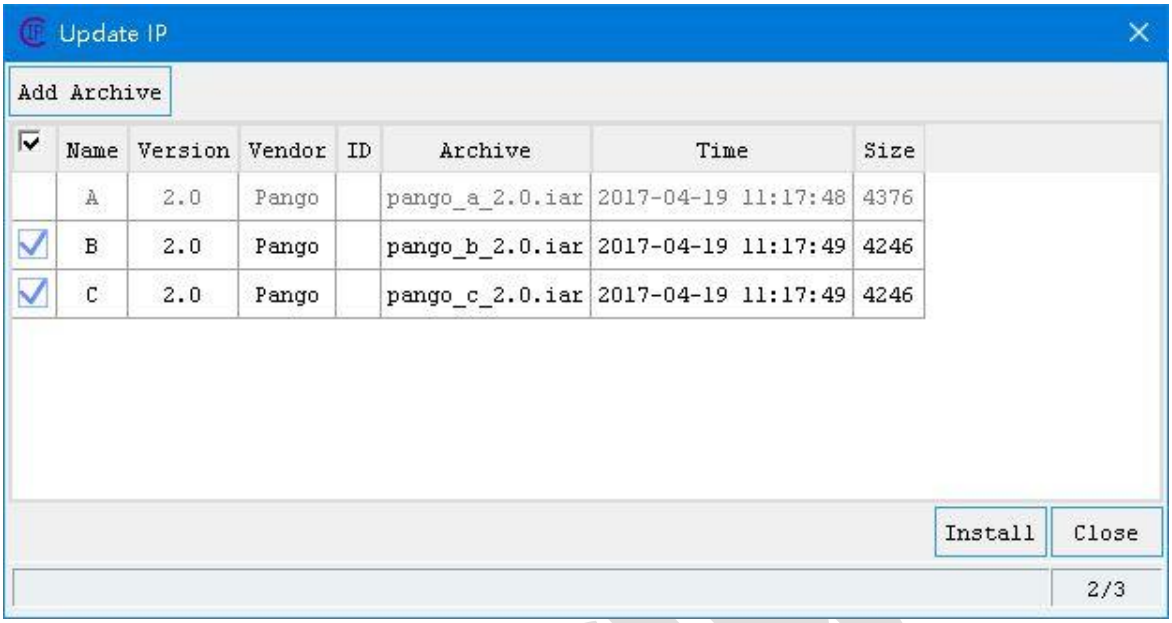
- 是否包含有多个的模型索引文件；
- 是否包含一个有效的模型索引文件；
- IP 的名称和版本是否已设置；
- 是否与当前软件工具的版本相兼容(注意 *ip-tar* 的版本需要与客户所使用的版本相同)；

如果还想要查看一个 IP 包中的完整内容，可以用下面的命令行打印出包内所有的文件名和目录名，但它不会进行任何检查性工作。

```
ip-tar -t <PACKAGE-FILE-NAME>
```

7.4 用户端更新

在 IP Compiler 主窗口中点击 *File-> Update* 打开更新对话框，可以将 IP 包装载进来进行查看(见下图)。不可用的 IP 包会用浅灰色表示出来，不可用指的是与当前软件工具不兼容，或者比当前已安装的 IP 模块版本更老旧，不可用并不代表 IP 包本身是错误的。



点击按钮 *Update* 开始更新勾选的 IP 模块。如果安装路径下发现其它已存在的 IP 索引文件，则会弹出警告对话框(见下图)，用户可以选择是否覆盖。

7.5 冲突解决

在安装 IP 的过程中可能会遇到与已有 IP 的冲突，或者安装路径上存在冲突，IPC 会按照一定的流程进行解决。

第一种情况：IPC 会检查要安装的 IP 是否已存在，即存在一个 vendor/id/version 都相同的 IP。如果存在，则 IPC 会弹窗提示用户是否进行覆盖，如果用户选择不覆盖，则退出安装。

第二种情况：在没有相同的 IP 存在时，IPC 会检查安装路径的各父路径中（直到 *<INSTALL_PATH>/ip/*）是否有其它 IP 索引文件 index.xml 存在，如果有，则会弹窗警告并退出安装。

第三种情况：缺省安装路径已存在，则会用 IP 的版本号和一个随机序列数尝试构造一个与原安装路径相并列的路径。但尝试的次数是有限的，如果超过预定次数还没有找到合适的路径，则 IPC 会弹窗提示用户是否覆盖原安装路径，如果用户选择不覆盖，则退出安装。

这样，最终的安装路径就确定下来了，可能就是原来的预定路径，也可能是一个新的路径。

附录：Tcl 过程函数索引

ui::addButton 54
ui::addCheckBox 40
ui::addComboBox 42
ui::addDoubleSpinBox 45
ui::addFrame 50
ui::addGrid 34
ui::addGroupBox 51
ui::addHBox 32
ui::addHSep 48
ui::addImage 47
ui::addLabel 46
ui::addLineEdit 38
ui::addList 43
ui::addPin 69
ui::addPinGroup 69
ui::addPinSpacing 70
ui::addRadioButton 41
ui::addSpacer 49
ui::addSpacing 49
ui::addSpinBox 44
ui::addStackedPage 53
ui::addStackedWidget 52
ui::addSynOption 72
ui::addTextEdit 39
ui::addVBox 33
ui::addVSep 48
ui::check 31
ui::clearInfo 72
ui::clearSynOptions 72
ui::count 59
ui::disable 56
ui::disablePin 70
ui::enable 56
ui::enablePin 70
ui::getArchPath 66
ui::getCurrentIndex 59
ui::getDefaultValue 62
ui::getDevice 65
ui::getFamily 65
ui::getInstallPath 65
ui::getInstName 64
ui::getInstPath 64
ui::getItems 63
ui::getMaxValue 62
ui::getMinValue 62
ui::getModelName 64
ui::getModelPath 64
ui::getModelVersion 64
ui::getPackage 65
ui::getPkgPinBank 67
ui::getPkgPinGroup 67
ui::getPkgPins 67
ui::getPkgPinSpec 67
ui::getSpeedGrade 65
ui::getText 57
ui::getToolVersion 66
ui::getTopModule 8
ui::getValue 61
ui::hide 55
ui::hideInfo 72
ui::hidePin 70
ui::hideSymbol 68
ui::init 30
ui::isEnabled 56
ui::isPkgPinDQS 67
ui::isVisible 55
ui::popupBalloon 74
ui::popupMessage 73
ui::print 74
ui::resetColor 60
ui::setColor 60
ui::setCurrentIndex 60
ui::setGridColumnStretch 34
ui::setGridRowStretch 35
ui::setImage 58
ui::setItems 63
ui::setMaxValue 62
ui::setMinValue 63
ui::setPin 71
ui::setPinIndex 71
ui::setPinPos 71
ui::setPinText 71
ui::setText 57
ui::setTextLen 59
ui::setTip 58
ui::setTopModule 8
ui::setValidator 59
ui::setValue 61
ui::show 55
ui::showInfo 72
ui::showPin 70
ui::showSymbol 68
ui::update 30

免责声明

版权声明

免责声明

1、本文档仅提供阶段性信息，所含内容可根据产品的实际情况随时更新，恕不另行通知。如因本文档使用不当造成的直接或间接损失，本公司不承担任何法律责任。

2、本文档按现状提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

3、公司保留任何时候在不事先声明的情况下对公司系列产品相关文档的修改权利。