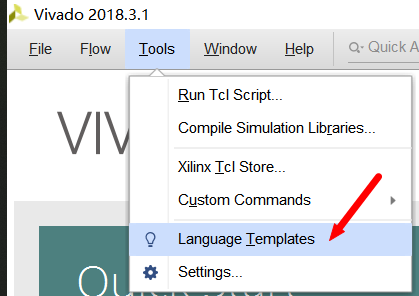
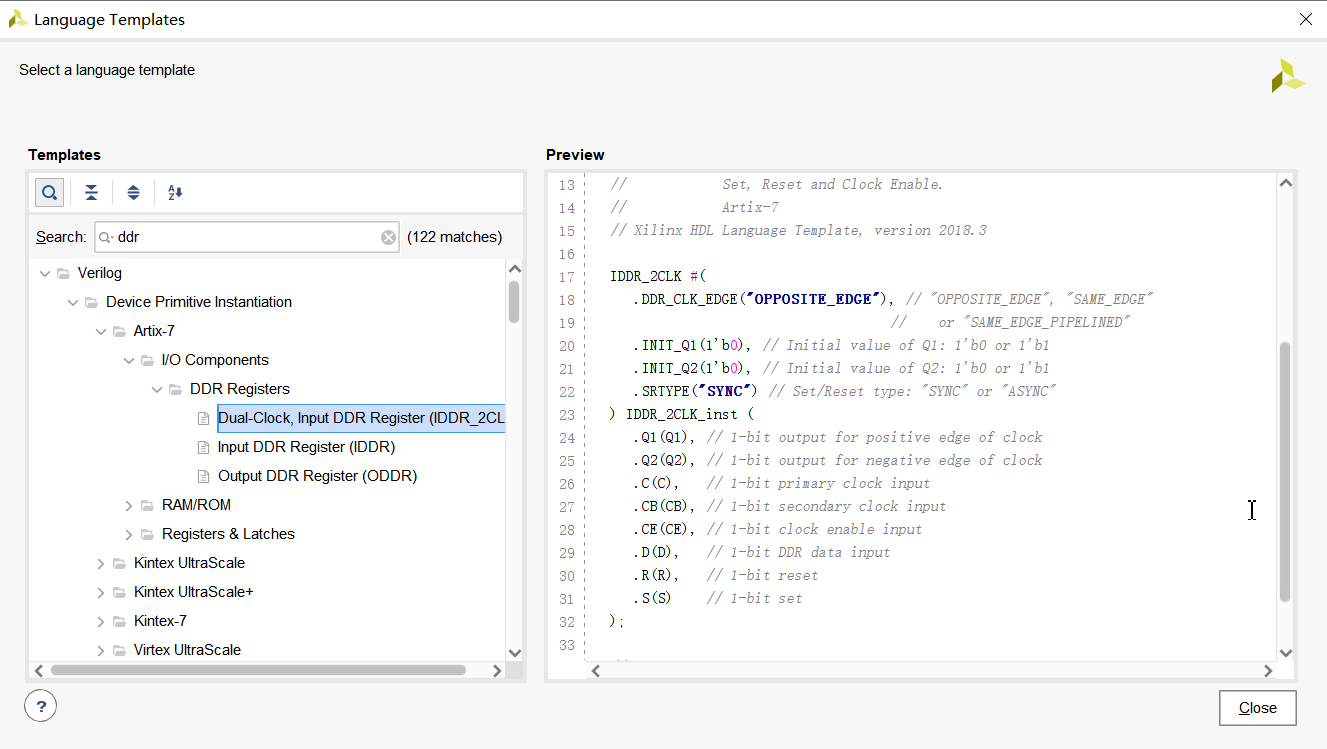
**Xilinx公司原语的使用方法**

原语，其英文名字为Primitive，是Xilinx针对其器件特征开发的一系列常用模块的名字，用户可以将其看成Xilinx公司为用户提供的库函数，类似于C++中的“cout”等关键字，是芯片中的基本元件，代表FPGA中实际拥有的硬件逻辑单元，如LUT，D触发器，RAM等，相当于软件中的机器语言。在实现过程中的翻译步骤时，要将所有的设计单元都转译为目标器件中的基本元件，否则就是不可实现的。原语在设计中可以直接例化使用，是最直接的代码输入方式，其和HDL语言的关系，类似于汇编语言和C语言的关系。   
  
Xilinx公司提供的原语，涵盖了FPGA开发的常用领域，但只有相应配置的硬件才能执行相应的原语，并不是所有的原语都可以在任何一款芯片上运行。在Verilog中使用原语非常简单，将其作为模块名直接例化即可。本节以**Virtex-4**平台介绍各类原语，因为该系列的原语类型是最全面的。其它系列芯片原语的使用方法是类似的。   
  
Xilinx公司的原语按照功能分为10类，包括：计算组件、I/O端口组件、寄存器和锁存器、时钟组件、处理器组件、移位寄存器、配置和检测组件、RAM/ROM组件、Slice/CLB组件以及G比特收发器组件。下面分别对其进行详细介绍。   
**Vivado 中**

**TOOL->Language Template**





**3.4.1 计算组件**   
  
计算组件值得就是DSP48核，也有人将其称为硬件乘法器，功能描述如表3-6所示。

 表3-6 计算组件清单

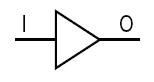
[点击看大图](http://www.openhw.org/data/08-03/4155_1205985007/1206435001.jpg)

DSP48 其结构为一个18\*18比特的有符号乘法器，且在后面还级联了一个带有可配置流水线的3输入加法器   
  
DSP48核由一个18比特的乘法后面级联一个48比特的加法器，乘法器和加法器的应用位宽分别可以在18、48比特内任意调整。其在乘加模块中有广泛应用，特别是各类滤波器系统中，不仅可以提高系统稳定性，还能够节省逻辑资源且工作在高速模式下。其在Verilog中的例化模版为：   
  
module fpga\_v4\_dsp48(   
         BCOUT, P, PCOUT, A, B, BCIN, C, CARRYIN,CARRYINSEL, CEA, CEB,   
         CEC, CECARRYIN, CECINSUB, CECTRL, CEM, CEP, CLK, OPMODE,   
         PCIN, RSTA, RSTB, RSTC, RSTCARRYIN, RSTM, RSTP, SUBTRACT);   
         output [17:0]BCOUT;   
         output [47:0] P, PCOUT; //   
         input [17:0] A, B; //   
         input [47:0] C, PCIN;   
         input [1:0] CARRYINSEL;   
         input [6:0] OPMODE;   
         input BCIN, CARRYIN,CEA,CEB, CEC,CECARRYIN,CECINSUB,CECTRL,CEM,   
         CEP,CLK, RSTA, RSTB,RSTC,RSTCARRYIN,RSTM,RSTP,SUBTRACT;   
  
         //对DSP48原语的功能进行配置。   
         DSP48 #(   
                      .AREG(1), // Number of pipeline registers on the A input, 0, 1 or 2   
                      .BREG(1), // Number of pipeline registers on the B input, 0, 1 or 2   
                      .B\_INPUT("DIRECT"),   
                       // B input DIRECT from fabric or CASCADE from another DSP48   
                     .CARRYINREG(1),   
                      // Number of pipeline registers for the CARRYIN input, 0 or 1   
                     .CARRYINSELREG(1),   
                      // Number of pipeline registers for the CARRYINSEL, 0 or 1   
                     .CREG(1), // Number of pipeline registers on the C input, 0 or 1   
                      .LEGACY\_MODE("MULT18X18S"),   
                      // Backward compatibility, NONE, MULT18X18 or MULT18X18S   
                     .MREG(1), // Number of multiplier pipeline registers, 0 or 1   
                     .OPMODEREG(1), // Number of pipeline regsiters on OPMODE input, 0 or 1   
                     .PREG(1), // Number of pipeline registers on the P output, 0 or 1   
                     .SUBTRACTREG(1)   
                      // Number of pipeline registers on the SUBTRACT input, 0 or 1   
                    ) fpga\_v4\_dsp48 (   
                     .BCOUT(BCOUT), // 18-bit B cascade output   
                     .P(P), // 48-bit product output   
                     .PCOUT(PCOUT), // 48-bit cascade output   
                     .A(A), // 18-bit A data input   
                     .B(B), // 18-bit B data input   
                     .BCIN(BCIN), // 18-bit B cascade input   
                     .C(C), // 48-bit cascade input   
                     .CARRYIN(CARRYIN), // Carry input signal   
                     .CARRYINSEL(CARRYINSEL), // 2-bit carry input select   
                     .CEA(CEA), // A data clock enable input   
                     .CEB(CEB), // B data clock enable input   
                     .CEC(CEC), // C data clock enable input   
                     .CECARRYIN(CECARRYIN), // CARRYIN clock enable input   
                     .CECINSUB(CECINSUB), // CINSUB clock enable input   
                     .CECTRL(CECTRL), // Clock Enable input for CTRL regsiters   
                     .CEM(CEM), // Clock Enable input for multiplier regsiters   
                     .CEP(CEP), // Clock Enable input for P regsiters   
                     .CLK(CLK), // Clock input   
                     .OPMODE(OPMODE), // 7-bit operation mode input   
                      .PCIN(PCIN), // 48-bit PCIN input   
                      .RSTA(RSTA), // Reset input for A pipeline registers   
                      .RSTB(RSTB), // Reset input for B pipeline registers   
                      .RSTC(RSTC), // Reset input for C pipeline registers   
                      .RSTCARRYIN(RSTCARRYIN), // Reset input for CARRYIN registers   
                      .RSTCTRL(RSTCTRL), // Reset input for CTRL registers   
                      .RSTM(RSTM), // Reset input for multiplier registers   
                      .RSTP(RSTP), // Reset input for P pipeline registers   
                      .SUBTRACT(SUBTRACT) // SUBTRACT input   
                   );   
  
endmodule   
  
**3.4.2 时钟组件**时钟组件包括各种全局时钟缓冲器、全局时钟复用器、普通I/O本地的时钟缓冲器以及高级数字时钟管理模块，如表3-7所示。

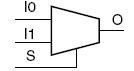
表3-7 时钟组件的清单

[](http://www.openhw.org/data/08-03/4155_1205985007/1206170248.jpg)

下面对几个常用时钟组件进行简单介绍，其余组件的使用方法是类似的。   
  
1．BUFG   
  
BUFG是具有高扇出的全局时钟缓冲器，一般由综合器自动推断并使用，其和同类原语的RTL结构如图3-28所示。全局时钟是具有高扇出驱动能力的缓冲器，可以将信号连到时钟抖动可以忽略不计的全局时钟网络，BUFG组件还可应用于典型的高扇出信号和网络，如复位信号和时钟使能信号。如果要对全局时钟实现PLL或DCM等时钟管理，则需要手动例化该缓冲器。其例化的代码模板如下所示：   
  
// BUFG: 全局时钟缓存（Global Clock Buffer），只能以内部信号驱动   
// Xilinx HDL库向导版本，ISE 9.1   
BUFG BUFG\_inst (   
.O(O), //时钟缓存输出信号   
.I(I) // /时钟缓存输入信号   
);   
// 结束BUFG\_ins模块的例化过程   
  
在综合结果分析中，其和同类原语的RTL结构如图3-32所示。

  
  
图3-32 全局时钟原语的RTL级结构示意图

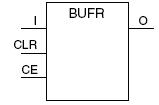
2. BUFMUX   
  
BUFMUX是全局时钟复用器，选择两个输入时钟I0或I1中的一个作为全局时钟，其和同类原语BUFMUX1的RTL级结构如图M所示。当选择信号S为低时，选择I0；否则输出I1，其真值表如表M所示。BUFMUX原语和BUFMUX1原语的功能一样，只是选择逻辑不同，对于BUFMUX1，当选择信号S为低时，选择I1；否则输出I0。  
  
BUFMUX原语的例化代码模板如下所示：   
  
BUFMUX原语的例化代码模板如下所示：   
// BUFGMUX: 全局时钟的2到1复用器（Global Clock Buffer 2-to-1 MUX）   
// 适用芯片：Virtex-II/II-Pro/4/5, Spartan-3/3E/3A   
// Xilinx HDL库向导版本，ISE 9.1   
BUFGMUX BUFGMUX\_inst (   
.O(O), //时钟复用器的输出信号   
.I0(I0), // 0时钟输入信号   
.I1(I1), //1时钟输入信号   
.S(S) // 时钟选择信号   
);   
// 结束BUFGMUX\_inst模块的例化过程   
  
需要注意的是：该原语只用用全局时钟处理，不能作为接口使用。在综合结果分析时，它和同类原语BUFMUX1的RTL级结构如图3-33所示。

  
图3-33 全局时钟复用器的RTL级结构示意图

3. BUFIO   
  
BUFIO是本地I/O时钟缓冲器，其RTL结构如图Ｍ所示，只有一个输入与输出，非常简单。BUFIO使用独立于全局时钟网络的专用时钟网络来驱动纵向I/O管脚，所以非常适合同步数据采集。BUFIO要求时钟和相应的I/O必须在同一时钟区域，而不同时钟网络的驱动需要BUFR原语来实现。需要注意的是，由于BUFIO引出的时钟只到达了I/O列，所以不能来驱动逻辑资源，如CLB和块RAM。   
  
BUFIO的例化代码模板如下：   
  
// BUFIO: 本地I/O时钟缓冲器（ Local Clock Buffer）   
// 适用芯片：**Virtex-4**/5   
// Xilinx HDL库向导版本，ISE 9.1   
BUFIO BUFIO\_inst (   
.O(O), //本地I/O时钟缓冲器的输出信号   
.I(I) //本地I/O时钟缓冲器的输入信号   
);   
// 结束BUFIO模块的例化过程   
在综合结果分析时，其RTL级结构如图3-34所示。

http://www.openhw.org/data/08-03/4155_1205985007/1211951798.jpg  
  
图3-34 本地I/O时钟缓冲器的RTL级结构示意图

4. BUFR   
  
BUFR是本地I/O时钟、逻辑缓冲器，其RTL结构如图Ｍ所示。BUFR和BUFIO都是将驱动时钟引入某一时钟区域的专用时钟网络，而独立于全局时钟网络；不同的是，BUFR不仅可以跨越不同的时钟区域（最多3个），还能够驱动I/O逻辑以及自身或邻近时钟区域的逻辑资源。BUFIO的输出和本地内部互联都能驱动BUFR组件。此外，BUFR能完成输入时钟1～8的整数分频。因此，BUFR是同步设计中实现跨时钟域以及串并转换的最佳方式。   
  
BUFIO的例化代码模板如下：   
  
// BUFR: 本地I/O时钟、逻辑缓冲器（Regional Clock Buffer）   
// 适用芯片：**Virtex-4**/5   
// Xilinx HDL库向导版本，ISE 9.1   
BUFR #(   
.BUFR\_DIVIDE("BYPASS"),   
//分频比，可选择 "BYPASS", "1", "2", "3", "4", "5", "6", "7", "8"。   
.SIM\_DEVICE("VIRTEX4")   
// 指定目标芯片, "VIRTEX4" 或者 "VIRTEX5"   
) BUFR\_inst (   
.O(O), //时钟缓存输出信号   
.CE(CE), //时钟使能信号，输入信号   
.CLR(CLR), //时钟缓存清空信号   
.I(I) // 时钟缓存输入信号   
);   
// 结束BUFR模块的例化过程   
  
需要注意的是：BUFIO和BUFR只能在**Virtex-4**系列以及更高系列芯片中使用。在综合结果分析时，其RTL结构如图3-35所示。

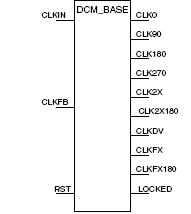
  
  
图3-35 本地I/O时钟、逻辑缓冲器的RTL级结构示意图

5. DCM\_BASE   
  
DCM\_BASE是基本数字时钟管理模块的缩写，是相位和频率可配置的数字锁相环电路，常用于FPGA系统中复杂的时钟管理。如果需要频率和相位动态重配置，则可以选用DCM\_ADV原语；如果需要相位动态偏移，可使用DCM\_PS原语。DCM系列原语的RTL结构如图3-8所示。

模块接口信号的说明如表3-8所列。

[](http://www.openhw.org/data/08-03/4155_1205985007/1211952164.jpg)

DCM\_BASE组件可以通过Xilinx的IP Wizard向导产生，也可以直接通过下面的例化代码直接使用。其Verilog的例化代码模板为：   
  
// DCM\_BASE: 基本数字时钟管理电路（Base Digital Clock Manager Circuit）   
// 适用芯片：**Virtex-4**/5   
// Xilinx HDL库向导版本，ISE 9.1   
DCM\_BASE #(   
.CLKDV\_DIVIDE(2.0),   
// CLKDV分频比可以设置为: 1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5   
// 7.0,7.5,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0 or 16.0   
.CLKFX\_DIVIDE(1), // Can be any integer from 1 to 32   
// CLKFX信号的分频比，可为1到32之间的任意整数   
.CLKFX\_MULTIPLY(4),   
// CLKFX信号的倍频比，可为2到32之间的任意整数   
.CLKIN\_DIVIDE\_BY\_2("FALSE"),   
// 输入信号2分频的使能信号，可设置为TRUE/FALSE   
.CLKIN\_PERIOD(10.0),   
// 指定输入时钟的周期，单位为ns，数值范围为1.25~1000.00。   
.CLKOUT\_PHASE\_SHIFT("NONE"),   
// 指定移相模式，可设置为NONE或FIXED   
.CLK\_FEEDBACK("1X"),   
// 指定反馈时钟的频率，可设置为NONE、1X或2X。相应的频率关系都是针对CLK0而言的。   
.DCM\_PERFORMANCE\_MODE("MAX\_SPEED"),   
// DCM模块性能模式，可设置为 MAX\_SPEED 或 MAX\_RANGE   
.DESKEW\_ADJUST("SYSTEM\_SYNCHRONOUS"),   
// 抖动调整，可设置为源同步、系统同步或0~15之间的任意整数   
.DFS\_FREQUENCY\_MODE("LOW"),   
// 数字频率合成模式，可设置为LOW或HIGH 两种频率模式   
.DLL\_FREQUENCY\_MODE("LOW"),   
// DLL的频率模式，可设置为LOW、HIGH或HIGH\_SER   
.DUTY\_CYCLE\_CORRECTION("TRUE"),   
// 设置是否采用双周期校正，可设为TRUE或FALSE   
.FACTORY\_JF(16'hf0f0),   
// 16比特的JF因子参数   
.PHASE\_SHIFT(0),   
// 固定相移的数值，可设置为 -255 ~ 1023之间的任意整数   
.STARTUP\_WAIT("FALSE")   
// 等DCM锁相后再延迟配置DONE管脚，可设置为TRUE/FALSE   
) DCM\_BASE\_inst (   
.CLK0(CLK0), // 0度移相的DCM时钟输出   
.CLK180(CLK180), // 180度移相的DCM时钟输出   
.CLK270(CLK270), // 270度移相的DCM时钟输出   
.CLK2X(CLK2X), // DCM模块的2倍频输出   
.CLK2X180(CLK2X180), // 经过180度相移的DCM模块2倍频输出   
.CLK90(CLK90), // 90度移相的DCM时钟输出   
.CLKDV(CLKDV), // DCM模块的分频输出，分频比为CLKDV\_DIVIDE   
.CLKFX(CLKFX), // DCM合成时钟输出，分频比为(M/D)   
.CLKFX180(CLKFX180), // 180度移相的DCM合成时钟输出   
.LOCKED(LOCKED), // DCM锁相状态输出信号   
.CLKFB(CLKFB), // DCM模块的反馈时钟信号   
.CLKIN(CLKIN), // DCM模块的时钟输入信号   
.RST(RST) // DCM 模块的异步复位信号   
);   
// 结束DCM\_BASE模块的例化过程   
  
在综合结果分析时，DCM系列原语的RTL结构如图3-36所示。

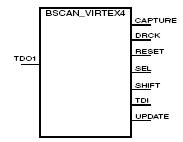
  
  
图3-36 DCM模块的RTL级结构示意图

**3.4.3 配置和检测组件**   
  
配置和检测组件提供了FPGA内部逻辑和JTAG扫描电路之间的数据交换以及控制功能，只要由6个原语组成，如表3-9所示。

表3-9 配置和检测原语列表



下面对**BSCAN**\_VIRTEX4组件进行简单介绍，其余组件的使用方法是类似的。   
  
1．**BSCAN**\_VIRTEX4   
  
当JTAG USER1/2/3/4指令被加载后，**BSCAN**\_VIRTEX4允许设计人员来检测TCK、TMS以及TDI等专用JTAG管脚的数据，并且可以将用户数据写入到TDO管脚上，这样可以在PC上通过JTAG链读取芯片内部的用户数据。   
  
**BSCAN**\_VIRTEX4的管脚信号说明如下：   
  
CAPTURE：位宽为1的输出信号，用于指示是否加载了用户指令，当JTAG接口处于CAPTURE-DR状态时，输出为高。   
DRCK：位宽为1的输出信号，用于监测JTAG电路的TCK信号。当JTAG链路处于用户指令模式或者JTAG接口为SHIFT-DR状态时，才有信号输出。   
RESET：位宽为1的输出信号，在加载用户指令时有效。当JTAG接口控制器处于TEST-LOGIC-RESET状态时置高。   
SEL：位宽为1的输出信号，高有效。用于指示USER1数据是否加载到JTAG指令寄存器中。在UPDATE-IR状态时有效，直到加载新的指令之前，一直保持有效电平。   
SHIFT：位宽为1的输出信号，加载用户指令时有效。当JTAG接口控制器处于SHIFT-DR状态时置高。   
TDI：位宽为1的输出信号，用于检测JTAG链的TDI信号。   
UPDATE：位宽为1的输出信号，加载USER1指令和USER2指令时有效。当JTAG接口控制器处于UPDATE-DR状态时置高。   
TDO：位宽为1的输入信号，可以将外部JTAG链的TDO信号直接连到该管脚上。   
在**Virtex-4**芯片中，有4个**BSCAN**\_VIRTEX4硬件原语可用。因此，其属性JTAG\_CHAIN的有效值为1～4，默认值为1。   
**BSCAN**\_VIRTEX4原语的例化代码模板如下所示：   
// **BSCAN**\_VIRETX4: 完成内部逻辑和JTAG接口连接的边界扫描原语（Boundary Scan primitive for connecting internal logic to JTAG interface.）   
// 适用芯片：**Virtex-4**/5   
// Xilinx HDL库向导版本，ISE 9.1   
**BSCAN**\_VIRETX4 #(   
.JTAG\_CHAIN(1)   
// 指定JTAG链用户指令，必须为1, 2, 3, 或4中的任何一个正整数   
) **BSCAN**\_VIRETX4\_inst (   
.CAPTURE(CAPTURE), // 捕获到的从TAP控制器的输出   
.DRCK(DRCK), // 用户函数服务的数据寄存器输出   
.RESET(RESET), // TAP控制器输出的复位信号   
.SEL(SEL), // 用户激活输出   
.SHIFT(SHIFT), // TAP控制器的移位输出   
.TDI(TDI), // TAP控制器的TDI输出信号   
.UPDATE(UPDATE), // TAP控制器的UPDATE输出信号   
.TDO(TDO) // 用户函数的数据输入信号   
);   
// 结束**BSCAN**\_VIRETX4:模块的例化过程   
  
在综合结果分析时，**BSCAN**\_VIRTEX4的RTL结构图如图3-37所示。

  
  
图3-37 **BSCAN**\_VIRTEX4原语的RTL图

**3.4.4 吉比特收发器组件**   
  
吉比特收发器组件用于调用RocketIO千兆位级收发器，支持从622 Mbps 到 6.5 Gbps的多速率应用，符合最广泛的芯片、背板和光学器件的标准及协议，具有高级Tx/Rx均衡技术，收发器最多可达24个，提供了完整的串行 I/O 解决方案。具体包括4个组件，如表3-10所示。

表3-10 吉比特收发器组件



由于吉比特收发器操作复杂，使用原语很容易出错，不易配置，因此需要在ISE中通过结构向导完成。有关吉比特收发器的原理和使用方法，将在第11章详细介绍。   
  
**3.4.5 I/O端口组件**   
  
I/O组件提供了本地时钟缓存、标准单端I/O缓存、差分I/O信号缓存、DDR专用I/O信号缓存、可变抽头延迟链、上拉、下拉以及单端信号和差分信号之间的相互转换，具体包括了21个原语，如表3-11所示。

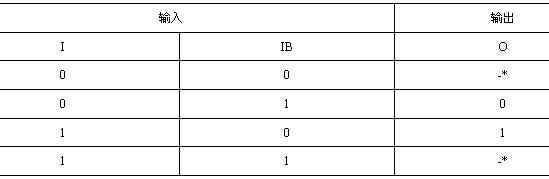
 表3-11 I/O端口组件   


下面对几个常用I/O组件进行简单介绍，其余组件的使用方法是类似的。  
  
1. BUFIO   
BUFIO是FPGA芯片内部简单的时钟输入、输出缓存器，其RTL结构如图3-38所示。BUFIO使用独立于全局时钟网络的专用时钟网络来驱动I/O列，因此非常适合用于源同步的数据采集。但要注意的是：BUFIO只能在单一的时钟区域内使用，不能跨时钟域操作。此外，BUFIO也不能用于驱动逻辑资源（CLB、块RAM等），因此其只能到达I/O列。

http://www.openhw.org/data/08-03/4155_1205985007/1206170903.jpg

图3-38 BUFIO的RTL结构图

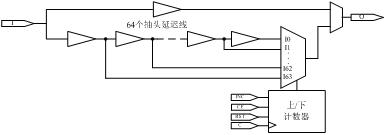
2. IBUFDS   
  
IBUFDS原语用于将差分输入信号转化成标准单端信号，且可加入可选延迟。在IBUFDS原语中，输入信号为I、IB，一个为主，一个为从，二者相位相反。   
  
IBUFDS的逻辑真值表如表3-12所列，其中“-\*”表示输出维持上一次的输出值，保持不变。

表3-12 IBUFDS原语的输入、输出真值表   
  
[](http://www.openhw.org/data/08-03/4155_1205985007/1211952683.jpg)

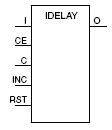
BUFDS原语的例化代码模板如下所示：   
  
// IBUFDS: 差分输入缓冲器（Differential Input Buffer）   
// 适用芯片：Virtex-II/II-Pro/4, Spartan-3/3E   
// Xilinx HDL库向导版本，ISE 9.1   
IBUFDS #(   
.DIFF\_TERM("FALSE"),   
// 差分终端，只有**Virtex-4**系列芯片才有，可设置为True/Flase   
.IOSTANDARD("DEFAULT")   
// 指定输入端口的电平标准，如果不确定，可设为DEFAULT   
) IBUFDS\_inst (   
.O(O), // 时钟缓冲输出   
.I(I), // 差分时钟的正端输入，需要和顶层模块的端口直接连接   
.IB(IB) // 差分时钟的负端输入，需要和顶层模块的端口直接连接   
);   
// 结束IBUFDS模块的例化过程   
  
在综合结果分析时，IBUFDS的RTL结构如图3-39所示。

http://www.openhw.org/data/08-03/4155_1205985007/1211952737.jpg  
  
图3-39 IBUFDS原语的RTL结构图

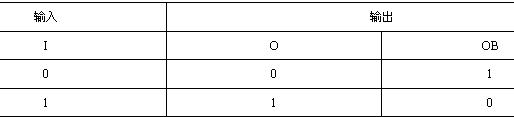
3. IDELAY   
  
在**Virtex-4**系列芯片中，每个用户I/O管脚的输入通路都有一个IDELAY模块，可用于数据信号或时钟信号，以使二者同步，准确采集输入数据。IDELAY具有可控的64抽头延迟线，每个抽头的延迟都是经过精密校准的78ps，且与进程、电压和温度特性无关，其内部结构如图3-40所示。

  
  
  
图3-40 IDELAY模块的RTL级结构图

IDELAY原语的信号说明如下：   
  
I：单比特输入信号，来自于IOB的串行输入数据；   
C：单比特输入信号，时钟输入信号；   
INC：单比特输入信号，用于增加或减少延迟抽头数；   
CE：单比特输入信号，使能延迟抽头数增加或减少的功能；   
RST：单比特输入信号，复位延迟链的延迟抽头数，如果没有编程输入，则为0；   
O：单比特输出信号。   
IDELAY原语的例化代码模板如下所示：   
// IDELAY: 输入延迟单元（Input Delay Element）   
// 适用芯片：Virtex-II/II-Pro/4, Spartan-3/3E   
// Xilinx HDL库向导版本，ISE 9.1   
IDELAY #(   
.IOBDELAY\_TYPE("DEFAULT"),   
// 输入延迟类型，可设置为 "DEFAULT", "FIXED" 或者 "VARIABLE"   
.IOBDELAY\_VALUE(0)   
// 输入延迟周期数，可设置为0~63之间的任意整数   
) IDELAY\_inst (   
.O(O), //1比特输出信号   
.C(C), // 1比特时钟输入信号   
.CE(CE), // 1比特时钟使能信号   
.I(I), // 1比特数据输入信号   
.INC(INC), // 1比特增量输入信号   
.RST(RST) //1比特复位输入信号   
);   
// 结束IDELAY模块的例化过程   
  
在综合结果分析时，IDELAY原语的RTL结构如图3-41所示。

  
  
图3-41 IDELAY原语的RTL结构图

4. OBUFDS原语   
  
OBUFDS将标准单端信号转换成差分信号，输出端口需要直接对应到顶层模块的输出信号，和IBUFDS为一对互逆操作。OBUFDS原语的真值表如表3-13所列。

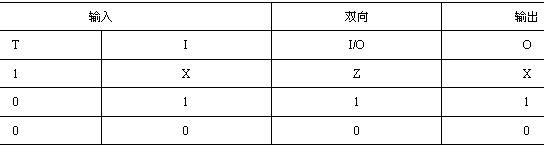
表3-13 OBUFDS原语的真值表   
  


OBUFDS原语的例化代码模板如下所示：   
  
// OBUFDS: 差分输出缓冲器（Differential Output Buffer）   
// 适用芯片：Virtex-II/II-Pro/4, Spartan-3/3E   
// Xilinx HDL库向导版本，ISE 9.1   
OBUFDS #(   
.IOSTANDARD("DEFAULT")   
// 指名输出端口的电平标准   
) OBUFDS\_inst (   
.O(O), // 差分正端输出，直接连接到顶层模块端口   
.OB(OB), // 差分负端输出，直接连接到顶层模块端口   
.I(I) // 缓冲器输入   
);   
// 结束OBUFDS模块的例化过程   
在综合结果分析时，OBUFDS原语的RTL结构如图3-42所示。

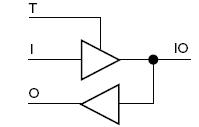
http://www.openhw.org/data/08-03/4155_1205985007/1211952990.jpg  
  
图3-42 OBUFDS的RTL结构图

5. IOBUF原语   
  
IOBUF原语是单端双向缓冲器，其I/O接口必须和指定的电平标准相对应，支持 LVTTL、LVCMOS15、LVCMOS18、LVCMOS25以及LVCMOS33等信号标准，同时还可通过DRIVE、 FAST 以及SLOW等约束来满足的不同驱动和抖动速率的需求。默认的驱动能力为12mA，低抖动。IOBUF由IBUF和OBUFT两个基本组件构成，当I/O端口为高阻时，其输出端口O为不定态。IOBUF原语的功能也可以通过其组成组件的互联来实现。   
  
IOBUF原语的输入输出真值表如表3-14所列。

表3-14 IOBUF原语的真值表

[](http://www.openhw.org/data/08-03/4155_1205985007/1211953089.jpg)

IOBUF原语的例化代码模板如下所示：   
  
// IOBUF: 单端双向缓冲器（Single-ended Bi-directional Buffer）   
// 适用芯片：所有芯片   
// Xilinx HDL库向导版本，ISE 9.1   
IOBUF #(   
.DRIVE(12),   
// 指定输出驱动的强度   
.IOSTANDARD("DEFAULT"),   
// 指定I/O电平的标准，不同的芯片支持的接口电平可能会有所不同   
.SLEW("SLOW")   
// 制定输出抖动速率   
) IOBUF\_inst (   
.O(O), // 缓冲器的单元输出   
.IO(IO), // 缓冲器的双向输出   
.I(I), // 缓冲器的输人   
.T(T) // 3态使能输入信号   
);   
// 结束IOBUF模块的例化过程   
  
在综合结果分析时，IOBUF原语的RTL结构如图3-43所示。

  
  
图3-43 OBUFDS的RTL结构图

6. PULLDOWN和PULLUP原语   
  
数字电路有三种状态：高电平、低电平、和高阻状态。有些应用场合不希望出现高阻状态，可以通过上拉电阻或下拉电阻的方式使其处于稳定状态，如图3-44所示。FPGA的I/O端口，可以通过外接电阻上下拉，也可以在芯片内部，通过配置完成上下拉。上拉电阻是用来解决总线驱动能力不足时提供电流的，而下拉电阻是用来吸收电流。通过FPGA内部配置完成上下拉，能有效节约电路板面积，是设计的首选方案。

  
  
图3-44 上、下拉电路示意图

上、下拉的原语分别为PULLUP和PULLDOWN。   
  
1）PULLUP原语的例化代码   
  
// PULLUP: 上拉原语（I/O Buffer Weak Pull-up）   
// 适用芯片：所有芯片   
// Xilinx HDL库向导版本，ISE 9.1   
PULLUP PULLUP\_inst (   
.O(O),   
//上拉输出，需要直接连接到设计的顶层模块端口上);   
// 结束PULLUP模块的例化过程   
  
2）PULLDOWN原语的例化代码   
  
// PULLDOWN:下拉原语（ I/O Buffer Weak Pull-down）   
// 适用芯片：所有芯片   
// Xilinx HDL库向导版本，ISE 9.1   
PULLDOWN PULLDOWN\_inst (   
.O(O),   
// 下拉输出，需要直接连接到设计的顶层模块端口上   
);   
// 结束PULLDOWN模块的例化过程   
  
**3.4.6 处理器组件**   
  
处理器组件主要包括高速以太网MAC控制器和PowerPC硬核，如表3-15所示。

表3-15处理器组件列表   
  
