

# Xilinx 原语的使用方法 1

## 3.4 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。

### 3.4.1 计算组件

计算组件指的就是 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

| 原语名   | 描述                                                 |
|-------|----------------------------------------------------|
| 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 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 registers
.CEM(CEM), // Clock Enable input for multiplier registers
.CEP(CEP), // Clock Enable input for P registers
.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

| 原语名             | 描述                       |
|-----------------|--------------------------|
| BUFG            | 全局时钟缓冲器                  |
| BUFGCE          | 全局时钟复用器，附带时钟使能信号和0状态输出   |
| BUFGCE_1        | 全局时钟复用缓冲器，附带时钟使能信号和1状态输出 |
| BUFGCTRL        | 全局时钟复用缓冲器                |
| BUFGMUX         | 全局时钟复用缓冲器，附带时钟使能信号和0状态输出 |
| BUFMUX_1        | 全局时钟复用器，附带0状态输出          |
| BUFGMUX_VIRTEX4 | Virtex-4器件特有的全局时钟复用缓冲器   |
| BUFIO           | I/O端口本地时钟缓冲器             |
| BUFR            | I/O端口和CLB的本地时钟缓冲器        |
| DCM_ADV         | 带有高级特征的数字时钟管理模块          |
| DCM_BASE        | 带有基本特征的数字时钟管理模块          |
| DCM_PS          | 带有基本特征和移相特征的数字时钟管理模块     |
| PMCD            | 匹配相位时钟分频器                |

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_inst 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。

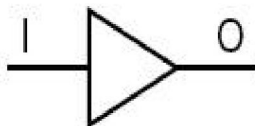


图 3-32 全局时钟原语的 RTL 级结构示意图

## 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：

```
// 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：该原语只用于全局时钟处理，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。

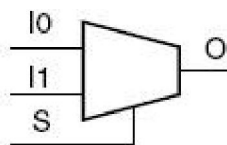


图 3-33 全局时钟复用器的 RTL 级结构示意图

### 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。

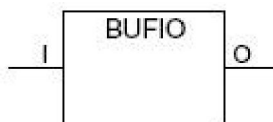


图 3-34 本地 I/O 时钟缓冲器的 RTL 级结构示意图

### 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: 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。

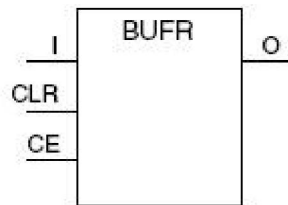


图 3-35 本地 I/O 时钟、逻辑缓冲器的 RTL 级结构示意图

## 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

| 管脚名          | 方向   | 位宽 | 简要描述                                                                                                                                       |
|--------------|------|----|--------------------------------------------------------------------------------------------------------------------------------------------|
| 相关的时钟输入、输出管脚 |      |    |                                                                                                                                            |
| CLK0         | 输出信号 | 1  | 输出和 DCM 模块输入管脚 CLKIN 有效频率相同的时钟。如果不选中 CLKIN_DIVIDE_BY_2 选项，在默认情况下，其大小等于 CLKIN。如果连接了 CLKFB，那么 CLK0 与 CLKIN 的相位也是对准的。                         |
| CLK90        | 输出信号 | 1  | CLK90 输出时钟的频率和 CLK0 大小一致，只是在相位上偏移了 90 度。                                                                                                   |
| CLK180       | 输出信号 | 1  | CLK180 输出时钟的频率和 CLK0 大小一致，只是在相位上偏移了 180 度，也就是 CLK0 的反相信号。                                                                                  |
| CLK270       | 输出信号 | 1  | CLK270 输出时钟的频率和 CLK0 大小一致，只是在相位上偏移了 270 度。                                                                                                 |
| CLK2X        | 输出信号 | 1  | CLK2X 输出时钟的频率为 CLK0 的两倍，相位亦和 CLK0 时对齐，且占空比为 50%。在 DCM 模块为达到锁相的稳定状态前，其占空比为 1/3，用来调整 DCM 模块锁相到正确的变化沿上。                                       |
| CLK2X180     | 输出信号 | 1  | CLK2X180 输出时钟的频率和 CLK2X 大小一致，只是在相位上有 180 度的偏移，即 CLK2X 的反相信号                                                                                |
| CLKDV        | 输出信号 | 1  | CLKDV 输出时钟的频率为 CLK0 频率的分数，分频比由 CLKDV_DIVIDE 参数决定。                                                                                          |
| CLKFX        | 输出信号 | 1  | CLKFX 输出时钟的频率由下式决定：<br>$\text{CLKFX 频率} = \text{CLKIN 有效频率} \times (M/D)$<br>其中，M 是倍频比，D 是分频比。当 CLKFB 管脚连接时，其相位和 CLK0、CLK2X 以及 CLKDV 是一致的。 |
| CLKFX180     | 输出信号 | 1  | CLKFX180 的时钟频率和 CLKFX 频率大小一致，只是移相了 180 度。                                                                                                  |
| CLKIN        | 输入信号 | 1  | 源时钟输入信号，其大小必须满足一定的范围，具体数值需要参考不同芯片的数据手册。CLKIN 的输入时钟必须来自下列缓存器：<br>1. 全局时钟输入缓存（IBUFG）<br>2. 内部全局时钟缓冲器（BUFG/BUFGCTRL）<br>3. 输入缓存器（IBUF）        |
| LOCKED       | 输出信号 | 1  | PLL 的同步输出，用于指示锁相环是否已完成锁相，可以完成用户操作                                                                                                          |
| RST          | 输入信号 | 1  | DCM 模块的复位信号，高有效。在复位期间，所有的输出管脚都为低电平。                                                                                                        |

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。

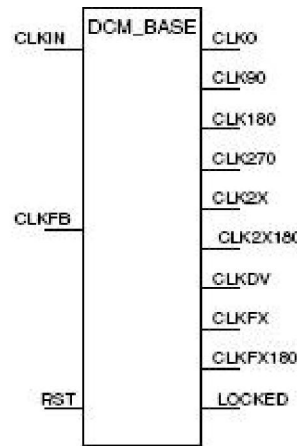


图 3-36 DCM 模块的 RTL 级结构示意图

### 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

| 原语                 | 描述                                   |
|--------------------|--------------------------------------|
| BSCAN_VIRTEX4      | 提供到VIRTEX-4边界扫描点的接入                  |
| CAPTURE_VIRTEX4    | VIRTEX-4边界扫描控制逻辑电路                   |
| FRAME_ECC_VIRTEX4  | 读入一帧VIRTEX-4配置数据，并能完成汉明、单错误纠正和双错误检测  |
| ICAP_VIRTEX4       | VIRTEX-4内部配置接入端口                     |
| STARTUP_VIRTEX4    | VIRTEX-4配置时钟、全局复位、全局3态控制和其他配置信号的用户接口 |
| USR_ACCESS_VIRTEX4 | 带有32位数据总线和有效数据指示端口的32比特寄存器           |

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。

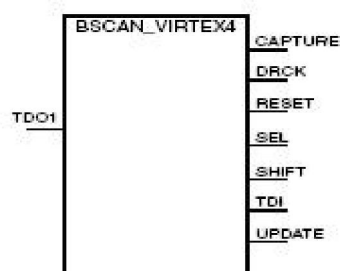


图 3-37 BSCAN\_VIRTEX4 原语的 RTL 图

### 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

| 原语          | 描述                                                                                       |
|-------------|------------------------------------------------------------------------------------------|
| GT11_CUSTOM | 速度范围可达622Mbps到11.1Gbps的RocketIO MGT,单片FPGA可提供8到24个高速收发器,VCO的频率高达2.5GHz-5.55GHz,且抖动小于1ns。 |
| GT11_DUAL   | RocketIO MGT Tile,包含两个GT11_CUSTOM                                                        |
| GT11CLK     | 能够完成差分包输入时钟、来自于Fabric的参考时钟以及驱动MGTs列两个垂直参考时钟总线的rxbclk三者时分复用的复用器                           |
| GT11CLK_MGT | 允许差分包输入来驱动MGTs列的两个垂直参考时钟总线                                                               |

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

| 原语         | 描述                      |
|------------|-------------------------|
| BUFIO      | I/O的本地时钟缓存              |
| DCIRESER   | FPGA配置成功后DCI状态机的复位信号    |
| IBUF       | 标准和容量可选择I/O单端口输入缓存      |
| IBUFDS     | 带可选择端口的差分信号输入缓存         |
| IBUFG      | 带可选择端口的专用输入缓存           |
| IBUFGDS    | 带可选择端口的专用差分信号输入缓存       |
| IDDR       | 用于接收外部DDR输入信号的专用输入寄存器   |
| IDELAY     | 专用的可变抽头输入延迟链            |
| IDELAYCTRL | IDELAY抽头数的控制模块          |
| IOBUF      | 带可选择端口的双向缓存             |
| IOBUFDS    | 低有效输出的三态差分信号I/O缓存       |
| ISERDES    | 专用I/O缓存的输入分解器           |
| KEEPER     | KEEPER符号                |
| OBUF       | 单端输出端口缓存                |
| OBUFT      | 带可选择端口的低有效输出的三态输出缓冲     |
| OBUFDS     | 带可选择端口的差分信号输出缓冲         |
| OBUFTDS    | 带可选择端口的低有效输出的三态差分输出缓冲   |
| ODDR       | 用于向外部DDR发送信号的专用输出寄存器    |
| OSERDES    | 用于快速实现输入源同步接口           |
| PULLDOWN   | 输入端寄存器下拉至0              |
| PULLUP     | 输入端寄存器、开路以及三态输出端口上拉至VCC |

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。

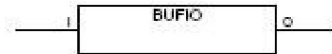


图 3-38 BUFIO 的 RTL 结构图

### 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

| 输入 |    | 输出 |
|----|----|----|
| I  | IB | O  |
| 0  | 0  | -* |
| 0  | 1  | 0  |
| 1  | 0  | 1  |
| 1  | 1  | -* |

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/False
    .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。

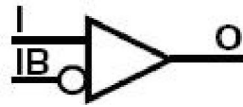


图 3-39 IBUFDS 原语的 RTL 结构图

### 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。

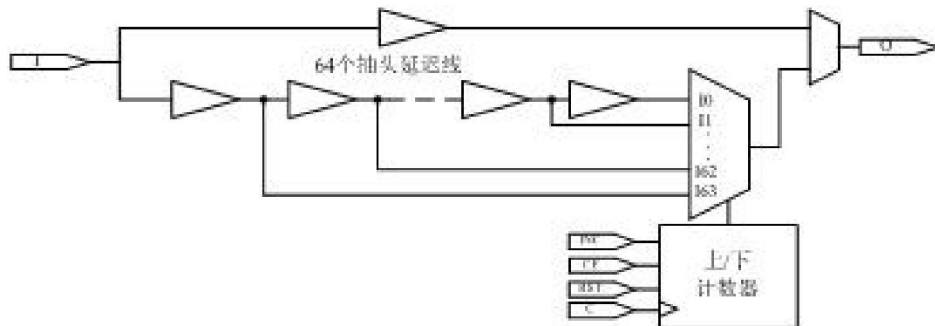


图 3-40 IDELAY 模块的 RTL 级结构图

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。

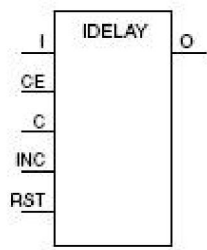


图 3-41 IDELAY 原语的 RTL 结构图

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

| 输入 | 输出 |    |
|----|----|----|
| I  | O  | OB |
| 0  | 0  | 1  |
| 1  | 1  | 0  |

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。

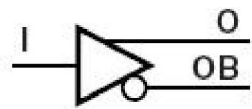


图 3-42 OBUFDS 的 RTL 结构图

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 LVTTTL、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

| 输入 |   | 双向  | 输出 |
|----|---|-----|----|
| T  | I | I/O | O  |
| 1  | X | Z   | X  |
| 0  | 1 | 1   | 1  |
| 0  | 0 | 0   | 0  |

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), // 缓冲器的输入
```

```
.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。

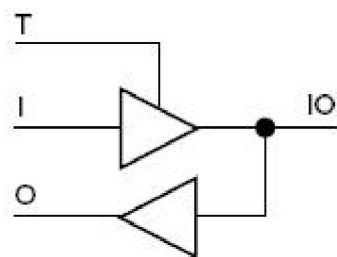


图 3-43 OBUFDS 的 RTL 结构图

## 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

| 原语             | 描述                               |
|----------------|----------------------------------|
| EMAC           | 集成的10/100/1000Mbps以太网<br>MAC层控制器 |
| PPC405_AD<br>V | PowerPC核的原语                      |

处理器组件是 Xilinx 的嵌入式解决方案，涉及软、硬件调试以及系统设计，将在第 8 章详细介绍。

## Xilinx 公司原语的使用方法 2

### 3.4.7 RAM/ROM 组件

RAM/ROM 组件可用于例化 FIFO、分布式 RAM、分布式 RAM、块 ROM 以及块 ROM，具体包括 12 个组件，如表 3-16 所示。

表 3-16 RAM/ROM 原语列表

| 原语             | 描述                                                                  |
|----------------|---------------------------------------------------------------------|
| FIFO16         | 基于 Virtex-4 块 RAM 的内嵌 FIFO                                          |
| RAM16X1D       | 深度为 16、位宽为 1 的静态同步双口 RAM                                            |
| RAM16X1S       | 深度为 16、位宽为 1 的静态同步 RAM                                              |
| RAM32X1S       | 深度为 32、位宽为 1 的静态同步 RAM                                              |
| RAM64X1S       | 深度为 64、位宽为 1 的静态同步 RAM                                              |
| RAMB16         | 单口块 RAM，位宽可配置成 1，2，4，9，18 或 36，其大小可配置成 16384 比特的数据存储，或者 2048 的奇偶存储器 |
| RAMB32_S64_ECC | 带有差错处理的深度为 64、位宽为 64 的同步双口块 RAM                                     |
| ROM16X1        | 深度为 16、位宽为 1 的 ROM                                                  |
| ROM32X1        | 深度为 32、位宽为 1 的 ROM                                                  |
| ROM64X1        | 深度为 64、位宽为 1 的 ROM                                                  |
| ROM128X1       | 深度为 128、位宽为 1 的 ROM                                                 |
| ROM256X1       | 深度为 256、位宽为 1 的 ROM                                                 |

下面主要介绍 FIFO、分布式双口 RAM 以及块双口 RAM 原语的使用，单口 RAM 和 ROM 原语的用法类似，限于篇幅，就不再介绍。

#### 1. RAM16X1S

RAM16X1S 是深度为 16 比特，位宽为 1 的同步 RAM。当写使能信号 WE 为低时，写端口的数据操作无效，RAM 内部的数据不会改变；当 WE 为高时，可以在任意地址中写入比特。为了保证数据的稳定性，地址和数据应该在 WCLK 的上升沿前保持稳定。输出信号 O 为 RAM 中由读地址信号所确定的地址中所存数据的值。此外，还可通过属性指定 RAM 的初始值。

RAM16X1S 原语的例化代码模版如下所示：

```
// RAM16X1S: 16 比特 1 深度同步 RAM (16 x 1 posedge write distributed (LUT) RAM)
// 适用芯片：所有芯片
// Xilinx HDL 库向导版本，ISE 9.1
RAM16X1S #(
.INIT(16'h0000))
```

```

//对 RAM 的内容进行初始化，这里初始化为全 1
) RAM16X1S_inst (
.O(O), // RAM output
.A0(A0), // RAM address[0] input
.A1(A1), // RAM address[1] input
.A2(A2), // RAM address[2] input
.A3(A3), // RAM address[3] input
.D(D), // RAM data input
.WCLK(WCLK), // Write clock input
.WE(WE) // Write enable input
);
// 结束 RAM16X1S 模块的例化过程

```

需要注意的是，RAM16X1S 原语是 Xilinx 独有的一类结构，在小数据量存储方面非常节省资源。在综合结果分析时，RAM16X1S 原语的 RTL 结构如图 3-45 所示。

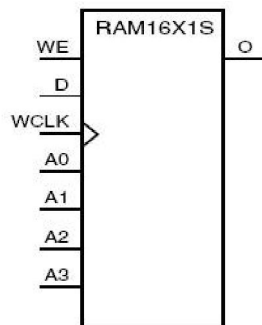


图 3-45 原语 RAM16X1S 的 RTL 结构图

## 2. RAMB16

RAMB16 是 FPGA 芯片中内嵌的双口块 RAM，数据位宽可配置成 1、2、4、9、18 以及 36 比特，每个块 RAM 的大小为 18 1024 比特，所以位宽越大，深度越小。块 RAM 在 FPGA 中按照矩阵的方式排列，其数量完全取决于芯片容量的大小。在使用中，可以添加坐标来约束块 RAM 的位置。例如：

```
LOC = RAMB16_X#Y#;
```

同样，也可以对块 RAM 完成初始化。块 RAM 是以硬核的方式内嵌到 FPGA 芯片中，不占用芯片的逻辑资源，是 FPGA 芯片内部非常宝贵的一种资源。在工作时，要尽量使用芯片的块 RAM 资源，不仅能保证较高的工作频率，还具有有很低的动态功耗。

RAMB16 的 Verilog 例化代码如下所示。

```

// RAMB16: 块 RAM (Virtex-4 16k+2k Parity Paramatizable BlockRAM
// 适用芯片：Virtex-4 芯片
// Xilinx HDL 库向导版本，ISE 9.1

```

[illegible]

[illegible]

```
.INIT_17(256'h0000000000000000000000000000000000000000),
.INIT_18(256'h0000000000000000000000000000000000000000),
.INIT_19(256'h0000000000000000000000000000000000000000),
.INIT_1A(256'h0000000000000000000000000000000000000000),
.INIT_1B(256'h0000000000000000000000000000000000000000),
.INIT_1C(256'h0000000000000000000000000000000000000000),
.INIT_1D(256'h0000000000000000000000000000000000000000),
.INIT_1E(256'h0000000000000000000000000000000000000000),
.INIT_1F(256'h0000000000000000000000000000000000000000),
.INIT_20(256'h0000000000000000000000000000000000000000),
.INIT_21(256'h0000000000000000000000000000000000000000),
.INIT_22(256'h0000000000000000000000000000000000000000),
.INIT_23(256'h0000000000000000000000000000000000000000),
.INIT_24(256'h0000000000000000000000000000000000000000),
.INIT_25(256'h0000000000000000000000000000000000000000),
.INIT_26(256'h0000000000000000000000000000000000000000),
.INIT_27(256'h0000000000000000000000000000000000000000),
.INIT_28(256'h0000000000000000000000000000000000000000),
.INIT_29(256'h0000000000000000000000000000000000000000),
.INIT_2A(256'h0000000000000000000000000000000000000000),
.INIT_2B(256'h0000000000000000000000000000000000000000),
.INIT_2C(256'h0000000000000000000000000000000000000000)
```

[illegible]





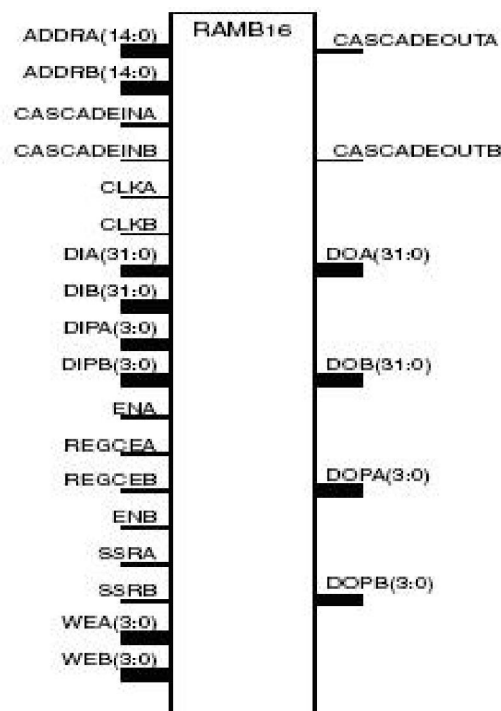


图 3-46 RAMB16 的 RTL 结构示意图

### 3.4.8 寄存器和锁存器

寄存器和锁存器是时序电路中常用的基本元件，其原语在表 3-17 中给出。

表 3-17 寄存器和锁存器原语列表

| 原语    | 描述                          |
|-------|-----------------------------|
| FDCPE | 带有时钟使能、异步预配置和清空信号的 D 触发器    |
| FDRSE | 带有同步时钟使能、同步预配置和清空信号的 RS 触发器 |
| LCDPE | 带有时钟使能、异步预配置和清空信号的透明数据锁存器   |

下面只介绍 FDCPE 原语的使用，其余两个原语的使用方法类似。

#### 1. FDCPE 原语

FDCPE 指带单数据输入信号 D、单输出 O、时钟使能信号 CE、异步复位 PRE 和异步清空信号 CLR 的 D 触发器。当 PRE 信号为高时，输出端 O 为高；当 CLR 为高时，输出端 O 为低；当 PRE 和 CLR 都为低、CE 信号为高时，输入信号 D 在时钟上升沿被加载到触发器中，并被送到输出端；当 PRE 和 CLR 都为低、CE 信号为低时，输出端保持不变。FDCPE 的真值表如表 3-18 所列。

表 3-18 FDCPE 原语的真值表

| 输入信号 |     |    |   |     | 输出信号 |
|------|-----|----|---|-----|------|
| CLR  | PRE | CE | D | C   | Q    |
| 1    | X   | X  | X | X   | 0    |
| 0    | 1   | X  | X | X   | 1    |
| 0    | 0   | 0  | X | X   | 保持不变 |
| 0    | 0   | 1  | 0 | 上升沿 | 0    |
| 0    | 0   | 1  | 1 | 上升沿 | 1    |

FDCPE 原语的 Verilog 实例化代码如下所示：

```
// FDCPE: D 触发器 (Single Data Rate D Flip-Flop )
// 适用芯片：所有 FPGA 芯片
// Xilinx HDL 库向导版本，ISE 9.1
FDCPE #(
.INIT(1'b0)
//初始化寄存器的值，可设置为 1'b0 或 1'b1
) FDCPE_inst (
.Q(Q), // 数据输出端口
.C(C), // 时钟输入端口
.CE(CE), // 时钟使能输入信号
.CLR(CLR), // 异步清空输入信号
.D(D), // 数据输入信号
.PRE(PRE) // 异步复位输入信号
);
// 结束 FDCPE 模块的例化过程
```

在综合结果分析时，FDCPE 原语的 RTL 结构如图 3-47 所示。

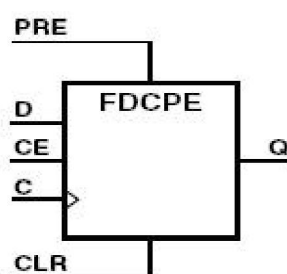


图 3-47 FDCPE 的 RTL 结构

### 3.4.9 移位寄存器组件

移位寄存器组件为 Xilinx 芯片所独有，由于属性的不同，具体有 8 个，如表 3-19 所列。

表 3-19 移位寄存器组件列表

| 原语       | 描述                                   |
|----------|--------------------------------------|
| SRL16    | 16 比特移位寄存器查找表                        |
| SRL16_1  | 时钟下降作用的 16 比特移位寄存器查找表                |
| SRL16E   | 带有时钟使能信号的 16 比特移位寄存器查找表              |
| SRL16E_1 | 带有时钟使能信号且在时钟下降沿作用的 16 比特移位寄存器查找表     |
| SRLC16   | 带有进位的 16 比特移位寄存器查找表                  |
| SRLC16_1 | 带有进位且在时钟下降沿作用的 16 比特移位寄存器查找表         |
| SRLC6E   | 带有时钟使能和进位信号的 16 比特移位寄存器查找表           |
| SRLC6E_1 | 带有时钟使能和进位信号，且在时钟下降沿作用的 16 比特移位寄存器查找表 |

各个移位寄存器原语都是在 SRL16 的基础上发展起来的，因此，本节主要介绍基本的 SRL16 原语。

### 1. SRL16 原语

SRL16 是基于查找表（LUT）的移位寄存器，在实际应用中既能节省资源，还能保证时序。其输入信号 A3、A2、A1 以及 A0 选择移位寄存器的读取地址，当写使能信号高有效时，输入信号将被加载到移位寄存器中。单个移位寄存器的深度可以是固定的，也可以动态调整，最大不能超过 16。需要更大深度的移位寄存器时，则需要将多个 SRL16 拼接起来。单个 SRL16 的地址信号线与输出的真值表如表 3-20 所示。

表 3-20 SRL16 的功能说明表

| 输入             |        |   | 输出                   |
|----------------|--------|---|----------------------|
| A <sub>m</sub> | CLK    | D | Q                    |
| A <sub>m</sub> | X      | X | Q(A <sub>m</sub> )   |
| A <sub>m</sub> | ↑（上升沿） | D | Q(A <sub>m-1</sub> ) |

m=0,1,2,3

SRL16 本质上是一种基于查找表（LUT）的移位寄存器，因此其位宽（B）和深度（D）需要满足查找表的结构特点，所占用 Slice 资源 M 的计算公式为：

$$M = B(R[D/16] + 1)$$

其中，R[]函数的意义是取整。从上式可以看出，深度为 1 的移位寄存器和深度为 16 的移位寄存器所占用的 Slice 资源是一样的。和触发器相比，SRL16 最大的特点就是占用 Slice 资源特别少。

只要固定 A3~A0 的信号电平，即可获得一个固定长度的移位寄存器，其长度范围从 1~16，可由公式（M）计算得到。当 A3~A0 全为 0 时，其寄存深度为 1；当 A3~A0 全为 1 时，其寄存深度为 16。

$$\text{Length} = (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1 (M)$$

SRL16 原语的 Verilog 实例化代码如下所示：

```
// SRL16: 16 位查找表移位寄存器 (16-bit shift register LUT operating on
posedge of clock)
// 适用芯片：所有 FPGA 芯片
// Xilinx HDL 库向导版本，ISE 9.1
SRL16 #(
.INIT(16'h0000)
// 初始化移位寄存器的值，可以为 16 比特任意整数
) SRL16_inst (
.Q(Q), // SRL16 的数据输出信号
.A0(A0), // 选择[0]输入
.A1(A1), // 选择[1]输入
.A2(A2), // 选择[2]输入
.A3(A3), // 选择[3]输入
.CLK(CLK), // 时钟输入信号
.D(D) // SRL16 的数据输入信号
);
// 结束 SRL16 模块的例化过程
```

在综合结果分析时，SRL16 原语的 RTL 级结构如图 3-48 所示。

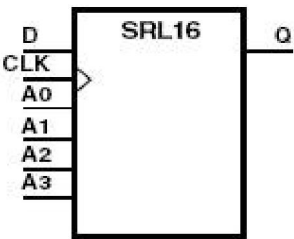


图 3-48 SRL16 的 RTL 级结构示意图

### 3.4.10 Slice/CLB 组件

Slice/CLB 组件涵盖了 Xilinx FPGA 中所有的逻辑单元，包括各种查找表、复用器以及逻辑操作等 21 个原语，如表 3-21 所列。

表 3-21 Slice/CLB 组件列表

| 原语       | 描述                     |
|----------|------------------------|
| BUFCF    | 快速连接缓冲                 |
| LUT1     | 带通用输出的 1 比特查找表         |
| LUT2     | 带通用输出的 2 比特查找表         |
| LUT3     | 带通用输出的 3 比特查找表         |
| LUT4     | 带通用输出的 4 比特查找表         |
| LUT1_D   | 带两个输出的 1 比特查找表         |
| LUT2_D   | 带两个输出的 2 比特查找表         |
| LUT3_D   | 带两个输出的 3 比特查找表         |
| LUT4_D   | 带两个输出的 4 比特查找表         |
| LUT1_L   | 带本地输出的 1 比特查找表         |
| LUT2_L   | 带本地输出的 2 比特查找表         |
| LUT3_L   | 带本地输出的 3 比特查找表         |
| LUT4_L   | 带本地输出的 4 比特查找表         |
| MULT_AND | 用于乘法的快速与门              |
| MUXCY    | 带有进位和通用输出的 2 到 1 复用器   |
| MUXCY_D  | 带有进位和双输出的 2 到 1 复用器    |
| MUXCY_L  | 带有进位和本地输出的 2 到 1 复用器   |
| MUXF5    | 基于查找表的 2 到 1 复用器，带通用输出 |
| MUXF5_D  | 基于查找表的 2 到 1 复用器，带双输出  |
| MUXF5_L  | 基于查找表的 2 到 1 复用器，带本地输出 |
| MUXF6    | 基于查找表的 2 到 1 复用器，带通用输出 |
| MUXF6_D  | 基于查找表的 2 到 1 复用器，带双输出  |
| MUXF6_L  | 基于查找表的 2 到 1 复用器，带本地输出 |
| MUXF7    | 基于查找表的 2 到 1 复用器，带通用输出 |
| MUXF7_D  | 基于查找表的 2 到 1 复用器，带双输出  |
| MUXF7_L  | 基于查找表的 2 到 1 复用器，带本地输出 |
| MUXF8    | 基于查找表的 2 到 1 复用器，带通用输出 |
| MUXF8_D  | 基于查找表的 2 到 1 复用器，带双输出  |
| MUXF8_L  | 基于查找表的 2 到 1 复用器，带本地输出 |
| XORCY    | 带通用输出进位逻辑的 XOR         |
| XORCY_D  | 带两个输出进位逻辑的 XOR         |
| XORCY_L  | 带本地输出进位逻辑的 XOR         |

本节主要介绍基本的单输出 1 比特查找表原语和通用输出 2 到 1 复用器原语的使用方法。

### 1. LUT1 原语

FPGA 内部的组合逻辑都可以通过 LUT 结构实现，常用的 LUT 结构有 LUT1、

LUT2、LUT3 以及 LUT4，其区别在于查找表输入比特宽度的不同。LUT1 是最简单的一种，常用于实现缓冲器和反相器，是带通用输出的 1 比特查找表。

LUT1 原语的 Verilog 实例化代码如下所示：

```
// LUT1: 1 比特输入的通用查找表 (1-input Look-Up Table with general
output)
// 适用芯片：所有 FPGA 芯片
// Xilinx HDL 库向导版本，ISE 9.1
LUT1 #(
  .INIT(2'b00)
  // 指定 LUT 的初始化内容
) LUT1_inst (
  .O(O), // LUT 的通用输出
  .I0(I0) // LUT 输入信号
);
// 结束 LUT1 模块的例化过程
```

在综合结构分析时，LUT1 原语的 RTL 级结构如图 3-49 所示。

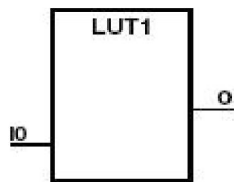


图 3-49 LUT1 的 RTL 级结构示意图

2. MUXF7 原语

MUXF7 是带通用输出的 2 到 1 查找表复用器，其控制信号 S 可以是任何内部信号，当其为低电平时，选择 I0；当其为高电平时选择 I1。MUXF7 需要占用一个完整的 CLB 资源，和其相关的查找表可组成 7 功能的查找表和 16 到 1 的选择器，可将 MUXF6 的输出连接到 MUXF7 的输入端。MUXF7 的真值表如表 3-22 所示。

表 3-22 MUXF7 的真值表

| 输入 |    |    | 输出 |
|----|----|----|----|
| S  | I0 | I1 | O  |
| 0  | I0 | X  | I0 |
| 1  | X  | I1 | I1 |
| X  | 0  | 0  | 0  |
| X  | 1  | 1  | 1  |

MUXF7 原语的实例化代码如下所示：

```
// MUXF7: 2 到 1LUT 复用器 (CLB MUX to tie two MUXF6's together with  
general output)  
// 适用芯片：Virtex-II/II-Pro 以及 Spartan-3/3E  
// Xilinx HDL 库向导版本，ISE 9.1  
MUXF7 MUXF7_inst (  
.O(O), // MUX 的通用输出信号  
.I0(I0),  
.I1(I1), // 复用器的输入信号，连接到 MUXF6 LO  
.S(S) // 复用器的输入选择信号  
);  
// End of MUXF7_inst instantiation
```

在综合结构分析时，MUXF7 原语的 RTL 级结构如图 3-50 所示。

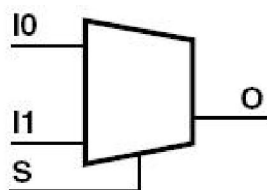


图 3-50 MUXF7 的 RTL 级结构示意图

本节给出了典型的 Xilinx 原语的使用，但更多的原语需要读者在实践中大量应用才能熟练掌握。只有全面掌握了原语，才能对 Xilinx FPGA 的资源有一个理性认识，从而将工程设计提升到艺术雕刻的高度，这是每一个 FPGA 开发人员的目标。详细的原语资料可在 ISE 安装目录下的“doc\usenglish\books\docs”文件夹中找到，其中 Virtex4 系列芯片的原语资料文件“v4ldl.pdf”位于“v4ldl”文件中，Virtex5 系列芯片的原语资料文件“v5ldl.pdf”位于“v5ldl”文件中，Spartan 3A 系列芯片的原语资料文件“s3adl.pdf”位于“s3adl”文件中，Spartan 3E 系列芯片的原语资料文件“s3edl.pdf”位于“s3edl”文件中。

### 3.5 本章小结

本章主要介绍基于 Xilinx 芯片的 Verilog 开发技术。首先介绍了典型的硬件设计思维；然后讨论了优秀的通用代码风格和 Xilinx 专用的代码风格，无论哪种风格都是稳定性、速度以及硬件资源三者优化，只不过前者从 FPGA 器件共有的特性出发，后者则面向 Xilinx 公司的 FPGA 芯片特点；最后，介绍了 Xilinx 公司 FPGA 器件的硬件原语，以及典型原语的使用方法。特别需要强调的一点是：对原语的全面掌握是 Xilinx FPGA 开发人员的基本要求。