Arora V 时钟资源（Clock）

用户指南

UG306-1.0.4, 2024-08-09

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止反言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2023/04/20	1.0	初始版本。
2023/09/12	1.0.1	更新表 5-13 PLLA 参数介绍中 FCLKIN 的取值范围。
2024/02/02	1.0.2	更新 4.4.1 原语介绍，新增 DLLDLY 单步延时时间的描述。
2024/07/05	1.0.3	增加 PLL PhaseShift 应用描述。
2024/08/09	1.0.4	● 增加 OSCB 模块内容； ● 更新适用器件内容； ● 增加 DHCE 驱动连接规则。

目录

图目录

表目录

1 关于本手册

1.1 手册内容

本文档介绍了高云®半导体 Arora V FPGA 产品的时钟资源的功能、原语定义及使用方法。

1.2 相关文档

通过登录高云半导体网站 www.gowinsemi.com.cn 可以下载、查看以下相关文档：

- DS981, Arora V 138K & 75K FPGA 产品数据手册
- DS1225, Arora V 60K FPGA 产品数据手册
- DS1103, Arora V 25K FPGA 产品数据手册
- DS1118, Arora V 15K FPGA 产品数据手册
- SUG100，Gowin 云源软件用户指南

1.3 术语、缩略语

表 1-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
CLKDIV	Clock Divider	时钟分频器
CLKDIV2	Clock Divider 2	高速时钟分频器
DCE	Dynamic Clock Enable	动态时钟使能
DCS	Dynamic Clock Selector	动态时钟选择器
DHCE	Dynamic HCLK Clock Enable	动态高速时钟使能
DLLDLY	DLL Delay	DLL 延迟
DQS	Bidirectional Data Strobe Circuit for DDR Memory	DDR 存储器双向数据选通电路
GCLK	Global Clock	全局时钟
HCLK	High-speed Clock	高速时钟

术语、缩略语	全称	含义
LW	Long Wire	长线
OSC	Oscillator	晶体振荡器
PCLK	Primary Clock	主时钟
PLL	Phase-locked Loop	锁相环
SSC	Spread Spectrum Clocking	扩频时钟

1.4 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接与公司联系：

网址：www.gowinsemi.com

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391

2 概述

本章介绍了高云®半导体 Arora V FPGA 产品的时钟资源，包括专用的时钟输入以及时钟布线资源。时钟的基础设施提供了一系列低电容、低偏移互连线，非常适合承载高频信号，最大限度地减少时钟偏差和提高性能，可应用于所有的时钟信号。

时钟资源对 FPGA 高性能的应用至关重要。Arora V FPGA 产品提供了专用全局时钟网络 GCLK（包括 PCLK 和 LW），直接驱动器件全局。除了 GCLK 资源，还提供了锁相环（PLL）、高速时钟 HCLK 和 DDR 存储器接口数据脉冲时钟 DQS 等时钟资源。

2.1 全局时钟

GCLK 又分 PCLK 和 LW。PCLK 可实现对全局的驱动。LW 一方面可以用作控制线，给 DFF 提供时钟使能（CE）、置复位（SET/RESET）信号；另一方面，还可以用作逻辑绕线，作为普通数据信号使用。

2.2 高速时钟

Arora V FPGA 产品的高速时钟 HCLK，具有低抖动和低偏差性能，可以支持 I/O 完成高性能数据传输，是专门针对源时钟同步的数据传输接口而设计的。一个 Bank 支持四路 HCLK。

3 全局时钟

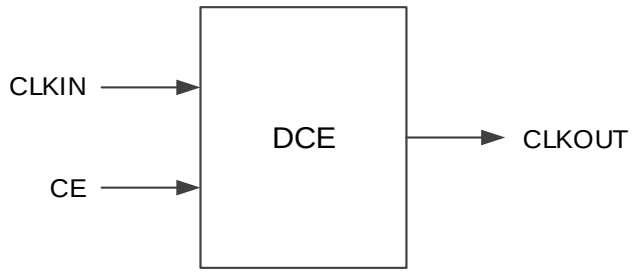
3.1 DCE

3.1.1 原语介绍

DCE 具有动态时钟控制功能，允许内部逻辑动态启用或禁用 GCLK 网络。当禁用 GCLK 时钟网络时，由该时钟驱动的所有逻辑都不再翻转，从而降低器件的总功耗。

端口示意图

图 3-1 DCE 端口示意图



端口介绍

表 3-1 DCE 端口介绍

端口名	I/O	描述
CLKIN	Input	时钟输入信号
CE	Input	时钟使能信号，高电平有效。
CLKOUT	Output	时钟输出信号

原语例化

可以直接实例化原语。

Verilog 例化：

```
DCE uut (
    .CLKIN(clkin),
```

```
        .CE(ce),
        .CLKOUT(clkout)
    );
```

VHDL 例化：

```
    COMPONENT DCE
        PORT(
            CLKOUT:OUT std_logic;
            CE:IN std_logic;
            CLKIN:IN std_logic
        );
    END COMPONENT;
    uut:DCE
    PORT MAP(
        CLKIN=>clkin,
        CLKOUT=>clkout,
        CE=>ce
    );
```

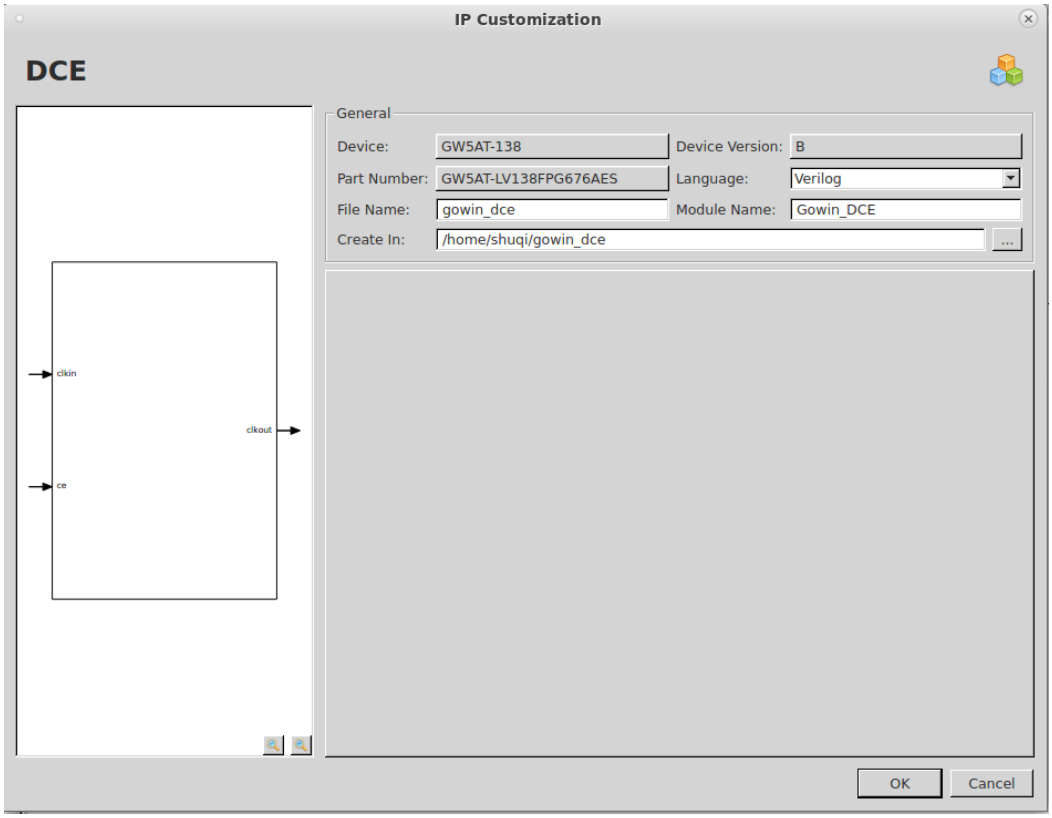
3.1.2 IP 调用

在 IP Core Generator 界面中单击“DCE”，界面右侧会显示 DCE 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“DCE”，弹出 DCE 的“IP Customization”窗口，该窗口包括“General”配置框和端口显示框图，如下图 3-2 所示。

图 3-2 DCE 的 IP Customization 窗口结构



1. General 配置框

General 配置框用于配置产生的 IP 设计文件的相关信息。

- Device：显示已配置的 Device 信息；
- Device Version：显示已配置的 Device Version 信息；
- Part Number：显示已配置的 Part Number 信息；
- Language：配置产生的 IP 设计文件的硬件描述语言。选择右侧下拉列表框，选择目标语言，支持 Verilog 和 VHDL；
- File Name：配置产生的 IP 设计文件的文件名。在右侧文本框可重新编辑文件名称；
- Module Name：配置产生的 IP 设计文件的 module name。在右侧文本框可重新编辑模块名称。Module Name 不能与原语名称相同，若相同，则报出 Error 提示；
- Create In：配置产生的 IP 设计文件的目标路径。可在右侧文本框中重新编辑目标路径，也可通过文本框右侧选择按钮选择目标路径。

2. 端口显示框图

端口显示框图显示 IP Core 的配置结果示例框图，如图 3-2 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_dce.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 DCE；
- IP 设计使用模板文件“gowin_dce_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_dce.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

3.2 DCS

3.2.1 原语介绍

DCS，动态时钟选择器，CLKOUT 可以在四个时钟输入之间进行动态切换。

功能描述

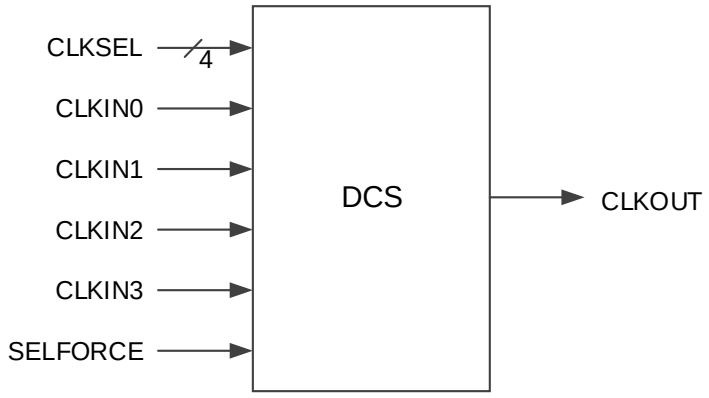
DCS 存在两种时钟切换模式，分别是“Non-Glitchless”和“Glitchless”模式。

在 Non-Glitchless 模式下，DCS 的作用类似于常规多路复用器，仅通过 CLKSEL 信号切换时钟信号，允许输出上的毛刺，实际情况取决于切换的时间。

在 Glitchless 模式下，通过参数 DCS_MODE 设置模式，配置 CLKSEL 信号动态切换时钟信号，可以避免输出时钟上的毛刺。

端口示意图

图 3-3 DCS 端口示意图



端口介绍

表 3-2 DCS 端口介绍

端口名	I/O	描述
CLKIN0	Input	时钟输入信号 0
CLKIN1	Input	时钟输入信号 1
CLKIN2	Input	时钟输入信号 2
CLKIN3	Input	时钟输入信号 3
CLKSEL	Input	时钟选择信号
SELFORCE	Input	强制模式选择 ● 0：glitchless 模式 ● 1：Non-glitchless 模式
CLKOUT	Output	时钟输出信号

参数介绍

表 3-3 DCS 参数介绍

参数名	取值范围	默认值	描述
DCS_MODE	"CLK0", "CLK1", "CLK2", "CLK3", "GND", "VCC", "RISING", "FALLING", "CLK0_GND", "CLK1_GND", "CLK2_GND", "CLK3_GND", "CLK0_VCC", "CLK1_VCC", "CLK2_VCC", "CLK3_VCC"	"RISING"	设置 DCS 模式

原语例化

可以直接实例化原语。

Verilog 例化：

```
DCS dcs_inst (
    .CLKIN0(clk0),
    .CLKIN1(clk1),
    .CLKIN2(clk2),
    .CLKIN3(clk3),
    .CLKSEL,
    .SELFORCE(selforce),
    .CLKOUT(clkout)
);
defparam dcs_inst.DCS_MODE=“RISING”;
```

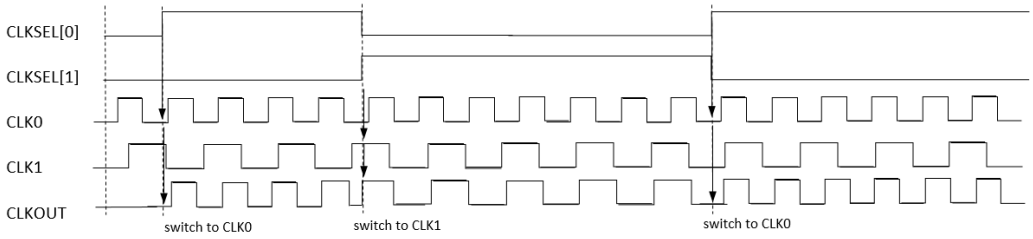
Vhdl 例化：

```
COMPONENT DCS
        GENERIC(DCS_MODE:string:="RISING");
            PORT(
                    CLK0:IN std_logic;
                    CLK1:IN std_logic;
                    CLK2:IN std_logic;
                    CLK3:IN std_logic;
                    CLKSEL:IN std_logic_vector(3 downto 0);
                    SELFORCE:IN std_logic;
                    CLKOUT:OUT std_logic
                );
END COMPONENT;
uut:DCS
        GENERIC MAP(DCS_MODE=>"RISING")
        PORT MAP(
            CLK0=>clk0,
            CLK1=>clk1,
            CLK2=>clk2,
            CLK3=>clk3,
            CLKSEL=>clksel,
            SELFORCE=>selforce,
            CLKOUT=>clkout
);
```

时序图

Non-Glitchless 模式时序如图 3-4 所示，CLKSEL[3]~CLKSEL[0]分别对应选择 CLKIN3~CLKIN0，高电平有效，转换时序相同。

图 3-4 Non-Glitchless 模式时序图



Glitchless 模式时序如图 3-5、图 3-6、图 3-7 和图 3-8 所示，用

CLKSEL[3]~CLKSEL[0]分别对应选择 CLKIN3~CLKIN0，转换时序相同。

图 3-5 DCS mode：RISING 时序图

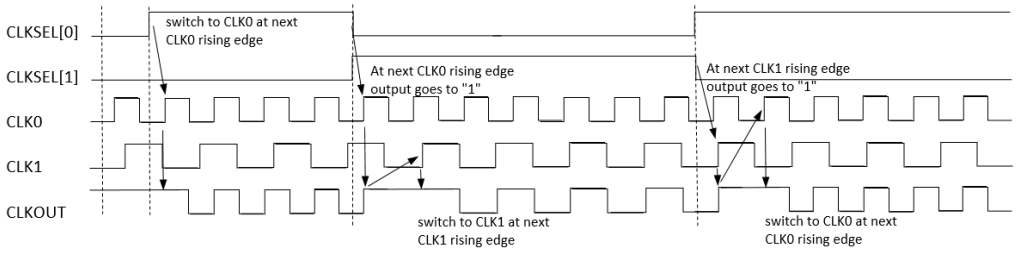


图 3-6 DCS mode：FALLING 时序图

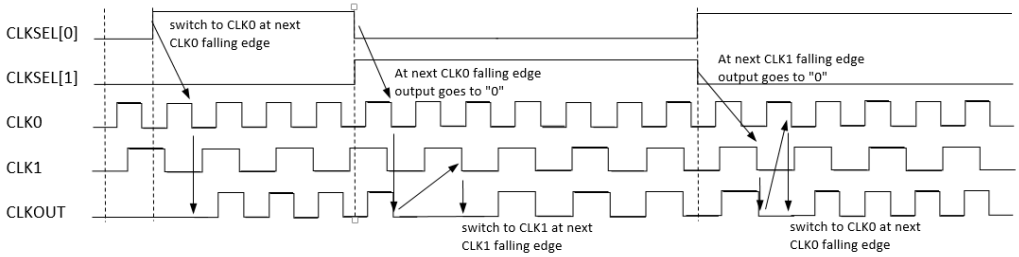


图 3-7 DCS mode：CLK0_GND 时序图

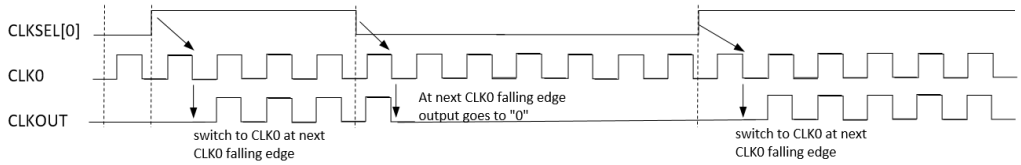
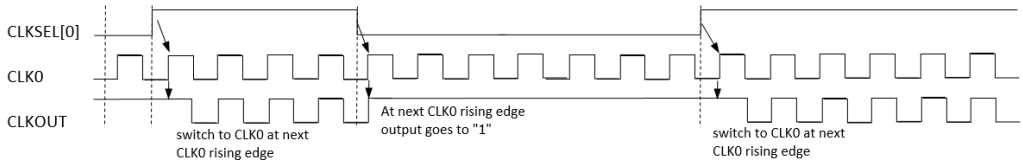


图 3-8 DCS mode：CLK0_VCC 时序图



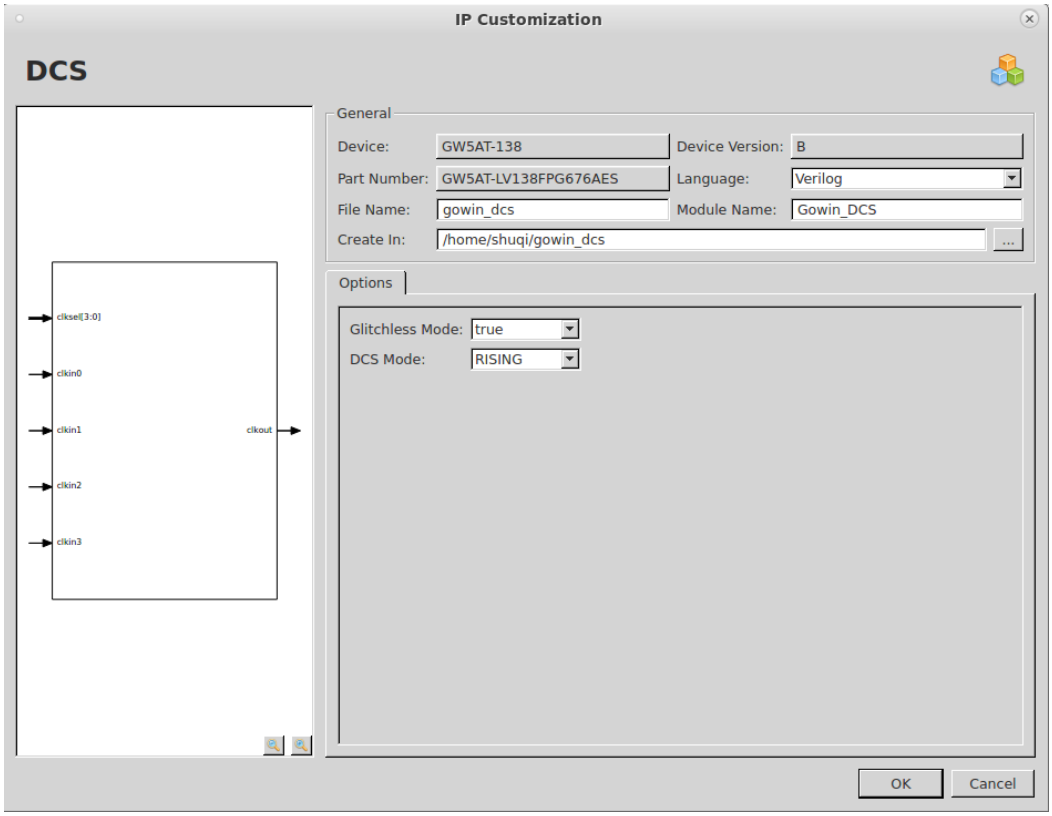
3.2.2 IP 调用

在 IP Core Generator 界面中单击“DCS”，界面右侧会显示 DCS 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“DCS”，弹出 DCS 的“IP Customization”窗口，该窗口包括“General”配置框、“Options”配置框和端口显示框图，如图 3-9 所示。

图 3-9 DCS 的 IP Customization 窗口结构



1. General 配置框
   General 配置框用于配置产生的 IP 设计文件的相关信息。DCS 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。
2. Options 配置框
   Options 配置框用于用户自定义配置 IP，Options 配置框如图 3-9 所示。
   - Glitchless Mode：使能/失能 Glitchless 模式。
   - DCS Mode：设置 DCS 模式。
3. 端口显示框图
   端口显示框图显示 IP Core 的配置结果示例框图，如图 3-9 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_dcs.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 DCS；
- IP 设计使用模板文件“gowin_dcs_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_dcs.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

4 高速时钟

4.1 DHCE

4.1.1 原语介绍

DHCE 可动态地打开/关闭 HCLK 高速时钟信号。

端口示意图

图 4-1 DHCE 端口示意图

端口介绍

表 4-1 DHCE 端口介绍

端口名	I/O	描述
CLKIN	input	时钟输入信号
CEN	input	时钟使能输入信号，低电平有效。
CLKOUT	output	时钟输出信号

连接规则

- DHCE 的输出可连接至 IOLOGIC 的 FCLK；
- DHCE 的输出可连接至 CLKDIV 的 HCLKIN；
- DHCE 的输出可连接至 DQS 的 FCLK；
- DHCE 的输出可连接至 PLL/PLLA 的 CLKIN、CLKFB；
- DHCE 的输出可连接至 DDRDLL 的 CLKIN；
- DHCE 的输出可连接至 CLKDIV2 的 HCLKIN。

原语例化

可以直接实例化原语。

Verilog 例化：

```
DHCE uut (
        .CLKIN(clkin),
        .CEN(cen),
        .CLKOUT(clkout)
);
```

Vhdl 例化：

```
COMPONENT DHCE
        PORT(
                CLKOUT:OUT std_logic;
                CEN:IN std_logic;
                CLKIN:IN std_logic
        );
END COMPONENT;
uut:DHCE
PORT MAP(
    CLKIN=>clkin,
    CLKOUT=>clkout,
    CEN=>cen
);
```

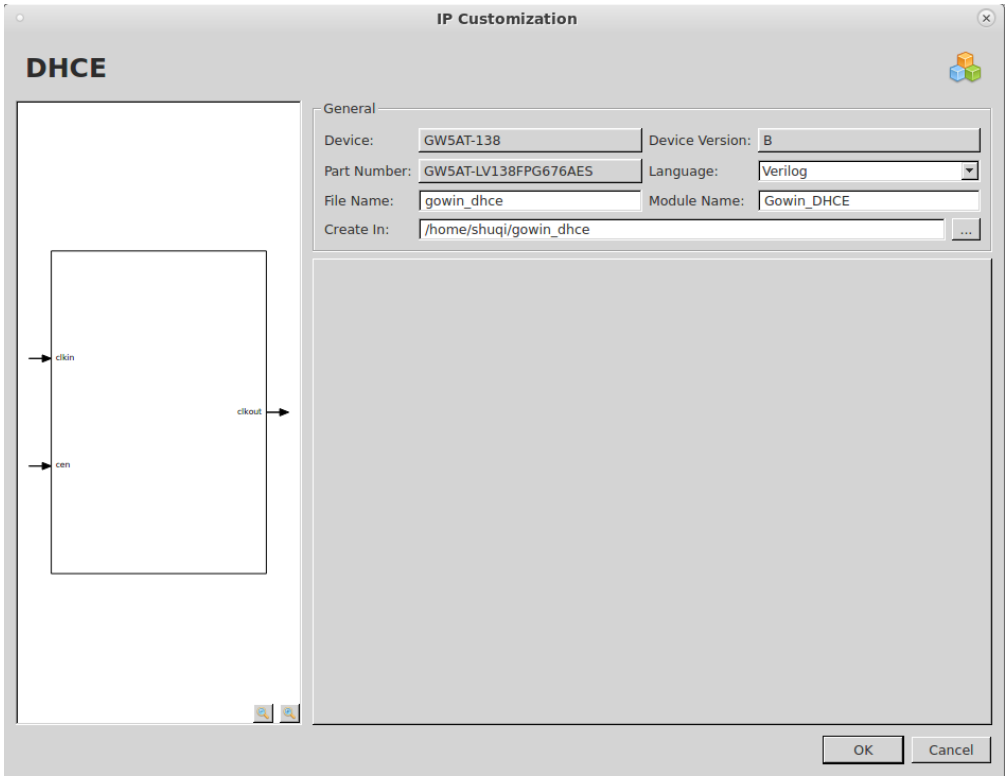
4.1.2 IP 调用

在 IP Core Generator 界面中单击“DHCE”，界面右侧会显示 DHCE 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“DHCE”，弹出 DHCE 的“IP Customization”窗口，该窗口包括“General”配置框和端口显示框图，如图 4-2 所示。

图 4-2 DHCE 的 IP Customization 窗口结构



1. General 配置框
   General 配置框用于配置产生的 IP 设计文件的相关信息。DHCE 的 General 配置框的使用与 DCE 模块类似，请参考 3.1.2 IP 调用。
2. 端口显示框图
   端口显示框图显示 IP Core 的配置结果示例框图，如图 4-2 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_dhce.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 DHCE；
- IP 设计使用模板文件“gowin_dhce_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_dhce.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

4.2 CLKDIV2

4.2.1 原语介绍

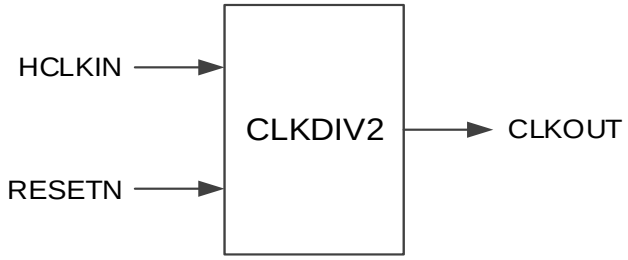
CLKDIV2 为时钟分频器，实现时钟的二分频调整。CLKDIV2 的输出只能驱动 IOLOGIC 的 FCLK、PLL 的 CLKIN 和 CLKFB、DQS 的 FCLK、CLKDIV 的 HCLKIN 和 DDRDLL 的 CLKIN。

功能描述

CLKDIV2 为高速时钟分频模块，生成与输入时钟相位一致的 2 分频时钟。

端口示意图

图 4-3 CLKDIV2 端口示意图



端口介绍

表 4-2 CLKDIV2 端口介绍

端口名	I/O	描述
HCLKIN	Input	时钟输入信号
RESETN	Input	异步复位信号，低电平有效。
CLKOUT	Output	时钟输出信号

原语例化

可以直接实例化原语。

Verilog 例化：

```
CLKDIV2 uut (
    .HCLKIN(hclkin),
    .RESETN(resetn),
    .CLKOUT(clkout)
);
```

VHDL 例化：

```
COMPONENT CLKDIV2
    PORT(
        HCLKIN:IN std_logic;
```

```
                RESETN:IN std_logic;
                CLKOUT:OUT std_logic
                );
END CONPONENT;
    uut:CLKDIV2
            PORT MAP(
                HCLKIN=>hclkin,
                RESETN=>resetn,
                CLKOUT=>clkout
            );
```

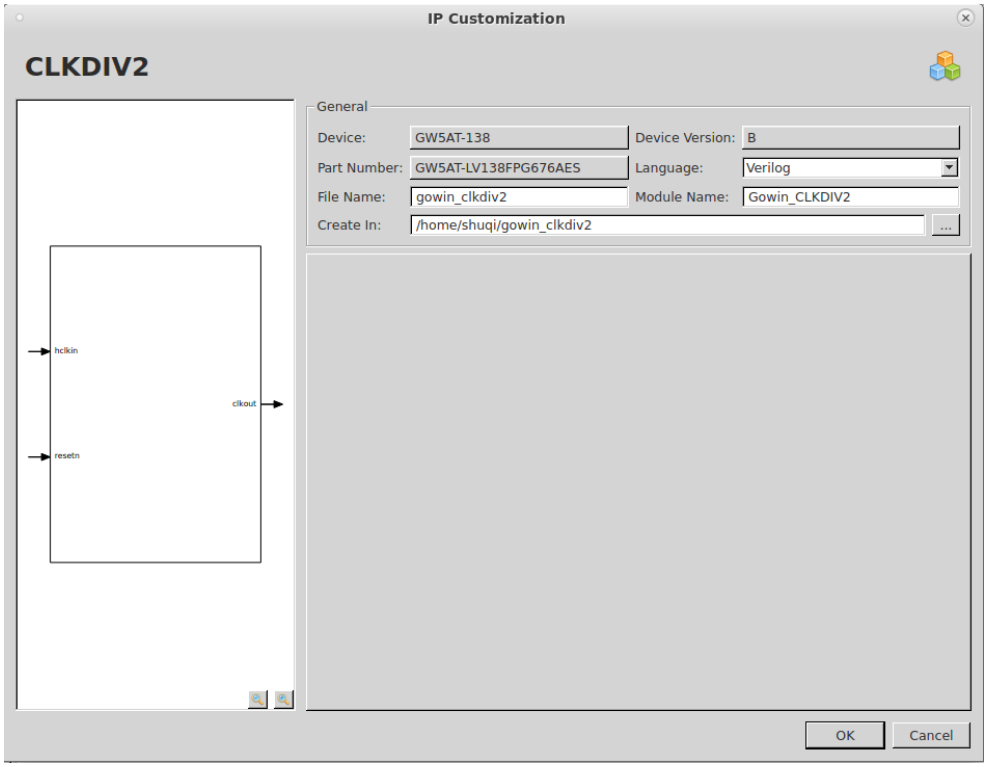
4.2.2 IP 调用

在 IP Core Generator 界面中单击“CLKDIV2”，界面右侧会显示 CLKDIV2 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“CLKDIV2”，弹出 CLKDIV2 的 "IP Customization”窗口，该窗口包括“General”配置框和端口显示框图，如图 4-4 所示。

图 4-4 CLKDIV2 的 IP Customization 窗口结构



3. General 配置框

General 配置框用于配置产生的 IP 设计文件的相关信息。CLKDIV2 的

General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。

4. 端口显示框图
   端口显示框图显示 IP Core 的配置结果示例框图，如图 4-4 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_clkdiv2.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 CLKDIV2；
- IP 设计使用模板文件“gowin_clkdiv2_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_clkdiv2.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

4.3 CLKDIV

4.3.1 原语介绍

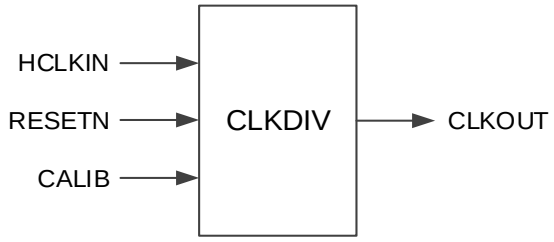
CLKDIV 为时钟分频器，实现时钟频率调整。

功能描述

CLKDIV 为高速时钟分频模块，生成和输入时钟相位一致的分频时钟，用于 IO 逻辑。

端口示意图

图 4-5 CLKDIV 端口示意图



端口介绍

表 4-3 CLKDIV 端口介绍

端口名	I/O	描述
HCLKIN	Input	时钟输入信号
RESETN	Input	异步复位信号，低电平有效，其中对 DIV_MODE 为 1 不起作用。
CALIB	Input	CALIB 输入信号，调整输出时钟。
CLKOUT	Output	时钟输出信号

其中，CALIB 信号可以和 IOLOGIC 中的 CALIB 配合使用，具体功能如下：

- 2 分频时，每 2 个下降沿调整一次相位，每次调整 180 度，2 次调整为一个周期；
- 3 分频时，每 2 个下降沿调整一次相位，每次调整约 120 度，3 次调整为一个周期；
- 3.5 分频时，每 1 个下降沿调整一次相位，每次调整约 102.8 度，7 次调整为一个周期；
- 4 分频时，每 2 个下降沿调整一次相位，每次调整 90 度，4 次调整为一个周期；
- 5 分频时，每 2 个下降沿调整一次相位，每次调整约 72 度，5 次调整为一个周期；
- 6 分频时，每 2 个下降沿调整一次相位，每次调整约 60 度，6 次调整为一个周期；
- 7 分频时，每 2 个下降沿调整一次相位，每次调整约 51.4 度，7 次调整为一个周期；
- 8 分频时，每 2 个下降沿调整一次相位，每次调整约 45 度，8 次调整为一个周期；
- 1 分频时，调整无效。

参数介绍

表 4-4 CLKDIV 参数介绍

参数名	取值范围	默认值	描述
DIV_MODE	1,2,3,3.5,4,5,6,7,8	2	设置时钟分频系数

原语例化

可以直接实例化原语。

Verilog 例化：

```
CLKDIV uut (
    .HCLKIN(hclkin),
    .RESETN(resetn),
    .CALIB(calib),
    .CLKOUT(clkout)
);
defparam clkdiv_inst.DIV_MODE="2";
```

VHDL 例化：

```
COMPONENT CLKDIV
        GENERIC(
            DIV_MODE:STRING:="2"
            );
        PORT(
            HCLKIN:IN std_logic;
            RESETN:IN std_logic;
            CALIB:IN std_logic;
            CLKOUT:OUT std_logic
         );
END CONPONENT;
  uut:CLKDIV
        GENERIC MAP(
                DIV_MODE=>"2"
                )
        PORT MAP(
                HCLKIN=>hclkin,
                RESETN=>resetn,
                CALIB=>calib,
                CLKOUT=>clkout
          );
```

4.3.2 IP 调用

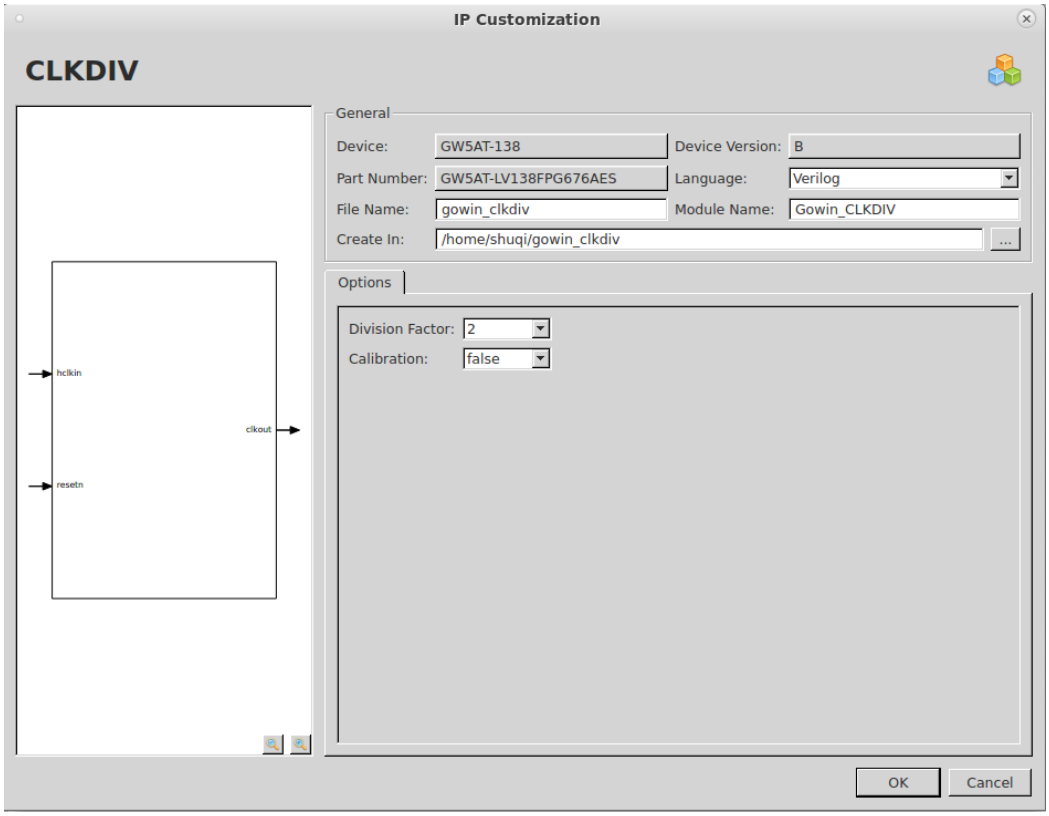
IP Core Generator 界面中单击“CLKDIV”，界面右侧会显示 CLKDIV 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“CLKDIV”，弹出 CLKDIV 的“IP Customization”窗口，该窗口包括“General”配置框、“Options”配置框

和端口显示框图，如图 4-6 所示。

图 4-6 CLKDIV 的 IP Customization 窗口结构



1. General 配置框
   General 配置框用于配置产生的 IP 设计文件的相关信息。CLKDIV 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。
2. Options 配置框
   Options 配置框用于用户自定义配置 IP，Options 配置框如图 4-6 所示。
   - Division Factor：除法因子。
   - Calibration：校准时钟使能/失能选项。
3. 端口显示框图
   端口显示框图显示 IP Core 的配置结果示例框图，如图 4-6 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_clkdiv.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 CLKDIV；
- IP 设计使用模板文件“gowin_clkdiv_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_clkdiv.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

4.4 DLLDLY

4.4.1 原语介绍

DLLDLY 为时钟延时模块，依据 CSTEP 或 DLLSTEP 信号对输入时钟进行调整，得到该时钟的延时调整输出。

功能描述

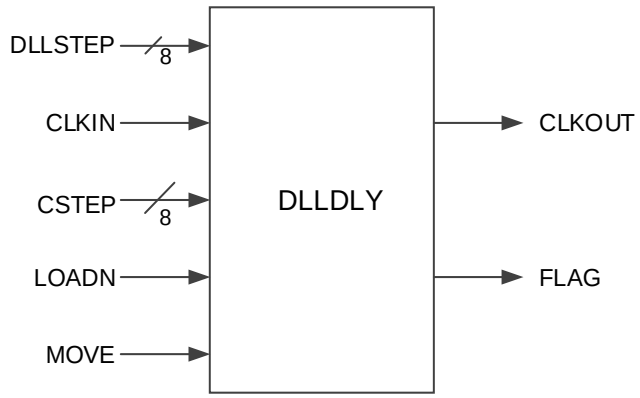
DLLDLY 根据 CSTEP 或 DLLSTEP 产生对应相位的延时，得到基于 CLKIN 的延时输出。根据 CSTEP 或 DLLSTEP 计算延时时，单步的延时时间约为 12.5ps。延时调整支持静态模式、动态模式和自适应模式。

注！

GW5A-25 A 版/ GW5AS-25 A 版/ GW5AR-25 A 版器件不支持自适应模式。

端口示意图

图 4-7 DLLDLY 端口示意图



端口介绍

表 4-5 DLLDLY 端口介绍

端口名	I/O	描述
CLKOUT	Output	时钟输出信号
FLAG	Output	自适应模式时输出标志，用以表示动态调整延时的 under-flow 或 over-flow。
DLLSTEP[7:0]	Input	延时步长输入信号，来源为 DDRDLL。
CLKIN	Input	时钟输入信号
CSTEP[7:0]	Input	可来自 CIU 绕线的延时步长输入信号
LOADN	Input	控制加载延时步长 ● 0: 加载延时步长； ● 1: 动态调整延时

端口名	I/O	描述
MOVE	Input	自适应模式时，MOVE 为下降沿时动态调整延时，每个脉冲移动一个延时步长。

参数介绍

表 4-6 DLLDLY 参数介绍

参数名	参数类型	取值范围	默认值	描述
DLY_SIGN	Binary	1'b0,1'b1	1'b0	设置调整延时的符号: 1'b0:'+' 1'b1: '-'
DLY_ADJ	Integer	0~255	0	延时调整设置: DLY_SIGN =0 DLY_ADJ; DLY_SIGN =1 - DLY_ADJ
STEP_SEL	Binary	1’b0,1’b1	1’b0	1’b0:DLLSTEP 1’b1:CSTEP
ADAPT_EN	String	"FALSE"、"TRUE"	"FALSE"	自适应模式使能
DYN_DLY_EN	String	"FALSE"、"TRUE"	"FALSE"	动态模式使能

原语例化

可以直接实例化原语。

Verilog 例化：

```
DLLDLY uut (
    .CLKIN(clkin),
    .DLLSTEP(dllstep[7:0]),
    .CSTEP(cstep[7:0]),
    .LOADN(loadn),
    .MOVE(move),
    .CLKOUT(clkout),
    .FLAG(flag)
);
defparam dlldly_0.DLY_SIGN=1'b0;
defparam dlldly_0.DLY_ADJ=0;
defparam dlldly_0.DYN_DLY_EN="FALSE";
defparam dlldly_0.ADAPT_EN="FALSE";
```

```
defparam dlldly_0.STEP_SEL=1'b0;
```

VHDL 例化：

```
COMPONENT DLLDLY
        GENERIC(
                DLY_SIGN:bit:='0';
                DLY_ADJ:integer:=0;
                DYN_DLY_EN : string := "FALSE" ;
                ADAPT_EN : string := "FALSE" ;
                STEP_SEL:bit:='0'
         );
            PORT(
                DLLSTEP:IN std_logic_vector(7 downto 0);
                CLKIN:IN std_logic;
                CSTEP:IN std_logic_vector(7 downto 0);
                LOADN,MOVE:IN std_logic;
                CLKOUT:OUT std_logic;
                FLAG:OUT std_logic
              );
END COMPONENT;
uut:DLLDLY
   GENERIC MAP(
                 DLY_SIGN=>'0',
                 DLY_ADJ=>0,
                 DYN_DLY_EN=> "FALSE",
                 ADAPT_EN=> "FALSE",
                 STEP_SEL=>'0'
    )
   PORT MAP(
      DLLSTEP=>dllstep,
      CLKIN=>clkin,
      CSTEP=>cstep,
      LOADN=>loadn,
      MOVE=>move,
      CLKOUT=>clkout,
      FLAG=>flag
```

);

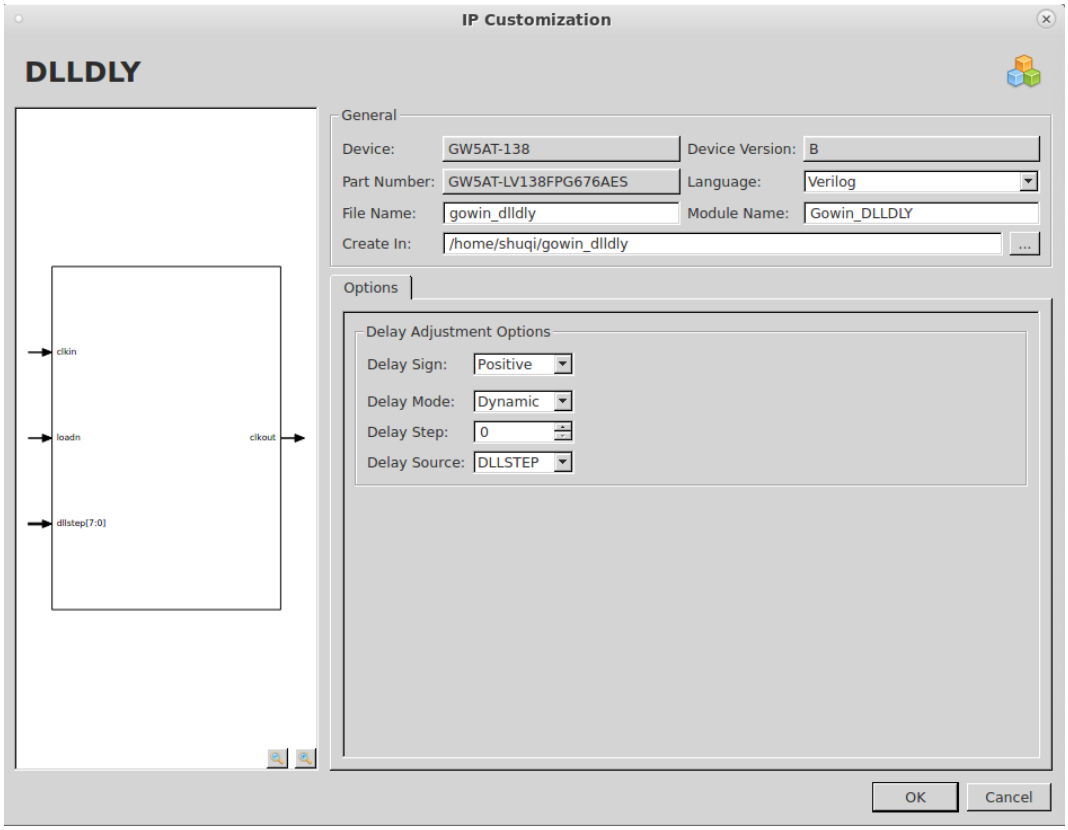
4.4.2 IP 调用

在 IP Core Generator 界面中单击“DLLDLY”，界面右侧会显示 DLLDLY 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“DLLDLY”，弹出 DLLDLY 的“IP Customization”窗口，该窗口包括“General”配置框、“Options”配置框和端口显示框图，如图 4-8 所示

图 4-8 DLLDLY 的 IP Customization 窗口结构



1. General 配置框
   General 配置框用于配置产生的 IP 设计文件的相关信息。DLLDLY 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。
2. Options 配置框
   Options 配置框用于用户自定义配置 IP，Options 配置框如图 4-8 所示。
   - Delay Sign：设置调整延时的符号。
   - Delay Mode：设置延时模式
   - Delay Step：设置延时步长

- Delay Source：延时来源

3. 端口显示框图
   端口显示框图显示 IP Core 的配置结果示例框图，如图 4-8 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_dlldly.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 DLLDLY；
- IP 设计使用模板文件“gowin_dlldly_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_dlldly.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

5 系统时钟

5.1 PLL

5.1.1 原语介绍

Arora V FPGA 提供了锁相环 PLL，支持 7 路时钟输出，每路时钟可独立基于给定的参考输入时钟进行时钟频率、相位和占空比调整。

适用器件

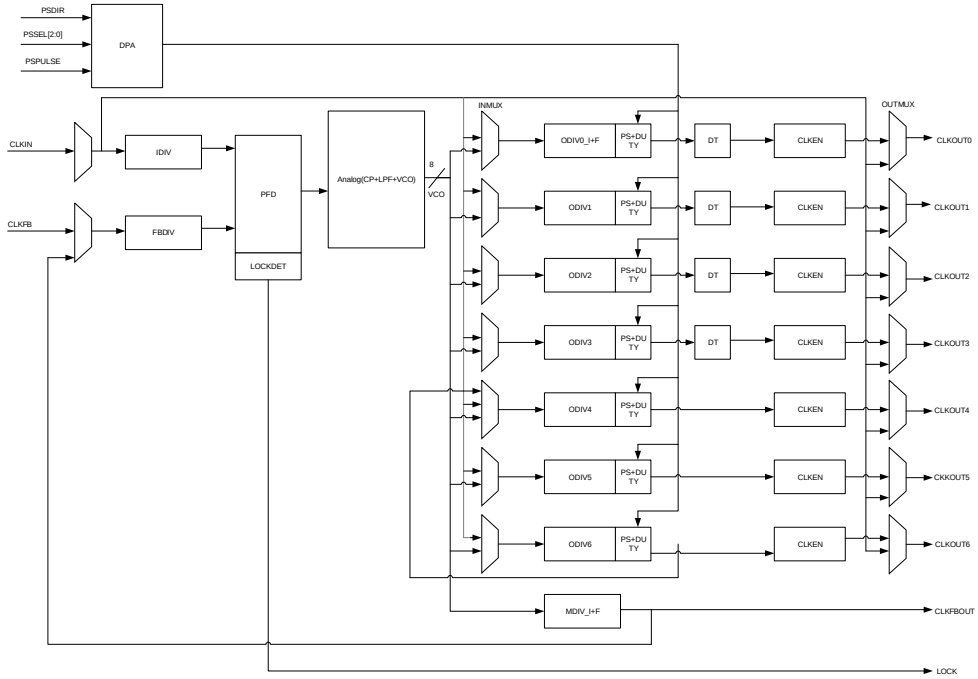
表 5-1 PLL 适用器件

家族	系列	器件
晨熙®	GW5AT	GW5AT-138, GW5AT-75
	GW5AST	GW5AST-138
	GW5A	GW5A-138
	GW5AS	GW5AS-138

功能描述

PLL 模块的结构框图如图 5-1 所示。

图 5-1 PLL 示意图



PLL 可基于给定的参考输入时钟进行时钟频率调整、相位调整、占空比调整来产生不同频率、相位和占空比的输出时钟。CLKOUT0 和 CLKFBOUT 支持 1/8 小数频率调整，CLKOUT0~CLKOUT3 支持动静态微调占空比。同时 PLL 支持 CLKOUT6 到 CLKOUT4 的内部级联，支持 SSC 功能，支持时钟去歪斜以实现 CLKIN 和 CLKOUT 的对齐。

若要得到正确的时钟输出，输入时钟频率必须按照 *FPGA 产品数据手册*中描述的频率范围进行设置。

PLL 可以对输入时钟 CLKIN 进行频率调整（倍频和分频），计算公式如下：

1. Fpfd=Fclkin/IDIV
2. Fclkfb=Fpfd*FBDIV
3. 根据不同反馈方式，VCO 频率计算公式不同:

- 内部反馈

  Fvco=Fclkfb*MDIV

- 外部反馈

  Fvco=Fclkfb*MDIV ---- CLKFBOUT 反馈到 CLKFB

  Fvco=Fclkfb*ODIVx ---- CLKOUTx 反馈到 CLKFB

4. Fclkfbout= Fvco/MDIV
5. 根据 INMUX 和 OUTMUX 选择不同的模式，CLKOUT 通道输出频率计算公式不同:

- VCO in 模式（INMUX 选择来自 VCO）：Fclkoutx=Fvco/ODIVx

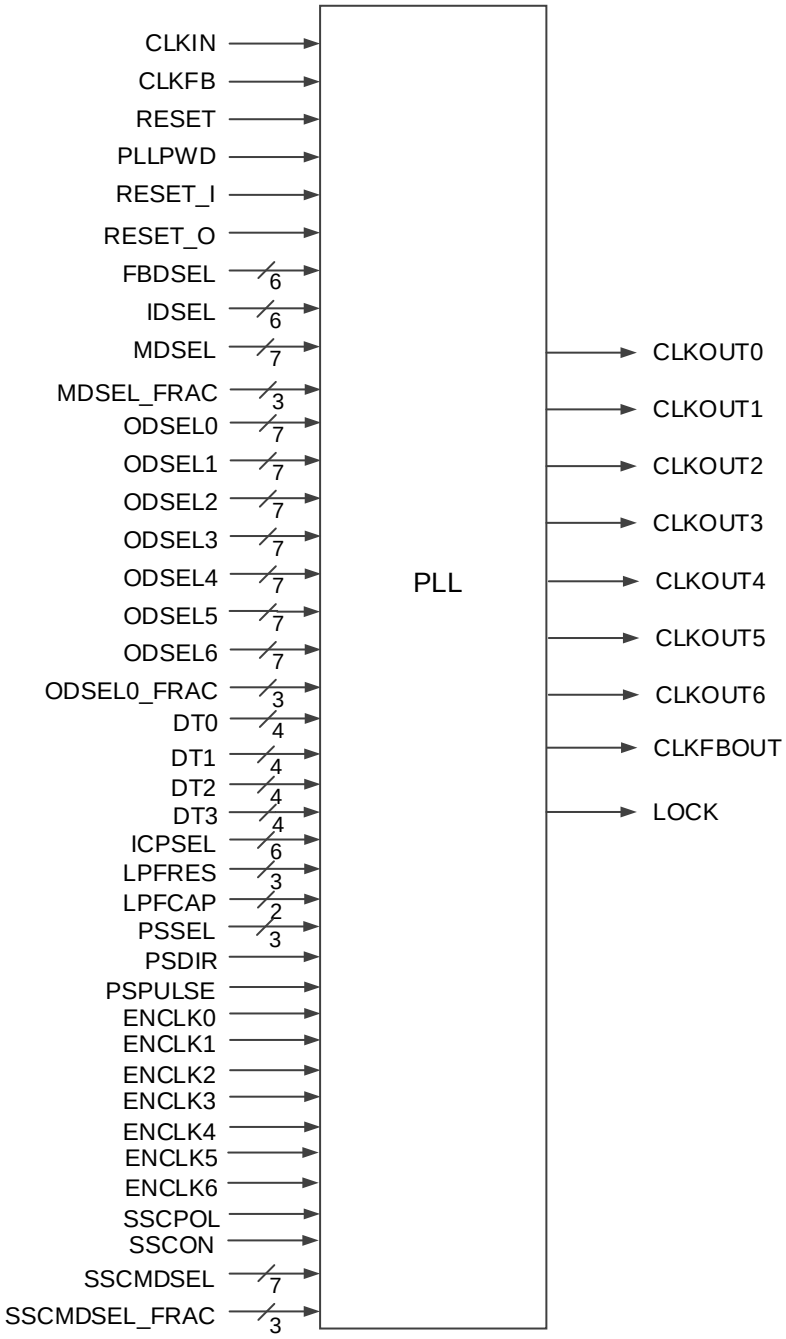
- Bypass in 模式（INMUX 选择来自 CLKIN）：Fclkoutx=Fclkin/ODIVx
- Bypass out 模式（OUTMUX 选择来自 CLKIN）：Fclkoutx=Fclkin
- CAS 模式（仅通道 4）：Fclkout4= Fclkout6[1]/ODIV4

注！

- Fclkin 为参考输入时钟 CLKIN 频率；
- Fclkoutx:x=0~6，为 0~6 通道的输出时钟频率；
- Fclkfb 为反馈输入时钟 CLKFB 频率；
- Fpfd 为 PFD 鉴相频率；
- IDIV、FBDIV、MDIV、ODIVx (x=0~6)为不同分频器的分频系数,即可通过调整不同分频系数得到期望频率的时钟信号；
- [1] Fclkout6 指的是 6 通道 ODIV6 输出的时钟。

端口示意图

图 5-2 PLL 端口示意图



端口介绍

表 5-2 PLL 端口介绍

端口名	I/O	描述
CLKIN	输入	参考时钟输入
CLKFB	输入	反馈时钟输入
RESET	输入	PLL 全部复位信号，复位数字电路，高电平有效。
PLLPWD	输入	PLL power down 信号，对模拟电路 power down，高电平有效。
RESET_I	输入	带 IDIV 的 PLL 全复位，高电平有效，一般用于内部测试使用。
RESET_O	输入	复位 ODIV 和部分数字电路，高电平有效，一般用于内部测试使用。
FBDSEL[5:0]	输入	动态控制 FBDIV 取值，范围 0~63，FBDIV 实际值为 64-FBDSEL。
IDSEL[5:0]	输入	动态控制 IDIV 取值，范围 0~63，IDIV 实际值为 64-IDSEL。
MDSEL[6:0]	输入	动态控制 MDIV 整数取值，范围 0~126,MDIV 实际值为 128-MDSEL，即 MDIV 取值范围 2~128。
MDSEL_FRAC[2:0]	输入	动态控制 MDIV 小数取值，适用 MDIV 整数范围 2~127, 小数取值 1/8。
ODSEL0[6:0]	输入	动态控制 ODIV0 整数取值，范围 0~127，ODIV0 实际值为 128-ODSEL0。
ODSEL0_FRAC[2:0]	输入	动态控制 ODIV0 小数取值，适用 ODIV0 整数范围 2~127,小数取值 1/8。
ODSEL1[6:0]	输入	动态控制 ODIV1 取值，范围 0~127,ODIV1 实际值为 128-ODSEL1。
ODSEL2[6:0]	输入	动态控制 ODIV2 取值，范围 0~127, ODIV2 实际值为 128-ODSEL2
ODSEL3[6:0]	输入	动态控制 ODIV3 取值，范围 0~127, ODIV3 实际值为 128-ODSEL3
ODSEL4[6:0]	输入	动态控制 ODIV4 取值，范围 0~127, ODIV4 实际值为 128-ODSEL4。
ODSEL5[6:0]	输入	动态控制 ODIV5 取值,范围 0~127, ODIV5 实际值为 128-ODSEL5
ODSEL6[6:0]	输入	动态控制 ODIV6 取值，范围 0~127, ODIV6 实际值为 128-ODSEL6。
DT0[3:0]	输入	动态微调控制 CLKOUT0 的占空比
DT1[3:0]	输入	动态微调控制 CLKOUT1 的占空比
DT2[3:0]	输入	动态微调控制 CLKOUT2 的占空比
DT3[3:0]	输入	动态微调控制 CLKOUT3 的占空比
ICPSEL[5:0]	输入	动态控制 ICP 电流大小，电流随着取值的增大而增大,值为 0 时电流最小。

端口名	I/O	描述
LPFRES[2:0]	输入	动态控制 LPFRES 大小，RES 取值范围由小到大为 R0~R7,R0 对应的带宽最大，R7 对应的带宽最小。
LPFCAP[1:0]	输入	动态控制 LPFCAP 大小，CAP 取值范围由小到大为 C0~C2。
PSDIR	输入	动态控制相位移动方向
PSSEL[2:0]	输入	动态控制相位移动通道选择
PSPULSE	输入	动态控制相位移动时钟脉冲
ENCLK0	输入	动态控制通道 0 时钟输出使能，若想使用动态使能则同时需静态参数 CLKOUT0_EN=”TRUE”。
ENCLK1	输入	动态控制通道 1 时钟输出使能，若想使用动态使能则同时需静态参数 CLKOUT1_EN=”TRUE”。
ENCLK2	输入	动态控制通道 2 时钟输出使能，若想使用动态使能则同时需静态参数 CLKOUT2_EN=”TRUE”。
ENCLK3	输入	动态控制通道 3 时钟输出使能，若想使用动态使能则同时需静态参数 CLKOUT3_EN=”TRUE”。
ENCLK4	输入	动态控制通道 4 时钟输出使能，若想使用动态使能则同时需静态参数 CLKOUT4_EN=”TRUE”。
ENCLK5	输入	动态控制通道 5 时钟输出使能,若想使用动态使能则同时需静态参数 CLKOUT5_EN=”TRUE”
ENCLK6	输入	动态控制通道 6 时钟输出使能，若想使用动态使能则同时需静态参数 CLKOUT6_EN=”TRUE”。
SSCPOL	输入	避免 timing 冲突的可选配置： ● 0：CLK Rising Edge ● 1：CLK Falling Edge
SSCON	输入	动态控制 SSC 使能
SSCMDSEL[6:0]	输入	动态控制 SSC MDIV 整数取值,建议 SSC MDIV 实际值范围 16~24（对应 Fpfd 为 50M）、32~48（对应 Fpfd 为 25M），SSC MDIV 实际值为 128-SSCMDSEL。
SSCMDSEL_FRAC[2:0]	输入	动态控制 SSC MDIV 小数取值，小数取值 1/8。
CLKOUT0	输出	0 通道时钟输出（默认）
CLKOUT1	输出	1 通道时钟输出
CLKOUT2	输出	2 通道时钟输出
CLKOUT3	输出	3 通道时钟输出
CLKOUT4	输出	4 通道时钟输出
CLKOUT5	输出	5 通道时钟输出
CLKOUT6	输出	6 通道时钟输出
CLKFBOUT	输出	反馈时钟输出
LOCK	输出	PLL 锁定指示： ● 1：锁定 ● 0：失锁

参数介绍

表 5-3 PLL 参数介绍

参数名	类型	取值范围	默认值	描述
FCLKIN	string	"10"~"400"	"100.0"	参考时钟频率(MHz)
IDIV_SEL	integer	1~64	1	IDIV 分频系数静态设置，对应实际取值为 1~64。
DYN_IDIV_SEL	string	"TRUE", "FALSE"	"FALSE"	IDIV 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 IDIV_SEL； ● TRUE: 动态，即选择信号 IDSEL。
FBDIV_SEL	integer	1~64	1	FBDIV 分频系数静态设置，对应实际取值为 1~64。
DYN_FBDIV_SEL	string	"TRUE", "FALSE"	"FALSE"	FBDIV 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 FBDIV_SEL； ● TRUE: 动态，即选择信号 FBDSEL。
ODIV0_SEL	integer	1~128	8	ODIV0 分频系数整数静态设置
ODIV0_FRAC_SEL	integer	0~7	0	ODIV0 分频系数小数静态设置
DYN_ODIV0_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV0 分频系数静态控制参数或动态控制信号选择 FALSE: 静态，即选择参数 ODIV0SEL 和 ODIV0FRAC_SEL； TRUE: 动态，即选择信号 ODSEL0 和 ODSEL0_FRAC。
ODIV1_SEL	integer	1~128	8	ODIV1 分频系数静态设置
DYN_ODIV1_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV1 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ODIV1_SEL； ● TRUE: 动态，即选择信号 ODSEL1。
ODIV2_SEL	integer	1~128	8	ODIV2 分频系数静态设置
DYN_ODIV2_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV2 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ODIV2_SEL； ● TRUE: 动态，即选择信号 ODSEL2。
ODIV3_SEL	integer	1~128	8	ODIV3 分频系数静态设置

参数名	类型	取值范围	默认值	描述
DYN_ODIV3_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV3 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ODIV3_SEL； ● TRUE: 动态，即选择信号 ODSEL3。
ODIV4_SEL	integer	1~128	8	ODIV4 分频系数静态设置
DYN_ODIV4_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV4 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ODIV4_SEL； ● TRUE: 动态，即选择信号 ODSEL4。
ODIV5_SEL	integer	1~128	8	ODIV5 分频系数静态设置
DYN_ODIV5_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV5 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ODIV5_SEL； ● TRUE: 动态，即选择信号 ODSEL5。
ODIV6_SEL	integer	1~128	8	ODIV6 分频系数静态设置
DYN_ODIV6_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV6 分频系数静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ODIV6_SEL； ● TRUE: 动态，即选择信号 ODSEL6。
DYN_MDIV_SEL	string	"TRUE", "FALSE"	“FALSE"	MDIV 分频系数静态控制参数 或动态控制信号选择 ● "FALSE"：静态，即选择参数 MDIV_SEL 和 MDSEL_FRAC； ● "TRUE"：动态，即选择信号 ODSEL6。
MDIV_SEL	integer	2~128	8	MDIV 分频系数整数静态设置
MDIV_FRAC_SEL	string	0~7	0	MDIV 分频系数小数静态设置
CLKOUT0_EN	string	"TRUE", "FALSE"	"TRUE"	0 通道时钟输出使能
CLKOUT1_EN	string	"TRUE", "FALSE"	"FALSE"	1 通道时钟输出使能
CLKOUT2_EN	string	"TRUE", "FALSE"	"FALSE"	2 通道时钟输出使能
CLKOUT3_EN	string	"TRUE", "FALSE"	"FALSE"	3 通道时钟输出使能

参数名	类型	取值范围	默认值	描述
CLKOUT4_EN	string	"TRUE", "FALSE"	"FALSE"	4 通道时钟输出使能
CLKOUT5_EN	string	"TRUE", "FALSE"	"FALSE"	5 通道时钟输出使能
CLKOUT6_EN	string	"TRUE", "FALSE"	"FALSE"	6 通道时钟输出使能
DYN_DT0_SEL	string	"TRUE", "FALSE"	"FALSE"	0 通道占空比微调静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 CLKOUT0_DT_DIR & CLKOUT0_DT_STEP； ● TRUE: 动态，即选择信号 DT0。
DYN_DT1_SEL	string	"TRUE", "FALSE"	"FALSE"	1 通道占空比微调静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 CLKOUT1_DT_DIR & CLKOUT1_DT_STEP； ● TRUE: 动态，即选择信号 DT1。
DYN_DT2_SEL	string	"TRUE", "FALSE"	"FALSE"	2 通道占空比微调静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 CLKOUT2_DT_DIR & CLKOUT2_DT_STEP； ● TRUE: 动态，即选择信号 DT2。
DYN_DT3_SEL	string	"TRUE", "FALSE"	"FALSE"	3 通道占空比微调静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 CLKOUT3_DT_DIR & CLKOUT3_DT_STEP； ● TRUE: 动态，即选择信号 DT3。
CLKOUT0_DT_DIR	binary	1'b1, 1'b0	1'b1	0 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿 ● 1'b0: - 占空比减少，以下降沿对齐为基准， 调整上升沿。
CLKOUT1_DT_DIR	binary	1'b1, 1'b0	1'b1	1 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿； ● 1'b0: - 占空比减少，以下降沿对齐为基准，调整上升沿。
CLKOUT2_DT_DIR	binary	1'b1, 1'b0	1'b1	2 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿； ● 1'b0: - 占空比减少，以下降沿对齐为基准，调整上升沿。

参数名	类型	取值范围	默认值	描述
CLKOUT3_DT_DIR	binary	1'b1, 1'b0	1'b1	3 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿； ● 1'b0: - 占空比减少，以下降沿对齐为基准，调整上升沿。
CLKOUT0_DT_STEP	integer	0,1,2,4	0	0 通道占空比静态微调步长，每步 50ps。
CLKOUT1_DT_STEP	integer	0,1,2,4	0	1 通道占空比静态微调步长，每步 50ps。
CLKOUT2_DT_STEP	integer	0,1,2,4	0	2 通道占空比静态微调步长，每步 50ps。
CLKOUT3_DT_STEP	integer	0,1,2,4	0	3 通道占空比静态微调步长，每步 50ps。
CLK0_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV0 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK0_OUT_SEL	binary	1'b0, 1'b1	1'b0	0 通道输出时钟来源选择 1'b0: 来自 ODIV0 输出 1'b1: 输出时钟旁路来自 CLKIN
CLK1_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV1 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK1_OUT_SEL	binary	1'b0, 1'b1	1'b0	1 通道输出时钟来源选择 ● 1'b0：来自 ODIV1 输出 ● 1'b1：输出时钟旁路来自 CLKIN
CLK2_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV2 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 2'b1: 输出时钟旁路来自 CLKIN
CLK2_OUT_SEL	binary	1'b0, 1'b1	1'b0	2 通道输出时钟来源选择 1'b0: 来自 ODIV2 输出 1'b1: 输出时钟旁路来自 CLKIN
CLK3_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV3 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK3_OUT_SEL	binary	1'b0, 1'b1	1'b0	3 通道输出时钟来源选择 ● 1'b0: 来自 ODIV3 的输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK4_IN_SEL	binary	2'b00, 2'b01,2'b10	2'b00	ODIV4 输入时钟来源选择 ● 2'b00: 来自 VCO 输出 ● 2'b01: 级联来自 CLKCAS_6 ● 2'b10: 输出时钟旁路来自 CLKIN

参数名	类型	取值范围	默认值	描述
CLK4_OUT_SEL	binary	1'b0, 1'b1	1'b0	4 通道输出时钟来源选择 ● 1'b0: 来自 ODIV4 的输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK5_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV5 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK5_OUT_SEL	binary	1'b0, 1'b1	1'b0	5 通道输出时钟来源选择 ● 1'b0: 来自 ODIV5 的输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLKFB_SEL	string	"INTERNAL", "EXTERNAL"	"INTERNAL"	CLKFB 来源选择 ● INTERNAL：来自内部 CLKOUT 反馈 ● EXTERNAL：来自外部信号反馈
DYN_DPA_EN	string	"TRUE", "FALSE"	"FALSE"	动态相移调整使能
CLKOUT0_PE_COARSE	integer	0~127	0	0 通道相移粗调静态设置
CLKOUT0_PE_FINE	integer	0~7	0	0 通道相移微调静态设置
CLKOUT1_PE_COARSE	integer	0~127	0	1 通道相移粗调静态设置
CLKOUT1_PE_FINE	integer	0~7	0	1 通道相移微调静态设置
CLKOUT2_PE_COARSE	integer	0~127	0	2 通道相移粗调静态设置
CLKOUT2_PE_FINE	integer	0~7	0	2 通道相移微调静态设置
CLKOUT3_PE_COARSE	integer	0~127	0	3 通道相移粗调静态设置
CLKOUT3_PE_FINE	integer	0~7	0	3 通道相移微调静态设置
CLKOUT4_PE_COARSE	integer	0~127	0	4 通道相移粗调静态设置
CLKOUT4_PE_FINE	integer	0~7	0	4 通道相移微调静态设置
CLKOUT5_PE_COARSE	integer	0~127	0	5 通道相移粗调静态设置
CLKOUT5_PE_FINE	integer	0~7	0	5 通道相移微调静态设置
CLKOUT6_PE_COARSE	integer	0~127	0	6 通道相移粗调静态设置
CLKOUT6_PE_FINE	integer	0~7	0	6 通道相移微调静态设置
DYN_PE0_SEL	string	“TRUE","FALSE"	“FALSE"	0 通道相位调整静态控制参数或动态控制信号选择

参数名	类型	取值范围	默认值	描述
				● "FALSE"：静态，即选择参数 CLKOUT0_PE_COARSE 和 CLKOUT0_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE1_SEL	string	“TRUE","FALSE"	“FALSE"	1 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT1_PE_COARSE 和 CLKOUT1_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE2_SEL	string	“TRUE","FALSE"	“FALSE"	2 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT2_PE_COARSE 和 CLKOUT2_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE3_SEL	string	“TRUE","FALSE"	“FALSE"	3 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT3_PE_COARSE 和 CLKOUT3_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE4_SEL	string	“TRUE","FALSE"	“FALSE"	4 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT4_PE_COARSE 和 CLKOUT4_PE_FINE ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE5_SEL	string	“TRUE","FALSE"	“FALSE"	5 通道相位调整静态控制参数或动态控制信号选择

参数名	类型	取值范围	默认值	描述
				● "FALSE"：静态，即选择参数 CLKOUT5_PE_COARSE 和 CLKOUT5_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE6_SEL	string	“TRUE","FALSE"	“FALSE"	6 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT6_PE_COARSE 和 CLKOUT6_PE_FINE ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现同时需，DYN_DPA_EN="TRUE"。
DE0_EN	string	“TRUE","FALSE"	“FALSE"	0 通道(ODIV0=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE0_SEL="TRUE" 时设置 CLKOUT0_PE_COARSE 和 CLKOUT0_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE1_EN	string	“TRUE","FALSE"	“FALSE"	1 通道(ODIV1=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE1_SEL="TRUE" 时设置 CLKOUT1_PE_COARSE 和 CLKOUT1_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE2_EN	string	“TRUE","FALSE"	“FALSE"	2 通道(ODIV2=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE2_SEL="TRUE" 时设置 CLKOUT2_PE_COARSE 和 CLKOUT2_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE3_EN	string	“TRUE","FALSE"	“FALSE"	3 通道(ODIV3=2~128)占空比调整使能 ● "FALSE"：50%占空比

参数名	类型	取值范围	默认值	描述
				● "TRUE"：DYN_PE3_SEL="TRUE" 时设置 CLKOUT3_PE_COARSE 和 CLKOUT3_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE4_EN	string	“TRUE","FALSE"	“FALSE"	4 通道(ODIV4=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE4_SEL="TRUE" 时设置 CLKOUT4_PE_COARSE 和 CLKOUT4_PE_FIN 作为 falling edge，结合动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE5_EN	string	“TRUE","FALSE"	“FALSE"	5 通道(ODIV5=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE5_SEL="TRUE" 时设置 CLKOUT5_PE_COARSE 和 CLKOUT5_PE_FIN 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE6_EN	string	“TRUE","FALSE"	“FALSE"	6 通道(ODIV6=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE6_SEL="TRUE" 时设置 CLKOUT6_PE_COARSE 和 CLKOUT6_PE_FIN 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
RESET_I_EN	string	"TRUE", "FALSE"	"FALSE"	使能动态信号 RESET_I，若需要使用 RESET_I 端口，需将该参数设为 TRUE。
RESET_O_EN	string	"TRUE", "FALSE"	"FALSE"	使能动态信号 RESET_O，若需要使用 RESET_O 端口，需将该参数设为 TRUE。
DYN_ICP_SEL	string	"TRUE", "FALSE"	"FALSE"	ICPSEL 静态控制参数或动态控制信号选择 ● FALSE: 静态，即选择参数 ICP_SEL； ● TRUE: 动态，即选择动态信号 ICPSEL。
ICP_SEL	binary	6'bXXXXXX, 6'b000000~6'b111111	6'bXXXXXX	ICP 电流静态设置 ● 6'bXXXXXX：表示软件会自动计算并设置该参数

参数名	类型	取值范围	默认值	描述
				● 6'b000000~6'b111111：用户若需自行设置，可根据需要在参数范围内设置
DYN_LPF_SEL	string	"TRUE", "FALSE"	"FALSE"	LPFRES 和 LPFCAP 静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 LPF_RES 和 LPF_CAP； ● "TRUE"：动态，即选择动态信号 LPFREST 和 LPFCAP。
LPF_RES	binary	3'bXXX,3'b000~3'b111	3'bXXX	LPRRES 静态设置 ● 3'bXXX：表示软件会自动计算并设置该参数 ● 3'b000~3'b111：用户若需设置密级程度可根据需要在参数范围内设置
LPF_CAP	binary	2'b00~2'b10	2'b00	LFPCAP 静态设置
SSC_EN	string	"TRUE", "FALSE"	"FALSE"	SSC 使能

输入时钟分频器（IDIV），用于控制进入 PLL 模块的输入时钟频率。该分频系数可以通过端口 IDSEL 动态调整，也可以通过参数 IDIV_SEL 静态调整，对应关系如表 5-4 所示。

IDIV 值与端口 IDSEL 之间的关系为 IDSEL=dec2bin(64-IDIV)。

表 5-4 IDIV 对应关系

IDSEL[5:0]（动态）	IDIV_SEL(静态)	IDIV 实际值
111111	1	1
111110	2	2
111101	3	3
111100	4	4
111011	5	5
111010	6	6
111001	7	7
111000	8	8
110111	9	9
……	……	……
000000	64	64

FBDIV 分频器用于分频反馈信号。该分频系数可以通过端口 FBDSEL 动态调整，也可以通过参数 FBDIV_SEL 静态调整，对应关系如表 5-5 所示。

FBDIV 值与端口 FBDSEL 之间的关系 FBDSEL=dec2bin(64-FBDIV)

表 5-5 FBDIV 对应关系

FBDSEL [5:0]（动态）	FBDIV_SEL（静态）	FBDIV 实际值
111111	1	1
111110	2	2
111101	3	3
111100	4	4
111011	5	5
111010	6	6
111001	7	7
111000	8	8
110111	9	9
……	……	……
000000	64	64

ODIV 输出分频器，0 通道支持整数分频和小数分频；1~6 通道仅支持整数分频。对于 0 通道，该分频系数可以通过 ODSEL0 和 ODSEL0_FRAC 动态调整，也可通过参数 ODIV0_SEL 和 ODIV0_FRAC_SEL 静态调整，整数分频系数对应关系如表 5-6 所示，小数分频系数对应关系如表 5-7 所示。

ODIV0 整数值与端口 ODSEL 之间的关系为 ODSEL0=dec2bin(128-ODIV0 整数值)。

表 5-6 ODIV0 整数分频对照表

ODSEL0 [6:0]（动态）	ODIV0_SEL（静态）	ODIV0 整数值
1111111	1	1
1111110	2	2
1111101	3	3
1111100	4	4
1111011	5	5
1111010	6	6
1111001	7	7
1111000	8	8
1110111	9	9
……	……	……
0000000	128	128

ODIV0 小数值与端口 ODSEL0_FRAC 之间的关系为 ODSEL0_FRAC=dec2bin(7-ODIV0 小数值/0.125)，且在整数分频为[2~127]时小数分频才有效。对于 1~6 通道，ODIV 分频系数可以通过 ODSELx(x=1~6)动态调整，也可以通过参数 ODIVx_SEL(x=1~6)静态调

整，对应关系可参考 0 通道的整数分频，即表 5-6。

表 5-7 ODIV0 小数分频对照表

ODSEL_FRAC[2:0]（动态）	ODIV0_FRAC_SEL（静态）	ODIV0 小数值
111	0	0*0.125=0
110	1	1*0.125=0.125
101	2	2*0.125=0.25
100	3	3*0.125=0.375
011	4	4*0.125=0.5
010	5	5*0.125=0.625
001	6	6*0.125=0.75
000	7	7*0.125=0.875

MDIV、CLKFB 分频器作用与 FBDIV 类似。该分频系数支持整数和小数分频，通过端口 MDSEL 和 MDSEL_FRAC 动态调整,也可以通过参数 MDIV_SEL 和 MDIV_FRAC_SEL 静态调整，整数分频系数对应关系如表 5-8 所示，小数分频系数对应关系如表 5-9 所示。

表 5-8 MDIV 整数分频对应关系

MDSEL [6:0]（动态）	MDIV_SEL（静态）	MDIV 整数值
1111110	2	2
1111101	3	3
1111100	4	4
1111011	5	5
1111010	6	6
1111001	7	7
1111000	8	8
1110111	9	9
……	……	……
0000000	128	128

MDIV 整数值与端口 MDSEL 之间的关系为 MDSEL=dec2bin(128-MDIV 整数值)，且 MDIV 取值范围为[2~128]。

表 5-9 MDIV 小数分频对应关系

MDSEL_FRAC[2:0]（动态）	MDIV_FRAC_SEL(静态)	MDIV 小数值
111	0	0*0.125=0
110	1	1*0.125=0.125
101	2	2*0.125=0.25
100	3	3*0.125=0.375
011	4	4*0.125=0.5

MDSEL_FRAC[2:0]（动态）	MDIV_FRAC_SEL(静态)	MDIV 小数值
010	5	5*0.125=0.625
001	6	6*0.125=0.75
000	7	7*0.125=0.875

MDIV 小数值与端口 MDSEL_FRAC 之间的关系 MDSEL_FRAC=dec2bin(7-MDIV 小数值/0.125) ，且在整数分频为[2~127]时小数分频才有效。

相位调整

PLL 相位调整支持静态和动态两种调整方式，且 0~6 通道均支持相位调整。以 MDIV 通道为基准，相位调整是指 ODIV0~6 相对于 MDIV 的相移。

静态相位调整通过设置参数 CLKOUTx_PE_COARSE 和 CLKOUTx_PE_FINE(x=0~6)来实现。

动态相位调整通过信号 PSSEL、PSDIR、PSPULSE 来实现。PSSEL 用于控制选择通道，PSDIR 用于控制加或减操作，一个 PSPULSE 脉冲下降沿 DYN_FINE 加/减 1，DYN_FINE 上溢或下溢时 DYN_COARSE 加 1 或减 1 操作，其中 DYN_COARSE 的值小于 ODIV。

相位调整计算公式如下:

PS=(COARSE+FINE/8)/ODIV*360，PS 范围[0,360)

0 通道 ODIV=ODIV0 整数值+ODIV0 小数值，1~6 通道 ODIV 仅为整数值。

注!

- DYN_FINE 和 DYN_COARSE 是由 DPA 产生的内部信号，通过 PSSEL、PSDIR、PSPULSE 配合产生，其作用和 CLKOUTx_PE_COARSE 和 CLKOUTx_PE_FINE 一致；
- 公式中 COARSE 和 FINE 指的是选择动态或静态调整之后真正作用于相位调整的值；
- 相位调整电路是针对 VCO 设计的，对 Bypass in 或 CAS 模式，相位调整公式虽仍适用,但需设置 FINE 为 0。

相位调整应用

PLL 所有的 CLKOUT 均支持 相位调整应用，为保证 IO 多路高速时钟之间的相位关系，强烈建议使用 CLKOUT [0:3]（硬件直连 HCLK）。

占空比调整

PLL 占空比调整只支持动态调整，且 0~6 通道均支持。占空比定义如下:

Duty cycle= (falling edge – rising edge)/cycle_period

其中 falling edge 的位置由静态相移设置决定，定义为 DUTY，rising edge 的位置由动态相移设置的 PHASE 决定，DYN_FINE 和

DYN_COARSE 是由 DPA 产生的内部信号。DUTY 和 PHASE 的计算公式如下(以 1 通道为例):

DUTY=(CLKOUT1_PE_COARSE+CLKOUT1_PE_FINE/8)

PHASE=(DYN_COARSE1+DYN_FINE1/8)

动态占空比计算:

- 若 DUTY>PHASE,Duty cycle=(DUTY-PHASE)/ODIV1
- 若 DUTY<PHASE,Duty cycle=(DUTY-PHASE)/ODIV1+1

注!

- ODIV=1 时不支持动态占空比调整，占空比固定为 50%；
- ODIV>=2 时，DUTY-PHASE 不支持(-0.5,0.5)之间的值；
- 占空比调整电路是针对 VCO 设计的，对 Bypass in 或 CAS 模式，不允许占空比调整，且此两种模式下，如果 ODIV(>2)为奇数时，占空比不是 50%(高电平<低电平，即占空比小于 50%)。

占空比微调

PLL 占空比微调支持静态和动态两种调整方式，且只有 0~3 通道支持占空比微调，通过设置占空比微调方向和步长来实现。微调方向为 1'b1 时，调节下降沿延时，占空比增加；微调方向为 1'b0 时，调节上升沿延时，占空比减小。具体延时值如表 5-10 所示(以 1 通道为例):

表 5-10 PLL 占空比微调对照表

动态调整	静态调整		占空比微调延时值
DT1[3:0]	CLKOUT1_DT_DIR	CLKOUT1_DT_STEP	
0111	1'b0	0	0
0110		1	-50ps
0101		2	-100ps
0011		4	-200ps
1111	1'b1	0	0
1110		1	+50ps
1101		2	+100ps
1011		4	+200ps

假设 0、1 通道输出相同频率时钟，对 1 通道时钟进行占空比微调，以 0 通道时钟为参考，具体时序如图 5-3 和图 5-4 所示。

图 5-3 通道 1 占空比微调时序图(微调方向为 1'b1，步数 1)

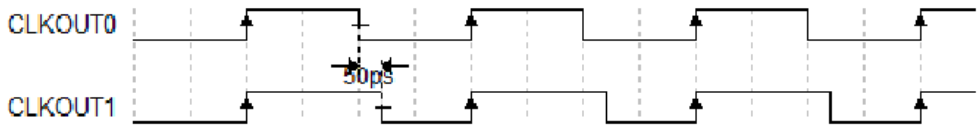
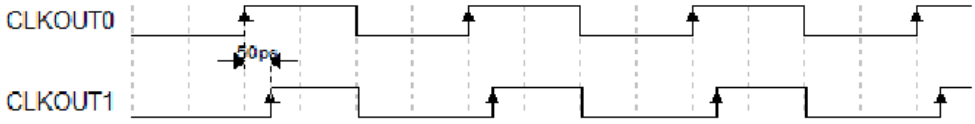


图 5-4 通道 1 占空比微调时序图(微调方向为 1'b0，步数为 1)



PhaseShift 应用

PLL 所有的 CLKOUT 均支持 PhaseShift 应用，为保证 IO 多路高速时钟之间的相位关系，强烈建议使用 CLKOUT [0:3]（硬件直连 HCLK）。

原语例化

可以直接实例化原语。

Verilog 例化：

```
PLL uut (
    .LOCK(lock),
    .CLKOUT0(clkout0),
    .CLKOUT1(clkout1),
    .CLKOUT2(clkout2),
    .CLKOUT3(clkout3),
    .CLKOUT4(clkout4),
    .CLKOUT5(clkout5),
    .CLKOUT6(clkout6),
    .CLKFBOUT(clkfbout),
    .CLKIN(clkin),
    .CLKFB(clkfb),
    .RESET(reset),
    .PLLPWD(pllpwd),
    .RESET_I(reseti),
    .RESET_O(reseto),
    .FBDSEL(fbdsel),
    .IDSEL(idsel),
    .MDSEL(mdsel),
    .MDSEL_FRAC(mdsel_frac),
    .ODSEL0(odesl0),
    .ODSEL0_FRAC(odesl0_frac),
    .ODSEL1(odesl1),
    .ODSEL2(odesl2),
    .ODSEL3(odesl3),
    .ODSEL4(odesl4),
    .ODSEL5(odesl5),
    .ODSEL6(odesl6),
    .DT0(dt0),
    .DT1(dt1),
```

```
        .DT2(dt2),
        .DT3(dt3),
        .ICPSEL(icpsel),
        .LPFRES(lpfres),
        .LPFCAP(lpfcap),
        .PSSEL(pssel),
        .PSDIR(psdir),
        .PSPULSE(pspulse),
        .ENCLK0(enclk0),
        .ENCLK1(enclk1),
        .ENCLK2(enclk2),
        .ENCLK3(enclk3),
        .ENCLK4(enclk4),
        .ENCLK5(enclk5),
        .ENCLK6(enclk6),
        .SSCPOL(sscpol),
        .SSCON(sscon),
        .SSCMDSEL(sscmdsel),
        .SSCMDSEL_FRAC(sscmdsel_frac)
    );
    defparam uut.CLK0_IN_SEL=1'b0;
    defparam uut.CLK0_OUT_SEL=1'b0;
    defparam uut.CLK1_IN_SEL=1'b0;
    defparam uut.CLK1_OUT_SEL=1'b0;
    defparam uut.CLK2_IN_SEL=1'b0;
    defparam uut.CLK2_OUT_SEL=1'b0;
    defparam uut.CLK3_IN_SEL=1'b0;
    defparam uut.CLK3_OUT_SEL=1'b0;
    defparam uut.CLK4_IN_SEL=2'b00;
    defparam uut.CLK4_OUT_SEL=1'b0;
    defparam uut.CLK5_IN_SEL=1'b0;
    defparam uut.CLK5_OUT_SEL=1'b0;
    defparam uut.CLK6_IN_SEL=1'b0;
    defparam uut.CLK6_OUT_SEL=1'b0;
    defparam uut.CLKFB_SEL="INTERNAL";
    defparam uut.CLKOUT0_DT_DIR=1'b1;
    defparam uut.CLKOUT0_DT_STEP=0;
    defparam uut.CLKOUT0_EN="TRUE";
    defparam uut.CLKOUT0_PE_COARSE=0;
    defparam uut.CLKOUT0_PE_FINE=0;
    defparam uut.CLKOUT1_DT_DIR=1'b1;
    defparam uut.CLKOUT1_DT_STEP=0;
```

```
defparam uut.CLKOUT1_EN=" TRUE ";
defparam uut.CLKOUT1_PE_COARSE=0;
defparam uut.CLKOUT1_PE_FINE=0;
defparam uut.CLKOUT2_DT_DIR=1'b1;
defparam uut.CLKOUT2_DT_STEP=0;
defparam uut.CLKOUT2_EN="FALSE";
defparam uut.CLKOUT2_PE_COARSE=0;
defparam uut.CLKOUT2_PE_FINE=0;
defparam uut.CLKOUT3_DT_DIR=1'b1;
defparam uut.CLKOUT3_DT_STEP=0;
defparam uut.CLKOUT3_EN=" TRUE ";
defparam uut.CLKOUT3_PE_COARSE=0;
defparam uut.CLKOUT3_PE_FINE=0;
defparam uut.CLKOUT4_EN=" TRUE ";
defparam uut.CLKOUT4_PE_COARSE=0;
defparam uut.CLKOUT4_PE_FINE=0;
defparam uut.CLKOUT5_EN=" TRUE ";
defparam uut.CLKOUT5_PE_COARSE=0;
defparam uut.CLKOUT5_PE_FINE=0;
defparam uut.CLKOUT6_EN=" TRUE ";
defparam uut.CLKOUT6_PE_COARSE=0;
defparam uut.CLKOUT6_PE_FINE=0;
defparam uut.DE0_EN="FALSE";
defparam uut.DE1_EN="FALSE";
defparam uut.DE2_EN="FALSE";
defparam uut.DE3_EN="FALSE";
defparam uut.DE4_EN="FALSE";
defparam uut.DE5_EN="FALSE";
defparam uut.DE6_EN="FALSE";
defparam uut.DYN_DPA_EN="FALSE";
defparam uut.DYN_DT0_SEL="FALSE";
defparam uut.DYN_DT1_SEL="FALSE";
defparam uut.DYN_DT2_SEL="FALSE";
defparam uut.DYN_DT3_SEL="FALSE";
defparam uut.DYN_FBDIV_SEL="FALSE";
defparam uut.DYN_ICP_SEL="FALSE";
defparam uut.DYN_IDIV_SEL="FALSE";
defparam uut.DYN_LPF_SEL="FALSE";
defparam uut.DYN_MDIV_SEL="FALSE";
defparam uut.DYN_ODIV0_SEL="FALSE";
defparam uut.DYN_ODIV1_SEL="FALSE";
defparam uut.DYN_ODIV2_SEL="FALSE";
```

```
defparam uut.DYN_ODIV3_SEL="FALSE";
defparam uut.DYN_ODIV4_SEL="FALSE";
defparam uut.DYN_ODIV5_SEL="FALSE";
defparam uut.DYN_ODIV6_SEL="FALSE";
defparam uut.DYN_PE0_SEL="FALSE";
defparam uut.DYN_PE1_SEL="FALSE";
defparam uut.DYN_PE2_SEL="FALSE";
defparam uut.DYN_PE3_SEL="FALSE";
defparam uut.DYN_PE4_SEL="FALSE";
defparam uut.DYN_PE5_SEL="FALSE";
defparam uut.DYN_PE6_SEL="FALSE";
defparam uut.FBDIV_SEL=1;
defparam uut.FCLKIN="100.0";
defparam uut.ICP_SEL=6'bXXXXXX;
defparam uut.IDIV_SEL=1;
defparam uut.LPF_CAP=2'b00;
defparam uut.LPF_RES=3'b000;
defparam uut.MDIV_FRAC_SEL=0;
defparam uut.MDIV_SEL=8;
defparam uut.ODIV0_FRAC_SEL=0;
defparam uut.ODIV0_SEL=8;
defparam uut.ODIV1_SEL=8;
defparam uut.ODIV2_SEL=8;
defparam uut.ODIV3_SEL=8;
defparam uut.ODIV4_SEL=8;
defparam uut.ODIV5_SEL=8;
defparam uut.ODIV6_SEL=8;
defparam uut.RESET_I_EN="FALSE";
defparam uut.RESET_O_EN="FALSE";
defparam uut.SSC_EN="FALSE";
```

VHDL 例化：

```
COMPONENT PLL
    GENERIC(
        FCLKIN : STRING := "100.0";
        DYN_IDIV_SEL : STRING := "FALSE";
        IDIV_SEL : integer := 1;
        DYN_FBDIV_SEL : STRING := "FALSE";
        FBDIV_SEL : integer := 1;
        DYN_ODIV0_SEL : STRING := "FALSE";
        ODIV00_SEL : integer := 8;
```

```
DYN_ODIV1_SEL : STRING := "FALSE";
ODIV1_SEL : integer := 8;
DYN_ODIV2_SEL : STRING := "FALSE";
ODIV2_SEL : integer := 8;
DYN_ODIV3_SEL : STRING := "FALSE";
ODIV3_SEL : integer := 8;
DYN_ODIV4_SEL : STRING := "FALSE";
ODIV4_SEL : integer := 8;
DYN_ODIV5_SEL : STRING := "FALSE";
ODIV5_SEL : integer := 8;
DYN_ODIV6_SEL : STRING := "FALSE";
ODIV6_SEL : integer := 8;
DYN_MDIV_SEL : STRING := "FALSE";
MDIV_SEL : integer := 8;
MDIV_FRAC_SEL : integer := 0;
ODIV0_FRAC_SEL : integer := 0;
CLKOUT0_EN : STRING := "TRUE";
CLKOUT1_EN : STRING := " FALSE ";
CLKOUT2_EN : STRING := " FALSE ";
CLKOUT3_EN : STRING := " FALSE ";
CLKOUT4_EN : STRING := " FALSE ";
CLKOUT5_EN : STRING := " FALSE ";
CLKOUT6_EN : STRING := " FALSE ";
DYN_DT0_SEL : STRING := "FALSE";
DYN_DT1_SEL : STRING := "FALSE";
DYN_DT2_SEL : STRING := "FALSE";
DYN_DT3_SEL : STRING := "FALSE";
CLKOUT0_DT_DIR : bit := '1';
CLKOUT1_DT_DIR : bit := '1';
CLKOUT2_DT_DIR : bit := '1';
CLKOUT3_DT_DIR : bit := '1';
CLKOUT0_DT_STEP : integer := 0;
CLKOUT1_DT_STEP : integer := 0;
CLKOUT2_DT_STEP : integer := 0;
CLKOUT3_DT_STEP : integer := 0;
```

```
CLK0_IN_SEL    : bit := '0';
CLK0_OUT_SEL : bit := '0';
CLK1_IN_SEL    : bit_vector := '0';
CLK1_OUT_SEL : bit := '0';
CLK2_IN_SEL    : bit_vector := '0';
CLK2_OUT_SEL : bit := '0';
CLK3_IN_SEL    : bit_vector := '0';
CLK3_OUT_SEL : bit := '0';
CLK4_IN_SEL    : bit_vector := "00";
CLK4_OUT_SEL : bit := '0';
CLK5_IN_SEL    : bit_vector := '0';
CLK5_OUT_SEL : bit := '0';
CLK6_IN_SEL    : bit_vector := '0';
CLK6_OUT_SEL : bit := '0';
CLKFB_SEL : STRING := "INTERNAL";
DYN_DPA_EN : STRING := "FALSE";
DYN_PE0_SEL : STRING := "FALSE";
DYN_PE1_SEL : STRING := "FALSE";
DYN_PE2_SEL : STRING := "FALSE";
DYN_PE3_SEL : STRING := "FALSE";
DYN_PE4_SEL : STRING := "FALSE";
DYN_PE5_SEL : STRING := "FALSE";
DYN_PE6_SEL : STRING := "FALSE";
CLKOUT0_PE_COARSE : integer := 0;
CLKOUT0_PE_FINE : integer := 0;
CLKOUT1_PE_COARSE : integer := 0;
CLKOUT1_PE_FINE : integer := 0;
CLKOUT2_PE_COARSE : integer := 0;
CLKOUT2_PE_FINE : integer := 0;
CLKOUT3_PE_COARSE : integer := 0;
CLKOUT3_PE_FINE : integer := 0;
CLKOUT4_PE_COARSE : integer := 0;
CLKOUT4_PE_FINE : integer := 0;
CLKOUT5_PE_COARSE : integer := 0;
CLKOUT5_PE_FINE : integer := 0;
```

```
        CLKOUT6_PE_COARSE : integer := 0;
        CLKOUT6_PE_FINE : integer := 0;
        DE0_EN : STRING := "FALSE";
        DE1_EN : STRING := "FALSE";
        DE2_EN : STRING := "FALSE";
        DE3_EN : STRING := "FALSE";
        DE4_EN : STRING := "FALSE";
        DE5_EN : STRING := "FALSE";
        DE6_EN : STRING := "FALSE";
        RESET_I_EN : STRING := "FALSE";
        RESET_O_EN : STRING := "FALSE";
        DYN_ICP_SEL : STRING := "FALSE";
        ICP_SEL : std_logic_vector(5 downto 0) := "XXXXXX";
        DYN_LPF_SEL : STRING := "FALSE";
        LPR_RES : std_logic_vector(2 downto 0) := "XXX";
        LPR_CAP : bit_vector := "00";
        SSC_EN : STRING := "FALSE"
    );
    PORT(
        CLKIN : IN std_logic;
        CLKFB : IN std_logic:='0';
        RESET, PLLPWD : IN std_logic:='0';
        RESET_I, RESET_O : IN std_logic:='0';
        IDSEL, FBDSEL : IN std_logic_vector(5 downto 0);
        ODSEL0,ODSEL1,ODSEL2,ODSEL3, DSEL4,ODSEL5,
    ODSEL6, MDSEL : IN std_logic_vector(6 downto 0);
        MDSEL_FRAC, ODSEL0_FRAC: IN std_logic_vector(2
downto 0);

        DT0,DT1,DT2,DT3 : IN std_logic_vector(3 downto 0);
        ICPSEL : IN std_logic_vector(5 downto 0);
        LPFRES : IN std_logic_vector(2 downto 0);
        LPFCAP : IN std_logic_vector(1 downto 0);
        PSSEL : IN std_logic_vector(2 downto 0);
        PSDIR,PSPULSE : IN std_logic;
```

```
                ENCLK0,ENCLK1,ENCLK2,ENCLK3 : IN std_logic;
                ENCLK4,ENCLK5,ENCLK6 : IN std_logic;
                SSCPOL, SSCON: IN std_logic;
                SSCMDSEL : IN std_logic_vector(6 downto 0);
                SSCMDSEL_FRAC : IN std_logic_vector(2 downto 0);
                LOCK : OUT std_logic;
                CLKOUT0, CLKOUT1 : OUT std_logic;
                CLKOUT2, CLKOUT3 : OUT std_logic;
                CLKOUT4, CLKOUT5 : OUT std_logic;
                CLKOUT6, CLKFBOUT : OUT std_logic
        );
END COMPONENT;
uut:PLL
        GENERIC MAP(
                FCLKIN => "100.0",
                DYN_IDIV_SEL => "FALSE",
                IDIV_SEL => 1,
                DYN_FBDIV_SEL => "FALSE",
                FBDIV_SEL => 1,
                DYN_ODIV0_SEL => "FALSE",
                ODIV00_SEL => 8,
                DYN_ODIV1_SEL => "FALSE",
                ODIV1_SEL => 8,
                DYN_ODIV2_SEL => "FALSE",
                ODIV2_SEL => 8,
                DYN_ODIV3_SEL => "FALSE",
                ODIV3_SEL => 8,
                DYN_ODIV4_SEL => "FALSE",
                ODIV4_SEL => 8,
                DYN_ODIV5_SEL => "FALSE",
                ODIV5_SEL => 8,
                DYN_ODIV6_SEL => "FALSE",
                ODIV6_SEL => 8,
                DYN_MDIV_SEL => "FALSE",
                MDIV_SEL => 8,
```

```
MDIV_FRAC_SEL => 0,
ODIV0_FRAC_SEL => 0,
CLKOUT0_EN => "TRUE",
CLKOUT1_EN => " FALSE ",
CLKOUT2_EN => " FALSE ",
CLKOUT3_EN => " FALSE ",
CLKOUT4_EN => " FALSE ",
CLKOUT5_EN => " FALSE ",
CLKOUT6_EN => " FALSE ",
DYN_DT0_SEL => "FALSE",
DYN_DT1_SEL => "FALSE",
DYN_DT2_SEL => "FALSE",
DYN_DT3_SEL => "FALSE",
CLKOUT0_DT_DIR => '1',
CLKOUT1_DT_DIR => '1',
CLKOUT2_DT_DIR => '1',
CLKOUT3_DT_DIR => '1',
CLKOUT0_DT_STEP => 0,
CLKOUT1_DT_STEP => 0,
CLKOUT2_DT_STEP => 0,
CLKOUT3_DT_STEP => 0,
CLK0_IN_SEL => '0',
CLK0_OUT_SEL => '0',
CLK1_IN_SEL => '0',
CLK1_OUT_SEL => '0',
CLK2_IN_SEL => '0',
CLK2_OUT_SEL => '0',
CLK3_IN_SEL => '0',
CLK3_OUT_SEL => '0',
CLK4_IN_SEL => "00",
CLK4_OUT_SEL => '0',
CLK5_IN_SEL => '0',
CLK5_OUT_SEL => '0',
CLK6_IN_SEL => '0',
CLK6_OUT_SEL => '0',
```

```
CLKFB_SEL=> "INTERNAL",
DYN_DPA_EN => "FALSE",
DYN_PE0_SEL => "FALSE",
DYN_PE1_SEL => "FALSE",
DYN_PE2_SEL => "FALSE",
DYN_PE3_SEL => "FALSE",
DYN_PE4_SEL => "FALSE",
DYN_PE5_SEL => "FALSE",
DYN_PE6_SEL => "FALSE",
CLKOUT0_PE_COARSE => 0,
CLKOUT0_PE_FINE => 0,
CLKOUT1_PE_COARSE => 0,
CLKOUT1_PE_FINE => 0,
CLKOUT2_PE_COARSE=> 0,
CLKOUT2_PE_FINE => 0,
CLKOUT3_PE_COARSE => 0,
CLKOUT3_PE_FINE => 0,
CLKOUT4_PE_COARSE => 0,
CLKOUT4_PE_FINE => 0,
CLKOUT5_PE_COARSE => 0,
CLKOUT5_PE_FINE => 0,
CLKOUT6_PE_COARSE => 0,
CLKOUT6_PE_FINE => 0,
DE0_EN => "FALSE",
DE1_EN => "FALSE",
DE2_EN => "FALSE",
DE3_EN => "FALSE",
DE4_EN => "FALSE",
DE5_EN => "FALSE",
DE6_EN => "FALSE",
RESET_I_EN => "FALSE",
RESET_O_EN => "FALSE",
DYN_ICP_SEL => "FALSE",
ICP_SEL => "XXXXXX",
DYN_LPF_SEL => "FALSE",
```

```
        LPR_RES => "XXX",
        LPR_CAP => "00",
        SSC_EN => "FALSE"
    )
    PORT MAP(
        LOCK=>lock,
        CLKOUT0=>clkout0,
        CLKOUT1=>clkout1,
        CLKOUT2=>clkout2,
        CLKOUT3=>clkout3,
        CLKOUT4=>clkout4,
        CLKOUT5=>clkout5,
        CLKOUT6=>clkout6,
        CLKFBOUT=>clkfbout,
        CLKIN=>clkin,
        CLKFB=>clkfb,
        RESET=>reset,
        PLLPWD=>pllpwd,
        RESET_I=>reseti,
        RESET_O=>reseto,
        FBDSEL=>fbdsel,
        IDSEL=>idsel,
        MDSEL=>mdsel,
        MDSEL_FRAC=>mdsel_frac,
        ODSEL0=>odesl0,
        ODSEL0_FRAC=>odesl0_frac,
        ODSEL1=>odesl1,
        ODSEL2=>odesl2,
        ODSEL3=>odesl3,
        ODSEL4=>odesl4,
        ODSEL5=>odesl5,
        ODSEL6=>odesl6,
        DT0=>dt0,
        DT1=>dt1,
        DT2=>dt2,
        DT3=>dt3,
        ICPSEL=>icpsel,
        LPFRES=>lpfres,
        LPFCAP=>lpfcap,
        PSSEL=>pssel,
        PSDIR=>psdir,
```

```
        PSPULSE=>pspulse,
        ENCLK0=>enclk0,
        ENCLK1=>enclk1,
        ENCLK2=>enclk2,
        ENCLK3=>enclk3,
        ENCLK4=>enclk4,
        ENCLK5=>enclk5,
        ENCLK6=>enclk6,
        SSCPOL=>sscpol,
        SSCON=>sscon,
        SSCMDSEL=>sscmdsel,
        SSCMDSEL_FRAC=>sscmdsel_frac
    );
```

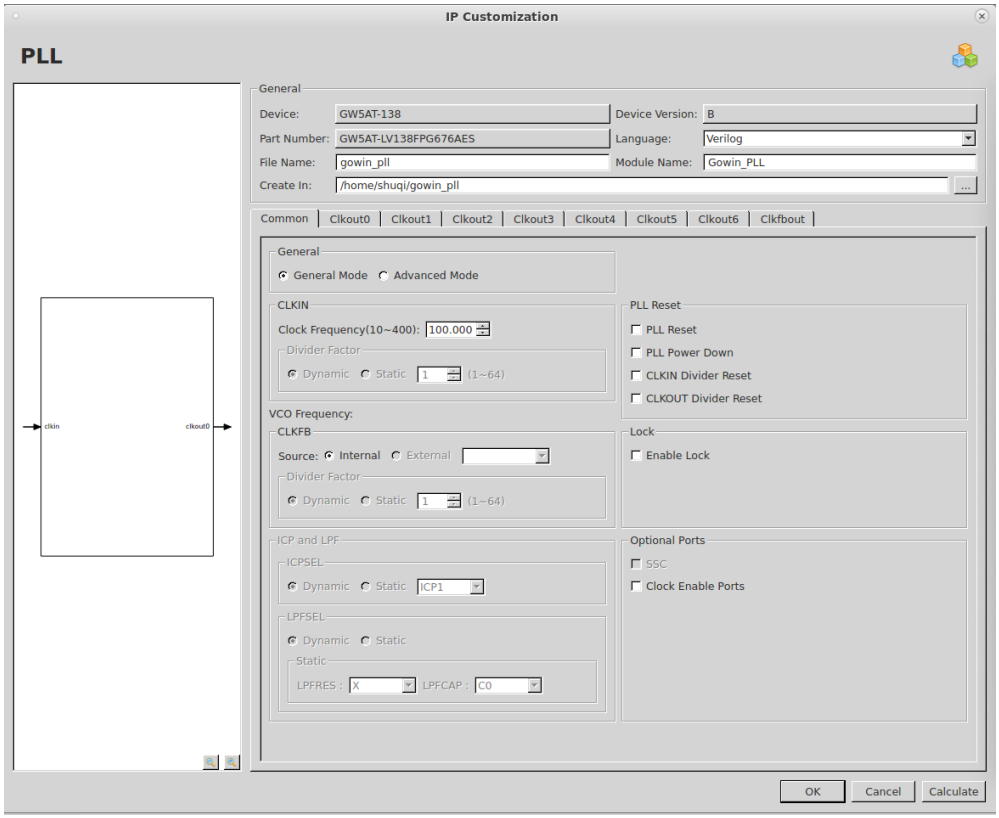
5.1.2 IP 调用

在 IP Core Generator 界面中，单击“PLL”，界面右侧会显示 PLL 的相关信息概要。

IP 配置

在 IP Core Generator 界面中双击“PLL”，弹出 PLL 的“IP Customization”窗口。该窗口包括“General”配置框、"Options"配置框和端口显示框图，如图 5-5 所示。

图 5-5 PLL 的 IP Customization 窗口结构



1. General 配置框
   General 配置框用于配置产生的 IP 设计文件的相关信息。PLL 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。
2. Options 配置框

   Options 配置框用于用户自定义配置 IP，Options 配置框如图 5-5 所示。

   - General：配置一般模式和高级模式。一般模式下输入输入时钟频率和输出时钟频率，软件会自动计算不同的分频系数；高级模式适用于高级用户，允许输入输入频率和不同分频系数得到预期的输出频率；
   - CLKIN：配置 PLL 输入时钟的频率，分频参数的设置
     - “Clock Frequency”（频率范围）配置输入时钟的频率；
     - “Divide Factor”可在高级模式下配置分频参数，支持动态模式“Dynamic”和静态模式“Static”，静态模式下可配置分频参数的具体数值，范围为 1~64。
     - “VCO Frequency”为计算得到的 VCO 的频率，只读。
   - CLKFB：配置 PLL 反馈时钟的源和倍频参数。
     - 配置反馈时钟的源时，“Source”选项可选择“Internal”和“External”；如果选择 Internal，则反馈来自内部；如果选 External，则反馈来自 CLKOUT0~6 和 CLKFBOUT 中的一个，用户可自行选择；
     - Divide Factor 可在高级模式下配置倍频参数，支持动态模式“Dynamic”和静态模式“Static”，静态模式下可配置倍频参数的具体数值，范围为 1~64。
   - ICP and LPF：动态控制 ICP 电流大小，电流随着取值的增大而增大，值为 0 时电流最小。
   - LPFSEL：
     - 动态控制 LPFRES 大小，RES 取值范围由小到大为 R0~R7，R0 对应的带宽最大，R7 对应的带宽最小；
     - 动态控制 LPFCAP 大小，CAP 取值范围由小到大为 C0~C2。
   - PLL Reset：
     - 如果选择“PLL Reset”，PLL 全部复位信号，复位数字电路，配置 PLLO 的 RESET 使能模式；
     - 如果选择“PLL Power Down”，则对模拟电路 power down；
     - 如果选择“CLKIN Divider Reset”，则配置使能 RESET_I；
     - 如果选择“CLKOUT Divider Reset”，则配置使能 RESET_O。

- Lock：PLL 锁定指示信号。
- Optional Ports
  - 频率扩展时钟（SSC）必须在 Advanced Mode 模式才能够被支持；
  - 如果选择“SSC”，则反馈只能来自内部。
- CLKOUT1(以 CLKOUT1 为例，其他可参考 CLKOUT1)

General 配置：

- 用户可选择使能 CLKOUT1~CLKOUT6，“CLKOUT0”默认为使能，且用户不可配置；
- 如果选择不使能“CLKOUT1”，则该页面的所有选项不可选，默认为不使能；
- 如果选择“Bypass”，且选择“Enable CLKOUT1 Divider”，则为 Bypass in 模式，反之不选 Enable CLKOUT1 Divider，则为 Bypass out 模式；
- Bypass 选项可配置输出时钟的旁路功能；“Enable CLKOUTA Divider”选项可配置 VCO 时钟的旁路功能；
- “Expected Frequency（频率范围）”在一般模式下配置期望的输出时钟 CLKOUTA 的频率，非 bypass 模式下范围为 6.25M~1000M；
- “Tolerance（%）”配置 CLKOUTA 期望频率和计算出的实际频率的允许误差；
- “Actual Frequency”显示经计算得出的 CLKOUTA 实际频率，无需用户配置。

CLKOUT1 Divider Factor 配置

- 用户可对 CLKOUT1 的输出分频器进行设置，可静态调整或者动态调整，范围为 1~128，配置不合理时，单击“Calculate”或“OK”，会弹出提示窗口提示错误。

Phase 配置：

- PLL 相位调整支持静态和动态两种调整方式，且 0~6 通道均支持相位调整。

Duty Cycle 配置：

- PLL 占空比调整只支持动态调整，且 0~6 通道均支持，占空比调整电路是针对 VCO 设计的，对 Bypass in 或级联模式，不允许占空比调整。

Duty Trim 配置

- PLL 占空比微调支持静态和动态两种调整方式，且只有 0~3 通道支持占空比微调。

- CLKFBOUT：

  - 可通过页面切换对 CLKFBOUT 进行参数配置，配置参数可以参考上文的 CLKOUT1。

- Calculate：

  - 计算配置的分频参数、倍频参数和 VCO 参数是否合理，若不合理，单击“Calculate”，弹出"error”窗口并且提示错误所在；若配置正确，单击“Calculate”，弹出"succeed”窗口提示配置成功。

3. 端口显示框图
   端口显示框图显示 IP Core 的配置结果示例框图，如图 5-5 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_pll.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 PLL；
- IP 设计使用模板文件“gowin_pll_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_pll.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

5.2 PLLA

5.2.1 原语介绍

Arora V FPGA 提供了锁相环 PLLA，支持 7 路时钟输出，每路时钟可独立基于给定的参考输入时钟进行时钟频率、相位和占空比调整。

适用器件

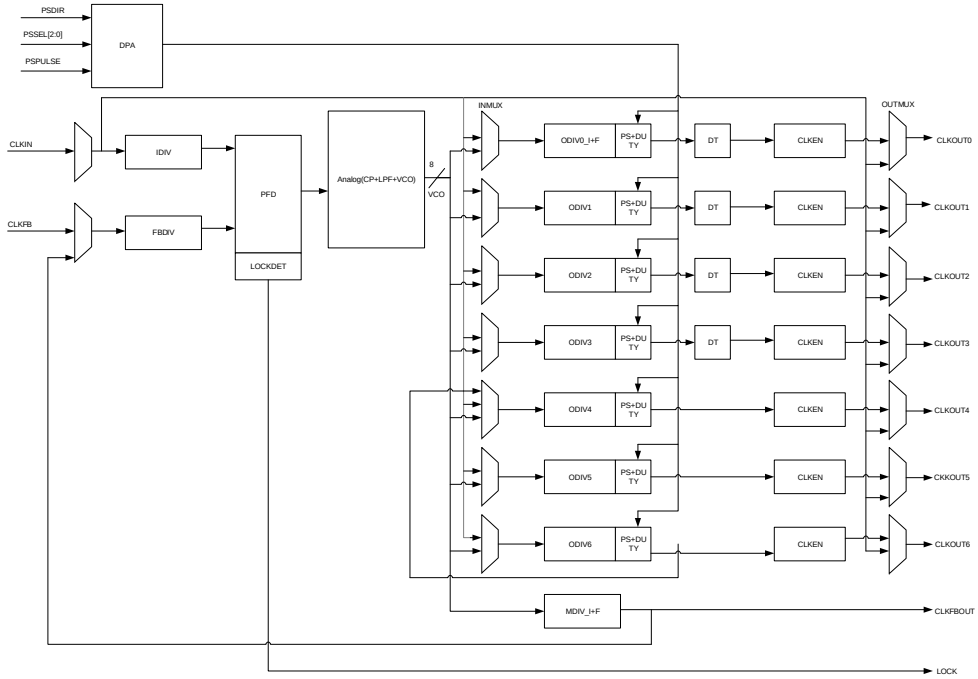
表 5-11 PLLA 适用器件

家族	系列	器件
晨熙®	GW5A	GW5A-25, GW5A-60
	GW5AR	GW5AR-25
	GW5AS	GW5AS-25
	GW5ANRT	GW5ANRT-15
	GW5ANT	GW5ANT-15
	GW5AT	GW5AT-15, GW5AT-60
	GW5ART	GW5ART-15

功能描述

PLLA 模块的结构框图如图 5 7 所示。

图 5-6 PLLA 示意图



PLLA 可基于给定的参考输入时钟进行时钟频率调整、相位调整、占空比调整来产生不同频率、相位和占空比的输出时钟。CLKOUT0 和 CLKFBOUT 支持 1/8 小数频率调整，CLKOUT0~CLKOUT3 支持动静态微调占空比。同时 PLLA 支持 CLKOUT6 到 CLKOUT4 的内部级联，支持 SSC 功能，支持时钟去歪斜以实现 CLKIN 和 CLKOUT 的对齐。

若要得到正确的时钟输出，输入时钟频率必须按照 *FPGA 产品数据手册*中描述的频率范围进行设置。

PLLA 可以对输入时钟 CLKIN 进行频率调整（倍频和分频），计算公式如下：

1. Fpfd=Fclkin/IDIV
2. Fclkfb=Fpfd*FBDIV
3. 根据不同反馈方式，VCO 频率计算公式不同:

- 内部反馈

  Fvco=Fclkfb*MDIV

- 外部反馈

  Fvco=Fclkfb*MDIV ---- CLKFBOUT 反馈到 CLKFB

  Fvco=Fclkfb*ODIVx ---- CLKOUTx 反馈到 CLKFB

4. Fclkfbout= Fvco/MDIV
5. 根据 INMUX 和 OUTMUX 选择不同的模式，CLKOUT 通道输出频率计算公式不同:

- VCO in 模式（INMUX 选择来自 VCO）：Fclkoutx=Fvco/ODIVx

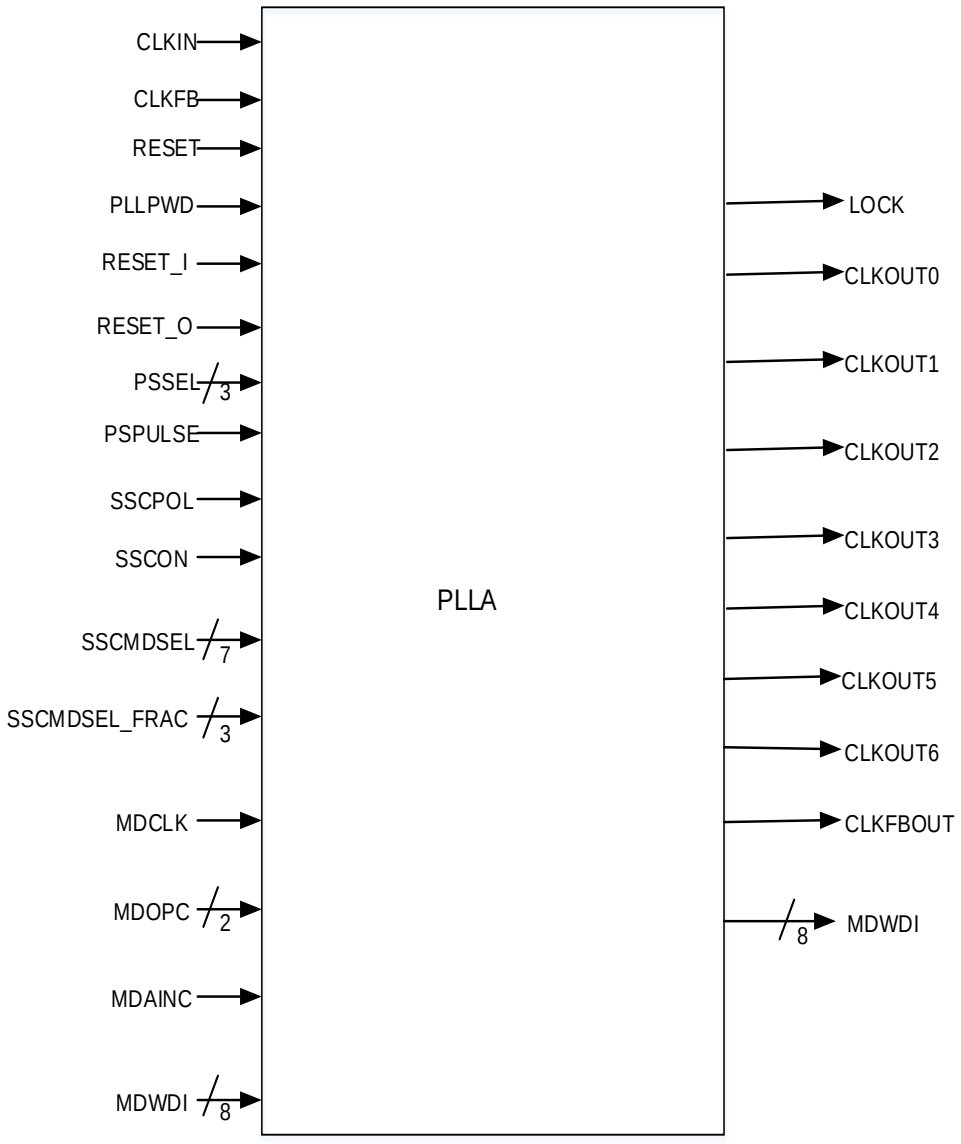
- Bypass in 模式（INMUX 选择来自 CLKIN）：Fclkoutx=Fclkin/ODIVx
- Bypass out 模式（OUTMUX 选择来自 CLKIN）：Fclkoutx=Fclkin
- CAS 模式（仅通道 4）：Fclkout4= Fclkout6[1]/ODIV4

注！

- Fclkin 为参考输入时钟 CLKIN 频率；
- Fclkoutx:x=0~6，为 0~6 通道的输出时钟频率；
- Fclkfb 为反馈输入时钟 CLKFB 频率；
- Fpfd 为 PFD 鉴相频率；
- IDIV、FBDIV、MDIV、ODIVx (x=0~6)为不同分频器的分频系数,即可通过调整不同分频系数得到期望频率的时钟信号；
- [1] Fclkout6 指的是 6 通道 ODIV6 输出的时钟。

端口示意图

图 5-7 PLLA 端口示意图



端口介绍

表 5-12 PLLA 端口介绍

端口名	I/O	描述
CLKIN	输入	参考时钟输入
CLKFB	输入	反馈时钟输入
RESET	输入	PLL 全部复位信号，复位数字电路，高电平有效。
PLLPWD	输入	PLL power down 信号，对模拟电路 power down，高电平有效。
RESET_I	输入	带 IDIV 的 PLL 全复位，高电平有效，一般用于内部测试使用。
RESET_O	输入	复位 ODIV 和部分数字电路，高电平有效，一般用于内部测试使用。
PSDIR	输入	动态控制相位移动方向
PSSEL[2:0]	输入	动态控制相位移动通道选择
PSPULSE	输入	动态控制相位移动时钟脉冲
SSCPOL	输入	避免 timing 冲突的可选配置： ● 0：CLK Rising Edge ● 1：CLK Falling Edge
SSCON	输入	动态控制 SSC 使能
SSCMDSEL[6:0]	输入	动态控制 SSC MDIV 整数取值,建议 SSC MDIV 实际值范围 16~24（对应 Fpfd 为 50M）、32~48（对应 Fpfd 为 25M），SSC MDIV 实际值为 128-SSCMDSEL。
SSCMDSEL_FRAC[2:0]	输入	动态控制 SSC MDIV 小数取值，小数取值 1/8。
CLKOUT0	输出	0 通道时钟输出（默认）
CLKOUT1	输出	1 通道时钟输出
CLKOUT2	输出	2 通道时钟输出
CLKOUT3	输出	3 通道时钟输出
CLKOUT4	输出	4 通道时钟输出
CLKOUT5	输出	5 通道时钟输出
CLKOUT6	输出	6 通道时钟输出
CLKFBOUT	输出	反馈时钟输出
LOCK	输出	PLL 锁定指示： ● 1：锁定 ● 0：失锁
MDCLK	输入	时钟输入信号,DRP 端口上的所有信号的翻转均由该时钟上升沿触发
MDOPC [1:0]	输入	操作类型编码: 00:无读写操作(NOOP) 10:读操作(RD) 01:写操作(WR) 11:无论 MDAINC 是否为高电平，寄存器地址保持不变。
MDAINC	输入	地址增加指示信号，高有效。
MDWDI[7:0]	输入	需要通过 mDRP 端口写入的数据

端口名	I/O	描述
MDRDO[7:0]	输出	从 mDRP 端口读出的数据

参数介绍

表 5-13 PLLA 参数介绍

参数名	类型	取值范围	默认值	描述
FCLKIN	string	"19"~"800"	"100.0"	参考时钟频率(MHz)
IDIV_SEL	integer	1~64	1	IDIV 分频系数静态设置，对应实际取值为 1~64。
FBDIV_SEL	integer	1~64	1	FBDIV 分频系数静态设置，对应实际取值为 1~64。
ODIV0_SEL	integer	1~128	8	ODIV0 分频系数整数静态设置
ODIV0_FRAC_SEL	integer	0~7	0	ODIV0 分频系数小数静态设置
ODIV1_SEL	integer	1~128	8	ODIV1 分频系数静态设置
ODIV2_SEL	integer	1~128	8	ODIV2 分频系数静态设置
ODIV3_SEL	integer	1~128	8	ODIV3 分频系数静态设置
ODIV4_SEL	integer	1~128	8	ODIV4 分频系数静态设置
ODIV5_SEL	integer	1~128	8	ODIV5 分频系数静态设置
ODIV6_SEL	integer	1~128	8	ODIV6 分频系数静态设置
MDIV_SEL	integer	2~128	8	MDIV 分频系数整数静态设置
MDIV_FRAC_SEL	string	0~7	0	MDIV 分频系数小数静态设置
CLKOUT0_EN	string	"TRUE", "FALSE"	"TRUE"	0 通道时钟输出使能
CLKOUT1_EN	string	"TRUE", "FALSE"	"FALSE"	1 通道时钟输出使能
CLKOUT2_EN	string	"TRUE", "FALSE"	"FALSE"	2 通道时钟输出使能
CLKOUT3_EN	string	"TRUE", "FALSE"	"FALSE"	3 通道时钟输出使能
CLKOUT4_EN	string	"TRUE", "FALSE"	"FALSE"	4 通道时钟输出使能
CLKOUT5_EN	string	"TRUE", "FALSE"	"FALSE"	5 通道时钟输出使能
CLKOUT6_EN	string	"TRUE", "FALSE"	"FALSE"	6 通道时钟输出使能
CLKOUT0_DT_DIR	binary	1'b1, 1'b0	1'b1	0 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿 ● 1'b0: - 占空比减少，以下降沿对齐为基准， 调整上升沿。

参数名	类型	取值范围	默认值	描述
CLKOUT1_DT_DIR	binary	1'b1, 1'b0	1'b1	1 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿； ● 1'b0: - 占空比减少，以下降沿对齐为基准，调整上升沿。
CLKOUT2_DT_DIR	binary	1'b1, 1'b0	1'b1	2 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿； ● 1'b0: - 占空比减少，以下降沿对齐为基准，调整上升沿。
CLKOUT3_DT_DIR	binary	1'b1, 1'b0	1'b1	3 通道占空比静态微调方向 ● 1'b1: + 占空比增加，以上升沿对齐为基准，调整下降沿； ● 1'b0: - 占空比减少，以下降沿对齐为基准，调整上升沿。
CLKOUT0_DT_STEP	integer	0,1,2,4	0	0 通道占空比静态微调步长，每步 50ps。
CLKOUT1_DT_STEP	integer	0,1,2,4	0	1 通道占空比静态微调步长，每步 50ps。
CLKOUT2_DT_STEP	integer	0,1,2,4	0	2 通道占空比静态微调步长，每步 50ps。
CLKOUT3_DT_STEP	integer	0,1,2,4	0	3 通道占空比静态微调步长，每步 50ps。
CLK0_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV0 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK0_OUT_SEL	binary	1'b0, 1'b1	1'b0	0 通道输出时钟来源选择 ● 1'b0: 来自 ODIV0 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK1_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV1 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK1_OUT_SEL	binary	1'b0, 1'b1	1'b0	1 通道输出时钟来源选择 ● 1'b0：来自 ODIV1 输出 ● 1'b1：输出时钟旁路来自 CLKIN
CLK2_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV2 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 2'b1: 输出时钟旁路来自 CLKIN
CLK2_OUT_SEL	binary	1'b0, 1'b1	1'b0	2 通道输出时钟来源选择 ● 1'b0: 来自 ODIV2 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK3_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV3 输入时钟来源选择

参数名	类型	取值范围	默认值	描述
				● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK3_OUT_SEL	binary	1'b0, 1'b1	1'b0	3 通道输出时钟来源选择 ● 1'b0: 来自 ODIV3 的输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK4_IN_SEL	binary	2'b00, 2'b01,2'b10	2'b00	ODIV4 输入时钟来源选择 ● 2'b00: 来自 VCO 输出 ● 2'b01: 级联来自 CLKCAS_6 ● 2'b10: 输出时钟旁路来自 CLKIN
CLK4_OUT_SEL	binary	1'b0, 1'b1	1'b0	4 通道输出时钟来源选择 ● 1'b0: 来自 ODIV4 的输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK5_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV5 输入时钟来源选择 ● 1'b0: 来自 VCO 输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLK5_OUT_SEL	binary	1'b0, 1'b1	1'b0	5 通道输出时钟来源选择 ● 1'b0: 来自 ODIV5 的输出 ● 1'b1: 输出时钟旁路来自 CLKIN
CLKFB_SEL	string	"INTERNAL", "EXTERNAL"	"INTERNAL"	CLKFB 来源选择 ● INTERNAL：来自内部 CLKOUT 反馈 ● EXTERNAL：来自外部信号反馈
DYN_DPA_EN	string	"TRUE", "FALSE"	"FALSE"	动态相移调整使能
CLKOUT0_PE_COARSE	integer	0~127	0	0 通道相移粗调静态设置
CLKOUT0_PE_FINE	integer	0~7	0	0 通道相移微调静态设置
CLKOUT1_PE_COARSE	integer	0~127	0	1 通道相移粗调静态设置
CLKOUT1_PE_FINE	integer	0~7	0	1 通道相移微调静态设置
CLKOUT2_PE_COARSE	integer	0~127	0	2 通道相移粗调静态设置
CLKOUT2_PE_FINE	integer	0~7	0	2 通道相移微调静态设置
CLKOUT3_PE_COARSE	integer	0~127	0	3 通道相移粗调静态设置
CLKOUT3_PE_FINE	integer	0~7	0	3 通道相移微调静态设置
CLKOUT4_PE_COARSE	integer	0~127	0	4 通道相移粗调静态设置
CLKOUT4_PE_FINE	integer	0~7	0	4 通道相移微调静态设置

参数名	类型	取值范围	默认值	描述
CLKOUT5_PE_COARSE	integer	0~127	0	5 通道相移粗调静态设置
CLKOUT5_PE_FINE	integer	0~7	0	5 通道相移微调静态设置
CLKOUT6_PE_COARSE	integer	0~127	0	6 通道相移粗调静态设置
CLKOUT6_PE_FINE	integer	0~7	0	6 通道相移微调静态设置
DYN_PE0_SEL	string	“TRUE","FALSE"	“FALSE"	0 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT0_PE_COARSE 和 CLKOUT0_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE1_SEL	string	“TRUE","FALSE"	“FALSE"	1 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT1_PE_COARSE 和 CLKOUT1_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE2_SEL	string	“TRUE","FALSE"	“FALSE"	2 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT2_PE_COARSE 和 CLKOUT2_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE3_SEL	string	“TRUE","FALSE"	“FALSE"	3 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT3_PE_COARSE 和 CLKOUT3_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE4_SEL	string	“TRUE","FALSE"	“FALSE"	4 通道相位调整静态控制参数或动态控制信号选择

参数名	类型	取值范围	默认值	描述
				● "FALSE"：静态，即选择参数 CLKOUT4_PE_COARSE 和 CLKOUT4_PE_FINE ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE5_SEL	string	“TRUE","FALSE"	“FALSE"	5 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT5_PE_COARSE 和 CLKOUT5_PE_FINE； ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现，同时需 DYN_DPA_EN="TRUE"。
DYN_PE6_SEL	string	“TRUE","FALSE"	“FALSE"	6 通道相位调整静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 CLKOUT6_PE_COARSE 和 CLKOUT6_PE_FINE ● "TRUE"：动态，即选择 DPA 动态信号(PSSEL、PSDIR 和 PSPULSE)来实现同时需，DYN_DPA_EN="TRUE"。
DE0_EN	string	“TRUE","FALSE"	“FALSE"	0 通道(ODIV0=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE0_SEL="TRUE" 时设置 CLKOUT0_PE_COARSE 和 CLKOUT0_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE1_EN	string	“TRUE","FALSE"	“FALSE"	1 通道(ODIV1=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE1_SEL="TRUE" 时设置 CLKOUT1_PE_COARSE 和 CLKOUT1_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE2_EN	string	“TRUE","FALSE"	“FALSE"	2 通道(ODIV2=2~128)占空比调整使能 ● "FALSE"：50%占空比

参数名	类型	取值范围	默认值	描述
				● "TRUE"：DYN_PE2_SEL="TRUE" 时设置 CLKOUT2_PE_COARSE 和 CLKOUT2_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE3_EN	string	“TRUE","FALSE"	“FALSE"	3 通道(ODIV3=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE3_SEL="TRUE" 时设置 CLKOUT3_PE_COARSE 和 CLKOUT3_PE_FINE 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE4_EN	string	“TRUE","FALSE"	“FALSE"	4 通道(ODIV4=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE4_SEL="TRUE" 时设置 CLKOUT4_PE_COARSE 和 CLKOUT4_PE_FIN 作为 falling edge，结合动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE5_EN	string	“TRUE","FALSE"	“FALSE"	5 通道(ODIV5=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE5_SEL="TRUE" 时设置 CLKOUT5_PE_COARSE 和 CLKOUT5_PE_FIN 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
DE6_EN	string	“TRUE","FALSE"	“FALSE"	6 通道(ODIV6=2~128)占空比调整使能 ● "FALSE"：50%占空比 ● "TRUE"：DYN_PE6_SEL="TRUE" 时设置 CLKOUT6_PE_COARSE 和 CLKOUT6_PE_FIN 作为 falling edge，动态相位调整作为 rising edge，实现动态占空比调整(falling edge - rising edge)。
RESET_I_EN	string	"TRUE", "FALSE"	"FALSE"	使能动态信号 RESET_I，若需要使用 RESET_I 端口，需将该参数设为 TRUE。
RESET_0_EN	string	"TRUE", "FALSE"	"FALSE"	使能动态信号 RESET_O，若需要使用 RESET_O 端口，需将该参数设为 TRUE。
DYN_ICP_SEL	string	"TRUE", "FALSE"	"FALSE"	ICPSEL 静态控制参数或动态控制信号选择

参数名	类型	取值范围	默认值	描述
				● FALSE: 静态，即选择参数 ICP_SEL； ● TRUE: 动态，即选择动态信号 ICPSEL。
ICP_SEL	binary	6'bXXXXXX, 6'b000000~6'b111111	6'bXXXXXX	ICP 电流静态设置 ● 6'bXXXXXX：表示软件会自动计算并设置该参数 ● 6'b000000~6'b111111：用户若需自行设置，可根据需要在参数范围内设置
DYN_LPF_SEL	string	"TRUE", "FALSE"	"FALSE"	LPFRES 和 LPFCAP 静态控制参数或动态控制信号选择 ● "FALSE"：静态，即选择参数 LPF_RES 和 LPF_CAP； ● "TRUE"：动态，即选择动态信号 LPFREST 和 LPFCAP。
LPF_RES	binary	3'bXXX,3'b000~3'b111	3'bXXX	LPRRES 静态设置 ● 3'bXXX：表示软件会自动计算并设置该参数 ● 3'b000~3'b111：用户若需设置密级程度可根据需要在参数范围内设置
LPF_CAP	binary	2'b00~2'b10	2'b00	LFPCAP 静态设置
SSC_EN	string	"TRUE", "FALSE"	"FALSE"	SSC 使能

输入时钟分频器（IDIV），用于控制进入 PLL 模块的输入时钟频率。该分频系数可以通过参数 IDIV_SEL 静态调整，对应关系如表 5-14 所示。

表 5-14 IDIV 对应关系

IDIV_SEL(静态)	IDIV 实际值
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
64	64

FBDIV 分频器用于分频反馈信号。该分频系数可以通过参数 FBDIV_SEL 静态调整，对应关系如表 5-15 所示。

表 5-15 FBDIV 对应关系

FBDIV_SEL（静态）	FBDIV 实际值
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
64	64

ODIV 输出分频器，0 通道支持整数分频和小数分频；1~6 通道仅支持整数分频。对于 0 通道，该分频系数可通过参数 ODIV0_SEL 和 ODIV0_FRAC_SEL 静态调整，整数分频系数对应关系如表 5-16 所示，小数分频系数对应关系如表 5-17 所示。

表 5-16 ODIV0 整数分频对照表

ODIV0_SEL（静态）	ODIV0 整数值
1	1
2	2
3	3

ODIV0_SEL（静态）	ODIV0 整数值
4	4
5	5
6	6
7	7
8	8
9	9
……	……
128	128

ODIV0 在整数分频为[2~127]时小数分频才有效。对于 1~6 通道，ODIV 分频系数可以通过 ODSELx(x=1~6)动态调整，也可以通过参数 ODIVx_SEL(x=1~6)静态调整，对应关系可参考 0 通道的小数分频，即表 5-17。

表 5-17 ODIV0 小数分频对照表

ODIV0_FRAC_SEL（静态）	ODIV0 小数值
0	0*0.125=0
1	1*0.125=0.125
2	2*0.125=0.25
3	3*0.125=0.375
4	4*0.125=0.5
5	5*0.125=0.625
6	6*0.125=0.75
7	7*0.125=0.875

MDIV、CLKFB 分频器作用与 FBDIV 类似。该分频系数支持整数和小数分频，可以通过参数 MDIV_SEL 和 MDIV_FRAC_SEL 静态调整，整数分频系数对应关系如表 5-18 所示，小数分频系数对应关系如表 5-19 所示。

表 5-18 MDIV 整数分频对应关系

MDIV_SEL（静态）	MDIV 整数值
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……

MDIV_SEL（静态）	MDIV 整数值
128	128

MDIV 整数值取值范围为[2~128]。

表 5-19 MDIV 小数分频对应关系

MDIV_FRAC_SEL(静态)	MDIV 小数值
0	0*0.125=0
1	1*0.125=0.125
2	2*0.125=0.25
3	3*0.125=0.375
4	4*0.125=0.5
5	5*0.125=0.625
6	6*0.125=0.75
7	7*0.125=0.875

MDIV 在整数分频为[2~127]时小数分频才有效。

相位调整

PLLA 相位调整支持静态和动态两种调整方式，且 0~6 通道均支持相位调整。以 MDIV 通道为基准，相位调整是指 ODIV0~6 相对于 MDIV 的相移。

静态相位调整通过设置参数 CLKOUTx_PE_COARSE 和 CLKOUTx_PE_FINE(x=0~6)来实现。

动态相位调整通过信号 PSSEL、PSDIR、PSPULSE 来实现。PSSEL 用于控制选择通道，PSDIR 用于控制加或减操作，一个 PSPULSE 脉冲下降沿 DYN_FINE 加/减 1，DYN_FINE 上溢或下溢时 DYN_COARSE 加 1 或减 1 操作，其中 DYN_COARSE 的值小于 ODIV。

相位调整计算公式如下:

PS=(COARSE+FINE/8)/ODIV*360，PS 范围[0,360)

0 通道 ODIV=ODIV0 整数值+ODIV0 小数值，1~6 通道 ODIV 仅为整数值。

注!

- DYN_FINE 和 DYN_COARSE 是由 DPA 产生的内部信号，通过 PSSEL、PSDIR、PSPULSE 配合产生，其作用和 CLKOUTx_PE_COARSE 和 CLKOUTx_PE_FINE 一致；
- 公式中 COARSE 和 FINE 指的是选择动态或静态调整之后真正作用于相位调整的值；
- 相位调整电路是针对 VCO 设计的，对 Bypass in 或 CAS 模式，相位调整公式虽仍适用,但需设置 FINE 为 0。

相位调整应用

PLL 所有的 CLKOUT 均支持 相位调整应用，为保证 IO 多路高速时钟之间的相位关系，强烈建议使用 CLKOUT [0:3]（硬件直连 HCLK）。

占空比调整

PLLA 占空比调整只支持动态调整，且 0~6 通道均支持。占空比定义如下：

Duty cycle= (falling edge – rising edge)/cycle_period

其中 falling edge 的位置由静态相移设置决定，定义为 DUTY，rising edge 的位置由动态相移设置的 PHASE 决定，DYN_FINE 和 DYN_COARSE 是由 DPA 产生的内部信号。DUTY 和 PHASE 的计算公式如下(以 1 通道为例):

DUTY=(CLKOUT1_PE_COARSE+CLKOUT1_PE_FINE/8)

PHASE=(DYN_COARSE1+DYN_FINE1/8)

动态占空比计算:

- 若 DUTY>PHASE,Duty cycle=(DUTY-PHASE)/ODIV1
- 若 DUTY<PHASE,Duty cycle=(DUTY-PHASE)/ODIV1+1

注!

- ODIV=1 时不支持动态占空比调整，占空比固定为 50%；
- ODIV>=2 时，DUTY-PHASE 不支持(-0.5,0.5)之间的值；
- 占空比调整电路是针对 VCO 设计的，对 Bypass in 或 CAS 模式，不允许占空比调整，且此两种模式下，如果 ODIV(>2)为奇数时，占空比不是 50%(高电平<低电平，即占空比小于 50%)。

占空比微调

PLLA 占空比微调支持静态和动态两种调整方式，且只有 0~3 通道支持占空比微调，通过设置占空比微调方向和步长来实现。微调方向为 1'b1 时，调节下降沿延时，占空比增加；微调方向为 1'b0 时，调节上升沿延时，占空比减小。具体延时值如表 5-20 所示(以 1 通道为例):

表 5-20 PLLA 占空比微调对照表

动态调整	静态调整		占空比微调延时值
DT1[3:0]	CLKOUT1_DT_DIR	CLKOUT1_DT_STEP	
0111	1'b0	0	0
0110		1	-50ps
0101		2	-100ps
0011		4	-200ps
1111	1'b1	0	0
1110		1	+50ps
1101		2	+100ps
1011		4	+200ps

假设 0、1 通道输出相同频率时钟，对 1 通道时钟进行占空比微调，以 0 通道时钟为参考，具体时序如图 5-8 和图 5-9 所示。

图 5-8 通道 1 占空比微调时序图(微调方向为 1'b1，步数 1)

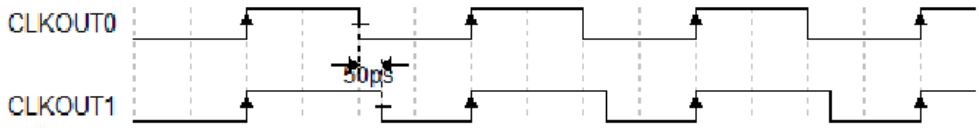
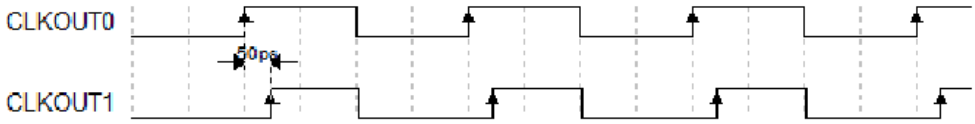


图 5-9 通道 1 占空比微调时序图(微调方向为 1'b0，步数为 1)



原语例化

可以直接实例化原语。

Verilog 例化：

```
PLLA uut (
    .LOCK(lock),
    .CLKOUT0(clkout0),
    .CLKOUT1(clkout1),
    .CLKOUT2(clkout2),
    .CLKOUT3(clkout3),
    .CLKOUT4(clkout4),
    .CLKOUT5(clkout5),
    .CLKOUT6(clkout6),
    .CLKFBOUT(clkfbout),
    .CLKIN(clkin),
    .CLKFB(clkfb),
    .RESET(reset),
    .PLLPWD(pllpwd),
    .RESET_I(reseti),
    .RESET_O(reseto),
    .PSSEL(pssel),
    .PSDIR(psdir),
    .PSPULSE(pspulse),
    .SSCPOL(sscpol),
    .SSCON(sscon),
    .SSCMDSEL(sscmdsel),
    .SSCMDSEL_FRAC(sscmdsel_frac),
```

```
        .MDCLK(mdclk),
        .MDOPC(msopc),
        .MDAINC(mdainc),
        .MDWDI(mdwdi),
        .MDRDO(mdrdo)
);
defparam uut.CLK0_IN_SEL=1'b0;
defparam uut.CLK0_OUT_SEL=1'b0;
defparam uut.CLK1_IN_SEL=1'b0;
defparam uut.CLK1_OUT_SEL=1'b0;
defparam uut.CLK2_IN_SEL=1'b0;
defparam uut.CLK2_OUT_SEL=1'b0;
defparam uut.CLK3_IN_SEL=1'b0;
defparam uut.CLK3_OUT_SEL=1'b0;
defparam uut.CLK4_IN_SEL=2'b00;
defparam uut.CLK4_OUT_SEL=1'b0;
defparam uut.CLK5_IN_SEL=1'b0;
defparam uut.CLK5_OUT_SEL=1'b0;
defparam uut.CLK6_IN_SEL=1'b0;
defparam uut.CLK6_OUT_SEL=1'b0;
defparam uut.CLKFB_SEL="INTERNAL";
defparam uut.CLKOUT0_DT_DIR=1'b1;
defparam uut.CLKOUT0_DT_STEP=0;
defparam uut.CLKOUT0_EN="TRUE";
defparam uut.CLKOUT0_PE_COARSE=0;
defparam uut.CLKOUT0_PE_FINE=0;
defparam uut.CLKOUT1_DT_DIR=1'b1;
defparam uut.CLKOUT1_DT_STEP=0;
defparam uut.CLKOUT1_EN="FALSE";
defparam uut.CLKOUT1_PE_COARSE=0;
defparam uut.CLKOUT1_PE_FINE=0;
defparam uut.CLKOUT2_DT_DIR=1'b1;
defparam uut.CLKOUT2_DT_STEP=0;
defparam uut.CLKOUT2_EN="FALSE";
defparam uut.CLKOUT2_PE_COARSE=0;
defparam uut.CLKOUT2_PE_FINE=0;
defparam uut.CLKOUT3_DT_DIR=1'b1;
defparam uut.CLKOUT3_DT_STEP=0;
defparam uut.CLKOUT3_EN="FALSE";
defparam uut.CLKOUT3_PE_COARSE=0;
defparam uut.CLKOUT3_PE_FINE=0;
defparam uut.CLKOUT4_EN=" TRUE ";
```

```
defparam uut.CLKOUT4_PE_COARSE=0;
defparam uut.CLKOUT4_PE_FINE=0;
defparam uut.CLKOUT5_EN="FALSE";
defparam uut.CLKOUT5_PE_COARSE=0;
defparam uut.CLKOUT5_PE_FINE=0;
defparam uut.CLKOUT6_EN="FALSE";
defparam uut.CLKOUT6_PE_COARSE=0;
defparam uut.CLKOUT6_PE_FINE=0;
defparam uut.DE0_EN="FALSE";
defparam uut.DE1_EN="FALSE";
defparam uut.DE2_EN="FALSE";
defparam uut.DE3_EN="FALSE";
defparam uut.DE4_EN="FALSE";
defparam uut.DE5_EN="FALSE";
defparam uut.DE6_EN="FALSE";
defparam uut.DYN_DPA_EN="FALSE";
defparam uut.DYN_PE0_SEL="FALSE";
defparam uut.DYN_PE1_SEL="FALSE";
defparam uut.DYN_PE2_SEL="FALSE";
defparam uut.DYN_PE3_SEL="FALSE";
defparam uut.DYN_PE4_SEL="FALSE";
defparam uut.DYN_PE5_SEL="FALSE";
defparam uut.DYN_PE6_SEL="FALSE";
defparam uut.FBDIV_SEL=1;
defparam uut.FCLKIN="100.0";
defparam uut.ICP_SEL=6'bXXXXXX;
defparam uut.IDIV_SEL=1;
defparam uut.LPF_CAP=2'b00;
defparam uut.LPF_RES=3'bXXX;
defparam uut.MDIV_FRAC_SEL=0;
defparam uut.MDIV_SEL=8;
defparam uut.ODIV0_FRAC_SEL=0;
defparam uut.ODIV0_SEL=8;
defparam uut.ODIV1_SEL=8;
defparam uut.ODIV2_SEL=8;
defparam uut.ODIV3_SEL=8;
defparam uut.ODIV4_SEL=8;
defparam uut.ODIV5_SEL=8;
defparam uut.ODIV6_SEL=8;
defparam uut.RESET_I_EN="FALSE";
defparam uut.RESET_O_EN="FALSE";
defparam uut.SSC_EN="FALSE";
```

VHDL 例化：

```
COMPONENT PLLA
    GENERIC(
        FCLKIN : STRING := "100.0";
        IDIV_SEL : integer := 1;
        FBDIV_SEL : integer := 1;
        ODIV00_SEL : integer := 8;
        ODIV1_SEL : integer := 8;
        ODIV2_SEL : integer := 8;
        ODIV3_SEL : integer := 8;
        ODIV4_SEL : integer := 8;
        ODIV5_SEL : integer := 8;
        ODIV6_SEL : integer := 8;
        MDIV_SEL : integer := 8;
        MDIV_FRAC_SEL : integer := 0;
        ODIV0_FRAC_SEL : integer := 0;
        CLKOUT0_EN : STRING := "TRUE";
        CLKOUT1_EN : STRING := " FALSE ";
        CLKOUT2_EN : STRING := " FALSE ";
        CLKOUT3_EN : STRING := " FALSE ";
        CLKOUT4_EN : STRING := " FALSE ";
        CLKOUT5_EN : STRING := " FALSE ";
        CLKOUT6_EN : STRING := " FALSE ";
        CLKOUT0_DT_DIR : bit := '1';
        CLKOUT1_DT_DIR : bit := '1';
        CLKOUT2_DT_DIR : bit := '1';
        CLKOUT3_DT_DIR : bit := '1';
        CLKOUT0_DT_STEP : integer := 0;
        CLKOUT1_DT_STEP : integer := 0;
        CLKOUT2_DT_STEP : integer := 0;
        CLKOUT3_DT_STEP : integer := 0;
        CLK0_IN_SEL   : bit := '0';
        CLK0_OUT_SEL : bit := '0';
        CLK1_IN_SEL   : bit_vector := '0';
        CLK1_OUT_SEL : bit := '0';
```

```
CLK2_IN_SEL    : bit_vector := '0';
CLK2_OUT_SEL : bit := '0';
CLK3_IN_SEL    : bit_vector := '0';
CLK3_OUT_SEL : bit := '0';
CLK4_IN_SEL    : bit_vector := "00";
CLK4_OUT_SEL : bit := '0';
CLK5_IN_SEL    : bit_vector := '0';
CLK5_OUT_SEL : bit := '0';
CLK6_IN_SEL    : bit_vector := '0';
CLK6_OUT_SEL : bit := '0';
CLKFB_SEL : STRING := "INTERNAL";
DYN_DPA_EN : STRING := "FALSE";
DYN_PE0_SEL : STRING := "FALSE";
DYN_PE1_SEL : STRING := "FALSE";
DYN_PE2_SEL : STRING := "FALSE";
DYN_PE3_SEL : STRING := "FALSE";
DYN_PE4_SEL : STRING := "FALSE";
DYN_PE5_SEL : STRING := "FALSE";
DYN_PE6_SEL : STRING := "FALSE";
CLKOUT0_PE_COARSE : integer := 0;
CLKOUT0_PE_FINE : integer := 0;
CLKOUT1_PE_COARSE : integer := 0;
CLKOUT1_PE_FINE : integer := 0;
CLKOUT2_PE_COARSE : integer := 0;
CLKOUT2_PE_FINE : integer := 0;
CLKOUT3_PE_COARSE : integer := 0;
CLKOUT3_PE_FINE : integer := 0;
CLKOUT4_PE_COARSE : integer := 0;
CLKOUT4_PE_FINE : integer := 0;
CLKOUT5_PE_COARSE : integer := 0;
CLKOUT5_PE_FINE : integer := 0;
CLKOUT6_PE_COARSE : integer := 0;
CLKOUT6_PE_FINE : integer := 0;
DE0_EN : STRING := "FALSE";
DE1_EN : STRING := "FALSE";
```

```
                DE2_EN : STRING := "FALSE";
                DE3_EN : STRING := "FALSE";
                DE4_EN : STRING := "FALSE";
                DE5_EN : STRING := "FALSE";
                DE6_EN : STRING := "FALSE";
                RESET_I_EN : STRING := "FALSE";
                RESET_O_EN : STRING := "FALSE";
                ICP_SEL : std_logic_vector(5 downto 0) := "XXXXXX";
                LPR_RES : std_logic_vector(2 downto 0) := "XXX";
                LPR_CAP : bit_vector := "00";
                SSC_EN : STRING := "FALSE"
        );
        PORT(
                CLKIN : IN std_logic;
                CLKFB : IN std_logic:='0';
                RESET, PLLPWD : IN std_logic:='0';
                RESET_I, RESET_O : IN std_logic:='0';
                PSSEL : IN std_logic_vector(2 downto 0);
                PSDIR,PSPULSE : IN std_logic;
                SSCPOL, SSCON: IN std_logic;
                SSCMDSEL : IN std_logic_vector(6 downto 0);
                SSCMDSEL_FRAC : IN std_logic_vector(2 downto 0);
                MDCLK, MDINC : IN std logic;
                MDOPC : IN std logic vector(1 downto 0);
                MDWDI : IN std logic vector(7 downto 0);
                MDRDO : OUT std logic vector(7 downto 0);
                LOCK : OUT std_logic;
                CLKOUT0, CLKOUT1 : OUT std_logic;
                CLKOUT2, CLKOUT3 : OUT std_logic;
                CLKOUT4, CLKOUT5 : OUT std_logic;
                CLKOUT6, CLKFBOUT : OUT std_logic
        );
END COMPONENT;
uut:PLLA
        GENERIC MAP(
```

```
FCLKIN => "100.0",
IDIV_SEL => 1,
FBDIV_SEL => 1,
ODIV0_SEL => 8,
ODIV1_SEL => 8,
ODIV2_SEL => 8,
ODIV3_SEL => 8,
ODIV4_SEL => 8,
ODIV5_SEL => 8,
ODIV6_SEL => 8,
MDIV_SEL => 8,
MDIV_FRAC_SEL => 0,
ODIV0_FRAC_SEL => 0,
CLKOUT0_EN => "TRUE",
CLKOUT1_EN => " FALSE ",
CLKOUT2_EN => " FALSE ",
CLKOUT3_EN => " FALSE ",
CLKOUT4_EN => " FALSE ",
CLKOUT5_EN => " FALSE ",
CLKOUT6_EN => " FALSE ",
CLKOUT0_DT_DIR => '1',
CLKOUT1_DT_DIR => '1',
CLKOUT2_DT_DIR => '1',
CLKOUT3_DT_DIR => '1',
CLKOUT0_DT_STEP => 0,
CLKOUT1_DT_STEP => 0,
CLKOUT2_DT_STEP => 0,
CLKOUT3_DT_STEP => 0,
CLK0_IN_SEL => '0',
CLK0_OUT_SEL => '0',
CLK1_IN_SEL => '0',
CLK1_OUT_SEL => '0',
CLK2_IN_SEL => '0',
CLK2_OUT_SEL => '0',
CLK3_IN_SEL => '0',
```

```
CLK3_OUT_SEL => '0',
CLK4_IN_SEL => "00",
CLK4_OUT_SEL => '0',
CLK5_IN_SEL => '0',
CLK5_OUT_SEL => '0',
CLK6_IN_SEL => '0',
CLK6_OUT_SEL => '0',
CLKFB_SEL=> "INTERNAL",
DYN_DPA_EN => "FALSE",
DYN_PE0_SEL => "FALSE",
DYN_PE1_SEL => "FALSE",
DYN_PE2_SEL => "FALSE",
DYN_PE3_SEL => "FALSE",
DYN_PE4_SEL => "FALSE",
DYN_PE5_SEL => "FALSE",
DYN_PE6_SEL => "FALSE",
CLKOUT0_PE_COARSE => 0,
CLKOUT0_PE_FINE => 0,
CLKOUT1_PE_COARSE => 0,
CLKOUT1_PE_FINE => 0,
CLKOUT2_PE_COARSE=> 0,
CLKOUT2_PE_FINE => 0,
CLKOUT3_PE_COARSE => 0,
CLKOUT3_PE_FINE => 0,
CLKOUT4_PE_COARSE => 0,
CLKOUT4_PE_FINE => 0,
CLKOUT5_PE_COARSE => 0,
CLKOUT5_PE_FINE => 0,
CLKOUT6_PE_COARSE => 0,
CLKOUT6_PE_FINE => 0,
DE0_EN => "FALSE",
DE1_EN => "FALSE",
DE2_EN => "FALSE",
DE3_EN => "FALSE",
DE4_EN => "FALSE",
```

```
        DE5_EN => "FALSE",
        DE6_EN => "FALSE",
        RESET_I_EN => "FALSE",
        RESET_O_EN => "FALSE",
        ICP_SEL => "XXXXXX",
        LPR_RES => "XXX",
        LPR_CAP => "00",
        SSC_EN => "FALSE"
    )
    PORT MAP(
        LOCK=>lock,
        CLKOUT0=>clkout0,
        CLKOUT1=>clkout1,
        CLKOUT2=>clkout2,
        CLKOUT3=>clkout3,
        CLKOUT4=>clkout4,
        CLKOUT5=>clkout5,
        CLKOUT6=>clkout6,
        CLKFBOUT=>clkfbout,
        CLKIN=>clkin,
        CLKFB=>clkfb,
        RESET=>reset,
        PLLPWD=>pllpwd,
        RESET_I=>reseti,
        RESET_O=>reseto,
        PSSEL=>pssel,
        PSDIR=>psdir,
        PSPULSE=>pspulse,
        SSCPOL=>sscpol,
        SSCON=>sscon,
        SSCMDSEL=>sscmdsel,
        SSCMDSEL_FRAC=>sscmdsel_frac,
        MDRDO=>mdrdo,
        MDWDI=>mdwdi,
        MDCLK=>mdclk,
        MDOPC=>mdopc,
        MDAINC=>mdainc
    );
```

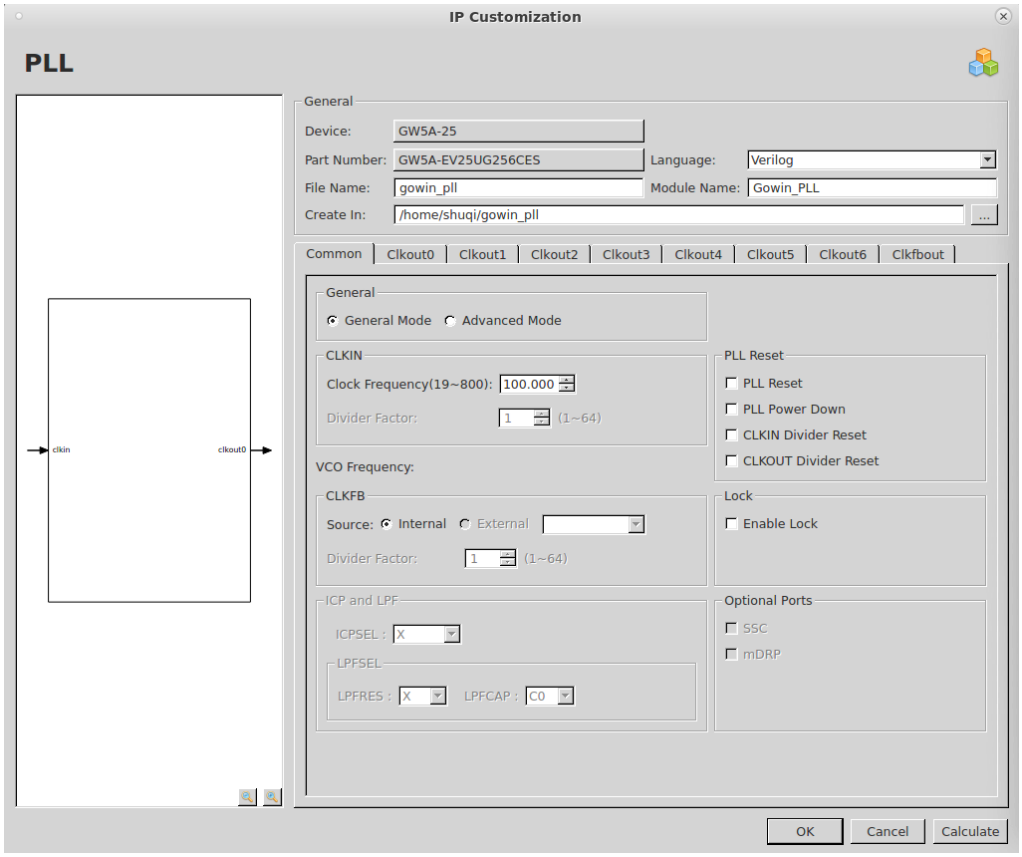
5.2.2 IP 调用

在 IP Core Generator 界面中，单击“PLLA”，界面右侧会显示 PLLA 的相关信息概要。

IP 配置

在 IP Core Generator 界面中双击“PLLA”，弹出 PLL 的“IP Customization”窗口，该窗口包括“General”配置框、"Options”配置框和端口显示框图。

图 5-10 PLLA 的 IP Customization 窗口结构



1. General 配置框

General 配置框用于配置产生的 IP 设计文件的相关信息。PLLA 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。

2. Options 配置框

PLLA 的 Options 配置框用于用户自定义配置 IP，Options 配置框的描述和 PLL 模块的类似，请参考 5.1.2 IP 调用。

3. 端口显示框图

端口显示框图显示 IP Core 的配置结果示例框图，如图 5-10 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以

默认配置为例进行介绍：

- IP 设计文件“gowin_pll.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 PLLA；
- IP 设计使用模板文件“gowin_pll_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_pll.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

5.3 DQS

5.3.1 原语介绍

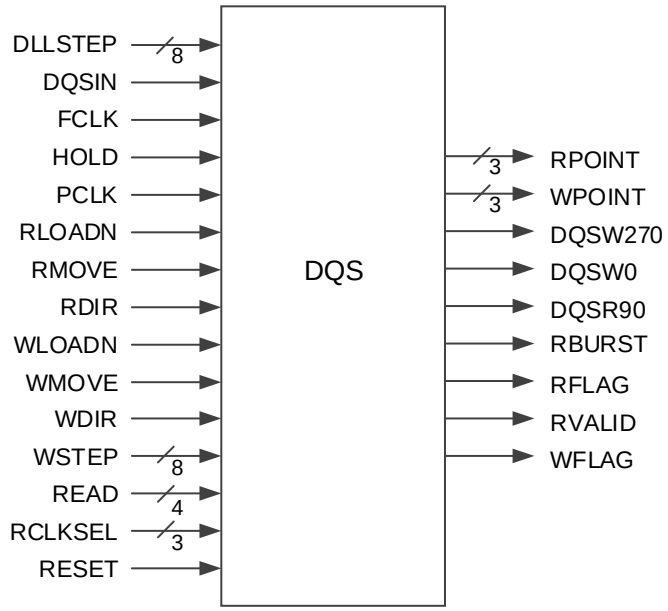
DQS 是 DDR 存储器接口双向数据选通脉冲电路。

功能描述

DQS 是内存控制器 IP 的关键器件，主要用于调整 DQSIN 与 DQSR90、DQSW0 与 DQSW270 信号间的相位关系并完成写平衡、读校准。

端口示意图

图 5-11 DQS 端口示意图



端口介绍

表 5-21 DQS 端口介绍

端口名	I/O	描述
DLLSTEP[7:0]	input	DQS 延时步长控制输入
DQSIN	input	DQS 输入，来自 IO PAD。
FCLK	input	快速时钟，可来自两个不同 FCLK 时钟树输出。
HOLD	input	用于 DQS 写入，停止写入相关信号来同步输出时钟；用于 DQS 读取，来复位 FIFO 计数器。
PCLK	input	主时钟，来自 PCLK 时钟树。
RDIR	input	调整 DDR 读取的延时方向 ● "0"：增加延时 ● "1"：减少延时
RLOADN	input	将 DDR 读取的最终延时步长复位至初始值，低电平有效。
RMOVE	input	RMOVE 为下降沿时改变 DDR 读取的延时步长，每个脉冲改变一次。
WDIR	input	调整 DDR 写入的延时方向 ● "0"：增加延时 ● "1"：减少延时
WLOADN	input	将 DDR 写入的最终延时步长复位至初始值，低电平有效。
WMOVE	input	WMOVE 为下降沿时改变 DDR 写入的延时步长，每个脉冲改变一次。
WSTEP[7:0]	input	用于 DDR 写均衡延时控制
READ[3:0]	input	READ 信号，用于 DDR 读模式。
RCLKSEL[2:0]	input	选择读时钟源和极性控制
RESET	input	DQS 复位输入，高电平有效。
RPOINT[2:0]	output	FIFO 控制读指针，作用于 IOLOGIC 的 RADDR，或通过绕线作用于用户逻辑。
WPOINT[2:0]	output	FIFO 控制写指针，作用于 IOLOGIC 的 WADDR，或通过绕线作用于用户逻辑。
DQSW0	output	PCLK/FCLK 0°相移输出，可作用于 IOLOGIC 的 TCLK，或通过绕线作用于用户逻辑。
DQSW270	output	PCLK/FCLK 270°相移输出，可作用于 IOLOGIC 的 TCLK，或通过绕线作用于用户逻辑。
DQSR90	output	DQSI 相移 90°输出，可作用于 IOLOGIC 的 ICLK，或通过绕线作用于用户逻辑。
RFLAG	output	READ 延时调整输出标志，用以表示读取延时调整 under-flow 或 over-flow。
WFLAG	output	WRITE 延时调整输出标志，用以表示写入延时调整 under-flow 或 over-flow。

端口名	I/O	描述
RVALID	output	READ 模式数据有效标志
RBURST	output	READ 突发检测输出

参数介绍

表 5-22 DQS 参数介绍

参数名	取值范围	默认值	描述
FIFO_MODE_SEL	1'b0 , 1'b1	1'b0	FIFO 模式选择 ● 1'b0: DDR memory 模式 ● 1'b1: GDDR 模式
RD_PNTR	"000", "001", "010", "011", "100", "101", "110", "111"	3'b000	FIFO 读指针设置
DQS_MODE	"X1", "X2_DDR2", "X2_DDR3", "X4", "X2_DDR3_EXT"	"X1"	DQS 模式选择
HWL	"false", "true"	"false"	update0/1 时序关系控制 ● "false": update1 比 update0 提前一个周期； ● "true": update1 和 update 0 时序相同。

连接规则

- DQS 的输入 DQSI 来自 IO PAD；
- DQS 的输出 RPOINT 可连接至 IOLOGIC 的 RADDR，也可作用于用户逻辑；
- DQS 的输出 WPOINT 可连接至 IOLOGIC 的 WADDR，也可作用于用户逻辑；
- DQS 的输出 DQSR90 可连接至 IOLOGIC 的 ICLK，也可作用于用户逻辑；
- DQS 的输出 DQSW0/ DQSW270 可连接至 IOLOGIC 的 TCLK，也可作用于用户逻辑。

原语例化

Verilog 例化：

```
DQS uut (
        .DQSIN(dqs),
        .PCLK(pclk),
        .FCLK(fclk),
        .RESET(reset),
        .READ(read),
        .RCLKSEL(rsel),
        .DLLSTEP(step),
        .WSTEP(wstep),
        .RLOADN(1'b0),
        .RMOVE(1'b0),
        .RDIR(1'b0),
        .WLOADN(1'b0),
        .WMOVE(1'b0),
        .WDIR(1'b0),
        .HOLD(hold),
        .DQSR90(dqsr90),
        .DQSW0(dqsw0),
        .DQSW270(dqsw270),
        .RPOINT(rpoint),
        .WPOINT(wpoint),
        .RVALID(rvalid),
        .RBURST(rburst),
        .RFLAG(rflag),
        .WFLAG(wflag)
);
defparam uut.DQS_MODE = "X1";
defparam uut.FIFO_MODE_SEL = 1'b0;
defparam uut.RD_PNTR = 3'b001;
```

VHDL 例化：

```
COMPONENT DQS
      GENERIC(
            FIFO_MODE_SEL:bit:='0';
```

```
                RD_PNTR : bit_vector:="000";
                DQS_MODE:string:="X1";
                HWL:string:="false"
        );
        PORT(
            DQSIN,PCLK,FCLK,RESET:IN std_logic;
            READ:IN std_logic_vector(3 downto 0);
            RCLKSEL:IN std_logic_vector(2 downto 0);
            DLLSTEP,WSTEP:IN std_logic_vector(7 downto 0);
            RLOADN,RMOVE,RDIR,HOLD:IN std_logic;
            WLOADN,WMOVE,WDIR:IN std_logic;
            DQSR90,DQSW0,DQSW270:OUT std_logic;
            RPOINT, WPOINT:OUT std_logic_vector(2 downto 0);
            RVALID,RBURST,RFLAG,WFLAG:OUT std_logic
        );
END COMPONENT;
uut:DQS
    GENERIC MAP(
                FIFO_MODE_SEL=>'0',
                RD_PNTR=>"000",
                DQS_MODE=>"X1",
                HWL=>"false"
        )
        PORT MAP(
            DQSIN=>dqsin,
            PCLK=>pclk,
            FCLK=>fclk,
            RESET=>reset,
            READ=>read,
            RCLKSEL=>rclksel,
            DLLSTEP=>step,
            WSTEP=>wstep,
            RLOADN=>rloadn,
            RMOVE=>rmove,
            RDIR=>rdir,
```

```
        HOLD=>hold,
        WLOADN=>wloadn,
        WMOVE=>wmove,
        WDIR=>wdir,
        DQSR90=>dqsr90,
        DQSW0=>dqsw0,
        DQSW270=>dqsw270,
        RPOINT=>rpoint,
        WPOINT=>wpoint,
        RVALID=>rvalid,
        RBURST=>rburst,
        RFLAG=>rflag,
        WFLAG=>wflag
    );
```

5.4 DDRDLL

5.4.1 原语介绍

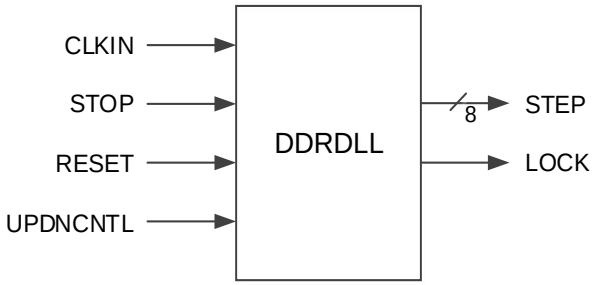
DDRDLL 产生不同相位延时通过 DLLDLY、DQS 等模块作用于 DDR 等接口模块。

功能描述

DDRDLL 可基于给定的输入时钟进行时钟相位调整产生不同相位的延时步长 STEP。计算过的输出 STEP 信号会驱动如 DQS、DLLDLY 模块，同时信号 STEP 也可通过布线送到用户逻辑中去。DDRDLL 的时钟输入来源包括 GCLK 和相邻的 HCLK。

端口示意图

图 5-12 DDRDLL 端口示意图



端口介绍

表 5-23 DDRDLL 端口介绍

端口名	I/O	描述
STEP[7:0]	Output	延时步长输出
LOCK	Output	锁定指示输出 ● 1：锁定 ● 0：失锁
CLKIN	Input	时钟输入
STOP	Input	停止输入时钟和内部震荡时钟
RESET	Input	异步复位输入，高电平有效。
UPDNCNTL	Input	DDRDLL 延时步长更新控制，低电平有效。

参数介绍

表 5-24 DDRDLL 参数介绍

参数名	参数类型	取值范围	默认值	描述
DLL_FORCE	Integer	0,1	0	DDRDLL 强制延时步长、锁定控制 ● TRUE：强制锁定，延时步长为 255（最大），用于较低的输入频率模式； ● FALSE：正常模式，通过 DDRDLL 生成延时步长和锁定信号。
CODESCAL	String	000,001,010,011,100,101, 110, 111	000	DDRDLL 相移配置（45°~135°） ● 000:101° ● 001:112° ● 010:123° ● 011:135° ● 100:79° ● 101:68° ● 110:57° ● 111:45°
SCAL_EN	String	true,false	true	DDRDLL 启用相位偏移功能: ● TRUE：启用，相位偏移根据参数 CODESCAL 设置； ● FALSE：禁用，默认 90°相移。
DIV_SEL	Integer	1’b0,1’b1	1’b0	DDRDLL 锁定模式选择: ● 1'b0：正常锁定模式 ● 1'b1：快速锁定模式

连接合法性规则

DDRDLL 的输出 STEP 可连接至 DQS、DLLDLY 模块，同时也可通过布线送到用户逻辑中去。

原语例化

Verilog 例化

```
DDRDLL uut (
    .STEP(step),
    .LOCK(lock),
    .CLKIN(clkin),
    .STOP(stop),
    .RESET(reset),
    .UPDNCNTL(1’b0)
);
defparam uut.DLL_FORCE = "FALSE";
defparam uut.CODESCAL = "000";
defparam uut.SCAL_EN = " TRUE";
defparam uut.DIV_SEL = 1'b0;
```

Vhdl 例化

```
 COMPONENT DLL
        GENERIC(
                DLL_FORCE: STRING := "FALSE";
                DIV_SEL: bit :='1';
                CODESCAL: STRING :="000";
                SCAL_EN: STRING :="true"
        );
            PORT(
                CLKIN:IN std_logic;
                STOP:IN std_logic;
                RESET:IN std_logic;
                UPDNCNTL:IN std_logic;
                LOCK:OUT std_logic;
                STEP:OUT std_logic_vector(7 downto 0)
             );
END COMPONENT;
uut:DLL
```

```
GENERIC MAP(
                DLL_FORCE=>” FALSE”,
                DIV_SEL=>'1',
                CODESCAL=>"000",
                SCAL_EN=>" TRUE"
)
 PORT MAP(
    CLKIN=>clkin,
    STOP=>stop,
    RESET=>reset,
    UPDNCNTL=>updncntl,
    LOCK=>lock,
    STEP=>step
   );
```

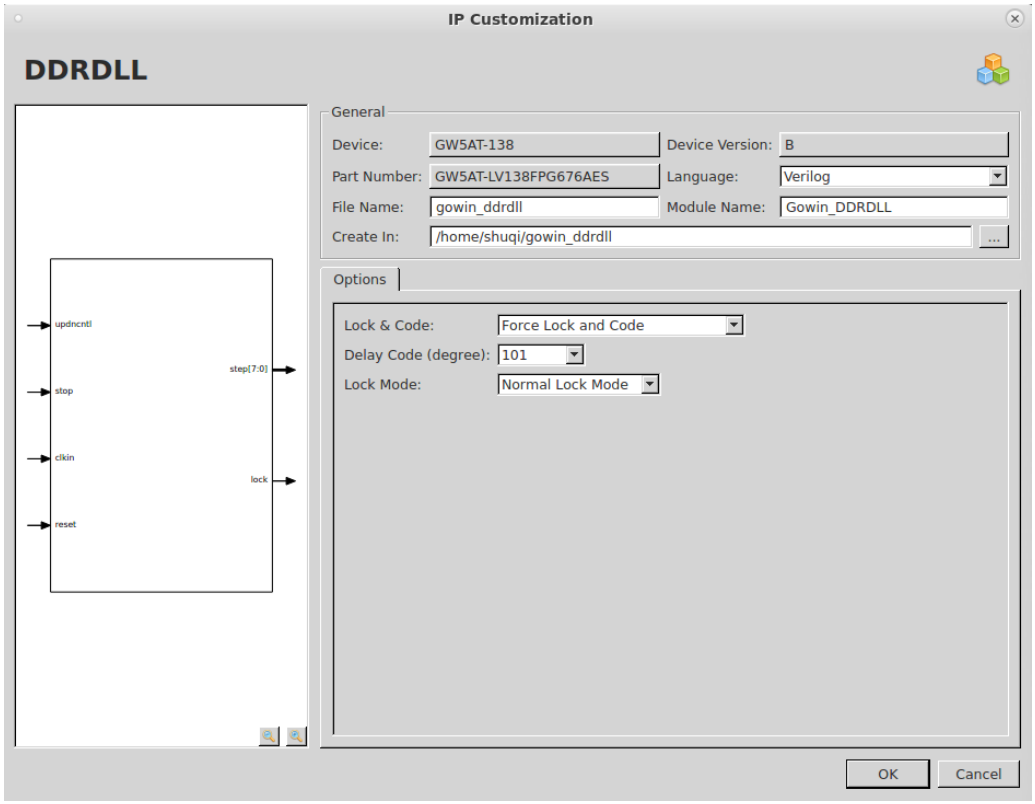
5.4.2 IP 调用

在 IP Core Generator 界面中，单击“DDRDLL”，界面右侧会显示 DDRDLL 的相关信息概要。

IP 配置

在 IP Core Generator 界面中双击“DDRDLL”，弹出 DDRDLL 的“IP Customization”窗口。

图 5-13 DDRDLL 的 IP Customization 窗口结构



1. General 配置框

General 配置框用于配置产生的 IP 设计文件的相关信息。DDRDLL 的 General 配置框的使用和 DCE 模块的类似，请参考参考 3.1.2 IP 调用。

2. Options 配置框

DDRDLL 的 Options 配置框用于用户自定义配置 IP，如图 5-13。

- Lock & Code：设置 DDRDLL 模式。
- Delay Code(degree)：延时设置；
- Lock Mode：设置锁定模式。

3. 端口显示框图

端口显示框图显示 IP Core 的配置结果示例框图，如图 5-13 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_ddrdll.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 DDRDLL；
- IP 设计使用模板文件“gowin_ddrdll_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_ddrdll.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

6 晶振时钟

6.1 OSC

6.1.1 原语介绍

OSC，可编程片内晶振。

适用器件

表 6-1 OSC 适用器件

家族	系列	器件
晨熙®	GW5AT	GW5AT-138, GW5AT-75
	GW5AST	GW5AST-138
	GW5A	GW5A-138
	GW5AS	GW5AS-138

功能描述

可编程片内晶振，为 MSPI 编程模式提供时钟源，还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。

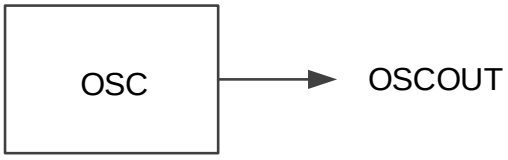
器件输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = f_{osc}/\mathrm{FREQ\_DIV}$；

其中f_{osc}为 210MHz OSC 振荡频率，根据器件除数 FREQ_DIV 为配置参数，取值为 3 和 2~126 的偶数。

端口示意图

图 6-1 OSC 端口示意图



端口介绍

表 6-2 OSC 端口介绍

端口名	I/O	描述
OSCOUT	output	OSC 输出时钟信号

参数介绍

表 6-3 OSC 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	3,2~126（偶数）	100	OSC 分频系数设置

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSC uut(
    .OSCOUT(oscout)
);
defparam uut.FREQ_DIV=100;
```

VHDL 例化：

```
COMPONENT OSC
        GENERIC(
                FREQ_DIV:integer:=100
         );
        PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSC
    GENERIC MAP(
                FREQ_DIV=>100
     )
    PORT MAP(OSCOUT=>oscout);
```

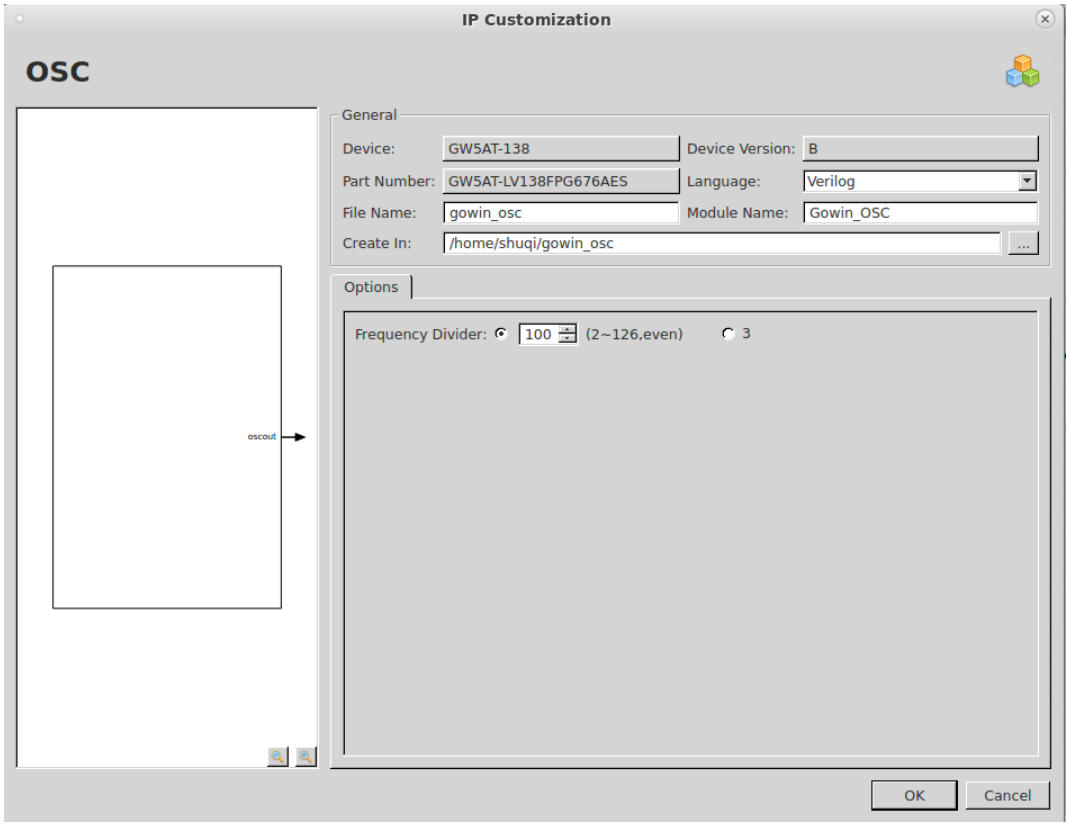
6.1.2 IP 调用

在 IP Core Generator 界面中单击 OSC，界面右侧会显示 OSC 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“OSC”，弹出 OSC 的“IP Customization”窗口，该窗口包括“ General”配置框、“Options”配置框和端口显示框图，如图 6 2 所示。

图 6-2 OSC 的 IP Customization 窗口结构



1. General 配置框
   General 配置框用于配置产生的 IP 设计文件的相关信息。OSC 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。
2. Options 配置框
   Options 配置框用于用户自定义配置 IP，Options 配置框如图 6-2 所示。
   - Frequency Divider：3 和 2~126 之间的偶数。
3. 端口显示框图

   端口显示框图显示 IP Core 的配置结果示例框图，如图 6-2 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_osc.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 OSC；
- IP 设计使用模板文件“gowin_osc_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_osc.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

6.2 OSCA

6.2.1 原语介绍

OSCA，可编程片内晶振。

适用器件

表 6-4 OSCA 适用器件

家族	系列	器件
晨熙®	GW5A	GW5A-25, GW5A-60
	GW5AR	GW5AR-25
	GW5AS	GW5AS-25
	GW5ANRT	GW5ANRT-15
	GW5ANT	GW5ANT-15
	GW5AT	GW5AT-15, GW5AT-60
	GW5ART	GW5ART-15

功能描述

可编程片内晶振，为 MSPI 编程模式提供时钟源，支持动态打开/关闭 OSC 功能。还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。

器件输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = f_{osc}/\mathrm{FREQ\_DIV}$；

其中f_{osc}为 210MHz OSC 振荡频率，根据器件除数 FREQ_DIV 为配置参数，取值为 3 和 2~126 的偶数。

端口示意图

图 6-3 OSCA 端口示意图

端口介绍

表 6-5 OSCA 端口介绍

端口名	I/O	描述
OSCOUT	output	OSC 输出时钟信号
OSCEN	input	时钟使能信号，高电平有效

参数介绍

表 6-6 OSCA 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	3,2~126（偶数）	100	OSC 分频系数设置

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSCA uut(
    .OSCOUT(oscout),
    .OSCEN(oscen)
);
defparam uut.FREQ_DIV=100;
```

VHDL 例化：

```
COMPONENT OSCA
        GENERIC(
                FREQ_DIV:integer:=100
         );
        PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSCA
    GENERIC MAP(
                 FREQ_DIV=>100
     )
    PORT MAP(
                OSCOUT=>oscout,
                OSCEN=>oscen
    );
```

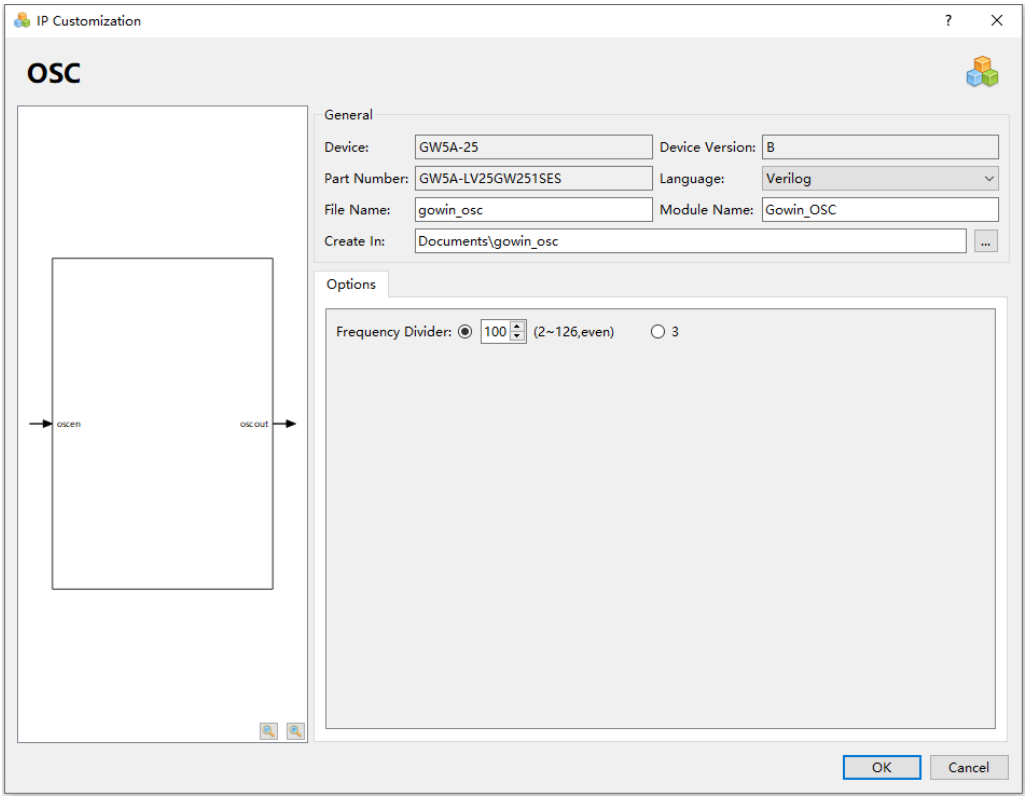
6.2.2 IP 调用

在 IP Core Generator 界面中单击 OSC，界面右侧会显示 OSCA 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“OSC”，弹出 OSCA 的“IP Customization”窗口，该窗口包括“ General”配置框、“Options”配置框和端口显示框图，如图 6-4 所示。

图 6-4 OSCA 的 IP Customization 窗口结构



1. General 配置框

General 配置框用于配置产生的 IP 设计文件的相关信息。OSCA 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。

2. Options 配置框

Options 配置框用于用户自定义配置 IP，OSCA 的 Options 配置框的描述和 OSC 模块的类似，请参考 6.1.2 IP 调用。

3. 端口显示框图

端口显示框图显示 IP Core 的配置结果示例框图，如图 6-4 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_osc.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 OSCA；
- IP 设计使用模板文件“gowin_osc_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_osc.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

6.3 OSCB

6.3.1 原语介绍

OSCB，可编程片内晶振。

适用器件

表 6-7 OSCB 适用器件

家族	系列	器件
晨熙®	GW5ANRT	GW5ANRT-15
	GW5ANT	GW5ANT-15
	GW5ART	GW5ART-15
	GW5AT	GW5AT-15

功能描述

可编程片内晶振，为 MSPI 编程模式提供时钟源，支持动态打开/关闭 OSC 功能，支持动态频率模式选择、动态 TRIM 功能。还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。

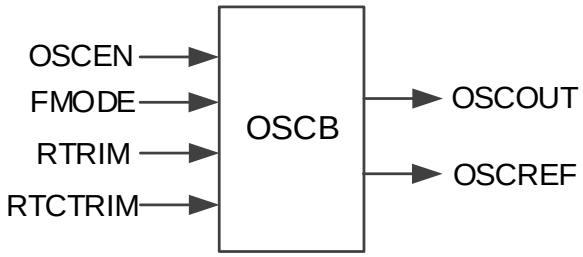
器件输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = f_{osc}/\mathrm{FREQ\_DIV}$；

其中f_{osc}为 210MHz 或 25MHz OSC 振荡频率，根据器件除数 FREQ_DIV 为配置参数，取值为 3 和 2~126 之间的偶数。

端口示意图

图 6-5 OSCB 端口示意图



端口介绍

表 6-8 OSCB 端口介绍

端口名	I/O	描述
OSCOUT	output	OSC 输出时钟信号
OSCREF	output	serDes 和 PLLA 的参考时钟。可输出高精度 25MHz 时钟，也可输出 210Mhz 时钟
OSCEN	input	时钟使能信号，高电平有效

端口名	I/O	描述
FMODE	input	频率模式选择，当参数 FREQ_MODE=”210”时可通过端口 FMODE 来动态选择 210MHz 或 25MHz 基准频率
RTRIM	input	动态调整频率
RTCTRIM	input	动态调整温度系数

参数介绍

表 6-9 OSCB 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	3,2~126（偶数）	10	OSC 分频系数设置
FREQ_MODE	“25”,”210”	“25”	频率模式选择设置
DYN_TRIM_EN	“FALSE”, “TRUE”	“FALSE”	动态 TRIM 使能设置

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSCB uut(
    .OSCOUT(oscout),
    .OSCREF(oscref),
    .FMODE(fmode),
    .RTRIM(rtrim),
    .RTCTRIM(rtctrim),
    .OSCEN(oscen)
);
defparam uut.FREQ_DIV=10;
defparam uut.FREQ_MODE="25";
defparam uut. DYN_TRIM_EN ="FALSE";
```

VHDL 例化：

```
COMPONENT OSCB
        GENERIC(
            FREQ_MODE : string := "25"; --"25":25M; "210":210M
            FREQ_DIV : integer := 10;    --3, 2~126(only even num)
            DYN_TRIM_EN : string := "FALSE" --"FALSE","TRUE"
```

```
    );
    PORT(
        OSCEN : in STD_LOGIC;
        OSCOUT: OUT STD_LOGIC;
        OSCREF: OUT STD_LOGIC;
        FMODE : in STD_LOGIC;
        RTRIM : in   std_logic_vector(7 downto 0);
        RTCTRIM : in   std_logic_vector(5 downto 0)
);
END COMPONENT;
uut:OSCB
    GENERIC MAP(
        FREQ_DIV=>10,
        FREQ_MODE =>"25",
        DYN_TRIM_EN =>"FALSE"
     )
    PORT MAP(
        OSCOUT=>oscout,
        OSCREF=>oscref,
        FMODE=>fmode,
        RTRIM=>rtrim,
        RTCTRIM=>rtctrim,
        OSCEN=>oscen
    );
```

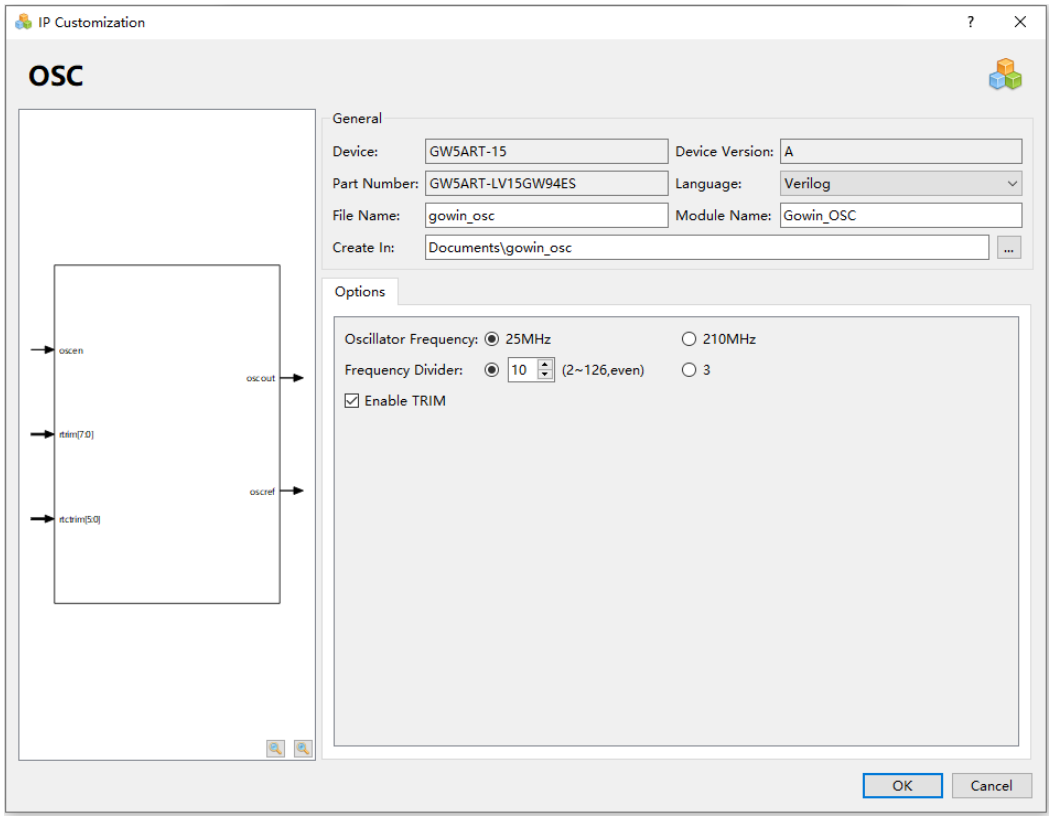
6.3.2 IP 调用

在 IP Core Generator 界面中单击 OSC，界面右侧会显示 OSCB 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“OSC”，弹出 OSCB 的“IP Customization”窗口，该窗口包括“ General”配置框、“Options”配置框和端口显示框图，如图 6-6 所示。

图 6-6 OSCB 的 IP Customization 窗口结构



1. General 配置框

General 配置框用于配置产生的 IP 设计文件的相关信息。OSCB 的 General 配置框的使用和 DCE 模块的类似，请参考 3.1.2 IP 调用。

2. Options 配置框

Options 配置框用于用户自定义配置 IP，OSCB 的 Options 配置框如图 6-4 所示。

- Oscillator Frequency：25MHz 或者 210MHz；
- Frequency Divider：3 和 2~126 的偶数；
- Enable TRIM：使能动态 TRIM 功能；
- Enable OSC Reference：Oscillator Frequency 选择 210MHz 且 Enable TRIM 为 disable 时可以设置 OSCREF 输出。

3. 端口显示框图

端口显示框图显示 IP Core 的配置结果示例框图，如图 6-6 所示。

IP 生成文件

IP 窗口配置完成后，产生以配置文件 File Name 命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_osc.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 OSCB；

- IP 设计使用模板文件“gowin_osc_tmp.v”，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_osc.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。


GOWIN高云
智慧逻辑 定制未来