# Gowin Clock

# User Guide

**UG286-1.9.8E, 02/02/2024**

### Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

## Revision History

| Date | Version | Description |
|---|---|---|
| 05/18/2016 | 1.05E | Initial version published. |
| 07/15/2016 | 1.06E | The graphics standardized. |
| 08/31/2016 | 1.07E | GW2A series of FPGA products supported. |
| 10/27/2016 | 1.08E | GW2A series of FPGA products supported. |
| 09/22/2017 | 1.09E | Modified according to the latest software source library. |
| 10/16/2017 | 1.10E | The description of GW1N and related diagrams added. |
| 01/05/2018 | 1.2E | The description of HCLK updated. |
| 04/20/2018 | 1.3E | PLL ports and parameters updated. |
| 12/04/2019 | 1.4E | The description of primitive updated. |
| 08/18/2020 | 1.5E | The chapter structure modified and content optimized. |
| 01/14/2021 | 1.6E | PLLO, OSCO, CLKDIVG and DCCG modules added. |
| 02/01/2021 | 1.7E | ● CLKOUTD3 timing of rPLL added.<br>● The description of PLLO updated.<br>● GW1NR-2, GW2AN-55C added. |
| 04/13/2021 | 1.7.1E | GWINZ-2 removed. |
| 06/18/2021 | 1.8E | ● GW1N-2B, GW1N-1P5, GW1N-1P5B, GW1NR-2B, GW2AN-18X, and GW2AN-9X added.<br>● Figures updated and Help information of IP Generation removed. |
| 09/10/2021 | 1.9E | Overview of PLL added. |
| 10/12/2021 | 1.9.1E | ● The introduction to the different modes for PLL IP core updated;<br>● The description of CLKDIVG updated. |
| 01/24/2022 | 1.9.2E | Sample code formatting adjusted. |
| 05/20/2022 | 1.9.3E | The description of OSCZ updated. |
| 07/22/2022 | 1.9.4E | ● The description of PLL updated.<br>● OSCZ supported devices updated. |
| 08/10/2022 | 1.9.5E | Device supported of rPLL and OSC updated. |
| 11/01/2022 | 1.9.6E | GW1NS-2, GW1NS-2C, GW1NSE-2C, GW1NSR-2, and GW1NSR-2C removed. |
| 09/12/2023 | 1.9.7E | ● The configuration box File modified to General and Device Version option added on the IP interface.<br>● The description of PLL duty cycle trim updated. |
| 2024/02/02 | 1.9.8E | 5.4.1 Primitive Introduction updated, adding information on delay step size of DLLDLY. |

# Contents

# List of Figures

# List of Tables

# 1 About This Guide

## 1.1 Purpose

This manual provides descriptions of the function, primitive definition and usage of Gowin clock.

## 1.2 Related Documents

The latest user guides are available on GOWINSEMI® Website. Refer to the related documents at www.gowinsemi.com:

- DS100, GW1N series of FPGA Products Data Sheet
- DS117, GW1NR series of FPGA Products Data Sheet
- DS821, GW1NS series of FPGA Products Data Sheet
- DS861, GW1NSR series of FPGA Products Data Sheet
- DS871, GW1NSE series of FPGA Products Data Sheet
- DS841, GW1NZ series of FPGA Products Data Sheet
- DS102, GW2A series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet

## 1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

**Table 1-1 Terminology and Abbreviations**

| Terminology and Abbreviations | Meaning |
|---|---|
| CIU | Configurable Interface Unit |
| CLKDIV | Clock Divider |
| CRU | Configurable Routing Unit |
| DCC | HCLK Duty Cycle Correction |
| DCS | Dynamic Clock Selector |
| DHCEN | Dynamic HCLK Clock Eanble with Inverted Gate |
| DLLDLY | DLL Delay |

| Terminology and Abbreviations | Meaning |
|---|---|
| DQCE | Dynamic Quadrant Clock Enable |
| DQS | Bidirectional Data Strobe Circuit for DDR Memory |
| GCLK | Global Clock |
| HCLK | High-speed Clock |
| LW | Long Wire |
| OSC | Oscillator |
| PCLK | Primary Clock |
| PLL | Phase-locked Loop |

## 1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

# 2 Overview

This chapter describes the clock resources of GOWINSEMI FPGA products, including the dedicated clock input, buffers, and routing resources. The basic clock resources include a series of low-capacity and low-offset interconnect wires that are suitable for high frequency signals and can help reduce clock jitter and improve the performance, which can be applied to all clock signls.

The clock resources and routing are critical for high-performance applications of FPGA. GOWINSEMI FPGA products provide the global clock (GCLK) including primary clock (PCLK) and long wire (LW), which connects directly to all the resources. Besides the global clock network, clock resources such as PLL, high-speed clock HCLK, and DQS in DDR memory interface are also provided.

## 2.1 Global Clock

GCLKs are distributed in quadrants in Gowin FPGA products, and GCLKs in LittleBee® (1K, 2K, 4K) family FPGA productsare divided into two quadrants, L and R, as shown in Figure 2-1. The GCLKs in LittleBee® (9K) and Arora family FPGA products are divided into four quadrants, BL, BR, TL, and TR, as shown in Figure 2-2. Each quadrant provides eight GCLK networks, and the selectable clock sources for each GCLK include dedicated clock input pins and common rouitng resources. Dedicated clock input pins can provide better clock performance.

LW can be used as a control wire to provide clock enable (CE) and reset (SET/RESET) signals to the DFF on one hand, and as a logic wire for general data signals on the other hand.

**Figure 2-1 Quadrant Distribution of GCLKs in the LittleBee® FPGA Family (1K, 2K, 4K)**

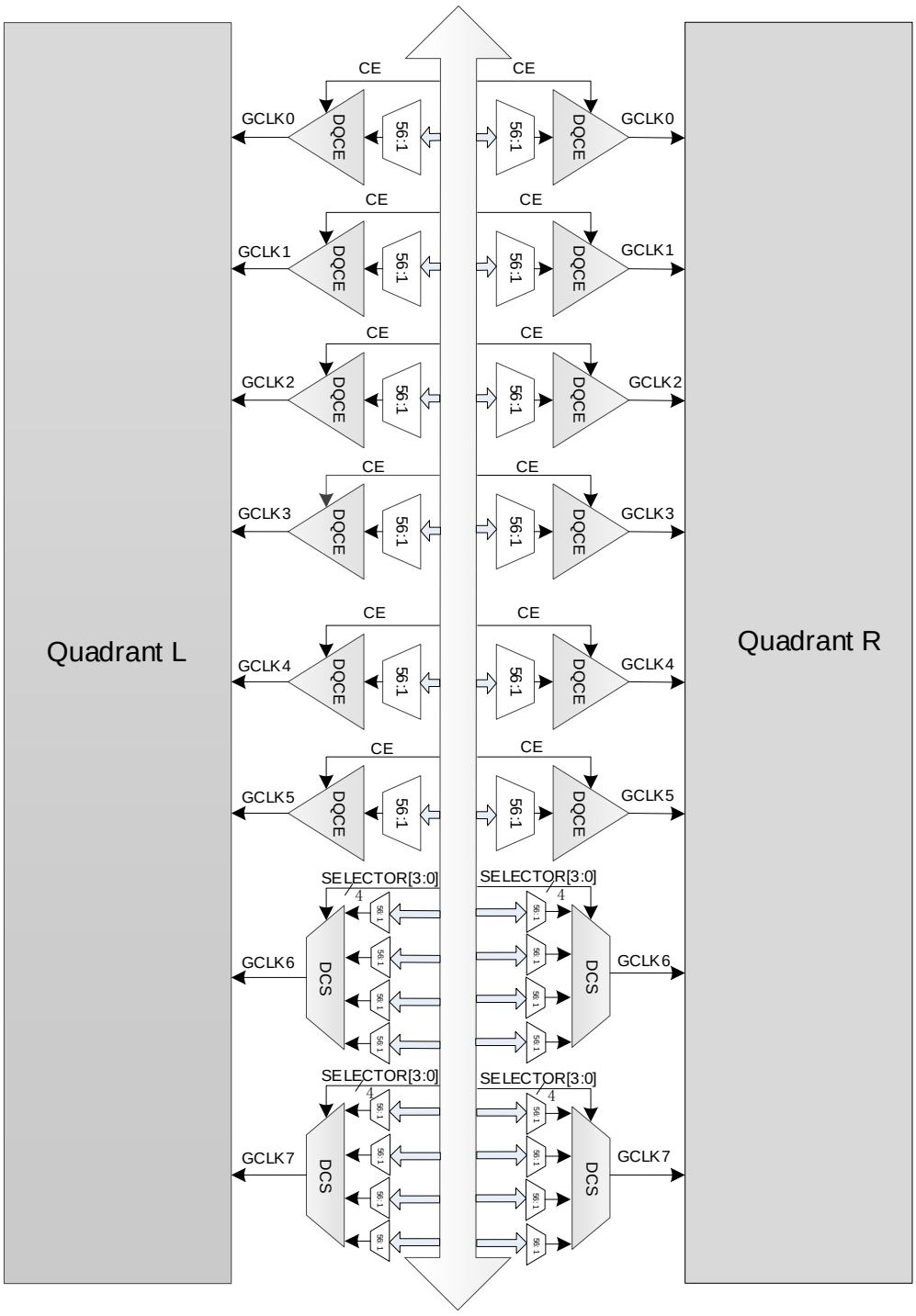
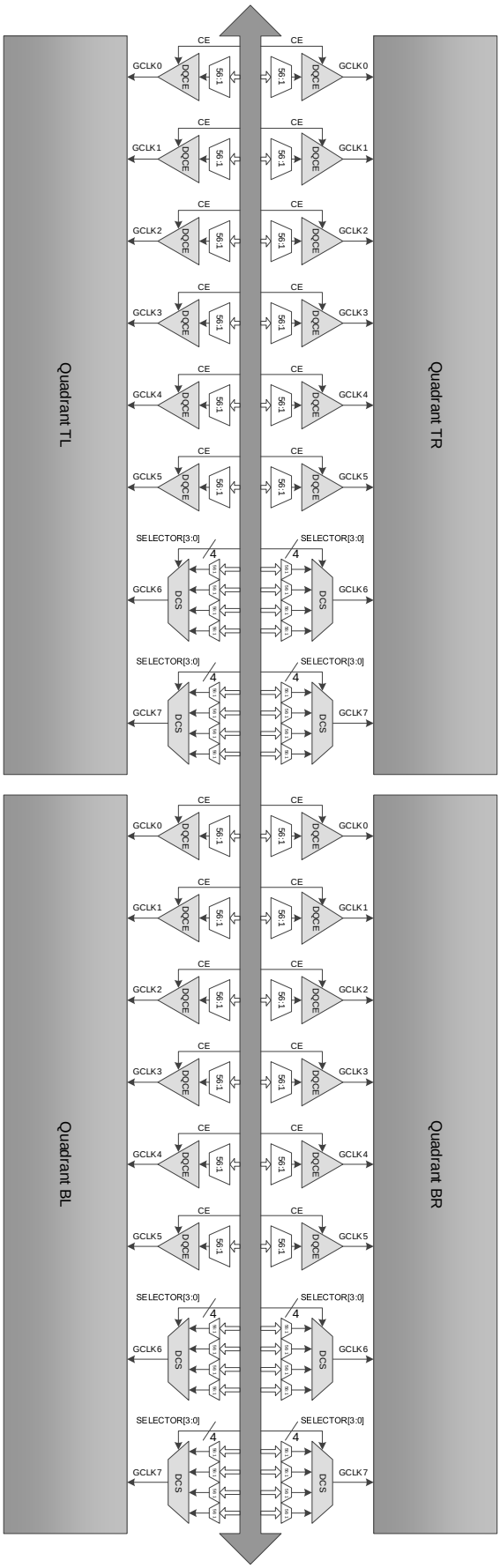


**Figure 2-2 Quadrant Distribution of GCLKs in the LittleBee® (9K) and Arora FPGA Family**



GCLK0~GCLK5 in each quadrant are dynamically controlled by DQCE to turn on/off. When GCLK0~GCLK5 in the quadrant are off, all the logic driven by them will not toggle; therefore, lower power consumption can be achieved.

GCLK6~GCLK7 in each quadrant are controlled by DCS. The internal logic can dynamically be selected within the four clock inputs through CRU and output a clock without glitches.

## 2.2 HCLK

HCLK is the high-speed clock in the GOWINSEMI FPGA products. It offers low jitter and low deviation, which can support high-speed data transfer. It is mainly suitable for source synchronous data transfer protocols. There is an HCLKMUX module in the middle of the high-speed clock HCLK. HCLKMUX can send the HCLK clock signal from bank to bank; as such, HCLK can be used flexibly.

For HCLK diagrams, please refer to the following related documents.

- DS100, GW1N series of FPGA Products Data Sheet
- DS117, GW1NR series of FPGA Products Data Sheet
- DS821, GW1NS series of FPGA Products Data Sheet
- DS861, GW1NSR series of FPGA Products Data Sheet
- DS871, GW1NSE series of FPGA Products Data Sheet
- DS841, GW1NZ series of FPGA Products Data Sheet
- DS102, GW2A series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet

## 2.3 PLL

Phase-locked Loop (PLL) is a feedback control circuit. The frequency and phase of the internal oscillator signal are controlled by the external input reference clock.

PLL blocks in the GOWINSEMI FPGA products provide the ability to synthesize clock frequencies. Frequency adjustment (multiplication and division), phase shift, and duty cycle adjustment can be realized by parameters configuration.

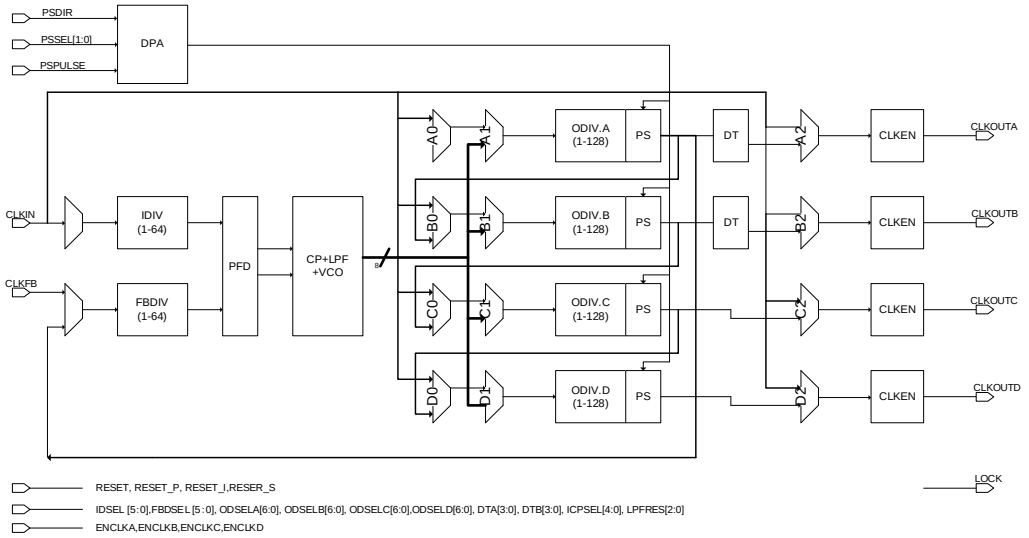
### 2.3.1 GW1N-1P5/GW1N-2/GW1NR-2/GW2AN-18X/GW2AN-9X

**Note!**

The description in this section only applies to the devices of GW1N-1P5, GW1N-2, GW1NR-2, GW2AN-18X, and GW2AN-9X.

See Figure 2-3 for the PLL structure of GW1N-1P5, GW1N-2, GW1NR-2, GW2AN-18X, and GW2AN-9X.

**Figure 2-3 PLL Structure**




See Table 2-1 for definitions of the PLL ports.

**Table 2-1 PLL Ports Definition**

| Port Name | Signal | Description |
|---|---|---|
| CLKIN | I | Reference clock input |
| CLKFB | I | Feedback clock input |
| RESET | I | PLL reset |
| RESET_P | I | PLL Power Down |
| RESET_I | I | PLL with IDIV reset |
| RESET_S | I | Only Channel B/C/D reset |
| IDSEL [5:0] | I | Dynamic IDIV control: 1~64 |
| FBDSEL [5:0] | I | Dynamic FBDIV control:1~64 |
| ODSELA[6:0] | I | Dynamic ODIVA control:1~128 |
| ODSELB[6:0] | I | Dynamic ODIVB control:1~128 |
| ODSELC[6:0] | I | Dynamic ODIVC control:1~128 |
| ODSELD[6:0] | I | Dynamic ODIVD control:1~128 |
| DTA[3:0] | I | Dynamic control of CLKOUTA dutycycle |
| DTB[3:0] | I | Dynamic control of CLKOUTB dutycycle |
| ICPSEL[4:0] | I | Dynamic control of ICP size |
| LPFRES[2:0] | I | Dynamic control LPFRES size |
| PSDIR | I | Dynamic control of phase shift direction |
| PSSEL[1:0] | I | Dynamic control of phase shift channel selection |
| PSPULSE | I | Dynamic control of phase shift clock |
| ENCLKA<br>ENCLKB<br>ENCLKC<br>ENCLKD | O | Dynamic control of clock output enable |
| CLKOUTA | O | Clock output of Channel A (by default) |
| CLKOUTB | O | Clock output of Channel B (by default) |

| Port Name | Signal | Description |
|---|---|---|
| CLKOUTC | O | Clock output of Channel C (by default) |
| CLKOUTD | O | Clock output of Channel D (by default) |
| LOCK | O | PLL lock status:<br>1: locked,<br>0: unlocked |

The PLL reference clock source can come from an external PLL pin or from internal routing GCLK, HCLK, or general data signal. PLL feedback signal can come from the external PLL feedback input or from internal routing GCLK, HCLK, or general data signal.

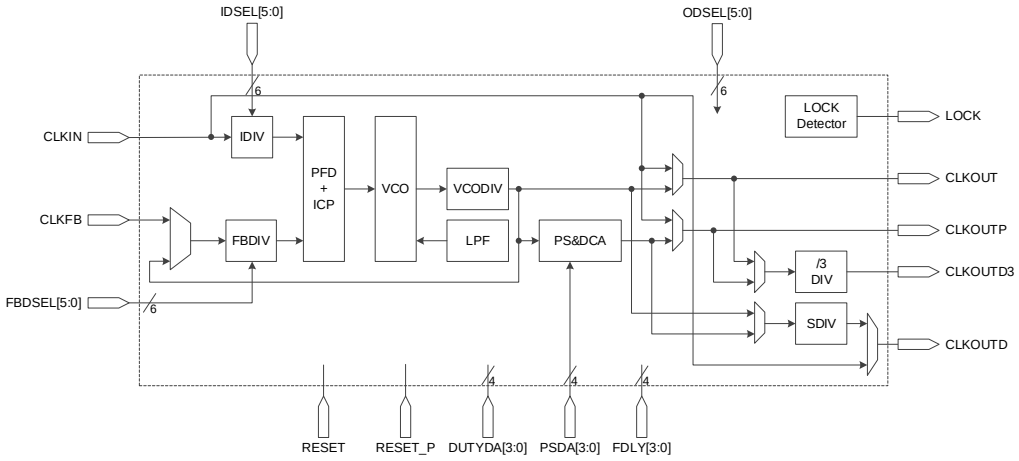
## 2.3.2 Other Devices in LittleBee® family and Arora family

**Note!**

The description in this section applies to all devices in LittleBee® family and Arora family except GW1N-1P5, GW1N-2, GW1NR-2, GW2AN-18X, and GW2AN-9X.

See Figure 2-4 for the PLL structure.

**Figure 2-4 PLL Structure**




See Table 2-2 for a definition of the PLL ports.

**Table 2-2 PLL Ports Definition**

| Port Name | Signal | Description |
|---|---|---|
| CLKIN | I | Reference clock input |
| CLKFB | I | Feedback clock input |
| RESET | I | PLL reset |
| RESET_P | I | PLL Power Down |
| IDSEL [5:0] | I | Dynamic IDIV control: 1~64 |
| FBDSEL [5:0] | I | Dynamic FBDIV control:1~64 |
| PSDA [3:0] | I | Dynamic phase control (rising edge effective) |
| DUTYDA [3:0] | I | Dynamic duty cycle control (falling edge effective) |
| FDLY[3:0] | I | CLKOUTP dynamic delay control |
| CLKOUT | O | Clock output with no phase and duty cycle |

| Port Name | Signal | Description |
|---|---|---|
| | | adjustment |
| CLKOUTP | O | Clock output with phase and duty cycle adjustment |
| CLKOUTD | O | Clock divider from CLKOUT and CLKOUTP (controlled by SDIV) |
| CLKOUTD3 | O | clock divider from CLKOUT and CLKOUTP (controlled by DIV3 with the constant division value 3) |
| LOCK | O | PLL lock status:<br>1: locked,<br>0: unlocked |

The PLL reference clock source can come from an external PLL pin or from internal routing GCLK, HCLK, or general data signal. PLL feedback signal can come from the external PLL feedback input or from internal routing GCLK, HCLK, or general data signal.

# 3 Global Clock

## 3.1 DQCE

### 3.1.1 Primitive Introduction

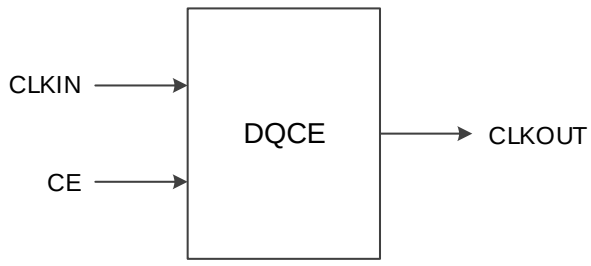
Gowin FPGA devices feature dynamic clock control that allows the internal logic to enable or disable the PCLK network in the quadrant. In addition, the DQCE dynamic clock control can be disabled to always enable the PCLK network. When the PCLK clock network is disabled, all logic driven by this clock is no longer toggled, thus reducing the total power of the device.

#### Functional Description

GCLK0~GCLK5 can be dynamically turned on/off through DQCE. When GCLK0~GCLK5 in the quadrant is off, all the internal logic driven by them will not toggle; therefore, lower power consumption can be achieved. Normal operation of the DQCE requires the CLKIN signal to have at least one falling edge change.

#### Port Diagram

**Figure 3-1 DQCE Diagram**




#### Port Description

**Table 3-1 DQCE Port Description**

| Port | I/O | Description |
|---|---|---|
| CLKIN | Input | Clock input signal |
| CE | Input | Clock enable signal, active-high. |
| CLKOUT | Output | Clock output signal |

#### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
DQCE dqce_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

**Vhdl Instantiation:**

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

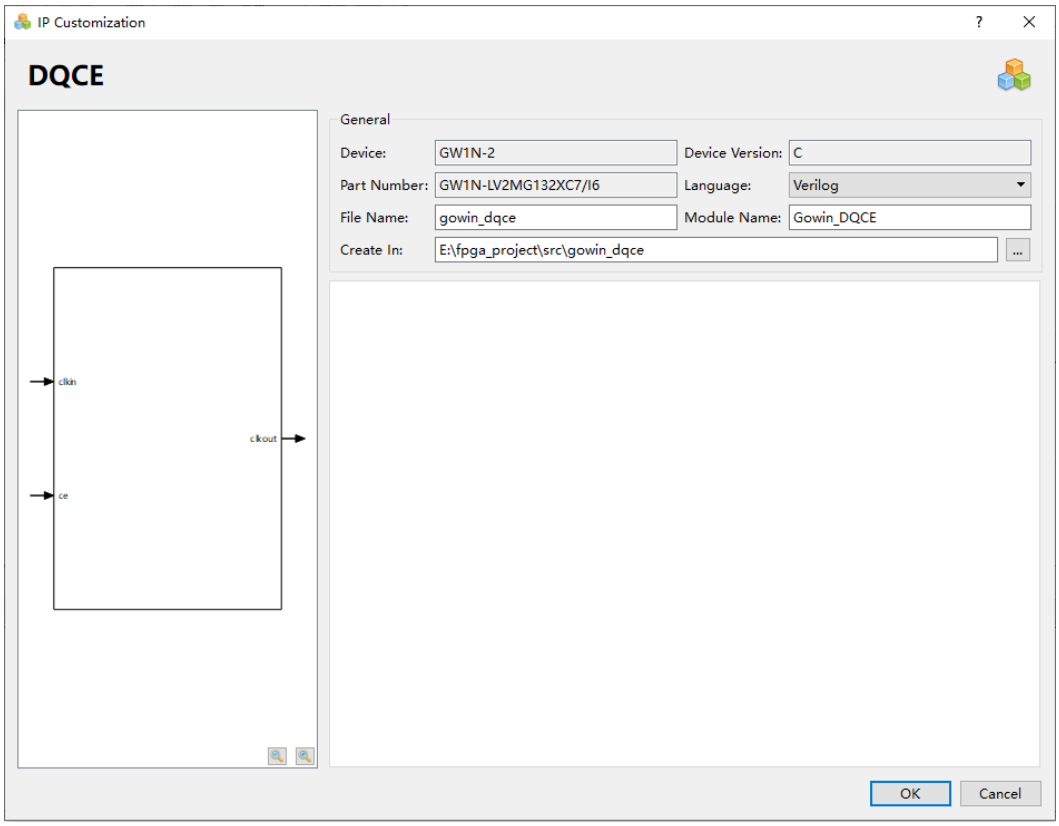
### 3.1.2 IP Generation

Click "DQCE" on the IP Core Generator and an overview of DQCE will be displayed on the right side of the interface.

#### IP Configuration

Double-click on "DQCE", and the "IP Customization" window pops up. It includes the "General", and port diagram, as shown in Figure 3-2.

**Figure 3-2 IP Customization of DQCE**




1. General
   The General configuration box is used to configure the generated IP design files.
   - Device: Select a device.
   - Device Version: Select a device version
   - Part Number: Select a part number
   - Language: Hardware description language used to generate the IP design files. Click the drop-down list to select the language, including Verilog and VHDL.
   - Module Name: The module name of the generated IP design files. Enter the module name in the text box. Module name cannot be the same as the primitive name. If it is the same, an error will be reported.
   - File Name: The name of the generated IP design files. Enter the file name in the text box.
   - Create In: The path in which the generated IP files will be stored. Enter the target path in the box or select the target path by clicking the right textbox.
2. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 3-2.

#### Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_dqce.v" file is a complete Verilog module to generate instantiated DQCE, and it is generated according to the IP configuration;
- "gowin_dqce_tmp.v" is the template file;
- "gowin_dqce.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 3.2 DCS

## 3.2.1 Primitive Introduction

There are two DCSs in each quadrant, corresponding to GCLK6 and GCLK7. The output of DCS is connected to GCLK6 or GCLK7, i.e., among eight GCLKs in a quadrant, GCLK6 and GCLK7 have dynamic clock selection (DCS) function. DCS clock selection signal, CLKSEL, comes from CIU, and the internal logic allows CLKOUT to dynamically switch among the four clock inputs via the CRU.

#### Functional Description

GCLK6~GCLK7 of each quadrant is controlled by the dynamic clock selector (DCS); select one of the four clocks as the global clock. The internal logic can be switched dynamically within the four clocks by CRU, and output a glitch-free clock.

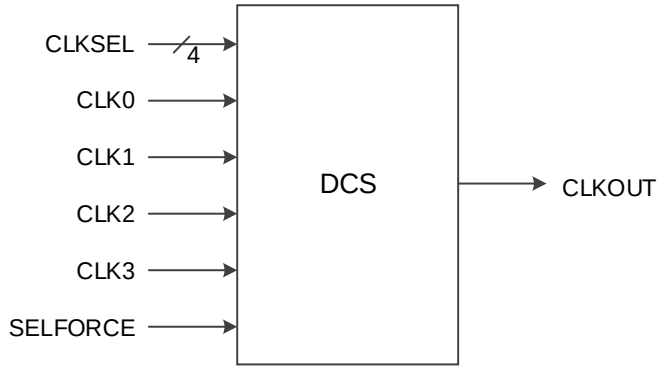
DCS has two clock switching modes, namely "Non-glitchless" and "Glitchless".

In Non-Glitchless mode, DCS acts like a multiplexer, switching clock signals only through CLKSEL signals, allowing glitch on the output, depending on the time of switching.

In Glitchless mode, it is possible to avoid glitch on the output clock that you can configure the CLKSEL signal to dynamically switch the clock signal by DCS_MODE.

## Port Diagram

**Figure 3-3 DCS Port Diagram**




## Port Description

**Table 3-2 DCS Port Description**

| Port | I/O | Description |
|---|---|---|
| CLK0 | Input | Clock input signal 0 |
| CLK1 | Input | Clock input signal 1 |
| CLK2 | Input | Clock Iinput signal 2 |
| CLK3 | Input | Clock input signal 3 |
| CLKSEL[3:0] | Input | Clock selection signal |
| SELFORCE | Input | Mandatory mode selection<br>0: Glitchless mode<br>1; Non-glitchless mode |
| CLKOUT | Output | Clock output signal |

## Parameter Description

**Table 3-3 DCS Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| 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" | Sets DCS mode |

## Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
DCS dcs_inst (
```

```
        .CLK0(clk0),
        .CLK1(clk1),
        .CLK2(clk2),
        .CLK3(clk3),
        .CLKSEL(clksel[3:0]),
        .SELFORCE(selforce),
        .CLKOUT(clkout)
    );
    defparam dcs_inst.DCS_MODE="RISING";
```

**Vhdl Instantiation:**

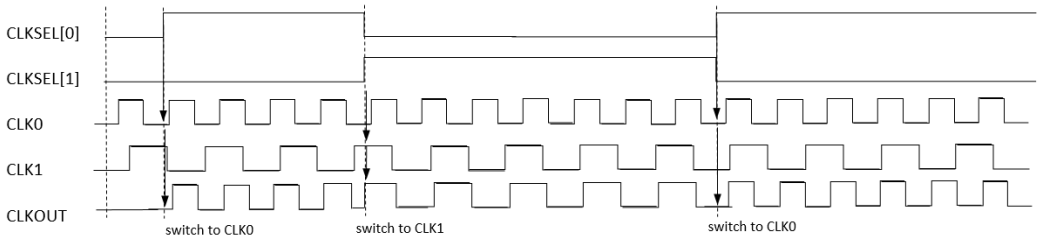
```
    COMPONENT DCS
            GENERIC(DCS_MODE:string:="RISING");
                PORT(
                        CLK0 : IN std_logic;
                        CLK1:IN std_logic;
                        CLK2:IN std_logic;
                        CLK3:IN std_logic;
                        CLKSEL:IN std_logic_vector(3 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
    );
```

### Timing Diagrams

The timing of Non-Glitchless mode is shown in Figure 3-4. CLKSEL

[3]~CLKSEL [0] are corresponding to CLK3~CLK0 with the same converting timing, active-high.

**Figure 3-4 Timing Diagram of Non-Glitchless**




The timing of Glitchless mode is shown in Figure 3-5~Figure 3-8 CLKSEL [3]~CLKSEL [0] are corresponding to CLK3~CLK0 with the same converting timing.

**Figure 3-5 RISING Timing Diagram in DCS Mode**

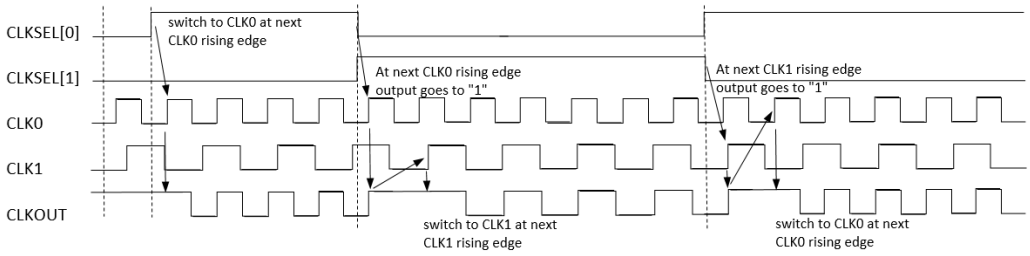



**Figure 3-6 FALLING Timing Diagram in DCS Mode**

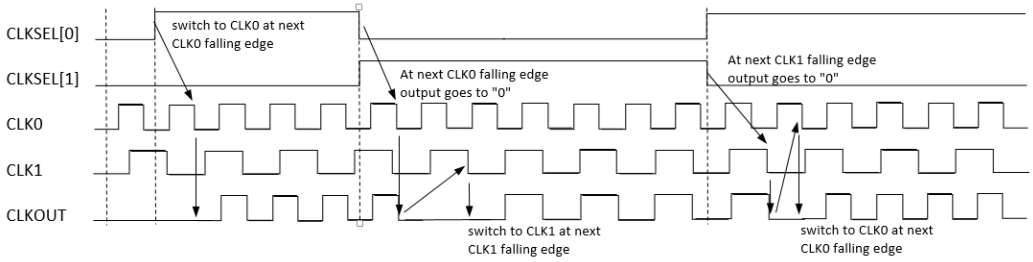



**Figure 3-7 CLK0_GND Timing Diagram in DCS Mode**

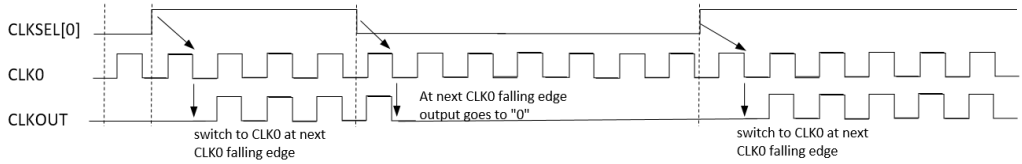
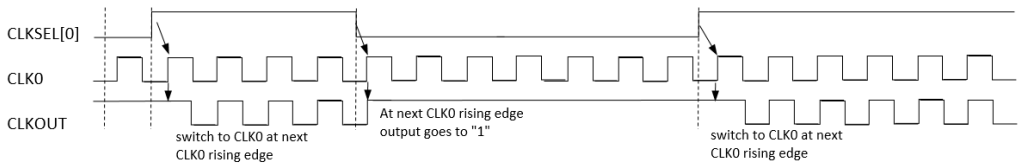



**Figure 3-8 CLK0_VCC Timing Diagram in DCS Mode**



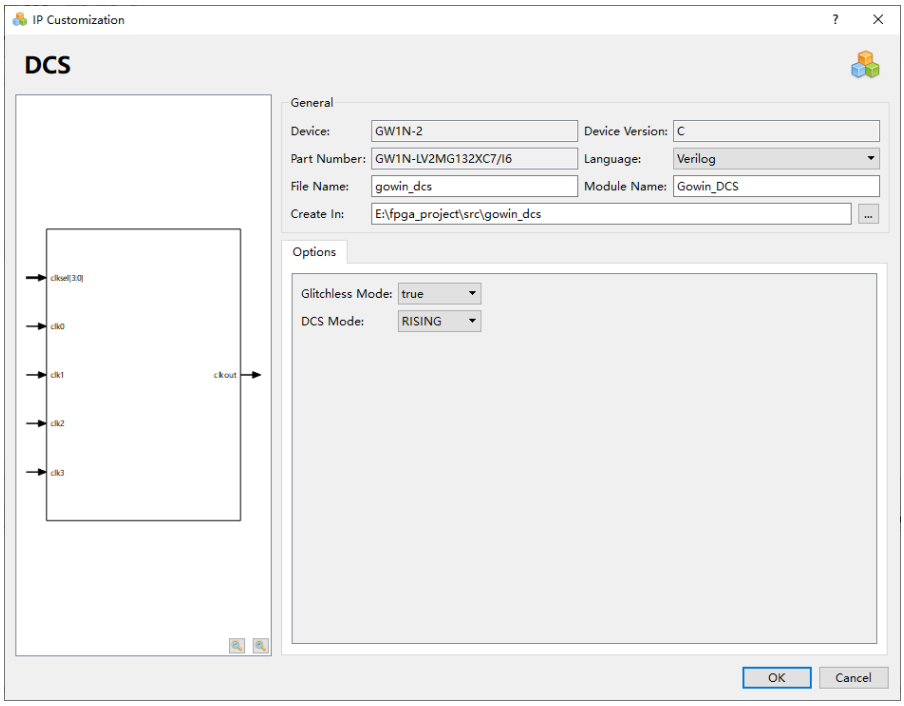
## 3.2.2 IP Generation

Click "DCS" on the "IP Core Generator" and an overview of DCS will be displayed on the right side of the interface.

### IP Configuration

Double-click on "DCS", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 3-9.

**Figure 3-9 IP Customization of DCS**




1. General
   The General configuration box is used to configure the generated IP design file.The DCS General configuration box is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options Configuration box is used to configure IP, as shown in Figure 3-9.
   - Glitchless Mode: Enables/ disables Glitchless
   - DCS Mode: Sets DCS mode
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 3-9.

### Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_dcs.v" file is a complete Verilog module to generate instantiated DCS, and it is generated according to the IP configuration;
- "gowin_dcs_tmp.v" is the template file;
- "gowin_dcs.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 4 HCLK

## 4.1 DHCEN

### 4.1.1 Primitive Introduction

You can use DHCEN to dynamically turn on/off HCLK, and DHCEN is conductive when CE is low.

#### Port Diagram

**Figure 4-1 DHCEN Port Diagram**


#### Port Description

**Table 4-1 DHCEN Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKIN | input | Clock input signal |
| CE | input | Clock enable input signal, active-low. |
| CLKOUT | output | Clock output signal |

#### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
DHCEN dhcen_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

**Vhdl Instantiation:**

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

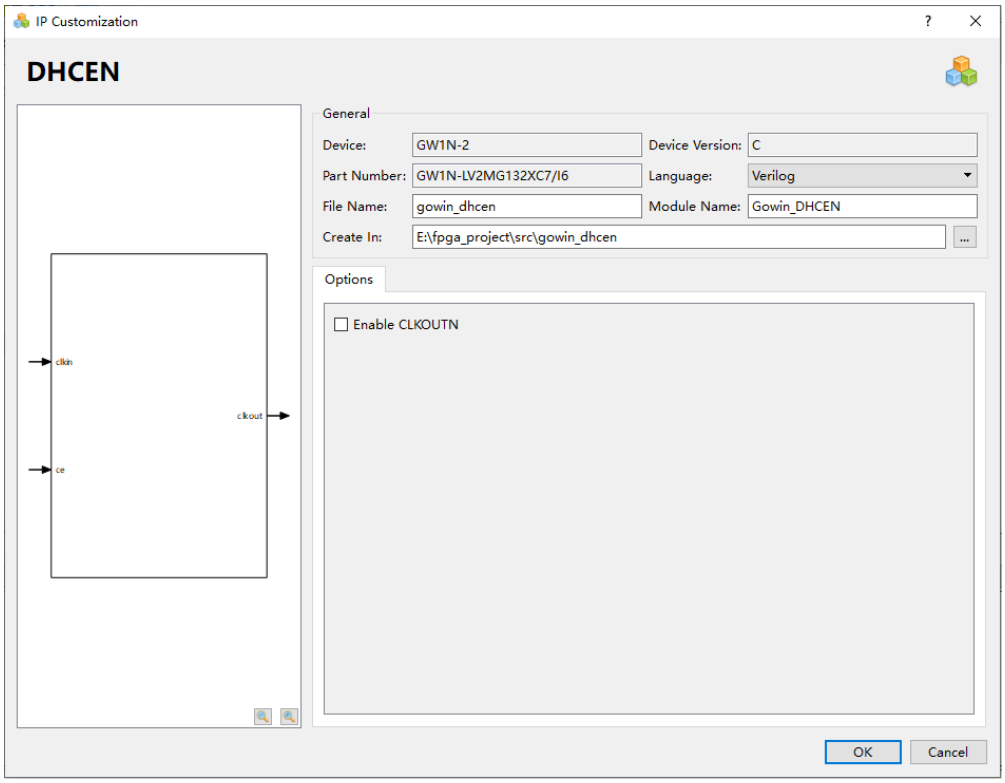
### 4.1.2 IP Generation

Click "DHCEN" on the "IP Core Generator" interface and an overview of related information about DHCEN will be displayed on the right side of the interface.

#### IP Configuration

Double-click on "DHCEN", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 4-2.

**Figure 4-2 IP Customization of DHCEN**




1. General
   The General configuration box is used to configure the generated IP design file. DHCEN General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to configure IP, as shown in Figure 4-2.Enable CLKOUTN: Instantiates DHCENC when enabled, and instantiates DHCEN when disabled.
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 4-2.

### Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_dhcen.v" file is a complete Verilog module to generate instantiated DHCEN, and it is generated according to the IP configuration;
- "gowin_dhcen_tmp.v" is the template file;
- "gowin_dhcen.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 4.2 DHCENC

## 4.2.1 Primitive Introduction

You can use DHCENC to dynamically turn on/off HCLK, and DHCENC is conductive when CE is low.

### Device Supported

**Table 4-2 DHCENC Device Supported**

| Product Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
| | GW1NR | GW1NR-9C, GW1NR-2, GW1NR-2B |

### Port Diagram

**Figure 4-3 DHCEN Port Diagram**


### Port Description

**Table 4-3 DHCENC Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKIN | input | Clock input signal |
| CE | input | Clock enable signal, active-low. |
| CLKOUT | output | Clock output signal |
| CLKOUTN | output | Clock output signal, CLKOUTN negated. |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
DHCENC dhcenc_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout),
    .CLKOUTN(clkoutn)
);
```

**Vhdl Instantiation:**

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

### 4.2.2 IP Generation

The IP interface and calling method of DHCENC and DHCEN are the same, you can see 4.1.2 IP Generation.

## 4.3 DCC

### 4.3.1 Primitive Introduction

HCLK duty cycle correction module (DCC)

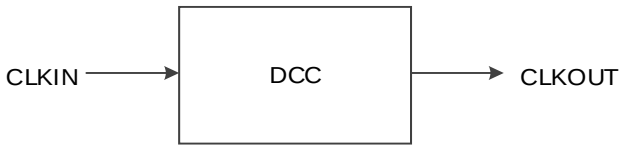
**Device Supported**

**Table 4-4 DCC Device Supported**

| Product Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-9C |
| | GW1NR | GW1NR-9C |

**Port Diagram**

**Figure 4-4 DCC Port Diagram**



### Port Description

**Table 4-5 DCC Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKIN | input | Clock input signal |
| CLKOUT | output | Clock output signal |

### Parameter Description

**Table 4-6 DCC Parameter Description**

| Parameter | Value | Default | Description |
|---|---|---|---|
| DCC_EN | 1'b1, 1'b0 | 1'b1 | 1'b1: Enables DCC<br>1'b0: Disables DCC |
| FCLKIN | – | 50.0 | Input clock frequency |

### Primitive Instantiation

**Verilog Instantiation:**

```
DCC dcc_inst (
      .CLKIN(clkin),
      .CLKOUT(clkout)
);
defparam dcc_inst.DCC_EN=1'b1;
defparam dcc_inst.FCLKIN=50.0;
```

**VHDL Instantiation：**

```
COMPONENT DCC
        GENERIC (
            DCC_EN : bit := '1';    --'1':enable dcc; '0': disable dcc
            FCLKIN : REAL := 50.0 --frequency of the clkin(M)
        );
         PORT(
              CLKOUT:OUT std_logic;
              CLKIN:IN std_logic
        );
END COMPONENT;
uut:DCC
GENERIC MAP(
                    DCC_EN=>'1',
                    FCLKIN=>50.0
```

```
                    )
    PORT MAP(
        CLKIN=>clkin,
        CLKOUT=>clkout
            );
```

# 4.4 DCCG

## 4.4.1 Primitive Introduction

HCLK duty cycle correction module (DCCG)

### Device Supported

**Table 4-7 DCCG Device Supported**

| Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
| | GW1NR | GW1NR-2, GW1NR-2B |

### Port Diagram

**Figure 4-5 DCCG Port Diagram**


### Port Description

**Table 4-8 DCCG Port Description**

| Port | I/O | Description |
|---|---|---|
| CLKIN | input | Clock input signal |
| CLKOUT | output | Clock output signal |

### Parameter Description

**Table 4-9 DCCG Parameter Description**

| Parameter | Value | Default | Description |
|---|---|---|---|
| DCC_MODE | 2'b00, 2'b01,<br>2'b10, 2'b11 | 2'b00 | 2'b00/2'b01:Buffered<br>2'b10: +80ps<br>2'b11: -80ps |
| FCLKIN | – | 50.0 | Input clock frequency |

### Primitive Instantiation

#### Verilog Instantiation:

```
DCCG dccg_inst (
    .CLKIN(clkin),
    .CLKOUT(clkout)
);
defparam dccg_inst.DCC_MODE=2'b00;
defparam dccg_inst.FCLKIN=50.0;
```

**VHDL Instantiation：**

```
COMPONENT DCCG
    GENERIC (
        DCC_MODE : bit_vector := "00";
        FCLKIN : REAL := 50.0 --frequency of the clkin(M)
    );
    PORT(
        CLKOUT:OUT std_logic;
        CLKIN:IN std_logic
    );
END COMPONENT;
uut:DCCG
GENERIC MAP(
                DCC_MODE=>"00",
                FCLKIN=>50.0
                )
PORT MAP(
    CLKIN=>clkin,
    CLKOUT=>clkout
    );
```

# 4.5 CLKDIV2

## 4.5.1 Primitive Introduction

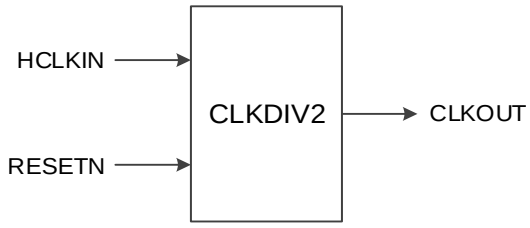
The CLKDIV2 is a clock divider. It realizes a divide-by-two clock. The output of CLKDIV2 can only drive CLKIN of DCC/DCCG, FCLK of IOLOGIC, CLKIN and CLKFB of PLL, FCLK of DQS, and HCLKIN of CLKDIV.

### Functional Description

CLKDIV2 is a HCLK divider module that generates a divide-by-two clock with the same phase as the input clock.

## Port Diagram

**Figure 4-6 CLKDIV2 Port Diagram**




## Port Description

**Table 4-10 CLKDIV2 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| HCLKIN | Input | Clock input signal |
| RESETN | Input | Asynchronous reset signal, active-low. |
| CLKOUT | Output | Clock output signal |

## Parameter Description

**Table 4-11 CLKDIV2 Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| GSREN | "false", "true" | "false" | Enable global reset |

## Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
CLKDIV2 clkdiv2_inst(
    .HCLKIN(hclkin),
    .RESETN(resetn),
    .CLKOUT(clkout)
);
defparam clkdiv2_inst.GSREN="false";
```

**Vhdl Instantiation:**

```
COMPONENT CLKDIV2
    GENERIC(
        GSREN:STRING:="false"
        );
    PORT(
```

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

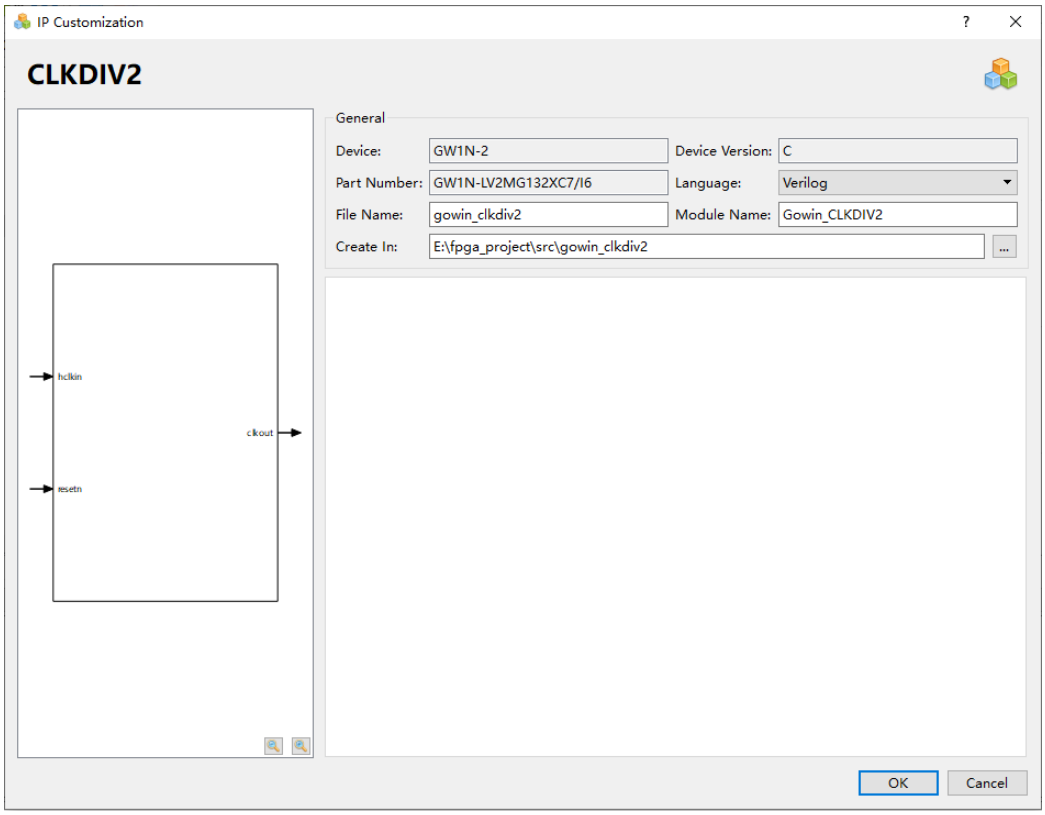
## 4.5.2 IP Generation

Click "CLKDIV2" on the "IP Core Generator" interface and an overview of CLKDIV2 will be displayed on the right side of the interface.

### IP Configuration

Double-click on "CLKDIV2", and the "IP Customization" window pops up. It includes the "General", and port diagram, as shown in Figure 4-7.

**Figure 4-7 IP Customization of CLKDIV2**




1. General
   The Genera configuration box is used to configure the generated IP design file. The CLKDIV2 General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 4-7.

### IP Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_clkdiv2.v" file is a complete Verilog module to generate instantiated CLKDIV2, and it is generated according to the IP configuration;
- "gowin_clkdiv2_tmp.v" is the template file;
- "gowin_clkdiv2.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5 System Clock

## 5.1 rPLL

### 5.1.1 Primitive Introduction

Gowin FPGA provides rPLLs to control the frequency and phase of the oscillation signals in the loop via external input reference clock signals.

#### Device Supported

**Table 5-1 rPLL Device Supported**

| Product Family | Series | Device |
|---|---|---|
| Arora | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AN | GW2AN-55C |
| | GW2AR | GW2AR-18,GW2AR-18C |
| | GW2ANR | GW2ANR-18C |
| LittleBee® | GW1N | GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1N-4D, GW1N-9, GW1N-9C |
| | GW1NR | GW1NR-1,GW1NR-4,GW1NR-4B,GW1NR-4D, GW1NR-9,GW1NR-9C |
| | GW1NRF | GW1NRF-4B |
| | GW1NS | GW1NS-2,GW1NS-2C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSR | GW1NSR-2,GW1NSR-2C |
| | GW1NZ | GW1NZ-1, GW1NZ-1C |

#### Functional Description

Based on the given input clock, rPLL adjusts clock phase, duty cycle, frequency (multiplication and division) to generate output clocks with different phases and frequencies.

rPLL can adjust the frequency of the input clock CLKIN (multiplication and division). The formulas are as follows:

$$f_{\mathrm{CLKOUT}} = (f_{CLKIN} * \mathrm{FBDIV})/\mathrm{IDIV}$$

$$f_{VCO} = f_{CLKOUT} * ODIV$$

$$f_{CLKOUTD} = f_{CLKOUT}/SDIV$$

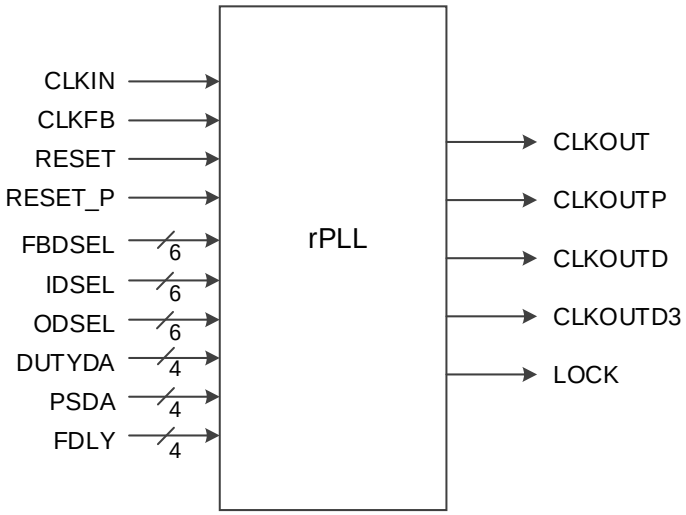
$$f_{PFD} = f_{CLKIN}/IDIV = f_{CLKOUT}/FBDIV$$

**Note!**

- $f_{CLKIN}$ is the input clock CLKIN frequency; $f_{CLKOUT}$ is the CLKOUT and CLKOUTP frequency; $f_{CLKOUTD}$ is the CLKOUTD frequency, and $f_{PFD}$ is the PFD phase detection frequency.
- IDIV, FBDIV, ODIV and SDIV are the frequency division coefficients of different frequency dividers, which can be adjusted to get the clock signal with expected frequency.
- For the frequency range of the rPLL, please refer to the *FPGA Product Datasheet*.

## Port Diagram

**Figure 5-1 rPLL Port Diagram**




## Port Description

**Table 5-2 rPLL Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKIN | Input | Reference clock input signal |
| CLKFB | Input | Feedback clock input signal |
| RESET | Input | rPLL asynchronous reset input signal, active-high. |
| RESET_P | Input | rPLL power down input signal, active-high. The output of CLKOUT/CLKOUTP/CLKOUTD/CLKOUTD3 is 0 when PLL is not in bypass mode and RESET_P is in high level. |
| FBDSEL[5:0] | Input | Dynamically controls FBDIV value, ranging from 0 to 63, and the actual value is 64-FBDSEL. |
| IDSEL[5:0] | Input | Dynamically controls IDIV value, ranging from 0 to 63, and the actual value is 64-IDSEL. |
| ODSEL[5:0] | Input | Dynamically controls ODIV value: 2,4,8,16,32,48,64,80,96,112,128 |
| DUTYDA[3:0] | Input | Duty cycle dynamic adjustment signal |
| PSDA[3:0] | Input | Phase dynamic adjustment signal |
| FDLY[3:0] | Input | Fine delay dynamic adjustment signal |

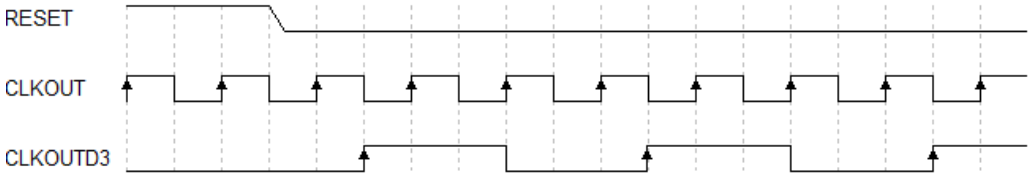
| Port Name | I/O | Description |
|---|---|---|
| CLKOUT | Output | rPLL clock output signal |
| LOCK | Output | rPLL locked indicator.1: locked, 0: unlocked. |
| CLKOUTP | Output | rPLL clock output signal with phase and duty cycle adjustment |
| CLKOUTD | Output | Clock output signal of rPLL through SDIV, output signal of CLKOUT or CLKOUTP through SDIV divider output signal. |
| CLKOUTD3 | Output | Clock output signal of rPLL through DIV3, output signal of CLKOUT or CLKOUTP through DIV3. |

CLKOUTD3 is a divide-by-three output clock signal with two input sources.

- If the input source is CLKOUT:

As shown in Figure 5-2, after the RESET is released, CLKOUTD3 goes high on the first falling edge of the CLKOUT and then goes low on the subsequent second rising edge.

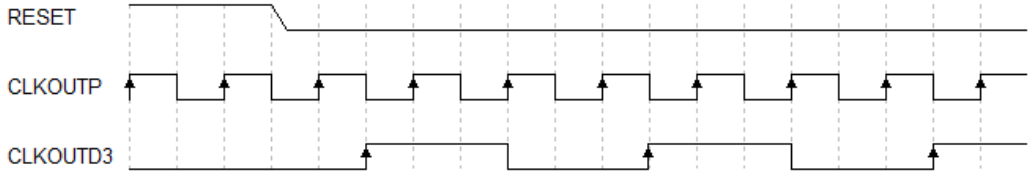
**Figure 5-2 CLKOUTD3 Tming Diagram with CLKOUT Input Source**




- If the input source is CLKOUTP:

As shown in Figure 5-3, after RESET reset is released, CLKOUTD3 goes high on the first falling edge of CLKOUTP and then goes low on the subsequent second rising edge.

**Figure 5-3 CLKOUTD3 Tming Diagram with CLKOUTP Input Source**



## Parameter Description

**Table 5-3 rPLL Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FCLKIN | "3"~ "500" | "100" | Reference clock frequency |
| IDIV_SEL | 0~63 | 0 | IDIV frequency division coefficient static setting |
| DYN_IDIV_SEL | "true", "false" | "false" | IDIV frequency division coefficient static control parameter or dynamic control signal selection.<br>● false: Static, that is, select the parameter IDIV_SEL.<br>● true: Dynamic, namely select signal IDSEL. |
| FBDIV_SEL | 0~63 | 0 | FBDIV frequency division coefficient static setting |
| DYN_FBDIV_SEL | "true", "false" | "false" | FBDIV frequency division coefficient static control parameter or dynamic control signal selection.<br>● false: Static, that is, select the parameter FBDIV_SEL.<br>● true: Dynamic, namely select signal FBDSEL. |
| ODIV_SEL | 2,4,8,16,32,48,64,80,96,112,128 | 8 | ODIV frequency division coefficient static setting |
| DYN_ODIV_SEL | "true", "false" | "false" | ODIV frequency division coefficient static control parameter or dynamic control signal selection.<br>● false: Static, that is, select the parameter ODIV_SEL.<br>● true: Dynamic, namely select signal ODSEL. |
| PSDA_SEL | "0000"~"1111" | "0000" | Phase static adjustment |
| DUTYDA_SEL | "0010"~"1110" | "1000" | Duty cycle static adjustment |
| DYN_DA_EN | "true", "false" | "false" | The dynamic signal is selected as the control of phase and duty cycle adjustment.<br>● false: Static control<br>● true: Dynamic control |
| CLKOUT_FT_DIR | 1'b1 | 1'b1 | CLKOUT trim direction setting<br>1'b1: Decrease |
| CLKOUT_DLY_STEP | 0,1,2,4 | 0 | CLKOUT trim coefficient |

| Name | Value | Default | Description |
|---|---|---|---|
| | | | setting<br>CLKOUT_DLY_STEP*delay(delay=50ps) |
| CLKOUTP_FT_DIR | 1'b1 | 1'b1 | CLKOUTP trim direction setting<br>1'b1: Decrease |
| CLKOUTP_DLY_STEP | 0,1,2 | 0 | CLKOUTP trim coefficient setting<br>CLKOUTP_DLY_STEP*dela y(delay=50ps) |
| DYN_SDIV_SEL | 2~128 ( Even ) | 2 | SDIV frequency division coefficient static setting |
| CLKFB_SEL | "internal", "external" | "internal" | CLKFB source selection<br>● internal: Feedback from internal CLKOUT.<br>● external: Feedback from external signal. |
| CLKOUTD_SRC | "CLKOUT", "CLKOUTP" | "CLKOUT" | CLKOUTD source selection |
| CLKOUTD3_SRC | "CLKOUT", "CLKOUTP" | "CLKOUT" | CLKOUTD3 source selection |
| CLKOUT_BYPASS | "True", "false" | "false" | Bypasses rPLL, and CLKOUT comes directly from CLKIN.<br>● true: CLKIN bypasses rPLL and acts directly on CLKOUT.<br>● false: Normal |
| CLKOUTP_BYPASS | "True", "false" | "false" | Bypasses rPLL, and CLKOUTP comes directly from CLKIN.<br>● true: CLKIN bypasses rPLL and acts directly on CLKOUTP<br>● false: Normal |
| CLKOUTD_BYPASS | "True", "false" | "false" | Bypasses rPLL, and CLKOUTD comes directly from CLKIN.<br>● true: CLKIN bypasses rPLL and acts directly on CLKOUTD.<br>● false: Normal |
| DEVICE | "GW1N-1", "GW1NR-1", "GW1N-1S", "GW1NZ-1", "GW1NZ-1C", "GW1N-4", "GW1N-4B", "GW1N-4D", | "GW1N-4" | Devices selected |

| Name | Value | Default | Description |
| --- | --- | --- | --- |
| | "GW1NR-4",<br>"GW1NR-4B",<br>"GW1NR-4D",<br>"GW1NRF-4B",<br>"GW1N-9",<br>"GW1N-9C",<br>"GW1NR-9",<br>"GW1NR-9C",<br>"GW2A-18",<br>"GW2AR-18",<br>"GW2A-55",<br>"GW2A-55C" | | |

**Table 5-4 IDSEL Port Parameter Comparison Table**

| IDSEL[5:0] | IDIV Static Parameter Value | IDIV Actual Value |
| --- | --- | --- |
| 111111 | 0 | 1 |
| 111110 | 1 | 2 |
| 111101 | 2 | 3 |
| 111100 | 3 | 4 |
| 111011 | 4 | 5 |
| 111010 | 5 | 6 |
| 111001 | 6 | 7 |
| 111000 | 7 | 8 |
| 110111 | 8 | 9 |
| …… | …… | …… |
| 000000 | 63 | 64 |

**Table 5-5 FBDSEL Port Parameter Comparison Table**

| FBDSEL [5:0] | FBDIV Static Parameter Value | FBDIV Actual Value |
| --- | --- | --- |
| 111111 | 0 | 1 |
| 111110 | 1 | 2 |
| 111101 | 2 | 3 |
| 111100 | 3 | 4 |
| 111011 | 4 | 5 |
| 111010 | 5 | 6 |
| 111001 | 6 | 7 |
| 111000 | 7 | 8 |
| 110111 | 8 | 9 |
| …… | …… | …… |
| 000000 | 63 | 64 |

**Table 5-6 ODSEL Port Parameter Comparison Table**

| ODSEL [5:0] | ODIV Parameter Value | ODIV Actual Value |
|---|---|---|
| 111111 | 2 | 2 |
| 111110 | 4 | 4 |
| 111100 | 8 | 8 |
| 111000 | 16 | 16 |
| 110000 | 32 | 32 |
| 101000 | 48 | 48 |
| 100000 | 64 | 64 |
| 011000 | 80 | 80 |
| 010000 | 96 | 96 |
| 001000 | 112 | 112 |
| 000000 | 128 | 128 |

**Table 5-7 rPLL Phase Parameter Adjustment**

| Parameter PSDA_SEL or Port PSDA Setting | Phase Shift |
|---|---|
| 0000 | 0° |
| 0001 | 22.5° |
| 0010 | 45° |
| 0011 | 67.5° |
| 0100 | 90° |
| 0101 | 112.5° |
| 0110 | 135° |
| 0111 | 157.5° |
| 1000 | 180° |
| 1001 | 202.5° |
| 1010 | 225° |
| 1011 | 247.5° |
| 1100 | 270° |
| 1101 | 292.5° |
| 1110 | 315° |
| 1111 | 337.5° |

**Table 5-8 rPLL Duty Cycle Parameter Adjustment**

| Parameters DUTYDA_SEL Setting | Duty Cycle Setting（/16） |
|---|---|
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |

| Parameters DUTYDA_SEL Setting | Duty Cycle Setting（/16） |
|---|---|
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | 10 |
| 1011 | 11 |
| 1100 | 12 |
| 1101 | 13 |
| 1110 | 14 |

You need to see phase shift setting for dynamic duty cycle adjustment. For example, when the phase shift setting is "0" (00000), the 50% duty cycle setting will be "8" (1000). If the phase shift setting is "180º", the 50% duty cycle setting is "0" (0000).

The calculation of dynamic duty cycle is as follows.

- If DUTYDA [3:0] > PSDA [3:0], DutyCycle=1/16 x (DUTYDA [3:0]-PSDA [3:0]).
- If DUTYDA [3:0] < PSDA [3:0], DutyCycle=1/16 x (16+ DUTYDA [3:0]-PSDA [3:0]).

**Note!**

The following three situations are not supported: DutyCycle = 0, 1, 15.

You can use FDLY [3: 0] to dynamically control CLKOUTP delay. Each step increases 0.125 ns, and you need to use the phase shift setting to achieve lagging[1] and leading[2].

**Note!**

- [1]CLKOUTP lags behind input signal.
- [2]CLKOUTP leads the input signal.

**Table 5-9 rPLL Ports Configuration**

| Port FDLY [3:0（GW1N-1/GW1N-1S） | Port FDLY [3:0] (other devices) | Delay Steps |
|---|---|---|
| 0000 | 1111 | 0 |
| 0001 | 1110 | 1 |
| 0010 | 1101 | 2 |
| 0100 | 1011 | 4 |
| 1000 | 0111 | 8 |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
rPLL rpll_inst(
        .CLKOUT(clkout),
        .LOCK(lock),
        .CLKOUTP(clkoutp),
        .CLKOUTD(clkoutd),
        .CLKOUTD3(clkoutd3),
        .RESET(reset),
        .RESET_P(reset_p),
        .CLKIN(clkin),
        .CLKFB(clkfb),
        .FBDSEL(fbdsel),
        .IDSEL(idsel),
        .ODSEL(odsel),
        .PSDA(psda),
        .DUTYDA(dutyda),
        .FDLY(fdly)
);
defparam rpll_inst.FCLKIN = "50";
defparam rpll_inst.DYN_IDIV_SEL = "false";
defparam rpll_inst.IDIV_SEL = 0;
defparam rpll_inst.DYN_FBDIV_SEL = "false";
defparam rpll_inst.FBDIV_SEL = 1;
defparam rpll_inst.ODIV_SEL = 8;
defparam rpll_inst.PSDA_SEL = "0100";
defparam rpll_inst.DYN_DA_EN = "false";
defparam rpll_inst.DUTYDA_SEL = "1000";
defparam rpll_inst.CLKOUT_FT_DIR = 1'b1;
defparam rpll_inst.CLKOUTP_FT_DIR = 1'b1;
defparam rpll_inst.CLKOUT_DLY_STEP = 0;
defparam rpll_inst.CLKOUTP_DLY_STEP = 0;
defparam rpll_inst.CLKFB_SEL ="external";
defparam rpll_inst.CLKOUT_BYPASS = "false";
defparam rpll_inst.CLKOUTP_BYPASS = "false";
defparam rpll_inst.CLKOUTD_BYPASS = "false";
```

```
defparam rpll_inst.DYN_SDIV_SEL = 2;
defparam rpll_inst.CLKOUTD_SRC = "CLKOUT";
defparam rpll_inst.CLKOUTD3_SRC = "CLKOUT";
defparam rpll_inst.DEVICE = "GW1N-4";
```

**Vhdl Instantiation:**

```
COMPONENT rPLL
    GENERIC(
            FCLKIN:STRING:= "100.0";
            DEVICE:STRING:= "GW1N-4";
            DYN_IDIV_SEL:STRING:="false";
            IDIV_SEL:integer:=0;
            DYN_FBDIV_SEL:STRING:="false";
            FBDIV_SEL:integer:=0;
            DYN_ODIV_SEL:STRING:="false";
            ODIV_SEL:integer:=8;
            PSDA_SEL:STRING:="0000";
            DYN_DA_EN:STRING:="false";
            DUTYDA_SEL:STRING:="1000";
            CLKOUT_FT_DIR:bit:='1';
            CLKOUTP_FT_DIR:bit:='1';
            CLKOUT_DLY_STEP:integer:=0;
            CLKOUTP_DLY_STEP:integer:=0;
            CLKOUTD3_SRC:STRING:="CLKOUT";
            CLKFB_SEL : STRING:="internal";
            CLKOUT_BYPASS:STRING:="false";
            CLKOUTP_BYPASS:STRING:="false";
            CLKOUTD_BYPASS:STRING:="false";
            CLKOUTD_SRC:STRING:="CLKOUT";
            DYN_SDIV_SEL:integer:=2
     );
        PORT(
            CLKIN:IN std_logic;
            CLKFB:IN std_logic;
            IDSEL:IN std_logic_vector(5 downto 0);
            FBDSEL:IN std_logic_vector(5 downto 0);
```

```
                ODSEL:IN std_logic_vector(5 downto 0);
                RESET:IN std_logic;
                RESET_P:IN std_logic;
                PSDA,FDLY:IN std_logic_vector(3 downto 0);
                DUTYDA:IN std_logic_vector(3 downto 0);
                LOCK:OUT std_logic;
                CLKOUT:OUT std_logic;
                CLKOUTD:OUT std_logic;
                CLKOUTP:OUT std_logic;
                CLKOUTD3:OUT std_logic
        );
END COMPONENT;
uut:rPLL
    GENERIC MAP(
                FCLKIN =>"100.0",
                DEVICE =>"GW2A-18",
                DYN_IDIV_SEL=>"false",
                IDIV_SEL=>0,
                DYN_FBDIV_SEL=>"false",
                FBDIV_SEL=>0,
                DYN_ODIV_SEL=>"false",
                ODIV_SEL=>8,
                PSDA_SEL=>"0000",
                DYN_DA_EN=>"false",
                DUTYDA_SEL=>"1000",
                CLKOUT_FT_DIR=>'1',
                CLKOUTP_FT_DIR=>'1',
                CLKOUT_DLY_STEP=>0,
                CLKOUTP_DLY_STEP=>0,
                CLKOUTD3_SRC=>"CLKOUT",
                CLKFB_SEL=>"internal",
                CLKOUT_BYPASS=>"false",
                CLKOUTP_BYPASS=>"false",
                CLKOUTD_BYPASS=>"false",
                CLKOUTD_SRC=>"CLKOUT",
```

```
                DYN_SDIV_SEL=>2
    )
PORT MAP (
    CLKIN=>clkin,
    CLKFB=>clkfb,
    IDSEL=>idsel,
    FBDSEL=>fbdsel,
    ODSEL=>odsel,
    RESET=>reset,
    RESET_P=>reset_p,
    PSDA=>psda,
    FDLY=>fdly,
    DUTYDA=>dutyda,
    LOCK=>lock,
    CLKOUT=>clkout,
    CLKOUTD=>clkoutd,
    CLKOUTP=>clkoutp ,
    CLKOUTD3=>clkoutd3
);
```

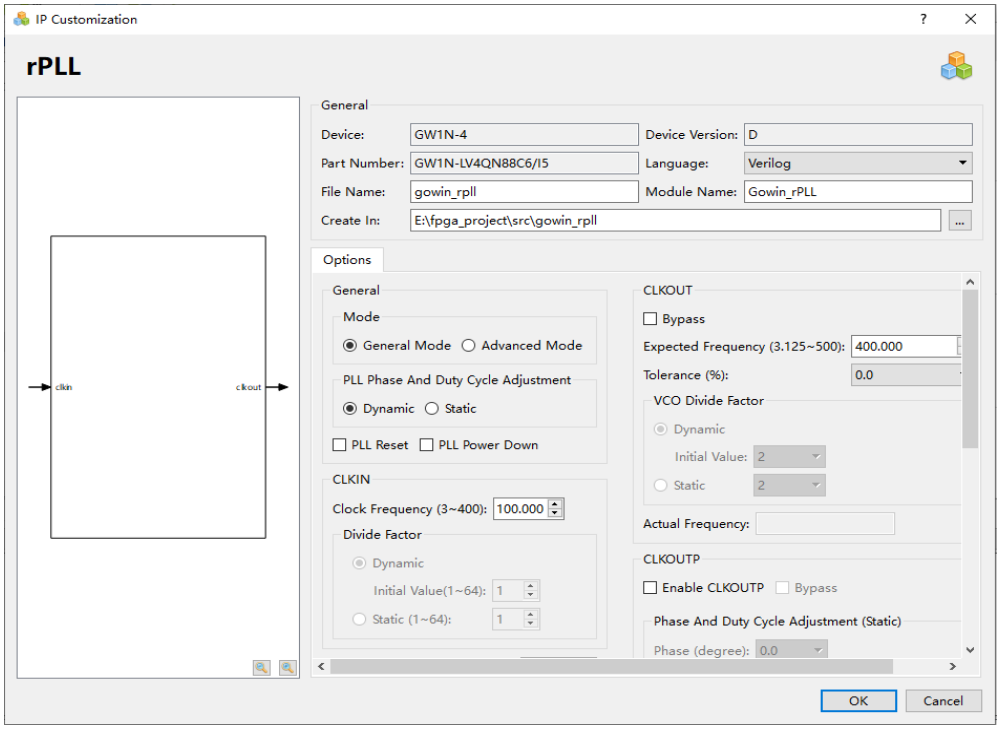
## 5.1.2 IP Generation

Click "rPLL" on the IP Core Generator interface and an overview of related information about rPLL will be displayed on the right side of the interface.

### IP Configuration

Double-click on the "rPLL" on the IP Core Generator interface to open the "IP Customization" window. It includes the "General", "Options", and ports diagram, as shown in Figure 5-4.

**Figure 5-4 IP Customization of rPLL**




1. General
   The General cnfiguration box is used to configure the generated IP design file. rPLL General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to configure IP by users, as shown in Figure 5-4.

- General: Configure "General Mode" or "Advanced Mode", select "Static Mode" or "Dynamic Mode" for PLL phase and duty cycle, and enable PLL Reset.
  - Mode: "General Mode" or "Advanced Mode". Input the input clock frequency and output clock frequency in general mode, the software will automatically calculate different division factors. Advanced mode is intended for advanced users and allows inputting the input frequency and different division factors to obtain the desired output frequency.
  - "PLL Phase And Duty Cycle Adjustment": "Static" Mode or "Dynamic" Mode.
  - "PLL Reset": Configures rPLL reset mode.
  - "PLL Power Down": Configures the reset_p port to put the rPLL in power saving mode.
- CLKIN: Configures input clock frequency, divide factor, and IDESEL Reset.

  - "Clock Frequency": Configures the frequency of the input clock with the range determined by the device.
  - Divide Factor: Sets the Divide Factor as "Dynamic" or "Static" in advanced mode. In Static mode, Divide Factor value can be set as a specific value, which ranges from 1 to 64. If the CLKOUT frequency is not in the range required by the device, an error prompt will pop up when you click "Calculate" or "OK" ; If the frequency of CLKIN/IDIV is not in the range required by the device, an error prompt will pop up when you click "Calculate" or "OK".

- CLKFB: Configures the source and divide factor of rPLL.
  - When configuring the source of the feedback clock, you can select Internal and External.
  - Divide Factor: Sets the Divide Factor as "Dynamic" or "Static" in advanced mode. In static mode, Divide Factor value can be set as a specific value, which ranges from 1 to 64. If the configuration is invalid, an error prompt will pop up when you click "Calculate" or "OK".
- Enable LOCK: Enables the LOCK port.
- CLKOUT: Configures the expected frequency, VCO, tolerance and actual frequency.
  - Bypass: Enables/ disables clkout bypass;
  - "Expected Frequency": Configures the expected frequency of the CLKOUT in general mode, with the range determined by the device.
  - Tolerance(%): Sets a tolerance for the CLKOUT between the expected frequency and actual frequency calculated.
  - VCO Divide Factor: Sets Divide Factor as "Dynamic" or "Static" in advanced mode. In static mode, the Divide Factor value can be set as a specific value, and the range is 2/4/8/16/32/48/64/80/96/112/128. If the configuration is invalid, an error prompt will pop up when you click "Calculate" or "OK".
  - Actual Frequency: The actual frequency that can be generated automatically.
- CLKOUTP: Configures the parameter of phase shift clock period trim, configures the adjustment parameter of phase and duty cycle, enables/disables reset of the phase shift clock.
  - Enable CLKOUTP: Configures phase shift clock output enable;
  - Bypass: Configures phase shift clock bypass enable;
  - Phase And Duty Cycle Adjustment (Static): Configures Phase (degree) and Duty Cycle (*1/16) in static mode;
- CLKOUTD: Configures the source, expected frequency, and divide

factor of the clock divider, and enables/disables CLKOUTD reset.

- Enable CLKOUTD: Configures frequency division clock output enable.
- Bypass: Configures frequency division clock bypass enable;
- Source: Configures the clock source for the frequency division clock output, CLKOUT and CLKOUTP.
- Expected Frequency: Sets the output clock frequency in General mode with the range determined by the device.
- Tolerance (%): Sets a tolerance for the CLKOUTD between the expected frequency and actual frequency calculated.
- Divide Factor (2~128): Selects the divide factor from the drop-down list in advanced mode. Only an even number between 2 and 128 can be selected. If an odd number is set, an error prompt will pop up when you click "OK".
- Actual Frequency: The actual frequency that can be generated automatically.

- CLKOUTD3: Select the source for divided-by-three clock output.
  - Enable CLKOUTD3: Enables/disables divided-by-three clock;
  - Source: Selects the clock source for the divided-by-three clock output, CLKOUT and CLKOUTP.
- Calculate: Calculates whether the current configuration is reasonable.
  - Calculate Divide Factor settings are based on the input/output frequency in general mode. If the actual frequency is different from the expected frequency, an Error prompt will pop up and the invalid value will be marked in red.
  - In "Advanced Mode", calculate the static division/multipilication frequency and VCO parameters.If the calculated results are invalid, click "Calculate", an Error prompt will pop up. If valid, click "Calculate", "succeed" prompt will pop up.

3. Port Diagram
   The port diagram is based on the IP Core configuration. The input/output number is updated in real time based on the configuration, as shown in Figure 5-4.

**IP Generated Files**

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_rpll.v" file is a complete Verilog module to generate instantiated rPLL, and it is generated according to the IP configuration;
- "gowin_rpll_tmp.v" is the template file;
- "gowin_rpll.ipc" file is IP configuration file. You can load the file to

configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5.2 PLLVR

## 5.2.1 Primitive Introduction

Gowin FPGA provides PLLVR (Phase_Locked Loop with regulator), which uses an external input reference clock signal to control the frequency and phase in the loop of internal oscillation signals.

### Device Supported

**Table 5-10 PLLVR Device Supported**

| Product Family | Series | Device |
|---|---|---|
| LittleBee® | GW1NS | GW1NS-4, GW1NS-4C, GW1NSR-4, GW1NSR-4C, GW1NSER-4C |

### Functional Description

PLLVR is a PLL with power regulation, and, based on the given input clock, it can adjust clock phase, duty cycle, frequency (multiplication and division) to output clocks with different phases and frequencies.

The performance of PLLVR is as follows:

PLLVR can adjust the frequency of the input clock CLKIN (multiply and division). The formulas are as follows:

$$f_{\mathrm{CLKOUT}} = (f_{CLKIN} * \mathrm{FBDIV})/\mathrm{IDIV}$$

$$f_{VCO} = f_{CLKOUT} * ODIV$$

$$f_{CLKOUTD} = f_{CLKOUT}/SDIV$$

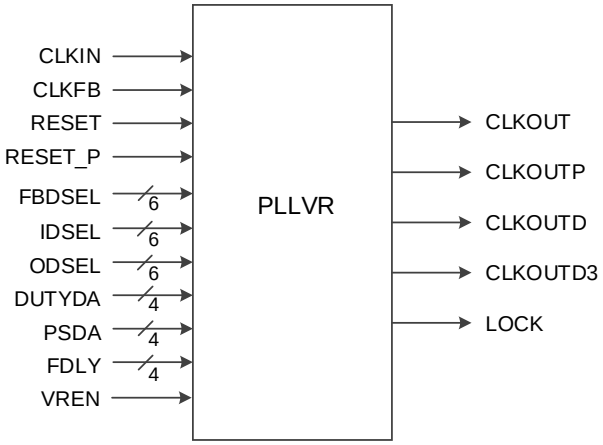
$$f_{PFD} = f_{CLKIN}/IDIV = f_{CLKOUT}/FBDIV$$

**Note!**

- $f_{CLKIN}$ is the input CLKIN frequency; $f_{CLKOUT}$ is the CLKOUT and CLKOUTP frequency; $f_{CLKOUTD}$ is the CLKOUTD frequency, and $f_{PFD}$ is the PFD phase discrimination frequency.
- IDIV, FBDIV, ODIV and SDIV are the actual frequency division coefficients of different frequency dividers, which can be adjusted to get the clock signal with expected frequency.
- For the frequency range of the PLLVR, please refer to the *FPGA Product Datasheet*.

## Port Diagram

**Figure 5-5 PLLVR Port Diagram**




## Port Description

**Table 5-11 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKIN | Input | Reference clock input signal |
| CLKFB | Input | Feedback clock input signal |
| RESET | Input | PLLVR asynchronous reset input signal, active-high. |
| RESET_P | Input | PLLVR power down input signal, active-high. The output of CLKOUT/CLKOUTP/CLKOUTD/CLKOUTD3 is 0 when PLL is not in bypass mode and RESET_P is in high level. |
| FBDSEL[5:0] | Input | Dynamically controls FBDIV value, ranging from 0 to 63, and the actual value is 64-FBDSEL. |
| IDSEL[5:0] | Input | Dynamically controls IDIV value, ranging from 0 to 63, and the actual value is 64-IDSEL. |
| ODSEL[5:0] | Input | Dynamicaly controls ODIV value: 2,4,8,16,32,48,64,80,96,112,128 |
| DUTYDA[3:0] | Input | Duty cycle dynamic adjustment signal |
| PSDA[3:0] | Input | Phase dynamic adjustment signal |
| FDLY[3:0] | Input | Fine delay dynamic adjustment signal |
| VREN | Input | PLLVR power supply enable signal, active high. |
| CLKOUT | Output | PLLVR clock output signal |
| LOCK | Output | PLLVR locked indicator, 1: locked, 0: unlocked. |
| CLKOUTP | Output | PLLVR clock output signal with phase and duty cycle adjustment |
| CLKOUTD | Output | Clock output singal of PLLVR through SDIV, output signal of CLKOUT or CLKOUTP through SDIV. |
| CLKOUTD3 | Output | Clock output signal of PLLVR through DIV3, output signal of CLKOUT or CLKOUTP through DIV3. |

**Note!**

CLKOUTD3 is the output clock signal of CLKOUT or CLKOUTP through DIV3. For its timing, you can refer to rPLL.

## Parameter Description

**Table 5-12 PLLVRParameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FCLKIN | 3~500 | 100 | Reference clock frequency |
| IDIV_SEL | 0~63 | 0 | IDIV frequency division coefficient static setting |
| DYN_IDIV_SEL | "true", "false" | "false" | IDIV frequency division coefficient static control parameter or dynamic control signal selection.<br>● false: Static, that is, select the parameter IDIV_SEL.<br>● true: Dynamic, namely select signal IDSEL. |
| FBDIV_SEL | 0~63 | 0 | FBDIV frequency division coefficient static setting |
| DYN_FBDIV_SEL | "true", "false" | "false" | FBDIV frequency division coefficient static control parameter or dynamic control signal selection.<br>● false: Static, that is, select the parameter FBDIV_SEL.<br>● true: Dynamic, namely select signal FBDSEL. |
| ODIV_SEL | 2,4,8,16,32,48,64,80,96,112,128 | 8 | ODIV frequency division coefficient static setting |
| DYN_ODIV_SEL | "true", "false" | "false" | ODIV frequency division coefficient static control parameter or dynamic control signal selection.<br>● false: Static, that is, select the parameter ODIV_SEL.<br>● true: Dynamic, |

| Name | Value | Default | Description |
|---|---|---|---|
| | | | namely select signal ODSEL. |
| PSDA_SEL | "0000"~"1111" | "0000" | Phase static adjustment |
| DUTYDA_SEL | "0010"~"1110" | "1000" | Duty cycle static adjustment |
| DYN_DA_EN | "true", "false" | "false" | The dynamic signal is selected as the control of phase and duty cycle adjustment.<br>● false: Static control<br>● true: Dynamic control |
| CLKOUT_FT_DIR | 1'b1 | 1'b1 | CLKOUT trim direction setting<br>1'b1: Decrease |
| CLKOUT_DLY_STEP | 0,1, 2,4 | 0 | CLKOUT trim coefficient setting<br>CLKOUT_DLY_STEP* delay(delay=50ps) |
| CLKOUTP_FT_DIR | 1'b1 | 1'b1 | CLKOUTP trim direction setting<br>1'b1: Decrease |
| CLKOUTP_DLY_STEP | 0,1, 2 | 0 | CLKOUTP trim coefficient setting<br>CLKOUTP_DLY_STEP*delay(delay=50ps) |
| DYN_SDIV_SEL | 2~128<br>(Even) | 2 | SDIV frequency division coefficient static setting |
| CLKFB_SEL | "internal", "external" | "internal" | CLKFB source selection<br>● internal: Feedback from internal CLKOUT.<br>● external: Feedback from external signals. |
| CLKOUTD_SRC | "CLKOUT", "CLKOUTP" | "CLKOUT" | CLKOUTD source selection |
| CLKOUTD3_SRC | "CLKOUT", "CLKOUTP" | "CLKOUT" | CLKOUTD3 source selection |
| CLKOUT_BYPASS | "true", "false" | "false" | Bypasses PLLVR, and CLKOUT comes directly from CLKIN.<br>● true: CLKIN bypasses PLLVR and acts directly on CLKOUT. |

| Name | Value | Default | Description |
|---|---|---|---|
| | | | ● false: Normal |
| CLKOUTP_BYPASS | "true", "false" | "false" | Bypasses PLLVR, and CLKOUT comes directly from CLKIN.<br>● true: CLKIN bypasses PLLVR and acts directly on CLKOUTP.<br>● false: Normal |
| CLKOUTD_BYPASS | "true", "false" | "false" | Bypasses PLLVR, and CLKOUT comes directly from CLKIN.<br>● true: CLKIN bypasses PLLVR and acts directly on CLKOUTD.<br>● false: Normal |
| DEVICE | "GW1NS-4"、"GW1NS-4C"、"GW1NSR-4"、"GW1NSR-4C"、"GW1NSER-4C" | "GW1NS-4" | Devices selected |

**Note!**

For IDSEL, FBDESL and ODSELport and parameter tables, phase and duty cycle tables, you can see those of rPLL.

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
PLLVR pllvr_inst(
    .CLKOUT(clkout),
    .LOCK(lock),
    .CLKOUTP(clkoutp),
    .CLKOUTD(clkoutd),
    .CLKOUTD3(clkoutd3),
    .VREN(vren),
    .RESET(reset),
    .RESET_P(reset_p),
    .CLKIN(clkin),
    .CLKFB(clkfb),
    .FBDSEL(fbdsel),
    .IDSEL(idsel),
```

```
        .ODSEL(odsel),
        .PSDA(psda),
        .DUTYDA(dutyda),
        .FDLY(fdly)
    );
    defparam pllvr_inst.FCLKIN = "50";
    defparam pllvr_inst.DYN_IDIV_SEL = "false";
    defparam pllvr_inst.IDIV_SEL = 0;
    defparam pllvr_inst.DYN_FBDIV_SEL = "false";
    defparam pllvr_inst.FBDIV_SEL = 1;
    defparam pllvr_inst.ODIV_SEL = 8;
    defparam pllvr_inst.PSDA_SEL = "0100";
    defparam pllvr_inst.DYN_DA_EN = "false";
    defparam pllvr_inst.DUTYDA_SEL = "1000";
    defparam pllvr_inst.CLKOUT_FT_DIR = 1'b1;
    defparam pllvr_inst.CLKOUTP_FT_DIR = 1'b1;
    defparam pllvr_inst.CLKOUT_DLY_STEP = 0;
    defparam pllvr_inst.CLKOUTP_DLY_STEP = 0;
    defparam pllvr_inst.CLKFB_SEL ="external";
    defparam pllvr_inst.CLKOUT_BYPASS = "false";
    defparam pllvr_inst.CLKOUTP_BYPASS = "false";
    defparam pllvr_inst.CLKOUTD_BYPASS = "false";
    defparam pllvr_inst.DYN_SDIV_SEL = 2;
    defparam pllvr_inst.CLKOUTD_SRC = "CLKOUT";
    defparam pllvr_inst.CLKOUTD3_SRC = "CLKOUT";
    defparam pllvr_inst.DEVICE = "GW1NS-4";
```

**Vhdl Instantiation:**

```
    COMPONENT PLLVR
        GENERIC(
                FCLKIN:STRING:= "100.0";
                DEVICE:STRING:= "GW1NS-4";
                DYN_IDIV_SEL:STRING:="false";
                IDIV_SEL:integer:=0;
                DYN_FBDIV_SEL:STRING:="false";
                FBDIV_SEL:integer:=0;
```

```
                DYN_ODIV_SEL:STRING:="false";
                ODIV_SEL:integer:=8;
                PSDA_SEL:STRING:="0000";
                DYN_DA_EN:STRING:="false";
                DUTYDA_SEL:STRING:="1000";
                CLKOUT_FT_DIR:bit:='1';
                CLKOUTP_FT_DIR:bit:='1';
                CLKOUT_DLY_STEP:integer:=0;
                CLKOUTP_DLY_STEP:integer:=0;
                CLKOUTD3_SRC:STRING:="CLKOUT";
                CLKFB_SEL : STRING:="internal";
                CLKOUT_BYPASS:STRING:="false";
                CLKOUTP_BYPASS:STRING:="false";
                CLKOUTD_BYPASS:STRING:="false";
                CLKOUTD_SRC:STRING:="CLKOUT";
                DYN_SDIV_SEL:integer:=2
        );
            PORT(
                CLKIN:IN std_logic;
                CLKFB:IN std_logic;
                IDSEL:IN std_logic_vector(5 downto 0);
                FBDSEL:IN std_logic_vector(5 downto 0);
                ODSEL:IN std_logic_vector(5 downto 0);
                VREN:IN std_logic;
                RESET:IN std_logic;
                RESET_P:IN std_logic;
                PSDA,FDLY:IN std_logic_vector(3 downto 0);
                DUTYDA:IN std_logic_vector(3 downto 0);
                LOCK:OUT std_logic;
                CLKOUT:OUT std_logic;
                CLKOUTD:OUT std_logic;
                CLKOUTP:OUT std_logic;
                CLKOUTD3:OUT std_logic
            );
    END COMPONENT;
```

```
uut:PLLVR
    GENERIC MAP(
                FCLKIN =>"100.0",
                DEVICE =>"GW1NS-4",
                DYN_IDIV_SEL=>"false",
                IDIV_SEL=>0,
                DYN_FBDIV_SEL=>"false",
                FBDIV_SEL=>0,
                DYN_ODIV_SEL=>"false",
                ODIV_SEL=>8,
                PSDA_SEL=>"0000",
                DYN_DA_EN=>"false",
                DUTYDA_SEL=>"1000",
                CLKOUT_FT_DIR=>'1',
                CLKOUTP_FT_DIR=>'1',
                CLKOUT_DLY_STEP=>0,
                CLKOUTP_DLY_STEP=>0,
                CLKOUTD3_SRC=>"CLKOUT",
                CLKFB_SEL=>"internal",
                CLKOUT_BYPASS=>"false",
                CLKOUTP_BYPASS=>"false",
                CLKOUTD_BYPASS=>"false",
                CLKOUTD_SRC=>"CLKOUT",
                DYN_SDIV_SEL=>2
    )
     PORT MAP (
        CLKIN=>clkin,
        CLKFB=>clkfb,
        IDSEL=>idsel,
        FBDSEL=>fbdsel,
        ODSEL=>odsel,
        VREN=>vren,
        RESET=>reset,
        RESET_P=>reset_p,
        PSDA=>psda,
```

```
        FDLY=>fdly,
        DUTYDA=>dutyda,
        LOCK=>lock,
        CLKOUT=>clkout,
        CLKOUTD=>clkoutd,
        CLKOUTP=>clkoutp ,
        CLKOUTD3=>clkoutd3
    );
```

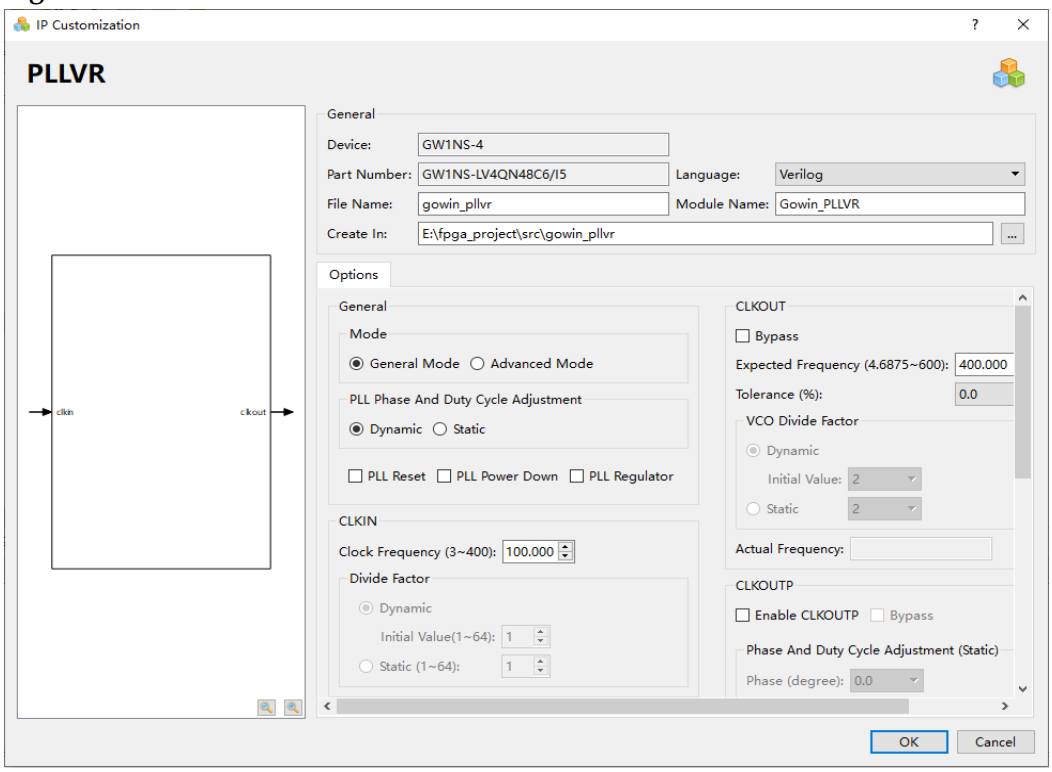
## 5.2.2 IP Generation

Click "PLLVR" on the IP Core Generator, and a brief introduction to the PLLVR will be displayed.

### IP Configuration

Double-click on the "PLLVR" to open the "IP Customization" window. It includes the "General", "Options", port diagram, and the "Help", as shown in Figure 5-6.

**Figure 5-6 IP Customization of PLLVR**




1. General
   The General configuration box is used to configure the generated IP design file. The PLLVR General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options

The Options configuration box is used to configure IP, as shown in Figure 5-6. PLLVR Options configuration box is similar to that of rPLL. For the details, please refer to rPLL, and PLL Regulator option is newly added.

3. Port Diagram
   The port diagram displays a sample diagram of the IP Core configuration. The number of input and output ports is updated in real time according to the Options configuration, as shown in Figure 5-4.

#### IP Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_pllvr.v" file is a complete Verilog module to generate instantiated PLLVR, and it is generated according to the IP configuration;
- "gowin_pllvr_tmp.v" is the template file;
- "gowin_pllvr.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5.3 PLLO

## 5.3.1 Primitive Introduction

Gowin FPGA products provide PLLO that supports four clock outputs and adjusts frequency, phase, and duty cycle based on a given input clock.

#### Device Supported

**Table 5-13 PLLO Device Supported**

| Product Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
| | GW1NR | GW1NR-2, GW1NR-2B |
| Arora | GW2AN | GW2AN-18X, GW2AN-9X |

#### Functional Description

PLLO supports four clock outputs and adjusts frequency (multiplication and division), phase, and duty cycle based on a given input clock to output clocks of different phases and frequencies. For the correct clock output, the input clock frequency must be set according to the frequency range described in the *FPGA Product Datasheet*.

PLLO can adjust the frequency of the input clock CLKIN (multiplication and division). The formulas are as follows:

$$f_{\mathrm{CLKOUTA}} = (f_{CLKIN} * \mathrm{FBDIV})/\mathrm{IDIV}$$

$$f_{VCO} = f_{CLKOUTA} * ODIVA$$

$$f_{CLKOUTX} = f_{IN\_ODIVX} / ODIVX$$

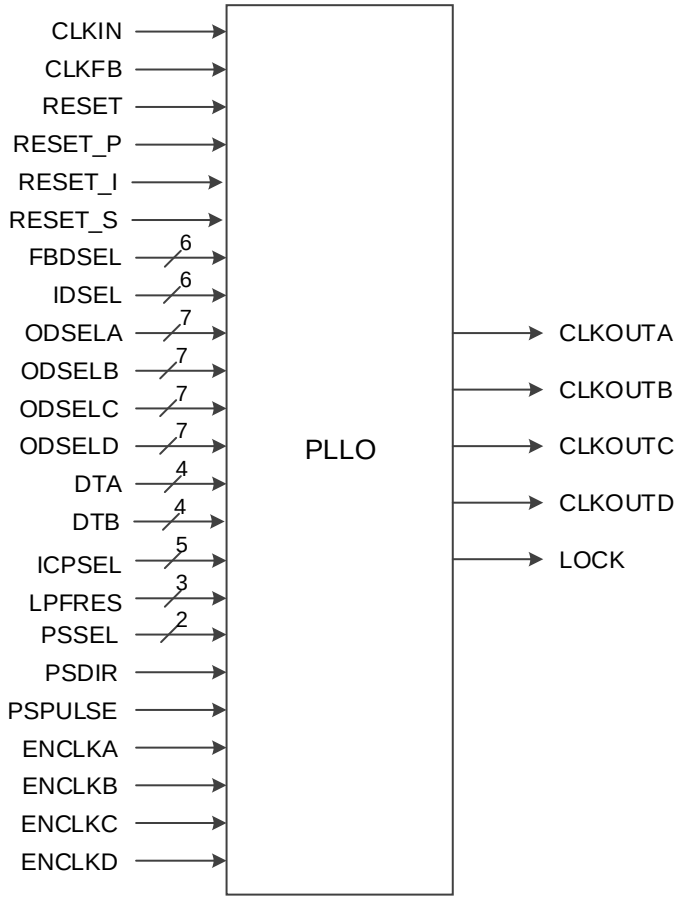
$$f_{PFD} = f_{CLKIN} / IDIV = f_{CLKOUTA} / FBDIV$$

**Note!**

- $f_{CLKIN}$ is the input CLKIN frequency;
- $f_{CLKOUTX}$: X=A/B/C/D is the output clock frequency of A/B/C/D channel, and ODIVX is the output division factor of A/B/C/D channel.
- $f_{IN\_ODIVX}$: X=A/B/C/D is the input clock frequency of ODIVX, and the default is $f_{VCO}$. It is connected according to the actual circuit when cascaded or bypassed.
- $f_{PFD}$ is the PFD phase discrimination frequency with a minimum value of not less than 3 MHz..
- IDIV, FBDIV, ODIV and SDIV are the actual frequency division coefficients of different dividers, which can be adjusted to get the clock signal with expected frequency.
- For the frequency range of the PLLO, please refer to the *FPGA Product Datasheet*.

## Port Diagram

**Figure 5-7 PLLO Port Diagram**




## Port Description

**Table 5-14 PLLO Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKIN | Input | Reference clock input signal |

| Port Name | I/O | Description |
|---|---|---|
| CLKFB | Input | Feedback clock input signal |
| RESET | Input | PLL reset signal, active-high. |
| RESET_P | Input | PLL power down signal, active-high. |
| RESET_I | Input | All with IDIV reset signal, including RESET and IDIV reset, active-high. |
| RESET_S | Input | Reset B/C/D channels, active-high. |
| FBDSEL[5:0] | Input | Dynamically controls FBDIV value, ranging from 0 to 63, and the actual value is 64-FBDSEL. |
| IDSEL[5:0] | Input | Dynamically controls IDIV value, ranging from 0 to 63, and the actual value is 64-IDSEL. |
| ODSELA[6:0] | Input | Dynamically controls ODIVA value, ranging from 0~127, and the actual value is 128-ODSELA. |
| ODSELB[6:0] | Input | Dynamically controls ODIVB value, ranging from 0~127, and the actual value is 128-ODSELA. |
| ODSELC[6:0] | Input | Dynamically controls ODIVC value, ranging from 0~127, and the actual value is 128-ODSELA. |
| ODSELD[6:0] | Input | Dynamically controls ODIVD value, ranging from 0~127, and the actual value is 128-ODSELA. |
| DTA[3:0] | Input | CLKOUTA duty cycle dynamic ddjustment signal |
| DTB[3:0] | Input | CLKOUTB duty cycle dynamic adjustment signal |
| ICPSEL[4:0] | Input | Dynamically controls ICP current. The current increases with the increase of the value, and the current is the smallest when the value is 0. |
| LPFRES[2:0] | Input | Dynamically controls the size of LPFRES. The value range of LPFRES is from small to large, which is R0~R7. The bandwidth corresponding to R0 is the largest, and the bandwidth corresponding to R7 is the smallest. |
| PSSEL[1:0] | Input | Dynamically controls phase shift channel selection |
| PSDIR | Input | Dynamically controls phase shift direction |
| PSPULSE | Input | Dynamically controls phase shift clock pulses |
| ENCLKA | Input | Dynamically controls the A channel clock output enable, if you want to use the dynamic enable, you need the static parameter CLKOUTA_EN = "TRUE". |
| ENCLKB | Input | Dynamically controls the B channel clock output enable, if you want to use dynamic enable, you need the static parameter CLKOUTB_EN = "TRUE". |
| ENCLKC | Input | Dynamically controls the C channel clock output enable, if you want to use dynamic enable, you need the static parameter CLKOUTC_EN = "TRUE". |
| ENCLKD | Input | Dynamically controls the D channel clock output enable, if you want to use dynamic enable, you need the static parameter CLKOUTD_EN = "TRUE". |
| CLKOUTA | Output | A channel clock output |
| CLKOUTB | Output | B channel clock output |
| CLKOUTC | Output | C channel clock output |

| Port Name | I/O | Description |
|---|---|---|
| CLKOUTD | Output | D channel clock output |
| LOCK | Output | PLL lock indication signal. 1: locked, 0: unlocked. |

## Parameter Description

**Table 5-15 PLLO Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FCLKIN | "3"~"400" | "100.0" | Reference clock frequency |
| IDIV_SEL | 0~63 | 0 | IDIV frequency division coefficient static setting, ranging from 1~64. |
| DYN_IDIV_SEL | "TRUE", "FALSE" | "FALSE" | IDIV frequency division coefficient static control parameter or dynamic control signal selection.<br>FLASE: Static, that is, select the parameter IDIV_SEL.<br>TRUE: Dynamic, namely select signal IDSEL. |
| FBDIV_SEL | 0~63 | 0 | FBDIV frequency division coefficient static setting, ranging from 1-64. |
| DYN_FBDIV_SEL | "TRUE", "FALSE" | "FALSE" | FBDIV frequency division coefficient static control parameter or dynamic control signal selection.<br>FLASE: Static, that is, select the parameter FBDIV_SEL.<br>TRUE: Dynamic, namely select signal FBDSEL. |
| ODIV_SEL | 1~128 | 4 | ODIVA frequency division coefficient static setting |
| DYN_ODIVA_SEL | "TRUE", "FALSE" | "FALSE" | ODIVA frequency division coefficient static control parameter or dynamic control signal selection.<br>FLASE: Static, that is, select the parameter ODIVA_SEL.<br>TRUE: Dynamic, namely select signal ODSELA. |
| ODIVB_SEL | 1~128 | 4 | ODIVB frequency division coefficient static setting |
| DYN_ODIVC_SEL | "TRUE", "FALSE" | "FALSE" | ODIVB frequency division coefficient static control parameter or dynamic control signal selection.<br>FLASE: Static, that is, select the parameter ODIVB_SEL.<br>TRUE: Dynamic, namely select signal ODSELB. |
| ODIVC_SEL | 1~128 | 4 | ODIVC frequency division coefficient static setting |
| DYN_ODIVC_SEL | "TRUE", "FALSE" | "FALSE" | ODIVC frequency division coefficient static control parameter or dynamic |

| Name | Value | Default | Description |
|---|---|---|---|
| | | | control signal selection.<br>FLASE: Static, that is, select the parameter ODIVC_SEL.<br>TRUE: Dynamic, namely select signal ODSELC. |
| ODIVD_SEL | 1~128 | 4 | ODIVD frequency division coefficient static setting |
| DYN_ODIVD_SEL | "TRUE", "FALSE" | "FALSE" | ODIVD frequency division coefficient static control parameter or dynamic control signal selection.<br>FLASE: Static, that is, select the parameter ODIVD_SEL.<br>TRUE: Dynamic, namely select signal ODSELD. |
| CLKOUTA_EN | "TRUE", "FALSE" | "TRUE" | A channel clock output enable |
| CLKOUTB_EN | "TRUE", "FALSE" | "TRUE" | B channel clock output enable |
| CLKOUTC_EN | "TRUE", "FALSE" | "TRUE" | C channel clock output enable |
| CLKOUTD_EN | "TRUE", "FALSE" | "TRUE" | D channel clock output enable |
| DYN_DTA_SEL | "TRUE", "FALSE" | "FALSE" | A channel duty cycle trim static control parameter or dynamic control signal selection.<br>FALSE: Static, that is, select ODIVC_SEL CLKOUTA_DT_DIR & CLKOUTA_DT_STEP<br>TRUE: Dynamic, namely select signal DTA |
| DYN_DTB_SEL | "TRUE", "FALSE" | "FALSE" | B channel duty cycle trim static control parameter or dynamic control signal selection.<br>FALSE: Static, that is, select CLKOUTB_DT_DIR & CLKOUTB_DT_STEP.<br>TRUE: Dynamic, namely select signal DTB. |
| CLKOUTA_DT_DIR | 1'b1, 1'b0 | 1'b1 | A channel duty cycle static trim direction.<br>1'b1: + means duty cycle increases, and adjusts the falling edge based on rising edge alignment.<br>1'b0: - means duty cycle decreases,and adjusts the rising edge based on falling edge alignment. |
| CLKOUTB_DT_DIR | 1'b1, 1'b0 | 1'b1 | B channel duty cycle static trim direction.<br>1'b1: + means duty cycle increases, and adjusts the falling edge based on |

| Name | Value | Default | Description |
|---|---|---|---|
| | | | rising edge alignment.<br>1'b0: - means duty cycle decreases,and adjusts the rising edge based on falling edge alignment. |
| CLKOUTA_DT_STEP | 0,1,2,4 | 0 | A channel duty cycle static trim step, 50ps per step. |
| CLKOUTB_DT_STEP | 0,1,2,4 | 0 | B channel duty cycle static trim step, 50ps per step. |
| CLKA_IN_SEL | 2'b00,2'b01,2'b11 | 2'b00 | ODIVA input clock source selction<br>2'b00/2'b01: From VCO output<br>2'b11: Bypass comes from CLKIN. |
| CLKA_OUT_SEL | 1'b0, 1'b1 | 1'b0 | A channel output clock source selection<br>1'b0: Form ODIVA output<br>1'b1: The output clock bypass comes from CLKIN. |
| CLKB_IN_SEL | 2'b00,2'b01,2'b10,2'b11 | 2'b00 | ODIVB input clock source selection<br>2'b00/2'b01: Form VCO output<br>2'b10: Cascade comes from CLKCAS_A.<br>2'b11: Bypass coms from CLKIN. |
| CLKB_OUT_SEL | 1'b0, 1'b1 | 1'b0 | B channel output clock source selection<br>1'b0: From ODIVB output<br>1'b1: The output clock bypass comes from CLKIN. |
| CLKC_IN_SEL | 2'b00,2'b01,2'b10,2'b11 | 2'b00 | ODIVC input clock source selction<br>2'b00/2'b01: From VCO output<br>2'b10: Cascade comes from CLKCAS_B<br>2'b11: Bypass coms from CLKIN |
| CLKC_OUT_SEL | 1'b0, 1'b1 | 1'b0 | C channel output clock source selection<br>1'b0: From ODIVC output<br>1'b1: The output clock bypass comes CLKIN. |
| CLKD_IN_SEL | 2'b00,2'b01,2'b10,2'b11 | 2'b00 | ODIVD input clock source selction<br>2'b00/2'b01: From VCO output<br>2'b10: Cascade comes from CLKCAS_C<br>2'b11: Bypass coms from CLKIN. |
| CLKD_OUT_SEL | 1'b0, 1'b1 | 1'b0 | D channel output clock source selection<br>1'b0: From ODIVD output<br>1'b1: The output clock bypass comes CLKIN. |
| CLKFB_SEL | "INTERNAL", | "INTERNAL" | CLKFB source selection<br>INTERNAL: From the feedback of |

| Name | Value | Default | Description |
|---|---|---|---|
| | "EXTERNAL" | | internal CLKOUTA<br>EXTERNAL: From the feedback of external signal |
| DYN_DPA_EN | "TRUE", "FALSE" | "FALSE" | Dynamic phase shift adjustment enable |
| DYN_PSB_SEL | "TRUE", "FALSE" | "FALSE" | B channel phase shift static control parameter or dynamic control signal selection<br>FALSE: Static, that is, select PSB_COARSE & PSB_FINE TRUE.<br>TRUE: Dynamic, that is, select DPA signals(PSSEL& PSDIR& PSPULSE) to achieve, while DYN_DPA_EN="TRUE" is required. |
| DYN_PSC_SEL | "TRUE", "FALSE" | "FALSE" | C channel phase shift static control parameter or dynamic control signal selection<br>FALSE: Static, that is, select PSC_COARSE & PSC_FINE TRUE<br>TRUE: Dynamic, that is, select DPA signals (PSSEL& PSDIR& PSPULSE) to achieve, while DYN_DPA_EN="TRUE" is required. |
| DYN_PSD_SEL | "TRUE", "FALSE" | "FALSE" | D channel phase shift static control parameter or dynamic control signal selection<br>FALSE: Static, that is, select PSD_COARSE & PSD_FINE TRUE.<br>TRUE: Dynamic, that is, select DPA signals (PSSEL& PSDIR& PSPULSE) to achieve, while DYN_DPA_EN="TRUE" is required. |
| PSB_COARSE | 1~128 | 1 | B-channel phase coarse shift static setting |
| PSB_FINE | 0~7 | 0 | B-channel phase fine shift static setting |
| PSC_COARSE | 1~128 | 1 | C-channel phase coarse shift static setting |
| PSC_FINE | 0~7 | 0 | C-channel phase fine shift static setting |
| PSD_COARSE | 1~128 | 1 | D-channel phase coarse shift static setting |
| PSD_FINE | 0~7 | 0 | D-channel phase fine shift static setting |
| DTMS_ENB | "TRUE", "FALSE" | "FALSE" | B channel (ODIVB=2~128) duty cycle adjustment enable<br>FALSE: 50% duty cycle<br>TRUE: When DYN_PSB_SEL="TRUE", set PSB_COARSE & PSB_FINE as the falling edge and combine it with |

| Name | Value | Default | Description |
|---|---|---|---|
| | | | dynamic phase shift as the rising edge to achieve dynamic adjustment of duty cycle (falling edge - rising edge). |
| DTMS_ENC | "TRUE", "FALSE" | "FALSE" | C channel (ODIVC=2~128) duty cycle adjustment enable<br>FALSE: 50% duty cycle<br>TRUE: When DYN_PSC_SEL="TRUE", set PSC_COARSE & PSC_FINE as the falling edge and combine it with dynamic phase shift as the rising edge to achieve dynamic adjustment of duty cycle (falling edge - rising edge). |
| DTMS_END | "TRUE", "FALSE" | "FALSE" | D channel (ODIVD=2~128) duty cycle adjustment enable<br>FALSE: 50% duty cycle<br>TRUE: When DYN_PSD_SEL="TRUE", set PSD_COARSE & PSD_FINE as the falling edge and combine it with dynamic phase shift as the rising edge to achieve dynamic adjustment of duty cycle (falling edge - rising edge). |
| RESET_I_EN | "TRUE", "FALSE" | "FALSE" | Enable the dynamic signal RESET_I, if you need to use the RESET_I port, you need to set this parameter to TRUE. |
| RESET_S_EN | "TRUE", "FALSE" | "FALSE" | Enable the dynamic signal RESET_S, if you need to use the RESET_S port, you need to set this parameter to TRUE. |
| DYN_ICP_SEL | "TRUE", "FALSE" | "FALSE" | ICPSEL static control parameter or dynamic control signal selection<br>FALSE: Static, i.e. the parameter ICP_SEL is selected.<br>TRUE: Dynamic, i.e. the dynamic signal ICPSEL is selected. |
| ICP_SEL | 5'bXXXXX, 5'b00000~5'b11111 | 5'bXXXXX | ICP current static setting<br>5'bXXXXX: Indicates that the software will automatically calculate and set this parameter.<br>5'b00000~5'b11111: If you need to set it by yourself, you can set it in the parameter range as needed. |
| DYN_RES_SEL | "TRUE", "FALSE" | "FALSE" | LPRREF static control parameter or dynamic control signal selection<br>FALSE: Static, i.e. select the parameter LPR_REF.<br>TRUE: Dynamic, i.e. select the dynamic signal LPFRES. |
| LPR_REF | 7'bXXXX | 7'bXXXX | LPRRES static setting |

| Name | Value | Default | Description |
|---|---|---|---|
| | XXX, 7'b0000000(R0),7'b0000001(R1),7'b0000010(R2),7'b0000100(R3),7'b0001000(R4),7'b0010000(R5),7'b0100000(R6),7'b1000000(R7) | XXX | 7'bXXXXXXX: Indicates that the software will automatically calculate and set the parameter.<br>7'b0000000~7'b1000000 (8 values): If you need to set it by yourself, you can choose to set it in the corresponding eight values as needed. |

**Table 5-16 IDSEL Port Parameter Comparison Table**

| IDSEL[5:0] | IDIV Static Value | IDIV Actual Value |
|---|---|---|
| 111111 | 0 | 1 |
| 111110 | 1 | 2 |
| 111101 | 2 | 3 |
| 111100 | 3 | 4 |
| 111011 | 4 | 5 |
| 111010 | 5 | 6 |
| 111001 | 6 | 7 |
| 111000 | 7 | 8 |
| 110111 | 8 | 9 |
| …… | …… | …… |
| 000000 | 63 | 64 |

**Table 5-17 FBDSEL Port Parameter Comparison Table**

| FBDSEL [5:0] | FBDIV Static Value | FBDIV Actual Value |
|---|---|---|
| 111111 | 0 | 1 |
| 111110 | 1 | 2 |
| 111101 | 2 | 3 |
| 111100 | 3 | 4 |
| 111011 | 4 | 5 |
| 111010 | 5 | 6 |
| 111001 | 6 | 7 |
| 111000 | 7 | 8 |
| 110111 | 8 | 9 |
| …… | …… | …… |

| FBDSEL [5:0] | FBDIV Static Value | FBDIV Actual Value |
|---|---|---|
| 000000 | 63 | 64 |

**Table 5-18 ODSELX（X=A/B/C/D）Port Parameter Comparison Table**

| ODSELX [6:0] | ODIVX Static Value | ODIVX Actual Value |
|---|---|---|
| 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 |

### Phase Shift

PLLO supports phase shift, which includes static phase shift and dynamic phase shift, and the dynamic phase shift is only supported by B/C/D channels. Static phase shift is achieved by setting the parameters PSX_COARSE and PSX_FINE (X=A/B/C/D). Dynamic phase shift is achieved by signals PSSEL, PSDIR and PSPULSE. PSSEL is used to control channel selection, PSDIR is used to control plus or minus operation. For a PSPULSE pulse falling edge, DYN_FINE is plus/minus 1. DYN_COARSE is plus or minus 1 when DYN_FINE overflows or underflows, where DYN_ COARSE value is less than or equal to ODIV.

The phase shift can be configured and calculated according to the following equation (using the B channel as an example)

- If COARSE_B< ODIVB, ps = (FINE_B/8 + COARSE_B)/ODIVB*360
- COARSE_B=ODIVB, ps = (FINE_B/8)/ODIVB*360

**Note!**

- DYN_FINE and DYN_COARSE are internal signals generated by the DPA through PSSEL, PSDIR, PSPULSE signals.
- FINE_B is the dynamic DYN_FINE_B or the static parameter PSB_FINE selected via DYN_PSB_SEL, and COARSE_B is the dynamic DYN_COARSE_B or the static parameter PSB_COARSE selected via DYN_PSB_SEL.
- When CLKX_IN_SEL(X=B/C/D), FINE_X (X=B/C/D) should be set to 0 when bypass or cascade is selected.

### Duty Cycle Adjustment

PLLO dynamic duty cycle adjustment is only supported by B/C/D channels. Duty cycle is defined as follows:

$$\text{Duty cycle} = (\text{falling edge} - \text{rising edge}) / \text{cycle\_period}$$

The position of the falling edge is determined by the static phase shift setting, defined as DUTY. The position of the rising edge is determined by the dynamic phase shift setting PHASE, DYN_FINE and DYN_COARSE are internal signals generated by the DPA, and you can see the relevant descriptions in the Duty Cycle Adjustment. The formula for calculating DUTY and PHASE is as follows (using channel B as an example).

$$\text{DUTY} = (\text{PSB\_FINE}/8 + \text{PSB\_COARSE})$$

$$\text{PHASE} = (\text{DYN\_FINEB}/8 + \text{DYN\_COARSEB})$$

The dynamic duty cycle calculation is as follows:

- If DUTY> PHASE, DutyCycle= (DUTY- PHASE) / ODIVB.
- If DUTY< PHASE, DutyCycle= (DUTY- PHASE) / ODIVB + 1.

**Note!**

- Dynamic duty cycle adjustment is not supported when ODIV=1, and the duty cycle is 50%.
- When ODIV>=2, DUTY- PHASE does not support a value between (-0.5, 0.5).
- When CLKX_IN_SEL(X=B/C/D) selects bypass or cascade, if ODIV(>2) is odd then duty cycle is not 50% (high < low, i.e. less than 50%).

### Duty Cycle Trim

The A and B channels of PLLO supports duty cycle trim, which is achieved by setting the direction and step of duty cycle; static and dynamic modes are supported.

1. Static

- Trim direction, controlled by the parameters CLKOUTA_DT_DIR/ CLKOUTB_DT_DIR.
- Trim step, controlled by the parameters CLKOUTA_DT_STEP/ CLKOUTB_DT_STEP.

2. Dynamic.

- Trim direction, controlled by ports DTA[3]/DTB[3].
- Trim step, controlled by ports7-DTA[2:0]/7-DTB[2:0].

**Table 5-19 PLLO Duty Trim Comparison Table**

| Direction | Step | Delay Value |
|---|---|---|
| 1'b0 | 0 | 0 |
| | 1 | -50ps |
| | 2 | -100ps |
| | 4 | -200ps |
| 1'b1 | 0 | 0 |
| | 1 | +50ps |
| | 2 | +100ps |

| Direction | Step | Delay Value |
|---|---|---|
| | 4 | +200ps |

A and B channels output the same frequency clock, and the B channel clock duty cycle is fined adjusted taking A channel clock as a reference. The specific timing is as shown in Figure 5-6 and Figure 5-7.

**Figure 5-8 B-channel Duty Cycle Trim Timing Diagram (Direction 1'b1, Step 1)**

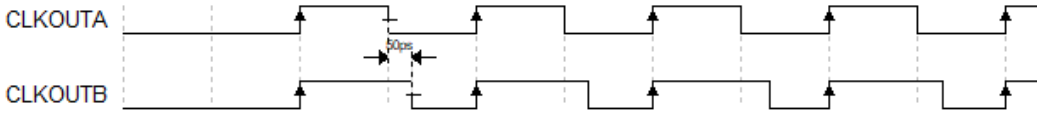
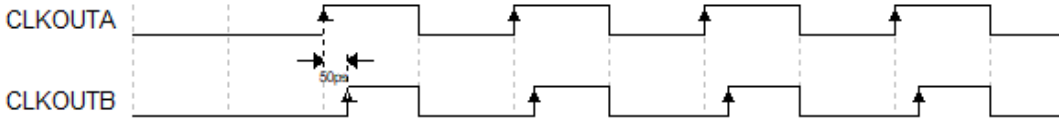



**Figure 5-9 B-channel Duty Cycle Trim Timing Diagram (Direction 1'b0,Step 1)**




### ICPSEL/ LPFRES Setting

PLLO supports ICPSEL and LPFRES setting, including static and dynamic. You can set the dynamic according to the actual needs, the static default is X. Gowin software will automatically calculate and configure if you need.

The value range of ICPSEL increases linearly from small to large, and can be divided into ICP1, ICP2, .....ICPN..... ICP31, ICP32, with a total of 32. ICP1 corresponds to the minimum current, and ICP32 corresponds to the maximum current. The value of ICP can be considered qualitatively as the larger N is, the larger ICP is, and the smaller N is, the smaller ICP is.

The range of LPRRES values from small to large is R0, R1, R2, R3, R4, R5, R6, R7. R0 corresponds to the largest bandwidth and R7 corresponds to the smallest bandwidth. Several typical values are given: R7->250KHz, R4->1.6MHz, R1->12MHz.

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation：**

```
PLLO pllo_inst (
    .LOCK(lock),
    .CLKOUTA(clkouta),
    .CLKOUTB(clkoutb),
    .CLKOUTC(clkoutc),
    .CLKOUTD(clkoutd),
```

```
        .CLKIN(clkin),
        .CLKFB(clkfb),
        .RESET(reset),
        .RESET_P(reset_p),
        .RESET_I(reset_i),
        .RESET_S(reset_s),
        .FBDSEL(fbdsel),
        .IDSEL(idsel),
        .ODSELA(odsela),
        .ODSELB(odselb),
        .ODSELC(odselc),
        .ODSELD(odseld),
        .DTA(dta),
        .DTB(dtb),
        .ICPSEL(icpsel),
        .LPFRES(lpfres),
        .PSSEL(pssel),
        .PSDIR(psdir),
        .PSPULSE(pspulse),
        .ENCLKA(enclka),
        .ENCLKB(enclkb),
        .ENCLKC(enclkc),
        .ENCLKD(enclkd)
    );

    defparam pllo_inst.FCLKIN = "100";
    defparam pllo_inst.DYN_IDIV_SEL = "FALSE";
    defparam pllo_inst.IDIV_SEL = 0;
    defparam pllo_inst.DYN_FBDIV_SEL = "FALSE";
    defparam pllo_inst.FBDIV_SEL = 0;
    defparam pllo_inst.DYN_ODIVA_SEL = "FALSE";
    defparam pllo_inst.ODIVA_SEL = 4;
    defparam pllo_inst.DYN_ODIVB_SEL = "FALSE";
    defparam pllo_inst.ODIVB_SEL = 4;
    defparam pllo_inst.DYN_ODIVC_SEL = "FALSE";
```

```
defparam pllo_inst.ODIVC_SEL = 4;
defparam pllo_inst.DYN_ODIVD_SEL = "FALSE";
defparam pllo_inst.ODIVD_SEL = 4;
defparam pllo_inst.CLKOUTA_EN = "TRUE";
defparam pllo_inst.CLKOUTB_EN = "FALSE";
defparam pllo_inst.CLKOUTC_EN = "FALSE";
defparam pllo_inst.CLKOUTD_EN = "FALSE";
defparam pllo_inst.DYN_DTA_SEL = "FALSE";
defparam pllo_inst.DYN_DTB_SEL = "FALSE";
defparam pllo_inst.CLKOUTA_DT_DIR = 1'b1;
defparam pllo_inst.CLKOUTB_DT_DIR = 1'b1;
defparam pllo_inst.CLKOUTA_DT_STEP = 0;
defparam pllo_inst.CLKOUTB_DT_STEP = 0;
defparam pllo_inst.CLKA_IN_SEL = 2'b00;
defparam pllo_inst.CLKA_OUT_SEL = 1'b0;
defparam pllo_inst.CLKB_IN_SEL = 2'b00;
defparam pllo_inst.CLKB_OUT_SEL = 1'b0;
defparam pllo_inst.CLKC_IN_SEL = 2'b00;
defparam pllo_inst.CLKC_OUT_SEL = 1'b0;
defparam pllo_inst.CLKD_IN_SEL = 2'b00;
defparam pllo_inst.CLKD_OUT_SEL = 1'b0;
defparam pllo_inst.CLKFB_SEL = "INTERNAL";
defparam pllo_inst.DYN_DPA_EN = "FALSE";
defparam pllo_inst.DYN_PSB_SEL = "FALSE";
defparam pllo_inst.DYN_PSC_SEL = "FALSE";
defparam pllo_inst.DYN_PSD_SEL = "FALSE";
defparam pllo_inst.PSB_COARSE = 1;
defparam pllo_inst.PSB_FINE = 0;
defparam pllo_inst.PSC_COARSE = 1;
defparam pllo_inst.PSC_FINE = 0;
defparam pllo_inst.PSD_COARSE = 1;
defparam pllo_inst.PSD_FINE = 0;
defparam pllo_inst.DTMS_ENB = "FALSE";
defparam pllo_inst.DTMS_ENC = "FALSE";
defparam pllo_inst.DTMS_END = "FALSE";
```

```
defparam pllo_inst.RESET_I_EN = "FALSE";
defparam pllo_inst.RESET_S_EN = "FALSE";
defparam pllo_inst.DYN_ICP_SEL = "FALSE";
defparam pllo_inst.ICP_SEL = 5'bXXXXX;
defparam pllo_inst.DYN_RES_SEL = "FALSE";
defparam pllo_inst.LPR_REF = 7'bXXXXXXX
```

**VHDL Instantiation：**

```
COMPONENT PLLO
    GENERIC (
            FCLKIN : STRING := "100.0";
            DYN_IDIV_SEL : STRING := "FALSE";
            IDIV_SEL : integer := 0;
            DYN_FBDIV_SEL : STRING := "FALSE";
            FBDIV_SEL : integer := 0;
            DYN_ODIVA_SEL : STRING := "FALSE";
            ODIVA_SEL : integer := 4;
            DYN_ODIVB_SEL : STRING := "FALSE";
            ODIVB_SEL : integer := 4;
            DYN_ODIVC_SEL : STRING := "FALSE";
            ODIVC_SEL : integer := 4;
            DYN_ODIVD_SEL : STRING := "FALSE";
            ODIVD_SEL : integer := 4;
            CLKOUTA_EN : STRING := "TRUE";
            CLKOUTB_EN : STRING := "TRUE";
            CLKOUTC_EN : STRING := "TRUE";
            CLKOUTD_EN : STRING := "TRUE";

            DYN_DTA_SEL : STRING := "FALSE";
            DYN_DTB_SEL : STRING := "FALSE";
            CLKOUTA_DT_DIR : bit := '1';
            CLKOUTB_DT_DIR : bit := '1';
            CLKOUTA_DT_STEP : integer := 0;
            CLKOUTB_DT_STEP : integer := 0;
            CLKA_IN_SEL    : bit_vector := "00";
            CLKA_OUT_SEL : bit := '0';
```

```
            CLKB_IN_SEL   : bit_vector := "00";
            CLKB_OUT_SEL : bit := '0';
            CLKC_IN_SEL   : bit_vector := "00";
            CLKC_OUT_SEL : bit := '0';
            CLKD_IN_SEL   : bit_vector := "00";
            CLKD_OUT_SEL : bit := '0';
            CLKFB_SEL : STRING := "INTERNAL";
            DYN_DPA_EN : STRING := "FALSE";
            DYN_PSB_SEL : STRING := "FALSE";
            DYN_PSC_SEL : STRING := "FALSE";
            DYN_PSD_SEL : STRING := "FALSE";
            PSB_COARSE : integer := 1;
            PSB_FINE : integer := 0;
            PSC_COARSE : integer := 1;
            PSC_FINE : integer := 0;
            PSD_COARSE : integer := 1;
            PSD_FINE : integer := 0;
            DTMS_ENB : STRING := "FALSE";
            DTMS_ENC : STRING := "FALSE";
            DTMS_END : STRING := "FALSE";
            RESET_I_EN : STRING := "FALSE";
            RESET_S_EN : STRING := "FALSE";
            DYN_ICP_SEL : STRING := "FALSE";
            ICP_SEL : std_logic_vector(4 downto 0) := "XXXXX";
            DYN_RES_SEL : STRING := "FALSE";
            LPR_REF : std_logic_vector(6 downto 0) := "XXXXXXX"
        );
        PORT (
            CLKIN : IN std_logic;
            CLKFB : IN std_logic:='0';
            RESET,RESET_P : IN std_logic:='0';
            RESET_I,RESET_S : IN std_logic:='0';
            IDSEL,FBDSEL : IN std_logic_vector(5 downto 0);
            ODSELA, ODSELB, ODSELC, ODSELD : IN
std_logic_vector(6 downto 0);
```

```
                DTA, DTB : IN std_logic_vector(3 downto 0);
                ICPSEL : IN std_logic_vector(4 downto 0);
                LPFRES : IN std_logic_vector(2 downto 0);
                PSSEL : IN std_logic_vector(1 downto 0);
                PSDIR,PSPULSE : IN std_logic;
                ENCLKA,ENCLKB,ENCLKC,ENCLKD : IN std_logic;
                LOCK : OUT std_logic;
                CLKOUTA : OUT std_logic;
                CLKOUTB : OUT std_logic;
                CLKOUTC : OUT std_logic;
                CLKOUTD : OUT std_logic
        );
END COMPONENT;
uut:PLLO
    GENERIC MAP(
        FCLKIN : STRING => "100.0";
            DYN_IDIV_SEL =>"FALSE";
            IDIV_SEL => 0;
          DYN_FBDIV_SEL=> "FALSE";
            FBDIV_SEL => 0;
            DYN_ODIVA_SEL =>"FALSE";
           ODIVA_SEL => 4;
            DYN_ODIVB_SEL=> "FALSE";
            ODIVB_SEL => 4;
            DYN_ODIVC_SEL => "FALSE";
          ODIVC_SEL => 4;
            DYN_ODIVD_SEL=> "FALSE";
            ODIVD_SEL => 4;
            CLKOUTA_EN => "TRUE";
            CLKOUTB_EN => "TRUE";
            CLKOUTC_EN => "TRUE";
            CLKOUTD_EN =>"TRUE";
            DYN_DTA_SEL =>"FALSE";
            DYN_DTB_SEL =>"FALSE";
            CLKOUTA_DT_DIR => '1';
```

```
                CLKOUTB_DT_DIR => '1';
                CLKOUTA_DT_STEP => 0;
                CLKOUTB_DT_STEP => 0;
                CLKA_IN_SEL => "00";
                CLKA_OUT_SEL => '0';
                CLKB_IN_SEL => "00";
                CLKB_OUT_SEL => '0';
                CLKC_IN_SEL => "00";
                CLKC_OUT_SEL => '0';
                CLKD_IN_SEL => "00";
                CLKD_OUT_SEL => '0';
                CLKFB_SEL => "INTERNAL";
                DYN_DPA_EN => "FALSE";
                DYN_PSB_SEL => "FALSE";
                DYN_PSC_SEL => "FALSE";
                DYN_PSD_SEL => "FALSE";
                PSA_COARSE => 0;
                PSA_FINE => 0;
                PSB_COARSE => 0;
                PSB_FINE => 0;
                PSC_COARSE => 0;
                PSC_FINE => 0;
                PSD_COARSE => 0;
                PSD_FINE => 0;
                DTMS_ENB => "FALSE";
                DTMS_ENC => "FALSE";
                DTMS_END => "FALSE";
                RESET_I_EN => "FALSE";
                RESET_S_EN => "FALSE";
                DYN_ICP_SEL => "FALSE";
                ICP_SEL => "XXXXX";
                DYN_RES_SEL => "FALSE";
                LPR_REF => "XXXXXXX"
    )
     PORT MAP(
```

```
        LOCK=>lock,
    CLKOUTA=> clkouta,
    CLKOUTB=>clkoutb,
    CLKOUTC=>clkoutc,
    CLKOUTD=>clkoutd,
    CLKIN=>clkin,
    CLKFB=>clkfb,
    RESET=>reset,
    RESET_P=>reset_p,
    RESET_I=>reset_i,
    RESET_S=>reset_s,
    FBDSEL=>fbdsel,
    IDSEL=>idsel,
    ODSELA=>odsela,
    ODSELB=>odselb,
    ODSELC=>odselc,
    ODSELD=>odseld,
    DTA=>dta,
    DTB=>dtb,
    ICPSEL=>icpsel,
    LPFRES=>lpfres,
    PSSEL=>pssel,
    PSDIR=>psdir,
    PSPULSE=>pspulse,
    ENCLKA=>enclka,
    ENCLKB=>enclkb,
    ENCLKC=>enclkc,
    ENCLKD=>enclkd
);
```

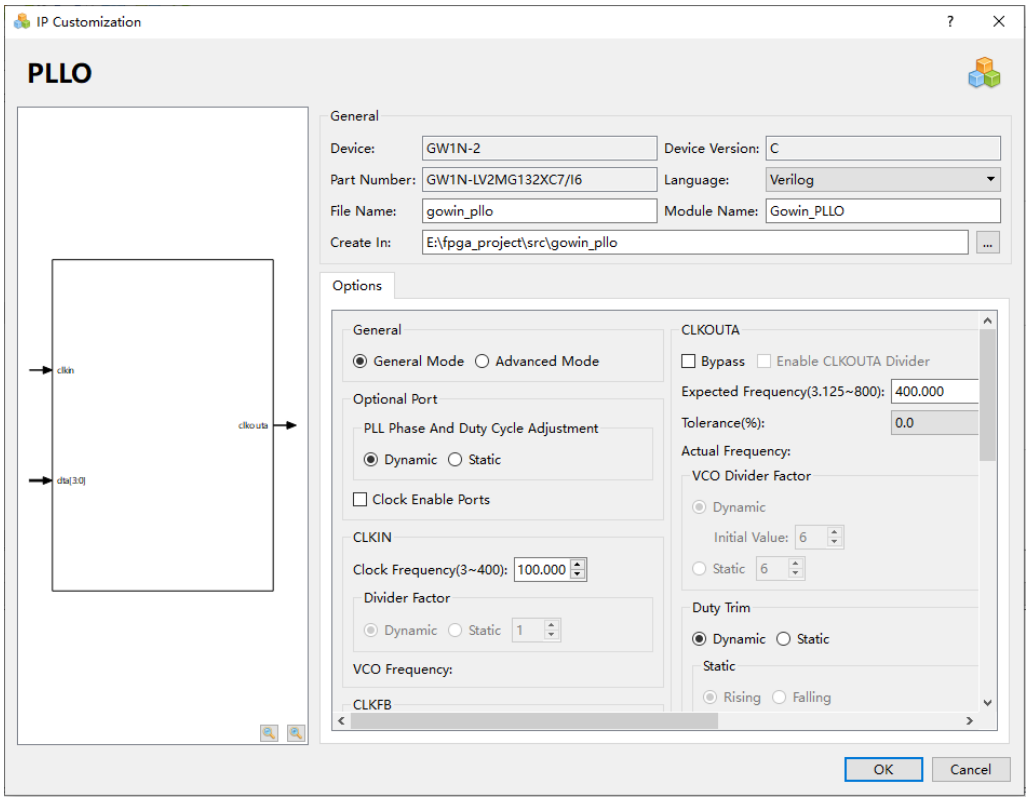
## 5.3.2 IP Generation

Click "PLLO" on the IP Core Generator, and a brief introduction to the PLLO will be displayed.

### IP Configuration

Double-click on the "PLLO" to open the "IP Customization" window. It includes the "General", "Options", and ports diagram, as shown in Figure 5-10.

**Figure 5-10 IP Customization of PLLO**




1. General
   The General configuration box is used to configure the generated IP design file. PLLO General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to configure IP, as shown in Figure 5-10.
   - General: Configures "General Mode" or "Advanced Mode". Input the input clock frequency and output clock frequency in general mode, the software will automatically calculate different division factors. Advanced mode is intended for advanced users and allows inputting the input frequency and different division factors to obtain the desired output frequency.
   - Optional Port: Configures dynamic and static modes for output phase and duty cycle adjustment, and enables PLLO output clock
     - "PLL Phase And Duty Cycle Adjustment": "Static" Mode or "Dynamic" Mode. You can configure the duty cycle of the output and the mode of the phase shift.
     - "Clock Enable Ports": Enables the output clock port of PLLO.
   - CLKIN: Configures input clock frequency and divide factor.
     - "Clock Frequency": Configures the frequency of the input clock with the range of 3~400MHz.

  - Divide Factor: Sets the Divide Factor as "Dynamic" or "Static" in advanced mode. In Static mode, Divide Factor value can be set as a specific value, ranging from 1 to 64. If the CLKOUT frequency is not in the range required by the device, an error prompt will pop up when you click "Calculate" or "OK" ; If the frequency of CLKIN/IDIV is not in the range required by the device, an error prompt will pop up when you click "Calculate" or "OK".
  - "VCO Frequency" is the calculated frequency of the VCO, read only.
- CLKFB: Configures the source and divide factor of PLLO.
  - When configuring the source of the feedback clock, you can select Internal and External.
  - Divide Factor: Sets the Divide Factor as "Dynamic" or "Static" in advanced mode. In static mode, Divide Factor value can be set as a specific value, ranging from 1 to 64. If the configuration is invalid, an error prompt will pop up when you click"Calculate" or "OK".
- ICP and LPF
  - ICPSEL: Configures ICP current, supports "Dynamic" and "Static". In static mode, you can configure ICP value, the range is ICP1 ~ ICP32, the default is X, meaning the software will automatically calculate and configure.
  - LPFRES: Configures low-pass filtering resistor, supports "Dynamic" and and "Static". In static mode, you can configure the value of RES, and the range is R0~R7, the default is X, meaning the software will automatically calculate and configure.
- PLL Reset
  - "PLL Reset": Configures the RESET enable mode of PLLO.
  - "PLL Power Down": Configures the RESET_P port to put the PLLO in power-down mode.
  - "CLKIN Divider Reset": Enables RESET_I.
  - "CLKOUTB/CLKOUTC/CLKOUTD Divider Reset": Enables RESET_S.
- Enable LOCK: Enables the LOCK port.
- CLKOUTA: Configures the expected frequency, VCO and duty trim of A channel.
  - Bypass: Enables/ disables bypass.
  - Enable CLKOUTA Divider: Configures VCO clock bypass.
  - "Expected Frequency": Configures the expected frequency of the output clock CLKOUTA in general mode, with the range of

3.125M~800M in non-bypass mode.

- Tolerance(%): Sets a tolerance for the CLKOUTA between expected frequency and actual frequency calculated.
- VCO Divide Factor: "Dynamic" or "Static" in advanced mode. In static mode, the Divide Factor value can be set as a specific value, and the range of 1~128. If the configuration is invalid, an error prompt will pop up when the user clicks "Calculate" or "OK".
- Actual Frequency: The actual frequency that can be generated automatically.
- Duty Trim: Trims duty cycle, and supports "Dynamic" and "Static". In static mode, it includes "Rising" and "Falling", and you can configure the specific value of "Step" as 0, 1, 2, 4.

- CLKOUTB: Configures the expected frequency, VCO, duty trim, etc. of B channel.
  - Bypass: Enables/disables bypass.
  - Enable CLKOUTB Divider: Configures VCO clock bypass.
  - "Expected Frequency": Configures the expected frequency of the output clock CLKOUTB in general mode, with the range of 3.125M~800M in non-bypass mode.
  - Tolerance(%): Sets a tolerance for the CLKOUTB between expected frequency and actual frequency calculated.
  - VCO Divide Factor: "Dynamic" or "Static" in advanced mode. In static mode, the Divide Factor value can be set as a specific value, and the range of 1~128. If the configuration is invalid, an error prompt will pop up when you click "Calculate" or "OK".
  - Actual Frequency: The actual frequency that can be generated automatically.
  - Duty Trim: Trims duty cycle, and supports "Dynamic" and "Static". In static mode, it includes "Rising" and "Falling", and you can configure the specific value of "Step" as 0, 1, 2, 4.
  - Phase (degree): Configures the adjusted phase degree, supports "Dynamic" and "Static". In the static mode, you can configure the phase degree.
  - Duty Cycle: Configures duty cycle, supports "Dynamic" and "Static". In static mode, it is 50%. In dynamic mode, you need to configure the phase and combine with dynamic DPA to achieve the duty cycle.
- CLKOUTC: Configures the expected frequency, VCO, duty trim, etc. of C channel.
  - Bypass: Enables/ disables bypass.
  - Enable CLKOUTC Divider: Configures VCO clock bypass.

- "Expected Frequency": Configures the expected frequency of the output clock CLKOUTC in general mode, with the range of 3.125M~800M in non-bypass mode.
- Tolerance(%): Sets a tolerance for the CLKOUTC between expected frequency and actual frequency calculated.
- VCO Divide Factor: "Dynamic" or "Static" in advanced mode. In static mode, the Divide Factor value can be set as a specific value, and the range of 1~128. If the configuration is invalid, an error prompt will pop up when you click "Calculate" or "OK".
- Actual Frequency: The actual frequency that can be generated automatically.
- Phase (degree): Configures the adjusted phase degree, supports "Dynamic" and "Static". In the static mode, you can configure the phase degree.
- Duty Cycle: Configures duty cycle, supports "Dynamic" and "Static". In static mode, it is 50%. In dynamic mode, you need to configure the phase and combine with dynamic DPA to achieve the duty cycle.

- CLKOUTD: Configures the expected frequency, VCO, duty trim, etc. of B channel.
  - Bypass: Enables/ disables bypass.
  - Enable CLKOUTD Divider: Configures VCO clock bypass.
  - "Expected Frequency": Configures the expected frequency of the output clock CLKOUTD in general mode, with the range of 3.125M~800M in non-bypass mode.
  - Tolerance(%): Sets a tolerance for the CLKOUTD between expected frequency and actual frequency calculated.
  - VCO Divide Factor: "Dynamic" or "Static" in advanced mode. In static mode, the Divide Factor value can be set as a specific value, and the range of 1~128. If the configuration is invalid, an error prompt will pop up when the user clicks "Calculate" or "OK".
  - Actual Frequency: The actual frequency that can be generated automatically.
  - Phase (degree): Configures the adjusted phase degree, supports "Dynamic" and "Static". In the static mode, you can configure the phase degree.
  - Duty Cycle: Configures duty cycle, supports "Dynamic" and "Static". In static mode, it is 50%. In dynamic mode, you need to configure the phase and combine with dynamic DPA to achieve the duty cycle.
- Calculate: Calculates whether the current configuration is reasonable.

- In "General Mode", calculate the division/multipilication frequency and VCO parameters. If the actual frequency is different from the expected frequency, an Error prompt will pop up.
- In "Advanced Mode", calculate the static division/multipilication frequency and VCO parameters.If the calculated results are invalid, click "Calculate", an Error prompt will pop up. If valid, click "Calculate", "succeed" prompt will pop up.

3. Port Diagram
   The port diagram is based on the IP Core configuration. The input/output number is updated in real time based on the "Options" configuration, as shown in Figure 5-10.

**IP Generated Files**

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_pllo.v " file is a complete Verilog module to generate instantiated PLLO, and it is generated according to the IP configuration;
- "gowin_pllo_tmp.v" is the template file;
- "gowin_pllo.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5.4 DLLDLY

## 5.4.1 Primitive Introduction

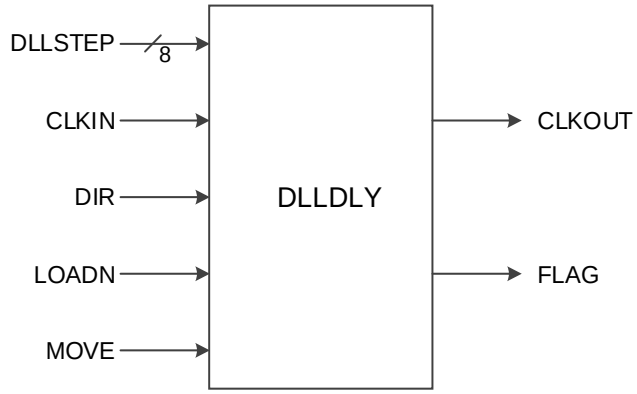
DLLDLY is the clock delay module that adjusts the input clock according to the DLLSTEP signal and outputs the delayed clock.

**Functional Description**

DLLDLY generates the corresponding phase delay based on DLLSTEP to get the delayed output based on CLKIN. The delay step size of DLLSTEP is about 30ps (GW1N(X) FPGAs) or about 18ps (GW2A(X) FPGAs).

## Port Diagram

**Figure 5-11 DLLDLY Port Diagram**




## Port Description

**Table 5-20 DLLDLY Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLKOUT | Output | Clock output signal |
| FLAG | Output | An output flag that represents under-flow or over-flow in dynamical delay adjustment |
| DLLSTEP[7:0] | Input | Delay step input signal |
| CLKIN | Input | Clock input signal |
| DIR | Input | Sets the direction of dynamic delay adjustment<br>0: Increases delay<br>1: Decreases delay |
| LOADN | Input | Controls the loading of DLLSTEP<br>0: Loads DLLSTEP<br>1: adjusts the delay dynamically |
| MOVE | Input | When MOVE is dynamically adjusted on falling edge, each pulse moves one delay step. |

## Parameter Description

**Table 5-21 DLLDLY Parameter Description**

| Name | Type | Value | Default | Description |
|---|---|---|---|---|
| DLL_INSEL | Integer | 1'b1 | 1'b1 | 1'b1: Normal mode, using DLLDLY delay module |
| DLY_SIGN | String | 1'b0,1'b1 | 1'b0 | Sets the sign of delay adjustment<br>1'b0:'+'<br>1'b1: '-' |
| DLY_ADJ | Integer | 0~255 | 0 | Delay adjustment setting<br>dly_sign=0<br>DLY_ADJ;<br>dly_sign=1 |

| Name | Type | Value | Default | Description |
|---|---|---|---|---|
| | | | | -256+ DLY_ADJ |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
DLLDLY dlldly_0 (
  .CLKIN(clkin),
  .DLLSTEP(step[7:0]),
  .DIR(dir),
  .LOADN(loadn),
  .MOVE(move),
  .CLKOUT(clkout),
   .FLAG(flag)
);
defparam dlldly_0.DLL_INSEL=1'b1;
defparam dlldly_0.DLY_SIGN=1'b1;
defparam dlldly_0.DLY_ADJ=0;
```

**Vhdl Instantiation:**

```
COMPONENT DLLDLY
        GENERIC(
                DLL_INSEL:bit:='0';
                DLY_SIGN:bit:='0';
                LY_ADJ:integer:=0
         );
            PORT(
                DLLSTEP:IN std_logic_vector(7 downto 0);
                CLKIN:IN std_logic;
                DIR,LOADN,MOVE:IN std_logic;
                CLKOUT:OUT std_logic;
                FLAG:OUT std_logic
              );
END COMPONENT;
uut:DLLDLY
```

```
GENERIC MAP(
                DLL_INSEL=>'1',
                DLY_SIGN=>'0',
                LY_ADJ=>0
 )
PORT MAP (
    DLLSTEP=>step,
    CLKIN=>clkin,
    DIR=>dir,
    LOADN=>loadn,
    MOVE=>move,
    CLKOUT=>clkout,
    FLAG=>flag
 );
```

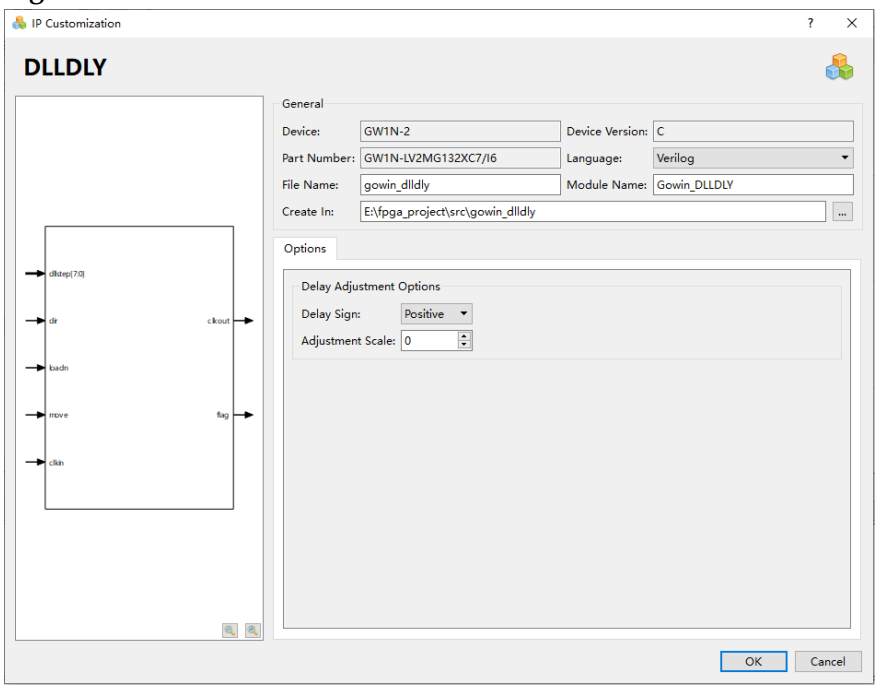
## 5.4.2 IP Generation

Click "DLLDLY" on the "IP Core Generator" interface and an overview of DLLDLY will be displayed on the right side of the interface.

### IP Configuration

Double-click on "DLLDLY", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 5-12.

**Figure 5-12 IP Customization of DLLDLY**



1. General
   The General configuration box is used to configure the generated IP design file. The DLLDLY General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to customize the IP, as shown in Figure 5-12.
   - Delay Sign: Sets the sign of delay.
   - Adjustment Scale: Delay adjustment.
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 5-12.

**IP Generated Files**

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_dlldly.v" file is a complete Verilog module to generate instantiated DLLDLY, and it is generated according to the IP configuration;
- "gowin_dlldly_tmp.v" is the template file;
- "gowin_dlldly.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5.5 CLKDIV

## 5.5.1 Primitive Introduction

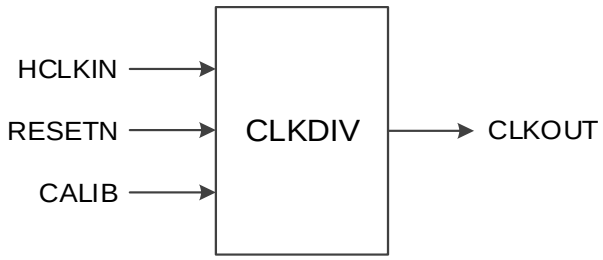
The CLKDIV is a clock frequency divider to adjust the clock frequency.

**Functional Description**

CLKDIV generates a divider clock that has the same phase with the input clock, which is used in the IO logic mode. GW1N-1S, GW1NS-4, GW1NS-4C, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-9, GW1N-9C, GW1NR-9, GW1NR-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B, GW1NR-2, and GW1NR-2B support 2/3.5/4/5/8 frequency division and other devices support 2/3.5/4/5 frequency division.

## Port Diagram

**Figure 5-13 CLKDIV Diagram**




## Port Description

**Table 5-22 CLKDIV Port Description**

| Port Name | I/O | Description |
|---|---|---|
| HCLKIN | Input | Clock input signal |
| RESETN | Input | Asynchronous reset signal, active-low |
| CALIB | Input | CALIB input signal, adjusting output clock |
| CLKOUT | Output | Clock output signal |

The above CALIB signal can be used in conjunction with CALIB in IOLOGIC, and the functions are as follows:

- For frequency divided by 2, the phase is adjusted every 2 falling edges with 180 degrees, and 2 adjustments are for one cycle.
- For frequency divided by 3.5, the phase is adjusted every 1 falling edge with about102.8 degrees, and 7 adjustments are for one cycle.
- For frequency divided by 4.5, the phase is adjusted every 2 falling edges with 90 degrees, and 4 adjustments are for one cycle.
- For frequency divided by 5, the phase is adjusted every 2 falling edges with about 72 degrees, and 5 adjustments are for one cycle.
- For frequency divided by 8, the phase is adjusted every 2 falling edges, with about 45 degrees, and 8 adjustments are for one cycle.

## Parameter Description

**Table 5-23 CLKDIV Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| DIV_MODE | 2, 3.5, 4, 5 ( 8) | 2 | Sets the clock frequency division coefficient |
| GSREN | "false", "true" | "false" | Enables global reset |

## Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
CLKDIV clkdiv_inst (
    .HCLKIN(hclkin),
    .RESETN(resetn),
    .CALIB(calib),
    .CLKOUT(clkout)
);
defparam clkdiv_inst.DIV_MODE="3.5";
defparam clkdiv_inst.GSREN="false";
```

**Vhdl Instantiation:**

```
COMPONENT CLKDIV
        GENERIC(
            DIV_MODE:STRING:="2";
            GSREN:STRING:="false"
            );
        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",
            GSREN=>"false"
            )
        PORT MAP (
            HCLKIN=>hclkin,
            RESETN=>resetn,
            CALIB=>calib,
            CLKOUT=>clkout
        );
```

## 5.5.2 IP Generation

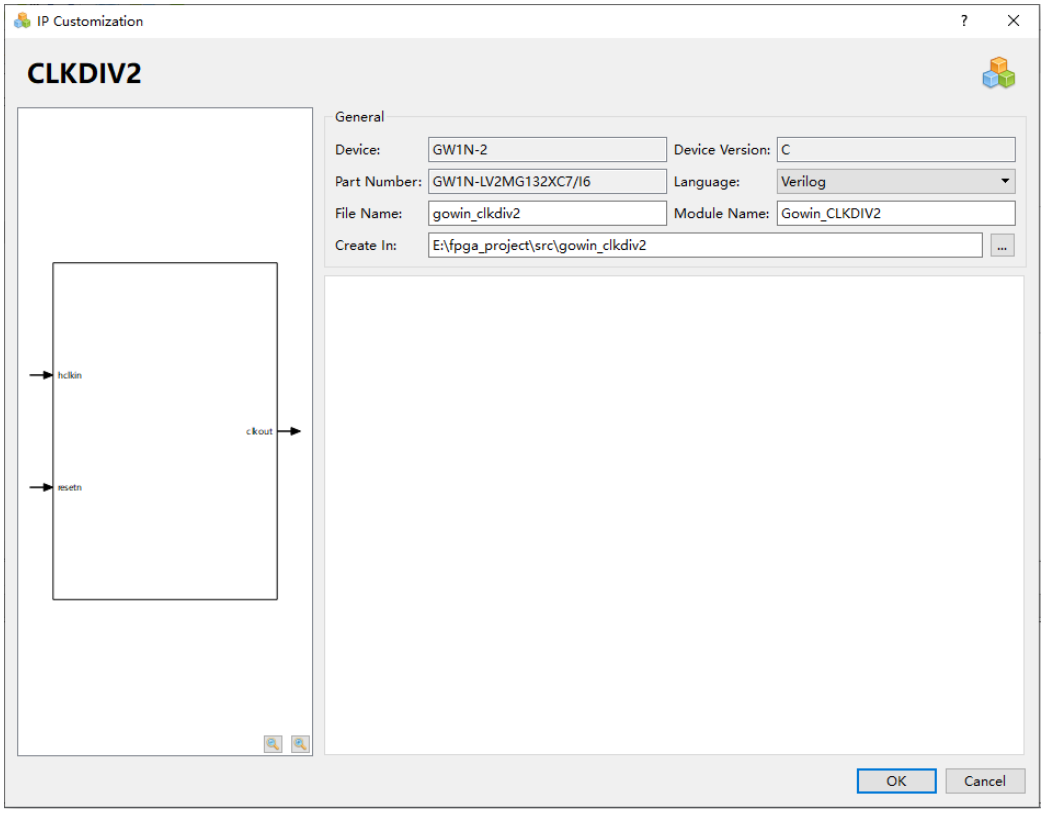
Click "CLKDIV" on the "IP Core Generator" interface and an overview

of CLKDIV will be displayed on the right side of the interface.

### IP Configuration

Double-click on "CLKDIV", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 5-14.

**Figure 5-14 IP Customization of CLKDIV**




1. General
   The General configuration box is used to configure the generated IP design file. The CLKDIV General configuration box is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to configure IP, as shown in Figure 5-14.
   - Division Factor: Division factor
   - Calibration: Enables/disables calibration
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 5-14.

### IP Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_clkdiv.v" file is a complete Verilog module to generate instantiated CLKDIV, and it is generated according to the IP configuration;
- "gowin_clkdiv_tmp.v" is the template file;
- "gowin_clkdiv.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5.6 CLKDIVG

## 5.6.1 Primitive Introduction

CLKDIVG is a clock divider to adjust the clock frequency

### Devices Supported

**Table 5-24 CLKDIVG Device Supported**

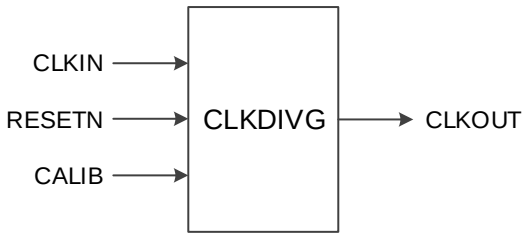
| Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
| | GW1NR | GW1NR-2, GW1NR-2B |
| Arora | GW2AN | GW2AN-18X, GW2AN-9X |

### Functional Description

The CLKDIVG is a clock divider module that generates a divider clock with the same phase as the input clock; there is only one CLKDIVG with a fixed position. Its input is from fixed IO and its functions are the same as those of CLKDIV.

### Port Diagram

**Figure 5-15 CLKDIVG Port Diagram**



### Port Description

**Table 5-25 CLKDIVG Port Description**

| Port | I/O | Description |
|---|---|---|
| CLKIN | Input | Clock input signal |
| RESETN | Input | Asynchronous reset signal, active-low. |
| CALIB | Input | CALIB input signal, adjusting output clock. |
| CLKOUT | Output | Clock output signal |

The above CALIB signal can be used in conjunction with CALIB in IOLOGIC; for the details, you can see the descriptions in 5.5 CLKDIV.

### Parameter Description

**Table 5-26 CLKDIVG Parameter Description**

| Parameter | Value | Default | Description |
|---|---|---|---|
| DIV_MODE | 2, 3.5, 4, 5, 8 | 2 | Sets the clock division factor |
| GSREN | "false", "true" | "false" | Enable global reset GSR |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation：**

```
CLKDIVG CLKDIVG_inst (
    .CLKIN(clkin),
    .RESETN(resetn),
    .CALIB(calib),
    .CLKOUT(clkout)
);
defparam CLKDIVG_inst.DIV_MODE="2";
defparam CLKDIVG_inst.GSREN="false";
```

**VHDL Instantiation：**

```
COMPONENT CLKDIVG
        GENERIC(
            DIV_MODE:STRING:="2";
            GSREN:STRING:="false"
            );
        PORT(
            CLKIN:IN std_logic;
```

```
                RESETN:IN std_logic;
                CALIB:IN std_logic;
                CLKOUT:OUT std_logic
                );
END CONPONENT;
    uut:CLKDIVG
        GENERIC MAP(
                DIV_MODE=>"2",
                GSREN=>"false"
                )
            PORT MAP(
                CLKIN=>clkin,
                RESETN=>resetn,
                CALIB=>calib,
                CLKOUT=>clkout
            );
```

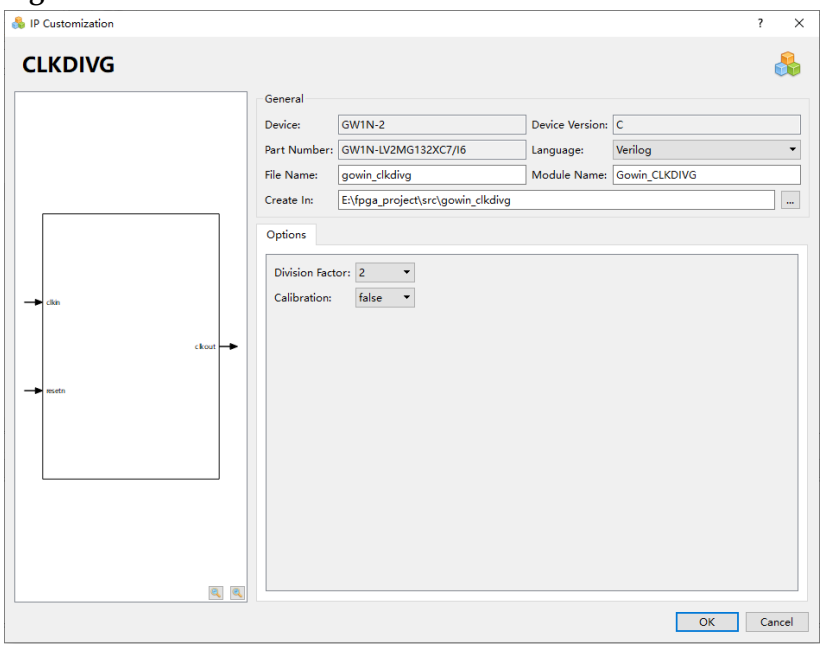
## 5.6.2 IP Generation

Click "CLKDIVG" on the "IP Core Generator" interface and an overview of CLKDIVG will be displayed on the right side of the interface.

### IP Configuration

Double-click on "CLKDIVG", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 5-16.

**Figure 5-16 IP Customization of CLKDIVG**



1. General
   The General configuration box is used to configure the generated IP design file.The CLKDIV General configuration is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to configure IP, as shown in Figure 5-16.
   - Division Factor: Division factor
   - Calibration: Enables/disables calibration
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 5-16.

**IP Generated Files**

After configuration, three files that are named after the "File Name" will be generated.

- " gowin_clkdivg.v " file is a complete Verilog module to generate instantiated CLKDIVG, and it is generated according to the IP configuration;
- "gowin_ clkdivg _tmp.v" is the template file;
- "gowin_ clkdivg.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 5.7 DQS

## 5.7.1 Primitive Introduction

DQS is a bi-directional data strobe circuit for DDR memory interface.

**Device Supported**

**Table 5-27 DQS Device Supported**

| Product Family | Series | Device |
|---|---|---|
| Arora | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AN | GW2AN-55C, GW2AN-18X, GW2AN-9X |
| | GW2AR | GW2AR-18,GW2AR-18C |
| | GW2ANR | GW2ANR-18C |

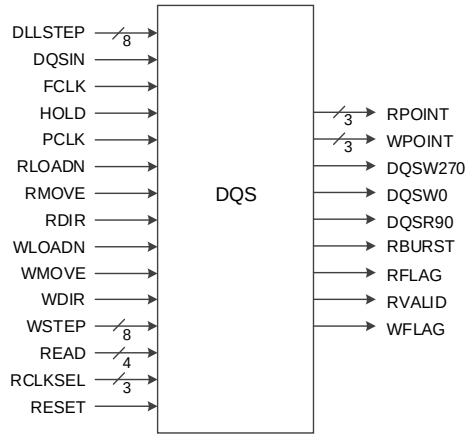
**Functional Description**

DQS is a key component of memory control IP, which is mainly used to adjust the phase relationship between DQSIN and DQSR90, DQSW0 and

DQSW270, and to complete write leveling and read calibration.

## Port Diagram

**Figure 5-17 DQS Port Diagram**




## Port Description

**Table 5-28 DQS Port Description**

| Port Name | I/O | Description |
|---|---|---|
| DLLSTEP[7:0] | input | DQS delay step control input |
| DQSIN | input | DQS input from IO PAD |
| FCLK | input | Fast clock comes from the output of two different FCLK clock trees. |
| HOLD | input | For DQS write, stop write signals to synchronize the output clock; For DQS read, reset FIFO counter |
| PCLK | input | Primary clock, from the PCLK clock tree |
| RDIR | input | Adjusts the delay direction of DDR read<br>"0" increases the delay.<br>"1" decreases the delay. |
| RLOADN | input | Resets the final delay step of DDR read, active-low. |
| RMOVE | input | Changes the delay step of DDR read on RMOVE falling edge, once per pulse. |
| WDIR | input | Adjusts the delay direction of DDR write<br>"0" increases the delay.<br>"1" decreases the delay. |
| WLOADN | input | Resets the final delay stepof DDR write, active-low. |
| WMOVE | input | Changes the delay step of DDR write on WMOVE falling edge, once per pulse. |
| WSTEP[7:0] | input | DDR write equalization delay control. |
| READ[3:0] | input | READ signal for DDR read mode |
| RCLKSEL[2:0] | input | Selects read clock source and polarity control |
| RESET | input | DQS reset input, active-high. |
| RPOINT[2:0] | output | FIFO read pointer works on RADDR in IOLOGIC or on user logic via routing. |

| Port Name | I/O | Description |
|---|---|---|
| WPOINT[2:0] | output | FIFO write pointer works on WADDR in IOLOGIC or on user logic via routing. |
| DQSW0 | output | PCLK/FCLK 0° phase shift output works on TCLK in IOLOGIC or on user logic via routing. |
| DQSW270 | output | PCLK/FCLK 270° phase shift output works on TCLK in IOLOGIC or on user logic via routing. |
| DQSR90 | output | DQSI 90° phase shift output works on TCLK in IOLOGIC or on user logic via routing. |
| RFLAG | output | An output flag that represents under-flow or over-flow in READ delay adjustment. |
| WFLAG | output | An output flag that represents under-flow or over-flow in WRITE delay adjustment. |
| RVALID | output | Data valid flag in READ mode |
| RBURST | output | READ burst detection output |

## Parameter Description

**Table 5-29 DQS Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FIFO_MODE_SEL | 1'b0 , 1'b1 | 1'b0 | FIFO mode selection<br>1'b0: DDR memory mode<br>1'b1: GDDR mode |
| RD_PNTR | "000", "001", "010", "011", "100", "101", "110", "111" | 3'b000 | FIFO read pointer setting |
| DQS_MODE | "X1", "X2_DDR2", "X2_DDR3", "X4", "X2_DDR3_EXT" | "X1" | DQS mode selection |
| HWL | "false", "true" | "false" | update0/1 timing relationship control<br>"False ": update1 is one cycle ahead of update0.<br>"True ": the timing of update1 and update0 are the same. |
| GSREN | "false", "true" | "false" | Enable global reset |

## Connection Rule

- The input DQSI of DQS comes from IO PAD.
- The output RPOINT of DQS can be connected to the RADDR of IOLOGIC and also can work on user logic.
- The output WPOINT of DQS can be connected to the WADDR of IOLOGIC and also can work on user logic.
- The output DQSR90 of DQS can be connected to the ICLK of IOLOGIC and also can work on user logic

- The output DQSW0/ DQSW270 of DQS can be connected to the TCLK of IOLOGIC and also can work on user logic.

### Primitive Instantiation

**Verilog Instantiation:**

```
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 Instantiation:**

```
COMPONENT DQS
```

```
        GENERIC(
                FIFO_MODE_SEL:bit:='0';
                RD_PNTR : bit_vector:="000";
                DQS_MODE:string:="X1";
                HWL:string:="false";
                GSREN : 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",
                        GSREN=>"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
    );
```

# 6 Crystal Oscillator Clock

## 6.1 Primitive Introduction

### 6.1.1 OSC

OSC, on-chip crystal oscillator.

**Device Supported**

**Table 6-1 OSC Device Supported**

| Product Family | Series | Device |
|---|---|---|
| Arora | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AN | GW2AN-55C |
| | GW2AR | GW2AR-18,GW2AR-18C |
| | GW2ANR | GW2ANR-18C |
| LittleBee® | GW1N | GW1N-4, GW1N-4B, GW1N-4D, GW1N-9, GW1N-9C |
| | GW1NR | GW1NR-4, GW1NR-4B, GW1NR-4D, GW1NR-9, GW1NR-9C |
| | GW1NRF | GW1NRF-4B |

**Functional Description**

GOWIN FPGA is embedded with a programmable on-chip oscillator, which provides a clock source for MSPI programming mode.The on-chip oscillator also provides a clock resource for user designs. Up to 64 clock frequencies can be obtained by setting the parameters.

The output clock frequency of the device can be calculated by the following formula:

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

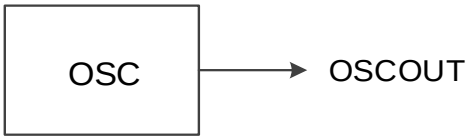
Among them, $f_{osc}$is the OSC oscillation frequency, the divisor FREQ_DIV is the configuration parameter, and the range is even numbers from 2 to 128.

**Note!**

f_osc takes different values according to different devices, 210MHz for GW1N-4, GW1NR-4, GW1N-4B, GW1NR-4B, GW1NRF-4B, GW1N-4D, GW1NR-4D devices, 250MHz for other supported devices.

## Port Diagram

**Figure 6-1 OSC Port Diagram**




## Port Description

**Table 6-2 OSC Port Description**

| Port Name | I/O | Description |
|---|---|---|
| OSCOUT | output | OSC output clock signal |

## Parameter Description

**Table 6-3 OSC Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FREQ_DIV | 2~128(even) | 100 | OSC frequency division coefficient setting |
| DEVICE | "GW1N-4", "GW1N-4B", "GW1N-4D" "GW1NR-4", "GW1NR-4B", "GW1NR-4D" "GW1NRF-4B", "GW1N-9", "GW1N-9C", "GW1NR-9", "GW1NR-9C", "GW2A-18", "GW2AR-18", "GW2A-55", "GW2A-55C", "GW2AN-55C" | GW1N-4 (GW1N series) GW2A-18 (GW2A series) | Devices selected |

## Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
OSC uut(
    .OSCOUT(oscout)
    );
```

```
defparam uut.FREQ_DIV=100;
defparam uut.DEVICE="GW2A-18";
```

**Vhdl Instantiation:**

```
COMPONENT OSC
    GENERIC(
            FREQ_DIV:integer:=100;
            DEVICE:string:="GW2A-18"
     );
    PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSC
  GENERIC MAP(
            FREQ_DIV=>100,
            DEVICE=>"GW2A-18"
   )
  PORT MAP(OSCOUT=>oscout);
```

## 6.1.2 OSCZ

OSCZ is an on-chip oscillator supporting dynamically shutdowning OSC.

### Device Supported

**Table 6-4 OSCZ Device Supported**

| Product Family | Series | Device |
|---|---|---|
| LittleBee® | GW1NS | GW1NS-4,GW1NS-4C |
| | GW1NSR | GW1NSR-4,GW1NSR-4C |
| | GW1NSER | GW1NSER-4C |
| | GW1NZ | GW1NZ-1, GW1NZ-1C |

### Functional Description

The GW1NZ series of FPGA products are embedded with a programmable on-chip oscillator, clock accuracy up to ±5%, and support to dynamically turn on/off OSC. The on-chip oscillator provides a clock source for MSPI programming and user designs. Up to 64 clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

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

$f_{oscz}$ is the OSC oscillation frequency; FREQ_DIV is the division configuration parameter with a range of 2~128, and it supports even numbers only.

**Note!**

$f_{oscz}$ takes different values according to different devices, 260MHz for C7 speed grade of GW1NS-4/GW1NS-4C/GW1NSR-4/GW1NSR-4CGW1NSER-4C devices , 250MHz for other speed grades of other supported devices.

## Port Diagram

**Figure 6-2 OSCZ Port Diagram**


## Port Description

**Table 6-5 OSCZ Port Description**

| Port Name | I/O | Description |
|---|---|---|
| OSCEN | input | OSC enable signal |
| OSCOUT | output | OSC clock output signal |

## Parameter Description

**Table 6-6 OSCZ Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FREQ_DIV | 2~128(even) | 100 | OSC frequency division coefficient setting |
| S_RATE | "SLOW", "FAST" | "SLOW" | For C7 of GWINS-4 devices, it needs to be set to"FAST"; for others, it is set to "SLOW". |

## Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

```
OSCZ uut(
    .OSCOUT(oscout),
    .OSCEN(oscen)
    );
defparam uut.FREQ_DIV=100;
defparam uut.S_RATE="SLOW";
```

**Vhdl Instantiation:**

```
COMPONENT OSCZ
        GENERIC(
                FREQ_DIV:integer:=100,
```

```
                    S_RATE:string:="SLOW"
        );
        PORT(
            OSCOUT:OUT STD_LOGIC;
            OSCEN:IN std_logic
          );
END COMPONENT;
uut:OSCZ
    GENERIC MAP(
                FREQ_DIV=>100,
                S_RATE=>"SLOW"
     )
    PORT MAP (
        OSCOUT=>oscout,
            OSCEN(oscen)
        );
```

### 6.1.3 OSCH

OSCH, on-chip crystal

#### Device Supported

**Table 6-7 OSCH Device Supported**

| Product Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-1,GW1N-1S |
| | GW1NR | GW1NR-1 |

#### Functional Description

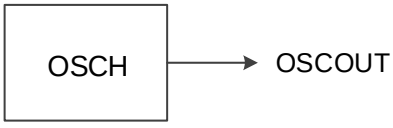
The on-chip oscillator provides a clock source for MSPI programming and user designs. Up to 64 clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

$f_{CLKOUT} = 240MHz/\mathrm{FREQ\_DIV};$

"FREQ_DIV" is the configuration parameter and has a range of 2~128. It supports even numbers only.

#### Port Diagram

**Figure 6-3 OSCH Port Diagram**



### Port Description

**Table 6-8 OSCH Port Description**

| Port Name | I/O | Description |
|---|---|---|
| OSCOUT | output | OSC clock output signal |

### Parameter Description

**Table 6-9 OSCH Parameter Description**

| Name | Value | Default | Description |
|---|---|---|---|
| FREQ_DIV | 2~128(even) | 100 | OSC frequency division coefficient setting |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

## 6.1.4 OSCO

OSCO is an on-chip crystal supporting dynamically shutdowning OSC and regulator power supply.

### Device Supported

**Table 6-10 OSCO Device Supported**

| Family | Series | Device |
|---|---|---|
| LittleBee® | GW1N | GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
| | GW1NR | GW1NR-2, GW1NR-2B |

### Functional Description

Gowin FPGA products are embedded with a programmable on-chip oscillator with the clock accuracy up to ±5%, and support dynamic on/off OSC and regulator power supply. The on-chip oscillator provides a clock source for MSPI programming and user designs. Up to 64 kinds of clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

$$f_{CLKOUT} = 250MHz/\text{FREQ\_DIV}$$

"FREQ_DIV" is the configuration parameter and has a range of 2~128, supporting even numbers only.

### Port Diagram

**Figure 6-4 OSCO Port Diagram**


### Port Description

**Table 6-11 OSCO Port Description**

| Port | I/O | Description |
|---|---|---|
| OSCEN | input | OSC enable signal, active-high. |
| OSCOUT | output | OSC clock output signal |

### Parameter Description

**Table 6-12 OSCO Parameter Description**

| Parameter | Value | Dedault | Description |
|---|---|---|---|
| FREQ_DIV | 2~128(even) | 100 | OSC division factor setting |
| REGULATOR_EN | 1'b0, 1'b1 | 1'b0 | 1'b0: OSCO is powered by VCC.<br>1'b1: OSCO is powered by Regulator. |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation：**

```
OSCO uut(
    .OSCOUT(oscout),
    .OSCEN(oscen)
    );
defparam uut.FREQ_DIV=100;
defparam uut. REGULATOR_EN =1'b0;
```

**VHDL Instantiation：**

```
COMPONENT OSCO
        GENERIC(
                FREQ_DIV:integer:=100;
                REGULATOR_EN : bit := '0'
         );
        PORT(
            OSCOUT:OUT STD_LOGIC;
            OSCEN:IN std_logic
          );
END COMPONENT;
uut:OSCO
    GENERIC MAP(
                 FREQ_DIV=>100,
                 REGULATOR_EN=> '0'
     )
    PORT MAP(
            OSCOUT=>oscout,
                OSCEN(oscen)
            );
```

## 6.1.5 OSCW

OSCW, on-chip crystal.

### Device Supported

**Table 6-13 OSCW Device Supported**

| Family | Series | Device |
|---|---|---|
| Arora | GW2AN | GW2AN-18X, GW2AN-9X |

### Functional Description

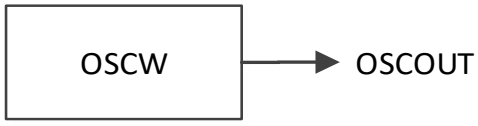
Gowin FPGA products are embedded with a programmable on-chip oscillator with the clock accuracy up to ±5%. The on-chip oscillator provides a clock source for MSPI programming and user design. Up to 64 kinds of clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

$$f_{CLKOUT} = 200MHz/\text{FREQ\_DIV}$$

"FREQ_DIV" is the configuration parameter and has a range of 2~128, supporting even numbers only.

### Port Diagram

**Figure 6-5 OSCW Port Diagran**




### Port Description

**Table 6-14 OSCO Port Description**

| Port | I/O | Description |
|---|---|---|
| OSCOUT | output | OSC clock output signal |

### Parameter Description

**Table 6-15 OSCW Parameter Description**

| Parameter | Value | Dedault | Description |
|---|---|---|---|
| FREQ_DIV | 2~128(even) | 80 | OSC division factor setting |

### Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool.

**Verilog Instantiation：**

```
OSCW uut(
    .OSCOUT(oscout)
    );
defparam uut.FREQ_DIV=80;
```

**VHDL Instantiation:**

```
COMPONENT OSCW
        GENERIC(
                FREQ_DIV:integer:=100
         );
        PORT(
```

```
            OSCOUT:OUT STD_LOGIC
        );
END COMPONENT;
uut:OSCW
    GENERIC MAP(
                FREQ_DIV=>80
     )
    PORT MAP(
            OSCOUT=>oscout
     );
```

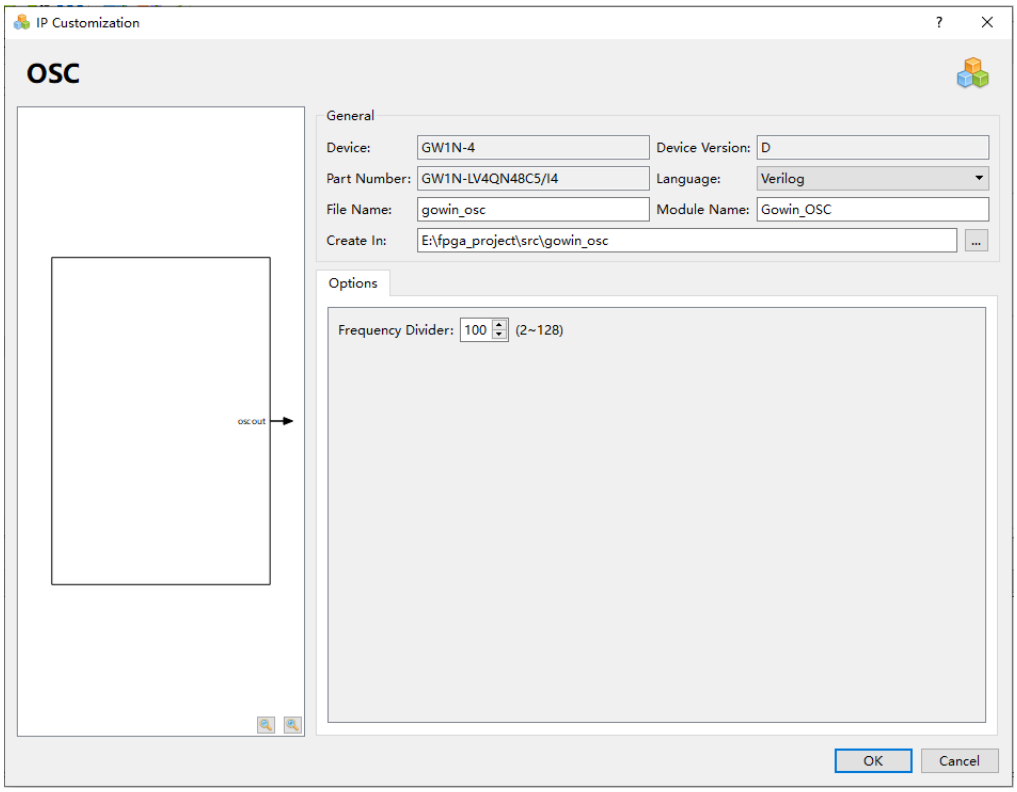
# 6.2 IP Generation

Click "OSC" on the "IP Core Generator" interface and an overview of related information about OSC will be displayed on the right side of the interface.

### IP Configuration

Double-click on "OSC", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 6-6.

**Figure 6-6 IP Customization of OSC**




1. General
   The File configuration box is used to configure the generated IP design file.The OSC General configuration box is similar to that of DQCE. For the details, please refer to the General description in 3.1.2 IP Generation.
2. Options
   The Options configuration box is used to configure IP, as shown Figure 6-6. Frequency Divider: Selects any even number from 2 to 128.
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 6-6.

### IP Generated Files

After configuration, three files that are named after the "File Name" will be generated.

- "gowin_osc.v" file is a complete Verilog module to generate instantiated OSC, and it is generated according to the IP configuration;
- "gowin_osc_tmp.v" is the template file;
- "gowin_osc.ipc" file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.


GOWIN
PROGRAMMING FOR THE FUTURE