Arora V Clock

User Guide

UG306-1.0E, 04/20/2023

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
04/20/2023	1.0E	Initial version published.

Contents

Contents iv
List of Figures vi
List of Tables viii
1 About This Guide 1
1.1 Purpose 1
1.2 Related Documents 1
1.3 Terminology and Abbreviations 1
1.4 Support and Feedback 2
2 Overview 3
2.1 Global Clock 3
2.2 HCLK 3
3 Global Clock 4
3.1 DCE 4
3.1.1 Primitive Introduction 4
3.1.2 IP Generation 5
3.2 DCS 7
3.2.1 Primitive Introduction 7
3.2.2 IP Generation 10
4 HCLK 13
4.1 DHCE 13
4.1.1 Primitive Introduction 13
4.1.2 IP Generation 14
4.2 CLKDIV2 15
4.2.1 Primitive Introduction 15
4.2.2 IP Generation 16
4.3 CLKDIV 18
4.3.1 Primitive Introduction 18
4.3.2 IP Generation 20
4.4 DLLDLY 21

4.4.1 Primitive Introduction 21
4.4.2 IP Generation 24
5 System Clock 27
5.1 PLL 27
5.1.1 Primitive Introduction 27
5.1.2 IP Generation 59
5.2 PLLA 63
5.2.1 Primitive Introduction 63
5.2.2 IP Generation 88
5.3 DQS 90
5.4 DDRDLL 95
5.4.1 Primitive Introduction 95
5.4.2 IP Generation 98
6 Crystal Oscillator Clock 100
6.1 OSC 100
6.1.1 Primitive Introduction 100
6.1.2 IP Generation 101
6.2 OSCA 103
6.2.1 Primitive Introduction 103
6.2.2 IP Generation 104

List of Figures

Figure 3-1 DCE Diagram 4
Figure 3-2 IP Customization of DCE 6
Figure 3-3 DCS Port Diagram 7
Figure 3-4 Timing Diagram of Non-Glitchless 9
Figure 3-5 RISING Timing Diagram in DCS Mode 10
Figure 3-6 FALLING Timing Diagram in DCS Mode 10
Figure 3-7 CLK0_GND Timing Diagram in DCS Mode 10
Figure 3-8 CLK0_VCC Timing Diagram in DCS Mode 10
Figure 3-9 IP Customization of DCS 11
Figure 4-1 DHCE Port Diagram 13
Figure 4-2 IP Customization of DHCE 14
Figure 4-3 CLKDIV2 Port Diagram 15
Figure 4-4 IP Customization of CLKDIV2 17
Figure 4-5 CLKDIV Diagram 18
Figure 4-6 IP Customization of CLKDIV 20
Figure 4-7 DLLDLY Port Diagram 22
Figure 4-8 IP Customization of DLLDLY 25
Figure 5-1 PLL Structure Diagram 28
Figure 5-2 PLL Port Diagram 30
Figure 5-3 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b1, One Step) 47
Figure 5-4 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b0, One Step) 47
Figure 5-5 IP Customization of PLL 60
Figure 5-6 PLLA Structure Diagram 63
Figure 5-7 PLLA Port Diagram 65
Figure 5-8 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b1, One Step) 78
Figure 5-9 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b0, One Step) 79
Figure 5-10 IP Customization of PLLA 89
Figure 5-11 DQS Port Diagram 90
Figure 5-12 DDRDLL Port Diagram 95
Figure 5-13 IP Customization of DDRDLL 98
Figure 6-1 OSC Port Diagram 100
Figure 6-2 IP Customization of OSC 102

Figure 6-3 OSC Port Diagram 103
Figure 6-4 IP Customization of OSCA 105

List of Tables

Table 1-1 Terminology and Abbreviations 1
Table 3-1 DCE Port Description 4
Table 3-2 DCS Port Description 8
Table 3-3 DCS Parameter Description 8
Table 4-1 DHCE Port Description 13
Table 4-2 CLKDIV2 Port Description 16
Table 4-3 CLKDIV Port Description 18
Table 4-4 CLKDIV Parameter Description 19
Table 4-5 DLLDLY Port Description 22
Table 4-6 DLLDLY Parameter Description 22
Table 5-1 PLL Device Supported 27
Table 5-2 PLL Port Description 30
Table 5-3 rPLL Parameter Description 33
Table 5-4 IDIV Dynamic and Static Correspondence 43
Table 5-5 FBDIV Dynamic and Static Correspondence 43
Table 5-6 ODIV0 Integer Division Contrast 44
Table 5-7 ODIV0 Fraction Division Contrast 44
Table 5-8 MDIV Integer Division Contrast 45
Table 5-9 MDIV Fraction Division Contrast 45
Table 5-10 Fine Adjustment Contrast of PLL Duty Cycle 47
Table 5-11 PLLA Device Supported 63
Table 5-12 PLLA Port Description 65
Table 5-13 PLLA Parameter Description 67
Table 5-14 IDIV Static Correspondence 74
Table 5-15 FBDIV Static Correspondence 75
Table 5-16 ODIV0 Integer Division Contrast 75
Table 5-17 ODIV0 Fraction Division Contrast 76
Table 5-18 MDIV Integer Division Contrast 76
Table 5-19 MDIV Fraction Division Contrast 77
Table 5-20 Fine Adjustment Contrast of PLLA Duty Cycle 78
Table 5-21 DQS Port Description 90
Table 5-22 DQS Parameter Description 91

Table 5-23 DDRDLL Port Description 95
Table 5-24 DDRDLL Parameter Description 95
Table 6-1 OSC Device Supported 100
Table 6-2 OSC Port Description 101
Table 6-3 OSC Parameter Description 101
Table 6-4 OSCA Device Supported 103
Table 6-5 OSCA Port Description 103
Table 6-6 OSCA Parameter Description 104

1 About This Guide

1.1 Purpose

This manual provides descriptions of the function, primitive and usage of GOWINSEMI Arora V clock resources.

1.2 Related Documents

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

- DS981, GW5AT series of FPGA Products Data Sheet
- DS1103, GW5A series of FPGA Products Data Sheet
- DS1104, GW5AST series of FPGA Products Data Sheet
- SUG100, Gowin Software User Guide

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	Description
CLKDIV	Clock Divider	–
CLKDIV2	Clock Divider 2	High-speed Clock Divider
DCE	Dynamic Clock Enable	–
DCS	Dynamic Clock Selector	–
DHCE	Dynamic HCLK Clock Enable	–
DLLDLY	DLL Delay	–
DQS	Bidirectional Data Strobe Circuit for DDR Memory	–
GCLK	Global Clock	–
HCLK	High-speed Clock	–
LW	Long Wire	–
OSC	Oscillator	–

Terminology and Abbreviations	Meaning	Description
PCLK	Primary Clock	–
PLL	Phase-locked Loop	–
SSC	Spread Spectrum Clocking	–

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 Arora V FPGA products, including the dedicated clock input 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 are critical for high-performance applications of FPGA. GOWINSEMI Arora V FPGA products provide the global clock network (GCLK), including primary clock (PCLK) and long wire (LW); GCLK connects directly to all the resources. Besides GCLK, clock resources such as PLL, high-speed clock (HCLK), and bidirectional data strobe circuit for DDR memory (DQS) are also provided.

2.1 Global Clock

GCLK includes PCLK and LW; PCLKs can connect directly to all the resources. LW can be used as a control wire to provide clock enable (CE) and set/reset (SET/RESET) signals to the DFF on the one hand; on the other hand, LW can also be used as a logic wire, that is, it can be used as a general data signal.

2.2 HCLK

HCLK is the high-speed clock in the GOWINSEMI Arora V FPGA products. It offers low jitter and low deviation, which can support high-speed data transfer. It is suitable for source synchronous data transfer protocols; and one bank supports four HCLKs.

3 Global Clock

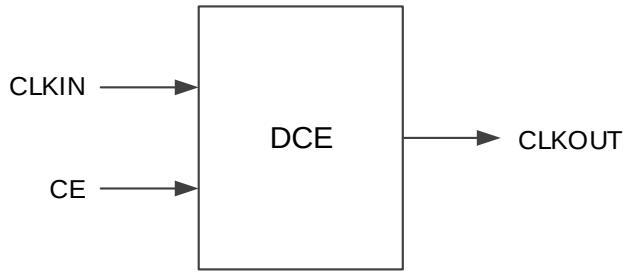
3.1 DCE

3.1.1 Primitive Introduction

Dynamic Clock Enable (DCE) allows the internal logic to dynamically enable or disable the GCLK network. When the GCLK clock network is disabled, all logic driven by this clock is no longer slewed, thus reducing the total device power.

Port Diagram

Figure 3-1 DCE Diagram



Port Description

Table 3-1 DCE 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.

Verilog Instantiation:

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

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

VHDL Instantiation：

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

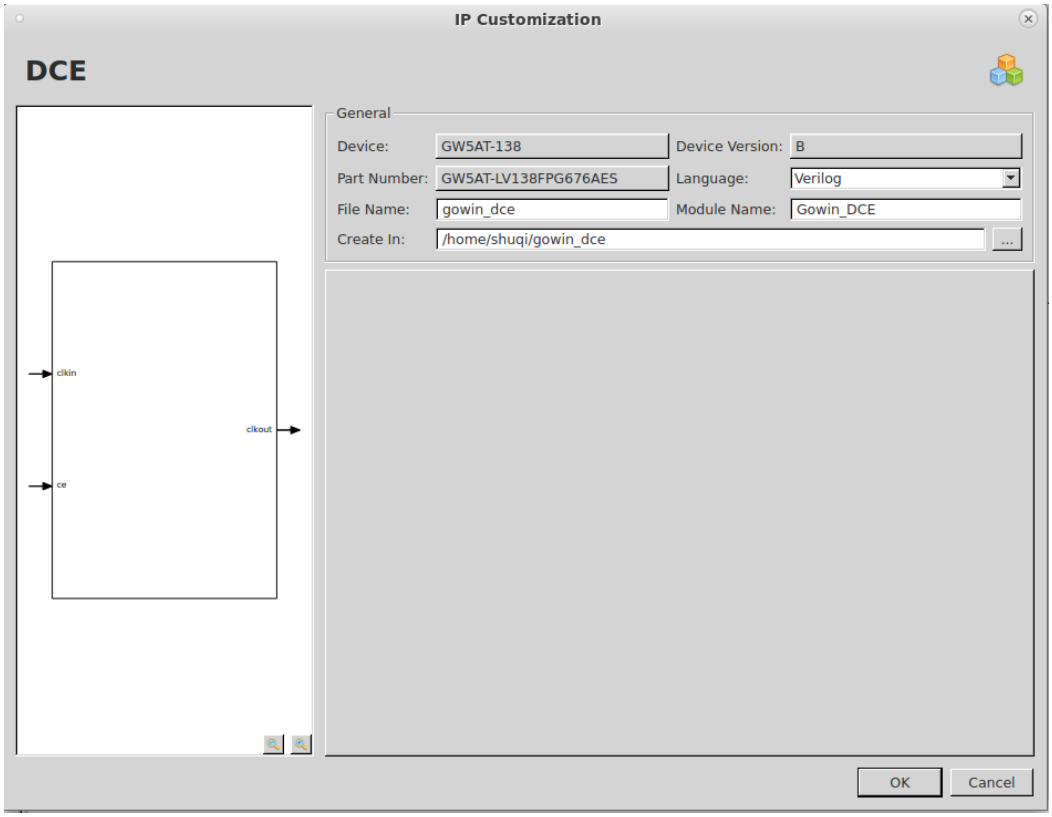
3.1.2 IP Generation

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

IP Configuration

Double-click on "DCE", 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 DCE



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_dce.v" file is a complete Verilog module to generate instantiated DCE, and it is generated according to the IP configuration.
- "gowin_dce_tmp.v" is the template file.
- "gowin_dce.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

DCS means dynamic clock selector. CLKOUT can be dynamically switched among the four clock inputs.

Functional Description

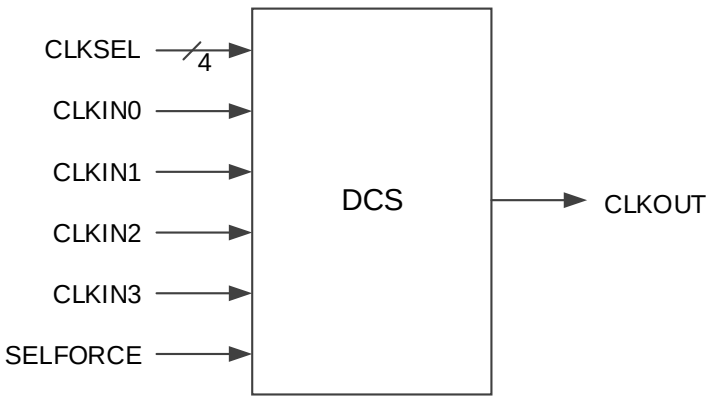
DCS has two clock switching modes, namely "Non-glitchless" and "Glitchless".

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

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 setting DCS_MODE.

Port Diagram

Figure 3-3 DCS Port Diagram



Port Description

Table 3-2 DCS Port Description

Port	I/O	Description
CLKIN0	Input	Clock input signal 0
CLKIN1	Input	Clock input signal 1
CLKIN2	Input	Clock Iinput signal 2
CLKIN3	Input	Clock input signal 3
CLKSEL	Input	Clock selected signal
SELFORCE	Input	Mandatory mode selected 0: Glitchless mode 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"	Set DCS mode

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

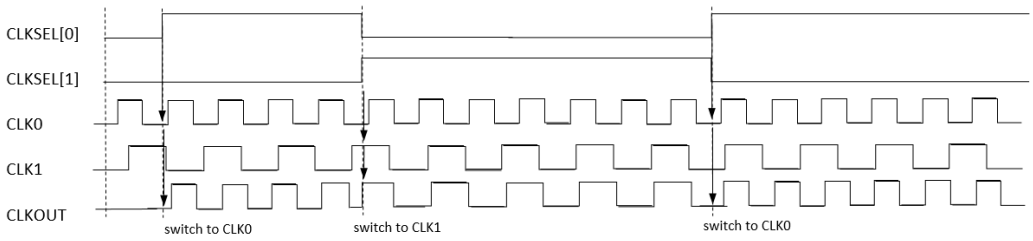
Vhdl Instantiation：

```
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 CLKIN3~CLKIN0 with the same conversion timing, active-high.

Figure 3-4 Timing Diagram of Non-Glitchless



The timing of Glitchless mode is shown from Figure 3-5 to Figure 3-8 CLKSEL [3]~CLKSEL [0] are corresponding to CLKIN3~CLKIN0 with the same conversion timing.

Figure 3-5 RISING Timing Diagram in DCS Mode

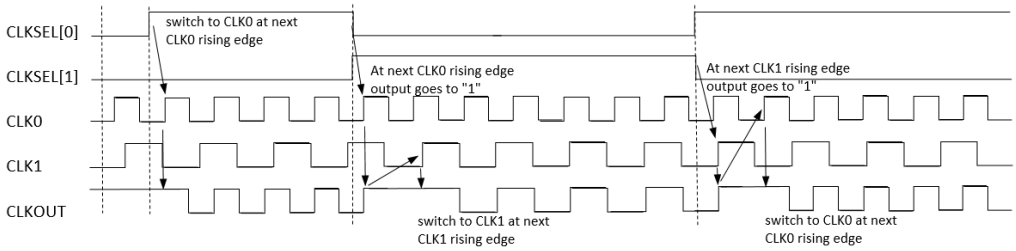


Figure 3-6 FALLING Timing Diagram in DCS Mode

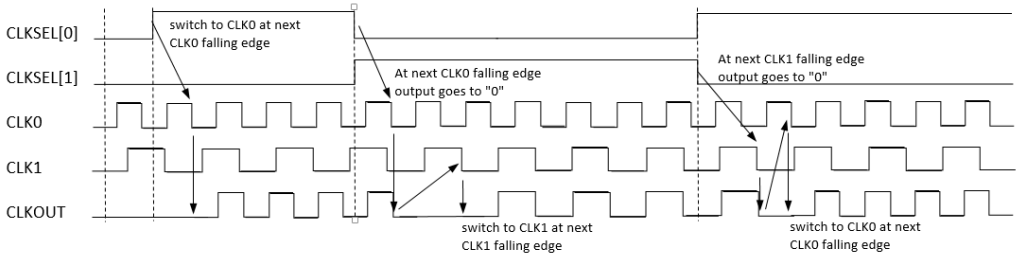


Figure 3-7 CLK0_GND Timing Diagram in DCS Mode

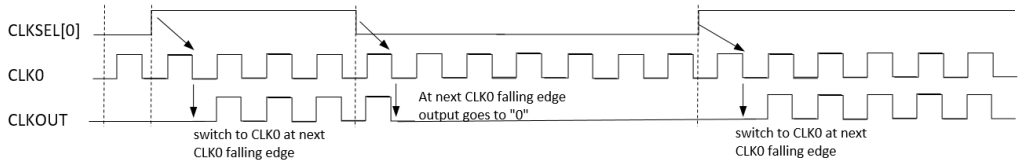
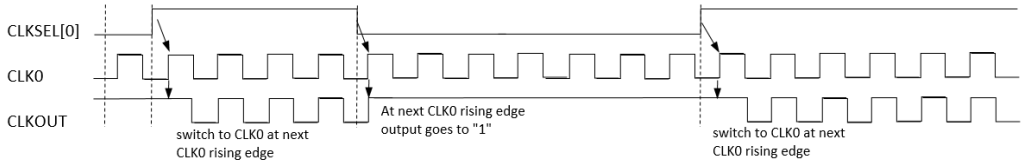


Figure 3-8 CLK0_VCC Timing Diagram in DCS Mode



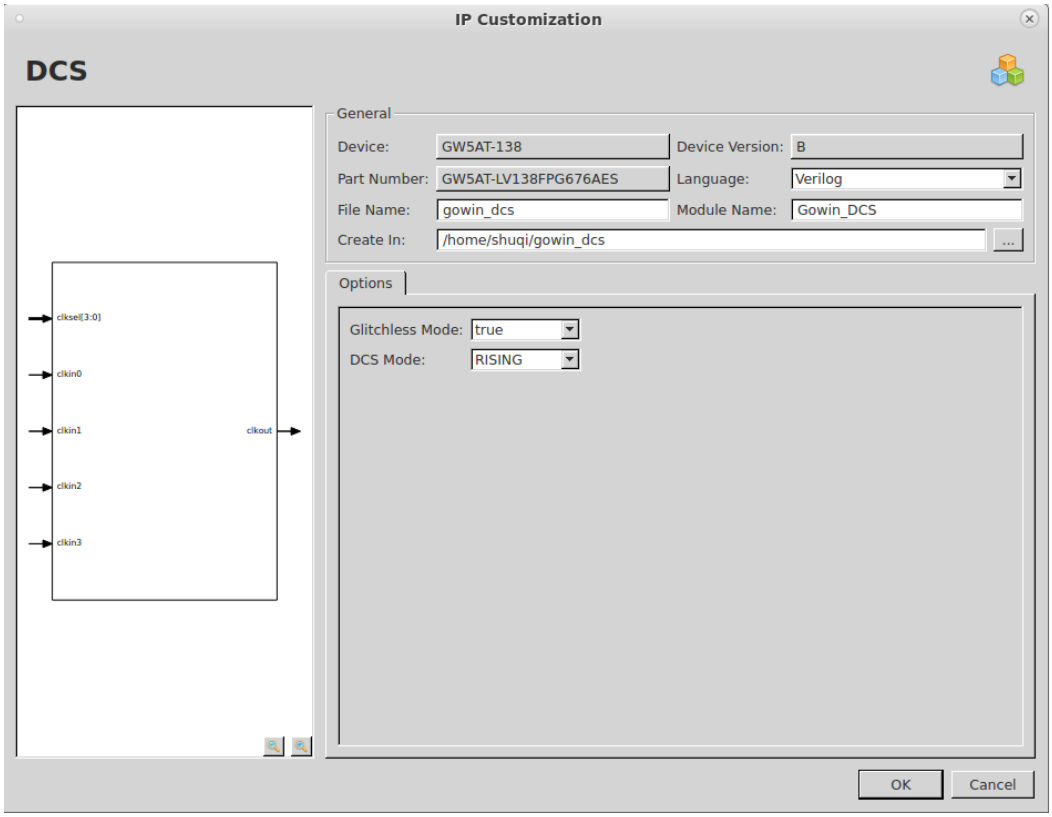
3.2.2 IP Generation

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 DCE. 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: Enable/disable Glitchless
   - DCS Mode: Set 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, it will generate three files that are named after the "File Name".

- "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 DHCE

4.1.1 Primitive Introduction

You can use DHCE to dynamically turn on/off HCLK signal.

Port Diagram

Figure 4-1 DHCE Port Diagram

Port Description

Table 4-1 DHCE Port Description

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

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

Vhdl Instantiation：

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

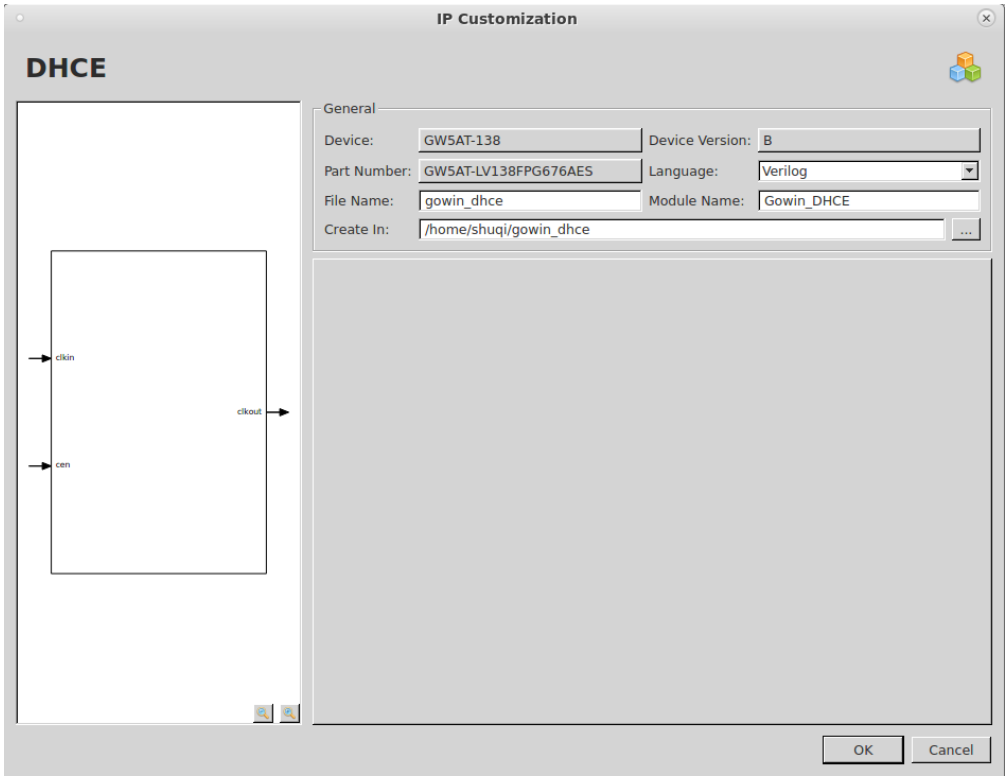
4.1.2 IP Generation

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

IP Configuration

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

Figure 4-2 IP Customization of DHCE



1. General
   The General configuration box is used to configure the generated IP design file. DHCE General configuration is similar to that of DCE. 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 4-2.

Generated Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_dhce.v" file is a complete Verilog module to generate instantiated DHCE, and it is generated according to the IP configuration.
- "gowin_dhce_tmp.v" is the template file.
- "gowin_dhce.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 CLKDIV2

4.2.1 Primitive Introduction

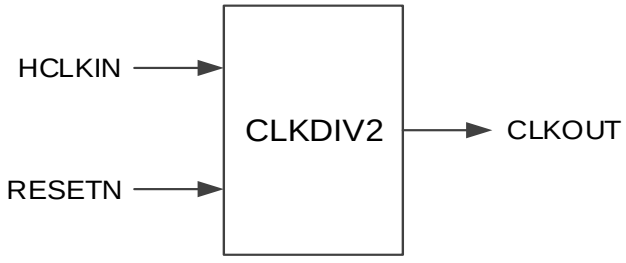
The CLKDIV2 is a clock divider. It realizes a divide-by-two clock. The output of CLKDIV2 can only drive FCLK of IOLOGIC, CLKIN and CLKFB of PLL, FCLK of DQS, HCLKIN of CLKDIV, and DDRDLL of CLKIN.

Functional Description

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

Port Diagram

Figure 4-3 CLKDIV2 Port Diagram



Port Description

Table 4-2 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

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

VHDL Instantiation:

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

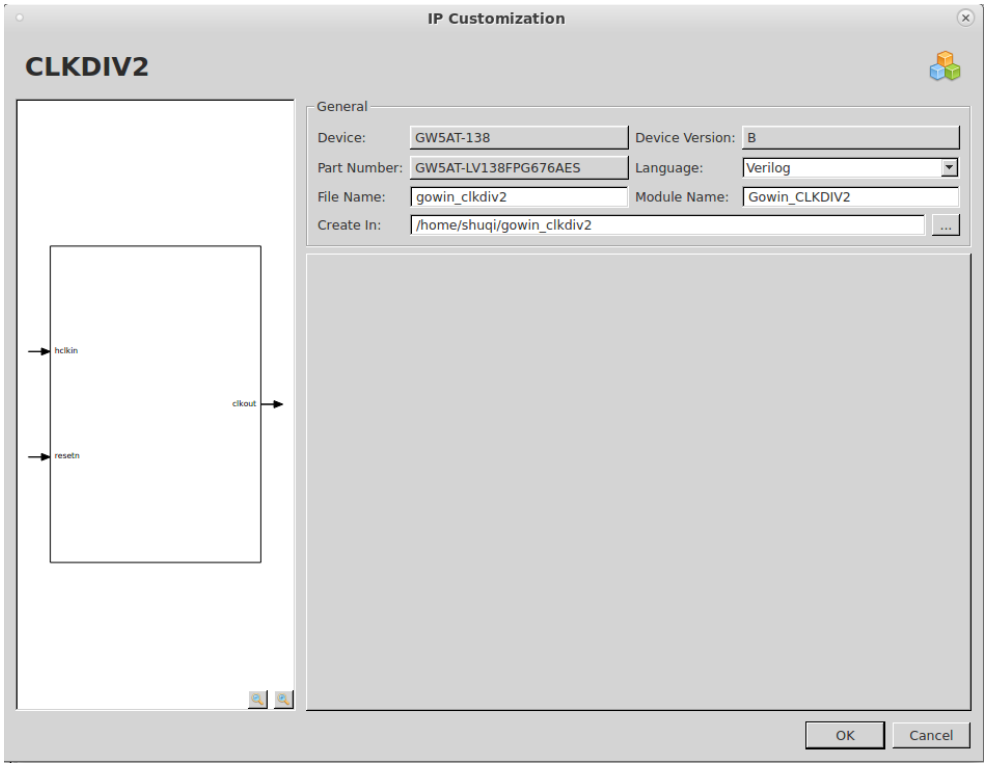
4.2.2 IP Generation

Click "CLKDIV2" on the "IP Core Generator" interface and an overview of related information about 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 4-4.

Figure 4-4 IP Customization of CLKDIV2



1. General
   The General configuration box is used to configure the generated IP design file. CLKDIV2 General configuration is similar to that of DCE. 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 4-4.

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.

4.3 CLKDIV

4.3.1 Primitive Introduction

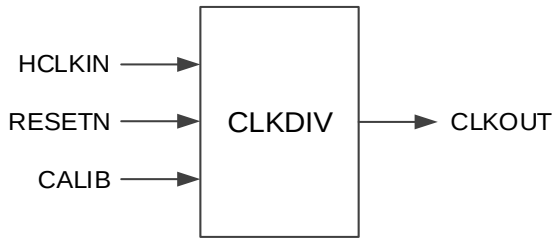
CLKDIV is a clock frequency divider to adjust the clock frequency.

Functional Description

CLKDIV generates a divider clock that has the same phase as that of input clock, which is used in the IO logic.

Port Diagram

Figure 4-5 CLKDIV Diagram



Port Description

Table 4-3 CLKDIV Port Description

Port Name	I/O	Description
HCLKIN	Input	Clock input signal
RESETN	Input	Asynchronous reset signal, active-low; it does not work for DIV_MODE of 1.
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, the phase is adjusted every 2 falling edges with 120 degrees, and 3 adjustments are for one cycle.
- For frequency divided by 3.5, the phase is adjusted every 1 falling edge with about 102.8 degrees, and 7 adjustments are for one cycle.
- For frequency divided by 4, 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 6, the phase is adjusted every 2 falling edges with 60 degrees, and 6 adjustments are for one cycle.
- For frequency divided by 7, the phase is adjusted every 2 falling edges

with 51.4 degrees, and 7 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.
- For frequency divided by 1, the adjustment is invalid.

Parameter Description

Table 4-4 CLKDIV Parameter Description

Name	Value	Default	Description
DIV_MODE	1,2,3,3.5,4,5,6,7,8	2	Set the clock frequency division coefficient

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

VHDL Instantiation：

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

```
PORT MAP(
        HCLKIN=>hclkin,
        RESETN=>resetn,
        CALIB=>calib,
        CLKOUT=>clkout
);
```

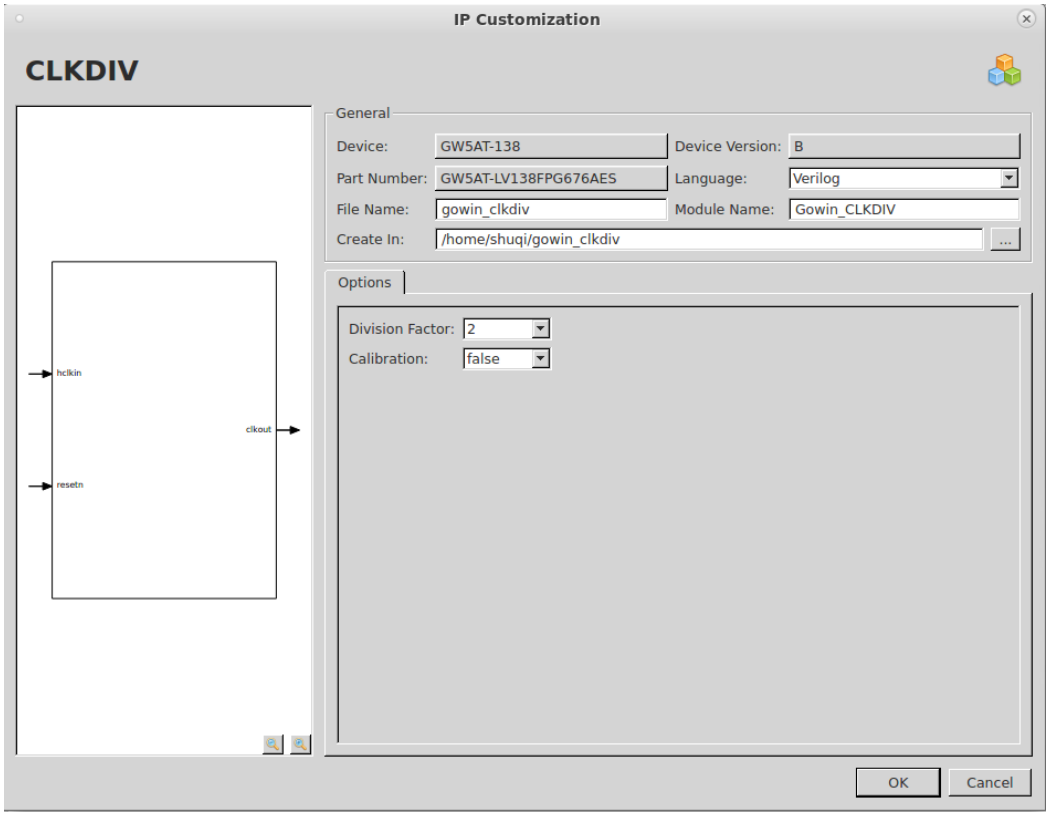
4.3.2 IP Generation

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 4-6.

Figure 4-6 IP Customization of CLKDIV



1. General
   The Genera configuration box is used to configure the generated IP design file. The CLKDIV2 General configuration is similar to that of DCE. 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.

- Division Factor: Select a division factor
- Calibration: Enable/ disable calibration

3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 4-6.

IP Generated Files

After configuration, it will generate three files that are named after the "File Name".

- "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.

4.4 DLLDLY

4.4.1 Primitive Introduction

DLLDLY is the clock delay module that adjusts the input clock according to CSTEP or DLLSTEP signal and gets delay adjustment output of the clock.

Functional Description

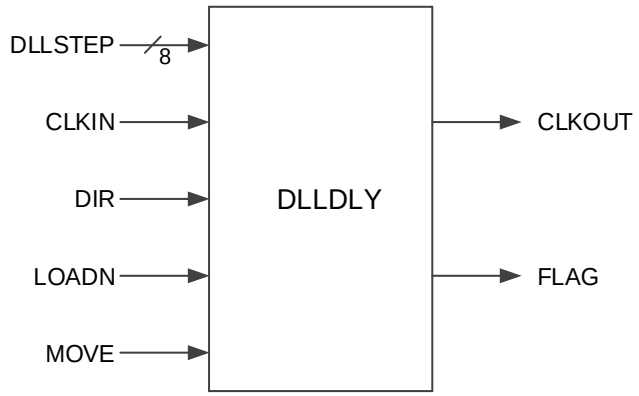
DLLDLY generates the corresponding phase delay according to CSTEP or DLLSTEP to get the delay output based on CLKIN. Delay adjustment supports static, dynamic, and adaptive modes.

Note!

GW5A-25 devices do not support the adaptive mode.

Port Diagram

Figure 4-7 DLLDLY Port Diagram



Port Description

Table 4-5 DLLDLY Port Description

Port Name	I/O	Description
CLKOUT	Output	Clock output signal
FLAG	Output	An output flag of adaptive mode that represents under-flow or over-flow for dynamical delay adjustment.
DLLSTEP[7:0]	Input	Delay step input signal,from DDRDLL.
CLKIN	Input	Clock input signal
CSTEP[7:0]	Input	Delay step input signal from CIU wiring
LOADN	Input	Control the loading of delay step ● 0: Load delay step ● 1: Adjust the delay dynamically
MOVE	Input	In the adaptive mode, the delay is dynamically adjusted on falling edge of MOVE, and each pulse moves one delay step.

Parameter Description

Table 4-6 DLLDLY Parameter Description

Name	Type	Value	Default	Description
DLY_SIGN	Binary	1'b0,1'b1	1'b0	Set the sign of delay adjustment 1'b0:'+' 1'b1: '-'
DLY_ADJ	Integer	0~255	0	Set delay adjustment DLY_SIGN =0 DLY_ADJ; DLY_SIGN =1 - DLY_ADJ

Name	Type	Value	Default	Description
STEP_SEL	Binary	1’b0,1’b1	1’b0	1’b0:DLLSTEP 1’b1:CSTEP
ADAPT_EN	String	"FALSE", "TRUE"	"FALSE"	Adaptive mode enable
DYN_DLY_EN	String	"FALSE", "TRUE"	"FALSE"	Dynamic mode enable

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

VHDL 例化：

```
COMPONENT DLLDLY
        GENERIC(
                DLY_SIGN:bit:='0';
                DLY_ADJ:integer:=0;
                DYN_DLY_EN : string := "FALSE" ;
                ADAPT_EN : string := "FALSE" ;
                STEP_SEL:bit:='0'
         );
            PORT(
                DLLSTEP:IN std_logic_vector(7 downto 0);
```

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

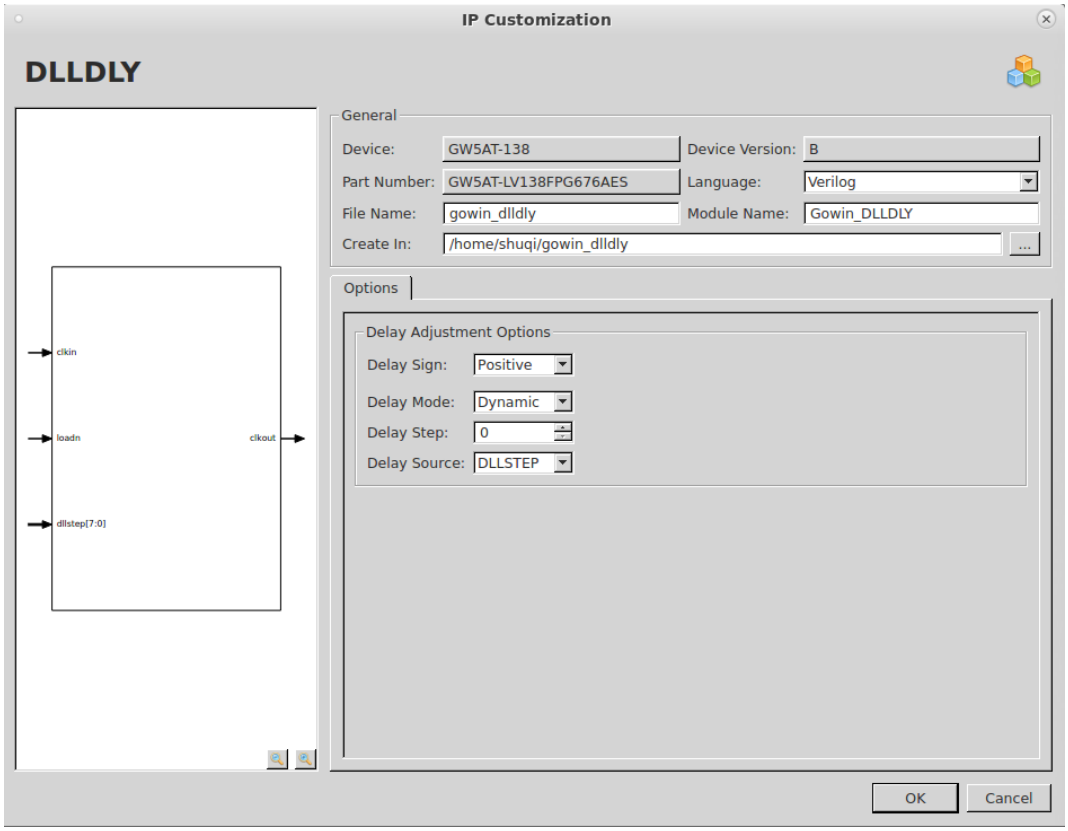
4.4.2 IP Generation

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 4-8.

Figure 4-8 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 DCE. 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 4-8.
   - Delay Sign: Set delay adjustment sign
   - Delay Mode: Set delay mode
   - Delay Step: Set delay step
   - Delay Source: Set delay source
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 4-8.

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 System Clock

5.1 PLL

5.1.1 Primitive Introduction

Arora V FPGA products provide phase-locked loop (PLL) and support seven clock outputs, and each clock supports independent adjustment of clock frequency, phase and duty cycle based on a given reference input clock.

Device Supported

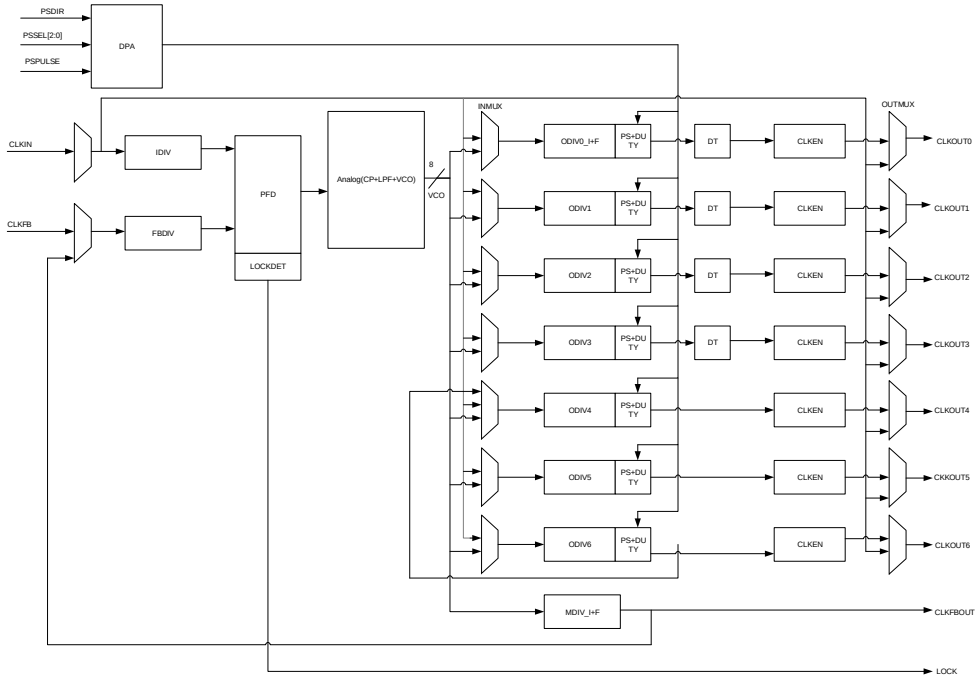
Table 5-1 PLL Device Supported

Product Family	Series	Device
Arora	GW5AT	GW5AT-138, GW5AT-138 B Version
	GW5AST	GW5AST-138 B Version

Functional Description

The PLL structure diagram is as shown in Figure 5-1.

Figure 5-1 PLL Structure Diagram



Based on the given input clock, PLL supports the adjustments of clock phase, duty cycle, frequency to generate output clocks with different phases, duty cycles, and frequencies. CLKOUT0 and CLKFBOUT support 1/8 fraction frequency adjustment; CLKOUT0~CLKOUT3 support dynamically and statically fine trimming duty cycle. PLL also supports internal cascading from CLKOUT6 to CLKOUT4, spread spectrum clocking (SSC), and clock de-skewing to achieve CLKIN and CLKOUT alignment.

To get the correct clock output, the input clock frequency must be set according to the frequency range described in the *FPGA Product Datasheet*.

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

1. Fpfd=Fclkin/IDIV
2. Fclkfb=Fpfd*FBDIV
3. Depending on different feedbacks, VCO frequency formula is different:
   - Internal feedback: Fvco=Fclkfb*MDIV
   - External feedback: Fvco=Fclkfb*MDIV ---- CLKFBOUT Feedback to CLKFB; Fvco=Fclkfb*ODIVx ---- CLKOUTx Feedback to CLKFB
4. Fclkfbout= Fvco/MDIV
5. Depending on the mode selected of INMUX and OUTMUX, the formula of the output frequency of CLKOUT channel is different:
   - VCO in mode (INMUX from VCO): Fclkoutx= Fvco/ODIVx
   - Bypass in mode (INMUX from CLKIN): Fclkoutx=Fclkin/ODIVx

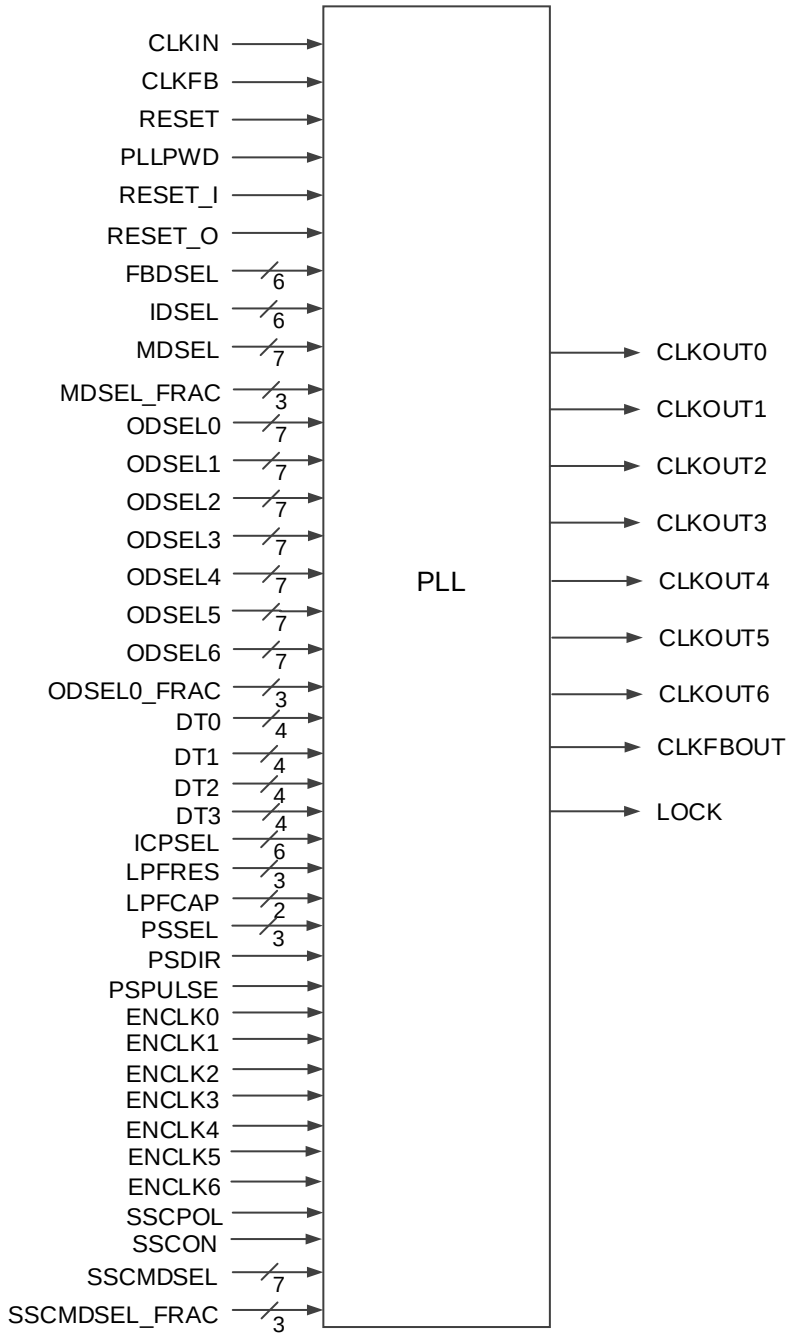
- Bypass out mode (OUTMUX from CLKIN): Fclkoutx=Fclkin
- CAS mode (channel 4 only): Fclkout4=Fclkout6[1]/ODIV4

Note!

- f_{CLKIN} is the input clock CLKIN frequency.
- Fclkoutx: x=0~6, the output clock frequency of channels 0~6.
- F_{clkfb} is the frequency of feebback input clock CLKFB.
- f_{PFD} is the PFD phase discrimination frequency.
- IDIV, FBDIV, MDIV, ODIVx (x=0~6) are the frequency division coefficients of different frequency dividers, which can be adjusted to get the clock signal with expected frequency.
- [1] Fclkout6 refers to the output clock of channel 6 ODIV6.

Port Diagram

Figure 5-2 PLL Port Diagram



Port Description

Table 5-2 PLL Port Description

Port	I/O	Description
CLKIN	Input	Reference clock input
CLKFB	Input	Feedback clock input
RESET	Input	All reset signal of PLL; reset digital circuit, active-high.

Port	I/O	Description
PLLPWD	Input	PLL power down signal; power down analog circuits, active-high.
RESET_I	Input	PLL global reset with IDIV, active-high; generally used for internal test.
RESET_O	Input	Reset ODIV and some digital circuits, active-high, generally used for internal test.
FBDSEL[5:0]	Input	Dynamic control of FBDIV value with range 0~63, the actual value of FBDIV is 64-FBDSEL.
IDSEL[5:0]	Input	Dynamic control of IDIV value with range 0~63, the actual value is 64-IDSEL.
MDSEL[6:0]	Input	Dynamical control of MDIV integer value with range 0~126, the actual value is 128-MDSEL, i.e. MDIV value range is 2~128.
MDSEL_FRAC[2:0]	Input	Dynamic control of MDIV fraction value, applicable for MDIV integer range 2~127, fraction value 1/8.
ODSEL0[6:0]	Input	Dynamic control of ODIV0 integer value with range 0~127, the actual value is 128-ODSEL0.
ODSEL0_FRAC[2:0]	Input	Dynamic control of ODIV0 fraction value, applicable for ODIV0 integer range 2~127, fraction value 1/8.
ODSEL1[6:0]	Input	Dynamical control of ODIV1 value with range 0~127, the actual value is 128-ODSEL1.
ODSEL2[6:0]	Input	Dynamical control of ODIV2 value with range 0~127, the actual value is 128-ODSEL2.
ODSEL3[6:0]	Input	Dynamical control of ODIV3 value with range 0~127, the actual value is 128-ODSEL3.
ODSEL4[6:0]	Input	Dynamical control of ODIV4 value with range 0~127, the actual value is 128-ODSEL4.
ODSEL5[6:0]	Input	Dynamical control of ODIV5 value with range 0~127, the actual value is 128-ODSEL5.
ODSEL6[6:0]	Input	Dynamical control of ODIV6 value with range 0~127, the actual value is 128-ODSEL6.
DT0[3:0]	Input	Dynamically trim CLKOUT0 duty cycle
DT1[3:0]	Input	Dynamically trim CLKOUT1 duty cycle
DT2[3:0]	Input	Dynamically trim CLKOUT2 duty cycle
DT3[3:0]	Input	Dynamically trim CLKOUT3 duty cycle
ICPSEL[5:0]	Input	Dynamic control of ICP current; the current increases as the value increases, and the current is minimum when the value is 0.
LPFRES[2:0]	Input	Dynamic control of LPFRES size with the value from small to large R0~R7; R0 corresponds to the largest bandwidth, R7 corresponds to the smallest bandwidth.
LPFCAP[1:0]	Input	Dynamically control of LPFCAP size with the value from small to large C0~C2.
PSDIR	Input	Dynamic control of phase shift direction
PSSEL[2:0]	Input	Dynamic control of channel selected for phase shift
PSPULSE	Input	Dynamic control of clock pulse of phase shift

Port	I/O	Description
ENCLK0	Input	Dynamic control of channel 0 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT0_EN="TRUE".
ENCLK1	Input	Dynamic control of channel 1 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT1_EN="TRUE".
ENCLK2	Input	Dynamic control of channel 2 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT2_EN="TRUE".
ENCLK3	Input	Dynamic control of channel 3 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT3_EN="TRUE".
ENCLK4	Input	Dynamic control of channel 4 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT4_EN="TRUE".
ENCLK5	Input	Dynamic control of channel 5 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT5_EN="TRUE".
ENCLK6	Input	Dynamic control of channel 6 clock output enable; if you want to use dynamic enable, you need to set static parameter CLKOUT6_EN="TRUE".
SSCPOL	Input	Optional configuration to avoid timing conflicts: ● 0：CLK Rising Edge ● 1：CLK Falling Edge
SSCON	Input	Dynamic control of SSC enable
SSCMDSEL[6:0]	Input	Dynamic control of SSC MDIV integer value, the recommended SSC MDIV actual value range is 16~24 (corresponding Fpfd of 50M), 32~48 (corresponding Fpfd of 25M), and the actual value is 128-SSCMDSEL.
SSCMDSEL_FRAC[2:0]	Input	Dynamic control of SSC MDIV fraction value, and the fraction value is 1/8.
CLKOUT0	Output	Channel 0 clock output (default)
CLKOUT1	Output	Channel 1 clock output
CLKOUT2	Output	Channel 2 clock output
CLKOUT3	Output	Channel 3 clock output
CLKOUT4	Output	Channel 4 clock output
CLKOUT5	Output	Channel 5 clock output
CLKOUT6	Output	Channel 6 clock output
CLKFBOUT	Output	Feedback clock output
LOCK	Output	PLL lock indication： ● 1: Lock ● 0: Unlock

Parameter Description

Table 5-3 rPLL Parameter Description

Name	Type	Value	Default	Description
FCLKIN	string	"10"~"400"	"100.0"	Reference clock frequency (MHz)
IDIV_SEL	integer	1~64	1	IDIV frequency division coefficient static setting, the actual value is 1~64.
DYN_IDIV_SEL	string	"TRUE", "FALSE"	"FALSE"	IDIV frequency division coefficient static control parameter or dynamic control signal selected: ● FALSE: Static, namely select the parameter IDIV_SEL. ● TRUE: Dynamic, namely select signal IDSEL.
FBDIV_SEL	integer	1~64	1	FBDIV frequency division coefficient static setting, the actual value is 1~64.
DYN_FBDIV_SEL	string	"TRUE", "FALSE"	"FALSE"	FBDIV frequency division coefficient static control parameter or dynamic control signal seleced. ● FALSE: Static, namely select the parameter FBDIV_SEL. ● TRUE: Dynamic, namely select signal FBDSEL.
ODIV0_SEL	integer	1~128	8	Integer static setting for ODIV0 frequency division coefficient
ODIV0_FRAC_SEL	integer	0~7	0	Fraction static setting for ODIV0 frequency division coefficient
DYN_ODIV0_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV0 frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ODIV0_SEL and ODIV0_FRAC_SEL. ● TRUE: Dynamic, namely select ODSEL0 and ODSEL0_FRAC signals.
ODIV1_SEL	integer	1~128	8	ODIV1 frequency division coefficient static setting
DYN_ODIV1_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV1 frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ODIV1_SEL. ● TRUE: Dynamic, namely select ODSEL1 signal.
ODIV2_SEL	integer	1~128	8	ODIV2 frequency division coefficient static setting
DYN_ODIV2_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV2 frequency division coefficient static control parameter or dynamic

Name	Type	Value	Default	Description
				control signal selected. ● FALSE: Static, namely select the parameter ODIV2_SEL. ● TRUE: Dynamic, namely select ODSEL2 signal.
ODIV3_SEL	integer	1~128	8	ODIV3 frequency division coefficient static setting
DYN_ODIV3_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV3 frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ODIV3_SEL. ● TRUE: Dynamic, namely select ODSEL3 signal.
ODIV4_SEL	integer	1~128	8	ODIV4 frequency division coefficient static setting
DYN_ODIV4_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV4 frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ODIV4_SEL. ● TRUE: Dynamic, namely select ODSEL4 signal.
ODIV5_SEL	integer	1~128	8	ODIV5 frequency division coefficient static setting
DYN_ODIV5_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV5 frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ODIV5_SEL. ● TRUE: Dynamic, namely select ODSEL5 signal.
ODIV6_SEL	integer	1~128	8	ODIV6 frequency division coefficient static setting
DYN_ODIV6_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV6 frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ODIV6_SEL. ● TRUE: Dynamic, namely select ODSEL6 signal.
DYN_MDIV_SEL	string	"TRUE", "FALSE"	“FALSE"	MDIV frequency division coefficient static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters MDIV_SEL and MDSEL_FRAC. ● TRUE: Dynamic, namely select ODSEL6 signal.
MDIV_SEL	integer	2~128	8	Integer static setting for MDIV

Name	Type	Value	Default	Description
				frequency division coefficient
MDIV_FRAC_SEL	string	0~7	0	Fraction static setting for MDIV frequency division coefficient
CLKOUT0_EN	string	"TRUE", "FALSE"	"TRUE"	Channel 0 clock output enable
CLKOUT1_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 1 clock output enable
CLKOUT2_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 2 clock output enable
CLKOUT3_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 3 clock output enable
CLKOUT4_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 4 clock output enable
CLKOUT5_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 5 clock output enable
CLKOUT6_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 6 clock output enable
DYN_DT0_SEL	string	"TRUE", "FALSE"	"FALSE"	Channel 0 duty cycle trimming static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT0_DT_DIR & CLKOUT0_DT_STEP. ● TRUE: Dynamic, namely select DT0 signal.
DYN_DT1_SEL	string	"TRUE", "FALSE"	"FALSE"	Channel 1 duty cycle trimming static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT1_DT_DIR & CLKOUT1_DT_STEP. ● TRUE: Dynamic, namely select DT1 signal.
DYN_DT2_SEL	string	"TRUE", "FALSE"	"FALSE"	Channel 2 duty cycle trimming static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT2_DT_DIR & CLKOUT2_DT_STEP. ● TRUE: Dynamic, namely select DT2 signal.
DYN_DT3_SEL	string	"TRUE", "FALSE"	"FALSE"	Channel 3 duty cycle trimming static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT3_DT_DIR &

Name	Type	Value	Default	Description
				CLKOUT3_DT_STEP. ● TRUE: Dynamic, namely select DT3 signal.
CLKOUT0_DT_DIR	binary	1'b1, 1'b0	1'b1	Static trimming direction of channel 0 duty cycle ● 1'b1: Duty cycle increase (+), adjust falling edge according to the rising edge alignment.. ● 1'b0: Duty cycle decrease (-), adjust rising edge according to the falling edge alignment.
CLKOUT1_DT_DIR	binary	1'b1, 1'b0	1'b1	Static trimming direction of channel 1 duty cycle ● 1'b1: Duty cycle increase (+); adjust falling edge according to the rising edge alignment. ● 1'b0: Duty cycle decrease (-); adjust rising edge according to the falling edge alignment.
CLKOUT2_DT_DIR	binary	1'b1, 1'b0	1'b1	Static trimming direction of channel 2 duty cycle ● 1'b1: Duty cycle increase (+); adjust falling edge according to the rising edge alignment. ● 1'b0: Duty cycle decrease (-); adjust rising edge according to the falling edge alignment.
CLKOUT3_DT_STEP	integer	0,1,2,4	0	Static trimming direction of channel 3 duty cycle ● 1'b1: Duty cycle increase (+); adjust falling edge according to the rising edge alignment. ● 1'b0: Duty cycle decrease (-); adjust rising edge according to the falling edge alignment.
CLKOUT0_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 0 duty cycle, 50ps of each step.
CLKOUT1_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 1 duty cycle, 50ps of each step.
CLKOUT2_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 2 duty cycle, 50ps of each step.
CLKOUT3_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 3 duty cycle, 50ps of each step.
CLK0_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV0 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK0_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 0 output clock source selected: ● 1'b0: From ODIV0 output

Name	Type	Value	Default	Description
				● 1'b1: Output clock bypass from CLKIN
CLK1_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV1 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK1_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 1 output clock source selected: ● 1'b0: From ODIV1 output ● 1'b1: Output clock bypass from CLKIN
CLK2_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV2 input clock source selected: ● 1'b0: From VCO output ● 2'b1: Output clock bypass from CLKIN
CLK2_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 2 output clock source selected: ● 1'b0: From ODIV2 output ● 1'b1: Output clock bypass from CLKIN
CLK3_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV3 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK3_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 3 output clock source selected: ● 1'b0: From ODIV3 output ● 1'b1: Output clock bypass from CLKIN
CLK4_IN_SEL	binary	2'b00, 2'b01,2’b10	2'b00	ODIV4 input clock source selected: ● 2'b00: From VCO output ● 2'b01: Cascade from CLKCAS_6 ● 2'b10: Output clock bypass from CLKIN
CLK4_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 4 output clock source selected: ● 1'b0: From ODIV4 output ● 1'b1: Output clock bypass from CLKIN
CLK5_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV5 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK5_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 5 output clock source selected: ● 1'b0: From ODIV5 output ● 1'b1: Output clock bypass from

Name	Type	Value	Default	Description
				CLKIN
CLKFB_SEL	string	"INTERNAL", "EXTERNAL"	"INTERNAL"	CLKFB source selected: ● INTERNAL: From internal CLKOUT feedback ● EXTERNAL: From external signal feedback
DYN_DPA_EN	string	"TRUE", "FALSE"	"FALSE"	Dynamic phase shift adjustment enable
CLKOUT0_PE_COARSE	integer	0~127	0	Static setting of channel 0 coarse phase shift
CLKOUT0_PE_FINE	integer	0~7	0	Static setting of channel 0 fine phase shift
CLKOUT1_PE_COARSE	integer	0~127	0	Static setting of channel 1 coarse phase shift
CLKOUT1_PE_FINE	integer	0~7	0	Static setting of channel 1 fine phase shift
CLKOUT2_PE_COARSE	integer	0~127	0	Static setting of channel 2 coarse phase shift
CLKOUT2_PE_FINE	integer	0~7	0	Static setting of channel 2 fine phase shift
CLKOUT3_PE_COARSE	integer	0~127	0	Static setting of channel 3 coarse phase shift
CLKOUT3_PE_FINE	integer	0~7	0	Static setting of channel 3 fine phase shift
CLKOUT4_PE_COARSE	integer	0~127	0	Static setting of channel 4 coarse phase shift
CLKOUT4_PE_FINE	integer	0~7	0	Static setting of channel 4 fine phase shift
CLKOUT5_PE_COARSE	integer	0~127	0	Static setting of channel 5 coarse phase shift
CLKOUT5_PE_FINE	integer	0~7	0	Static setting of channel 5 fine phase shift
CLKOUT6_PE_COARSE	integer	0~127	0	Static setting of channel 6 coarse phase shift
CLKOUT6_PE_FINE	integer	0~7	0	Static setting of channel 6 fine phase shift
DYN_PE0_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 0 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT0_PE_COARSE and CLKOUT0_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE1_SE	string	“TRUE","FALSE	“FALSE"	Channel 1 phase shift static control

Name	Type	Value	Default	Description
L		"		parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT1_PE_COARSE and CLKOUT1_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE2_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 2 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT2_PE_COARSE and CLKOUT2_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE3_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 3 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT3_PE_COARSE and CLKOUT3_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE4_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 4 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT4_PE_COARSE and CLKOUT4_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE5_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 5 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT5_PE_COARSE and CLKOUT5_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with

Name	Type	Value	Default	Description
				DYN_DPA_EN="TRUE".
DYN_PE6_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 6 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT6_PE_COARSE and CLKOUT6_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DE0_EN	string	“TRUE","FALSE"	“FALSE"	Channel 0 (ODIV0=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE0_SEL="TRUE", set CLKOUT0_PE_COARSE and CLKOUT0_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE1_EN	string	“TRUE","FALSE"	“FALSE"	Channel 1 (ODIV1=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE1_SEL="TRUE", set CLKOUT1_PE_COARSE and CLKOUT1_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE2_EN	string	“TRUE","FALSE"	“FALSE"	Channel 2 (ODIV2=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE2_SEL="TRUE", set CLKOUT2_PE_COARSE and CLKOUT2_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE3_EN	string	“TRUE","FALSE	“FALSE"	Channel 3 (ODIV3=2~128) duty

Name	Type	Value	Default	Description
		"		cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE3_SEL="TRUE", set CLKOUT3_PE_COARSE and CLKOUT3_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE4_EN	string	“TRUE","FALSE"	“FALSE"	Channel 4 (ODIV4=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE4_SEL="TRUE", set CLKOUT4_PE_COARSE and CLKOUT4_PE_FIN as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE5_EN	string	“TRUE","FALSE"	“FALSE"	Channel 5 (ODIV5=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE5_SEL="TRUE", set CLKOUT5_PE_COARSE and CLKOUT5_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE6_EN	string	“TRUE","FALSE"	“FALSE"	Channel 6 (ODIV6=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE6_SEL="TRUE", set CLKOUT6_PE_COARSE and CLKOUT6_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
RESET_I_EN	string	"TRUE",	"FALSE"	Enable the dynamic signal

Name	Type	Value	Default	Description
		"FALSE"		RESET_I. If you need to use the RESET_I port, you need to set this parameter to TRUE.
RESET_O_EN	string	"TRUE", "FALSE"	"FALSE"	Enable the dynamic signal RESET_O. if you need to use the RESET_O port, you need to set this parameter to TRUE.
DYN_ICP_SEL	string	"TRUE", "FALSE"	"FALSE"	ICPSEL static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ICP_SEL. ● TRUE: Dynamic, namely select dynamic signal ICPSEL.
ICP_SEL	binary	6'bXXXXXX, 6'b000000~6'b111111	6'bXXXXXX	ICP current static setting: ● 6'bXXXXXX: Indicates that Gowin Software will automatically calculate and set this parameter. ● 6'b000000~6'b111111: You can set the parameter within the range as requirements.
DYN_LPF_SEL	string	"TRUE", "FALSE"	"FALSE"	LPFRES and LPFCAP static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters LPF_RES and LPF_CAP. ● TRUE: Dynamic, namely select dynamic signals LPFREST and LPFCAP.
LPF_RES	binary	3'bXXX,3'b000~3'b111	3'bXXX	LPRRES static setting: ● 3'bXXX: Indicates that Gowin Software will automatically calculate and set this parameter. ● 3'b000~3'b111: You can set the parameter within the range as requirements.
LPF_CAP	binary	2'b00~2'b10	2'b00	LFPCAP static setting
SSC_EN	string	"TRUE", "FALSE"	"FALSE"	SSC enable

The input clock divider (IDIV) is used to control the input clock frequency into the PLL module. This divider coefficient can be adjusted dynamically through the port IDSEL or statically through the parameter IDIV_SEL, the correspondence is shown in Table 5-4.

The relationship between IDIV value and port IDSEL is IDSEL=dec2bin(64-IDIV).

Table 5-4 IDIV Dynamic and Static Correspondence

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

The FBDIV divider is used to divide the feedback signal. This divider coefficient can be adjusted dynamically by the port FBDSEL or statically by the parameter FBDIV_SEL, and the correspondence is shown in Table 5-5.

The relationship between FBDIV value and FBDSEL port is FBDSEL=dec2bin(64-FBDIV).

Table 5-5 FBDIV Dynamic and Static Correspondence

FBDSEL [5:0] (Dynamic)	FBDIV_SEL (Static)	FBDIV Actual Value
111111	1	1
111110	2	2
111101	3	3
111100	4	4
111011	5	5
111010	6	6
111001	7	7
111000	8	8
110111	9	9
……	……	……
000000	64	64

For output divider (ODIV), channel 0 supports integer divider and fraction divider; channel 1~6 support integer divider only. For channel 0, this dividing coefficient can be adjusted dynamically by ODSEL0 and ODSEL0_FRAC, or statically by the parameters ODIV0_SEL and ODIV0_FRAC_SEL. The correspondence of integer division coefficient is shown in Table 5-6, and the correspondence of fraction division coefficient is shown in Table 5-7.

The relationship between ODIV0 integer value and ODSEL port is ODSEL0=dec2bin(128-ODIV0 integer value).

Table 5-6 ODIV0 Integer Division Contrast

ODSEL0 [6:0 (Dynamic)	ODIV0_SEL (Static)	ODIV0 Integer 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

The relationship between ODIV0 fraction value and ODSEL0_FRAC port is ODSEL0_FRAC=dec2bin(7-ODIV0 fraction value/0.125), and the fraction division is valid only when the integer division is [2~127]. For channel 1~6, the ODIV division coefficient can be adjusted dynamically through ODSELx(x=1~6) or statically through the parameter ODIVx_SEL(x=1~6); for the correspondence, you can refer to Table 5-6.

Table 5-7 ODIV0 Fraction Division Contrast

ODSEL_FRAC[2:0] (Dynamic)	ODIV0_FRAC_SEL (Static)	ODIV Fraction Value
111	0	0*0.125=0
110	1	1*0.125=0.125
101	2	2*0.125=0.25
100	3	3*0.125=0.375
011	4	4*0.125=0.5
010	5	5*0.125=0.625
001	6	6*0.125=0.75
000	7	7*0.125=0.875

MDIV and CLKFB dividers work similarly to FBDIV. The division coefficients support integer and fraction, which can be adjusted dynamically through MDSEL and MDSEL_FRAC ports, and statically through MDIV_SEL and MDIV_FRAC_SEL parameters. The correspondence of integer division coefficients is shown in Table 5-8, and the correspondence of fraction division coefficients is shown in Table 5-9.

Table 5-8 MDIV Integer Division Contrast

MDSEL [6:0] (Dynamic)	MDIV_SEL (Static)	MDIV IInteger Value
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

The relationship between MDIV integer value and MDSEL port is MDSEL=dec2bin(128-MDIV integer value), and the value range of MDIV is [2~128].

Table 5-9 MDIV Fraction Division Contrast

MDSEL_FRAC[2:0] (Dynamic)	MDIV_FRAC_SEL (Static)	MDIV Fraction Value
111	0	0*0.125=0
110	1	1*0.125=0.125
101	2	2*0.125=0.25
100	3	3*0.125=0.375
011	4	4*0.125=0.5
010	5	5*0.125=0.625
001	6	6*0.125=0.75
000	7	7*0.125=0.875

The relationship between MDIV fraction value and MDSEL_FRAC port is MDSEL_FRAC=dec2bin(7-MDIV fraction value/0.125), and the fraction division frequency is valid when the integer division frequency is [2~127].

Phase Shift

PLL phase shift supports both static and dynamic adjustments and phase shift is supported by 0~6 channels. Taking MDIV channel as the reference, the phase shift is the shift of ODIV0~6 relative to MDIV.

Static phase shift is achieved by setting the parameters CLKOUTx_PE_COARSE and CLKOUTx_PE_FINE (x=0~6).

Dynamic phase shift is achieved through signals PSSEL, PSDIR and PSPULSE. PSSEL is used to control the channel selected; PSDIR is used to control the increase/decrease operations; a PSPULSE pulse falling edge DYN_FINE plus/minus 1; DYN_COARSE plus or minus 1 when

DYN_FINE overflows or underflows, and DYN_ COARSE value is less than ODIV.

The phase shift formula is as follows:

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

Channel 0 ODIV = ODIV0 integer value + ODIV0 fraction value; for channel 1~6, ODIV only takes integer value.

Note!

- DYN_FINE and DYN_COARSE are internal signals generated by DPA through PSSEL, PSDIR, PSPULSE.
- COARSE and FINE in the formula refers to the values that really act on the phase shift after selecting dynamic or static adjustment.
- The phase shift circuit is designed for VCO; for Bypass in or CAS mode, the formula is still available, but FINE needs to be set to 0.

Duty Cycle Adjustment

PLL duty cycle adjustment is supported by channel 0~6, and only supports dynamic adjustment. Duty cycle is defined as follows:

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

The position of the falling edge is determined by the static phase shift setting and is 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. Taking channel 1 as an example, the formulas of DUTY and PHASE are as follows:

DUTY=(CLKOUT1_PE_COARSE+CLKOUT1_PE_FINE/8)

PHASE=(DYN_COARSE1+DYN_FINE1/8)

Dynamic duty cycle formula:

If DUTY>PHASE, Duty cycle=(DUTY-PHASE)/ODIV1

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

Note!

- ODIV=1 does not support dynamic duty cycle adjustment, duty cycle is fixed at 50%.
- DUTY-PHASE does not support values between (-0.5,0.5) when ODIV>=2.
- Duty cycle adjustment circuit is designed for VCO, for Bypass in or CAS mode, duty cycle adjustment is not allowed, and in these two modes, if ODIV(>2) is odd, the duty cycle is not 50% (high level < low level, i.e. duty cycle is less than 50%).

Duty Cycle Fine Adjustment

PLL duty cycle fine adjustment supports both static and dynamic adjustments, and only supported by channel 0~3. The adjustment is achieved by setting the direction and step. When the trimming direction is 1'b1, the duty cycle increases by adjusting the falling edge delay; when the trimming direction is 1'b0, the duty cycle decreases by adjusting the rising edge delay. Taking channel 1 as an example, the details are shown in Table 5-10.

Table 5-10 Fine Adjustment Contrast of PLL Duty Cycle

Dynamic	Static		Delay Value
DT1[3:0]	CLKOUT1_DT_DIR	CLKOUT1_DT_STEP	
0111	1'b0	0	0
0110		1	-50ps
0101		2	-100ps
0011		4	-200ps
1111	1'b1	0	0
1110		1	+50ps
1101		2	+100ps
1011		4	+200ps

Assuming that channel 0 and 1 output the same frequency clock, channel 1 clock duty cycle needs fine adjustnent, channel 0 clock as a reference, the timing as shown in Figure 5-3 and Figure 5-4.

Figure 5-3 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b1, One Step)

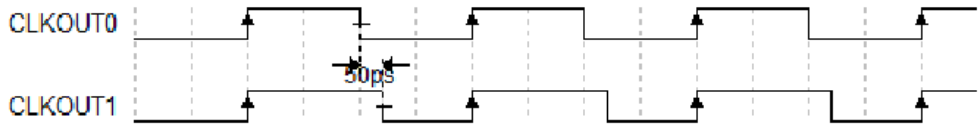
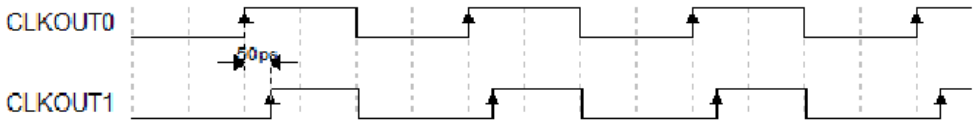


Figure 5-4 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b0, One Step)



Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

```
PLL uut (
  .LOCK(lock),
  .CLKOUT0(clkout0),
  .CLKOUT1(clkout1),
  .CLKOUT2(clkout2),
  .CLKOUT3(clkout3),
  .CLKOUT4(clkout4),
```

```
.CLKOUT5(clkout5),
.CLKOUT6(clkout6),
.CLKFBOUT(clkfbout),
.CLKIN(clkin),
.CLKFB(clkfb),
.RESET(reset),
.PLLPWD(pllpwd),
.RESET_I(reseti),
.RESET_O(reseto),
.FBDSEL(fbdsel),
.IDSEL(idsel),
.MDSEL(mdsel),
.MDSEL_FRAC(mdsel_frac),
.ODSEL0(odesl0),
.ODSEL0_FRAC(odesl0_frac),
.ODSEL1(odesl1),
.ODSEL2(odesl2),
.ODSEL3(odesl3),
.ODSEL4(odesl4),
.ODSEL5(odesl5),
.ODSEL6(odesl6),
.DT0(dt0),
.DT1(dt1),
.DT2(dt2),
.DT3(dt3),
.ICPSEL(icpsel),
.LPFRES(lpfres),
.LPFCAP(lpfcap),
.PSSEL(pssel),
.PSDIR(psdir),
.PSPULSE(pspulse),
.ENCLK0(enclk0),
.ENCLK1(enclk1),
.ENCLK2(enclk2),
.ENCLK3(enclk3),
```

```
        .ENCLK4(enclk4),
        .ENCLK5(enclk5),
        .ENCLK6(enclk6),
        .SSCPOL(sscpol),
        .SSCON(sscon),
        .SSCMDSEL(sscmdsel),
        .SSCMDSEL_FRAC(sscmdsel_frac)
    );
    defparam uut.CLK0_IN_SEL=1'b0;
    defparam uut.CLK0_OUT_SEL=1'b0;
    defparam uut.CLK1_IN_SEL=1'b0;
    defparam uut.CLK1_OUT_SEL=1'b0;
    defparam uut.CLK2_IN_SEL=1'b0;
    defparam uut.CLK2_OUT_SEL=1'b0;
    defparam uut.CLK3_IN_SEL=1'b0;
    defparam uut.CLK3_OUT_SEL=1'b0;
    defparam uut.CLK4_IN_SEL=2'b00;
    defparam uut.CLK4_OUT_SEL=1'b0;
    defparam uut.CLK5_IN_SEL=1'b0;
    defparam uut.CLK5_OUT_SEL=1'b0;
    defparam uut.CLK6_IN_SEL=1'b0;
    defparam uut.CLK6_OUT_SEL=1'b0;
    defparam uut.CLKFB_SEL="INTERNAL";
    defparam uut.CLKOUT0_DT_DIR=1'b1;
    defparam uut.CLKOUT0_DT_STEP=0;
    defparam uut.CLKOUT0_EN="TRUE";
    defparam uut.CLKOUT0_PE_COARSE=0;
    defparam uut.CLKOUT0_PE_FINE=0;
    defparam uut.CLKOUT1_DT_DIR=1'b1;
    defparam uut.CLKOUT1_DT_STEP=0;
    defparam uut.CLKOUT1_EN=" TRUE ";
    defparam uut.CLKOUT1_PE_COARSE=0;
    defparam uut.CLKOUT1_PE_FINE=0;
    defparam uut.CLKOUT2_DT_DIR=1'b1;
    defparam uut.CLKOUT2_DT_STEP=0;
```

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

```
defparam uut.DYN_ODIV1_SEL="FALSE";
defparam uut.DYN_ODIV2_SEL="FALSE";
defparam uut.DYN_ODIV3_SEL="FALSE";
defparam uut.DYN_ODIV4_SEL="FALSE";
defparam uut.DYN_ODIV5_SEL="FALSE";
defparam uut.DYN_ODIV6_SEL="FALSE";
defparam uut.DYN_PE0_SEL="FALSE";
defparam uut.DYN_PE1_SEL="FALSE";
defparam uut.DYN_PE2_SEL="FALSE";
defparam uut.DYN_PE3_SEL="FALSE";
defparam uut.DYN_PE4_SEL="FALSE";
defparam uut.DYN_PE5_SEL="FALSE";
defparam uut.DYN_PE6_SEL="FALSE";
defparam uut.FBDIV_SEL=1;
defparam uut.FCLKIN="100.0";
defparam uut.ICP_SEL=6'bXXXXXX;
defparam uut.IDIV_SEL=1;
defparam uut.LPF_CAP=2'b00;
defparam uut.LPF_RES=3'b000;
defparam uut.MDIV_FRAC_SEL=0;
defparam uut.MDIV_SEL=8;
defparam uut.ODIV0_FRAC_SEL=0;
defparam uut.ODIV0_SEL=8;
defparam uut.ODIV1_SEL=8;
defparam uut.ODIV2_SEL=8;
defparam uut.ODIV3_SEL=8;
defparam uut.ODIV4_SEL=8;
defparam uut.ODIV5_SEL=8;
defparam uut.ODIV6_SEL=8;
defparam uut.RESET_I_EN="FALSE";
defparam uut.RESET_O_EN="FALSE";
defparam uut.SSC_EN="FALSE";
```

VHDL Instantiation：

```
COMPONENT PLL
    GENERIC(
```

```
FCLKIN : STRING := "100.0";
DYN_IDIV_SEL : STRING := "FALSE";
IDIV_SEL : integer := 1;
DYN_FBDIV_SEL : STRING := "FALSE";
FBDIV_SEL : integer := 1;
DYN_ODIV0_SEL : STRING := "FALSE";
ODIV00_SEL : integer := 8;
DYN_ODIV1_SEL : STRING := "FALSE";
ODIV1_SEL : integer := 8;
DYN_ODIV2_SEL : STRING := "FALSE";
ODIV2_SEL : integer := 8;
DYN_ODIV3_SEL : STRING := "FALSE";
ODIV3_SEL : integer := 8;
DYN_ODIV4_SEL : STRING := "FALSE";
ODIV4_SEL : integer := 8;
DYN_ODIV5_SEL : STRING := "FALSE";
ODIV5_SEL : integer := 8;
DYN_ODIV6_SEL : STRING := "FALSE";
ODIV6_SEL : integer := 8;
DYN_MDIV_SEL : STRING := "FALSE";
MDIV_SEL : integer := 8;
MDIV_FRAC_SEL : integer := 0;
ODIV0_FRAC_SEL : integer := 0;
CLKOUT0_EN : STRING := "TRUE";
CLKOUT1_EN : STRING := " FALSE ";
CLKOUT2_EN : STRING := " FALSE ";
CLKOUT3_EN : STRING := " FALSE ";
CLKOUT4_EN : STRING := " FALSE ";
CLKOUT5_EN : STRING := " FALSE ";
CLKOUT6_EN : STRING := " FALSE ";
DYN_DT0_SEL : STRING := "FALSE";
DYN_DT1_SEL : STRING := "FALSE";
DYN_DT2_SEL : STRING := "FALSE";
DYN_DT3_SEL : STRING := "FALSE";
CLKOUT0_DT_DIR : bit := '1';
```

```
CLKOUT1_DT_DIR : bit := '1';
CLKOUT2_DT_DIR : bit := '1';
CLKOUT3_DT_DIR : bit := '1';
CLKOUT0_DT_STEP : integer := 0;
CLKOUT1_DT_STEP : integer := 0;
CLKOUT2_DT_STEP : integer := 0;
CLKOUT3_DT_STEP : integer := 0;
CLK0_IN_SEL    : bit := '0';
CLK0_OUT_SEL : bit := '0';
CLK1_IN_SEL    : bit_vector := '0';
CLK1_OUT_SEL : bit := '0';
CLK2_IN_SEL    : bit_vector := '0';
CLK2_OUT_SEL : bit := '0';
CLK3_IN_SEL    : bit_vector := '0';
CLK3_OUT_SEL : bit := '0';
CLK4_IN_SEL    : bit_vector := "00";
CLK4_OUT_SEL : bit := '0';
CLK5_IN_SEL    : bit_vector := '0';
CLK5_OUT_SEL : bit := '0';
CLK6_IN_SEL    : bit_vector := '0';
CLK6_OUT_SEL : bit := '0';
CLKFB_SEL : STRING := "INTERNAL";
DYN_DPA_EN : STRING := "FALSE";
DYN_PE0_SEL : STRING := "FALSE";
DYN_PE1_SEL : STRING := "FALSE";
DYN_PE2_SEL : STRING := "FALSE";
DYN_PE3_SEL : STRING := "FALSE";
DYN_PE4_SEL : STRING := "FALSE";
DYN_PE5_SEL : STRING := "FALSE";
DYN_PE6_SEL : STRING := "FALSE";
CLKOUT0_PE_COARSE : integer := 1;
CLKOUT0_PE_FINE : integer := 0;
CLKOUT1_PE_COARSE : integer := 0;
CLKOUT1_PE_FINE : integer := 0;
CLKOUT2_PE_COARSE : integer := 1;
```

```
                CLKOUT2_PE_FINE : integer := 0;
                CLKOUT3_PE_COARSE : integer := 0;
                CLKOUT3_PE_FINE : integer := 0;
                CLKOUT4_PE_COARSE : integer := 1;
                CLKOUT4_PE_FINE : integer := 0;
                CLKOUT5_PE_COARSE : integer := 0;
                CLKOUT5_PE_FINE : integer := 0;
                CLKOUT6_PE_COARSE : integer := 1;
                CLKOUT6_PE_FINE : integer := 0;
                DE0_EN : STRING := "FALSE";
                DE1_EN : STRING := "FALSE";
                DE2_EN : STRING := "FALSE";
                DE3_EN : STRING := "FALSE";
                DE4_EN : STRING := "FALSE";
                DE5_EN : STRING := "FALSE";
                DE6_EN : STRING := "FALSE";
                RESET_I_EN : STRING := "FALSE";
                RESET_O_EN : STRING := "FALSE";
                DYN_ICP_SEL : STRING := "FALSE";
                ICP_SEL : std_logic_vector(5 downto 0) := "XXXXXX";
                DYN_LPF_SEL : STRING := "FALSE";
                LPR_RES : std_logic_vector(2 downto 0) := "XXX";
                LPR_CAP : bit_vector := "00";
                SSC_EN : STRING := "FALSE"
        );
        PORT(
                CLKIN : IN std_logic;
                CLKFB : IN std_logic:='0';
                RESET, PLLPWD : IN std_logic:='0';
                RESET_I, RESET_O : IN std_logic:='0';
                IDSEL, FBDSEL : IN std_logic_vector(5 downto 0);
                ODSEL0,ODSEL1,ODSEL2,ODSEL3, DSEL4,ODSEL5,
            ODSEL6, MDSEL : IN std_logic_vector(6 downto 0);
                MDSEL_FRAC, ODSEL0_FRAC: IN std_logic_vector(2
downto 0);
```

```
                DT0,DT1,DT2,DT3 : IN std_logic_vector(3 downto 0);
                ICPSEL : IN std_logic_vector(5 downto 0);
                LPFRES : IN std_logic_vector(2 downto 0);
                LPFCAP : IN std_logic_vector(1 downto 0);
                PSSEL : IN std_logic_vector(2 downto 0);
                PSDIR,PSPULSE : IN std_logic;
                ENCLK0,ENCLK1,ENCLK2,ENCLK3 : IN std_logic;
                ENCLK4,ENCLK5,ENCLK6 : IN std_logic;
                SSCPOL, SSCON: IN std_logic;
                SSCMDSEL : IN std_logic_vector(6 downto 0);
                SSCMDSEL_FRAC : IN std_logic_vector(2 downto 0);
                LOCK : OUT std_logic;
                CLKOUT0, CLKOUT1 : OUT std_logic;
                CLKOUT2, CLKOUT3 : OUT std_logic;
                CLKOUT4, CLKOUT5 : OUT std_logic;
                CLKOUT6, CLKFBOUT : OUT std_logic
        );
END COMPONENT;
uut:PLL
        GENERIC MAP(
                FCLKIN => "100.0",
                DYN_IDIV_SEL => "FALSE",
                IDIV_SEL => 1,
                DYN_FBDIV_SEL => "FALSE",
                FBDIV_SEL => 1,
                DYN_ODIV0_SEL => "FALSE",
                ODIV00_SEL => 8,
                DYN_ODIV1_SEL => "FALSE",
                ODIV1_SEL => 8,
                DYN_ODIV2_SEL => "FALSE",
                ODIV2_SEL => 8,
                DYN_ODIV3_SEL => "FALSE",
                ODIV3_SEL => 8,
                DYN_ODIV4_SEL => "FALSE",
```

```
ODIV4_SEL => 8,
DYN_ODIV5_SEL => "FALSE",
ODIV5_SEL => 8,
DYN_ODIV6_SEL => "FALSE",
ODIV6_SEL => 8,
DYN_MDIV_SEL => "FALSE",
MDIV_SEL => 8,
MDIV_FRAC_SEL => 0,
ODIV0_FRAC_SEL => 0,
CLKOUT0_EN => "TRUE",
CLKOUT1_EN => " FALSE ",
CLKOUT2_EN => " FALSE ",
CLKOUT3_EN => " FALSE ",
CLKOUT4_EN => " FALSE ",
CLKOUT5_EN => " FALSE ",
CLKOUT6_EN => " FALSE ",
DYN_DT0_SEL => "FALSE",
DYN_DT1_SEL => "FALSE",
DYN_DT2_SEL => "FALSE",
DYN_DT3_SEL => "FALSE",
CLKOUT0_DT_DIR => '1',
CLKOUT1_DT_DIR => '1',
CLKOUT2_DT_DIR => '1',
CLKOUT3_DT_DIR => '1',
CLKOUT0_DT_STEP => 0,
CLKOUT1_DT_STEP => 0,
CLKOUT2_DT_STEP => 0,
CLKOUT3_DT_STEP => 0,
CLK0_IN_SEL => '0',
CLK0_OUT_SEL => '0',
CLK1_IN_SEL => '0',
CLK1_OUT_SEL => '0',
CLK2_IN_SEL => '0',
CLK2_OUT_SEL => '0',
CLK3_IN_SEL => '0',
```

```
CLK3_OUT_SEL => '0',
CLK4_IN_SEL => "00",
CLK4_OUT_SEL => '0',
CLK5_IN_SEL => '0',
CLK5_OUT_SEL => '0',
CLK6_IN_SEL => '0',
CLK6_OUT_SEL => '0',
CLKFB_SEL=> "INTERNAL",
DYN_DPA_EN => "FALSE",
DYN_PE0_SEL => "FALSE",
DYN_PE1_SEL => "FALSE",
DYN_PE2_SEL => "FALSE",
DYN_PE3_SEL => "FALSE",
DYN_PE4_SEL => "FALSE",
DYN_PE5_SEL => "FALSE",
DYN_PE6_SEL => "FALSE",
CLKOUT0_PE_COARSE => 1,
CLKOUT0_PE_FINE => 0,
CLKOUT1_PE_COARSE => 0,
CLKOUT1_PE_FINE => 0,
CLKOUT2_PE_COARSE=> 1,
CLKOUT2_PE_FINE => 0,
CLKOUT3_PE_COARSE => 0,
CLKOUT3_PE_FINE => 0,
CLKOUT4_PE_COARSE => 1,
CLKOUT4_PE_FINE => 0,
CLKOUT5_PE_COARSE => 0,
CLKOUT5_PE_FINE => 0,
CLKOUT6_PE_COARSE => 1,
CLKOUT6_PE_FINE => 0,
DE0_EN => "FALSE",
DE1_EN => "FALSE",
DE2_EN => "FALSE",
DE3_EN => "FALSE",
DE4_EN => "FALSE",
```

```
        DE5_EN => "FALSE",
        DE6_EN => "FALSE",
        RESET_I_EN => "FALSE",
        RESET_O_EN => "FALSE",
        DYN_ICP_SEL => "FALSE",
        ICP_SEL => "XXXXXX",
        DYN_LPF_SEL => "FALSE",
        LPR_RES => "XXX",
        LPR_CAP => "00",
        SSC_EN => "FALSE"
)
 PORT MAP(
LOCK=>lock,
   CLKOUT0=>clkout0,
   CLKOUT1=>clkout1,
   CLKOUT2=>clkout2,
   CLKOUT3=>clkout3,
   CLKOUT4=>clkout4,
   CLKOUT5=>clkout5,
   CLKOUT6=>clkout6,
   CLKFBOUT=>clkfbout,
   CLKIN=>clkin,
   CLKFB=>clkfb,
   RESET=>reset,
   PLLPWD=>pllpwd,
   RESET_I=>reseti,
   RESET_O=>reseto,
   FBDSEL=>fbdsel,
   IDSEL=>idsel,
   MDSEL=>mdsel,
   MDSEL_FRAC=>mdsel_frac,
   ODSEL0=>odesl0,
   ODSEL0_FRAC=>odesl0_frac,
   ODSEL1=>odesl1,
   ODSEL2=>odesl2,
```

```
        ODSEL3=>odesl3,
        ODSEL4=>odesl4,
        ODSEL5=>odesl5,
        ODSEL6=>odesl6,
        DT0=>dt0,
        DT1=>dt1,
        DT2=>dt2,
        DT3=>dt3,
        ICPSEL=>icpsel,
        LPFRES=>lpfres,
        LPFCAP=>lpfcap,
        PSSEL=>pssel,
        PSDIR=>psdir,
        PSPULSE=>pspulse,
        ENCLK0=>enclk0,
        ENCLK1=>enclk1,
        ENCLK2=>enclk2,
        ENCLK3=>enclk3,
        ENCLK4=>enclk4,
        ENCLK5=>enclk5,
        ENCLK6=>enclk6,
        SSCPOL=>sscpol,
        SSCON=>sscon,
        SSCMDSEL=>sscmdsel,
            SSCMDSEL_FRAC=>sscmdsel_frac
    );
```

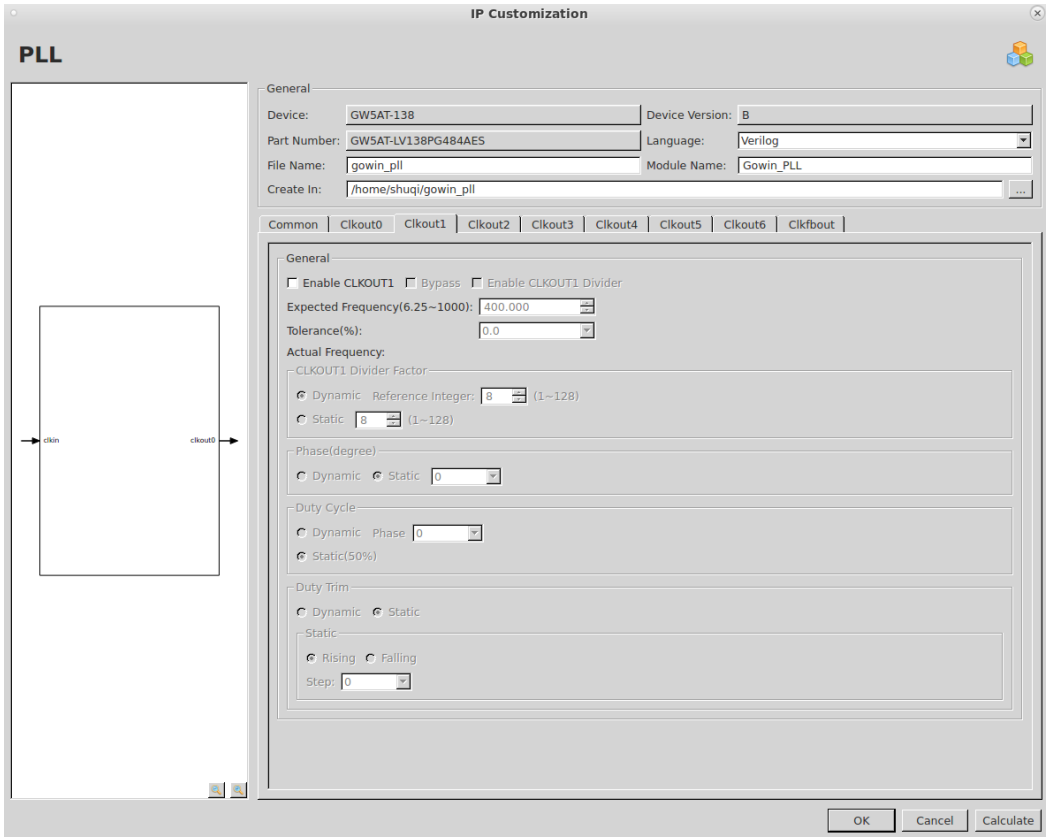
5.1.2 IP Generation

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

IP Configuration

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

Figure 5-5 IP Customization of PLL



1. General
   The General configuration box is used to configure the generated IP design file. PLL General configuration is similar to that of DCE. 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 5-5.
   - General: Configure "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.
   - CLKIN: Configure PLL input clock frequency, and divide factor.
     - "Clock Frequency": Configure the frequency of the input clock.
     - "Divide Factor": Set the Divide Factor in advanced mode and support "Dynamic" or "Static". In Static mode, Divide Factor value can be set as a specific value, which ranges from 1 to 64.
     - "VCO Frequency": Calculated VCO frequency, read only.
   - CLKFB: Configure the source and divide factor of PLL.

- When configuring the source of the feedback clock, you can select "Internal" and "External". If Internal is selected, the feedback comes from internal; if External is selected, the feedback comes from one of CLKOUT0~6 and CLKFBOUT, which you can select as required.
- Divide Factor: Set the Divide Factor in advanced mode and support "Dynamic" or "Static". In Static mode, Divide Factor value can be set as a specific value, which ranges from 1 to 64.

- ICP and LPF: Control the ICP current dynamically; the current increases as the value increases, and the current is minimized when the value is 0.
- LPFSEL
  - Control the LPFRES dynamically with RES value ranging from R0 to R7; R0 corresponds to the largest bandwidth and R7 corresponds to the smallest bandwidth.
  - Control the LPFCAP dynamically with CAP value ranging from small to large C0~C2.
- PLL Reset
  - If "PLL Reset" is selected, all reset signals of PLL, reset digital circuit, and configure the RESET enable mode of PLLO.
  - If "PLL Power Down" is selected, the analog circuit is power down.
  - If "CLKIN Divider Reset" is selected, enable RESET_I.
  - If "CLKOUT Divider Reset" is selected, enable RESET_O.
- Lock: PLL lock indictor signal
- Optional Ports
  - Spread spectrum clocking (SSC) is supported in Advanced mode.
  - If "SSC" is selected, the feedback can be only from the Internal.
- CLKOUT1: (Taking CLKOUT1 as an example, you can refer to CLKOUT1 for other Clkouts.)

General:

- You can choose to enable CLKOUT1~CLKOUT6, and CLKOUT0 is enabled by default and can not be configured.
- If you choose to disable CLKOUT1, all options on this page can not be selected, and the default is disable.
- If "Bypass" and "Enable CLKOUT1 Divider" are selected, it is Bypass in mode; if "Enable CLKOUT1 Divider" is not selected, it is Bypass out mode.

- Bypass: Configure the bypass of the output clock; the "Enable CLKOUTA Divider" configures the bypass of the VCO clock.
- Expected Frequency: Configure the expected frequency of the CLKOUTA in general mode, with a range of 6.25M to 1000M in non-bypass mode.
- Tolerance(%): Set a tolerance for the CLKOUT between the expected frequency and actual frequency calculated.
- Actual Frequency: The actual frequency that can be generated automatically.

CLKOUT1 Divider Factor:

- You can set the output divider of CLKOUT1, supporting static and dynamic adjustments, with the range of 1~128. When the configuration is not reasonable, click "Calculate" or "OK", an error prompt will pop up.

Phase:

- PLL phase supports static and dynamic adjustments, and 0~6 channels all support the phase shift.

Duty Cycle:

- PLL duty cycle only supports dynamic adjustment, and and 0~6 channels all support this adjustment. The duty cycle adjustment circuit is designed for VCO, and duty cycle adjustment is not supported for Bypass in or cascade mode.

Duty Trim:

- PLL duty cycle trim supports static and dynamic adjustments, and only 0~3 channels support duty cycle trim.

CLKFBOUT:

- CLKFBOUT can be configured, and the configuration parameters can be referred to the above CLKOUT1.

Calculate:

- Calculate whether the divider, multiplier and VCO parameters are reasonable; if not, click "Calculate", an "error" window will pop up; if correct, click "Calculate", the "succeed" window will pop up to indicate the successful configuration.

3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration as shown in Figure 5-5.

IP Generated Files

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

- "gowin_pll.v" file is a complete Verilog module to generate instantiated PLL, and it is generated according to the IP configuration.

- "gowin_pll_tmp.v" is the template file.
- "gowin_pll.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 PLLA

5.2.1 Primitive Introduction

Arora V FPGA products provide phase-locked loop (PLLA) and support seven clock outputs, and each clock supports independent adjustment of clock frequency, phase and duty cycle based on a given reference input clock.

Device Supported

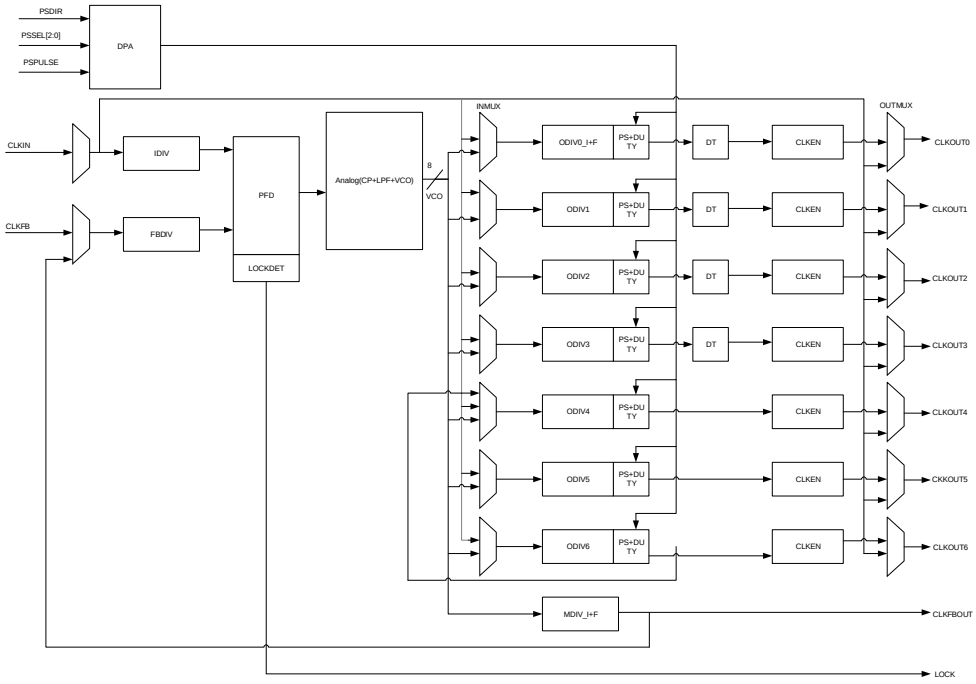
Table 5-11 PLLA Device Supported

Product Family	Series	Device
Arora	GW5A	GW5A-25

Functional Description

The PLLA structure diagram is as shown in Figure 5-1.

Figure 5-6 PLLA Structure Diagram



Based on the given reference input clock, PLLA supports the adjustments of clock phase, duty cycle, frequency to generate output clocks with different phases, duty cycles, and frequencies. CLKOUT0 and

CLKFBOUT support 1/8 fraction frequency adjustment; CLKOUT0~CLKOUT3 support static trimming duty cycle. PLLA also supports internal cascading from CLKOUT6 to CLKOUT4, SSC, and clock de-skewing to achieve CLKIN and CLKOUT alignment.

To get the correct clock output, the input clock frequency must be set according to the frequency range described in the *FPGA Product Datasheet*.

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

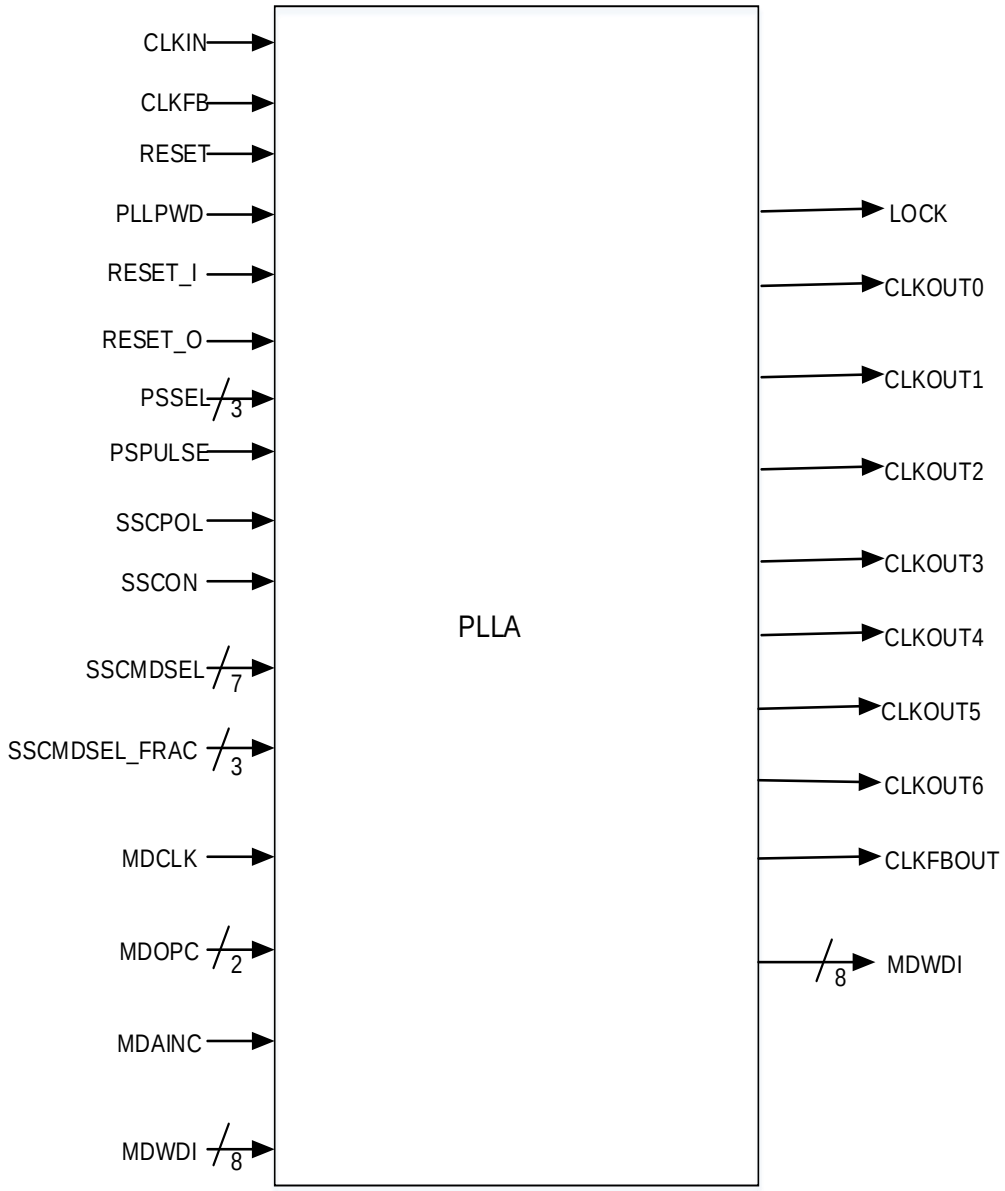
1. Fpfd=Fclkin/IDIV
2. Fclkfb=Fpfd*FBDIV
3. Depending on different feedbacks, VCO frequency formula is different:
   - Internal feedback: Fvco=Fclkfb*MDIV
   - External feedback: Fvco=Fclkfb*MDIV ---- CLKFBOUT Feedback to CLKFB; Fvco=Fclkfb*ODIVx ---- CLKOUTx Feedback to CLKFB
4. Fclkfbout= Fvco/MDIV
5. Depending on the mode selected of INMUX and OUTMUX, the formula of the output frequency of CLKOUT channel is different:
   - VCO in mode (INMUX from VCO): Fclkoutx= Fvco/ODIVx
   - Bypass in mode (INMUX from CLKIN): Fclkoutx=Fclkin/ODIVx
   - Bypass out mode (OUTMUX from CLKIN): Fclkoutx=Fclkin
   - CAS mode (channel 4 only): Fclkout4=Fclkout6[1]/ODIV4

Note!

- F_{CLKIN} is the input clock CLKIN frequency.
- Fclkoutx: x=0~6, the output clock frequency of channels 0~6.
- F_{clkfb} is the frequency of feebback input clock CLKFB.
- F_{PFD} is the PFD phase detection frequency.
- IDIV, FBDIV, MDIV, ODIVx (x=0~6) are the frequency division coefficients of different frequency dividers, which can be adjusted to get the clock signal with expected frequency.
- [1] Fclkout6 refers to the output clock of channel 6 ODIV6.

Port Diagram

Figure 5-7 PLLA Port Diagram



Port Description

Table 5-12 PLLA Port Description

Port	I/O	Description
CLKIN	Input	Reference clock input
CLKFB	Input	Feedback clock input
RESET	Input	All reset signal of PLL; reset digital circuit, active-high.
PLLPWD	Input	PLL power down signal; power down analog circuits, active-high.
RESET_I	Input	PLL global reset with IDIV, active-high; generally used for internal test.

Port	I/O	Description
RESET_O	Input	Reset ODIV and some digital circuits, active-high, generally used for internal test.
PSDIR	Input	Dynamic control of phase shift direction
PSSEL[2:0]	Input	Dynamic control of channel selected for phase shift
PSPULSE	Input	Dynamic control of clock pulse of phase shift
SSCPOL	Input	Optional configuration to avoid timing conflicts: ● 0：CLK Rising Edge ● 1：CLK Falling Edge
SSCON	Input	Dynamic control of SSC enable
SSCMDSEL[6:0]	Input	Dynamic control of SSC MDIV integer value, the recommended SSC MDIV actual value range is 16~24 (corresponding Fpfd of 50M), 32~48 (corresponding Fpfd of 25M), and the actual value is 128-SSCMDSEL.
SSCMDSEL_FRAC[2:0]	Input	Dynamic control of SSC MDIV fraction value, and the fraction value is 1/8.
CLKOUT0	Output	Channel 0 clock output (default)
CLKOUT1	Output	Channel 1 clock output
CLKOUT2	Output	Channel 2 clock output
CLKOUT3	Output	Channel 3 clock output
CLKOUT4	Output	Channel 4 clock output
CLKOUT5	Output	Channel 5 clock output
CLKOUT6	Output	Channel 6clock output
CLKFBOUT	Output	Feedback clock output
LOCK	Output	PLL lock indication： ● 1: Lock ● 0: Unlock
MDCLK	Input	Clock input signal, all signals on the DRP port are triggered by the rising edge of the this clock
MDOPC [1:0]	Input	Operation type coding: 00: No read/write operation (NOOP) 10: Read operation (RD) 01: Write operation(WR) 11: Whether MDAINC is high or not, the register address remains unchanged
MDAINC	Input	Address increase indication signal, active-high.
MDWDI[7:0]	Input	Data to be written through the mDRP port
MDRDO[7:0]	Output	Data to be read from the mDRP port

Parameter Description

Table 5-13 PLLA Parameter Description

Name	Type	Value	Default	Description
FCLKIN	string	"10"~"400"	"100.0"	Reference clock frequency (MHz)
IDIV_SEL	integer	1~64	1	IDIV frequency division coefficient static setting, the actual value is 1~64.
FBDIV_SEL	integer	1~64	1	FBDIV frequency division coefficient static setting, the actual value is 1~64.
ODIV0_SEL	integer	1~128	8	Integer static setting for ODIV0 frequency division coefficient
ODIV0_FRAC_SEL	integer	0~7	0	Fraction static setting for ODIV0 frequency division coefficient
ODIV1_SEL	integer	1~128	8	ODIV1 frequency division coefficient static setting
ODIV2_SEL	integer	1~128	8	ODIV2 frequency division coefficient static setting
ODIV3_SEL	integer	1~128	8	ODIV3 frequency division coefficient static setting
ODIV4_SEL	integer	1~128	8	ODIV4 frequency division coefficient static setting
ODIV5_SEL	integer	1~128	8	ODIV5 frequency division coefficient static setting
ODIV6_SEL	integer	1~128	8	ODIV6 frequency division coefficient static setting
MDIV_SEL	integer	2~128	8	Integer static setting for MDIV frequency division coefficient
MDIV_FRAC_SEL	string	0~7	0	Fraction static setting for MDIV frequency division coefficient
CLKOUT0_EN	string	"TRUE", "FALSE"	"TRUE"	Channel 0 clock output enable
CLKOUT1_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 1 clock output enable
CLKOUT2_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 2 clock output enable
CLKOUT3_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 3 clock output enable
CLKOUT4_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 4 clock output enable
CLKOUT5_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 5 clock output enable
CLKOUT6_EN	string	"TRUE", "FALSE"	"FALSE"	Channel 6 clock output enable
CLKOUT0_DT_DIR	binary	1'b1, 1'b0	1'b1	Static trimming direction of channel 0 duty cycle ● 1'b1: Duty cycle increase (+), adjust falling edge according to

Name	Type	Value	Default	Description
				the rising edge alignment.. ● 1'b0: Duty cycle decrease (-), adjust rising edge according to the falling edge alignment.
CLKOUT1_DT_DIR	binary	1'b1, 1'b0	1'b1	Static trimming direction of channel 1 duty cycle ● 1'b1: Duty cycle increase (+); adjust falling edge according to the rising edge alignment. ● 1'b0: Duty cycle decrease (-); adjust rising edge according to the falling edge alignment.
CLKOUT2_DT_DIR	binary	1'b1, 1'b0	1'b1	Static trimming direction of channel 2 duty cycle ● 1'b1: Duty cycle increase (+); adjust falling edge according to the rising edge alignment. ● 1'b0: Duty cycle decrease (-); adjust rising edge according to the falling edge alignment.
CLKOUT3_DT_STEP	integer	0,1,2,4	0	Static trimming direction of channel 3 duty cycle ● 1'b1: Duty cycle increase (+); adjust falling edge according to the rising edge alignment. ● 1'b0: Duty cycle decrease (-); adjust rising edge according to the falling edge alignment.
CLKOUT0_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 0 duty cycle, 50ps of each step.
CLKOUT1_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 1 duty cycle, 50ps of each step.
CLKOUT2_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 2 duty cycle, 50ps of each step.
CLKOUT3_DT_STEP	integer	0,1,2,4	0	Static trimming step of channel 3 duty cycle, 50ps of each step.
CLK0_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV0 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK0_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 0 output clock source selected: ● 1'b0: From ODIV0 output ● 1'b1: Output clock bypass from CLKIN
CLK1_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV1 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN

Name	Type	Value	Default	Description
CLK1_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 1 output clock source selected: ● 1'b0: From ODIVA1 output ● 1'b1: Output clock bypass from CLKIN
CLK2_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV2 input clock source selected: ● 1'b0: From VCO output ● 2'b1: Output clock bypass from CLKIN
CLK2_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 2 output clock source selected: ● 1'b0: From ODIVA2 output ● 1'b1: Output clock bypass from CLKIN
CLK3_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV3 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK3_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 3 output clock source selected: ● 1'b0: From ODIVA3 output ● 1'b1: Output clock bypass from CLKIN
CLK4_IN_SEL	binary	2'b00, 2'b01,2’b10	2'b00	ODIV4 input clock source selected: ● 2'b00: From VCO output ● 2'b01: Cascade from CLKCAS_6 ● 2'b10: Output clock bypass from CLKIN
CLK4_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 4 output clock source selected: ● 1'b0: From ODIVA4 output ● 1'b1: Output clock bypass from CLKIN
CLK5_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV5 input clock source selected: ● 1'b0: From VCO output ● 1'b1: Output clock bypass from CLKIN
CLK5_OUT_SEL	binary	1'b0, 1'b1	1'b0	Channel 5 output clock source selected: ● 1'b0: From ODIVA5 output ● 1'b1: Output clock bypass from CLKIN
CLKFB_SEL	string	"INTERNAL", "EXTERNAL"	"INTERNAL"	CLKFB source selected: ● INTERNAL: From internal CLKOUT feedback ● EXTERNAL: From external signal feedback

Name	Type	Value	Default	Description
DYN_DPA_EN	string	"TRUE", "FALSE"	"FALSE"	Dynamic phase shift adjustment enable
CLKOUT0_PE_COARSE	integer	0~127	0	Static setting of channel 0 coarse phase shift
CLKOUT0_PE_FINE	integer	0~7	0	Static setting of channel 0 fine phase shift
CLKOUT1_PE_COARSE	integer	0~127	0	Static setting of channel 1 coarse phase shift
CLKOUT1_PE_FINE	integer	0~7	0	Static setting of channel 1 fine phase shift
CLKOUT2_PE_COARSE	integer	0~127	0	Static setting of channel 2 coarse phase shift
CLKOUT2_PE_FINE	integer	0~7	0	Static setting of channel 2 fine phase shift
CLKOUT3_PE_COARSE	integer	0~127	0	Static setting of channel 3 coarse phase shift
CLKOUT3_PE_FINE	integer	0~7	0	Static setting of channel 3 fine phase shift
CLKOUT4_PE_COARSE	integer	0~127	0	Static setting of channel 4 coarse phase shift
CLKOUT4_PE_FINE	integer	0~7	0	Static setting of channel 4 fine phase shift
CLKOUT5_PE_COARSE	integer	0~127	0	Static setting of channel 5 coarse phase shift
CLKOUT5_PE_FINE	integer	0~7	0	Static setting of channel 5 fine phase shift
CLKOUT6_PE_COARSE	integer	0~127	0	Static setting of channel 6 coarse phase shift
CLKOUT6_PE_FINE	integer	0~7	0	Static setting of channel 6 fine phase shift
DYN_PE0_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 0 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT0_PE_COARSE and CLKOUT0_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE1_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 1 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT1_PE_COARSE and CLKOUT1_PE_FINE. ● TRUE: Dynamic, namely select

Name	Type	Value	Default	Description
				dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE2_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 2 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT2_PE_COARSE and CLKOUT2_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE3_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 3 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT3_PE_COARSE and CLKOUT3_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE4_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 4 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT4_PE_COARSE and CLKOUT4_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE5_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 5 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters CLKOUT5_PE_COARSE and CLKOUT5_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DYN_PE6_SEL	string	“TRUE","FALSE"	“FALSE"	Channel 6 phase shift static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters

Name	Type	Value	Default	Description
				CLKOUT6_PE_COARSE and CLKOUT6_PE_FINE. ● TRUE: Dynamic, namely select dynamic signal DPA (PSSEL, PSDIR and PSPULSE) with DYN_DPA_EN="TRUE".
DE0_EN	string	“TRUE","FALSE"	“FALSE"	Channel 0 (ODIV0=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE0_SEL="TRUE", set CLKOUT0_PE_COARSE and CLKOUT0_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE1_EN	string	“TRUE","FALSE"	“FALSE"	Channel 1 (ODIV1=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE1_SEL="TRUE", set CLKOUT1_PE_COARSE and CLKOUT1_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE2_EN	string	“TRUE","FALSE"	“FALSE"	Channel 2 (ODIV2=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE2_SEL="TRUE", set CLKOUT2_PE_COARSE and CLKOUT2_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE3_EN	string	“TRUE","FALSE"	“FALSE"	Channel 3 (ODIV3=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE3_SEL="TRUE", set CLKOUT3_PE_COARSE and

Name	Type	Value	Default	Description
				CLKOUT3_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE4_EN	string	“TRUE","FALSE"	“FALSE"	Channel 4 (ODIV4=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE4_SEL="TRUE", set CLKOUT4_PE_COARSE and CLKOUT4_PE_FIN as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE5_EN	string	“TRUE","FALSE"	“FALSE"	Channel 5 (ODIV5=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE5_SEL="TRUE", set CLKOUT5_PE_COARSE and CLKOUT5_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
DE6_EN	string	“TRUE","FALSE"	“FALSE"	Channel 6 (ODIV6=2~128) duty cycle adjustment enable: ● "FALSE": 50% duty cycle ● "TRUE": When DYN_PE6_SEL="TRUE", set CLKOUT6_PE_COARSE and CLKOUT6_PE_FINE as the falling edge, and dynamically adjust the phase as the rising edge to achieve dynamic duty cycle adjustment (falling edge-rising edge).
RESET_I_EN	string	"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_O_EN	string	"TRUE", "FALSE"	"FALSE"	Enable the dynamic signal RESET_O. if you need to use the RESET_O port, you need to set this

Name	Type	Value	Default	Description
				parameter to TRUE.
DYN_ICP_SEL	string	"TRUE", "FALSE"	"FALSE"	ICPSEL static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameter ICP_SEL. ● TRUE: Dynamic, namely select dynamic signal ICPSEL.
ICP_SEL	binary	6'bXXXXXX, 6'b000000~6'b111111	6'bXXXXXX	ICP current static setting: ● 6'bXXXXXX: Indicates that Gowin Software will automatically calculate and set this parameter. ● 6'b000000~6'b111111: You can set the parameter within the range as requirements.
DYN_LPF_SEL	string	"TRUE", "FALSE"	"FALSE"	LPFRES and LPFCAP static control parameter or dynamic control signal selected. ● FALSE: Static, namely select the parameters LPF_RES and LPF_CAP. ● TRUE: Dynamic, namely select dynamic signals LPFREST and LPFCAP.
LPF_RES	binary	3'bXXX,3'b000~3'b111	3'bXXX	LPRRES static setting: ● 3'bXXX: Indicates that Gowin Software will automatically calculate and set this parameter. ● 3'b000~3'b111: You can set the parameter within the range as requirements.
LPF_CAP	binary	2'b00~2'b10	2'b00	LFPCAP static setting
SSC_EN	string	"TRUE", "FALSE"	"FALSE"	SSC enable

The input clock divider (IDIV) is used to control the input clock frequency into the PLL module. This divider coefficient can be adjusted statically through the parameter IDIV_SEL, the correspondence is shown in Table 5-14.

Table 5-14 IDIV Static Correspondence

IDIV_SEL (Static)	IDIV Actual Value
1	1
2	2
3	3
4	4
5	5

IDIV_SEL (Static)	IDIV Actual Value
6	6
7	7
8	8
9	9
……	……
64	64

The FBDIV divider is used to divide the feedback signal. This divider coefficient can be adjusted statically by the parameter FBDIV_SEL, and the correspondence is shown in Table 5-15.

Table 5-15 FBDIV Static Correspondence

FBDIV_SEL (Static)	FBDIV Actual Value
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
64	64

For output divider (ODIV), channel 0 supports integer divider and fraction divider; channel 1~6 support integer divider only. For channel 0, this dividing coefficient can be adjusted statically by the parameters ODIV0_SEL and ODIV0_FRAC_SEL. The correspondence of integer division coefficient is shown in Table 5-16, and the correspondence of fraction division coefficient is shown in Table 5-17.

Table 5-16 ODIV0 Integer Division Contrast

ODIV0_SEL (Static)	ODIV0 Integer Value
1	1
2	2
3	3
4	4
5	5
6	6
7	7

ODIV0_SEL (Static)	ODIV0 Integer Value
8	8
9	9
……	……
128	128

ODIV0 fraction division is valid only when the integer division is [2~127]. For channel 1~6, the ODIV division coefficient can be adjusted dynamically through ODSELx(x=1~6) or statically through the parameter ODIVx_SEL(x=1~6); for the correspondence, you can refer to Table 5-17.

Table 5-17 ODIV0 Fraction Division Contrast

ODIV0_FRAC_SEL (Static)	ODIV Fraction Value
0	0*0.125=0
1	1*0.125=0.125
2	2*0.125=0.25
3	3*0.125=0.375
4	4*0.125=0.5
5	5*0.125=0.625
6	6*0.125=0.75
7	7*0.125=0.875

MDIV and CLKFB dividers work similarly to FBDIV. The division coefficients support integer and fraction, which can be adjusted statically through MDIV_SEL and MDIV_FRAC_SEL parameters. The correspondence of integer division coefficients is shown in Table 5-18, and the correspondence of fraction division coefficients is shown in Table 5-19.

Table 5-18 MDIV Integer Division Contrast

MDIV_SEL (Static)	MDIV IInteger Value
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
128	128

MDIV integer range is [2~128].

Table 5-19 MDIV Fraction Division Contrast

MDIV_FRAC_SEL (Static)	MDIV Fraction Value
0	0*0.125=0
1	1*0.125=0.125
2	2*0.125=0.25
3	3*0.125=0.375
4	4*0.125=0.5
5	5*0.125=0.625
6	6*0.125=0.75
7	7*0.125=0.875

MDIV fraction division frequency is valid when the integer division frequency is [2~127].

Phase Shift

PLLA phase shift supports both static and dynamic adjustments and phase shift is supported by 0~6 channels. Taking MDIV channel as the reference, the phase shift is the shift of ODIV0~6 relative to MDIV.

Static phase shift is achieved by setting the parameters CLKOUTx_PE_COARSE and CLKOUTx_PE_FINE (x=0~6).

Dynamic phase shift is achieved through signals PSSEL, PSDIR and PSPULSE. PSSEL is used to control the channel selected; PSDIR is used to control the increase/decrease operations; a PSPULSE pulse falling edge DYN_FINE plus/minus 1; DYN_COARSE plus or minus 1 when DYN_FINE overflows or underflows, and DYN_ COARSE value is less than ODIV.

The phase shift formula is as follows:

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

Channel 0 ODIV = ODIV0 integer value + ODIV0 fraction value; for channel 1~6, ODIV only takes integer value.

Note!

- DYN_FINE and DYN_COARSE are internal signals generated by DPA through PSSEL, PSDIR, PSPULSE.
- COARSE and FINE in the formula refers to the values that really act on the phase shift after selecting dynamic or static adjustment.
- The phase shift circuit is designed for VCO; for Bypass in or CAS mode, the formula is still available, but FINE needs to be set to 0.

Duty Cycle Adjustment

PLLA duty cycle adjustment is supported by channel 0~6, and only supports dynamic adjustment. Duty cycle is defined as follows:

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

The position of the falling edge is determined by the static phase shift setting and is 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. Taking channel 1 as an example, the formulas of DUTY and PHASE are as follows:

DUTY=(CLKOUT1_PE_COARSE+CLKOUT1_PE_FINE/8)

PHASE=(DYN_COARSE1+DYN_FINE1/8)

Dynamic duty cycle formula:

If DUTY>PHASE, Duty cycle=(DUTY-PHASE)/ODIV1

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

Note!

- ODIV=1 does not support dynamic duty cycle adjustment, duty cycle is fixed at 50%.
- DUTY-PHASE does not support values between (-0.5,0.5) when ODIV>=2.
- Duty cycle adjustment circuit is designed for VCO, for Bypass in or CAS mode, duty cycle adjustment is not allowed, and in these two modes, if ODIV(>2) is odd, the duty cycle is not 50% (high level < low level, i.e. duty cycle is less than 50%).

Duty Cycle Fine Adjustment

PLLA duty cycle fine adjustment supports both static and dynamic adjustments, and only supported by channel 0~3. The adjustment is achieved by setting the direction and step. When the trimming direction is 1'b1, the duty cycle increases by adjusting the falling edge delay; when the trimming direction is 1'b0, the duty cycle decreases by adjusting the rising edge delay. Taking channel 1 as an example, the details are shown in Table 5-20.

Table 5-20 Fine Adjustment Contrast of PLLA Duty Cycle

Dynamic	Static		Delay Value
DT1[3:0]	CLKOUT1_DT_DIR	CLKOUT1_DT_STEP	
0111	1'b0	0	0
0110		1	-50ps
0101		2	-100ps
0011		4	-200ps
1111	1'b1	0	0
1110		1	+50ps
1101		2	+100ps
1011		4	+200ps

Assuming that channel 0 and 1 output the same frequency clock, channel 1 clock duty cycle needs fine adjustnent, channel 0 clock as a reference, the timing as shown in Figure 5-8 and Figure 5-9.

Figure 5-8 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction

1'b1, One Step)

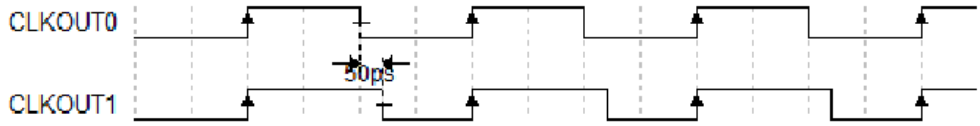
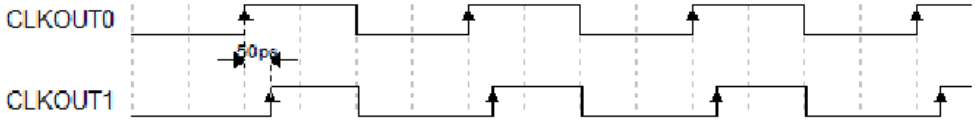


Figure 5-9 Channel 1 Duty Cycle Fine Adjustment Timing Diagram (Direction 1'b0, One Step)



Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

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

```
defparam uut.CLKOUT2_EN="FALSE";
defparam uut.CLKOUT2_PE_COARSE=0;
defparam uut.CLKOUT2_PE_FINE=0;
defparam uut.CLKOUT3_DT_DIR=1'b1;
defparam uut.CLKOUT3_DT_STEP=0;
defparam uut.CLKOUT3_EN="FALSE";
defparam uut.CLKOUT3_PE_COARSE=0;
defparam uut.CLKOUT3_PE_FINE=0;
defparam uut.CLKOUT4_EN=" TRUE ";
defparam uut.CLKOUT4_PE_COARSE=0;
defparam uut.CLKOUT4_PE_FINE=0;
defparam uut.CLKOUT5_EN="FALSE";
defparam uut.CLKOUT5_PE_COARSE=0;
defparam uut.CLKOUT5_PE_FINE=0;
defparam uut.CLKOUT6_EN="FALSE";
defparam uut.CLKOUT6_PE_COARSE=0;
defparam uut.CLKOUT6_PE_FINE=0;
defparam uut.DE0_EN="FALSE";
defparam uut.DE1_EN="FALSE";
defparam uut.DE2_EN="FALSE";
defparam uut.DE3_EN="FALSE";
defparam uut.DE4_EN="FALSE";
defparam uut.DE5_EN="FALSE";
defparam uut.DE6_EN="FALSE";
defparam uut.DYN_DPA_EN="FALSE";
defparam uut.DYN_PE0_SEL="FALSE";
defparam uut.DYN_PE1_SEL="FALSE";
defparam uut.DYN_PE2_SEL="FALSE";
defparam uut.DYN_PE3_SEL="FALSE";
defparam uut.DYN_PE4_SEL="FALSE";
defparam uut.DYN_PE5_SEL="FALSE";
defparam uut.DYN_PE6_SEL="FALSE";
defparam uut.FBDIV_SEL=1;
defparam uut.FCLKIN="100.0";
defparam uut.ICP_SEL=6'bXXXXXX;
```

```
defparam uut.IDIV_SEL=1;
defparam uut.LPF_CAP=2'b00;
defparam uut.LPF_RES=3'bXXX;
defparam uut.MDIV_FRAC_SEL=0;
defparam uut.MDIV_SEL=8;
defparam uut.ODIV0_FRAC_SEL=0;
defparam uut.ODIV0_SEL=8;
defparam uut.ODIV1_SEL=8;
defparam uut.ODIV2_SEL=8;
defparam uut.ODIV3_SEL=8;
defparam uut.ODIV4_SEL=8;
defparam uut.ODIV5_SEL=8;
defparam uut.ODIV6_SEL=8;
defparam uut.RESET_I_EN="FALSE";
defparam uut.RESET_O_EN="FALSE";
defparam uut.SSC_EN="FALSE";
```

VHDL Instantiation：

```
COMPONENT PLLA
    GENERIC(
        FCLKIN : STRING := "100.0";
        IDIV_SEL : integer := 1;
        FBDIV_SEL : integer := 1;
        ODIV00_SEL : integer := 8;
        ODIV1_SEL : integer := 8;
        ODIV2_SEL : integer := 8;
        ODIV3_SEL : integer := 8;
        ODIV4_SEL : integer := 8;
        ODIV5_SEL : integer := 8;
        ODIV6_SEL : integer := 8;
        MDIV_SEL : integer := 8;
        MDIV_FRAC_SEL : integer := 0;
        ODIV0_FRAC_SEL : integer := 0;
        CLKOUT0_EN : STRING := "TRUE";
        CLKOUT1_EN : STRING := " FALSE ";
        CLKOUT2_EN : STRING := " FALSE ";
```

```
    CLKOUT3_EN : STRING := " FALSE ";
    CLKOUT4_EN : STRING := " FALSE ";
    CLKOUT5_EN : STRING := " FALSE ";
CLKOUT6_EN : STRING := " FALSE ";
   CLKOUT0_DT_DIR : bit := '1';
    CLKOUT1_DT_DIR : bit := '1';
   CLKOUT2_DT_DIR : bit := '1';
    CLKOUT3_DT_DIR : bit := '1';
    CLKOUT0_DT_STEP : integer := 0;
    CLKOUT1_DT_STEP : integer := 0;
    CLKOUT2_DT_STEP : integer := 0;
    CLKOUT3_DT_STEP : integer := 0;
   CLK0_IN_SEL    : bit := '0';
    CLK0_OUT_SEL : bit := '0';
    CLK1_IN_SEL    : bit_vector := '0';
    CLK1_OUT_SEL : bit := '0';
    CLK2_IN_SEL    : bit_vector := '0';
    CLK2_OUT_SEL : bit := '0';
    CLK3_IN_SEL    : bit_vector := '0';
    CLK3_OUT_SEL : bit := '0';
CLK4_IN_SEL    : bit_vector := "00";
    CLK4_OUT_SEL : bit := '0';
CLK5_IN_SEL    : bit_vector := '0';
    CLK5_OUT_SEL : bit := '0';
CLK6_IN_SEL    : bit_vector := '0';
    CLK6_OUT_SEL : bit := '0';
    CLKFB_SEL : STRING := "INTERNAL";
    DYN_DPA_EN : STRING := "FALSE";
    DYN_PE0_SEL : STRING := "FALSE";
    DYN_PE1_SEL : STRING := "FALSE";
    DYN_PE2_SEL : STRING := "FALSE";
DYN_PE3_SEL : STRING := "FALSE";
    DYN_PE4_SEL : STRING := "FALSE";
    DYN_PE5_SEL : STRING := "FALSE";
    DYN_PE6_SEL : STRING := "FALSE";
```

```
        CLKOUT0_PE_COARSE : integer := 0;
        CLKOUT0_PE_FINE : integer := 0;
        CLKOUT1_PE_COARSE : integer := 0;
        CLKOUT1_PE_FINE : integer := 0;
        CLKOUT2_PE_COARSE : integer := 0;
        CLKOUT2_PE_FINE : integer := 0;
        CLKOUT3_PE_COARSE : integer := 0;
        CLKOUT3_PE_FINE : integer := 0;
        CLKOUT4_PE_COARSE : integer := 0;
        CLKOUT4_PE_FINE : integer := 0;
        CLKOUT5_PE_COARSE : integer := 0;
        CLKOUT5_PE_FINE : integer := 0;
        CLKOUT6_PE_COARSE : integer := 0;
        CLKOUT6_PE_FINE : integer := 0;
        DE0_EN : STRING := "FALSE";
        DE1_EN : STRING := "FALSE";
        DE2_EN : STRING := "FALSE";
      DE3_EN : STRING := "FALSE";
        DE4_EN : STRING := "FALSE";
        DE5_EN : STRING := "FALSE";
        DE6_EN : STRING := "FALSE";
        RESET_I_EN : STRING := "FALSE";
        RESET_O_EN : STRING := "FALSE";
        ICP_SEL : std_logic_vector(5 downto 0) := "XXXXXX";
       LPR_RES : std_logic_vector(2 downto 0) := "XXX";
       LPR_CAP : bit_vector := "00";
      SSC_EN : STRING := "FALSE"
 );
PORT(
        CLKIN : IN std_logic;
        CLKFB : IN std_logic:='0';
        RESET, PLLPWD : IN std_logic:='0';
        RESET_I, RESET_O : IN std_logic:='0';
        PSSEL : IN std_logic_vector(2 downto 0);
        PSDIR,PSPULSE : IN std_logic;
```

```
                SSCPOL, SSCON: IN std_logic;
                SSCMDSEL : IN std_logic_vector(6 downto 0);
                SSCMDSEL_FRAC : IN std_logic_vector(2 downto 0);
                MDCLK, MDINC : IN std logic;
                MDOPC : IN std logic vector(1 downto 0);
                MDWDI : IN std logic vector(7 downto 0);
                MDRDO : OUT std logic vector(7 downto 0);
                LOCK : OUT std_logic;
                CLKOUT0, CLKOUT1 : OUT std_logic;
                CLKOUT2, CLKOUT3 : OUT std_logic;
                CLKOUT4, CLKOUT5 : OUT std_logic;
                CLKOUT6, CLKFBOUT : OUT std_logic
        );
END COMPONENT;
uut:PLLA
        GENERIC MAP(
                FCLKIN => "100.0",
                IDIV_SEL => 1,
                FBDIV_SEL => 1,
                ODIV0_SEL => 8,
                ODIV1_SEL => 8,
                ODIV2_SEL => 8,
                ODIV3_SEL => 8,
                ODIV4_SEL => 8,
                ODIV5_SEL => 8,
                ODIV6_SEL => 8,
                MDIV_SEL => 8,
                MDIV_FRAC_SEL => 0,
                ODIV0_FRAC_SEL => 0,
                CLKOUT0_EN => "TRUE",
                CLKOUT1_EN => " FALSE ",
                CLKOUT2_EN => " FALSE ",
                CLKOUT3_EN => " FALSE ",
                CLKOUT4_EN => " FALSE ",
                CLKOUT5_EN => " FALSE ",
```

```
CLKOUT6_EN => " FALSE ",
    CLKOUT0_DT_DIR => '1',
    CLKOUT1_DT_DIR => '1',
    CLKOUT2_DT_DIR => '1',
    CLKOUT3_DT_DIR => '1',
    CLKOUT0_DT_STEP => 0,
    CLKOUT1_DT_STEP => 0,
    CLKOUT2_DT_STEP => 0,
    CLKOUT3_DT_STEP => 0,
   CLK0_IN_SEL => '0',
    CLK0_OUT_SEL => '0',
    CLK1_IN_SEL => '0',
    CLK1_OUT_SEL => '0',
    CLK2_IN_SEL => '0',
    CLK2_OUT_SEL => '0',
    CLK3_IN_SEL => '0',
    CLK3_OUT_SEL => '0',
CLK4_IN_SEL => "00",
    CLK4_OUT_SEL => '0',
CLK5_IN_SEL => '0',
    CLK5_OUT_SEL => '0',
CLK6_IN_SEL => '0',
    CLK6_OUT_SEL => '0',
    CLKFB_SEL=> "INTERNAL",
    DYN_DPA_EN => "FALSE",
    DYN_PE0_SEL => "FALSE",
    DYN_PE1_SEL => "FALSE",
    DYN_PE2_SEL => "FALSE",
DYN_PE3_SEL => "FALSE",
    DYN_PE4_SEL => "FALSE",
    DYN_PE5_SEL => "FALSE",
    DYN_PE6_SEL => "FALSE",
    CLKOUT0_PE_COARSE => 0,
    CLKOUT0_PE_FINE => 0,
    CLKOUT1_PE_COARSE => 0,
```

```
        CLKOUT1_PE_FINE => 0,
        CLKOUT2_PE_COARSE=> 0,
        CLKOUT2_PE_FINE => 0,
        CLKOUT3_PE_COARSE => 0,
        CLKOUT3_PE_FINE => 0,
        CLKOUT4_PE_COARSE => 0,
        CLKOUT4_PE_FINE => 0,
        CLKOUT5_PE_COARSE => 0,
        CLKOUT5_PE_FINE => 0,
        CLKOUT6_PE_COARSE => 0,
        CLKOUT6_PE_FINE => 0,
        DE0_EN => "FALSE",
        DE1_EN => "FALSE",
        DE2_EN => "FALSE",
      DE3_EN => "FALSE",
        DE4_EN => "FALSE",
        DE5_EN => "FALSE",
        DE6_EN => "FALSE",
        RESET_I_EN => "FALSE",
        RESET_O_EN => "FALSE",
        ICP_SEL => "XXXXXX",
       LPR_RES => "XXX",
       LPR_CAP => "00",
      SSC_EN => "FALSE"
)
 PORT MAP(
LOCK=>lock,
   CLKOUT0=>clkout0,
   CLKOUT1=>clkout1,
   CLKOUT2=>clkout2,
   CLKOUT3=>clkout3,
   CLKOUT4=>clkout4,
   CLKOUT5=>clkout5,
   CLKOUT6=>clkout6,
   CLKFBOUT=>clkfbout,
```

```
        CLKIN=>clkin,
        CLKFB=>clkfb,
        RESET=>reset,
        PLLPWD=>pllpwd,
        RESET_I=>reseti,
        RESET_O=>reseto,
        PSSEL=>pssel,
        PSDIR=>psdir,
        PSPULSE=>pspulse,
        SSCPOL=>sscpol,
        SSCON=>sscon,
        SSCMDSEL=>sscmdsel,
          SSCMDSEL_FRAC=>sscmdsel_frac,
          MDRDO=>mdrdo,
          MDWDI=>mdwdi,
          MDCLK=>mdclk,
          MDOPC=>mdopc,
          MDAINC=>mdainc
    );
```

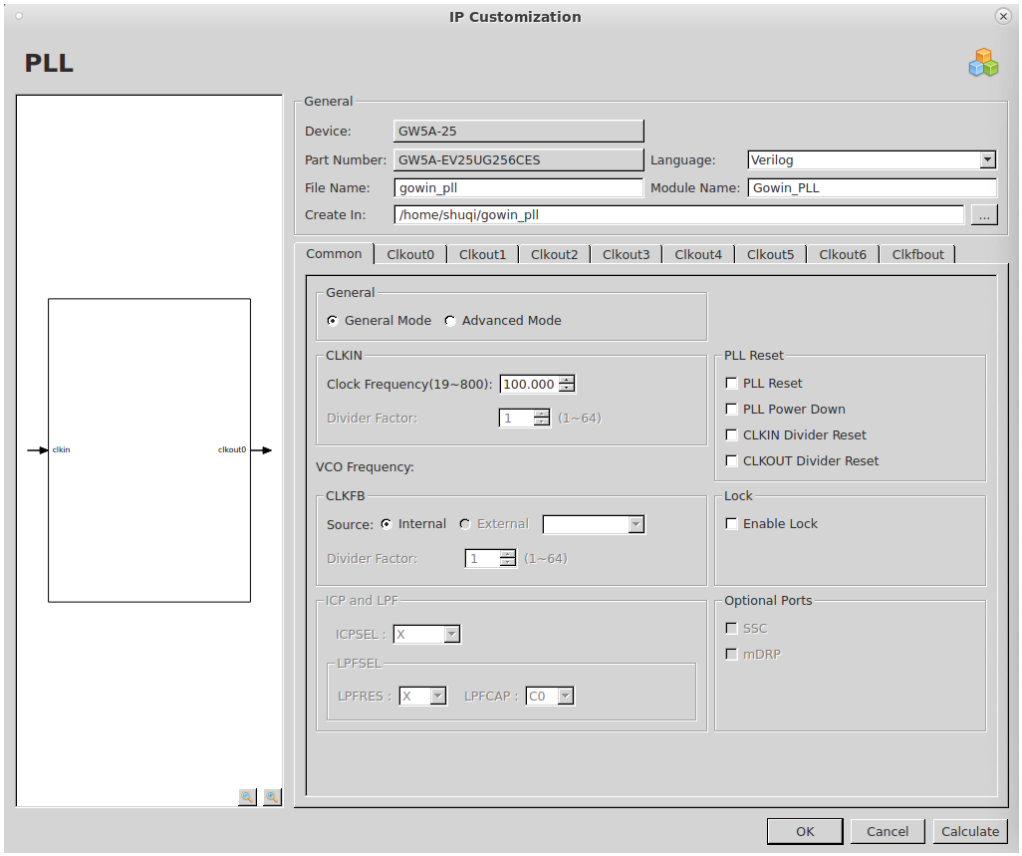
5.2.2 IP Generation

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

IP Configuration

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

Figure 5-10 IP Customization of PLLA



1. General
   The General configuration box is used to configure the generated IP design file. The PLLA General configuration is similar to that of DCE. 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, The PLLA General configuration is similar to that of PLL. For the details, please refer to the Options description in 5.1.2 IP Generation.
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 5-10.

IP Generated Files

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

- "gowin_plla.v" file is a complete Verilog module to generate instantiated PLLA, and it is generated according to the IP configuration.
- "gowin_plla_tmp.v" is the template file.
- "gowin_plla.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 DQS

Primitive Introduction

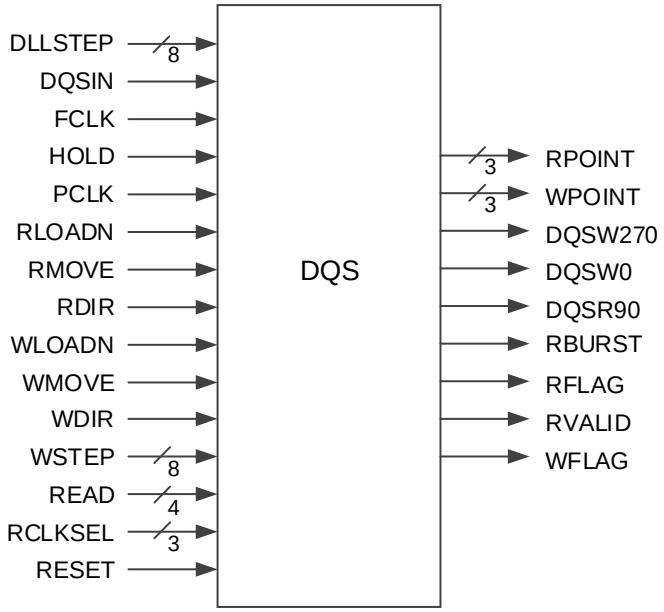
DQS is a bi-directional data strobe pulse circuit for DDR memory interface.

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 to complete write balance and read calibration.

Port Diagram

Figure 5-11 DQS Port Diagram



Port Description

Table 5-21 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 "0" increases the delay.

Port Name	I/O	Description
		"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 ● "0" increases the delay. ● "1" decreases the delay.
WLOADN	input	Resets the final delay step of 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.
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 5-22 DQS Parameter Description

Name	Value	Default	Description
FIFO_MODE_SEL	1'b0 , 1'b1	1'b0	FIFO mode selected ● 1'b0: DDR memory mode ● 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",	"X1"	DQS mode selected

Name	Value	Default	Description
	"X2_DDR3_EXT"		
HWL	"false", "true"	"false"	update0/1 timing relationship control ● "False ": update1 is one cycle ahead of update0. ● "True ": the timing of update1 and update 0 are the same.

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
 );
```

5.4 DDRDLL

5.4.1 Primitive Introduction

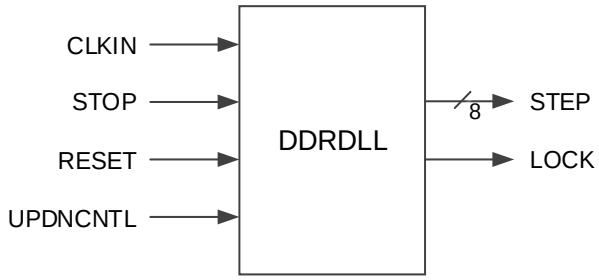
DDRDLL acts on DDR and other interface modules through DLLDLY, DQS and other modules when DDRDLL generates different phase delays.

Functional Description

DDRDLL can generate different phase delay steps based on the given input clock for clock phase shift. The calculated output STEP signal will drive DQS, DLLDLY modules, and at the same time the STEP can also be routed to user logic. The clock input sources of DDRDLL include GCLK and the adjacent HCLK.

Port Diagram

Figure 5-12 DDRDLL Port Diagram



Port Description

Table 5-23 DDRDLL Port Description

Port	I/O	Description
STEP[7:0]	Output	Delay step output
LOCK	Output	Lock indicator ● 1: Lock ● 0: Unlock
CLKIN	Input	Clock input
STOP	Input	Stop clock input and internal oscillation clock
RESET	Input	Asynchronous reset input, active-high.
UPDNCNTL	Input	DDRDLL delay step update control, active-low.

Parameter Description

Table 5-24 DDRDLL Parameter Description

Parameter	Type	Value	Default	Description
DLL_FORCE	Integer	0,1	0	Forced delay step and lock control of DDRDLL

Parameter	Type	Value	Default	Description
				● TRUE: Forced lock with a delay step of 255 (maximum) for lower input frequency mode. ● FALSE: Normal mode, delay step and lock signal generated by DDRDLL.
CODESCAL	String	000,001,010,011,100,101, 110, 111	000	DDRDLL phase shift configuration (45°~135°) ● 000:101° ● 001:112° ● 010:123° ● 011:135° ● 100:79° ● 101:68° ● 110:57° ● 111:45°
SCAL_EN	String	true,false	true	DDRDLL phase shift enable: ● TRUE: Enabled, phase shift is set according to the parameter CODESCAL. ● FALSE: Disabled, 90° phase shift by default.
DIV_SEL	Integer	1’b0,1’b1	1’b0	DDRDLL lock mode selected: ● 1'b0: Normal ● 1'b1: Fast

Connection Rule

The output STEP of DDRDLL can be connected to DQS, DLLDLY modules and also can be routed to user logic.

Primitive Instantiation

Verilog Instantiation:

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

```
defparam uut.SCAL_EN = " TRUE";
defparam uut.DIV_SEL = 1'b0;
```

Vhdl Instantiation:

```
COMPONENT DLL
        GENERIC(
                DLL_FORCE: STRING := "FALSE";
                DIV_SEL: bit :='1';
                CODESCAL: STRING :="000";
                SCAL_EN: STRING :="true"
        );
            PORT(
                CLKIN:IN std_logic;
                STOP:IN std_logic;
                RESET:IN std_logic;
                UPDNCNTL:IN std_logic;
                LOCK:OUT std_logic;
                STEP:OUT std_logic_vector(7 downto 0)
             );
END COMPONENT;
uut:DLL
    GENERIC MAP(
                  DLL_FORCE=>” FALSE”,
                  DIV_SEL=>'1',
                  CODESCAL=>"000",
                  SCAL_EN=>" TRUE"
    )
     PORT MAP(
        CLKIN=>clkin,
        STOP=>stop,
        RESET=>reset,
        UPDNCNTL=>updncntl,
        LOCK=>lock,
        STEP=>step
        );
```

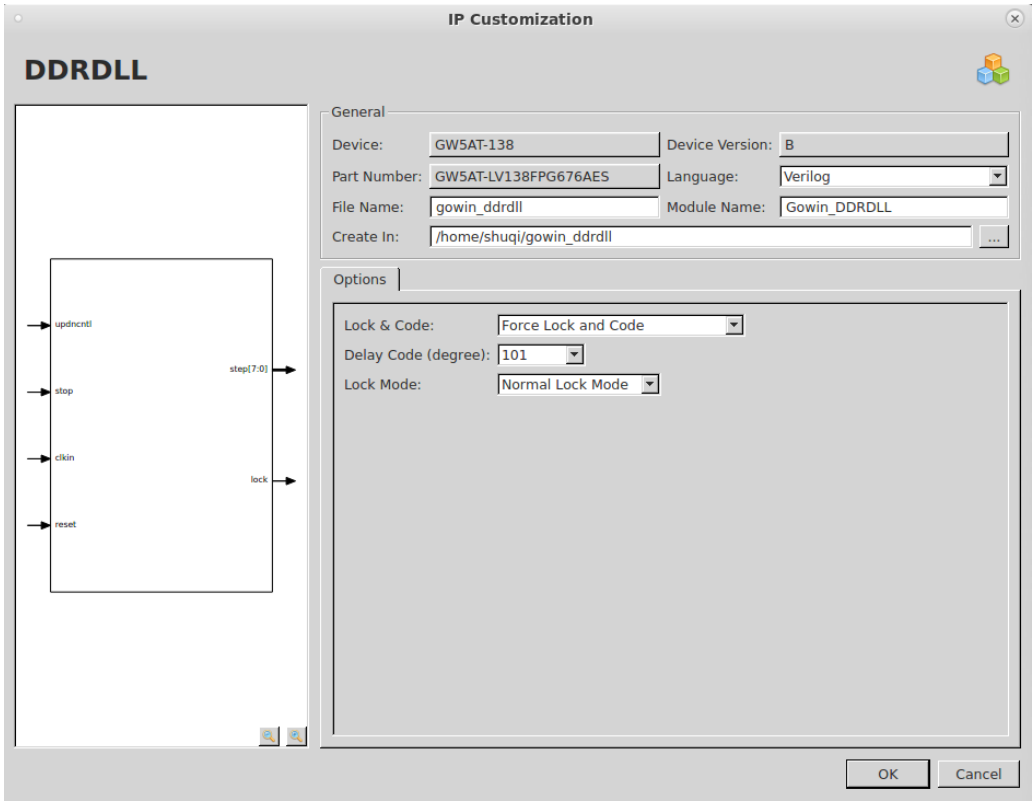
5.4.2 IP Generation

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

IP Configuration

Double-click on "DDRDLL", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 5-13.

Figure 5-13 IP Customization of DDRDLL



1. General
   The General configuration box is used to configure the generated IP design file. The DDRDLL General configuration is similar to that of DCE. 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 5-13.
   - Lock & Code: Set DDRDLL mode
   - Delay Code(degree): Set delay
   - Lock Mode: Set lock mode
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 5-13.

IP Generated Files

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

- "gowin_ddrdll.v" file is a complete Verilog module to generate instantiated DDRDLL, and it is generated according to the IP configuration.
- "gowin_ddrdll_tmp.v" is the template file.
- "gowin_ddrdll.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.

6 Crystal Oscillator Clock

6.1 OSC

6.1.1 Primitive Introduction

OSC, programmable on-chip crystal oscillator.

Device Supported

Table 6-1 OSC Device Supported

Product Family	Series	Device
Arora	GW5AT	GW5AT-138, GW5AT-138B
	GW5AST	GW5AST-138B

Functional Description

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

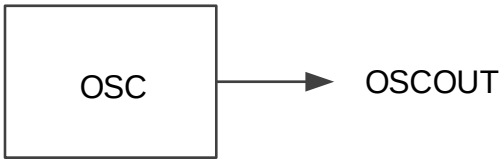
The output clock frequency of the device can be calculated by the following formula:

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

f_{osc} is the OSC oscillation frequency, the divisor FREQ_DIV is the configuration parameter, and the range is the even numbers from 2 to 128 and 3.

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	3,2~126 (even)	100	OSC frequency division coefficient setting

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

VHDL Instantiation:

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

6.1.2 IP Generation

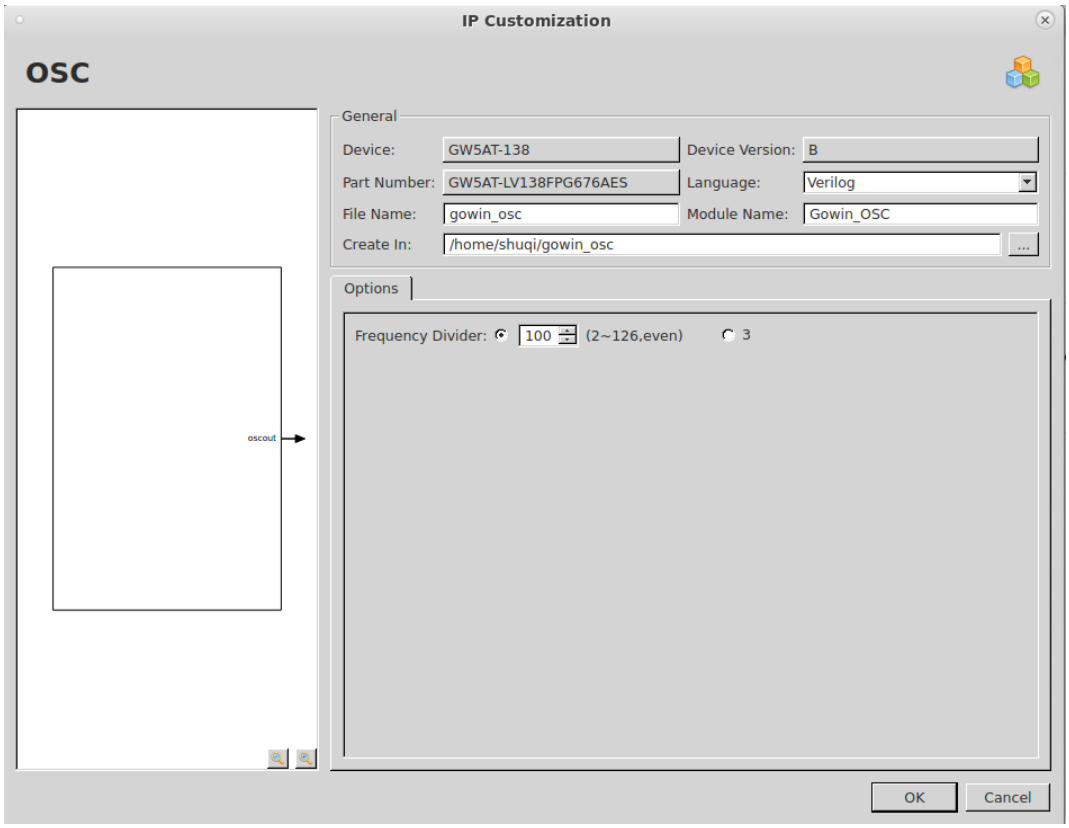
Click "OSC" on the "IP Core Generator" interface and an overview of OSC will be displayed on the right side of the interface.

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

6-2.

Figure 6-2 IP Customization of OSC



1. General
   The General configuration box is used to configure the generated IP design file. The OSC General configuration is similar to that of DCE. 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 6-2.
   - Frequency Divider: Set DDRDLL mode
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 6-2.

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.

6.2 OSCA

6.2.1 Primitive Introduction

OSCA, programmable on-chip crystal oscillator.

Device Supported

Table 6-4 OSCA Device Supported

Product Family	Series	Device
Arora	GW5A	GW5A-25

Functional Description

The programmable on-chip oscillator provides a clock source for MSPI programming mode, supports OSC dynamic on/off, and 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:

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

f_{osc} is the OSC oscillation frequency, the divisor FREQ_DIV is the configuration parameter, and the range is even numbers from 2 to 126 and 3.

Port Diagram

Figure 6-3 OSCA Port Diagram

Port Description

Table 6-5 OSCA Port Description

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

Parameter Description

Table 6-6 OSCA Parameter Description

Name	Value	Default	Description
FREQ_DIV	3,2~126 (even)	100	OSC frequency division coefficient setting

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

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

VHDL Instantiation:

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

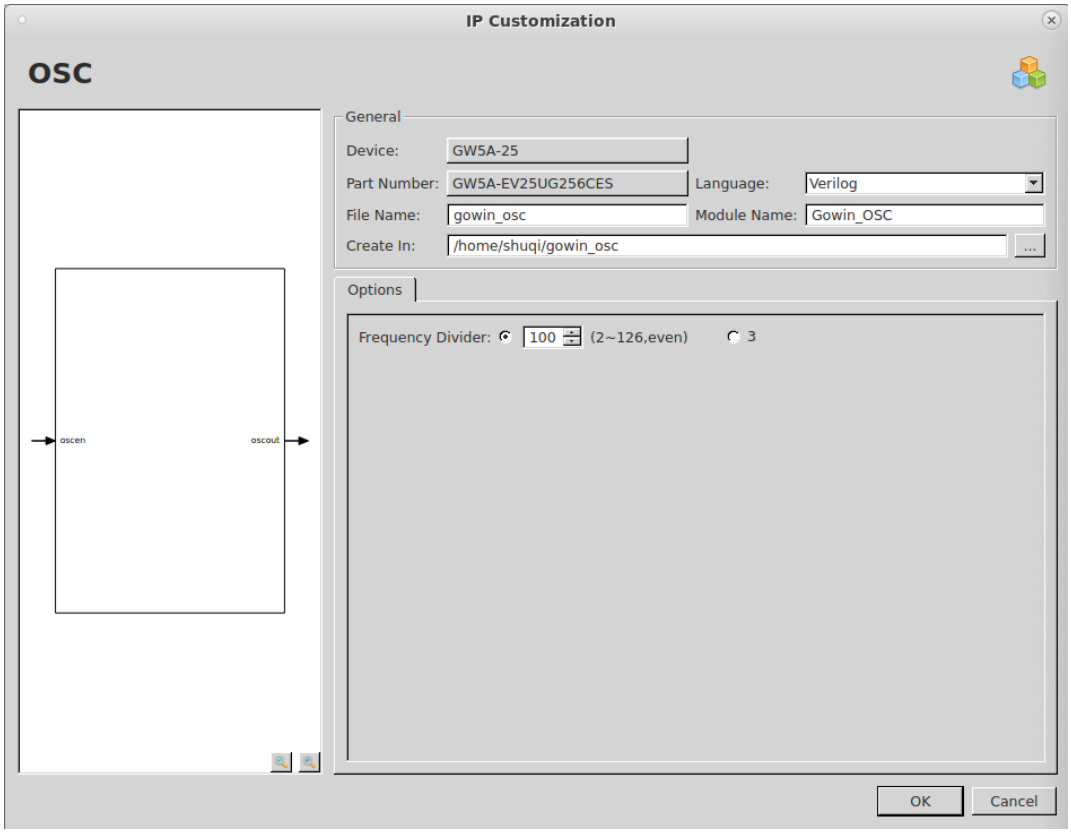
6.2.2 IP Generation

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

IP Configuration

Double-click on "OSCA", and the "IP Customization" window pops up. It includes the "General", "Options", and port diagram, as shown in Figure 6-4.

Figure 6-4 IP Customization of OSCA



1. General
   The General configuration box is used to configure the generated IP design file. The OSCA General configuration is similar to that of DCE. 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 6-4. The OSCA General configuration is similar to that of OSC. For the details, please refer to the Options description in 6.1.2 IP Generation.
3. Port Diagram
   The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 6-4.

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 OSCA, 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