Arora V Clock
ユーザーガイド

UG306-1.0J, 2023-04-20

免責事項

当社は、GOWINSEMI Terms and Conditions of Sale(GOWINSEMI取引条件)に規定されている内容を除き、(明示的か又は黙示的かに拘わらず)いかなる保証もせず、また、知的財産権や材料の使用によりあなたのハードウェア、ソフトウェア、データ、又は財産が被った損害についても責任を負いません。当社は、事前の通知なく、いつでも本文書の内容を変更することができます。本文書を参照する何れの団体及び個人も、最新の文書やエラッタ(不具合情報)については、当社に問い合わせる必要があります。

バージョン履歴

日付	バージョン	説明
2023/04/20	1.0J	初版。

目次

図一覧

表一覧

1 本マニュアルについて

1.1 マニュアル内容

本マニュアルでは、Arora V FPGA 製品のクロックリソースの機能、プリミティブの定義、およびその使用法について説明します。

1.2 関連ドキュメント

GOWIN セミコンダクターの公式 Web サイト www.gowinsemi.com/ja から、以下の関連ドキュメントがダウンロード、参考できます：

- GW5AT シリーズ FPGA 製品データシート(DS981)
- GW5A シリーズ FPGA 製品データシート(DS1103)
- GW5AST シリーズ FPGA 製品データシート(DS1104)
- Gowin ソフトウェア ユーザーガイド(SUG100)

1.3 用語、略語

表 1-1 に、本マニュアルで使用される用語、略語、及びその意味を示します。

表 1-1 用語、略語

用語、略語	正式名称	意味
CLKDIV	Clock Divider	クロック分周器
CLKDIV2	Clock Divider 2	高速クロック分周器
DCE	Dynamic Clock Enable	ダイナミック・クロック・イネーブル
DCS	Dynamic Clock Selector	ダイナミック・クロック・セレクタ
DHCE	Dynamic HCLK Clock Enable	ダイナミック高速クロック・イネーブル
DLLDLY	DLL Delay	DLL 遅延
DQS	Bidirectional Data Strobe Circuit for DDR Memory	双方向データストローブ回路

用語、略語	正式名称	意味
GCLK	Global Clock	グローバルクロック
HCLK	High-speed Clock	高速クロック
LW	Long Wire	ロングワイヤ
OSC	Oscillator	オシレータ
PCLK	Primary Clock	プライマリクロック
PLL	Phase-locked Loop	位相同期回路
SSC	Spread Spectrum Clocking	スペクトラム拡散クロック

1.4 テクニカル・サポートとフィードバック

GOWIN セミコンダクターは、包括的な技術サポートをご提供しています。使用に関するご質問、ご意見については、直接弊社までお問い合わせください。

Web サイト：www.gowinsemi.com/ja

E-mail：support@gowinsemi.com

2 概要

本章では、GOWIN セミコンダクターArora V FPGA 製品の、専用のクロック入力およびクロック配線リソースを含むクロックリソースについて紹介します。基本的なクロックリソースとして、高周波信号に適した低電気容量、低スキューの配線が提供されています。これらのリソースは最大限までクロックスキューを減少してパフォーマンスを向上させることができ、すべてのクロック信号に適用できます。

クロックリソースは、FPGA の高性能アプリケーションにとって重要です。Arora V FPGA 製品は、デバイスをグローバルに直接駆動する専用のグローバルクロック・ネットワーク GCLK(PCLK および LW を含む)を提供しています。さらに、位相同期回路(PLL)、高速クロック(HCLK)、および DQS 等のクロックリソースも提供されています。

2.1 グローバルクロック

GCLK には、PCLK と LW が含まれています。PCLK はすべてのリソースに直接接続することができます。LW は、DFF にクロックイネーブル(CE)およびセット/リセット(SET/RESET)信号を提供するための制御ラインとして使用できると同時に、論理配線として、つまり通常のデータ信号として使用することもできます。

2.2 高速クロック

低ジッタと低スキューという特徴を備えた高速クロック(HCLK)は、I/O の高性能のデータ転送を可能にし、主にソース同期データ転送インターフェースに適しています。1 つの Bank では 4 つの HCLK がサポートされます。

3 グローバルクロック

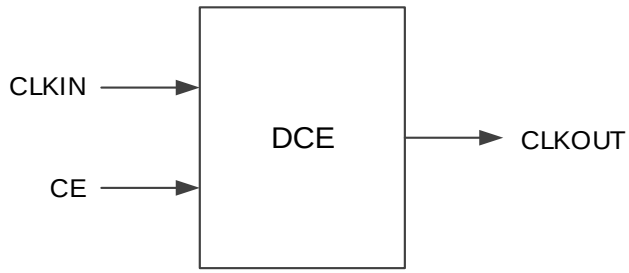
3.1 DCE

3.1.1 プリミティブの紹介

DCE(ダイナミック・クロック・イネーブル)は、GCLK ネットワークを内部ロジックにより動的に有効または無効にすることを可能にします。GCLK クロックネットワークが無効にされると、そのクロックによって駆動されるすべてのロジックは反転しなくなるため、デバイスの全体的な消費電力が低下します。

ポート図

図 3-1 DCE のポート図



ポートの説明

表 3-1 DCE のポートの説明

ポート名	I/O	説明
CLKIN	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
CLKOUT	出力	クロック出力信号

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

```
DCE uut (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

VHDL でのインスタンス化：

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

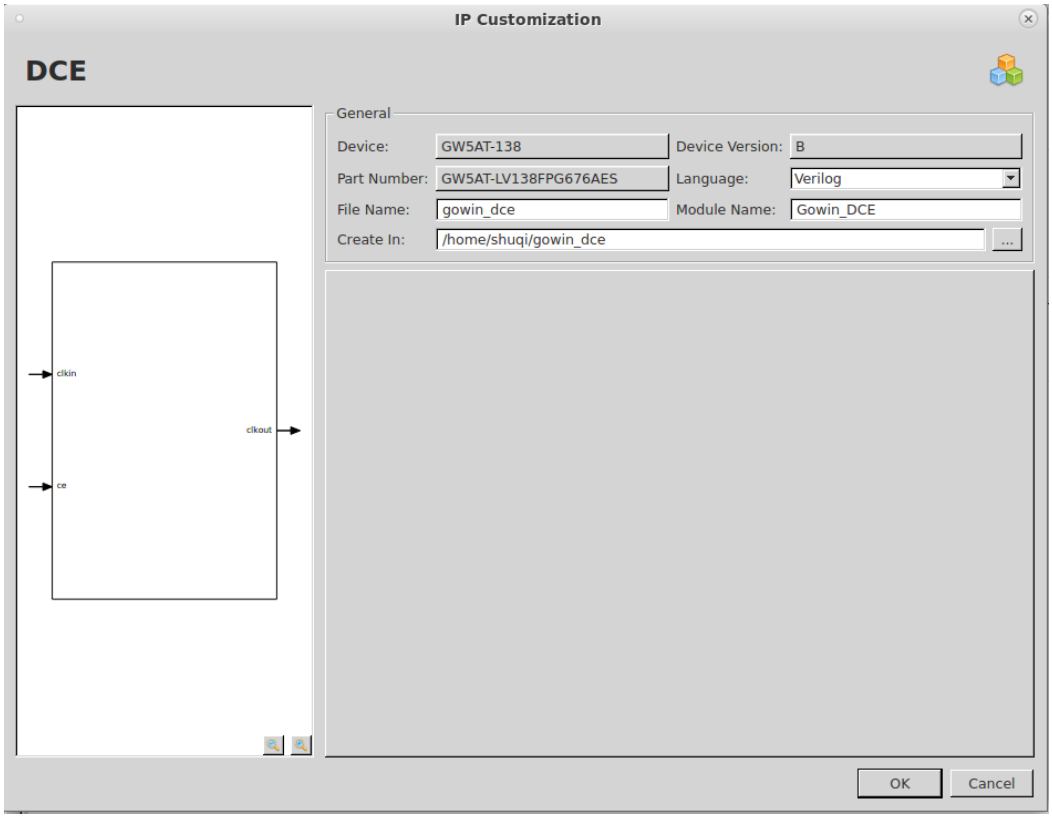
3.1.2 IP の呼び出し

IP Core Generator のインターフェースで“DCE”をクリックすると、右側に DCE の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“DCE”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、“General”構成タブおよびポート図があります(図 3-2)。

図 3-2 DCE IP の構成ウィンドウ



1. General 構成タブ

General 構成タブは、IP ファイルの構成に使用されます。

- Device：対象デバイス。
- Device Version：デバイスのバージョン。
- Part Number：部品番号。
- Language：IP を実現するハードウェア記述言語。右側のドロップダウン・リストからターゲット言語(Verilog または VHDL)を選択します。
- File Name：生成される IP ファイルのファイル名。右側のテキストボックスで再編集できます。
- Module Name：生成される IP ファイルのモジュール名。右側のテキストボックスで編集できます。Module Name をプリミティブ名と同じにすることはできません。同じである場合、エラーが報告されます。
- Create In：生成される IP ファイルのパス。右側のテキストボックスでパスを直接編集するか、テキストボックスの右側にある選択ボタンを使用してパスを選択できます。

2. ポート図

ポート図は、IP Core の構成結果を表示します(図 3-2)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_dce.v”は完全な verilog モジュールです。
- “gowin_dce_tmp.v”は IP のテンプレートファイルです。
- “gowin_dce.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

3.2 DCS

3.2.1 プリミティブの紹介

DCS(ダイナミック・クロック・セレクタ)により、CLKOUT を 4 つのクロック入力間で動的に切り替えることができます。

機能の説明

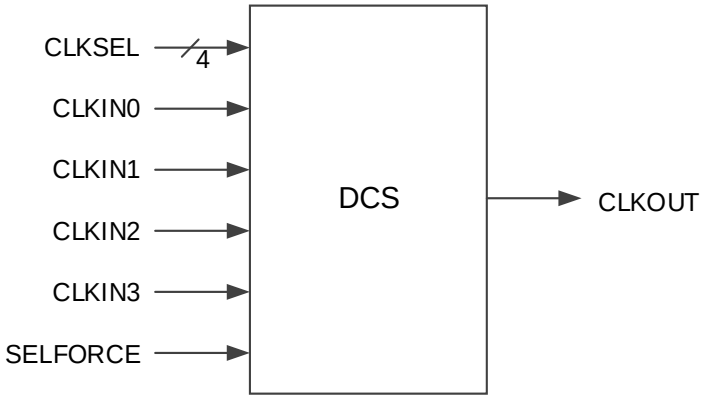
DCS には、「Non-Glitchless」モードと「Glitchless」モードの 2 つのクロック切り替えモードがあります。

Non-Glitchless モードでは、DCS は通常のマルチプレクサとして機能し、CLKSEL 信号のみを利用してクロック信号を切り替えます。出力のグリッチが許容されます。

Glitchless モードでは、パラメータ DCS_MODE を使用してモードを設定できます。CLKSEL 信号を構成してクロック信号を動的に切り替えることにより、出力クロックのグリッチを回避できます。

ポート図

図 3-3 DCS のポート図



ポートの説明

表 3-2 DCS のポートの説明

ポート名	I/O	説明
CLKIN0	入力	クロック入力信号 0
CLKIN1	入力	クロック入力信号 1
CLKIN2	入力	クロック入力信号 2
CLKIN3	入力	クロック入力信号 3
CLKSEL	入力	クロック選択信号
SELFORCE	入力	強制モード選択 ● 0：glitchless モード ● 1：Non-glitchless モード
CLKOUT	出力	クロック出力信号

パラメータの説明

表 3-3 DCS のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
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"	DCS モードを設定します。

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

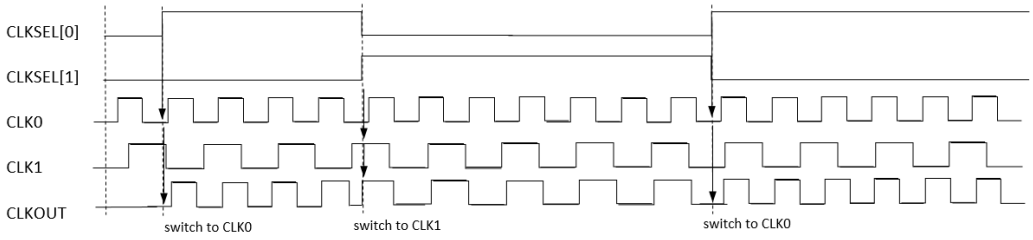
VHDL でのインスタンス化：

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

タイミング図

Non-Glitchless モードのタイミングは図 3-4 に示すとおりです。CLKSEL[3]~CLKSEL[0]はそれぞれ CLKIN3～CLKIN0 を選択するために使用され、アクティブ High で、切り替えタイミングは同じです。

図 3-4 Non-Glitchless モードのタイミング図



Glitchless モードのタイミングは図 3-5～図 3-8 に示すとおりです。CLKSEL[3]~CLKSEL[0]はそれぞれ CLKIN3～CLKIN0 を選択するために使用され、切り替えタイミングは同じです。

図 3-5 DCS mode が RISING の場合のタイミング

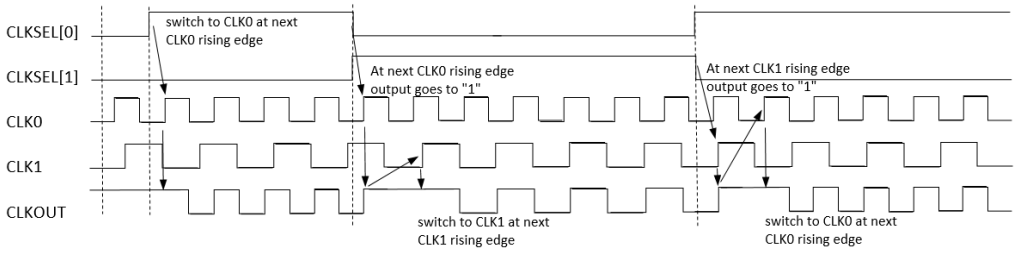


図 3-6 DCS mode が FALLING の場合のタイミング

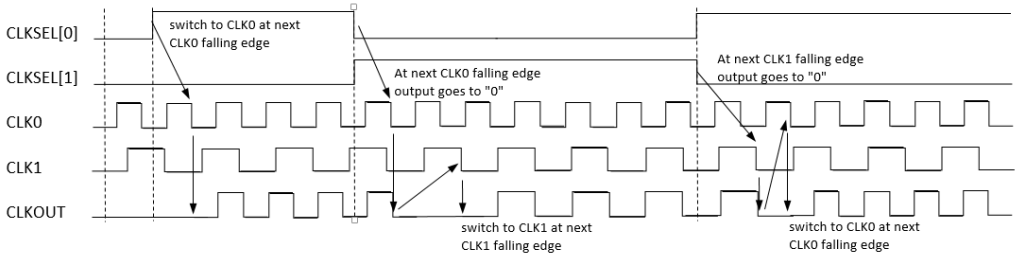


図 3-7 DCS mode が CLK0_GND の場合のタイミング

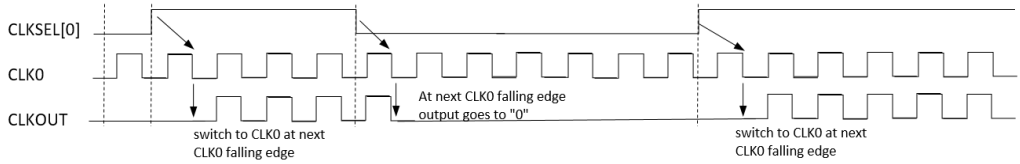
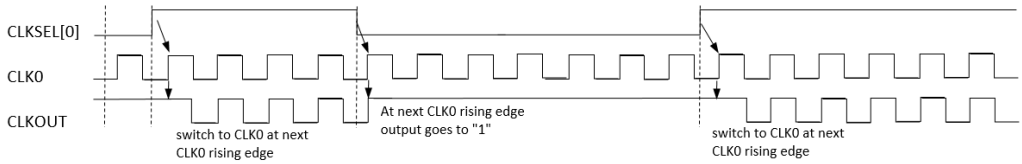


図 3-8 DCS mode が CLK0_VCC の場合のタイミング



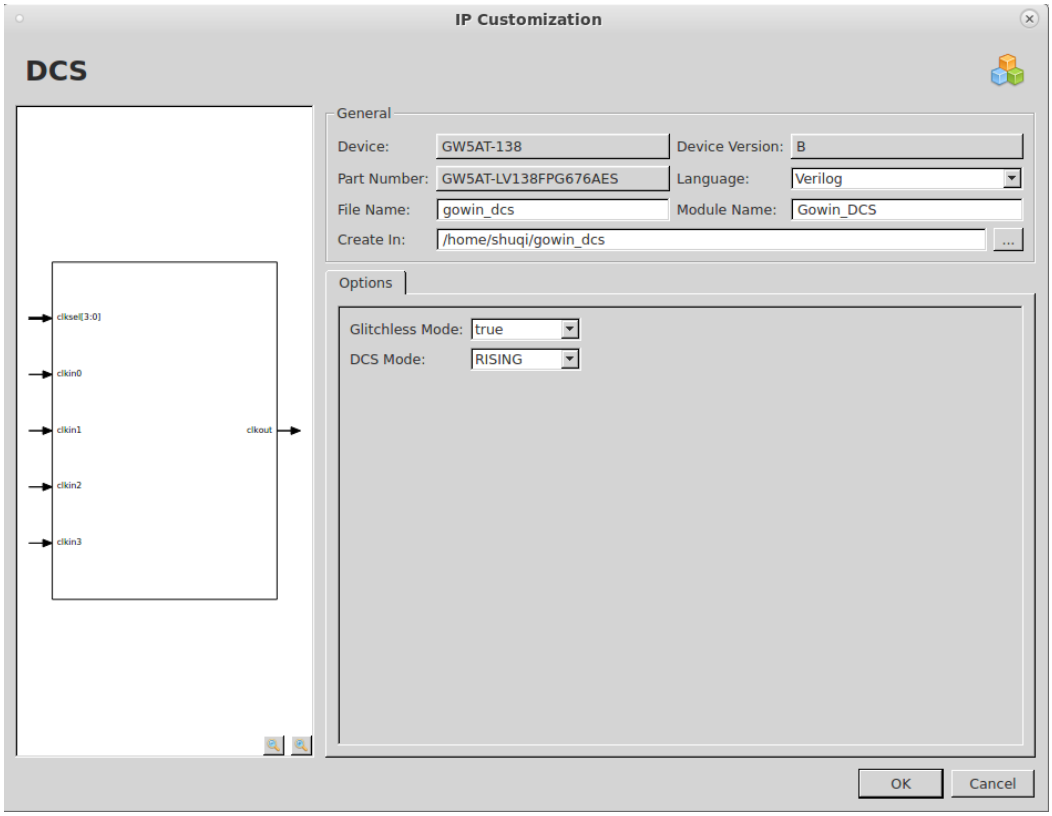
3.2.2 IP の呼び出し

IP Core Generator のインターフェースで“DCS”をクリックすると、右側に DCS の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“DCS”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブ、Options 構成タブ、およびポート図があります(図 3-9)。

図 3-9 DCS IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。DCS の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. Options 構成タブ
   Options 構成タブは IP のカスタマイズに使用されます(図 3-9)。
   - Glitchless Mode：Glitchless モードのイネーブルオプション。
   - DCS Mode：DCS モードの設定。
3. ポート図
   ポート図は、IP Core の構成結果を表示します(図 3-9)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_dcs.v”は完全な verilog モジュールです。
- gowin_dcs_tmp.v は IP のテンプレートファイルです。
- gowin_dcs.ipc は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

4 高速クロック

4.1 DHCE

4.1.1 プリミティブの紹介

DHCE は、HCLK 高速クロック信号を動的にオン/オフすることができます。

ポート図

図 4-1 DHCE のポート図

ポートの説明

表 4-1 DHCE のポートの説明

ポート名	I/O	説明
CLKIN	入力	クロック入力信号
CEN	入力	クロックイネーブル信号、アクティブ Low
CLKOUT	出力	クロック出力信号

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

```
);
```

VHDL でのインスタンス化：

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

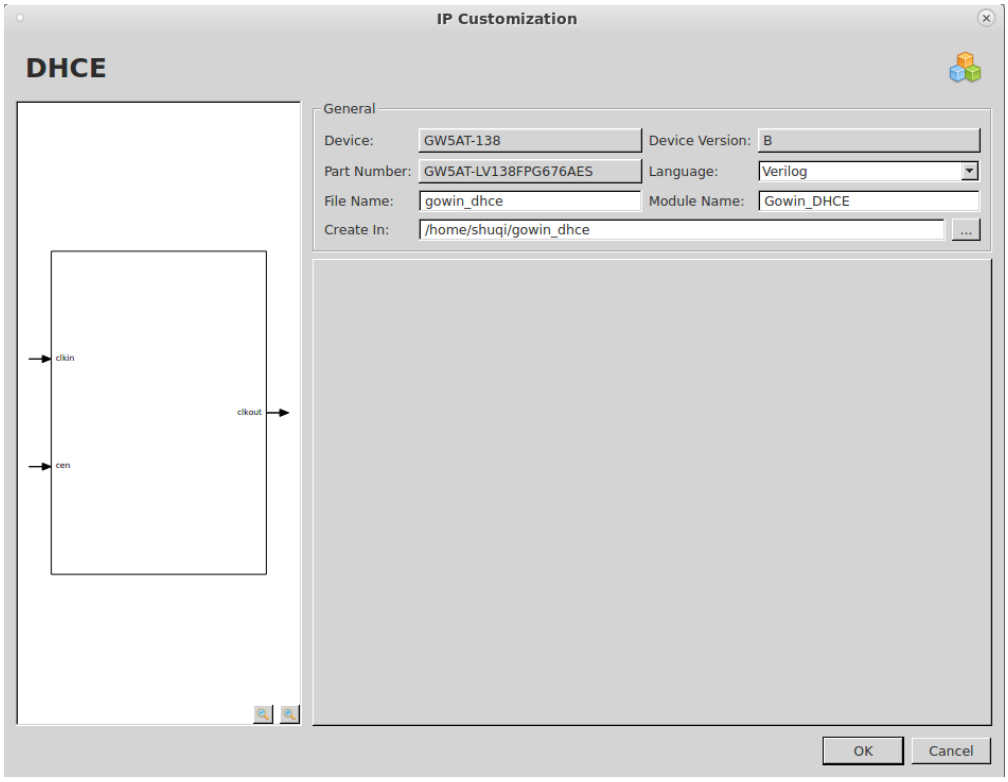
4.1.2 IP の呼び出し

IP Core Generator のインターフェースで“DHCE”をクリックすると、右側に DHCE の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“DHCE”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブおよびポート図があります(図 4-2)。

図 4-2 DHCE IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。DHCE の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. ポート図
   ポート図は、IP Core の構成結果を表示します(図 4-2)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_dhce.v”は完全な verilog モジュールです。
- “gowin_dhce_tmp.v”は IP のテンプレートファイルです。
- “gowin_dhce.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

4.2 CLKDIV2

4.2.1 プリミティブの紹介

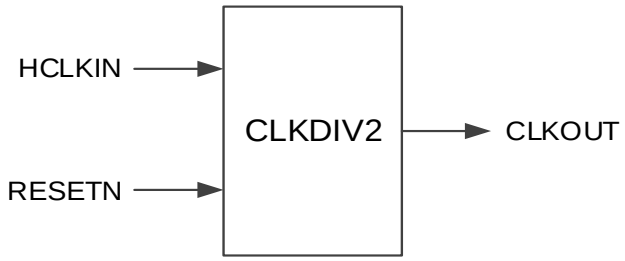
CLKDIV2 は、クロック周波数を 2 分周するクロック分周器です。CLKDIV2 の出力は、IOLOGIC の FCLK、PLL の CLKIN と CLKFB、DQS の FCLK、CLKDIV の HCLKIN、および DDRDLL の CLKIN のみを駆動可能です。

機能の説明

CLKDIV2 は、入力クロックと位相が一致する 2 分周クロックを生成する高速クロック分周モジュールです。

ポート図

図 4-3 CLKDIV2 のポート図



ポートの説明

表 4-2 CLKDIV2 のポートの説明

ポート名	I/O	説明
HCLKIN	入力	クロック入力信号
RESETN	入力	非同期リセット信号、アクティブ Low。
CLKOUT	出力	クロック出力信号

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

VHDL でのインスタンス化：

```
COMPONENT CLKDIV2
```

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

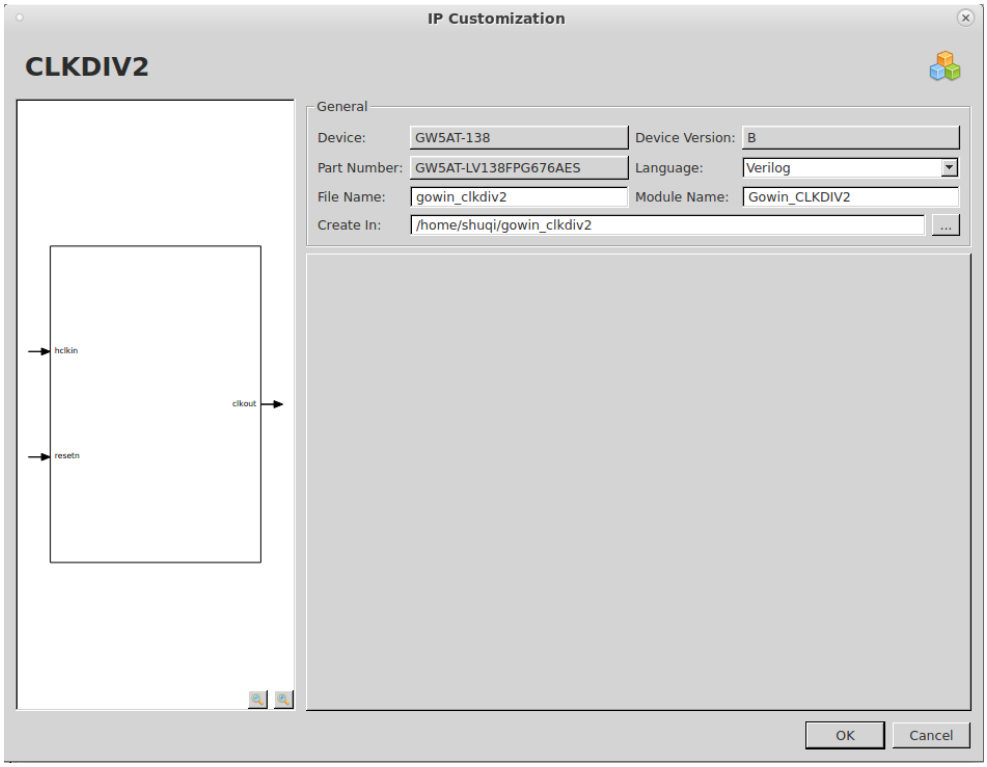
4.2.2 IP の呼び出し

IP Core Generator のインターフェースで“CLKDIV2”をクリックすると、右側に CLKDIV2 の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“CLKDIV2”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブおよびポート図があります(4)。

図 4-4 CLKDIV2 IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。CLKDIV2 の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. ポート図
   ポート図は、IP Core の構成結果を表示します(4)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_clkdiv2.v”は完全な verilog モジュールです。
- “gowin_clkdiv2_tmp.v”は IP のテンプレートファイルです。
- “gowin_clkdiv2.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

4.3 CLKDIV

4.3.1 プリミティブの紹介

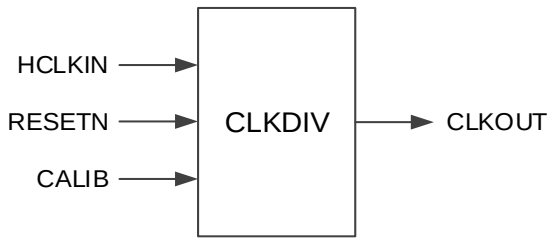
CLKDIV は、クロック周波数を調整するクロック分周器です。

機能の説明

CLKDIV は、入力クロックと位相が一致し、IO ロジックに使用される分周クロックを生成する高速クロック分周モジュールです。

ポート図

図 4-5 CLKDIV のポート図



ポートの説明

表 4-3 CLKDIV のポートの説明

ポート名	I/O	説明
HCLKIN	入力	クロック入力信号
RESETN	入力	非同期リセット信号、アクティブ Low。 DIV_MODE が 1 の場合、機能しません。
CALIB	入力	CALIB 入力。クロック出力を調整します。
CLKOUT	出力	クロック出力信号

CALIB 信号は、IOLOGIC の CALIB と併用することができます。その機能は次のとおりです。

- 2 分周の場合、2 立ち下がりエッジごとに 180 度の位相を調整し、2 回の調整で 1 サイクルとします。
- 3 分周の場合、2 立ち下がりエッジごとに約 120 度の位相を調整し、3 回の調整で 1 サイクルとします。
- 3.5 分周の場合、1 立ち下がりエッジごとに約 102.8 度の位相を調整し、7 回の調整で 1 サイクルとします。
- 4 分周の場合、2 立ち下がりエッジごとに 90 度の位相を調整し、4 回の調整で 1 サイクルとします。
- 5 分周の場合、2 立ち下がりエッジごとに約 72 度の位相を調整し、5 回の調整で 1 サイクルとします。
- 6 分周の場合、2 立ち下がりエッジごとに約 60 度の位相を調整し、6 回の調整で 1 サイクルとします。
- 7 分周の場合、2 立ち下がりエッジごとに約 51.4 度の位相を調整し、7 回の調整で 1 サイクルとします。
- 8 分周の場合、2 立ち下がりエッジごとに約 45 度の位相を調整し、8 回の調整で 1 サイクルとします。
- 1 分周の場合、調整は無効となりす。

パラメータの説明

表 4-4 CLKDIV のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
DIV_MODE	1,2,3,3.5,4,5,6,7,8	2	クロックの分周係数を設定します。

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

VHDL でのインスタンス化：

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

4.3.2 IP の呼び出し

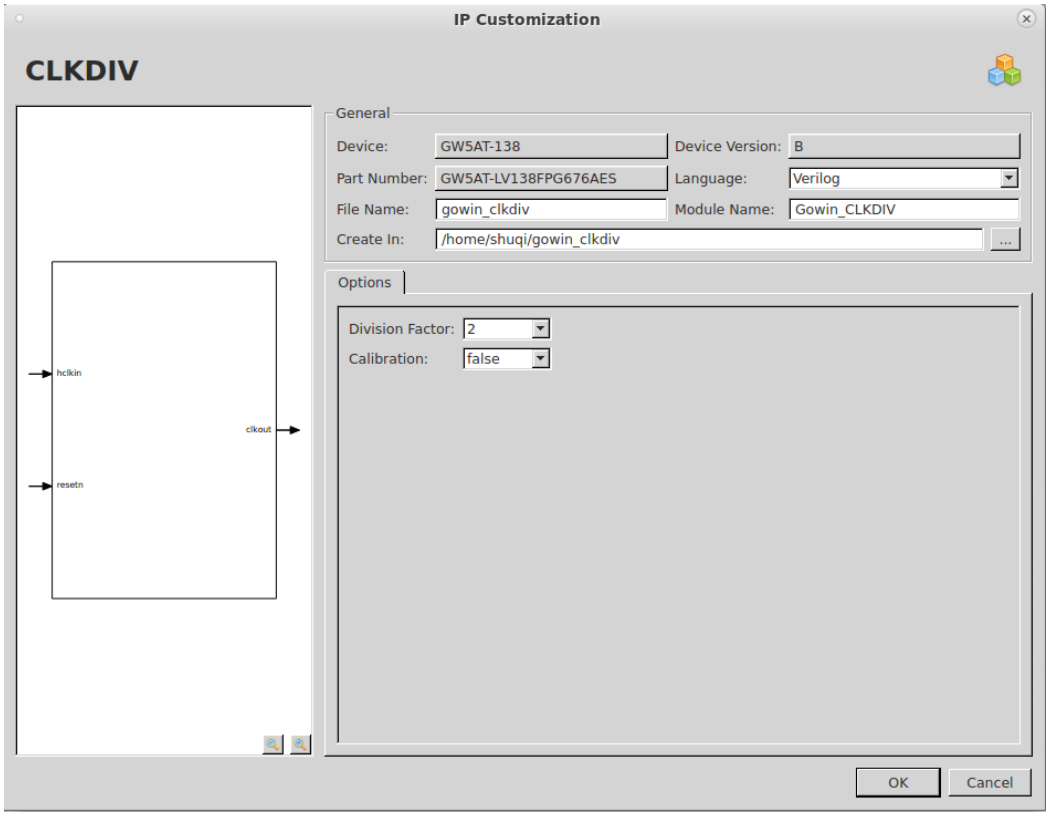
IP Core Generator のインターフェースで“CLKDIV”をクリックすると、右側に CLKDIV の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“CLKDIV”をダブルクリック

すると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブ、Options 構成タブ、およびポート図があります(図 4-6)。

図 4-6 CLKDIV IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。CLKDIV の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. Options 構成タブ
   Options 構成タブは IP のカスタマイズに使用されます(図 4-6)。
   - Division Factor：除算係数。
   - Calibration：校正クロックのイネーブルオプション。
3. ポート図
   ポート図は、IP Core の構成結果を表示します(図 4-6)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_clkdiv.v”は完全な verilog モジュールです。
- “gowin_clkdiv_tmp.v”は IP のテンプレートファイルです。
- “gowin_clkdiv.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

4.4 DLLDLY

4.4.1 プリミティブの紹介

クロック遅延モジュールとして、DLLDLY は CSTEP 信号または DLLSTEP 信号に基づきクロックを遅延して出力します。

機能の説明

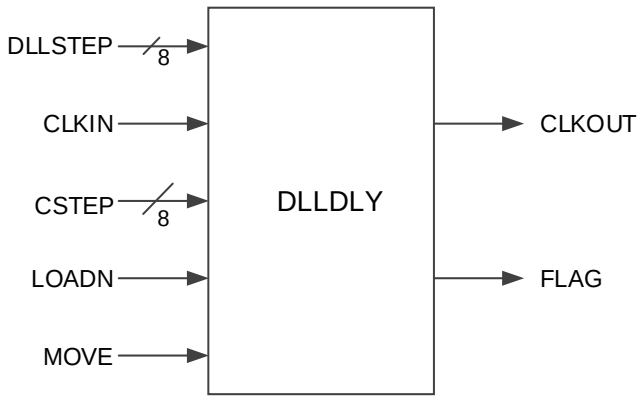
DLLDLY は、CSTEP または DLLSTEP に従って対応する位相の遅延を生成します。これによって CLKIN に基づいた遅延出力が実現されます。静的モード、動的モード、およびアダプティブモードという遅延モードがサポートされます。

注記：

GW5A-25 デバイスはアダプティブモードをサポートしません。

ポート図

図 4-7 DLLDLY のポート図



ポートの説明

表 4-5 DLLDLY のポートの説明

ポート名	I/O	説明
CLKOUT	出力	クロック出力信号
FLAG	出力	動的遅延調整の under-flow または over-flow を示すアダプティブモードの場合の出力フラグ。
DLLSTEP[7:0]	入力	DDRDLL からの遅延ステップ入力信号。
CLKIN	入力	クロック入力信号
CSTEP[7:0]	入力	CIU 配線から取得可能な遅延ステップ入力信号

ポート名	I/O	説明
LOADN	入力	遅延ステップのロードの制御 ● 0: 遅延ステップをロードします。 ● 1: 遅延を動的に調整します。
MOVE	入力	アダプティブモードの場合、MOVE の立ち下がりエッジで遅延を動的に調整し、パルスごとに 1 遅延ステップ移動します

パラメータの説明

表 4-6 DLLDLY のパラメータの説明

パラメータ名	パラメータのタイプ	値の範囲	デフォルト値	説明
DLY_SIGN	Binary	1'b0,1'b1	1'b0	遅延調整の符号を設定します： 1'b0:'+' 1'b1: '-'
DLY_ADJ	Integer	0~255	0	遅延調整を設定します： DLY_SIGN =0 DLY_ADJ; DLY_SIGN =1 - DLY_ADJ
STEP_SEL	Binary	1’b0,1’b1	1’b0	1’b0:DLLSTEP 1’b1:CSTEP
ADAPT_EN	String	"FALSE"、"TRUE"	"FALSE"	アダプティブモードのイネーブル
DYN_DLY_EN	String	"FALSE"、"TRUE"	"FALSE"	動的モードのイネーブル

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

```
DLLDLY uut (
    .CLKIN(clkin),
    .DLLSTEP(dllstep[7:0]),
    .CSTEP(cstep[7:0]),
    .LOADN(loadn),
    .MOVE(move),
    .CLKOUT(clkout),
    .FLAG(flag)
```

```
);
defparam dlldly_0.DLY_SIGN=1'b0;
defparam dlldly_0.DLY_ADJ=0;
defparam dlldly_0.DYN_DLY_EN="FALSE";
defparam dlldly_0.ADAPT_EN="FALSE";
defparam dlldly_0.STEP_SEL=1’b0;
```

VHDL でのインスタンス化：

```
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
    );
```

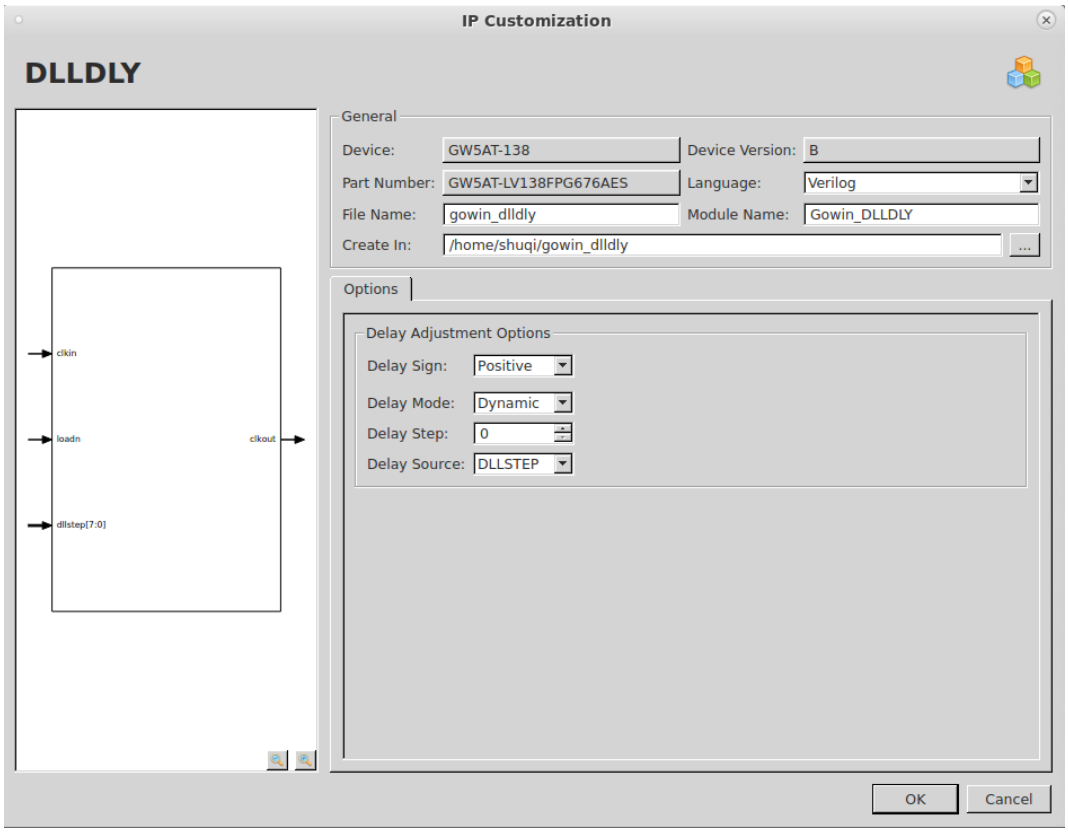
4.4.2 IP の呼び出し

IP Core Generator のインターフェースで“DLLDLY”をクリックすると、右側に DLLDLY の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“DLLDLY”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブ、Options 構成タブ、およびポート図があります(図 4-8)。

図 4-8 DLLDLY IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。DLLDLY の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. Options 構成タブ
   Options 構成タブは IP のカスタマイズに使用されます(図 4-8)。
   - Delay Sign：遅延調整の符号を設定します。
   - Delay Mode：遅延モードを設定します。
   - Delay Step：遅延ステップを設定します。
   - Delay Source：遅延ソース。

3. ポート図
   ポート図は、IP Core の構成結果を表示します(図 4-8)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_dlldly.v”は完全な verilog モジュールです。
- “gowin_dlldly_tmp.v”は IP のテンプレートファイルです。
- “gowin_dlldly.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

5 システムクロック

5.1 PLL

5.1.1 プリミティブの紹介

Arora V FPGA は、7 つのクロック出力をサポートする位相同期回路 PLL を提供します。各出力クロックは、リファレンスクロックに基づいて周波数、位相、およびデューティサイクルを独立して調整することをサポートします。

サポートされるデバイス

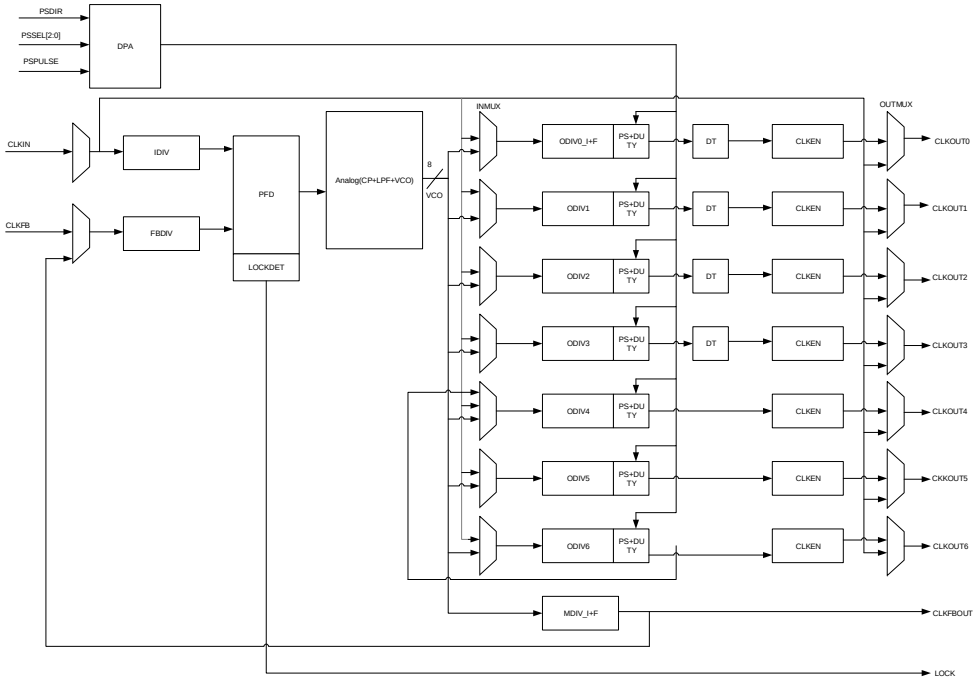
表 5-1 PLL 対応デバイス

ファミリー	シリーズ	デバイス
Arora	GW5AT	GW5AT-138、BバージョンのGW5AT-138
	GW5AST	BバージョンのGW5AST-138

機能の説明

PLL のアーキテクチャは、図 5-1 に示す通りです。

図 5-1 PLL の説明図



PLL は、与えられたリファレンスクロックをもとに、周波数、位相、およびデューティサイクルを調整し、異なる周波数、位相、およびデューティサイクルの出力クロックを生成することができます。CLKOUT0 と CLKFBOUT は 1/8 フラクショナル周波数調整をサポートし、CLKOUT0 ～CLKOUT3 は動的および静的なデューティサイクル微調整をサポートします。さらに、PLL は、CLKOUT6 から CLKOUT4 の内部カスケード、SSC 機能、および CLKIN と CLKOUT の揃いのためのクロック・デスキューをサポートしています。

正しいクロック出力を得るには、FPGA 製品データシートに記載されている周波数範囲に従って入力クロック周波数を設定する必要があります。

PLL は入力クロック(CLKIN)に対して周波数調整(逓倍及び分周)を行うことができます。その計算式は以下の通りです：

1. Fpfd=Fclkin/IDIV
2. Fclkfb=Fpfd*FBDIV
3. フィードバック方式により、VCO 周波数の計算式は異なります。

- 内部フィードバック

Fvco=Fclkfb*MDIV

- 外部からのフィードバック

Fvco=Fclkfb*MDIV ---- CLKFBOUT は CLKFB にフィードバックされます。

Fvco=Fclkfb*ODIVx ---- CLKOUTx は CLKFB にフィードバックされ

ます。

4. Fclkfbout= Fvco/MDIV

5. INMUX と OUTMUX のモードにより、CLKOUT チャンネルの出力周波数の計算式は異なります。

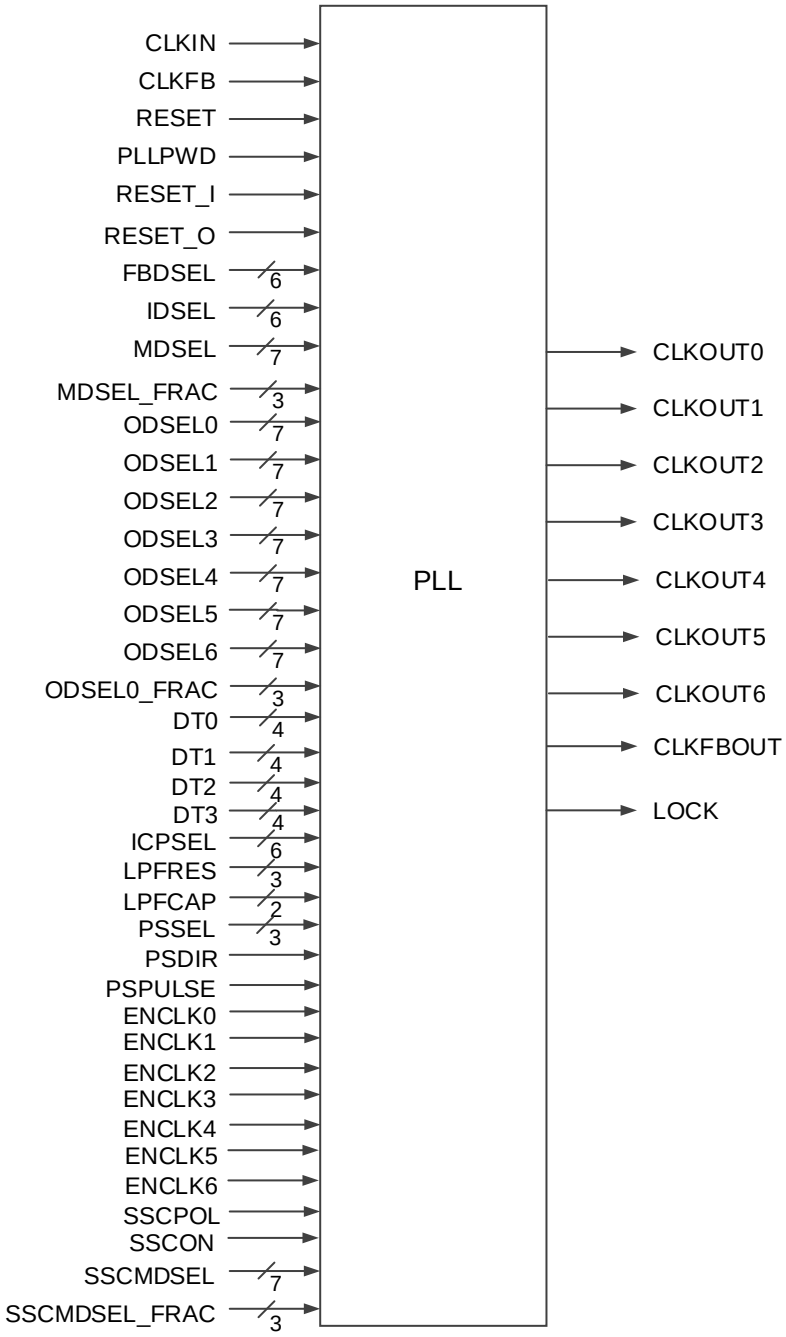
- VCO in モード（INMUX が VCO を選択する場合）：
  Fclkoutx=Fvco/ODIVx
- Bypass in モード（INMUX が CLKIN を選択する場合）：
  Fclkoutx=Fclkin/ODIVx
- Bypass out モード（OUTMUX が CLKIN を選択する場合）：
  Fclkoutx=Fclkin
- CAS モード（チャンネル 4 のみ）： Fclkout4= Fclkout6[1]/ODIV4

注記：

- Fclkin は、入力リファレンスクロック CLKIN の周波数です。
- Fclkoutx(x=0~6)は、0~6 チャネルの出力クロックの周波数です
- Fclkfb は、フィードバック入力クロック CLKFB の周波数です。
- Fpfd は位相検出器の周波数です。
- IDIV、FBDIV、MDIV、ODIVx (x=0~6)は、各分周器の分周係数です。分周係数を調整することにより、所望の周波数のクロック信号を生成することができます。
- [1] Fclkout6 は、チャネル 6 の出力周波数です。

ポート図

図 5-2 PLL のポート図



ポートの説明

表 5-2 PLL のポートの説明

ポート名	I/O	説明
CLKIN	入力	リファレンスクロック入力
CLKFB	入力	フィードバッククロック入力
RESET	入力	PLL 全部リセット信号。デジタル回路をリセットします。アクティブ High。
PLLPWD	入力	PLL パワーダウン信号。 アナログ回路のパワーダウンに使用されます。 アクティブ High。
RESET_I	入力	IDIV リセット機能付きの PLL 全部リセット信号。通常、内部テストに使用されます。アクティブ High。
RESET_O	入力	ODIV および一部のデジタル回路のリセット信号。通常、内部テストに使用されます。アクティブ High。
FBDSEL[5:0]	入力	FBDIV の値の動的制御。範囲は 0~63。FBDIV の実際の値は 64 - FBDSEL。
IDSEL[5:0]	入力	IDIV の値の動的制御。範囲は 0~63。IDIV の実際の値は 64 - IDSEL。
MDSEL[6:0]	入力	MDIV の整数値の動的制御。範囲は 0~126。MDIV の実際の値は 128-MDSEL(つまり 2~128)。
MDSEL_FRAC[2:0]	入力	MDIV の小数値の動的制御。MDIV の整数値が 2~127 の場合に適用されます。小数値は 1/8(0.125)単位でのインクリメントです。
ODSEL0[6:0]	入力	ODIV0 の整数値の動的制御。範囲は 0~127。ODIV0 の実際の値は 128-ODSEL0。
ODSEL0_FRAC[2:0]	入力	ODIV0 の小数値の動的制御。ODIV0 の整数値が 2~127 の場合に適用されます。小数値は 1/8(0.125)単位でのインクリメントです。
ODSEL1[6:0]	入力	ODIV1 の値の動的制御。範囲は 0~127。ODIV1 の実際の値は 128-ODSEL1。
ODSEL2[6:0]	入力	ODIV2 の値の動的制御。範囲は 0~127。ODIV2 の実際の値は 128-ODSEL2。
ODSEL3[6:0]	入力	ODIV3 の値の動的制御。範囲は 0~127。ODIV3 の実際の値は 128-ODSEL3。
ODSEL4[6:0]	入力	ODIV4 の値の動的制御。範囲は 0~127。ODIV4 の実際の値は 128-ODSEL4。
ODSEL5[6:0]	入力	ODIV5 の値の動的制御。範囲は 0~127。ODIV5 の実際の値は 128-ODSEL5。
ODSEL6[6:0]	入力	ODIV6 の値の動的制御。範囲は 0~127。ODIV6 の実際の値は 128-ODSEL6。
DT0[3:0]	入力	CLKOUT0 のデューティサイクルの動的微調整
DT1[3:0]	入力	CLKOUT1 のデューティサイクルの動的微調整
DT2[3:0]	入力	CLKOUT2 のデューティサイクルの動的微調整

ポート名	I/O	説明
DT3[3:0]	入力	CLKOUT3 のデューティサイクルの動的微調整
ICPSEL[5:0]	入力	ICP 電流の動的制御。この値の増加とともに電流が増加します。
LPFRES[2:0]	入力	LPFRES の値の動的制御。範囲は R0~R7。R0 は最大の帯域幅に対応し、R7 は最小の帯域幅に対応します。
LPFCAP[1:0]	入力	LPFCAP の値の動的制御。範囲は C0~C2。
PSDIR	入力	位相シフト方向の動的制御
PSSEL[2:0]	入力	位相シフトチャネル選択の動的制御
PSPULSE	入力	位相シフトパルスの動的制御
ENCLK0	入力	チャネル 0 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT0_EN = "TRUE"にする必要があります。
ENCLK1	入力	チャネル 1 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT1_EN = "TRUE"にする必要があります。
ENCLK2	入力	チャネル 2 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT2_EN = "TRUE"にする必要があります。
ENCLK3	入力	チャネル 3 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT3_EN = "TRUE"にする必要があります。
ENCLK4	入力	チャネル 4 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT4_EN = "TRUE"にする必要があります。
ENCLK5	入力	チャネル 5 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT5_EN = "TRUE"にする必要があります。
ENCLK6	入力	チャネル 6 のクロック出力イネーブルの動的制御。動的イネーブルを使用するには、静的パラメータ CLKOUT6_EN = "TRUE"にする必要があります。
SSCPOL	入力	タイミングの競合を回避するためのオプションの設定： ● 0：CLK Rising Edge ● 1：CLK Falling Edge
SSCON	入力	SSC イネーブルの動的制御
SSCMDSEL[6:0]	入力	SSC MDIV の整数値の動的制御。推奨される SSC MDIV の値の範囲は 16~24（50M の Fpfd に対応）、32~48（25M の Fpfd に対応）です。SSC MDIV の実際の値は 128-SSCMDSEL。
SSCMDSEL_FRAC[2:0]	入力	SSC MDIV の小数値の動的制御。小数値は 1/8(0.125)単位でのインクリメントです。
CLKOUT0	出力	チャネル 0 のクロック出力(デフォルト)

ポート名	I/O	説明
CLKOUT1	出力	チャネル 1 のクロック出力
CLKOUT2	出力	チャネル 2 のクロック出力
CLKOUT3	出力	チャネル 3 のクロック出力
CLKOUT4	出力	チャネル 4 のクロック出力
CLKOUT5	出力	チャネル 5 のクロック出力
CLKOUT6	出力	チャネル 6 のクロック出力
CLKFBOUT	出力	フィードバッククロック出力
LOCK	出力	PLL のロック状態を示します。 ● 1：ロック ● 0：ロック解除

パラメータの説明

表 5-3 PLL のパラメータの説明

パラメータ名	タイプ	値の範囲	デフォルト値	説明
FCLKIN	string	"10"~"400"	"100.0"	リファレンスクロックの周波数(MHz)
IDIV_SEL	integer	1~64	1	IDIV 分周係数の静的設定。実際の 1~64 に対応します。
DYN_IDIV_SEL	string	"TRUE", "FALSE"	"FALSE"	IDIV 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ IDIV_SEL を選択します ● TRUE: 動的制御、つまり信号 IDSEL を選択します
FBDIV_SEL	integer	1~64	1	FBDIV 分周係数の静的設定。実際の 1~64 に対応します。
DYN_FBDIV_SEL	string	"TRUE", "FALSE"	"FALSE"	FBDIV 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ FBDIV_SEL を選択します ● TRUE: 動的制御、つまり信号 FBDSEL を選択します
ODIV0_SEL	integer	1~128	8	ODIV0 分周係数の整数値の静的設定
ODIV0_FRAC_SEL	integer	0~7	0	ODIV0 分周係数の小数値の静的設定
DYN_ODIV0_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV0 分周係数の静的制御パラメータまたは動的制御信号の選択 FALSE: 静的制御、つまりパラメータ ODIV0SEL と ODIV0FRAC_SEL を選択します TRUE: 動的制御、つまり信号 ODSEL0 と ODSEL0_FRAC を選択します。

パラメータ名	タイプ	値の範囲	デフォルト値	説明
ODIV1_SEL	integer	1~128	8	ODIV1 分周係数の静的設定
DYN_ODIV1_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV1 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ODIV1_SEL を選択します。 ● TRUE: 動的制御、つまり信号 ODSEL1 を選択します
ODIV2_SEL	integer	1~128	8	ODIV2 分周係数の静的設定
DYN_ODIV2_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV2 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ODIV2_SEL を選択します ● TRUE: 動的制御、つまり信号 ODSEL2 を選択します
ODIV3_SEL	integer	1~128	8	ODIV3 分周係数の静的設定
DYN_ODIV3_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV3 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ODIV3_SEL を選択します ● TRUE: 動的制御、つまり信号 ODSEL3 を選択します
ODIV4_SEL	integer	1~128	8	ODIV4 分周係数の静的設定
DYN_ODIV4_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV4 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ODIV4_SEL を選択します ● TRUE: 動的制御、つまり信号 ODSEL4 を選択します
ODIV5_SEL	integer	1~128	8	ODIV5 分周係数の静的設定
DYN_ODIV5_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV5 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ODIV5_SEL を選択します ● TRUE: 動的制御、つまり信号 ODSEL5 を選択します
ODIV6_SEL	integer	1~128	8	ODIV6 分周係数の静的設定
DYN_ODIV6_SEL	string	"TRUE", "FALSE"	"FALSE"	ODIV6 分周係数の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ODIV6_SEL を選択します ● TRUE: 動的制御、つまり信号 ODSEL6 を選択します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
DYN_MDIV_SEL	string	"TRUE", "FALSE"	“FALSE"	MDIV 分周係数の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ MDIV_SEL と MDSEL_FRAC を選択します ● "TRUE"：動的制御、つまり信号 ODSEL6 を選択します
MDIV_SEL	integer	2~128	8	MDIV 分周係数の整数値の静的設定
MDIV_FRAC_SEL	string	0~7	0	MDIV 分周係数の小数値の静的設定
CLKOUT0_EN	string	"TRUE", "FALSE"	"TRUE"	チャネル 0 のクロック出力イネーブル
CLKOUT1_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 1 のクロック出力イネーブル
CLKOUT2_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 2 のクロック出力イネーブル
CLKOUT3_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 3 のクロック出力イネーブル
CLKOUT4_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 4 のクロック出力イネーブル
CLKOUT5_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 5 のクロック出力イネーブル
CLKOUT6_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 6 のクロック出力イネーブル
DYN_DT0_SEL	string	"TRUE", "FALSE"	"FALSE"	チャネル 0 デューティサイクル微調整の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ CLKOUT0_DT_DIR と CLKOUT0_DT_STEP を選択します ● TRUE: 動的制御、つまり信号 DT0 を選択します
DYN_DT1_SEL	string	"TRUE", "FALSE"	"FALSE"	チャネル 1 デューティサイクル微調整の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ CLKOUT1_DT_DIR と CLKOUT1_DT_STEP を選択します ● TRUE: 動的制御、つまり信号 DT1 を選択します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
DYN_DT2_SEL	string	"TRUE", "FALSE"	"FALSE"	チャネル 2 デューティサイクル微調整の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ CLKOUT2_DT_DIR と CLKOUT2_DT_STEP を選択します ● TRUE: 動的制御、つまり信号 DT2 を選択します
DYN_DT3_SEL	string	"TRUE", "FALSE"	"FALSE"	チャネル 3 デューティサイクル微調整の静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ CLKOUT3_DT_DIR と CLKOUT3_DT_STEP を選択します ● TRUE: 動的制御、つまり信号 DT3 を選択します
CLKOUT0_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 0 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT1_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 1 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT2_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 2 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT3_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 3 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
CLKOUT0_DT_STEP	integer	0,1,2,4	0	チャネル 0 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLKOUT1_DT_STEP	integer	0,1,2,4	0	チャネル 1 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLKOUT2_DT_STEP	integer	0,1,2,4	0	チャネル 2 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLKOUT3_DT_STEP	integer	0,1,2,4	0	チャネル 3 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLK0_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV0 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK0_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 0 出力クロックソースの選択 1'b0: ODIV0 の出力 1'b1: CLKIN(バイパス)
CLK1_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV1 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK1_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 1 出力クロックソースの選択 ● 1’b0：ODIV1 の出力 ● 1’b1：CLKIN(バイパス)
CLK2_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV2 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 2'b1: CLKIN(バイパス)
CLK2_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 2 出力クロックソースの選択 1'b0: ODIV2 の出力 1'b1: CLKIN(バイパス)
CLK3_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV3 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK3_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 3 出力クロックソースの選択 ● 1'b0: ODIV3 の出力 ● 1'b1: CLKIN(バイパス)
CLK4_IN_SEL	binary	2'b00, 2'b01,2’b10	2'b00	ODIV4 入力クロックソースの選択 ● 2'b00: VCO の出力

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● 2’b01: CLKCAS_6(カスケード) ● 2'b10: CLKIN(バイパス)
CLK4_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 4 出力クロックソースの選択 ● 1'b0: ODIV4 の出力 ● 1'b1: CLKIN(バイパス)
CLK5_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV5 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK5_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 5 出力クロックソースの選択 ● 1'b0: ODIV5 の出力 ● 1'b1: CLKIN(バイパス)
CLKFB_SEL	string	"INTERNAL", "EXTERNAL"	"INTERNAL"	CLKFB ソースの選択 ● INTERNAL：内部 CLKOUT からのフィードバック ● EXTERNAL：外部信号からのフィードバック
DYN_DPA_EN	string	"TRUE", "FALSE"	"FALSE"	動的位相シフト調整イネーブル
CLKOUT0_PE_COARSE	integer	0~127	0	チャネル 0 位相シフト粗調整の静的設定
CLKOUT0_PE_FINE	integer	0~7	0	チャネル 0 位相シフト微調整の静的設定
CLKOUT1_PE_COARSE	integer	0~127	0	チャネル 1 位相シフト粗調整の静的設定
CLKOUT1_PE_FINE	integer	0~7	0	チャネル 1 位相シフト微調整の静的設定
CLKOUT2_PE_COARSE	integer	0~127	0	チャネル 2 位相シフト粗調整の静的設定
CLKOUT2_PE_FINE	integer	0~7	0	チャネル 2 位相シフト微調整の静的設定
CLKOUT3_PE_COARSE	integer	0~127	0	チャネル 3 位相シフト粗調整の静的設定
CLKOUT3_PE_FINE	integer	0~7	0	チャネル 3 位相シフト微調整の静的設定
CLKOUT4_PE_COARSE	integer	0~127	0	チャネル 4 位相シフト粗調整の静的設定
CLKOUT4_PE_FINE	integer	0~7	0	チャネル 4 位相シフト微調整の静的設定
CLKOUT5_PE_COARSE	integer	0~127	0	チャネル 5 位相シフト粗調整の静的設定

パラメータ名	タイプ	値の範囲	デフォルト値	説明
CLKOUT5_PE_FINE	integer	0~7	0	チャネル 5 位相シフト微調整の静的設定
CLKOUT6_PE_COARSE	integer	0~127	0	チャネル 6 位相シフト粗調整の静的設定
CLKOUT6_PE_FINE	integer	0~7	0	チャネル 6 位相シフト微調整の静的設定
DYN_PE0_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 0 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT0_PE_COARSE と CLKOUT0_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE1_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 1 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT1_PE_COARSE と CLKOUT1_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE2_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 2 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT2_PE_COARSE と CLKOUT2_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE3_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 3 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT3_PE_COARSE と CLKOUT3_PE_FINE を選択します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE4_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 4 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT4_PE_COARSE と CLKOUT4_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE5_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 5 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT5_PE_COARSE と CLKOUT5_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE6_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 6 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT6_PE_COARSE と CLKOUT6_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DE0_EN	string	“TRUE","FALSE"	“FALSE"	チャネル 0(ODIV0 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：DYN_PE0_SEL="TRUE"の場合、CLKOUT0_PE_COARSEとCLKOUT0_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE1_EN	string	“TRUE","FALSE"	“FALSE"	チャネル1(ODIV1 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE1_SEL="TRUE"の場合、CLKOUT1_PE_COARSEとCLKOUT1_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE2_EN	string	“TRUE","FALSE"	“FALSE"	チャネル2(ODIV2 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE2_SEL="TRUE"の場合、CLKOUT2_PE_COARSEとCLKOUT2_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE3_EN	string	“TRUE","FALSE"	“FALSE"	チャネル3(ODIV3 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：DYN_PE3_SEL="TRUE"の場合、CLKOUT3_PE_COARSEとCLKOUT3_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE4_EN	string	“TRUE","FALSE"	“FALSE"	チャネル4(ODIV4 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE4_SEL="TRUE"の場合、CLKOUT4_PE_COARSEとCLKOUT4_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE5_EN	string	“TRUE","FALSE"	“FALSE"	チャネル5(ODIV5 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE5_SEL="TRUE"の場合、CLKOUT5_PE_COARSEとCLKOUT5_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE6_EN	string	“TRUE","FALSE"	“FALSE"	チャネル6(ODIV6 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：DYN_PE6_SEL="TRUE" の場合、CLKOUT6_PE_COARSE と CLKOUT6_PE_FINE を立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
RESET_I_EN	string	"TRUE", "FALSE"	"FALSE"	動的信号 RESET_I イネーブル。RESET_I ポートを使用するには、このパラメータを TRUE に設定する必要があります。
RESET_O_EN	string	"TRUE", "FALSE"	"FALSE"	動的信号 RESET_O イネーブル。RESET_O ポートを使用するには、このパラメータを TRUE に設定する必要があります。
DYN_ICP_SEL	string	"TRUE", "FALSE"	"FALSE"	ICPSEL 静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ICP_SEL を選択します ● TRUE: 動的制御、つまり動的信号 ICPSEL を選択します
ICP_SEL	binary	6'bXXXXXX, 6'b000000~6'b111111	6'bXXXXXX	ICP 電流の静的設定 ● 6'bXXXXXX：ソフトウェアが自動的に計算してこのパラメータを設定します ● 6'b000000~6'b111111：ユーザーは、必要に応じてパラメータ範囲内で設定できます
DYN_LPF_SEL	string	"TRUE", "FALSE"	"FALSE"	LPFRES と LPFCAP 静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ LPF_RES と LPF_CAP を選択します ● "TRUE"：動的制御、つまり動的信号 LPFREST と LPFCAP を選択します
LPF_RES	binary	3'bXXX,3'b000~3'b111	3'bXXX	LPRRES の静的設定 ● 3'bXXX：ソフトウェアが自動的に計算してこのパラメータを設定します ● 3'b000~3'b111：ユーザーは、必要に応じてパラメータ範囲内で設定できます

パラメータ名	タイプ	値の範囲	デフォルト値	説明
LPF_CAP	binary	2'b00~2'b10	2'b00	LFPCAP の静的設定
SSC_EN	string	"TRUE", "FALSE"	"FALSE"	SSC イネーブル

入力クロック分周器(IDIV)は、PLL モジュールへの入力クロック周波数を制御するために使用されます。この分周係数は、ポート IDSEL で動的に調整するか、またはパラメータ IDIV_SEL で静的に調整することができます。その対応関係を表 5-4 に示します。

IDIV 値とポート IDSEL の関係は、IDSEL=dec2bin(64-IDIV)です。

表 5-4 IDIV の対応関係

IDSEL[5:0]（動的）	IDIV_SEL(静的)	IDIV の実際値
111111	1	1
111110	2	2
111101	3	3
111100	4	4
111011	5	5
111010	6	6
111001	7	7
111000	8	8
110111	9	9
……	……	……
000000	64	64

FBDIV 分周器は、フィードバック信号の分周に使用されます。この分周係数は、ポート FBDSEL で動的に調整するか、またはパラメータ FBDIV_SEL で静的に調整することができます。その対応関係を表 5-5 に示します。

FBDIV 値とポート FBDSEL の関係は、FBDSEL=dec2bin(64-FBDIV)です

表 5-5 FBDIV の対応関係

FBDSEL [5:0]（動的）	FBDIV_SEL（静的）	FBDIV の実際値
111111	1	1
111110	2	2
111101	3	3
111100	4	4
111011	5	5
111010	6	6
111001	7	7

FBDSEL [5:0]（動的）	FBDIV_SEL（静的）	FBDIV の実際値
111000	8	8
110111	9	9
……	……	……
000000	64	64

出力分周器（ODIV）のチャンネル 0 は整数分周と小数分周をサポートし、チャンネル 1～6 は整数分周のみをサポートします。チャンネル 0 の場合、この分周係数はポート ODSEL0 と ODSEL0_FRAC で動的に調整するか、または パラメータ ODIV0_SEL と ODIV0_FRAC_SEL で静的に調整することができます。 整数の分周係数の対応関係を表 5-6 に示し、小数の分周係数の対応関係を表 5-7 に示します。

ODIV0 整数値とポート ODSEL の関係は、ODSEL0=dec2bin(128-ODIV0 整数値)です。

表 5-6 ODIV0 整数値の対応関係

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

ODIV0 小数値とポート ODSEL0_FRAC の関係は、ODSEL0_FRAC=dec2bin(7-ODIV0 小数値/0.125)です。また、整数分周値が 2~127 の場合のみ小数分周が有効です。チャンネル 1~6 の場合、ODIV 分周係数を ODSELx(x=1~6)で動的に調整するか、またはパラメータ ODIVx_SEL(x=1~6)で静的に調整することができます。その対応関係についてはチャンネル 0 の整数分周、すなわち表 5-6 を参照することができます。

表 5-7 ODIV0 小数値の対応関係

ODSEL_FRAC[2:0]（動的）	ODIV0_FRAC_SEL（静的）	ODIV0 小数値
111	0	0*0.125=0
110	1	1*0.125=0.125

ODSEL_FRAC[2:0]（動的）	ODIV0_FRAC_SEL（静的）	ODIV0 小数値
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 分周器の機能は FBDIV と同様です。整数と小数の分周係数がサポートされます。この分周係数はポート MDSEL と MDSEL_FRAC で動的に調整するか、またはパラメータ MDIV_SEL と MDIV_FRAC_SEL で静的に調整することができます。整数の分周係数の対応関係を表 5-8 に示し、小数の分周係数の対応関係を表 5-9 に示します。

表 5-8 MDIV 整数値の対応関係

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

MDIV 整数値とポート MDSEL の関係は MDSEL=dec2bin(128-MDIV 整数値)です。また、MDIV 値の範囲は 2~128 です。

表 5-9 MDIV 小数値の対応関係

MDSEL_FRAC[2:0]（動的）	MDIV_FRAC_SEL(静的)	MDIV 小数値
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 小数値とポート MDSEL_FRAC の関係は、MDSEL_FRAC=dec2bin(7-MDIV 小数値/0.125)です。また、整数分周値が 2~127 の場合のみ小数分周が有効です。

位相の調整

PLL は位相の静的調整と動的調整をサポートします。また、チャンネル 0~6 はすべて位相調整をサポートします。位相調整は、MDIV チャンネルを基準とした、MDIV に対する ODIV0~6 の位相シフトです。

静的位相調整は、パラメータ CLKOUTx_PE_COARSE と CLKOUTx_PE_FINE (x=0~6)の設定により実現されます。

動的位相調整は、信号 PSSEL、PSDIR、および PSPULSE によって実現されます。PSSEL はチャネルの選択に使用され、PSDIR は加算または減算の制御に使用されます。PSPULSE パルスの立ち下がりエッジごとに、DYN_FINE は 1 増加/減少し、DYN_FINE のオーバーフローまたはアンダーフローの場合、DYN_COARSE は 1 増加/減少します。DYN_COARSE の値は ODIV より小さいです。

位相調整の計算式は次のとおりです。

PS=(COARSE+FINE/8)/ODIV*360，PS の範囲は[0,360)です。

チャネル 0 の場合、ODIV=ODIV0 整数値+ODIV0 小数値です。チャネル 1~6 の場合、ODIV は整数値のみです。

注記：

- DYN_FINE および DYN_COARSE は、DPA によって生成される内部信号であり、PSSEL、PSDIR、および PSPULSE の協働により生成されます。その機能は CLKOUTx_PE_COARSE および CLKOUTx_PE_FINE と同じです。
- 式中の COARSE と FINE は、動的または静的調整を選択した場合の、位相調整に実際に有効な値を指します。
- 位相調整回路は、VCO 向けに設計されています。Bypass in または CAS モードの場合、位相調整式は適用できますが，FINE を 0 に設定する必要があります。

デューティサイクルの調整

PLL は、デューティサイクルの動的調整のみをサポートします。また、チャンネル 0~6 はすべてデューティサイクルの調整をサポートします。デューティサイクルは次のように定義されます。

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

falling edge の位置は、静的位相シフト設定によって決定され、DUTY として定義されます。rising edge の位置は、動的位相シフト設定の PHASE によって決定されます。DYN_FINE および DYN_COARSE は、DPA によって生成される内部信号です。位相調整セクションの関連説明を参照してください。DUTY と PHASE の計算式は次のとおりです(チャネル 1 の場合)。

DUTY=(CLKOUT1_PE_COARSE+CLKOUT1_PE_FINE/8)

PHASE=(DYN_COARSE1+DYN_FINE1/8)

動的デューティサイクルの計算：

- DUTY>PHASE の場合、Duty cycle=(DUTY-PHASE)/ODIV1
- DUTY<PHASE の場合、Duty cycle=(DUTY-PHASE)/ODIV1+1

注記：

- ODIV = 1 の場合、デューティサイクルの動的調整はサポートされておらず、デューティサイクルは常に 50%です。
- ODIV> = 2 の場合、DUTY-PHASE は(-0.5, 0.5)内の値をサポートしません。
- デューティサイクル調整回路は、VCO 向けに設計されています。Bypass in または CAS モードの場合、デューティサイクルの調整はサポートされていません。さらに、Bypass in または CAS モードでは、ODIV(>2)が奇数の場合、デューティサイクルは 50%ではありません（High レベル＜Low レベル、すなわちデューティサイクルは 50%未満となります）。

デューティサイクルの微調整

PLL は、デューティサイクルの静的微調整と動的微調整をサポートします(チャネル 0~3 のみ)。デューティサイクルの微調整は、デューティサイクルの微調整方向とステップを設定することでサポートされます。微調整方向が 1'b1 の場合、立ち下がりエッジの遅延が調整されてデューティサイクルが増加します。微調整方向が 1'b0 の場合、立ち上がりエッジの遅延が調整されてデューティサイクルが減少します。遅延値の詳細は、次に示すとおりです(チャネル 1 の場合)。

表 5-10 PLL のデューティサイクルの微調整の参照テーブル

動的調整	静的調整		デューティサイクルの微調整の遅延値
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

例えば、同じ周波数のクロックを出力するチャンネル 1 とチャンネル 2 がある場合、チャンネル 1 のクロックのデューティサイクルの微調整タイミングは、図 5-3 と図 5-4 に示すとおりです(チャネル 0 は基準として使用される)。

図 5-3 チャネル 1 のデューティサイクルの微調整タイミング図(微調整方向が 1'

b1、ステップ数が 1)

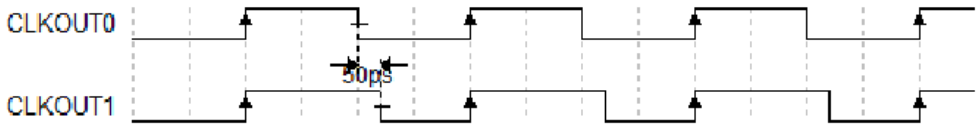
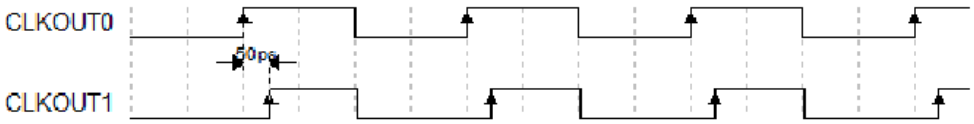


図 5-4 チャネル 1 のデューティサイクルの微調整タイミング図(微調整方向が 1’b0、ステップ数が 1)



プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

```
        .ODSEL4(odesl4),
        .ODSEL5(odesl5),
        .ODSEL6(odesl6),
        .DT0(dt0),
        .DT1(dt1),
        .DT2(dt2),
        .DT3(dt3),
        .ICPSEL(icpsel),
        .LPFRES(lpfres),
        .LPFCAP(lpfcap),
        .PSSEL(pssel),
        .PSDIR(psdir),
        .PSPULSE(pspulse),
        .ENCLK0(enclk0),
        .ENCLK1(enclk1),
        .ENCLK2(enclk2),
        .ENCLK3(enclk3),
        .ENCLK4(enclk4),
        .ENCLK5(enclk5),
        .ENCLK6(enclk6),
        .SSCPOL(sscpol),
        .SSCON(sscon),
        .SSCMDSEL(sscmdsel),
        .SSCMDSEL_FRAC(sscmdsel_frac)
    );
    defparam uut.CLK0_IN_SEL=1'b0;
    defparam uut.CLK0_OUT_SEL=1'b0;
    defparam uut.CLK1_IN_SEL=1'b0;
    defparam uut.CLK1_OUT_SEL=1'b0;
    defparam uut.CLK2_IN_SEL=1'b0;
    defparam uut.CLK2_OUT_SEL=1'b0;
    defparam uut.CLK3_IN_SEL=1'b0;
    defparam uut.CLK3_OUT_SEL=1'b0;
    defparam uut.CLK4_IN_SEL=2'b00;
    defparam uut.CLK4_OUT_SEL=1'b0;
    defparam uut.CLK5_IN_SEL=1'b0;
    defparam uut.CLK5_OUT_SEL=1'b0;
    defparam uut.CLK6_IN_SEL=1'b0;
    defparam uut.CLK6_OUT_SEL=1'b0;
    defparam uut.CLKFB_SEL="INTERNAL";
    defparam uut.CLKOUT0_DT_DIR=1'b1;
    defparam uut.CLKOUT0_DT_STEP=0;
```

```
defparam uut.CLKOUT0_EN="TRUE";
defparam uut.CLKOUT0_PE_COARSE=0;
defparam uut.CLKOUT0_PE_FINE=0;
defparam uut.CLKOUT1_DT_DIR=1'b1;
defparam uut.CLKOUT1_DT_STEP=0;
defparam uut.CLKOUT1_EN=" TRUE ";
defparam uut.CLKOUT1_PE_COARSE=0;
defparam uut.CLKOUT1_PE_FINE=0;
defparam uut.CLKOUT2_DT_DIR=1'b1;
defparam uut.CLKOUT2_DT_STEP=0;
defparam uut.CLKOUT2_EN="FALSE";
defparam uut.CLKOUT2_PE_COARSE=0;
defparam uut.CLKOUT2_PE_FINE=0;
defparam uut.CLKOUT3_DT_DIR=1'b1;
defparam uut.CLKOUT3_DT_STEP=0;
defparam uut.CLKOUT3_EN=" TRUE ";
defparam uut.CLKOUT3_PE_COARSE=0;
defparam uut.CLKOUT3_PE_FINE=0;
defparam uut.CLKOUT4_EN=" TRUE ";
defparam uut.CLKOUT4_PE_COARSE=0;
defparam uut.CLKOUT4_PE_FINE=0;
defparam uut.CLKOUT5_EN=" TRUE ";
defparam uut.CLKOUT5_PE_COARSE=0;
defparam uut.CLKOUT5_PE_FINE=0;
defparam uut.CLKOUT6_EN=" TRUE ";
defparam uut.CLKOUT6_PE_COARSE=0;
defparam uut.CLKOUT6_PE_FINE=0;
defparam uut.DE0_EN="FALSE";
defparam uut.DE1_EN="FALSE";
defparam uut.DE2_EN="FALSE";
defparam uut.DE3_EN="FALSE";
defparam uut.DE4_EN="FALSE";
defparam uut.DE5_EN="FALSE";
defparam uut.DE6_EN="FALSE";
defparam uut.DYN_DPA_EN="FALSE";
defparam uut.DYN_DT0_SEL="FALSE";
defparam uut.DYN_DT1_SEL="FALSE";
defparam uut.DYN_DT2_SEL="FALSE";
defparam uut.DYN_DT3_SEL="FALSE";
defparam uut.DYN_FBDIV_SEL="FALSE";
defparam uut.DYN_ICP_SEL="FALSE";
defparam uut.DYN_IDIV_SEL="FALSE";
```

```
defparam uut.DYN_LPF_SEL="FALSE";
defparam uut.DYN_MDIV_SEL="FALSE";
defparam uut.DYN_ODIV0_SEL="FALSE";
defparam uut.DYN_ODIV1_SEL="FALSE";
defparam uut.DYN_ODIV2_SEL="FALSE";
defparam uut.DYN_ODIV3_SEL="FALSE";
defparam uut.DYN_ODIV4_SEL="FALSE";
defparam uut.DYN_ODIV5_SEL="FALSE";
defparam uut.DYN_ODIV6_SEL="FALSE";
defparam uut.DYN_PE0_SEL="FALSE";
defparam uut.DYN_PE1_SEL="FALSE";
defparam uut.DYN_PE2_SEL="FALSE";
defparam uut.DYN_PE3_SEL="FALSE";
defparam uut.DYN_PE4_SEL="FALSE";
defparam uut.DYN_PE5_SEL="FALSE";
defparam uut.DYN_PE6_SEL="FALSE";
defparam uut.FBDIV_SEL=1;
defparam uut.FCLKIN="100.0";
defparam uut.ICP_SEL=6'bXXXXXX;
defparam uut.IDIV_SEL=1;
defparam uut.LPF_CAP=2'b00;
defparam uut.LPF_RES=3'b000;
defparam uut.MDIV_FRAC_SEL=0;
defparam uut.MDIV_SEL=8;
defparam uut.ODIV0_FRAC_SEL=0;
defparam uut.ODIV0_SEL=8;
defparam uut.ODIV1_SEL=8;
defparam uut.ODIV2_SEL=8;
defparam uut.ODIV3_SEL=8;
defparam uut.ODIV4_SEL=8;
defparam uut.ODIV5_SEL=8;
defparam uut.ODIV6_SEL=8;
defparam uut.RESET_I_EN="FALSE";
defparam uut.RESET_O_EN="FALSE";
defparam uut.SSC_EN="FALSE";
```

VHDL でのインスタンス化：

```
COMPONENT PLL
    GENERIC(
        FCLKIN : STRING := "100.0";
        DYN_IDIV_SEL : STRING := "FALSE";
        IDIV_SEL : integer := 1;
```

```
DYN_FBDIV_SEL : STRING := "FALSE";
FBDIV_SEL : integer := 1;
DYN_ODIV0_SEL : STRING := "FALSE";
ODIV00_SEL : integer := 8;
DYN_ODIV1_SEL : STRING := "FALSE";
ODIV1_SEL : integer := 8;
DYN_ODIV2_SEL : STRING := "FALSE";
ODIV2_SEL : integer := 8;
DYN_ODIV3_SEL : STRING := "FALSE";
ODIV3_SEL : integer := 8;
DYN_ODIV4_SEL : STRING := "FALSE";
ODIV4_SEL : integer := 8;
DYN_ODIV5_SEL : STRING := "FALSE";
ODIV5_SEL : integer := 8;
DYN_ODIV6_SEL : STRING := "FALSE";
ODIV6_SEL : integer := 8;
DYN_MDIV_SEL : STRING := "FALSE";
MDIV_SEL : integer := 8;
MDIV_FRAC_SEL : integer := 0;
ODIV0_FRAC_SEL : integer := 0;
CLKOUT0_EN : STRING := "TRUE";
CLKOUT1_EN : STRING := " FALSE ";
CLKOUT2_EN : STRING := " FALSE ";
CLKOUT3_EN : STRING := " FALSE ";
CLKOUT4_EN : STRING := " FALSE ";
CLKOUT5_EN : STRING := " FALSE ";
CLKOUT6_EN : STRING := " FALSE ";
DYN_DT0_SEL : STRING := "FALSE";
DYN_DT1_SEL : STRING := "FALSE";
DYN_DT2_SEL : STRING := "FALSE";
DYN_DT3_SEL : STRING := "FALSE";
CLKOUT0_DT_DIR : bit := '1';
CLKOUT1_DT_DIR : bit := '1';
CLKOUT2_DT_DIR : bit := '1';
CLKOUT3_DT_DIR : bit := '1';
```

```
CLKOUT0_DT_STEP : integer := 0;
CLKOUT1_DT_STEP : integer := 0;
CLKOUT2_DT_STEP : integer := 0;
CLKOUT3_DT_STEP : integer := 0;
CLK0_IN_SEL   : bit := '0';
CLK0_OUT_SEL : bit := '0';
CLK1_IN_SEL   : bit_vector := '0';
CLK1_OUT_SEL : bit := '0';
CLK2_IN_SEL   : bit_vector := '0';
CLK2_OUT_SEL : bit := '0';
CLK3_IN_SEL   : bit_vector := '0';
CLK3_OUT_SEL : bit := '0';
CLK4_IN_SEL   : bit_vector := "00";
CLK4_OUT_SEL : bit := '0';
CLK5_IN_SEL   : bit_vector := '0';
CLK5_OUT_SEL : bit := '0';
CLK6_IN_SEL   : bit_vector := '0';
CLK6_OUT_SEL : bit := '0';
CLKFB_SEL : STRING := "INTERNAL";
DYN_DPA_EN : STRING := "FALSE";
DYN_PE0_SEL : STRING := "FALSE";
DYN_PE1_SEL : STRING := "FALSE";
DYN_PE2_SEL : STRING := "FALSE";
DYN_PE3_SEL : STRING := "FALSE";
DYN_PE4_SEL : STRING := "FALSE";
DYN_PE5_SEL : STRING := "FALSE";
DYN_PE6_SEL : STRING := "FALSE";
CLKOUT0_PE_COARSE : integer := 0;
CLKOUT0_PE_FINE : integer := 0;
CLKOUT1_PE_COARSE : integer := 0;
CLKOUT1_PE_FINE : integer := 0;
CLKOUT2_PE_COARSE : integer := 0;
CLKOUT2_PE_FINE : integer := 0;
CLKOUT3_PE_COARSE : integer := 0;
CLKOUT3_PE_FINE : integer := 0;
```

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

        DT0,DT1,DT2,DT3 : IN std_logic_vector(3 downto 0);
        ICPSEL : IN std_logic_vector(5 downto 0);
```

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

```
DYN_ODIV6_SEL => "FALSE",
ODIV6_SEL => 8,
DYN_MDIV_SEL => "FALSE",
MDIV_SEL => 8,
MDIV_FRAC_SEL => 0,
ODIV0_FRAC_SEL => 0,
CLKOUT0_EN => "TRUE",
CLKOUT1_EN => " FALSE ",
CLKOUT2_EN => " FALSE ",
CLKOUT3_EN => " FALSE ",
CLKOUT4_EN => " FALSE ",
CLKOUT5_EN => " FALSE ",
CLKOUT6_EN => " FALSE ",
DYN_DT0_SEL => "FALSE",
DYN_DT1_SEL => "FALSE",
DYN_DT2_SEL => "FALSE",
DYN_DT3_SEL => "FALSE",
CLKOUT0_DT_DIR => '1',
CLKOUT1_DT_DIR => '1',
CLKOUT2_DT_DIR => '1',
CLKOUT3_DT_DIR => '1',
CLKOUT0_DT_STEP => 0,
CLKOUT1_DT_STEP => 0,
CLKOUT2_DT_STEP => 0,
CLKOUT3_DT_STEP => 0,
CLK0_IN_SEL => '0',
CLK0_OUT_SEL => '0',
CLK1_IN_SEL => '0',
CLK1_OUT_SEL => '0',
CLK2_IN_SEL => '0',
CLK2_OUT_SEL => '0',
CLK3_IN_SEL => '0',
CLK3_OUT_SEL => '0',
CLK4_IN_SEL => "00",
CLK4_OUT_SEL => '0',
```

```
CLK5_IN_SEL => '0',
CLK5_OUT_SEL => '0',
CLK6_IN_SEL => '0',
CLK6_OUT_SEL => '0',
CLKFB_SEL=> "INTERNAL",
DYN_DPA_EN => "FALSE",
DYN_PE0_SEL => "FALSE",
DYN_PE1_SEL => "FALSE",
DYN_PE2_SEL => "FALSE",
DYN_PE3_SEL => "FALSE",
DYN_PE4_SEL => "FALSE",
DYN_PE5_SEL => "FALSE",
DYN_PE6_SEL => "FALSE",
CLKOUT0_PE_COARSE => 0,
CLKOUT0_PE_FINE => 0,
CLKOUT1_PE_COARSE => 0,
CLKOUT1_PE_FINE => 0,
CLKOUT2_PE_COARSE=> 0,
CLKOUT2_PE_FINE => 0,
CLKOUT3_PE_COARSE => 0,
CLKOUT3_PE_FINE => 0,
CLKOUT4_PE_COARSE => 0,
CLKOUT4_PE_FINE => 0,
CLKOUT5_PE_COARSE => 0,
CLKOUT5_PE_FINE => 0,
CLKOUT6_PE_COARSE => 0,
CLKOUT6_PE_FINE => 0,
DE0_EN => "FALSE",
DE1_EN => "FALSE",
DE2_EN => "FALSE",
DE3_EN => "FALSE",
DE4_EN => "FALSE",
DE5_EN => "FALSE",
DE6_EN => "FALSE",
RESET_I_EN => "FALSE",
```

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

```
        LPFRES=>lpfres,
        LPFCAP=>lpfcap,
        PSSEL=>pssel,
        PSDIR=>psdir,
        PSPULSE=>pspulse,
        ENCLK0=>enclk0,
        ENCLK1=>enclk1,
        ENCLK2=>enclk2,
        ENCLK3=>enclk3,
        ENCLK4=>enclk4,
        ENCLK5=>enclk5,
        ENCLK6=>enclk6,
        SSCPOL=>sscpol,
        SSCON=>sscon,
        SSCMDSEL=>sscmdsel,
        SSCMDSEL_FRAC=>sscmdsel_frac
    );
```

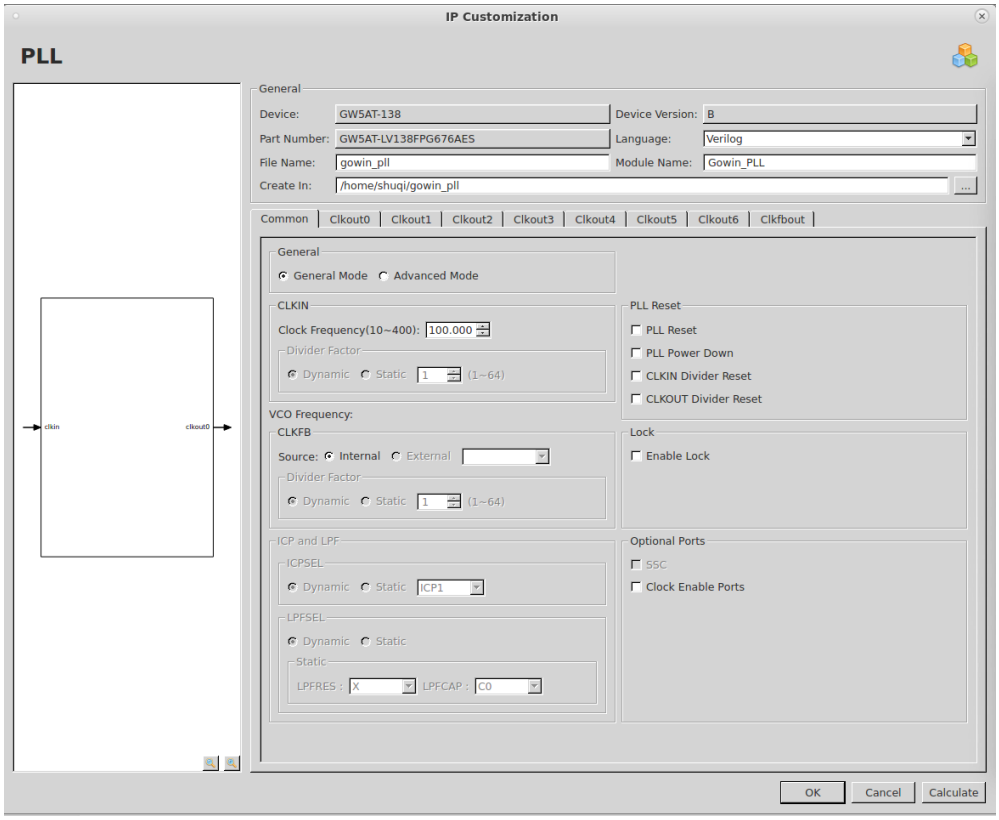
5.1.2 IP の呼び出し

IP Core Generator のインターフェースで“PLL”をクリックすると、右側に PLL の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“PLL”をダブルクリックすると、PLL の“IP Customization”ウィンドウがポップアップします。このウィンドウには General 構成タブ、Options 構成タブ、およびポート図があります(図 5-5)。

図 5-5 PLL IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。PLL の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. Options 構成タブ

   Options 構成タブは、IP のカスタマイズに使用されます(図 5-5)。

   - General：一般モードまたはアドバンスモードを選択します。一般モードでは、入力クロック周波数と出力クロック周波数を入力すると、ソフトウェアが自動的に周波数分割係数を計算します。アドバンスモードはアドバンスユーザーに適しており、アドバンスモードはでは、入力周波数と周波数分割係数を入力して期待される出力を得ることができます。
   - CLKIN：PLL 入力クロックの周波数、分周パラメータなどを構成します。
     - “Clock Frequency(周波数範囲)”：入力クロックの周波数を構成します。
     - “Divide Factor”：アドバンスモードで分周パラメータを構成するために使用され、動的モード“Dynamic”と静的モード“Static”をサポートします。静的モードでは分周パラメータ数値(範囲は 1~64)を構成できます。

- VCO Frequency：計算された VCO の周波数であり、読み出し専用です。

- CLKFB：PLL フィードバッククロックのソースと周波数逓倍パラメータを構成します。
  - フィードバッククロックのソース(Source)として Internal と External を選択できます。Internal を選択した場合は、内部からのフィードバックであり、External を選択した場合は、CLKOUT0～6 と CLKFBOUT のいずれかからのフィードバックです（選択可）。
  - “Divide Factor”：アドバンスモードで周波数逓倍パラメータを構成するために使用され、動的モード“Dynamic”と静的モード“Static”をサポートします。静的モードでは周波数逓倍パラメータ数値(範囲は 1~64)を構成できます。
- ICP and LPF：ICP 電流の動的制御。この値の増加とともに電流が増加します。
- LPFSEL：
  - LPFRES の値の動的制御。範囲は R0~R7。R0 は最大の帯域幅に対応し、R7 は最小の帯域幅に対応します。
  - LPFCAP の値の動的制御。範囲は C0~C2。
- PLL Reset：
  - “PLL Reset”：チェックすると、デジタル回路がリセットされます。
  - “PLL Power Down”：チェックすると、アナログ回路がパワーダウンします。
  - “CLKIN Divider Reset”：チェックすると、RESET_I が有効になります。
  - “CLKOUT Divider Reset”：RESET_O が有効になります。
- Lock：PLL のロック状態信号。
- Optional Ports
  - スペクトラム拡散クロック（SSC）は、Advanced Mode でのみサポートされます。
  - “SSC”を選択した場合、フィードバックのソースは内部のみになります。
- CLKOUT1 (CLKOUT1 を例に説明します)

General の構成：

- CLKOUT1～CLKOUT6 を有効にすることができます。“CLKOUT0”はデフォルトで有効になっており、ユーザーに

よる構成はできません。

- “CLKOUT1”を有効にしていない場合、このページのすべてのオプションは選択不可能で、デフォルトでは有効にされていません。
- “Bypass”が選択されて“Enable CLKOUT1 Divider”が選択された場合、モードは Bypass in となり、“Bypass”が選択されて“Enable CLKOUT1 Divider”が選択されていない場合、モードは Bypass out となります。
- “Bypass”オプションは、出力クロックのバイパス機能の構成に使用されます。“Enable CLKOUTA Divider”オプションは、VCO クロックのバイパス機能の構成に使用されます。
- “Expected Frequency(周波数範囲)”：一般モードで目的の出力クロック CLKOUTA の周波数を構成します。非 bypass モードでは、範囲は 6.25M~1000MHz です。
- “Tolerance(%)”：CLKOUTA の目的の周波数と算出された実際の周波数の許容誤差を構成します。
- “Actual Frequency”：算出された CLKOUTA の実際の出力周波数を表示します。

CLKOUT1 Divider Factor の構成

- CLKOUT1 の出力分周器は、1～128 範囲内の静的調整または動的調整をサポートします。構成が妥当でない場合、“Calculate”または“OK”をクリックすると、エラーが報告されます。

Phase の構成：

- PLL は位相の静的調整と動的調整をサポートします。また、チャンネル 0~6 はすべて位相調整をサポートします。

Duty Cycle の構成：

- PLL は、デューティサイクルの動的調整のみをサポートします。また、チャンネル 0~6 はすべてデューティサイクルの調整をサポートします。 デューティサイクル調整回路は VCO 用に設計されており、Bypass in およびカスケードモードではデューティサイクル調整を行うことができません。

Duty Trim の構成

-PLL は、デューティサイクルの静的微調整と動的微調整をサポートします(チャネル 0~3 のみ)。

- CLKFBOUT：

- CLKFBOUT のパラメータの構成につていは、上記の CLKOUT1 を参照してください。

- Calculate：

  - 構成された分周パラメータ、周波数逓倍パラメータ、VCO パラメータが妥当か計算します。妥当でない場合は、“Calculate”をクリックすると“error”メッセージが報告され、どこにエラーがあるかが示されます。構成が正しい場合は、“Calculate”をクリックすると、構成が成功したことを示す“succeed”ウィンドウが表示されます。

3. ポート図
   ポート図は、IP Core の構成結果を表示します(図 5-5)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_pll.v”は完全な verilog モジュールです。
- “gowin_pll_tmp.v”は IP のテンプレートファイルです。
- “gowin_pll.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

5.2 PLLA

5.2.1 プリミティブの紹介

Arora V FPGA は、7 つのクロック出力をサポートする位相同期回路 PLLA を提供します。各出力クロックは、リファレンスクロックに基づいて周波数、位相、およびデューティサイクルを独立して調整することをサポートします。

サポートされるデバイス

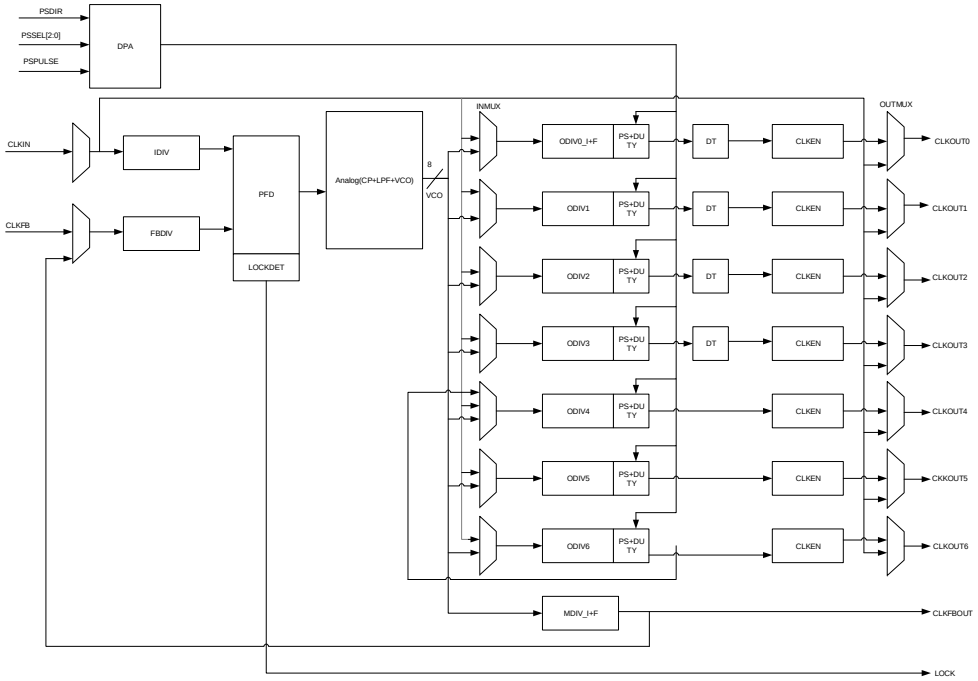
表 5-11 PLLA 対応デバイス

ファミリー	シリーズ	デバイス
Arora	GW5A	GW5A-25

機能の説明

PLLA のアーキテクチャは、下図に示す通りです。

図 5-6 PLLA の説明図



PLLA は、与えられたリファレンスクロックをもとに、周波数、位相、およびデューティサイクルを調整し、異なる周波数、位相、およびデューティサイクルの出力クロックを生成することができます。CLKOUT0 と CLKFBOUT は 1/8 フラクショナル周波数調整をサポートし、CLKOUT0～CLKOUT3 は動的および静的なデューティサイクル微調整をサポートします。さらに、PLLA は、CLKOUT6 から CLKOUT4 の内部カスケード、SSC 機能、および CLKIN と CLKOUT の揃いのためのクロック・デスキューをサポートしています。

正しいクロック出力を得るには、FPGA 製品データシートに記載されている周波数範囲に従って入力クロック周波数を設定する必要があります。

PLLA は入力クロック(CLKIN)に対して周波数調整(逓倍及び分周)を行うことができます。その計算式は以下の通りです：

1. Fpfd=Fclkin/IDIV
2. Fclkfb=Fpfd*FBDIV
3. フィードバック方式により、VCO 周波数の計算式は異なります。

- 内部フィードバック

Fvco=Fclkfb*MDIV

- 外部からのフィードバック

Fvco=Fclkfb*MDIV ---- CLKFBOUT は CLKFB にフィードバックされます。

Fvco=Fclkfb*ODIVx ---- CLKOUTx は CLKFB にフィードバックされ

ます。

4. Fclkfbout= Fvco/MDIV

5. INMUX と OUTMUX のモードにより、CLKOUT チャンネルの出力周波数の計算式は異なります。

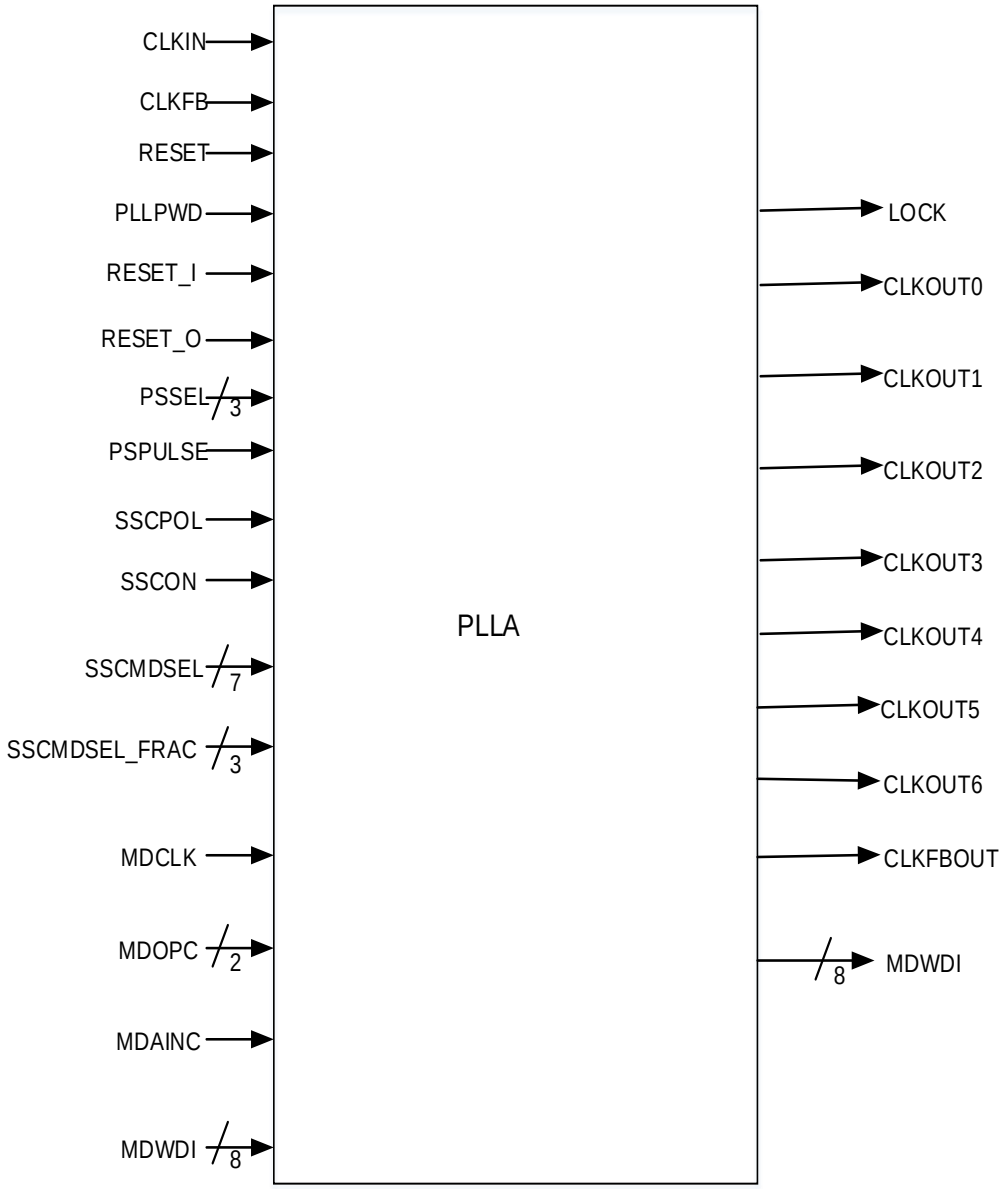
- VCO in モード（INMUX が VCO を選択する場合）：
  Fclkoutx=Fvco/ODIVx
- Bypass in モード（INMUX が CLKIN を選択する場合）：
  Fclkoutx=Fclkin/ODIVx
- Bypass out モード（OUTMUX が CLKIN を選択する場合）：
  Fclkoutx=Fclkin
- CAS モード（チャンネル 4 のみ）： Fclkout4= Fclkout6[1]/ODIV4

注記：

- Fclkin は、入力リファレンスクロック CLKIN の周波数です。
- Fclkoutx(x=0~6)は、0~6 チャネルの出力クロックの周波数です
- Fclkfb は、フィードバック入力クロック CLKFB の周波数です。
- Fpfd は位相検出器の周波数です。
- IDIV、FBDIV、MDIV、ODIVx (x=0~6)は、各分周器の分周係数です。分周係数を調整することにより、所望の周波数のクロック信号を生成することができます。
- [1] Fclkout6 は、チャネル 6 の出力周波数です。

ポート図

図 5-7 PLLA のポート図



ポートの説明

表 5-12 PLLA のポートの説明

ポート名	I/O	説明
CLKIN	入力	リファレンスクロック入力
CLKFB	入力	フィードバッククロック入力
RESET	入力	PLL 全部リセット信号。デジタル回路をリセットします。アクティブ High。
PLLPWD	入力	PLL パワーダウン信号。 アナログ回路のパワーダウンに使用されます。 アクティブ High。
RESET_I	入力	IDIV リセット機能付きの PLL 全部リセット信号。通常、内部テストに使用されます。アクティブ High。
RESET_O	入力	ODIV および一部のデジタル回路のリセット信号。通常、内部テストに使用されます。アクティブ High。
PSDIR	入力	位相シフト方向の動的制御
PSSEL[2:0]	入力	位相シフトチャネル選択の動的制御
PSPULSE	入力	位相シフトパルスの動的制御
SSCPOL	入力	タイミングの競合を回避するためのオプションの設定： ● 0：CLK Rising Edge ● 1：CLK Falling Edge
SSCON	入力	SSC イネーブルの動的制御
SSCMDSEL[6:0]	入力	SSC MDIV の整数値の動的制御。推奨される SSC MDIV の値の範囲は 16~24（50M の Fpfd に対応）、32~48（25M の Fpfd に対応）です。SSC MDIV の実際の値は 128-SSCMDSEL。
SSCMDSEL_FRAC[2:0]	入力	SSC MDIV の小数値の動的制御。小数値は 1/8(0.125) 単位でのインク リ メ ン トです。
CLKOUT0	出力	チャネル 0 のクロック出力(デフォルト)
CLKOUT1	出力	チャネル 1 のクロック出力
CLKOUT2	出力	チャネル 2 のクロック出力
CLKOUT3	出力	チャネル 3 のクロック出力
CLKOUT4	出力	チャネル 4 のクロック出力
CLKOUT5	出力	チャネル 5 のクロック出力
CLKOUT6	出力	チャネル 6 のクロック出力
CLKFBOUT	出力	フィードバッククロック出力
LOCK	出力	PLL のロック状態を示します。 ● 1：ロック ● 0：ロック解除
MDCLK	入力	クロック入力信号。DRP ポートのすべての信号の反転は、このクロックの立ち上がりエッジでトリガーされます。

ポート名	I/O	説明
MDOPC [1:0]	入力	操作タイプのコード： 00：読み出し/書き込み操作なし（NOOP） 10：読み出し操作（RD） 01：書き込み操作(WR) 11：MDAINC が High であるかどうかに関係なく、レジスタ アドレスは変更されません。
MDAINC	入力	アドレス増加の指示信号、アクティブ High。
MDWDI[7:0]	入力	mDRP ポートを介して書き込まれるデータ
MDRDO[7:0]	出力	mDRP ポートから読み出されるデータ

パラメータの説明

表 5-13 PLLA のパラメータの説明

パラメータ名	タイプ	値の範囲	デフォルト値	説明
FCLKIN	string	"10"~"400"	"100.0"	リファレンスクロックの周波数(MHz)
IDIV_SEL	integer	1~64	1	IDIV 分周係数の静的設定。実際の 1~64 に対応します。
FBDIV_SEL	integer	1~64	1	FBDIV 分周係数の静的設定。実際の 1~64 に対応します。
ODIV0_SEL	integer	1~128	8	ODIV0 分周係数の整数値の静的設定
ODIV0_FRAC_SEL	integer	0~7	0	ODIV0 分周係数の小数値の静的設定
ODIV1_SEL	integer	1~128	8	ODIV1 分周係数の静的設定
ODIV2_SEL	integer	1~128	8	ODIV2 分周係数の静的設定
ODIV3_SEL	integer	1~128	8	ODIV3 分周係数の静的設定
ODIV4_SEL	integer	1~128	8	ODIV4 分周係数の静的設定
ODIV5_SEL	integer	1~128	8	ODIV5 分周係数の静的設定
ODIV6_SEL	integer	1~128	8	ODIV6 分周係数の静的設定
MDIV_SEL	integer	2~128	8	MDIV 分周係数の整数値の静的設定
MDIV_FRAC_SEL	string	0~7	0	MDIV 分周係数の小数値の静的設定
CLKOUT0_EN	string	"TRUE", "FALSE"	"TRUE"	チャネル 0 のクロック出力イネーブル
CLKOUT1_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 1 のクロック出力イネーブル
CLKOUT2_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 2 のクロック出力イネーブル
CLKOUT3_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 3 のクロック出力イネーブル

パラメータ名	タイプ	値の範囲	デフォルト値	説明
CLKOUT4_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 4 のクロック出力イネーブル
CLKOUT5_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 5 のクロック出力イネーブル
CLKOUT6_EN	string	"TRUE", "FALSE"	"FALSE"	チャネル 6 のクロック出力イネーブル
CLKOUT0_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 0 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT1_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 1 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT2_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 2 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT3_DT_DIR	binary	1'b1, 1'b0	1'b1	チャネル 3 デューティサイクルの静的微調整方向 ● 1'b1: 立ち上がりエッジ揃えでデューティサイクルが増加します ● 1'b0: 立ち下がりエッジ揃えでデューティサイクルが減少します
CLKOUT0_DT_STEP	integer	0,1,2,4	0	チャネル 0 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLKOUT1_DT_STEP	integer	0,1,2,4	0	チャネル 1 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLKOUT2_DT_STEP	integer	0,1,2,4	0	チャネル 2 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLKOUT3_DT_STEP	integer	0,1,2,4	0	チャネル 3 デューティサイクル静的微調整のステップ数。ステップごとに 50ps。
CLK0_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV0 入力クロックソースの選択 ● 1'b0: VCO の出力

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● 1'b1: CLKIN(バイパス)
CLK0_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 0 出力クロックソースの選択 ● 1'b0: ODIV0 の出力 ● 1'b1: CLKIN(バイパス)
CLK1_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV1 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK1_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 1 出力クロックソースの選択 ● 1’b0：ODIV1 の出力 ● 1’b1：CLKIN(バイパス)
CLK2_IN_SEL	binary	1'b0,1'b1	1'b0	ODIV2 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 2'b1: CLKIN(バイパス)
CLK2_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 2 出力クロックソースの選択 ● 1'b0: ODIV2 の出力 ● 1'b1: CLKIN(バイパス)
CLK3_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV3 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK3_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 3 出力クロックソースの選択 ● 1'b0: ODIV3 の出力 ● 1'b1: CLKIN(バイパス)
CLK4_IN_SEL	binary	2'b00, 2'b01,2’b10	2'b00	ODIV4 入力クロックソースの選択 ● 2'b00: VCO の出力 ● 2’b01: CLKCAS_6(カスケード) ● 2'b10: CLKIN(バイパス)
CLK4_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 4 出力クロックソースの選択 ● 1'b0: ODIV4 の出力 ● 1'b1: CLKIN(バイパス)
CLK5_IN_SEL	binary	1'b0, 1'b1	1'b0	ODIV5 入力クロックソースの選択 ● 1'b0: VCO の出力 ● 1'b1: CLKIN(バイパス)
CLK5_OUT_SEL	binary	1'b0, 1'b1	1'b0	チャンネル 5 出力クロックソースの選択 ● 1'b0: ODIV5 の出力 ● 1'b1: CLKIN(バイパス)

パラメータ名	タイプ	値の範囲	デフォルト値	説明
CLKFB_SEL	string	"INTERNAL", "EXTERNAL"	"INTERNAL"	CLKFB ソースの選択 ● INTERNAL：内部 CLKOUT からのフィードバック ● EXTERNAL：外部信号からのフィードバック
DYN_DPA_EN	string	"TRUE", "FALSE"	"FALSE"	動的位相シフト調整イネーブル
CLKOUT0_PE_COARSE	integer	0~127	0	チャネル 0 位相シフト粗調整の静的設定
CLKOUT0_PE_FINE	integer	0~7	0	チャネル 0 位相シフト微調整の静的設定
CLKOUT1_PE_COARSE	integer	0~127	0	チャネル 1 位相シフト粗調整の静的設定
CLKOUT1_PE_FINE	integer	0~7	0	チャネル 1 位相シフト微調整の静的設定
CLKOUT2_PE_COARSE	integer	0~127	0	チャネル 2 位相シフト粗調整の静的設定
CLKOUT2_PE_FINE	integer	0~7	0	チャネル 2 位相シフト微調整の静的設定
CLKOUT3_PE_COARSE	integer	0~127	0	チャネル 3 位相シフト粗調整の静的設定
CLKOUT3_PE_FINE	integer	0~7	0	チャネル 3 位相シフト微調整の静的設定
CLKOUT4_PE_COARSE	integer	0~127	0	チャネル 4 位相シフト粗調整の静的設定
CLKOUT4_PE_FINE	integer	0~7	0	チャネル 4 位相シフト微調整の静的設定
CLKOUT5_PE_COARSE	integer	0~127	0	チャネル 5 位相シフト粗調整の静的設定
CLKOUT5_PE_FINE	integer	0~7	0	チャネル 5 位相シフト微調整の静的設定
CLKOUT6_PE_COARSE	integer	0~127	0	チャネル 6 位相シフト粗調整の静的設定
CLKOUT6_PE_FINE	integer	0~7	0	チャネル 6 位相シフト微調整の静的設定
DYN_PE0_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 0 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT0_PE_COARSE と CLKOUT0_PE_FINE を選択します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE1_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 1 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT1_PE_COARSE と CLKOUT1_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE2_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 2 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT2_PE_COARSE と CLKOUT2_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE3_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 3 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT3_PE_COARSE と CLKOUT3_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE4_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 4 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT4_PE_COARSE と CLKOUT4_PE_FINE を選択します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE5_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 5 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT5_PE_COARSE と CLKOUT5_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DYN_PE6_SEL	string	“TRUE","FALSE"	“FALSE"	チャネル 6 位相調整の静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ CLKOUT6_PE_COARSE と CLKOUT6_PE_FINE を選択します ● "TRUE"：動的制御、つまり DPA 動的信号(PSSEL、PSDIR、および PSPULSE)を選択します。また、DYN_DPA_EN="TRUE"にする必要があります。
DE0_EN	string	“TRUE","FALSE"	“FALSE"	チャネル 0(ODIV0 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE0_SEL="TRUE" の場合、CLKOUT0_PE_COARSE と CLKOUT0_PE_FINE を立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE1_EN	string	“TRUE","FALSE"	“FALSE"	チャネル 1(ODIV1 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：DYN_PE1_SEL="TRUE"の場合、CLKOUT1_PE_COARSEとCLKOUT1_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE2_EN	string	“TRUE","FALSE"	“FALSE"	チャネル2(ODIV2 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE2_SEL="TRUE"の場合、CLKOUT2_PE_COARSEとCLKOUT2_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE3_EN	string	“TRUE","FALSE"	“FALSE"	チャネル3(ODIV3 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE3_SEL="TRUE"の場合、CLKOUT3_PE_COARSEとCLKOUT3_PE_FINEを立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE4_EN	string	“TRUE","FALSE"	“FALSE"	チャネル4(ODIV4 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● "TRUE"：DYN_PE4_SEL="TRUE" の場合、CLKOUT4_PE_COARSE と CLKOUT4_PE_FINE を立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE5_EN	string	“TRUE","FALSE"	“FALSE"	チャネル 5(ODIV5 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE5_SEL="TRUE" の場合、CLKOUT5_PE_COARSE と CLKOUT5_PE_FINE を立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
DE6_EN	string	“TRUE","FALSE"	“FALSE"	チャネル 6(ODIV6 = 2～128)のデューティサイクル調整イネーブル ● "FALSE"：50%のデューティサイクル ● "TRUE"：DYN_PE6_SEL="TRUE" の場合、CLKOUT6_PE_COARSE と CLKOUT6_PE_FINE を立ち下がりエッジとして設定し、動的位相調整を立ち上がりエッジとして設定することで、動的デューティサイクル調整(立ち下がりエッジ - 立ち上がりエッジ)を実現します。
RESET_I_EN	string	"TRUE", "FALSE"	"FALSE"	動的信号 RESET_I イネーブル。RESET_I ポートを使用するには、このパラメータを TRUE に設定する必要があります。
RESET_0_EN	string	"TRUE", "FALSE"	"FALSE"	動的信号 RESET_O イネーブル。RESET_O ポートを使用するには、このパラメータを TRUE に設定する必要があります。
DYN_ICP_SEL	string	"TRUE", "FALSE"	"FALSE"	ICPSEL 静的制御パラメータまたは動的制御信号の選択 ● FALSE: 静的制御、つまりパラメータ ICP_SEL を選択します

パラメータ名	タイプ	値の範囲	デフォルト値	説明
				● TRUE: 動的制御、つまり動的信号 ICPSEL を選択します
ICP_SEL	binary	6'bXXXXXX, 6'b000000~6'b111111	6'bXXXXXX	ICP 電流の静的設定 ● 6'bXXXXXX：ソフトウェアが自動的に計算してこのパラメータを設定します ● 6'b000000~6'b111111：ユーザーは、必要に応じてパラメータ範囲内で設定できます
DYN_LPF_SEL	string	"TRUE", "FALSE"	"FALSE"	LPFRES と LPFCAP 静的制御パラメータまたは動的制御信号の選択 ● "FALSE"：静的制御、つまりパラメータ LPF_RES と LPF_CAP を選択します ● "TRUE"：動的制御、つまり動的信号 LPFREST と LPFCAP を選択します
LPF_RES	binary	3'bXXX,3'b000~3'b111	3'bXXX	LPRRES の静的設定 ● 3'bXXX：ソフトウェアが自動的に計算してこのパラメータを設定します ● 3'b000~3'b111：ユーザーは、必要に応じてパラメータ範囲内で設定できます
LPF_CAP	binary	2'b00~2'b10	2'b00	LFPCAP の静的設定
SSC_EN	string	"TRUE", "FALSE"	"FALSE"	SSC イネーブル

入力クロック分周器(IDIV)は、PLL モジュールへの入力クロック周波数を制御するために使用されます。この分周係数は、パラメータ IDIV_SEL で静的に調整することができます。その対応関係を表 5-14 に示します。

表 5-14 IDIV の対応関係

IDIV_SEL(静的)	IDIV の実際値
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
64	64

FBDIV 分周器は、フィードバック信号の分周に使用されます。この分周係数は、パラメータ FBDIV_SEL で静的に調整することができます。その対応関係を表 5-15 に示します。

表 5-15 FBDIV の対応関係

FBDIV_SEL（静的）	FBDIV の実際値
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
64	64

出力分周器（ODIV）のチャンネル 0 は整数分周と小数分周をサポートし、チャンネル 1～6 は整数分周のみをサポートします。チャンネル 0 の場合、この分周係数はパラメータ ODIV0_SEL と ODIV0_FRAC_SEL で静的に調整することができます。 整数の分周係数の対応関係を表 5-16 に示し、小数の分周係数の対応関係を表 5-17 に示します。

表 5-16 ODIV0 整数値の対応関係

ODIV0_SEL（静的）	ODIV0 整数値
1	1

ODIV0_SEL（静的）	ODIV0 整数値
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
……	……
128	128

ODIV0 は、整数分周値が 2~127 の場合のみ小数分周が有効です。チャンネル 1~6 の場合、ODIV 分周係数を ODSELx(x=1~6)で動的に調整するか、またはパラメータ ODIVx_SEL(x=1~6)で静的に調整することができます。その対応関係についてはチャンネル 0 の小数分周、すなわち表 5-17 を参照してください。

表 5-17 ODIV0 小数値の対応関係

ODIV0_FRAC_SEL（静的）	ODIV0 小数値
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 分周器の機能は FBDIV と同様です。整数と小数の分周係数がサポートされます。この分周係数はパラメータ MDIV_SEL と MDIV_FRAC_SEL で静的に調整することができます。整数の分周係数の対応関係を表 5-18 に示し、小数の分周係数の対応関係を表 5-19 に示します。

表 5-18 MDIV 整数値の対応関係

MDIV_SEL（静的）	MDIV 整数値
2	2
3	3
4	4
5	5

MDIV_SEL（静的）	MDIV 整数値
6	6
7	7
8	8
9	9
……	……
128	128

MDIV 整数値の範囲は 2~128 です。

表 5-19 MDIV 小数値の対応関係

MDIV_FRAC_SEL(静的)	MDIV 小数値
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 は、整数分周値が 2~127 の場合のみ小数分周が有効です。

位相の調整

PLLA は位相の静的調整と動的調整をサポートします。また、チャンネル 0~6 はすべて位相調整をサポートします。位相調整は、MDIV チャンネルを基準とした、MDIV に対する ODIV0~6 の位相シフトです。

静的位相調整は、パラメータ CLKOUTx_PE_COARSE と CLKOUTx_PE_FINE (x=0~6)の設定により実現されます。

動的位相調整は、信号 PSSEL、PSDIR、および PSPULSE によって実現されます。PSSEL はチャネルの選択に使用され、PSDIR は加算または減算の制御に使用されます。PSPULSE パルスの立ち下がりエッジごとに、DYN_FINE は 1 増加/減少し、DYN_FINE のオーバーフローまたはアンダーフローの場合、DYN_COARSE は 1 増加/減少します。DYN_COARSE の値は ODIV より小さいです。

位相調整の計算式は次のとおりです。

PS=(COARSE+FINE/8)/ODIV*360，PS の範囲は[0,360)です。

チャネル 0 の場合、ODIV=ODIV0 整数値+ODIV0 小数値です。チャネル 1~6 の場合、ODIV は整数値のみです。

注記：

- DYN_FINE および DYN_COARSE は、DPA によって生成される内部信号であり、PSSEL、PSDIR、および PSPULSE の協働により生成されます。その機能は CLKOUTx_PE_COARSE および CLKOUTx_PE_FINE と同じです。
- 式中の COARSE と FINE は、動的または静的調整を選択した場合の、位相調整に実際に有効な値を指します。
- 位相調整回路は、VCO 向けに設計されています。Bypass in または CAS モードの場合、位相調整式は適用できますが，FINE を 0 に設定する必要があります。

デューティサイクルの調整

PLLA は、デューティサイクルの動的調整のみをサポートします。また、チャンネル 0~6 はすべてデューティサイクルの調整をサポートします。デューティサイクルは次のように定義されます。

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

falling edge の位置は、静的位相シフト設定によって決定され、DUTY として定義されます。rising edge の位置は、動的位相シフト設定の PHASE によって決定されます。DYN_FINE および DYN_COARSE は、DPA によって生成される内部信号です。位相調整セクションの関連説明を参照してください。DUTY と PHASE の計算式は次のとおりです(チャネル 1 の場合)。

DUTY=(CLKOUT1_PE_COARSE+CLKOUT1_PE_FINE/8)

PHASE=(DYN_COARSE1+DYN_FINE1/8)

動的デューティサイクルの計算：

- DUTY>PHASE の場合、Duty cycle=(DUTY-PHASE)/ODIV1
- DUTY<PHASE の場合、Duty cycle=(DUTY-PHASE)/ODIV1+1

注記：

- ODIV = 1 の場合、デューティサイクルの動的調整はサポートされておらず、デューティサイクルは常に 50%です。
- ODIV> = 2 の場合、DUTY-PHASE は(-0.5, 0.5)内の値をサポートしません。
- デューティサイクル調整回路は、VCO 向けに設計されています。Bypass in または CAS モードの場合、デューティサイクルの調整はサポートされていません。さらに、Bypass in または CAS モードでは、ODIV(>2)が奇数の場合、デューティサイクルは 50%ではありません（High レベル＜Low レベル、すなわちデューティサイクルは 50%未満となります）。

デューティサイクルの微調整

PLLA は、デューティサイクルの静的微調整と動的微調整をサポートします(チャネル 0~3 のみ)。デューティサイクルの微調整は、デューティサイクルの微調整方向とステップを設定することでサポートされます。微調整方向が 1'b1 の場合、立ち下がりエッジの遅延が調整されてデューティサイクルが増加します。微調整方向が 1'b0 の場合、立ち上がりエッジの遅延が調整されてデューティサイクルが減少します。遅延値の詳細は、表 5-20 に示すとおりです(チャネル 1 の場合)。

表 5-20 PLLA のデューティサイクルの微調整の参照テーブル

動的調整	静的調整		デューティサイクルの微調整の遅延値
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

例えば、同じ周波数のクロックを出力するチャンネル 1 とチャンネル 2 がある場合、チャンネル 1 のクロックのデューティサイクルの微調整タイミングは、図 5-8 と図 5-9 に示すとおりです(チャネル 0 は基準として使用される)。

図 5-8 チャネル 1 のデューティサイクルの微調整タイミング図(微調整方向が 1'b1、ステップ数が 1)

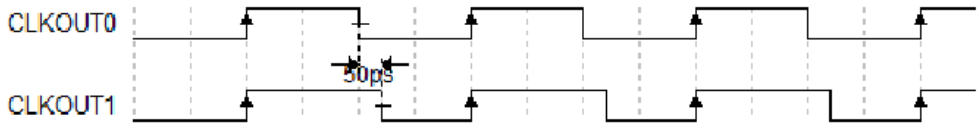
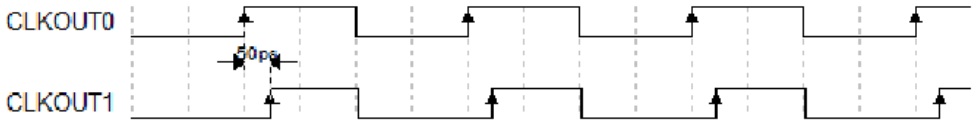


図 5-9 チャネル 1 のデューティサイクルの微調整タイミング図(微調整方向が 1'b0、ステップ数が 1)



プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

```
        .CLKOUT5(clkout5),
        .CLKOUT6(clkout6),
        .CLKFBOUT(clkfbout),
        .CLKIN(clkin),
        .CLKFB(clkfb),
        .RESET(reset),
        .PLLPWD(pllpwd),
        .RESET_I(reseti),
        .RESET_O(reseto),
        .PSSEL(pssel),
        .PSDIR(psdir),
        .PSPULSE(pspulse),
        .SSCPOL(sscpol),
        .SSCON(sscon),
        .SSCMDSEL(sscmdsel),
        .SSCMDSEL_FRAC(sscmdsel_frac),
        .MDCLK(mdclk),
        .MDOPC(msopc),
        .MDAINC(mdainc),
        .MDWDI(mdwdi),
        .MDRDO(mdrdo)
    );
    defparam uut.CLK0_IN_SEL=1'b0;
    defparam uut.CLK0_OUT_SEL=1'b0;
    defparam uut.CLK1_IN_SEL=1'b0;
    defparam uut.CLK1_OUT_SEL=1'b0;
    defparam uut.CLK2_IN_SEL=1'b0;
    defparam uut.CLK2_OUT_SEL=1'b0;
    defparam uut.CLK3_IN_SEL=1'b0;
    defparam uut.CLK3_OUT_SEL=1'b0;
    defparam uut.CLK4_IN_SEL=2'b00;
    defparam uut.CLK4_OUT_SEL=1'b0;
    defparam uut.CLK5_IN_SEL=1'b0;
    defparam uut.CLK5_OUT_SEL=1'b0;
    defparam uut.CLK6_IN_SEL=1'b0;
    defparam uut.CLK6_OUT_SEL=1'b0;
    defparam uut.CLKFB_SEL="INTERNAL";
    defparam uut.CLKOUT0_DT_DIR=1'b1;
    defparam uut.CLKOUT0_DT_STEP=0;
    defparam uut.CLKOUT0_EN="TRUE";
    defparam uut.CLKOUT0_PE_COARSE=0;
    defparam uut.CLKOUT0_PE_FINE=0;
```

```
defparam uut.CLKOUT1_DT_DIR=1'b1;
defparam uut.CLKOUT1_DT_STEP=0;
defparam uut.CLKOUT1_EN="FALSE";
defparam uut.CLKOUT1_PE_COARSE=0;
defparam uut.CLKOUT1_PE_FINE=0;
defparam uut.CLKOUT2_DT_DIR=1'b1;
defparam uut.CLKOUT2_DT_STEP=0;
defparam uut.CLKOUT2_EN="FALSE";
defparam uut.CLKOUT2_PE_COARSE=0;
defparam uut.CLKOUT2_PE_FINE=0;
defparam uut.CLKOUT3_DT_DIR=1'b1;
defparam uut.CLKOUT3_DT_STEP=0;
defparam uut.CLKOUT3_EN="FALSE";
defparam uut.CLKOUT3_PE_COARSE=0;
defparam uut.CLKOUT3_PE_FINE=0;
defparam uut.CLKOUT4_EN=" TRUE ";
defparam uut.CLKOUT4_PE_COARSE=0;
defparam uut.CLKOUT4_PE_FINE=0;
defparam uut.CLKOUT5_EN="FALSE";
defparam uut.CLKOUT5_PE_COARSE=0;
defparam uut.CLKOUT5_PE_FINE=0;
defparam uut.CLKOUT6_EN="FALSE";
defparam uut.CLKOUT6_PE_COARSE=0;
defparam uut.CLKOUT6_PE_FINE=0;
defparam uut.DE0_EN="FALSE";
defparam uut.DE1_EN="FALSE";
defparam uut.DE2_EN="FALSE";
defparam uut.DE3_EN="FALSE";
defparam uut.DE4_EN="FALSE";
defparam uut.DE5_EN="FALSE";
defparam uut.DE6_EN="FALSE";
defparam uut.DYN_DPA_EN="FALSE";
defparam uut.DYN_PE0_SEL="FALSE";
defparam uut.DYN_PE1_SEL="FALSE";
defparam uut.DYN_PE2_SEL="FALSE";
defparam uut.DYN_PE3_SEL="FALSE";
defparam uut.DYN_PE4_SEL="FALSE";
defparam uut.DYN_PE5_SEL="FALSE";
defparam uut.DYN_PE6_SEL="FALSE";
defparam uut.FBDIV_SEL=1;
defparam uut.FCLKIN="100.0";
defparam uut.ICP_SEL=6'bXXXXXX;
```

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

VHDL でのインスタンス化：

```
COMPONENT PLLA
    GENERIC(
        FCLKIN : STRING := "100.0";
        IDIV_SEL : integer := 1;
        FBDIV_SEL : integer := 1;
        ODIV00_SEL : integer := 8;
        ODIV1_SEL : integer := 8;
        ODIV2_SEL : integer := 8;
        ODIV3_SEL : integer := 8;
        ODIV4_SEL : integer := 8;
        ODIV5_SEL : integer := 8;
        ODIV6_SEL : integer := 8;
        MDIV_SEL : integer := 8;
        MDIV_FRAC_SEL : integer := 0;
        ODIV0_FRAC_SEL : integer := 0;
        CLKOUT0_EN : STRING := "TRUE";
        CLKOUT1_EN : STRING := " FALSE ";
        CLKOUT2_EN : STRING := " FALSE ";
        CLKOUT3_EN : STRING := " FALSE ";
        CLKOUT4_EN : STRING := " FALSE ";
```

```
CLKOUT5_EN : STRING := " FALSE ";
CLKOUT6_EN : STRING := " FALSE ";
CLKOUT0_DT_DIR : bit := '1';
CLKOUT1_DT_DIR : bit := '1';
CLKOUT2_DT_DIR : bit := '1';
CLKOUT3_DT_DIR : bit := '1';
CLKOUT0_DT_STEP : integer := 0;
CLKOUT1_DT_STEP : integer := 0;
CLKOUT2_DT_STEP : integer := 0;
CLKOUT3_DT_STEP : integer := 0;
CLK0_IN_SEL   : bit := '0';
CLK0_OUT_SEL : bit := '0';
CLK1_IN_SEL   : bit_vector := '0';
CLK1_OUT_SEL : bit := '0';
CLK2_IN_SEL   : bit_vector := '0';
CLK2_OUT_SEL : bit := '0';
CLK3_IN_SEL   : bit_vector := '0';
CLK3_OUT_SEL : bit := '0';
CLK4_IN_SEL   : bit_vector := "00";
CLK4_OUT_SEL : bit := '0';
CLK5_IN_SEL   : bit_vector := '0';
CLK5_OUT_SEL : bit := '0';
CLK6_IN_SEL   : bit_vector := '0';
CLK6_OUT_SEL : bit := '0';
CLKFB_SEL : STRING := "INTERNAL";
DYN_DPA_EN : STRING := "FALSE";
DYN_PE0_SEL : STRING := "FALSE";
DYN_PE1_SEL : STRING := "FALSE";
DYN_PE2_SEL : STRING := "FALSE";
DYN_PE3_SEL : STRING := "FALSE";
DYN_PE4_SEL : STRING := "FALSE";
DYN_PE5_SEL : STRING := "FALSE";
DYN_PE6_SEL : STRING := "FALSE";
CLKOUT0_PE_COARSE : integer := 0;
CLKOUT0_PE_FINE : integer := 0;
```

```
        CLKOUT1_PE_COARSE : integer := 0;
        CLKOUT1_PE_FINE : integer := 0;
        CLKOUT2_PE_COARSE : integer := 0;
        CLKOUT2_PE_FINE : integer := 0;
        CLKOUT3_PE_COARSE : integer := 0;
        CLKOUT3_PE_FINE : integer := 0;
        CLKOUT4_PE_COARSE : integer := 0;
        CLKOUT4_PE_FINE : integer := 0;
        CLKOUT5_PE_COARSE : integer := 0;
        CLKOUT5_PE_FINE : integer := 0;
        CLKOUT6_PE_COARSE : integer := 0;
        CLKOUT6_PE_FINE : integer := 0;
        DE0_EN : STRING := "FALSE";
        DE1_EN : STRING := "FALSE";
        DE2_EN : STRING := "FALSE";
        DE3_EN : STRING := "FALSE";
        DE4_EN : STRING := "FALSE";
        DE5_EN : STRING := "FALSE";
        DE6_EN : STRING := "FALSE";
        RESET_I_EN : STRING := "FALSE";
        RESET_O_EN : STRING := "FALSE";
        ICP_SEL : std_logic_vector(5 downto 0) := "XXXXXX";
        LPR_RES : std_logic_vector(2 downto 0) := "XXX";
        LPR_CAP : bit_vector := "00";
        SSC_EN : STRING := "FALSE"
    );
    PORT(
        CLKIN : IN std_logic;
        CLKFB : IN std_logic:='0';
        RESET, PLLPWD : IN std_logic:='0';
        RESET_I, RESET_O : IN std_logic:='0';
        PSSEL : IN std_logic_vector(2 downto 0);
        PSDIR,PSPULSE : IN std_logic;
        SSCPOL, SSCON: IN std_logic;
        SSCMDSEL : IN std_logic_vector(6 downto 0);
```

```
                SSCMDSEL_FRAC : IN std_logic_vector(2 downto 0);
                MDCLK, MDINC : IN std logic;
                MDOPC : IN std logic vector(1 downto 0);
                MDWDI : IN std logic vector(7 downto 0);
                MDRDO : OUT std logic vector(7 downto 0);
                LOCK : OUT std_logic;
                CLKOUT0, CLKOUT1 : OUT std_logic;
                CLKOUT2, CLKOUT3 : OUT std_logic;
                CLKOUT4, CLKOUT5 : OUT std_logic;
                CLKOUT6, CLKFBOUT : OUT std_logic
        );
END COMPONENT;
uut:PLLA
        GENERIC MAP(
                FCLKIN => "100.0",
                IDIV_SEL => 1,
                FBDIV_SEL => 1,
                ODIV0_SEL => 8,
                ODIV1_SEL => 8,
                ODIV2_SEL => 8,
                ODIV3_SEL => 8,
                ODIV4_SEL => 8,
                ODIV5_SEL => 8,
                ODIV6_SEL => 8,
                MDIV_SEL => 8,
                MDIV_FRAC_SEL => 0,
                ODIV0_FRAC_SEL => 0,
                CLKOUT0_EN => "TRUE",
                CLKOUT1_EN => " FALSE ",
                CLKOUT2_EN => " FALSE ",
                CLKOUT3_EN => " FALSE ",
                CLKOUT4_EN => " FALSE ",
                CLKOUT5_EN => " FALSE ",
                CLKOUT6_EN => " FALSE ",
                CLKOUT0_DT_DIR => '1',
```

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

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

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

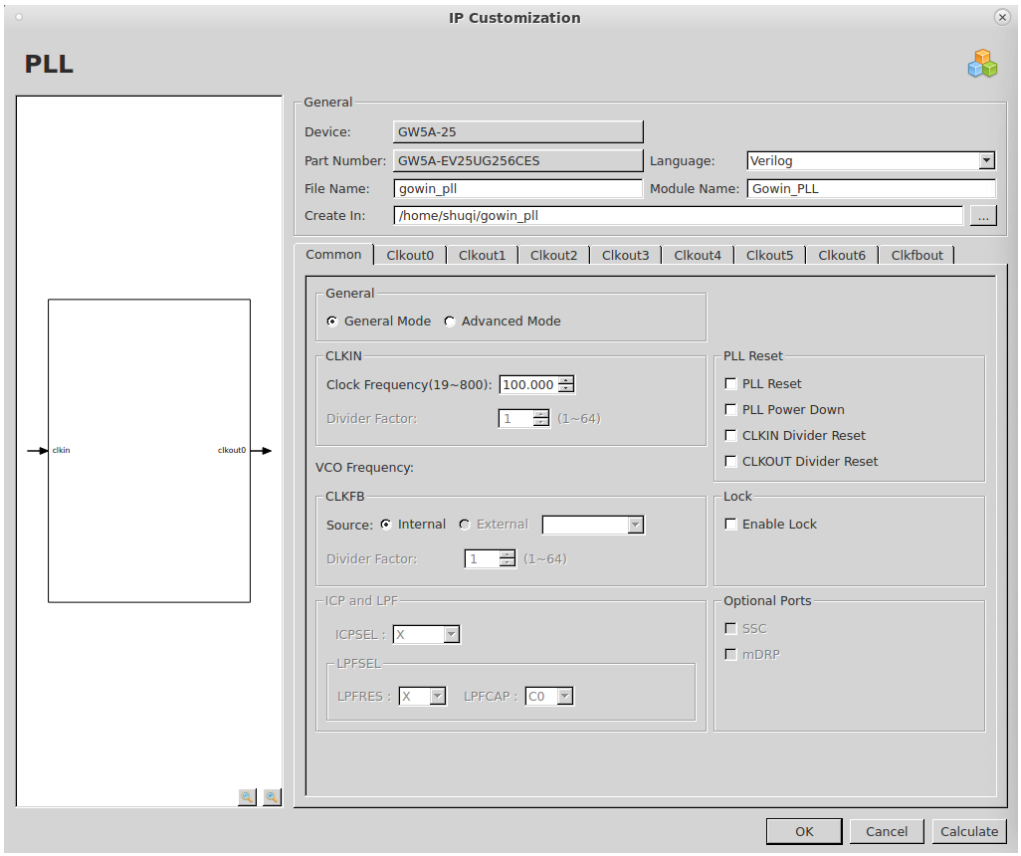
5.2.2 IP の呼び出し

IP Core Generator のインターフェースで“PLLA”をクリックすると、右側に PLLA の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“PLLA”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブ、Options 構成タブ、およびポート図があります。

図 5-10 PLLA IP の構成ウィンドウ



1. General 構成タブ

General 構成タブは、IP ファイルの構成に使用されます。PLLA の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。

2. Options 構成タブ

PLLA の Options 構成タブの使用は PLL モジュールと同様であるので、5.1.2 IP の呼び出しを参照してください。

3. ポート図
ポート図は、IP Core の構成結果を表示します(図 5-10)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_pll.v”は完全な verilog モジュールです。
- “gowin_pll_tmp.v”は IP のテンプレートファイルです。
- “gowin_pll.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

5.3 DQS

5.3.1 プリミティブの紹介

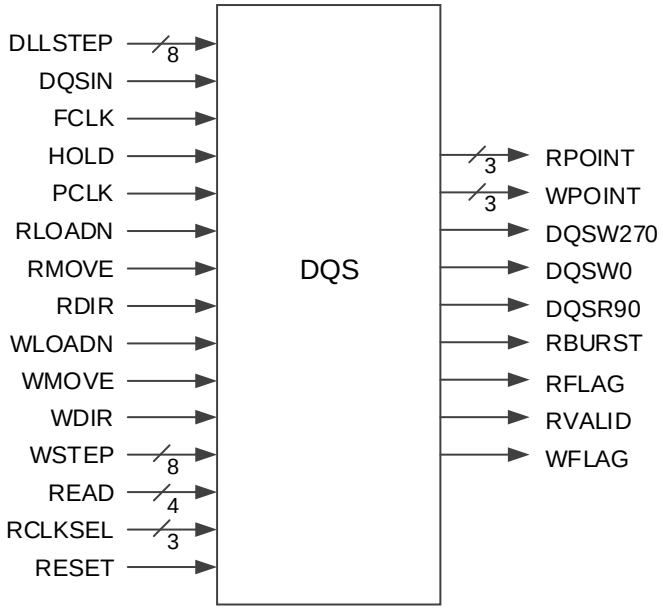
DQS は DDR メモリインターフェースの双方向データストローブ回路です。

機能の説明

DQS は、主に DQSIN と DQSR90、DQSW0 と DQSW270 の信号間の位相関係の調整、および書き込みレベリングと読み出しのキャリブレーションに使用される、メモリコントローラ IP のキーコンポーネントです。

ポート図

図 5-11 DQS のポート図



ポートの説明

表 5-21 DQS のポートの説明

ポート名	I/O	説明
DLLSTEP[7:0]	入力	DQS 遅延ステップ制御入力
DQSIN	入力	IO PAD からの DQS 入力
FCLK	入力	2 つの異なる FCLK クロックツリーの出力から得る高速クロック
HOLD	入力	DQS 書き込みに使用の場合、関連する信号の書き込みを停止して出力クロックに同期させるために使用されます。DQS 読み出しに使用の場合、FIFO カウンターをリセットするために使用されます。
PCLK	入力	PCLK クロックツリーからのプライマリクロック
RDIR	入力	DDR 読み出しの遅延方向を調整します ● "0"：遅延を増やします ● "1"：遅延を減らします
RLOADN	入力	DDR 読み出しの最終遅延ステップを初期化します。アクティブ Low
RMOVE	入力	RMOVE が立ち下がりエッジのときに DDR 読み出しの遅延ステップを変更します。パルスごとに 1 回変更します
WDIR	入力	DDR 書き込みの遅延方向を調整します ● "0"：遅延を増やします ● "1"：遅延を減らします
WLOADN	入力	DDR 書き込みの最終遅延ステップを初期化します。アクティブ Low
WMOVE	入力	WMOVE が立ち下がりエッジのときに DDR 書き込みの遅延ステップを変更します。パルスごとに 1 回変更します
WSTEP[7:0]	入力	DDR 書き込みレベリング遅延制御に使用されます
READ[3:0]	入力	DDR 読み出しモードに使用される READ 信号
RCLKSEL[2:0]	入力	読み出しクロックソースの選択と極性制御
RESET	入力	DQS リセット入力、アクティブ High
RPOINT[2:0]	出力	IOLOGIC の RADDR に作用する、または配線によりユーザーロジックに作用する FIFO 読み出しポインター
WPOINT[2:0]	出力	IOLOGIC の WADDR に作用する、または配線によりユーザーロジックに作用する FIFO 書き込みポインター
DQSW0	出力	IOLOGIC の TCLK に作用する、または配線によりユーザーロジックに作用する PCLK/FCLK 0°位相シフト出力。

ポート名	I/O	説明
DQSW270	出力	IOLOGIC の TCLK に作用する、または配線によりユーザーロジックに作用する PCLK/FCLK 270° 位相シフト出力。
DQSR90	出力	IOLOGIC の ICLK に作用する、または配線によりユーザーロジックに作用する DQSI 90° 位相シフト出力。
RFLAG	出力	読み出し遅延調整の under-flow または over-flow を示す READ 遅延調整出力フラグ
WFLAG	出力	書き込み遅延調整の under-flow または over-flow を示す WRITE 遅延調整出力フラグ
RVALID	出力	READ モードのデータ有効フラグ
RBURST	出力	READ バースト検出出力

パラメータの説明

表 5-22 DQS のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
FIFO_MODE_SEL	1'b0 , 1'b1	1'b0	FIFO モードの選択 ● 1'b0: DDR memory モード ● 1'b1: GDDR モード
RD_PNTR	"000", "001", "010", "011", "100", "101", "110", "111"	3'b000	FIFO 読み出しポインターの設定
DQS_MODE	"X1”, "X2_DDR2", "X2_DDR3", "X4", "X2_DDR3_EXT"	"X1"	DQS モードの選択
HWL	"false", "true"	"false"	update0/1 のタイミング関係の制御 ● "false": update1 は update0 より 1 サイクル先です。 ● "true": update1 と update 0 のタイミングは同じです。

接続ルール

- DQS の入力 DQSI は IO PAD からのものです。
- DQS の出力 RPOINT は、IOLOGIC の RADDR に接続するか、ユーザーロジックに作用できます。
- DQS の出力 WPOINT は、IOLOGIC の WADDR に接続するか、ユー

ザーロジックに作用できます。

- DQS の出力 DQSR90 は、IOLOGIC の ICLK に接続するか、ユーザーロジックに作用できます。
- DQS の出力 DQSW0/DQSW270 は、IOLOGIC の TCLK に接続するか、ユーザーロジックに作用できます。

プリミティブのインスタンス化

Verilog でのインスタンス化：

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

VHDL でのインスタンス化：

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

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

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

5.4 DDRDLL

5.4.1 プリミティブの紹介

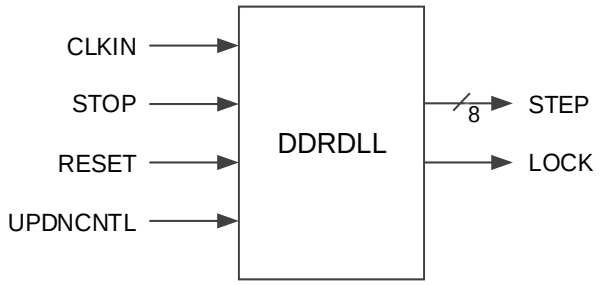
DDRDLL が生成する位相遅延は、DLLDLY や DQS などのモジュールを介して、DDR などのインターフェースモジュールに作用します。

機能の説明

DDRDLL は、与えられた入力クロックに基づいて位相の調整を実行し、さまざまな位相遅延に対応するステップ数を生成することができます。算出された STEP 信号は DQS や DLLDLY モジュールなどを駆動します。また、STEP 信号は配線によりユーザーロジックに接続することも可能です。DDRDLL のクロック入力ソースは、GCLK、または隣接する HCLK です。

ポート図

図 5-12 DDRDLL のポート図



ポートの説明

表 5-23 DDRDLL のポートの説明

ポート名	I/O	説明
STEP[7:0]	出力	遅延ステップ出力
LOCK	出力	ロック状態の指示 ● 1：ロック ● 0：ロック解除
CLKIN	入力	クロック入力
STOP	入力	入力クロックと内部発振クロックの停止
RESET	入力	非同期リセット、アクティブ High
UPDNCNTL	入力	DDRDLL 遅延ステップ更新の制御、アクティブ Low

パラメータの説明

表 5-24 DDRDLL のパラメータの説明

パラメータ名	パラメータのタイプ	値の範囲	デフォルト値	説明
DLL_FORCE	Integer	0,1	0	DDRDLL 強制遅延ステップおよびロックの制御 ● TRUE：強制ロック。低入力周波数モードの場合に使用されます。遅延ステップは最大で 255。 ● FALSE：ノーマルモード。DDRDLL で遅延ステップとロック信号を生成します。
CODESCAL	String	000,001,010,011,100,101, 110, 111	000	DDRDLL の位相シフトの構成 (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 の位相シフト機能の有効／無効を設定します。 ● TRUE：有効。パラメータ CODESCAL により位相シフトを設定します。 ● FALSE：無効。位相シフトはデフォルトで 90°。

パラメータ名	パラメータのタイプ	値の範囲	デフォルト値	説明
DIV_SEL	Integer	1’b0,1’b1	1’b0	DDRDLL ロックモードの選択： ● 1'b0：ノーマル・ロックモード ● 1'b1:ファースト・ロックモード

接続のルール

DDRDLL の出力 STEP は、DQS、DLLDLY モジュールに接続するか、配線によりユーザーロジックに接続することができます。

プリミティブのインスタンス化

Verilog でのインスタンス化：

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

VHDL でのインスタンス化

```
COMPONENT DLL
    GENERIC(
        DLL_FORCE: STRING := "FALSE";
        DIV_SEL: bit :='1';
        CODESCAL: STRING :="000";
        SCAL_EN: STRING :="true"
    );
    PORT(
        CLKIN:IN std_logic;
        STOP:IN std_logic;
```

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

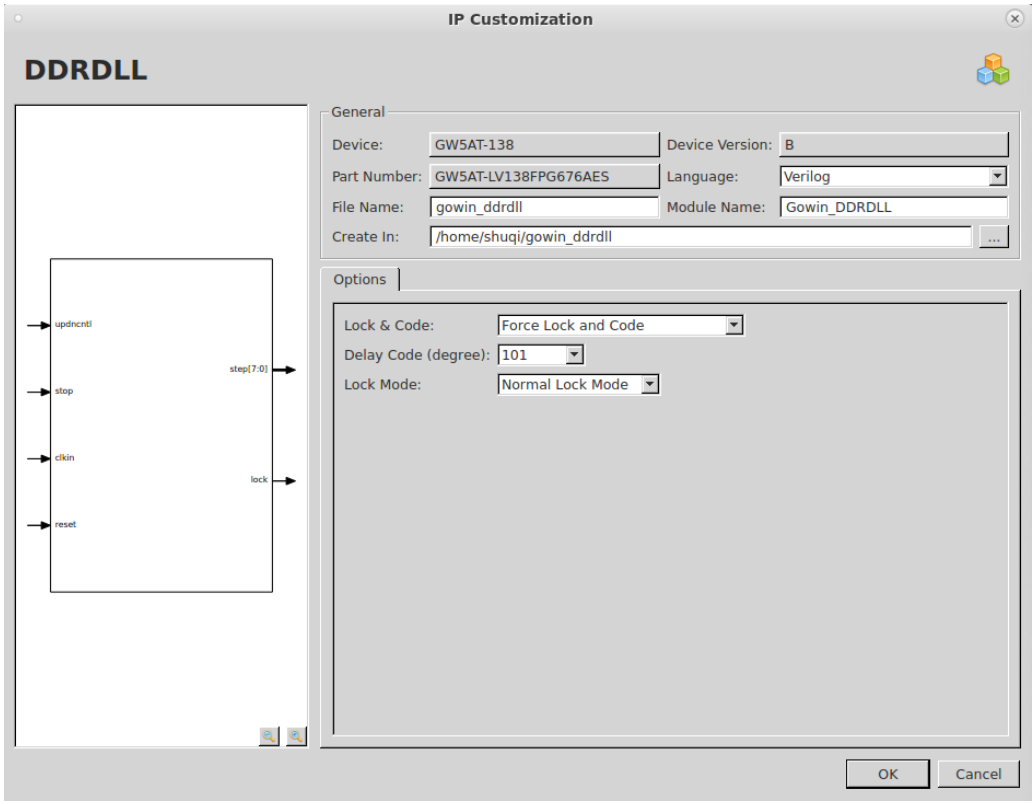
5.4.2 IP の呼び出し

IP Core Generator のインターフェースで“DDRDLL”をクリックすると、右側に DDRDLL の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“DDRDLL”をダブルクリックすると、DDRDLL の“IP Customization”ウィンドウがポップアップします。

図 5-13 DDRDLL IP の構成ウィンドウ



1. General 構成タブ

General 構成タブは、IP ファイルの構成に使用されます。DDRDLL の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。

2. Options 構成タブ

DDRDLL の Options 構成タブは IP のカスタマイズに使用されます(図 5-13)。

- Lock & Code：DDRDLL モードを設定します。
- Delay Code(degree)：遅延を設定します。
- Lock Mode：ロックモードを設定します。

3. ポート図
ポート図は、IP Core の構成結果を表示します(図 5-13)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_ddrdll.v”は完全な verilog モジュールです。
- “gowin_ddrdll_tmp.v”は IP のテンプレートファイルです。
- “gowin_ddrdll.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

6 オシレータ

6.1 OSC

6.1.1 プリミティブの紹介

OSC はプログラマブルなオンチップオシレータです。

サポートされるデバイス

表 6-1 OSC 対応デバイス

ファミリー	シリーズ	デバイス
Arora	GW5AT	GW5AT-138、B バージョンの GW5AT-138
	GW5AST	B バージョンの GW5AST-138

機能の説明

このプログラマブルなオンチップオシレータは、MSPI コンフィギュレーションモードまたはユーザーデザイン用のクロックソースとして使用できます。 動作パラメータの設定により、最大 64 のクロック周波数を取得できます。

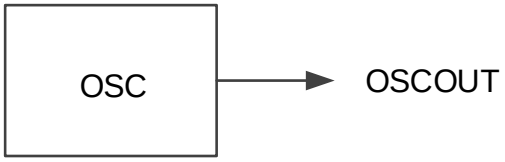
出力クロック周波数は以下の式で計算できます。

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

ここで、f_{osc}は 210MHz の OSC 発振周波数で、除数 FREQ_DIV は設定されるパラメータ(3 または 2～126 の偶数)です。

ポート図

図 6-1 OSC のポート図



ポートの説明

表 6-2 OSC のポートの説明

ポート名	I/O	説明
OSCOUT	出力	OSC の出力

パラメータの説明

表 6-3 OSC のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
FREQ_DIV	3 または 2~126 の偶数	100	OSC 分周係数の設定

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

VHDL でのインスタンス化：

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

6.1.2 IP の呼び出し

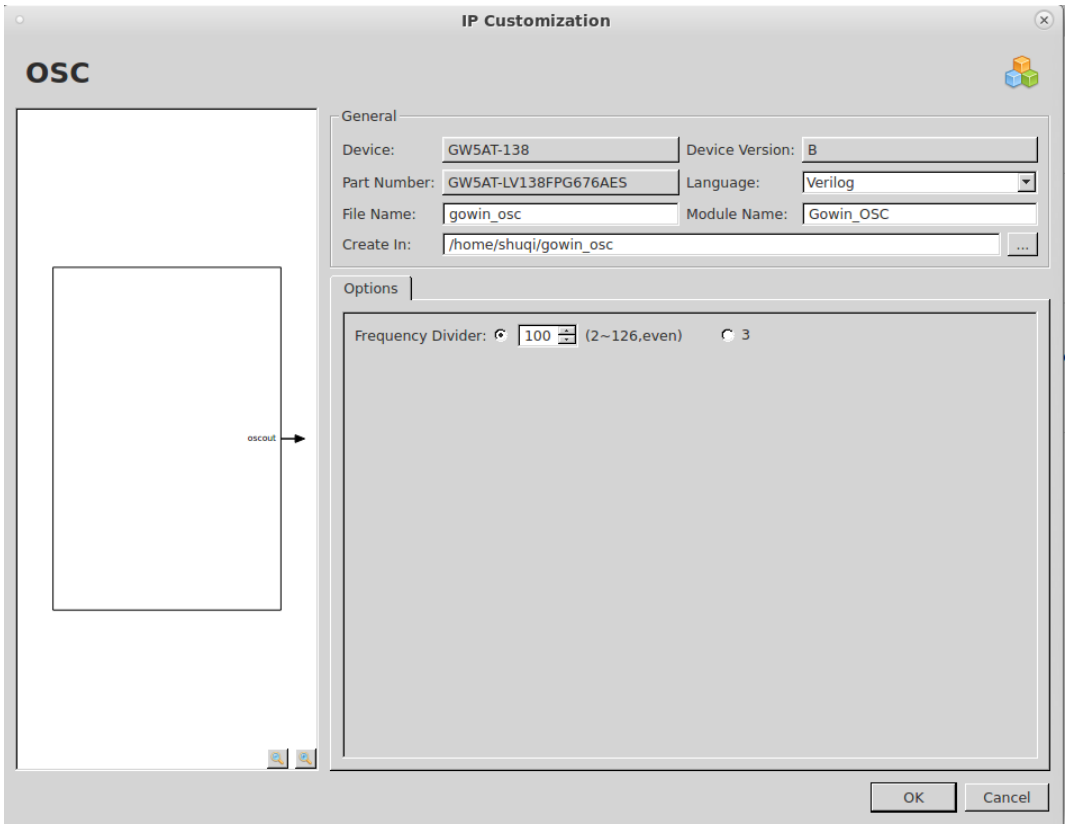
IP Core Generator のインターフェースで OSC をクリックすると、右側に OSC の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“OSC”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィン

ドウには、General 構成タブ、Options 構成タブ、およびポート図があります(図 6-2)。

図 6-2 OSC IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。OSC の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. Options 構成タブ
   Options 構成タブは IP のカスタマイズに使用されます(図 6-2)。
   - Frequency Divider：分周値。この値の範囲は 3 または 2~126 の偶数です。
3. ポート図

   ポート図は、IP Core の構成結果を表示します(図 6-2)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_osc.v”は完全な verilog モジュールです。
- “gowin_osc_tmp.v”は IP のテンプレートファイルです。
- gowin_osc.ipc は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

6.2 OSCA

6.2.1 プリミティブの紹介

OSCA はプログラマブルなオンチップオシレータです。

サポートされるデバイス

表 6-4 OSCA 対応デバイス

ファミリー	シリーズ	デバイス
Arora	GW5A	GW5A-25

機能の説明

このプログラマブルなオンチップオシレータは、MSPI コンフィギュレーションモードにクロックソースを提供し、動的 ON/OFF をサポートします。さらに、ユーザーデザインにクロックソースを提供することもできます。動作パラメータを設定することにより、最大 64 のクロック周波数を取得できます。

出力クロック周波数は以下の式で計算できます。

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

ここで、f_{osc}は 210MHz の OSC 発振周波数で、除数 FREQ_DIV は設定されるパラメータ(3 または 2～126 の偶数)です。

ポート図

図 6-3 OSCA のポート図

ポートの説明

表 6-5 OSCA のポートの説明

ポート名	I/O	説明
OSCOUT	出力	OSC の出力
OSCEN	入力	クロックイネーブル信号、アクティブ High

パラメータの説明

表 6-6 OSCA のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
FREQ_DIV	3 または 2~126 の偶数	100	OSC 分周係数の設定

プリミティブのインスタンス化

プリミティブを直接インスタンス化することができます。

Verilog でのインスタンス化：

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

VHDL でのインスタンス化：

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

6.2.2 IP の呼び出し

IP Core Generator のインターフェースで OSCA をクリックすると、右側に OSCA の概要が表示されます。

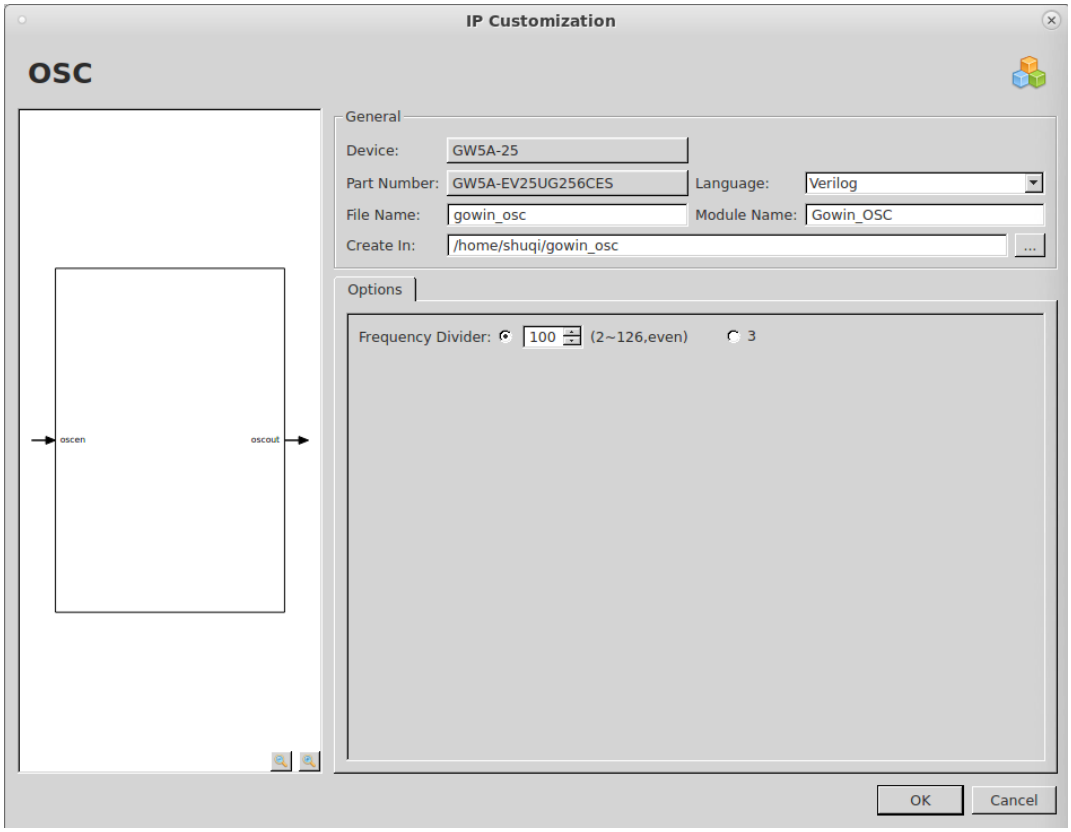
IP の構成

IP Core Generator インターフェースで“OSCA”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、General 構成タブ、Options 構成タブ、およびポート図があ

ります(図 6-4)。

IP Core Generator のインターフェースで OSCA をクリックすると、右側に OSCA の概要が表示されます。

図 6-4 OSCA IP の構成ウィンドウ



1. General 構成タブ
   General 構成タブは、IP ファイルの構成に使用されます。OSCA の General 構成タブの使用は DCE モジュールと同様であるので、3.1.2 IP の呼び出しを参照してください。
2. Options 構成タブ

   OSCA の Options 構成タブの使用は OSC モジュールと同様であるので、6.1.2 IP の呼び出しを参照してください。
3. ポート図

   ポート図は、IP Core の構成結果を表示します(図 6-4)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_osc.v”は完全な verilog モジュールです。
- “gowin_osc_tmp.v”は IP のテンプレートファイルです。
- “gowin_osc.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。


GOWIN高云
智慧逻辑 定制未来