



Power Management and Calculation for CrossLink Devices

Technical Note

FPGA-TN-02018 Version 1.1

October 2018

Contents

Acronyms in This Document	4
1. Introduction	5
2. Power Management Unit	5
2.1. PMU Block	5
2.1.1. PMU State Machine	6
2.1.2. State Transitions	7
2.1.3. Types of Triggers	8
2.1.4. Using the PMU	9
2.2. PMU Module, Primitive and Attributes	11
2.2.1. PMU Port Definitions and Parameters	12
2.2.2. PMU Control Bus and Primitive Attributes	13
2.3. Using Diamond Clarity Designer GUI to Generate PMU	19
3. User Standby Mode during Normal State	21
3.1. Usage of User Standby Mode	21
3.1.1. User Standby Mode for PLL	22
3.1.2. User Standby Mode for Oscillator	22
3.1.3. User Standby Mode for MIPI DPHY	22
3.1.4. User Standby Mode for Inputs and Outputs	22
4. Power Consumption and Calculation	25
4.1. Power Calculator	25
4.1.1. Typical and Worst Case Process	26
4.1.2. Junction Temperature	26
4.1.3. Maximum Safe Ambient Temperature	26
4.1.4. Operating Temperature Range	26
4.1.5. Dynamic Power Multiplier	27
4.1.6. Peak Startup Current	27
4.1.7. Power Budgeting	27
4.1.8. Device Operational Limits	27
4.1.9. Dynamic Power Savings	28
4.1.10. Activity Factor Calculation	28
4.1.11. Power Calculator Assumptions	28
4.2. Using Power Calculator for CrossLink Devices	29
4.2.1. Power Calculator in Normal State, and User Standby Mode	29
4.2.2. Power Calculator in Sleep State	30
4.2.3. Calculating Average Power	31
5. Thermal Management	32
5.1. Thermal Impedance and Airflow	32
5.2. Thermal Management in Power Calculator	32
5.3. DELPHI Models	33
6. Conclusion	33
References	34
Technical Support Assistance	34
Revision History	34

Figures

Figure 2.1. PMU Block Diagram	5
Figure 2.2. CrossLink PMU State Machine	6
Figure 2.3. Using USRWKUPN Pin for sleep and wakeup (functional depiction, not timing)	10
Figure 2.4. Using GPIO for SLEEP when directly routed to PMU (functional depiction, not timing)	10
Figure 2.5. PMU HDL Primitive –PMUA	12
Figure 2.6. PMU Control Bus Write Operation	14
Figure 2.7. Clarity Designer PMU Selection	19
Figure 2.8. PMU Instance Name Options Window	20
Figure 2.9. PMU Configuration GUI Window	20
Figure 3.1. User Standby Mode Use Case Example Block Diagram	21
Figure 3.2. Primitive for Bank Controller for Inputs – Referenced and Differential	23
Figure 3.3. INRDB Simulation Primitive	23
Figure 3.4. Bank Controller for LVDS Outputs Primitive	24
Figure 3.5. LVDSOB Simulation Primitive	24
Figure 4.1. Power Calculator (Summary Tab for CrossLink Devices)	25
Figure 4.2. Power Controller Window for User Standby Mode options	29
Figure 4.3. Power Calculator – Power Modes/ Avg. Power Tab (Power by Supply)	30
Figure 4.4. Power Calculator – Power Modes/ Avg. Power Tab (Power by Blocks)	31

Tables

Table 2.1. PMU HDL Primitive Port Definitions	13
Table 2.2. PMU Control Bus Address Mapping	14
Table 2.3. PMU Control Register (PMUCR) bit map	15
Table 2.4. PMU Control Register (PMUCR) Component Port Definition	16
Table 2.5. PMU Watch Dog Timer Control Register 1 (PMUWDTCR1) bit map	17
Table 2.6. PMUWDTCR1 Definitions and Settings	17
Table 2.7. PMU Watch Dog Timer Counter 1 (PMUWDCNT1) bit map	17
Table 2.8. PMU Watch Dog Timer Counter 2 (PMUWDCNT2) bit map	18
Table 2.9. PMU Watch Dog Timer Counter 3 (PMUWDCNT3) bit map	18
Table 2.10. PMU Watch Dog Timer Counter 3 (PMUWDCNT3) bit map	18
Table 3.1. Blocks Supporting User Standby Mode	21
Table 3.2. BCINRD Primitive Attributes	23
Table 3.3. BCINRD Simulation Behavior	23
Table 3.4. BCLVDSOB Primitive Attributes	24
Table 3.5. BCLVDSOB Simulation Behavior	24

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AF%	Activity Factor Percentage
AP	Application Processor
EBR	Embedded Block RAM
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HDL	Hardware Descriptive Language (Verilog or VHDL)
IO	Input Output
LUT	Look-Up Table
LVDS	Low Voltage Differential Signalling
OSC	Oscillator
PLL	Phase Locked Loop
PMUCR	PMU Control Register
PMUWDTCR	PMU Watch Dog Timer Control Register
PMUWDTCNT	PMU Watch Dog Timer Counter
T_J, T_A, T_B, T_C	Junction, Ambient, Board and Case Temperatures
USR	User Interface
WDT	Watch Dog Timer

1. Introduction

CrossLink™ devices from Lattice Semiconductor help stretch the battery life of an application by lowering the power consumption. CrossLink is a low power FPGA device with a configurable Power Management Unit (PMU) block that places the device in lower power consumption state (*Sleep State*) to save power, reduce average power consumption and increase battery life for the applications it is used in.

This document serves as a usage guide for managing and determining power consumption of the CrossLink device family. The document details the conceptual and functional description along with a guide to utilize the features of the PMU. Power consumption in Diamond Power Calculator (power saving feature of CrossLink) is also described.

2. Power Management Unit

The Power Management Unit (PMU) is a configurable IP available in CrossLink devices. The PMU allows the device to be placed in a lower power consumption state called Sleep State when the device is not active. This helps reduce the average power consumption of the device which can be beneficial in portable or mobile applications.

The HDL (Verilog or VHDL) module can be generated through the Clarity Designer in Diamond software. It is configurable based on application requirements. Each configuration settings help set up options unique to the application and can place the device in Sleep State or wake up to active Normal State. PMU works closely with user applications or application processor, providing dynamic options to user like wake up (using different signals), or when to wake (using timers).

2.1. PMU Block

Power Management Unit has two active states – Normal State and Sleep State. The device is active and running in Normal State. During Sleep State, the device is placed in a low power consuming state for periods when the device is not required to be active (or operational).

Figure 2.1 shows an overview of the PMU IP. The functional and usage descriptions of the PMU are detailed in the sections below.

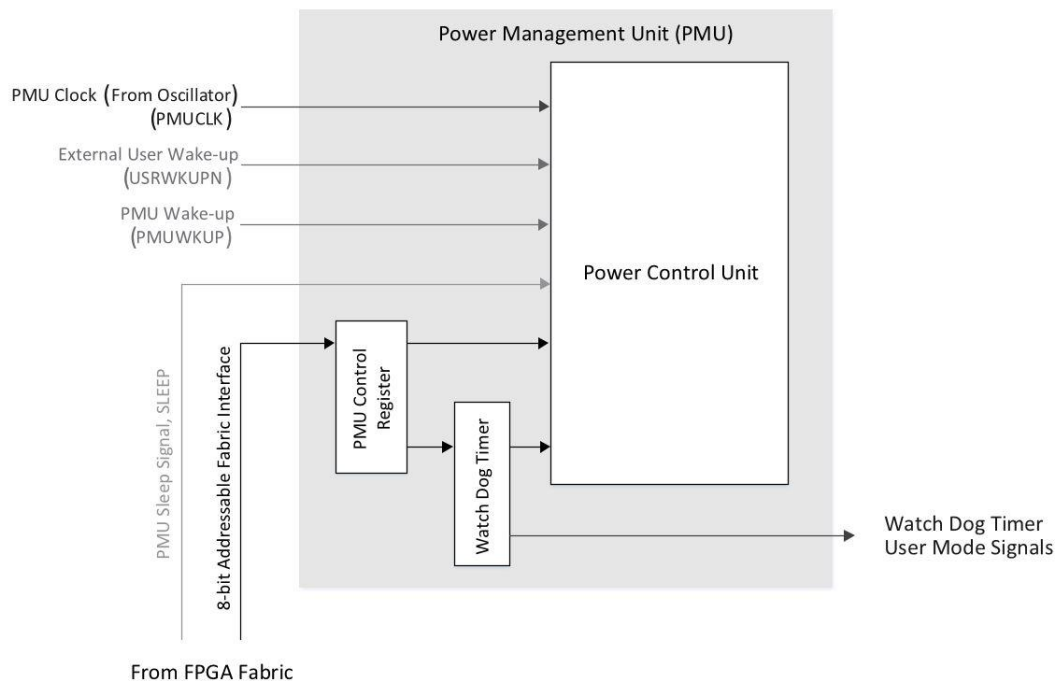


Figure 2.1. PMU Block Diagram

The low speed PMU clock coming in from the internal oscillator is the heart of the PMU. This connection is hardwired from the oscillator to the PMU. The speed of operation for the PMU is fixed at 10 kHz.

The brain of the PMU IP is a Power Control Unit, which essentially controls the various power supplies of the device. Power Control Unit has hard wired connections to the device's internals that control the power supply switches. These connections are not user accessible.

There are a number of inputs to the Power Control Units. One of the most important ones is the PMU Sleep Signal (SLEEP). This is an active high signal, which when triggered, sends a command to the Power Control Unit to place the device in a low power Sleep State.

There are two wake up signals – External User Wake-Up (USRWKUPN) and PMU Wake-Up (PMUWKUP). External User Wake-Up (USRWKUPN) is a shared dedicated pin that is hardwired to the PMU block.

Along with the Power Control Unit, there are two other blocks inside the PMU — PMU Control Register and Watch Dog Timer.

PMU Control Register

The PMU Control Register (also referred to as PMUCR in this document) stores the configuration settings of the PMU, such as how does the PMU wake up (from external pin or Watch Dog Timer expiry), the settings of the Watch Dog Timer, and so on.

PMUCR also has a dedicated 8-bit addressable PMU Control Bus interface to the FPGA fabric. This interface is very useful for applications that demand dynamic update of the PMU configuration as it can be used to update the configuration. The FPGA fabric has to be active (that is Normal State) in order to access the PMUCR via this interface. During the Sleep State, the FPGA fabric interface is inactive.

Watch Dog Timer

The Watch Dog Timer (WDT) is the final block of the PMU, and is used to set timers. The PMU can be configured to wake up on the expiry of these WDT timers. An example for a timer would be the timer available in our cell phones. You can set a timer for your device to wake up, say, in 10 minutes. When the PMU places the CrossLink device in Sleep State, the Watch Dog Timer is available exclusively for the PMU, whether it is set to wake up the device upon expiry or not.

WDT is not used by PMU in the Normal State, and has an optional user mode available. This WDT resource is made available to the application to perform functions that require timer(s). If the SLEEP signal is sent to the PMU, then going to Sleep State takes precedence over the user mode of Watch Dog Timer and it stops all operations and goes to sleep. Any required housekeeping needs to be taken care of at the user end (see the [Using the PMU](#) section on page 9).

2.1.1. PMU State Machine

As already mentioned, there are two states for PMU in CrossLink devices – Normal State and Sleep State. Both are mutually exclusive and the device can be placed in either one of the states. [Figure 2.2](#) shows the PMU's State Machine and triggers that helps transition one to the other.

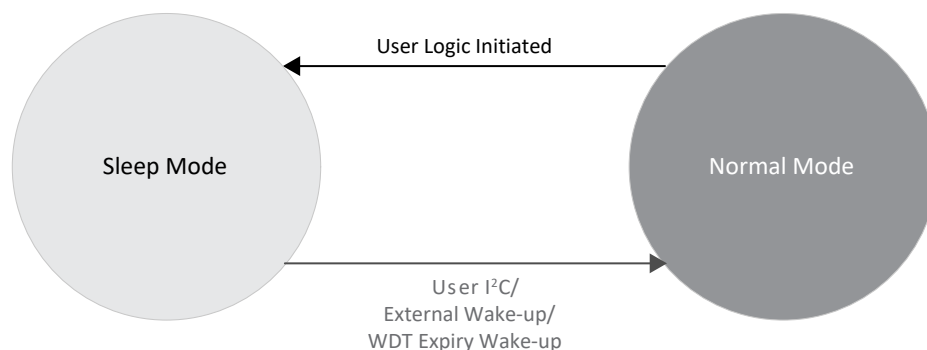


Figure 2.2. CrossLink PMU State Machine

Each state is discussed in detail below. It is important to understand the transition to each states, and the triggers that change the state from one to the other.

2.1.1.1. Normal State

All elements of the device are active in the Normal State to the extent required by the design. The device is fully active and performing as required by the application.

The device is at its highest power consumption in the Normal State. There are certain blocks of the device that can be placed in User Standby mode if they are not required by the application (see the [User Standby Mode during Normal State](#) section on page 21). These blocks (PLL, I/Os, OSC, and MIPI DPHY) can be placed in a lower power state when they are neither used nor needed by the application. This helps save some power in the Normal State; however User Standby Mode is different from the Sleep State. In the User Standby Mode, these blocks have a smaller static leakage current associated with them; there is no dynamic current because these are inactive.

PMU's Sleep State, on the other hand, makes the device inactive and places everything in the lower power. Sleep State overrides the User Standby mode.

2.1.1.2. Sleep State

In the Sleep State, the device is power gated and is not operational. The configuration of the device (the application bitstream that the device is programmed with) and EBR contents are retained. When it transitions to Normal State, the device will operate with these contents preserved.

The device is at its lowest power during the Sleep State. This helps reduce the overall power consumption for the device.

The PMU block stays powered on and active along with the associated I/Os even during Sleep State.

Retention in Sleep State

- CrossLink devices retain configuration SRAM, EBR contents when in Sleep State
- All registers within the IPs (EBR registers for example), and PFU registers will not retain their state
- User I²C registers retain their state
- The I/Os hold their values during the Sleep State. The last value of each I/O is held to avoid issues with other devices connected downstream from the CrossLink device.
- Differential I/Os do not hold their value; all differential I/Os are tri-stated

2.1.2. State Transitions

[Figure 2.2](#) shows the two state transitions to move from one state to the other.

2.1.2.1. Normal to Sleep

This transition happens when user requests the device to go to sleep. The trigger can come from an external source or generated within the application; the SLEEP signal is toggled high and the PMU goes to Sleep State.

The PMU can only be placed in Sleep State through the SLEEP signal coming in from the FPGA fabric. There are no external pins wired directly to place the device to sleep.

Example

Assume an external trigger or signal coming from an AP (Application Processor), requesting the CrossLink device to be placed in Sleep State. This signal comes in through an I/O and initiates a 'go to sleep' command to the application logic. This logic can then initiate the SLEEP signal to PMU through the fabric interface, and the PMU places the device in Sleep State. The signal can be routed directly to the PMU's SLEEP input via FPGA fabric in a simple scenario. For more complex design requirements, Lattice recommends a more graceful shutdown of the functions and operations being performed, and then initiate the SLEEP signal to the PMU using a state machine (see the [Using the PMU](#) section on page 9).

2.1.2.2. Sleep to Normal

Before looking at triggers needed to wake up the device from the Sleep State, let us look at the device in this state. PMU block, along with associated I/Os, are active; the FPGA fabric has been power gated and inactive; the configuration (or programming) of the device along with EBR contents are preserved; and the I/Os are holding their state.

We can wake the device up in a number of ways. One simple way is by using the External User Wake-Up (USRWKUPN). This option should be enabled in the module generated through Diamond's Clarity Designer. USRWKUPN is an active low signal, users toggle it low and device wakes up (adherence to timing requirements to wake up the device is required). USRWKUPN is a dedicated shared pin, and if required by the application, must be selected in the Clarity Designer GUI.

Another way to wake up the device is through Watch Dog Timer (WDT) expiry. The Watch Dog Timer is set as per application requirement – when the device needs to wake up and upon expiry of the timer, WDT sends a trigger to the Power Control Unit to wake up. In the Diamond Clarity Designer GUI, users can place some initial timers in the GUI (see the [Using Diamond Clarity Designer GUI to Generate PMU](#) section on page 19). If left blank, these are set to 0 seconds. The values of the timers can be updated dynamically; this has to be taken care of through the application.

The device wake up can also be initiated internally through Always-On power domain blocks – PSEQ Control Block and via user I²C.

The device can always be reconfigured, even in Sleep State, but this will wake it up. This is **not** a recommended method to wake up, because all the changes made in the device will be lost (EBR contents and PMUCR settings could be updated during operation of the device). It reconfigures the device to its very first original bitstream and initializes everything. Although not recommended, it can be used to wake up the device.

Sleep State to Normal State transition has multiple triggers; however, there is no priority when it comes to these triggers. All triggers cause the same transition, and hence are dealt on a first come first serve basis.

2.1.3. Types of Triggers

There are a number of triggers that cause the PMU to change the device from Normal State to Sleep State, and vice-versa. These triggers can be categorized into two main types based on how the trigger is originated.

2.1.3.1. Internal Triggers

These are triggers generated from the logic within the fabric.

Looking at the transition from Sleep State to the Normal State, it is seen that Watch Dog Timer expiry is an internal trigger.

The trigger that changes the device state from Normal to Sleep can be internal if generated within the FPGA fabric. This also contains decoded commands in internal Always-On logic that are sent from user I²C port from an external signal.

2.1.3.2. External Triggers

These are triggers generated from external pin on General Purpose I/O (GPIO) pins.

For example, the Application Processor may require the device to go to Sleep. This signal, when directly routed to the SLEEP port of the PMU, can be considered an External Trigger.

2.1.3.3. Recommendations

It is recommended that all triggers be internal, including ones changing from Normal State → Sleep State. This is further discussed in the [Using the PMU](#) section on page 9. Even if the request to place the device to Sleep State is coming off-chip, it is recommended that the request be processed and housekeeping taken care of, before generating a SLEEP signal.

The triggers causing the device to wake up from Sleep State → Normal State can be either internal (WDT expiry) or external (user I²C/ External). The device cannot be woken up via FPGA fabric because the FPGA fabric is powered down.

2.1.4. Using the PMU

The previous sections have touched on a few recommendations for changing states of the device. This section covers typical (and recommended) use case for the PMU. It is not mandatory to strictly adhere to this use case, but is provided as a guideline and best practices model.

It is essential to include the PMU IP in the FPGA design if it is to be able to place the CrossLink device into lower power Sleep State. Diamond Clarity Designer tool can be used to configure the PMU (see the [Using Diamond Clarity Designer GUI to Generate PMU](#) section on page 19). When instantiated in the design, PMU is always On, and uses the low speed clock from internal oscillator of the device to perform its operations.

2.1.4.1. Recommended Use Case

A recommended use case scenario is controlling the PMU through a user implemented state machine. This state machine is unique to each application and controls the interface to the PMU through the FPGA fabric interface (SLEEP and 8-Bit Addressable PMUCR interface).

An external event or request to go to sleep can come to this state machine. The state machine then processes the request, manages the operation(s) that the CrossLink device is handling, performs housekeeping of these operation(s), updates the PMUCR (if needed) and then initiates the SLEEP signal.

The PMU IP available to the user does not implement any fixed state machines, because the requirements of each application is unique. PMU IP is a more open and simple interface that can be interfaced with the user state machine that will process the state transitions request. Any dynamic updates, as needed, for the PMU configuration can be done through 8-bit addressable PMUCR (see [PMU Control Register \(PMUCR\)](#) section on page 15). For example, the application processor can request the device to go to sleep for 2 hours. If the 2 hour time is different from what was initially programmed in the WDT, the state machine will have to update the WDT with the new 2 hour time and then initiate SLEEP.

PMUCR can be used to enable and set Watch Dog Timer, and to enable and disable various triggers. This interface is referred to as System Control Bus Interface (SCI Interface), and is accessed via the fabric using CIB interface signals. Usage of SCI interface is discussed in the [PMU Control Bus and Primitive Attributes](#) section on page 13.

Lastly, based on PMU configuration, the device can be woken up using I²C interface, or using PMUWKUP or USRWKUPN pins.

2.1.4.2. Using SLEEP Signal

Lattice recommends that SLEEP signal in CrossLink devices be used under the recommended settings. When not used correctly, the device can stay in permanent Sleep State and not wake up.

Here are a couple of scenarios when SLEEP is directly routed to the PMU.

Routing SLEEP signal to USRWKUPN Pin

In CrossLink devices, the USRWKUPN pin is a dedicated dual purpose shared pin.

One of the functions of the pin is to be used as User Wake Up port. This function on the pin is hardwired and directly connected to the Power Management Unit (PMU). The other function of this pin is as a General Purpose IO. To use this pin as User Wake Up port, persistence on this pin must be turned On, otherwise it acts as a General Purpose Input/Output (GPIO) pin. Turning On persistence on this pin makes it available as GPIO during Normal State, and as User Wake Up pin during Sleep State.

Take an example when we need to just place the CrossLink device to sleep. The device is operating in the Normal State; and the SLEEP signal is directly routed to USRWKUPN pin.

The pin can then be used to place the device to Sleep State by toggling the signal high. When the device is in Sleep State, this pin resumes its wake up function (USRWKUPN) and can be used to wake up the device. USRWKUPN is active low and toggling this pin low initiates the trigger for the device to wake up.

So in Normal State, the pin acts as trigger to change state from Normal to Sleep, and when in Sleep State, it can be used to transition from Sleep to Normal.

[Figure 2.3](#) shows a functional depiction of how the change in signal puts the device to sleep and wake up when routed to the same pin (USRWKUPN).

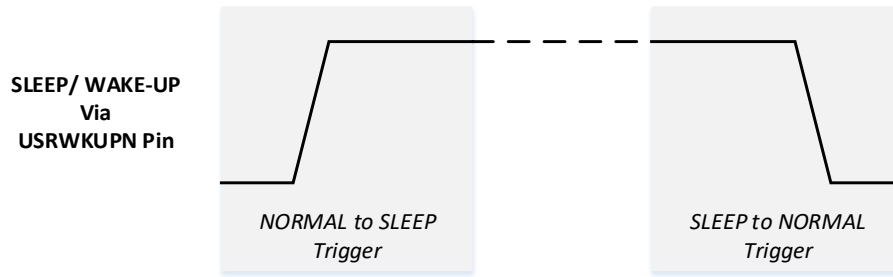


Figure 2.3. Using USRWKUPN Pin for sleep and wakeup (functional depiction, not timing)

Routing SLEEP signal to GPIO

Another use case scenario is when SLEEP signal is routed to a General Purpose Input/Output (or GPIO) for the CrossLink device.

Take an example where the SLEEP signal needs to be directly routed to PMU. Routing SLEEP signal this way should always be avoided under all conditions, because it will not wake up the device once placed into Sleep State.

Let us look at what effects directly routing SLEEP signal has and a recommended workaround.

The device is operating in the Normal State; and the SLEEP signal is coming in to the PMU via GPIO routed through FPGA fabric. SLEEP signal is going high, and the PMU places the device in Sleep State.

Now, the device needs to be woken up (via any trigger). The fabric comes alive and GPIOs would be holding value before going to sleep. The signal drives high, routed through FPGA fabric, and to PMU that triggers the device to go to sleep. The device gets stuck in this permanent Sleep State when the SLEEP signal is using GPIO.

Lattice recommends that users follow the recommended method of using application specific state machine (discussed earlier in this section). If you need to use a GPIO for the SLEEP signal that needs to be directly fed into the PMU, then register this signal which is reset by a Global Set-Reset (GSR). By employing a GSR reset register, the PMU will never see the high when waking up the device and this will avoid glitches or placing the device in Sleep State inadvertently. [Figure 2.4](#) shows the block diagram for the same.

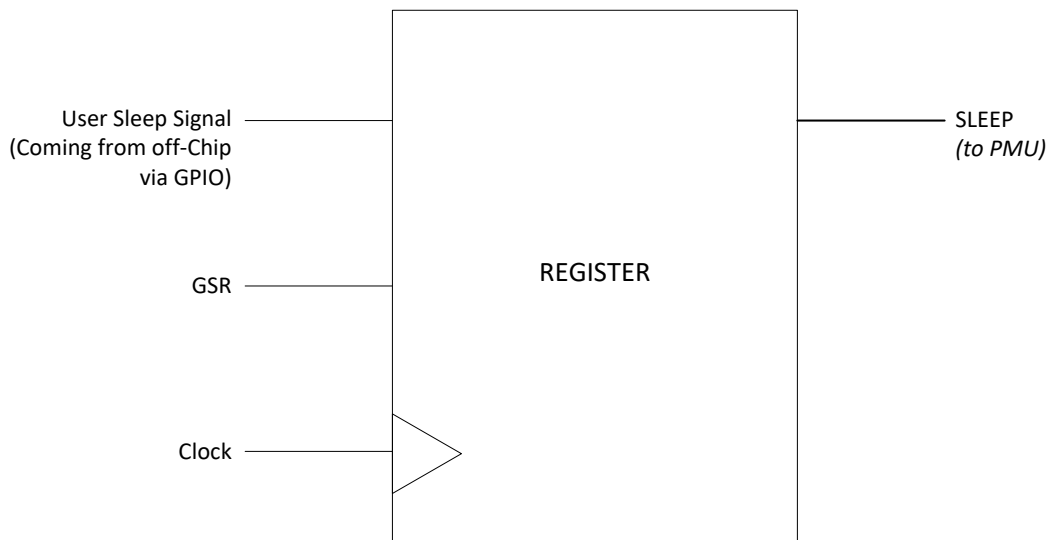


Figure 2.4. Using GPIO for SLEEP when directly routed to PMU (functional depiction, not timing)

CrossLink devices retain configuration SRAM, EBR contents when in Sleep state. All the registers within the IPs (for example EBR registers), and PFU registers do not retain their state during the sleep mode. User I²C registers retain their state.

2.1.4.3. Global Clocks, Memories and PMU

Most FPGA designs use memories. The CrossLink device has built in Embedded Block RAM (EBR) blocks that are complemented by the LUT-based Distributed Memories.

When creating designs that employ EBRs or Distributed RAMs, use global clocks. Use of non-global clocks can potentially cause issues with memory contents retention/ corruption issues when changing states. When using non-global clocks, all read or write operations must be completed (and stopped) before entering the Sleep mode.

The guidelines under [Recommended Use Case](#) section on page 9 should be strictly followed when using non-global clocks in the designs. User state machine should take care of housekeeping when going into Sleep State. All activities related to memory contents must be stopped before the device goes into Sleep State to avoid potential memory corruption issues.

Diamond software identifies this situation and generates an appropriate warning for the user.

2.1.4.4. State of the I/Os

CrossLink devices hold the state of the I/Os during Sleep State. The last value of each I/Os is held to avoid any issues with other devices connected downstream from the CrossLink devices. The Differential I/Os do not hold their value and are all tri-stated.

When the CrossLink device is coming out of Sleep State (that is device is waking up), the fabric becomes active and it is reset. As the fabric comes out of reset, it starts to drive some of the I/Os; the values it drives to can potentially conflict with the last frozen state of the I/Os.

This however does not affect how the CrossLink device operates. To avoid potential conflict between freeze I/Os and what the fabric will drive coming out of the Sleep State, we recommend some housekeeping before placing the device to sleep; like wrapping up the operations being performed, and ensuring the I/Os states before going to sleep is what they will still be after the device wakes up.

2.1.4.5. Synthesis Attribute for PMU Primitive

Users can enable the PMU in their code. Watch Dog Timer user mode in Normal state is optional for PMU.

To avoid optimization of PMU when the user mode for Watch Dog Timer is not enabled, the wrapper should include the `syn_noprune` attribute for the instance. This attribute prevents instance optimization for black-box module with unused output ports.

Here is an example of the syntax:

```
/* synthesis syn_noprune=1 */
```

2.2. PMU Module, Primitive and Attributes

The HDL module gets generated for the PMU and serves as PMU Primitive. It can be used in user design via module generated by Clarity Designer, or as direct PMU Primitive instantiation. The block level diagram for HDL module is shown in [Figure 2.5](#).

Certain inputs and outputs are optional based on the application/ design requirements. Clarity Designer provides these options and allows configurations needed for each unique application.

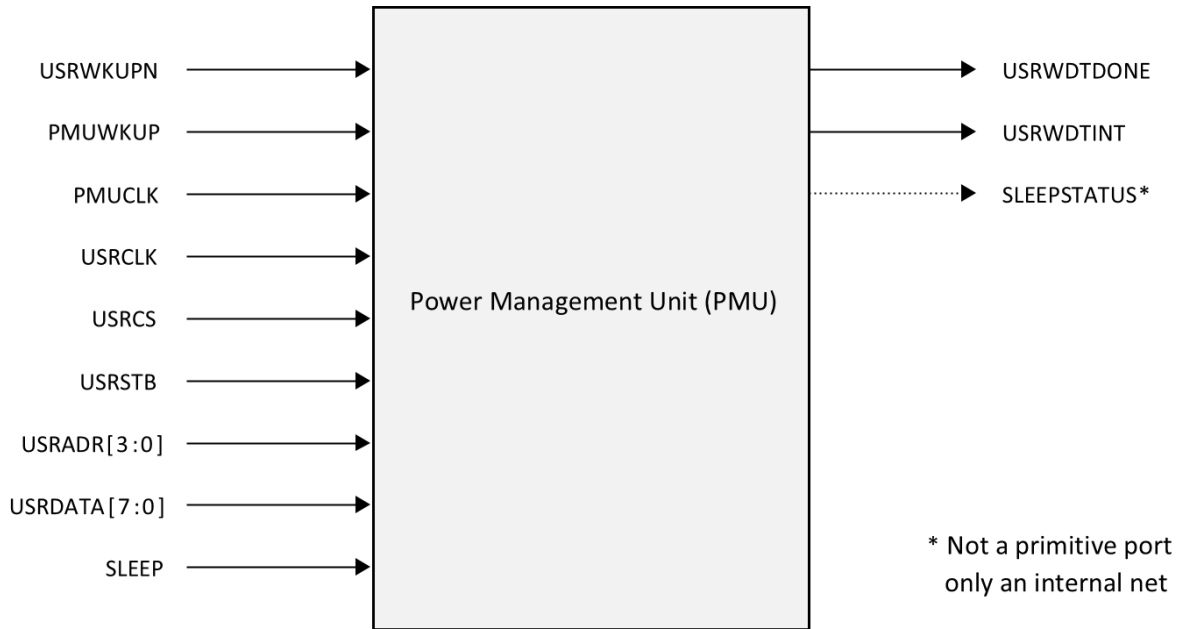


Figure 2.5. PMU HDL Primitive –PMUA

2.2.1. PMU Port Definitions and Parameters

Table 2.1 on the next page describes the ports available in the PMU, along with their functions, and default values.

When the PMU is not used in a design, it has certain default settings that can be different from the settings when it is used in a design. Table 2.1 provides values for both cases. The FPGA fabric interface that comprises of PMU Control Register and the SLEEP signal is tied low when PMU is not in use. When used, this interface is available to update the PMU settings as needed.

Watch Dog Timer (WDT) User Mode makes the WDT available for use in Normal State. During Sleep State, WDT is available for use by PMU and not available for the application.

Table 2.1. PMU HDL Primitive Port Definitions

Port Name	Description	Width	Direction	Signal Source	Port Default	Optional Port Requirements	Default GUI Options	Note
USRWKUPN	User Provided Wake Up Signal	1	Input	External Pin	1 – Disabled	0 – Enabled 1 – Disabled	1 – Disabled	1, 3
PMUWKUP	PMU Wake Up Signal coming from Configuration Logic	1	Input	Configuration Logic/ User Pin	0 – Disabled	1 – Enabled 0 – Disabled	1 – Enabled	2, 3
PMUCLK	PMU Clock coming in from Internal Oscillator	1	Input	Internal Oscillator	0	—	—	—
USRCLK	PMU Control Register User Interface – Clock, Chip Select, Standby, Address and Data	1	Input	From FPGA Fabric	0	—	—	—
USRCS		1	Input	From FPGA Fabric	0	—	—	—
USRSTB		1	Input	From FPGA Fabric	0	—	—	—
USRADR[3:0]		4	Input	From FPGA Fabric	0	—	—	—
USRDATA[7:0]		8	Input	From FPGA Fabric	0	—	—	—
SLEEP	Sleep Signal	1	Input	From FPGA Fabric	0	—	—	—
USRWDTDONE	Watch Dog Timer User Mode Done Signal	1	Output	To FPGA Fabric	—	1 – Enabled 0 – Disabled	0 – Disabled	3
USRWDTINT	Watch Dog Timer User Mode Interrupt Signal	1	Output	To FPGA Fabric	—	1 – Enabled 0 – Disabled	0 – Disabled	3

Notes:

1. If the user does not select the User Wakeup option in the interface (see the [Using Diamond Clarity Designer GUI to Generate PMU](#) section on page 19), this port is set to DISABLED (or 1).
2. PMUWKUP is connected to I2CO (of the two) user I²C which is hardwired to the PMU block.
3. The ports available in the wrapper change according to the selections made by users in the interface section. Some of these options are as follows:
 - a. USRWKUPN (active low signal) is disabled by default. Selections in the interface change it as follows:
 - When Enabled (0), the port is available in the wrapper.
 - When Disabled, the port is not brought out as user port, and is connected to '1' inside the wrapper.
 - b. PMUWKUP is the user I²C wake up. This port is available in the wrapper when user enables the User I²C Wake Up
 - When Enabled (1), the port is available in the wrapper.
 - When Disabled, the port is not brought out as user port, and is connected to '0' inside the wrapper.
 - c. USRWDTDONE is the user mode for WDT in Normal mode.
 - When Enabled (1), the port is available in the wrapper.
 - When Disabled, the port is not brought out as user port (and can be left unconnected).
 - d. USRWDTINT is the trigger for the user mode for WDT in Normal mode.
 - When Enabled (1), the port is available in the wrapper.
 - When Disabled, the port is not brought out as user port (and can be left unconnected).

2.2.2. PMU Control Bus and Primitive Attributes

PMU Control Bus is an 8-bit addressable fabric interface through which the application can access various registers and settings in the PMU. This interface can be used to update the PMU Control Register (PMUCR), (if needed) before placing the device in the Sleep mode. Updating the PMUCR provides an ability to change these settings. If the user does not update the PMUCR before going to Sleep State, then the original settings used during configuration of the PMU are kept. If the PMUCR is updated, then new settings take effect, and the device operation is updated accordingly. Changes made to these settings when device is operational are effective as long as the device is not rebooted. If the device is rebooted, the initial settings are programmed in the registers.

Additionally, the PMU Control Bus interface can be used to update PMU Watch Dog Control Register 1 (PMUWDTCR1). If the Watch Dog Timers (WDT) is being used in the design, users are required to preset the three 32-bit WDT Count values for the PMU Watch Dog Counters (PMUWDCNT1, PMUWDCNT2, and PMUWDCNT3). The details of each of these registers are discussed in the sections that follow.

One of the features of the PMU is that the settings and the attributes can be changed dynamically through the 8-bit addressable PMU Control Bus interface to the block, based on the application requirement; for example, an alarm or timer settings that can be set for different times. Understanding of these attributes is very essential if the user wishes to be able to change these settings.

Table 2.2 provides the list of addresses and their corresponding usage through the PMU Control Bus interface. Each address corresponds to the 8-bit data that can be written on to those registers. Few addresses are not available in the CrossLink device family; these are indicated as N/A (or Not Available).

Table 2.2. PMU Control Bus Address Mapping

Address USRADR[3:0]	Name	Data USRDATA[7:0]	Usage	Access
0000	N/A	—	Not Available	—
0001	PMUCR	8-bit	PMU Control Register	Write
0010	N/A	—	Not Available	—
0011	PMUWDTCR1	8-bit	PMU Watch Dog Timer Control Register1	Write
0100	PMUWDCNT1_3	8-bit	PMU WDT Count Register1_3	Write
0101	PMUWDCNT1_2	8-bit	PMU WDT Count Register1_2	Write
0110	PMUWDCNT1_1	8-bit	PMU WDT Count Register1_1	Write
0111	PMUWDCNT1_0	8-bit	PMU WDT Count Register1_0	Write
1000	PMUWDCNT2_3	8-bit	PMU WDT Count Register2_3	Write
1001	PMUWDCNT2_2	8-bit	PMU WDT Count Register2_2	Write
1010	PMUWDCNT2_1	8-bit	PMU WDT Count Register2_1	Write
1011	PMUWDCNT2_0	8-bit	PMU WDT Count Register2_0	Write
1100	PMUWDCNT3_3	8-bit	PMU WDT Count Register3_3	Write
1101	PMUWDCNT3_2	8-bit	PMU WDT Count Register3_2	Write
1110	PMUWDCNT3_1	8-bit	PMU WDT Count Register3_1	Write
1111	PMUWDCNT3_0	8-bit	PMU WDT Count Register3_0	Write

Typical PMU Control Bus write operation is demonstrated in Figure 2.6.

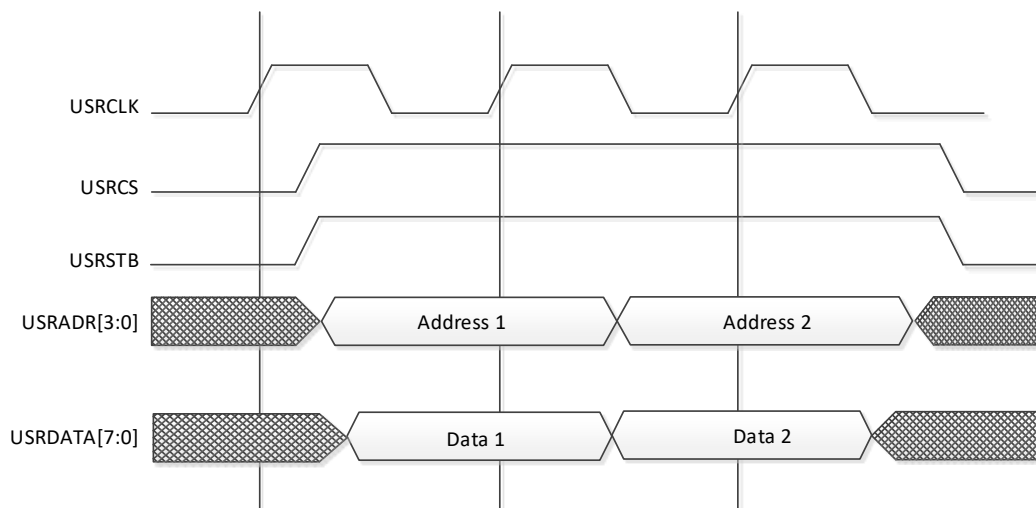


Figure 2.6. PMU Control Bus Write Operation

The PMU Control Bus does not support read operation, since PMU IP does not require data read back functionality.

2.2.2.1. PMU Control Register (PMUCR)

The first address (0001) through the PMU Control Bus interface corresponds to the PMU Control Register (PMUCR). This interface can be used to update the PMUCR, (if needed) before placing the device in the Sleep mode. If the user does not update the PMUCR before going to sleep, then the original settings from Diamond Clarity Designer configuration for the PMU are used. If the PMUCR is updated, then new settings takes into effect, and the device operation is updated accordingly.

PMU Control Register (PMUCR) is an 8-bit user accessible register that can be used to update the settings like how PMU wakes up the device; through Watch Dog Timer, or external wake up signal and so on.

During the PMU configuration through Diamond Clarity Designer, the PMUCR will be set to certain value based on the selected GUI options. These will become the initial settings for the PMU (and PMUCR), and these can be easily changed through the 8-bit addressable PMU Control Bus interface to the PMU.

[Table 2.3](#) shows what each of the bits of 8-bit PMUCR stands for. For details on each attribute check the detailed attribute in [Table 2.4](#).

Table 2.3. PMU Control Register (PMUCR) bit map

Bit #	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Attribute	PMUEN	UWDTEN	UWDTMD	UWDTINT	EXTEN	N/A	I2CEN	WDTEN

[Table 2.4](#) shows the detailed selections of the all the bits along with the various values. PMUCR[7] bit is set to 1 when PMU is instantiated in the design; if PMU is not used, this bit is set to 0 and PMU is disabled.

Table 2.4. PMU Control Register (PMUCR) Component Port Definition

GUI Selection	Attribute	PMUCR Bit	Description	Core	GUI Option	Value Range	Default Value	Notes
—	PMUEN	PMUCR[7]	PMU Enable	—	—	1– Enabled 0 – Disabled	—	1, 2
Enable User Mode for Watch Dog Timer in <i>Normal State</i>	UWDTEN	PMUCR[6]	User WDT Enable	Core/ Sim	Check Box	1– Enabled 0 – Disabled	0 – Disabled	3
Mode Selection	UWDTMD	PMUCR[5]	User WDT Mode Selection	Core/ Sim	Drop Down Menu	COUNT_ONCE , COUNT_REPEAT	COUNT_ONCE	3
Interrupt Enable	UWDTINT	PMUCR[4]	User WDT Interrupt Enable	Core/ Sim	Check Box	1– Enabled 0 – Disabled	1– Enabled	—
External User Wake Up	EXTEN	PMUCR[3]	External Pin Interrupt Enable	Core/ Sim	Check Box	1– Enabled 0 – Disabled	0 – Disabled	—
—	Not Available	PMUCR[2]	—	—	—	—	0 – Disabled	—
User I2C Wake Up	I2CEN	PMUCR[1]	Slave I2C Interrupt Enable	Core/ Sim	Check Box	1– Enabled 0 – Disabled	1– Enabled	—
Watch Dog Timer (WDT) Expiry Wake Up	WDTEN	PMUCR[0]	Watch Dog Timer Interrupt Enable	Core/ Sim	Check Box	1– Enabled 0 – Disabled	1– Enabled	—

Notes:

1. PMUCR [7] is automatically set to ENABLED (1) when PMU is included in the design. If the PMU is not instantiated, then the PMUCR [7] is DISABLED (set to 0). In the disabled state, none of the features or interfaces of the PMU are available.
2. When the user enables the User mode for the built-In WDT in Normal Mode, then UWDTEN is set to ENABLED. The other GUI options should get active based on this check box, that is when the User mode for the WDT is ENABLED, and then the Mode Selection and Interrupt Enable Bit option will be available for users to set. UWDTEN and WDTEN are applicable in mutually exclusive states (Normal and Sleep), and they apply to the built-in Watch Dog Timer.
3. In Normal mode the Watch Dog Timer is available to access for user logic via PMU Control Bus by enabling the UWDTEN option. In sleep mode, the user logic is inactive and hence the Watch Dog Timer is not used for user logic. When set properly, WDT can be set to wake up the device from Sleep (to *Normal State*), once the WDT count expires.

2.2.2.2. PMU Watch Dog Timer Control Register 1 (PMUWDCR1) Settings

The next available address is 0011, and it corresponds to the PMU Watch Dog Timer Control Register 1 (PMUWDCR1). This register is used for selecting the counter that will be used by the Watch Dog Counter (discussed in the following section) to wake up the device.

Although this is an 8-bit register, only the two lowest bits are available to select between the three counters available.

Table 2.5 shows the bit mapping for PMUWDCR1.

Table 2.5. PMU Watch Dog Timer Control Register 1 (PMUWDTCR1) bit map

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
N/A						Sleep Counter Selection (SLPCNT)	

Settings for the two bits correspond to the following selections:

- 00 = WDT Disabled.
- 01 = Use PMUWDCNT1 for WDT for Sleep Mode if WDT is enabled.
- 10 = Use PMUWDCNT2 for WDT for Sleep Mode if WDT is enabled.
- 11 = Use PMUWDCNT3 for WDT for Sleep Mode if WDT is enabled.

Table 2.6 lists the options available in the GUI for the selection of Watch Dog Timer to use when waking up. The default value (when WDT wake option is selection) is set to Watch Dog Counter # 1.

Table 2.6. PMUWDTCR1 Definitions and Settings

GUI Selection	Attribute	PMUWDTCR1 Bits	Description	Core/Sim	GUI Option	Value Range	Default Value
Select the Watch Dog Timer (WDT) Counter that wakes up the device	SLPCNT	1:0	Sleep Mode WDT Count Select	Core/Sim	Drop Down Menu	COUNTER_1, COUNTER_2, COUNTER_3	COUNTER_1

As an example, if the data 00110010 is written to the PMUWDTCR1, then the six MSB are ignored (that is 001100) and only the two LSBs are considered (10); these indicate to the PMU that it needs to wake up using the expiry of Watch Dog Counter #2 (PMUWDCNT2), after Sleep phase initiation.

2.2.2.3. Watch Dog Timer Counters and their Initialization Options

The next set of addresses from 0100 through 1111 on the PMU Control Bus corresponds to the Watch Dog Timer Counters. These are the counters that are set to a specific value (in terms of time) upon expiry of which the PMU wakes up the device.

There are three Watch Dog Timer Counters and each counter is 32 bit. Four addresses cover the 32-bit space for each counter (each address corresponding to 8-bits). The three counters are PMUWDCNT1, PMUWDCNT2, and PMUWDCNT3.

The PMU WDT Count Registers, total 32 bits, holds the binary number of PMU clock cycle for built-in Watch Dog Timer to achieve desired timing. For normal application, user should set the counter prior to entering power saving mode if it is set to be used.

Each counter should be written sequentially from MSB to LSB; for example PMUWDCNT1_3 to PMUWDCNT1_0 when writing PMUWDCNT1. On receiving interrupts, the PMU takes a clock cycle to activate the programmable sequence logic to perform the enter and exit power saving mode, so the logical PMU_WDT count setting should be always larger than 1.

The mapping of each of the counters is shown in the following tables.

PMUWDCNT1 comprises of four bytes PMUWDCNT1_3, PMUWDCNT1_2, PMUWDCNT1_1, and PMUWDCNT1_0. The bit mapping is as shown in the Table 2.7.

Table 2.7. PMU Watch Dog Timer Counter 1 (PMUWDCNT1) bit map

PMUWDCNT1 [31:0]			
PMUWDCNT1_3 [7:0]	PMUWDCNT1_2 [7:0]	PMUWDCNT1_1 [7:0]	PMUWDCNT1_0 [7:0]

PMUWDCNT2 comprises of four bytes PMUWDCNT2_3, PMUWDCNT2_2, PMUWDCNT2_1, and PMUWDCNT2_0. The bit mapping is as shown in the [Table 2.8](#).

Table 2.8. PMU Watch Dog Timer Counter 2 (PMUWDCNT2) bit map

PMUWDCNT2 [31:0]			
PMUWDCNT2_3 [7:0]	PMUWDCNT2_2 [7:0]	PMUWDCNT2_1 [7:0]	PMUWDCNT2_0 [7:0]

PMUWDCNT3 comprises of four bytes PMUWDCNT3_3, PMUWDCNT3_2, PMUWDCNT3_1, and PMUWDCNT3_0. The bit mapping is as shown in the [Table 2.9](#).

Table 2.9. PMU Watch Dog Timer Counter 3 (PMUWDCNT3) bit map

PMUWDCNT3 [31:0]			
PMUWDCNT3_3 [7:0]	PMUWDCNT3_2 [7:0]	PMUWDCNT3_1 [7:0]	PMUWDCNT3_0 [7:0]

Each of these registers is 32-bit. All values in the 32-bit range are supported and user accessible. [Table 2.10](#) provides the Clarity Designer GUI options.

Table 2.10. PMU Watch Dog Timer Counter 3 (PMUWDCNT3) bit map

GUI Selection	Attribute	PMUWDCNT Bits	Description	Core/Sim	GUI Option	Value Range (hh:mm:ss)	Default Value
WDT Counter 1	PMUWDCNT1 [31:0]	31:0	Preset value for WDT Counter 1	Core/Sim	Text Box	00:00:00 To 08:00:00	00:00:00
WDT Counter 2	PMUWDCNT2[31:0]	31:0	Preset value for WDT Counter 2	Core/Sim	Text Box	00:00:00 To 08:00:00	00:00:00
WDT Counter 3	PMUWDCNT3[31:0]	31:0	Preset value for WDT Counter 3	Core/Sim	Text Box	00:00:00 To 08:00:00	00:00:00

When configuring the PMU module in the Diamond Clarity Designer, users can preset the values of these registers to a set count value they would like their WDT to count to. These values serve as the initial values. If the application desires fixed counter values and does not need to change, set these counters in the Clarity Designer. If no value is provided, then these are all set to zeros (0s).

Clarity Designer require values to be set in terms of time, that is Hours: Minutes: Seconds (HH:MM:SS). At the application level, when accessing the registers through the PMU Control Bus Interface, the users need to provide the binary values corresponding to the desired time. The next section describes the conversion between the two.

The maximum value of the counter can be set to 8 hours.

PMU Watch Dog Timer Count Calculation


PMU runs on the clock from the internal oscillator. The default frequency for the Low Speed Oscillator is 10 kHz. The user inputs to the WDT counters preset value is in Hours: Minutes: Seconds (HH:MM:SS). Clarity designer automatically converts the given value into corresponding 32-bit number. When the application is accessing the PMUWDCNT registers, 32-bit value is expected.

For directly writing to the PMUWDCNT registers, the time needs to be converted to its corresponding 32-bit value. The formula below can be used for settings these values. This is a simple example that shows the calculation.

Frequency of the PMU Clock, f	=	10 kHz
Time Period, t	=	1/f
	=	100 μ sec
User Input (Time)	=	hh:mm:ss
	=	00:30:00
Total Number of Seconds, T	=	hh*3600+mm*60+ss
	=	0*3600+30*60+0
	=	1800 sec.
Number of clock cycles, n	=	T/t
	=	1800 sec / 100 μ sec
	=	18,000,000
PMUWDCNT0	=	BINARY (n)
	=	BINARY (18,000,000)
	=	00000001 00010010 10101000 10000000

2.3. Using Diamond Clarity Designer GUI to Generate PMU

Diamond Clarity Designer can be used to generate an HDL wrapper for the PMU.

1. When a project is created, launch Clarity Designer using **Tools > Clarity Designer** option or by clicking on the  button on the toolbar.
2. Provide the name of the project and select the HDL language of choice. The window appears as [Figure 2.7](#).
3. Select the module you wish to generate. In this case, select **pmu**, under **Architecture_Modules**, as shown in [Figure 2.7](#).

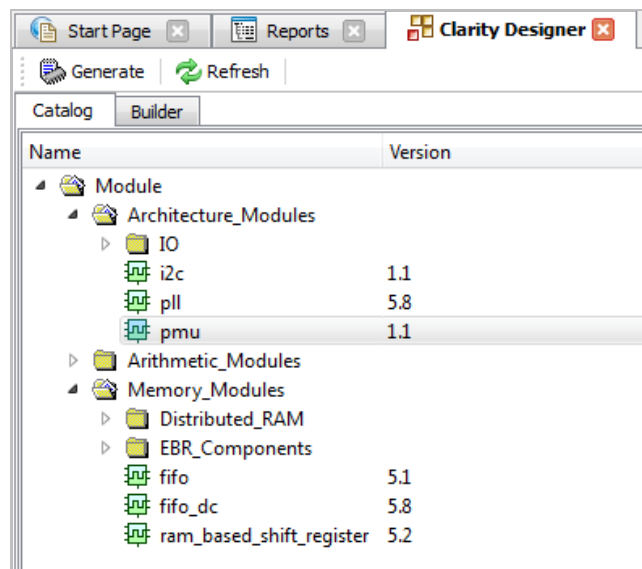


Figure 2.7. Clarity Designer PMU Selection

4. Double click on **pmu** to open another window (Figure 2.8) that requires you to provide an instance name (we use *pmu_test_instance* as our instance name for example),
5. Click **Customize**.

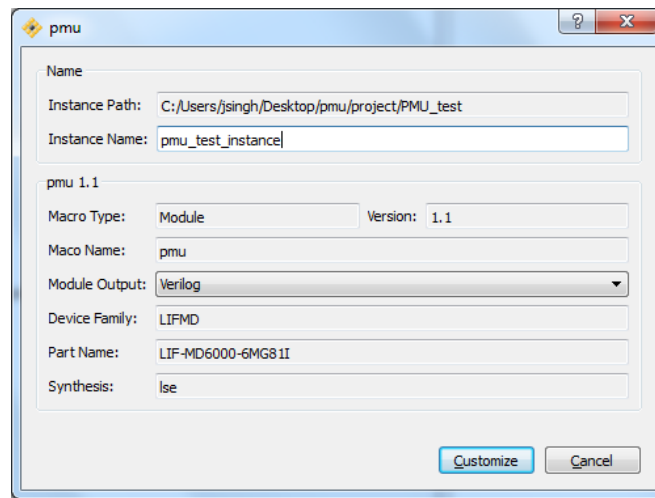


Figure 2.8. PMU Instance Name Options Window

6. The next screen shows the various options to configure the PMU. All these options have been discussed in the earlier sections. Figure 2.9 shows the Configuration GUI.

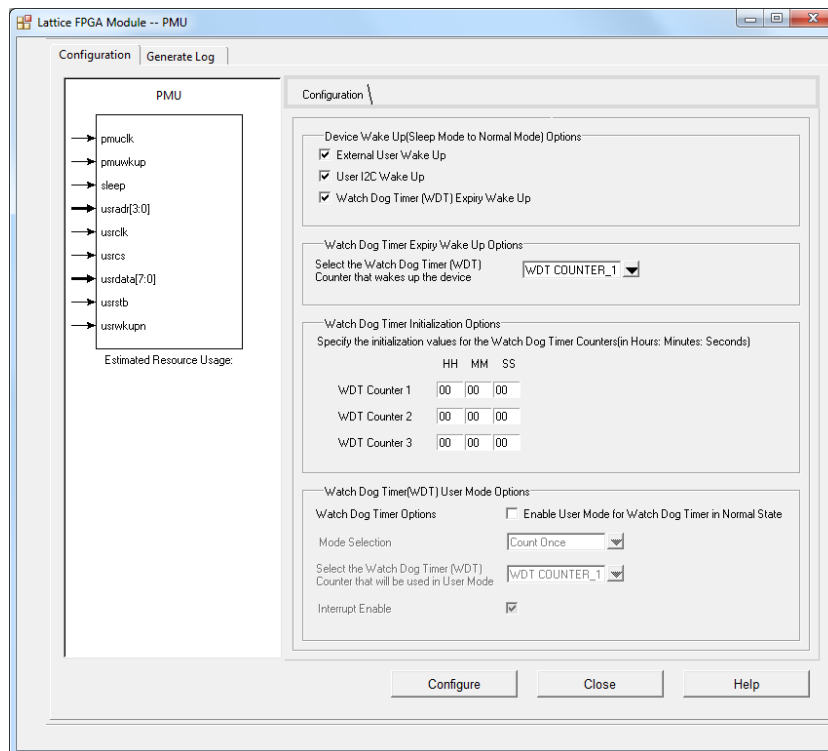


Figure 2.9. PMU Configuration GUI Window

7. When all selections are made, click **Configure** button and then click **Close** button.
8. Click **Generate** button to include the SBX file in the Diamond project file.

If you need to include the HDL file, you can select the option to *add an existing file*, browse to the Clarity Designer project location and add the HDL file.

3. User Standby Mode during Normal State

User Standby Mode for CrossLink devices is useful in reducing power consumption when the device is in operation state, that is Normal State. User Standby Mode is a dynamically controlled option where some blocks can be placed in a User Standby Mode when the block is not required by the application. The reduction in power is achieved by placing these blocks in a low leakage state.

Take care when enabling the User Standby Mode for the blocks. This renders the blocks non-functional. Only the block capable of being placed in User Standby Mode should be placed in the state if it is not used in the application.

This feature is available only in Normal State. Sleep State overrides the Standby Mode, and places the whole device in lower power state.

Table 3.1 provides a list of blocks that have User Standby Mode and the signal required to place it in a Standby Mode.

Table 3.1. Blocks Supporting User Standby Mode

Block	User Standby Control Signal
PLL	Through User Standby (USRSTDBY) Signal
IO	Through LVDS output buffer disable (per bank control) and INR disable (per bank control) (BCINRD and BCLVDSOB signals)
OSC	Through OSC Enable port
MIPI DPHY	Through the USRSTDBY port

3.1. Usage of User Standby Mode

Each block with User Standby Mode option has a signal that can be accessed through FPGA fabric via soft logic. Unlike the Power Management Unit, there is no controller available in the hardware. Application should include a soft controller for accessing User Standby Mode features. This soft controller can place the (unused) blocks in the User Standby Mode, and hence provide the power savings.

Figure 3.1 gives an example block diagram of the User Standby Mode use case. The diagram shows a soft user logic (like a state machine) that can place each block in User Standby Mode – together or selectively.

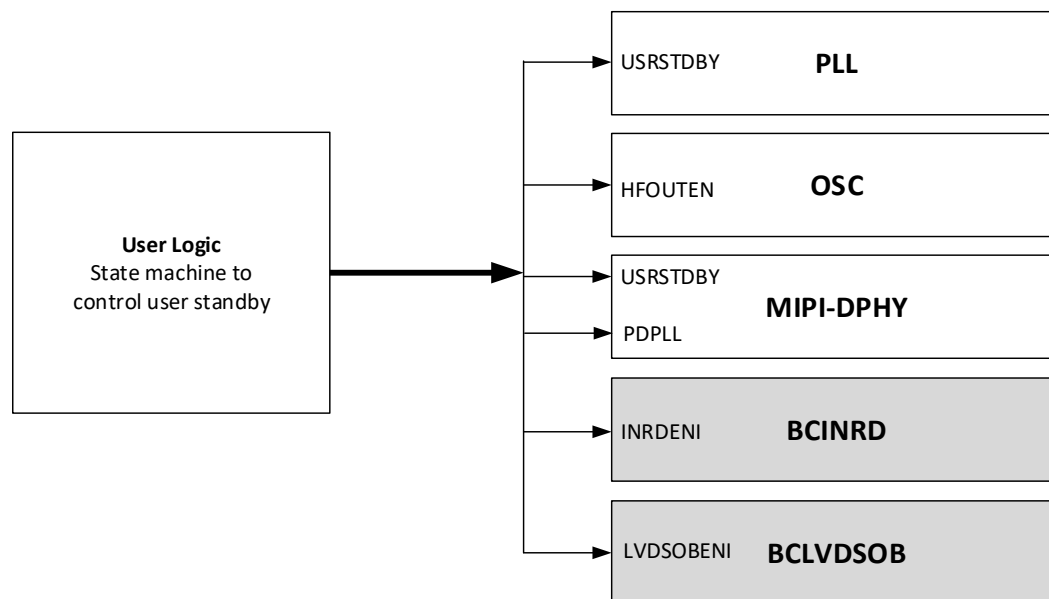


Figure 3.1. User Standby Mode Use Case Example Block Diagram

The next sections discuss how each of the blocks can be placed in User Standby Mode, the signals that can be used, and recommendations.

3.1.1. User Standby Mode for PLL

The CrossLink device PLL contains a User Standby Mode that allows the PLL to be placed into a standby state to save power when not needed in the design. User Standby Mode is very similar to holding the PLL in reset since the Voltage Controlled Oscillator (VCO) is turned off and needs to regain lock when exiting standby. In both cases, reset and standby mode, the PLL retains its programming.

When generating the PLL using Diamond Clarity Designer, the standby option can be enabled by checking **Provide Standby Port** under **Optional Ports** tab in the PLL configuration GUI. This adds the User Standby Port (USRSTDBY) which is an active high signal.

This port can then be connected to the soft controller for dynamic control during periods when PLL is not required.

3.1.2. User Standby Mode for Oscillator

The CrossLink device includes an internal oscillator (OSCI) that generates a high frequency output and a low frequency output. Low frequency output, running 10 kHz is always on output and is used by blocks like Power Management Unit (PMU).

The high frequency output can generate maximum 48 MHz clock, with output divider settings at 1, 2, 4 and 8. The high frequency output clock can be turned off for power savings. The signal that controls the high frequency output clock is High Frequency Output Enable (or HFOUTEN).

HFOUTEN is an active high signal, and when high, turns off the high frequency of the internal oscillator.

3.1.3. User Standby Mode for MIPI DPHY

DPHY block in the CrossLink device can also be placed in a lower User Standby Mode in *Normal State* when it is not required. There are two signals that can place the block in standby – USRSTDBY and PDPLL.

User Standby signal for DPHY (or USRSTDBY) is an active high signal that can power down the DPHY block. USRSTDBY signal is always included in the module generated using Diamond Clarity Designer, during both modes – Transmit and Receive.

Power Down PLL (PDPLL) is also an active high signal and can power down the PLL for the DPHY block. Powering down the PLL resets the PLL, and thus it can be used to reset the DPHY PLL. PDPLL signal is always included in the module generated using Diamond Clarity Designer, during Transmit mode. If the application does not require DPHY PLL block to be powered down, then this signal should be tied low (0).

3.1.4. User Standby Mode for Inputs and Outputs

Referenced, differential and LVDS I/O standards consume more power than other I/O standards and are not always required to be active. Bank Controller allows the designer to turn these I/Os off dynamically on a per-bank selection. The Dynamic INRD (input referenced and differential I/Os) is used to turn off referenced and differential inputs. Dynamic LVDS control is used to turn off the LVDS output driver.

These options are available during *Normal State* only. In order to use these feature, the application must instantiate the primitives that control these bank controllers. Primitives and corresponding simulation behavior are discussed in the sections below. The control signals to enable and disable this feature has to be controlled by user logic.

Each of the bank controllers – INRD and LVDSOB work independent of each other.

3.1.4.1. Bank Controller for Input Reference and Differential

In the CrossLink device, the Dynamic Bank Controller is used to power down banks that have Referenced and Differential inputs. The control is dynamic and if needed can release these inputs, including bidirectional I/Os. The next sections discuss the primitive and simulation of BCINRD.

BCINRD Primitive

BCINRD Dynamic Bank Controller is represented with primitives as shown in Figure 3.2. Each instantiation controls a single bank; so if the control is required for both banks, two BCINRD primitives have to be instantiated, one for each bank (bank 1 and bank 2).

This disables the referenced and differential receivers and any bank controller reference circuits that consume dynamic power, resulting in reduction in power consumption by these circuits. The Inputs Reference and Differential Enable (or INRDENI) signal is an active high signal that can place these inputs in User Standby Mode.

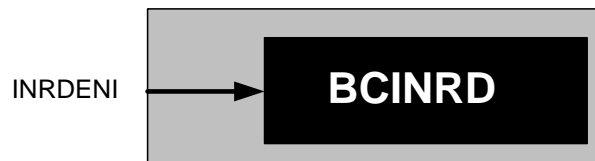


Figure 3.2. Primitive for Bank Controller for Inputs – Referenced and Differential

BCINRD Attributes

Table 3.2 shows the attributes for the BCINRD, with values 1 and 2 representing each bank.

Table 3.2. BCINRD Primitive Attributes

Port	Attribute	Value	Default	Description
INRDENI	—	—	—	Dynamic signal to enable and disable Bank INRD
—	BANKID	1, 2	2	Bank ID for each bank for Dynamic power control INRD

BCINRD Simulation

For application utilizing BCINRD, Diamond only supports post-Place & Route simulation. The functional diagram for the simulation model and simulation behavior for each inputs is as shown in Figure 3.3.

For functional simulation, that is before Place & Route, the design requires users to manually instantiate the simulation primitives as per Figure 3.3 (INRDB) on their inputs and bidirectional buffers that are affected by the bank controller.

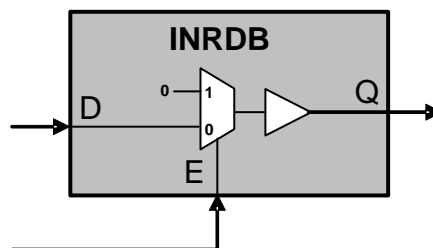


Figure 3.3. INRDB Simulation Primitive

Table 3.3 shows the behavior of the INRDB simulation primitive.

Table 3.3. BCINRD Simulation Behavior

Port	Port Name for Input Buffer	Output
D	Input	—
E	Enable	—
Q	Output	0, when E =1 D, when E=0

3.1.4.2. Bank Controller for LVDS Output Buffers

In the CrossLink device, the Dynamic Bank Controller is used to power down banks that have LVDS Outputs. The control is dynamic and can release these outputs as needed. The primitive and simulation of BCINRD is detailed next.

BCLVDSOB Primitive

BCLVDSOB Dynamic Bank Controller is represented with primitives as shown in Figure 3.4. Each instantiation controls a single bank; so if the control is required for both banks, two BCLVDSOB primitives have to be instantiated, one for each bank (Bank 1 and Bank 2).

This disables the LVDS output receiver circuits that consume dynamic power, resulting in reduction in power consumption by these circuits. The LVDS Output Enable (or LVDSENI) signal is an active high signal that can place these outputs in User Standby Mode.

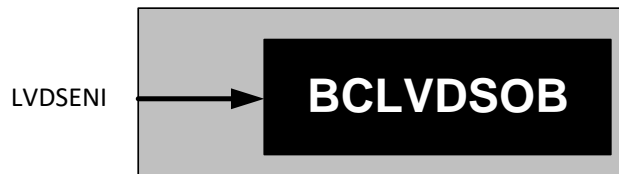


Figure 3.4. Bank Controller for LVDS Outputs Primitive

BCLVDSOB Attributes

Table 3.4 shows the attributes for the BCLVDSOB, with values 1 and 2 representing each bank.

Table 3.4. BCLVDSOB Primitive Attributes

Port	Attribute	Value	Default	Description
LVDSENI	—	—	—	Dynamic signal to enable and disable Bank LVDS Outputs
—	BANKID	1, 2	2	Bank ID for each bank for Dynamic power control LVDS Outputs

BCLVDSOB Simulation

For application utilizing BCLVDSOB, Diamond only supports post-Place & Route simulation. The functional diagram for the simulation model and simulation behavior for each inputs is as shown in Figure 3.5.

For functional simulation, that is before Place & Route, the design requires users to manually instantiate the simulation primitives as per Figure 3.5 (LVDSOB) on their output buffers that are affected by the bank controller.

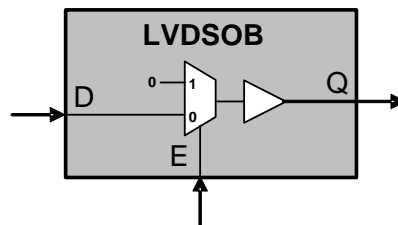


Figure 3.5. LVDSOB Simulation Primitive

Table 3.5 shows the behavior of the LVDSOB simulation primitive.

Table 3.5. BCLVDSOB Simulation Behavior

Port	Port Name for Output Buffer	Output
D	Input	—
E	Enable	—
Q	Output	Z (High Impedance), when E =1; D, when E=0

4. Power Consumption and Calculation

Diamond software includes a Power Calculator tool that can determine the power consumption of the CrossLink devices. Power Calculator is the fastest power simulation tool available in the industry. It offers two modes – *Estimation mode* (for what-if analysis) and *Calculation mode* (for the more accurate application-specific power consumption by importing NCD design files). The engine performs each calculation quickly and accurately.

When running the Power Calculator tool in **Estimation mode**, designers provide estimates of the utilization of various components and the tool provides an estimate of the power consumption. This is a good start, especially for what-if analyses and device selection.

Calculation mode, on the other hand, is a more accurate approach, where the designer imports the actual device utilization by importing the post place and route netlist design file (or NCD) file. Additionally, Power Calculator supports features like Trace Report (or TWR) import, to get the clock frequencies for various clocks. Trace Report only includes frequencies of the clocks nets that are constrained in the Preference file. Users are still required to provide frequencies of the clocks that are not included in Preference file (and hence the Trace Report).

The default Activity Factor (AF%) for dynamic power calculation is set to 10% in the Power Calculator. Users can change the default AF for the entire project or for each clock net individually. Activity Factor is discussed in detail in the [Activity Factor Calculation](#) section on page 28.

4.1. Power Calculator

Refer to the Diamond® Tutorial under Diamond Startup Page. When you go through the procedure, the Lattice Power Calculator main window appears as shown in [Figure 4.1](#)

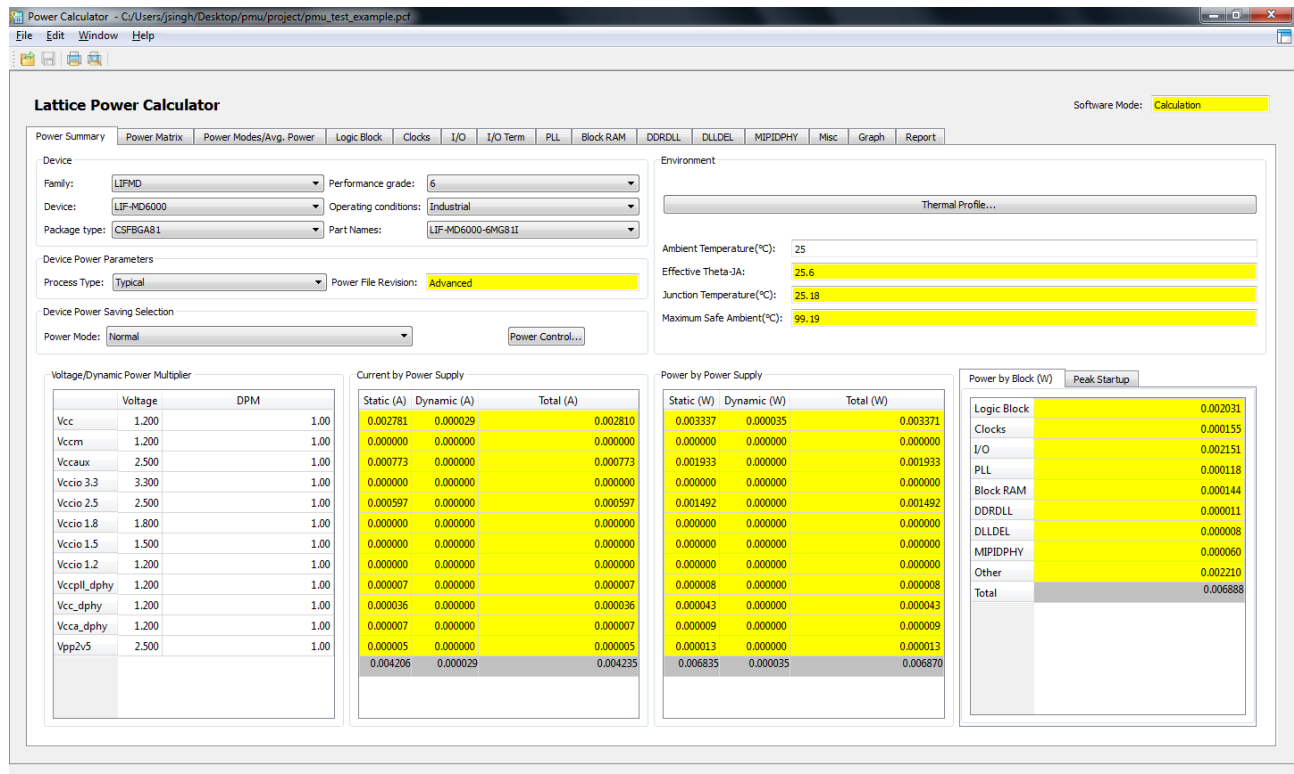


Figure 4.1. Power Calculator (Summary Tab for CrossLink Devices)

It is important to understand how the options available in Power Calculator affect the power. For example, if the ambient temperature is changed, the junction temperature is affected according to the following equation:

$$T_J = T_A + \theta_{JA_EFFECTIVE} * P$$

Where T_J and T_A are the junction and ambient temperatures, respectively, and P is the power.

$\theta_{JA_EFFECTIVE}$ is the effective thermal impedance between the die and its environment.

The junction temperature is directly dependent on the ambient temperature. An increase in T_A increases T_J and result in an increase of the static leakage component.

Power can be affected by selecting the Process Type or the frequency at which the application runs. Process primarily changes the static leakage (or Static Power) and frequency changes dynamic power (increasing the frequency of toggling increases the dynamic component of power).

4.1.1. Typical and Worst Case Process

Process variation is a naturally occurring variation in the attributes of transistors (length, widths, oxide thickness) when integrated circuits are fabricated. Process variation causes measurable and predictable variance in the output performance of all circuits.

Diamond Power Calculator provides option to select the Typical (the mean current/ power of distribution) and Worst Case (the maximum current/ power of distribution) as a result of the variation. Process variation primarily affects the static leakage component of the device.

Process Type selection in the Power Calculator allows users to understand the current and power variation of the devices, and predicts accurate results that are application specific. These can be used for power supply design, battery capacity design and thermal management for each application.

4.1.2. Junction Temperature

Junction temperature is the temperature of the die during operation. It is one of the most important factors that affects the device power. For a fixed junction temperature, voltage and device package combination, static power is fixed.

Ambient temperature affects the junction temperature. Devices operating in a high-temperature environment have higher leakage since their junction temperature is higher. Power Calculator models the interdependence of ambient and junction temperature. When the user provides an ambient temperature, it is rolled into an algorithm that calculates the junction temperature and power through an iterative process to find the thermal equilibrium of the system (device running with the design) with respect to its environment (T_A , airflow and so on).

Thermal impedance plays an important role in determining the devices behavior, and the tool takes it into account to calculate maximum safe ambient temperature. See the [Thermal Management](#) section on page 32 for details on how the Power Calculator uses thermal impedances.

4.1.3. Maximum Safe Ambient Temperature

Maximum Safe Ambient Temperature is one of the most important numbers displayed in the Summary tab of the Power Calculator. This is the maximum ambient temperature at which the device application can run without violating the junction temperature limits for the grade of the device (commercial or industrial).

Power Calculator uses an algorithm to accurately predict this temperature. The algorithm adjusts itself as the user changes options such as voltage, process, frequency, activity factor, thermal impedance and so on (or any factor that may affect the power dissipation of the device).

Thus it becomes extremely important to provide more accurate inputs to the tool, for more accurate predicted results.

4.1.4. Operating Temperature Range

When designing a system, engineers must make sure a device operates at specified temperatures within the system environment. This is particularly important to consider before a system is designed. With Power Calculator, users can predict device thermodynamics and estimate the dynamic power budget. The ability to estimate device's operating temperature prior to board design also allows the designer to better plan for power budgeting and thermal management.

Although total power, ambient temperature, thermal resistance and airflow all contribute to device thermodynamics, the junction temperature as specified in Operating Conditions in the Datasheet is the key to device operation.

The allowed junction temperature range is 0°C to 85°C for commercial grade devices and –40°C to 105°C for industrial grade devices. If the junction temperature of the die is not within these temperature ranges, the performance, reliability and functionality can get affected.

4.1.5. Dynamic Power Multiplier

In general, for semiconductor devices, the dynamic power consumption is independent of the variation in process and temperature. Power Calculator follows this rule, and hence any changes to either process or temperature will not change the dynamic current/ power.

In order to provide users with an option to add some safeguards, Power Calculator includes the Dynamic Power Multiplier (or DPM) column right next to the voltage supplies on the Summary tab. DPM allows users to add a multiplier for the dynamic current for each individual power supplies. This multiplier is included in the dynamic current equation.

Dynamic Power Multiplier has a default value of 1 that means the dynamic power will be what is predicted by the Power Calculator. If the user wishes to add, say 20%, additional dynamic power, the DPM can be set to 1.2 (1 + 20%) and it can be placed against the appropriate power supply. This increases the dynamic power for that supply by 20% and provides users with some guard band.

4.1.6. Peak Startup Current

The bottom right panel of the Summary tab in Diamond Power Calculator includes the Peak Startup tab. This tab provides an important piece of information, especially for designing the capacity for power supply or batteries.

Peak Startup current tab provides the current CrossLink device will pull on each power supply when it is powered on. In certain cases, as applicable, this tab also includes the time period for which this current will last.

4.1.7. Power Budgeting

Both Peak Startup current and device's operational current should be looked into when budgeting for power supply or battery capacity. It is recommended that the higher of the two numbers be used as reference for capacity calculations.

4.1.8. Device Operational Limits

Power Calculator provides the power dissipation of a design under a given set of conditions. It also predicts the junction temperature (T_j) for the design. Any time this junction temperature is outside the limits specified in Data Sheet, the viability of operating the device at this junction temperature must be re-evaluated.

A commercial device is likely to show speed degradation with a junction temperature above 85°C and an industrial device at a junction temperature will degrade above 100°C. It is required that the die temperature be kept below these limits to achieve the guaranteed speed operation.

Operating a device at a higher temperature also means a higher static current (and hence power). The difference between static current and total current (both static and dynamic currents) at a given temperature provides the power budget available. This is also very useful in power supply or battery capacity designs.

If the device runs at a current higher than this budget, the total ICC is also higher. This causes the die temperature to rise above the specified operating conditions. The four factors of power, ambient temperature, thermal resistance and airflow, can also be varied and controlled to reduce the junction temperature of the device. Power Calculator is a powerful tool to help system designers to properly budget the FPGA power that, in turn, helps improve overall system reliability.

Power Calculator clearly indicates when the application is running power higher than recommended. If the Junction temperature goes beyond the grade limits, the box turns red in the summary tab. The junction temperature calculations are provided up to 125°C, which is also the reliability limit of the CrossLink devices. If the junction temperature is beyond 125°C under the conditions provided, the tool indicates it by 125+. This shows that the device achieves thermal equilibrium beyond 125°C.

In certain cases, the thermal equilibrium cannot be achieved by the application under given conditions. In such situation, the device can continue to generate heat and not be able to dissipate. Power Calculator has a built-in algorithm to determine such situations. This is indicated with 125++ under junction temperature calculation. This situation should be avoided under all circumstances, as this can cause permanent damage to the device.

4.1.9. Dynamic Power Savings

The Power Calculator dynamically estimates the power when the Bank Controller and other power save features are implemented in a design. CrossLink device has a number of options to control power; these can range from Bank Controllers to the disabling individual PLLs/ DLLs.

On the Summary tab of Power Calculator, there is a Power Controller button. Clicking the button launches a window to perform what-if analysis to evaluate power consumption when different dynamic power options are used.

See the [Using Power Calculator for CrossLink Devices](#) section on page 29 for details on how to use these features.

4.1.10. Activity Factor Calculation

The Activity Factor % (or AF%) is defined as the percentage of frequency (or time) that a signal is active or toggling the output. Most resources associated with a clock domain are running or toggling at some percentage of the frequency at which the clock is running. Users must provide this value as a percentage under the AF% column in the Power Calculator tool.

Another term for I/Os is the I/O Toggle Rate. The AF% is applicable to the PFU, Routing, and Memory Read Write Ports, and so on. The activity of I/Os is determined by the signals provided by the user (in the case of inputs) or as an output of the design (in the case of outputs). The rates at which the I/Os toggle define their activity. The I/O Toggle Rate or the I/O Toggle Frequency is a better measure of their activity.

The Toggle Rate (or TR) in MHz of the output is defined in the following equation:

$$\text{Toggle Rate (MHz)} = 1/2 * f * \text{AF\%}$$

Users are required to provide the TR (MHz) value for the I/O instead of providing the frequency and AF% for other resources. AF can be calculated for each routing resource, output or PFU. However, this involves long calculations. The general recommendation for a design occupying roughly 30% to 70% of the device is an AF% between 15% and 25%. This is an average value. The accurate value of an AF depends upon clock frequency, stimulus to the design and the final output.

Power Calculator allows users to import a VCD file from their simulation to accurately assess the activity factor of their design (Edit > Open Simulation File). The VCD file is based on Post-P&R simulation and it is an ASCII file generated by the simulator. Please check the simulation tool documentation on how to generate VCD file. The AF calculated from the VCD file is based on how accurate the test-bench or stimulus is for the simulation.

4.1.11. Power Calculator Assumptions

The following are the assumptions made by the Power Calculator.

- The Power Calculator tool uses default ambient temperature of 25°C (room temperature). This default temperature can be changed.
- Users can define the ambient temperature (T_A) for device junction temperature (T_J) calculation based on the power estimation. T_J is calculated from the user-entered T_A and the power calculation of typical room temperature.
- I/O power consumption is based on an output loading of 5 pF. Designers have the ability to change this capacitive loading.
- Users can estimate power dissipation and current for each type of power supply (V_{CC} and V_{CCIO}).
- The nominal V_{CC} is used by default to calculate power consumption. A lower or higher V_{CC} can be chosen from a list of available values.
- θ_{JA} can be changed to better estimate the operating system manually or by entering Airflow in Linear Feet per Minute (LFM) along with a Heat Sink options.
- The default value of the I/O types for CrossLink devices is LVCMOS25, 8 mA.
- The activity factor (AF) is defined as the toggle rate of the registered output. For example, assuming that the input of a flip-flop is changing at every clock cycle, 100% AF of a flip-flop running at 100 MHz, is 50 MHz.

- The default activity factor for logic is 10%.
- Users can import the VCD file from their simulation to get activity factor based on the simulation. It is to be noted that the AF from VCD is as good as the coverage in the simulation.
- Unused I/Os are configured as LVCMOS Inputs with weak internal pull ups. These are generally powered off the VCCIO that is connected to the bank.
- Users have an option to import the Frequency from trace report (TWR) or preference file (LPF).
- The operating junction temperature range for commercial grade devices is 0°C to 85°C, and Industrial is –40°C to 100°C For details, refer to the DC Electrical Characteristics section of FPGA-TN-02007, [CrossLink Family Data Sheet](#).
- For thermal impedance, Power Calculator default value is based on a medium size board (6" x 6", 6 layers), with no heatsink and 200 LFM airflow. This can be changed as needed under Thermal Profile section in the tool.

4.2. Using Power Calculator for CrossLink Devices

The CrossLink device has *Normal State*, along with the User Standby Mode, and the *Sleep State*. Power Calculator can calculate the power consumption of the device in various states and options available to the users. The easiest way is to either import the NCD file (in calculation mode) or providing resource utilization (in estimation mode). Once the tool is setup with resources, provide frequency (or import using TWR file), update activity factor, check the thermal environment of the system and update ambient temperature of operation.

This section covers ways to determine power consumption of various states and modes for the CrossLink devices.

4.2.1. Power Calculator in Normal State, and User Standby Mode

Power Calculator can calculate power in the *Normal State*, and its corresponding User Standby Mode. After importing the NCD, and providing all the inputs, the power consumption of the device is when it is operational. Summary tab provides the static and dynamic, current/ power.

The *Normal State* power can be with or without the User Standby Mode enabled. By default the User Standby Mode is disabled; thus the *Normal State* power consumption when the Power Calculator is first setup is without User Standby Mode power savings. If the application includes User Standby Mode blocks, Diamond Power Calculator has option to toggle these options to switch *Normal State* power consumption with and without User Standby Mode. These options can be toggled using the Power Controller button (right next to the Process Type selection on Summary tab).

[Figure 4.2](#) shows the Power Control window tabs (Inputs/ Outputs and Other).

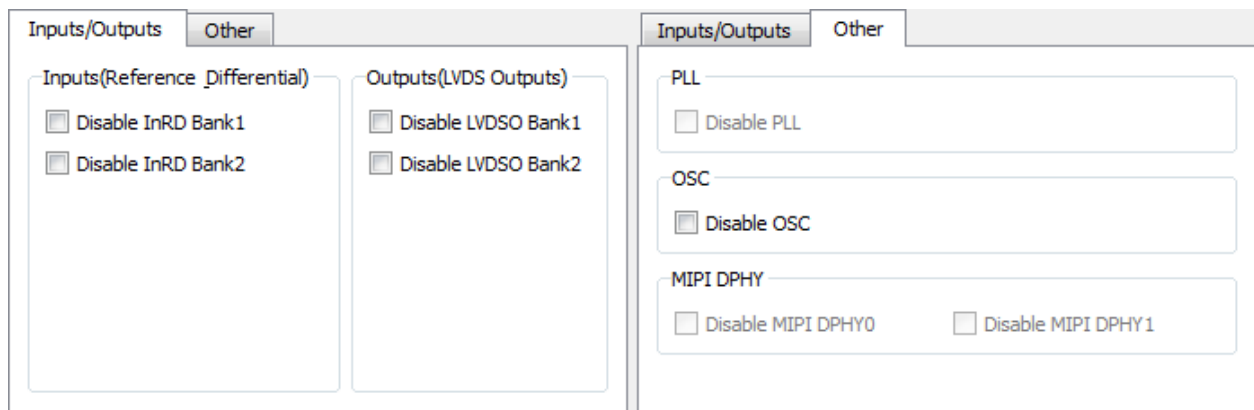


Figure 4.2. Power Controller Window for User Standby Mode options

[Figure 4.2](#) uses an example application that includes Bank Controllers for I/Os, and Oscillator Control. By default, these options are unchecked, that is User Standby Mode is not enabled. By checking the checkboxes, the Power Calculator tool recalculates power for *Normal State* with User Standby Mode enabled.

Power Calculator provides power with or without User Standby Mode mutually exclusively. It does not provide power for both options together. The *Normal State* power used in average power consumption calculations (discussed in the following section) is based on the options with or without User Standby Mode, but never both.

4.2.2. Power Calculator in Sleep State

Power Calculator has a built-in feature that calculates the power consumption of CrossLink device during *Sleep State*. Device in this state is not completely turned off. Parts of the device are still on, such as the Power Management Unit (PMU), the circuitry that retains the SRAM contents (that stores device configuration), EBR and Distributed RAM contents, I/O states, and so on. Hence the *Sleep State* is not a device turned off state, and has some minimal leakage. In order to get the *Sleep State* power consumption, go to Power Modes/ Avg. Power tab. The right hand side pane of the tab indicates the current and power for each supply in the *Sleep State*.

The left hand pane provides the power consumption for the application, that is the same as summarized on the Summary tab. This power matches the power if some of the User Standby features are enabled. If certain power controller options are selected, they become part of the *Normal State* power, and that is what gets populated here.

Figure 4.3 shows the Power by Supply sub-tab of the Power Modes/ Avg. Power tab.

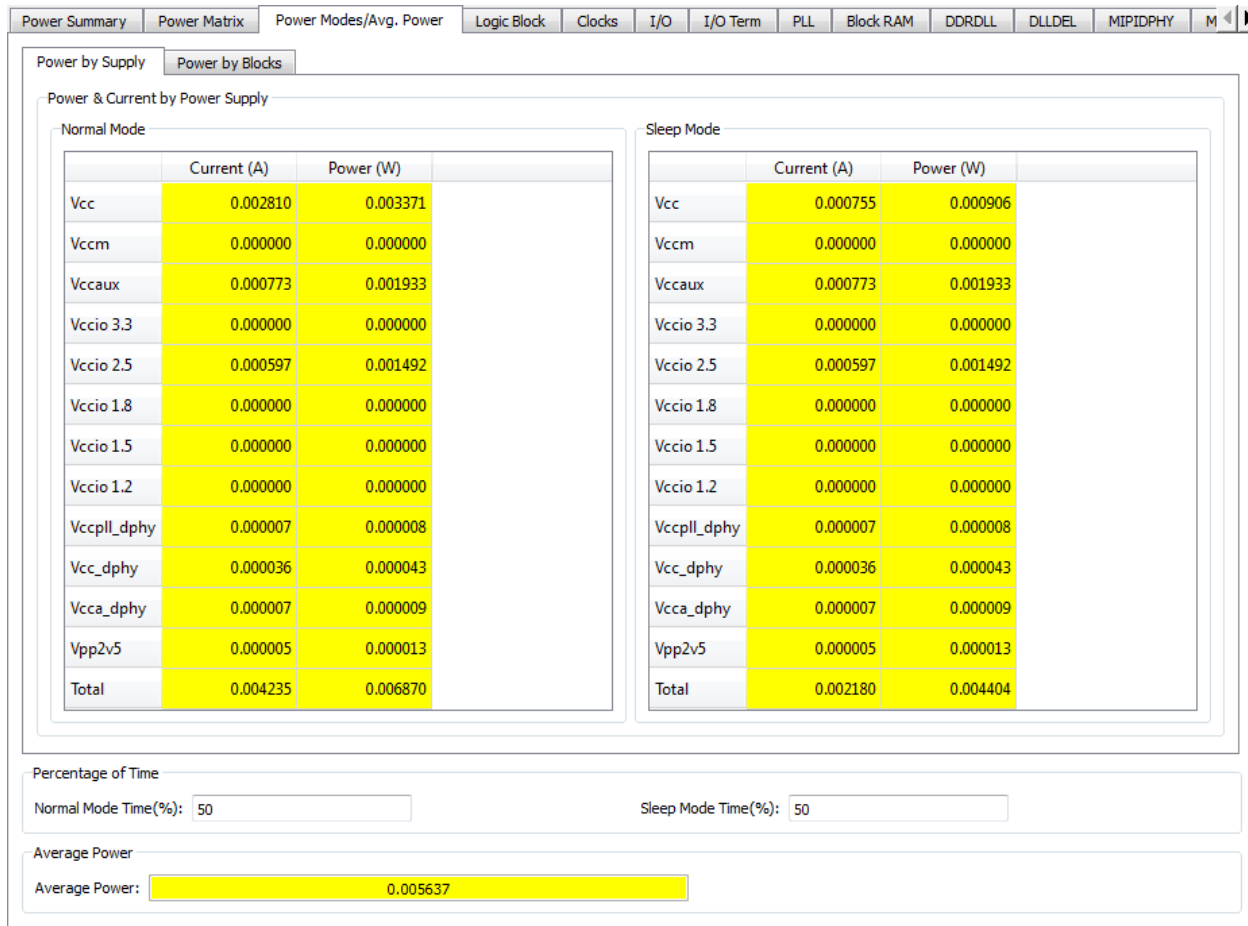


Figure 4.3. Power Calculator – Power Modes/ Avg. Power Tab (Power by Supply)

Figure 4.4 shows the Power by Blocks sub-tab of the Power Modes/ Avg. Power tab.

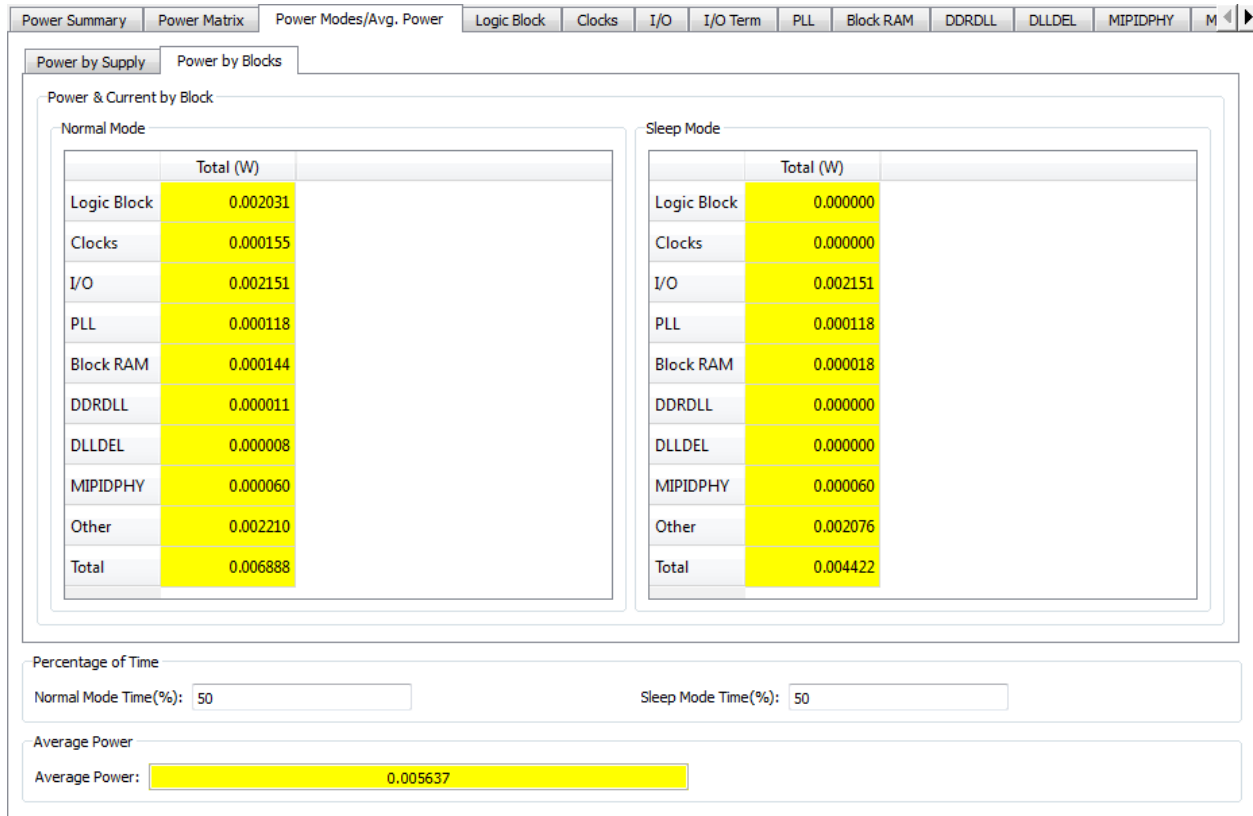


Figure 4.4. Power Calculator – Power Modes/ Avg. Power Tab (Power by Blocks)

4.2.3. Calculating Average Power

Application can use the Power management Unit (PMU) to switch between the *Normal State* and *Sleep State* based on the requirements. When the services of CrossLink device are not needed, the device can operate at a lower power consuming *Sleep State*. Overall, this reduces the power consumption.

Power Modes/ Avg. Power tab also helps calculate the average power consumption of the CrossLink device. The bottom pane of this tab has Percentage of Time and Average Power sections, as shown in Figure 4.3 and Figure 4.4. Average power is calculated based on the percentage of time the device is in each state.

The average power is consumed as per the following formula:

$$P_{AVG} = P_{NORMAL} * \% \text{ of Time in } Normal \text{ State} + P_{SLEEP} * \% \text{ of Time in } Sleep \text{ State}$$

Where, P_{AVG} , P_{NORMAL} , P_{SLEEP} are average power, *Normal State* power, and *Sleep State* power respectively.

For example, if the device is in each state for half the time (50%), then the Average Power consumption is average of *Normal State* and *Sleep State* power. *Normal State* power can be either with or without the User Standby Mode options.

The default value of the time percentages is set to 50%, and it can be changed based on application.

5. Thermal Management

To improve reliability and prevent device failure, all electronic devices are required to dissipate the heat generated while running. Thermal management refers to the techniques used to improve heat dissipation. The methods include use of heatsinks, fans for air cooling and also other forms of cooling like liquid cooling in modern computers.

CrossLink devices are designed to be low power devices, and by combining options like using PMU, the average power consumption and hence the heat generated, can be reduced. This section covers understanding and ways to improve thermal management for applications using CrossLink devices.

5.1. Thermal Impedance and Airflow

A common method for characterizing a packaged device's thermal performance is with Thermal Impedance, represented by θ . For a semiconductor device, thermal resistance indicates the steady state temperature rise of the die junction above a given reference for each watt of power (heat) dissipated at the die surface. Its units are degree Celsius per Watt ($^{\circ}\text{C}/\text{W}$).

The most common examples are:

- θ_{JA} , Thermal Resistance Junction-to-Ambient (in $^{\circ}\text{C}/\text{W}$)
- θ_{JC} , Thermal Resistance Junction-to-Case (in $^{\circ}\text{C}/\text{W}$)
- θ_{JB} , Thermal Resistance Junction-to-Board (in $^{\circ}\text{C}/\text{W}$)

Knowing the reference (that is ambient, case, or board) temperature, the power, P, and the relevant θ value, the junction temperature can be calculated per following equations.

- $T_J = T_A + \theta_{JA} * P$
- $T_J = T_C + \theta_{JC} * P$
- $T_J = T_B + \theta_{JB} * P$

Where T_J , T_A , T_C and T_B are the junction, ambient, case (or package) and board temperatures (in $^{\circ}\text{C}$), respectively. P is the total power dissipation of the device.

θ_{JA} is commonly used with natural and forced convection air-cooled systems. θ_{JC} is useful when the package has a high conductivity case mounted directly to a PCB or heatsink. And θ_{JB} applies when the board temperature adjacent to the package is known.

To improve airflow effectiveness, it is important to maximize the amount of air that flows over the device or the surface area of the heat sink. The airflow around the device can be increased by providing an additional fan or increasing the output of the existing fan. If this is not possible, baffling the airflow to direct it across the device may help. This means the addition of sheet metal or objects to provide the mechanical airflow guides to guide air to the target device. Often the addition of simple baffles can eliminate the need for an extra fan. In addition, the order in which air passes over devices can impact the amount of heat dissipated.

5.2. Thermal Management in Power Calculator

Diamond Power Calculator utilizes the ambient temperature ($^{\circ}\text{C}$) to calculate the junction temperature ($^{\circ}\text{C}$) based on the θ_{JA} for the targeted device. Users can also provide the airflow values (in Linear Feet per Minute or LFM, that is also same as Cubic Feet per Minute) to obtain a more accurate junction temperature value.

Thermal Environment options in the Power Calculators provide a thermal impedances for JEDEC (4"x4", 2S2P), small (6"x6", 6 layered board), medium (8"x8", 8 layered board) and large (10"x10", or larger board). There are also options to select copper heatsinks of different sized fins. These impedances and options are provided as a guidance. Users have an option to provide their own thermal impedance too.

As each application board can be different, it is recommended that accurate thermal impedance be provided. These can be obtained by measuring using a thermal lab or by using thermal simulation software.

5.3. DELPHI Models

DELPHI Models are thermo-mechanical models that can be used to simulate thermal behavior of the electronic devices in a system. These tools can simulate the thermal behavior of the system and predict the thermal impedance for each component. The thermal impedance obtained using simulation methods can be entered in the Thermal Environment in Power Calculator.

DELPHI Models for CrossLink device can be downloaded from the web and they are compatible with Mentor Graphics' FloTHERM® and Ansys Icepak tools.

6. Conclusion

CrossLink devices provide users with a number of options for managing power, and the Power Calculator tool complements by calculating power in different states of the device.

By utilizing the user friendly interface to access these features in the Diamond software, applications can utilize these features to maximum extent, and have predictable and reliable performance for these devices.

References

For more information, refer to the following documents:

- FPGA-DS-02007, [CrossLink Family Data Sheet](#)
- FPGA-TN-02012, [CrossLink High-Speed I/O Interface](#)
- FPGA-TN-02013, [CrossLink Hardware Checklist](#)
- FPGA-TN-02014, [CrossLink Programming and Configuration Usage Guide](#)
- FPGA-TN-02015, [CrossLink sysCLOCK PLL/DLL Design and Usage Guide](#)
- FPGA-TN-02016, [CrossLink sysI/O Usage Guide](#)
- FPGA-TN-02017, [CrossLink Memory Usage Guide](#)
- FPGA-TN-02019, [CrossLink I2C Hardened IP Usage Guide](#)
- FPGA-TN-02020, [Advanced CrossLink I2C Hardened IP Reference Guide](#)

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.1, October 2018

Section	Change Summary
All	Changed document status from preliminary to final.

Revision 1.0, July 2016

Section	Change Summary
All	Updated document numbers.

Revision 1.0, May 2016

Section	Change Summary
All	First preliminary release.



7th Floor, 111 SW 5th Avenue
Portland, OR 97204, USA
T 503.268.8000
www.latticesemi.com