PLL Avalon Interface Core

Core Overview

PLL Avalon Interface Core 用于对 PLL 的动态相移，该 IP Core 提供了一个 Avalon-MM Slave 接口用于跟 Nios 或用户逻辑通讯，和一个 PLL 端口用于跟 PLL 通讯.

Configuration

PLL Avalon Interface Core 没有需要用户配置的参数，但在使用 IP Core 时，请确保您拥有一个打开了动态相移功能的 PLL，并且知晓 PLL 单步相移的精度，以便您能正确的使用该 IP Core 对时钟进行相移.

Related Information

- ALTPLL Megafunction User Guide
- PLLCore, Quartus II 9.1 Handbook, Volume 5

Software Programming Model

Software Files

PLL Avalon Interface Core 提供了以下软件文件，这些文件提供了硬件的底层接口.

- **pll_interface.h**—这个文件提供了访问底层硬件的函数定义.
- **pll_interface.c**—这个文件包含访问底层硬件函数的实现.

Register Map

Table 1-1: Register Map for PLL Avalon Interface Core

Offset	Register Name	R/W	31..30	29..9	8..2	1	0
0	status	R/O	(1)			phasedone	locked
1	control	R/W	(1)			pfdena	areset
2	phase reconfig control	R/W	Phase	(1)		counter_number	
3	—	—	Undefined				

(1) 保留位. 读取保留位时会返回不确定的值，向保留位写时可设置为 0.

Status Register

可以通过访问状态寄存器来获取 PLL 的状态. 向状态寄存器写是无效的.

Table 1-2: Status Register

Bit Number	Bit Name	Value after reset	Description
0	locked	1	连接到 PLL 上的 `locked` 引脚.
1	phasedone	0	连接到 PLL 上的 `phasedone` 引脚.
2:31	—	—	保留. 读取的值未定义

Control Register

可以通过控制寄存器来控制 PLL. 也可以回读控制寄存器.

Table 1-3: Control Register

Bit Number	Bit Name	Value after reset	Description
0	areset	1	连接到 PLL 上的 `areset` 引脚. 向该位写 1 会使 PLL 复位.
1	pfdena	0	连接到 PLL 上的 `pfdena` 引脚. 向该位写 0 会禁用相移频率探测.
2:31	—	—	保留. 读取的值未定义.

Phase Reconfig Control Register

可以通过相位重配置寄存器来动态相移.

Table 1-4: Control Register

Bit Number	Bit Name	Value after reset	Description
0:8	counter_number	—	一个 9 位的二进制数来代表需要位移的时钟. 查询 Table 1-5 来获取详细信息.
9:29	—	—	保留. 读取的值未定义.
30:31	phase	—	01：对指定的时钟正向位移 10：对指定的时钟负向位移 00 和 11：不进行位移

下表列出了 counter_number 和时钟的对应关系. 比如, 设置 100 000 000 来选择 C0, 设置 100 000 001 来选择 C1.

Table 1-5: Control Register

Counter_Number[8:0]	Counter Selection
0 0000 0000	All output counters
0 0000 0001	M counter
> 0 0000 0001	Undefined
1 0000 0000	C0
1 0000 0001	C1
1 0000 0010	C2
…	…
1 0000 1000	C8
1 0000 1001	C9
> 1 0000 1001	Undefined

Software Function Introduction

pll_read_locked ()

Prototype:	`unsigned char pll_read_locked(unsigned int addr)`
Include:	< pll_interface.h>
Description:	addr 为 IP Core 的基地址, 调用该函数读取状态寄存器.
Returns:	PLL 的 locked 值.

pll_read_phasedone ()

Prototype:	`unsigned char pll_read_phasedone(unsigned int addr)`
Include:	< pll_interface.h>
Description:	addr 为 IP Core 的基地址, 调用该函数读取状态寄存器.
Returns:	PLL 的 phasedone 值.

pll_set_areset ()

Prototype:	`void pll_set_areset(unsigned int addr,` `unsigned char areset)`
Include:	< pll_interface.h>
Description:	调用该函数设置 areset 的值.
Returns:	无.

pll_set_pfdena ()

Prototype:	`void pll_set_pfdena(unsigned int addr,` `unsigned char pfdena)`
Include:	< pll_interface.h>
Description:	调用该函数设置 pfdena 的值.
Returns:	无.

pll_phase_up ()

Prototype:	`void pll_phase_up(unsigned int addr,` `unsigned short conter)`
Include:	< pll_interface.h>
Description:	调用该函数对指定的时钟进行一次正向位移.
Returns:	无.

pll_phase_down ()

Prototype:	`void pll_phase_down(unsigned int addr,` `unsigned short conter)`
Include:	< pll_interface.h>
Description:	调用该函数对指定的时钟进行一次负向位移.
Returns:	无.

pll_phase_ups()

Prototype:	`void pll_phase_ups(unsigned int addr,`
	`unsigned short conter,`
	`unsigned short cnt)`

Include:	< pll_interface.h>
Description:	调用该函数对指定的时钟进行多次正向位移. cnt 为位移次数.
Returns:	无.

pll_phase_downs()

Prototype:	`void pll_phase_downs(unsigned int addr,`
	`unsigned short conter,`
	`unsigned short cnt)`

Include:	< pll_interface.h>
Description:	调用该函数对指定的时钟进行多次负向位移. cnt 为位移次数.
Returns:	无.

Document Revision History

Data	Version	Changes
October 2015	1.0	第一次发布