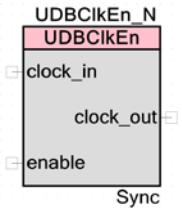
UDBClkEn
1.0

Features



- Clock enable support
- Addition of synchronization on a clock when needed

General Description

The UDBClkEn component supports precise control over clocking behavior.

When to Use a UDBClkEn

The UDBClkEn component can be used to:

- Apply a level-sensitive enable to any clock signal.
- Force the clock signal to be synchronous to BUS_CLK. If the incoming clock is already synchronous, no changes are made. If the incoming clock is asynchronous, logic is inserted to synchronize to BUS_CLK.
- Indicate that a clock signal is allowed to be asynchronous. Several UDB elements (Control Register, Status Register, and Datapath) must normally be clocked with a synchronous clock. PSoC Creator only allows these to be clocked with an asynchronous clock if that clock comes from a UDBClkEn component in Async mode. This feature should only be used after analyzing the potential clock crossing issues associated with communicating with the CPU operating on the BUS_CLK clock domain.

Input/Output Connections

This section describes the various input and output connections for the UDBClkEn component.

enable – Input

Level-sensitive clock enable signal applied to the input clock to form the output clock. If the enable signal is not synchronous to the clock_in signal, a synchronizer is automatically implemented. In addition, if the enable signal is not synchronous to the clock_in signal, then the pulse width of the enable signal must be at least the period of clock_in plus 2ns.

clock_in – Input

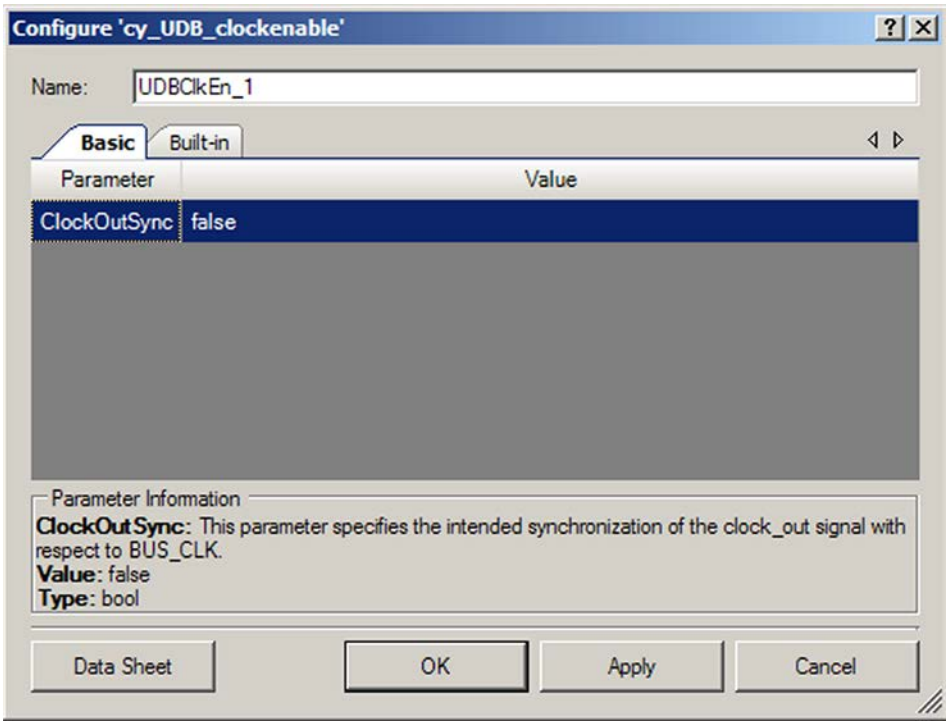
Incoming clock signal.

clock_out – Output

Resulting clock derived from enable and clock_in signals that meet the synchronization setting characteristics. This signal can only be connected to the clock input of a registered element.

Component Parameters

Drag a UDBClkEn component onto your design and double-click it to open the **Configure** dialog/



The UDBClkEn component provides the following parameter.

ClockOutSync

This parameter specifies the intended synchronization of the clock_out signal with respect to BUS_CLK. A setting of **true** forces the clock to be synchronized, if the clock_in signal is not already synchronous to BUS_CLK. A setting of **false** leaves the synchronization unchanged from the synchronization of clock_in.

Resources

Depending on the synchronization of the incoming signals and clock_out, 0 or 1 synchronizer and 0 or 1 macrocell will be used.

Functional Description

Each UDB has four clock-control blocks. These are used for each of the two PLDs, the datapath, and the status/control registers. The clock control block selects the clock signal and, optionally, a clock enable signal. The enable capability of the UDBClkEn component is directly supported by this clock enable signal.

Synchronization of clock_in is required if it is asynchronous from BUS_CLK and the ClockOutSync has been set to true. In this case, the double synchronizer mode of a status register is used to synchronize the clock signal before using the clock as the input to the clock control block.

Component Changes

Version 1.0 is the first release of the UDBClkEn component.

© Cypress Semiconductor Corporation, 2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.