

# PrimeCell® General Purpose Input/Output (PL061) Cycle Model

**Version 9.1.0**

## **User Guide**

**Non-Confidential**



# PrimeCell® General Purpose Input/Output (PL061) Cycle Model

## User Guide

Copyright © 2017 ARM Limited. All rights reserved.

### Release Information

The following changes have been made to this document.

Change History			
Date	Issue	Confidentiality	Change
February 2017	A	Non-Confidential	Restamp release

### Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM Limited (“ARM”). **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version shall prevail.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to ARM’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement specifically covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. You must follow the ARM trademark usage guidelines <http://www.arm.com/about/trademarks/guidelines/index.php>.

Copyright © ARM Limited or its affiliates. All rights reserved.  
ARM Limited. Company 02557590 registered in England.  
110 Fulbourn Road, Cambridge, England CB1 9NJ.

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

**Confidentiality Status**

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

**Product Status**

The information in this document is final, that is for a developed product.

**Web Address**

<http://www.arm.com>



# Contents

## **Chapter 1.** **Using the Cycle Model in SoC Designer**

GPIO PL061 Cycle Model Functionality .....	1
Fully Functional and Accurate Features .....	2
Fully Functional and Approximate Features .....	2
Unsupported Hardware Features .....	2
Adding and Configuring the SoC Designer Component .....	3
SoC Designer Component Files .....	3
Adding the Cycle Model to the Component Library .....	4
Adding the Component to the SoC Designer Canvas .....	4
Available Component ESL Ports .....	5
Setting Component Parameters .....	6
Debug Features .....	7
Registers Information .....	7
Available Profiling Data .....	9



# Preface

A Cycle Model component is a library developed from ARM intellectual property (IP) that is generated through Cycle Model Studio™. The Cycle Model then can be used within a virtual platform tool, for example, SoC Designer.

## About This Guide

This guide provides all the information needed to configure and use the Cycle Model in SoC Designer.

## Audience

This guide is intended for experienced hardware and software developers who create components for use with SoC Designer. You should be familiar with the following products and technology:

- SoC Designer
- Hardware design verification
- Verilog or SystemVerilog programming language

## Conventions

This guide uses the following conventions:

Convention	Description	Example
<code>courier</code>	Commands, functions, variables, routines, and code examples that are set apart from ordinary text.	<code>sparseMem_t SparseMemCreateNew();</code>
<i>italic</i>	New or unusual words or phrases appearing for the first time.	<i>Transactors</i> provide the entry and exit points for data ...
<b>bold</b>	Action that the user performs.	Click <b>Close</b> to close the dialog.
<text>	Values that you fill in, or that the system automatically supplies.	<platform>/ represents the name of various platforms.
[ text ]	Square brackets [ ] indicate optional text.	\$CARBON_HOME/bin/modelstudio [ <filename> ]
[ text1   text2 ]	The vertical bar   indicates “OR,” meaning that you can supply text1 or text 2.	\$CARBON_HOME/bin/modelstudio [<name>.symtab.db   <name>.ccfg ]

Also note the following references:

- References to C code implicitly apply to C++ as well.
- File names ending in .cc, .cpp, or .cxx indicate a C++ source file.

## Further reading

This section lists related publications. The following publications provide information that relate directly to SoC Designer:

- *SoC Designer Installation Guide*
- *SoC Designer User Guide*
- *SoC Designer Standard Component Library Reference Manual*

The following publications provide reference information about ARM® products:

- *AMBA 3 AHB-Lite Overview*
- *AMBA Specification (Rev 2.0)*
- *AMBA AHB Transaction Level Modeling Specification*
- *Architecture Reference Manual*

See <http://infocenter.arm.com/help/index.jsp> for access to ARM documentation.

The following publications provide additional information on simulation:

- IEEE 1666™ SystemC Language Reference Manual, (IEEE Standards Association)
- SPIRIT User Guide, Revision 1.2, SPIRIT Consortium.

## Glossary

AMBA	<i>Advanced Microcontroller Bus Architecture.</i> The ARM open standard on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a System-on-Chip (SoC).
AHB	<i>Advanced High-performance Bus.</i> A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol.
APB	<i>Advanced Peripheral Bus.</i> A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports.
AXI	<i>Advanced eXtensible Interface.</i> A bus protocol that is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.
Cycle Model	A software object created by the Cycle Model Studio (or <i>Cycle Model Compiler</i> ) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.
Cycle Model Studio	Graphical tool for generating, validating, and executing hardware-accurate software models. It creates a Cycle Model, and it also takes a Cycle Model as input and generates a component that can be used in SoC Designer, Platform Architect, or Accellera SystemC for simulation.
CASI	<i>ESL API Simulation Interface</i> , is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	<i>ESL API Debug Interface</i> , enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	<i>ESL API Profiling Interface</i> , enables collecting historical data from a component and displaying the results in various formats.
Component	Building blocks used to create simulated systems. Components are connected together with unidirectional transaction-level or signal-level connections.
ESL	<i>Electronic System Level.</i> A type of design and verification methodology that models the behavior of an entire system using a high-level language such as C or C++.
HDL	<i>Hardware Description Language.</i> A language for formal description of electronic circuits, for example, Verilog.
RTL	<i>Register Transfer Level.</i> A high-level hardware description language (HDL) for defining digital circuits.
SoC Designer	High-performance, cycle accurate simulation framework which is targeted at System-on-a-Chip hardware and software debug as well as architectural exploration.
SystemC	SystemC is a single, unified design and verification language that enables verification at the system level, independent of any detailed hardware and software implementation, as well as enabling co-verification with RTL design.
Transactor	<i>Transaction adaptors.</i> You add transactors to your component to connect your component directly to transaction level interface ports for your particular platform.

# Chapter 1

## Using the Cycle Model in SoC Designer

This chapter describes the functionality of the Model component, and how to use it in SoC Designer. It contains the following sections:

- [GPIO PL061 Cycle Model Functionality](#)
- [Adding and Configuring the SoC Designer Component](#)
- [Available Component ESL Ports](#)
- [Setting Component Parameters](#)
- [Debug Features](#)
- [Available Profiling Data](#)

### 1.1 GPIO PL061 Cycle Model Functionality

The PrimeCell GPIO is an AMBA compliant System-on-Chip peripheral. GPIO is an AMBA slave module, and connects to the APB. This Cycle Model supports receive and transmit direction transaction port for data transfer.

This section provides a summary of the functionality of the Cycle Model compared to that of the hardware. For details of the functionality of the hardware that the Cycle Model simulates, refer to the *ARM PrimeCell® General Purpose Input/Output (PL061) Technical Reference Manual*.

- [Fully Functional and Accurate Features](#)
- [Fully Functional and Approximate Features](#)
- [Unsupported Hardware Features](#)

### 1.1.1 Fully Functional and Accurate Features

The following features of the GPIO PL061 hardware implementation are fully implemented in the PL061 GPIO Cycle Model.

- Compliance to the AMBA (Rev 2.0) Specification.
- Eight individually programmable input/output pins; default to *input* at reset. *They can be configured only through software mode via the APB bus interface.*
- Programmable interrupt generation capability from a transition or level condition on any input pin.
- Bit masking in both read and write operations through address lines.
- Identification registers that uniquely identify the PrimeCell GPIO.
- All the internal registers with address map are supported, except for the functionality of the GPIOAFSEL register.
- Combined interrupt pin is supported.

### 1.1.2 Fully Functional and Approximate Features

- The APB interface is modeled as an APB transaction interface. For more information on this, refer to *RVModelLib\_MxAPB.pdf* provided with AMBA2 package for SoC Designer Cycle Model Library.
  - In particular, this APB interface does not contain a *reset* port and *clock* port. Instead, the *reset* port and *clock* port are provided separately.

### 1.1.3 Unsupported Hardware Features

The following features of the GPIO PL061 hardware are not implemented in the PL061 GPIO Cycle Model:

- Eight GPIOMIS interrupt output pins are not supported. Only a combined interrupt pin is supported.
- Hardware control capability of PrimeCell GPIO lines for different system configurations is not supported. Therefore, the eight input/output GPIO pins cannot be programmed through the hardware control interface.
- Features of following registers are not supported:
  - The Mode control select register, GPIOAFSEL, is not supported completely. Therefore, this Cycle Model works only in Software control mode, not in Hardware control mode.

## 1.2 Adding and Configuring the SoC Designer Component

The following topics briefly describe how to use the component. See the *SoC Designer User Guide* for more information.

- [SoC Designer Component Files](#)
- [Adding the Cycle Model to the Component Library](#)
- [Adding the Component to the SoC Designer Canvas](#)

### 1.2.1 SoC Designer Component Files

The component files are the final output from the Cycle Model Studio compile and are the input to SoC Designer. There are two versions of the component; an optimized *release* version for normal operation, and a *debug* version.

On Linux the *debug* version of the component is compiled without optimizations and includes debug symbols for use with gdb. The *release* version is compiled without debug information and is optimized for performance.

On Windows the *debug* version of the component is compiled referencing the debug runtime libraries, so it can be linked with the debug version of SoC Designer. The *release* version is compiled referencing the release runtime library. Both release and debug versions generate debug symbols for use with the Visual C++ debugger on Windows.

The provided component files are listed below:

**Table 1-1 Cycle SoC Designer Component Files**

Platform	File	Description
Linux	maxlib.lib<model_name>.conf	SoC Designer configuration file
	lib<component_name>.mx.so	SoC Designer component runtime file
	lib<component_name>.mx_DBG.so	SoC Designer component debug file
Windows	maxlib.lib<model_name>.windows.conf	SoC Designer configuration file
	lib<component_name>.mx.dll	SoC Designer component runtime file
	lib<component_name>.mx_DBG.dll	SoC Designer component debug file

Additionally, this User Guide PDF file is provided with the component.

## 1.2.2 Adding the Cycle Model to the Component Library

The compiled Cycle Model component is provided as a configuration file (*.conf*). To make the component available in the Component Window in SoC Designer Canvas, perform the following steps:

1. Launch SoC Designer Canvas.
2. From the *File* menu, select **Preferences**.
3. Click on **Component Library** in the list on the left.
4. Under the *Additional Component Configuration Files* window, click **Add**.
5. Browse to the location where the SoC Designer model is located and select the component configuration file:
  - `maxlib.lib<model_name>.conf` (for Linux)
  - `maxlib.lib<model_name>.windows.conf` (for Windows)
6. Click **OK**.
7. To save the preferences permanently, click the **OK & Save** button.

The component is now available from the SoC Designer *Component Window*.

## 1.2.3 Adding the Component to the SoC Designer Canvas

Locate the component in the *Component Window* and drag it out to the Canvas.

## 1.3 Available Component ESL Ports

Table 1-2 describes the ESL ports that are exposed in SoC Designer. See the *ARM PrimeCell® GPIO (PL061) Technical Reference Manual* for more information.

**Table 1-2 ESL Component Ports**

ESL Port	Description	Direction	Type
GPIO_IN	8-bit port used for receiving data from a signal master port. The data received via this port is displayed in the GPIODATA register.	Input	Signal slave
apb	APB port for memory mapped register accesses and data. This interface is expected to be connected to an AXI compliant interface via some AXI2APB bridge. Refer to the GPIO PL061 TRM for details of the memory mapped registers.	Input	Transaction slave
reset	Input reset. Reset port for receiving hardware reset signal. It should be high for at least one cycle for hardware reset to occur.	Input	Signal slave
clk-in	Input clock. The GPIO component is clocked at the frequency of the clock connected to this port. If <i>clk-in</i> is not connected, the clock frequency is taken from SoC Designer System Properties.	Input	Clock slave
GPIO_OUT	8-bit port used to transmit data to a connected signal slave port. The data transmitted via this port is taken from the GPIODATA register, however only those bits that are configured as output are taken - this information is available in the GPIODIR register.	Output	Signal master
intp	This port is used to provide interrupts to the interrupt controller. The value contains the interrupt number and the extended value contains the interrupt low or high value.	Output	Signal master

All pins that are not listed in this table have been either tied or disconnected for performance reasons.

## 1.4 Setting Component Parameters

You can change the settings of all the component parameters in SoC Designer Canvas, and of some of the parameters in SoC Designer Simulator. To modify the component's parameters:

1. In the Canvas, right-click on the component and select **Edit Parameters...**. You can also double-click the component. The *Edit Parameters* dialog box appears.
2. In the *Parameters* window, double-click the **Value** field of the parameter that you want to modify.
3. If it is a text field, type a new value in the *Value* field. If a menu choice is offered, select the desired option. The parameters are described in Table 1-3.

**Table 1-3 Component Parameters**

Name	Description	Allowed Values	Default Value	Runtime <sup>1</sup>
Align Waveforms	When set to <i>true</i> , waveforms dumped from the component are aligned with the SoC Designer simulation time. The reset sequence, however, is not included in the dumped data.  When set to <i>false</i> , the reset sequence is dumped to the waveform data, however, the component time is not aligned with the SoC Designer time.	true, false	true	No
apb Base Address	Base address for APB region accessed via the <i>apb</i> slave port.	0x0 - 0xffffffff	0x0	No
apb Enable Debug Messages	Whether debug messages are logged for the <i>apb</i> port.	true, false	false	Yes
apb Size	APB region size.	0x0-0xFFFFFFFF	0x100000000	No
Carbon DB Path	Sets the directory path to the database file.	Not Used	empty	No
Dump Waveforms	Whether SoC Designer dumps waveforms for this component.	true, false	false	Yes
Enable Debug Messages	Enable or disable the capture of debug messages.	true, false	false	Yes
intp int_0 id	Interrupt number for GPIO Interrupt. Always comes on “intp” signal master port	0-31	0	Yes
Waveform File <sup>2</sup>	Name of the waveform file.	<i>string</i>	pl061_gpio.fsdb	No
Waveform Timescale	Sets the timescale to be used in the waveform.	Many values in drop-down	1 ns	No

1. *Yes* means the parameter can be dynamically changed during simulation, *No* means it can be changed only when building the system, *Reset* means it can be changed during simulation, but its new value will be taken into account only at the next reset.

2. When enabled, SoC Designer writes accumulated waveforms to the waveform file in the following situations: when the waveform buffer fills, when validation is paused and when validation finishes, and at the end of each validation run.

## 1.5 Debug Features

The GPIO PL061 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate and control the registers in the SoC Designer Simulator.

### 1.5.1 Registers Information

This section lists the register views available for the GPIO PL061 Cycle Model in the SoC Designer Simulator. The GPIO PL061 Cycle Model has four sets of registers that are accessible via the debug interface. Registers are grouped into sets according to functional area.

- [General Registers](#)
- [Peripheral ID Registers](#)
- [PrimeCell ID Registers](#)
- [Test Registers](#)

See the *ARM PrimeCell® GPIO (PL061) Technical Reference Manual* for detailed descriptions of these registers.

### 1.5.1.1 General Registers

Table 1-4 shows the General registers.

**Table 1-4 General Registers Summary**

Register	Description	Type
GPIODATA <sup>1</sup>	Data register	read-only
GPIODIR	Data Direction register	read-write
GPIOIS	Interrupt Sense register	read-write
GPIOIBE	Interrupt Both-Edges register	read-write
GPIOIEV	Interrupt Event register	read-write
GPIOIE	Interrupt Mask register	read-only
GPIORIS	Raw Interrupt Status register	read-only
GPIOMIS	Masked Interrupt Status register	read-only
GPIOIC	Interrupt Clear register	read-only
GPIOAFSEL	Interrupt Mode Control Select register	read-write

1. The values seen from this register may be different than what is seen through debug. In debug you will see the entire register. When viewed through the APB port, values read and written to the register are based on the address used to access the GPIODATA register. For further information refer the PL061 TRM.

### 1.5.1.2 Peripheral ID Registers

Table 1-5 shows the Peripheral Identification registers.

**Table 1-5 Peripheral ID Registers Summary**

Register	Description	Type
GPIOPeriphID0	Peripheral Identification register bits 7:0. Identifies the part number of the peripheral.	read-only
GPIOPeriphID1	Peripheral Identification register bits 15:8. Identifies the part number and designer of the peripheral.	read-only
GPIOPeriphID2	Peripheral Identification register bits 23:16. Identifies the revision and designer of the peripheral.	read-only
GPIOPeriphID3	Peripheral Identification register bits 31:24. Identifies the configuration of the peripheral.	read-only

### 1.5.1.3 PrimeCell ID Registers

Table 1-6 shows the PrimeCell Identification registers.

**Table 1-6 PrimeCell ID Registers Summary**

Register	Description	Type
GPIOPCellIID0	PrimeCell Identification register bits 7:0. Determines the reset value.	read-only
GPIOPCellIID1	PrimeCell Identification register bits 15:8. Determines the reset value.	read-only
GPIOPCellIID2	PrimeCell Identification register bits 23:16. Determines the reset value.	read-only
GPIOPCellIID3	PrimeCell Identification register bits 31:24. Determines the reset value.	read-only

### 1.5.1.4 Test Registers

Table 1-7 shows the Test registers.

**Table 1-7 Test Registers Summary**

Register	Description	Type
GPIOITCR	Integration Test Control register	read-only
GPIOITIP1	Integration Test Input read/set register	read-only
GPIOITIP2	Integration Test Input read/set register	read-only
GPIOITOP1	Integration Test Output set register	read-only
GPIOITOP2	Integration Test Output read register	read-only
GPIOITOP3	Integration Test Output read/set register	read-only

## 1.6 Available Profiling Data

The GPIO PL061 Cycle Model component has no profiling capabilities.

