ARM® CoreLink™ CG092 AHB Flash Cache

Revision: r0p0

Technical Reference Manual

ARM DDI 0569A (ID040616)

ARM CoreLink CG092 AHB Flash Cache
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
15 March 2016	A	Non-Confidential	First release for r0p0.

Proprietary Notice

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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
ARM CoreLink CG092 AHB Flash Cache Technical Reference Manual

Preface

This preface introduces the *ARM® CoreLink™ CG092 AHB Flash Cache Technical Reference Manual*. It contains the following sections:

- *About this book* on page vii.
- *Feedback* on page x.

About this book

This book is for the *ARM® CoreLink™ CG092 AHB Flash Cache* component.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r0p0, where:

r*m* Identifies the major revision of the product, for example, r0.

p*n* Identifies the minor revision or modification status of the product, for example, p0.

Intended audience

This book is written for experienced hardware and SoC engineers who might or might not have experience with ARM products. Such engineers typically have experience of writing Verilog and of performing synthesis, but might have limited experience of integrating and implementing ARM products.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level view of the CG092 and a description of its features.

Chapter 2 *Functional Description*

Read this for a description of the major interfaces and components of the CG092.

Chapter 3 *Programmers Model*

Read this for a description of the address map and registers of the CG092.

Appendix A *Signal Descriptions*

Read this for an overview of the signals present in the CG092.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary

The *ARM Glossary* is a list of terms used in ARM documentation, together with definitions for those terms. The *ARM Glossary* does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

The *ARM Glossary* is available on the ARM Infocenter at http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Conventions

Conventions that this book can use are described in:

- *Typographical conventions* on page viii.
- *Timing diagrams* on page viii.
- *Signals* on page ix.

Typographical conventions

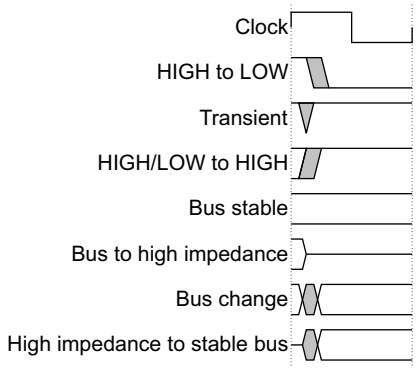
The following table describes the typographical conventions:

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace` *`italic`*	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com for access to ARM documentation.

See www.arm.com/cmsis for embedded software development resources including the *Cortex® Microcontroller Software Interface Standard* (CMSIS).

Note

Additional information about the product is provided in the release notes included with the product package.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Cortex®-M System Design Kit Technical Reference Manual* (ARM DDI 0479). http://infocenter.arm.com/help/topic/com.arm.doc.ddi0479c/index.html
- *ARM® IoT for Cortex-M Technical Reference Manual* (ARM DDI0551).

The following confidential books are only available to licensees or require registration with ARM:

- *ARM® CoreLink™ CG092 AHB Flash Cache Configuration and Integration Manual* (ARM DIT 0065).
- *ARM® AMBA® 3 APB Protocol Specification* (ARM IHI 0024). http://infocenter.arm.com/help/topic/com.arm.doc.ihi0024c
- ARM® *AMBA® 3 AHB-Lite Protocol Specification* (ARM IHI 0033). http://infocenter.arm.com/help/topic/com.arm.doc.ihi0033a

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- The title.
- The number, ARM DDI 0569A.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the ARM® CoreLink™ CG092 AHB Flash Cache.

It contains the following sections:

1.1 Features of the CG092

The CG092 AHB Flash Cache is an instruction cache designed to be instantiated between the bus interconnect and the eFlash controller. The CG092 has the following characteristics:

- Configurable cache size (minimum 256 bytes/way).
- Four words per cacheline.
- Supports 2-way set associative cache, or 1-way fully associative cache.
- Configurable address bus size (based on flash memory size) so that tag memory size can be minimized.
- SRAM power-control handshaking to an external power management unit.
- Supports automatic and manual SRAM power up and power down (with simple handshaking).

  If valid data is in the powered-down cache because the cache is in a low-power state, the cache contents should not be invalidated on wake up. The software can therefore save energy by avoiding invaliding the cache RAMs on wake up.
- Supports automatic or manual cache invalidate in the enabling sequence.

  This behavior can be overridden.
- 32 bit AHB slave interface to the AHB master in the system processor.
- 32 bit APB slave interface to the memory-mapped registers of the CG092.
- 128-bit AHB master interface to the eFlash.

  Note

  An eFlash controller is not supplied with the CG092 component.
- Interrupt request generated on SRAM power or manual invalidation errors.
- Optional run-time support for prefetch to improve performance when executing a sequence of code that has not been read before.

  Note

  The prefetching performance impact is application dependent and might have a negative impact on eFlash power consumption.
- Optional compile-time support configurable performance counters that measure cache hits and misses.

  Exported cache hit and cache miss status signals can be used by performance measurement logic implemented at SoC level.

1.2 Compliance

The CG092 complies with, or includes components that comply with:

- *Advanced High Performance Bus* (AHB-Lite) protocol. See the *AMBA® 3 AHB-Lite Protocol Specification*.
- *Advanced Peripheral Bus* (APB) protocol. See the *AMBA® APB Protocol Specification* (Rev 2.0).

1.3 Limitations

This section describes limitations of the CG092 design.

1.3.1 Cache configuration

Configure the CG092 (using CCR.SET_MAN_POW, CCR.SET_MAN_INV and CCR.SET_PREFETCH) only when the cache is in disabled state (SR.CS equal to 0) to prevent unintended cache behavior or an interrupt request.

1.3.2 Cache command requests

Manual power request (CCR.POW_REQ) cannot be cleared when cache is in manual power mode and the cache is not in disabled state, otherwise a power error interrupt is raised and the cache disables itself:

- Manual invalidation request (CCR.INV_REQ) must be set only when CG092 is in disabled state, otherwise an invalidation error interrupt is raised and the cache disables itself.
- Manual invalidation request will not start an invalidation process if it is configured to work in automatic invalidation request mode.
- Invalidation requests cannot be cleared via the APB bus, it is automatically cleared when the invalidation process finishes (either started manually or automatically) or when power or invalidation error occurs.

Note

The manual power request has no effect on CG092 behavior if it is configured to work in automatic power request mode.

1.3.3 Cache enabling/disabling

Cache becomes disabled/enabled on burst boundaries. If the CCR.EN is set/unset during a burst, the CG092 enables/disables itself only after the current burst is finished. The CG092 therefore starts/stops caching (cache lookup, making linefills, prefetching) after the current burst.

After CCR.EN is cleared, the CG092 can preform a prefetch transaction on the AHB master interface while the last AHB burst finishes on the Slave interface. In this case:

- The CG092 waits with disabling until the prefetch transaction AHB data phase is finished.
- No new linefill, prefetch is allowed to be executed from this point.
- After the prefetch is finished, CG092 starts disabling.

Note

Because the cache disables itself on burst boundaries, if a power error occurs the cache issues a power error interrupt but cannot disable itself immediately on detecting SRAM resource loss (**RAMPWRACK**=0). Invalid data might be returned on the AHB Slave port if a cache hit happens.

This limitation is valid only when the burst is on the bus with **HSELS**=1, otherwise the CG092 can switch state independently from the burst boundaries.

There are some restrictions on cache enable depending on cache state and the **APOW** and **AINV** signals. See *Enabling the cache* on page 2-10.

1.3.4 Linefill

This section describes linefill limitations.

Debug Accesses

Debug accesses (indicated by **HMASTERS** = 1):

- Are not cached.
- Are not looked up in the cache.
- Do not cause linefill or instruction prefetching.

These transactions are simply bypassed by the CG092.

Error Response

If the CG092 receives an error response (**HRESPM** = 1) for a read request, it will not update the internal linefill buffer and will not write the data into the SRAM.

If the read request was initiated by the AHB master the CG092 will propagate the received error response to the initiator, but if the read request was a result of prefetching the response is simply ignored.

1.3.5 Write through the cache

The CG092 is not a generic data cache:

- Write operation does not update the data in the cache and does not invalidate the cache either.
- If the flash memory has been updated, a cache invalidate operation must be executed by software.

1.3.6 APB Interface

The CG092 configuration and status registers can be accessed by the APB interface. The APB slave interface of the CG092 conforms to the AMBA APB Protocol Specification v2.0. with the following limitations:

- **PSTRB** input is ignored.
- **PSLVERR** is tied to LOW inside the CG092. The design never indicates transfer failure.
- Only **PPROT[0]** protection type is handled. Both normal and privileged reads give back the right register value but only privileged write accesses can write the registers. Non-privileged write accesses are ignored and no transfer failure is indicated on **PSLVERR**.

1.3.7 AHB slave interface

The 32-bit AHB slave interface of the CG092 is a fully functional AHB-Lite interface with the following limitations and extensions:

- The **HBURSTS** port is not used.
- Locked transfers are not supported. **HMASTLOCKS** port is not used.

- Non-AHB Lite port **HMASTERS** input is not implemented for debug access. All debug accesses are bypassed by the cache.

1.3.8 AHB Master Interface

The 128-bit AHB master interface of the CG092 is an AHB-Lite interface with the following limitations and extensions:

- The **HBURSTM** port is tied to `0b000`. The CG092 always makes SINGLE transfers.
- Locked transfers are not supported. **HMASTLOCKM** port is tied to `0b0`.
- **HTRANSM[0]** is tied to LOW inside the cache thus it will not propagates transfers from AHB slave to AHB master interface unmodified even in bypass mode. SEQ transfers are translated to NONSEQ transfers while BUSY transfers are translated to IDLE transfers.

Note

There is no power or speed gain in burst operations, so AHB bursts on are not supported on the AHB master interface:

- From the time the CG092 is enabled, the direct connection between the AHB Slave and Master interface is closed. The CG092 drives **HTRANSM** to IDLE and makes only NONSEQ transaction if a bypass, linefill or prefetch transaction must be executed.

  HSELM is continuously driven with `0b1` and transactions received with **HSELS** `0b0` on the AHB slave interface are not propagated to the AHB master interface. (**HTRANSM** remains in IDLE).
- The CG092 is transparent if disabled. If therefore a master/interconnect driving the AHB slave interface cancels/terminates a burst, this will appear on the AHB master interface. The CG092 might change control signals on the AHB master interface during a waited transfer (terminating and starting a new address phase)

1.4 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

Chapter 2
Functional Description

This chapter describes the functionality of the CG092.

It contains the following sections:

2.1 About the CG092

The CG092 is a simple cache for on-chip embedded flash (eFlash). The CG092 design is optimized for fetching Cortex-M3 or Cortex-M4 instructions directly from an eFlash. The main benefit of the CG092 is improved power efficiency, but there are also improvements in code fetching performance.

The figure below shows the connections in a typical Flash subsystem:

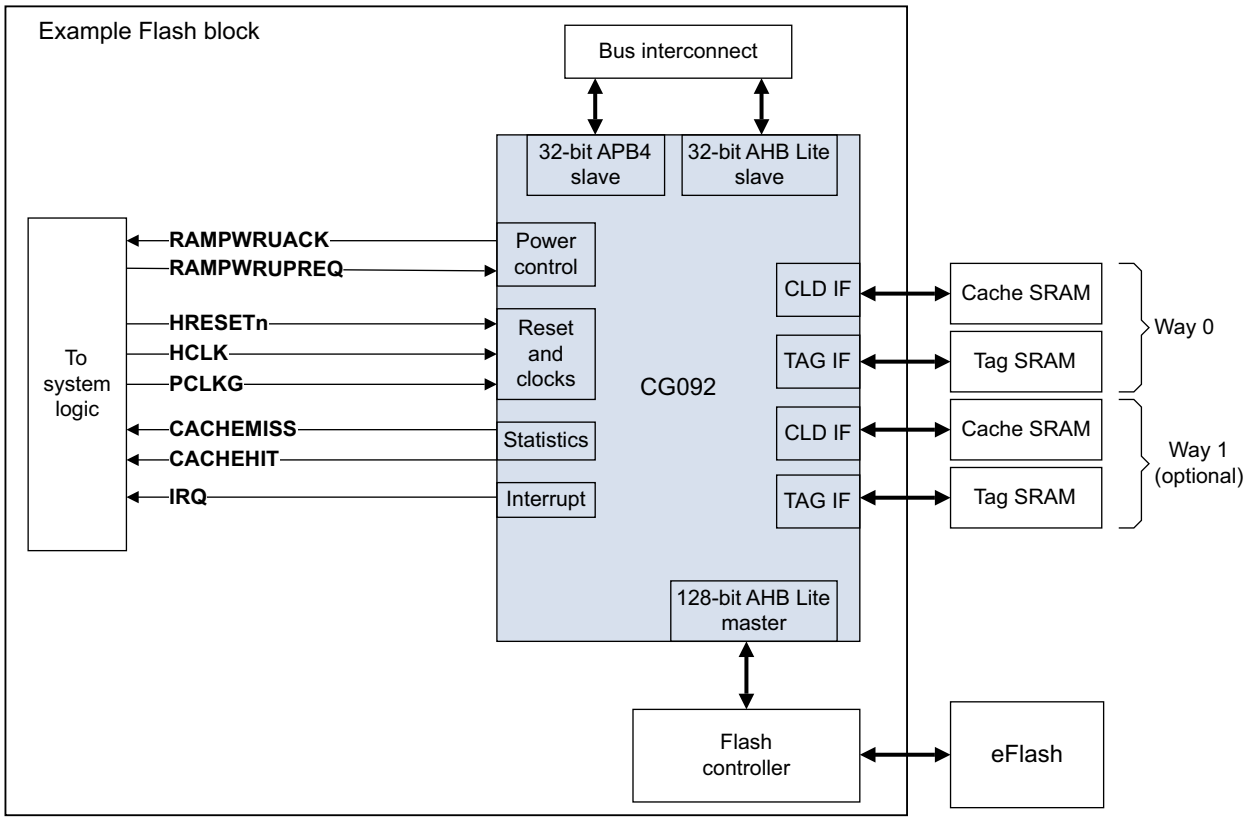


Figure 2-1 Example eFlash implementation

2.2 Interfaces

This section describes each interface in the CG092:

- *Clocking and reset*.
- *Cache SRAM power control*.
- *Interrupt request* on page 2-4.
- *Statistics ports* on page 2-5.
- *32-bit APB slave* on page 2-5.
- *128-bit AHB-Lite master* on page 2-6.
- *32-bit AHB-Lite slave* on page 2-6.
- *Cache TAG SRAM* on page 2-6.
- *Cache Line Data SRAM* on page 2-7.

For a full list of component signals, see Appendix A *Signal Descriptions*.

2.2.1 Clocking and reset

All ports are asynchronous and active high except the ones where stated opposite. Inputs and outputs are not registered (except where it is indicated) to reach zero latency SRAM accesses and transaction bypass where it is possible.

Table 2-1 Clock and reset signals

Signal Name	Direction	Description
HCLK	Input	AHB Bus clock. This clock is used for all always on logic.
PCLKG	Input	Gated clock input for register interface (APB). It must be the same frequency and same phase as HCLK. Can be gated, when there are no APB activities. It is expected to run while APB interface PSEL signal is asserted.
HRESETn	Input	Active low asynchronous AHB reset. Reset all or a subset of the CG092 registers depending on the RESET_ALL_REGS parameter.

HCLK and **PCLKG** clocks are handled as synchronous clocks by the design internally.

HCLK must not be gated even if the CG092 is disabled because the AHB data path multiplexing logics requires it for proper operation.

2.2.2 Cache SRAM power control

The **RAMPWRUPREQ** and **RAMPWRUPACK** signals are implement a full handshake mechanism. To guarantee the safe and correct handshake process the supported cache enable/disable and invalidation scenarios must be used.

Table 2-2 Cache SRAM power control interface signals

Signal Name	Direction	Description
RAMPWRUPREQ	Output	SRAM power up request. Indicating an SRAM resource request from the cache. Port is registered and synchronous to **HCLK**.
RAMPWRUPACK	Input	SRAM powers up acknowledge. Indicates that the requested SRAM resource is available. Port is synchronized to **HCLK** clock domain.

2.2.3 Interrupt request

The CG092 implements one interrupt output port (IRQ) and handles the following interrupts:

- Power Error interrupt.
- Manual Invalidation Error interrupt.

In addition to the HW interrupt signal, you can determine the interrupt state by reading the IRQSTAT interrupt status register.

The following table describes the interrupt interface:

Table 2-3 Interrupt request interface signals

Signal Name	Direction	Description
IRQ	Output	Interrupt request line. Interrupt request is an active high level signal. Port is registered and synchronous to **HCLK**.

Power error interrupt

This interrupt occurs if the CG092 is enabled while the SRAM resource is removed.

This error type indicates that either:

- The external SRAM resource is not available (**RAMPWRUPACK** = `0b0` and SR.POW_STAT = `0b0`)
- The CG092 is instructed to deassert **RAMPWRUPREQ** while the cache is enabled thus it would cause an error to power down the SRAMs.

The power error interrupt is raised if any of the following conditions are met:

- The CG092 is enabled when SRAM resource is removed.
- The CG092 is instructed to make a manual invalidation while SRAM resource was not available or removed.
- The CG092 is enabled and operates in manual power request mode while it is instructed to release SRAM resource (CCR.POW_REQ=0). In this case the power request is not cleared, but the cache disables itself due to the error interrupt.

Manual invalidation error interrupt

This interrupt occurs if the manual invalidate process starts and the CG092 is enabled.

Manual invalidation process must not be running while the CG092 is enabled, because an incoming cacheable read transaction can corrupt the invalidation process and the cache might return an invalid read response. Manual invalidation error occurs when manual invalidation request has happened when the CG092 was enabled.

On error detection:

1. The cache prevents the start of the invalidation process
2. Clears the CCR.INV_REQ bit.
3. Raises an interrupt request if it is not masked and disables the cache.

To resume normal operation:

1. Clear the interrupt request
2. Re-initiate a manual or automatic invalidation.

Interrupt masking

Interrupts can be masked from the IRQMASK register. The cause of the interrupt request indication can be read from the IRQSTAT interrupt status register. The user can clear an interrupt by writing `0b1` to the corresponding status bitfield.

Each type of error interrupt regardless of the IRQMASK setting disables the cache to safely bypass all transaction and clears the CCR.EN bit.

The CG092 cannot be enabled again (CCR.EN bit set) until the error cause is resolved and the corresponding IRQSTAT bitfield is cleared.

To resume normal operation, the cache must be re-initiated with invalidation according to the programming sequences described in the Functional Description section.

Note

Invalidation is required because the CG092 cannot indicate whether an SRAM write caused the interrupt, thus SRAM content might be corrupted.

2.2.4 Statistics ports

Statistics ports are present and always driven. The following table lists the interface signals.

Table 2-4 Statistics interface

Signal Name	Direction	Description
CACHEMISS	Output	Active high single cycle pulses indicating that a cache miss happened during cache look up. Port is registered and synchronous to **HCLK**.
CACHEHIT	Output	Active high single cycle pulses indicating that a cache hit happened during cache look up. Port is registered and synchronous to **HCLK**.

The system designer can select generation of status registers that count cache hits and misses.

If more detailed statistics are required, the SoC designer can use the **CACHEHIT** and **CACHEMISS** signals and implement custom statistics collection logic. See *Cache Statistic Hit Register, CSHR* on page 3-7 and *Cache Statistic Miss Register, CSMR* on page 3-7.

2.2.5 32-bit APB slave

The configuration and status registers are accessed by the 32-bit APB slave interface of the CG092. The interface conforms to the AMBA APB4 Protocol Specification v2.0 with the following limitations:

- **PSTRB** input is not used.
- **PSLVERR** is tied to LOW inside the CG092. The design never indicates transfer failure.

- Only **PPROT[0]** protection type is handled. Both normal and privileged reads give back the right register value but only privileged write accesses can write the registers. Non-privileged write accesses are ignored and no transfer failure is indicated on **PSLVERR**.

2.2.6 128-bit AHB-Lite master

The 128-bit AHB master interface of the CG092 connects to the Flash controller. The interface is an AHB-Lite interface with the following limitations and extensions:

- **HBURST** is tied to `0b000`. The CG092 component always makes SINGLE transfers.
- **HMASTLOCK** is tied to `0b0`. Locked transfers are not supported.
- **HTRANSM[0]** is tied to LOW inside the cache thus it will not propagates transfers from AHB slave to AHB master interface unmodified even in bypass mode. SEQ transfers are translated to NONSEQ transfers while BUSY transfers are translated to IDLE transfers.

HBURSTM and **HTRANSM** limitations are specified because there is no power or speed gain in burst operations. Thus the CG092 does not support AHB bursts on the AHB master interface and has the following limitations:

- From the time the CG092 is enabled, the direct connection between the AHB Slave and Master interface is closed.

  The CG092 drives **HTRANSM** to IDLE and makes only **NSEQ** transaction if a bypass, linefill or prefetch transaction must to be executed.

  HSELM is continuously driven with `0b1` and transactions received with **HSELS** `0b0` on the AHB slave interface are not propagated to the AHB master interface. (**HTRANSM** remains in IDLE)

- The CG092 is transparent if disabled, therefore if a master/interconnect driving the AHB slave interface cancels/terminates a burst, this will appear on the AHB master interface The control signals might change on the AHB master interface during a waited transfer (terminating and starting a new address phase)

2.2.7 32-bit AHB-Lite slave

The 32-bit AHB slave interface of the CG092 connects to the system bus to provide data access to the cache. The interface is an AHB-Lite interface with the following limitations and extensions:

- **HBURST** port is not used.
- Locked transfers are not supported. **HMASTLOCK** port is not used.
- Non-AHB Lite port **HMASTERS** input implemented for debug access. All debug accesses are bypassed by the cache.

2.2.8 Cache TAG SRAM

TAG SRAM clocks must have the same frequency as **HCLK**.

The CG092 expects the read data on the next rising clock edge after the read request was asserted (single cycle access).

Note

The TAG SRAM data bus width depends on the configuration options made by the implementation engineer.

Table 2-5 Cache interface for way 0 TAG SRAM

Signal Name	Direction	Description
RAMTAG0ADDR[CW-5:0]	Output	Parameterized width data address bus
RAMTAG0WE	Output	Write control (same cycle as address)
RAMTAG0RD	Output	Read control (same cycle as address)
RAMTAG0CS	Output	Chip select (same cycle as address)
RAMTAG0WDATA[(AW-CW):0]	Output	Write data (same cycle as address)
RAMTAG0RDATA[(AW-CW):0]	Input	Read data (1 cycle after address)

Table 2-6 Cache interface for way 1 TAG SRAM

Signal Name	Direction	Description
RAMTAG1ADDR[CW-5:0]	Output	Parametrized width data address bus
RAMTAG1WE	Output	Write control (same cycle as address)
RAMTAG1RD	Output	Read control (same cycle as address)
RAMTAG1CS	Output	Chip select (same cycle as address)
RAMTAG1WDATA[(AW-CW):0]	Output	Write data (same cycle as address)
RAMTAG1RDATA[(AW-CW):0]	Input	Read data (1 cycle after address). Tie this port either low or high if direct mapped cache type is set (**CACHE_WAY** = 1).

2.2.9 Cache Line Data SRAM

The following requirements are made on the external system:

- SRAM clocks must have the same frequency and same phase as **HCLK**.
- The CG092 expects to get the read data on the next rising clock edge after the read request was asserted (single cycle access).
- **RAMCLDxRD** and **RAMCLDxCS** vectors hold the same one-hot code value during read transactions to permit reading only a 32-bit portion of the 128-bit CLD SRAM interface. Reading only 32 bits saves some power.
- **RAMCLDxRD** and **RAMCLDxCS** signals are decoded from the **RAMCLDxADDR[3:2]** port value and only the corresponding part of the **RAMCLDxRDATA[127:0]** is read inside the CG092.

  For example if **RAMCLD1ADDR[3:2]** = `0b01`
  - **RAMCLD1RD** and **RAMCLD1CS** are both `0b0010`.
  - **RAMCLD1ADDR[63:32]** is read.

— **RAMCLD1ADDR[127:64]** and **RAMCLD1ADDR[31:0]** are don't cares.

Table 2-7 Cache SRAM interface for way 0 Cache Line Data (CLD) SRAM

Signal Name	Direction	Description
RAMCLD0ADDR[CW-5:0]	Output	Parametrized width data address bus
RAMCLD0WE	Output	Write control (same cycle as address)
RAMCLD0RD[3:0]	Output	Read control (same cycle as address)
RAMCLD0CS[3:0]	Output	Chip select (same cycle as address)
RAMCLD0WDATA[127:0]	Output	Write data (same cycle as address)
RAMCLD0RDATA[127:0]	Input	Read data (1 cycle after address)

Table 2-8 Cache SRAM interface for way 1 Cache Line Data (CLD) SRAM

Signal Name	Direction	Description
RAMCLD1ADDR[CW-5:0]	Output	Parametrized width data address bus
RAMCLD1WE	Output	Write control (same cycle as address)
RAMCLD1RD[3:0]	Output	Read control (same cycle as address)
RAMCLD1CS[3:0]	Output	Chip select (same cycle as address)
RAMCLD1WDATA[127:0]	Output	Write data (same cycle as address)
RAMCLD1RDATA[127:0]	Input	Read data (1 cycle after address) Tie this port either low or high if direct mapped cache type is set (**CACHE_WAY** = 1).

Note

CLD SRAM interface read (RD) and chip select (CS) ports are 4-bit vectors while the write enable (WE) is only a single bit port. The CG092 always updates a full 128bit line, while during a cache hit only 32-bit portion of a full 128-bit line is read to save some power.

2.3 Operation

This section describes the behavior of the CG092 in normal operation.

2.3.1 Cache disabled

When the CG092 is disabled all read and write accesses are bypassed. Only the **HTRANS** property of the bypassed transactions is changing. See *Limitations* on page 1-4.

There is no additional latency in terms of clock cycles.

2.3.2 Cache enabled

The following operating modes are possible if the cache is enabled:

Bypass

When the CG092 is enabled and fully operational it works as an instruction cache thus all write, debug and non-cacheable accesses are bypassed. One cycle latency is required per transfer, and the latency can increase if there is an on-going pre-fetch access in the downstream AHB.

Cache hit

If a cacheable read transaction data is found in the cache memory (cache hit) no transaction will be generated on the AHB master interface. Cache sends **OKAY** response on the AHB slave interface with the looked up data. No wait state is required.

Linefill

Cache miss happens when cacheable read transaction data is looked up but not found in the cache memory. Cache miss events cause linefill transactions on the AHB master port with the following properties:

- Transfer type NONSEQ (**HTRANSM[1:0]** = `0b10`)
- Transfer size 4-word line (**HSIZEM[2:0]** = `0b100`)
- Fixed cacheable and bufferable protection control (**HPROTM[3]** = 1 and **HRPOTM[2]** = 1)
- All other properties are set according the incoming read transaction causing the linefill

The CG092 stores the linefill data in the cache memory and sends a response on the AHB slave interface as well. One cycle latency is required per transfer, and the latency can increase if there is an on-going pre-fetch access in the downstream AHB.

Prefetch

The pre-fetch feature is a programmable option. When the prefetch feature is enabled by CCR.SET_PREFETCH, the design does the prefetch after linefill if the following conditions are true:

- The transaction causing linefill is opcode fetch (**HPROTS[0]** = 0)
- The data in the linefill buffer has already been written to SRAM
- The downstream AHB bus is free

Prefetch transactions on the AHB master port are generated with the following properties:

- Transfer type NONSEQ (**HTRANSM[1:0]** = `0b10`)
- Transfer size 4-word line (**HSIZEM[2:0]** = `0b100`)
- Fix cacheable, bufferable and opcode fetch protection control (**HPROTM[3]** = 1, **HRPOTM[2]** = 1 and **HPROTM[0]** = 0)
- All other properties are set according the previous linefill transaction. (Address is incremented with `0x10`).

Note

The prefetching performance impact is application dependent and might have a negative impact on eFlash power consumption.

The prefetch feature only helps if the code has not been cached. In many cases, the performance of the system might be reduced by prefetching because, once the prefetch is started, it cannot be abandoned. For example, if two cache misses occur, the flash must finish the prefetch access before reading the data requested by the processor.

2.3.3 Latency

When the cache is disabled, all the transfers from upstream AHB interface are bypassed to downstream AHB interface without latency, within the same clock cycle.

The access latencies are as follows if the cache is enabled:

- A read with cache hit is completed with 0 wait state.
- A cacheable read with cache miss is passed on to downstream AHB with 1 cycle latency. Additional latency might be required if the cache is doing a pre-fetch operation.
- A non-cacheable read is passed on to downstream AHB with 1 cycle latency.
- All write accesses are passed on to downstream AHB with 1 cycle latency.

2.3.4 Enabling the cache

The tables in this section use the following terminology:

APOW Automatic Power Control: **CCR.SET_MAN_POW**= 0 selects automatic cache RAM power control mode.

AINV Automatic Invalidation request: **CCR.SET_MAN_INV** = 0 selects automatic cache invalidate.

SRAM resource

SRAM resource are considered to be an CG092 resource.

SRAM resource request

CG092 requests access to SRAMs (using **RAMPWRUPREQ**).

SRAM resource availability

SRAMs are accessible for read and write (indicated by **RAMPWRUPACK**).

The following table describes how to enable the CG092 with different configuration settings if the SRAM content is invalid, such as at power on.

Table 2-9 Cache enable with invalid SRAM contents

APOW	AINV	Control sequence
0	0	1. Check that cache is disabled (SR[1:0]=0). 2. Manual power request. 3. Wait until SRAM resource is available and SR[4] is set. 4. Issue manual invalidate request. 5. Wait until invalidate request has completed and CCR[1]=0. 6. Enable cache.
0	1	1. Check that cache is disabled (SR[1:0]=0). 2. Manual power request. 3. Wait until SRAM resource is available and SR[4] is set. 4. Enable cache. **Note** Enabling cache before SRAMs are accessible causes a power error.
1	0	**Caution** Not supported. Not deterministic behavior.
1	1	1. Check that cache is disabled (SR[1:0]=0). 2. Enable cache.

The following table describes how to enable the CG092 with different configuration settings if the SRAM contents are valid, such as a wake up from retention.

Table 2-10 Cache enable with invalid SRAM contents

APOW	AINV	Control sequence
0	0	1. Check that cache is disabled (SR[1:0]=0). 2. Issue manual power request. 3. Wait until SRAM resource is available and SR[4] is set. 4. Enable cache. **Note** Enabling cache before SRAMs are accessible causes a power error.

Table 2-10 Cache enable with invalid SRAM contents (continued)

APOW	AINV	Control sequence
0	1	**Caution** Not supported. Not deterministic behavior.
1	0	1. Check that cache is disabled (SR[1:0]=0). 2. Enable cache.
1	1	**Caution** Not supported. Not deterministic behavior.

2.3.5 Disabling the cache

The following table describes the required steps to disable the CG092 with different configuration settings.

Table 2-11 Cache disable sequence

APOW	AINV	Control sequence
0	0	1. Disable Cache 2. Clear manual power request (optional) **Note** SRAM resource request deasserted only if manual power request control bit is clear (**CCR[2]**=0).
0	1	1. Disable Cache 2. Clear manual power request (optional) **Note** SRAM resource request deasserted only if manual power request control bit is clear (**CCR[2]**=0).
1	0	Disable Cache **Note** SRAM resource request deasserted automatically.
1	1	Disable Cache **Note** SRAM resource request deasserted automatically.

Note

Ensure that **RAMPWRUPACK** is deasserted (SR.POW_STAT=0) whenever the **RAMPWRUPREQ** is deasserted, either automatically or manually, to realize a full handshake mechanism on the SRAM power interface.

2.3.6 Invalidating SRAM

This section describes how to invalidate the CG092 SRAMs when the cache is enabled and resume normal operation.

The followings conditions must be fulfilled to successfully invalidate the SRAM:

- SRAM resources must be available for the CG092 from the time the invalidation is requested until the invalidation is finished (**RAMPWRUPACK** = `0b1` and **SR.POW_STAT** = `0b1`).

  If SRAM resources are not available:

  — Invalidation fails.

  — Error status is set in the **IRQSTAT.POW_ERR** status registers.

  — The interrupt request line is asserted (if not masked via **IRQMASK.POW_ERR**).

  — The CG092 goes to idle state (if it was not in idle state before) and bypasses all transactions.

- Manual invalidation must not be triggered when the CG092 is enabled. This action causes an interrupt if not masked (**IRQMASK.MAN_INV_ERR**).

  Note

  If the CG092 is enabled during manual invalidation, entry to the enable state is delayed until the manual invalidation process is finished to avoid a invalidation/transaction failure.

- Manual invalidation can be triggered only when the CG092 is operates in manual invalidation mode (**CCR.SET_MAN_INV** = `0b1`).

  Manual invalidation is requested by setting **CCR.INV** bitfield. This bitfield is automatically cleared when invalidation procedure is finished.

The following table describes the required steps to invalidate the cache with different configuration settings

Table 2-12 Cache invalidation sequence

APOW	AINV	Control sequence
0	0	1. Disable cache. 2. Check that cache is disabled (SR[1:0]=0). 3. Manual invalidate request. Optionally, wait until invalidate is complete and CCR[1] is cleared. 4. Enable cache. **Note** SRAM resource request deasserted only if manual power request control bit is clear.

Table 2-12 Cache invalidation sequence (continued)

APOW	AINV	Control sequence
0	1	1. Disable cache. 2. Check that cache is disabled (SR[1:0]=0). 3. Enable cache.
1	0	**Caution** Not supported. Not deterministic behavior.
1	1	1. Disable cache. 2. Check that cache is disabled (SR[1:0]=0). 3. Enable cache.

2.3.7 Performance monitoring logic

Both cache hit and cache miss pulses can be counted separately by 32-bit internal registers that saturate at `0xFFFF_FFFF`. The hit/miss counters value can be read from the CSHR and CSMR registers and can be cleared by a write access to the desired register.

Note

Cache hit/miss counters are optional. The SoC designer must set configuration parameter GEN_STAT_LOGIC to 1 to generate the related counter logic.

The **CACHEHIT** and **CACHEMISS** signals are always enabled however, and can be used by external performance monitoring logic.

Cache hit The cache hit event is an active-high single cycle pulse generated when an incoming cacheable read transaction (no debug access counted) requested data is found in the cache memory (buffers, SRAMs). Cache hit event pulses are ported to **CACHEHIT** output.

Cache miss The cache miss event is an active high single cycle pulse generated when an incoming cacheable read transaction (no debug access counted) requested data is not found in the cache memory (buffers, SRAMs). Cache miss event pulses are ported to **CACHEMISS** output.

Chapter 3
Programmers Model

This chapter describes the CG092 registers, and provides information on how to program a SoC that contains an implementation of the CG092.

It contains the following sections:

3.1 About this programmers model

This section describes the registers in the CG092 and their function.

The following information applies to all registers:

- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in unpredictable behavior.
- Unless otherwise stated in the accompanying text:
  - — Do not modify undefined register bits.
  - — Ignore undefined register bits on reads.
  - — All register bits are reset to a logic 0 by a system or power up reset.
- The following describes the access type:
  RW Read and write.
  RO Read-only.
  WO Write-only.
  RAZ Read as zero. Bits in a register that always return 0 when read.

3.2 Register summary

The table below summarizes the registers in the CG092.

Table 3-1 CG092 registers

Register name	Type	Width	Reset Value	Address Offset	Description
CCR	RW	32	0x00000040	0x000	*Configuration and Control Register, CCR* on page 3-4
SR	RO	32	0x00000000	0x004	*Status Register, SR* on page 3-4
IRQMASK	RW	32	0x00000000	0x008	*Interrupt Request Mask register, IRQMASK* on page 3-5
IRQSTAT	RW	32	0x00000000	0x00C	*Interrupt Request Status register, IRQSTAT* on page 3-5
HWPARAMS	RO	32	-	0x010	*Hardware Parameters register, HWPARAMS* on page 3-6
CSHR	RW	32	0x00000000	0x014	*Cache Statistic Hit Register, CSHR* on page 3-7
CSMR	RW	32	0x00000000	0x018	*Cache Statistic Miss Register, CSMR* on page 3-7
PIDR4	RO	32	0x00000004	0xFD0	*Peripheral ID Register, PIDR4* on page 3-7
PIDR5	RO	32	0x00000000	0xFD4	*Peripheral ID Register, PIDR5* on page 3-8
PIDR6	RO	32	0x00000000	0xFD8	*Peripheral ID Register, PIDR6* on page 3-8
PIDR7	RO	32	0x00000000	0xFDC	*Peripheral ID Register, PIDR7* on page 3-8
PIDR0	RO	32	0x00000029	0xFE0	*Peripheral ID Register, PIDR0* on page 3-9
PIDR1	RO	32	0x000000B8	0xFE4	*Peripheral ID Register, PIDR1* on page 3-9
PIDR2	RO	32	0x0000000B	0xFE8	*Peripheral ID Register, PIDR2* on page 3-9
PIDR3	RO	32	0x00000000	0xFEC	*Peripheral ID Register, PIDR3* on page 3-10
CIDR0	RO	32	0x0000000D	0xFF0	*Component ID Register, CIDR0* on page 3-10
CIDR1	RO	32	0x000000F0	0xFF4	*Component ID Register, CIDR1* on page 3-10
CIDR2	RO	32	0x00000005	0xFF8	*Component ID Register, CIDR2* on page 3-11
CIDR3	RO	32	0x000000B1	0xFFC	*Component ID Register, CIDR3* on page 3-11

3.3 Register descriptions

Each register description provides information about the register, such as usage constraints, configurations, attributes, and bit assignments.

3.3.1 Configuration and Control Register, CCR

Configuration and control register.

RW register at offset `0x000`.

Table 3-2 CCR register

Bits	Name	Description	Access	Reset
[31:7]	Reserved	-	RO, RAZ	0
6	STATISTIC_EN	Enable statistics logic: 0: Disabled. Counters are stalled. 1: Enable. Counters are running.	RW	1
5	SET_PREFETCH	Cache Prefetch Setting: 0: Disable prefetch. 1: Enable prefetch.	RW	0
4	SET_MAN_INV	Cache Invalidate Setting: 0: Automatic cache invalidate when cache enabled. 1: Manual cache invalidate mode.	RW	0
3	SET_MAN_POW	Power Control Setting: 0: Automatic. 1: Manual.	RW	0
2	POW_REQ	Manual SRAM power request.	RW	0
1	INV_REQ	Manual invalidate request. Functional only when SET_MAN_INV is set. Automatically cleared when invalidation is finished or power or invalidation error occurs. Cannot be cleared manually. Manual invalidation request must be set only when the cache is disabled, otherwise it causes an invalidation error interrupt.	RW	0
0	EN	Cache Enable: 0: Disable cache. 1: Enable cache.	RW	0

3.3.2 Status Register, SR

Status register.

RO register at offset 0x004.

Table 3-3 SR register

Bits	Name	Description	Access	Reset
[31:5]	Reserved	-	RO, RAZ	0
4	POW_STAT	SRAM power acknowledges. Real-time registered value of RAMPWRUPACK port.	RO	0
3	Reserved	-	RO, RAZ	0
2	INV_STAT	Invalidating Status. Indicates if invalidation process is ongoing.	RO	0
[1:0]	CS	Cache status: 0: Cache disabled. 1: Cache enabling. 2: Cache enabled. 3: Cache disabling.	RO	0

3.3.3 Interrupt Request Mask register, IRQMASK

Interrupt request mask register. Set to 0 to enable interrupts for events, and set to 1 to mask interrupts.

RW register at offset 0x008.

Table 3-4 IRQMASK register

Bits	Name	Description	Access	Reset
[31:2]	Reserved	-	RO, RAZ	0
1	MAN_INV_ERR	Mask interrupt request on manual invalidation error indication (IRQSTAT.MAN_INV_ERR is set).	RW	0
0	POW_ERR	Mask interrupt request on Power Error indication (IRQSTAT.POW_ERR is set).	RW	0

3.3.4 Interrupt Request Status register, IRQSTAT

Interrupt Request Status Register. IRQSTAT register status bits cannot be masked. They are set on the corresponding error event regardless of IRQMASK settings.

RW register at offset 0x00C.

Table 3-5 IRQSTAT register

Bits	Name	Description	Access	Reset
[31:2]	Reserved	-	RO, RAZ	0
1	MAN_INV_ERR	Manual invalidation error status. Set when manual invalidation is requested meanwhile the cache is not disabled. Write 1 to clear.	RW	0
0	POW_ERR	SRAM power error. Write 1 to clear. Power acknowledge de-asserted during operation. Manual power request de-asserted while cache is enabled and operating in manual power mode.	RW	0

3.3.5 Hardware Parameters register, HWPARAMS

Hardware parameters register holding implementation-defined parameter values.

RO register at offset 0x010.

Table 3-6 HWPARAMS register

Bits	Name	Description	Access	Reset
[31:14]	Reserved	-	RO, RAZ	0
13	GEN_STAT_LOGIC	Indicates GEN_STAT_LOGIC hardware parameter value.	RO	1
12	RESET_ALL_REGS	Indicates RESET_ALL_REGS hardware parameter value.	RO	1
[11:10]	CACHE_WAY	Implementation-defined value for number of cache ways: • 1: One cache way. • 2: Two cache ways.	RO	2
[9:5]	CW	Implementation-defined value for cache way width: • 8: 256B. • 9: 512B. • 11: 2KB. • 12: 4KB.	RO	0x0C
[4:0]	AW	Implementation-defined value for AHB-Lite bus width for the flash address space: • 18: 256KB. • 19: 512KB.	RO	0x12

3.3.6 Cache Statistic Hit Register, CSHR

Cache Statistic Hit Register contains the cache hit count.

Including this register in a design is optional. If not present and the register is accessed, a slave **OKAY** response is given and the register address is RAZ.

RW register at offset `0x014`.

Table 3-7 CSHR register

Bits	Name	Description	Access	Reset
`[31:0]`	CSHR	Counts the number of cache hits during cache look up. Only cacheable read transactions are looked up by the CG092. Writing to the register clears the contents.	RW	0

The count value in the registers saturate at `0xFFFF_FFFF`. Perform a write access to a register to clear the counter.

3.3.7 Cache Statistic Miss Register, CSMR

Cache Statistic Miss Register contains the cache miss count.

Including this register in a design is optional. If not present and the register is accessed, a slave **OKAY** response is given and the register address is RAZ.

RW register at offset `0x018`.

Table 3-8 CMSR register

Bits	Name	Description	Access	Reset
`[31:0]`	CSMR	Counts the number of cache misses during cache look up. Only cacheable read transactions are looked up by the CG092. Writing to the register clears the contents.	RW	0

The count value in the registers saturate at `0xFFFF_FFFF`. Perform a write access to a register to clear the counter.

3.3.8 Peripheral ID Register, PIDR4

Peripheral ID register.

RO register at offset 0xFD0.

Table 3-9 PIDR4 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:4]	SIZE	-	RO	0x0
[3:0]	DES_2	JEP 106 continuation code	RO	0x4

3.3.9 Peripheral ID Register, PIDR5

Peripheral ID register.

RO register at offset 0xFD4.

Table 3-10 PIDR5 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	Reserved	-	RO, RAZ	0

3.3.10 Peripheral ID Register, PIDR6

Peripheral ID register.

RO register at offset 0xFD8.

Table 3-11 PIDR6 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	Reserved	-	RO, RAZ	0

3.3.11 Peripheral ID Register, PIDR7

Peripheral ID register.

RO register at offset 0xFDC.

Table 3-12 PIDR7 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	Reserved	-	RO, RAZ	0

3.3.12 Peripheral ID Register, PIDR0

Peripheral ID register.

RO register at offset 0xFE0.

Table 3-13 PIDR0 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	PART_0	Part number bits [7:0].	RO	0x29

3.3.13 Peripheral ID Register, PIDR1

Peripheral ID register.

RO register at offset 0xFE4.

Table 3-14 PIDR1 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:4]	DES_0	PJEP106 identification code bits [3:0].	RO	0xB
[3:0]	PART_1	Part number bits [11:8].	RO	0x8

3.3.14 Peripheral ID Register, PIDR2

Peripheral ID register.

RO register at offset 0xFE8.

Table 3-15 PIDR2 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:4]	REVISION	Revision number of the peripheral	RO	0x0
3	JEDEC	Always set. Indicates that a JEDEC assigned value is used.	RO	1
[2:0]	DES_1	JEP106 identification code bits [11:8].	RO	0x3

3.3.15 Peripheral ID Register, PIDR3

Peripheral ID register.

RO register at offset 0xFEC.

Table 3-16 PIDR3 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:4]	REVAND	ECO revision.	RO	0x0
[2:0]	CMOD	Customer modified number.	RO	0x0

3.3.16 Component ID Register, CIDR0

Component ID register.

RO register at offset 0xFF0.

Table 3-17 CIDR0 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	PRMBL_0	-	RO	0x0D

3.3.17 Component ID Register, CIDR1

Component ID register.

RO register at offset 0xFF4.

Table 3-18 CIDR1 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:4]	CLASS	Component Class. Returns 0xE for a generic IP component.	RO	0x0F
[3:0]	PRMBL_1	-	RO	0x0

3.3.18 Component ID Register, CIDR2

Component ID register.

RO register at offset 0xFF8.

Table 3-19 CIDR2 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	PRMBL_2	-	RO	0x05

3.3.19 Component ID Register, CIDR3

Component ID register.

RO register at offset 0xFFC.

Table 3-20 CIDR3 register

Bits	Name	Description	Access	Reset
[31:8]	Reserved	-	RO, RAZ	0
[7:0]	PRMBL_3	-	RO	0xB1

3.4 Using the CG092

This section describes the sequences for enabling, disabling, or invalidating the cache in different modes.

The registers used are the CCR register and the SR register. For more information on the registers, see *Register descriptions* on page 3-4.

3.4.1 Auto power and auto invalidate mode

This section describes enable and disable sequences when in the auto power and auto invalidate mode.

Enabling the cache

To enable the cache:

1. Set the operation mode by setting the CCR.SET_MAN_POW bit to 0 and the CCR.SET_MAN_INV bit to 0.

   (Set register CCR to `0x00`, which is the reset value.)

2. Enable Cache by setting the CCR.EN bit to 1.

   (Set register CCR to `0x01`,)

3. Optionally wait until the SR.CS bits are equal to 2 which indicates that the cache is enabled.

Disabling the cache

To disable the cache:

1. Set the CCR.EN bit to 0.

   (Set register CCR to `0x00`.)

3.4.2 Manual power and auto invalidate mode

This section describes enable and disable sequences when in the manual power and auto invalidate mode.

Enabling the cache

To enable the cache:

1. Set operation mode by setting the CCR.SET_MAN_POW bit to 1 and the CCR.SET_MAN_INV bit to 0.

   (Set register CCR to `0x08`.)

2. Request power by setting the CCR.POW_REQ bit to 1.

   (Set CCR to `0x0C`.)

3. Wait until the SR.POW_STAT bits are equal to 1 which indicates that the power up has completed.

4. Enable the cache by setting the CCR.EN bit to 1.

   (Set register CCR to `0x0D`.)

Disabling the cache

To disable the cache:

1. Set the CCR.EN bit to 0.
   (Set register CCR to `0xC`,)
2. Optionally power down the SRAMs by setting the CCR.POW_REQ bit to 0.
   (Set register CCR to `0x18`.)

3.4.3 Manual power and manual invalidate mode

This section describes enable and disable sequences when in the manual power and manual invalidate mode.

Enabling the cache and invalidating the SRAM

To enable the cache:

1. Set operation mode by setting the CCR.SET_MAN_POW bit to 1 and bit CCR.SET_MAN_INV to 1.
   (Set register CCR to `0x18`.)
2. Request power by setting the CCR.POW_REQ bit to1.
   (Set register CCR to `0x1C`.)
3. Wait until the SR.POW_STAT bits are equal to 1 which indicates that power up has completed.
4. Request manual invalidation by setting the CCR.INV_REQ bit to 1.
   (Set register CCR to `0x1E`.)
5. Wait until the CCR.INV_REQ bits are equal to 0 which indicates that the invalidation has finished.
6. Enable the cache by setting the CCR.EN bit to 1.
   (Set register CCR to `0x1D`.)

Enabling the cache without invalidating the SRAM

To enable the cache:

1. Set operation mode by setting the CCR.SET_MAN_POW bit to 1 and the CCR.SET_MAN_INV bit to 1.
   (Set register CCR to `0x18`.)
2. Request power by setting the CCR.POW_REQ bit to1.
   (Set register CCR to `0x1C`.)
3. Wait until the SR.POW_STAT bits are equal to 1 which indicates that power up has completed.
4. Enable the cache by setting the CCR.EN bit to 1.
   (Set register CCR to `0x1D`.)

Disabling the cache

To disable the cache:

1. Disable the cache by setting the CCR.EN bit to 0.
   (Set register CCR to `0x1C`.)
2. Optionally power down the SRAMs by setting the CCR.POW_REQ bit to 0.
   (Set register CCR to `0x18`.)

Invalidating the cache

To invalidate the cache if it is enabled:

1. Disable the cache by setting the CCR.EN bit to 0.
   (Set register CCR to `0x1C`.)
2. Wait until the SR.CS bit is equal to 0 which indicates that the cache is disabled.
3. Request manual invalidation by setting bit CCR.INV_REQ to 1.
   (Set register CCR to `0x1E`.)
4. Wait until the CCR.INV_REQ bit is equal to 0 which indicates that the invalidation has finished.
5. Enable the cache by setting the CCR.EN bit to 1.
   (Set register CCR to `0x1D`.)

Appendix A
Signal Descriptions

This chapter lists signals for the CG092.

Note

Some signal widths are implementation defined.

It contains the following sections:

A.1 Clock and reset signals

The table below lists the clock and reset signals for the CG092 subsystem.

Table A-1 Clock and reset signals

Signal Name	Direction	Description
HCLK	Input	AHB Bus clock. This clock is used for all always on logic.
PCLKG	Input	Gated clock input for register interface (APB). It must be the same frequency and same phase as HCLK. Can be gated, when there are no APB activities. It is expected to run while APB interface PSEL signal is asserted.
HRESETn	Input	Active low asynchronous AHB reset. Reset all or a subset of the CG092 registers depending on the RESET_ALL_REGS parameter.

A.2 Cache SRAM power control interface

The table below lists the power-control signals for the CG092 subsystem.

Table A-2 Cache SRAM power control interface signals

Signal Name	Direction	Description
RAMPWRUPREQ	Output	SRAM power up request. Indicating an SRAM resource request from the cache. Port is registered and synchronous to **HCLK**.
RAMPWRUPACK	Input	SRAM powers up acknowledge. Indicates that the requested SRAM resource is available. Port is synchronized to **HCLK** clock domain.

A.3 Interrupt signals

The table below lists the interrupt signal for the CG092 subsystem.

Table A-3 Interrupt signal

Name	Direction	Width	Description
IRQ	Output	1	CG092 interrupt output

A.4 SRAM signals

The tables below list the SRAM signals for the CG092 subsystem. Way 0 is always present but way 1 is optional and depends on the configuration options selected by the SoC designer.

Table A-4 DATA SRAM signals for way 0

Name	Direction	Width	Description
RAMCLD0ADDR	Output	Implementation defined	Parametrized width data address bus.
RAMCLD0WE	Output	1	Write control for 128 bits (same cycle as address).
RAMCLD0RD	Output	4	Read control per word (same cycle as address).
RAMCLD0CS	Output	4	Chip select per word (same cycle as address).
RAMCLD0WDATA	Output	128	Write data (same cycle as address).
RAMCLD0RDATA	Input	128	Read data (1 cycle after address).

Table A-5 DATA SRAM signals for way 1

Name	Direction	Width	Description
RAMCLD1ADDR	Output	Implementation defined	Parametrized width data address bus.
RAMCLD1WE	Output	1	Write control for 128 bits (same cycle as address).
RAMCLD1RD	Output	4	Read control per word (same cycle as address).
RAMCLD1CS	Output	4	Chip select per word (same cycle as address).
RAMCLD1WDATA	Output	128	Write data (same cycle as address).
RAMCLD1RDATA	Input	128	Read data (1 cycle after address).

The tables below lists the TAG SRAM signals for the eFlash subsystem. Way 0 is always present but way 1 is optional and depends on the configuration options selected by the SoC designer.

Table A-6 TAG SRAM signals for way 0

Name	Direction	Width	Description
RAMTAG0ADDR	Output	Implementation defined	Parametrized width data address bus
RAMTAG0WE	Output	1	Write control (same cycle as address)
RAMTAG0RD	Output	1	Read control (same cycle as address)

Table A-6 TAG SRAM signals for way 0 (continued)

Name	Direction	Width	Description
RAMTAG0CS	Output	1	Chip select (same cycle as address)
RAMTAG0WDATA	Output	Implementation defined	Write data (same cycle as address)
RAMTAG0RDATA	Input	Implementation defined	Read data (1 cycle after address)

Table A-7 TAG SRAM signals for way 1

Name	Direction	Width	Description
RAMTAG1ADDR	Output	Implementation defined	Parametrized width data address bus
RAMTAG1WE	Output	1	Write control (same cycle as address)
RAMTAG1RD	Output	1	Read control (same cycle as address)
RAMTAG1CS	Output	1	Chip select (same cycle as address)
RAMTAG1WDATA	Output	Implementation defined	Write data (same cycle as address)
RAMTAG1RDATA	Input	Implementation defined	Read data (1 cycle after address)

A.5 Statistics signals

The table below lists the statistics signals for the CG092 subsystem.

Table A-8 Statistics signals

Name	Direction	Width	Description
CACHEMISS	Output	1	Active high single cycle pulses indicating that a cache miss happened during cache look up.
CACHEHIT	Output	1	Active high single cycle pulses indicating that a cache hit happened during cache look up

A.6 AHB slave signals

The table below lists the signals for the AHB slave interfaces.

Table A-9 AHB slave port signals

Name	Direction	Width	Description
HSELS	Input	1	Slave Select.
HADDRS	Input	AW	Address bus.
HTRANSS	Input	2	Transfer Type.
HWRITES	Input	1	Transfer Direction.
HSIZES	Input	3	Transfer Size.
HBURSTS	Input	3	Burst type.
HPROTS	Input	4	Protection Control.
HMASTERS	Input	4	Master Select.
HWDATAS	Input	32	Write Data.
HMASTLOCKS	Input	1	Locked Sequence.
HRDATAS	Output	32	Read data bus.
HREADYS	Output	1	HREADY feedback.
HREADYOUTS	Output	1	When high, the signal indicates that a transfer has finished on the bus. This signal can be driven low to extent a transfer.
HRESPS	Output	1	Transfer response.

A.7 APB slave signals

The table below lists the signals for the APB slave interface to the memory-mapped registers in the CG092.

Table A-10 APB slave port signals

Name	Direction	Width	Description
PSEL	Input	1	Slave select signal.
PENABLE	Input	1	Strobe to time all accesses. Used to indicate the second cycle of an APB transfer.
PADDR	Input	12 [11:0]	Address bus.
PWRITE	Input	1	APB transfer direction.
PWDATA	Input	32	32-bit write data bus.
PRDATA	Output	32	32-bit read data bus.
PREADY	Output	1	Driven LOW if extra wait states are required to complete the transfer.
PSLVERR	Output	1	Indicates SLVERR response.
PSTRB	Input	4	Write strobes. This signal indicates which byte lanes to update during a write transfer. There is one write strobe for each eight bits of the write data bus. **PSTRB[n]** corresponds to **PWDATA[(8n + 7):(8n)]**. Write strobes must not be active during a read transfer
PPROT	Input	3	Protection type.

A.8 AHB-Lite master signals

The table below lists the signals for the AHB master interface to the eFlash controller.

Table A-11 AHB master port signals

Name	Direction	Width	Description
HSELM	Output	1	Slave Select.
HADDRM	Output	AW	Address bus.
HTRANSM	Output	2	Transfer Type.
HWRITEM	Output	1	Transfer Direction.
HSIZEM	Output	3	Transfer Size.
HBURSTM	Output	3	Burst type.
HPROTM	Output	4	Protection Control.
HWDATAM	Output	128	Write Data.
HMASTLOCKM	Output	1	Locked Sequence.
HREADYM	Output	1	Transfer done.
HRDATAM	Input	128	Read data bus.
HREADYOUTM	Input	1	**HREADY** feedback.
HRESPM	Input	1	Transfer response.

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Issue A

Change	Location	Affects
First release	-	-