

CFG Pegasus Last Level Cache

***Technical Reference Manual***

**Version: PEGASUS-20.Q1**

**Revision: 0.0**

**Intel Confidential**

CFG Pegasus Last Level Cache Technical Reference Manual

**About This Document**

This document describes the architecture of Pegasus Last Level Cache and NocStudio commands for Pegasus. Using Pegasus, users can configure cache hierarchy to boost system performance.

**Audience**

This document is intended for users of NocStudio Orion and Gemini:

* SoC Architects
* NoC Architects
* NoC Designers

Prerequisite

Before proceeding, you should generally understand:

* Basics of Network on Chip technology
* AMBA interconnect standards

**Related Documents**

The following documents can be used as a reference to this document.

* CFG NocStudio Orion User Manual
* CFG NocStudio Gemini User Manual

Customer Support

For technical support about this product and general information, contact CFG Support.

Revision History

|  |  |  |
| --- | --- | --- |
| Revision | Date | Updates |
| 0.0 | Jan 29, 2020 | Initial Version |

Contents

[About This Document 2](#_Toc496630624)

[Audience 2](#_Toc496630625)

[Prerequisite 2](#_Toc496630626)

[Related Documents 2](#_Toc496630627)

[Customer Support 2](#_Toc496630628)

[1 Introduction 9](#_Toc496630629)

[2 Functional Description 11](#_Toc496630630)

[2.1 Relationship to Coherency 12](#_Toc496630631)

[2.2 Multiple LLCs 12](#_Toc496630632)

[2.3 Flexible Connectivity 13](#_Toc496630633)

[2.4 Logically Partitioned RAM Arrays 14](#_Toc496630634)

[2.5 Single-ported RAMs 15](#_Toc496630635)

[2.6 Flexible Timing of Arrays 15](#_Toc496630636)

[2.7 Banking of the RAMs 16](#_Toc496630637)

[2.8 Flexible Capacity and Associativity 17](#_Toc496630638)

[2.9 Way Groups and Data Banking: Relationship 17](#_Toc496630639)

[2.10 Scratchpad RAM Mode 18](#_Toc496630640)

[2.11 Replacement Policy 18](#_Toc496630641)

[2.12 Partial Reads and Writes 19](#_Toc496630642)

[2.13 Trust-Zone Bit 19](#_Toc496630643)

[2.14 Allocation control 19](#_Toc496630644)

[2.14.1 Way Allocation Controls 19](#_Toc496630645)

[2.14.2 Dynamic control of allocation behavior 20](#_Toc496630646)

[2.14.3 Static control of allocation behavior 21](#_Toc496630647)

[2.15 ECC Support 21](#_Toc496630648)

[2.16 Configurable Index and Tag Bits 21](#_Toc496630649)

[2.17 Control Sequences 22](#_Toc496630650)

[2.18 Cache Maintenance Instructions 22](#_Toc496630651)

[2.19 AXI Exclusive Functionality 23](#_Toc496630652)

[3 NocStudio Commands and Properties for LLC 25](#_Toc496630653)

[3.1 Adding a Last Level Cache 25](#_Toc496630654)

[3.2 LLC with 2 Slave Ports 25](#_Toc496630655)

[3.3 Configurable properties of the LLC 26](#_Toc496630656)

[3.4 Grouping LLC’s 28](#_Toc496630657)

[3.5 Configuring Pegasus as a Scratchpad RAM 29](#_Toc496630658)

[4 Programmers Model 31](#_Toc496630659)

[4.1 Transitions for Way Group State 31](#_Toc496630660)

[4.1.1 Cache Mode to Disabled Mode Transition 31](#_Toc496630661)

[4.1.2 RAM Mode to Disabled Mode 32](#_Toc496630662)

[4.1.3 Disabled Mode to Cache Mode 32](#_Toc496630663)

[4.1.4 Disabled Mode to RAM Mode 33](#_Toc496630664)

[4.2 Register-based Access of RAMs 33](#_Toc496630665)

[4.3 LLC Allocation Controls 34](#_Toc496630666)

[4.4 LLC Host Registers 34](#_Toc496630667)

[4.4.1 LLC\_ALLOC\_ARCACHE\_EN 34](#_Toc496630668)

[4.4.2 LLC\_ALLOC\_AWCACHE\_EN 35](#_Toc496630669)

[4.4.3 LLC\_ALLOC\_RD\_EN 35](#_Toc496630670)

[4.4.4 LLC\_ALLOC\_WR\_EN 36](#_Toc496630671)

[4.4.5 LLC\_CACHE\_WAY\_ENABLE 36](#_Toc496630672)

[4.4.6 LLC\_CLASS\_ALLOC 37](#_Toc496630673)

[4.4.7 LLC\_DATA\_INV\_CTL 38](#_Toc496630674)

[4.4.8 LLC\_ECC\_DATA\_INFO 38](#_Toc496630675)

[4.4.9 LLC\_ECC\_DISABLE 39](#_Toc496630676)

[4.4.10 LLC\_ECC\_TAG\_INFO 40](#_Toc496630677)

[4.4.11 LLC\_EVENT\_COUNTER 41](#_Toc496630678)

[4.4.12 LLC\_EVENT\_COUNTER\_MASK 41](#_Toc496630679)

[4.4.13 LLC\_GLOBAL\_ALLOC 42](#_Toc496630680)

[4.4.14 LLC\_INDIRECT\_RAM\_CONT 43](#_Toc496630681)

[4.4.15 LLC\_INDIRECT\_TRIGGER 43](#_Toc496630682)

[4.4.16 LLC\_INTERRUPT\_ERR 45](#_Toc496630683)

[4.4.17 LLC\_RAM\_ADDRESS\_BASE 46](#_Toc496630684)

[4.4.18 LLC\_INTERRUPT\_MASK 46](#_Toc496630685)

[4.4.19 LLC\_RAM\_WAY\_ENABLE 47](#_Toc496630686)

[4.4.20 LLC\_RAM\_WAY\_SECURE 48](#_Toc496630687)

[4.4.21 LLC\_TAG\_INV\_CTL 48](#_Toc496630688)

[4.4.22 LLC\_WAY\_FLUSH 49](#_Toc496630689)

Figures

[Figure 1: Last Level Cache sits behind coherency controller 10](#_Toc496630690)

[Figure 2: LLC has an ACE-lite and AXI port 11](#_Toc496630691)

[Figure 3: State Tracking in LLC Tags 11](#_Toc496630692)

[Figure 4: Multiple Parallel LLCs 12](#_Toc496630693)

[Figure 5: Dedicated caches connectivity 12](#_Toc496630694)

[Figure 6: Flexible Connectivity 13](#_Toc496630695)

[Figure 7: LLC has Controller, Tag Arrays and Data Arrays. Controller has separate interface to access the arrays. 14](#_Toc496630696)

[Figure 8: RAM access show flexible latency and repeat rate 15](#_Toc496630697)

[Figure 9: Cache with banked data arrays 15](#_Toc496630698)

[Figure 10: Ways and Banking are related 16](#_Toc496630699)

[Figure 11: Portions of cache can be modified to act as a scratchpad RAM 17](#_Toc496630700)

[Figure 12: Programmable allocation vectors 19](#_Toc496630701)

[Figure 13: Non-coherent DMA bypasses LLC 22](#_Toc496630702)

[Figure 14: Use of second slave port in LLC 24](#_Toc496630703)

[Figure 15: Each WayGroup supports 3 operation states 30](#_Toc496630704)

# Introduction

Pegasus is a highly customizable and configurable last level cache that can eliminate memory bottlenecks and boost overall system performance.

Pegasus can act as a memory bandwidth multiplier. Whenever a memory read or write hits a line that’s present in the cache, the access to memory can be avoided. This reduction in the accesses to memory reduces the utilized memory bandwidth, effectively increase the available memory bandwidth of the system. A cache hit rate of 50%, for instance, would allow 2X the number of memory requests by locally completing half of them and only sending the other half to memory.

Pegasus also increases system performance by reducing average latency. For every request that hits in the cache, the latency of going to memory can be eliminated and the request can be processed locally.

Each request that is completed by Pegasus also reduces dynamic power. Off-chip accesses to DRAM consume significant dynamic power. Cache hits eliminate this power consumption with a much lower power on-chip RAM access.

Pegasus allows architects significant control of their design by supporting a multitude of flexible cache hierarchies. It can be configured as a memory cache or as a coherent-only cache.

Pegasus can be configured as a coherent-only cache. Only coherent or IO coherent accesses will be sent to the cache in this configuration. This limits the benefits of the cache to these coherent accesses but can provide a lower latency and lower area solution. Non-coherent accesses go directly to memory, which can have a number of indirect benefits including support for larger than 64B requests.

If Pegasus is configured as a memory cache, all accesses to the specified address range will go to the cache and perform a cache lookup. This allows non-coherent accesses to gain the latency and bandwidth benefits of caching.

Pegasus as a coherent-only cache needs to support cache maintenance operations to push data to memory in order to facilitate communication between coherent and non-coherent devices. These are used when a memory space moves from one Shareability Domain to another. If Pegasus is configured as a memory cache, no cache maintenance is needed as requests all requests will see the same data.

The cache hierarchy connectivity can be created to have redundant paths so that the LLC can be entirely disabled and traffic can be routed directly to memory instead. This allows the entire LLC to be powered off and traffic to take a more direct route in the network.

Based on system requirements such as cache capacity and total coherent bandwidth, architects can add multiple instance of Pegasus, and customizes them before placing them in the interconnect. The benefits that Pegasus brings are:

* Lower latency by placing Pegasus where they are accessed the most.
* Reduce congestion by handling requests locally & using caches to reduce traffic to memory.
* Improve die utilization by placing Pegasus in empty die space