

CFG Gemini

***Technical Reference Manual***

**Version: GEMINI-20.Q1**

**Revision: 0.0**

**Intel Confidential**

CFG Gemini Technical Reference Manual

About This Document

This document is written for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) using Gemini Cache Coherent Interconnect.

Audience

This document is intended for users of NocStudio:

* NoC Architects
* NoC Designers
* SoC Architects

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 Gemini User Manual
* CFG Gemini IP Integration Spec

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](#_Toc496630034)

[Audience 2](#_Toc496630035)

[Prerequisite 2](#_Toc496630036)

[Related Documents 2](#_Toc496630037)

[Customer Support 2](#_Toc496630038)

[1 Introduction 16](#_Toc496630039)

[1.1 NetSpeed Gemini Overview 16](#_Toc496630040)

[1.2 Configurability Options 17](#_Toc496630041)

[1.3 Interfaces 17](#_Toc496630042)

[2 Functional Description: Coherent Components 22](#_Toc496630043)

[2.1 Coherency IP Components 22](#_Toc496630044)

[2.2 CCC (Cache Coherency Controller) 22](#_Toc496630045)

[2.3 IOCB (IO Coherency Bridge) 24](#_Toc496630046)

[2.4 DVM (Distributed Virtual Memory Controller) 26](#_Toc496630047)

[2.5 Directory Configurations 27](#_Toc496630048)

[2.6 Snoop Filter Support 37](#_Toc496630049)

[2.7 ACE Master Cache Capacity 37](#_Toc496630050)

[2.8 Speculative Fetch Control 38](#_Toc496630051)

[2.9 Fast Tap Support 38](#_Toc496630052)

[2.10 CD Channel configuration 39](#_Toc496630053)

[2.11 Transaction handling 40](#_Toc496630054)

[2.12 Deadlock Avoidance Unit 43](#_Toc496630055)

[2.13 Coherency Connect/Disconnect 47](#_Toc496630056)

[2.14 Invalidation Engine 48](#_Toc496630057)

[2.15 QoS overrides for derivative coherent transactions 48](#_Toc496630058)

[3 Cache Hierarchy Configurability 50](#_Toc496630059)

[3.1 Asymmetric Hierarchies 50](#_Toc496630060)

[3.2 Different Ranges for the same Slave 51](#_Toc496630061)

[3.3 Adding Slices for CCC or LLC 51](#_Toc496630062)

[3.4 Different Slicing between CCC and LLC 52](#_Toc496630063)

[3.5 Different Number of CCCs and LLCs 53](#_Toc496630064)

[3.6 Combining These Design Choices 53](#_Toc496630065)

[3.7 Non-Coherent Access Point 54](#_Toc496630066)

[4 Functional Description: Non-Coherent Components 56](#_Toc496630067)

[4.1 Bridging from Host to NoC 56](#_Toc496630068)

[4.2 Address Maps and Configurability Options 69](#_Toc496630069)

[4.3 AID Handling and transaction ordering 72](#_Toc496630070)

[4.4 Splitting of AMBA transactions 73](#_Toc496630071)

[4.5 Width Conversion 74](#_Toc496630072)

[4.6 Virtual slave interface 74](#_Toc496630073)

[4.7 Interrupt 77](#_Toc496630074)

[4.8 Restrictions 78](#_Toc496630075)

[5 Functional Description: System Interconnect 79](#_Toc496630076)

[5.1 NoC Topology 79](#_Toc496630077)

[5.2 Routers 82](#_Toc496630078)

[5.3 Information Transport in the NoC 83](#_Toc496630079)

[5.4 Clocking 88](#_Toc496630080)

[6 Low Power Support 93](#_Toc496630081)

[6.1 NetSpeed Power Management Architecture 93](#_Toc496630082)

[6.2 Power Domains and Relationships to Clock and Voltage Domains 94](#_Toc496630083)

[6.3 Low Power Signaling Interface Between PMU and NSPS 95](#_Toc496630084)

[6.4 Fencing and Draining 97](#_Toc496630085)

[6.5 Low Power Signals 100](#_Toc496630086)

[6.6 Bridge Logic in Host Power Domain 101](#_Toc496630087)

[6.7 AHB Lite Master Bridge (AHBLM) 102](#_Toc496630088)

[6.8 Regbus Master and Tunnel 102](#_Toc496630089)

[6.9 Shared Interface Bridge 102](#_Toc496630090)

[6.10 Always On Power Domains 102](#_Toc496630091)

[6.11 Restrictions 103](#_Toc496630092)

[7 Low Power Support: Coherency Components 104](#_Toc496630093)

[7.1 Coherent IP and their Master Ports must share a power domain 104](#_Toc496630094)

[7.2 Coherency Connect/Disconnect 104](#_Toc496630095)

[7.3 DVM Power Domain as Decode Error 105](#_Toc496630096)

[7.4 IOCB Power Down 105](#_Toc496630097)

[7.5 CCC Power Down 105](#_Toc496630098)

[7.6 LLC Power Down 105](#_Toc496630099)

[7.7 Restrictions 106](#_Toc496630100)

[8 Deadlock Avoidance 107](#_Toc496630101)

[8.1 Quick Primer on Deadlocks 107](#_Toc496630102)

[8.2 Constructing Deadlock-Free Interconnects 109](#_Toc496630103)

[8.3 Protection against Slave dependency behavior 110](#_Toc496630104)

[9 Security 112](#_Toc496630105)

[9.1 Connectivity-Based Firewalls 112](#_Toc496630106)

[9.2 Address Range Connectivity 112](#_Toc496630107)

[9.3 Propagation of TrustZone Bit 112](#_Toc496630108)

[9.4 Selective Traffic Filtering 113](#_Toc496630109)

[9.5 Programmable Address Map 114](#_Toc496630110)

[9.6 Interface Overrides 114](#_Toc496630111)

[9.7 User Bits 115](#_Toc496630112)

[10 Quality of Service Support 116](#_Toc496630113)

[10.1 Traffic Classes 117](#_Toc496630114)

[10.2 Strict priority based allocation 119](#_Toc496630115)

[10.3 Weighted bandwidth allocation 123](#_Toc496630116)

[10.4 Rate Limiting Hosts 126](#_Toc496630117)

[10.5 Dynamic Priority Support for Isochronous Traffic 132](#_Toc496630118)

[11 NoC Serviceability: Regbus Layer 133](#_Toc496630119)

[11.1 The Register Bus 133](#_Toc496630120)

[11.2 NoC Registers 138](#_Toc496630121)

[11.3 Error Responses To Register Accesses 139](#_Toc496630122)

[11.4 User Register Bus Access 140](#_Toc496630123)

[11.5 Register Bus Master Interface 141](#_Toc496630124)

[11.6 Expected Usage of Register Bus Master 144](#_Toc496630125)

[11.7 Ring Slave to Host Interface 144](#_Toc496630126)

[11.8 Atomic Operations 145](#_Toc496630127)

[11.9 Restrictions 148](#_Toc496630128)

[12 Safety and Reliability 150](#_Toc496630129)

[12.1 Transport error detection and correction 150](#_Toc496630130)

[12.2 End to end transport error checking 150](#_Toc496630131)

[12.3 Hop to hop Error checking 152](#_Toc496630132)

[12.4 Interface Parity 153](#_Toc496630133)

[12.5 Configuring Error checking 157](#_Toc496630134)

[12.6 Error reporting 158](#_Toc496630135)

[12.7 Configuration and Status register protection 158](#_Toc496630136)

[13 Programmers Model 159](#_Toc496630137)

[13.1 Router Registers 159](#_Toc496630138)

[13.2 Streaming Bridge registers 171](#_Toc496630139)

[13.3 AMBA Registers 185](#_Toc496630140)

[13.4 Regbus Registers 211](#_Toc496630141)

[13.5 Protocol Converter Registers 211](#_Toc496630142)

[13.6 DVM Host Registers 218](#_Toc496630143)

[13.7 CCC Host Registers 222](#_Toc496630144)

[13.8 LLC Host Registers 234](#_Toc496630145)

[14 Appendix A: AMBA Protocol Support 251](#_Toc496630146)

[14.1 ACE / AXI4 Feature Adoption 251](#_Toc496630147)

[14.2 AMBA Signal Adoption 253](#_Toc496630148)

[14.3 AXI4-Lite Feature Adoption 256](#_Toc496630149)

[14.4 AXI3 Feature Adoption 257](#_Toc496630150)

[14.5 AHB-Lite Feature Adoption 257](#_Toc496630151)

[14.6 APB Feature Adoption 258](#_Toc496630152)

Figures

[Figure 1: NetSpeed Gemini Overview 13](file:///C:\SVN\trunk__doc\release_docs\Technical%20Reference%20Manual\Master%20Docs\Unwrapped\NetSpeed%20Gemini%20Technical%20Reference%20Manual.docx#_Toc496630153)

[Figure 2: ACE Port 15](#_Toc496630154)

[Figure 3: ACE-lite 16](#_Toc496630155)

[Figure 4: ACE-lite+DVM 16](#_Toc496630156)

[Figure 5: CCC Port Diagram 19](#_Toc496630157)

[Figure 6: IOCB Port Diagram 21](#_Toc496630158)

[Figure 7: Multi-IOCB diagram 22](#_Toc496630159)

[Figure 8: DVM module port diagram 23](#_Toc496630160)

[Figure 9: Directory organized as two tag arrays 29](#_Toc496630161)

[Figure 10: Example RAM latency == 1 30](#_Toc496630162)

[Figure 11: Example of 2 cycle latency, 1/2 cycle bandwidth 31](#_Toc496630163)

[Figure 12: Example directory RAM with 3 cycle latency, 1/2 cycle bandwidth 31](#_Toc496630164)

[Figure 13: Fast Tap 36](#_Toc496630165)

[Figure 14: ACE/PCI Deadlock 40](#_Toc496630166)

[Figure 15: Deadlock Avoidance Unit in the System 41](#_Toc496630167)

[Figure 16: RAM based write queue 43](#_Toc496630168)

[Figure 17: Coherent and Non-Coherent Devices 47](#_Toc496630169)

[Figure 18: Asymmetric Cache Hierarchy 47](#_Toc496630170)

[Figure 19: Different ranges of to the same memory controller 48](#_Toc496630171)

[Figure 20: Uniform sliced address space 49](#_Toc496630172)

[Figure 21: Different Slicing between CCC and LLC 49](#_Toc496630173)

[Figure 22: Different number of CCCs and LLCs 50](#_Toc496630174)

[Figure 23: Example Coherency Cache hierarchy 51](#_Toc496630175)

[Figure 24: Example Memory Cache hierarchy 52](#_Toc496630176)

[Figure 25. AXI4 Master Bridge 54](#_Toc496630177)

[Figure 26. AXI4 Slave Bridge 57](#_Toc496630178)

[Figure 27. AXI to AXI4-Lite Bridge 59](#_Toc496630179)

[Figure 28. AXI to APB Bridge 60](#_Toc496630180)

[Figure 29. AHB-Lite to AXI Master Bridge 61](#_Toc496630181)

[Figure 30. AXI to AHB-Lite Bridge 62](#_Toc496630182)

[Figure 31 Shared Interface Bridge 63](#_Toc496630183)

[Figure 32: Reorder Bridge NoC Component 63](#_Toc496630184)

[Figure 33: Master with small address space can access regions in a much larger address space 67](#_Toc496630185)

[Figure 34: Multiple system address ranges can be compressed to appear as a contiguous slave region 68](#_Toc496630186)

[Figure 35: Simple XOR hash function 68](#_Toc496630187)

[Figure 36: AR interface additions 72](#_Toc496630188)

[Figure 37 AW-W interface additions 74](#_Toc496630189)

[Figure 38. Bridge and Router Functions in the NoC 76](#_Toc496630190)

[Figure 39. A 4x4 Homogeneous Grid or Mesh Interconnect 77](#_Toc496630191)

[Figure 40 A 4x4 heterogeneous grid or mesh interconnect 78](#_Toc496630192)

[Figure 41. Schematic Symbol of a NocStudio RTL-Library Router Component 80](#_Toc496630193)

[Figure 42. Information Transport in the NoC 81](#_Toc496630194)

[Figure 43. NoC Packet Organized in Two Different Flit Sizes 82](#_Toc496630195)

[Figure 44. Merging and Dividing Flits with Various Channel Widths 84](#_Toc496630196)

[Figure 45. Combinations of Flit Merging and Division at Router Ports 84](#_Toc496630197)

[Figure 46. Multiple Clock Domains and Clock Crossings 85](#_Toc496630198)

[Figure 47 In-Link Domain crossing structure 87](#_Toc496630199)

[Figure 48. Clock Gating as Implemented in NetSpeed NoC 88](#_Toc496630200)

[Figure 49 NetSpeed Power Management Architecture 90](#_Toc496630201)

[Figure 50 Power Down Waveform Sequence 93](#_Toc496630202)

[Figure 51 Power Up Sequence Waveforms 94](#_Toc496630203)

[Figure 52 QDENY Waveform Sequence 96](#_Toc496630204)

[Figure 53. Simple Deadlock Graph with Two Resources 105](#_Toc496630205)

[Figure 54. Simple Interconnect with Deadlock 105](#_Toc496630206)

[Figure 55 - System Level Dependency Graph 107](#_Toc496630207)

[Figure 56: System with 2 masters and one slave 117](#_Toc496630208)

[Figure 57: Class/Priority mapping 118](#_Toc496630209)

[Figure 58: Snapshot of Performance Simulation with same class, same priority 118](#_Toc496630210)

[Figure 59:Snapshot of performance simulation with 2 classes, different priority 119](#_Toc496630211)

[Figure 60:Snapshot of performance simulation with 2 classes, same priority 119](#_Toc496630212)

[Figure 61: 3 master, 1 slave system 121](#_Toc496630213)

[Figure 62: Performance results with ratios 3:2:1 122](#_Toc496630214)

[Figure 63 - Token Count increases at specified rate. Packet transmit decrements count. 125](#_Toc496630215)

[Figure 64 - Token Bucket 126](#_Toc496630216)

[Figure 65. Regbus Master Bridge 131](#_Toc496630217)

[Figure 66. Regbus Layer Communication 132](#_Toc496630218)

[Figure 67. Regbus Tunnel Connects Primary NoC Layer to Regbus Layer 133](#_Toc496630219)

[Figure 68: Regbus Address Map 135](#_Toc496630220)

[Figure 69: Register bus master bridge 138](#_Toc496630221)

[Figure 70: Waveform showing read requests and responses at the register bus master interface 141](#_Toc496630222)

[Figure 71: Waveform showing write requests and responses at the register bus master interface 141](#_Toc496630223)

[Figure 72 : Waveform showing ring slave read requests and responses (4B and 8B) 145](#_Toc496630224)

[Figure 73 : Waveform showing ring slave write requests and responses (4B and 8B) 145](#_Toc496630225)

[Figure 74 Coherency Connect Request Register (SYSCOREQ). 214](#_Toc496630226)

[Figure 75 Coherency Connect Request Status Register (SYSCOACK). 215](#_Toc496630227)

Tables

[Table 1: Coherency Port Protocol Restrictions 20](#_Toc496630228)

[Table 2: Inner and Outer Shareable Domain Visibility 28](#_Toc496630229)

[Table 3 NSPS to PMU Interface 100](#_Toc496630230)

[Table 4 PMU to Power Domain Interface 101](#_Toc496630231)

[Table 5 - Rate limiter Configuration Register 130](#_Toc496630232)

[Table 6: Register attribute table 138](#_Toc496630233)

[Table 7: Register bit attribute table 139](#_Toc496630234)

[Table 8: Response table for NoC Register Accesses 139](#_Toc496630235)

[Table 9: Response table for User Register Bus Accesses 140](#_Toc496630236)

[Table 10: Register Bus Master Interface signals 142](#_Toc496630237)

[Table 11: Register slave to host interface 145](#_Toc496630238)

[Table 12 ID register. 160](#_Toc496630239)

[Table 13 RPERR register. 161](#_Toc496630240)

[Table 14 RPERRM register. 162](#_Toc496630241)

[Table 15 RE register. 163](#_Toc496630242)

[Table 16 REC register. 163](#_Toc496630243)

[Table 17 RECC register. 164](#_Toc496630244)

[Table 18 REM register. 165](#_Toc496630245)

[Table 19 RIVCS register. 167](#_Toc496630246)

[Table 20 ROEC register. 168](#_Toc496630247)

[Table 21 ROECC register. 169](#_Toc496630248)

[Table 22 ROVCS register. 170](#_Toc496630249)

[Table 23 BRPERR0 register. 171](#_Toc496630250)

[Table 24 BRPERR1 register. 172](#_Toc496630251)

[Table 25 BRPERRM0 register. 173](#_Toc496630252)

[Table 26BRPERRM1 register. 173](#_Toc496630253)

[Table 27 BRS register. 174](#_Toc496630254)

[Table 28 BRUS register. 175](#_Toc496630255)

[Table 29 BTPERR register. 176](#_Toc496630256)

[Table 30 BTPERRM register. 177](#_Toc496630257)

[Table 31 BTRL register. 177](#_Toc496630258)

[Table 32 BTUS\_0 register. 178](#_Toc496630259)

[Table 33 BTUS\_1 register. 180](#_Toc496630260)

[Table 34 RXE register. 181](#_Toc496630261)

[Table 35 RXEM register. 182](#_Toc496630262)

[Table 36 RXID register. 182](#_Toc496630263)

[Table 37 TXE register. 184](#_Toc496630264)

[Table 38 TXEM register. 184](#_Toc496630265)

[Table 39 TXID register. 185](#_Toc496630266)

[Table 40 AM\_ADBASE register. 187](#_Toc496630267)

[Table 41 AM\_ADMASK register. 187](#_Toc496630268)

[Table 42 AM\_ADRELOCSLV register. 188](#_Toc496630269)

[Table 43 AM\_ADRELOCSYS register. 188](#_Toc496630270)

[Table 44 AM\_AROVRD register. 189](#_Toc496630271)

[Table 45 AM\_AWOVRD register. 190](#_Toc496630272)

[Table 46 AM\_BRIDGE\_ID register. 190](#_Toc496630273)

[Table 47 AM\_CADDR register. 191](#_Toc496630274)

[Table 48 AM\_CADDRMSK register. 191](#_Toc496630275)

[Table 49 AM\_CCMD0 register. 192](#_Toc496630276)

[Table 50 AM\_CCMD1 register. 193](#_Toc496630277)

[Table 51 AM\_CCMDMSK1 register. 194](#_Toc496630278)

[Table 52 AM\_CFG register. 195](#_Toc496630279)

[Table 53 AM\_CNTR0 register. 195](#_Toc496630280)

[Table 54 AM\_CNTR1 register. 196](#_Toc496630281)

[Table 55 AM\_ERA register. 196](#_Toc496630282)

[Table 56 AM\_ERR register. 198](#_Toc496630283)

[Table 57 AM\_EWA register. 198](#_Toc496630284)

[Table 58 AM\_HASH\_FUNC register. 199](#_Toc496630285)

[Table 59 AM\_INTM register. 200](#_Toc496630286)

[Table 60AM\_LATNUM0 register. 201](#_Toc496630287)

[Table 61 AM\_LATNUM1 register. 201](#_Toc496630288)

[Table 62 AM\_OSSLV register. 202](#_Toc496630289)

[Table 63 AM\_STS register. 203](#_Toc496630290)

[Table 64 AM\_TOCFG register. 204](#_Toc496630291)

[Table 65 AM\_TOSLVID register. 204](#_Toc496630292)

[Table 66 AS\_AROVRD register. 205](#_Toc496630293)

[Table 67 AS\_AWOVRD register. 206](#_Toc496630294)

[Table 68 AS\_BRIDGE\_ID register. 206](#_Toc496630295)

[Table 69 AS\_CCMD register. 206](#_Toc496630296)

[Table 70 AS\_CCMDMSK register. 207](#_Toc496630297)

[Table 71 AS\_CNTR register. 207](#_Toc496630298)

[Table 72 AS\_ERR register. 208](#_Toc496630299)

[Table 73 AS\_INTM register. 209](#_Toc496630300)

[Table 74 AS\_STS register. 210](#_Toc496630301)

[Table 75 SYSCOACK\_REG register. 210](#_Toc496630302)

[Table 76 SYSCOREQ\_REG register. 211](#_Toc496630303)

[Table 77 AM\_NOCVER\_ID register. 211](#_Toc496630304)

[Table 78 AHBM\_CTL register. 212](#_Toc496630305)

[Table 79 AHBM\_IM register. 212](#_Toc496630306)

[Table 80 AHBM\_STS register. 213](#_Toc496630307)

[Table 81 AHBS\_CTL register. 213](#_Toc496630308)

[Table 82 AHBS\_IM register. 213](#_Toc496630309)

[Table 83 AHBS\_STS register. 214](#_Toc496630310)

[Table 84 APBSLV\_BRIDGE\_ID register. 214](#_Toc496630311)

[Table 85 APBSLV\_BRIDGE\_VERSION register. 215](#_Toc496630312)

[Table 86 APBSLV\_SLVS\_SLEEP\_STATUS register. 216](#_Toc496630313)

[Table 87 DVM\_ACTIVE\_VECTOR register. 219](#_Toc496630314)

[Table 88 DVM\_AGENT\_DISABLE\_STATUS register. 220](#_Toc496630315)

[Table 89 DVM\_FAULT\_LOG register. 221](#_Toc496630316)

[Table 90 DVM\_STS register. 221](#_Toc496630317)

[Table 91 CCC\_ACTIVE\_VECTOR\_0 register. 222](#_Toc496630318)

[Table 92 CCC\_AGENT\_DISABLE\_STATUS register. 223](#_Toc496630319)

[Table 93 CCC\_DIRECTORY\_INV register. 224](#_Toc496630320)

[Table 94 CCC\_ECC\_DISABLE register. 225](#_Toc496630321)

[Table 95 CCC\_ECC\_INFO register. 226](#_Toc496630322)

[Table 96 CCC\_EVENT\_COUNTER\_MASK register. 227](#_Toc496630323)

[Table 97 CCC\_EVENT\_COUNTER\_VALUE register. 227](#_Toc496630324)

[Table 98 CCC\_HASH\_BYPASS register. 228](#_Toc496630325)

[Table 99 CCC\_INDIRECT\_ACCESS\_TRIG register. 230](#_Toc496630326)

[Table 100 CCC\_INDIRECT\_RAM\_CONT register. 230](#_Toc496630327)

[Table 101 CCC\_INTERRUPT\_ERR register. 231](#_Toc496630328)

[Table 102 CCC\_INTERRUPT\_MASK register. 231](#_Toc496630329)

[Table 103 CCC\_LLC\_CONTROL register. 232](#_Toc496630330)

[Table 104CCC\_QOS\_OVERRIDES register. 233](#_Toc496630331)

[Table 105 CCC\_SPEC\_FETCH register. 234](#_Toc496630332)

[Table 106 LLC\_ALLOC\_ARCACHE\_EN register. 235](#_Toc496630333)

[Table 107 LLC\_ALLOC\_AWCACHE\_EN register. 235](#_Toc496630334)

[Table 108LLC\_ALLOC\_RD\_EN register. 236](#_Toc496630335)

[Table 109 LLC\_ALLOC\_WR\_EN register. 236](#_Toc496630336)

[Table 110 LLC\_CACHE\_WAY\_ENABLE register. 237](#_Toc496630337)

[Table 111 LLC\_CLASS\_ALLOC register. 238](#_Toc496630338)

[Table 112 LLC\_DATA\_INV\_CTL register. 238](#_Toc496630339)

[Table 113 LLC\_ECC\_DATA\_INFO register. 239](#_Toc496630340)

[Table 114 LLC\_ECC\_DISABLE register. 240](#_Toc496630341)

[Table 115 LLC\_ECC\_TAG\_INFO register. 241](#_Toc496630342)

[Table 116 LLC\_EVENT\_COUNTER register. 241](#_Toc496630343)

[Table 117 LLC\_EVENT\_COUNTER\_MASK register. 242](#_Toc496630344)

[Table 118 LLC\_GLOBAL\_ALLOC register. 243](#_Toc496630345)

[Table 119 LLC\_INDIRECT\_RAM\_CONT register. 243](#_Toc496630346)

[Table 120 LLC\_INDIRECT\_TRIGGER register. 245](#_Toc496630347)

[Table 121 LLC\_INTERRUPT\_ERR register. 246](#_Toc496630348)

[Table 122 LLC\_RAM\_ADDRESS\_BASE register. 246](#_Toc496630349)

[Table 123 LLC\_INTERRUPT\_MASK register 247](#_Toc496630350)

[Table 124 LLC\_RAM\_WAY\_ENABLE register. 248](#_Toc496630351)

[Table 125 LLC\_RAM\_WAY\_SECURE register. 248](#_Toc496630352)

[Table 126 LLC\_TAG\_INV\_CTL register. 249](#_Toc496630353)

[Table 127 LLC\_WAY\_FLUSH register. 250](#_Toc496630354)

# Introduction

## CFG Gemini Overview

CFG Gemini is a cache-coherent, high-performance Network-on-chip (NoC) IP that is used for rapidly designing and analyzing highly efficient and scalable cache-coherent interconnects for a wide variety of SoCs. To quickly produce efficient high-performance cache-coherent NoC IPs, Gemini uses a requirements-driven design approach and an innovative directory-based design. Using Gemini, SoC architects can connect anywhere from 1 to 64 fully cache-coherent CPU clusters, GPUs or other compute units. It also supports 1 to 200 I/O coherent and non-coherent agents. Gemini is built upon following the following fundamental design principles.

### Requirements driven approach

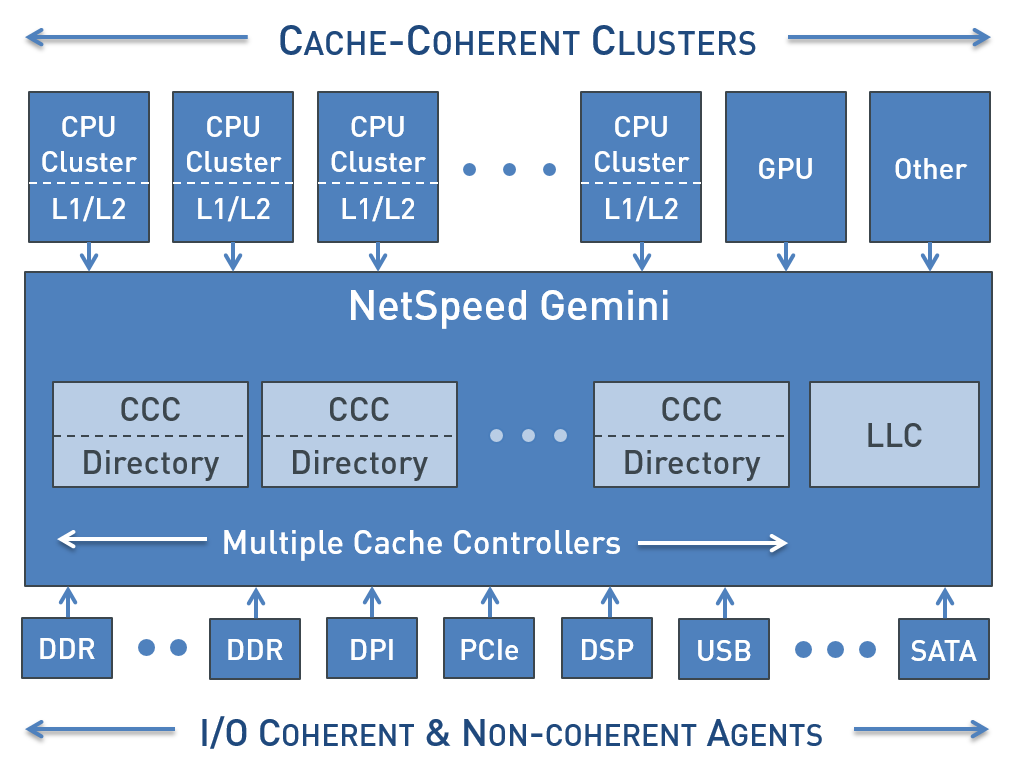
Gemini is based on a highly distributed architecture, where both the interconnect fabric and the coherency components can be scaled independently. Based on system requirements such as cache capacity and total coherent bandwidth, coherency components are added, customized and placed in the interconnect. The fabric itself can be designed and customized based on fine-grained requirements such as total and per-flow system bandwidth and chip layout. Gemini uses an algorithmic directory whose capacity scales linearly with number of caches and limits all unnecessary snoop traffic giving power efficiency and faster responses. Furthermore, the directory is partitioned in several slices and distributed across the fabric to provide unlimited scalability and flexibility. Another unique advantage of Gemini is that it can support both coherent and non-coherent traffic in the same design, which avoids the need to build multiple interconnects and inefficient bridges to connect them. Figure 1 illustrates the logical picture of a Gemini interconnect, where a variety of non-coherent and coherent masters and slaves are interconnected through a fabric which includes a distributed cache coherency directory and last level cache (LLC).

Figure 1: CFG Gemini Overview

### Physically aware latency optimized design

Gemini design is physically aware of the layout of the on-chip system components producing an interconnect topology that is customized for the SoC layout. Being physically aware ensures that wiring congestions does not occur late in the design cycle and appropriate number of buffers and pipeline stages are present at various fabric channels to enable smooth backend design. Furthermore, Gemini’s IP components can be partitioned, distributed and co-located near the coherent masters that typically utilize them, providing improved latency and power efficiency. Latency sensitive traffic can use dedicated connections to reduce arbitration and congestions, and 16 levels of QoS are supported for fine-grained bandwidth allocation and prioritization. Based on the system traffic specification and SoC physical layout, Gemini NoC topology, coherency and fabric components, and their placements are automatically computed using machine-learning and graph theory algorithms to optimize the design for area & power.

## Configurability Options

CFG Gemini provides user configurability and flexibility across multiple design dimensions. In addition to providing flexibility of number of ports, coherency participation, etc., significant configurability is also provided to the architect in defining the cache hierarchy and asymmetric traffic based on the system characteristics such as hit rates. Gemini is configured and optimized using NocStudio - an architecture exploration platform based on machine learning and graph theory algorithms. Using NocStudio, architects can design interconnects with high level specification and tradeoff power, performance and area with real-time feedback on SoC bandwidth and latency, coherency performance etc. from NocStudio SoC performance simulator.

## Interfaces

### Ports and Port Protocols

Since Gemini is an extension of Orion, it supports all of the Orion functionality and port types. This means AXI4, AXI3, AXI-lite, AHB and APB port types can all be added to a Gemini configuration.

In addition to non-coherent port types, Gemini support 3 additional port protocols. These are ACE, ACE-lite, and ACE-lite+DVM.

### ACE

The ACE interface protocol is used for fully cache coherent agents, like a CPU or a coherent GPU. ACE includes read and write channels, and their corresponding response channels. It also has channels for snoops and snoop responses. Finally, it has additional signals called RACK and WACK used as part of the coherency protocol to indicate response arrival.



Figure 2: ACE Port

The figure above shows the ACE port, including the various channels.

Note that the CD channel, which is used for snoop data response, is an optional channel. Gemini allows an ACE port to be configured with or without the CD channel.

### ACE-lite

The second standard port protocol in the coherency space is the ACE-lite port. This port is designed for agents without caches that want to perform reads and writes to the coherent address space. WriteUnique (or WriteLineUnique) allows writes to perform coherent snoops and invalidation/flushes of a cache line before writing to the line. This ensures writes will be seen by cache coherent devices since they’ll have to read the new cache line value from memory. A ReadOnce command reads a copy of a line but goes through the coherency mechanism to make sure that it sees the most recent copy.



Figure 3: ACE-lite

The figure above shows the ACE-lite port interfaces. There is no snoop or snoop response channels, and no RACK or WACK signals.

### ACE-lite+DVM

Unlike the ACE and ACE-lite channels, there is no explicit reference to the ACE-lite+DVM port protocol within the AXI4/ACE specification. This hybrid protocol looks similar to an ACE interface without the CD channel but behaves differently.

This protocol is intended to be used by agents that have no caches but do have a built-in MMU. The MMU-400 or MMU-500 products are examples of this protocol.



Figure 4: ACE-lite+DVM

As shown in the figure, this agent has the typical read and write paths of an ACE-lite agent, IO coherent reads and writes. It also has a snoop and snoop response channel (AC and CR), which are used only for DVM snoops since the agent never keeps a cached copy of a line. Since the DVM protocol does not use a CD bus (it never sends data with the snoop response), it is not included in this interface. It also has no need for the RACK or WACK signaling.

### ACE-lite Converted Bridges

Some components use older non-coherent protocols, like AXI 4. While the components are built as non-coherent, they may want to participate in the coherency protocol using IO coherency. Gemini supports conversion from some of these legacy protocols to ACE-lite. Eligible requests will convert simple reads and writes into ReadOnce and WriteUnique requests.

The following protocols can be converted to ACE-lite:

1. AHB-lite Master
2. AXI 4 Master
3. AXI 3 Master

The conversion hardware is built into the Gemini bridges and can be enabled during NoC construction using NocStudio. A bridge property called *cc\_acelite\_conversion* will convert the bridge to Ace-lite.

Bridges setup for ACE-lite conversion allow both IO-coherent and non-coherent requests to be made. For any address range that does not support coherency, the request will stay as ReadNoSnoop or WriteNoSnoop, with AxDOMAIN set to System Shareable or Non-Shareable, depending on the AxCACHE bits. When AxCACHE[3:2]==2’b00, the request is mapped to System Shareable. Otherwise, it is mapped to Non-Shareable.

For coherency ranges, the AxCACHE bits control whether a request will participate in the coherent protocol. If AxCACHE bits indicate non-cacheable (AxCACHE[3:2]==2’b00), the requests will be sent as WriteNoSnoop and ReadNoSnoop with AxDOMAIN = System Shareable.

If AxCACHE[3:2] indicates a cacheable space, the request will be converted to a ReadOnce or WriteUnique, with AxDOMAIN equal to Outer Shareable.

Note that in the non-converted bridges, these would be sent as Non-Shareable domain and WriteNoSnoop or ReadNoSnoop.

One exception to this conversion is when a request is sent with AxLOCK==1. Since ACE doesn’t support the Exclusive functionality for ACE-lite, this request will be issued as ReadNoSnoop or WriteNoSnoop with either System or Non-Shareable domain.

### Master and Slave ports

The port protocols described above (ACE, ACE-lite, ACE-lite+DVM, and legacy->ACE-lite converted) cannot be used uniformly as master or slave ports. The following table describes the limitations.

Table 1: Coherency Port Protocol Restrictions

|  |  |  |  |
| --- | --- | --- | --- |
| Protocol | Master Port Type | Slave Port Type | Description |
| ACE | Yes | No | ACE port is used by coherent masters. ACE slave interface only exists on interconnect IP, so agents cannot use this as a slave port type. |
| ACE-lite | Yes | Yes | An ACE-lite slave can accept barriers, cache maintenance operations, as well as reads and writes. |
| ACE-lite+DVM | Yes | No | Like the ACE port, only a master can utilize this port protocol. |
| ACE-lite conversion | Yes | No | Only masters can convert to ACE-lite. |