GN412x PCI Express

Family Reference Manual

Revision History

Version	ECR	Date	Changes and Modifications
0	151928	June 2009	New document.

Contents

1. Introduction 11
    1.1 Features 11
    1.2 Live on Power-up 13
    1.3 FPGA On-the-Fly Configuration Loader 13
    1.4 Local Bus Interface 13
    1.5 Virtual Channel Support 14
    1.6 PCI Express Application Layer 14
    1.7 Interrupt Controller 14
    1.8 2-Wire Serial Controller 14
    1.9 How to Use this Family Reference Manual 15
        1.9.1 Family Reference Manual Conventions 16
    1.10 Getting Help from Gennum 17
    1.11 Getting Answers to PCI Express Related Questions 17
2. Overview 18
    2.1 GN412x Signals 19
3. Initialization 20
    3.1 Reset Initialization 20
        3.1.1 Initializing the Internal Registers 21
        3.1.2 Selecting an Initialization Mode 21
        3.1.3 Initialization from a 2-Wire EEPROM 22
        3.1.4 Initialization Using a Local Processor 25
        3.1.5 Initialization Using the PCIe Configuration Space 26
    3.2 OS Initialization 26
    3.3 PCI Express Configuration Space Initialization 26
    3.4 Clock Configuration 27
        3.4.1 PCI Express Clock 27
        3.4.2 Local Clock 27
4. PCI Express Link 31
    4.1 Physical Interface 31
        4.1.1 Signal Polarity Inversion 31
        4.1.2 PCI Express CEM Signals 32
        4.1.3 PCI Express PHY Control 32
    4.2 PCI Express Traffic Routing 32

4.3 Configuration Space 33
4.3.1 Type 0 PCI Configuration Space 34
4.3.2 PCI Express Capability 36
4.3.3 Power Management Capability 36
4.3.4 Device Serial Number Capability 37
4.3.5 Virtual Channel Capability 37
4.3.6 MSI Capability 38
4.4 Bridge Operation 39
4.4.1 Flow Control 39
5. Local Bus Interface 40
5.1 Local Bus Signals 40
5.1.1 L2P DDR Source Synchronous Signals 42
5.1.2 P2L DDR Source Synchronous Signals 42
5.1.3 Signal Naming Conventions 42
5.1.4 Local Bus Clocking 42
5.1.5 Local Bus Control Signals 42
5.1.6 Signals for Inbound Dataflow (PCIe-to-Local) 43
5.1.7 Signals for Outbound Dataflow (Local-to-PCIe) 45
5.2 Local Bus Protocol 46
5.2.1 Header Format 48
5.2.2 PCI Express Target Transactions 51
5.2.3 PCI Express Master Transactions 54
6. Interrupt Control Unit 57
6.1 Overview 57
6.2 Operation 58
7. Peripherals 59
7.1 2-Wire Interface 59
7.1.1 Master Mode 60
7.1.2 Slave Mode 65
7.1.3 2-Wire Interface Additional Information 67
7.2 General Purpose IO Block 67
8. FPGA Configuration Loader 70
8.1 Overview 70
8.2 Operation 70
8.3 Xilinx FPGA Configuration 71
8.4 Altera FPGA Configuration 74
8.5 FCL Programming 75
8.5.1 FCL Register Map 76
8.5.2 GPIO FPGA Configuration 77
8.5.3 FCL FPGA Configuration 77
8.5.4 FCL FSM FPGA Configuration 78
8.6 Additional information 79
9. Device Application 80
9.1 PCI Express Reference Clock Routing 80
9.2 PCI Express Power Supply Distribution 80

10. Internal Registers 82
10.1 Internal Register Access Modes 82
10.1.1 PCI Express Configuration Access 82
10.1.2 PCI Express BAR4 Access 82
10.1.3 2-Wire Access 82
10.2 Register Conventions 83
10.2.1 Register Bit Types 83
10.2.2 Numbering Conventions 84
10.2.3 Reserved Register Bits 84
10.3 Register Map 84
10.4 Register Descriptions 88
10.4.1 PCI Type 0 Configuration Header Registers and Related Control Registers 89
10.4.2 Power Management Capability Registers 102
10.4.3 Message Signalled Interrupt Registers 106
10.4.4 PCI Express Capability Registers 109
10.4.5 Device Serial Number Registers 116
10.4.6 Virtual Channel Capability Registers 118
10.4.7 GN412x System Registers 125
10.4.8 2-Wire Interface Registers 139
10.4.9 GPIO Registers 146
10.4.10 FPGA Configuration Loader Registers 158

List of Figures

Figure 1-1: GN412x with FPGA Simplified Block Diagram 12
Figure 2-1: GN412x Block Diagram 18
Figure 2-2: GN412x Signal Groups Diagram 19
Figure 3-1: Reset Input and Output Operation 20
Figure 3-2: Serial EEPROM Initialization Schematics 25
Figure 3-3: Connection for Initialization Using the Local Processor 26
Figure 3-4: GN412x Device Clocking 27
Figure 3-5: GN412x Local Bus Clock Generation PLL 28
Figure 4-1: PCI Express Interface Signals 31
Figure 4-2: Internal Register Space 34
Figure 5-1: Local Bus Interface Signals 40
Figure 5-2: Local Bus Timing 41
Figure 5-3: PCIe to Local Bus Target Write Transaction 52
Figure 5-4: PCIe to Local Bus Target Read Transaction 53
Figure 5-5: Local-to-PCIe Master Write Transaction 54
Figure 5-6: Operation of the L2P_RDY Signal During an L2P Transaction 55
Figure 6-1: Interrupt Controller Block Diagram 57
Figure 7-1: APB and Peripheral Devices Block Diagram. 59
Figure 7-2: 2-Wire Interface in Master Mode 61
Figure 7-3: Master Read Transfer (Normal Address Mode) 61
Figure 7-4: Master Read Transfer (Extended Address Mode) 61
Figure 7-5: Master Byte Read 63
Figure 7-6: Master Page Read 63
Figure 7-7: Master Write Transfer (Normal Address Mode) 63
Figure 7-8: Master Write Transfer (Extended Address Mode) 63
Figure 7-9: Master Byte Write 64
Figure 7-10: Master Page Write 64
Figure 7-11: 2-Wire Interface in Slave Mode 65
Figure 7-12: Slave Double Word Read 66
Figure 7-13: Slave Double Word Page Read 66
Figure 7-14: Slave Double Word Write 67
Figure 7-15: Slave Double Word Page Write 67
Figure 7-16: GPIO Block Diagram 68
Figure 8-1: FPGA Configuration Loader Block Diagram. 71
Figure 8-2: Serial Programming Interface: Xilinx Spartan-3(A) and Platform Flash example. 73
Figure 8-3: Serial Programming Interface: Altera Cyclone example. 75
Figure 9-1: PCI Express Related Power Supply Design 81

List of Tables

Table 3-1: Initialization Options 21
Table 3-2: Serial EEPROMs Compatible with the GN412x 23
Table 3-3: EEPROM Interpretation by the GN412x 24
Table 3-4: Address Format for EEPROM Interpretation 24
Table 3-5: GN412x Clock Selection and Control 28
Table 4-1: Flow Control Credits Advertised by the GN412x 39
Table 5-1: Signals for Inbound PCIe to Local Packet Flow 44
Table 5-2: Signals for Outbound Local to PCIe Packet Flow 45
Table 5-3: Summary of Local Bus Transaction Types 47
Table 5-4: Header Format for P2L Writes (Target Writes) 48
Table 5-5: Header Format for P2L Read Requests (Target Read Request) 48
Table 5-6: Header Format for L2P Read Completions (Master Read Completion Without Data) 48
Table 5-7: Header Format for L2P Read Completions (Master Read Completion With Data) 48
Table 5-8: Header Format for L2P Writes (32 bit Address Specified) 48
Table 5-9: Header Format for L2P Writes (64 bit Address Specified) 49
Table 5-10: Header Format for L2P Read Requests (32 bit Address Specified) 49
Table 5-11: Header Format for L2P Read Requests (64 bit Address Specified) 49
Table 5-12: Header Format for P2L Read Completions (Target Read Completion Without Data) 49
Table 5-13: Header Format for P2L Read Completions (Target Read Completion With Data) 49
Table 5-14: Local Bus Header Fields 50
Table 7-1: 2-Wire Interface Register Map 60
Table 7-2: GPIO Definition (when bypass mode is enabled via GPIO_BYPASS_MODE register) 68
Table 8-1: FCL Interface 71
Table 8-2: FCL Register Map 76
Table 10-1: Description of Register Bit Types 84
Table 10-2: Register Map 84
Table 10-3: PCI_VENDOR 89
Table 10-4: PCI_DEVICE 89
Table 10-5: PCI_CMD 89
Table 10-6: PCI_STAT 91
Table 10-7: PCI_REVISION 92
Table 10-8: PCI_CLASS_CODE 92
Table 10-9: PCI_CACHE 92
Table 10-10: PCI_LATENCY 93
Table 10-11: PCI_HEADER 93
Table 10-12: PCI_BIST 93
Table 10-13: PCI_BAR0_LOW 94
Table 10-14: PCI_BAR0_HIGH 95
Table 10-15: PCI_BAR2_LOW 96
Table 10-16: PCI_BAR2_HIGH 97
Table 10-17: PCI_BAR4_LOW 97
Table 10-18: PCI_BAR4_HIGH 98
Table 10-19: PCI_CIS 98
Table 10-20: PCI_SUB_VENDOR 98
Table 10-21: PCI_SUB_ID 99

Table 10-22: PCI_ROM_BASE 99
Table 10-23: PM_CAP_POINTER 100
Table 10-24: PCI_INT_LINE 100
Table 10-25: PCI_INT_PIN 100
Table 10-26: PCI_MIN_GNT 101
Table 10-27: PCI_MAX_LAT 101
Table 10-28: PM_CAP_ID 102
Table 10-29: PM_NEXT_ID 102
Table 10-30: PM_CAP 103
Table 10-31: PM_CSR 104
Table 10-32: PM_CSR_BSE 105
Table 10-33: PM_DATA 105
Table 10-34: MSI_CAP_ID 106
Table 10-35: MSI_NEXT_ID 106
Table 10-36: MSI_CONTROL 106
Table 10-37: MSI_ADDRESS_LOW 107
Table 10-38: MSI_ADDRESS_HIGH 107
Table 10-39: MSI_DATA 108
Table 10-40: PCIE_CAP_ID 109
Table 10-41: PCIE_NEXT_ID 109
Table 10-42: PCIE_CAPABILITY 109
Table 10-43: PCIE_DEVICE_CAP 110
Table 10-44: PCIE_DCR 111
Table 10-45: PCIE_DSR 112
Table 10-46: PCIE_LINK_CAP 113
Table 10-47: PCIE_LCR 114
Table 10-48: PCIE_LSR 115
Table 10-49: DSN_CAP 116
Table 10-50: DSN_LOW 116
Table 10-51: DSN_HIGH 117
Table 10-52: VC_CAP 118
Table 10-53: VC_PORT_CAP_1 118
Table 10-54: VC_PORT_CAP_2 119
Table 10-55: VC_PCR 119
Table 10-56: VC_PSR 120
Table 10-57: VC_RESOURCE_CAP0 120
Table 10-58: VC_RESOURCE_CR0 121
Table 10-59: VC_RESOURCE_SR0 122
Table 10-60: VC_RESOURCE_CAP1 122
Table 10-61: VC_RESOURCE_CR1 123
Table 10-62: VC_RESOURCE_SR1 124
Table 10-63: PCI_SYS_CFG_SYSTEM 125
Table 10-64: LB_CTL 127
Table 10-65: CLK_CSR 128
Table 10-66: PCI_BAR_CONFIG 129
Table 10-67: INT_CTRL 130
Table 10-68: INT_STAT 131
Table 10-69: PEX_ERROR_STAT 132
Table 10-70: INT_CFG0-7 133
Table 10-71: PCI_TO_ACK_TIME 134
Table 10-72: PEX_CDN_CFG1 134
Table 10-73: PEX_CDN_CFG2 135

Table 10-74: PHY_TEST_CONTROL 135
Table 10-75: PHY_CONTROL 136
Table 10-76: CDN_LOCK 138
Table 10-77: TWI_CTRL 139
Table 10-78: TWI_STATUS 140
Table 10-79: TWI_ADDRESS 140
Table 10-80: TWI_DATA 141
Table 10-81: TWI_IRT_STATUS 141
Table 10-82: TWI_TR_SIZE 143
Table 10-83: TWI_SLV_MON 143
Table 10-84: TWI_TO 144
Table 10-85: TWI_IR_MASK 144
Table 10-86: TWI_IR_EN 144
Table 10-87: TWI_IR_DIS 145
Table 10-88: GPIO_BYPASS_MODE 146
Table 10-89: GPIO_DIRECTION_MODE 147
Table 10-90: GPIO_OUTPUT_ENABLE 148
Table 10-91: GPIO_OUTPUT_VALUE 149
Table 10-92: GPIO_INPUT_VALUE 150
Table 10-93: GPIO_INT_MASK 151
Table 10-94: GPIO_INT_MASK_CLR 152
Table 10-95: GPIO_INT_MASK_SET 153
Table 10-96: GPIO_INT_STATUS 154
Table 10-97: GPIO_INT_TYPE 155
Table 10-98: GPIO_INT_VALUE 156
Table 10-99: GPIO_INT_ON_ANY 157
Table 10-100: FCL_CTRL 158
Table 10-101: FCL_STATUS 159
Table 10-102: FCL_IODATA_IN 159
Table 10-103: FCL_IODATA_OUT 160
Table 10-104: FCL_EN 160
Table 10-105: FCL_TIMER_0 161
Table 10-106: FCL_TIMER_1 161
Table 10-107: FCL_CLK_DIV 162
Table 10-108: FCL_IRQ 162
Table 10-109: FCL_TIMER_CTRL 163
Table 10-110: FCL_IM 163
Table 10-111: FCL_TIMER2_0 164
Table 10-112: FCL_TIMER2_1 164
Table 10-113: FCL_FIFO_DATA 164

List of Acronyms

Acronym	Description
APB	Advanced Peripheral Bus
ASPM	Active State Power Management
BAR4	Base Address Registers
DSNC (DSN_)	Device Serial Number Capability
FCL (FCL_)	FGPA Configuration Loader
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GPIO	General Purpose Input Output
LPVC	Low-Priority Virtual Channel
MSI (MSI_)	Message Signalled Interrupt
PCIEC (PCIEC_)	PCI Express Capability
PEX	PCI Express Digital Core with PIPE and Generic Transaction Layer
PM (PM_)	Power Management
RCB	Read Completion Boundary
SPRI (SPRI_)	Serial Programming Interface
TBD	To Be Defined
TWI (TWI_)	2-wire interface
UR	Unsupported Request
VC (VC_)	PCI Express Virtual Channel Capability

List of References

Reference Number	Reference Title
[1]	PCI Express™ Base Specification Revision 1.1, PCI-SIG, March 28, 2005
[2]	Inter Integrated Circuit (I^2C) Technical Data Sheet, Cadence, I-IPA01-0045-TDS Rev 11, February 2007
[3]	The I^2C-bus Specification, Philips Semiconductor, Version 2.1, January 2000
[4]	Configuration Handbook, Altera, 1.4, October 2007
[5]	Spartan-3 Generation Configuration User Guide, Xilinx, UG332 (v1.2) May 23, 2007
[6]	The 3.3V Configuration of Spartan-3 FPGAs, XAPP453 (v1.1) April 3, 2006

1. Introduction

For the past decade, PCI has been a dominant interconnect for both PC and embedded systems. With the shift to high-speed serial interfaces, PCI Express® is quickly replacing parallel PCI. As a leader in providing solutions for high-speed serial communications, Gennum has developed the GN412x family of PCI bridge controller components to complement FPGA devices. The GN412x family, comprised of the GN4124 and GN4121, is specifically designed to take advantage of the architectural features of low-cost FPGA devices that do not have PCI Express capable SerDes on-chip. The result is a low-cost bridging solution for high-performance native PCI Express bridging.

The GN412x is also desirable companion to PCIe capable FPGA devices, where the requirement for firmware upgrading and on-the-fly reconfiguration are required.

The GN4124 is a x4-lane PCI Express to local bus bridge and the GN4121 is a single lane variant. Both devices in the family are designed to work as a companions for FPGA devices to provide a complete bridging solution for general applications. In addition to a PCI Express compliant PHY interface (x4-lane in the case of the GN4124 and x1-lane for the GN4121), the GN412x devices contain the link and transaction layers, and an applications interface that is ideally suited to FPGA interfacing using a small number of pins.

Since the PCI Express transaction/link IP is hard-wired into the GN412x, there is no need to license PCIe IP. The level of integration and very low power operation of the GN412x make it an ideal alternative to using a PIPE PHY, where IP licensing and the cost of FPGA resources and power consumption is unattractive by comparison. Using the GN412x, allows FPGA resources to be spent on what differentiates the product, rather than on implementing the PCI Express protocol.

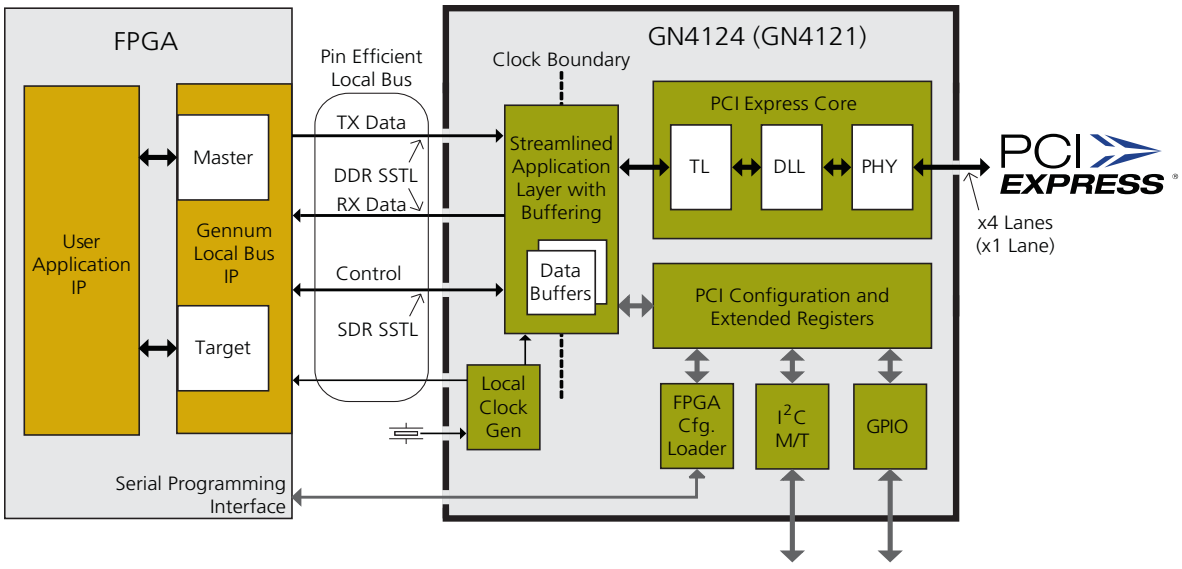
1.1 Features

- 4 Lane PCI Express interface for GN4124 OR 1 Lane PCI Express interface for GN4121
  - Complies with PCI Express Base Specification 1.1
  - On-chip PHY, transaction, and link layer eliminates the cost of IP licensing
  - Two hardware virtual channels supported by the GN4124 (the GN4121 provides a single hardware virtual channel)
  - Payload size of up to 512 bytes with up to three outstanding transactions in each direction per virtual channel
  - Supports 3x64-bit base address registers
  - Provides flexible power management capability

- Provides pin efficient local bus interface for easy attachment to popular low-cost FPGA devices
  - Uses DDR SSTL I/O for high-speed local bus data transfer (up to 800MB/s for the GN4124 and 500MB/s for the GN4121)
  - FPGA source code provided for 64-bit master/target read/write buses for easy user logic attachment
  - Local bus may be operated asynchronously to the PCIe clock rate for power optimization
- "Live" on power up
  - On-chip type 0 PCI configuration space enables auto detection without FPGA activity
  - On-chip extended configuration space supports power management, serial number, MSI, and PCIe capability registers
- FPGA bitstream loader
  - Allows easy configuration of the attached FPGA through PCIe
  - Provides on-the-fly FPGA reconfiguration capability
- 2-wire master/target
  - Boot master mode allows PCI configuration space defaults to be loaded from a small EEPROM upon system reset
  - General master mode allows attached 2-wire devices to be read/written
  - Target mode allows internal registers to be accessed from an external circuit or processor

A simplified block diagram of the GN412x chip is shown in Figure 1-1.

Figure 1-1: GN412x with FPGA Simplified Block Diagram



1.2 Live on Power-up

Since the GN412x contains a complete type 0 PCI configuration space, it is live on power-up so that a plug-and-play BIOS can auto-detect it and enumerate it without an attached FPGA having to be configured.

1.3 FPGA On-the-Fly Configuration Loader

An FPGA bitstream may be downloaded from the host system over PCIe to the attached FPGA using the on-chip FPGA configuration loader. This eliminates the expense of a dedicated FPGA ROM and makes on-the-fly reconfiguration and firmware upgrades simple. The ability to dynamically configure an attached FPGA over PCIe makes the GN412x an ideal companion to all ranges of FPGA devices, including large SerDes capable devices, that require reconfiguration or firmware upgrades over PCIe.

1.4 Local Bus Interface

The local bus interface uses a combination of single and dual data rate SSTL I/O to accomplish very high data rates in the fewest possible pins. A singe data rate clock is used for SSTL control signals and separate dual data rate source synchronous clocking is used for the DDR SSTL data. The SDR control signals operate at up to 200MHz (125MHz for the GN4121) and the DDR I/O operate at up to 400MT/s across (250MT/s for the GN4121) 16 bits using a 200MHz (125MHz) DDR clock. This provides 800MB/s in each direction for the GN4124 and 500MB/s in each direction for the GN4124.

The local bus may operate asynchronously from the PCI Express rate. In order to save power, the local bus clock can operate at the lowest possible rate required by an application.

The local bus protocol facilitates four types of transactions:

- PCIe-to-Local Target Writes: A PCIe agent (such as the host processor/root complex) writes data to the local bus.
- PCIe-to-Local Target Reads: A PCIe agent reads data from the local bus. Reads are split into a request phase (address phase) and a completion phase (data phase).
- Local-to-PCIe Master Writes: The attached FPGA writes data to a PCIe device (such as host memory via a root complex).
- Local-to-PCIe Master Reads: The attached FPGA reads data from a PCIe device.

The PCIe-to-Local transactions would typical involve a target controller implemented in the FPGA. Local-to-PCIe Master transactions allow a DMA controller in the FPGA to access PCI Express devices.

1.5 Virtual Channel Support

The GN4124 has two independent virtual channels that support the eight PCIe defined traffic classes. This enables high local bus utilization by supporting non-blocking traffic between virtual channels. This is accomplished with separate on-chip buffering resources for each of the two virtual channels. For example, when write buffering is full for VC0 and VC1 has room, then VC1 traffic may proceed without reference to the state of VC0. The GN4121 can only support VCO in hardware, even though VC1 related registers are available for the sake of software compatibility.

Note: The GN4121 device supports one virtual channel.

Virtual channels may be used to separate different types of application traffic. For example, a DMA engine in the FPGA may be aggressively reading and writing host memory to stream video data. At the same time another agent in the FPGA may need to communicate low bandwidth, latency sensitive synchronization information. If the two types of traffic are segregated in terms of virtual channels and traffic classes, then the low latency traffic can be allowed to pass the high bandwidth traffic.

1.6 PCI Express Application Layer

The on-chip applications layer transfers data between the PCI Express port and an attached FPGA using the local bus interface. It provides a mechanism to access internal registers through configuration space access and through one of the Base Address Registers (BAR4). The applications layer supports the transmission of message signalled interrupts.

1.7 Interrupt Controller

A flexible interrupt controller automatically generates PCIe message signalled interrupts from either external pins (GPIO pins) or internally generated interrupt sources. The interrupt controller can route any interrupt source to up to four GPIO pins.

1.8 2-Wire Serial Controller

An on-chip I^2C compatible controller provides both a master and target mode. After device reset, default configuration register values, such as Subsystem Vendor ID and BAR sizes, can be automatically loaded from a small serial EEPROM. After initialization, an external 2-wire master can access on-chip registers to read/write them.

1.9 How to Use this Family Reference Manual

The GN412x Family Reference Manual includes detailed information on GN412x-based systems programming and design. However, there are other complementary documents to assist designers available on the Gennum Web site: www.gennum.com/mygennum. A complete set of documentation includes the following:

- GN4124 Data Sheet (Document ID: 48407) OR GN4121 Data Sheet (Document ID: 51239)
- GN412x PCI Express Family Reference Manual (this document)
- GN4124 Master List of Documents & Electronic Files (Document ID: 52423) OR GN4121 Master List of Documents & Electronic Files (Document ID: 52571), which provides a summary of the content of the documentation, to help navigate the content
- Reference Design Kit (RDK) board and the asssociated documentation

Following chapters detail the operation of the major functional blocks inside the GN412x:

- 3. Initialization
- 4. PCI Express Link
- 5. Local Bus Interface
- 6. Interrupt Control Unit
- 7. Peripherals
- 8. FPGA Configuration Loader
- 9. Device Application
- 10. Internal Registers

Before finalizing a system design based on the GN412x, please contact Gennum to verify that you have the most recent specifications.

Gennum is constantly trying to improve the quality of its product documentation. If you have any questions or comments, please contact Gennum Technical Support.

1.9.1 Family Reference Manual Conventions

Throughout this document, references to the GN412x may be applied to either of the devices and indicate that they operate the same. Where there is information specific to one or the other of the devices, the specific device (GN4124 or GN4121) will be named.

The GN412x can act as either a "requester" or "completer". A requester is a device that first introduces a transaction sequence into the PCI Express domain. In legacy PCI, this would equate to a "bus master" and a completer would be the equivalent of a "bus target".

In this document, PCIe master and PCIe requester may be used interchangeably. Similarly, completer and target may also be used interchangeably.

In documenting and labelling cycles and transactions, the following conventions are used:

- In general, transactions are specified as requester followed by completer. Therefore, a PCIe-to-local write is a transaction where a requester device, attached to the GN412x, transmits a write request to the GN412x that results in a write cycle on the local bus. Since all local bus writes are posted, there is no response at the local bus level[1].
- A read transaction happens in two steps: the read request initiated by the requester, followed by the read completion initiated by the completer. This is also referred to as a split transaction[2].

1.9.1.1 Numbering Conventions

Throughout this data sheet, hexadecimal numbers will use a 0x prefix. Multiple bits represented as binary will be shown in double quotes as in "1010" while single quotes will be used to denote single binary bits (either '1', or '0').

Unless otherwise noted, little endian conventions will be used. Consequently, the binary value "1000" represents bit 3 ='1' and bits 2 down-to 0 = '0'.

The following numbers are the same value:

- Example: Decimal 165 = 0xA5 = "10100101"

Binary numbers may use "X" to represent "don't care" bits:

- Example: "101X" represents both "1010" and "1011"

1.9.1.2 Data Units

In keeping with PCI tradition, a 32-bit quantity will be referred to as a DW (double word) or alternately as a 32-bit word. Similarly, a 64-bit quantity may be referred to as a QW (quad word) or a 64-bit word.

1. PCIe write requests are acknowledged below the application layer. Consequently, there is no need to generate any response to write cycles at the local bus level.
2. A split transaction is a single logical transfer containing an initial transaction (the Request) terminated at the target (the Completer), followed by one or more transactions initiated by the Completer in response to the Request.

1.9.1.3 Register Naming Conventions

Each register in the GN412x is given a mnemonic label. In each register, each bit and bit field are also given a mnemonic. A register bit or bit field is fully identified as: REGISTER.BITFIELD where REGISTER refers to the mnemonic of the register and BITFIELD refers to a bit or bit field inside the register.

1.10 Getting Help from Gennum

For technical support, contact Gennum by telephone or e-mail. E-mail ensures the quickest response. The most up-to-date technical support information is also posted on the Gennum website. E-mail: vbapps@gennum.com.

1.11 Getting Answers to PCI Express Related Questions

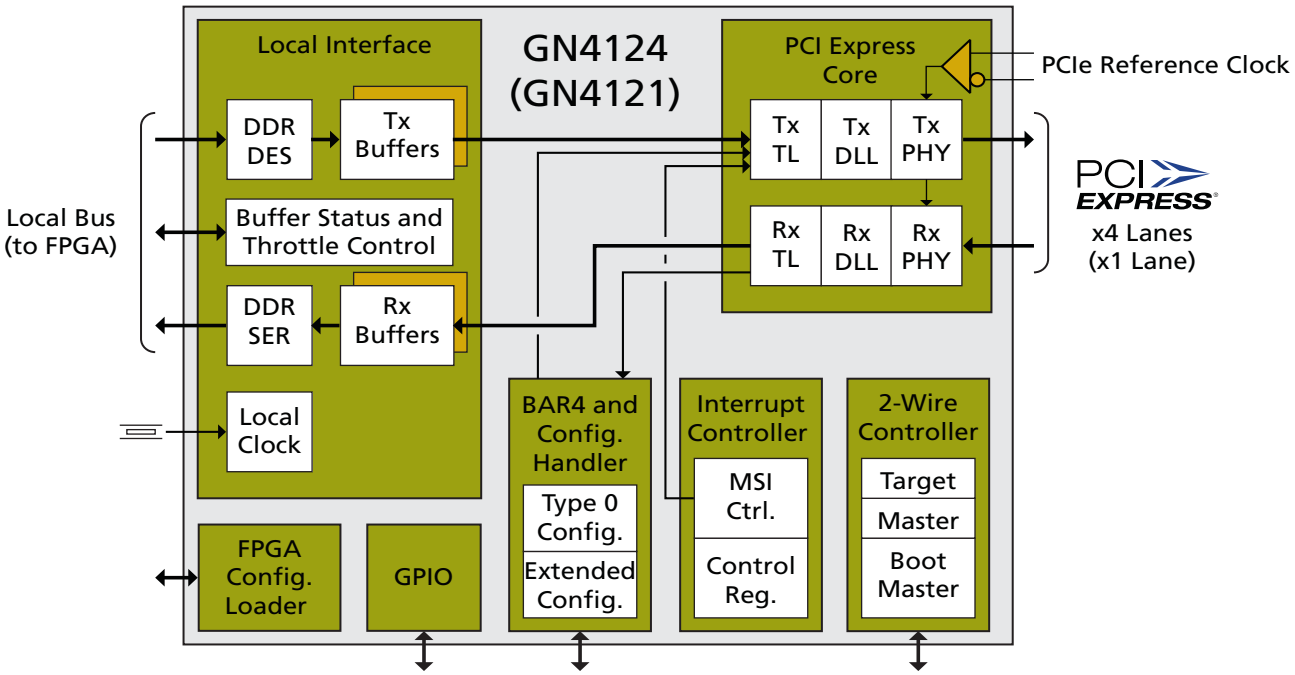
This data sheet assumes a basic understanding of the PCI Express Specification. If you are looking for a copy of the specification please contact the PCI Special Interest group at 503-619-0569 or visit their Web site at: http://www.pcisig.com.

If you are not familiar with the PCI Express specification, a good place to start is by reading one of several books on the subject. One of the most popular is PCI Express System Architecture written by Tom Shanley, Don Anderson, and Ravi Budruk (published by MindShare Inc.).

2. Overview

A block diagram of the GN412x is depicted in Figure 2-1.

Figure 2-1: GN412x Block Diagram



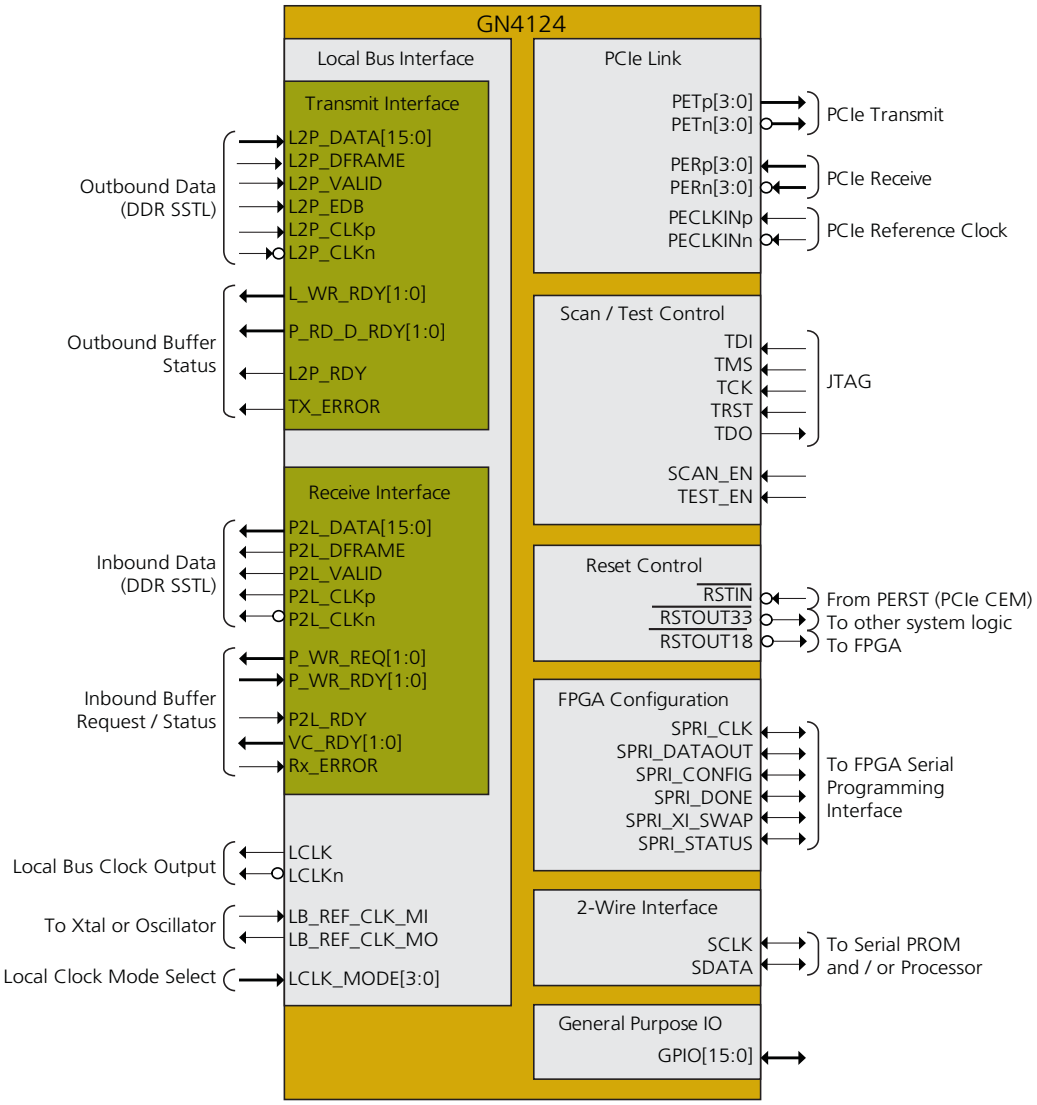
Each of the internal blocks is described in detail in the following chapters. They are:

- The PCI Express link is described in 3. PCI Express Link. This includes a description of the PCI Express related configuration registers.
- The Local bus is described in 4. Local Bus Interface.
- The interrupt controller is described in 5. Interrupt Control Unit.
- The boot master mode of the 2-wire controller is described in 3.1.3 Initialization from a 2-Wire EEPROM.
- General purpose master/target mode of the 2-wire controller is described in 7.1 2-Wire Interface.
- General purpose IO are described in 7.2 General Purpose IO Block.
- Details of all the internal registers and their respective register bit fields are described in 7. Internal Registers.

2.1 GN412x Signals

Figure 2-2 depicts the signals of the GN412x laid out in their logical groupings.

Figure 2-2: GN412x Signal Groups Diagram



Note for GN4121 device, there is a reduction in pins, as a result
L_WR_RDY[1:0] becomes L_WR_RDY,
P_RD_D_RDY[1:0] becomes P_RD_D_RDY,
P_WR_REQ[1:0] becomes P_WR_REQ,
P_WR_RDY[1:0] becomes P_WR_RDY,
VC_RDY[1:0] becomes VC_RDY,
PETp[3:0] becomes PETp,
PETn[3:0] becomes PETn,
PERp[3:0] becomes PERp, and
PERn[3:0] becomes PERn.

3. Initialization

Initialization of the GN412x usually occurs when the system is reset. The GN412x is reset when the $\overline{\text{RSTIN}}$ signal is driven active LOW. When $\overline{\text{RSTIN}}$ is later de-asserted, the GN412x's internal registers can be initialized via the PCI Express bus, an attached 2-wire[1] EEPROM, or from a processor through the 2-wire interface. This chapter describes reset options and register initialization procedures.

There are two types of initialization to be discussed:

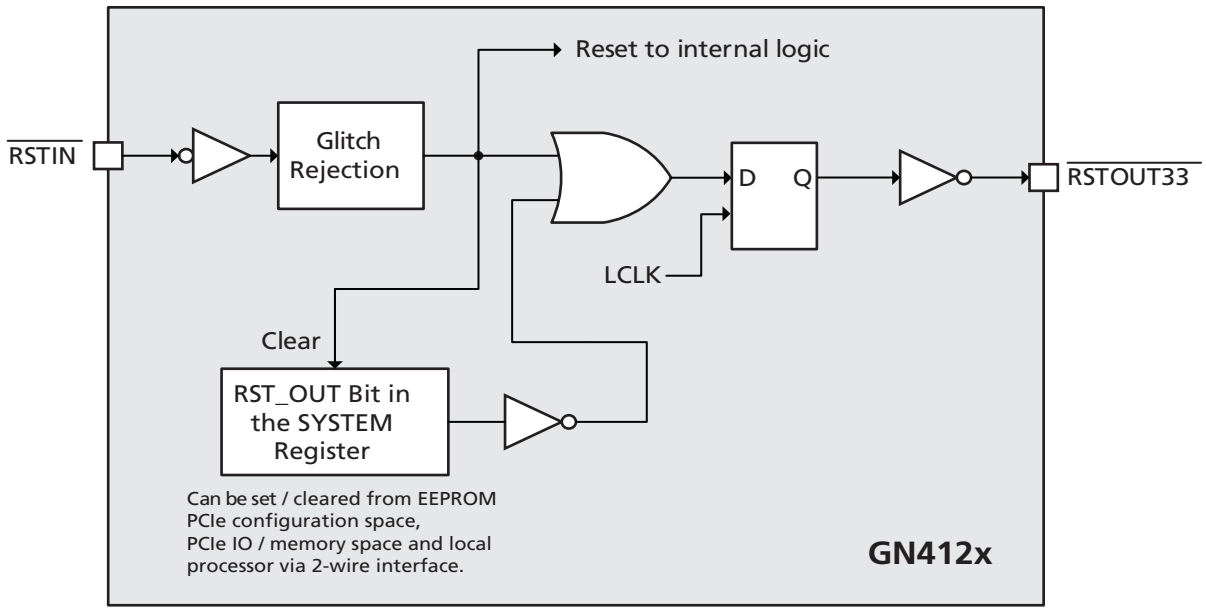
- Reset initialization: Initialization that happens after reset is released and before the PCIe BIOS or host operating system interacts with the GN412x.
- OS initialization: Initialization that happens after reset initialization when the PCIe BIOS or host operating system interacts with the GN412x.

3.1 Reset Initialization

The GN412x chip is reset by driving active (LOW) the $\overline{\text{RSTIN}}$ pin. For a PCIe Endpoint application, $\overline{\text{RSTIN}}$ is driven from the $\overline{\text{PERST}}$ (PCI Express Reset bar) signal of the PCIe connector. RSTIN doesn't have to be synchronous to any particular clock and is relatively immune to noise spikes due to its high level of input hysteresis.

Reset input and output operation is depicted in Figure 3-1.

Figure 3-1: Reset Input and Output Operation



1. The 2-wire interface is compatible with the I^2C specification. I^2C is a trademark of Koninklijke Philips Electronics NV.

3.1.1 Initializing the Internal Registers

There are three options for initializing the GN412x internal registers set after reset:

- Initialization from the 2-wire serial EEPROM interface.
- Initialization from the PCIe bus side via configuration space.
- Initialization from the 2-wire interface by an external 2-wire master.

3.1.2 Selecting an Initialization Mode

In order to support Plug-and-Play operation, an endpoint device must provide configuration space information back to the host operating system so that it knows which drivers to load. Since the GN412x may be used in multiple applications from multiple vendors, the default configuration space settings would not allow each application variant to be uniquely identified. In order to provide unique application dependant configuration space settings, some of the defaults must be overwritten. This will include at least the Subsystem Vendor ID and Subsystem ID. These registers will allow an operating system to determine the driver to load during the "OS initialization" phase.

There are three reset initialization modes that can be selected. These are shown in Table 3-1.

Table 3-1: Initialization Options

EEPROM Port Connection	EEPROM_EN State	Bits in PCI_SYS_CFG_SYSTEM Register		Comment
		CFG_RETRY	RST_OUT	
SDATA and SCLK connected to valid EEPROM device[1]	'1'	From EEPROM	From EEPROM	Initialization from EEPROM Mode: This option can be used in Plug-and-Play applications.
SDATA pulled HIGH during reset	'0'	1	1	Local Processor Mode: Typically used for initialization via local processor (or other 2-wire capable agent) using the 2-wire port. This option can be used in Plug-and-Play applications.
SDATA pulled LOW during reset	'0'	0	0	Embedded Mode: Typically used for initialization via PCIe in an embedded application where the components in the system are controlled.

1. A valid EEPROM connection requires SDATA and SCLK to be connected to an EEPROM configured to respond at device address 6.

For general purpose endpoint applications, the “Initialization from EEPROM Mode” is recommended.

Note: After the input reset has been released, 10 SCLK clocks are performed followed by a STOP cycle to ensure that a reset didn't falsely see the SDATA signal LOW. This situation can occur if the GN412x is reset in the middle of an EEPROM download caused by a prior reset release.

3.1.3 Initialization from a 2-Wire EEPROM

The GN412x registers can be initialized, prior to interrogation by the system BIOS or operating system, from a 2-wire serial EEPROM. This mode is referred to as “Boot Master” mode. Boot master mode is different from the general purpose master mode provided by the 2-wire controller registers described in 2-Wire Interface Registers (on page 139). Boot master mode is only initiated immediately following de-assertion of the $\overline{\text{RSTIN}}$ pin and before an upstream PCIe agent accesses the GN412x. By contrast, the general master mode is accessed by an agent, typically the host processor, through the PCI Express interface using BAR4 access.

Boot master capability is enabled when the EEPROM_EN signal is tied HIGH. In this mode, the GN412x acts as a 2-wire master and downloads values for part of the internal register block from a 2-wire EEPROM connected to the SCLK and SDATA pins. Any subset of writable register bits may be written from EEPROM. This includes the RST_OUT and CFG_RETRY bits in the PCI_SYS_CFG_SYSTEM (on page 125) register.

For normal operation, RST_OUT bit should be set to ‘1’ and CFG_RETRY bit set to ‘0’ by the EEPROM. Setting RST_OUT bit to ‘1’ will cause the $\overline{\text{RSTOUT33}}$ pin to be de-asserted, so that the local subsystem can be released from reset upon completion of the EEPROM download. Clearing the CFG_RETRY bit will cause the GN412x to complete configuration cycles issued by the root complex.

The other registers that will typically require initialization prior to interrogation by the system BIOS or operating system are:

- PCI_CLASS_CODE (on page 92)
- PCI_BAR0_LOW (on page 94), PCI_BAR0_HIGH (on page 95), PCI_BAR2_LOW (on page 96), PCI_BAR2_HIGH (on page 97): To determine the address range and BAR type.
- PCI_BAR4_LOW (on page 97), PCI_BAR4_HIGH (on page 98), PCI_BAR_CONFIG (on page 128): To determine the size of the base address registers BAR0, BAR2, and the ROM base address register.
- PCI_SUB_VENDOR (on page 98): Should be programmed with the Vendor ID, which is provided by the PCI SIG to the manufacturer of the card using the GN412x. It should NOT be programmed to the Gennum PCI ID!
- PCI_SUB_SYS (on page 99): The subsystem vendors subsystem ID.
- DSN_LOW (on page 116), DSN_HIGH (on page 117): If a device serial number is to be used.

- PM_CSR (on page 104), PM_DATA (on page 105): To advertise the correct power consumed and dissipated for various power states.

The boot master mode interface is designed to work with 24C02 style serial EEPROMs. Table 3-3 shows serial EEPROMs compatible with this interface. EEPROMs larger than 256 bytes may be used and locations above 0xFF in the EEPROM will be ignored during initialization.

When the reset input is de-asserted, the SCLK pin pulses ten times ending with a STOP condition. A sequential read from the EEPROM is performed starting from 0x00, and the data is interpreted as address/data combinations described later.

The address of the serial EEPROM pins must be strapped to six (A[2:0] = “110”).

Other EEPROM addresses are available for other purposes such as reading the configuration information from SDRAM modules, which usually have an EEPROM using the same 2-wire protocol. In this case, boot software can read the SDRAM parameters from the SDRAM DIMM through software control and, then program an SDRAM controller appropriately.

Serial EEPROMs known to be compatible with the GN412x are shown in Table 3-2.

Note: It is highly recommended that all initial designs use an EEPROM for configuration, even if you do not intend to use one in production. This can save a lot of debug time in the early stages of your design. In production, the EEPROM can be de-populated.

Table 3-2: Serial EEPROMs Compatible with the GN412x

Manufacturer	Device[1]
Atmel	AT24C02A
Microchip	24LC024
ISSI	IS24C02A

1. The GN412x requires 3.3V EEPROM operation.

3.1.3.1 Programming the Serial EEPROM

The EEPROM contents are interpreted as sets of 6 bytes: a 2-byte address and a 4-byte data word. Each address / data set can be used to configure one 32-bit register or several registers that may be packed into a 32-bit naturally aligned word. The EEPROM contents are packed as shown in Table 3-3.

Table 3-3: EEPROM Interpretation by the GN412x

Byte Address	EEPROM Contents
0x00	ADDRESS 0 low bits [7:0]
0x01	ADDRESS 0 high bits [15:8]
0x02	DATA 0 bits [7:0]
0x03	DATA 0 bits [15:8]
0x04	DATA 0 bits [23:16]
0x05	DATA 0 bits [31:24]
0x06	ADDRESS 1 low bits [7:0]
0x07	ADDRESS 1 high bits [15:8]
0x08	DATA 1 bits [7:0]
0x09	DATA 1 bits [15:8]
0x0A	DATA 1 bits [23:16]
0x0B	DATA 1 bits [31:24]
. . .	. . .
N*6	ADDRESS N low bits [7:0]
N*6+1	ADDRESS N high bits [15:8]
N*6+2	DATA N bits [7:0]
N*6+3	DATA N bits [15:8]
N*6+4	DATA N bits [23:16]
N*6+5	DATA N bits [31:24]
N*6+6	0xFF
N*6+7	0xFF

The address format is given below:

Table 3-4: Address Format for EEPROM Interpretation

ADDRESS Bits															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BE3	BE2	BE1	BE0	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	X	X

Each 32 bits of data is written into the internal registers using the address bits A[11:2]. The data is aligned to a 32-bit boundary, and thus A1 and A0 are not considered. The upper 4 bits (BE[3:0]) provide an enable-per-byte for each byte of the 32 bit word. (BE[0] corresponds to Byte 0 of 32-bit word; BE[1] corresponds to Byte 1; BE[2] corresponds to

Byte 2; and BE[3] corresponds to Byte 3.) For each register byte that is to be written by the EEPROM data, the corresponding BE needs to be set to '1'.

The download sequence ends immediately when PCI_SYS_CFG_SYSTEM.CFG_READY is set to 0, therefore the last entry of the EEPROM sets the PCI_SYS_CFG_SYSTEM.CFG_READY to 0.

Programming of the serial EEPROM is straightforward on most commercial EPROM-programmers. An assembler may be used to generate a HEX file for download to a programmer.

The GN412x may be used to program the EEPROM directly in the target application using the 2-wire controller (see Section 7.1.1). This capability is provided by the GenDiag utility available from Gennum.

3.1.3.2 Timing Considerations when Initializing via the Serial EEPROM

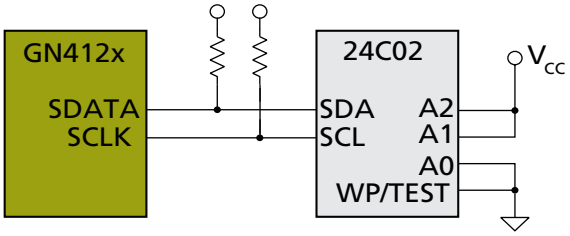
The clock for the serial EEPROM is derived from the PCIe clock and is nominally 100kHz. The EEPROM download time is calculated as:

10µs (SCLK Period) x 261 (number of bytes) x 9 (SCLK's per byte) = 23.49ms

The schematics for serial EEPROM initialization are given in Figure 3-2.

Note: Configuration access from PCIe will cause "Configuration Request Retry" completion status to be returned to the root complex until the download is complete.

Figure 3-2: Serial EEPROM Initialization Schematics



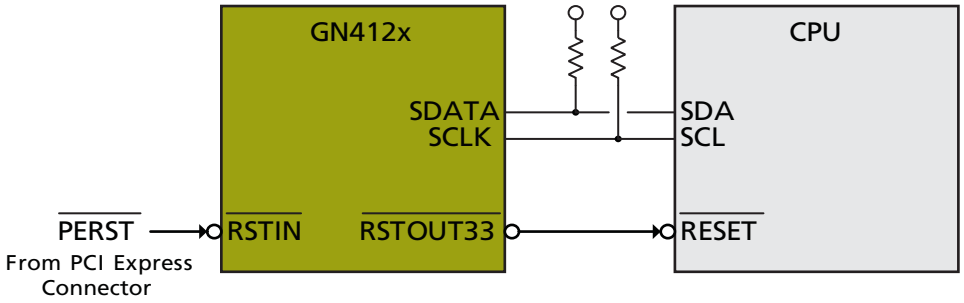
3.1.3.3 Determining that Initialization via EEPROM is Complete

When the bridge is being initialized from an EEPROM, the $\overline{\text{RSTOUT33}}$ signal can be used to inform the local subsystem that the download is complete. In the EEPROM, the RST_OUT bit for the PCI_SYS_CFG_SYSTEM (on page 125) register should be set. This causes the $\overline{\text{RSTOUT33}}$ signal to be driven HIGH (de-asserted) after the EEPROM has been downloaded.

3.1.4 Initialization Using a Local Processor

It may be desirable to use a local processor or 2-wire agent to initialize the contents of the internal registers of the GN412x and save the cost of the serial EEPROM device. This is best accomplished by pulling the SDATA pin HIGH on the GN412x and connecting the processor as shown in Figure 3-3.

Figure 3-3: Connection for Initialization Using the Local Processor



Note: Note: By default, the GN412x will respond to a 2-wire address of 5 ("101"). More information on the 2-wire interface controller is found in 7.1 2-Wire Interface.

When connected as shown in Figure 3-3, PCI_SYS_CFG_SYSTEM.CFG_RETRY is set when $\overline{\text{RSTIN}}$ is asserted and remains that way until the local CPU clears it. PCI_SYS_CFG_SYSTEM.CFG_RETRY is used to cause a PCI configuration access made to the GN412x to receive a Configuration Request Retry Status (CRS) in response until PCI_SYS_CFG_SYSTEM.CFG_RETRY is cleared from the local CPU. This ensures that the local CPU gets a chance to properly initialize before the PCIe BIOS tries to enumerate the GN412x. Once the local CPU has initialized the internal registers, PCI_SYS_CFG_SYSTEM.CFG_RETRY can be cleared to allow the host PCIe agent to enumerate the GN412x.

Note: All PCI Express functions, except configuration space access, are disabled following a hardware reset. For example, all PCIe accesses are ignored by the GN412x until enabled by software (with the exception of PCI configuration cycles as described to the left).

3.1.5 Initialization Using the PCIe Configuration Space

The GN412x can be initialized from a PCIe root complex using configuration space accesses. Access to the configuration space of the GN412x from PCIe is achieved by performing type 0 configuration transactions. PCIe transactions of this type in the address range 0x00-0xFF are forwarded to the internal configuration registers.

The GN412x accepts only Type 0 (standard configuration cycle) configuration accesses. Type 1 (PCI to PCI bridge configuration cycles) configuration accesses are ignored.

3.2 OS Initialization

Plug-and-Play initialization consists of root complex initiated type 0 configuration transactions to the GN412x. During this phase of initialization, the host operating system will read the configuration space registers and determine how to enumerate the bridge. During enumeration, the BIOS or operating system will typically write values into the base address registers (BAR) as it establishes the memory map of the system.

3.3 PCI Express Configuration Space Initialization

The PCI Express configuration space of the GN412x contains two types of registers: registers that are specified by the PCI Express standard; and registers that are specific to the GN412x. The following register groups are present in the GN412x:

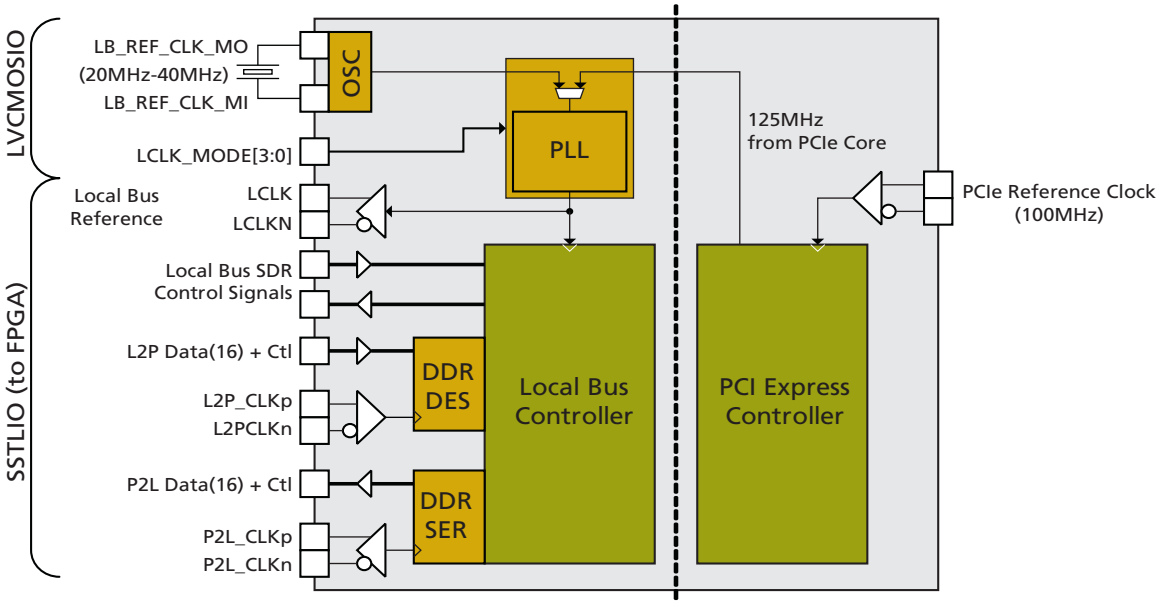
- Type 0 Configuration space
- PCI Express Capability (registers that start with PCIEC_)
- PCI Power Management Capability (registers that start with PM_)
- MSI Capability (registers that start with MSI_)

- Device Serial Number Capability (registers that start with DSN_)
- PCI Express Virtual Channel Capability (registers that start with VC_)

3.4 Clock Configuration

The GN412x has two main clock domains: the PCIe domain derived from the PCI Express reference clock; and the local clock. The clock scheme is depicted in Figure 3-4.

Figure 3-4: GN412x Device Clocking



3.4.1 PCI Express Clock

In an endpoint card application, the PCI Express reference clock on the GN412x is typically driven by the reference clock available on the card edge connector. Alternately, it may be driven by another 100MHz reference that meets the PCI Express reference clock specification.

3.4.2 Local Clock

There are 3 local clocks used by the GN412x and attached FPGA. They are:

- **LCLK/LCLKn**: The primary clock generated by the GN412x and driven to the FPGA in the form of a differential SSTL output.
- **P2L_CLKp/n**: The source synchronous clock used by the GN412x to communicate data to the FPGA. It is derived from the same source as LCLK/LCLKn.
- **L2P_CLKp/n**: The source synchronous clock generated by the attached FPFA to communicate data to the GN412x. It is derived by the FPGA from LCLK/LCLKn.

The local clock LCLK/LCLKn may be derived from either the PCI Express clock or a low frequency crystal oscillator. The options are described in Table 3-5.

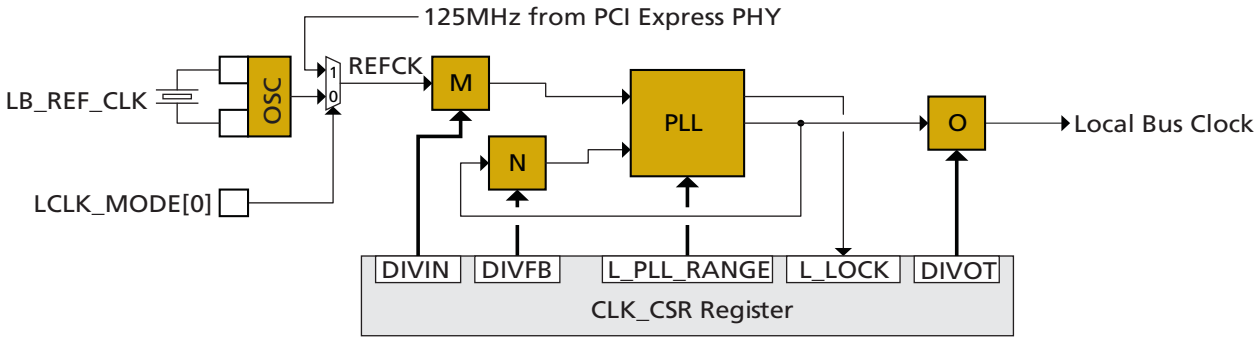
Table 3-5: GN412x Clock Selection and Control

Signal	Description
LCLK_MODE[3]	Control IDDQ of the pads. Set to low for normal operation.
LCLK_MODE[2]	ControlsPLL Bypass. '0' = The reference for the LCLK PLL is taken from an external xtal or oscillator. This is recommended for low and predictable LCLK clock jitter. '1' = The reference for the LCLK PLL is taken from the 125MHz clock generated from the PCI Express link.[1]
LCLK_MODE[1]	Resets the PLL test clock divider. '0' = Resets the PLL test clock divider, so that PLL_TEST_OUT = '0'. '1' = PLL_TEST_OUT pin outputs a clock with a frequency of the PLL clock divided by 1024. This is used for test purposes.
LCLK_MODE[0]	Selects the source for the LCLK PLL. '0' = LB_REF_CLK oscillator (20-40MHz). This is recommended for low and predictable LCLK clock jitter. '1' = 125MHz clock generated from the PCI Express link.

1. When the GN412x is used in an add-in card and the PCI Express clock is taken from the card edge connector, it is recommended to use LCLK_MODE[2]='0' and use an external xtal or oscillator for the local clock. This is because the host system may have highly variable clock quality and may employ spread spectrum clocking. See the schematics for the GN4124/GN4121 RDK boards for implementation details.

The local bus clock is also controlled through the CLK_CSR register. Either of the clock sources (as selected through LCLK_MODE[0]) can be divided and multiplied using a PLL to create the desired clock ratio.

Figure 3-5: GN412x Local Bus Clock Generation PLL



The local clock frequency is determined as:

$$\text{LCLK...Frequency} = \frac{\text{REFCK} \times (N + 1)}{[(M + 1) \times (O + 1)]}$$

Where N = the value or the 7 bit DIVFB register field

M = the value of the 7 bit DIVIN register field

O = the value of the 6 bit DIVOT register field

REFCK/DIVIN must be between 10 and 200MHz.

For DIVOUT values >0, VCO out must be between 500 and 1000MHz. (i.e. input to the output divider, DIVOT)

Example 3-1: To generate a 200MHz clock from the 125MHz PHY reference clock

DIVOT = 0x03 so that the PLL oscillator operates at 800MHz to generate a 200MHz clock.

DIVIN = 0x04 to divide 125MHz by 5 to get 25MHz.

DIVFB = 0x1F (31) for a 32x multiplier so that the 25MHz divided reference generates an 800MHz VCO output.

$$LCLK..Frequency \quad \frac{REFCK \times (N \quad 1)}{[(M \quad 1) \times (O \quad 1)]} \quad \frac{125MHz \times (31 \quad 1)}{[(4 \quad 1) \times (3 \quad 1)]} \quad 200MHz$$

Example 3-2: To generate a 200MHz clock from the 25MHz xtal via the LB_REF_CLK pins

Using a 25MHz xtal to generate a 200MHz LCLK is similar except that the input divider is not required.

DIVOT = 0x03 so that the PLL oscillator operates at 800MHz to generate a 200MHz clock

DIVIN = 0x00 to divide 25MHz by 1

DIVFB = 0x1F (31) for a 32x multiplier so that the 25MHz reference generates an 800MHz VCO output.

$$LCLK..Frequency \quad \frac{REFCK \times (N \quad 1)}{[(M \quad 1) \times (O \quad 1)]} \quad \frac{25MHz \times (31 \quad 1)}{[(0 \quad 1) \times (3 \quad 1)]} \quad 200MHz$$

Example 3-3: To generate a 100MHz clock from the 25MHz xtal via the LB_REF_CLK pins

Using a 25MHz xtal to generate a 100MHz LCLK is similar to the 200MHz case.

DIVOT = 0x07 so that the PLL oscillator operates at 800MHz to generate a 100MHz clock

DIVIN = 0x00 to divide 25MHz by 1

DIVFB = 0x1F (31) for a 32x multiplier so that the 25MHz reference generates an 800MHz VCO output.

$$LCLK..Frequency \quad \frac{REFCK \times (N \quad 1)}{[(M \quad 1) \times (O \quad 1)]} \quad \frac{25MHz \times (31 \quad 1)}{[(0 \quad 1) \times (7 \quad 1)]} \quad 100MHz$$

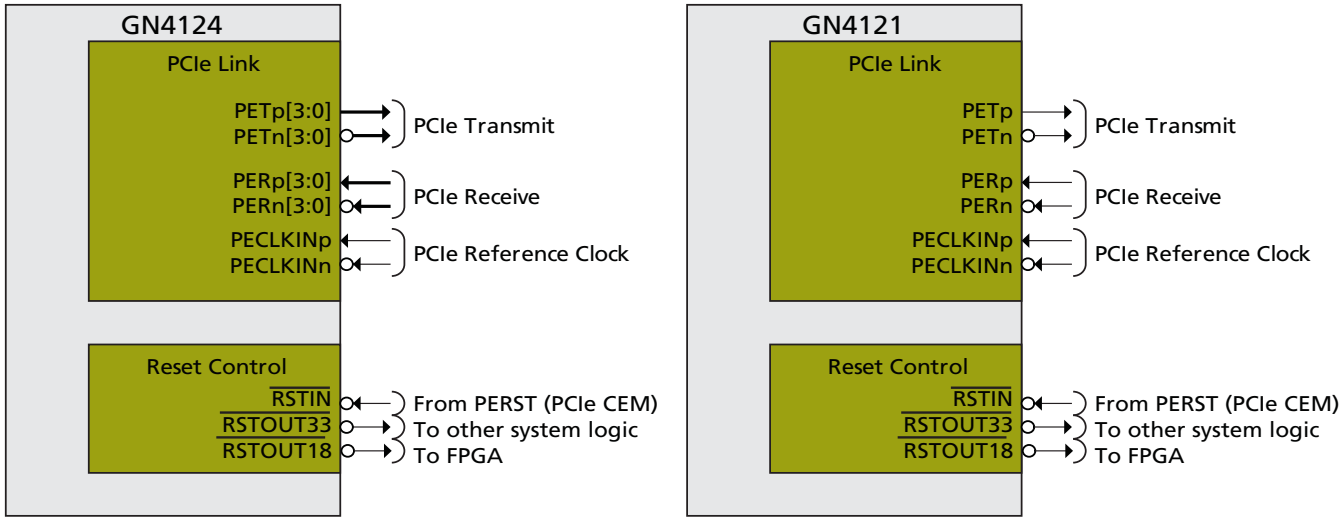
4. PCI Express Link

This section describes the PCI Express interface of the GN412x.

4.1 Physical Interface

The PCI Express signals on the GN412x are shown in Figure 4-1.

Figure 4-1: PCI Express Interface Signals



PETp/n[3:0] and PERp/n[3:0] are the high-speed transmit receive pairs for the 4 lanes of the GN4124 PCI Express interface. A 100MHz differential reference clock is also used (PECLKINp/n).

Note: The GN4121 device only has the high-speed transmit receive pairs of PETp/n and PERp/n.

The placement of high-speed signals on the GN412x is designed to route easily to a PCI Express connector as defined in the PCI Express Card Electro Mechanical Specification 1.1[1] as an endpoint device.

4.1.1 Signal Polarity Inversion

The PCI Express receive inputs on the GN4124 (PERp[3:0]/PERn[3:0]) can be connected using either the advertised polarity, or an inverted polarity. Inverted polarity may be chosen in order to simplify the PCB layout by avoiding signal crossover and additional vias.

Note: GN4121 has only PERp/PERn pins available.

The GN412x will automatically detect and compensate for polarity inversion during link training. No pin or register settings are required to facilitate this.

Polarity may be inverted independently on a per-lane basis as required.

1. See the PCI Express Card Electromechanical Specification Rev 1.1 available from www.pcisig.com.

4.1.2 PCI Express CEM Signals

When implementing a standard PCI Express endpoint card as defined by the PCI Express Card Electromechanical Specification (CEM), there are some additional signals that may be required. They are:

- **JTAG pins**: The GN412x does have a JTAG interface, and this may be implemented and daisy chained with other JTAG interfaces on the board. However, in practice this is rarely done.
- **System Management Bus (SMCLK, SMDAT)**: These are application specific and not required by the GN412x. These signals can be implemented outside of the GN412x.
- **WAKE**: This signal may be driven by the FPGA to cause re-activation of the link after power-down.
- **PRSNT1, PRSNT2**: Hot-Plug presence detect are typically connected together and require no other connection on an add-in card.

4.1.3 PCI Express PHY Control

The PCI Express PHY has several operational parameters that are controlled through the PHY_CONTROL (on page 136) register.

- **DEQ[3:0]**: Controls the transmitter de-emphasis. This value should be set to 0x8 (-3.35 dB) for typical operation. However, this value can be changed together with DTX to optimize signal integrity.
- **DTX[3:0]**: Controls the relative drive strength of the PCI Express transmitter output of the GN412x. DTX should be set to 0x2 for typical operation. However, it may be set to other values depending on the application.
- **WAKE**: This signal may be driven by the FPGA to cause re-activation of the link after power-down.
- **PRSNT1, PRSNT2**: Hot-Plug presence detect are typically connected together and require no other connection on an add-in card.

4.2 PCI Express Traffic Routing

PCI Express packets received by the GN412x are inspected and routed to the appropriate functions in the chip. This is handled in the following manner:

- Properly formed memory or IO read/write requests that cause an address match with either BAR0, BAR2, or the expansion ROM BAR will be routed to the local bus.
- Properly formed completion packets (initiated by requests from the FPGA) will be routed back to the local bus.
- BAR4 traffic (access to the internal registers of the GN412x) is serviced internally by the GN412x.
- Configuration space traffic is serviced internally by the GN412x.
- Messages received such as ‘Set Slot Power Limit’ and PME_Turn_Off handled by the on-chip Application Layer.

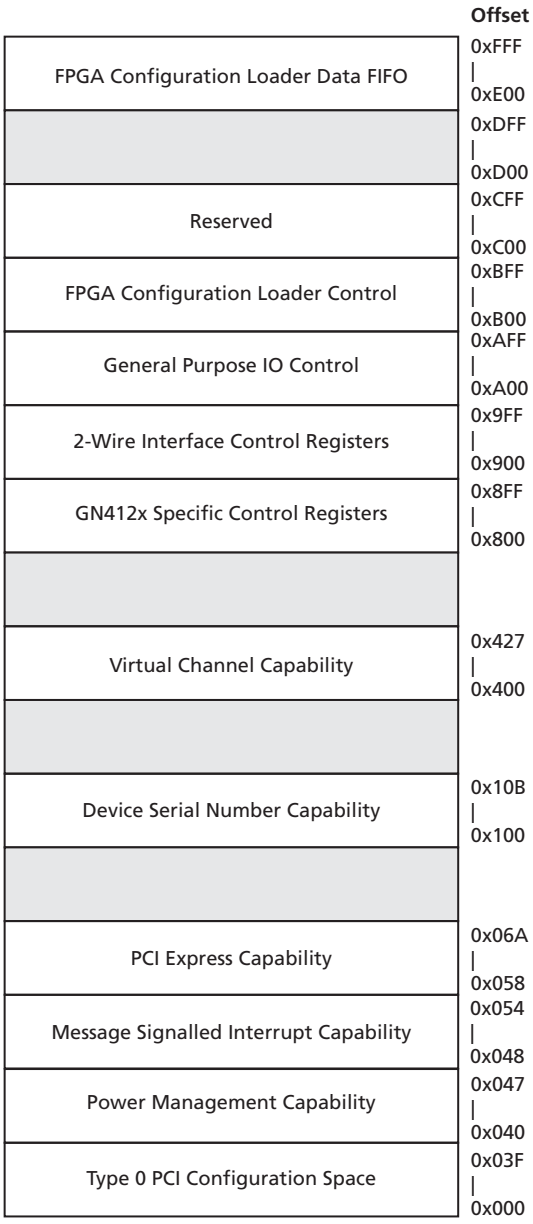
4.3 Configuration Space

The configuration space of the GN412x contains a standard type 0 PCI configuration space. Also, there are a set of additional capabilities as defined by the PCI Express Specification. These are:

- PCI Express Capability
- PCI Express Virtual Channel Capability
- MSI Capability
- Power Management Capability
- Device Serial Number Capability

In addition to the PCI/PCI Express recognized registers, there are a number of device specific registers.

Figure 4-2: Internal Register Space



4.3.1 Type 0 PCI Configuration Space

The type 0 configuration space is backwards compatible with legacy PCI and supports plug-and-play BIOS and operating systems. The Device ID (PCI_DEVICE (on page 89) register) and Vendor ID (PCI_VENDOR (on page 89) register) uniquely identify the GN412x and cannot be changed. However, system designers can uniquely identify their sub-system using the PCI_SUB_VENDOR (on page 98) and PCI_SUB_SYS (on page 99) registers[1].

1. 3.1.3 Initialization from a 2-Wire EEPROM describes how the values of internal registers such as PCI_SUB_VENDOR (on page 98) and PCI_SUB_SYS (on page 99) may be set through a serial PROM device.

Three base address registers are provided by the GN412x as part of the type 0 configuration space. They are:

- BAR0: provides a PCIe-to-Local address region in 64-bit memory space. It is controlled through PCI_BAR0_LOW (on page 94), PCI_BAR0_HIGH (on page 95), and PCI_BAR_CONFIG (on page 128).
- BAR2[1]: provides a second PCIe-to-Local address region. It is controlled through PCI_BAR2_LOW (on page 96), PCI_BAR2_HIGH (on page 97), and PCI_BAR_CONFIG (on page 128) registers.
- BAR4: provides access to the internal registers of the GN412x through PCIe memory space access. It is controlled through PCI_BAR4_LOW (on page 97) and PCI_BAR4_HIGH (on page 98) registers. The size of BAR4 is fixed at 4096 bytes.

4.3.1.1 Plug-and-Play Operation

Plug-and-Play (PnP) is the process of having the host system BIOS and/or operating system scan for endpoint devices, initialize them, and determine the proper driver to load. Proper driver identification will typically require at least several key registers to be set properly:

- **Vendor ID**: This is a 16 bit value that indicates the vendor of the chip. In this case, Gennum is the chip maker, and so the Gennum Vendor ID of 0x1A39 is used. Vendor IDs are determined by the PCI SIG and allocated to member companies. The PCI_VENDOR register at offset 0x0000 in the type zero configuration space of the GN412x contains the Vendor ID.
- **Device ID**: This is a 16 bit value that is used to uniquely identify the chip model provided by the silicon vendor. In the case of the GN412x, the value is 0x0004 and found at location 0x0002 (the PCI_DEVICE register) in the type zero configuration space.
- **Subsystem Vendor ID**: This is a 16 bit value that indicates the subsystem (board) vendor. This should be set to the PCI SIG assigned Vendor ID of the board manufacturer. The Subsystem Vendor ID register is located at offset 0x002C (the PCI_SUB_VENDOR (on page 98) register). It should be pre-loaded from the EEPROM with the correct value.
- **Subsystem ID**: This is 16 bit value that indicates the subsystem (board) that is being made by the manufacturer. The Subsystem ID register is located at offset 0x002E (the PCI_SUB_ID register). It should be pre-loaded from the EEPROM with the manufacturer assigned value.

When used in a closed embedded system, the PCI_SUB_VENDOR (on page 98) and PCI_SUB_SYS (on page 99) values may not be required. This will depend on the embedded system OS and software.

When the GN412x is used in a plug-and-play environment, the PCI_SUB_VENDOR (on page 98) and PCI_SUB_SYS (on page 99) register must be populated with the Vendor ID belonging to the developer of the subsystem. The Gennum Vendor ID must not be used

1. **Note**: There is no BAR1 as the register address for BAR1. It becomes the upper address base bits for BAR0.

without permission of Gennum. A valid Vendor ID may be obtained from the PCI Special Interest Group (www.pcisig.com).

For board developers who are not PCI SIG members, Gennum will allow the limited use of the Gennum Vendor ID providing that the developer signs a sub-licensing agreement that will restrict the use of the Gennum Vendor ID to a small range of Subsystem IDs. That way, Gennum can avoid having multiple board vendors advertising the same Subsystem Vendor ID/Subsystem ID combination to the host OS, as this can cause driver conflicts.

4.3.1.2 PCIe-to-Local BAR Operation

The PCI-to-local BARs (BAR0 and BAR2) are 64-bit memory BARs. The size of BAR0 and BAR2 are independently determined by the corresponding size fields in the PCI_BAR_CONFIG (on page 128) register. BARs may be up to 4GB in size. However, 32-bit operating systems will be unable to map a region of this size. Consequently, large BAR sizes should only be used in 64 bit mode.

During enumeration of a PCIe device, the operating system or BIOS will write '1' into all bits of a BAR in order to determine its size. The number of upper bits that read back as '1' indicate the size of the bar. For example, if BAR0 is written with 0xFFFFFFFF, and then reads back as 0xFF000000; then it has a size of 16MB, since bits 23:0 are masked off (224=16MB). A "BAR hit" requires bits 31:24 to match the contents of the corresponding BAR contents.

The size of BAR0/BAR2 is determined by the contents of the PCI_BAR_CONFIG (on page 128) register and that size determines which bits in PCI_BAR0_LOW (on page 94) / PCI_BAR2_HIGH (on page 97) get masked off.

4.3.2 PCI Express Capability

The PCI Express Capability registers are required for PCI Express devices. Registers in the GN412x associated with the PCI Express Capability are:

- PCIE_CAP_ID (on page 109): PCI Express Capability ID
- PCIE_NEXT_ID (on page 109): PCI Express Next Capability Pointer
- PCIE_CAPABILITY (on page 109): PCI Express Capabilities
- PCIE_DEVICE_CAP (on page 110): PCI Express Device Capabilities
- PCIE_DCR (on page 111): PCI Express Device Control
- PCIE_DSR (on page 112): PCI Express Device Status

4.3.3 Power Management Capability

The power management capability is implemented in the registers that begin with PM_. These are:

- PM_CAP_ID (on page 102): Power Management Capability ID
- PM_NEXT_ID (on page 102): Power Management Next Capability Pointer
- PM_CAP (on page 103): Power Management Capability Register

- PM_CSR (on page 104): Power Management Control/Status Register
- PM_CSR_BSE (on page 105): PM_CSR Bridge Support Extensions Register
- PM_DATA (on page 105): Power Management Data Register (works in conjunction with PM_CSR register)

The power management state of the GN412x can be communicated to an attached FPGA through one of the multi-use GPIO pins. The FPGA can then use this state to prepare to enter powerdown or a low power state. Techniques such as clock slow down, clock gating, or power down of circuitry may be employed as is suitable for a given application.

The FPGA must not initiate any further requests if the device is in a low power state as programmed by the host.

In the event of a PCI Express PME_Turn_Off message being received, the GN412x will acknowledge the message and prepare for shutdown. This will assert the low power control pin to the FPGA, so that it can ready the subsystem for loss of power.

4.3.4 Device Serial Number Capability

The device serial number capability allows a 64-bit serial number to be implemented. This would typically be stored and loaded from an attached serial EEPROM as described in 3.1.3 Initialization from a 2-Wire EEPROM.

Register names starting with DSN_ identify them as device serial number capability registers. They are:

- DSN_CAP (on page 116): PCI Express Device Serial Number Capability
- DSN_LOW (on page 116): PCI Express Device Serial Number Lower Bits
- DSN_HIGH (on page 117): PCI Express Device Serial Number Upper Bits

4.3.5 Virtual Channel Capability

The GN412x supports up to two virtual channels. Each virtual channel has independent buffering. This prevents one virtual channel from stalling traffic from another virtual channel.

Register names starting with VC_ identify them as virtual channel capability registers. They are:

- VC_CAP (on page 118): Virtual Channel Capability
- VC_PORT_CAP_1 (on page 118): Virtual Channel Port Capability - Register 1
- VC_PORT_CAP_2 (on page 119): Virtual Channel Port Capability - Register 2
- VC_PCR (on page 119): Virtual Channel Port Control Register
- VC_PSR (on page 120): Virtual Channel Port Status Register
- VC_RESOURCE_CAP0 (on page 120): Virtual Channel 0 Resource Capability Register
- VC_RESOURCE_CAP1 (on page 122): Virtual Channel 1 Resource Capability Register

- VC_RESOURCE_CR0 (on page 121): Virtual Channel 0 Resource Control Register
- VC_RESOURCE_CR1 (on page 123): Virtual Channel 1 Resource Control Register
- VC_RESOURCE_SR0 (on page 122): Virtual Channel 0 Resource Control Register
- VC_RESOURCE_SR1 (on page 124): Virtual Channel 1 Resource Control Register

4.3.5.1 Virtual Channel Setup

The GN4121 supports a single virtual channel. Consequently, the VC_COUNT bits in the VC_PORT_CAP_1 register should be set to 0x0. This will advertize to the host OS that the GN4121 endpoint has only a single VC (the default VC0). By default, VC_COUNT is set to 0x0 upon hardware reset.

The GN4124 also defaults to a single VC. In order to enable it be recognized by the host root complex as being capable of supporting a second VC, the VC_COUNT bits in the VC_PORT_CAP_1 register should be set to 0x1 prior to system enumeration[1]. This will advertize to the host OS that the GN4124 endpoint has an additional VC (VC1) beyond the default VC0. While the GN4124 may advertize 2 VCs, the root complex and/or switches may not support VC1. In this case, the OS will not enable VC1. This can be determined by reading the ENABLE bit in the VC_RESOURCE_CR1 register. Please note that changing of any of the virtual channel settings requires the enabling of the Device Serial Number capability. Also, enabling of the VC1 on the GN4124 requires the enabling of the Device Serial Number capability.

The changing of any virtual channel reqires the enabling of the Device Serial Number capability. Also, enabling of the VC1 on the GN4124 requires the enabling of the Device Serial Number capability. Refer to DSN_CAP (on page 116).

4.3.6 MSI Capability

Message Signaled Interrupts (MSI) is an optional feature that enables a device function to request service by writing a system-specified data value to a system-specified address (using a PCI DWORD memory write transaction). The Message Signalled Interrupt capability is used to identify and configure an MSI capable device.

The MSI Registers begin with MSI_ and are:

- MSI_CAP_ID (on page 106): MSI Capability ID
- MSI_NEXT_ID (on page 106): MSI Next Capability Pointer
- MSI_CONTROL (on page 106): MSI Message Control Register
- MSI_ADDRESS_LOW (on page 107): MSI Lower Address
- MSI_ADDRESS_HIGH (on page 107): MSI Upper Address
- MSI_DATA (on page 108): MSI Message Data

1. This is typicaly accomplished using the 2-wire EEPROM. See Section 3.1.3 on page 22.

4.4 Bridge Operation

This section describes how PCI Express transactions are translated into local bus transactions and vice versa. It also describes how data is buffered.

4.4.1 Flow Control

The PCI Express link must track credits for flow control. Flow control information is transferred using Flow Control Packets (FCPs), which are a type of data link layer packet. Each virtual channel has independent flow control.

There are six types of information tracked by flow control for each virtual channel.

Table 4-1: Flow Control Credits Advertised by the GN412x

Mnemonic	Setting	Description
PH	16	Posted Request headers
PD	90	Posted Request Data payload
NPH	2	Non-Posted Request headers
NPD	2	Non-Posted Request Data payload
CPLH	infinite	Completion headers
CPLD	infinite	Completion Data payload

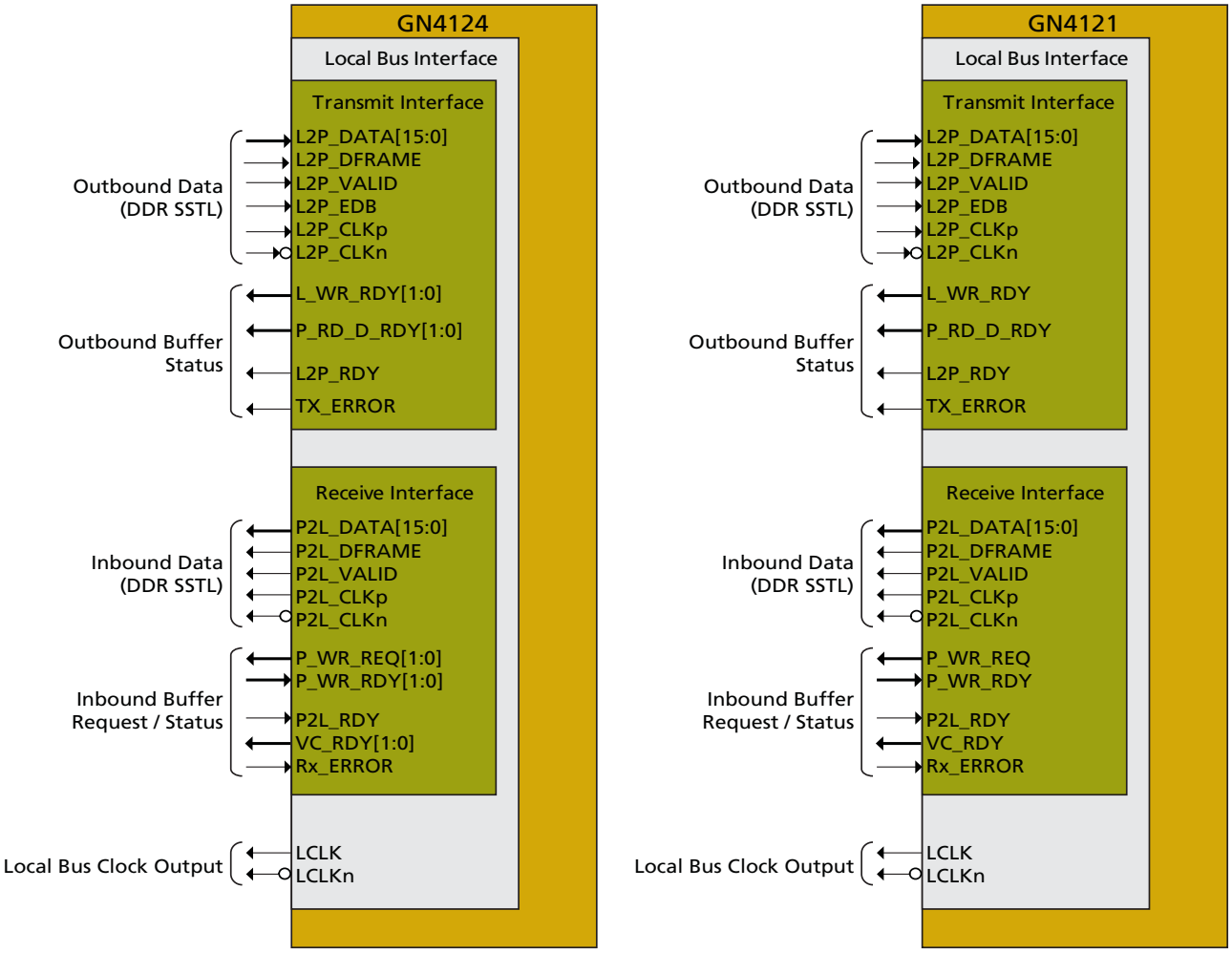
5. Local Bus Interface

This section describes the local bus signals and protocol of the GN412x. For detailed timing information, refer to Local Bus Timing section in the GN412x Data Sheet.

5.1 Local Bus Signals

The local bus interface signals are shown in Figure 5-1.

Figure 5-1: Local Bus Interface Signals



The local bus interface of the GN412x is composed of two primary sets of high-speed signals:

- Data and control signals associated with traffic received by the PCI Express link on the GN412x (Inbound).
- Data and control signals associated with traffic transmitted by the PCI Express link on the GN412x (Outbound).

Within these two groups of signals, there are two main signal types:

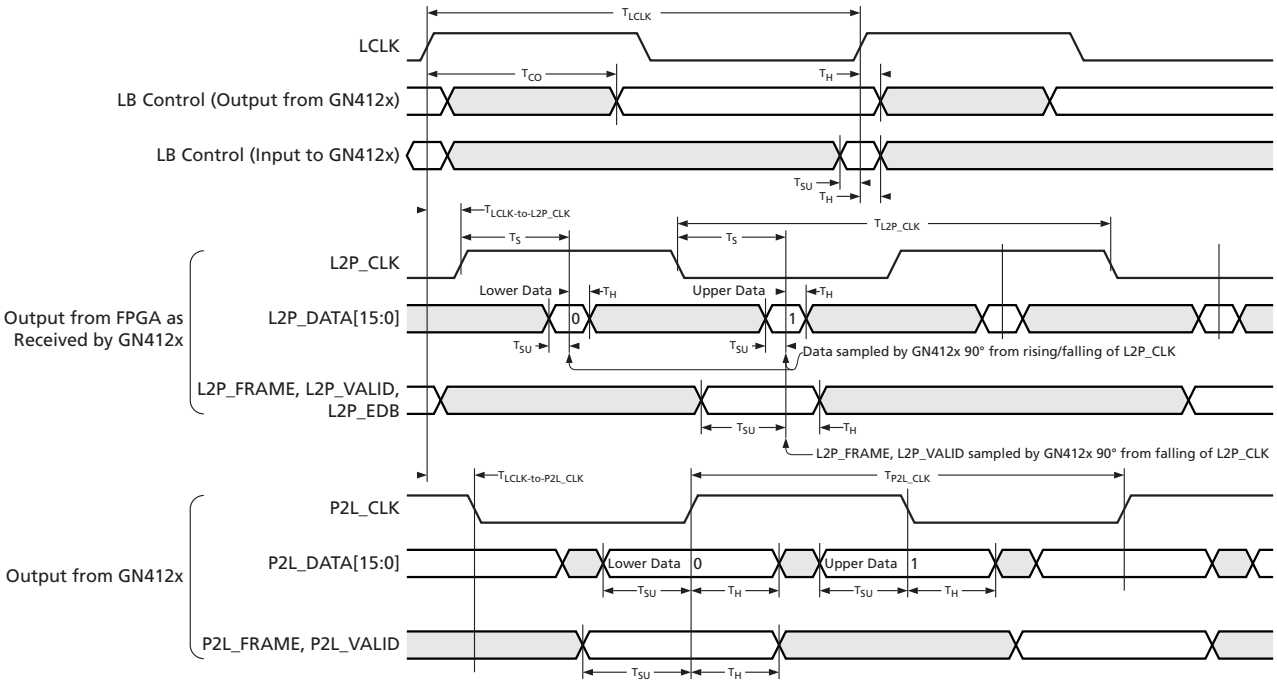
- Control signals: SSTL IO that operate as single data rate signals. Most of the input signals may be driven asynchronously by the FPGA and are synchronized within the GN412x.
- Data signals are SSTL IO that operate up to 400MT/s[1] using a source synchronous clock. They operate pseudo synchronously to the control signals and LCLK. Their source synchronous clock runs at exactly the rate of LCLK with valid data on both the rising and falling edge of the SSC.

All local bus signals are unidirectional. Outputs from the GN412x are always driven (not floated as high impedance) under normal operating conditions[2].

The local bus control signals are fully registered in and out. Consequently, there is always at least a 2 clock delay from the GN412x receiving a signal and then responding at the output. The fully registered interface ensures that the full LCLK period is available for signals to propagate from the GN412x to an attached FPGA without additional insertion delays of combinatorial logic.

The timing relationships of the DDR and SDR local bus signals are shown in Figure 5-2.

Figure 5-2: Local Bus Timing



The full details of the GN4124 and GN4121 local bus timing is found in the associated Data Sheets.

1. MT/s = mega-transfers per second.
2. The local bus control signal outputs from the GN412x are either driven LOW or become HIGH impedance when RSTIN is asserted.

5.1.1 L2P DDR Source Synchronous Signals

An attached FPGA will use the signals L2P_CLKp/n, L2P_DATA, L2P_DFRAME, and L2P_VALID to send data to the GN412x. As Figure 5-2 depicts, the GN412x applies a 90° phase shift (T_S) to L2P_CLK when sampling L2P_DATA. This simplifies the FPGA circuit and removes the necessity of providing clock / data alignment in the FPGA at the cost of additional clock management resources.

5.1.2 P2L DDR Source Synchronous Signals

The GN412x sends data to the attached FPGA using the signals P2L_CLKp/n, P2L_DATA, P2L_DFRAME, and P2L_VALID. In order to simplify the FPGA receive circuit, the GN412x aligns P2L_DATA to the rising / falling edges of P2L_CLK. This is represented in the lower section of Figure 5-2, where a setup / hold window (T_{SU}, T_H) is provided by the GN412x.

By the time the clock and data reach the FPGA, the setup / hold window will be reduced by lane-to-lane skew across the P2L_DATA bus. Consequently, it is important to match the trace delay across the signals P2L_CLK, P2L_DATA, P2L_DFRAME, and P2L_VALID.

More detailed description of the local bus timing, including AC timing parameters, is provided in the respective GN4124 and GN4121 Data Sheets.

5.1.3 Signal Naming Conventions

In order to simplify understanding of the local bus signals, there are some basic naming rules used:

- Signals beginning in L_ are associated with operations where a local bus agent is the “requester” or “master” in the legacy PCI sense.
- Signals beginning in P_ are associated with operations where an external PCI Express agent is the “requester”.
- Signals with the prefix L2P_ are associated with any transaction where information flows from the local bus towards the PCI Express interface using the DDR SSTL inputs in the GN412x.
- Signals with the prefix P2L_ are associated with any transaction where information flows from the PCI Express interface towards the local bus using the DDR SSTL outputs on the GN412x.

5.1.4 Local Bus Clocking

Options for local bus clocking are described in 3.4.2 Local Clock.

5.1.5 Local Bus Control Signals

There are two main sets of control signals:

- **PCI Express-to-Local Target Control Signals**: These control data flow for when an external PCIe agent initiates read or write cycles to the GN412x. In this case, the GN412x is a PCIe “target” device.

- **Local-to-PCI Express Master Control Signals**: These control data flow for when the GN412x initiates read or write cycles to an external PCIe device. In this case, the GN412x is a PCIe “master” device.

The control signals are generally in the form of:

- ***_REQ (request)**: The GN412x wants to send data to the FPGA.
- ***_RDY (ready)**: The receiver of the particular data type is able to accept that type of data. This signal is used to throttle data to prevent buffer overflow. RDY may be asserted prior to a REQ being asserted so that when REQ is asserted, the data can begin to transfer immediately. There are multiple RDY signals: one for each type of transaction (or virtual channel in some cases) rather than having one combined RDY. This prevents a transaction that uses one set of buffering resources that are busy from blocking transactions from another resource where buffers are available.
- ***_FRAME (data frame)**: This indicates that the sender of the data is sending data. FRAME is pulsed in a manner similar to parallel PCI or PCI-X. It is asserted together with the first data in the packet and released one clock before the packet ends. This provides an easy START and END of packet indicator.

There are two sets of RDY signals: one per virtual channel. This allows transaction of different traffic classes to be separately buffered, so that high priority traffic may pass that of lower priority. For example, a high priority traffic class can be routed to virtual channel 1 and lower priority traffic to VC0. The mapping of traffic classes to virtual channels is controlled through the Virtual Channel Resource Control Registers VC_RESOURCE_CR0 (on page 121) and VC_RESOURCE_CR1 (on page 123).

Each virtual channel has its own set of buffers to prevent data blockage on one VC from blocking data flow on the other VC.

5.1.6 Signals for Inbound Dataflow (PCIe-to-Local)

Packets that originate from the PCIe link and are sent to the local bus interface are referred to as PCIe-to-Local packets or inbound packets. This includes the following types of packets:

- **PCIe-to-local Target Writes**: When an external PCIe agent, acting as a requester, initiates a write transaction, this is a PCI-to-local target write.
- **PCIe-to-local Target Read Request**: This is effectively the address phase of a read operation that is initiated by an external PCIe requester. The data phase (completion packet) uses the outbound (local-to-PCIe) direction.
- **Local-to-PCIe Master Read Completion**: This is effectively the data phase for a previously issued Local-to-PCIe master read request. The address phase is initiated by the local bus attached FPGA, where the GN412x acts as a PCIe requester.

The signals associated with the above type of traffic are detailed in Table 5-1.

Table 5-1: Signals for Inbound PCIe to Local Packet Flow

Name	Definition	Type and Direction[1]	Description of Operation
P2L_DATA[15:0]	PCIe to Local Data Synchronous to P2L_CLKp/n	DDR SSTL Output	**GN412x**: Packet data received by the GN412x, processed and forwarded to the local bus. The GN412x always processes data in 32-bit words. That is, P2L_DATA[15:0] is always driven as groups of 4 bytes.
			FPGA: Data for PCI-to-local target writes, PCI-to-local target read requests, and local-to-PCI master read responses.
P2L_DFRAME	PCIe to Local Data Frame Synchronous to P2L_CLKp/n	DDR[2] SSTL Output	**GN412x**: A version of P2L_FRAME that is synchronized to the source synchronous P2L_CLK.
			FPGA: Used to synchronize the START / END of packets with respect to P2L_CLK, so that P2L_DATA[15:0] can be recovered into the LCLK domain.
P2L_VALID	PCIe to Local Data Valid Synchronous to P2L_CLKp/n	DDR SSTL Output	**GN412x**: Asserted by the GN412x to indicate that valid data is being driven onto the P2L_DATA during a P2L_DFRAME cycle. The GN412x will only de-assert this signal, when P2L_RDY is de-asserted.
			FPGA: Used to qualify the sampling of P2L_DATA during a P2L_DFRAME cycle.
P2L_CLKp/n	PCIe to Local Clock	Differential SSTL Output	**GN412x**: A dual data rate, source synchronous clock used to synchronize P2L_DATA[15:0], P2L_DFRAME, and P2L_VALID.
			FPGA: The rising and falling edges of P2L_CLK are used to sample P2L_DATA[15:0].
P_WR_REQ[1:0] for GN4124 OR P_WR_REQ for GN4121	PCIe Write Request Synchronous to LCLK	SSTL Output	**GN412x**: This signal is asserted by the GN412x whenever there is a valid incoming PCIe write request corresponding to one of the base address registers (BAR0 or BAR2[3]). There is one request signal per virtual channel.
			FPGA: Used to determine that there is a PCIe target write request. The FPGA needs to fulfil the request by asserting P_WR_RDY.
P_WR_RDY[1:0] for GN4124 OR P_WR_RDY for GN4121	PCIe Write Ready Sampled by LCLK but may be driven asynchronously	SSTL Input	**GN412x**: Used to throttle PCI-to-local write cycles.
			FPGA: Driven by the FPGA whenever it is able to accept a PCIe target write request.
P2L_RDY	PCIe-to-Local Buffer Ready Sampled by LCLK but may be driven asynchronously	SSTL Input	**GN412x**: Used to pause the activity of a data transfer on the P2L_DATA bus.
			FPGA: This signal is normally driven active by the FPGA. However, P2L_RDY may be de-asserted to provide a temporary halt in data transfer over P2L_DATA when a packet is already in progress.

1. Direction is relative to the GN412x.
2. P2L_DFRAME and P2L_VALID are sampled by the DDR clock (P2L_CLK). However, they are to be applied to each 32 bits of data transfer.
3. Configuration transactions and BAR4 transactions are handled by the GN412x without any direct interaction with the local bus.

5.1.7 Signals for Outbound Dataflow (Local-to-PCIe)

Packets that move from the local bus towards the PCIe link are referred to as Local-to-PCIe packets or outbound packets. This includes the following types of packets:

- **Local-to-PCIe Master Writes**: When a circuit on the local bus acts as a PCIe master, and initiates a write transaction, this is a local-to-PCIe master write.
- **Local-to-PCIe Master Read Request**: This is effectively the address phase of a read operation on the PCIe link. The data phase uses the inbound (PCIe-to-local) direction.
- **PCIe-to-local Target Read Completion**: This is effectively the data phase for a previously issued PCIe-to-local target read request.

The signals associated with the above type of traffic are detailed in Table 5-2.

Table 5-2: Signals for Outbound Local to PCIe Packet Flow

Name	Definition	Type and Direction[1]	Description of Operation
L2P_DATA[15:0]	Local-to-PCIe Data Synchronous to L2P_CLKp/n	DDR SSTL Input	**GN412x**: Packet data that is outbound to the PCI Express interface is received on these pins.
			FPGA: Data for local-to-PCI master writes, local-to-PCI master read requests, and PCI-to-local target read completions are sent by the FPGA on these pins.
L2P_DFRAME	Local-to-PCIe Data Frame Synchronous to L2P_CLKp/n	DDR[2] SSTL Input	**GN412x**: Used to synchronize the START/END of packets with respect to L2P_CLK, so that L2P_DATA[15:0] can be recovered into the LCLK domain.
			FPGA: The FPGA asserts this signal to indicate the START of a packet, and then de-asserts it during the last 32 bits of the packet.
L2P_VALID	Local-to-PCIe Data Valid Synchronous to L2P_CLKp/n	DDR SSTL Input	**GN412x**: Used to qualify the sampling of L2P_DATA during a L2P_DFRAME cycle.
			FPGA: Asserted by the FPGA to indicate that valid data is being driven onto the L2P_DATA during a L2P_DFRAME cycle. The FPGA should only de-assert this signal when L2P_RDY is de-asserted.
L2P_EDB	Local-to-PCIe Data Valid	SSTL Input	**End-of-Packet Bad Flag**: When a packet is considered bad, it is terminated with EDB.
L2P_CLKp/n	Local-to-PCIe Clock	Differential SSTL Input	**GN412x**: A dual data rate, source synchronous clock used to sample L2P_DATA[15:0], L2P_DFRAME, and L2P_VALID. L2P_DATA[15:0] are sampled by the GN412x, 90° from the rising and falling edges of L2P_CLK.
			FPGA: L2P_CLK is sent from the FPGA with the same timing as the L2P_DATA[15:0] and L2P_DFRAME signals.
L_WR_RDY[1:0] for GN4124 OR L_WR_RDY for GN4121	Local-to-PCIe Write Ready Driven by LCLK but may be treated as asynchronous	SSTL Output	**GN412x**: Driven by the GN412x whenever it is able to accept a Local-to-PCIe master write for a given VC.
			FPGA: Used to throttle local-to-PCIe write cycles.

Name	Definition	Type and Direction[1]	Description of Operation
P_RD_D_RDY[1:0] for GN4124 OR P_RD_D_RDY for GN4121	PCIe-to-Local Read Response Data Ready Driven by LCLK but may be treated as asynchronous	SSTL Output	**GN412x**: Driven by the GN412x whenever it is able to accept the response data from a PCIe-to-Local target read request. This signal is usually driven active to indicate that the GN412x is ready to accept a read request. However, it may be de-asserted briefly during PCIe access to internal registers or during MSI events.
			FPGA: Used to throttle read requests. The FPGA will not initiate transfer of read response data, unless P_RD_D_RDY is asserted.
L2P_RDY	Local-to-PCIe Buffer Ready Should be treated as asynchronous	SSTL Output	**GN412x**: This signal is normally driven active by the GN412x. However, L2P_RDY is de-asserted if the buffer receiving L2P_DATA is about to become full.
			FPGA: Used to pause the activity of a data transfer on the L2P_DATA bus. When L2P_RDY is de-asserted by the GN412x, the FPGA should stop transferring data by de-asserting L2P_VALID within 3 LCLK cycles.
TX_ERROR	Receive Error Should be treated as asynchronous	SSTL Input	**GN412x**: Asserted by the GN412x whenever it detects an unexpected transaction or malformed packet in the L2P direction. This also signals an overflow condition on the incoming FIFO.
			FPGA: Optionally used by the FPGA for debugging purposes. Under normal operation, the GN412x will never have cause to assert TX_ERROR.

1. Direction is relative to the GN412x.
2. L2P_DFRAME and L2P_VALID are sampled by the DDR clock (L2P_CLK). However, they are applied to each 32 bits of data transfer.

Note that there are no request signals for the local-to-PCI direction. That is because the GN412x will assert the ready signal for each channel as soon as there is buffer space available. The attached FPGA can decide the priority and order of transfers in the outbound direction by simply asserting L2P_DATA, L2P_DFRAME etc. as required.

5.2 Local Bus Protocol

The local bus protocol is designed to optimize use of the P2L_DATA and L2P_DATA signals by minimizing idle cycles and transaction start-up latency. This is accomplished by having multiple RDY signals to differentiate transaction types and virtual channels. With multiple RDY signals, traffic that is backed up on one virtual channel or transaction type will not block transfer of another VC or transaction type.

All local bus data traffic happens in multiples of 4 bytes aligned on a 32-bit boundary. Data that is not aligned or is in quantities of less than 4 bytes, is accomplished through byte enables that are part of the header. Read and write data bytes are always aligned to their lane. Consequently, a write of 4 bytes to address 3 would be accomplished by a data payload size of 8 bytes (64 bits), where the first 3 bytes (bytes 2:0) would be disabled, and the last byte (byte 7) would also be disabled. Bytes 6:3 would be enabled.

Since byte lane enables are employed, bits 1:0 of address information is not used.

There are four types of local bus transactions using six local bus packet types, as indicated in Table 5-3. Write transactions on the local bus are always posted. Consequently, there is one packet type involved per write transaction. Reads involve a request from one direction and a completion from the other direction.

Table 5-3: Summary of Local Bus Transaction Types

Transaction Type	Packet Type	Requester	Initiator	Direction	Signals
PCIe-to-local Target Write	PCIe-to-local Target Write	PCIe	PCIe	Inbound (Rx)	P2L_DATA[15:0] P2L_DFRAME P2L_CLK P_WR_REQ[1:0] (for GN4124) OR P_WR_REQ (for GN4121) P_WR_RDY[1:0] (for GN4124) OR P_WR_RDY (for GN4121)
PCIe-to-local Target Read	PCIe-to-local Target Read Request (address phase)	PCIe	PCIe	Inbound (Rx)	P2L_DATA[15:0] P2L_DFRAME P2L_CLK
	PCIe-to-local Target Read Completion (data phase)	PCIe	Local	Outbound (Tx)	L2P_DATA[15:0] L2P_DFRAME L2P_CLK P_RD_D_RDY[1:0] (for GN4124) OR P_RD_D_RDY (for GN4121)
Local-to-PCIe Master Write	Local-to-PCIe Master Write	Local	Local	Outbound (Tx)	L2P_DATA[15:0] L2P_DFRAME L2P_CLK L_WR_RDY[1:0] (for GN4124) OR L_WR_RDY (for GN4121)
Local-to-PCIe Master Read	Local-to-PCIe Master Read Request (address phase)	Local	Local	Outbound (Tx)	L2P_DATA[15:0] L2P_DFRAME L2P_CLK
	Local-to-PCIe Master Read Completion (data phase)	Local	PCIe	Inbound (Rx)	P2L_DATA[15:0] P2L_DFRAME P2L_CLK

5.2.1 Header Format

Transactions on the local bus are in the form of packets. Packets always begin with a header to indicate the type of transaction and attributes for the transaction. When the transaction is a read or write request, an address will also be included. The header may be followed by data, as in the case for writes and read completions.

The following tables depict the header layout for local bus transactions. See Table 5-14 for the interpretation of the fields in the headers.

5.2.1.1 Headers on the P2L Bus

This section covers headers generated by the GN412x and sent to the FPGA.

Table 5-4: Header Format for P2L Writes (Target Writes)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC			S	TYPE=0x2				LBE				FBE							V			LENGTH									
ADDRESS[31:2]																														B	

Table 5-5: Header Format for P2L Read Requests (Target Read Request)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC			S	TYPE=0x0				LBE				FBE							V	CID		LENGTH									
ADDRESS[31:2]																														B	

Table 5-6: Header Format for L2P Read Completions (Master Read Completion Without Data)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC				TYPE=0x4										STAT		L			V	CID		LENGTH									
Reserved																															

Table 5-7: Header Format for L2P Read Completions (Master Read Completion With Data)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC				TYPE=0x5										STAT		L			V	CID		LENGTH									

5.2.1.2 Headers on the L2P Bus

This section covers headers generated by the FPGA and sent to the GN412x

Table 5-8: Header Format for L2P Writes (32 bit Address Specified)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC			S	TYPE=0x2				LBE				FBE							V			LENGTH									
ADDRESS[31:2]																															

Table 5-9: Header Format for L2P Writes (64 bit Address Specified)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC			S	TYPE=0x3				LBE				FBE							V			LENGTH									
ADDRESS[63:32]																															
ADDRESS[31:2]																															

In order to be PCI Express compliant, 32 bit writes should be used when 64 bit addressing is supported and address bits 63:32 are all zero. The L2P write header generates a PCI Express header of the same address size. The use of a 64 bit header where A(63:32) = 0x00000000 is not PCI Express compliant.

Table 5-10: Header Format for L2P Read Requests (32 bit Address Specified)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC			S	TYPE=0x0				LBE				FBE							V	CID		LENGTH									
ADDRESS[31:2]																															

Table 5-11: Header Format for L2P Read Requests (64 bit Address Specified)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC			S	TYPE=0x1				LBE				FBE							V	CID		LENGTH									
ADDRESS[63:32]																															
ADDRESS[31:2]																															

In order to be PCI Express compliant, 32 bit read requests should be used when 64 bit addressing is supported and address bits 63:32 are all zero. The L2P read request header generates a PCI Express header of the same address size. The use of a 64 bit header where A(63:32) = 0x00000000 is not PCI Express compliant.

Table 5-12: Header Format for P2L Read Completions (Target Read Completion Without Data)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC				TYPE=0x4										STAT		L			V	CID		LENGTH									
Reserved																															

Table 5-13: Header Format for P2L Read Completions (Target Read Completion With Data)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC				TYPE=0x5										STAT		L			V	CID		LENGTH									

The fields for the above are interpreted in Table 5-14.

Table 5-14: Local Bus Header Fields

# Bits	Mnemonic	Description
2	STAT	**Completion Status**: • "00" Successful completion • "01" Unsupported requests • "10" Completer abort • "11" Completion time-out
1	S	**No Snoop Attribute**: mapped to the No Snoop bit in the PCI Express header.
4	TYPE	**Header Type**: Determines the transaction type as listed below: • "0000" memory read, 32-bit address[1] • "0001" memory read, 64-bit address • "0010" memory write, 32-bit address • "0011" memory write, 64-bit address • "0100" Completions without data • "0101" Completions with data • others Reserved
4	LBE	**Last Byte Enable**: The active high byte enables for the last aligned 32-bit word of a transfer.
4	FBE	**First Byte Enable**: The active high byte enables for the first aligned 32-bit word of a transfer. If only a single word is transferred, FBE is used. The least significant bit of FBE (FBE[0]) is related to the low byte of a 32 bit word (bits 7:0). For example, a byte write to system address 0x0 would result in FBE[3:0]="0001", while a byte write to system address 0x3 would result in FBE[3:0]="1000".
1	V	**Virtual Channel Number**: Contains a '0' for VC0 and '1' for VC1. Packets sent to the GN4121 should always have V set to '0'.
2	CID	**Completion Identifier**: Used to match a read request with a completion. This bit field is used only for read transactions. In the case of a PCIe-to-Local Target Read Request, CID is generated by the GN412x. The Local-to-PCI Master Read Completion CID is originally generated by the FPGA during the Read Request phase of the transaction and then sent back to the FPGA with the completion data.
1	L	When asserted '1', it denotes last completion of a split completion. This is required when a larger request is broken into smaller packets by the completer or a packet switch in the path of the completion. If "L" is not asserted, then the completion is not complete.
3	TC	**Traffic Class**: Traffic Class is a Transaction Layer Packet label that is transmitted unmodified end-to-end through the fabric. At every service point (e.g. switch) within the fabric, Traffic Class labels are used to apply appropriate servicing policies. Each Traffic Class label defines an unique ordering domain - no ordering guarantees are provided for packets that contain different Traffic Class labels.
10	LENGTH	**Length of the Data Portion of the Packet in 32-bit Multiples**: The value of 0x000 represents the maximum length of 1024 DWORDS / 4096 bytes.
32/64	ADDRESS	**Address of the Start Location of the Data**: Address is always aligned to a 32-bit boundary, so that bits 1:0 are not required. Byte addressability is provided through the byte enables (FBE, LBE).

# Bits	Mnemonic	Description
2	B	**Indicates the Base Address Register Hit**: It is encoded as: • "00" = BAR0 • "01" = BAR2. • "10" = Expansion ROM • "11" = Reserved

Note: Shaded fields are reserved.

1. When the upper 32 bits of a 64 bit address is 0x00000000, then the 32 bit read/write type should be used. Otherwise, the PCI Express request will be a 64 bit request type with A[63:32]=0x00000000, and this violates the specification. The GN412x will not convert a 64 bit request to a 32 bit address, even if A[63:32]=0x00000000.

5.2.2 PCI Express Target Transactions

A PCI Express target transaction is a read or write operation that is initiated by an external PCI Express agent, such as a host device / root complex attached to the GN412x.

5.2.2.1 PCI Express Target Write Transaction

When a PCIe agent writes data to a local bus device, this is referred to as a PCIe target write or alternately a PCIe-to-local write.

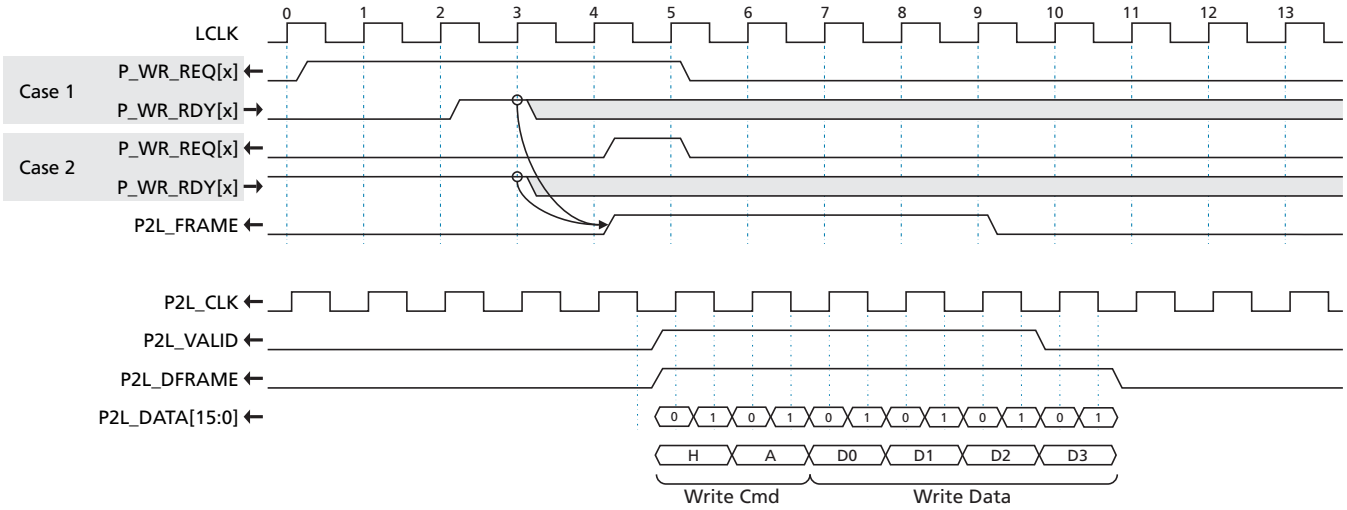
Figure 5-3 shows a burst write transaction. Two cases are shown to illustrate the operation of P_WR_RDY. The first case illustrates the situation where P_WR_RDY is de-asserted. The request is asserted at LCLK 0. If P_WR_RDY is responding to the request, then the earliest it can respond is LCLK 2. The GN412x then asserts P2L_FRAME[1] 2 clocks later (LCLK 4) and the data lanes follow suit.

In the second case, P_WR_RDY has been asserted well in advance of the request, so that the GN412x can immediately assert P2L_FRAME along with P_WR_REQ (LCLK 4). The GN412x may de-assert P_WR_REQ just after the assertion of P2L_FRAME, if there are no further transaction pending after the current transaction completes.[2]

Note: For the GN4124 device, there are the pins: P_WR_RDY[1:0], and P_WR_REQ[1:0], while for the GN4121 device, there are the pins: P_WR_RDY and P_WR_REQ.

1. P2L_FRAME is a version of P2L_DFRAME that is synchronized to LCLK. It is used here as a reference to show how the internal state machine of the local bus is tracking with respect to P2L_DFRAME and P2L_DATA.
2. In the figure, the arrows next to signal names indicates direction of signal flow. It is the convention of this document to depict the GN412x on the right side and the FPGA on the left.

Figure 5-3: PCIe to Local Bus Target Write Transaction



The write command shown in Figure 5-3 consists of a 32-bit header (H), followed by a 32-bit address (A), and concluded by a burst of 4 x 32-bit data transfers.

While PCI Express supports a 64-bit address space, only the lower portion of the address is passed through to the local bus. Since the base address registers (BAR0 or BAR2) decode the upper bits of the 64-bit address, only the lower bits are required to form an index within the decoded window. The header then indicates which BAR decoded the address.

The header format for the target write request packet is given in Table 5-4.

The format for the 32-bit address is a byte address, where bits 1:0 are reserved and should be ignored by the FPGA. Depending on the size of BAR0/BAR2, some of the upper address bits will be masked off.

General rules about RDY signals:

- *_RDY may be asserted or de-asserted during any LCLK cycle.
- When the initiator side of the local bus detects RDY asserted for a specific type of cycle, then it may begin a transfer (after the turn around latency). Once an initiator has started a transfer, the state of RDY will not affect the transfer. In other words, the de-assertion of RDY will not affect a transfer already under way. The receiver of the packet must have sufficient space for a maximum sized packet.

General rules about Request signals:

- *_REQ may be asserted without the requester being obligated to perform a transaction. The requester may then de-assert REQ.
- *_REQ may be de-asserted as soon as the associated transaction begins or it can remain asserted throughout the cycle.
- *_REQ may be asserted at any time even if a previous transaction is under way.
- If the corresponding RDY signal has been asserted for at least 2 LCLK cycles and REQ is asserted, then FRAME may be asserted at the same time as REQ. Alternately, FRAME may be delayed by the requester.

5.2.2.2 PCI Express Target Read Transaction

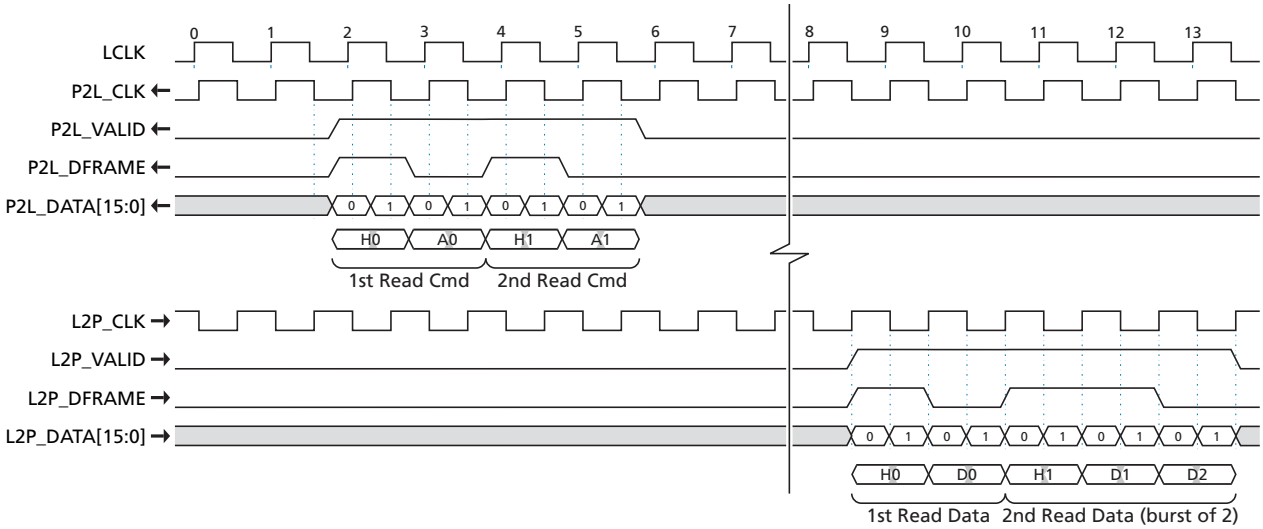
When a PCIe agent wants to read data from a local bus device, this is referred to as a PCIe target read.

Target read transactions begin similar to a target write transaction, except that reads require a completion by sending data back to the requester. Since all GN412x target write transactions are, in effect, posted writes, there is no completion required.

PCIe target read transactions are split into 2 parts: a read request and a read completion.

The transaction begins when the GN412x receives a read request from an external PCIe agent. The transaction must fall within the address range of either BAR0 or BAR2 (the base address registers used for PCI-to-local forwarding).

Figure 5-4: PCIe to Local Bus Target Read Transaction



Referring to Figure 5-4, the PCIe read request consists of a read request header sent via the P2L interface to the attached FPGA. In the case of Figure 5-4, 2 read requests are shown back-to-back. By default, the GN412x will issue only one outstanding read per virtual channel. This is controlled using the P_RD_CREDIT0/P_RD_CREDIT1 control bits in the LB_CTL register. When P_RD_CREDIT0/P_RD_CREDIT1 are set to '1', each corresponding virtual channel will support up to 3 outstanding PCIe-to-local reads.

P2L_FRAME in the LCLK domain is retimed into P2L_DFRAME in the P2L_CLK domain where it is aligned with the header and address phases of the request (H0/A0). A second read request is seen immediately after the first (H1/A1).

After some period of time, the read data is sent back to the GN412x as indicated by L2P_DFRAME. The second L2P_DFRAME cycle shows the returned data for the second request and is extended due to the fact that it is a burst transaction.

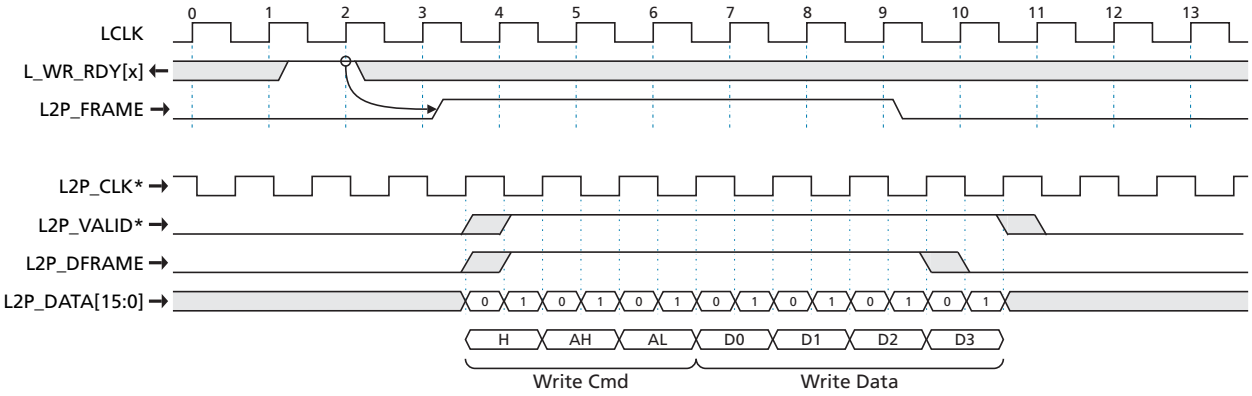
5.2.3 PCI Express Master Transactions

A PCI Express Master transaction is when an FPGA on the local bus generates PCIe transactions as a requester. Master transactions are typically used by a peripheral device to perform high-throughput data transfer.

5.2.3.1 PCI Express Master Write Transaction

From the local bus perspective, a local-to-PCIe master write is similar to a PCIe-to-local target write except that the data flows in the opposite direction. A local bus agent may initiate a local-to-PCIe master write whenever L_WR_RDY for a particular VC is asserted. This is illustrated in Figure 5-5.

Figure 5-5: Local-to-PCIe Master Write Transaction



The transaction begins with a command header (H), followed by a 64-bit address (AH=address high, AL=address low), and concludes with the data (D0-D3). In this case, the data is a burst of 4 x 32-bit words. L2P_DFRAME assertion marks the beginning of the packet. L2P_DFRAME is then de-asserted for the last 32-bit word of data to mark the END of the packet.

5.2.3.2 PCI Express Master Write Exception

Since all write transactions are posted, there is no message coming back to indicate when a write has failed. When a local-to-PCIe write generates an exception condition, this state is captured in the INT_STAT (on page 110) register, and can be used to signal an interrupt.

5.2.3.3 PCI Express Master Read Transaction

Credit-based flow control is used for issuing Local-to-PCIe master reads. The GN412x can accept up to three outstanding read requests (credits) for each VC. Once a read completion has been fulfilled, then there is an additional “credit” available to that particular VC to allow another request to be issued. The FPGA must track how many reads are outstanding, in order to insure that it doesn’t deplete the available credits. In order to simplify the design of the master read controller in FPGA, the designer may chose to issue only one read request at a time. This will degrade throughput as a consequence, but allows circuit simplification.

5.2.3.4 PCI Express Master Read Exception

When local-to-PCIe read generates an exception condition, a completion packet will be returned to the FPGA indicating that an exception was generated. This exception packet will terminate the transaction. The returned packet will consist of a header (P2L_DFRAME HIGH) followed by an idle cycle (P2L_DFRAME LOW). The exception condition is indicated in the STAT field of the header. See Table 5-6 and Table 5-14.

5.2.3.5 L2P_RDY Operation

Outbound packets (from FPGA-to-PCIe) sent to the L2P_DATA inputs are queued into a FIFO, as they are forwarded to the PCI Express transaction controller. The FIFO acts as a clock boundary to allow the local bus to operate at a rate independent of the PCI Express rate. This FIFO may become "full" if either the message signalled interrupt controller or a PCIe read access to internal registers or configuration space is competing for the transaction controller resource.

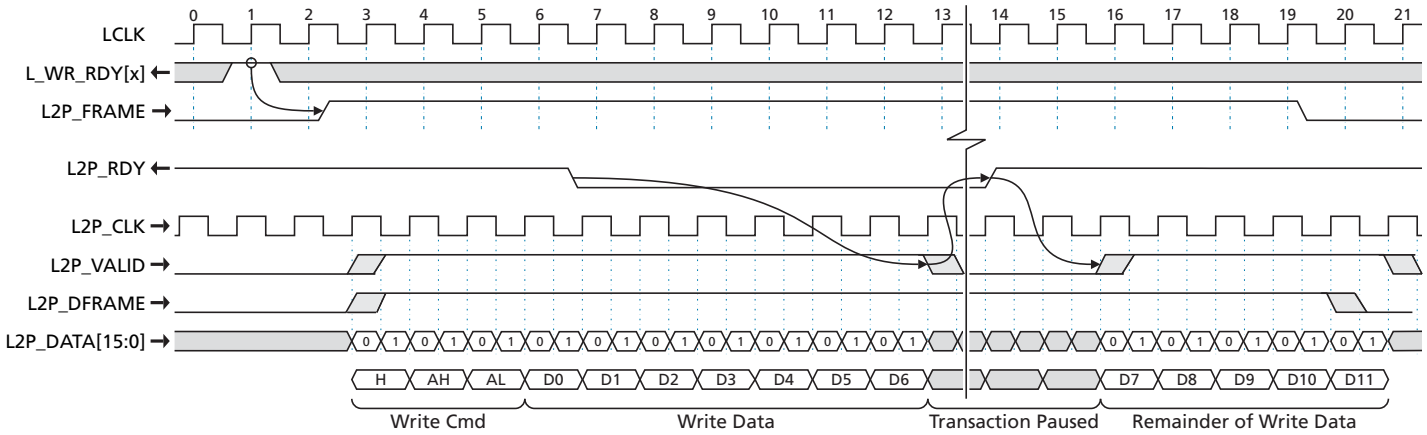
L2P_RDY is used to throttle the FPGA to prevent the L2P FIFO from becoming full under these conditions. L2P_RDY will be de-asserted by the GN412x 7 clock cycles in advance of a possible full condition. L2P_RDY differs from L_WR_RDY[1:0] and P_RD_D_RDY[1:0] in that it needs to be continuously monitored during a transaction. By contrast, the other L_WR_RDY[1:0] and P_RD_D_RDY[1:0] signals are only evaluated at the beginning of a transaction.

Note: L_WR_RDY[1:0] & P_RD_D_RDY[1:0] are available for GN4124 device; while L_WR_RDY & P_RD_D_RDY are available for GN4121 device.

When L2P_RDY is de-asserted, the FPGA controller should de-assert L2P_VALID within 7 LCLK cycles. When L2P_VALID is de-asserted, L2P_DATA and L2P_DFRAME should be "frozen" in the state reflecting the next data transfer. When L2P_RDY is asserted again, L2P_VALID may be asserted to continue the frozen transaction.

An example of L2P_RDY usage is given in Figure 5-6.

Figure 5-6: Operation of the L2P_RDY Signal During an L2P Transaction



In the transaction of Figure 5-6:

- An L2P_FRAME cycle starts normally at clock 2.
- At clock 6, the GN412x de-asserts L2P_RDY.

- L2P_RDY is sampled de-asserted by the FPGA on clock 5, and it must respond within 7 clocks by de-asserting L2P_VALID.
- At clock 14, L2P_RDY is back to the “ready” state, and data transfer can resume.
- During the time that L2P_VALID is de-asserted, the data on the L2P bus is not used by the GN412x.

L2P_RDY will only be de-asserted for brief periods during some MSI or internal register reads and these are relatively infrequent under typical use. Consequently, the impact on performance is minimal.

5.2.3.6 P2L_RDY Operation

The operation of P2L_RDY is symmetrical to the operation of L2P_RDY except in the other direction. The local bus controller implemented in FPGA doesn’t have to implement this function. However, the mechanism is there if required.

5.2.3.7 Traffic Class/Virtual Channel Mapping

The mapping of traffic classes to virtual channels is controlled through the TC-VC mapping register bits TC_VC_MAP in the VC_RESOURCE_CR0 (on page 121) and VC_RESOURCE_CR1 (on page 123) registers. By default, all traffic classes are mapped to VC0.

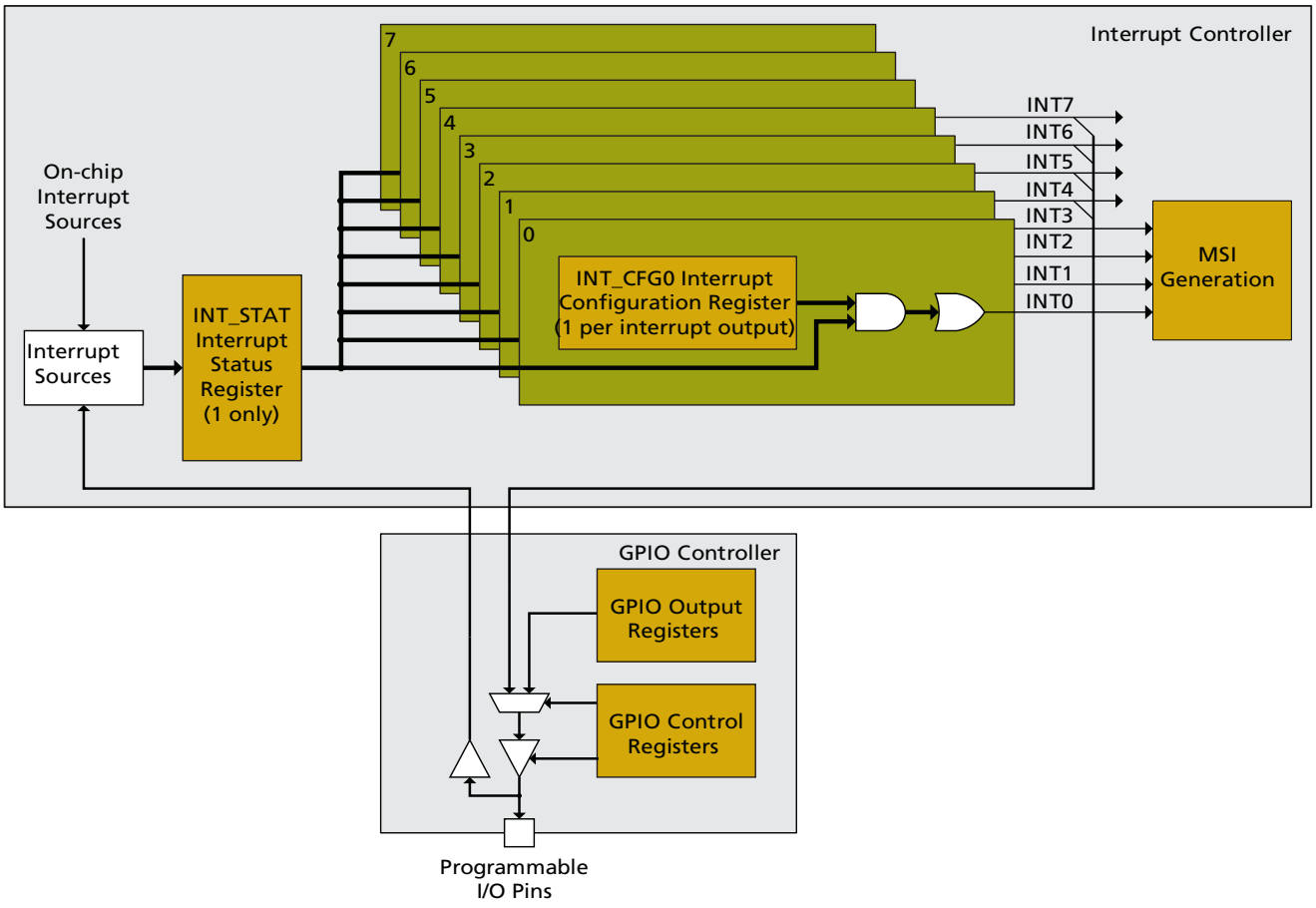
6. Interrupt Control Unit

This chapter discusses the operation of the interrupt controller including PCIe message signalled interrupts (MSI), internally generated interrupts, and local interrupt via the GPIO pins.

6.1 Overview

The interrupt controller is depicted in Figure 6-1. It collects all sources of interrupts, routes them through a shared status register, and then provides a configuration register per output. This allows any interrupt source or sources to be routed to any interrupt output.

Figure 6-1: Interrupt Controller Block Diagram



6.2 Operation

The 32-bit INT_STAT (on page 110) register allows an interrupt service routine (ISR) to see all sources of interrupts by reading a single register. A value of '1' in a specific register bit corresponds to that interrupt source being TRUE or asserted. In other words, it acts as an active LOW mask.

Interrupt sources may be either internal sources such as the 2-wire controller or external pins from the GPIO controller.

Each interrupt output has a corresponding INT_CFG0-7 (on page 111) register to individually enable each interrupt source. When more than one source is enabled for a specific output, the interrupts are "ORed" together, so that if any of the enabled interrupts are asserted, then the interrupt output will be asserted. The polarity of the INT_CFGx bits is active HIGH to enable. That is, a '1' in any bit indicates that the corresponding interrupt source is enabled. A '0' indicates disabled.

The first four interrupts are routed to the MSI controller where they generate PCI Express interrupt packets.

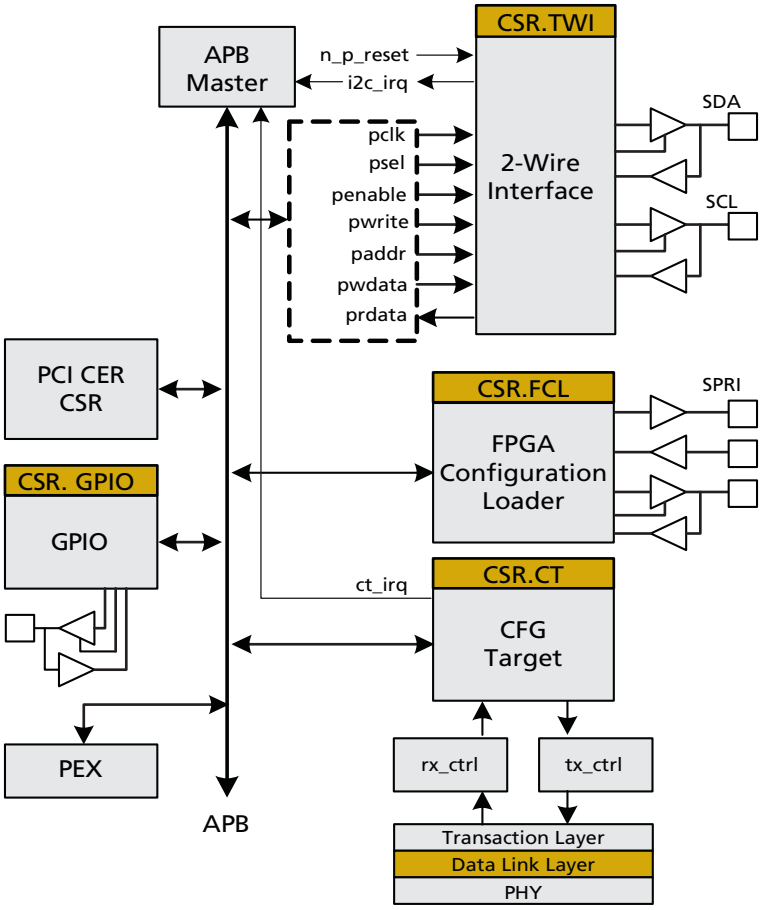
Interrupt outputs 7-4 are routed to the GPIO block where they may be asserted onto GPIO pins according to the settings of the GPIO control registers. Refer to 10.4.9 GPIO Registers.

For a particular interrupt output, software can determine what interrupt source or sources caused the interrupt. This can be done by performing a bitwise AND of the status register value, INT_STAT (on page 110), with the corresponding configuration register value, INT_CFG0-7 (on page 111), for that interrupt output.

7. Peripherals

The GN412x uses Advanced Peripheral Bus (APB) as a peripheral devices backbone.

Figure 7-1: APB and Peripheral Devices Block Diagram.



7.1 2-Wire Interface

The 2-wire interface block in GN412x operates in two different modes. In the auto mode, the block is control by an on-chip state machine, and the 2-wire interface operates only in slave mode. The block automatically responds to read and write requests with the default slave address of 0x05. The entire internal register space of the GN412x can be access in this mode of operation. In the host control mode, the host has complete access to the registers within the 2-wire interface block. The host requires control the operation of the 2-wire interface via the registers to perform any read or write transfers. In this mode of operation, the block can be configure in master or slave mode. Refer to the register LB_CTL (on page 127) on switching the two modes of operation on the 2-wire interface block.

7.1.1 Master Mode and 7.1.2 Slave Mode describe the steps that the host performs, when the 2-wire interface block is configured in host control mode.

The following registers can be configured to determine the functionality of the interface.

Table 7-1: 2-Wire Interface Register Map

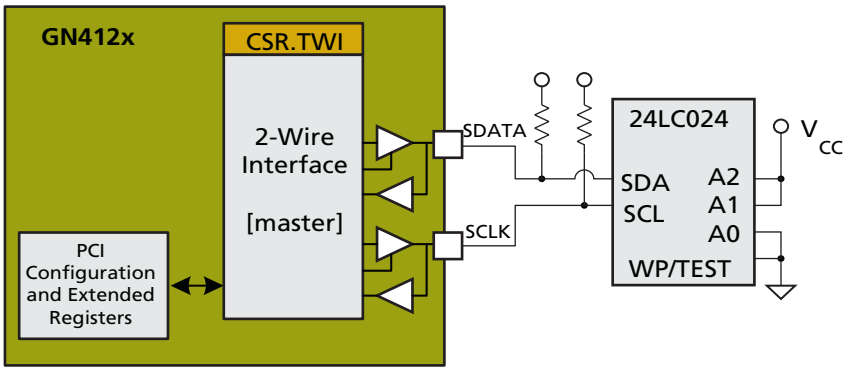
Register Address	Type	Reset Value	Function
0x0900	R/W	0x0000	TWI_CTRL (on page 139): Control Register
0x0904	RO	0x0000	TWI_STATUS (on page 140): Status Register
0x0908	R/W	0x0000	TWI_ADDRESS (on page 140): 2-wire Interface Address Register
0x090C	R/W	0x0000	TWI_DATA (on page 141): 2-wire Interface Data Register
0x0910	RO	0x0000	TWI_IRT_STATUS (on page 141): Interrupt Status Register
0x0914	R/W	0x0000	TWI_TR_SIZE (on page 143): Transfer Size Register
0x0918	R/W	0x0000	TWI_SLV_MON (on page 143): Slave Monitor Pause Register
0x091C	R/W	0x001F	TWI_TO (on page 144): Time Out Register
0x0920	RO	0x02FF	TWI_IR_MASK (on page 144): Interrupt Mask Register
0x0924	WO	0x0000	TWI_IR_EN (on page 144): Interrupt Enable Register
0x0928	WO	0x0000	TWI_IR_DIS (on page 145): Interrupt Disable Register

For details regarding the 2-wire interface register map, please refer to 10.4.8 2-Wire Interface Registers.

7.1.1 Master Mode

Figure 7-2 is a block diagram of the 2-wire interface in master mode. The EEPROM is connected to the GN412x via the TWI interface, and the host can access the EEPROM content using master read and write transfers. To read from the EEPROM the host is required to perform a write transfer followed by a read transfer, as illustrated by Figure 7-5. To write to the EEPROM the host needs to complete the master write transfer, as illustrated by Figure 7-9. Please consult to the target TWI device data sheet on the requirement for 2-wire interface access.

Figure 7-2: 2-Wire Interface in Master Mode



7.1.1.1 Read Transfer

The 2-wire interface transfer is performed according to Figure 7-3 and Figure 7-4.

Figure 7-3: Master Read Transfer (Normal Address Mode)

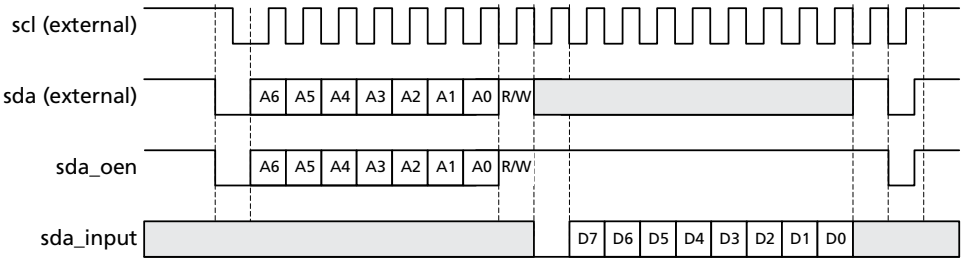
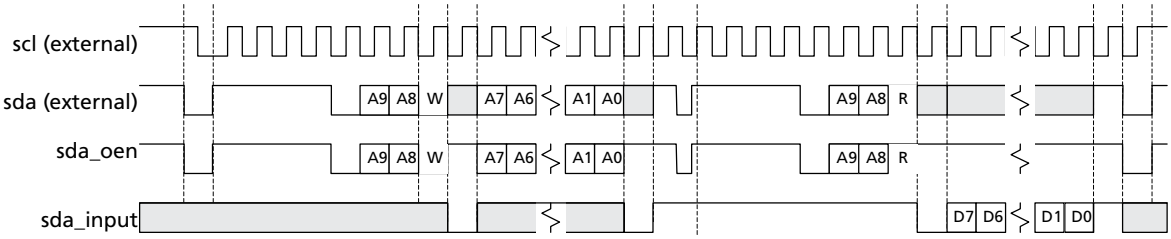


Figure 7-4: Master Read Transfer (Extended Address Mode)



To accomplish an 2-wire interface read transfer, the host must perform the following steps:

1. Write to the Control Register (TWI_CTRL (on page 139)) to set up SCLK speed and addressing mode.
2. Set MS, ACKEN, CLR_FIFO bits and RW bit in the TWI_CTRL (on page 139).
3. If the host wants to hold the bus after the data is received, it must also set the HOLD bit.
4. Write the slave address in TWI_ADDRESS (on page 140). This initiates the 2-wire interface transfer.
5. Write the number of requested bytes in the TWI_TR_SIZE (on page 143).

If TWI_CTRL.HOLD is set, the transfer does not continue until the host reads the 2-wire interface data register (TWI_DATA (on page 141)).

The host is notified of any available received data in two ways:

- If an outstanding transfer size is more than the FIFO size -2, a data interrupt is generated (TWI_IRT_STATUS.DATA bit set) when there are two free locations available in the FIFO.
- If an outstanding transfer size is less than FIFO size -2, a transfer complete interrupt is generated (TWI_IRT_STATUS.COMP bit set) when the outstanding transfer size bytes are received.

In both cases, TWI_STATUS.RXDV is set.

The host does not need to clear TWI_CTRL.ACKEN bit. The 2-wire interface automatically returns NACK after receiving the last expected byte, and terminates the transfer by generating a STOP condition.

If the TWI_CTRL.HOLD bit is set during a master read transfer, the 2-wire interface drives the SCLK line LOW if FIFO is full, and there are more outstanding bytes to be received until the host reads the FIFO through TWI_DATA.

If at any point the slave responds with NACK while the master transmits a slave address for master read transfer, the transfer automatically terminates and a transfer NACK interrupt is generated (TWI_IRT_STATUS.NACK bit set). The outstanding amount of data is read from TWI_TR_SIZE.

If at any point the SCLK line is held LOW by the master or the accessed slave for more than the period specified in the time out register (TWI_TO (on page 144)), a TWI_IRT_STATUS.TO is generated. This interrupt serves only as a notification to the host and does not affect the ongoing transfer.

The 2-wire interface module, as a master, is capable of performing combined transfers as specified by the 2-wire interface specification. It can perform more than one transfer without releasing the 2-wire interface bus, and these transfers are separated by a repeated START condition (Sr).

The host can realize combined transfers by writing to the address register (TWI_ADDRESS (on page 140)) before the end of the previous transfer. The control register must be written to before accessing the address register, so that the correct transfer direction and slave addressing mode is set up for the second part in a combined transfer.

The host must set the TWI_CTRL.HOLD bit in the beginning of the first part. When the host receives transfer complete interrupt, it can then initiate the second part of the combined transfer.

Figure 7-5: Master Byte Read

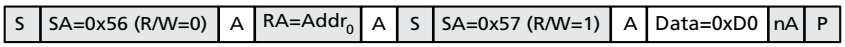
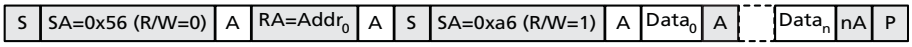


Figure 7-6: Master Page Read



Attention: Legend for Figure 7-5 & Figure 7-6:
S: Start,
SA: Slave Address,
RA: Read Address,
A: Acknowledge,
nA: non Acknowledge,
P: Stop

: Sent out from TWI block

: Sent out from Slave Device

7.1.1.2 Write Transfer

The 2-wire interface transfer is performed according to Figure 7-7 and Figure 7-8.

Figure 7-7: Master Write Transfer (Normal Address Mode)

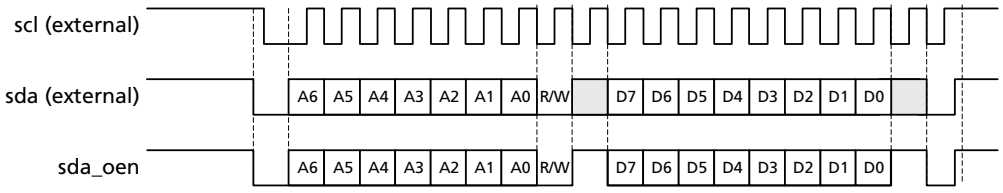
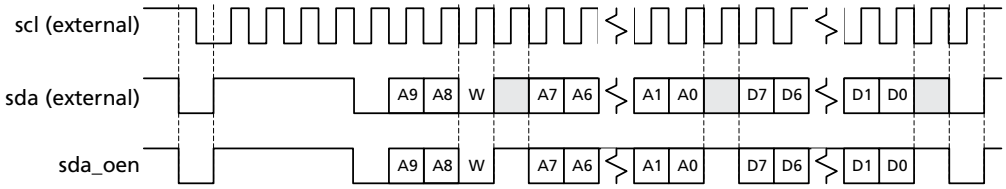


Figure 7-8: Master Write Transfer (Extended Address Mode)



To accomplish an 2-wire interface write transfer, the host must perform the following steps:

1. Write to the Control Register to set up SCLK speed and addressing mode.
2. Set the MS, ENACK and CLR_FIFO bits and clear the RW bit in the TWI_CTRL (on page 139).
3. Supply more data by writing to the TWI_DATA (on page 141) register. This will cause the 2-wire interface to continue with the transfer writing more data to the slave
4. Load the data to be sent to the slave. This is done by writing to TWI_DATA (on page 141). The data is pushed in the FIFO each time the host writes to TWI_DATA.

The 2-wire interface transfer is performed according to the following timing diagrams. If all data provided by the host is transferred successfully, the host is notified by the transfer complete interrupt with the TWI_IRT_STATUS.COMP being set in the interrupt status register. A data interrupt is generated whenever there are only two bytes left in the transmission.

If the TWI_CTRL.HOLD bit is not set when the data is transmitted, the 2-wire interface generates a STOP condition and terminates the transfer. If HOLD bit is set, the 2-wire interface will hold the SCLK line LOW once the data is transmitted. The host is notified of this event by an interrupt as follows.

- a transfer complete interrupt is generated (TWI_IRT_STATUS.COMP bit set).

In both cases, TXDV bit in the TWI_STATUS (on page 140) register is cleared.

At this point, the host can proceed in three ways:

- Clear the HOLD bit. This will cause the 2-wire interface to generate a STOP condition.
- Supply more data by writing to the 2-wire interface register. This will cause the 2-wire interface to continue with the transfer writing more data to the slave.
- Perform combined format transfer. This is accomplished by first writing to the control register and, if necessary, change transfer direction or addressing mode. The host must then write to 2-wire interface address register, which leads to a RESTART condition being generated by the 2-wire interface.

If at any point when the slave responds with a NACK, the transfer automatically terminates, and a transfer NACK interrupt is generated (TWI_IRT_STATUS.NACK bit set).

The outstanding amount of data minus one is read by the transfer size register. Unless it is the very last byte written by the host into the FIFO and was a NACK byte, TWI_STATUS.TXDV will remain HIGH. In such a case, the host must clear the FIFO by setting TWI_CTRL.CLR_FIFO bit.

If at any point the SCLK line is held LOW by the master or the accessed slave for more than the period specified in the time out register (TWI_TO (on page 144)), a TWI_IRT_STATUS.TO interrupt is generated. The outstanding amount of data minus one is read from transfer size register (TWI_TR_SIZE (on page 143)). This interrupt serves only as a notification to the host and does not affect the ongoing transfer.

Figure 7-9: Master Byte Write

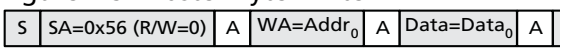
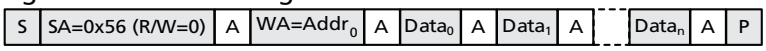


Figure 7-10: Master Page Write



Attention: Legend for Figure 7-9 & Figure 7-10:
S: Start,
SA: Slave Address,
WA: Write Address,
A: Acknowledge,
nA: non Acknowledge,
P: Stop

: Sent out from TWI block

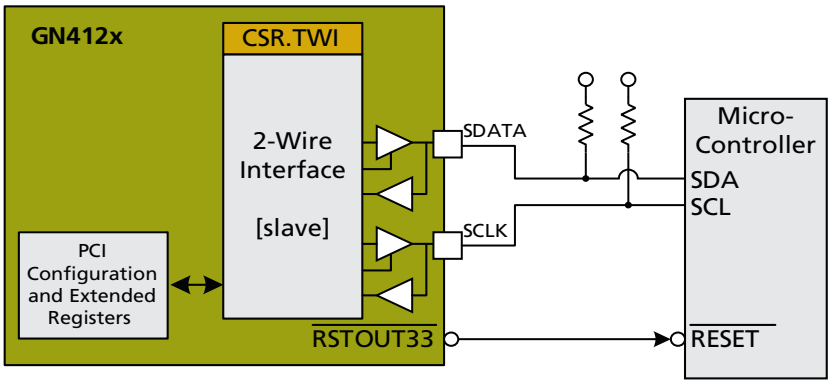
: Sent out from Slave Device

7.1.2 Slave Mode

The 2-wire interface is set up as a slave by clearing the TWI_CTRL.MS bit. The 2-wire interface slave must be given an unique identifying address by writing to the 2-wire interface address register (TWI_ADDRESS (on page 140)). The SCLK speed must also be set up at least as fast as the fastest SCLK frequency expected to be seen.

Figure 7-11 is a block diagram of the 2-wire interface in slave mode. A microprocessor is connected to the GN412x via the TWI interface, and it acts as a TWI bus master device. The host must program the TWI block to respond to read and write request originating from the TWI master device.

Figure 7-11: 2-Wire Interface in Slave Mode



7.1.2.1 Read Transfer

The slave becomes transmitter, after recognizing the entire slave address [GN412x 2-wire interface default device address is set to 0x05] sent by master and the R/W field in the last address byte sent is HIGH. This means that the slave has been requested to send data over the 2-wire interface bus. The host is notified through an interrupt and TWI_IRT_STATUS.DATA is set. At the same time, the SCLK line is held LOW to allow the host to supply data to the 2-wire interface slave before the 2-wire interface master starts sampling the SDATA line.

The host must supply data for transmission through the 2-wire interface data register (TWI_DATA (on page 141)), so that SCLK line is released and transfer continues. If it does not write to the 2-wire interface data register before the time out period expires, an interrupt is generated and a TWI_IRT_STATUS.TO interrupt flag is set. The SCLK line remains LOW as kept by the 2-wire interface. This prevents any activity on the 2-wire interface bus to happen. Once the host writes to the 2-wire interface data register, transfer continues.

The host can continue to load data through the 2-wire interface data register while transfer is in progress. The amount of data loaded in the FIFO might be a system known parameter or communicated in advance through a higher level protocol using the 2-wire interface bus.

An interrupt is generated and TWI_IRT_STATUS.DATA is set as when there are only two valid bytes left in the FIFO for transmission. At this point, the TWI_STATUS.TXDV flag is still set. If the 2-wire interface master returns a NACK on the last byte transmitted, an

interrupt is generated and the TWI_IRT_STATUS.COMP interrupt flag is set as soon as the 2-wire interface master generates a STOP condition. If the master acknowledges the last byte, transfer must continue, TWI_IRT_STATUS.DATA is set and an interrupt is generated.

At this point, the TWI_STATUS.TXDV flag is cleared. If the 2-wire interface master terminates the transfer before all the data is sent by the slave, the host is notified of the event by an interrupt and the TWI_IRT_STATUS.NACK interrupt flag is set. At this point TWI_STATUS.TXDV remains set. Transfer size register (TWI_TR_SIZE (on page 143)) indicates the remaining bytes. The host must set TWI_CTRL.CLR_FIFO bit to clear the FIFO and TWI_STATUS.TXDV bit respectively.

Figure 7-12: Slave Double Word Read

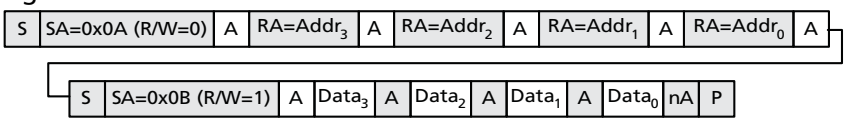
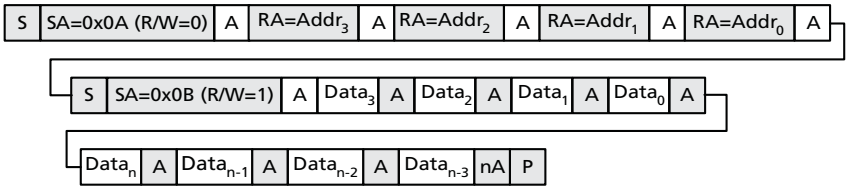


Figure 7-13: Slave Double Word Page Read



Attention: Legend for Figure 7-12 & Figure 7-13:
S: Start,
SA: Slave Address,
RA: Read Address,
A: Acknowledge,
nA: non Acknowledge,
P: Stop

: Sent out from TWI block

: Sent out from Slave Device

7.1.2.2 Write Transfer

The slave becomes receiver, after recognizing the entire slave address sent by the master and the R/W bit in the first address byte is LOW. This means that the master is about to end one or more data bytes to the slave over the 2-wire interface bus.

An interrupt is generated, and the TWI_IRT_STATUS.DATA interrupt flag is set, when there are only two free locations left in the FIFO. Once a byte is acknowledged by the 2-wire interface slave, TWI_STATUS.RXDV bit is set indicating that new data has been received. The host reads the received data through the 2-wire interface data register (TWI_DATA (on page 141)).

Whenever 2-wire interface master generates STOP condition, an interrupt is generated, and the TWI_IRT_STATUS.COMP interrupt flag is set.

The transfer size register contains the number of bytes received that are available in the FIFO. This number is decremented on each read of the 2-wire interface data register by the host.

If the host is not designed to respond quickly enough to interrupts indicating that more data is received, it can set the TWI_CTRL.HOLD bit to avoid overflow conditions.

If the TWI_CTRL.HOLD bit is set, the 2-wire interface keeps the SCLK line LOW until the host clears resources for data reception. This prevents the master from continuing with the transfer and sending any more data, which can otherwise cause an overflow condition in the slave. The host clears resources for data reception by reading 2-wire interface data register.

If TWI_CTRL.HOLD bit is set and the 2-wire interface keeps the SCLK line LOW for longer than the time out period, an interrupt is generated and the TWI_IRT_STATUS.TO interrupt flag is set. This is a notification only and does not change the status of the current transfer.

Figure 7-14: Slave Double Word Write

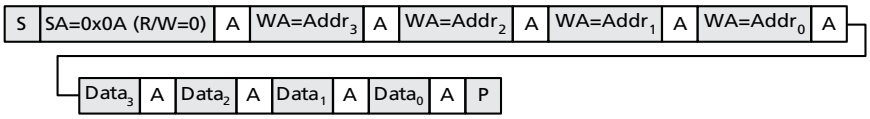
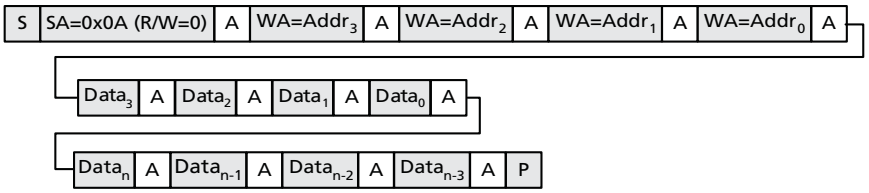


Figure 7-15: Slave Double Word Page Write



Attention: Legend for Figure 7-14 & Figure 7-15:
S: Start,
SA: Slave Address,
WA: Write Address,
A: Acknowledge,
nA: non Acknowledge,
P: Stop

: Sent out from TWI block

: Sent out from Slave Device

7.1.3 2-Wire Interface Additional Information

For the 2-wire protocol specification, please refer to:

- The I²C-bus Specification, Philips Semiconductor, Version 2.1, January 2000

For the 2-wire interface register map, please refer to 10.4.8 2-Wire Interface Registers.

For the GN412x initialization using 2-wire interface, please refer to 3. Initialization.

7.2 General Purpose IO Block

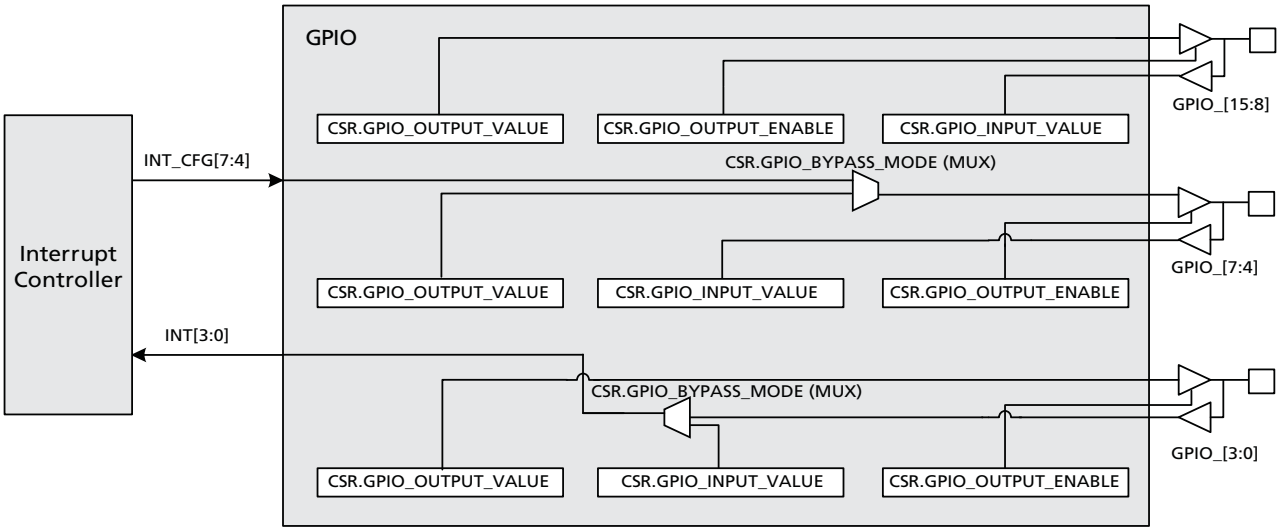
The General Purpose Input Output (GPIO) block provides a way to communicate with the external environment in a flexible manner to support control and / or status signalling between the GN412x and other devices.

Also, the GPIO block can receive interrupt requests either internally generated by the bridge or externally generated in which case the GPIO can pass interrupt request to the interrupt controller.

The GPIO can initialize interrupt request via software, by writing to the CSR.GPIO registers.

The GPIO registers are accessible from either the PCIe link (through BAR4 memory access) or the 2-wire interface in auto mode configuration.

Figure 7-16: GPIO Block Diagram



For the GPIO register map please refer to 10.4.9 GPIO Registers.

Each GPIO pin can be switch into bypass mode via GPIO_BYPASS_MODE register settings. The bypass mode provides alternative functions to the GPIO pins. Table 7-2: describes the alternative function for the GPIO pins. When a GPIO pin is configured in bypass mode the pin is no longer in control by the GPIO block, thus I/O and interrupt generation functionality through the block is disabled.

Table 7-2: GPIO Definition (when bypass mode is enabled via GPIO_BYPASS_MODE register)

GPIO #	Function	Input / Output	Description
0	EXT_IRQ_0	Input	External interrupt 0 via GPIO_0 pin.
1	EXT_IRQ_1	Input	External interrupt 1 via GPIO_1 pin.
2	EXT_IRQ_2	Input	External interrupt 2 via GPIO_2 pin.
3	EXT_IRQ_3	Input	External interrupt 3 via GPIO_3 pin.
4	INT4	Output	INT4 output signal from the interrupt control unit.
5	INT5	Output	INT5 output signal from the interrupt control unit.
6	INT6	Output	INT6 output signal from the interrupt control unit.
7	INT7	Output	INT7 output signal from the interrupt control unit.
8	PM_LP	Output	Signal HIGH when device is in non-D0 power state or a PME_TURN_OFF message is received from the host.
9	Reserved	Input	Input is ignored.
10	Reserved	Input	Input is ignored.
11	Reserved	Input	Input is ignored.

GPIO #	Function	Input / Output	Description
12	Reserved	Input	Input is ignored.
13	Reserved	Input	Input is ignored.
14	Reserved	Input	Input is ignored.
15	Reserved	Input	Input is ignored.

8. FPGA Configuration Loader

8.1 Overview

The FPGA Configuration Loader (FCL) allows FPGA configuration bitstream download via a Serial Programming Interface (SPRI). A bitstream file contains all binary bit information that is required for FPGA configuration. It is generated using FPGA vendor specific tool, since the information contained in the bitstream is proprietary. Since the GN412x has a configuration space on-chip, it can boot-up and be enumerated in a system without the attached FPGA being programmed.

That capability, and the FCL capability eliminate the need for configuration flash memory to exist on the add-in card. Instead, the host device driver can configure the FPGA by reading a bitstream file and downloading it into the FPGA via the GN412x. FPGA firmware upgrades can be accomplished by simply supplying a new bitstream file as part of the driver release.

8.2 Operation

The FCL system registers provide direct control over the state of the SPRI. A clock divider register, FCL_CLK_DIV (on page 162), controls the clock rate of the output data bits and is set to match the capability of the attached FPGA. The FPGA programming bitstream is loaded through BAR4 memory space. Depending on the mode of operation the bitstream data can be loaded into the FCL data FIFO or directly driving the SPRI output pins.

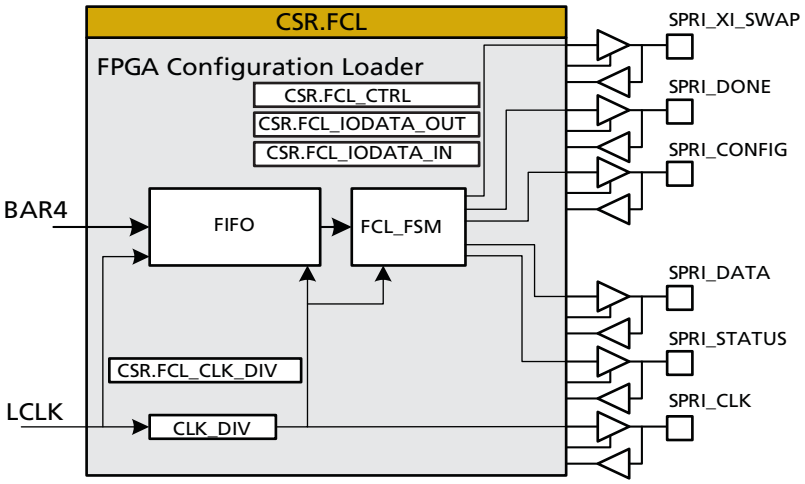
The SPRI operates in three modes, one GPIO and two FCL programming:

- **GPIO Mode**: When SPRI and the FCL FSM are disabled, IO outputs are controlled by the FCL_IODATA_OUT (on page 160) register, IO inputs are registered by FCL_IODATA_IN (on page 159) register, and IO output enables are driven by FCL_EN (on page 160) register. Host can perform bit-bang programming operation through the SPRI.
- **FCL Programming Mode**: The FCL is used to configure an attached FPGA. Host controls the SPRI interface signals via the FCL_IODATA_OUT register to perform FCL configuration setup procedure. Once the FPGA is setup for configuration, the host loads the bitstream data into the FCL FIFO. The FIFO pops 32 bits word each time and serialize to the SPRI output pins.
- **FCL FSM Programming Mode**: An internal finite state machine (FSM) is used to carry out the complete configuration process by controlling the FCL. Host loads the bitstream data into the FCL FIFO and the FCL FSM reads data out of FIFO and drives selected IOs synchronously to divided LCLK. The FIFO pops 32 bits word each time and serialize to the SPRI output pins.

The FCL_FSM reads data out of FIFO and drives selected IOs synchronously to divided LCLK.

After power up, the FCL supports Xilinx Spartan3A FPGA as default. The FCL configuration could be altered using device configuration stored in a external EEPROM device or through direct software access to the FCL registers using BAR4.

Figure 8-1: FPGA Configuration Loader Block Diagram.



To support both Altera and Xilinx devices and to support future changes to the configuration protocol, FPGA configuration is initiated via SPRI GPIO pins. The control signals shown in Table 8-1 are required to program Altera Cyclone and Xilinx Spartan-3 devices.

A brief description of the FCL interface is depicted in Table 8-1.

Table 8-1: FCL Interface

I/O	Xilinx	Altera	FCL	Function
Output	CCLK	DCLK	SPRI_CLOCK	Configuration Clock
Output	DIN	DATA0	SPRI_DATA	Configuration Data
Output	PROG_B	nCONFIG	SPRI_CONFIG	Begin Configuration Process
Input	DONE	CONF_DONE	SPRI_DONE	Configuration Complete
Output	HSWAP	-	SPRI_XI_SWAP	User I/O Pull up Control
Input	INIT_B	nSTATUS	SPRI_STATUS	Xilinx: Initialization Indicator Altera: Error Indicator

8.3 Xilinx FPGA Configuration

The FCL and FCL FSM programming modes support Xilinx slave serial mode configuration. The implementation of the FCL FIFO allows the configuration bit stream to be paused, while host refills the FIFO with additional data. The software will be responsible for selecting a suitable configuration clock speed for the chosen FPGA to meet the specific requirement. If an FIFO underflow occurs, the configuration clock will be stopped, and an underflow interrupt will be generated to request the FIFO to be filled with more data.

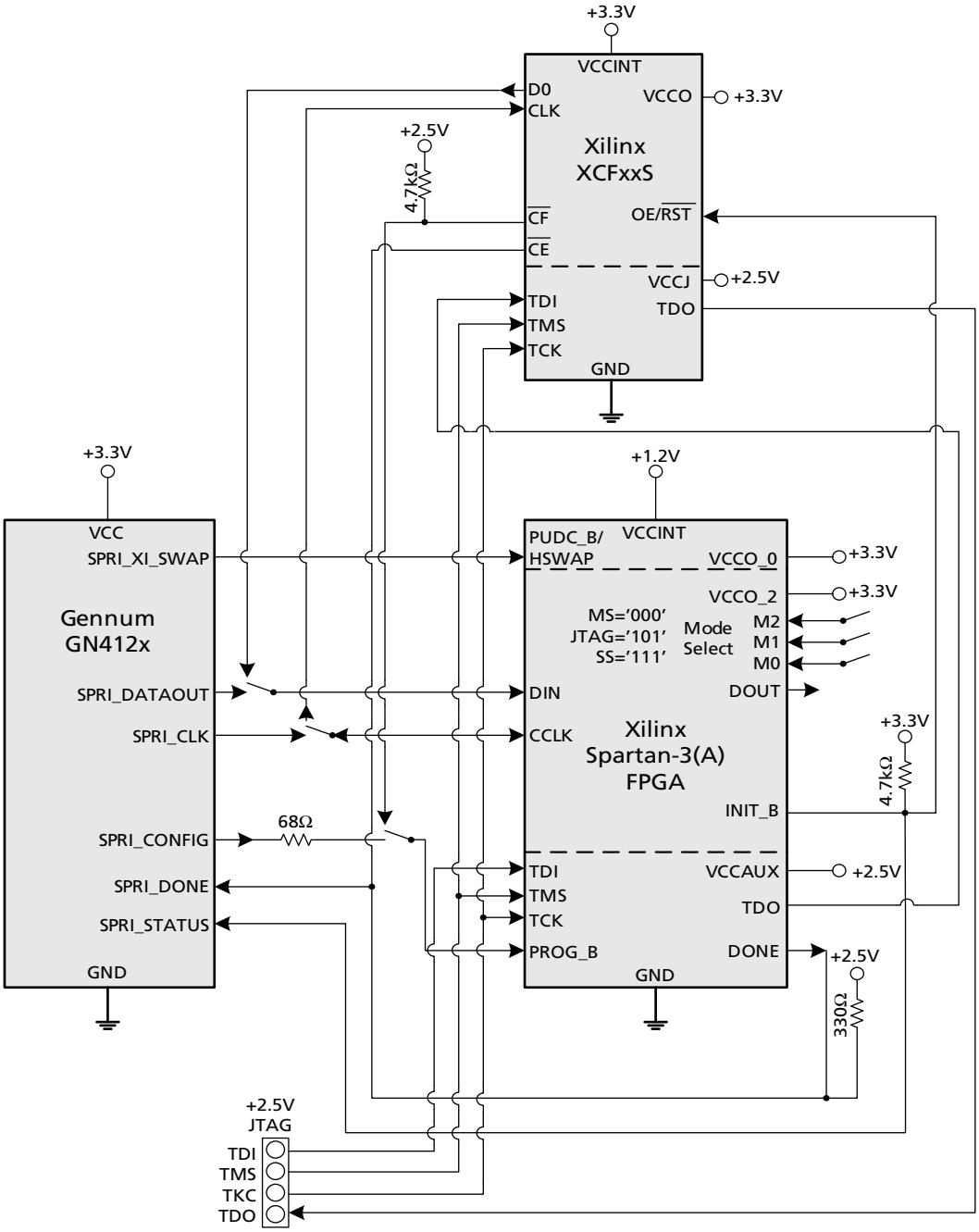
In slave serial mode, set when M[2:0] = “111“ on the FPGA pins, the FCL of GN412x writes the configuration data into the FPGA, using the synchronous serial interface shown in Figure 8-2. The serial configuration data is presented on the FPGA’s DIN input

pin with sufficient setup time before each rising edge of the externally generated CCLK clock input.

The FCL FSM starts the configuration process by pulsing / PROG_B and monitoring that the INIT_B pin goes HIGH, indicating that the FPGA is ready to receive its first data. The host then continues supplying data and clock signals until either the DONE pin goes HIGH, indicating a successful configuration, or until the INIT_B pin goes LOW, indicating a configuration error. The width of the PROG_B assertion is programmable via the FCL_TIMER registers. The host can also program a delay after the successful rising edge detection of the INT_B signal before the FCL generates clock and data through the SPRI, via FCL_TIMER2 registers.

Figure 8-2 depicts the configuration interface between GN412x and Xilinx Spartan-3 FPGA.

Figure 8-2: Serial Programming Interface: Xilinx Spartan-3(A) and Platform Flash example.



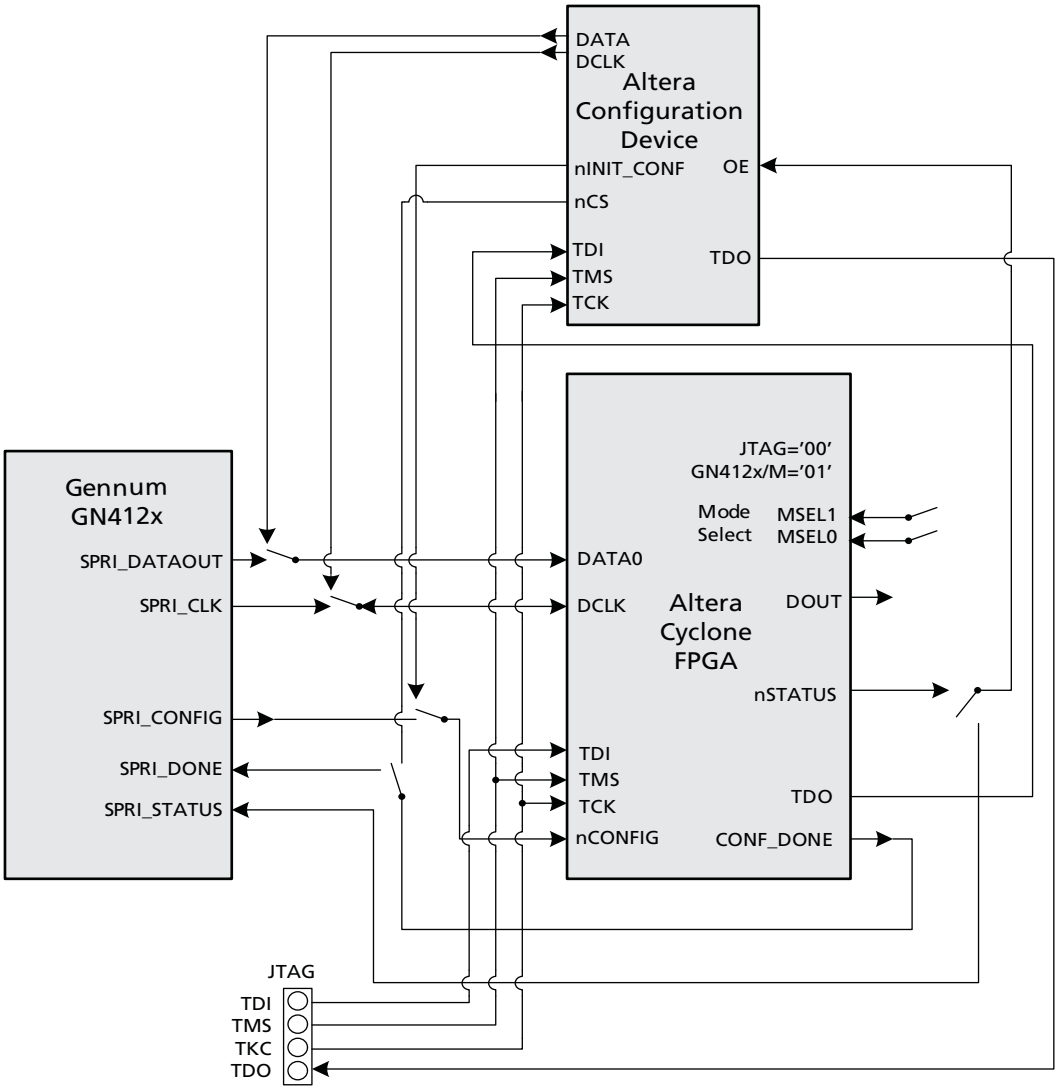
8.4 Altera FPGA Configuration

Configuring Altera FPGAs from an external source is termed passive serial mode.

Passive serial mode can refer to various methods of configuration, such as via a download cable or via a serial configuration device. The GN412x will configure the Cyclone FPGA based on the 'PS Configuration from a Microprocessor' where the GN412x is in effect the microprocessor. This method requires control and monitoring of fewer signals than the other methods. To initiate configuration, nCONFIG is pulsed LOW to HIGH and the target device must then release STATUS. The FCL then places the required configuration data on the DATA0 pin. The data is clocked continuously into the target device using DCLK, until the CONF_DONE pin goes HIGH. The nSTATUS pin is driven LOW during configuration, if there is a configuration error. In this instance, the configuration process can be restarted by pulsing nCONFIG.

Figure 8-3 depicts the configuration interface between GN412x and Altera Cyclone FPGA. In this specific configuration the FCL and FCL FSM also supports the Altera Cyclone FPGA. The sequence of the SPRI interface signals controlled by the FCL FSM is identical to the Xilinx configuration described in 8.3 Xilinx FPGA Configurations, which is similar to the Altera Cyclone configuration.

Figure 8-3: Serial Programming Interface: Altera Cyclone example.



8.5 FCL Programming

There exist some timing restrictions on how the FPGA configuration setup signals are applied and sampled. To help meet these setup timing restrictions, there is a 32-bit programmable timer (FCL_TIME* registers) that can be used when SPRI does not operate in FCL FSM mode (FCL_CTRL.FSM_EN is clear). An interrupt is provided to indicate when the programmed timer value has been reached (FCL_IRQ.TIMER).

8.5.1 FCL Register Map

Table 8-2: FCL Register Map

Base Address: 0xB00			
Address Offset	**R/W**	**Reset Value**	**Function**
0x000	R/W	0x0000	FCL_CTRL (on page 158): Configuration Control Register
0x004	R/W	0x0000	FCL_STATUS (on page 159): Status and Setup Register
0x008	RO	0x0000	FCL_IODATA_IN (on page 159): GPIO Input Register
0x0C	R/W	0x0000	FCL_IODATA_OUT (on page 160): GPIO Output Register
0x10	R/W	0x0000	FCL_EN (on page 160): Output Enable Register
0x14	R/W	0x0000	FCL_TIMER_0 (on page 161): Program Pulse Timer 0
0x18	R/W	0x0000	FCL_TIMER_1 (on page 161): Program Pulse Timer 0
0x1C	R/W	0x0000	FCL_CLK_DIV (on page 162): Clock Divider Value
0x20	RO	0x0000	FCL_IRQ (on page 162): Interrupt Request Register
0x24	WO	0x0000	FCL_TIMER_CTRL (on page 163): Timer Control Register
0x28	WO	0x0000	FCL_IM (on page 163): Interrupt Mask Register
0x2C	R/W	0x0000	FCL_TIMER2_0 (on page 164): Program Pulse Timer 1
0x30	R/W	0x0000	FCL_TIMER2_1 (on page 164): Program Pulse Timer 1

Please refer to 10.4.10 FPGA Configuration Loader Registers for more details regarding FCL registers.

8.5.2 GPIO FPGA Configuration

In GPIO mode, there are 8-bit input, output and enable registers, which can be programmed appropriately to control the SPRI signals for general purpose requirements. The host can carry out the FPGA configuration procedure by accessing these registers to update the SPRI signals manually.

Example 8-1: Configuring FPGA in GPIO mode

1. Enable interrupts by setting FCL_IM (on page 163) appropriately.
2. Set output enables to the appropriate configuration by setting FCL_IODATA_OUT[5:0] =”010111.”
3. Set FCL_CTRL[2:0] = “000” to enable GPIO mode of operation.
4. Follow the configuration sequence specified by the FPGA vendor. Write to FCL_IODATA_OUT register to toggle the SPRI_CLK, SPRI_DATAOUT, SPRI_CONFIG, and SPRI_XI_SWAP output pins. Read from FCL_IODATA_IN register to sample the SPRI_DONE and SPRI_STATUS input pins.

8.5.3 FCL FPGA Configuration

In FCL mode, the SPRI_CLK and SPRI_DATAOUT is controlled by hardware. Bit stream data stored in the FCL FIFO will be flushed automatically when FCL_CTRL.SEND_CFG_DATA is asserted. The host software is responsible to prepare the FPGA into the correct state for accepting bit stream data.

Example 8-2: Configuring FPGA in FCL mode

1. Enable interrupts by setting FCL_IM (on page 163) appropriately
2. Set output enables to the appropriate configuration by setting FCL_IODATA_OUT[5:0] =”010111.”
3. Set FCL clock divider by writing to the FCL_CLK_DIV register.
4. Configure FCL_CTRL.LAST_BYTE_CNT according to the total byte length of the bitstream data. For revision 2 silicon, also set FCL_CTRL[8] to “1” to enable SPRI_CLK stopping.
5. Set FCL_CTRL[2:0] = “010” to enable the configuration process by FCL.
6. Pulse FCL_STATUS.SPRI_CONFIG LOW to initiate configuration process — by writing FCL_IODATA_OUT[2] HIGH, LOW, then HIGH.
7. The FPGA must respond by setting FCL_STATUS.SPRI_STATUS HIGH to indicate the FPGA is ready to accept the configuration data and clock. Wait for FCL_IRQ[0] indicating rising edge detection of SPRI_STATUS.
8. Pre fill the FCL FIFO with bit stream data by writing into FCL_FIFO_DATA register.

9. Once FCL_STATUS.SPRI_STATUS is “1”, set FCL_CTRL.SEND_CFG_DATA HIGH to send the configuration data and clock.
10. Continue to send bit stream data to the FIFO until all data are sent. For revision 2 silicon, set FCL_CTRL.DATA_PUSH_COMP to “1” after all data is written to the FIFO, to indicate completion.
11. Read FCL_STATUS.SPRI_STATUS to determine if the configuration is successful.

8.5.4 FCL FSM FPGA Configuration

In FCL FSM mode, all SPRI signals are controlled by the hardware finite state machine to carry out the FPGA configuration. Bit stream data stored in the FCL FIFO, and it will be flushed automatically once the FSM is started.

Example 8-3: Configuring FGPA in FCL FSM mode

1. Enable interrupts by setting FCL_IM (on page 163) appropriately
2. Set output enables to the appropriate configuration by setting FCL_IODATA_OUT[5:0] =”010111.”
3. Disable timer feature by writing 0 to the FCL_TIMER_CTRL register.
4. Set FCL clock divider by writing to the FCL_CLK_DIV register.
5. Write to FCL_TIMER registers to setup the SPRI_CONFIG pulse width (in numbers of divided FCL output clock).
6. Write to FCL_TIMER2 registers to setup the delay before clock and bitstream data are sent out (in numbers of divided FCL output clock).
7. Profile the FCL FIFO with bit stream data by writing into FCL_FIFO_DATA register.
8. Configure FCL_CTRL.LAST_BYTE_CNT according to the total byte length of the bitstream data. For revision 2 silicon, also set FCL_CTRL[8] to “1” to enable SPRI_CLK stopping.
9. Set FCL_CTRL[2:0] = “111” to start the configuration process by FCL FSM. For revision 2 silicon, set FCL_CTRL.DATA_PUSH_COMP to “1” after all data is written to the FIFO, to indicate completion.
10. Continue to send bit stream data to the FIFO until all data are sent.
11. Once all the data has been transferred, the FCL will wait until it receives the FCL_STATUS.SPRI_DONE from the FPGA, before asserting FCL_IRQ.CONFIG_DONE to indicate successful configuration. If an error occurs during the configuration process the FCL_IRQ.CONFIG_ERROR will be asserted.

8.6 Additional information

Additional information can be found:

Configuration Handbook, Altera, 1.4, October 2007.

Spartan-3 Generation Configuration User Guide, Xilinx, UG332 (v1.2) May 23, 2007.

The 3.3V Configuration of Spartan-3 FPGAs, XAPP453 (v1.1) April 3, 2006.

9. Device Application

The following high-frequency design rules should be considered to achieve optimum performance of the GN412x:

- Use carefully designed controlled-impedance transmission lines with minimal local discontinuities for all high-speed data signals. In particular, the PCI Express transmit/receive and reference clock signals and the SSTL local bus IO.
- Place decoupling capacitors as close as possible to the Power balls.
- Recommended to have low/medium/high capacitance values for bypassing critical supplies with multiple stitching of vias to ground. Capacitors should be placed with smallest values closest to the chip and largest value away from chip.
- High-speed signal routing layers should be separated by reference planes that are unbroken over the region of high-speed signalling. This is to minimize cross talk between adjacent layers.

A trade-off must be made between signal integrity and the cost of the design. The Reference Design Kit support files (available through www.gennum.com/mygennum) provide an example of an acceptable and proven design and PCB layout.

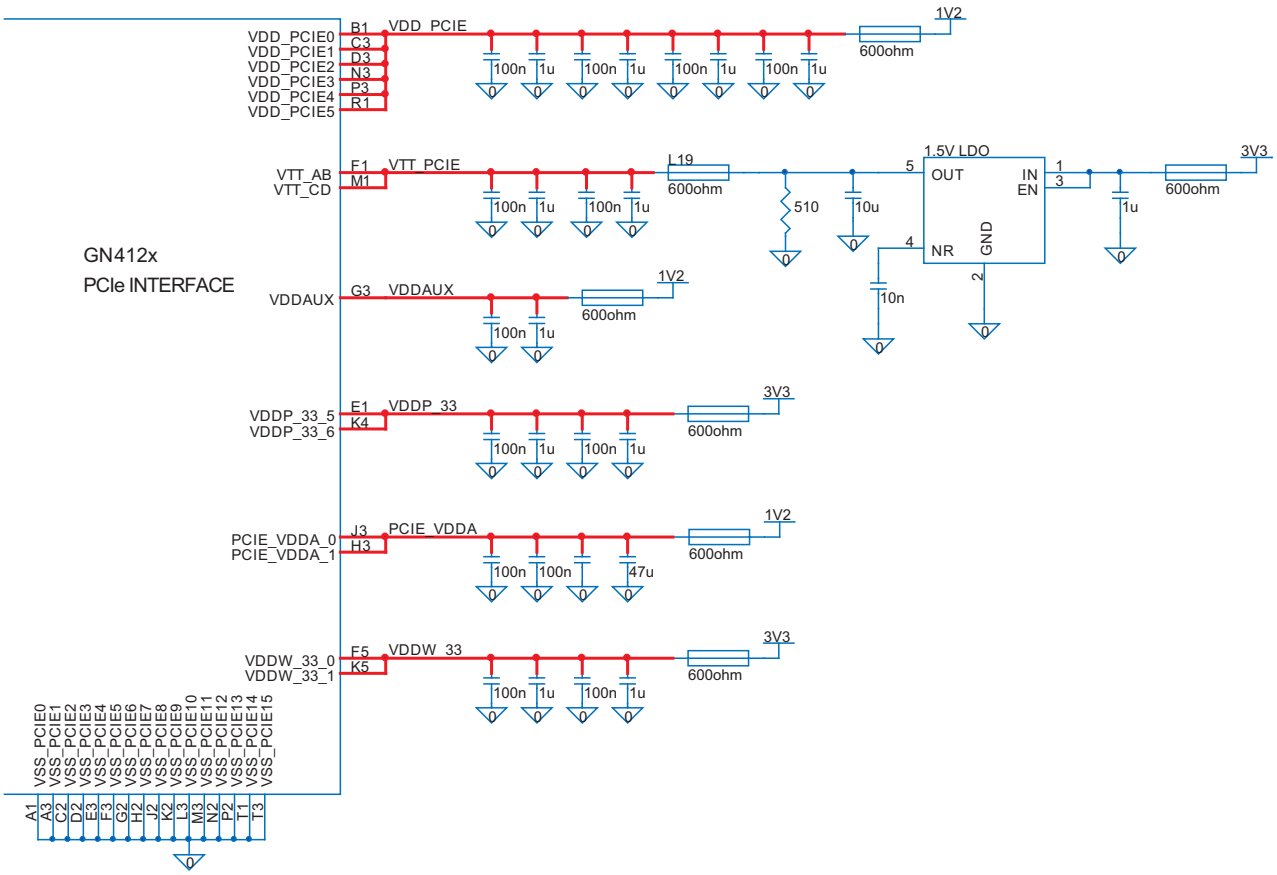
9.1 PCI Express Reference Clock Routing

The PCI Express reference clock inputs PECLKINp/PECLKINn should be AC coupled to the clock driver (either a clock driver chip or from the card edge connector in the case of an endpoint adapter application). PECLKINp/PECLKINn should be routed using controlled impedance techniques.

9.2 PCI Express Power Supply Distribution

The PCI Express PHY on the GN412x has some special power supply requirements. These are depicted in Figure 9-1. The critical power connections on the GN412x should be routed using localized power planes that provide a low impedance path to the bypass capacitors (shown in *red* in Figure 9-1).

Figure 9-1: PCI Express Related Power Supply Design



10. Internal Registers

10.1 Internal Register Access Modes

The internal registers of the GN412x are located within a 4 kilobyte region. There are four ways the register space can be accessed:

- Using the PCI Express configuration space mechanism
- Using PCI Express BAR4 access as memory read/write
- From an EEPROM using the 2-wire interface where the GN412x acts as a 2-wire master device
- 2-wire interface when GN412x TWI block is configured in auto mode, acting as a 2-wire slave device

10.1.1 PCI Express Configuration Access

The first 256 bytes of the register space are accessible from the PCI-compatible configuration space, and type 0 of the PCI configuration header is implemented. The first 4KBytes of register space is accessible using the PCI Express extended configuration space mechanism.

Configuration space access is the method of access used by a host system BIOS or operating system for enumeration. In this mode of access, some register bits will behave differently from other modes of access. This is to accommodate the needs of enumeration. For example, registers like the subsystem vendor ID (PCI_VENDOR on page 89) are read only when accessed through PCI configuration space. However, this same register is read / write in other modes of access.

10.1.2 PCI Express BAR4 Access

The entire 4KBytes space containing all the internal configuration registers of the GN412x is accessible through BAR4 memory space. Once the BIOS enumeration process has completed, a base address is assigned and the software can access all internal registers with read and write capability. This is the recommended method for software to access the register space of the GN412x.

10.1.3 2-Wire Access

The internal register space is also accessible through the 2-wire interface in either master or slave mode.

10.1.3.1 2-Wire Boot Master Mode

In 2-wire boot master mode, the GN412x acts as a 2-wire master to read data from an attached EEPROM. 2-wire boot master mode is activated upon device initialization when a valid 2-wire EEPROM device is attached to the 2-wire pins of the GN412x and the EEPROM_EN signal is strapped HIGH. Just after the de-assertion of the $\overline{\text{RSTIN}}$ pin,

the contents of an attached 2-wire EEPROM will be loaded and used to write internal registers. See 3. Initialization for more details.

2-wire boot master mode can be used to pre-load and overwrite default values of key configuration registers prior to system enumeration. This allows application specific defaults to be established prior to BIOS/OS activity so that the endpoint device can be uniquely identified.

2-wire boot master mode should be differentiated from the general purpose 2-wire master mode, when TWI is configured in host control mode, that is used to access external 2-wire interface slave device. For more information on the 2-wire boot master mode, refer to 3.1.3 Initialization from a 2-Wire EEPROM.

10.1.3.2 2-Wire Slave Mode

The 2-wire interface block accepts read and write commands from an external master when it is configured in auto mode. This is useful for attachment to the FPGA or a processor to allow local control of the GN412x internal registers.

For more information on the 2-wire slave mode of register access, refer to 7.1.2 Slave Mode.

10.2 Register Conventions

This section describes the location and attributes of the GN412x registers. Programming details for each register are given in the appropriate previous sections.

10.2.1 Register Bit Types

Internal register access happens differently for BAR4 access compared to configuration access depending on the state of the CFG_RETRY bits in the PCI_SYS_CFG_SYSTEM (on page 125) register. The CFG_RETRY bits will be de-asserted (‘00’) during the plug-and-play enumeration process. This changes the behaviour of all register bits designated as “FR”.

Each bit in the GN412x registers is readable and writable according to the designations listed in the “Type” column of the register description tables.

The PCI Configuration Registers provide the required type 0 PCI configuration header in compliance with the PCI specification.

Please note that the register offset 0x800 and above cannot be accessed by the configuration space. Registers above 0x800 can only be accessed through the BAR4 memory access method. It is also strongly recommended not to perform any write access to the PCI Configuration Registers (offset below 0x800) through BAR4 after the PCI enumeration process, as any changes in that area can lead to catastrophic system failure.

Table 10-1: Description of Register Bit Types

Type	Description	Config Space Access	BAR4 and 2-wire Access
R	Read Only: internally driven, cannot be modified by writing to them.	Read Only	Read Only
RW	Read/Write: initialized to their default value as indicated in the "Reset" column of the register description tables.	Read/Write	Read/Write
FR	Firmware Initialized, configuration Read Only: These bits are read only when accessed through configuration space and PCI_SYS_CFG_SYSTEM.CFG_RETRY = '00'. Otherwise, they are read/write. They may be automatically loaded during boot initialization when a valid 2-wire EEPROM is attached to the 2-wire master pins and EEPROM mode is enabled.	See description	Read/Write

10.2.2 Numbering Conventions

In the following register descriptions, hexadecimal numbers will use a 0x prefix. Multiple bits represented as binary will be shown in double quotes as in "1010" while single quotes will be used to denote single binary bits (either '1', or '0'). Unless otherwise noted, little endian conventions will be used.

10.2.3 Reserved Register Bits

Register bits that are shown as reserved, should always be written to '0'. For future compatibility, software that uses the results of a register read should mask off all reserved bits. Also, when performing read-modify-write cycles, software should mask off (zero out) all reserved bits, before the write operation.

10.3 Register Map

Table 10-2 shows the register mappings.

Table 10-2: Register Map

Bit Offset				Base Offset
31:24	23:16	15:8	7:0	
PCI_DEVICE (on page 89)		PCI_VENDOR (on page 89)		0x000
PCI_STAT (on page 91)		PCI_CMD (on page 89)		0x004
PCI_CLASS_CODE (on page 92)			PCI_REVISION (on page 92)	0x008
PCI_BIST (on page 93)	PCI_HEADER (on page 93)	PCI_LATENCY (on page 93)	PCI_CACHE (on page 92)	0x00C
PCI_BAR0_LOW (on page 94)				0x010
PCI_BAR0_HIGH (on page 95)				0x014

Bit Offset				Base Offset
31:24	23:16	15:8	7:0	
PCI_BAR2_LOW (on page 96)				0x018
PCI_BAR2_HIGH (on page 97)				0x01C
PCI_BAR4_LOW (on page 97)				0x020
PCI_BAR4_HIGH (on page 98)				0x024
Reserved			PCI_CIS (on page 98)	0x028
Reserved	PCI_SUB_SYS (on page 99)	Reserved	PCI_SUB_VENDOR (on page 98)	0x02C
Reserved		PCI_ROM_BASE (on page 99)		0x030
Reserved			PM_CAP_POINTER (on page 100)	0x034
Reserved				0x038
PCI_INT_PIN (on page 100)	PCI_MIN_GNT (on page 101)	PCI_INT_LINE (on page 100)	PCI_MAX_LAT (on page 101)	0x03C
PM_CAP (on page 103)		PM_NEXT_ID (on page 102)	PM_CAP_ID (on page 102)	0x040
PM_DATA (on page 105)	PM_CSR_BSE (on page 105)	PM_CSR (on page 104)		0x044
MSI_CONTROL (on page 106)		MSI_NEXT_ID (on page 106)	MSI_CAP_ID (on page 106)	0x048
MSI_ADDRESS_LOW (on page 107)				0x04C
MSI_ADDRESS_HIGH (on page 107)				0x050
Reserved		MSI_DATA (on page 108)		0x054
PCIE_CAPABILITY (on page 109)		PCIE_NEXT_ID (on page 109)	PCIE_CAP_ID (on page 109)	0x058
PCIE_DEVICE_CAP (on page 110)				0x05C
PCIE_DSR (on page 112)		PCIE_DCR (on page 111)		0x060
PCIE_LINK_CAP (on page 113)				0x064
PCIE_LSR (on page 115)		PCIE_LCR (on page 114)		0x068
Reserved				0x06C to 0x0FC
DSN_CAP (on page 116)				0x100
DSN_LOW (on page 116)				0x104
DSN_HIGH (on page 117)				0x108

Bit Offset				Base Offset
31:24	23:16	15:8	7:0	
Reserved				0x10C to 0x3FC
VC_CAP (on page 118)				0x400
VC_PORT_CAP_1 (on page 118)				0x404
VC_PORT_CAP_2 (on page 119)				0x408
VC_PSR (on page 120)		VC_PCR (on page 119)		0x40C
VC_RESOURCE_CAP0 (on page 120)				0x410
VC_RESOURCE_CR0 (on page 121)				0x414
VC_RESOURCE_SR0 (on page 122)		Reserved		0x418
VC_RESOURCE_CAP1 (on page 122)				0x41C
VC_RESOURCE_CR1 (on page 123)				0X420
VC_RESOURCE_SR1 (on page 124)		Reserved		0x424
Reserved				0x428 to 0x7FC
PCI_SYS_CFG_SYSTEM (on page 125)				0x800
LB_CTL (on page 127)				0x804
CLK_CSR (on page 128)				0x808
Reserved				0x80C
INT_CTRL (on page 130)				0x810
INT_STAT (on page 131)				0x814
PEX_ERROR_STAT (on page 132)				0x818
INT_CFG0-7 (on page 133)				0x820 to 0x83C
PCI_TO_ACK_TIME (on page 134)				0x840
PEX_CDN_CFG1 (on page 134)				0x844
PEX_CDN_CFG2 (on page 135)				0x848
PHY_TEST_CONTROL (on page 135)				0x84C
PHY_CONTROL (on page 136)				0x850
CDN_LOCK (on page 138)				0x854

Bit Offset				Base Offset
31:24	23:16	15:8	7:0	
Reserved				0x858 to 0x8CC
TWI_CTRL (on page 139)				0x900
Reserved		TWI_STATUS (on page 140)		0x904
Reserved		TWI_ADDRESS (on page 140)		0x908
Reserved		TWI_DATA (on page 141)		0x90C
Reserved		TWI_IRT_STATUS (on page 141)		0x910
Reserved			TWI_TR_SIZE (on page 143)	0x914
Reserved			TWI_SLV_MON (on page 143)	0x918
Reserved		TWI_TO (on page 144)		0x91C
Reserved		TWI_IR_MASK (on page 144)		0x920
Reserved		TWI_IR_EN (on page 144)		0x924
Reserved		TWI_IR_DIS (on page 145)		0x928
GPIO_BYPASS_MODE (on page 146)				0xA00
GPIO_DIRECTION_MODE (on page 147)				0xA04
GPIO_OUTPUT_ENABLE (on page 148)				0xA08
GPIO_OUTPUT_VALUE (on page 149)				0xA0C
GPIO_INPUT_VALUE (on page 150)				0xA10
GPIO_INT_MASK (on page 151)				0xA14
GPIO_INT_MASK_CLR (Note: formerly GPIO_INT_ENABLE) (on page 152)				0xA18
GPIO_INT_MASK_SET (Note: formerly GPIO_INT_DISABLE) (on page 153)				0xA1C
GPIO_INT_STATUS (on page 154)				0xA20
Reserved		GPIO_INT_TYPE (on page 155)		0xA24
GPIO_INT_VALUE (on page 156)				0xA28
GPIO_INT_ON_ANY (on page 157)				0xA2C
Reserved				0xA30 to 0xAFC
Reserved		FCL_CTRL (on page 158)		0xB00
Reserved		FCL_STATUS (on page 159)		0xB04
Reserved		FCL_IODATA_IN (on page 159)		0xB08

Bit Offset				Base Offset
31:24	23:16	15:8	7:0	
Reserved			FCL_IODATA_OUT (on page 160)	0xB0C
Reserved			FCL_EN (on page 160)	0xB10
Reserved			FCL_TIMER_0 (on page 161)	0xB14
Reserved			FCL_TIMER_1 (on page 161)	0xB18
Reserved			FCL_CLK_DIV (on page 162)	0xB1C
Reserved			FCL_IRQ (on page 162)	0xB20
Reserved	FCL_TIMER_CTRL (on page 163)			0xB24
Reserved			FCL_IM (on page 163)	0xB28
Reserved			FCL_TIMER2_0 (on page 164)	0xB2C
Reserved			FCL_TIMER2_1 (on page 164)	0xB30
Reserved				0xB34 to 0xDFC
FCL_FIFO_DATA (on page 164)				0xE00

10.4 Register Descriptions

This contains the following sections:

- 10.4.1 PCI Type 0 Configuration Header Registers and Related Control Registers
- 10.4.2 Power Management Capability Registers
- 10.4.3 Message Signalled Interrupt Registers
- 10.4.4 PCI Express Capability Registers
- 10.4.5 Device Serial Number Registers
- 10.4.6 Virtual Channel Capability Registers
- 10.4.7 GN412x System Registers
- 10.4.8 2-Wire Interface Registers
- 10.4.9 GPIO Registers
- 10.4.10 FPGA Configuration Loader Registers

10.4.1 PCI Type 0 Configuration Header Registers and Related Control Registers

PCI_VENDOR

PCI Vendor ID

Address: 0x000

Size: 16-bits

Table 10-3: PCI_VENDOR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	VENDOR	R	0x1A39		**Vendor ID:** This register identifies Gennum as the vendor of the device.

PCI_DEVICE

PCI Device ID

Address: 0x002

Size: 16-bits

Table 10-4: PCI_DEVICE

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	DEVICE	R	0x0004		**Device ID:** A Device ID of 0x0004 identifies this device as either a GN4124 or a GN4121.

PCI_CMD

PCI Command Register

Address: 0x004

Size: 16-bits

Table 10-5: PCI_CMD

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:11	RESERVED	R	0x0		Reserved
10	INT_DIS	RW	0x0	1-0	**Interrupt Disable:** Disable INTx message generation. 1 Disable INTx message 0 Enable INTx message

Table 10-5: PCI_CMD

Bits	Mnemonic	Type	Reset	Valid Range	Description
9	FBB_EN	R	0x0		**Fast Back-to-Back Enable:** This bit is hard wired to '0' as it is not relevant for PCI Express.
8	SERR_EN	RW	0x0	1-0	**System Error Enable:** Controls the operation of the system error. See description of the SYS_ERR bit in the PCI_STAT register. 1 System error enabled: 0 System error disabled:
7	IDSEL_WAIT	R	0x0		**IDSEL Stepping / Wait Cycle Control:** does not apply to PCI Express.
6	PAR_EN	RW	0x0	1-0	**Parity Error Enable:** When PAR_EN is enabled, the PER_ERR bit in the PCI_STAT will be set if a poisoned TLP is received: 1 GN412x reports parity errors (enabled). 0 GN412x ignores parity errors.
5	VGA_EN	R	0x0		**VGA Palette Snoop**: This does not apply to PCI Express.
4	MWI_EN	R	0x0		**Memory Write and Invalidate**: This bit is hard wired to '0', as it is not relevant for PCI Express.
3	SPECIAL_EN	R	0x0		**Special Cycle Enable**: This bit is hard wired to '0', as it is not relevant for PCI Express.
2	MASTER_EN	RW	0x0	1-0	**PCI Master Enable** 1 GN412x can issue PCIe requests. This bit must be set '1' in order for local bus devices to initiate master cycles or for MSI requests to be initiated. 0 GN412x will not issue any PCIe requests including Memory, I/O, and MSI requests.
1	MEM_EN	RW	0x0	1-0	**Memory Access Enable** 1 GN412x responds to PCIe memory accesses. 0 GN412x ignores memory requests and generates a UR (Unsupported Request) response when a PCIe memory accesses is received.
0	IO_EN	RW	0x0	1-0	**I/O Access Enable**: The internal register space should only be mapped into memory space. Consequently, this register bit is hardwired to 0x0.

PCI_STAT

PCI Status Register

Address: 0x006

Size: 16-bits

Table 10-6: PCI_STAT

Bits	Mnemonic	Type	Reset	Valid Range	Description
15	PAR_ERR	RW1	0x0	1-0	**Parity Error:** Set to ‘1’ whenever the GN412x receives a Poisoned TLP, regardless of the state the Parity Error Enable bit (PAR_EN) in the Command register (PCI_CMD). Cleared by writing ‘1’ to this same bit.
14	SYS_ERR	RW1	0x0	1-0	**System Error:** Set to ‘1’ when the GN412x sends a PCIe ERR_FATAL or ERR_NONFATAL Message, and the SERR_EN bit in the PCI_CMD register is set (‘1’). Cleared by writing ‘1’ to this bit.
13	M_ABORT	RW1	0x0	1-0	**Master Abort:** Set to ‘1’ when the GN412x, acting as a Requester, receives a Completion with Unsupported Request Completion Status. Cleared by writing ‘1' to this bit.
12	T_ABORT_RX	RW1	0x0	1-0	**Received Target Abort:** Set to ‘1’ when the GN412x, acting as a Requester, receives a Completion with Completer Abort Completion Status. Cleared by writing ‘1' to this bit.
11	T_ABORT_TX	RW1	0x0	1-0	**Signalled Target Abort:** Set to ‘1’ when the GN412x completes a Request using Completer Abort Completion Status. Cleared by writing ‘1' to this bit.
10:9	DEVSEL	R	0x0		**Device Select Timing:** This has no relevance to PCI Express.
8	PAR_REP	RW1	0x0	1-0	**Data Parity Error Report**: Set to ‘1’ by the GN412x when the PAR_EN bit in PCI_CMD is enabled (set to ‘1’) and either of the following two conditions occurs: • GN412x receives a Completion marked poisoned • GN412x poisons a write Request Cleared by writing ‘1' back to the same bit.
7	FAST_BACK	R	0x0		**Fast Back-to-Back Target Enable**: This bit is hard wired to ‘0’, as it is not relevant for PCI Express.
6	RESERVED	R	0x0		Reserved
5	66MHZ	R	0x0		**66MHz Capable PCI Device**: This is obsolete for PCI Express, and it is hardwired to 0x0 for the GN412x.
4	CAP_LIST	R	0x1		**Capabilities List**: Set to ‘1’ to indicate that extended capabilities exist.
3	INT_STAT	R	0x0		**Interrupt Status**
2:0	RESERVED	R	0x0		Reserved

PCI_REVISION

PCI Revision ID

Address: 0x008

Size: 8-bits

Table 10-7: PCI_REVISION

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	REV	R			**Revision ID:** This register identifies the silicon stepping. 0x00 Indicates Revision 1 of the silicon 0x01 Indicates Revision 2 of the silicon

PCI_CLASS_CODE

PCI Class Code

Address: 0x009

Size: 24-bits

Table 10-8: PCI_CLASS_CODE

Bits	Mnemonic	Type	Reset	Valid Range	Description
23:16	BASE_CLASS	FR	0x0		**PCI Base Class Code:** May be set according to the needs of the application. See the PCI Local Bus Specification.
15:8	SUB_CLASS	FR	0x0		**PCI Sub Class Code:** May be set according to the needs of the application. See the PCI Local Bus Specification.
7:0	PROG_IF	FR	0x0		**PCI Programming Interface Code:** May be set according to the needs of the application. See the PCI Local Bus Specification.

PCI_CACHE

PCI Cache Line Size

Address: 0x00C

Size: 8-bits

Table 10-9: PCI_CACHE

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	LINE_SIZE	RW	0x0		**Cache Line Size**: Implemented for legacy purposes as a read/write register. It has no impact on the internal operation of the GN412x.

PCI_LATENCY

PCI Latency Timer

Address: 0x00D

Size: 8-bits

Table 10-10: PCI_LATENCY

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	LT	R	0x0		**Latency Timer**: These bits are hard wired to '0', as the latency timer is not relevant for PCI Express.

PCI_HEADER

PCI Header Type

Address: 0x00E

Size: 8-bits

Table 10-11: PCI_HEADER

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	HDR_TYPE	R	0x0		**Header Type**: Reads back as all '0' to indicate a type 0 configuration header.

PCI_BIST

PCI Built-in Self Test

Address: 0x00F

Size: 8-bits

Table 10-12: PCI_BIST

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	BIST	R	0x0		**Built in Self Test**: Not implemented; reads back as all 'zeroes'.

PCI_BAR0_LOW

PCI BAR0 Lower Bits

Address: 0x010

Size: 32-bits

Table 10-13: PCI_BAR0_LOW

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:4	BASE	RW[1]	0x0		**Base Address**: These bits determine the location of BAR0 in PCI Express memory address space. The lower bits of BASE are masked off (always read '0') as appropriate for the aperture size, which is determined by the SIZE0 field of the PCI_BAR_CONFIG register. This masking ensures that automatic configuration software is able to correctly determine the size of the aperture by writing '1' into all the bits and, then reading back '1' only in the bits that are relevant for the particular size of aperture. See PCI_BAR_CONFIG (on page 128) for more information.
3	PREFETCH	R	0x0	1-0	**Prefetchable**: Used to indicate that access to BAR0 may be prefetched. 1 Enable read prefetching for this BAR. 0 Disable read prefetching for this BAR. The setting of this bit has no effect on the operation of the GN412x.
2:1	TYPE	R	0x2		**Address Range Type**: Determines the size of the base address register: "00" Locate anywhere in 32-bit access space. PCI_BAR0_HIGH is disabled for this selection. "10" Locate anywhere in 64-bit access space. The upper address is taken from PCI_BAR0_HIGH. Others Reserved
0	IO	R	0x0	1-0	**IO or Memory Space Access**: The internal register space should only be mapped into memory space. Consequently, this register bit is hardwired to 0x0.

1. Lower bits are masked off according to the SIZE0 bits in the PCI_BAR_CONFIG register.

PCI_BAR0_HIGH

PCI BAR0 Upper Bits

Address: 0x014

Size: 32-bits

Table 10-14: PCI_BAR0_HIGH

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	BASE	RW	0x0		**Base Address bits 63:32**: These bits determine the location of BAR0 in 64-bit PCI Express memory space. When BAR0 is configured as a 64-bit BAR (PCI_BAR0_LOW.TYPE="10"), a "BAR hit" is only achieved when PCI Express address bits 63:32 match the value of PCI_BAR0_HIGH and there is a match with the PCI_BAR0_LOW. When PCI_BAR0_LOW.TYPE="00" (32-bit address space), then PCI_BAR0_HIGH.BASE will read back as all zero.

PCI_BAR2_LOW

PCI BAR2 Lower Bits

Address: 0x018

Size: 32-bits

Table 10-15: PCI_BAR2_LOW

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:4	BASE	RW[1]	0x0		**Base Address**: These bits determine the location of BAR2 in PCI Express memory address space. The lower bits of BASE are masked off (always read '0') as appropriate for the aperture size, which is determined by the SIZE2 field of the PCI_BAR_CONFIG register. This masking ensures that automatic configuration software is able to correctly determine the size of the aperture by writing '1' into all the bits and, then reading back '1' only in the bits that are relevant for the particular size of aperture. See PCI_BAR_CONFIG (on page 128) for more information.
3	PREFETCH	FR	0x0	1-0	**Prefetchable**: Used to indicate that access to BAR2 may be prefetched 1 Enable read prefetching for this BAR. 0 Disable read prefetching for this BAR. The setting of this bit has no effect on the operation of the GN412x.
2:1	TYPE	R	0x2		**Address Range Type**: Determines the size of the base address register: "00" Locate anywhere in 32-bit access space. PCI_BAR2_HIGH is disabled for this selection. "10" Locate anywhere in 64-bit access space. The upper address is taken from PCI_BAR2_HIGH. Others Reserved
0	IO	R	0x0	1-0	**IO or Memory Space Access**: The internal register space should only be mapped into memory space. Consequently, this register bit is hardwired to 0x0.

1. Lower bits are masked off according to the SIZE2 bits in the PCI_BAR_CONFIG register.

PCI_BAR2_HIGH

PCI BAR2 Upper Bits

Address: 0x01C

Size: 32-bits

Table 10-16: PCI_BAR2_HIGH

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	BASE	RW	0x0		**Base Address bits 63:32**: These bits determine the location of BAR2 in 64-bit PCI Express memory space. When BAR2 is configured as a 64-bit BAR (PCI_BAR2_LOW.TYPE="10"), a "BAR hit" is only achieved when PCI Express address bits 63:32 match the value of PCI_BAR2_HIGH, and there is a match with the PCI_BAR2_LOW. When PCI_BAR2_LOW.TYPE="00" (32 bit address space), then PCI_BAR2_HIGH.BASE will read back as all zero.

PCI_BAR4_LOW

PCI BAR4 Lower Bits

Address: 0x020

Size: 32-bits

Table 10-17: PCI_BAR4_LOW

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:12	BASE	RW	0x0		**Base Address**: These bits determine the location of the register access in the aperture PCI Express space. When configured as a 64-bit BAR (PCI_BAR4_LOW.TYPE="10"), a "BAR hit" is only achieved when PCI Express address bits 63:32 match the value of PCI_BAR4_HIGH, and there is a match with the PCI_BAR4_LOW. When PCI_BAR4_LOW.TYPE="00" (32,bit address space), then PCI_BAR4_HIGH.BASE will read back as all zero. Aperture size of BAR4 address space is fixed at 4096 bytes.
11:4	RESERVED	R	0x0		Reserved
3	PREFETCH	R	0x0	1-0	**Prefetchable**: Access to the internal registers of the GN412x should not be prefetched. Consequently, this register bit is hardwired to 0x0.
2:1	TYPE	R	0x2		**Address Range Type**: Determines the size of the base address register: "00" Locate anywhere in 32-bit access space. PCI_BAR4_HIGH is disabled for this selection. "10" Locate anywhere in 64-bit access space. The upper address is taken from PCI_BAR4_HIGH. Others Reserved
0	IO	R	0x0	1-0	**IO or Memory Space Access**: The internal register space should only be mapped into memory space. Consequently, this register bit is hardwired to 0x0.

PCI_BAR4_HIGH

PCI BAR4 Upper Bits

Address: 0x024

Size: 32-bits

Table 10-18: PCI_BAR4_HIGH

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	BASE	RW	0x0		**Base Address**: These bits determine the location of the register aperture in 64-bit PCI Express memory space. When BAR4 is configured as a 64-bit BAR (PCI_BAR4_LOW.TYPE="10"), a "BAR hit" is only achieved when PCI Express address bits 63:32 match the value of PCI_BAR0_HIGH and there is a match with the PCI_BAR4_LOW. When PCI_BAR4_LOW.TYPE="00" (32-bit address space), then PCI_BAR4_HIGH.BASE will read back as all zero.

PCI_CIS

Cardbus CIS Pointer

Address: 0x028

Size: 32-bits

Table 10-19: PCI_CIS

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	CIS	FR	0x00		**Cardbus CIS Pointer**:

PCI_SUB_VENDOR

PCI Subsystem Vendor ID

Address: 0x02C

Size: 16-bits

Table 10-20: PCI_SUB_VENDOR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	VENDOR	FR	0x1A39		**Subsystem Vendor**: Provided for the developer of the board employing the GN412x to identify the vendor of the subsystem. Required for plug-and-play operation.

PCI_SUB_SYS

PCI Subsystem ID

Address: 0x02E

Size: 16-bits

Table 10-21: PCI_SUB_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	SUBSYSTEM	FR	0x04		**Subsystem ID**: Provided for the developer of the board employing the GN412x to identify their subsystem to operating systems and drivers. Required for plug-and-play operation.

PCI_ROM_BASE

PCI Expansion ROM

Address: 0x030

Size: 32-bits

Table 10-22: PCI_ROM_BASE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	BASE	RW	0x0		**Base Address**: These bits determine the location of the ROM access aperture in PCI Express space. The bits of BASE are masked off (always read '0') as appropriate for the aperture size, which is determined by the ROM_SIZE field of the PCI_BAR_CONFIG register. This masking ensures that automatic configuration software is able to correctly determine the size of the aperture by writing '1' into all the bits and, then reading back '1' only in the bits that are relevant for the particular size of aperture. See PCI_BAR_CONFIG (on page 128) for more information. This field can only be written when ROM_SIZE field of the PCI_BAR_CONFIG register is set to non-zero value.
15:1	RESERVED	R	0x0		Reserved
0	ENABLE	RW	0x0	1-0	**Expansion ROM Enable**: 1 Enabled 0 Disabled. This field can only be written when ROM_SIZE field of the PCI_BAR_CONFIG register is set to non-zero value.

PM_CAP_POINTER

Power Management Extended Capability Pointer

Address: 0x034

Size: 8-bits

Table 10-23: PM_CAP_POINTER

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	POINTER	R	0x40		**Pointer to the PCI Express Capability**: The pointer allows a root complex to determine the location of the PCI Express Capability Registers.

PCI_INT_LINE

PCI Interrupt Line Register (Legacy)

Address: 0x03C

Size: 8-bits

Table 10-24: PCI_INT_LINE

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	LINE	RW	0x0		**Interrupt Line**: For auto-configuration reporting only. Has no effect on the internal operation of the GN412x.

PCI_INT_PIN

PCI Interrupt Pin Register (Legacy)

Address: 0x03D

Size: 8-bits

Table 10-25: PCI_INT_PIN

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	PIN	FR	0x0		**Interrupt Pin**: For auto-configuration reporting only. 0x00 = No INTx# pin used 0x01 = INTA# used 0x02 = INTB# used 0x03 = INTC# used 0x04 = INTD# used

PCI_MIN_GNT

PCI Minimum Grant Register (Legacy)

Address: 0x03E

Size: 8-bits

Table 10-26: PCI_MIN_GNT

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	MIN_GNT	R	0x0		**Minimum Grant**: This is a legacy PCI register and is unused in the GN412x.

PCI_MAX_LAT

PCI Maximum Latency Register (Legacy)

Address: 0x03F

Size: 8-bits

Table 10-27: PCI_MAX_LAT

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	MAX_LAT	R	0x0		**Maximum Latency**: This is a legacy PCI register and is unused in the GN412x.

10.4.2 Power Management Capability Registers

PM_CAP_ID

Power Management Capability ID

Address: 0x040

Size: 8-bits

Table 10-28: PM_CAP_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	ID	R	0x01		**Power Management Capability ID**: Indicates that power management capability is implemented.

PM_NEXT_ID

Power Management Next Capability Pointer

Address: 0x041

Size: 8-bits

Table 10-29: PM_NEXT_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	OFFSET	R	see text		**Offset to the Next Capability ID**: This is a pointer to the next capability register set in the PCI-compatible configuration space. If the MSI Capability is enabled the offset value will be 0x48, else the value will be 0x58. Refer to 4.3.6 MSI Capability for more details.

PM_CAP

Power Management Capability Register

Address: 0x042

Size: 16-bits

Table 10-30: PM_CAP

Bits	Mnemonic	Type	Reset	Valid Range	Description
15	PME_COLD	FR	0x0	0-1	**PME Support for D3$_{cold}$**: $\overline{PME}$ is not asserted for D3$_{cold}$.
14:11	PME_SUPT	FR	0x0		**PME Support**: Indicates the power states in which the function may assert $\overline{PME}$. A value of '0' for any bit indicates that the function is not capable of asserting $\overline{PME}$ while in that power state. See the following table: BIT — Value — Power State Support 11 — "XXXX1" — $\overline{PME}$ can be asserted from D0 12 — "XXX1X" — $\overline{PME}$ can be asserted from D1 13 — "XX1XX" — $\overline{PME}$ can be asserted from D2 14 — "X1XXX" — $\overline{PME}$ can be asserted from D3$_{hot}$
10	D2_SUPT	FR	0x0	0-1	**D2 Support**: When set to '1', support for the D2 power state is enabled. This has no effect on the internal operation of the GN412x.
9	D1_SUPT	FR	0x0	0-1	**D1 Support**: When set to '1', support for the D1 power state is enabled. This has no effect on the internal operation of the GN412x.
8:6	AUX_CURRENT	R	0x0		**Auxiliary Current**: Hardwired to 0x0.
5	DSI	FR	0x0	0-1	**Device Specific Initialization**: For operating system reporting purposes. This has no effect on the internal operation of the GN412x.
4	RESERVED	R	0x0		Reserved
3	PME_CLK	R	0x0		**PME Clock**: Does not apply to PCI Express.
2:0	PM_VER	R	0x3		**Power Management Version**: The value 0f 0x3 indicates version 1.2 of the PCI Power Management Specification.

PM_CSR

Power Management Control/Status Register

Address: 0x044

Size: 16-bits

Table 10-31: PM_CSR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15	PME_STAT	RW	—		**PME Status**: This bit is set when the function would normally assert the $\overline{PME}$ signal (from $D3_{cold}$) independent of the state of the PME_EN bit. Writing a "1" to this bit will clear it and cause the function to stop asserting a $\overline{PME}$ (if enabled). Writing a "0" has no effect.
14:13	DATA_SCALE	FR	0x0	0-3	**Data Scale**: Reports the scale factor for power consumption/dissipation data.
12:9	DATA_SEL	RW	0x0	0-7	**Data Select**: Selects one of the power consumption/dissipation data entries. This register value works with PM_DATA (on page 105), which will read this selected data. The mapping is as follows: 0 Power consumed in D0 state 1 Power consumed in D1 state 2 Power consumed in D2 state 3 Power consumed in D3 state 4 Power dissipated in D0 state 5 Power dissipated in D1 state 6 Power dissipated in D2 state 7 Power dissipated in D3 state
8	PME_EN	RW	0x0	0-1	**PME Enable**: When set to '1', the function may assert $\overline{PME}$. When cleared to '0', the function may not assert $\overline{PME}$.
7:2	RESERVED	R	0x0	N/A	Reserved
1:0	PWR_STATE	RW	0x0	0-3	**Power State**: This field is used to determine the power state of the power management function, and to set the function into a new power state. These bits do not alter the internal operation of the GN412x. Instead, they are used to drive the signals PWR_STATE, so that the attached circuit can use the information to gate clocks, disable power supplies, or other changes that affect power consumption. The definition of the field values is provided in the following table: PWR_STATE Power State "00" D0 "01" D1 "10" D2 "11" $D3_{hot}$

PM_CSR_BSE

PM_CSR Bridge Support Extensions Register

Address: 0x046

Size: 8 -bits

Table 10-32: PM_CSR_BSE

Bits	Mnemonic	Type	Reset	Valid Range	Description
7	BPCC_EN	R	0x0	0-1	**Bus Power/Clock Control Enable**: Only implemented in PCI-to-PCI bridges.
6	B2_B3	R	0x0	0-1	**B2/B3 Support for $D3_{hot}$**: Only implemented in PCI-to-PCI bridges.
5:0	RESERVED	R	0x0		Reserved

PM_DATA

Power Management Data Register

Address: 0x047

Size: 8-bits

Table 10-33: PM_DATA

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	DATA	R	0x0		**Power Consumption/Dissipation Data**: This register reports the data selected (by PM_CSR.DATA_SEL) from the consumption/dissipation data entries.

10.4.3 Message Signalled Interrupt Registers

MSI_CAP_ID

MSI Capability ID

Address: 0x048

Size: 8-bits

Table 10-34: MSI_CAP_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	ID	R	0x05		**MSI Capability ID**: Indicates that PCI Express capability is implemented.

MSI_NEXT_ID

MSI Next Capability Pointer

Address: 0x049

Size: 8-bits

Table 10-35: MSI_NEXT_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	OFFSET	R	0x58		**Offset to the Next Capability ID**: This points to the PCI Express Capability.

MSI_CONTROL

MSI Message Control Register

Address: 0x04A

Size: 16-bits

Table 10-36: MSI_CONTROL

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:8	RESERVED	R	0x0		Reserved
7	64BIT	R	0x1		**64-Bit Capable**: The GN412x supports 64-bit MSI.
6:4	MUL_MSG_EN	RW	0x0	0-7	**Multiple Message Enable**: Number of messages allocated by the host system.

Table 10-36: MSI_CONTROL

Bits	Mnemonic	Type	Reset	Valid Range	Description
3:1	MUL_MSG_CAP	FR	0x0	0-7	**Multiple Message Capable**: Number of messages requested for allocation.
0	MSI_EN	RW	0x0	0-1	**MSI Enable**: Must be set to '1' in order for interrupt messages to be transmitted by the GN412x. Setting to '0' enables INTx Messages to deliver interrupts (i.e. legacy interrupts).

MSI_ADDRESS_LOW

MSI Lower Address

Address: 0x04C

Size: 32-bits

Table 10-37: MSI_ADDRESS_LOW

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:2	A_LOW	RW	0x00		**Address Low**: Corresponds to A(31:2) for the message address. Note that address 1:0 is always "00".
1:0	RESERVED	R	0x0		Reserved

MSI_ADDRESS_HIGH

MSI Upper Address

Address: 0x050

Size: 32-bits

Table 10-38: MSI_ADDRESS_HIGH

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	A_HIGH	RW	0x00		**Address High**: Corresponds to A(63:32) for the message address.

MSI_DATA

MSI Message Data

Address: 0x054

Size: 16-bits

Table 10-39: MSI_DATA

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	DATA	RW	0x00		**Message Data**: Controls the lower bits of the data portion of the interrupt message.

10.4.4 PCI Express Capability Registers

PCIE_CAP_ID

PCI Express Capability ID

Address: 0x058

Size: 8-bits

Table 10-40: PCIE_CAP_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	ID	R	0x10		**PCI Express Capability ID**: Indicates that PCI Express capability is implemented.

PCIE_NEXT_ID

PCI Express Next Capability Pointer

Address: 0x059

Size: 8-bits

Table 10-41: PCIE_NEXT_ID

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:0	OFFSET	R	0x0		**Offset to the Next Capability ID**: This is a pointer to the next capability. This is the final capability register set in the PCI-compatible configuration space.

PCIE_CAPABILITY

PCI Express Capabilities

Address: 0x05A

Size: 16-bits

Table 10-42: PCIE_CAPABILITY

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:14	RESERVED	R	0x0		Reserved
13:9	INT_MSG_NUM	FR	0x1	0-31	**Interrupt Message Number**: This field contains the vector number that will be used for any MSI interrupts that are generated in relation to the status bits in the PCIe Capability structure.

Table 10-42: PCIE_CAPABILITY

Bits	Mnemonic	Type	Reset	Valid Range	Description
8	SLOT	R	0x0		**Slot Implemented**: Set to 0x0 for an endpoint device.
7:4	TYPE	FR	0x0	0-15	**PCI Express Device/Port Type**: Set to 0x0 to indicate an endpoint device.
3:0	VERSION	R	0x2		**PCI Express Capability Version**: Hardwired to 0x2, as per the PCI Express Capability Structure Expansion ECN.

PCIE_DEVICE_CAP

PCI Express Device Capabilities

Address: 0x05C

Size: 32-bits

Table 10-43: PCIE_DEVICE_CAP

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:28	RESERVED	R	0x0		Reserved
27:26	PWR_SCALE	FR	0x0		**Captured Slot Power Limit Scale**:
25:18	PWR_VALUE	FR	0x0		**Captured Slot Power Limit Value**:
17:16	RESERVED	R	0x0		Reserved
15	ROLE_ERROR	FR	0x1		**Role-Based Error Reporting**: Required by all PCIe 1.1 compliant devices. Must be set to '1'.
14:12	RESERVED	R	0x0		Reserved
11:9	L1_LATENCY	FR	0x0	0-7	**Endpoint L1 Acceptable Latency**: This field indicates the acceptable latency that an endpoint can withstand due to the transition from L1 state to the L0 state. The default value of 0x0 should typically be used.
8:6	L0S_LATENCY	FR	0x0	0-7	**Endpoint L0s Acceptable Latency**: This field indicates the acceptable latency that an endpoint can withstand due to the transition from L0s state to the L0 state. The default value of 0x0 should typically be used.
5	TAG_SIZE	FR	0x0		**Extended Tag Field Supported**: Set according to bit 8 in PCIE_DCR on page 111.
4:3	PHANTOM	R	0x0		**Phantom Functions Supported**:
2:0	MAX_PAYLOAD	FR	0x0	0-1	**Max_Payload_Size Supported**: This field indicates the maximum payload size that the device / function can support for TLPs. This is written for PCIe 1.1 compliancy with advanced error reporting.

PCIE_DCR

PCI Express Device Control Register

Address: 0x060

Size: 16-bits

Table 10-44: PCIE_DCR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15	BRIDGE_RETRY	FR	0x0		**Bridge Configuration Retry Enable**: Not used by the GN412x. Used only for PCI Express to PCI/PCI-X bridges. Must be set to '0'.
14:12	MAX_READ_SIZE	RW	0x2	0-5	**Maximum Read Request Size**: This field sets the maximum read request size for the GN412x as a requester. Defined encodings for this field are: MAX_READ_SIZE Maximum Read Request Size 0x0 128 0x1 256 0x2 512 0x3 1024 0x4 2048 0x5 4096 others Reserved **Note**: The industry standard is 512, but the GN412x can accommodate higher maximum read request sizes. The PCIe enumeration process done by the system BIOS will update the register accordingly, based on the available hardware.
11	NO_SNOOP	RW	0x1	0-1	**Enable No Snoop**: Controls the No Snoop setting of requester attributes.
10	AUX_PWR_PM_EN	R	0x0	0-1	**Auxiliary (AUX) Power PM Enable**: The GN412x devices do not use this feature.
9	PHANTOM_EN	R	0x0		**Phantom Functions Enable:** The GN412x doesn't use this feature.
8	TAG_SIZE	FR	0x0	0-1	**Extended Tag Field Enabled:** When set, this bit enables the GN412x to use an 8-bit tag field as a requester. If the bit is cleared, the GN412x is restricted to a 5-bit tag field.
7:5	MAX_PAYLOAD	RW	0x0	0-2	**Maximum Payload Size:** Written by the host system to specify the maximum payload size that the GN412x will transmit. 0x0 128 0x1 256 0x2 512 others Reserved
4	RELAX	RW	0x1	0-1	**Enable Relaxed Ordering:** If this bit is set, the device is permitted to set the Relaxed Ordering bit in the Attributes field of transactions. It initiates that transactions do not require strong write ordering.
3	UNSUPPORT_EN	RW	0x0	0-1	**Unsupported Request Reporting Enable**: This bit, in conjunction with other bits, controls the signalling of Unsupported Requests by sending Error Messages.
2	ERR_FATAL_EN	RW	0x0	0-1	**Fatal Error Reporting Enable**: This bit, in conjunction with other bits, controls sending ERR_FATAL Messages.

Table 10-44: PCIE_DCR

Bits	Mnemonic	Type	Reset	Valid Range	Description
1	ERR_NONFATAL_EN	RW	0x0	0-1	**Non-Fatal Error Reporting Enable**: This bit, in conjunction with other bits, controls sending ERR_NONFATAL Messages.
0	ERR_COR_EN	RW	0x0	0-1	**Correctable Error Reporting Enable**: This bit, in conjunction with other bits, controls sending ERR_COR Messages.

PCIE_DSR

PCI Express Device Status Register

Address: 0x062

Size: 16-bits

Table 10-45: PCIE_DSR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	R	0x0		Reserved
5	PENDING	R	0x0		**Transactions Pending**: When set, this bit indicates that the GN412x has issued Non-Posted Requests, which have not been completed. This bit is cleared only when all outstanding Non-Posted Requests have completed, or have been terminated by the Completion Timeout mechanism.
4	AUX_PWR_DET	R	—		**Auxiliary Power Detected**: This bit reflects the state of the AUX_PWR_DET setting in PCI_SYS_CFG_SYSTEM register.
3	UNSUPPORT_DET	RW	0x0	0-1	**Unsupported Request Detect**: This bit indicates that the GN412x received an Unsupported Request. Error. Errors are logged in this register, regardless of whether error reporting is enabled or not in PCIE_DCR.UNSUPPORT_EN.
2	ERR_FATAL_DET	RW	0x0	0-1	**Fatal Error Detected**: This bit indicates status of fatal errors detected. Errors are logged in this register, regardless of whether error reporting is enabled or not in PCIE_DCR.ERR_FATAL_EN.
1	ERR_NONFATAL_DET	RW	0x0	0-1	**Non-Fatal Error Detected**: This bit indicates status of non fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in PCIE_DCR.ERR_NONFATAL_EN.
0	ERR_COR_DET	RW	0x0	0-1	**Correctable Error Detected**: This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in PCIE_DCR.ERR_COR_EN.

PCIE_LINK_CAP

PCI Express Link Capabilities Register

Address: 0x064

Size: 32-bits

Table 10-46: PCIE_LINK_CAP

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:24	PORT	FR	0x0		**Port Number**:
23:21	RESERVED	R	0x0		Reserved
20	DLL_ACTIVE	FR	0x0		**Data Link Layer Active Reporting Capable**: Not used by an endpoint device. Must be set to '0'.
19	SUPRIZE	FR	0x0		**Surprise Down Error Reporting Capable**: Not used by an endpoint device. Must be set to '0'.
18	CLOCK_PM	FR	0x0		**Clock Power Management**: Not supported by the GN412x.
17:15	L1_EXIT	FR	0x7	0-7	**L1 Exit Latency**:L1s exit latency for the link.
14:12	L0S_EXIT	FR	0x7	0-7	**L0s Exit Latency**: L0s exit latency for the link.
11:10	ASPM_SUPPORT	FR	0x3	0-3	**Active State Power Management (ASPM) Support**
9:4	MAX_LINK_ WIDTH	FR	see text		**Maximum Link Width**: This value must be set to 0x4 for the GN4124 (4 lanes). This value must be set to 0x1 for the GN4121 (1 lane).
3:0	MAX_LINK_ SPEED	FR	0x1		**Maximum Link Speed**: The value of 0x1 indicates 2.5Gb/s per lane. Must set to 0x1.

PCIE_LCR

PCI Express Link Control Register

Address: 0x068

Size: 16-bits

Table 10-47: PCIE_LCR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:9	RESERVED	R	0x0		Reserved
8	CLOCK_PM_EN	RW	0x0		**Enable Clock Power Management**: This bit is only RW if PCIE_LINK_CAP.CLOCK_PM is set to 1.
7	EXT_SYNC	RW	0x0	0-1	**Extended Sync**: When set, this forces the transmission of additional ordered sets when exiting the L0s state and when in the recovery state. This mode provides external devices (e.g., logic analyzers) monitoring the link, time to achieve bit and symbol lock, before the link enters the L0 state and resumes communication.
6	COMMON_CLK	RW	0x0	0-1	**Common Clock Configuration**: This bit, when set, indicates that this component and the component at the opposite end of this link are operating with a distributed common reference clock. A value of 0x0 indicates that the GN412x and the component at the opposite end of this link are operating with asynchronous reference clock. This value is used to report the correct L0s and L1 exit latencies. See 3.4 Clock Configuration. Link retrain is required when this bit is changed.
5	LINK_RETRAIN	R	0x0		**Retrain Link**: Hardwired to 0x0 for the GN412x as an endpoint.
4	LINK_DISABLE	R	0x0		**Link Disable**: Hardwired to 0x0 for the GN412x as an endpoint.
3	RCB	RW	0x0	0-1	**Read Completion Boundary (RCB)**: May be set by configuration software to indicate the RCB value of the root port upstream from the endpoint. The setting has no effect on the operation of the GN412x. Defined encodings are: 0x0 64 byte 0x1 128 byte
2	RESERVED	R	0x0		Reserved
1:0	ASPM_CTL	RW	0x0	0-3	**Active State Link PM Control**: This field controls the level of ASPM supported on the given PCI Express link. Defined encodings are: 0x0 Disabled 0x1 L0s entry enabled 0x2 L1 entry enabled 0x3 L0s and L1 entry enabled

PCIE_LSR

PCI Express Link Status Register

Address: 0x06A

Size: 16-bits

Table 10-48: PCIE_LSR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:14	RESERVED	R	0x0		Reserved
13	DLL_ACTIVE	R	0x0		**Data Link Layer Active**: Not used by an endpoint device.
12	CLOCK_MODE	FR	0x1		**Slot Clock Configuration**: When set to '1', this bit indicates that the component uses the same physical reference clock that the platform provides on the connector. When cleared to '0', an asynchronous reference clock is used. See 3.4 Clock Configuration.
11	LINK_TRAIN	R	0x0		**Link Training**: Hardwired to 0x0 for the GN412x as an endpoint.
10	LINK_ERROR	R	0x0		**Link Error**: No longer used in version 1.1 of the PCI Express Specification.
9:4	LINK_WIDTH	R	0x0	1, 2, 4	**Negotiated Link Width**: Indicates the negotiated width of the PCI Express link. The possible values are: 0x1 single lane 0x2 2 lanes (Not applicable for the GN4121) 0x4 4 lanes (Not applicable for the GN4121)
3:0	LINK_SPEED	R	0x1		**Link Speed**: Indicates the negotiated Link speed of the PCI Express Link. The value of 0x1 indicates 2.5Gb/s per lane.

10.4.5 Device Serial Number Registers

Note: Registers at address 0x100, 0x104 and 0x108 will become read only and have a value of 0x0 unless both the Device Serial number capability and the Virtual Channel capability are enabled, via PCI_SYS_CFG_SYSTEM register.

DSN_CAP

PCI Express Device Serial Number Capability

Address: 0x100

Size: 32-bits

Table 10-49: DSN_CAP

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:20	NEXT	R	See text.		**Pointer to Next Enhanced Capability**: Offset to the next PCI Express extended capability register set. If the Virtual Channel capability is enabled the offset value will be 0x400, else the offset value will be 0x0.
19:16	VERSION	R	0x1		**Capability Version**:
15:0	ID	R	0x0003		**Capability Identifier**: Set to 0x3 to indicate that it is the device serial number capability.

DSN_LOW

PCI Express Device Serial Number Lower Bits

Address: 0x104

Size: 32-bits

Table 10-50: DSN_LOW

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	SN	FR	0x0		**Serial Number Low Bits**:

DSN_HIGH

PCI Express Device Serial Number Upper Bits

Address: 0x108

Size: 32-bits

Table 10-51: DSN_HIGH

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	SN	FR	0x0		**Serial Number High Bits**:

10.4.6 Virtual Channel Capability Registers

Note: A memory alias to the Virtual Channel Capability Register set at address 0x100 is automatically enabled when Device Serial Number capability is disabled and Virtual Channel capability is enabled. All Virtual Channel Capability requires both Device Serial Number capability and Virtual Channel Capability enable, via the PCI_SYS_CFG_SYSTEM register, before any changes can be made. This applies to all FR and RW register fields.

VC_CAP

Virtual Channel Capability

Address: 0x400

Size: 32-bits

Table 10-52: VC_CAP

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:20	NEXT	R	0x00		**Pointer to Next Enhanced Capability**: Hardwired to 0x0 to indicate that this is the last capability in the list.
19:16	VERSION	R	0x1		**Capability Version**:
15:0	ID	R	0x0002		**Capability Identifier**: Set to 0x2 to indicate that it is the virtual channel capability (non-multi-function).

VC_PORT_CAP_1

Virtual Channel Port Capability - Register 1

Address: 0x404

Size: 32-bits

Table 10-53: VC_PORT_CAP_1

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:12	RESERVED	R	0x0		Reserved
11:10	TABLE_SIZE	R	0x0		**Port Arbitration Table Entry Size**: Set to 0x0 for an endpoint device.
9:8	REF_CLK	R	0x0		**Reference Clock**: Set to 0x0 for an endpoint device.
7	RESERVED	R	0x0		Reserved
6:4	LP_VC_COUNT	FR	0x0	0-7	**Low Priority Extended VC Count**: Indicates the number of (extended) virtual channels, in addition to the default VC belonging to the low-priority VC (LPVC) group that has the lowest priority with respect to other VC resources in a strict priority VC Arbitration.

Table 10-53: VC_PORT_CAP_1

Bits	Mnemonic	Type	Reset	Valid Range	Description
3	RESERVED	R	0x0		Reserved
2:0	VC_COUNT	FR	0x0	0-1[1]	**Extended VC Count**: Indicates the number of (extended) virtual channels, in addition to the default VC supported by this device.

1. VC_COUNT should be set to 0x0 for the GN4121.

VC_PORT_CAP_2

Virtual Channel Port Capability - Register 2

Address: 0x408

Size: 32-bits

Table 10-54: VC_PORT_CAP_2

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:24	TABLE_OFFSET	R	0x0		**VC Arbitration Table Offset**: Set to 0x0. GN412x does not support VC arbitration table.
23:8	RESERVED	R	0x0		Reserved
7:0	ARBITRATION	R	0x0		**VC Arbitration Capability**:

VC_PCR

Virtual Channel Port Control Register

Address: 0x40C

Size: 16-bits

Table 10-55: VC_PCR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:4	RESERVED	R	0x0		Reserved
3:1	ARB_SELECT	RW	0x0		**VC Arbitration Select**: Select one of the supported LPVC scheme.
0	LOAD_TABLE	R	0x0		**Load VC Arbitration Table**: Hardwired to 0x0, as the GN412x does not support arbitration table.

VC_PSR

Virtual Channel Port Status Register

Address: 0x40E

Size: 16-bits

Table 10-56: VC_PSR

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:1	RESERVED	R	0x0		Reserved
0	TABLE_STATUS	R	0x0		**VC Arbitration Table Status**: Hardwired to 0x0, as the GN412x is an endpoint only.

VC_RESOURCE_CAP0

Virtual Channel 0 Resource Capability Register

Address: 0x410

Size: 32-bits

Table 10-57: VC_RESOURCE_CAP0

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:24	TABLE_OFFSET	R	0x0		**Port Arbitration Table Offset**: Hardwired to 0x0, as the GN412x is an endpoint only.
23	RESERVED	R	0x0		Reserved
22:16	MAX_TSLOTS	R	0x0		**Maximum Time Slots**: Hardwired to 0x0, as the GN412x is an endpoint only.
15	REJECT_SNOOP	R	0x0		**Reject Snoop Transactions**: Hardwired to 0x0, as the GN412x is an endpoint only.
14	APS	FR	0x0		**Advanced Packet Switching**: In previous versions of the PCI Express specification, this bit was used to indicate Advanced Packet Switching. System software must ignore the value read from this bit.
13:8	RESERVED	R	0x0		Reserved
7:0	ARB_CAP	R	0x0		**Port Arbitration Capability**: Hardwired to 0x0, as the GN412x is an endpoint only.

VC_RESOURCE_CR0

Virtual Channel 0 Resource Control Register

Address: 0x414

Size: 32-bits

Table 10-58: VC_RESOURCE_CR0

Bits	Mnemonic	Type	Reset	Valid Range	Description
31	ENABLE	R	0x1		**VC0 Enable**: Since VC0 is the default VC, it is always enabled ('1').
30:27	RESERVED	R	0x0		Reserved
26:24	ID	R	0x0		**VC0 ID**: ID for the virtual channel. Since VC0 is the default VC, this field is RO, and it is hardwired to 0x0.
23:20	RESERVED	R	0x0		Reserved
19:17	ARB_SELECT	R	0x0		**Port Arbitration Select**: Hardwired to 0x0, as the GN412x is an endpoint only.
16	LOAD_TABLE	R	0x0		**Load Port Arbitration Table**: Hardwired to 0x0, as the GN412x is an endpoint only.
15:8	RESERVED	R	0x0		Reserved
7:0	TC_VC_MAP	RW	0xFF		**TC/VC0 Map**: This field indicates the TCs that are mapped to the VC resource. Bit locations within this field correspond to TC values. For example, when bit 7 is set in this field, TC7 is mapped to this VC resource. When more than one bit in this field is set, it indicates that multiple TCs are mapped to the VC resource. In order to remove one or more TCs from the TC/VC Map of an enabled VC, software must ensure that no new or outstanding transactions with the TC labels are targeted at the given link. Default value is 0xFF for VC0 to indicate that it will handle all TC values by default. Bit 0 of this register is read only, and it is hardwired to 0x1 while the upper bits are RW.

VC_RESOURCE_SR0

Virtual Channel 0 Status Control Register

Address: 0x41A

Size: 16-bits

Table 10-59: VC_RESOURCE_SR0

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:2	RESERVED	R	0x0		Reserved
1	NEGOTIATING	R	0x0		**VC Negotiation Pending**: This bit indicates whether the virtual channel negotiation (initialization or disabling) is in pending state. When this bit is set by hardware, it indicates that the VC resource has not completed the process of negotiation. This bit is cleared by hardware after the VC negotiation is complete (on exit from the FC_INIT2 state). Before using a virtual channel, software must check whether the VC Negotiation Pending fields for that virtual channel are cleared in both components on the link.
0	TABLE_STATUS	R	0x0		**Port Arbitration Table Status**: Hardwired to 0x0, as the GN412x is an endpoint only.

VC_RESOURCE_CAP1

Virtual Channel 1 Resource Capability Register

Address: 0x41C

Size: 32-bits

Table 10-60: VC_RESOURCE_CAP1

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:24	TABLE_OFFSET	R	0x0		**Port Arbitration Table Offset**: Hardwired to 0x0, as the GN412x is an endpoint only.
23	RESERVED	R	0x0		Reserved
22:16	MAX_TSLOTS	R	0x0		**Maximum Time Slots**: Hardwired to 0x0, as the GN412x is an endpoint only.
15	REJECT_SNOOP	R	0x0		**Reject Snoop Transactions**: Hardwired to 0x0, as the GN412x is an endpoint only.
14	APS	FR	0x0		**Advanced Packet Switching**: In previous versions of the PCI Express specification, this bit was used to indicate Advanced Packet Switching. System software must ignore the value read from this bit.
13:8	RESERVED	R	0x0		Reserved
7:0	ARB_CAP	R	0x0		**Port Arbitration Capability**: Hardwired to 0x0, as the GN412x is an endpoint only.

VC_RESOURCE_CR1

Virtual Channel 1 Resource Control Register
Note: The GN4121 device only supports 1 virtual channel.

Address: 0x420

Size: 32-bits

Table 10-61: VC_RESOURCE_CR1

Bits	Mnemonic	Type	Reset	Valid Range	Description
31	ENABLE	RW	0x0	0-1[1]	**VC1 Enable**: When set ('1'), VC1 is enabled. This bit should always be cleared to '0' for the GN4121. To enable VC1, the VC enable bits must be set in both components on a link. To disable a VC1, the VC1 enable bits must be cleared in both components on the link. Software must ensure that no traffic is using a VC1 at the time it is disabled. Software must fully disable VC1 in both components on the link before re-enabling it.
30:27	RESERVED	R	0x0		Reserved
26:24	ID	RW	0x0		**VC1 ID**: Assigns an ID for the virtual channel. This field cannot be modified, when a VC is already enabled.
23:20	RESERVED	R	0x0		Reserved
19:17	ARB_SELECT	R	0x0		**Port Arbitration Select**: Hardwired to 0x0, as the GN412x is an endpoint only.
16	LOAD_TABLE	R	0x0		**Load Port Arbitration Table**: Hardwired to 0x0, as the GN412x is an endpoint only.
15:8	RESERVED	R	0x0		Reserved
7:0	TC_VC_MAP	RW	0x00		**TC/VC1 Map**: This field indicates the TCs that are mapped to the VC resource. Bit locations within this field correspond to TC values. For example, when bit 7 is set in this field, TC7 is mapped to this VC resource. When more than one bit in this field is set, it indicates that multiple TCs are mapped to the VC resource. In order to remove one or more TCs from the TC/VC Map of an enabled VC, software must ensure that no new or outstanding transactions with the TC labels are targeted at the given Link. Bit 0 of this register is read only, and it is hardwired to 0x0.

1. Should be set to 0x0 for the GN4121

VC_RESOURCE_SR1

Virtual Channel 1 Resource Status Register
Note: The GN4121 device only supports 1 virtual channel.

Address: 0x426

Size: 16-bits

Table 10-62: VC_RESOURCE_SR1

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:2	RESERVED	R	0x0		Reserved
1	NEGOTIATING	R	0x0		**VC Negotiation Pending**: This bit indicates whether the virtual channel negotiation (initialization or disabling) is in pending state. When this bit is set by hardware, it indicates that the VC resource has not completed the process of negotiation. This bit is cleared by hardware after the VC negotiation is complete (on exit from the FC_INIT2 state). Before using a virtual channel, software must check whether the VC Negotiation Pending fields for that virtual channel are cleared in both components on the Link.
0	TABLE_STATUS	R	0x0		**Port Arbitration Table Status**: Hardwired to 0x0, as the GN412x is an endpoint only.

10.4.7 GN412x System Registers

PCI_SYS_CFG_SYSTEM

PCI Express System Configuration Registers

Address: 0x800

Size: 32-bits

Table 10-63: PCI_SYS_CFG_SYSTEM

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:21	RESERVED	RO	0x0		Reserved
20	AUX_PWR_DET	RW	0x0	0-1	**AUX Power Detected Setting**: Controls state of AUX_PWR_DET bit in PCIE_DSR register.
19	VC_EN	RW	0x0	0-1	**Virtual Channel Capability**: Indicates that the virtual channel capability is to be enabled. When this bit is '0' (cleared), the virtual channel capability will be excluded from the linked list of capabilities. While the virtual channel registers will remain operational, a BIOS or operating system will not see the virtual channel capability in the linked list of capabilities. Consequently, it will determine that the virtual channel capability is not implemented.
18	SN_EN	RW	0x0	0-1	**Serial Number Capability Enable**: Indicates that the serial number capability is to be enabled. When this bit is '0' (cleared), the serial number capability will be excluded from the linked list of capabilities. While the serial number registers will remain operational, a BIOS or operating system will not see the serial number capability in the linked list of capabilities. Consequently, it will determine that the serial number capability is not implemented. When SN_EN='1', then either the PCIE_NEXT_ID register, PM_NEXT_ID register, or MSI_NEXT_ID register will contain a pointer to SN_CAP_ID, so that a BIOS or operating system will detect the serial number capability and enumerate it accordingly.
17	MSI_EN	RW	0x0	0-1	**MSI Capability Enable**: Indicates that MSI capability is to be enabled. When this bit is '0' (cleared), the MSI capability will be excluded from the linked list of capabilities. While the registers will remain operational, a BIOS or operating system will not see the MSI capability in the linked list of capabilities. Consequently, it will determine that MSI is not implemented. When MSI_EN='1', then either the PCIE_NEXT_ID register or PM_NEXT_ID register will contain a pointer to MSI_CAP_ID, so that a BIOS or operating system will detect the MSI capability and enumerate it accordingly.
16	RESERVED	RO	0x0		Reserved

Table 10-63: PCI_SYS_CFG_SYSTEM

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:14	RSTOUT	RW	0x0	0-3	**Reset Output Control**: When written to "01", the reset output ($\overline{\text{RSTOUT33}}$) is de-asserted. When written to "00", $\overline{\text{RSTOUT33}}$ is asserted. If this bit field is written to "01" during 2-wire configuration download, the output reset is not de-asserted until after the full configuration download is complete. Writing any other value ("10" or "11") will result in the register bits remaining unchanged. This feature can be used to allow software to modify other bit fields in the PCI_SYS_CFG_SYSTEM register without changing $\overline{\text{RSTOUT33}}$. 0x00 To assert. 0x01 To de-assert. Others These values are ignored and have no effect.
13:12	LB_EN	RW	0x0	0-3	**Local Bus Enable**: When written to "00", the local bus output signals are either driven LOW or set to a high impedance mode. When written to "01", the local bus is enabled for normal operation. Writing any other value ("10" or "11") will result in the register bits remaining unchanged. 0x01 To enable. 0x00 To disable. 0x01 Normal operation. Others Written with other values has no effect, and it is ignored.
11:8	RESERVED	RO	0x0		Reserved
7:6	CFG_RETRY	FR	0x1	0-1	**Configuration Retry Enable**: When written to "01", any Configuration Request received by the GN412x will generate a Configuration Request Retry Status (CRS) in response. When written with "00", normal configuration access is performed. 0x00 Written when configuration is complete. 0x01 To enable return of CRS to host. OthersWritten with other values has no effect, and it is ignored. The default value of CFG_RETRY is dependant on the way that the GN412x is set up. See 3.1.3 Initialization from a 2-Wire EEPROM for more details.
5:0	RESERVED	RO	0x0		Reserved

LB_CTL

Local Bus Control Registers

Address: 0x804

Size: 32-bits

Table 10-64: LB_CTL

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:17	RESERVED	RO	0x0		Reserved
16	TWI_HOST_ MODE_EN	RW	0x0		Enable TWI host mode of operation. See 7.1 2-Wire Interface.
15:12	P2L_VC_ARB	RW	0x4	0-15	**Arbitration between Virtual Channels in the PCIe-to-Local Direction**: This bit field determines the priority between the two virtual channels for the use of the P2L_DATA outputs for the case that both VC have data to send to the local bus FPGA. The priority may be overridden by the state of the P_WR_RDY(1:0) pins (for GN4124, while only P_WR_RDY for GN4121)and the P_RD_D_RDY(1:0) pins (for GN4124, while only P_RD_D_RDY for GN4121). The priority is given as: P2L_VC_ARB — Arbitration Scheme 0 — VC1 high priority. Fixed priority scheme. 4 — Round robin with equal weighting. 8 — VC0 high priority. Fixed priority scheme.
11:10	RESERVED	RO	0x0		Reserved
9	P_RD_CREDIT1	RW	0x1	0-1	**Outstanding Read Credit Control for VC1**: Controls the number of outstanding read requests that the local bus target controller must track for VC1. Control of VC0 is independent. When P_RD_CREDIT1 is '0', then the GN412x will issue only a single read request for VC1, and then wait for a corresponding read completion before issuing another VC1 read request. With P_RD_CREDIT1 set to '1', the GN412x may issue up to three outstanding requests to VC1.
8	RD_MASK_EN1	RW	0x0	0-1	**Read requests masking enable for VC1**: If set, posted and completion requests are allowed to pass non-posted read requests.
7:2	RESERVED	RO	0x0		Reserved
1	P_RD_CREDIT0	RW	0x1	0-1	**Outstanding Read Credit Control for VC0**: Controls the number of outstanding read requests that the local bus target controller must track for VC0. When P_RD_CREDIT0 is '0', then the GN412x will issue only a single read request for VC1, and then wait for a corresponding read completion before issuing another VC0 read request. With P_RD_CREDIT0 set to '1', the GN412x may issue up to three outstanding requests to VC0.
0	RD_MASK_EN0	RW	0x0	0-1	**Read requests masking enable for VC0**: If set, posted and completion requests are allowed to pass non-posted read requests.

CLK_CSR

Clock Status Control Registers

Address: 0x808

Size: 32-bits

Table 10-65: CLK_CSR

Bits	Mnemonic	Type	Reset	Valid Range	Description
31	L_LOCK	RO	0x0	0-1	**LCLK PLL Lock Status**: Set by hardware (‘1’) when the local bus clock (LCLK/LCLKN) is locked. Cleared (‘0’) when the local bus clock is not locked.
30	P2L_LOCK	RO	0x0	0-1	**P2L DLL Lock Status**: Set by hardware (‘1’) when the DLL associated with P2L_CLKp/n is locked. Cleared (‘0’) when P2L_CLKp/n is not locked.
29	L2P_LOCK	RO	0x0	0-1	**L2P DLL Lock Status**: Set by hardware (‘1’) when the DLL associated with L2P_CLKp/n is locked. Cleared (‘0’) when P2L_CLKp/n is not locked.
28	FB	RW	0x0	0-1	**Test feature**: Should be left at the default ‘0’ for normal operation.
27	RST	RW	0x0	0-1	**PLL Reset**: Asserted ‘1’ to place the PLL into reset mode. RST should be zero under normal local bus operation.
26:20	DIVIN	RW	0x0		**Input Divider**: Sets the divider ratio for the input reference of the local bus clock PLL. See text.
19	L2P_DLL_RST	RO	0x0		Reserved. Set to ‘0’.
18:12	DIVFB	RW	0x13		**Feedback Divider**: Sets the divider ratio for the feedback path of the local bus clock PLL. This results in a clock multiplier being applied. See text.
11:10	RESERVED	RO	0x0		Reserved
9:4	DIVOT	RW	0x4		**Output Divider**: Sets the divider ratio for the output path of the local bus clock PLL. See text.
3	FS_EN	RW	0x1	0-1	**Enable for the PLL Internal Feedback Path**: Should be set to 1 for normal operation.
2:0	L_PLL_RANGE	RW	0x4	0-7	**Local Bus PLL Filter Range**: Sets the loop filter to work with the post-reference divider frequency (REFCK / DIVIN). Choose the highest valid range for the best jitter performance.

L_PLL_RANGE	Filter Range	L_PLL_RANGE	Filter Range
0x0	Bypass	0x4	21 to 42MHz
0x1	5 to 10MHz	0x5	34 to 68MHz
0x2	8 to 16MHz	0x6	54 to 108MHz
0x3	13 to 26MHz	0x7	88 to 200MHz

PCI_BAR_CONFIG

PCI BAR Configuration Register
Although these values are RW, they should only be set FR or by local processor method before BIOS enumeration. Changing the BAR sizes after enumeration may lead to improper address decoding and could cause a system crash.

Address: 0x80C

Size: 16-bits

Table 10-66: PCI_BAR_CONFIG

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:12	ROM_SIZE	RW	0x0		**ROM Size**: This determines the aperture size of PCI_ROM_BASE according to the following table:
					SIZE Bits / Size / Valid BASE Bits
					"0000" / Disabled / -
					"0001" / 2KB / 31:21
					"0010" / 4KB / 31:22
					"0011" / 8KB / 31:22
					"0100" / 16KB / 31:22
					"0101" / 32KB / 31:22
					"0110" / 64KB / 31:22
					"0111" / 128KB / 31:22
					"1000" / 256KB / 31:22
					"1001" / 512KB / 31:22
					"1010" / 1MB / 31:22
					"1011" / 2MB / 31:22
					"1100" / 4MB / 31:22
					"1101" / 8MB / 31:23
					"1110" / 16MB / 31:24
					"1111" / Reserved / Reserved
11:8	SIZE2	RW	0x0		**BAR2 Size**: This determines the aperture size of BAR2 according to the following table:
					SIZE2 Bits / Size / Valid BASE Bits (TYPE=32-bit / TYPE=64-bit)
					"0000" / 1MB / 31:20 / 63:20
					"0001" / 2MB / 31:21 / 63:21
					"0010" / 4MB / 31:22 / 63:22
					"0011" / 8MB / 31:23 / 63:23
					"0100" / 16MB / 31:24 / 63:24
					"0101" / 32MB / 31:25 / 63:25
					"0110" / 64MB / 31:26 / 63:26
					"0111" / 128MB / 31:27 / 63:27
					"1000" / 256MB / 31:28 / 63:28
					"1001" / 512MB / 31:29 / 63:29
					"1010" / 1GB / 31:30 / 63:30
					"1011" / 2GB / 31 / 63:31
					"1100" / 4GB / — / 63:32
					others / Reserved / Reserved / Reserved

Table 10-66: PCI_BAR_CONFIG

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:4	RESERVED	R	0x0		Reserved
3:0	SIZE0	RW	0x0		**BAR0 Size**: This determines the aperture size of BAR0 according to the following table:

SIZE0 Bits	Size	Valid BASE Bits	
		TYPE=32-bit	TYPE=64-bit
“0000”	1MB	31:20	63:20
“0001”	2MB	31:21	63:21
“0010”	4MB	31:22	63:22
“0011”	8MB	31:23	63:23
“0100”	16MB	31:24	63:24
“0101”	32MB	31:25	63:25
“0110”	64MB	31:26	63:26
“0111”	128MB	31:27	63:27
“1000”	256MB	31:28	63:28
“1001”	512MB	31:29	63:29
“1010”	1GB	31:30	63:30
“1011”	2GB	31	63:31
“1100”	4GB	—	63:32
others	Reserved	Reserved	Reserved

INT_CTRL

Interrupt Control Registers

Address: 0x810

Size: 32-bits

Table 10-67: INT_CTRL

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:4	RESERVED	RO	0x0		Reserved
3	I3_IN_EDGE	RW	0x0	0-1	**INT3 Edge or Level Sensitive Input**: Write ‘0’ to enable high-level sensitive trigger on external interrupt 3. Write ‘1’ to enable rising edge trigger on external interrupt 3.
2	I2_IN_EDGE	RW	0x0	0-1	**INT2 Edge or Level Sensitive Input**: Write ‘0’ to enable high-level sensitive trigger on external interrupt 2. Write ‘1’ to enable rising edge trigger on external interrupt 2.

Table 10-67: INT_CTRL

Bits	Mnemonic	Type	Reset	Valid Range	Description
1	I1_IN_EDGE	RW	0x0	0-1	**INT1 Edge or Level Sensitive Input**: Write '0' to enable high-level sensitive trigger on external interrupt 1. Write '1' to enable rising edge trigger on external interrupt 1.
0	I0_IN_EDGE	RW	0x0	0-1	**INT0 Edge or Level Sensitive Input**: Write '0' to enable high-level sensitive trigger on external interrupt 0. Write '1' to enable rising edge trigger on external interrupt 0.

INT_STAT

Interrupt Status Registers

Address: 0x814

Size: 32-bits

Table 10-68: INT_STAT

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved
15	GPIO	RO	0x0	0-1	**Interrupt status from the GPIO Block.** Read will not clear the interrupt source.
14	ALI6	RW	0x0	0-1	**Local Bus Receive Error**: This interrupt is asserted when the local bus RX_ERROR signal is driven by an external circuit. Write 1 to clear the interrupt.
13	ALI5	RW	0x0	0-1	**Local Bus Transmit Error**: This interrupt is asserted when the local bus TX_ERROR signal is driven by the GN412x. This can occur for the following reasons: • L2P FIFO in the GN412x overflows • Malformed packets are driven into the L2P interface. Write 1 to clear the interrupt.
12	ALI4	RW	0x0	0-1	**P2L FIFO Overflow Error for VC1**: The local bus P2L RX FIFO presents a full watermark indicator to halt the further popping of data. If the watermark level is incorrectly configured, then the FIFO may overflow. This interrupt is for debug purposes and should never be asserted in a correctly configured design. Write 1 to clear the interrupt.
11	ALI3	RW	0x0	0-1	**P2L FIFO Overflow Error for VC0**: See description for ALI4, P2L FIFO Overflow Error for VC1. Write 1 to clear the interrupt.
10	ALI2	RW	0x0	0-1	**BAR4 Target Error**: This interrupt is driven when an access to BAR4 for more than one DW is made. The only exception is the FCL data FIFO that is able to accept burst writes. BAR4 errors are prevented through proper implementation of the driver code and enforcing the programming model of the device. Write 1 to clear the interrupt.
9	ALI1	RW	0x0	0-1	**PCIe TX Controller Error**: Indicates that the PCI Express controller has rejected a request. This is for debug purposes and should not assert under normal operation. Write 1 to clear the interrupt.

Table 10-68: INT_STAT

Bits	Mnemonic	Type	Reset	Valid Range	Description
8	ALI0	RW	0x0	0-1	**PCIe Controller Internal Error**: Asserted due to receipt of malformed TLPs, completion time-outs etc. An error status register (PEX_ERROR_STAT) is also provided to identify the nature of the error. Write 1 to clear the interrupt.
7	INT3	RW	0x0	0-1	**External Interrupt 3**: **Note**: In legacy interrupt mode, the INTx assertion/de-assertion message, whether it is A, B, C or D, is set by the PCI_INT_PIN (on page 100). The GN412x is a single function device, so only one legacy interrupt can be valid. Write 1 to clear the interrupt.
6	INT2	RW	0x0	0-1	**External Interrupt 2**: **Note**: In legacy interrupt mode, the INTx assertion/de-assertion message, whether it is A, B, C or D, is set by the PCI_INT_PIN (on page 100). The GN412x is a single function device, so only one legacy interrupt can be valid. Write 1 to clear the interrupt.
5	INT1	RW	0x0	0-1	**External Interrupt 1**: **Note**: In legacy interrupt mode, the INTx assertion/de-assertion message, whether it is A, B, C or D, is set by the PCI_INT_PIN (on page 100). The GN412x is a single function device, so only one legacy interrupt can be valid. Write 1 to clear the interrupt.
4	INT0	RW	0x0	0-1	**External Interrupt 0**: **Note**: In legacy interrupt mode, the INTx assertion/de-assertion message, whether it is A, B, C or D, is set by the PCI_INT_PIN (on page 100). The GN412x is a single function device, so only one legacy interrupt can be valid. Write 1 to clear the interrupt.
3	SWI1	RW	0x0	0-1	**Software Interrupt 1**: This interrupt source is controlled through software. Written to '1' to assert the interrupt, and written to '0' to clear it.
2	SWI0	RW	0x0	0-1	**Software Interrupt 0**: This interrupt source is controlled through software. Written to '1' to assert the interrupt, and written to '0' to clear it.
1	FCLI	RO	0x0	0-1	**FCL Interrupt**: Status only. Read will not clear the interrupt source.
0	TWI	RO	0x0	0-1	**I^2C Interrupt**: Status only. Read will not clear the interrupt source.

PEX_ERROR_STAT

PEX (PCI Express Digital Core with PIPE and Generic Transaction Layer) Error Status Registers

Address: 0x818

Size: 32-bits

Table 10-69: PEX_ERROR_STAT

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:26	RESERVED	RO	0x0		Reserved
25:20	PEX_REJECT	RO	0x0		**PEX Reject Bus Status**:
19:18	RESERVED	RO	0x0		Reserved
17:0	PEX_ERROR	RO	0x0		**PEX Error Status**:

INT_CFG0-7[1]

Interrupt Configuration Registers

Address: 0x820 to 0x83C

Size: 8 x 32-bits

Table 10-70: INT_CFG0-7

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved
15	GPIO	RW	0x0	0-1	**GPIO Interrupt Enable**: '1' = enabled; '0' = disabled.
14	ALI6	RW	0x0	0-1	**AL Interrupt 6 Enable**: '1' = enabled; '0' = disabled.
13	ALI5	RW	0x0	0-1	**AL Interrupt 5 Enable**: '1' = enabled; '0' = disabled.
12	ALI4	RW	0x0	0-1	**AL Interrupt 4 Enable**: '1' = enabled; '0' = disabled.
11	ALI3	RW	0x0	0-1	**AL Interrupt 3 Enable**: '1' = enabled; '0' = disabled.
10	ALI2	RW	0x0	0-1	**AL Interrupt 2 Enable**: '1' = enabled; '0' = disabled.
9	ALI1	RW	0x0	0-1	**AL Interrupt 1 Enable**: '1' = enabled; '0' = disabled.
8	ALI0	RW	0x0	0-1	**AL Interrupt 0 Enable**: '1' = enabled; '0' = disabled.
7	INT3	RW	0x0	0-1	**External Interrupt 3 Enable**: '1' = enabled; '0' = disabled.
6	INT2	RW	0x0	0-1	**External Interrupt 2 Enable**: '1' = enabled; '0' = disabled.
5	INT1	RW	0x0	0-1	**External Interrupt 1 Enable**: '1' = enabled; '0' = disabled.
4	INT0	RW	0x0	0-1	**External Interrupt 0 Enable**: '1' = enabled; '0' = disabled.
3	SWI1	RW	0x0	0-1	**Software Interrupt Enable**: '1' = enabled; '0' = disabled.
2	SWI0	RW	0x0	0-1	**Software Interrupt Enable**: '1' = enabled; '0' = disabled.
1	FCLI	RW	0x0	0-1	**FCL Interrupt Enable**: '1' = enabled; '0' = disabled.
0	TWI	RW	0x0	0-1	**I^2C Interrupt Enable**: '1' = enabled; '0' = disabled.

1. There are 8 interrupt configuration registers. The first 4 (INT_CFG0-3) are used to generate MSI. The upper 4 (INT_CFG4-7) are routed to the GPIO block.

PCI_TO_ACK_TIME

PCI Time-out Acknowledgement Register

Address: 0x840

Size: 32-bits

Table 10-71: PCI_TO_ACK_TIME

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:20	RESERVED	RO	0x0		Reserved
19:0	TO_ACK_TIME	RW	0x0		This is the time to wait from receipt of PME_TURN_OFF message until PM schedules a PME_TO_ACK message. A value of 0 means never.

PEX_CDN_CFG1

PEX Configuration Registers

Address: 0x844

Size: 32-bits

Table 10-72: PEX_CDN_CFG1

Bits	Mnemonic	Type	Reset	Valid Range	Description
31	RESERVED	RO	0x0		Reserved
30:24	ACK_LAT	FRO	0x7F		
23:16	TX_NO_FTS	FRO	0x0F		
15	TP_CHECK_EN	FRO	0x0		DO NOT CHANGE!
14	EMPTY_TX_EN	FRO	0x0		DO NOT CHANGE!
13	RX_PHY_ER_EN	FRO	0x0		DO NOT CHANGE!
12	STRICT_TSMN	FRO	0x0		DO NOT CHANGE!
11	CPL_TO_RETRY	FRO	0x0		DO NOT CHANGE!
10	LINK_TYPE	FRO	0x0		DO NOT CHANGE!
9:0	REPLAY_LIMIT	FRO	0x3FF		

PEX_CDN_CFG2

PEX Configuration Registers

Address: 0x848

Size: 32-bits

Table 10-73: PEX_CDN_CFG2

Bits	Mnemonic	Type	Reset	Valid Range	Description
31	RESERVED	RO	0x0		Reserved
30:20	L1_ASPM_TIME	FRO	0x7FF		
19:18	RESERVED	RO	0x0		Reserved
17:8	L0S_IDLE_TIME	FRO	0x3FF		
7:4	RESERVED	RO	0x0		Reserved
3	L1_ASPM_EN	FRO	0x1	0-1	**L1 Enable (via ASPM)**:
2	L1R_EN	FRO	0x1	0-1	**L1 Re-entry Enable After Exit L1**:
1	L1_EN	FRO	0x1	0-1	**L1 Enable (via non-D0)**:
0	L0S_EN	FRO	0x1	0-1	**L0s Enable**:

PHY_TEST_CONTROL

PCI Express PHY Test Control Registers

Address: 0x84C

Size: 32-bits

Table 10-74: PHY_TEST_CONTROL

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:29	RESERVED	RO	0x0		Reserved
28	LBI_LPBK	RW	0x0	0-1	**Loopback LBI**:
27:24	PRBSSEL	RW	0x0	0-15	**PRBS Select:** Test mode only. Should be set to 0x0 for normal use.
23	PRBSSYNC	RW	0x0	0-1	**PRBS Sync:** Test mode only. Should be set to 0x0 for normal use.
22	PRBSERRORACK	RW	0x0	0-1	**PRBS Error Acknowledge:** Test mode only. Should be set to 0x0 for normal use.
21:18	LPBK	RW	0x0		**Loopback Enable:** Test mode only. Should be set to 0x0 for normal use.
17:14	PARLPBK	RW	0x0		Parallel Loopback Enable: Test mode only. Should be set to 0x0 for normal use.
13	TESTMODE	RW	0x0	0-1	**Test Mode Enable:** Test mode only. Should be set to 0x0 for normal use.
12	CTCDISABLE	RW	0x0	0-1	Test mode only. Should be set to 0x0 for normal use.
11:8	PRBSERRORH	RO	0x0		**Latched PRBS Error:** Test mode only.

Table 10-74: PHY_TEST_CONTROL

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:4	PRBSERROR	RO	0x0		**PRBS Error:** Test mode only.
3:0	PRBS_ERROR_ COUNT	RO	0x0	0-15	**PRBS Error Count:** Test mode only.

PHY_CONTROL

PCI Express PHY Control Registers

Address: 0x850

Size: 32-bits

Table 10-75: PHY_CONTROL

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:12	RESERVED	RO	0x0		Reserved
11:10	RXEQCTL	RW	0x0		
9	HIDRV	RW	0x0	0-1	
8	LODRV	RW	0x0	0-1	

Table 10-75: PHY_CONTROL

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:4	DTX	RW	0x0		Tx Drive Control: controls the relative drive of the PCI Express transmitter:
					DTX Value / Drive Strength Multiplier
					0x0 1.00
					0x1 1.05
					0x2 1.10
					0x3 1.15
					0x4 1.20
					0x5 1.25
					0x6 1.30
					0x7 1.35
					0x8 0.60
					0x9 0.65
					0xA 0.70
					0xB 0.75
					0xC 0.80
					0xD 0.85
					0xE 0.90
					0xF 0.95
3:0	DEQ	RW	0x0		Tx De-emphasis Control: controls the amount of PCI Express transmitter de-emphasis:
					DEQ Value / De-emphasis (dB)
					0x0 0.00
					0x1 -0.35
					0x2 -0.72
					0x3 -1.11
					0x4 -1.51
					0x5 -1.94
					0x6 -2.38
					0x7 -2.85
					0x8 -3.35
					0x9 -3.88
					0xA -4.44
					0xB -5.04
					0xC -5.68
					0xD -6.38
					0xE -7.13
					0xF -7.96

CDN_LOCK

Lock Register

Address: 0x854

Size: 32-bits

Table 10-76: CDN_LOCK

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:0	UNLOCK	RW	0x0		Value to unlock write access to PEX_CDN_CFG1 and PEX_CDN_CFG2 registers.

10.4.8 2-Wire Interface Registers

TWI_CTRL

2-Wire Interface Control Register

Address: 0x900

Size: 16-bits

Table 10-77: TWI_CTRL

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:14	DIVISOR_A	RW	0x0	0-3	**Divisor for Stage A Clock Divider**: Divides the input PCLK frequency by DIVISOR_A + 1.
13:8	DIVISOR_B	RW	0x0		**Divisor for Stage B Clock Divider**: Divides the output frequency from DIVISOR_A by DIVISOR_B + 1.
7	RESERVED	RW	0x0		Reserved.
6	CLR_FIFO	RW	0x0	0-1	**Clear FIFO**: This bit is implemented only if FIFO is implemented. Setting this bit to 1 automatically initializes the FIFO to all zeros and clears TWI_TR_SIZE register. This flag is used in both master and slave modes. It is automatically cleared on the next APB clock after being set.
5	SLVMON	RW	0x0	0-1	**Slave Monitor Mode**: 1 Monitor mode. 2-wire master transmits slave address and terminate if ACK. Repeat if NACK. 0 Normal operation. 2-wire master transmits slave address and transfers data if ACK. This bit is used in master mode only.
4	HOLD	RW	0x0	0-1	**Hold 2-wire SCLK LOW**: Holds until host services the data resources or clears this bit: 1 When no more data is available for transmit or no more data can be received, hold the SCLK line LOW until serviced by the host. 0 Allow the transfer to terminate as soon as all the data has been transmitted or received. This bit has the same meaning in both master and slave modes.
3	ACKEN	RW	0x0	0-1	**Enable Transmission of ACK When Master-Receiver**: 1 Acknowledge enabled, and ACK transmitted. 0 Acknowledge disabled, and NACK transmitted. This bit must always be set if FIFO is implemented.
2	NEA	RW	0x0	0-1	**Normal Extended Address**: Addressing modes are: 1 Normal (7-bit) address. 0 Extended (10-bit) address. This bit is used in master mode only.
1	MS	RW	0x0	0-1	Overall Interface Mode: 1 Master 0 Slave
0	RW	RW	0x0	0-1	**Direction of Transfer**: 1 Master receiver. 0 Master transmitter. This bit is used in master mode only.

TWI_STATUS

2-Wire Interface Status Register

Address: 0x904

Size: 16-bits

Table 10-78: TWI_STATUS

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:9	RESERVED	RW	0x0		Reserved.
8	BA	RW	0x0	0-1	**Bus Active**: Indicates there is an ongoing transfer on the 2-wire bus. The 2-wire controller is not necessarily involved in it.
7	RXOVF	RW	0x0	0-1	**Receiver Overflow**: This flag is set when the receiver receives a byte of data before the previous byte has been read by the host.
6	TXDV	RW	0x0	0-1	**Transmit Data Valid**: Indicates that there is still a byte of data to be transmitted by the interface. This bit is implemented only if FIFO is implemented.
5	RXDV	RW	0x0	0-1	**Receiver Data Valid**: Indicates that there is valid new data to be read from the interface.
4	RESERVED	RW	0x0		Reserved.
3	RXRW	RW	0x0	0-1	**RX Read / Write**: Indicates the mode of the transmission received from a master.
2:0	RESERVED	RW	0x0		Reserved.

TWI_ADDRESS

2-Wire Interface Address Register

Address: 0x908

Size: 16-bits

Table 10-79: TWI_ADDRESS

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:10	RESERVED	RW	0x0		Reserved.
9:0	ADD	RW	0x0		**Address**: Normal addressing mode uses ADD[6:0]. Extended addressing mode uses ADD[9:0]. APB write access to this register always initiates a transfer, if the 2-wire is in master mode.

TWI_DATA

2-Wire Interface Data Register

Address: 0x90C

Size: 16-bits

Table 10-80: TWI_DATA

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:8	RESERVED	RW	0x0		Reserved.
7:0	DATA	RW	0x0		**Data**: When written to, the data register sets data to transmit. When read from, the data register reads the last received byte of data.

TWI_IRT_STATUS

2-Wire Interface Interrupt Status Register. This register is cleared on reading.

Address: 0x910

Size: 16-bits

Table 10-81: TWI_IRT_STATUS

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:10	RESERVED	RO	0x0		Reserved.
9	ARB_LOST	RO	0x0	0-1	**Arbitration Lost for Master Mode**: This bit is set if a master loses bus ownership during a transfer due to ongoing arbitration.
8	RESERVED	RO	0x0		Reserved.
7	RX_UNF	RO	0x0	0-1	**FIFO Receive Underflow**: This bit is available only if FIFO implemented in the design. Set if the host attempts to read from the 2-wire data register (TWI_DATA) more times than the value of the transfer size register (TWI_TR_SIZE) plus one.
6	TX_OVF	RO	0x0	0-1	**FIFO Transmit Overflow**: This bit is available only if FIFO implemented in the design. Set if the host attempts to write to TWI_DATA more times than the FIFO depth.
5	RX_OVF	RO	0x0	0-1	**Receive Overflow for Master Read or Slave Receiver**: If FIFO is not implemented, this bit is set whenever there is valid data in TWI_DATA and a new byte is received. The second byte is not acknowledged, and contents of TWI_DATA remains unchanged. If FIFO is implemented, this bit is set whenever FIFO is full and a new byte is received. The new byte is not acknowledged, and contents of the FIFO remains unchanged.
4	SLV_RDY	RO	0x0	0-1	**Monitored Slave Ready**: This bit is set only if 2-wire interface is in master mode, TWI_CTRL.SLVMON bit is set and the addressed slave returns ACK.
3	TO	RO	0x0	0-1	**Transfer Time Out for Master and Slave Mode**: This bit is set whenever the 2-wire SCLK line is kept LOW for longer time than the value that is specified by the time out register (TWI_TO).

Table 10-81: TWI_IRT_STATUS

Bits	Mnemonic	Type	Reset	Valid Range	Description
2	NACK	RO	0x0	0-1	**Transfer Not Acknowledged**: **Master Mode**: This bit is set whenever the accessed slave responds with a NACK during address or data byte transfer. **Slave Mode**: This bit is set if FIFO is implemented, and it is in slave transmitter mode, when a master terminates the transfer before all data supplied by the host is transmitted.
1	DATA	RO	0x0	0-1	**More Data**: **Master Write or Slave Transmitter**: If FIFO is not implemented, this bit is set as soon as TWI_DATA is loaded in the output shift register of the 2-wire interface. If FIFO is implemented, this bit is set whenever there are only 2 bytes left in the FIFO. In slave transmitter mode, this bit is also set if the FIFO is emptied, but 2-wire master returned ACK on the last byte transmitted by the slave. **Master Read or Slave Receiver**: If FIFO is not implemented in the design, this bit is set whenever a byte is received and stored in the 2-wire register. If FIFO is implemented, this bit is set whenever there are only 2 free locations in the FIFO.
0	COMP	RO	0x0	0-1	**Transfer Complete**: **Master Mode**: In master write, this bit is always set when all the supplied data is successfully written to the slave and transfer is about to be terminated with STOP sequence. If FIFO is implemented and hold bit is set, COMP bit is also set as soon as the data is successfully written to the slave, but transfer is not terminated at this point. This allows for combined transfers to be performed even when FIFO is implemented. If the host clears the HOLD bit instead of continuing the transfer, COMP bit is set again during the STOP sequence generation. In master read, this bit is set when all the requested data has been successfully read from a slave and the transfer is to be terminated with STOP sequence. **Slave Mode**: In slave receive, this bit is set whenever the master terminates the transfer by generating STOP sequence. In slave transmit, this bit is set whenever all the data supplied by the host is transmitted and the last byte was not acknowledged by the master, which terminates the transfer with STOP sequence.

TWI_TR_SIZE

2-Wire Interface Transfer Size Register

Address: 0x914

Size: 8-bits

Table 10-82: TWI_TR_SIZE

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:4	RESERVED	RW	0x0		Reserved.
3:0	TRANSFER_SIZE	RW	0x0	0-15	**Transfer Size**: This register is available only if FIFO is implemented in the design. Its meaning varies according to the operating mode, as follows: • **Master Transmitter Mode**: Number of data bytes still not transmitted minus one. • **Master Receiver Mode**: Number of data bytes that are still expected to be received. • **Slave Transmitter Mode**: Number of bytes remaining in the FIFO after the master terminates the transfer. • **Slave Receiver Mode**: Number of valid data bytes in the FIFO. This register is cleared, if TWI_CTRL.CLR_FIFO is set.

TWI_SLV_MON

2-Wire Interface Slave Monitor Pause Register

Address: 0x918

Size: 8-bits

Table 10-83: TWI_SLV_MON

Bits	Mnemonic	Type	Reset	Valid Range	Description
7:4	RESERVED	RW	0x0		Reserved.
3:0	TRANSFER_SIZE	RW	0x0	0-15	**Transfer Size**: This register is associated with the slave monitor mode of the 2-wire interface. It is meaningful only when the module is in master mode and TWI_CTRL.SLVMON is set. This register defines the pause interval between consecutive attempts to address the slave once a write to TWI_ADDRESS is done by the host. It represents the number of SCLK cycles minus one between two attempts. The reset value of the register is 0, which results in the master again trying to access the slave immediately after unsuccessful attempt.

TWI_TO

2-Wire Interface Time Out Register

Address: 0x91C

Size: 16-bits

Table 10-84: TWI_TO

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	TIME_OUT	RW	0x1F		**Time Out**.

TWI_IR_MASK

2-Wire Interface Interrupt Mask Register

Address: 0x920

Size: 16-bits

Table 10-85: TWI_IR_MASK

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	IR_MASK	RO	0x2FF		**Interrupt Mask**: '1' indicates interrupt is disable for the respective bit. See TWI_IRT_STATUS (on page 141) for bit definition.

TWI_IR_EN

2-Wire Interface Interrupt Enable Register

Address: 0x924

Size: 16-bits

Table 10-86: TWI_IR_EN

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	IR_EN	WO	0x0		**Interrupt Enable**: Set the associated bit in TWI_IR_MASK to 0.

TWI_IR_DIS

2-Wire Interface Interrupt Disable Register

Address: 0x928

Size: 16-bits

Table 10-87: TWI_IR_DIS

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	IR_DIS	WO	0x0		**Interrupt Disable**: Set the associated bit in TWI_IR_MASK to 1.

10.4.9 GPIO Registers

GPIO_BYPASS_MODE

If bit *x* is 1, set pin *x* to bypass mode. If bit *x* is 0, set pin *x* to GPIO mode.
x can be GPIO 0 to 15.

Address: 0xA00

Size: 32-bits

Table 10-88: GPIO_BYPASS_MODE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	BM_15	RW	0x0	0-1	**General Purpose Data Bypass Mode 15.**
14	BM_14	RW	0x0	0-1	**General Purpose Data Bypass Mode 14.**
13	BM_13	RW	0x0	0-1	**General Purpose Data Bypass Mode 13.**
12	BM_12	RW	0x0	0-1	**General Purpose Data Bypass Mode 12.**
11	BM_11	RW	0x0	0-1	**General Purpose Data Bypass Mode 11.**
10	BM_10	RW	0x0	0-1	**General Purpose Data Bypass Mode 10.**
9	BM_9	RW	0x0	0-1	**General Purpose Data Bypass Mode 9.**
8	BM_8	RW	0x0	0-1	**General Purpose Data Bypass Mode 8.**
7	BM_7	RW	0x0	0-1	**General Purpose Data Bypass Mode 7.**
6	BM_6	RW	0x0	0-1	**General Purpose Data Bypass Mode 6.**
5	BM_5	RW	0x0	0-1	**General Purpose Data Bypass Mode 5.**
4	BM_4	RW	0x0	0-1	**General Purpose Data Bypass Mode 4.**
3	BM_3	RW	0x0	0-1	**General Purpose Data Bypass Mode 3.**
2	BM_2	RW	0x0	0-1	**General Purpose Data Bypass Mode 2.**
1	BM_1	RW	0x0	0-1	**General Purpose Data Bypass Mode 1.**
0	BM_0	RW	0x0	0-1	**General Purpose Data Bypass Mode 0.**

GPIO_DIRECTION_MODE

If bit *x* is 1, set pin *x* to input mode. If bit *x* is 0, set pin *x* to output mode.
x can be GPIO 0 to 15.

****** WARNING**: SETTING THE DIRECTION MODE TO OUTPUT INHIBITS THE GPIO INTERRUPTS FROM BEING GENERATED. ****

Address: 0xA04

Size: 32-bits

Table 10-89: GPIO_DIRECTION_MODE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	DM_15	RW	0x0	0-1	**General Purpose Data Direction Mode 15.**
14	DM_14	RW	0x0	0-1	**General Purpose Data Direction Mode 14.**
13	DM_13	RW	0x0	0-1	**General Purpose Data Direction Mode 13.**
12	DM_12	RW	0x0	0-1	**General Purpose Data Direction Mode 12.**
11	DM_11	RW	0x0	0-1	**General Purpose Data Direction Mode 11.**
10	DM_10	RW	0x0	0-1	**General Purpose Data Direction Mode 10.**
9	DM_9	RW	0x0	0-1	**General Purpose Data Direction Mode 9.**
8	DM_8	RW	0x0	0-1	**General Purpose Data Direction Mode 8.**
7	DM_7	RW	0x0	0-1	**General Purpose Data Direction Mode 7.**
6	DM_6	RW	0x0	0-1	**General Purpose Data Direction Mode 6.**
5	DM_5	RW	0x0	0-1	**General Purpose Data Direction Mode 5.**
4	DM_4	RW	0x0	0-1	**General Purpose Data Direction Mode 4.**
3	DM_3	RW	0x0	0-1	**General Purpose Data Direction Mode 3.**
2	DM_2	RW	0x0	0-1	**General Purpose Data Direction Mode 2.**
1	DM_1	RW	0x0	0-1	**General Purpose Data Direction Mode 1.**
0	DM_0	RW	0x0	0-1	**General Purpose Data Direction Mode 0.**

GPIO_OUTPUT_ENABLE

General Purpose IO Configuration Register:
If bit *x* is 1, enable the outpt driver on pin *x*. If bit *x* is 0, disable the output driver on pin *x*. Ignored if the pin is set to bypass mode.
x can be GPIO 0 to 15.

Address: 0xA08

Size: 32-bits

Table 10-90: GPIO_OUTPUT_ENABLE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	OE_15	RW	0x0	0-1	**General Purpose Data Output Enable 15.**
14	OE_14	RW	0x0	0-1	**General Purpose Data Output Enable 14.**
13	OE_13	RW	0x0	0-1	**General Purpose Data Output Enable 13.**
12	OE_12	RW	0x0	0-1	**General Purpose Data Output Enable 12.**
11	OE_11	RW	0x0	0-1	**General Purpose Data Output Enable 11.**
10	OE_10	RW	0x0	0-1	**General Purpose Data Output Enable 10.**
9	OE_9	RW	0x0	0-1	**General Purpose Data Output Enable 9.**
8	OE_8	RW	0x0	0-1	**General Purpose Data Output Enable 8.**
7	OE_7	RW	0x0	0-1	**General Purpose Data Output Enable 7.**
6	OE_6	RW	0x0	0-1	**General Purpose Data Output Enable 6.**
5	OE_5	RW	0x0	0-1	**General Purpose Data Output Enable 5.**
4	OE_4	RW	0x0	0-1	**General Purpose Data Output Enable 4.**
3	OE_3	RW	0x0	0-1	**General Purpose Data Output Enable 3.**
2	OE_2	RW	0x0	0-1	**General Purpose Data Output Enable 2.**
1	OE_1	RW	0x0	0-1	**General Purpose Data Output Enable 1.**
0	OE_0	RW	0x0	0-1	**General Purpose Data Output Enable 0.**

GPIO_OUTPUT_VALUE

General Purpose IO Data Register:
This register contains the value to be driven out of the pins. The output will only appear at the port if the pin has GPIO bypass mode disabled, set to output direction mode, and output driver is enabled.

****WARNING: Trying to change the state of the GPIO pins using this register will not generate an interrupt message. You have to use an external pin signal to change the state of a GPIO pin to generate an interrupt message.****

Address: 0xA0C

Size: 32-bits

Table 10-91: GPIO_OUTPUT_VALUE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	DATA_OUT_15	RW	0x0	0-1	**General Purpose Data In 15.**
14	DATA_OUT_14	RW	0x0	0-1	**General Purpose Data In 14.**
13	DATA_OUT_13	RW	0x0	0-1	**General Purpose Data In 13.**
12	DATA_OUT_12	RW	0x0	0-1	**General Purpose Data In 12.**
11	DATA_OUT_11	RW	0x0	0-1	**General Purpose Data In 11.**
10	DATA_OUT_10	RW	0x0	0-1	**General Purpose Data In 10.**
9	DATA_OUT_9	RW	0x0	0-1	**General Purpose Data In 9.**
8	DATA_OUT_8	RW	0x0	0-1	**General Purpose Data In 8.**
7	DATA_OUT_7	RW	0x0	0-1	**General Purpose Data In 7.**
6	DATA_OUT_6	RW	0x0	0-1	**General Purpose Data In 6.**
5	DATA_OUT_5	RW	0x0	0-1	**General Purpose Data In 5.**
4	DATA_OUT_4	RW	0x0	0-1	**General Purpose Data In 4.**
3	DATA_OUT_3	RW	0x0	0-1	**General Purpose Data In 3.**
2	DATA_OUT_2	RW	0x0	0-1	**General Purpose Data In 2.**
1	DATA_OUT_1	RW	0x0	0-1	**General Purpose Data In 1.**
0	DATA_OUT_0	RW	0x0	0-1	**General Purpose Data In 0.**

GPIO_INPUT_VALUE

General Purpose IO Data In Register:
The input value is read from this register, regardless of the pin mode.

Address: 0xA10

Size: 32-bits

Table 10-92: GPIO_INPUT_VALUE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	DATA_IN_15	RO	0x0	0-1	**General Purpose Data In 15.**
14	DATA_IN_14	RO	0x0	0-1	**General Purpose Data In 14.**
13	DATA_IN_13	RO	0x0	0-1	**General Purpose Data In 13.**
12	DATA_IN_12	RO	0x0	0-1	**General Purpose Data In 12.**
11	DATA_IN_11	RO	0x0	0-1	**General Purpose Data In 11.**
10	DATA_IN_10	RO	0x0	0-1	**General Purpose Data In 10.**
9	DATA_IN_9	RO	0x0	0-1	**General Purpose Data In 9.**
8	DATA_IN_8	RO	0x0	0-1	**General Purpose Data In 8.**
7	DATA_IN_7	RO	0x0	0-1	**General Purpose Data In 7.**
6	DATA_IN_6	RO	0x0	0-1	**General Purpose Data In 6.**
5	DATA_IN_5	RO	0x0	0-1	**General Purpose Data In 5.**
4	DATA_IN_4	RO	0x0	0-1	**General Purpose Data In 4.**
3	DATA_IN_3	RO	0x0	0-1	**General Purpose Data In 3.**
2	DATA_IN_2	RO	0x0	0-1	**General Purpose Data In 2.**
1	DATA_IN_1	RO	0x0	0-1	**General Purpose Data In 1.**
0	DATA_IN_0	RO	0x0	0-1	**General Purpose Data In 0.**

GPIO_INT_MASK

This register is used to mask interrupt events being signalled by GPIO_INT. The register is an active high mask. That is, an interrupt source will be disabled when the corresponding INT_MASK_x bit is '1'. The value of GPIO_INT_MASK is modified by the GPIO_INT_MASK_SET and GPIO_INT_MASK_CLR registers.

Address: 0xA14

Size: 32-bits

Table 10-93: GPIO_INT_MASK

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	INT_MASK_15	RO	0x0	0-1	**General Purpose IO Interrupt Mask 15**: When set to '1', the interrupt source 15 is disabled (masked off). When cleared to '0', the interrupt source 15 will be forwarded to the main interrupt controller.
14	INT_MASK_14	RO	0x0	0-1	**General Purpose IO Interrupt Mask 14.**
13	INT_MASK_13	RO	0x0	0-1	**General Purpose IO Interrupt Mask 13.**
12	INT_MASK_12	RO	0x0	0-1	**General Purpose IO Interrupt Mask 12.**
11	INT_MASK_11	RO	0x0	0-1	**General Purpose IO Interrupt Mask 11.**
10	INT_MASK_10	RO	0x0	0-1	**General Purpose IO Interrupt Mask 10.**
9	INT_MASK_9	RO	0x0	0-1	**General Purpose IO Interrupt Mask 9.**
8	INT_MASK_8	RO	0x0	0-1	**General Purpose IO Interrupt Mask 8.**
7	INT_MASK_7	RO	0x0	0-1	**General Purpose IO Interrupt Mask 7.**
6	INT_MASK_6	RO	0x0	0-1	**General Purpose IO Interrupt Mask 6.**
5	INT_MASK_5	RO	0x0	0-1	**General Purpose IO Interrupt Mask 5.**
4	INT_MASK_4	RO	0x0	0-1	**General Purpose IO Interrupt Mask 4.**
3	INT_MASK_3	RO	0x0	0-1	**General Purpose IO Interrupt Mask 3.**
2	INT_MASK_2	RO	0x0	0-1	**General Purpose IO Interrupt Mask 2.**
1	INT_MASK_1	RO	0x0	0-1	**General Purpose IO Interrupt Mask 1.**
0	INT_MASK_0	RO	0x0	0-1	**General Purpose IO Interrupt Mask 0.**

GPIO_INT_MASK_CLR (Note: formerly GPIO_INT_ENABLE)

This register is written in order to clear bits in the GPIO_INT_MASK register and thus enable interrupts. Each MASK_CLR_x bit written with '1' will cause each of the corresponding bits of GPIO_INT_MASK_x to be cleared.[1]

Address: 0xA18

Size: 32-bits

Table 10-94: GPIO_INT_MASK_CLR

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	MASK_CLR_15	RW	0x0	0-1	**Clear GPIO Interrupt Mask 15:** When this bit is written to '1', then GPIO_INT_MASK.INT_MASK_15 is cleared to '0'. When MASK_CLR_15 is written with '0', it has no effect on the state of GPIO_INT_MASK or any other function of the GN412x. The value of this bit is indeterminate when read back.
14	MASK_CLR_14	RW	0x0	0-1	**Clear GPIO Interrupt Mask 14.**
13	MASK_CLR_13	RW	0x0	0-1	**Clear GPIO Interrupt Mask 13.**
12	MASK_CLR_12	RW	0x0	0-1	**Clear GPIO Interrupt Mask 12.**
11	MASK_CLR_11	RW	0x0	0-1	**Clear GPIO Interrupt Mask 11.**
10	MASK_CLR_10	RW	0x0	0-1	**Clear GPIO Interrupt Mask 10.**
9	MASK_CLR_9	RW	0x0	0-1	**Clear GPIO Interrupt Mask 9.**
8	MASK_CLR_8	RW	0x0	0-1	**Clear GPIO Interrupt Mask 8.**
7	MASK_CLR_7	RW	0x0	0-1	**Clear GPIO Interrupt Mask 7.**
6	MASK_CLR_6	RW	0x0	0-1	**Clear GPIO Interrupt Mask 6.**
5	MASK_CLR_5	RW	0x0	0-1	**Clear GPIO Interrupt Mask 5.**
4	MASK_CLR_4	RW	0x0	0-1	**Clear GPIO Interrupt Mask 4.**
3	MASK_CLR_3	RW	0x0	0-1	**Clear GPIO Interrupt Mask 3.**
2	MASK_CLR_2	RW	0x0	0-1	**Clear GPIO Interrupt Mask 2.**
1	MASK_CLR_1	RW	0x0	0-1	**Clear GPIO Interrupt Mask 1.**
0	MASK_CLR_0	RW	0x0	0-1	**Clear GPIO Interrupt Mask 0.**

1. Rather than directly writing the value of the GPIO_INT_MASK, a "set"/"clear" mechanism is used so that multiple software/driver process threads can manipulate the mask without the vulnerabilities of a read-modify-write approach that creates software race conditions.

GPIO_INT_MASK_SET (Note: formerly GPIO_INT_DISABLE)

This register is written in order to set bits in the GPIO_INT_MASK register, and thus disable interrrupts. Each MASK_SET_X bit written with '1' will cause each of the corresponding bits of the GPIO_INT_MASK register to be set.

Address: 0xA1C

Size: 32-bits

Table 10-95: GPIO_INT_MASK_SET

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	MASK_SET_15	RW	0x0	0-1	**Set GPIO Interrupt Mask 15:** When this bit is written to '1', then GPIO_INT_MASK.INT_MASK_15 is set to '1'. When MASK_SET_15 is written with '0', it has no effect on the state of GPIO_INT_MASK or any other function of the GN412x. The value of this bit is indeterminate when read back.
14	MASK_SET_14	RW	0x0	0-1	**Set GPIO Interrupt Mask 14.**
13	MASK_SET_13	RW	0x0	0-1	**Set GPIO Interrupt Mask 13.**
12	MASK_SET_12	RW	0x0	0-1	**Set GPIO Interrupt Mask 12.**
11	MASK_SET_11	RW	0x0	0-1	**Set GPIO Interrupt Mask 11.**
10	MASK_SET_10	RW	0x0	0-1	**Set GPIO Interrupt Mask 10.**
9	MASK_SET_9	RW	0x0	0-1	**Set GPIO Interrupt Mask 9.**
8	MASK_SET_8	RW	0x0	0-1	**Set GPIO Interrupt Mask 8.**
7	MASK_SET_7	RW	0x0	0-1	**Set GPIO Interrupt Mask 7.**
6	MASK_SET_6	RW	0x0	0-1	**Set GPIO Interrupt Mask 6.**
5	MASK_SET_5	RW	0x0	0-1	**Set GPIO Interrupt Mask 5.**
4	MASK_SET_4	RW	0x0	0-1	**Set GPIO Interrupt Mask 4.**
3	MASK_SET_3	RW	0x0	0-1	**Set GPIO Interrupt Mask 3.**
2	MASK_SET_2	RW	0x0	0-1	**Set GPIO Interrupt Mask 2.**
1	MASK_SET_1	RW	0x0	0-1	**Set GPIO Interrupt Mask 1.**
0	MASK_SET_0	RW	0x0	0-1	**Set GPIO Interrupt Mask 0.**

GPIO_INT_STATUS

If bit *x* is 1, an interrupt-generating event has occurred on bit *x* of GPIO_INT_VALUE (GPIO_INT_STATUS is set regardless of GPIO_INT_MASK). This register is cleared on reading.
x can be GPIO 0 to 15.

Address: 0xA20

Size: 32-bits

Table 10-96: GPIO_INT_STATUS

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	INT_STATUS_15	RW	0x0	0-1	**General Purpose IO Interrupt Status 15:** Reads back as '1' to indicate that an interrupt event has occurred for GPIO bit 15. Cleared whenever the register is read.
14	INT_STATUS_14	RW	0x0	0-1	**General Purpose IO Interrupt Status 14.**
13	INT_STATUS_13	RW	0x0	0-1	**General Purpose IO Interrupt Status 13.**
12	INT_STATUS_12	RW	0x0	0-1	**General Purpose IO Interrupt Status 12.**
11	INT_STATUS_11	RW	0x0	0-1	**General Purpose IO Interrupt Status 11.**
10	INT_STATUS_10	RW	0x0	0-1	**General Purpose IO Interrupt Status 10.**
9	INT_STATUS_9	RW	0x0	0-1	**General Purpose IO Interrupt Status 9.**
8	INT_STATUS_8	RW	0x0	0-1	**General Purpose IO Interrupt Status 8.**
7	INT_STATUS_7	RW	0x0	0-1	**General Purpose IO Interrupt Status 7.**
6	INT_STATUS_6	RW	0x0	0-1	**General Purpose IO Interrupt Status 6.**
5	INT_STATUS_5	RW	0x0	0-1	**General Purpose IO Interrupt Status 5.**
4	INT_STATUS_4	RW	0x0	0-1	**General Purpose IO Interrupt Status 4.**
3	INT_STATUS_3	RW	0x0	0-1	**General Purpose IO Interrupt Status 3.**
2	INT_STATUS_2	RW	0x0	0-1	**General Purpose IO Interrupt Status 2.**
1	INT_STATUS_1	RW	0x0	0-1	**General Purpose IO Interrupt Status 1.**
0	INT_STATUS_0	RW	0x0	0-1	**General Purpose IO Interrupt Status 0.**

GPIO_INT_TYPE

If bit *x* is 1, interrupt is level triggered. If bit *x* is 0, interrupt is edge triggered.
x can be GPIO 0 to 15.

Address: 0xA24

Size: 32-bits

Table 10-97: GPIO_INT_TYPE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	INT_TYPE_15	RW	0x0	0-1	**General Purpose IO Interrupt Type 15.**
14	INT_TYPE_14	RW	0x0	0-1	**General Purpose IO Interrupt Type 14.**
13	INT_TYPE_13	RW	0x0	0-1	**General Purpose IO Interrupt Type 13.**
12	INT_TYPE_12	RW	0x0	0-1	**General Purpose IO Interrupt Type 12.**
11	INT_TYPE_11	RW	0x0	0-1	**General Purpose IO Interrupt Type 11.**
10	INT_TYPE_10	RW	0x0	0-1	**General Purpose IO Interrupt Type 10.**
9	INT_TYPE_9	RW	0x0	0-1	**General Purpose IO Interrupt Type 9.**
8	INT_TYPE_8	RW	0x0	0-1	**General Purpose IO Interrupt Type 8.**
7	INT_TYPE_7	RW	0x0	0-1	**General Purpose IO Interrupt Type 7.**
6	INT_TYPE_6	RW	0x0	0-1	**General Purpose IO Interrupt Type 6.**
5	INT_TYPE_5	RW	0x0	0-1	**General Purpose IO Interrupt Type 5.**
4	INT_TYPE_4	RW	0x0	0-1	**General Purpose IO Interrupt Type 4.**
3	INT_TYPE_3	RW	0x0	0-1	**General Purpose IO Interrupt Type 3.**
2	INT_TYPE_2	RW	0x0	0-1	**General Purpose IO Interrupt Type 2.**
1	INT_TYPE_1	RW	0x0	0-1	**General Purpose IO Interrupt Type 1.**
0	INT_TYPE_0	RW	0x0	0-1	**General Purpose IO Interrupt Type 0.**

GPIO_INT_VALUE

GPIO interrupt polarity: If bit *x* is 1, interrupt is triggered on high level or rising edge, depending on GPIO_INT_TYPE.INT_TYPE_x value. If bit *x* is 0, interrupt is triggered on low level or falling edge, depending on `GPIO_INT_TYPE.INT_TYPE` value.

Address: 0xA28

Size: 32-bits

Table 10-98: GPIO_INT_VALUE

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	INT_VALUE_15	RW	0x0	0-1	**General Purpose IO Interrupt Value 15.**
14	INT_VALUE_14	RW	0x0	0-1	**General Purpose IO Interrupt Value 14.**
13	INT_VALUE_13	RW	0x0	0-1	**General Purpose IO Interrupt Value 13.**
12	INT_VALUE_12	RW	0x0	0-1	**General Purpose IO Interrupt Value 12.**
11	INT_VALUE_11	RW	0x0	0-1	**General Purpose IO Interrupt Value 11.**
10	INT_VALUE_10	RW	0x0	0-1	**General Purpose IO Interrupt Value 10.**
9	INT_VALUE_9	RW	0x0	0-1	**General Purpose IO Interrupt Value 9.**
8	INT_VALUE_8	RW	0x0	0-1	**General Purpose IO Interrupt Value 8.**
7	INT_VALUE_7	RW	0x0	0-1	**General Purpose IO Interrupt Value 7.**
6	INT_VALUE_6	RW	0x0	0-1	**General Purpose IO Interrupt Value 6.**
5	INT_VALUE_5	RW	0x0	0-1	**General Purpose IO Interrupt Value 5.**
4	INT_VALUE_4	RW	0x0	0-1	**General Purpose IO Interrupt Value 4.**
3	INT_VALUE_3	RW	0x0	0-1	**General Purpose IO Interrupt Value 3.**
2	INT_VALUE_2	RW	0x0	0-1	**General Purpose IO Interrupt Value 2.**
1	INT_VALUE_1	RW	0x0	0-1	**General Purpose IO Interrupt Value 1.**
0	INT_VALUE_0	RW	0x0	0-1	**General Purpose IO Interrupt Value 0.**

GPIO_INT_ON_ANY

If bit *x* is 1, then edge triggering occurs on any edge, otherwise edge specified in GPIO_INT_VALUE.INT_VALUE_x triggers an interrupt (INT_ON_ANY_x is ignored if GPIO_INT_TYPE.INT_TYPE_x=1).

Address: 0xA2C

Size: 32-bits

Table 10-99: GPIO_INT_ON_ANY

Bits	Mnemonic	Type	Reset	Valid Range	Description
31:16	RESERVED	RO	0x0		Reserved.
15	INT_ON_ANY_15	RW	0x0	0-1	**General Purpose IO Interrupt On Any 15.**
14	INT_ON_ANY_14	RW	0x0	0-1	**General Purpose IO Interrupt On Any 14.**
13	INT_ON_ANY_13	RW	0x0	0-1	**General Purpose IO Interrupt On Any 13.**
12	INT_ON_ANY_12	RW	0x0	0-1	**General Purpose IO Interrupt On Any 12.**
11	INT_ON_ANY_11	RW	0x0	0-1	**General Purpose IO Interrupt On Any 11.**
10	INT_ON_ANY_10	RW	0x0	0-1	**General Purpose IO Interrupt On Any 10.**
9	INT_ON_ANY_9	RW	0x0	0-1	**General Purpose IO Interrupt On Any 9.**
8	INT_ON_ANY_8	RW	0x0	0-1	**General Purpose IO Interrupt On Any 8.**
7	INT_ON_ANY_7	RW	0x0	0-1	**General Purpose IO Interrupt On Any 7.**
6	INT_ON_ANY_6	RW	0x0	0-1	**General Purpose IO Interrupt On Any 6.**
5	INT_ON_ANY_5	RW	0x0	0-1	**General Purpose IO Interrupt On Any 5.**
4	INT_ON_ANY_4	RW	0x0	0-1	**General Purpose IO Interrupt On Any 4.**
3	INT_ON_ANY_3	RW	0x0	0-1	**General Purpose IO Interrupt On Any 3.**
2	INT_ON_ANY_2	RW	0x0	0-1	**General Purpose IO Interrupt On Any 2.**
1	INT_ON_ANY_1	RW	0x0	0-1	**General Purpose IO Interrupt On Any 1.**
0	INT_ON_ANY_0	RW	0x0	0-1	**General Purpose IO Interrupt On Any 0.**

10.4.10 FPGA Configuration Loader Registers

The FPGA Configuration Loader (FCL) services Serial Programming Interface (SPRI).

FCL_CTRL

FPGA Configuration Loader Control Register

Address: 0xB00

Size: 16-bits

Table 10-100: FCL_CTRL

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:9	RESERVED	RO	0x0		Reserved.
8	SPRI_CLK_STOP_EN	RW	0x0	0-1	(revision 2 silicon only) Enables stopping of the SPRI_CLK, when the FCL FIFO is almost empty, to prevent underflow situation. It is always set to "1" when FCL_CTRL[2:1] is non-zero value.
7	DATA_PUSH_COMP	RW	0x0		(revision 2 silicon only) When FCL_CTRL[8] is set to "1", this bit must also be set to "1" when all data has been written into the FCL FIFO via FCL_FIFO_DATA register.
6	RESET	RW	0x0	0-1	This synchronous reset must only be applied when the `fcl_clk_div` register is set to '0' or some registers that run off the divided clock will not be reset. This prevents locking out the software during a reset.
5:4	LAST_BYTE_CNT	RW	0x0	0-3	Indicates the number of bytes from the last word popped from the FIFO that are to be sent to the FPGA on spri_data: 00 4 bytes 01 3 bytes 10 2 bytes 11 1 bytes
3	SEND_CFG_DATA	RW	0x0	0-1	Send configuration data and clock during GPIO configuration.
2	FSM_EN	RW	0x0	0-1	**Enable FSM Configuration Mode**: Use FSM to configure FPGA, instead of GPIO registers.
1	SPRI_EN	RW	0x0	0-1	**Enable FPGA Configuration Mode**: Configures GPIO pins for configuration operation.
0	START_FSM	RW	0x0	0-1	**Starts FSM Configuration**: Automatically reset after FSM begin configuration.

FCL_STATUS

FPGA Configuration Loader Status Register

Address: 0xB04

Size: 16-bits

Table 10-101: FCL_STATUS

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	RO	0x0		Reserved.
5	SPRI_STATUS	RO	0x0	0-1	**State of SPRI_STATUS input.**
4	SPRI_XI_SWAP	RO	0x0	0-1	**State of SPRI_XI_SWAP input.**
3	SPRI_DONE	RO	0x0	0-1	**State of SPRI_DONE input.**
2	SPRI_CONFIG	RO	0x0	0-1	**State of SPRI_CONFIG output.**
1	SPRI_DATA	RO	0x0	0-1	**State of SPRI_DATA output.**
0	SPRI_CLOCK	RO	0x0	0-1	**State of SPRI_CLOCK output.**

FCL_IODATA_IN

FPGA Configuration Loader Data In Register

Address: 0xB08

Size: 16-bits

Table 10-102: FCL_IODATA_IN

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	RO	0x0		Reserved.
5:0	GPIO_DATA_IN	RO	0x0		**GPIO Input from the following SPRI signals**: bit[0] SPRI_CLKOUT bit[1] SPRI_DATAOUT bit[2] SPRI_CONFIG bit[3] SPRI_DONE (when FCL_CTRL.SPRI_EN = 0) bit[4] SPRI_XI_SWAP bit[5] SPRI_STATUS (when FCL_CTRL.SPRI_EN = 0)

FCL_IODATA_OUT

FPGA Configuration Loader Data Out Register

Address: 0xB0C

Size: 16-bits

Table 10-103: FCL_IODATA_OUT

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	RO	0x0		Reserved.
5:0	GPIO_DATA_OUT	RW	0x4		**GPIO Output of the following SPRI signals**: bit[0] SPRI_CLKOUT bit[1] SPRI_DATAOUT bit[2] SPRI_CONFIG bit[3] SPRI_DONE (when FCL_CTRL.SPRI_EN = 0) bit[4] SPRI_XI_SWAP bit[5] SPRI_STATUS (when FCL_CTRL.SPRI_EN = 0)

FCL_EN

FPGA Configuration Loader Enable Register

Address: 0xB10

Size: 16-bits

Table 10-104: FCL_EN

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	RO	0x0		Reserved.
5:0	OUTPUT_EN	RW	0x17		**GPIO Output Enable for the following SPRI signals**: bit[0] SPRI_CLKOUT bit[1] SPRI_DATAOUT bit[2] SPRI_CONFIG bit[3] SPRI_DONE bit[4] SPRI_XI_SWAP bit[5] SPRI_STATUS

FCL_TIMER_0

FPGA Configuration Loader Timer 0 Register

Address: 0xB14

Size: 16-bits

Table 10-105: FCL_TIMER_0

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	TIMER_0	RW	0x0		**Timer, Least Significant 16-bits**: Use in conjunction with FCL_TIMER_1 on page 161. Sets interrupt when timer reaches this value.

FCL_TIMER_1

FPGA Configuration Loader Timer 1 Register

Address: 0xB18

Size: 16-bits

Table 10-106: FCL_TIMER_1

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	TIMER_1	RW	0x0		**Timer, Most Significant 16-bits**: Use in conjunction with FCL_TIMER_0 on page 161.

FCL_CLK_DIV

FPGA Configuration Clock Divider Register

Address: 0xB1C

Size: 16-bits

Table 10-107: FCL_CLK_DIV

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:5	RESERVED	RO	0x0		Reserved.
4:0	CLK_DIV	RW	0x0		**Clock Divider Value**: Divides PCLK, which is always 125MHz. Valid range has a maximum of 16. The output FCL clock value would be: 0 PCLK / 2 1 PCLK / 4 2 PCLK / 8 3 PCLK / 16 ..... 15 PCLK / 65536

FCL_IRQ

FPGA Configuration Loader Interrupt Request Register. This register is cleared on read.

Address: 0xB20

Size: 16-bits

Table 10-108: FCL_IRQ

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	RO	0x0		Reserved.
5	FIFO_HALFFULL	RO	0x0	0-1	**Configuration FIFO Half Full Indicator**.
4	FIFO_UNDRFL	RO	0x0	0-1	**Configuration FIFO Underflow**.
3	CONFIG_DONE	RO	0x0	0-1	**Configuration Done**: Valid in FSM and GPIO configuration modes only.
2	CONFIG_ERROR	RO	0x0	0-1	**Configuration Error**: Reset on write. Valid in FSM and GPIO configuration modes only.
1	TIMER	RO	0x0	0-1	**Timer End Count Reached**: Reset on write.
0	SPRI_STATUS	RO	0x0	0-1	**SPRI_STATUS Received**: Indicates that configuration data and clock can be applied. Valid in GPIO configuration mode only.

FCL_TIMER_CTRL

FPGA Configuration Loader Timer Control Register

Address: 0xB24

Size: 16-bits

Table 10-109: FCL_TIMER_CTRL

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:2	RESERVED	RO	0x0		Reserved.
1	RESET	RW	0x0	0-1	**Timer Reset**.
0	GO	RW	0x0	0-1	**Timer Go**: Starts the counter.

FCL_IM

FPGA Configuration Loader Interrupt Mask Register

Address: 0xB28

Size: 16-bits

Table 10-110: FCL_IM

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:6	RESERVED	RO	0x0		Reserved.
5	FIFO_HALFFULL	RW	0x0	0-1	When set to '1', enables configuration FIFO half full interrupt (FCL_IRQ.FIFO_HALFFULL).
4	FIFO_UNDRFL	RW	0x0	0-1	When set to '1', enables configuration FIFO underflow interrupt (FCL_IRQ.FIFO_UNDRFL).
3	CONFIG_DONE	RW	0x0	0-1	When set to '1', enables Configuration Done interrupt (FCL_IRQ.CONFIG_DONE).
2	CONFIG_ERROR	RW	0x0	0-1	When set to '1', enables Configuration error interrupt (FCL_IRQ.CONFIG_ERROR).
1	TIMER	RW	0x0	0-1	When set to '1', enables Timer end count reached interrupt (FCL_IRQ.TIMER).
0	SPRI_STATUS	RW	0x0	0-1	When set to '1', enables SPRI_STATUS received interrupt (FCL_IRQ.SPRI_STATUS).

FCL_TIMER2_0

FPGA Configuration Loader Generic Timer2 - Lower 16-bits

Address: 0xB2C

Size: 16-bits

Table 10-111: FCL_TIMER2_0

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	TIMER2_0	RW	0x0		Timer2 - Least significant 16-bits.

FCL_TIMER2_1

FPGA Configuration Loader Generic Timer2 - Upper 16-bits

Address: 0xB30

Size: 16-bits

Table 10-112: FCL_TIMER2_1

Bits	Mnemonic	Type	Reset	Valid Range	Description
15:0	TIMER2_1	RW	0x0		Timer2 - Most significant 16-bits.

FCL_FIFO_DATA

FPGA Configuration Loader Data FIFO Input Register

Address: 0xE00

Size: 32-bits

Table 10-113: FCL_FIFO_DATA

Bits	Mnemonic	Type	Reset	Valid Range	Description
32:0	FIFO_DATA	RW	0x0		32 bits of data are pushed into the FIFO.

Index

Numerics

2-wire access 82
2-wire interface 59
2-wire interface registers 139
2-wire serial controller 14

A

additional information, 2-wire interface 67
additional information, FPGA configuration loader 79
Altera programming 74
application layer
    PCI Express 14

B

BAR4 access, PCI Express 82
bridge operation 39

C

CEM signals, PCI Express 32
clock
    local 27
    PCI Express 27
    routing, PCIe reference 80
clock configuration 27
configuration
    clock 27
configuration access, PCI Express 82
configuration loader
    FPGA on-the-fly 13
configuration space 33
    type 0 PCI 34
conventions
    data sheet 16
    numbering 16, 84
    register 83
    register naming 17

D

data sheet conventions 16
data sheet usage 15
design rules 80
device application 80
device serial number capability 37
device serial number registers 116

F

FCL programming 75
features 11
flow control 39
FPGA configuration loader 70
FPGA configuration loader registers 158
FPGA on-the-fly configuration loader 13
FPGA operation 70

G

general purpose IO block 67
Gennum contact information 17
getting answers to PCI Express related questions 17
getting help from Gennum 17
GN412x signals 19
GN412x system registers 125
GPIO block 67
GPIO registers 146

H

header format 48
how to use this data sheet 15

I

I2C access 82
I^2C master / target block 57
I2C serial controller 14
initialization 20
    OS 26
    PCI Express configuration space 26
    reset 20
initialization from a 2-wire EEPROM 22
initialization using a local processor 25
initialization using the PCIe configuration space 26
initialization, selecting a mode 21
initializing the internal registers 21
internal register access modes 82
internal registers 82

interrupt control unit 57
interrupt controller 14
introduction 11

L

L2P DDR source synchronous signals 42
live on power-up 13
local bus clocking 42
local bus control signals 42
local bus interface 13, 40
local bus protocol 46
local bus signals 40
local clock 27
Local-to-PCIe
    signals for outbound dataflow 45

M

master mode 57
master mode, 2-wire interface 61
master transactions, PCI Express 54
message signalled interrupt registers 106
MSI capability 38

N

numbering conventions 16, 84

O

operation, interrupt control unit 58
OS initialization 26
overview 18
overview, interrupt control unit 57
overview, PPGA configuration loader 70

P

P2L DDR source synchronous signals 42
PCI Express application layer 14
PCI Express BAR4 access 82
PCI Express capability 36
PCI Express capability registers 109
PCI Express CEM signals 32
PCI Express clock 27
PCI Express configuration access 82
PCI Express configuration space initialization 26
PCI Express contact information 17
PCI Express link 31
PCI Express master transactions 54
PCI Express reference clock routing 80
PCI Express reference power supply distribution 80
PCI Express target transactions 51
PCI Express traffic routing 32
PCI Special Interest Group 17
PCI type 0 configuration header registers 89
PCIe-to-Local
    signals for inbound dataflow 43
peripherals 59
physical interface 31
power management capability 36
power management capability registers 102
power supply distribution
    PCI Express 80
power-up
    live 13
protocol
    local bus 46

R

register
    2-wire interface 139
    CDN_LOCK 138
    CLK_CSR 128
    device serial number 116
    DSN_CAP 116
    DSN_HIGH 117
    DSN_LOW 116
    FCL_CLK_DIV 162
    FCL_CTRL 158
    FCL_EN 160
    FCL_FIFO_DATA 164
    FCL_IM 163
    FCL_IODATA_IN 159
    FCL_IODATA_OUT 160
    FCL_IRQ 162
    FCL_STATUS 159
    FCL_TIMER_0 161
    FCL_TIMER_1 161

FCL_TIMER_CTRL 163
FCL_TIMER2_0 164
FCL_TIMER2_1 164
FPGA configuration loader 158
GN412x system 125
GPIO 146
GPIO_BYPASS_MODE 146
GPIO_DIRECTION_MODE 147
GPIO_INPUT_VALUE 150
GPIO_INT_DISABLE 153
GPIO_INT_ENABLE 152
GPIO_INT_MASK 151
GPIO_INT_ON_ANY 157
GPIO_INT_STATUS 154
GPIO_INT_TYPE 155
GPIO_INT_VALUE 156
GPIO_OUTPUT_ENABLE 148
GPIO_OUTPUT_VALUE 149
INT_CFG0-7 133
INT_CTRL 130
INT_STAT 131
LB_CTL 127
message signalled interrupt 106
MSI_ADDRESS_HIGH 107
MSI_ADDRESS_LOW 107
MSI_CAP_ID 106
MSI_CONTROL 106
MSI_DATA 108
MSI_NEXT_ID 106
PCI Express capability 109
PCI type 0 configuration header 89
PCI_BAR_CONFIG 128
PCI_BAR_HIGH 98
PCI_BAR0_HIGH 95
PCI_BAR0_LOW 94
PCI_BAR2_HIGH 97
PCI_BAR2_LOW 96
PCI_BAR4_LOW 97
PCI_BIST 93
PCI_CACHE 92
PCI_CAP 103
PCI_CAP_ID 102
PCI_CLASS_CODE 92
PCI_CMD 89
PCI_CSI 98
PCI_CSR 104
PCI_CSR_BSE 105
PCI_DEVICE 89
PCI_HEADER 93
PCI_INT_LINE 100
PCI_INT_PIN 100
PCI_LATENCY 93
PCI_MAX_LAT 101
PCI_MIN_GNT 101
PCI_NEXT_ID 102
PCI_REVISION 92
PCI_ROM_BASE 99
PCI_STAT 91
PCI_SUB_SYS 99
PCI_SUB_VENDOR 98
PCI_SYS_CFG_SYSTEM 125
PCI_TO_ACK_TIME 134
PCI_VENDOR 89
PCIE_CAP_ID 109
PCIE_CAPABILITY 109
PCIE_DCR 111
PCIE_DEVICE_CAP 110
PCIE_DSR 112
PCIE_LCR 114
PCIE_LINK_CAP 113
PCIE_LSR 115
PCIE_NEXT_ID 109
PEX_CDN_CFG1 134
PEX_CDN_CFG2 135
PEX_ERROR_STAT 132
PM_CAP_POINTER 100
PM_DATA 105
power management capability 102
RASER_CONTROL 136
RASER_TEST_CONTROL 135
TWI_ADDRESS 140
TWI_CTRL 139
TWI_DATA 141
TWI_IR_DIS 145

TWI_IR_EN 144
TWI_IR_MASK 144
TWI_IRT_STATUS 141
TWI_SLV_MON 143
TWI_STATUS 140
TWI_TO 144
TWI_TR_SIZE 143
VC_CAP 118
VC_PCR 119
VC_PORT_CAP_1 118
VC_PORT_CAP_2 119
VC_PSR 120
VC_RESOURCE_CAP0 120
VC_RESOURCE_CAP1 122
VC_RESOURCE_CR0 121
VC_RESOURCE_CR1 123
VC_RESOURCE_SR0 122
VC_RESOURCE_SR1 124
virtual channel capability 118
register bit types 83
register conventions 83
register descriptions 88
register map 84
register naming conventions 17
reserved register bits 84
reset initialization 20

S

selecting an initialization mode 21
signal naming conventions 42
signal polarity inversion 31
signals
GN412x 19
L2P DDR source synchronous 42
local bus control 42
P2L DDR source synchronous 42
signals for inbound dataflow (PCIe-to-Local) 43
signals for outbound dataflow (Local-to-PCIe) 45
slave mode, 2-wire interface 65

T

target mode 57
target transactions, PCI Express 51
traffic routing, PCI Express 32
type 0 PCI configuration space 34

V

virtual channel capability 37
virtual channel capability registers 118
virtual channel support 14

X

Xilinx programming 71

DOCUMENT IDENTIFICATION
FAMILY REFERENCE MANUAL

Information relating to this product and the application or design described herein is believed to be reliable, however such information is provided as a guide only and Gennum assumes no liability for any errors in this document, or for the application or design described herein. Gennum reserves the right to make changes to the product or this document at any time without notice.

CAUTION

ELECTROSTATIC SENSITIVE DEVICES

DO NOT OPEN PACKAGES OR HANDLE EXCEPT AT A STATIC-FREE WORKSTATION

GENNUM CORPORATE HEADQUARTERS
4281 Harvester Road, Burlington, Ontario L7L 5M4 Canada
Phone: +1 (905) 632-2996
Fax: +1 (905) 632-2055
E-mail: corporate@gennum.com
www.gennum.com

OTTAWA
232 Herzberg Road, Suite 101
Kanata, Ontario K2K 2A1
Canada
Phone: +1 (613) 270-0458
Fax: +1 (613) 270-0429

CALGARY
3553 - 31st St. N.W., Suite 210
Calgary, Alberta T2L 2K7
Canada
Phone: +1 (403) 284-2672

UNITED KINGDOM
North Building, Walden Court
Parsonage Lane,
Bishop's Stortford Hertfordshire, CM23 5DB
United Kingdom
Phone: +44 1279 714170
Fax: +44 1279 714171

INDIA
#208(A), Nirmala Plaza,
Airport Road, Forest Park Square
Bhubaneswar 751009
India
Phone: +91 (674) 653-4815
Fax: +91 (674) 259-5733

SNOWBUSH IP - A DIVISION OF GENNUM
439 University Ave. Suite 1700
Toronto, Ontario M5G 1Y8
Canada
Phone: +1 (416) 925-5643
Fax: +1 (416) 925-0581
E-mail: sales@snowbush.com
Web Site: http://www.snowbush.com

MEXICO
288-A Paseo de Maravillas
Jesus Ma., Aguascalientes
Mexico 20900
Phone: +1 (416) 848-0328

JAPAN KK
Shinjuku Green Tower Building 27F
6-14-1, Nishi Shinjuku
Shinjuku-ku, Tokyo, 160-0023
Japan
Phone: +81 (03) 3349-5501
Fax: +81 (03) 3349-5505
E-mail: gennum-japan@gennum.com
Web Site: http://www.gennum.co.jp

TAIWAN
6F-4, No.51, Sec.2, Keelung Rd.
Sinyi District, Taipei City 11502
Taiwan R.O.C.
Phone: (886) 2-8732-8879
Fax: (886) 2-8732-8870
E-mail: gennum-taiwan@gennum.com

GERMANY
Hainbuchenstraße 2
80935 Muenchen (Munich), Germany
Phone: +49-89-35831696
Fax: +49-89-35804653
E-mail: gennum-germany@gennum.com

NORTH AMERICA WESTERN REGION
Bayshore Plaza
2107 N 1st Street, Suite #300
San Jose, CA 95131
United States
Phone: +1 (408) 392-9454
Fax: +1 (408) 392-9427
E-mail: naw_sales@gennum.com

NORTH AMERICA EASTERN REGION
4281 Harvester Road
Burlington, Ontario L7L 5M4
Canada
Phone: +1 (905) 632-2996
Fax: +1 (905) 632-2055
E-mail: nae_sales@gennum.com

KOREA
8F Jinnex Lakeview Bldg.
65-2, Bangidong, Songpagu
Seoul, Korea 138-828
Phone: +82-2-414-2991
Fax: +82-2-414-2998
E-mail: gennum-korea@gennum.com