Galileo Technology

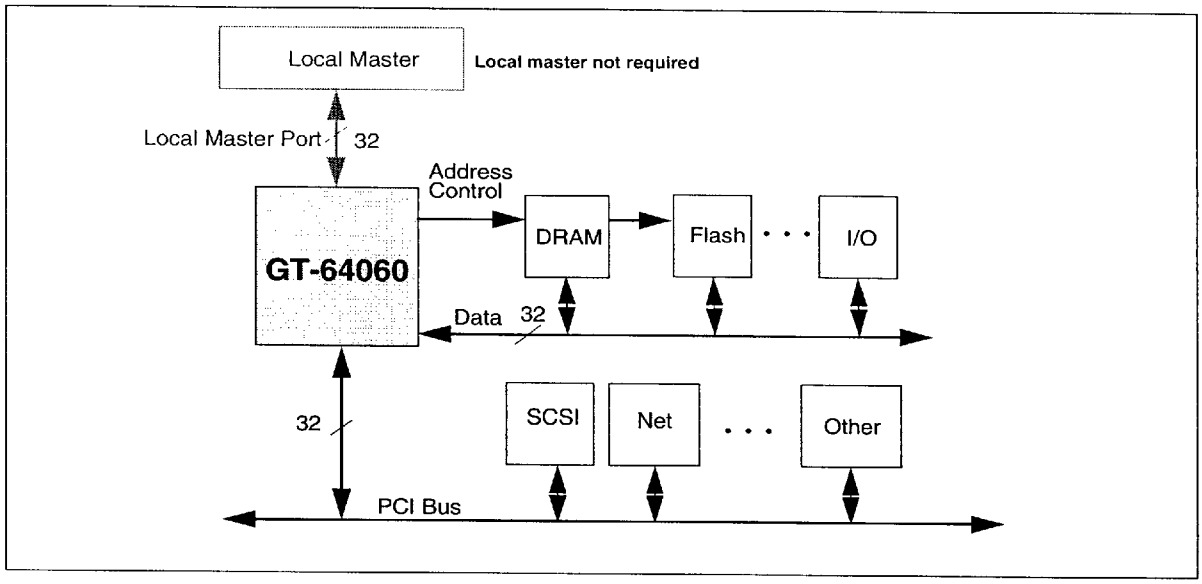
GT-64060
High-Integration PCI Bridge/ Memory Controller

Product Preview
Revision 0.7
4/1/97

Please contact Galileo Technology for possible updates before finalizing a design.

FEATURES

- High-integration PCI bridge/memory controller with three bus architecture
  - 32-bit Device/Memory Bus
  - 32-bit Multiplexed Master Port
  - 32-bit PCI interface
- Up to 50MHz master port and device bus frequency
  - Up to 150Mbytes/sec bandwidth
  - Asynchronous with respect to PCI bus
- 32-bit Master Port
  - Can access both PCI and device/memory bus
  - Can be interfaced to a wide array of ASICs, CPUs, and DSPs
- EDO/Fast Page Mode DRAM controller
  - 512MB address space
  - 256KB-16MB device depth
  - 1- 4 banks with 32-bit, or 64-bit interleaved width
  - Different size for each bank
- Device controller
  - 5 chip selects with programmable timing
  - Supports several types of standard memories (ROM / Flash / SRAM) and I/O controllers
  - Up to 160MB address space
  - External wait state support
  - 8-,16- and 32-bit width device support
  - 64-bit interleaving supported
  - External parity support
- Three 24-bit and one 32-bit timers/counters
- DMA controller
  - Four independent channels
  - Chaining via linked lists of records
  - Byte alignment on source and destination
  - Transfers through a 32-byte internal FIFO
  - Moves data between PCI, memory, and devices
- PCI bus
  - Fully compatible with PCI Revision 2.1
  - High performance through 96-bytes of posted write and read prefetch buffers
  - 32-bit PCI master and slave operations
  - Provides clock speed of up to 33MHz with no wait states on PCI
  - Supports burst operations on PCI for efficient data transfer
  - Supports doorbells between Host and PCI
  - Supports flexible byte swapping
  - Synchronization Barrier support from Master Port to PCI and from PCI to Master Port.
- Master Port to PCI bridge
  - Translates Master Port cycles into PCI I/O or Memory cycles
  - Generates Configuration, Interrupt Acknowledge and Special cycles on PCI
- PCI to Device/Memory bridge
  - Supports fast back-to-back transactions
  - Flexible address mapping of both DRAM and devices from PCI side
  - Supports memory and I/O transactions to internal configuration registers
  - Supports locked operations
- Plug and Play Support
  - PC compatible configuration registers
  - PCI configuration header can be loaded from boot PROM
  - PCI configuration registers are accessed from both Local Master Port and PCI side
- 5V Supply Voltage (PCI and Peripherals)
  - 3.3V to 5V Local Master Port interface
- 208 PQFP



www.galileoT.com info@galileoT.com Tel: (408)451-1400 Fax: (408)451-1404

NOTICE: This is a Product Preview Data Sheet

This document is a PRODUCT PREVIEW data sheet and describes a device not currently in production. The specifications within this document are incomplete and will change. DO NOT BASE A FINAL DESIGN on the information within this document. PRODUCT PREVIEW data sheets are intended to give the designer advance information about upcoming Galileo products, and are usually covered under Non-Disclosure Agreements.

1. OVERVIEW

The GT-64060 PCIntegrator™ provides a "silicon toolbox" for designers wishing to connect a wide array of devices and memories to the PCI bus. Applications include:

- PCI DRAM/SRAM/Flash controller
- CPU to memory/PCI bridge
- PCI to slave device bridge

The architecture of the GT-64060 supports several system implementations for different applications and cost/performance points. It is possible to design a powerful system with minimal glue logic, or add commodity logic (controlled by the GT-64060) for differentiated system architectures that attain higher performance.

The GT-64060 has a three bus architecture:

- A 32-bit interface to the Local Master Port (MasAD bus)
- A 32-bit interface to the memory and device subsystem.
- A 32-bit interface to the PCI bus.

The three buses are de-coupled from each other in most accesses, enabling concurrent operation of the Local Master Port, PCI devices, and accesses to memory. For example, the Local Master Port can write to the on-chip write buffer, a DMA agent can move data from DRAM to its own buffers, and a PCI device can write into an on-chip FIFO simultaneously.

1.1 Local Master Port Interface

The GT-64060 Local Master Port (LMP) allows local bus masters to access the PCI and memory/device buses. The LMP protocol supports byte and 32-bit word operations with burst lengths up to 8 words. With a maximum frequency of 50MHz, the Local Master can transfer in excess of 150Mbytes/sec.

1.2 DRAM and Device Interface

The GT-64060 has a flexible DRAM controller. It supports EDO as well as standard page mode DRAMs. With 60ns standard DRAMs, the GT-64060 can return data at 7-2-2-2-2-2-2-2[1] with 32-bit DRAMs and at 7-1-1-1-1-1-1-1 with 64-bit DRAMs to the Local Master Port (at 50Mhz local bus speed). The DRAM controller supports different depth devices in each bank for base configuration at manufacturing, and allowing for field upgrades by end users.

The GT-64060 memory controller supports different types of memory and I/O devices. It has the control signals and the timing programmability to support devices like Flash, EPROMs, SRAMs, FIFOs, and I/O controllers, from 8-bit to 64-bit width.

Parity generation and checking is supported externally and is optional for each bank of DRAM or any other device on the memory bus.

1.3 PCI Interface

The GT-64060 interfaces directly with the PCI bus. It can be either a master initiating a PCI bus operation, or a target responding to a PCI bus operation.

The GT-64060 incorporates 96-bytes of posted write and read prefetch buffers for efficient data transfer between the Local Master Port/DMA to PCI, and PCI to main memory.

The GT-64060 becomes a PCI bus master when the Local Master Port or the internal DMA engine initiates a bus cycle to a PCI device. It translates the Local Master Port cycle into the appropriate PCI bus cycle. These cycles can be either Memory, Interrupt Acknowledge, Special, I/O, or Configuration cycles.

The GT-64060 acts as a target when a PCI device initiates a memory access (or an I/O access in the case of internal registers). It responds to all memory read/write accesses, as well as to all configuration and I/O cycles in the case of internal registers.

1. Note that Galileo uses "total clock" nomenclature and not "wait-state" nomenclature. This means that 7-1-1-1... means 7 clocks to the firs

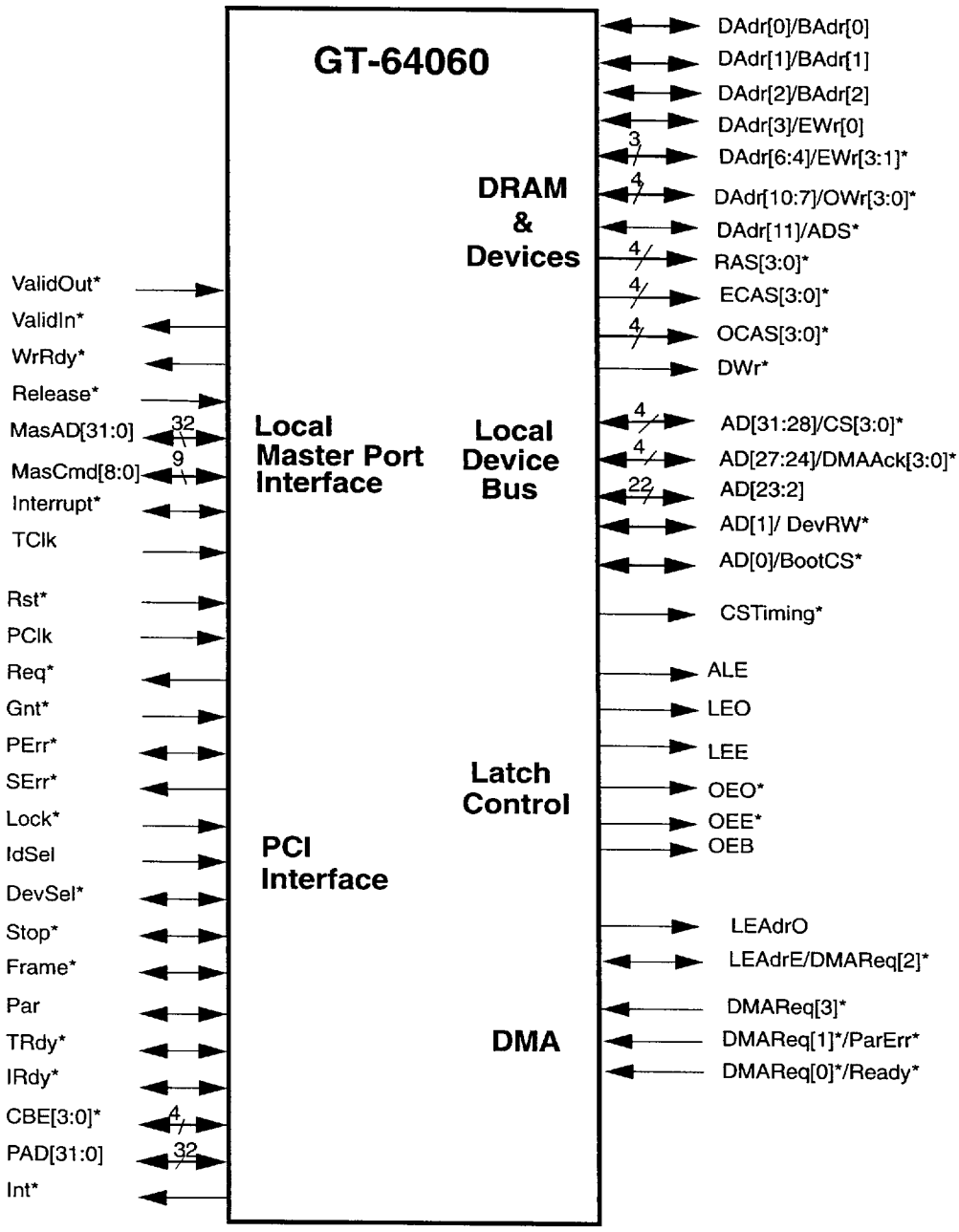
The GT-64060 contains the required PCI configuration registers. All the internal registers, including the PCI configuration registers, can be accessed from both the Local Master Port and the PCI bus. The GT-64060 configuration register set is fully PC Plug and Play compatible.

1.4 DMA Engines

The GT-64060 incorporates four high performance DMA engines. Each DMA engine has the capability to transfer data between PCI devices and main memory or between devices residing on the memory bus. The DMA uses an internal 32-byte FIFO for temporary storage of DMA data. Source and destination addresses can be non-aligned on any byte address boundary. The DMA channels can be programmed by the Local Master Port or by PCI masters, or without Local Master Port intervention via a linked list of records that is loaded by the DMA controller into the channel's working set when a DMA transaction ends. The DMA supports increment/decrement/hold on source and destination addresses independently.

2. PIN INFORMATION

2.1 Logic Symbol



2.2 Pin Assignment Table

Pin Name	Type	Description
Local Master Port Interface		
Release*	I	**Release Interface:** Signals to the GT-64060 that the local bus master has released the MasAD and MasCmd buses for completion of a master read request.
WrRdy*	O	**Write Ready:** The GT-64060 signals that it can accept a master port write request (i.e. there is room in the write posting FIFO.)
ValidIn*	O	**Valid Input:** The GT-64060 signals that it is driving valid data on the MasAD bus, and a valid data identifier on the MasCmd bus.
ValidOut*	I	**Valid Output:** Signals that the local master is driving valid address or data on the MasAD bus and a valid command or data identifier on the MasCmd bus.
MasAD[31:0]	I/O	**Master Port Address/Data Bus:** A 32-bit address and data bus for communication between the local master and GT-64060.
MasCmd[8:0]	I/O	**System Command/Data Identifier Bus:** A 9-bit bus for command and data identifier transmission between the local master and GT-64060.
Interrupt*	I/O	**Interrupt:** An "OR" of all the internal interrupt sources on the GT-64060. This pin is also sampled as an input at reset for configuration purposes.
TClk	I	**Clock:** The input clock to the GT-64060 (up to 50MHz). TClk is used for both the Local Master Port and Device interface. TClk must be driven for all applications, including those that do not use the Local Master Port.
Vref	I	**Voltage Reference:** This pin sets the voltage for logical high on the Local Master Port interface pins. For 3.3V masters this pin must be tied to a voltage divider between 3.3V and 5.0V (see text). 5V masters should tie this pin to 5V.
PCI Interface		
PClk	I	**PCI Clock:** It provides the timing for the PCI-related bus transaction. The PCI clock range is between 0 and 33MHz.
Rst*	I	**Reset:** Resets the GT-64060 to its initial state. This signal must be asserted for at least 10 PCI clock cycles. When in the reset state, all PCI output pins are put into tristate and all open drain signals are floated.
PAD[31:0]	I/O	**Address/Data:** 32-bit multiplexed PCI address and data lines. During the first clock of the transaction, PAD[31:0] contains a physical byte address (32 bits). During subsequent clock cycles, PAD[31:0] contains data.
CBE[3:0]*	I/O	**Bus Command/Byte Enable:** These are multiplexed on the same PCI pins. During the address phase of the transaction, CBE[3:0]* provide the bus command. During the data phase, these lines provide the byte enable. Byte enable determines which bytes carry valid data.

Pin Name	Type	Description
Par	I/O	**Parity:** Calculated by the GT-64060 as an even parity bit for the PAD[31:0] and CBE[3:0]* lines.
Frame*	I/O	**Frame:** It is asserted by the GT-64060 to indicate the beginning and duration of a master transaction. Frame* asserts to indicate the beginning of the cycle. While Frame* is asserted, data transfer continues. Frame* deasserts to indicate that the next data phase is the final data phase transaction. Frame* is monitored by the GT-64060 when it acts as a target.
IRdy*	I/O	**Initiator Ready:** It indicates the bus master's ability to complete the current data phase of the transaction. A data phase is completed on any clock when both TRdy* and IRdy* are asserted. Wait cycles are inserted until TRdy* and IRdy* are asserted together.
TRdy*	I/O	**Target Ready:** It indicates the target agent's ability to complete the current data phase of the transaction. A data phase is completed on any clock when both TRdy* and IRdy* are asserted. Wait cycles are inserted until TRdy* and IRdy* are asserted together.
Stop*	I/O	**Stop:** It indicates that the current target is requesting the bus master to stop the current transaction. As a master, the GT-64060 responds to the assertion of Stop* by disconnecting, retrying or aborting. As a target, the GT-64060 asserts Stop* to retry or disconnect.
Lock*	I	**Lock:** It indicates an atomic operation that may require multiple transactions to complete. When the GT-64060 is a PCI target, Lock* is sampled on the rising edge of the PClk when Frame* is asserted. If Lock* is sampled asserted, the GT-64060 enters into a locked state and remains in this state until Lock* is sampled deasserted on the following rising edge of PClk, when Frame* is sampled asserted.
IdSel	I	**Initialization Device Select:** It asserts to act as a chip select during PCI configuration read and write transactions.
DevSel*	I/O	**Device Select:** It is asserted by the target of the current access. When the GT-64060 is bus master, it expects the target to assert DevSel* within 5 bus cycles, confirming the access. If the target does not assert DevSel* within the required bus cycles, the GT-64060 aborts the cycle. As a target, when the GT-64060 recognizes its transaction, it asserts DevSel* in a medium speed (two cycles after the assertion of Frame*).
Req*	O	**Bus Request:** It is asserted by the GT-64060 to indicate to the bus arbiter that it desires use of the bus.
Gnt*	I	**Bus Grant:** It asserts to indicate to the GT-64060 that access to the bus is granted.
PErr*	I/O	**Parity Error:** It asserts when a data parity error is detected.
SErr*	O	**System Error:** It asserts when a serious system error (not necessarily a PCI error) is detected. The GT-64060 asserts the SErr* two cycles after the failing address.

Pin Name	Type	Description
Int*	O	**Interrupt Request:** It is asserted by the GT-64060 when one of the unmasked internal interrupt sources is asserted.
DRAM & Devices		
DWr*	O	**DRAM Write:** It is LOW when the GT-64060 writes to the DRAM.
DAdr[0]/BAdr[0]	O	**DRAM Address 0 / Burst Address 0:** This pin has two functions. In an access to a DRAM bank, this pin functions as a DRAM address bit. In write and read accesses from devices that are 8-bit wide, this pin functions as byte address 0 in the packing process of data into 64-bits. In accesses to a word wide (32-bit) device, this bit functions as address 0 in a burst access (equivalent to MasAD[2]). Not used for 16/64 bit devices.
DAdr[1]/BAdr[1]	O	**DRAM Address [1] / Burst Address [1]:** In DRAM accesses, this pin functions as an address bit. In read accesses to devices that are 8-, or 16-bit wide, BAdr[2:1] function as a half word address in the packing process of data into 64 bits. In accesses to a 32-bit bank, BAdr[2:1] function as part of the (two MSB) burst address bits of an address into an eight word line or when packing/unpacking a 64-bit access (equivalent to MasAD[4:3]). In accesses to a 64-bit bank, BAdr[2:1] function as the two burst address bits of a four double word line (equivalent to MasAD[4:3]).
DAdr[2]/BAdr[2]	O	**DRAM Address [2] / Burst Address [2]:** In DRAM accesses, this pin functions as an address bit. In read access to devices that are 8- or 16-bit wide, BAdr[2:1] function as a half word address in the packing process of data into 64 bits. In accesses to a 32-bit bank, BAdr[2:1] function as part of the (two MSB) burst address bits of an address into an eight word line or when packing/unpacking a 64-bit access (equivalent to MasAD[4:3]). In accesses to a 64-bit bank, BAdr[2:1] function as the two burst address bits of a four double word line (equivalent to MasAD[4:3]).
DAdr[3]/EWr[0]*	O	**DRAM Address [3] / Even Bank Byte Write [0]:** In DRAM accesses this pin functions as DRAM address. In device writes it functions as a byte write enable indication to the even bank byte 0.
DAdr[6:4]/ EWr[3:1]*	I/O	**DRAM Address [6:4] / Even Bank Byte Write [3:1]:** In DRAM accesses these pins function as DRAM address. In device writes, they function as byte write enable indications to the even bank bytes [3:1]. These pins are sampled as inputs at reset for configuration purposes.
DAdr[10:7]/ OWr[3:0]*	I/O	**DRAM Address [10:7] / Odd Bank Byte Write [3:0]:** In DRAM accesses these pins function as DRAM address. In device writes, they function as byte write enable indications to the odd bank bytes [3:0]. These pins are sampled as inputs at reset for configuration purposes.

Pin Name	Type	Description
DAdr[11]/ADS*	I/O	**DRAM Address [11] / Address Strobe:** In DRAM accesses this pin functions as a DRAM address. In device accesses it is active for one cycle when the address for the device is on the AD bus. Optionally, this pin is software configurable to only behave as ADS* via bit 17 of the DRAM Configuration register. This pin is sampled as an input at reset for configuration purposes.
RAS[3:0]*	O	**Row Address Select:** Supports four banks of DRAM. The DRAM banks can be 32-(36-) bit or 64-(72-) bit wide.
ECAS[3:0]*	O	**Even Column Address Select:** Supports byte writes/reads to the even bank of the DRAM (when interleaved.) If the bank is not interleaved, ECAS[3:0]* is the same as OCAS[3:0]*.
OCAS[3:0]*	O	**Odd Column Address Select:** Supports byte writes/reads to the odd bank of the DRAM (when interleaved.) If the bank is not interleaved, OCAS[3:0]* is the same as ECAS[3:0]*.
Local AD Bus		
AD[31:28]/CS[3:0]*	I/O	**Data [31:28] / Chip Select [3:0]:** In the data phase, the pins function as data bits [31:28]. In the address phase, Device Chip Selects are valid (and should be latched). The Chip Selects need to be qualified with the CSTiming* signal. Latching is done via ALE.
AD[27:24]/ DMAAck[3:0]*	I/O	**Data [27:24] / DMA Acknowledge[3:0]:** In the data phase, the pins function as data bits [27:24]. In the address phase, DMA Acknowledges are valid (and should be latched). They need to be qualified with the CSTiming* signal. Latching is done via ALE.
AD[23:2]	I/O	**Address/Data[23:2]:** Multiplexed address and data bus to the DRAM (data only) and the devices (address and data).
AD[1]/DevRW*	I/O	**Data [1] / Device Read-Write:** In the data phase it is data bit 1. In the address phase, it indicates if an access to a device is a read ('1') or a write ('0'). Latching is done via ALE.
AD[0]/BootCS*	I/O	**Data [0]/ Boot Chip Select:** In the data phase it is data bit 0. In the address phase, it is the boot device chip select. Latching is done via ALE.
CSTiming*	O	**Chip Select Timing:** Active for the number of cycles that the device that is currently being accessed was programmed to. Used to qualify the CS[3:0]*, BootCS and the DMAAck[3:0]* signals.
Latch Control		
ALE	O	**Address Latch Enable:** Used to latch the Address, BootCS*, CS[3:0]*, DevRW* and DMAAck[3:0]* from the AD bus.
LEO	O	**Latch Enable Odd:** Used to latch data to or from the odd bank devices.
LEE	O	**Latch Enable Even:** Used to latch data to or from the even bank devices.
OEO*	O	**Output Enable Odd:** Output data from the latch of the odd bank to the AD bus.
OEE*	O	**Output Enable Even:** Output data from the latch of the even bank to the AD bus.

Pin Name	Type	Description
OEB	O	**Output Enable Write:** Output data from the latch of the AD bus to the memory bus. This signal is only active during writes to DRAM or devices, and its polarity is programmable at reset.
LEAdrO	O	**Latch Enable Address Odd:** Used to latch the DRAM address and device burst address of the odd bank.
LEAdrE/ DMAReq[2]*	I/O	**Latch Enable Address Even / DMA Request:** Multiplexed signal that can be used to latch the DRAM address and device address of the even bank or, as a DMA request indication by an external device. Its function is designated at reset.
DMA		
DMAReq[3]*/Auto-Load*	I	**DMA Request[3]:** DMA request indication by an external device. This pin is sampled on Rst* to enable auto-load mode of PCI configuration registers. 0 - Auto-load mode Enabled 1 - Auto-load mode Disabled
DMAReq[1]*/ ParErr*	I	**DMA Request [1] / DMA Parity Error:** DMA request indication by an external device or parity error indication by external logic. The function of this pin is programmable at reset.
DMAReq[0]*/ Ready*	I	**DMA Request [0] / Ready:** This pin has two functions: it serves as a DMA request indication by an external device, or as a cycle extender (when inactive during a device access, an access will extend until Ready* is asserted). The function of this pin is programmable at reset.

3. FUNCTIONAL DESCRIPTION

3.1 Local Master Port Interface

The GT-64060 Local Master Port (LMP) allows a master device to gain access to the GT-64060's internal registers, PCI interface and the memory/device bus (AD bus). The LMP allows for burst accesses from one to 32 bytes in length.

Use of the LMP is optional as the GT-64060 can be programmed entirely from the PCI bus.

3.1.1 Local Master Port Signals

The LMP incorporates the following signals:

- MasAD[31:0] - Master Address/Data. This bus transfers multiplexed address/data.
- MasCmd[8:0] - Master Port Command. The MasCmd bus transfers information about the access (read/write, size) and the data (good/bad, last word.)
- ValidOut* - Indicates that the local master is driving valid address/data/command on the LMP.
- ValidIn* - Indicates that the GT-64060 is driving valid data/command on the LMP.
- WrReady* - Indicates that the GT-64060 is capable of accepting a write transaction up to 8 words in length.
- Release* - Indicates to the GT-64060 that the local master will not drive the LMP after the current clock cycle (i.e. the local master is floating the MasAD and MasCmd bus for completion of a read.)

The LMP bus is synchronous with respect to TClk and is locked with respect to the AD bus. The LMP may be asynchrnous with respect to the PCI bus or locked to the PCI bus for lower synchronization latency.

3.1.2 MasAD and MasCmd Buses

The MasAD[31:0] bus is a 32-bit multiplexed address/data bus. The local master drives address for a single cycle then either drives data (for a write) or floats the bus is anticipation of returned data (for a read.)

The MasCmd[8:0] bus conveys information about the transaction such as the direction (read/write), the size (byte, short, word, multi-word) and the status of the data (good/bad/last.) MasCmd is driven by the local master during the address phase of a transaction (with direction/size information) and for the duration of a write (with good/bad/last information.) The GT-64060 drives MasCmd during the data phase of read transactions.

The encodings for MasCmd[8:0] are shown in the tables below. Note that many encodings are not defined; these encodings are reserved and must not be used. A summary of bit usage is shown below.

TABLE 1. MasCmd Bit Summary

MasCmd Bit	Function
MasCmd[8]	0 = Transaction information (read/write/size) 1 = Data information (good/bad/last)
MasCmd[7]	Indicates last data/not last data during data cycles. Must be '0' for address cycles.
MasCmd[6]	0 = Read transaction (during address cycles) 1 = Write transaction (during address cycles) Must be '0' for data cycles.
MasCmd[5]	Indicates error status for data cycles. Must be '0' for address cycles.
MasCmd[4]	Reserved. Always '1'
MasCmd[3:0]	Encoded to indicate size of the transfer

TABLE 2. Address Phase Master Port MasCmd[8:0] Encodings (driven by local bus master)

MasCmd[8:0] Encoding[1]									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
0	0	0	0	1	1	0	0	0	RdByte	Read a single byte
0	0	0	0	1	1	0	0	1	RdShort	Read 16 bits
0	0	0	0	1	1	0	1	0	RdTriByte	Read 3 bytes
0	0	0	0	1	1	0	1	1	RdWord	Read 4 bytes (single word)
0	0	0	0	1	1	1	X	X	Rd2Words	Read 2 words (8 bytes) in a burst
0	0	0	0	1	0	X	X	0	Rd4Words	Read 4 words (16 bytes) in a burst
0	0	0	0	1	0	X	0	1	Rd8Words	Read 8 words (32 bytes) in a burst
0	0	1	0	1	1	0	0	0	WrByte	Write a single byte
0	0	1	0	1	1	0	0	1	WrShort	Write 16 bits
0	0	1	0	1	1	0	1	0	WrTriByte	Write 3 bytes
0	0	1	0	1	1	0	1	1	WrWord	Write 4 bytes (single word)
0	0	1	0	1	1	1	X	X	Wr2Words	Write 2 words (8 bytes) in a burst
0	0	1	0	1	0	X	X	0	Wr4Words	Write 4 words (16 bytes) in a burst
0	0	1	0	1	0	X	0	1	Wr8Words	Write 8 words (32 bytes) in a burst

1. 'X' denotes "don't care" but 'X' signals must be driven to a valid 0/1.

TABLE 3. Read Response Master Port MasCmd[8:0] Encodings (driven by GT-64060)

MasCmd[8:0] Encoding[1]									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
1	0	0	E	X	X	X	X	X	RD	Indicates valid data within a burst E = 0 Data is good E = 1 Data is erroneous
1	1	0	E	X	X	X	X	X	REOD	Indicates last valid data in a burst E = 0 Data is good E = 1 Data is erroneous

1. 'X' denotes "don't care" but 'X' signals are driven to a valid 0/1 by GT-64060.

TABLE 4. Master Port Write MasCmd[8:0] Encodings (driven by local master)

MasCmd[8:0] Encoding[1]									Command Mnemonic	Command Description
8	7	6	5	4	3	2	1	0		
1	0	0	E	X	X	X	X	X	WD	Indicates valid data within a burst E = 0 Data is good E = 1 Data is erroneous
1	1	0	E	X	X	X	X	X	WEOD	Indicates last valid data in a burst E = 0 Data is good E = 1 Data is erroneous

1. 'X' denotes "don't care" but 'X' signals are driven to a valid 0/1 by GT-64060.

3.1.3 LMP Read Protocol

LMP reads occur in three phases:

- The address phase in which address information is driven on the MasAD bus and command information is driven on MasCmd.
- The mid-burst data phase during which the GT-64060 drives data on the MasAD bus and a read response on MasCmd. The mid-burst data phase is entered between the address phase and the last-burst data phase.
- The last-burst data phase during which the GT-64060 drives data on the MasAD bus and a read end-of-data (REOD) response on MasCmd.

The address phase for all transactions begin with the assertion of ValidOut* to the GT-64060. Valid address and command information must be present on MasAD and MasCmd during this phase. Release* must also be asserted to the GT-64060 to indicate that the local master is releasing mastership of the MasAD/MasCmd buses to the GT-64060 for completion of the read. ValidOut* is deasserted at the end of the phase since the Local Master is no longer driving information on MasAD/MasCmd.

For transactions longer than 32 bits, the mid-burst data phase is entered next. The GT-64060 will drive valid data on MasAD, a valid read response (mnemonic = RD) on MasCmd, and will assert ValidIn* to qualify the MasAD and MasCmd buses (see Figure 1).

The GT-64060 transitions to the last-burst data phase on the last datum of the transfer. This state is differentiated by from the mid-burst state by the REOD command driven on the MasCmd bus. The last-burst data phase is also entered for the datum returned for a single word, or sub-word, read.

On the clock cycle following REOD, the GT-64060 floats the MasAD and MasCmd buses, returning ownership to the local master.

Figure 1: Single Word Read Through Master

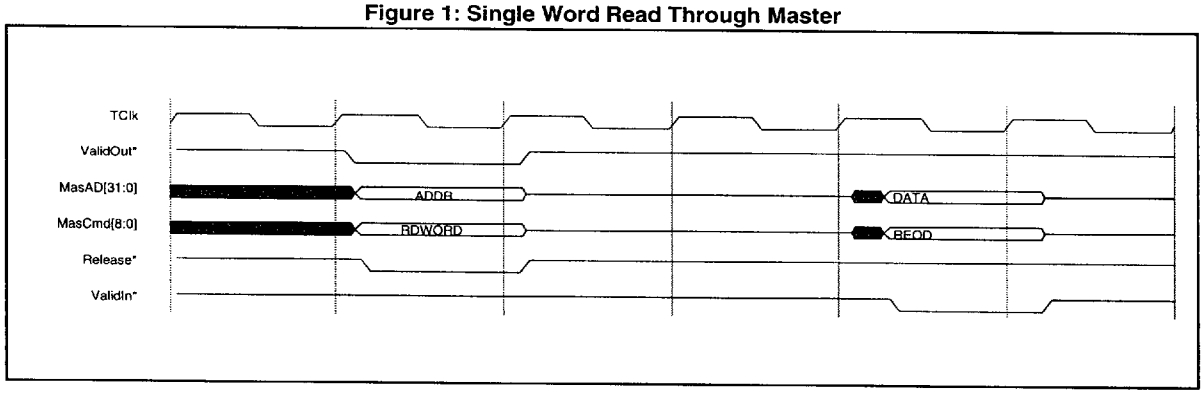
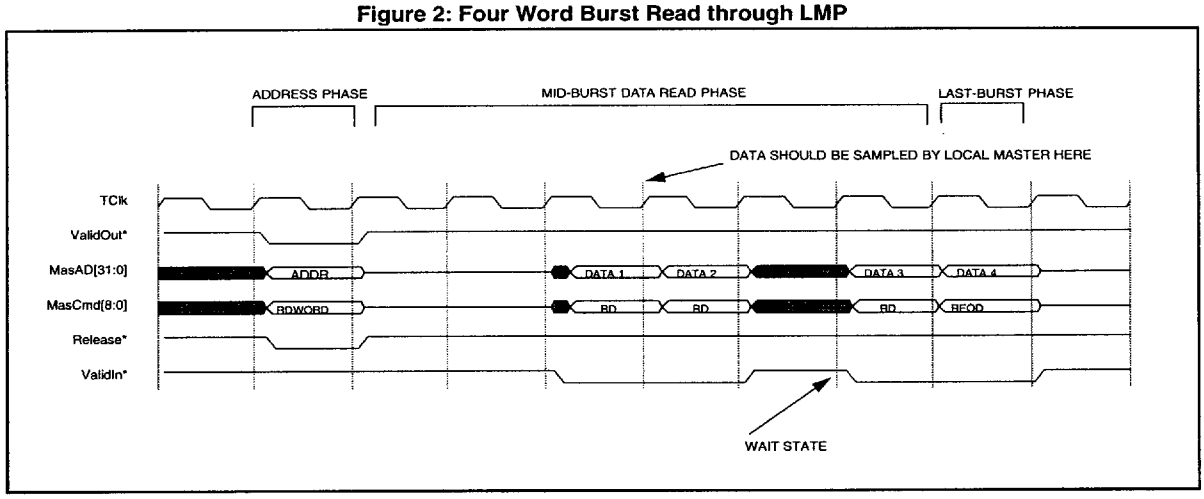


Figure 2: Four Word Burst Read through LMP



3.1.4 LMP Write Protocol

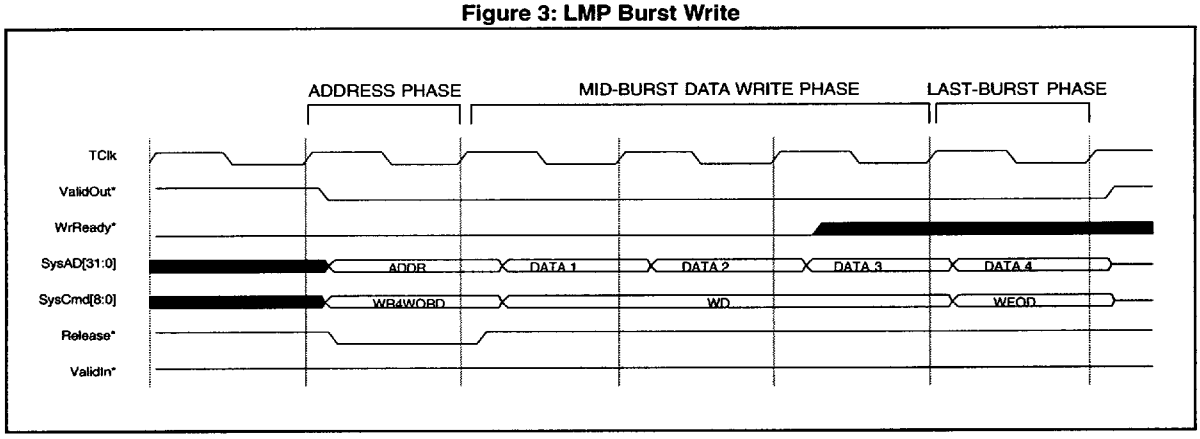
LMP writes occur in three phases:

- The address phase in which address information is driven on the MasAD bus and command information is driven on MasCmd.
- The mid-burst write data phase during which the Local Master drives data on the MasAD bus and a write command (mnemonic = WD) on MasCmd. The mid-burst write data phase is entered between the address phase and the last-burst write data phase.
- The last-burst write data phase during which the Local Master drives data on the MasAD bus and a write end-of-data (WEOD) command on MasCmd.

The address phase for write transactions begin with the assertion of ValidOut* to the GT-64060. Valid address and command information must be present on MasAD and MasCmd during this phase. Release* remains high for write transactions since the Local Master is not relinquishing ownership of the bus. ValidOut* is remains asserted throughout a write transaction as the Local Master always driving valid information on MasAD/MasCmd.

For transactions longer than 32 bits, the mid-burst data write phase is entered next. The Local Master drives valid data on MasAD, a valid write command (mnemonic = WD) on MasCmd (see Figure 3).

Figure 3: LMP Burst Write



The GT-64060 transitions to the last-burst write data phase on the last datum of the transfer. This state is differentiated by from the mid-burst state by the WEOD command driven on the MasCmd bus. The last-burst data phase is also entered for the datum written for a single word, or sub-word, write. On the clock cycle following WEOD, the GT-64060 returns to the idle state.

NOTE: *LMP writes cannot be issued as long as WrReady* is deasserted (HIGH). If WrReady* is high and an LMP write is attempted, data from previous write cycles may be corrupted (see section 3.1.5.)*

3.1.5 Operation of WrRdy* and the Internal Write Posting Queues

The GT-64060's Local Master Port includes a write posting queue that absorbs local port writes at zero wait-states. The write posting queue has 4 address entries and 16 32-bit data entries (see FigX.) The GT-64060 signals if there is "room" in the LMP write posting queue by asserting WrReady*. If WrReady* is asserted then the Local Master may issue a write of up to 8 words. The Local Master must not issue a write when WrReady* is deasserted (HIGH). Failure to follow this restriction will result in the loss of the data from previous write cycles still posted in the queue.

WrRdy* will be deasserted the cycle following the following:

- The address FIFO has two valid entries and a third address is being pushed, or...
- The address FIFO has more than two valid entries, or...
- The address FIFO has two valid entries or more, the data FIFO has four valid entries and a fifth one is being pushed, or...
- The address FIFO has two valid entries or more, the data FIFO has more than four valid entries, or...
- The address FIFO has one valid entry, the data FIFO has six valid entries and a seventh one is being pushed, or...
- The address FIFO has one valid entry, the data FIFO has more than six valid entries.

WrRdy* will be re-asserted the cycle following a transation away from the states above.

It is not necessary to take the above scenarios in to account when designing a system with the GT-64060. You simply need to sample the WrReady* pin before allowing a write to proceed. Some local master devices do not allow the address phase of a transfer to be extended. For such devices, you may need to disallow the local master bus ownership through bus hold or backoff mechanisms. Examples of how to do this can be found in the GT-64060 applications notes on our website.

3.1.6 LMP Data Alignment and Endianess

[next rev]

3.1.7 Burst Order

The GT-64060 supports only sub-block ordered bursts. Sub-block ordered bursts are optimized for the burst patterns used by most synchronous SRAMs and SDRAMs. Table 5 shows the addressing order for bursts.

TABLE 5. LMP Burst Order

Starting Offset	Burst Address Sequence
0x0	0x0, 0x4, 0x8, 0xC, 0x10, 0x14, 0x18, 0x1C
0x4	
0x8	
0xC	
0x10	
0x14	

TABLE 5. LMP Burst Order

Starting Offset	Burst Address Sequence
0x18	
0x1C	

If you are using a local master that uses linear order bursting, you must inhibit bursting for some starting addresses. For example, both sub-block and linear bursting follow the same addressing pattern when the address starts at offset 0x0 (0x0, 0x4, 0x8, 0xC...) Bursts starting on an 0x4 offset, however, will follow the linear addressing patterns (0x4, 0x8, 0xC, 0x0.) Such bursts are not supported by the GT-64060. Many masters allow you to terminate bursting through an external pin. With such masters a simple PAL may be used to check the starting address and inhibit bursting when the starting address will

3.1.8 LMP Interface Tips

While the LMP protocol may appear complicated at first glance, it is actually extremely powerful and adaptable to a wide range of bus protocols. In this section, we'll give you a few tips to adapt the protocol to your local master.

3.1.8.1 Address Phase Interface Issues

During the address phase you must encode the following information into the MasCmd[8:0] bus:

- Whether the access is a read or a write (set/clear MasCmd[6]).
- How long is the access (1, 2, 3, 4, 8, or 16, 32 bytes). (MasCmd[4:0]).

Usually the read/write indication from the local master can be gated directly to MasCmd[6], though the polarity may need to be changed. Most processors also give a burst length indication which can be re-encoded into MasCmd[4:0].

Occasionally, the burst length indication may not be immediately obvious on a processor. Many processors however, will only burst in response to a cache line fill, and this is usually a fixed length (8 or 16 bytes.) Cache fills are almost always indicated by an external pin. In short, you can usually find a way to determine the burst length before the cycle starts. Processors that give no such indication, either direct or indirect, will not be able to perform burst accesses through the LMP.

The ValidOut* and Release* signals must also be derived from the local master's interface signals. These signals are typically derived from the start-of-cycle and read/write indications.

3.1.8.2 Read Data Phase Issues

In response to read requests, the GT-64060 will drive valid data on SysAD[31:0], a valid data identifier on SysCmd[8:0], and a valid data/command indication on ValidIn*. Local masters that use a READY signal can use ValidIn* as READY and can ignore SysCmd[8:0]. Optionally, the ERROR bit in MasCmd[5] may be sampled to detect data errors on reads.

The interface logic must float MasCmd[8:0] during reads to prevent contention on this bus.

3.1.8.3 Write Data Phase Issues

The interface logic must drive MasCmd[8:0] during writes with the proper bus command. You must also take into account the functionality of WrReady* before allowing writes to proceed.

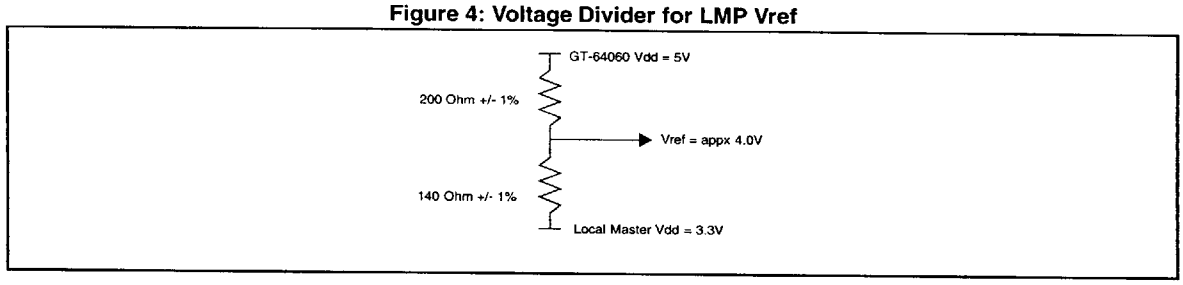
All LMP writes complete in zero wait-states.

3.1.9 LMP Voltage Reference

The GT-64060 processor interface voltage swing is controlled by the Vref pin. The voltage on this pin controls the Voh level for all Local Master Port interface pins. For 3.3V masters, Vref should be tied to a voltage divider between the

3.3V supply and the 5V supply as shown in Figure 4. Systems using 5V masters must tie Vref to the 5V supply. The actual Voh level is set to Vref-0.7V (a diode-drop from Vref.)

Figure 4: Voltage Divider for LMP Vref



4. Address Space Decode

The GT-64060 uses a distributed address decoding scheme. Each "master" unit (Local Master Port or PCI) has a separate address decoding logic and registers. The DMA controller uses the address mapping of the Local Master Port interface.

Address space for the different system resources is programmable and can be programmed differently in each "master" unit. The system resources are divided into eight groups: RAS[1:0], RAS[3:2], CS[2:0], CS[3] & BootCS, Internal, PCI I/O, and PCI memory0 and PCI memory. Each group can have a minimum of 2 Mbytes and a maximum of 256 Mbytes of address space.

The individual devices in the device groups (e.g. RAS[0]) are further sub decoded to 1 Mbyte resolution within the memory/device interface. The sub decoding is not distributed in the different "master" units but is centralized in the Device Controller unit. The system resources groups can be mapped into a 64 Gbyte address space for Local Master Port accesses and into 4 Gbyte address space for the DMA and PCI accesses.

When the Local Master Port tries to access an address that is not supported, the GT-64060 will latch the address into the Bus Error registers, and will issue a bus error (over MasCmd[5]) if the access was a read access, and an interrupt if it was a read or write access. It is possible that an access can be a hit in the primary decoders (i.e. the MasAD decoders or the PCI decoders) and miss in the Device Controller. This will result in a bus error as well.

5. Device/Memory Controller (AD Bus)

5.1 Memory Control

All memory and I/O devices in a GT-64060 system are connected to the AD bus; the MasAD bus is used primarily as a point-to-point connection between Local Master Port and chipset. The GT-64060 AD bus can support 8-, 16-, and 32-bit devices. 64-bit interleaving is also supported on the AD bus to improve performance.

5.2 DRAM Controller

The DRAM controller supports page mode and EDO DRAM. The depth of the DRAM devices can vary for each bank separately from 256K to 16M, and the width of each bank may be 32 bits or 64 bits interleaved. With these options, each DRAM bank size can vary from 1 Mbyte to 128 Mbytes. Furthermore, 0.5K, 1K, 2K, and 4K refresh DRAMs can be used, as well as asymmetric RAS/ CAS addressing.

Some of the DRAM timing parameters are programmable to allow for different system timing optimizations. RAS-to-CAS delay can be programmed for two or three cycles, and CAS can be LOW for one or two cycles.

DRAM performance in Local Master Port read accesses is 7-1-1-1-1-1-1-1 at 50MHz (when interleaved 64-bit memory is implemented), which means 4 wait states to first data and zero wait states for each additional word. DMA and PCI burst accesses can be one per clock for a maximum of 8 consecutive 32-bit words.

Refresh can be programmed to different frequencies of occurrences by the refresh counter. For example, if the refresh counter is programmed to 0x200, then at 50MHz a refresh sequence will occur every 10uS (20nS x 200). Staggered and non-staggered refresh modes are supported. In staggered mode, the four banks of DRAM will be refreshed with one cycle delay between each bank, while in non-staggered mode all four banks will be refreshed together.

5.3 Device Controller

The device controller has programmable timing parameters for each device bank to accommodate different device types (e.g. Flash, SRAM, ROM, I/O Controllers). The devices share the local AD bus with the DRAM, but unlike the DRAM, the devices use the AD bus as a multiplexed address and data bus. In the address phase, the device controller puts on the bus 22-bits of address, four general purpose Chip Select signals (CS[3:0]*), one Boot Chip Select (BootCS*), four DMA Acknowledge signals (DMAAck[3:0]*), and an indication as to whether the access is a read or write (DevRW*).

A bus cycle starts by the assertion of ALE and ADS* for one cycle when a CS* signal and/or a DMAAck* signal are active. The CS* and DMAAck* need to be externally latched and qualified with CSTiming*. The CSTiming* signal will be valid for the programmable number of cycles of the specific CS* that is active.

There are eight byte write signals (EWr[3:0]* and OWr[3:0]*). The write signals can be shaped by specifying the following: the number of cycles from the assertion of ADS* to the first assertion of write, the number of cycles the write pulse is active (LOW) - could be extended by Ready*, and the number of cycles the write signal is non-active between consecutive writes. The timing parameters of the write signals determine the length of active CS* (or DMAAck*) signals, as well as the external latch control and burst address change timing.

For read cycles, a device access time to first data (could be extended by Ready*) and to the following data (could be extended by Ready*) in burst accesses defines the cycle parameters. The access time determines the timing that data will be latched, and when the burst address will change.

The device controller supports up to 8 word burst accesses. The burst address is supported by a three bit wide address bus (BAdr[2:0]) that is different from the multiplexed AD bus. The same bus also supports the packing of data into a 64-bit double word, in reads from devices that are 8-bits to 32-bits wide. Devices that are 8-bits or 16-bits wide only are supported by partial reads (up to 64-bits). The controller supports Local Master Port writes of 1 to 8 bytes to 8-bit or 16-bit wide devices. It supports DMA/PCI writes of 1 to 4 bytes to 8-bit or 16-bit wide devices. 64-bit interleaved devices are also supported.

5.4 Ready* Support

The Ready* pin is sampled on two different occasions: on the last rising edge of the WrActive phase during a write cycle and one clock before the data is sampled to the GT-64060 during both AccToFirst and AccToNext phases. During all other phases Ready* is not sampled by the GT64011.

If Ready* is not asserted during these clocks, the WrActive, AccToFirst or AccToNext phases are extended until Ready* is asserted again. See the timing diagrams added for read and write cycles that are controlled by Ready*.

5.5 Parity Support

Memory or device parity generation and checking is supported with external logic. The external logic should generate parity in write accesses to devices and memory and check parity in read accesses. When a parity error is detected by the external logic it needs to drive the ParErr* pin of the GT-64060. The GT-64060 has a programmable parity integrity bit for each bank, which indicates if parity is supported.

In Local Master Port read accesses from a 32-bit device or memory, the GT-64060 will not assert MasCmd[4] even if the bank that was accessed has the parity integrity bit set. If a parity error is detected in this case (indicated by ParErr*), the GT-64060 will return the data with MasCmd[5] asserted and will cause a parity error interrupt.

In DMA read accesses, detection of a parity error from a bank with the parity integrity bit set, will cause an interrupt.

In the case of PCI read accesses, the GT-64060 will assert SErr* (if unmasked) if the bank that data was read from has the parity integrity bit set, and will assert a parity error interrupt. The GT-64060 will generate and check word (32- bits) parity on data that is read from the PCI with compliance to the PCI requirements for every transaction. A parity error detection on the PCI will cause the assertion of PErr*.

5.6 DMA Controller

The DMA controller can move data between devices on the AD bus, between devices on the PCI bus, or between devices on the AD bus and devices on the PCI bus. All DMA transfers use an internal 32-byte FIFO for moving data. Data is transferred from the source device into the internal FIFO, and from the internal FIFO to the destination device. The length of each transfer of DMA can be limited to 1, 2, 4, 8, 16 or 32 bytes. Accesses can be non-aligned both in the source and the destination. The DMA can be programmed to move up to 64 KBytes of data in each transaction.

The DMA controller supports chained and non-chained modes of operation. In the non-chained mode the Local Master Port or the PCI program the DMA channel for each DMA transaction. In chained mode, the DMA controller programs itself for the next DMA operation by fetching the information from a linked list of records in memory.

The DMA controller can be programmed to assert an interrupt in chained mode at the end of every DMA transaction, or when the Next Pointer Register is Null and Byte Count reaches terminal count. In non-chained mode, the DMA will assert an interrupt every time the Byte Count reaches terminal count.

DMA accesses can be initiated by an external request by asserting one of the four DMAReq[3:0]* pins (Demand mode), or by setting an internal bit in a register (Block mode).

Accesses by the four DMA channels can be prioritized via a programmable arbiter. Channels 0 and 1 are in one group and channels 2 and 3 are in another group. Inside each group, the priority can be fixed so a selected channel number can have a higher priority, or both can have the same priority in round-robin fashion. The same scheme applies between the two groups, they can have fixed or round-robin priority.

6. PCI Bus

The GT-64060 includes a Revision 2.1 compliant PCI interface. As a PCI device, the GT-64060 can be either a master initiating a PCI bus operation or a target responding to a PCI bus operation.

6.1 PCI Master Operation

When the Local Master Port or the internal DMA machine initiates a bus cycle to a PCI device, the GT-64060 becomes a PCI bus master and translates the cycle into the appropriate PCI bus cycle. Supported master PCI cycles are:

- Memory Read
- Memory Write
- Memory Read line
- Memory Write & Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Interrupt Acknowledge
- Special Cycle

Memory Write & Invalidate and Memory Read Line cycles are carried out when the transaction accessing PCI memory space requests a data transfer equal to the PCI cache line size. When the PCI cache line size is set equal to 0, the GT-64060 will never issue Memory Write & Invalidate or Memory Read Line cycles.

As a master, the GT-64060 does not issue Dual Address cycles or Lock cycles on the PCI.

The PCI posted write buffer in the GT-64060 permits the Local Master Port to complete Local Master Port-to-PCI memory writes even if the PCI bus is busy. The posted data is written to the PCI device when the PCI bus is available.

6.1.1 PCI Master LMP Address Space Decode and Translation

Local masters access the PCI space through the PCI Memory 0, PCI Memory 1 and PCI I/O decoders in LMP address space. LMP accesses that are claimed by these decoders are translated into the appropriate PCI cycles. The address seen on the LMP bus is copied directly to the PCI bus. For example, if and accesses to 0x1200.0040 is programmed to be bridged as a memory read from PCI, then the active PCI address for this cycle will be 0x1200.0040. Access to the full PCI space is possible by relocating the LMP PCI decoders within LMP space as needed.

6.1.2 PCI Master LMP Byte Swapping

All accesses to PCI space through the LMP can have the data byte order swapped as the data moves through the GT-64060. Byte swapping is turned on via the ByteSwap bit in the PCI Internal Command register (0xc00.) Note that this bit does not affect the byte order of data moved through the target interface (PCI to DRAM, for example).

6.1.3 PCI Master FIFOs

The PCI master interface includes a FIFO of 8 entries, each 32 bit. During writes to the PCI interface, it receives write data from the Local Master Port interface or the DMA unit. When the PCI bus is granted, the FIFO delivers the write data to the target on the PCI bus.

Upon receiving the first 32-bit word from the Local Master Port interface or DMA unit, the PCI master interface will request the PCI bus (if the GT-64060 is not already parked). Once granted, the appropriate write cycle is started on the PCI bus.

During reads, the PCI master interface FIFO receives read data from the PCI bus and delivers it to the Local Master Port interface or the DMA unit. Upon receiving the first 32-bit word from the PCI target, the data is forwarded to the requesting unit (Local Master Port interface or DMA unit). The GT-64060 supports sub-block ordering during Local Master Port reads, therefore if the original read request address is not aligned to a cache line boundary, the first 32-bit word returned to the requesting unit will be delayed until it is received from the PCI target, since reads across the PCI

bus are linear.

The GT-64060 internal architecture allows zero wait-state data transfer over the PCI bus (Irdy* continuously asserted) during both master reads and writes.

6.1.4 PCI Master DMA

The GT-64060's internal DMA engines can act as PCI bus masters while transferring data to/from the PCI bus. The DMA engines will only issue memory space read and write cycles. The type of cycle issued follows the same rules as for the LMP. The DMA engines can transfer data between PCI devices using the on-chip DMA FIFOs for temporary storage.

6.1.5 PCI Master RETRY Counter

RETRY's detected by the PCI master interface are normally handled transparently from the point of view of the LMP or DMA engines. In some rare circumstances, however, a target device may RETRY the GT-64060 excessively (or forever.) The Retry Counter can be used to recover from this condition. Every time the number of RETRYs equals the value in the Retry Counter, the GT-64060 will abort the cycle and send an interrupt to the CPU. If the cycle was a read, undefined data is returned and the ERROR bit is set in the data command.

The Retry Counter can be disabled by setting the Retry count to zero.

6.2 PCI Target Interface

The GT-64060 responses to the following PCI cycles as a target device:

- Memory Read
- Memory Write
- Memory Read Line
- Memory Read Multiple
- Memory Write and Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write

The GT-64060 will lock a cache line (32-bytes) in the local memory address space when responding to Lock sequences on the PCI bus. The GT-64060 will not act as a target for Interrupt Acknowledge, Special, and Dual Address cycles (these cycles will be ignored.)

6.2.1 PCI Target FIFOs

The GT-64060 incorporates dual 32-byte posted write/read prefetch buffers to allow full memory (AD) and PCI bus concurrency. The dual FIFOs operate in a "ping-pong" fashion, each FIFO alternating between filling and draining.

When the GT-64060 is the target of PCI write cycles, data is first written to one of the FIFOs. When the first FIFO fills up (32 bytes), the data is written to the destination from the first FIFO while the second FIFO is filled. This "ping-pong" operation continues as long as data is received from the PCI bus.

Occasionally the PCI target interface cannot drain the FIFOs (i.e. write to local memory) as fast as data is received. This will only happen when access to memory is prevented (possibly by excessive LMP accesses) or when the target memory is particularly slow. In this case, the GT-64060's PCI target interface will issue a DISCONNECT to the PCI bus.

Figure 5: PCI Target Interface "Ping-Pong" FIFOs

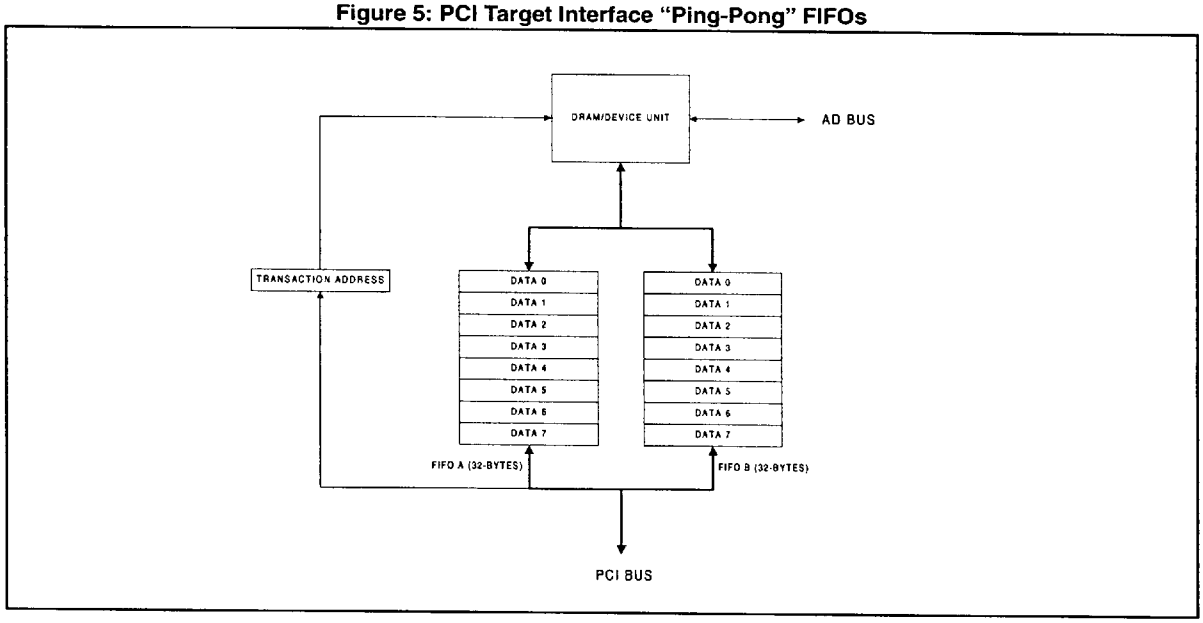
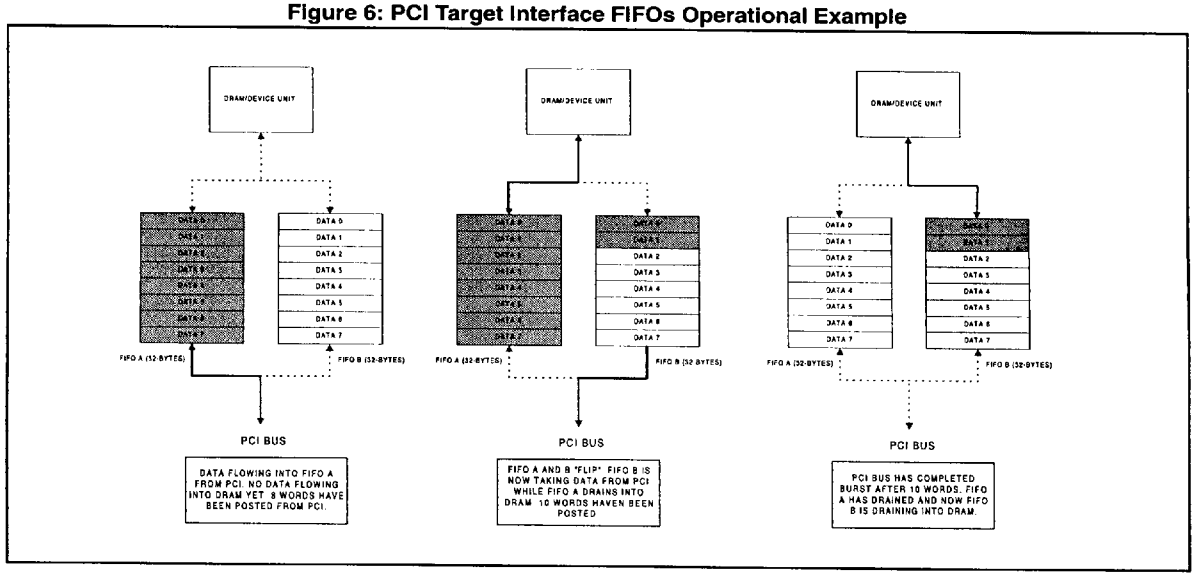


Figure 6: PCI Target Interface FIFOs Operational Example



The target FIFOs are also used for read prefetch. The Memory Read Multiple (MRM) cycle is the only prefetchable read cycle. In response to any target PCI read cycle, the GT-64060 will read an entire cache line (32 bytes) from memory into one of the target FIFOs.

If the read is a Memory Read Multiple, as soon as at least two words are delivered from the FIFO to the PCI bus, another 32 bytes is prefetched into the second FIFO. In this case, the GT-64060 is essentially "guessing" that MRM cycle will be longer than 32 bytes.

In a non-prefetchable read cycle, data is not fetched into the second FIFO until after all data from the first FIFO is delivered to the PCI bus.

Cycles to internal registers and Configuration cycles are non-postable or prefetchable.

6.2.2 PCI Target Address Space Decode and Byte Swapping

The GT-64060 decodes accesses on the PCI bus for which it may be a target by the values programmed into its Base Address Registers (BARs). There are two sets of BARs: regular BARs (in PCI Function 0) and swap BARs (in PCI Function 1). Accesses decoded by the swap BARs are passed with to/from the target memory device after converting the endianess of the data (e.g. little-endian to big-endian). Accesses decoded by the regular BARs, by comparison, are passed without modifying the data to/from the target memory device.

The GT-64060 uses a two stage decode process for accesses through the PCI target interface. Once a PCI accesses is determined to be a "hit" based on the BAR comparison, the address is passed to the Device Unit for sub-decode. For example, Base Address Register 0 in Function 0 (BAR0) decodes non-byte swapped accesses to the DRAM controlled by either RAS0* or RAS1*. The GT-64060 then uses the values programmed into the RAS0 Low and RAS0 High decode registers to determine if the access is to the DRAM connected to RAS0. Note that the second stage decoders are shared with the LMP (see "LMP Address Space Decode" for a nice picture showing this.)

If the target PCI address "hits" based on the BAR decode, then misses in the Device Unit, then the PCI bus will be RETRIED.

6.2.3 Tweaking the Performance of the Target Interface

The GT-64060 includes special performance tuning features for the PCI target interface. The Timeout0 and Timeout1 registers allow the designer to force the GT-64060 to wait either longer than normal, or shorter than normal, before issuing a RETRY/DISCONNECT. The Timeout0 value sets the number of clocks the GT-64060 will wait for the first data of an access before issuing a RETRY. The Timeout1 value sets the number of clocks the GT-64060 will wait between subsequent data phases during an access before issuing a DISCONNECT. The PCI 2.1 specifications sets the maximum for both of these at 16 clocks (Timeout0) and 8 clocks (Timeout1) repsectively. However, in many systems, especially those with long DRAM latencies, it may be necessary to "bend" these restrictions.[1]

If you see what appears to be excessive RETRY/DISCONNECT behavior in your system, try lengthening the Timeout0/1 values. It may be because there is a lot of memory activity due to the CPU or DMA engines, and the PCI interface cannot get to the DRAM within 16 clocks.

6.3 PCI Synchronization Barriers

The GT-64060 considers some cycles to be "synchronization barrier" cycles. In such cycles, the GT-64060 makes sure that at the end of the cycle there remains no posted data within the chip.

The target "synchronization barrier" cycles are Lock Read and Configuration Read. If there is no posted data within the GT-64060, the cycle ends normally. If after a retry period there is still posted data, the cycle will be retried. Until the original cycle ends, any other (different address/command) "synchronization barrier" cycles will be retried. Lock Read is a "synchronization barrier" cycle which lasts during the entire Lock period, i.e. when the slave is locked all Configuration Reads will be retried. Also, all cycles addressed to internal registers will be retried until Lock ends.

The Local Master Port interface treats I/O Reads to PCI and Configuration Reads as "synchronization barrier" cycles as well. These reads will be responded to once no posted data remains within the GT-64060.

6.4 PCI Master Configuration

The GT-64060 translates Local Master Port read and write cycles into configuration cycles using PCI configuration mechanism #1 (per the PCI spec). Mechanism #1 defines a way to translate the Local Master Port cycles into both PCI configuration cycles on the PCI bus, and accesses to the GT-64060's internal configuration registers.

The GT-64060 includes two registers: Configuration Address (at offset 0xcf8) and Configuration Data (at offset 0xcfc). The mechanism for accessing configuration registers is to write a value into the Configuration Address register that specifies:

1. The PCI specification also states that that a master may not enforce target rules. In other words, even if the GT-64060 takes longer than 16 clocks to return the first data, the master must just wait patiently.

- PCI bus number (usually bus 0, the bus attached directly to the GT-64060)
- The device on that bus
- The function number within the device
- The configuration register within that device/function being accessed

A subsequent read or write to the Configuration Data register (at 0xcfc) then causes the GT-64060 to translate that Configuration Address value to the requested cycle on the PCI bus.

If the BusNum field in the Configuration Address register equals '0' but the DevNum field is other than '0', a Type0 access is performed which addresses a device attached to the local PCI bus. If the BusNum field in the Configuration Address register is other than '0', a Type1 access is done which addresses a device attached to a remote PCI bus.

The GT-64060 performs address stepping for PCI configuration cycles. This allows for the use of the high-order PCI AD signals as IdSel signals through resistive coupling.[1] Table 6 shows DevNum to IdSel mapping.

TABLE 6. DevNum to IdSel Mapping

DevNum[15:11]	PAD[31:11]
00001	0 0000 0000 0000 0000 0001
00010	0 0000 0000 0000 0000 0010
00011	0 0000 0000 0000 0000 0100
00100	0 0000 0000 0000 0000 1000
- - -	- - -
10101	1 0000 0000 0000 0000 0000
00000, 10110 - 11111	0 0000 0000 0000 0000 0000

The Local Master Port accesses the GT-64060's internal configuration registers when the fields DevNum and BusNum in the Configuration Address register are equal to '0'. The GT-64060 configuration registers are also accessed from the PCI bus when the GT-64060 is a target responding to PCI configuration read and write cycles.

Note: The Local Master Port interface unit cannot distinguish between an access to the GT-64060 PCI configuration space and an access to an external PCI device configuration space. This is because both are accessed using an access to the GT-64060 internal space (i.e. Configuration Data register). When the Local Master Port is operating in big-endian mode, any access to the GT-64060 internal space undergoes byte swapping as all internal registers are little-endian. With the Local Master Port operating in big-endian mode and the PCI ByteSwap bit (bit [0] @ 0xc00) set to '0' (i.e., swap bytes), bytes will be swapped once for PCI configuration accesses intended for the GT-64060 configuration space but will be swapped twice for PCI configuration accesses intended for devices external to the GT-64060. This requires the software to format write data and interpret read data differently for PCI configuration accesses to the GT-64060's registers and configuration accesses through the GT-64060 to an external device.

IMPORTANT: *The configuration enable bit (ConfigEn) in the Configuration Address register must be set before the Configuration Data register is read or written. Failure to do this will "lock up" the GT-64060 and require a RESET to recover.*

6.4.1 Special Cycles and Interrupt Acknowledge

A Special cycle is generated whenever the Configuration Data register is written to while the Configuration Address register has been previously written with '0' for BusNum, '1f' for DevNum, '7' for FunctNum and '0' for RegNum.

1. "Resitive Coupling" is a fancy way of saying "hook a resistor from ADx to IdSel" on a given device. Look at the Galileo-4PB backplane schematics for examples.

An Interrupt acknowledge cycle is generated whenever the Interrupt Acknowledge (0xc34) register is read.

6.5 Target Configuration and Plug and Play

The GT-64060 includes all of the required plug and play PCI configuration registers. These registers, as well as the GT-64060's internal registers, may be accessed from both the Local Master Port and the PCI bus.

The GT-64060 acts as a two function device when being configured from the PCI bus. The base address registers available in Function 0 are used to decode accesses for whcih there is no byte swapping; Function 1 is used to decode byte swapped addresses. All other registers are shared between Function 0 and Function 1.

6.5.1 Plug and Play Base Address Register Sizing

Systems adhering to the plug and play configuration standard determine the size of a base address register's decode range by first writing 0xFFFF.FFFF to the BAR, then reading back the value contained in the BAR. Any bits that were unchanged (i.e. read back a zero) indicate that they cannot be set and are therefore not part of the address comparison. With this information the size of the decode region can be determined.[1]

The GT-64060 responds to BAR sizing requests based on the values programmed into the Bank Size Registers. These registers can be loaded automatically after RESET from the system ROM (see below).

6.5.2 PCI Autoconfiguration at RESET

Eight PCI registers can be automatically loaded after Rst*. Autoconfiguration mode is enabled by asserting the DMAReq[3]* LOW on Rst*. Any PCI transactions targeted for the GT-64060 will be retried while the loading of the PCI configuration registers is in process.

The PCI register values are loaded from the ROM controlled by BootCS* are shown in Table 7, below.

TABLE 7. PCI Registers Loaded at RESET

Register	Offset	Boot Device Address
Device and Vendor ID	0x000	0x1fffffe0
Class Code and Revision ID	0x008	0x1fffffe4
Subsystem Device and Vendor ID	0x02c	0x1fffffe8
Interrupt Pin and Line	0x03c	0x1fffffec
RAS[1:0]* Bank Size	0xc08	0x1ffffff0
RAS[3:2]* Bank Size	0xc0c	0x1ffffff4
CS[2:0]* Bank Size	0xc10	0x1ffffff8
CS[3]* & Boot CS* Bank Size	0xc14	0x1ffffffc

6.6 PCI Bus/Device Bus/LMP Clock Synchronization

The PCI interface is designed to run asynchronously with respect to the AD and LMP buses. The synchronization delay between these two clock domains can be reduced, however, by running the interfaces in synchronized mode. An example would be having the LMP/AD buses running at 50MHz and the PCI bus running at a 25MHz frequency that was derived from the 50MHz.

Latency through the GT-64060 is reduced to a minimum when synchronized mode is selected. The synchonization mode is set via the SyncMode bits in the PCI Internal Command register (0xc00).

1. Please refer to the PCI specification for more information on the BAR sizing process.

7. Interrupt Controller

The interrupt controller groups all the internal interrupt sources and asserts an interrupt to the Local Master Port or to the PCI when one or more internal interrupts are asserted. There is one Cause register and two Mask registers. The Cause register has one bit for each interrupt source. If the source asserts an interrupt, its respective bit in the Cause register will be set. This bit can be read by the Local Master Port or by the PCI. The interrupt will be acknowledged by the Local Master Port or by the PCI by resetting its bit in the Cause register (writing zero to the specific bit and one to all the other bits.). Each interrupt source has one mask bit in the Local Master Port Mask register and one bit in the PCI Mask register. A zero in the Local Master Port Mask register bit will mask the interrupt from asserting an interrupt to the Local Master Port. A zero in the PCI Mask register bit will mask the interrupt from asserting an interrupt to the PCI.

8. Timer/Counters

The GT-64060 has three 24-bit and one 32 bit timers/counters. When programmed as a counter, the counter will decrement every clock, will set an interrupt and will stop counting. In the timer mode, it will set the interrupt but will reload to the initial value and continue to count down. The initial value for each timer counter is programmable. Write accesses are done to the timer/counter register but read accesses (from the same address) are directly from the counter outputs.

9. Reset Configuration

The GT-64060 must acquire some knowledge about the system before it is configured by the software. Special modes of operation are sampled on Reset in order to enable the GT-64060 to function in consistence with the specific system it is used in. Certain pins must be pulled up or down (4K7 recommended) externally to accomplish this. The following configuration pins are continuously sampled from Rst* assertion until 3 TClk cycles after Rst* is deasserted.

Interrupt*:	Endianess
0-	Big endian data format
1-	Little endian data format
DAdr[11:10]:	Device Boot Bus Width
00-	8 bits
01-	16 bits
10-	32 bits
11-	64 bits
DAdr[9]:	LEAdrE/DMAReq[2]* Selection
0-	DMAReq[2]*
1-	LEAdrE
DAdr[8]:	OEB Polarity
0-	Active LOW
1-	Active HIGH
DAdr[7]:	External Latches Presence
0-	Latches are present
1-	System without latches
DAdr[6]:	Address matching with CS[2:0] base/size
0-	Enable
1-	Disable
DAdr[5]:	DMAReq[1]*/ParErr* Selection
0-	DMAReq[1]* (no parity)
1-	ParErr*
DAdr[4]:	DMAReq[0]*/Ready* Selection
0-	Ready*
1-	DMAReq[0]*
DAdr[3]:	Address matching with CS[3] & BootCS base/size

0-	Enable
1-	Disable
DAdr[2]:	Address matching with Swapped RAS[1:0] base/size
0-	Enable
1-	Disable
DAdr[1]:	Address matching with Swapped RAS[3:2] base/size
4-	Enable
1-	Disable
DAdr[0]:	Address matching with Swapped CS[3] & Boot CS base/size
0-	Enable
1-	Disable
DMAReq[3]*:	Autoload Enable
0-	Enable
1-	Disable
DMAReq[1]*/ ParErr*:	Address matching with interal registers I/O mapped base/size
0-	Enable
1-	Disable
DMAReq[0]*/ Ready*:	Address matching with RAS[3:2] base size
0-	Enable
1-	Disable

Notes:

1. LEAdrE/DMAReq[2]* should be selected '0' whenever LEAdrE is not used in the system (e.g., DRAM is not operating in decrement mode).

2. If Autoload enable is sampled low during reset, device and vendor ID[31:0], class code and revision ID[31:0], Subsystem ID and Subsystem Vendor ID[31:0], Interrupt Pin, Interrupt Lines[15:0], RAS[1:0]* Bank Size, RAS[3:2]* Bank Size, CS[2:0]* Bank Size, CS[3] & BootCS* Bank Size will be autoloaded from address 0x1FFF.FFE0.

10. REGISTER TABLES

The GT-64060's internal registers can be accessed by the Local Master Port or by the PCI. They are memory-mapped for the Local Master Port and memory- or I/O-mapped for the PCI. The registers' address is comprised of the value in the "Internal Space Decode" register and the register Offset. The value in the "Internal Space Decode" register [10:0] is matched against bits [31:21] of the actual address; therefore, this value should be the actual address bits [31:21] shifted right once.

For example, to access "Channel 0 DMA Byte Count" register (offset 0x800) immediately after Reset*, the full address will be the default value in the "Internal Space Decode" register which is 0x0a0 shifted left once, which gives 0x140, two zero's and the offset 0x800, to become a 32-bit address of 0x14000800. The location of the registers in the memory space can be changed by changing the value programmed into the "Internal Space Decode" register. For example after changing the value in the "Internal Space Decode" register by writing to 0x14000068 a value of "0bd", an access to the "Channel 0 DMA Byte Count" register will be with 0x17a00800.

An access to a PCI configuration register is done differently than accesses to all other registers. The access is done indirectly by writing the PCI configuration register offset into the Configuration Address register and then reading or writing the data from/to the Configuration Data register. For example, to read data from the Status and Command register, the register offset "0x004" has to be written into the Configuration Address register, offset 0xcf8 (or full address from the previous example 0xbd000cf8). Then, reading from the Configuration Data register (offset 0xcfc), will return the data of the Status and Command register.

10.1 Register Map

Description	Offset
Local Master Port Interface	
Local Master Port Interface Configuration	0x000
Processor Address Space	
RAS[1:0] Low Decode Address	0x008
RAS[1:0] High Decode Address	0x010
RAS[3:2] Low Decode Address	0x018
RAS[3:2] High Decode Address	0x020
CS[2:0] Low Decode Address	0x028
CS[2:0] High Decode Address	0x030
CS[3] & Boot CS Low Decode Address	0x038
CS[3] & Boot CS High Decode Address	0x040
PCI I/O Low Decode Address	0x048
PCI I/O High Decode Address	0x050
PCI Memory0 Low Decode Address	0x058
PCI Memory0 High Decode Address	0x060
Internal Space Decode	0x068
Bus Error Address Low Processor	0x070
Read Only '0'	0x078
PCI memory Low Decode Address	0x080
PCI memory High Decode Address	0x088
DRAM and Device Address Space	
RAS[0] Low Decode Address	0x400
RAS[0] High Decode Address	0x404
RAS[1] Low Decode Address	0x408
RAS[1] High Decode Address	0x40c
RAS[2] Low Decode Address	0x410
RAS[2] High Decode Address	0x414
RAS[3] Low Decode Address	0x418
RAS[3] High Decode Address	0x41c
CS[0] Low Decode Address	0x420
CS[0] High Decode Address	0x424

CS[1] Low Decode Address	0x428
CS[1] High Decode Address	0x42c
CS[2] Low Decode Address	0x430
CS[2] High Decode Address	0x434
CS[3] Low Decode Address	0x438
CS[3] High Decode Address	0x43c
Boot CS Low Decode Address	0x440
Boot CS High Decode Address	0x444
Address Decode Error	0x470
DRAM Configuration	
DRAM Configuration	0x448
DRAM Parameters	
DRAM Bank0 Parameters	0x44c
DRAM Bank1 Parameters	0x450
DRAM Bank2 Parameters	0x454
DRAM Bank3 Parameters	0x458
Device Parameters	
Device Bank0 Parameters	0x45c
Device Bank1 Parameters	0x460
Device Bank2 Parameters	0x464
Device Bank3 Parameters	0x468
Device Boot Bank Parameters	0x46c
DMA Record	
Channel 0 DMA Byte Count	0x800
Channel 1 DMA Byte Count	0x804
Channel 2 DMA Byte Count	0x808
Channel 3 DMA Byte Count	0x80c
Channel 0 DMA Source Address	0x810
Channel 1 DMA Source Address	0x814
Channel 2 DMA Source Address	0x818
Channel 3 DMA Source Address	0x81c
Channel 0 DMA Destination Address	0x820

Channel 1 DMA Destination Address	0x824
Channel 2 DMA Destination Address	0x828
Channel 3 DMA Destination Address	0x82c
Channel 0 Next Record Pointer	0x830
Channel 1 Next Record Pointer	0x834
Channel 2 Next Record Pointer	0x838
Channel 3 Next Record Pointer	0x83c
DMA Channel Control	
Channel 0 Control	0x840
Channel 1 Control	0x844
Channel 2 Control	0x848
Channel 3 Control	0x84c
DMA Arbiter	
Arbiter Control	0x860
Timer/Counter	
Timer /Counter 0	0x850
Timer /Counter 1	0x854
Timer /Counter 2	0x858
Timer /Counter 3	0x85c
Timer /Counter Control	0x864
PCI Internal	
Command	0xc00
Time Out & Retry	0xc04
RAS[1:0] Bank Size	0xc08
RAS[3:2] Bank Size	0xc0c
CS[2:0] Bank Size	0xc10
CS[3] & Boot CS Bank Size	0xc14
SErr Mask	0xc28
Interrupt Acknowledge	0xc34
Base Address Registers' Enable	0xc3c
Configuration Address	0xcf8
Configuration Data	0xcfc
Interrupts	

Interrupt Cause	0xc18
Local Master Port Mask	0xc1c
PCI Mask	0xc24
PCI Configuration	
Device and Vendor ID	0x000
Status and Command	0x004
Class Code and Revision ID	0x008
BIST, Header Type, Latency Timer, Cache Line	0x00c
RAS[1:0] Base Address	0x010
RAS[3:2] Base Address	0x014
Subsystem Device and Vendor ID	0x02c
CS[2:0] Base Address	0x018
CS[3] & Boot CS Base Address	0x01c
Internal Registers Memory Mapped Base Address	0x020
Internal Registers I/O Mapped Base Address	0x024
Expansion ROM Base Address Register	0x030
Interrupt Pin and Line	0x03c
PCI Configuration, Function 1	
RAS[1:0] Swapped Base Address	0x010
CAS[3:2] Swapped Base Address	0x014
CS[3] and Boot CS Swapped Base Address and Expansion ROM Base	0x01c

10.2 Local Master Port Interface

The Local Master Port Interface Configuration register determines which of the different MIPS write protocols is supported, as well as Local Master Port endianess. The differences in protocol are minimal and they include support for pipelined write, etc.

Local Master Port Interface Configuration, Offset: 0x000

Bits	Field name	Function	Initial Value
10:0	Reserved	Read Only '0'.	0x0

Bits	Field name	Function	Initial Value
11	WriteMod	Write mode. 0 - Pipelined writes mode 1 - R4000 mode (2 dead-cycles minimum between consecutive address-phases)	0x0
12	Endianess	Byte Orientation. 0 - Big Endian 1 - Little Endian	Sampled at reset via the Interrupt* pin

10.3 Processor Address Space

The Decode Address registers determine which physical device group will be accessed when the Local Master Port issues an address. The decode to the specific bank (RAS or CAS) in each group is done in the memory control unit. The address decoding is done by comparing bits 31:28 of the address to bits 10:7 of the Low field of all the Low Decode registers to find a match, and by comparing address bits 27:21 to be greater than or equal to bits 6:0 of the Low fields, and less than or equal to the High field. When an address is out of range (of all the Decode Address registers), the Local Master Port will be interrupted during a write and read access and a bus error will be asserted during a read access. The invalid address will be captured in the Bus Error Address Low and High registers. The DMA controller uses the Processor's address decoding.

RAS[1:0] Low Decode Address, Offset: 0x008

Bits	Field Name	Function	Initial Value
10:0	Low	DRAM banks 1 and 0 will be accessed when the decoded addresses are between Low and High.	0x000

RAS[1:0] High Decode Address, Offset: 0x010

Bits	Field Name	Function	Initial Value
6:0	High	DRAM banks 1 and 0 will be accessed when the decoded addresses are between Low and High.	0x07

RAS[3:2] Low Decode Address, Offset: 0x018

Bits	Field Name	Function	Initial Value
10:0	Low	DRAM banks 3 and 2 will be accessed when the decoded addresses are between Low and High.	0x008

RAS[3:2] High Decode Address, Offset: 0x020

Bits	Field Name	Function	Initial Value
6:0	High	DRAM banks 3 and 2 will be accessed when the decoded addresses are between Low and High.	0x0f

CS[2:0] Low Decode Address, Offset: 0x028

Bits	Field Name	Function	Initial Value
10:0	Low	Device banks 2, 1 and 0 will be accessed when the decoded addresses are between Low and High.	0x0e0

CS[2:0] High Decode Address, Offset: 0x030

Bits	Field Name	Function	Initial Value
6:0	High	Device banks 2, 1 and 0 will be accessed when the decoded addresses are between Low and High.	0x70

CS[3] & Boot CS Low Decode Address, Offset: 0x038

Bits	Field Name	Function	Initial Value
10:0	Low	Device bank 3 and the boot bank will be accessed when the decoded addresses are between Low and High.	0x0f8

CS[3] & Boot CS High Decode Address, Offset: 0x040

Bits	Field Name	Function	Initial Value
6:0	High	Device bank 3 and the boot bank will be accessed when the decoded addresses are between Low and High.	0x7f

PCI I/O Low Decode Address, Offset: 0x048

Bits	Field Name	Function	Initial Value
10:0	Low	The PCI I/O address space will be accessed when the decoded addresses are between Low and High.	0x080

PCI I/O High Decode Address, Offset: 0x050

Bits	Field Name	Function	Initial Value
6:0	High	The PCI I/O address space will be accessed when the decoded addresses are between Low and High.	0x0f

PCI Memory Low Decode Address, Offset: 0x058

Bits	Field Name	Function	Initial Value
10:0	Low	The PCI memory0 address space will be accessed when the decoded addresses are between Low and High.	0x090

PCI Memory High Decode Address, Offset: 0x060

Bits	Field Name	Function	Initial Value
6:0	High	The PCI memory0 address space will be accessed when the decoded addresses are between Low and High.	0x1f

Internal Space Decode, Offset: 0x068

Bits	Field Name	Function	Initial Value
10:0	IntDecode	Registers inside the GT-64060 will be accessed when MasAD bits 35:21 match the value programmed in bits 14:0.	0x0a0

Bus Error Address Processor, Offset: 0x070

Bits	Field Name	Function	Initial Value
31:0	IlegLoAdd	This register captures bits 31:0 of an illegal 32-bit address.	0x00000000

Reserved, Offset: 0x078

Bits	Field Name	Function	Initial Value
31:0	Reserved	Read Only '0'.	0x0

PCI memory Low Decode Address, Offset: 0x080

Bits	Field Name	Function	Initial Value
10:0	Low	The PCI memory address space will be accessed when the decoded addresses are between Low and High.	0x090

PCI memory High Decode Address, Offset: 0x088

Bits	Field Name	Function	Initial Value
6:0	High	The PCI memory address space will be accessed when the decoded addresses are between Low and High.	0x1f

10.4 DRAM and Device Address Space

The Decode Address registers determine which physical device will be accessed when the Local Master Port, DMA, or PCI issue an address. The address decoding is done by comparing address bits 27:20 to be greater than or equal to the value in the Low fields, and less than or equal to the value in the High fields. In case that no match occurs, an interrupt will be issued and the address causing the error will be latched in the Address Decode Error register. This error can occur when the Local Master Port or PCI decoding matches the address while the sub-decoding done in the memory unit doesn't match any of the addresses defined in the address space.

Note: If the value of the "Low Field" is greater than the value of the "High Field", the specified bank is disabled.

RAS[0] Low Decode Address, Offset: 0x400

Bits	Field Name	Function	Initial Value
7:0	Low	DRAM bank 0 will be accessed when the decoded addresses are between Low and High.	0x00

RAS[0] High Decode Address, Offset: 0x404

Bits	Field Name	Function	Initial Value
7:0	High	DRAM bank 0 will be accessed when the decoded addresses are between Low and High.	0x07

RAS[1] Low Decode Address, Offset: 0x408

Bits	Field Name	Function	Initial Value
7:0	Low	DRAM bank 1 will be accessed when the decoded addresses are between Low and High.	0x08

RAS[1] High Decode Address, Offset: 0x40c

Bits	Field Name	Function	Initial Value
7:0	High	DRAM bank 1 will be accessed when the decoded addresses are between Low and High.	0x0f

RAS[2] Low Decode Address, Offset: 0x410

Bits	Field Name	Function	Initial Value
7:0	Low	DRAM bank 2 will be accessed when the decoded addresses are between Low and High.	0x10

RAS[2] High Decode Address, Offset: 0x414

Bits	Field Name	Function	Initial Value
7:0	High	DRAM bank 2 will be accessed when the decoded addresses are between Low and High.	0x17

RAS[3] Low Decode Address, Offset: 0x418

Bits	Field Name	Function	Initial Value
7:0	Low	DRAM bank 3 will be accessed when the decoded addresses are between Low and High.	0x18

RAS[3] High Decode Address, Offset: 0x41c

Bits	Field Name	Function	Initial Value
7:0	High	DRAM bank 3 will be accessed when the decoded addresses are between Low and High.	0x1f

CS[0] Low Decode Address, Offset: 0x420

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 0 will be accessed when the decoded addresses are between Low and High.	0xc0

CS[0] High Decode Address, Offset: 0x424

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 0 will be accessed when the decoded addresses are between Low and High.	0xc7

CS[1] Low Decode Address, Offset: 0x428

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 1 will be accessed when the decoded addresses are between Low and High.	0xc8

CS[1] High Decode Address, Offset: 0x42c

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 1 will be accessed when the decoded addresses are between Low and High.	0xcf

CS[2] Low Decode Address, Offset: 0x430

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 2 will be accessed when the decoded addresses are between Low and High.	0xd0

CS[2] High Decode Address, Offset: 0x434

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 2 will be accessed when the decoded addresses are between Low and High.	0xdf

CS[3] Low Decode Address, Offset: 0x438

Bits	Field Name	Function	Initial Value
7:0	Low	Device bank 3 will be accessed when the decoded addresses are between Low and High.	0xf0

CS[3] High Decode Address, Offset: 0x43c

Bits	Field Name	Function	Initial Value
7:0	High	Device bank 3 will be accessed when the decoded addresses are between Low and High.	0xfb

Boot CS Low Decode Address, Offset: 0x440

Bits	Field Name	Function	Initial Value
7:0	Low	Boot bank will be accessed when the decoded addresses are between Low and High.	0xfc

Boot CS High Decode Address, Offset: 0x444

Bits	Field Name	Function	Initial Value
7:0	High	Boot bank will be accessed when the decoded addresses are between Low and High.	0xff

Address Decode Error, Offset: 0x470

Bits	Field Name	Function	Initial Value
31:0	ErrAddr	The addresses of accesses to invalid address ranges (those not in the range programmed in the DRAM or device decode registers) will be captured in this register.	0xffffffff

10.5 DRAM Configuration

The DRAM Configuration register specifies refresh parameters and optional usage of two of the GT-64060 pins related to the DRAM controller. The time between refresh cycles is programmable, with the option to refresh all the banks at the same time or in staggered fashion. The pin functionality of DRAM address bit 11 can be programmed to be ADS* only for systems that do not have deep DRAMs. This pin can also be programmed to be ADS* in device accesses, and to function as DRAM address 11 in DRAM accesses.

DRAM Configuration, Offset: 0x448

Bits	Field name	Function	Initial Value
13:0	RefIntCnt	Refresh interval count value.	0x0200
15:14	Reserved		
16	StagRef	Staggered refresh. 0 - Staggered refresh 1- All banks are refreshed together	0x0
17	ADSFunct	Defines the function of the DAdr[11]/ADS* pin. 0 - ADS* in device accesses & DRAM address [11] in DRAM accesses. 1 - ADS* only	0x0
18	DRAMLatch	Sets the latch operation mode. 0 -The latch control signals are active. 1 - The external data latches are transparent in DRAM accesses when CAS is programmed to be one cycle long	0x0

10.6 DRAM Parameters

DRAM timing parameters, bank width, 32-bit wide bank location, parity support, and refresh support for different DRAM sizes, can be set for each DRAM bank independently. The number of cycles CAS* is active (LOW) in read or write accesses can be programmed to one or two cycles. The number of cycles between the cycle RAS* becomes active and the cycle CAS* becomes active in reads or writes is programmable as well.

DRAM Bank0 Parameters, Offset: 0x44c

Bits	Field name	Function	Initial Value
0	CASWr	The number of cycles CAS* is LOW in a write access. 0 - One cycle 1 - Two cycles	0x1
1	RAStoCASWr	The number of cycles between RAS* going active and CAS* going active in a write access. 0 - Two cycles 1 - Three cycles	0x1
2	CASRd	The number of cycles CAS* is LOW in a read access. 0 - One cycle 1 - Two cycles	0x1
3	RAStoCASRd	The number of cycles between RAS* going active and CAS* going active in a read access. 0 - Two cycles 1 - Three cycles	0x1

Bits	Field name	Function	Initial Value
5:4	Refresh	DRAM type support. 00 - 1/2K Refresh (9 bits row, 9 to 12 bits column) 01 - 1K Refresh (10 bits row, 9 to 12 bits column) 10 - 2K Refresh (11 bits row, 9 to 12 bits column) 11 - 4K Refresh (12 bits row, 9 to 12 bits column)	0x0
6	BankWidth	Width of DRAM bank. 0- 32 (36) bit wide DRAM 1- reserved	0x0
7	BankLoc	Location of a 32-bit wide bank. 0- Even 1- Odd	0x0
8	Parity	Parity support for the bank. 0- No parity support 1- Parity supported	0x0
9	Reserved	Must be Programmed '0'.	0x0

DRAM Bank1 Parameters, Offset: 0x450

Bits	Field Name	Function	Initial Value
8:0	Various	Fields function as in DRAM Bank0.	0xf

DRAM Bank2 Parameters, Offset: 0x454

Bits	Field Name	Function	Initial Value
8:0	Various	Fields function as in DRAM Bank0.	0xf

DRAM Bank3 Parameters, Offset: 0x458

Bits	Field Name	Function	Initial Value
8:0	Various	Fields function as in DRAM Bank0.	0xf

10.7 Device Parameters

Device parameters can be different for each bank. The shape of the different control signals that are active in a device access can be programmed. The access time of the device (in number of cycles) during read accesses should be programmed into the AccToFirst field, to set the time that data from the device will be latched into the external latch. AccToNext should be programmed for the time that data from the device can be latched in consecutive accesses during burst accesses. To prevent bus contention, the TurnOff field specifies the number of cycles (from the deassertion of CSTiming*) to the beginning of the next bus transaction. The write signals pulse should be shaped as well. The parameters specify the number of cycles from the beginning of the cycle to the assertion of the write signals (ADSToWr), the number of cycles the write is active (WrActive), and the number of cycles the write signals are inactive (WrHigh) between consecutive writes in a burst access.

Device width can be programmed to 8-, 16-, or 32-bits (default is 32-bits except for the Boot bank). The device controller can pack data during reads from a word that is less than 32-bits wide. For devices that are less then 32-bits, the device can be located in the odd or the even bank. Devices that are 8-bits or 16-bits wide only support partial reads (up to 64-bits).

Performance when reading from a device can be optimized to save a cycle in each access by making the latch transparent for devices that can supply the data to the GT-64060 on time, without the need to be latched.

Device Bank0 Parameters, Offset: 0x45c

Bits	Field name	Function	Initial Value
2:0	TurnOff	The number of cycles between the deassertion of DevOE* (an externally extracted signal which is the logical OR between CSTiming* and inverted DevRW*) to a new AD bus cycle.	0x7
6:3	AccToFirst	The number of cycles in a read access from the assertion of CS* to the cycle that the data will be latched (by the external latches). Can be extended via the Ready* pin.	0xf
10:7	AccToNext	The number of cycles in a read access from the cycle that the first data was latched to the cycle that the next data will be latched (in burst accesses). Can be extended via the Ready* pin.	0xf
13:11	ADStoWr	The number of cycles from ADS* active to the assertion of EWr* or OWr*.	0x7
16:14	WrActive	The number of cycles EWr* or OWr* are active. Can be extended via the Ready* pin.	0x7
19:17	WrHigh	The number of cycles between deassertion and assertion of EWr* or OWr*.	0x7
21:20	DevWidth	Device width. 00 - 8 bits 01 - 16 bits 10 - 32 bits 11 - 64 bits	0x2
22	Reserved	Must be programmed '1'.	0x1
23	DevLoc	32-bit, 16-bit, or 8-bit device location. 0 - Even bank 1 - Odd bank	0x0
24	Reserved	Read only.	0x0
25	LatchFunct	Latch function in read cycles. 0 - Always transparent 1 - Latch enable signals are active.	0x0
27:26	Reserved	Read only.	0x1
29:28	Reserved	Read only.	0x1
30	Parity	Parity support for the bank. 0- No parity support 1- Parity supported	0x0

Device Bank1 Parameters, Offset: 0x460

Bits	Field Name	Function	Initial Value
30:0	Various	Fields function as in Device Bank0.	0x146fffff

Device Bank2 Parameters, Offset: 0x464

Bits	Field Name	Function	Initial Value
30:0	Various	Fields function as in Device Bank0.	0x146fffff

Device Bank3 Parameters, Offset: 0x468

Bits	Field Name	Function	Initial Value
30:0	Various	Fields function as in Device Bank0.	0x146fffff

Device Boot Bank Parameters, Offset: 0x46c

Bits	Field Name	Function	Initial Value
30:0	Various	Fields function as in Device Bank0.	0x14?fffff

In case of the Boot Bank, bits 23:20 are shown as '?' because bits 21:20 are sampled at reset via DAdr[11:10] to define the width of the boot device.

10.8 DMA Record

Each DMA record includes four registers: byte count, source address, destination address, and a pointer to the next record. The record can be written by the Local Master Port, PCI, or DMA controller in the process of fetching a new record from memory.

Channel 0 DMA Byte Count, Offset: 0x800

Bits	Field Name	Function	Initial Value
15:0	ByteCt	The number of bytes that are left in DMA transfers.	0x0

Channel 1 DMA Byte Count, Offset: 0x804

Bits	Field Name	Function	Initial Value
15:0	ByteCt	The number of bytes that are left in DMA transfers.	0x0

Channel 2 DMA Byte Count, Offset: 0x808

Bits	Field Name	Function	Initial Value
15:0	ByteCt	The number of bytes that are left in DMA transfers.	0x0

Channel 3 DMA Byte Count, Offset: 0x80c

Bits	Field Name	Function	Initial Value
15:0	ByteCt	The number of bytes that are left in DMA transfers.	0x0

Channel 0 DMA Source Address, Offset: 0x810

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address that the DMA controller will read the data from.	0x0

Channel 1 DMA Source Address, Offset: 0x814

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address that the DMA controller will read the data from.	0x0

Channel 2 DMA Source Address, Offset: 0x818

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address that the DMA controller will read the data from.	0x0

Channel 3 DMA Source Address, Offset: 0x81c

Bits	Field Name	Function	Initial Value
31:0	SrcAdd	The address that the DMA controller will read the data from.	0x0

Channel 0 DMA Destination Address, Offset: 0x820

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address that the DMA controller will write the data to.	0x0

Channel 1 DMA Destination Address, Offset: 0x824

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address that the DMA controller will write the data to.	0x0

Channel 2 DMA Destination Address, Offset: 0x828

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address that the DMA controller will write the data to.	0x0

Channel 3 DMA Destination Address, Offset: 0x82c

Bits	Field Name	Function	Initial Value
31:0	DestAdd	The address that the DMA controller will write the data to.	0x0

Channel 0 Next Record Pointer, Offset: 0x830

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list).	0x0

Channel 1 Next Record Pointer, Offset: 0x834

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list).	0x0

Channel 2 Next Record Pointer, Offset: 0x838

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list).	0x0

Channel 3 Next Record Pointer, Offset: 0x83c

Bits	Field Name	Function	Initial Value
31:0	NextRecPtr	The address for the next record of DMA. A value of 0 means a NULL pointer (end of the chained list).	0x0

10.9 DMA Channel Control

Each DMA channel has a control register to set its mode of operation independently of the other three channels. A channel can be programmed to transfer data through the GT-64060. The DMA reads data from the source address (PCI, Devices or DRAM) into an internal 32-byte FIFO. From the internal FIFO, the data is written to a destination address (PCI as master, Devices or DRAM) that can be independent from the source address.

Source addresses and destination addresses can be programmed to increment, decrement, or hold the same value throughout the DMA transfer. For devices that can absorb a limited number of bytes at a time, the channel can be programmed to limit the number of bytes transferred in each DMA cycle. DMA accesses can be initiated by an external source (Demand mode) by asserting one of the four DMAReq[3:0]* pins, or by an internal request (Block mode) until the byte count reaches zero.

All four channels have chaining support via linked lists of records. When the chaining mode is enabled, the DMA controller will fetch the information (the record) for a new DMA transfer directly out of memory (or a device or the PCI) without involving the Local Master Port. The location of the next record is in the Next Record Pointer register (NextRecPtr) and the DMA controller will fetch records every DMA transfer end until it reaches the NULL pointer (NULL pointer is zero).

There are several mechanisms for status and control of the DMA operations. An status interrupt can be programmed to be asserted every time the DMA byte count reaches zero, or only when byte count reaches zero and the record is the

last record in the chain (the NextRecPtr is NULL). In addition, there is a status bit that indicates if a channel is active or not. A channel is active when it is enabled and its byte count is other than zero, or in chained mode when both its byte count is not equal to zero or its NextRecPtr is not equal to NULL, or when it is disabled and its internal FIFO is not empty. A channel can be controlled by disabling it temporarily, and a next record fetch can be forced through Fet-NexRec in Chained mode even if the current DMA has not ended.

Channel 0 Control, Offset: 0x840

Bits	Field name	Function	Initial Value
0	Reserved	Must be '0'	0x0
1	Reserved	Must be '0'	0x0
3:2	SrcDir	Source Direction. 00 - Increment source address 01 - Decrement source address 10 - Hold in the same value	0x0
5:4	DestDir	Destination Direction. 00 - Increment destination address 01 - Decrement destination address 10 - Hold in the same value	0x0
8:6	DatTransLim	Data Transfer Limit in each DMA access. 101 - 1 Byte 110 - 2 Bytes 000 - 4 Bytes 001 - 8 Bytes 011 - 16 Bytes 111 - 32 Bytes	0x0
9	ChainMod	Chained Mode. 0 - Chained mode; when a DMA access is terminated, the parameters of the next DMA access will come from a record in memory that a NextRecPtr register points at. 1 - Non-Chained mode; only the values that are programmed by the Local Master Port (or PCI) directly into the ByteCt, SrcAdd, and DestAdd registers are used.	0x0
10	IntMode	Interrupt Mode. 0 - Interrupt asserted every time the DMA byte count reaches terminal count. 1 - Interrupt every NULL pointer (in Chained mode)	0x0
11	TransMod	Transfer Mode. 0 - Demand 1 - Block	0x0
12	ChanEn	Channel Enable. 0 - Disable 1 - Enable	0x0

Bits	Field name	Function	Initial Value
13	FetNexRec	Fetch Next Record. 1 - Forces a fetch of the next record (even if the current DMA has not ended). This bit is reset after fetch is completed (meaningful only in Chained mode).	0x0
14	DMAActSt	DMA Activity Status (read only). 0 - Channel is not active 1 - Channel is active	0x0

Channel 1 Control, Offset: 0x844

Bits	Field Name	Function	Initial Value
14:0	Various	Fields function as in Channel 0 Control.	0x0

Channel 2 Control, Offset: 0x848

Bits	Field Name	Function	Initial Value
14:0	Various	Fields function as in Channel 0 Control.	0x0

Channel 3 Control, Offset: 0x84c

Bits	Field Name	Function	Initial Value
14:0	Various	Fields function as in Channel 0 Control.	0x0

10.9.1 DMA Programming Notes

10.9.1.1 Non-Chained mode

In this mode, Source, Destination and Byte Count should be initialized prior to enabling the channel. ChainMod should be set to '1'.

10.9.1.2 Chained mode

All the channel record's parameters for the current transaction (Source, Byte Count, Destination and Next Record Pointer) should be initialized in memory devices or PCI devices. The address of the first record should be initialized by writing it to the NextRecPtr of the channel. The channel should be enabled via ChanEn=1, the FetNexRec should be set to '1' and the ChainMod should be set to '0'.

10.9.1.3 Restarting a disabled channel

In Non-Chained mode, ChanEn should be set to '1'.

In Chained mode, the software should find out if the first fetch took place. If it did, only ChanEn should be set to '1'. If it did not, the FetNexRec should also be set to '1'.

10.9.1.4 Reprogramming an active channel

The channel should first be disabled via ChanEn=0. Then it must be assured that the channel is no longer active (for example by polling the DMAActSt of the channel). New DMA parameters should be programmed prior to re-enabling

the channel via ChanEn=1.

10.10 DMA Arbiter

The DMA controller has a programmable arbitration scheme between its four channels. The channels are grouped into two groups, one group includes channel 0 and 1, and the other group includes channels 2 and 3. The channels in each group can be programmed to have priority so that a selected channel has the higher priority, or to have the same priority in round robin. The priority between the two groups can be programmed in a similar way so that a selected group has a higher priority, or to have the same priority in round robin. The priority scheme has additional flexibility with the programmable Priority Option. With the Priority Option the DMA bandwidth allocation can be divided in a fairer way. For example, if the PrioOpt bit is set to '0' and the PrioGrps field is set as '10', the requesting devices will get the DMA in the order 0,1,2,0,1,3,0,1,2,0,1,3,..... (assuming that PrioChan1/0 and PrioChan3/2 are set to round robin), while if the PrioOpt bit is set to '1' the requesting devices will get the DMA in the order 0,1,0,1,0,1,2,3,2,3,2,3,..... The DMA arbiter control register can be reprogrammed any time regardless of the channels' status (active or not active).

Some arbitration examples follow to facilitate the understanding of this register:

1. Assuming all 4 channels are requested all the time,
   with Arbiter Control register = 0x40, the order will be: 0,2,1,3,0,2,1,3,.....
   with Arbiter Control register = 0x0, the order will be: 0,1,2,3,0,1,2,3,.....
2. Assuming 3 channels are requested (0,1,2),
   with Arbiter Control register = 0x40, the order will be: 0,2,1,2,0,2,1,2,.....
   with Arbiter Control register = 0x0, the order will be: 0,1,2,0,1,2,0,1,2,.....
3. Assuming all 4 channels are requested,
   with Arbiter Control register = 0x45, the order will be: 1,3,1,3,1,3,.....,0,2,0,2,0,2,.....
   with Arbiter Control register = 0x5, the order will be: 1,0,3,2,1,0,3,2,1,0,3,2,.....
4. Assuming 3 channels are requested (0,1,2),
   with Arbiter Control register = 0x45, the order will be: 1,2,1,2,1,2,.....,0,0,0,0,0,0,.....
   with Arbiter Control register = 0x5, the order will be: 0,1,2,0,1,2,.....
5. Assuming all 4 channels are requested,
   with Arbiter Control register = 0x55, the order will be: 3,3,3,3,3,3,2,2,2,2,2,1,1,1,1,0,0,0,.....
   with Arbiter Control register = 0x15, the order will be: 3,2,1,3,2,0,3,2,1,3,2,0,.....
6. Assuming 3 channels are requested (0,1,2),
   with Arbiter Control register = 0x55, the order will be: 2,2,2,2,1,1,1,1,0,0,0,0,.....
   with Arbiter Control register = 0x15, the order will be: 2,1,2,0,2,1,2,0,.....
7. Assuming 3 channels are requested (0,2,3),
   with Arbiter Control register = 0x55, the order will be: 3,3,3,...,2,2,2,...,0,0,0,.....
   with Arbiter Control register = 0x15, the order will be: 3,2,0,3,2,0,3,2,0,.....

10.11 Arbiter Control, Offset: 0x860

Bits	Field name	Function	Initial Value
1:0	PrioChan1/0	Priority between Channel 0 and Channel 1. 00 - Round Robin 01 - Priority to channel 1 over channel 0 10 - Priority to channel 0 over channel 1 11 - Reserved	0x0
3:2	PrioChan3/2	Priority between Channel 2 and Channel 3. 00 - Round Robin 01 - Priority to channel 3 over 2 10 - Priority to channel 2 over 3 11 - Reserved	0x0
5:4	PrioGrps	Priority between the group of channels 0/1 and the group of channels 2/3. 00 - Round Robin 01 - Priority to channels 2/3 over 0/1 10 - Priority to channels 0/1 over 2/3 11 - Reserved	0x0
6	PrioOpt	Defines the arbiter behavior for high priority device. 0 - High priority device will relinquish the bus for a requesting device for one DMA transaction after it was serviced. 1 - High priority device will be granted as long as it requests the bus.	0x0

10.12 Timer / Counter

There are three 24-bit wide and one 32-bit wide timer/counter on the GT-64060. Each one can be selected to operate as a timer or as a counter. In Counter mode, the counter will count down to terminal count, will stop and issue an interrupt. In Timer mode, it will count down, will issue an interrupt on terminal count, and will reload itself to the programmed value and continue to count. Reads from the counter or timer are done from the counter itself, while writes are to its register. For example, note that even though the registers are programmed to an initial value of '0' the counters will read 0xffffff. In order to reprogram a timer/counter, it should first be disabled, then it should be loaded with a new value and after that it should be enabled as appropriate (counter or timer).

Timer/Counter 0, Offset: 0x850

Bits	Field Name	Function	Initial Value
31:0	TC0Value	The counter or timer initial value.	0x0

Timer/Counter 1, Offset: 0x854

Bits	Field Name	Function	Initial Value
23:0	TC1Value	The counter or timer initial value.	0x0

Timer/Counter 2, Offset: 0x858

Bits	Field Name	Function	Initial Value
23:0	TC2Value	The counter or timer initial value.	0x0

Timer/Counter 3, Offset: 0x85c

Bits	Field Name	Function	Initial Value
23:0	TC3Value	The counter or timer initial value.	0x0

Timer/Counter Control, Offset: 0x864

Bits	Field name	Function	Initial Value
0	EnTC0	The timer/counter will count only when it is enabled. 0 - Disable 1 - Enable	0x0
1	SelTC0	Timer or counter selection. 0 - Counter 1 - Timer	0x0
2	EnTC1	The timer/counter will count only when it is enabled. 0 - Disable 1 - Enable	0x0
3	SelTC1	Timer or counter selection. 0 - Counter 1 - Timer	0x0
4	EnTC2	The timer/counter will count only when it is enabled. 0 - Disable 1 - Enable	0x0
5	SelTC2	Timer or counter selection . 0 - Counter 1 - Timer	0x0
6	EnTC3	The timer/counter will count only when it is enabled. 0 - Disable 1 - Enable	0x0
7	SelTC3	Timer or counter selection. 0 - Counter 1 - Timer	0x0

10.13 PCI Internal

The PCI internal registers include the following functions: byte swapping and clock ratios, timing for retries, DRAM and Device bank sizes, interrupt acknowledge, system error masking, configuration address, and configuration data.

Command, Offset: 0xc00

Bits	Field name	Function	Initial Value
0	ByteSwap	When set to zero, the GT-64060 swaps the incoming and outgoing PCI data.	Set to the same value sampled at reset into bit[12] of the Local Master Port Interface Configuration register.
2:1	SyncMode	Indicates the ratio between TClk and PClk as follows: 00 - When the PClk ranges from DC to 33MHz (default mode, works in all cases; use following settings for higher performance) 01 - When PClk is HIGHER than half the TClk frequency (e.g. when TClk is 50MHz, SyncMode can be set to 01 if the PCI frequency is HIGHER than 25 MHz). 1x - When the two clocks are synchronized and PClk is HIGHER than half of TCLK (e.g. TClk = 50MHz, PClk = 25MHz)	0x0

Time Out & ReTry, Offset: 0xc04

Bits	Field name	Function	Initial Value
7:0	Timeout0	Specifies in PCI clock units the number of clocks the GT-64060, as a slave, holds the PCI bus before the generation of retry termination. Used for the first data transfer.	0x0f
15:8	Timeout1	Specifies in PCI clock units the number of clocks the GT-64060, as a slave, holds the PCI bus before the generation of disconnect termination. Used for data transfers following the first data.The number of PCI clock cycles between the last Trdy* rise and the Stop* falling are n+1, where 'n' is the Timeout1 value.	0x07
23:16	RetryCtr	Specifies the number of retries of the GT-64060 Master. The GT-64060 generates an interrupt when this timer expires. A value of 0x00 means "retry forever". The number in RetryCtr does not include the first access of the transaction.	0x00

RAS[1:0] Bank Size, Offset: 0xc08

Bits	Field Name	Function	Initial Value
31:12	BankSize	Specifies the RAS[1:0] address mapping in conjunction with the RAS[1:0] Base Address register. Set to '0' indicates that the corresponding bit in the address and in the base address must be equal in order to have a hit. Set to '1' indicates that the corresponding bit in the address is a don't-care. For example, bit 12 set to '1' indicates that the RAS[1:0] size is 8KBytes (address bits [12:0] are changeable/don't-care). The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).	0x00fff

RAS[3:2] Bank Size, Offset: 0xc0c

Bits	Field Name	Function	Initial Value
31:12	BankSize	Specifies the RAS[3:2] address mapping in conjunction with the RAS[3:2] Base Address register. Set to '0' indicates that the corresponding bit in the address and in the base address must be equal in order to have a hit. Set to '1' indicates that the corresponding bit in the address is a don't-care. For example, bit 12 set to '1' indicates that the RAS[3:2] size is 8KBytes (address bits [12:0] are changeable/don't-care). The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).	0x00fff

CS[2:0] Bank Size, Offset: 0xc10

Bits	Field Name	Function	Initial Value
31:12	BankSize	Specifies the CS[2:0] address mapping in conjunction with the CS[2:0] Base Address register. Set to '0' indicates that the corresponding bit in the address and in the base address must be equal in order to have a hit. Set to '1' indicates that the corresponding bit in the address is a don't-care. For example, bit 12 set to '1' indicates that the CS[2:0] size is 8KBytes (address bits [12:0] are changeable/don't-care). The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).	0x01fff

CS[3] and Boot CS Bank Size, Offset: 0xc14

Bits	Field Name	Function	Initial Value
31:12	BankSize	Specifies the CS[3] and Boot CS address mapping in conjunction with the CS[3] and Boot CS Base Address register. Set to '0' indicates that the corresponding bit in the address and in the base address must be equal in order to have a hit. Set to '1' indicates that the corresponding bit in the address is a don't-care. For example, bit 12 set to '1' indicates that the CS[3]/ Boot CS size is 8KBytes (address bits [12:0] are changeable/don't-care). The set bits in the Bank Size must be sequential (e.g. 000...001, 000...011, 000...111 are correct values, whereas 000...010 and 000...100 are not).	0x00fff

SErr Mask, Offset: 0xc28

Bits	Field Name	Function	Initial Value
0	AddrErr	Mask bit. When set, SErr* is asserted when the GT-64060 detects a parity error on the address lines.	0x0
1	MasWrErr	Mask bit. When set, SErr* is asserted when the GT-64060 detects a parity error during a master write operation.	0x0
2	MasRdErr	Mask bit. When set, SErr* is asserted when the GT-64060 detects a parity error during a master read operation.	0x0
3	MemErr	Mask bit. When set, SErr* is asserted when a memory parity error has been detected (applicable only when an external parity checking device is used).	0x0
4	MasAbort	Mask bit. When set, SErr* is asserted when the GT-64060 performs master abort.	0x0
5	TarAbort	Mask bit. When set, SErr* is asserted when the GT-64060 detects a target abort.	0x0

Interrupt Acknowledge, Offset: 0xc34

Bits	Field Name	Function	Initial Value
31:0	IntAck	The data is meaningless. A Local Master Port read operation to this register causes the GT-64060 to perform an Interrupt Acknowledge cycle on the PCI bus.	0x00000000

Base Address Registers' Enable, Offset: 0xc3c

Bits	Field name	Function	Initial Value
8	RAS[1:0]En	Controls address matching with RAS[1:0] base/size. 0 - Enable 1 - Disable	0x0
7	RAS[3:2]En	Controls address matching with RAS[3:2] base/size. 0 - Enable 1 - Disable	Sampled at Reset via DMAReq[0]*/Ready*
6	CS[2:0]En	Controls address matching with CS[2:0] base/size. 0 - Enable 1 - Disable	Sampled at Reset via DAdr[6]
5	CS[3] & Boot CSEn	Controls address matching with CS[3] & Boot CS base/size. 0 - Enable 1 - Disable	Sampled at Reset via DAdr[3]
4	IntMeMEn	Controls address matching with internal registers-Memory mapped base/size. 0 - Enable 1 - Disable	0x0
3	IntIOEn	Controls address matching with internal registers I/O mapped base/size. 0 - Enable 1 - Disable	Sampled at Reset via DMAReq[1]*/ParErr*
2	SwRAS[1:0]En	Controls address matching with Swapped RAS[1:0] base/size. 0 - Enable 1 - Disable	Sampled at Reset via DAdr[2]
1	SwRAS[3:2]En	Controls address matching with Swapped RAS[3:2] base/size. 0 - Enable 1 - Disable	Sampled at Reset via DAdr[1]
0	SwCS[3] & Boot CSEn	Controls address matching with Swapped CS[3] & Boot CS base/size. 0 - Enable 1 - Disable	Sampled at Reset via DAdr[0]

The GT-64060 prevents disabling both memory mapped base/size address matching and I/O mapped base/size address matching at the same time (bits 3 and 4 cannot simultaneously be set to 1).

If a base address register is disabled (corresponding bit is set to 1), a configuration read from this base address register will report a 0, and not its real value (as read from a reserved/non-implemented register).

Configuration Address, Offset: 0xcf8

Bits	Field Name	Function	Initial Value
7:2	RegNum	Indicates the register number.	0x00
10:8	FunctNum	Indicates the function type.	0x0
15:11	DevNum	Indicates the device number.	0x00

Bits	Field Name	Function	Initial Value
23:16	BusNum	Indicates the bus number.	0x00
31	ConfigEn	When set, an access to the Configuration Data register is translated into a Configuration or Special cycle on the PCI bus.	0x0

Configuration Data, Offset: 0xcfc

Bits	Field Name	Function	Initial Value
31:0	Config	The data is transferred to/from the PCI bus when the Local Master Port accesses this register and the ConfigEn bit in the Configuration Address register is set. A Local Master Port access to this register causes the GT-64060 to perform a Configuration or Special cycle on the PCI bus.	0x000

10.14 Interrupts

IntSum in the Interrupt Cause register is the logical OR of bits[29:1], regardless of the Mask registers' values. This is in order to be notified via polling if any interrupt occurred within the GT-64060. Therefore, bit[0] of both the Local Master Port Mask and PCI Mask registers is read-only '0'.

LMPIntSum in the Interrupt Cause register is the logical OR of bits[29:26,20:1], masked by bits[29:26,20:1] of the LMP Mask register. Therefore, bits[25:21] of the LMP Mask register, being non-relevant to interrupts directed to the Local Master Port, are read-only '0'. Also bits[31:30], being summaries, are read-only '0'.

PCIIntSum in the Interrupt Cause register is the logical OR of bits[25:1], masked by bits[25:1] of the PCI Mask register. Therefore, bits[29:26] of the PCI Mask register, being non-relevant to interrupts directed to the PCI, are read-only '0'. Also bits[31:30], being summaries, are read-only '0'.

Interrupt Cause, Offset: 0xc18 *(all bits are cleared by writing a value of '0' by the Local Master Port or PCI, unless stated otherwise)*

Bits	Field Name	Function	Initial Value
0	IntSum	Interrupt summary. Logical OR of all the interrupt bits, regardless of the Mask registers' values.	0x0 Read only
1	MemOut	Asserts when the Local Master Port accesses an address out of range in the memory decoding or a burst access to 8-/16-bit devices.	0x0
2	DMAOut	Asserts when the DMA accesses an address out of range.	0x0
3	LMPOut	Asserts when the Local Master Port accesses an address out of range.	0x0
4	DMA0Comp	Asserts at completion of DMA Channel 0 transfer.	0x0
5	DMA1Comp	Asserts at completion of DMA Channel 1 transfer.	0x0
6	DMA2Comp	Asserts at completion of DMA Channel 2 transfer.	0x0
7	DMA3Comp	Asserts at completion of DMA Channel 3 transfer.	0x0
8	T0Exp	Asserts when Timer 0 expires.	0x0
9	T1Exp	Asserts when Timer 1 expires.	0x0
10	T2Exp	Asserts when Timer 2 expires.	0x0

Bits	Field Name	Function	Initial Value
11	T3Exp	Asserts when Timer 3 expires.	0x0
12	MasRdErr	Asserts when the GT-64060 detects a parity error during a master read operation.	0x0
13	SlvWrErr	Asserts when the GT-64060 detects a parity error during a slave write operation.	0x0
14	MasWrErr	Asserts when the GT-64060 detects a parity error during a master write operation.	0x0
15	SlvRdErr	Asserts when the GT-64060 detects a parity error during a slave read operation.	0x0
16	AddrErr	Asserts when the GT-64060 detects a parity error on the address lines.	0x0
17	MemErr	Asserts when a memory parity error is detected. Applicable only when an external parity checking device is used.	0x0
18	MasAbort	Asserts upon master abort.	0x0
19	TarAbort	Asserts upon target abort.	0x0
20	RetryCtr	Asserts when the retry counter expires.	0x0
25:21	LMPInt	These bits are set by the Local Master Port by writing '0' to generate an interrupt on the PCI bus. They are cleared when the PCI writes '0'.	0x0
29:26	PCIInt	These bits are set by the PCI by writing '0' to generate an interrupt on the Local Master Port. They are cleared when the Local Master Port writes '0'.	0x0
30	LMPIntSum	Interrupt summary. Logical OR of bits[29:26,20:1], masked by bits[29:26,20:1] of the LMP Mask register.	0x0
31	PCIIntSum	Interrupt summary. Logical OR of bits[25:1], masked by bits[25:1] of the PCI Mask register.	0x0

LMP Mask, Offset: 0xc1c

Bits	Field Name	Function	Initial Value
31:0	LMPMask	Mask to the Local Master Port interrupt line for the appropriate bits in the Interrupt Cause register. Bits 0, 25:21, 31:30 are read-only "0".	0x00000000

PCI Mask, Offset: 0xc24

Bits	Field Name	Function	Initial Value
31:0	PCIMask	Mask to the PCI interrupt line for the appropriate bits in the Interrupt Cause register. Bits 0, 31:26 are read-only "0".	0x00000000

Device and Vendor ID, Offset: 0x000

Bits	Field name	Function	Initial Value
31:16	DevID	Provides the unique GT-64060 ID number (0x146).	0x4146
15:0	VenID	Provides the manufacturer of the GT-64060 (0x11ab).	0x11ab

Status and Command, Offset: 0x004

Bits	Field name	Function	Initial Value
0	IOEn	Controls the GT-64060's response to I/O accesses. 0 - Disable 1 - Enable	0x0
1	MEMEn	Controls the GT-64060's response to Memory accesses. 0 - Disable 1 - Enable	0x0
2	MasEn	Controls the GT-64060's ability to act as a master on the PCI bus. 0 - Disable 1 - Enable	0x0
4	MemWrInv	Controls the GT-64060's ability to generate Memory Write & Invalidate command on the PCI bus. 0 - Disable 1 - Enable	0x0
6	PErrEn	Controls the GT-64060's ability to respond to parity errors on the PCI by asserting the PErr* pin. 0 - Disable 1 - Enable	0x0
8	SErrEn	Controls the GT-64060's ability to assert the SErr* pin. 0 - Disable 1 - Enable	0x0
23	TarFastBB	Read only bit. Indicates that the GT-64060 is capable of accepting fast back-to-back transactions on the PCI bus.	0x1
24	DataParDet	This bit is set by the GT-64060 when it detects a data parity error during master operation.	0x0
27:25	DevSelTim	These pins indicate the GT-64060 's DevSel timing (medium), per the PCI standard.	0x1 Read only
28	TarAbort	This bit is set upon Target Abort.	0x0
29	MasAbort	This pin is set upon Master Abort.	0x0
30	SysErr	This pin is set upon System Error.	0x0
31	DetParErr	This pin is set upon detection of Parity error (in both, master and slave operations).	0x0

Class Code and Revision ID, Offset: 0x008

Bits	Field name	Function	Initial Value
7:0	RevID	Indicates the GT-64060 Revision number.	0x01
31:24	BaseClass	Indicates the GT-64060 Base Class (0x6 - bridge device).	0x06
23:16	SubClass	Indicates the GT-64060 Subclass (0x0 - host bridge).	0x00

BIST, Header Type, Latency Timer, Cache Line, Offset: 0x00c

Bits	Field name	Function	Initial Value
7:0	CacheLine	Specifies the GT-64060's cache line size	0x00
15:8	LatTimer	Specifies in units of PCI bus clocks the value of the latency timer of the GT-64060.	0x00
23:16	HeadType	Specifies the layout of bytes 10h through 3fh.	0x00
31:24	BIST	Built In Self Test, Reserved	0x00

The BIST Field is reserved and is hardwired to 0.

Device and Vendor ID (0x000), Class Code and Revision ID (0x008), and Header Type (0x00e) fields are ready only from the PCI bus. These fields can be modified and read via the Local Master Port bus.

For more information on these fields, please refer to the PCI specification.

Access of PCI masters to DRAM banks, Devices and internal space is achieved once there is a match between the address presented over the PCI bus and the space defined by the respective Base/Size register pair. The GT-64060 incorporates three Swapped Base Address registers for RAS[1:0], RAS[3:2] and CS[3] & BootCS. When the address matches a Swapped Base Address register (and of course should not match its respective non-Swap Base Address register), the data transferred will undergo the opposite to what is indicated by the ByteSwap bit (bit[0] of 0xc00). e.g. using this mechanism, one could write data directly to DRAM and read it byte-swapped without Local Master Port processing.

The Size registers could not define a zero size space. In order to enable the system designer to use addresses which are within a certain space without having the GT-64060 respond to these addresses, a Base Address Enable register is incorporated. A disabled space will not trigger device response should the address fall within the space defined by its Base/Size register pair.

RAS[1:0] Base Address, Offset: 0x010

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of RAS[1:0] (see RAS[1:0] Bank Size).	0x00000

RAS[3:2] Base Address, Offset: 0x014

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of RAS[3:2] (see RAS[3:2] Bank Size).	0x01000

CS[2:0] Base Address, Offset: 0x018

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of CS[2:0] (see CS[2:0] Bank Size).	0x1c000

CS[3] and Boot CS Base Address, Offset: 0x01c

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of CS[3] and Boot CS (see CS[3] and Boot CS Bank Size).	0x1f000

Internal Registers Memory Mapped Base Address, Offset: 0x020

Bits	Field Name	Function	Initial Value
31:12	MemMapBase	Defines the address assignment of the GT-64060's internal registers.	0x14000

Subsystem Device and Vendor ID, Offset: 0x02c

Bits	Field name	Function	Initial Value
31:16	DevID	Provides the unique GT-64060 ID number (0x146).	0x0146
15:0	VenID	Provides the manufacturer of the GT-64060 (0x11ab).	0x11ab

Internal Registers I/O Mapped Base Address, Offset: 0x024

Bits	Field Name	Function	Initial Value
31:12	IOMapBase	Defines the address assignment of the GT-64060's internal registers.	0x14000

Expansion ROM Base Address Register, Offset: 0x030

Bits	Field Name	Function	Initial Value
31:12	ERBase	Defines the address of the expansion ROM memory space region assigned to the GT-64060. This is where the expansion ROM code will appear in system memory when bit 0 of this register contains a value of 1 and bit 1 of this device's Command Register contains a value of 1.	0x1f000
0	ERDecEn	Expansion ROM Decode Enable 0 - Disable 1 - Enable	0x0

Interrupt Pin and Line, Offset: 0x03c

Bits	Field name	Function	Initial Value
7:0	IntLine	Provides interrupt line routing information.	0x00
15:8	IntPin	Indicates which interrupt pin is used by the GT-64060. The GT-64060 uses INTA.	0x01

10.14.1 FUNCTION 1 CONFIGURATION REGISTERS

If one or more of the following Swap Base Address Registers is enabled after Rst* (see Reset Configuration), the GT-64060 will be configured as a multi-function device (bit 7 in the Header Type register set to 1, 0xe). To access any of the Swap Base Address Registers, a configuration access addressed to function #1 should be used with the appropriate offset. If a different offset other than 0x010, 0x014, or 0x01c is accessed when specifying function #1, the transaction will access the corresponding offset register in function #0. Configuration transactions to any other function number will be ignored.

If none of the Swap Base Address Registers are enabled after Rst*, the GT-64060 will be configured as a single function device (bit 7 in the Header Type register set to 0, 0xe) and the function #1 configuration registers will be unaccessable.

Function 1 RAS[1:0] Swapped Base Address, Offset: 0x010

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of swapped RAS[1:0] (see RAS[1:0] Bank Size).	0x0000

Function 1 CAS[3:2] Swapped Base Address, Offset: 0x014

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of swapped RAS[3:2] (see RAS[3:2] Bank Size).	0x01000

Function 1 CS[3] and Boot CS Swapped Base Address, Offset: 0x01c

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of swapped CS[3] and Boot CS (see CS[3] and Boot CS Bank Size).	0x1f000

For more information on these fields, please refer to the PCI specification.

11. RESTRICTIONS

11.1 Local Master Port Interface

a) The Local Master Port should not attempt an access before 10 TClk cycles following deassertion of Rst* have expired.

b) The Local Master Port Interface supports only DDDDDDDD and DXDXDXDXDXDXDXDX write patterns.

c) A PCI I/O read intended for synchronization barrier should not be more than one long word (4 bytes). Any PCI I/O read of more than 4 bytes will be carried out without checking the internal FIFOs.

d) A write of more than 4 bytes to internal space will be ignored. A read of more than 4 bytes to internal space will result in transaction termination with bus-error indication (MasCmd[5] equal '1').

11.2 Memory Interface

a) Unless the boot device is 32-bits wide, the boot will be on the even bank.

b) All Device Parameters (section 3.7) must be greater or equal to 3. i.e., AccToFirst, AccToNext, ADSToWr, WrActive and WrHigh.

c) When working with an 8- or 16-bit bus from Local Master Port, a read/write operation can not exceed 64-bits (8 bytes).

d) When working with an 8- or 16-bit bus from DMA/PCI, a read/write operation can't exceed 32-bits (4 bytes).

e) When an erroneous address is issued or a burst operation is performed to an 8- or 16-bit device, the GT-64060 forces an interrupt (unless masked). If a sequence of address misses occurs, there will be no other interrupt prior to resetting the appropriate bit in the cause register and no new address will be registered in the Address Decode Error register (0x470) prior to reading it.

f) When the Local Master Port reads from an address which is decoded in the Local Master Port Interface Unit as being a hit for CS[2:0]* or CS[4:3]* and decoded as a miss in the DRAM/Device Interface Unit, the cycle will complete only if Ready* is asserted (i.e., driven low). Although being a result of improper and inconsistent programming of the address space defining registers, the following 2 workarounds exist:

- Ready* should always be asserted (low) when CSTiming* is inactive (high).
- If the Ready* signal is not needed in the system, the DMAReq[0]/Ready* pin should either be programmed as Ready* and constantly driven active (low) or be programmed as DMAReq[0]*.

11.3 PCI Interface

Note: No PCI access should be attempted before 6 PClk cycles following deassertion of Rst* have expired.

11.3.1 PCI Master

a) Latency count, as specified in LatTimer, should not be programmed to less than 6.

11.3.2 PCI Slave

a) The set bits in the Bank Size registers must be sequential.

b) When the slave is locked, in order to prevent a deadlock, all transactions to internal registers (I/O or memory cycles) are not supported (retry will be issued).

c) All access to memory are prefetchable and therefore should not be regarded as non-destructive.

11.3.3 PCI Cache Line Size

If the cache line size field (0x00c) is equal to 0 (default), the GT-64060 as a PCI Master will not initiate "Memory Read Line" and "Memory Write & Invalidate" transactions.

If the cache line size field is equal to 7, the GT-64060 will interpret this as an 8 word cache line. As a PCI Master, it will intiate "Memory Read Line" and "Memory Write & Invalidate" transactions if there are 8 words in the transactions.

If the cache line size field is not set to 0 or 7, the GT-64060 will behave unpredictably.

11.4 DMA

a) Transfers of less than 4 bytes are not supported.

b) In order to restart a channel after it has been enabled, it should first be checked that the DMAActSt bit is set to NOT ACTIVE (see section 3.9).

c) When Source or Destination address is decremented, both addresses should be word-aligned (that is, A1 and A0 should be both zero), and Byte Count should be a multiple of 4 (this applies for burst limits greater than 4 bytes).

d) When burst limit is less than or equal to 4 bytes, no support in decrement.

e) When using the address hold option in the source direction (SrcDir in section 3.9), the source address should keep the following rules:

- word-aligned if burst limit is greater or equal to 4 bytes.
- bits [1:0] equal to 00, 01 or 10 if burst limit is equal to 2 bytes.
- no restriction for burst limit equal to 1 byte.

f) When using the address hold option in the destination direction (DestDir in section 3.9), the following rules should be kept:

- both Source and Destination addresses should be word-aligned if burst limit is greater or equal to 4 bytes.
- bit [0] of both Source and Destination addresses should be equal to 0 if burst limit is equal to 2 bytes.
- no restriction for burst limit equal to 1 byte.

g) Fly-by is not supported.

h) Records' addresses should be a multiple of 16.

12. PINOUT TABLE, 208 pin PQFP (sorted by number)

Pin #	Signal Name	Pin #	Signal Name	Pin #	Signal Name
1	VDD	36	PAD[15]	71	MasCmd[1]
2	PAD[27]	37	PAD[14]	72	MasCmd[0]
3	PAD[26]	38	VSS	73	MasAD[0]
4	PAD[25]	39	PAD[13]	74	MasAD[1]
5	PAD[24]	40	PAD[12]	75	MasAD[2]
6	CBE[3]*	41	PAD[11]	76	MasAD[3]
7	IDSel	42	PAD[10]	77	MasAD[4]
8	VSS	43	PAD[9]	78	MasAD[5]
9	VDD	44	VSS	79	VSS
10	PAD[23]	45	VDD	80	VRef
11	PAD[22]	46	PAD[8]	81	VDD
12	PAD[21]	47	CBE[0]*	82	MasAD[6]
13	PAD[20]	48	PAD[7]	83	MasAD[7]
14	PAD[19]	49	PAD[6]	84	MasAD[8]
15	VSS	50	PAD[5]	85	MasAD[9]
16	VDD	51	VSS	86	MasAD[10]
17	VSS	52	VDD	87	MasAD[11]
18	PAD[18]	53	PAD[4]	88	MasAD[12]
19	PAD[17]	54	PAD[3]	89	MasAD[13]
20	PAD[16]	55	PAD[2]	90	VSS
21	CBE[2]*	56	PAD[1]	91	TClk
22	Frame*	57	PAD[0]	92	VDD
23	VSS	58	VSS	93	MasAD[14]
24	VDD	59	Should be tied to VDD	94	MasAD[15]
25	IRdy*	60	DMAReq[3]*	95	MasAD[16]
26	TRdy*	61	Interrupt*	96	MasAD[17]
27	DevSel*	62	MasCmd[8]	97	MasAD[18]
28	Stop*	63	MasCmd[7]	98	VSS
29	Lock*	64	MasCmd[6]	99	VDD
30	PErr*	65	MasCmd[5]	100	MasAD[19]
31	VSS	66	MasCmd[4]	101	MasAD[20]
32	VDD	67	VSS	102	MasAD[21]
33	SErr*	68	VDD	103	MasAD[22]
34	Par	69	MasCmd[3]	104	MasAD[23]
35	CBE[1]*	70	MasCmd[2]	105	MasAD[24]

Pin #	Signal Name	Pin #	Signal Name	Pin #	Signal Name
106	MasAD[25]	141	AD[23]	176	OCAS[2]*
107	MasAD[26]	142	AD[22]	177	OCAS[1]*
108	MasAD[27]	143	VSS	178	OCAS[0]*
109	MasAD[28]	144	VDD	179	RAS[3]*
110	VSS	145	AD[21]	180	RAS[2]*
111	VDD	146	AD[20]	181	RAS[1]*
112	MasAD[29]	147	AD[19]	182	RAS[0]*
113	MasAD[30]	148	AD[18]	183	DAdr[11]/ADS*
114	MasAD[31]	149	AD[17]	184	DAdr[10]/OWr[3]*
115	ValidOut*	150	AD[16]	185	VSS
116	ValidIn*	151	AD[15]	186	DAdr[9]/OWr[2]*
117	WrRdy*	152	AD[14]	187	DAdr[8]/OWr[1]*
118	Release*	153	AD[13]	188	DAdr[7]/OWr[0]*
119	DMAReq[0]*/Ready*	154	VSS	189	DAdr[6]/EWr[3]*
120	DMAReq[1]*/ParErr*	155	AD[12]	190	DAdr[5]/EWr[2]*
121	LEAdrE/DMAReq[2]*	156	AD[11]	191	DAdr[4]/EWr[1]*
122	LEAdrO	157	AD[10]	192	DAdr[3]/EWr[0]*
123	OEB	158	AD[9]	193	DAdr[2]/BAdr[2]
124	OEE*	159	AD[8]	194	DAdr[1]/BAdr[1]
125	OEO*	160	AD[7]	195	DAdr[0]/BAdr[0]
126	VSS	161	AD[6]	196	Int*
127	LEE	162	AD[5]	197	Rst*
128	LEO	163	AD[4]	198	VDD
129	ALE	164	AD[3]	199	PClk
130	CSTiming*	165	VSS	200	VSS
131	AD[31]/CS[3]*	166	AD[2]	201	Gnt*
132	AD[30]/CS[2]*	167	AD[1]/DevRW*	202	Req*
133	AD[29]/CS[1]*	168	AD[0]/BootCS*	203	VSS
134	AD[28]/CS[0]*	169	DWr*	204	PAD[31]
135	AD[27]/DMAAck[3]*	170	ECAS[3]*	205	PAD[30]
136	AD[26]/DMAAck[2]*	171	ECAS[2]*	206	PAD[29]
137	AD[25]/DMAAck[1]*	172	ECAS[1]*	207	PAD[28]
138	VSS	173	ECAS[0]*	208	VSS
139	VDD	174	VDD		
140	AD[24]/DMAAck[0]*	175	OCAS[3]*		

13. DC CHARACTERISTICS - *PRELIMINARY/SUBJECT TO CHANGE*

NOTICE: Specs are TARGETS

13.1 Absolute Maximum Ratings

Symbol	Parameter	Min.	Max.	Unit
Vdd	Supply Voltage	-0.3	6.5	V
Vi	Input Voltage	-0.3	Vdd+0.3	V
Vo	Output Voltage	-0.3	Vdd+0.3	V
Io	Output Current		24	mA
Iik	Input Protect Diode Current		+-20	mA
Iok	Output Protect Diode Current		+-20	mA
Tc	Operating Case Temperature	0	70	C
Tstg	Storage Temperature	-40	125	C
ESD			2000	V

13.2 Recommended Operating Conditions

Symbol	Parameter	Min.	Typ.	Max.	Unit
Vdd	Supply Voltage	4.75		5.25	V
Vi	Input Voltage	0		Vdd	V
Vo	Output Voltage	0		Vdd	V
Tc	Operating Case Temperature	0		70	C
Cin	Input Capacitance		7.2		pF
Cout	Output Capacitance		7.2		pF

13.3 DC Electrical Characteristics Over Operating Range

(Tc=0-70°C; Vdd=+5V, +/-5%)

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Vih	Input HIGH level	Guaranteed Logic HIGH level	2.0		Vdd + 0.5V	V
Vil	Input LOW level	Guaranteed Logic LOW level	-0.5		0.8	V
Voh	Output HIGH Voltage	IoH = 2 mA IoH = 4 mA IoH = 8 mA IoH = 12 mA IoH = 16 mA IoH = 24 mA	2.4			V

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Vol	Output LOW Voltage	IoL = 2 mA IoL = 4 mA IoL = 8 mA IoL = 12 mA IoL = 16 mA IoL = 24 mA			0.4	V
Iih	Input HIGH Current				+-1	uA
Iil	Input LOW Current				+-1	uA
Iozh	High Impedance Output Current				+-1	uA
Iozl	High Impedance Output Current				+-1	uA
Vh	Input Hysteresis		TBD	TBD	TBD	mV
Icc	Operating Current				400	mA

NOTE: Pullup/Pulldown resistors are 45KOhm minimum, 65KOhm typical, 80KOhm maximum.

13.4 Thermal Characteristics

Symbol	Max.	Unit
Θjc	TBD	C/W
Θja	TBD, Estimated to be 5 C/W	C/W

14. AC TIMING - *TARGETS/SUBJECT TO CHANGE*

(TC= 0-70°C; VDD= +5V, +/- 5%)

Symbol	Signals	Description	Min	Max	Unit
t1	TClk	Pulse Width High	8		nS
t2	TClk	Pulse Width Low	8		nS
t3	TClk	Clock Period	20	30	nS
t4	TClk	Rise Time		3	n
t5	TClk	Fall Time		3	n
t6	Rst*	Active	10		TCl
t7	DMAReq[3]*, LEAdrE/ DMAReq[2]*, DMAReq[1]*/ParErr*, AD[31:0]	Setup	5		nS
t8	DMAReq[0]*/Ready*,	Setup	11		nS
t9	WrRdy*, ValidIn*, MasCmd[8:0], Interrupt*	Delay	2	12	nS
t10	DAdr[11:0]	Delay (Row Address)	3	18	nS
t11	DAdr[11:0]	Delay (Column Address)	3	11	nS
t12	BAdr[2:0], ADS*, MasAD[31:0]	Delay	2	13	nS
t13	EWr[3:0]*, OWr[3:0]*	Delay From TClk Falling Edge	2	13	nS
t14	DWr*, CSTiming*, ALE,	Delay	2	10	nS
t15	ECAS[3:0]*, OCAS[3:0]*, OEB, OEO*, OEE*, AD[31:0]	Delay	2	9	nS
t16	ALE, LEO, LEE	Delay from TClk Falling Edge	2	10	nS
t17	ValidOut*, Release*, DMAReq[3]*, LEAdrE/ DMAReq[2]*, DMAReq[1]*/ParErr*, MasAD[63:0], Mas-Cmd[8:0], AD[31:0], DMAReq[0]*/Ready*	Hold	1		nS
t18	ValidOut*, Release*,MasAD[31:0], MasCmd[8:0]	Setup	3		nS
t19	LEAdrE/DMAReq[2]*, LEAdrO	Delay	2	10	nS

t20	LEAdrE/DMAReq[2]*, LEAdrO	Delay From TClk Falling Edge	2	10	nS
t21	LEO, LEE	Delay	2	9	nS
t22	RAS[3:0]*	Delay	3	8	nS

Symbol	Signals	Description	Min	Max	Unit
	PClk, Frame*, IRdy*, TRdy*, DevSel*, Stop*, PErr*, Par, PAD[31:0], CBE[3:0]*, Gnt*, IdSel, Lock*, Req*, SErr*, Int*	See PCI Specification Rev. 2.1. Referred to PClk.			

Notes:

1. All Delays, Setup, and Hold times are referred to TClk rising edge, unless stated otherwise.

2. All outputs are specified for 50pF load except: WrRdy*, ValidIn*, ALE, LEAdrO, LEAdrE/DMAReq[2]* - 30pF, Interrupt* - 20 pF.

3. TClk frequency must be equal or greater than PClk frequency.

4. Maximum allowable TClk to PClk skew in sync. mode is 2nS.

14.1 AC Timing Waveforms

AC timing waveforms will be included in the next revision of this document. For now, please refer to the GT-64010A timing waveforms.

15. Functional Waveforms

A full set of functional waveforms can be found on our FTP site at ftp.galileot.com. Functional waveforms for the GT-64060 are identical to the GT-64011 (see below.) The next revision of this document will include a subset of these waveforms.

16. Applications

This will be in the next revision of this document.

17. Relationship to the GT-64011

The GT-64060 is identical to the GT-64011. It is marketed as a seperate device in order to appeal to a different segment of the market. The MIPs bus has historically instilled fear in the average designer. This is probably due to the difficulty of interfacing to the old R2000/3000 bus (multiple clocks, tight timings, etc.) The R4000 bus, for which the GT-64060 is designed, is much easier to interface to, but still intimidates some customers. By documenting the bus protocol, and giving some interfacing tips, we hope to open up the powerful GT-6401x family to a wider range of applications.

The GT-64060 and GT-64011 are priced identically and can be substituted for each other on a Bill of Materials.

18. Revision History

Rev Number	Comments
0.7	First non-NDA release. (3/97)