Galileo Technology™

System Controller with PCI Interface for R4XXX/ R5000 Family CPUs

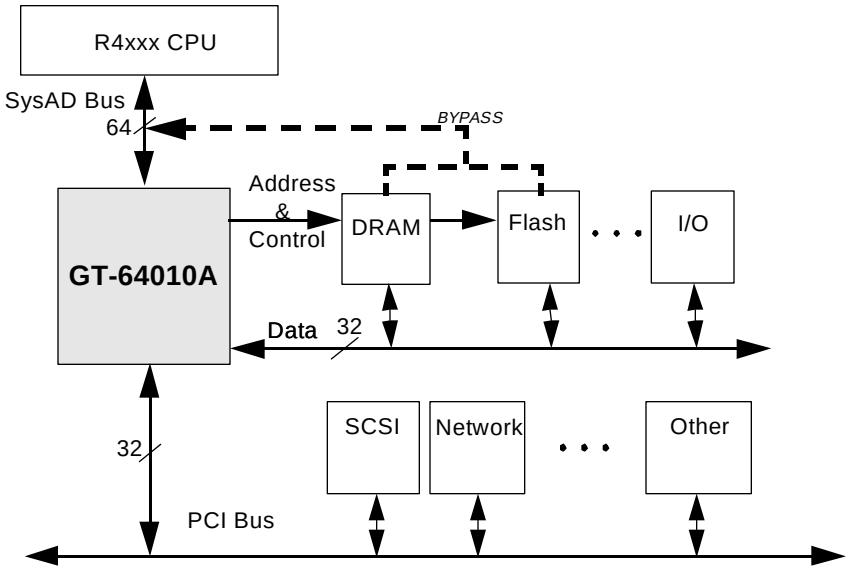
GT-64010A

Preliminary
Revision 1.1
December 1996

NOTE: Always contact Galileo Technology for possible updates before starting a design.

FEATURES

- Integrated system controller with PCI bus interface for high performance embedded controller applications
- Supports the R4600/4650/4700/R5000 CPUs
- Up to 50MHz CPU bus frequency
- 64 byte write buffer
  - 64-bit wide
  - 8 levels deep
- 256KB or 512KB zero-wait-state secondary cache support by using GT-64012 (R4600/R4700)
- DRAM controller
  - Page mode and EDO DRAMs
  - 512MB address space
  - 256KB-16MB device depth
  - 1- 4 banks supported directly
  - 32-bit or 64-bit data width
  - Different size for each bank
- Device controller
  - 5 chip selects
  - Programmable timing for each chip select
  - Supports several types of standard memories (ROM / Flash / SRAM) and I/O controllers
  - Up to 160MB address space
  - External wait support
  - 8-,16-,32- and 64-bit width device (and boot) support
- External parity support for user selected banks of DRAM and devices
- 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 2.1 Specification
  - High performance PCI interfaces via 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 (asynchronous from CPU bus)
  - Supports burst operations on PCI for efficient data transfer
  - Supports doorbells between Host and PCI
  - Supports flexible byte swapping - no need for CPU intervention.
  - Synchronization Barrier support from CPU to PCI and from PCI to CPU.
- Host to PCI bridge
  - Translates CPU cycles into PCI I/O or Memory cycles
  - Generates configuration Interrupt Acknowledge and Special cycles on PCI
- PCI to Main 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
- PCI configuration registers are accessed from both CPU and PCI side
- Three 24-bit wide and one 32-bit wide timers/counters
- 5V Operation (3.3V operation using inexpensive support components
- 256 PQFP or 272 Ball-BGA



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

1. OVERVIEW

The GT-64010A is a highly integrated system controller that supports middle to high performance embedded control applications with state-of-the-art 64-bit MIPS processors, while significantly reducing their cost, complexity, device count, and board space.

The architecture of the chip supports several optional system architectures 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-64010A) for differentiated system architectures that attain higher performance.

The GT-64010A has a three bus architecture:

a) 64-bit interface to the CPU,
b) 32-bit interface to the memory subsystem, with a bypassing option to create a 64-bit path to the CPU,
c) 32-bit interface to the PCI bus.

The three buses are de-coupled from each other in most accesses, enabling concurrent operation of the CPU, PCI devices, and accesses to memory. For example, the CPU 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 at the same time.

1.1 Processor Interface

The GT-64010A supports without glue logic the IDT/MIPS family of R4xxx and R5000 Orion processors. It supports bus frequencies of up to 50MHz, while the processor can operate internally at 80 to 200MHz. The GT-64010A has a deep write buffer with the ability to absorb several write transactions from the CPU. For systems that want to increase performance even more, the GT-64010A supports the Galileo GT-64012 secondary cache controller. Systems with CPUs that run internally at 3x or more of the external frequency can particularly benefit from this option.

1.2 DRAM and Device Interface

The GT-64010A has a flexible DRAM controller. It supports EDO (Hyperpage) as well as standard page mode DRAMs. With 60ns standard DRAMs, the GT-64010A can return data at 7-2-2-2 to the CPU (four wait states to the first access). The DRAM controller supports different depth devices in each bank for base configuration at manufacturing, and allowing for field upgrades by end users. It supports 32-bit wide DRAMs for high granularity in applications where small memory size is desirable, or 64-bit wide DRAM where higher memory performance is needed.

The GT-64010A 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-64010A 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-64010A incorporates 96-bytes of posted write and read prefetch buffers for efficient data transfer between the CPU/DMA to PCI and PCI to main memory.

The GT-64010A becomes a PCI bus master when the CPU or the internal DMA engine initiates a bus cycle to a PCI device. It translates the CPU cycle into the appropriate PCI bus cycle. These cycles can be either Memory, Interrupt Acknowledge, Special, I/O, or Configuration cycles.

The GT-64010A 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.

The GT-64010A contains the required PCI configuration registers. All the internal registers, including the PCI configu-

ration registers, can be accessed from both the CPU and the PCI.

The GT-64010A can also act as a PCI to Memory bridge, even without the presence of the CPU.

1.4 DMA Engines

The GT-64010A 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 32-/64-bit 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 CPU or by PCI masters, or without CPU 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.

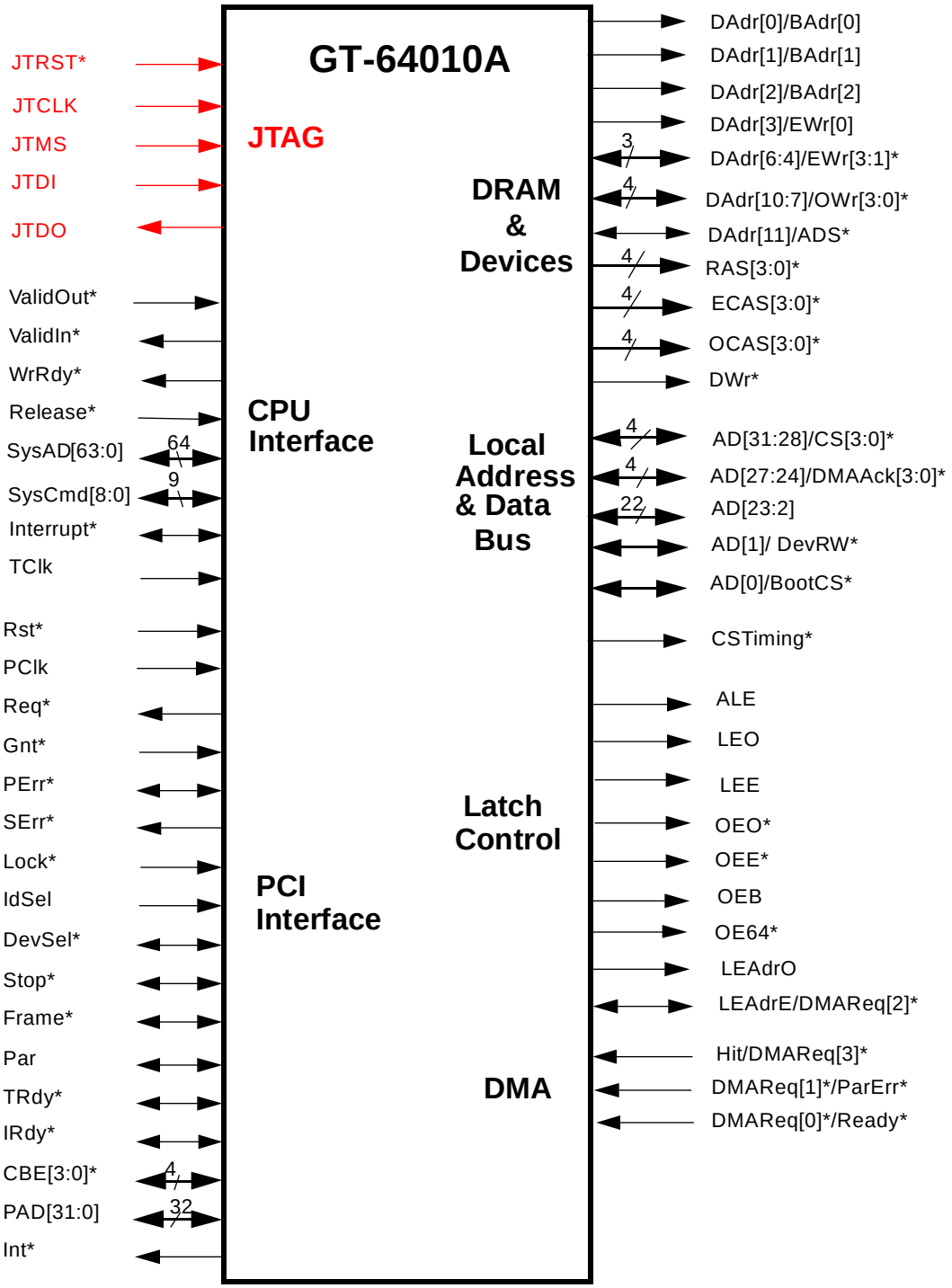
2. ORDERING INFORMATION

Please use the part numbers in the table below when placing orders for GT-64010 devices.

Package Type	Part Number
256 Pin PQFP	GT-64010A-P
272 Pin BGA	GT-64010A-B

3. PIN INFORMATION

3.1 Logic Symbol



3.2. Pin Assignment Table

Pin Name	Type	Description
CPU Interface		
Release*	I	**Release Interface:** Signals to the GT-64010A that the processor is releasing the system interface to slave state.
WrRdy*	O	**Write Ready:** The GT-64010A signals that it can accept a processor write request.
ValidIn*	O	**Valid Input:** The GT-64010A signals that it is driving valid data on the SysAD bus, and a valid data identifier on the SysCmd bus.
ValidOut*	I	**Valid Output:** Signals that the processor is driving valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
SysAD[63:0]	I/O	**System Address/Data Bus:** A 64-bit address and data bus for communication between the processor and GT-64010A.
SysCmd[8:0]	I/O	**System Command/Data Identifier Bus:** A 9-bit bus for command and data identifier transmission between the processor and GT-64010A.
Interrupt*	I/O	**Interrupt:** An “OR” of all the internal interrupt sources on the GT-64010A. This pin is also sampled as an input at reset for configuration purposes.
TClk	I	**Clock:** The input clock to the GT-64010A (up to 50MHz).
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-64010A 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.
Par	I/O	**Parity:** Calculated by the GT-64010A 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-64010A 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-64010A when it acts as a target.

Pin Name	Type	Description
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-64010A responds to the assertion of Stop* by disconnecting, retrying or aborting. As a target, the GT-64010A asserts Stop* to retry or disconnect.
Lock*	I	**Lock:** It indicates an atomic operation that may require multiple transactions to complete. When the GT-64010A is a PCI target, Lock* is sampled on the rising edge of the PCIk when Frame* is asserted. If Lock* is sampled asserted, the GT-64010A enters into a locked state and remains in this state until Lock* is sampled deasserted on the following rising edge of PCIk, 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-64010A 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-64010A aborts the cycle. As a target, when the GT-64010A 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-64010A to indicate to the bus arbiter that it desires use of the bus.
Gnt*	I	**Bus Grant:** It asserts to indicate to the GT-64010A 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-64010A asserts the SErr* two cycles after the failing address.
Int*	O	**Interrupt Request:** It is asserted by the GT-64010A when one of the unmasked internal interrupt sources is asserted.
DRAM & Devices		
DWr*	O	**DRAM Write:** It is LOW when the GT-64010A 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 (e.g. boot ROM), 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 SysAD[2]). Not used for 16/64 bit devices.

Pin Name	Type	Description
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 SysAD[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 SysAD[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 SysAD[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 SysAD[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.
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.
OCAS[3:0]*	O	**Odd Column Address Select:** Supports byte writes/reads to the odd bank of the DRAM.
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.

Pin Name	Type	Description
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.
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.
OE64*	O	**Output Enable 64-bit:** Output enable for the latch from memory to the SysAD bus.
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		
Hit/DMAReq[3]*	I	**Hit / DMA Request:** This pin can be configured as a hit indication from a secondary cache device or as a DMA request indication by an external device, depending on the setting of bit 9 of the CPU Interface Configuration register.
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.

Pin Name	Type	Description
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.
JTAG		
JTRST*	I	**JTAG Reset:** Asynchronous reset to test logic.
JTCLK	I	**JTAG Clock:** Clock for test logic. JTMS and JTDI are received on the rising edge, JTDO is driven from the falling edge. This signal determines the shifting rate.
JTMS	I	**JTAG Mode Select:** A broadcast signal which controls test logic operation.
JTDI	I	**JTAG Data In:** Serial data input.
JTDO	O	**JTAG Data Out:** Serial data output. Tri-state changes on negative change of JTCLK.

4. FUNCTIONAL DESCRIPTION

4.1 CPU Interface

The GT-64010A supports the R4XXX and R5000 family of 64-bit CPUs. It supports the IDT R4600, R4650, R4700, and R5000 at up to 50MHz bus frequency. The GT-64010A has a 64 byte, 8 level deep, write buffer that can absorb up to four CPU write transactions. Standard R4XXX write and the R4600 pipelined write modes are supported (the R4600 reissue option is not supported). CPU read requests are supported via two possible paths: reads from the PCI, and from 32-bit wide (or less) devices are performed through the GT-64010A; CPU reads from 64-bit devices bypass the GT-64010A through an external latch that is controlled by the GT-64010A. The interface supports the big and little endian options of the CPU.

4.2 Secondary Cache Support

The GT- 64010 supports the GT- 64012 secondary cache controller for the R4600/R4700 CPU's. The Hit input pin indicates to the GT-64010A if the current CPU read can be serviced from the cache memory or it needs to be serviced by the GT-64010A. Flush and Invalidate operations on the secondary cache are ignored by the GT-64010A.

4.3 Address Space Decode

The GT-64010A uses a distributed address decoding scheme. Each "master" unit (CPU or PCI) has a separate address decoding logic and registers. The DMA controller uses the address mapping of the CPU 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 seven groups: RAS[1:0], RAS[3:2], CS[2:0], CS[3] & BootCS, Internal, PCI I/O, 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. 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 CPU accesses and into 4 Gbyte address space for the DMA and PCI accesses.

When the CPU tries to access an address that is not supported, the GT-64010A will latch the address into the Bus Error registers, and will issue a bus error (over SysCmd[5]) if the access was a read access, and an interrupt if it was a read or write access.

4.4 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-64010A. The GT-64010A has a programmable parity integrity bit for each bank, which indicates if parity is supported.

In read accesses by the CPU from a 72-bit bank DRAM or device, the GT-64010A will assert SysCmd[4] to indicate to the CPU if it needs to check parity for the current access. In this case the GT-64010A relies on the CPU parity checking mechanism. In CPU read accesses from a 32-bit device or memory, the GT-64010A will not assert SysCmd[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-64010A will return the data with SysCmd[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-64010A will assert SErr* if the bank that data was read from has the parity integrity bit set, and will assert a parity error interrupt. The GT-64010A 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*. For the connection of memory buses, refer to section 8.

4.5 Memory Control

The GT64010A system can be assembled in different configurations which are up to the user.

There are two main configurations: without data latches on the AD bus, meaning that the system is implemented with a 32-bit or narrower bus, and with latches on the AD bus, which allows for 64-bit bus accesses.

For details regarding the connection of memory, see section 8.

4.6 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 can be 32- or 64-bits. 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 to two or three cycles, and CAS can be LOW for one or two cycles.

DRAM performance in CPU read accesses is 7-2-2-2, which means 4 wait states to first data and one wait state for each additional double 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.

4.7 Device Controller

This 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, four for the even bank (EWr[3:0]*) and four for the odd bank (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 CPU 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.

4.7.1. 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-64010A during both AccToFirst and AccToNext phases. During all other phases Ready* is not sampled by the GT64010A.

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*

4.8 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 CPU 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.

4.9 PCI Bus

The GT-64010A interfaces directly with the PCI bus. As a PCI device, the GT-64010A can be either a master initiating a PCI bus operation or a target responding to a PCI bus operation. When the CPU or the internal DMA machine initiates a bus cycle to a PCI device, the GT-64010A becomes a PCI bus master and translates the CPU cycle into the appropriate PCI bus cycle. The 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

As a master, the GT-64010A does not issue Dual Address cycles or Lock cycles on the PCI.

Memory Write & Invalidate and Memory Read Line cycles are carried out when the transaction addresses the PCI memory space requiring a data transfer equal to the cache line size. For details on Configuration, Interrupt Acknowledge and Special cycles, refer to section 3.14 (PCI Configuration registers).

The PCI posted write buffer in the GT-64010A permits the CPU to complete CPU-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.

When a PCI bus master initiates an access, the GT-64010A becomes the target of the PCI bus cycle and responds to the read/write access. The GT-64010A responses to the following PCI cycles:

- 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-64010A will lock a cache line (32-bytes) in the memory address space when responding to Lock sequences on the PCI bus.

The GT-64010A will not act as a target (slave) for Interrupt Acknowledge, Special, and Dual Address cycles.

The GT-64010A incorporates dual 32-bytes posted write/read prefetch buffers to allow full memory and PCI bus concurrence. In every posted write cycle, the data is first written to the buffers. When a buffer fills up (32 bytes), the data is written to the destination while the second buffer is being filled up. I/O and Configuration cycles are non-postable.

The Memory Read Multiple cycle is the only prefetchable read cycle. Every time there is at least two words within the first buffer, data is being prefetched into the second buffer. In a non-prefetchable read cycle, the data is written into the second buffer after the first one gets full.

The master's FIFO includes 8 entries each 32 bit.

During writes, it receives write data from the CPU interface or the DMA unit. When the PCI bus is granted, it delivers the write data to the target on the PCI bus. Upon receiving the first doubleword from the CPU interface or DMA unit, the machine which handles the PCI bus protocol is triggered to request the bus (if the GT-64010A is not already parked). Once granted, the appropriate write cycle is started on the PCI bus.

During reads, the master's FIFO receives read data from the PCI bus and delivers it to the CPU interface or DMA unit. Upon receiving the first doubleword from the PCI target, the data is forwarded to the requesting unit (CPU interface or DMA unit). The GT-64010A supports sub-block ordering during CPU reads, therefore if the original read request address is not aligned to a cache line boundary, the first doubleword 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-64010A internal architecture allows zero wait state data transfer over the PCI bus (Irdy* continuously asserted) during both reads and writes.

As a PCI device, the GT-64010A contains all of the required PCI configuration registers. These registers, as well as the GT-64010A's internal registers are accessed from both the CPU and the PCI bus.

The GT-64010A considers some cycles as being “synchronization barrier” cycles. In such cycles, the GT-64010A makes sure that at the end of the cycle there remains no posted data within the chip. The slave “synchronization barrier” cycles are Lock Read and Configuration Read. If there is no posted data within the GT-64010A, 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 CPU 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-64010A.

As a target, the GT-64010A can be programmed not to respond to specific address spaces. Also it can be programmed to swap or not swap bytes, dependent on transaction nature (write/read) and address space. For example, the GT-64010A can transfer written words directly to a certain DRAM bank and read these words byte-swapped, independent of other banks. This allows PCI to PCI byte swapping without CPU intervention.

4.10 Interrupt Controller

The interrupt controller groups all the internal interrupt sources and asserts an interrupt to the CPU 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 CPU or by the PCI. The interrupt will be acknowledged by the CPU 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 CPU Mask register and one bit in the PCI Mask register. A zero in the CPU Mask register bit will mask the interrupt from asserting an interrupt to the CPU. A zero in the PCI Mask register bit will mask the interrupt from asserting an interrupt to the PCI.

4.11 Timer/Counters

The GT-64010A 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.

4.12 Reset Configuration

The GT-64010A 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-64010A 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.

Pin	Configuration Function
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]:	Reserved
0-	Must be sampled LOW
Dadr[5]:	DMAReq[1]*/ParErr* Selection
0-	DMAReq[1]* (no parity)
1-	ParErr*
Dadr[4]:	DMAReq[0]*/Ready* Selection
0-	Ready*
1-	DMAReq[0]*

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

4.13 JTAG

The GT-64010A incorporates JTAG as per IEEE-1149.1. The following OpCodes are supported:

- BYPASS (111)
- EXTEST (000)
- SAMPLE (010)
- IDCODE (001)

The IDCODE Register fields’ values in the GT-64010A are:

Manufacturer (11 bits) - 0x08a

Part Number (16 bits) - 0x0146

Version (4 bits) - 0x2

For the GT-64010A signal ordering of boundary chain please refer to JTAG Appendix, section 11.3.

5. REGISTER TABLES

The GT-64010A's internal registers can be accessed by the CPU or by the PCI. They are memory-mapped for the CPU 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 [14:0] is matched against bits [35:21] of the actual address; therefore, this value should be the actual address bits [35:21] 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 36-bit address of 0x014000800. 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 0x014000068 a value of "0bd", an access to the "Channel 0 DMA Byte Count" register will be with 0x017a00800.

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 0x0bd000cf8). Then, reading from the Configuration Data register (offset 0xcfc), will return the data of the Status and Command register.

5.1 Register Map

Description	Offset
CPU Interface	
CPU 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 Memory Low Decode Address	0x058
PCI Memory High Decode Address	0x060
Internal Space Decode	0x068
Bus Error Address Low Processor	0x070
Bus Error Address High Processor	0x078
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
Configuration Address	0xcf8
Configuration Data	0xcfc
Interrupts	
Interrupt Cause	0xc18
CPU Mask	0xc1c
PCI Mask	0xc24
PCI Configuration	
Device and Vendor ID	0x000
Status and Command	0x004
Class Code and Revision ID	0x008
Header Type, Latency Timer, Cache Line	0x00c
RAS[1:0] Base Address	0x010
RAS[1:0] Swapped Base Address	0x028
RAS[3:2] Base Address	0x014
RAS[3:2] Swapped Base Address	0x02c
CS[2:0] Base Address	0x018
CS[3] & Boot CS Base Address	0x01c
CS[3] & Boot CS Swapped Base Address	0x030
Internal Registers Memory Mapped Base Address	0x020
Internal Registers I/O Mapped Base Address	0x024
Base Address Registers' Enable	0x034
Interrupt Pin and Line	0x03c

5.2 CPU Interface

The CPU Interface Configuration register determines which of the different MIPS write protocols is supported, as well as secondary cache issues and CPU endianess. The differences in protocol are minimal and they include support for pipelined write, etc. The GT-64010A supports the Galileo GT-64012 secondary cache controller, and thus when the CachePres bit is set to 1, the GT-64010A will not service cacheable read requests if there is a Hit indication from the secondary cache. The CacheOpMap bits indicate which address bits will be used for cache flush and cache invalidate operation by the GT-64012. When the GT-64010A finds a match between a '1' in the read address and the CacheOp-Map bits that are set to '1', it will ignore the access, and the GT-64012 will flush or invalidate the secondary cache. For example, setting bit 8 to '1' indicates that SysAD[35] is connected to one of the two GT-64012's TagOp inputs, setting bit 7 to '1' indicates that SysAD[34] is connected to one of them, etc.

CPU Interface Configuration, Offset: 0x000

Bits	Field name	Function	Initial Value
8:0	CacheOpMap	Cache operation mapping. Indicates which address bits will be used for cache flush and cache invalidate operation by the GT-64012. Bits 8:0 correspond to SysAd[35:27].	0x0
9	CachePres	Secondary cache support. 0 - GT-64012 not present 1 - GT-64012 present	0x0
10	Reserved	Must be '0'.	0x0
11	WriteMod	Write mode. 0 - Pipelined writes mode 1 - R4XXX 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

5.3 Processor Address Space

The Decode Address registers determine which physical device group will be accessed when the CPU 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 35:28 of the address to bits 14:7 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 CPU will be interrupted during a write 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, with the exception of bits 35:32.

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

Bits	Field Name	Function	Initial Value
14:0	Low	DRAM banks 1 and 0 will be accessed when the decoded addresses are between Low and High.	0x0000

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
14:0	Low	DRAM banks 3 and 2 will be accessed when the decoded addresses are between Low and High.	0x0008

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
14:0	Low	Device banks 2, 1 and 0 will be accessed when the decoded addresses are between Low and High.	0x00e0

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
14:0	Low	Device bank 3 and the boot bank will be accessed when the decoded addresses are between Low and High.	0x00f8

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
14:0	Low	The PCI I/O address space will be accessed when the decoded addresses are between Low and High.	0x0080

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
14:0	Low	The PCI memory address space will be accessed when the decoded addresses are between Low and High.	0x0090

PCI Memory High Decode Address, Offset: 0x060

Bits	Field Name	Function	Initial Value
7:0	High	The PCI memory 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
14:0	IntDecode	Registers inside the GT-64010A will be accessed when SysAD bits 35:21 match the value programmed in bits 14:0.	0x00a0

Bus Error Address Low Processor, Offset: 0x070

Bits	Field Name	Function	Initial Value
31:0	IlegLoAdd	This register captures bits 31:0 of an illegal 64-bit address.	0x00000000

Bus Error Address High Processor, Offset: 0x078

Bits	Field Name	Function	Initial Value
3:0	IlegHiAdd	This register captures bits 35:32 of an illegal 64-bit address.	0x0

5.4 DRAM and Device Address Space

The Decode Address registers determine which physical device will be accessed when the CPU, 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 CPU 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.

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

5.5 DRAM Configuration

The DRAM Configuration register specifies refresh parameters and optional usage of two of the GT-64010A 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

5.6 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
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- 64 (72) bit wide DRAM	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

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

5.7 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-, 32-, or 64-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 64-bits wide. For devices that are less then 64-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-64010A 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	Read only.	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

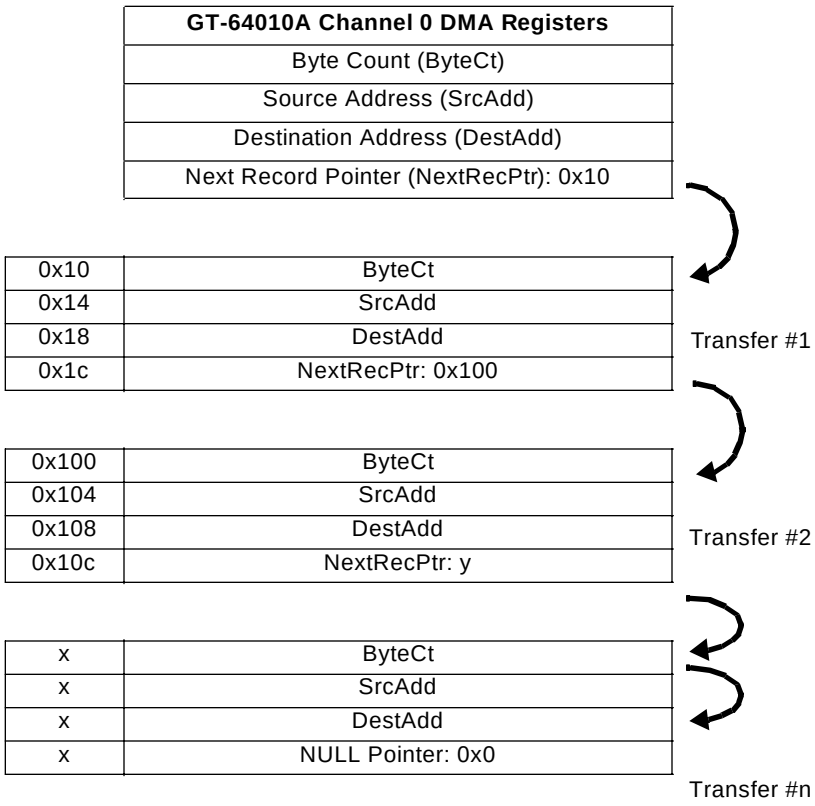
Bits	Field Name	Function	Initial Value
30:0	Various	Fields function as in Device Bank0.	0x14?fffff

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.

5.8 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 CPU, PCI, or DMA controller in the process of fetching a new record from

memory. The structure of the record is illustrated in the following example:



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

5.9 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-64010A. 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) with-

out involving the CPU. 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 FetNexRec 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 CPU (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
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

5.10 Additional Notes On DMA Programming

5.10.1. 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’.

5.10.2. 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’.

5.10.3. 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’.

5.10.4. 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.

5.11 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,.....

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

5.12 Timer / Counter

There are three 24-bit wide and one 32-bit wide timers/counters on the GT-64010A. 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

5.13 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-64010A swaps the incoming and outgoing PCI data.	Set to the same value sampled at reset into bit[12] of the CPU 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 (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-64010A, 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-64010A, 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-64010A Master. The GT-64010A 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-64010A detects a parity error on the address lines.	0x0
1	MasWrErr	Mask bit. When set, SErr* is asserted when the GT-64010A detects a parity error during a master write operation.	0x0
2	MasRdErr	Mask bit. When set, SErr* is asserted when the GT-64010A 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-64010A performs master abort.	0x0
5	TarAbort	Mask bit. When set, SErr* is asserted when the GT-64010A detects a target abort.	0x0

Interrupt Acknowledge, Offset: 0xc34

Bits	Field Name	Function	Initial Value
31:0	IntAck	The data is meaningless. A CPU read operation to this register causes the GT-64010A to perform an Interrupt Acknowledge cycle on the PCI bus.	0x00000000

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
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 CPU accesses this register and the ConfigEn bit in the Configuration Address register is set. A CPU access to this register causes the GT-64010A to perform a Configuration or Special cycle on the PCI bus.	0x000

5.14 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-64010A. Therefore, bit[0] of both the CPU Mask and PCI Mask registers is read-only '0'.

CPUIntSum in the Interrupt Cause register is the logical OR of bits[29:26,20:1], masked by bits[29:26,20:1] of the CPU Mask register. Therefore, bits[25:21] of the CPU Mask register, being non-relevant to interrupts directed to the CPU, 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

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 CPU 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	CPUOut	Asserts when the CPU 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
11	T3Exp	Asserts when Timer 3 expires.	0x0
12	MasRdErr	Asserts when the GT-64010A detects a parity error during a master read operation.	0x0
13	SlvWrErr	Asserts when the GT-64010A detects a parity error during a slave write operation.	0x0
14	MasWrErr	Asserts when the GT-64010A detects a parity error during a master write operation.	0x0
15	SlvRdErr	Asserts when the GT-64010A detects a parity error during a slave read operation.	0x0
16	AddrErr	Asserts when the GT-64010A 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	CPUInt	These bits are set by the CPU 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 CPU. They are cleared when the CPU writes '0'.	0x0
30	CPUIntSum	Interrupt summary. Logical OR of bits[29:26,20:1], masked by bits[29:26,20:1] of the CPU Mask register.	0x0
31	PCIIntSum	Interrupt summary. Logical OR of bits[25:1], masked by bits[25:1] of the PCI Mask register.	0x0
	All bits are cleared by writing a value of '0' by the CPU or PCI, unless stated otherwise.		

CPU Mask, Offset: 0xc1c

Bits	Field Name	Function	Initial Value
31:0	CPUMask	Mask to the CPU 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

5.15 PCI Configuration

The GT-64010A contains the required PCI configuration registers. These registers are accessed from both the CPU and the PCI. The GT-64010A translates CPU read and write cycles into configuration cycles using the PCI configuration mechanism #1. Mechanism #1 defines a way to translate the CPU cycles into both, PCI configuration cycles on the PCI and accesses to the GT-64010A’s internal configuration registers. The GT-64010A includes two registers: Configuration Address (in offset 0xcf8) and Configuration Data (in offset 0xcfc). The general mechanism for accessing the configuration registers is to write a value into Configuration Address that specifies the PCI bus, the device on that bus and the configuration register in that device being accessed. A read or write to Configuration Data will then causes the GT-64010A 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 done which addresses a device attached to the local PCI bus (for DevNum to IdSel mapping, refer to the table following this paragraph). 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 CPU accesses the GT-64010A’s internal configuration registers when the fields DevNum and BusNum in the Configuration Address register are equal to ‘0’. The GT-64010A configuration registers are also accessed from the PCI using the normal PCI read and write configuration cycles.

Note: The CPU Interface unit cannot distinguish between an access to the GT-64010A PCI configuration space and an access to some other PCI device configuration space. This is because both are accessed using an access to the GT-64010A internal space (i.e. Configuration Data register). When the CPU is operating in big endian mode, any access to the GT-64010A internal space undergoes byte swapping as all internal registers are little endian. With the CPU 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-64010A configuration space but will be swapped twice for PCI configuration accesses intended for devices external to the GT-64010A. This requires the software to format write data and interpret read data differently for PCI configuration accesses to the GT-64010A and configuration accesses through the GT-64010A. See also appendix 11.1.

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

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.

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

Device and Vendor ID, Offset: 0x000

Bits	Field name	Function	Initial Value
31:16	DevID	Provides the unique GT-64010A ID number (0x146).	0x0146
15:0	VenID	Provides the manufacturer of the GT-64010A (0x11ab).	0x11ab

Status and Command, Offset: 0x004

Bits	Field name	Function	Initial Value
0	IOEn	Controls the GT-64010A's response to I/O accesses. 0 - Disable 1 - Enable	0x0
1	MEMEn	Controls the GT-64010A's response to Memory accesses. 0 - Disable 1 - Enable	0x0
2	MasEn	Controls the GT-64010A's ability to act as a master on the PCI bus. 0 - Disable 1 - Enable	0x0
4	MemWrInv	Controls the GT-64010A's ability to generate Memory Write & Invalidate command on the PCI bus. 0 - Disable 1 - Enable	0x0
6	PErrEn	Controls the GT-64010A'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-64010A's ability to assert the SErr* pin. 0 - Disable 1 - Enable	0x0
23	TarFastBB	Read only bit. Indicates that the GT-64010A is capable of accepting fast back-to-back transactions on the PCI bus.	0x1
24	DataParDet	This bit is set by the GT-64010A when it detects a data parity error during master operation.	0x0
27:25	DevSelTim	These pins indicate the GT-64010A '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-64010A Revision number.	0x02
31:24	BaseClass	Indicates the GT-64010A Base Class (0x6 - bridge device).	0x06
23:16	SubClass	Indicates the GT-64010A Subclass (0x0 - host bridge).	0x00

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

Bits	Field name	Function	Initial Value
7:0	CacheLine	Specifies the GT-64010A's cache line size (size=8).	0x07
15:8	LatTimer	Specifies in units of PCI bus clocks the value of the latency timer of the GT-64010A.	0x00
23:16	HeadType	Specifies the layout of bytes 10h through 3fh.	0x00

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-64010A 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 CPU 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-64010A 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[1:0] Swapped Base Address, Offset: 0x028

Bits	Field Name	Function	Initial Value
31:12	Base	Defines the address assignment of Swapped 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

RAS[3:2] Swapped Base Address, Offset: 0x02c

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

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

CS[3] and Boot CS Swapped Base Address, Offset: 0x030

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

Internal Registers Memory Mapped Base Address, Offset: 0x020

Bits	Field Name	Function	Initial Value
31:12	MemMapBase	Defines the address assignment of the GT-64010A's internal registers.	0x14000

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-64010A's internal registers.	0x14000

Base Address Registers' Enable, Offset: 0x034

Bits	Field name	Function	Initial Value
8	RAS[1:0]En	Controls address matching with RAS[1:0] base/size. 1 - Disable 0 - Enable	0x0
7	RAS[3:2]En	Controls address matching with RAS[3:2] base/size. 1 - Disable 0 - Enable	0x0
6	CS[2:0]En	Controls address matching with CS[2:0] base/size. 1 - Disable 0 - Enable	0x0
5	CS[3] & Boot CSEn	Controls address matching with CS[3] & Boot CS base/size. 1 - Disable 0 - Enable	0x0
4	IntMeMEn	Controls address matching with internal registers-Memory mapped base/size. 1 - Disable 0 - Enable	0x0
3	IntIOEn	Controls address matching with internal registers I/O mapped base/size. 1 - Disable 0 - Enable	0x0
2	SwRAS[1:0]En	Controls address matching with Swapped RAS[1:0] base/size. 1 - Disable 0 - Enable	0x1
1	SwRAS[3:2]En	Controls address matching with Swapped RAS[3:2] base/size. 1 - Disable 0 - Enable	0x1
0	SwCS[3] & Boot CSEn	Controls address matching with Swapped CS[3] & Boot CS base/size. 1 - Disable 0 - Enable	0x1

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-64010A. The GT-64010A uses INTA.	0x01

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

6. RESTRICTIONS

6.1 CPU Interface

a) The CPU should not attempt an access before 10 TClk cycles following deassertion of Rst* have expired.
b) CacheOpMap should not be written to a value other than 0 unless CachePres bit is set (see section 3.2).
c) The CPU Interface supports only DDDD and DXDXDXDX write patterns.
d) 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.
e) 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 (SysCmd[5] equal '1').

6.2 Memory Interface

a) If latches are not present, all banks must be programmed to be on the even bus. Programming the registers to 64-bit mode or to dynamically controlled latches will result in an error.
b) Unless the boot device is 64-bits wide, the boot will be on the even bank.
c) All Device Parameters (section 3.7) must be greater or equal to 3. i.e., AccToFirst, AccToNext, ADSToWr, WrActive and WrHigh.
d) When working with an 8- or 16-bit bus from CPU, a read/write operation can not exceed 64-bits (8 bytes).
e) When working with an 8- or 16-bit bus from DMA/PCI, a read/write operation can't exceed 32-bits (4 bytes).
f) When an erroneous address is issued or a burst operation is performed to an 8- or 16-bit device, the GT-64010A 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.
g) When the CPU reads from an address which is decoded in the CPU 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]*.

6.3 PCI Interface

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

6.3.1. Master

a) Latency count, as specified in LatTimer (section 3.14), should not be programmed to less than 6.

6.3.2. 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).

6.4 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.
   - Fly-by is not supported.

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

7. PINOUT TABLE

7.1 256 pin PQFP (sorted by number)

Pin #	Signal Name	Pin #	Signal Name	Pin #	Signal Name
1	VDD	36	PAD[15]	71	SysCmd[2]
2	VSS	37	VSS	72	SysCmd[1]
3	PAD[27]	38	PAD[14]	73	SysCmd[0]
4	PAD[26]	39	VSS	74	SysAD[0]
5	PAD[25]	40	PAD[13]	75	SysAD[1]
6	PAD[24]	41	PAD[12]	76	SysAD[2]
7	CBE[3]*	42	PAD[11]	77	SysAD[3]
8	IdSel	43	PAD[10]	78	SysAD[4]
9	VDD	44	PAD[9]	79	SysAD[5]
10	VSS	45	VDD	80	VSS
11	PAD[23]	46	VSS	81	VDD
12	PAD[22]	47	PAD[8]	82	SysAD[6]
13	PAD[21]	48	CBE[0]*	83	SysAD[7]
14	PAD[20]	49	PAD[7]	84	SysAD[8]
15	PAD[19]	50	PAD[6]	85	SysAD[9]
16	VSS	51	PAD[5]	86	SysAD[10]
17	PAD[18]	52	VDD	87	SysAD[11]
18	PAD[17]	53	VSS	88	SysAD[12]
19	PAD[16]	54	PAD[4]	89	SysAD[13]
20	CBE[2]*	55	PAD[3]	90	VSS
21	Frame*	56	PAD[2]	91	TClk
22	VDD	57	PAD[1]	92	VDD
23	VSS	58	PAD[0]	93	SysAD[14]
24	IRdy*	59	OE64*	94	SysAD[15]
25	TRdy*	60	Hit/DMAReq[3]*	95	SysAD[16]
26	DevSel*	61	Interrupt*	96	SysAD[17]
27	Stop*	62	VSS	97	SysAD[18]
28	VDD	63	SysCmd[8]	98	VSS
29	Lock*	64	SysCmd[7]	99	VDD
30	PErr*	65	SysCmd[6]	100	SysAD[19]
31	VDD	66	SysCmd[5]	101	VSS
32	VSS	67	SysCmd[4]	102	SysAD[20]
33	SErr*	68	VSS	103	SysAD[21]
34	Par	69	VDD	104	SysAD[22]

Pin #	Signal Name	Pin #	Signal Name	Pin #	Signal Name
35	CBE[1]*	70	SysCmd[3]	105	SysAD[23]
106	SysAD[24]	144	SysAD[53]	182	AD[21]
107	SysAD[25]	145	SysAD[54]	183	AD[20]
108	SysAD[26]	146	SysAD[55]	184	AD[19]
109	SysAD[27]	147	SysAD[56]	185	AD[18]
110	SysAD[28]	148	VSS	186	AD[17]
111	VSS	149	SysAD[57]	187	AD[16]
112	VDD	150	SysAD[58]	188	AD[15]
113	SysAD[29]	151	SysAD[59]	189	AD[14]
114	SysAD[30]	152	SysAD[60]	190	AD[13]
115	SysAD[31]	153	SysAD[61]	191	VSS
116	SysAD[32]	154	SysAD[62]	192	AD[12]
117	ValidOut*	155	SysAD[63]	193	AD[11]
118	ValidIn*	156	DMAReq[0]*/Ready*	194	AD[10]
119	WrRdy*	157	DMAReq[1]*/ParErr*	195	AD[9]
120	Release*	158	LEAdrE/DMAReq[2]*	196	AD[8]
121	SysAD[33]	159	LEAdrO	197	AD[7]
122	SysAD[34]	160	VDD	198	AD[6]
123	SysAD[35]	161	VSS	199	AD[5]
124	SysAD[36]	162	OEB	200	AD[4]
125	VSS	163	OEE*	201	AD[3]
126	SysAD[37]	164	OEO*	202	VSS
127	SysAD[38]	165	LEE	203	AD[2]
128	SysAD[39]	166	LEO	204	AD[1]/DevRW*
129	SysAD[40]	167	ALE	205	AD[0]/BootCS*
130	SysAD[41]	168	VSS	206	DWr*
131	SysAD[42]	169	CSTiming*	207	ECAS[3]*
132	SysAD[43]	170	AD[31]/CS[3]*	208	ECAS[2]*
133	SysAD[44]	171	AD[30]/CS[2]*	209	ECAS[1]*
134	SysAD[45]	172	AD[29]/CS[1]*	210	ECAS[0]*
135	SysAD[46]	173	AD[28]/CS[0]*	211	VDD
136	VSS	174	AD[27]/DMAAck[3]*	212	VSS

Pin #	Signal Name	Pin #	Signal Name	Pin #	Signal Name
137	VDD	175	AD[26]/DMAAck[2]*	213	OCAS[3]*
138	SysAD[47]	176	AD[25]/DMAAck[1]*	214	OCAS[2]*
139	SysAD[48]	177	AD[24]/DMAAck[0]*	215	OCAS[1]*
140	SysAD[49]	178	AD[23]	216	OCAS[0]*
141	SysAD[50]	179	AD[22]	217	RAS[3]*
142	SysAD[51]	180	VSS	218	RAS[2]*
143	SysAD[52]	181	VDD	219	RAS[1]*
220	RAS[0]*	233	DAdr[5]/EWr[2]*	246	VDD
221	VSS	234	DAdr[4]/EWr[1]*	247	PClk
222	VDD	235	DAdr[3]/EWr[0]*	248	VSS
223	DAdr[11]/ADS*	236	DAdr[2]/BAdr[2]	249	Gnt*
224	VDD	237	DAdr[1]/BAdr[1]	250	Req*
225	VSS	238	DAdr[0]/BAdr[0]	251	VSS
226	DAdr[10]/OWr[3]*	239	JTRST*	252	VDD
227	DAdr[9]/OWr[2]*	240	JTDI	253	PAD[31]
228	DAdr[8]/OWr[1]*	241	JTMS	254	PAD[30]
229	DAdr[7]/OWr[0]*	242	JTDO	255	PAD[29]
230	DAdr[6]/EWr[3]*	243	JTCLK	256	PAD[28]
231	VSS	244	Int*		
232	VDD	245	Rst*		

7.2 5.2 256-Ball BGA + 16-Ball Ground Matrix Pinout (sorted by numbers)

NOTE: 272 Balls altogether, 256-ball periphery balls in a 4-row array, plus 16-ball ground matrix in the center.

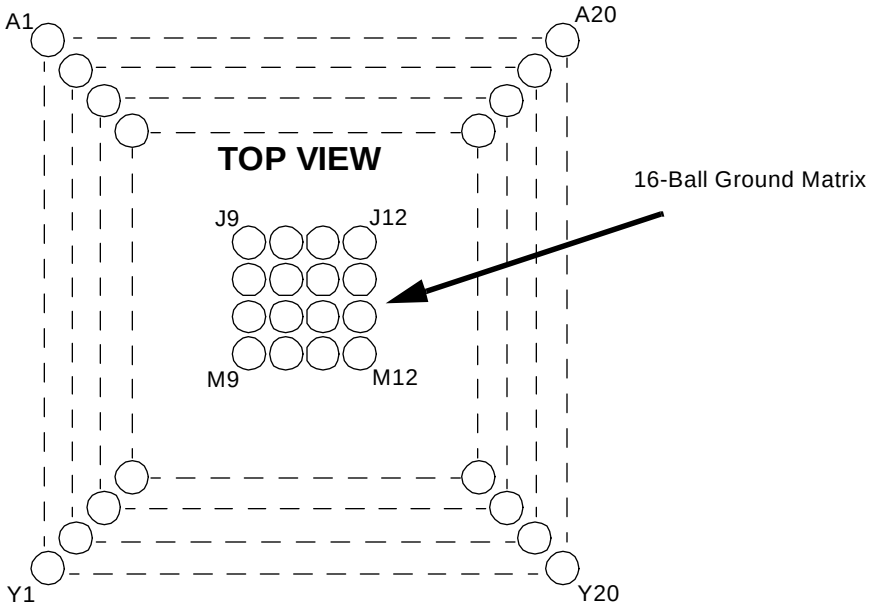
Ball #	Signal Name	Ball #	Signal Name	Ball #	Signal Name
A1	AD[13]	B16	SysAD[52]	D11	VDD
A2	AD[14]	B17	SysAD[49]	D12	VSS
A3	AD[16]	B18	SysAD[46]	D13	VSS
A4	AD[19]	B19	SysAD[43]	D14	VDD
A5	AD[22]	B20	SysAD[41]	D15	VDD
A6	AD[25]/DMAAck[1]*	C1	AD[10]	D16	VSS
A7	AD[28]/CS[0]*	C2	AD[9]	D17	VSS
A8	AD[31]/CS[3]*	C3	AD[17]	D18	SysAD[36]
A9	LEO	C4	AD[20]	D19	SysAD[37]
A10	OEE*	C5	AD[23]	D20	SysAD[35]
A11	LEAdrE/DMAReq[2]*	C6	AD[26]/DMAAck[2]*	E1	AD[4]

Ball #	Signal Name	Ball #	Signal Name	Ball #	Signal Name
A12	SysAD[63]	C7	AD[29]/CS[1]*	E2	AD[3]
A13	SysAD[60]	C8	CSTiming*	E3	AD[5]
A14	SysAD[57]	C9	LEE	E4	VSS
A15	SysAD[54]	C10	OEB	E17	VSS
A16	SysAD[51]	C11	DMAReq[1]*/ParErr*	E18	WrRdy*
A17	SysAD[48]	C12	SysAD[62]	E19	SysAD[34]
A18	SysAD[45]	C13	SysAD[59]	E20	Release*
A19	SysAD[42]	C14	SysAD[56]	F1	AD[1]/DevRW*
A20	SysAD[40]	C15	SysAD[53]	F2	AD[0]/BootCS*
B1	AD[12]	C16	SysAD[50]	F3	AD[2]
B2	AD[11]	C17	SysAD[47]	F4	VDD
B3	AD[15]	C18	SysAD[44]	F17	SysAD[33]
B4	AD[18]	C19	SysAD[39]	F18	SysAD[32]
B5	AD[21]	C20	SysAD[38]	F19	Validin*
B6	AD[24]/DMAAck[0]*	D1	AD[7]	F20	ValidOut*
B7	AD[27]/DMAAck[3]*	D2	AD[6]	G1	ECAS[3]*
B8	AD[30]/CS[2]*	D3	AD[8]	G2	ECAS[2]*
B9	ALE	D4	VSS	G3	DWr*
B10	OEO*	D5	VSS	G4	VDD
B11	LEAdrO	D6	VDD	G17	VDD
B12	DMAReq[0]*/Ready*	D7	VDD	G18	SysAD[28]
B13	SysAD[61]	D8	VSS	G19	SysAD[31]
B14	SysAD[58]	D9	VSS	G20	SysAD[30]
B15	SysAD[55]	D10	VDD	H1	ECAS[0]*
H2	OCAS[3]*	L12	VSS	R18	SysAD[2]
H3	ECAS[1]*	L17	SysAD[14]	R19	SysAD[4]
H4	VSS	L18	SysAD[15]	R20	SysAD[3]
H17	SysAD[29]	L19	SysAD[17]	T1	JTRST*
H18	SysAD[25]	L20	SysAD[16]	T2	JTMS
H19	SysAD[27]	M1	DAdr[9]/OWr[2]*	T3	JTDI
H20	SysAD[26]	M2	DAdr[11]/ADS*	T4	VSS
J1	OCAS[1]*	M3	DAdr[10]/OWr[3]*	T17	VSS
J2	OCAS[0]*	M4	VSS	T18	SysCmd[0]
J3	OCAS[2]*	M9	VSS	T19	SysAD[1]
J4	VSS	M10	VSS	T20	SysAD[0]
J9	VSS	M11	VSS	U1	JTDO
J10	VSS	M12	VSS	U2	JTCLK
J11	VSS	M17	VSS	U3	VSS

Ball #	Signal Name	Ball #	Signal Name	Ball #	Signal Name
J12	VSS	M18	SysAD[12]	U4	VSS
J17	VSS	M19	SysAD[13]	U5	VSS
J18	SysAD[22]	M20	TClk	U6	VDD
J19	SysAD[24]	N1	DAdr[7]/OWr[0]*	U7	VDD
J20	SysAD[23]	N2	DAdr[8]/OWr[1]*	U8	VSS
K1	RAS[3]*	N3	DAdr[6]/EWr[3]*	U9	VSS
K2	RAS[2]*	N4	VSS	U10	VDD
K3	VDD	N17	SysAD[6]	U11	VDD
K4	VDD	N18	SysAD[9]	U12	VSS
K9	VSS	N19	SysAD[11]	U13	VSS
K10	VSS	N20	SysAD[10]	U14	VDD
K11	VSS	P1	DAdr[3]/EWr[0]*	U15	VDD
K12	VSS	P2	DAdr[5]/EWr[2]*	U16	VSS
K17	SysAD[18]	P3	DAdr[4]/EWr[1]*	U17	VSS
K18	SysAD[19]	P4	VDD	U18	SysCmd[3]
K19	SysAD[21]	P17	VDD	U19	SysCmd[1]
K20	SysAD[20]	P18	SysAD[5]	U20	SysCmd[2]
L1	RAS[1]*	P19	SysAD[8]	V1	Int*
L2	RAS[0]*	P20	SysAD[7]	V2	PClk
L3	VDD	R1	DAdr[1]/BAdr[1]	V3	PAD[31]
L4	VDD	R2	DAdr[2]/BAdr[2]	V4	PAD[28]
L9	VSS	R3	DAdr[0]/BAdr[0]	V5	PAD[25]
L10	VSS	R4	VDD	V6	IdSel
L11	VSS	R17	VDD	V7	PAD[21]
V8	PAD[18]	W6	PAD[22]	Y4	PAD[27]
V9	CBE[2]*	W7	PAD[19]	Y5	PAD[24]
V10	TRdy*	W8	PAD[16]	Y6	PAD[23]
V11	Lock*	W9	IRdy*	Y7	PAD[20]
V12	Par	W10	Stop*	Y8	PAD[17]
V13	PAD[14]	W11	SErr*	Y9	Frame*
V14	PAD[11]	W12	PAD[15]	Y10	DevSel*
V15	PAD[8]	W13	PAD[12]	Y11	PErr*
V16	PAD[6]	W14	PAD[9]	Y12	CBE[1]*
V17	PAD[3]	W15	PAD[7]	Y13	PAD[13]
V18	PAD[0]	W16	PAD[4]	Y14	PAD[10]
V19	SysCmd[4]	W17	PAD[1]	Y15	CBE[0]*
V20	SysCmd[5]	W18	Hit/DMAReq[3]*	Y16	PAD[5]
W1	Rst*	W19	SysCmd[8]	Y17	PAD[2]

Ball #	Signal Name	Ball #	Signal Name	Ball #	Signal Name
W2	Req*	W20	SysCmd[6]	Y18	OE64*
W3	PAD[29]	Y1	VSS	Y19	Interrupt*
W4	PAD[26]	Y2	Gnt*	Y20	SysCmd[7]
W5	CBE[3]*	Y3	PAD[30]		

BALL SEQUENCE



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

8.1 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
Top	Operating Temperature	0	85	oC
Tstg	Storage Temperature	-40	125	oC
ESD			2000	V

8.2 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
Top	Operating Temperature	0		70	oC
Cin	Input Capacitance		7.2		pF
Cout	Output Capacitance		7.2		pF

8.3 DC Electrical Characteristics Over Operating Range

(TC=0-70^{o}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	Vdd = 4.5V Vdd = 5.0V Vdd = 5.5V	0.52 0.54 0.56		0.60 0.61 0.62	mV
Icc	Operating Current				400	mA

NOTE:

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

8.4 Package Thermal Characteristics 256PQFP Package

Symbol	Max.	Unit
Θjc	12	C/W
Θja	35	C/W

8.5 Package Thermal Characteristics 272 BGA Package

Symbol	Max.	Unit
Θjc	9.6	C/W
Θja	22	C/W

9. AC TIMING - *PRELIMINARY/SUBJECT TO CHANGE*

(TC= 0-70^{O}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	nS
t5	TClk	Fall Time		3	nS
t6	Rst*	Active	10		TClk
t7	Hit/DMAReq[3]*, LEAdrE/DMAReq[2]*, DMAReq[1]*/ParErr*, AD[31:0]	Setup	5		nS
t8	DMAReq[0]*/Ready*,	Setup	11		nS
t9	WrRdy*, ValidIn*, OE64* SysCmd[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*, SysAD[63:0]	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*, Hit/DMAReq[3]*, LEAdrE/DMAReq[2]*, DMAReq[1]*/ParErr*, SysAD[63:0], SysCmd[8:0], AD[31:0], DMAReq[0]*/Ready*	Hold	1		nS
t18	ValidOut*, Release*,SysAD[63:0], SysCmd[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.			
	All Inputs including JTRST*, JTDI, JTMS	Setup to JTCLK	12		nS
	All Inputs including JTRST*, JTDI, JTMS	Hold from JTCLK	1		nS
	All Outputs	Delay from JTCLK Float to Valid delay Drive to Float delay	2 2 2	10 15 15	nS
	JTDO	Delay from JTCLK falling	2	10	nS

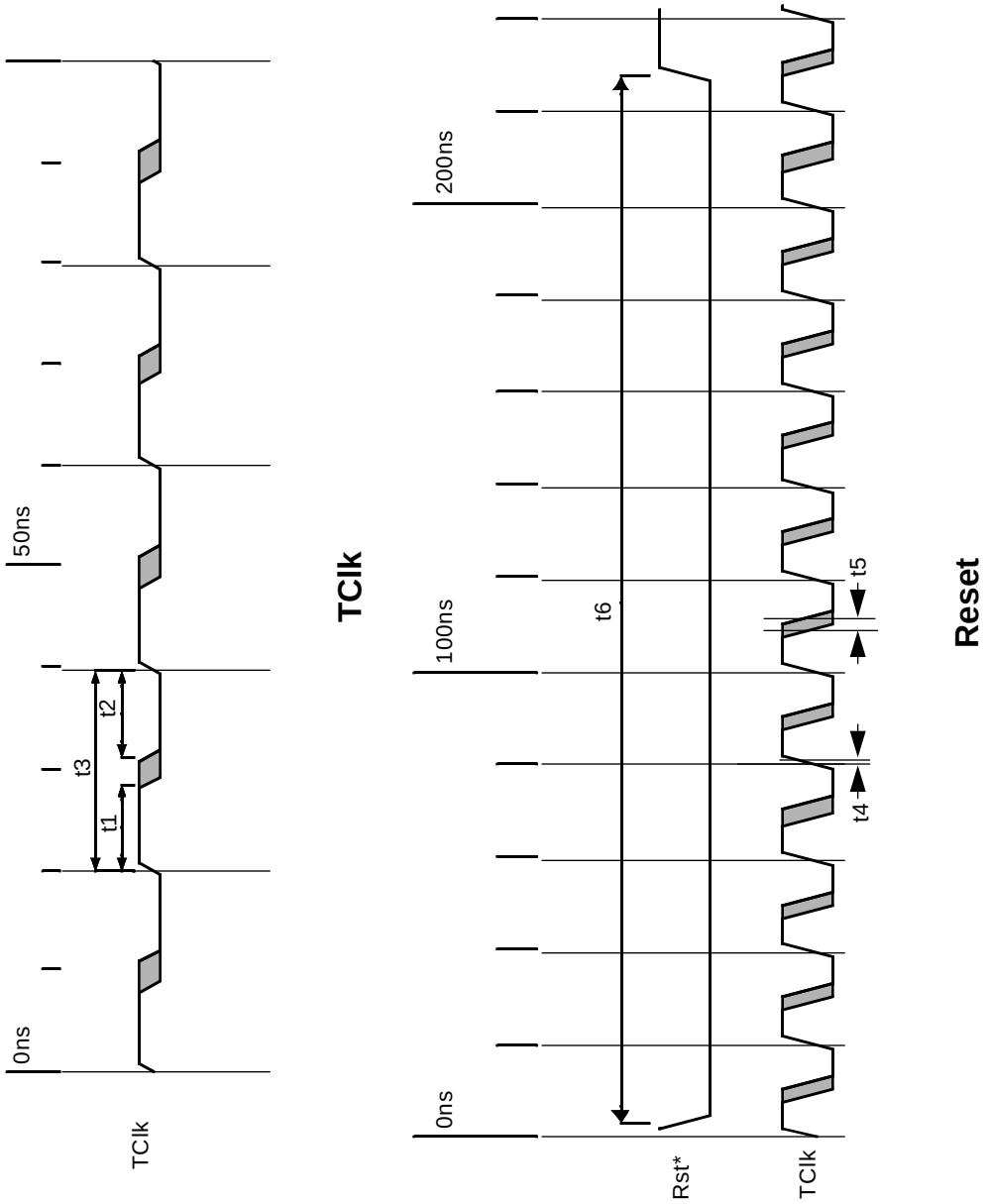
Notes:

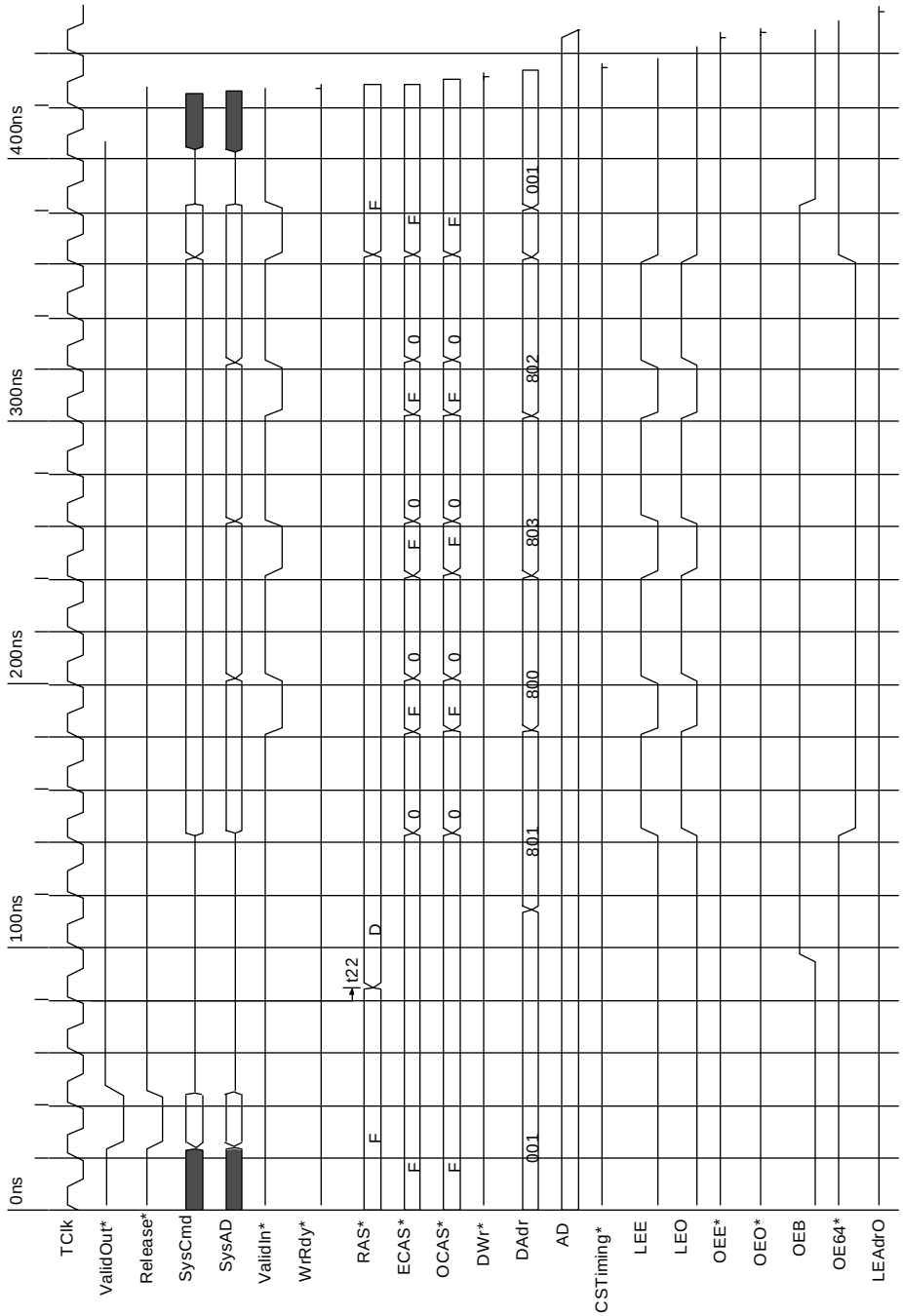
1. All Delays, Setup, and Hold times are referred to TClk rising edge, unless stated otherwise.

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

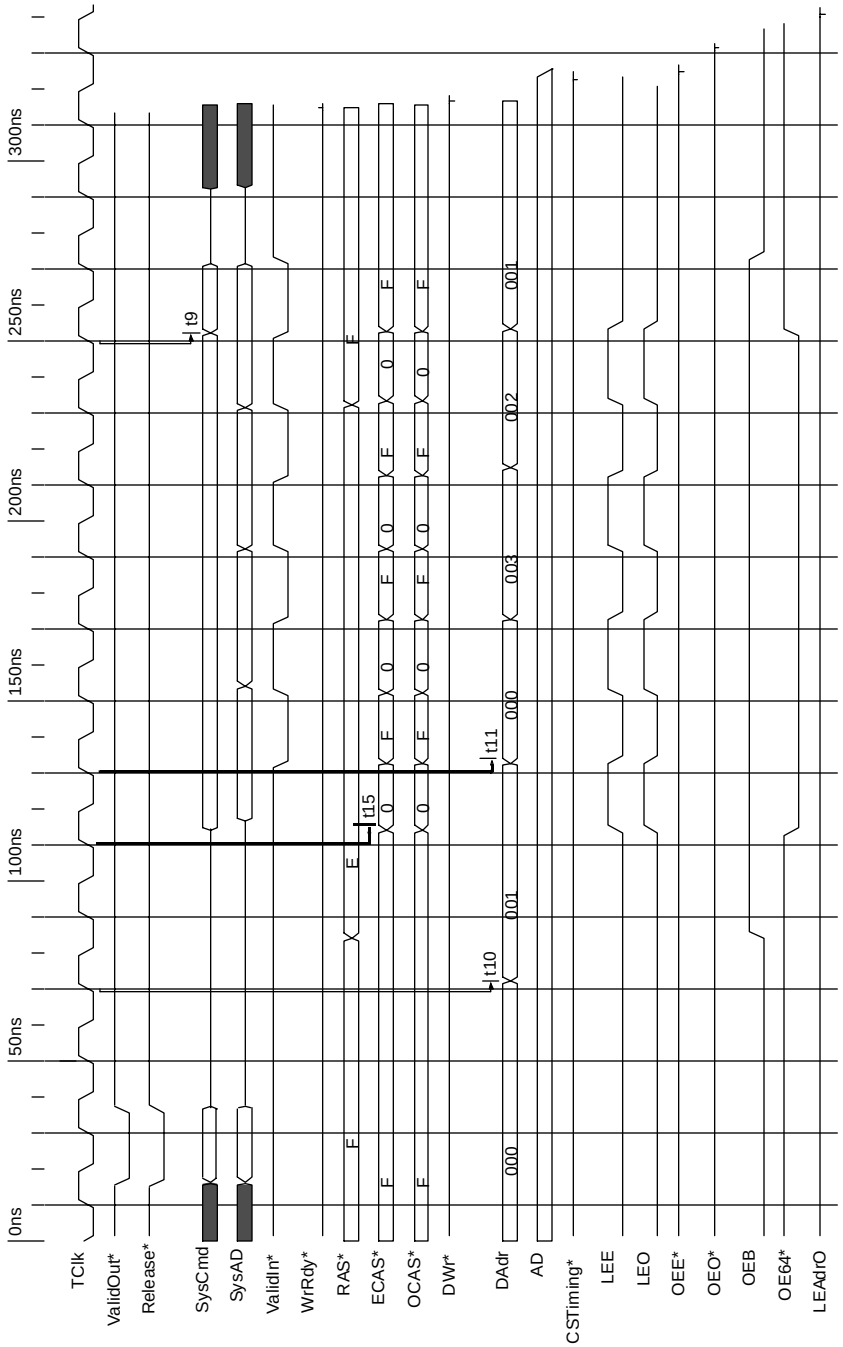
3. TClk frequency must be equal or greater than PClk frequency.

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

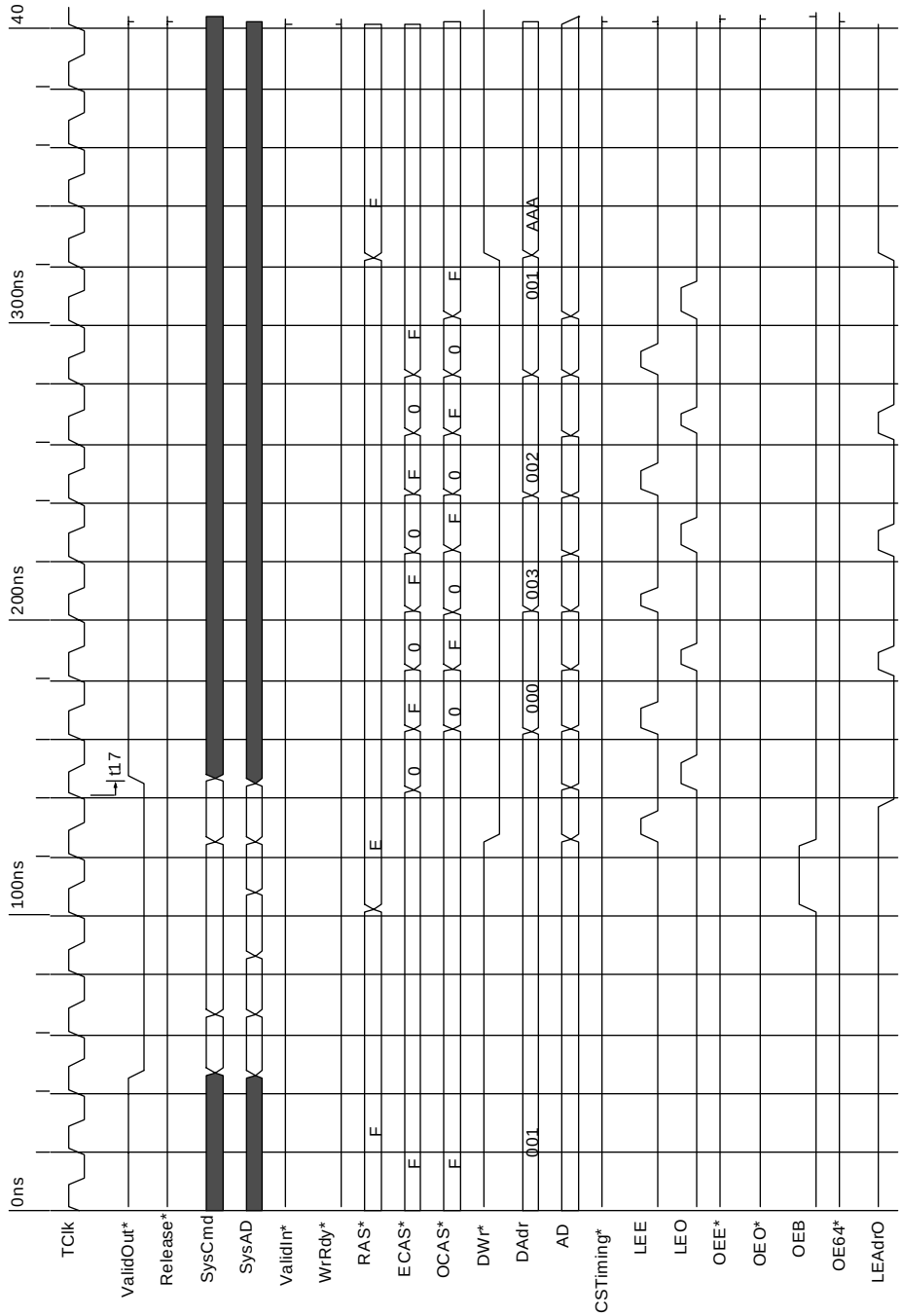

TClk
50ns
0ns
t3
t2
t1
TClk
Reset
200ns
100ns
0ns
t6
t5
t4
Rst*
TClk



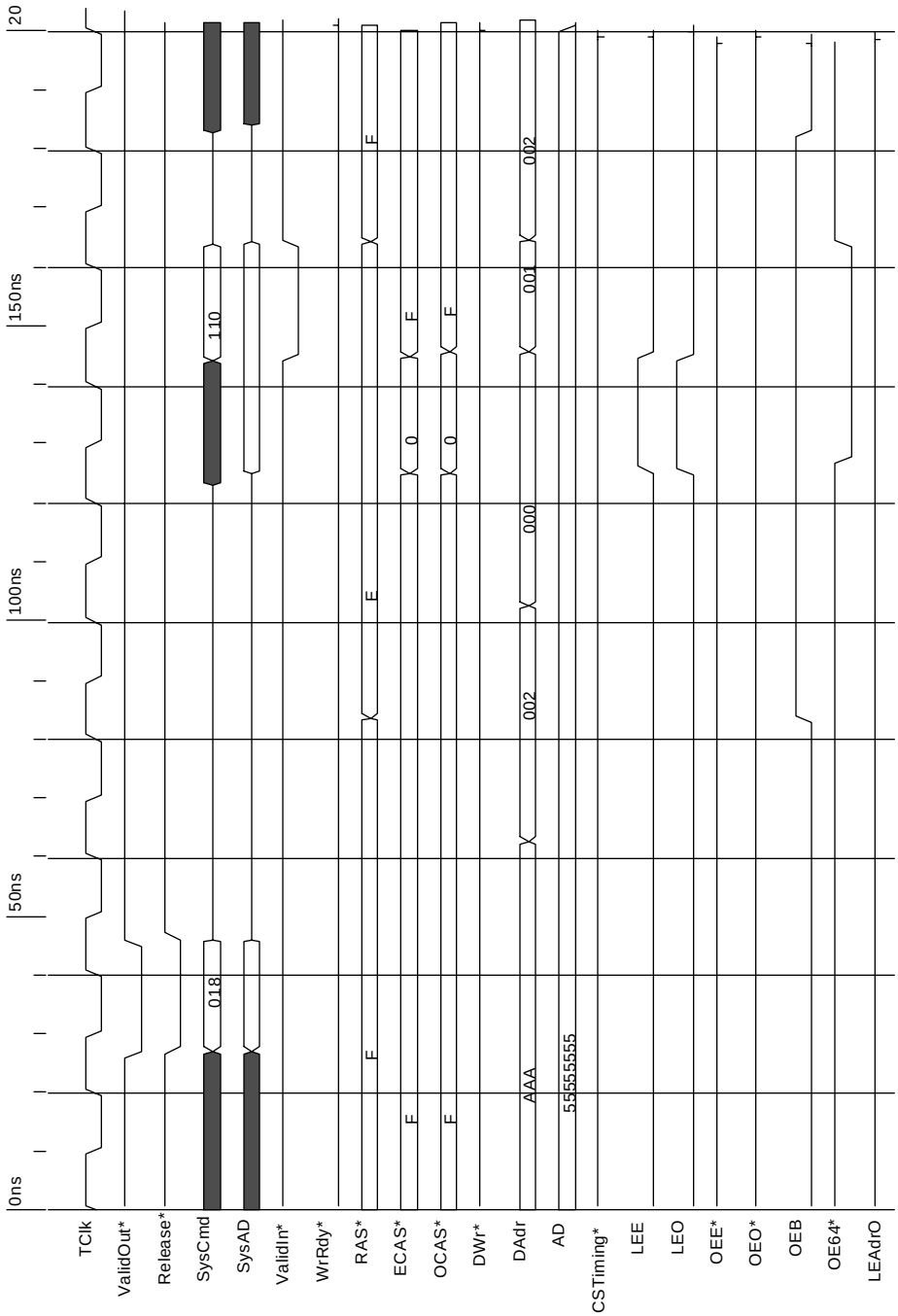
CPU Burst Read from 64-Bit Standard DRAM (CAS Low 2 Cycles)



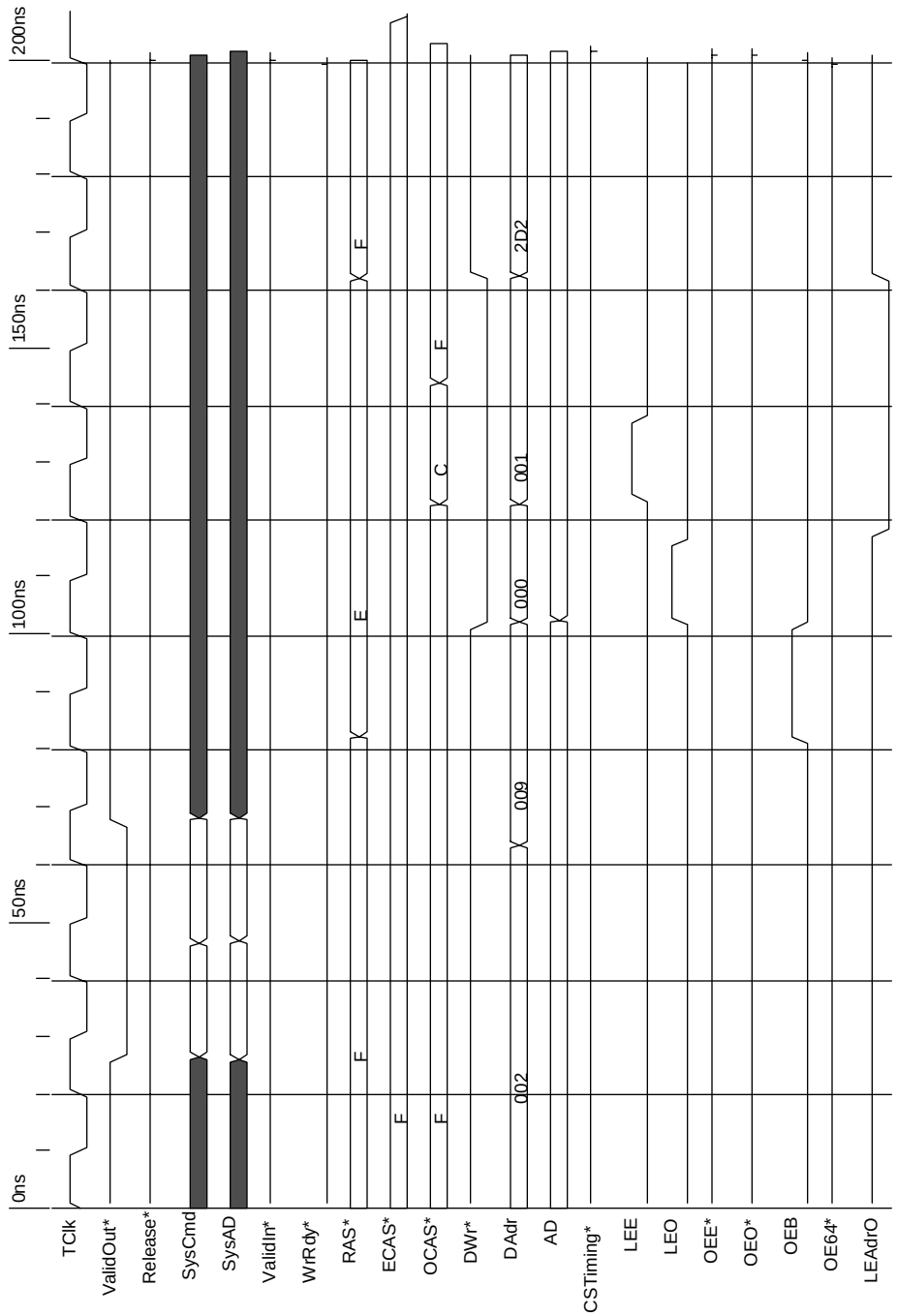
CPU Burst Read from 64-Bit Standard DRAM (Fastest RAS & CAS)



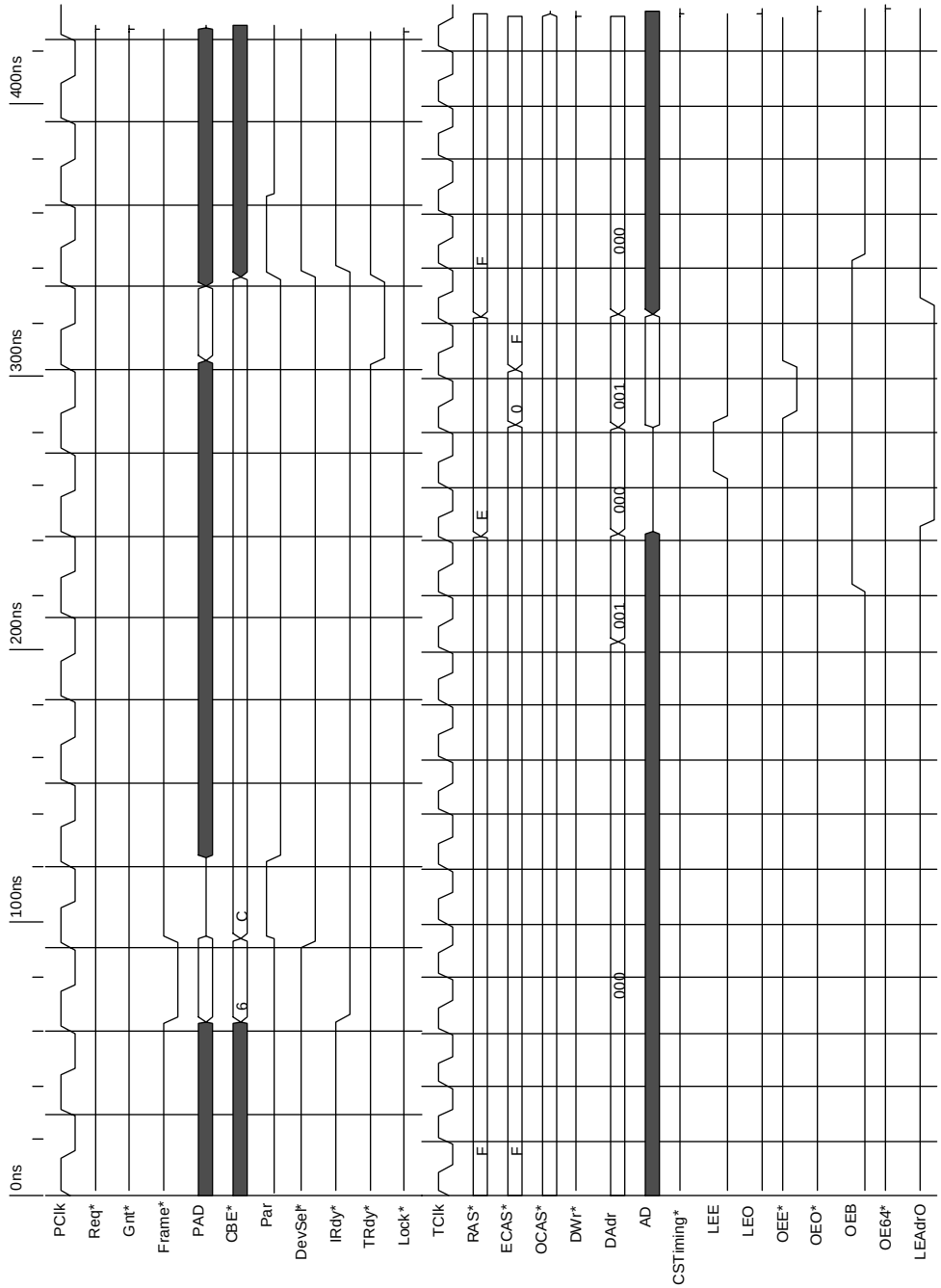
CPU Burst Write to 64-Bit Standard DRAM (Fastest RAS & CAS)



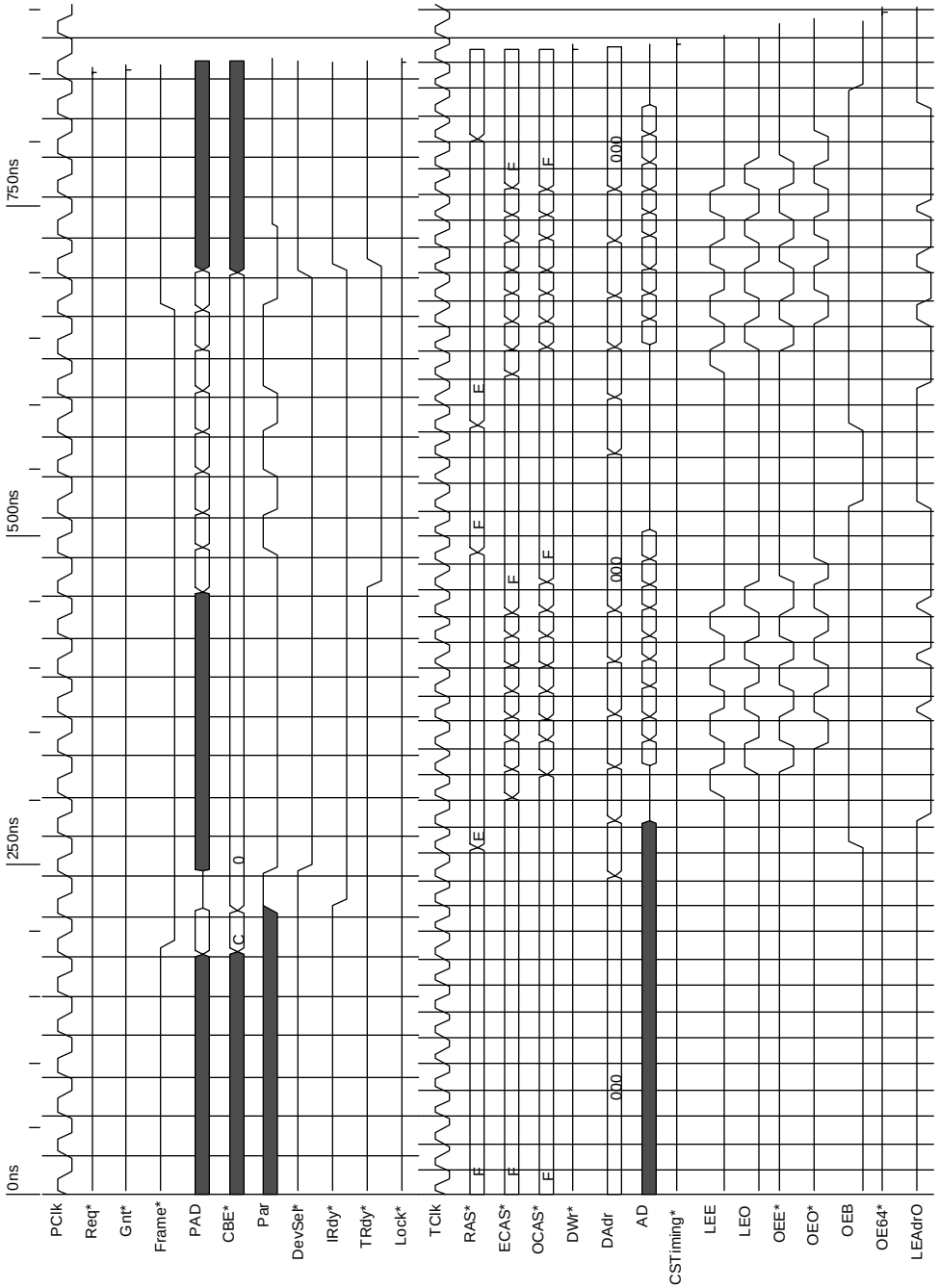
CPU Byte Read from 64-Bit Standard DRAM (Fastest RAS & CAS)



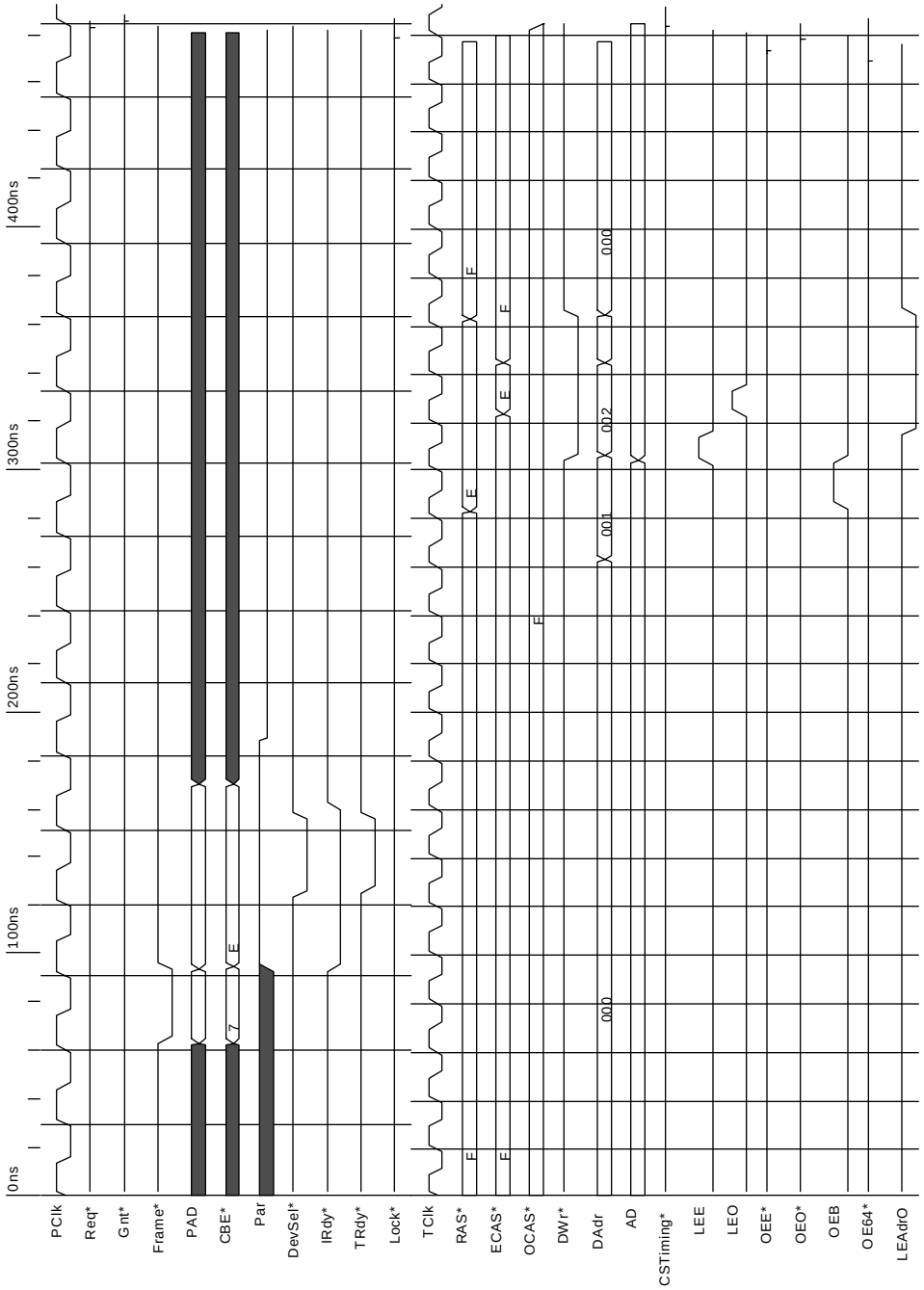
CPU 2 Bytes Write to 64-Bit Standard DRAM (Fastest RAS & CAS)



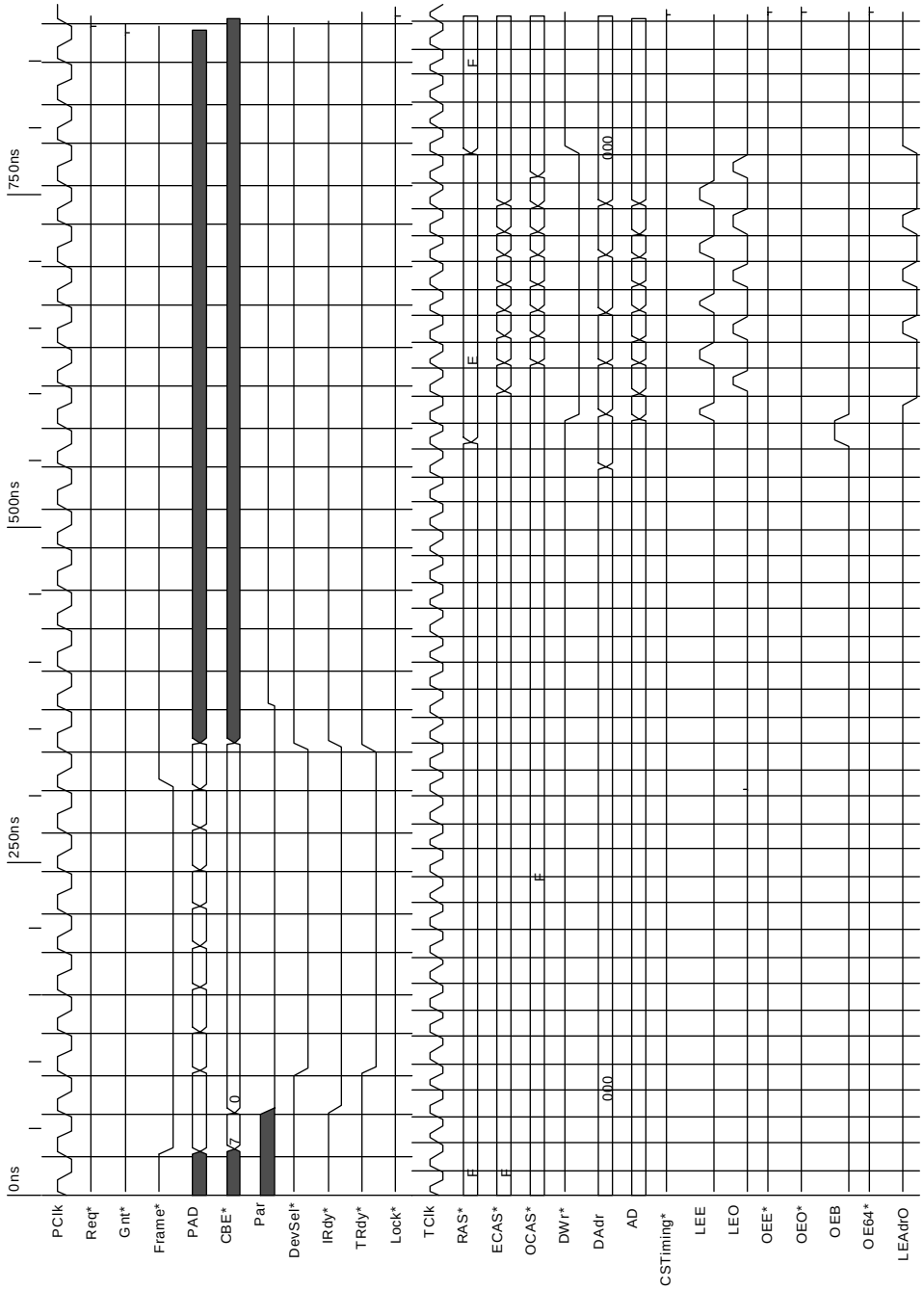
PCI 2 Bytes Read from Standard DRAM



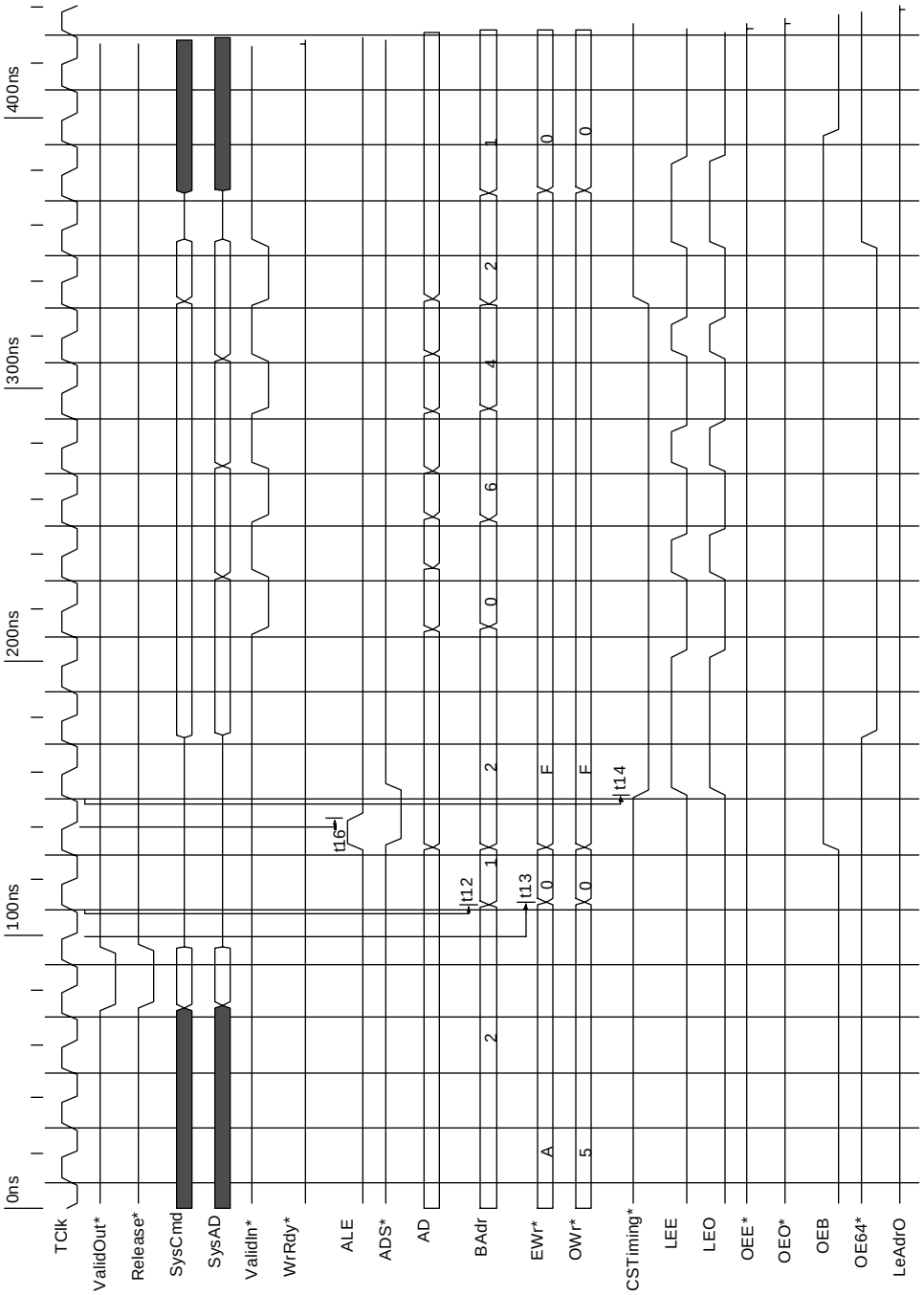
PCI 32 Bytes Read Multiple from DRAM



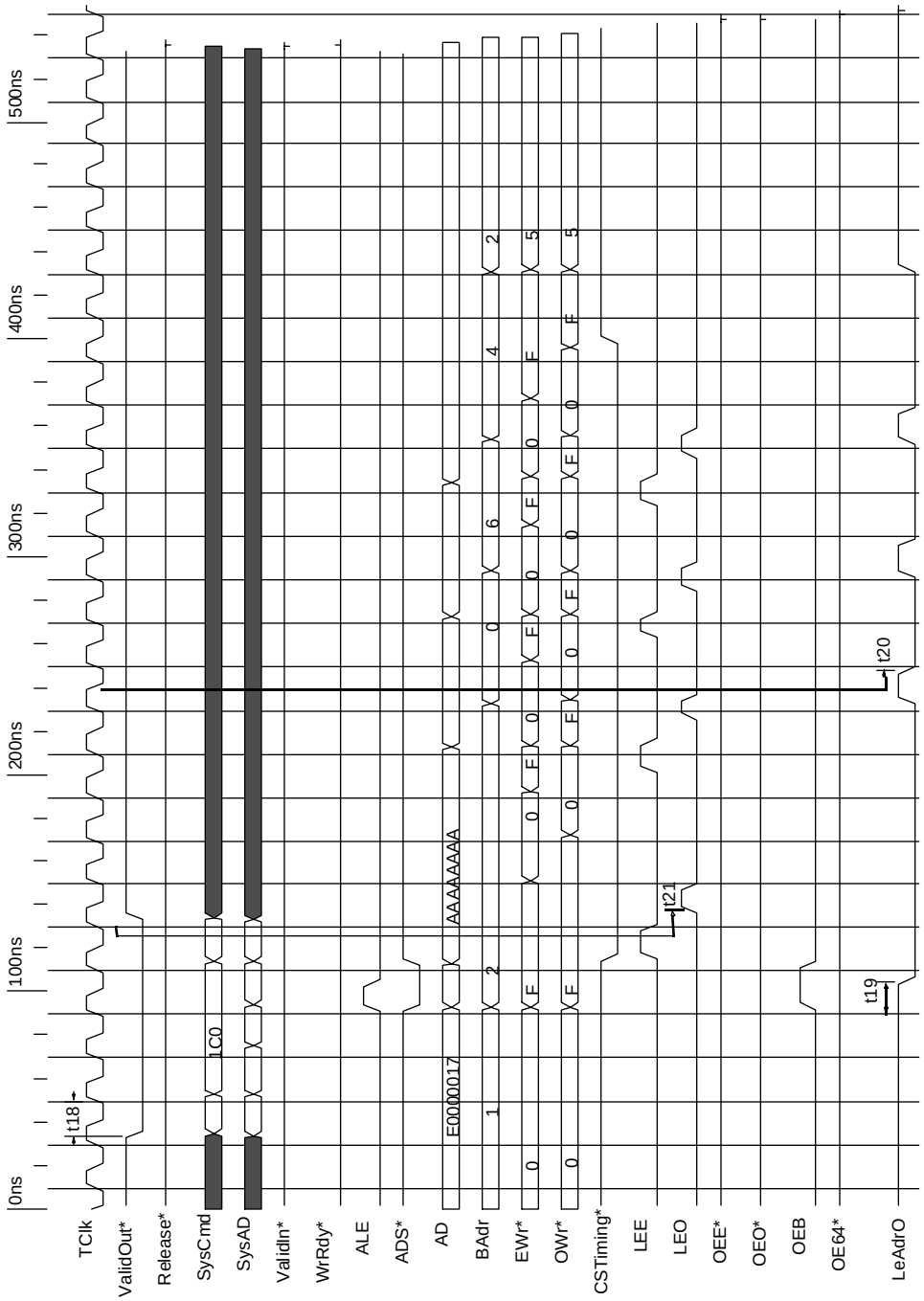
PCI Byte Write to DRAM



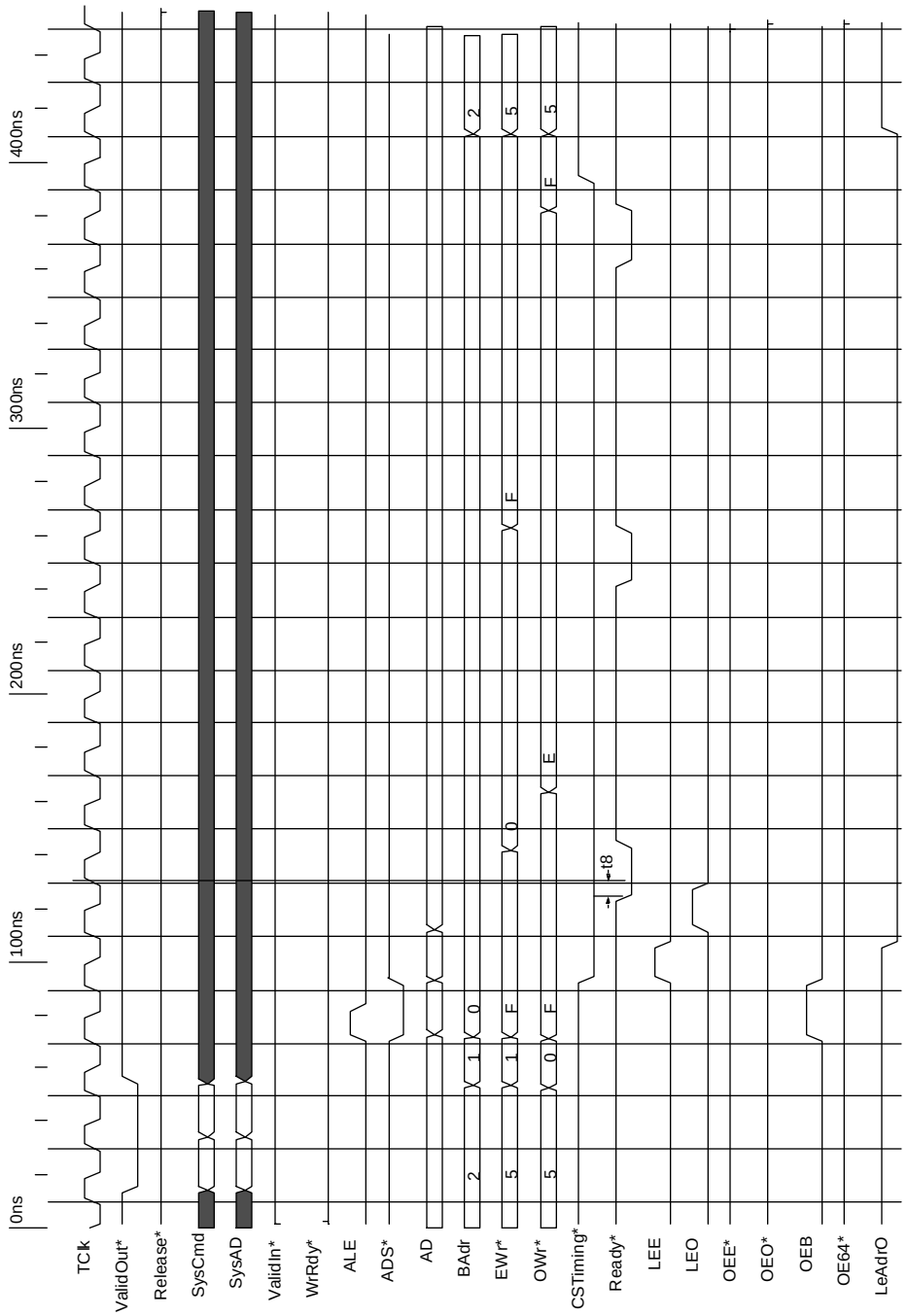
PCI 32 Bytes Write to DRAM



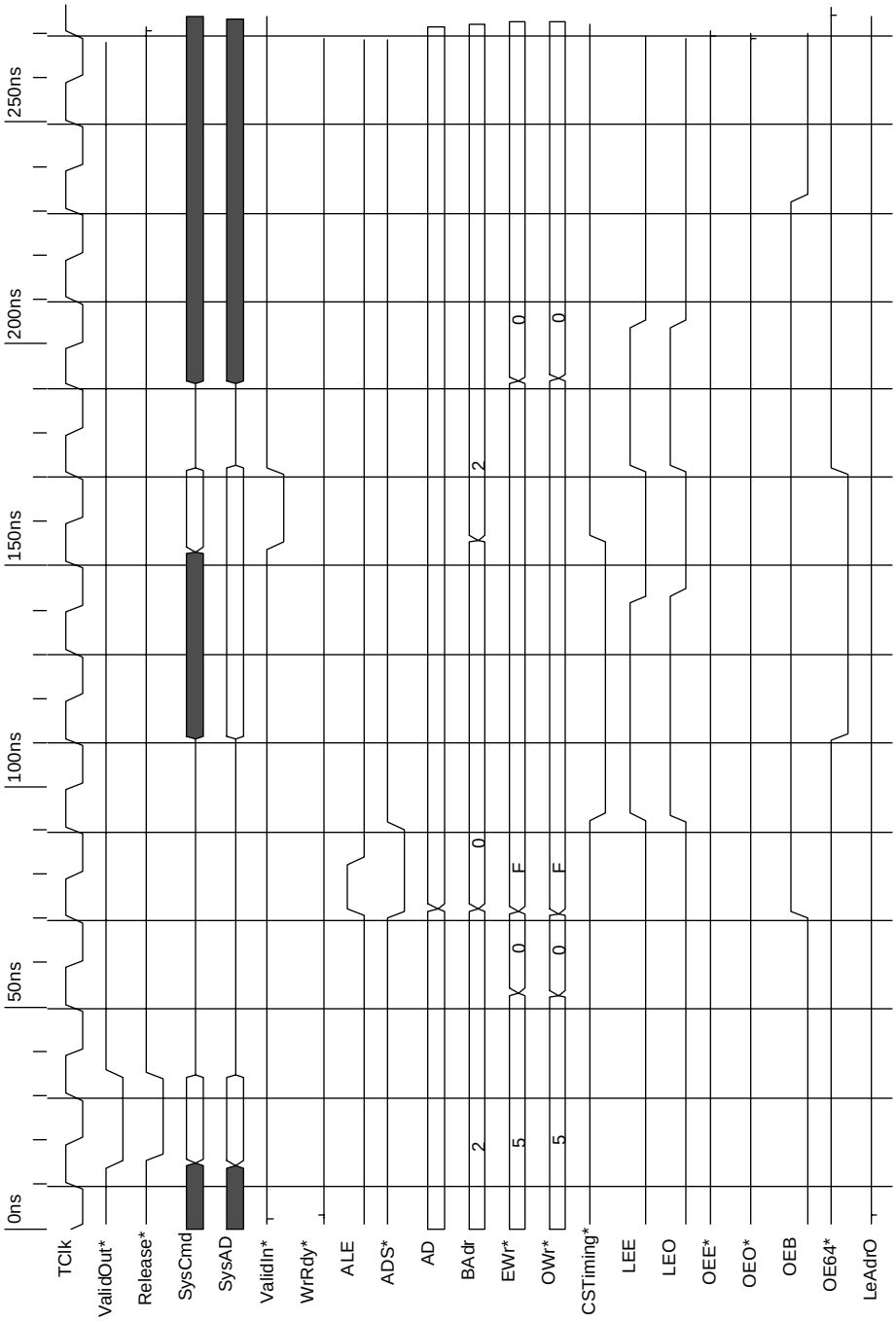
CPU Burst Read from Device (AccToFirst =4, AccToNext=2)



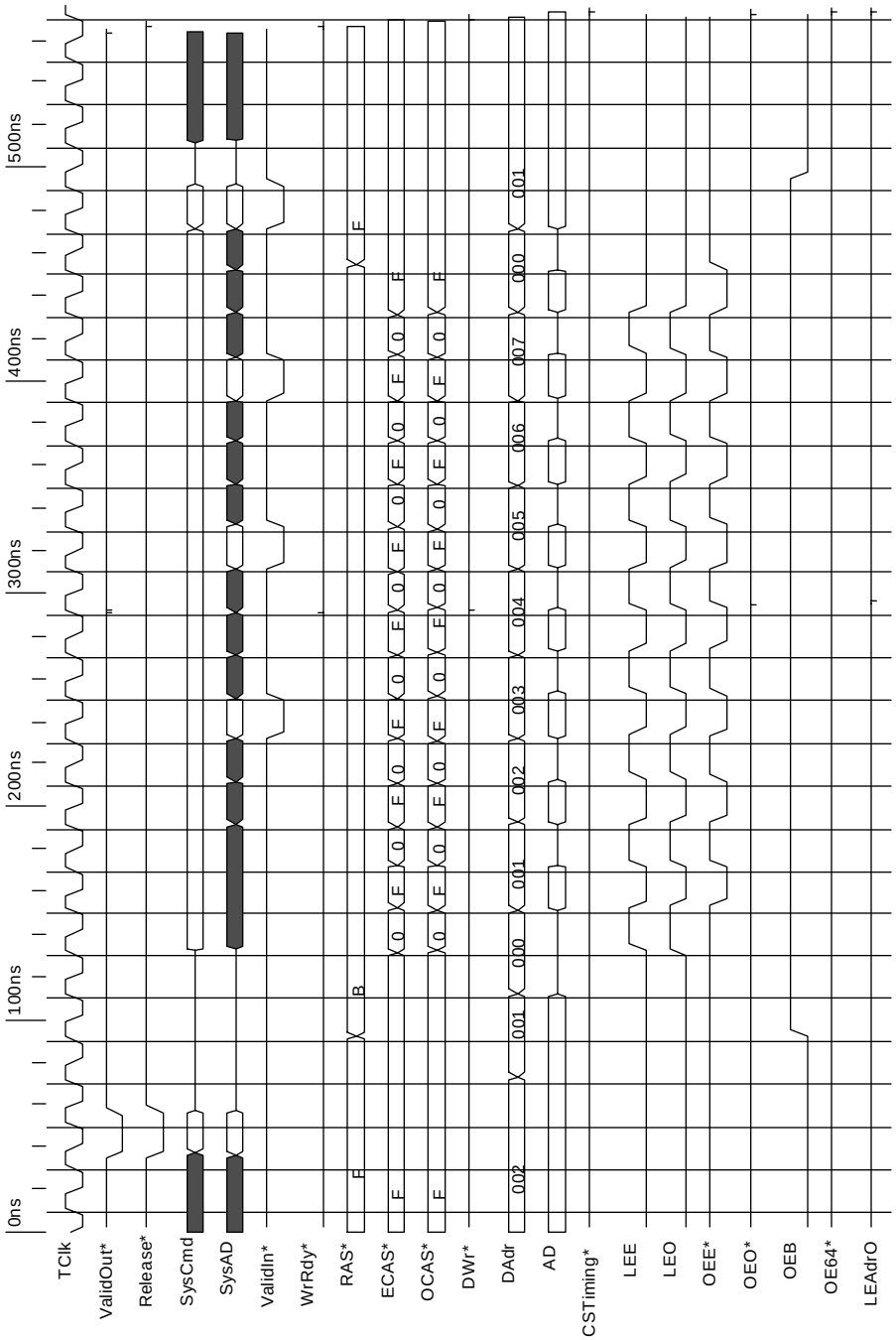
CPU Burst Write to 64-Bit Device (ADStoWr=3, WrActive=2, WrHigh=1)



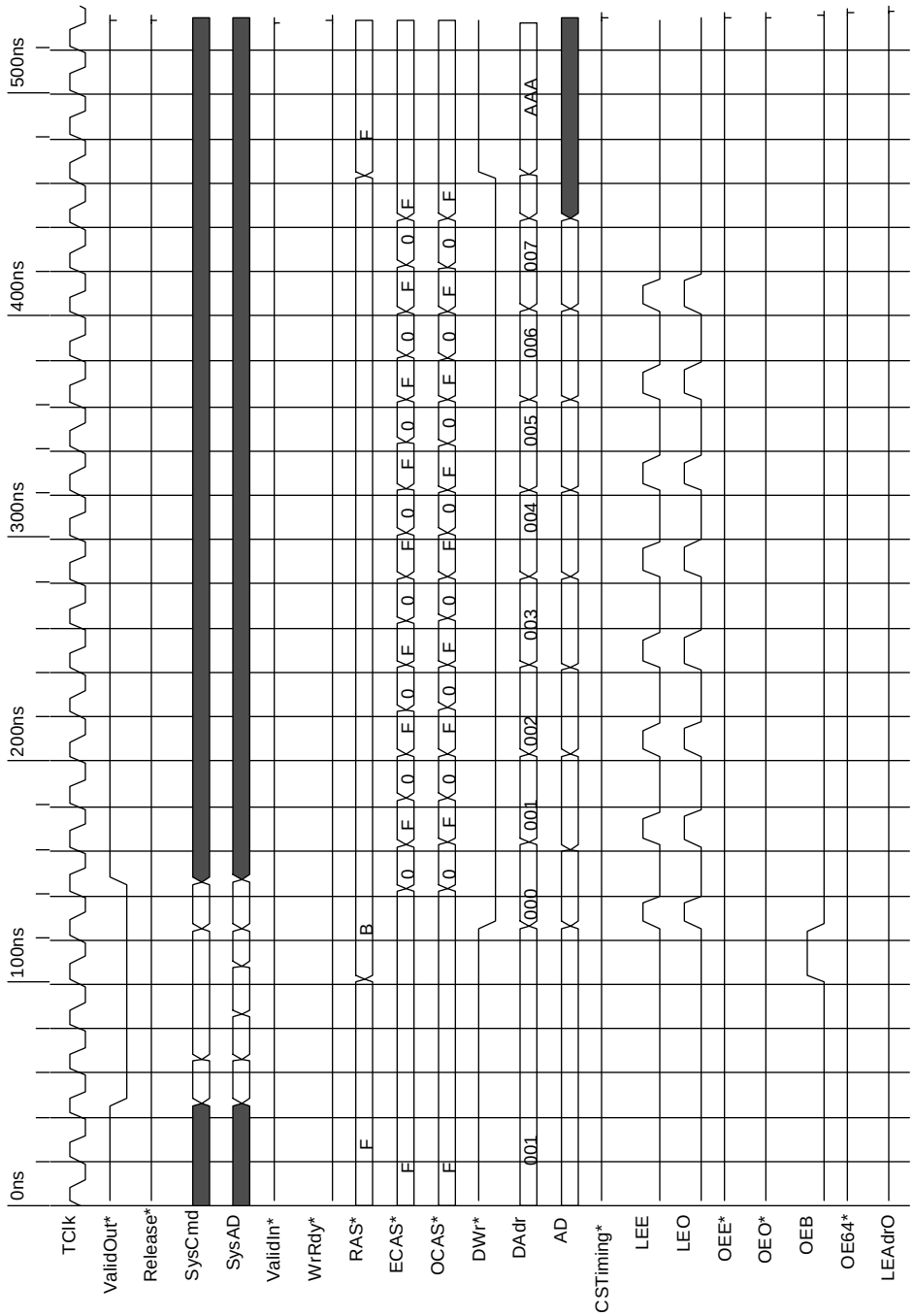
CPU 5 Bytes Write to 64-Bit Device (ADSToWr=3, WrActive=2, WrHigh=1)



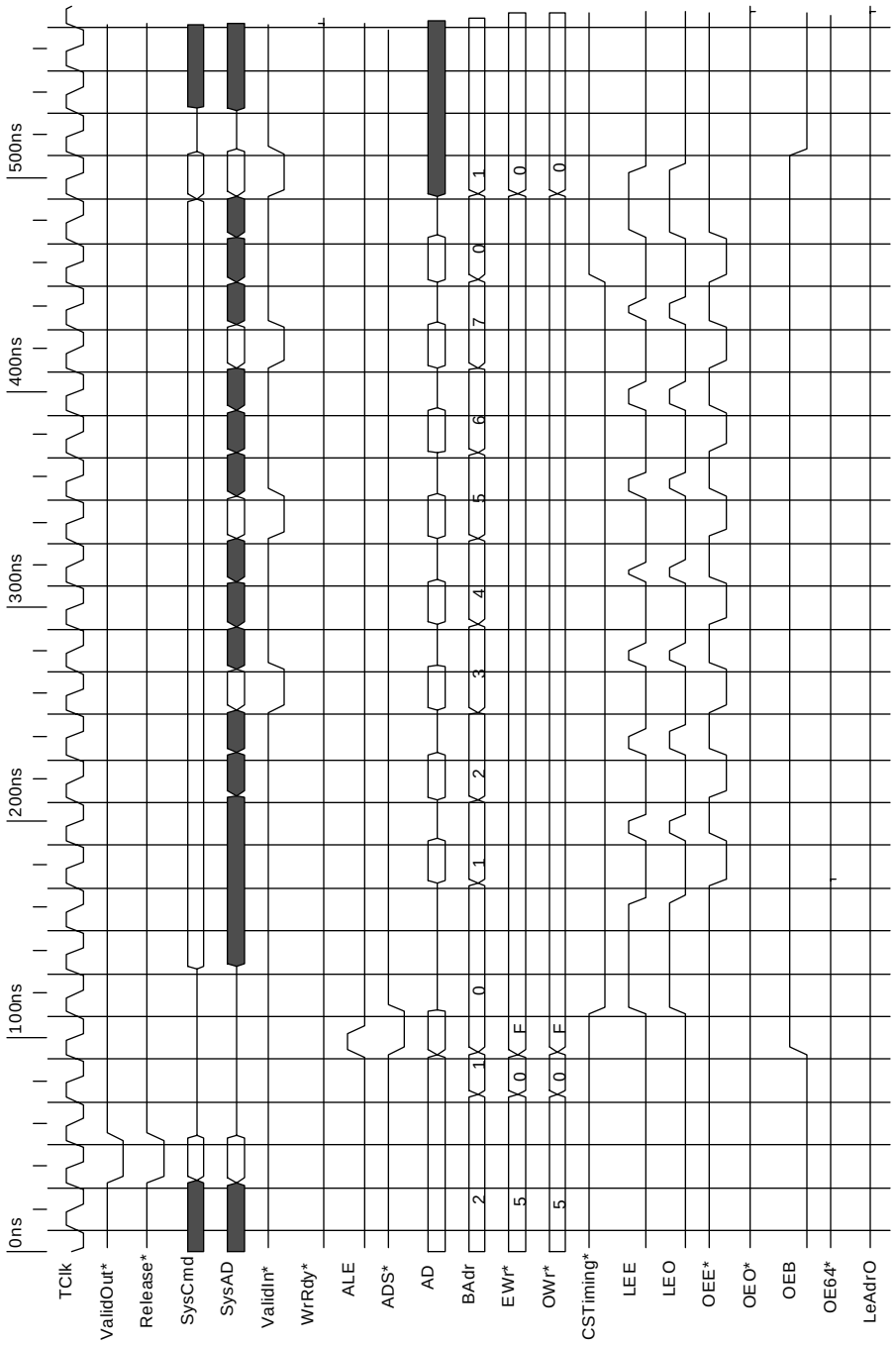
CPU 3 Bytes Read from 64-Bit Device (ADSToWr=3, WrActive=2, WrHigh=1)



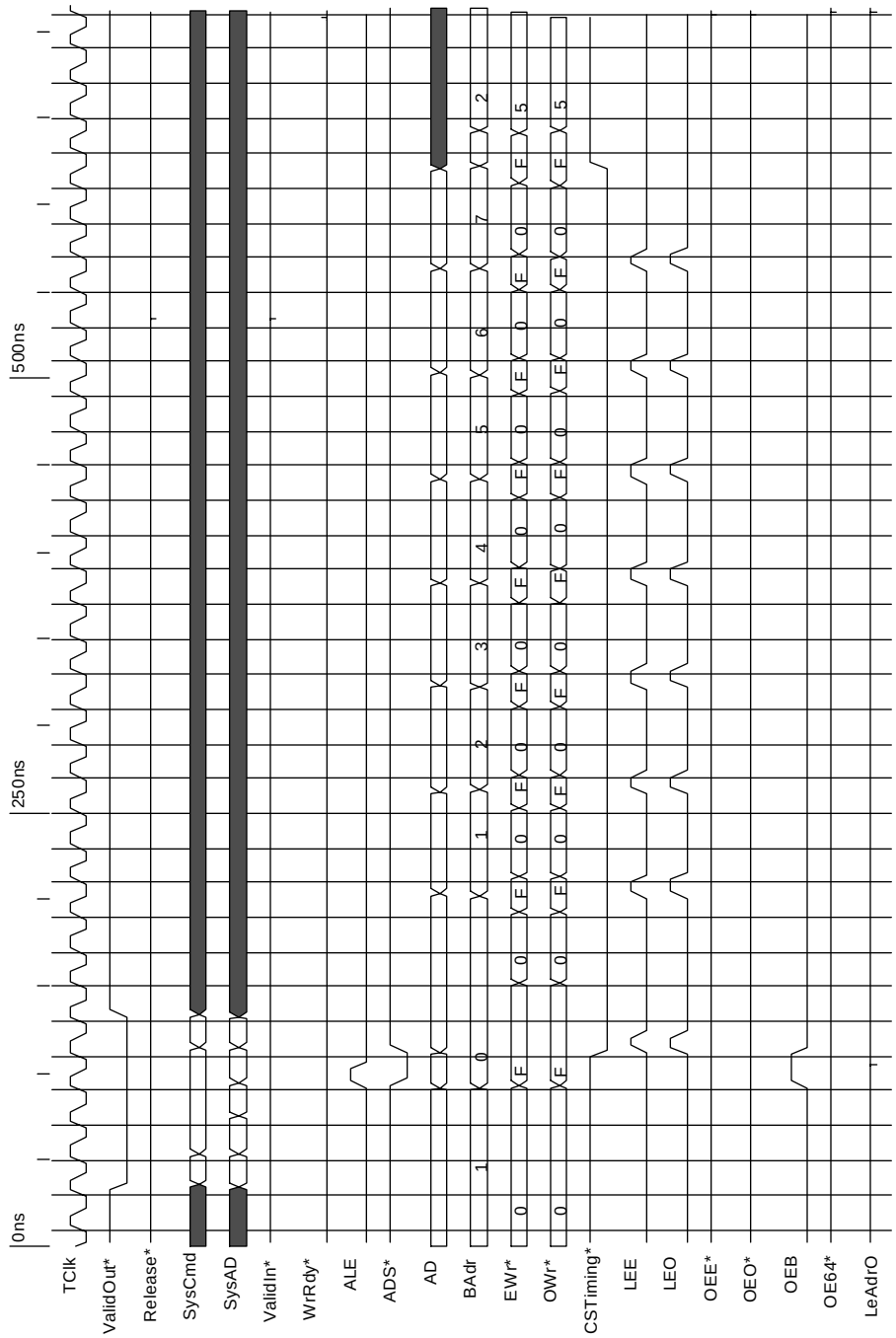
CPU Burst Read from 32-Bit DRAM (Fastest RAS & CAS)



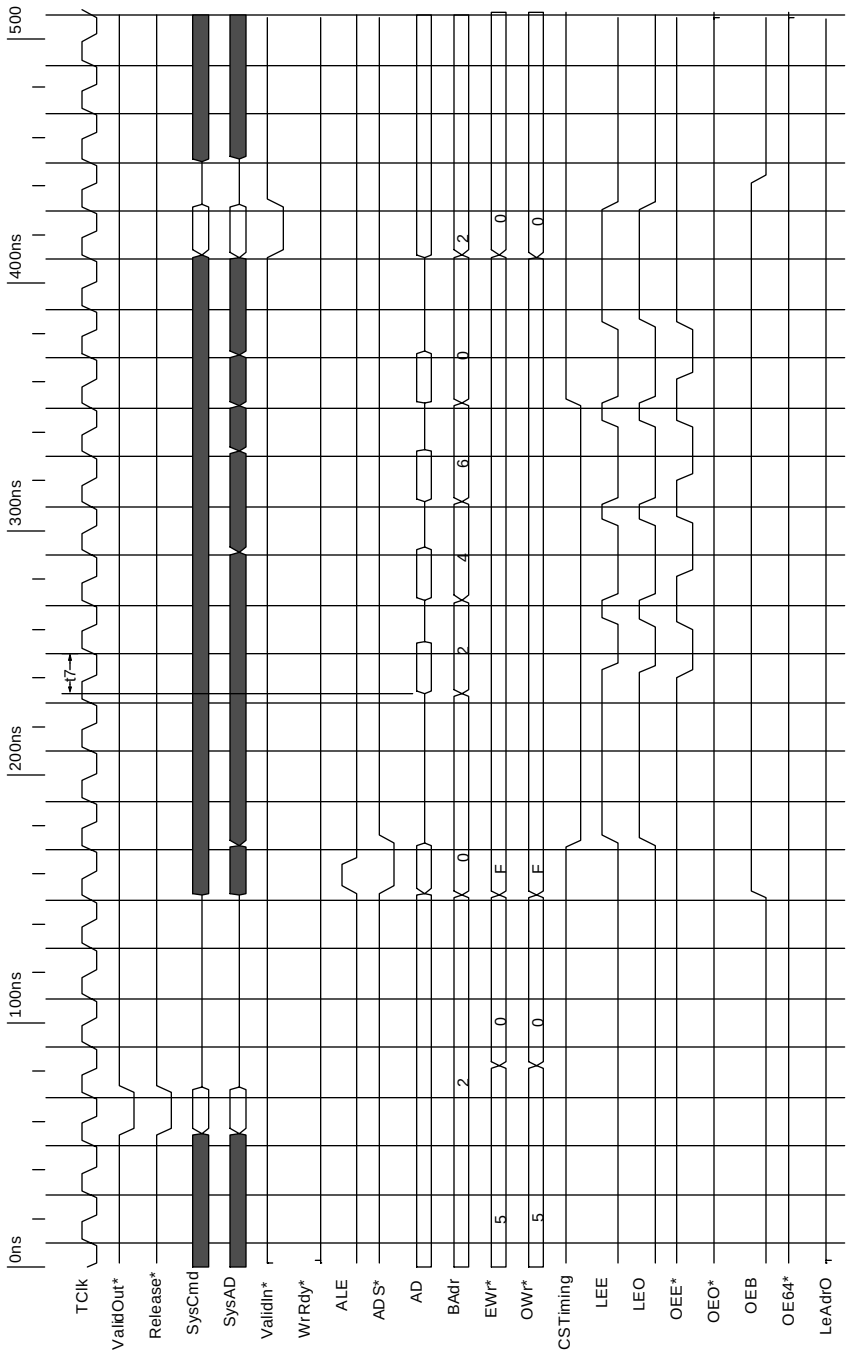
CPU Write Burst to 32-Bit DRAM



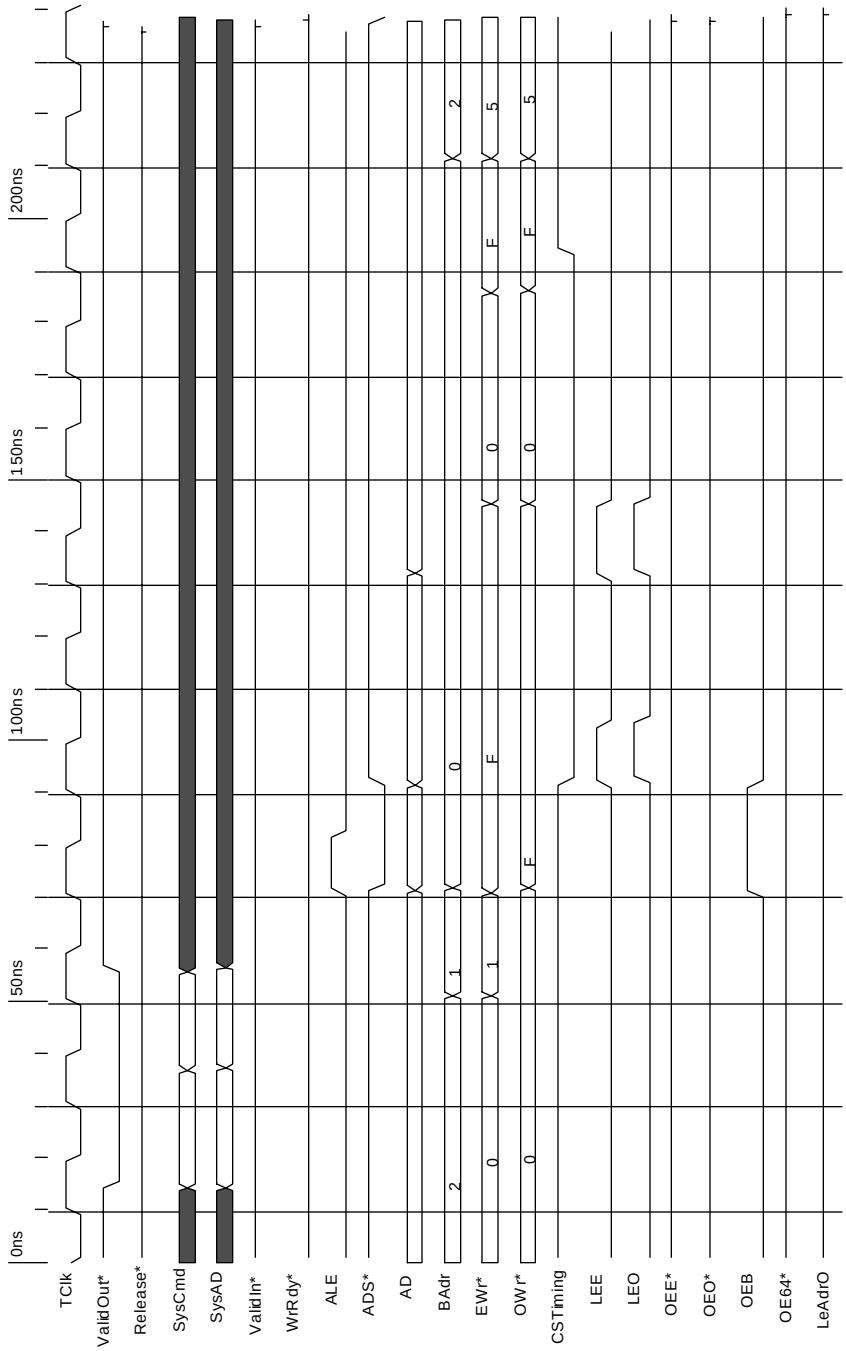
CPU Burst Read from 32-Bit Device (AccToFirst=4, AccToNext=2)



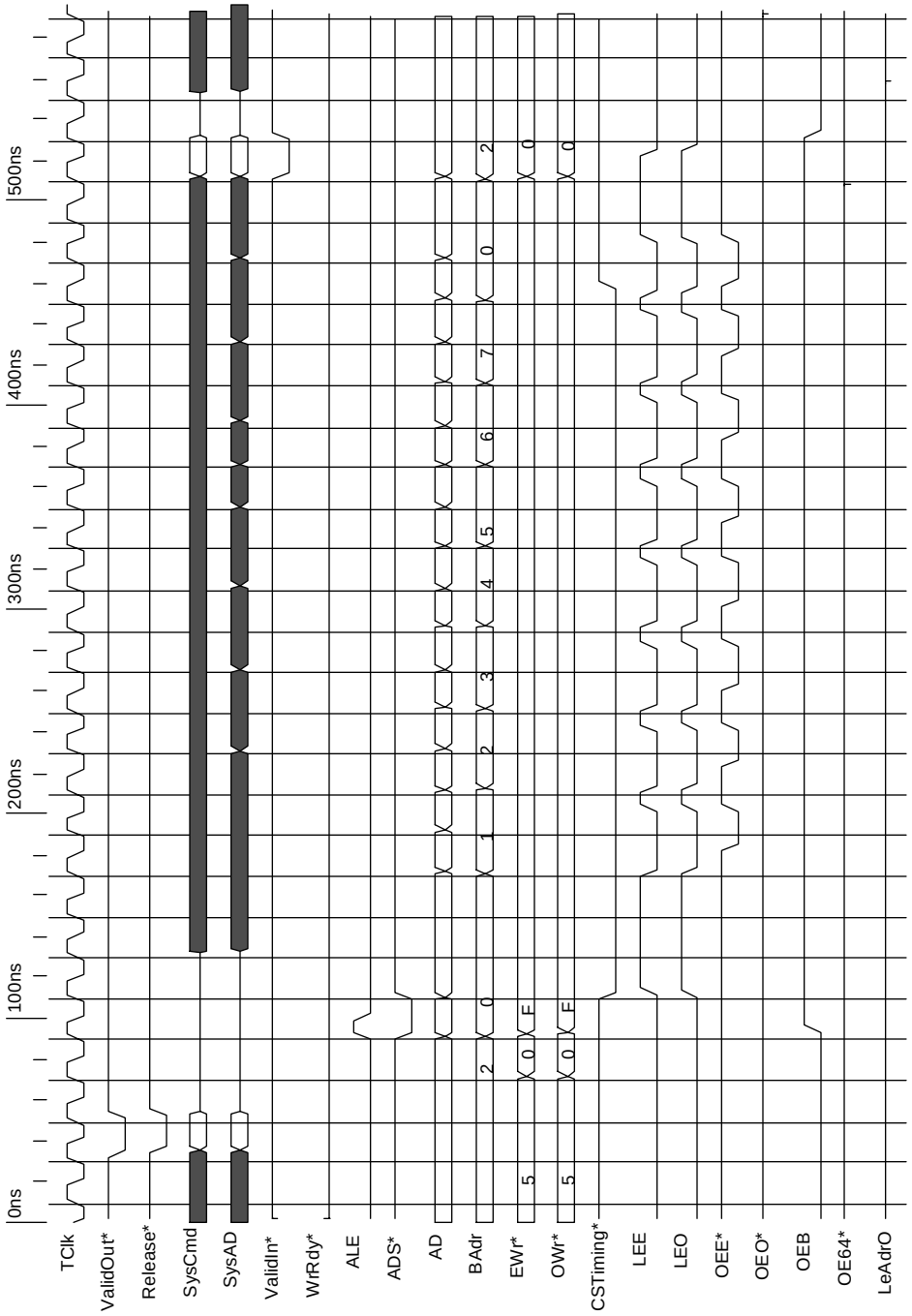
CPU Burst Write to 32-Bit Device (ADStoWr=3, WrActive=2, WrHigh=1)



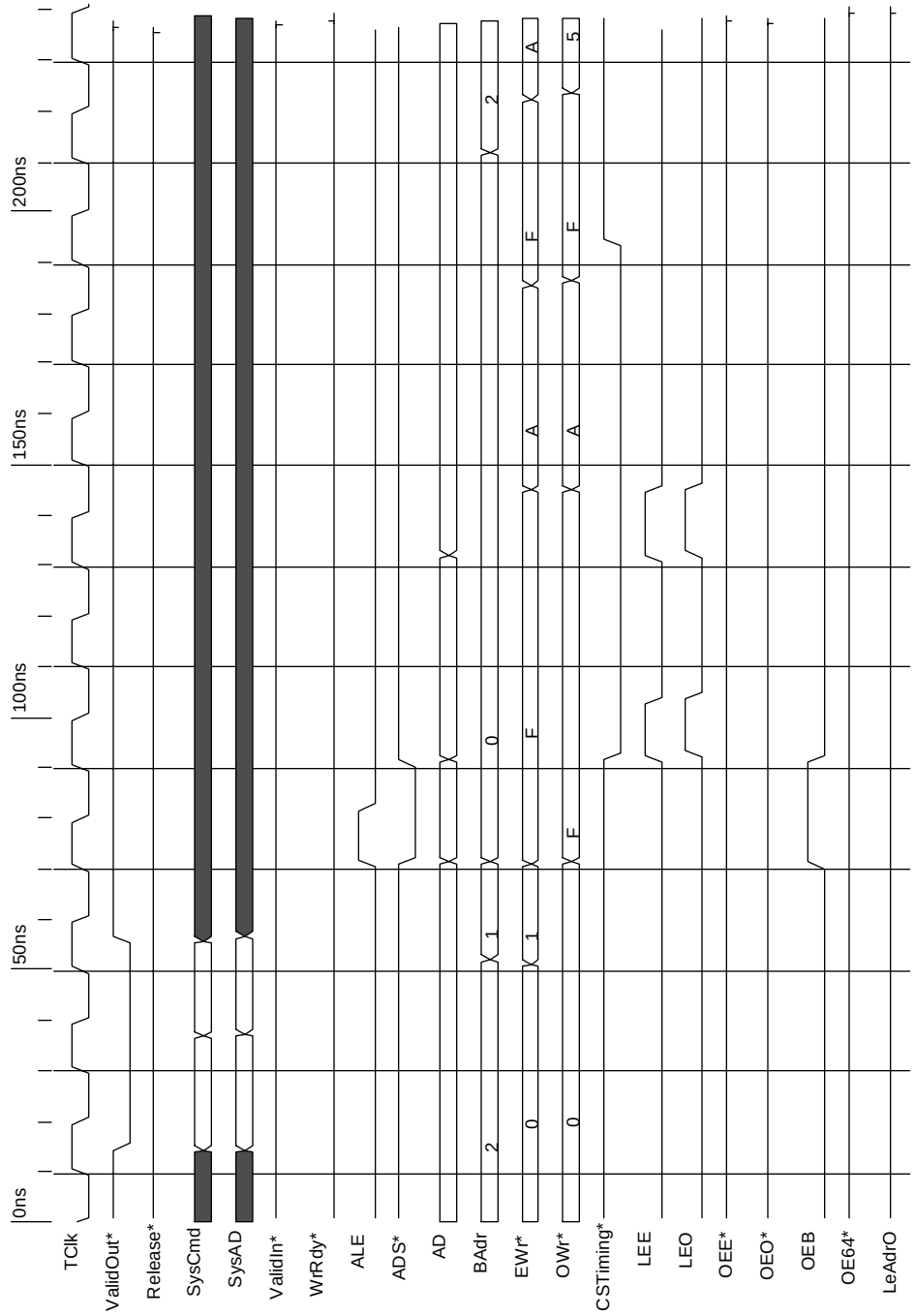
CPU 8 Bytes Read from 16-Bit Device (AccToFirst=4, AccToNext=2)



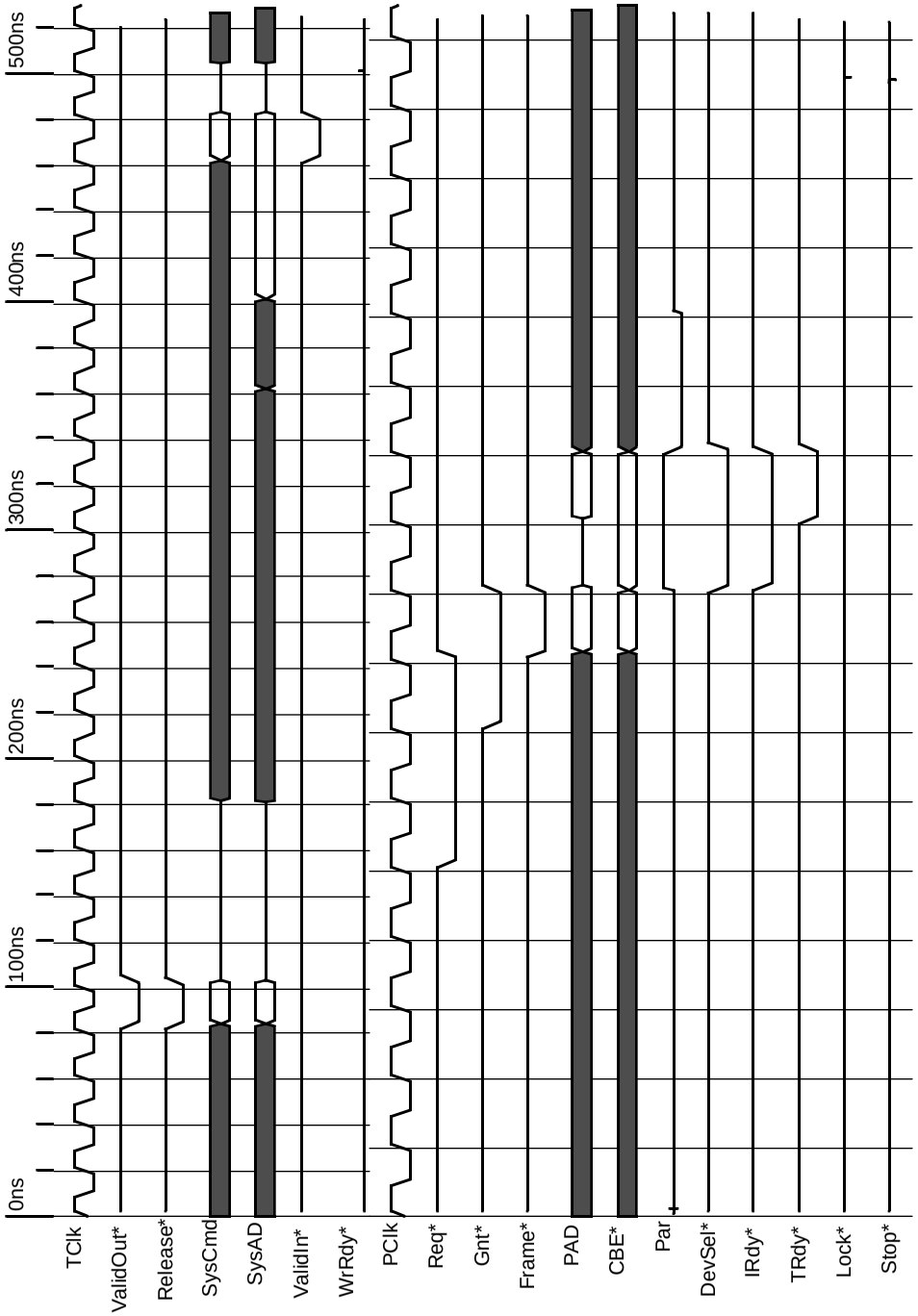
CPU 2 Bytes Write to 16-bit Device (ADStoWr=3, WrActive=2, WrHigh=1)



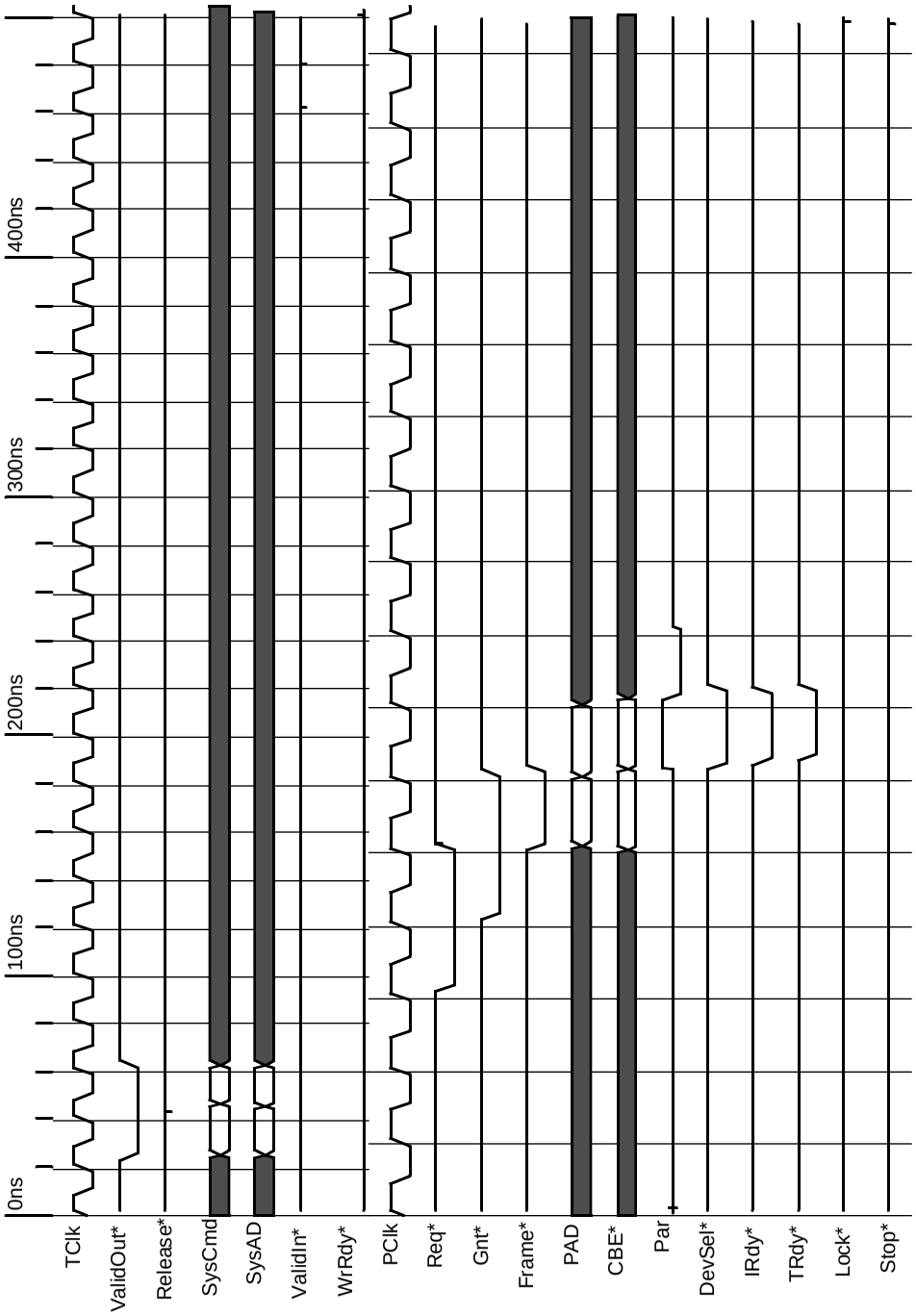
CPU 8 Bytes Read from 8-Bit Device (AccToFirst=4, AccToNext=2)



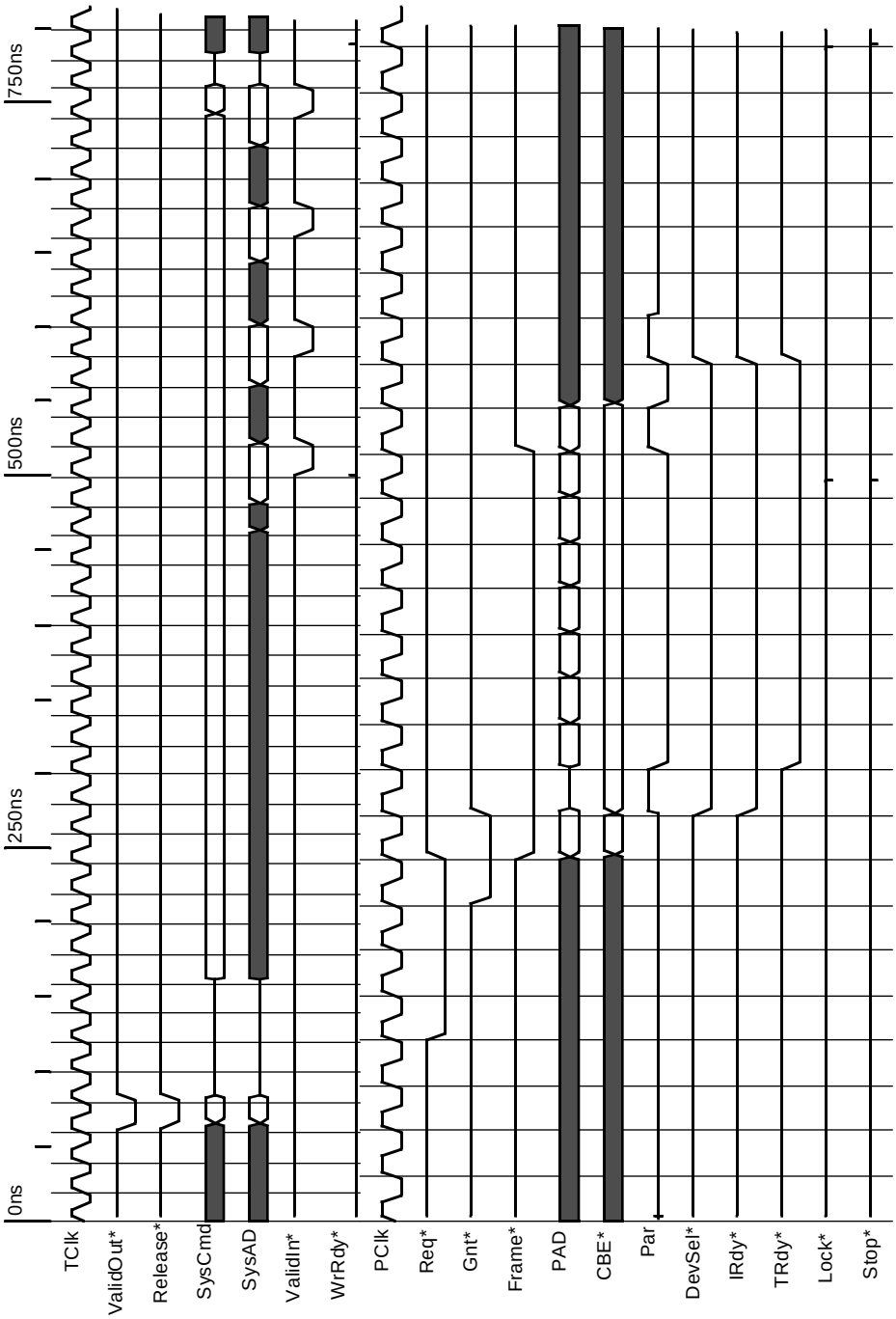
CPU Byte Write to 8-Bit Device (ADStoWr=3, WrActive=2, WrHigh=1)



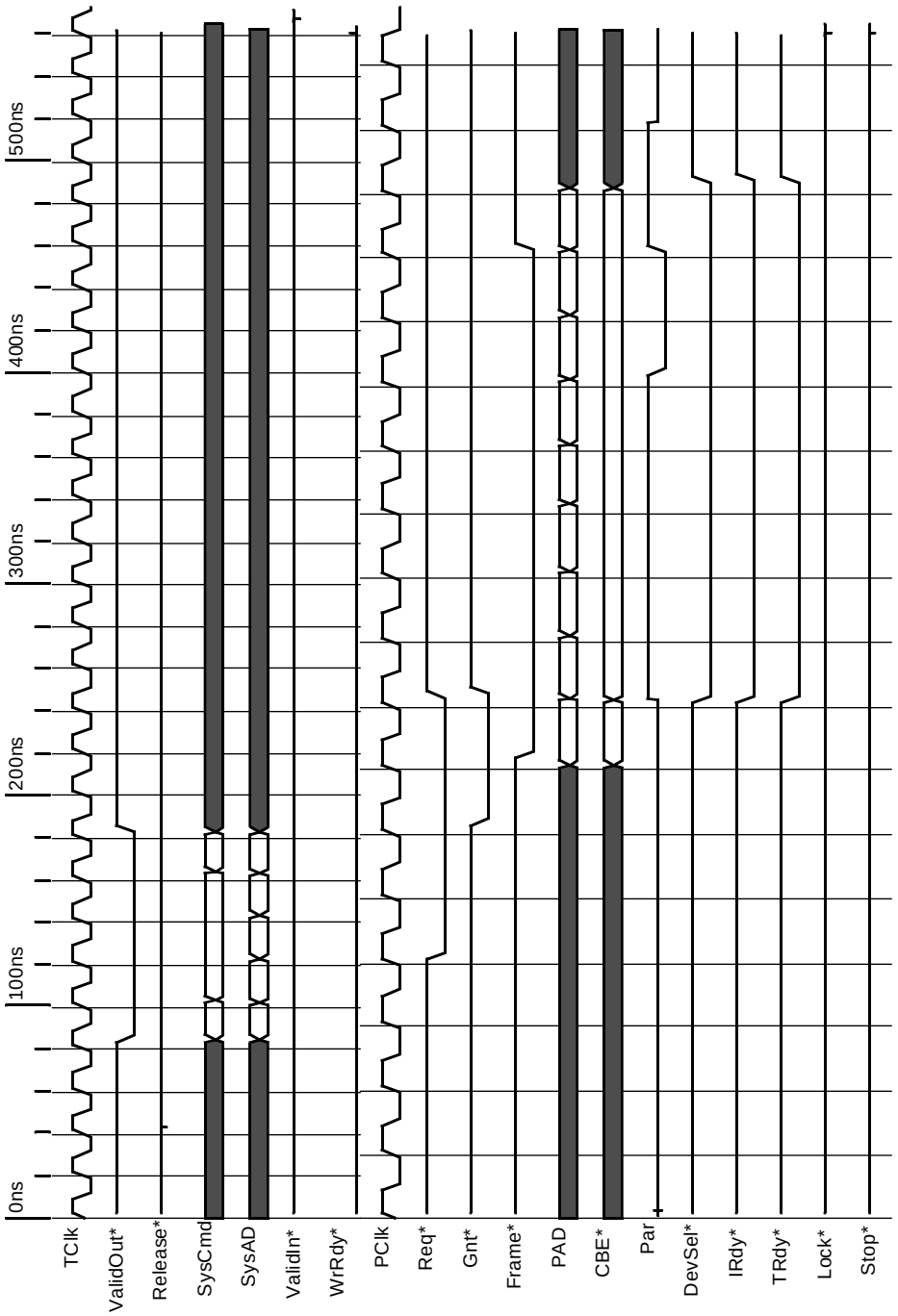
CPU Byte Read from PCI



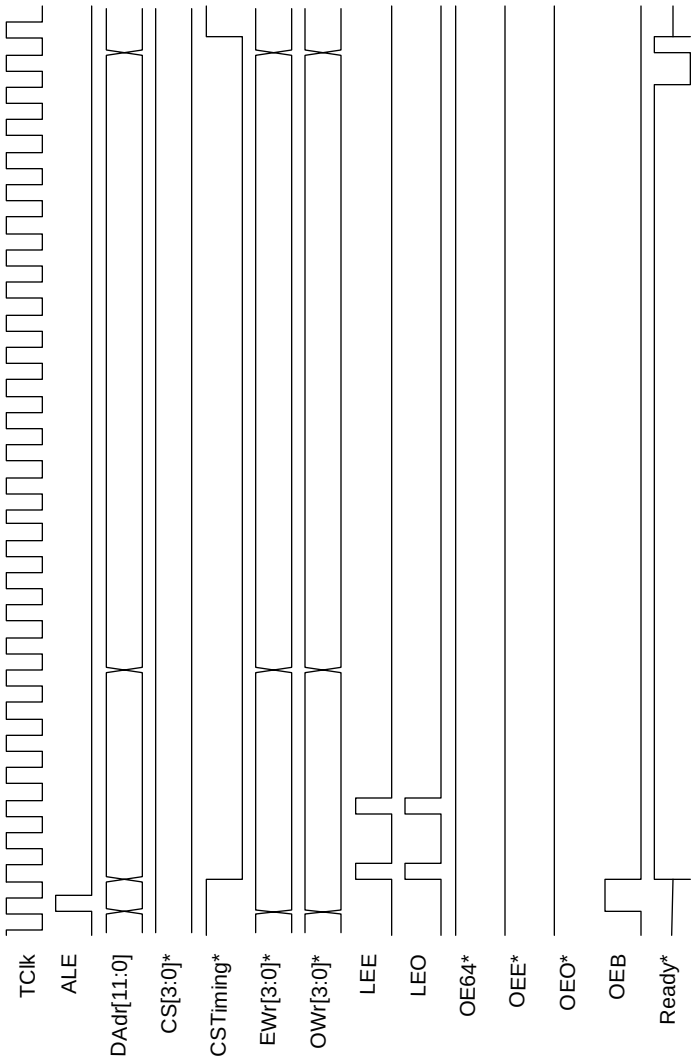
CPU 2 Bytes Write to PCI



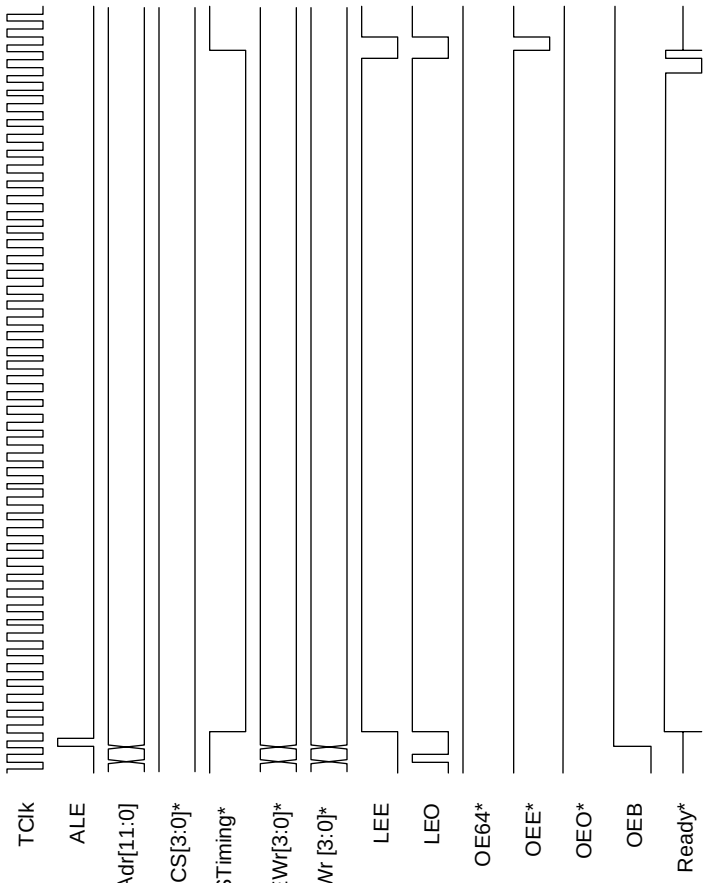
CPU Burst Read from PCI



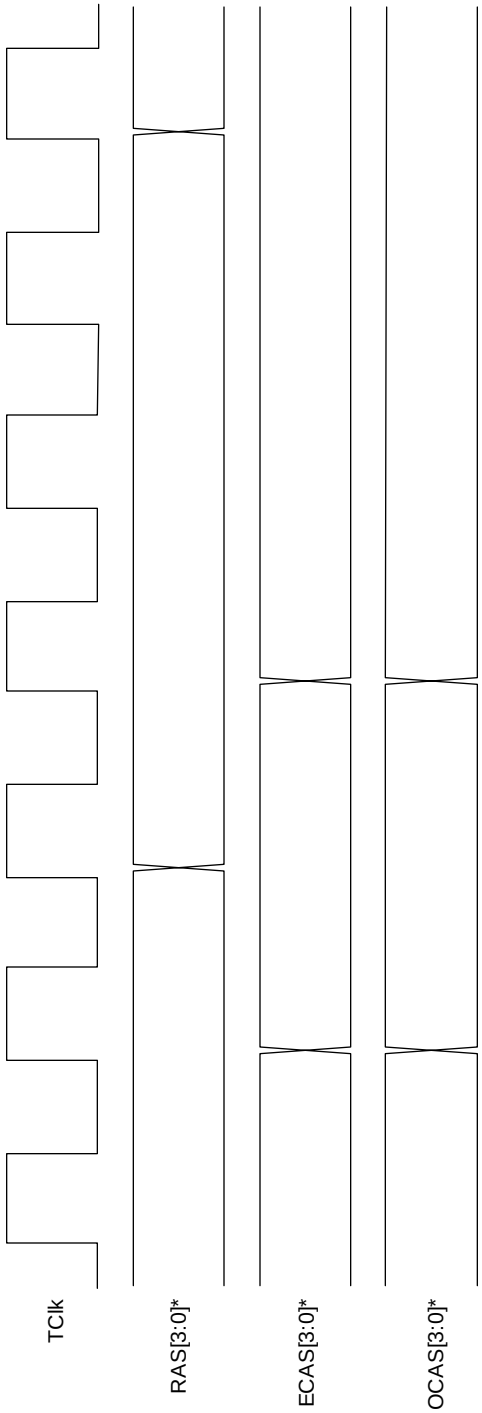
CPU Burst Write to PCI



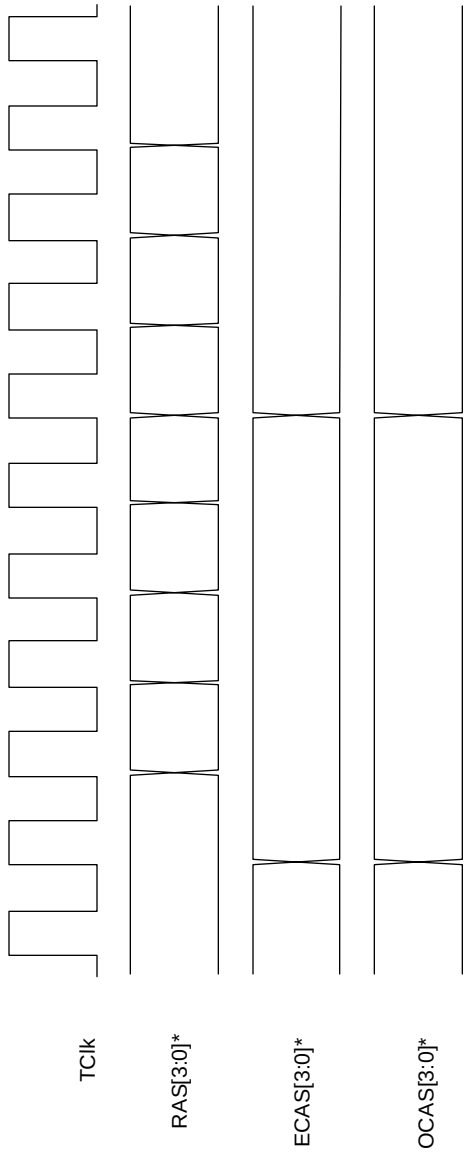
Write Operation Controlled by Ready*



Read Operation Controlled by Ready*



Non Staggered Refresh



Staggered Refresh

10. APPLICATIONS: CONNECTING THE MEMORY BUSES

In order to connect the memory (DRAM and Devices) correctly, it is necessary to properly choose the system configuration. The GT-64010A supports two main configuration modes: without data latches or with data latches.

10.1 Working Without Data Latches

Only systems that do not have 64-bits wide memories can work without latches. In this configuration the only external latch required is for Device Control and Address.

Connection	Memory Width	Connect...	To...
DRAM Address	32-bit	DAdr[11:0]	DRAM address pins
Device Address[1]	32-bit	{dev_adr[21:2],DAdr[2:0]}	Device address pins
	16-bit	{dev_adr[21:0],DAdr[2:1]}	Device address pins
	8-bit	{dev_adr[21:0],DAdr[2:0]}	Device address pins
DRAM Data[2,3]	32-bit	AD[31:0]	DRAM data pins
Device Data[2,3]	32-bit	AD[31:0]	Device data pins
	16-bit	AD[16:0]	Device data pins
	8-bit	AD[7:0]	Device data pins
Device Control	32-bit or less	AD[31:0]	Control latch inputs
		ALE	Control latch LE
		Control latch bit[0] output	Becomes BootCS*
		Control latch bit[1] output	Becomes DevRW*
		Control latch bit[23:2] outputs	Becomes dev_adr [21:0]
		Control latch bit[27:24] outputs	Becomes DMAAck[3:0]*
		Control latch bit[31:28] outputs	Becomes CS[3:0]*

Notes:

1. dev_adr[21:0] is the output of the control latch connected to AD[23:2], sampled by ALE.

2. Regardless of data endianess,

   ECAS[0]* and EWr[0]* always correspond to AD[7:0]

   ECAS[1]* and EWr[1]* always correspond to AD[15:8].

   ECAS[2]* and EWr[2]* always correspond to AD[23:16].

   ECAS[3]* and EWr[3]* always correspond to AD[31:24].

3. For load balancing, one can connect OWr[3:0]* and OCAS[3:0]* as well.

10.2 Working With Data Latches

10.2.1. 64-bit DRAM

Connection	Memory Width	Connect...	To...
DRAM Address[1]	64-bit	DAdr[11:0]	Even latch inputs
		DAdr[11:0]	Odd latch inputs
		Even latch outputs	Even DRAM address pins
		Odd latch outputs	Odd DRAM address pins
		LEAdrE	Even latch LE
		LEAdrO	Odd latch LE
DRAM Address[2]	64-bit	DAdr[11:0][4]	Even DRAM address pins
		DAdr[11:0]	Odd latch inputs
		Odd latch outputs	Odd DRAM address pins
		LEAdrO	Odd latch LE
DRAM Address[3]	64-bit	DAdr[11:0][4]	Odd DRAM address pins
		DAdr[11:0]	Even latch inputs
		Even latch outputs	Even DRAM address pins
		LEAdrO	Even latch LE
DRAM Data (Latched)[8]	64-bit	AD[31:0]	Even latch I/Os A side
		Even latch I/Os B side	Even bank data pins
		'0'	Even latch CLKAB and CLKBA
		LEE	Even latch LEAB and LEBA
		OEE*	Even latch OEBA*
		OEB[5]	Even latch OEAB
		AD[31:0]	Odd latch I/Os A side
		Odd latch I/Os B side	Odd bank data pins
		'0'	Odd latch CLKAB and CLKBA
		LEO	Odd latch LEAB and LEBA
		OEO*	Odd latch OEBA*
		OEB[5]	Odd latch OEAB
DRAM Data (Bypass)[8]	64-bit	SysAD[31:0][6]	Even bypass latch outputs
		SysAD[63:32][7]	Odd bypass latch outputs
		Even bypass latch inputs	Even bank data pins
		Odd bypass latch inputs	Odd bank data pins
		LEE	Even bypass latch LE
		LEO	Odd bypass latch LE
		OE64*	Even and odd bypass latches OE*

Notes:
1. System supports decrement.
2. System is little endian without decrement.
3. System is big endian without decrement.
4. Signals can be optionally buffered.

5. Be careful if OEB is programmed at reset to have reverse polarity.

6. SysADC[3:0] from the CPU too, if parity is supported.

7. SysADC[7:4] from the CPU too, if partiy is supported.

8. Regardless of endianess,

   ECAS[0]* always corresponds to SysAD[7:0] and AD[7:0].

   ECAS[1]* always corresponds to SysAD[15:8] and AD[15:8].

   ECAS[2]* always corresponds to SysAD[23:16] and AD[23:16].

   ECAS[3]* always corresponds to SysAD[31:24] and AD[31:24].

   OCAS[0]* always corresponds to SysAD[39:32] and AD[7:0].

   OCAS[1]* always corresponds to SysAD[47:40] and AD[15:8].

   OCAS[2]* always corresponds to SysAD[55:48] and AD[23:16].

   OCAS[3]* always corresponds to SysAD[63:56] and AD[31:24].

10.2.2. 32-bit DRAM

It is the same for odd and even bank connections. It is optional to have one bidirectional buffer.

Connection	Memory Width	Connect...	To...
DRAM Address[4]	32-bit	DAdr[11:0]	DRAM address pins
DRAM Data (Latched)[1,2]	32-bit	AD[31:0]	Even latch I/Os A side
		Even latch I/Os B side	Even bank data pins
		'0'	Even latch CLKAB and CLKBA
		LEE	Even latch LEAB and LEBA
		OEE*	Even latch OEBA*
		OEB[3]	Even latch OEAB
DRAM Data (No Latch)[1,2]	32-bit	AD[31:0]	Even bank data pins

Notes:

1. Example shows even bank choice, but odd bank can be selected instead.

2. Regardless of endianess,

   ECAS[0]* always corresponds to SysAD[7:0], SysAD[39:32] and AD[7:0].

   ECAS[1]* always corresponds to SysAD[15:8], SysAD[47:40] and AD[15:8].

   ECAS[2]* always corresponds to SysAD[23:16], SysAD[55:48] and AD[23:16].

   ECAS[3]* always corresponds to SysAD[31:24], SysAD[63:56] and AD[31:24].

3. Be careful if OEB is programmed at reset to have reverse polarity.

4. Signals can be optionally buffered for load balancing.

10.2.3. 64-bit Devices

Connection	Memory Width	Connect...	To...
Device Address[1]	64-bit	DAdr[2:0] DAdr[2:0] Even latch outputs Odd latch outputs LEAdrE LEAdrO {dev_adr[21:2], BAdrE[2:1]}[4] {dev_adr[21:2}, BAdrO[2:1]}[4]	Even latch inputs Odd latch inputs Become burst address even (BAdrE[2:0]) Become burst address odd (BAdrO[2:0]) Even latch LE Odd latch LE Even bank address pins Odd bank address pins
Device Address[2]	64-bit	DAdr[2:0] Odd latch outputs LEAdrO {dev_adr[21:2], DAdr[2:1]}[4] {dev_adr[21:2}, BAdrO[2:1]}[4]	Odd latch inputs Become burst address odd (BAdrO[2:0]) Odd latch LE Even bank address pins Odd bank address pins
Device Address[3]	64-bit	DAdr[2:0] Even latch outputs LEAdrO {dev_adr[21:2], BAdrE[2:1]}[4] {dev_adr[21:2}, DAdr[2:1]}[4]	Even latch inputs Become burst address even (BAdrE[2:0]) Even latch LE Even bank address pins Odd bank address pins
Device Data (Latched)[6]	64-bit	AD[31:0] Even latch I/Os B side '0' LEE OEE* OEB[5] AD[31:0] Odd latch I/Os B side '0' LEO OEO* OEB[5]	Even latch I/Os A side Even bank data pins Even latch CLKAB and CLKBA Even latch LEAB and LEBA Even latch OEBA* Even latch OEAB Odd latch I/Os A side Odd bank data pins Odd latch CLKAB and CLKBA Odd latch LEAB and LEBA Odd latch OEBA* Odd latch OEAB

Connection	Memory Width	Connect...	To...
Device Data (Bypass)[6]	64-bit	SysAD[31:0]	Even bypass latch outputs
		SysAD[63:32]	Odd bypass latch outputs
		Even bypass latch inputs	Even bank data pins
		Odd bypass latch inputs	Odd bank data pins
		LEE	Even bypass latch LE
		LEO	Odd bypass latch LE
		OE64*	Even and odd bypass latches OE*
Device Control	64-bit	AD[31:0]	Control latch inputs
		ALE	Control latch LE
		Control latch bit[0] output	Becomes BootCS*
		Control latch bit[1] output	Becomes DevRW*
		Control latch bit[23:2] outputs	Becomes dev_adr [21:0]
		Control latch bit[27:24] outputs	Becomes DMAAck[3:0]*
		Control latch bit[31:28] outputs	Becomes CS[3:0]*

Notes:

1. System supports decrement.
2. System is litttle endian without decrement.
3. System is big endian without decrement.
4. dev_adr[21:0] is the output of the latch connected to AD[23:2], sampled by ALE.
5. Be careful if OEB is programmed at reset to have reverse polarity.
6. Regardless of endianess,

   EWr[0]* always corresponds to SysAD[7:0], SysAD[39:32] and AD[7:0].

   EWr[1]* always corresponds to SysAD[15:8], SysAD[47:40] and AD[15:8].

   EWr[2]* always corresponds to SysAD[23:16], SysAD[55:48] and AD[23:16].

   EWr[3]* always corresponds to SysAD[31:24], SysAD[63:56] and AD[31:24].

   OWr[0]* always corresponds to SysAD[39:32] and AD[7:0].

   OWr[1]* always corresponds to SysAD[47:40] and AD[15:8].

   OWr[2]* always corresponds to SysAD[55:48] and AD[23:16].

   OWr[3]* always corresponds to SysAD[63:56] and AD[31:24].

10.2.4. 32-bit or Less Devices

It is optional to have one bidirectional buffer.

Connection	Memory Width	Connect...	To...
Device Address[1]	32-bit	{dev_adr[21:2], DAdr[2:0]}	Device address pins
	16-bit	{dev_adr[21:0], DAdr[2:1]}	Device address pins
	8-bit	{dev_adr[21:0], DAdr[2:0]}	Device address pins
Device Data (Latched)[2,3]	32-bit or less	AD[31:0] or [15:0] or [7:0]	Even latch I/Os A side
		Even latch I/Os B side	Even bank data pins
		'0'	Even latch CLKAB and CLKBA
		LEE	Even latch LEAB and LEBA
		OEE*	Even latch OEBA*
		OEB[3]	Even latch OEAB
Device Data (No Latch)	32-bit or less	AD[31:0] or [15:0] or [7:0]	Device data pins
Device Control	32-bit or less	AD[31:0]	Control latch inputs
		ALE	Control latch LE
		Control latch bit[0] output	Becomes BootCS*
		Control latch bit[1] output	Becomes DevRW*
		Control latch bit[23:2] outputs	Becomes dev_adr [21:0]
		Control latch bit[27:24] outputs	Becomes DMAAck[3:0]*
		Control latch bit[31:28] outputs	Becomes CS[3:0]*

Notes:

1. dev_adr[21:0] is the output of the latch connected to AD[23:2], sampled by ALE

2. Bidirectional latch is optional for buffering, and either odd or even bank can be chosen. This example shows an even bank connection.

3. Regardless of endianess,

   EWr[0]* always corresponds to SysAD[7:0] and AD[7:0] regardless of endianess.

   EWr[1]* always corresponds to SysAD[15:8] and AD[15:8] regardless of endianess.

   EWr[2]* always corresponds to SysAD[23:16] and AD[23:16] regardless of endianess.

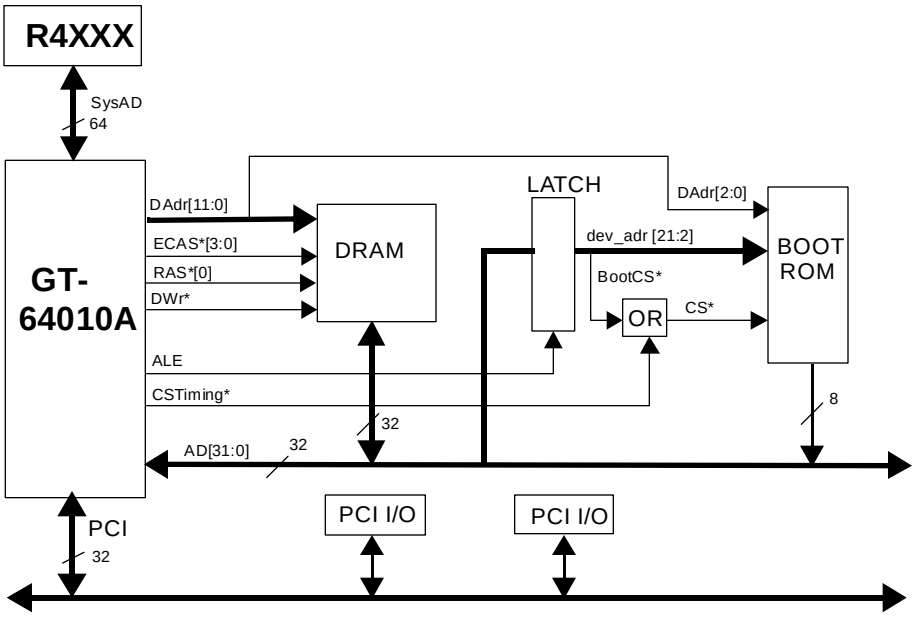
   EWr[3]* always corresponds to SysAD[31:24] and AD[31:24] regardless of endianess.

4. Be careful if OEB is programmed at reset to have reverse polarity.

11. APPLICATIONS: SYSTEM CONFIGURATIONS

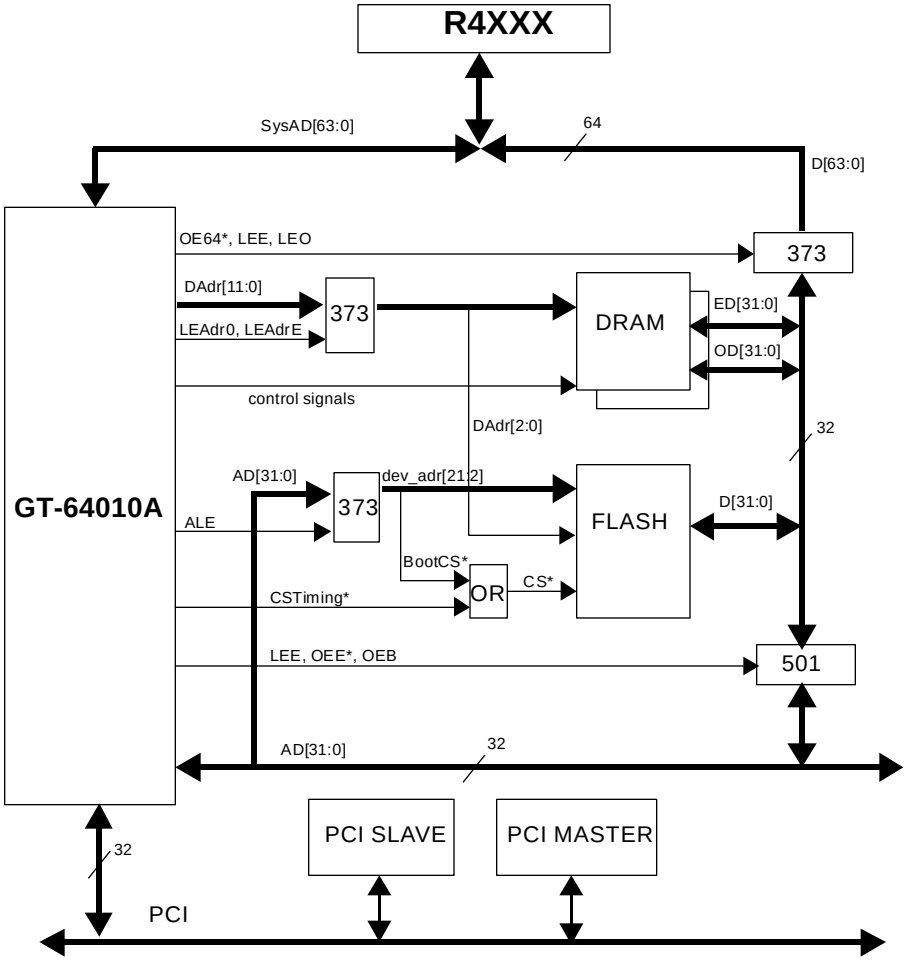
11.1 Minimal System Configuration

A minimal system configuration is shown below. It includes an 8-bit wide boot ROM, a 32-bit wide DRAM, and a PCI interface to industry standard I/O devices. This configuration can be appealing to applications that need high performance and are limited to a minimal board space.



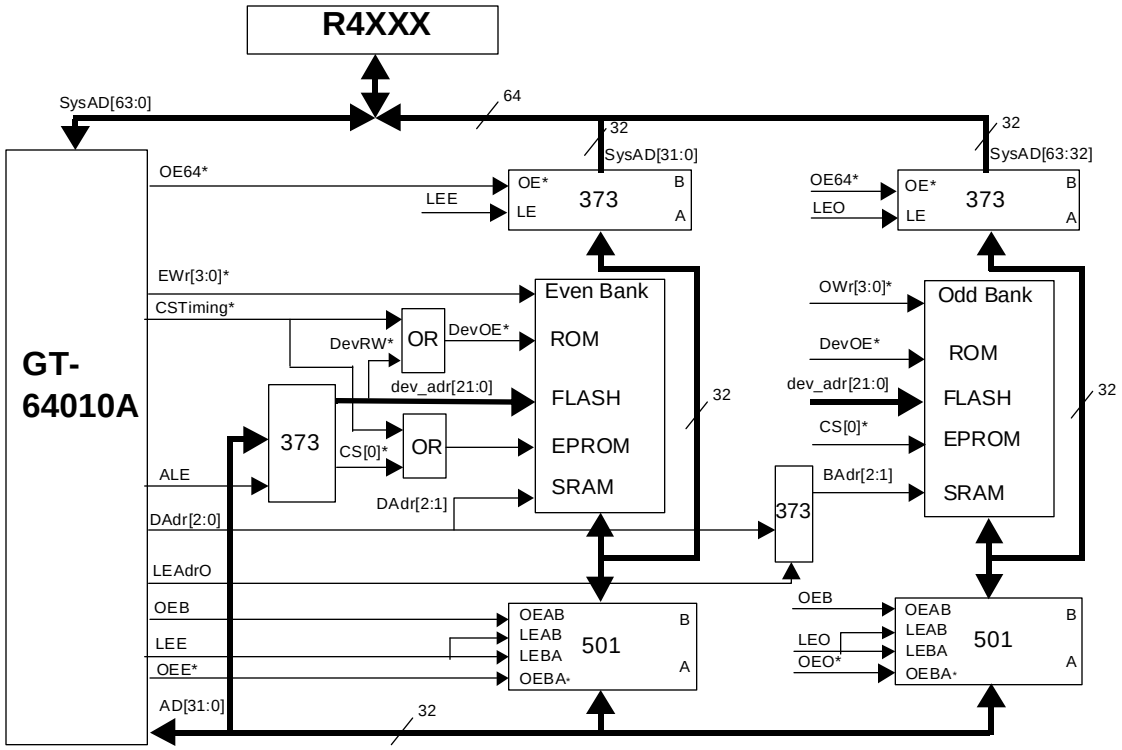
11.2 Typical System

The high performance system shown below includes 64-bit wide memories and 32-bit I/O devices on the PCI bus. The system includes Flash for storing code and data, and DRAM as main memory. In this system data can be moved via DMA or PCI master accesses between the PCI bus and the Flash or DRAM at full bus bandwidth, at 200 Mbytes per second. The CPU can read from the DRAM at peak bandwidth of 200Mbytes per second. CPU writes have peak bandwidth of 400Mbytes per second for all devices through the GT-64010A on-chip write buffer.



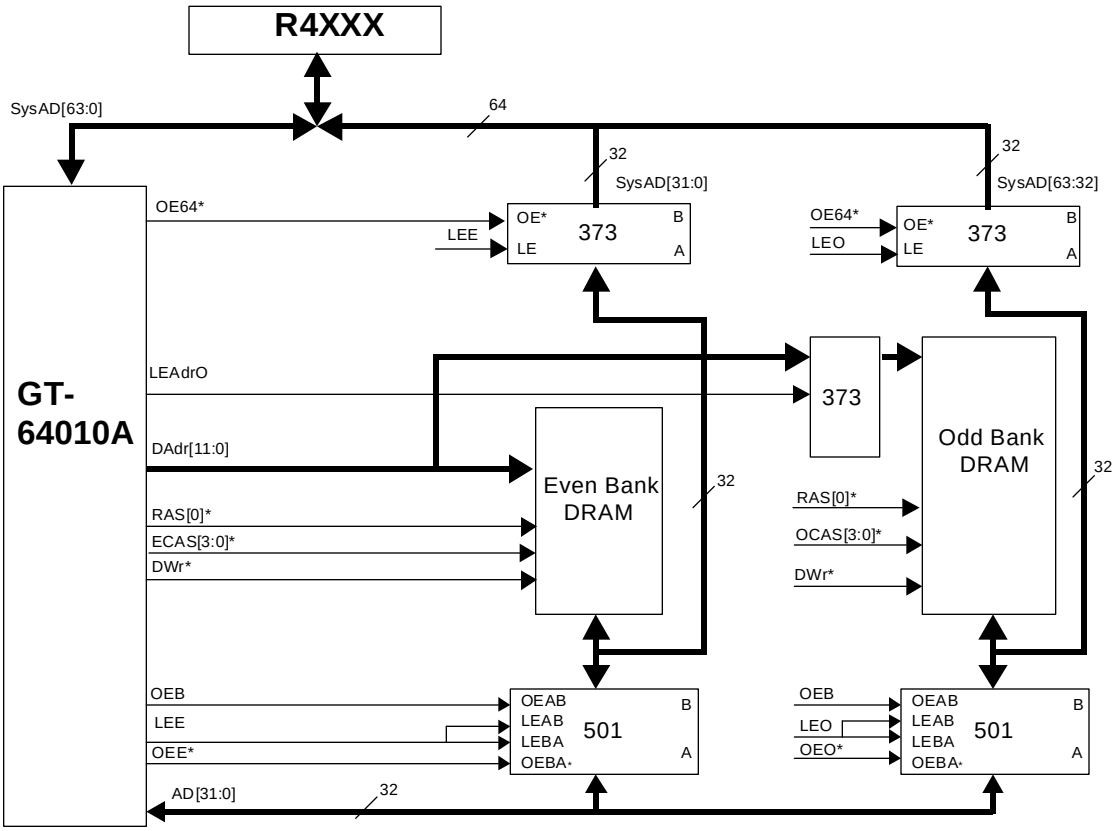
11.3 Interface to Asynchronous Devices

In this case, we show the connectivity between the GT-64010A and 64 bit-wide standard memory devices (e.g. SRAM). In this example the latches selected are industry standard FCT16501for data, FCT16373 for the address, and FCT16373 for the burst address for the odd bank. The data latches that interface to the AD bus are used to interleave the data in read and write access from the GT-64010A and enable data rate of one data per cycle at 50MHz (200 Mbytes per second peak rate). The data latches that interface the SysAD bus are used only in one direction in CPU reads. The SysAD latches enable low latency to first word and up to 400 Mbytes per second bandwidth in CPU burst reads. FCT16373 logic chips are used to latch the address, the CS* and the DevRW*for the Devices. The latch for the odd bank burst address is needed for the address interleaving. This system configuration is for little endian, no decrement.



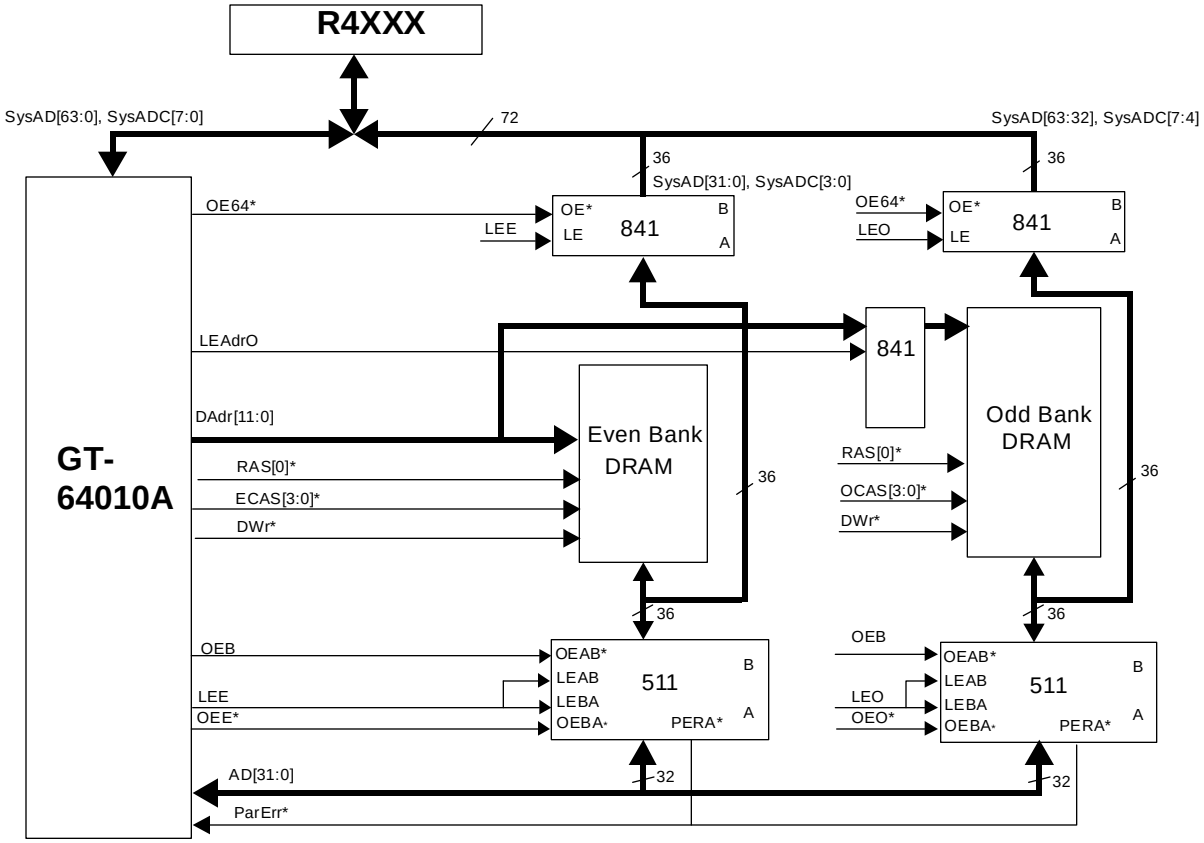
11.4 Interface to DRAM

The DRAM and the device interface in a typical system share the same data latches and address latches. In this example the DRAM is 64 -bit wide and the even and odd banks share the same RAS[0]* pin. Address for the odd bank is driven from an FCT16373 latch which is controlled by LEAdrO. The data latches used in this example are FCT16373 for the interface to the SysAD and FCT16501 to the AD bus. This system configuration is for little endian, no decrement.

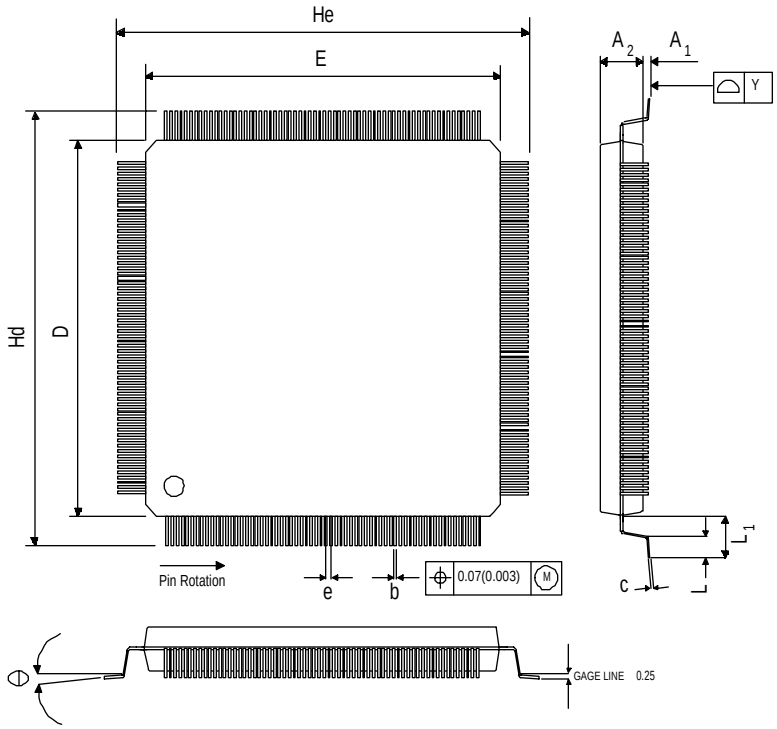


11.5 A System With Parity

In this case, the FCT16511 is used to generate parity for all the write accesses to the devices and the DRAM. In a CPU read access the FCT16511 will check parity and will assert the ParErr* signal when an error is detected. The GT-64010A will indicate to the CPU that the returned data is erroneous via SysCmd[5] and will interrupt the CPU if the bank that the CPU read from was programmed to have parity integrity checks. The GT-64010A also asserts SysCmd[4] to indicate to the CPU if the read access was from an address with parity integrity so that the CPU will check parity internally as well. In PCI read accesses, an active ParErr* to a bank that has parity integrity, will cause an assertion of PErr* on the PCI. Notice that with the FCT16511 the OEAB* polarity is active low, as opposed to the FCT16501, and thus OEB from the GT-64010A should be programmed accordingly at reset. This system configuration is for little endian, no decrement.



12. PACKAGING

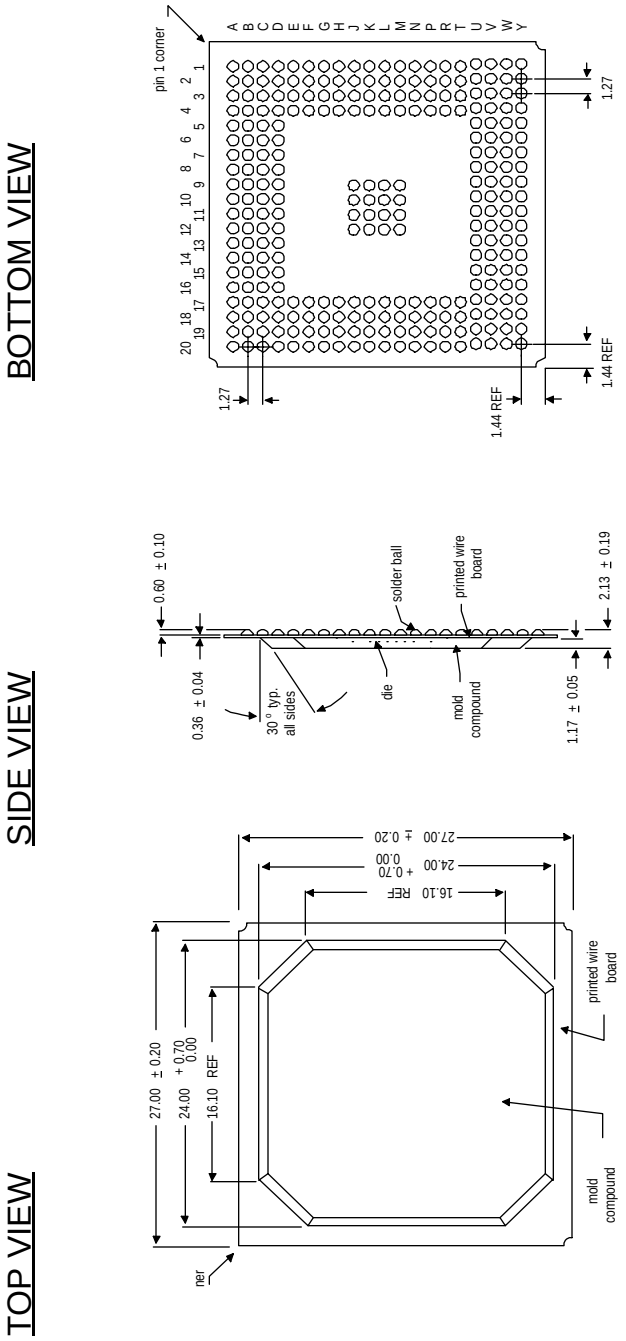


256 LEAD PQFP PACKAGE OUTLINE

SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A_1	0.05	0.25	0.50
A_2	3.17	3.32	3.47
b	0.13	0.18	0.23
c	0.09		0.20
D	27.90	28.00	28.10
E	27.90	28.00	28.10
e		0.40	
Hd	30.35	30.60	30.85
He	30.35	30.60	30.85
L	0.45	0.60	0.75
L_1		1.30	
Y			0.08
Q	0		10

272 BGA PACKAGE OUTLINE DRAWING

ALL DIMENSIONS IN MILLIMETERS



13. Appendices

13.1 GT-64010A Behavior With Little/big Endian Data Formats

13.1.1. Background

There are two bits in the GT-64010A which control byte swapping. One is located in the CPU Interface unit's mode register (0x000) bit 12, the other in PCI Interface unit's command register (0xc00) bit 0. Both bits are given the same value as sampled at reset via pullup/pulldown on Interrupt* pin. Both can be otherwise programmed after reset is deasserted. As a master rule, if both bits are set to '1', the GT-64010A assumes Little-endian data format and NO byte swapping is done within the device.

13.1.2. Nomenclature

W - Word, 32 bit data. (R4600 terminology)

DW - Double-Word, 64 bit data. (R4600 terminology)

Even address - address of which A[2] == 0. In little-endian format this address points to the LEAST significant W of a DW, in Big-endian format this address points to the MOST significant W of a DW.

Odd address - address of which A[2] == 1. In little-endian format this address points to the MOST significant W of a DW, in Big-endian format this address points to the LEAST significant W of a DW.

Even word - LEAST significant W of a DW.

Odd word - MOST significant W of a DW

a) Bit 12 of the CPU Interface unit's mode register (0x000) affects the following:

a1) Set to '1' (Little-endian mode)

- No byte swapping within the CPU Interface unit on any data transfer.

a2) Set to '0' (Big-endian mode)

- Byte swapping of data transfers to/from GT-64010A internal registers (including Configuration Data register, 0xcfc).
- No byte swapping of data transfers of which the source/target is external.

b) Bit 0 of the PCI Interface unit's command register (0xc00) affects the following:

b1) Set to '1' (No byte swapping)

- No byte swapping within the PCI Interface unit of any data transfer.

b2) Set to '0' (Byte swapping)

- No byte swapping of data transfers to/from PCI Interface unit's internal registers.
- Byte swapping of data transfers of which the source/target is external

c) Here is a table which describes all combinations of the resources and swapping bits with a sample data. ('CPU bit' means the CPU Interface unit's mode register (0x000) bit 12, 'PCI bit' means the PCI Interface unit's command register (0xc00) bit 0). The sample data is 04030201h.

	SWAP Bits (CPU bit:PCI bit)			
Resource	11	00	01	10
Internal Registers (CPU access)	04030201	01020304	01020304	04030201
Internal Registers (PCI access)	04030201	04030201	04030201	04030201
Internal PCI Configuration Registers (CPU access)	04030201	01020304	01020304	04030201
Internal PCI Configuration Registers (PCI access)	04030201	04030201	04030201	04030201
External PCI Configuration Registers	04030201	04030201	01020304	01020304
Memory (DRAM and Devices) (CPU access)	04030201	04030201	04030201	04030201
Memory (DRAM and Devices) (PCI access)	04030201	01020304	04030201	01020304
CPU to PCI (Except external PCI Configuration Registers)	04030201	01020304	04030201	01020304

13.2 Address Decoding

13.2.1. CPU Interface Unit's Address Decode

An address generated by the CPU is decoded within the CPU Interface unit as follows:

- address bits [35:28] are checked for **equality** to CPU Interface unit's Low Decode Registers' bits [14:7]
- address bits [27:21] are checked to be **greater** or **equal** to CPU Interface unit's Low Decode Registers' bits [6:0]
- address bits [27:21] are checked to be **less** or **equal** to CPU Interface unit's High Decode registers' bits [6:0]

An address is said to be a **hit** when a match occurs in **all** three criterias for the **same** Low and High decode register pair.

13.2.2. DMA Address Decode

The DMA uses three types of address: **Source**, **Destination** and **Pointer**. (Pointer address is used only for chained mode) All three are decoded the same way as follows:

- address bits [31:28] are checked for **equality** to CPU Interface unit's Low Decode Registers' bits [10:7]
- address bits [27:21] are checked to be **greater** or **equal** to CPU Interface unit's Low Decode Registers' bits [6:0]
- address bits [27:21] are checked to be **less** or **equal** to CPU Interface unit's High Decode registers' bits [6:0]

An address is said to be a **hit** when a match occurs in **all** three criterias for the **same** Low and High decode register pair.

13.2.3. PCI Interface Unit's Address Decode

An address generated by a PCI Master is decoded within the PCI Interface Unit's Slave as follows:

- address bits for which the corresponding bits in the Size Registers are '**0**' are checked for **equality** to the **corre-**

sponding bits of the Base Registers.

An address is said to be a **hit** when a match occurs with a certain Base Register in all bits pointed by its respective Size Register.

Example:

RAS[3:2] Bank Size Register = 03fffh ---> address bits to compare are: AD[31:26] ('1's are don't care)

Therefore, an address of which bits [31:26] **equal** RAS[3:2] Base Address Register's bits [31:26] is considered a hit and will access RAS[3:2] space.

13.2.4. DRAM/Device Interface Unit's Address Decode

The DRAM/Device Interface Unit receives an address from either the CPU Interface Unit, PCI Interface Unit or DMA. This address has already been decoded primarily in the originating unit to be targeted either to RAS[1:0], RAS[3:2], CS[2:0], CS[3] & BootCS or Internal spaces. The DRAM/Device Interface Unit now further decodes the incoming address as follows:

- address bits [27:20] are checked to be **greater** or **equal** to DRAM/Device Interface unit's Low Decode Registers' bits [7:0]
- address bits [27:20] are checked to be **less** or **equal** to DRAM/Device Interface unit's High Decode registers' bits [7:0]

An address is said to be a **hit** when a match occurs in **both** criterias for the **same** Low and High decode register pair.

13.3 JTAG - Signal Ordering

The following list describes the ordering of the GT-64010A signals in the boundary scan chain:

NOTE: Oe* suffix to outputs/bidirectionals denotes the respective active low Output-enable control.

Signal Name	Chain Position	Signal Name	Chain Position
Req*	0	DAdr[0]/BAdr[0]Oe*	8
ReqOe*	1	DAdr[1]/BAdr[1]	9
Gnt*	2	DAdr[2]/BAdr[2]	10
PClk	3	DAdr[2]/BAdr[2]Oe*	11
Rst*	4	DAdr[3]/EWr[0]*	12
Int*	5	DAdr[3]/EWr[0]Oe*	13
IntOe*	6	DAdr[6:4]/EWr[3:1]*	16:14
DAdr[0]/BAdr[0]	7	DAdr[6:4]/EWr[3:1]Oe*	17
DAdr[10:7]/OWr[3:0]*	21:18	OEE*	88
DAdr[10:7]/OWr[3:0]Oe*	22	OEEOe*	89
DAdr[11]/ADS*	23	OEB	90
DAdr[11]/ADSOe*	24	OEBOe*	91
RAS[3:0]*	28:25	LEAdrO	92
RAS[3:0]Oe*	29	LEAdrOOe*	93
OCAS[3:0]*	33:30	LEAdrE/DMAReq[2]*	94
OCAS[3:0]Oe*	34	LEAdrE/DMAReq[2]Oe*	95
ECAS[3:0]*	38:35	DMAReq[1]*/ParErr*	96
ECAS[3:0]Oe*	39	DMAReq[0]*/Ready*	97
DWr*	40	SysAD[56:63]	105:98
DWrOe*	41	SysAD[56:63]Oe*	106
AD[0]/BootCS*	42	SysAD[48:55]	114:107
AD[1]/DevRW*	43	SysAD[48:55]Oe*	115
AD[7:2]	49:44	SysAD[40:47]	123:116
AD[0]/BootCSOe*, AD[1]/DevRWOe*, AD[7:2]Oe*	50	SysAD[40:47]Oe*	124
AD[15:8]	58:51	SysAD[32:39]	132:125
AD[15:8]Oe*	59	SysAD[32:39]Oe*	133
AD[23:16]	67:60	Release*	134
AD[23:16]Oe*	68	WrRdy*	135
AD[27:24]/DMAAck[3:0]*	72:69	WrRdyOe*	136
AD[31:28]/CS[3:0]*	76:73	Validin*	137
AD[27:24]/DMAAck[3:0]Oe*, AD[31:28]/CS[3:0]Oe*	77	ValidinOe*	138
CSTiming*	78	ValidOut*	139

Signal Name	Chain Position	Signal Name	Chain Position
CSTimingOe*	79	SysAD[24:31]	147:140
ALE	80	SysAD[24:31]Oe*	148
ALEOe*	81	SysAD[16:23]	156:149
LEO	82	SysAD[16:23]Oe*	157
LEOOe*	83	TClk	158
LEE	84	SysAD[8:15]	166:159
LEEOe*	85	SysAD[8:15]Oe*	167
OEO*	86	SysAD[0:7]	175:168
OEOOe* (spelling excercise...)	87	SysAD[0:7]Oe*	176
SysCmd[3:0]	180:177	PErrOe*	220
SysCmd[3:0]Oe*	181	Lock*	221
SysCmd[8:4]	186:182	Stop*	222
SysCmd[8:4]Oe*	187	StopOe*	223
Interrupt*	188	DevSel*	224
InterruptOe*	189	DevSelOe*	225
Hit/DMAReq[3]*	190	TRdy*	226
OE64*	191	TRdyOe*	227
OE64Oe*	192	IRdy*	228
PAD[3:0]	196:193	IRdyOe*	229
PAD[3:0]Oe*	197	Frame*	230
PAD[7:4]	201:198	FrameOe*	231
PAD[7:4]Oe*	202	CBE[2]*	232
CBE[0]*	203	PAD[19:16]	236:233
PAD[11:8]	207:204	PAD[19:16]Oe*	237
PAD[11:8]Oe*	208	PAD[23:20]	241:238
PAD[15:12]	212:209	PAD[23:20]Oe*	242
PAD[15:12]Oe*	213	IdSel	243
CBE[1]*	214	CBE[3]*	244
Par	215	CBE[3:0]Oe*	245
ParOe*	216	PAD[27:24]	249:246
SErr*	217	PAD[27:24]Oe*	250
SErrOe*	218	PAD[31:28]	254:251
PErr*	219	PAD[31:28]Oe*	255

Notes:

1. SysAD ordering in the boundary scan chain is reversed.i.e., for ascending chain order, SysAD lane significance is in descending order.

2. The TRIBYP OpCode is not implemented. If tri-stating all GT-64010A outputs and bidirectionals is needed, use the EXTEST OpCode to disable all Oe* controls in the chain.

13.4 Connecting the GT-64010A without a CPU

When the GT-64010A is used in a system without a CPU, the CPU Interface unit's inputs should be pulled up, i.e., SysAD, SysCmd, ValidOut*, Release* and Hit.

13.5 Disabling JTAG

If the JTAG is not to be used, the following should be pulled-up/down:

JTRST* - pulldown

JTDI - pullup

JTMS - pulldown

JTCLK - pulldown

13.6 DRAM Address Partitioning

This appendix includes an explenation on how the DRAM address is formed in respect to the XKRefresh that the GT64010A is programmed to. Following is a table showing how the address is split into Row and Column addresses during Ras phase and Cas phase.

KRefresh	# of Row bits	Row bits	Column bits			
1/2	9	5...13	22...20	16...14	19...17	4...2
1	10	5...14	23...21	16...15	20...17	4...2
2	11	5...15	24...22	16	21...17	4...2
4	12	5...16	25...17			4...2

Notes: The table refers to 32 bit DRAM configuration. For 64 bit configuration, '4...2' is changed to '4...3'.

Example:

For 1KRefresh the DAdr bus will look as follows:

During Row phase, DAdr[9:0] will hold SysAD[14:5] (or PAD[14:5]).

During Column phase, DAdr[11:0] = {SysAD[23:21],SysAD[16:15],SysAD[20:17],SysAD[4:2]}.(or PAD...)

14 Revision History

Table 1: Revision History

Document Type	Revision Number	Date	Comments
PRELIMINARY REV.	1.0	7/96	First revision of PRELIMINARY REVISION for general distribution.
PRELIMINARY REV.	1.1	12/96	**Section 5.12**: Changed 4 24bit timers to 3 24bit and 1 32bit timer. **Section 8.3**: Changed Operating current from 280mA to 400mA. **Section 8.4**: Added Thermal Package Characteristics for 256PQFP. **Section 8.5**: Added Thermal Package Characteristics for 272 BGA **Section 14**: Updated revision history.