PowerPC 403 to PCI 9060ES Application Note

PCI 9060/403 AN

May 10, 1996
Version 0.4

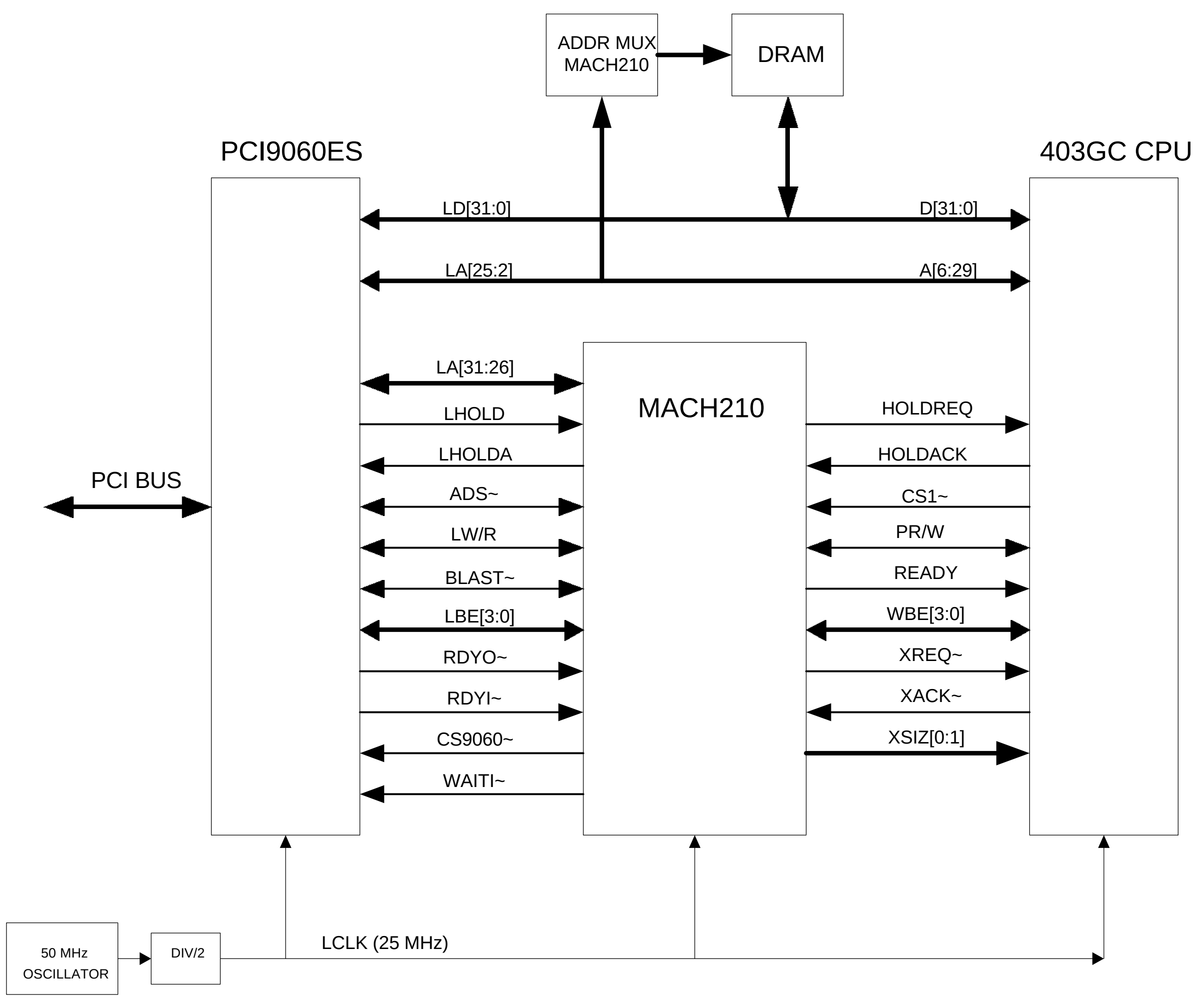
PowerPC 403 to PCIbus Application Note

Features

- Embedded system containing PowerPC 403 with a PCIbus interface
- PCI 9060ES chip supports master, slave and PCI configuration cycles
- 403local bus runs asynchronously to the PCI clock.
- FIFOs in PCI 9060ES support continuous burst transfers between 403 local bus and PCIbus

General Description

This application note describes how to interface the PowerPC 403GC CPU to the PCI bus using the PLX PCI9060ES PCI to Local Bus Bridge chip. The PCI9060ES has both direct master and direct slave transfer capabilities. The direct master mode allows a device (403GC) on the local bus to perform memory, I/O and configuration cycles to the PCI bus. The direct slave mode allows a master device on the PCI bus to access memory (DRAM) on the local bus. **The 403GA may also be used with little or no modification to the design.**



PLX Technology, Inc, 690 Potrero Avenue, Sunnyvale, CA 94086, Phone 408-774-9060, Fax 408-774-2169

Table of Contents

1. Introduction

This application note describes how to interface the PowerPC 403GC CPU to the PCI bus using the PLX PCI9060ES Local Bus Bridge chip. **The 403GA may be used in place of the GC with little or no modification to the design.** (Contact PLX for more information). The PCI9060ES has both direct master and direct slave transfer capabilities. The direct master mode allows a device (403GC) on the local bus to perform memory, I/O and configuration cycles to the PCI bus. The direct slave mode allows a master device on the PCI bus to access memory (DRAM) on the local bus. The PCI9060ES allows the local bus to operate asynchronously to the PCI bus through the use of bi-directional FIFOs. In this application the PCI bus operates at 33 MHz while the local bus is clocked at 25 MHz. The following block diagram shows the basic connections between the major components required for this application.

IMPORTANT NOTE ABOUT REVISION 0.3: The design in Revision 0.3 (this revision) of this application note allows 16 word bursting. In Revision 0.1 the design allowed only four word bursting. To achieve this improvement, the design was changed from using two Mach 210's (Rev 0.1 design) to one Mach 210 and three 16V8's. The PAL equations and schematics included with this application note reflect the changes. However, the text and timing diagrams still reflect the 0.1 (4 word burst) operation. Although the differences are minor, this is noted to avoid confusion. PLX is now updating the text and timing diagrams to reflect the change from four word burst to 16 word burst.

Note that the address and data buses on the 403GC designate bit 0 as the **most** significant bit. Also, the 403GC does not produce the upper 6 address bits, so its maximum addressing range for one bank is 64 Mbytes.

NOTE: PLX provides a "Technical Update", available from the FTP site on the Web page that contains design notes, spec updates and errata on the PLX chips. All designers should obtain this information when performing a design. The web page is at www.plxtech.com. Contact PLX by e-mail, phone or fax for the FTP passwords.

2. Direct Master Mode

2.1 Overview

In the direct master mode, the 403GC can access the PCI bus using memory, I/O or configuration cycles. One of the 8 available memory banks provided by the 403GC is used to access the PCI bus through the PCI9060ES.

There are also a number of local configuration registers in the PCI9060ES which must be programmed by the 403GC before accesses can be made to the PCI bus. These configuration registers define base addresses, address ranges, and local bus characteristics.

2.2 403GC Bank Register

One of the eight 403GC bank registers is used to access the PCI9060ES. The size of this bank register should be programmed to 64 Mbytes to provide the largest possible window into the PCI address space. The 64 Mbytes is then subdivided into three spaces:

- Internal PCI9060ES Local Configuration Registers
- PCI I/O or Configuration Cycles
- PCI Memory Cycles

In this application, the first 16 Mbytes is allocated to configuration registers, the second 16 Mbytes to I/O and configuration cycles, and the last 32 Mbytes to PCI memory cycles. Other configurations are possible if more 403GC bank registers are used, or if the resolution of the address space decoder is increased. Following is a suggested configuration for the bank register used to access the PCI9060ES:

Configuration Bit	Suggested Value	Description
31 (SD) SRAM	1	SRAM style interface
30:28 (TH)	000	Transfer hold count
27 (WFB)	0	Write byte enable off timing
26 (WBN)	0	Write byte enable on timing
25 (OEN)	0	Output enable on timing
24 (CSN)	0	Chip select on timing
23:18 (TWT)	00000	Transfer wait
17 (RE)	1	Ready Enable
16:15 (BW)	10	Bus Width (32 bits)
14 (BME)	0	Burst mode enable (disabled)
13 (SLF)	1	Sequential Line Fills
12:11 (BU)	11	Bank Usage (R/W)
10:8 (BS)	110	Bank Size (64 Mbytes)
7:0 (BAS)	user defined	Bank Address Select

2.3 Bursting

Since the 403GC only performs burst cycles when accessing cached memory, this bank will not be configured for bursting. Also, the 403GC gives no indication during a cycle that it is performing a burst. Therefore, BLAST (end of burst) will always be asserted into the PCI9060ES during direct master or internal register cycles. The PCI9060ES does support bursting to the PCI bus if the master of the local bus is capable of bursting to the local side of the PCI9060. Applications which have a DMA controller on the local bus could utilize the direct master bursting feature of the PCI9060ES.

2.4 Read Cycles

A direct master or configuration read cycle is initiated by the 403GC when it asserts the chip select assigned to the PCI9060 and PCI bus. It also asserts an address (A6:29), read/write status (PR/W), and byte enables (WBE[0:3]). The PCI state machine in the PCI MACH device detects this cycle and transitions to state P4 where the address is strobed into the PCI9060ES using ADS. The WBE signals are mapped into the LBE inputs to the PCI9060ES, and the PR/W signal is inverted to become LW/R. Since the 403GC only produces 26 address bits, the upper six address bits to the PCI9060ES are forced to zero.

The READY input to the 403GC is negated, causing it to insert wait states. The PCI9060ES has a WAITI~ input pin which allows a master to control the duration of the read data presented by the PCI9060ES. At the beginning of a read cycle, this input is asserted. The PCI9060ES then runs the requested PCI or internal register cycle and asserts RDYO~ when the data is available. The state machine jumps to state P6 where READY is asserted to the 403GC. Once READY has been detected, the data will be sampled by the 403GC at the end of the next clock period. The state machine jumps to state P7 where the WAITI~ signal is negated, allowing the PCI9060ES to complete the read cycle. The 403GC reads the data at the end of this cycle. The state machine returns to PIDLE and waits for another chip select from the 403GC.

2.5 Write Cycles

A direct master or configuration write cycle is initiated by the 403GC when it asserts the chip select assigned to the PCI9060 and PCI bus. As with read cycles, it also asserts an address (A6:29), read/write status (PR/W), and byte enables (WBE[0:3]). The PCI state machine in the PCI MACH device detects this cycle and transitions to state P0 where the address is strobed into the PCI9060ES using ADS. The byte enables, read/write status, and address are mapped in the same way as read cycles.

The READY input to the 403GC is negated, causing it to insert wait states. The PCI9060ES then runs the requested PCI or internal register cycle and asserts RDYO~ when the write has been completed. The state machine jumps to state P2 where READY is asserted to the 403GC. Once READY has been detected, the 403GC will complete the write cycle at the end of the next clock period. The state machine jumps to state P3 during the last clock cycle of the transfer. The state machine then returns to PIDLE and waits for another chip select from the 403GC.

2.6 PCI9060ES Configuration Registers

The following local configuration registers must be programmed before direct master accesses to the PCI bus can occur:

Register	Offset
Range for Direct Master to PCI	9Ch
Local Base Address for Direct Master to PCI Memory	A0h
Local Base Address for Direct Master to PCI IO/CFG	A4h
PCI Base Address (Re-Map) for Direct Master to PCI	A8h
PCI Configuration Address Register for Direct Master to PCI IO/CFG	ACh

These registers are accessed by running a 403GC cycle to the PCI9060ES with the 9060 chip select (CS9060~) asserted. The address offsets for each register are shown in the table.

3. Direct Slave Mode

3.1 Overview

In the direct slave mode, a master device on the PCI bus can access the DRAM which is connected to the 403GC bus. The direct slave FIFOS in the PCI9060ES allows 3-2-2-2 bursting to and from the fast page mode DRAM.

3.2 Arbitration

When the PCI9060ES has decoded and accepted a PCI cycle which is to be passed through to the local bus, the LHOLD output is asserted. This signal is passed on through the PCI MACH to the 403GC HOLDREQ input. When the 403GC is ready to release the local bus, it asserts HOLDACK. This signal is connected directly to the LHOLDA input of the PCI9060ES. The PCI9060ES now has control of the local bus, and can begin its cycle. When the PCI9060ES is finished, it negates LHOLD, thus giving the bus back to the 403GC. During burst reads, the 403GC doesn’t finish the DRAM cycle until after the PCI9060ES is done, so the HOLDREQ signal is held for two extra clock cycles.

3.3 Read Cycles

A direct master read cycle is initiated by the PCI9060ES when it asserts address strobe (ADS~). It also asserts an address (A[31:2]), write/read status (LW/R), byte enables (LBE[3:0]), and burst last (BLAST~). The DRAMCTL state machine in the PCI MACH device detects this cycle and transitions to state S7 where a DRAM read cycle is initiated.

The XREQ~ and XSIZ[0:1] inputs to the 403GC are used to initiate a DRAM cycle. The XSIZ inputs are decoded as follows:

XSIZ[0:1]	Operation

00	Byte Transfer (8 bits)
01	Halfword Transfer (16 bits)
10	Fullword Transfer (32 bits)
11	Burst Fullword Transfer

The byte address is determined by the WBE2(A30) and WBE3(A31) inputs to the 403GC, and are derived from the LBE outputs of the PCI9060ES. During read cycles, all transfers are converted to full word transfers by the PCI9060ES. After the state machine asserts XREQ, it checks BLAST~ to determine if this is a single or burst transfer. If it is a single transfer, it waits in state S11 for the 403GC to assert XACK~, indicating that the read data is available. The RDYI~ input to the PCI9060ES is asserted, causing the read data to be loaded into the direct slave read FIFO. For a burst cycle, the state machine waits in state S8, where XREQ~ is continuously activated. When BLAST~ is asserted, the PCI9060ES is ready to finish the read burst. The 403GC always reads one extra word after XREQ~ is negated, but in this application, the extra data is simply ignored.

In this application the PCI9060ES is programmed to burst a maximum of 4 fullwords for every address strobe. Burst transfers do not cross 16 byte boundaries, and are sequential. Therefore the column address counter in the DRAMMUX MACH only needs to be two bits wide. For longer burst lengths, the size of the column address burst counter must be increased, and a carry output is needed to stop the PCI9060ES from bursting when the counter is about to roll over. The BTERM input to the PCI9060ES is used to perform this function.

3.4 Write Cycles

A direct master write cycle is initiated by the PCI9060ES when it asserts address strobe (ADS~). It also asserts an address (A[31:2]), write/read status (LW/R), byte enables (LBE[3:0]), and burst last (BLAST~). The DRAMCTL state machine in the PCI MACH device detects this cycle and transitions to state S1 where a DRAM write cycle is initiated. If an unaligned write cycle is detected, then the state machine will go to state S5. More about unaligned transfers later.

The XREQ and XSIZ[0:1] inputs to the 403GC are used to initiate a DRAM cycle. The XSIZ inputs are decoded as follows:

XSIZ[0:1]	Operation
00	Byte Transfer (8 bits)
01	Halfword Transfer (16 bits)
10	Fullword Transfer (32 bits)
11	Burst Fullword Transfer

The byte address is determined by the WBE2(A30) and WBE3(A31) inputs to the 403GC, and is derived from the LBE outputs of the PCI9060ES. After the state machine asserts XREQ~, it checks BLAST~ to determine if this is a single or burst transfer. If it is a single transfer, it waits in state S4 for the 403GC to assert XACK~, indicating that the data has been written. The RDYI~ input to the PCI9060ES is asserted, causing the write cycle to be completed. For a burst cycle, the state machine waits in state S2, where

XREQ~ is continuously activated. When BLAST~ is asserted, the PCI9060ES is ready to finish the write burst. The 403GC always writes one extra word after XREQ~ is negated. In this application, the RDYI~ input to the PCI9060ES is negated while the last word is being written into the 403GC twice. The causes the PCI9060ES to keep the same data on the bus. The address counter in the DRAMMUX is also prevented from incrementing, so the last word is just written to the same location twice. When the last word is being written, RDYI~ is re-asserted to allow the PCI9060ES to complete the write burst.

3.4.1 Unaligned transfers

The 403GC only accepts the following types of transfers to DRAM from an external master:

- Fullword transfer
- Halfword transfers on halfword boundaries
- Single Byte transfer

The PCI bus allows any combination of byte enables during a write cycle. There are eight different 2 and 3 byte transfers which are considered unaligned by the 403GC. These transfers are converted to multiple single byte transfers by the DRAMCTL state machine in the PCI MACH. The first byte address is loaded into a counter at the beginning of the cycle. This counter is then incremented after each byte transferred by a value determined by the arrangement of bytes being written. The XSIZ inputs to the 403GC are forced to 00, thus requesting single byte transfers. A byte counter is also loaded at the beginning of the cycle, and is used to keep track of how many bytes to transfer.

3.5 PCI9060ES Configuration Registers

The following local configuration registers must be programmed before direct slave accesses to the local bus DRAM can occur:

Register	Offset
PCI Command Register	04h
PCI Base Address for Local Address Space 0	18h
Range for PCI to Local Address Space 0	80h
Local Base Address (Re-map) for PCI to Local Address Space 0	84h
Local Arbitration Register	88h
Big/Little Endian Descriptor Register	8Ch
Bus Region Descriptors for PCI to Local Accesses	98h

These registers are accessed by running a 403GC cycle to the PCI9060ES with the 9060 chip select (CS9060~) asserted. The address offsets for each register are shown in the table.

4. Critical Timing

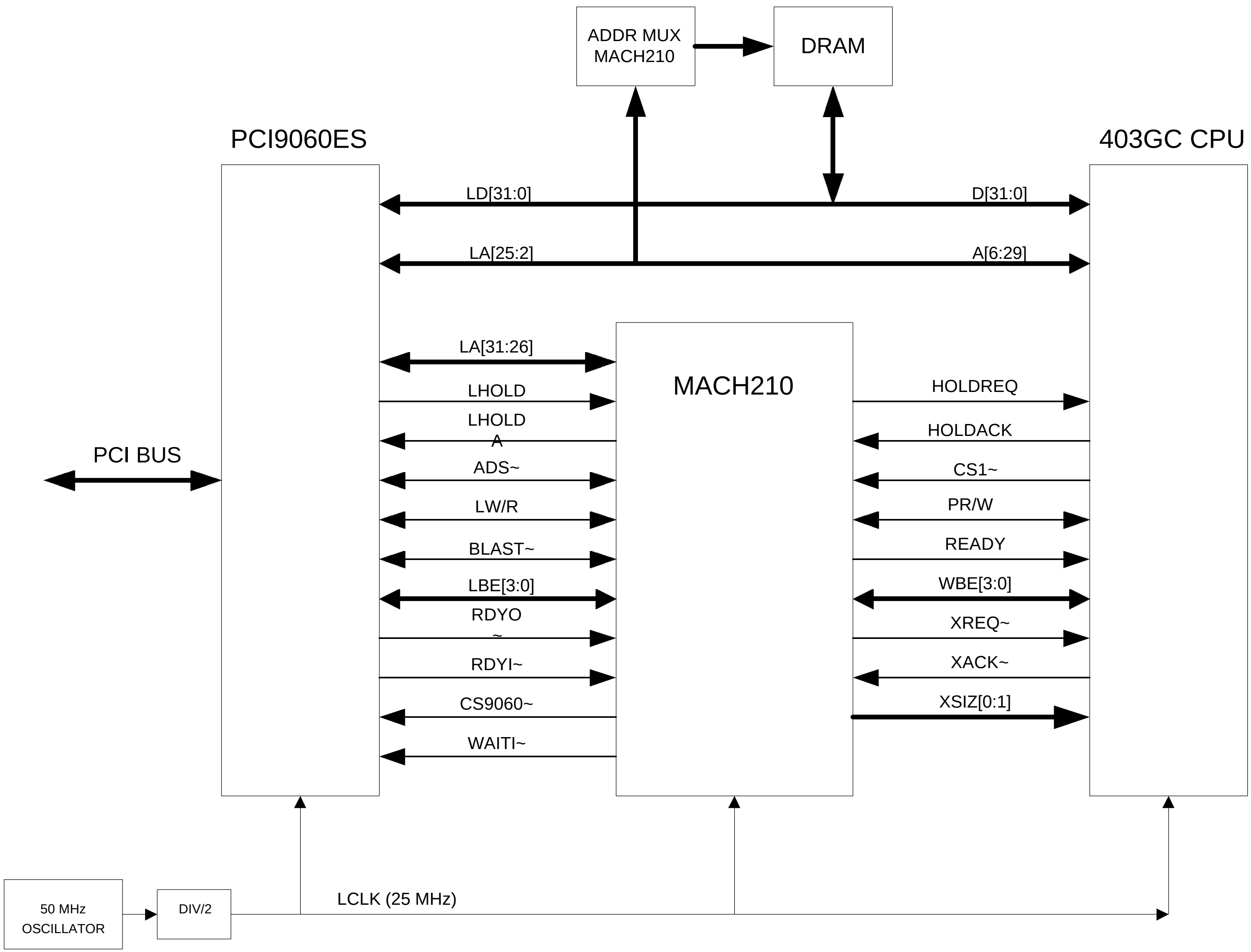
4.1 DRAM Timing

The most critical timing parameter in this design is the DRAM address mux when the 403GC is bursting to or from DRAM. The 25 MHz 403GC has a clock to address delay of 4 to 15 nsec. The delay of the DRAMMUX mach is 7.5 nsec. The minimum clock to CAS delay is ½ the clock period (20 nsec at 25 MHz) plus 4 nsec. If the clock has a 50% duty cycle, then a column address setup time of only 1.5 nsec is provided. This setup time could be increased by using a 33 MHz 403GC which has a clock to address of 4 to 13 nsec, or by using several 5 nsec PALs.

The write data setup time is also very tight. The clock to data of the PCI9060ES is 20 nsec, providing 4 nsec of setup time before the falling edge of CAS.

APPENDIX

Schematics, PAL equations and Timing Diagrams for 16 Word Burst Implementation



ADDR MUX
MACH210
DRAM
PCI9060ES
403GC CPU
LD[31:0]
D[31:0]
LA[25:2]
A[6:29]
LA[31:26]
MACH210
LHOLD
LHOLDA
ADS~
LW/R
BLAST~
LBE[3:0]
RDYO~
RDYI~
CS9060~
WAITI~
HOLDREQ
HOLDACK
CS1~
PR/W
READY
WBE[3:0]
XREQ~
XACK~
XSIZ[0:1]
PCI BUS
50 MHz
OSCILLATOR
DIV/2
LCLK (25 MHz)

```
"****************************************************************
"*Proprietary Rights Notice:                              *
"*                                                        *
"*This material contains the valuable proprietary and trade     *
"*secret information of PLX Technology of Mountain View,        *
"*California.  No part of such information                *
"*may be disclosed, used, reproduced or transmitted in any      *
"*form or by any means--electronic, mechanical or otherwise--   *
"*including photocopying and recording in connection with any   *
"*information storage or retrieval system, without permission   *
"*in writing from PLX Technology.                         *
"****************************************************************
"*Copyright Notice: Copyright (C) 1996, PLX Technology          *
"*            All rights reserved.                        *
"****************************************************************


module  dmuxa

"The following two options are needed, but can't be invoked
"from the source file or from a batch command.  Therefore,
"must compile using the ABEL5 menu.

"options '-reduce none -retain';

title
'
                PLX Technology
**********  PROPRIETARY  INFORMATION  **********

      DMUXA:  DRAM Address Multiplexer
      Engineer:  DLR
      Product:  PowerPC/PCI9060ES
      Part Number:    XXX-XXXX-XXXX
      Revision 2.0    02-12-96

      Copyright PLX, 1996
'

"device declaration
   dmuxa  device  'P16V8C';

"HISTORY
"REV    DATE       AUTHOR            Checksum
"1     02-07-96    David Raaum        0x6e65
"2     02-12-96    David Raaum        0x6de5
"          Don't increment counter when carry is asserted

"This device performs the row and column address multiplexing for
"bits 9:7 of the DRAM address.
"When the PCI9060 is accessing the DRAM, the column address
"is latched if AMUXCAS is asserted.  This is required because
"the 403 performs an extra read after the PCI9060 has finished,
```

```
"and it is desirable to maintain a stable address until the last
"CAS has occurred.
"This device also produces the increment signal for the
"burst address counter.

@page

"pin assignments

    "INPUTS

    !blast       pin 1;  "burst last from 9060
    la9          pin 2;  "address bit 9
    la10         pin 3;  "address bit 10
    la18         pin 4;  "address bit 18
    la19         pin 5;  "address bit 19
    la20         pin 6;  "address bit 20
    la21         pin 7;  "address bit 21
    holdack      pin 8;  "hold acknowledge from 403GC
    amuxcas      pin 9;  "high for column address
    carry        pin 18; "burst counter carry
    !xack        pin 11; "403 transfer acknowledge

    "OUTPUTS

    mxa7         pin 12; "DRAM multiplexed addr bit 7
    mxa8         pin 13; "DRAM multiplexed addr bit 8
    incr         pin 14; "increment address counter
    colr7        pin 15; "column latch bit 7
    colr8        pin 16; "column latch bit 8
    colr9        pin 17; "column latch bit 9
    mxa9         pin 19; "DRAM multiplexed addr bit 9

    mxa7         istype 'com,pos';
    mxa8         istype 'com,pos';
    mxa9         istype 'com,pos';
    incr         istype 'com,pos';

    colr7        istype 'com,pos';
    colr8        istype 'com,pos';
    colr9        istype 'com,pos';


"constant declarations
    L     = 0;    "define low
    H     = 1;    "define high
    OFF   = 0;
    ON    = 1;
    X     = .X.;  "define don't care
    Z     = .Z.;  "define tri-state
    CK    = .C.;  "define clock

"set definitions
```

```
    COLR   = [colr9..colr7];
    ROW    = [la20..la18];
    COL    = [la21,la10..la9];
    MXA    = [mxa9..mxa7];


"macro definitions
latch  macro (d,le,q) {(((?d)&(?le)) # ((?q)&!(?le)) # ((?d)&(?q)))}

equations

"-----------------------------------------------------
"
" Combinatorial Logic Equations
"
"-----------------------------------------------------

COLR = latch(COL,!amuxcas,COLR);


when !amuxcas then MXA = ROW
 else when !holdack then MXA = COL
  else MXA = COLR;

incr = xack & !blast & !carry;

end
```

```
module  dmuxb

"The following two options are needed, but can't be invoked
"from the source file or from a batch command.  Therefore,
"must compile using the ABEL5 menu.

"options '-reduce none -retain';

title
'
                PLX Technology
**********  PROPRIETARY  INFORMATION  **********

      DMUXB:  DRAM Address Multiplexer
      Engineer:  DLR
      Product:  PowerPC/PCI9060ES
      Part Number:    XXX-XXXX-XXXX
      Revision 1.0    02-07-96

      Copyright PLX, 1996
'

"device declaration
   dmuxb  device  'P16V8C';

"HISTORY
"REV    DATE       AUTHOR            Checksum
"1      02-07-96    David Raaum        0x6e63

"This device performs the row and column address multiplexing for
"bits 6:4 of the DRAM address.
"When the PCI9060 is accessing the DRAM, the column address
"is latched if AMUXCAS is asserted.  This is required because
"the 403 performs an extra read after the PCI9060 has finished,
"and it is desirable to maintain a stable address until the last
"CAS has occurred.
```

```
"This device also produces the BTERM (burst terminate) signal to
"the PCI9060.

@page

"pin assignments

    "INPUTS

    carry           pin 1;  "burst counter carry
    la6             pin 2;  "address bit 6
    la7             pin 3;  "address bit 7
    la8             pin 4;  "address bit 8
    la15            pin 5;  "address bit 15
    la16            pin 6;  "address bit 16
    la17            pin 7;  "address bit 17
    holdack         pin 8;  "hold acknowledge from 403GC
    amuxcas         pin 9;  "high for column address
    !xack           pin 11; "403 transfer acknowledge

    "OUTPUTS

    mxa4            pin 12; "DRAM multiplexed addr bit 4
    mxa5            pin 13; "DRAM multiplexed addr bit 5
    colr4           pin 15; "column latch bit 4
    colr5           pin 16; "column latch bit 5
    colr6           pin 17; "column latch bit 6
    !bterm          pin 18; "burst terminate
    mxa6            pin 19; "DRAM multiplexed addr bit 6

    mxa4            istype 'com,pos';
    mxa5            istype 'com,pos';
    mxa6            istype 'com,pos';
    bterm           istype 'com,neg';

    colr4           istype 'com,pos';
    colr5           istype 'com,pos';
    colr6           istype 'com,pos';


"constant declarations
    L       = 0;    "define low
    H       = 1;    "define high
    OFF     = 0;
    ON      = 1;
    X       = .X.;  "define don't care
    Z       = .Z.;  "define tri-state
    CK      = .C.;  "define clock

"set definitions

    COLR   = [colr6..colr4];
    ROW    = [la17..la15];
    COL    = [la8..la6];
```

```
    MXA    = [mxa6..mxa4];


"macro definitions
latch  macro (d,le,q) {(((?d)&(?le)) # ((?q)&!(?le)) # ((?d)&(?q)))}

equations

"-------------------------------------------------------
"
" Combinatorial Logic Equations
"
"-------------------------------------------------------

COLR = latch(COL,!amuxcas,COLR);


when !amuxcas then MXA = ROW
 else when !holdack then MXA = COL
  else MXA = COLR;

bterm = xack & carry;

end
```

```
"*****************************************************************
"*Proprietary Rights Notice:                               *
"*                                                        *
"*This material contains the valuable proprietary and trade     *
"*secret information of PLX Technology of Mountain View,        *
"*California.  No part of such information                      *
"*may be disclosed, used, reproduced or transmitted in any      *
"*form or by any means--electronic, mechanical or otherwise--   *
"*including photocopying and recording in connection with any   *
"*information storage or retrieval system, without permission   *
"*in writing from PLX Technology.                               *
"*****************************************************************
"*Copyright Notice: Copyright (C) 1996, PLX Technology          *
"*            All rights reserved.                              *
"*****************************************************************


module  dmuxc

"The following two options are needed, but can't be invoked
"from the source file or from a batch command.  Therefore,
"must compile using the ABEL5 menu.

"options '-reduce none -retain';

title
'
                PLX Technology
**********  PROPRIETARY  INFORMATION  **********

      DMUXC:  DRAM Address Multiplexer
      Engineer:  DLR
      Product:  PowerPC/PCI9060ES
      Part Number:    XXX-XXXX-XXXX
      Revision 1.0    02-07-96

      Copyright PLX, 1996
'

"device declaration
   dmuxc  device  'P16V8C';

"HISTORY
"REV    DATE       AUTHOR             Checksum
"1      02-07-96    David Raaum         0x43fd

"This device performs the row and column address multiplexing for
"bits 3:0 of the DRAM address.

@page

"pin assignments
```

```
    "INPUTS

    la2          pin 1;  "address bit 2
    la3          pin 2;  "address bit 3
    la4          pin 3;  "address bit 4
    la5          pin 4;  "address bit 5
    la11         pin 5;  "address bit 11
    la12         pin 6;  "address bit 12
    la13         pin 7;  "address bit 13
    la14         pin 8;  "address bit 14
    holdack      pin 9;  "hold acknowledge from 403GC
    amuxcas      pin 11; "high for column address
    colr0        pin 14; "column burst address bit 2
    colr1        pin 15; "column burst address bit 3
    colr2        pin 16; "column burst address bit 4
    colr3        pin 17; "column burst address bit 5

    "OUTPUTS

    mxa0         pin 12; "DRAM multiplexed addr bit 0
    mxa1         pin 13; "DRAM multiplexed addr bit 1
    mxa2         pin 18; "DRAM multiplexed addr bit 2
    mxa3         pin 19; "DRAM multiplexed addr bit 3

    mxa0         istype 'com,pos';
    mxa1         istype 'com,pos';
    mxa2         istype 'com,pos';
    mxa3         istype 'com,pos';


"constant declarations
    L     = 0;    "define low
    H     = 1;    "define high
    OFF   = 0;
    ON    = 1;
    X     = .X.;  "define don't care
    Z     = .Z.;  "define tri-state
    CK    = .C.;  "define clock

"set definitions

    COLR   = [colr3..colr0];
    ROW    = [la14..la11];
    COL    = [la5..la2];
    MXA    = [mxa3..mxa0];


"macro definitions
latch  macro (d,le,q) {(((?d)&(?le)) # ((?q)&!(?le)) # ((?d)&(?q)))}

equations

"------------------------------------------------------
"
```

```
" Combinatorial Logic Equations
"
"--------------------------------------------------------


when !amuxcas then MXA = ROW
 else when !holdack then MXA = COL
  else MXA = COLR;

end
```

```
"****************************************************************
"*Proprietary Rights Notice:                                    *
"*                                                              *
"*This material contains the valuable proprietary and trade     *
"*secret information of PLX Technology of Mountain View,        *
"*California.  No part of such information                      *
"*may be disclosed, used, reproduced or transmitted in any      *
"*form or by any means--electronic, mechanical or otherwise--   *
"*including photocopying and recording in connection with any   *
"*information storage or retrieval system, without permission   *
"*in writing from PLX Technology.                               *
"****************************************************************
"*Copyright Notice: Copyright (C) 1996, PLX Technology          *
"*          All rights reserved.                                *
"****************************************************************


module  drammux


title
'
                PLX Technology
**********  PROPRIETARY  INFORMATION  **********

       DRAMMUX:  DRAM Address Multiplexer
       Engineer:  DLR
       Product:  PowerPC/PCI9060ES
       Part Number:    XXX-XXXX-XXXX
       Revision 1.0    01-25-96

       Copyright PLX, 1996
'

"device declaration
   drammux  device  'MACH210A';

"HISTORY
"REV    DATE       AUTHOR          Checksum
"1      01-25-96    David Raaum        0x1e00

"This device performs the row and column address multiplexing for
"the DRAM.  When the PCI9060 is accessing the DRAM, the column address
"is latched, and a counter provides the least significant two bits of
"the burst address.

@page

"pin assignments

   "INPUTS

   lclk          pin 13;   "Local bus clock (25 MHz)
   !lreset        pin 32;   "Local reset
```

```
    "9060/403GC address bus
    "for 403GC a29..a10
    la2..la21       pin 3,8,20,9,7,6,4,2,38,40,
                    42,37,31,29,27,25,30,14,16,18;

    amuxcas         pin 10;   "high for column address
    holdack         pin 11;   "hold acknowledge from 403GC
    !xack           pin 5;   "403GC transfer acknowledge
    !blast          pin 36;   "local burst last signal
    !ads            pin 35;   "address strobe
    !oe             pin 33;   "output enable


    "OUTPUTS

    "DRAM multiplexed address
    mxa0..mxa9      pin 39,15,28,26,24,17,19,21,43,41;

    "BURIED LOGIC

    colr9..colr0    node ;  "column address register/counter

    mxa0..mxa9      istype 'com,pos';

    colr9           istype 'reg';
    colr8           istype 'reg';
    colr7           istype 'reg';
    colr6           istype 'reg';
    colr5           istype 'reg';
    colr4           istype 'reg';
    colr3           istype 'reg';
    colr2           istype 'reg';
    colr1           istype 'reg';
    colr0           istype 'reg';

"constant declarations
    L       = 0;    "define low
    H       = 1;    "define high
    OFF     = 0;
    ON      = 1;
    X       = .X.;  "define don't care
    Z       = .Z.;  "define tri-state
    CK      = .C.;  "define clock

"set definitions

    COLR   = [colr9..colr0];
    COLRL  = [colr1..colr0];
    COLRU  = [colr9..colr2];
    ROW    = [la20..la11];
    COL    = [la21,la10..la2];
    MXA    = [mxa9..mxa0];
```

```
equations

"-----------------------------------------------------
"
" Clocks and Output Enables
"
"-----------------------------------------------------

"Clock assignments

COLR.CLK = lclk;

"Output enables

MXA.oe = oe;

"-----------------------------------------------------
"
" Combinatorial Logic Equations
"
"-----------------------------------------------------


when !amuxcas then MXA = ROW
 else when !holdack then MXA = COL
  else MXA = COLR;


"-----------------------------------------------------
"
" Sequential Logic
"
"-----------------------------------------------------


" Burst address counter
when lreset then COLRL := 0
 else when ads then COLRL := [la3,la2];
  else when !blast & xack then COLRL := COLRL + 1
   else COLRL := COLRL;

" Column address register
when lreset then COLRU := 0
 else when ads then COLRU := [la21,la10..la4];
  else COLRU := COLRU;


"-----------------------------------------------------
"
" Test Vectors
"
"-----------------------------------------------------
```

```
test_vectors
([lclk,oe,lreset, ads,holdack,amuxcas,blast,xack, ROW, COL] -> MXA)
[CK,L,X, X,X,X,X,X, X,X] -> Z;
[CK,L,H, X,X,X,X,X, X,X] -> Z;
[CK,H,H, L,L,L,L,L, 0,0] -> 0;


"Check row address
@const i = 1;
@repeat 10 {
[CK,H,L, L,L,L,L,L, @expr i;,0] -> @expr i;;
[CK,H,L, L,H,L,L,L, @expr i;,0] -> @expr i;;
@const i = i < 1;
}

"Check column address
@const i = 1;
@repeat 10 {
[CK,H,L, L,L,H,L,L, 0,@expr i;] -> @expr i;;
@const i = i < 1;
}

"Check column address register from 9060
@const i = 1;
@repeat 10 {
[CK,H,L, H,H,H,L,L, 0,@expr i;] -> @expr i;;
[CK,H,L, L,H,H,L,L, 0,       0] -> @expr i;;
@const i = i < 1;
}

"Check burst counter
@const i = 0;
@repeat 3 {
[CK,H,L, H,H,H,L,L, 0,@expr i;] -> @expr i;;
[CK,H,L, L,H,H,L,L, 0,       0] -> @expr i;;
[CK,H,L, L,H,H,H,H, 0,       0] -> @expr i;;
[CK,H,L, L,H,H,L,H, 0,       0] -> @expr i+1;;
@const i = i + 1;
}

end
```

```
"***************************************************************
"*Proprietary Rights Notice:                                   *
"*                                                             *
"*This material contains the valuable proprietary and trade    *
"*secret information of PLX Technology of Mountain View,       *
"*California.  No part of such information                     *
"*may be disclosed, used, reproduced or transmitted in any     *
"*form or by any means--electronic, mechanical or otherwise--  *
"*including photocopying and recording in connection with any  *
"*information storage or retrieval system, without permission  *
"*in writing from PLX Technology.                              *
"***************************************************************
"*Copyright Notice: Copyright (C) 1996, PLX Technology         *
"*        All rights reserved.                                 *
"***************************************************************


module  pci


title
'
              PLX Technology
**********  PROPRIETARY  INFORMATION  **********

     PCICTL:  PCI9060 to 403GC Control
     Engineer:  DLR
     Product:  PowerPC/PCI9060ES
     Part Number:    XXX-XXXX-XXXX
     Revision 5.0    05-06-96

     Copyright PLX, 1996
'

"device declaration
   pci  device  'MACH210A';

"HISTORY
"REV    DATE       AUTHOR            Checksum
"1     01-25-96    David Raaum        0x71b1
"2     02-12-96    David Raaum        0x584f
"      Use burst counter carry to stop a burst cycle
"3     02-17-96    David Raaum        0x49df
"      Make XREQ~ combinational
"4     03-20-96    David Raaum        0x
"      Swap LBE signals in the UNALIGNED, A0, and A1 equations for big endian
"5     05-06-96    David Raaum        0x4a1a
"      CS9060 should be declared active low

"This pal monitors direct slave cycles from the PCI9060ES.
"When a direct slave cycle is detected, the 403GC local bus
"is acquired and a single or burst cycle is run to the
"local DRAM.
```

```
"A dedicated chip select from the 403GC is used to access
"the PCI9060.  Depending on the address, the access can be
"to either the PCI9060 internal registers, or to the PCI BUS.
"Only single cycle transfers are supported, although an
"external master on the local bus could perform burst
"transfers to the PCI bus.

@page

"pin assignments

    "INPUTS

    lclk          pin 13;   "Local bus clock (25 MHz)
    !lreset       pin 32;   "Local reset
    lhold         pin 16;   "local bus hold
    holdack       pin 11;   "hold acknowledge from 403GC
    !xack         pin 10;   "transfer ack from 403GC
    !cs           pin 20;   "chip select from 403GC
    !rdyo         pin 35;   "ready output from 9060ES
    la25          pin 29;   "403 addr bit 6 (LA25)
    la24          pin 30;   "403 addr bit 7 (LA24)
    carry         pin 25;   "burst counter carry
    !oe           pin 33;   "output enable


    "OUTPUTS

    !xreq         pin 19;   "transfer request
    xsiz1..xsiz0    pin 28,26;"transfer size
    !rdyi         pin 15;   "ready into 9060ES
    holdreq       pin 17;   "hold request to 403GC
    ready         pin 4;    "403GC ready
    _waiti        pin 6;    "wait input to 9060
    !cs9060       pin 5;    "Chip select for 9060 internal registers
    la28..la31      pin 43,14,21,18; "upper 9060 address bits

    "IN/OUT
    !ads          pin 2;   "address strobe
    lw_r          pin 8;   "Local write/read status
    !lbe3..!lbe0    pin 42,7,37,39;   "local byte enables
    !blast        pin 3;   "local burst last signal
    pr_w          pin 40;   "403GC read/write status

    wbe0          pin 38;   "byte enb 0 when 403 is master;
                            "A4 (LA27) when 9060 is master

    wbe1          pin 36;   "byte enb 1 when 403 is master;
                            "A5 (LA26) when 9060 is master

    wbe2          pin 31;   "byte enb 2 when 403 is master;
                            "A30 (LA1) when 9060 is master

    wbe3          pin 27;   "byte enb 3 when 403 is master;
```

```
                        "A31 (LA0) when 9060 is master

la26..la27      pin 41,9;   "9060 address bits

"BURIED LOGIC

q0              node ;  "state variable
q1              node ;  "state variable
q2              node ;  "state variable
q3              node ;  "state variable
q4              node ;  "state variable
rdyenb          node ;  "ready enable
uctr0           node ;  "unaligned counter bit 0
uctr1           node ;  "unaligned counter bit 1
uadr30          node ;  "unaligned addr bit 30
uadr31          node ;  "unaligned addr bit 31
inc2            node ;  "increment unaligned addr by 2
inc3            node ;  "increment unaligned addr by 3
usiz0           node ;  "unaligned size bit 0
usiz1           node ;  "unaligned size bit 1
unalign         node ;  "unaligned transfer
cycreq          node ;  "PCI9060 has request a cycle
wrburst         node ;  "write burst in progress
xreqsm          node ;  "xreq from state machine


xreq            istype 'com,neg';
_waiti          istype 'reg,neg';
xsiz1           istype 'com,pos';
xsiz0           istype 'com,pos';
rdyi            istype 'com,neg';
holdreq         istype 'com,pos';
ready           istype 'reg,pos';
wbe0            istype 'com,neg';
wbe1            istype 'com,neg';
wbe2            istype 'com,neg';
wbe3            istype 'com,neg';
cs9060          istype 'com,neg';
la26            istype 'com,pos';
la27            istype 'com,pos';
la28            istype 'com,pos';
la29            istype 'com,pos';
la30            istype 'com,pos';
la31            istype 'com,pos';

pr_w            istype 'com,neg';
lw_r            istype 'com,neg';
ads             istype 'reg,neg';
blast           istype 'com,neg';
lbe0            istype 'com,neg';
lbe1            istype 'com,neg';
lbe2            istype 'com,neg';
lbe3            istype 'com,neg';
```

```
    q0              istype 'reg';
    q1              istype 'reg';
    q2              istype 'reg';
    q3              istype 'reg';
    q4              istype 'reg';
    xreqsm          istype 'reg';
    uctr0           istype 'reg';
    uctr1           istype 'reg';
    uadr30          istype 'reg';
    uadr31          istype 'reg';
    cycreq          istype 'reg';
    rdyenb          istype 'reg';
    wrburst         istype 'reg';
    usiz0           istype 'com';
    usiz1           istype 'com';
    unalign         istype 'com';
    inc2            istype 'com';
    inc3            istype 'com';

"constant declarations
    L       = 0;    "define low
    H       = 1;    "define high
    OFF     = 0;
    ON      = 1;
    X       = .X.;  "define don't care
    Z       = .Z.;  "define tri-state
    CK      = .C.;  "define clock

"set definitions

    XSIZ   = [xsiz0,xsiz1];
    USIZ   = [usiz1,usiz0];
    LBE    = [lbe3,lbe2,lbe1,lbe0];
    UCOUNT = [uctr1,uctr0];
    UADDR  = [uadr30,uadr31];
    HILA   = [la31,la30,la29,la28,la27,la26];
    WBE    = [wbe0,wbe1,wbe2,wbe3];

"define state variables as positive

    waiti = !_waiti;

"state declarations


    DRAMCTL = [xreqsm,rdyenb, q0,q1,q2];
    IDLE    = [L,L, L,L,L]; "00
    S0      = [L,L, L,L,H]; "01 wait for ads
    S1      = [H,H, L,L,L]; "18 start aligned write
    S2      = [H,H, H,L,L]; "1C burst write
    S3      = [L,L, H,L,L]; "04 last write
    S4      = [L,H, L,L,L]; "08 overrun write
    S5      = [H,L, L,L,L]; "10 start unaligned write
    S6      = [L,L, L,H,L]; "02 wait for xack
```

```
    S7      = [H,H, L,H,L]; "1A start read
    S8      = [H,H, H,H,L]; "1E burst read
    S9      = [L,H, H,L,L]; "0C blast detected, remove xreq
    S10     = [L,L, H,H,L]; "06 wait for 403GC to finish
    S11     = [L,H, H,H,L]; "0E single cycle read


    PCICTL = [ads,ready,waiti,q3,q4];
    PIDLE  = [L,H,L, L,L];
    P0     = [H,L,L, L,L];  "strobe addr into 9060
    P1     = [L,L,L, L,L];  "wait for RDYO from 9060
    P2     = [L,H,L, H,L];  "assert rdy
    P3     = [L,H,L, L,H];  "403 writes data

    P4     = [H,L,H, L,L];  "strobe addr into 9060
    P5     = [L,L,H, H,L];  "wait for RDYO from 9060
    P6     = [L,H,H, L,L];  "assert rdy
    P7     = [L,H,L, H,H];  "403 reads data

    P8     = [H,H,H, H,H];  "all high

"intermediate equations

READ = !lw_r;
WRITE = lw_r;
BURST = !blast;

"Number of bytes transfered by 9060
BYTE = ((!lbe3 & !lbe2 & !lbe1 & lbe0) #
       (!lbe3 & !lbe2 & lbe1  & !lbe0) #
       (!lbe3 &  lbe2 & !lbe1 & !lbe0) #
       (lbe3  & !lbe2 & !lbe1 & !lbe0));

HALFWORD = ((!lbe3 & !lbe2 & lbe1 & lbe0) #
           (lbe3 & lbe2 & !lbe1 & !lbe0));

FULLWORD = (lbe3 & lbe2 & lbe1 & lbe0);

"Some 9060 operations are unaligned with respect to 403GC
"The 403 only accepts long words, half words aligned at
"address 0 or 2, or single byte transfers.  All other
"types of transfers from the 9060 must be broken up into
"multiple single byte transfers.

"Types of two byte unaligned transfers
UNALIGN2A = (!lbe0 & lbe1  & lbe2  & !lbe3);
UNALIGN2B = (lbe0 & !lbe1  & !lbe2  & lbe3);
UNALIGN2C = (lbe0 & !lbe1  & lbe2  & !lbe3);
UNALIGN2D = (!lbe0 & lbe1  & !lbe2  & lbe3);
UNALIGN2 = (WRITE & (UNALIGN2A # UNALIGN2B # UNALIGN2C # UNALIGN2D));

"Types of three byte unaligned transfers
UNALIGN3A = (!lbe0 & lbe1  & lbe2  & lbe3);
```

```
UNALIGN3B = (lbe0 & lbe1  & lbe2  & !lbe3);
UNALIGN3C = (lbe0 & !lbe1  & lbe2  & lbe3);
UNALIGN3D = (lbe0 & lbe1  & !lbe2  & lbe3);
UNALIGN3 = (WRITE & (UNALIGN3A # UNALIGN3B # UNALIGN3C # UNALIGN3D));

UNALIGN =  UNALIGN2 # UNALIGN3;

"Unaligned address increment value depends on type of unalignment
INC2 = (xack &
        ((UNALIGN2C # UNALIGN2D) #
         (UNALIGN3C & uadr31) #
         (UNALIGN3D & !uadr30)));

INC3 = (xack & UNALIGN2B);


" Unaligned transfer is complete
UDONE    = (UCOUNT == 0);

" Address bits 0 and 1 derived from the 9060 LBEs
" This is the address of the first active byte in a word
A1 = (!lbe2 & !lbe3);
A0 = ((!lbe3 & !lbe1) # (!lbe3 & lbe2));
BYTEADDR = [A1,A0];

"Time to start a new cycle from the 9060
CYCSTART = (ads.PIN # cycreq);


equations

"-------------------------------------------------------
"
" Clocks and Output Enables
"
"-------------------------------------------------------

"Clock assignments

DRAMCTL.CLK = lclk;
PCICTL.CLK = lclk;
UCOUNT.CLK = lclk;
UADDR.CLK = lclk;
cycreq.CLK = lclk;
wrburst.CLK = lclk;

"Output enables

holdreq.oe = oe;
xreq.oe = oe;
rdyi.oe = oe;
waiti.oe = oe;
cs9060.oe = oe;
ready.oe = oe;
```

```
"when 9060 is local bus master
pr_w.oe = oe & holdack;
XSIZ.oe = oe & holdack;
WBE.oe = oe & holdack;

"when 403 is local bus master
ads.oe   = oe & !holdack;
lw_r.oe  = oe & !holdack;
LBE.oe   = oe & !holdack;
blast.oe = oe & !holdack;
HILA.oe  = oe & !holdack;

"-----------------------------------------------------
"
" Combinatorial Logic Equations
"
"-----------------------------------------------------


xreq = (lhold & ads) # xreqsm;
rdyi = rdyenb & xack;
pr_w = !lw_r;
wbe0 = la27;                              // 403 addr A4
wbe1 = la26;                              // 403 addr A5
wbe2 = !wrburst & blast & WRITE & !uadr30;  // 403 addr A30
wbe3 = !wrburst & blast & WRITE & !uadr31;  // 403 addr A31 (LSB)

when UNALIGN3 then USIZ = 2
 else when UNALIGN2 then USIZ = 1
  else USIZ = 0;

when BURST then XSIZ = 3
 else when FULLWORD then XSIZ = 2
  else when HALFWORD then XSIZ = 1
   else XSIZ = 0; "(BYTES and UNALIGNED)

lw_r = !pr_w;
LBE = !WBE;
blast = ON;
cs9060 = cs & !la25 & !la24;
HILA = 0;
holdreq = lhold # (DRAMCTL == S10);

unalign  = UNALIGN;
inc2 = INC2;
inc3 = INC3;

"-----------------------------------------------------
"
" Sequential Logic
"
"-----------------------------------------------------
```

```
"Set cycle request in case state machine is busy when
"ads occurs.
cycreq := !lreset & ((holdack & ads) # (cycreq & !xreq));

"Need to know if write burst in progress, so that A30 and A31
" can be kept at 00.
when lreset then wrburst := 0
 else when DRAMCTL == S2 then wrburst := 1
  else when (DRAMCTL == S4) & xack then wrburst := 0
   else wrburst := wrburst;

" Unaligned byte counter
when lreset then UCOUNT := 0
 else when CYCSTART then UCOUNT := USIZ
  else when xack then UCOUNT := UCOUNT-1
   else UCOUNT := UCOUNT;

" Unaligned address counter
when lreset then UADDR := 0
 else when CYCSTART then UADDR := BYTEADDR
  else when inc3 & WRITE then UADDR := UADDR+3
   else when inc2 & WRITE then UADDR := UADDR+2
    else when xack & WRITE then UADDR := UADDR+1
     else UADDR := UADDR;



"--------------------------------------------------------
"
" PCI9060 DRAM Read/Write State Machine
"
"--------------------------------------------------------

state_diagram DRAMCTL

state IDLE:
    if lreset then IDLE
     else if lhold then S0
      else IDLE;

state S0:
    if lreset then IDLE
     else if !lhold then IDLE
      else if !CYCSTART then S0
       else if READ then S7
        else if WRITE & unalign then S5
         else S1;

state S1:
    if lreset then IDLE
     else if !blast then S2
      else S4;

state S2:
```

```
    if lreset then IDLE
     else if (blast # carry) then S3
      else S2;

state S3:
    if lreset then IDLE
      else if xack then S4
       else S3;

state S4:
    if lreset then IDLE
     else if xack then IDLE;
       else S4;

state S5:
    if lreset then IDLE
     else if UDONE then S4;
      else S6;

state S6:
    if lreset then IDLE
     else if xack then S5
      else S6;

state S7:
    if lreset then IDLE
     else if blast then S11  "single read
      else S8;              "burst read

state S8:
    if lreset then IDLE
     else if (blast # carry) then S9
      else S8;

state S9:
    if lreset then IDLE
     else if xack then S10
      else S9;

state S10:
    if lreset # xack then IDLE
     else S10;

state S11:
    if lreset # xack then IDLE
     else S11;

"-------------------------------------------------------
"
" 403GC Direct PCI Master State Machine
"
"-------------------------------------------------------

state_diagram PCICTL
```

```
state PIDLE:
    if lreset then PIDLE
     else if cs & !pr_w then P0
      else if cs & pr_w then P4
       else PIDLE;

state P0:
    if lreset then PIDLE
     else P1;

state P1:
    if lreset then PIDLE
     else if rdyo then P2
      else P1;

state P2:
    if lreset then PIDLE
     else P3;

state P3:
    goto PIDLE;


state P4:
    if lreset then PIDLE
     else P5;

state P5:
    if lreset then PIDLE
     else if rdyo then P6
      else P5;

state P6:
    if lreset then PIDLE
     else P7;

state P7:
    goto PIDLE;

state P8:
    goto PIDLE;

"-------------------------------------------------------
"
"TEST VECTORS
"
"-------------------------------------------------------

"-------------------------------------------------------
"
"PCI9060 is master
"
"-------------------------------------------------------
```

```
test_vectors
([lclk,oe,lreset, lhold,holdack,cs, ads,lw_r,blast,LBE,carry, xack,HILA] ->
[holdreq,xreq,xsiz0,xsiz1,pr_w, WBE,rdyi])
[CK,L,X, X,L,L, X,X,X,X,X, X,X] -> [Z,Z,Z,Z,Z, Z,Z];


test_vectors
([lclk,oe,lreset, lhold,holdack,cs, ads,lw_r,blast,LBE,carry, xack,HILA] ->
[DRAMCTL,holdreq,xsiz0,xsiz1,pr_w, WBE,rdyi,unalign,UCOUNT])
[CK,H,H, L,L,L, X,L,X,0,X, X,X]     -> [IDLE,L,Z,Z,Z, Z,L,X,0];
[CK,H,L, L,L,L, L,L,X,0,X, L,X]     -> [IDLE,L,Z,Z,Z, Z,L,X,0];
[CK,H,L, L,L,L, L,L,X,0,X, L,X]     -> [IDLE,L,Z,Z,Z, Z,L,X,0];

"Test high address
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,H,H,L, ^h0,L,L,0];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h01] -> [IDLE,L,H,H,L, ^h4,L,L,0];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h02] -> [IDLE,L,H,H,L, ^h8,L,L,0];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h03] -> [IDLE,L,H,H,L, ^hc,L,L,0];

"Single word write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,0];
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,0];
[CK,H,L, H,H,L, H,H,H,^hf,L, L,^h3F] -> [S1,  H,1,0,L, ^hf,L,L,0]; "V12
[CK,H,L, H,H,L, L,H,H,^hf,L, L,^h3F] -> [S4,  H,1,0,L, ^hf,L,L,0];
[CK,H,L, H,H,L, L,H,H,^hf,L, L,^h3F] -> [S4,  H,1,0,L, ^hf,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];


"Single byte 0 write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V18
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h1,L, L,^h3e] -> [S1,  H,0,0,L, ^h8,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h1,L, L,^h3e] -> [S4,  H,0,0,L, ^h8,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h1,L, L,^h3e] -> [S4,  H,0,0,L, ^h8,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Single byte 1 write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V27
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h2,L, L,^h3d] -> [S1,  H,0,0,L, ^h5,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h2,L, L,^h3d] -> [S4,  H,0,0,L, ^h5,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h2,L, L,^h3d] -> [S4,  H,0,0,L, ^h5,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];
```

```
"Single byte 2 write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V36
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h4,L, L,^h3c] -> [S1,  H,0,0,L, ^h2,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h4,L, L,^h3c] -> [S4,  H,0,0,L, ^h2,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h4,L, L,^h3c] -> [S4,  H,0,0,L, ^h2,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Single byte 3 write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V45
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h8,L, L,^h3f] -> [S1,  H,0,0,L, ^hf,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h8,L, L,^h3f] -> [S4,  H,0,0,L, ^hf,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h8,L, L,^h3f] -> [S4,  H,0,0,L, ^hf,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Lower Halfword write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V54
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h3,L, L,^h3c] -> [S1,  H,0,1,L, ^h1,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h3,L, L,^h3c] -> [S4,  H,0,1,L, ^h1,L,L,0];
[CK,H,L, H,H,L, L,H,H,^h3,L, L,^h3c] -> [S4,  H,0,1,L, ^h1,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Upper Halfword write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V63
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,L,^hc,L, L,^h3c] -> [S1,  H,1,1,L, ^h0,L,L,0];
[CK,H,L, H,H,L, L,H,H,^hc,L, L,^h3c] -> [S4,  H,0,1,L, ^h3,L,L,0];
[CK,H,L, H,H,L, L,H,H,^hc,L, L,^h3c] -> [S4,  H,0,1,L, ^h3,L,L,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type A two byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V72
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h6,L, L,^h3c] -> [S5,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h6,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h6,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h6,L, H,^h3c] -> [S5,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h6,L, L,^h3c] -> [S4,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h6,L, L,^h3c] -> [S4,  H,0,0,L, ^h1,L,H,0];
```

```
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type B two byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V84
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h9,L, L,^h3c] -> [S5,  H,0,0,L, ^h3,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h9,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h9,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h9,L, H,^h3c] -> [S5,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h9,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h9,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type C two byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V96
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^hA,L, L,^h3c] -> [S5,  H,0,0,L, ^h3,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hA,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hA,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hA,L, H,^h3c] -> [S5,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hA,L, L,^h3c] -> [S4,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hA,L, L,^h3c] -> [S4,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type D two byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V108
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h5,L, L,^h3c] -> [S5,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h5,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h5,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h5,L, H,^h3c] -> [S5,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h5,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h5,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];


"Unaligned type A three byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V120
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^h7,L, L,^h3c] -> [S5,  H,0,0,L, ^h2,L,H,2];
[CK,H,L, H,H,L, L,H,H,^h7,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,2];
[CK,H,L, H,H,L, L,H,H,^h7,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,2];
```

```
[CK,H,L, H,H,L, L,H,H,^h7,L, H,^h3c] -> [S5,  H,0,0,L, ^h1,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h7,L, L,^h3c] -> [S6,  H,0,0,L, ^h1,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h7,L, L,^h3c] -> [S6,  H,0,0,L, ^h1,L,H,1];
[CK,H,L, H,H,L, L,H,H,^h7,L, H,^h3c] -> [S5,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h7,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^h7,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type B three byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V135
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^hE,L, L,^h3c] -> [S5,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hE,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hE,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hE,L, H,^h3c] -> [S5,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hE,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hE,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hE,L, H,^h3c] -> [S5,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hE,L, L,^h3c] -> [S4,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hE,L, L,^h3c] -> [S4,  H,0,0,L, ^h1,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type C three byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V150
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^hB,L, L,^h3c] -> [S5,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hB,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hB,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hB,L, H,^h3c] -> [S5,  H,0,0,L, ^h1,L,H,1]; "V156
[CK,H,L, H,H,L, L,H,H,^hB,L, L,^h3c] -> [S6,  H,0,0,L, ^h1,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hB,L, L,^h3c] -> [S6,  H,0,0,L, ^h1,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hB,L, H,^h3c] -> [S5,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hB,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hB,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];

"Unaligned type D three byte write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V165
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,3];
[CK,H,L, H,H,L, H,H,H,^hD,L, L,^h3c] -> [S5,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hD,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hD,L, L,^h3c] -> [S6,  H,0,0,L, ^h3,L,H,2];
[CK,H,L, H,H,L, L,H,H,^hD,L, H,^h3c] -> [S5,  H,0,0,L, ^h2,L,H,1]; "V171
[CK,H,L, H,H,L, L,H,H,^hD,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
[CK,H,L, H,H,L, L,H,H,^hD,L, L,^h3c] -> [S6,  H,0,0,L, ^h2,L,H,1];
```

```
[CK,H,L, H,H,L, L,H,H,^hD,L, H,^h3c] -> [S5,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hD,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,H,^hD,L, L,^h3c] -> [S4,  H,0,0,L, ^h0,L,H,0];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,3];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,3];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];


"Single word read from DRAM
[CK,H,L, H,L,L, L,L,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V180
[CK,H,L, H,L,L, L,L,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,L,L,^h0,L, L,^h00] -> [S0,  H,1,1,H, ^h0,L,L,3];
[CK,H,L, H,H,L, H,L,H,^hf,L, L,^h3F] -> [S7,  H,1,0,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,H,^hf,L, L,^h3F] -> [S11, H,1,0,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,H,^hf,L, L,^h3F] -> [S11, H,1,0,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,L,^h0,L, H,^h00] -> [IDLE,H,1,1,H, ^h0,L,L,3];
[CK,H,L, L,H,L, L,L,L,^h0,L, L,^h00] -> [IDLE,L,1,1,H, ^h0,L,L,3];
[CK,H,L, L,L,L, L,L,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,3];


"Burst read from DRAM
[CK,H,L, H,L,L, L,L,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3]; "V189
[CK,H,L, H,L,L, L,L,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,3];
[CK,H,L, H,H,L, L,L,L,^h0,L, L,^h00] -> [S0,  H,1,1,H, ^h0,L,L,3];
[CK,H,L, H,H,L, H,L,L,^hf,L, L,^h3F] -> [S7,  H,1,1,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,L,^hf,L, L,^h3F] -> [S8,  H,1,1,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,L,^hf,L, L,^h3F] -> [S8,  H,1,1,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,H,^hf,L, L,^h3F] -> [S9,  H,1,0,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,H,^hf,L, L,^h3F] -> [S9,  H,1,0,H, ^hc,L,L,0];
[CK,H,L, H,H,L, L,L,H,^hf,L, H,^h3F] -> [S10, H,1,0,H, ^hc,L,L,3];
[CK,H,L, H,H,L, L,L,H,^hf,L, L,^h3F] -> [S10, H,1,0,H, ^hc,L,L,3];
[CK,H,L, H,H,L, L,L,H,^hf,L, L,^h3F] -> [S10, H,1,0,H, ^hc,L,L,3];
[CK,H,L, H,H,L, L,L,L,^h0,L, H,^h00] -> [IDLE,H,1,1,H, ^h0,L,L,2];
[CK,H,L, L,H,L, L,L,L,^h0,L, L,^h00] -> [IDLE,L,1,1,H, ^h0,L,L,2];
[CK,H,L, L,L,L, L,L,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,2];

"Burst write to DRAM
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,2]; "V203
[CK,H,L, H,L,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,Z,Z,Z,   Z,L,X,2];
[CK,H,L, H,H,L, L,H,L,^h0,L, L,^h00] -> [S0,  H,1,1,L, ^h0,L,L,2];
[CK,H,L, H,H,L, H,H,L,^hf,L, L,^h3F] -> [S1,  H,1,1,L, ^hc,L,L,0];
[CK,H,L, H,H,L, L,H,L,^hf,L, L,^h3F] -> [S2,  H,1,1,L, ^hc,L,L,0];
[CK,H,L, H,H,L, L,H,L,^hf,L, L,^h3F] -> [S2,  H,1,1,L, ^hc,L,L,0];
[CK,H,L, H,H,L, L,H,H,^hf,L, L,^h3F] -> [S3,  H,1,0,L, ^hc,L,L,0];
[CK,H,L, H,H,L, L,H,H,^hf,L, L,^h3F] -> [S3,  H,1,0,L, ^hc,L,L,0];
[CK,H,L, H,H,L, L,H,H,^hf,L, H,^h3F] -> [S4,  H,1,0,L, ^hc,H,L,3];
[CK,H,L, H,H,L, L,H,H,^hf,L, L,^h3F] -> [S4,  H,1,0,L, ^hc,L,L,3];
[CK,H,L, H,H,L, L,H,L,^h0,L, H,^h00] -> [IDLE,H,1,1,L, ^h0,L,L,2];
[CK,H,L, L,H,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,1,1,L, ^h0,L,L,2];
[CK,H,L, L,L,L, L,H,L,^h0,L, L,^h00] -> [IDLE,L,Z,Z,Z,   Z,L,X,2];
```

```
"-------------------------------------------------------
"
"403GC is master
"
"-------------------------------------------------------

test_vectors
([lclk,oe,lreset,lhold,holdack, cs,la25,la24,pr_w, WBE,rdyo] ->
[ads,waiti,ready,lw_r,cs9060, LBE,HILA,blast])
[CK,L,X,X,L, L,X,X,L, 0,X] -> [Z,Z,Z,Z,Z, Z,Z,Z];

test_vectors
([lclk,oe,lreset,lhold,holdack, cs,la25,la24,pr_w, WBE,rdyo] ->
[PCICTL,lw_r,cs9060, LBE,HILA,blast])
[CK,H,H,X,L, L,X,X,L, 0,X] -> [PIDLE,H,L, ^hf,0,H];
[CK,H,L,L,L, L,X,X,L, 0,L] -> [PIDLE,H,L, ^hf,0,H];

"Read from 9060 internal register (32 bit)
[CK,H,L,L,L, H,L,L,H, 0,L] -> [P4,   L,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,H, 0,L] -> [P5,   L,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,H, 0,L] -> [P5,   L,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,H, 0,H] -> [P6,   L,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,H, 0,H] -> [P7,   L,H, ^hf,0,H];
[CK,H,L,L,L, L,L,L,H, 0,L] -> [PIDLE,L,L, ^hf,0,H];

"Write to 9060 internal register (32 bit)
[CK,H,L,L,L, H,L,L,L, 0,L] -> [P0,   H,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,L, 0,L] -> [P1,   H,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,L, 0,L] -> [P1,   H,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,L, 0,H] -> [P2,   H,H, ^hf,0,H];
[CK,H,L,L,L, H,L,L,L, 0,H] -> [P3,   H,H, ^hf,0,H];
[CK,H,L,L,L, L,L,L,L, 0,L] -> [PIDLE,H,L, ^hf,0,H];

"Write to 9060 internal register (8 bit)
[CK,H,L,L,L, H,L,L,L, ^h7,L] -> [P0,   H,H, ^h8,0,H];
[CK,H,L,L,L, H,L,L,L, ^h7,L] -> [P1,   H,H, ^h8,0,H];
[CK,H,L,L,L, H,L,L,L, ^h7,L] -> [P1,   H,H, ^h8,0,H];
[CK,H,L,L,L, H,L,L,L, ^h7,H] -> [P2,   H,H, ^h8,0,H];
[CK,H,L,L,L, H,L,L,L, ^h7,H] -> [P3,   H,H, ^h8,0,H];
[CK,H,L,L,L, L,L,L,L, ^h7,L] -> [PIDLE,H,L, ^h8,0,H];


"Direct master read from 9060 (32 bit)
[CK,H,L,L,L, H,L,H,H, 0,L] -> [P4,   L,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,H, 0,L] -> [P5,   L,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,H, 0,L] -> [P5,   L,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,H, 0,H] -> [P6,   L,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,H, 0,H] -> [P7,   L,L, ^hf,0,H];
[CK,H,L,L,L, L,L,H,H, 0,L] -> [PIDLE,L,L, ^hf,0,H];


"Direct master write to 9060 (32 bit)
[CK,H,L,L,L, H,L,H,L, 0,L] -> [P0,   H,L, ^hf,0,H];
```

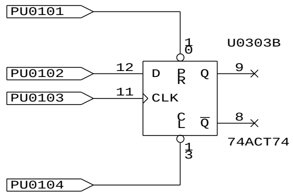
```
[CK,H,L,L,L, H,L,H,L, 0,L] -> [P1,   H,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,L, 0,L] -> [P1,   H,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,L, 0,H] -> [P2,   H,L, ^hf,0,H];
[CK,H,L,L,L, H,L,H,L, 0,H] -> [P3,   H,L, ^hf,0,H];
[CK,H,L,L,L, L,L,H,L, 0,L] -> [PIDLE,H,L, ^hf,0,H];

test_vectors
([lclk,oe,lreset,lhold,holdack, cs,la25,la24,pr_w, WBE,rdyo] -> [ads,waiti,ready,lw_r,cs9060,
LBE,HILA,blast])

"Get hold request from 9060, 403GC asserts holdack
" to test diable of these signals
[CK,H,L,H,H, L,X,X,L, 0,L] -> [Z,L,H,Z,L, Z,Z,Z];

end
```

|LINK
|N2.SCH
|N3.SCH
|N4.SCH
|N5.SCH
|N6.SCH
|N7.SCH

ECN HISTORY

DESCRIPTION	DATE	APPROVAL
REV 1 --	01/24/96	
REV 2 -- CHANGED DRMMUX	2/17/96	

SPARE GATES:



THIS DRAWING IS THE EXCLUSIVE PROPERTY OF PLX, ISSUED IN STRICT CONFIDENCE AND SHALL NOT, WITHOUT PRIOR WRITTEN PERMISSION OF PLX BE REPRODUCED, COPIED, OR USED FOR ANY PURPOSE WHATSOEVER EXCEPT AS IDENTIFIED BY PLX.

APPROVALS	DATE
DRAWN D. RAAUM	01/24/96
CHECKED	
PROD. ENG.	
DESIGN ENG. D. RAAUM	01/24/96

PLX TECHNOLOGY

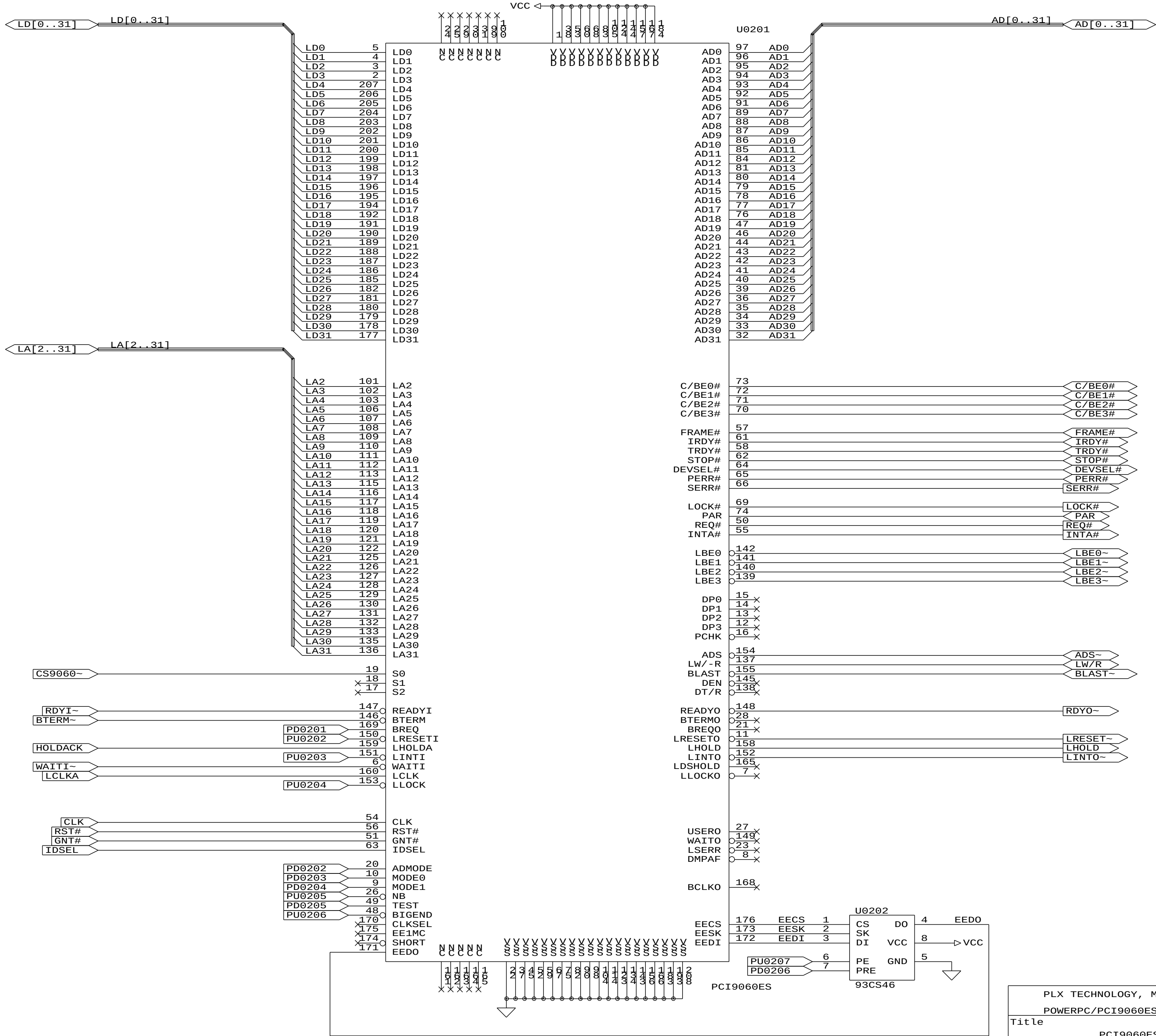
PLX TECHNOLOGY, MOUNTAIN VIEW, CA

POWERPC/PCI9060ES SCHEMATIC DIAGRAM

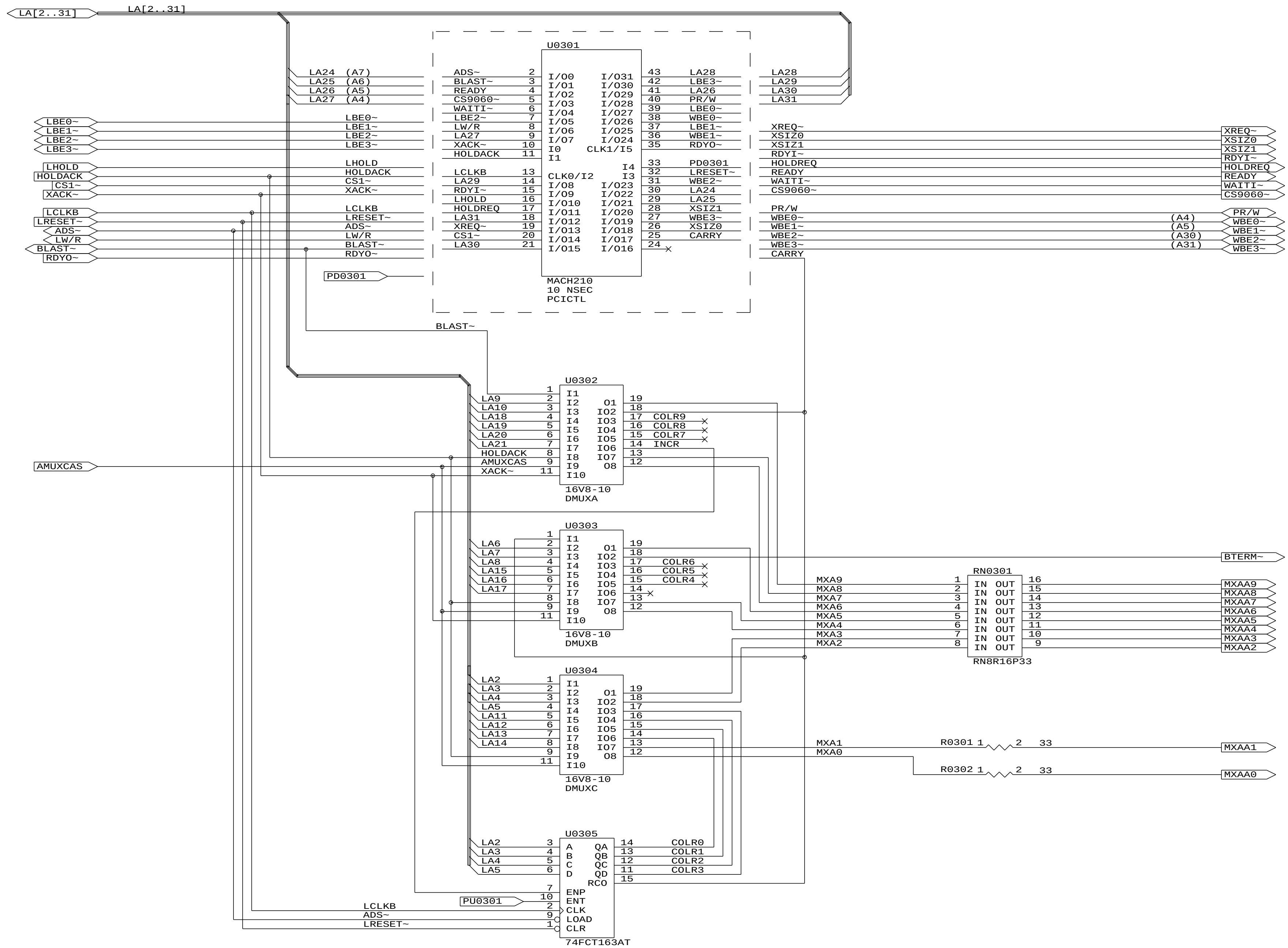
Title: TITLE PAGE

Size	Document Number	REV
B	XXX-XXXX-XXXX	2

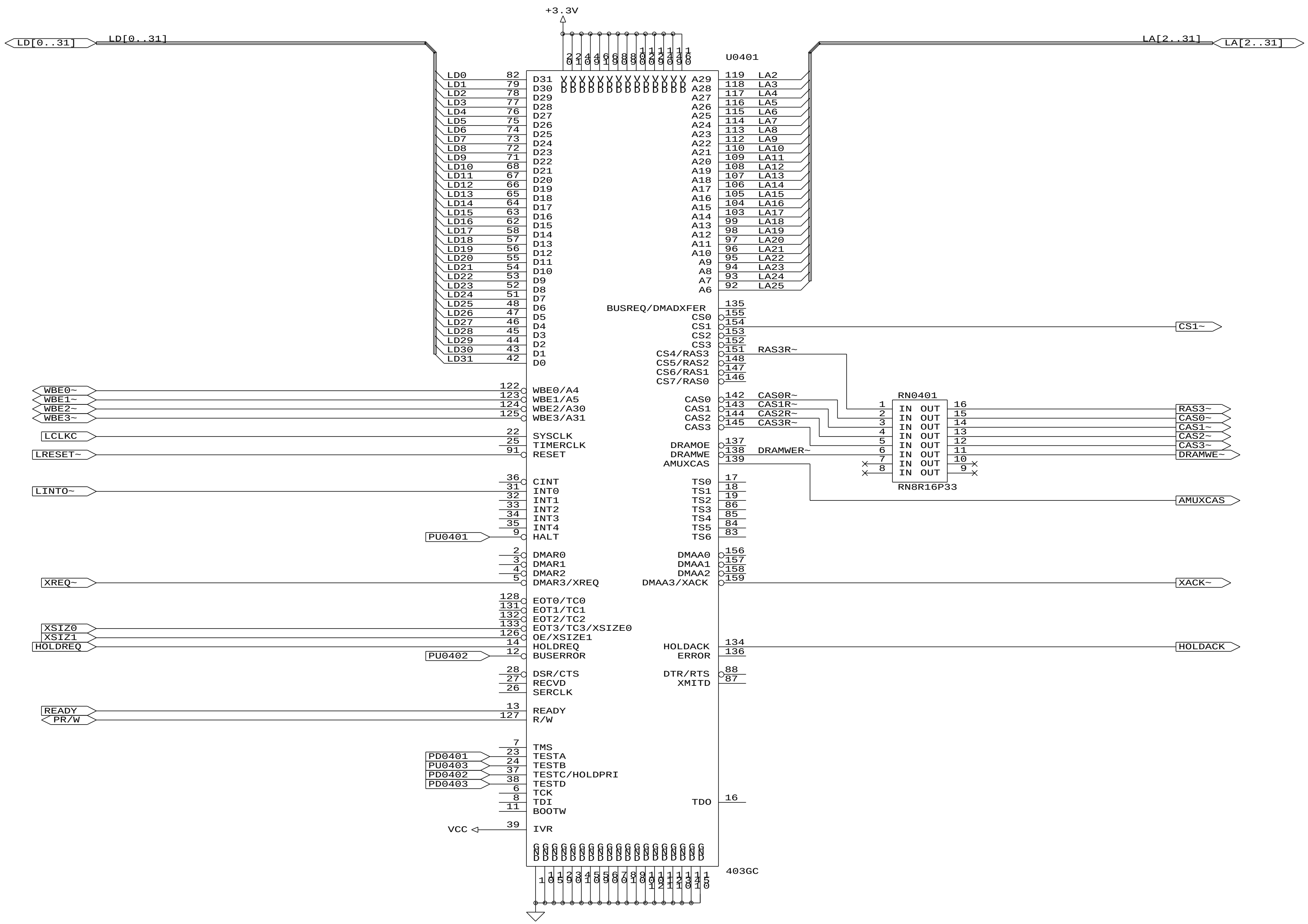
Date: February 17, 1996 Sheet 1 of 7



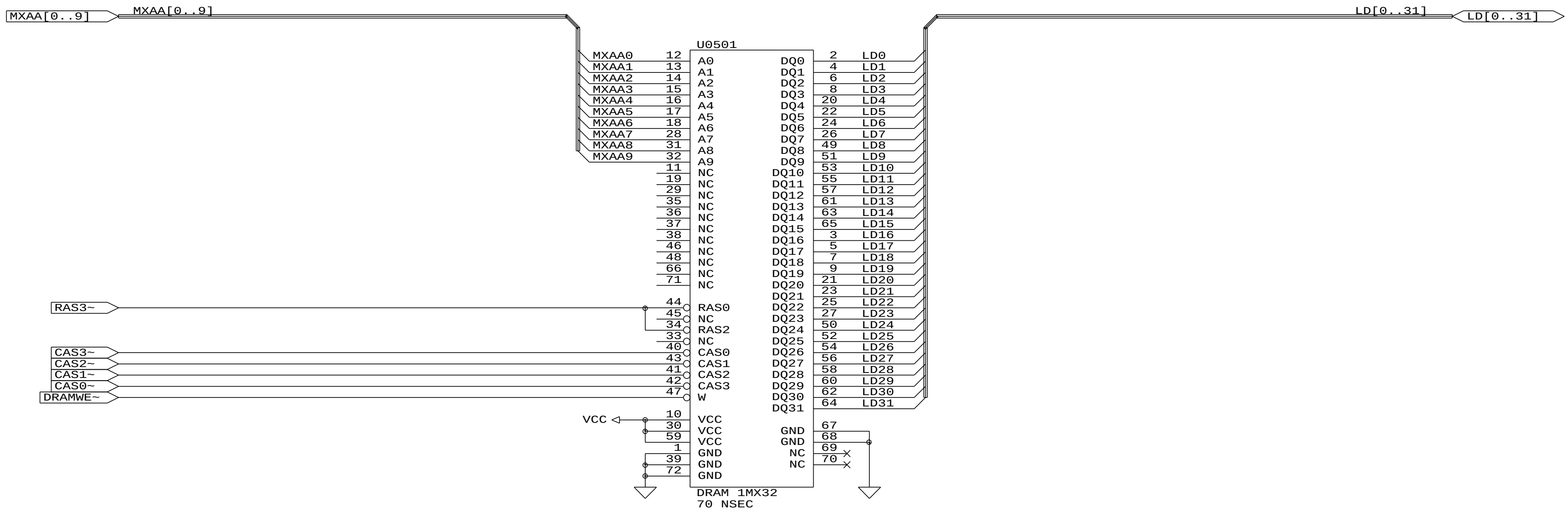
PLX TECHNOLOGY, MOUNTAIN VIEW, CA		
POWERPC/PCI9060ES SCHEMATIC DIAGRAM		
Title: PCI9060ES CONTROLLER		
Size: B	Document Number: XXX-XXXX-XXXX	REV: 2
Date: February 17, 1996	Sheet 2 of 7	



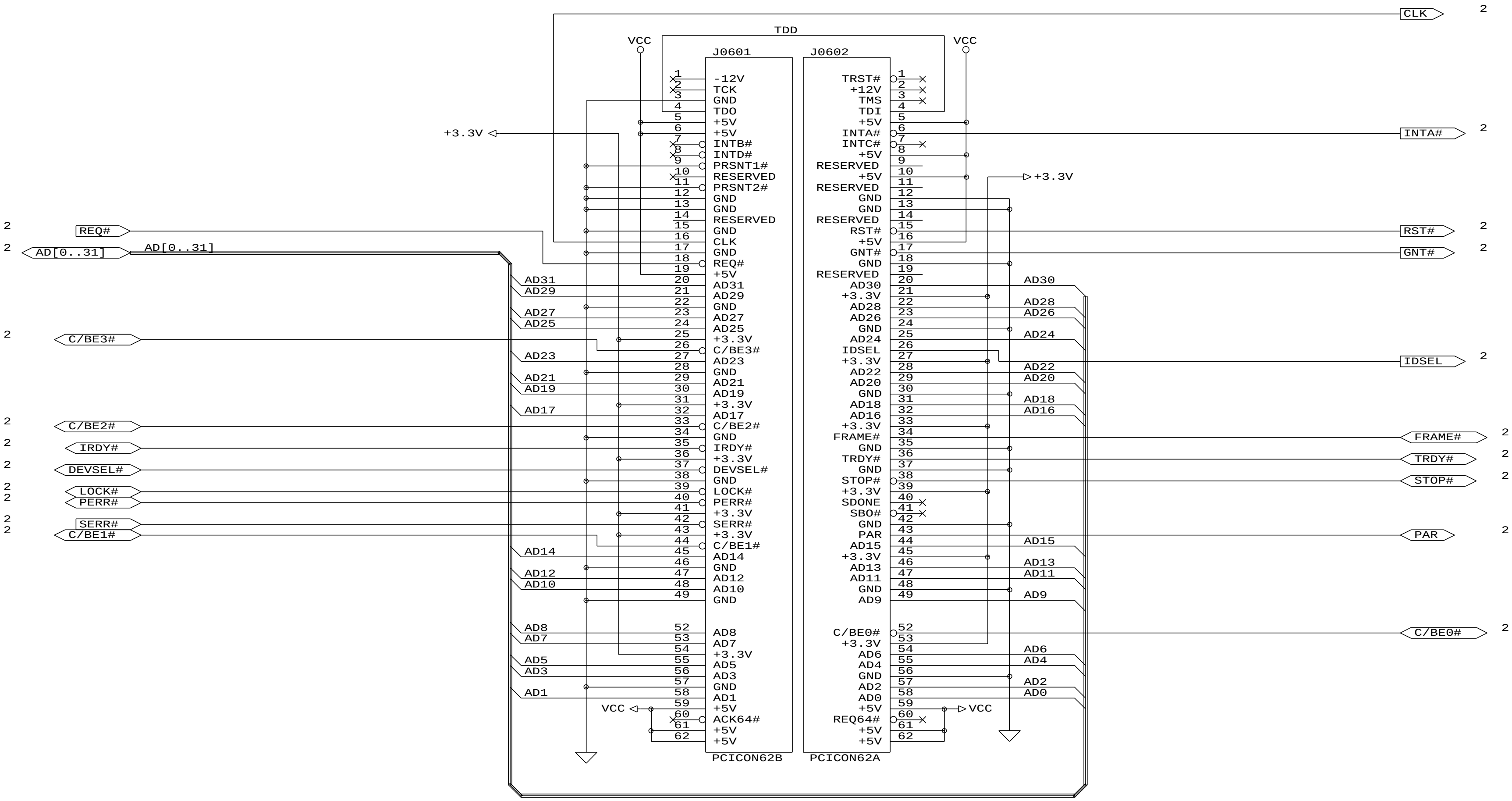
PLX TECHNOLOGY, MOUNTAIN VIEW, CA POWERPC/PCI9060ES SCHEMATIC DIAGRAM		
Title: LOCAL BUS CONTROL		
Size: B	Document Number: XXX-XXXX-XXXX	REV: 2
Date: February 17, 1996	Sheet 3 of 7	



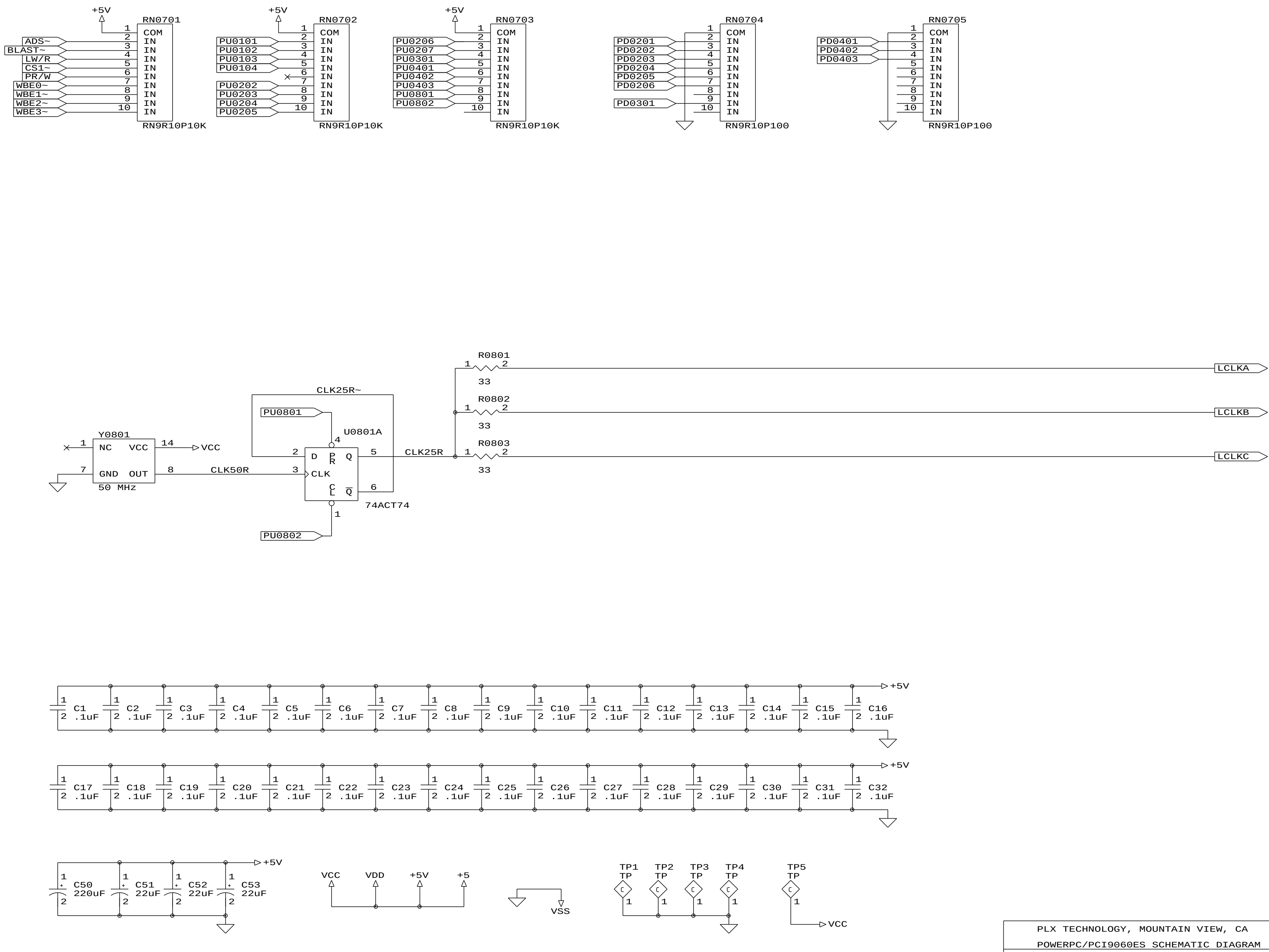
PLX TECHNOLOGY, MOUNTAIN VIEW, CA		
POWERPC/PCI9060ES SCHEMATIC DIAGRAM		
Title	POWERPC 403GC CPU	
Size	Document Number	REV
B	XXX-XXXX-XXXX	2
Date: February 17, 1996	Sheet 4 of 7	



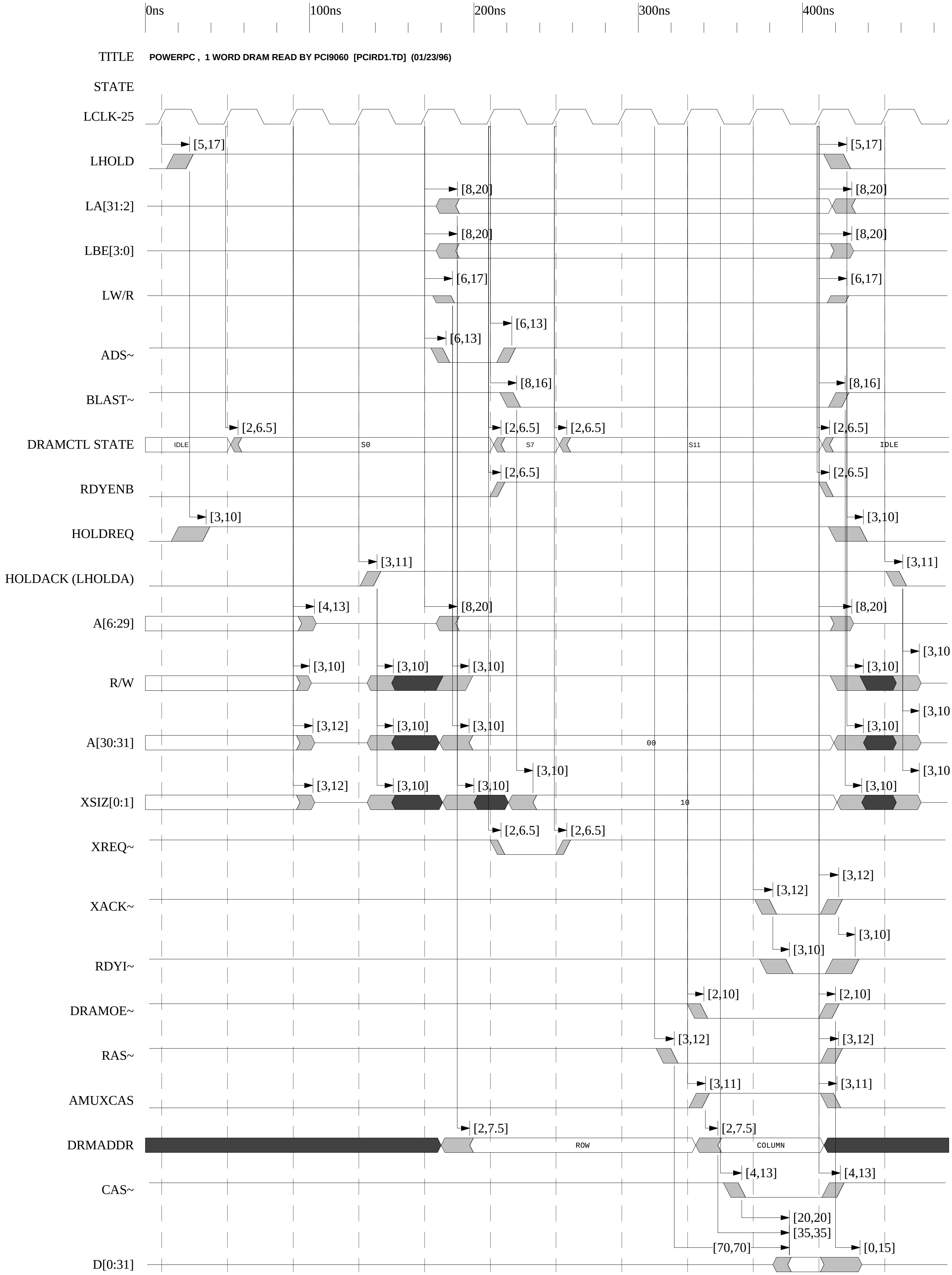
PLX TECHNOLOGY, MOUNTAIN VIEW, CA POWERPC/PCI9060ES SCHEMATIC DIAGRAM		
Title 1Mx32 DRAM		
Size B	Document Number XXX-XXXX-XXXX	REV 2
Date: February 17, 1996	Sheet 5 of 7	



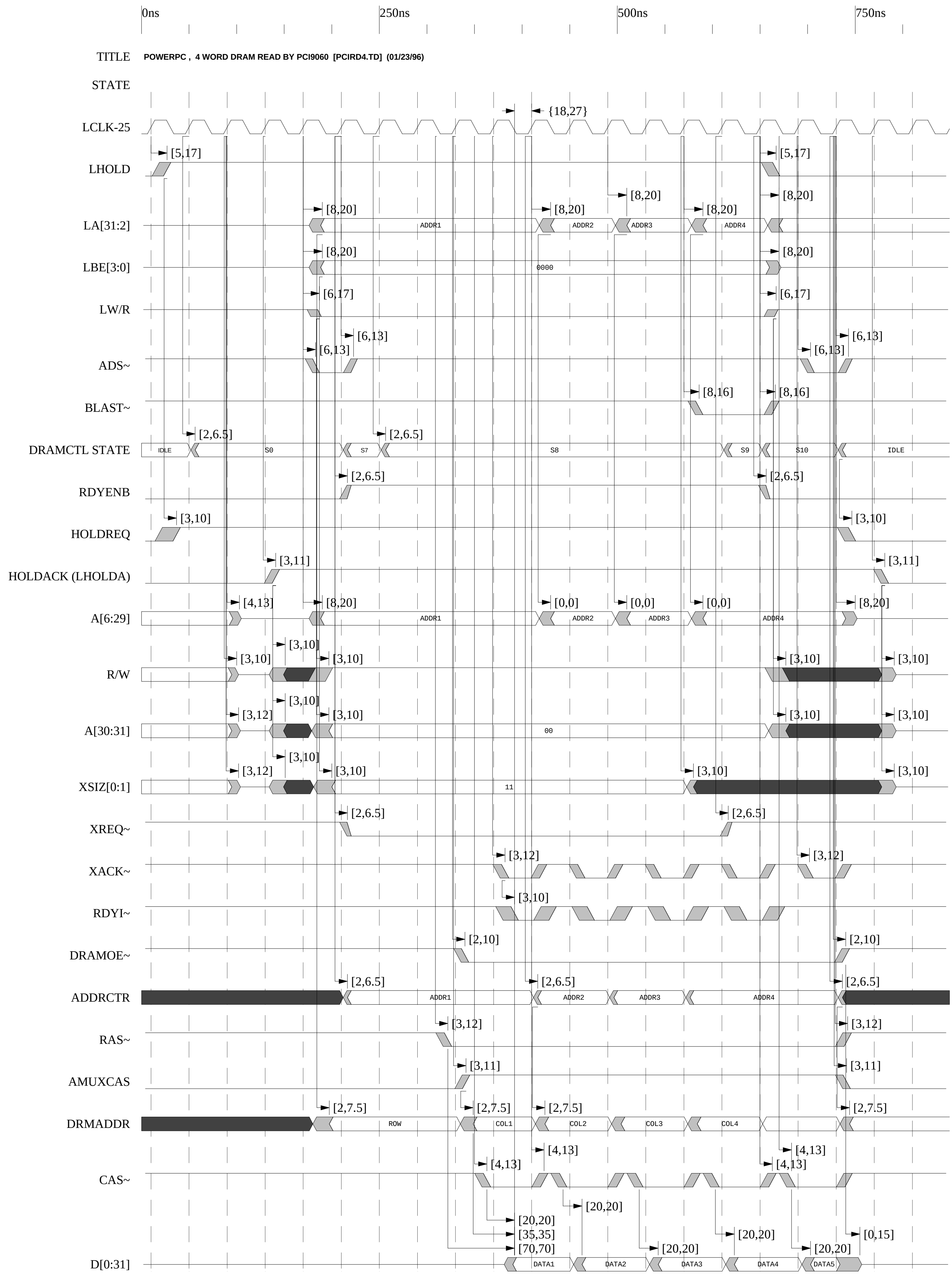
PLX TECHNOLOGY, MOUNTAIN VIEW, CA		
POWERPC/PCI9060ES SCHEMATIC DIAGRAM		
Title: PCI CONNECTOR		
Size: B	Document Number: XXX-XXXX-XXXX	REV: 2
Date: February 17, 1996	Sheet 6 of 7	

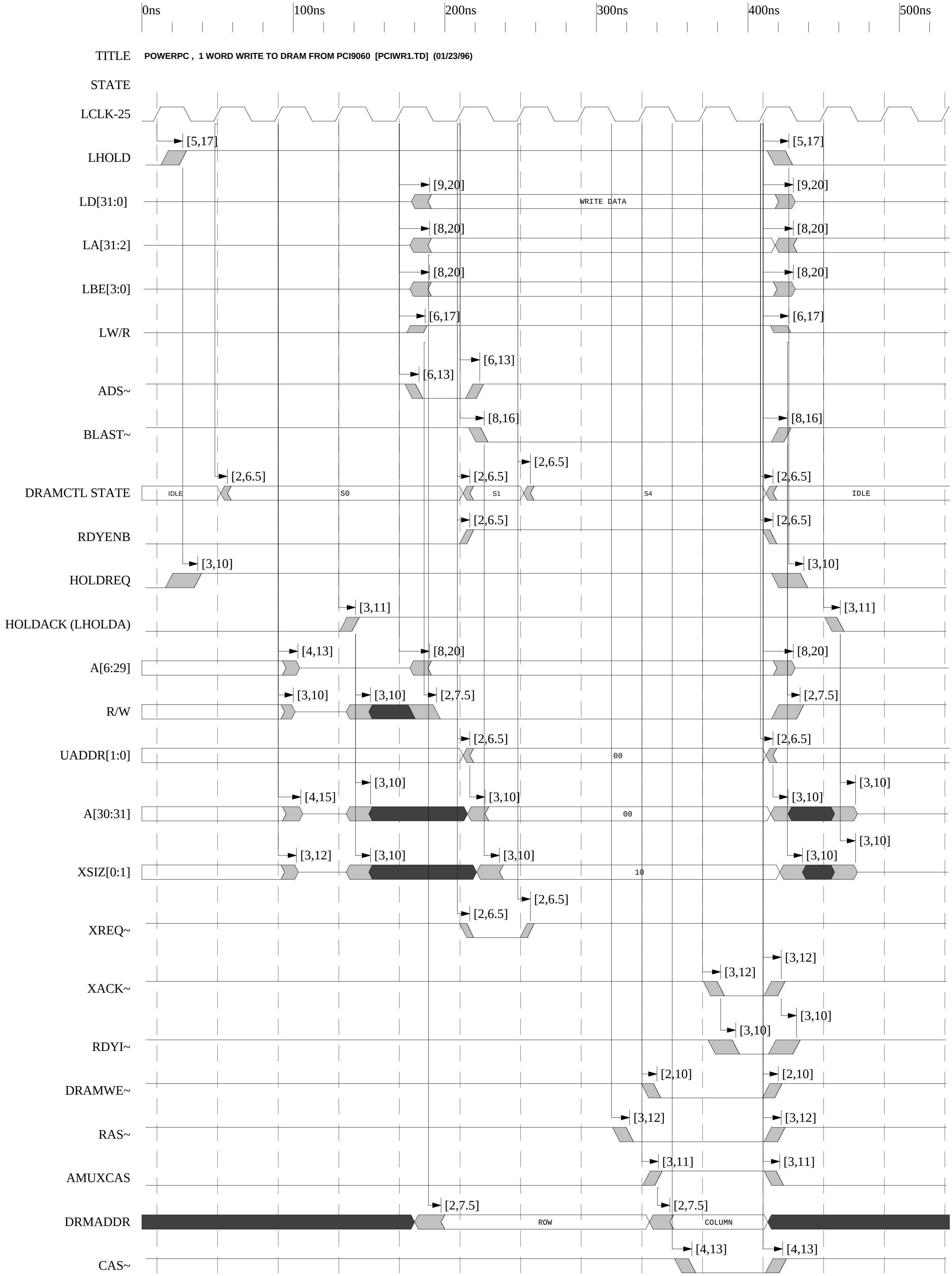


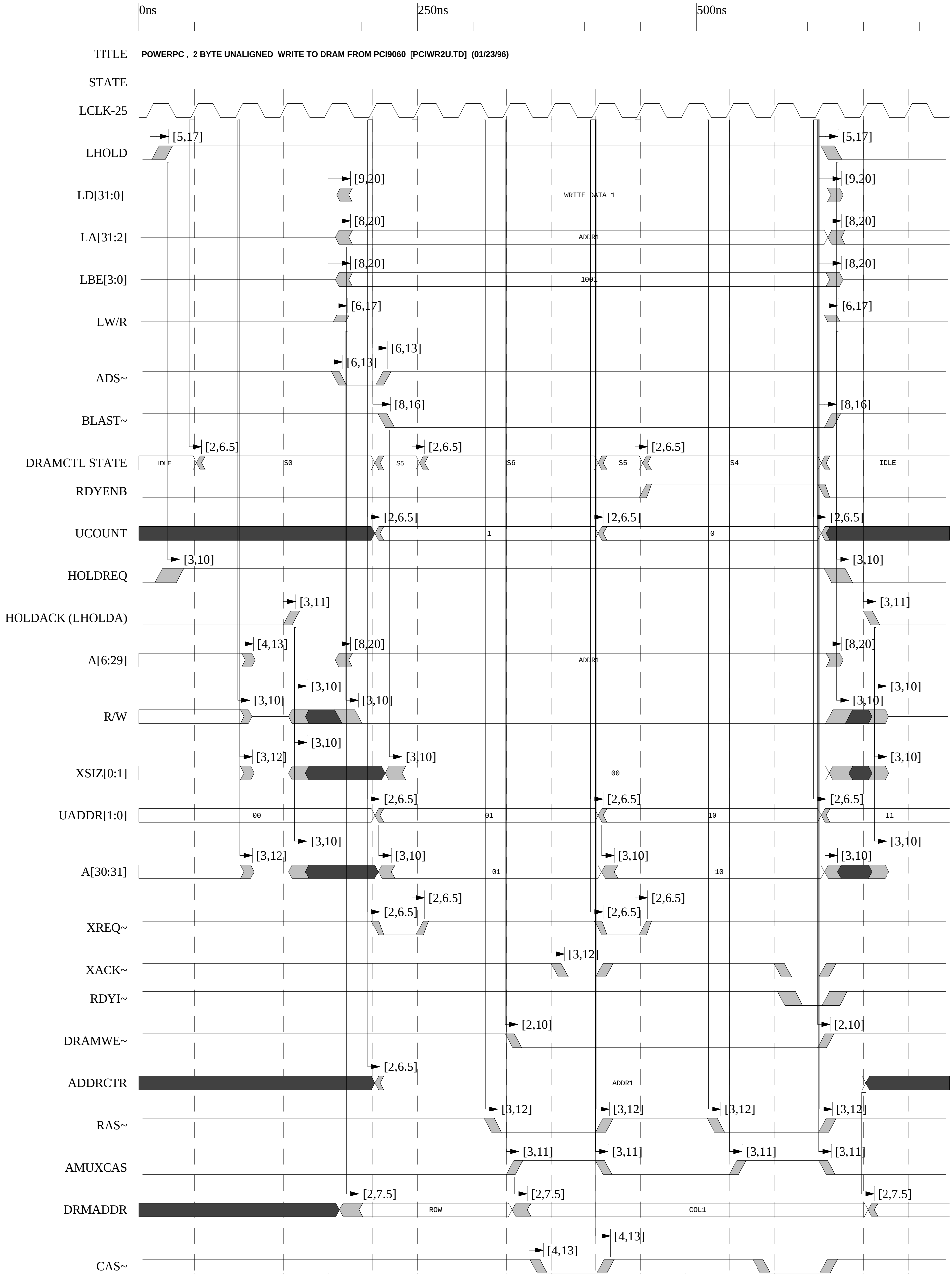
PLX TECHNOLOGY, MOUNTAIN VIEW, CA			
POWERPC/PCI9060ES SCHEMATIC DIAGRAM			
Title	RESISTORS, CAPACITORS		
Size	Document Number		REV
B	XXX-XXXX-XXXX		2
Date:	February 17, 1996	Sheet 7 of 7	



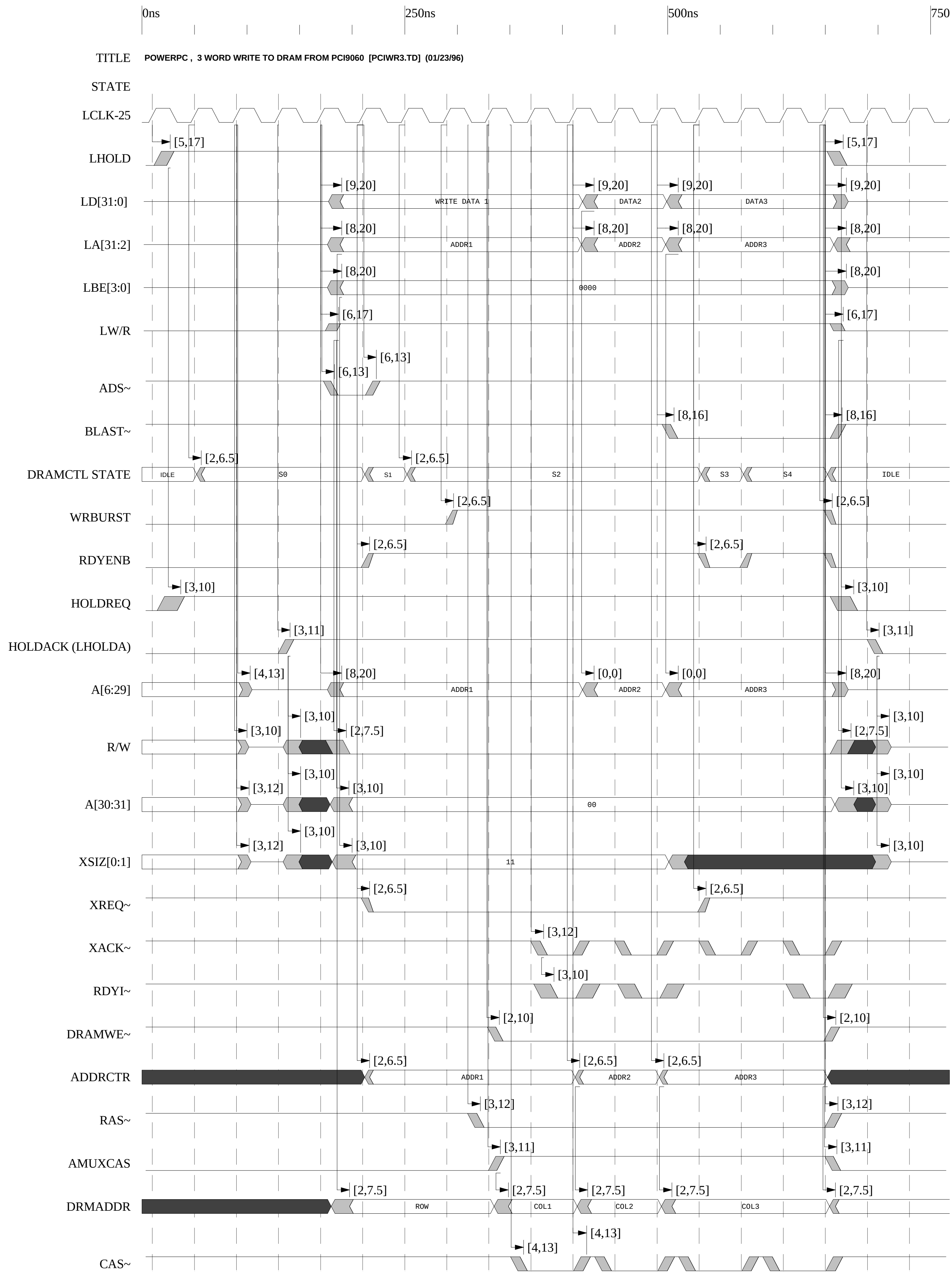
0ns
100ns
200ns
300ns
400ns
TITLE
POWERPC , 1 WORD DRAM READ BY PCI9060 [PCIRD1.TD] (01/23/96)
STATE
LCLK-25
LHOLD
LA[31:2]
LBE[3:0]
LW/R
ADS~
BLAST~
DRAMCTL STATE
IDLE
S0
S7
S11
RDYENB
HOLDREQ
HOLDACK (LHOLDA)
A[6:29]
R/W
A[30:31]
00
XSIZ[0:1]
10
XREQ~
XACK~
RDYI~
DRAMOE~
RAS~
AMUXCAS
DRMADDR
ROW
COLUMN
CAS~
D[0:31]
[5,17]
[8,20]
[6,17]
[6,13]
[8,16]
[2,6.5]
[3,10]
[3,11]
[4,13]
[3,12]
[2,10]
[2,7.5]
[20,20]
[35,35]
[70,70]
[0,15]


0ns
250ns
500ns
750ns
TITLE
POWERPC , 4 WORD DRAM READ BY PCI9060 [PCIRD4.TD] (01/23/96)
STATE
LCLK-25
LHOLD
LA[31:2]
LBE[3:0]
LW/R
ADS~
BLAST~
DRAMCTL STATE
RDYENB
HOLDREQ
HOLDACK (LHOLDA)
A[6:29]
R/W
A[30:31]
XSIZ[0:1]
XREQ~
XACK~
RDYI~
DRAMOE~
ADDRCTR
RAS~
AMUXCAS
DRMADDR
CAS~
D[0:31]
{18,27}
[5,17]
[8,20]
[6,17]
[6,13]
[8,16]
[2,6.5]
[3,10]
[3,11]
[4,13]
[0,0]
[3,12]
[2,10]
[2,7.5]
[20,20]
[35,35]
[70,70]
[0,15]
IDLE
S0
S7
S8
S9
S10
ADDR1
ADDR2
ADDR3
ADDR4
0000
00
11
ROW
COL1
COL2
COL3
COL4
DATA1
DATA2
DATA3
DATA4
DATA5

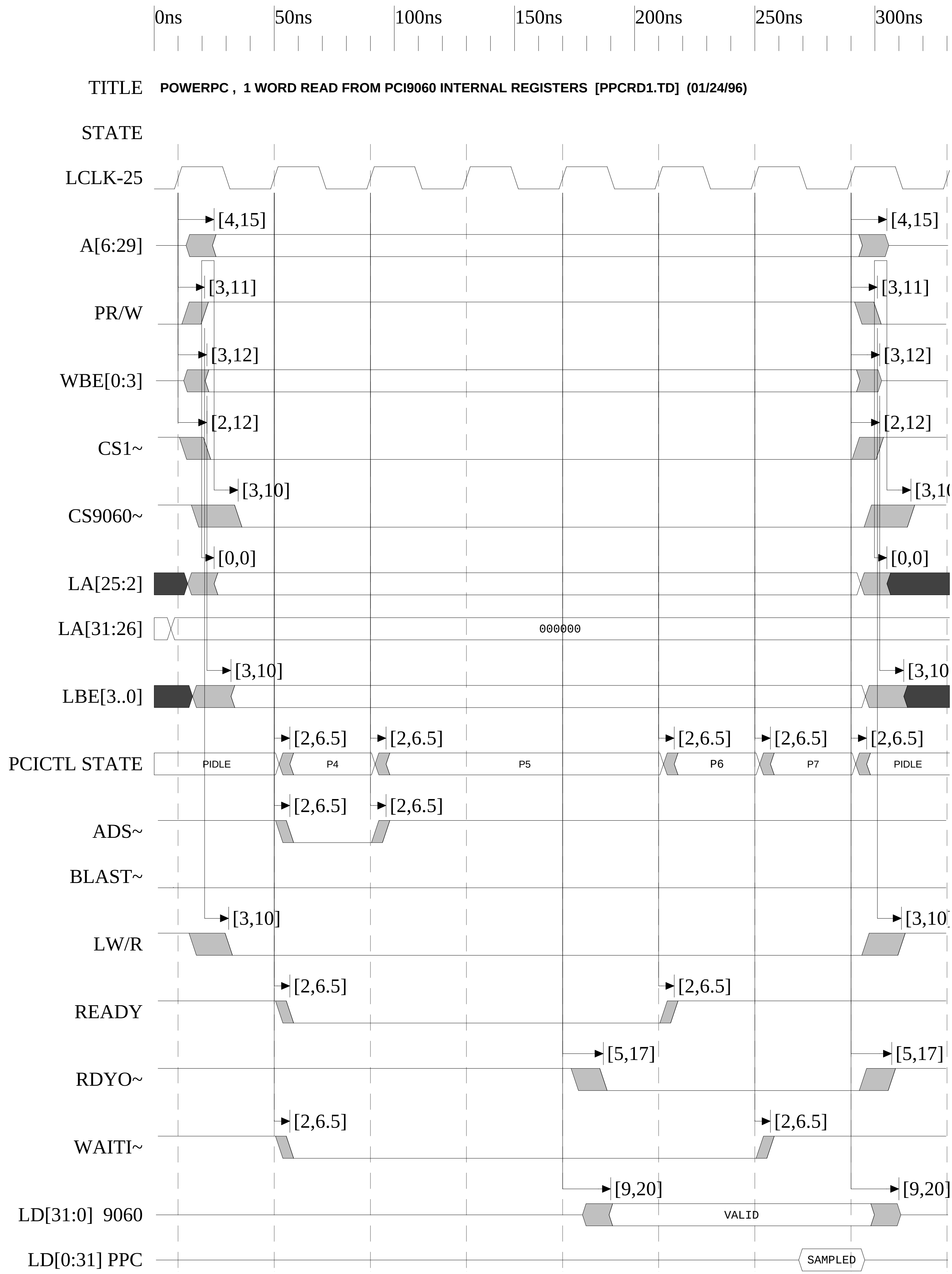

POWERPC , 1 WORD WRITE TO DRAM FROM PCI9060 [PCIWR1.TD] (01/23/96)
0ns
100ns
200ns
300ns
400ns
500ns
TITLE
STATE
LCLK-25
LHOLD
LD[31:0]
LA[31:2]
LBE[3:0]
LW/R
ADS~
BLAST~
DRAMCTL STATE
RDYENB
HOLDREQ
HOLDACK (LHOLDA)
A[6:29]
R/W
UADDR[1:0]
A[30:31]
XSIZ[0:1]
XREQ~
XACK~
RDYI~
DRAMWE~
RAS~
AMUXCAS
DRMADDR
CAS~
WRITE DATA
IDLE
S0
S1
S4
00
10
ROW
COLUMN



0ns
250ns
500ns
TITLE
POWERPC , 2 BYTE UNALIGNED WRITE TO DRAM FROM PCI9060 [PCIWR2U.TD] (01/23/96)
STATE
LCLK-25
LHOLD
LD[31:0]
WRITE DATA 1
LA[31:2]
ADDR1
LBE[3:0]
1001
LW/R
ADS~
BLAST~
DRAMCTL STATE
IDLE
S0
S5
S6
S5
S4
IDLE
RDYENB
UCOUNT
1
0
HOLDREQ
HOLDACK (LHOLDA)
A[6:29]
ADDR1
R/W
XSIZ[0:1]
00
UADDR[1:0]
00
01
10
11
A[30:31]
01
10
XREQ~
XACK~
RDYI~
DRAMWE~
ADDRCTR
ADDR1
RAS~
AMUXCAS
DRMADDR
ROW
COL1
CAS~



0ns
250ns
500ns
750
TITLE
POWERPC , 3 WORD WRITE TO DRAM FROM PCI9060 [PCIWR3.TD] (01/23/96)
STATE
LCLK-25
LHOLD
LD[31:0]
WRITE DATA 1
DATA2
DATA3
LA[31:2]
ADDR1
ADDR2
ADDR3
LBE[3:0]
0000
LW/R
ADS~
BLAST~
DRAMCTL STATE
IDLE
S0
S1
S2
S3
S4
WRBURST
RDYENB
HOLDREQ
HOLDACK (LHOLDA)
A[6:29]
R/W
A[30:31]
00
XSIZ[0:1]
11
XREQ~
XACK~
RDYI~
DRAMWE~
ADDRCTR
RAS~
AMUXCAS
DRMADDR
ROW
COL1
COL2
COL3
CAS~
[5,17]
[9,20]
[8,20]
[6,17]
[6,13]
[8,16]
[2,6.5]
[3,10]
[3,11]
[4,13]
[0,0]
[2,7.5]
[3,12]
[2,10]


0ns
50ns
100ns
150ns
200ns
250ns
300ns
TITLE
POWERPC , 1 WORD READ FROM PCI9060 INTERNAL REGISTERS [PPCRD1.TD] (01/24/96)
STATE
LCLK-25
A[6:29]
PR/W
WBE[0:3]
CS1~
CS9060~
LA[25:2]
LA[31:26]
000000
LBE[3..0]
PCICTL STATE
PIDLE
P4
P5
P6
P7
ADS~
BLAST~
LW/R
READY
RDYO~
WAITI~
LD[31:0] 9060
VALID
LD[0:31] PPC
SAMPLED
[4,15]
[3,11]
[3,12]
[2,12]
[3,10]
[0,0]
[2,6.5]
[5,17]
[9,20]