MiniRISC® LR4500 Superscalar Microprocessor

Technical Manual

Order Number C14043

Document DB14-000066-00, First Edition (May 1998)
This document describes revision A of LSI Logic Corporation's MiniRISC® LR4500 Superscalar Microprocessor and will remain the official reference source for all revisions of this product until rescinded by an update.

To receive product literature, call us at 1.800.574.4286 (U.S. and Canada); +32.11.300.531 (Europe); 408.433.7700 (outside U.S., Canada, and Europe) and ask for Department JDS; or visit us at http://www.lsilogic.com.

Contents

Customer Feedback

Figures

Tables

Preface

This book is the primary reference and technical manual for the MiniRISC® LR4500 Superscalar Microprocessor reference device. The book contains a complete functional description of the LR4500, with physical and electrical specifications.

Audience

This book assumes that you are familiar with microprocessors and related support devices. The book targets:

- Engineers and managers who are evaluating the LR4500 for possible use in system design
- Engineers who are designing the LR4500 into a system

Organization

This book has the following chapters:

- Chapter 1, **Introduction,** provides an overview of the reference device, defines the device's context on an evaluation board, and lists the device's features.
- Chapter 2, **Functional Blocks**, provides information about all functional blocks that are part of the LR4500: the CW4011 shell, the synchronous DRAM Controller (DRAMC), the SCbus/Lbus Controller (SCLC), the PLL clock circuit, and the ICEport UART. The chapter also describes the LR4500 pipeline architecture.
- Chapter 3, **Programming Model**, provides information about the LR4500 programming model, including a list of LR4500 registers, information about memory mapping, and descriptions of the LR4500 registers used to configure the system.

- Chapter 4, **Instruction Set**, lists and describes the instructions that make up the LR4500 instruction set, defines the instruction set extensions, and describes CPU instruction opcode bit encoding.
- Chapter 5, **Bus Interface Descriptions**, describes the interface signals for the major LR4500 interfaces including the signals that provide the external interface between the LR4500 and external devices, and the interface signals internal to the LR4500. The chapter also describes the buffering required for certain interface signals.
- Chapter 6, **DRAM Controller and Memory Bus**, describes the synchronous DRAM Controller and the memory bus. It also provides timing information for different DRAM transactions.
- Chapter 7, **SCbus and Local I/O Bus Converter Module**, describes the Lbus and explains how the LR4500 interacts with the Lbus through the SCLC module.
- Chapter 8, **Cache Configuration and Maintenance**, describes the I-cache and D-cache configurations and explains how to maintain the caches after power is turned on.
- Chapter 9, **ICEport**, describes the ICEport building block that provides a full-duplex serial UART (universal asynchronous receive and transmit) port for the LR4500.
- Chapter 10, **Organization of Clock and Exception Signals**, describes the organization of the LR4500 clock signals and the exception-handling signals.
- Chapter 11, **Specifications**, defines the electrical characteristics of the LR4500. It also provides packaging information, including the mechanical layout of the LR4500, the chip's dimensions, and pin locations.

Related Publications

MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual, Order Number C14040

MiniRISC BDMR4011 Evaluation Board User's Guide, Order Number C14052.

Conventions Used in This Manual

"Assert" means to drive a signal true or active; "deassert" means to drive a signal false or inactive.

Hexadecimal numbers are indicated by the prefix "0x," for example, 0x32CF.

The following notational conventions are used throughout this manual.

Notation	Example	Meaning and Use
courier typeface	`.nwk` file	Names of commands, files, symbols, parts, directories, modules, and macrocells are shown in courier typeface.
bold typeface	**fd1sp**	In a command line, keywords are shown in bold, nonitalic typeface. Enter them exactly as shown.

Chapter 1
Introduction

This chapter provides an overview and defines the features of the MiniRISC LR4500 Superscalar Microprocessor reference device.

1.1 LR4500 Overview

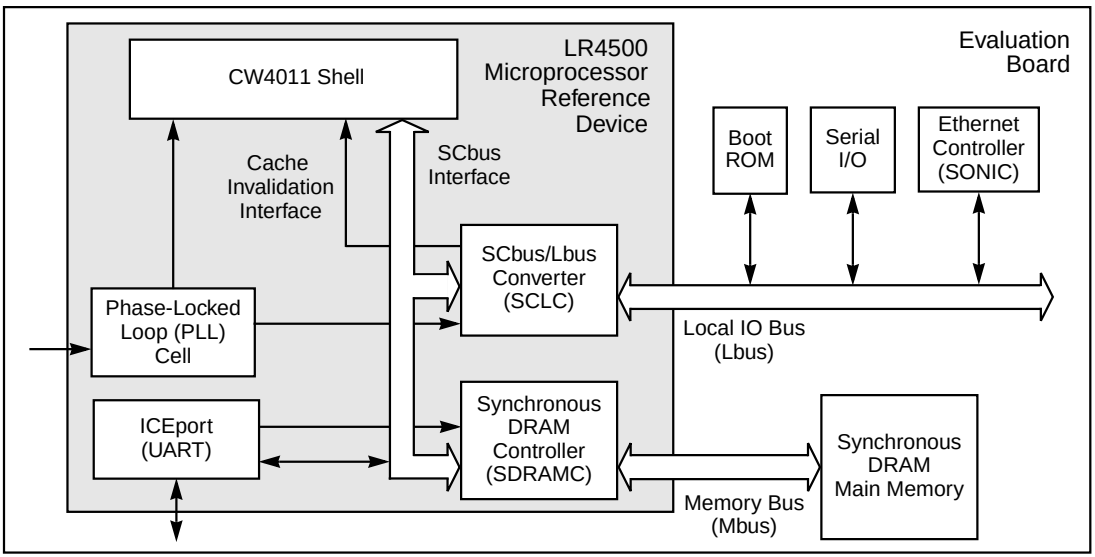
The LR4500 is a chip implementation of the MiniRISC CW4011 microprocessor core and shell. It is the second LSI Logic implementation of a 32-bit MIPS II compatible, superscalar CPU. As shown in Figure 1.1, the LR4500 contains the following circuitry, housed on an evaluation board:

- The maximum configuration CW4011 shell, which is an unencrypted Verilog model that contains
  - CW4011 core
  - Multiply/Divide unit (MDU)
  - Instruction cache (I-cache)
  - Data cache (D-cache)
  - Memory Management Unit (MMU) without Translation Look-aside Buffer (TLB)
  - Write Back Buffer (WB).

  You can configure certain modules in the shell by programming the Configuration Register in CW4011 Coprocessor 0 (CP0). For example, you can turn off the MDU and the WB. Depending on the application, the LR4500 generally uses all modules.

- The DRAM controller that provides the Memory Bus (Mbus) interface between the LR4500 and external synchronous memory devices.
- The SCbus/Lbus Converter that controls the Local I/O bus (Lbus) and external Lbus devices
- A Phase-Locked Loop (PLL) circuit that supplies clock inputs to the other modules in the LR4500
- The ICEport UART (universal asynchronous receiver/transmitter) is used to download SerialICE™ application software and to debug the LR4500.

Figure 1.1 Block Diagram of LR4500 and Evaluation Board Circuitry



Chapter 2, "Functional Blocks," provides detailed information about the functional elements of the LR4500.

The LR4500 is housed on an evaluation board (BDLR4500) that allows you to use and test the LR4500. In addition to the LR4500, the board contains:

- The DRAM array that communicates with the LR4500 through the Mbus.

The Lbus that allows you to plug in devices such as a Boot-ROM, serial I/O devices, and an external Ethernet controller. (The Lbus is a simple, generic interface for peripheral devices such as ROMs, RAMs, UARTs. It has a demultiplexed 32-bit address bus and 32-bit data bus, and it is similar to the 486 VLbus.)

1.2 LR4500 Features

The LR4500 Microprocessor has the following features:

- System clock operating at up to 100 MHz, with 150 Dhrystone MIPS performance
- Superscalar microprocessor support for the MIPS II 32-bit instruction set:
  - Up to two instructions executed per clock cycle
  - Four-deep write buffer
  - Load scheduling
  - R3000/R4000 compatible mode for the Exception Return and Status Register
- 32-bit timer (R4000 compatible)
- SCbus watchdog timer with error reporting features
- Full internal SCAN testing
- SerialICE debugging support provided through the ICEport UART interface
- SCbus/Lbus converter to control the Lbus and external Lbus devices
- Easily implemented interface to the SONIC Ethernet controller
- Synchronous DRAM Controller, with 64-bit wide data transfer, interfaces to the following 16-Mbit SDRAMs (synchronous DRAMs):
  - 1-M x 16-bit SDRAM devices in an 8-Mbyte or 16-Mbyte configuration
  - 2-M x 8-bit SDRAM devices in a 16-Mbyte or 32-Mbyte configuration
  - 4-M x 4-bit SDRAM devices in a 32-Mbyte or 64-Mbyte configuration

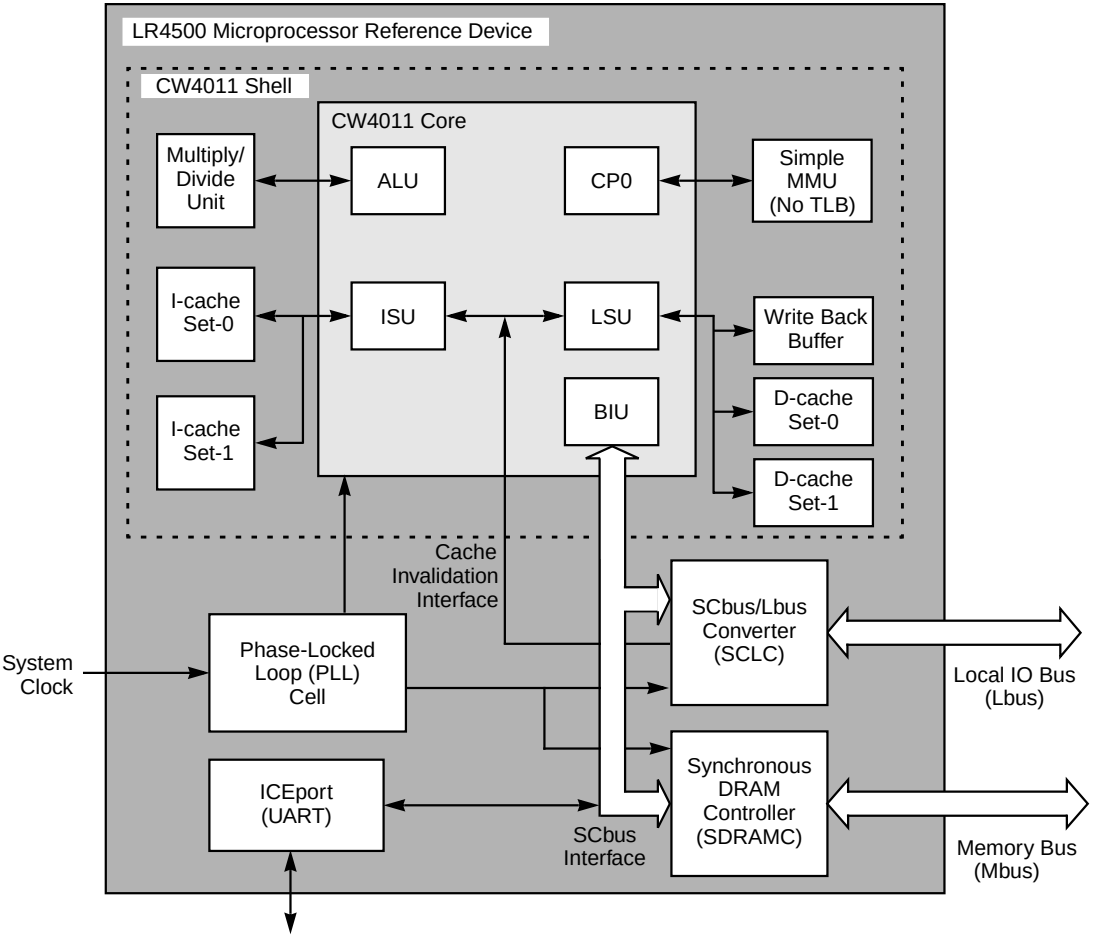
- PLL circuit for internal system clock; synchronizes internal system clock with an external clock
- 3.3 V operation
- LR4500 power - 4.19 mA (at 3.46 V and 100 MHz)
- Packaged in a 256-pin PQFPt (Plastic Quad Flat Package)
- Cache configuration:
  - Direct-mapped or two-way set-associative I-cache and D-cache
  - 1-Kbyte, 2-Kbyte, 4-Kbyte, or 8-Kbyte cache sets, organized as either direct-mapped (single set) cache with maximum cache size of 8 Kbytes, or as a two-way set-associative cache with a maximum cache size of 16 Kbytes.
- Simplified kseg0 and kseg1 Memory Management Unit without TLB
- Fast multiplier supporting multiply-accumulate operations
- High-performance multiplier delivers three-cycle latency and one cycle throughput for MAC (multiply with accumulate) instructions
- Support for both big-endian and little-endian formats

Chapter 2
Functional Blocks

This chapter describes each of the LR4500 functional blocks and the LR4500 pipeline architecture.

The chapter is divided into the following sections:

Figure 2.1 LR4500 Block Diagram



2.1 CW4011 Shell

The CW4011 shell consists of the CW4011 core, the MDU (Multiply/Divide Unit), the MMU (Memory Management Unit), the WB (Write Back Buffer), the I-cache (instruction cache), and the D-cache (data cache).

With the exception of the CW4011 core, you can turn off certain modules in the CW4011 shell, for example the MDU or the WB, to fit your own ASIC design.

2.1.1 CW4011 Core

The CW4011 core is an encrypted Verilog RTL model that is part of LSI Logic's CoreWare® Library. The CW4011 core is a predefined hardmacro that contains the following basic microprocessor elements:

- Instruction Scheduling Unit (ISU)
- Load/Store Unit (LSU)
- Arithmetic Logic Unit (ALU)
- Coprocessor 0 (CP0)
- Bus Interface Unit (BIU)

The CW4011 core executes all MIPS II 32-bit based instructions except for multiply/divide instructions. These instructions are handled by the MDU, which is part of the CW4011 shell.

For detailed information about the CW4011 core, refer to the *MiniRISC CW4011 Superscalar Microprocessor Technical Manual*.

2.1.2 Multiply/Divide Unit

The multiply/divide unit supports multiply-add/subtract operations as well as multiply and divide. The multiply instruction executes in three cycles. The multiply-add/subtract instruction is optimized to one cycle.

2.1.3 Memory Management Unit Shell

The MMU does not have a TLB. *kseg0* and *kseg1* are mapped to the first 512 Mbyte space, which is the bottom of the memory space. The *kuseg* and *kseg2* blocks are directly mapped to the physical address space without any change. "Section 3.2, on page 3-4, provides more information on this subject.

2.1.4 Write Back Buffer

The CW4011 core uses this buffer when the D-cache operates in write back mode. When a cache miss occurs and the victim entry contains a dirty line, the dirty data is written into the Write Back Buffer instead of the main memory. This reduces the latency of the cache refill for missed addresses. Data in the Write Back Buffer is written into the main memory after the refill is completed.

2.1.5 Caches

The LR4500 has separate instruction and data caches—I-cache and D-cache—that can be organized as direct-mapped or two-way set-associative caches. The cache controllers support configurations of 1, 2, 4, or 8 Kbytes for each set. Thus, the smallest supported configuration is a 1-Kbyte direct-mapped cache, and the largest is a 16-Kbyte two-way set-associative cache, with 8 Kbytes per set. You can select between Write Back and Write Through modes. You can also configure both sets of the D-cache and one set of the I-cache for scratchpad RAM mode. Refer to Chapter 8, "Cache Configuration and Maintenance," for more information on this subject.

2.2 Synchronous DRAM Controller

The DRAM Controller is part of the LR4500 reference device, and it is external to the CW4011 shell, as shown in Figure 2.1 on page 2-2. It generates DRAM transactions in response to requests from the CW4011 core or from the SCLC module. The DRAM Controller also generates initialization cycles and refresh cycles for DRAM. Chapter 6, "DRAM Controller and Memory Bus," provides detailed information about DRAM and the DRAM controller.

2.3 SCbus to Local I/O Bus Converter

The SCLC module provides an interface between the internal CW4011 microprocessor bus (SCbus) and the external Local I/O bus (Lbus). The Lbus connects such devices as boot-ROM, serial I/O devices, and the Ethernet Controller to the LR4500.

The SCbus is a 32-bit address, 64-bit data bus. The Lbus, which is a subset of the industrial standard VLbus, is a 32-bit address, 32-bit data bus. The CW4011 uses the SCLC module to access devices on the Lbus. Devices on the Lbus access the DRAM main memory through the SCLC module and the DRAM Controller.

The CW4011 microprocessor generally has ownership of the SCbus and the Lbus. When a device on the Lbus wants to access the DRAM, it asserts the bus hold request signal on the Lbus. The SCLC module

detects the asserted signal and then asserts the bus hold request to the CW4011. The CW4011 asserts the grant signal to the SCLC module, and the SCLC module then asserts the hold acknowledge signal to the Lbus device. 'SCbus and Local I/O Bus Converter Module" provides detailed information about the Lbus.

2.4 PLL Clock Circuit

The PLL circuit is an LSI Logic PLL cell (`pllpgmcb`) part that drives the clock signals to the CW4011 shell and the other modules that are part of the LR4500. The system clock, SCLKp, drives the PLLREFp input.

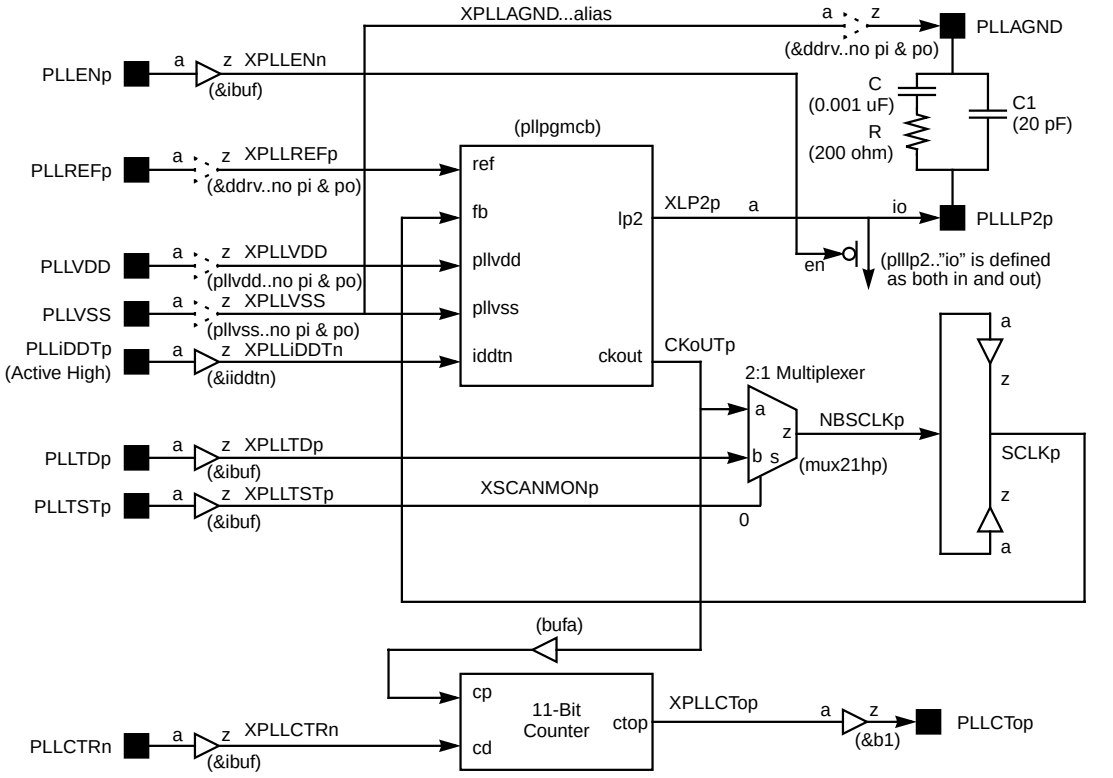
Figure 2.1 on page 2-2 shows the relationship between the PLL circuit and the other LR4500 modules. Figure 2.2 shows the layout of the LR4500 PLL circuit.

When using the PLL circuit, you must observe the following design requirements:

- Provide capacitance devices.
- Provide a resistor between PLLLP2p and PLLAGND.
- Connect other PLL circuit inputs to V_{DD} or GND.
- Leave PLLCTop outputs open.

For more information about the PLL circuit, refer to the LSI Logic *G10®-p Cell-Based ASIC Products Design Manual* and *G10-p Cell-Based ASIC Products Databook*.

Figure 2.2 LR4500 PLL Circuit Diagram



2.5 ICEport UART

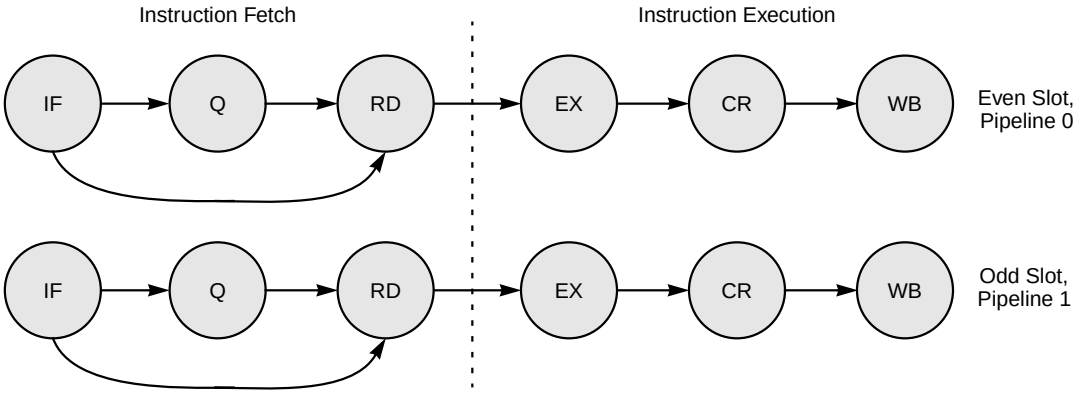
The ICEport is a full-duplex serial UART port. It is used for downloading application software and debugging the LR4500. The ICEport works with an ICE controller at baud rates up to 1 MBaud, and it is integrated with the SCLC and DRAM controller modules on the SCbus. 'ICEport' 'ICEport' provides detailed information about the ICEport UART.

2.6 Pipeline Architecture

The LR4500 has two identical concurrent five-stage pipelines that are part of the CW4011 core. These pipelines provide the LR4500 with its superscalar capabilities. As shown in Figure 2.3, there is an even pipeline and an odd pipeline.

Each of the five pipeline stages can be viewed as a pair of instruction "slots" (one slot for each pipeline.) So an instruction in the even pipeline at the EX stage may be referred to as 'the even slot of the EX stage.' In addition to the five basic pipeline stages, each pipeline also has a conditional queuing stage (Q).

Figure 2.3 LR4500 Instruction Pipeline



1. Branch instruction encountered.
2. Q state bypassed.

The first two pipeline stages and the queuing stage are used during instruction fetch, and the last three stages are used during instruction execution. Once a stage has accepted an instruction from the previous stage it must hold the instruction for re-execution in case the pipeline stalls. The pipeline stages perform the following functions:

- IF (Instruction Fetch). The LR4500 fetches the instruction during the first stage.
- Q (Queuing). This conditional queueing stage boosts branch instructions. Depending on the branches and register conflicts, instructions may either enter this stage or be advanced straight to the RD stage.

- RD (Read). During this stage, any required operands are read from the Register File while the instruction is being decoded.
- EX (Execution). This stage performs a number of functions: all instructions are executed, conditional branches are resolved, the address calculation for load and store instructions is performed.
- CR (Cache Read). This stage is used to access the cache for load and store instructions. Data is returned to the register bypass logic at the end of this stage.
- WB (Write Back). Results are written into the Register File during this stage.

For a more detailed description of the CW4011 pipeline, refer to the *MiniRISC Superscalar Microprocessor Core Technical Manual*.

Chapter 3
Programming Model

This chapter provides information about the LR4500 Microprocessor programming model. The term ‘programming model’ refers to the way in which data is arranged in registers and in memory.

The chapter

- Provides a list of LR4500 registers on page 3-2
- Describes LR4500 memory mapping on page 3-4
- Describes how to configure the system using LR4500 registers on page 3-5

In addition, the following sections in other chapters of this manual provide supplementary information related to the programming model:

- “DRAM Controller and Memory Bus” on page 6-1
- “SCbus Timeout Watchdog Timer” on page 7-8
- “Cache Configuration and Maintenance” on page 8-1

The *MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual* describes the Memory Management Unit and Coprocessor 0 (CP0).

3.1 Register Set

Table 3.1 lists the LR4500 registers and provides the physical and virtual addresses, and the register numbers. The registers are listed in functional blocks and arranged alphabetically within each functional block.

Table 3.1 LR4500 Registers

Register Name	Physical Address	Virtual Address	Number
CP0 Exception Processing Registers			
BadVAddr (Bad Virtual Address)	Physical and virtual addresses are not applicable to CP0 exception processing registers.		8
BDA (Breakpoint Data Address)			19
BDAM (Breakpoint Data Address Mask)			21
BPC (Breakpoint Program Counter)			18
BPCM (Breakpoint PC Mask)			20
Cause			13
CCC (Configuration and Cache Control)[1]			16
Compare			11
Count			9
DCS (Debug Control Status)			7
EPC (Exception Program Counter)			14
ErrorPC			30
LLAdr (Load Linked Address)			17
PRId (Processor Revision Identifier)			15
Status			12

Table 3.1 LR4500 Registers (Cont.)

Register Name	Physical Address	Virtual Address	Number
Lbus Controller Registers			
External Vectored Interrupt[2]	0x 1010 0008	0x B010 0008	N/A
SCbus Error Address[3]	0x 1010 0000	0x B010 0000	N/A
SCbus Error Status[4]	0x 1010 0004	0x B010 0004	N/A
DRAM Controller Registers			
DRAM Refresh Register[5]	0x 1000 0004	0x B000 0004	N/A
DRAM Controller Configuration[6]	0x 1000 0000	0x B000 0000	N/A
ICEport Registers			
Rx Status[7]	0x10FF 0000[8]	0xB0FF 0000[8]	N/A
Rx Setup[7]	0x10FF 0000[8]	0xB0FF 0000[8]	N/A
Rx Data	0x10FF 0004	0xB0FF 0004	N/A
Tx Status	0x10FF 0008	0xB0FF 0008	N/A
Tx Data	0x10FF 000C	0xB0FF 000C	N/A

1. See “CCC Register” on page 3-5
2. See “External Vectored Interrupt Register” on page 3-11
3. See “SCbus Error Address Register” on page 3-10
4. See “SCbus Error Status Register” on page 3-10
5. See “DRAM Refresh” on page 6-12
6. See “DRAM Controller Configuration Register” on page 6-5
7. See “ICEport Registers” on page 9-9
   All other registers listed in this table are described in the *MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual*
8. The physical address for the Rx Status Register is the same as the physical address for the Rx Setup Register. Similarly, the virtual addresses are the same. The Rx Status Register is a read register and the Rx Setup Register is a write register. This means that when the addresses are accessed, the register accessed depends on the condition of the read/write signal.

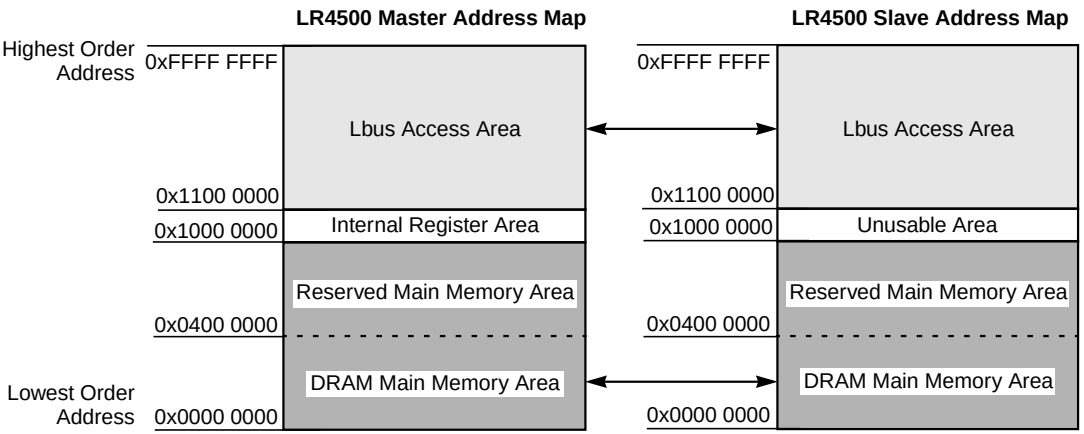
3.2 Memory Mapping

Figure 3.1 shows the physical memory map of the LR4500 reference device, where the LR4500 is master of the Lbus and an Lbus device is slave, and the physical memory map where an Lbus device is the Lbus master and the LR4500 is slave. In both cases, address spaces are linear 4-Gbyte spaces. Lbus master devices cannot access LR4500 internal memory-mapped registers.

Synchronous DRAM main memory that is interfaced to the LR4500 is located at address space 0x0000 0000 through 0x03FF FFFF. The LR4500 works as an Lbus slave device for this 64-Mbyte memory space. There is no guarantee that memory devices exist in the entire 64-Mbyte area. Software, in the form of a setup/bootstrap utility or equivalent must check installed memory size when the system is initialized. The upper 192-Mbyte space is reserved as an extended main memory area.

LR4500 internal registers for DRAM Controller and error reporting are located in the Internal Registers area between addresses 0x1000 0000 and 0x10FF FFFF. These registers must be accessed through *kseg1*, the uncached unmapped area. The virtual address for these registers is 0xB000 0000 through 0xB0FF FFFF.

Figure 3.1 LR4500 Master/Slave Memory Map



3.3 System Configuration

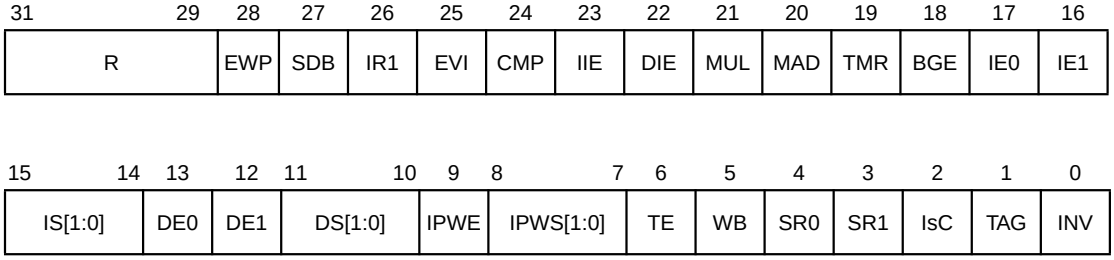
LR4500 has a number of features that allow you to modify the system configuration. This section describes the Configuration and Cache Control (CCC) Register, which is part of the CW4011 core, and several Lbus registers, which are part of the SCLC module. You can also configure the DRAM, as described in Chapter 6, “DRAM Controller and Memory Bus.”

3.3.1 CCC Register

The Configuration and Cache Control (CCC) Register is part of CP0, the system coprocessor. The CCC Register allows you to use software to configure various pieces of the core design, such as the Bus Interface Unit (BIU) and the controllers for the I-cache and D-cache.

You can read from the CCC Register using the MFC0 instruction, and write to it using the MTC0 instruction. Table 4.14 on page 4-14 describes these instructions. The register’s address in CP0 is ‘16.’ Figure 3.2 shows the bit configuration of the CCC Register. All bits are initialized to 0 at reset, so that the caches are not available until the register is programmed.

Figure 3.2 CCC Register



R **Reserved** **[31:29]**
This field is reserved. The bits are cleared to 0.

EWP **External Write Priority** **28**
This bit defines the arbitration priority on the SCbus between a data read and data write transaction in the

4-deep write buffer. Clearing the bit to 0 gives a higher priority to a data read request if the read address does not match the write address in the write buffer. Setting the bit to 1 gives higher priority to data write transactions.

SDB **Scan Debug** **27**
This bit is reserved. It must be set to 0.

IR1 **I-cache Scratchpad RAM** **26**
Setting this bit to 1 enables scratchpad RAM mode in Set-1 of the I-cache. Clearing it to 0 disables scratchpad RAM mode.

EVI **External Vectored Interrupt** **25**
This bit enables and disables external vectored interrupt. Setting the bit to 1 enables the interrupt and clearing it to 0 disables the interrupt.

CMP **R3000 Compatibility** **24**
This bit enables and disables R3000 compatibility mode. Setting the bit to 1 enables the mode and clearing it to 0 disables the mode.

IIE **I-cache Invalidate Enable** **23**
This bit enables and disables the I-cache invalidate request. Setting the bit to 1 enables the request and clearing it to 0 disables the request.

DIE **D-cache Invalidate Enable** **22**
This bit enables and disables the D-cache invalidate request. Setting the bit to 1 enables the request and clearing the bit to 0 disables the request.

MUL **Multiplier** **21**
This bit enables and disables the hardware multiplier. Setting the bit to 1 enables the multiplier and clearing the bit to 0 disables the multiplier.

MAD **Multiply Accumulate** **20**
This bit allows the multiplier to support accumulate extensions. Setting the bit to 1 enables the feature and clearing the bit disables the feature. When this bit is set, MUL must also be set.

TMR **Timer** **19**

Setting this bit to 1 enables the timer facility associated with the CW4011 core's Count and Compare Registers. When this bit is set, and the value of the Count Register equals the value of the Compare Register, interrupt bit IP7 in the Cause Register is set. IP7 causes an interrupt in the next execution cycle, provided that interrupts are enabled by setting the Interrupt Enable bit in the Status Register to 1 and clearing the Error Level and Exception Level bits in the Status Register to 0.

BGE **BIU Bus Grant Enable** **18**

This bit enables and disables the BIU bus grant. Setting this bit to 1 enables the external bus master. Clearing it to 0 causes the CW4011 core to ignore the external bus master.

IE0 **I-cache Set-0 Enable** **17**

This bit enables and disables Set-0 of the I-cache. Setting the bit to 1 enables Set-0 and clearing the bit to 0 disables Set-0.

IE1 **I-cache Set-1 Enable** **16**

This bit enables and disables Set-1 of the I-cache. Setting the bit to 1 enables Set-1 and clearing the bit to 0 disables Set-1.

IS[1:0] **I-cache Size** **[15:14]**

The IS[1:0] field determines the size of each I-cache set. The field settings are defined as follows:

IS1	IS0	Cache Set Size
0	0	1 Kbyte
0	1	2 Kbyte
1	0	4 Kbyte
1	1	8 Kbyte

DE0 **D-cache Set-0 Enable** **13**

This bit enables and disables Set-0 of the D-cache. Setting the bit to 1 enables Set-0 and clearing the bit to 0 disables Set-0.

DE1 **D-cache Set-1 Enable** **12**

This bit enables and disables Set-1 of the D-cache. Setting the bit to 1 enables Set-1 and clearing it to 0 disables Set-1.

DS[1:0] **D-cache Size** **[15:14]**

The DS[1:0] field determines the size of each D-cache set. The field settings are defined as follows:

DS1	DS0	Cache Set Size
0	0	1 Kbyte
0	1	2 Kbyte
1	0	4 Kbyte
1	1	8 Kbyte

IPWE **In-Page Write Enable** **9**

This bit enables and disables in-page write operations. Setting the bit to 1 enables in-page write and clearing it to 0 disables in-page write.

IPWS[1:0] **In-Page Write Size** **[8:7]**

The IPWS[1:0] field determines the size of the I-cache set. The field settings are defined as follows:

IPWS1	IPWS0	In-Page Write Size
0	0	1 Kbyte
0	1	2 Kbyte
1	0	4 Kbyte
1	1	8 Kbyte

TE **TLB Enable** **6**

This bit enables and disables the TLB. Since the LR4500 does not support a full TLB, this bit has no effect.

WB **Write Back** **5**

This bit defines the caching algorithm, for *kseg0*. It also defines the caching algorithm for kuseg and *kseg2* if there is no TLB or if the TLB is disabled. Setting the bit to 1 enables a write back operation and clearing the bit to 0 enables a write through operation.

SR0 **Scratchpad RAM Mode Set-0** **4**
This bit enables and disables scratchpad RAM mode for Set-0 of the D-cache. Setting the bit to 1 enables scratchpad mode and clearing it to 0 disables scratchpad mode.

SR1 **Scratchpad RAM Mode Set-1** **3**
This bit enables and disables scratchpad RAM mode for Set-1 of the D-cache. Setting the bit to 1 enables scratchpad mode and clearing it to 0 disables scratchpad mode.

IsC **Isolate Cache** **2**
This bit enables isolate cache mode. This means that stores to the cache are not propagated to external memory. Setting the bit to 1 enables the mode and clearing the bit to 0 disables the mode.

TAG **Tag Test Mode** **1**
This bit enables and disables tag test mode, which is used for cache maintenance. Setting the bit to 1 enables the mode, which means that load and store operations access the Tag RAMs and sample the tag bits Tag Data, Hit, Write Back (D-cache only), and Valid. Clearing the bit to 0 disables tag test mode. This bit is used when IsC = 1.

INV **Invalidate Cache Mode** **0**
This bit enables and disables invalidate cache mode, which is used for cache maintenance. Setting the bit to 1 enables the mode. Clearing the bit to 0 disables invalidate cache mode. This bit is used with IsC = 1.

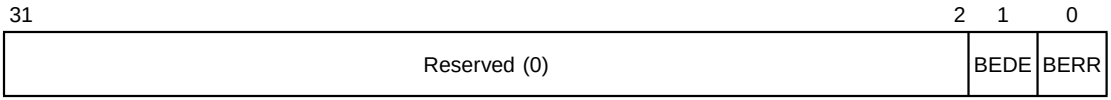
3.3.2 Lbus Controller Registers

The Lbus controller has three 32-bit registers that store information about SCbus errors and interrupts. They are the SCbus Error Status Register, the SCbus Error Address Register, and the External Vectored Interrupt Register. You must access these registers through *kseg1*. Access to an unused address causes an SCbus timeout error.

3.3.2.1 SCbus Error Status Register

The SCbus Status Register stores the bus error detect enable bit, BEDE, and the bus error detected bit, BERR. The register's virtual address is 0xB010 0004 and its physical address is 0x1010 0004. For further information about this register, refer to "SCbus Timeout Watchdog Timer" on page 7-8.

Figure 3.3 SCbus Status Register



Reserved **Reserved** **[31:2]**
This field is reserved. The bits are cleared to 0.

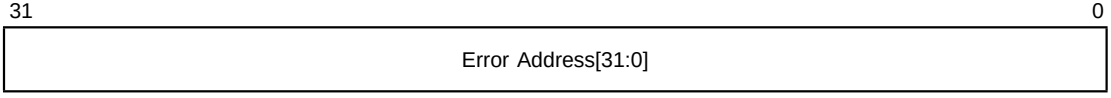
BEDE **Bus Error Detect Enable** **1**
When this bit is set to 1, the LR4500 is enabled to detect SCbus errors.

BERR **Bus Error** **0**
This bit is set to 1 when a bus error has been detected.

3.3.2.2 SCbus Error Address Register

The SCbus Error Address Register stores the address of the transaction that has caused the bus error. The address remains stored during the period that the bus error bit, BERR, is set. The register's virtual address is 0xB010 0000 and its physical address is 0x1010 0000. For further information about this register, refer to "SCbus Timeout Watchdog Timer" on page 7-8.

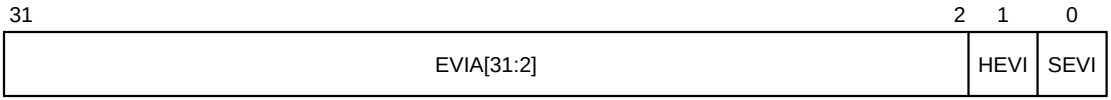
Figure 3.4 SCbus Error Address Register



3.3.2.3 External Vectored Interrupt Register

The External Vectored Interrupt Register supports the LR4500 interrupt exception feature called External Vectored Interrupt. The register's virtual address is 0xB010 0008 and its physical address is 0x1010 0008. For further information about this register, refer to "SCbus Timeout Watchdog Timer" on page 7-8.

Figure 3.5 External Vectored Interrupt Register



EVIA[31:2] **External Vectored Interrupt Address** **[31:2]**
This field contains the exception vector address.

HEVI **Hardware External Vectored Interrupt** **1**
This bit is set when the error that caused the interrupt is a hardware error.

SEVI **Software External Vectored Interrupt** **1**
This bit is set when the error that caused the interrupt is a software error.

Chapter 4
Instruction Set

This chapter provides information about the LR4500 instruction set. It includes:

- A list of the LR4500 instructions and a definition of each instruction.
- Definitions of the instruction set extensions.
- CPU instruction opcode bit encoding.

4.1 Instruction Set

Table 4.14 lists and describes the instructions that make up the LR4500 instruction set. The chip supports both MIPS I and 32-bit MIPS II instructions and also implements additional extended instructions that are specific to the LR4500.

The instructions are arranged alphabetically within the following functional groups:

- Branch Likely Instructions, in Table 4.10 on page 4-11
- Trap Instructions, in Table 4.11 on page 4-12
- Special Instructions, in Table 4.12 on page 4-13
- CP0 Instructions, in Table 4.13 on page 4-13
- Cache Maintenance Instructions, in Table 4.14 on page 4-14

Table 4.1 describes the load and store instructions.

Table 4.1 Load and Store Instructions

Instruction	Format and Description
Load Byte	`LB rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Sign extend the contents of addressed byte and load into `rt`.
Load Byte Unsigned	`LBU rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Zero extend the contents of addressed byte and load into `rt`.
Load Halfword	`LH rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Sign extend the contents of addressed halfword and load into `rt`.
Load Halfword Unsigned	`LHU rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Zero extend contents of addressed halfword and load into `rt`.
Load Word	`LW rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address, and load the addressed word into `rt`.
Load Word Left	`LWL rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Shift addressed word left so that addressed byte is left most byte of a word. Merge bytes from memory with contents of register `rt` and load result into register `rt`.
Load Word Right	`LWR rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Shift addressed word right so that addressed byte is right most byte of a word. Merge bytes from memory with contents of register `rt` and load result into register `rt`.

Table 4.1 Load and Store Instructions (Cont.)

Instruction	Format and Description
Store Byte	`SB rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Store least significant byte of register `rt` at addressed location.
Store Halfword	`SH rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Store least significant halfword of register `rt` at addressed location.
Store Word	`SW rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Store contents of register `rt` at addressed location.
Store Word Left	`SWL rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Shift contents of register `rt` left so that the left most byte of the word is in the position of the addressed byte. Store word containing shifted bytes into word at addressed byte.
Store Word Right	`SWR rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of register base to form address. Shift contents of register `rt` right so that the right most byte of the word is in the position of the addressed byte. Store word containing shifted bytes into word at addressed byte.

Table 4.2 describes the load linked MIPS II instructions.

Table 4.2 Load Linked MIPS II Instructions

Instruction	Format and Description
Load Linked	`LL rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of the register base to form the address. Load the addressed word into register `rt`.
Store Conditional	`SC rt, offset(base)` Sign extend the 16-bit `offset` and add to the contents of the register base to form the address. Conditionally store register `rt` at the address, based on whether the load-link has been “broken.”
Synchronize	`SYNC` Complete all outstanding load and store instructions before allowing any new load or store instruction to start.

Table 4.3 describes the ALU immediate instructions.

Table 4.3 ALU Immediate Instructions

Instruction	Format and Description
Add Immediate	`ADDI rt, rs, immediate` Add 16-bit, sign extended `immediate` to register `rs` and place 32-bit result in register `rt`. Trap on two's complement overflow.
Add Immediate Unsigned	`ADDIU rt, rs, immediate` Add 16-bit, sign extended `immediate` to register `rs` and place 32-bit result in register `rt`. Do not trap on overflow.
AND Immediate	`ANDI rt, rs, immediate` Zero extend 16-bit `immediate`, AND with contents of register `rs`, and place result in register `rt`.
Exclusive OR Immediate	`XORI rt, rs, immediate` Zero extend 16-bit `immediate`, exclusive OR with contents of register `rs`, and place result in register `rt`.
Load Upper Immediate	`LUI rt, immediate` Shift 16-bit `immediate` left 16 bits. Set least-significant 16 bits of word to zeros. Store result in register `rt`.
OR Immediate	`ORI rt, rs, immediate` Zero extend 16-bit `immediate`, OR with contents of register `rs`, and place result in register `rt`.
Set on Less than Immediate	`SLTI rt, rs, immediate` Compare 16-bit, sign extended `immediate` with register `rs` as signed 32-bit integers. Result = 1 if `rs` is less than immediate; otherwise result = 0. Place result in register `rt`.
Set on Less than Immediate Unsigned	`SLTIU rt, rs, immediate` Compare 16-bit, sign extended `immediate` with register `rs` as unsigned 32-bit integers. Result = 1 if `rs` is less than `immediate`; otherwise result = 0. Place result in register `rt`.

Table 4.4 describes the ALU three-operand register type instructions.

Table 4.4 ALU Three-Operand Register Type Instructions

Instruction	Format and Description
Add	`ADD rd, rs, rt` Add contents of registers `rs` and `rt` and place 32-bit result in register `rd`. Trap on two's complement overflow.
Add Unsigned	`ADDU rd, rs, rt` Add contents of registers `rs` and `rt` and place 32-bit result in register `rd`. Do not trap on overflow.
AND	`AND rd, rs, rt` Bitwise AND contents of registers `rs` and `rt` and place result in register `rd`.
Exclusive OR	`XOR rd, rs, rt` Bitwise exclusive OR contents of registers `rs` and `rt` and place result in register `rd`.
NOR	`NOR rd, rs, rt` Bitwise NOR contents of registers `rs` and `rt` and place result in register `rd`.
OR	`OR rd, rs, rt` Bitwise OR contents of registers `rs` and `rt` and place result in register `rd`.
Set on Less than	`SLT rd, rs, rt` Compare contents of registers `rt` and `rs` (as signed, 32-bit integers). If register `rs` is less than `rt`, `rd` = 1; otherwise, `rd` = 0.
Set on Less than Unsigned	`SLTU rd, rs, rt` Compare contents of registers `rt` and `rs` (as unsigned, 32-bit integers). If register `rs` is less than `rt`, `rd` = 1; otherwise, `rd` = 0.
Subtract	`SUB rd, rs, rt` Subtract contents of registers `rt` from `rs` and place 32-bit result in register `rd`. Trap on two's complement overflow.
Subtract Unsigned	`SUBU rd, rs, rt` Subtract contents of register `rt` from `rs` and place 32-bit result in register `rd`. Do not trap on overflow.

Table 4.5 describes the shift instructions.

Table 4.5 Shift Instructions

Instruction	Format and Description
Shift Left Logical	`SLL rd, rt, shamt` Shift contents of register `rt` left by `shamt` bits, inserting zeros into low-order bits. Place 32-bit result in register `rd`.
Shift Left Logical Variable	`SLLV rd, rt, rs` Shift contents of register `rt` left. Low-order 5 bits of register `rs` specify the number of bits to shift. Insert zeros into low-order bits of `rt` and place 32-bit result in register `rd`.
Shift Right Arithmetic	`SRA, rd, rt, shamt` Shift contents of register `rt` right by `shamt` bits, sign extending the high-order bits. Place 32-bit result in register `rd`.
Shift Right Arithmetic Variable	`SRAV rd, rt, rs` Shift contents of register `rt` right. Low-order 5 bits of register `rs` specify the number of bits to shift. Sign extend the high-order bits of `rt` and place 32-bit result in register `rd`.
Shift Right Logical	`SRL rd, rt, shamt` Shift contents of register `rt` right by `shamt` bits, inserting zeros into high-order bits. Place 32-bit result in register `rd`.
Shift Right Logical Variable	`SRLV rd, rt, rs` Shift contents of register `rt` right. Low-order 5 bits of register `rs` specify the number of bits to shift. Insert zeros into high-order bits of `rt` and place 32-bit result in register `rd`.

Table 4.6 describes the multiply/divide instructions.

Table 4.6 Multiply/Divide Instructions

Instruction	Format and Description
Divide	`DIV rs, rt` Divide contents of registers `rs` by the contents of `rt` as two's complement values. Place the 32-bit quotient in special register EntryLo and the 32-bit remainder in EntryHi.
Divide Unsigned	`DIVU rs, rt` Divide contents of registers `rs` by the contents of `rt` as unsigned values. Place the 32-bit quotient in special register EntryLo and the 32-bit remainder in EntryHi.
Move from HI	`MFHI rd` Move contents of special register EntryHi to register `rd`.
Move from LO	`MFLO rd` Move contents of special register EntryLo to register `rd`.
Move to HI	`MTHI rs` Move contents of register `rs` to special register EntryHi.
Move to LO	`MTLO rs` Move contents of register `rd` to special register EntryLo.
Multiply	`MULT rs, rt` Multiply contents of registers `rs` and `rt` as two's complement values. Place the 64-bit results in special registers EntryHi and EntryLo. (The EntryLo and EntryHi Registers are read/write registers that access the TLB.)
Multiply Unsigned	`MULTU rs, rt` Multiply contents of registers `rs` and `rt` as unsigned values. Place 64-bit results in special registers EntryHi and EntryLo.

Table 4.7 describes the extended computational instructions.

Table 4.7 Extended Computational Instructions

Instruction	Format and Description
Add Circular Immediate	`ADDCIU rt, rs, immediate` The 16-bit immediate is sign extended and added to the contents of general register `rs`, with the result masked by the value in CP0's CMask Register according to the formula: $rt = (rs_{31...cmask} \| \| (rs + signextended\_imed)_{cmask - 1...0})$
Find First Clear Bit	`FFC rd, rs` Starting at the most significant bit in register `rs`, find the first bit which is set to 0, and return the bit number in register `rd`. If no bit is set, return with all bits of `rd` set to 1.
Find First Set Bit	`FFS rd, rs` Starting at the most significant bit in register `rs`, find the first bit which is set to 1, and return the bit number in register `rd`. If no bit is set, return with all bits of `rd` set to 1.
Maximum	`MAX rd, rs, rt` Compare the contents of registers `rs` and `rt` as two's complement values. The larger value is stored in register `rd`.
Minimum	`MIN rd, rs, rt` Compare the contents of registers `rs` and `rt` as two's complement values. The smaller value is stored in register `rd`.
Multiply/Add	`MADD rs, rt` Multiply contents of registers `rs` and `rt` as two's complement values. Add 64-bit results to the contents of the EntryLo Register and EntryHi Register, and place the results in EntryLo and EntryHi. (The EntryLo and EntryHi Registers are read/write registers that access the TLB.)
Multiply/Add Unsigned	`MADDU rs, rt` Multiply contents of registers `rs` and `rt` as unsigned values. Add 64-bit results to the contents of the EntryLo Register and EntryHi Register, and place the results in EntryLo and EntryHi.
Multiply/Subtract	`MSUB rs, rt` Multiply contents of registers `rs` and `rt` as two's complement values. Subtract the 64-bit results from the contents of the EntryLo Register and EntryHi Register, and place the results in EntryLo and EntryHi.
Multiply/Subtract Unsigned	`MSUBU rs, rt` Multiply contents of registers `rs` and `rt` as unsigned values. Subtract the 64-bit results from the contents of the EntryLo Register and EntryHi Register, and place the results in EntryLo and EntryHi.
(Sheet 1 of 2)	

Table 4.7 Extended Computational Instructions (Cont.)

Instruction	Format and Description
Select and Shift Left	`SELSL rd, rs, rt` Using register `rs` and `rt` as a 64-bit register pair, and the contents of the CP0's Rotate Register as the shift count, shift the register pair `rs`/`rt` left the number of bits specified in the Rotate Register, and place the most significant 32-bit value in result register `rd`.
Select and Shift Right	`SELSR rd, rs, rt` Using register `rs` and `rt` as a 64-bit register pair, and the contents of the CP0's Rotate Register as the shift count, shift the register pair `rs`/`rt` right the number of bits specified in the Rotate Register, and place the least significant 32-bit value in result register `rd`.
(Sheet 2 of 2)	

Table 4.8 describes the jump instructions.

Table 4.8 Jump Instructions

Instruction	Format and Description
Jump	`J target` Shift 26-bit target address left two bits, combine with four high-order bits of PC, and jump to address with a one-instruction delay.
Jump and Link	`JAL target` Shift 26-bit target address left two bits, combine with four high-order bits of PC, and jump to address with a one-instruction delay. Place address of instruction following delay slot in r31 (link register).
Jump and Link Register	`JALR rs, rd` Jump to address contained in register `rs` with a one-instruction delay. Place address of instruction following delay slot in `rd`.
Jump Register	`JR rs` Jump to address contained in register `rs` with a one-instruction delay.

Table 4.9 describes the branch instructions.

Table 4.9 Branch Instructions

Instruction	Format and Description
Branch on Equal[1]	`BEQ rs, rt, offset` Branch to target address if register `rs` is equal to register `rt`.
Branch on Greater than or Equal to Zero	`BGEZ rs, offset` Branch to target address if register `rs` is greater than or equal to 0.
Branch on Greater than or Equal to Zero and Link	`BGEZAL rs, offset` Place address of instruction following delay slot in register r31 (link register). Branch to target address if register `rs` is greater than or equal to 0.
Branch on Greater than Zero	`BGTZ rs, offset` Branch to target address if register `rs` is greater than 0.
Branch on Less than or Equal to Zero	`BLEZ rs, offset` Branch to target address if register `rs` is less than or equal to 0.
Branch on Less than Zero	`BLTZ rs, offset` Branch to target address if register `rs` is less than 0.
Branch on Less than Zero and Link	`BLTZAL rs, offset` Place address of instruction following delay slot in register r31 (link register). Branch to target address if register `rs` is less than 0.
Branch on Not Equal	`BNE rs, rt, offset` Branch to target address if register `rs` does not equal register `rt`.

1. All branch-instructions target addresses are computed as follows: add the address of instruction in the delay slot and the 16-bit offset (shifted left two bits and sign-extended to 32 bits). All branches occur with a delay of one instruction.

Table 4.10 describes the branch likely instructions.

Table 4.10 Branch Likely Instructions

Instruction	Format and Description
Branch on Equal Likely	`BEQL rs, rt, offset` Branch to target address if register `rs` is equal to register `rt`.
Branch on Greater than or Equal to Zero Likely	`BGEZL rs, offset` Branch to target address if register `rs` is greater than or equal to 0.
Branch on Greater than or Equal to Zero and Link Likely	`BGEZALL rs, offset` Place address of instruction following delay slot in register r31 (link register). Branch to target address if register `rs` is greater than or equal to 0.
Branch on Greater than Zero Likely	`BGTZL rs, offset` Branch to target address if register `rs` is greater than 0.
Branch on Less than or Equal to Zero Likely	`BLEZL rs, offset` Branch to target address if register `rs` is less than or equal to 0.
Branch on Less than Zero and Link Likely	`BLTZALL rs, offset` Place address of instruction following delay slot in register r31 (link register). Branch to target address if register `rs` is less than 0.
Branch on Less than Zero Likely	`BLTZL rs, offset` Branch to target address if register `rs` is less than 0.
Branch on Not Equal Likely	`BNEL rs, rt, offset` Branch to target address if register `rs` does not equal register `rt`.

Table 4.11 describes the trap instructions.

Table 4.11 Trap Instructions

Instruction	Format and Description
Trap if Equal	`TEQ rs, rt` Trap if register `rs` is equal to register `rt`.
Trap if Equal Immediate	`TEQI rs, immediate` Trap if register `rs` is equal to the immediate value.
Trap if Greater than or Equal	`TGE rs, rt` Trap if register `rs` is greater than or equal to register `rt`.
Trap if Greater than or Equal Immediate	`TGEI rs, immediate` Trap if register `rs` is greater than or equal to the immediate value.
Trap if Greater than or Equal Immediate Unsigned	`TGEIU rs, immediate` Trap if register `rs` is greater than or equal to the immediate value.
Trap if Greater than or Equal Unsigned	`TGEU rs, rt` Trap if register `rs` is greater than or equal to register `rt`.
Trap if Less than	`TLT rs, rt` Trap if register `rs` is less than register `rt`.
Trap if Less than Immediate	`TLTI rs, immediate` Trap if register `rs` is less than the immediate value.
Trap if Less than Immediate Unsigned	`TLTIU rs, immediate` Trap if register `rs` is less than the immediate value.
Trap if Less than Unsigned	`TLTU rs, rt` Trap if register `rs` is less than register `rt`.
Trap if Not Equal	`TNE rs, rt` Trap if register `rs` is not equal to `rt`.
Trap if Not Equal Immediate	`TNEI rs, immediate` Trap if register `rs` is not equal to the immediate value.

Table 4.12 describes the special instructions.

Table 4.12 Special Instructions

Instruction	Format and Description
Breakpoint	BREAK Initiate breakpoint trap, immediately transferring control to exception handler.
System Call	SYSCALL Initiate system call trap, immediately transferring control to exception handler.

Table 4.13 describes the CP0 instructions.

Table 4.13 CP0 Instructions

Instruction	Format and Description
Exception Return[1]	ERET (R4000 Mode) Load the PC from ErrorEPC(SR2 = 1:Error Exception) or EPC(SR2=0:Exception) and clear ERL bit (SR2 = 1) or EXL bit (SR2 = 0) in the Status Register. SR2 is Status register bit[2].
Move from CP0	MFC0 rt, rd Load contents of CP0 register rd into CPU register rt.
Move to CP0	MTC0 rt, rd Load contents of CPU register rt into CP0 register rd.
Restore From Exception[1]	RFE (R3000 Mode) Restore previous interrupt mask and mode bits of the Status register into current status bits. Restore old status bits into previous status bits.
Wait for Interrupt	WAITI Stop execution of instructions and place the processor in a power save (stall) condition until a hardware interrupt, NMI (nonmaskable interrupt), or reset is received.

1. ERET and RFE cannot be legal at the same time. The one that is not legal causes a reserved instruction exception.

Table 4.14 describes the cache maintenance instructions.

Table 4.14 Cache Maintenance Instructions

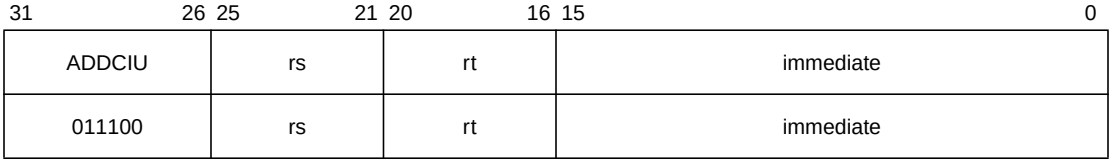
Instruction	Format and Description
Flush D-cache	`FLUSHD` Flush D-cache. 256 stall cycles will be needed.
Flush I-cache	`FLUSHI` Flush I-cache. 256 stall cycles will be needed.
Flush I-cache & D-cache	`FLUSHID` Flush both I-cache and D-cache in 256 stall cycles.
Write Back	`WB offset(base)` Write back a D-cache line addressed by offset + GPR[base]. This instruction applies to both D-cache sets.

4.2 CW4011 Instruction Set Extensions

This section defines the CW4011 instruction set extensions.

ADDCIU **Add with Circular Mask Immediate**

Format



31 26	25 21	20 16	15 0
ADDCIU	rs	rt	immediate
011100	rs	rt	immediate

Syntax **`ADDCIU rt, rs, immediate`**

Description The immediate field of the instruction is sign extended and added to the contents of general register `rs`, the result is masked with the expanded value in special register CMask according to the equation shown below. The CMask register is CP0 register number 24, whose valid bits are [4:0].

The carries resulting from the addition of the sign extended `offset` are not propagated into the final result beyond bit CMask - 1.

Operation

```
T: sign_extend_immed = (immediate15)16 || immediate15..0
   GPR[rt] = GPR[rs]31..cmask || (GPR[rs] +
    sign_extend_immed)cmask - 1..0
```

Exceptions None

FFC **Find First Clear Bit**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	0	rd	0	FFC
000000	rs	0	rd	00000	001011

Syntax **`FFC rd, rs`**

Description The contents of general register `rs` are examined starting with the most significant bit. The bit number of the first clear bit is returned in general register `rd`. If no bit is set, all ones are returned in `rd`.

Exceptions None

FFS **Find First Set Bit**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	0	rd	0	FFS
000000	rs	0	rd	00000	001010

Syntax **`FFS rd, rs`**

Description The contents of general register `rs` are examined starting with the most significant bit. The bit number of the first set bit is returned in general register `rd`. If no bit is set, all ones are returned in `rd`.

Exceptions None

FLUSHD **FLUSH Data Cache**

Format

31 26	25 21	20 16	15 0
CACHE	0	FLUSHD	0
101111	00000	00010	0

Syntax **FLUSHD**

Description FLUSHD flushes all Data Cache lines and causes stall cycles for 256 clocks, regardless of the cache size.

Exceptions None

FLUSHI **FLUSH Instruction Cache**

Format

31 26	25 21	20 16	15 0
CACHE	0	FLUSHI	0
101111	00000	00001	0

Syntax **FLUSHI**

Description FLUSHI flushes all Instruction Cache lines and causes stall cycles for 256 clocks, regardless of the cache size.

Exceptions None

FLUSHID **FLUSH Instruction and Data Cache**

Format

31 26	25 21	20 16	15 0
CACHE	0	FLUSHID	0
101111	00000	00011	0

Syntax **`FLUSHID`**

Description `FLUSHID` flushes all Data and Instruction Cache lines and causes stall cycles for 256 clocks, regardless of the cache size.

Exceptions None

MADD **Multiply Add**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	0	0	MADD
000000	rs	rt	0	00000	011100

Syntax **`MADD rs, rt`**

Description The contents of general register `rs` and the contents of general register `rt` are multiplied. Both operands are treated as 32-bit two's complement values. When the operation is completed, the doubleword result is added to special register pair HI/LO.

No overflow exception occurs under any circumstances.

This instruction is only available when the chip has multiplier-accumulator module hardware and MAD/MUL are set to one in the Cache Configuration and Control (CCC) register.

MADD executes in multiple cycles, depending on the number of significant bits in the operands. Refer to Table 4.15 on page 4-31.

Operation

```
T:      t <- (HI || LO) + (GPR[rs] * GPR[rt])
        LO <- t31..0, HI <- t63..32
```

Exceptions None

MADDU **Multiply Add Unsigned**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	0	0	MADDU
000000	rs	rt	0	00000	011101

Syntax **`MADDU rs, rt`**

Description The contents of general register `rs` and the contents of general register `rt` are multiplied with both operands treated as 32-bit unsigned values. When the operation is completed, the doubleword result is added to special register pair HI/LO.

No overflow exception occurs under any circumstances.

This instruction is only available when the chip has multiplier-accumulator module hardware and MAD/MUL are set to one in the CCC register.

The instruction executes in multiple cycles, depending on the number of significant bits in the operands. Refer to Table 4.15 on page 4-31.

Operation

```
T:      t <- (HI || LO) + ((0||GPR[rs]) * (0||GPR[rt]))
        LO <- t31..0, HI <- t63..32
```

Exceptions None

MAX **Maximum**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	rd	0	MAX
000000	rs	rt	rd	00000	101001

Syntax **`MAX rd, rs, rt`**

Description The source operands `rs` and `rt` are compared as two's complement values. The larger value is stored in the `rd` register.

Operation

```
T:      if GPR[rs] > GPR[rt] then
              GPR[rd] <- GPR[rs]
        else
              GPR[rd] <- GPR[rt]
        endif
```

Exceptions None

MIN **Minimum**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	rd	0	MIN
000000	rs	rt	rd	00000	101000

Syntax **`MIN rd, rs, rt`**

Description The source operands `rs` and `rt` are compared as two's complement values. The smaller value is stored in the `rd` register.

Operation

```
T:      if GPR[rs] < GPR[rt] then
              GPR[rd] <- GPR[rs]
        else
              GPR[rd] <- GPR[rt]
        endif
```

Exceptions None

MSUB **Multiply Subtract**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	0	0	MSUB
000000	rs	rt	0	00000	011110

Syntax **`MSUB rs, rt`**

Description The contents of general register `rs` and `rt` are multiplied and both operands are treated as 32-bit two's complement values. When the operation is complete, the doubleword result is subtracted from special register pair HI/LO.

No overflow exception occurs under any circumstances.

This instruction is only available when the chip has multiplier-accumulator module hardware and MAD/MUL are set to one in the CCC register.

The instruction executes in multiple cycles, depending on the number of significant bits in the operands. Refer to Table 4.15 on page 4-31.

Operation

```
T:      t <- (HI || LO) - (GPR[rs] * GPR[rt])
        LO <- t31..0, HI <- t63..32
```

Exceptions None

MSUBU **Multiply Subtract Unsigned**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	0	0	MSUBU
000000	rs	rt	0	00000	011111

Syntax **`MSUBU rs, rt`**

Description The contents of general register `rs` and `rt` are multiplied and both operands are treated as 32-bit unsigned values. When the operation is completed, the doubleword result is subtracted from special register pair HI/LO.

No overflow exception occurs under any circumstances.

This instruction is only available when the chip has multiplier-accumulator module hardware and MAD/MUL are set to one in the CCC register.

The instruction executes in multiple cycles, depending on the number of significant bits in the operands. Refer to Table 4.15 on page 4-31.

Operation

```
T:      t <- (HI || LO) - ((0||GPR[rs]) * (0||GPR[rt]))
        LO <- t31..0, HI <- t63..32
```

Exceptions None

SELSL **Select and Shift Left**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	rd	0	SELSL
000000	rs	rt	rd	00000	000101

Syntax **SELSL rd, rs, rt**

Description The contents of general register rs and rt are combined to form a 64-bit doubleword. The doubleword is shifted left the number of bits specified in the CP0 register ROTATE, and the upper 32 bits of the result are placed in general register rd. This ROTATE register is CP0 register number 23, with valid bits [4:0].

Operation

```
T:      s <- ROTATE4..0
        GPR[rd] <- GPR[rs]31 - s..0 || GPR[rt]31..32 - s
```

Exceptions None

SELSR **Select and Shift Right**

Format

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL	rs	rt	rd	0	SELSR
000000	rs	rt	rd	00000	000001

Syntax **SELSR rd, rs, rt**

Description The contents of general register `rs` and `rt` are combined to form a 64-bit doubleword. The doubleword is shifted right the number of bits specified in CP0 register ROTATE, and the lower 32 bits of the result are placed in general register `rd`. This ROTATE register is CP0 register number 23. Valid bits are [4:0].

Operation

```
T:      s <- ROTATE4..0
        GPR[rd] <- GPR[rs]s - 1..0 || GPR[rt]31..s
```

Exceptions None

WAITI **Wait for Interrupt**

Format

31 26	25 21	20 16	15 11	10 6	5 0
COP0		0	0	0	WAITI
010000	10000	00000	00000	00000	100000

Syntax **`WAITI`**

Description When this instruction is executed, the main processor clock stops and execution of instructions is halted. Execution resumes when a hardware interrupt, NMI, or reset exception is received. While it is in wait mode, the processor is in a power saving mode, using very little current because the clock is turned off to most of the circuitry.

`WAITI` must be followed by two or more No-Operation instructions, otherwise, the results may be undefined. Refer to Appendix A, “Programmer’s Notes in the *MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual* for further information.

Exceptions None

WB **Write Back Data Cache**

Format

31 26	25 21	20 16	15 0
CACHE	base	WB	offset
101111	base	00100	offset

Syntax **`WB offset(base)`**

Description Eight words of the Data cache line addressed by `offset` + `GPR[base]` are written back to memory if the line is dirty. Upper bits of `offset` + `GPR[base]` are ignored.

Exceptions None

4.3 CPU Instruction Opcode Bit Encoding

Tables 4.15 through 4.21 show the opcode bit encoding for CW4011 instructions. The following keys are referenced in the tables:

***rxf1** Operation codes marked with *rxf1 cause reserved instruction exceptions in all current implementations and are reserved for future versions of the architecture.

***rxf2** Operation codes marked with *rxf2 cause reserved instruction exceptions in all current implementations and are reserved for future versions of the architecture. *rxf2 is separated from other reserved instructions for COPz. These are not detected as reserved instruction codes that cause an exception on the R3000. The R4000 detects them.

***rx40** An operation code marked with *rx40 causes a reserved instruction exception on R4000 and CW4011 processors (when in R4000 mode). It is used as a Restore From Exception (RFE) instruction on the R3000, LR33000, LR33300, and CW4011 in R3000 mode.

***rx64** Operation codes marked with *rx64 cause a reserved instruction exception. They are 64-bit instructions on R4000.

***nrx** Operation codes marked with *nrx are invalid but do not cause reserved instruction exceptions in CW4011 implementations.

x1 Operation codes marked with x1 are originally extended instructions in CW4011 implementations. They are reserved instructions that cause an exception on R4000.

x2 The operation code CACHE marked with x2 is valid only for CW4011 processors with CP0 enabled and causes a reserved instruction exception with CP0 disabled. Bits [20:16] are subopcodes. They are instructions for cache maintenance, and the functions are not compatible with R4000. Recommended mnemonics are `FLUSHI`, `FLUSHD`, `FLUSHID`, and `WB offset(base)`. Undefined opcodes of CACHE instructions do not cause reserved instruction exception in CW4011 implementations.

x3 Operation codes marked with x3 are originally extended instructions in CW4011 implementations. They are used for 64-bit multiply and divide instructions on R4000. If the MUL bit or MAD bit in the CCC register is zero, they cause a reserved instruction exception. The CCC register is described in detail in Section 3.3.1, "CCC Register," on page 3-5.

x4 Operation codes marked with x4 cause a reserved instruction exception if the MUL bit in the CCC register is zero.

x5 The operation code ERET marked with x5 is valid only on the R4000 and CW4011 in R4000 mode.

x6 Operation codes marked with x6 are coprocessor-3 instructions, which are not available on R4000. These are available on the R3000 and CW4011.

Table 4.15 CW4011 Opcode Bit Encoding

	[28:26]			Opcode				
[31:29]	0	1	2	3	4	5	6	7
0	SPECIAL	REGIMM	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0	COP1	COP2	COP3[x6]	BEQL	BNEL	BLEZL	BGTZL
3	*rx64	*rx64	*rx64	*rx64	ADDCIU[x1]	*rxf1	*rxf1	*rxf1
4	LB	LH	LWL	LW	LBU	LHU	LWR	*rx64
5	SB	SH	SWL	SW	*rx64	*rx64	SWR	CACHE[x2]
6	LL	LWC1	LWC2	LWC3[x6]	*rx64	*rx64	*rx64	*rx64
7	SC	SWC1	SWC2	SWC3[x6]	*rx64	*rx64	*rx64	*rx64

Table 4.16 SPECIAL Opcode Bit Encoding

	[2:0]			SPECIAL Function				
[5:3]	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	SLL	SELSR^{x1}	SRL	SRA	SLLV	SELSL^{x1}	SRLV	SRAV
1	JR	JALR	FFS^{x1}	FFC^{x1}	SYSCALL	BREAK	*rxf1	SYNC
2	MFHI^{x4}	MTHI^{x4}	MFLO^{x4}	MTLO^{x4}	*rx64	*rxf1	*rx64	*rx64
3	MULT^{x4}	MULTU^{x4}	DIV^{x4}	DIVU^{x4}	MADD^{x3}	MADDU^{x3}	MSUB^{x3}	MSUBU^{x3}
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	MIN^{x1}	MAX^{x1}	SLT	SLTU	*rx64	*rx64	*rx64	*rx64
6	TGE	TGEU	TLT	TLTU	TEQ	*rxf1	TNE	*rxf1
7	*rx64	*rxf1	*rx64	*rx64	*rx64	*rxf1	*rx64	*rx64

Table 4.17 REGIMM Opcode rt Bit Encoding

	[18:16]			REGIMM rt				
[20:19]	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	BLTZ	BGEZ	BLTZL	BGEZL	*rxf1	*rxf1	*rxf1	*rxf1
1	TGEI	TGEIU	TLTI	TLTIU	TEQI	*rxf1	TNEI	*rxf1
2	BLTZAL	BGEZAL	BLTZALL	BGEZALL	*rxf1	*rxf1	*rxf1	*rxf1
3	*rxf1	*rxf1	*rxf1	*rxf1	*rxf1	*rxf1	*rxf1	*rxf1

Table 4.18 CACHE^{x2} Opcode rt Bit Encoding

[20:19] \ [18:16]	CACHE^{x2} rt 0	1	2	3	4	5	6	7
0	*nrx	FLUSHI^{x2}	FLUSHD^{x2}	FLUSHID^{x2}	WB^{x2}	*nrx	*nrx	*nrx
1	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
2	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
3	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx

Table 4.19 COPz rs Opcode Bit Encoding

[25:24] \ [23:21]	COPz rs 0	1	2	3	4	5	6	7
0	MFCz	*rx64	CFCz	*rxf2	MTCz	*rx64	CTCz	*rxf2
1	BC	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2
2	COPz (Coprocessor defined instructions)							
3								

Table 4.20 COPz rt Opcode Bit Encoding

[20:19] \ [18:16]	COPz rt 0	1	2	3	4	5	6	7
0	BCF	BCT	BCFL	BCTL	*rxf2	*rxf2	*rxf2	*rxf2
1	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2
2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2
3	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2	*rxf2

Table 4.21 CP0 Opcode Bit Encoding

	[2:0]	CP0 Function						
[5:3]	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	*nrx	TLBR	TLBWI	*nrx	*nrx	*nrx	TLBWR	*nrx
1	TLBP	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
2	RFE^{rx40}	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
3	$ERET^{x5}$	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
4	$WAITI^{x1}$	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
5	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
6	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx
7	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx	*nrx

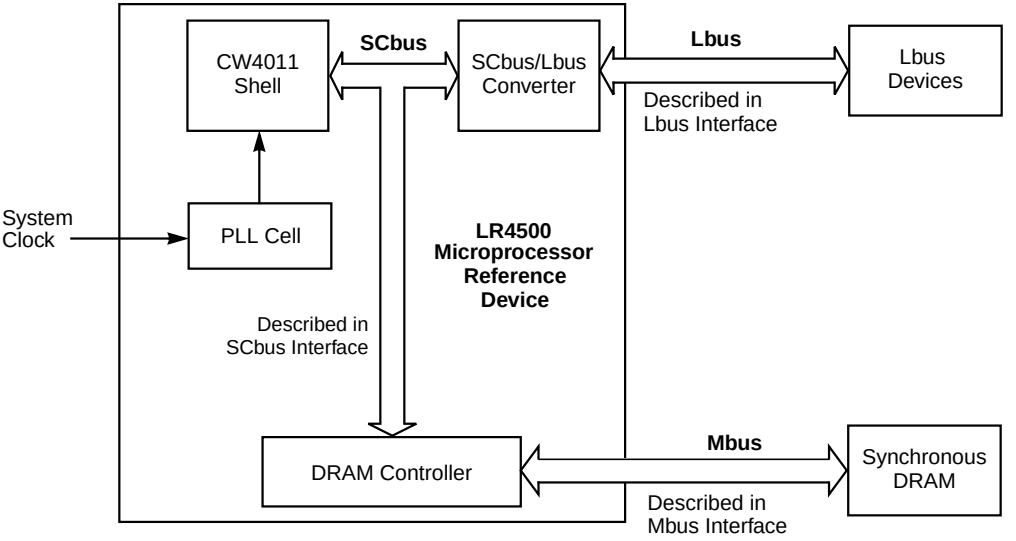
Chapter 5
Bus Interface Descriptions

This chapter describes the LR4500 interface signals and associated buffer circuitry required for SCbus signals. The chapter provides information on the following subjects:

- External interfaces
  - The Mbus provides the interface between the LR4500 DRAM Controller and the external DRAM array, see page 5-2.
  - The Lbus provides the interface between the SCLC and devices on the Lbus, see page 5-5.
  - The Phase-Locked Loop (PLL) interface provides the interface between the PLL clock generator and the CW4011 shell, see page 5-9.
  - Test-signal input pins allow LSI Logic to test the LR4500, see page 5-10.
  - A core monitor signal allows you to monitor the behavior of the CW4011 core, see page 5-11.
- Internal interfaces
  - The SCbus interfaces between the CW4011 shell and the DRAMC and SCLC, see page 5-11.
  - External buffering required for certain SCbus signals (see page 5-19).
  - The reset/interrupt interface between the CW4011 shell and the DRAMC and SCLC, see page 5-22.

Figure 5.1 shows the three major buses—the SCbus, Mbus, and Lbus.

Figure 5.1 LR4500 Interfaces



5.1 External Interfaces

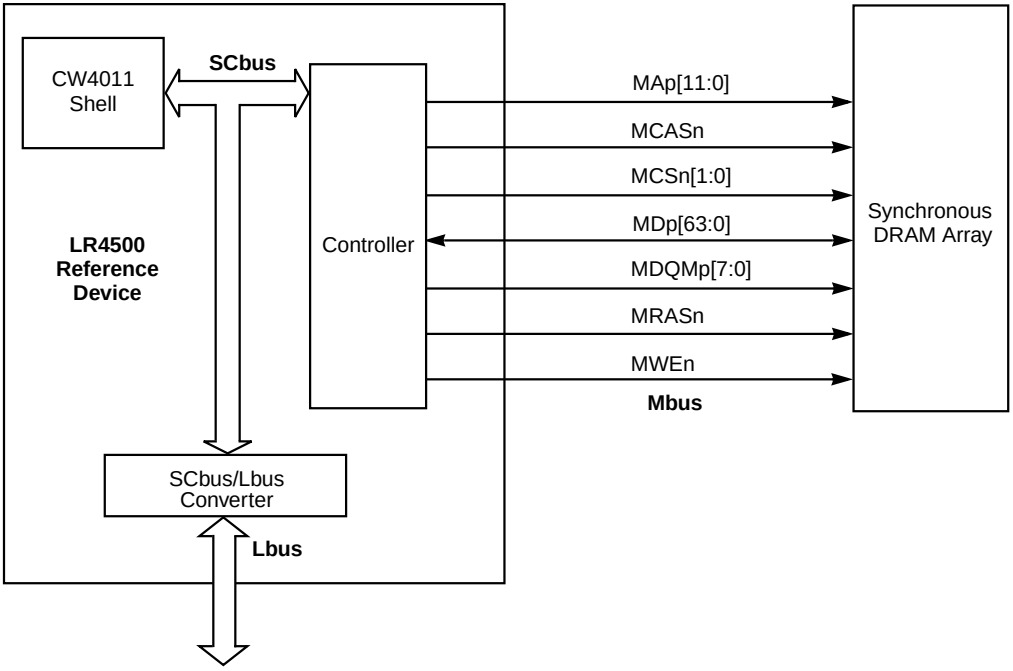
This section describes the LR4500 external interfaces: Mbus, Lbus, PLL, test, and core monitor.

Each signal definition contains the mnemonic and the full signal name. Active LOW signals have an ‘n’ suffix, for example, SCRESETn. Active HIGH signals have a ‘p’ suffix, for example, MDQMp. ‘Assert’ means to drive the signal TRUE or active. ‘Deassert’ means to drive the signal FALSE or inactive.

5.1.1 Mbus Interface

Figure 5.2 shows the 89 Mbus signals that the LR4500 uses to connect the LR4500 DRAM Controller to the synchronous DRAMs in the main memory array. Inputs and outputs are referenced to the LR4500 reference device and more specifically to the DRAM Controller.

Figure 5.2 Mbus Interface



MAp[11:0] **Multiplexed Memory Address Bus** **Output**
These multiplexed signals carry row and column addresses. MRASn strobes the row addresses into the DRAMs, and MCASn strobes the column addresses. “Address Bit Assignment” on page 6-4 provides detailed information about the address bus.

During memory initialization, the DRAM Controller uses MAp[11:0] to write the 12-bit Mode Register in each DRAM.

MCASn **Memory Column Address Strobe** **Output**
The LR4500 asserts MCASn to strobe memory column addresses into the memory devices.

MCSn[1:0] **Memory Chip Select** **Output**

MCSn[1:0] select between Banks 0 and 1 in the DRAM array. The DRAM Controller asserts MSCn0 to select the DRAMs that make up Bank 0, and asserts MSCn1 to select the DRAMs in Bank 1. If only one bank of DRAMs is installed, the DRAM Controller asserts MCSn0.

MDp[63:0] **Memory Data Bus** **Bidirectional**

This 64-bit bidirectional data bus carries data between the LR4500 and the memory array. The direction of data flow is controlled by MWEn.

MDQMp[7:0] **Memory Data Enable/Mask** **Output**

This is an 8-bit data mask used only during write operations. When asserted, each bit of the mask selects one byte of data, as shown in the examples below, to enable write operations in individual bytes of the data word.

8-Bit Wide DRAMs

Mask	Byte Selected	DRAM Byte Number
MDQMp 7	MDp[63:56]	7
MDQMp 6	MDp[55:48]	6
MDQMp 5	MDp[47:40]	5
MDQMp 4	MDp[39:32]	4
MDQMp 3	MDp[31:24]	3
MDQMp 2	MDp[23:16]	2
MDQMp 1	MDp[15:8]	1
MDQMp 0	MDp[7:0]	0

16-Bit Wide DRAMs

Mask	Byte Selected	DRAM Byte Number
MDQMp 7	MDp[63:56]	3 (Upper Byte)
MDQMp 6	MDp[55:48]	3 (Lower Byte)
MDQMp 5	MDp[47:40]	2 (Upper Byte)
MDQMp 4	MDp[39:32]	2 (Lower Byte)
MDQMp 3	MDp[31:24]	1 (Upper Byte)
MDQMp 2	MDp[23:16]	1 (Lower Byte)
MDQMp 1	MDp[15:8]	0 (Upper Byte)
MDQMp 0	MDp[7:0]	0 (Lower Byte)

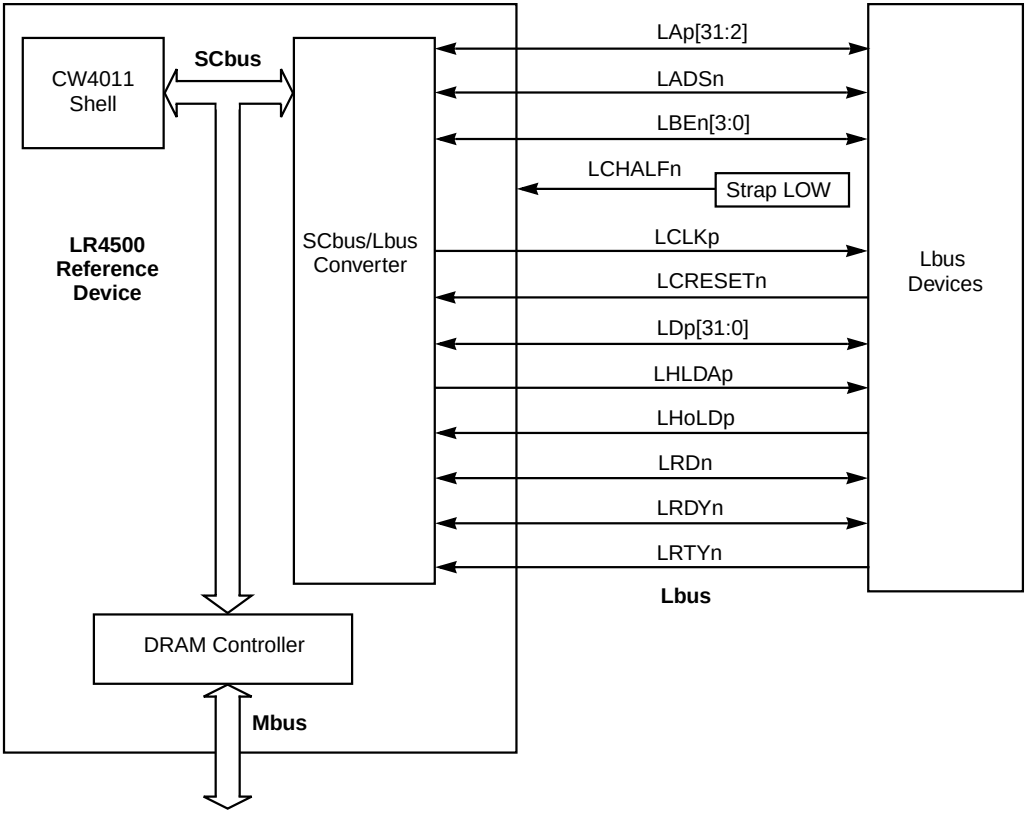
MRASn **Memory Row Address Strobe** **Output**
The LR4500 asserts MRASn to strobe memory row addresses into the memory devices.

MWEn **Memory Write Enable** **Output**
The LR4500 asserts MWEn to enable a write operation and deasserts MWEn to enable a read operation.

5.1.2 Lbus Interface

Figure 5.3 shows the 75 Lbus signals that connect the SCLC in the LR4500 with external Lbus devices. The LR4500 functions as the bus master to access external devices on the Lbus, such as boot-ROM, serial devices, and the Ethernet Controller. These Lbus devices can also function as bus master to access the DRAM through the SCLC and the DRAMC.

Figure 5.3 Lbus Interface



Inputs and outputs on the Lbus are referenced to the LR4500. Since either the LR4500 or an Lbus device can be bus master, some signals that are typically unidirectional, such as the read signal LRDn, are bidirectional on the Lbus. When the LR4500 is bus master and asserts LRDn, it enables a read operation in one of the Lbus devices. When an Lbus device is bus master, it can assert LRDn to read data from the DRAM.

LAp[31:2] **Lbus Address Bus** **Bidirectional**
When the LR4500 is master of the Lbus, it outputs the address that is used to access one of the devices on the Lbus. If one of the devices on the Lbus is bus master, it outputs to the LR4500 the address used to access the DRAM.

LADSn **Lbus Address Strobe** **Bidirectional**

This signal strobes the Lbus addresses. The bus master asserts it at the first LCLKp cycle of a transaction. When the LR4500 asserts LHLDAp and grants bus ownership to an Lbus master device, the master device inputs this signal to the LR4500. When the LR4500 is the bus master, it inputs this signal to the Lbus device. The initiating device must synchronize this signal to the Lbus clock, LCLKp.

LBEn[3:0] **Lbus Byte Enable Signals** **Bidirectional**

The master device drives these signals active (LOW), to enable data on the Lbus, as shown below. The read/write signal, LRDn, controls the direction of data flow. Byte operations, that is operations where individual bytes are selected, occur only during write transactions.

Byte Enable Signal	Byte Bits	Byte Number
LBEn 3	LDp[31:24]	Byte 3
LBEn 2	LDp[23:16]	Byte 2
LBEn 1	LDp[15:8]	Byte 1
LBEn 0	LDp[7:0]	Byte 0

LCHALFn **Lbus Clock Speed** **Input**

This signal sets the clock speed for the Lbus. When a device on the Lbus drives this signal HIGH, it divides the SCbus clock (SCLKp) by two, and the LR4500 outputs a clock signal (LCLKp) that is one half the frequency of SCLKp. When the signal is LOW, it divides the SCLKp by four, and the LR4500 outputs LCLKp at one quarter the frequency of SCLKp.

LCLKp **Lbus Clock** **Output**

This output is derived from the SCbus clock, SCLKp. Lbus clock rate is either half or quarter the clock rate of SCLKp, depending on the state of the LCHALFn input to the LR4500.

LCRESETn **LCLK Divider Reset** **Input**

LSI Logic uses this signal for testing. You can deassert the signal by strapping it HIGH.

LDp[31:0] **Lbus Data Bus** **Bidirectional**
This 32-bit bidirectional data bus transfers data between the devices on the Lbus and the LR4500. The read/write signal, LRDn, controls the direction of data flow on the Lbus.

LHLDAp **Lbus Hold Acknowledge** **Output**
The LR4500 asserts this signal in response to an LHoLDp input from an Lbus device. When asserted, the signal grants a bus hold and allows the Lbus device to take bus ownership.

LHoLDp **Lbus Hold Request** **Input**
An Lbus device asserts LHoLDp to request ownership of the Lbus. The initiating device must synchronize the signal to the rising edge of LCLKp.

LRDn **Lbus Read** **Bidirectional**
The master device asserts this signal to enable a read operation and deasserts it to enable a write operation. When the LR4500 asserts LHLDAp and grants bus ownership to an Lbus master device, the master device inputs this signal to the LR4500. When the LR4500 is the bus master, it inputs this signal to the Lbus device.

LRDYn **Lbus Data Ready** **Bidirectional**
When it is asserted, this signal terminates a transaction. When the LR4500 asserts LHLDAp and grants bus ownership to an Lbus device, the LR4500 inputs this signal to the LR4500. When the LR4500 is the bus master, the Lbus device inputs this signal to the LR4500. The initiating device must synchronize this signal to the Lbus clock, LCLKp.

LRTYn **Lbus Retry** **Input**
When an Lbus slave device asserts this signal and inputs it to the LR4500, the LR4500 temporarily aborts any transaction in progress and initiates the transaction again later. The initiating Lbus device must synchronize the signal to the rising edge of LCLKp.

5.1.3 Phase-Locked Loop (PLL) Clock Signals

The PLL circuit generates the clock inputs for the CW4011 shell and for the other modules that are part of the LR4500. The test signals associated with the PLL circuit are not for general use and are therefore deasserted by strapping them LOW if they are active-high signals, and strapping them HIGH if they are active-low signals.

This section describes the PLL signals. “PLL Clock Circuit” on page 2-5 provides further information on this subject.

PLLAGND **PLL Analog Ground** **Ground**
This is the analog ground for the PLL circuit. You must connect an RC (resistor/capacitor) circuit for the PLL filter between pins PLLLP2p and PLLAGND on the PLL circuit, as shown in Figure 2.2, on page 2-6.

PLLCTop **Test Counter** **Open Output from PLL Circuit**
This signal is an open pin on the board.

PLLCTRn **Test Counter Reset** **Input to PLL Circuit**
This signal is strapped LOW.

PLLENp **VCO Enable(1)/Disable(0)** **Strapped Input**
This input to the PLL circuit enables the PLL circuit when the input is active (HIGH), and disables the PLL circuit when it is LOW. The signal is strapped HIGH on the main circuit board so that the PLL circuit is always enabled.

PLLiDDTp **Test Enable Input** **Input to PLL Circuit**
This signal enables test inputs when it is active (HIGH). It is strapped LOW on the main circuit board.

PLLLP2p **VCO Input and Loop Filter** **Filter Pin**
You must connect an RC (resistor/capacitor) circuit for the PLL filter between pins PLLLP2p and PLLAGND on the PLL circuit, as shown in Figure 2.2, on page 2-6.

PLLREFp **System Clock Reference** **Input**
This is the system reference clock that is input to the CW4011 by the PLL circuit.

PLLTDp **Test Data (Clock)** **Input to PLL Circuit**
This signal is strapped LOW.

PLLTSTp **Test Enable** **Input to PLL Circuit**
This signal is strapped LOW, which means that testing is generally disabled.

PLLVDD **PLL Power** **Input to PLL Circuit**
This signal provides VDD power.

PLLVSS **PLL Ground** **Ground**
This is the ground for the PLL circuit.

5.1.4 Test Signals

There are nine pins on the LR4500 chip that allow designers at LSI Logic to test the device using an LSI Logic tester. When the pins are not being used for testing, you must deassert all inputs by strapping active-high signals LOW and active-low signals HIGH. You must leave all outputs unconnected. Inputs are referenced to and outputs are referenced from the LR4500.

ICECLKp **ICE Serial Bit Clock Rate X16** **Input**
The ICEport requires an off-chip pin with a clock that runs at 16 times the serial transmit/receive rate.

ICERXp **RX Serial Bit Receive** **Input**
This signal carries the ICEport UART (universal asynchronous receiver/transmitter) serial input data stream.

ICETXP **TX Serial Bit Transmit** **Output**
This output sends out the ICEport UART serial data stream.

PARAMOUTp **Parametric NAND Tree—LSI Logic Use Only** **Output**
This output is used to check the parametric NAND tree. It is reserved for factory use during testing. Leave it unconnected on the board.

SCANCRip **CW4011 Core Scan—LSI Logic Use Only** **Input**
Strap this signal LOW on the board.

SCANCRop **CW4011 Core Scan—LSI Logic Use Only** **Output**
Leave this signal unconnected.

SCANENBp **Global LR4500 Scan Enable—LSI Logic Use Only** **Input**

Strap this signal LOW on the board.

TESTMp **Test Mode for Scan—LSI Logic Use Only** **Input**

This input is reserved for factory use during testing. Strap it LOW on the board.

ZSTATEn **Global 3-State Control—LSI Logic Use Only** **Input**

This input is reserved for factory use during testing. Strap it HIGH on the board.

5.1.5 CW4011 Core Monitor Signal

The LR4500 has one pin that enables you to monitor the behavior of the CW4011 core.

MCLKp **Internal Clock Monitor** **Output**

This output from the internal clock allows you to check the clock phase. When you are not using the pin to check the clock, the output should be unconnected.

5.2 Internal Interface

This section describes the LR4500 internal interfaces: SCbus, external buffering required for certain SCbus signals, and the reset/interrupt interface.

Each signal definition contains the mnemonic and the full signal name. Active LOW signals have an ‘n’ suffix, for example, SCRESETn. Active HIGH signals have a ‘p’ suffix, for example, MDQMp. ‘Assert’ means to drive the signal TRUE or active. ‘Deassert’ means to drive the signal FALSE or inactive.

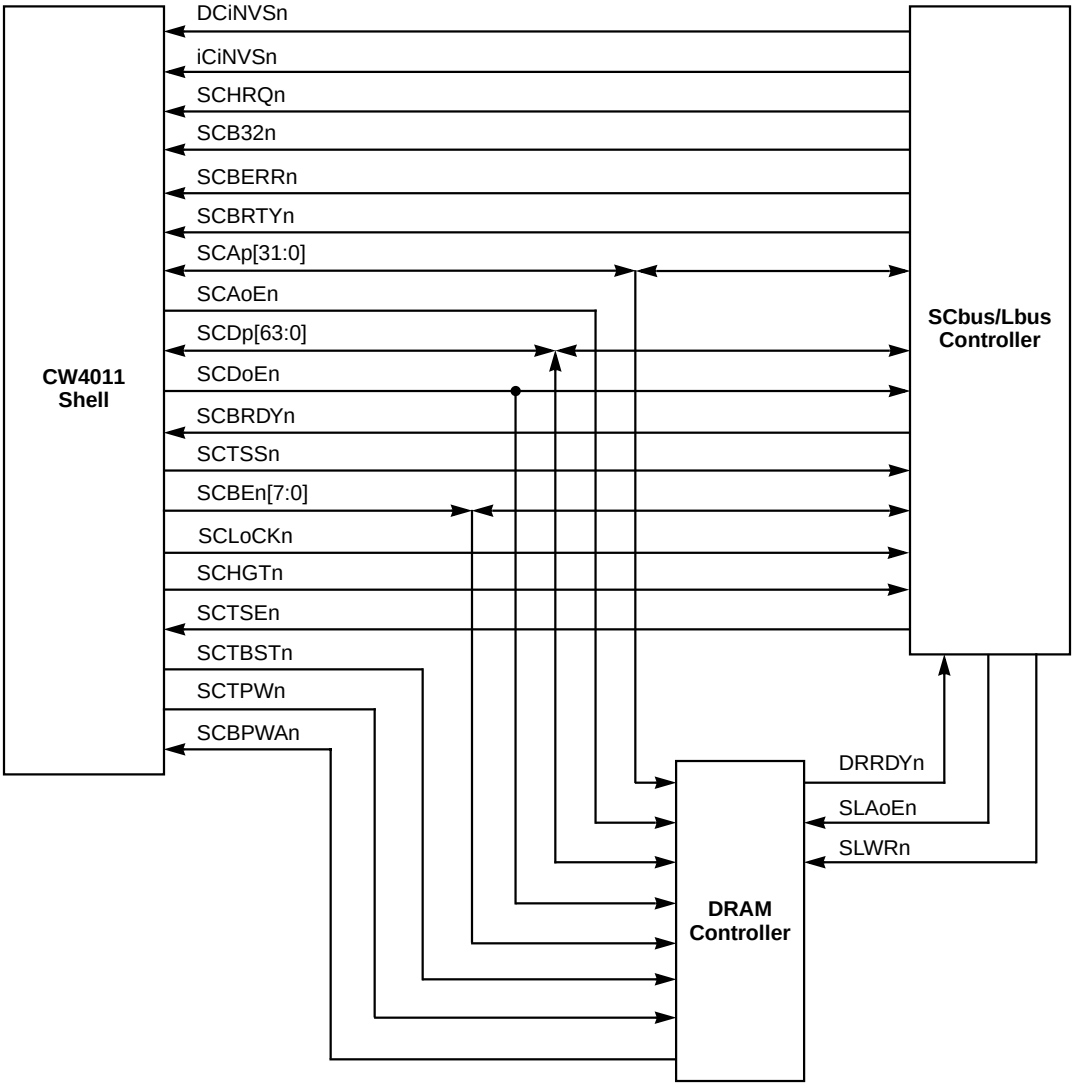
5.2.1 SCbus Interface

The SCbus interface provides the link between CW4011 core elements that are part of the CW4011 shell and the DRAMC and SCLC modules that are external to the CW4011 shell, but are part of the LR4500 reference device. Figure 5.4 shows a simplified view of this interface.

In the interface between the CW4011 shell and the SCLC, either module can function as the bus master or bus slave. In the interface between the CW4011 and the DRAM Controller, the CW4011 shell is always the master.

Figure 5.4 also shows the cache invalidation signals (iCiNVSn and DCiNVSn) that are input to the CW4011 shell from the SCLC, and the address enable, write enable, and bus ready signals that interface between the DRAMC and the SCLC. The signals described in this section are shown in Figure 5.4.

Figure 5.4 SCbus Interface



DCiNVSn **D-cache Invalidation Strobe** **Input to Shell from SCLC**

The SCLC asserts this signal to indicate that the CiNVAp invalidation address bus is valid for D-cache and there is a need for a snooping sequence. If the cache tag is not coincident with higher address bits, the line is not invalidated.

DRRDYn **DRAM Ready** **Output from DRAMC to SCLC**

The DRAMC asserts DRRDYn when the current DRAM transaction is terminated, indicating that the bus is available. The signal remains active (LOW) until the next transaction starts. The DRAMC outputs the signal to the SCLC, which merges DRRDYn with its own bus ready signal (page 5-19) and drives SCBRDYn, which is output to the CW4011 shell.

iCiNVSn **I-cache Invalidation Strobe** **Input to Shell from SCLC**

The SCLC asserts this signal to indicate that the CiNVAp invalidation address bus is valid for I-cache and there is a need for a snooping sequence. If the cache tag is not coincident with higher address bits, the line is not invalidated.

SCAp[31:0] **Address Bus** **Bidirectional between Shell and SCLC Input to DRAMC**

The CW4011 asserts the signals on this bus and outputs them to the SCLC or the DRAMC. The Lbus master can also assert SCAp[31:0] and output the address to the CW4011 shell through the SCLC. SCAp[31:0] is the 32-bit address bus used for instruction fetch and data read/write operations. The bus signals are valid only when the address output enable signal, SCAoEn, is asserted. The enable signal remains valid throughout the operation until SCBRDYn, SCBRTYn, or SCBERRn is asserted.

SCAoEn **Address Output Enable** **Output from Shell to DRAMC**

The CW4011 asserts this signal, to indicate that the address bus lines, SCAp[31:0], are valid. The signal remains active throughout the bus transaction. SCAoEn also enables SCTBSTn, SCTBEn, and SCTPWn. This signal is never valid at the same time as SLAoEn (the Address Output Enable signal output from the SCLC shell to the DRAMC (page 5-19). When TESTMp is asserted, SCAoEn is asserted. Other address output enable signals must be deasserted when TESTMp is asserted.

SCB32n **32-Bit Bus Width Sizing Input to Shell from SCLC**

The external Lbus slave asserts SCB32n to indicate that the SCbus needs 32-bit bus sizing. The CW4011 core samples this signal on the rising edge of the clock that synchronizes the SCbus ready signal, SCBRDYn. If SCB32n is asserted for a 64-bit transaction, which is a doubleword transaction or part of a burst transaction, the bus interface unit in the CW4011 core generates a subsequent 32-bit word transaction and packs data to 64 bits for a read transaction or unpacks data to 32 bits for a write transaction.

SCBEn[7:0] **Byte Enable Bidirectional between Shell and SCLC and Input to DRAMC**

SCBEn[7:0] indicates which byte positions are valid for a read or write transaction. The CW4011 asserts the signals and outputs them to the SCLC or the DRAMC. The Lbus device can also assert the signals and input them to the CW4011 through the SCLC. Only one of the signals is asserted during a byte read or byte write transaction. All signals are asserted for a doubleword or burst transaction.

SCTBEn Signal	Valid Byte positions
0	SCDop[7:0]
1	SCDop[15:8]
2	SCDop[23:16]
3	SCDop[31:24]
4	SCDop[39:32]
5	SCDop[47:40]
6	SCDop[55:48]
7	SCDop[63:56]
All	Doubleword or Burst transaction

SCBERRn **Bus Error** **Input to Shell from SCLC**

The Lbus master device asserts SCBERRn to terminate the current transaction when a bus error occurs. If SCBRDYn, or the bus retry signal, SCBRTYn, is asserted at the same time as SCBERRn, SCBERRn has higher priority. SCBERRn is reported to the CP0, which in turn generates an exception.

SCBPWAn **Bus In-Page Write Accept** **Input to Shell from DRAMC**

The DRAMC asserts SCBPWAn to indicate that it accepts in-page write transactions. The CW4011 samples the signal on the rising edge of the clock that synchronizes SCBRDYn. If the CW4011 has not asserted SCTPWn, asserting or deasserting SCBPWAn has no significance.

SCBRDYn **Bus Ready** **Input to Shell from SCLC**

The SCLC asserts SCBRDYn when the current transaction is terminated, indicating that the SCbus is available. The signal remains active (LOW) until the next transaction starts. The SCLC deasserts the signal to indicate that the SCbus is not available. The SCLC receives a bus-ready signal, DRRDYn, from the DRAMC (page 5-19), merges DRRDYn with the SCLC bus ready signal, and drives SCBRDYn, which is output to the CW4011 shell.

SCBRTYn **Bus Retry** **Input to Shell from SCLC**

The Lbus master device asserts SCBRTYn when the current transaction has been terminated unsuccessfully and must be retried later. The control state goes back to the idle state, then all bus requests are arbitrated again. If there are no higher priority requests and the Lbus master has asserted SCTSEn, there is one idle state between the first transaction and a retry transaction. If SCBRDYn and SCBRTYn are asserted at the same time, SCBRTYn has the higher priority.

SCDp[63:0] **Data Bus** **Bidirectional between Shell, SCLC, and DRAMC**

SCDp[63:0] are the data bus signals. They are output from the CW4011 shell for data read/write operations and for data write back to the D-cache. They are input to the shell for data read and instruction fetch transactions. The

CW4011 shell samples the signals on the rising edge of the clock when SCBRDYn is asserted. The signals are valid throughout a write transaction in which the CW4011 writes to DRAM through the DRAMC, or the Lbus device writes to the CW4011 or DRAMC through the SCLC. Byte ordering is little endian.

SCDoEn **Data Output Enable Output from Shell to SCLC and DRAMC**

The CW4011 asserts SCDoEn throughout a write transaction and outputs it to the SCLC or the DRAMC. The signal indicates that the current transaction is a write transaction, and it also enables data output. It performs the same function for a CW4011 write transaction to DRAM that SLWRn (page 5-19) performs for an SCLC write transaction to DRAM. When TESTMp is asserted, SCDoEn is asserted. Other data output enables must be deasserted when TESTMp is asserted.

SCHGTn **Bus Hold Grant Output from Shell to SCLC**

The CW4011's bus interface unit enters the hold state and asserts SCHGTn to indicate that it is releasing SCbus ownership in response to a bus hold request (SCHRQn) from one of the devices on the Lbus.

SCHRQn **Bus Hold Request Input to Shell from SCLC**

SCHRQn indicates that a device on the Lbus is requesting ownership of the SCbus. Bus hold request has the highest priority during bus arbitration. However, it cannot break continuous transactions of in-page writes and burst read/write transactions if those transactions are supported by an asserted SCTSEn. In such a case, SCHRQn must wait until SCTSEn is deasserted.

SCLoCKn **Bus Lock Output from Shell to SCLC**

The CW4011 asserts SCLoCKn to indicate that it wishes to lock the SCbus and restrict ownership. The CW4011 asserts the signal when a read transaction is started by executing a Load Link instruction in an uncached area or a write through cached area. It deasserts the signal just before a write transaction is started by executing a Store Conditional instruction. During read and write transactions, the CW4011 asserts the signal continuously, preventing ownership from changing during one of these transactions. If a Store Conditional

transaction hits the D-cache in a write back cached read while SCLoCKn is asserted, an incorrect condition exists, and the CW4011 deasserts SCLoCKn without completing any bus transactions.

SCTBSTn **Burst Transaction** **Output from Shell to DRAMC**
The CW4011 asserts SCTBSTn and outputs it to the DRAMC to indicate that a transaction is taking place during which four doublewords will be moved, and that the first doubleword is currently being moved. The CW4011 deasserts the signal after the first word has been transferred and during singleword transactions.

SCTPWn **Next Transaction Is In-Page Write** **Output from Shell to DRAMC**
The CW4011 asserts this signal to indicate that the next transaction is in the same DRAM page as the current transaction, as defined in the Configuration Register. When the CW4011 asserts SCTPWn, a maximum of four write transactions take place one after the other, even if there is an instruction fetch request or data read request. If there are four continuous write transactions, the CW4011 asserts SCTPWn from the first through the last (fourth) transaction. The CW4011 asserts SCTPWn from the beginning of one in-page write transaction to the end of that transaction. The write buffer in the CW4011's LSU checks to see if the subsequent write request is in the same page.

SCTSEn **Transaction Start Enable** **Input to Shell from SCLC**
SCTSEn enables or disables a new SCbus transaction. Transaction requests are arbitrated only when SCTSEn is asserted. During the time SCTSEn is deasserted, the CW4011 core's bus interface unit repeats the idle state. If it is necessary to insert an idle cycle between two transactions, the Lbus device may deassert SCTSEn then assert it when SCBRDYn is asserted.

SCTSSn **Transaction Start Strobe** **Output from Shell to SCLC**
The CW4011 asserts SCTSSn for one clock cycle at the beginning of a transaction to indicate that a transaction has started. If the next transaction begins immediately, the CW4011 asserts SCTSSn continuously.

SLAoEn **Address Output Enable** **Output from SCLC to DRAMC**

The SCLC asserts this signal to indicate that the address bus lines, SCAp[31:0], are valid. The signal remains active throughout the bus transaction. SCAoEn also enables SCTBSTn, SCTBEn, and SCTPWn. This signal is not valid at the same time as SCAoEn, which is the Address Output Enable signal output from the CW4011 shell to the SCLC described on page 5-14.

SLWRn **SCLC Write Enable** **Output from SCLC to DRAMC**

The SCLC asserts this signal throughout a DRAM write operation and outputs it to the DRAMC. It performs the same function for a CW4011 write transaction to DRAM that SCDoEn (page 5-17) performs for an SCLC write transaction to DRAM.

5.2.2 External Buffering for SCbus Signals

You must provide external buffering for certain SCbus signals, including:

- Address bus SCAp[31:0]
- Address output enable signals SCAoEn and SLAoEN
- SC data bus SCDp[63:0]
- Data output enable SCDoEn
- SCbus byte enable SCBEn[7:0]

Figure 5.5 shows an example of a buffer configuration in which the bidirectional address bus is buffered at the SCLC and CW4011 ends by BTS4A*32 3-state buffers. When the CW4011 asserts SCAoEn, the signal enables the buffer at the CW4011 end. When the SCLC asserts SLAoEn, the signal enables the buffer at the SCLC end.

During device scan test, CW4011 buffers should drive the bus. When TESTMp is asserted the output enable signal, SCAoEn, is asserted. Other address output enable signals must be deasserted.

Figure 5.5 Buffering for SCAp[31:0] Address Bus

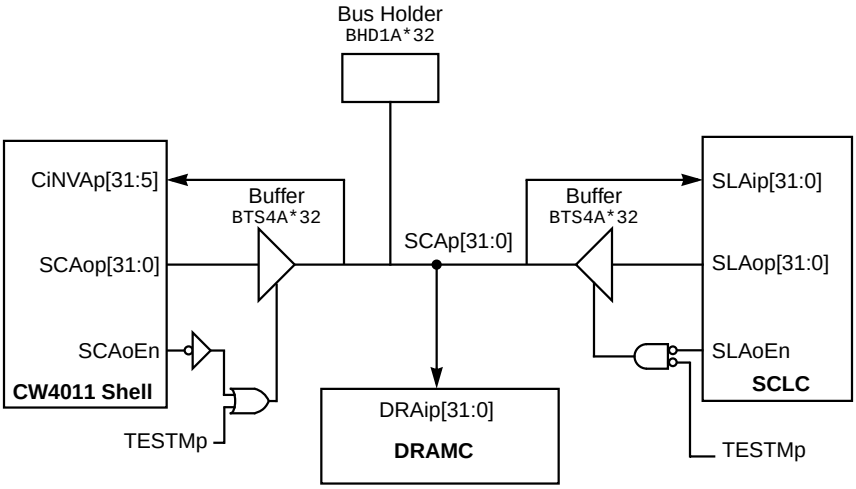


Figure 5.6 shows an example of a buffer configuration in which the SC data bus is buffered at the SCLC and CW4011 ends by BTS4A*64 3-state buffers. When the CW4011 asserts SCDoEn, the signal enables the buffer at the CW4011 end. When the SCLC asserts SLDoEn, the signal enables the buffer at the SCLC end. A BTS4A*64 buffer also buffers the data output from the DRAMC. This buffer is enabled when the DRAMC asserts DRDoEn.

During device scan test, the CW4011 buffer should drive the bus. When TESTMp is asserted, SCDoEn is asserted and other address output enables must be deasserted.

Figure 5.6 Buffering for SCDp[63:0] Data Bus

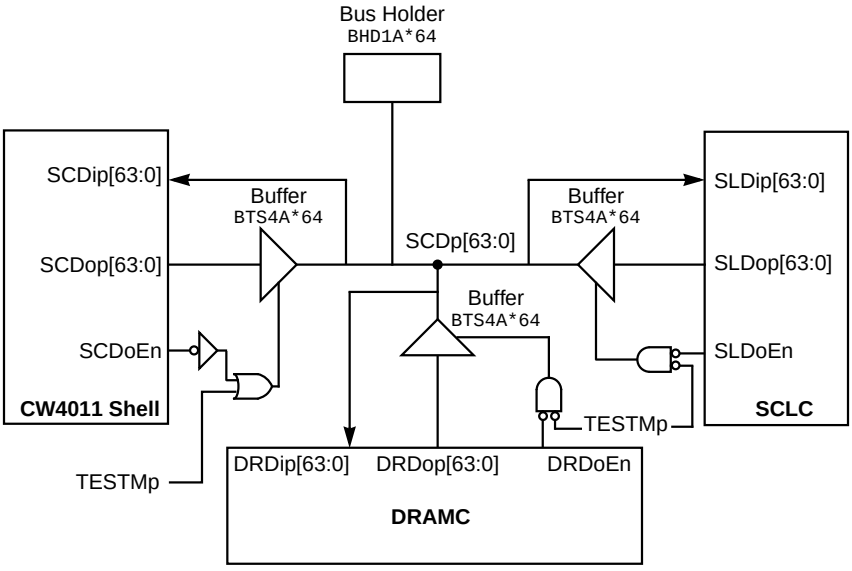
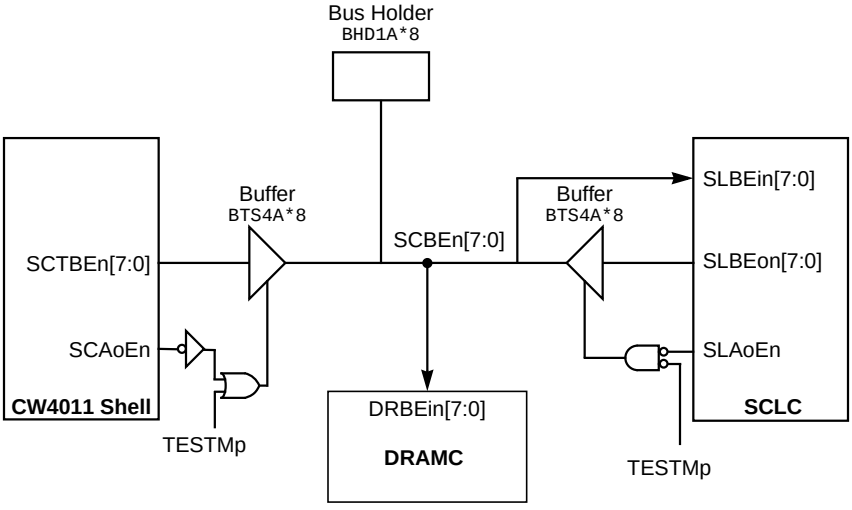


Figure 5.7 shows an example of a buffer configuration in which the SC byte enable signals, SCTBEn[7:0], are buffered at the SCLC and CW4011 ends by BTS4A*8 3-state buffers. When the CW4011 asserts SCAoEn, the signal enables the buffer at the CW4011 end. When the SCLC asserts SLAoEn, the signal enables the buffer at the SCLC end.

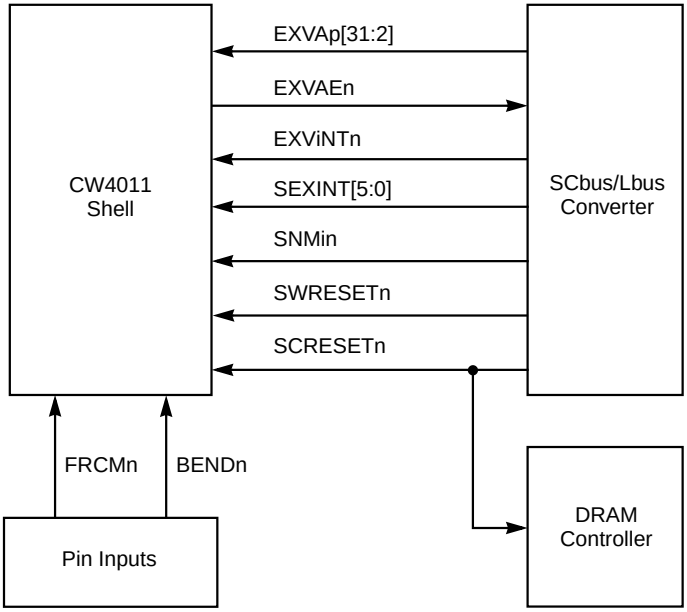
Figure 5.7 Buffering for SCBEn[7:0] Byte Enable



5.2.3 CW4011 Shell Reset/Interrupt Interface

Figure 5.8 shows the internal interface that links the CW4011 shell, the SCLC, and the DRAMC to provide the reset and interrupt signals required by CP0.

Figure 5.8 Shell Reset/Interrupt Interface



BENDn **Big Endian (Strap Input)** **Input to Shell**
This input affects the byte positions for sizing and load/store data alignment. When the input is LOW (asserted), the CW4011 works with big endian byte ordering. When HIGH, byte ordering is little endian.

EXVAp[31:2] **External Vectored Interrupt Address**
Input to Shell from SCLC
The CW4011 shell accepts the external vectored interrupt address when the SCLC asserts EXVApEn. The CW4011 writes the address directly into the program counter. The address bus must remain stable until EXVApEn is asserted.

EXVApEn **EXVAp Enable** **Output from Shell to SCLC**
This is the enable signal for the vectored interrupt address. The CW4011 asserts this signal to acknowledge the address.

EXViNTn **External Vectored Input** **Input to Shell from SCLC**
The SCLC drives this signal. When the CW4011 shell receives the signal, it generates an external vectored interrupt exception.

FRCMn **Force Cache Miss (Strap Input)** **Input to Shell**
This input is used for system debug. Under normal operating conditions, you should deassert FRCMn by strapping it HIGH. To use it for debug, you should assert it by tying it LOW. When LOW, the signal forces a cache miss for the I-cache and the D-cache in the CW4011 shell. The CW4011 treats this event as an access to an uncached area. The CW4011 can then read and write all instructions and data as uncached, regardless of the memory segment and the MMU.

SEXiNTn[5:0] **External Interrupt Input Signals [5:0]** **Input to Shell**
The SCLC asserts one of the SEXiNT signals to cause the CP0 in the CW4011 core to generate an interrupt exception. The assertion is registered in the IP field of the CW4011 Cause Register. The SCLC should continue to assert the signals until the exception routine has serviced the interrupt.

The CW4011 does not recognize interrupts if the interrupt enable bit in the Status Register is not set. The CW4011 can therefore disable individual interrupt inputs by clearing the related bits. However, the interrupt inputs are still registered in the IP field of the Cause Register.

External Interrupt Input signals [5:0] are synchronized to the system clock, SCLKp, in the SCLC.

The *MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual* provides information about the CW4011 CP0 registers.

SCRESETn **Cold Reset** **Input to Shell**

This input initiates a cold reset for the LR4500. Inside the LR4500, this signal is synchronized to the system clock, SCLKp. The LR4500 enters the cold reset condition when the SCLC asserts SCRESETn. CP0 initiates a cold reset exception when the SCLC deasserts SCRESETn.

SNMin **Nonmaskable Interrupt** **Input to Shell**

This input is synchronized in the SCLC to the system clock, SCLKp. When the SCLC asserts this signal, the CW4011 recognizes a nonmaskable interrupt. The CP0 then generates a nonmaskable interrupt exception (0xBFC0 0000).

SWRESETn **Warm Reset** **Input to Shell**

This input initiates a warm reset for the LR4500. This signal is synchronized to the system clock, SCLKp inside the LR4500. The LR4500 enters the warm reset condition when the SCLC asserts SWRESETn. CP0 initiates a cold reset exception when the SCLC deasserts SWRESETn.

Chapter 6
DRAM Controller and Memory Bus

This chapter describes the synchronous DRAM Controller and the memory bus. It defines:

- DRAM types compatible with the LR4500 on page 6-1
- Address space available for the DRAM on page 6-1
- Memory interface on page 6-2
- Memory address bit assignment on page 6-4
- DRAM Controller Configuration Register on page 6-5
- DRAM Mode Register on page 6-10
- DRAM refresh requirements, and the DRAM Controller Refresh Register and Refresh counter on page 6-12
- DRAM commands on page 6-14
- Initializing the DRAM on page 6-15

6.1 DRAM Types and Available DRAM Address Area

The LR4500 Microprocessor reference device interfaces directly to synchronous DRAMs, without any glue logic, through a 64-bit memory data bus. When the DRAM is arranged in two banks, the chip select signals, MCSn[1:0], select between the two banks.

Table 6.1 shows different DRAM configurations and the address ranges assigned to the memory banks. There is no programmable feature that defines the DRAM size and configuration. The utility setup/bootstrap

program should check the amount of installed DRAM when the system is initially powered up.

Table 6.1 DRAM Configurations

DRAM Type	Number of Banks	Number of DRAMs	Memory Size	Bank 0 Address Range	Bank 1 Address Range
1 M x 16	1	4	8 Mbyte	0x0000 0000 – 0x007F FFFF	None
1 M x 16	2	8	16 Mbyte	0x0000 0000 – 0x007F FFFF	0x0200 0000 – 0x027F FFFF
2 M x 8	1	8	16 Mbyte	0x0000 0000 – 0x00FF FFFF	None
2 M x 8	2	16	32 Mbyte	0x0000 0000 – 0x00FF FFFF	0x0200 0000 – 0x02FF FFFF
4 M x 4	1	16	32 Mbyte	0x0000 0000 – 0x01FF FFFF	None
4 M x 4	2	32	64 Mbyte	0x0000 0000 – 0x01FF FFFF	0x0200 0000 – 0x03FF FFFF

6.2 Memory Interface

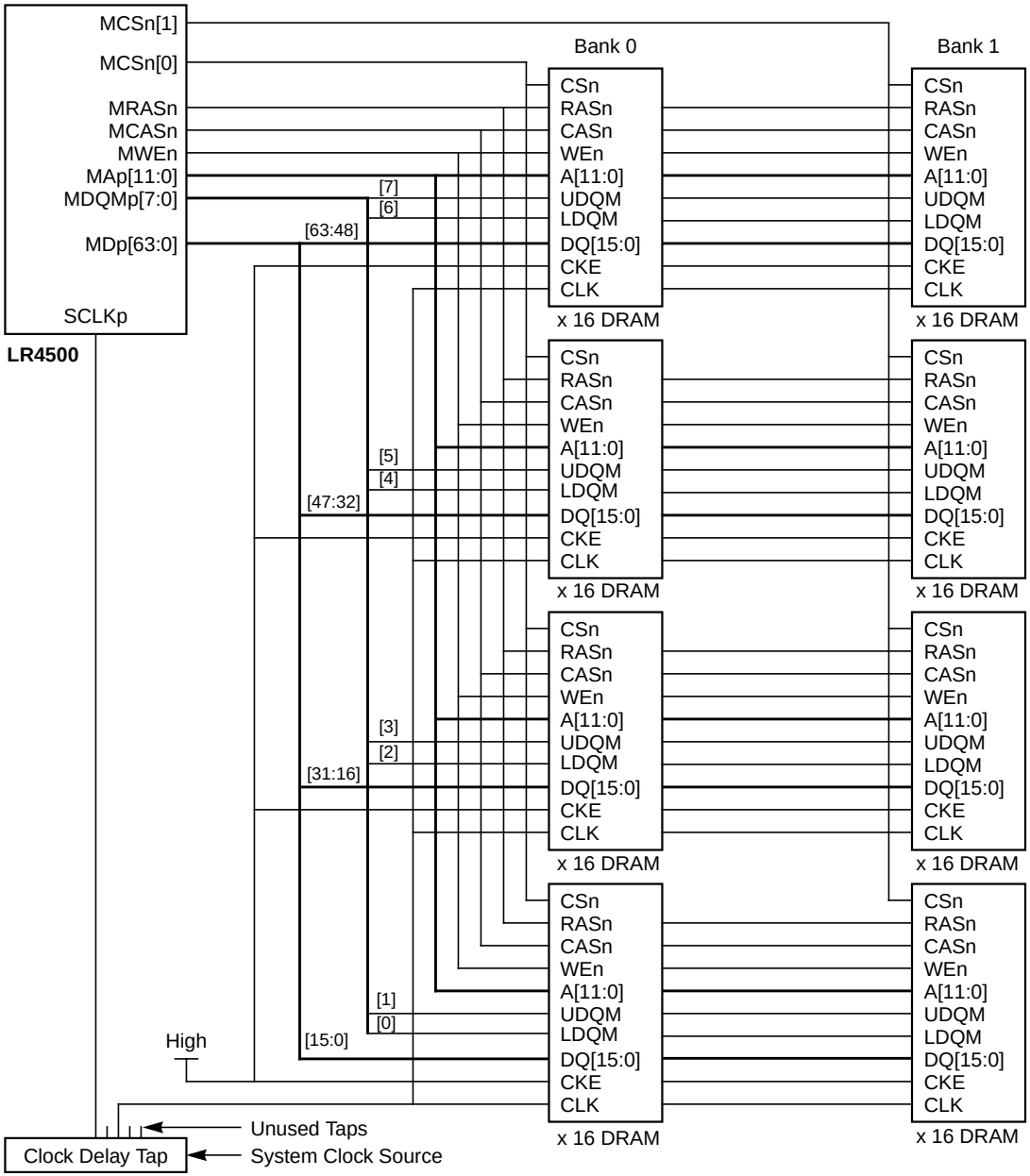
Figure 6.1 shows the interface between the LR4500 Mbus and the DRAMs. In the example shown, eight DRAM devices with a 16-bit data bus are arranged in two memory banks, providing 16 Mbytes of memory. This is the configuration shown in line 2 of Table 6.1. Note that this configuration does not have continuous memory space.

A clock-delay tap provides the clock input for the DRAMs. The clock enable (CKE) inputs to the DRAMs are tied HIGH, which means that they are always asserted.

The LR4500 selects between Bank 0 and Bank 1 of the DRAM by means of the chip select signals, MCSn[1:0]. It asserts MCSn[0] to select the four DRAMs in Bank 0, and MCSn[1] to select the four DRAMs in Bank 1. The LR4500 distributes address (MAp[11:0]), row address strobe and column address strobe (MRAS and MCAS), and the write enable signal (MWEn) to all DRAMs. Data (MDp[63:0]) and the data mask

(MDQMp[7:0]) are distributed to each byte in the DRAM array, with MDQMp[7] masking byte 7 (bits [63:56]), and so forth.

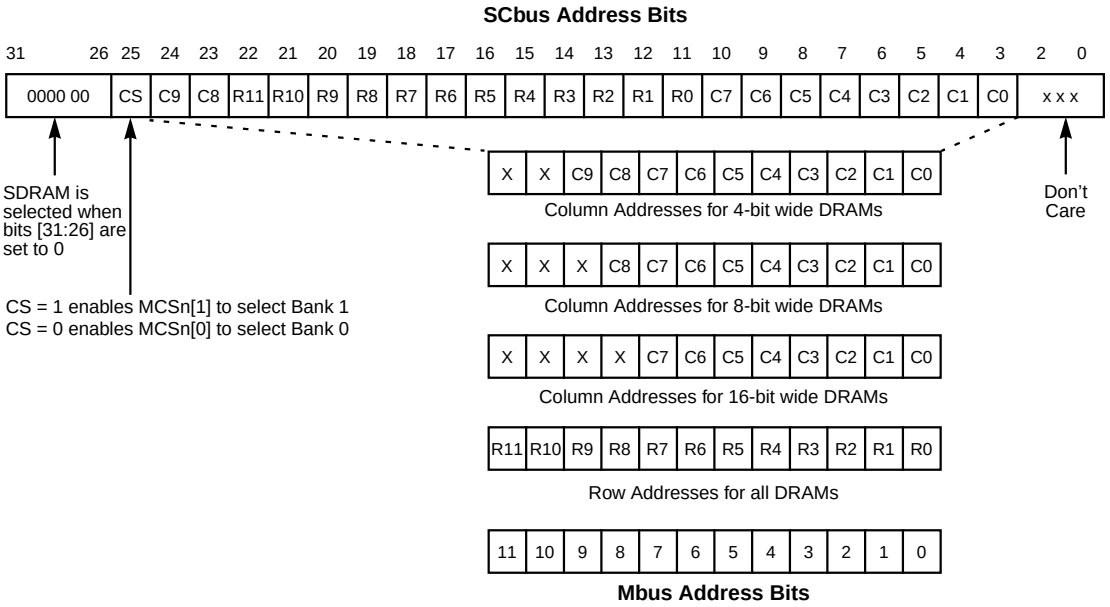
Figure 6.1 LR4500 Interface with DRAM



6.3 Address Bit Assignment

The DRAM Controller in the LR4500 derives the DRAM addresses, MAp[11:0], from the 32 SCbus address bits output by the bus master, which may be the CW4011 core or the SCLC module. The controller outputs the address bits on the Mbus, assigning the bits as shown in Figure 6.2. The byte select signals, MDQMp[7:0], are derived directly from the byte enable signals, SCTBEn[7:0].

Figure 6.2 SCbus DRAM Address Bit Assignment



X indicates that the bit is not used in this configuration
x indicates a 'don't care' situation

The MAp[11:0] 12-bit address bus multiplexes row and column addresses. Table 6.2 lists the SCbus address and Mbus address bit assignments.

Table 6.2 SCbus Address and Mbus Address Bit Assignment

SCbus Address Bits	Address Bit Function	Mbus Address Bits
SCTBEn[7:0]	Byte selection during write operations	MDQMp[7:0]
SCAp[10:3][1]	Column addresses C[7:0]	MAp[7:0]
SCAp[22:12]	Row addresses R[11:0]	MAp[11:0]
SCAp[24:23]	Column addresses C[9:8]	MAp[9:8]
SCAp[25]	Chip selection	MCSn[1:0]
SCAp[31:26][2]	All zeros for DRAM access	Activate DRAM access

1. These bits are used for column addresses in 16-bit wide DRAMs. They are used in conjunction with SC[23] for 8-bit wide DRAMs, and with SC[24:23] for 4-bit wide DRAMs. Since SC[10:3] supply column addresses, the DRAM page size is 1 Kbyte for all DRAMs.
2. To access the DRAM, SC[31:26] must be set to 0. Otherwise, the DRAM controller will not respond to an SCbus transaction.

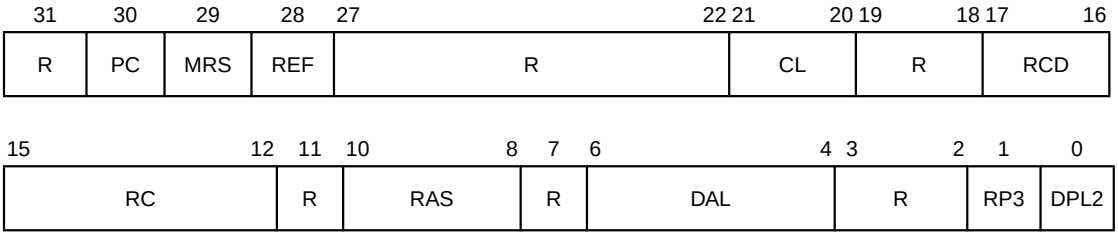
6.4 DRAM Controller Configuration Register

The DRAM Controller Configuration Register allows you to configure various features of the DRAM. The virtual and physical addresses for the register are shown below:

Virtual Address	Physical Address
0x B000 0000	0x 1000 0000

Figure 6.3 shows the format recommended for the DRAM Controller Configuration Register.

Figure 6.3 DRAM Controller Configuration Register Format



R **Reserved** **31 [27:22] [19:18] 11, 7 [3:2]**
These bits are not used. They should be cleared to 0.

PC **Precharge Command** **30**
This bit enables the manual precharge command. If the CPU sets the bit to 1, the DRAM Controller generates one precharge command cycle for both banks. The CPU sets the bit at power up. Initialization clears the bit automatically.

MRS **Mode Register Set** **29**
If the CPU sets the bit to 1, the subsequent Store Word operation to the DRAM area generates a Mode Register Set command. The row address bits in the SC address bus (SCAp[22:11]) select the addressed location during this type of operation. SCAp[31:23] and SCAp[10:0], which are the mode bits, must be set to 0. The CPU clears the MRS bit when the word operation has been completed.

REF **Refresh Cycle** **28**
This bit enables the manual refresh cycle request (REF). If the CPU sets it to 1, one refresh cycle is generated for both memory banks. This bit is cleared automatically when the refresh cycle has been completed. You can also generate REF using the refresh counter, as described in “DRAM Refresh” on page 6-12.

CL **CAS Latency** **[21:20]**
You can set CAS Latency by programming the bits in this field. You should select one of the following settings:

Bit 21	Bit 20	Cache Latency Modes
0	1	1
1	0	2
1	1	3

Although you can define all DRAM timing parameters independently, CAS Latency defines the relationship between other timing parameters. Table 6.3 on page 6-10 shows the relationships between CAS Latency, DRAM frequency, and other configuration settings.

Bits [6:4] in the DRAM Mode Register also select the cycle modes. You must set or clear the bits in both registers, as described in CL CAS Latency [6:4] on page 6-11, to select the required mode.

RCD **Active RAS to Read/Write Command Period Cycles [17:16]**

You can program the bits in this field to select the number of active clock cycles for a read or write operation. You can select one of the following settings:

Bit 17	Bit 16	Active Clock Cycles
0	1	1
1	0	2
1	1	3

RC **Refresh to Refresh/Active Command Period Cycles [15:12]**

This field allows you to select the number of active read/write cycles between refresh cycles. You can program these bits as follows:

Bit 15	Bit 14	Bit 13	Bit 12	Active Read/ Write Cycles
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10

RAS **Active to Precharge Command Period [10:8]**

This field allows you to select the number of clock cycles that RAS should stay active until the memory has been precharged. You can program the bits as follows:

Bit 10	Bit 9	Bit 8	Active RAS Cycles
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

DAL **Data In to Active/Refresh Command Period [6:4]**

This field allows you to select the number of active clock cycles between the time data input is valid until the refresh command is asserted. You can set the field as follows:

Bit 6	Bit 5	Bit 4	Active DAL Cycles
0	1	1	3
1	0	0	4
1	0	1	5

RP3 **Precharge to Active Command Period 1**

This bit allows you to select the number of active clock cycles in the period between precharge and an active read or write command. If you set the bit to 1, there are three clock cycles. If you clear the bit to 0, there are two clock cycles.

DPL2 **Data In to Precharge Command Period 0**

This bit allows you to select the number of active clock cycles in the period between the input of valid data to the assertion of the precharge command. If you set the bit to 1, there are two clock cycles. If you clear the bit to 0, there is one clock cycle.

The relationship between latency and frequency varies, depending on the DRAM specification. Table 6.3 shows an example of the timing parameters for three NEC DRAMS—uPD4516821, uPD4516421, and uPD4516161 DRAMs. For the fastest access time, you should use a

DRAM with a maximum clock frequency of 10 ns. Refer to the NEC user's manual for further information.

Table 6.3 Relationship Between Frequency and Latency

Clock Frequencies			Register Bit Settings						
(-10)[1]	(-12)[2]	(-13)[3]	CL	RCD	RC	RAS	DAL	RP	DPL
100 MHz	83 MHz	75 MHz	3	3	10	7	5	3	2
80 MHz	-	-	3	3	8	6	5	3	2
66 MHz	55 MHz	50 MHz	2	2	7	5	3	2	1
33 MHz	27 MHz	25 MHz	1	1	4	3	2(3)[4]	1(2)[5]	1

1. Maximum clock frequency for the 10 ns version of the DRAM.
2. Maximum clock frequency for the 12 ns version of the DRAM.
3. Maximum clock frequency for the 13 ns version of the DRAM.
4. NEC recommends 2 clock cycles for DAL when CL is set to 1. However, the LR4500 DRAM Controller requires 3 clock cycles for DAL.
5. NEC recommends 1 clock cycle for RP when CL is set to 1. However, the LR4500 DRAM controller requires 2 clock cycles for RP.

6.5 DRAM Mode Register

The LR4500 supports a number of programmable modes by means of the DRAM Mode Register. The modes you can program are:

- Cache Write Through and Write Back

  Cache write through mode allows all data that is updated in the cache to be updated at the same time in external memory. In cache write back mode, main memory is only updated when the cache line is reallocated or is written back using the write back instruction. This mode does not apply in the LR4500, since the Burst Length field is set to 0.

- Burst Length

  Defines the number of words to be output or input during read and write cycles. In the LR4500, the Burst Length field is set to 0.

- Wrap Type

  Specifies the order in which burst data is addressed. This mode does not apply in the LR4500, since the Burst Length field is set to 0.

- CAS Latency

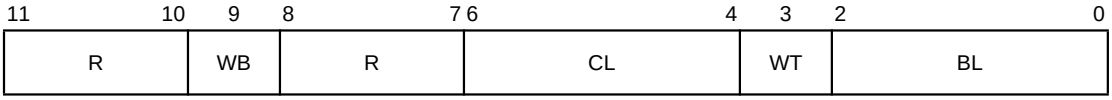
  Defines the number of clock cycles that must occur before data is available.

- Auto Precharge

  This mode is not used in the LR4500.

Figure 6.4 shows the format of the 12-bit DRAM Mode Register. This register is programmed by a mode write command and is located in an SDRAM.

Figure 6.4 DRAM Mode Register Format



When you power up the DRAM, the boot program precharges the DRAM devices. You should refer to the documentation supplied with the DRAM for further information on precharging. After precharge, you must set the MRS bit in the DRAM Controller Configuration Register (see page 6-6).

R **Reserved** **[11:10] [8:7]**
These bits are not used. They are set to 0.

WB **Cache Write Through and Write Back** **9**
You can select cache write through mode by setting this bit to 1. Setting the bit to 0 selects write back mode. However, there is no significance to write through and write back modes in LR4500 transactions, since the burst length is one word.

CL **CAS Latency** **[6:4]**
You can select among one-, two-, and three-cycle modes by programming bits [6:4]. You should select one of the following settings. All other settings are reserved.

Bit 6	Bit 5	Bit 4	Cache Latency Modes
0	0	1	1
0	1	0	2
0	1	1	3

Bits [21:20] in the DRAM Controller Configuration Register also select the cycle modes. You must set or clear the bits in both registers, as described in CL CAS Latency [21:20] on page 6-6, to select the required mode.

WT **Wrap Type** **3**

Sequential mode is compatible with SCbus burst ordering. Since burst length is one word for the LR4500, wrap type is not used in the LR4500, so you should clear this bit to 0 to enable sequential accesses. Setting the bit to 1 enables interleaved accesses.

BL **Burst Length** **[2:0]**

You can select single-cycle mode by clearing bits [2:0] of this register to 0b000. The SCbus requests four double-words as a burst block. With a data bus width of 64 bits, the LR4500 supports the request with multiple CAS accesses.

6.6 DRAM Refresh

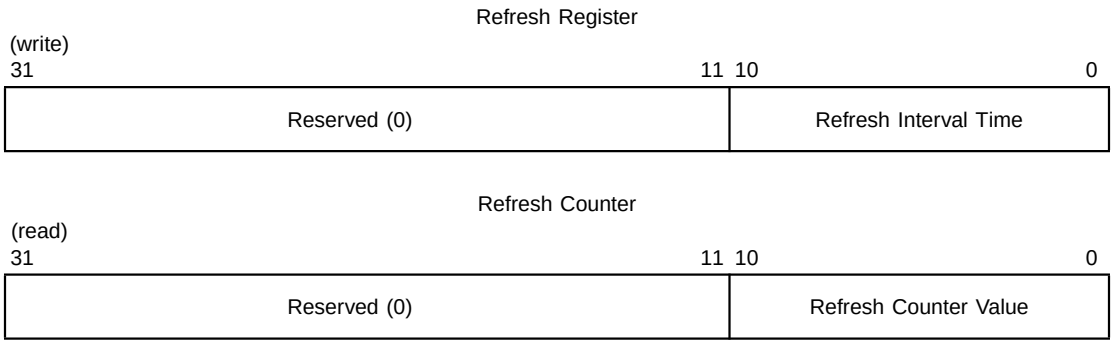
The DRAM Controller needs to refresh the 2048 rows in the synchronous DRAM every 32 milliseconds. The controller also needs to set up a 15,625 ns (15.625 µs) refresh interval. For example, if the maximum clock frequency is 66 MHz, the controller must issue a DRAM refresh command every 1,041 clock cycles.

The DRAM Controller has an 11-bit refresh interval timer that generates the refresh command. The refresh interval timer, shown in Figure 6.5, consists of one 11-bit register, referred to as the Refresh Register, which stores the refresh interval time; and one 11-bit binary countdown register, referred to as the Refresh Counter, which stores the refresh counter value, and is decremented by each system clock input. The Refresh Register address is shown below.

Virtual Address	Physical Address
0x B000 0004	0x 1000 0004

When the system is initialized, the DRAM Controller writes the Refresh Interval Time data into the Refresh Register. The same data is stored in the Refresh Counter as the Refresh Counter Value. The DRAM Controller reads the contents of both registers only during testing.

Figure 6.5 DRAM Refresh Interval Timer



After a cold reset, the counter stops counting. Once the DRAM Controller has written the value for the Refresh Interval Time into the Refresh Register, the counter loads the same initial value and starts counting by decrementing the initial value by 1 at each clock input. When the counter has counted down to 1, the DRAM Controller sets the REF bit in the LR4500 Configuration and Cache Control Register (page 3-5), requesting a refresh command. The initial value is then reloaded and the process starts again. Note that the counter never counts down to 0. If a DRAM transaction is proceeding when the DRAM Controller issues the refresh command, the status of the refresh command is 'pending,' and a refresh command cycle is generated when the preceding transaction has been completed. Only a cold reset can stop the refresh counter.

The setting of the Refresh Register is derived from the DRAM clock cycle value, the required refresh interval (15,625 ns), and the CAS latency (CL) setting. The CL setting determines the number of clock cycles required before data is available. Table 6.4 lists Refresh Register programming values for four operating frequencies. You can calculate value A by dividing the refresh interval by the microprocessor's clock cycle time. You can calculate the value programmed into the Refresh Register by subtracting the number of clock cycles required for the longest transaction, which is a burst read transaction (B), from the value A. In the first example shown, the Refresh Register should be set to 1551 (0x61A).

Table 6.4 Refresh Register Programming Values

Clock Frequency	Clock Cycle Time	Value (A)[1]	Number of Clock Cycles Required (B)[2]	Refresh Register Programmed Value[3] Decimal (Hex)	CL Setting
100 MHz	10 ns	1563	12	1551 (0x61A)	3
80 MHz	12.5 ns	1250	12	1238 (0x406)	3
66 MHz	15 ns	1041	10	1031 (0x407)	2
50 MHz	20 ns	781	8	773 (0x305)	1

1. Value A is derived from the required refresh interval time (15,625 ns) divided by the clock cycle time (12.5 ns, and so forth).
2. Number of clock cycles required for a burst read transaction.
3. The Refresh Register programmed values is derived from Value A minus Value B.

Table 6.5 shows how the Refresh Register settings are arrived at. The example shown in the table is for the 80 MHz, 12.5 ns DRAM.

Table 6.5 Refresh Register Setting for 80 MHz 12.5 ns DRAM

Register Bits	31	30	29	28	27-11	10	9	8	7	6	5	4	3	2	1	0
Binary Setting	0	0	0	0	x	1	0	0	1	1	0	1	0	1	1	0
Hex Value	0				x	4			0				6			
Decimal Value	Not used					1 2 3 8										

6.7 DRAM Commands

This section describes the DRAM commands used by the LR4500 DRAM Controller. These commands are the chip select commands (MCSn[1:0]), row and column addresses strobes (RASn and CASn), and the write enable command (MWEn). The DRAM Controller does not use the DRAM's Self-Refresh Entry Command and Burst Stop Command. In addition, for a No Operation (Nop), the DRAM Controller deasserts the chip select outputs MCSn[1:0] and the other control signals.

Table 6.6 summarizes the settings of the Mbus control signals and the DRAM commands they generate.

- SCAp[31:0] indicates the SCbus address bit n associated with the Memory bus address bit, MAp[31:0]
- ~ indicates an inverted signal
- ( ) indicates a don't care condition, but one in which the signals are output.

For detailed information about DRAM commands, refer to the datasheet supplied with the SDRAM.

Table 6.6 Summary of DRAM Commands and Mbus Control Signals

Command	MCSn[1]	MCSn[0]	MRASn	MCASn	MWEn	MAp[11]	MAp[10]	MAp[9:0]
No operation[1]	High	High	High	High	High	(SCA22)	(SCA21)	(SCA20:11)
Mode Register set	Low	Low	Low	Low	Low	SCA22	SCA21	SCA[20:11]
Row active	~SCA25	SCA25	Low	High	High	SCA22	SCA21	SCA[20:11]
Precharge[2]	Low	Low	Low	High	Low	SCA22	High	SCA[20:11]
Write[3]	~SCA25	SCA25	High	Low	Low	SCA22	Low	SCA24,23,[10:3]
Read[3]	~SCA25	SCA25	High	Low	High	SCA22	Low	SCA24,23,[10:3]
CAS before RAS refresh	Low	Low	Low	Low	High	SCA22	SCA21	SCA[20:11]

1. MCSn[1:0] must both be kept high for no-operation conditions
2. Both banks are always precharged
3. When Write or Read commands are sent for a burst transaction, MAp[1:0] are incremented by the order of wrap around, starting from the requested address, for example, 01, 10, 11, then 00

6.8 Initializing DRAM and Programming the Mode Register

Before the DRAM Controller can access the DRAM for a normal read or write transaction, the boot program must initialize the DRAM through the DRAM Controller. After power on, the DRAM Controller goes through the following initialization process:

- Precharges the DRAM
- Programs the DRAM's Mode Register
- Refreshes the DRAM array twice

The CPU can initiate this process by:

1. Programming the DRAM Configuration Register with the PC, MRS, and REF bits set to 1.
2. Programming the DRAM Mode Register by initiating a store word operation to one of the addresses shown below:

CAS Latency (CL)	Physical Address	Virtual Address
1	0x0010 8000	0xA010 8000
2	0x0011 0000	0xA011 0000
3	0x0011 8000	0xA011 8000

   The address value is programmed to the SDRAM. Write data is ignored.

3. Programming the DRAM Refresh Register.

Once the DRAM Controller has initialized the DRAM, it can initiate various types of DRAM accesses.

Figure 6.6 shows the timing requirements for the DRAM initialization sequence. Table 6.7 lists the signals referenced in Figure 6.6 and in subsequent timing diagrams. The signals are arranged in alphabetical order.

Table 6.7 Timing Signals

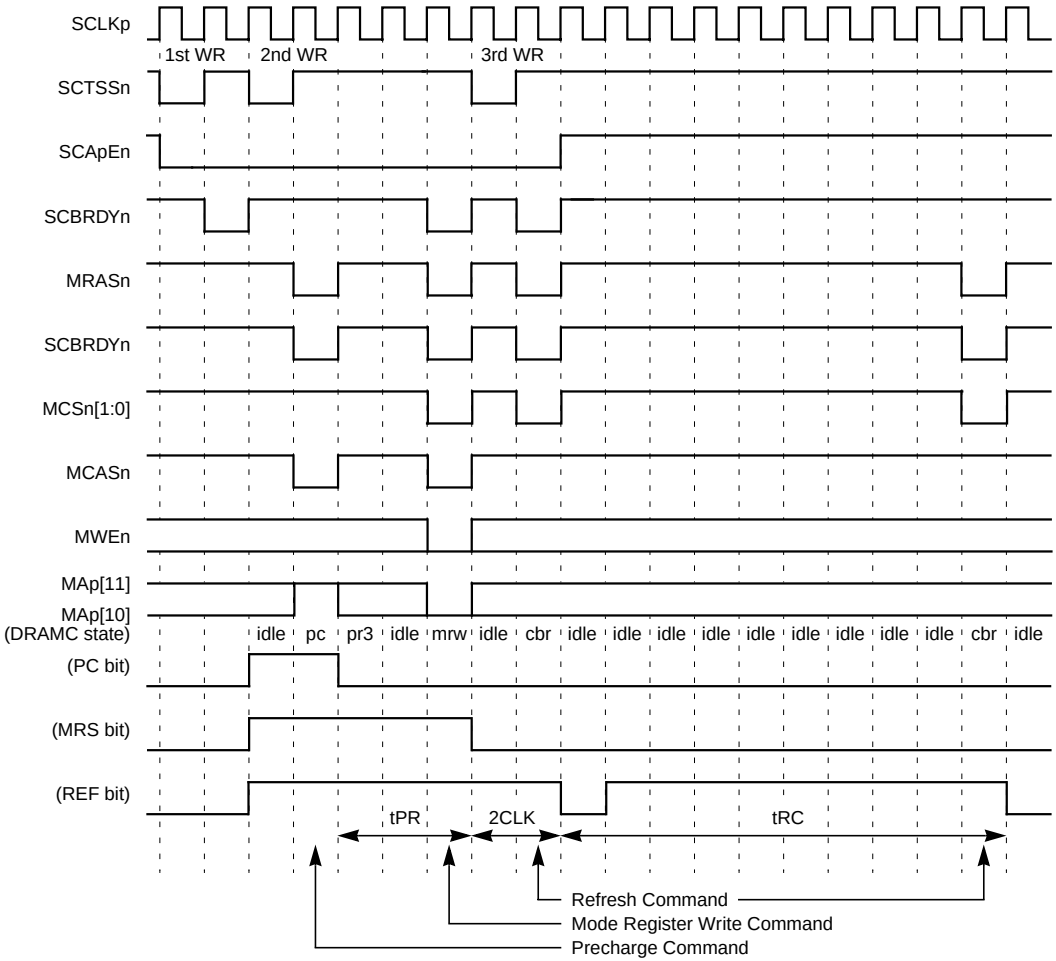
Signal Name	Description	Other References
AoEREQp (internal)	Address Output Enable Request	
DCiNVSn	D-cache Invalidation Strobe	See *MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual.*
DoEREQp (internal)	Data Output Enable Request	
DRAMC state	State of the DRAM Controller	
DRRDY	Data Ready	
ICiNVSn	I-cache Invalidation Strobe	See *MiniRISC CW4011 Superscalar Microprocessor Core Technical Manual.*
LADSn	Lbus Address Strobe	Signals with an L prefix are Lbus signals. You will find more detailed information about these signals in "Lbus Interface" on page 5-5.
LA(o)p	Lbus Address (Output from LR4500)	
LAoEn	Lbus Address Enable	
LBEn	Lbus Byte Enable	
LCLKp	Lbus Clock	
LDp	Lbus Data	
LDip	Lbus Write Data	
LDop	Lbus Read Data	
LDoEn	Lbus Data Output Enable	
LHLDAp	Lbus Hold Acknowledge	
LHoLDp	Lbus Hold Request	
LRD(o)n	Lbus Data (Output to LR4500)	
LRDYn	Lbus Data Ready	
LRDYoEn	Lbus Data Ready Output Enable	
LSLRDYin (internal)	Lbus Sampled Ready	

(Sheet 1 of 2)

Table 6.7 Timing Signals (Cont.)

Signal Name	Description	Other References
MAp	Memory Address	Signals with an M prefix are Mbus (memory bus) signals. You will find more detailed information about these signals in "Mbus Interface" on page 5-2.
MCASn	Memory Column Address Strobe	
MCSn	Memory Chip Select	
MDp	Memory Data	
MDQMp	Memory Data Enable/Mask	
MRASn	Memory Row Address Strobe	
MWEn	Memory Write Enable	
SCAp	SCbus Address	Signals with an SC prefix are CW4011 core SCbus signals. You will find more detailed information about these signals in the LSI Logic Technical Manual *MiniRISC CW4011 Superscalar Microprocessor Core.*
SCBEn (SCTBEn)	SCbus Enable	
SCBRDYn	SCbus Ready	
SCDp	SCbus Data	
SCDoEn	SCbus Data Output Enable	
SCHGTn	SCbus Hold Grant	
SCHRQn	SCbus Hold Request	
SCLKp	System Clock	
SCTBSTn	SCbus Burst Transaction	
SCTPWn	SCbus Next Transaction is in Write Page	
SCTSEn	SCbus Transaction Start Enable	
SCTSSn	SCbus Transaction Start Strobe	
SLDoEn	SCLC SCbus Data Output Enable	
Bit Names		
MRS	Mode Register Set	page 6-6
PC	Precharge Command	page 6-6
REF	Refresh	Cycle
(Sheet 2 of 2)		

Figure 6.6 Timing Requirements for DRAM Initialization Sequence



Notes:

- 1st WR is the write to the DRAM Configuration Register.
- 2nd WR is the write to the DRAM Mode Register.
- 3rd WR is the write to the DRAM Refresh Register.
- tPR = 3, tRC = 10.

6.9 DRAM Transactions

Once the DRAM Controller has initialized the DRAM, it can initiate various types of DRAM accesses. This section shows the timing requirements for three typical DRAM accesses:

- Figure 6.7 shows the timing for a single burst read transaction.
- Figure 6.8 (page 6-22) shows the timing for two continuous write transactions.
- Figure 6.9 (page 6-23) shows the timing for a burst write transaction.

In all cases, the DRAM is an 80 MHz, 10 ns device. Other timing parameters for this device are as follows:

- CAS Latency (CL) = 3. (Refer to page 6-6 for further information about CL.)
- Active RAS to Read/Write Command Period Cycles (RCD) = 3. (Refer to page 6-7 for further information about RCD.)
- Refresh to Refresh/Active Command Period Cycles (RC) = 8. (Refer to page 6-7 for further information about RC.)
- Active to Precharge Command Period (RAS) = 6. (Refer to page 6-8 for further information about RAS.)
- Precharge to Active Command Period (RP) = 3. (Refer to page 6-8 for further information about RP.)
- Data In to Precharge Command Period (DPL) = 2. (Refer to page 6-8 for further information about DPL.)
- Data In to Active/Refresh Command Period (DAL) = 5. (Refer to page 6-8 for further information about DAL.)

Figure 6.7 Single Burst Read Transaction

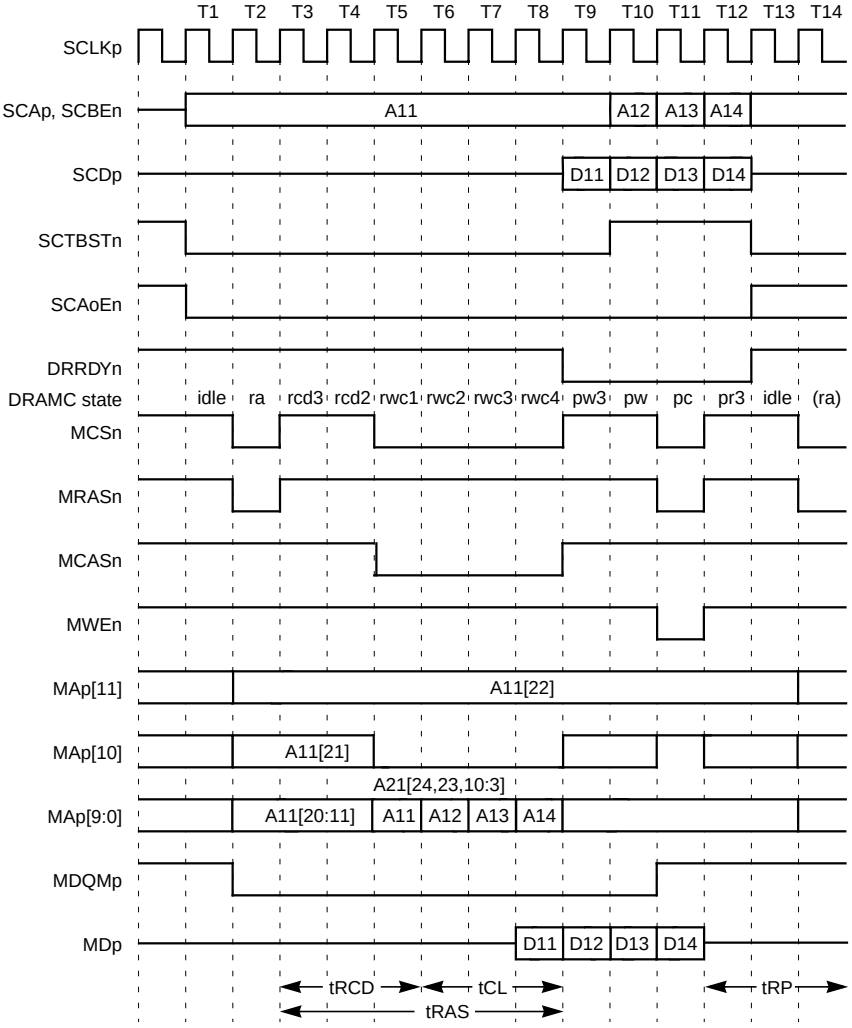


Figure 6.8 Two Continuous Single Write Transactions

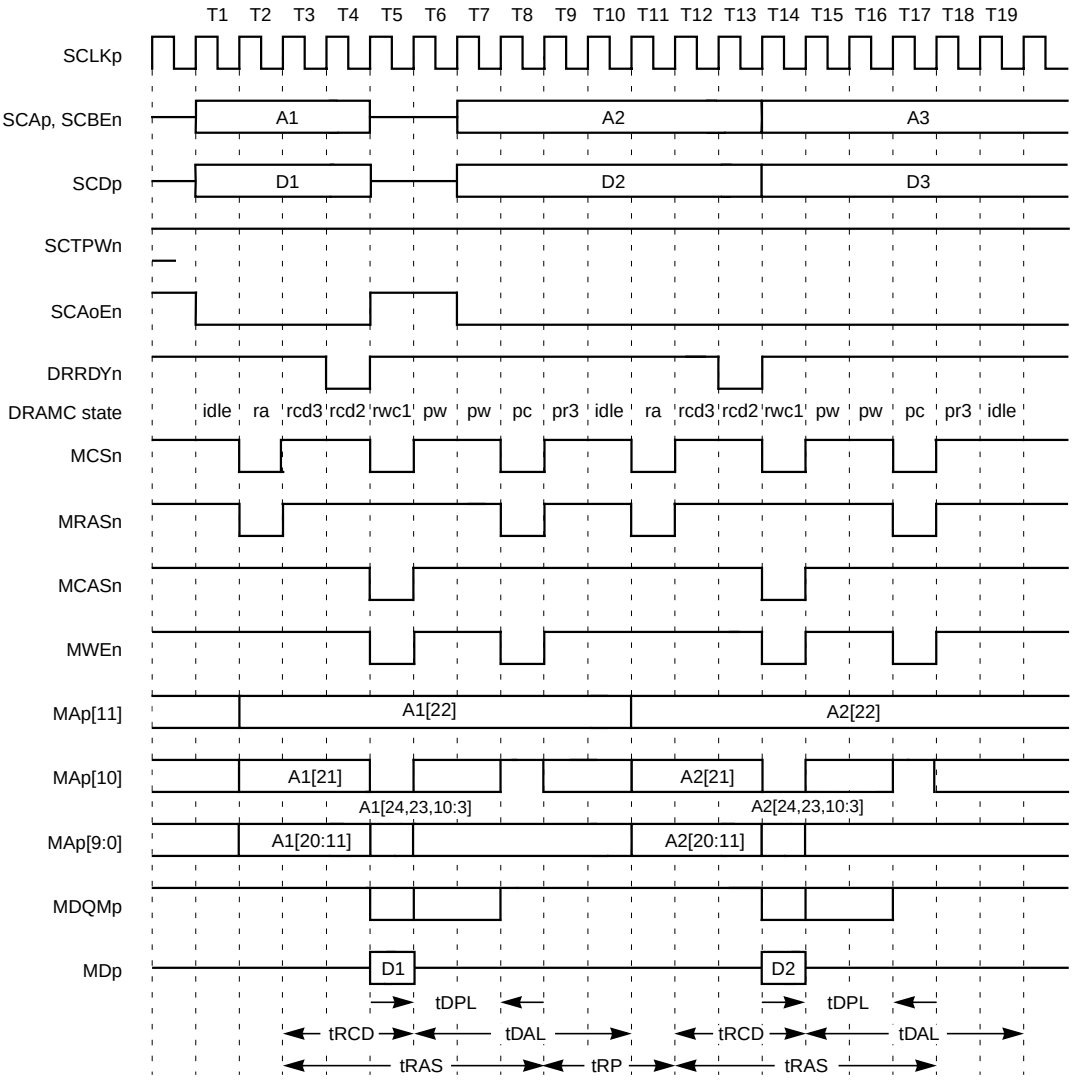
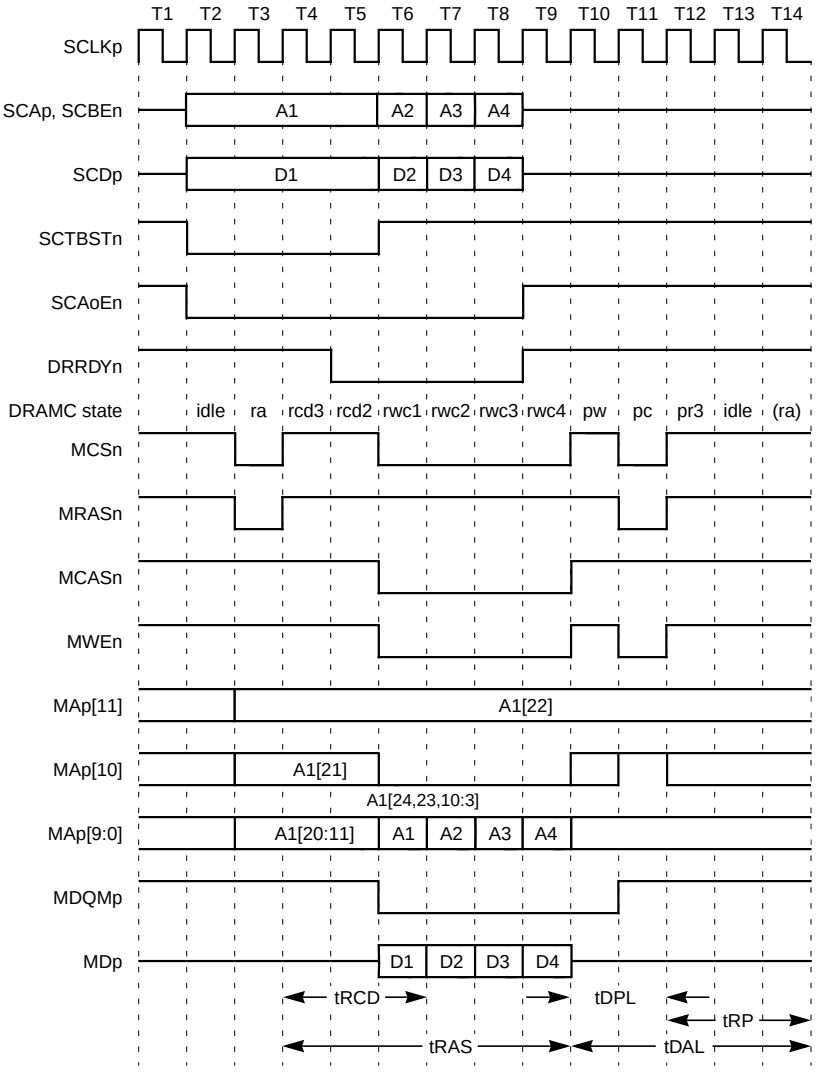


Figure 6.9 Burst Write Transaction



Chapter 7
SCbus and Local I/O Bus Converter Module

This chapter discusses the Lbus, and describes how the LR4500 Microprocessor reference device interacts with the Lbus through the SCLC module. The chapter contains the following sections:

7.1 Lbus Features

The Lbus is similar to the VLbus or 486 bus, which has a demultiplexed 32-bit address bus and a 32-bit data bus. "Lbus Interface" on page 5-5 provides a list of Lbus signals.

There are certain differences between the Lbus and the VLbus, as shown below:

Feature	Lbus	VLbus
I/O space	No	Yes
Interrupt acknowledge cycle	No	Yes
Support for single transactions	Yes	Yes
Support for burst transactions	No	Yes
HOLD/HLDA bus arbitration	Yes	Yes
Bus retry input	Yes	Yes
Uses Lbus clock, LCLKp	Yes	Yes

The Lbus is synchronized by the Lbus clock, LCLKp, which is derived from the CW4011 system clock, SCLKp. The LR4500 outputs the LCLKp to the Lbus.

The LR4500 can function as the Lbus master or the Lbus slave. If the LR4500 is master, it starts an Lbus transaction while LHLDAp is deasserted. If an Lbus device wants to control the Lbus and initiate a bus transaction, it must first take ownership of the bus by issuing a bus hold request (by asserting LHoLDp) to the LR4500. The LR4500 returns a bus hold acknowledge signal (by asserting LHLDAp) to the Lbus device, granting bus ownership. When this occurs, the Lbus device may initiate Lbus transactions.

The Lbus master starts a transaction on the Lbus by asserting the Address Strobe, (LADSn). At this time, the master must also drive valid information on the address bus and the byte enable lines. The Lbus master uses LRDn signal to control the direction of the data transfer. The master must present the appropriate level on this signal at the same time it asserts strobe signal LADSn. During a write transaction, the master must also drive valid data on the data bus.

When the transaction has been successfully completed, the selected slave device asserts LRDYn, indicating that the Lbus is ready for another transaction. The master must continue to drive all signals until it samples LRDYn. If the transaction is a read transaction, the slave device must place valid data on the bus before it asserts LRDYn.

7.2 LR4500 as Master on the Lbus

The LR4500 is the master of the Lbus when the CW4011 accesses an address in the Lbus area located in the physical address range 0x1100 0000 through 0xFFFF FFFF. The Lbus device must assert a data ready or bus retry signal and input it to the LR4500 within 256 SCLKp cycles. Otherwise, the SCbus watchdog timer terminates the SCbus transaction by asserting a bus error signal. Figure 7.1 shows the timing requirements for an Lbus read transaction generated by the CW4011 core.

Figure 7.1 Timing Requirements for an SCbus-to-Lbus Read Transaction

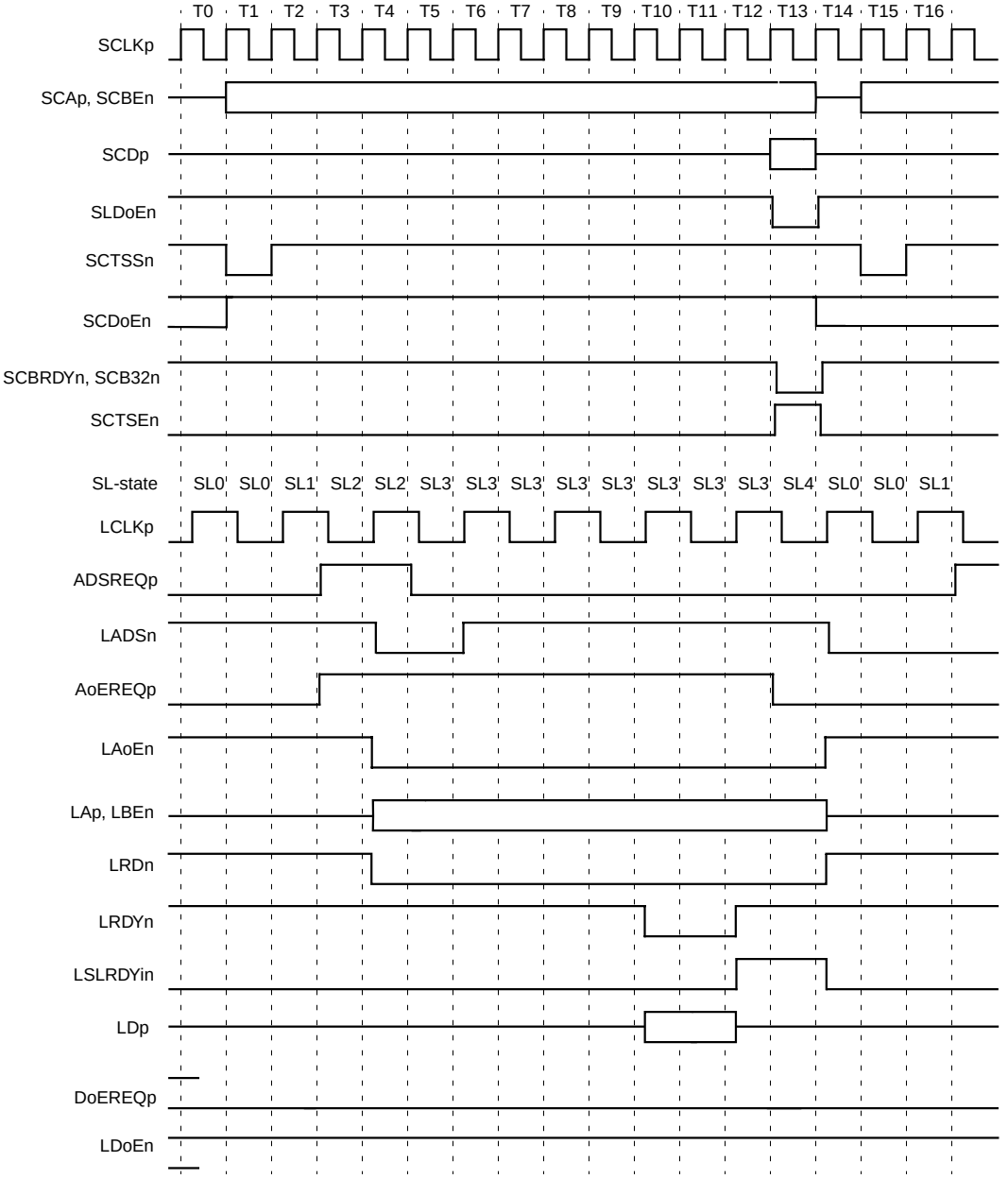
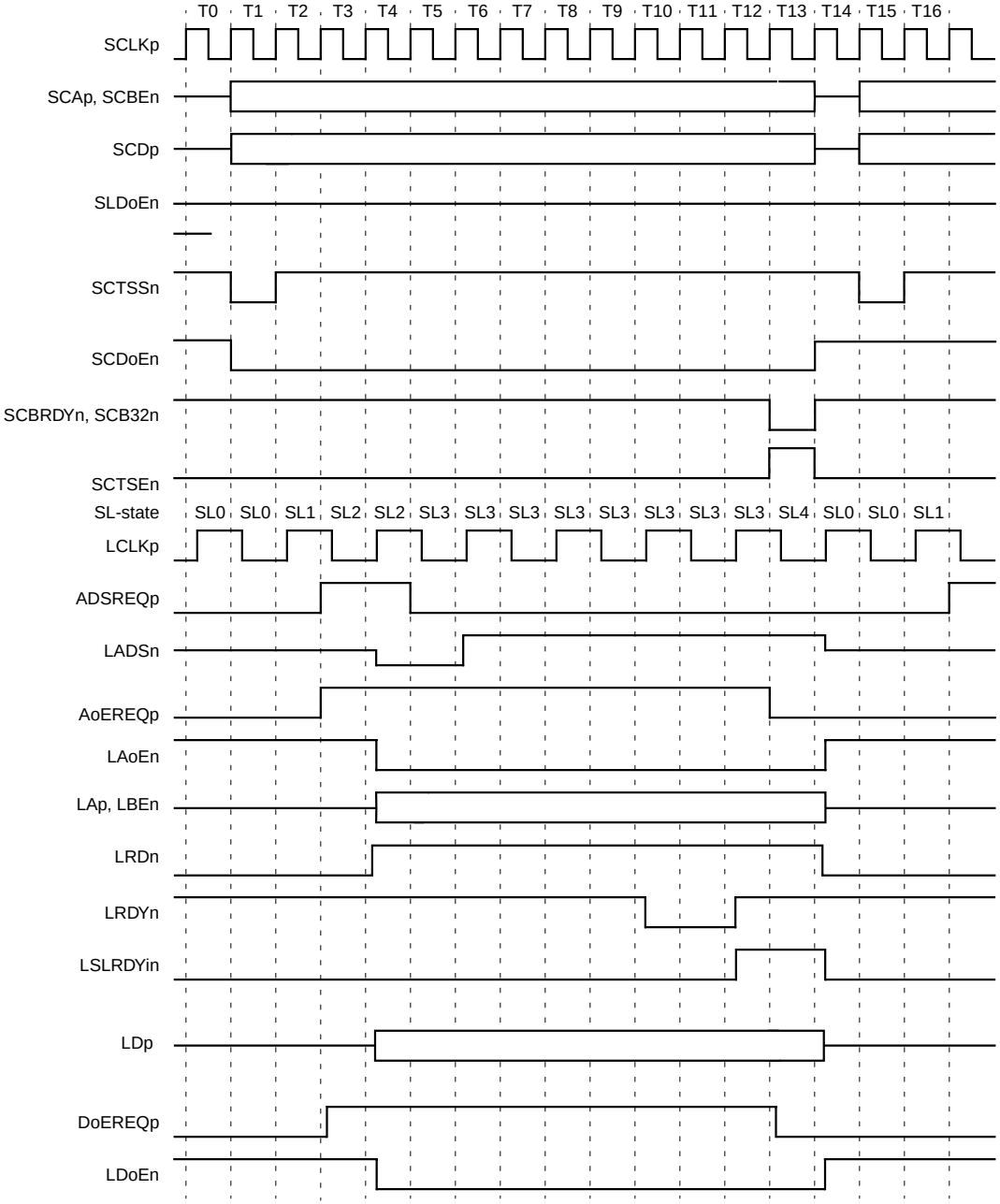


Figure 7.2 Timing Requirements for an SCbus-to-Lbus Write Transaction



In the examples shown in Figures 7.1 and 7.2, the CW4011 initiates a SCbus transaction at T1. The SCLC module, which is part of the LR4500, checks the phase LCLKp clock. At T4 and T5, the SCLC asserts address strobe, LADSn. During a write transaction, the SCLC must output data on the Lbus on the rising edge of LCLKp. The Lbus transaction starts at T4. At T12, the SCLC samples the LRDYn signal on the rising edge of LCLKp. The SCLC asserts the SCbus data ready signal, SCBRDYn, at T13. At the same time it asserts the bus sizing request signal, SCB32n. During a read transaction, the SCLC samples data on the Lbus when it samples LRDYn. If the transaction is a write transaction, the CW4011 places data on the SCbus at T13.

7.3 LR4500 as Slave on the Lbus

LR4500 functions as a slave on the Lbus when an Lbus device, such as a SONIC Ethernet Controller, initiates a bus transaction. The Lbus device accesses the system DRAM through the DRAM Controller, which is part of the LR4500 and the LR4500 acts as a slave memory controller. The address being accessed must fall in the range 0x 0000 0000 through 0x 03FF FFFF. The LR4500 does not assert the data ready signal for the address range 0x 0400 0000 through 0x FFFF FFFF, since the transaction is treated as a read/write transaction between an Lbus master and an Lbus slave. Figure 7.3 shows the timing requirements for an Lbus-to-SCbus read transaction. Figure 7.4 shows the timing requirements for an Lbus-to-SCbus write transaction.

At T1, the LR4500 samples LHoLDp on the rising edge of LCLKp. At T2, the SCLC module, which is part of the LR4500, asserts SCbus hold request, SCHRQn. The CW4011 asserts the SCbus hold grant signal, SCHGTn, at T4. At T7, the SCLC module asserts the Lbus hold acknowledge signal, LHLDAp, on the rising edge of LCLKp. While LHLDAp is asserted, the SCLC module asserts LRDYoEn to drive LRDYn. At T9 or later, the Lbus master starts an Lbus transaction. The SCLC samples LADSn on the rising edge of LCLKp. If the signal is asserted, the SCLC module knows the Lbus master has initiated an Lbus transaction. At T12, the SCLC module decodes sampled address inputs and starts an SCbus transaction if the address is in the DRAM area. The DRAM Controller asserts the data ready signal, DRRDYn, when a transaction is completed. At T17 and T18, the SCLC module asserts LRDYn and the Lbus transaction is completed.

Figure 7.3 Timing Requirements for Lbus-to-SCbus Read Transaction

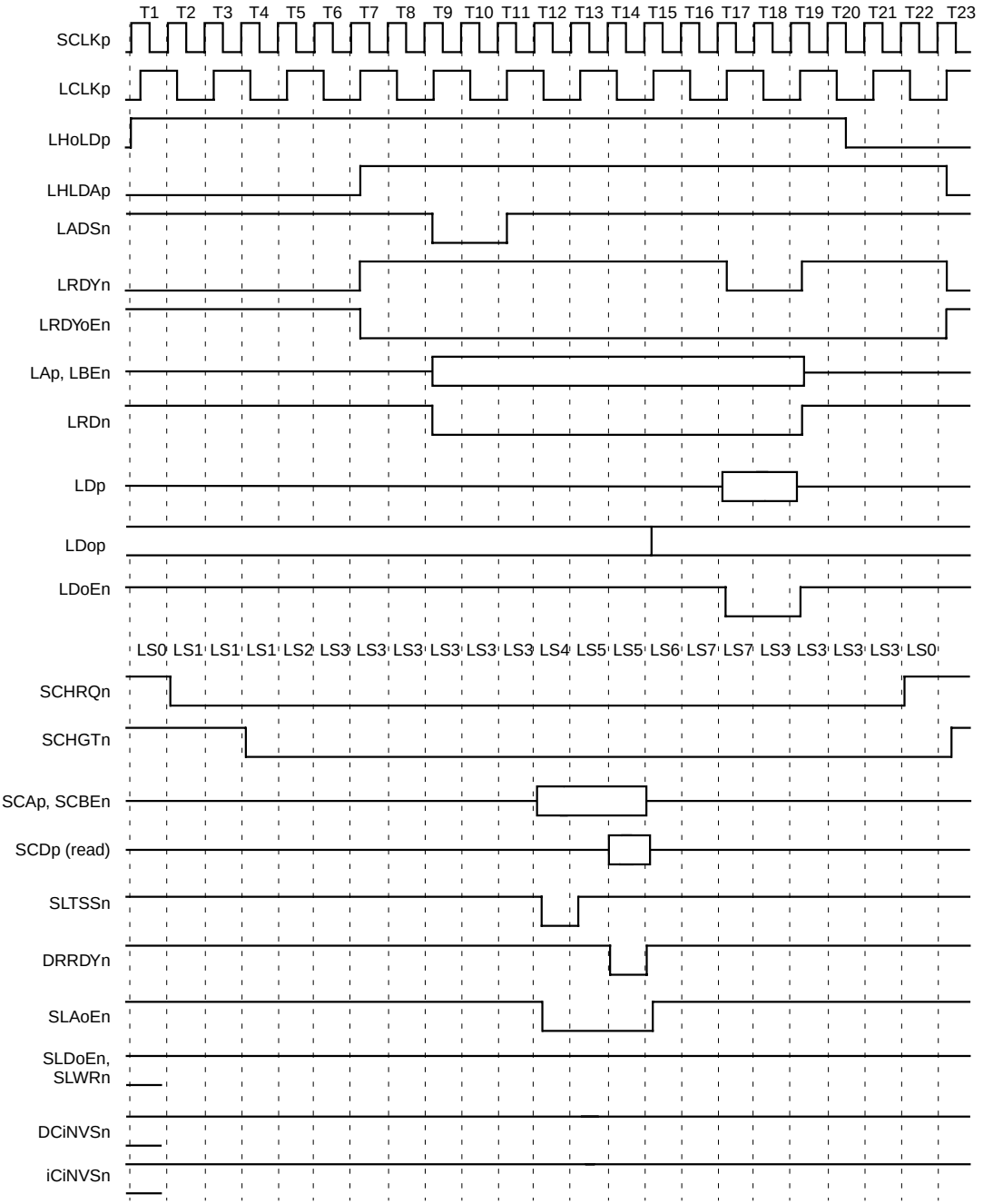
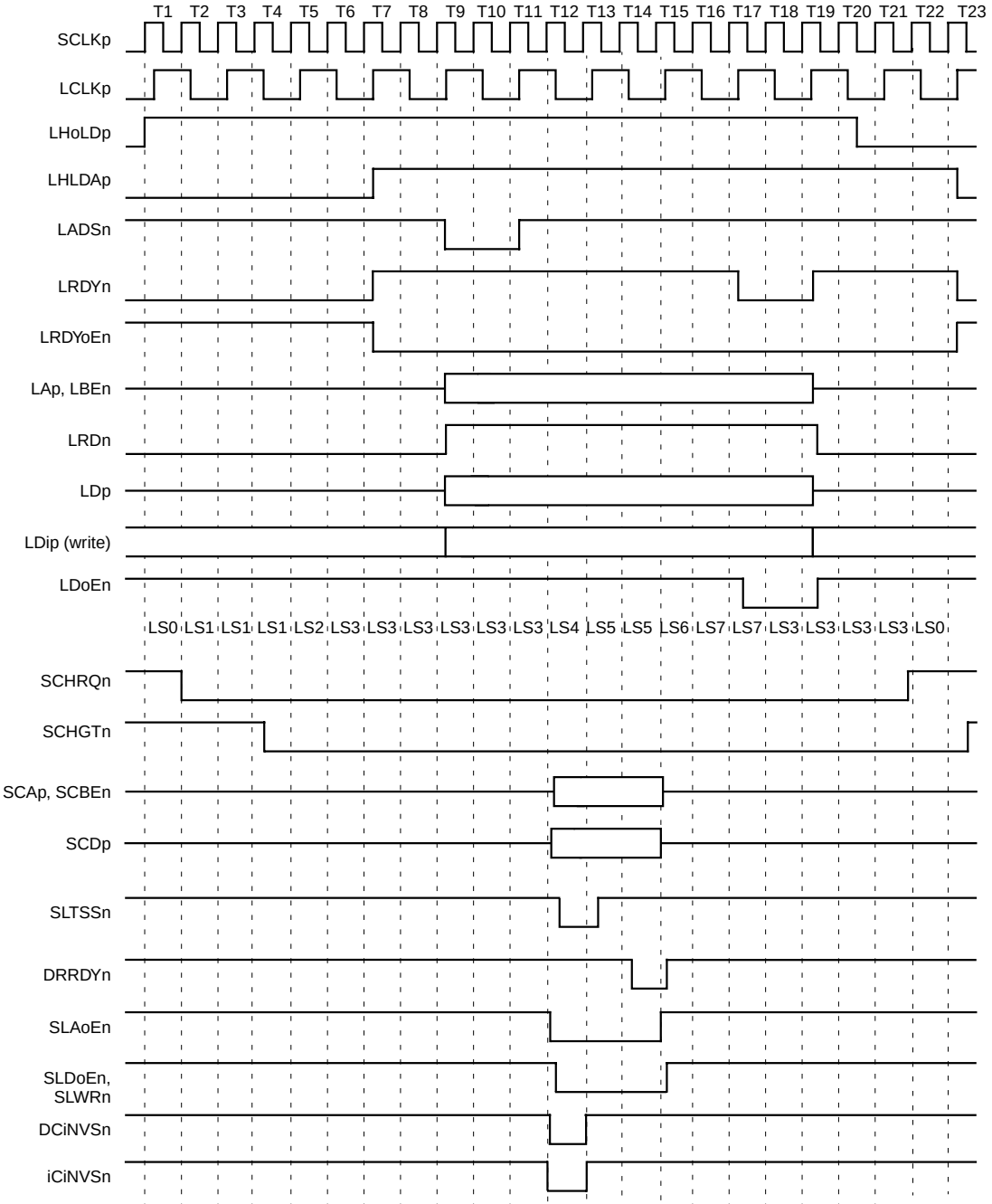


Figure 7.4 Timing Requirements for Lbus-to-SCbus Write Transaction



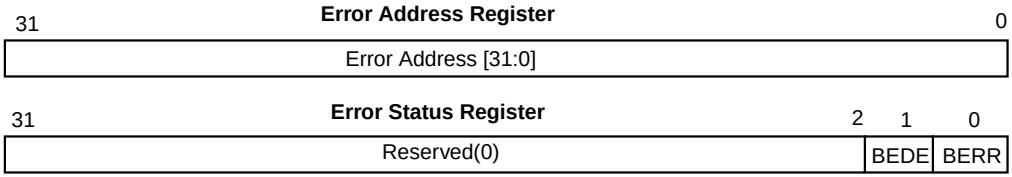
7.4 SCbus Timeout Watchdog Timer

The SCLC module in the LR4500 has a watchdog timer that it uses to time out SCbus transactions. The timer monitors the number of clock cycles for each SCbus transaction generated by the CW4011. It does not care about SCbus transactions to SCLC internal registers, or SCbus transactions generated by the SCLC module, since these transactions will never result in a timeout.

When the CW4011 initiates an SCbus transaction, the transaction must be completed within 256 SCLKp clock cycles, timed from the cycle in which SCTSSn is asserted until the cycle in which SCBRDYn or SCBRTYn is asserted. If the transaction takes longer than 256 clock cycles, the timer terminates the transaction by asserting SCBERRn, which causes a bus error exception.

The LR4500 has two registers that control SCbus timeout errors. As shown in Figure 7.5, they are the Error Status Register and the Error Address Register. The status register stores bus error detect enable (BEDE) and bus error detected (BERR) bits. The SCbus watchdog timer starts when the BEDE bit is set. If a timeout error occurs, the timer sets the BERR bit. During the time that BERR is set, the address register stores the SCbus transactions address and further bus error detection is inhibited. The SCLC can clear the bit by writing a 1 to it. A cold reset clears both bits.

Figure 7.5 SCbus Error Address and Error Status Register Bit Format



BEDE: Bus Error Detect Enable(1)
BERR: Bus Error was Detected

The physical and virtual addresses for the registers are as follows:

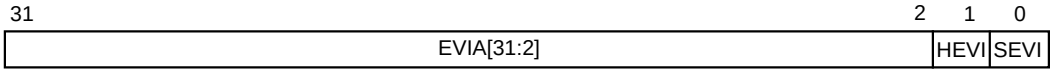
Register	Physical Address	Virtual Address
SCbus Error Address	0x1010 0000	0x B010 0000
SCbus Error Status	0x1010 0004	0x B010 0004

The sections "SCbus Error Address Register" and "SCbus Error Status Register" on page 3-10 provide further information about these registers.

7.5 External Vectored Interrupt (EVInt) Support

The LR4500 has a special interrupt exception input feature called External Vectored Interrupt. The SCLC module provides test support for this feature with the EXVI control register shown in Figure 7.6.

Figure 7.6 External Vectored Interrupt Register Bit Format



HEVI: Hardware External Vectored Interrupt
SEVI: Software External Vectored Interrupt

The physical and virtual addresses for the register are as follows:

Register	Physical Address	Virtual Address
External Vectored Interrupt	0x1010 0008	0x B010 0008

The section "External Vectored Interrupt Register" on page 3-11 provides further information about these registers.

When the LR4500 reads an exception vector address from EXVAp[31:2], it writes the address to the program counter. The EVIA[31:2] bits in the EXVI Register are connected to the EXVAp[31:2] bus to provide the vector address. When the LR4500 accepts an EVInt exception, it clears the EVIA bits to zero indicating that the timing was correct when the CW4011 sampled EXVAp[31:2].

If software sets the SEVI bit in the EXVI Register, the EVInt input of LR4500 is asserted and causes an exception. External Vectored Interrupts are enabled in the CCC Register, and interrupts are enabled in the Status Register. The software must write the extended address to the EVIA bit at the same time that it sets the SEVI bit.

The nonmaskable interrupt input to LR4500, NMI, can be used to cause an external vectored interrupt, EVInt. This bit selects the function of the NMI/EVInt pin. If the bit is cleared to 0, the pin generates a nonmaskable interrupt (NMI). If the bit is set to 1, the pin generates an external vectored interrupt (EVInt). The address is still supplied by the EVIA bits. If HEVI is cleared to 0, the falling edge of NMI causes an NMI exception. If HEVI is set to 1, the falling edge of NMI causes an EVInt exception provided that the interrupt enable bit the Status Register is set.

A cold reset clears all bits of the EXVI Register. The EVIA[31:2] bits are cleared when the CW4011 reads them. EVIA[31:2] must be programmed again if they are to be used again.

Chapter 8
Cache Configuration and Maintenance

This section describes the LR4500 Microprocessor I-cache and D-cache configurations, and explains how to maintain the caches after power is turned on.

8.1 Cache Configuration

LR4500 takes advantage of the largest I-cache and D-cache available. As described in Section 3.3.1, “CCC Register,” on page 3-5, you can use the CCC Register in CP0 to program certain features of the caches. This allows you to evaluate the performance of different cache configurations and select the one most appropriate for your application. You can configure the I-cache and D-cache independently of each other. You can program the CCC Register to implement the following features:

- Select the cache operating size.

  Smaller cache configurations need wider tag bits. The LR4500 uses the maximum number of words for the maximum configuration and the widest tag bits for the minimum configuration. To set the size, you program bits IS[1:0] for the I-cache, and DS[1:0] for the D-cache, as shown in Table 8.1.

Table 8.1 Cache Size and Accessing

Cache Bit Settings			
IE1 or DE1	**IE0 or DE0**	**IS[1:0] or DS[1:0]**	**Configuration**
0	0	X X[1]	No cache
0	1	0 0	1 Kbyte direct mapped
0	1	0 1	2 Kbyte direct mapped
0	1	1 0	4 Kbyte direct mapped
0	1	1 1	8 Kbyte direct mapped
1	1	0 0	2 Kbyte two-way set-associative
1	1	0 1	4 Kbyte two-way set-associative
1	1	1 0	8 Kbyte two-way set-associative
1	1	1 1	16 Kbyte two-way set-associative

1. The setting of these bits does not matter.

- Select between direct-mapped and two-way set-associative caching.

  To do this you program bits IE[1:0] for the I-cache and DE[1:0] for the D-cache, as shown in Table 8.1. IE1 and IE0 enable I-cache Set-1 and Set-0, respectively, and DE1 and DE0 enable D-cache Set-1 and Set-0, respectively. In the example shown in Table 8.1, Set-0 is enabled for both the I-cache and the D-cache when you require direct mapping, and Set-1 is disabled for both caches. When you select two-way set-associative caching, both sets are enabled for both caches. Note that when you select two-way set-associative caching, total cache capacity is doubled, since you are using both cache sets.

- Configure the D-cache as scratchpad RAM.

  Prior to configuring a set's associativity as scratchpad RAM, you must use cache isolation and tag test modes to program the corresponding tag memory to contain the desired physical addresses. When using isolate cache mode, stores to cache are not propagated to external memory. To initiate isolate cache and test tags mode, you must set bits IsC and TAG in the CCC Register, as described in Section 3.3.1, "CCC Register," on page 3-5.

Once this process is complete, you can configure the D-cache as scratchpad RAM by programming bits DE0 and SR0 to configure D-cache Set-0, and DE1 and SR1 to configure D-cache Set-1, as shown in Table 8.2.

Table 8.2 D-cache Scratchpad RAM Configuration

Cache Bit Settings		
DE0 or DE1	**SR0 or SR1**	**Configuration**
0	X[1]	Disabled
1	0	Cache memory
1	1	Data Scratchpad RAM

1. The setting of these bits does not matter.

- Configure the I-cache as instruction RAM.

  Prior to configuring Set-1 as instruction RAM, you must use cache isolation and tag test modes to program the tag memory to contain the desired physical addresses. In addition, you must program the corresponding data fields to contain the instruction code which is to remain resident in the cache. To initiate isolate cache and test tags mode, you must set bits IsC and TAG in the CCC Register.

  Once this process is complete, you can configure the I-cache as instruction RAM by programming bits IE1 and IR1 to configure I-cache Set-1, as shown in Table 8.3.

Table 8.3 I-cache Instruction RAM Configuration

I-cache Set-1 Bit Settings		
IE1	**IR1**	**Configuration**
0	X[1]	Disabled
1	0	Cache memory
1	1	Instruction RAM

1. The setting of this bit does not matter.

8.2 Cache Maintenance

When power is turned on to the LR4500, valid bits in the tag memories have random values. Before you program the CCC Register to select a cache configuration to enable caches, you must make sure that Cache Tag valid bits are cleared.

CW4011 core has the following instructions that you can use to flush the caches:

- FLUSHID flushes the I-cache and the D-cache
- FLUSHI flushes the I-cache
- FLUSHD flushes the D-cache

These instructions do not have any operand. To invalidate I-cache and D-cache during reset initialization, use FLUSHID. Each flush instruction causes stall cycles for 256 clock cycles, regardless of cache size. You must execute the instructions from the *kseg1* uncached and unmapped area.

Chapter 9
ICEport

This chapter describes the CW4011 ICEport building block and is divided into the following sections:

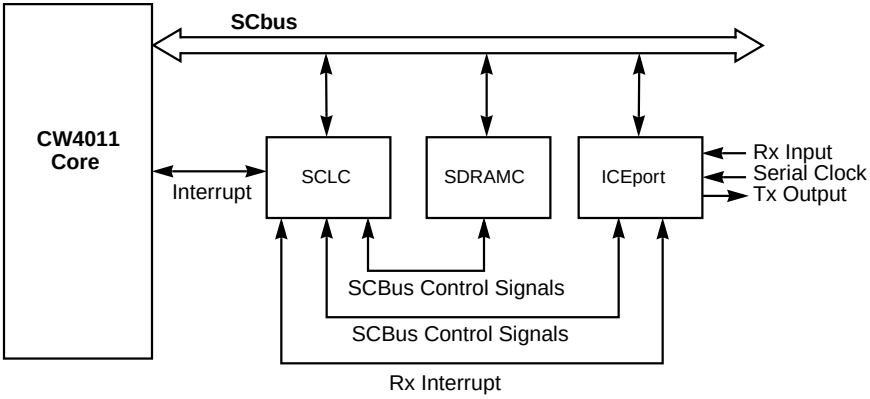
9.1 Overview

The ICEport is a full-duplex serial UART (universal asynchronous receive and transmit) port available from LSI Logic. It is an integral part of the LR4500 Microprocessor reference device. You can use the ICEport to download core application software and as a CW4011 debugging tool. The ICEport works with the ICEcontroller at baud rates typically up to 1 Mbaud/s[1], providing 800 Kbits of data per second.

Figure 9.1 shows a block diagram of a CW4011 system with the ICEport installed. In the LR4500 configuration, the ICEport is integrated with the SCLC and SDRAMC modules on the SCbus.

1. Actual baud rate depends on software, ROM access times, and the CPU clock.

Figure 9.1 CW4011 Design with ICEport



9.2 ICEport Features

The ICEport has the following features:

- Full-duplex operation.
- Requires clock support at 16 times the transfer bit rate to define receiving (Rx) and transmitting (Tx) rates. This clock is common for Rx and Tx, and may be either an external clock or one generated internally from the system clock.
- Rx ready signal to indicate that a byte of data has been received and is in the data byte input buffer.
- Separate status and data registers for Rx and Tx. The Rx Status Register contains one bit that indicates received data is in the ICEport, and one bit that indicates an overrun in the Rx input buffer. The Tx Status Register contains one bit that indicates the ICEport is ready to transmit data.
- Serial-receive and clock input do not require an active signal when the ICEport is unused. During reset, the Tx UART port defaults to an idle state and transmits an idle signal.

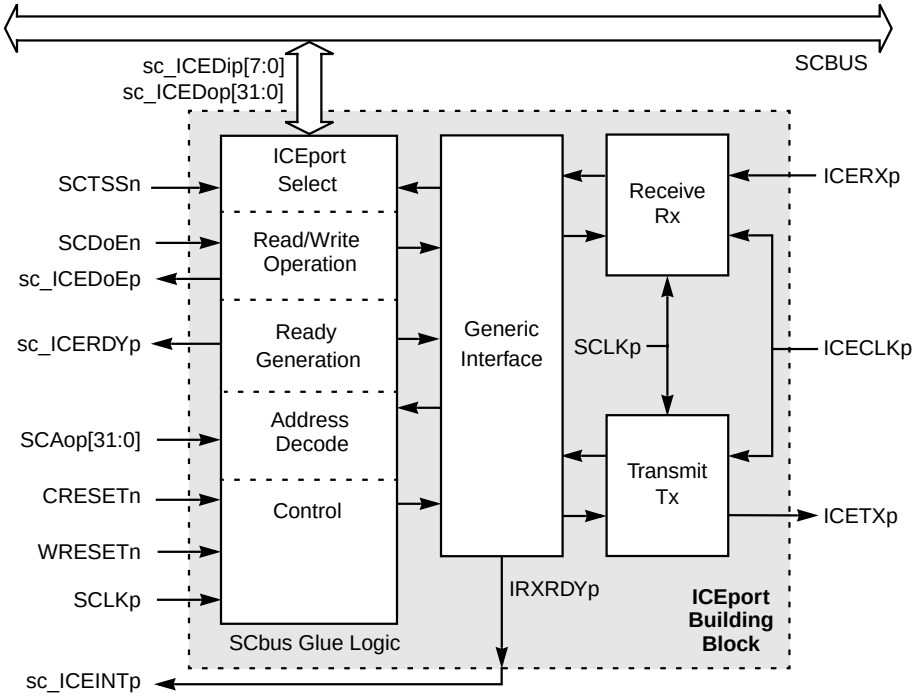
9.3 ICEport Functional Blocks

The CW4011 ICEport design has been partitioned into three logical blocks:

- Receive and Transmit logic block that sends and receives the ICETXp and ICERXp signals.
- The Generic Interface logic block, common to most core designs, that implement a SerialICE ICEport.
- The SCbus Interface logic block that connects the ICEport with the rest of the CW4011 core by means of SCbus signals.

Figure 9.2 shows how these blocks interact with each other and interface to other core logic and external logic.

Figure 9.2 ICEport Block Diagram



9.3.1 Receive and Transmit Interface Logic

The Receive Rx and Transmit Tx blocks make up the serial interface. The Rx block receives the ICERXp bit stream, and the Tx block transmits the ICETXp bit stream. Both blocks receive the internal CPU clock (SCLKp) and the external x 16 bit rate clock (ICECLKp). Both blocks synchronize timing between the ICECLKp and SCLKp timing domains. All interface signals between the Rx and Tx blocks and the Generic Interface are synchronized to SCLKp, since the Generic Interface logic block runs on SCLKp only.

9.3.2 Generic Interface Logic

The Generic Interface block connects the Tx and Rx blocks to a specific core bus interface, which is the SCbus for the CW4011. The ICEport directly outputs only the IRXRDYp signal, which must be enabled in the Rx Setup Register. When enabled, the IRXRDYp signal indicates that Rx data has been received. IRXRDYp is tied to the processor interrupt signal (sc_ICEINTp) and may be used for interrupt generation as described in Section 9.6.4.1, “Receive (Rx) Block.”

9.3.3 SCbus Interface Logic

The SCbus Interface logic block connects the Generic Interface to the CW4011 SCbus signals. The SCbus is the main internal CW4011 bus that allows a bus master to exchange information with the CW4011 core.

In SCbus transactions, the ICEport decodes the SCbus address line and checks the transaction start signal (SCTSSn) to see if the current SCbus transaction involves the ICEport. If the current transaction involves the ICEport, the SCbus Interface logic either places appropriate data on the data bus or writes data into an ICEport internal register, depending on whether the current operation is a read or a write transaction. Once either transaction is complete, the ICEport asserts the acknowledge signal (sc_ICERDYp) and the SCbus Interface logic begins to monitor SCbus transactions again.

Please be aware that the ICEport does not use the same SCbus protocol as other CW4011 core components. The ICEport uses only a certain subset of the SCbus signals and combines several SCbus acknowledge signals into a single ICEport signal. See Section 9.4.1, “Monitored SCbus Signals” page 9-6 and Section 9.4.2, “Other SCbus Signals” page 9-7 for more information on ICEport SCbus interaction.

9.4 ICEport Signals

This section describes the signals that comprise the bit-level interface of the ICEport. The following paragraphs outline the conventions used in the signal descriptions:

- The signals are described in alphabetical order by mnemonic within each functional group. Each signal definition contains the mnemonic and the full signal name.
- The mnemonics for signals that are active HIGH, or for clock signals with a positive rising edge, end with a “p;” signals that are active LOW end with “n.”
- The term *assert* means to drive TRUE or active; *deassert* means to drive FALSE or inactive.
- *Input* and *Output* in the signal headings refer to I/Os with respect to the ICEport, not with the core. For example, SCTSSn is a core output, but it is considered an ICEport input and therefore is labeled “Input.”
- All input signals, except for ICERXp and ICECLKp, are read on the positive edge of SCLKp and must therefore be generated synchronously with SCLKp.
- All output signals (except ICETXp) are also generated synchronously at the rising edge of the SCLKp clock. The ICETXp signal is synchronous to the rising edge of ICECLKp, except during a reset where ICETXp is asserted asynchronously to ICECLKp.
- In normal serial send and receive operations through the ICEport, ICECLKp runs at 16 times the rate of serial bit transmission/receive. This allows ICECLKp to define the bit width for each UART serial bit. The ICEport assumes that each serial bit for both receive and transmit is 16 x ICECLKp, or 16 ICECLKp cycles.

Table 9.1 summarizes the ICEport signals. Detailed descriptions follow the table. Note that the SCbus master can either be the SCLC module or the CW4011 processor. External logic refers to logic not related to the CW4011 core, the SCLC, or the ICEport.

Table 9.1 ICEport Signals

Group	Signal	I/O	Source/Target	Description
Monitored SCbus Signals	CRESETn	Input	SCLC	Cold Reset
	WRESETn	Input	SCLC	Warm Reset
	SCAop[31:2]	Input	SCbus master	SCbus Address
	SCDoEn	Input	SCbus master	Data Output Enable (write low/read high)
	SCTSSn	Input	SCbus master	Transaction Start Signal
Other SCbus Signals	sc_ICEDip[31:0]	Input	SCbus	SCBus Input Data Bus
	sc_ICEDop[31:0]	Output	SCbus	SCBus Output Data Bus
	sc_ICEDoEp	Output	SCbus	SCBus Output Data Valid
	sc_ICERDYp	Output	SCLC	ICEport Ready
	sc_ICEINTp	Output	SCLC	ICEport Interrupt
ICEport Scan and Clocking Signals	SCLKp	Input	External Logic	System Clock
	ICECLKp	Input	External Logic	ICE Bit Rate Clock x 16
	ICERXp	Input	External Logic	RX Serial Bit Receive
	ICETXp	Output	External Logic	TX Serial Bit Transmit
	SE	Input	External Logic	Scan Test Mode Enable
	SI	Input	External Logic	Scan Test Input
	SO	Input	External Logic	Scan Test Output
	TESTMp	Output	External Logic	Scan Test Setup

9.4.1 Monitored SCbus Signals

This section lists the SCbus signals that the ICEport monitors and outlines how the ICEport uses these signals. For a more complete description of these signals, refer to Section 5.2.1, “SCbus Interface.”

CRESETn **Cold Reset** **Input**
When the core asserts CRESETn, it resets the ICEport and all ICEport registers. CRESETn and WRESETn are internally merged in the ICEport.

SCAop[31:0] **SCbus Address Bus** **Input**

SCAop[31:0] is the address bus. The ICEport monitors this bus and SCTSSn for data read/write operations involving the ICEport. When an SCbus transaction involves the ICEport, the ICEport decodes SCAop[31:0] to decide which internal register the transaction targets.

SCDoEn **SCbus Data Output Enable** **Input**

The value of SCDoEn determines whether the present SCbus transaction is a write or a read transaction. If it is a write, SCDoEn is driven LOW; if it is a read, SCDoEn is driven HIGH. The ICEport monitors SCDoEn so that it may perform the correct action for either a read or a write transaction.

SCTSSn **SCbus Transaction Start Signal** **Input**

The core asserts SCTSSn for one clock cycle at the beginning of a transaction to announce that a new transaction has begun. Asserting SCTSSn when address SCAop[31:2] is valid initiates an ICEport read/write operation.

WRESETn **Warm Reset** **Input**

When the core asserts WRESETn, it resets the ICEport and all ICEport registers. CRESETn and WRESETn are internally merged in the ICEport.

9.4.2 Other SCbus Signals

The signals described in this section enable ICEport read and write operations and transfer data for these operations.

sc_ICEDip[7:0]

SCbus Input Data Bus **Input**

This is the SCbus input data bus. For write operations to the ICEport, data is transferred to the ICEport through this bus. On the positive edge of the same SCLKp that asserts sc_ICERDYp, the core writes data into the ICEport.

sc_ICEDop[31:0]

SCbus Output Data Bus **Output**

This is the SCbus output data bus. For read operations from the ICEport, the ICEport will place data onto this bus. Data on this bus is valid for one clock cycle and only when the sc_ICEDoEp signal is asserted.

sc_ICEDoEp **SCbus Output Data Valid** **Output**

This signal is used to drive three-state buffers in the LR4500. Asserting this signal indicates that the sc_ICEDop[31:0] bus is valid during the current cycle. sc_ICEDoEp is asserted for read transactions only and lasts for only one SCLKp cycle.

sc_ICEINTp **ICEport Interrupt** **Output**

If this signal is enabled by the RxRXRDYPe bit in the RX Setup register, the ICEport asserts sc_ICEINTp once it receives a valid byte of off-chip data. sc_ICEINTp is input to the SCLC module, which then generates an interrupt to the core in the SCLC.

sc_ICERDYp **ICEport Ready** **Output**

Asserting this signal HIGH informs the core or the SCLC module that the current transaction on the SCbus has finished. sc_ICERDYp encompasses both the SCB32n and SCBRDYn SCbus control signals.

9.4.3 ICEport Scan and Clocking I/O Signals

These signals are the scan and clocking I/O signals for the ICEport.

ICECLKp **ICE Serial Bit Clock Rate x 16** **Input**

The ICEport requires that this off-chip signal have a clock frequency 16 times greater than the serial transmit/receive rate. The ICEport assumes each serial/transmit bit is 16 ICECLKp cycles long.

ICERXp **RX Serial Bit Receive** **Input**

This is an off-chip input that holds the UART serial input data stream. Each received bit is 16 ICECLKp cycles long.

ICETXp	**TX Serial Bit Transmit**	**Output**
	This is an off-chip output that holds the UART serial output data stream. Each transmitted bit is 16 ICECLKp cycles long.	
SCLKp	**System Clock**	**Input**
	SCLKp is the global system clock input from the CW4011 core.	
SE	**Scan Test Mode Enable**	**Input**
	Asserting SE HIGH enables the scan chain; deasserting SE LOW disables the scan operation. The TESTMp signals must also be continuously asserted to enable the entire scan test.	
SI	**Scan Test Input**	**Input**
	SI is the scan chain data input signal.	
SO	**Scan Test Output**	**Output**
	SO is the scan chain data output signal.	
TESTMp	**Scan Test Setup**	**Input**
	When asserted HIGH, this signal sets up the scan test, so that scan mode is possible in the SCLKp clock domain. TESTMp signals must be continuously asserted to enable the scan test. The ICECLKp signal is ignored while TESTMp is enabling the scan test mode.	

9.5 ICEport Registers

All ICEport registers are memory-mapped as shown in Table 9.2. The default ICEport virtual base address is set to 0xB0FF 0000 (0x10FF 0000 physical address). Users can customize the ICEport address by altering the addresses in the HDL models. However, the last nibble (bits 0 to three) must be kept the same, since these four bits determine which ICEport register to access. The addresses must also be both unmapped to prevent an installed MMU from remapping memory addresses and uncached to maintain data congruency. For these reasons, LSI Logic suggests using unmapped and uncached memory space kseg1.

Table 9.2 ICEport Register Addresses

Physical Address	Virtual Address	Access	Register
0x10FF0000[1]	0xB0FF0000[1]	Read	Rx Status Register
0x10FF0000[1]	0xB0FF0000[1]	Write	Rx Setup Register
0x10FF0004	0xB0FF0004	Read	Rx Data Register
0x10FF0008	0xB0FF0008	Read	Tx Status Register
0x10FF000C	0xB0FF000C	Write	Tx Data Register

1. The physical address for the Rx Status Register is the same as the physical address for the Rx Setup Register. Similarly, the virtual addresses are the same. The Rx Status Register is a read register and the Rx Setup Register is a write register. This means that when the addresses are accessed, the register accessed depends on the condition of the read/write signal.

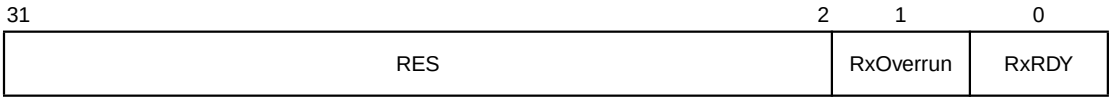
All register read transactions return zeros for bits [31:8], and data for bits [7:0]. For read transactions, the register bits are mapped with SCDip[31] to sc_ICEDop[31], and so on. For write transactions, the register bits are mapped with SCDop[7] to sc_ICEDip[7], and so on. During write transactions, data on SCDop[31:8] is ignored, write transactions to read-only registers are ignored, and read transactions from write-only registers return undefined data.

All registers must be accessed using word accesses only, to avoid conflict between big-endian and little-endian data structures, and to avoid partial update problems.

9.5.1 Rx Status Register

The read-only Rx Status Register provides status information for ICEport receive operations and indicates the state of the Rx Data Register. Figure 9.3 shows the Rx Status Register.

Figure 9.3 Rx Status Register



RES **Reserved Bits** **[31:2]**
These bits are reserved for use by LSI Logic and are read as zeros.

RxOverrun **Rx Overrun** **1**
This bit is set to one when an Rx overrun error occurs. An Rx overrun error occurs when a new Rx byte is received, as indicated by RxRDY, before the previous Rx byte has been read. For an overrun error, the new byte is not accepted and the pending byte in the Rx Data Register is not lost.

When the RxOverrun bit is set, it signals that at least one byte from the serial input stream of the new frame has been lost. RxOverrun is cleared when the Rx Status Register is read. This ensures that if another overrun occurs between the Rx Status Register read and the Rx Data Register read that this overrun will set RxOverrun.

RXOverrun clears to zero during an ICEport reset.

RxRDY **Rx Byte Ready** **0**
When the Rx block receives a byte, this bit is set to 1. RxRDY clears to zero when the Rx Data Register is read, and at reset. The IRXRDYp (sc_ICEINTp) output signal, if enabled, reflects the state of the RxRDY bit.

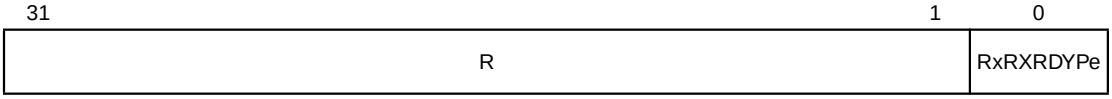
9.5.2 Rx Setup Register

The write-only Rx Setup Register enables and disables the sc_ICEINTp interrupt signal when the RxRDY bit in the Rx Status Register is set. If software clears the RxRXRDYPe bit to zero, then the ICEport interrupt signal sc_ICEINTp is disabled. This feature was added to allow software to disable the interrupt signal, IRXRDYp (sc_ICEINTp).

In the LR4500 reference device, the ICEport's interrupt IRXRDYp (sc_ICEINTp) is tied to interrupt 4, which is a maskable interrupt.

Figure 9.4 shows the Rx Setup Register, with bit field descriptions following the figure.

Figure 9.4 Rx Setup Register



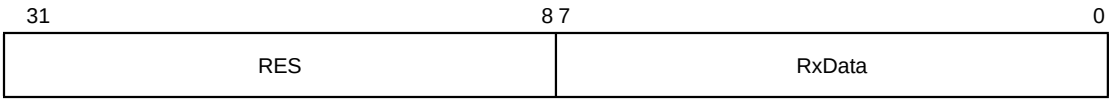
R **Reserved Bits** **[31:1]**
These bits are reserved for LSI Logic and any writes to these bits are ignored.

RxRXRDYPe **sc_ICEINTp (IRXRDYp) Enable** **0**
When this bit is set to one, the sc_ICEINTp signal reflects the state of the RxRDY bit in the Rx Status Register. When software clears RxRXRDYPe to zero, the sc_ICEINTp signal is continually deasserted. RXRXRDYPe clears to zero during an ICEport reset.

9.5.3 Rx Data Register

The read-only Rx Data Register, shown in Figure 9.5, holds received data in bits [7:0]. RxData is valid only when the RxRDY bit in the Rx Status Register is set. The Rx Data Register is undefined after an ICEport reset.

Figure 9.5 Rx Data Register



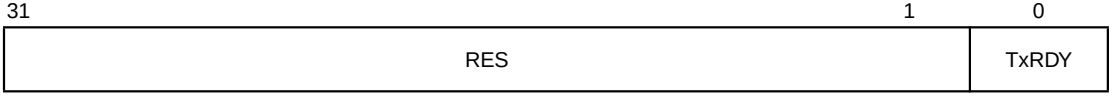
RES **Reserved Bits** **[31:8]**
These bits are reserved for LSI Logic and are read as zeros.

RxData **Received Bit Stream** **[7:0], R**
This bit field holds data received from the ICERXp serial input signal. Data held in RxData is valid only when the RxRDY bit in the Rx Status Register is set. RxData is undefined after an ICEport reset.

9.5.4 Tx Status Register

The Tx Status Register, shown in Figure 9.6, provides status information for Tx operations.

Figure 9.6 Tx Status Register



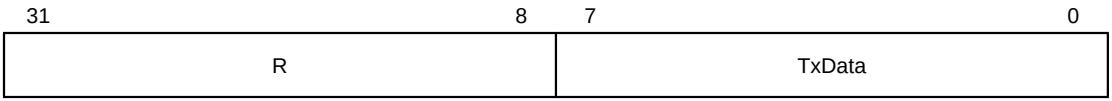
RES **Reserved Bits** **[31:1]**
These bits are reserved for LSI Logic and are read as zeros.

TxRDY **Tx Ready** **0, R**
This bit is set to one when either the Tx Data Register is ready for the next transmit byte, or after reset. TxRDY remains set during and after the transmission of Tx data. TxRDY clears to zero during a write transaction to the Tx Data Register. TxRDY is set to one after an ICEport reset.

9.5.5 Tx Data Register

The write-only Tx Data Register, shown in Figure 9.7, holds the serial transmission data.

Figure 9.7 Tx Data Register



R **Reserved Bits** **[31:8]**
These bits are reserved for LSI Logic and any write transactions to these bits are ignored.

TxData **Transmitted Bit Stream** **[7:0]**
When the TxRDY bit in the Tx Status Register is set, data that is to be transmitted through ICETXP may be written to the TxData bits. Write transactions to the TxData bits when TxRDY is zero are ignored.

9.6 ICEport Operations

This section describes the different ICEport operations, and is divided into the following sections:

9.6.1 SCbus Read/Write Transactions

All read or write transactions to the ICEport occur through the SCbus. Both transactions require two cycles once SCbus arbitration is decided. For either transaction, the bus master must first win arbitration for SCbus control and decide to initiate a transaction. The bus master then places the target address for the transaction on SCAop[31:0] and asserts SCTSSn for one cycle to indicate the start of a new transaction. The

ICEport constantly decodes SCAop[31:0] and monitors SCTSSn for transactions that target the ICEport. If the ICEport is the target of a transaction, it checks the SCDoEn signal to determine whether this transaction is a read or a write transaction.

For a read transaction, the ICEport places data on the sc_ICEDop output bus, asserts both sc_ICERDYp, and then asserts sc_ICEDoEp at the next cycle.

For a write transaction, the ICEport latches the data on sc_ICEDip into the proper register on the next rising edge of the clock, and asserts sc_ICERDYp at the following clock cycle. In order to ensure that information is not lost, the SCbus master must hold the SCAop[31:0], SCDoEn, and SCDop[31:0] signals until the ICEport asserts the sc_ICERDYp acknowledge signal.

For data transfer, the SCDop[7:0] output bus connects to the ICEport sc_ICEDip[31:0] input bus. The SCDip[31:0] input bus connects to the ICEport sc_ICEDop[31:0] output bus. The upper 32 bits of both SCbus data buses, SCDop[63:32] and SCDip[63:32], are not used for ICEport transactions.

Figure 9.8 shows the timing relationships for an ICEport read transaction, and Figure 9.9 shows the timing relationships for an ICEport write transaction. In both examples, CRESETn, WRESETn, and TESTMp are assumed to be deasserted throughout the transaction, and these signals are not shown in the figures. All read/write transactions are synchronous to the rising edge of the SCLKp. Detailed cycle descriptions follow the figures.

Figure 9.8 Read Transaction

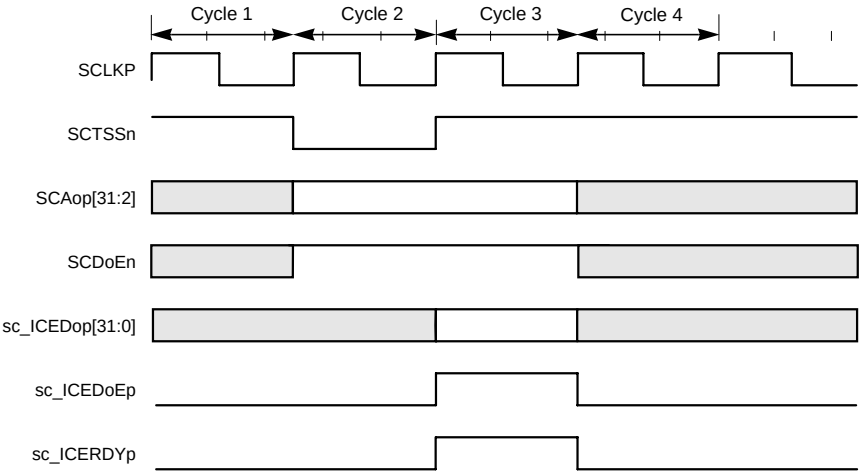
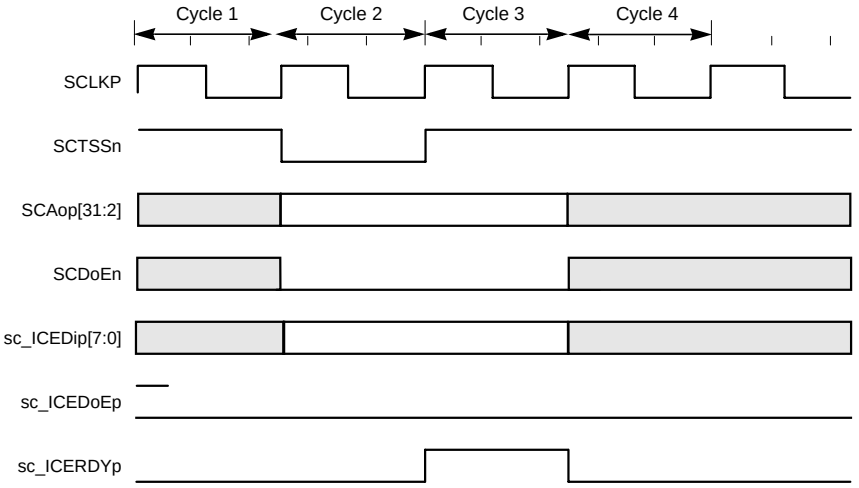


Figure 9.9 Write Transaction



The following comments apply to cycles 1 through 4 in Figure 9.8 and Figure 9.9.

Cycle 1: The bus master wins arbitration of the SCbus.

Cycle 2: The bus master asserts SCTSSn for one cycle to indicate the start of a new transaction. The bus master also places the target address on SCAop[31:0] and asserts SCDoEn for a write transaction, or deasserts SCDoEn for a read transaction. For a write transaction, the bus master also drives SCDop[31:0] with the data to be transferred.

Cycle 3: The ICEport recognizes that it is the transaction target. For a read transaction, the ICEport places the appropriate data on the sc_ICEDop[31:0] bus and asserts sc_ICEDoEp. For a write transaction, the ICEport writes sc_ICEDip[7:0] data into the appropriate register. The ICEport then asserts sc_ICERDYp to indicate that the transaction has finished.

Cycle 4: The ICEport deasserts sc_ICERDYp at the rising edge of SCLKp. For a read transaction, the ICEport also deasserts sc_ICEDoEp and the SCbus master must latch the data on the rising edge of SCLKp at the start of this cycle. At the end of Cycle 4, the ICEport is ready to begin a new transaction.

9.6.2 Reset

An ICEport system reset occurs when either CRESETn or WRESETn is asserted for at least one SCLKp cycle. CRESETn must be asserted when the system is powered up to set the ICEport in a predefined state. Since the reset signals are synchronous to SCLKp, the ICEport can be reset even if the ICECLKp clock is not running.

An ICEport system reset performs the following functions:

- RxOverrun and RxRDY bits in the Rx Status Register are cleared, indicating that the Rx Data Register is undefined.
- The RxRXRDYPe bit in the Rx Setup Register is cleared. This causes the IRXRDYp (sc_ICEINTp) signal to be deasserted.
- The TxRDY bit in the Tx Status Register is set.

9.6.3 The Serial Bit Stream

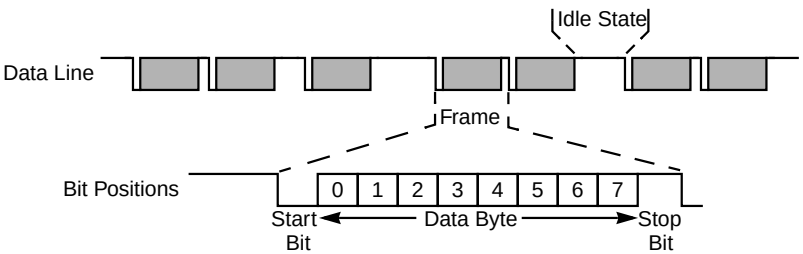
The ICEport receives data on ICERXp and transmits data on ICETXp in serial bit streams. In the Receive (Rx) block, the ICEport receives data. When no data is being transferred, the Transmit (Tx) block holds ICETXp idle HIGH.

Figure 9.10 shows an interpretation of the serial bit stream on the data line. The data bytes are received in frames, with each frame consisting of three pieces:

- a start bit, always LOW
- a byte of data, transmitted at a true level from LSB (bit 0) to MSB (bit 7)
- a stop bit, always HIGH

All bits in a frame are the 16 ICECLKp cycles long. The data line remains HIGH after the stop bit when the line goes idle, until the next start bit drives the line LOW.

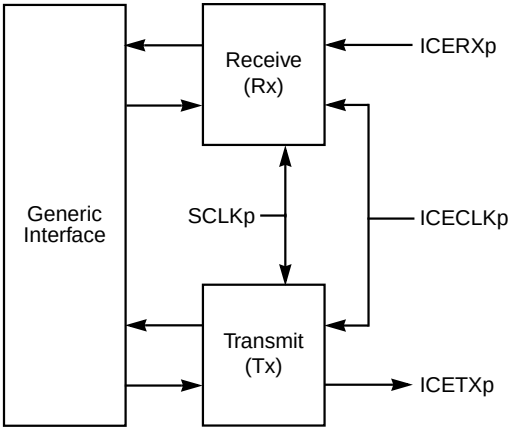
Figure 9.10 Serial Bit Stream



9.6.4 ICEport Receive and Transmit

There are two ICEport serial interface blocks, specifically the receive (Rx) and transmit (Tx) blocks. The Rx block receives the ICERXp bit stream, and the Tx block transmits the ICETXp bit stream. Both blocks receive the internal CPU clock (SCLKp) and the external bit rate clock (ICECLKp). Both blocks synchronize timing between the ICECLKp and SCLKp timing domains. Figure 9.11 shows a block diagram of the Rx and Tx blocks with I/O and clock signals.

Figure 9.11 Rx and Tx Blocks



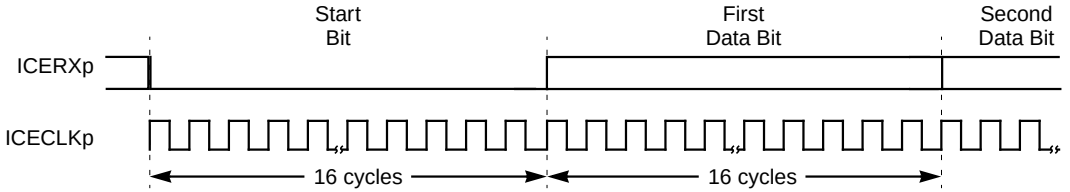
9.6.4.1 Receive (Rx) Block

ICERXp is the serial data input to the ICEport. The Rx block receives the ICERXp signal and reads it on the rising edge of ICECLKp, which can be used to generate both the transmit and receive data clocks, but usually two different clocks are implemented. This is not a problem, provided the difference between the two clock frequencies is below a certain limit, as outlined in Section 9.6.5, “Clock Domains and Properties.”

The Rx block is synchronized when ICERXp has been HIGH for nine bit times (144 ICECLKp cycles) or more, which indicates that the data line is in an idle state. The Rx block must be synchronized after power on, reset, serial cable connection, or any other event that would alter Rx block synchronization.

After synchronization, the Rx block begins sampling ICERXp on the rising edge of each ICECLKp signal. When the Rx block samples a LOW ICERXp value, the Rx block recognizes this as the start bit of a new data frame and prepares for the serial data stream. The width of each received bit is assumed to be 16 ICECLKp cycles, even though the clock that generated the data for ICERXp may be different from the ICECLKp. The value of ICERXp at the rising edge of the eighth ICECLKp is assumed to be the value of the bit, and the bit is then received. If the start bit is HIGH, the frame is ignored. In this case, the ICERXp LOW value that indicated the start of the frame was accidental. Figure 9.12 shows the Serial Bit clocking relative to ICECLKp.

Figure 9.12 Received Bit Timing



The Rx block places in the Rx Data Register the eight data bits received after the start bit. The first data bit received after the start bit is the LSB (bit 0), and the eighth data bit received is the MSB (bit 7). The eight data bits received between the start and stop bits are all true level values.

A valid HIGH stop bit received at the end of the frame sets the RxRDY bit in the Rx Status Register. The IRXRDYp (sc_ICEINTp) output reflects the state of the RxRDY bit, if IRXRDYp is enabled by the RxRXRDYPe bit in the Rx Setup Register. IRXRDYp can be used as an interrupt to ensure that the CPU reads the data received, thus avoiding overruns. If the stop bit is LOW, the frame is ignored.

A received data byte is not placed in the Rx Data register until a valid stop bit is received. This data byte will be available through the next data byte (frame) receive, until the next valid stop bit refreshes the Rx Data register. In other words, a previously received data byte is present in the Rx Data register for at least nine bit cycles (144 ICECLKp cycles) after a new start bit for a new frame is received.

If a previously received byte has not been read when a new byte is ready for the Rx Data register, an overrun error occurs. When an overrun error occurs, the ICEport sets the RxOverrun bit int he Rx Status register, and the new frame is discarded.

If the ICEport receives an invalid stop bit, the stop bit is not recorded by the ICEport registers, but the frame is still discarded. The ICEport will not accept a new start bit until the previous frame has been finished by a valid stop bit or a HIGH value on ICERXp. This ensures that the ICEport will not indicate a runaway receive if ICERXp is continuously either HIGH or LOW. Therefore, the ICEport will not receive a frame after reset if ICERXp is continuously either HIGH or LOW.

When the Rx block receives the stop bit correctly, a LOW value in the bit stream immediately following the stop bit will start the next frame. The start bit must be allowed to begin quickly, since ICECLKp may be slower than the clock that generates the data for ICERXp. In such a case, the next received frame may start on the next sample ICECLKp.

9.6.4.2 Transmit (Tx) Block

The ICETXp signal is the ICEport serial data output and can carry new data every 16 ICECLKp cycles. When there is no data for transmission, ICETXp is held HIGH in an idle state. During this idle state, the TxRDY bit in the Tx Status Register is set to one, which indicates that transmission may be initiated by placing data in the Tx Data Register. After data is written to the Tx Data Register, the ICEport clears the TxRDY bit to zero.

Start bit transmission begins on the rising edge of ICECLKp and the first data bit starts transmitting 16 ICECLKp clock cycles later. Every bit of the transmitted frame has a width of 16 ICECLKp cycles. The Tx Data Register LSB (bit 0) is transmitted just after the start bit; the MSB (bit 7) is sent just before the stop bit. All data bits are transmitted true level, with zeros sent as LOW values and ones sent as HIGH values.

The ICEport sets the TxRDY bit in the Tx Status Register when data bit 7 (the end of the byte) begins transmitting. As soon as TxRDY is set, the next data byte to transmit can be written to the Tx Data Register. Writing to the Tx Data Register while either data bit 7 or the stop bit is transmitting ensures that the ICETXp signal will not be idle. If the next data byte is not written to the Tx Data Register before the stop bit is transmitted, the Tx block will idle for a number of ICECLKp cycles, until new data is available in the Tx Data Register.

9.6.5 Clock Domains and Properties

Since data commonly moves between the ICECLKp domain and the Rx clock domain, these two clocks must have frequencies within certain limits. The difference between the ICECLKp frequency and the ICERXp clock frequency may be no more than ± 1%, with ICERXp jitter margins ± 10% of the bit width. This jitter can originate from transmission cables or different timing in LOW-to-HIGH and HIGH-to-LOW transitions.

The UART receiving the output from ICETXp may, however, require less difference between the two frequencies, and this requirement must be observed.

The ICECLKp signal may be derived from SCLKp by using a divider. This method frees a pin since ICECLKp no longer requires an external pin. The operation of the ICEport does not change in any way if ICECLKp is derived from SCLKp, but the frequency difference of ±1% must be adhered to regardless of the clock rate.

The ICEport may also transfer data internally between the two clock domains (between ICEport and the core). For these transactions, the ICECLKp frequency can be at most one quarter of the SCLKp frequency. No matter what the frequency difference between ICECLKp and SCLKp, the bus master must have enough time to read received data before new data arrives, otherwise, an overrun error will occur.

9.7 ICEport Pin Buffers and Drivers

The choice of ICEport external pin buffers and drivers will vary with each design. However, this section provides a few general recommendations for any design using an ICEport. Please note that the pin reserved for ICECLKp may be conserved if the ICEport clock is internally derived from SCLKp, as described in Section 9.6.5, “Clock Domains and Properties.”.

- The buffer for input pin ICERXp should be a 5 V-compatible schmitt trigger with an internal pull-up resistor, since the incoming signal may be noisy and driven from a 5 V source. An internal pull-up resistor is recommended so that ICERXp can be left unconnected if the ICEport is unused.
- The driver for the ICETXp output pin should be a 4 mA driver, with a reduced slew rate to avoid reflections.

Chapter 10
Organization of Clock and Exception Signals

This section describes the organization of the LR4500 Microprocessor clock circuitry that controls the LR4500's clock inputs and outputs, and LR4500 synchronization circuitry that handles exception inputs.

10.1 Clock Circuitry

The PLL circuit supplies the CW4011 core with the system clock, SCLKp. Figure 10.1 shows how the PLL output is distributed to internal LR4500 modules, such as the DRAM Controller and the SCLC, as well as to the CW4011 core itself. The phase time of the SCLKp inputs is the same for all internal modules.

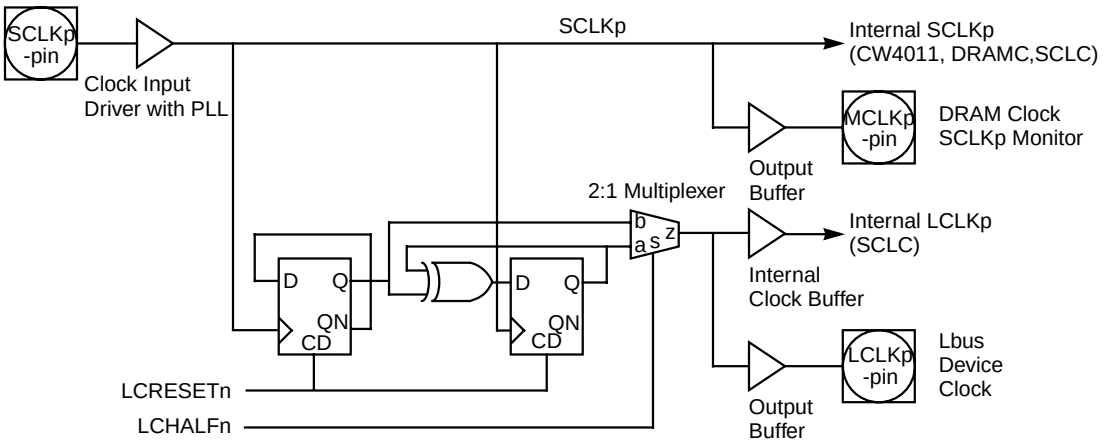
The LR4500 buffers SCLKp and outputs it as MCLKp, which monitors the internal clock, defines relative AC specifications for SCLKp synchronized inputs and outputs, and may be used as the DRAM clock.

The LR4500 generates the clock for the Lbus by dividing SCLKp either by 2 or by 4. SCLKp is passed through a two-stage D-type flip-flop, as shown in Figure 10.1, and output to a 2:1 multiplexer, which is controlled by the LCHALFn input. Multiplexer input 'b' generates the 1/2 clock while multiplexer input 'a' generates the 1/4 clock. When LCHALFn is HIGH, it enables the input on pin 'b' to be output on pin 'z' of the multiplexer; when LCHALFn is LOW, it enables the input on pin 'a' to be output. The two-stage flip-flop is reset when LCRESETn goes LOW.

The Lbus clock, LCLKp, is buffered and used as an internal clock for the SCLC. It is also output on the Lbus to provide the clock for Lbus devices. Devices on the Lbus sample all inputs on the rising edge of LCLKp, and synchronize all outputs to the rising edge of LCLKp.

Table 10.1 summarizes the clock generation process. Figure 10.2 shows the timing requirements for the CW4011 and Lbus clocks.

Figure 10.1 LR4500 PLL Clock Circuitry



Multiplexer input ‘b’ generates the 1/2 clock, multiplexer input ‘a’ generates the 1/4 clock

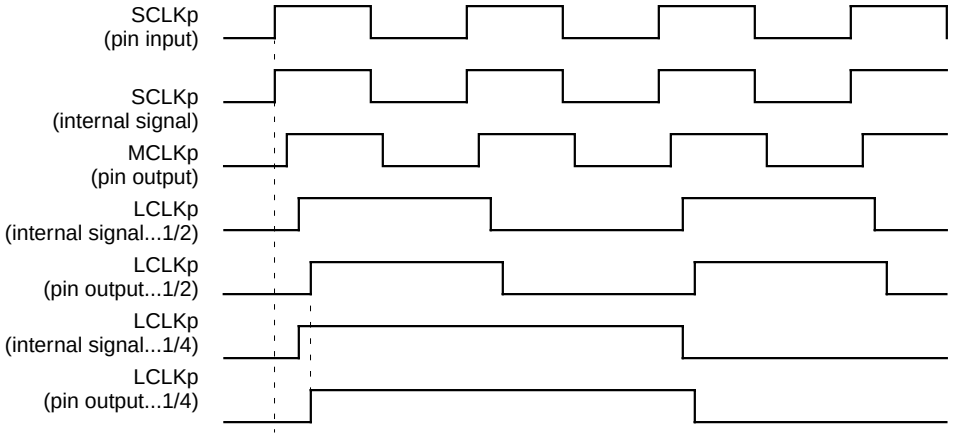
When LCHALFn is HIGH, it enables the input on pin ‘b’ to be output on pin ‘z’ of the multiplexer, generating the 1/2 clock

When LCHALFn is LOW, it enables the input on pin ‘a’ to be output on pin ‘z’ of the multiplexer, generating the 1/4 clock

Table 10.1 Summary of LR4500 Clocks

Clock Name	Source	Frequency	Comments
SCLKp	Pin input SCLKp	DC to 100 MHz	CW4011 clock
MCLKp	SCLKp	Same as SCLKp frequency	DRAM clock, SCLKp monitor clock
LCLKp	SCLKp divided by 2, or SCLKp divided by 4	1/2 or 1/4 of SCLKp frequency	Lbus clock

Figure 10.2 Timing Requirements for the CW4011 and Lbus Clocks



10.2 Exception Inputs

Exception inputs to the LR4500 may be asynchronous. These inputs include:

- Cold reset exception input, SCRESETn
- Warm reset exception input, SWRESETn
- Nonmaskable interrupt exception, SNMin
- External interrupt exceptions, SEXTiNTn[5:0]

The SCLC module in the LR4500 has a synchronization circuit that synchronizes these inputs to the system clock, SCLKp. As shown in Figure 10.3, the synchronization circuit consists of a series of D-type flip-flops that are clocked on the rising edge of SCLKp. The exception inputs reset the first stage, Flip-Flop A. On the rising edge of SCLKp, the Q output from A is passed to the D-input of Flip-Flop B. The next SCLKp input clocks this stage, and the Q output from B is passed to the D-input of the final stage, which outputs synchronous exception signals on the rising edge of the third SCLKp. Figure 10.4 shows the timing requirements for the synchronization circuit.

Figure 10.3 Exception Inputs Synchronization Circuitry

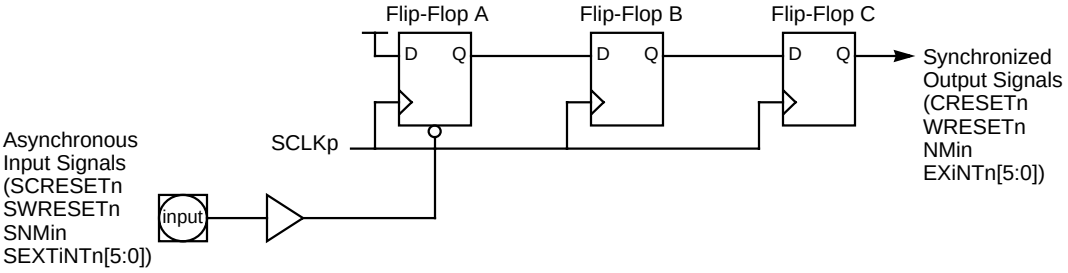
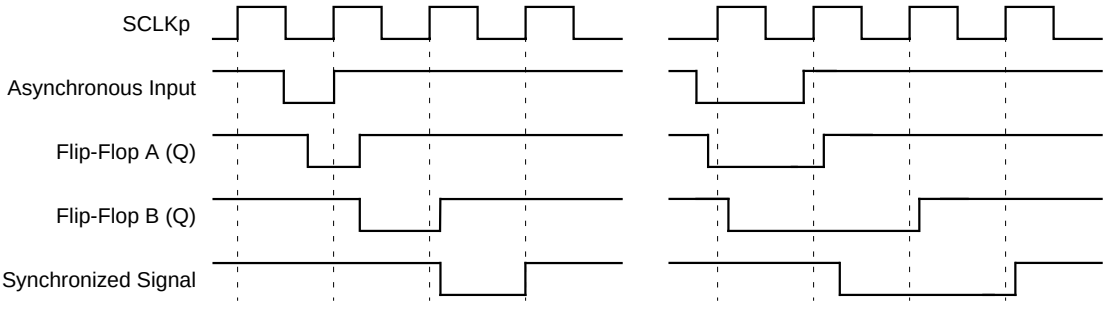


Figure 10.4 Timing Requirements for Synchronization Circuit



Case 1

Asynchronous Input width less than the SCLKp cycle time

Case 2

Asynchronous Input width greater than the SCLKp cycle time

Chapter 11
Specifications

This chapter provides the specifications for the LR4500 Microprocessor.

The chapter contains the following sections:

11.1 Electrical Characteristics

This section defines the electrical characteristics of the LR4500 Reference Device.

11.1.1 Absolute Maximum Ratings

Table 11.1 lists the absolute maximum ratings of the LR4500.

Table 11.1 Absolute Maximum Ratings

Symbol	Parameter	Limits (Referenced to VSS)	Unit
VDD	DC supply	– 0.3 to + 3.9	V (Volts)
VIN	Input voltage	– 1.0 to V_{DD} + 0.3	V (Volts)
VIN	5 V compatible input voltage	– 1.0 to + 6.5	V (Volts)
IIN	DC input current	± 10	A (microamperes)[1]
TSTG	Storage temperature range	– 40 to + 125	°C (degrees Centigrade)

1. Except for power pins.

11.1.2 Recommended Operating Conditions

Table 11.2 lists the recommended operating conditions for the LR4500.

Table 11.2 Recommended Operating Conditions

Symbol	Parameter	Limits (Referenced to VSS)	Unit
VDD	DC supply, commercial	+ 3.15 to 3.45	Volts
TC	Case temperature	85 ˚C	Degrees Centigrade)

11.1.3 Input/Output Capacitance

Table 11.3 lists the capacitance of the LR4500's input and output signals.

Table 11.3 Input/Output Capacitance

Symbol	Parameter	Limits (Referenced to VSS)	Unit
CIN	Input capacitance	3.0	pF (picafarads)
COUT	Output capacitance	3.0	pF
CIO	I/O buffer capacitance	3.0	pF

11.1.4 DC Characteristics

Table 11.4 lists the LR4500's DC characteristics.

Table 11.4 DC Characteristics

Symbol	Parameter	Condition	Limits			Unit
			Min.[1]	Typ.[2]	Max.[3]	
VIL	Input voltage low	Not applicable	– 0.5	—	0.8	V (Volts)
VIH	Input voltage high		2.0	—	VDD + 0.3	V
VOL	Output voltage low		—	0.2	0.4	V
VOH	Output voltage high		2.4	VDD - 0.3	—	V
IIL	Input leakage current	VDD = max. VIN = VDD or VSS	– 10	± 1	+ 10	A (microamperes)
IOZ	3-state output leakage current	VDD = max. VIN = VDD or VSS	– 10	± 1	+ 10	A

1. Minimum
2. Typical
3. Maximum

11.1.5 AC Timing Specifications

Table 11.5 lists the AC timing specifications for the LR4500. Figure 11.1 (page 11-5) shows timing relationships. The specifications are valid in the temperature range 0–85 ˚C case; VDD 3.3 V, ± 5%. Setup and hold times, which are relevant only for inputs to the LR4500, are referenced to the rising edge of the system clock (SCLKp) or the Lbus clock (LCLKp). The valid maximum times are equivalent to hold time for the LR4500's outputs. They are not relevant for the inputs. They are referenced to the rising edge of SCLKp or LCLKp. For 3-state signals, valid maximum times include the period from high z to valid and valid to high z. (The z indicates the 3-state or 'off' condition of the signal.)

Table 11.5 LR4500 AC Timing Specifications

Signal Name	Reference Clock	Input/ Output (I/O)	Loading (pF)	Buffer Type	Input Timing		Output Timing— Valid Maximum (ns)
					Setup (ns)	Hold (ns)	
MAp[11:0]	MCLKp	O	50	b8rp	—	—	6.2
MDp[63:0]	MCLKp	I/O	15	bd2c	1.0	0.5	6.8
MCSn[1:0]	MCLKp	O	30	b6r	—	—	5.2
MRASn	MCLKp	O	50	b8rp	—	—	5.5
MCASn	MCLKp	O	50	b8rp	—	—	5.9
MWEn	MCLKp	O	50	b8rp	—	—	6.0
MDQMp[7:0]	MCLKp	O	15	b2	—	—	5.0
LCLKp	SCLKp	O	50	b12	—	—	3.2
LAp[31:2]	LCLKp	I/O	50	bd4crf	10.0	0.0	9.7
LDp[31:0]	LCLKp	I/O	50	bd4crf	3.5	0.5	12.7
LBEn[3:0]	LCLKp	I/O	50	bd4crf	9.0	0.0	9.8
LRDn[1]	LCLKp	I/O	50	bd4crf	9.0	0.0	8.3
LADSn[1]	LCLKp	I/O	50	bd4crf	2.0	0.0	8.0
LRDYn[1]	LCLKp	I/O	50	bd4crf	2.0	0.5	8.0
LRTYn[1]	LCLKp	I	—	ibuff	2.0	0.5	—

Table 11.5 (Cont.) LR4500 AC Timing Specifications

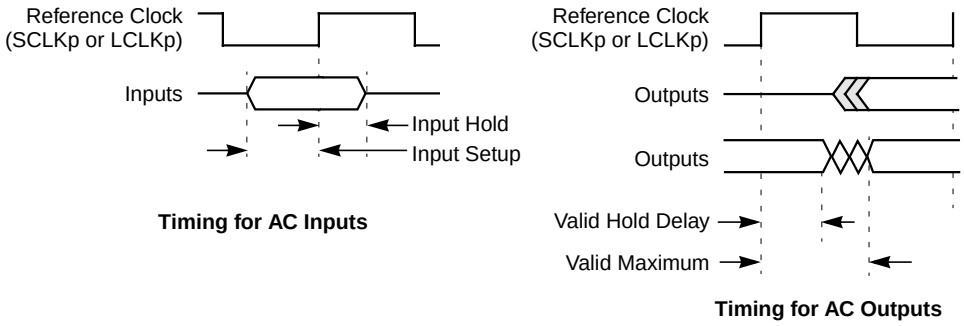
Signal Name	Reference Clock	Input/ Output (I/O)	Loading (pF)	Buffer Type	Input Timing		Output Timing— Valid Maximum (ns)
					Setup (ns)	Hold (ns)	
LHoLDp[1]	LCLKp	I	—	ibuff	3.5	0.0	—
LHLDAp[1]	LCLKp	O	30	b2	—	—	6.0
SCRESETn[1, 2]	SCLKp	I	—	schmitcf	1.5	0.5	—
SWRESETn[1, 2]	SCLKp	I	—	ibuff	0.0	1.0	—
SNMin[1, 2]	SCLKp	I	—	ibuff	0.0	1.0	—
SEXINTn[5:0][1, 2]	SCLKp	I	—	ibuff	0.0	1.0	—
ICERXp	SCLKp	I	—	schmitcf	0.5	1.0	—
ICETXp	SCLKp	I	50	b4	—	—	9.0

1. Setup and hold times guaranteed by design.
2. These are asynchronous inputs that are synchronized in the LR4500. Input setup and hold times specify the times these signals are sampled.

The following parameters are critical and you should check them carefully.

1. Mbus outputs valid minimum—DRAM requirement time is 1 ns.
2. Lbus outputs valid minimum—related data hold-time parameters for Lbus devices.

Figure 11.1 AC Timing for LR4500 Inputs and Outputs



11.2 Packaging

This section provides packaging information for the LR4500 Reference Device. Figure 11.2 shows the mechanical layout and dimensions, and the pin locations.

Figure 11.2 256 PQFPt Mechanical Drawing

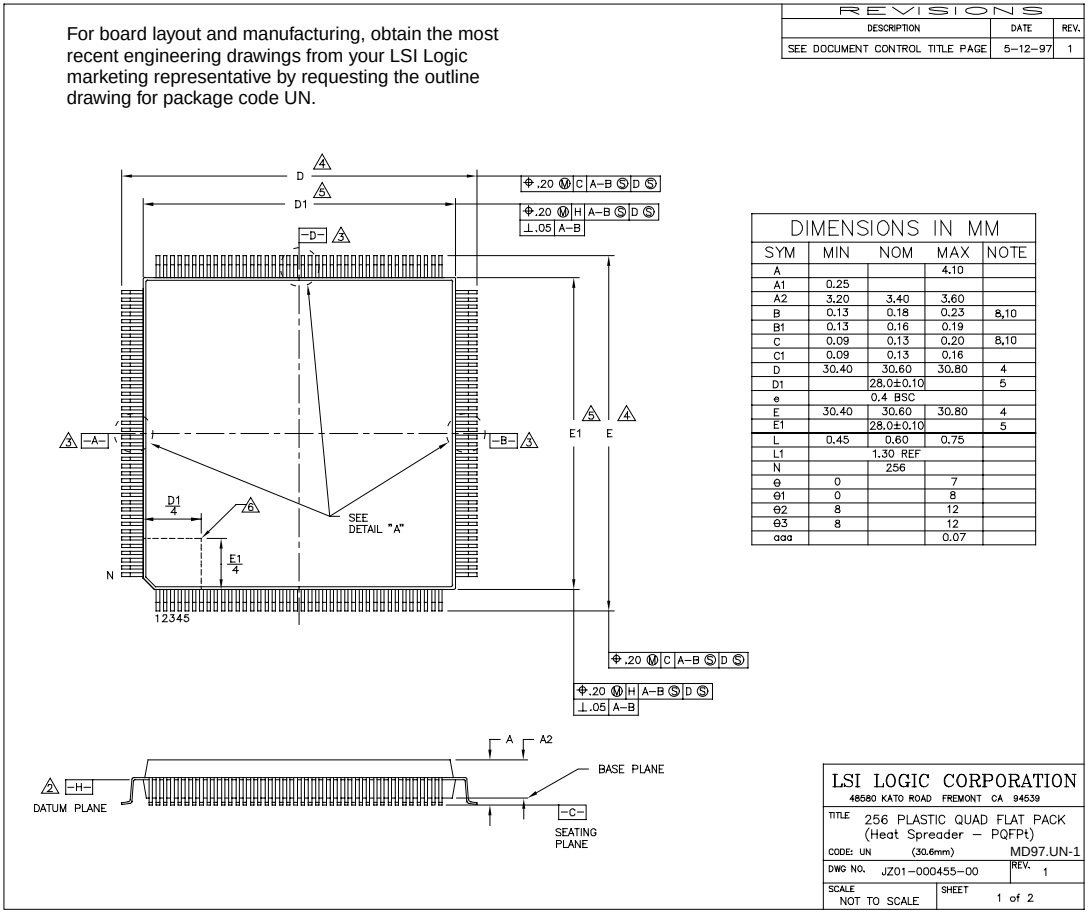
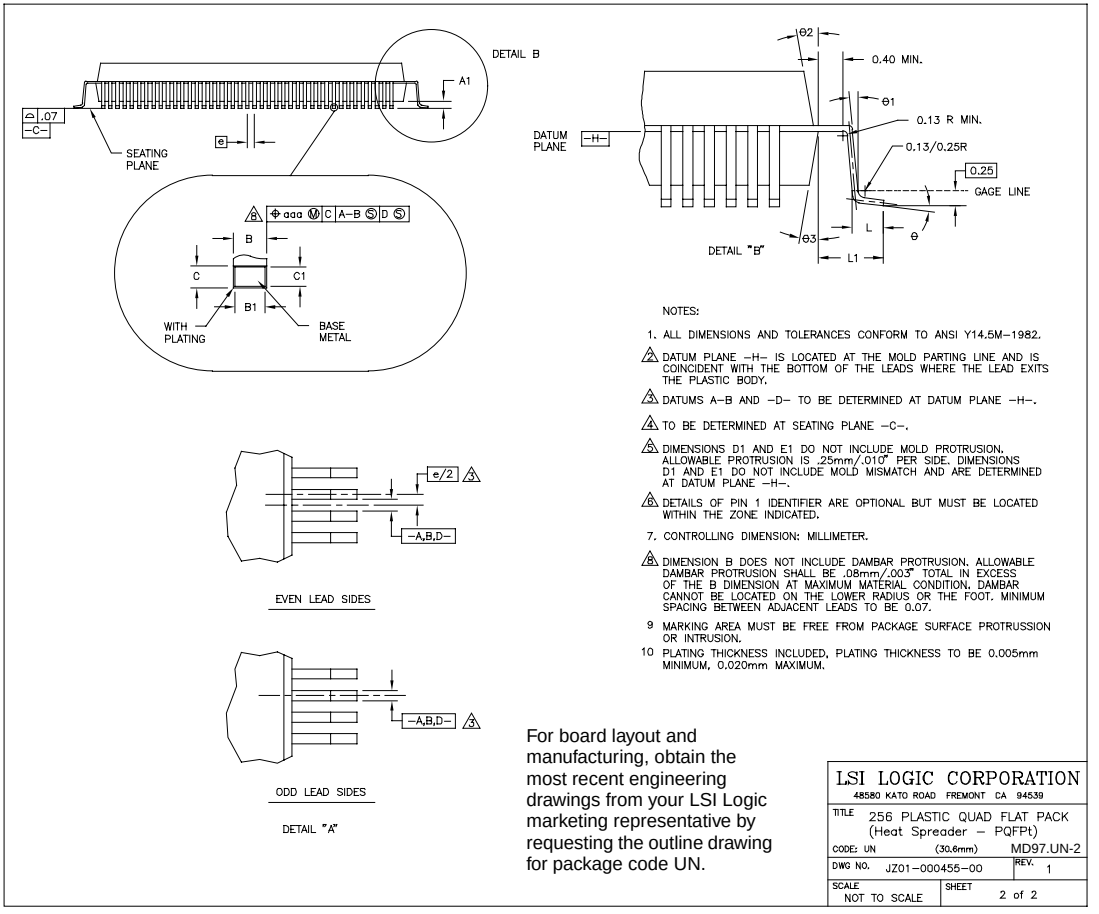


Figure 11.2 (Cont.)256 PQFPt Mechanical Drawing



NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1982.
2. DATUM PLANE –H– IS LOCATED AT THE MOLD PARTING LINE AND IS COINCIDENT WITH THE BOTTOM OF THE LEADS WHERE THE LEAD EXITS THE PLASTIC BODY.
3. DATUMS A–B AND –D– TO BE DETERMINED AT DATUM PLANE –H–.
4. TO BE DETERMINED AT SEATING PLANE –C–.
5. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS .25mm/.010" PER SIDE. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE –H–.
6. DETAILS OF PIN 1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE ZONE INDICATED.
7. CONTROLLING DIMENSION: MILLIMETER.
8. DIMENSION B DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE .08mm/.003" TOTAL IN EXCESS OF THE B DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACING BETWEEN ADJACENT LEADS TO BE 0.07.
9. MARKING AREA MUST BE FREE FROM PACKAGE SURFACE PROTRUSSION OR INTRUSION.
10. PLATING THICKNESS INCLUDED, PLATING THICKNESS TO BE 0.005mm MINIMUM, 0.020mm MAXIMUM.

For board layout and manufacturing, obtain the most recent engineering drawings from your LSI Logic marketing representative by requesting the outline drawing for package code UN.

LSI LOGIC CORPORATION		
48580 KATO ROAD FREMONT CA 94539		
TITLE: 256 PLASTIC QUAD FLAT PACK (Heat Spreader – PQFPt)		
CODE: UN	(30.6mm)	MD97.UN-2
DWG NO. JZ01–000455–00		REV. 1
SCALE NOT TO SCALE	SHEET 2 of 2	

11.3 Pinouts

This section defines the LR4500 pinouts. Figure 11.3 shows an outline of the device and identifies the pins. Table 11.6 lists the pinouts alphabetically.

Figure 11.3 256 PQFPt Pinouts

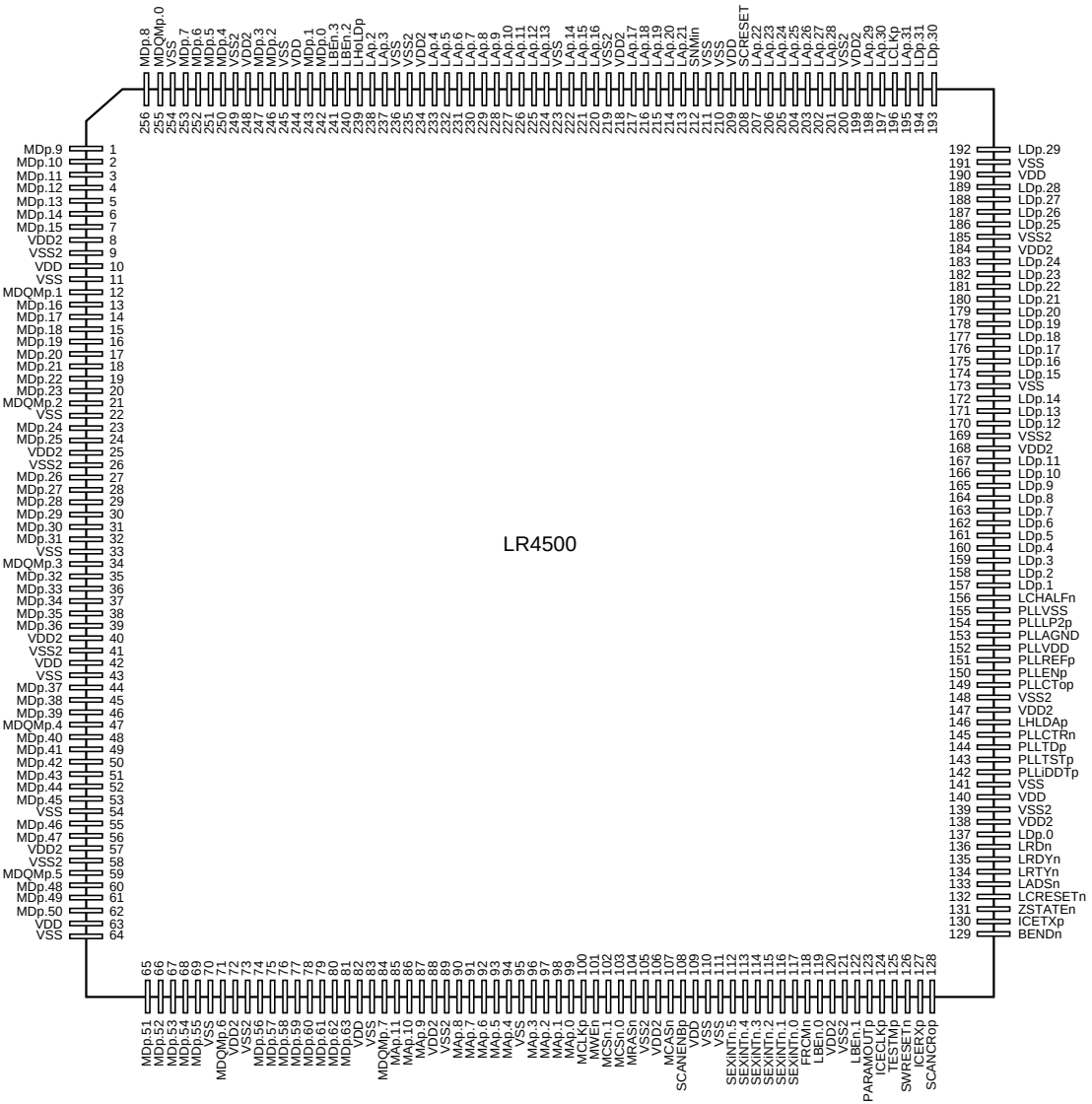


Table 11.6 LR4500 Alphabetical Pin List

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
BENDn	129	LDp.8	164	MDp.7	253	MDp.59	77	VDD2	72
FRCMn	118	LDp.9	165	MDp.8	256	MDp.60	78	VDD2	88
ICECLKp/		LDp.10	166	MDp.9	1	MDp.61	79	VDD2	106
SCANRip	124	LDp.11	167	MDp.10	2	MDp.62	80	VDD2	120
ICERXp	127	LDp.12	170	MDp.11	3	MDp.63	81	VDD2	138
ICETXp	130	LDp.13	171	MDp.12	4	MDQMp.0	255	VDD2	147
LAp.2	238	LDp.14	172	MDp.13	5	MDQMp.1	12	VDD2	168
LAp.3	237	LDp.15	174	MDp.14	6	MDQMp.2	21	VDD2	184
LAp.4	233	LDp.16	175	MDp.15	7	MDQMp.3	34	VDD2	199
LAp.5	232	LDp.17	176	MDp.16	13	MDQMp.4	47	VDD2	218
LAp.6	231	LDp.18	177	MDp.17	14	MDQMp.5	59	VDD2	234
LAp.7	230	LDp.19	178	MDp.18	15	MDQMp.6	71	VDD2	248
LAp.8	229	LDp.20	179	MDp.19	16	MDQMp.7	84	VSS	11
LAp.9	228	LDp.21	180	MDp.20	17	MRASn	104	VSS	22
LAp.10	227	LDp.22	181	MDp.21	18	MWEn	101	VSS	33
LAp.11	226	LDp.23	182	MDp.22	19	PARAMOUTp	123	VSS	43
LAp.12	225	LDp.24	183	MDp.23	20	PLLAGND	153	VSS	54
LAp.13	224	LDp.25	186	MDp.24	23	PLLCTRn	145	VSS	64
LAp.14	222	LDp.26	187	MDp.25	24	PLLCTop	149	VSS	70
LAp.15	221	LDp.27	188	MDp.26	27	PLLENp	150	VSS	83
LAp.16	220	LDp.28	189	MDp.27	28	PLLiDDTp	142	VSS	95
LAp.17	217	LDp.29	192	MDp.28	29	PLLLP2p	154	VSS	110
LAp.18	216	LDp.30	193	MDp.29	30	PLLREFp	151	VSS	111
LAp.19	215	LDp.31	194	MDp.30	31	PLLTDp	144	VSS	141
LAp.20	214	LHLDAp	146	MDp.31	32	PLLTSTp	143	VSS	173
LAp.21	213	LHoLDp	239	MDp.32	35	PLLVDD	152	VSS	191
LAp.22	207	LRDn	136	MDp.33	36	PLLVSS	155	VSS	210
LAp.23	206	LRDYn	135	MDp.34	37	SCANCRop	128	VSS	211
LAp.24	205	LRTYn	134	MDp.35	38	SCANENBp	108	VSS	223
LAp.25	204	MAp.0	99	MDp.36	39	SCRESETn	208	VSS	236
LAp.26	203	MAp.1	98	MDp.37	44	SEXiNTn.0	117	VSS	245
LAp.27	202	MAp.2	97	MDp.38	45	SEXiNTn.1	116	VSS	254
LAp.28	201	MAp.3	96	MDp.39	46	SEXiNTn.2	115	VSS2	9
LAp.29	198	MAp.4	94	MDp.40	48	SEXiNTn.3	114	VSS2	26
LAp.30	197	MAp.5	93	MDp.41	49	SEXiNTn.4	113	VSS2	41
LAp.31	195	MAp.6	92	MDp.42	50	SEXiNTn.5	112	VSS2	58
LADSn	133	MAp.7	91	MDp.43	51	SNMin	212	VSS2	73
LBEn.0	119	MAp.8	90	MDp.44	52	SWRESETn	126	VSS2	89
LBEn.1	122	MAp.9	87	MDp.45	53	TESTMp	125	VSS2	105
LBEn.2	240	MAp.10	86	MDp.46	55	VDD	10	VSS2	121
LBEn.3	241	MAp.11	85	MDp.47	56	VDD	42	VSS2	139
LCHALFn	156	MCASn	107	MDp.48	60	VDD	63	VSS2	148
LCLKp	196	MCLKp	100	MDp.49	61	VDD	82	VSS2	169
LCRESETn	132	MCSn.0	103	MDp.50	62	VDD	109	VSS2	185
LDp.0	137	MCSn.1	102	MDp.51	65	VDD	140	VSS2	200
LDp.1	157	MDp.0	242	MDp.52	66	VDD	190	VSS2	219
LDp.2	158	MDp.1	243	MDp.53	67	VDD	209	VSS2	235
LDp.3	159	MDp.2	246	MDp.54	68	VDD	244	VSS2	249
LDp.4	160	MDp.3	247	MDp.55	69	VDD2	8	ZSTATEn	131
LDp.5	161	MDp.4	250	MDp.56	74	VDD2	25		
LDp.6	162	MDp.5	251	MDp.57	75	VDD2	40		
LDp.7	163	MDp.6	252	MDp.58	76	VDD2	57		

1. *NC* pins are not connected.

Customer Feedback

We would appreciate your feedback on this document. Please copy the following page, add your comments, and fax it to us at the number shown.

If appropriate, please also fax copies of any marked-up pages from this document.

Important: Please include your name, phone number, fax number, and company address so that we may contact you directly for clarification or additional information.

Thank you for your help in improving the quality of our documents.

Reader's Comments

Fax your comments to: LSI Logic Corporation
Technical Publications
M/S E-198
Fax: 408.433.4333

Please tell us how you rate this document: *MiniRISC LR4500 Superscalar Microprocessor.* Place a check mark in the appropriate blank for each category.

	Excellent	Good	Average	Fair	Poor
Completeness of information	____	____	____	____	____
Clarity of information	____	____	____	____	____
Ease of finding information	____	____	____	____	____
Technical content	____	____	____	____	____
Usefulness of examples and illustrations	____	____	____	____	____
Overall manual	____	____	____	____	____

What could we do to improve this document?

_____

_____

_____

_____

If you found errors in this document, please specify the error and page number. If appropriate, please fax a marked-up copy of the page(s).

_____

_____

_____

_____

Please complete the information below so that we may contact you directly for clarification or additional information.

Name _____ Date _____

Telephone _____ Fax _____

Title _____

Department _____ Mail Stop _____

Company Name _____

Street _____

City, State, Zip _____

U.S. Distributors by State

H. H. Hamilton Hallmark
W. E. Wyle Electronics

Alabama
Huntsville
H. H. Tel: 205.837.8700
W. E. Tel: 800.964.9953

Alaska
H. H. Tel: 800.332.8638

Arizona
Phoenix
H. H. Tel: 602.736.7000
W. E. Tel: 800.528.4040
Tucson
H. H. Tel: 520.742.0515

Arkansas
H. H. Tel: 800.327.9989

California
Irvine
H. H. Tel: 714.789.4100
W. E. Tel: 800.626.9953
Los Angeles
H. H. Tel: 818.594.0404
W. E. Tel: 800.288.9953
Sacramento
H. H. Tel: 916.632.4500
W. E. Tel: 800.627.9953
San Diego
H. H. Tel: 619.571.7540
W. E. Tel: 800.829.9953
San Jose
H. H. Tel: 408.435.3500
Santa Clara
W. E. Tel: 800.866.9953
Woodland Hills
H. H. Tel: 818.594.0404

Colorado
Denver
H. H. Tel: 303.790.1662
W. E. Tel: 800.933.9953

Connecticut
Chesire
H. H. Tel: 203.271.5700
Wallingford
W. E. Tel: 800.605.9953

Delaware
North/South
H. H. Tel: 800.526.4812
Tel: 800.638.5988

Florida
Fort Lauderdale
H. H. Tel: 305.484.5482
W. E. Tel: 800.568.9953
Orlando
H. H. Tel: 407.657.3300
W. E. Tel: 407.740.7450
Tampa
W. E. Tel: 800.395.9953
St. Petersburg
H. H. Tel: 813.507.5000

Georgia
Atlanta
H. H. Tel: 770.623.4400
W. E. Tel: 800.876.9953

Hawaii
H. H. Tel: 800.851.2282

Idaho
H. H. Tel: 801.266.2022

Illinois
North/South
H. H. Tel: 847.797.7300
Tel: 314.291.5350
Chicago
W. E. Tel: 800.853.9953

Indiana
Indianapolis
H. H. Tel: 317.575.3500
W. E. Tel: 317.581.6152

Iowa
Cedar Rapids
H. H. Tel: 319.393.0033

Kansas
Kansas City
H. H. Tel: 913.663.7900

Kentucky
Central/Northern/ Western
H. H. Tel: 800.984.9503
Tel: 800.767.0329
Tel: 800.829.0146

Louisiana
North/South
H. H. Tel: 800.231.0253
Tel: 800.231.5575

Maine
H. H. Tel: 800.272.9255

Maryland
Baltimore
H. H. Tel: 410.720.3400
W. E. Tel: 800.863.9953

Massachusetts
Boston
H. H. Tel: 508.532.9808
W. E. Tel: 800.444.9953
Marlborough
W. E. Tel: 508.480.9900

Michigan
Detroit
H. H. Tel: 313.416.5800
W. E. Tel: 888.318.9953
Grandville
H. H. Tel: 616.531.0345

Minnesota
Minneapolis
H. H. Tel: 612.881.2600
W. E. Tel: 800.860.9953

Mississippi
H. H. Tel: 800.633.2918

Missouri
St. Louis
H. H. Tel: 314.291.5350

Montana
H. H. Tel: 800.526.1741

Nebraska
H. H. Tel: 800.332.4375

Nevada
Las Vegas
H. H. Tel: 800.528.8471
W. E. Tel: 702.765.7117

New Hampshire
H. H. Tel: 800.272.9255

New Jersey
North/South
H. H. Tel: 201.515.1641
Tel: 609.222.6400
Oradell
W. E. Tel: 201.261.3200
Pine Brook
W. E. Tel: 800.862.9953

New Mexico
Albuquerque
H. H. Tel: 505.293.5119

New York
Long Island
H. H. Tel: 516.434.7400
W. E. Tel: 800.861.9953
Rochester
H. H. Tel: 716.475.9130
W. E. Tel: 800.319.9953
Syracuse
H. H. Tel: 315.453.4000

North Carolina
Raleigh
H. H. Tel: 919.872.0712
W. E. Tel: 800.560.9953

North Dakota
H. H. Tel: 800.829.0116

Ohio
Cleveland
H. H. Tel: 216.498.1100
W. E. Tel: 800.763.9953
Dayton
H. H. Tel: 614.888.3313
W. E. Tel: 800.763.9953

Oklahoma
Tulsa
H. H. Tel: 918.459.6000

Oregon
Portland
H. H. Tel: 503.526.6200
W. E. Tel: 800.879.9953

Pennsylvania
Pittsburgh
H. H. Tel: 412.281.4150
Philadelphia
H. H. Tel: 800.526.4812
W. E. Tel: 800.871.9953

Rhode Island
H. H. 800.272.9255

South Carolina
H. H. Tel: 919.872.0712

South Dakota
H. H. Tel: 800.829.0116

Tennessee
East/West
H. H. Tel: 800.241.8182
Tel: 800.633.2918

Texas
Austin
H. H. Tel: 512.219.3700
W. E. Tel: 800.365.9953
Dallas
H. H. Tel: 214.553.4300
W. E. Tel: 800.955.9953
El Paso
H. H. Tel: 800.526.9238
Houston
H. H. Tel: 713.781.6100
W. E. Tel: 800.888.9953
Rio Grande Valley
H. H. Tel: 210.412.2047

Utah
Draper
W. E. Tel: 800.414.4144
Salt Lake City
H. H. Tel: 801.365.3800
W. E. Tel: 800.477.9953

Vermont
H. H. Tel: 800.272.9255

Virginia
H. H. Tel: 800.638.5988

Washington
Seattle
H. H. Tel: 206.882.7000
W. E. Tel: 800.248.9953

Wisconsin
Milwaukee
H. H. Tel: 414.513.1500
W. E. Tel: 800.867.9953

Wyoming
H. H. Tel: 800.332.9326

Sales Offices and Design Resource Centers

LSI Logic Corporation
Corporate Headquarters
Tel: 408.433.8000
Fax: 408.433.8989

NORTH AMERICA

California
Irvine
♦ Tel: 714.553.5600
Fax: 714.474.8101

San Diego
Tel: 619.613.8300
Fax: 619.613.8350

Silicon Valley
Sales Office
Tel: 408.433.8000
Fax: 408.954.3353
Design Center
♦ Tel: 408.433.8000
Fax: 408.433.7695

Colorado
Boulder
Tel: 303.447.3800
Fax: 303.541.0641

Florida
Boca Raton
Tel: 561.989.3236
Fax: 561.989.3237

Illinois
Schaumburg
♦ Tel: 847.995.1600
Fax: 847.995.1622

Kentucky
Bowling Green
Tel: 502.793.0010
Fax: 502.793.0040

Maryland
Bethesda
Tel: 301.897.5800
Fax: 301.897.8389

Massachusetts
Waltham
♦ Tel: 781.890.0180
Fax: 781.890.6158

Minnesota
Minneapolis
♦ Tel: 612.921.8300
Fax: 612.921.8399

New Jersey
Edison
♦ Tel: 732.549.4500
Fax: 732.549.4802

New York
New York
Tel: 716.223.8820
Fax: 716.223.8822

North Carolina
Raleigh
Tel: 919.785.4520
Fax: 919.783.8909

Oregon
Beaverton
Tel: 503.645.0589
Fax: 503.645.6612

Texas
Austin
Tel: 512.388.7294
Fax: 512.388.4171

Dallas
♦ Tel: 972.509.0350
Fax: 972.509.0349

Washington
Issaquah
Tel: 425.837.1733
Fax: 425.837.1734

Canada
Ontario
Ottawa
♦ Tel: 613.592.1263
Fax: 613.592.3253

Toronto
♦ Tel: 416.620.7400
Fax: 416.620.5005

Quebec
Montreal
♦ Tel: 514.694.2417
Fax: 514.694.2699

INTERNATIONAL

Australia
Reptechnic Pty Ltd
New South Wales
♦ Tel: 612.9953.9844
Fax: 612.9953.9683

Denmark
LSI Logic Development Centre
Ballerup
Tel: 45.44.86.55.55
Fax: 45.44.86.55.56

France
LSI Logic S.A.
Immeuble Europa
Paris
♦ Tel: 33.1.34.63.13.13
Fax: 33.1.34.63.13.19

Germany
LSI Logic GmbH
Munich
♦ Tel: 49.89.4.58.33.0
Fax: 49.89.4.58.33.108

Stuttgart
Tel: 49.711.13.96.90
Fax: 49.711.86.61.428

Hong Kong
AVT Industrial Ltd
Hong Kong
Tel: 852.2428.0008
Fax: 852.2401.2105

India
LogiCAD India Private Ltd
Bangalore
♦ Tel: 91.80.526.2500
Fax: 91.80.338.6591

Israel
LSI Logic
Ramat Hasharon
♦ Tel: 972.3.5.480480
Fax: 972.3.5.403747

VLSI Development Centre
Netanya
♦ Tel: 972.9.657190
Fax: 972.9.657194

Italy
LSI Logic S.P.A.
Milano
♦ Tel: 39.39.687371
Fax: 39.39.6057867

Japan
LSI Logic K.K.
Tokyo
♦ Tel: 81.3.5463.7821
Fax: 81.3.5463.7820

Osaka
♦ Tel: 81.6.947.5281
Fax: 81.6.947.5287

Korea
LSI Logic Corporation of Korea Ltd.
Seoul
♦ Tel: 82.2.528.3400
Fax: 82.2.528.2250

The Netherlands
LSI Logic Europe Ltd
Eindhoven
Tel: 31.40.265.3580
Fax: 31.40.296.2109

Singapore
LSI Logic Pte Ltd
Singapore
♦ Tel: 65.334.9061
Fax: 65.334.4749

Spain
LSI Logic S.A.
Madrid
♦ Tel: 34.1.556.07.09
Fax: 34.1.556.75.65

Sweden
LSI Logic AB
Stockholm
♦ Tel: 46.8.444.15.00
Fax: 46.8.750.66.47

Switzerland
LSI Logic Sulzer AG
Brugg/Biel
Tel: 41.32.536363
Fax: 41.32.536367

Taiwan
LSI Logic Asia-Pacific
Taipei
♦ Tel: 886.2.2718.7828
Fax: 886.2.2718.8869

Jeilin Technology Corporation, Ltd.
Taipei
Tel: 886.2.2248.4828
Fax: 886.2.2242.4397

Lumax International Corporation, Ltd
Taipei
Tel: 886.2.2788.3656
Fax: 886.2.2788.3568

United Kingdom
LSI Logic Europe Ltd
Bracknell
♦ Tel: 44.1344.426544
Fax: 44.1344.481039

♦ Sales Offices with Design Resource Centers