OpenRISC 1200 version 2

Supplementary Programmer's Reference Manual

Jeremy Bennett

Revision 0.0.1
27 Apr 10

Copyright

Revision History

When updating this revision history, ensure the bookmark *revno* is updated to reside on the latest revision number and *revdate* to reside on the revision date.

Revision	Date	Author	Description
0.0.1	27 Apr 10	Jeremy Bennett	Initial draft, covering just PIC

Table of Contents

Acronyms & Abbreviations

ETLA	Extended TLA
PIC	Programmable Interrupt Controller
PICMR	PIC Mask Register
PICSR	PIC Status Register
TLA	Three Letter Acryonym
UPR	Unit Present Register

Table 1-1. Acronyms and Abbreviations

References

1 Damjan Lampret, 6 Sep 01. *OpenRISC 1200 IP Core Specification*, rev 0.7. Available from www.opencores.org.

2 Igor Mohor, 14 Apr 04. *SoC Debug Interface, rev 3.0.* Available from www.opencores.org.

3 OpenCores, 25 Nov 05. *OpenRISC 1000 Architecture Manual*. Available from www.opencores.org.

1 About this Manual

1.1 Introduction

The OpenRISC 1000 system architecture manual [3] defines the architecture for a family of open-source, synthesizable RISC microprocessor cores. The OpenRISC 1200 is a 32-bit implementation of that architecture [1].

A new version of the OpenRISC 1200 was created by engineers from ORSoC AB in 2009 and published at www.opencores.org. This new version makes a number of improvements to the original implementation.

As the basis for design, the architecture and specification documents are fixed. This document exists to capture information about changes from those specifications and to give further detail and clarification where required.

This document is currently an OpenOffice document controlled under Subversion at www.opencores.org. Its dynamic nature means that it would be better represented in a Wiki, when that becomes available.

The chapters of the document match those in the architecture manual for convenience [3]. However this does mean that early revisions have a large number of empty chapters.

1.2 Authors and Contributors

The main authors are shown on the title page. However the contents of this document draw on the contributions of the wider OpenRISC community, who we list here in alphabetical order.

If you have contributed to this manual but your name isn't listed here, it is not meant as a slight—we simply don't know about it. Send an email to the author of the latest revision, and we'll correct the situation.

Name	Contribution
Julius Baxter	Advice on PIC operation
John Eaton	Advice on PIC operation
Raul Fajardo	Advice on PIC operation

Table 1-2. Contributors to this manual.

1.3 Typography

In this manual, fonts are used as follows:

- Programming examples are shown in a fixed width font, `thus`.
- Emphasis is shown *thus*, strong emphasis, **thus**.
- UPPER CASE items may be either acronyms or register mode fields that can be written by software. Some common acronyms appear in the glossary.

- Square brackets [] indicate an addressed field in a register or a numbered register in a register file.

However users should take advantage of the OpenOffice styles which have been created to enforce this.

1.4 Conventions

l.mnemonic	Identifies an ORBIS32/64 instruction.
lv.mnemonic	Identifies an ORVDX32/64 instruction.
lf.mnemonic	Identifies an ORFPX32/64 instruction.
0x	Indicates a hexadecimal number.
rA	Instruction syntax used to identify a general purpose register
REG[FIELD]	Syntax used to identify specific bit(s) of a general or special purpose register. FIELD can be a name of one bit or a group of bits or a numerical range constructed from two values separated by a colon.
X	In certain contexts, this indicates a ‘don't care’.
N	In certain contexts, this indicates an undefined numerical value.
Implementation	An actual processor implementing the OpenRISC 1000 architecture.
Unit	Sometimes referred to as a coprocessor. An implemented unit usually with some special registers and controlling instructions. It can be defined by the architecture or it may be custom.
Exception	A vectored transfer of control to supervisor software through an exception vector table. A way in which a processor can request operating system assistance (division by zero, TLB miss, external interrupt etc).
Privileged	An instruction (or register) that can only be executed (or accessed) when the processor is in supervisor mode (when SR[SM]=1).

Table 1-3. Conventions

1.5 Numbering

All numbers are decimal or hexadecimal unless otherwise indicated. The prefix 0x indicates a hexadecimal number. Decimal numbers don't have a special prefix. Binary and other numbers are marked with their base.

2 Architecture Overview

3 Addressing Modes and Operand Conventions

4 Register Set

5 Instruction Set

6 Exception Model

7 Memory Model

8 Memory Management

9 Cache Model and Cache Coherency

10 Debug Unit (Optional)

The debug interface from a programmer's perspective is unchanged in the OpenRISC 1200 version 2. However a new physical debug interface has been adopted [2]. This will affect those concerned with the representation of JTAG packets to drive the debug interface.

11 Performance Counters Unit (Optional)

12 Power Management (Optional)

13 Programmable Interrupt Controller (Optional)

The OpenRISC 1000 Architecture Manual [] describes an optional simple programmer interrupt controller (PIC) capable of handling up to 32 separate interrupt lines driving the Interrupt Exception. The presence of the PIC is indicated in the Unit Present Register bby UPR[PICP].

The PIC is controlled by two registers, PICMR and PICSR. PICMR is a mask for the incoming interrupts. PICSR is a status register indicating which interrupt lines have been asserted. By convention, interrupts 0 and 1 in PICMR are tied to 1, so that these interrupts lines are non-maskable.

A simple block diagram of the PIC is shown in Figure 13-1.

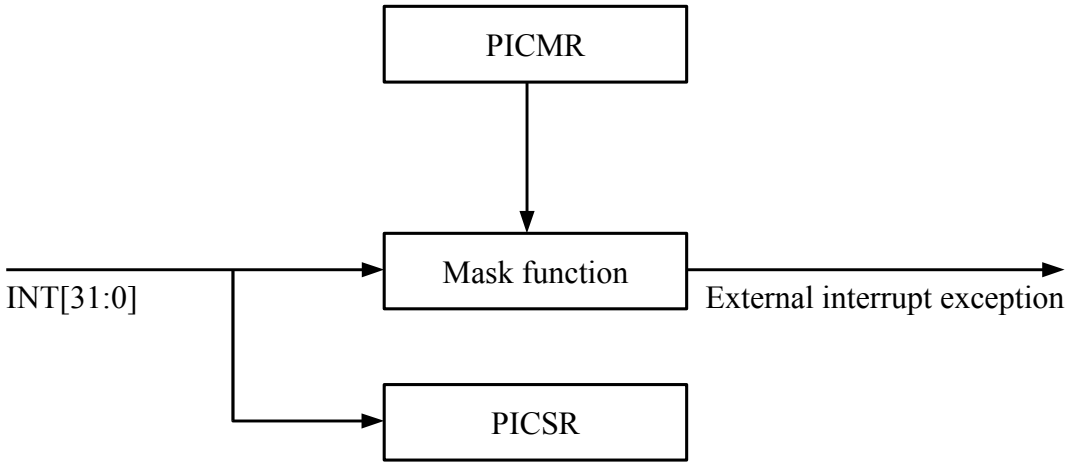

Figure 13-1: Testing

13.1 Functionality

In the OpenRISC 1200 version 2, the functionality has changed. The PIC offers a latched level-sensitive interrupt.

Once an interrupt line is latched (i.e. its value appears in PICSR), no new interrupts can be triggered for that line until its bit in PICSR is cleared. The usual sequence for an interrupt handler is then as follows.

1 Peripheral asserts interrupt, which is latched and triggers handler.

2 Handler processes interrupt.

3 Handler notifies peripheral that the interrupt has been processed (typically via a memory mapped register).

4 Peripheral deasserts interrupt.

5 Handler clears corresponding bit in PICSR and returns.

It is assumed that the peripheral will deassert its interrupt promptly (within 1-2 cycles). Otherwise on exiting the interrupt handler, having cleared PICSR, the level sensitive interrupt will immediately retrigger.

14 Tick Timer Facility (Optional)

15 OpenRISC 1000 Implementations

16 Application Binary Interface

There have been some minor changes in the ABI used with the OpenRISC 1200 version 2.