OpenRISC 1200 IP Core Specification

Author: Damjan Lampret
lampret@opencores.org

Rev. 0.3
April 16, 2001

Preliminary Draft

Revision History

Rev.	Date	Author	Description
0.1	28/3/01	Damjan Lampret	First Draft
0.2	16/4/01	Damjan Lampret	First time published
0.3	29/4/01	Damjan Lampret	All chapters almost finished. Some bugs hidden waiting for an update. Awaiting feedback.

Table Of Contents

Table Of Figures

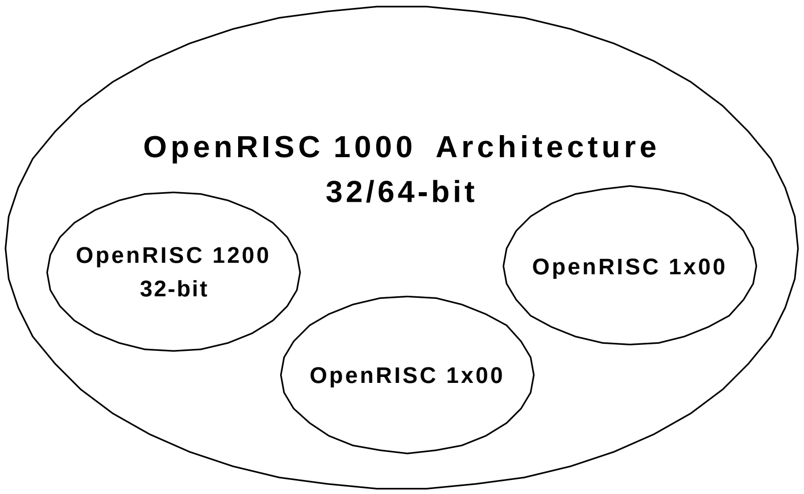
Table Of Tables

1 Introduction

Purpose of this document is to define specifications of the OpenRISC 1200 implementation. This specification defines all implementation specific variables that are not part of the general architecture specification. This includes type and size of data and instruction caches, type and size of data and instruction MMUs, details of all execution pipelines, implementation of exception unit, interrupt controller and other supplemental units.
This document does not cover general architecture topics like instruction set, memory addressing modes and other architectural definitions. See *OpenRISC 1000 System Architecture Manual* for more information about architecture.

OpenRISC Family

OpenRISC 1000 is architecture for a family of free, open source RISC processor cores. As architecture, OpenRISC 1000 allows for a spectrum of chip and system implementations at a variety of price/performance points for a range of applications. It is a 32/64-bit load and store RISC architecture designed with emphasis on performance, simplicity, low power requirements, scalability and versatility. OpenRISC 1000 architecture targets medium and high performance networking, embedded, automotive and portable computer environments.



All OpenRISC implementations, whose first digit in identification number is ‘1’, belong to OpenRISC 1000 family. Second digit defines which features of OpenRISC 1000 architecture are implemented and in which way they are implemented. Last two digits define how an implementations is configured before it is used in a real application.

OpenRISC 1200

The OR1200 is a 32-bit scalar RISC with Harvard microarchitecture, 5 stage integer pipeline, virtual memory support (MMU) and basic DSP capabilities.
Default caches are 2-way set-associative 8KB data cache and 2-way set-associative 8KB instruction cache, each with 16-byte line size. Both caches are physically tagged.
By default MMUs are implemented and they are constructed of 64-entry hash based 2-way set-associative data TLB and 64-entry hash based 2-way set-associative instruction TLB.
Supplemental facilities include debug unit for real-time debugging, high resolution tick timer, programmable interrupt controller and power management support.
When implemented in a typical 0.18u 6LM process it should provide over 400 dhrystone 2.1 MIPS at 400MHz and 400 DSP MAC 32x32 operations, at least 30% more than any other competitor in this class. OR1200 in default configuration has about 1M transistors.

OR1200 is intended for embedded, portable and networking applications. It can successfully compete with latest scalar 32-bit RISC processors in his class and can efficiently run any modern operating system.
Competitors include ARM10, ARC and Tensilica RISC processors.

Features

The following lists the main features of OR1200 IP core:

- All major characteristics of the core can be set by the user
- High performance of 400 Dhrystone 2.1 MIPS at 400 MHz using 0.18u process
- High performance cache and MMU subsystems
- WISHBONE SoC Interconnection Rev. B compliant interface

2 Architecture

Figure 1 below shows general architecture of OR1200 IP core. It consists of several building blocks:

- CPU/DSP central block
- N-way set-associative data cache
- N-way set-associative instruction cache
- Data MMU based on hash based DTLB
- Instruction MMU based on hash based ITLB
- Power management unit and power management interface
- Tick timer
- Debug unit and development interface
- Interrupt controller and interrupt interface
- Instruction and Data WISHBONE host interfaces

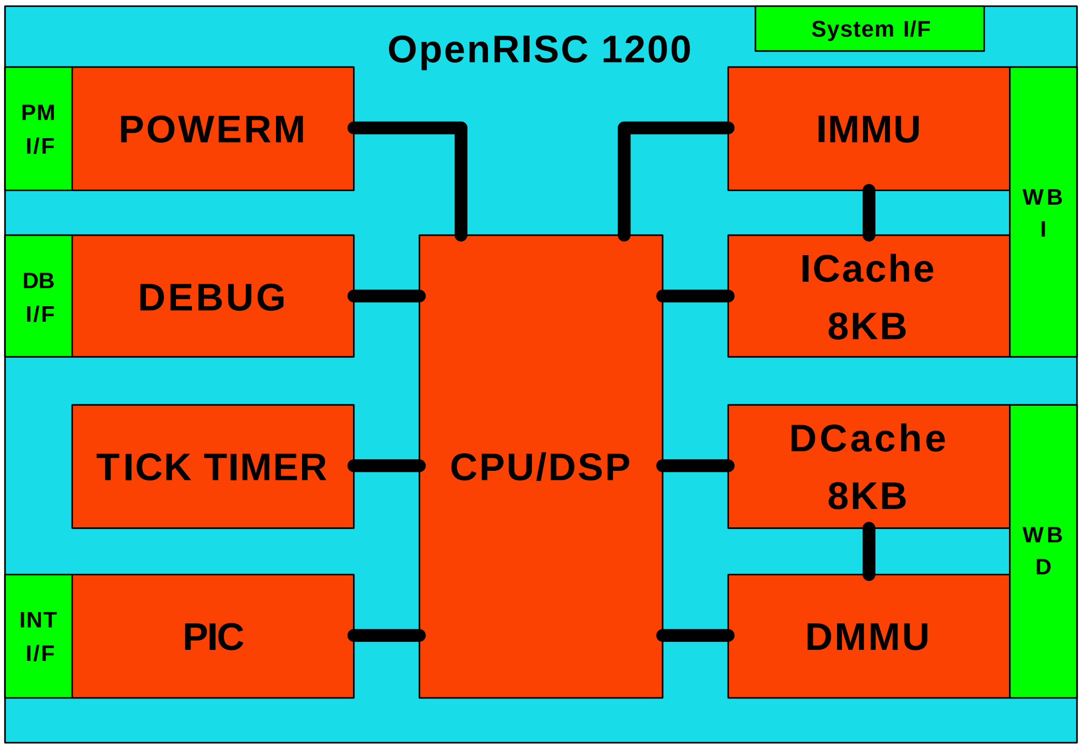


Figure 1. Core's Architecture

CPU/DSP

CPU/DSP is a central part of the OR1200 RISC processor. Figure 2 shows basic block diagram of the CPU/DSP.
OR1200 CPU/DSP implements only 32-bit part of the OpenRISC 1000 architecture. 64-bit part of the architecture as well as floating-point and vector operations are not implemented in OR1200.

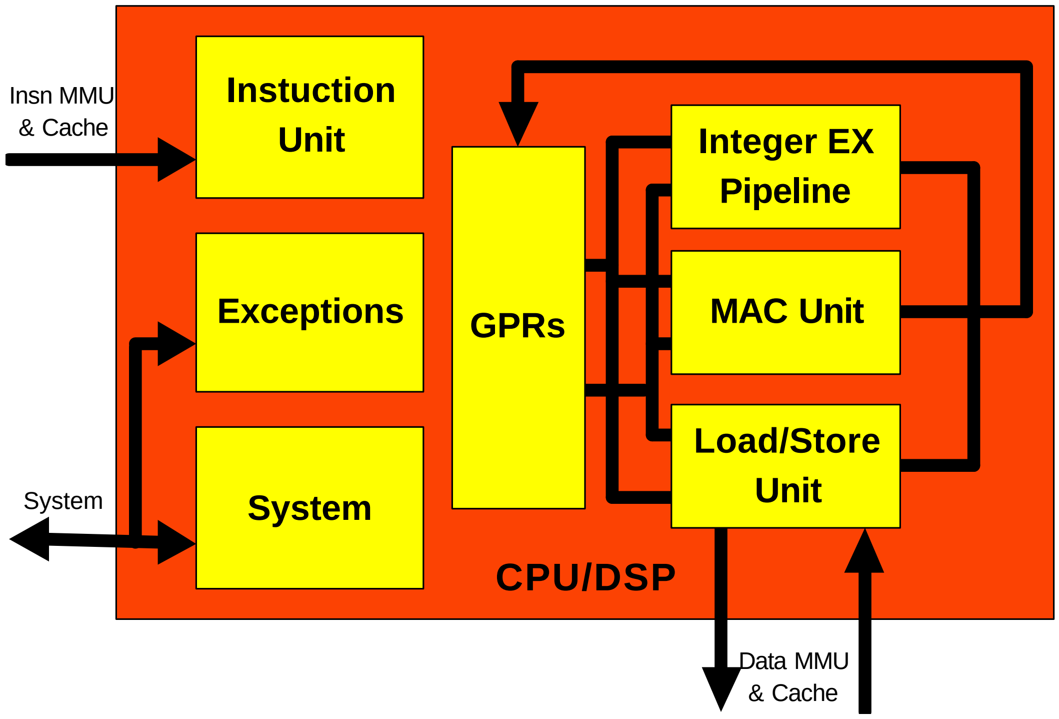


Figure 2. CPU/DSP Block Diagram

Instruction unit

The instruction unit implements the basic instruction pipeline, fetches instructions from the memory subsystem, dispatches them to available execution units, and maintains a state history to ensure a precise exception model and that operations finish in order. It also executes conditional branch and unconditional jump instructions.

The sequencer can dispatch a sequential instruction on each clock if the appropriate execution unit is available. The execution unit must discern whether source data is available and to ensure that no other instruction is targeting the same destination register.

Instruction unit handles only ORBIS32 instruction class. ORFPX32/64 and ORVDX64 instruction classes are not supported by current OR1200.

General-Purpose Registers

OpenRISC 1200 implements 32 general-purpose 32-bit registers. OpenRISC 1000 architecture also support shadow copies of register file to implement fast switching between working contexts, however this feature is not implemented in current OR1200 implementation.

OR1200 implements general-purpose register file as two synchronous dual-port memories with capacity of 32 words by 32 bits per word.

Load/Store Unit

The load/store unit (LSU) transfers all data between the GPRs and the CPU's internal bus. It is implemented as an independent execution unit so that stalls in memory subsystem only affect master pipeline if there is a data dependency.

The following are LSU's main features:

- all load/store instruction implemented in hardware (atomic instructions included)
- address entry buffer
- pipelined operation
- aligned accesses for fast memory access

When load and store instructions are issued, the LSU determines if all operands are available. These operands include the following:

- address register operand
- source data register operand (for store instructions)
- destination data register operand (for load instructions)

Integer Execution Pipeline

The core implements the following types of 32-bit integer instructions:

- Arithmetic instructions
- Compare instructions

- Logical instructions
- Rotate and shift instructions

Most integer instructions can execute in one cycle. For details about timing see table TBD.

MAC Unit

The MAC unit executes DSP MAC operations. MAC operations are 32x32 with 48-bit accumulator. MAC unit is fully pipelined and can accept new MAC operation in each new clock cycle.

System Unit

The system unit connects all other signals of the CPU/DSP that are not connected through instruction and data interfaces. It also implements all system special-purpose registers (e.g. supervisor register).

Exceptions

Core exceptions can be generated when an exception condition occurs. Exception sources in OR1200 include the following:

- External interrupt request
- Certain memory access condition
- Internal errors, such as an attempt to execute unimplemented opcode
- System call
- Internal exception, such as breakpoint exceptions

Exception handling is transparent to user software and uses the same mechanism to handle all types of exceptions. When an exception is taken, control is transferred to an exception handler at an offset defined by for the type of exception encountered. Exceptions are handled in supervisor mode.

Data Cache

The default configuration of OR1200 data cache is 8-Kbyte, two-way set associative data cache, which allows rapid core access to data. However data cache can be configured according to the Table 1.

	Direct mapped	2-way set associative	4-way set associative	8-way set associative
1KB per set	1KB	2KB	4KB	8KB
2KB per set	2KB	4KB	8KB	16KB
4KB per set	4KB	**8KB (default)**	16KB	32KB
8KB per set	8KB	16KB	32KB	64KB

Table 1. Possible Data Cache Configurations of OR1200

Features:

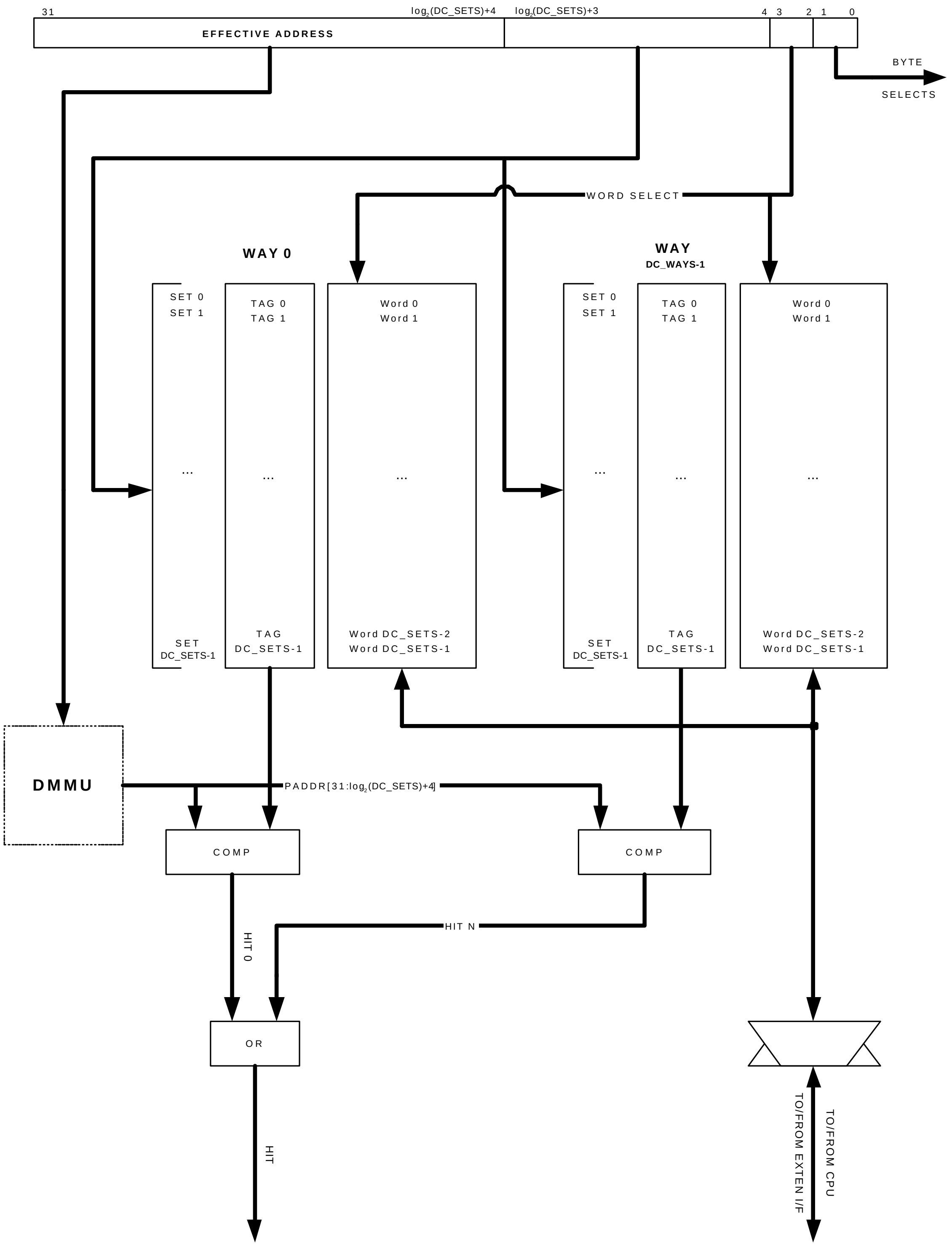
- data cache is separate from instruction cache (Harvard architecture)
- 1-4 way set associative with default of 2
- data cache implements a least-recently used (LRU) replacement algorithm within each set
- the cache directory is physically addressed. The physical address tag is stored in the cache directory
- write-through operation
- it can be disabled, invalidated or locked by writing to cache special purpose registers
- individual cache lines can be locked so that frequently accessed data are guaranteed to be resident in the cache

On a miss, the cache is filled in with 16-byte bursts. The burst fill is performed as a critical-word-first operation; the critical word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to cache fill latency. Data cache provides storage for cache tags and performs cache line replacement function.

Data cache is tightly coupled to external interface to allow efficient access to the system memory controller.

The data cache supplies data to the GPRs by means of a 32-bit interface to the load/store unit. The LSU provides all logic required to calculate effective addresses, handles data alignment to and from the data cache, and provides sequencing for load and store operations. Write operations to the data cache can be performed on a byte, half-word or word basis.

The data cache is organized as 256 sets of two lines. Each line consists of 16 bytes, state bits and an address tag.



Each line contains four contiguous words from memory that are loaded from a four-word aligned boundary. As a result, cache lines are aligned with page boundaries.

Instruction Cache

The default configuration of OR1200 instruction cache is 8-Kbyte, two-way set associative instruction cache, which allows rapid core access to instructions. However instruction cache can be configured according to the Table 2.

	Direct mapped	2-way set associative	4-way set associative	8-way set associative
1KB per set	1KB	2KB	4KB	8KB
2KB per set	2KB	4KB	8KB	16KB
4KB per set	4KB	**8KB (default)**	16KB	32KB
8KB per set	8KB	16KB	32KB	64KB

Table 2. Possible Instruction Cache Configurations of OR1200

Features:

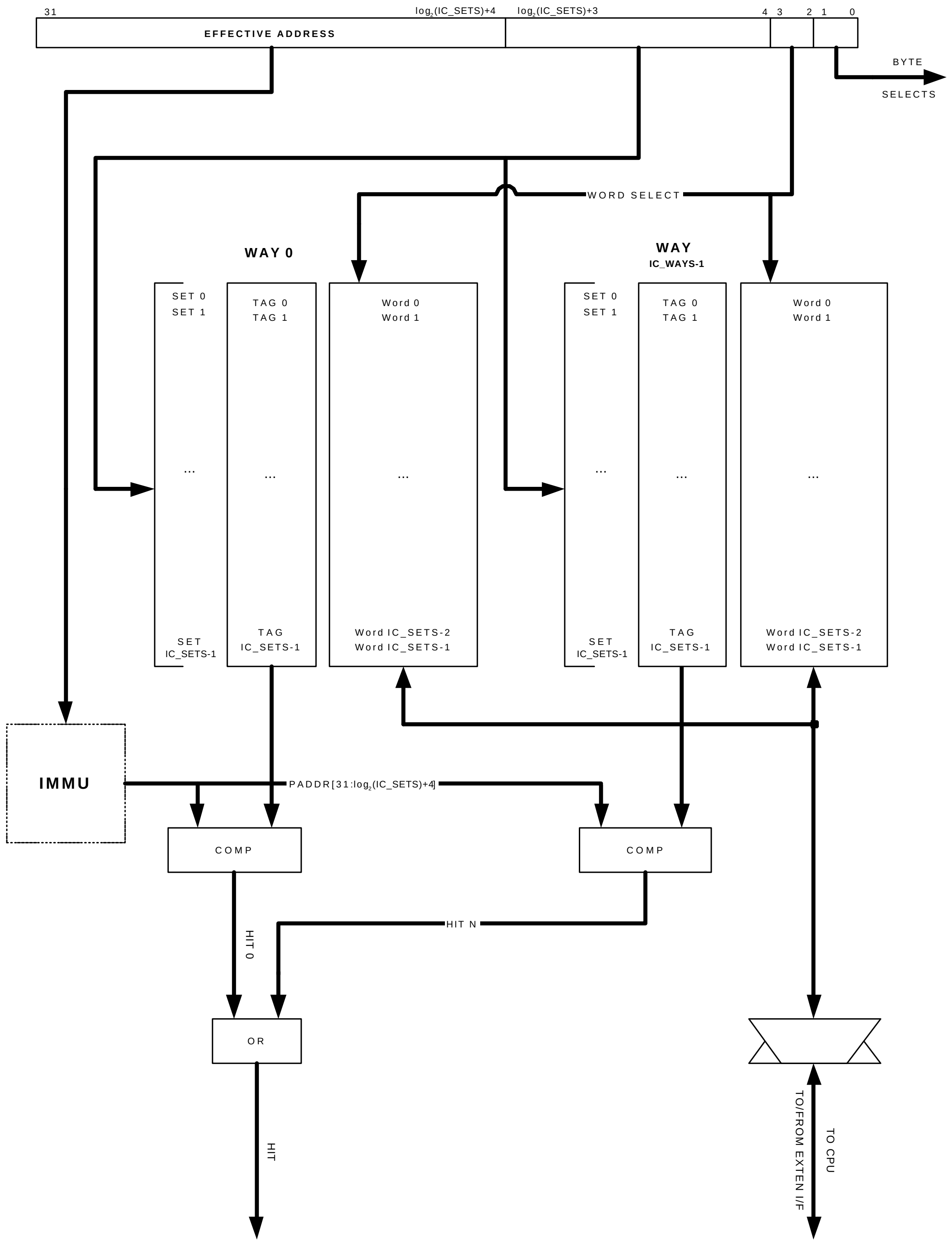
- instruction cache is separate from data cache (Harvard architecture)
- 1-4 way set associative with default of 2
- instruction cache implements a least-recently used (LRU) replacement algorithm within each set
- the cache directory is physically addressed. The physical address tag is stored in the cache directory
- it can be disabled, invalidated or locked by writing to cache special purpose registers
- individual cache lines can be locked so that frequently accessed instructions are guranteed to be resident in the cache

On a miss, the cache is filled in with 16-byte bursts. The burst fill is performed as a critical-word-first operation; the critical word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to cache fill latency. Instruction cache provides storage for cache tags and performs cache line replacement function.

Instruction cache is tightly coupled to external interface to allow efficient access to the system memory controller.

The instruction cache supplies instructions to the instruction sequencer by means of a 32-bit interface to the instruction fetch subunit. The instruction fetch subunit provides all logic required to calculate effective addresses.

The data cache is organized as 256 sets of two lines. Each line consists of 16 bytes, state bits and an address tag.



Each line contains four contiguous words from memory that are loaded from a four-word aligned boundary. As a result, cache lines are aligned with page boundaries.

Data MMU

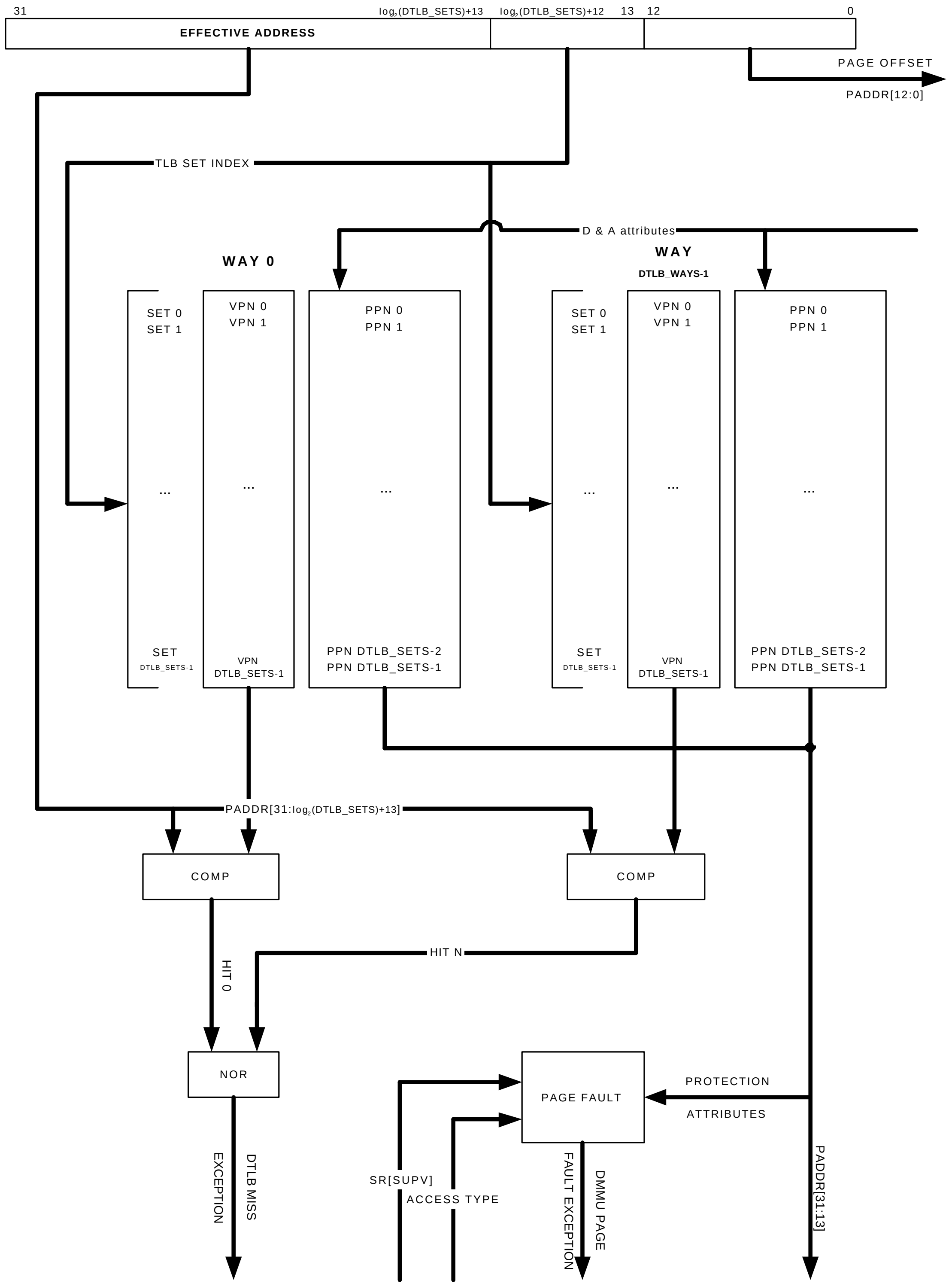
The OR1200 implements a virtual memory management scheme that provides cache control, memory access protection and effective-to-physical address translation. Protection granularity is as defined by OpenRISC 1000 architecture - 8-Kbyte and 16-Mbyte pages.

	Direct mapped	2-way set associative
16 entries per way	16 DTLB entries	32 DTLB entries
32 entries per way	32 DTLB entries	**64 DTLB entries (default)**
64 entries per way	64 DTLB entries	128 DTLB entries
128 entries per way	128 DTLB entries	256 DTLB entries

Table 3. Possible Data TLB Configurations of OR1200

Features:

- data MMU is separate from instruction MMU
- two pages sizes - 8-Kbyte and 16-Mbyte
  - dirty bit
  - accessed bit
  - caching-inhibited attribute
- comprehensive page protection scheme
- 1-2 way set associative hash based translation lookaside buffer (DTLB) with the default of 2 ways and the following features:
  - miss and fault exceptions
  - software tablewalk
  - locking of one DTLB sets to ensure fast translation of critical data
  - high performance because of hashed based design
  - variable number DTLB entries with default of 32 per each way



The MMU hardware supports two-level software tablewalk.

Instruction MMU

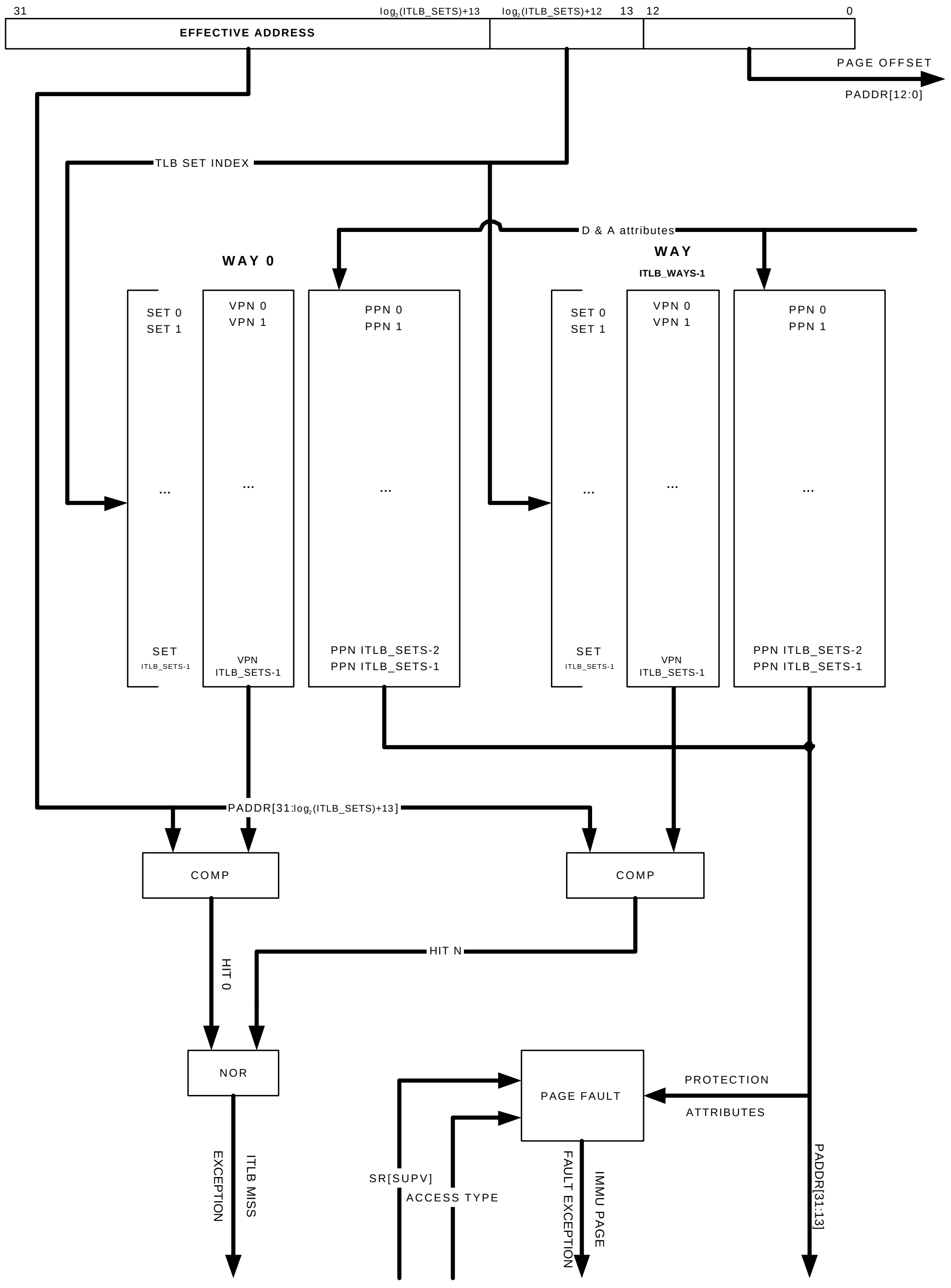
The OR1200 implements a virtual memory management scheme that provides cache control, memory access protection and effective-to-physical address translation. Protection granularity is as defined by OpenRISC 1000 architecture - 8-Kbyte and 16-Mbyte pages.

	Direct mapped	2-way set associative
16 entries per way	16 ITLB entries	32 ITLB entries
32 entries per way	32 ITLB entries	**64 ITLB entries (default)**
64 entries per way	64 ITLB entries	128 ITLB entries
128 entries per way	128 ITLB entries	256 ITLB entries

Table 4. Possible Instruction TLB Configurations of OR1200

Features:

- instruction MMU is separate from data MMU
- two pages sizes - 8-Kbyte and 16-Mbyte
    - accessed bit
- comprehensive page protection scheme
- 1-2 way set associative hash based translation lookaside buffer (ITLB) with the default of 2 ways and the following features:
    - miss and fault exceptions
    - software tablewalk
    - locking of one ITLB sets to ensure fast translation of critical data
    - high performance because of hashed based design
    - Variable number of ITLB entries with default of 32 entries per way



The MMU hardware supports two-level software tablewalk.

Programmable Interrupt Controller

The interrupt controller receives interrupts from external sources and forwards them as low or high priority interrupt exception to the CPU core.

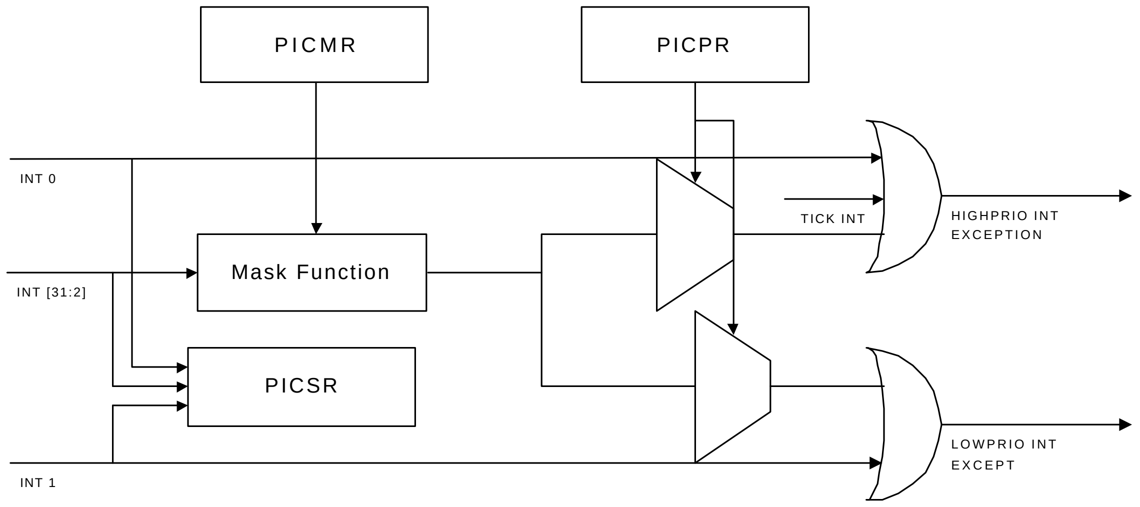


Table 5. Block Diagram of the Interrupt Controller

Programmable interrupt controller has three special-purpose registers and 32 interrupt inputs. Interrupt input 0 and 1 are always enabled and connected to high and low priority interrupt input, respectively.

30 other interrupt inputs can be masked and assigned low or high priority through programming special-purpose registers.

Tick Timer

OR1200 implements tick timer facility. Basically this is a timer that is clocked by RISC clock and is used by the operating system to precisely measure time and schedule system tasks.

OR1200 precisely follow architectural definition of the tick timer facility:

- Maximum timer count of 2^32 clock cycles
- Maximum time period of 2^28 clock cycles between interrupts
- Maskable tick timer interrupt
- Single run or restartable timer

Tick timer operates from independent clock source so that doze power management mode can be implemented.

Power Management Support

To optimize power consumption, the OR1200 provides low-power modes that can be used to dynamically activate and deactivate certain internal modules.

OR1200 has three major features to minimize power consumption:

- Slow and Idle Modes (SW controlled clock freq reduction)
- Doze and Sleep Modes (interrupt wake-up)
- Dynamic Clock gating (on clock by clock basis unit clock gating)

Power Minimization Feature	Approx Power Consumption Reduction
Slow and Idle mode	2x – 10x
Doze mode	100x
Sleep mode	200x
Dynamic clock gating	2x – 4x

Table 6. Power Consumption

Slow down mode takes advantage of the low-power dividers in external clock generation circuitry to enable full functionality, but at a lower frequency so that a power consumption is reduced.
PMR[SDF] 4 bits are broadcasted on **pm_clksd** and external clock generation for the RISC should adapt RISC clock frequency according to the value on **pm_clksd**.

When software initiates the doze mode, software processing on the core suspends. The clocks to the RISC internal modules are disabled except to the tick timer. However any other on-chip blocks can continue to function as normal.
The OR1200 will leave doze mode and enter normal mode when a pending interrupt occurs.

In sleep mode, all OR1200 internal units are disabled and clocks gated. Optionally implementation may choose to lower the operating voltage of the OR1200 core.
The OR1200 should leave sleep mode and enter normal mode when a pending interrupt occurs.

If enabled, the clock gating feature automatically disables clock subtrees to major RISC internal blocks on a clock cycle basis. These blocks are CPU, IC, DC, IMMU and DMMU. This mode can be used in a combination with other low-power modes.
Cache or MMU blocks that are already disabled when software enables this mode, have completely disabled clock subtrees until clock gating is disabled or until the blocks are again enabled.

Debug unit

Debug unit assists software developers to debug their systems. It provides support for watchpoints, breakpoints and program-flow control registers.

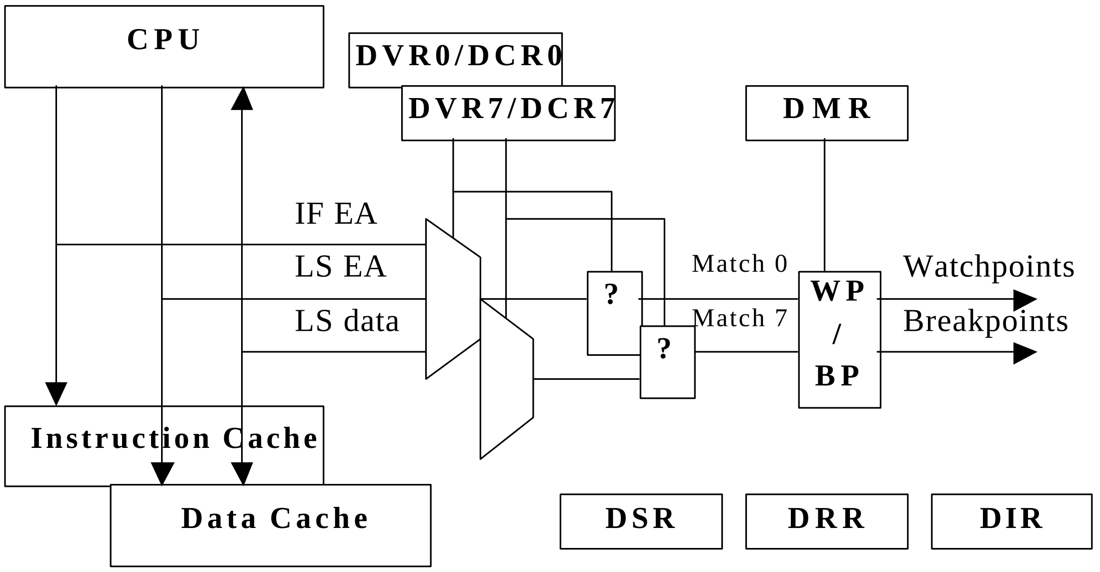


Figure 3. Block Diagram of Debug Unit

Watchpoints and breakpoints are events triggered by program- or data-flow matching the conditions programmed in the debug registers. Breakpoints unlike watchpoints also suspend execution of the current program-flow and start breakpoint exception.

Clocks & Reset

The OR1200 core has several clock inputs. Clock input clk_cpu clocks CPU/DSP block and all other parts of the RISC that do not have separate clocks. Data cache is clocked by clk_dc, instruction cache is clocked by clk_ic, data MMU is clocked by clk_dmmu, instruction MMU is clocked by clk_immu and tick timer is clocked by clk_tt. All clocks must have the same phase and as low clock skew as possible.

OR1200 has asynchronous reset signal. Reset signal *rst*, when asserted high, immediately resets all flip-flops inside OR1200. When deasserted, OR1200 will start reset exception.

WISHBONE Interfaces

WISHBONE interfaces connect OR1200 core to external peripherals and external memory subsystem. They are WISHBONE SoC Interconnection specification Rev. B compliant. The implementation implements a 32-bit bus width and does not support other bus widths.

WISHBONE COMPATIBLE

3 Operation

This section describes the operation of the OR1200 core. For operations that pertain to the architectural definitions, see *OpenRISC 1000 System Architecture Manual*.

Reset

OR1200 has one asynchronous reset signal that can be used by a soft and hard reset on a higher system hierarchy levels.

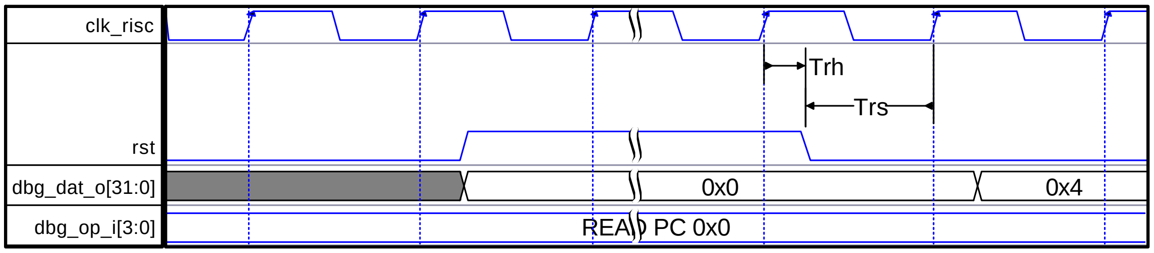


Figure 4. Power-Up and Reset Sequence

Figure 4 shows how asynchronous reset is applied after powering up the OR1200 core. Reset is connected to asynchronous reset of almost all flip-flops inside RISC core. Special care must be taken to ensure hold and setup times of all flip-flops compared to main RISC clock.

If system implements gated clocks, then clock gating can be used to ensure proper reset timing.

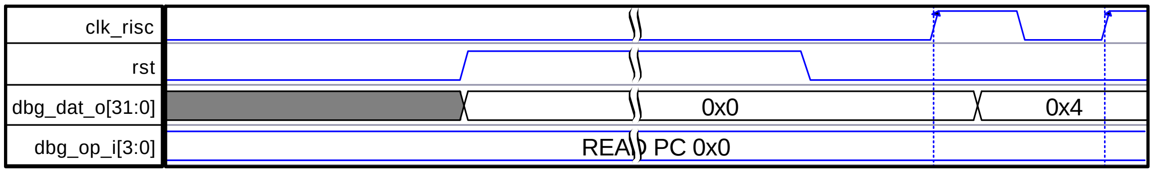


Figure 5. Power-Up and Reset Sequence w/ Gated Clock

CPU/DSP

CPU/DSP is implementation of the 32-bit part of the OpenRISC 1000 architecture and only a subset of all features is implemented.

Instructions

Table 7 shows all instructions implemented by OR1200.

Insn	31	31	31	31	31	26	25	25	25	25	21	20	20	20	20	16	15	15	15	15	11	10	10	8	7	6	5	4	3	3	3	0
l.add	Opcode 0x38						D					A					B					reserved			opcode 0x0		reserved		opcode 0x0			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.addc	opcode 0x38						D					A					B					reserved			opcode 0x0		reserved		opcode 0x1			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.addi	opcode 0x25						D					A					I															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.and	opcode 0x38						D					A					B					reserved			opcode 0x0		reserved		opcode 0x3			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.andi	opcode 0x28						D					A					K															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.bf	Opcode 0x4						N																									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.bnf	opcode 0x3						N																									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.j	opcode 0x0						N																									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.jal	opcode 0x1						N																									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.jalr	opcode 0x12						reserved										B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.jr	opcode 0x11						reserved										B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.lbs	opcode 0x22						D					A					I															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

l.lbz	opcode 0x21						D					A					I															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.lhs	opcode 0x24						D					A					I															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.lhz	opcode 0x23						D					A					I															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.movhi	opcode 0x6						D					reserved					K															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.mul	opcode 0x38						D					A					B					reserved			opcode 0x3		reserved		opcode 0x6			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.muli	opcode 0x2b						D					A					I															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.mulu	opcode 0x38						D					A					B					reserved			opcode 0x3		reserved		opcode 0xb			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.nop	opcode 0x15								reserved																							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.or	opcode 0x38						D					A					B					reserved			opcode 0x0		reserved		opcode 0x4			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.ori	opcode 0x29						D					A					K															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.rfe	opcode 0x9						Reserved																									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.rori	opcode 0x2d						D					A					reserved								opcode 0x4			L				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sb	opcode 0x36						I					A					B					I										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfeq	opcode 0x720											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfges	opcode 0x72b											A					B					reserved										

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfgeu	opcode 0x723											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfgts	opcode 0x72a											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfgtu	opcode 0x722											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfles	opcode 0x72d											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfleu	opcode 0x725											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sflts	opcode 0x72c											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfltu	opcode 0x724											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sfne	opcode 0x721											A					B					reserved										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sh	opcode 0x37						I					A					B					I										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sll	opcode 0x38						D					A					B					reserved			opcode 0x8							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.slli	opcode 0x2d						D					A					reserved								opcode 0x0			L				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.sra	opcode 0x38						D					A					B					reserved			opcode 0x28							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.srai	opcode 0x2d						D					A					reserved								opcode 0x2			L				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l.srl	opcode 0x38						D					A					B					reserved			opcode 0x18							
	3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	10	9	8	7	6	5	4	3	2	1	0

	1 0 9 8 7 6	5 4 3 2 1	0 9 8 7 6	5 4 3 2 1				
l.srli	opcode 0x2d	D	A	reserved		opcode 0x1	L	
	31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8	7 6	5 4	3 2 1 0
l.sub	opcode 0x38	D	A	B	reserved	opcode 0x0	reserved	opcode 0x2
	31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8	7 6	5 4	3 2 1 0
l.subi	opcode 0x27	D	A	I				
	31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8	7 6	5 4	3 2 1 0
l.sw	opcode 0x35	I	A	B	I			
	31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8	7 6	5 4	3 2 1 0
l.sys	opcode 0x20000				K			
	31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8	7 6	5 4	3 2 1 0
l.xor	opcode 0x38	D	A	B	reserved	opcode 0x0	reserved	opcode 0x5
	31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8	7 6	5 4	3 2 1 0
l.xori	Opcode 0x2a	D	A	I				

Table 7. List of Implemented 32-bit Instructions

For a complete description how instruction operate refer to *OpenRISC 1000 System Architecture Manual*.

Instruction Unit

Instruction unit generates instruction fetch effective address and fetches instructions from instruction cache. Each clock cycle one instruction can be fetched. Instruction fetch EA is further translated into physical address by IMMU.

General-Purpose Registers

General-purpose register file can supply two read operands each clock cycle and store one result in a destination register.

GPRs can be also read and written through development interface.

Load/Store Unit

LSU can execute one load instruction every two clock cycles assuming load instruction have a hit in the data cache. Execution of store instructions takes one clock cycle assuming they have a hit in the data cache.

LSU performs calculation of the load/store effective address. EA is furher translated into physical address by DMMU.

Load/store effective address and load and store data can be also accessed through development interface.

Integer Execution Pipeline

The core implements the following types of 32-bit integer instructions:

- Arithmetic instructions
- Compare instructions
- Logical instructions
- Rotate and shift instructions

Instruction Group	Clock Cycles to Execute
Arithmetic except Multiply/Divide	1
Multiply	3
Divide	Not implemented
Compare	1
Logical	1
Rotate and Shift	1
Others	1

Table 8. Execution Time of Integer Instructions

Table 8 lists execution times for instructions executed by integer execution pipeline. Most instructions are executed in one clock cycle.

MAC Unit

MAC unit executes l.mac instructions. MAC unit implements 32x32 fully pipelined multiplier and 48-bit accumulator. MAC unit can accept one new l.mac instruction each clock cycle.

System Unit

System unit implements system control and status special-purpose registers and executes all l.mtspr/l.mfspr instructions.

Exceptions

The core implements a precise exception model. This means that when an exception is taken, the following conditions are met:

- Subsequent instructions in program flow are discarded
- Previous instructions finish and write back their results
- The address of faulting instruction is saved in EPCR registers and the machine state is saved to ESR registers

EXCEPTION TYPE	VECTOR OFFSET	CAUSING CONDITIONS
Reset	0x100	Caused by reset.
Bus Error	0x200	Caused by an attempt to access invalid physical address.
Data Page Fault	0x300	Generated artificially by DTLB miss exception handler when no matching PTE found in page tables or page protection violation for load/store operations.
Instruction Page Fault	0x400	Generated artificially by ITLB miss exception handler when no matching PTE found in page tables or page protection violation for instruction fetch.
Low Priority External Interrupt	0x500	Low priority external interrupt asserted.
Alignment	0x600	Load/store access to naturally not aligned location.
Illegal Instruction	0x700	Illegal instruction in the instruction stream.
High Priority External Interrupt	0x800	High priority external interrupt asserted.
D-TLB Miss	0x900	No matching entry in DTLB (DTLB miss).
I-TLB Miss	0xA00	No matching entry in ITLB (ITLB miss).
System Call	0xC00	System call initiated by software.
Breakpoint	0xD00	Initiated by the debug unit.

Table 9. List of Implemented Exceptions

The OR1200 exception support does not include support for fast context switching.

Data Cache Operation

Data Cache Load/Store Access

Load/store unit requests data from the data cache and stores them into the general-purpose register file and forwards them to integer execution units. Therefore LSU is tightly coupled with the data cache.

If there is no data cache line miss nor DTLB miss, load operations take two clock cycles to execute and store operations take one clock cycle to execute. LSU does all the data alignment work.

Data can be written to the data cache on a word, half-word or byte basis. Since data cache only operates in write-through mode, all writes are immediately written back to main memory or to the next level of caches.

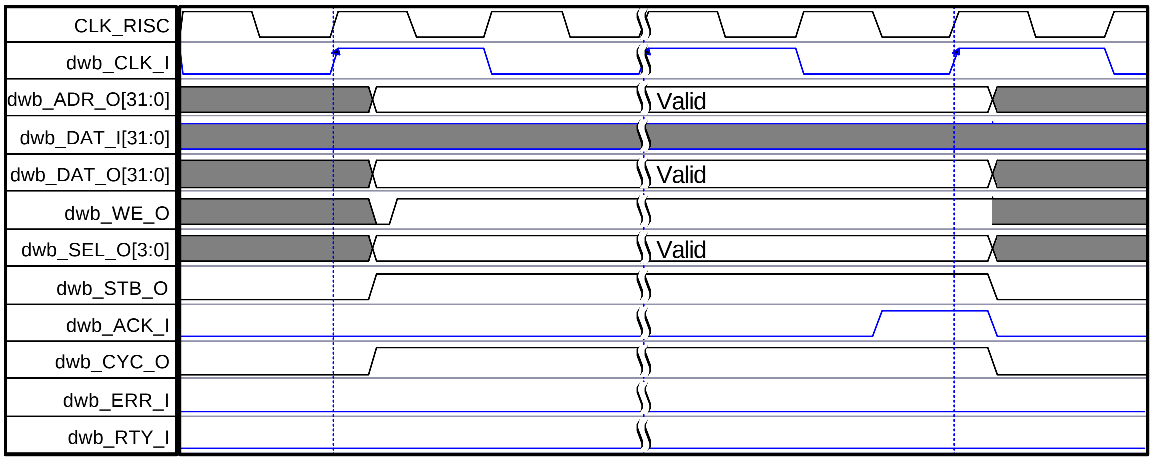


Figure 6. WISHBONE Write Cycle

Figure 6 shows how a write-through cycle on data WISHBONE interface is performed when a store instruction hits in the data cache.
If **dwb_ERR_I** or **dwb_RTY_I** is asserted instead of usual **dwb_ACK_I**, bus error exception is invoked.

Data Cache Line Fill Operation

When executing load instruction and a cache miss occurs, a 4 beat sequential read burst with critical word first is performed. Critical word is forwarded to the load/store unit to minimize performance loss because of the cache miss.

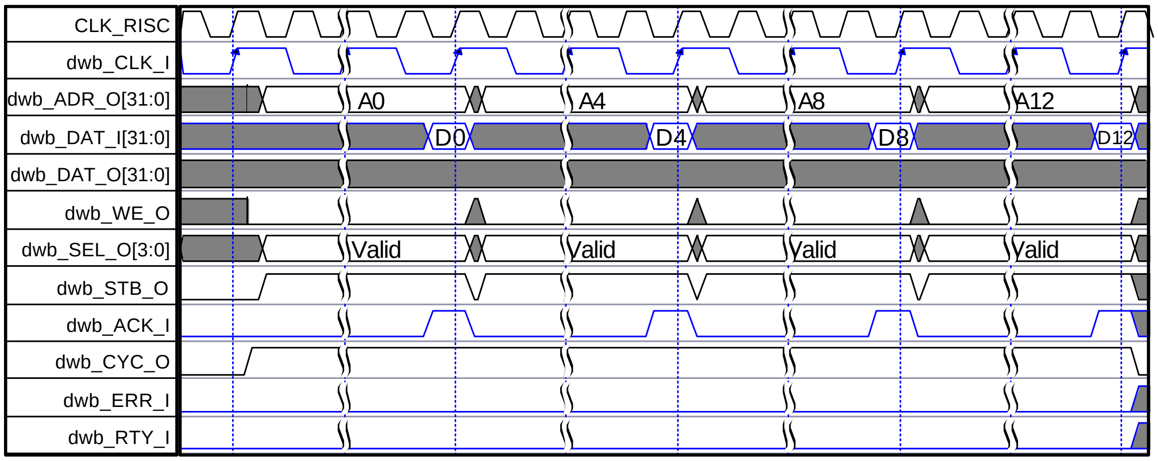


Figure 7. WISHBONE Block Read Cycle

Figure 7 shows how a cache line is read in WISHBONE read block cycle composed out of four read transfers.
If **dwb_ERR_I** or **dwb_RTY_I** is asserted instead of usual **dwb_ACK_I**, bus error exception is invoked.

When executing store instruction and a cache miss occurs, a 4 beat sequential read burst with critical word first is performed. After read burst single word write is performed to write data of the store instruction back to main memory or next level of caches. Regardless of the wideness of store instruction, always a word write is performed.

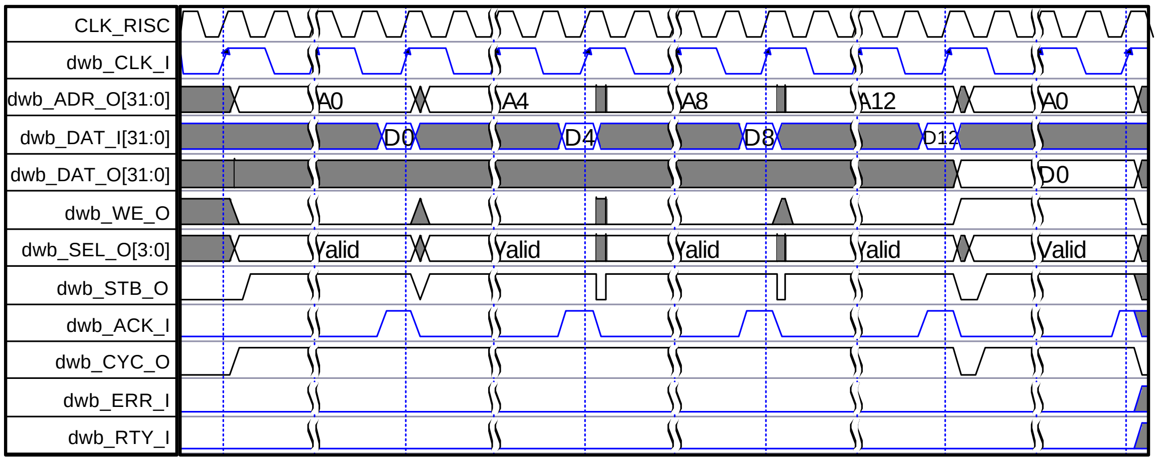


Figure 8. WISHBONE Block Read/Write Cycle

Figure 8 shows how a cache line is read in WISHBONE read block cycle followed by a write transfer.
If **dwb_ERR_I** or **dwb_RTY_I** is asserted instead of usual **dwb_ACK_I**, bus error exception is invoked.

Cache/Memory Coherency

Data cache in OR1200 operates only in write-through mode. Furthermore OR1200 is not intended for use in multiprocessor environments. Therefore no support for coherency between local data cache and caches of other processors or main memory is implemented.

Data Cache Enabling/Disabling

Data cache is disabled at power up. Entire data cache can be enabled by setting bit SR[DCE] to one. Before data cache is enabled, it must be invalidated.

Data Cache Invalidation

Data cache in OR1200 does not support invalidation of entire data cache. Normal procedure to invalidate entire data cache is to cycle through all data cache lines and invalidate each line separately.

Data Cache Locking

Data cache implements way locking bits in data cache control register DCCR. Bits LWx lock individual ways when they are set to one.

Data Cache Line Prefetch

Data cache line prefetch is optional in the OpenRISC 1000 architecture and is not implemented in OR1200.

Data Cache Line Flush

Because data cache operates only in write-through mode, data cache line flush performs only line invalidation. Operation is performed by writing effective address to the DCBFR register.

Virtually the is no difference between data cache line flush and data cache line invalidate operation.

Data Cache Line Invalidate

Data cache line invalidate invalidates a single data cache line. Operation is performed by writing effective address to the DCBIR register.

Data Cache Line Write-back

Data cache line write-back operation does not do anything because data cache operates only in write-through mode.

Data Cache Line Lock

Locking of individual data cache lines is not implemented in OR1200.

Instruction Cache Operation

Instruction Cache Instruction Fetch Access

Instruction unit requests instruction from the instruction cache and forwards them to the instruction queue inside instruction unit. Therefore instruction unit is tightly coupled with the instruction cache.

If there is no instruction cache line miss nor ITLB miss, instruction fetch operation takes one clock cycle to execute.

Instruction cache cannot be explicitly modified like data cache can be with store instructions.

Instruction Cache Line Fill Operation

On a cache miss, a 4 beat sequential read burst with critical word first is performed. Critical word is forwarded to the instruction unit to minimize performance loss because of the cache miss.

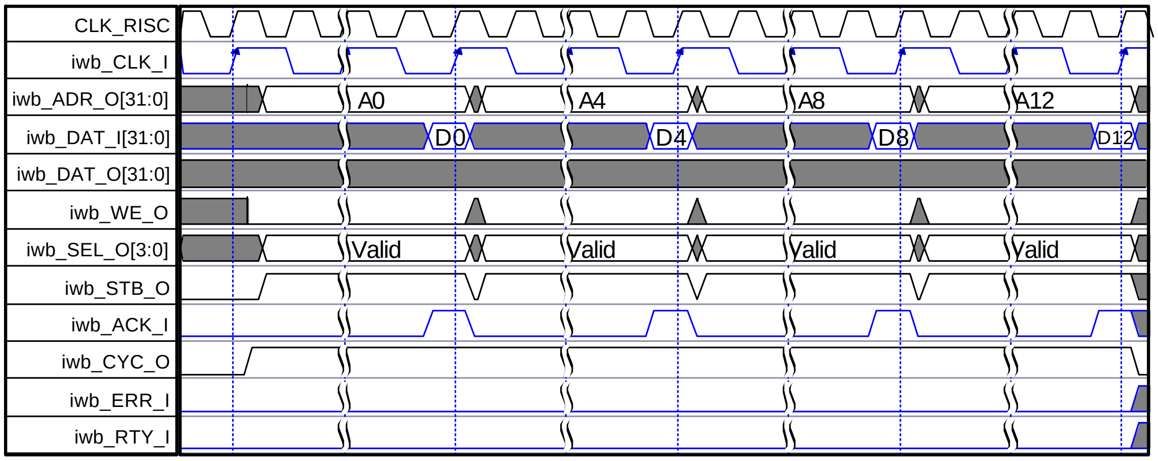


Figure 9. WISHBONE Block Read Cycle

Figure 9 shows how a cache line is read in WISHBONE read block cycle composed out of four read transfers.
If **iwb_ERR_I** or **iwb_RTY_I** is asserted instead of usual **dwb_ACK_I**, bus error exception is invoked.

Cache/Memory Coherency

OR1200 is not intended for use in multiprocessor environments. Therefore no support for coherency between local instruction cache and caches of other processors or main memory is implemented.

Instruction Cache Enabling/Disabling

Instruction cache is disabled at power up. Entire instruction cache can be enabled by setting bit SR[ICE] to one. Before instruction cache is enabled, it must be invalidated.

Instruction Cache Invalidation

Instruction cache in OR1200 does not support invalidation of entire instruction cache. Normal procedure to invalidate entire instruction cache is to cycle through all instruction cache lines and invalidate each line separately.

Instruction Cache Locking

Instruction cache implements way locking bits in instruction cache control register ICCR. Bits LWx lock individual ways when they are set to one.

Instruction Cache Line Prefetch

Instruction cache line prefetch is optional in the OpenRISC 1000 architecture and is not implemented in OR1200.

Instruction Cache Line Invalidate

Instruction cache line invalidate invalidates a single instruction cache line. Operation is performed by writing effective address to the ICBIR register.

Instruction Cache Line Lock

Locking of individual instruction cache lines is not implemented in OR1200.

Data MMU

Translation Disabled

Load/store address translation can be disabled by clearing bit SR[DME]. If translation is disabled, then physical address used to access data cache and optionally provided on **dwb_ADDR_O**, is the same as load/store effective address.

Translation Enabled

Load/store address translation can be enabled by setting bit SR[DME]. If translation is enabled, it provides load/store effective address to physical address translation, page protection and page attributes for memory accesses.

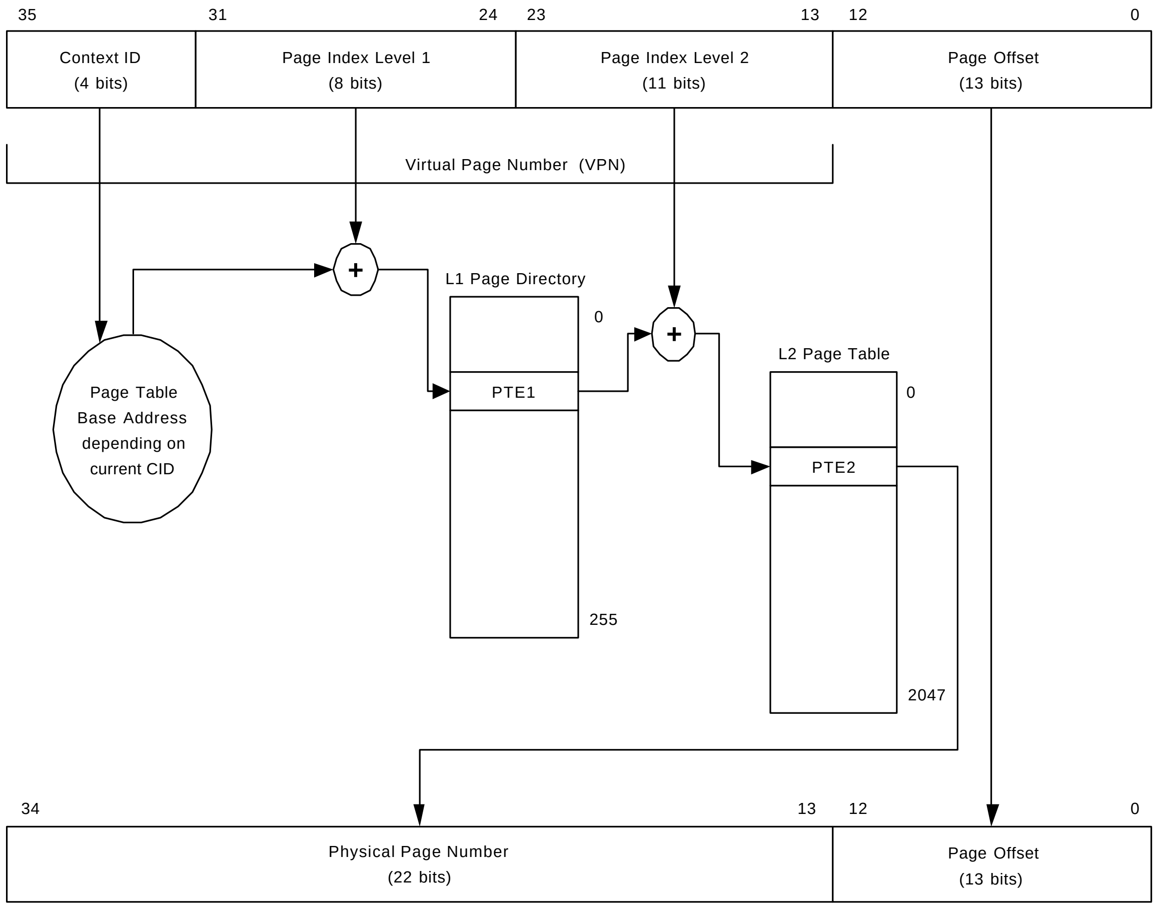

35
31
24
23
13
12
0
Context ID (4 bits)
Page Index Level 1 (8 bits)
Page Index Level 2 (11 bits)
Page Offset (13 bits)
Virtual Page Number (VPN)
+
L1 Page Directory
0
PTE1
255
+
L2 Page Table
0
PTE2
2047
Page Table Base Address depending on current CID
34
13
12
0
Physical Page Number (22 bits)
Page Offset (13 bits)

Figure 10. 32-bit Address Translation Mechanism using Two-Level Page Table

In OR1200 case, page tables must be managed by operating system's virtual memory management subsystem. Figure 10 shows address translation using two-level page table. Refer to *OpenRISC 1000 System Architecture Manual* for one-level page table address translation as well as for details about address translation and page table content.

DMMUCR and Flush of Entire DTLB

DMMUCR is not implemented in OR1200. Therefore page table base pointer (PTBP) must be stored in software variable. Flush of entire DTLB must be performed by software flush of every DTLB entry separately. Software flush is performed by manually writing accessed (A) and dirty (D) bits from the TLB entries back to PTEs.

Page Protection

After a virtual address is determined to be within a page covered by the valid PTE, the access is validated by the memory protection mechanism. If this protection mechanism prohibits the access, a data page fault exception is generated.

The memory protection mechanism allows selectively granting read access and write access for both supervisor and user modes. The page protection mechanism provides protection at all page level granularities.

Protection attribute	Meaning
DMMUPR[SREx]	Enable load operations in supervisor mode to the page.
DMMUPR[SWEx]	Enable store operations in supervisor mode to the page.
DMMUPR[UREx]	Enable load operations in user mode to the page.
DMMUPR[UWEx]	Enable store operations in user mode to the page.

Table 10. Protection Attributes for Load/Store Accesses

Table 10 lists page protection attributes defined in DMMUPR protection register. For the individual page appropriate strategy out of seven possible strategies programmed in DMMU protection register is selected with the PPI field of the PTE.

DTLB Entry Reload

OR1200 does not implement DTLB entry reloads in hardware. Instead software routine must be used to search page table for correct page table entry (PTE) and copy it into the DTLB. Software is responsible for maintaining accessed and dirty bits in the page tables.

When LSU computes load/store effective address whose physical address is not already cached by DTLB, a DTLB miss exception is invoked.

DTLB reload routine must load the correct PTE to correct DTLBMR and DTLBTR register from one of possible DTLB ways.

DTLB Entry Invalidation

Special-purpose register DTLBEIR must be written with the effective address and corresponding DTLB entry will be invalidated in the local DTLB.

Locking DTLB Entries

Since all DTLB entry reloads are performed in software, there is no hardware locking of DTLB entries. Instead it is up to the software reload routine to avoid replacing some of the entries if so desired.

Page Attribute – Dirty (D)

Dirty (D) attribute resides both in the PTE in page table and in the copy of PTE in the TLB. Every time when page is modified by a store operation, dirty bit is set.

Software must write back the dirty bit from the TLB to the page table.

In cases when access operation to the page fails, it is not defined whether dirty bit should be set or not. Since dirty bit is merely a hint to the operating system, it is up to the operating system to decide.

It is up to the operating system to determine when to explicitly clear the dirty bit for a given page.

Page Attribute – Accessed (A)

Accessed (A) attribute resides both in the PTE in page table and in the copy of PTE in the TLB. Every time when page is accessed by a load or store operation, accessed bit is set.

Software must write back the accessed bit from the TLB to the page table.

In cases when access operation to the page fails, it is not defined whether accessed bit should be set or not. Since accessed bit is merely a hint to the operating system, it is up to the operating system to decide.

It is up to the operating system to determine when to explicitly clear the accessed bit for a given page.

Page Attribute – Weakly Ordered Memory (WOM)

Weakly ordered memory (WOM) attribute is not needed in OR1200 because all memory accesses are serialized and therefore this attribute is not implemented.

Page Attribute – Write-Back Cache (WBC)

Write-back cache (WBC) attribute is not needed in OR1200 because data cache operates only in write-through mode and therefore this attribute is not implemented.

Page Attribute – Caching-Inhibited (CI)

Caching-inhibited (CI) attribute is set by the operating system for all pages that need to be accessed directly without caching their content. This is usually the case when I/O registers are memory mapped and all reads and writes must be always performed directly to the external interface and not to the data cache.

Page Attribute – Cache Coherency (CC)

Cache coherency (CC) attribute is not needed in OR1200 because it doesn't implement support for multiprocessor environments and because data cache operates only in write-through mode and therefore this attribute is not implemented.

Instruction MMU

Translation Disabled

Instruction fetch address translation can be disabled by clearing bit SR[IME]. If translation is disabled, then physical address used to access instruction cache and optionally provided on **iwb_ADDR_O**, is the same as instruction fetch effective address.

Translation Enabled

Instruction fetch address translation can be enabled by setting bit SR[IME]. If translation is enabled, it provides instruction fetch effective address to physical address translation, page protection and page attributes for instruction fetch accesses.

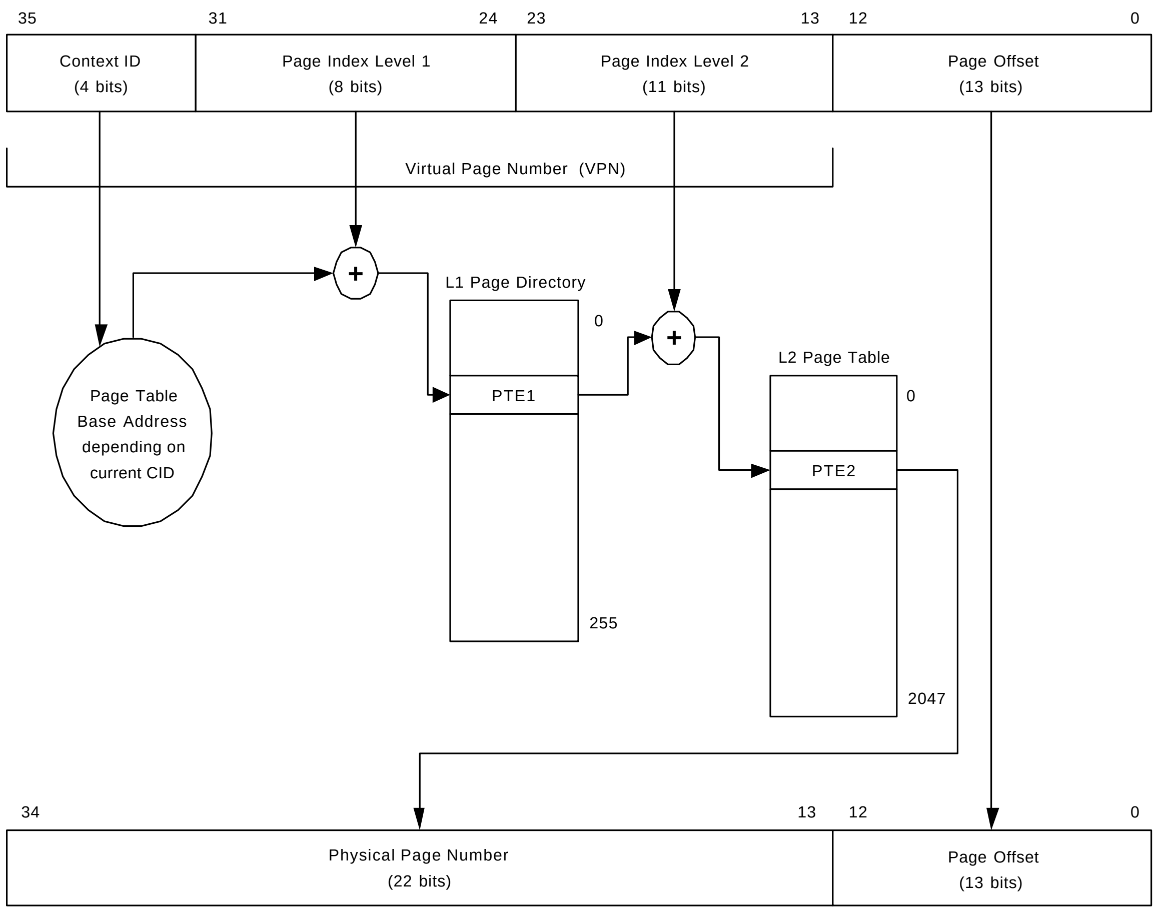

35
31
24
23
13
12
0
Context ID (4 bits)
Page Index Level 1 (8 bits)
Page Index Level 2 (11 bits)
Page Offset (13 bits)
Virtual Page Number (VPN)
+
L1 Page Directory
0
PTE1
255
+
L2 Page Table
0
PTE2
2047
Page Table Base Address depending on current CID
34
13
12
0
Physical Page Number (22 bits)
Page Offset (13 bits)

Figure 11. 32-bit Address Translation Mechanism using Two-Level Page Table

In OR1200 case, page tables must be managed by operating system's virtual memory management subsystem. Figure 10 shows address translation using two-level page table. Refer to *OpenRISC 1000 System Architecture Manual* for one-level page table address translation as well as for details about address translation and page table content.

IMMUCR and Flush of Entire ITLB

IMMUCR is not implemented in OR1200. Therefore page table base pointer (PTBP) must be stored in software variable. Flush of entire ITLB must be performed by software flush of every ITLB entry separately. Software flush is performed by manually writing accessed (A) bit from the TLB entries back to PTEs.

Page Protection

After a virtual address is determined to be within a page covered by the valid PTE, the access is validated by the memory protection mechanism. If this protection mechanism prohibits the access, an instruction page fault exception is generated.

The memory protection mechanism allows selectively granting execute access for both supervisor and user modes. The page protection mechanism provides protection at all page level granularities.

Protection attribute	Meaning
IMMUPR[SXEx]	Enable execute operations in supervisor mode of the page.
IMMUPR[UXEx]	Enable execute operations in user mode of the page.

Table 11. Protection Attributes for Instruction Fetch Accesses

Table 10 lists page protection attributes defined in IMMUPR protection register. For the individual page appropriate strategy out of seven possible strategies programmed in IMMU protection register is selected with the PPI field of the PTE.

ITLB Entry Reload

OR1200 does not implement ITLB entry reloads in hardware. Instead software routine must be used to search page table for correct page table entry (PTE) and copy it into the ITLB. Software is responsible for maintaining accessed bit in the page tables.

When LSU computes instruction fetch effective address whose physical address is not already cached by ITLB, an ITLB miss exception is invoked.

ITLB reload routine must load the correct PTE to correct ITLBMR and ITLBTR register from one of possible ITLB ways.

ITLB Entry Invalidation

Special-purpose register ITLBEIR must be written with the effective address and corresponding ITLB entry will be invalidated in the local ITLB.

Locking ITLB Entries

Since all ITLB entry reloads are performed in software, there is no hardware locking of ITLB entries. Instead it is up to the software reload routine to avoid replacing some of the entries if so desired.

Page Attribute – Dirty (D)

Dirty (D) attribute resides in the PTE but it is not used by the IMMU.

Page Attribute – Accessed (A)

Accessed (A) attribute resides both in the PTE in page table and in the copy of PTE in the TLB. Every time when page is accessed by a load or store operation, accessed bit is set.

Software must write back the accessed bit from the TLB to the page table.

In cases when access operation to the page fails, it is not defined whether accessed bit should be set or not. Since accessed bit is merely a hint to the operating system, it is up to the operating system to decide.

It is up to the operating system to determine when to explicitly clear the accessed bit for a given page.

Page Attribute – Weakly Ordered Memory (WOM)

Weakly ordered memory (WOM) attribute is not needed in OR1200 because all instruction fetch accesses are serialized and therefore this attribute is not implemented.

Page Attribute – Write-Back Cache (WBC)

Write-back cache (WBC) attribute resides in the PTE but it is not used by the IMMU.

Page Attribute – Caching-Inhibited (CI)

Caching-inhibited (CI) attribute is set by the operating system for all pages that need to be accessed directly without caching their content.

Page Attribute – Cache Coherency (CC)

Cache coherency (CC) attribute resides in the PTE but it is not used by the IMMU.

Programmable Interrupt Controller

PICMR special-purpose register is used to mask or unmask up to 30 programmable interrupt sources. PICPR special-purpose register is used to assign low or high priority to maximum of 30 interrupt sources.

PICSR special-purpose register is used to determine status of each interrupt input. Bits in PICSR represent status of the interrupt inputs and the actual interrupt must be cleared in the device that is the source of a pending interrupt.

Tick Timer

Tick timer facility is enabled with TTCR[TTE]. TTIR is incremented with each clock cycle and a high priority interrupt can be asserted whenever lower 28 bits of TTIR match TTCR[TP] and TTCR[IE] is set.

TTIR restarts counting from zero when match event happens and TTCR[SR] is cleared. If TTCR[SR] is set, TTIR is halted when match event happens and TTIR must be cleared (writing 1 to TTCR[SR]) to start counting again from zero.

Power Management

Clock Gating and Frequency Changing Versus CPU Stalling

If system doesn't support clock gating and if changing clock frequency in slow down mode is not possible, CPU can be stalled for certain number of clock cycles. This is much lower benefit on power consumption however it still reduces power consumption.

Slow Down Mode

Slow down mode is software controlled with the 4-bit value in PMR[SDF]. Lower value specifies higher expected performance from the processor core. Usually PMR[SDF] is dynamically set by the operating system's idle routine, that monitors the usage of the processor core.

PMR[SDF] is broadcasted on **pm_clksd**. External clock generator should adjust clock frequency according to the value of **pm_clksd**. Exact slow down factors are not defined but 0xF should go all the way down to 32.768 KHz.

With **pm_clksd** equal to 0xF, **pm_lvolt** is asserted. This is an indication for the external power supply to lower the voltage.

Doze Mode

To switch to doze mode, software should set the PMR[DME]. Once an interrupt is received by the programmable interrupt controller (PIC), **pm_wakeup** is asserted and external clock generation circuitry should enable all clocks. Once clocks are running RISC is switched back again to the normal mode and PMR[DME] is cleared.

When doze mode is enabled, **pm_dc_gate**, **pm_ic_gate**, **pm_dmmu_gate**, **pm_immu_gate** and **pm_cpugate** are asserted. As a result all clocks except **clk_tt** should be gated by external clock generation circuitry.

Sleep Mode

To switch to sleep mode, software should set the PMR[SME]. Once an interrupt is received by the programmable interrupt controller (PIC), **pm_wakeup** is asserted and external clock generation should enable all clocks. Once clocks are running, RISC is switched back again to the normal mode and PMR[SME] is cleared.

When sleep mode is enabled, **pm_dc_gate**, **pm_ic_gate**, **pm_dmmu_gate**, **pm_immu_gate**, **pm_cpu_gate** and **pm_tt_gate** are asserted. As a result all clocks including **clk_tt** should be gated by external clock generation circuitry.

In sleep mode, **pm_lvolt** is asserted. This is an indication for the external power supply to lower the voltage.

Clock Gating

To enable clock gating feature, software should set the PMR[DCGE]. This feature operates gate signals independently of doze and sleep mode.

When a particular RISC unit is not used, its clock is dynamically gated. Clock gate signals dynamically managed in this way are **pm_dc_gate**, **pm_ic_gate**, **pm_dmmu_gate** and **pm_immu_gate**.

Disabled Units Forcs Clock Gating

Units that are disabled in special-purpose register SR, have their clock gate signals asserted. Cleared bits SR[DCE], SR[ICE], SR[DME] and SR[IME] directly force assertion of **pm_dc_gate**, **pm_ic_gate**, **pm_dmmu_gate** and **pm_immu_gate**.

Debug Unit

Debug unit can be controlled through development interface or it can operate independently programmed and handled by RISC resident debug software.

Watchpoints

DVR/DCR pairs are used to compare instruction fetch or load/store EA and load/store data to the value stored in DVRs. Matches can be combined into more complex matches and transformed into watchpoints by programming DMR1 register.

By programming DMR2 register, watchpoints can be further processed by counting them and reporting them as a breakpoint exception. DWCR registers controls when watchpoint counters match predefined value in order to generate counter watchpoints.

Active watchpoints are broadcasted on development interface on **dbg_wp_o**.

Breakpoint Exception

DMR2[WGB] bits specify which watchpoints invoke breakpoint exception. By invoking breakpoint exception, target resident debugger can be built.

Breakpoint is broadcasted on development interface on **dbg_bp_o**.

Development Interface

An additional *development and debug interface IP core* may be used to connect OpenRISC 1200 to standard debuggers using IEEE.1149.1 (JTAG) protocol.

Debugging Through Development Interface

The DSR special-purpose register specifies which exceptions cause the core to stop the execution of the exception handler and turn over control to development interface. It can be programmed by the resident debug software or by the development interface.

The DRR special-purpose register is specifies which event caused the core to stop the execution of program flow and turned over control to the development interface. It should be cleared by the resident debug software or by the development interface.

The DIR special-purpose register holds the next instruction to be executed by the core. It can be programmed only by the development interface. Instruction in the DIR is executed only once and the register must be set with the new instruction to increment the program flow.

Reading PC, Load/Store EA, Load Data, Store Data, Instruction

Crucial information like program counter (PC), load/store effective address (LSEA), load data, store data and current instruction in execution pipeline can be asynchronously read through the development interface.

dbg_op_i[2:0]	Meaning
0x0	Reading Program Counter (PC)
0x1	Reading Load/Store Effective Address
0x2	Reading Load Data
0x3	Reading Store Data
0x4	Reading SPR
0x5	Writing SPR
0x6	Reading Instruction in Execution Pipeline
0x7	Reserved

Table 12. Development Interface Operation Commands

Table 12 lists operation commands that control what is read or written through development interface. All reads except reads and writes of SPRs are asynchronous.

Reading and Writing SPRs Through Development Interface

For reads and write to SPRs **dbg_op_i** must be set to 0x4 and 0x5, respectively.

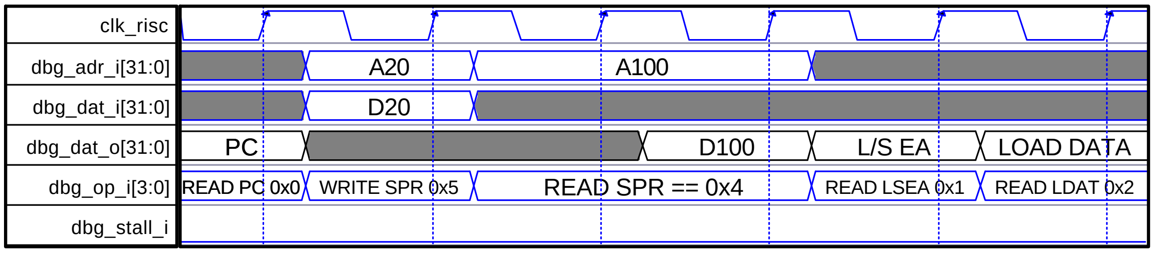


Figure 12. Development Interface Cycles

Figure 12 shows development interface cycles. Writes must be synchronous to the main RISC clock positive edge and should take one clock cycle. Reads must take two clock cycles because access to synchronous cache lines or to TLB entries introduces one clock cycle of delay.

If required, external debugger can stop the CPU core by asserting **dbg_stall_i**. This way it can have enough time to read all interesting registers from the RISC or guarantee that writes into SPRs are performed without RISC writing to the same registers.

Tracking Data Flow

An external debugger can monitor and record data flow inside the RISC for debugging purposes and profiling analysis. This is accomplished by monitoring status of the load/store unit, load/store effective address and load/store data, all available at the development interface.

dbg_lss_o[3:0]	Load/Store Instruction in Execution
0x0	No load/store instruction in execution
0x1	*Reserved for load doubleword*
0x2	Load byte and zero extend
0x3	Load byte and sign extend
0x4	Load halfword and zero extend
0x5	Load halfword and sign extend
0x6	Load singleword and zero extend
0x7	Load singleword and sign extend
0x8	*Reserved for store doubleword*
0x9	*Reserved*
0xA	Store byte
0xB	*Reserved*
0xC	Store halfword
0xD	*Reserved*

0xE	Store singleword
0xF	*Reserved*

Table 13. Status of the Load/Store Unit

External trace buffer can capture all interesting data flow events by analyzing status of the load/store unit available on **dbg_lss_o**. Table 13 lists different status encoding for the load/store unit.

Tracking Program Flow

An external debugger can monitor and record program flow inside the RISC for debugging purposes and profiling analysis. This is accomplished by monitoring status of the instruction unit, PC and fetched instruction word, all available at the development interface.

dbg_is_o[1:0]	Instruction Fetch Status
0x0	No instruction fetch in progress
0x1	Normal instruction fetch
0x2	Executing branch instruction
0x3	Fetching instruction in delay slot

Table 14. Status of the Instruction Unit

External trace buffer can capture all interesting program flow events by analyzing status of the instruction unit available on **dbg_is_o**. Table 14 lists different status encoding for the instruction unit.

Triggering External Watcpoint Event

Figure 13 shows how development interface can assert **dbg_ewt_I** and cause watchpoint event. If programmed, external watchpoint event will cause a breakpoint exception.

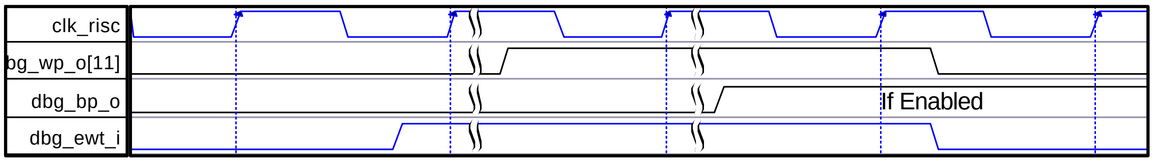


Figure 13. Assertion of External Watchpoint Trigger

4

Registers

This section describes all registers inside the OR1200 core. Shifting *GRP* number 27 bits left and adding *REG* number computes the address of each special-purpose register. All registers are 32 bits wide from software perspective. *USER MODE* and *SUPV MODE* specify the valid access types for each register in user mode and supervisor mode of operation. R/W stands for read and write access and R stands for read only access.

Registers list

GRP #	REG #	REG NAME	USER MODE	SUPV MODE	DESCRIPTION
-	-	GPR0-GPR31	R/W	R/W	General-Purpose Registers
0	1	VR	-	Read Only	Version Register
0	2	UPR	-	Read Only	Unit Present Register
0	3	SR	-	R/W	Supervision Register
0	16	EPCR0	-	R/W	Exception PC Register
0	48	EEAR0	-	R/W	Exception EA Register
0	64	ESR0	-	R/W	Exception SR Register
0	80	CCR0	R/W	R/W	Condition Code Registers
1	0-255	DTLBMR0-DTLBMR255	-	Write Only	Data TLB Match Registers
1	256-511	DTLBTR0-DTLBTR255	-	Write Only	Data TLB Translate Registers
1	512	DMMUCR1	-	R/W	Data MMU Control Register 1
1	513	DMMUCR2	-	R/W	Data MMU Control Register 2
2	0-255	ITLBMR0-ITLBMR255	-	Write Only	Instruction TLB Match Registers
2	256-511	ITLBTR0-ITLBTR255	-	Write Only	Instruction TLB Translate Registers
2	512	IMMUCR1	-	R/W	Instruction MMU Control Register 1
2	513	IMMUCR2	-	R/W	Instruction MMU Control Register 2
3	0	DCCR	–	R/W	DC Control Register
3	2	DCBFR	W	W	DC Line Flush Register

3	3	DCBIR	–	W	DC Line Invalidate Register
4	0	ICCR	–	R/W	IC Control Register
4	3	ICBIR	W	W	IC Line Invalidate Register
5	0	MACLO	R/W	R/W	MAC Low
5	1	MACHI	R/W	R/W	MAC High
6	0-3	DVR0-DVR3	-	R/W	Debug Value Registers
6	8-11	DCR0-DCR3	-	R/W	Debug Control Registers
6	16	DMR1	-	R/W	Debug Mode Register 1
6	17	DMR2	-	R/W	Debug Mode Register 2
6	18-19	DCWR0-DCWR1	-	R/W	Debug Watchpoint Counter Registers
6	20	DSR	-	R/W	Debug Stop Register
6	21	DRR	-	R/W	Debug Reason Register
6	22	DIR	-	R/W	Debug Instruction Register
8	0	PMR	-	R/W	Power Management Register
9	0	PICMR	-	R/W	PIC Mask Register
9	1	PICPR	-	R/W	PIC Priority Register
9	2	PICSR	-	R/W	PIC Status Register
10	0	TTCR	-	R/W	Tick Timer Control Register
10	1	TTIR	R/W	R/W	Tick Timer Incrementing Register

Table 15. List of All Registers

Table 15 lists all OpenRISC 1000 special-purpose registers implemented in OR1200. Registers VR and UPR are described below. For description of other registers refer to *OpenRISC 1000 System Architecture Manual* document.

Register VR description

Special-purpose register VR identifies the version (model) and revision level of the OpenRISC 1000 processor. It also specifies possible standard template on which this implementation is based.

Bit #	Access	Reset	Description
5:0	R	Revision	REV Revision number
15:6	R	0x0	Reserved
31:16	R	0x12xx	VER Version number for OR1200 is fixed at 0x1200.

Table 16. VR Register

Register UPR description

Special-purpose register UPR identifies the units present in the processor. It has a bit for each implemented unit or functionality. Lower sixteen bits identify present units defined in the OpenRISC 1000 architecture. Upper sixteen bits define present custom units.

Bit #	Access	Reset	Description
0	R	1	UP UPR present
1	R	1	DCP Data cache present
2	R	1	ICP Instruction cache present
3	R	1	DMP Data MMU present
4	R	1	IMP Instruction MMU present
5	R	1	OB32P ORBIS32 present
6	R	0	OB64P ORBIS64 not present
7	R	0	OF32P ORFPX32 not present
8	R	0	OF64P ORFPX64 not present
9	R	0	OV32P ORVDX32 not present
10	R	0	OV64P ORVDX64 not present
11	R	0	PCUP Performance counters unit not present
12	R	1	DUP Debug unit present
13	R	1	PICP Programmable interrupt controller present
14	R	1	TTP Tick timer present
15	R	0	SRP Shadowed registers not present
31:16	R	0xXXXX	CUP The user of the OR1200 core adds custom units.

Table 17. UPR Register

5 IO ports

OR1200 IP core has several interfaces. Figure 14 below shows all interfaces:

- Instruction and data WISHBONE host interfaces
- Power management interface
- Development interface
- Interrupts interface

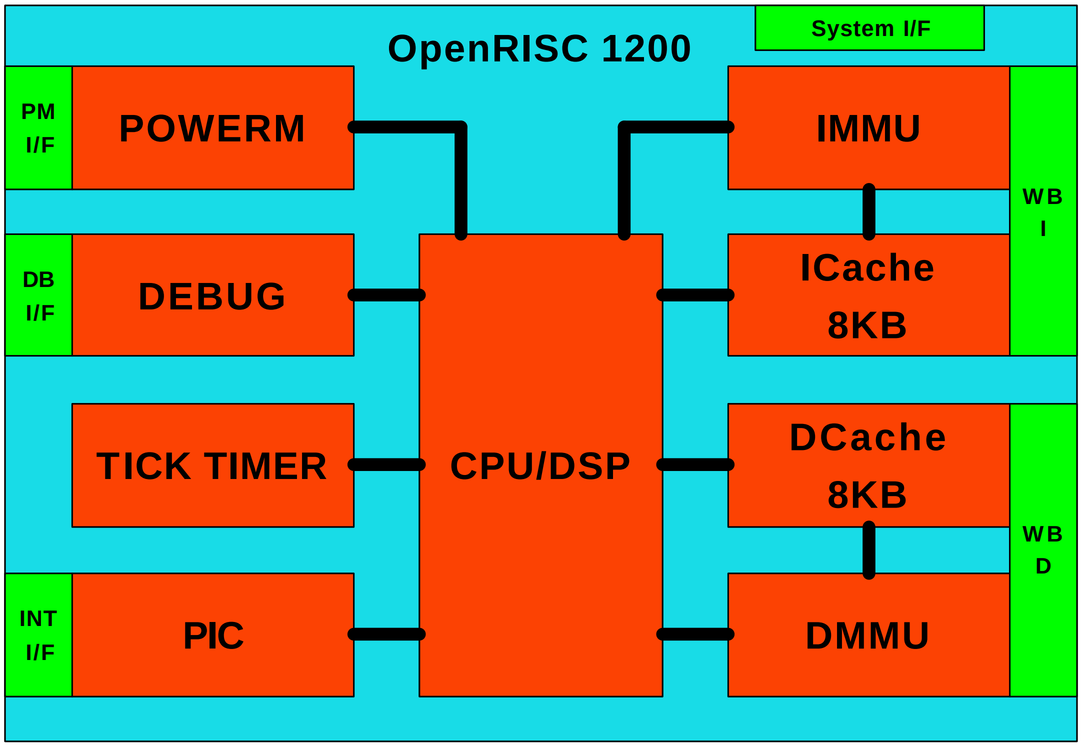


Figure 14. Core's Interfaces

Instruction WISHBONE Master Interface

OR1200 has two master WISHBONE Rev B compliant interfaces. Instruction interface is used to connect OR1200 core to memory subsystem for purpose of fetching instructions or instruction cache lines.

Port	Width	Direction	Description
iwb_CLK_I	1	Input	Clock input
iwb_RST_I	1	Input	Reset input
iwb_CYC_O	1	Output	Indicates valid bus cycle (core select)
iwb_ADR_O	32	Outputs	Address outputs
iwb_DAT_I	32	Inputs	Data inputs
iwb_DAT_O	32	Outputs	Data outputs
iwb_SEL_O	4	Outputs	Indicates valid bytes on data bus (during valid cycle it must be 0xf)
iwb_ACK_I	1	Input	Acknowledgment input (indicates normal transaction termination)
iwb_ERR_I	1	Input	Error acknowledgment input (indicates an abnormal transaction termination)
iwb_RTY_I	1	Input	In OR1200 treated same way as iwb_ERR_I.
iwb_WE_O	1	Output	Write transaction when asserted high
iwb_STB_O	1	Outputs	Indicates valid data transfer cycle

Table 18. Instruction WISHBONE Master Interface' Signals

Data WISHBONE Master Interface

OR1200 has two master WISHBONE Rev B compliant interfaces. Data interface is used to connect OR1200 core to external peripherals and memory subsystem for purpose of reading and writing data or data cache lines.

Port	Width	Direction	Description
dwb_CLK_I	1	Input	Clock input
dwb_RST_I	1	Input	Reset input
dwb_CYC_O	1	Output	Indicates valid bus cycle (core select)
dwb_ADR_O	32	Outputs	Address outputs
dwb_DAT_I	32	Inputs	Data inputs
dwb_DAT_O	32	Outputs	Data outputs
dwb_SEL_O	4	Outputs	Indicates valid bytes on data bus (during valid cycle it must be 0xf)
dwb_ACK_I	1	Input	Acknowledgment input (indicates normal transaction termination)
dwb_ERR_I	1	Input	Error acknowledgment input (indicates an abnormal transaction termination)
dwb_RTY_I	1	Input	In OR1200 treated same way as dwb_ERR_I.
dwb_WE_O	1	Output	Write transaction when asserted high
dwb_STB_O	1	Outputs	Indicates valid data transfer cycle

Table 19. Data WISHBONE Master Interface' Signals

System Interface

System interface connects reset, clock and other system signals to the OR1200 core.

Port	Width	Direction	Description
rst	1	Input	Asynchronous reset
clk_cpu	1	Input	Main clock input to the RISC
clk_dc	1	Input	Data cache clock
clk_ic	1	Input	Instruction cache clock
clk_dmmu	1	Input	Data MMU clock
clk_immu	1	Input	Instruction MMU clock
clk_tt	1	Input	Tick timer clock

Table 20. System Interface Signals

Development Interface

Development interface connects external development port to the RISC's internal debug facility. Debug facility allows control over program execution inside RISC, setting of breakpoints and watchpoints, and tracing of instruction and data flows.

Port	Width	Direction	Description
dbg_dat_o	32	Output	Transfer of data from RISC to external development interface
dbg_dat_i	32	Input	Transfer of data from external development interface to RISC
dbg_adr_i	32	Input	Address of special-purpose register to be read or written
dbg_op_i	3	Input	Operation select for development interface
dbg_lss_o	4	Output	Status of load/store unit
dbg_is_o	2	Output	Status of instruction fetch unit
dbg_wp_o	11	Output	Status of watchpoints
dbg_bp_o	1	Output	Status of the breakpoint
dbg_stall_i	1	Input	Stalls RISC CPU core
dbg_ewt_i	1	Input	External watchpoint trigger

Table 21. Development Interface

Power Management Interface

Power management interface provides signals for interfacing RISC core with external power management circuitry. External power management circuitry is required to implement functions that are technology specific and cannot be implemented inside OR1200 core.

Port	Width	Direction	Generation	Description
pm_clksd	4	Output	Static (in SW)	Slow down outputs that control

				reduction of RISC clock frequency
pm_cpustall	1	Input	-	Synchronous stall of the RISC's CPU core
pm_dc_gate	1	Output	Dynamic (in HW)	Gating of data cache clock
pm_ic_gate	1	Output	Dynamic (in HW)	Gating of instruction cache clock
pm_dmmu_gate	1	Output	Dynamic (in HW)	Gating of data MMU clock
pm_immu_gate	1	Output	Dynamic (in HW)	Gating of instruction MMU clock
pm_tt_gate	1	Output	Dynamic (in HW)	Gating of tick timer clock
pm_cpu_gate	1	Output	Static (in SW)	Gating of main CPU clock
pm_wakeup	1	Output	Dynamic (in HW)	Activate all clocks
pm_lvolt	1	Output	Static (in SW)	Lower voltage

Table 22. Power Management Interface

Interrupt Interface

Interrupt interface has interrupt inputs for interfacing external peripheral's interrupt outputs to the RISC core. All interrupt inputs are evaluated on positive edge of main RISC clock.

Port	Width	Direction	Description
pic_ints	PIC_INTS	Input	External interrupts

Table 23. Interrupt Interface

A

Core HW Configuration

This section describes parameters that are set by the user of the core and define configuration of the core. Parameters must be set by the user before actual use of the core in simulation or synthesis.

Variable Name	Range	Default	Description
EADDR_WIDTH	32	32	Effective address width
VADDR_WIDTH	32	32	Virtual address width
PADDR_WIDTH	24 – 36	32	Physical address width
DATA_WIDTH	32	32	Data width / Operation width
DC_IMPL	0 – 1	1	Data cache implementation
DC_SETS	4 – 512	256	Data cache number of sets
DC_WAYS	1 – 4	2	Data cache number of ways
DC_LINE	16	16	Data cache line size
IC_IMPL	0 – 1	1	Instruction cache implementation
IC_SETS	4 – 512	256	Instruction cache number of sets
IC_WAYS	1 – 4	2	Instruction cache number of ways
IC_LINE	16	16	Instruction cache line size in bytes
DMMU_IMPL	0 – 1	1	Data MMU implementation
DTLB_SETS	2 - 256	64	Data TLB number of sets
DTLB_WAYS	1 – 4	2	Data TLB number of ways
IMMU_IMPL	0 – 1	1	Instruction MMU implementation
ITLB_SETS	2 - 256	64	Instruction TLB number of sets
ITLB_WAYS	1 – 4	2	Instruction TLB number of ways
PIC_INTS	2 – 32	30	Number of interrupt inputs