OpenRISC 1000

Native Bus Interface Manual

18/Mar/2000, DRAFT

1 About this Manual

1.1 Brief Introduction

OpenRISC 1000 native bus interface manual describes native OpenRISC 1000 bus interface and its operation. It presents signals of OpenRISC 1000 architecture compliant processors and how they perform bus transactions on native OpenRISC 1000 bus.

1.2 Authors

If you have contributed to this manual and your name isn't listed here, it is not meant as a slight. We just don't know about it. Send email to the maintainer(s), and we'll correct the situation.

NAME	E-MAIL	CONTRIBUTION
Damjan Lampret	lampret@opencores.org	Initial document
Jimmy C. Chen	jimmy@ee.nctu.edu.tw	NBI and timing updated

1.3 Revision History

REVISION DATE	BY	MODIFICATIONS
17/Mar/2000	Damjan Lampret	Initial document
15/Jun/2000	Jimmy C. Chen	NBI and timing updated

1.4 Work in Progress

This document is *work in progress*. Latest version is always available from OPENCORES CVS. See details how to get it on http://www.opencores.org/.

We are currently looking for people working on this document and for a maintainer of this document. If you would like to contribute send an email to one of the authors.

1.5 Fonts in this manual

In this manual, fonts are used as follows:

- **Bold** font is used for emphasis
- `UPPER CASE` items are signal names
- Square brackets [ ] indicate portion of a bus with a signal name or a number representing a signal or with two numbers separated by a semicolon representing a group of signal

2 OpenRISC 1000 Bus Interfaces

2.1 Introduction

OpenRISC 1000 architecture does not define any specific bus to be used in OpenRISC 1000 compliant processors. It allows any standard or proprietary bus interface including native OpenRISC 1000 bus interface. Figure 2-1 shows how OpenRISC 1000 architecture is bus independent.

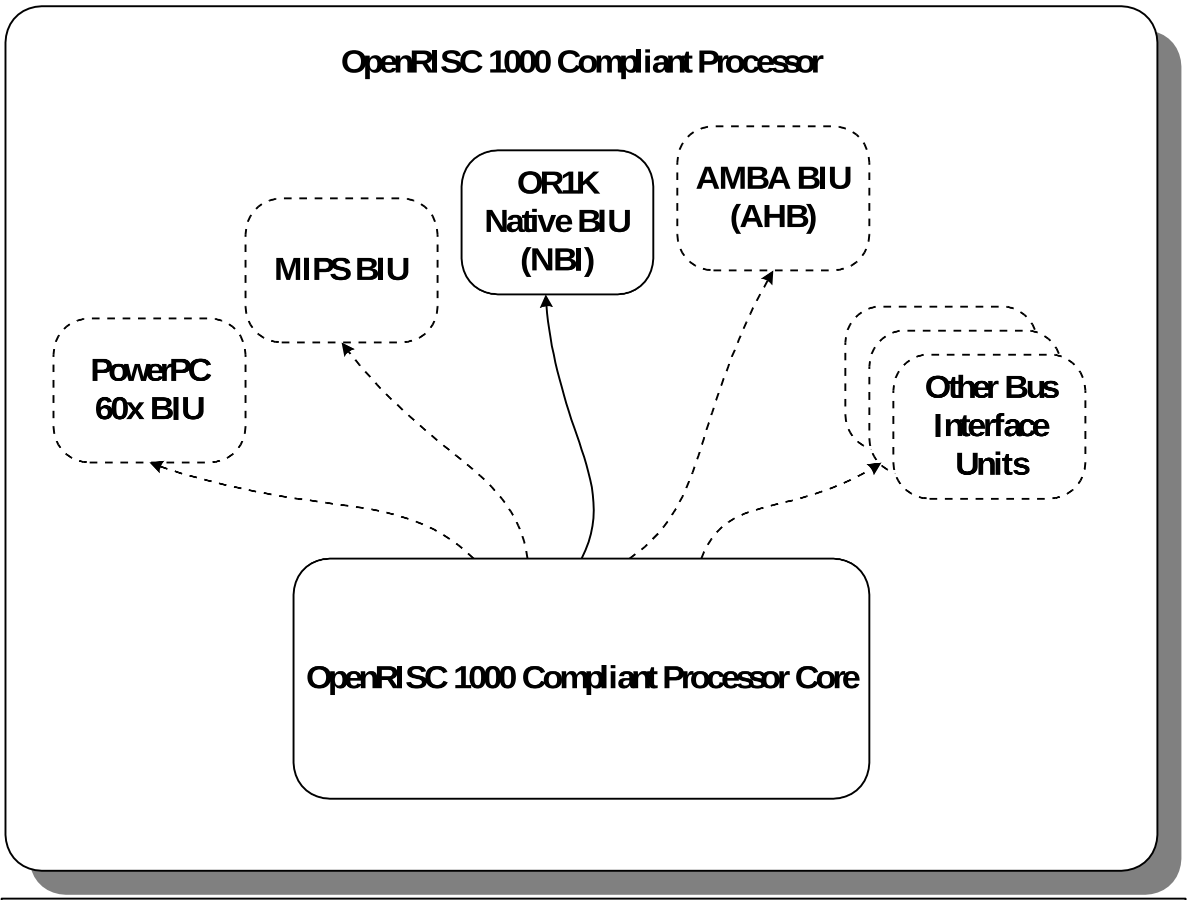

OpenRISC 1000 Compliant Processor
OR1K Native BIU (NBI)
AMBA BIU (AHB)
MIPS BIU
PowerPC 60x BIU
Other Bus Interface Units
OpenRISC 1000 Compliant Processor Core

Figure 2-1. OpenRISC 1000 architecture is bus indepedent

At the moment only native bus interface is under development.

TODO:

- add more about features required from a bus to be used in or1k system.

- call for developers to implement various BIUs for standard buses.

2.2 Acronyms and Abbreviations

BIU	Bus interface unit
CPU	Central processing unit
EA	Effective address
FPU	Floating-point unit
PLL	Phased-locked loop
R/W	Read/Write
RISC	Reduced instruction set computing
OR1K	OpenRISC 1000

2.3 Conventions

0x	Prefix indicates a hexadecimal number.
`~SIGNAL`	Active low signal
`SIGNAL[FIELD]`	Syntax used to identify specific signal of a bus or a group of signals. FIELD can be a name of a one or a group of signals or a decimal number or 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.

2.4 Numbering

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

3 OpenRISC 1000 Native Bus Interface

3.1 Introduction

Native bus interface is simple and easy to understand. It is synchronous, operates with clock CLK and supports burst transfers. It has a variable width data port and variable physical address space support. In addition, there are several handshake signals and variable number of interrupt inputs. The interface has a simple timing specification and is capable of transferring data between the processor and memory at a peak rate of 320MB/sec with a 100MHz bus and 32-bit data port.

The native bus interface can transfer *DATA_WIDTH*[2] data in one cycle. However, the rate at which data is transferred to/from the bus master is determined by the data transfer capability of the slave device. The slave device can exchange data with the bus master at any transfer rate.

TODO: Define correct terminology (processor vs bus master vs master agent; external device vs slave device vs external agent etc??) since OR1K native bus interface manual can be broader than just native BIU spec. Instead could define entire OR1K systems.

3.2 Specifications

We expect the following from native OpenRISC 1000 bus interface:

- Simple bus interface that allows simple bus interface implementations with low number of gates
- Easily understandable bus protocol that would speed up development of peripherals for OpenRISC 1000 processors
- Similar bus protocol to other standard buses that would allow easy migration once other standard bus interfaces become available (MIPS bus, PowerPC 60x bus, AMBA High Speed Bus)
- Small number of interface signals

[2] Set by the user at BIU synthesis time in configuration file.

3.3 Signal Descriptions

The processor outputs start to change at the rising edge of **`CLK`**. The processor input is latched at the rising edge of **`CLK`**.

Figure 3-1 illustrates signals of native bus interface of OpenRISC 1000 compatible processor.

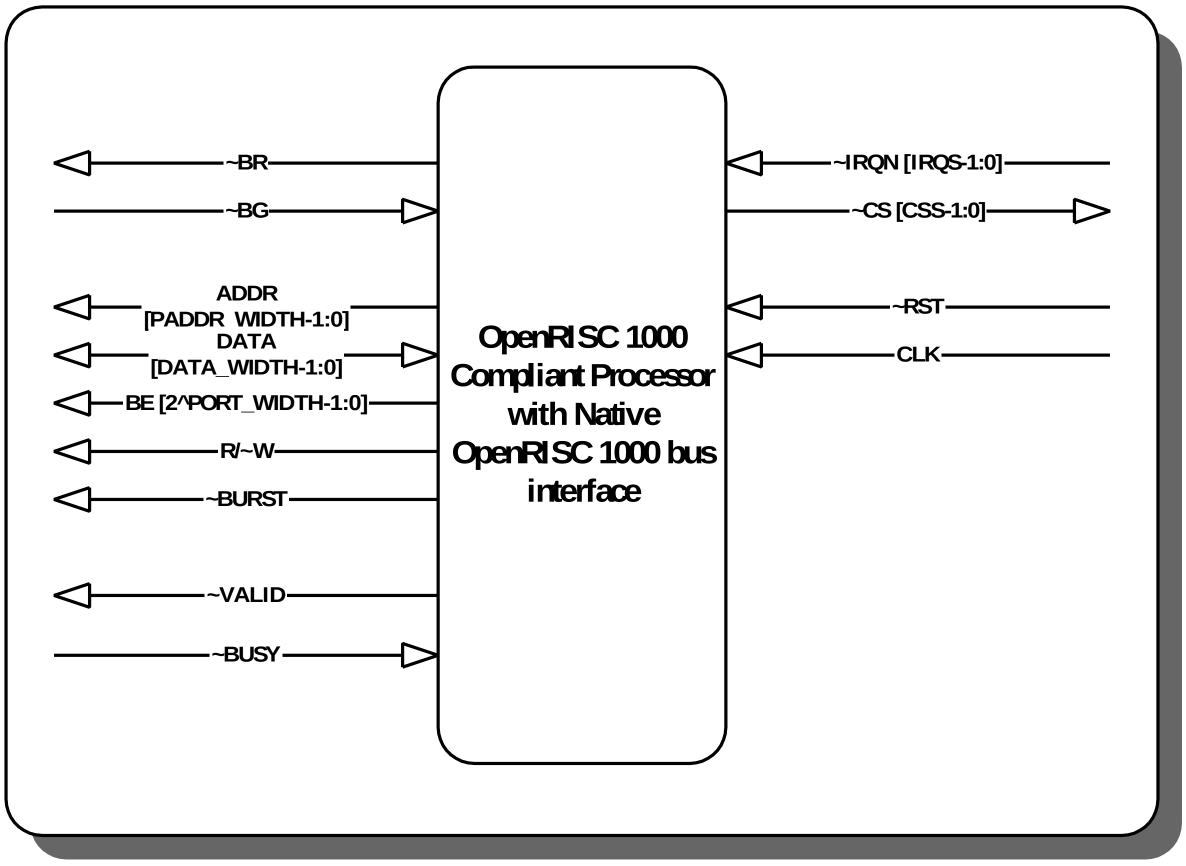


Figure 3-1. OpenRISC 1000 native bus signals

TODO: Signals need more precise definition, timing notes, state meaning etc.

3.3.1 Arbitration Signals

`~BR` is driven by the processor when it requests mastership of the bus. It may be driven for one or more clock cycles until **`~BG`** is asserted by external bus arbitration logic. Arbitration logic can also be implemented in the processor (implementation specific).

~BG is driven by external bus arbiter and indicates to the processor that can assume mastership of the bus. If there is only one master in the system then its **~BG** can be tied to ground.

3.3.2 Address Transfer Signals

ADDR bus transfer signals are used to transmit physical address of the data to be transferred. They are bi-directional signal. On burst transfers, the address bus presents word aligned address containing the critical address first. Address during burst operations is incremented by the master.

Width of address bus is set by the user at synthesis time with *PADDR_WIDTH*.

3.3.3 Data Transfer Signals

DATA bus transfer signals are used to transmit data. They are bi-directional signal.

Width of data bus is set by the user at synthesis time with *DATA_WIDTH*.

3.3.4 Transfer Attribute Signals

BEn signals are driven by the bus master to indicate to the slave device which data bus byte lines contain valid data. Number of byte enable signals is set by the user at synthesis time with *PORT_WIDTH*.

R/~W is driven by the bus master to indicate to the slave device direction of the transaction.

~BURST is driven by the bus master to indicate to the slave device burst type of transaction. Usually burst transactions are used to flush and refill cache lines. Burst transaction are similar to single-beat transaction except that they last longer.

3.3.5 Transaction Handshake Signals

~BUSY is used by an slave device to indicate to the bus master whether it can accept a new read or write transaction. Processor samples this signal before deasserting the address on read and write transactions.

~VALID is driven by the bus master to indicate to the slave device that there is a valid address on **ADDR** bus (and data on **DATA** bus in case of a write transaction).

3.3.6 Chip Select Signals (optional)

`~CS`n are driven by the processor to select slave device that is the target of the current transaction. Target device can responds to initiated transaction with delivery of data in case of read transaction or with acceptance of data in case of write transaction. If target device is not ready to process transaction it must assert **`~BUSY`** signal.

When processor is not a bus master it monitors valid transactions on the bus and generates chip select signals for other masters.

Number of chip select signals is set by the user.

3.3.7 Interrupt Signals

`~IRQ`n are asserted active by peripheral devices to request processor's intervention. If interrupt request is not masked an external interrupt exception is taken. Interrupt request is deasserted by the processor with peripheral specific action. Usually this means a write in one of the peripheral's interrupt control registers.

Interrupt request signals are processed with software interrupt priority scheduling. By default the highest priority has **`~IRQ0`** followed by **`~IRQ1`** etc. There is no special non-maskable interrupt request signal since NMI can be implemented with **`~IRQ0`** and never masking its operation.

Number of interrupt request signals is set by the user.

3.3.8 Clock Signals

OpenRISC 1000 native bus interface uses a single clock input **`CLK.`** It sets the frequency of operation for the bus interface. Internally processor uses a PLL circuit to generate master clock for all of the CPU circuitry. Internal clock can be a multiple of **`CLK`** frequency allowing the CPU to operate at an equal or greater clock than the bus interface.

3.3.9 Reset Signals

OpenRISC 1000 native bus interface uses a single reset input **`~RST`**. When **`~RST`** signal is asynchronously asserted active, all the internal states are initialized to default values (most of them are zero) and reset exception is taken after deassertation.

There is no soft reset signal.

3.4 BIU Registers

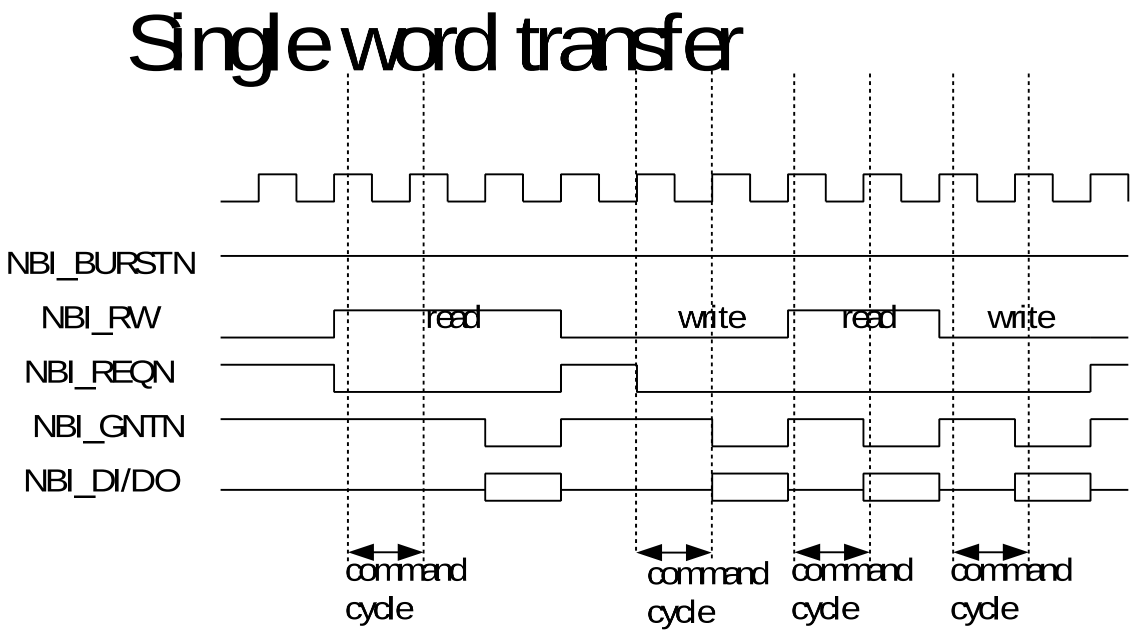
TODO: Should we have any config registers in BIU? If not how one can set address ranges for chip selects (at synthesis time?)? How about masking IRQs?

TBD

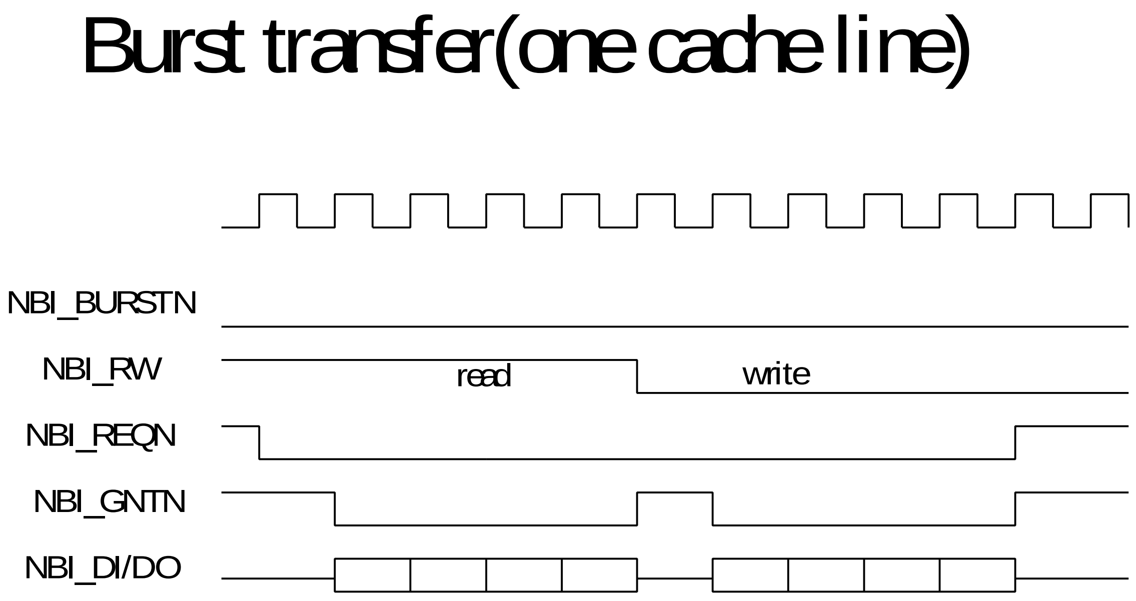
3.5 Timing Diagram

The native bus interface(NBI) is used in original OR1K chip design. Bus transaction is initiated by REQN. Read or write transaction is indicated by NBI_RW. If NBI_RW is 1, then it is a read transaction; if it is 0, then it is a write one. NBI_GNTN is a grant signal issued by external bridge or arbiter. NBI_BURSTN is the burst indicator. If NBI_BURSTN is zero, one cache line burst access will be initiated. The timing diagram is shown in 3.5.1, 3.5.2.

3.5.1 Single Read/Write Transaction



3.5.2 Burst Read/Write Transaction



3.5.3 Interrupt Request

Interrupt request signal is low active. It must remain active until interrupt service routine deactivates the device which issuing the interrupt request.